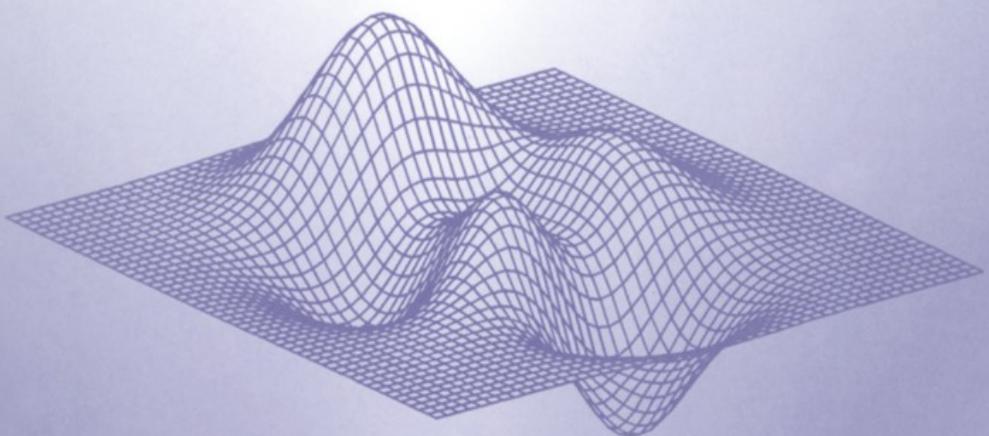


NONCONVEX OPTIMIZATION AND ITS APPLICATIONS

Practical Bilevel Optimization Algorithms and Applications

Jonathan F. Bard



Springer-Science+Business Media, B.V.

Practical Bilevel Optimization: Algorithms and Applications

Nonconvex Optimization and Its Applications

Volume 30

Managing Editors:

Panos Pardalos

University of Florida, U.S.A.

Reiner Horst

University of Trier, Germany

Advisory Board:

Ding-Zhu Du

University of Minnesota, U.S.A.

C. A. Floudas

Princeton University, U.S.A.

J. Mockus

Stanford University, U.S.A.

H. D. Sherali

Virginia Polytechnic Institute and State University, U.S.A.

The titles published in this series are listed at the end of this volume.

Practical Bilevel Optimization

Algorithms and Applications

by

Jonathan F. Bard

*Graduate Program in Operations Research,
Department of Mechanical Engineering,
The University of Texas,
Austin, Texas, U.S.A.*



SPRINGER-SCIENCE+BUSINESS MEDIA, B.V.

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN 978-1-4419-4807-6 ISBN 978-1-4757-2836-1 (eBook)
DOI 10.1007/978-1-4757-2836-1

Printed on acid-free paper

All Rights Reserved

©1998 Springer Science+Business Media Dordrecht

Originally published by Kluwer Academic Publishers in 1998

No part of the material protected by this copyright notice may be reproduced or
utilized in any form or by any means, electronic or mechanical,
including photocopying, recording or by any information storage and
retrieval system, without written permission from the copyright owner

CONTENTS

PREFACE	xi
Part I MATHEMATICAL PROGRAMMING	1
1 INTRODUCTION	3
1.1 Model Development	5
1.2 Early Work and Applications	8
1.3 Structural Considerations	10
1.3.1 Indifference Points and Nonexistence of Solutions	11
1.3.2 Significance of Order of Play	12
1.4 Scope and Outline	14
2 LINEAR PROGRAMMING	17
2.1 Introduction	17
2.1.1 Basic Solutions	22
2.1.2 Fundamental Theorem	23
2.1.3 Convex Properties	25
2.2 Simplex Method	29
2.2.1 Pivoting	30
2.2.2 Determining the Leaving Variable	32
2.2.3 Moving toward Optimality	33
2.2.4 Degeneracy and Cycling	37
2.3 Geometry of Simplex Method	41
2.3.1 Finiteness of Algorithm	42
2.3.2 Adjacency and Bounded Edges	42
2.3.3 Unboundedness	44
2.3.4 Finding Adjacent Extreme Points	47
2.3.5 Main Geometric Argument	47
2.3.6 Alternative Optima and Uniqueness	48

2.3.7	Ranking Extreme Points	49
2.4	Additional Features	51
2.4.1	Phase 1 and Artificial Variables	51
2.4.2	Bounded Variables	54
2.4.3	Kuhn-Tucker Conditions and the Linear Complementarity Problem	57
2.5	Duality	59
2.5.1	Primal–Dual Relationship	61
2.5.2	Dual Theorems	61
2.5.3	Economic Interpretation	66
2.5.4	Sensitivity Analysis	66
2.5.5	Dual Simplex Method	72
3	INTEGER PROGRAMMING	76
3.1	Introduction	76
3.1.1	Models with Integer Variables	78
3.1.2	Solving Integer Programs	83
3.2	Enumerative Methods	87
3.2.1	Definitions and Concepts	89
3.2.2	Generic Branch and Bound Algorithm	91
3.2.3	Branch and Bound Using LP Relaxation	92
3.2.4	Implementation Issues	96
3.2.5	Zero–One Implicit Enumeration	102
3.2.6	General Branching and Data Structures	106
3.3	Cutting Planes	109
3.3.1	Method of Integer Forms	110
3.3.2	Primal All-Integer Cuts	114
3.3.3	Cuts with Unit Coefficients	118
3.3.4	Valid Inequalities	121
3.4	Benders Decomposition for Mixed–Integer Linear Programming	127
3.4.1	Reformulation of MILP	127
3.4.2	Algorithm	130
3.5	Unimodularity	133
4	NONLINEAR PROGRAMMING	137
4.1	Introduction	137
4.1.1	Classification of Problems	140
4.1.2	Difficulties Resulting from Nonlinearities	142
4.1.3	Notation	143

4.2	Optimality Conditions	155
4.2.1	Unconstrained Problems	156
4.2.2	Nonnegative Variables	157
4.2.3	Equality Constraints	159
4.2.4	Inequality Constraints	164
4.2.5	Convex Optimization	167
4.3	Search Techniques for Unconstrained Problems	169
4.3.1	One-Dimensional Linear Search Techniques	170
4.3.2	Multidimensional Search Techniques	175
4.4	Algorithms For Constrained Optimization	181
4.4.1	Primal Methods	181
4.4.2	Penalty Methods	185
4.4.3	Sequential Quadratic Programming	188
Part II BILEVEL PROGRAMMING		193
5	LINEAR BLP: CONTINUOUS VARIABLES	195
5.1	Introduction	195
5.2	Theoretical Properties	198
5.3	Algorithms for the Linear Bilevel Programming Problem	202
5.3.1	K th-Best Algorithm	203
5.3.2	Kuhn-Tucker Approach	204
5.3.3	Complementarity Approach	209
5.3.4	Variable Elimination Algorithm	213
5.3.5	Penalty Function Approach	218
5.4	Computational Comparisons	222
6	LINEAR BLP: DISCRETE VARIABLES	232
6.1	Introduction	232
6.2	Properties of the Zero–One Linear BLPP	233
6.2.1	Reductions to Linear Three–Level Programs	237
6.2.2	Algorithmic Implications	244
6.3	Properties of the Mixed–Integer Linear BLPP	245
6.4	Moore–Bard Algorithm for the Mixed–Integer Linear BLPP	247
6.4.1	Branch and Bound Notation	248
6.4.2	Bounding Theorems	249
6.4.3	Algorithm	250
6.4.4	Computational Experience	254
6.4.5	Assessment	257

6.5	Algorithm for the Discrete Linear BLPP	258
6.5.1	Algorithm	259
6.5.2	Computational Experience	266
6.5.3	Assessment	267
7	CONVEX BILEVEL PROGRAMMING	269
7.1	Introduction	269
7.2	Descent Approaches for the Quadratic BLPP	272
7.2.1	An <i>EIR</i> Point Descent Algorithm	274
7.2.2	A Modified Steepest Descent Approach	276
7.2.3	Hybrid Approach and Concave Minimization	283
7.3	Branch and Bound Algorithm	284
7.4	Variable Elimination Algorithm	290
7.4.1	Convex-Quadratic BLPP	291
7.4.2	Computational Experience	296
8	GENERAL BILEVEL PROGRAMMING	301
8.1	Introduction	301
8.1.1	Independence of Irrelevant Constraints	305
8.1.2	Preview of Algorithms	309
8.2	Branch and Bound Algorithm	311
8.3	Double Penalty Function Method	320
8.4	Rectangular Partitioning	327
8.5	Steepest Descent Direction	332
8.5.1	Necessary Optimality Conditions	333
8.5.2	Overview of Descent Method	335
8.6	Subgradient Descent – Bundle Method	339
8.6.1	Preliminaries	341
8.6.2	Optimality Conditions and Stability of Local Solutions	344
8.6.3	Leader Predominate Algorithm	347
8.7	Transformation to Concave Program	352
8.8	Assessment of Algorithms	359
9	HEURISTICS	361
9.1	Introduction	361
9.2	Artificial Intelligence-Based Approaches	362
9.2.1	Overview of Genetic Algorithms	363
9.2.2	GABBA	364
9.2.3	Grid Search Technique	369

9.2.4	Comparison of GABBA and Grid Search	370
9.2.5	Simulated Annealing Algorithm (SABBA)	373
9.2.6	Comparison of SABBA and Grid Search	374
9.2.7	Assessment	375
9.3	Hybrid Tabu–Descent Algorithm	375
9.3.1	Initialization Procedure	376
9.3.2	Tabu Phase	378
9.3.3	Numerical Results	381
9.3.4	Discussion	386
Part III APPLICATIONS		389
10 TRANSPORTATION NETWORK DESIGN		391
10.1	Introduction	391
10.2	Rural Highway Network	392
10.3	Decision Variables	394
10.4	BLP Formulation	395
10.5	Objective Functions	397
10.5.1	Travel Time Functions	397
10.5.2	Operating Costs	399
10.5.3	Accident Costs	402
10.5.4	Improvement and Maintenance Costs	403
10.5.5	Additivity of Cost Functions	405
10.6	Conservation of Flow Constraints	407
10.7	Solution of Empirical Problem	410
10.8	Conclusions	412
11 PRODUCTION PLANNING		414
11.1	Introduction	414
11.2	Mathematical Developments	415
11.2.1	Formulation as a Bilevel Program	415
11.2.2	Interpretation and Technical Considerations	418
11.3	Application Associated with Electric Motor Production	419
11.3.1	Solution to Continuous BLP Model	423
11.3.2	Noncooperative Implications of the Model	424
11.3.3	Solution to Mixed–Integer BLP Model	425
11.4	Discussion of Results	426

12 DETERMINING PRICE SUPPORT LEVELS FOR BIOFUEL CROPS	428
12.1 Introduction	428
12.2 Mathematical Model	430
12.3 Description of Algorithms	434
12.3.1 Industry Model	435
12.3.2 Grid Search Algorithm (GSA)	435
12.3.3 Nonlinear Programming Approach	436
12.3.4 QP Formulation for Follower's Problem	439
12.4 Implementation	440
12.4.1 Overall System Design and Components	440
12.4.2 GAMS Model Structure Determination	443
12.4.3 Model Evaluation Subsystem	446
12.5 Computational Results	449
12.5.1 Grid Search Solutions	449
12.5.2 Output from SQP	450
12.6 Discussion	452
REFERENCES	455
INDEX	469

PREFACE

The use of optimization techniques has become integral to the design and analysis of most industrial and socio-economic systems. Great strides have been made recently in the solution of large-scale problems arising in such areas as production planning, airline scheduling, government regulation, and engineering design, to name a few. Analysts have found, however, that standard mathematical programming models are often inadequate in these situations because more than a single objective function and a single decision maker are involved. Multiple objective programming deals with the extension of optimization techniques to account for several objective functions, while game theory deals with the inter-personal dynamics surrounding conflict. Bilevel programming, the focus of this book, is in a narrow sense the combination of the two. It addresses the problem in which two decision makers, each with their individual objectives, act and react in a noncooperative, sequential manner. The actions of one affect the choices and payoffs available to the other but neither player can completely dominate the other in the traditional sense.

Over the last 20 years there has been a steady growth in research related to theory and solution methodologies for bilevel programming. This interest stems from the inherent complexity and consequent challenge of the underlying mathematics, as well as the applicability of the bilevel model to many real-world situations. The primary aim of this book is to provide a historical perspective on algorithmic development and to highlight those implementations that have proven to be the most efficient in their class. This aim, however, applies mainly to the linear version of the problem since there have been only a handful of algorithms developed for the nonlinear and discrete cases. Claims of relative efficiency would be problematic for those implementations. A corollary aim of the book is to provide a sampling of applications in order to demonstrate the versatility of the basic model and the limitations of current technology.

Prior to undertaking this project, I was involved with Professor Kiyotaka Shimizu of Keio University and Professor Yo Ishizuka of Sophia University in writing the text *Nondifferentiable and Two-Level Mathematical Programming*. As the title implies, the book treats the general area of two-level programming (as distinguished therein from bilevel programming), and contains a comprehensive discussion of nondifferentiable optimization. The final product was highly theoretical, aimed primarily at the research community. Only two chapters dealt specifically with the bilevel programming problem or what is sometimes referred to as the Stackelberg game. The need

for a greater discussion of algorithms and applications became apparent soon after publication. I felt that an unfulfilled demand still existed for a reference text that offered an integrated treatment of the field from the practitioner's point of view. This was the motivation for the sequel.

The intended audience here includes management scientists, operations researchers, industrial engineers, mathematicians and economists. Students with a background in deterministic operations research methods should be on solid ground in reading the text. Nevertheless, anyone with training in mathematics at a level found in a typical undergraduate engineering or science curricula should be able to handle most of the material. To facilitate those new to optimization, I have provided several chapters on mathematical programming. This material constitutes the first part of the book. It is written in sufficient detail to permit its use in a first-year graduate course on bilevel programming, with perhaps the first third of the course geared towards basic optimization.

In writing a text of this scope, it is impossible to thank all those who have helped or contributed in a significant way. A few individuals, though, should be singled out for special recognition. The first is Jim Falk who put me in touch with bilevel programming during my graduate days at The George Washington University, and whose basic research in nonlinear optimization has provided the foundations for many of the developments to date. Next there are Professors Shimizu and Ishizuka who were collaborators on my first bilevel text, and Jim Moore and Tom Edmunds who were two of my most prolific Ph.D. students. Their research is at the core of several of the most successful bilevel codes. I am also indebted to Gilles Savard and Luis Vicente for their many insights, and for giving me permission to use their work in several chapters. Similar thanks go to David Boyce, G. Anandalingam, Charles Macal, Patrice Marcotte, Joaquim Júdice, Jiming Liu and Leon Lasdon. In addition, there are those whose research I have cited repeatedly and who have made notable contributions to the field, including Mark Karwan, Wayne Bialis, A. Faustino, Pierre Hansen, Brigitte Jaumard, Omar Ben-Ayed and Charles Blair.

Finally, I wish to express my gratitude to the Department of Mechanical Engineering at the University of Texas where I have been a faculty member since 1984. If I had been asked to define a more supportive academic environment before coming to Texas, I am not sure what it would have been. I am deeply indebted to my colleagues on the faculty and to the graduate students with whom I have worked and taught over the years for their continued concern and support.

J. F. Bard

PART I

MATHEMATICAL PROGRAMMING

INTRODUCTION

The central focus of this book is on the development and implementation of algorithms for solving bilevel programs. The bilevel programming problem (BLPP) can be viewed as a static version of the noncooperative, two-person game introduced by Von Stackelberg [S16] in the context of unbalanced economic markets. In the basic model, control of the decision variables is partitioned amongst the players who seek to optimize their individual payoff functions. Perfect information is assumed so that both players know the objective and feasible choices available to the other. The fact that the game is said to be ‘static’ implies that each player has only one move. The *leader* goes first and attempts to minimize net costs. In so doing, he must anticipate all possible responses of his opponent, termed the *follower*. The follower observes the leader’s decision and reacts in a way that is personally optimal without regard to extramural effects. Because the set of feasible choices available to either player is interdependent, the leader’s decision affects both the follower’s payoff and allowable actions, and vice versa.

The bilevel programming model is motivated by the fact that decision making in large, hierarchical organizations rarely proceeds from a single point of view. The most prominent aspects of such organizations are specialization and coordination. The former arises from a practical need to isolate individual jobs or operations and to assign them to specialized units. This leads to departmentalization; however, to accomplish the overall objective, the specialized units must be coordinated. The related process is referred to in organization theory as *control* and divides itself quite naturally into two parts: the first is the establishment of individual goals and operating rules for each member, and the second is the enforcement of those rules within the organizational setting.

An important control variable in the theory of departmentalization is the degree of self-containment of the organization units. A unit is self-contained to the extent and degree that the conditions for carrying out its functions are independent of what is done elsewhere in the system. The corporate or higher-level unit is then faced

with the coordination problem of favorably resolving the divisional unit interactions. Mathematical programming has often been used as the basis for modeling these interactions with decomposition techniques providing solutions to problems of large scale (e.g., see [D4, G6]). The central idea underlying these techniques is very simple and can be envisioned as the following algorithmic process, say, in a manufacturing environment: top management, with its set of goals, asks each division of the company to calculate and submit an optimal production plan as though it were operating in isolation. Once the plans are submitted, they are modified with the overall objective of the company in mind. Marginal profit figures are used to successively reformulate the divisional plans at each stage in the algorithm. An output plan ultimately emerges that is optimal for the company as a whole and which therefore represents the solution to the original programming problem.

Although this procedure attempts to mimic corporate behavior it fails on two counts. The first relates to the assumption that it is possible to derive a single objective or utility function which adequately captures the goals of both top management and each subordinate division. The second stems from lack of communication among the components of the organization; at an intermediary stage of the calculations there is no guarantee that each division's plans will satisfy the corporate constraints. In particular, if the production of some output by division k imposes burdens on other divisions by using up a scarce company resource, or by causing an upward shift in the cost functions pertaining to some other company operation, division k 's calculation is likely to lead it to overproduce this item from the point of view of the company because the costs to other divisions will not enter its accounts. This is the classical problem of external diseconomies. Similarly, if one of division k 's outputs yields external economies where a rise in its production increases the profitability of other divisions, division k (just considering its own gains in its calculations) may not produce enough of this product to maximize the profits of the firm. This may result in a final solution that does not realistically reflect the production plan that probably would have been achieved had each division been given the degree of autonomy it exercises in practice.

Another way of treating the multilevel nature of the resource allocation problem is through goal programming. Ruefli [R4] was the first to apply this technique by proposing a generalized goal decomposition model. Freeland and Baker [F8] expanded on this work and developed a model capable of representing a wide range of operational characteristics, including informational autonomy, interdependent strategies, and 'bounded rationality' or individual goals. Others have used combinations of the above strategies to solve problems related to government regulation, distribution and control (e.g., see [D4]). The bilevel programming approach discussed herein assumes that decision making between levels proceeds sequentially but with some amount of independence to account for the divergence of corporate and subordinate objectives. Although we do not explicitly treat the case where there is more than one follower or division, most of the results can be extended to this situation. All that is needed is a few mild assumptions related to equilibrium analysis; that is, at the divisional

level each unit simultaneously attempts to optimize its own payoff function and, in so doing, produces a balance of opposing forces. This issue is further discussed in [B3].

1.1 MODEL DEVELOPMENT

A distinguishing characteristic of multilevel systems is that the decision maker at one level may be able to influence the behavior of a decision maker at another level but not completely control his actions. In addition, the objective functions of each unit may, in part, be determined by variables controlled by other units operating at parallel or subordinate levels. For example, policies introduced by corporate management relating to resource allocation and benefits may curtail the set of strategies available to divisional management. In turn, policies adopted at the lower levels affecting productivity and marketing may play a role in determining overall profitability and growth. Common features of multilevel organizations are:

- (a) interactive decision making units exist within a predominantly hierarchical structure;
- (b) each subordinate level executes its policies after, and in view of, decisions made at a superordinate level;
- (c) each unit independently maximizes net benefits (minimizes net costs), but is affected by the actions of other units through externalities; and
- (d) extramural effects enter a decision maker's problem through his objective function and feasible strategy set.

The need for specialization and decentralization has traditionally been met by the establishment of profit centers. In this context, divisions or departments are viewed as more or less independent units charged with the responsibility of operating in the best possible manner so as to maximize profit under the given constraints imposed by the corporate management. The problem of decentralization is essentially how to design and impose constraints on the department units so that the well-being of the overall organization is assured. The traditional way to coordinate decentralized systems is by means of the pricing mechanism; coordination is designed by analogy with the operation of a free market or competitive economy. Exchange of products between departments is allowed and internal prices are specified to assure accountability. The problem of effective decentralization reduces then, to the selection of the internal prices.

The bilevel programming model incorporates the above features. The framework embodies a corporate management unit at the higher level and one or more divisional or subordinate units at the lower level. The latter may be viewed as either separate

operating divisions of an organization or as departments within a firm such as production, finance and sales. This structure can be extended beyond two levels with the realization that attending behavioral and operational relationships become much more difficult to conceptualize and describe.

To formulate the problem mathematically, suppose that the higher-level decision maker or leader has control over the vector $\mathbf{x} \in X \subseteq R^n$ and that the subunits, collectively called the follower, have control over the vector $\mathbf{y} \in Y \subseteq R^m$. The leader goes first and selects an \mathbf{x} in an attempt to minimize $F(\mathbf{x}, \mathbf{y}(\mathbf{x}))$ subject perhaps to some additional constraints. The notation $\mathbf{y}(\mathbf{x})$ stresses the fact that the leader's problem is implicit in the \mathbf{y} variables. (Although for the sake of simplicity this notation will not be maintained in general, it should be remembered that \mathbf{y} is always a function of \mathbf{x} .) The follower observes the leader's actions and reacts by selecting a \mathbf{y} to minimize his objective function $f(\mathbf{x}, \mathbf{y})$, subject to a set of constraints in the \mathbf{y} variables for the particular value of \mathbf{x} chosen. When the feasible region of either player can be described by inequality constraints, the general BLPP is written as:

$$\begin{aligned} & \min_{\mathbf{x} \in X} F(\mathbf{x}, \mathbf{y}) \\ & \text{subject to } \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\ & \min_{\mathbf{y} \in Y} f(\mathbf{x}, \mathbf{y}) \\ & \text{subject to } \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \end{aligned} \tag{1.1}$$

where $F, f : R^n \times R^m \rightarrow R^1$, $\mathbf{G} : R^n \times R^m \rightarrow R^p$, and $\mathbf{g} : R^n \times R^m \rightarrow R^q$ are assumed to be continuous, twice differentiable functions. The sets X and Y place additional restrictions on the variables such as nonnegativity or integrality. When convenient nonnegativity restrictions on the variables will be subsumed in the functions $\mathbf{G}(\mathbf{x}, \mathbf{y})$ and $\mathbf{g}(\mathbf{x}, \mathbf{y})$. Depending on the functional forms of F, f, \mathbf{G} , and \mathbf{g} we get different versions of the BLPP. The simplest is the case where all the functions are linear. This problem has garnered the most attention and is discussed at length in Chapter 5. (As an aside, we note that for consistency most formulations in the text are stated as minimization problems absent of equality constraints. Nevertheless, such constraints can be added at either level, while one or both of the 'min' operators in (1.1) can be replaced by the 'max' operator without consequence.)

The area of multilevel programming was first defined in the mid-1970s by Chandler and Norton [C1] as a generalization of mathematical programming. In this context, the constraint region is implicitly determined by a series of optimization problems which must be solved in a predetermined sequence (cf. [B26]). Alternatively, the problem can be viewed as an l -person (l -levels), nonzero-sum game with perfect information (for the higher-level players at least) where the order of play is specified at the outset and the players' strategy sets are no longer assumed to be disjoint. Consequently, the moves available to a player change as the game progresses and hence may be limited by the actions of the preceding players. When interdependent strategy sets are introduced the difficulty of the overall problem markedly increases.

The control theorists were the first to formalize these notions but stopped short of algorithmic development (e.g., see [S13]).

Multilevel programming differs from the conventional formulation of the l -person game in that in the former the players are required to move in turn. When the moves are assumed to occur simultaneously, disagreement often arises as to which of several measures is most likely to predict the actual outcome. Such arguments are avoided in multilevel programming by appealing to its natural relationship with the standard form of the mathematical program. Solutions are defined accordingly. This leaves us free to focus on the challenging nature of the computations.

As a word of caution, the terminology surrounding the field of bilevel (multilevel) programming is often ambiguous. In this book, we take a relaxed view and use the terms bilevel programming, two-level programming, Stackelberg game, hierarchical optimization and variants thereof to refer to problem (1.1). In a more general context, two-level optimization or two-level mathematical programming refers to various kinds of nontraditional and reinterpreted problems exhibiting a hierarchical structure. The common element in each is the parametric approach taken in the analysis. Typically, the upper-level decision maker determines the optimal values of his control variables (which become parameters in the lower-level problem) so as to minimize his objective. The lower-level decision maker minimizes his objective with respect to a second set of control variables under the given parameter values. This arrangement gives rise to a general optimization problem containing subsidiary optimization problems, any of which can be viewed parametrically at some point in the analysis. Examples include the two-level design/planning problem, general resource allocation problems for hierarchical decentralized systems, decomposition methods for large-scale nonlinear programming, min-max problems, satisfaction optimization problems, as well as the Stackelberg problem. Each is a special case of (1.1) and is characterized by several objective functions, a two-level structure and a parametric interpretation of the variables (see [S10] for an indepth treatment of the broader subject). Absent from the above list is the multiple objective programming problem which does not fit the hierarchical structure.

Although the focus of the book is primarily on the BLPP, we mention one special case where the upper-level objective and constraint functions, F and \mathbf{G} , contain extremal-value functions (rather than extremal solutions) derived from the lower-level problem. The two most common extremal-value functions are

$$w(\mathbf{x}) \triangleq f(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) = \min_{\mathbf{y} \in S(\mathbf{x})} f(\mathbf{x}, \mathbf{y}) \quad (1.2a)$$

$$W(\mathbf{x}) \triangleq f(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) = \max_{\mathbf{y} \in S(\mathbf{x})} f(\mathbf{x}, \mathbf{y}) \quad (1.2b)$$

where $\mathbf{y}^*(\mathbf{x})$ denotes the extremal solution, namely the minimal or maximal solution to (1.2a) or (1.2b), respectively, and $S(\mathbf{x})$ is a constraint set for \mathbf{y} given by $S(\mathbf{x}) \triangleq \{\mathbf{y} \in Y : \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}\}$. In the literature, $w(\mathbf{x})$ and $W(\mathbf{x})$ are called

the minimal-value function and the maximal-value function, respectively, or the extremal-value functions together [F3]. Sometimes the term “optimal-value function” is used instead. Although these functions are continuous, they are not, in general, differentiable. Furthermore, the extremal solution $\mathbf{y}^*(\mathbf{x})$ is neither continuous nor differentiable which makes the BLPP so difficult to solve. A special case of (1.1) is

$$\begin{aligned} & \min_{\mathbf{x} \in X} F(\mathbf{x}, w(\mathbf{x})) \\ & \text{subject to } \mathbf{G}(\mathbf{x}, w(\mathbf{x})) \leq \mathbf{0} \\ & \quad w(\mathbf{x}) = \min_{\mathbf{y} \in Y} f(\mathbf{x}, \mathbf{y}) \\ & \quad \text{subject to } \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \end{aligned} \tag{1.3}$$

where the upper-level objective function and constraint set include the extremal-value function $w(\mathbf{x})$. In this problem, the leader evaluates the performance of the lower-level system through $w(\mathbf{x})$ which is obtained by solving the follower’s mathematical program in \mathbf{y} for \mathbf{x} fixed. A number of two-level optimization problems can be put in the form of (1.3) including the parametric design problem [S9, T1] and the infinitely constrained mathematical programming problem [G4].

1.2 EARLY WORK AND APPLICATIONS

Max-min formulations for strategic planning and pursuit-evasion games could be considered the first applications of bilevel programming. In these models, the players have directly opposing objectives, implying $f \hat{\equiv} -F$. This specialized form has been treated extensively elsewhere and will not be discussed here. The interested reader is referred to [D1] or Chapter 9 in [S10]. With regard to the full bilevel model, most applications appearing in the literature have dealt with central economic planning at the regional or national level. In this context, the government is considered the leader and controls a set of policy variables such as tax rates, subsidies, import quotas, and price supports (e.g., see several of the papers in [M14]). The particular industry targeted for regulation is viewed as the follower. In most cases, the follower tries to maximize net income subject to the prevailing technological, economic, and governmental constraints. Possible leader objectives include maximizing employment, maximizing production of a given product, or minimizing the use of certain resources.

The early work of Candler and Norton [C1] focusing on agricultural development in northern Mexico, illustrates how bilevel programming can be used to analyze the dynamics of a regulated economy. Similarly, Fortuny-Amat and McCarl [F7] present a regional model that pits fertilizer suppliers against local farm communities, while Aiyoshi and Shimizu [A2] and Bard [B3] discuss resource allocation in a decentralized firm. In the case of the latter, a central unit supplies resources to its manufacturing facilities which make decisions concerning production mix and output. Organizational procedures and conflicting objectives over efficiency, quality and performance lead to

a hierarchical formulation. In a work related to the original Stackelberg model of a single leader–follower oligopolistic market in which a few firms supply a homogeneous product, Sherali [S6] presents an extension to M leader firms and discusses issues related to the existence, uniqueness, and derivation of equilibrium solutions. His analysis provides sufficient conditions for some useful convexity and differentiability properties associated with the followers' reaction curves.

In a recent study, the French Ministry of Agriculture used bilevel programming to examine the economics of promoting biofuel production from farm crops [B8]. The stumbling block to such a program is that the petro-chemical industry's costs for producing fuels from hydrocarbon-based raw materials is significantly less than it is for producing biofuels. Without incentives in the form of tax credits, industry will not buy farm output for conversion. The problem faced by the government is to determine the level of tax credits for each final product or biofuel that industry can produce while minimizing public outlays. A secondary objective is to realize some predefined level of land usage for nonfood crops. Industry is assumed to be neutral in this scenario and will produce any biofuel that is profitable. In the model, the agricultural sector is represented by a subset of farms in an agriculturally intensive region of France and is a profit maximizer. It will use the land available for nonfood crops only as long as the revenue generated from this activity exceeds the difference between the set-aside payments now received directly from the government and the maintenance costs incurred under the current support program. The resultant bilevel model contains 3628 variables and 3230 constraints at the lower level, and 8 variables and 10 constraints at the upper level. Both objective functions are quadratic and all constraints are linear. The details are presented in Chapter 12.

In an earlier effort, Anandalingam and Apprey [A6] investigated the problem of conflict resolution by postulating the existence of an arbitrator who acts as the leader in a Stackelberg game. They presented models for different configurations of the resulting linear bilevel programs and proposed a series of solution algorithms. The models were illustrated with an application involving a conflict over water between India and Bangladesh. It was shown that both parties could benefit if they submitted to arbitration run by a disinterested third party such as the United Nations.

More recently, researchers have applied bilevel formulations to the network design problem ([B20, M3]) arising in transportation systems. In the accompanying formulation, a central planner controls investment costs at the system level, while operational costs depend on traffic flows, which are determined by the individual users' route selection. Because users are assumed to make decisions so as to maximize their peculiar utility functions, their choices do not necessarily coincide (and may, in fact, conflict) with the choices that are optimal for the system. Nevertheless, the central planner can influence the users' choices by improving some links, making them relatively more attractive than the others. In deciding on these improvements, the central planner tries to influence the users' preferences in such a way that total costs are minimized.

The partition of the control variables between the upper and lower levels naturally leads to a bilevel formulation.

A conceptual framework for the optimization of Tunisia’s inter-regional highways was proposed by Ben-Ayed et al. [B19]. Their formulation includes 2683 variables (2571 at the lower level) and 820 constraints (all at the lower level); the follower’s problem can be divided into two separate subproblems as a direct consequence of the bilevel approach. The first centers on the user-optimized flow requirement (user-equilibrium) and the second on the nonconvex improvement functions. Because none of the standard algorithmic approaches discussed in later chapters can handle problems of this size, a specialized algorithm was devised to deal with each of the two lower-level problems separately. At each iteration, the algorithm tries to find a better compromise with the user, while including the smallest possible number of nonconvex improvement functions to get the exact solution with the minimum computational effort. The authors claim that despite the large number of variables and constraints, optimality was achieved (see Chapter 10 for the details).

In 1992, Anandalingam and Friesz [A7] edited a special issue of *Annals of Operations Research* that summarized the developments in bilevel programming to date and presented a new body of theory and algorithms. Seven of the 16 papers featured applications on network design, facilities location, electric utility planning, and supplier–customer interactions in acceptance sampling. In a more recent edited volume, several theoretical advances were described and many new applications were highlighted [M14].

In Part III of the book, a number of the applications mentioned above are discussed with an eye toward implementation. Before getting to the specifics of the theory and the design of algorithms, we present a number of characteristics of the BLPP that have complicated efforts to design efficient computational schemes.

1.3 STRUCTURAL CONSIDERATIONS

In studying optimization problems, it is important to know whether solutions exist and, if so, what form they take. Under certain circumstances problem (1.1) can be written as a standard mathematical program. But even when all the functions are linear, the resultant model is nonconvex and difficult to solve. In Chapter 5, we show that the feasible region (known as the inducible region) of the linear BLPP is equivalent to a piecewise linear equality constraint. In Chapter 7, this result is extended to the convex quadratic case where F is strictly convex, f is quadratic, and \mathbf{G} and \mathbf{g} are linear. We now present several curious properties associated with problem (1.1). To those familiar with nonconvex programming, these may seem counterintuitive.

1.3.1 Indifference Points and Nonexistence of Solutions

Unlike general mathematical programs, the BLPP may not possess a solution even when F , f , \mathbf{G} and \mathbf{g} are continuous and the sets X and Y are compact. To see this consider problem (1.1) and suppose that the leader selects a point $\hat{\mathbf{x}}$. The follower is then faced with a simple minimization problem parameterized by the vector $\hat{\mathbf{x}}$. In certain instances, the solution set to this problem may contain more than one member. For example, if all the constraint functions are linear, it is possible that $\mathbf{y}(\hat{\mathbf{x}})$, the set of all solutions to the lower-level problem for $\mathbf{x} = \hat{\mathbf{x}}$ fixed, might consist of some nontrivial subset of a hyperplane. This would mean that the follower would be indifferent to any point on that hyperplane; however, the leader might not feel the same indifference with respect to his own objective function. His best result might only be realized at one particular point in $\mathbf{y}(\hat{\mathbf{x}})$, but there may be no way to induce the follower to show any preference for that point. It may further be true that if the leader chooses any point other than $\hat{\mathbf{x}}$, his minimum cost will never be realized. This situation is illustrated in the following example:

$$\min_{\mathbf{x}} \left\{ F = (x_1, x_2) \begin{pmatrix} 2 & 3 \\ 4 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} : \begin{array}{l} x_1 \geq 0, x_2 \geq 0 \\ x_1 + x_2 = 1 \end{array} \right\} \quad (1.4a)$$

$$\min_{\mathbf{y}} \left\{ f = (x_1, x_2) \begin{pmatrix} -1 & -4 \\ -3 & -2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} : \begin{array}{l} y_1 \geq 0, y_2 \geq 0 \\ y_1 + y_2 = 1 \end{array} \right\} \quad (1.4b)$$

Note the structure of (1.4a,b) is identical to that of a bimatrix game but now the order of play is sequential rather than simultaneous.

The solution \mathbf{y} to the lower-level problem as a function of \mathbf{x} is

$$\mathbf{y}(\mathbf{x}) = \begin{cases} (1, 0) & \text{for } x_1 + 3x_2 > 4x_1 + 2x_2; \text{ i.e., } x_1 < 1/4 \\ y_1 + y_2 = 1 & \text{for } x_1 = 1/4 \\ (0, 1) & \text{for } x_1 > 1/4 \end{cases}$$

Substituting these values into (1.4a), the upper-level problem becomes

$$\min_{\mathbf{x}} F = \begin{cases} 2x_1 + 4x_2 & ; x_1 < 1/4 \\ 2y_1 + 3/2 & (0 \leq y_1 \leq 1) ; x_1 = 1/4 \\ 3x_1 + x_2 & ; x_1 > 1/4 \end{cases} \quad (1.5)$$

$$\text{subject to } x_1 + x_2 = 1, x_1 \geq 0, x_2 \geq 0$$

At $x_1 = 1/4$, F is not well-defined and an attempt to solve (1.5) leads to difficulties. This can be seen by substituting $1 - x_1$ for x_2 and plotting the value of the objective function F as x_1 varies between zero and one. Figure 1.1 depicts the resultant graph.

The smallest payoff ($F = 1.5$) for the leader occurs when he selects $\mathbf{x} = (1/4, 3/4)$ and the follower selects $\mathbf{y} = (0, 1)$. There is no guarantee, however, that the follower

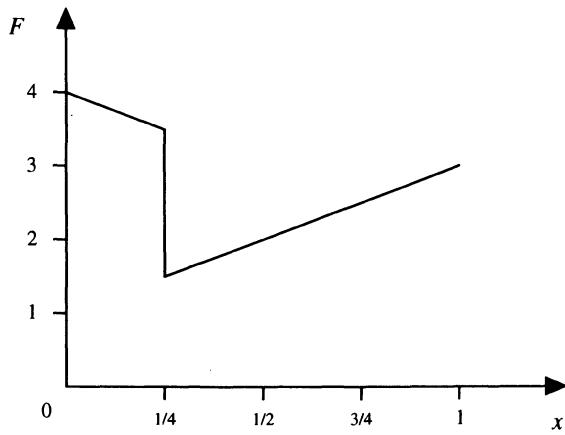


Figure 1.1 Bilevel program with no solution

will choose $(0, 1)$ since he is indifferent to any point on the line $y_1 + y_2 = 1$. The only way to assure this selection is for the leader to pick a point such that $x_1 > 1/4$, say,

$$\mathbf{x}^* = (1/4 + \varepsilon, 3/4 - \varepsilon)$$

where $\varepsilon > 0$ is arbitrarily small. The corresponding payoff is $F^* = 1.5 + 2\varepsilon$ which is not the best result that he could have achieved. Thus there is no sure way for the leader to realize his minimum payoff.

This result suggests that some type of cooperation, perhaps in the form of side payments, would work to the advantage of both players. Such a change in the rules, though, is outside the context of the current problem and will not be considered here. Note that if cooperation among the players were permitted, it might be mutually beneficial for one player to accept a higher cost than he might ordinarily realize in order for the other player to realize a much smaller cost. The difference could then be split. In Chapter 5 we derive an alternative representation of the BLPP that is equivalent to a standard mathematical program. In the accompanying reformulation ambiguities or indifference arising in solution values are resolved by giving the leader complete control over all multiple optimal solutions $\mathbf{y}(\mathbf{x})$. As a result the transformed problem will always have a solution. In Section 5.1, an auxiliary problem is presented that can be used to determine if the derived solution actually solves the bilevel program in its original form.

1.3.2 Significance of Order of Play

Unlike the rules in noncooperative game theory where each player must choose a strategy simultaneously, the definition of multilevel programming requires that the

leader moves first. To demonstrate the significance of the order of play, let us reverse the structure of problem (1.4a,b). The new problem becomes

$$\begin{aligned} \min_{\mathbf{y}} \quad & f = -(x_1 + 3x_2)y_1 - (4x_1 + 2x_2)y_2 \\ \text{subject to} \quad & y_1 + y_2 = 1, \quad y_1 \geq 0, \quad y_2 \geq 0 \end{aligned} \quad (1.6a)$$

$$\begin{aligned} \min_{\mathbf{x}} \quad & F = (2y_1 + 3y_2)x_1 + (4y_1 + y_2)x_2 \\ \text{subject to} \quad & x_1 + x_2 = 1, \quad x_1 \geq 0, \quad x_2 \geq 0 \end{aligned} \quad (1.6b)$$

The solution to the lower-level problem (1.6b) for \mathbf{y} fixed is

$$\mathbf{x}(\mathbf{y}) = \begin{cases} (1, 0) & \text{for } 2y_1 + 3y_2 < 4y_1 + y_2; \text{ i.e., } y_1 > 1/2 \\ x_1 + x_2 = 1 & \text{for } y_1 = 1/2 \\ (0, 1) & \text{for } y_1 < 1/2 \end{cases}$$

We can now rewrite the upper-level problem by substituting $\mathbf{x}(\mathbf{y})$ into (1.6a) giving

$$\begin{aligned} \min_{\mathbf{y}} \quad & f = \left\{ \begin{array}{ll} -y_1 - 4y_2 & ; y_1 > 1/2 \\ -(3 - 2x_1)y_1 - (2x_1 + 2)y_2 & ; y_1 = 1/2 \\ -3y_1 - 2y_2 & ; y_1 < 1/2 \end{array} \right\} \\ \text{subject to} \quad & y_1 + y_2 = 1, \quad y_1 \geq 0, \quad y_2 \geq 0 \\ \\ & = \min_{0 \leq y_1 \leq 1} \left\{ \begin{array}{ll} -4 + 3y_1 & ; y_1 > 1/2 \\ -5/2 & ; y_1 = 1/2 \\ -2 - y_1 & ; y_1 < 1/2 \end{array} \right\} \end{aligned}$$

The solution $\mathbf{y}^* = (1/2, 1/2)$ can readily be determined from the plot of f for $0 \leq y_1 \leq 1$ given in Fig. 1.2.

The corresponding solution \mathbf{x}^* for the lower-level player is any point within the set $\{(x_1, x_2) : x_1 + x_2 = 1, x_1 \geq 0, x_2 \geq 0\}$. Thus as in problem (1.4a,b), the lower-level player is indifferent to a range of points but now the upper-level player will receive the same payoff regardless of the lower-level player's choice.

As an aside, if we assume that the leader in problem (1.4a,b) realizes his minimum cost, the two problems can be compared at their solution points. The corresponding data are presented in Table 1.1. The last column in the table contains the solution to the bimatrix game as defined by a Nash equilibrium [B14]. It can be seen that if the player who controls the \mathbf{x} variables makes the first move, he will realize a smaller cost than if the situation was reversed or if both players moved simultaneously. Of course, in most practical settings the order of play is determined logically by the underlying dynamics (e.g., by government regulation and industry's response) so changing the order would make no sense.

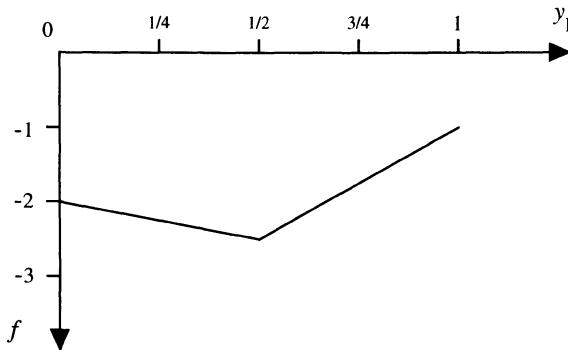


Figure 1.2 Solution to problem (1.6a,b)

Table 1.1 Significance of order of play

	Problem (1.4a,b)	Problem (1.6a,b)	Bimatrix game
Solution (\mathbf{x})	(1/4, 3/4)	$x_1 + x_2 = 1$	(1/4, 3/4)
Cost (F)	1.5	2.5	2.5
Solution (\mathbf{y})	(0, 1)	(1/2, 1/2)	(1/2, 1/2)
Cost (f)	-2.5	-2.5	-2.5

1.4 SCOPE AND OUTLINE

The basic problem of decentralized control, as viewed from the corporate level, is one of inducing its subordinate units to increase those activities which yield external economies and to decrease those which produce external diseconomies by just the right amount. While standard decomposition techniques have been used successfully in many areas, they generally fail to address this problem in full. As mentioned, the bilevel programming model makes up for many of their inherent shortcomings by recognizing that the decision making process in a hierarchical decentralized organization is both sequential and multiobjective. In most such organizations, it is either impractical or impossible for the higher-level decision maker to impose his utility function on the subordinate divisions. The main feature of the BLP model is that it provides pairwise sensitivity information between corporate and divisional payoffs, while permitting each unit to pursue its individual objective.

From its inception, bilevel programming has captured the interest of a broad range of researchers and practitioners alike. The first efforts concentrated on algorithmic development for the linear version of the problem. Once the field became established,

mathematical programmers were quick to derive new theory, thereby widening the possibilities of solving more general problems. In parallel, a number of real-world applications began to spring up, many of which were too big to be solved with available algorithms. This led to the development of ad hoc procedures and ultimately to the adaptation of intelligent heuristics. One of the goals of this book is to highlight these milestones. In so doing we provide a ready reference for those new to bilevel programming as well as for those wrestling with practical problems and applications. Operations researchers, management scientists, control theorists, economists, and game theorists constitute the main audience.

The book is divided into three parts. For completeness, Part I contains summary chapters on linear, integer and nonlinear programming. The presentation is intended to be rigorous but not necessarily complete. The aim is to provide sufficient background material of a theoretical and methodological nature to allow the reader to understand how bilevel programming algorithms work and what is needed for their implementation. For example, to appreciate the the K th-best algorithm in Section 5.3.1, it is necessary to understand vertex enumeration in a linear programming context. We therefore discuss the simplex method in detail but not interior point methods since there are no BLP algorithms that are based on this approach. However, we do discuss penalty methods in the nonlinear programming chapter since several BLP algorithms use penalty functions to eliminate constraints. What is missing in Part I is a general treatment of nondifferentiable optimization and global solution techniques. To have included such material would have greatly increased the length of the book and gone beyond its intended scope. Readers wanting more background in these areas should see [H10, K4, L3]. Complementary material on the theory of two-level structures can be found in [S10] which also provides an indepth treatment of nondifferentiable mathematical programming.

Part II is the heart of the book and is devoted to BLP algorithms, implementation issues and supporting theory. Chapter 5 begins with a formal definition of the linear BLPP and presents the standard terminology. The two principal concepts are the *rational reaction set* and the *inducible region*. A number of theoretical properties are highlighted and five algorithms are discussed. Each is designed to find a global optimum. Although there have been nearly two dozen such algorithms proposed, the presentation is limited to those that are deemed either the most efficient or the most important from a conceptual point of view. A section on computational comparisons is included to give the reader an appreciation for the effort and efficiency with which problems of various size can be solved. Chapter 6 extends the linear model to include discrete variables. Several versions of the problem where the variables assigned to a player are either all continuous or all integer, are examined from a theoretical point of view. The mixed-integer case is then investigated and a branch and bound algorithm is presented. This is followed by the development of a specialized algorithm for the zero-one case.

Chapters 7 and 8 deal with the convex and the general bilevel programming problem, respectively. The emphasis in the former chapter is on the quadratic case. The two algorithms featured are extensions of those designed to solve the linear BLPP. In Chapter 8, we discuss a variety of solution techniques such as penalty methods, steepest descent, rectangular partitioning, subgradient optimization, and concave programming. The nonconvex, nondifferentiable nature of the problem, though, severely undermines our ability to verify that global optimality has been achieved. In most cases, the best that can be expected are local optima. The final chapter in Part II addresses heuristics. Surprisingly, there has been little work in this area and only on the linear problem. Accompanying computational experience suggests some promise in the use of the more well-known procedures such as tabu search, genetic algorithms, and simulated annealing but ample room exists for improvement.

The third part of the book deals with applications. The intent is to provide a representative sampling of what has been done on a practical level, as well as to point out the shortcomings of current technology. The first application focuses on transportation network design. The resultant model is nonlinear and is solved with an ad hoc procedure. The work is of historical importance because it broke with the traditional approach to highway planning and design at the time. Chapter 11 discusses a production planning problem with inexact demand – a worst case scenario when demand is unknown. The bilevel programming formulation is linear but contains both continuous and discrete variables. Solutions are obtained with the algorithm presented in Section 6.4. The final chapter highlights a government regulation problem associated with the promotion of biofuels. Both objective functions are quadratic and all the constraints are linear. The resultant model is a large-scale almost linear BLPP. The nonlinearity in the leader's objective function rendered the problem unsolvable with standard BLP techniques so an “engineering” approach was taken. This required a tradeoff between theory and practice, a situation common in the real world.

In summary, the book is intended as a reference for students, researchers and practitioners wishing to advance their appreciation for the fundamentals and application of bilevel programming. It is geared towards those with a background in optimization at the graduate level. As such, it could serve as text for a one semester course on, say, hierarchical optimization, in either an operations research or a management science program.

LINEAR PROGRAMMING

2.1 INTRODUCTION

In general, constrained optimization problems can be written as

$$\min \{f(\mathbf{x}) : \mathbf{h}(\mathbf{x}) = \mathbf{0}, \mathbf{g}(\mathbf{x}) \leq \mathbf{0}\}$$

where $\mathbf{x} \in R^n$, $f : R^n \rightarrow R^1$, $\mathbf{h} : R^n \rightarrow R^m$ and $\mathbf{g} : R^n \rightarrow R^q$. The simplest form of this problem is realized when the functions $f(\mathbf{x})$, $\mathbf{h}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ are all linear in \mathbf{x} . The resulting model is known as a linear program (LP) and plays a central role in virtually every branch of optimization. Many real situations can be formulated or approximated as LPs, optimal solutions are relatively easy to calculate, and computer codes for solving very large instances consisting of millions of variables and tens of thousands of constraints are commercially available. Another attractive feature of linear programs is that various subsidiary questions related, for example, to the sensitivity of the optimal solution to changes in the data and the inclusion of additional variables and constraints can be analyzed with little effort.

The importance of this topic cannot be overstated. Linear programming algorithms are frequently used as subroutines for solving more difficult optimization problems in nonlinear and integer programming. The linear bilevel programming problem (BLPP) or Stackelberg game discussed in Chapter 5 is a case in point. The primary purpose of this chapter is to provide a foundation for the presentation of algorithms for solving that problem. The most successful algorithms in this regard have tried to exploit the polyhedral nature of the feasible region of the BLPP using branch and bound techniques to narrow the solution space. This can be done most effectively with some form of vertex enumeration — the idea behind the simplex method of linear programming. Therefore, we concentrate on this method alone with only passing reference to the more recently developed techniques that restrict the search for a solution to the interior of the feasible region. These are known as *interior point methods* (e.g., see [C4, H9, K2, L8, M7, M8]).

The linear programming problem was initially stated using quasi-economic terminology which to some extent obscures the basic numerical processes that are involved in the analysis. Our presentation aims to make these processes clear while retaining the traditional nomenclature. One main feature of the traditional approach is that the linear programming is expressed in *standard form*

$$\min f(\mathbf{x}) \triangleq \mathbf{c}\mathbf{x} \quad (2.1a)$$

$$\text{subject to } \mathbf{A}\mathbf{x} = \mathbf{b} \quad (2.1b)$$

$$\mathbf{x} \geq \mathbf{0} \quad (2.1c)$$

where \mathbf{c} is an n -dimensional row vector, \mathbf{b} an m -dimensional column vector, \mathbf{A} an $m \times n$ matrix with $m \leq n$, and $\mathbf{x} \in R^n$. Thus the allowable constraints on the variables are either linear equations or nonnegative bounds. The coefficients \mathbf{c} in the objective function are referred to as costs, the \mathbf{A} matrix comprises the technological coefficients, and the right-hand side vector \mathbf{b} denotes the resource levels. It is generally assumed that the data $(\mathbf{c}, \mathbf{A}, \mathbf{b})$ defining the problem are known and deterministic. An example with four variables ($n = 4$) and two technological constraints ($m = 2$) is

$$\begin{aligned} \min \quad & x_1 + 2x_2 + 3x_3 + 4x_4 \\ \text{subject to} \quad & x_1 + x_2 + x_3 + x_4 = 1 \\ & x_1 + x_3 - 3x_4 = \frac{1}{2} \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned} \quad (2.2)$$

More general LPs can be reduced to standard form without undue difficulty, albeit with some possible loss in efficiency. For instance, a general linear inequality $\mathbf{a}^T \mathbf{x} \leq b$ can be transformed by adding a *slack variable* $x_s \in R^1$ to the left-hand side to get $\mathbf{a}^T \mathbf{x} + x_s = b$ with the nonnegativity requirement $x_s \geq 0$. To convert a “greater than or equal to” inequality of the form $\mathbf{a}^T \mathbf{x} \geq b$ to an equation, a nonnegative *surplus variable* x_s is subtracted from the left-hand side to get $\mathbf{a}^T \mathbf{x} - x_s = b$.

Standard form usually assumes that the resource vector \mathbf{b} is greater than or equal to zero. It should be clear that by suitably multiplying by minus one and adjoining slack and surplus variables, any set of linear inequalities can be converted to the form given in (2.1b) with $\mathbf{b} \geq \mathbf{0}$. More general bounds $x_j \geq l_j$ can be dealt with by a shift in the origin; i.e., by replacing x_j with $\hat{x}_j + l_j$, where $\hat{x}_j \geq 0$. In fact very little is lost in complexity if the bounds in (2.1c) are expressed as $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$. Also, if a linear program is given in standard form except that one or more of the decision variables is not required to be nonnegative, the problem can be transformed into standard form by either of two simple techniques.

Free variable method 1: Suppose in (2.1c), x_j is not restricted to be nonnegative and hence can take on any real value. We then write $x_j = x_j^+ - x_j^-$ and require that $x_j^+ \geq 0$ and $x_j^- \geq 0$. If we substitute $x_j^+ - x_j^-$ for x_j everywhere in (2.1a) and (2.1b), the linearity of the constraints is preserved and all variables are now required to be nonnegative. The problem is then expressed in terms of the $n + 1$ variables

$x_1, x_2, \dots, x_j^+, x_j^-, \dots, x_n$. Of course, there is a certain amount of redundancy introduced by this technique because a constant added to x_j^+ and x_j^- does not change x_j (that is, the representation of a given value of x_j is not unique). Nevertheless, this does not hinder the simplex method.

Free variables method 2: A second approach when x_j is unrestricted in sign is to eliminate x_j together with one of the constraint equations. Take any one of the m equations in (2.1b) that has a nonzero coefficient for x_j , say, equation 1,

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1j}x_j + \cdots + a_{1n}x_n = b_1 \quad (2.3)$$

where $a_{1j} \neq 0$. The variable x_j can be expressed as a linear combination of the other variables plus a constant. Substituting this expression for x_j everywhere in (2.1a) and (2.1b) gives a new problem of exactly the same form but in terms of the variables $x_1, x_2, \dots, x_{j-1}, x_{j+1}, \dots, x_n$ only. Moreover, the equation used to determine x_j is now identically zero and can also be eliminated. This scheme is valid since any combination of nonnegative variables $x_1, x_2, \dots, x_{j-1}, x_{j+1}, \dots, x_n$ produces a feasible x_j from (2.3). Note that this would not necessarily be true if x_j were originally restricted to be nonnegative. As a result of this simplification, we obtain a standard linear program having $n - 1$ variables and $m - 1$ equality constraints. After a solution is found to the reduced problem the value of x_j can be determined from (2.3).

As is true with any optimization problem, it is important to realize that a linear program may have no solution, either because there is no feasible point (the problem is *infeasible*), or because $f(\mathbf{x}) \rightarrow -\infty$ for \mathbf{x} in the feasible region (the problem is *unbounded*). Nevertheless, we show that there is no difficulty in detecting these situations so we concentrate on the usual case in which a (possibly nonunique) solution exists. It is also convenient to assume that the equality constraints (2.1b) are linearly independent, implying that the rank of the \mathbf{A} matrix is m . In theory, this can always be achieved by either removing dependent equations or adding artificial variables, although in practice numerical difficulties might arise if this dependence is not detected.

Considering (2.1) in more detail, if $m = n$, then the equations $\mathbf{Ax} = \mathbf{b}$ determine a unique solution under the independence assumption, and the objective function $c\mathbf{x}$ and the bounds $\mathbf{x} \geq \mathbf{0}$ play no part. In most cases, however, $m < n$ so that the system $\mathbf{Ax} = \mathbf{b}$ is underdetermined and $n - m$ degrees of freedom remain. In particular, the system can determine only m variables, given values for the remaining $n - m$ variables. For example, the equations $\mathbf{Ax} = \mathbf{b}$ in (2.2) can be rearranged as

$$\begin{aligned} x_1 &= \frac{1}{2} - x_3 + 3x_4 \\ x_2 &= \frac{1}{2} - 4x_4 \end{aligned} \quad (2.4)$$

which determines x_1 and x_2 in terms of x_3 and x_4 , or alternatively as

$$\begin{aligned}x_1 &= \frac{7}{8} - \frac{3}{4}x_2 - x_3 \\x_4 &= \frac{1}{8} - \frac{1}{4}x_2\end{aligned}\tag{2.5}$$

which determines x_1 and x_4 from x_2 and x_3 , and so on. It is important to consider what values these remaining $n - m$ variables can take in the standard form of the problem. The objective function $c\mathbf{x}$ is linear and so contains no curvature which can give rise to a minimizing point. Hence such a point must be created by the conditions $x_j \geq 0$ becoming active on the boundary of the feasible region. For example, if (2.5) is used to eliminate the variable x_1 and x_4 from problem (2.2), then the objective function can be written as

$$f = x_1 + 2x_2 + 3x_3 + 4x_4 = \frac{11}{8} + \frac{1}{4}x_2 + 2x_3\tag{2.6}$$

which clearly has no minimum value unless the conditions $x_2 \geq 0$, $x_3 \geq 0$ are imposed. In this case, the minimum occurs when $x_2 = 0$, $x_3 = 0$.

To illustrate the nature of an LP solution graphically, consider the simpler constraint set $2x_1 + x_2 = 3$ and $x_1 \geq 0$, $x_2 \geq 0$. The feasible region is shown in Fig. 2.1 as the bold line joining points $\mathbf{a} = (0, 3)$ and $\mathbf{b} = (\frac{3}{2}, 0)$. When the objective function $f(\mathbf{x})$ is linear the solution must occur at either \mathbf{a} or \mathbf{b} with either $x_1 = 0$ or $x_2 = 0$. For example, for $f = -3x_1 + 4x_2$ or $f = x_1 + x_2$ the solution occurs at \mathbf{a} (try other linear functions). If $f = 2x_1 + x_2$, however, any point on the line segment connecting \mathbf{a} and \mathbf{b} provides the same objective function value, implying that a solution need not be unique.

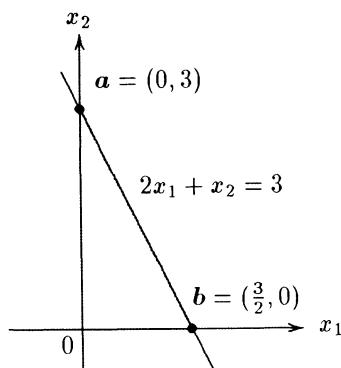


Figure 2.1 Constraints for simple LP

The example in Fig. 2.1 demonstrates that if the feasible region is bounded, a solution of an LP problem in standard form always exists at one particular extreme point or vertex of the feasible region, with at least $n - m$ variables equal to zero and the remaining m variables being uniquely determined by the equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ and taking nonnegative values. This result is fundamental to the development of algorithms for solving LPs and will be established vigorously in Section 2.1.3. Recall Weierstrass' Theorem which states in its most elementary form that a continuous function f defined on a compact set S has a minimum point in S (see Section 4.1.3). If S is not compact, it still can be shown that if a finite solution exists to an LP, there is at least one vertex solution. In our case the set $S = \{\mathbf{x} \in R^n : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$ denotes the feasible region (2.1b)–(2.1c).

The main difficulty in linear programming is to find which $n - m$ variables take zero value at the solution. The brute force approach to making this determination is to enumerate all $\binom{n}{m}$ combinations of variables, solve the corresponding set of m linear equations in m variables, evaluate the objective function at each feasible combination, and select the best result. For all but the smallest problems, this approach is highly inefficient. For the linear BLPP, a variant of this idea has proven successful. In Section 2.3.7, we give an algorithm for ranking (and hence enumerating) all vertices of a polytope.

The earliest algorithm for solving the linear program is due to George Dantzig and is called the simplex method. The basic idea is to traverse the vertices of the underlying polytope in a systematic way that avoids investigating infeasible points. Different combinations of variables are examined one at a time without ever returning to a previously explored combination (vertex). The simplex method still predominates today with different variations existing, depending on which intermediate quantities are computed and which techniques are used to handle the linear algebra. The earliest tableau form was quickly superseded by the more efficient revised simplex method that is described in Section 2.2. In the last few years, computational schemes using matrix factorizations have been adopted to control round-off errors more effectively. For large sparse problems with up to 10^6 variables, LU decomposition with threshold pivoting (see [S17]) is the method of choice for storing a representation of the basis inverse. This replaced the product form of the inverse which can still be found in some older implementations. Larger LPs that have a network structure can be solved efficiently with specialized codes. Nevertheless, even if a problem has a well-defined solution, all current simplex-type methods may have difficulty finding it if degeneracy is present. This issue is taken up in Section 2.2.4.

2.1.1 Basic Solutions

In working toward a solution to a linear program, it is convenient to start with an analysis of the system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ given in (2.1b), where once again, \mathbf{x} is an n -dimensional vector, \mathbf{b} is an m -dimensional vector, and \mathbf{A} is an $m \times n$ matrix. Suppose that from the n columns of \mathbf{A} we select a set of m linearly independent columns (such a set exists if the rank of \mathbf{A} is m). For notational simplicity assume that we select the first m columns of \mathbf{A} and denote the corresponding $m \times m$ matrix by \mathbf{B} . The matrix \mathbf{B} is then nonsingular and can be uniquely determined by solving the equation

$$\mathbf{B}\mathbf{x}_B = \mathbf{b}$$

for the m -dimensional vector \mathbf{x}_B . By putting $\mathbf{x} = (\mathbf{x}_B, \mathbf{0})$, that is, by setting the first m components of \mathbf{x} to those of \mathbf{x}_B and the remaining components to zero, we obtain a solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$. This leads to the following definition.

Definition 2.1.1 Given a set of m simultaneous linear equations (2.1b) in n unknowns, let \mathbf{B} be any nonsingular $m \times m$ matrix made up of columns of \mathbf{A} . If all the $n - m$ components of \mathbf{x} not associated with columns of \mathbf{B} are set equal to zero, the solution to the resulting set of equations is said to be a *basic solution* to (2.1b) with respect to the basis \mathbf{B} . The components of \mathbf{x} associated with columns of \mathbf{B} are called *basic variables*.

The m linearly independent columns of \mathbf{B} can be regarded as a basis for the space \mathbb{R}^m , hence the terminology. A basic solution corresponds to an expression for the vector \mathbf{b} as a linear combination of these basis vectors. This interpretation is further discussed in Section 2.1.2.

In some instances, (2.1b) may have no basic solution. However, to avoid trivialities and nonessential difficulties, a number of elementary assumptions regarding the nature of \mathbf{A} will be made. These have already been mentioned: the first is that the number of \mathbf{x} variables exceeds the number of equality constraints ($n > m$); the second is that the rows of \mathbf{A} are linearly independent. A linear dependence among the rows of \mathbf{A} would imply either a redundancy in the m equations that could be eliminated or contradictory constraints and hence no solution to (2.1b).

Given the assumption that \mathbf{A} has full row rank, the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ always has a solution and, in fact, it will always have one basic solution; however, the basic variables in a solution are not necessarily all nonzero. This is noted in the following definition.

Definition 2.1.2 A *degenerate basic solution* is said to occur if one or more of the basic variables in a basic solution has value zero.

Thus in a nondegenerate basic solution the basic variables, and hence the basis \mathbf{B} , can be immediately identified from the positive components of the solution. The same

cannot be said for a degenerate basic solution because a subset of the zero-valued basic and nonbasic variables can be interchanged. This implies some amount of ambiguity but does not cause any difficulties.

So far in the discussion we have only treated the equality constraints of the linear program to the exclusion of the nonnegativity constraints on the variables. We now want to consider the full set of constraints given by (2.1b) and (2.1c) for an LP in standard form.

Definition 2.1.3 A vector $\mathbf{x} \in S = \{\mathbf{x} \in R^n : \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq 0\}$ is said to be *feasible* to the linear programming problem in standard form; a feasible solution that is also basic is said to be a *basic feasible solution*. If this solution is degenerate, it is called a *degenerate basic feasible solution*.

2.1.2 Fundamental Theorem

In this section, we establish the relationship between optimality and basic feasible solutions in the fundamental theorem of linear programming. The proof is as important as the theorem itself because it underlies the development of the simplex algorithm. The results tell us that when seeking a solution to an LP it is only necessary to consider basic feasible solutions.

Theorem 2.1.1 Given a linear program in standard form (2.1) where \mathbf{A} is an $m \times n$ matrix of rank m ,

- i) if there is a feasible solution, there is a basic feasible solution;
- ii) if there is an optimal feasible solution, there is an optimal basic feasible solution.

Proof of (i): Denote the columns of \mathbf{A} by $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n$ and let $\mathbf{x} = (x_1, x_2, \dots, x_n)$ be a feasible solution. In terms of the columns of \mathbf{A} , a solution can be written as

$$\mathbf{a}_1 x_1 + \mathbf{a}_2 x_2 + \cdots + \mathbf{a}_n x_n = \mathbf{b}$$

Assume that p of the variables x_j are greater than zero, and for convenience, that they are the first p variables. This gives

$$\mathbf{a}_1 x_1 + \mathbf{a}_2 x_2 + \cdots + \mathbf{a}_p x_p = \mathbf{b} \quad (2.7)$$

Two cases must now be considered corresponding to whether or not the columns $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_p$ are linearly independent.

Case 1: Assume $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_p$ are linearly independent, implying that $p \leq m$. If $p = m$, the solution is basic and the proof is complete. If $p < m$, the fact that \mathbf{A} has rank m means that $m - p$ vectors can be found so that the resulting set of m vectors is linearly independent (see Corollary 2.1.1). Assigning the value zero to the corresponding $m - p$ variables yields a (degenerate) basic feasible solution.

Case 2: Assume $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_p$ are linearly dependent, implying that there is a non-trivial linear combination of these vectors that is zero. Thus there are constants w_1, w_2, \dots, w_p at least one of which is positive such that

$$\mathbf{a}_1 w_1 + \mathbf{a}_2 w_2 + \cdots + \mathbf{a}_p w_p = \mathbf{0} \quad (2.8)$$

Multiplying each equation in (2.8) by a scalar ε and subtracting it from (2.7), we obtain

$$\mathbf{a}_1(x_1 - \varepsilon w_1) + \mathbf{a}_2(x_2 - \varepsilon w_2) + \cdots + \mathbf{a}_p(x_p - \varepsilon w_p) = \mathbf{b}$$

which holds for every ε . In other words, for each ε the components $(x_j - \varepsilon w_j)$ correspond to a solution of the linear equalities; however, they may violate $(x_j - \varepsilon w_j) \geq 0$. Denoting $\mathbf{w} = (w_1, w_2, \dots, w_p, 0, 0, \dots, 0)$, we see that for any ε ,

$$\mathbf{x} - \varepsilon \mathbf{w} \quad (2.9)$$

is a solution to the equality constraints. For $\varepsilon = 0$, this reduces to the original feasible solution \mathbf{x} . As ε is increased from zero, the various components increase, decrease, or remain the same, depending on whether the corresponding component w_j is positive, negative or zero. Because at least one w_j is positive, at least one component will decrease as ε is increased. Now, let us increase ε to the first point where one or more components become zero; i.e.,

$$\varepsilon = \min \left\{ \frac{x_j}{w_j} \mid w_j > 0 \right\}$$

For this value of ε the solution given by (2.9) is feasible and has at most $p - 1$ positive variables. Repeating this process as necessary, we can eliminate positive variables until we have a solution with corresponding columns that are linearly independent. The resultant situation reduces to case 1. ■

Proof of (ii): Let $\mathbf{x} = (x_1, x_2, \dots, x_n)$ be an optimal feasible solution, and, as in the proof of (i), suppose there are exactly p positive variables x_1, x_2, \dots, x_p . Again there are two cases with case 1 corresponding to linear independence exactly as before.

Case 2 is also the same except that we now must show that for any ε the solution (2.9) is optimal. To see this, note that the value of the solution is given by

$$\mathbf{c}\mathbf{x} - \varepsilon \mathbf{c}\mathbf{w} \quad (2.10)$$

For sufficiently small ε positive or negative, $\mathbf{x} - \varepsilon \mathbf{w}$ is a feasible solution to the LP. If $\mathbf{c}\mathbf{w} \neq 0$, an ε of small magnitude and proper sign could be determined to render (2.10) smaller than $\mathbf{c}\mathbf{x}$ while maintaining feasibility. This would violate the assumption that \mathbf{x} is optimal, implying $\mathbf{c}\mathbf{w} = 0$. This establishes that the new feasible solution with fewer positive components is also optimal. The remainder of the proof can be completed exactly as in part (i). ■

This theorem reduces the task of solving LPs to that of searching over basic feasible solutions. For a problem having n variables and m constraints, there are at most

$n!/m!(n-m)!$ basic solutions corresponding to the number of ways of selecting m of n columns. Although it would be extremely inefficient to examine each combination, by expanding on the technique used to prove Theorem 2.1.1, the simplex algorithm can be derived. Before getting to the specifics, we note that the proof given above is of a simple algebraic character. In the next section the geometric interpretation of the theorem is explored in terms of the general theory of convex sets.

2.1.3 Convex Properties

Thus far in the development of linear programming, we have focused the discussion on common properties of systems of linear equations. A second approach, leading to an alternative derivation of the fundamental theorem and perhaps a clearer geometric understanding of the result, can be pursued in terms of the theory of convex sets. The principal link between the algebraic and geometric theories is the formal relation between basic feasible solutions of linear equalities in standard form and extreme points of polytopes.

Definition 2.1.4 A point \mathbf{x} in the convex set $C \subset R^n$ is said to be an *extreme point* of C if there are no two distinct points \mathbf{x}^1 and \mathbf{x}^2 in C such that $\mathbf{x} = \alpha\mathbf{x}^1 + (1 - \alpha)\mathbf{x}^2$ for some $\alpha \in (0, 1)$.

An extreme point is thus a point that does not lie strictly within the line segment connecting two other points in the set. The extreme points of a triangle, for example, are its three vertices; every point on the circumference of a circle is an extreme point.

Theorem 2.1.2 The set of all feasible solutions to the linear programming problem is a convex set.

Proof: For the trivial case where the feasible region S is a singleton, the theorem is of course true. For the more general case, we need to show that every convex combination of any two feasible solutions is also feasible. Assume that there are at least two solutions \mathbf{x}^1 and \mathbf{x}^2 with

$$\mathbf{A}\mathbf{x}^1 = \mathbf{b}, \mathbf{x}^1 \geq \mathbf{0} \text{ and } \mathbf{A}\mathbf{x}^2 = \mathbf{b}, \mathbf{x}^2 \geq \mathbf{0}$$

For $0 \leq \alpha \leq 1$, let $\mathbf{x} = \alpha\mathbf{x}^1 + (1 - \alpha)\mathbf{x}^2$ be any convex combination of \mathbf{x}^1 and \mathbf{x}^2 . We note that all elements of the vector \mathbf{x} are nonnegative; i.e., $\mathbf{x} \geq \mathbf{0}$. Substituting for \mathbf{x} in the linear equalities gives

$$\mathbf{A}\mathbf{x} = \mathbf{A}[\alpha\mathbf{x}^1 + (1 - \alpha)\mathbf{x}^2] = \alpha\mathbf{A}\mathbf{x}^1 + (1 - \alpha)\mathbf{A}\mathbf{x}^2 = \alpha\mathbf{b} + (1 - \alpha)\mathbf{b} = \mathbf{b}$$

which shows that \mathbf{x} is feasible. ■

As before, we shall denote the convex set of solutions to the linear programming problem by S . Because S is determined by the intersection of a finite number of linear

equalities (2.1b) and inequalities (2.1c), its boundary (assuming S is nonempty) will consist of sections of the corresponding hyperplanes. If S is bounded and nonempty, it is called a convex polyhedron and the LP will have a finite solution. If S is unbounded, it is referred to more generally as a polytope, and the LP may or may not have a finite solution. By Theorem 2.1.2, if a problem has more than one solution, in reality, it has an infinite number of solutions.

Before proceeding with the developments, we note that if S is a *convex polyhedron* then it is equivalently the convex hull of its extreme points. That is, every feasible solution in S can be represented as a convex combination of the extreme feasible solutions in S . This has implications for solving large-scale linear models. (By definition, a convex polyhedron has a finite number of extreme points.) An unbounded S also has a finite number of extreme points, but not all points in S can be represented as a convex combination of these extreme points. More will be said about this in Section 2.3.3. For the moment, we will assume that S is nonempty and bounded. As will be shown in later sections of this chapter, computational procedures are available to determine whether S is empty or whether the problem has an unbounded minimum.

From the above discussion, one might surmise that extreme points play an important role in solving linear programs. We prove this in the following theorem.

Theorem 2.1.3 The objective function (2.1a) assumes its minimum at an extreme point of the convex polyhedron S generated by the set of feasible solutions to the linear program. If it assumes its minimum at more than one extreme point, then it takes on the same value for every convex combination of those particular points.

Proof: The assumption that S is a convex polyhedron means that it has a finite number of extreme points. Let us denote the objective function by $f(\mathbf{x})$, the extreme points by $\hat{\mathbf{x}}^1, \hat{\mathbf{x}}^2, \dots, \hat{\mathbf{x}}^s$, and the optimal solution by \mathbf{x}^* . This means that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for $\mathbf{x} \in S$. If \mathbf{x}^* is an extreme point the first part of the theorem is true; if \mathbf{x}^* is not an extreme point then we can write \mathbf{x}^* as a convex combination of the s extreme points in S : $\mathbf{x}^* = \sum_{i=1}^s \alpha_i \hat{\mathbf{x}}^i$, for $\alpha_i \geq 0$ and $\sum_{i=1}^s \alpha_i = 1$. Then, noting that $f(\mathbf{x})$ is a linear functional, we have

$$\begin{aligned} f(\mathbf{x}^*) &= f\left(\sum_{i=1}^s \alpha_i \hat{\mathbf{x}}^i\right) = f(\alpha_1 \hat{\mathbf{x}}^1 + \alpha_2 \hat{\mathbf{x}}^2 + \cdots + \alpha_s \hat{\mathbf{x}}^s) \\ &= \alpha_1 f(\hat{\mathbf{x}}^1) + \alpha_2 f(\hat{\mathbf{x}}^2) + \cdots + \alpha_s f(\hat{\mathbf{x}}^s) \\ &\geq \alpha_1 f(\hat{\mathbf{x}}^m) + \alpha_2 f(\hat{\mathbf{x}}^m) + \cdots + \alpha_s f(\hat{\mathbf{x}}^m) \\ &= f(\hat{\mathbf{x}}^m) = f^* \end{aligned}$$

where $f(\hat{\mathbf{x}}^m) = \min\{f(\hat{\mathbf{x}}^i) : i = 1, \dots, s\}$. Because we assumed $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in S$, we must have $f(\mathbf{x}^*) = f(\hat{\mathbf{x}}^m) = f^*$. Therefore, there is an extreme point $\hat{\mathbf{x}}^m$ at which the objective function assumes its minimum value.

To prove the second part of the theorem, let $f(\mathbf{x})$ assume its minimum at more than one extreme point, say at $\hat{\mathbf{x}}^1, \hat{\mathbf{x}}^2, \dots, \hat{\mathbf{x}}^q$. This means that $f(\hat{\mathbf{x}}^1) = f(\hat{\mathbf{x}}^2) = \dots = f(\hat{\mathbf{x}}^q) = f^*$. Now let \mathbf{x} be any convex combination of the above $\hat{\mathbf{x}}^i$:

$$\mathbf{x} = \sum_{i=1}^q \alpha_i \hat{\mathbf{x}}^i \text{ for } \alpha_i \geq 0 \text{ and } \sum_i \alpha_i = 1$$

Then

$$\begin{aligned} f(\mathbf{x}^*) &= f(\alpha_1 \hat{\mathbf{x}}^1 + \alpha_2 \hat{\mathbf{x}}^2 + \dots + \alpha_q \hat{\mathbf{x}}^q) \\ &= \alpha_1 f(\hat{\mathbf{x}}^1) + \alpha_2 f(\hat{\mathbf{x}}^2) + \dots + \alpha_q f(\hat{\mathbf{x}}^q) = \sum_i \alpha_i f^* = f^* \end{aligned}$$

which establishes the result. ■

Recall that a feasible solution is a vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$ with all $x_j \geq 0$ such that

$$\mathbf{a}_1 x_1 + \mathbf{a}_2 x_2 + \dots + \mathbf{a}_n x_n = \mathbf{b}$$

Assume that we have found a set of k vectors that is linearly independent and that there exists a nonnegative combination of these vectors equal to \mathbf{b} . Let this set of (column) vectors be $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$. We then have the following theorem.

Theorem 2.1.4 If a set of $k \leq m$ vectors, $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$ can be found that is linearly independent such that

$$\mathbf{a}_1 x_1 + \mathbf{a}_2 x_2 + \dots + \mathbf{a}_k x_k = \mathbf{b}$$

and all $x_j \geq 0$, then the point $\mathbf{x} = (x_1, x_2, \dots, x_k, 0, \dots, 0)$ is an extreme point of the convex set S of feasible solutions. Here \mathbf{x} is an n -dimensional vector whose last $n - k$ components are zero.

Proof: If \mathbf{x} is not an extreme point then it can be written as a convex combination of two other points \mathbf{x}^1 and \mathbf{x}^2 in S . This means $\mathbf{x} = \alpha \mathbf{x}^1 + (1 - \alpha) \mathbf{x}^2$ for $0 < \alpha < 1$. Because all the components x_j of \mathbf{x} are nonnegative and $\alpha \in (0, 1)$, the last $n - k$ components of \mathbf{x}^1 and \mathbf{x}^2 must also equal zero; that is,

$$\mathbf{x}^1 = (x_1^{(1)}, x_2^{(1)}, \dots, x_k^{(1)}, 0, \dots, 0) \text{ and } \mathbf{x}^2 = (x_1^{(2)}, x_2^{(2)}, \dots, x_k^{(2)}, 0, \dots, 0)$$

Given \mathbf{x}^1 and \mathbf{x}^2 are feasible, we have $\mathbf{A}\mathbf{x}^1 = \mathbf{b}$ and $\mathbf{A}\mathbf{x}^2 = \mathbf{b}$, or more explicitly

$$\mathbf{a}_1 x_1^{(1)} + \mathbf{a}_2 x_2^{(1)} + \dots + \mathbf{a}_k x_k^{(1)} = \mathbf{b} \text{ and } \mathbf{a}_1 x_1^{(2)} + \mathbf{a}_2 x_2^{(2)} + \dots + \mathbf{a}_k x_k^{(2)} = \mathbf{b}$$

But $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k\}$ is a linearly independent set and from linear algebra we know that \mathbf{b} can be expressed as a unique linear combination of the elements $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$. This implies that $x_j = x_j^{(1)} = x_j^{(2)}$. Therefore, \mathbf{x} cannot be expressed as a convex combination of two distinct points in S and so must be an extreme point. ■

Theorem 2.1.5 If $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is an extreme point of S , then the vectors associated with positive x_j form a linearly independent set. It follows that at most m of the x_j are positive.

Proof: The proof is by contradiction. Let the first k components of \mathbf{x} be nonzero so that $\sum_{j=1}^k \mathbf{a}_j x_j = \mathbf{b}$. Assume that $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k\}$ is a linear dependent set. Then there exists a linear combination of these vectors that equal the zero vector,

$$\mathbf{a}_1 d_1 + \mathbf{a}_2 d_2 + \cdots + \mathbf{a}_k d_k = \mathbf{0} \quad (2.11)$$

with at least one $d_j \neq 0$. From the hypothesis of the theorem we have

$$\mathbf{a}_1 x_1 + \mathbf{a}_2 x_2 + \cdots + \mathbf{a}_k x_k = \mathbf{b} \quad (2.12)$$

For some $\varepsilon > 0$, we multiply (2.11) by ε and add and subtract the result from (2.12) to obtain the two equations

$$\sum_{j=1}^k \mathbf{a}_j x_j + \varepsilon \sum_{j=1}^k \mathbf{a}_j d_j = \mathbf{b} \quad \text{and} \quad \sum_{j=1}^k \mathbf{a}_j x_j - \varepsilon \sum_{j=1}^k \mathbf{a}_j d_j = \mathbf{b}$$

This gives two solutions to $\mathbf{Ax} = \mathbf{b}$ which may or may not satisfy the nonnegativity constraint $\mathbf{x} \geq \mathbf{0}$:

$$\mathbf{x}^1 = (x_1 + \varepsilon d_1, x_2 + \varepsilon d_2, \dots, x_k + \varepsilon d_k) \text{ and } \mathbf{x}^2 = (x_1 - \varepsilon d_1, x_2 - \varepsilon d_2, \dots, x_k - \varepsilon d_k)$$

But because $x_j > 0$, we can make ε as small as necessary to assure that the first k components of both \mathbf{x}^1 and \mathbf{x}^2 are positive. So for an appropriate ε , \mathbf{x}^1 and \mathbf{x}^2 are feasible solutions. But $\mathbf{x} = \frac{1}{2}\mathbf{x}^1 + \frac{1}{2}\mathbf{x}^2$ which contradicts the hypothesis that \mathbf{x} is an extreme point. This implies that set of vectors $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k\}$ cannot be linearly dependent.

Since every set of $m+1$ vectors in m -dimensional space is necessarily linearly dependent, we cannot have more than m positive x_j . If we did the proof of the main part of the theorem would imply that there exists $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m, \mathbf{a}_{m+1}$ linearly independent vectors. ■

Without loss of generality it can be assumed that the technological matrix \mathbf{A} associated with the linear programming problem always contains a set of m linearly independent vectors. If this property is not evident when a particular problem is being solved, the original set of vectors can be augmented by a set of m linearly independent vectors and the extended problem is solved instead. The details are explained in Section 2.4.1.

Corollary 2.1.1 Associated with every extreme point in S is a set of m linearly independent vectors from the given set $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\}$.

Proof: In Theorem 2.1.5 it was shown that there are $k \leq m$ such vectors. For $k = m$, the corollary is proved. Assume that $k < m$ and that we can find only $\mathbf{a}_{k+1}, \mathbf{a}_{k+2}, \dots, \mathbf{a}_r$ additional vectors such that the set $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k, \mathbf{a}_{k+1}, \dots, \mathbf{a}_r\}$ for $r < m$ is linearly independent. This implies that the remaining $n - r$ vectors are dependent on $\mathbf{a}_1, \dots, \mathbf{a}_r$. But this contradicts the assumption that we always have a set of m linearly independent vectors in the given set $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n$. As a consequence, there must be m linearly independent vectors $\mathbf{a}_1, \dots, \mathbf{a}_m$ associated with every extreme point such that

$$\sum_{j=1}^k \mathbf{a}_j x_j + \sum_{j=k+1}^m \mathbf{a}_j 0 = \mathbf{b}$$

This completes the proof. ■

Summarizing the results of this subsection, we have:

1. There is an extreme point of S at which the objective function takes on its minimum.
2. Every basic feasible solution corresponds to an extreme point of S ; i.e., there is a unique mapping of basic feasible solutions to extreme points.
3. Every extreme point of S has m linearly independent vectors taken from the columns of \mathbf{A} associated with it (the mapping of extreme points to bases may not be unique).

2.2 SIMPLEX METHOD

The simplex method for solving an LP in standard form generates a sequence of feasible points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots$ that terminates at an optimal solution under the assumption of nondegeneracy and boundedness. Because there exists an extreme point at which the solution occurs, the algorithm is designed so that each iterate $\mathbf{x}^{(k)}$ is an extreme point. Thus $n - m$ variables have zero value at $\mathbf{x}^{(k)}$ and are termed nonbasic variables. The remaining m variables take on nonnegative values (positive values under the nondegeneracy assumption) and are called basic variables. The simplex method makes systematic changes to these sets after each iteration to find the combination that gives the optimal solution. Each change is known as a pivot and is taken up next.

2.2.1 Pivoting

To obtain a firm grasp of the simplex procedure, it is essential that one first understands the process of pivoting in a set of simultaneous linear equations. Denoting the columns of \mathbf{A} in (2.1b) by $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n$, we get

$$\mathbf{a}_1 x_1 + \mathbf{a}_2 x_2 + \cdots + \mathbf{a}_n x_n = \mathbf{b} \quad (2.13)$$

which expresses \mathbf{b} as a linear combination of the \mathbf{a}_j vectors. If $m < n$ and the vectors \mathbf{a}_j span R^n then there is not a unique solution but a whole family of solutions. The vector \mathbf{b} has a unique representation, however, as a linear combination of a given linearly independent subset of these vectors. The corresponding solution with $n - m$ x_j variables set to zero is a basic solution to (2.13).

Suppose now that we assume that the n variables are permuted so that the basic variables are the first m components of \mathbf{x} . Then we can write $\mathbf{x} = (\mathbf{x}_B, \mathbf{x}_N)$, where \mathbf{x}_B and \mathbf{x}_N refer to the basic and nonbasic variables, respectively. The matrix \mathbf{A} can also be partitioned similarly into $\mathbf{A} = [\mathbf{B}, \mathbf{N}]$, where \mathbf{B} is the $m \times m$ basis matrix and \mathbf{N} is $m \times (n - m)$. The equations $\mathbf{Ax} = \mathbf{b}$ can thus be written

$$[\mathbf{B}, \mathbf{N}] \begin{pmatrix} \mathbf{x}_B \\ \mathbf{x}_N \end{pmatrix} = \mathbf{B}\mathbf{x}_B + \mathbf{N}\mathbf{x}_N = \mathbf{b} \quad (2.14)$$

Multiplying through by \mathbf{B}^{-1} gives

$$\mathbf{x}_B + \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N = \mathbf{B}^{-1}\mathbf{b} \quad (2.15)$$

which can be written in detached coefficient form giving rise to the following *canonical tableau*

$$\begin{array}{ccccccccc} 1 & 0 & \cdots & 0 & \alpha_{1,m+1} & \alpha_{1,m+2} & \cdots & \alpha_{1n} & \bar{b}_1 \\ 0 & 1 & \cdot & 0 & \alpha_{2,m+1} & \alpha_{2,m+2} & \cdot & \alpha_{2n} & \bar{b}_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ 0 & 0 & \cdot & 1 & \alpha_{m,m+1} & \alpha_{m,m+2} & \cdot & \alpha_{mn} & \bar{b}_m \end{array} \quad (2.16)$$

where the $m \times (n - m)$ matrix (α) is equivalently the matrix $\mathbf{B}^{-1}\mathbf{N}$ and the last column $\bar{\mathbf{b}}$ is $\mathbf{B}^{-1}\mathbf{b}$.

The first m vectors in (2.16) form a basis. Consequently, every other vector represented in the tableau can be expressed as a linear combination of these basis vectors by simply reading the coefficients down the corresponding column. For example,

$$\mathbf{a}_j = \alpha_{1j}\mathbf{a}_1 + \alpha_{2j}\mathbf{a}_2 + \cdots + \alpha_{mj}\mathbf{a}_m \quad (2.17)$$

The tableau can be interpreted as giving the representations of the vectors \mathbf{a}_j in terms of the basis; the j th column of the tableau is the representation for the vector \mathbf{a}_j . Moreover, the expression for \mathbf{b} in terms of the basis is given in the last column.

Now consider the operation of replacing one member of the basis by an outside column. Suppose for example that we wish to replace the basis vector \mathbf{a}_p , $1 \leq p \leq m$, by the vector \mathbf{a}_q . Provided that the first m vectors with \mathbf{a}_p replaced by \mathbf{a}_q are linearly independent these vectors constitute a basis and every vector can be expressed as a linear combination of this new basis. To find the new representations of the vectors we must update the tableau. The linear independence condition holds if and only if $\alpha_{pq} \neq 0$.

Any vector \mathbf{a}_j can be written in terms of the old array using (2.17); for \mathbf{a}_q we have

$$\mathbf{a}_q = \sum_{\substack{i=1 \\ i \neq p}}^m \alpha_{iq} \mathbf{a}_i + \alpha_{pq} \mathbf{a}_p$$

Solving for \mathbf{a}_p ,

$$\mathbf{a}_p = \frac{1}{\alpha_{pq}} \mathbf{a}_q - \sum_{\substack{i=1 \\ i \neq p}}^m \frac{\alpha_{iq}}{\alpha_{pq}} \mathbf{a}_i \quad (2.18)$$

and substituting (2.18) into (2.17) gives

$$\mathbf{a}_j = \sum_{\substack{i=1 \\ i \neq p}}^m \left(\alpha_{ij} - \frac{\alpha_{iq}}{\alpha_{pq}} \alpha_{pj} \right) \mathbf{a}_i + \frac{\alpha_{pj}}{\alpha_{pq}} \mathbf{a}_q \quad (2.19)$$

Denoting the coefficients of the new tableau by α'_{ij} , we obtain immediately from (2.19)

$$\begin{cases} \alpha'_{ij} = \alpha_{ij} - \frac{\alpha_{iq}}{\alpha_{pq}} \alpha_{pj}, & i \neq p \\ \alpha'_{pj} = \frac{\alpha_{pj}}{\alpha_{pq}} \end{cases}$$

In linear algebra terms, the pivoting operation described above is known as *Gauss-Jordan elimination*.

If a system of equations is not originally given in canonical form, we can put it into this form by adjoining the m unit vectors to the tableau and, starting with these vectors as the basis, successively replace each of them with columns of \mathbf{A} using the pivot operation.

Example 2.2.1 Suppose that we wish to solve the system of simultaneous equations

$$\begin{aligned} x_1 + x_2 - x_3 &= 5 \\ 2x_1 - 3x_2 + x_3 &= 3 \\ -x_1 + 2x_2 - x_3 &= -1 \end{aligned}$$

To obtain an initial basis, we add three variables x_4 , x_5 and x_6 to get

$$\begin{array}{rcl} x_1 + x_2 - x_3 + x_4 & = & 5 \\ 2x_1 - 3x_2 + x_3 + x_5 & = & 3 \\ -x_1 + 2x_2 - x_3 + x_6 & = & -1 \end{array}$$

Now to find the basic solution in terms of the variables x_1 , x_2 and x_3 , we set up the tableau below:

x_1	x_2	x_3	x_4	x_5	x_6	\bar{b}
1	1	-1	1	0	0	5
2	-3	1	0	1	0	3
-1	2	-1	0	0	1	-1

The first pivot element, which is circled, is $\alpha_{11} = 1$ and corresponds to the replacement of x_4 by x_1 as a basic variable. After pivoting we obtain the array

x_1	x_2	x_3	x_4	x_5	x_6	\bar{b}
1	1	-1	1	0	0	5
0	-5	3	-2	1	0	-7
0	3	-2	1	0	1	4

where the next pivot element $\alpha_{22} = -5$ indicates that x_5 is to be replaced by x_2 . Continuing through two additional tableaus gives us the final basic solution $x_1 = 4$, $x_2 = 2$, $x_3 = 1$ with nonbasic variables $x_4 = x_5 = x_6 = 0$.

2.2.2 Determining the Leaving Variable

An arbitrary selection of m linearly independent columns from \mathbf{A} does not guarantee that the corresponding basic variables will satisfy the nonnegativity condition $x \geq 0$. Even if we start with a known feasible basis, the pivot operation, which takes one basis into another will not in general preserve feasibility. Special conditions must be met to maintain feasibility from one iteration to the next. How this is done lies at the heart of the simplex method.

Suppose we have the basic feasible solution (BFS) $\mathbf{x} = (x_1, x_2, \dots, x_m, 0, \dots, 0)$ or the equivalent representation

$$x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 + \cdots + x_m \mathbf{a}_m = \mathbf{b} \quad (2.20)$$

Under the nondegeneracy assumption, $x_i > 0$, $i = 1, \dots, m$. Now let us represent the vector \mathbf{a}_q , $q > m$, in terms of the current basis as follows:

$$\mathbf{a}_q = \alpha_{1q} \mathbf{a}_1 + \alpha_{2q} \mathbf{a}_2 + \cdots + \alpha_{mq} \mathbf{a}_m \quad (2.21)$$

where recall that $\boldsymbol{\alpha}_q = (\alpha_{1q}, \dots, \alpha_{mq})^T$ is the $(q-m+1)$ st column of $\mathbf{B}^{-1} \mathbf{N}$. Multiplying (2.21) by the parameter $\theta \geq 0$ and subtracting it from (2.20) gives

$$(x_1 - \theta \alpha_{1q}) \mathbf{a}_1 + (x_2 - \theta \alpha_{2q}) \mathbf{a}_2 + \cdots + (x_m - \theta \alpha_{mq}) \mathbf{a}_m + \theta \mathbf{a}_q = \mathbf{b} \quad (2.22)$$

For $\theta \geq 0$ we have \mathbf{b} as a linear combination of at most $m+1$ vectors. (In fact, (2.22) is a line in m -space which, we will see in Section 2.3.2, represents an edge of the polytope S .) For $\theta = 0$ we have the original BFS; as θ is increased from zero the

coefficient of \mathbf{a}_q increases, and for small enough θ , (2.22) gives a feasible but nonbasic solution. The coefficients of the other vectors will either increase or decrease linearly as θ increases. If any decrease, to maintain feasibility we may set θ equal to the value corresponding to the first instance where one (or more) of the coefficients vanishes. That is

$$\theta = \min_i \left\{ \frac{x_i}{\alpha_{iq}} \mid \alpha_{iq} > 0 \right\} \quad (2.23)$$

Placing the value of θ found in (2.23) in (2.22) yields a new BFS with the vector \mathbf{a}_q replaced by the vector \mathbf{a}_p , where p corresponds to the minimizing index in (2.23). The calculation of θ is known as the *minimum ratio test*. If the minimum is achieved at more than a single index i , then the new solution is degenerate and any of the vectors with zero coefficient can be regarded as the one that left the basis.

If none of the α_{iq} 's are positive, then all coefficients in (2.22) increase (or remain the same) as θ is increased, and no new basic feasible solution emerges. This means that there are feasible solutions to (2.1b)–(2.1c) that have arbitrarily large coefficients so that the set S is unbounded. As we shall see, this case is of special significance in the simplex method.

2.2.3 Moving toward Optimality

To find an optimal solution we begin with any basic feasible solution and then attempt to find another with a smaller objective function value f . Suppose that we have a BFS given by (2.15) which is equivalent to

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}, \quad \mathbf{x}_N = \mathbf{0} \quad (2.24)$$

Let $\mathbf{c} = (\mathbf{c}_B, \mathbf{c}_N)$, and thus

$$f = \mathbf{c}_B \mathbf{x}_B + \mathbf{c}_N \mathbf{x}_N \quad (2.25)$$

Using (2.15) to eliminate \mathbf{x}_B from (2.25) gives

$$f = \mathbf{c}_B \mathbf{B}^{-1}\mathbf{b} + (\mathbf{c}_N - \mathbf{c}_B \mathbf{B}^{-1}\mathbf{N}) \mathbf{x}_N \quad (2.26)$$

Substituting $\mathbf{x}_N = \mathbf{0}$ in (2.26) yields $f = \mathbf{c}_B \mathbf{B}^{-1}\mathbf{b}$ as the value of the basic solution given by (2.24). For convenience, let $f^0 \triangleq \mathbf{c}_B \mathbf{B}^{-1}\mathbf{b}$, $\mathbf{x}_B \triangleq (x_{B_1}, \dots, x_{B_m})$, and Q be the index set of columns in \mathbf{N} . Then (2.15) and (2.26) can be written as

$$x_{B_i} = \bar{b}_i - \sum_{j \in Q} \alpha_{ij} x_j \quad (2.27a)$$

$$f = f^0 + \sum_{j \in Q} \bar{c}_j x_j \quad (2.27b)$$

where $\bar{c}_j = c_j - \mathbf{c}_B \mathbf{B}^{-1} \mathbf{a}_j$, $j \in Q$; that is, the coefficients of the nonbasic variables in (2.26). These values are known as the *reduced cost coefficients* or the *relative cost*

coefficients. The solution given by (2.24) is obtained by setting $x_j = 0$ for all $j \in Q$ in (2.27).

Assume that \mathbf{x}_B is nondegenerate and $\bar{c}_j < 0$ for some $j \in Q$, say $j = q$. Then by increasing x_q and keeping all other nonbasic variables fixed at zero, f decreases linearly with slope \bar{c}_q . Also, x_{B_i} is a linear function of x_q with slope $-\alpha_{iq}$. As was shown in Section 2.2.2, if $\alpha_{iq} > 0$, then $x_{B_i} \geq 0$ as long as $x_q \leq \bar{b}_i/\alpha_{iq} \triangleq \theta_{iq}$. When $x_q = \theta_{iq}$, $x_{B_i} = 0$.

Let x_{B_p} be any basic variable with

$$0 < \theta_{pq} = \min_{i=1,\dots,m} \{\theta_{iq} : \alpha_{iq} > 0\}$$

Then by keeping all nonbasic variables equal to zero except x_q , letting $\theta \triangleq \theta_{pq}$, and setting

$$\begin{aligned} x_q &= \theta \\ x_{B_i} &= \bar{b}_i - \alpha_{iq}\theta, \quad i = 1, \dots, m \end{aligned}$$

a new BFS is obtained with $x_q > 0$, $x_{B_p} = 0$, and $f = f^0 + \bar{c}_q\theta$. Because $\theta > 0$ and $\bar{c}_q < 0$, f is decreased. Observe that if there is no q such that $\alpha_{iq} > 0$, then x_q can be made arbitrarily large while the basic variables remain nonnegative. From (2.27b) we see that the objective function f can thus be made arbitrarily small implying that the problem is unbounded.

Finding the new BFS corresponds to solving the p th equation of (2.27a) for x_q , eliminating x_q from the other equations for $i \neq p$, and then setting $x_{B_p} = 0$ and $x_j = 0$ for all $j \in Q \setminus \{q\}$. Thus whenever there is a nondegenerate BFS with $\bar{c}_j < 0$ for some $j \in Q$, say $j = q$, and $\alpha_{iq} > 0$ for at least one i , a better BFS can be found by exchanging one column of N for one of \mathbf{B} . This result is summarized in the following theorem.

Theorem 2.2.1 Given a nondegenerate BFS with corresponding objective function value f^0 , suppose that for some j there holds $\bar{c}_j < 0$ for $j \in Q$. Then there is a feasible solution with objective function value $f < f^0$. If the column \mathbf{a}_j can be substituted for some vector in the original basis to yield a new BFS, this new solution will have $f < f^0$. If no \mathbf{a}_j can be found, then the solution set S is unbounded and the objective function can be made arbitrarily small (toward minus infinity).

On the other hand, suppose that \mathbf{B} is a basis matrix which yields vector \mathbf{x}^* with $\bar{c}_j \geq 0$ for all $j \in Q$ and corresponding f^* . From (2.27b) the objective function can be written as

$$f^* = \mathbf{c}_B \mathbf{B}^{-1} \mathbf{b} + \sum_{j \in Q} \bar{c}_j x_j^* \tag{2.28}$$

which does not contain \mathbf{x}_B . From the fact that $\bar{c}_j \geq 0$ and $x_j^* \geq 0$ for all $j \in Q$, we can conclude that $\mathbf{c}_B \mathbf{B}^{-1} \mathbf{b}$ is a lower bound on f . Because $\mathbf{x}_B^* = \mathbf{B}^{-1} \mathbf{b}$, $\mathbf{x}_N^* = \mathbf{0}$ is a feasible and achieves this lower bound, it is optimal. Thus we have proved

Theorem 2.2.2 The basic solution given in (2.27a) is optimal if

- i) $x_{B_i} \geq 0$, $i = 1, \dots, m$ (feasibility)
- ii) $\bar{c}_j \geq 0$, for all $j \in Q$.

Finding improved basic feasible solutions, as described above, is essentially the simplex algorithm. Extensive experience has shown that starting with an initial basis, an optimal solution can be found in about m , or perhaps $3m/2$ pivot operations. As such, if m is much smaller than n , that is, the matrix \mathbf{A} has far fewer rows than columns, pivots will occur in only a small fraction of the columns during the course of optimization. To take advantage of this fact, the revised form of the simplex method was developed and will be described below. It starts with the inverse \mathbf{B}^{-1} of the current basis, and the current solution $\mathbf{x}_B = \bar{\mathbf{b}} = \mathbf{B}^{-1} \mathbf{b}$.

Algorithm

- Step 1 (Optimality test) Calculate the current reduced cost coefficients $\bar{\mathbf{c}} = \mathbf{c}_N - \mathbf{c}_B \mathbf{B}^{-1} N$. This can be done most efficiently by first calculating the vector $\lambda^T = \mathbf{c}_B \mathbf{B}^{-1}$ and then the reduced cost vector $\bar{\mathbf{c}} = \mathbf{c}_N - \lambda^T N$. If $\bar{\mathbf{c}} \geq \mathbf{0}$ stop, the current solution is optimal.
- Step 2 (Entering variable) Determine which vector \mathbf{a}_q is to enter the basis by selecting the most negative cost coefficient; calculate $\alpha_q = \mathbf{B}^{-1} \mathbf{a}_q$ which gives the vector \mathbf{a}_q in terms of the current basis.
- Step 3 (Leaving variable) If no $\alpha_{iq} > 0$, stop; the problem is unbounded. Otherwise, calculate the ratios \bar{b}_i / α_{iq} for $\alpha_{iq} > 0$ to determine which vector is to leave the basis. Break ties arbitrarily.
- Step 4 (Pivoting) Update \mathbf{B}^{-1} and the current solution and $\mathbf{B}^{-1} \mathbf{b}$. Return to Step 1.

The computations in Step 1 where the new set of reduced cost coefficients is found are known as the *pricing out* operation. In Step 4, the simplest way to update \mathbf{B}^{-1} is by applying the usual pivot operations to an array consisting of \mathbf{B}^{-1} and α_q , where the pivot element is the appropriate component of α_q . Of course $\mathbf{B}^{-1} \mathbf{b}$ can be updated at the same time by adjoining it as another column.

To begin the procedure, one needs an initial basic feasible solution and the inverse of the accompanying basis. In most problems, the initial basis is taken to be the identity matrix of appropriate dimension resulting from the adjoining of slack, surplus or artificial variables to the original formulation (artificial variables are discussed in Section 2.4.1). In practice, more elaborate procedures are used in Step 2 to determine

the entering variable and to take into account the possibility of degeneracy, and in Step 4 to keep track of \mathbf{B}^{-1} (see [B25] and [S17], respectively).

Example 2.2.2 To illustrate the computations of the revised simplex method we will solve the following problem

$$\begin{array}{ll} \min & -3x_1 - x_2 - 3x_3 \\ \text{subject to} & 2x_1 + x_2 + x_3 \leq 2 \\ & x_1 + 2x_2 + 3x_3 \leq 5 \\ & 2x_1 + 2x_2 + x_3 \leq 6 \\ & x_1, x_2, x_3 \geq 0 \end{array}$$

To put the constraints into standard form, we introduce three slack variables x_4 , x_5 and x_6 which leads to the following tableau

x_1	x_2	x_3	x_4	x_5	x_6	b
2	1	1	1	0	0	2
1	2	3	0	1	0	5
2	2	1	0	0	1	6

with objective function coefficients $\mathbf{c} = (-3, -1, -3, 0, 0, 0)$. We start with an initial BFS and corresponding \mathbf{B}^{-1} as shown in the inverse tableau below

Basic variable	\mathbf{B}^{-1}			\mathbf{x}_B
4	1	0	0	2
5	0	1	0	5
6	0	0	1	6

Beginning at Step 1, we compute $\boldsymbol{\lambda}^T = (0, 0, 0)\mathbf{B}^{-1} = (0, 0, 0)$ and then $\bar{\mathbf{c}} = \mathbf{c}_N - \boldsymbol{\lambda}^T \mathbf{N} = (-3, -1, -3)$. At Step 2, rather than picking the most negative reduced cost coefficient, we decide to bring x_2 into the basis (this simplifies the calculations and demonstrates that any $\bar{c}_j < 0$ is acceptable). Its current representation is found by multiplying by $\mathbf{B}^{-1} = I_3$, giving

Basic variable	\mathbf{B}^{-1}			\mathbf{x}_B	α_2
4	1	0	0	2	(1)
5	0	1	0	5	2
6	0	0	1	6	2

After performing the minimum ratio test in the usual manner, we select the pivot element as indicated. The updated inverse tableau becomes

Basic variable	\mathbf{B}^{-1}			\mathbf{x}_B	α_3
2	1	0	0	2	1
5	-2	1	0	1	(1)
6	-2	0	1	2	-1

then $\lambda^T = (-1, 0, 0)\mathbf{B}^{-1} = (-1, 0, 0)$ and $\bar{c}_1 = -1$, $\bar{c}_3 = -2$, $\bar{c}_4 = 1$. We select a_3 to enter. The updated α_3 is given in the above tableau. Pivoting as indicated, we obtain

Basic variable	\mathbf{B}^{-1}			\mathbf{x}_B	α_1
2	1	-1	0	1	(5)
3	-2	1	0	1	-3
6	-4	1	1	3	-5

which leads to $\lambda^T = (-1, -3, 0)\mathbf{B}^{-1} = (3, -2, 0)$ and $\bar{c}_1 = -7$, $\bar{c}_4 = -3$, $\bar{c}_5 = 2$. We select a_1 to enter and pivot on the indicated element, giving

Basic Variable	\mathbf{B}^{-1}			\mathbf{x}_B
1	3/5	-1/5	0	1/5
3	-1/5	2/5	0	8/5
6	-1	0	1	4

with $\lambda^T = (-3, -3, 0)\mathbf{B}^{-1} = (-6/5, -3/5, 0)$ and $\bar{c}_2 = 7/5$, $\bar{c}_4 = 6/5$, $\bar{c}_5 = 3/5$. Because all the \bar{c}_j are nonnegative, we conclude that the solution $\mathbf{x}^* = (1/5, 0, 8/5, 0, 0, 4)$ is optimal.

2.2.4 Degeneracy and Cycling

A degenerate basic feasible solution occurs with, for some p and q , $\bar{c}_q < 0$, $\bar{b}_p = 0$, and $\alpha_{pq} > 0$. By introducing x_q into the basis, x_{B_p} (or some other basic variable equal to 0) must leave the basis and although a new basis is obtained, the solution and objective function value remain the same. The new basic solution $\mathbf{x}_{B_i} = \bar{b}_i$, $i \neq p$, $x_q = 0$, $x_j = 0$ otherwise, is also degenerate. Examples have been constructed in which a finite sequence of degenerate bases obtained by the simplex algorithm produces a cycle where no progress is made toward the solution; that is, the sequence $\mathbf{B}^1, \dots, \mathbf{B}^{k-1}, \mathbf{B}^k$ occurs with $\mathbf{B}^k = \mathbf{B}^1$.

This unwelcome phenomenon is called *cycling*. A wealth of empirical evidence indicates that cycling does not occur with any regularity in real applications so it mostly

of academic interest for linear programming. Nevertheless, the simplex algorithm can be modified so that it will never occur. The procedure used is of primary importance in some integer programming algorithms where LPs are repeatedly solved as subproblems, and involves specializing Step 3 of the revised simplex method. Recall that in Step 3 a variable is chosen to leave the basis. When there is more than one candidate, an arbitrary choice is made. By giving a special rule for breaking ties, the possibility of cycling can be avoided. Before presenting the rule, some new definitions are introduced.

A vector $\mathbf{v} \neq \mathbf{0}$ is called *lexicographically positive* if its first nonzero component is positive. The notation $\mathbf{v} \succ \mathbf{0}$ is used to denote this situation. For example $(0, 0, 3, -9, 4) \succ \mathbf{0}$. A vector \mathbf{v} is said to be *lexicographically greater* than a vector \mathbf{u} if $\mathbf{v} - \mathbf{u} \succ \mathbf{0}$. A sequence of vectors $\{\mathbf{v}^t\}$ is called *lexicographically increasing* if $\mathbf{v}^{t+1} - \mathbf{v}^t \succ \mathbf{0}$ for all t . If $-\mathbf{v} \succ \mathbf{0}$, \mathbf{v} is called *lexicographically negative* (denoted by $\mathbf{v} \prec \mathbf{0}$) and if $\mathbf{v} - \mathbf{u} \prec \mathbf{0}$, \mathbf{v} is *lexicographically smaller* than \mathbf{u} . The notation $\mathbf{v} \succeq \mathbf{0}$ means that $\mathbf{v} = \mathbf{0}$ or $\mathbf{v} \succ \mathbf{0}$.

The changes required in the simplex algorithm to avoid cycling can best be described using the tableau below which represents an initial BFS.

	x_j								
x_{B_1}	b_1	1	\cdots	0	\cdots	0	\cdots	α_{1j}	\cdots
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
x_{B_i}	\bar{b}_i	0	\cdots	1	\cdots	0	\cdots	α_{ij}	\cdots
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
x_{B_m}	\bar{b}_m	0	\cdots	0	\cdots	1	\cdots	α_{mj}	\cdots
$-f$	f^0	0	\cdots	0	\cdots	0	\cdots	\bar{c}_j	\cdots

In the tableau, an m th order identity matrix is inserted between the solution column and the columns corresponding to the nonbasic variables. The last row contains the objective function. The m unit columns are not used to determine the variable to enter the basis; however, at each iteration, they are transformed by applying the standard pivoting rules given in the simplex method to update \mathbf{B}^{-1} . As will be shown, the only purpose of these columns is to identify the variable to leave the basis when the choice at Step 3 is not unique.

Let $\mathbf{v}^0 = (f^0, 0, \dots, 0)$ and $\mathbf{v}^i = (\bar{b}_i, 0, \dots, 1, \dots, 0)$, $i = 1, \dots, m$, be the coefficients in the first $m + 1$ columns of the i th row of the above tableau. Because $\bar{b}_i \geq 0$, it follows that $\mathbf{v}^i \succ \mathbf{0}$, $i = 1, \dots, m$.

Suppose that $\bar{c}_q < 0$ and x_q is chosen to enter the basis. Let $E(q) = \{i : 1 \leq i \leq m, \alpha_{iq} > 0\}$ and $\mathbf{u}^i = \mathbf{v}^i / \alpha_{iq}$ for all $i \in E(q)$. Assume that \mathbf{u}^p is the lexicographically

smallest of these vectors, that is,

$$\mathbf{u}^p = \operatorname{lexmin}_{i \in E(q)} \mathbf{u}^i \quad (2.29)$$

The vector \mathbf{u}^p must be unique because the \mathbf{v}^i 's are linearly independent. We now choice x_{B_p} to leave the basis. Note that x_{B_p} determined from (2.29) satisfies the minimum ratio rule specified in the simplex algorithm since

$$\mathbf{u}^p = \operatorname{lexmin}_{i \in E(q)} \mathbf{u}^i \implies \frac{\bar{b}_p}{\alpha_{pq}} = \min_{i \in E(q)} \frac{\bar{b}_i}{\alpha_{iq}}$$

Pivoting on row p and column q yields

$$\begin{aligned} \hat{\mathbf{v}}^p &= \frac{\mathbf{v}^p}{\alpha_{pq}} \\ \hat{\mathbf{v}}^i &= \mathbf{v}^i - \frac{\alpha_{iq}}{\alpha_{pq}} \mathbf{v}^p = \mathbf{v}^i - \alpha_{iq} \hat{\mathbf{v}}^p, \quad i \neq p \end{aligned}$$

Since $\mathbf{v}^p \succ \mathbf{0}$ and $\alpha_{pq} > 0$, $\hat{\mathbf{v}}^p \succ \mathbf{0}$. If $\alpha_{iq} \leq 0$, then $-\alpha_{iq} \hat{\mathbf{v}}^p \succeq \mathbf{0}$ and $\hat{\mathbf{v}}^i \succ \mathbf{0}$. If $\alpha_{iq} > 0$, we have

$$\mathbf{u}^i - \mathbf{u}^p = \frac{\mathbf{v}^i}{\alpha_{iq}} - \hat{\mathbf{v}}^p \succ \mathbf{0} \quad (2.30)$$

from (2.29). Multiplying (2.30) by $\alpha_{iq} > 0$ yields $\hat{\mathbf{v}}^i \succ \mathbf{0}$.

Thus $\hat{\mathbf{v}}^i$, $i = 1, \dots, m$, are lexicographically positive as well as linearly independent since adding multiples of one vector to another does not destroy independence. By induction, under the pivot row selection rule given by (2.29), $\hat{\mathbf{v}}^i$, $i = 1, \dots, m$, remains linearly independent and lexicographically positive at each iteration.

At the k th iteration the tableau is completely determined by the basis \mathbf{B}^k . Thus if tableaus k and t are such that $\mathbf{v}_k^0 \neq \mathbf{v}_t^0$, then $\mathbf{B}^k \neq \mathbf{B}^t$. Note that if the sequence $\{\mathbf{v}_t^0\}$ is lexicographically increasing, no distinct t_1 and t_2 exist such that $\mathbf{v}_{t_1}^0 = \mathbf{v}_{t_2}^0$. However,

$$\mathbf{v}_{t+1}^0 = \mathbf{v}_t^0 - \frac{\bar{c}_q^t}{\alpha_{pq}^t} \mathbf{v}_t^p = \mathbf{v}_t^0 - \bar{c}_q^t \mathbf{v}_{t+1}^p$$

so that $\mathbf{v}_{t+1}^p \succ \mathbf{0}$ and $\bar{c}_q^t < 0$ imply $\mathbf{v}_{t+1}^0 \succ \mathbf{v}_t^0$.

Example 2.2.3 Consider the following tableau which indicates that x_5 is the entering variable.

	\bar{b}				x_4	x_5	↓
x_1	0	1	0	0	1	1	
x_2	0	0	1	0	2	4	
\leftarrow	x_3	0	0	0	-1	3	
	$-f$	0	0	0	2	-1	

The ratio test at Step 3 of the simplex algorithm would permit the removal of either x_1 , x_2 or x_3 because $\theta_{i5} = \bar{b}_i/\alpha_{i5} = 0$ for $i = 1, 2, 3$. Nevertheless, $\mathbf{u}^3 = (0, 0, 0, 1/3) \prec \mathbf{u}^2 = (0, 0, 1/4, 0) \prec \mathbf{u}^1 = (0, 1, 0, 0)$ so x_3 is chosen as the departing variable. Observe that in the updated tableau given below, $\hat{\mathbf{v}}^i \succ \mathbf{0}$ and $\hat{\mathbf{v}}^0 - \mathbf{v}^0 \succ \mathbf{0}$. The reader can verify that if x_1 or x_2 were chosen to leave, then $\hat{\mathbf{v}}^3 \prec \mathbf{0}$.

	\bar{b}				x_4	x_3
x_1	0	1	0	-1/3	4/3	-1/3
x_2	0	0	1	-4/3	10/3	-4/3
x_5	0	0	0	1/3	-1/3	1/3
$-f$	0	0	0	1/3	5/3	1/3

In addition to the lexicographic rule for preventing cycling, a number of other techniques have been proposed. These are discussed, for example, by Murty [M18] but are likewise computational inefficient. As a practical matter, cycling occurs only in rare instances so the designers of most commercial LP codes have simply ignored it. A second reason for ignoring cycling relates to computer round-off errors. In this regard, it can be argued that the updated right-hand-side values \bar{b}_i are usually perturbed from their actual values due to round-off errors so one rarely encounters exact zero values for \bar{b}_i . In fact, the lexicographic rule has an equivalent interpretation as a perturbation technique in which the right-hand-side values are modified slightly to make the polytope nondegenerate.

The only real case of practical concern where cycling may present difficulties is in network optimization. For those linear programs that have a network structure, specialized codes are available that provide significant computational advantage over the simplex method. Fortunately, cycling prevention rules are easy to implement in these codes.

The final issue to address concerns a phenomenon related to cycling where the simplex algorithm goes through an exceedingly long (though finite) sequence of degenerate pivots whose number may be exponential in the size (m and n) of the problem. This phenomenon is known as *stalling*. The term arises because with increasing problem size, the algorithm can spend an enormous amount of time at a degenerate vertex before verifying optimality or moving on. Besides preventing cycling one would like to preclude stalling by guaranteeing that the length of a sequence of degenerate pivots

is bounded from above by a polynomial in m and n . This issue is further discussed in [B15].

2.3 GEOMETRY OF SIMPLEX METHOD

There are two ways, in general, to interpret the simplex method geometrically. The first, and perhaps the most natural, is in *activity space* where \mathbf{x} is represented. As discussed in Section 2.1.3, the feasible region is viewed directly as a convex set, and basic feasible solutions are extreme points. Adjacent extreme points are points that lie on a common edge.

The second geometrical interpretation is in *requirements space*, the space where the columns of \mathbf{A} and \mathbf{b} are represented. Equation (2.13) depicts the fundamental relation. An example for $m = 2$ and $n = 4$ is shown in Fig. 2.2. Here a feasible solution is defined by \mathbf{b} lying in a ‘pos cone’ associated with the columns of \mathbf{A} . As described in [M18], a nondegenerate basic feasible solution will use exactly m positive weights. In the figure a basic feasible solution can be constructed with positive weights on \mathbf{a}_1 and \mathbf{a}_2 because \mathbf{b} lies between them. Such a solution cannot be constructed using \mathbf{a}_1 and \mathbf{a}_4 . Suppose we start with \mathbf{a}_1 and \mathbf{a}_2 as the initial basis. Then an adjacent basis is found by bringing in some other vector. If \mathbf{a}_3 is brought in, then clearly \mathbf{a}_2 must leave. On the other hand, if \mathbf{a}_4 is introduced, \mathbf{a}_1 must leave.

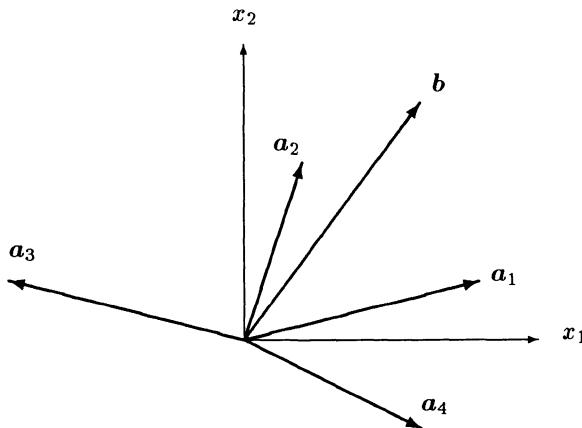


Figure 2.2 Constraint representation in requirements space

2.3.1 Finiteness of Algorithm

Consider again the LP in standard form (2.1) where \mathbf{A} is an $m \times n$ matrix of rank m . If the problem has an optimal feasible solution, then it has an optimal BFS by Theorem 2.1.1. Therefore, it is sufficient to search among the finite number of BFSs of (2.1) for an optimum. This is what the simplex method does.

If (2.1) is a nondegenerate LP, every BFS has exactly m positive components and hence, has a unique associated basis. In the case, when the simplex algorithm is used to solve (2.1), the minimum ratio will turn out to be strictly positive in every pivot step so the objective function will decrease at each iteration. This means that a basis that appears once in the course of the algorithm can never reappear. Because the total number of bases is finite, the algorithm must terminate after a finite number of pivot steps with either an optimal basic vector or by satisfying the unboundedness criterion .

Now suppose that (2.1) is a degenerate LP so that when applying the simplex method, a degenerate pivot may occur. The next pivot step in the algorithm might also turn out to be degenerate leading to cycling. During these pivot steps, the algorithm is just moving among bases, all of which are associated with the same extreme point. Nevertheless, with a pivot row choice rule for resolving degeneracy, such as the lexicographic minimum ratio rule (2.29), it is possible to avoid returning to a previously encountered basis. This leads to the following theorem.

Theorem 2.3.1 Starting with a primal feasible basis, the simplex algorithm (with some technique for resolving degeneracy) finds, after a finite number of pivots, one of the following:

1. An optimal feasible basis whose corresponding canonical tableau satisfies the optimality criterion.
2. A primal feasible basis whose canonical tableau satisfies the unboundedness criterion.

2.3.2 Adjacency and Bounded Edges

In this subsection we see how the simplex algorithm walks from one feasible vertex in S to another before a termination criterion is satisfied. Consider the convex polytope depicted in Fig. 2.3. Evidently every point on the line segment joining the extreme points \mathbf{x}^1 and \mathbf{x}^2 cannot be expressed as a convex combination of any pair of points in the polytope that are not on this line segment. This, however, is not true of points on the line segment joining the extreme points \mathbf{x}^1 and \mathbf{x}^3 . Extreme points \mathbf{x}^1 and \mathbf{x}^2 are known as adjacent extreme points of the convex polytope. The extreme points \mathbf{x}^1 and \mathbf{x}^3 are not adjacent.

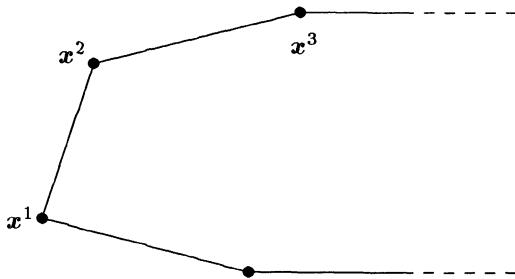


Figure 2.3 Adjacent edges of convex polytope

Definition 2.3.1 (*Geometric characterization of adjacency*) Two extreme points $\tilde{\mathbf{x}}$ and $\hat{\mathbf{x}}$ of a convex polytope S are said to be adjacent iff every point $\bar{\mathbf{x}}$ on the line segment joining them has the property that if $\bar{\mathbf{x}} = \alpha\mathbf{x}^1 + (1 - \alpha)\mathbf{x}^2$, where $0 < \alpha < 1$ and $\mathbf{x}^1, \mathbf{x}^2 \in S$, then both \mathbf{x}^1 and \mathbf{x}^2 must themselves be on the line segment joining $\tilde{\mathbf{x}}$ and $\hat{\mathbf{x}}$. The line segment joining a pair of adjacent extreme points is called a *bounded edge* or *edge* of the convex polytope.

Definition 2.3.2 (*Algebraic characterization of adjacency*) When dealing with the convex polytope S associated with an LP, there is a simple characterization of adjacency in terms of the rank of a set of vectors. That is, two distinct extreme points of S , $\tilde{\mathbf{x}}$ and $\hat{\mathbf{x}}$, are adjacent iff the rank of the set $\{\mathbf{a}_j : j \text{ such that either } \tilde{x}_j \text{ or } \hat{x}_j \text{ or both are } > 0\}$ is one less than its cardinality. In other words, this set is linearly dependent and contains a column vector such that when this column vector is deleted from the set, the remaining set is linearly independent.

Theorem 2.3.2 Let S be the set of feasible solutions of (2.1). Let $\tilde{\mathbf{x}}$ and $\hat{\mathbf{x}}$ be two distinct BFSs of S , and $J = \{j : j \text{ such that either } \tilde{x}_j \text{ or } \hat{x}_j \text{ or both are } > 0\}$. Let $L = \{\mathbf{x}(\alpha) = \alpha\tilde{\mathbf{x}} + (1 - \alpha)\hat{\mathbf{x}}, 0 \leq \alpha \leq 1\}$. The following statements are equivalent:

1. If a point in L can be expressed as a convex combination of $\mathbf{x}^1, \mathbf{x}^2 \in S$, both \mathbf{x}^1 and \mathbf{x}^2 are themselves in L .
2. The rank of $\{\mathbf{a}_j : j \in J\}$ is one less than its cardinality.

The algebraic characterization of adjacency provides a convenient method for checking whether a given pair of BFSs of (2.1) is adjacent. Now suppose $\mathbf{B} = (\mathbf{a}_1, \dots, \mathbf{a}_m)$ is the current basis with corresponding BFS $\tilde{\mathbf{x}}$. Therefore, $\tilde{x}_{m+1} = \tilde{x}_{m+2} = \dots = \tilde{x}_n = 0$. Let the entering variable in this step of the simplex algorithm be \tilde{x}_{m+1} with pivot column $\boldsymbol{\alpha}_{m+1} = \mathbf{B}^{-1}\mathbf{a}_{m+1} = (\alpha_{1,m+1}, \dots, \alpha_{m,m+1})^T$. We assume that $\boldsymbol{\alpha}_{m+1} \not\leq \mathbf{0}$. The case where $\boldsymbol{\alpha}_{m+1} \leq \mathbf{0}$ corresponds to unboundedness and is taken up in Section 2.3.3.

During the pivot step, the remaining nonbasic variables x_{m+2}, \dots, x_n are fixed at 0. Only the nonbasic variable x_{m+1} is changed from its present value of 0 to a

nonnegative δ and the values of the basic variables are modified to assure that the equality constraints are satisfied. From the discussion in Section 2.2.3 this leads to the solution $\mathbf{x}(\delta) = (\tilde{x}_1 - \alpha_{1,m+1}\delta, \dots, \tilde{x}_m - \alpha_{m,m+1}\delta, \delta, 0, \dots, 0)$. The maximum value that δ can take in this pivot step is the *minimum ratio* θ which is determined to assure that $\mathbf{x}(\delta)$ remains nonnegative. If the minimum ratio θ is zero, the new BFS is $\tilde{\mathbf{x}}$ itself. In this case the pivot step is degenerate and we have the following result.

1. In a degenerate pivot step, the simplex algorithm remains at the same BFS, but it obtains a new basis associated with it.

If the minimum ratio θ in this pivot step is strictly positive, the simplex algorithm moves to the new BFS $\mathbf{x}(\theta)$, which is distinct from $\tilde{\mathbf{x}}$. Clearly, the line segment joining $\tilde{\mathbf{x}}$ and $\mathbf{x}(\theta)$ is itself generated by varying δ in $\mathbf{x}(\delta)$ from 0 to θ . If $\tilde{\mathbf{x}}$ is a point on this line segment then $\bar{x}_{m+2} = \dots = \bar{x}_n = 0$. From Definition 2.3.2 we can then argue that if $\bar{\mathbf{x}} = \alpha\mathbf{x}^1 + (1 - \alpha)\mathbf{x}^2$, where $0 < \alpha < 1$ and $\mathbf{x}^1, \mathbf{x}^2$ are feasible to (2.1) both \mathbf{x}^1 and \mathbf{x}^2 must be points of the form $\mathbf{x}(\delta)$. This implies that $\tilde{\mathbf{x}}$ and $\mathbf{x}(\theta)$ are themselves adjacent extreme points of S which, as a direct consequence of Theorem 2.3.2, leads to the following result.

2. During a nondegenerate pivot, the simplex algorithm moves from one extreme point $\tilde{\mathbf{x}}$ of the feasible set S to an adjacent extreme point $\mathbf{x}(\theta)$. The edge joining the extreme points is generated by giving the entering variable, call it \tilde{x}_{m+1} , all possible values between 0 and the minimum ratio θ . The equation of the corresponding edge is given by

$$\begin{aligned} x_i &= \tilde{x}_i - \alpha_{i,m+1}\delta, \quad i = 1, \dots, m \\ x_{m+1} &= \delta \end{aligned}$$

for $0 \leq \delta \leq \theta$.

2.3.3 Unboundedness

Although most real problems are bounded, when using decomposition methods to solve large-scale applications, the subproblems that result are often unbounded. Therefore, it is worth investigating the geometry associated with this condition. We do so from the point of view presented in Chapter 3 of [M18]. The omitted proofs can be found therein.

Consider the system of equations in nonnegative variables in (2.1b). A homogeneous solution corresponding to this system is a vector $\mathbf{y} \in R^n$ satisfying

$$\mathbf{A}\mathbf{y} = \mathbf{0}, \quad \mathbf{y} \geq \mathbf{0}$$

The set of all homogeneous solutions is a convex polyhedral cone. If $\tilde{\mathbf{x}}$ is feasible to (2.1) and $\tilde{\mathbf{y}}$ is a corresponding homogeneous solution, then $\tilde{\mathbf{x}} + \delta\tilde{\mathbf{y}}$ is also a feasible solution to (2.1) for any $\delta \geq 0$.

Theorem 2.3.3 (Resolution Theorem I) Every feasible solution of (2.1) can be expressed as the sum of (i) a convex combination of BFSs of (2.1) and (ii) a homogeneous solution corresponding to this system.

Proof: Suppose $\tilde{\mathbf{x}}$ is a feasible solution of (2.1). Let $\{j : \tilde{x}_j > 0\} = \{j_1, \dots, j_k\}$ which means that the set of column vectors of \mathbf{A} used by $\tilde{\mathbf{x}}$ is $\{\mathbf{a}_{j_1}, \dots, \mathbf{a}_{j_k}\}$. The proof is based upon induction on the number of k , the number of columns vectors in this set.

Case 1: Suppose $k = 0$. This can only happen when $\tilde{\mathbf{x}} = \mathbf{0}$, and since $\tilde{\mathbf{x}}$ is assumed to be feasible, $\mathbf{b} = \mathbf{0}$. Thus $\tilde{\mathbf{x}} = \mathbf{0}$ is itself a BFS of (2.1) and also a homogeneous solution. Therefore

$$\tilde{\mathbf{x}} = \underbrace{\mathbf{0}}_{\text{BFS}} + \underbrace{\mathbf{0}}_{\substack{\text{Homogeneous} \\ \text{solution}}}$$

and the theorem holds.

Case 2: $k \geq 1$.

Induction Hypothesis: Suppose the theorem holds for every feasible solution that uses a set of $k - 1$ or less column vectors of \mathbf{A} .

It must be shown that under this hypothesis, the theorem also holds for $\tilde{\mathbf{x}}$ which uses a set of k column vectors of \mathbf{A} . If $\{\mathbf{a}_{j_1}, \dots, \mathbf{a}_{j_k}\}$ is linearly independent then $\tilde{\mathbf{x}}$ is a BFS of (2.1) so

$$\tilde{\mathbf{x}} = \underbrace{\tilde{\mathbf{x}}}_{\text{BFS}} + \underbrace{\mathbf{0}}_{\substack{\text{Homogeneous} \\ \text{solution}}}$$

and the result follows. For the case where the set $\{\mathbf{a}_{j_1}, \dots, \mathbf{a}_{j_k}\}$ is not linearly independent, the reader is referred to [M18]. ■

Boundedness of Convex Polytopes: If the set of feasible solutions $S = \{\mathbf{x} \in R^n : \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ is nonempty, by Resolution Theorem I, it is a convex polyhedron (i.e., bounded) iff $\mathbf{Ay} = \mathbf{0}, \mathbf{y} \geq \mathbf{0}$ has a unique solution; namely, $\mathbf{y} = \mathbf{0}$.

Extreme Homogeneous Solutions: A homogeneous solution corresponding to (2.1) is called an *extreme homogeneous solution* iff it is a BFS of

$$\begin{aligned} \mathbf{Ay} &= \mathbf{0}, \quad \mathbf{y} \geq \mathbf{0} \\ \sum_{j=1}^n y_j &= 1 \end{aligned} \tag{2.31}$$

Thus there are only a finite number of distinct extreme homogeneous solutions. The summation constraint in (2.31) normalizes the solution and eliminates $\mathbf{0}$ from consideration.

Lemma 2.3.1 Either $\mathbf{0}$ is the unique homogeneous solution or every homogeneous solution corresponding to (2.1) can be expressed as a nonnegative linear combination of extreme homogeneous solutions.

Theorem 2.3.4 (*Resolution Theorem II for Convex Polytopes*) Let S be the set of feasible solutions to (2.1).

1. If $\mathbf{0}$ is the unique homogeneous solution corresponding to (2.1), every feasible solution can be expressed as a convex combination of BFSs of (2.1).
2. If $\mathbf{0}$ is not the only homogeneous solution corresponding to (2.1), every feasible solution is the sum of a convex combination of BFSs of (2.1) and a nonnegative combination of extreme homogeneous solutions corresponding to (2.1).

Proof: Follows from Resolution Theorem I and Lemma 2.3.1. ■

Corollary 2.3.1 If the LP (2.1) has a feasible solution, it has an optimal feasible solution iff $f(\mathbf{y}) \geq 0$ for every homogeneous solution \mathbf{y} corresponding to (2.1).

Corollary 2.3.2 Suppose there exists an extreme homogeneous solution \mathbf{y}^k corresponding to (2.1) such that $c\mathbf{y}^k < 0$. If (2.1) is feasible, $f(\mathbf{x})$ is unbounded below on the set S of feasible solutions.

Corollary 2.3.3 If $f(\mathbf{x})$ is unbounded below on the set of feasible solutions of the LP (2.1), it remains unbounded for all values of the vector \mathbf{b} that preserve feasibility.

Proof: Because $f(\mathbf{x})$ is unbounded below on S , by Corollaries 2.3.1 and 2.3.2 there exists a \mathbf{y}^1 satisfying $A\mathbf{y}^1 = \mathbf{0}$, $c\mathbf{y}^1 < 0$, $\mathbf{y}^1 \geq \mathbf{0}$. Suppose \mathbf{b} is changed to \mathbf{b}^1 ; let \mathbf{x}^1 be a feasible solution to the modified LP. Then $\mathbf{x}^1 + \delta\mathbf{y}^1$ is also feasible for all $\delta \geq 0$, and since $c(\mathbf{x}^1 + \delta\mathbf{y}^1) = c\mathbf{x}^1 + \delta(c\mathbf{y}^1)$ tends to $-\infty$ as δ tends to $+\infty$ (because $c\mathbf{y}^1 < 0$), $f(\mathbf{x})$ is unbounded below as well in this modified problem. ■

Corollary 2.3.4 Unboundedness of the set of feasible solutions: Let S denote the set of feasible solutions of (2.1) and let S^0 denote the set of homogeneous solutions corresponding to this system. The set S is unbounded iff $S \neq \emptyset$ and S^0 contains a nonzero point (i.e., the system (2.31) has a nonzero solution). Conversely, if $S^0 = \{\mathbf{0}\}$, S is bounded.

Proof: Follows from Theorem 2.3.4. ■

Corollary 2.3.5 When (2.1) has a feasible solution, a necessary and sufficient condition for it to have a finite optimal feasible solution is that the optimal objective function value in the LP: $\min \{c\mathbf{x} : A\mathbf{x} = \mathbf{0}, \mathbf{x} \geq \mathbf{0}\}$ is zero.

Proof: Follows from Corollary 2.3.1. ■

2.3.4 Finding Adjacent Extreme Points

For the LP (2.1), let $\tilde{\mathbf{x}}$ be an extreme point of S . In a variety of applications as well as in some algorithms for the linear bilevel programming problem, it may be necessary to do one or more of the following.

1. Generate all adjacent extreme points of $\tilde{\mathbf{x}}$ on S .
2. Generate all adjacent extreme points of $\tilde{\mathbf{x}}$ on S such that $f(\mathbf{x}) \geq f(\tilde{\mathbf{x}})$.
3. Generate all the bounded edges and unbounded edges of S containing $\tilde{\mathbf{x}}$.

When $\tilde{\mathbf{x}}$ is a nondegenerate BFS the computations for these three cases are easy, as discussed below. On the other hand, when $\tilde{\mathbf{x}}$ is a degenerate BFS the amount of work and bookkeeping goes up substantially. For this case, see [M9].

Suppose $\tilde{\mathbf{x}}$ is a nondegenerate BFS. Then exactly m of the $\tilde{\mathbf{x}}$ are positive. Let $J = \{j_1, \dots, j_m\} = \{j : j \text{ such that } \tilde{x}_j > 0\}$ in this case. Then $\mathbf{x}_B = (\tilde{x}_{j_1}, \dots, \tilde{x}_{j_m})$ is the unique basic vector for (2.1) associated with the BFS $\tilde{\mathbf{x}}$. Now obtain the canonical tableau for (2.1) with respect to the basic vector \mathbf{x}_B and denote the entries by α_{ij} , \bar{b}_i , \bar{c}_j , \bar{f} as usual. Using this notation, $\bar{b}_i = \tilde{x}_i$ for $i = 1, \dots, m$, and $\bar{f} = f(\tilde{\mathbf{x}})$. By the nondegeneracy of $\tilde{\mathbf{x}}$, $\bar{b}_i > 0$ for all i . For each $j \notin J$, compute $\theta_j = \min \{\bar{b}_i / \alpha_{ij} : i \text{ such that } \alpha_{ij} > 0\}$, or $+\infty$ if $\alpha_{ij} \leq 0$ for all i . The parameter θ_j is the minimum ratio when x_j enters the basis. For each $s \notin J$, define $\mathbf{x}^s(\delta) = (x_1^s(\delta), \dots, x_n^s(\delta))$, where

$$\begin{aligned} x_{j_i}^s(\delta) &= \bar{b}_i - \alpha_{is}\delta \quad \text{for } i = 1, \dots, m \\ x_S^s(\delta) &= \delta \\ x_j^s(\delta) &= 0 \quad \text{for all } j \notin J \cup \{s\} \end{aligned}$$

Then the set $\{\mathbf{x}^s(\theta_s) : s \notin J \text{ such that } \theta_s \text{ is finite}\}$ is the set of adjacent extreme points of $\tilde{\mathbf{x}}$ in S . The set of adjacent extreme points of $\tilde{\mathbf{x}}$ on S at which $f(\mathbf{x}) \geq f(\tilde{\mathbf{x}})$ is $\{\mathbf{x}^s(\theta_s) : s \notin J \text{ such that } \bar{c}_s \geq 0 \text{ and } \theta_s \text{ is finite}\}$. The set of unbounded edges of S through $\tilde{\mathbf{x}}$ is $\{\{\mathbf{x}^s(\delta) : \delta \geq 0\} : s \notin J \text{ such that } \theta_s = +\infty\}$. The set of bounded edges of S through $\tilde{\mathbf{x}}$ is $\{\{\mathbf{x}^s(\delta) : \delta \geq 0\} : s \notin J \text{ such that } \theta_s \text{ is finite}\}$.

2.3.5 Main Geometric Argument

Based on the discussions in the above sections, we can state the central mathematical result associated with the simplex method. Let S be a convex polytope in R^n and let $f(\mathbf{x})$ be a linear objective functional defined on it.

1. If \mathbf{x}^* is any extreme point of S either \mathbf{x}^* minimizes $f(\mathbf{x})$ on S or there exists an edge (bounded or unbounded) of S through \mathbf{x}^* such that $f(\mathbf{x})$ decreases strictly as we move along this edge away from \mathbf{x}^* .

2. If $f(\mathbf{x})$ is bounded below on S and $\tilde{\mathbf{x}}$ is any extreme point of S , either $\tilde{\mathbf{x}}$ minimizes $f(\mathbf{x})$ on S or there exists an adjacent extreme point of $\tilde{\mathbf{x}}$ on S , say $\hat{\mathbf{x}}$, such that $f(\hat{\mathbf{x}}) < f(\tilde{\mathbf{x}})$.

For minimizing a linear functional $f(\mathbf{x})$ on a convex polytope S , the simplex algorithm starts at an extreme point of S and at each iteration moves along an edge of S incident to this extreme point, such that $f(\mathbf{x})$ decreases. Algorithms of this type are known as adjacent vertex methods. The class of optimization problems that can be solved by such methods include LPs, fractional linear programs, multiple objective linear programs, linear bilevel programs, quadratic programs, separable nonlinear programs, and a subclass of nonlinear programs whose objective functions satisfy certain monotonicity properties. In most of these cases, though, more efficient methods are available.

2.3.6 Alternative Optima and Uniqueness

Let \mathbf{B}^* be an optimal BFS to the LP (2.1) with optimal objective function value f^* , and reduced cost coefficients \bar{c}_j . By the fundamental optimality criterion any feasible solution \mathbf{x} that makes $f(\mathbf{x}) = f^*$ is also an optimal solution to this problem. Hence the set of optimal feasible solutions to (2.1) must satisfy

$$\begin{aligned}\mathbf{A}\mathbf{x} &= \mathbf{b}, \mathbf{x} \geq \mathbf{0} \\ f(\mathbf{x}) &= c\mathbf{x} = f^*\end{aligned}$$

This leads to an alternative proof of Theorem 2.1.3 which can be rephrased as:

1. The set of optimal feasible solutions of an LP is itself a convex polytope, and
2. every convex combination of optimal feasible solutions is also optimal.

Because \mathbf{B}^* is an optimal basis, all the reduced cost coefficients \bar{c}_j must be nonnegative. Letting Q be the set of nonbasic variables, at optimality (2.27b) can be written as

$$f(\mathbf{x}) = f^* + \sum_{j \in Q} \bar{c}_j x_j$$

From this equation we see that every feasible solution to (2.1) in which the x_j variables with strictly positive \bar{c}_j 's are zero, is an optimal feasible solution. This is known as the *complementary slackness condition* and implies that if $T = \{j : j \text{ such that } \bar{c}_j > 0\}$, the set of optimal feasible solutions of (2.1) is the set of feasible solutions of (2.1) in which $x_j = 0$ for all $j \in T$.

This means that if x_q is nonbasic at the optimum and $\bar{c}_q = 0$, and if the operation of bringing x_q into the basis involves a nondegenerate pivot, doing so would yield an

alternative optimal BFS. This leads to the following result which is sufficient (but not necessary) for (2.1) to have a unique optimal solution.

Proposition 2.3.1 If there exists an optimal basis for the LP (2.1) in which the reduced cost coefficients of all the nonbasic variables are strictly positive, the corresponding BFS is the unique optimum.

2.3.7 Ranking Extreme Points

One of the first algorithms purposed for solving the linear bilevel programming problem is based on the idea of examining the vertices adjacent to the vertex at which the leader's objective function is independently minimized without regard to the follower's objective. The procedure parallels the idea of ranking the extreme points of a polytope and is discussed below.

Let S be the set of feasible solutions to the LP (2.1) which is assumed to have a finite optimum. Let E be the set of extreme points in S . When the simplex method is applied to this LP, it finds an optimal BFS, $\mathbf{x}^1 \in E$ at which $f(\mathbf{x})$ achieves its minimum value in S . Let \mathbf{x}^2 be an extreme point of S satisfying $f(\mathbf{x}^2) = \min\{f(\mathbf{x}) : \mathbf{x} \in E \setminus \{\mathbf{x}^1\}\}$. Then \mathbf{x}^2 is known as the second best extreme point solution for (2.1) or the second best BFS. It is possible that $f(\mathbf{x}^2) = f(\mathbf{x}^1)$, implying that \mathbf{x}^2 is an alternative optimum.

In ranking the extreme points of S in nondecreasing order of the value of $f(\mathbf{x})$ we are aiming for a sequence $\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3, \dots$ with the property that

$$f(\mathbf{x}^{r+1}) = \min\{f(\mathbf{x}) : \mathbf{x} \in E \setminus \{\mathbf{x}^1, \dots, \mathbf{x}^r\}\} \quad (2.32)$$

for each $r \geq 1$. When determined in this manner, \mathbf{x}^{r+1} is known at the $(r+1)$ st best extreme point or BFS. Before presenting the algorithm we prove some theoretical results underlying its development.

Proposition 2.3.2 Let $\tilde{\mathbf{x}}$ and $\hat{\mathbf{x}}$ be a pair of extreme points of S . Then there exists and edge path of S between $\tilde{\mathbf{x}}$ and $\hat{\mathbf{x}}$.

Proof: For $\tilde{\mathbf{x}}$ a BFS of (2.1), let $J = \{j : \tilde{x}_j > 0\} = \{j_1, \dots, j_k\}$ where $k \leq m$. Define $d_j = 0$ if $j \in J$, 1 otherwise, and let $\psi(\mathbf{x}) = \sum_{j=1}^n d_j x_j$. Then $\psi(\tilde{\mathbf{x}}) = 0$ and for every $\mathbf{x} \in S$, $\mathbf{x} \neq \tilde{\mathbf{x}}$ we have $\psi(\mathbf{x}) > 0$. This means that $\tilde{\mathbf{x}}$ is the unique optimal solution of the LP: $\min\{\psi(\mathbf{x}) : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$. Apply the simplex algorithm to solve this problem starting with $\hat{\mathbf{x}}$ as the initial BFS. Because $\tilde{\mathbf{x}}$ is the unique optimum of this LP, the algorithm walks along an edge path of S which must terminate at $\tilde{\mathbf{x}}$. ■

Proposition 2.3.3 Let \mathbf{x}^1 be an optimal BFS of (2.1) and let $\tilde{\mathbf{x}}$ be another BFS. There exists an edge path of S from $\tilde{\mathbf{x}}$ to \mathbf{x}^1 with the property that as we walk along this path from $\tilde{\mathbf{x}}$ to \mathbf{x}^1 , the value of $f(\mathbf{x})$ is nonincreasing.

Proof: If $\mathbf{c} = \mathbf{0}$, then all the feasible solutions of (2.1) have the same objective function value of zero and the result follows from Proposition 2.3.2. Assume now that $\mathbf{c} \neq \mathbf{0}$. Apply the simplex algorithm to solve (2.1) starting from the BFS $\tilde{\mathbf{x}}$. If \mathbf{x}^1 is the unique optimum the algorithm traces an edge path of S terminating at \mathbf{x}^1 along which $f(\mathbf{x})$ is nonincreasing. If \mathbf{x}^1 is not the unique optimum, it may terminate at an alternative optimal BFS $\hat{\mathbf{x}}$. In this case the set of all optimal solutions is given by $F = \{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}, f(\mathbf{x}^1) = \mathbf{c}\mathbf{x}, \mathbf{x} \geq \mathbf{0}\}$. Because F is a face of S , all extreme points and edges of F are similarly extreme points and edges of S . Now \mathbf{x}^1 and $\hat{\mathbf{x}}$ are two extreme points of F . From Proposition 2.3.2 there exists an edge path in F from $\hat{\mathbf{x}}$ to \mathbf{x}^1 and so every point on this path satisfies $f(\mathbf{x}) = f(\hat{\mathbf{x}}) = f(\mathbf{x}^1)$. Combining the edge path from $\tilde{\mathbf{x}}$ to $\hat{\mathbf{x}}$ obtained earlier during the simplex algorithm with the edge path from $\hat{\mathbf{x}}$ to \mathbf{x}^1 in F , we have the required path. ■

Proposition 2.3.4 Suppose that the r best extreme points of S in the ranked sequence, $\mathbf{x}^1, \dots, \mathbf{x}^r$, have already been determined. Then the $(r+1)$ st best extreme point of S , \mathbf{x}^{r+1} , is adjacent to (and distinct from) one of these r points.

Proof: Let $\{y^1, \dots, y^t\}$ be the set of all adjacent extreme points of \mathbf{x}^1 , or \mathbf{x}^2, \dots or \mathbf{x}^r on S excluding $\mathbf{x}^1, \dots, \mathbf{x}^r$. Let \mathbf{x}^{r+1} be the point in $\{y^1, \dots, y^t\}$ that minimizes $f(\mathbf{x})$ among them. If $\hat{\mathbf{x}}$ is any extreme point of S distinct from $\mathbf{x}^1, \dots, \mathbf{x}^r, y^1, \dots, y^t$, any edge path in S from \mathbf{x}^1 to $\hat{\mathbf{x}}$ must contain one of the points y^1, \dots, y^t as an intermediate point. Thus by Proposition 2.3.3, we must have $f(\hat{\mathbf{x}}) \geq f(\mathbf{x}^{r+1})$, implying that \mathbf{x}^{r+1} determined in this manner satisfies (2.32). ■

Ranking algorithm: If the r best extreme points of S have been determined, Proposition 2.3.4 says that is only necessary to look among the corresponding adjacent extreme points for the $(r+1)$ st. This provides the main result for ranking the extreme points of S in nondecreasing order of the value of $f(\mathbf{x})$. The algorithm, taken from [M12], begins with an optimal BFS \mathbf{x}^1 of (2.1).

Step 1: Let \mathbf{x}^1 be the first element in the ranked sequence. Obtain all adjacent extreme points of \mathbf{x}^1 in S and store these in a LIST in increasing order of the objective function $f(\mathbf{x})$ from top to bottom. The procedure described in Section 2.3.4 can be used to generate the adjacent extreme points. While storing the extreme points in the list, it is convenient to store the corresponding basic vectors, their values, and the corresponding objective function values.

General Step: Suppose the extreme points $\mathbf{x}^1, \dots, \mathbf{x}^r$ in the ranked sequence have already been obtained. The list at this stage contains all the adjacent extreme points of $\mathbf{x}^1, \dots, \mathbf{x}^r$, excluding $\mathbf{x}^1, \dots, \mathbf{x}^r$, arranged in increasing order of their objective

function values from top to bottom. Select the extreme point at the top of the list and make it \mathbf{x}^{r+1} , the next extreme point in the ranked sequence, and remove it from the list. Obtain all the adjacent extreme points of \mathbf{x}^{r+1} in S which satisfy $f(\mathbf{x}) \geq f(\mathbf{x}^{r+1})$. If any one of them is not in the current list and not equal to any of the extreme points in the ranked sequence obtained so far, insert it in the appropriate slot in the list according to its objective function value. With this new list, continue until as many extreme points as required are obtained.

For large problems, the list of ranked order extreme points can get quite long since it is necessary to retain data on all extreme points adjacent to the first r . This is true even if we wish to obtain only the first k ($> r$) best extreme points. The $(r + 1)$ st best extreme point may in fact be adjacent to any of the first r extreme points and not necessarily the r th. A final observation should be made about degeneracy. If the course of the algorithm, if a degenerate BFS appears in the ranked sequence, then finding all its adjacent extreme points requires additional bookkeeping and most likely an increase in computational complexity, as mentioned in Section 2.3.4.

2.4 ADDITIONAL FEATURES

In this section two extensions to the basic simplex method are discussed that are needed in practice. The first concerns the construction of an initial basic feasible solution; the second deals with the case where both upper and lower bounds are present on some of the variables. We conclude with a brief presentation of the linear complementarity problem and show that any linear program can be modeled equivalently. This result has led to the development of several algorithms for the linear bilevel programming problem, as we shall see in Chapter 5.

2.4.1 Phase 1 and Artificial Variables

To start the simplex algorithm a basic feasible solution is required. Recall that the Fundamental Theorem 2.1.1 states in part that if an LP in standard form has a feasible solution it has a BFS. In some problems, a BFS can be found by inspection. Suppose $\mathbf{b} \geq \mathbf{0}$ and \mathbf{A} contains m unit vectors. Then the columns of \mathbf{A} can be permuted to obtain the matrix $(\mathbf{A}', \mathbf{I}_m)$, where \mathbf{I}_m is the $m \times m$ identity matrix. Writing (2.1b) as $\mathbf{I}_m \mathbf{x}_B + \mathbf{A}' \mathbf{x}_N = \mathbf{b}$, it is clear that $\mathbf{x}_B = \mathbf{b}$, $\mathbf{x}_N = \mathbf{0}$ is a BFS. This case occurs in the situation where the original problem is

$$\begin{aligned} \min \quad & f(\mathbf{x}) = \mathbf{c}\mathbf{x} \\ \text{subject to} \quad & \mathbf{Ax} \leq \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0} \end{aligned}$$

and m slack variables are added to each constraint giving

$$\begin{aligned} \min \quad & f(\mathbf{x}) = \mathbf{c}\mathbf{x} \\ \text{subject to} \quad & \mathbf{Ax} + \mathbf{I}_m \mathbf{x}_S = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}, \quad \mathbf{x}_S \geq \mathbf{0} \end{aligned}$$

where $\mathbf{x}_S = \mathbf{x}_B$ and $\mathbf{x} = \mathbf{x}_N$.

Assume now that the constraints are written in standard form $\mathbf{Ax} = \mathbf{b} \geq \mathbf{0}$ and \mathbf{A} does not contain an identity matrix. Without loss of generality, \mathbf{A} can be partitioned as

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{I}_k \\ \mathbf{A}_2 & \mathbf{0} \end{pmatrix}$$

where \mathbf{I}_k is an identity matrix of order k ($0 \leq k \leq m$). Thus the constraints can be written as

$$\begin{aligned} \mathbf{A}_1 \mathbf{x}_N + \mathbf{I}_k \mathbf{x}_k &= \mathbf{b}_1 \\ \mathbf{A}_2 \mathbf{x}_N &= \mathbf{b}_2 \end{aligned} \tag{2.33}$$

where $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2)^T$. Now consider a related LP with an $(m - k)$ -dimensional vector $\mathbf{x}_A \geq \mathbf{0}$ of artificial variables and the constraint set

$$\begin{aligned} \mathbf{A}_1 \mathbf{x}_N + \mathbf{I}_k \mathbf{x}_k &= \mathbf{b}_1 \\ \mathbf{A}_2 \mathbf{x}_N + \mathbf{I}_{m-k} \mathbf{x}_A &= \mathbf{b}_2 \end{aligned} \tag{2.34}$$

These constraints have an obvious BFS $\mathbf{x}_k = \mathbf{b}_1$, $\mathbf{x}_A = \mathbf{b}_2$, and $\mathbf{x}_N = \mathbf{0}$. Moreover, (2.33) has a BFS iff (2.34) has a BFS with $\mathbf{x}_A = \mathbf{0}$. Beginning with this BFS, our goal is to find a BFS with $\mathbf{x}_A = \mathbf{0}$ or show that none exists. This is known as *phase 1* of the simplex method and can be achieved by solving the LP

$$\begin{aligned} \min \quad & \omega(\mathbf{x}) = \mathbf{1} \mathbf{x}_A \\ \text{subject to} \quad & \mathbf{x}_0 - \mathbf{c}_N \mathbf{x}_N - \mathbf{c}_k \mathbf{x}_k = 0 \\ & \mathbf{A}_1 \mathbf{x}_N + \mathbf{I}_k \mathbf{x}_k = \mathbf{b}_1 \\ & \mathbf{A}_2 \mathbf{x}_N + \mathbf{I}_{m-k} \mathbf{x}_A = \mathbf{b}_2 \\ & \mathbf{x}_N, \mathbf{x}_k, \mathbf{x}_A \geq \mathbf{0}, \quad \mathbf{x}_0 \text{ unrestricted} \end{aligned} \tag{2.35}$$

where $\mathbf{1}$ is the row vector $(1, \dots, 1)$ of appropriate length. The variable x_0 always remains basic. The initial BFS to (2.35) is $\mathbf{x}_k = \mathbf{b}_1 - \mathbf{A}_1 \mathbf{x}_N$, $\mathbf{x}_A = \mathbf{b}_2 - \mathbf{A}_2 \mathbf{x}_N$, $\mathbf{x}_0 = \mathbf{c}_k \mathbf{b}_1 + (\mathbf{c}_N - \mathbf{c}_k \mathbf{A}_1) \mathbf{x}_N$, $\mathbf{x}_N = \mathbf{0}$, and objective function value $\omega(\mathbf{x}) = \mathbf{1}(\mathbf{b}_2 - \mathbf{A}_2 \mathbf{x}_N)$.

For any feasible solution to (2.35), $\mathbf{x}_A > \mathbf{0}$ implies $\omega(\mathbf{x}) > 0$, and $\mathbf{x}_A = \mathbf{0}$ implies $\omega(\mathbf{x}) = 0$. Thus if an optimal solution to the phase 1 problem (2.35) is positive, (2.34) is infeasible as is the original problem. In contrast, any BFS to (2.35) with $\omega(\mathbf{x}) = 0$ provides a BFS to (2.34). This is obvious if such a BFS contains no artificial variables; for the case where the BFS contains artificial variables, it must be degenerate. Noting that \mathbf{A} is assumed to have full row rank, by appealing to Corollary 2.1.1, we know that is possible to pivot in an equivalent number of original problem variables.

Once such a solution is found, the ω row, the artificial variables, and any original problem variables with positive reduced cost coefficients are dropped. (Actually, the artificial variables can be dropped when they leave the basis.) We then switch to phase 2, and continue with the minimization of x_0 .

Example 2.4.1 Consider the following LP with $n = 5$, $m = 2$ written in standard form.

$$\begin{array}{lll} \min & 5x_1 & + 21x_3 \\ \text{subject to} & x_1 - x_2 + 6x_3 - x_4 & = 2 \\ & x_1 + x_2 + 2x_3 - x_5 & = 1 \\ & x_1, \dots, x_5 \geq 0 \end{array}$$

To solve this problem with the two-phase method, we introduce two artificial variables x_6 and x_7 and phase 1 objective: $\min \omega = x_6 + x_7$. Rewriting the objective row by substituting for x_6 and x_7 gives

$$\begin{aligned} \omega &= 2 - x_1 + x_2 - 6x_3 + x_4 + 1 - x_1 - x_2 - 2x_3 + x_5 \\ &= 3 - 2x_1 - 8x_3 + x_4 + x_5 \end{aligned}$$

This leads to the first canonical tableau

									↓	
		\bar{b}	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
	x_0	2	5	0	21	0	0	0	0	
←	x_6	2	1	-1	(6)	-1	0	1	0	
	x_7	1	1	1	2	0	-1	0	1	
	$-\omega$	-3	-2	0	-8	1	1	0	0	

Pivoting on the indicated element gives

									↓	
		\bar{b}	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
	x_0	-7	3/2	7/2	0	7/2	0	-7/2	0	
	x_3	1/3	1	-1	1	-1	0	1/6	0	
←	x_7	1/3	2/3	(4/3)	0	1/3	-1	-1/3	1	
	$-\omega$	-1/3	-2/3	-4/3	0	-1/3	1	4/3	0	

A second pivot eliminates the artificial variables from the basis so they can be dropped along with the ω row. Switching to phase 2, we now minimize x_0 .

							↓	
		\bar{b}	x_1	x_2	x_3	x_4	x_5	
	x_0	-63/8	-1/4	0	0	21/8	21/8	
	x_3	3/8	1/4	0	1	-1/8	-1/8	
←	x_2	1/4	(1/2)	1	0	1/4	-3/4	

Pivoting out x_2 and pivoting in x_1 gives the next tableau which is optimal.

	\bar{b}	x_1	x_2	x_3	x_4	x_5
x_0	-31/4	0	1/2	0	11/4	9/4
x_3	1/4	0	-1/2	1	-1/4	1/4
x_1	1/2	1	2	0	1/2	-3/2

This gives the final solution $f^* = -x_0 = 31/4$, $\mathbf{x}^* = (1/2, 0, 1/4, 0, 0, 0)$.

2.4.2 Bounded Variables

The most general case of the nonnegativity constraint (2.1c) $\mathbf{x} \geq \mathbf{0}$ restricts a particular variable x_j to be between two bounds; i.e., $l_j \leq x_j \leq u_j$. As mentioned in Section 2.1, it is always possible to redefine a decision variable so that its lower bound is zero. Recall this is done by replacing x_j with $\hat{x}_j + l_j$ and its bounds with $0 \leq \hat{x}_j \leq \hat{u}_j = u_j - l_j$, $j = 1, \dots, n$. To put the resultant formulation into standard form, it is necessary to add slack variables to the upper bound inequality giving $\hat{x}_j + y_j = \hat{u}_j$, where the n slack variables $y_j \geq 0$. This leads to the following model

$$\min_{(\hat{\mathbf{x}}, \mathbf{y})} f(\hat{\mathbf{x}}) = \mathbf{c}\hat{\mathbf{x}} \quad (2.36a)$$

$$\text{subject to } \mathbf{A}\hat{\mathbf{x}} = \mathbf{b} \quad (2.36b)$$

$$\hat{\mathbf{x}} + \mathbf{y} = \hat{\mathbf{u}} \quad (2.36c)$$

$$\hat{\mathbf{x}} \geq \mathbf{0}, \mathbf{y} \geq \mathbf{0} \quad (2.36d)$$

Although the simplex method can be applied to problem (2.36), there is a high price to be paid in terms of computing and storage requirements. The addition of constraint (2.36c) has the effect of increasing the dimensions of the basis from an $m \times m$ matrix to an $(m+n) \times (m+n)$ matrix. This is extremely inefficient and, we will see below, unnecessary. In fact, upper bounds can be treated implicitly in the algorithm, in the same manner in which the lower bounds are treated.

To describe the augmented procedures, we introduce a new definition.

Definition 2.4.1 An *extended basic feasible solution* associated with an LP with variable bounds is a feasible solution for which $n-m$ variables are equal to either their lower (zero) or upper bound; the remaining m (basic) variables correspond to linearly independent columns of \mathbf{A} .

For convenience, we assume that every extended BFS is nondegenerate, which means that the m basic variables take values different from their bounds. Now suppose we start with an extended BFS and examine the nonbasic variables (the variables that are at one of their bounds) as possible candidates to enter the basis. A variable at its lower bound can only be increased, and an increase will only be beneficial

if its reduced cost coefficient is negative. A variable at its upper bound can only be decreased, and a decrease will only be beneficial if its reduced cost coefficient is positive. The value of nonbasic variables can be continuously changed until either (i) the value of a basic variable becomes equal to one of its bounds or (ii) the nonbasic variable being modified reaches its opposite bound. If (i) occurs the corresponding basic variable is declared nonbasic and the nonbasic variable takes its place in the basis. If (ii) occurs first, then the basis is not changed. The simultaneous occurrence of (i) and (ii) results in a degenerate solution which we have ruled out for now. The procedure continues until no further improvement is possible.

For notational simplicity, we consider a problem in which each of the variables has a lower bound of zero and arbitrary upper bound; i.e., $\min\{\mathbf{c}\mathbf{x} : \mathbf{Ax} = \mathbf{b}, \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}\}$. Let Q denote the set of nonbasic variables, $Q_l = \{j : j \in Q, x_j = 0\}$ and $Q_u = \{j : j \in Q, x_j = u_j\}$. Modifying equations in (2.27) to reflect this partition gives

$$\begin{aligned} x_{B_i} &= \bar{b}_i - \sum_{j \in Q_l} \alpha_{ij} x_j - \sum_{j \in Q_u} \alpha_{ij} x_j, \quad i = 1, \dots, m \\ f &= f^0 + \sum_{j \in Q_l} \bar{c}_j x_j + \sum_{j \in Q_u} \bar{c}_j x_j \end{aligned}$$

with current values of x_{B_i} and f

$$\begin{aligned} x_{B_i} &= \bar{b}_i - \sum_{j \in Q_u} \alpha_{ij} u_j, \quad i = 1, \dots, m \\ f &= f^0 + \sum_{j \in Q_u} \bar{c}_j u_j \end{aligned}$$

Proceeding as in Section 2.2.3, it is easy to show that for $j \in Q_u$, $\bar{c}_j > 0$ implies that f can be decreased by moving x_j away from its upper bound u_j (perhaps making it basic). Analogous to Theorem 2.2.2, we have

Theorem 2.4.1 Given an extended basic feasible solution, a sufficient condition for optimality is

- i) $0 \leq x_{B_i} \leq u_{B_i}$, $i = 1, \dots, m$ (feasibility)
- ii) $\bar{c}_j \geq 0$, for all $j \in Q_l$
- iii) $\bar{c}_j \leq 0$, for all $j \in Q_u$.

To incorporate these ideas into the simplex algorithm, we keep a record of whether a nonbasic variable is at zero (lower bound) or at its upper bound. In testing for optimality and choosing a variable to enter the basis, the opposite sign convention on \bar{c}_j is used for a variable at its upper bounds ($\bar{c}_j \leq 0$ for optimality; $\bar{c}_j > 0$ for candidate to enter the basis). The rule for selecting a departing variable must also be modified to yield a first basic variable that reaches zero or its upper bound.

Suppose that x_q , $q \in Q_l$ is the entering variable. The first variable to reach zero (call it x_{B_p}), as discussed in Section 2.2.3, is determined from the minimum ratio test

$$\theta_{pq} = \min_{i=1,\dots,m} \left\{ \frac{\bar{b}_i}{\alpha_{iq}} \mid \alpha_{iq} > 0 \right\}$$

The first variable to reach its upper bound (excluding x_q) is given by x_{B_s} , determined from

$$\beta_{sq} = \min_{i=1,\dots,m} \left\{ \frac{\bar{b}_i - u_{B_i}}{\alpha_{iq}} \mid \alpha_{iq} < 0 \right\}$$

Finally, it is necessary to maintain $x_q \leq u_q$. Thus the departing variable is determined from

$$\min\{\theta_{pq}, \beta_{sq}, u_q\}$$

where ties are broken arbitrarily (the nondegeneracy assumption implies that the choice will be unique). If θ_{pq} or β_{sq} is minimum, x_{B_p} or x_{B_s} , respectively, departs and a simplex iteration is executed. If u_q is the minimum, x_q is both the entering and leaving variable so there is no change of basis. Nevertheless, the change of x_q from zero to u_q must be recorded and the solution vector and objective value must be updated as follows

$$\begin{aligned}\hat{x}_{B_i} &= \bar{b}_i - \alpha_{iq}u_q, \quad i = 1, \dots, m \\ \hat{f}^0 &= f^0 + \bar{c}_q u_q\end{aligned}$$

The case in which x_q , $q \in Q_u$ is the entering variable yields similar results. The first variable to reach zero (excluding x_q) is given by x_{B_p} determined from

$$\hat{\theta}_{pq} = \max_{i=1,\dots,m} \left\{ \frac{\bar{b}_i}{\alpha_{iq}} \mid \alpha_{iq} < 0 \right\}$$

Here $\hat{\theta}_{pq} \leq 0$ represents the change allowed in x_q , so if $u_q < -\hat{\theta}_{pq}$, x_q reaches zero before x_{B_p} . The first variable to reach its upper bound is x_{B_s} determined from

$$\hat{\beta}_{sq} = \min_{i=1,\dots,m} \left\{ \frac{\bar{b}_i - u_{B_i}}{\alpha_{iq}} \mid \alpha_{iq} > 0 \right\}$$

In this case, the departing variable is determined from

$$\min\{\hat{\theta}_{pq}, \hat{\beta}_{sq}, -u_q\}$$

Algorithm Modifications

When bounded variables are present, the computations in the simplex algorithm outlined in Section 2.2.3 proceed as usual except that the choice of the pivot element must be modified slightly. To describe the modifications, it is convenient to introduce the notation $x_j^+ = x_j$, $x_j^- = u_j - x_j$, which allows us to maintain the optimality

condition $\bar{c}_j \geq 0$ for all j nonbasic. As the method progresses we change back and forth from x_j^+ to x_j^- , depending on whether the variable x_j has most recently been at its lower or upper bound, respectively.

Using this notation, the procedure for moving from one extended BFS to another can be easily implemented by following the strategy outlined above.

Step 1. Determine a nonbasic variable x_q^+ for which $\bar{c}_q < 0$. If no such variable exists, stop; the current solution is optimal.

Step 2. Evaluate the three terms

a) u_q (upper bound on x_q)

$$\text{b) } \min \left\{ \frac{\bar{b}_i}{\alpha_{iq}} \mid \alpha_{iq} > 0 \right\}$$

$$\text{c) } \min \left\{ \frac{\bar{b}_i - u_q}{\alpha_{iq}} \mid \alpha_{iq} < 0 \right\}$$

Step 3. According to which number in Step 2 is smallest, update the basis inverse \mathbf{B}^{-1} and the current solution $\mathbf{B}^{-1}\mathbf{b}$ at Step 4 of the revised simplex method as follows.

- a) The variable x_q goes to its opposite bound. Subtract u_q times updated column q (α_q) from $\mathbf{B}^{-1}\mathbf{b}$. Multiply column q as well as \bar{c}_q by minus one and note a change in the sign of the variable. The basis doesn't change and no pivot is required.
- b) Suppose p is the minimizing index in (b) of Step 2. Then the p th basic variable returns to its old bound and we pivot on the pq th element α_{pq} .
- c) Suppose p is the minimizing index in (c) of Step 2. Then the p th basic variable goes to its opposite bound. Subtract u_p from \bar{b}_p , change the sign of α_{pp} , pivot on the pq th element, and note a change in the sign of variable p .

Return to Step 1.

2.4.3 Kuhn-Tucker Conditions and the Linear Complementarity Problem

Let us consider the LP (2.1) in slightly different form: $\min\{\mathbf{c}\mathbf{x} : \mathbf{A}\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq 0\}$ and define the corresponding Lagrangian as in Section 4.2.3.

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{c}\mathbf{x} - \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) - \boldsymbol{\mu}^T\mathbf{x}$$

where λ and μ are the m - and n -dimensional Kuhn-Tucker or Lagrange multipliers for the technological and nonnegativity constraints, respectively. Writing the Kuhn-Tucker necessary conditions (see Theorem 4.2.4) for optimality gives

$$\begin{aligned}\nabla_x \mathcal{L}(x, \lambda, \mu) &= c - \lambda^T A - \mu^T = 0 \\ \nabla_\lambda \mathcal{L}(x, \lambda, \mu) &= Ax - b \geq 0 \\ \nabla_\mu \mathcal{L}(x, \lambda, \mu) &= x \geq 0 \\ \lambda^T (Ax - b) &= 0 \\ \mu^T x &= 0 \\ \lambda \geq 0, \quad \mu \geq 0\end{aligned}$$

which, due to the convexity of the linear program, are sufficient conditions as well. This system can be expressed equivalently as

$$\begin{aligned}Ax - b &\geq 0, \quad x \geq 0 \\ \lambda^T A &\leq c, \quad \lambda \geq 0 \\ \lambda^T (Ax - b) &= (c - \lambda^T A)x = 0\end{aligned}\tag{2.37}$$

which we show in the next section are exactly the primal and dual conditions for optimality. The last equation in (2.37) implies $\lambda^T b = cx$. Introducing slack variables u and v , the system becomes

$$Ax - v = b\tag{2.38a}$$

$$A^T \lambda + u = c^T\tag{2.38b}$$

$$cx - b^T \lambda = 0\tag{2.38c}$$

$$x \geq 0, \quad \lambda \geq 0, \quad u \geq 0, \quad v \geq 0\tag{2.38d}$$

Thus if $(\hat{x}, \hat{\lambda}, \hat{u}, \hat{v})$ is a feasible solution to (2.38), then \hat{x} is an optimal feasible solution to the LP defined with inequality constraints. Conversely, if \hat{x} is an optimal solution to the LP with optimal basis \hat{B} , then for $\hat{\lambda} = c_B B^{-1}$ there exist vectors \hat{u} and \hat{v} that satisfy (2.38).

This simple analysis has shown that solving an LP is equivalent to solving the system of linear equations (2.38) in nonnegative variables. Hence, any algorithm for solving such systems can be used to solve LPs directly without any need for optimization (in fact, this is what interior point methods do). Conversely, any algorithm for solving LPs can be used to solve a system of linear equations in nonnegative variables by solving a phase-1-type problem.

Going one step farther and rewriting (2.38a)–(2.38b) as $v = Ax - b$, $u = A^T \lambda - c^T$, and noting that (2.37) implies $u^T x + v^T \lambda = 0$, we get a slightly different representa-

tion of the optimality conditions. Now let

$$\zeta = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} \quad \boldsymbol{\eta} = \begin{pmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{pmatrix} \quad \mathbf{M} = \begin{pmatrix} \mathbf{0} & -\mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{pmatrix} \quad \boldsymbol{\nu} = \begin{pmatrix} \mathbf{c}^T \\ -\mathbf{b} \end{pmatrix}$$

Solving a linear program written with inequality constraints is equivalent to solving the system

$$\begin{aligned} \zeta - \mathbf{M}\boldsymbol{\eta} &= \boldsymbol{\nu} \\ \zeta \geq \mathbf{0}, \quad \boldsymbol{\eta} &\geq \mathbf{0} \\ \zeta^T \boldsymbol{\eta} &= 0 \end{aligned} \tag{2.39}$$

The constraints $\zeta \geq \mathbf{0}$, $\boldsymbol{\eta} \geq \mathbf{0}$, $\zeta^T \boldsymbol{\eta} = 0$ imply that $\zeta_j \eta_j = 0$ for all j . Hence at most one variable in each complementary pair (ζ_j, η_j) can take a positive value in any solution. Problems of the form (2.39) are known as linear complementarity problems and have a number of applications in optimization.

2.5 DUALITY

In this section, an elegant symmetrical relationship is derived for the linear programming problem. It will be shown that associated with every LP is a corresponding dual LP. Both problems are constructed from the same underlying cost and constraint data but in such a way that if one is a minimization problem the other is a maximization problem, and the optimal values of either objective function, if finite, are equal. The variables of the dual problem can be interpreted as prices associated with the constraints of the original or primal problem, and through this association it is possible to give an economic characterization of the dual whenever there is such a characterization for the primal.

Initially, we depart from the standard form of the LP (2.1) to highlight the symmetry of the relationship. Specifically, the following pair of LPs denoted by (P) and (D), respectively, are considered.

Primal	Dual
$\min f = \mathbf{c}\mathbf{x}$	$\max \phi = \boldsymbol{\lambda}^T \mathbf{b}$
subject to $\mathbf{Ax} \geq \mathbf{b}$	subject to $\boldsymbol{\lambda}^T \mathbf{A} \leq \mathbf{c}^T$
$\mathbf{x} \geq \mathbf{0}$	$\boldsymbol{\lambda} \geq \mathbf{0}$

(2.40)

where $\mathbf{x} \in R^n$ and $\boldsymbol{\lambda} \in R^m$ are the primal and dual variables, respectively, and the coefficients are as previously defined.

Example 2.4.1 with the surplus variables removed is

$$\begin{aligned} \min f = & 5x_1 + 21x_3 \\ \text{subject to } & x_1 - x_2 + 6x_3 \geq 2 \\ & x_1 + x_2 + 2x_3 \leq 1 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

The dual of this problem is

$$\begin{aligned} \max \phi = & 2\lambda_1 + \lambda_2 \\ \text{subject to } & \lambda_1 + \lambda_2 \leq 5 \\ & -\lambda_1 + \lambda_2 \leq 0 \\ & 6\lambda_1 + 2\lambda_2 \leq 21 \\ & \lambda_1, \lambda_2 \geq 0 \end{aligned}$$

The optimal solution including slack variables is $\lambda^* = (11/4, 9/4, 0, 1/2, 0)$, $f^* = 31/4$. The corresponding solution to the primal problem including surplus variables is $f^* = 31/4$, $\mathbf{x}^* = (1/2, 0, 1/4, 0, 0, 0)$. Note that the primal and dual objective values are equal. This is always the case.

The dual of any LP can be found by converting it to the form of the primal problem (P). For example, given an LP in standard form

$$\begin{aligned} \min f = & c\mathbf{x} \\ \text{subject to } & A\mathbf{x} = b \\ & \mathbf{x} \geq 0 \end{aligned}$$

we can replace the constraints $A\mathbf{x} = b$ as two inequalities: $A\mathbf{x} \leq b$ and $-A\mathbf{x} \leq -b$ so the coefficient matrix becomes $\begin{pmatrix} A \\ -A \end{pmatrix}$ and the right-hand-side vector becomes $\begin{pmatrix} b \\ -b \end{pmatrix}$. Introducing a partitioned dual row vector (γ_1, γ_2) , the corresponding dual is

$$\begin{aligned} \max \phi = & \gamma_1 b - \gamma_2 b \\ \text{subject to } & \gamma_1 A - \gamma_2 A \leq c^T \\ & \gamma_1 \geq 0, \gamma_2 \geq 0 \end{aligned}$$

Letting $\lambda^T = \gamma_1 - \gamma_2$ we may simplify the representation of this problem to obtain the following pair:

<i>Primal</i>	<i>Dual</i>	
$\min f = c\mathbf{x}$	$\max \phi = \lambda^T b$	(2.41)
subject to $A\mathbf{x} = b$	subject to $\lambda^T A \leq c^T$	
$\mathbf{x} \geq 0$		

This is the asymmetric form of the duality relation. Similar transformations can be worked out for any linear program by first putting the primal in the form of

(2.40), constructing the dual, and then simplifying the latter to account for special structure. We say two LPs are equivalent if one can be transformed into another so that feasible solutions, optimal solutions and corresponding dual solutions are preserved; e.g., (2.40) and (2.41) are equivalent primal–dual representations.

2.5.1 Primal–Dual Relationship

Although it is always possible to transform an LP into the primal–dual pairs in either (2.40) or (2.41), it is often inconvenient to make this transformation for problems with a variety of constraints and variable types. Therefore, it is useful to state the rules for constructing a dual problem. To begin, there will be one dual variable associated with each constraint in the primal (excluding nonnegativity restrictions on individual variables, if any). If the primal constraint is a \geq inequality, the dual variable is restricted to being nonnegative, and vice versa. If the primal constraint is an equality, the dual variable is unrestricted in sign.

There is one dual constraint corresponding to each primal variable. Thus each column vector of the primal tableau leads to a constraint in the dual problem. Let λ^T be the row vector of dual variables. Let a_j be the column vector of the coefficients of the primal variable x_j and c_j its objective function coefficient. Then the dual constraint corresponding to x_j is $\lambda^T a_j = c_j$ if x_j is unrestricted in sign in the primal problem. If x_j is restricted to being nonnegative (nonpositive), the corresponding inequality in the dual problem is $\lambda^T a_j \leq c_j$ ($\lambda^T a_j \geq c_j$).

If the primal is a minimization problem, the dual is a maximization problem and vice versa. The right-hand-side constants of the primal are the coefficients of the dual objective function and vice versa. Any LP can be designated the primal whether or not it matches a standard structure.

2.5.2 Dual Theorems

To this point, the relation between the primal and dual problems has been simply a formal one based on what might appear as an arbitrary definition. The material in Section 2.4.3 relating to the Kuhn-Tucker conditions, though, suggests otherwise. In this section, a deeper connection between the pair is derived in terms of feasibility, optimality and bounds.

Proposition 2.5.1 Dual of the dual is primal.

Proof: For any primal–dual pair, this can be verified by writing out the corresponding dual and defining the variables appropriately. ■

Proposition 2.5.2 Duals of equivalent problems are equivalent. Let (P) refer to an LP and let (D) be its dual. Let (\hat{P}) be an LP that is equivalent to (P) . Let (\hat{D}) be the dual of (\hat{P}) . Then (\hat{D}) is equivalent to (D) ; i.e., they have the same optimal objective function values or they are both infeasible.

Proof: By construction. ■

Theorem 2.5.1 (Weak Duality Theorem) In a primal–dual pair of LPs, let \mathbf{x} be a primal feasible solution and $f(\mathbf{x})$ the corresponding value of the primal objective function that is to be minimized. Let $\boldsymbol{\lambda}$ be a dual feasible solution and $\phi(\boldsymbol{\lambda})$ the corresponding dual objective function that is to be maximized. Then $f(\mathbf{x}) \geq \phi(\boldsymbol{\lambda})$.

Proof: We consider the case where the primal and the dual are stated as in (2.40). Then

$$\begin{aligned} \mathbf{A}\mathbf{x} &\geq \mathbf{b} && \text{(because } \mathbf{x} \text{ is primal feasible)} \\ \boldsymbol{\lambda}^T \mathbf{A}\mathbf{x} &\geq \boldsymbol{\lambda}^T \mathbf{b} && \text{(because } \boldsymbol{\lambda} \geq 0) \\ \boldsymbol{\lambda}^T \mathbf{A} &\leq \mathbf{c} && \text{(because } \boldsymbol{\lambda} \text{ is dual feasible)} \\ \boldsymbol{\lambda}^T \mathbf{A}\mathbf{x} &\leq \boldsymbol{\lambda}^T \mathbf{c} && \text{(because } \mathbf{x} \geq 0) \end{aligned} \quad (2.42)$$

Combining (2.42) and (2.43) we get $\boldsymbol{\lambda}^T \mathbf{c} \geq \boldsymbol{\lambda}^T \mathbf{A}\mathbf{x} \geq \boldsymbol{\lambda}^T \mathbf{b}$; that is, $f(\mathbf{x}) \geq \phi(\boldsymbol{\lambda})$, which proves the theorem when the primal and dual are stated in this form. In general, every LP can be transformed into an equivalent problem identical to the primal in (2.40). ■

Corollaries of the Weak Duality Theorem

Considering any primal–dual pair of LPs, let the primal refer to the minimization problem and the dual to the maximization problem in the pair.

1. The primal objective value of any primal feasible solution is an upper bound to the maximum value of the dual objective in the dual problem.
2. The dual objective value of any dual feasible solution is a lower bound to the minimum value of the primal objective in the primal problem.
3. If the primal problem is feasible and its objective function is unbounded below on the primal feasible solution set, the dual problem cannot have a feasible solution.
4. If the dual problem is feasible and its objective function is unbounded above on the dual feasible solution set, the primal problem cannot have a feasible solution.
5. The converse of (3) is the following: If the dual problem is infeasible and the primal problem is feasible, the primal objective function is unbounded below on the primal feasible solution set. Similarly, the converse of (4) is: If the primal problem is infeasible and the dual problem is feasible, the dual objective function is unbounded above on the dual feasible solution set.

It is possible that both the primal and the dual problems in a primal–dual pair have no feasible solutions. For example, consider the following:

$$\begin{array}{ll} \min f = & 3x_1 - 5x_2 \\ \text{subject to} & x_1 - x_2 = 1 \\ & -x_1 + x_2 = 3 \\ & x_1 \geq 0, x_2 \geq 0 \end{array} \quad \begin{array}{ll} \max \phi = & \lambda_1 + \lambda_2 \\ \text{subject to} & \lambda_1 - \lambda_2 \leq 3 \\ & -\lambda_1 + \lambda_2 \leq -5 \\ & \lambda_1, \lambda_2 \text{ unrestricted} \end{array}$$

Clearly both problems in the pair are infeasible. Thus even though the result in (5) is true, the fact that the dual problem is infeasible in a primal–dual pair of LPs does not imply that the primal objective function is unbounded on the primal feasible solution set, unless it is known that the primal is feasible.

Theorem 2.5.2 (Sufficient Optimality Criterion) In a primal–dual pair of LPs, let $\phi(\mathbf{x})$ be the primal objective function and $f(\boldsymbol{\lambda})$ be the dual objective function. If $\hat{\mathbf{x}}$, $\hat{\boldsymbol{\lambda}}$ are a pair of primal and dual feasible solutions satisfying $f(\hat{\mathbf{x}}) = \phi(\hat{\boldsymbol{\lambda}})$, then $\hat{\mathbf{x}}$ is an optimal feasible solution of the primal and $\hat{\boldsymbol{\lambda}}$ is an optimum feasible solution of the dual.

Proof: Suppose the primal denotes the minimization problem in the primal–dual pair. Let \mathbf{x} be any primal feasible solution. By the weak duality theorem, we have $f(\mathbf{x}) \geq \phi(\hat{\boldsymbol{\lambda}})$ because $\hat{\boldsymbol{\lambda}}$ is dual feasible. But $f(\hat{\mathbf{x}}) = \phi(\hat{\boldsymbol{\lambda}})$ by hypothesis. So $f(\mathbf{x}) \geq f(\hat{\mathbf{x}})$ for all \mathbf{x} primal feasible. Thus $\hat{\mathbf{x}}$ is optimal to the primal problem. Similarly, $\hat{\boldsymbol{\lambda}}$ is optimal to the dual problem. ■

Theorem 2.5.3 (Fundamental Duality Theorem) In a primal–dual pair of LPs, if either the primal or the dual problem has an optimal feasible solution, then the other does also and the two optimal objective values are equal.

Proof: We will prove the result for the case where the primal and dual problems are stated as in (2.41). Solving the primal problem by the simplex algorithm in Section 2.2.3 yields an optimal solution $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$, $\mathbf{x}_N = \mathbf{0}$ with $\bar{\mathbf{c}} = \mathbf{c}_N - \mathbf{c}_B\mathbf{B}^{-1}\mathbf{N} \geq \mathbf{0}$, which can be written $[\mathbf{c}_B, \mathbf{c}_N - \mathbf{c}_B\mathbf{B}^{-1}(\mathbf{B}, \mathbf{N})] = \mathbf{c} - \mathbf{c}_B\mathbf{B}^{-1}\mathbf{A} \geq \mathbf{0}$. Now if we define $\boldsymbol{\lambda}^T = \mathbf{c}_B\mathbf{B}^{-1}$ we have $\boldsymbol{\lambda}^T\mathbf{A} \leq \mathbf{c}$ and $f(\mathbf{x}) = \mathbf{c}_B\mathbf{x}_B = \mathbf{c}_B\mathbf{B}^{-1}\mathbf{b} = \boldsymbol{\lambda}^T\mathbf{b} = \phi(\boldsymbol{\lambda})$. By the sufficient optimality criterion, Theorem 2.5.2, $\boldsymbol{\lambda}$ is a dual optimal solution. This completes the proof when the primal and dual are as stated here. ■

In general, every LP can be transformed into an equivalent problem in standard form. This equivalent problem is of the same type as the primal in (2.41), hence the proof applies. Also, by Proposition 2.5.2, the dual of the equivalent problem in standard form is equivalent to the dual of the original problem. Thus the fundamental duality theorem must hold for it too. ■

Corollaries of the Fundamental Duality Theorem

Alternative statement of the duality theorem: “If both problems in a primal–dual pair of LPs have feasible solutions, then both have optimal feasible solutions and the

optimal objective values of the two problems are equal.” This is easily proved by using the weak duality theorem and the fundamental duality theorem.

Separation property of objective values: Consider a primal–dual pair of LPs and suppose the minimization problem in the pair is the primal with the objective function $f(\mathbf{x})$. Suppose the dual objective function is $\phi(\boldsymbol{\lambda})$. If both problems have feasible solutions, then the values assumed by the two objective functions at feasible solutions of the respective problems are separated on the real line; i.e., $\phi(\boldsymbol{\lambda}) \leq f(\mathbf{x})$ for all feasible \mathbf{x} and $\boldsymbol{\lambda}$.

Primal objective unbounded: If the primal is the minimization problem in a primal–dual pair, and if the primal is feasible and the dual is infeasible, then the primal cannot have an optimal feasible solution, that is, the primal objective function is unbounded below.

Dual objective unbounded: If the dual is the maximization problem in a primal–dual pair, and if the dual is feasible and the primal infeasible, then the dual cannot have an optimal feasible solution, that is, the dual objective function is unbounded above.

Proposition 2.5.3 (Necessary and Sufficient Optimality Conditions) Consider a primal–dual pair of LPs. Let $\mathbf{x}, \boldsymbol{\lambda}$ be the vectors of primal and dual variables and let $f(\mathbf{x}), \phi(\boldsymbol{\lambda})$ be the primal and dual objective functions, respectively. If $\hat{\mathbf{x}}$ is a primal feasible solution, it is an optimal solution of the primal problem iff there exists a dual feasible solution $\hat{\boldsymbol{\lambda}}$ satisfying $f(\mathbf{x}) = \phi(\boldsymbol{\lambda})$.

Proof: Follows directly from Theorems 2.5.3 and 2.5.2. ■

Theorem 2.5.4 (Complementary Slackness Theorem) A pair of primal and dual feasible solutions are optimal to the respective problems in a primal–dual pair of LPs iff whenever these feasible solutions make a slack variable in one problem strictly positive, the value (in these feasible solutions) of the associated nonnegative variable of the other problem is zero.

For the primal–dual pair (2.40) the theorem has the following interpretation. Whenever

$$v_i = \sum_{j=1}^n a_{ij}x_j - b_i > 0 \quad \text{we have } \lambda_i = 0 \quad (2.44a)$$

$$u_j = c_j - \sum_{i=1}^m a_{ij}\lambda_i > 0 \quad \text{we have } x_j = 0 \quad (2.44b)$$

Alternatively, we have

$$v_i\lambda_i = \left(\sum_{j=1}^n a_{ij}x_j - b_i \right) \lambda_i = 0, \quad i = 1, \dots, m \quad (2.45a)$$

$$u_j x_j = \left(c_j - \sum_{i=1}^m a_{ij} \lambda_i \right) x_j = 0, \quad j = 1, \dots, n \quad (2.45b)$$

Remark 1 Conditions (2.44) or (2.45) only require that if $v_i > 0$, then $\lambda_i = 0$. They do not require that if $v_i = 0$, then λ_i must be > 0 ; that is, both v_i and λ_i could be zero and the conditions of the complementary slackness theorem would be satisfied. Moreover, conditions (2.44) and (2.45) automatically imply that if $\lambda_i > 0$, then $v_i = 0$, and that if $x_j > 0$, then $u_j = 0$.

Remark 2 The complementary slackness theorem does not say anything about the values of unrestricted variables (corresponding to equality constraints in the other problem) in a pair of optimal feasible solutions of the primal and dual problems, respectively. It is concerned only with nonnegative variables of one problem and the slack variables corresponding to the associated inequality constraints in the other problem of primal-dual LPs.

Corollary 2.5.1 Consider a primal-dual pair of LPs. Let \mathbf{x}^* be an optimal feasible solution of the primal LP. Then the following statements can be made about every dual optimum feasible solution.

1. If x_j is a variable restricted to be nonnegative in the primal problem and if $x_j^* > 0$, then the dual inequality constraint associated with the primal variable x_j is satisfied as an equation by every dual optimal feasible solution.
2. If the primal problem consists of any inequality constraints, let \mathbf{v}^* represent the values of the corresponding slack variables at the primal feasible solution \mathbf{x} . Then if a slack variable $v_i > 0$, the dual variable associated with it is equal to zero in every dual optimal feasible solution.

Corresponding symmetric statements can be made about the nonnegatively restricted dual variables in $\boldsymbol{\lambda}^*$ when they take positive values, and primal inequality constraints that must hold as equalities in every primal optimal solution.

Proposition 2.5.4 (Necessary and Sufficient Optimality Conditions) Let \mathbf{x} and $\boldsymbol{\lambda}$ be the vectors of variables in an LP and its dual, respectively. If $\hat{\mathbf{x}}$ is a primal feasible solution it is an optimal solution iff there exists a dual feasible solution $\hat{\boldsymbol{\lambda}}$ such that $\hat{\mathbf{x}}$ and $\hat{\boldsymbol{\lambda}}$ together satisfy the complementary slackness conditions for optimality in this primal-dual pair.

Proof: Follows directly from Theorems 2.5.3 and 2.5.4. ■

Given an optimal feasible solution of one of the problems in the primal-dual pair, the above results can be used to characterize the set of all optimal feasible solutions of the other problem.

2.5.3 Economic Interpretation

Consider the primal–dual pair given by (2.40). Assume that the primal problem represents a cost center, such as a government agency, that provides a set of services to its clients, and that the dual represents a consulting firm that can provide the same set of services with various complements of resources. For the former, suppose that the demand for service type i is b_i , $i = 1, \dots, m$, which can be met with n different types of resources. Let a_{ij} be the amount of service type i provided by one unit of resource j at cost c_j . The problem is to decide how many units of resource j to employ, denoted by x_j , so that all demand is met at minimum cost.

In contrast, the problem faced by the consulting firm is to determine how to price its services to maximize its profit while assuring that the cost to the customer is no greater than he can achieve by employing his own resources to provide the services. To construct a model, let λ_i be the price charged per unit of service type i . The constraints on the prices are $\sum_{i=1}^m \lambda_i a_{ij} \leq c_j$ for all $j = 1, \dots, n$. Given these prices, the cost center checks whether it can meet its service levels less expensively by contracting with the consulting firm rather than by acquiring or employing its own resources. Because the cost per unit of resource j is c_j , and it can get the same complement of service provided by one unit of resource j from the consulting firm at a cost of $\sum_{i=1}^m \lambda_i a_{ij}$, it would not be worthwhile for the cost center to use any amount of resource j if $\sum_{i=1}^m \lambda_i a_{ij} < c_j$. In this case, then, we have $x_j = 0$.

Conversely, if the consulting firm associates a positive price per unit of service type i ($\lambda_i > 0$), then the cost center would do best by meeting exactly its service demands. This implies that when $\lambda_i > 0$ the cost center will try to satisfy $\sum_{j=1}^n a_{ij} x_j = b_i$. A similar interpretation can be given about the prices that the consulting firm will adopt knowing the amounts of resources that the cost center is using.

These are precisely the complementarity slackness conditions for the primal–dual pair (2.40). When they are satisfied, there is no incentive for the cost center to change the amount of resources it employs or for the consulting firm to change its prices. The minimum cost incurred by the former is exactly the maximum revenue realized by the latter. This results in an economic equilibrium where cost $(\sum_{j=1}^n c_j x_j) =$ revenue $(\sum_{i=1}^m \lambda_i b_i)$.

2.5.4 Sensitivity Analysis

We have just seen that the optimal values of the dual variables in a linear programming problem can be interpreted as prices. In this section this interpretation is explored in further detail.

Suppose in an LP in standard form (2.1) the optimal basis is \mathbf{B} with solution $(\mathbf{x}_B, \mathbf{0})$, where $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$. A solution to the corresponding dual problem is $\boldsymbol{\lambda}^T = \mathbf{c}_B \mathbf{B}^{-1}$. Now, assuming nondegeneracy, small changes in the vector \mathbf{b} will not cause the optimal basis to change. Thus for $\mathbf{b} + \Delta\mathbf{b}$ the optimal solution is

$$\mathbf{x} = (\mathbf{x}_B + \Delta\mathbf{x}_B, \mathbf{0})$$

where $\Delta\mathbf{x}_B = \mathbf{B}^{-1}\Delta\mathbf{b}$. Thus the corresponding increment in the cost function is

$$\Delta f = \mathbf{c}_B \Delta\mathbf{x}_B = \boldsymbol{\lambda}^T \Delta\mathbf{b}$$

This equation shows that $\boldsymbol{\lambda}$ gives the sensitivity of the optimal cost with respect to small changes in the vector \mathbf{b} . In other words, if a new problem were solved with \mathbf{b} changed to $\mathbf{b} + \Delta\mathbf{b}$, the change in the optimal value of the objective function would be $\boldsymbol{\lambda}^T \Delta\mathbf{b}$.

This interpretation says that λ_i directly reflects the change in cost due to a change in the i th component of the vector \mathbf{b} . Thus λ_i may be viewed equivalently as the *marginal price* of the component b_i , since if b_i is changed to $b_i + \Delta b_i$ the value of the optimal solution changes by $\lambda_i \Delta b_i$. When the constraints $\mathbf{A}\mathbf{x} = \mathbf{b}$ are written as $\mathbf{Ax} \geq \mathbf{b}$ as in (2.40), the dual variables are nonnegative implying that for λ_i positive, a positive change in b_i will produce an increase in the objective function value. In economic terms, it is common to refer to the dual variables as shadow prices.

Example 2.5.1 The shadow prices are associated with constraints but they are often used to evaluate prices or cost coefficients associated with variables of the primal problem. As an example, suppose we have an \mathbf{A} matrix which represents the daily operation of an oil refinery and a particular variable x_j representing the purchase of crude oil feedstock, with a cost of \$22.65/barrel ($c_j = 22.65$). The refinery wants to minimize its costs. There is an upper limit on the purchase of this oil of 50,000 barrels/day at this price. This is represented by the constraint

$$x_j + x_s = 50,000$$

where x_s is the associated slack variable. Assume at the optimal x_s has a reduced cost of -\$2.17/barrel: what does this mean?

The shadow price on the constraint is -\$2.17/barrel but this does not mean that we should only pay \$2.17 for another barrel of crude. It means we should be prepared to pay another \$2.17/barrel for an opportunity to purchase extra supplies given that any further purchases would cost \$22.65/barrel; i.e., the objective function will decrease by \$2.17 for every extra barrel we can purchase at the price c_j already in the cost row. This means we should be prepared to bid up to $22.65 + 2.17 = \$25.82/\text{barrel}$ on the spot market for extra supplies of that crude. Note that \$25.82/barrel is the breakeven price, in that we decrease our objective function f if we can purchase a barrel for less than this price, increase f if we purchase for more, and make no change at all to f if we purchase for exactly \$25.82/barrel.

The reduced cost of a variable nonbasic at its lower bound is often referred to as the *opportunity cost* of that variable. If management made the (nonoptimal) decision of increasing that nonbasic from its lower bound, the reduced cost gives the increase in f per unit increase in the variable (for a certain range). This represents the opportunity loss in departing from the optimal solution.

Ranging: For reasons that practitioners understand implicitly, it is often said that postoptimality analysis is the most important part of the LP calculations. The majority of the coefficients that appear in an LP are rarely known with certainty and so have to be estimated from historical or empirical data. Under these circumstances we would like to know the range of variation of these coefficients for which the optimal solution remains optimal; i.e., the basis does not change. Three categories are investigated below: cost coefficients c_j , right-hand-side terms b_i , and matrix coefficients a_{ij} .

Changes in the Cost Row

(a) *Nonbasic variable* The change in the cost coefficient of a nonbasic variable affects the reduced cost of that variable only, and the change is in direct proportion. If δ is a perturbation associated with the original cost coefficient c_q , then at optimality we can write the reduced cost coefficient of nonbasic variable x_q as $\bar{c}_q(\delta) = c_q + \delta - \boldsymbol{\lambda}^T \mathbf{a}_q$. In order for the current basis \mathbf{B} to remain optimal, we must have $\bar{c}_q(\delta) \geq 0$. This means

$$\delta \geq \boldsymbol{\lambda}^T \mathbf{a}_q - c_q = -\bar{c}_q$$

The reduced cost coefficients of all the other variables are independent of c_q and so will remain nonnegative. If a δ is chosen that violates this inequality, x_q would be identified as the entering variable and we would continue the application of the simplex method until a terminal basis for the modified problem was found.

It is worth mentioning that in most commercial LP codes there is another range given at the same time as well — the range over which x_q can be increased from zero before a change of basis occurs. When $\delta = -\bar{c}_q$, the reduced cost is zero implying that x_q can be increased without affecting the value of the objective function. The maximum value it can take without effecting a change in basis is given by $\min_i \{\bar{b}_i / \alpha_{iq} : \alpha_{iq} > 0\}$ which is the minimum ratio test in Step 3 of the simplex method.

(b) *Basic variable* A change in the cost coefficient of a basic variable may affect the reduced cost of all the nonbasic variables. Let \mathbf{e}_i be the i th unit vector of length m and suppose we increment the cost coefficient of the i th basic variable by δ ; i.e., $\mathbf{c}_B \leftarrow \mathbf{c}_B + \delta \mathbf{e}_i^T$. This gives $\boldsymbol{\lambda}^T(\delta) = (\mathbf{c}_B + \delta \mathbf{e}_i^T) \mathbf{B}^{-1}$ so the dual vector is an affine function of δ . The reduced cost of the q th nonbasic variable is now

$$\begin{aligned}\bar{c}_q(\delta) &= c_q - (\mathbf{c}_B + \delta \mathbf{e}_i^T) \mathbf{B}^{-1} \mathbf{a}_q \\ &= c_q - \mathbf{c}_B \mathbf{B}^{-1} \mathbf{a}_q - \delta \mathbf{e}_i^T \mathbf{B}^{-1} \mathbf{a}_q \\ &= \bar{c}_q - \delta \alpha_{iq}\end{aligned}$$

where $\alpha_{iq} = (\mathbf{B}^{-1} \mathbf{a}_q)_i$ is the i th component of the updated column \mathbf{a}_q . This value is found for the nonbasic variable x_q by solving $\mathbf{B}^T \mathbf{y} = \mathbf{e}_i$ for \mathbf{y} , then computing $\alpha_{iq} = \mathbf{y}^T \mathbf{a}_q$. (Obviously, if $\alpha_{iq} = 0$ for any x_q , the reduced cost does not change.)

For a solution to remain optimal, we must have $\bar{c}_q(\delta) \geq 0$, or

$$\bar{c}_q - \delta \alpha_{iq} \geq 0 \quad \forall q \quad (2.46)$$

where \bar{c}_q is the reduced cost at the current optimum. This constraint produces bounds on δ . For a basic variable, the *range* over which c_i can vary and the current solution remain optimal is given by $c_i + \delta$, where

$$\max_q \left\{ \frac{\bar{c}_q}{\alpha_{iq}} \mid \alpha_{iq} < 0 \right\} \leq \delta \leq \min_q \left\{ \frac{\bar{c}_q}{\alpha_{iq}} \mid \alpha_{iq} > 0 \right\}$$

since this is the range for which (2.46) is satisfied. If there is no $\alpha_{iq} > 0$, then $\delta < \infty$ likewise if there is no $\alpha_{iq} < 0$, then $\delta > -\infty$.

Example 2.5.2 Suppose we have an optimal solution to an LP given in tableau form with attached variables

$$\begin{aligned} \min_{\mathbf{x} \geq \mathbf{0}} \quad & -f = -31.5 - 3.5x_4 - 0.1x_3 - 0.25x_5 \\ \text{subject to} \quad & x_1 = 3.2 - 1.0x_4 - 0.5x_3 - 0.6x_5 \\ & x_2 = 1.5 + 0.5x_4 + 1.0x_3 - 1.0x_5 \\ & x_6 = 5.6 - 2.0x_4 - 0.5x_3 - 1.0x_5 \end{aligned} \quad (2.47)$$

If the cost coefficient of x_2 becomes $c_2 + \delta$, the reduced costs of the nonbasic variables become:

$$\begin{aligned} x_4 : \bar{c}_4(\delta) &= 3.5 - \delta(-0.5) \\ x_3 : \bar{c}_3(\delta) &= 0.1 - \delta(-1.0) \\ x_5 : \bar{c}_5(\delta) &= 0.25 - \delta(+1.0) \end{aligned}$$

Note that $x_{B_i} = \bar{b}_i - \sum_{j \in \{3,4,5\}} \alpha_{ij} x_j$ for $i = 1, 2, 6$, so α_{ij} is the negative of the number appearing in equations (2.47). The range that δ can take is given by

$$\begin{aligned} \max \left\{ \frac{3.5}{-0.5}, \frac{0.1}{-1.0} \right\} \leq \delta &\leq \min \left\{ \frac{0.25}{1.0} \right\} \\ -0.1 \leq \delta &\leq 0.25 \end{aligned}$$

When δ assumes one of the limits of its range, a reduced cost becomes zero. In this example, for $\delta = -0.1$ the reduced cost of x_3 is zero, so that if the cost coefficient of x_2 decreases by 0.1 or more it becomes advantageous for x_3 to become active. The minimum ratio test: $\min\{3.2/0.5, 5.6/0.5\}$ then indicates that 6.4 units of x_3 can be produced before x_1 becomes zero and a change of basis is required.

Changes in the Right-Hand-Side Vector

We wish to investigate the effect of a change $b_i \leftarrow b_i + \delta$ for some $1 \leq i \leq m$. It is usual to consider the case where b_i is the right-hand side of an inequality constraint, which therefore has a slack variable associated with it. The goal is to determine the range over which the current solution remains optimal. If the constraint is an equality, it can be analyzed by regarding its associated artificial variable as a positive slack (which must be nonbasic for a feasible solution).

(a) *Basic slack variable:* If the slack variable associated with the i th constraint is basic the constraint is not binding at the optimum. The analysis is simple: the value of the slack gives the range over which the right-hand side b_i can be reduced (increased for \geq constraint). The solution remains feasible and optimal for the range $b_i + \delta$, where

$$\begin{aligned}\hat{x}_s &\leq \delta < \infty && \text{for } \leq \text{ type constraint} \\ -\infty &< \delta \leq \hat{x}_s && \text{for } \geq \text{ type constraint}\end{aligned}$$

where \hat{x}_s is the value of the associated slack variable.

(b) *Nonbasic slack variable:* If a slack variable is nonbasic at zero, then the original inequality constraint is binding at the optimum. At first sight it would seem that because the constraint is binding, there is no possibility of changing the right-hand side term, particularly in decreasing the value of b_i (for \leq type constraints). It turns out that by changing the vector \mathbf{b} we also change \mathbf{x}_B ($= \mathbf{B}^{-1}\mathbf{b}$) so there is a range over which \mathbf{x}_B remains nonnegative. For the associated values we still retain an optimal feasible solution in the sense that the basis does not change. (Note that both \mathbf{x}_B and $f = \mathbf{c}_B \mathbf{x}_B$ change value.)

Consider the constraint

$$a_{k1}x_1 + a_{k2}x_2 + \cdots + x_s = b_k \quad (2.48)$$

where x_s is the slack variable. If the right-hand side becomes $b_k + \delta$, rearranging (2.48) gives

$$a_{k1}x_1 + a_{k2}x_2 + \cdots + (x_s - \delta) = b_k \quad (2.49)$$

so that $(x_s - \delta)$ replaces x_s . Thus if x_s is nonbasic at zero in the final tableau, we have the expression

$$\mathbf{x}_B = \bar{\mathbf{b}} - \boldsymbol{\alpha}_s(-\delta)$$

where $\boldsymbol{\alpha}_s$ is the updated column in the tableau corresponding to x_s . Because \mathbf{x}_B must remain nonnegative, we have $\bar{\mathbf{b}} + \delta\boldsymbol{\alpha}_s \geq 0$ which is used to solve for the range over which δ can vary.

$$\max_i \left\{ \frac{\bar{b}_i}{-\alpha_{is}} \mid \alpha_{is} > 0 \right\} \leq \delta \leq \min_i \left\{ \frac{\bar{b}_i}{-\alpha_{is}} \mid \alpha_{is} < 0 \right\}$$

If there is no $\alpha_{is} > 0$, then $\delta > -\infty$; if there is no $\alpha_{is} < 0$, then $\delta < \infty$.

For \geq type constraints, δ changes sign. This follows because we can analyze $\sum_{j=1}^n a_{ij}x_j \geq b_i$ in the form $-\sum_{j=1}^n a_{ij}x_j \leq -b_i$, so that $-(x_s + \delta)$ replaces $(x_s - \delta)$ in eq. (2.49). Another way of seeing this is to consider the change to the right-hand side in the form

$$\mathbf{b}(\delta) = \mathbf{b} + \delta \mathbf{e}_k$$

Thus the new value of \mathbf{x}_B is given by

$$\begin{aligned}\mathbf{x}_B(\delta) &= \mathbf{B}^{-1}\mathbf{b}(\delta) = \mathbf{B}^{-1}\mathbf{b} + \delta \mathbf{B}^{-1}\mathbf{e}_k \\ &= \bar{\mathbf{b}} + \delta \mathbf{B}^{-1}\mathbf{e}_k\end{aligned}$$

But as

$$\alpha_s = \mathbf{B}^{-1}\mathbf{e}_k \quad \text{for a } \leq \text{ type constraint}$$

and as

$$\alpha_s = -\mathbf{B}^{-1}\mathbf{e}_k \quad \text{for a } \geq \text{ type constraint}$$

since the column corresponding to the slack variable is $+\mathbf{e}_k$ for a \leq constraint and $-\mathbf{e}_k$ for a \geq constraint. Thus we have

$$\begin{aligned}\bar{\mathbf{b}} - \alpha_s(-\delta) &\geq \mathbf{0} \quad \text{for a } \leq \text{ type constraint, and} \\ \bar{\mathbf{b}} - \alpha_s(+\delta) &\geq \mathbf{0} \quad \text{for a } \geq \text{ type constraint}\end{aligned}$$

Example 2.5.3 Consider Example 2.5.2 again and suppose x_4 represents a slack variable for a particular constraint i (\leq type). If the coefficient b_i is varied by an amount δ , we have

$$\begin{aligned}x_1(\delta) &= 3.2 - 1.0(-\delta) \quad \text{that is, } \bar{\mathbf{b}} = (3.2, 1.5, 5.6)^T \\ x_2(\delta) &= 1.5 + 0.5(-\delta) \quad \alpha_s = (1.0, -0.5, 2.0)^T \\ x_6(\delta) &= 5.6 - 2.0(-\delta)\end{aligned}$$

Thus

$$\begin{aligned}x_1(\delta) \geq 0 \text{ for } 3.2 - 1.0(-\delta) \geq 0, \text{ that is, } \delta &\geq \frac{3.2}{-1.0} \\ x_2(\delta) \geq 0 \text{ for } 1.5 + 0.5(-\delta) \geq 0, \text{ that is, } \delta &\leq \frac{1.5}{0.5} \\ x_6(\delta) \geq 0 \text{ for } 5.6 - 2.0(-\delta) \geq 0, \text{ that is, } \delta &\geq \frac{5.6}{-2.0}\end{aligned}$$

Therefore, δ can vary in the range

$$\max \left\{ \frac{3.2}{-1.0}, \frac{5.6}{-2.0} \right\} \leq \delta \leq \min \left\{ \frac{1.5}{0.5} \right\}$$

$$-2.8 \leq \delta \leq 3.0$$

Changes in Matrix Coefficients

The technological coefficients a_{ij} are usually known with much more certainty than the cost row or right-hand-side vector, since they customarily represent some physical interaction between variables and are not subject to the same market fluctuations as costs and demands. We shall consider changes to the coefficients of nonbasic variables only; changes to coefficients of basic variables alters the basis matrix \mathbf{B} and are rather complicated to analyze (see [M18]).

Consider the j th nonbasic variable with corresponding column \mathbf{a}_j . If the i th element of \mathbf{a}_j is changed by an amount δ , this affects the reduced cost \bar{c}_j as follows.

If

$$\mathbf{a}_j(\delta) = \mathbf{a}_j + \delta \mathbf{e}_i$$

then

$$\begin{aligned}\bar{c}_j(\delta) &= c_j - \boldsymbol{\lambda}^T (\mathbf{a}_j + \delta \mathbf{e}_i) \\ &= \bar{c}_j - \delta \boldsymbol{\lambda}^T \mathbf{e}_i \\ &= \bar{c}_j - \delta \lambda_i\end{aligned}$$

where $\boldsymbol{\lambda}^T (= \mathbf{c}_B \mathbf{B}^{-1})$ is the dual vector. Thus the solution remains optimal as long as $\bar{c}_j(\delta) \geq 0$. The corresponding range for δ is

$$\delta \leqq \frac{\bar{c}_j}{\lambda_i} \quad \text{for } \lambda_i > 0$$

$$\delta \geqq \frac{\bar{c}_j}{\lambda_i} \quad \text{for } \lambda_i < 0$$

2.5.5 Dual Simplex Method

The guiding principle of the simplex method is to maintain primal feasibility ($x_{B_i} = \bar{b}_i \geq 0$, $i = 1, \dots, m$) and complementary slackness ($x_j \bar{c}_j = 0$, for all j) at each iteration while working toward nonnegative reduced costs ($\bar{c}_j \geq 0$, for all j). This last condition is known as dual feasibility because it is equivalent to satisfying the dual constraints $\boldsymbol{\lambda}^T \mathbf{A} \leqq \mathbf{c}$, where $\boldsymbol{\lambda}^T = \mathbf{c}_B \mathbf{B}^{-1}$. Any basis that satisfies primal feasibility and dual feasibility automatically satisfies the complementary slackness condition and is hence optimal. In this section, we present the dual simplex algorithm which maintains dual feasibility and complementarity at each iteration while striving for primal feasibility.

We are interested in such an algorithm because often we have available a basic solution to an LP that is not feasible but which prices out optimally; that is, the corresponding multipliers are feasible for the dual problem. In the simplex tableau this situation

corresponds to having no negative reduced cost coefficients but an infeasible basic solution. Such a situation may arise, for example, if a solution to a certain LP is calculated and then a new problem is constructed by changing the vector \mathbf{b} . Here, a basic feasible solution to the dual is available and hence it is desirable to pivot in such a way as to optimize the dual.

Rather than constructing a tableau for the dual problem (which, if the primal is in standard form, involves m free variables and n nonnegative slack variables), it is more efficient to work on the dual from the primal tableau. The complete technique based on this idea is the dual simplex method.

Given the linear programming problem in standard form: $\min\{\mathbf{c}\mathbf{x} : \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$, suppose a basis \mathbf{B} is known such that $\boldsymbol{\lambda}^T = \mathbf{c}_B \mathbf{B}^{-1}$ is feasible for the dual. In this case we say that the corresponding basic solution to the primal, $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$, is *dual feasible*. If $\mathbf{x}_B \geq \mathbf{0}$ then this solution is also primal feasible and hence optimal.

The given vector $\boldsymbol{\lambda}$ is feasible for the dual and thus satisfies $\boldsymbol{\lambda}^T \mathbf{a}_j \leq c_j$, for $j = 1, \dots, n$. Indeed, assuming as usual that the basis is the first m columns of \mathbf{A} , we have m equalities

$$\boldsymbol{\lambda}^T \mathbf{a}_j = c_j, \quad j = 1, \dots, m$$

and (barring degeneracy in the dual) $n - m$ inequalities

$$\boldsymbol{\lambda}^T \mathbf{a}_j < c_j, \quad j = m + 1, \dots, n$$

To develop one cycle of the dual simplex method, we find a new vector $\hat{\boldsymbol{\lambda}}$ such that one of the equalities becomes an inequality and one of the inequalities becomes an equality, while at the same time increasing the value of the dual objective function. The m equalities in the new solution then determine a new basis.

Denote the i th row of \mathbf{B}^{-1} by β_i . Then for

$$\hat{\boldsymbol{\lambda}}^T = \boldsymbol{\lambda}^T - \theta \beta_i$$

we have $\hat{\boldsymbol{\lambda}}^T \mathbf{a}_j = \boldsymbol{\lambda}^T \mathbf{a}_j - \theta \beta_i \mathbf{a}_j$. Noting that $\beta_i \mathbf{a}_j = \alpha_{ij}$, the ij th element of the tableau, we have

$$\hat{\boldsymbol{\lambda}}^T \mathbf{a}_j = c_j, \quad j = 1, \dots, m; \quad j \neq i \tag{2.50a}$$

$$\hat{\boldsymbol{\lambda}}^T \mathbf{a}_j = c_i - \theta \tag{2.50b}$$

$$\hat{\boldsymbol{\lambda}}^T \mathbf{a}_j = \boldsymbol{\lambda}^T \mathbf{a}_j - \theta \alpha_{ij}, \quad j = m + 1, \dots, n \tag{2.50c}$$

Also,

$$\hat{\boldsymbol{\lambda}}^T \mathbf{b} = \boldsymbol{\lambda}^T \mathbf{b} - \theta \mathbf{x}_{B_i} \tag{2.51}$$

These last equations lead directly to the algorithm.

- Step 1 Given a dual feasible basic solution \mathbf{x}_B , if $\mathbf{x}_B \geq \mathbf{0}$ the solution is optimal. If \mathbf{x}_B contains one or more negative components, select an index i such that $x_{B_i} < 0$.
- Step 2 If all $\alpha_{ij} \geq 0$, $j = 1, \dots, n$, then the dual has no maximum (this follows from (2.50) because λ is feasible for all $\theta > 0$). If $\alpha_{ij} < 0$ for some j , let

$$\theta = \frac{\bar{c}_p}{-\alpha_{ip}} = \min_j \left\{ \frac{\bar{c}_j}{-\alpha_{ij}} \mid \alpha_{ij} < 0 \right\} \quad (2.52)$$

where p is the index that corresponds to the minimum ratio.

- Step 3 Form a new basis B by replacing a_i with a_p . Using this basis determine the corresponding basic dual feasible solution \mathbf{x}_B and return to Step 1.

At Step 1 we choose a leaving variable, usually the most negative; at Step 2 we choose an entering variable based on a ratio test whose primal counterpart is (2.23). The proof that the algorithm converges to the optimal solution is similar in detail to the proof for the primal simplex method. The essential observations are: (a) from the choice of p in (2.52) and from (2.50) the new solution will again be dual feasible; (b) by (2.51) and the choice $x_{B_i} < 0$, the value of the dual objective will increase; (c) the procedure cannot terminate at a nonoptimal point; and (d) since there are only a finite number of bases, the optimum must be achieved in a finite number of steps when a mechanism for taking degeneracy into account is included.

Example 2.5.4 A form of problem arising frequently is that of minimizing a positive combination of nonnegative variables subject to a series of "greater than" type inequalities having positive coefficients. Such problems, as given below, are natural candidates for application of the dual simplex procedure.

$$\begin{aligned} \min f = & 3x_1 + 4x_2 + 5x_3 \\ \text{subject to } & x_1 + 2x_2 + 3x_3 \geq 5 \\ & 2x_1 + 2x_2 + x_3 \geq 6 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

By introducing surplus variables x_4 and x_5 , and by changing the sign of the inequalities we obtain the initial tableau:

	x_1	x_2	x_3	x_4	x_5	\bar{b}
x_4	-1	-2	-3	1	0	-5
x_5	(-2)	-2	-1	0	1	-6
$-f$	3	4	5	0	0	0

Initial tableau

The basis corresponds to a dual feasible solution since all of the \bar{c}_j 's are nonnegative. We select any $x_{B_i} < 0$, say $x_5 = -6$, to remove from the set of basic variables. To find the appropriate pivot element in the second row we compute the ratios $-\bar{c}_j/\alpha_{2j}$ and select the minimum positive value: $\min\{3/2, 4/2, 5/1\} = 1.5$ corresponding to x_1 . This yields the indicated pivot. Continuing, the remaining tableaus are

	x_1	x_2	x_3	x_4	x_5	\bar{b}
x_4	0	(-1)	-5/2	1	-1/2	-2
x_1	1	1	1/2	0	-1/2	3
$-f$	0	1	7/2	0	3/2	-9

Second tableau

	x_1	x_2	x_3	x_4	x_5	\bar{b}
x_2	0	1	5/2	-1	1/2	2
x_1	1	0	-2	1	-1	1
$-f$	0	0	1	1	1	-11

Final tableau

The third tableau yields a feasible solution to the primal which must be optimal. Thus the solution is $x_1 = 1$, $x_2 = 2$, $x_3 = 0$.

INTEGER PROGRAMMING

3.1 INTRODUCTION

There are many situations in which it is desirable for some or all of the variables in a mathematical programming model to take on integer values. The corresponding models are said to be integer programs. Three general cases include the *integer linear programming* (ILP) problem:

$$\begin{aligned} \min f(\mathbf{x}) &= \mathbf{c}\mathbf{x} \\ \text{subject to } &\mathbf{Ax} = \mathbf{b} \\ &\mathbf{x} \geq \mathbf{0} \text{ and integer} \end{aligned} \tag{3.1}$$

where \mathbf{c} is an n -dimensional row vector, \mathbf{b} is an m -dimensional column vector, \mathbf{A} is an $m \times n$ matrix, and \mathbf{x} is an n -dimensional decision vector restricted to be nonnegative and integer (written alternatively as $\mathbf{x} \in \mathbb{Z}_+^n$); the *mixed-integer linear programming* (MILP) problem:

$$\begin{aligned} \min f(\mathbf{x}, \mathbf{y}) &= \mathbf{c}\mathbf{x} + \mathbf{d}\mathbf{y} \\ \text{subject to } &\mathbf{Ax} + \mathbf{By} = \mathbf{b} \\ &\mathbf{x} \geq \mathbf{0} \text{ and integer} \\ &\mathbf{y} \geq \mathbf{0} \end{aligned} \tag{3.2}$$

where \mathbf{d} is a s -dimensional row vector, \mathbf{B} is a $m \times s$ matrix, and \mathbf{y} is a s -dimensional vector of nonnegative continuous variables; and the *general mixed-integer programming* problem:

$$\begin{aligned} \min f(\mathbf{x}, \mathbf{y}) \\ \text{subject to } &\mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{b} \\ &\mathbf{x} \geq \mathbf{0} \text{ and integer} \\ &\mathbf{y} \geq \mathbf{0} \end{aligned} \tag{3.3}$$

where $f : R^n \times R^s \rightarrow R^1$ and $\mathbf{h} : R^n \times R^s \rightarrow R^m$. In the first two cases it is assumed without loss of generality that all model parameters are integer valued.

In model (3.1), when all the variables are restricted to be either 0 or 1, the resultant formulation is called a pure 0-1 integer program (IP). In model (3.2), any constraints separable in the variables, i.e., of the form $\mathbf{A}_1\mathbf{x} = \mathbf{b}_x$ or $\mathbf{B}_1\mathbf{y} = \mathbf{b}_y$, are assumed to be included in the equations $\mathbf{Ax} + \mathbf{By} = \mathbf{b}$, where the matrices \mathbf{A}_1 and \mathbf{B}_1 , and the vectors \mathbf{b}_x and \mathbf{b}_y are of conformal dimension. Similarly, for model (3.3), constraints of the form $\mathbf{h}_1(\mathbf{x}) = \mathbf{b}_x$ and $\mathbf{h}_2(\mathbf{y}) = \mathbf{b}_y$ are assumed to be included in $\mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{b}$.

It is somewhat surprising that there is no algorithm for solving integer programs that has an efficiency comparable to that of the simplex method or any of the interior point methods for linear programming — surprising because integers are usually thought of as being more elementary than real numbers. The algorithms for IPs that are now available in commercial software packages can handle models with up to a few hundred integer variables. The actual number depends strongly on the structure of the model, the ingenuity with which any special structure is taken into account, and the nature of the computer facilities. This is in contrast to most linear programming packages which can handle a huge number of variables and constraints.

Given this difference in the difficulty of solving integer and linear programs, it is worthwhile to ask when an IP can be tackled by simply omitting the restrictions of the decision variables to integer values and solving the corresponding LP. There are two circumstances in which this is possible. In particular, there are certain types of linear models, such as transportation models whose special structure guarantees that if an optimal solution exists, there is an optimal integer solution. The underlying property of the \mathbf{A} matrix is called *total unimodularity* and is further discussed in Section 3.5. In models of this sort any restrictions of decision variables to integer values can be omitted.

The second situation where the integrality requirement can be relaxed occurs when the range of a variable is large enough so that rounding its value in an optimal LP solution to the closest integer value will lead to approximately the same objective function value. In a production type model, for example, each variable may represent the output of an item that can be manufactured in unit quantities such as televisions or microprocessors. Solving such a model as an LP and then rounding is usually quite satisfactory. This is especially true in many practical applications where the real errors are in the input data and not the rounded results.

Apart from these two circumstances, integrality restrictions are a crucial part of the model. As a simple illustration, consider the 0-1 knapsack problem:

$$\begin{aligned} \min f &= -24x_1 - 28x_2 - 23x_3 - 19x_4 - 32x_5 \\ \text{subject to } &29x_1 + 37x_2 + 31x_3 + 26x_4 + 44x_5 \leq 140 \\ &x_j = 0 \text{ or } 1, \quad j = 1, 2, \dots, 5 \end{aligned} \tag{3.4}$$

The optimal solution to (3.4) is $\mathbf{x}^* = (1, 1, 0, 1, 1)$, $f^* = 103$. However, the continuous LP optimum is $\mathbf{x}_{LP} = (1, 1, 1, 1, \frac{17}{44})$, $f_{LP} = 106.36$. Rounding $x_5 = \frac{17}{44}$ up to 1 produces an infeasible solution, while rounding x_5 down to 0 produces the feasible solution $\mathbf{x}' = (1, 1, 1, 1, 0)$, $f' = 94$. The latter is far from optimal since $f' = 94$ is only 91% of $f^* = 103$. Moreover, there is no straightforward way of obtaining the optimum from \mathbf{x}' .

3.1.1 Models with Integer Variables

In this section, we describe how IP models can be formulated for a number of different situations. For the production model mentioned above, if integer solutions were actually required because only a few units of each item were going to be manufactured, converting the LP model to an IP model is immediate by adding the requirement that the decision variables have integer values. In many cases, though, this is not possible; rather one must proceed by introducing special integer-valued variables into the model.

Disjunctive constraints: Suppose that we have a linear programming problem in which a choice can be made between two resources. Although each resource gives rise to a constraint, only one of the constraints needs to be considered. As an example, suppose we have the constraints:

$$\begin{aligned} &\text{either } 7x_1 + 4x_2 \leq 19 \\ &\text{or } 2x_1 + 5x_2 \leq 14 \end{aligned}$$

This is equivalent to the system

$$\begin{aligned} 7x_1 + 4x_2 &\leq 19 + M\delta \\ 2x_1 + 5x_2 &\leq 14 + M(1 - \delta) \\ \delta &= 0 \text{ or } 1 \end{aligned}$$

where M is a very large positive number (large enough so adding it to the right-hand side causes the constraint to become redundant). Now, when $\delta = 0$, the first constraint holds and the second is inactive (redundant); when $\delta = 1$, the reverse is true. The variable δ is called a binary variable because it can take on values of only 0 or 1.

k out of m constraints must hold: The previous situation can be extended to the case in which only k of a possible m constraints must hold, where $k < m$. Suppose we have

$$g_1(\mathbf{x}) \leq b_1$$

$$g_2(\mathbf{x}) \leq b_2$$

$$\vdots$$

$$g_m(\mathbf{x}) \leq b_m$$

Then the equivalent formulation is

$$\begin{aligned} g_1(\mathbf{x}) &\leq b_1 + M\delta_1 \\ g_2(\mathbf{x}) &\leq b_2 + M\delta_2 \\ &\vdots \\ g_m(\mathbf{x}) &\leq b_m + M\delta_m \\ \delta_1 + \delta_2 + \cdots + \delta_m &= m - k \\ \delta_i &= 0 \text{ or } 1, \quad i = 1, \dots, m \end{aligned}$$

Here, when $\delta_i = 1$, the i th constraint becomes redundant. Thus we must eliminate $m - k$ constraints to ensure that k constraints hold. Requiring the sum of the δ_i to equal $m - k$ accomplishes just this because each δ_i is binary.

Conditional constraints: Suppose we have a situation in which one constraint strictly holding forces a second constraint to hold; i.e.,

$$g(\mathbf{x}) < b \Rightarrow h(\mathbf{x}) \leq d \quad (3.5)$$

Now, because $g(\mathbf{x}) \geq b$ places no requirement on the value of $h(\mathbf{x})$ we can formulate this situation by using an approach similar to that used for the disjunctive constraints. Specifically, since $p \Rightarrow q$ is equivalent to (not p) or q or both, (3.5) is equivalent to

$$g(\mathbf{x}) \geq b \text{ or } h(\mathbf{x}) \leq d \text{ or both}$$

which can be transformed into

$$\begin{aligned} g(\mathbf{x}) &\geq b - M(1 - \delta) \\ h(\mathbf{x}) &\leq d + M\delta \\ \delta &= 0 \text{ or } 1 \end{aligned}$$

To see this, observe that $\delta = 0$ yields $h(\mathbf{x}) \leq d$ with $g(\mathbf{x})$ unrestricted; when $\delta = 1$ we have $g(\mathbf{x}) \geq b$ and $h(\mathbf{x})$ unrestricted.

Discrete-valued variables and functions: Suppose that a decision variable can only take on one of a finite set of values; i.e., $x_j \in D_j = \{d_1, d_2, \dots, d_r\}$. We can model this as

$$x_j = d_1\delta_1 + d_2\delta_2 + \cdots + d_r\delta_r \quad (3.6a)$$

$$\delta_1 + \delta_2 + \cdots + \delta_r = 1 \quad (3.6b)$$

$$\delta_i = 0 \text{ or } 1, \quad i = 1, \dots, r \quad (3.6c)$$

This arrangement allows only one of the δ_i variables to be positive, which in turn forces x_j to take the value of the corresponding d_i . Similarly, when a function $g(x_1, \dots, x_n)$ of the decision variables is restricted to one of a finite set of values we can use the same approach by simply substituting $g(x_i, \dots, x_n)$ for x_j in (3.6a).

When x_j is integer and lies in the interval $0 \leq x_j \leq u_j$, transformation (3.6a)–(3.6c) can be applied by defining $D_j = \{0, 1, \dots, u_j\}$; however, in this simple case a transformation requiring few variables is

$$\begin{aligned} x_j &= \sum_{i=0}^{t_j} 2^i \delta_{ij} \leq u_j \\ \delta_{ij} &= 0 \text{ or } 1, \quad i = 1, \dots, t_j \end{aligned} \quad (3.7)$$

where $2^{t_j} \leq u_j < 2^{t_j+1}$.

Special ordered sets (SOS): When it is required to choose a single element from a set, such as in (3.6a)–(3.6c), we have what is called a *special ordered set*. Each SOS contains an ordered set of integer or continuous variables, x_j . The use of this structure leads to simplified models and efficient computations. Typically, two types of SOS are available in commercial mathematical programming packages. Type 1 (SOS1) have the restriction that exactly one x_j is nonzero; i.e., $\sum_j x_j = 1$. These constraints are inherent in assignment models and other network flow models. Type 2 (SOS2) have the restriction that at most 2 adjacent x_j are nonzero. These sets are used in the next subsection to model nonlinear functions.

Sometimes SOS1 are combined with the structure for convex combinations: $\sum_j x_j = 1$, $0 \leq x_j \leq 1$, to give what can be viewed as a Type 3 SOS. In this case, x_j is a continuous variable. A variation on this form is $\sum_{j \in J} x_j \leq u$, $x_j \geq 0$ for all $j \in J$. When every variable in the model appears in at most one constraint of this type, we have what are called *generalized upper bound* (GUB) constraints. The advantage of recognizing this structure is that the simplex method can be modified to handle implicitly GUB constraints just like it handles variable bounds [M18]. This reduces the size of the basis and hence the computational effort.

Piecewise linear approximation of nonlinear functions: Consider the problem of approximating a nonlinear function $f_j(x_j)$, $0 \leq x_j \leq u_j$, as shown in Fig. 3.1. This curve has been approximated by a set of line segments producing a *piecewise linearization*. Once the grid $D_j = \{0 = d_{1j}, d_{2j}, \dots, d_{rj} = u_j\}$ is chosen, the function $g_j(x_j)$ given by the set of line segments can be represented as follows. Let

$$x_j = \sum_{i=1}^r \lambda_i d_{ij} \quad (3.8a)$$

$$\text{and} \quad g_j(x_j) = \sum_{i=1}^r \lambda_i f_{ij} \quad (3.8b)$$

$$\sum_{i=1}^r \lambda_i = 1, \quad \lambda_i \geq 0, \quad i = 1, \dots, r \quad (3.8c)$$

$$\lambda_i \in \text{SOS2} \quad (3.8d)$$

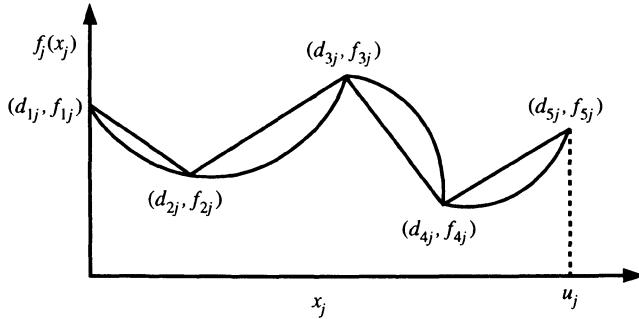


Figure 3.1 Approximating a nonlinear function with line segments

For a given \hat{x}_j , $g_j(\hat{x}_j)$ determined by (3.8b) is a point on the set of line segments if no more than two of the λ_i are positive and adjacent; i.e., the form λ_i, λ_{i+1} . This condition is stated in (3.8d) and can be achieved with the additional constraints

$$\begin{aligned} \lambda_1 &\leqq \delta_1 \\ \lambda_i &\leqq \delta_{i-1} + \delta_i, \quad i = 2, \dots, r-1 \\ \lambda_r &\leqq \delta_{r-1} \\ \sum_{i=1}^{r-1} \delta_i &= 1 \\ \delta_i &= 0 \text{ or } 1, \quad i = 1, \dots, r-1 \end{aligned} \tag{3.9}$$

From (3.9) it follows that for some t , $1 \leqq t \leq r-1$, $\delta_t = 1$ and $\delta_i = 0$, $i \neq t$. Thus $\lambda_t \leqq 1$, $\lambda_{t+1} \leqq 1$, and $\lambda_i = 0$ otherwise.

Fixed-charge problems: In many situations, undertaking an activity means that a fixed charge or setup cost is incurred in addition to be variable costs associated with the level of the activity. This can be represented as (see Fig. 3.2)

$$f_j(x_j) = \begin{cases} d_j + c_j x_j & \text{when } x_j > 0 \\ 0 & \text{when } x_j = 0 \end{cases}$$

When $f_j(x_j)$ is to be minimized and $d_j > 0$, this situation can be modeled by introducing a binary variable; i.e.,

$$\begin{aligned} f_j(x_j) &= c_j x_j + d_j \delta_j \\ x_j(1 - \delta_j) &= 0 \\ x_j &\geqq 0, \delta_j = 0 \text{ or } 1 \end{aligned} \tag{3.10}$$

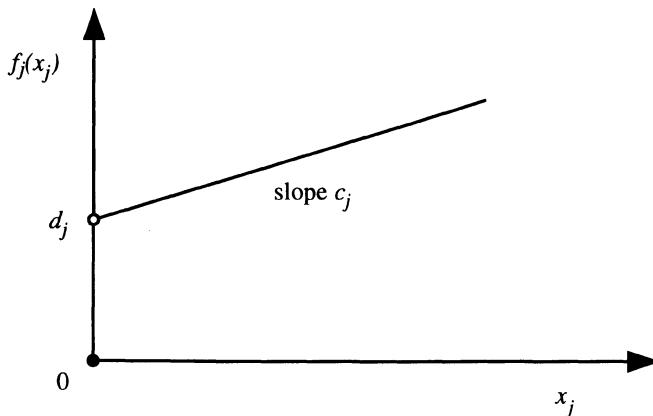


Figure 3.2 Example of fixed charge function

where δ_j is an indicator of whether or not activity j is undertaken. The second constraint guarantees that when $x_j > 0$, $\delta_j = 1$. Minimizing f_j will assure $\delta_j = 0$ when $x_j = 0$.

The nonlinearity in (3.10) can be eliminated if a finite upper bound u_j is known for x_j by replacing it with the linear constraint

$$x_j \leq u_j \delta_j \quad (3.11)$$

If $\delta_j = 1$, (3.11) is trivially satisfied; however, $\delta_j = 0 \Rightarrow x_j = 0$. Again when $x_j = 0$, minimization implies that $\delta_j = 0$. Thus assuming $d_j > 0$ for all j , a fixed-charge problem with linear constraints can be written as the MILP.

$$\begin{aligned} \min \quad & \sum_{j=1}^n (c_j x_j + d_j \delta_j) \\ \text{subject to} \quad & \sum_{j=1}^n a_{ij} x_j = b_i \quad i = 1, \dots, m \\ & x_j - u_j \delta_j \leq 0 \quad j = 1, \dots, n \\ & x_j \geq 0, \delta_j = 0, 1 \quad j = 1, \dots, n \end{aligned} \quad (3.12)$$

A specific application of (3.12) is the single commodity *plant location problem*. In particular, there are n customers with the j th requiring b_j units of the commodity. Also, there are m locations in which plants may operate to satisfy the demands. There is a fixed charge d_j for opening plant i , and the unit cost of supplying customer j from plant i is c_{ij} . The capacity of the plant is u_i . The MILP is

$$\begin{aligned}
\min \quad & \sum_{i=1}^m \left(\sum_{j=1}^n c_{ij} x_{ij} + d_i \delta_i \right) \\
\text{subject to} \quad & \sum_{i=1}^m x_{ij} = b_j \quad j = 1, \dots, n \\
& \sum_{j=1}^n x_{ij} - u_j \delta_j \leq 0 \quad i = 1, \dots, m \\
& x_{ij} \geq 0, \quad \delta_i = 0, 1 \quad \forall i \text{ and } j
\end{aligned}$$

The number of problems of practical interest that can be modeled as integer programs in such fields as engineering, business, and economics abounds. It is important to remember, however, that our ability to model these problems far outpaces our ability to solve them optimally. In the remainder of this chapter we highlight the standard techniques that have been developed for this purpose. Their effectiveness will depend on the problem size and structure. In later chapters, we specialize these techniques to solve linear and nonlinear bilevel programming problems in continuous variables. When integer variables are present we will see that the same techniques can be used by the computational difficulties increase substantially.

3.1.2 Solving Integer Programs

There are two fundamental approaches to solving IPs: *enumeration* and *cutting planes*. When we are dealing with linear models both require, for the most part, the solution of linear programs as subproblems. The LP obtained by dropping the integrality constraints from the ILP (3.1) or the MILP (3.2) will be referred to as the corresponding LP. We also refer to this LP as the *relaxation* of the IP. In general, the problem

$$\min \{f(\mathbf{x}) : \mathbf{x} \in S^1\} \tag{3.13a}$$

is said to be a relaxation of the problem

$$\min \{f(\mathbf{x}) : \mathbf{x} \in S^2\} \tag{3.13b}$$

if $S^1 \supseteq S^2$. Similarly, (3.13b) is said to be a *restriction* of (3.13a). Note that if \mathbf{x}^1 is an optimal solution to (3.13a) and \mathbf{x}^2 an optimal solution to (3.13b), then $f(\mathbf{x}^1) \leq f(\mathbf{x}^2)$. Furthermore, if $\mathbf{x}^1 \in S^2$, then \mathbf{x}^1 is an optimal solution to (3.13b).

From the concept of relaxation, it follows that if the corresponding LP has an optimal solution \mathbf{x}^1 which is an integer vector, then \mathbf{x}^1 is a feasible solution and thus an optimal solution to the IP. As mentioned, there is a class of ILPs for which the corresponding LP always has an integer optimal solution. In this class of problems, the relaxation of the ILP to the corresponding LP does not change the essence of the problem.

In general, the optimum to the corresponding LP is not integer valued. This is illustrated in the example below which is used to sketch the two solution techniques.

Example 3.1.1

$$\begin{aligned} \min f &= 2x_1 - x_2 \\ \text{subject to } &9x_1 - 3x_2 - x_3 = 11 \\ &x_1 + 2x_2 + x_4 = 10 \\ &2x_1 - x_2 + x_5 = 7 \\ &x_1, \dots, x_5 \geq 0 \text{ and integer} \end{aligned}$$

Because x_4 and x_5 are slack variables and x_3 is a surplus variable, the problem can be written as

$$\begin{aligned} \min f &= 2x_1 - x_2 \\ \text{subject to } &9x_1 - 3x_2 \geq 11 \\ &x_1 + 2x_2 \leq 10 \\ &2x_1 - x_2 \leq 7 \\ &x_1, x_2 \geq 0 \text{ and integer} \end{aligned}$$

The feasible region of the corresponding LP is shown in Fig. 3.3. The parallel dashed lines represent isovalue contours of the objective function f and the darkened circles represent feasible points. From the figure it is clear that there are 10 feasible solutions and that the optimum is $x_1 = 2$, $x_2 = 2$, $f = 2$.

Enumeration: Even without examining Fig. 3.3, it is possible to get an upper bound on the number of feasible points. From the nonnegativity conditions coupled with the second constraint we see that $0 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 5$. When $x_2 = 0$, the first constraint tells us that $x_1 \geq 2$ and when $x_2 = 5$, the third constraint implies $x_1 \leq 6$. Thus the constraints $2 \leq x_1 \leq 6$ and $0 \leq x_2 \leq 5$ along with the integrality requirements limit the number of feasible points to not more than 30. For such a small problem, these 30 points could be completely enumerated to find that 10 are feasible, 20 are infeasible, and $\mathbf{x} = (2, 2)$ is optimal.

Using a bit more imagination, we could have added the second and third constraints to give $3x_1 + x_2 \leq 17$. When $x_2 = 0$, $3x_1 \leq 17$ and integrality imply that $x_1 \leq 5$. This reduces the upper bound on the number of feasible points from 30 to 24. Also, multiplying the first constraint by -1 and the second by 4 and adding gives $9x_2 \leq 35$ or $x_2 \leq 3$. This further reduces the number of feasible points to no more than 16.

Next we can pick a feasible solution such as $x_1 = 2$, $x_2 = 0$ and evaluate the objective function, giving $f(2, 0) = 4$. Thus every optimal solution to Example 3.1.1 must satisfy $2x_1 - x_2 \leq 4$. Now $x_2 \leq 3$ implies $2x_1 \leq 7$ or $x_1 \leq 3$. This means that the candidates for optimality have been reduced to the set

$$Q = \{(x_1, x_2) : x_1 = 2, 3 \text{ and } x_2 = 0, 1, 2, 3\}$$

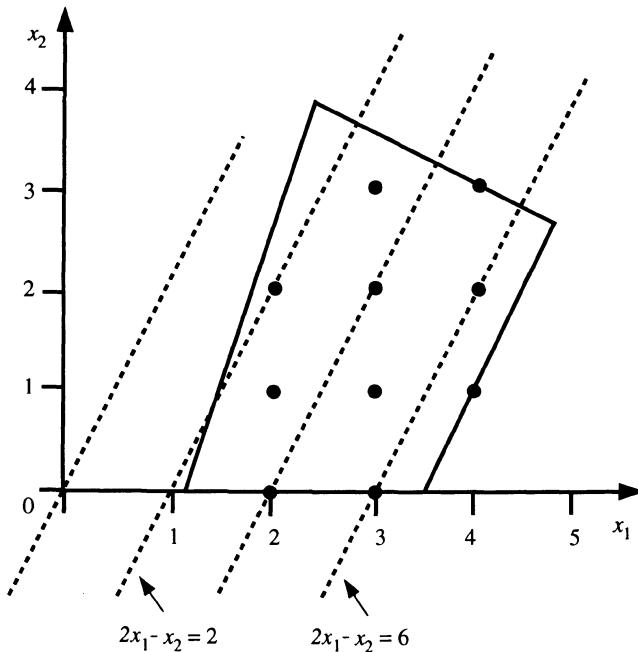


Figure 3.3 Feasible region of isovalue contours for Example 3.1.1

Of these 8 points, (3,1) and (3,0) yield objective values larger than 4. The optimal value of the corresponding LP is $f_{LP} = 1\frac{4}{21}$ at $\boldsymbol{x}_{LP} = \left(2\frac{10}{21}, 3\frac{16}{21}\right)$. Since the objective value at optimality for Example 3.1.1 must be integral, it follows that $2x_1 - x_2 \geq 2$. This constraint is not satisfied at (2,3) so the candidates for optimality have been reduced to the set

$$Q = \{(2,0), (2,1), (2,2), (3,2), (3,3)\}$$

The intent of this simple analysis has been to show how enumeration arguments can be systematically applied to reduce the size of the feasible region, perhaps to the point where it is possible to identify the optimal solution by inspection. In the analysis, we made use of implied bounds on variables, rounding, linear combinations of constraints, and bounds on the objective function value. In the next section, we elaborate these ideas.

Cutting planes: The second class of approaches to solving ILPs is based on the generation of a sequence of linear inequalities that “cut off” part of the feasible region of the corresponding LP while leaving all feasible integer points in tact. When enough of these cutting hyperplane have been generated, the ILP will have the same optimal

solution as the corresponding LP. In Fig. 3.4, two cutting (hyper)planes are shown that accomplish this for Example 3.1.1. When the first ($x_1 - x_2 \geq 0$) is added and the LP is re-solved, the new solution occurs at the intersection of cutting plane 1 and constraint $4x_1 - x_2 = 5$; i.e., at $\mathbf{x} = \left(\frac{5}{3}, \frac{5}{3}\right)$. After the second cutting plane ($x_1 \geq 2$) is added, the ILP and the LP solution coincide at $\mathbf{x} = (2, 2)$.

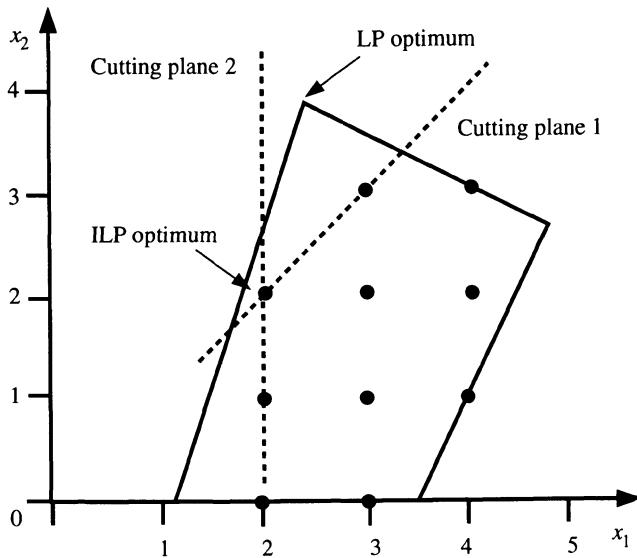


Figure 3.4 Use of cutting planes to reduce feasible region of ILP

To formalize the use of cutting planes, suppose that the set $S = \{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ integer} of feasible solutions to an ILP is bounded. Let n^+ be the number of points in S and define its *convex hull* to be

$$S^+ = \left\{ \mathbf{y} : \mathbf{y} = \sum_{k=1}^{n^+} \alpha_k \mathbf{x}^k, \alpha_k \geq 0, \sum_{k=1}^{n^+} \alpha_k = 1, \mathbf{x}^k \in S \right\}$$

Note that

$$S \subseteq S^+ \subseteq T = \{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$$

where T is the set of feasible solutions to the corresponding LP. For Example 3.1.1, in order to construct S^+ it would be necessary to introduce three additional cutting planes: $x_2 \leq 3$, $x_1 \leq 4$, and $x_1 - x_2 \leq 3$. In general, it is very difficult to identify all the cutting planes (actually facets) needed to construct the convex hull of integer solutions to an ILP. For special cases such as the traveling salesman problem and the knapsack problem, research over the last 20 years has produced a significant number of these facets (see [N3]). However, as this example demonstrates, it is rarely

necessary to construct the entire convex hull in order for the ILP optimum and the corresponding LP optimum to coincide.

3.2 ENUMERATIVE METHODS

The presence in an IP of a finite number of discrete decision variables with finite bounds suggests that it might be possible to enumerate all combinations of values and pick the solution that provides the smallest objective function value. Because the number of combinations grows exponentially with the number of variables, this is only practical when the problem contains a handful of variables defined over a limited range. The idea, though, is at the center of what are known as *branch and bound* (B&B) procedures which attempt to perform the enumeration in an intelligent way so that not all combinations have to be examined. Depending on the technique being used, the particular procedures are sometimes called implicit enumeration, progressive separation and evaluation, tree search, and strategic partitioning. Regardless of the name, B&B has two appealing qualities. First, it can be applied to the mixed-integer problem and to the pure integer problem in essentially the same way, so a single method works for both. Second, B&B typically yields a succession of feasible integer solutions (as opposed to traditional cutting plane approaches in which feasibility is not obtained until the problem is solved), so that if the computations are terminated due to time or memory restrictions, the current best solution becomes a candidate for the optimum.

Branch and bound methods can often be tailored to exploit special problem structures, thereby allowing these structures to be handled with greater efficiency and reduced computer memory. In fact, except in very general terms, there is no one method but a whole collection of methods that share a number of common characteristics. The tailoring aspect further permits B&B to be applied directly to many kinds of combinatorial problems without first going through an intermediate stage of introducing specific integer variables and linear constraints to yield an IP formulation. The direct application of methods to problems is invaluable when the use of an intervening IP formulation may drastically increase problem size or otherwise obscure exploitable problem characteristics.

For pure and mixed-integer linear programming many of the most effective B&B procedures are based on the use of the simplex method. These will be examined first. For the moment, let us assume that we are dealing with a 0-1 ILP, although the ideas are the same for the more general case. What changes is essentially the bookkeeping and the notation.

Simplex-based methods begin by solving the original problem as a linear program. The appropriate LP is obtained by replacing the 0-1 constraints $x_j \in \{0, 1\}$, $j =$

$1, \dots, n$, with the relaxed constraints, $0 \leq x_j \leq 1$, for all j . If an integer solution is not realized, one of the integer variables that is fractional in the LP solution, denoted by (say) x_k , where k is the variable index, is selected and two descendants of the original problem are created; one in which $x_k = 0$ and the other in which $x_k = 1$. This operation is called *branching*. Because x_k is a 0-1 variable, and the descendant problems are exactly the same as the parent except for the assignment of a specific value to x_k , the solution to one of the two descendants must in fact be the solution to the original problem. Thus the latter has been replaced with two IPs that now must be solved. The expectation is that these modified IPs will be easier to solve than their parent because each is more restricted than the parent, having fewer variables that must be assigned integer values.

The process is then repeated, selecting one of the problems that remains to be solved as the current IP and treating it exactly the same as the original. This in turn may create two new problems to replace the one currently under consideration unless, for example, the current problem is infeasible. Repeated iterations eventually produce an integer solution (if one exists) for one of the current IPs which becomes a candidate for the optimal solution to the original problem. Keeping track of the best of these candidates and its attendant objective function value provides an additional way to weed out descendants of the original IP (perhaps several generations removed) that would be unprofitable to explore in the sense that they could not possibly yield the overall optimum. The branching operations are often represented in a rooted binary search tree where each vertex has one predecessor and at most two successors.

To formalize these concepts, let f^* denote the objective function value of the incumbent and f^o the objective function value of the corresponding LP relaxation. Then, whenever a current IP minimization problem is solved as an LP one of four alternatives arises:

1. The LP has no feasible solution (in which case the current IP also has no feasible solution).
2. The LP has an optimal solution $f^o \geq f^*$ (in which case the current IP optimum $f' \geq f^o \geq f^*$ and therefore cannot provide an improvement over the incumbent).
3. The optimal solution to the LP is integer and feasible, and yields $f^o < f^*$ (in which case the solution is optimal for the current IP and provides an improved incumbent for the original IP; f^* is therefore reset to f^o).
4. None of the foregoing occurs; i.e., the optimal LP solution exists, satisfies $f^o < f^*$, but is not integer and hence not feasible to the current IP.

In each of the first three cases, the current IP is disposed of simply by solving the LP. Such an IP is said to be fathomed. A problem that is fathomed as a result of (3) yields particularly useful information because it allows us to update the incumbent. If the

problem is not fathomed and hence winds up in (4), further exploration or branching is required.

3.2.1 Definitions and Concepts

To formalize the ideas of relaxation and enumeration, let S be the feasible region of the IP given in (3.1) and assume that $S \neq \emptyset$ and the optimal solution $f_{IP} \triangleq f(\mathbf{x}^*) > -\infty$. The set $\{S^i : i = 1, \dots, k\}$ is called a *division* of S if $\bigcup_{i=1}^k S^i = S$. A division is called a *partition* if $S^i \cap S^j = \emptyset$ for $i, j = 1, \dots, k$, $i \neq j$. The following results are taken from [N3].

Proposition 3.2.1 Let

$$f_{IP}^i = \min \{c\mathbf{x} : \mathbf{x} \in S^i\} \quad (IP^i)$$

where $\{S^i\}_{i=1}^k$ is a division of S . Then $f_{IP} = \min_{i=1, \dots, k} \{f_{IP}^i\}$.

This proposition expresses the concept of divide and conquer where instead of optimizing over S , we try to solve the problem by optimizing over smaller sets and then integrating the solutions. The division is frequently done recursively as depicted in Fig. 3.5. Here the children of a given node (e.g., $\{S^{21}, S^{22}, S^{23}\}$ are the children of S^2) represent a division of the feasible region of their parent. When $S \subseteq B^n$ (where B^n is set of all binary n -tuples), a simple way of doing the recursive division is shown in Fig 3.6. In this search tree $S^{\delta_1 \dots \delta_k} = S \cap \{\mathbf{x} \in B^n : x_j = \delta_j \in \{0, 1\} \text{ for } j = 1, \dots, k\}$ and the division is a partition of S . On the left-hand side of the tree, the variables are enumerated in the order (x_1, x_3, x_2) ; on the right-hand side the order is (x_1, x_2, x_3) . For a given problem, the particular branching rules and intermediate results determine how the tree will actually be constructed.

Carried to the extreme, division reduces to total enumeration of the elements of S , and so is not a workable approach for problems with more than a handful of integer variables. For any expectations of success, enumerative methods need to avoid dividing the initial set into too many subsets. Now, suppose S has been divided into subsets $\{S^1, \dots, S^k\}$. If we can establish that no further division of S^i is necessary, we say that the search tree can be *pruned* or *fathomed* at the node corresponding to S^i .

Proposition 3.2.2 The search tree can be fathomed at the node corresponding to S^i if any one of the following three conditions holds.

1. Infeasibility: $S^i = \emptyset$.
2. Integrality: An optimal integer solution to IP^i has been obtained.
3. Value dominance: $f_{IP}^i \geq f_{IP}$.

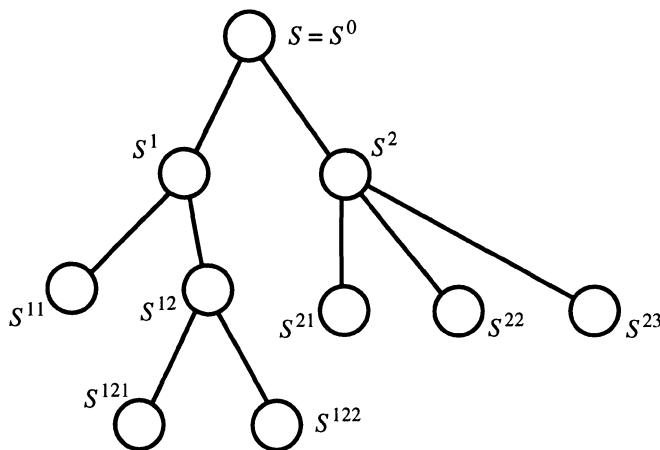


Figure 3.5 Example of search tree used during enumeration

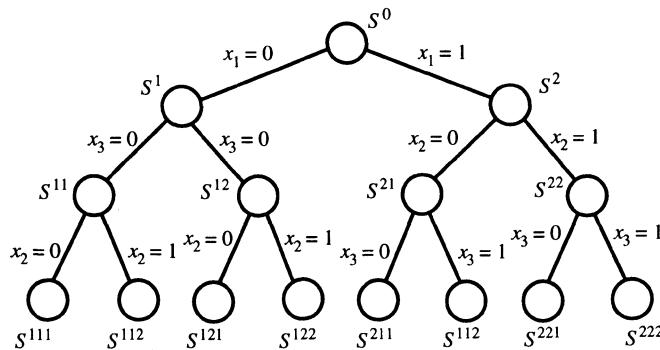


Figure 3.6 Typical binary search tree

Because IP^i may be almost as difficult as the original problem, we would like to be able to apply Proposition 3.2.2 without necessarily having to solve IP^i . To accomplish this, we use relaxation or duality. Let RP^i be a relaxation of IP^i with $S^i \subseteq S_R^i$ and $f_R^i(\mathbf{x}) \leq c\mathbf{x}$ for $\mathbf{x} \in S^i$.

Proposition 3.2.3 The search tree can be fathomed at the node corresponding to S^i if any one of the following three conditions holds.

1. RP^i is infeasible.
2. An optimal solution \mathbf{x}_R^i to RP^i satisfies $\mathbf{x}_R^i \in S^i$ and $f_R^i = c\mathbf{x}_R^i$.
3. $f_R^i \geq \bar{f}_{IP}$, where \bar{f}_{IP} is the value of some feasible solution to IP.

Proof: Condition 1 implies $S^i = \emptyset$. Condition 2 implies that \mathbf{x}_R^i is an optimal solution to IP^i . Condition 3 implies $f_{IP}^i \geq f_{IP}$. ■

Let DP^i be the dual to the LP obtained from IP^i by relaxing the integrality requirements.

Proposition 3.2.4 The search tree can be fathomed at the node corresponding to S^i if any one of the following two conditions holds.

1. The objective value of DP^i is unbounded from above.
2. DP^i has a feasible solution of value equal to or greater than \bar{f}_{IP} .

Proof: Condition 1 implies $S^i = \emptyset$. Condition 2 implies $f_{IP}^i \geq f_{IP}$. ■

Comparing Propositions 3.2.3 and 3.2.4, we see that RP^i must be solved to optimality before value dominance can be applied, but value dominance may be applicable with respect to dual feasible solutions that are not optimal. In contrast, RP may yield a feasible solution to IP that establishes or improves the upper bound \bar{f}_{IP} .

3.2.2 Generic Branch and Bound Algorithm

Enumerative relaxation algorithms are frequently called branch and bound or implicit enumeration and are represented by a search tree. In this context, a path in the tree from the root node to node j is denoted by P_j . Each edge in P_j imposes a constraint, and each node represents the original problem constraints plus the additional constraints given by the edges. A node that is not fathomed and whose corresponding constraint set has not been divided is said to be *live*. Branching means choosing a live node to consider next for fathoming or division. There are many possible rules for branching. A common procedure is to branch to one of the successor nodes of the current node under consideration. If the current node j is fathomed, one simply *backtracks* along P_j until a node having at least one live successor is encountered. One of these successor nodes is chosen from branching. If there are no live nodes remaining the enumeration is complete.

We now give an algorithm for solving an IP. In the description, \mathcal{L} is a collection of integer programs $\{IP\}$, each of which is of form $f_{IP}^i = \min \{\mathbf{c}\mathbf{x} : \mathbf{x} \in S^i\}$ where $S^i \subseteq S$. Associated with each problem in \mathcal{L} is a lower bound $\underline{f}^i \leq f_{IP}^i$.

Algorithm

Step 1 (Initialization) $\mathcal{L} = \{IP\}$, $S^0 = S$, $\underline{f}^0 = -\infty$, and $\bar{f}_{IP} = \infty$.

Step 2 (Termination test) If $\mathcal{L} = \emptyset$, then the solution \mathbf{x}^0 that yielded $\bar{f}_{IP} = \mathbf{c}\mathbf{x}^0$ is optimal.

Step 3 (Branching) Select a problem IP^i from \mathcal{L} and put $\mathcal{L} \leftarrow \mathcal{L} \setminus IP^i$. Solve its relaxation RP^i . Let f_R^i be the optimal value of the relaxation and let \mathbf{x}_R^i be an optimal solution if one exists.

Step 4 (Fathoming)

- a. If $f_R^i \geq \bar{f}_{IP}$, go to Step 2. (Note if the relaxation is solved by a dual algorithm, then the step is applicable as soon as the dual value reaches or exceeds \bar{f}_{IP} .)
- b. If $\mathbf{x}_R^i \notin S^i$, go to Step 5.
- c. If $\mathbf{x}_R^i \in S^i$ and $c\mathbf{x}_R^i < \bar{f}_{IP}$, put $\bar{f}_{IP} \leftarrow c\mathbf{x}_R^i$ and delete from \mathcal{L} all problems with $\underline{f}^i \geq \bar{f}_{IP}$. Go to Step 2.

Step 5 (Division) Let $\{S^{ij}\}_{j=1}^k$ be a division of S^i . Add problems $\{IP^{ij}\}_{j=1}^k$ to \mathcal{L} , where $\underline{f}^{ij} = f_R^i$ for $j = 1, \dots, k$. Go to Step 2.

Virtually all commercial codes for solving mixed-integer linear programs of the type given in (3.2) use LP relaxation and partition (as opposed to the more general division). We now focus attention on this specific case, but restrict the presentation to the ILP. The approach would be exactly the same for the MILP.

3.2.3 Branch and Bound Using LP Relaxation

Recall that the feasible region for the ILP is given by $S = \{\mathbf{x} : \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq 0 \text{ and integer}\}$. The corresponding continuous relaxation will be denoted by $T = \{\mathbf{x} : \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq 0\}$, where $S \subseteq T$. To simplify the notation a bit, let node i in the search tree be denoted by v_i with the root node given by v_0 . For a path P_i in the tree connecting v_0 to v_i , let S^i be the intersection of S with the set of points satisfying the constraints given by the edges of P_i . If P_i has $k + 1$ nodes order as

$$v_0 = v_{i(0)}, v_{i(1)}, \dots, v_{i(k-1)}, v_{i(k)} = v_i$$

then

$$S = S^{i(0)} \supseteq S^{i(1)} \supseteq \dots \supseteq S^{i(k)} = S^i$$

Now suppose that the enumeration is at node i in the tree. The problem considered at v_i is

$$f_{IP}^i = \min \{f(\mathbf{x}) : \mathbf{x} \in S^i\} \quad (3.14)$$

A lower bound $\underline{f}^i \leq f_{IP}^i$ may be calculated by considering the relaxation of (3.14)

$$\underline{f}^i = f_{LP}^i = \min \{f(\mathbf{x}) : \mathbf{x} \in T^i \supseteq S^i\} \quad (3.15)$$

which we call LP^i . Note that a lower bound at a node is valid for any of its successors; i.e., if v_k is a successor of v_i , then $T^i \supseteq S^i \supseteq S^k$.

For the ILP we have

$$S^i = \{\mathbf{x} : \mathbf{A}^i \mathbf{x} = \mathbf{b}^i, \mathbf{x} \geq \mathbf{0} \text{ and integer}\} \quad (3.16a)$$

$$T^i = \{\mathbf{x} : \mathbf{A}^i \mathbf{x} = \mathbf{b}^i, \mathbf{x} \geq \mathbf{0}\} \quad (3.16b)$$

so \underline{f}^i is calculated by solving the corresponding LP.

Partition

Suppose that the LP (3.15) is solved at v_i and the solution \mathbf{x}^i is not all-integer. In particular, some basic variable $x_{B_j} = \lfloor \bar{b}_j \rfloor + \phi_j$, $0 < \phi_j < 1$. Then a partition of S^i is

$$\left\{ S^i \cap \{\mathbf{x} : x_{B_j} \leq \lfloor \bar{b}_j \rfloor\}, S^i \cap \{\mathbf{x} : x_{B_j} \geq \lceil \bar{b}_j \rceil\} \right\} \quad (3.17)$$

where $\lceil \gamma \rceil$ denotes the smallest integer greater than or equal to γ , and $\lfloor \gamma \rfloor$ denotes the largest integer less than or equal to γ .

Finiteness

Assume that an upper bound u_j is known for each variable x_j , $j = 1, \dots, n$. (If this is not the case, a bounding constraint such as $\sum_j x_j \leq M$, where M is a large positive number, can be added to the constraint set.) Let

$$S^i = \{\mathbf{x} : \mathbf{A} \mathbf{x} = \mathbf{b}, 0 \leq \alpha_j^i \leq x_j \leq \beta_j^i \leq u_j, x_j \text{ integer}, j = 1, \dots, n\} \quad (3.18)$$

at node v_i , where α_j^i and β_j^i are integers determined from (3.17), and $\alpha_j^0 = 0$ and $\beta_j^0 = u_j$. Now define $N_j(i)$ as the number of integers in the interval $[\alpha_j^i, \beta_j^i]$ and h_i as the maximum number of edges that can emanate from node i ; that is,

$$N_j(i) = \beta_j^i - \alpha_j^i + 1$$

$$h_i = \prod_{j=1}^n N_j(i)$$

If v_k is a successor of v_i , then from (3.17) it follows that for some t

$$N_t(k) < N_t(i)$$

and

$$N_j(k) = N_j(i), j \neq t$$

Thus $h_k < h_i$ so as more edges are added to the search tree, the number of remaining potential edges strictly decreases. It follows that the algorithm is finite since no path can contain more than $h_0 = \prod_{j=1}^n (1 + u_j)$ edges. Further note that

$$h_i = 1 \Rightarrow \alpha_j^i = \beta_j^i, j = 1, \dots, n$$

so if $(\alpha_1^i, \dots, \alpha_n^i) \in S^i$, then $S^i = T^i = \{(\alpha_1^i, \dots, \alpha_n^i)\}$. If $(\alpha_1^i, \dots, \alpha_n^i) \notin S^i$, then $S^i = T^i = \emptyset$. The case $h_i = 0$ never occurs because any node i having $h_i = 1$ is fathomed and if $h_i > 1$, then the two successors, v_k and v_t , have $1 \leq h_k < h_i$ and $1 \leq h_t < h_i$.

The LPs resulting from the partition (3.17) are especially manageable since the added constraints are simply lower and upper bounds on the individual variables. They are typically solved by the dual simplex algorithm for bounded variables (e.g., see [M18], Chapter 11). The size of the search tree is highly dependent on the quality of the bounds produced by these LPs. In particular, we have the following result

Proposition 3.2.5 If node i of the search tree with constraints S^i is such that $\min\{\mathbf{c}\mathbf{x} : \mathbf{x} \in T^i\} \geq f_{IP}$ then node i can be fathomed.

Proposition 3.2.5 indicates that, regardless of how we develop the tree, the bounds (quality of the relaxations) are the primary factor in the efficiency of the B&B algorithm. Nevertheless, tree development strategies related to selecting the next subproblem corresponding to a live node and which fractional variable should be chosen for partitioning are also important. These issues are discussed by numerous authors (e.g., see [S1]).

Algorithm for ILP

- Step 1 (Initialization) Begin at node $i = 0$, where S^0 is given in (3.18), $\underline{f}^0 = -\infty$, and $\bar{f}^0 = \infty$. Go to Step 2.
- Step 2 (Branching) If no live nodes exist, go to Step 7; otherwise select a live node i . If the LP (3.15) has been solved, go to Step 3; otherwise go to Step 4.
- Step 3 (Partitioning) Choose a variable x_{B_j} with $\phi_j > 0$ and partition S^i as in (3.17). Go to Step 2.
- Step 4 (Solving LP) Solve the LP relaxation (3.15). If the LP is not feasible, fathom v_i and go to Step 2. If the LP has an optimal solution \mathbf{x}^i , let $\underline{f}^i = \lceil f_{LP}^i \rceil$ and go to Step 5. (Note that \underline{f}^i can be rounded up because every integer solution yields an integer objective function value, assuming as we are that the cost coefficients are themselves integers.)
- Step 5 (Fathoming by integrality) If \mathbf{x}^i is not all-integer, go to Step 6; otherwise, let $\bar{f}^i = f^i$ and fathom v_i . Let $\bar{f}^0 = \min\{\bar{f}^0, \bar{f}^i\}$ and go to Step 6.
- Step 6 (Fathoming by bounds) Fathom any node i such that $\underline{f}^i \geq \bar{f}^0$ and go to Step 2.
- Step 7 (Termination) Terminate. If $\bar{f}^0 = \infty$, there is no feasible solution. If $\bar{f}^0 < \infty$, the feasible solution that yielded \bar{f}^0 is optimal.

The use of the algorithm is illustrated in the example below. To perform the computations, a branching strategy is needed at Step 2 and a variable selection strategy is needed at Step 3. With regard to the former, one such strategy is to select the live node with the smallest LP objective function value. The aim is to quickly find a good feasible solution. However, this procedure requires storing at least the objective function values, bases, and variable bounds associated with all live nodes, a requirement that often becomes excessive. A second rule, which we will adopt here, is to select the most recently created node associated with the \leq bound. This is known as a *depth-first* search. At Step 3, the basic variable with the largest fractional component ϕ_j will be chosen for partitioning.

Example 3.2.1

$$\begin{aligned} \min f(\mathbf{x}) = & \quad 7x_2 + 3x_3 + 4x_4 \\ \text{subject to} \quad & -x_1 + x_2 + 2x_3 + 3x_4 = 8 \\ & -x_1 + 4x_2 + 3x_3 + 4x_4 - x_5 = 13 \\ & x_1, \dots, x_5 \geq 0 \text{ integer} \end{aligned}$$

To begin, the only live node is v_0 so we go to Step 4 and solve the corresponding LP. The optimal solution is $f_{LP}^0 = 14.2$ so we can set $\underline{f}^0 = 15$. The decision vector $\mathbf{x}^0 = (0, 2/5, 19/5, 0, 0)$ which is not all-integer. Following the logic, we arrive at Step 3 and choose x_3 for partitioning. Two new nodes are created as a result, the first v_1 is associated with $x_3 \leq 3$ and the second v_2 with $x_3 \geq 4$. Solving the augmented LP at v_1 gives $f_{LP}^1 = 14.2$ and $\mathbf{x}^1 = (1/2, 0, 3, 1/2, 0)$ which does not provide an improvement in the objective function lower bound because $\underline{f}^1 = [f_{LP}^1] = 15$. We now arbitrarily choose x_2 for partitioning and create v_3 and v_4 defined by the constraints $x_2 \leq 0$ and $x_2 \geq 1$, respectively. Solving the LP at v_3 gives an all-integer solution $\mathbf{x}^3 = (4, 0, 1, 2, 0)$ with $f_{LP}^3 = 17$ so we can set $\bar{f}^0 = 17$ at Step 5 and fathom the node.

Nodes v_2 and v_4 remain live. We branch to v_4 , the most recently created of the two. Solving the LP gives $f_{LP}^4 = 16\frac{1}{3}$ with $\mathbf{x}^4 = (0, 7/3, 1, 0, 1/3)$ which is fractional. However, $\underline{f}^4 = [f_{LP}^4] = 17$ so at Step 6 we see that $\underline{f}^4 = \bar{f}^0$ indicating that v_4 can be fathomed. The only live node is v_2 . Adding $x_3 \geq 4$ to the original formulation and solving the LP gives $f_{LP}^2 = 14\frac{1}{3}$ with $\mathbf{x}^2 = (1/3, 1/3, 0, 4, 0)$ and $\underline{f}^2 = [f_{LP}^2] = 15$. Fathoming is not possible so we go to Step 2 and then to Step 3 and arbitrarily choose x_2 for partitioning. The two new nodes, v_5 and v_6 , are defined by the constraints $x_2 \leq 0$ and $x_2 \geq 1$, respectively. Branching on the \leq constraint and solving the LP gives an all-integer solution $\mathbf{x}^5 = (2, 0, 5, 0, 0)$ with $f_{LP}^5 = \bar{f}^5 = 15$. At Step 5, the upper bound is updated to give $\bar{f}^0 = \min \{17, 15\} = 15$ and v_5 is fathomed. We now have $\bar{f}^0 = f^2$ implying that v_6 can be fathomed as well because a child cannot have a smaller objective function value than its parent. The optimal solution is $f^* = 15$ and $\mathbf{x}^* = (2, 0, 5, 0, 0)$. The full search tree is shown in Fig. 3.7.

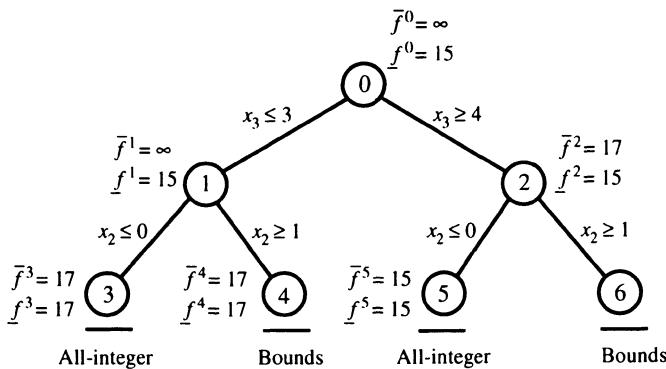


Figure 3.7 Search tree for Example 3.2.1

3.2.4 Implementation Issues

The basic algorithms presented in the previous subsections can be modified easily to accommodate special model structures as well as unique problem characteristics. As discussed in Part II of the book, branching rules and node selection can play a significant role in determining the efficiency of an algorithm. Although each type of problem may require a unique search tree strategy, several general rules of both a theoretical and empirical nature, have been shown to lead to much quicker run times. The most effective are related to penalty computations, node selection, and partitioning. Each is discussed presently.

Node Selection

Information must be stored at each node in the search tree. In part, this involves keeping track of the basis inverse and the relevant upper and lower bounds. One can think of having a *list* of nodes, with newly generated information being put on top of the list. Branching involves retrieving the necessary information about a node from somewhere on the list. The question then is, given a list L of active subproblems or equivalently, a partial tree of unfathomed or live nodes, which node should be examined next? The available options can be divided into two categories: (1) *a priori* rules that determine in advance the order in which the tree will be developed; and (2) adaptive rules that select a node using information such as bounds or function values associated with the live nodes.

The most commonly used *a priori* type rule is *depth-first search plus backtracking*, which is also known as last-in, first-out (LIFO). In this approach, if the current node is not fathomed, the next node considered is one of its two children. Backtracking means that when a node is fathomed we go back on the path from this node toward the root until we encounter the first unfathomed node which has a child that has not yet been considered. If we indicate a procedure for choosing the branching variables

and specify, for example, that the left child is considered first, then depth-first search plus backtracking becomes a complete *a priori* rule.

As Nemhauser and Wolsey point out, depth-first search has two principle advantages.

1. The LP relaxation for a child is obtained from the LP relaxation of its parent by the addition of a simple lower- or upper-bound constraint. Hence given the optimal solution of the parent node, we can efficiently reoptimize with the dual simplex algorithm without a basis reinversion or a transfer of data between storage and active memory.
2. As we go further down the tree, we are more likely to find feasible solutions. The success of a B&B scheme depends on having a good upper bound \bar{f}_{IP} for value dominance fathoming.

A second (essentially) *a priori* rule is known as *breadth-first search* which is just the opposite of depth-first search. The level of a node in an enumeration tree is the number of edges in the unique path between it and the root. In breadth-first search, all nodes at a given level are considered before any nodes at the next level. If tight upper and lower bounds are available early on, this approach can be quite effective because it may significantly reduce the size of the overall tree. The number of new nodes potentially doubles with each successive level. The default option in most commercial codes is depth first. Limited breadth first is usually available but storage requirements can quickly eat up memory if this approach is not controlled.

When a node is fathomed, a backtracking strategy is employed to determine the next node to examine. In most large-scale problems, this is done adaptively but several popular rules have proven to be effective:

- a. Choose a node that has to be considered in any case. By Proposition 3.2.2, if there is a unique node with the smallest lower bound it must be examined at some point. This leads to the *best lower bound* rule; i.e., when a node has been fathomed, select next from all the live nodes, that which has the smallest lower bound. Thus if \mathcal{L} is the set of live nodes, select an $i \in \mathcal{L}$ that minimizes \underline{f}^i .
- b. Choose a node that has a high likelihood of containing an optimal solution. The motivation for this rule is that once we have found an optimal solution, even if we are unable to prove immediately that it is optimal, we will have obtained the smallest possible value of \bar{f}_{IP} . This is extremely important for subsequent fathoming. Moreover, if the computations are halted before convergence, the incumbent is the optimum even if it has not been verified. Suppose $\hat{f}^i \geq \underline{f}^i$ is an estimate of f_{IP}^i . The *best estimate* rule is to choose an $i \in \mathcal{L}$ that minimizes \hat{f}^i . Below we discuss a procedure for finding \hat{f}^i .
- c. Although trying to find an optimal solution is highly desirable, it may be more practical to try to find a feasible solution \hat{x} quickly such that $c\hat{x} < \bar{f}_{IP}$. One

way to do this is to use the *quick improvement* criterion

$$\max_{i \in \mathcal{L}} \frac{\bar{f}_{IP} - f^i}{\bar{f}_{IP} - \hat{f}^i}$$

Note that node i with $\hat{f}^i < \bar{f}_{IP}$ will be preferred to node j with $\hat{f}^j \geq \bar{f}_{IP}$. Moreover, preference will be given to nodes for which $\bar{f}_{IP} - \hat{f}^i$ is small. The expectation is that such nodes will yield a feasible solution quickly. A number of commercial codes use this criterion as the default once a feasible solution is obtained.

Penalties

Empirical evidence shows that the choice of a branching variable can have a significant impact on the running time of an algorithm. Frequently, there are a few variables that need to be fixed at integer values and then the rest turn out to be integer-valued in the LP solution. Because robust methods for identifying such variables have not been established, a common way to choosing a branching variable is by user-specified priorities. In particular, *degradations* or *penalties* attempt to estimate the increase in f^i that is caused by requiring x_j to be integral. Consider any node such that the LP relaxation (3.15) has a noninteger solution given by

$$x_{B_i} = \bar{b}_i - \sum_{j \in Q} \alpha_{ij} x_j, \quad i = 1, \dots, m$$

and $x_j = 0$ for all $j \in Q$, where Q is the set of nonbasic variables. For simplicity, assume that the LP has been solved by the ordinary dual (as opposed to the upper bounded dual) simplex algorithm. Now, for i such that $\phi_i > 0$, if the constraint $x_{B_i} \leq \lfloor \bar{b}_i \rfloor$ is imposed, it is introduced in the tableau as

$$\begin{aligned} s_i &= \lfloor \bar{b}_i \rfloor - x_{B_i} = \lfloor \bar{b}_i \rfloor - \bar{b}_i + \sum_{j \in Q} \alpha_{ij} x_j \\ &= -\phi_i + \sum_{j \in Q} \alpha_{ij} x_j \end{aligned}$$

The pivot column q will be determined by

$$\frac{\bar{c}_q}{\alpha_{iq}} = \min_{j \in Q} \left\{ \frac{\bar{c}_j}{\alpha_{ij}} : \alpha_{ij} > 0 \right\}$$

to assure that all the reduced costs $\bar{c}_j - \bar{c}_q \alpha_{ij} / \alpha_{iq} \geq 0$ for all j so the next basis remains dual feasible. The corresponding increase in the objective function f^0 after the first dual simplex iteration is $\phi_i(\bar{c}_q / \alpha_{iq})$. Thus we introduce the *down penalty*

$$D_i = \min_{j \in Q} \left\{ \phi_i \frac{\bar{c}_j}{\alpha_{ij}} : \alpha_{ij} > 0 \right\}$$

for partitioning on row i . It can be viewed as a lower bound on the increase in f^0 since further simplex iterations may be required to restore primal feasibility. If $\alpha_{ij} \leq 0$ for

all $j \in Q$, let $D_i = \infty$; in this case x_{B_i} is called a *monotone increasing* variable and is discussed later in this section.

An *up penalty* is derived similarly. Note that

$$x_{B_i} \geq \lceil \bar{b}_i \rceil \Leftrightarrow s_i = -1 + \phi_i - \sum_{j \in Q} \alpha_{ij} x_j$$

so

$$U_i = \min_{j \in Q} \left\{ (\phi_i - 1) \frac{\bar{c}_j}{\alpha_{ij}} : \alpha_{ij} < 0 \right\}$$

If $\alpha_{ij} \geq 0$ for all $j \in Q$, let $U_i = \infty$; in this case x_{B_i} is called a *monotone decreasing* variable. Combining these two results we see that one or the other penalty must be incurred when the partitioning is based on x_{B_i} , so a lower bound on the increase in f^0 is

$$P_i = \min \{U_i, D_i\}$$

The penalties U_i and D_i were derived without taking into account the integrality requirement on the nonbasic variables. That is, in order to have an integer solution, some nonbasic variable must become positive, and therefore greater than or equal to 1. If x_q is such a variable, the corresponding increase in f^0 is at least \bar{c}_q so that

$$P = \min_{j \in Q} \{\bar{c}_j\}$$

is a valid bound which does not depend on the partitioning row. This observation has greater implications for fathoming certain variables. Let \bar{f}^i and \underline{f}^i be upper and lower bounds, respectively, at node i . If $\bar{c}_q \geq \bar{f}^i - \underline{f}^i$ then introducing x_q into the basis cannot lead to an improvement in the objective function value beyond \bar{f}^i . This means that x_q can be eliminated from LP^i as well as its descendants. Note that if $\bar{c}_q = \bar{f}^i - \underline{f}^i$ and x_q is eliminated, we may be eliminating alternative optima. This could have ramifications in problems where not all optimal solutions are qualitatively the same.

Another bound is available from the constraint associated with the method of integer forms which is discussed in Section 3.3.1:

$$s_i = -\phi_i + \sum_{j \in Q} \phi_{ij} x_j$$

where ϕ_{ij} is the fractional part of α_{ij} . If this constraint were added to the simplex tableau, the first dual pivot would yield an increase in f^0 of

$$\bar{P}_i = \min_{j \in Q} \left\{ \frac{\phi_i \bar{c}_j}{\phi_{ij}} : \phi_{ij} > 0 \right\}$$

Collectively then, a lower bound on the increase in f^0 is

$$P^* = \max \left(P, \max_{i: \phi_i > 0} \max \{P_i, \bar{P}_i\} \right)$$

This allows us to compute a stronger lower bound at Step 4 of the algorithm for the ILP. If P^* is obtained at node k , $\underline{f}^k = [f_{LP}^k + P^*]$. Alternatively, we can compute an estimate \hat{f}^k by assuming that the degradations for each variable are independent.

$$\hat{f}^k = f_{LP}^k + \sum_{j \in Q} \max \left(\bar{c}_j, \min_{i: \alpha_{ij} > 0} \left\{ \phi_i \frac{\bar{c}_j}{\alpha_{ij}} \right\}, \min_{i: \alpha_{ij} < 0} \left\{ (\phi_i - 1) \frac{\bar{c}_j}{\alpha_{ij}} \right\}, \min_{i: \phi_i > 0} \left\{ \frac{\phi_i \bar{c}_j}{\phi_{ij}} \right\} \right)$$

It should be noted that the penalties U_i and D_i are costly to calculate relative to the information they offer so many commercial codes have abandoned them. As a substitute, they allow for the specification of penalty coefficients p_j^+ and p_j^- as part of the input giving $U_i = p_j^+(1 - \phi_i)$ and $D_i = p_j^-\phi_i$. When $p_j^+ = p_j^- = 1$, and the branching variable x_{B_i} is selected such that

$$\max_{i: \phi_i > 0} \min \{U_i, D_i\}$$

the criterion is called maximum integer infeasibility. The idea in general is that a variable whose smallest degradation is largest is most important for achieving integrality. Other rules are also possible, such as $\max_{i: \phi_i > 0} \max \{U_i, D_i\}$. The idea here is that one branch may easily be fathomed by value dominance.

Monotone Variables and Improved Bounds

At Step 3 of the B&B algorithm for the ILP, partitioning has the effect of replacing one node with two, expanding the tree geometrically. For large problems it may not be possible to achieve convergence in a reasonable amount of time, thereby undermining the enumerative procedure. Nevertheless, consider the case where a monotone variable is present at v_j . If row i is chosen for partitioning, v_j will have only one successor. If x_{B_i} is monotone decreasing (increasing), which means that $U_i(D_i)$ is infinite, one need only consider $x_{B_i} \leq \lfloor \bar{b}_i \rfloor$ ($x_{B_i} \geq \lceil \bar{b}_i \rceil$).

When monotone variables are not present, a rule for partitioning and branching that has proven effective in practice is:

1. Partition based on row k where $\hat{P}_k = \max_i \max \{U_i, D_i\}$.
2. If $\hat{P}_k = U_k$ (D_k) branch to $x_{B_k} \leq \lfloor \bar{b}_k \rfloor$ ($x_{B_k} \geq \lceil \bar{b}_k \rceil$).

The rational behind this rule is to take advantage of the fact that the node which is not examined initially has a larger lower bound than its complement. The expectation is that this node will be fathomed at Step 6 and so will never be considered. Suppose, for example, that the incumbent is \bar{f}^0 and that the lower bound at the current node ℓ is \underline{f}^ℓ .

If we partition on row k , the lower bounds at the two successor nodes of ℓ are $\underline{f}^\ell + D_k$ and $\underline{f}^\ell + U_k$, respectively. The branching rule says to examine the node with the smaller bound first, anticipating that before it is necessary to examine its counterpart a new incumbent, call it \bar{f}^{new} , will be found such that $\underline{f}^\ell + \max\{U_k, D_k\} \geq \bar{f}^{\text{new}}$

Special Ordered Sets

Many integer programs with binary variables have SOS type 1 constraints of the form

$$\sum_{j \in S^i} x_j = 1 \quad \text{for } i = 1, \dots, p \quad (3.19)$$

where S^i are disjoint subsets of $N = \{1, \dots, n\}$. Suppose in a solution of an LP relaxation we have $0 < x_k < 1$ for some $k \in S^i$. Conventional branching on x_k given in (3.17) is equivalent to $x_k = 0$ and $\sum_{j \in S^i \setminus \{k\}} x_j = 0$ since the latter equality is equivalent to $x_k = 1$. Now unless there is a good reason for singling out x_k as a variable that is likely to equal 1, the $x_k = 1$ branch probably contains relatively few solutions compared to the $x_k = 0$ branch. If this is the case, almost no progress will have been made since the node with $x_k = 0$ corresponds to nearly the same feasible region as that of its parent.

For this reason, it is advisable to divide the feasible region associated with the parent into nearly equal parts. To accomplish this we partition the set S^i into two disjoint subsets S^{i1} and S^{i2} such that $S^{i2} = S^i \setminus S^{i1}$. This leads to the subset branching rule

$$\sum_{j \in S^{i1}} x_j = 0 \quad \text{or} \quad \sum_{j \in S^{i2}} x_j = 0 \quad (3.20)$$

The conventional rule is a special case of (3.19) with $S^{i1} = \{k\}$. We can use (3.20) for any S^{i1} such that $k \in S^{i1}$ and $\sum_{j \in S^{i1}} x_j < 1$ but it is reasonable to have S^{i1} and S^{i2} of roughly the same cardinality. A simple way to implement (3.20) is by indexing the variables in (3.19) as $x_{i_1}, x_{i_2}, \dots, x_{i_t}$. The choice of S^{i1} is then specified by an index j , $1 \leq j \leq t-1$, and $S^{i1} = \{i_1, \dots, i_j\}$. The only difficulty with the approach is that the bookkeeping requirements can become a burden when compared to the last-in first-out rule of depth-first search. In the next section, we give a simple data structure for representing a binary search tree. To simplify the data structures for subset branching, Ogryczak [O1] proposed a transformation scheme that results in an equivalent binary search tree. If the problem contains n binary variables x_j ($j = 1, \dots, n$), his methods requires the addition of n new binary variables y_j and n new equality constraints but releases the binary restriction on the original n variables. The transformation is:

$$\begin{aligned} y_1 &= x_1 \\ y_j &= y_{j-1} + x_j \quad \text{for } j = 2, \dots, n \end{aligned}$$

with $y_j \in \{0, 1\}$. To make the formulation more compact, it is possible to eliminate the x_j variables by introducing one additional series of constraints:

$$y_1 \leq y_2 \leq \dots \leq y_{r-1} \leq 1$$

where $y_n = 1$ by virtual of the equality in (3.19). Note that for the same reason, it is always possible to eliminate one of the x_j variables.

3.2.5 Zero–One Implicit Enumeration

When all the decision variables in ILP (3.1) are restricted to the values 0 or 1, the resultant problem is referred to as a 0-1 integer program. In theory, all ILPs can be converted to 0-1 IPs as long as an upper bound u_j is known for each x_j . Equation (3.7) gives the transformation from an integer variable to a binary variable. For large u_j , though, the binary representation is likely to yield too many variables to be practical.

The general branch and bound algorithm can be modified to solve 0-1 problems and in so doing, we refer to the procedure as *implicit enumeration*. Because there are n variables, we could solve a 0-1 problem by enumerating all 2^n combinations. The term implicit enumeration implies that many of these will be discarded or fathomed by various feasibility tests and bounds without having to address them explicitly.

Consider the ILP

$$\min \quad f(\mathbf{x}) = \mathbf{c}\mathbf{x} \quad (3.21a)$$

$$\text{subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \text{ binary} \quad (3.21b)$$

Without loss of generality, we can assume that $c_j \geq 0$, since any x_j with $c_j < 0$ can be replaced by $x'_j = 1 - x_j$ yielding a binary problem with a nonnegative \mathbf{c} vector.

For the 0-1 IP (3.21a,b), the equivalent of the initial feasible region (3.16a) is $S^0 = \{\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \text{ binary}\}$. Given a path P_k from v_0 to v_k , the partition at v_k is determined by choosing a free variable x_j and introducing

$$\left\{ S^k \cap \{\mathbf{x} : x_j = 0\}, S^k \cap \{\mathbf{x} : x_j = 1\} \right\} \quad (3.22)$$

The path P_k corresponds to an assignment of binary values to a subset of the variables. Such an assignment is called a *partial solution*. We denote the index set of assigned variables by $W_k \subseteq N$ and let

$$S_k^+ = \{j : j \in W_k \text{ and } x_j = 1\}$$

$$S_k^- = \{j : j \in W_k \text{ and } x_j = 0\}$$

$$S_k^0 = \{j : j \notin W_k\}$$

A completion of W_k is an assignment of binary variables to the free variables specified by the index set S_k^0 .

Bounds

The problem considered at v_k is

$$\begin{aligned} \min f(\mathbf{x}) &= \sum_{j \in S_k^0} c_j x_j + \sum_{j \in S_k^+} c_j \\ \text{subject to } & \sum_{j \in S_k^0} a_{ij} x_j \leq b_i - \sum_{j \in S_k^+} a_{ij} = s_i, \quad i = 1, \dots, m \\ & x_j = 0 \text{ or } 1, \quad j \in S_k^0 \end{aligned} \quad (3.23)$$

Let $T^k = \{\mathbf{x} : x_j = 0 \text{ or } 1, j \in S_k^0\}$. Because $c_j \geq 0$, the relaxed solution \mathbf{x}^k is obtained by setting $x_j = 0, j \in S_k^0$. Thus $\underline{f}^k = \sum_{j \in S_k^+} c_j$. In addition, if $\mathbf{s} = (s_1, \dots, s_m) \geq 0$ then \mathbf{x}^k is feasible to (3.23) and $\bar{f}^k = f_{IP}^k = \sum_{j \in S_k^+} c_j$.

Fathoming

Again, node k can be fathomed if

- (a) $\underline{f}^k = \bar{f}^k$
- (b) $\underline{f}^k \geq \bar{f}^0$

using the bounds derived above. Note that (a) occurs when \mathbf{x}^k is feasible to (3.23). A sufficient condition for (b) to hold is also simple to evaluate. Suppose that for some i

$$t_i = \sum_{j \in S_k^0} \min \{0, a_{ij}\} > s_i$$

In this case, no completion of W_k can satisfy constraint i so $\underline{f}^k = \infty$, and v_k is fathomed. For example, consider the constraint

$$\sum_{j \in S_k^0} a_{ij} x_j = -5x_1 + 4x_2 + 2x_3 - 3x_4 \leq -11 = s_i$$

The computations give $t_i = -8 > s_i = -11$ so node k is fathomed.

Partitioning and Branching

We would like to identify the subset of free variables at v_k of which at least one must be equal to 1 in a feasible completion of W_k . Let $Q_k = \{i : s_i < 0\}$. If $Q_k = \emptyset$, then v_k is fathomed since \mathbf{x}^k is feasible. If $Q_k \neq \emptyset$, let

$$R_k = \{j : j \in S_k^0 \text{ and } a_{ij} < 0 \text{ for some } i \in Q_k\}$$

At least one variable whose index is an element of R_k must equal 1 in any feasible completion of W_k . Therefore, we can partition on some $x_j, j \in R_k$, and then branch to the successor node corresponding to $x_j = 1$. The following rule chooses such a $j \in R_k$ in an attempt to move toward feasibility. Define

$$I_k = \sum_{i=1}^m \max \{0, -s_i\} = - \sum_{i \in Q_k} s_i$$

to be the infeasibility of (3.23). By choosing x_j , the infeasibility at the successor node is

$$I_k(j) = \sum_{i=1}^m \max \{0, -s_i + a_{ij}\}$$

and x_p is selected such that

$$I_k(p) = \min_{j \in R_k} I_k(j)$$

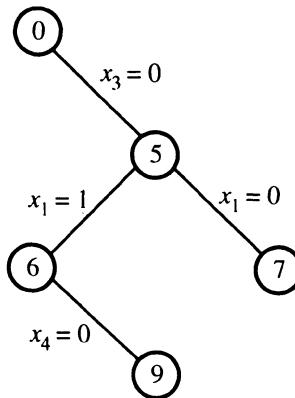
For example, if the constraints at v_k are

$$\begin{aligned} -6x_1 - 2x_2 + 2x_3 &\leq -3 \\ -3x_1 - 4x_2 + x_3 &\leq -2 \\ 7x_1 + 5x_2 - 5x_3 &\leq 4 \end{aligned}$$

then $R_k = \{1, 2\}$, $I_k(1) = 3$ and $I_k(2) = 2$ so x_2 is chosen as the partitioning variable.

Adapting the rule of branching to the most recently generated node simplifies the representation of the search tree. A consequence of the rule and of branching to $x_k = 1$ is that the path P_k uniquely determines the remaining enumeration required. For instance, if $n = 4$ in Fig. 3.8, then at v_9 it is known that all solutions containing $x_3 = 1$ or $x_3 = 0$, $x_1 = 1$ and $x_4 = 1$ have been enumerated, at least implicitly. There remain six possible solutions: $(x_1, x_2, x_3, x_4) = (1, 1, 0, 0), (1, 0, 0, 0), (0, 0, 0, 0), (0, 0, 0, 1), (0, 1, 0, 0), (0, 1, 0, 1)$.

It is possible to present P_k concisely in vector form. In Fig. 3.8, P_9 is represented by the vector $(\underline{3}, 1, \underline{4})$, where the order of the components indicates the level in the tree. Indices appear in the P_k vector if they are in W_k . They appear underlined if they are in S_k^- . (In practice, of course, it is not possible to “underline” indices so one way to account for this condition is to introduce an accompanying vector containing a +1 or -1 in each component. See next subsection.) When branching to $x_p = 1$ from v_k , we simply change P_k to (P_k, p) . In Fig. 3.8, branching to $x_2 = 1$ from v_9 yields $(\underline{3}, 1, \underline{4}, 2)$. In backtracking, we underline the rightmost nonunderlined entry and erase all entries to its right. In the figure, to backtrack from v_9 to v_7 , the vector is transformed from $(\underline{3}, 1, \underline{4})$ to $(\underline{3}, \underline{1})$. When all entries in P_k are underlined and node k is fathomed, every element in W_k has been evaluated at zero and one, so the enumeration is complete. The following algorithm for implementing these ideas is due to Egon Balas.

**Figure 3.8** Example of binary search tree**Additive Algorithm**

- Step 1 (Initialization) At v_0 , $S_0^0 = \{1, \dots, n\}$, $\underline{f}^0 = -\infty$, and $\bar{f}^0 = \infty$. Go to Step 2.
- Step 2 (Calculating bounds) At v_k , let $\underline{f}^k = \sum_{j \in S_k^+} c_j$. If $s_i \geq 0$ for all i , let $\bar{f}^k = f_{IP}^k = \underline{f}^k$ and let $\bar{f}^0 = \min\{\bar{f}^0, \bar{f}^k\}$. Go to Step 3.
- Step 3 (Fathoming) If $t_i > s_i$ for any i , or if $\underline{f}^k = \bar{f}^k$, or if $\underline{f}^k \geq \bar{f}^0$, fathom v_k and go to Step 4. If v_k is live, go to Step 5.
- Step 4 (Backtracking) If no live node exists, go to Step 6. Otherwise branch to the newest live node and go to Step 2.
- Step 5 (Partitioning and branching) Partition on x_p as in (3.22), where $I_k(p) = \min_{j \in R_k} I_k(j)$. Branch in the direction of $x_p = 1$. Go to Step 2.
- Step 6 (Termination) If $\bar{f}^0 = \infty$, there is no feasible solution. If $\bar{f}^0 < \infty$, the feasible solution that yielded \bar{f}^0 is optimal.

Example 3.2.2

$$\begin{aligned}
 \min f(\mathbf{x}) &= 5x_1 + 7x_2 + 10x_3 + 3x_4 + x_5 \\
 \text{subject to} \quad &-x_1 + 3x_2 - 5x_3 - x_4 + 4x_5 \leq -2 \\
 &2x_1 - 6x_2 + 3x_3 + 2x_4 - 2x_5 \leq 0 \\
 &x_2 - 2x_3 + x_4 + x_5 \leq -1 \\
 &x_1, \dots, x_5 = 0 \text{ or } 1
 \end{aligned}$$

At Step 1, we have $S_0^0 = \{1, \dots, 5\}$, $S_0^+ = S_0^- = \emptyset$, $\underline{f}^0 = -\infty$, $\bar{f}^0 = \infty$, $s = (-2, 0, -1)$; $I_0 = 3$, and $R_0 = \{1, 3, 4\}$. Because all variables are free, we go to Step 5 and evaluate $I_0(1) = 4$, $I_0(3) = 3$, and $I_0(4) = 5$. The minimum is associated with $p = 3$ so we select x_3 for partitioning and branch in the direction of $x_3 = 1$. The tree is shown in Fig. 3.9.

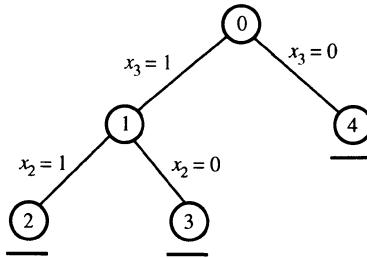


Figure 3.9 Search tree for Example 3.2.2

At v_1 , $P_1 = (3)$, $\underline{f}^1 = 10$, $s = (3, -3, 1)$, $R_1 = \{2, 5\}$, $I_1(2) = 0$, $I_1(5) = 2$, so we choose x_2 for partitioning. No feasible solution is available yet so $\bar{f}^0 = \infty$. At v_2 , $P_2 = (3, 2)$, $\underline{f}^1 = 17$, and $s = (0, 3, 0)$ so $\bar{f}^2 = 17$. We now put $\bar{f}^0 = \bar{f}^2 = 17$ and fathom v_2 . At Step 4, we backtrack to v_3 .

The computations continue at v_3 , with $P_3 = (3, \underline{2})$, $s = (3, -3, 1)$, and $R_3 = \{5\}$. But $t_2 = -2 > s_2 = -3$ so v_3 is fathomed and we backtrack to v_4 . At v_4 , $P_4 = (\underline{3})$, $s = (-2, 0, -1)$, and $R_4 = \{1, 4\}$. But $t_3 = 0 > s_3 = -1$ so v_4 is fathomed. At this point there are no live nodes so we go to Step 6 and terminate. The optimal solution is $f_{IP} = 17$, $x^* = (0, 1, 1, 0, 0)$.

Many additional tests have been proposed that can reduce the enumeration at the expense of added calculation. In general, they are straightforward and easy to implement. The interested reader is referred to [S1] among others.

3.2.6 General Branching and Data Structures

It is an easy matter to modify the branching rules in the additive algorithm to make it more general. For example, the requirement of branching to $x_p = 1$ in Step 5 can be relaxed by altering the vector representation of P_k . If branching to $x_p = 0$ prior to considering the node where $x_p = 1$ is desired, the bookkeeping scheme has to be extended a bit. In particular, if $j \in W_k$, let it appear in P_k as

$$\begin{cases} j & \text{if } j \in S_k^+ \text{ and } x_j = 0 \text{ has not been considered} \\ \underline{j} & \text{if } j \in S_k^+ \text{ and } x_j = 0 \text{ has been considered} \\ -j & \text{if } j \in S_k^- \text{ and } x_j = 1 \text{ has not been considered} \\ \underline{-j} & \text{if } j \in S_k^- \text{ and } x_j = 1 \text{ has been considered} \end{cases}$$

The vector P_k is updated as before except that in backtracking, after erasing the underlined entries on the right, the rightmost remaining entry is underlined and its sign is changed. For example, if the order of nodes considered in Fig. 3.9 had been v_1, v_3, v_2, v_4 , the sequence of vectors would have been $(3), (3, -2), (3, 2), (-3)$.

Branching can further be generalized by choosing to fix more than one variable at a time. Instead of branching to a successor of v_k , one can branch to any node on the path out of v_k . As long as backtracking is done as specified above, there is no danger of overlooking the optimal solution. For example, in the tree shown in Fig. 3.10, one could branch directly from v_1 to v_6 . The vector sequence would be $(3), (3, -2, -4)$.

This option is especially valuable for starting the enumeration. It allows us to begin at a node other than v_0 . In particular, if a good feasible solution \mathbf{x}^0 is known at the outset, we can start at the corresponding node. Let $\{j : x_j^0 = 1\} = \{j_1, \dots, j_k\}$; then the vector representation of the corresponding node is (j_1, \dots, j_k) . Because \mathbf{x}^0 is feasible the node is fathomed, $\bar{f}^0 = c\mathbf{x}^0$ and we backtrack. In Fig. 3.10, if we were to start at v_3 , we would set $P_3 = (3, 2)$. It is also possible to initiate the computations at an infeasible node. In this case, backtracking does not occur automatically.

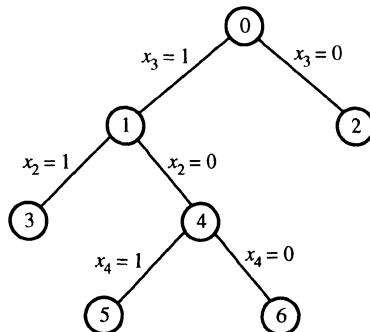


Figure 3.10 Binary search tree with arbitrary branching

It is possible to generalize the above data structure when using a LIFO rule to accommodate variables that can take on any integer value. Assume that in an LP solution to an integer linear program $x_1 = 4.6$. The partitioning step leading to the development of a search tree requires adding two disjunction constraints to the problem; i.e., either

$$x_1 \leq 4 \quad \text{or} \quad x_1 \geq 5$$

To keep track of these constraints, the order in which they are added, the direction of their inequalities, whether or not both have been explored, as well as the corresponding information for subsequent constraints, we introduce the following modified data structure.

- P Path vector indicating the depth of the search tree (number of additional bound constraints imposed). Each element corresponds to the variable begin restricted. A negative sign indicates that the other side of the inequality has not been explored.
- Σ Sign vector associated with P . Each element is either -1 or $+1$, the former indication $a \leq$ inequality, the latter $a \geq$ inequality.
- Ω Value vector associated with P . Each element gives the right-hand-side value of the variable bound in the inequality.

(Note that it is possible to combine Σ and Ω into a single vector.)

Example 3.2.3 Assume that after three iterations in the solution to an ILP the following bounds have been imposed: $x_2 \leq 3$, $x_3 \leq 4$, $x_1 \geq 6$. This gives rise to the search tree shown in Fig. 3.11. The corresponding data structures are $P = (-2, -3, -1)$, $\Sigma = (-1, -1, +1)$, $\Omega = (3, 4, 8)$.

Additional restrictions are added by extending the three vectors accordingly. For example, if the next restriction was $x_2 \geq 2$, we would have $P = (-2, -3, -1, -2)$, $\Sigma = (-1, -1, +1, +1)$, $\Omega = (3, 4, 8, 2)$. Note that the same variable can appear in P more than once.

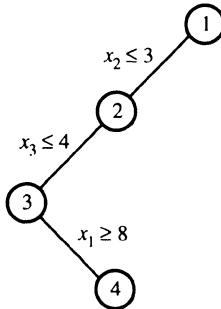


Figure 3.11 General branch and bound tree

Backtracking: When a node is fathomed it is necessary to backtrack. This means creating a new problem by generating the complement of the inequality of the most recent node whose complement has not yet been explored. Again this is accomplished by scanning the P vector to find the rightmost element whose sign is negative. The

sign is then made positive and all elements to its right are removed. The corresponding elements in Σ and Ω are also removed. Next, the sign of the element in Σ associated with the rightmost negative element in P is changed and the same element in Ω is updated. If the element in Σ went from -1 to $+1$ (indicating that the new inequality is \geq), the corresponding element in Ω is increased by 1; if the element in Σ went from $+1$ to -1 (indicating that the new inequality is \leq), the corresponding element in Ω is decreased by 1. When the P vector is empty the search is complete and the algorithm stops with the optimal solution.

Example 3.2.3 (continued) Assume that in the example shown in Fig. 3.11, node 4 is fathomed. Figure 3.12 shows the extended tree that results from backtracking. This gives rise to the following vectors: $P = (-2, -3, +1)$, $\Sigma = (-1, -1, -1)$, $\Omega = (3, 4, 7)$. If node 5 is also fathomed backtracking would give $P = (-2, +3)$, $\Sigma = (-1, +1)$, $\Omega = (3, 5)$. This process continues until all combinations of restrictions have been (implicitly) explored.

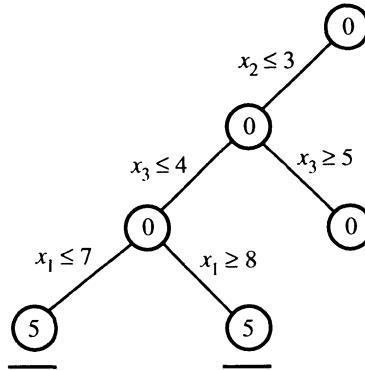


Figure 3.12 Extension of tree with backtracking

3.3 CUTTING PLANES

The idea behind the cutting plane approach for solving ILPs and MILPs is to generate additional linear constraints which, when added to the original problem, produce an optimal LP solution with integer-valued decision variables. In general, the additional constraints, called *cuts*, successively remove portions of the feasible region until an optimum is found, without eliminating any integer-valued solutions.

The use of cutting planes in the oldest approach to solving integer programs dating back to the work of Dantzig, Fulkerson and Johnson in 1954 on the TSP. In 1958, Gomory developed an algorithm applicable to the general ILP. We discuss four different procedures. The first is a dual approach (i.e., no feasible solution is available until optimality is reached) developed by Gomory in 1963 called the *method of integer*

forms. The second is a primal approach (all intermediate solutions are feasible) developed by Ben-Israel and Charnes in 1962 and later modified by Young and Glover. The third is a simplified version of the dual cuts where all the coefficients are unit value. We conclude with an outline of polyhedral theory and the notion of strong valid inequalities.

3.3.1 Method of Integer Forms

Consider the ILP (3.1) will all–integer data and note that any set of values of the decision variables satisfying the constraints must also satisfy any relations derived by row operations on the individual constraints. For example, we could add all the constraints to obtain

$$\sum_{i=1}^m \sum_{j=1}^n a_{ij} x_j = \sum_{i=1}^m b_i$$

Now suppose that we perform some manipulations to obtain a new constraint of the form

$$\sum_{j=1}^n a_j x_j = b \quad (3.24)$$

When we restrict the decision variables to be nonnegative and integer–valued, it must be true that

$$\sum_{j=1}^n \lfloor a_j \rfloor x_j \leq b \quad (3.25)$$

Moreover, because the left-hand side of eq. (3.25) must be an integer, we have

$$\sum_{j=1}^n \lfloor a_j \rfloor x_j \leq \lfloor b \rfloor \quad (3.26)$$

Writing (3.26) as an equality by introducing a slack variable s gives

$$\sum_{j=1}^n \lfloor a_j \rfloor x_j + s = \lfloor b \rfloor \quad (3.27)$$

where $s \geq 0$ and integer. Now, using the notation $a_j = \lfloor a_j \rfloor + \phi_j$, $b = \lfloor b \rfloor + \phi$, we can rewrite (3.24) as

$$\sum_{j=1}^n (\lfloor a_j \rfloor + \phi_j) x_j = \lfloor b \rfloor + \phi \quad (3.28)$$

Subtracting (3.28) from (3.27) yields

$$\sum_{j=1}^n -\phi_j x_j + s = -\phi \quad (3.29)$$

This relationship defines a Gomory fractional cut. For x_B , a basic variable with a noninteger value, the actual cutting plane is

$$\sum_{j \in Q} (-\phi_{ij})x_j + s = -\phi_i \quad (3.30)$$

where Q is the set of nonbasic variables. A more general version of (3.30) is derived in Section 3.3.3.

Notice that s must be negative in (3.30) so it is not feasible. Moreover, if the cut is added to an optimal tableau, the current solution will still have nonnegative reduced costs. This implies that dual simplex algorithm should be used to reoptimize. It can be shown that eq. (3.30) will be satisfied by all integer-valued feasible solutions, so it will never eliminate the optimum. This leads to the following rudimentary algorithm.

Algorithm for the Method of Integer Forms

- Step 1 (Initialization) Solve the LP relaxation of the ILP (3.1) and go to Step 2.
- Step 2 (Optimality test) If the solution to the LP relaxation is all-integer, it is optimal to (3.1); if not, go to Step 3.
- Step 3 (Cutting and pivoting) Choose a row r with $\phi_r > 0$ and add the constraint (3.30) with $i = r$ to the bottom of the tableau. Reoptimize using the dual simplex method. (This may take more than one iteration.) Go to Step 2.

At Step 3, there may be many rows that could serve as the source for the cut. To guarantee finite convergence, we

- (a) maintain $\alpha_j \triangleq (\bar{c}_j, \alpha_{1j}, \dots, \alpha_{mj})^T \succ 0$ for all $j \in Q$ (nonbasic);
- (b) delete the row corresponding to a slack variable from a cut if such a variable becomes basic during the restoration of primal feasibility; and
- (c) choose the source row to be the topmost row with $\phi_r > 0$.

At the completion of Step 1, the solution is dual feasible. However, if there is dual degeneracy in the optimal LP solution, there may be one or more columns having $\alpha_j \prec 0$, $j \in Q$. That is, the nonbasic column α_j is lexicographically negative (see Section 2.2.4 for more discussion of lexicographic ordering). It may be the case that a simple rearrangement of the rows will yield $\alpha_j \succ 0$, for all $j \in Q$. If not, this condition can be achieved by the addition of a superfluous constraint. Because we have assumed that the constraints of (3.1) are bounded, there exists a suitably large integer M such that the constraint

$$\sum_{j \in Q} x_j \leq M \quad \text{or} \quad s = M - \sum_{j \in Q} x_j, \quad s \geq 0 \quad \text{and integer}$$

is implied by the constraint set of (3.1). Add the equation $\sum_{j \in Q} x_j + s = M$ as the first row of the tableau. Since the coefficients of this row are +1 for all $j \in Q$, we have $\alpha_j \succ 0$ for all j nonbasic.

If pivot rows were never deleted, $\alpha_j \succ 0$ for all $j \in Q$ would be maintained by choosing the pivot column q such that

$$\bar{\alpha}_q = \operatorname{lexmax}_{j \in Q_r} \bar{\alpha}_j$$

where r is the pivot row, $\bar{\alpha}_j = \alpha_j / \alpha_{rj}$, and $Q_r = \{j : j \in Q, \alpha_{rj} < 0\}$. Since cut rows are put at the bottom of the tableau, they do not disturb the lexicography property. It remains to be shown that dropping the pivot row after pivoting on a column corresponding to a slack variable from the cut in accordance with (b) above, does not destroy $\alpha_j \succ 0$ for any $j \in Q$. The interested reader is referred to [N3] for the details.

Example 3.3.1

$$\begin{array}{lll} \min & f & = -4x_1 - 11x_2 \\ \text{subject to} & 2x_1 - x_2 + x_3 & = 4 \\ & 2x_1 + 5x_2 + x_4 & = 16 \\ & -x_1 + 2x_2 + x_5 & = 4 \\ & x_1, \dots, x_5 & \geq 0 \text{ and integer} \end{array}$$

The initial and final tableaux are given in Fig. 3.13. The optimal solution to the linear program is $\mathbf{x}^* = (4/3, 8/3, 4, 0, 0)$ with $f_{LP}^* = -104/3$. This means that we can use x_1 and x_2 to generate cuts of the form (3.30). For x_1 we have

$$-\frac{2}{9}x_4 + \left(-\frac{4}{9}\right)x_5 + s_1 = -\frac{1}{3}$$

and for x_2 we have

$$-\frac{1}{9}x_4 + \left(-\frac{2}{9}\right)x_5 + s_2 = -\frac{2}{3}$$

In practice, the cut is often chosen by using the basic variable with the largest fractional part ϕ_i (in this case x_2). Adding this cut to the problem yields the tableau in Fig. 3.14a. Since the basis shown is optimal but not feasible, we make a dual simplex pivot. The exiting variable is s_2 and the entering variable is the one with the minimum value of θ determined by (2.23); in this case it is x_5 . The results of the pivot are shown in Fig. 3.14b. All basic variables are integer-valued and the reduced costs are positive for the nonbasic variables, implying that the solution is optimal. We have $\mathbf{x}^* = (3, 2, 0, 0, 3)$ with $f_{IP}^* = -34$.

The problem in Example 3.3.1 can also be interpreted geometrically as shown in Fig. 3.15. The cut is $-\frac{1}{9}x_4 - \frac{2}{9}x_5 \leq -\frac{2}{3}$ without the slack variable. Substituting

	x_1	x_2	x_3	x_4	x_5	\bar{b}_i
$-f$	-4	-11	0	0	0	0
x_3	2	-1	1	0	0	4
x_4	2	5	0	1	0	16
x_5	-1	(2)	0	0	1	4

(a)

	x_1	x_2	x_3	x_4	x_5	\bar{b}_i
$-f$	0	0	0	19/9	2/9	104/3
x_3	0	0	1	-1/3	4/3	4
x_1	1	0	0	2/9	-5/9	4/3
x_2	0	1	0	1/9	2/9	8/3

(b)

Figure 3.13 (a) Initial tableau for LP, Example 3.3.1; (b) Final tableau for LP

	x_1	x_2	x_3	x_4	x_5	s_2	\bar{b}_i
$-f$	0	0	0	19/9	2/9	0	104/3
x_3	0	0	1	-1/3	4/3	0	4
x_1	1	0	0	2/9	-5/9	0	4/3
x_2	0	1	0	1/9	2/9	0	8/3
s_2	0	0	0	-1/9	(-2/9)	1	-2/3

(a)

	x_1	x_2	x_3	x_4	x_5	s_2	\bar{b}_i
$-f$	0	0	0	2	0	1	34
x_3	0	0	1	-1	0	6	0
x_1	1	0	0	1/2	0	-5/2	3
x_2	0	1	0	0	0	1	2
x_5	0	0	0	1/2	1	-9/2	3

(b)

Figure 3.14 (a) First cut added to tableau, Example 3.3.1; (b) Optimal tableau for IP

$x_4 = 16 - (2x_1 + 5x_2)$ and $x_5 = 4 - (-x_1 + 2x_2)$ gives

$$-\frac{1}{9}(16 - 2x_1 - 5x_2) - \frac{2}{9}(4 + x_1 - 2x_2) \leq -\frac{2}{3}$$

or

$$x_2 \leq 2$$

Adding the cut reduces the feasible region of the LP relaxation in Fig. 3.15 to an integer polytope where each vertex has an all-integer representation. Optimizing over the reduced polytope thus produces the optimal integer solution after only one iteration. However, this is rarely if ever the case in practice. In general, fractional cuts of the type given in (3.30) are neither facets nor faces of the underlying polytope defined by the convex hull of all integer solutions. Many cuts are often needed before convergence occurs. For this reason, the method of integer forms has fallen out of favor.

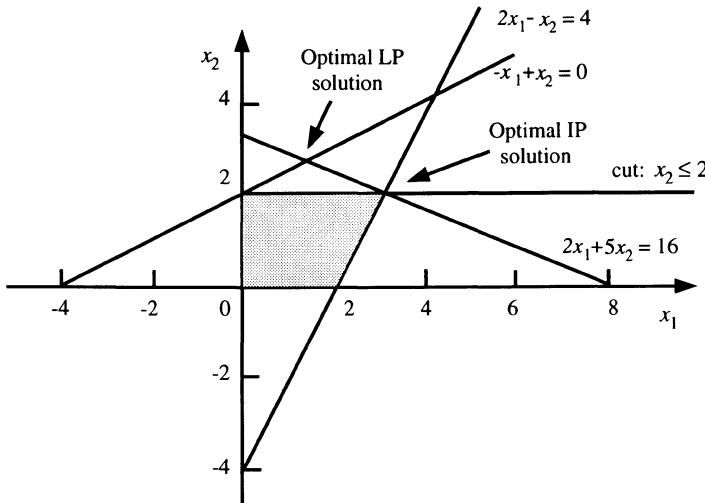


Figure 3.15 Graphical interpretation of Example 3.3.1

3.3.2 Primal All-Integer Cuts

The method of integer forms is a dual approach to the ILP (3.1) and will not produce an integer-valued solution if the algorithm is terminated prior to convergence, perhaps due to excessive run time. To avoid this situation, primal methods have been developed. The following algorithm is called the primal all-integer method because it always maintains a primal feasible tableau that contains only integers. The idea is as follows: Suppose we start with an all-integer tableau, which is always possible

with rational data. For a minimization problem, the cut is generated by any nonbasic variable with a negative reduced cost, say x_k . Let

$$\frac{\bar{b}_r}{\alpha_{rk}} = \min_{i=1,\dots,m} \frac{\bar{b}_i}{\alpha_{ik}} \quad (3.31)$$

where $\alpha_{ik} \geq 1$. Now if $\alpha_{rk} = 1$, performing a standard simplex pivot will make α_{rk} the pivot element and maintain an all-integer tableau. However, when $\alpha_{rk} > 1$, we add the cut

$$\sum_{j \in Q} \left\lfloor \frac{\alpha_{rj}}{\alpha_{rk}} \right\rfloor x_j + s = \left\lfloor \frac{\bar{b}_r}{\alpha_{rk}} \right\rfloor \quad (3.32)$$

where $s \geq 0$ and integer. This ensures a pivot element of 1 because $\alpha_{rk}/\alpha_{rk} = 1$ which maintains integrality.

Algorithm

- Step 1 (Initialization) Begin with an all-integer primal feasible tableau. If an obvious one does not exist, employ a phase 1 procedure. Go to Step 2.
- Step 2 (Test for optimality) If the solution is dual feasible, it is optimal to (3.1). If not, go to Step 3.
- Step 3 (Cutting and pivoting) Choose a column $k \in Q$ with $\bar{c}_k < 0$ and a row r satisfying (3.31). If $\alpha_{rk} = 1$, execute a primal simplex iteration with x_k as the entering variable and x_{B_r} as the leaving variables. If $\alpha_{rk} > 1$, adjoin the constraint (3.32) to the bottom of the tableau. Execute a primal simplex iteration with x_s as the leaving variable and x_k as the entering variable. In any case, if a slack variable from a cut becomes basic, its row is deleted. Return to Step 2.

Example 3.3.2

$$\begin{aligned} \min f &= -x_1 - 4x_2 \\ \text{subject to } &5x_1 + 7x_2 + x_3 = 21 \\ &-x_1 + 3x_2 + x_4 = 8 \\ &x_1, \dots, x_4 \geq 0 \text{ and integer} \end{aligned}$$

The initial tableau is given in Fig. 3.16a. Variable x_2 has the most negative reduced cost. Evaluating (3.31) gives $\bar{b}_r/\alpha_{r2} = \min\left(\frac{21}{7}, \frac{8}{3}\right) = \frac{8}{3}$ which means that $r = 2$ and $\alpha_{22} = 3$. Because $\alpha_{22} > 1$, we use (3.32) to generate the cut

$$\left\lfloor -\frac{1}{3} \right\rfloor x_1 + \left\lfloor \frac{3}{3} \right\rfloor x_2 + s_1 = \left\lfloor \frac{8}{3} \right\rfloor$$

or

$$-x_1 + x_2 + s_1 = 2$$

which is added to the original tableau, as shown in Fig. 3.16b. After pivoting in Step 3, we get the tableau shown in Fig. 3.17a. Variable x_1 has a negative reduced cost. Evaluating (3.31) gives $\bar{b}_r/\alpha_{r1} = \min\left(\frac{7}{12}, \frac{2}{2}\right)$ which means that $r = 1$ and $\alpha_{11} = 12$. Again because $\alpha_{11} > 1$, we generate the cut

$$\left\lfloor \frac{12}{12} \right\rfloor x_1 + \left\lfloor \frac{-7}{12} \right\rfloor s_1 + s_2 = \left\lfloor \frac{7}{12} \right\rfloor$$

or

$$x_1 - s_1 + s_2 = 0$$

which is added to the second tableau, as shown in Fig. 3.17b. The pivot results are given in Fig. 3.18a. Variable s_1 has a negative reduced cost. Evaluating (3.31) gives $\bar{b}_r/\alpha_{rs_1} = \min\left(\frac{7}{5}, \frac{7}{5}\right) = \frac{7}{5}$ which means that $r = 1$ and $\alpha_{1s_1} = 5$. We generate the cut

$$\left\lfloor \frac{5}{5} \right\rfloor s_1 + \left\lfloor \frac{-12}{5} \right\rfloor s_2 + s_3 = \left\lfloor \frac{7}{5} \right\rfloor$$

or

$$s_1 - 3s_2 + s_3 = 1$$

which is added to the third tableau, as shown in Fig. 3.18b. The result of the pivot is shown in Fig. 3.19 where it can be seen that the solution is optimal. We have $\mathbf{x}^* = (1, 2, 2, 3)$ with $f^* = 9$.

	x_1	x_2	x_3	x_4	\bar{b}_i
$-f$	-1	-4	0	0	0
x_3	5	7	1	0	21
x_4	-1	3	0	1	8

(a)

	x_1	x_2	x_3	x_4	s_1	\bar{b}_i
$-f$	-1	-4	0	0	0	0
x_3	5	7	1	0	0	21
x_4	-1	3	0	1	0	8
s_1	-1	①	0	0	1	2

(b)

Figure 3.16 (a) Initial tableau for Example 3.3.2; (b) Initial tableau with first cut

	x_1	x_2	x_3	x_4	s_1	\bar{b}_i
$-f$	-5	0	0	0	4	8
x_3	12	0	1	0	-7	7
x_4	2	0	0	1	-3	2
x_2	-1	1	0	0	2	2

(a)

	x_1	x_2	x_3	x_4	s_1	s_2	\bar{b}_i
$-f$	-5	0	0	0	4	0	8
x_3	12	0	1	0	-7	0	7
x_4	2	0	0	1	-3	0	2
x_2	-1	1	0	0	2	0	2
s_2	1	0	0	0	-1	1	0

↓

→

(b)

Figure 3.17 (a) Second tableau, Example 3.3.2; (b) Second tableau with second cut

A geometric interpretation of Example 3.3.2 is given in Fig. 3.20. The first cut was

$$-x_1 + x_2 + s_1 = 2 \quad \text{or} \quad -x_1 + x_2 \leq 2$$

the second cut was $x_1 - s_1 + s_2 = 0$, but $s_1 = 2 + x_1 - x_2$ so we have upon substitution $x_2 \leq 2$. The third cut was $s_1 - 3s_2 + s_3 = 1$ which after a similar substitution for s_1 and s_2 gives $x_1 + 2x_2 \leq 5$.

The imposition of the first cut resulted in the vertex at $x_1 = 0, x_2 = 2$. The imposition of the second cut resulted in a degenerate pivot so there was no movement from the first vertex. The third cut produced the optimal solution.

Finiteness

If the cut row is never degenerate ($\bar{b}_r/\alpha_{rk} \geq 1$), the sequence of objective function values $\{f^t\}$ is decreasing and the primal all-integer algorithm is finite. While this condition is sufficient, it is not necessary for finite convergence as demonstrated in Example 3.3.2. To prove finiteness, it must be shown that an infinite sequence of degenerate cuts is impossible. This can be accomplished by giving specific pivot column and source row selection rules. The details are discussed in [S1].

	x_1	x_2	x_3	x_4	s_1	s_2	\bar{b}_i
$-f$	0	0	0	0	-1	5	8
x_3	0	0	1	0	5	-12	7
x_4	0	0	0	1	-1	-2	2
x_2	0	1	0	0	0	1	2
x_1	1	0	0	0	-1	1	0

(a)

	x_1	x_2	x_3	x_4	s_1	s_2	s_3	\bar{b}_i
$-f$	0	0	0	0	-1	5	0	8
x_3	0	0	1	0	5	-12	0	7
x_4	0	0	0	1	-1	-2	0	2
x_2	0	1	0	0	0	1	0	2
x_1	1	0	0	0	-1	1	0	0
s_3	0	0	0	0	(1)	-3	1	1

↓

→

(b)

Figure 3.18 (a) Third tableau, Example 3.3.2; (b) Third tableau with third cut

	x_1	x_2	x_3	x_4	s_1	s_2	s_3	\bar{b}_i
$-f$	0	0	0	0	0	2	1	9
x_3	0	0	1	0	0	3	-5	2
x_4	0	0	0	1	1	-5	1	3
x_2	0	1	0	0	0	1	0	2
x_1	1	0	0	0	0	-2	1	1
s_1	0	0	0	0	1	-3	1	1

Figure 3.19 Final tableau, Example 3.3.2

3.3.3 Cuts with Unit Coefficients

In this section a simpler class of cuts than (3.30) and (3.32) is presented but only a subclass of these can yield a finite algorithm. We begin by noting that if for any i , $\phi_i > 0$ in a basic solution determined from (2.27a), then a necessary condition for an integer solution is that at least one of the nonbasic variables be positive. Because the nonbasic variables are also constrained to be integer, this means that at least one of

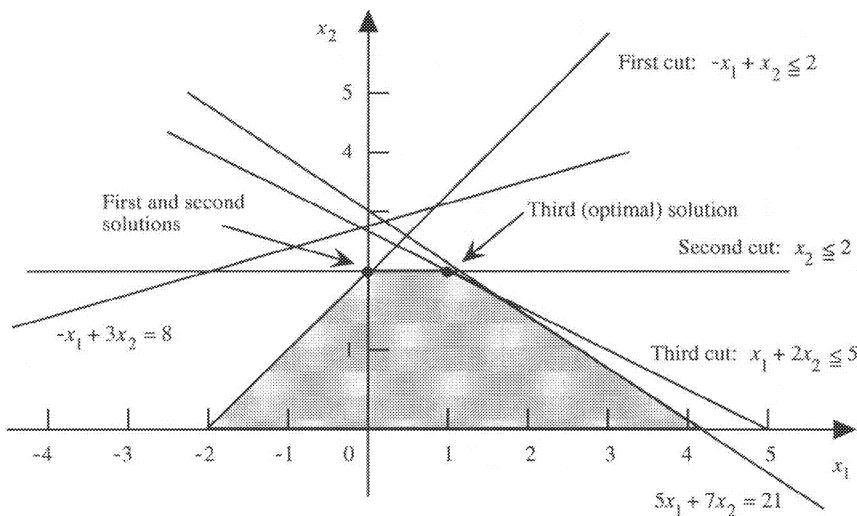


Figure 3.20 Graphical representation of Example 3.3.2

them must be equal to or greater than one. Thus

$$\sum_{j \in Q} x_j \geq 1 \quad (3.33)$$

This cut excludes the current basic solution but no feasible integer solutions. In Section 6.4, we use a version of (3.33) with the additive algorithm discussed in Section 3.2.5 to solve the 0-1 bilevel programming problem. A nice property of (3.33) is that it does not depend on the α_{ij} . One simply checks if any basic variables are noninteger, and, if so, adds the constraint. Nevertheless, using only cuts derived from (3.33) cannot, in general, yield a finite algorithm. It can be shown that if \mathbf{x}^* is an optimal solution to the ILP (3.1), a necessary (but not sufficient) condition for an algorithm based on (3.33) to converge to \mathbf{x}^* is that \mathbf{x}^* be on an edge (a line joining two adjacent extreme points) of the constraint set of the LP relaxation $T = \{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$.

Improved versions of (3.33) are obtained by noting that $\phi_i > 0$ and x_{B_i} integer imply that

$$\sum_{j \in Q_i} x_j \geq 1, \quad Q_i = \{j : j \in Q, \alpha_{ij} \neq 0\} \quad (3.34)$$

and more strongly

$$\sum_{j \in Q_i^0} x_j \geq 1, \quad Q_i^0 = \{j : j \in Q, \phi_{ij} > 0\} \quad (3.35)$$

For the cuts (3.34) and (3.35) a finite algorithm exists although the details are unimportant since it is unlikely that an algorithm based on these cuts is going to be better

than an algorithm based on the cut $\sum_{j \in Q} \phi_{ij} x_j \geq \phi_i$ which is equivalent to (3.30). An essential property used to prove the finiteness of the algorithm based on (3.30) is that if the cut is taken from the objective row, one dual simplex iteration reduces the objective function by at least its fractional part. The use of (3.34) and (3.35) from the objective row only guarantees a reduction of $1/D$ in one dual simplex iteration ($D = |\det \mathbf{B}|$, the absolute value of the determinant of the basis \mathbf{B}). This reduction, however, is sufficient to yield a convergent algorithm. In contrast, when (3.33) is used and the objective function is not integer, one dual simplex iteration may not reduce the objective function.

The cuts (3.30) and (3.35), when taken from the same row, have positive coefficients for the same subset of nonbasic variables. But there is a much closer relationship. To see this, let h be any rational number and x_{B_i} a basic variable. Multiplying (2.27a) by h gives

$$hx_{B_i} + \sum_{j \in Q} h\alpha_{ij} x_j = h\bar{b}_i \quad (3.36)$$

Taking the integer part of the left-hand side of (3.36) and noting that $x_j \geq 0$ leads to

$$\lfloor h \rfloor x_{B_i} + \sum_{j \in Q} \lfloor h\alpha_{ij} \rfloor x_j \leq h\bar{b}_i \quad (3.37)$$

But the left-hand side of (3.37) cannot exceed the integer part of the right-hand side which implies that

$$\lfloor h \rfloor x_{B_i} + \sum_{j \in Q} \lfloor h\alpha_{ij} \rfloor x_j \leq \lfloor h\bar{b}_i \rfloor \quad (3.38)$$

Now, multiplying (2.27a) by the integer part of h yields

$$\lfloor h \rfloor x_{B_i} + \sum_{j \in Q} \lfloor h \rfloor \alpha_{ij} x_j = \lfloor h \rfloor \bar{b}_i \quad (3.39)$$

Subtracting (3.38) from (3.39) produces what is called the *fundamental cut*

$$\sum_{j \in Q} (\lfloor h \rfloor \alpha_{ij} - \lfloor h \alpha_{ij} \rfloor) x_j \geq \lfloor h \rfloor \bar{b}_i - \lfloor h \bar{b}_i \rfloor \quad (3.40)$$

The cutting plane (3.30) can be derived from (3.40) by setting $h = 1$. By setting $h = -1$ in (3.40) we obtain

$$\sum_{j \in Q_i^0} (1 - \phi_{ij}) x_j \geq 1 - \phi_i \quad (3.41)$$

Removing the slack variable in (3.30) and adding it to (3.41) yields (3.35).

3.3.4 Valid Inequalities

A more powerful use of cutting planes to solve IPs involves polyhedral theory. In linear programming, we are given a description of the set of feasible points by a set of linear inequalities $P = \{\mathbf{x} \in R_+^n : \mathbf{Ax} \leq \mathbf{b}\}$. When an LP is solved by the simplex method, we are not interested in issues such as the dimension of P and which inequalities are necessary for the description of P . Integer programming is different. Typically, we are given a set $S \subseteq Z_+^n$ of feasible points described implicitly by, say, the integer solutions to a linear inequality system $S = \{\mathbf{x} \in Z_+^n : \mathbf{Ax} \leq \mathbf{b}\}$. If we could find an explicit representation of this set also in terms of linear inequalities then we could solve the IP by minimizing the objective function over this second set of linear inequalities. To formalize this idea, a few definitions are needed. This material is taken primarily from [B1] and [N3].

Definition 3.3.1 Given a set $S \subseteq R^n$, a point $\mathbf{x} \in R^n$ is a *convex combination* of points of S if there exists a finite set of points $\{\mathbf{x}^i\}_{i=1}^t$ in S and a $\lambda \in R_+^t$ with $\sum_{i=1}^t \lambda_i = 1$ and $\mathbf{x} = \sum_{i=1}^t \lambda_i \mathbf{x}^i$. The *convex hull* of S , denoted by $\text{conv}(S)$, is the set of all points that are convex combinations of points in S .

Figure 3.21 shows the convex hull of a set of integral points in R^n . It can be seen that $\text{conv}(S)$ can be described by a finite set of linear inequalities and that $\min\{\mathbf{c}\mathbf{x} : \mathbf{x} \in S\} = \min\{\mathbf{c}\mathbf{x} : \mathbf{x} \in \text{conv}(S)\}$. Further, the latter problem is a linear program. The actual process of identifying the inequalities necessary to describe $\text{conv}(S)$ and hence being able to solve an LP to find the solution to an IP is not as easy as it might sound. Success has only been achieved for certain classes of combinatorial optimization problems, such as the TSP, which exhibit special structure.

Definition 3.3.2 A set of points $\mathbf{x}^1, \dots, \mathbf{x}^k \in R^n$ is *linearly independent* if the unique solution of $\sum_{i=1}^k \lambda_i \mathbf{x}^i = \mathbf{0}$ is $\lambda_i = 0, i = 1, \dots, k$.

Note that the maximum number of linearly independent points in R^n is n .

Proposition 3.3.1 If \mathbf{A} is an $m \times n$ matrix, the maximum number of linearly independent rows of \mathbf{A} equals the maximum number of linearly independent columns of \mathbf{A} .

Definition 3.3.3 The maximum number of linearly independent rows (columns) of \mathbf{A} is the *rank* of \mathbf{A} and is denote by $\text{rank}(\mathbf{A})$.

Proposition 3.3.2 The following statements are equivalent

- (a) $\{\mathbf{x} \in R^n : \mathbf{Ax} = \mathbf{b}\} \neq \emptyset$

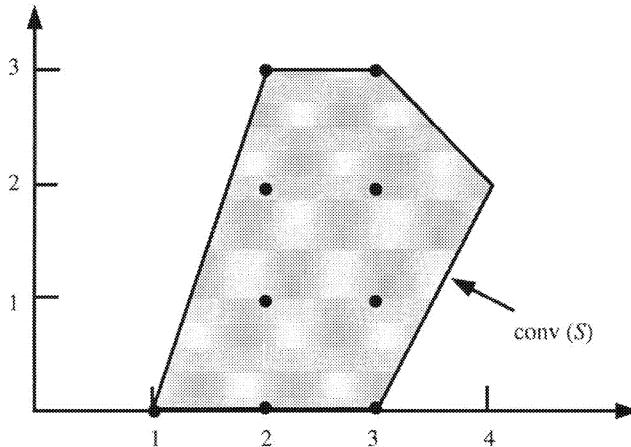


Figure 3.21 Example of S (black dots) and corresponding $\text{conv}(S)$

$$(b) \text{ rank } (\mathbf{A}) = \text{rank}(\mathbf{A}, \mathbf{b})$$

When dealing with linear equalities and inequalities it is often more advantageous to use the concept of affine independence.

Definition 3.3.4 A set of points $\mathbf{x}^1, \dots, \mathbf{x}^k \in R^n$ is *affinely independent* if the unique solution of $\sum_{i=1}^k \alpha_i \mathbf{x}^i = \mathbf{0}$, $\sum_{i=1}^k \alpha_i = 0$ is $\alpha_i = 0$, $i = 1, \dots, k$.

Linear independence implies affine independence but the converse is not true. Also, the *affine hull* of S , denoted by $\text{aff}(S)$, is defined in a similar manner as the convex hull but using affine combinations instead of convex combinations of vectors.

Proposition 3.3.3 The following statements are equivalent:

- (a) $\mathbf{x}^1, \dots, \mathbf{x}^i \in R^n$ are affinely independent
- (b) $\mathbf{x}^2 - \mathbf{x}^1, \dots, \mathbf{x}^k - \mathbf{x}^1$ are linearly independent
- (c) $(\mathbf{x}^1, -1), \dots, (\mathbf{x}^k, -1) \in R^{n+1}$ are linearly independent

Note that the maximum number of affinely independent points in R^n is $n + 1$ (e.g., n linearly independent points and the zero vector). The following proposition is used frequently in proving results concerning polyhedra.

Proposition 3.3.4 If $\{\mathbf{x} \in R^n : \mathbf{Ax} = \mathbf{b}\} \neq \emptyset$, the maximum number of affinely independent solutions of $\mathbf{Ax} = \mathbf{b}$ is $n + 1 - \text{rank}(\mathbf{A})$.

We now turn to some definitions related to polyhedra.

Definition 3.3.5 A polyhedron $P \subseteq R^n$ is a set of points that satisfies a finite number of linear inequalities; that is, $P = \{\mathbf{x} \in R^n : \mathbf{Ax} \leq \mathbf{b}\}$, where (\mathbf{A}, \mathbf{b}) is an $m \times (n+1)$ matrix. A polyhedron is said to be *rational* if there exists an $m' \times (n+1)$ matrix $(\mathbf{A}', \mathbf{b}')$ with rational coefficients such that $P = \{\mathbf{x} \in R^n : \mathbf{A}'\mathbf{x} \leq \mathbf{b}'\}$.

Definition 3.3.6 A polyhedron P is of *dimension* k , denoted by $\dim(P) = k$, if the maximum number of affinely independent points in P is $k+1$.

Definition 3.3.7 A polyhedron $P \subseteq R^n$ is *full-dimensional* if $\dim(P) = n$.

It can be shown that if P is not full-dimensional, then at least one of the inequalities $\mathbf{a}_i \mathbf{x} \leq b_i$ is satisfied at equality by all points in P . Alternatively, Definition 3.3.7 is equivalent to saying that there is no hyperplane containing P . In developing a polyhedral description of the integer feasible region S , it is important to distinguish those inequalities that are in fact equalities for the particular problem. To do this, let $M = \{1, \dots, m\}$, $M^= = \{i \in M : \mathbf{a}_i \mathbf{x} = b_i \text{ for all } \mathbf{x} \in P\}$, and $M^\leq = \{i \in M : \mathbf{a}_i \mathbf{x} < b_i \text{ for some } \mathbf{x} \in P\} = M \setminus M^=$. Let $(\mathbf{A}^=, \mathbf{b}^=)$, $(\mathbf{A}^\leq, \mathbf{b}^\leq)$ be the corresponding rows of (\mathbf{A}, \mathbf{b}) . Accordingly, $P = \{\mathbf{x} \in R^n : \mathbf{A}^= \mathbf{x} = \mathbf{b}^=, \mathbf{A}^\leq \mathbf{x} \leq \mathbf{b}^\leq\}$. Note that if $i \in M^\leq$, then (\mathbf{a}_i, b_i) cannot be written as a linear combination of the rows of $(\mathbf{A}^=, \mathbf{b}^=)$. This follows because \mathbf{a}_i is linearly independent from the rows of $\mathbf{A}^=$, otherwise we could write $\mathbf{a}_i \mathbf{x} = b_i$ for all $\mathbf{x} \in P$ which would contradict the fact that $i \in M^\leq$.

Definition 3.3.8 The vector $\mathbf{x} \in P$ is called an *inner point* of P if $\mathbf{a}_i \mathbf{x} < b_i$ for all $i \in M^\leq$.

Definition 3.3.9 The vector $\mathbf{x} \in P$ is called an *interior point* of P if $\mathbf{a}_i \mathbf{x} < b_i$ for all $i \in M$.

Proposition 3.3.5 Every nonempty polyhedron P has an inner point.

The next step is to relate the dimension of P to the rank of its equality matrix $(\mathbf{A}^=, \mathbf{b}^=)$. Below it will always be assumed that $P \neq \emptyset$ although the following result is still valid with the convention that if $P = \emptyset$, then $\dim(P) = -1$.

Proposition 3.3.6 If $P \subseteq R^n$, then $\dim(P) + \text{rank}(\mathbf{A}^=, \mathbf{b}^=) = n$.

Corollary 3.3.1 A polyhedron P is full-dimensional if and only if it has an interior point.

We now come to what we mean by a valid inequality and a strong cutting plane. Given a polyhedron $P = \{\mathbf{x} \in R^n : \mathbf{Ax} \leq \mathbf{b}\}$, we would like to know which of the inequalities $\mathbf{a}_i \mathbf{x} \leq b_i$ are necessary in the description of P and which can be dropped. In fact, the corresponding set of inequalities can be shown to be invariant up to a scalar multiplication.

Definition 3.3.10 The inequality $\pi\mathbf{x} \leq \pi_0$ [or (π, π_0)] is called a *valid inequality* for P if it is satisfied by all points in P .

Note that (π, π_0) is a valid inequality if and only if P lies in the half space $\{\mathbf{x} \in R^n : \pi\mathbf{x} \leq \pi_0\}$, or equivalently if and only if $\max \{\pi\mathbf{x} : \mathbf{x} \in P\} \leq \pi_0$.

Definition 3.3.11 If (π, π_0) is a valid inequality for P and $F = \{\mathbf{x} \in P : \pi\mathbf{x} = \pi_0\}$, F is called a *face* of P and we say that (π, π_0) represents F . A face is said to be *proper* if $F \neq \emptyset$ and $F \neq P$.

The face F represented by (π, π_0) is nonempty if and only if $\max \{\pi\mathbf{x} : \mathbf{x} \in P\} = \pi_0$. When F is nonempty, we say that (π, π_0) *supports* P . In trying to eliminate inequalities that are superfluous, we can discard inequalities $a_i\mathbf{x} \leq b_i$ that are not supports of P . Hence from now on we assume that all inequalities $a_i\mathbf{x} \leq b_i$ for $i \in M$ support P and therefore represent nonempty faces.

Proposition 3.3.7 If $P = \{\mathbf{x} \in R^n : \mathbf{Ax} \leq \mathbf{b}\}$ with equality set $M^= \subseteq M$, and F is a nonempty face of P , then $F = \{\mathbf{x} \in R^n : a_i\mathbf{x} = b_i \text{ for } i \in M_F^=, a_i\mathbf{x} \leq b_i \text{ for } i \in M_F^<\}$ is a polyhedron, where $M_F^= \subseteq M^=$ and $M_F^< = M \setminus M_F^=$. The number of distinct faces of P is finite.

Note that by Proposition 3.3.7, if F is a proper face of P , then $\dim(F) < \dim(P)$. In particular, the dimension of F is k if the maximum number of affinely independent points that lie in F is $k + 1$.

Definition 3.3.12 A face F of P is a *facet* of P if $\dim(F) = \dim(P) - 1$.

Proposition 3.3.8 If F is a facet of P , there exists some inequality $a_k\mathbf{x} \leq b_k$ for $k \in M^<$ representing F .

Proposition 3.3.9 For each facet F of P , one of the inequalities representing F is necessary in the description of P .

Besides being necessary, the facets are sufficient for the description of P .

Proposition 3.3.10 Every inequality $a_k\mathbf{x} \leq b_k$ for $k \in M^<$ that represents a face of P of dimension less than $\dim(P) - 1$ is irrelevant for the description of P .

Various types of valid inequalities for an ILP are illustrated in Figure 3.22 (the relaxed feasible region is denoted by T and the LP solution by \mathbf{x}_{LP}^*). Ideally, we would like to find all the facets of the underlying polytope associated with the convex hull of feasible integral points ($\text{conv}(S)$ in Fig. 3.21). The fractional and all-integer cuts discussed in previous subsections are not in general facets or even faces, and hence

are considered weak. This is the principal reason why the corresponding methods have not been popular lately.

In any problem, the question arises as to when are two inequalities equivalent. The answer is straightforward. The set

$$\{\mathbf{x} : \mathbf{A}^= \mathbf{x} = \mathbf{b}^=, \boldsymbol{\pi} \mathbf{x} \leq \boldsymbol{\pi}_0\} = \{\mathbf{x} : \mathbf{A}^= \mathbf{x} = \mathbf{b}^=, (\lambda \boldsymbol{\pi} + \boldsymbol{\mu} \mathbf{A}^=) \mathbf{x} \leq \lambda \boldsymbol{\pi}_0 + \boldsymbol{\mu} \mathbf{b}^=\}$$

for all $\lambda > 0$ and all $\boldsymbol{\mu} \in R^{|\mathbf{M}^=|}$. Hence we say that $(\boldsymbol{\pi}^1, \boldsymbol{\pi}_0^1)$ and $(\boldsymbol{\pi}^2, \boldsymbol{\pi}_0^2)$ are *equivalent* or *identical inequalities* with respect to P when $(\boldsymbol{\pi}^2, \boldsymbol{\pi}_0^2) = \lambda(\boldsymbol{\pi}^1, \boldsymbol{\pi}_0^1) + \boldsymbol{\mu}(\mathbf{A}^=, \mathbf{b}^=)$ for some λ and $\boldsymbol{\mu} \in R^{|\mathbf{M}^=|}$.

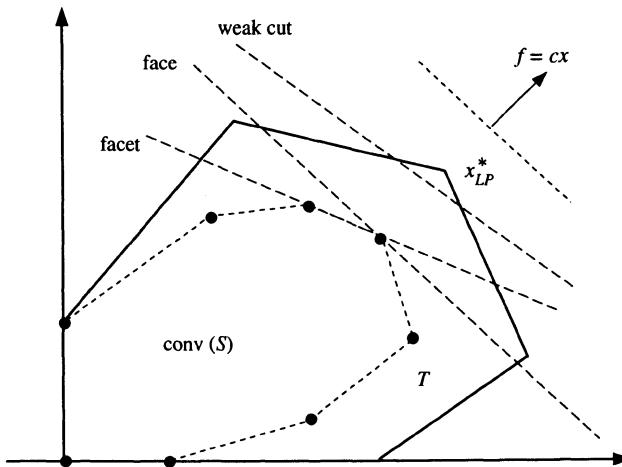


Figure 3.22 Different quality cutting planes for ILPs

The main results given thus far can be summarized as follows:

- A full-dimensional polyhedron P has a unique (to within a scalar multiplication) minimal representation by a finite set of linear inequalities. In particular, for each facet F_i of P there is an inequality $\mathbf{a}_i \mathbf{x} \leq b_i$ (unique to within a scalar multiplication representing F_i) and $P = \{\mathbf{x} \in R^n : \mathbf{a}_i \mathbf{x} \leq b_i \text{ for } i = 1, \dots, r\}$.
- If $\dim(P) = n - k$ with $k > 0$, then $P = \{\mathbf{x} \in R^n : \mathbf{a}_i \mathbf{x} = b_i \text{ for } i = 1, \dots, k; \mathbf{a}_i \mathbf{x} \leq b_i \text{ for } i = k+1, \dots, k+t\}$. For $i = 1, \dots, k$, $\{\mathbf{a}_i, b_i\}$ are a maximal set of linearly independent rows of $(\mathbf{A}^=, \mathbf{b}^=)$, and for $i = k+1, \dots, k+t$, (\mathbf{a}_i, b_i) is any inequality from the equivalence class of inequalities representing facet F_i .

Recall that our primary goal is to find inequality systems to represent the convex hull of the set of feasible integer points to an ILP. For this purpose, facets or facet defining inequalities are of particular importance. We now give a theorem that characterizes facets and that is useful in establishing when a valid inequality is a facet.

Theorem 3.3.1 Let $(\mathbf{A}^=, \mathbf{b}^=)$ be the equality set of $P \subseteq R^n$ and let $F = \{\mathbf{x} \in P : \boldsymbol{\pi}\mathbf{x} = \boldsymbol{\pi}_0\}$ be a proper face of P . The following statements are equivalent:

1. F is a facet of P .
2. F is a maximal proper face of P .
3. $\dim(F) = \dim(P) - 1$.
4. If $\boldsymbol{\gamma}\mathbf{x} = \gamma_0$ for all $\mathbf{x} \in F$ then $(\boldsymbol{\gamma}, \gamma_0) = (\lambda\boldsymbol{\pi} + \boldsymbol{\mu}\mathbf{A}^=, \lambda\boldsymbol{\pi}_0 + \boldsymbol{\mu}\mathbf{b}^=)$ for some $\lambda \in R^1$ and some $\boldsymbol{\mu} \in R^{|\mathbf{M}^=|}$.

Conditions 3 and 4 provide two basic methods to prove that a given inequality $\boldsymbol{\pi}\mathbf{x} \leq \boldsymbol{\pi}_0$ defines a facet of a polyhedron P . In both cases, of course, one has to check that $\boldsymbol{\pi}\mathbf{x} \leq \boldsymbol{\pi}_0$ is valid and that P is not contained in $\{\mathbf{x} \in R^n : \boldsymbol{\pi}\mathbf{x} = \boldsymbol{\pi}_0\}$. This is usually trivial.

The first method consists of exhibiting a set of $k = \dim(P)$ vectors (usually vertices of P) $\mathbf{x}^1, \dots, \mathbf{x}^k \in P$ satisfying $\boldsymbol{\pi}\mathbf{x}^i = \boldsymbol{\pi}_0$, $i = 1, \dots, k$, and showing that these factors are affinely independent. (If $\boldsymbol{\pi}_0 \neq 0$, this is equivalent to showing that the k vectors are linearly independent.) In some cases this method is easy to apply since the linear (or affine) independence of simply structured vectors like the unit vectors are all that is necessary. These are easy to check for independence.

In most nontrivial cases, the second method based on Condition 4 of Theorem 3.3.1, is more suitable. One proceeds by assuming the existence of a valid inequality $\boldsymbol{\gamma}\mathbf{x} \leq \gamma_0$ with $\{\mathbf{x} \in P : \boldsymbol{\pi}\mathbf{x} = \boldsymbol{\pi}_0\} \subseteq \{\mathbf{x} \in P : \boldsymbol{\gamma}\mathbf{x} = \gamma_0\}$ [G13]. Using the known equation system $\mathbf{A}^=\mathbf{x} = \mathbf{b}^=$ for P , one can determine a $\boldsymbol{\mu} \in R^{|\mathbf{M}^=|}$ such that $\bar{\boldsymbol{\gamma}} \triangleq \boldsymbol{\gamma} + \boldsymbol{\mu}\mathbf{A}^=$ has certain useful properties; i.e., some of the coefficients of $\bar{\boldsymbol{\gamma}}$ are equal to the corresponding coefficients of the given vector $\boldsymbol{\pi}$. Then using known properties of the points \mathbf{x} in P satisfying $\boldsymbol{\pi}\mathbf{x} = \boldsymbol{\pi}_0$, one determines the still unknown coefficients of $\bar{\boldsymbol{\gamma}}$ iteratively. If it turns out that $\bar{\boldsymbol{\gamma}} = \lambda\boldsymbol{\pi} + \boldsymbol{\mu}\mathbf{A}^=$ for some $\lambda \geq 0$ and $\boldsymbol{\mu} \in R^{|\mathbf{M}^=|}$, then Condition 4 implies that $\boldsymbol{\pi}\mathbf{x} \leq \boldsymbol{\pi}_0$ defines a facet of P .

Most of the research deriving from polyhedral theory has centered on finding strong valid inequalities for mathematical formulations associated with the generalized assignment problem, various versions of the TSP, vehicle routing problems, set covering problems, and production planning problems. In addition, many valid inequalities exist for knapsack- and variable upper bound network flow-type constraints. Once these cuts are identified, they are typically embedded in a branch and cut enumeration scheme. In this methodology, a search tree is created, but unlike traditional branch and bound, cuts are added to the model at each node of the tree. Issues related to implementation are discussed in [G14, H5, K7, P1]. To date, there have been no applications of cutting plane techniques to bilevel programs.

3.4 BENDERS DECOMPOSITION FOR MIXED-INTEGER LINEAR PROGRAMMING

Enumeration methods discussed in Section 3.2 can be extended directly to solve mixed-integer linear programs of the form (3.2). It is also straightforward to extend Gomory's fractional cuts to the MILP but not the all-integer primal cuts. With regard to polyhedral theory, the most success has been achieved on combinatorial optimization problems with regular structure rather than general MILPs.

An alternative approach to solving MILPs is to use duality theory to rewrite (3.2) as a pure integer program (3.1). The advantage of doing this occurs when the number of integer variables in the MILP is small. However, in the equivalent formulation it is often computationally impossible to obtain all the constraints of the integer program, since there is one constraint for each extreme point and extreme ray of the dual polyhedron. There is usually a vast number of such extreme points and extreme rays. To overcome this difficulty, J.F. Benders proposed a decomposition technique in which the equivalent IP is solved after generating only a subset of its constraints. The Benders procedure partitions the MILP into an integer and a linear program, consisting respectively of the integer and continuous variables of the original problem. The algorithm works by successively solving an LP and an IP. The LP produces an extreme point or extreme ray and one or two constraints for the IP at each iteration. Also, the value of the LP gives an upper bound on the optimal solution of the MILP. When solved, the IP yields a nondecreasing lower bound. When the two bounds coincide, the optimal MILP solution has been found and the process terminates. Before giving the details, we note that Benders' procedure is applicable to any mathematical program in which the variables can be partitioned into two subsets so that when the variables in one are assigned values the remaining problem reduces to a linear program.

3.4.1 Reformulation of MILP

Consider the mixed-integer linear program (3.2) rewritten with greater than or equal to inequalities:

$$\begin{array}{ll} \min & \mathbf{c}\mathbf{x} + \mathbf{d}\mathbf{y} \\ \text{subject to} & \mathbf{Ax} + \mathbf{By} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \text{ and integer} \\ & \mathbf{y} \geq \mathbf{0} \end{array} \quad (3.42)$$

If we let $X \subseteq \mathbb{Z}_+^n$ denote the set of all feasible nonnegative integer vectors \mathbf{x} , then (3.42) may be written as

$$\min_{\mathbf{x} \in X} \left(\mathbf{c}\mathbf{x} + \min (\mathbf{d}\mathbf{y} : \mathbf{By} \geq \mathbf{b} - \mathbf{Ax}, \mathbf{y} \geq \mathbf{0}) \right) \quad (3.43)$$

For a fixed \mathbf{x} , the inner minimization problem is the linear program

$$\begin{aligned} \min \quad & d\mathbf{y} \\ \text{subject to} \quad & \mathbf{B}\mathbf{y} \geq \mathbf{b} - \mathbf{A}\mathbf{x} \\ & \mathbf{y} \geq 0 \end{aligned} \tag{3.44}$$

and its dual is

$$\begin{aligned} \max \quad & \mathbf{u}(\mathbf{b} - \mathbf{A}\mathbf{x}) \\ \text{subject to} \quad & \mathbf{u}\mathbf{B} \leq \mathbf{d} \\ & \mathbf{u} \geq 0 \end{aligned} \tag{3.45}$$

If the set $U = \{\mathbf{u} : \mathbf{u}\mathbf{B} \leq \mathbf{d}, \mathbf{u} \geq 0\}$ is empty, the dual problem (3.45) is infeasible, and from duality theory the primal problem (3.44) is either unbounded or infeasible. In the former case, the original MILP (3.42) is unbounded; however, in any real-world situation the mixed-integer problem is bounded or can be made so by placing sufficiently large upper bounds on the decision variables. Therefore, we shall assume that the set U is nonempty. Also, observe that the polyhedron U is independent of \mathbf{x} and so no matter what the value of \mathbf{x} , the maximum over U of $\mathbf{u}(\mathbf{b} - \mathbf{A}\mathbf{x})$ occurs at an extreme point of U or it grows without bound along one of its extreme rays. But U has a finite number of extreme points and extreme rays. Denote the extreme points by \mathbf{u}^p , ($p = 1, \dots, n_p$), and the direction of the extreme rays (which can be found by enumerating all extreme rays of $\mathbf{u}\mathbf{B} \leq \mathbf{d}, \mathbf{u} \geq 0$) by \mathbf{v}^r , $r = 1, \dots, n_r$. Now, if for some \mathbf{x} there exists a \mathbf{v}^r such that $\mathbf{v}^r(\mathbf{b} - \mathbf{A}\mathbf{x}) > 0$, then the maximum of $\mathbf{u}(\mathbf{b} - \mathbf{A}\mathbf{x})$ for $\mathbf{u} \in U$ is unbounded. On the other hand, if the maximum of $\mathbf{u}(\mathbf{b} - \mathbf{A}\mathbf{x})$ for $\mathbf{u} \in U$ is unbounded, then for this \mathbf{x} there is a direction \mathbf{v}^r such that $\mathbf{v}^r(\mathbf{b} - \mathbf{A}\mathbf{x}) > 0$. When problem (3.45) is unbounded, duality theory tells us that problem (3.44) has no feasible solution and thus the MILP (3.42) has no solution for that \mathbf{x} . As a consequence

$$\mathbf{v}^r(\mathbf{b} - \mathbf{A}\mathbf{x}) \leq 0, \quad r = 1, \dots, n_r$$

provides necessary and sufficient conditions on \mathbf{x} for feasible solutions \mathbf{y} to exist for the MILP.

The set \mathbf{x} of feasible vectors can be defined as

$$\mathbf{v}^r(\mathbf{b} - \mathbf{A}\mathbf{x}) \leq 0, \quad (r = 1, \dots, n_r), \quad \mathbf{x} \geq 0 \text{ and integer}$$

Also, for any $\mathbf{x} \in X$, the inner minimization problem in (3.43) is equal to the maximum of the dual problem (3.45). Thus the MILP is

$$\begin{aligned} \min_{\mathbf{x}} \quad & (\mathbf{c}\mathbf{x} + \max_{p=1, \dots, n_p} \mathbf{u}^p(\mathbf{b} - \mathbf{A}\mathbf{x})) \\ \text{subject to} \quad & \mathbf{v}^r(\mathbf{b} - \mathbf{A}\mathbf{x}) \leq 0, \quad r = 1, \dots, n_r \\ & \mathbf{x} \geq 0 \text{ and integer} \end{aligned}$$

Letting

$$z = \mathbf{c}\mathbf{x} + \max_{p=1, \dots, n_p} \mathbf{u}^p(\mathbf{b} - \mathbf{A}\mathbf{x})$$

we have $z \geq \mathbf{c}\mathbf{x} + \mathbf{u}^p(\mathbf{b} - \mathbf{A}\mathbf{x})$ for $p = 1, \dots, n_p$, and the MILP is equivalent to the integer program (really a mixed-integer linear program with one unrestricted continuous variable z)

$$\begin{array}{ll} \min_{\mathbf{x}, z} & z \\ \text{subject to} & z \geq \mathbf{c}\mathbf{x} + \mathbf{u}^p(\mathbf{b} - \mathbf{A}\mathbf{x}) \quad p = 1, \dots, n_p \\ & \mathbf{0} \geq \mathbf{v}^r(\mathbf{b} - \mathbf{A}\mathbf{x}) \quad r = 1, \dots, n_r \\ & \mathbf{x} \geq \mathbf{0} \text{ and integer} \end{array} \quad (3.46)$$

To clarify the discussion, we transform a small MILP into the IP given in (3.46).

Example 3.4.1 The mixed-integer linear program is

$$\begin{array}{ll} \min & x + y_1 + y_3 \\ \text{subject to} & 2x + y_1 - 6y_2 - 5y_3 \geq 1 \\ & 3x - y_1 + y_2 + 2y_3 \geq 2 \\ & x \geq 0 \text{ and integer; } y_1, y_2, y_3 \geq 0 \end{array}$$

The linear program (3.44) is

$$\begin{array}{ll} \min & y_1 + y_3 \\ \text{subject to} & y_1 - 6y_2 - 5y_3 \geq 1 - 2x \\ & -y_1 + y_2 + 2y_3 \geq 2 - 3x \\ & y_1, y_2, y_3 \geq 0 \end{array}$$

The corresponding dual (3.45) is

$$\begin{array}{ll} \max & (1 - 2x)u_1 + (2 - 3x)u_2 \\ \text{subject to} & u_1 - u_2 \leq 1 \\ & -6u_1 + u_2 \leq 0 \\ & -5u_1 + 2u_2 \leq 1 \\ & u_1, u_2 \geq 0 \end{array}$$

The set U has three extreme points $\mathbf{u}^1 = (1, 0)$, $\mathbf{u}^2 = (0, 0)$, and $\mathbf{u}^3 = (1/7, 6/7)$, and two extreme rays with directions $\mathbf{v}^1 = (1, 1)$ and $\mathbf{v}^2 = (2/5, 1)$. Thus the MILP is equivalent to the IP

$$\begin{aligned}
 & \min_{x,z} && z \\
 \text{subject to} & && z \geq x + (1 - 2x) = 1 - x \\
 & && z \geq x \\
 & && z \geq x + \frac{1}{7}(1 - 2x) + \frac{6}{7}(2 - 3x) = \frac{13}{7} - \frac{20}{7}x \\
 & && 0 \geq (1 - 2x) + (2 - 3x) = 3 - 5x \\
 & && 0 \geq \frac{2}{5}(1 - 2x) + 1(2 - 3x) = \frac{12}{5} - \frac{19}{5}x \\
 & && x \geq 0 \text{ and integer}
 \end{aligned}$$

The optimal solution is $x^* = 1$ and $z^* = 1$. To find the optimal values of \mathbf{y} , the linear program is solved with $x = 1$. By inspection, since the right-hand side of the LP becomes $1 - 2(1) = -1$ and $2 - 3(1) = -1$, any vector $\mathbf{y} = (0, y_2, 0)$ with $0 \leq y_2 \leq 1/6$ is optimal. Note that for $x = 1$, $u_1(1 - 2x) + u_2(2 - 3x) = -u_1 - u_2$, and thus the optimal solution to the dual LP is $u_1 = u_2 = 0$ with the objective function $-u_1 - u_2 = 0$.

3.4.2 Algorithm

The derivation of the IP (3.46) suggests a procedure for solving the MILP (3.42). Observe that for a fixed nonnegative integer vector \mathbf{x} we can solve the dual LP (3.45). As this problem is more restricted than the mixed-integer linear program, the value of its maximal solution when added to $c\mathbf{x}$ is an upper bound on the optimal solution to (3.42). Furthermore, by solving (3.45) we obtain an extreme point of U or the direction of one of its extreme rays, and thus a constraint for the IP (3.46). Now suppose the IP is solved with this one constraint. Its optimal solution is a lower bound on the best solution to the MILP. Also, it yields a nonnegative integer vector \mathbf{x} for the dual problem (3.45), which, when solved produces a new extreme point or ray direction and thus a second inequality for the IP. The IP, now with two constraints is solved to yield a new lower bound no worse than the first, and a second nonnegative integer vector \mathbf{x} . With the new value of \mathbf{x} , the dual LP can be solved, and so forth.

The process terminates when the lower bound equals the upper bound. To find the optimal \mathbf{y} vector, problem (3.44) is solved with \mathbf{x} equal to its optimal value. An iterative solution technique incorporating the above ideas follows.

Step 1 (Initialization) Let $P(1)$ and $R(1)$ be any known subset of extreme points and extreme rays of U , respectively. Let $k = 1$, where k counts the iterations. If $P(1) = R(1) = \emptyset$, let z^1 be arbitrarily small, \mathbf{x}^1 be any nonnegative integer vector, and go to Step 3; otherwise go to Step 2.

Step 2 (Solving the IP) Solve the IP

$$\begin{aligned} z^k &= \min_{\mathbf{x}, z} z \\ &\quad z \geq \mathbf{c}\mathbf{x} + \mathbf{u}^p(\mathbf{b} - \mathbf{A}\mathbf{x}) \quad p \in P(k) \\ &\quad 0 \geq \mathbf{v}^r(\mathbf{b} - \mathbf{A}\mathbf{x}) \quad r \in R(k) \\ &\quad \mathbf{x} \geq 0 \text{ and integer} \end{aligned} \tag{3.47}$$

- (a) If no feasible solution exists to (3.47), then there is no feasible solution to (3.46), since the constraints of the former are included in those of the latter. Therefore, (3.42) has no solution.
- (b) If (3.47) has an optimal solution, call it (z^k, \mathbf{x}^k) . Go to Step 3.
- (c) If (3.47) is unbounded, let z^k be arbitrarily small and \mathbf{x}^k be any feasible solution to (3.47) with $z = z^k$. Go to Step 3.

In cases (b) and (c), $z^* \geq z^k \geq z^{k-1}$ since (3.47) at iteration $k-1$ is a relaxation of (3.47) at iteration k , and since (3.47) at any iteration is a relaxation of the original problem (3.42).

Step 3 (Solving the dual LP) Solve the LP

$$\begin{aligned} z^*(\mathbf{x}^k) &= \mathbf{c}\mathbf{x}^k + \max_{\mathbf{u}} \mathbf{u}(\mathbf{b} - \mathbf{A}\mathbf{x}^k) \\ &\quad \mathbf{u}\mathbf{B} \leqq \mathbf{d} \\ &\quad \mathbf{u} \geqq \mathbf{0} \end{aligned} \tag{3.48}$$

- (a) If $z^*(\mathbf{x}^k) \rightarrow \infty$ along an extreme ray $\mathbf{u}^k + \theta\mathbf{v}^k$ (for $\theta \rightarrow \infty$), let $P(k+1) = P(k) \cup \{\mathbf{u}^k\}$ and $R(k+1) = R(k) \cup \{\mathbf{v}^k\}$. Put $k \leftarrow k+1$ and go to Step 2.
- (b) If (3.48) has an optimal solution, call it \mathbf{u}^k , and let the associated solution to (3.44) be \mathbf{y}^k . Using the fact that for some $\mathbf{x}' \in X$ the solutions to (3.44) and (3.45) are equal and provide an upper bound on z^* when $\mathbf{c}\mathbf{x}'$ is added to their objective values, we have $z^*(\mathbf{x}^k) \geq z^* \geq z^k$. It follows that if $z^*(\mathbf{x}^k) = z^k$, an optimal solution to (3.42) is given by $(\mathbf{x}^k, \mathbf{y}^k)$. If $z^*(\mathbf{x}^k) > z^k$, let $P(k+1) = P(k) \cup \{\mathbf{u}^k\}$, put $k \leftarrow k+1$, and go to Step 2.

Because an additional extreme point or extreme ray is generated at each iteration, the number of iterations of the algorithm is bounded from above by $|P| + |R|$. To see this consider the possible results of Step 3 at iteration k :

- (1) $z^*(\mathbf{x}^k) \rightarrow \infty$, which implies that $\mathbf{v}^k(\mathbf{b} - \mathbf{A}\mathbf{x}^k) > 0$. It follows that $\mathbf{v}^k \notin R(k)$ since \mathbf{x}^k is feasible to (3.47). Thus $|R(k+1)| = |R(k)| + 1$.
- (2) $\infty > z^*(\mathbf{x}^k) > z^k$. In this case it follows that $\mathbf{u}^k \notin P(k)$ since $\mathbf{c}\mathbf{x}^k + \mathbf{u}^k(\mathbf{b} - \mathbf{A}\mathbf{x}^k) = z^*(\mathbf{x}^k) > z^k \geq \mathbf{c}\mathbf{x}^k + \mathbf{u}^p(\mathbf{b} - \mathbf{A}\mathbf{x}^k)$ for every $p \in P(k)$. Thus $|P(k+1)| = |P(k)| + 1$.

(3) $\infty > z^*(\mathbf{x}^k) = z^k$ implies termination.

Example 3.4.1 (continued)

Step 1 We initialize the sets $P(1) = R(1) = \emptyset$, set $z^1 = -\infty$, $x^1 = 0$, and go to Step 3.

Step 3 The dual LP with $x^1 = 0$ is

$$\begin{array}{ll}\max & u_1 + 2u_2 \\ \text{subject to} & u_1 - u_2 \leq 1 \\ & -6u_1 + u_2 \leq 0 \\ & -5u_1 + 2u_2 \leq 1 \\ & u_1, u_2 \geq 0\end{array}$$

The solution is unbounded. Suppose the simplex computations yield an extreme ray at the point $\mathbf{u}^1 = (1, 0)$ with direction $\mathbf{v}^1 = (1, 1)$. Accordingly, we put $P(2) = \{(1, 0)\}, R(2) = \{(1, 1)\}$ and go to Step 2.

Step 2 The new constraints are computed as follows:

$$\begin{aligned}z &\geq 1x + 1(1 - 2x) + 0(2 - 3x) = 1 - x \\0 &\geq 1(1 - 2x) + 1(2 - 3x) = 3 - 5x\end{aligned}$$

The IP is

$$\begin{array}{l}\min_{x,z} z \\ z \geq 1 - x \\ 0 \geq 3 - 5x \\ x \geq 0 \text{ and integer}\end{array}$$

For this problem the minimal value of z can be arbitrarily small. This corresponds to case (c) so we set x^2 to any positive integer M and go to Step 3.

Step 3 Solve the dual LP

$$\begin{aligned}z^*(M) &= 1M + \max u_1(1 - 2M) + u_2(2 - 3M) \\&\text{subject to } (u_1, u_2) \in U\end{aligned}$$

The optimal solution is $u_1^2 = 0, u_2^2 = 0$ and $z^*(M) = M$. We set $P(2) = \{(1, 0), (0, 0)\}$, $k = 3$ and go to Step 2.

Step 2 The new constraint is computed as follows: $x + 0(1 - 2x) + 0(2 - 3x) = x \leq z$.
The IP is

$$\begin{aligned} \min_{x,z} \quad & z \\ \text{s.t.} \quad & z \geq 1 - x \\ & z \geq x \\ & 0 \geq 3 - 5x \\ & x \geq 0 \text{ and integer} \end{aligned}$$

which has optimal solution $x^3 = 1$ and $z^3 = 1$.

Step 3 The dual LP with $x = 1$ is

$$\begin{aligned} z^*(1) = 1 + \max_{(u_1, u_2)} \quad & -u_1 - u_2 \\ \text{subject to} \quad & (u_1, u_2) \in U \end{aligned}$$

which has optimal solution $u_1^3 = 0$, $u_2^3 = 0$ and $z^*(1) = 1$. Thus $z^*(x^3 = 1) = z^2 = 1$ so we terminate. To find $\mathbf{y} = (y_1, y_2, y_3)$, problem (3.44) is solved with $x = 1$. This gives $y_1 = y_3 = 0$ and any y_2 such that $0 \leq y_2 \leq 1/6$.

3.5 UNIMODULARITY

In this section we define a class of integer programs for which the linear programming optimal solution always satisfies the integrality constraints. Such problems can therefore be solved using the simplex method although many specialized algorithms exist. Let us begin by considering an LP with constraints

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0} \tag{3.49}$$

where \mathbf{A} and \mathbf{b} are assumed to be integer matrices. As in Chapter 2, we partition \mathbf{A} as (\mathbf{B}, \mathbf{N}) , such that \mathbf{B} is nonsingular, and write the equations of (3.49) as $\mathbf{B}\mathbf{x}_B + \mathbf{N}\mathbf{x}_N = \mathbf{b}$. Recall that a basic solution to (3.49) is $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$, $\mathbf{x}_N = \mathbf{0}$. Thus a sufficient condition for a basic solution to be integer is for \mathbf{B}^{-1} to be an integer matrix. To derive necessary conditions we introduce the notion of unimodularity.

Definition 3.5.1 A square integer matrix \mathbf{B} is called *unimodular* if the absolute value of its determinant equals 1; i.e., $|\det \mathbf{B}| = 1$. An integer $m \times n$ matrix \mathbf{A} is *totally unimodular* if every square nonsingular submatrix is unimodular; i.e., the determinant of every square submatrix is either 0, 1 or -1 .

From linear algebra we know that if \mathbf{B} is nonsingular, then $\mathbf{B}^{-1} = \mathbf{B}^+ / |\det \mathbf{B}|$, where \mathbf{B}^+ is the adjoint of \mathbf{B} and is similarly an integer matrix. This implies:

Theorem 3.5.1 If \mathbf{A} is totally unimodular, then every basic solution $(\mathbf{x}_B, \mathbf{x}_N) = (\mathbf{B}^{-1}\mathbf{b}, \mathbf{0})$ of (3.49) is integer.

The example below shows that total unimodularity of \mathbf{A} is not necessary for integer basic solutions even when $\mathbf{b} \in \mathbb{R}^m$ is integer. Consider

$$\begin{aligned} 5x_1 + 2x_2 &= b_1 \\ 2x_1 + x_2 &= b_2 \\ b_1, b_2 &\text{ integer} \end{aligned}$$

Although $\mathbf{A} = \begin{bmatrix} 5 & 2 \\ 2 & 1 \end{bmatrix}$ is not totally unimodular, the unique solution to these equations is integer.

For LPs with inequality constraints, Theorem 3.5.1 can be strengthened. In particular, given the constraint set $S(\mathbf{b}) = \{\mathbf{x} : \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$, total unimodularity of \mathbf{A} is necessary and sufficient for all extreme points of $S(\mathbf{b})$ to be integer for every vector \mathbf{b} . This result is the core of next theorem whose proof can be found in [N3].

Theorem 3.5.2 Let \mathbf{A} be an integer matrix. The following statements are equivalent:

1. \mathbf{A} is totally unimodular.
2. The extreme points (if any) of $S(\mathbf{b}) = \{\mathbf{x} : \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ are integer for arbitrary integer \mathbf{b} .
3. Every square nonsingular submatrix of \mathbf{A} has an integer inverse.

Example 3.5.1 (Assignment Problem) We wish to assign three jobs to three machines so that each job i ($i = 1, 2, 3$) is assigned to exactly one machine j ($j = 1, 2, 3$). If c_{ij} is the processing cost of job i on machine j , then the problem is to make the assignments so as to minimize total cost. Let x_{ij} be a binary variable equal to 1 if job i is assigned to job j , and 0 otherwise. The corresponding mathematical formulation is

$$\begin{aligned} \text{minimize } & \sum_{i=1}^3 \sum_{j=1}^3 c_{ij} x_{ij} \\ \text{subject to } & \sum_{i=1}^3 x_{ij} = 1, \quad j = 1, 2, 3 \\ & \sum_{j=1}^3 x_{ij} = 1, \quad i = 1, 2, 3 \\ & x_{ij} = 0 \text{ or } 1, \quad \forall i, j \end{aligned}$$

For this problem, the coefficient matrix \mathbf{A} can be written as

$$\begin{array}{c} \text{Machine constraints} \\ \left\{ \begin{array}{c} \begin{array}{ccccccccc} x_{11} & x_{12} & x_{13} & x_{21} & x_{22} & x_{23} & x_{31} & x_{32} & x_{33} \end{array} \\ \left[\begin{array}{ccccccccc} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{array} \right] \end{array} \right. \\ \text{Job constraints} \\ \left\{ \begin{array}{c} \begin{array}{ccccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array} \end{array} \right. \end{array}$$

Note that the \mathbf{A} matrix has the following properties:

- (a) Each element is either 0 or 1, hence the determinant of every square submatrix of size 1 is either 0 or 1.
- (b) Every column has exactly two 1's, the first corresponding to a machine constraint and the second to a job constraint.
- (c) The sum of the rows corresponding to the machine constraints is the same as the sum of the job constraints (a vector of all 1's). Hence, the rows of \mathbf{A} are not linearly independent so $\text{rank}(\mathbf{A}) < 6$, and the determinant of every square matrix of size 6 is 0.

Using these observations we can prove by induction that the matrix \mathbf{A} is unimodular. In particular, suppose that the determinant of all submatrices of \mathbf{A} of size up to $m-1$ is either 0, +1, or -1 and let \mathbf{B} be a square submatrix of size m . There are three cases.

- (1) If \mathbf{B} has a column of 0's, $|\det \mathbf{B}| = 0$.
- (2) If \mathbf{B} has a column containing only a single 1, expand the determinant of \mathbf{B} by the cofactors of that column. Thus $|\det \mathbf{B}| = |\det \mathbf{B}'|$, where \mathbf{B}' is the submatrix of \mathbf{B} obtained by deleting a column which has a single 1 and the corresponding row. By hypothesis $|\det \mathbf{B}'| = 0, 1, \text{ or } -1$; hence $|\det \mathbf{B}| = 0, 1, \text{ or } -1$ as well.
- (3) If every column of \mathbf{B} has two 1's, then the first 1 in each column is from the machine constraints and the second 1 is the job constraints. As in observation (c) above, it follows that the rows of \mathbf{B} are dependent and thus $|\det \mathbf{B}| = 0$.

From this analysis, it should be evident that a necessary condition for \mathbf{A} to be totally unimodular is $a_{ij} = 0, 1, -1$ for all i and j . Nevertheless, it is not easy in general to tell whether a matrix consisting only of $(0, +1, -1)$ is totally unimodular. The next theorem gives a sufficient condition that is extremely useful for ILPs associated with graphs.

Theorem 3.5.3 An integer matrix \mathbf{A} with $a_{ij} = 0, 1, -1$ for all i and j is totally unimodular if

1. No more than two nonzero elements appear in each column.
2. The rows can be partitioned into two subsets Q_1 and Q_2 such that
 - (a) if a column contains two nonzero elements with the same sign, one element is in each of the subsets;
 - (b) if a column contains two nonzero elements of opposite sign, both elements are in the same subset.

Note that an incidence matrix of a directed graph satisfies the conditions of Theorem 3.5.3. Every column has one (+1) and one (-1). Thus Condition 1 is satisfied and Condition 2 can be fulfilled with Q_1 the index set of the rows and $Q_2 = \emptyset$.

The incidence matrix \mathbf{A} in Example 3.5.1 corresponds to Condition 2(a) so we would set $Q_1 = \{1, 2, 3\}$ and $Q_2 = \{4, 5, 6\}$. The assignment problem, which gave rise to this incidence matrix, belongs to a class of LPs associated with graphs that have \mathbf{A} matrices satisfying the conditions of Theorem 3.5.3. The classical transportation problem, the transshipment problem, the shortest path problem, the minimum cost network flow problem, and the minimum spanning tree problem are the most well-known members of this class. More detail is given in [A1]. To date, there have been no applications of bilevel programming that have included pure network-type problems in the overall model.

NONLINEAR PROGRAMMING

4.1 INTRODUCTION

The central concern of this chapter is with the study of theory and algorithms for mathematical programs of the following general form:

$$\begin{aligned}
 & \min f(\boldsymbol{x}) \\
 \text{subject to } & h_i(\boldsymbol{x}) = 0, \quad i = 1, \dots, m \\
 & g_i(\boldsymbol{x}) \geq 0, \quad i = 1, \dots, q \\
 & \boldsymbol{x} \in X
 \end{aligned} \tag{4.1}$$

where $f, h_1, \dots, h_m, g_1, \dots, g_q$ are functions whose range is defined on R^1 , X is a subset of R^n that places additional restrictions on the variables such as bounds, and \boldsymbol{x} is a vector of n components x_1, \dots, x_n . A value of \boldsymbol{x} satisfying the constraints of (4.1) is called a *feasible* solution to the problem. The collection of all such solutions forms the feasible region. The nonlinear programming problem (NLP) then, is to find a feasible point $\bar{\boldsymbol{x}}$ such that $f(\bar{\boldsymbol{x}}) \leq f(\boldsymbol{x})$ for each feasible point \boldsymbol{x} . Such a point is called the *optimal* solution, or simply, a solution to (4.1). Needless to say, an NLP can be stated as a maximization problem, and that some or all of the inequality constraints in (4.1) can be written as $g_i(\boldsymbol{x}) \leq 0$ for $i = 1, \dots, q$. In the special case when the objective function in (4.1) is linear and all the constraints, including the set X , can be represented by linear inequalities or equations, the above problem reduces to a linear program as discussed in Chapter 2.

Example 4.1.1 Basic concepts are illustrated in the following problem for $\boldsymbol{x} \in R^2$:

$$\begin{aligned}
 & \min \quad f(x_1, x_2) = (x_1 - 3)^2 + (x_2 - 2)^2 \\
 \text{subject to } & g_1(x_1, x_2) = x_1^2 - x_2 - 3 \leq 0 \\
 & g_2(x_1, x_2) = x_2 - 1 \leq 0 \\
 & g_3(x_1, x_2) = x_1 \geq 0
 \end{aligned}$$

The feasible region is depicted in Fig. 4.1. The problem is to find the point in the feasible region with the smallest possible $f(x_1, x_2)$ value. Note that the points (x_1, x_2) satisfying $(x_1 - 3)^2 + (x_2 - 2)^2 = c$ represent a circle with radius \sqrt{c} and center at $(3, 2)$. This circle is called an *isovalue contour* of the objective function having value c . Because we wish to minimize c , we must find the circle with the smallest radius that intersects the feasible region. As shown in Fig. 4.1, the smallest such circle has $c = 2$ and intersects the feasible region at $(2, 1)$ which is the optimal solution. The corresponding objective function value $f(2, 1) = 2$.

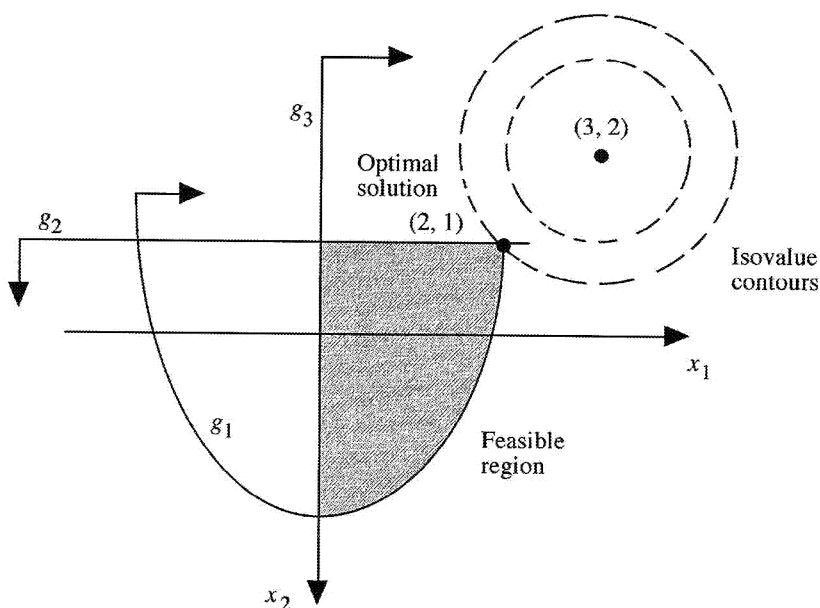


Figure 4.1 Geometric solution for Example 4.1.1

Example 4.1.2

$$\begin{aligned} \min \quad & f(x_1, x_2) = |x_1 - 2| + |x_2 - 2| \\ \text{subject to} \quad & h_1(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0 \\ & g_1(x_1, x_2) = x_1 - x_2^2 \geq 0 \end{aligned}$$

The dashed lines in Fig. 4.2 represent isovalue contours of the objective function. The feasible region is the arc of the circle lying within the parabola. A solution to the problem is any point in the feasible region with smallest objective function isovalue. This is seen by inspection to be $(\sqrt{2}/2, \sqrt{2}/2)$. If the equality constraint is removed

the solution becomes $(2, \sqrt{2}/2)$. If both constraints are removed the solution is at $(2, 2)$ which is termed an unconstrained minimum.

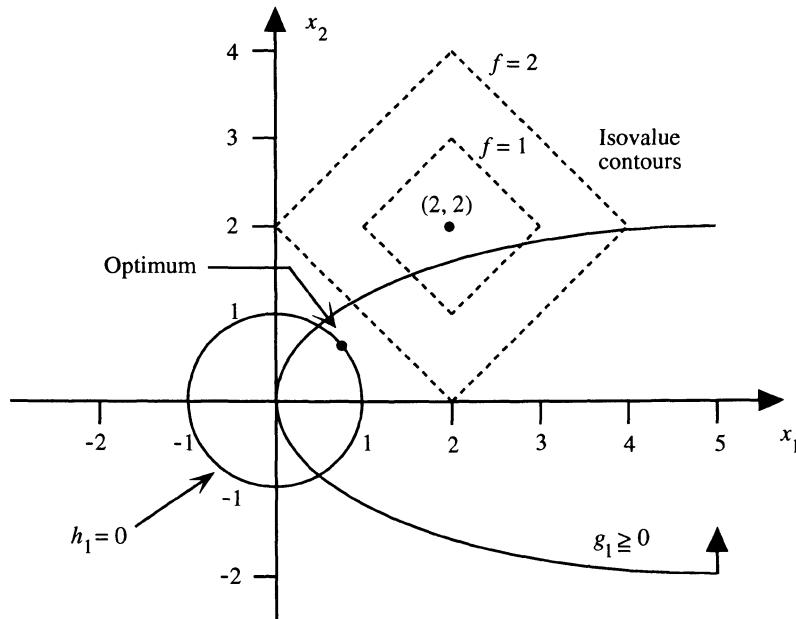


Figure 4.2 Geometric solution for Example 4.1.2

Notice that all of the functions in Examples 4.1.1 and 4.1.2 are differentiable accept the second objection function. Usually, we require that each function in problem (4.1) be continuous. Much of the theory of nonlinear programming concerns the case when the functions are continuously differentiable, or twice continuously differentiable. In these instances it is possible to prove theorems that characterize solutions to (4.1) and, in turn, influence the development of solution algorithms.

Within the last 3 decades a wide variety of mathematical programming problems have been dealt with successfully. As might be expected, the linear model has received the most attention with great strides being made for large sparse problems and network formulations. Impressive results have also been obtained for the quadratic programming problem (QP), where $f(\mathbf{x})$ assumes a positive semidefinite quadratic form, and the constraints are linear. Most commercial interior point codes for solving LPs now contain a QP solver as well.

When $f(\mathbf{x})$ is a convex, separable function and the constraints are linear, a number of unique algorithms have been developed. Special-purpose algorithms also exist for the slightly more general case where $f(\mathbf{x})$ is convex and the constraints are linear.

NLPs characterized by $f(\mathbf{x})$ and $-g_i(\mathbf{x})$ convex, and $h_i(\mathbf{x})$ linear have received particular attention. When these conditions prevail (4.1) is called a *convex program*. The smoothness of the functions makes the problem well behaved, and the convexity assumption assures that the feasible region is convex and, most importantly, that any local solution (relative minimum) is also a global solution.

When the functions in an NLP assume an arbitrary form, the developments mentioned above are often applicable only in a *local* sense; that is, they hold if \mathbf{x} is restricted to a suitable neighborhood such that the requisite conditions hold in that neighborhood. Fortunately, some results can be easily extended to characterize local solutions of (4.1) when the desired convexity is absent. The general nonconvex problem, where convexity may not even exist in the neighborhood of a relative minimum, has remained rather intractable. Most of the results in this area have been of theoretical interest only and have yet to be translated into efficient algorithms. For an indepth treatment of the issues surrounding global optimization, the reader is referred to [H10].

4.1.1 Classification of Problems

The vast majority of optimization problems can be expressed in the form of (4.1). Even those that do not fit into this framework can often be posed as a sequence of standard problems. However, the existence of a standard, very general representation does not imply that all distinctions among problems should be ignored. When faced with any problem, it is usually advantageous to determine special characteristics that allow it to be solved more efficiently. For example, it may be possible to omit tests for situations that cannot occur, or to avoid recomputing quantities that are invariant.

The most extreme form of classification would be to assign every problem to a separate category. But such an approach is based on the false premise that every difference is significant with respect to obtaining solutions. It is unlikely, for instance, that a small change in the number of variables in a model would dictate the algorithmic scheme. Although no set of categories is ideal for every circumstance, a reasonable classification scheme can be developed based on balancing the improvements in efficiency that are possible by taking advantage of special properties against the additional complexity of providing a larger selection of methods and software.

The most obvious distinctions in problems involve variations in the mathematical characteristics of the objective functions and constraints. For example, the objective function may be very smooth in some cases, and discontinuous in others; the problem functions may be of a simple form with well understood properties, or their computations may require the solution of several complicated sub-problems. The following table from [G7] gives a typical classification scheme based on the nature of the problem functions, where significant algorithmic advantage can be taken of each characteristic. For example, a particular problem might be categorized as the mini-

mization of a smooth nonlinear function subject to upper and lower bounds on the variables.

Properties of $f(\mathbf{x})$	Properties of $h_i(\mathbf{x}), g_i(\mathbf{x})$
Function of a single variable	No constraints
Linear function	Simple bounds
Sum of squares of linear functions	Linear functions
Quadratic function	Sparse linear functions
Sum of squares of nonlinear functions	Smooth nonlinear functions
Smooth nonlinear function	Sparse nonlinear functions
Sparse nonlinear function	Nonsmooth nonlinear functions
Nonsmooth nonlinear function	

Other features may also be used to distinguish among optimization problems. The size of a problem affects both the storage and the amount of computational effort required to obtain the solution, and hence techniques that are effective for a problem with a few variables are usually unsuitable when there are hundreds of thousands of variables. However, the definition of *size* is by no means absolute, and must situations are environment-dependent. A method that strains a microcomputer may run comfortably on a workstation; a user faced with a daily scheduling problem has a different perspective on what constitutes computing time than someone investigating capital expansion strategies over the long term.

Another way in which problems vary involves the computable information that may be available to an algorithm during the solution process. For example, in one instance it may be possible to compute analytic first and second derivatives of the objective function, while in another case only the function values may be provided. A future refinement of such a classification would reflect the computational burden associated with obtaining the information. The best strategy for a problem clearly depends on the relative effort required to compute the function value compared to the operations associated with the solution method.

Finally, applications of optimization may include special needs that reflect the source of the problem and the framework within which it is to be solved. Such external factors often dictate requirements in solving the problem that are not contained in its mathematical statement. In some problems it may be highly desirable for certain constraints to be satisfied exactly at every iteration. The required accuracy also differs with the application; for example, it might be wasteful to solve an optimization problem with maximum accuracy when the results are used only in a minor way within some outer iteration.

4.1.2 Difficulties Resulting from Nonlinearities

Before discussing any particular class of nonlinear programs, we shall examine some of the characteristics associated with nonlinear forms that can make it much more difficult to solve NLPs than LPs. Recall that linear programs have the following properties.

- (a) The set of feasible solutions (i.e., all n -tuples (x_1, \dots, x_n) that satisfy (2.1b,c)) is convex.
- (b) The set of all n -tuples (x_1, \dots, x_n) which yields a specified value of the objective function is a hyperplane.
- (c) A local extremum is also a global extremum over the set of feasible solutions.
- (d) If the optimal value of the objective function is finite, at least one of the extreme points of the convex set of feasible solutions will be an optimum. Furthermore, starting at any extreme point in the feasible region, it is possible to reach an optimal extreme point in a series of steps such that at each step one moves only to an adjacent extreme point.

When nonlinearities are present in the formulation, none of these properties may hold. Consider the following problem with linear constraints and a separable objective function:

$$\begin{aligned} \max f(x_1, x_2) &= 25(x_1 - 2)^2 + (x_2 - 2)^2 \\ \text{subject to} \quad &x_1 + x_2 \geq 2 \\ &-x_1 + x_2 \leq 2 \\ &x_1 + x_2 \leq 6 \\ &x_1 - 3x_2 \leq 2 \\ &x_1, x_2 \leq 0 \end{aligned}$$

The graphical representation of the feasible region is shown in Fig. 4.3 along with several isovalue contours. Note that extreme point $(5, 1)$ yields the global maximum, and that extreme point $(0, 2)$ yields a relative maximum different from the global. In addition, the value of the objective function at $(0, 2)$ is greater than the value at either of the adjacent extreme points $(2, 0)$ and $(2, 4)$. Thus we have the case where the global maximum occurs at an extreme point of the feasible region and that local solutions exist at other extreme points. This situation is common when we are maximizing a convex function over a polyhedral constraint region. If we were minimizing the same objective function over the same set of linear constraints, the global optimum would occur at $(2, 2)$ which is in the interior of the feasible region.

When a programming problem contains nonlinear constraints, it need no longer be true that the feasible region is convex. In fact, the set of solutions may consist of several disconnected parts. Suppose, for example, that the constraints of a particular problem are

$$(x_1 - 1)x_2 \leq 1, \quad x_1 + x_2 \geq 3.5$$

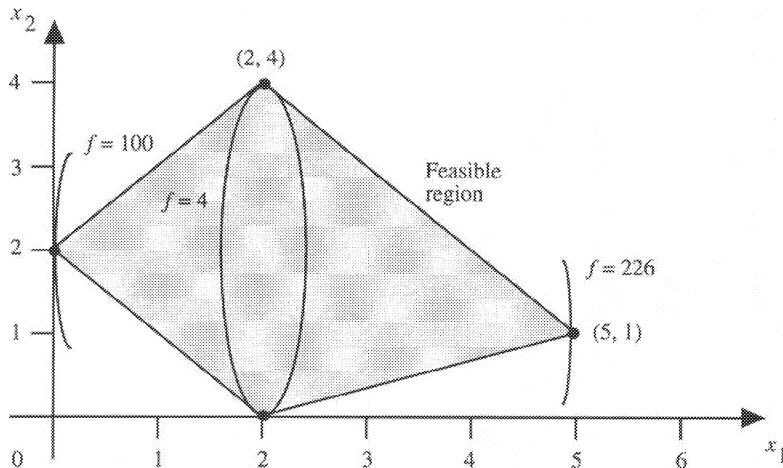


Figure 4.3 Feasible region illustrating local optima

and that the variables must be nonnegative. The set of feasible solutions consists of two disjoint parts, neither of which is convex. This is shown in Fig. 4.4. In such a case, local solutions different from the global solution may exist even when the objective function is linear.

For NLPs having multiple local optima yielding different objective function values, most of the computational techniques available can do no better than converge to a relative minimum. They will not, in general, provide an indication that global optimality has been achieved. Nevertheless, methods for finding local optima are often very useful in practice, especially if one has a good starting point or knows the neighborhood in which the global optimum lies.

4.1.3 Notation

The purpose of this section is to provide a selective discussion of mathematical definitions, notation, and results that are used in the remainder of the text. Although some of the notation has been used previously without definition, it is reintroduced here formally for completeness. For more detail, the reader should consult any number of texts on linear algebra and real analysis.

We denote by R^1 the real line and by R^n the space of all n -dimensional vectors. Intervals of real numbers or extended real numbers are denoted by bracket-parentheses notation. For example, for $a \in R^1$ or $a = -\infty$ and $b \in R^1$ or $b = +\infty$, the interval $(a, b]$ is equivalent to the set $\{x : a < x \leq b\}$. Given any subset $S \subset R^1$ which is

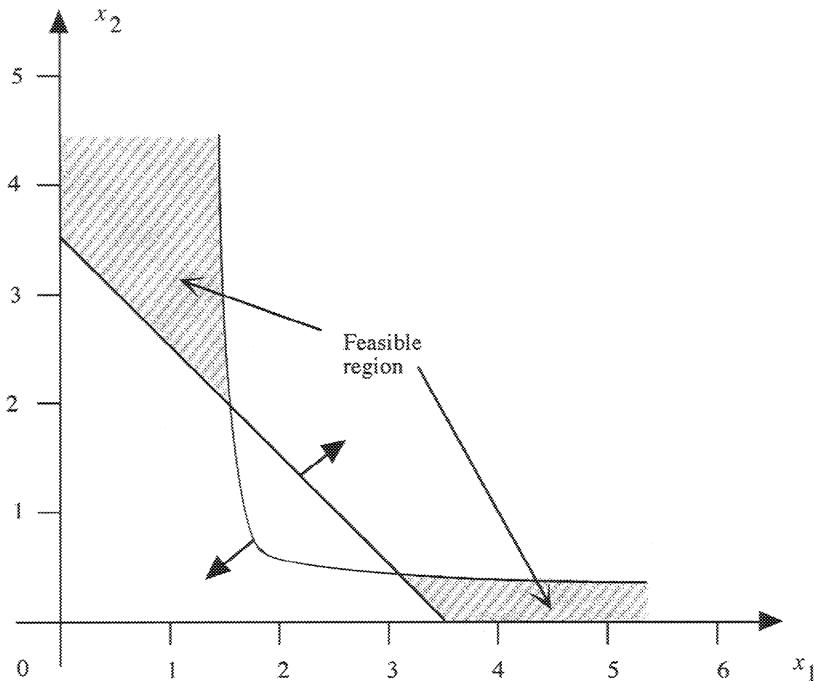


Figure 4.4 Example of nonconvex, disjoint feasible region

bounded above (below), we denote by $\sup S$ ($\inf S$) the least upper bound (greatest lower bound) of S . If S is unbounded above (below) we write $\sup S = +\infty$ ($\inf S = -\infty$). In general, every vector is considered to be a column vector unless otherwise stated. The exception is the gradient of a function $f : R^n \rightarrow R^1$ denoted by $\nabla f(\mathbf{x})$ and defined presently. The transpose of an $m \times n$ matrix \mathbf{A} is denoted by \mathbf{A}^T . A vector \mathbf{x} is treated as an $n \times 1$ matrix so \mathbf{x}^T denotes a $1 \times n$ matrix or row vector. Occasionally, the transpose symbol on a vector will be omitted when there is no ambiguity.

A symmetric $n \times n$ matrix \mathbf{A} is said to be positive semidefinite if $\mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0$ for all $\mathbf{x} \in R^n$. In this case, we write $\mathbf{A} \geq 0$. We say that \mathbf{A} is *positive definite* if $\mathbf{x}^T \mathbf{A} \mathbf{x} > 0$ for all $\mathbf{x} \neq 0$, and write $\mathbf{A} > 0$. When we say that \mathbf{A} is positive (semi)definite, it is implicitly assumed that it is symmetric. A symmetric $n \times n$ matrix \mathbf{A} has n real eigenvalues $\gamma_1, \dots, \gamma_n$ and n nonzero real eigenvectors ν_1, \dots, ν_n which are mutually orthogonal. It can be shown that

$$\gamma \mathbf{x}^T \mathbf{x} \leq \mathbf{x}^T \mathbf{A} \mathbf{x} \leq \Gamma \mathbf{x}^T \mathbf{x}, \quad \forall \mathbf{x} \in R^n \quad (4.2)$$

where

$$\gamma = \min\{\gamma_1, \dots, \gamma_n\}, \quad \Gamma = \max\{\gamma_1, \dots, \gamma_n\}$$

For \mathbf{x} equal to the eigenvector corresponding to $\Gamma(\gamma)$, the inequality on the right (left) in (4.2) becomes an equality. It follows that $\mathbf{A} > 0$ ($\mathbf{A} \geq 0$), if and only if the eigenvalues of \mathbf{A} are positive (nonnegative). If \mathbf{A} is positive definite, there exists a unique positive definite matrix the square of which equals \mathbf{A} . This is the matrix that has the same eigenvectors as \mathbf{A} and has eigenvalues the square roots of the eigenvalues of \mathbf{A} . We denote this matrix by $\mathbf{A}^{1/2}$.

A second set of conditions presented in the following proposition can also be used to test whether \mathbf{A} is positive definite.

Proposition 4.1.1 A necessary and sufficient condition for a real symmetric matrix $\mathbf{A} = [a_{ij}] \in R^{n \times n}$ to be positive definite is $a_{11} > 0$ and each determinant (also called principal minor)

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} > 0, \quad \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} > 0, \quad \dots \quad |\mathbf{A}| > 0$$

Let \mathbf{A} and \mathbf{B} be square matrices and \mathbf{C} be a matrix of appropriate dimension. The equation

$$(\mathbf{A} + \mathbf{C}\mathbf{B}\mathbf{C}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{C}(\mathbf{B}^{-1} + \mathbf{C}^T\mathbf{A}^{-1}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{A}^{-1} \quad (4.3)$$

holds provided all the inverses appearing in (4.3) hold. The validity of this equation can be seen by multiplying the right-hand side by $(\mathbf{A} + \mathbf{C}\mathbf{B}\mathbf{C}^T)$ and showing that the product is the identity matrix.

Consider the partitioned square matrix \mathbf{M} of the form

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}$$

Its inverse can be written conveniently as

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{Q} & -\mathbf{Q}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}\mathbf{Q} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}\mathbf{Q}\mathbf{B}\mathbf{D}^{-1} \end{bmatrix}$$

where

$$\mathbf{Q} = (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}$$

provided all the inverses appearing above exist. The proof is obtained once again by multiplying \mathbf{M} with the expression for \mathbf{M}^{-1} and verifying that the product yields the identity matrix.

Topological Notions

As a measure of distance, we shall use throughout the standard Euclidean norm in R^n denoted by $\|\cdot\|$; that is, for a vector $\mathbf{x} \in R^n$, we write $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}}$. The Euclidean norm of an $m \times n$ matrix \mathbf{A} is similarly denoted $\|\cdot\|$ and is given by

$$\|\mathbf{A}\| = \max_{\mathbf{x} \neq 0} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} = \max_{\mathbf{x} \neq 0} \frac{\sqrt{\mathbf{x}^T \mathbf{A}^T \mathbf{Ax}}}{\sqrt{\mathbf{x}^T \mathbf{x}}}$$

In view of (4.2), we have $\|\mathbf{A}\| = \sqrt{\text{max eigenvalue}(\mathbf{A}^T \mathbf{A})}$. If \mathbf{A} is symmetric, then

if $\gamma_1, \dots, \gamma_n$ are its (real) eigenvalues, the eigenvalues of \mathbf{A}^2 are $\gamma_1^2, \dots, \gamma_n^2$, and we obtain $\|\mathbf{A}\| = \max\{|\gamma_1|, \dots, |\gamma_n|\}$.

A sequence of vectors $\mathbf{x}^0, \mathbf{x}^1, \dots, \mathbf{x}^k, \dots$ in R^n , denoted by $\{\mathbf{x}^k\}$, is said to converge to a limit vector $\bar{\mathbf{x}}$ if $\|\mathbf{x}^k - \bar{\mathbf{x}}\| \rightarrow 0$ as $k \rightarrow \infty$ (that is, given an $\varepsilon > 0$, there is an N such that for all $k \geq N$ we have $\|\mathbf{x}^k - \bar{\mathbf{x}}\| < \varepsilon$). If $\{\mathbf{x}^k\}$ converges to $\bar{\mathbf{x}}$ we write $\mathbf{x}^k \rightarrow \bar{\mathbf{x}}$ or $\lim_{k \rightarrow \infty} \mathbf{x}^k = \bar{\mathbf{x}}$. Similarly, for a sequence of $m \times n$ matrices $\{\mathbf{A}^k\}$, we write $\mathbf{A}^k \rightarrow \bar{\mathbf{A}}$ or $\lim_{k \rightarrow \infty} \mathbf{A}^k = \bar{\mathbf{A}}$ if $\|\mathbf{A}^k - \bar{\mathbf{A}}\| \rightarrow 0$ as $k \rightarrow \infty$. Convergence of both vector and matrix sequences is equivalent to convergence of each of the sequences of their coordinates or elements.

Given a sequence $\{\mathbf{x}^k\}$, the subsequence $\{\mathbf{x}^k : k \in K\}$ corresponding to an infinite index set K is denoted by $\{\mathbf{x}^k\}_K$. A vector $\bar{\mathbf{x}}$ is said to be a *limit point* of a sequence $\{\mathbf{x}^k\}$ if there is a subsequence $\{\mathbf{x}^k\}_K$ which converges to $\bar{\mathbf{x}}$. A sequence of real numbers $\{r^k\}$ which is monotonically nondecreasing (nonincreasing), i.e., satisfies $r^k \leq r^{k+1}$ ($r^k \geq r^{k+1}$) for all k , must either converge to a real number or be unbounded above (below) in which case we write $\lim_{k \rightarrow \infty} r^k = +\infty$ ($\lim_{k \rightarrow \infty} r^k = -\infty$). Given any bounded sequence of real numbers $\{r^k\}$, we may consider the sequence $\{s^k\}$ where $s^k = \sup\{r^i : i \geq k\}$. Because this sequence is monotonically nonincreasing and bounded, it must have a limit called the *limit superior* of $\{r^k\}$ and denoted by $\limsup_{k \rightarrow \infty} r^k$. The *limit inferior* of $\{r^k\}$ is similarly defined and denoted $\liminf_{k \rightarrow \infty} r^k$. If $\{r^k\}$ is unbounded above, we write $\limsup_{k \rightarrow \infty} r^k = +\infty$, and if it is unbounded below, we write $\limsup_{k \rightarrow \infty} r^k = -\infty$.

Opened, Closed, and Compact Sets

For the vector $\mathbf{x} \in R^n$ and a scalar $\varepsilon > 0$, we denote the neighborhood or open sphere centered at \mathbf{x} with radius ε by $N_\varepsilon(\mathbf{x})$; i.e.,

$$N_\varepsilon(\mathbf{x}) = \{\mathbf{y} : \|\mathbf{y} - \mathbf{x}\| < \varepsilon\}.$$

A subset S of R^n is said to be *open*, if for every vector $\mathbf{x} \in S$ one can find an $\varepsilon > 0$ such that $N_\varepsilon(\mathbf{x}) \subset S$. The *interior* of a set $S \subset R^n$ is the set of all $\mathbf{x} \in S$ for which there exists $\varepsilon > 0$ such that $N_\varepsilon(\mathbf{x}) \subset S$. A set S is *closed* if and only if its complement in R^n is open. Equivalently S is closed if and only if every convergent sequence $\{\mathbf{x}^k\}$

with elements in S converges to a point which also belongs to S . A subset S of R^n is said to be *compact* if and only if it is both closed and bounded (i.e., it is closed and for some $M > 0$ we have $\|\mathbf{x}\| \leq M$ for all $\mathbf{x} \in S$). A set S is compact if and only if every sequence $\{\mathbf{x}^k\}$ with elements in S has at least one limit point which belongs to S . Another important fact is that if $S_0, S_1, \dots, S_k, \dots$ is a sequence of nonempty compact sets in R^n such that $S_k \supset S_{k+1}$ for all k then the intersection $\bigcap_{k=0}^{\infty} S_k$ is a nonempty and compact set.

Continuous Functions

A function f mapping a set $S_1 \subset R^n$ into a set $S_2 \subset R^m$ is denoted by $f : S_1 \rightarrow S_2$. The function f is said to be continuous at $\mathbf{x} \in S$ if $f(\mathbf{x}^k) \rightarrow f(\mathbf{x})$ whenever $\mathbf{x}^k \rightarrow \mathbf{x}$. Equivalently f is continuous at \mathbf{x} if given $\varepsilon > 0$ there is a $\delta > 0$ such that $\|\mathbf{y} - \mathbf{x}\| < \delta$ and $\mathbf{y} \in S$ implies $\|f(\mathbf{y}) - f(\mathbf{x})\| < \varepsilon$. The function f is said to be continuous over S (or simply continuous) if it is continuous at every point $\mathbf{x} \in S$. If S_1 , S_2 , and S_3 are sets and $f_1 : S_1 \rightarrow S_2$ and $f_2 : S_2 \rightarrow S_3$ are functions, the function $f_2 \cdot f_1 : S_1 \rightarrow S_3$ defined by $(f_2 \cdot f_1)(\mathbf{x}) = f_2[f_1(\mathbf{x})]$ is called the *composition* of f_1 and f_2 . If $f_1 : R^n \rightarrow R^m$ and $f_2 : R^m \rightarrow R^p$ are continuous, then $f_2 \cdot f_1$ is also continuous.

Differentiable Functions

A real-valued function $f : X \rightarrow R^1$, where $X \subset R^n$ is an open set is said to be *continuously differentiable* if the partial derivatives $\partial f(\mathbf{x})/\partial x_1, \dots, \partial f(\mathbf{x})/\partial x_n$ exist for all $\mathbf{x} \in X$ and are continuous functions of \mathbf{x} over X . In this case we write $f \in C^1$ over X . More generally, we write $f \in C^r$ over X for a function $f : X \rightarrow R^1$, where $X \subset R^n$ is an open set if all the partial derivatives of order r exist and are continuous functions of \mathbf{x} over X . If $f \in C^1$ on X , the *gradient* of f at a point $\mathbf{x} \in X$ is defined to be the row vector

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_n} \right]$$

If $f \in C^2$ over X , the *Hessian* of f at \mathbf{x} is defined to be the symmetric $n \times n$ matrix having $\partial^2 f(\mathbf{x})/\partial x_i \partial x_j$ as the ij th element

$$\nabla^2 f(\mathbf{x}) = \left[\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} \right]$$

If $\mathbf{f} : X \rightarrow R^m$ where $X \subset R^m$, then \mathbf{f} will be represented alternatively by the column vector of its components f_1, \dots, f_m

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix}$$

If X is open, we write $\mathbf{f} \in C^r$ on X if $f_1 \in C^r, \dots, f_m \in C^r$ on X . The $m \times n$ matrix $\nabla \mathbf{f}(\mathbf{x})$ is written

$$\nabla \mathbf{f}(\mathbf{x}) = [\nabla f_1(\mathbf{x})^T, \dots, \nabla f_m(\mathbf{x})^T]^T$$

and is known as the *Jacobian* of the vector-valued function \mathbf{f} .

On occasion, it will be necessary to consider gradients of functions with respect to a subset of the variables only. If $f : R^{n+s} \rightarrow R^1$ is a real-valued function of (\mathbf{x}, \mathbf{y}) where $\mathbf{x} = (x_1, \dots, x_n) \in R^n$ and $\mathbf{y} = (y_1, \dots, y_s) \in R^s$, we write

$$\nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) = \left[\frac{\partial f(\mathbf{x}, \mathbf{y})}{\partial x_1}, \dots, \frac{\partial f(\mathbf{x}, \mathbf{y})}{\partial x_n} \right]$$

$$\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) = \left[\frac{\partial f(\mathbf{x}, \mathbf{y})}{\partial y_1}, \dots, \frac{\partial f(\mathbf{x}, \mathbf{y})}{\partial y_s} \right]$$

$$\nabla_{\mathbf{xx}} f(\mathbf{x}, \mathbf{y}) = \left[\frac{\partial f(\mathbf{x}, \mathbf{y})}{\partial x_i \partial x_j} \right] \quad \nabla_{\mathbf{xy}} f(\mathbf{x}, \mathbf{y}) = \left[\frac{\partial f(\mathbf{x}, \mathbf{y})}{\partial x_i \partial y_j} \right]$$

$$\nabla_{\mathbf{yy}} f(\mathbf{x}, \mathbf{y}) = \left[\frac{\partial f(\mathbf{x}, \mathbf{y})}{\partial y_i \partial y_j} \right]$$

If $\mathbf{f} : R^{n+s} \rightarrow R^m$, $\mathbf{f} = (f_1, \dots, f_m)$, we write

$$\nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}, \mathbf{y}) = [\nabla_{\mathbf{x}} f_1(\mathbf{x}, \mathbf{y})^T, \dots, \nabla_{\mathbf{x}} f_m(\mathbf{x}, \mathbf{y})^T]^T$$

$$\nabla_{\mathbf{y}} \mathbf{f}(\mathbf{x}, \mathbf{y}) = [\nabla_{\mathbf{y}} f_1(\mathbf{x}, \mathbf{y})^T, \dots, \nabla_{\mathbf{y}} f_m(\mathbf{x}, \mathbf{y})^T]^T$$

For $\boldsymbol{\eta} : R^s \rightarrow R^m$ and $\boldsymbol{\theta} : R^n \rightarrow R^s$, consider the function $\mathbf{f} : R^n \rightarrow R^m$ defined by $\mathbf{f}(\mathbf{x}) = \boldsymbol{\eta}[\boldsymbol{\theta}(\mathbf{x})]$. Then if $\boldsymbol{\eta} \in C^r$ and $\boldsymbol{\theta} \in C^r$, we also have $\mathbf{f} \in C^r$. The chain rule of differentiation is stated in terms of this notation as

$$\nabla \mathbf{f}(\mathbf{x}) = \nabla_{\boldsymbol{\theta}} \boldsymbol{\eta}[\boldsymbol{\theta}(\mathbf{x})] \nabla_{\mathbf{x}} \boldsymbol{\theta}(\mathbf{x})$$

Directional Derivative

Let $\mathbf{v} = (v_1, \dots, v_n)$ be a nonzero column vector in R^n . Let $L = \{\mathbf{x}^0 + t\mathbf{v} : t \in R^1\}$. Then L is the line in R^n that passes through \mathbf{x}^0 in the direction \mathbf{v} . If

$$\lim_{t \rightarrow 0} \frac{f(\mathbf{x}^0 + t\mathbf{v}) - f(\mathbf{x}^0)}{t}$$

exists, then the limit is called the derivative of f at \mathbf{x}^0 in the direction \mathbf{v} (*directional derivative*). In this case we write

$$D_{\mathbf{v}} f(\mathbf{x}^0) = \lim_{t \rightarrow 0} \frac{f(\mathbf{x}^0 + t\mathbf{v}) - f(\mathbf{x}^0)}{t}$$

When f is continuous and has continuous partial derivatives throughout some neighborhood of \mathbf{x}^0 , we have

$$D_{\mathbf{v}} f(\mathbf{x}^0) = \lim_{t \rightarrow 0} \frac{f(\mathbf{x}^0 + t\mathbf{v}) - f(\mathbf{x}^0)}{t} = \sum_{j=1}^n \frac{\partial f(\mathbf{x}^0)}{\partial x_j} v_j$$

Now suppose that \mathbf{v} is a unit vector; i.e., $\|\mathbf{v}\| = 1$. Then $D_{\mathbf{v}} f(\mathbf{x}^0)$ is the slope of the tangent line at $(\mathbf{x}^0, f(\mathbf{x}^0))$ in the cross-section of the graph of f with the vertical plane passing through L . Also, $D_{\mathbf{v}} f(\mathbf{x}^0)$ gives the instantaneous rate of change of f at \mathbf{x}^0 in the direction of \mathbf{v} . If $\mathbf{v} = \mathbf{e}_j = (0, \dots, 0, 1, 0, \dots, 0)$, where the only nonzero entry of \mathbf{e}_j is in the j th position, then $D_{\mathbf{e}_j} f(\mathbf{x}^0) = \partial f(\mathbf{x}^0)/\partial x_j$.

Whenever f is differentiable at \mathbf{x}^0 , the directional derivative can be computed by $D_{\mathbf{v}} f(\mathbf{x}^0) = \nabla f(\mathbf{x}^0) \cdot \mathbf{v}$. In addition, we have

$$\max_{\|\mathbf{v}\|=1} D_{\mathbf{v}} f(\mathbf{x}^0) = \|\nabla f(\mathbf{x}^0)\|$$

and if $\nabla f(\mathbf{x}^0) \neq 0$, then $\|\nabla f(\mathbf{x}^0)\|$ occurs whenever $\mathbf{v} = \nabla f(\mathbf{x}^0)^T / \|\nabla f(\mathbf{x}^0)\|$. Likewise

$$\min_{\|\mathbf{v}\|=1} D_{\mathbf{v}} f(\mathbf{x}^0) = -\|\nabla f(\mathbf{x}^0)\|$$

which occurs when $\mathbf{v} = -\nabla f(\mathbf{x}^0) / \|\nabla f(\mathbf{x}^0)\|$. Thus the gradient vector $\nabla f(\mathbf{x}^0)$ points in the direction of steepest ascent while its negative points in the direction of steepest descent. This result has important implications for algorithmic development.

If a differentiable function f has either a relative maximum or a relative minimum at any interior point \mathbf{x}^0 of its domain, then it necessarily follows that $\nabla f(\mathbf{x}^0) = 0$. This observation is formalized in the next section. Such points are called *stationary points*. Of course, a relative extrema can occur on the boundary of the feasible region as well.

When $f : S \rightarrow R^1$ is continuous for S a closed and bounded subset of R^n , there exist points \mathbf{x}^* and \mathbf{y}^* in S such that $f(\mathbf{x}^*) = \min\{f(\mathbf{x}) : \mathbf{x} \in S\}$ and $f(\mathbf{y}^*) = \max\{f(\mathbf{x}) : \mathbf{x} \in S\}$. This result is known as the *Weierstrass Theorem* and ensures the existence of extrema of continuous functions defined over compact sets.

When $f(\mathbf{x}) = c$, each $\partial f(\mathbf{x}^0)/\partial x_j$ is continuous at \mathbf{x}^0 , and $\nabla f(\mathbf{x}^0)$ is not the zero vector, then $\nabla f(\mathbf{x}^0)$ is perpendicular to the set $\{\mathbf{x} : f(\mathbf{x}) = c\}$, where c is a constant. Also, an equation of the *tangent plane* to the surface $f(\mathbf{x}) = c$ at \mathbf{x}^0 is

$$0 = \nabla f(\mathbf{x}^0)(\mathbf{x} - \mathbf{x}^0) = \sum_{j=1}^n \frac{\partial f(\mathbf{x}^0)}{\partial x_j} (x_j - x_j^0)$$

For more general surfaces, say $S = \{\mathbf{x} : h_i(\mathbf{x}) = b_i, i = 1, \dots, m\}$, we define a *curve* to be a family of points $\mathbf{x}(\tau) \in S$ continuously parameterized by $\tau \in [a, b] \subset R^1$. The curve is differentiable if $\dot{\mathbf{x}}(\tau) \triangleq d\mathbf{x}(\tau)/d\tau$ exists, and is twice differentiable if $\ddot{\mathbf{x}}(\tau)$ exists. A curve is said to pass through the point \mathbf{x}^* if $\mathbf{x}^* = \mathbf{x}(\tau^*)$ for some $\tau^* \in [a, b]$. The derivative of the curve at \mathbf{x}^* is defined as $\dot{\mathbf{x}}(\tau^*)$ and is itself a vector in R^n . Now consider all differentiable curves on S passing through a point \mathbf{x}^* . The *tangent plane* at \mathbf{x}^* is defined as the collection of the derivatives at \mathbf{x}^* of all these differentiable curves. The tangent plane is a subspace in R^n .

Mean Value Theorems and Taylor Series Expansions

Many of the fundamental results in nonlinear programming are obtained with the help of a Taylor series expansion of the problem functions. Closely related to the Taylor series is the mean value theorem which states: If $f \in C^1$ in a region containing the line segment $[\mathbf{x}, \mathbf{y}]$, then there is an $\alpha \in [0, 1]$ such that

$$f(\mathbf{y}) = f(\mathbf{x}) + \nabla f(\alpha \mathbf{x} + (1 - \alpha)\mathbf{y})(\mathbf{y} - \mathbf{x})$$

The corresponding first order Taylor series is written as

$$f(\mathbf{y}) = f(\mathbf{x}) + \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}) + o(\|\mathbf{y} - \mathbf{x}\|)$$

where $o(\|\mathbf{y} - \mathbf{x}\|)$ denotes terms that go to zero faster than $\|\mathbf{y} - \mathbf{x}\|$ and hence can be ignored when \mathbf{y} is in a neighborhood of \mathbf{x} (see below).

Furthermore, if $f \in C^2$ then there is an $\alpha \in [0, 1]$ such that

$$f(\mathbf{y}) = f(\mathbf{x}) + \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}) + \frac{1}{2}(\mathbf{y} - \mathbf{x})^T \mathbf{F}(\alpha \mathbf{x} + (1 - \alpha)\mathbf{y})(\mathbf{y} - \mathbf{x})$$

where \mathbf{F} denotes the Hessian of f . The second order Taylor series is written as

$$f(\mathbf{y}) = f(\mathbf{x}) + \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}) + \frac{1}{2}(\mathbf{y} - \mathbf{x})^T \mathbf{F}(\mathbf{x})(\mathbf{y} - \mathbf{x}) + o(\|\mathbf{y} - \mathbf{x}\|^2)$$

Implicit Function Theorem

Suppose we have a set of m equations in n variables

$$h_i(\mathbf{x}) = 0, \quad i = 1, \dots, m$$

where $\mathbf{x} \in R^n$, $n > m$. The implicit function theorem addresses the question as to whether by fixing $n - m$ of the variables, the equations can be solved for the remaining m variables in the form

$$x_i = \phi_i(x_{m+1}, x_{m+2}, \dots, x_n), \quad i = 1, \dots, m$$

The functions ϕ_i , if they exist, are called *implicit functions*.

Theorem 4.1.1 Let $\mathbf{x}^0 = (x_1^0, x_2^0, \dots, x_n^0)$ be a point in R^n satisfying the properties

- i) The functions $h_i \in C^r$ ($i = 1, \dots, m$), in some neighborhood of \mathbf{x}^0 , for some $r \geq 1$.
- ii) $h_i(\mathbf{x}^0) = 0$, $i = 1, \dots, m$
- iii) The $m \times m$ Jacobian matrix

$$J = \begin{bmatrix} \frac{\partial h_1(\mathbf{x}^0)}{\partial x_1} & \dots & \frac{\partial h_1(\mathbf{x}^0)}{\partial x_m} \\ \vdots & & \vdots \\ \frac{\partial h_m(\mathbf{x}^0)}{\partial x_1} & \dots & \frac{\partial h_m(\mathbf{x}^0)}{\partial x_m} \end{bmatrix}$$

is nonsingular.

Then there is a neighborhood of $\hat{\mathbf{x}}^0 = (x_{m+1}^0, x_{m+2}^0, \dots, x_n^0) \in R^{n-m}$ such that for $\hat{\mathbf{x}} = (x_{m+1}, x_{m+2}, \dots, x_n)$ in this neighborhood there are functions $\phi_i(\hat{\mathbf{x}})$ ($i = 1, \dots, m$) such that

- i) $\phi_i \in C^r$
- ii) $x_i^0 = \phi_i(\hat{\mathbf{x}}^0)$, $i = 1, \dots, m$
- iii) $h_i(\phi_1(\hat{\mathbf{x}}), \phi_2(\hat{\mathbf{x}}), \dots, \phi_m(\hat{\mathbf{x}}), \hat{\mathbf{x}}) = 0$, $i = 1, \dots, m$

Example 4.1.3 Consider the equation $x_1^2 + x_2 = 0$. A solution is $x_1 = 0, x_2 = 0$. In a neighborhood of this solution, though, there is no function ϕ such that $x_1 = \phi(x_2)$ so condition (iii) above is violated. At any other solution, however, such a ϕ exists.

Example 4.1.4 Let \mathbf{A} be an $m \times n$ matrix ($m < n$) and consider the system of linear equations $\mathbf{Ax} = \mathbf{b}$. If \mathbf{A} is partitioned as $\mathbf{A} = [\mathbf{B}, \mathbf{C}]$ where \mathbf{B} is $m \times m$ then condition (iii) is satisfied if and only if \mathbf{B} is nonsingular. This corresponds exactly with what the theory of linear equations tells us, implying that the implicit function theorem can be regarded as a nonlinear generalization of linear theory.

Higher Order Terms

If f is a real-valued function of a real variable, the notation $f(x) = O(x)$ means that $f(x)$ goes to zero at least as fast as x does. More precisely, it means that there exists a $K \geq 0$ such that

$$\left| \frac{f(x)}{x} \right| \leq K \text{ as } x \rightarrow 0$$

The notation $f(x) = o(x)$ means that $f(x)$ goes to zero faster than x does, or equivalently, that the K above is zero.

Convexity

A set $S \subset R^n$ is said to be convex if for every $\mathbf{x}, \mathbf{y} \in S$ and $\alpha \in [0, 1]$ we have $\alpha\mathbf{x} + (1 - \alpha)\mathbf{y} \in S$. A function $f : S \rightarrow R^1$ is said to be convex over the convex set S if for every $\mathbf{x}, \mathbf{y} \in S$ we have

$$f(\alpha\mathbf{x} + (1 - \alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1 - \alpha)f(\mathbf{y}) \quad (4.4)$$

If f is convex and $f \in C^1$ over an open convex set S , then

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \quad (4.5)$$

In addition, if $f \in C^2$ over S , then $\nabla^2 f(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in S$. Conversely, if $f \in C^1$ over S and (4.5) holds, or if $f \in C^2$ over S and $\nabla^2 f(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in S$, then f is convex over S .

Condition (4.4) says that the value of a convex function is always overestimated by linear interpolation, while (4.5) says that it is always underestimated by linear extrapolation. If f is convex then $-f$ is said to be *concave*. Conditions (4.4) and (4.5) hold for concave functions with the inequality signs reversed.

Rate of Convergence

A critical characteristic of algorithms is the speed with which they converge to a limit. Given a sequence $\{\mathbf{x}^k\} \subset R^n$ with $\mathbf{x}^k \rightarrow \mathbf{x}^*$, the typical approach is to measure speed of convergence in terms of an error function $e : R^n \rightarrow R^1$ satisfying $e(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in R^n$ and $e(\mathbf{x}^*) = 0$. Typical choices are

$$e(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^*\|, \quad e(\mathbf{x}) = |f(\mathbf{x}) - f(\mathbf{x}^*)|$$

where f is the objective function of the problem. The sequence $\{e(\mathbf{x}^k)\}$ is then compared with some standard sequence, the most common being geometric progressions of the form $r_k = q\beta^k$, where $q > 0$ and $\beta \in (0, 1)$ are scalars, and $r_k = q\beta^{p_k}$, where $p_k > 0$. There is no reason for selecting these particular sequences for comparison other than the fact that they represent a sufficiently wide class which is adequate and convenient.

Definition 4.1.1 Given two scalar sequences $\{e_k\}$ and $\{r_k\}$ with

$$0 \leq e_k, \quad 0 \leq r_k, \quad e_k \rightarrow 0, \quad r_k \rightarrow 0$$

we say that $\{e_k\}$ converges faster than $\{r_k\}$ if there exists an index $K \geq 0$ such that

$$0 \leq e_k \leq r_k \quad \forall k \geq K$$

We say that $\{e_k\}$ converges slower than $\{r_k\}$ if there exists an index $K \geq 0$ such that

$$0 \leq r_k \leq e_k \quad \forall k \geq K$$

Definition 4.1.2 Consider a scalar sequence $\{e_k\}$ with $e_k \geq 0$, $e_k \rightarrow 0$. The sequence $\{e_k\}$ is said to converge *at least linearly with convergence ratio β* , where $0 < \beta < 1$, if it converges faster than all geometric progressions of the form $q\bar{\beta}^k$, where $q > 0$, $\bar{\beta} \in (\beta, 1)$. It is said to converge *at most linearly with convergence ratio β* if it converges slower than all geometric progressions of the form $q\bar{\beta}^k$, where $q > 0$, $\bar{\beta} \in (0, \beta)$. It is said to converge *linearly with convergence ratio β* if it converges both at least and at most linearly with convergence ratio β . It is said to converge *superlinearly* or *sublinearly* if it converges faster or slower, respectively, than every sequence of the form $q\beta^k$, where $q > 0$ and $\beta \in (0, 1)$.

Example 4.1.5 (1) The following sequences all converge linearly with convergence ratio β :

$$q\beta^k, \quad q\left(\beta + \frac{1}{k}\right)^k, \quad q\left(\beta - \frac{1}{k}\right)^k, \quad q\beta^{k+(1/k)}$$

where $q > 0$ and $\beta \in (0, 1)$. This result follows by straightforward verification of the definition or by making use of Proposition 4.1.2 below.

(2) Let $0 < \beta_1 < \beta_2 < 1$, and consider the sequence $\{e_k\}$ defined by

$$e_k = \beta_1^k \beta_2^k, \quad e_{2k+1} = \beta_1^{k+1} \beta_2^k$$

Then $\{e_k\}$ converges at least linearly with convergence ratio β_2 and converges linearly with convergence ratio β_1 . Actually, $\{e_k\}$ can be shown to converge linearly with convergence ratio $\sqrt{\beta_1 \beta_2}$ with the help of Proposition 4.1.2.

(3) The sequence $\{1/k\}$ converges sublinearly and every sequence of the form $q\beta^{pk}$, where $q > 0$, $\beta \in (0, 1)$, $p_k > 1$, can be shown to converge superlinearly. Again, these results follow from Proposition 4.1.2.

Proposition 4.1.2 Let $\{e_k\}$ be a scalar sequence with $e_k \geq 0$, $e_k \rightarrow 0$. Then the following hold true:

- (a) The sequence $\{e_k\}$ converges at least linearly with convergence ratio $\beta \in (0, 1)$ if and only if

$$\limsup_{k \rightarrow \infty} e_k^{1/k} \leq \beta$$

It converges at most linearly with convergence ratio $\beta \in (0, 1)$ if and only if

$$\liminf_{k \rightarrow \infty} e_k^{1/k} \geq \beta$$

It converges linearly with convergence ratio $\beta \in (0, 1)$ if and only if

$$\lim_{k \rightarrow \infty} e_k^{1/k} = \beta$$

- (b) If $\{e_k\}$ converges faster (slower) than some geometric progression of the form $q\beta^k$, $q > 0$, $\beta \in (0, 1)$ then it converges at least (at most) linearly with convergence ratio β .
- (c) Assume that $e_k \neq 0$ for all k , and denote

$$\beta_1 = \liminf_{k \rightarrow \infty} \frac{e_{k+1}}{e_k}, \quad \beta_2 = \limsup_{k \rightarrow \infty} \frac{e_{k+1}}{e_k}$$

If $0 < \beta_1 < \beta_2 < 1$, then $\{e_k\}$ converges at least linearly with convergence ratio β_2 and at most linearly with convergence ratio β_1 .

- (d) Assume that $e_k \neq 0$ for all k and that

$$\lim_{k \rightarrow \infty} \frac{e_{k+1}}{e_k} = \beta$$

If $0 < \beta < 1$, then $\{e_k\}$ converges linearly with convergence ratio β . If $\beta = 0$, then $\{e_k\}$ converges superlinearly. If $\beta = 1$, then $\{e_k\}$ converges sublinearly.

The proof can be found in [B21].

In each of the above expressions, if e_k converges to a point $e^* \neq 0$, then we would replace e_k with $|e_k - e^*|$. Note that portions of (a) and (c) are stated in terms of limit superior and limit inferior instead of just limit, and $e_k = 0$ is ruled out in (c) and (d). This was to ensure that the results are applicable to any sequence; however, such assumptions are rarely necessary in actual analysis since sequences generated by algorithms are generally well behaved.

Most optimization algorithms that are of practical interest produce sequences converging either linearly or superlinearly. The former is quite satisfactory provided that the convergence ratio β is not very close to unity. Algorithms that may produce sequences having sublinear convergence rates are dismissed from consideration in most optimization work as computationally inefficient. Several optimization algorithms possess superlinear convergence for particular classes of problems so for this reason, it is important to quantify this concept further.

Definition 4.1.3 (*Order of convergence*) Consider a scalar sequence $\{e_k\}$ with $e_k \geq 0$ converging superlinearly to zero. Then $\{e_k\}$ is said to converge *at least superlinearly with order p*, where $p > 1$, if it converges faster than all sequences of the form $q\beta^{\bar{p}_k}$, where $q > 0$, $\beta \in (0, 1)$, and $\bar{p}_k \in (1, p)$. It is said to converge *at most superlinearly with order p* if it converges slower than all sequences of the form $q\beta^{\bar{p}_k}$. It is said to converge superlinearly with order p , if it converges both at least and at most superlinearly with order p .

It should be noted that the order of convergence as well as all other notions related to speed of convergence that have been introduced, are determined only by the properties

of the sequence that hold as $k \rightarrow \infty$. Loosely, we refer to this as the *tail* of the sequence and say that the order of convergence is a measure of how good the worst part of the tail is. Larger values of the order p imply faster convergence since the distance from the limit point e^* is reduced, at least in the tail, by the p th power in a single step. If the sequence has order p and (as is the usual case) the limit

$$\beta = \lim_{k \rightarrow \infty} \frac{|e_{k+1} - e^*|}{|e_k - e^*|}$$

exists, then asymptotically we have $|e_{k+1} - e^*| = \beta |e_k - e^*|^p$.

Example 4.1.6 The sequence with $e_k = a^k$, where $0 < a < 1$, converges to zero with order unity, since $r_{k+1}/r_k = a$.

Example 4.1.7 The sequence with $e_k = a^{(2^k)}$, where $0 < a < 1$, converges to zero with order two, since $r_{k+1}/r_k^2 = 1$.

4.2 OPTIMALITY CONDITIONS

The way most mathematical programs are solved is by applying an algorithm that searches for a point in the feasible region which satisfies a set of optimality conditions. Depending on the formulation and the properties of the underlying functions, it may only be possible to verify that a point satisfies necessary conditions for a relative extremum. In other cases, usually when the functions meet certain convexity requirements, it may be possible to establish sufficiency. As such, the aim of this section is to outline the optimality conditions associated with various classes of nonlinear programs. Throughout the discussion it will be assumed that we are dealing with functions that are at least twice continuously differentiable. For the theory of nondifferentiable optimization, see [F6], [K4] or [S10] among others. All the results are stated for the minimization of the objective function but comparable results hold for maximization as well.

Definition 4.2.1 A point $\mathbf{x}^* \in X$ is said to be a *relative* or *local minimum* of f over X if there is an $\varepsilon > 0$ such that $f(\mathbf{x}) \geq f(\mathbf{x}^*)$ for all $\mathbf{x} \in X$ within a distance ε of \mathbf{x}^* (that is, $\mathbf{x} \in X$ and $\|\mathbf{x} - \mathbf{x}^*\| < \varepsilon$). If $f(\mathbf{x}) > f(\mathbf{x}^*)$ for all $\mathbf{x} \in X$, $\mathbf{x} \neq \mathbf{x}^*$ within a distance ε of \mathbf{x}^* , then \mathbf{x}^* is said to be a *strict relative minimum* of f over X .

Definition 4.2.2 A point $\mathbf{x}^* \in X$ is said to be a *global minimum* of f over X if $f(\mathbf{x}) \geq f(\mathbf{x}^*)$ for all $\mathbf{x} \in X$. If $f(\mathbf{x}) > f(\mathbf{x}^*)$ for all $\mathbf{x} \in X$, $\mathbf{x} \neq \mathbf{x}^*$, then \mathbf{x}^* is said to be a *strict global minimum* of f over X .

We begin with an examination of the unconstrained problem. Next we study the case where the decision variables are restricted to be nonnegative followed by the classical

case where only equalities are present. We then discuss the most general version of the problem where the feasible region is defined by a combination of equality and inequality constraints, as indicated in (4.1). Although it is possible to identify several special cases, such as when the constraints in (4.1) are linear and X is the nonnegative orthant, these will not be singled out for individual treatment. Also, most proofs have been omitted. The interested reader can either work out the details himself or refer to any number of NLP texts (e.g., [B16, G7]). We conclude this section with a review of convex theory.

4.2.1 Unconstrained Problems

The simplest situation that we address concerns the minimization of a function f in the absence of any constraints. This problem can be written as follows:

$$\min\{f(\mathbf{x}) : \mathbf{x} \in R^n\}$$

where $f : R^n \rightarrow R^1$. Without additional assumptions on the nature of f we will most likely have to be content with finding a point that is a relative minimum. Let \mathbf{x}^* be such a point and assume that f is twice continuously differentiable in a neighborhood $N_\varepsilon(\mathbf{x}^*)$ of \mathbf{x}^* , where $\mathbf{x} \in N_\varepsilon(\mathbf{x}^*)$ implies that $f(\mathbf{x}) \geq f(\mathbf{x}^*)$. Then for every direction vector \mathbf{v} , where $\|\mathbf{v}\| = 1$, we have

$$0 \leq D_{\mathbf{v}} f(\mathbf{x}^*) = \lim_{t \rightarrow 0} \frac{f(\mathbf{x}^* + t\mathbf{v}) - f(\mathbf{x}^*)}{t} = \nabla f(\mathbf{x}^*)\mathbf{v}$$

Thus it follows by considering the cases where $\mathbf{v} = \mathbf{e}_j$ and $\mathbf{v} = -\mathbf{e}_j$ that $\partial f(\mathbf{x}^*)/\partial x_j = 0$. Because this is true for every j , we have our first necessary condition.

$$\nabla f(\mathbf{x}^*) = 0 \tag{4.6}$$

for a relative minimum at \mathbf{x}^* .

Notice that every point \mathbf{x} in $N_\varepsilon(\mathbf{x}^*)$ can be expressed in the form $\mathbf{x} = \mathbf{x}^* + t\mathbf{v}$, where $t > 0$ (in fact, $0 < t < \varepsilon$ since $t = \|t\mathbf{v}\| = \|\mathbf{x} - \mathbf{x}^*\| < \varepsilon$). Now, applying the mean value theorem we obtain

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) = f(\mathbf{x}^* + t\mathbf{v}) = f(\mathbf{x}^*) + t\nabla f(\mathbf{x}^*)\mathbf{v} + \frac{t^2}{2}\mathbf{v}^T \mathbf{F}(\mathbf{x}^* + \alpha t\mathbf{v})\mathbf{v} \tag{4.7}$$

where \mathbf{F} is the Hessian matrix of f and $\alpha \in [0, 1]$. But $\nabla f(\mathbf{x}^*) = 0$ and $\frac{t^2}{2} > 0$ so a second necessary condition for f to have a relative minimum at \mathbf{x}^* is

$$0 \leq \mathbf{v}^T \mathbf{F}(\mathbf{x}^* + \alpha t\mathbf{v})\mathbf{v} \tag{4.8}$$

From continuity arguments it follows that (4.8) holds when $\alpha = 0$. This leads to following result:

Theorem 4.2.1 Let $f : R^n \rightarrow R^1$ be twice continuously differentiable throughout a neighborhood of \mathbf{x}^* . If f has a relative minimum at \mathbf{x}^* , then it necessarily follows that

- (i) $\nabla f(\mathbf{x}^*) = 0$
- (ii) $\mathbf{F}(\mathbf{x}^*)$ is positive semidefinite.

Sufficient conditions for a relative maximum are given in the next theorem. The proof is left as an exercise.

Theorem 4.2.2 Let $f : R^n \rightarrow R^1$ be twice continuously differentiable throughout a neighborhood of \mathbf{x}^* . Then a sufficient condition for $f(\mathbf{x})$ to have a strict relative minimum at \mathbf{x}^* , where (4.6) holds, is that $\mathbf{F}(\mathbf{x}^*)$ be positive definite.

Example 4.2.1 Let $f : R^2 \rightarrow R^1$ be defined by $f(\mathbf{x}) = 3x_1^3 + x_2^2 - 9x_1 + 4x_2$. Find the relative extreme values of this function.

Solution: Notice that

$$\nabla f(\mathbf{x}) = [\begin{array}{cc} 0 & 0 \end{array}] = [\begin{array}{cc} 9x_1^2 - 9 & 2x_2 + 4 \end{array}]$$

implies that $x_1 = \pm 1$ and $x_2 = -2$. Checking $\mathbf{x} = (1, -2)$, we have

$$\mathbf{F}(1, -2) = [\begin{array}{cc} 18 & 0 \\ 0 & 2 \end{array}]$$

which is positive definite since $\mathbf{v}^T \mathbf{F}(1, -2) \mathbf{v} = 18v_1^2 + v_2^2 > 0$ when $\mathbf{v} \neq 0$. Thus $(1, -2)$ gives a strict relative minimum. Next consider $\mathbf{x} = (-1, -2)$ with Hessian

$$\mathbf{F}(-1, -2) = [\begin{array}{cc} -18 & 0 \\ 0 & 2 \end{array}]$$

Now we have $\mathbf{v}^T \mathbf{F}(-1, -2) \mathbf{v} = -18v_1^2 + v_2^2$ which may equal 0 when $\mathbf{v} \neq 0$. Thus the sufficient condition for $(-1, -2)$ to be either a relative minimum or maximum is not satisfied. Actually, the second necessary condition in Theorem 4.2.1 for either a relative minimum or maximum is not satisfied. Therefore, $(1, -2)$ gives the only relative extreme value of f .

4.2.2 Nonnegative Variables

Again let $f : R^n \rightarrow R^1$ and consider the problem

$$\min\{f(\mathbf{x}) : \mathbf{x} \geq \mathbf{0}\} \tag{4.9}$$

Suppose that f has a relative minimum at \mathbf{x}^* , where $\mathbf{x}^* \geq \mathbf{0}$. Then there exists a neighborhood $N_\epsilon(\mathbf{x}^*)$ of \mathbf{x}^* so that whenever $\mathbf{x} \in N_\epsilon(\mathbf{x}^*)$ and $\mathbf{x} \geq \mathbf{0}$, we have

$f(\mathbf{x}) \geq f(\mathbf{x}^*)$. Now write $\mathbf{x} = \mathbf{x}^* + t\mathbf{v}$, where \mathbf{v} is a direction vector and $t > 0$. Assuming that f is twice continuously differentiable throughout $N_\epsilon(\mathbf{x}^*)$, the mean value theorem (4.7) implies that

$$0 \leq \nabla f(\mathbf{x}^*)\mathbf{v} + \frac{t}{2}\mathbf{v}^T \mathbf{F}(\mathbf{x}^* + \alpha t\mathbf{v})\mathbf{v}$$

which is obtained by canceling terms and dividing through by t . Taking the limit as $t \rightarrow 0$ gives $0 \leq \nabla f(\mathbf{x}^*)\mathbf{v}$. If $\mathbf{x}^* > 0$, we know that $\nabla f(\mathbf{x}^*) = \mathbf{0}$. Suppose that some $x_j^* = 0$. Let $\mathbf{v} = \mathbf{e}_j$. Then $0 \leq \nabla f(\mathbf{x}^*)\mathbf{e}_j = \partial f(\mathbf{x}^*)/\partial x_j$. Finally, if $\mathbf{x}^* \not> 0$ but $x_j^* > 0$, consider the restriction of the domain of f to $X = \{(x_1^*, \dots, x_{j-1}^*, z_j, x_{j+1}^*, \dots, x_n^*): z_j \geq 0\}$. Then f restricted to X has a relative minimum at \mathbf{x}^* . It follows that $\partial f(\mathbf{x}^*)/\partial x_j = 0$. Thus necessary conditions for a relative minimum of f at \mathbf{x}^* include

$$\begin{aligned} \frac{\partial f(\mathbf{x}^*)}{\partial x_j} &= 0, & \text{if } x_j^* > 0 \\ \frac{\partial f(\mathbf{x}^*)}{\partial x_j} &\geq 0, & \text{if } x_j^* = 0 \end{aligned}$$

These results are summarized in the following proposition.

Proposition 4.2.1 Necessary conditions for a relative minimum of f in problem (4.9) to occur at \mathbf{x}^* include

$$\nabla f(\mathbf{x}^*) \geq \mathbf{0}, \quad \nabla f(\mathbf{x}^*)\mathbf{x}^* = \mathbf{0}, \quad \mathbf{x}^* \geq \mathbf{0} \quad (4.10)$$

where f is twice continuously differentiable throughout a neighborhood of \mathbf{x}^* .

Example 4.2.2 Minimize $f(\mathbf{x}) = 3x_1^2 + x_2^2 + x_3^2 - 2x_1x_2 - 2x_1x_3 - 2x_1$ subject to $\mathbf{x} \geq \mathbf{0}$.

Solution: From (4.10) we have the following necessary conditions for a relative minimum:

$$(1) \quad 0 \leq \frac{\partial f}{\partial x_1} = 6x_1 - 2x_2 - 2x_3 - 2$$

$$(2) \quad 0 = x_1 \frac{\partial f}{\partial x_1} = x_1(6x_1 - 2x_2 - 2x_3 - 2)$$

$$(3) \quad 0 \leq \frac{\partial f}{\partial x_2} = 2x_2 - 2x_1$$

$$(4) \quad 0 = x_2 \frac{\partial f}{\partial x_2} = x_2(2x_2 - 2x_1)$$

$$(5) \quad 0 \leq \frac{\partial f}{\partial x_3} = 2x_3 - 2x_1$$

$$(6) \quad 0 = x_3 \frac{\partial f}{\partial x_3} = x_3(2x_3 - 2x_1)$$

$$(7) \quad x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$$

From (4) we see that $x_2 = 0$ or $x_1 = x_2$. When $x_2 = 0$, (3) and (7) imply that $x_1 = 0$. From (6) then, $x_3 = 0$. But this contradicts (1) so $x_2 \neq 0$ and $x_1 = x_2$.

Condition (6) implies that $x_3 = 0$ or $x_1 = x_2 = x_3$. If $x_3 = 0$, then (4), (5) and (7) imply that $x_1 = x_2 = x_3 = 0$. But this situation has been ruled out. Thus, $x_1 = x_2 = x_3$, and from (2) we get $x_1 = 0$ or $x_1 = 1$. Since $x_1 \neq 0$, the only possible relative minimum of f occurs when $x_1 = x_2 = x_3 = 1$. The Hessian at $\mathbf{x}^* = (1, 1, 1)$ is

$$\mathbf{F}(1, 1, 1) = \begin{bmatrix} 6 & -2 & -2 \\ -2 & 2 & 0 \\ -2 & 0 & 2 \end{bmatrix}$$

which, from Proposition 4.1.1, is seen to be positive definite. Thus f is strictly convex and has a relative minimum at \mathbf{x}^* . It follows from Theorem 4.2.10 in Section 4.2.5 that $f(\mathbf{x}^*) = 1$ is a global minimum.

4.2.3 Equality Constraints

Let $f : R^n \rightarrow R^1$ and consider the problem

$$\min\{f(\mathbf{x}) : \mathbf{h}(\mathbf{x}) = \mathbf{0}\} \tag{4.11}$$

where $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \dots, h_m(\mathbf{x}))$, each $h_i : R^n \rightarrow R^1$, and $m < n$. Let problem (4.11) have a relative minimum at \mathbf{x}^* . Then there exists an $\varepsilon > 0$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ whenever $\mathbf{x} \in N_\varepsilon(\mathbf{x}^*)$ and $\mathbf{h}(\mathbf{x}) = \mathbf{0}$. Suppose that $f \in C^2$ throughout $N_\varepsilon(\mathbf{x}^*)$, and assume that each $\partial h_i(\mathbf{x})/\partial x_j$ is once continuously differentiable as well throughout this neighborhood. Next, let the rank of the following Jacobian matrix be m .

$$\frac{\partial \mathbf{h}(\mathbf{x}^*)}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial h_1(\mathbf{x}^*)}{\partial x_1} & \dots & \frac{\partial h_1(\mathbf{x}^*)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m(\mathbf{x}^*)}{\partial x_1} & \dots & \frac{\partial h_m(\mathbf{x}^*)}{\partial x_n} \end{bmatrix} \tag{4.12}$$

This is equivalent to saying that \mathbf{x}^* is a regular point [L7].

Definition 4.2.3 A point \mathbf{x}^* satisfying the constraints $\mathbf{h}(\mathbf{x}^*) = \mathbf{0}$ is said to be a *regular point* of the constraints if the gradient vectors $\nabla h_1(\mathbf{x}^*), \dots, \nabla h_m(\mathbf{x}^*)$ are linearly independent.

At regular points it is possible to characterize the tangent plane in terms of the gradients of the constraint functions.

Theorem 4.2.3 At a regular point \mathbf{x}^* of the surface $S = \{\mathbf{x} : \mathbf{h}(\mathbf{x}) = 0\}$ the tangent plane is equal to $T = \{\mathbf{y} : \nabla \mathbf{h}(\mathbf{x})\mathbf{y} = 0\}$.

To simplify the following argument we shall assume that the first m columns of (4.12) are linearly independent. According to the implicit function theorem in Section 4.1.3, given m continuously differentiable functions of n variables, where $m < n$, for which the Jacobian matrix has rank m , it is possible to solve for m of the variables in a neighborhood of \mathbf{x}^* in terms of the remaining $n - m$ variables; i.e., $x_i = \phi_i(x_{m+1}, \dots, x_n)$ for $i = 1, \dots, m$. Moreover, the functions ϕ_i are unique and continuously differentiable throughout the neighborhood of \mathbf{x}^* . Now let $\mathbf{x}' = (x_1, \dots, x_m)$ and $\mathbf{x}'' = (x_{m+1}, \dots, x_n)$. Then $\mathbf{x}' = (\phi_1(\mathbf{x}''), \dots, \phi_m(\mathbf{x}''))$ which we will denote by $\phi(\mathbf{x}'')$. Using this notation, problem (4.11) can be restated as

$$\min f(\mathbf{x}', \mathbf{x}'') = \min f(\phi(\mathbf{x}''), \mathbf{x}'') = \min \Phi(\mathbf{x}'') \quad (4.13)$$

where $\Phi(\mathbf{x}'') = (\phi(\mathbf{x}''), \mathbf{x}'')^T$. Since problem (4.13) must have a relative minimum at $\mathbf{x}''^* = (x_{m+1}^*, \dots, x_n^*)$, it follows from Theorem 4.2.1 that $\nabla \Phi(\mathbf{x}'') = 0$. But

$$\frac{\partial \Phi}{\partial x_j} = \sum_{i=m+1}^n \frac{\partial f}{\partial x_i} \frac{\partial x_i}{\partial x_j} + \sum_{i=1}^m \frac{\partial f}{\partial x_i} \frac{\partial \phi_i}{\partial x_j} = \frac{\partial f}{\partial x_j} + \left[\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_m} \right] \begin{bmatrix} \frac{\partial \phi_1}{\partial x_j} \\ \vdots \\ \frac{\partial \phi_m}{\partial x_j} \end{bmatrix}$$

Thus letting $\nabla f(\mathbf{x}') = [\partial f / \partial x_1, \dots, \partial f / \partial x_m]$, $\nabla f(\mathbf{x}'') = [\partial f / \partial x_{m+1}, \dots, \partial f / \partial x_n]$ and $\partial \phi / \partial \mathbf{x}''$ be an $m \times (n-m)$ matrix composed of columns of the form $[\partial \phi_1 / \partial x_j, \dots, \partial \phi_m / \partial x_j]^T$, it follows that

$$\mathbf{0} = \nabla \Phi(\mathbf{x}'') = \nabla f(\mathbf{x}'') + \nabla f(\mathbf{x}') \frac{\partial \phi}{\partial \mathbf{x}''} \quad (4.14)$$

must hold at \mathbf{x}''^* . Also, $\mathbf{0} = \mathbf{h}(\mathbf{x}) = \mathbf{h}(\phi(\mathbf{x}''), \mathbf{x}'')$. Hence, a similar set of calculations shows that

$$\mathbf{0} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}''} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}'} \frac{\partial \phi}{\partial \mathbf{x}''} \quad (4.15)$$

must hold at \mathbf{x}''^* , where $\partial \mathbf{h} / \partial \mathbf{x}'$ and $\partial \mathbf{h} / \partial \mathbf{x}''$ are the submatrices of the first m and last $n - m$ columns of (4.12), respectively. Now, from (4.15) we have

$$\frac{\partial \phi}{\partial \mathbf{x}''} = - \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}'} \right)^{-1} \frac{\partial \mathbf{h}}{\partial \mathbf{x}''}$$

so it follows from (4.14) that

$$\nabla f(\mathbf{x}'')^T - \boldsymbol{\lambda}^T \frac{\partial \mathbf{h}}{\partial \mathbf{x}'} = 0 \quad (4.16)$$

where

$$\boldsymbol{\lambda}^T = \nabla f(\mathbf{x}') \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}'} \right)^{-1}$$

at \mathbf{x}''^* . Rearranging gives

$$\nabla f(\mathbf{x}') - \boldsymbol{\lambda}^T \frac{\partial \mathbf{h}}{\partial \mathbf{x}'} = \mathbf{0} \quad (4.17)$$

Combining eqs. (4.16) and (4.17), and recalling the initial constraints of problem (4.11), we have the following first order necessary conditions:

$$\nabla f(\mathbf{x}^*) - \boldsymbol{\lambda}^T \nabla \mathbf{h}(\mathbf{x}^*) = \mathbf{0} \quad \text{and} \quad \mathbf{h}(\mathbf{x}^*) = \mathbf{0}$$

for a relative minimum to occur at \mathbf{x}^* , where $\nabla \mathbf{h}(\mathbf{x}^*)$ is an $m \times n$ matrix. Letting $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{x})$, these necessary conditions can be stated in the following manner.

Theorem 4.2.4 In problem (4.11), let f have a relative minimum at \mathbf{x}^* . If f and each component h_i of \mathbf{h} are twice continuously differentiable throughout a neighborhood of \mathbf{x}^* , and if the Jacobian matrix (4.12) has full row rank m , then there exists a $\boldsymbol{\lambda}^* \in R^m$ such that

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*)}{\partial \mathbf{x}} &= \nabla f(\mathbf{x}^*) - \boldsymbol{\lambda}^T \nabla \mathbf{h}(\mathbf{x}^*) = \mathbf{0} \\ \frac{\partial \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*)}{\partial \boldsymbol{\lambda}} &= -\mathbf{h}(\mathbf{x}^*) = \mathbf{0} \end{aligned} \quad (4.18)$$

The function \mathcal{L} is known as the *Lagrangian* and the components λ_i of $\boldsymbol{\lambda}$ are variously called *Lagrange multipliers* or dual variables. When the equality constraints are written with a nonzero right-hand-side value, say $\mathbf{b} \in R^m$, each Lagrange multiplier satisfies $\lambda_i = \partial f / \partial b_i$ at \mathbf{x}^* , and hence measures the sensitivity of the optimal value of $f(\mathbf{x})$ to changes in the constant b_i .

Second order necessary and sufficient conditions for a relative minimum can be developed in the same way as was done for the unconstrained case. It should be noted that the first order necessary conditions for a relative maximum of $f(\mathbf{x})$ subject to $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ are that there exists a $\boldsymbol{\lambda}^*$ such that

$$\nabla f(\mathbf{x}^*) + \boldsymbol{\lambda}^T \nabla \mathbf{h}(\mathbf{x}^*) = \mathbf{0} \quad \text{and} \quad \mathbf{h}(\mathbf{x}^*) = \mathbf{0} \quad (4.19)$$

Notice that these conditions can be obtained from $\partial \mathcal{L} / \partial \mathbf{x} = \mathbf{0}$ and $\partial \mathcal{L} / \partial \boldsymbol{\lambda} = \mathbf{0}$ by using either $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = -f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{x})$ or $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{x})$. Because the $\boldsymbol{\lambda}$ variables are unrestricted, if the point $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ satisfies (4.18) then the point $(\mathbf{x}^*, -\boldsymbol{\lambda}^*)$ will satisfy (4.19) so it is not possible to tell whether a maximum or minimum is at hand. Further testing is required.

Example 4.2.3 Find the relative extrema of the function $f(x_1, x_2) = 2x_1 x_2$ subject to $x_1^2 + x_2^2 = 1$.

Solution: Let $\mathcal{L}(x_1, x_2, \lambda) = 2x_1x_2 - \lambda(x_1^2 + x_2^2 - 1)$. Then condition (4.17) becomes

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial x_1} &= 2x_2 - 2x_1\lambda = 0 \\ \frac{\partial \mathcal{L}}{\partial x_2} &= 2x_1 - 2x_2\lambda = 0 \\ \frac{\partial \mathcal{L}}{\partial \lambda} &= x_1^2 + x_2^2 - 1 = 0\end{aligned}$$

giving three equations in three unknowns. Solving the first equation for λ we get $\lambda = x_2/x_1$ as long as $x_1 \neq 0$. Substituting this value into the second equation gives $x_1^2 = x_2^2$ which, from the third equation, implies that $2x_1^2 = 1$. Thus $x_1 = \pm\sqrt{2}/2$, $x_2 = \pm\sqrt{2}/2$ and $\lambda = +1$ or -1 , depending on the values of x_1 and x_2 . In fact, we have the following four possible solutions: $(\sqrt{2}/2, \sqrt{2}/2, 1)$, $(-\sqrt{2}/2, -\sqrt{2}/2, 1)$, $(\sqrt{2}/2, -\sqrt{2}/2, -1)$, and $(-\sqrt{2}/2, \sqrt{2}/2, -1)$ for (x_1, x_2, λ) . The case where $x_1 = 0$ can be ruled out because it implies that $x_2 = 0$ which would violate the third equation. By the same reasoning $x_2 \neq 0$.

The results so far are based only on first order necessary conditions for a relative extrema. Therefore, we only know that if optimal solutions exist, they must occur at one or more of these points. Because the constraint region $S = \{(x_1, x_2) : x_1^2 + x_2^2 = 1\}$ is closed and bounded the Weierstrass theorem implies that f has both a maximum and minimum over S . By inspection, we see that the minimum value of f is -1 and occurs at both $(-\sqrt{2}/2, \sqrt{2}/2)$ and $(\sqrt{2}/2, -\sqrt{2}/2)$. Similarly, we determine that the maximum occurs at both $(\sqrt{2}/2, \sqrt{2}/2)$ and $(-\sqrt{2}/2, -\sqrt{2}/2)$ yielding an objective function value of 1 .

Using arguments for classical calculus, we will now derive second order necessary and sufficient conditions for \mathbf{x}^* to be a local minimum of (4.11). It is assumed that $f, \mathbf{h} \in C^2$.

Theorem 4.2.5 (Second Order Necessary Conditions) Suppose that \mathbf{x}^* is a local minimum of $f(\mathbf{x})$ subject to $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ as well as a regular point of these constraints. Then there exists a vector $\boldsymbol{\lambda} \in R^m$ such that

$$\nabla f(\mathbf{x}^*) - \boldsymbol{\lambda}^T \nabla \mathbf{h}(\mathbf{x}^*) = \mathbf{0} \quad (4.20)$$

If we denote by T the tangent plane $T = \{\mathbf{y} : \nabla \mathbf{h}(\mathbf{x}^*)^T \mathbf{y} = \mathbf{0}\}$, then the matrix

$$\mathbf{L}(\mathbf{x}^*) \triangleq \mathbf{F}(\mathbf{x}^*) - \boldsymbol{\lambda}^T \mathbf{H}(\mathbf{x}^*)$$

is positive semidefinite on T ; that is, $\mathbf{y}^T \mathbf{L}(\mathbf{x}^*) \mathbf{y} \geq 0$ for all $\mathbf{y} \in T$.

Proof: Let \mathbf{y} be any vector in the tangent plane at \mathbf{x}^* and let $\mathbf{x}(\tau)$ be a twice differentiable curve on the constraint surface S passing through \mathbf{x}^* with derivative \mathbf{y}

at \mathbf{x}^* ; that is, $\mathbf{x}(0) = \mathbf{x}^*$, $\frac{d}{d\tau}\mathbf{x}(0) = \dot{\mathbf{x}}(0) = \mathbf{y}$, and $\mathbf{h}(\mathbf{x}(\tau)) = \mathbf{0}$ for $-a \leq \tau \leq a$ for some $a > 0$. From elementary calculus we have

$$\left. \frac{d^2}{d\tau^2} f(\mathbf{x}(\tau)) \right|_{\tau=0} \geq 0 \quad (4.21)$$

Taking the second derivative of $f(\mathbf{x}(\tau))$ with respect to τ at $\tau = 0$ gives

$$\left. \frac{d^2}{d\tau^2} f(\mathbf{x}(\tau)) \right|_{\tau=0} = \dot{\mathbf{x}}(0)^T \mathbf{F}(\mathbf{x}^*) \dot{\mathbf{x}}(0) + \nabla f(\mathbf{x}^*) \ddot{\mathbf{x}}(0) \quad (4.22)$$

Now, differentiating the relation $\lambda^T \mathbf{h}(\mathbf{x}(\tau)) = 0$ twice, we get

$$\dot{\mathbf{x}}(0)^T \lambda^T \mathbf{H}(\mathbf{x}^*) \dot{\mathbf{x}}(0) + \lambda^T \nabla \mathbf{h}(\mathbf{x}^*) \ddot{\mathbf{x}}(0) \quad (4.23)$$

Subtracting (4.23) from (4.22) and taking into account (4.20) and (4.21), yields the result

$$\left. \frac{d^2}{d\tau^2} f(\mathbf{x}(\tau)) \right|_{\tau=0} = \dot{\mathbf{x}}(0)^T \mathbf{L}(\mathbf{x}^*) \dot{\mathbf{x}}(0) \geq 0$$

Because $\dot{\mathbf{x}}(0)$ is an arbitrary point in T , the stated conclusion follows immediately. ■

The matrix $\mathbf{L}(\mathbf{x})$ is the Hessian of the Lagrangian and plays a central role in the development of algorithms. The following sufficiency conditions are also stated in terms of \mathbf{L} . For a proof, see [N2].

Theorem 4.2.6 (Second Order Sufficiency Conditions) Suppose there is a point \mathbf{x}^* satisfying $\mathbf{h}(\mathbf{x}^*) = \mathbf{0}$ and a $\lambda \in R^m$ such that $\nabla f(\mathbf{x}^*) - \lambda^T \nabla \mathbf{h}(\mathbf{x}^*) = \mathbf{0}$. Also, suppose that the matrix $\mathbf{L}(\mathbf{x}^*) = \mathbf{F}(\mathbf{x}^*) - \lambda^T \mathbf{H}(\mathbf{x}^*)$ is positive definite on $T = \{\mathbf{y} : \nabla \mathbf{h}(\mathbf{x}^*) \mathbf{y} = \mathbf{0}\}$, that is, for all $\mathbf{y} \in T$, $\mathbf{y} \neq \mathbf{0}$, we have $\mathbf{y}^T \mathbf{L}(\mathbf{x}^*) \mathbf{y} > 0$. Then \mathbf{x}^* is a strict (unique) local minimum of f subject to $\mathbf{h}(\mathbf{x}) = \mathbf{0}$.

Continuing with Example 4.2.3, we have the matrix $\mathbf{L}(\mathbf{x}^*) = \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ which is independent of \mathbf{x}^* . Also, $T = \{(y_1, y_2) : x_1^* y_1 + x_2^* y_2 = 0\}$. At either minimizing point, $\lambda = -1$ so $\mathbf{L} = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$, $T = \{(y_1, y_2) : y_1 = y_2\}$ and $\mathbf{y}^T \mathbf{L} \mathbf{y} = (y_1, y_2) \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = 8y_1^2 \geq 0$ which satisfies the second order necessary and sufficient conditions for a relative minimum. At either maximizing point, $\lambda = 1$, $\mathbf{L} = \begin{bmatrix} -2 & 2 \\ 2 & -2 \end{bmatrix}$, $T = \{(y_1, y_2) : y_1 = -y_2\}$ and $\mathbf{y}^T \mathbf{L} \mathbf{y} = -8y_1^2 \leq 0$ so \mathbf{L} is negative definite on the tangent plane T and we are at a relative maximum.

4.2.4 Inequality Constraints

The most general NLP model that we consider includes both equality and inequality constraints:

$$\min\{f(\mathbf{x}) : \mathbf{h}(\mathbf{x}) = \mathbf{0}, \mathbf{g}(\mathbf{x}) \geq \mathbf{0}\} \quad (4.24)$$

where $f : R^n \rightarrow R^1$, $\mathbf{h} : R^n \rightarrow R^m$, $\mathbf{g} : R^n \rightarrow R^q$, and all the functions are in C^2 . Although it is possible and sometimes convenient to treat variable bounds explicitly, we assume that they are included in the vector \mathbf{g} .

To derive first and second order optimality conditions for problem (4.24), it is necessary to suppose that the constraints satisfy certain regularity conditions or constraint qualifications. A full treatment of this topic can be found in [B16], [F5], or [F6]. The accompanying results are important from a theoretical point of view but less so from an algorithmic point of view. Consequently, we take a practical approach and simply generalize the methodology used in the above developments associated with the equality constrained problem (4.11).

Definition 4.2.4 Let \mathbf{x}^* be a point satisfying the constraints $\mathbf{h}(\mathbf{x}^*) = \mathbf{0}$, $\mathbf{g}(\mathbf{x}^*) \geq \mathbf{0}$ and let J be the set of indices j for which $g_j(\mathbf{x}^*) = 0$. Then \mathbf{x}^* is said to be a *regular point* of these constraints if the gradient vectors, $\nabla h_i(\mathbf{x}^*)$ ($1 \leq i \leq m$), $\nabla g_j(\mathbf{x}^*)$ ($j \in J$) are linearly independent.

This definition says that \mathbf{x}^* is a regular point if the gradients of the binding or active constraints are linearly independent. It leads to perhaps the most important result in differential optimization.

Theorem 4.2.7 (Kuhn-Tucker Conditions) Let \mathbf{x}^* be a relative minimum for problem (4.24) and suppose that \mathbf{x}^* is regular point for the constraints. Then there exists a vector $\boldsymbol{\lambda} \in R^m$ and a vector $\boldsymbol{\mu} \in R^q$ such that

$$\nabla f(\mathbf{x}^*) - \boldsymbol{\lambda}^T \nabla \mathbf{h}(\mathbf{x}^*) - \boldsymbol{\mu}^T \nabla \mathbf{g}(\mathbf{x}^*) = \mathbf{0} \quad (4.25a)$$

$$\boldsymbol{\mu}^T \mathbf{g}(\mathbf{x}^*) = 0 \quad (4.25b)$$

$$\boldsymbol{\mu} \geq \mathbf{0} \quad (4.25c)$$

$$\mathbf{h}(\mathbf{x}^*) = \mathbf{0}, \mathbf{g}(\mathbf{x}^*) \geq \mathbf{0} \quad (4.25d)$$

Proof: Constraint (4.25b) is called a complementarity condition; because $\boldsymbol{\mu} \geq \mathbf{0}$ and $\mathbf{g}(\mathbf{x}^*) \geq \mathbf{0}$, it is equivalent to saying that a component of $\boldsymbol{\mu}$ may be nonzero only if the corresponding constraint is active. Now, since \mathbf{x}^* is a relative minimum over the feasible region, it is also a relative minimum over the reduced feasible region defined by setting the active constraints to zero. Thus for the resulting equality constrained problem defined in a neighborhood of \mathbf{x}^* , there exists Lagrange multipliers as derived in Section 4.2.3. We can therefore conclude that (4.25a) holds with $\mu_j = 0$ if $g_j(\mathbf{x}^*) \neq 0$. This means that (4.25b) also holds.

It remains to be shown that $\mu \geq 0$. Suppose that $\mu_k < 0$ for some $k \in J$. Let S and T be the surface and tangent space, respectively, defined by all other active constraints at \mathbf{x}^* . By the regularity assumption, there is a \mathbf{y} such that $\mathbf{y} \in T$ and $\nabla g_k(\mathbf{x}^*)\mathbf{y} > 0$; i.e., the linear independence of the gradients of the active constraints and the fact that $\mathbf{y} \perp \{\nabla h_i, \nabla g_j; i \in \{1, \dots, m\}, j \in J \setminus \{k\}\}$ imply that $\nabla g_k(\mathbf{x}^*)\mathbf{y} \neq 0$ so we can always find a $\mathbf{y} \in T$ such that $\nabla g_k(\mathbf{x}^*)\mathbf{y} > 0$. Let $\mathbf{x}(\tau)$ be a curve on S passing through \mathbf{x}^* (at $\tau = 0$) with $\dot{\mathbf{x}}(0) = \mathbf{y}$. Then for small $\tau \geq 0$, $\mathbf{x}(\tau)$ is feasible and

$$\frac{d}{d\tau} f(\mathbf{x}(\tau)) \Big|_{\tau=0} = \nabla f(\mathbf{x}^*)\mathbf{y} - \lambda^T \nabla \mathbf{h}(\mathbf{x}^*)\mathbf{y} - \mu^T \nabla \mathbf{g}(\mathbf{x}^*)\mathbf{y} = 0 \quad (\text{from (4.25a)})$$

But the fact that $\mathbf{y} \in T$ implies that $\nabla f(\mathbf{x}^*)\mathbf{y} - \mu_k \nabla g_k(\mathbf{x}^*)\mathbf{y} = 0$, or $\nabla f(\mathbf{x}^*)\mathbf{y} = \mu_k \nabla g_k(\mathbf{x}^*)\mathbf{y} < 0$ which contradicts the optimality of \mathbf{x}^* . ■

Second order necessary and sufficient conditions for problems with inequality constraints are derived by similarly considering only the equality constrained problem that results from the active constraints at a candidate solution \mathbf{x}^* . The appropriate tangent plane is the one corresponding to the active constraints.

Theorem 4.2.8 (Second Order Necessary Conditions) Suppose that the functions $f, \mathbf{h}, \mathbf{g} \in C^2$ and that \mathbf{x}^* is a regular point of the constraints in (4.24). If \mathbf{x}^* is a relative minimum of (4.24), then there exists a vector $\lambda \in R^m$ and a vector $\mu \in R^q$, $\mu \geq 0$ such that (4.25a)–(4.25d) holds and

$$\mathbf{L}(\mathbf{x}^*) = \mathbf{F}(\mathbf{x}^*) - \lambda^T \mathbf{H}(\mathbf{x}^*) - \mu^T \mathbf{G}(\mathbf{x}^*)$$

is positive semidefinite on the tangent subspace of the active constraints at \mathbf{x}^* .

Proof: If \mathbf{x}^* is a relative minimum over the constraints in (4.24), it is also a relative minimum for the problem with the active constraints taken as equalities. ■

By analogy with the unconstrained situation, one can guess that the second order sufficiency conditions for problem (4.24) require that $\mathbf{L}(\mathbf{x}^*)$ be positive definite on the tangent plane T . This is indeed the result in most situations; however, if there are degenerate inequality constraints (that is, active constraints with associated Lagrange multipliers at zero), we must require $\mathbf{L}(\mathbf{x}^*)$ to be positive definite on a subspace that is larger than T .

Theorem 4.2.9 (Second Order Sufficient Conditions) Let $f, \mathbf{h}, \mathbf{g} \in C^2$. Sufficient conditions that a point \mathbf{x}^* satisfying the constraints in (4.24) is a strict relative minimum of (4.24) is that there exist $\lambda \in R^m$, $\mu \in R^q$ such that the constraints in (4.25a)–(4.25d) hold and that the Hessian matrix $\mathbf{L}(\mathbf{x}^*) = \mathbf{F}(\mathbf{x}^*) - \lambda^T \mathbf{H}(\mathbf{x}^*) - \mu^T \mathbf{G}(\mathbf{x}^*)$ is positive definite on the subspace

$$T' = \{\mathbf{y} : \nabla \mathbf{h}(\mathbf{x}^*)\mathbf{y} = 0, \nabla g_j(\mathbf{x}^*)\mathbf{y} = 0 \text{ for all } j \in J'\}$$

where $J' = \{j : g_j(\mathbf{x}^*) = 0, \mu_j > 0\}$.

Proof [L7]: We prove the result by contradiction. Assume that \mathbf{x}^* is not a strict relative minimum. Let $\{\mathbf{y}^k\}$ be a sequence of feasible points converging to \mathbf{x}^* such that $f(\mathbf{y}^k) \leq f(\mathbf{x}^*)$ and write each \mathbf{y}^k in the form $\mathbf{y}^k = \mathbf{x}^* + \delta_k \mathbf{s}^k$ with $\|\mathbf{s}^k\| = 1$, $\delta_k > 0$. We may assume that $\delta_k \rightarrow 0$ and $\mathbf{s}^k \rightarrow \mathbf{s}^*$ which implies that $\nabla f(\mathbf{x}^*) \mathbf{s}^* \leq 0$, and for each $i = 1, \dots, m$, $\nabla h_i(\mathbf{x}^*) \mathbf{s}^* = 0$. Also for each active constraint g_j we have $g_j(\mathbf{y}^k) - g_j(\mathbf{x}^*) \geq 0$, and hence $\nabla g_j(\mathbf{x}^*) \mathbf{s}^* \geq 0$. These conditions follow from a first order Taylor series expansion of f , h_i and g_j around the point \mathbf{x}^* and from the fact that the sequence $\{\mathbf{y}^k\}$ is feasible.

If $\nabla g_j(\mathbf{x}^*) \mathbf{s}^* = 0$ for all $j \in J'$, then the proof is equivalent to that given for Theorem 4.2.8. If $\nabla g_j(\mathbf{x}^*) \mathbf{s}^* > 0$ for at least one $j \in J'$, then

$$0 \geq \nabla f(\mathbf{x}^*) \mathbf{s}^* = \boldsymbol{\lambda}^T \nabla h(\mathbf{x}^*) \mathbf{s}^* + \boldsymbol{\mu}^T \nabla g(\mathbf{x}^*) \mathbf{s}^* > 0$$

which is a contradiction. ■

If all the active inequality constraints have strictly positive Lagrange multipliers so there is no degeneracy, then the set J' includes all the active inequalities. In this case the sufficient condition is that the Hessian of the Lagrangian be positive definite on T , the tangent plane of the active constraints. Unfortunately, it is usually not possible to test whether or not the second order optimality conditions are satisfied for all but the simplest of problems. Commercial NLP codes are designed to find a point that satisfies the Kuhn-Tucker conditions (4.25a)–(4.25d) to within some acceptable tolerance only. It is left to the user to assess the nature of the solution.

Example 4.2.4 Consider the problem

$$\begin{array}{ll} \min & f = 2x_1^2 + x_2^2 + \frac{1}{5}x_3^2 + x_1x_3 - x_1x_2 + x_1 - \frac{1}{2}x_3 \\ \text{subject to} & h_1 = x_1 + x_2 = 3 \\ & g_1 = -x_1^2 - x_2^2 + x_3 \geq 0 \\ & g_2 = -x_1 - x_2 - 2x_3 \geq -16 \\ & g_3 = x_1 \geq 0 \\ & g_4 = x_2 \geq 0 \\ & g_5 = x_3 \geq 0 \end{array}$$

The first order necessary conditions, excluding the constraints, are

$$\begin{pmatrix} 4x_1 + x_3 - x_2 + 1 \\ 2x_2 - x_1 \\ \frac{2}{5}x_3 + x_1 - \frac{1}{2} \end{pmatrix} - \lambda_1 \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} - \mu_1 \begin{pmatrix} -2x_1 \\ -2x_2 \\ 1 \end{pmatrix} - \mu_2 \begin{pmatrix} -1 \\ -1 \\ -2 \end{pmatrix}$$

$$\begin{aligned}
-\mu_3 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} - \mu_4 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} - \mu_5 \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} &= 0 \\
\mu_1(-x_1^2 - x_2^2 + x_3) &= 0 \\
\mu_2(-x_1 - x_2 - 2x_3 + 16) &= 0 \\
\mu_3(x_1) = \mu_4(x_2) = \mu_5(x_3) &= 0 \\
\mu_j \geq 0, j &= 1, \dots, 5
\end{aligned}$$

To find a solution we must check various combinations of active constraints, compute the values of the corresponding multipliers and variables, and see if the results are feasible. In this problem there are 5 inequality constraints so there are $2^5 = 32$ combinations that, at least theoretically, have to be examined. Most of these are seen to be infeasible. For example, let us assume that only g_2 is active so $\mu_j = 0$, for all $j \neq 2$. Given $h_1 = x_1 + x_2 = 3$, this means that $x_3 = 13/2$. The third stationarity condition then yields $(2/5)(13/2) + x_1 - (1/2) + 2\mu_2 = 0$ implying that $\mu_2 < 0$ which is not feasible. Now consider the case where only g_1 is active. One possible solution is $\bar{\mathbf{x}} = (1, 2, 5)$. The third stationarity condition gives $\mu_1 = 5/2$, and either the first or second stationarity conditions gives $\lambda_1 = 13$. Thus $\bar{\mathbf{x}}$ with corresponding λ and μ values satisfies the first order necessary conditions for a relative minimum.

The next step is to check the second order conditions by examining the matrix $L(\bar{\mathbf{x}})$:

$$\begin{bmatrix} 4 & -1 & 1 \\ -1 & 2 & 0 \\ 1 & 0 & 2/5 \end{bmatrix} + \frac{5}{2} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 9 & -1 & 1 \\ -1 & 7 & 0 \\ 1 & 0 & 2/5 \end{bmatrix}$$

From Proposition 4.1.1 we see that this matrix is positive definite on R^3 so it must be positive definite on the required tangent subspace. Therefore, $\bar{\mathbf{x}} = (1, 2, 5)$ is a unique local minimum.

4.2.5 Convex Optimization

A convex programming problem is one of minimizing a convex function (or maximizing a concave function) over a convex feasible region. The main theorem of convex optimization is the following:

Theorem 4.2.10 Any local minimum of a convex programming problem is a global minimum.

Note that the problem may have a number of points at which the global minimum occurs but the set of all such points is convex. No distinct (i.e., separated) local minima may exist with different function values. A linear program (2.1) is the simplest case of a convex program. The following results help us characterize and recognize less obvious instances.

Proposition 4.2.2 If $g(\mathbf{x})$ is convex, then the set $\{\mathbf{x} : g(\mathbf{x}) \leq b\}$ is convex for all scalars b .

Proposition 4.2.3 The intersection of any number of convex sets is convex.

As a consequence of Propositions 4.2.2 and 4.2.3, we have: If the functions f and g_i are convex, then

$$\min f(\mathbf{x}) \quad (4.26a)$$

$$\text{subject to } \mathbf{a}_i \mathbf{x} = b_i \quad i = 1, \dots, m \quad (4.26b)$$

$$g_i(\mathbf{x}) \leq d_i \quad i = 1, \dots, q \quad (4.26c)$$

is a convex programming problem, where \mathbf{a}_i is an n -dimensional row vector.

This result implies that convex programs can be identified by determining if the objective and constraint functions of the problem are convex. It is thus of interest to better characterize convex functions. The main results in this regard are given below.

Proposition 4.2.4 If $f(\mathbf{x})$ has continuous first and second partial derivatives, the following statements are equivalent:

- (a) $f(\mathbf{x})$ is convex.
- (b) $f(\mathbf{x}^2) \geq f(\mathbf{x}^1) + \nabla f(\mathbf{x}^1)(\mathbf{x}^2 - \mathbf{x}^1)$ for any two points $\mathbf{x}^1, \mathbf{x}^2$.
- (c) The matrix of second partial derivatives of $f(\mathbf{x})$, denoted by $\nabla^2 f(\mathbf{x})$, is positive semidefinite for all points \mathbf{x} .

Part (b) of the proposition states that the function evaluated at any point, call it \mathbf{x}^2 , never lies below its tangent plane passing through any other point \mathbf{x}^1 .

Proposition 4.2.5 A positive semidefinite quadratic form is convex.

This result is a direct consequence of part (c) of Proposition 4.2.4.

Proposition 4.2.6 A positive linear combination of convex functions is convex.

Proposition 4.2.7 A function $f(\mathbf{x})$ is convex if and only if the 1-dimensional function $g(\alpha) = f(\mathbf{x} + \alpha \mathbf{d})$ is convex for all fixed \mathbf{x} and \mathbf{d} .

Because $f(\mathbf{x} + \alpha\mathbf{d})$ is the function evaluated at points along a line in direction \mathbf{d} passing through the point \mathbf{x} , Proposition 4.2.7 says that a convex function is convex along any line. This affords a means of telling whether a given function of n variables is convex. If any line in n -dimensional space can be found along which $g(\alpha)$ is not convex, neither is $f(\mathbf{x})$.

Mixed problems: The principal theoretical difficulty in dealing with the general NLP given in (4.1) is that if $h(\mathbf{x})$ is nonlinear, the set $S = \{\mathbf{x} : h(\mathbf{x}) = \mathbf{b}\}$ is rarely if ever convex. This is evident geometrically, since most nonlinear functions have graphs that are curved surfaces. Hence, the set S will usually be a curved surface also, and the line segment joining any two points on this surface will generally not lie on the surface.

As a consequence of the above, problem (4.1) may not be a convex programming problem in the variables x_1, \dots, x_n if any of the functions $h_i(\mathbf{x})$ are nonlinear. This of course, does not preclude efficient solution of such problems, but it does make it more difficult to guarantee the absence of local optima different from the global and to generate sharp theoretical results. In many cases the equality constraints may be used to eliminate some of the variables, leaving a problem with only inequality constraints and fewer variables. Even if the equalities are difficult to solve analytically it may still be worthwhile solving them numerically.

Role of convexity: Although convexity is desirable, many real-world problems turn out to be nonconvex. In addition, there is no simple way to test an NLP for convexity because there is no simple way to test a nonlinear function that is not a familiar form for this property. Nevertheless, results obtained under convexity assumptions can often give insight into the properties of more general problems. Sometimes such results may even be carried over to nonconvex problems albeit in a weaker form (see discussion of quasi- and pseudo-convexity in [B16]. For example, it is usually impossible to prove that a given algorithm will find the global minimum of an NLP unless the problem is convex. In the nonconvex case, however, many such algorithms will at least find a local minimum.

4.3 SEARCH TECHNIQUES FOR UNCONSTRAINED PROBLEMS

The basic approach to solving almost all mathematical programs is to select an initial point and a direction in which the objective function is improving, and then move in that direction until either an extremum is reached or feasibility is violated. In either case, a new direction is computed and the process is repeated. A check for convergence is made at the end of each iteration. At the heart of this approach is a one-dimensional search where the length of the move, called the step size, is determined. That is, given

a point \mathbf{x}^k and a direction \mathbf{d}^k , the aim is to find an optimal step size t_k that moves us to the next point $\mathbf{x}^{k+1} = \mathbf{x}^k + t_k \mathbf{d}^k$.

4.3.1 One-Dimensional Linear Search Techniques

If $\theta(\mathbf{x})$ is the function to be minimized, then at each iteration of a *line search algorithm*, we are faced with the subproblem of finding the value t^* that minimizes $f(t) \triangleq \theta(\mathbf{x} + t\mathbf{d})$. If θ is not differentiable, then neither is f . If θ is differentiable, then $f'(t) = \nabla\theta(\mathbf{x} + t\mathbf{d})\mathbf{d}$ so to find a point t such that $f'(t) = 0$, we have to solve the equation $\nabla\theta(\mathbf{x} + t\mathbf{d})\mathbf{d} = 0$ which is usually nonlinear in t . But even if t satisfies $f'(t) = 0$ it is not necessarily a minimum; it may be a maximum or a saddle point. For these reasons, it is best to use numerical techniques for minimizing the function f .

Out of practical considerations we assume that $t \in [a, b]$, the interval of uncertainty. This leads to the one-dimensional problem

$$\min\{f(t) : t \in [a, b]\} \quad (4.27)$$

For simplicity it will also be assumed that f is continuous and *unimodal* in the interval $[a, b]$ implying that f has a single minimum t^* ; that is, for $t \in [a, b]$ and $f(t) \neq f(t^*)$, f is strictly decreasing when $t < t^*$ and strictly increasing when $t > t^*$ (see Fig. 4.5). Unimodal functions are sometimes called *strictly quasiconvex* functions [B16]. During a search procedure, if we could exclude portions of $[a, b]$ that do not contain the minimum, then the interval of uncertainty is reduced. The following theorem shows that it is possible to obtain a reduction by evaluating two points within the interval.

Theorem 4.3.1 Let $f : R^1 \rightarrow R^1$ be continuous and unimodal over the interval $[a, b]$. Let $t_1, t_2 \in [a, b]$ such that $t_1 < t_2$. If $f(t_1) > f(t_2)$, then $f(\tau) > f(t_2)$ for all $\tau \in [a, t_1]$. If $f(t_1) \leq f(t_2)$, then $f(\tau) \geq f(t_1)$ for all $\tau \in (t_2, b]$.

The two cases are illustrated in Fig. 4.6. Of course, if $f(t_1) = f(t_2)$, then $t^* \in [t_1, t_2]$, a fact that should be taken into account in any algorithmic implementation. A line search algorithm is one that systematically evaluates f , eliminating subintervals of $[a, b]$ until the location of the minimum point t^* is determined to a desired level of accuracy. We now present a few of the most popular approaches to solving problem (4.27).

Golden section: At a general iteration of the golden section method, let the interval of uncertainty be $[a_k, b_k]$. By Theorem 4.3.1 the new interval of uncertainty $[a_{k+1}, b_{k+1}]$ is given by $[t_{1k}, b_k]$ if $f(t_{1k}) > f(t_{2k})$ and by $[a_k, t_{2k}]$ if $f(t_{1k}) \leq f(t_{2k})$. The points t_{1k} and t_{2k} are selected such that

1. The length of the new interval of uncertainty $b_{k+1} - a_{k+1}$ does not depend upon the outcome of the k th iteration; that is, on whether $f(t_{1k}) > f(t_{2k})$

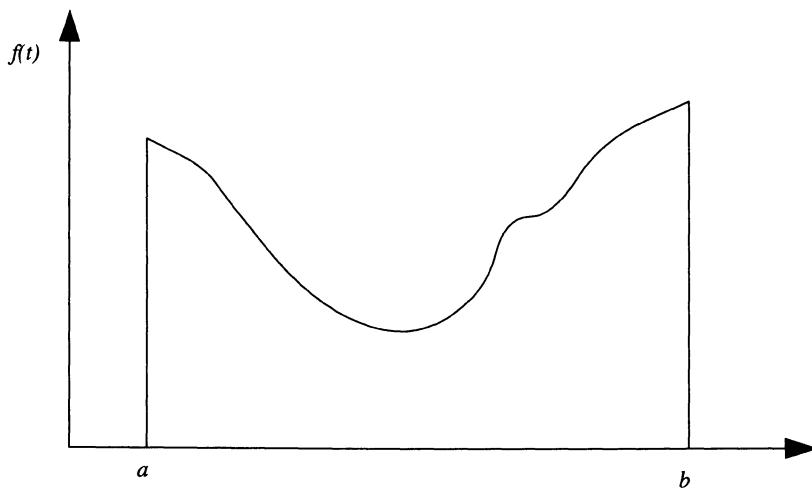


Figure 4.5 Example of unimodal function

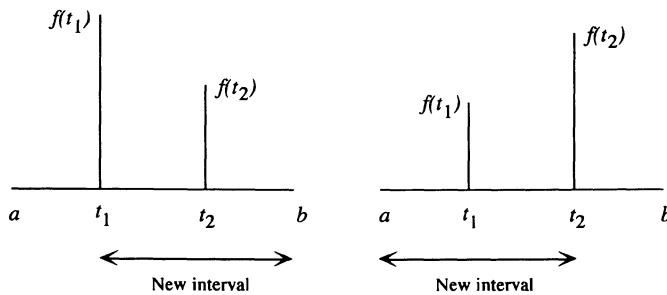


Figure 4.6 Reducing the interval of uncertainty

or $f(t_{1k}) \leq f(t_{2k})$. Therefore, we must have $b_{k+1} - t_{1k} = t_{2k} - a_{k+1}$ so if t_{1k} is of the form

$$t_{1k} = a_k + (1 - \alpha)(b_k - a_k) \quad (4.28)$$

where $\alpha \in (0, 1)$, then t_{2k} must be of the form

$$t_{2k} = a_k + \alpha(b_k - a_k) \quad (4.29)$$

so that

$$b_{k+1} - a_{k+1} = \alpha(b_k - a_k)$$

2. As $t_{1,k+1}$ and $t_{2,k+1}$ are selected for the purpose of a new iteration, either $t_{1,k+1}$ coincides with t_{2k} or $t_{2,k+1}$ coincides with t_{1k} . If this could be realized,

then during iteration $k + 1$, only one extra observation is needed. To see this, consider Fig. 4.7 and the following two cases.

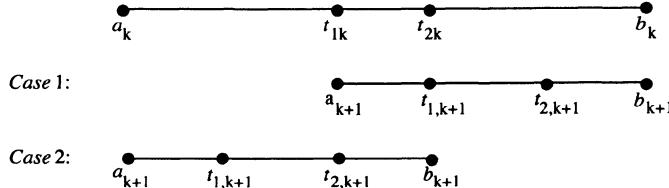


Figure 4.7 Illustration of golden section rule

Case 1: $f(t_{1k}) > f(t_{2k})$

In this case, $a_{k+1} = t_{1k}$ and $b_{k+1} = b_k$. To satisfy $t_{1,k+1} = t_{2k}$, and applying (4.28) with k replaced by $k + 1$, we get

$$t_{2k} = t_{1,k+1} = a_{k+1} + (1 - \alpha)(b_{k+1} - a_{k+1}) = t_{1k} + (1 - \alpha)(b_k - t_{1k})$$

Substituting the expressions for t_{1k} and t_{2k} from (4.28) and (4.29) into the above equation gives $\alpha^2 + \alpha - 1 = 0$.

Case 2: $f(t_{1k}) \leq f(t_{2k})$

In this case, $a_{k+1} = a_k$ and $b_{k+1} = t_{2k}$. To satisfy $t_{2,k+1} = t_{1k}$, and applying (4.29) with k replaced by $k + 1$, we get

$$t_{1k} = t_{2,k+1} = a_{k+1} + \alpha(b_{k+1} - a_{k+1}) = a_k + \alpha(t_{2k} - a_k)$$

In light of (4.28) and (4.29), the above equation similarly gives $\alpha^2 + \alpha - 1 = 0$.

The roots of the equation $\alpha^2 + \alpha - 1 = 0$ are approximately $\alpha = 0.618$ and $\alpha = -1.618$. Because α must be in the interval $(0, 1)$ we have $\alpha = 0.618$ which is known as the *golden section ratio*. To summarize, if at iteration k , t_{1k} and t_{2k} are chosen according to (4.28) and (4.29), where $\alpha = 0.618$, then the interval of uncertainty is reduced by a factor of 0.618. At the first iteration, two readings are needed at t_{1k} and t_{2k} , but at each subsequent iteration, only one evaluation is needed since either $t_{1,k+1} = t_{2k}$ or $t_{2,k+1} = t_{1k}$.

Fibonacci search: If only n evaluations are to be made in the original interval of uncertainty, the Fibonacci search method provides the optimal locations so the final interval of uncertainty is minimal. The procedure is based on the Fibonacci sequence $\{F_\nu\}$ defined as follows:

$$F_{\nu+1} = F_\nu + F_{\nu-1}, \quad \nu = 1, 2, \dots \quad (4.30)$$

$$F_0 = F_1 = 1$$

At iteration k , suppose that the interval of uncertainty is $[a_k, b_k]$. Consider the two points t_{1k} and t_{2k} given below, where n is the total number of function evaluations planned.

$$t_{1k} = a_k + \frac{F_{n-k-1}}{F_{n-k+1}}(b_k - a_k), \quad k = 1, \dots, n-1 \quad (4.31)$$

$$t_{2k} = a_k + \frac{F_{n-k}}{F_{n-k+1}}(b_k - a_k), \quad k = 1, \dots, n-1 \quad (4.32)$$

By Theorem 4.3.1 the new interval of uncertainty $[a_{k+1}, b_{k+1}]$ is given by $[t_{1k}, b_k]$ if $f(t_{1k}) > f(t_{2k})$ and by $[a_k, t_{2k}]$ if $f(t_{1k}) \leq f(t_{2k})$. In the former case, noting (4.31) and letting $\nu = n - k$ in (4.30), we get

$$b_{k+1} - a_{k+1} = b_k - t_{1k} = b_k - a_k - \frac{F_{n-k-1}}{F_{n-k+1}}(b_k - a_k) = \frac{F_{n-k}}{F_{n-k+1}}(b_k - a_k) \quad (4.33)$$

In the latter case, noting (4.32) we get

$$b_{k+1} - a_{k+1} = t_{2k} - a_k = \frac{F_{n-k}}{F_{n-k+1}}(b_k - a_k) \quad (4.34)$$

so in either instance, the interval of uncertainty is reduced by a factor of F_{n-k}/F_{n-k+1} . After the k th iteration, the width of uncertainty is $w_k = (F_{n-k+1}/F_n)(b_1 - a_1)$. It is interesting to note that as $n \rightarrow \infty$, $w_k/w_{k+1} \rightarrow 0.618$, the golden section ratio.

At iteration $k+1$ it can be shown that either $t_{1,k+1} = t_{2k}$ or $t_{2,k+1} = t_{1k}$, so that only one functional evaluation is needed. This means that at the end of iteration $n-2$, we have completed $n-1$ functional evaluations. Further, for $k = n-1$, it follows from (4.33) and (4.34) that $t_{1,n-1} = t_{2,n-1} = \frac{1}{2}(a_{n-1} + b_{n-1})$. Now, since either $t_{1,n-1} = t_{2,n-2}$ or $t_{2,n-1} = t_{1,n-2}$, theoretically no new observations are to be taken; however, to further reduce the interval of uncertainty, the last observation is placed slightly to the right or to the left of the midpoint $t_{1,n-1} = t_{2,n-1}$ so that $\frac{1}{2}(a_{n-1} + b_{n-1})$ is the length of the final interval of uncertainty. In selecting n , we note that $w_n = (b_1 - a_1)/F_n$ so n should be chosen to reflect this accuracy.

Newton's method: Suppose that $f : R^1 \rightarrow R^1$ as well as $f \in C^2$. In approaching problem (4.27), also suppose that at a point t_k where a measurement is made it is possible to evaluate three numbers $f(t_k)$, $f'(t_k)$ and $f''(t_k)$. This means that it is possible to construct a quadratic function q which agrees with f up to second derivatives at t_k . Let

$$q(t) = f(t_k) + f'(t_k)(t - t_k) + \frac{1}{2}f''(t_k)(t - t_k)^2 \quad (4.35)$$

We may then calculate an estimate t_{k+1} of the minimum point of f by finding the point where the derivative of q vanishes. Thus setting

$$0 = q'(t_{k+1}) = f'(t_k) + f''(t_k)(t_{k+1} - t_k)$$

we find

$$t_{k+1} = t_k - \frac{f'(t_k)}{f''(t_k)} \quad (4.36)$$

This process can then be repeated until some convergence criterion is met. We note immediately that the new point t_{k+1} obtained from (4.36) does not depend on the value of $f(t_k)$. Newton's method can more simply be viewed as a technique for iteratively solving equations of the form $g(t) = 0$ where $g(t) \triangleq f'(t)$ when applied to the line search problem. In this notation, we have $t_{k+1} = t_k - g(t_k)/g'(t_k)$. The following theorem gives sufficient conditions under which the method will converge to a stationary point.

Theorem 4.3.2 Let the function $g(t) = f'(t)$ have a continuous second derivative and let t^* satisfy $g(t^*) = 0, g'(t^*) \neq 0$. Then if t_1 is sufficiently close to t^* , the sequence $\{t_k\}_{k=1}^\infty$ generated by Newton's method (4.36) converges to t^* with an order of convergence of at least 2.

Method of false position: Newton's method relies on fitting a quadratic with information obtained from a single point and by requiring the function f to be twice differentiable. By using more points, less information is required at each of them. In particular, it is possible to fit a quadratic using $f(t_k), f'(t_k)$ and $f'(t_{k-1})$; i.e.,

$$q(t) = f(t_k) + f'(t_k)(t - t_k) + \frac{f'(t_{k-1}) - f'(t_k)}{t_{k-1} - t_k} \frac{(t - t_k)^2}{2}$$

An estimate t_{k+1} can now be determined by finding the point where the derivative of q vanishes. This lead to

$$t_{k+1} = t_k - \left[\frac{t_{k-1} - t_k}{f'(t_{k-1}) - f'(t_k)} \right] \quad (4.37)$$

Comparing this formula with that of Newton's method (4.36), we see again that the value $f(t_k)$ does not enter, implying that our fit could have been passed through either $f(t_k)$ or $f(t_{k-1})$. Also the formula can be regarded as an approximation to Newton's method where the second derivative is replaced by the difference of two first derivatives.

The next theorem gives sufficient conditions for the method of false position to converge, and somewhat surprisingly asserts that the order of convergence is the reciprocal of the golden section ratio.

Theorem 4.3.3 Let $g(t) = f'(t)$ have continuous second derivatives and suppose t^* is such that $g(t^*) = 0, g'(t^*) \neq 0$. Then for t_1 sufficiently close to t^* , the sequence $\{t_k\}_{k=1}^\infty$ generated by the method of false position (4.37) converges to t^* with order 1.618.

4.3.2 Multidimensional Search Techniques

We now turn to a consideration of methods for the unconstrained minimization of functions defined over R^n . As we have said a necessary condition for a differentiable function f to possess a minimum at $\mathbf{x}^* \in R^n$ is that $\nabla f(\mathbf{x}^*) = \mathbf{0}$. Such a point is called a *stationary point*. Thus we are faced with a problem of finding a solution to the system of nonlinear equations $\nabla f(\mathbf{x}) = \mathbf{0}$. A principal aspect of any procedure for solving these equations involves executing a sequence of line searches of the type just discussed; however, we note that a stationary point may be a local minimum, local maximum, or a saddle point. For f twice continuously differentiable, sufficient conditions for \mathbf{x}^* to be a local minimum are $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\nabla^2 f(\mathbf{x}^*)$ positive definite.

General Descent Algorithm

1. Start with an initial point \mathbf{x}^0 . Set the iteration counter k to 0.
2. Choose a descent direction \mathbf{d}^k .
3. Perform a line search to choose a step size t_k such that
$$\phi_k(t_k) = f(\mathbf{x}^k + t_k \mathbf{d}^k) < \phi_k(t_{k-1})$$
4. Set $\mathbf{x}^{k+1} = \mathbf{x}^k + t_k \mathbf{d}^k$.
5. Evaluate convergence criteria. If satisfied, stop; otherwise put $k \leftarrow k + 1$ and go to Step 2.

An exact line search is one which chooses t_k as the first local minimum of $\phi_k(t_k)$ at Step 3, i.e., the one with the smallest t value. Finding this minimum to high accuracy is overly time consuming so modern NLP algorithms use a variety of inexact line search procedures often involving polynomial fits, as in the method of false position. With regard to termination, stationarity is commonly used as one of several criteria in Step 5. In this case, we stop when the gradient is small. However, if f is scaled by multiplying it by a positive scale factor c , then $\nabla(c f(\mathbf{x})) = c \nabla f(\mathbf{x})$ so what constitutes “small” may be arbitrary. As a consequence, it is necessary to add other criteria, such as

$$\frac{|f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)|}{1 + |f(\mathbf{x}^k)|} < \varepsilon \quad \text{for } s \text{ consecutive values of } k$$

In this case, the iterations stop whenever the fractional change in the objective function is less than a user-defined tolerance $\varepsilon > 0$ for s consecutive cycles. The NLP code GRG2 (see [L1]), for example, uses $\varepsilon = 10^{-4}$ and $s = 3$ as default values.

The remaining issue concerning the general descent algorithm centers on the determination of the direction \mathbf{d}^k at Step 2. Several procedures are discussed below.

Steepest descent: The oldest method for choosing a search direction is known as steepest descent and sets \mathbf{d}^k to the negative gradient; i.e., $\mathbf{d}^k = -\nabla f(\mathbf{x}^k)$. The algorithm then generates the sequence $\{\mathbf{x}^k\}_{k=1}^\infty$ given by

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \nabla f(\mathbf{x}^k) \quad (4.38)$$

where t_k is the smallest $t \geq 0$ such that

$$f(\mathbf{x}^k - t_k \nabla f(\mathbf{x}^k)) = \min_{t \geq 0} f(\mathbf{x}^k - t \nabla f(\mathbf{x}^k))$$

assuming an exact one-dimensional line search is performed. This leads to the following result.

Theorem 4.3.4 Suppose f has continuous partial derivatives and that $\{\mathbf{x} : f(\mathbf{x}) \leq f(\mathbf{x}^1)\}$ is compact. Then any limit point of the sequence generated by the steepest descent algorithm (4.38) is a stationary point.

The difficulty with steepest descent is that it accompanied by excessive zigzagging when the isovalue contours of the function to be minimized differ significantly from concentric circles. This is highly inefficient. If t_k minimizes $\phi_k(t_k)$ then $\phi'_k(t_k) = \nabla f(\mathbf{x}^k + t_k \mathbf{d}^k) \mathbf{d}^k = 0$. If $\mathbf{d}^k = -\nabla f(\mathbf{x}^k)$ then the latter equality indicates that successive search directions are orthogonal, hence zigzagging for eccentric isovalue contours.

Newton's method: The steepest descent algorithm relies on first order approximations to the function f at the points \mathbf{x}^k . For a highly nonlinear function with continuous second partials, it is preferable to use second order approximations. In addition to tracking more closely the curvature of f , the second order approximations have better convergence properties. This is the idea behind Newton's method for functions of more than one variable. Using the second order Taylor series approximation

$$f(\mathbf{x}) \approx q(\mathbf{x}) = f(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^k) \mathbf{F}(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k)$$

where $\mathbf{F}(\mathbf{x}^k)$ is the $n \times n$ Hessian matrix, the point \mathbf{x}^{k+1} is chosen to minimize the approximation. These points must satisfy $\nabla q(\mathbf{x}^{k+1}) = \mathbf{0}$ or $\nabla f(\mathbf{x}^k) + (\mathbf{x}^{k+1} - \mathbf{x}^k) \mathbf{F}(\mathbf{x}^k) = \mathbf{0}$. This is a linear system with zero, one, or an infinite number of solutions.

Newton's method cannot be expected to converge to a stationary point. It is intended for use primarily in regions of R^n close to a strict local minimum of f . In such a region, it can be shown that the function f is strictly convex, implying that the Hessian matrix is positive definite and invertible. The descent algorithm is given by

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \nabla f(\mathbf{x}^k) [\mathbf{F}(\mathbf{x}^k)]^{-1} \quad (4.39)$$

Comparing the optimal gradient descent algorithm (4.38) to (4.39), we see that they are both special cases of the general algorithm

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \nabla f(\mathbf{x}^k) \mathbf{M}_k$$

for suitable matrices \mathbf{M}_k . In regions distant from a local minimum, the step can be chosen by selecting t_k as we did for the steepest descent algorithm where $\mathbf{M}_k = \mathbf{I}$. Near a local minimum, we take $t_k = 1$ and $\mathbf{M}_k = [\mathbf{F}(\mathbf{x}^k)]^{-1}$ to obtain (4.39). Global convergence to a stationary point can be guaranteed by looking at the eigenvalues of $\mathbf{F}(\mathbf{x}^k)$ in the following manner. At each point \mathbf{x}^k , calculate the eigenvalues of $\mathbf{F}(\mathbf{x}^k)$ and let ε_k be the smallest nonnegative scalar such that the matrix $\varepsilon_k \mathbf{I} + \mathbf{F}(\mathbf{x}^k)$ has eigenvalues greater than or equal to a preselected $\delta > 0$. This implies that the matrix is positive definite. The descent step is given by

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \nabla f(\mathbf{x}^k) [\varepsilon_k \mathbf{I} + \mathbf{F}(\mathbf{x}^k)]^{-1}$$

where $t_k \geq 0$ is chosen to minimize the quantity

$$f(\mathbf{x}^k - t_k \nabla f(\mathbf{x}^k)) [\varepsilon_k \mathbf{I} + \mathbf{F}(\mathbf{x}^k)]^{-1}$$

If the algorithm enters a region near a strict local minimum where the smallest eigenvalue of $\mathbf{F}(\mathbf{x}^k)$ is as great as δ , then this algorithm reduces to Newton's method. Outside of such regions, the function is made to look positive definite. Selection of the proper $\delta > 0$ is an art because a small value can produce nearly singular matrices, whereas a large value can delay or preclude convergence to a stationary point by Newton's method.

An inherent difficulty with the aforementioned second order methods is the necessity of calculating and inverting the Hessian matrix \mathbf{F} at each step. Procedures have been devised for approximating the Hessian and its inverse resulting in faster iterative schemes for finding a minimum of f . These are known as *conjugate direction methods* and are now discussed.

Conjugate direction methods: This class of methods can be regarded as somewhere between steepest descent and Newton's method. The underlying calculations are often developed and analyzed for the purely quadratic problem

$$\min \left\{ \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} - \mathbf{b}^T \mathbf{x} \right\} \quad (4.40)$$

where \mathbf{Q} is an $n \times n$ symmetric positive definite matrix, and then extended to the more general case.

Definition 4.3.1 Given a symmetric matrix \mathbf{Q} , two vectors \mathbf{d}^1 and \mathbf{d}^2 are said to be \mathbf{Q} -orthogonal or conjugate with respect to \mathbf{Q} if $(\mathbf{d}^1)^T \mathbf{Q} \mathbf{d}^2 = 0$.

Proposition 4.3.1 If \mathbf{Q} is positive definite and the set of nonzero vectors $\mathbf{d}^0, \mathbf{d}^1, \dots, \mathbf{d}^k$ are \mathbf{Q} -orthogonal, then these vectors are linearly independent.

As a direct result of Proposition 4.3.1, if \mathbf{Q} is an $n \times n$ positive definite matrix and $\mathbf{d}^0, \mathbf{d}^1, \dots, \mathbf{d}^{n-1}$ are \mathbf{Q} -orthogonal, then the solution \mathbf{x}^* to (4.40) can be expanded in terms of these vectors giving

$$\mathbf{x}^* = \alpha_0 \mathbf{d}^0 + \dots + \alpha_{n-1} \mathbf{d}^{n-1}$$

for some set of α_i 's. Multiplying by \mathbf{Q} and then taking the scalar product with \mathbf{d}^i yields

$$\alpha_i = \frac{\mathbf{d}^i \mathbf{Q} \mathbf{x}^*}{\mathbf{d}^i \mathbf{Q} \mathbf{d}^i} = \frac{\mathbf{d}^i \mathbf{b}}{\mathbf{d}^i \mathbf{Q} \mathbf{d}^i} \quad (4.41)$$

where $\mathbf{Q}\mathbf{x} = \mathbf{b}$ are the necessary and sufficient optimality conditions for problem (4.40). Note, the transpose symbol T has been omitted in (4.41) for convenience. The result in (4.41) shows that the parameters α_i and consequently the solution \mathbf{x}^* can be found by the evaluation of simple scalar products. As a consequence, we have

Theorem 4.3.5 (Conjugate Direction Theorem) Let $\{\mathbf{d}^i\}_{i=0}^{n-1}$ be a set of nonzero \mathbf{Q} -orthogonal vectors. For any $\mathbf{x}^0 \in R^n$ the sequence $\{\mathbf{x}^k\}$ generated according to

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{d}^k, \quad k \geq 0$$

with

$$\alpha_k = \frac{\mathbf{g}^k \mathbf{d}^k}{\mathbf{d}^k \mathbf{Q} \mathbf{d}^k}$$

and

$$\mathbf{g}^k = \mathbf{Q}\mathbf{x}^k - \mathbf{b}$$

converges to the unique solution, \mathbf{x}^* , of $\mathbf{Q}\mathbf{x} = \mathbf{b}$ after n steps; that is, $\mathbf{x}^n = \mathbf{x}^*$.

To convert the results of this theorem into an algorithm it is necessary to specify how the conjugate directions \mathbf{d}^k are selected at each iteration. The simplest procedure is presented in the following algorithm.

Conjugate Gradient Algorithm

Starting with any $\mathbf{x}^0 \in R^n$ define $\mathbf{d}^0 = -\mathbf{g}^0 = \mathbf{b} - \mathbf{Q}\mathbf{x}^0$ and iterate as follows

$$\begin{aligned} \mathbf{x}^{k+1} &= \mathbf{x}^k + \alpha_k \mathbf{d}^k \\ \alpha_k &= \frac{\mathbf{g}^k \mathbf{d}^k}{\mathbf{d}^k \mathbf{Q} \mathbf{d}^k} \\ \mathbf{g}^{k+1} &= -\mathbf{g}^{k+1} + \beta_k \mathbf{d}^k \\ \beta_k &= \frac{\mathbf{g}^{k+1} \mathbf{Q} \mathbf{d}^k}{\mathbf{d}^k \mathbf{Q} \mathbf{d}^k} \end{aligned}$$

where $\mathbf{g}^k = \mathbf{Q}\mathbf{x}^k - \mathbf{b}$.

For $k = 0$, the algorithm takes a steepest descent step; each succeeding step moves in a direction that is a linear combination of the current gradient and the preceding direction vector. The simple updating formulas make the procedure easy to implement and for a quadratic objective function, guarantee a solution in at most n iterations.

The general unconstrained optimization problem $\min\{f(\mathbf{x}) : \mathbf{x} \in R^n\}$ can be attacked by making suitable approximations to the components in the conjugate gradient algorithm. The most common are quadratic approximations where the following associations are made at \mathbf{x}^k :

$$\mathbf{g}^k \leftrightarrow \nabla f(\mathbf{x}^k), \quad \mathbf{Q} \leftrightarrow \mathbf{F}(\mathbf{x}^k)$$

Substituting these values into the algorithm allows us to evaluate all the necessary quantities and parameters. If f is quadratic, these associations are, of course, identities. An attractive feature of the algorithm is that, just as in the pure form of Newton's method, no line search is required at any stage. Its undesirable features are that $\mathbf{F}(\mathbf{x}^k)$ must be evaluated at each point, which is often impractical, and that the algorithm is not, in the form presented, globally convergent (see [L7]).

Quasi-Newton methods: Again working under the assumption that the evaluation of the inverse Hessian matrix is impractical or too costly, the idea underlying quasi-Newton methods is to use an approximation in place of the true inverse. The form varies with the different approaches. In general, an approximation is constructed iteratively as the descent procedure progresses. The current approximation, denoted by \mathbf{H}_k , is then used at each stage to define the next descent direction. Ideally, the approximations converge to the true inverse at the solution point and the overall procedure behaves like Newton's method. We now show how the inverse Hessian can be built up from gradient information obtained at various points.

Let f be a function on R^n that has continuous second partial derivatives. If for two points $\mathbf{x}^k, \mathbf{x}^{k+1}$ we define $\mathbf{g}^k = \nabla f(\mathbf{x}^k)$, $\mathbf{g}^{k+1} = \nabla f(\mathbf{x}^{k+1})$ and $\mathbf{p}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$, then

$$\mathbf{g}^{k+1} - \mathbf{g}^k \approx \mathbf{F}(\mathbf{x}^k)\mathbf{p}^k$$

If the Hessian \mathbf{F} is constant, then we have

$$\mathbf{q}^k \triangleq \mathbf{g}^{k+1} - \mathbf{g}^k = \mathbf{F}(\mathbf{x}^k)\mathbf{p}^k \tag{4.42}$$

and we see that the evaluation of the gradient at two points gives information about \mathbf{F} . If n linearly independent directions $\mathbf{p}^0, \mathbf{p}^1, \dots, \mathbf{p}^{n-1}$ and the corresponding \mathbf{q}^k values are known, then \mathbf{F} is uniquely determined. Letting \mathbf{P} and \mathbf{Q} be the $n \times n$ matrices with columns \mathbf{p}^k and \mathbf{q}^k , respectively, we have

$$\mathbf{F} = \mathbf{Q}\mathbf{P}^{-1}$$

It is natural to attempt to construct successive approximations \mathbf{H}_k to \mathbf{F} based on data obtained from the first k iterations of a descent process in such a way that if \mathbf{F}^{-1} were constant the approximation would be consistent with (4.42) for these iterations. Specifically, if \mathbf{F} were constant \mathbf{H}_{k+1} would satisfy

$$\mathbf{H}_{k+1}\mathbf{q}^i = \mathbf{p}^i, \quad 0 \leq i \leq k \quad (4.43)$$

After n linearly independent steps we would then have $\mathbf{H}_n = \mathbf{F}^{-1}$.

For any $k < n$ the problem of constructing a suitable \mathbf{H}_k , which in general serves as an approximation to the inverse Hessian and which in the case of constant \mathbf{F} satisfies (4.43), admits an infinite number of solutions since there are more degrees of freedom than there are constraints. Thus a particular method can take into account additional information. The earliest and one of the most clever schemes for constructing the inverse Hessian was originally proposed by Davidon and subsequently developed by Fletcher and Powell. It has the desirable property that for a quadratic objective, it simultaneously generates the directions of the conjugate gradient method while constructing the inverse Hessian. The DFP procedure starts with any symmetric positive definite matrix \mathbf{H}_0 , any point \mathbf{x}^0 , and the counter $k = 0$. At the k th iteration of the we have:

1. Set $\mathbf{d}^k = -\mathbf{H}_k\mathbf{g}^k$.
2. Minimize $f(\mathbf{x}^k + t\mathbf{d}^k)$ with respect to $t \geq 0$ to obtain t_k , \mathbf{x}^{k+1} , $\mathbf{p}^k = t_k\mathbf{d}^k$, and \mathbf{g}^{k+1} .
3. Set $\mathbf{q}^k = \mathbf{g}^{k+1} - \mathbf{g}^k$ and

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\mathbf{p}^k(\mathbf{p}^k)^T}{(\mathbf{p}^k)^T \mathbf{q}^k} + \frac{\mathbf{H}_k \mathbf{q}^k (\mathbf{q}^k)^T \mathbf{H}_k}{(\mathbf{q}^k)^T \mathbf{H}_k \mathbf{q}^k}$$

Put $k \leftarrow k + 1$ and return to Step 1.

Note that the algorithm calls for a line search at Step 2 even though it is a conjugate gradient method.

An extension of the DFP algorithm involves estimating and iteratively updating not only the inverse Hessian but the Hessian, as well. The combined procedure is known as the BFGS method and is perhaps the most widely used. Under various and somewhat stringent conditions, all quasi-Newton methods can be shown to be globally convergent. If, on the other hand, they are restarted every n or $n + 1$ steps by resetting the approximate inverse Hessian to its initial value, then global convergence is guaranteed by the presence of the first descent step of each cycle. Luenberger provides some test results comparing steepest descent, DFP, DFP with restart, and self-scaling as a function of the error in the step size t during the line search.

4.4 ALGORITHMS FOR CONSTRAINED OPTIMIZATION

Methods for solving a constrained optimization problem in n variables and m constraints can be roughly divided into four categories that depend on the dimension of the space in which the accompanying algorithm works. Primal methods work in $n - m$ space, penalty methods work in n space, dual and cutting plane methods work in m space, and Lagrangian methods work in $n + m$ space. Each represent different approaches and are founded on different aspects of NLP theory. Nevertheless, there are also strong interconnections between them, both in final form of implementation and in their performance. An analysis of the rates of convergence of most practical algorithms are determined by the structure of the Hessian of the Lagrangian much like the structure of the Hessian of the objective function determines the rates of convergence for most unconstrained methods.

In the remainder of this section, we present several of the most popular procedures for solving problem (4.1). The first two, Zoutendijk's feasible direction method and the gradient projection method, are primal approaches and are quite intuitive. The third is the now classical penalty approach developed by Fiacco and McCormick and is perhaps the simplest to implement; the fourth is a Lagrangian approach known as sequential quadratic programming.

4.4.1 Primal Methods

In solving a nonlinear program, primal or feasible direction methods work on the original problem directly by searching the feasible region for an optimal solution. Each point generated in the process is feasible and the value of the objective function constantly decreases. These methods have three significant advantages: (1) if they terminate before confirming optimality (which is very often the case with all procedures), the current point is feasible; (2) if they generate a convergent sequence, it can usually be shown that the limit point of that sequence must be at least a local minimum; (3) they do not rely on special structure, such as convexity, so they are quite general. Notable disadvantages are that they require a phase 1 procedure to obtain an initial feasible point and that they are all plagued, particularly when the problem constraints are nonlinear, with computational difficulties arising from the necessity to remain within the constraint region as the algorithm progresses. The convergence rates of primal methods are competitive with those of other procedures, and for problems with linear constraints, they are often among the most efficient.

Feasible direction methods embody the same philosophy as the techniques of unconstrained minimization discussed previously but are designed to deal with inequality constraints. Briefly, the idea is to pick a starting point satisfying the constraints and to find a direction such that (1) a small move in that direction remains feasible, and

(2) the objective function improves. One then moves a finite distance in the determined direction, obtaining a new and better point, and repeats the process until no direction satisfying both (1) and (2) can be found. In general, the terminal point is a constrained local (but not necessarily global) minimum of the problem. A direction satisfying both (1) and (2) is called a *usable feasible direction*. There are many ways of choosing such directions, hence many different primal methods.

Zoutendijk's method: Consider the problem

$$\min\{f(\mathbf{x}) : g_i(\mathbf{x}) \geq 0, i = 1, \dots, m\} \quad (4.44)$$

The constraint set is $S = \{\mathbf{x} : g_i(\mathbf{x}) \geq 0, i = 1, \dots, m\}$. Assume that the starting point $\mathbf{x}^0 \in S$. The problem is to choose a vector \mathbf{d} whose direction is both usable and feasible. Let $g_i(\mathbf{x}^0) = 0, i \in I$, where the indices in I correspond to the binding constraints at \mathbf{x}^0 . For feasible direction \mathbf{d} , a small move along this vector beginning at the point \mathbf{x}^0 makes no binding constraints negative, i.e.,

$$\frac{d}{dt}g_i(\mathbf{x}^0 + t\mathbf{d}) \Big|_{t=0} = \nabla g_i(\mathbf{x}^0)\mathbf{d} \geq 0, \quad i \in I$$

A usable feasible vector has the additional property that

$$\frac{d}{dt}f(\mathbf{x}^0 + t\mathbf{d}) \Big|_{t=0} = \nabla f(\mathbf{x}^0)\mathbf{d} < 0$$

Therefore the function initially decreases along the vector. In searching for a “best” vector \mathbf{d} along which to move, one could choose that feasible vector minimizing $\nabla f(\mathbf{x}^0)\mathbf{d}$. If some of the binding constraints were nonlinear, however, this could lead to certain difficulties. In particular, starting at \mathbf{x}^0 the feasible direction, \mathbf{d}^0 , that minimizes $\nabla f(\mathbf{x}^0)\mathbf{d}$ is the projection of $-\nabla f(\mathbf{x}^0)$ onto the tangent plane through \mathbf{x}^0 . Because the constraint surface is curved, movement along \mathbf{d}^0 for any finite distance violates the constraint. Thus a recovery move must be made to return to the feasible region. Repetitions of the procedure lead to inefficient zigzagging. As a consequence, when looking for a locally best direction it is wise to choose one that, in addition to decreasing f , also moves away from the boundaries of the nonlinear constraints. The expectation is that this will avoid zigzagging. Such a direction is the solution of the following problem:

$$\min_{\mathbf{d}, \xi} \xi \quad (4.45a)$$

$$\text{subject to } \nabla g_i(\mathbf{x}^0)\mathbf{d} + \theta_i \xi \geq 0, \quad i \in I \quad (4.45b)$$

$$\nabla f(\mathbf{x}^0)\mathbf{d} - \xi \leq 0 \quad (4.45c)$$

$$\mathbf{d}^T \mathbf{d} = 1 \quad (4.45d)$$

where $0 \leq \theta_i \leq 1$ is selected by the user. If all $\theta_i = 1$, then any vector (\mathbf{d}, ξ) satisfying (4.45b,c) with $\xi < 0$ is a usable feasible direction. That with minimum ξ value is

a best direction which simultaneously makes $\nabla f(\mathbf{x}^0)\mathbf{d}$ as negative and $\nabla g_i(\mathbf{x}^0)\mathbf{d}$ as positive as possible; i.e., steers away from the nonlinear constraint boundaries. Other values of θ_i enable one to emphasize certain constraint boundaries relative to others. Equation (4.45d) is a normalization requirement ensuring that minimum ξ be finite. If it were not included and a vector (\mathbf{d}, ξ) existed satisfying (4.45b,c) with ξ negative, then ξ could be made to approach $-\infty$, since (4.45b,c) are not homogeneous. Other normalizations are also possible.

Because the vectors ∇f and ∇g_i are evaluated at a fixed point \mathbf{x}^0 , the above direction-finding problem is almost linear, the only nonlinearity being (4.45d). Zoutendijk showed that this constraint can be handled by a modified version of the simplex method so problem (4.45a)–(4.45d) may be solved with reasonable efficiency. Note that if some of the original constraints in (4.44) were given as equalities, the algorithm would have to be modified slightly.

Of course, once a direction has been determined, the step size must still be found. This problem may be dealt with almost as in the unconstrained case. It is still desirable to minimize the objective function along the vector \mathbf{d} , but now no constraint may be violated. Thus t is determined to minimize $f(\mathbf{x}^k + t\mathbf{d}^k)$ subject to the constraint $\mathbf{x}^k + t\mathbf{d}^k \in S$. Any of the techniques discussed in Section 4.3 can be used. A new point is thus determined and the direction-finding problem is re-solved. If at some point the minimum $\xi \geq 0$, then there is no feasible direction satisfying $\nabla f(\mathbf{x}^0)\mathbf{d} < 0$ and the procedure terminates. The final point will generally be a local minimum of the problem. Zoutendijk showed that for convex programs the procedure converges to the global minimum.

Gradient projection method: At each iteration of Zoutendijk's method, an optimization problem must be solved to find a direction in which to move. Although the direction found was in some sense best, the procedure can be time consuming. An alternative motivated by steepest descent is provided by the gradient projection method of Rosen. Here a usable feasible direction is found without solving an optimization problem. The direction, however, may not be locally best in any sense. The method is efficient when all constraints are linear, although less so in the nonlinear case. It is of particular interest here because of the direct way in which it uses the Kuhn-Tucker conditions both to generate new directions and as a stopping criterion. Only the procedure for linear constraints will be discussed. The problem is of the form (4.44), but the constraints are linear and will be written

$$\mathbf{a}_i \mathbf{x} \geqq b_i, \quad i = 1, \dots, m$$

where \mathbf{a}_i is an n -dimensional row vector. The equation $\mathbf{a}_i \mathbf{x} = b_i$ represents a hyperplane and generally forms one of the boundaries of the constraint set. The procedure starts at some point \mathbf{x}^0 satisfying the constraints and at the k th iteration proceeds as follows:

1. Calculate $\nabla f(\mathbf{x}^k)$.

2. Associate with \mathbf{x}^k those hyperplanes passing through \mathbf{x}^k .
3. Find the projection of $-\nabla f(\mathbf{x}^k)$ onto the intersection of the hyperplanes associated with \mathbf{x}^k . In case there are no such hyperplanes (as when \mathbf{x}^k is interior to the constraint set), the intersection is the whole space and thus the projection is $-\nabla f(\mathbf{x}^k)$ itself.
4. If the projection is not the zero vector, then minimize along the projection vector subject to the constraints, as described previously, put $k \leftarrow k + 1$, and return to Step 1.
5. If the projection vector is zero, then $\nabla f(\mathbf{x}^k)$ may be written

$$\nabla f(\mathbf{x}^k) = \sum_i u_i \mathbf{a}_i \quad (4.46)$$

that is, as a linear combination of the vectors \mathbf{a}_i which are normal to the hyperplanes associated with \mathbf{x}^k (see below).

- (a) If all $u_i \geq 0$, then \mathbf{x}^k is the solution of the problem because the Kuhn-Tucker conditions are satisfied with the u_i 's as the Lagrange multipliers
- (b) Otherwise, define a new set of hyperplanes to be associated with \mathbf{x}^k by deleting from the present set the equality for which $u_i < 0$ is smallest, and return to Step 3.

Relation (4.46) in Step 5 can be explained geometrically. Consider the intersection of two hyperplanes in a line \mathbf{l} . Let \mathbf{x} be a point on the intersection. The vectors normal to these two planes, call them \mathbf{a}_1 and \mathbf{a}_2 , are the gradients of the corresponding constraint functions. Since \mathbf{a}_1 and \mathbf{a}_2 are normal to their respective hyperplanes, they are normal to the line \mathbf{l} . Moreover, if \mathbf{a}_1 and \mathbf{a}_2 are linearly independent (i.e., nonparallel in two dimensions), then any vector normal to the intersection can be written as a linear combination of \mathbf{a}_1 and \mathbf{a}_2 . If the projection of $\nabla f(\mathbf{x})$ onto \mathbf{l} at \mathbf{x} is 0, then $\nabla f(\mathbf{x})$ is perpendicular to \mathbf{l} at \mathbf{x} . Thus $\nabla f(\mathbf{x}) = u_1 \mathbf{a}_1 + u_2 \mathbf{a}_2$ which is (4.46) for two intersecting planes.

To explain Step 5(a), note that for linear constraints the vectors \mathbf{a}_i are the gradients $\nabla g_i(\mathbf{x})$ in the Kuhn-Tucker conditions. Since $u_i > 0$ only for those constraints for which $\mathbf{a}_i \mathbf{x} = 0$ and $u_i = 0$ otherwise $u_i g_i(\mathbf{x}) = 0$ is satisfied for all i . Thus the Kuhn-Tucker conditions are satisfied. For Step 5(b), it can be shown (e.g., see [B16]) that the procedure of dropping a hyperplane corresponding to $u_i < 0$ and projecting $-\nabla f$ onto the remaining constraints leads to a usable feasible direction, as does the original projection in Step 3 of $-\nabla f$ onto the intersection of all binding constraints. If equality constraints were included in the original formulation, these tests would have to be altered slightly to account for the fact that the corresponding Lagrange multipliers would be unrestricted variables.

Most of the computational effort in Rosen's method is in computing the various projections. From linear algebra we know that the projection of a given vector onto a

vector space is another vector which can be obtained from the original by multiplying it by a projection matrix \mathbf{P} . Rosen showed that to project onto the intersection of the constraints $\mathbf{a}_i \mathbf{x} = b_i$ for $i \in I$, the appropriate matrix is

$$\mathbf{P} = \mathbf{I} - \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$$

where \mathbf{A} is a matrix whose columns are the vectors \mathbf{a}_i^T . Of course, it is necessary to recompute \mathbf{P} when the set of constraints changes, and this can be cumbersome if one set differs radically from the next. The limitation of dropping one constraint at a time allows the formation of successive projection matrices one from the other by partitioning, since successive \mathbf{A} matrices differ in only one column.

It is not difficult to see why the overall procedure is not as efficient when the constraints are nonlinear. In this case, the projection is made onto the tangent planes to the constraints. Movement along these planes may then, unlike the linear case, violate the constraints. Thus a recovery step is needed and zigzagging can easily occur. For a generalization of the algorithm when nonlinear constraints are present, see [L7]; for a discussion of a third primal approach known as the *generalized reduced gradient* (GRG) method, see [L1].

4.4.2 Penalty Methods

Penalty and barrier methods approximate a constrained optimization problem with an unconstrained one and then apply standard search techniques to obtain solutions. The approximation is accomplished in the case of penalty methods by adding a term to the objective function that prescribes a high cost for violation of the constraints. In the case of barrier methods a term is added that favors points in the interior of the feasible region over those near the boundary. For a problem with n variables and m constraints, both approaches work directly in the n -dimensional space of the variables. In this section we discuss only penalty methods since barrier methods embody the same principles.

Consider the problem

$$\begin{array}{ll} \min & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in S \end{array} \quad (4.47)$$

where f is continuous function on R^n and S is a constraint set in R^n . In most applications S is defined explicitly by a number of functional constraints, but in this section the more general description in (4.47) can be handled. The idea of a penalty function method is to replace problem (4.47) by an unconstrained approximation of the form

$$\min f(\mathbf{x}) + cP(\mathbf{x}) \quad (4.48)$$

where c is a positive constant and P is a function on R^n satisfying: (i) P is continuous, (ii) $P(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in R^n$, and (iii) $P(\mathbf{x}) = 0$ if and only if $\mathbf{x} \in S$.

Example 4.4.1 Suppose S is defined by a number of inequality constraints: $S = \{\mathbf{x} : g_i(\mathbf{x}) \geq 0, i = 1, \dots, m\}$. A very useful penalty function in this case is

$$P(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^m (\min\{0, g_i(\mathbf{x})\})^2$$

The function $cP(\mathbf{x})$ is illustrated in Fig. 4.8 for the one-dimensional case with $g_1(x) = b - x$, $g_2(x) = x - a$.

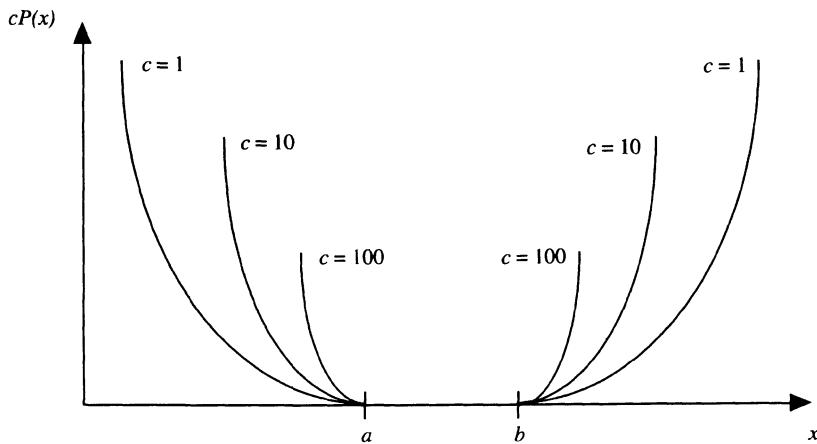


Figure 4.8 Illustration of penalty function

For large c it is clear that the minimum point of problem (4.48) will be in a region where P is small. Thus for increasing c it is expected that the corresponding solution points will approach the feasible region S and, subject to being close, will minimize f . Ideally then, as $c \rightarrow \infty$ the solution point of the penalty problem will converge to a solution of the constrained problem.

Method: The procedure for solving problem (4.47) by the penalty function method is this: Let $\{c_k\}$, $k = 1, 2, \dots$, be a sequence tending to infinity such that for each k , $c_k \geq 0$, $c_{k+1} > c_k$. Define the function

$$q(c, \mathbf{x}) = f(\mathbf{x}) + cP(\mathbf{x}) \quad (4.49)$$

For each k solve the problem

$$\min\{q(c_k, \mathbf{x}) : \mathbf{x} \in R^n\} \quad (4.50)$$

obtaining a solution point \mathbf{x}^k .

It is assumed here that for each k , problem (4.50) has a solution. This will be true, for example, if $q(c, \mathbf{x})$ increases without bounds as $|\mathbf{x}| \rightarrow \infty$.

Convergence: The following lemma gives a set of inequalities that follow directly from the definition of \mathbf{x}^k and the inequality $c_{k+1} > c_k$.

Lemma 4.4.1

$$q(c_k, \mathbf{x}^k) \leq q(c_{k+1}, \mathbf{x}^{k+1}) \quad (4.51)$$

$$P(\mathbf{x}^k) \geq P(\mathbf{x}^{k+1}) \quad (4.52)$$

$$f(\mathbf{x}^k) \leq f(\mathbf{x}^{k+1}) \quad (4.53)$$

Proof:

$$\begin{aligned} q(c_{k+1}, \mathbf{x}^{k+1}) &= f(\mathbf{x}^{k+1}) + c_{k+1}P(\mathbf{x}^{k+1}) \geq f(\mathbf{x}^{k+1}) + c_kP(\mathbf{x}^{k+1}) \\ &\geq f(\mathbf{x}^k) + c_kP(\mathbf{x}^k) = q(c_k, \mathbf{x}^k), \end{aligned}$$

which proves (4.51). We also have

$$f(\mathbf{x}^k) + c_kP(\mathbf{x}^k) \leq f(\mathbf{x}^{k+1}) + c_kP(\mathbf{x}^{k+1}) \quad (4.54)$$

$$f(\mathbf{x}^{k+1}) + c_{k+1}P(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k) + c_{k+1}P(\mathbf{x}^k) \quad (4.55)$$

Adding (4.54) and (4.55) yields $(c_{k+1} - c_k)P(\mathbf{x}^{k+1}) \leq (c_{k+1} - c_k)P(\mathbf{x}^k)$ which proves (4.52). Finally, $f(\mathbf{x}^{k+1}) + c_kP(\mathbf{x}^{k+1}) \geq f(\mathbf{x}^k) + c_kP(\mathbf{x}^k)$, and hence using (4.52) we obtain (4.53). ■

Lemma 4.4.2 Let \mathbf{x}^* be a solution to problem (4.47). Then for each k

$$f(\mathbf{x}^*) \geq q(c_k, \mathbf{x}^k) \geq f(\mathbf{x}^k)$$

Proof: $f(\mathbf{x}^*) = f(\mathbf{x}^*) + c_kP(\mathbf{x}^*) \geq f(\mathbf{x}^k) + c_kP(\mathbf{x}^k) \geq f(\mathbf{x}^k)$. ■

Global convergence of the penalty method, or more precisely verification that any limit point of the sequence is a solution, follows easily from the two lemmas above.

Theorem 4.4.1 Let $\{\mathbf{x}^k\}$ be a sequence generated by the penalty method. Then any limit point of the sequence is a solution to (4.47).

Proof: See [L7].

Penalty methods are of great interest to practitioners because they offer a simple straightforward way to handle constrained optimization problems that can be implemented without sophisticated computer programming and that possess much the same degree of generality as primal methods. Fiacco and McCormick [F5] were the first to provide an integrated theoretical framework for these methods. In their work on the interior point unconstrained minimization technique, they advocated the use

of the logarithmic penalty (really barrier) function. The corresponding unconstrained problem is:

$$\min \left\{ f(\mathbf{x}) - r \sum_{i=1}^m \ln g_i(\mathbf{x}) \right\} \quad (4.56)$$

where $r > 0$ goes to zero rather than infinity. The equivalence of (4.48) and (4.56) can be seen by setting $c = 1/r$ and noting that the negative sign in front of the summation in (4.56) is to compensate for the negativity of the logarithmic function for arguments below one.

For a given r , the stationarity condition is

$$\nabla f(\mathbf{x}(r)) - \sum_{i=1}^m \frac{r}{g_i(\mathbf{x}(r))} \nabla g_i(\mathbf{x}(r)) = \mathbf{0}$$

In addition to convergence, Fiacco and McCormick showed that as $r \rightarrow 0$, $r/g_i(\mathbf{x}(r)) \rightarrow \mu_i^*$, the optimal Lagrange multiplier for inequality i . This result is suggested by the fraction in the above summation.

Although penalty and barrier methods met with great initial success, their slow rates of convergence due to ill-conditioning of the associated Hessian led researchers to pursue other approaches. With the advent of interior point methods for linear programming, algorithmists have taken a renewed look at penalty methods and have been able to achieve much greater efficiency than previously thought possible (e.g., see [N1]). As a point of interest, one of the first algorithms proposed for solving bilevel programs was partially based on penalty techniques [A3]. More will be said about this in Chapter 8.

4.4.3 Sequential Quadratic Programming

Successive linear programming (SLP) methods solve a sequence of linear approximations to the original nonlinear program. In this respect they are similar to Zoutendijk's method but they do not require that feasibility be maintained at each iteration. Recall that if $f(\mathbf{x})$ is a nonlinear function and \mathbf{x}^c is the current value for \mathbf{x} , then the first order Taylor series expansion of $f(\mathbf{x})$ around \mathbf{x}^c is

$$f(\mathbf{x}) = f(\mathbf{x}^c + \Delta\mathbf{x}) \approx f(\mathbf{x}^c) + \nabla f(\mathbf{x}^c)(\Delta\mathbf{x}) \quad (4.57)$$

where $\Delta\mathbf{x}$ is the direction of movement. Given initial values for the variables, in SLP all nonlinear functions are replaced by their linear approximations as in (4.57). The variables in the resulting LP are the Δx_j 's representing changes from the current values. It is common to place upper and lower bounds on each Δx_j , given that the linear approximation is reasonably accurate only in some neighborhood of the initial point.

The resulting LP is solved and if the new point provides an improvement it becomes the incumbent and the process is repeated. If the new point does not yield an improvement, the step bounds may need to be reduced or we may be close enough to an optimum to stop. Successive points generated by this procedure need not be feasible even if the initial point is. However, the amount of infeasibility generally is reduced as the iterations proceed.

Successive quadratic programming (SQP) methods solve a sequence of quadratic programming approximations to the original nonlinear program [F2]. By definition, QPs have a quadratic objective function, linear constraints, and bounds on the variables. A number of efficient procedures are available for solving them. As in SLP, the linear constraints are first order approximations of the actual constraints about the current point. The quadratic objective function used, however, is not just the second order Taylor series approximation to the original objective function but a variation based on the Lagrangian.

We will derive the procedure for the equality constrained version of problem (4.1); i.e.,

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{subject to} \quad & h_i(\mathbf{x}) = 0, \quad i = 1, \dots, m \end{aligned} \quad (4.58)$$

The Lagrangian for this problem is $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\lambda}^T h(\mathbf{x})$. Recall that first order necessary conditions for the point \mathbf{x}^c to be a local minimum of (4.58) are that there exist Lagrange multipliers $\boldsymbol{\lambda}^c$ such that

$$\begin{aligned} \nabla \mathcal{L}_{\mathbf{x}}(\mathbf{x}^c, \boldsymbol{\lambda}^c) &= \nabla f(\mathbf{x}^c) - (\boldsymbol{\lambda}^c)^T \nabla h(\mathbf{x}^c) = \mathbf{0} \\ h(\mathbf{x}^c) &= \mathbf{0} \end{aligned} \quad (4.59)$$

Applying Newton's method to solve the system of equations (4.59), requires their linearization at the current point yielding the linear system

$$\begin{bmatrix} \nabla \mathcal{L}_{\mathbf{x}}^2(\mathbf{x}^c, \boldsymbol{\lambda}^c) & -\nabla^2 h(\mathbf{x}^c) \\ -\nabla^2 h(\mathbf{x}^c)^T & \mathbf{0} \end{bmatrix} \begin{pmatrix} \Delta \mathbf{x} \\ \Delta \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \nabla \mathcal{L}_{\mathbf{x}}(\mathbf{x}^c, \boldsymbol{\lambda}^c) \\ h(\mathbf{x}^c) \end{pmatrix} \quad (4.60)$$

It is easy to show that if $(\Delta \mathbf{x}, \Delta \boldsymbol{\lambda})$ satisfies (4.60) then $(\Delta \mathbf{x}, \boldsymbol{\lambda}^c + \Delta \boldsymbol{\lambda})$ will satisfy the necessary conditions for the optimality of the following QP.

$$\begin{aligned} \min \quad & f(\mathbf{x}^c)(\Delta \mathbf{x}) + \frac{1}{2} \Delta \mathbf{x}^T \nabla \mathcal{L}_{\mathbf{x}}^2(\mathbf{x}^c, \boldsymbol{\lambda}^c) \Delta \mathbf{x} \\ \text{subject to} \quad & h(\mathbf{x}^c) + \nabla h(\mathbf{x}^c) \Delta \mathbf{x} = \mathbf{0} \end{aligned} \quad (4.61)$$

On the other hand, if $\Delta \mathbf{x}^* = \mathbf{0}$ is the solution to this problem, we can show that \mathbf{x}^c satisfies the necessary conditions (4.59) for a local minimum of the original problem. First, since $\Delta \mathbf{x}^* = \mathbf{0}$, $h(\mathbf{x}^c) = \mathbf{0}$ and \mathbf{x}^c is feasible to (4.58). Now, because $\Delta \mathbf{x}^*$

solves (4.61), there exists a λ^* such that the gradient of the Lagrangian function for (4.61) evaluated at $\Delta\mathbf{x}^* = \mathbf{0}$ is also equal to zero; i.e., $\nabla f(\mathbf{x}^c) - (\lambda^*)^T \nabla \mathbf{h}(\mathbf{x}^c) = \mathbf{0}$. The Lagrange multipliers λ^* can serve as the Lagrange multipliers for the original problem and hence the necessary conditions (4.59) are satisfied by $(\mathbf{x}^c, \lambda^*)$.

The extension to inequality constraints is straightforward; they are linearized and included in the Lagrangian when computing the Hessian matrix, L , of the Lagrangian. Linear constraints and variable bounds contained in the original problem are included directly in the constraint region of (4.61). Of course, the matrix L need not be positive definite, even at the optimal solution of the NLP, so the QP may not have a minimum. Fortunately, positive definite approximations of L can be used so the QP will have an optimal solution if it is feasible. Such approximations can be obtained by a slight modification of the popular BFGS updating formula used in unconstrained minimization. This formula requires only the gradient of the Lagrangian function so second derivatives of the problem functions need not be computed.

Since the QP can be derived from Newton's method applied to the necessary conditions for the optimum of the NLP, if one simply accepts the solution of the QP as defining the next point, the algorithm behaves like Newton's method; i.e., it converges rapidly near an optimum but may not converge from a poor initial point. If $\Delta\mathbf{x}$ is viewed as a search direction, the convergence properties can be improved. However, since both objective function improvement and reduction of the constraint infeasibilities need to be taken into account, the function to be minimized in the line search process must incorporate both. Two possibilities that have been suggested are the exact penalty function and the Lagrangian function. The Lagrangian is suitable for the following reasons:

1. On the tangent plane to the active constraints, it has a minimum at the optimal solution to the NLP.
2. It initially decreases along the direction $\Delta\mathbf{x}$.

If the penalty weight is large enough, the exact penalty function also has property (2) and is minimized at the optimal solution of the NLP.

Relative advantages and disadvantages: Table 4.1, taken from [L1], summarizes the relative merits of SLP, SQP and GRG algorithms, focusing on their application to problems with many nonlinear equality constraints. One feature appears as both an advantage and a disadvantage – whether or not the algorithm can violate the nonlinear constraints of the problem by relatively large amounts during the solution process.

SLP and SQP usually generate points yielding large violations. This can cause difficulties, especially in models with log or fractional power expressions, since negative arguments for these functions may be generated. Such problems have been documented in reference to complex chemical process examples in which SLP and some

exterior penalty-type algorithms failed while an implementation of the GRG method succeeded and was quite efficient. On the other hand, algorithms that do not attempt to satisfy the equalities at each step can be faster than those that do. The fact that SLP and SQP satisfy all linear constraints at each iteration should ease the aforementioned difficulties but do not eliminate them.

There are situations in which the optimization process must be interrupted before the algorithm has reached optimality and the current point must be used or discarded. Such cases are common in on-line process control where temporal constraints force immediate decisions. In these situations, maintaining feasibility during the optimization process may be a requirement for the optimizer inasmuch as constraint violations make a solution unusable. Clearly, all three algorithms have advantages that will dictate their use in certain situations. For large problems, SLP software is used most widely because it is relatively easy to implement given a good LP system. Nevertheless, large-scale versions of GRG and SQP have become increasingly popular.

Table 4.1 Relative merits of SLP, SQP, and GRG algorithms

Algorithm	Relative advantages	Relative disadvantage
SLP	<ul style="list-style-type: none"> ■ Easy to implement ■ Widely used in practice ■ Rapid convergence when optimum is at a vertex ■ Can handle very large problems ■ Does not attempt to satisfy equalities at each iteration ■ Can benefit from improvements in LP solvers 	<ul style="list-style-type: none"> ■ May converge slowly on problems with nonvertex optima ■ Will usually violate nonlinear constraints until convergence, often by large amounts
SQP	<ul style="list-style-type: none"> ■ Generally speaking, requires fewest functions and gradient evaluations of all three algorithms (by far) ■ Does not attempt to satisfy equalities at each iteration 	<ul style="list-style-type: none"> ■ Will usually violate nonlinear constraints until convergence, often by large amounts ■ Harder than SLP to implement ■ Requires a good QP solver
GRG	<ul style="list-style-type: none"> ■ Probably most robust of all three methods ■ Versatile—especially good for unconstrained or linearly constrained problems but also works well for nonlinear constraints ■ Can utilize existing process simulators employing Newton's method ■ Once it reaches a feasible solution it remains feasible and then can be stopped at any stage with an improved solution 	<ul style="list-style-type: none"> ■ Hardest to implement ■ Needs to satisfy equalities at each step of the algorithm

PART II

BILEVEL PROGRAMMING

LINEAR BILEVEL PROGRAMMING: CONTINUOUS VARIABLES

5.1 INTRODUCTION

The vast majority of research on bilevel programming has centered on the linear version of the problem, alternatively known as the linear Stackelberg game. In this chapter we present several of the most successful algorithms that have been developed for this case, and when possible, compare their performance. We begin with some basic notation and a discussion of the theoretical character of the problem.

For $\mathbf{x} \in X \subset R^n$, $\mathbf{y} \in Y \subset R^m$, $F : X \times Y \rightarrow R^1$, and $f : X \times Y \rightarrow R^1$, the linear bilevel programming problem (BLPP) can be written as follows:

$$\min_{\mathbf{x} \in X} F(\mathbf{x}, \mathbf{y}) = \mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y} \quad (5.1a)$$

$$\text{subject to } \mathbf{A}_1 \mathbf{x} + \mathbf{B}_1 \mathbf{y} \leq \mathbf{b}_1 \quad (5.1b)$$

$$\min_{\mathbf{y} \in Y} f(\mathbf{x}, \mathbf{y}) = \mathbf{c}_2 \mathbf{x} + \mathbf{d}_2 \mathbf{y} \quad (5.1c)$$

$$\text{subject to } \mathbf{A}_2 \mathbf{x} + \mathbf{B}_2 \mathbf{y} \leq \mathbf{b}_2 \quad (5.1d)$$

where $\mathbf{c}_1, \mathbf{c}_2 \in R^n$, $\mathbf{d}_1, \mathbf{d}_2 \in R^m$, $\mathbf{b}_1 \in R^p$, $\mathbf{b}_2 \in R^q$, $\mathbf{A}_1 \in R^{p \times n}$, $\mathbf{B}_1 \in R^{p \times m}$, $\mathbf{A}_2 \in R^{q \times n}$, $\mathbf{B}_2 \in R^{q \times m}$. The sets X and Y place additional restrictions on the variables, such as upper and lower bounds or integrality requirements. Of course, once the leader selects an \mathbf{x} , the first term in the follower's objective function becomes a constant and can be removed from the problem. In this case, we replace $f(\mathbf{x}, \mathbf{y})$ with $f(\mathbf{y})$.

The sequential nature of the decisions implies that \mathbf{y} can be viewed as a function of \mathbf{x} ; i.e., $\mathbf{y} = \mathbf{y}(\mathbf{x})$. For convenience, we will refrain from writing this dependency explicitly unless there is reason to call attention to it.

Definition 5.1.1

(a) Constraint region of the BLPP:

$$S \triangleq \{(\mathbf{x}, \mathbf{y}) : \mathbf{x} \in X, \mathbf{y} \in Y, \mathbf{A}_1\mathbf{x} + \mathbf{B}_1\mathbf{y} \leq \mathbf{b}_1, \mathbf{A}_2\mathbf{x} + \mathbf{B}_2\mathbf{y} \leq \mathbf{b}_2\}$$

(b) Feasible set for the follower for each fixed $\mathbf{x} \in X$:

$$S(\mathbf{x}) \triangleq \{\mathbf{y} \in Y : \mathbf{B}_2\mathbf{y} \leq \mathbf{b}_2 - \mathbf{A}_2\mathbf{x}\}$$

(c) Projection of S onto the leader's decision space:

$$S(X) \triangleq \{\mathbf{x} \in X : \exists \mathbf{y} \in Y, \mathbf{A}_1\mathbf{x} + \mathbf{B}_1\mathbf{y} \leq \mathbf{b}_1, \mathbf{A}_2\mathbf{x} + \mathbf{B}_2\mathbf{y} \leq \mathbf{b}_2\}$$

(d) Follower's rational reaction set for $\mathbf{x} \in S(X)$:

$$P(\mathbf{x}) \triangleq \{\mathbf{y} \in Y : \mathbf{y} \in \arg \min [f(\mathbf{x}, \hat{\mathbf{y}}) : \hat{\mathbf{y}} \in S(\mathbf{x})]\}$$

(e) Inducible region:

$$IR \triangleq \{(\mathbf{x}, \mathbf{y}) : (\mathbf{x}, \mathbf{y}) \in S, \mathbf{y} \in P(\mathbf{x})\}$$

To ensure that (5.1) is well posed it is common to assume that S is nonempty and compact, and that for all decisions taken by the leader, the follower has some room to respond; i.e., $P(\mathbf{x}) \neq \emptyset$. The rational reaction set $P(\mathbf{x})$ defines the response while the inducible region IR represents the set over which the leader may optimize. Thus in terms of the above notation, the BLPP can be written as

$$\min \{F(\mathbf{x}, \mathbf{y}) : (\mathbf{x}, \mathbf{y}) \in IR\} \quad (5.2)$$

Even with the stated assumptions, problem (5.2) may not have a solution. In particular, if $P(\mathbf{x})$ is not single-valued for all permissible \mathbf{x} , the leader may not achieve his minimum payoff over IR (see Sections 1.3 and 8.1). To avoid this situation in the presentation of algorithms, it will be assumed that $P(\mathbf{x})$ is a point-to-point map. This assumption does not appear to be unduly restrictive because a simple check is available to see if the solution to (5.1), call it $(\mathbf{x}^*, \mathbf{y}^*)$, is unique. At such a point, all that is necessary is to solve the following linear program:

$$\min \{\mathbf{d}_2\mathbf{y} : (\mathbf{x}, \mathbf{y}) \in S, \mathbf{c}_1\mathbf{x} + \mathbf{d}_1\mathbf{y} = \mathbf{c}_1\mathbf{x}^* + \mathbf{d}_1\mathbf{y}^*\} \quad (5.3)$$

If the corresponding solution, say $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$, produces an objective function value $\mathbf{d}_2\hat{\mathbf{y}} < \mathbf{d}_2\mathbf{y}^*$ than the uniqueness condition does not hold.

It should be mentioned that in practice the leader will incur some cost in determining the decision space $S(X)$ over which he may operate. For example, when the BLPP

is used as a model for a decentralized firm with headquarters representing the leader and the divisions representing the follower, coordination of lower level activities by headquarters requires detailed knowledge of production capacities, technological capabilities, and routine operating procedures. Up-to-date information in these areas is not likely to be available to corporate planners without constant monitoring and oversight.

Example 5.1.1 Consider the following BLPP with $x \in R^1$, $y \in R^1$ and $X = \{x \geq 0\}$, $Y = \{y \geq 0\}$.

$$\begin{aligned} & \min_{x \geq 0} F(x, y) = x - 4y \\ \text{subject to } & \min_{y \geq 0} f(y) = y \\ \text{subject to } & -x - y \leq -3 \\ & -2x + y \leq 0 \\ & 2x + y \leq 12 \\ & -3x + 2y \leq -4 \end{aligned}$$

The polyhedron in Fig. 5.1 depicts the constraint region S for the example. For fixed $x \in S(X) = \{x : 1 \leq x \leq 4\}$, the follower's feasible set $S(x)$ is represented by the points on a vertical line contained within the polyhedron at the particular x . Because the follower's objective is to minimize y , the rational reaction set is the lowest feasible point on that vertical line, and is given by $P(x) = \max\{-x + 3, (3x - 4)/2\}$. The inducible region is the emboldened portion of the perimeter of S . It is seen to be the collection of rational reactions as x varies over $S(X)$.

The optimal solution to the example occurs at the point $(x^*, y^*) = (4, 4)$ with $F^* = -12$ and $f^* = 4$. The leader has a better objective value at $(3, 6)$ but this point is not in the inducible region. If he selects $x = 3$, the follower will respond with $y = 2.5$ giving $F(3, 2.5) = -7$ and $f(2.5) = 2.5$. This is clearly better for the follower but not for the leader. Note also that the point $(1, 2)$ is a local optimum. Any movement away from this point in the inducible region reduces the leader's objective value.

A few additional observations can be made by examining the geometry of Example 5.1.1. The first is that the optimal solution is not Pareto optimal. Any point in the intersection of S and the polar cone formed by the gradients of $-F$ and $-f$ with origin at $(4, 4)$ dominates that vertex. In fact, Marcotte and Savard [M4] have shown that unless the follower's terms in F and f are collinear (i.e., $\mathbf{d}_1 = \alpha \mathbf{d}_2$, for $\alpha > 0$), there is no assurance that the solution to the BLPP will be Pareto optimal. This means that there is no relationship between the two models. The second observation is that the inducible region is made up of a piecewise linear equality constraint derived from the faces of S . The third is that the optimum occurs at an extreme point of IR which is also an extreme point of S . Each of these observations will be made more precise in the next section.

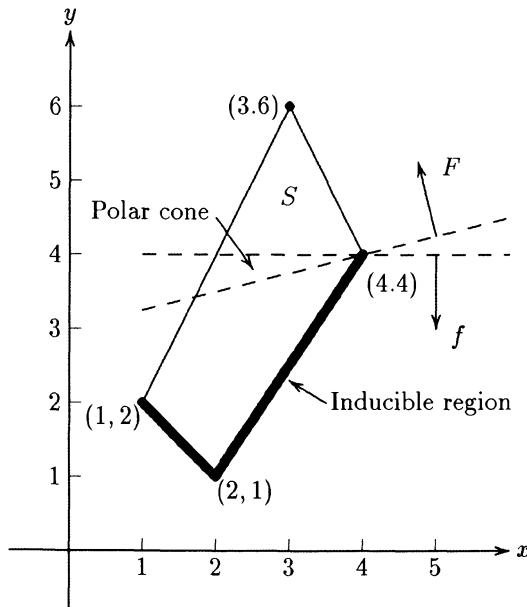


Figure 5.1 Geometry of linear BLPP

5.2 THEORETICAL PROPERTIES

The linear BLPP (5.1) was first shown to be NP-hard by Jeroslow [J2] using satisfiability arguments common in computer science. Bard [B7] provided an alternative proof by constructively reducing the problem of maximizing a strictly convex quadratic function over a polyhedron to a linear max-min problem. The latter is a special case of (5.1) obtained by omitting constraints (5.1b) and setting $c_2 = -c_1$, $d_2 = -d_1$. This result was strengthened by Hansen et al. [H1] in the following proposition.

Proposition 5.2.1 The linear max-min problem is strongly NP-hard.

Their proof is based on reduction to kernel, which was shown to be NP-hard by Chvátal (see [G1]). A kernel K of a graph is a vertex set that is stable (no two vertices of K are adjacent) and absorbing (any vertex not in K is adjacent to a vertex of K). An immediate consequence of the proposition is:

Corollary 5.2.1 The linear BLPP is strongly NP-hard.

From a computational point of view, we would like to develop algorithms whose running times are bounded by a polynomial in the problem size usually characterized by the number of variables, constraints, and bits necessary to represent the input data. A fully polynomial approximation scheme provides an ε -optimal solution to a given problem in time polynomial in the problem size and ε . It follows from the above corollary that it is not likely that we will be able to find such a scheme for (5.1).

The issue of algorithms is taken up in Section 5.3. We now turn our attention to the geometry of the solution space. In the following theorems taken from [B5], it is shown that when the linear BLPP is written as a standard mathematical program (5.2), the resultant constraint set or inducible region is comprised of connected faces of S and that a solution occurs at a vertex. For ease of presentation, it will be assumed that $P(\mathbf{x})$ is single-valued and bounded, S is bounded and nonempty, and that $Y = \{\mathbf{y} \geq 0\}$.

Theorem 5.2.1 The inducible region can be written equivalently as a piecewise linear equality constraint comprised of supporting hyperplanes of S .

Proof: To see this, let us begin by writing the inducible region explicitly as follows:

$$IR = \{(\mathbf{x}, \mathbf{y}) \in S : \mathbf{d}_2 \mathbf{y} = \min[\mathbf{d}_2 \hat{\mathbf{y}} : \mathbf{B}_2 \hat{\mathbf{y}} \leqq \mathbf{b}_2 - \mathbf{A}_2 \mathbf{x}, \hat{\mathbf{y}} \geqq 0]\}$$

Now define

$$Q(\mathbf{x}) \triangleq \min\{\mathbf{d}_2 \mathbf{y} : \mathbf{B}_2 \mathbf{y} \leqq \mathbf{b}_2 - \mathbf{A}_2 \mathbf{x}, \mathbf{y} \geqq 0\} \quad (5.4)$$

For each value of $\mathbf{x} \in S(X)$ the resulting feasible region to problem (5.2.1) is nonempty and compact. Thus $Q(\mathbf{x})$, which is a linear program parameterized in \mathbf{x} , always has a solution. From duality theory we get

$$\max\{\mathbf{u}(\mathbf{A}_2 \mathbf{x} - \mathbf{b}_2) : \mathbf{u} \mathbf{B}_2 \geqq -\mathbf{d}_2, \mathbf{u} \geqq 0\} \quad (5.5)$$

which has the same optimal value as (5.4) at the solution \mathbf{u}^* . Let $\mathbf{u}^1, \dots, \mathbf{u}^s$ be a listing of all the vertices of the constraint region of (5.5) given by $U = \{\mathbf{u} : \mathbf{u} \mathbf{B}_2 \geqq -\mathbf{d}_2, \mathbf{u} \geqq 0\}$. Because we know that a solution to (5.5) occurs at a vertex of U we get the equivalent problem

$$\max\{\mathbf{u}^j(\mathbf{A}_2 \mathbf{x} - \mathbf{b}_2) : \mathbf{u}^j \in \{\mathbf{u}^1, \dots, \mathbf{u}^s\}\} \quad (5.6)$$

which demonstrates that $Q(\mathbf{x})$ is a piecewise linear function. Rewriting IR as

$$IR = \{(\mathbf{x}, \mathbf{y}) \in S : Q(\mathbf{x}) - \mathbf{d}_2 \mathbf{y} = 0\} \quad (5.7)$$

yields the desired result. ■

Corollary 5.2.2 The linear BLPP (5.1) is equivalent to minimizing F over a feasible region comprised of a piecewise linear equality constraint.

The function $Q(\mathbf{x})$ defined by (5.4) is convex and continuous. In general, because we are minimizing a linear function $F = \mathbf{c}_1\mathbf{x} + \mathbf{d}_1\mathbf{y}$ over IR , and because F is bounded below on S by, say, $\min\{\mathbf{c}_1\mathbf{x} + \mathbf{d}_1\mathbf{y} : (\mathbf{x}, \mathbf{y}) \in S\}$, the following can be concluded.

Corollary 5.2.3 A solution to the linear BLPP occurs at a vertex of IR .

This result was prefigured by Example 5.1.1 and similarly proven by Bialas and Karwan [B23] who noted that (5.2) could be written equivalently as

$$\min\{\mathbf{c}_1\mathbf{x} + \mathbf{d}_1\mathbf{y} : (\mathbf{x}, \mathbf{y}) \in \text{co}IR\}$$

where $\text{co}IR$ is the convex hull of the inducible region. This can be seen in Fig. 5.1. Of course, $\text{co}IR$ is not the same as IR , but in the next theorem it is shown that the solution vertex $(\mathbf{x}^*, \mathbf{y}^*)$ of IR is also a vertex of S .

Theorem 5.2.2 The solution $(\mathbf{x}^*, \mathbf{y}^*)$ of the linear BLPP occurs at a vertex of S .

Proof: Let $(\mathbf{x}^1, \mathbf{y}^1), \dots, (\mathbf{x}^r, \mathbf{y}^r)$ be the distinct vertices of S . Since any point in S can be written as a convex combination of these vertices, let $(\mathbf{x}^*, \mathbf{y}^*) = \sum_{i=1}^r \alpha_i(\mathbf{x}^i, \mathbf{y}^i)$, where $\sum_{i=1}^r \alpha_i = 1$, $\alpha_i \geq 0$, $i = 1, \dots, r$ and $r \leq r$. It must be shown that $r = 1$. To see this let us write the constraints of (5.1) at $(\mathbf{x}^*, \mathbf{y}^*)$ in their piecewise linear form (5.7).

$$\begin{aligned} 0 &= Q(\mathbf{x}^*) - \mathbf{d}_2\mathbf{y}^* \\ &= Q\left(\sum_i \alpha_i \mathbf{x}^i\right) - \mathbf{d}_2\left(\sum_i \alpha_i \mathbf{y}^i\right) \\ &\leq \sum_i \alpha_i Q(\mathbf{x}^i) - \sum_i \alpha_i \mathbf{d}_2\mathbf{y}^i \quad \text{by convexity of } Q(\mathbf{x}) \\ &= \sum_i \alpha_i (Q(\mathbf{x}^i) - \mathbf{d}_2\mathbf{y}^i) \end{aligned}$$

But by definition,

$$Q(\mathbf{x}^i) = \min_{\mathbf{y} \in S(\mathbf{x}^i)} \mathbf{d}_2\mathbf{y} \leq \mathbf{d}_2\mathbf{y}^i$$

Therefore, $Q(\mathbf{x}^i) - \mathbf{d}_2\mathbf{y}^i \leq 0$, $i = 1, \dots, r$. Noting that $\alpha_i > 0$, $i = 1, \dots, r$, the equality in the preceding expression must hold or else a contradiction would result in the sequence above. Consequently, $Q(\mathbf{x}^i) - \mathbf{d}_2\mathbf{y}^i = 0$ for all i . This implies that $(\mathbf{x}^i, \mathbf{y}^i) \in IR$, $i = 1, \dots, r$, and that $(\mathbf{x}^*, \mathbf{y}^*)$ can be written as a convex combination of points in IR . Because $(\mathbf{x}^*, \mathbf{y}^*)$ is a vertex of IR , a contradiction results unless $r = 1$. ■

In general, at the solution $(\mathbf{x}^*, \mathbf{y}^*)$ the hyperplane $\{(\mathbf{x}, \mathbf{y}) : \mathbf{c}_1\mathbf{x} + \mathbf{d}_1\mathbf{y} = \mathbf{c}_1\mathbf{x}^* + \mathbf{d}_1\mathbf{y}^*\}$ will not be a support of the set S . This can be seen in Fig. 5.1 where $F = x - 4y$ cuts through S at the solution $(4, 4)$. Furthermore, from the proof of Theorem 5.2.2 it can be seen that any vertex of IR is also a vertex of S , implying that IR consists

of faces of S . Comparable results were derived by Bialas and Karwan who began by showing that any point in S that strictly contributes in any convex combination of points in S to form a point in IR must also be in IR . This leads to the following:

Corollary 5.2.4 If \mathbf{x} is an extreme point of IR , then it is an extreme point of S .

A final point about the solution of the linear BLPP can be inferred from Corollary 5.2.2. Because the inducible region is not in general convex, the set of optimal solutions to (5.1) when not single-valued is not necessarily convex. This can be seen by replacing the leader's objective function in Example 5.1.1 with $F = 2x - 3y$. Now both $(1, 2)$ and $(4, 4)$ are globally optimal but no point on the line joining them is in the inducible region.

In searching for a way to solve the linear BLPP, it would be helpful to have an explicit representation of IR rather than the implicit representation given by (5.7). This can be achieved by replacing the follower's problem (5.1c,d) with his Kuhn-Tucker conditions and append the resultant system to the leader's problem. Letting $\mathbf{u} \in R^q$ and $\mathbf{v} \in R^m$ be the dual variables associated with constraints (5.1d) and $\mathbf{y} \geq 0$, respectively, we have the following proposition.

Proposition 5.2.2 A necessary condition that $(\mathbf{x}^*, \mathbf{y}^*)$ solves the linear BLPP (5.1) is that there exist (row) vectors \mathbf{u}^* and \mathbf{v}^* such that $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{u}^*, \mathbf{v}^*)$ solves:

$$\min \mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y} \quad (5.8a)$$

$$\text{subject to } \mathbf{A}_1 \mathbf{x} + \mathbf{B}_1 \mathbf{y} \leq \mathbf{b}_1 \quad (5.8b)$$

$$\mathbf{u} \mathbf{B}_2 - \mathbf{v} = -\mathbf{d}_2 \quad (5.8c)$$

$$\mathbf{u}(\mathbf{b}_2 - \mathbf{A}_2 \mathbf{x} - \mathbf{B}_2 \mathbf{y}) + \mathbf{v} \mathbf{y} = 0 \quad (5.8d)$$

$$\mathbf{A}_2 \mathbf{x} + \mathbf{B}_2 \mathbf{y} \leq \mathbf{b}_2 \quad (5.8e)$$

$$\mathbf{x} \geq 0, \mathbf{y} \geq 0, \mathbf{u} \geq 0, \mathbf{v} \geq 0 \quad (5.8f)$$

This formulation has played a key role in the development of algorithms. One advantage that it offers is that it allows for a more robust model to be solved without introducing any new computational difficulties. In particular, by replacing the follower's objective function (5.1c) with a quadratic form

$$f(\mathbf{x}, \mathbf{y}) = \mathbf{c}_2 \mathbf{x} + \mathbf{d}_2 \mathbf{y} + \frac{1}{2} \mathbf{y}^T \mathbf{Q}_1 \mathbf{y} + \frac{1}{2} \mathbf{y}^T \mathbf{Q}_2 \mathbf{y} \quad (5.9)$$

where \mathbf{Q}_1 is an $n \times m$ matrix and \mathbf{Q}_2 is an $m \times m$ symmetric positive semidefinite matrix, the only thing that changes in (5.8) is constraint (5.8c). The new constraint remains linear but now includes all problem variables; i.e.,

$$\mathbf{x}^T \mathbf{Q}_1 + \mathbf{y}^T \mathbf{Q}_2 + \mathbf{u} \mathbf{B}_2 - \mathbf{v} = -\mathbf{d}_2 \quad (5.10)$$

From a conceptual point of view, (5.8) is a standard mathematical program and should be relatively easy to solve because all but one constraint is linear. Nevertheless, virtually all commercial nonlinear codes find complementarity terms like (5.8d)

notoriously difficult to handle so some ingenuity is required to maintain feasibility and guarantee global optimality.

5.3 ALGORITHMS FOR THE LINEAR BILEVEL PROGRAMMING PROBLEM

There have been nearly two dozen algorithms proposed for solving the linear BLPP since the field caught the attention of researchers in the mid-1970s. Many of these are of academic interest only because they are either impractical to implement or grossly inefficient. In this section, we present five of the more successful schemes and highlight their relative performance.

In general, there are three different approaches to solving (5.1) that can be considered workable. The first makes use of Corollary 2.1.1 and Theorem 5.2.2 and involves some form of vertex enumeration in the context of the simplex method. Candler and Townsley [C2] were the first to develop an algorithm that was globally optimal. Their scheme repeatedly solves two linear programs, one for the leader in all of the x variables and a subset of the y variables associated with an optimal basis to the follower's problem, and the other for the follower with all the x variables fixed. In a systematic way they explore optimal bases of the follower's problem for x fixed and then return to the leader's problem with the corresponding basic y variables. By focusing on the reduced cost coefficients of the y variables not in an optimal basis of the follower's problem, they are able to provide a monotonic decrease in the number of follower bases that have to be examined. Bialas and Karwan [B23] offered a different approach that systematically explores vertices beginning with the basis associated with the optimal solution to the linear program created by removing (5.1c). This is known as the *high point problem*; their algorithm is discussed in Section 5.3.1.

The second and most popular method for solving the linear BLPP is known as the "Kuhn-Tucker" approach and concentrates on (5.8). The fundamental idea is to use a branch and bound strategy to deal with the complementarity constraint (5.8d). Omitting or relaxing this constraint leaves a standard linear program which is easy to solve. The various methods proposed employ different techniques for assuring that complementarity is ultimately satisfied (e.g., see [B11, F7, H1, J4]).

The third method is based on some form of penalty approach. Aiyoshi and Shimizu [A3] addressed the general BLPP by first converting the follower's problem to an unconstrained mathematical program using a barrier method. The corresponding stationarity conditions are then appended to the leader's problem which is solved repeatedly for decreasing values of the barrier parameter. To guarantee convergence the follower's objective function must be strictly convex. This rules out the linear case, at least in theory. A different approach using an exterior penalty method was

proposed by Shimizu and Lu [S11] that simply requires convexity of all the functions to guarantee global convergence.

In the approach of White and Anandalingam [W4], the gap between the primal and dual solutions of the follower's problem for \mathbf{x} fixed is used as a penalty term in the leader's problem. Although this results in a nonlinear objective function, it can be decomposed to provide a set of linear programs conditioned on either the decision variables (\mathbf{x}, \mathbf{y}) or the dual variables \mathbf{u} of the follower's problem. They show that an exact penalty function exists that yields the global solution. Related theory and algorithmic details are highlighted in Section 5.3.5.

A fourth category of algorithms tries to extract gradient information from the lower-level problem in an effort to compute directional derivatives and subgradients of F (see [D3, F1]). Existing procedures commonly assume: (i) $\mathbf{y}(\mathbf{x})$ is locally unique for each fixed \mathbf{x} , and (ii) $\mathbf{y}(\mathbf{x})$ is continuously differentiable. To ensure these properties, nondegenerate conditions consisting of strong second-order sufficiency, linear independence, and strict complementary slackness are needed in the follower's problem at all points $\mathbf{y}(\mathbf{x})$ for \mathbf{x} fixed. To date, computational experience has been limited and no general codes are available. In addition, the restrictive nature of the assumptions makes the methodology comparably unsuited for the linear BLPP so we will not discuss it further in this chapter.

5.3.1 Kth-Best Algorithm

In light of Theorem 5.2.2 and the related theory, an extreme point search of the constraint region S might provide an efficient basis for a solution algorithm. Bialas and Karwan [B23] proposed what they call the “ K th-best” algorithm for this purpose. Under the conditions that the inducible region IR is bounded and the rational reaction set $P(\mathbf{x})$ is single-valued for all \mathbf{x} , it finds a global optimum to (5.1). At the heart of the algorithm is the extreme point ranking procedure discussed in Section 2.3.7.

To begin, let $(\mathbf{x}_{[1]}, \mathbf{y}_{[1]}), (\mathbf{x}_{[2]}, \mathbf{y}_{[2]}), \dots, (\mathbf{x}_{[N]}, \mathbf{y}_{[N]})$ denote the N ordered basic feasible solutions to the linear programming problem

$$\min\{\mathbf{c}_1\mathbf{x} + \mathbf{d}_1\mathbf{y} : (\mathbf{x}, \mathbf{y}) \in S\} \quad (5.11)$$

such that $\mathbf{c}_1\mathbf{x}_{[i]} + \mathbf{d}_1\mathbf{y}_{[i]} \leq \mathbf{c}_1\mathbf{x}_{[i+1]} + \mathbf{d}_1\mathbf{y}_{[i+1]}$, $i = 1, \dots, N - 1$. Then solving (5.1) is equivalent to finding the index $K^* \triangleq \min\{i \in \{1, \dots, N\} : (\mathbf{x}_{[i]}, \mathbf{y}_{[i]}) \in IR\}$ yielding the global optimum $(\mathbf{x}_{[K^*]}, \mathbf{y}_{[K^*]})$. This requires finding the (K^*)th best extreme point solution to problem (5.11) and may be accomplished with the following procedure.

Step 1 Put $i \leftarrow 1$. Solve problem (5.11) with the simplex method to obtain the optimal solution $(\mathbf{x}_{[1]}, \mathbf{y}_{[1]})$. Let $W = \{(\mathbf{x}_{[1]}, \mathbf{y}_{[1]})\}$ and $T = \emptyset$. Go to Step 2.

Step 2 Solve the follower's linear program below with the bounded simplex method to see if the current point is in the rational reaction set $P(\mathbf{x}_{[i]})$.

$$\min \{\mathbf{d}_2 \mathbf{y} : \mathbf{y} \in P(\mathbf{x}_{[i]})\} \quad (5.12)$$

Let $\tilde{\mathbf{y}}$ denote the optimal solution to (5.12). If $\tilde{\mathbf{y}} = \mathbf{y}_{[i]}$, stop; $(\mathbf{x}_{[i]}, \mathbf{y}_{[i]})$ is the global optimum to (5.1) with $K^* = i$. Otherwise, go to Step 3.

Step 3 Let $W_{[i]}$ denote the set of adjacent extreme points of $(\mathbf{x}_{[i]}, \mathbf{y}_{[i]})$ such that $(\mathbf{x}, \mathbf{y}) \in W_{[i]}$ implies $\mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y} \geq \mathbf{c}_1 \mathbf{x}_{[i]} + \mathbf{d}_1 \mathbf{y}_{[i]}$. Let $T = T \cup \{(\mathbf{x}_{[i]}, \mathbf{y}_{[i]})\}$ and $W = (W \cup W_{[i]}) \setminus T$. Go to Step 4.

Step 4 Set $i \leftarrow i + 1$ and choose $(\mathbf{x}_{[i]}, \mathbf{y}_{[i]})$ so that

$$\mathbf{c}_1 \mathbf{x}_{[i]} + \mathbf{d}_1 \mathbf{y}_{[i]} = \min \{ \mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y} : (\mathbf{x}, \mathbf{y}) \in W \}$$

Go to Step 2.

As noted in Section 2.3.7, the K th best extreme point solution of a linear program is adjacent to either the 1st, 2nd, ..., or $(K - 1)$ st extreme point. Thus a simple implementation of the above algorithm consists minimally of storing indices for the columns of the bases corresponding to the first $(K - 1)$ best points and computing which extreme point adjacent to these has the minimum level one objective function value F . It might also be worthwhile to store the reduced costs associated with the $(K - 1)$ best bases.

At Step 2, a test is repeatedly performed to see if the current point is an element of the inducible region. Because each successive pair of points is adjacent, this process can be carried out efficiently by the dual simplex method. A major advantage of the algorithm is that even if storage or computational limits are reached before convergence, upper and lower bounds on the optimal solution are generated by the procedure. In Section 5.4, we present some computational results. For more discussion and a description of related algorithms, the reader is referred to [B24].

5.3.2 Kuhn-Tucker Approach

The most direct approach to solving the linear BLPP is to solve the equivalent mathematical program given in (5.8). In practice, this is not as straightforward as it sounds; however, since this problem is linear except for the complementary slackness term (5.8d) it is natural to ask if there is a way to relax or even bypass this term in an efficient manner and perhaps solve a series of "easier" problems rather than (5.8).

To see how this might be done, let us write all the inequalities in the follower's problem as $g_i(\mathbf{x}, \mathbf{y}) \geq 0$, $i = 1, \dots, q + m$, and note that complementary slackness simply means $u_i g_i(\mathbf{x}, \mathbf{y}) = 0$ ($i = 1, \dots, q + m$). Fortuny-Amat and McCarl [F7] where the first to propose a scheme that skirted these multiplicative terms. Essentially, they converted (5.8) into a mixed-integer linear program by replacing (5.8d) with the following set of inequalities: $u_i \leq M z_i$, $g_i \leq M(1 - z_i)$, where M is a sufficiently large constant and $z_i \in \{0, 1\}$ for all i . The resultant problem contains $q + m$ additional variables and $2(q + m)$ additional constraints. It can be solved with a standard zero-one mixed-integer code but not necessarily with any degree of efficiency. Limited computational experience suggests that a large portion of the underlying search tree has to be enumerated.

Bard and Falk [B10] took an alternative approach to the complementarity term (5.8d) that involved rewriting it as the sum of piecewise linear, separable functions and then using a globally convergent nonlinear code to find solutions. To illustrate, let us write (5.8d) symbolically as $\sum_i u_i g_i = 0$ and note that this expression is equivalent to

$$\sum_{i=1}^{q+m} \min\{u_i, g_i\} = 0 \quad (5.13)$$

Going one step farther, (5.13) can be rewritten as

$$\sum_{i=1}^{q+m} (\min\{0, (g_i - u_i)\} + u_i) = 0 \quad (5.14)$$

Now, by replacing $g_i - u_i$ with a new set of variables w_i , $i = 1, \dots, q + m$, we get

$$\sum_{i=1}^{q+m} (\min\{0, w_i\} + u_i) = 0 \quad (5.15a)$$

$$w_i - g_i + u_i = 0, \quad i = 1, \dots, q + m \quad (5.15b)$$

which is the desired result; i.e., constraints (5.15a,b) are equivalent to (5.8d).

At first glance it doesn't look like this transformation makes the linear BLPP any easier to solve. In essence, a complementarity term has been replaced with a piecewise linear, separable term (5.15a) and a set of $q + m$ linear equalities (5.15b). The major advantage of this formulation, though, is that algorithms exist for finding global optima. Bard and Falk used a branch and bound code that works by enclosing the feasible region of the now separable nonconvex program in a polyhedron which is then divided into disjoint subsets. A lower bound on the optimal value of the problem is found by minimizing the objective function (5.8a) over each of these subsets and selecting the smallest value obtained. A check for the solution is made which, if successful, yields a global optimum of the piecewise linear approximation to the separable nonconvex problem. If the check fails, the subset corresponding to the

smallest lower bound is further subdivided into either two or three new polyhedra and the process continues as before with new and sharper bounds being determined.

The approach of Fortuny-Amat and McCarl is quite similar to that proposed by Bard and Falk. Both require the addition of $q + m$ variables and the explicit satisfaction of the complementary slackness, albeit in different ways. In a later study, Bard and Moore [B11] developed an implicit approach to satisfying this constraint which proved to be much more efficient. The basic idea of their algorithm is to suppress the complementarity term and solve the resulting linear subproblem. At each iteration, a check is made to see if (5.8d) is satisfied. If so, the corresponding point is in the inducible region and hence a potential solution to (5.1); if not, a branch and bound scheme is used to implicitly examine all combinations of complementary slackness.

Before presenting the algorithm, we introduce some notation similar to that used in Section 3.2. Let $W = \{1, \dots, q + m\}$ be the index set for the terms in (5.8d), and let \bar{F} be the incumbent upper bound on the leader's objective function. At the k th level of the search tree we define a subset of indices $W_k \subset W$, and a path P_k corresponding to an assignment of either $u_i = 0$ or $g_i = 0$ for $i \in W_k$. Now let

$$\begin{aligned} S_k^+ &= \{i : i \in W_k \text{ and } u_i = 0\} \\ S_k^- &= \{i : i \in W_k \text{ and } g_i = 0\} \\ S_k^0 &= \{i : i \notin W_k\} \end{aligned}$$

For $i \in S_k^0$, the variables u_i and g_i are free to assume any nonnegative values in the solution of (5.8) with (5.8d) omitted, so complementary slackness will not necessarily be satisfied.

Algorithm

Step 0 (Initialization) Set $k = 0$, $S_k^+ = \emptyset$, $S_k^- = \emptyset$, $S_k^0 = \{1, \dots, q + m\}$, and $\bar{F} = \infty$.

Step 1 (Iteration k) Set $u_i = 0$ for $i \in S_k^+$ and $g_i = 0$ for $i \in S_k^-$. Attempt to solve (5.8) without (5.8d). If the resultant problem is infeasible, go to Step 5; otherwise, put $k \leftarrow k + 1$ and label the solution $(\mathbf{x}^k, \mathbf{y}^k, \mathbf{u}^k)$.

Step 2 (Fathoming) If $F(\mathbf{x}^k, \mathbf{y}^k) \geq \bar{F}$, go to Step 5.

Step 3 (Branching) If $u_i^k g_i(\mathbf{x}^k, \mathbf{y}^k) = 0$, $i = 1, \dots, q + m$, go to Step 4; otherwise, select i for which $u_i^k g_i(\mathbf{x}^k, \mathbf{y}^k) = 0$ is largest and label it i_1 . Put $S_k^+ \leftarrow S_k^+ \cup \{i_1\}$, $S_k^0 \leftarrow S_k^0 \setminus \{i_1\}$, $S_k^- \leftarrow S_k^-$, append i_1 to P_k , and go to Step 1.

Step 4 (Updating) $\bar{F} = F(\mathbf{x}^k, \mathbf{y}^k)$.

Step 5 (Backtracking) If no live node exists, go to Step 6. Otherwise branch to the newest live vertex and update S_k^+ , S_k^- , S_k^0 , and P_k as discussed below. Go to Step 1.

Step 6 (Termination) If $\bar{F} = \infty$, there is no feasible solution to (5.1). Otherwise, declare the feasible point associated with \bar{F} the optimal solution.

After initialization, Step 1 is designed to find a new point which is potentially bilevel feasible. If no solution exists, or the solution does not offer an improvement over the incumbent (Step 2), the algorithm goes to Step 5 and backtracks. At Step 3, a check is made to determine if the complementary slackness conditions are satisfied. In practice, if $|u_i g_i| < 10^{-6}$ it is considered to be zero. Confirmation indicates that a feasible solution of the bilevel program has been found and at Step 4 the upper bound on the leader's objective function is updated. Alternatively, if the complementary slackness conditions are not satisfied, the term with the largest product is used at Step 3 to provide the branching variable. Branching is always done on the Kuhn-Tucker multiplier.

At Step 5, the backtracking operation is performed. Note that a live node is one associated with a subproblem that has not yet been fathomed at either Step 1 due to infeasibility or at Step 2 due to bounding, and whose solution violates at least one complementary slackness condition. To facilitate bookkeeping, the path P_k in the branch and bound tree is represented by an ℓ -dimensional vector, where ℓ is the current depth of the tree. The order of the components of P_k is determined by their level in the tree. Indices only appear in P_k if they are in either S_k^+ or S_k^- with the entries underlined if they are in S_k^- . Because the algorithm always branches on a Kuhn-Tucker multiplier first, backtracking is accomplished by finding the rightmost non-underlined component of P_k , underlining it, and erasing all entries to the right. The newly underlined entry is deleted from S_k^+ and added to S_k^- ; the erased entries are deleted from S_k^- and added to S_k^0 .

If we arrive at Step 6 and $\bar{F} = \infty$, then we conclude that the original constraint region S is empty. This will only be the case if the first subproblem at Step 1 is infeasible. Alternatively, the algorithm terminates with the incumbent whose optimality is now established.

Proposition 5.3.1 Under the uniqueness assumption associated with the rational reaction set, $P(\mathbf{x})$, the algorithm terminates with the global optimum of the BLPP (5.1).

Proof. The algorithm forces satisfaction of the complementary slackness conditions in problem (5.2.5) which is an equivalent representation of (5.1.1). By implicitly considering all combinations of $u_i g_i(\mathbf{x}, \mathbf{y}) = 0$ at Steps 3 and 5, the optimal solution cannot be overlooked. ■

Example 5.3.1

$$\begin{aligned}
 & \min_{\mathbf{x} \geq 0} F(\mathbf{x}, \mathbf{y}) = -8x_1 - 4x_2 + 4y_1 - 40y_2 - 4y_3 \\
 \text{subject to } & \min_{\mathbf{y} \geq 0} f(\mathbf{x}, \mathbf{y}) = x_1 + 2x_2 + y_1 + y_2 + 2y_3 \\
 \text{subject to} & -y_1 + y_2 + y_3 \leq 1 \\
 & 2x_1 - y_1 + 2y_2 - 0.5y_3 \leq 1 \\
 & 2x_2 + 2y_1 - y_2 - 0.5y_3 \leq 1
 \end{aligned}$$

This example was taken from [C2]. When it is rewritten in its equivalent form (5.8), six Kuhn-Tucker multipliers appear implying that it may be necessary to solve as many as $2^7 - 1 = 127$ subproblems before terminating. In fact, the optimal solution is uncovered on the fourth iteration but is not confirmed until 10 subproblems are examined.

More specifically, after initializing the data, the algorithm finds a feasible solution to the Kuhn-Tucker representation with the complementary slackness conditions omitted and proceeds to Step 3. The current point, $\mathbf{x}^1 = (0, 0)$, $\mathbf{y}^1 = (1.5, 1.5, 1)$, $\mathbf{u}^1 = (0, 0, 0, 1, 1, 2)$, with $F(\mathbf{x}^1, \mathbf{y}^1) = -58$ does not satisfy complementarity so a branching variable is selected (u_6) and the index sets are updated, giving $S_1^+ = \{6\}$, $S_1^- = \emptyset$, $S_1^0 = \{1, 2, 3, 4, 5\}$ and $P_1 = (6)$. In the next two iterations, the algorithm branches on u_5 and u_4 , respectively. Now, three levels down in the tree, the current subproblem at Step 1 turns out to be infeasible so the algorithm goes to Step 5 and backtracks. The index sets are $S_3^+ = \{5, 6\}$, $S_3^- = \{4\}$, and $S_3^0 = \{1, 2, 3\}$, and $P_3 = (6, 5, 4)$. Going to Step 1, a feasible solution is found which passes the test at Step 2 and satisfies the complementary slackness conditions at Step 3. Continuing at Step 4, $\bar{F} = -29.2$. Backtracking at Step 5 yields $S_4^+ = \{6\}$, $S_4^- = \{5\}$, and $S_4^0 = \{1, 2, 3, 4\}$, and $P_4 = (6, 5)$. Returning to Step 1, another feasible solution is found, but at Step 2, the value of the leader's objective function is greater than the incumbent upper bound, so the algorithm goes to Step 5 and backtracks, giving $S_5^+ = \emptyset$, $S_5^- = \{6\}$, $S_5^0 = \{1, 2, 3, 4, 5\}$, and $P_4 = (6)$. The calculations continue until no live vertices exist. The optimal solution is $\mathbf{x}^* = (0, 0.9)$, $\mathbf{y}^* = (0, 0.6, 0.4)$, $\mathbf{u}^* = (0, 1, 3, 6, 0, 0)$ with $F^* = -29.2$. The full branch and bound tree is shown in Fig. 5.2.

By way of comparison, when this problem was solved with the separable programming approach of Bard and Falk, the optimal solution was uncovered on the 51st iteration but not recognized until iteration 103. (Each iteration required the solution of a linear program in $17 (= n + 3m + 2q)$ variables and $13 (= 2(q + m) + 1)$ constraints.) This result typifies the relative performance of these two algorithms.

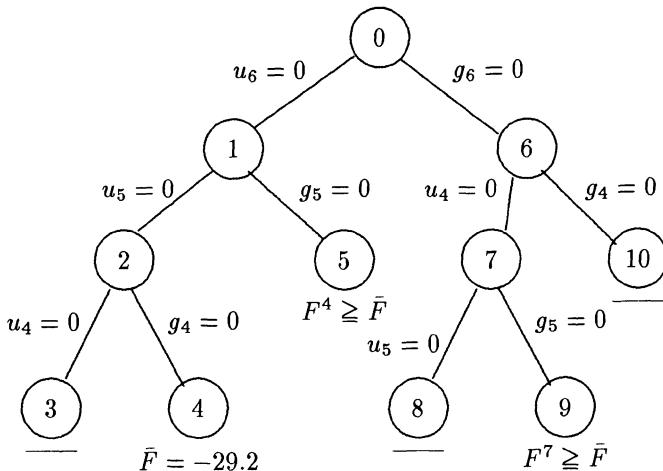


Figure 5.2 Search tree for Example 5.3.1

5.3.3 Complementarity Approach

Closely related to the Kuhn-Tucker approach is the idea of viewing (5.8) as a complementarity problem within an optimization problem. This can be seen more clearly by rewriting (5.8) as the following *minimum linear complementarity problem* (MLCP). In the literature, constraint (5.8b) has always been omitted.

$$\min \quad \mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y} \quad (5.16a)$$

$$\text{subject to } \mathbf{w} = \mathbf{b}_2 - \mathbf{A}_2 \mathbf{x} - \mathbf{B}_2 \mathbf{y} \quad (5.16b)$$

$$\mathbf{v} = \mathbf{d}_2 + \mathbf{u} \mathbf{B}_2 \quad (5.16c)$$

$$\mathbf{x} \geq 0, \mathbf{y} \geq 0, \mathbf{u} \geq 0, \mathbf{v} \geq 0, \mathbf{w} \geq 0, \mathbf{u} \mathbf{w} = \mathbf{v} \mathbf{y} = 0 \quad (5.16d)$$

Bialas and Karwan [B24] were the first to propose a transformation of (5.16) into a parametric linear complementarity problem (LCP). They then used a restricted basis entry algorithm, known as the parametric complementary pivot algorithm, to find solutions to the corresponding LCP. Their method must be considered a heuristic, however, because it does not always converge to the optimum. Subsequently, Júdice and Faustino [J4] provided modifications that guaranteed global convergence (see [H2] for more discussion on these types of optimization problems). In their method discussed below, a parameter λ is introduced and the objective function (5.16a) is

replaced by the constraint $\mathbf{c}_1\mathbf{x} + \mathbf{d}_1\mathbf{y} \leq \lambda$ to arrive at the following parametric LCP(λ):

$$\begin{pmatrix} \mathbf{w} \\ \mathbf{v} \\ \alpha \end{pmatrix} = \begin{pmatrix} \mathbf{b}_2 \\ \mathbf{d}_2 \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ 1 \end{pmatrix} \lambda + \begin{pmatrix} -\mathbf{A}_2 & -\mathbf{B}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_2^T \\ -\mathbf{c}_1 & -\mathbf{d}_1 & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{u} \end{pmatrix} \quad (5.17)$$

$$\mathbf{x} \geq \mathbf{0}, \mathbf{y} \geq \mathbf{0}, \mathbf{u} \geq \mathbf{0}, \mathbf{v} \geq \mathbf{0}, \mathbf{w} \geq \mathbf{0}, \alpha \geq 0, \mathbf{u}\mathbf{w} = \mathbf{v}\mathbf{y} = 0$$

The global minimum $(\mathbf{x}^*, \mathbf{y}^*)$ of the linear BLPP is similarly the solution of the LCP(λ^*), where λ^* is the smallest value of λ such that (5.17) is feasible. To find $(\mathbf{x}^*, \mathbf{y}^*)$ the method starts by solving the LCP(λ_0) obtained from LCP(λ) by omitting the constraint $\mathbf{c}_1\mathbf{x} + \mathbf{d}_1\mathbf{y} \leq \lambda$. Let $(\mathbf{x}^0, \mathbf{y}^0)$ be the solution of this LCP and let $\lambda_0 = \mathbf{c}_1\mathbf{x}^0 + \mathbf{d}_1\mathbf{y}^0$. The algorithm then solves a sequence of LCPs(λ_k), where $\{\lambda_k\}$ is a decreasing sequence defined by

$$\lambda_k = \mathbf{c}_1\mathbf{x}^{k-1} + \mathbf{d}_1\mathbf{y}^{k-1} - \gamma|\mathbf{c}_1\mathbf{x}^{k-1} + \mathbf{d}_1\mathbf{y}^{k-1}|$$

with $(\mathbf{x}^{k-1}, \mathbf{y}^{k-1})$ being a solution of LCP(λ_{k-1}) and γ a small positive number. The computations are terminated at, say, iteration k when LCP(λ_k) is no longer feasible. When this occurs, the solution $(\mathbf{x}^{k-1}, \mathbf{y}^{k-1})$ of the LCP(λ_{k-1}) satisfies

$$0 \leq \mathbf{c}_1\mathbf{x}^{k-1} + \mathbf{d}_1\mathbf{y}^{k-1} - F^* \leq \gamma|\mathbf{c}_1\mathbf{x}^{k-1} + \mathbf{d}_1\mathbf{y}^{k-1}|$$

where F^* is the optimal value of the leader's objective function. Hence, if the linear BLPP has an optimal solution, the sequential (SLCP) algorithm finds an ε -optimal solution of (5.1), where

$$\varepsilon = \gamma|\mathbf{c}_1\mathbf{x}^{k-1} + \mathbf{d}_1\mathbf{y}^{k-1}| \quad (5.18)$$

In practice, if γ is quite small (0.001 is recommended), the solution $(\mathbf{x}^{k-1}, \mathbf{y}^{k-1})$ of the last feasible LCP(λ_{k-1}) is in many cases a global minimum of (5.1).

SLCP Algorithm

Step 0: Set $k = 0$.

General Step: Solve the LCP(λ_k). If no feasible solution exists go to Exit; otherwise let $(\mathbf{x}^k, \mathbf{y}^k)$ be the solution of this LCP. Set $\lambda_{k+1} = \mathbf{c}_1\mathbf{x}^k + \mathbf{d}_1\mathbf{y}^k - \gamma|\mathbf{c}_1\mathbf{x}^k + \mathbf{d}_1\mathbf{y}^k|$ where $\gamma > 0$. Set $k = k + 1$ and repeat.

Exit: If $k = 0$, problem (5.1) is infeasible; i.e., $S = \emptyset$. Otherwise $(\mathbf{x}^{k-1}, \mathbf{y}^{k-1})$ is an ε -optimal minimum of the linear BLPP, where ε is given by (5.18).

The efficiency of this algorithm depends strongly on the procedure used to solve LCP(λ). While a number of algorithms exist for solving the linear complementarity

problem (see [M19]), most employ a restricted basis entry logic and cannot be applied directly to problem (5.17). To see this, we write the LCP(λ) in the following form:

$$\begin{aligned}\omega &= \mathbf{q} + M\zeta + N\mathbf{x} \\ \omega &\geq 0, \quad \zeta \geq 0, \quad \mathbf{x} \geq 0 \\ \omega_i \zeta_i &= 0, \quad i = 1, \dots, q+m\end{aligned}\tag{5.19}$$

where $\omega = (\mathbf{w}, \mathbf{v}, \alpha)^T \in R^{q+m+1}$, $\zeta = (\mathbf{u}, \mathbf{y})^T \in R^{q+m}$, $\mathbf{x} \in R^n$, and the vector \mathbf{q} and matrices M and N are derived from (5.17) through the appropriate permutations. Now let z_0 be an artificial variable and \mathbf{p} a nonnegative vector satisfying $p_i > 0$ for all i such that $q_i < 0$, and consider the following linear program:

$$\begin{aligned}\min \quad & z_0 \\ \text{subject to} \quad & \omega = \mathbf{q} + z_0 \mathbf{p} + M\zeta + N\mathbf{x} \\ & \omega \geq 0, \quad \zeta \geq 0, \quad \mathbf{x} \geq 0, \quad z_0 \geq 0\end{aligned}\tag{5.20}$$

Restricted basis simplex procedures are extensions of the phase 1 method with a single artificial variable (see [M18]). They attempt to solve (5.19) by only using complementary feasible solutions to (5.20); i.e., solutions satisfying the third constraint in (5.19). To assure that complementarity is preserved at each iteration, a nonbasic variable ζ_i (or ω_i) with a negative reduced cost coefficient can only be a candidate to enter the basis if its complement ω_i (or ζ_i) is also nonbasic or becomes nonbasic upon pivoting. Because of this additional criterion, the procedure may terminate in one of three states:

1. An optimal solution $(\omega^*, \zeta^*, \mathbf{x}^*)$ to (5.20) is obtained with $z_0 = 0$.
2. An optimal solution $(\omega^*, \zeta^*, \mathbf{x}^*)$ to (5.20) is obtained with $z_0 > 0$.
3. A nonoptimal basic solution to (5.20) with no entering variable candidates.

In the first case, $(\omega^*, \zeta^*, \mathbf{x}^*)$ is a solution to the LCP (5.19). In the second case, no feasible solution exists to the current problem. In the third case, it is possible to reduce the objective function but not without violating complementarity so no conclusion can be drawn about the existence of a solution of (5.19). To skirt this difficulty, Júdice and Faustino have proposed a hybrid enumerative scheme that systematically explores the tree in Fig. 5.3, where i_1, i_2, \dots are the integers from the set $\{1, \dots, q+m\}$. An initial feasible solution is obtained at node 1 by solving the linear program (5.20). Each one of the subsequent nodes k is generated by solving a subproblem that involves minimizing a variable ζ_i or ω_i subject to the linear constraints of the LCP (5.19) and a some constraints of the form $\zeta_i = 0$ or $\omega_i = 0$. For instance, to generate node 3 in Fig. 5.3 it is necessary to solve the linear program

$$\begin{aligned}\min \quad & z_{i_2} \\ \text{subject to} \quad & \omega = \mathbf{q} + M\zeta + N\mathbf{x}, \quad \omega \geq 0, \quad \zeta \geq 0, \quad \mathbf{x} \geq 0 \\ & z_{i_1} = 0\end{aligned}$$

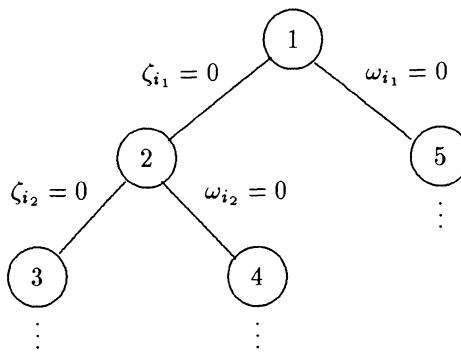


Figure 5.3 Search tree for enumerative approach to $LCP(\lambda)$

Such a linear program can be solved by a modification of phase 2 of the simplex method that exploits the same idea of controlling the number of pairs of basic complementary variables. Two cases may occur:

- (i) If the variable being minimized has value zero, then it is fixed at zero in all descendent paths of the tree.
- (ii) If the minimum value of the variable is positive, the branch is pruned and the node is fathomed.

The enumerative procedure attempts to solve the LCP by generating successive nodes of the tree according to the above process. The algorithm either finds a solution to the LCP (the first complementary feasible solution) or establishes that none exists. In the latter case, all nodes of the tree that were generated are fathomed.

Testing has shown that the hybrid method is efficient if a complementary solution is found early on or it is quickly verified that none exists. As might be expected, there are some heuristic rules relating to the choice of the node to explore next and the choice of the branching pair (ζ_i, ω_i) that can improve performance. In addition, investigating bases adjacent to the current solution can lead to improvement before iterating. In a related paper, Júdice and Faustino [J5] extend these ideas and develop an algorithm for solving the linear-quadratic BLPP where the follower's objective function has the form given in (5.9).

5.3.4 Variable Elimination Algorithm

Hansen, Jaumard and Savard [H1] developed a branch and bound-type algorithm that is not based on the Kuhn-Tucker formulation (5.8) but instead tries to determine which of the follower's constraints are binding at the optimal solution. If these were known, it would be an easy matter to obtain the optimum to (5.8) by setting the multipliers of the nonbinding constraints to zero and solving the resultant linear program. From a computational point of view, by identifying or designating a constraint as binding, it is possible to eliminate one of the follower's variables and hence reduce the size of his problem. In the extreme, the follower's subproblem becomes null and backtracking can take place. We now present the underlying theory and discuss some of the algorithmic details. An important component of the method is the use of penalties, similar to those used in mixed-integer programming, to determine the implications of making a nonbinding constraint in the leader's subproblem tight by setting a particular positive slack variable to zero.

At the center of variable eliminate algorithm is a set of necessary optimality conditions expressed in terms of the tightness of the constraints in the follower's subproblem for \mathbf{x} fixed; i.e., $S(\mathbf{x}) = \{\mathbf{y} : \mathbf{B}_2\mathbf{y} \leq \mathbf{b}_2 - \mathbf{A}_2\mathbf{x}, \mathbf{y} \geq 0\}$. For each of the $q + m$ constraints in $S(\mathbf{x})$, associate a Boolean variable α_i equal to 1 if the constraint is tight and equal to 0 otherwise.

Theorem 5.3.1 In any rational solution to the linear BLPP the tightness of the constraints in the follower's subproblem is such that

$$\sum_{i \in \{i : B_{ij}^2 > 0\}} \alpha_i \geq 1 \quad \text{if } d_{2j} < 0 \quad (5.21a)$$

$$\sum_{i \in \{i : B_{ij}^2 < 0\}} \alpha_i + \alpha_{q+j} \geq 1 \quad \text{if } d_{2j} > 0 \quad (5.21b)$$

for $j = 1, \dots, m$, where B_{ij}^2 is the ij th component of \mathbf{B}_2 .

The proof is based on the fact that if either of these conditions were violated, it would be possible for the follower to increase or decrease, respectively, the corresponding value of y_j so as to make at least one of the nonbinding constraints indexed in the summations tight while improving his objective function value by $d_{2j}\Delta y_j$. Because all optimal solutions to the linear BLPP are rational it follows immediately that (5.21a) and (5.21b) must be satisfied for all $j \in \{1, \dots, m\}$ for $d_{2j} < 0$ and $d_{2j} > 0$, respectively.

In the proposed procedure, branching is done by fixing one or more binary variables α_i to 0 or 1. In the simplest case, one variable at a time is selected for branching; alternatively, one of the logical relationships (5.21a) or (5.21b) can be chosen. For

example, if $\alpha_{i_1} + \alpha_{i_2} + \alpha_{i_3} \dots \geq 1$ is the candidate, branching may done according to the rule $\alpha_{i_1} = 1$ or ($\alpha_{i_1} = 0$ and $\alpha_{i_2} = 1$) or ($\alpha_{i_1} = \alpha_{i_2} = 0$ and $\alpha_{i_3} = 1$) and so on.

If $\alpha_i = 1$, the i th constraint in the follower's subproblem becomes an equality. It can then be used to eliminate one of the follower's variables y_j (when $i > q$, we have $j = i - q$, so $y_j = 0$). New logical relations of the type (5.21a,b) can then be derived. Of course, variable elimination does not reduce the number of structural constraints in the follower's subproblem because nonnegativity must be preserved. This is achieved by replacing $y_j \geq 0$ with an inequality whose components are used in the substitution to eliminate y_j . If there are several possible choices for y_j , the variable that produces the smallest fill-in (i.e., smallest number of nonzero coefficients) is chosen.

If a subproblem is obtained in which no more \mathbf{y} variables remain, its optimal solution is found by solving the problem obtained by deleting the follower's objective function (5.1c) from (5.1). The resultant problem is called *leader relaxation* (LR) because the leader controls all remaining variables. To determine if the solution $(\mathbf{x}_L, \mathbf{y}_L)$ is rational, the follower's subproblem must be solved for the \mathbf{x} variables fixed at \mathbf{x}_L . If the follower's objective function is the same for the y_j values obtained from the tight constraints used to eliminate them and for the optimal y_j values obtained from solving the follower's subproblem, then $(\mathbf{x}_L, \mathbf{y}_L)$ is in the inducible region.

If $\alpha_i = 0$, the i th constraint in the follower's subproblem becomes a strict inequality, and from the complementary slackness theorem of linear programming, the i th variable in the dual of the follower's subproblem must be equal to 0. Because of the difficulty in handling constraints requiring that a variable be strictly positive in linear programming, the dual of the follower's subproblem will be solved instead of the primal in one test of the algorithm described below. When many α_i variables are fixed at 0, this dual problem may be infeasible. From the duality theorem and the assumption that the constraint region S is bounded, we then know that the follower's primal subproblem coupled with the strict positivity constraints is also infeasible.

In practice, variable elimination can be performed by pivoting in a simplex tableau; i.e., fixing the eliminated variable at zero and removing it from the basis. This is easily implemented in virtually all commercial linear programming packages such a OSL and CPLEX. In the proposed branch-and-bound algorithm, depth-first search is used and the subset of logical relations (denoted by R) is updated after either branching or backtracking. When a branch corresponding to $\alpha_i = 0$ is explored, all logical relations involving α_i are simplified by deleting α_i . When a branch corresponding to $\alpha_i = 1$ is explored, all logical relations involving α_i are deleted since they are trivially satisfied. Then new relations (5.21a) or (5.21b) are obtained after a variable y_j has been eliminated. Finally, redundant relations are eliminated from R . (A logical relation $r_k \triangleq \sum_{i \in I_k} \alpha_i \geq 1$, where I_k denotes the set of indices of the logical variables in r_k , is redundant if R contains another logical relation $r_\ell \triangleq \sum_{i \in I_\ell} \alpha_i \geq 1$ such that $I_\ell \subset I_k$; i.e., satisfaction of $r_\ell \geq 1$ implies that $r_k \geq 1$ is satisfied given that

all variables α_i appearing in r_ℓ also appear in r_k .) When backtracking occurs, one reverts to the set R corresponding to the most recently explored node for which one branch is unexplored and then explores this branch, updating R accordingly.

When branching is done on the α_i variables, at most $2^{q+m+1} - 1$ subproblems will be generated. If multiple branching is used this number will be less because subproblems with $\alpha_i = 0$ for all $i \in I_k$ for relations $r_k \in R$ used for branching are excluded.

As in many other algorithms for the linear BLPP, linear programming will be used to obtain bounds on the optimal value. As mentioned, this will be achieved by deleting the follower's objective functions (5.1c) (in the original problem or in the current subproblem in which some variables y_j have been eliminated). Solving the resultant linear program LR gives such a lower bound denoted by \underline{F}_L . The effect of fixing an α_i at 1; i.e., satisfying a constraint as an equality can be anticipated to some extent by computing a penalty as is commonly done in mixed-integer programming.

Consider the equations (2.27a,b) corresponding to an optimal tableau of the LR of the current subproblem:

$$\begin{aligned} F &= \underline{F}_L + \sum_{j \in Q} \bar{c}_j x_j \\ x_i &= \bar{b}_i - \sum_{j \in Q} \bar{A}_{ij} x_j, \quad i \in P \end{aligned}$$

where P denotes index set of basic variables and Q the index set of nonbasic variables. Let $Q_j = \{j \in Q : \bar{A}_{ij} > 0\}$ for a given i . Then if x_i is the slack variable of the i th constraint the down penalty for setting x_i to 0 is

$$p_i = \bar{b}_i \min_{j \in Q_j} \left\{ \frac{\bar{c}_j}{\bar{A}_{ij}} \right\}$$

This penalty corresponds to the increase in the value of F_L during the first dual-simplex iteration after adding the constraint $x_i \leq 0$. Moreover, if $r_k = \sum_{i \in I_k} \alpha_i \geq 1$ is a logical relation of the type (5.21a,b) which must be satisfied by any rational solution, then at least one of the positive slack variables x_i ($i \in I_k$) must be set to 0. It follows that

$$\underline{F}'_L = \underline{F}_L + \min_{i \in I_k} p_i$$

is a lower bound on the optimal value of the current subproblem. Finally, taking into account all logical relations of type (5.21a,b) leads to a stronger lower bound

$$\underline{F}''_L = \underline{F}'_L + \max_{k \in \{k : r_k \in R\}} \min_{i \in I_k} p_i$$

In the algorithm that follows, the current subproblem is characterized by (i) objective functions and constraints of type (5.1) in \mathbf{x} where a subset of the \mathbf{y} variables

has been eliminated, (ii) the vector α specifying which of the initial constraints are tight, loose or free, (iii) the logical relations obtained from monotonicity, (iv) the list of eliminated variables and the equalities defining their values. As mentioned, the elimination of variables can be done by pivoting and fixing of nonbasic variables at 0. For simplicity of exposition no distinction will be made between eliminated and noneliminated variables when stating the rules of the algorithm.

In the presentation, two relaxations are used. The first is the leader's relaxation (LR) obtained by omitting (5.1c) and eliminating a subset of the y variables; the second is the follower's relaxation (FR) which consists of (5.1c,d) for x fixed at, say, \hat{x} and a subset of the y variables eliminated. Also considered is the follower's subproblem FS which consists of (5.1c,d), for x fixed at \hat{x} , and is used to find a point or to see if a point is in the inducible region.

Algorithm HJS

- Step *a* (Initialization) Obtain an initial solution $(x_h, y_h) \in IR$ with a heuristic (see below). Initialize the incumbent solution $(x_{opt}, y_{opt}) \leftarrow (x_h, y_h)$ and the incumbent objective value $F_{opt} \leftarrow c_1 x_{opt} + d_1 y_{opt}$. If no initial solution can be found, initialize (x_h, y_h) to an arbitrary value and set $F_{opt} = \infty$. Let all variables α_i ($i = 1, \dots, q+m$) be free and set $R = \emptyset$.
- Step *b* (First direct optimality test) Solve LR and let (x_L, y_L) denote an optimal solution. If $F_L = c_1 x_L + d_1 y_L \geq F_{opt}$, go to Step *m* (backtracking).
- Step *c* (First direct feasibility test) Solve the dual of FR. If it has no feasible solution, go to Step *m*.
- Step *d* (Direct resolution test: Part 1) Consider again the optimal solution (x_L, y_L) of LR. Check if this point is rational for the current subproblem; i.e., solve FR(x_L), and let y_F be an optimal solution. If $d_2 y_L = d_2 y_F$ then $(x_L, y_L) \in IR$; otherwise go to Step *f*.
- Step *e* (Direct resolution test: Part 2) Consider again the optimal solution (x_L, y_L) of LR. Check if (x_L, y_L) of LR is rational for the initial problem; i.e., solve FS(x_L) and let y_{FS} be an optimal solution. If $d_2 y_L = d_2 y_{FS}$ then $(x_L, y_L) \in IR$; otherwise go to Step *f*. Update F_{opt} and (x_{opt}, y_{opt}) if $F_{opt} > c_1 x_L + d_1 y_L$ and go to Step *m*.
- Step *f* (Second direct optimality test) Compute all penalties p_i associated with strictly positive slack variables in the optimal tableau of LR. Set the other p_i equal to 0. Then for all k such that $r_k \in R$, compute

$$\pi_k = \min_{i \in I_k} p_i$$

and set

$$\Pi = \max_{k \in \{k : r_k \in R\}} \pi_k$$

If $F_{opt} \leq F_L + \Pi$, go to Step *m*.

Step *g* (Second direct feasibility test) If LR is infeasible, go to Step *m*.

Step *h* (First conditional optimality test) Consider again LR with the penalties π_i . For all i such that $F_{opt} \leq F_L + \pi_i$, fix α_i at 0 (i nonbinding) and update R .

Step *i* (Third direct optimality test) If R contains a relation r_k such that $\alpha_j = 0$ for all $j \in I_k$, go to Step *m*.

Step *j* (Relational optimality test) For all remaining y_j appearing in FR, add to R the logical relations (5.21a,b) on the α_i variables if they are not redundant. Remove from R those relations that have become redundant.

Step *k* (Second conditional optimality test) If R contains a relation r_k such that $\alpha_j = 0$ for all $j \in I_k$ except for one index i , set the corresponding α_i to 1. Eliminate from the subproblem a variable y_j remaining in the i th constraint such that fill-in is minimum and return to Step *b*.

Step *l* (Branching) Apply a branching rule to choose either a free variable α_i or a relation $r_k \in R$ for which all variables α_i with $i \in I_k$ are free. In the former case, branch by fixing α_i at 1. In the latter case, branch by fixing the first variable α_i in r_k to 1. Eliminate a variable y_j remaining in the i th constraint such that fill-in is minimum. Return to Step *b*.

Step *m* (Backtracking) If branching took place on a variable, find the last α_i branched on equal to 1, set this $\alpha_i = 0$ and free the α_j fixed at 0 after α_i was fixed at 1. Otherwise consider the last logical relation r_k for which less than $|I_k|$ branches have been explored; consider the next branch. If there is no such variable or relation, stop. Otherwise update the current subproblem and return to Step *b*.

Various heuristics may be used to obtain a rational solution at Step *a* in the absence of first-level constraints (5.1b). The one used by Hansen et al. involves solving LR with the objective function $\lambda c_1 \mathbf{x} + (1 - \lambda) \mathbf{d}_2 \mathbf{y}$; i.e., a weighted sum of the leader's objective function for the \mathbf{x} variables and of the follower's objective function for the \mathbf{y} variables. They chose λ to be equal to $\frac{n+m}{n+2m}$. Better heuristic solutions could be obtained at higher computational cost, for example, by varying λ between 0 and 1 in the following objective function $(1 - \lambda)(c_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y}) + \lambda \mathbf{d}_2 \mathbf{y}$ and keeping the first rational solution found (cf. [B2]), or by using the method proposed by Júdice and Faustino [J4].

Step *l* requires the implementation of a branching rule. In their paper, Hansen et al. investigated 7 such rules. The two presented below gave the best results. In the description, s_i denotes the slack variable associated with constraint i in the follower's subproblem and u_i the corresponding dual variable for $i = 1, \dots, q + m$.

1. BR5: *Multiple or binary branching.* (i) Select a relation $r_k \in R$ which has at least two variables, say α_{i_1} and α_{i_2} , whose corresponding slacks s_{i_1} and s_{i_2} are in the basis, such that $\sum_{i \in I_k} u_i s_i$ is maximum. (ii) If there is no such relation select the α_i associated with the largest product $u_i s_i$. This is the same rule suggested by Bard and Moore [B11].
2. BR6: Same as rule BR5 except in (ii), branch on the α_i variable with the largest penalty p_i among those for which $u_i s_i > 0$.

5.3.5 Penalty Function Approach

In the development of their separable programming algorithm for solving (5.1), Bard and Falk suggested a penalty approach based on the Kuhn-Tucker formulation (5.8). The idea was to place the complementary slackness term (5.8d) in the objective function weighted by a large positive constant K . This idea was subsequently refined by Anandalingam and White [A9] who took an equivalent approach of appending the duality gap of the follower's problem to the leader's objective. This structure leads to a decomposition of the derived problem into a series of linear programs. In their first algorithm, only local optimality was assured; in a later work [W4] they were able to obtain global solutions.

The proposed approach exploits the fact that if $(\mathbf{x}, \mathbf{y}) \in IR$, the duality gap for the follower is zero. For the linear BLPP with $X = \{\mathbf{x} \geq 0\}$ and $Y = \{\mathbf{y} \geq 0\}$, the follower's primal problem is given by (5.1c,d) for $\mathbf{x} \in S(X)$, and ignoring the constant term $c_2 \mathbf{x}$, the follower's dual problem (5.5) is

$$\max_{\mathbf{u}} \mathbf{u}(\mathbf{A}_2 \mathbf{x} - \mathbf{b}_2) \quad (5.22a)$$

$$\text{subject to } \mathbf{u} \mathbf{B}_2 \geq -\mathbf{d}_2 \quad (5.22b)$$

$$\mathbf{u} \geq 0 \quad (5.22c)$$

Given \mathbf{x} and some values of \mathbf{u} and \mathbf{y} that satisfy the primal and dual constraints of the follower's problem, the optimal value of his objective function lies in the interval $[c_2 \mathbf{x} + \mathbf{u}(\mathbf{A}_2 \mathbf{x} - \mathbf{b}_2), c_2 \mathbf{x} + \mathbf{d}_2 \mathbf{y}]$. When the duality gap, given by $\Delta(\mathbf{x}, \mathbf{y}, \mathbf{u}) = [\mathbf{d}_2 \mathbf{y} - \mathbf{u}(\mathbf{A}_2 \mathbf{x} - \mathbf{b}_2)]$, is zero, the solution \mathbf{y} would be in the rational reaction set $P(\mathbf{x})$ and hence optimal for the particular value of \mathbf{x} . Thus, as Proposition 5.3.4 below confirms, it is possible to formulate (5.1) as

$$\begin{aligned} P(K) &= \min \widehat{F} = c_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y} + K[\mathbf{d}_2 \mathbf{y} - \mathbf{u}(\mathbf{A}_2 \mathbf{x} - \mathbf{b}_2)] \\ &\text{subject to } \mathbf{A}_1 \mathbf{x} + \mathbf{B}_1 \mathbf{y} \leq \mathbf{b}_1 \\ &\quad \mathbf{u} \mathbf{B}_2 \geq -\mathbf{d}_2 \\ &\quad \mathbf{A}_2 \mathbf{x} + \mathbf{B}_2 \mathbf{y} \leq \mathbf{b}_2 \\ &\quad \mathbf{x}, \mathbf{y}, \mathbf{u} \geq 0 \end{aligned} \quad (5.23)$$

where $K \in R_+$. Note that (5.23) is similar to (5.8) except that the dual variables \mathbf{v} associated with the nonnegativity requirements $\mathbf{y} \geq \mathbf{0}$ have been removed and equation (5.8c) takes the form of the second inequality in (5.23). Complementarity $\mathbf{v}\mathbf{y} = 0$ as well as $\mathbf{u}(\mathbf{b}_2 - \mathbf{A}_2\mathbf{x} - \mathbf{B}_2\mathbf{y}) = 0$ is assured when the follower's duality gap $\Delta(\mathbf{x}, \mathbf{y}, \mathbf{u})$ is zero. This is a direct consequence of the Fundamental Duality Theorem 2.5.3 and the Complementary Slackness Theorem 2.5.4.

Now let the feasible region of (\mathbf{x}, \mathbf{y}) be denoted by S and the feasible region of \mathbf{u} be given by $U = \{\mathbf{u} : \mathbf{u}\mathbf{B}_2 \geq -\mathbf{d}_2, \mathbf{u} \geq \mathbf{0}\}$. Assume that S and U are nonempty bounded polyhedra and denote their extreme points by S^E and U^E , respectively. The following four theorems are from [A9].

Theorem 5.3.2 For a given value of $\mathbf{u} \in U$ and fixed $K \in R_+$, define

$$\Theta(\mathbf{u}, K) = \min_{\mathbf{x}, \mathbf{y}} \{\hat{F}(\mathbf{x}, \mathbf{y}, \mathbf{u}, K) : (\mathbf{x}, \mathbf{y}) \in S\} \quad (5.24)$$

Then $\Theta(\cdot, K)$ is concave on R^q and a solution to the problem

$$\min_{\mathbf{u}} \{\Theta(\mathbf{u}, K) : \mathbf{u} \in U\} \quad (5.25)$$

will occur at some $\mathbf{u}^* \in U^E$.

Theorem 5.3.3 For fixed $K \in R_+$, an optimal solution to problem (5.23) is achievable in $S^E \times U^E$, and $S^E \times U^E = (S \times U)^E$.

Theorem 5.3.4 Let $(\mathbf{x}^*, \mathbf{y}^*)$ solve the linear BLPP (5.1) and assume that the rational reaction set $P(\mathbf{x}^*)$ is unique. Then there exists a finite value $K^* \in R_+$ for which an optimal solution to the penalty function problem (5.23) yields an optimal solution to (5.1) for all $K \geq K^*$.

Theorem 5.3.5 If $(\mathbf{x}(K), \mathbf{y}(K), \mathbf{u}(K))$ solves $P(K)$ as a function of K , the leader's objective function $F(\mathbf{x}(K), \mathbf{y}(K))$ is monotonically nondecreasing and the duality gap, $\Delta(\mathbf{x}(K), \mathbf{y}(K), \mathbf{u}(K))$, of the follower's problem is monotonically nonincreasing in the value of the penalty parameter K .

The proofs of Theorems 5.3.2, 5.3.3, and 5.3.5 require that S^E and U^E are bounded. With regard to Theorem 5.3.4, a special situation arises when $\mathbf{d}_1 = \alpha\mathbf{d}_2$, $\alpha > 0$; i.e., when the second term of leader's and follower's objective functions are parallel. In this case, $K^* = 0$.

Theorem 5.3.3 provides the foundation for an algorithm that could be used to derive a quasi-local optimum for the linear BLPP. For a given K , the first obvious step is to begin with an arbitrary $(\mathbf{x}^0, \mathbf{y}^0)$ and solve the linear program $\min\{\hat{F}(\mathbf{x}^0, \mathbf{y}^0, \mathbf{u}, K) : \mathbf{u} \in U^E\}$ to obtain an optimal $\mathbf{u}^0 = \mathbf{u}(\mathbf{x}^0, \mathbf{y}^0, K)$. Then with $\mathbf{u} = \mathbf{u}^0$, find $(\mathbf{x}^1, \mathbf{y}^1) \in \arg \min\{\hat{F}(\mathbf{x}, \mathbf{y}, \mathbf{u}^0, K) : (\mathbf{x}, \mathbf{y}) \in S^E\}$. Next find $\mathbf{u}^1 = \mathbf{u}(\mathbf{x}^1, \mathbf{y}^1, K)$ and repeat. As

Wendell and Hurter [W3] have pointed out, in general, a partial optimal solution may not be locally optimal. Nevertheless, because $\widehat{F}(\cdot, \cdot, K)$ is bilinear, the type of problem being solved belongs to a class of problems for which Wendell and Hurter show that a partial optimum for $P(K)$ is locally optimal as well. If K is large enough, this solution will be a local optimum for the linear BLPP. It will be shown presently how the penalty function approach can be adapted to find global optima.

The algorithm begins with a large value of K that is increased until problem (5.1) is solved. If an appropriate K^* for Theorem 5.3.4 is found, then only this value need be used. However, K^* may be very large and hence lead to computational instabilities. It is possible though, for the penalty contribution in (5.23) to become zero for a smaller value of K than any upper bound employed, in which case the solution obtained will solve (5.1).

Development of Algorithm

Consider problem (5.23). For a given K and \mathbf{u} , let $(\mathbf{x}(\mathbf{u}, K), \mathbf{y}(\mathbf{u}, K))$ be a solution to (5.24). Some properties of the penalty function formulation represented by $P(K)$ and its relaxation $\Theta(\cdot, K)$, are now considered.

Theorem 5.3.6 For $\mathbf{u}, \mathbf{w} \in U$, we have

$$\Theta(\mathbf{w}, K) \leq \Theta(\mathbf{u}, K) - K(\mathbf{w} - \mathbf{u})(\mathbf{A}_2 \mathbf{x}(\mathbf{u}, K) - \mathbf{b}_2)$$

Proof: Follows from the concavity of $\Theta(\mathbf{w}, K)$. ■

For $\mathbf{u}, \mathbf{w} \in U$ and fixed $K \in R_+$, define

$$\Phi(\mathbf{u}, \mathbf{w}, K) = (\mathbf{w} - \mathbf{u})(\mathbf{A}_2 \mathbf{x}(\mathbf{u}, K) - \mathbf{b}_2)$$

Then from Theorem 5.3.6, if $\Phi(\mathbf{u}^1, \mathbf{w}, K) > 0$, we have

$$\mathbf{u}^1 \notin \arg \min \{\Theta(\mathbf{u}, K) : \mathbf{u} \in U\} \quad (5.26)$$

That is, there exists some $\mathbf{w} \in U$ that gives a smaller value than $\Theta(\mathbf{u}^1, K)$.

The result in (5.26) provides a mechanism for choosing the next vertex in any solution procedure for minimizing $\Theta(\cdot, K)$. Suppose that at a particular iteration \mathbf{u}^1 is the current vertex. Using \mathbf{u}^1 we obtain $\Theta(\mathbf{u}^1, K)$ and a corresponding optimal solution $(\mathbf{x}(\mathbf{u}^1, K), \mathbf{y}(\mathbf{u}^1, K))$. The next step is to examine the adjacent vertices $\{\mathbf{u}^{1s}\}$ of \mathbf{u}^1 . Let $N(\mathbf{u}^1)$ be the number of such vertices. If $\Theta(\mathbf{u}^{1s}, K) < \Theta(\mathbf{u}^1, K)$ for some s , select \mathbf{u}^{1s} as the next vertex to go to and set $\mathbf{u}^1 = \mathbf{u}^{1s}$. Otherwise, check to see if $\Phi(\mathbf{u}^1, \mathbf{u}^*, K) > 0$ for some $\mathbf{u}^*(\mathbf{u}^1, K) \in U^E$. If so, select $\mathbf{u}^*(\mathbf{u}^1, K)$ as the next vertex to examine and set $\mathbf{u}^1 = \mathbf{u}^*(\mathbf{u}^1, K)$. Repeat the process. If neither of these cases arise, then \mathbf{u}^1 is a local optimum of $\Theta(\cdot, K)$ over U and a procedure is needed to find the next local optimum. Global optimality is reached at $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{u}^*)$ when the smallest possible

value of $\widehat{F}(\cdot, \cdot, \cdot, K)$ is achieved and the duality gap is zero; i.e., $\Delta(\mathbf{x}^*, \mathbf{y}^*, \mathbf{u}^*) = 0$. The latter condition is achieved monotonically (Theorem 5.3.5) and at a finite K (Theorem 5.3.4). Thus a procedure that will increase K in incremental steps and obtain a global optimum of $P(K)$ for each value of K will yield a global optimum of the linear BLPP. It is also possible to find a local optimum of (5.1) by finding a local optimum of $\widehat{F}(\cdot, \cdot, \cdot, K)$ for each value of K , and increasing K in small increments until $\Delta(\mathbf{x}(K), \mathbf{y}(K), \mathbf{u}(K)) = 0$, where $\Delta(\mathbf{x}(K), \mathbf{y}(K), \mathbf{u}(K))$ is a local optimum.

Algorithm

Step 0 Choose K (large) and $\mathbf{u}^1 \in U^E$. Set $\Theta^1 = \infty$ and $\widehat{\mathbf{u}}^1 = \mathbf{u}^1$.

Step 1 Solve (5.24) to find $\Theta(\mathbf{u}^1, K)$, $\mathbf{x}(\mathbf{u}^1, K)$ and $\mathbf{y}(\mathbf{u}^1, K)$; set $\Theta^1 = \min\{\Theta^1, \Theta(\mathbf{u}^1, K)\}$ and

$$\widehat{\mathbf{u}}^1 = \begin{cases} \mathbf{u}^1 & \text{if } \Theta(\mathbf{u}^1, K) < \Theta^1 \\ \widehat{\mathbf{u}}^1 & \text{if } \Theta(\mathbf{u}^1, K) \geq \Theta^1 \end{cases}$$

Step 2 Let $\{\mathbf{u}^{1s}\}$ be the adjacent vertices of \mathbf{u}^1 , $s \in \{1, N(\mathbf{u}^1)\}$. If $\Theta(\mathbf{u}^{1s}, K) < \Theta^1$ for some s , set $\Theta^1 = \Theta(\mathbf{u}^{1s}, K)$ and $\mathbf{u}^1 = \mathbf{u}^{1s}$. Continue examining adjacent vertices until none remain.

Step 3 If $\Theta(\mathbf{u}^{1s}, K) \geq \Theta^1$ for all s , find $\Gamma(\mathbf{u}^1, K) = \max\{\Phi(\mathbf{u}^1, \mathbf{u}, K) : \mathbf{u} \in U\}$. Obtain $\mathbf{u}^*(\mathbf{u}^1, K)$.

Step 4 If $\Gamma(\mathbf{u}^1, K) > 0$, set $\mathbf{u}^1 = \mathbf{u}^*(\mathbf{u}^1, K)$ and go to Step 1.

Step 5 If $\Gamma(\mathbf{u}^1, K) \leq 0$, extend unit rays $\{\mathbf{t}^{1s}\}$ along edges from \mathbf{u}^1 and find $\alpha_s = \max\{\alpha \geq 0 : \Gamma(\mathbf{u}^1 + \alpha \mathbf{t}^{1s}, K) \geq \Theta^1\}$, $s \in \{1, N(\mathbf{u}^1)\}$.

Step 6 Let $\nu^{1s} = \mathbf{u}^1 + \alpha_s \mathbf{t}^{1s}$, $s \in \{1, N(\mathbf{u}^1)\}$, $\Lambda(\mathbf{u}^1) = \{\boldsymbol{\lambda} = (\boldsymbol{\mu}, \gamma) \in R^{q+1} : \boldsymbol{\mu}\mathbf{u}^1 - \gamma \leq 0, \boldsymbol{\mu}\nu^{1s} - \gamma \geq 0, s \in \{1, N(\mathbf{u}^1)\}, \gamma \in [1, -1]\}$, and for $\mathbf{u} \in U$, $G(\mathbf{u}, \mathbf{u}^1) = \max\{\boldsymbol{\mu}\mathbf{u}^1 - \gamma : \boldsymbol{\lambda} \in \Lambda(\mathbf{u}^1)\}$.

Step 7 Let $\mathbf{u}^{1*} \in \arg \min\{G(\mathbf{u}, \mathbf{u}^1) : \mathbf{u} \in U\}$.

Step 8 If $G(\mathbf{u}^{1*}, \mathbf{u}^1) \geq 0$, then $\widehat{\mathbf{u}}^1 \in \arg \min\{\Theta(\mathbf{u}, K) : \mathbf{u} \in U\}$ and the optimal value of $\Theta(\cdot, K)$ is reached for the particular K , with the solution $(\mathbf{x}(\mathbf{u}^1, K), \mathbf{y}(\mathbf{u}^1, K))$. Go to Step 10.

Step 9 If $G(\mathbf{u}^{1*}, \mathbf{u}^1) < 0$, set $\mathbf{u}^1 = \mathbf{u}^{1*}$ and go to Step 1.

Step 10 If $\Delta(\mathbf{x}(\widehat{\mathbf{u}}^1, K), \mathbf{y}(\widehat{\mathbf{u}}^1, K), \widehat{\mathbf{u}}^1) > 0$, set $K = K + \delta$ and go to Step 1; otherwise, $\Delta(\mathbf{x}(\widehat{\mathbf{u}}^1, K), \mathbf{y}(\widehat{\mathbf{u}}^1, K), \widehat{\mathbf{u}}^1) = 0$ and $\mathbf{x}(\widehat{\mathbf{u}}^1, K)$, $\mathbf{y}(\widehat{\mathbf{u}}^1, K)$ solves (5.1).

Obtaining adjacent vertices in Step 2 can be done quite simply by pivoting in the simplex algorithm. Some bookkeeping is required to make sure that the algorithm

does not return to a previously examined vertex. For more discussion, see Section 2.3.4. When one leaves Step 2, only the current vertex needs to be stored and carried forward.

Steps 5 – 8 are modifications of the original algorithm proposed by Tuy [H10]. It involves generating cones at local optima (Step 5), making sure that the particular cone includes the feasible region (Steps 6 – 7), and testing to see if a vertex included in the cone is better than the local optimal solution for the current value of K (Step 8). Each time the algorithm passes through Steps 5 – 8, the size of the feasible region that is under examination is reduced. Additional bookkeeping is required to keep track of these cones.

Degeneracy: If U is degenerate, then in Step 2 the adjacent vertices may produce the same value of Θ as \mathbf{u}^1 . The algorithm then skips to Step 5 and proceeds from there. If W is nondegenerate, then in Step 6, $\Lambda(\mathbf{u}^1)$ can be replaced by the singleton

$$\begin{aligned}\Lambda(\mathbf{u}^1) = \{\lambda = (\boldsymbol{\mu}, \gamma) \in R^{q+1} : \boldsymbol{\mu}\mathbf{u}^1 - \gamma \geq 0, \ \boldsymbol{\mu}\mathbf{v}^{1s} - \gamma = 0, \\ s \in \{1, N(\mathbf{u}^1)\}, \ \gamma \in [1, -1]\}\end{aligned}$$

In either case, the optimality of $\Theta(\hat{\mathbf{u}}^1, K)$ in Step 8 follows because, under the given assumptions, we must have $U \subset \text{co}\{\mathbf{u}^1, \{\mathbf{v}^{1s}\}\}$, and then using concavity of $\Theta(\cdot, K)$, \mathbf{u}^1 is optimal.

Cycling: The algorithm terminates in a finite number of steps due to the limited number of vertices unless there is cycling. Cycling occurs at iteration r if the best solution \mathbf{u}_r^{1*} obtained at Step 9 is the same as \mathbf{u}_q^1 for some $q < r$. To avoid this difficulty, let U^V be the set of vertices of U that is fathomed at Steps 2, 4 and 5. Let A^V be the set of vertices of U that is adjacent to some $\mathbf{u} \in U^V$. If $A^V \setminus U^V = \emptyset$, then $U^V = U^E$ and the process terminates with an optimal solution. If cycling occurs at iteration r , and $A^V \setminus U^V \neq \emptyset$, then select any $\mathbf{u} \in A^V \setminus U^V$ instead of \mathbf{u}_r^{1*} . Because the set U^V is edge connected, the process will terminate in a finite number of steps.

5.4 COMPUTATIONAL COMPARISONS

No computational experience was reported by Bialas and Karwan [B23] for the K th-best algorithm but limited experiments were carried out by White and Anandalingam [W4]. The results are presented below. Bard and Moore [B11] where the first to provide a comprehensive set of test results for algorithms designed to solve the linear BLPP. They investigated the efficiency of their branch and bound approach by randomly generating and solving a series of representative problems as follows. For all cases reported, p was 0. For the primary cases, the coefficients of the \mathbf{A}_2 and \mathbf{B}_2 matrices ranged between 15 and 45 with approximately 25% of the entries being less than

0. Each had a density of about 0.4. The coefficients of the two objective functions varied between 20 and 20 with approximately 50% being less than 0. The number of constraints in each problem was set at 0.4 times the total number of variables, and the right-hand-side values ranged between 0 and 50. The signs of the constraints had a 0.7 probability of being \leq and a 0.3 probability of being \geq .

All computations were performed on an IBM 3081-D using the vs FORTRAN compiler. As coded, the subproblems encountered at Step 1 were solved with the linear programming subroutine library XMP written by Roy Marsten. Multiplier values required to be zero on a given iteration were accommodated by fixing their upper and lower bounds at zero. Similarly, constraints required to be binding are satisfied by setting their slacks to zero. Both of these operations are easily handled in XMP by a subroutine call. All variable bounds were treated implicitly so additional constraints were not needed in the formulation.

Table 5.1 summarizes their computational experience. Each entry represents the average of 10 randomly generated problems. In all, 110 problems were solved ranging in size from 40 to 100 variables, and 16 to 40 constraints. Performance measures include CPU time (seconds), the number of nodes in the search tree, and the node at which the optimal solution is found. Also, data for the largest and smallest search trees are given as a measure of variability.

As expected, the CPU time grew exponentially with the size of the problem, but more importantly, it depended on the way the variables are partitioned between the players. As the number of variables controlled by the follower increased, the number of variables included in the branch and bound process increased along with the average computational burden. Compare, for example, the two cases with 90 variables (and 36 constraints). On average, 81 additional CPU seconds were required for the case where $m = 45$.

Table 5.1 also shows that large differences in computational effort often result among problems of equivalent size. For the 100 variable case, 34 subproblems had to be solved at one extreme and 476 at the other. The corresponding CPU times were 85 seconds and 804 seconds, respectively. In general, the optimum was not uncovered until 60% to 70% of the realized tree was examined. This implies that if the algorithm is stopped prematurely the best solution might be missed. A final point to be made about the empirical results is that about 45% of the nodes in the search tree were fathomed due to infeasibility, and rarely (only 5% of the time) due to a solution being in the inductive region. The remaining 50% were fathomed at Step 2 when the relaxed solution was greater than or equal to the incumbent. It should be noted that when the follower's objective function is a quadratic, as given in (5.9), little difference was observed in the algorithm's performance.

Table 5.1 Computational results for Bard-Moore algorithm

No. of variables ($n + m$)	Follower variables (m)	No. of constraints (q)	CPU time (sec)	Average no. of nodes	No. of nodes (range)	Optimal solution (node)
40	12	16	4.44	18	6 – 42	11
40	16	16	8.50	40	8 – 202	27
50	15	20	17.24	39	10 – 112	26
50	20	20	16.46	35	18 – 124	23
50	25	20	32.39	73	16 – 218	46
50	30	20	179.01	447	30 – 1250	391
70	28	28	106.99	96	32 – 270	67
70	35	28	122.26	106	26 – 246	84
90	36	36	352.37	138	30 – 384	81
90	45	36	433.14	185	48 – 374	122
100	40	40	294.22	159	34 – 476	120

To see how performance varied with the density of the \mathbf{A} and \mathbf{B} matrices two problems sets were investigated, and only the linear case was considered. The first problem set was characterized by parameter values $(n, m, q) = (25, 25, 20)$, and the second set by $(35, 35, 28)$. The results for density factors of 0.2, 0.3 and 0.4 are reported in Table 5.2. All entries represent an average of 10 cases. As the density is reduced from 0.4 to 0.3, the average CPU time stayed about the same while the number of nodes in the search tree increased modestly. For a density factor of 0.2, however, a significant drop in CPU time is observed. In the first case, this is accompanied by an increase in the average size of the tree from 73 nodes to 100 nodes, and in the second case by a decrease from 106 nodes to 82 nodes.

In the remainder of this section we compare the above findings with those reported by others. Although Fortuny-Amat and McCarl [F7] did not seriously test their scheme, it is an easy matter to do so. Bard and Moore investigated a few 20 variable problems with ZOOM (a zero-one, mixed-integer extension of XMP) and found that the accompanying run times and search trees were 10 to 100 times larger than those for their algorithm. Similar results were obtained for the separable approach of Bard and Falk [B10]. The parametric complementary pivot algorithm of Bialas and Karwan and Bard's grid search algorithm are technically heuristics so direct comparisons with globally convergent schemes are problematic.

Variable elimination approach: The variable elimination algorithm of Hansen et al. was coded in FORTRAN 77 and tested on a Sun 3/50 for one series of problems and on a Sun Sparcstation for another. Problems were generated randomly with the same characteristics assumed by Bard and Moore (BM) with densities of 40%, 17% and 8%. To avoid empty columns, coefficients were generated column after column with

Table 5.2 Additional results for BM algorithm with different matrix densities

No. of variables ($n + m$)	Follower variables (m)	No. of constraints (q)	CPU time (sec)	Average no. of nodes	No. of nodes (range)	Optimal solution (node)
Matrix density = 0.4						
50	25	20	32.39	73	16 – 218	46
70	35	28	122.26	106	26 – 246	84
Matrix density = 0.3						
50	25	20	32.11	86	16 – 208	68
70	35	28	135.52	119	26 – 332	95
Matrix density = 0.2						
50	25	20	22.29	100	22 – 382	59
70	35	28	67.84	82	16 – 236	55

these densities. To avoid unbounded optimal solutions in the follower's subproblem if all coefficients in a column corresponding to a variable y_j with $d_{2j} < 0$ were negative, the first is multiplied by -1 ; then boundedness is checked and unbounded problems deleted.

The main parameters considered were the numbers n and m of leader and follower variables, p and q of first-level and second-level constraints and density d of the coefficient matrix. For each set of these values in each series of tests, 10 instances were solved. Mean (μ) and median (\hat{m}) values as well as standard deviations (σ) for the number of nodes in the branch and bound tree were computed and the corresponding CPU times (seconds) were recorded. XMP was used to solve all linear programs and to fix variables at their bounds.

The first class of experiments was aimed at streamlining the algorithm and assessing which of its features were the most useful. Numerous branching rules were tested and the following observations made:

- (i) The numbers of nodes generated and computation times were extremely sensitive to the chosen branching rule;
- (ii) Multiple-branching on logical relations involving few α_i was less efficient than dichotomous branching on the α_i ;
- (iii) Best results were obtained by a hybrid rule that begins with multiple branching and then switches to dichotomous branching when no more logical relations with few α_i are available;
- (iv) With all branching rules, difficulty of resolution varied greatly from problem to problem; often $\sigma > \hat{m}$ and $\hat{m} \ll \mu$. Many problems were easy to solve with

only a handful of nodes in the branch and bound tree, while a few required several thousand nodes.

In a second series of tests, the effect of heuristics was assessed by obtaining an upper bound on the reduction in computational effort they provided. In particular, a set of previously solved instances was solved again first using no heuristic and then assuming the optimal value to be known *a priori*. Although some problems remained difficult to solve, *a priori* knowledge of the best solution significantly decreased the computational effort (average CPU times were reduced by 21.3% to 42.2%).

In the next series of tests, Hansen et al. assessed the effect of penalties by solving the same problems without and with them. The results showed that:

- (i) Using penalties markedly reduced the number of nodes in the branch and bound tree (the average reduction in the number of nodes was between 25.0% and 68.3%);
- (ii) CPU times were not much affected (both increases and decreases were observed) except for the largest test problems in which case using penalties reduced mean CPU time by 45.8%.

In another class of experiments the HJS algorithm was compared to the BM algorithm. The results are given in Table 5.3 when no first level constraints are present; i.e., $p = 0$. A direct comparison with the complementarity approach of Júdice and Faustino was not possible because the code could not be obtained from the authors. From the table, the following observations can be made.

- (i) The HJS algorithm with branching rule BR5 always outperformed the BM algorithm. The same is true for branching rule BR6 with two exceptions.
- (ii) The ratios of computation times for algorithms BM and HJS with branching rule BR5 are between 1.4 and 9.6 when $d = 40\%$, 3.5 and 49.0 when $d = 17\%$, 23 and 47.1 when $d = 8\%$, but vary widely in each of these cases.
- (iii) This ratio increases with problem size.
- (iv) Similar conclusions hold even if no heuristic is used or if the optimal value of the linear BLPP is assumed to be known *a priori*.

The apparent reasons why the HJS algorithm is faster than the BM algorithm are stated as follows (probably in decreasing order of importance):

- (i) Use of logical relations (5.21a,b): for sparse problems some of these logical relations may have only one α ; and thus lead to problem simplification and possibly to an optimal solution without branching;
- (ii) Solution of LR and the dual of FR separately instead of jointly, thus reducing the size of the linear programs solved in various tests (algorithm BM includes the dual variables in the master program and hence contains q more variables than does the HJS algorithm);

Table 5.3 Comparison of HJS and BM algorithms on problems with no first level constraints

n	20		50		42		35		60		45	
m	30	20	20	28	28	28	35	35	30	36	45	36
q	20		28		28		28		36		36	
	nodes	CPU										
40% μ	175.2	130.8	67.6	148.2	158.4	306.9	579.0	1081	203.0	932.6	766.2	2957
BM	239.0	169.8	60.6	116.7	199.3	337.6	1244	2219	165.6	719.3	877.7	3469
σ	68	59.2	30	75.0	82	172.2	146	306.2	148	733.1	364	1259
40% μ	105.8	92.4	10.2	13.0	32.6	47.5	211.2	339.7	60.2	136.2	441.8	1292
BR5	195.8	159.0	20.5	19.3	44.1	59.9	449.4	692.2	58.2	127.2	520.2	1513
\hat{m}	5	9.6	1	3.6	9	15.1	41	80.5	43	92.7	81	244.3
40% μ	271.2	223.7	14.2	16.7	72.2	96.7	398.6	614.9	110.2	235.4	1638	4662
BR6	638.5	506.6	31.6	29.1	108.3	132.7	659.8	988.9	116.3	243.5	2274	6486
\hat{m}	5	9.6	1	3.6	9	14.9	71	122.6	61	129.7	117	345.5
17% μ	162.4	46.7	34.6	42.7	180.0	192.8	753.8	788.6	86.0	249.7	551.8	1098
BM	165.2	45.2	24.2	22.4	252.7	265.0	1080	1057	137.4	350.6	1460	2659
σ	28	12.3	20	28.2	32	39.5	252	275.7	36	116.8	80	267.2
17% μ	18.2	13.3	8.6	12.2	5.9	7.9	81.4	104.5	4.8	12.4	20.8	58.2
BR5	21.3	14.4	10.1	12.9	5.2	5.7	96.2	114.7	7.6	13.4	32.1	79.0
\hat{m}	5	5.9	3	5.2	3	6.4	17	30.4	1	5.9	5	17.5
17% μ	24.6	16.4	4.2	4.9	5.6	8.2	53.8	74.4	6.6	15.2	27.6	70.8
BR6	34.4	20.9	3.9	3.3	5.3	5.7	62.6	80.4	12.4	20.8	51.3	114.6
\hat{m}	5	5.9	3	3.8	3	6.4	15	26.7	1	5.9	5	17.5
8% μ							76.6	14.4	58.0	40.6	2290	1317
BM							98.2	14.4	60.6	44.2	4522	2511
σ							40	9.6	24	20.0	600	359.1
8% μ							5.2	6.4	6.0	9.0	30.4	58.6
BR5							4.5	4.0	5.8	6.5	42.1	65.2
\hat{m}							3	4.4	3	5.9	19	45.4
8% μ							5.4	6.3	6.0	8.6	17.6	38.4
BR6							4.9	4.0	5.7	6.2	15.3	27.9
\hat{m}							3	4.4	3	6.0	9	27.0

(iii) Use of hybrid branching rules;

(iv) Introduction of penalties.

Hansen et al. also considered BLPP instances where some of the second level constraints were transferred to the leader's problem. These instances were solved by the HJS algorithm and a version of the BM algorithm modified slightly to allow for first-level constraints with y variables. The results did not indicate any discernible pattern or change in computational difficulty. For those problems where the CPU time increased for the HJS algorithm the increase was restricted to a factor of 3 or less. Comparatively speaking, the HJS algorithm outperformed the BM algorithm in all cases with ratios of CPU times higher when there were first-level constraints (up from 2.83 to 53.3 on average) than when there were none.

Table 5.3 (continued)

n	70		60		50		70		60		70	
m	30		40		50		50		60		60	
q	40		48		40		48		48		52	
	nodes	CPU										
40% μ	77.4	519.5	128.8	434.4								
BM	53.9	342.5	97.2	293.8								
\hat{m}	54	383.0	92	302.2								
40% μ	17.2	54.2	48.6	160.8								
BR5	22.4	55.8	72.7	215.0								
\hat{m}	3	29.3	7	32.2								
40% μ	18.0	60.3	97.8	315.5								
BR6	30.8	96.4	134.5	422.4								
\hat{m}	7	29.3	33	97.5								
17% μ	260.6	926.2	1033	1740								
BM	506.8	1559	2498	3920								
\hat{m}	62	290.8	44	119.6								
17% μ	9.4	25.1	10.0	35.3								
BR5	10.4	21.3	11.9	29.5								
\hat{m}	5	14.1	1	12.9								
17% μ	9.6	25.7	8.4	30.8								
BR6	9.7	20.0	11.5	28.7								
\hat{m}	5	16.6	1	12.8								
8% μ	77.4	84.7	211.4	153.4	2863	2497	2114	3623	1985	4235	3664	7295
BM	62.8	76.2	147.7	118.7	4968	4559	3120	4998	2851	5855	3470	6333
\hat{m}	46	50.0	168	100.1	396	329.2	518	1205	662	1417	2138	4023
8% μ	5.8	10.8	6.4	18.0	17.6	56.8	30.4	112.3	24.0	126.0	26.2	154.8
BR5	5.3	9.3	3.6	9.4	12.8	35.7	31.3	98.5	29.4	126.6	26.2	140.8
\hat{m}	3	6.1	5	14.2	11	41.1	19	83.3	11	69.5	11	68.7
8% μ	5.2	9.5	7.6	20.6	22.6	67.0	34.2	121.3	36.4	174.6	22.6	133.7
BR6	4.0	6.7	5.0	13.0	14.9	42.8	29.4	95.6	50.6	205.9	20.3	109.4
\hat{m}	3	6.1	5	14.1	21	59.0	25	85.2	15	76.8	11	70.8

Sequential linear complementarity approach: Júdice and Faustino [J4] tested their SLCP algorithm against the Kuhn-Tucker approach of Bard and Moore by implementing their own version of the branch and bound BM algorithm. For the computations, they randomly generated two classes of problems. In the first class, denoted by TPN, all cost coefficients c_1 , d_1 and d_2 in the two objective functions were programmed to be nonnegative. The intent was to simulate a nonconflictual situation. In the second class, denoted by TPG, conflict was introduced by allowing some of the follower's cost coefficients in d_2 to be negative.

For each set of problems, 5 instances were generated and solved on a CDC CYBER 180-830. The first set of runs was designed to fine tune the SLCP algorithm and investigate the importance of several proposed modifications. The second set was for comparison purposes. The results are presented in Table 5.4. Output includes:

NI number of simplex pivot operations required by either algorithm

ND number of nodes required by the BM algorithm

- NS indication that the algorithm did not terminate within 20,000 pivots
 SP sparsity of second level constraint matrices ($\mathbf{A}_2, \mathbf{B}_2$)
 CPU CPU time in seconds

for the best (B), worst (W) and average (A) performance in terms of simplex pivot operations. Because the SLCP algorithm is only guaranteed of finding an ε -optimal solution, a column labeled “OPT” is included in the table to indicate whether or not a global optimum was found. The symbol Y (N) specifies when the solutions of the two algorithms agree (do not agree); i.e., when the SLCP algorithm has found the optimum. The value of ε on the same line as the Y or N means that the SLCP results are at least $\varepsilon\%$ of the optimum. When an algorithm did not terminate within 20,000 simplex pivots in each instance in a problem set, the number of times that it did so successfully for that problem set is written on the line marked “A.”

The results in Table 5.4 suggest that the BM algorithm is competitive with the SLCP algorithm for the BLPPs of smaller dimension contained in the first four problem sets but is noticeably less efficient for the larger problems. Of the 20 instances in TP6 and TP8, the BM algorithm was only able to solve 7 while SLCP appears to have solved 13 (the original paper is not clear on this). In total, the latter found the global minimum in more than 60% of the test problems.

In the final phase of their testing, Júdice and Faustino tried starting the BM algorithm with the best solution found by the SLCP method. Although the average performance of the former improved somewhat when started with an incumbent, in no instances was it able to find the optimum when it was previously unable to do so.

Penalty approach: White and Anandalingam [W4] compared their approach with the K th-best algorithm of Bialas and Karwan and the branch and bound code of Bard and Moore. Table 5.5 presents their results for 5 different sizes of randomly generated problems with follower constraints only (i.e., $p = 0$). Each line in the table represents the average of 10 instances so a total of 50 problems were attempted. All computations were performed on an ATT PC6300+ microcomputer with an Intel 80286 microprocessor and an 80287 math coprocessor. The linear programs arising in both the penalty function algorithm and the K th-best algorithm were solved with LINDO. The main drivers containing the logic of the algorithms were written in PASCAL. Recall that the Bard-Moore code uses XMP to solve the linear programs.

In generating the problem data, the objective function coefficients were randomly selected from the interval $[-10, +10]$. No attempt was made to see if the pairwise component values of \mathbf{d}_1 and \mathbf{d}_2 were the same or close. The constraint coefficients were randomly selected from the interval $[-5, +5]$ while matrix densities varied from

Table 5.4 Computational results for SLC approach

Problem	Dimensions				SLCP Algorithm			BM Algorithm			
	n	m	q	SP		NI	CPU	OPT	ND	NI	CPU
TPN2	50	30	30	16.7%	B	75	2.0	0, Y	9	7108	2.7
					A	299	8.7	3	41	427	12.8
					W	774	23.3	0, Y	99	1239	38.7
TPG2	50	30	30	16.7%	B	205	6.7	0.1, Y	13	138	4.2
					A	406	13.3	4	49	373	11.4
					W	1032	32.3	0, Y	173	1023	31.6
TPN4	120	50	50	15%	B	99	5.5	0, Y	3	93	4.8
					A	1719	111	5	135	4309	29
					W	4116	270	1, Y	399	15,284	1045
TPG4	120	50	50	15%	B	435	39.1	0, Y	11	478	35.8
					A	2681	190	5	159	3744	261
					W	8275	565	1, Y	407	10,245	738
TPN6	300	100	100	7.1%	B	517	67	0.1, Y	7	355	44.7
					A	5828	746	4	—	3	—
					W	14,799	1873	0, ?	—	NS	—
TPG6	300	100	100	7.1%	B	1298	287	0.1, Y	5	789	112
					A	6664	943	4	—	0	—
					W	25,590	3266	0, ?	—	NS	—
TPN8	250	150	150	7.1%	B	8289	1240	5, ?	—	NS	—
					A	—	—	—	—	0	—
					W	NS	—	—	—	NS	—
TPG8	250	150	150	7.1%	B	5444	1500	0.1, N	129	9171	1803
					A	10,140	2282	4	—	1	—
					W	16,779	3410	5, ?	—	NS	—

Table 5.5 Comparative performance of penalty function algorithm

Leader variables (n)	Follower variables (m)	Constraints (q)	Kth-best (CPU sec)	BM algorithm (CPU sec)	Penalty method (CPU sec)
5	10	6	127.6	55.2	59.2
6	14	8	111.7	81.9	87.2
8	17	10	186.2	102.1	102.7
15	30	20	1200.9	151.7	167.8
50	50	100	—	1043.9	1821.3

10% to 75% nonzero terms. The authors did not identify which of these instances took the longest to solve or which ones produced degenerate solutions.

The first test was to ensure that the problem instances were feasible from the point of view of the leader. Infeasible problems (around 50%) were discarded and not included in the results. To start the penalty function method, a large value of K must be chosen. This was done through experimentation. First, a “very” large value (say 1000) and a “reasonable” value (say 10) were chosen in turn and the algorithm run. If the final results were very different, then K was set at 10 and each time Step 10 was reached, K was increased by $\delta = 1$ (i.e., a 10% increase) until optimality changed. If the final results for $K = 10$ and $K = 1000$ were the same, the solution was taken as optimal. This test involves an extra step and additional computational time; however, the test reduced the average time of convergence. The choice of δ also influences computational time but its impact was not evaluated.

As can be seen in Table 5.5, the penalty function method does not do as well as the BM algorithm in terms of CPU time but is comparable for the smaller problems. For the larger problems the BM algorithm is superior. Both easily outperformed the K th-best algorithm which was not thought to be very efficient.

LINEAR BILEVEL PROGRAMMING: DISCRETE VARIABLES

6.1 INTRODUCTION

In many optimization problems a subset of the variables is restricted to take on discrete values. This can complicate the problem by several orders of magnitude and render all but the smallest instances unsolvable. In this chapter, we investigate several versions of the linear BLPP with discrete variables and discuss the prospect of finding global optima using branch and bound.

To specify the model, let \mathbf{x}_1 be an n_1 -dimensional vector of continuous variables and \mathbf{x}_2 be an n_2 -dimensional vector of discrete variables, where $\mathbf{x} \triangleq (\mathbf{x}_1, \mathbf{x}_2)$ and $n = n_1 + n_2$. Similarly define \mathbf{y}_1 as an m_1 -dimensional vector of continuous variables and \mathbf{y}_2 as an m_2 -dimensional vector of discrete variables, where $\mathbf{y} \triangleq (\mathbf{y}_1, \mathbf{y}_2)$ and $m = m_1 + m_2$. This leads to

$$\min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}) = \mathbf{c}_{11}\mathbf{x}_1 + \mathbf{c}_{12}\mathbf{x}_2 + \mathbf{d}_{11}\mathbf{y}_1 + \mathbf{d}_{12}\mathbf{y}_2 \quad (6.1a)$$

$$\text{subject to } \mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 + \mathbf{B}_{11}\mathbf{y}_1 + \mathbf{B}_{12}\mathbf{y}_2 \leq \mathbf{b}_1 \quad (6.1b)$$

$$\mathbf{x}_1 \geq \mathbf{0}, \mathbf{x}_2 \geq \mathbf{0} \text{ integer} \quad (6.1c)$$

$$\min_{\mathbf{y}} f(\mathbf{y}) = \mathbf{d}_{21}\mathbf{y}_1 + \mathbf{d}_{22}\mathbf{y}_2 \quad (6.1d)$$

$$\text{subject to } \mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2 + \mathbf{B}_{21}\mathbf{y}_1 + \mathbf{B}_{22}\mathbf{y}_2 \leq \mathbf{b}_2 \quad (6.1e)$$

$$\mathbf{y}_1 \geq \mathbf{0}, \mathbf{y}_2 \geq \mathbf{0} \text{ integer} \quad (6.1f)$$

where all vectors and matrices are of conformal dimension and the linear terms in \mathbf{x} have been omitted from the follower's objective function in (6.1d). Note that it may be desirable to explicitly include additional restrictions on the variables such as upper and lower bounds. In that case, we would have $\mathbf{x} \in X = \{\mathbf{x} : l_j^1 \leq x_j \leq u_j^1, j = 1, \dots, n\}$ and $\mathbf{y} \in Y = \{\mathbf{y} : l_j^2 \leq y_j \leq u_j^2, j = 1, \dots, m\}$.

In the next section, we investigate the properties of various versions of the zero-one linear BLPP where each player controls either all continuous or all binary variables. This is followed by a study of the more general case where the variables may be mixed, as in (6.1c) and (6.1f). The last two sections present algorithms for the mixed-integer formulation and the discrete formulation, respectively.

6.2 PROPERTIES OF THE ZERO-ONE LINEAR BLPP

The aim of this section is to analyze the properties of the linear BLPP when some or all of the variables are restricted to binary values. The majority of material closely follows the work of Vicente et al. [V3]. We begin by studying the geometry of the feasible set and discuss the existence of optimal solutions. Equivalences are then established between the different classes of zero-one linear bilevel programs and a particular linear three-level program. These equivalences are based on the introduction of a concave penalty function and can be used to design penalty methods for solving the corresponding BLPPs. While the results are specific to the zero-one problem, it should be noted that any bounded integer variable can be expressed as the weighted sum of binary variables (see eq. (3.7)). Nevertheless, this transformation is inefficient from an algorithmic point of view and is only recommended when no better alternative is available.

Because our focus is on specific instances of (6.1), it will be convenient to consider the problem in the form of (5.1) without reference to which variables are continuous and which are discrete; i.e.,

$$\begin{aligned} \min_{\mathbf{x} \in X} F(\mathbf{x}, \mathbf{y}) &= \mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y} \\ \text{subject to } \mathbf{A}_1 \mathbf{x} + \mathbf{B}_1 \mathbf{y} &\leq \mathbf{b}_1 \\ \min_{\mathbf{y} \in Y} f(\mathbf{y}) &= \mathbf{d}_2 \mathbf{y} \\ \text{subject to } \mathbf{A}_2 \mathbf{x} + \mathbf{B}_2 \mathbf{y} &\leq \mathbf{b}_2 \end{aligned}$$

where $\mathbf{c}_1 \in R^n$, $\mathbf{d}_1, \mathbf{d}_2 \in R^m$, $\mathbf{b}_1 \in R^p$, $\mathbf{b}_2 \in R^q$, $\mathbf{A}_1 \in R^{p \times n}$, $\mathbf{B}_1 \in R^{p \times m}$, $\mathbf{A}_2 \in R^{q \times n}$, $\mathbf{B}_2 \in R^{q \times m}$, $X \subseteq R^n$ and $Y \subseteq R^m$.

In addition to the definitions in Section 5.1, let

$$S_L(\mathbf{y}) = \{\mathbf{x} \in X : \mathbf{A}_2 \mathbf{x} \leq \mathbf{b}_2 - \mathbf{B}_2 \mathbf{y}\},$$

for all values of $\mathbf{y} \in Y$, and

$$S_U = \{(\mathbf{x}, \mathbf{y}) : \mathbf{A}_1 \mathbf{x} + \mathbf{B}_1 \mathbf{y} \leq \mathbf{b}_1\}$$

For each $\bar{\mathbf{x}} \in X$, it will be assumed that the optimal solution of the lower-level problem is unique. Along with the linear bilevel programming problem (L-BLPP) where $X = R^n$ and $Y = R^m$, we are interested in the following three models:

- (i) discrete linear bilevel programming problem (DL-BLPP), where $X = B^n$ and $Y = B^m$;
- (ii) discrete-continuous linear bilevel programming problem (DCL-BLPP), where $X = B^n$ and $Y = R^m$; and
- (iii) continuous-discrete linear bilevel programming problem (CDL-BLPP), where $X = R^n$ and $Y = B^m$.

The existence of optimal solutions for these problems depends on the presence or absence of upper-level constraints. The following property is valid for L-BLPP, DL-BLPP, DCL-BLPP and CDL-BLPP, and can be confirmed from their definitions for the more general case where $X = Z^n$ and $Y = Z^m$.

Property 6.2.1 If $S_U = R^{n+m}$ then IR is nonempty if $S \neq \emptyset$. If $S_U \neq R^{n+m}$ then IR is nonempty if there exists an $\bar{\mathbf{x}} \in X$ such that $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in S_U$.

The inducible regions associated with these problems imply two simple relationships that can be used for computing lower and upper bounds, once again for $X = Z^n$ and $Y = Z^m$.

Property 6.2.2 The inducible regions of DCL-BLPP and DL-BLPP are respectively included in the inducible regions of L-BLPP and CDL-BLPP.

The proof of Property 6.2.2 is straightforward; the property itself is demonstrated in the following example.

Example 6.2.1 Let $n = m = 1$, $q = 4$, $S_U = R^2$, $d_2 = 1$,

$$\mathbf{A}_2 = \begin{bmatrix} 1 \\ -1 \\ 5 \\ -5 \end{bmatrix}, \quad \mathbf{B}_2 = \begin{bmatrix} 1 \\ 1 \\ -4 \\ -4 \end{bmatrix}, \quad \text{and} \quad \mathbf{b}_2 = \begin{bmatrix} 2 \\ 2 \\ 10 \\ 10 \end{bmatrix}.$$

Figure 6.1 depicts the inducible regions associated with the four problems under investigation for these data. The following observations can be made:

- (i) the optimal solution of DL-BLPP may be an interior point of the set S ;
- (ii) the inducible region of CDL-BLPP may be a noncompact set and thus the problem may have no optimal solution even when $IR \neq \emptyset$. This can be illustrated

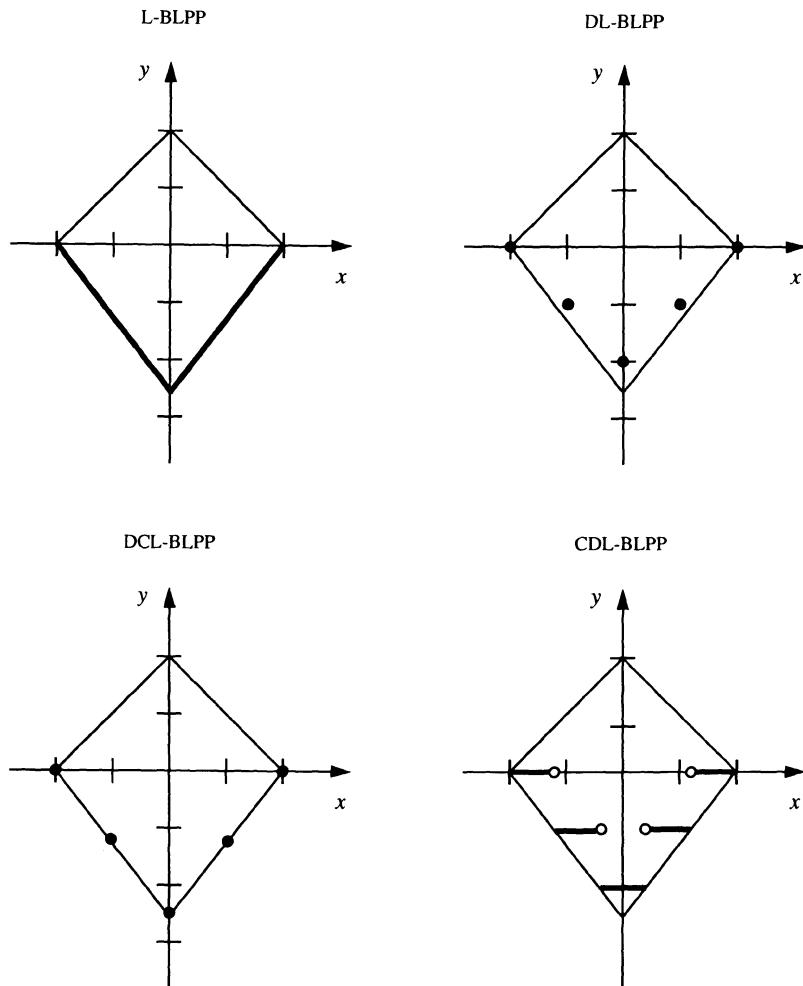


Figure 6.1 Inducible regions for several versions of the linear BLPP

by considering the CDL-BLPP with $n = m = 1$, $S_U = R^2$, $q = 3$, $c_1 = d_1 = 1$, $d_2 = -1$,

$$\mathbf{A}_2 = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}, \quad \mathbf{B}_2 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, \quad \text{and} \quad \mathbf{b}_2 = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}.$$

For these data, $IR = \{(x, y) : \{0, 2\}, \{(0, \frac{1}{2}], 1\}, \{(\frac{1}{2}, 1], 0\}\}$. The leader would like to pick x as small as possible but greater than $\frac{1}{2}$ to force the follower to choose $y = 0$. This would produce the best result for the leader. However, if the leader

chooses $x = \frac{1}{2}$, the follower chooses $y = 1$, producing a degradation in the leader's objective function. Hence, without some form of cooperation there is no solution to this CDL-BLPP.

We now address the existence of optimal solutions.

Property 6.2.3 For the L-BLPP, let S be a bounded set, i.e., a polytope. If $S_U = R^{n+m}$ then L-BLPP, DL-BLPP and DCL-BLPP have an optimal solution if $S \neq \emptyset$. If $S_U \neq R^{n+m}$ then L-BLPP, DL-BLPP and DCL-BLPP have an optimal solution if there exists an $\bar{x} \in X$ such that $(\bar{x}, \bar{y}) \in S_U$.

This property states that if S is bounded sufficient conditions for the existence of optimal solutions coincide with the conditions under which the inducible region is nonempty. The proof is trivial for DL-BLPP and DCL-BLPP since the inducible regions are nonempty finite discrete sets. For a discussion of the L-BLPP case, see Section 5.1 or [E1]. What remains is the investigation of the inducible region of CDL-BLPP and the characterization of its optimal solution.

Lemma 6.2.1 Consider the problem CDL-BLPP, where $S_U = R^{n+m}$ and $S \neq \emptyset$. Then IR is composed of a finite union of quasi-polyhedral sets, i.e., a set whose closure is a polyhedral set.

Proof: Let Y_\emptyset be the subset of Y consisting of all points $\mathbf{y} \in Y$ such that $S_L(\mathbf{y}) \neq \emptyset$. Suppose that $Y_\emptyset = \{\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(k)}\}$ for some positive integer k and assume without loss of generality that

$$\mathbf{d}_2 \mathbf{y}^{(i)} \leq \mathbf{d}_2 \mathbf{y}^{(i+1)}, \quad i = 1, \dots, k-1$$

It is a simple matter to see that $P^{(1)} = S_L(\mathbf{y}^{(1)}) \times \{\mathbf{y}^{(1)}\}$ is a polyhedral set and is included in IR . If $S_L(\mathbf{y}^{(2)}) - S_L(\mathbf{y}^{(1)}) \neq \emptyset$, then all points in $S_L(\mathbf{y}^{(2)}) - S_L(\mathbf{y}^{(1)})$ parameterize lower-level problems that have $\mathbf{y}^{(2)}$ as an optimal solution. Hence $P^{(2)} = (S_L(\mathbf{y}^{(2)}) - S_L(\mathbf{y}^{(1)})) \times \{\mathbf{y}^{(2)}\}$ is also included in the inducible region, and the closure of $P^{(2)}$ is a finite union of polyhedral sets.

Continuing in this manner, the following partition of the inducible region is obtained:

$$IR = \bigcup_{i \in I} P^{(i)}$$

where $P^{(i)} \triangleq (S_L(\mathbf{y}^{(i)}) - \bigcup_{j=1}^{i-1} S_L(\mathbf{y}^{(j)})) \times \{\mathbf{y}^{(i)}\}$ for all $i \in I = \{i \in \{1, \dots, k\} : S_L(\mathbf{y}^{(i)}) - \bigcup_{j=1}^{i-1} S_L(\mathbf{y}^{(j)}) \neq \emptyset\}$ and is such that the closure of $P^{(i)}$ is a finite union of polyhedral sets. ■

The case where $S_U \neq R^{n+m}$ is addressed in the same way. In fact, we only need to assume that there exists an $\bar{x} \in X$ such that $(\bar{x}, \bar{y}) \in S_U$ and define the sets $P^{(i)}$ as $(S_L(\mathbf{y}^{(i)}) - \bigcup_{j=1}^{i-1} S_L(\mathbf{y}^{(j)})) \times \{\mathbf{y}^{(i)}\} \cap S_U$ (in case they are nonempty).

It has been shown above that CDL-BLPP may not have an optimal solution even when $S_U = R^{n+m}$ and the inducible region is nonempty. When an optimal solution exists it is possible to guarantee that it is a boundary point of the set S .

Theorem 6.2.1 Let $S_U = R^{n+m}$, $S \neq \emptyset$ and suppose that there exists an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ to CDL-BLPP. Then $(\mathbf{x}^*, \mathbf{y}^*)$ is a boundary (bd) point of S .

Proof: By Lemma 6.2.1 there exists an $i \in I$ such that $(\mathbf{x}^*, \mathbf{y}^*) \in P^{(i)}$ with $\mathbf{y}^* = \mathbf{y}^{(i)}$. Now, given that $P^{(i)}$ is composed of a finite union of quasi-polyhedral sets, we have $(\mathbf{x}^*, \mathbf{y}^{(i)}) \in \mathcal{P} \times \{\mathbf{y}^{(i)}\}$, where $\mathcal{P} \subset R^n$ is a quasi-polyhedral set. Also, the linearity of the upper-level function implies that $\mathbf{x}^* \in \text{bd}(\mathcal{P}) \cap \mathcal{P}$. Hence $\mathbf{x}^* \in \text{bd}(S'_L(\mathbf{y}^{(i)})) \cap S'_L(\mathbf{y}^{(i)})$, where $S'_L(\mathbf{y}^{(i)})$ is given by $S'_L(\mathbf{y}^{(i)}) = S_L(\mathbf{y}^{(i)}) - \bigcup_{j=1}^{i-1} S_L(\mathbf{y}^{(j)})$.

From the definition of the set $S'_L(\mathbf{y}^{(i)})$ it follows that

$$\text{bd}\left(S'_L(\mathbf{y}^{(i)})\right) \subset \text{bd}\left(S_L(\mathbf{y}^{(i)})\right) \cup \text{bd}\left(\bigcup_{j=1}^{i-1} S_L(\mathbf{y}^{(j)})\right).$$

Since $\mathbf{x}^* \notin \text{bd}\left(\bigcup_{j=1}^{i-1} S_L(\mathbf{y}^{(j)})\right)$, then $\mathbf{x}^* \in \text{bd}(S_L(\mathbf{y}^{(i)}))$ and $(\mathbf{x}^*, \mathbf{y}^*) \in \text{bd}(S_L(\mathbf{y}^{(i)})) \times \{\mathbf{y}^{(i)}\} \subset \text{bd}(S)$. This completes the proof. ■

6.2.1 Reductions to Linear Three-Level Programs

Let $X = \{0, 1\}^n$, $Y = \{0, 1\}^m$ and e_x and e_y be vectors of ones with dimensions n and m , respectively. Consider the standard optimization problem:

$$\begin{array}{ll} \min_{\mathbf{x} \in X} & c_1 \mathbf{x} + d_1 \mathbf{y} \\ \text{subject to} & (\mathbf{x}, \mathbf{y}) \in \mathcal{P} \end{array} \quad (6.2)$$

where $\mathcal{P} = \bigcup_{i=1}^{\ell} P_i$ such that P_i is a polyhedral set for $i = 1, \dots, \ell$, and ℓ is a positive integer. It can be assumed without loss of generality that $c_1 \leq 0$ and $d_1 \leq 0$. If the i th component of c_1 (respectively d_1) is positive, for example, it is always possible to define a new variable $x'_i = 1 - x_i$ ($y'_i = 1 - y_i$) and make the appropriate substitutions to yield an equivalent program satisfying this requirement. Denote the optimal solution of problem (6.2) by $(\mathbf{x}^*, \mathbf{y}^*)$, and let $\theta : R^n \rightarrow R$ be a continuous function such that $\theta(\mathbf{x}) \geq 0$ for all $\mathbf{0} \leq \mathbf{x} \leq e_x$ and $\theta(\mathbf{x}) = 0$ if and only if $\mathbf{x} \in X$. The following theorem is a generalization of a result established by Kalantari and Rosen [K1] for integer linear programs.

Theorem 6.2.2 If θ is a concave function there exists a positive real number M such that (6.2) and the following problem

$$\begin{aligned} \min \quad & c_1\mathbf{x} + d_1\mathbf{y} + M\theta(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{0} \leq \mathbf{x} \leq \mathbf{e}_x \\ & (\mathbf{x}, \mathbf{y}) \in \mathcal{P} \end{aligned} \tag{6.3}$$

have the same optimal solutions.

Proof: Define \bar{E} as the set of all extreme points of the polyhedra $\{(\mathbf{x}, \mathbf{y}) : \mathbf{0} \leq \mathbf{x} \leq \mathbf{e}_x\} \cap P_i$ for $i = 1, \dots, \ell$ and E as the set $\{(\mathbf{x}, \mathbf{y}) : \mathbf{0} \leq \mathbf{x} \leq \mathbf{e}_x\} \cap \mathcal{P}$. Let $(\mathbf{x}^0, \mathbf{y}^0)$ be the optimal solution of the problem

$$\begin{aligned} \min \quad & c_1\mathbf{x} + d_1\mathbf{y} \\ \text{subject to} \quad & \mathbf{0} \leq \mathbf{x} \leq \mathbf{e}_x \\ & (\mathbf{x}, \mathbf{y}) \in \mathcal{P} \end{aligned}$$

and $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = \operatorname{argmin}\{\theta(\mathbf{x}) : (\mathbf{x}, \mathbf{y}) \in \bar{E}\}$. If $\theta(\bar{\mathbf{x}}) = 0$ then the result is true for any positive value of M . Suppose now that $\theta(\bar{\mathbf{x}}) > 0$. Furthermore let $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ be the optimal solution of (6.3) for $M > M_0 = (-c_1\mathbf{x}^0 - d_1\mathbf{y}^0)/\theta(\bar{\mathbf{x}})$. Since θ is a concave function, $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ belongs to \bar{E} and $\theta(\tilde{\mathbf{x}}) \leq \theta(\bar{\mathbf{x}})$ by definition of $\bar{\mathbf{x}}$.

Suppose now that $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ is an element of $\bar{E} \setminus (X \times R^m \cap E)$; i.e., $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ is a noninteger extreme point of P_i for some $i \in \{1, \dots, \ell\}$. Because $M > M_0$ and $\theta(\tilde{\mathbf{x}}) \leq \theta(\bar{\mathbf{x}})$, we have

$$c_1\mathbf{x}^* + d_1\mathbf{y}^* - (c_1\tilde{\mathbf{x}} + d_1\tilde{\mathbf{y}}) - M\theta(\tilde{\mathbf{x}}) < c_1\mathbf{x}^* + d_1\mathbf{y}^* - (c_1\tilde{\mathbf{x}} + d_1\tilde{\mathbf{y}}) + c_1\mathbf{x}^0 + d_1\mathbf{y}^0$$

But $c_1 \leq 0$, $d_1 \leq 0$ and $c_1\tilde{\mathbf{x}} + d_1\tilde{\mathbf{y}} \geq c_1\mathbf{x}^0 + d_1\mathbf{y}^0$. Hence

$$c_1\mathbf{x}^* + d_1\mathbf{y}^* < c_1\tilde{\mathbf{x}} + d_1\tilde{\mathbf{y}} + M\theta(\tilde{\mathbf{x}})$$

which is a contradiction. Thus $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \in X \times R^m \cap E$ and since $(\mathbf{x}^*, \mathbf{y}^*)$ is also in $X \times R^m \cap E$, we have established that

$$c_1\tilde{\mathbf{x}} + d_1\tilde{\mathbf{y}} + M\theta(\tilde{\mathbf{x}}) \leq c_1\mathbf{x}^* + d_1\mathbf{y}^* + M\theta(\mathbf{x}^*).$$

This, in turn, is equivalent to

$$c_1\tilde{\mathbf{x}} + d_1\tilde{\mathbf{y}} \leq c_1\mathbf{x}^* + d_1\mathbf{y}^*$$

which completes the proof. ■

DCL-BLPP and DL-BLPP

The discrete-continuous bilevel programming problem can be stated as

$$\begin{aligned} & \min_{\mathbf{x} \in X} c_1 \mathbf{x} + d_1 \mathbf{y} \\ & \text{subject to } A_1 \mathbf{x} + B_1 \mathbf{y} \leq b_1 \\ & \quad \min_{\mathbf{y}} d_2 \mathbf{y} \\ & \quad \text{subject to } A_2 \mathbf{x} + B_2 \mathbf{y} \leq b_2 \end{aligned}$$

The next theorem shows that this problem is equivalent to a linear bilevel programming problem.

Theorem 6.2.3 There exists a positive real M such that DCL-BLPP and the linear bilevel programming problem, L-BLPP(M),

$$\begin{aligned} & \min_{\mathbf{x}} c_1 \mathbf{x} + d_1 \mathbf{y} + M e_x^T \mathbf{u} \\ & \text{subject to } A_1 \mathbf{x} + B_1 \mathbf{y} \leq b_1 \\ & \quad \mathbf{0} \leq \mathbf{x} \leq e_x \\ & \quad \min_{\mathbf{y}, \mathbf{u}} d_2 \mathbf{y} - e_x^T \mathbf{u} \\ & \quad \text{subject to } A_2 \mathbf{x} + B_2 \mathbf{y} \leq b_2 \\ & \quad \mathbf{u} \leq \mathbf{x} \\ & \quad \mathbf{u} \leq e_x - \mathbf{x} \end{aligned}$$

have the same optimal solutions.

Proof: Recall a result of linear bilevel programming which states that the set $\{(\mathbf{x}, \mathbf{y}) : A_1 \mathbf{x} + B_1 \mathbf{y} \leq b_1, \mathbf{y} \in \arg \min(d_2 \mathbf{y} : B_2 \mathbf{y} \leq b_2 - A_2 \mathbf{x})\}$ is a finite union of polyhedral sets. Hence DCL-BLPP is a special case of problem (6.2) mentioned in Theorem 6.2.2. Now, if we let $\theta(\mathbf{x}) = \sum_{j=1}^n \min\{x_j, 1 - x_j\}$, which is a concave function in \mathbf{x} , we can apply this theorem to assure the existence of a real positive M such that DCL-BLPP and

$$\begin{aligned} & \min_{\mathbf{x}} c_1 \mathbf{x} + d_1 \mathbf{y} + M \sum_{j=1}^n \min\{x_j, 1 - x_j\} \\ & \text{subject to } A_1 \mathbf{x} + B_1 \mathbf{y} \leq b_1 \\ & \quad \mathbf{0} \leq \mathbf{x} \leq e_x \\ & \quad \min_{\mathbf{y}} d_2 \mathbf{y} \\ & \quad \text{subject to } A_2 \mathbf{x} + B_2 \mathbf{y} \leq b_2 \end{aligned}$$

have the same optimal solutions. However, this latter problem is the same as L-BLPP(M), since lower-level optimality in L-BLPP(M) enforces u_j to be equal to $\min\{x_j, 1 - x_j\}$ for all $j \in \{1, \dots, n\}$. ■

Alternative ways of reformulating DCL-BLPP can be obtained by choosing different functions $\theta(\mathbf{x})$. For instance, the following quadratic-linear bilevel programming problem, QL-BLPP(M)

$$\begin{aligned} & \min_{\mathbf{x}} \mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y} + M \mathbf{x}^T (\mathbf{e}_x - \mathbf{x}) \\ \text{subject to } & \mathbf{A}_1 \mathbf{x} + \mathbf{B}_1 \mathbf{y} \leq \mathbf{b}_1 \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{e}_x \\ & \min_{\mathbf{y}} \mathbf{d}_2 \mathbf{y} \\ \text{subject to } & \mathbf{A}_2 \mathbf{x} + \mathbf{B}_2 \mathbf{y} \leq \mathbf{b}_2 \end{aligned}$$

also has the same optimal solution of DCL-BLPP. Although the upper-level objective function of QL-BLPP(M) is concave, this problem has n fewer lower-level variables and $2n$ fewer lower-level constraints than L-BLPP(M). The benefits and drawbacks of the latter reformulation over the former are discussed in Section 6.2.2.

Now consider DL-BLPP with the sets $X = \{0, 1\}^n$ and $Y = \{0, 1\}^m$.

$$\begin{aligned} & \min_{\mathbf{x} \in X} \mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y} \\ \text{subject to } & \mathbf{A}_1 \mathbf{x} + \mathbf{B}_1 \mathbf{y} \leq \mathbf{b}_1 \\ & \min_{\mathbf{y} \in Y} \mathbf{d}_2 \mathbf{y} \\ \text{subject to } & \mathbf{A}_2 \mathbf{x} + \mathbf{B}_2 \mathbf{y} \leq \mathbf{b}_2 \end{aligned}$$

The next theorem shows that this problem can always be reduced to a linear three-level program.

Theorem 6.2.4 There exist two positive real numbers M_x and M_y such that DL-BLPP and the following linear three-level program:

$$\begin{aligned} & \min_{\mathbf{x}} \mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y} + M_x \mathbf{e}_x^T \mathbf{u} \\ \text{subject to } & \mathbf{A}_1 \mathbf{x} + \mathbf{B}_1 \mathbf{y} \leq \mathbf{b}_1 \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{e}_x \\ & \min_{\mathbf{u}, \mathbf{y}} \mathbf{d}_2 \mathbf{y} - \mathbf{e}_x^T \mathbf{u} + M_y \mathbf{e}_y^T \mathbf{v} \\ \text{subject to } & \mathbf{A}_2 \mathbf{x} + \mathbf{B}_2 \mathbf{y} \leq \mathbf{b}_2 \\ & \mathbf{0} \leq \mathbf{y} \leq \mathbf{e}_y \\ & \mathbf{u} \leqq \mathbf{x} \\ & \mathbf{u} \leqq \mathbf{e}_x - \mathbf{x} \\ & \min_{\mathbf{v}} -\mathbf{e}_y^T \mathbf{v} \\ \text{subject to } & \mathbf{v} \leqq \mathbf{y} \\ & \mathbf{v} \leqq \mathbf{e}_y - \mathbf{y} \end{aligned} \tag{6.4}$$

have the same optimal solutions.

Proof: It follows from the definition of DL-BLPP that its inducible region is a set of points. Thus the feasible set of DL-BLPP is a finite union of polyhedral sets so Theorem 6.2.2 can be applied to show that there exists a positive real M_x such that DL-BLPP and the problem

$$\begin{aligned}
 & \min_{\mathbf{x}} \mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y} + M_x \mathbf{e}_x^T \mathbf{u} \\
 & \text{subject to } \mathbf{A}_1 \mathbf{x} + \mathbf{B}_1 \mathbf{y} \leq \mathbf{b}_1 \\
 & \quad \mathbf{0} \leqq \mathbf{x} \leqq \mathbf{e}_x \\
 & \quad \min_{\mathbf{u}, \mathbf{y} \in Y} \mathbf{d}_2 \mathbf{y} - \mathbf{e}_x^T \mathbf{u} \\
 & \quad \text{subject to } \mathbf{A}_2 \mathbf{x} + \mathbf{B}_2 \mathbf{y} \leq \mathbf{b}_2 \\
 & \quad \quad \mathbf{u} \leqq \mathbf{x} \\
 & \quad \quad \mathbf{u} \leqq \mathbf{e}_x - \mathbf{x}
 \end{aligned} \tag{6.5}$$

have the same optimal solutions.

Because X is a finite set we can now apply Theorem 6.2.2 to the lower-level problem in (6.5) for $\mathbf{x} \in X$ fixed. Hence there exists a positive real $M_y(\mathbf{x})$ such that the latter problem and the BLPP defined by the second and third levels in (6.4) have the same optimal solutions for each $\mathbf{x} \in X$. The proof is completed by setting $M_y = \max_{\mathbf{x} \in X'} M_y(\mathbf{x})$, where X' is given by $\{\mathbf{x} \in X : (\mathbf{x}, \mathbf{y}) \in S\}$. ■

CDL-BLPP

By way of example, we now show that CDL-BLPP, in contrast to DCL-BLPP and DL-BLPP, may not be equivalent to a linear multilevel program.

$$\begin{aligned}
 & \min_x -3x - y \\
 & \text{subject to } x \leqq \frac{3}{4} \\
 & \quad \min_{y \in \{0,1\}} -y \\
 & \quad \text{subject to } 2x + y \leqq 2
 \end{aligned} \tag{6.6}$$

Furthermore, let $M > 0$ and consider the linear three-level program

$$\begin{aligned}
 & \min_x -3x - y \\
 \text{subject to } & x \leq \frac{3}{4} \\
 & \min_y -y + Mv \\
 \text{subject to } & 2x + y \leq 2 \\
 & 0 \leq y \leq 1 \\
 & \min_v -v \\
 \text{subject to } & v \leq y \\
 & v \leq 1 - y
 \end{aligned} \tag{6.7}$$

The inducible region of problem (6.6) is given by

$$[0, \frac{1}{2}] \times \{1\} \cup [\frac{1}{2}, \frac{3}{4}] \times \{0\},$$

and its optimal solution is attained at $(\frac{1}{2}, 1)$. Next it is shown that for each $M > 0$ the x and y components of the optimal solution of the linear three-level program (6.7) are different from this optimal solution.

We start by analyzing the subset of the inducible region of (6.7) containing all the points (x, y, v) such that $x \geq \frac{1}{2}$. As mentioned, the BLPP defined by the second and third levels in (6.7) can be written as:

$$\begin{aligned}
 & \min_y -y + M \min\{y, 1 - y\} \\
 \text{subject to } & 2x + y \leq 2 \\
 & 0 \leq y \leq 1
 \end{aligned} \tag{6.8}$$

The following two cases may occur.

(i) If $0 \leq y \leq \frac{1}{2}$, then $v = \min\{y, 1 - y\} = y$ and problem (6.8) takes the form:

$$\begin{aligned}
 & \min_y (-1 + M)y \\
 \text{subject to } & 0 \leq y \leq \min\{\frac{1}{2}, 2 - 2x\} = \frac{1}{2}
 \end{aligned}$$

(ii) If $\frac{1}{2} \leq y \leq 1$, $v = \min\{y, 1 - y\} = 1 - y$ and we can write (6.8) as

$$\begin{aligned}
 & \min_y (-1 - M)y + M \\
 \text{subject to } & \frac{1}{2} \leq y \leq \min\{1, 2 - 2x\} = 2 - 2x
 \end{aligned}$$

Based on these equivalences, we can find the optimal solution of (6.7). In so doing, two cases must be considered, depending on the value of M .

- (i) If $0 < M \leq 1$, the subset of the inducible region containing all points (x, y, v) such that $x \geq \frac{1}{2}$ is given by

$$\{(x, y, v) : 2x + y = 2, \frac{1}{2} \leq x \leq \frac{3}{4}, v = 1 - y\}.$$

Hence the optimal solution of (6.7) is attained at the point $(x^*, y^*, v^*) = (\frac{3}{4}, \frac{1}{2}, \frac{1}{2})$.

- (ii) If $M \geq 1$, the same subset of the inducible region is the union of the sets

$$\left\{ (x, y, v) : 2x + y = 2, \frac{1}{2} \leq x \leq \frac{1}{2} + \frac{1}{2M+2}, v = 1 - y \right\}$$

and

$$\left\{ (x, y, v) : \frac{1}{2} + \frac{1}{2M+2} < x \leq \frac{3}{4}, y = 0, v = y \right\}$$

Thus the optimal solution of (6.7) is given by

$$(x^*, y^*, v^*) = \left(\frac{1}{2} + \frac{1}{2M+2}, 1 - \frac{1}{1+M}, 1 - \frac{1}{1+M} \right)$$

This example shows that there exists no finite value of M for which problems (6.6) and (6.7) have the same optimal solutions.

Although it is not possible to guarantee the existence of a finite value of the penalty parameter in this case, we observe that if $\{M_k\}$ is a sequence of positive values satisfying $\lim_k M_k = +\infty$, then the corresponding sequence of optimal solutions $\{(x^{(k)}, y^{(k)}, v^{(k)})\}$ of (6.7) satisfies $\lim_k (x^{(k)}, y^{(k)}) = (x^*, y^*)$, where (x^*, y^*) is an optimal solution of (6.6).

This result is valid for any CDL-BLPP. To show this, we restate CDL-BLPP as

$$\begin{aligned} & \min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}) \\ \text{subject to } & \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\ & \min_{\mathbf{y} \in Y} f(\mathbf{y}) \\ \text{subject to } & g(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \end{aligned} \tag{6.9}$$

with $Y = \{0, 1\}^m$, $F(\mathbf{x}, \mathbf{y}) = \mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y}$, $f(\mathbf{y}) = \mathbf{d}_2 \mathbf{y}$, $\mathbf{G}(\mathbf{x}, \mathbf{y}) = \mathbf{A}_1 \mathbf{x} + \mathbf{B}_1 \mathbf{y} - \mathbf{b}_1$, and

$$g(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \mathbf{A}_2 \mathbf{x} + \mathbf{B}_2 \mathbf{y} - \mathbf{b}_2 \\ \mathbf{y} - e_y \\ -\mathbf{y} \end{bmatrix}$$

Denote the optimal solution of this problem by $(\mathbf{x}^*, \mathbf{y}^*)$. As before, let $\theta : R^m \rightarrow R^1$ be a continuous function satisfying $\theta(\mathbf{y}) \geq 0$, $\mathbf{y} \in R^m$ and $\theta(\mathbf{y}) = 0$ if and only if

$\mathbf{y} \in Y$, and let $M > 0$. As such, consider the BLPP

$$\begin{aligned} & \min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}) \\ \text{subject to } & \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\ & \min_{\mathbf{y}} f(\mathbf{y}) + M\theta(\mathbf{y}) \\ \text{subject to } & \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \end{aligned} \tag{6.10}$$

for the sequence $\{M_k\}$ of values of M satisfying $M_{k+1} > M_k \geq 0$ ($k = 1, 2, \dots$), and $\lim_k M_k = +\infty$. Let $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$ be the optimal solution of (6.10) and $\mathbf{y}(\mathbf{x}^{(k)})$ be the optimal solution of the lower-level problem of CDL-BLPP (6.9) when \mathbf{x} is equal to $\mathbf{x}^{(k)}$. The definitions of the vectors $\mathbf{x}^{(k)}$, $\mathbf{y}^{(k)}$ and $\mathbf{y}(\mathbf{x}^{(k)})$, and the theory of penalty function methods (see Section 4.4.2) imply:

Lemma 6.2.2 For any positive integer k ,

$$f(\mathbf{y}^{(k)}) + M_k\theta(\mathbf{y}^{(k)}) \leq f(\mathbf{y}(\mathbf{x}^{(k)})).$$

This leads to the limiting results stated above. For the proof, see [V3].

Theorem 6.2.5 If θ is a concave function, then any limit point $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ of the sequence $\{(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})\}$ is a solution of CDL-BLPP (6.9).

6.2.2 Algorithmic Implications

Theorem 6.2.3 shows that there exists a positive constant M such that the optimal solution of DCL-BLPP can be obtained by solving a penalized linear BLPP. As was the case with the penalty function approach for the pure linear BLPP discussed in Section 5.3.5, however, the value of M is not known in advance. In practical terms, this means that it is necessary to solve a series of linear BLPPs for a sequence of M_k values ($k = 0, 1, \dots$), starting with, say,

$$M_0 = (c_1 \mathbf{x}^0 + d_1 \mathbf{y}^0) / \theta(\mathbf{x}^0)$$

where $(\mathbf{x}^0, \mathbf{y}^0)$ is an initial point. As in Theorem 6.2.2, $(\mathbf{x}^0, \mathbf{y}^0)$ can be taken as the optimum of the relaxed version of DCL-BLPP. The sequence terminates when a value \bar{M} has been found such that the optimal solution of L-BLPP(\bar{M}) (or QL-BLPP(M)) is integral with respect to the upper-level variables. The corresponding optimal solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ of this linear (or quadratic-linear) BLPP is an optimal solution of DCL-BLPP.

If L-BLPP(M) is used, a linear bilevel program must be solved at each iteration. Alternatively, if QL-BLPP(M) is used it is necessary to solve a concave-linear bilevel program at each iteration, which represents a significant increase in work. In such circumstances, it might be advisable to employ a heuristic that finds verifiably high

quality feasible solutions and forego the guarantee of global optimality. The algorithm presented in [V2] could be used in this regard to find a local solution to each concave-linear bilevel subproblem.

The use of the linear three-level formulation presented in Theorem 6.2.4 for solving DL-BLPP, and by implication CDL-BLPP when coupled with Theorem 6.2.5, seems to be much more involved than solving the DCL-BLPP using the linear bilevel program equivalent introduced in Theorem 6.2.3 (for a branch-and-bound approach to DCL-BLPP, see [W2]). This is due to the fact that no efficient algorithms exist for the three-level problem. Related theory is limited and the few existing computational procedures are little more than ad hoc extensions of bilevel approaches (e.g., see [B4, W1]). Opportunities remain plentiful for research in this area.

6.3 PROPERTIES OF THE MIXED-INTEGER LINEAR BLPP

Algorithms designed to solve integer programs generally rely on some form of separation, relaxation, and fathoming to construct ever tighter bounds on the solution. As discussed in Chapter 3, separation is usually accomplished by placing contradictory constraints on a single integer variable. This approach is directly applicable to the mixed-integer BLPP. The natural relaxation derives from the removal of the integrality requirements on the variables. Fathoming, however, presents several difficulties. In mixed-integer programming (MIP), candidate subproblems are created by separation. Relaxation follows and some technique is employed to determine if the relaxed subproblem contains the optimal solution. Accordingly, if the subproblem does not contain a feasible solution better than the incumbent (best feasible solution yet found), the subproblem can be dismissed from further consideration. This leads to three general fathoming rules:

Rule 1 The relaxed subproblem has no feasible solution.

Rule 2 The solution of the relaxed subproblem is no less than the value of the incumbent.

Rule 3 The solution of the relaxed subproblem is feasible to the original problem.

Unfortunately, only rule 1 in its original form still holds for the mixed-integer BLPP. Rule 2 requires strong qualification, and rule 3 must be discarded altogether. The following examples demonstrate these points.

Example 6.3.1 For scalars x and y , consider

$$\begin{aligned}
 & \min_x F(x, y) = -x - 10y \\
 & \min_y f(y) = y \\
 & \text{subject to } -25x + 20y \leq 30 \\
 & \quad x + 2y \leq 10 \\
 & \quad 2x - y \leq 15 \\
 & \quad 2x + 10y \geq 15 \\
 & \quad x, y \geq 0 \text{ integer}
 \end{aligned}$$

The BLPP constraint region S for this example is shown in Fig. 6.2. When the integrality requirements are relaxed the solution is $(x, y) = (8, 1)$ with $F(x, y) = -18$ (note that this point is in the inducible region). The true optimum, though, obtained by enforcing integrality is $(x^*, y^*) = (2, 2)$ with $F(x^*, y^*) = -22$. As a consequence, we have:

Observation 1 The solution of the relaxed BLPP does not provide a valid bound on the solution of the mixed-integer BLPP.

Observation 2 Solutions to the relaxed BLPP that are in the inducible region cannot, in general, be fathomed.

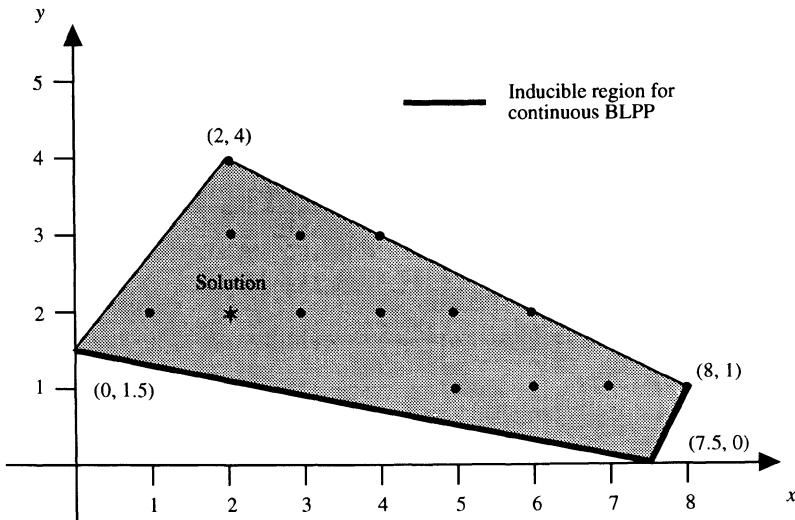


Figure 6.2 Constraint region for Example 6.3.1

The next example illustrates how a branch and bound approach to the mixed-integer BLPP can be thwarted if rule 3 is applied.

Example 6.3.2

$$\begin{aligned}
 \min_x \quad & F(x, y) = x + 2y \\
 \min_y \quad & f(y) = -y \\
 \text{subject to} \quad & -x + 2.5y \leq 3.75 \\
 & x + 2.5y \geq 3.75 \\
 & 2.5x + y \leq 8.75 \\
 & x, y \geq 0 \text{ integer}
 \end{aligned}$$

The BLPP constraint region for this example contains three integer points: (2,1), (2,2) and (3,1). If the leader picks $x = 2$, the follower chooses $y = 2$, so $F = 6$. If the leader's decision is $x = 3$, the follower's choice is $y = 1$, so $F = 5$. Therefore, the optimal solution of the problem is $(x^*, y^*) = (3, 1)$ with $F = 5$.

In a typical depth-first branch and bound scheme, separation is done by forming subproblems with contradictory constraints on a single integer variable, and relaxation is accomplished by ignoring the integrality requirements. Thus each subproblem is a mixed-integer BLPP with tighter bounds placed on the integer variables. One of many search trees that could have arisen for this example is shown in Fig. 6.3. At node 0, the relaxed solution is $(x, y) = (0, 1.5)$ with $F = 3$. Separating on the y variable and adding the constraint $y \geq 2$ yields the subproblem at node 1. Solving the corresponding relaxation yields $(x, y) = (1.25, 2)$ with $F = 5.25$. Further separation, fathoming due to infeasibility, and backtracking, eventually puts us at node 9 whose solution is $(x, y) = (2, 1)$ with $F = 4$. This point, although integer and in both S and the inducible region of the subproblem, is not bilevel feasible, and hence cannot be fathomed. The variable x must be further constrained in order to uncover the optimal solution. In addition, note that the objective function value obtained at node 9 is not a valid bound; and that at node 7, if y had been selected as the branching variable, the optimal solution would not have been found. Thus we see:

Observation 3 All integer solutions to the relaxed BLPP with some of the follower's variables restricted cannot, in general, be fathomed.

6.4 MOORE-BARD ALGORITHM FOR THE MIXED-INTEGER LINEAR BLPP

In this section, the ideas underlying the Kuhn-Tucker approach to the linear BLPP discussed in Section 5.3.2 are combined with the basic ideas of branch and bound described in Section 3.2. As in general integer programming, the practical limits on the size of problems that are solvable is a function of the number of integer variables in model (6.1a)–(6.1f). Consequently, several heuristics have been proposed to cut down on the size of the search tree. Computational results for several different procedures are presented at the end of the section.

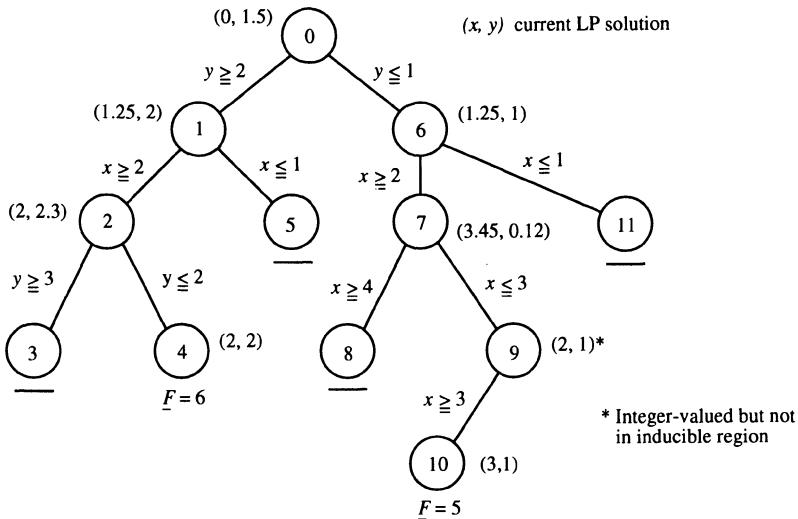


Figure 6.3 Search tree for Example 6.3.2

6.4.1 Branch and Bound Notation

Before formally stating the conditions under which the second type of fathoming is applicable, some additional notation must be introduced. Let

$N = \{1, \dots, n+m\}$ be the index set of decision variables;

$N^1 = \{1, \dots, n_2\}$ be the index set of the integer variables x_2 controlled by the leader;

$N^2 = \{1, \dots, m_2\}$ be the index set of integer variables y_2 controlled by the follower;

$u^1 = n_2$ -dimensional vector of original upper bounds on the integer variables controlled by the leader;

$u^2 = m_2$ -dimensional vector of original upper bounds on the integer variables controlled by the follower.

If an integer variable is unbounded above, then the corresponding entry in the upper bound vector is ∞ . The initial lower bound on each integer variable is assumed to be zero. For subproblem k , the sets of bounds on the variables are

$$H_k^1 = \{(\alpha^{1k}_j, \beta^{1k}_j) : 0 \leq \alpha^{1k}_j \leq x_{2j} \leq \beta^{1k}_j \leq u_j^1, j \in N^1\} \quad (6.11a)$$

$$H_k^2 = \{(\alpha^{2k}_j, \beta^{2k}_j) : 0 \leq \alpha^{2k}_j \leq y_{2j} \leq \beta^{2k}_j \leq u_j^2, j \in N^2\} \quad (6.11b)$$

where α^{1k} and β^{1k} are n_2 -dimensional vectors of lower and upper bounds, respectively, placed on the integer variables controlled by the leader, and α^{2k} and β^{2k} are m_2 -dimensional vectors of lower and upper bounds, respectively, placed on the integer variables controlled by the follower. For subproblem k , the notation $H_k^2(0, \infty)$ is used to indicate that no bounds other than the original bounds specified in problem (6.1) are placed on the integer variables controlled by the follower. In addition, by $H_\ell^1 \subset H_k^1$ we mean $\alpha_j^{1k} \leq \alpha_j^{1\ell}$ and $\beta_j^{1k} \geq \beta_j^{1\ell}$. Thus if node k is along the path to node ℓ , the subproblem associated with node ℓ is derived from the subproblem associated with node k , implying that $H_\ell^1 \subset H_k^1$ and $H_\ell^2 \subset H_k^2$.

The index sets of the integer variables that are restricted in subproblem k are

$$S_k^1 = \{j : \alpha_j^{1k} > 0 \text{ or } \beta_j^{1k} < u_j^1, j \in N^1\}$$

$$S_k^2 = \{j : \alpha_j^{2k} > 0 \text{ or } \beta_j^{2k} < u_j^2, j \in N^2\}$$

Accordingly, a restricted variable is one which has additional lower or upper bounds placed on it other than those included in the original formulation (6.1a)–(6.1f).

Now define F_k^C as the optimal solution of the relaxed BLPP for subproblem k ; F_k^C is found by dropping the integrality requirements and solving the resultant BLPP. The actual subproblem consists of (6.1) augmented by the bound constraints (6.11a,b), but without the requirement that \mathbf{x}_2 and \mathbf{y}_2 be integer. The *high point* solution to this subproblem is found by solving the linear program that results when the follower's objective function (6.1d) is removed from the formulation. Denote this value by F_k^H for subproblem k .

6.4.2 Bounding Theorems

Sufficient conditions are derived below that indicate when the solution of a relaxed subproblem may be used as an upper bound for the mixed-integer BLPP. As a consequence, subproblems whose solutions satisfy rule 2 above may be fathomed.

Theorem 6.4.1 Given H_k^1 and $H_k^2(0, \infty)$, let $(\mathbf{x}^k, \mathbf{y}^k)$ be the high point solution to the corresponding relaxed BLPP. Then $F_k^H = F(\mathbf{x}^k, \mathbf{y}^k)$ is a lower bound on the solution of the mixed-integer BLPP at node k .

Proof: Let $(\mathbf{y}^{*\ell}, \mathbf{y}^{*\ell})$ solve the mixed-integer BLPP at node ℓ where $H_\ell^1 \subset H_k^1$ and $H_\ell^2 \subset H_k^2(0, \infty)$. Assume $F(\mathbf{x}^{*\ell}, \mathbf{y}^{*\ell}) < F_k^H$. But this leads to a contradiction because $(\mathbf{x}^{*\ell}, \mathbf{y}^{*\ell})$ is a feasible solution of the high point problem at node k . ■

This result says that the high point solution at node k may be used as a bound to determine if the subproblem can be fathomed. This bound is only applicable when no restrictions on the integer variables controlled by the follower have been made along

the path to node k (i.e., $S_k^2 = \emptyset$). In other words, once the leader has made a decision, the follower is free to optimize his objective function without regard to any a priori or “artificial” restrictions.

The following theorem indicates when F_k^H provides a valid lower bound for the case where $S_k^2 \neq \emptyset$.

Theorem 6.4.2 Given H_k^1 and H_k^2 , let $(\mathbf{x}^k, \mathbf{y}^k)$ be the high point solution to the corresponding relaxed BLPP. Then $F_k^H = F(\mathbf{x}^k, \mathbf{y}^k)$ is a lower bound on the mixed-integer BLPP defined by the current path in the tree if none of the y_{2j}^k are at either $\alpha_j^{2k} > 0$ or $\beta_j^{2k} < u_j^2$ for $j \in S_k^2$.

Proof: See [M15]. ■

For Theorem 6.4.2 to be applicable none of the restricted integer variables controlled by the follower may be at their bound in the high point solution. This is a fairly strong condition that may not arise frequently enough to furnish good bounds. The following corollary offers some improvement.

Corollary 6.4.1 Given H_k^1 and H_k^2 , let $(\mathbf{x}^k, \mathbf{y}^k)$ be the high point solution of the corresponding relaxed BLPP with the restrictions in H_k^2 relaxed. Then $F_k^H = F(\mathbf{x}^k, \mathbf{y}^k)$ is a lower bound on the mixed-integer BLPP defined by the current path in the tree.

Proof: Relaxing the restrictions in H_k^2 is equivalent to replacing H_k^2 with $H_k^2(0, \infty)$. Thus Theorem 6.4.1 may be invoked. ■

Unfortunately, it doesn’t appear that any stronger bounds are available. In the BLPP, once the leader has made his decision, the follower is free to respond without regard to any a priori restrictions encountered by the leader in the branch and bound tree. This contrasts sharply with the standard mixed-integer program where all such bounds are valid.

6.4.3 Algorithm

The algorithm presented for solving (6.1a)–(6.1f) takes a depth-first branch-and-bound approach, incorporating the modifications discussed above. In particular, fathoming is only done when the relaxed BLPP is infeasible or when the lower bound at a node, as determined by the high point, is greater than or equal to the value of the incumbent, denoted by \bar{F} . The necessary bookkeeping is facilitated by explicit reference to the sets H_k^1 , H_k^2 , S_k^1 and S_k^2 .

Step 0 (Initialization) Put $k \leftarrow 0$. Set the parameters in H_k^1 and H_k^2 to the bounds of the mixed-integer BLPP. Set $S_k^1 = \emptyset$, $S_k^2 = \emptyset$, $\bar{F} = \infty$.

- Step 1 (Lower bounds and fathoming) Attempt to find the high point solution of the relaxed version of (6.1) and (6.11a,b) with no integrality requirements, and calculate F_k^H . If infeasible or $F_k^H \geq \bar{F}$, go to Step 6.
- Step 2 (Continuous solution) Attempt to solve the relaxed BLPP. If infeasible, go to Step 6. If successful, label the solution $(\mathbf{x}^k, \mathbf{y}^k)$ and the objective function value F_k^C .
- Step 3 (Branching) If the integrality requirements are satisfied by $(\mathbf{x}^k, \mathbf{y}^k)$, go to Step 4. Otherwise, select an x_{2j}^k , $j \in N^1$, or y_{2j}^k , $j \in N^2$, which is fractional-valued. Place a new bound on the selected variable. Put $k \leftarrow k + 1$ and update H_k^1 , H_k^2 , S_k^1 and S_k^2 . Go to Step 1.
- Step 4 (Bilevel feasible solution) Fix \mathbf{x} at \mathbf{x}^k and solve the follower's problem to obtain $(\mathbf{x}^k, \hat{\mathbf{y}}^k)$. Compute $F(\mathbf{x}^k, \hat{\mathbf{y}}^k)$ and put $\bar{F} = \min\{\bar{F}, F(\mathbf{x}^k, \hat{\mathbf{y}}^k)\}$.
- Step 5 (Integer branching) If $\alpha_j^{1k} = \beta_j^{1k}$ for each $j \in N^1$, and $\alpha_j^{2k} = \beta_j^{2k}$ for each $j \in N^2$, go to Step 6. Otherwise, select an integer variable such that $\alpha_j^{1k} \neq \beta_j^{1k}$, $j \in N^1$, or $\alpha_j^{2k} \neq \beta_j^{2k}$, $j \in N^2$, and place a new bound on it. Put $k \leftarrow k + 1$ and update H_k^1 , H_k^2 , S_k^1 and S_k^2 . Go to Step 1.
- Step 6 (Backtracking) If no live node exists, go to Step 7. Otherwise, branch to the newest live node, put $k \leftarrow k + 1$ and update H_k^1 , H_k^2 , S_k^1 and S_k^2 . Go to Step 1.
- Step 7 (Termination) If $\bar{F} = \infty$, there is no feasible solution to (6.1). Otherwise, terminate with the point in the inducible region associated with \bar{F} .

At Step 0, parameter values in H_k^1 and H_k^2 are initialized and the index sets, S_k^1 and S_k^2 , used in the search tree are set to \emptyset . At Step 1, the relaxation of problem (6.1) augmented by (6.11a,b) is solved to find the high point solution and a potential lower bound. The relaxed BLPP is solved at Step 2 to advance the search for bilevel feasibility.

Branching occurs at Step 3 where many rules for selecting the branching variable are possible. The rule that performed best chooses the integer variable with the largest fractional part and branches in the direction that is most likely to decrease the leader's objective function. For example, if x_{2j}^k , $j \in N^1$, is chosen at iteration k , and $c_{12,j} < 0$, then the restriction $x_{2j}^k \geq \lfloor x_{2j}^k \rfloor + 1$ is imposed on subproblem $k + 1$ (where $\lfloor \cdot \rfloor$ denotes the largest integer less than or equal to the argument). If no integer variable has a fractional value, the algorithm proceeds to Step 4 where a point in the inducible region is found. This is accomplished by fixing the variables controlled by the leader at their values arising at Step 2 and solving the MIP (6.1d,e,f). Call $(\mathbf{x}^k, \hat{\mathbf{y}}^k)$ the solution. If $F(\mathbf{x}^k, \hat{\mathbf{y}}^k) < \bar{F}$, the incumbent is updated. Next, the algorithm goes to Step 5 where an integer variable is selected for branching. Recall from Examples 6.3.1 and 6.3.2

that it is not possible to fathom an all integer solution. A list processing scheme is used to make the branching decision. If all the integer variables are fixed in the search tree, the algorithm goes to Step 6 and backtracks. Termination occurs at Step 7. If $\bar{F} \neq \infty$ the optimality of the incumbent must be addressed. The following conditions are sufficient to guarantee that the true solution has been found.

Proposition 6.4.1 If all the variables controlled by the leader are discrete, the algorithm finds the optimal solution to the mixed-integer linear BLPP (6.1).

Proof: Steps 3 and 5 assure that all possible combinations of \mathbf{x} are implicitly considered; the subproblem solved at Step 4 assures that all incumbent solutions are in the inducible region. ■

Proposition 6.4.2 Assume an optimum exists for problem (6.1) and that all the variables controlled by the follower are continuous. Then, if fathoming rules 2 and 3 are applied, the algorithm always terminates with the optimum.

Proof: In this case, $N^2 = \emptyset$ so the bounds found at Step 2 are always valid. That is, the continuous BLPP solution, F_k^C , is always less than or equal to the true optimum at node k because the latter is always feasible to the former, and no a priori restrictions are ever placed on the follower's variables. An additional implication is that a solution satisfying integrality can be fathomed. Thus rules 2 and 3 can now be applied. Arguments similar to those made in the proof of Proposition 6.4.1 assure convergence. ■

Proposition 6.4.1 states that when at least one of the variables controlled by the leader is continuous the algorithm may not produce an optimal solution. Fortunately, this situation can only occur when the optimum is not well defined. Moore and Bard [M15] give an example where the rational reaction set for a mixed-integer linear BLPP is discontinuous and hence not closed. The above algorithm cannot recognize this situation and iterates until the normal termination conditions are met, stopping with an incorrect point as the optimum. Aside for the cases identified by Propositions 6.4.1 and 6.4.2, it remains an open question whether convergence can be proven for the general model.

Bounding and Fathoming Heuristics

For problems with more than a few integer variables, the computational burden may become excessive if good bounds aren't forthcoming. But even then, optimality may never be confirmed as Example 6.3.2 demonstrates. (The procedure always terminates with a point in the inducible region, though.) As a consequence, a series of heuristics designed to make the algorithm less enumerative has been investigated. Each represents a tradeoff between accuracy and computational effort.

To begin, we consider using the solution of the relaxed BLPP obtained at Step 2 as an alternative lower bound. This can be incorporated in the algorithm by replacing Step 2 with:

- Step 2a Attempt to solve the relaxed BLPP. If infeasible, go to Step 6. Alternatively, label the solution $(\mathbf{x}^k, \mathbf{y}^k)$ and compute $F(\mathbf{x}^k, \mathbf{y}^k)$. If $F(\mathbf{x}^k, \mathbf{y}^k) \geq \bar{F}$, go to Step 6; otherwise go to Step 3.

Next, we introduce three new fathoming rules based on integrality. Each is meant as a replacement for Step 4.

- Step 4a Fix \mathbf{x} at \mathbf{x}^k and solve the follower's problem to obtain $(\mathbf{x}^k, \hat{\mathbf{y}}^k)$. Compute $F(\mathbf{x}^k, \hat{\mathbf{y}}^k)$. If $F(\mathbf{x}^k, \hat{\mathbf{y}}^k) < \bar{F}$, put $\bar{F} \leftarrow F(\mathbf{x}^k, \hat{\mathbf{y}}^k)$ and go to Step 5; otherwise go to Step 6.

- Step 4b Fix \mathbf{x} at \mathbf{x}^k and solve the follower's problem to obtain $(\mathbf{x}^k, \hat{\mathbf{y}}^k)$. Compute $F(\mathbf{x}^k, \hat{\mathbf{y}}^k)$ and put $\bar{F} = \min\{\bar{F}, F(\mathbf{x}^k, \hat{\mathbf{y}}^k)\}$. If $\hat{\mathbf{y}}^k = \mathbf{y}^k$ go to Step 6; otherwise go to Step 5.

- Step 4c Fix \mathbf{x} at \mathbf{x}^k and solve the follower's problem to obtain $(\mathbf{x}^k, \hat{\mathbf{y}}^k)$. Compute $F(\mathbf{x}^k, \hat{\mathbf{y}}^k)$ and put $\bar{F} = \min\{\bar{F}, F(\mathbf{x}^k, \hat{\mathbf{y}}^k)\}$. Go to Step 6.

Step 4a allows the algorithm to backtrack when the point in the inducible region derived from a solution at the current node has a value greater than or equal to that of the incumbent. Step 4b sends the algorithm to Step 6 when the solution at the current node satisfies the integrality requirements and is in the inducible region. Step 4c permits backtracking simply when the integrality requirements are satisfied. Table 6.1 lists the seven additional algorithms that can be formed by using various combinations of the above steps. Each terminates with a feasible, but not necessarily optimal, solution to (6.1).

Table 6.1 Variants of basic algorithm

No.	Description
1	Basic algorithm
2	Algorithm 1 with Step 4 replaced by Step 4a
3	Algorithm 1 with Step 4 replaced by Step 4b
4	Algorithm 1 with Step 4 replaced by Step 4c
5	Algorithm 1 with Step 2 replaced by Step 2a
6	Algorithm 5 with Step 4 replaced by Step 4a
7	Algorithm 5 with Step 4 replaced by Step 4b
8	Algorithm 5 with Step 4 replaced by Step 4c

6.4.4 Computational Experience

To judge the comparative performance of the eight algorithms, Moore and Bard randomly generated and solved 50 test problems. Their experimental design included 10 classes of problems, each distinguished by the number of integer and continuous variables controlled by the players. In all cases, the coefficients of the \mathbf{A} and \mathbf{B} matrices took on values between -15 and 45 with approximately 25% of the entries being less than zero. Each had a density of about 40%. The coefficients of the two objective functions varied between -20 and 20 with approximately 50% being less than zero. The number of constraints in each problem was set at 0.4 times the total number of variables, and the right-hand-side values ranged between 0 and 50 (see Table 6.2). The signs of the constraints had a 0.7 probability of being \leq and a 0.3 probability of being \geq .

All computations were performed on an IBM 3081-D using the VS FORTRAN compiler. This machine is roughly equivalent in speed to a Sun Sparcstation 10. The continuous BLPPs arising at Step 4 were solved with the Bard-Moore code described in Section 5.3.2 after making minor modifications to account for the complementarity conditions associated with the follower's integer variables in the search tree.

Results

Five separate runs were made for each class of problems listed in Table 6.2. The results are displayed in Table 6.3 for the average CPU time (in seconds) required by each algorithm to reach termination. As might have been expected, Algorithms 4 and 8, which fathom integer solutions without regard to feasibility, performed best, while Algorithm 1 proved to be an order of magnitude slower. It should be noted that a limit of 995 nodes was placed on all problems to guard against excessive run times. The need for this restriction became apparent after a form of cycling was observed at Step 5. In particular, Algorithm 1 had a tendency to get into a loop where one variable would be continually selected for branching until all of its possible integer values were explored.

When the 995 node limit was reached the incumbent was presented as the solution. Algorithm 1 reached this limit in eight of the 50 problems examined. Algorithm 3, however, encountered this condition only three times, while Algorithms 5 and 7 terminated prematurely only twice. Algorithms 2, 4, 6 and 8 always ran to completion. Looping was not evident in any of the problems arising in classes 1, 7, 9 and 10.

Table 6.4 displays the average number of nodes needed by each algorithm to reach termination, and Table 6.5 gives the average number of nodes required to find the best feasible solution. As can be seen, the results in the three aforementioned tables are highly correlated. A question remains, though, as to the quality of the final solutions.

Table 6.2 Number of variables in test problems

Class	No. of variables ($n + m$)†	No. of follower variables (m)	No. of integer leader variables (n_2)	No. of integer follower variables (m_2)
1	15	5	2	3
2	15	5	3	3
3	20	10	5	5
4	25	10	5	5
5	30	15	5	5
6	30	15	7	8
7	35	15	5	5
8	35	15	8	9
9	40	20	5	5
10	40	20	10	10

†Number of constraints $q = 0.4(n + m)$.

Table 6.3 Average CPU time (seconds) for algorithms†

Problem Class	Algorithm							
	1	2	3	4	5	6	7	8
1	5	1	3	1	3	1	3	1
2	210	1	1	1	1	1	1	1
3	186	27	105	17	49	12	35	9
4	73	5	5	4	6	5	5	4
5	678	77	837	70	683	57	730	50
6	244	46	113	41	99	36	97	32
7	170	42	33	28	41	38	28	24
8	634	178	422	175	89	72	84	66
9	366	103	235	81	118	78	107	59
10	730	272	454	311	123	123	118	118

†Each problem class comprises 5 instances

On 11 of the 50 problems, the algorithms did not reach the same feasible solution. The bottom row of Table 6.5 indicates the number of times each algorithm failed to arrive at the presumed optimum. On four of the 11 disagreements, Algorithm 1 did not provide the best feasible solution. In each case, the algorithm terminated because it exceeded the node limit. Algorithms 3, 5 and 7 each terminated once at the node limit without reaching a feasible solution as good as that found by one of

Table 6.4 Average number of nodes in search tree

Problem Class	Algorithm							
	1	2	3	4	5	6	7	8
1	86	21	56	18	51	21	49	18
2	250	12	10	8	14	12	10	8
3	541	60	313	54	169	32	166	26
4	243	11	11	8	12	10	10	7
5	483	49	415	42	343	26	362	20
6	380	40	239	35	225	27	223	24
7	90	15	15	11	13	11	9	6
8	366	115	249	117	51	40	47	37
9	136	30	84	25	41	20	38	15
10	128	42	70	40	30	30	27	27

the other algorithms. On three of the problems that the algorithms did not reach the same solution, node limit violation was the reason. For the fourth problem, the best feasible solution was found by Algorithm 3 alone. On this particular problem, Algorithm 3 terminated because it had reached the node limit. Algorithms 1, 5 and 7 terminated at the node limit with solutions less than the one found by Algorithm 3. The other algorithms terminated normally without achieving the same solution as Algorithm 3.

Table 6.5 Average number of nodes to find best feasible solution

Problem Class	Algorithm							
	1	2	3	4	5	6	7	8
1	60	19	56	18	49	19	48	18
2	44	10	10	8	12	10	10	8
3	212	33	201	28	148	27	146	23
4	10	7	6	6	7	7	6	6
5	78	26	60	22	62	20	57	16
6	216	23	216	21	209	22	208	21
7	90	15	15	11	13	11	9	6
8	59	8	8	5	10	8	8	5
9	107	26	74	23	38	17	37	14
10	36	20	27	20	20	20	20	20
Failures	4	5	6	8	4	5	6	8

For the other seven disagreements, Algorithm 1 alone found the best feasible solution three times. Of the four remaining cases, Algorithms 1, 2, 5 and 6 found the best solution twice, Algorithms 1, 3, 5 and 7 found the best solution once, and Algorithms 1, 2, 3, 5, 6 and 7 found the best solution once.

From an examination of the individual runs, it was seen that CPU time varied greatly with both problem size and specific realization. This point is illustrated by the results obtained for the class 5 problems. For half the algorithms, the average CPU time exceeded that for all other classes. In addition, fluctuations of up to 300 seconds within the class were observed for Algorithms 1 and 5.

6.4.5 Assessment

An approximate ranking of the algorithms by CPU time from fastest to slowest yields 8, 6, 4, 2, 7, 5, 3 and 1. As expected, reduced accuracy often accompanied an increase in speed. Algorithms 4 and 8, for example, failed to reach the best available solution in eight of the 50 instances. In particular, for three problems in class 3 a average discrepancy of 662% was observed; for problems in classes 5, 6 and 7 the results were significantly better but discrepancies were still present. Recall that Algorithms 4 and 8 backtrack when a point satisfying the integrality requirements is uncovered. Similarly, Algorithms 2 and 6 failed to reach the best feasible solution 10 percent of the time. They backtrack when no improvement is provided by a solution that satisfies the integrality requirements. Algorithms 3 and 7 failed to find the best feasible solution on six of the problems, while Algorithm 5 failed on four. Algorithms 3 and 7 backtrack when the solution of (6.1) augmented by the bound constraints (6.11a,b) is in the inducible region, and Algorithm 5 backtracks when the solution of the continuous version of (6.1) augmented by the bounds (6.11a,b) is no better than the incumbent.

In summary, if it is desirable to find the best feasible solution regardless of CPU time, Algorithm 1 without a node limit is the proper choice. However, if a tradeoff can be made between the quality of the solution and CPU time, then Algorithm 5 offers a good compromise. If CPU time is of primary importance, Algorithm 8 is recommended. Nevertheless, the best feasible solutions uncovered by each of the algorithms invariably arise well within 995 nodes, and in many instances, before the 50th subproblem is solved.

6.5 ALGORITHM FOR THE DISCRETE LINEAR BLPP

This section highlights the Bard-Moore [B13] algorithm developed for the pure integer linear version of the BLPP with binary restrictions on the variables. The problem is:

$$\min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}) = \mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y} \quad (6.12a)$$

$$\text{subject to } \mathbf{A}_1 \mathbf{x} + \mathbf{B}_1 \mathbf{y} \leq \mathbf{b}_1 \quad (6.12b)$$

$$\mathbf{x} \in X = \{0, 1\}^n \quad (6.12c)$$

$$\min_{\mathbf{y}} f(\mathbf{y}) = \mathbf{d}_2 \mathbf{y} \quad (6.12d)$$

$$\text{subject to } \mathbf{A}_2 \mathbf{x} + \mathbf{B}_2 \mathbf{y} \leq \mathbf{b}_2 \quad (6.12e)$$

$$\mathbf{y} \in Y = \{0, 1\}^m \quad (6.12f)$$

where $\mathbf{c}_1 \in R^n$, $\mathbf{d}_1, \mathbf{d}_2 \in R^m$, $\mathbf{b}_1 \in R^p$, $\mathbf{b}_2 \in R^q$, $\mathbf{A}_1 \in R^{p \times n}$, $\mathbf{B}_1 \in R^{p \times m}$, $\mathbf{A}_2 \in R^{q \times n}$, $\mathbf{B}_2 \in R^{q \times m}$. The elements of these arrays are assumed to be integer constants. Once \mathbf{x} is chosen, the follower's problem (6.12d)–(6.12f) becomes a 0-1 integer linear program in \mathbf{y} only.

The foundation for the algorithm is the implicit enumeration scheme of Balas described in Section 3.2.5. However, the proposed methodology can be easily modified to solve (6.12a)–(6.12f) for the more general case where the \mathbf{y} variables are unrestricted integers. The present discussion and implementation, though, are limited to the binary case.

The key to the algorithm is in the recognition that any solution to the 0-1 integer linear BLPP must have \mathbf{y} lying in the follower's rational reaction set $P(\mathbf{x})$. By limiting the search to this set while constantly seeking improvement in the leader's objective function it is possible to quickly uncover good points in the inducible region. This is achieved by formulating and repeatedly solving instances of the following parameterized integer program (cf. [R3]) derived from (6.12a)–(6.12f).

$$\min_{\mathbf{y}} f(\mathbf{y}) = \mathbf{d}_2 \mathbf{y} \quad (6.13a)$$

$$\text{subject to } \mathbf{A}_1 \mathbf{x} + \mathbf{B}_1 \mathbf{y} \leq \mathbf{b}_1 \quad (6.13b)$$

$$\mathbf{A}_2 \mathbf{x} + \mathbf{B}_2 \mathbf{y} \leq \mathbf{b}_2 \quad (6.13c)$$

$$F(\mathbf{x}, \mathbf{y}) = \mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y} \leq \alpha \quad (6.13d)$$

$$\sum_{j=1}^n x_j \geq \beta \quad (6.13e)$$

$$\mathbf{x} \in X, \mathbf{y} \in Y \quad (6.13f)$$

where α and β are right-hand-side parameters initially taken as ∞ and 0, respectively. Constraint (6.13d) forces a tradeoff between the two objective functions, and is identical to the cut used by Bialas and Karwan [B24] in their complementary pivoting heuristic for the continuous BLPP. (In contrast to Bialas and Karwan who require $d_1 \leq 0$, no restrictions are placed on any of the problem coefficients here.) Lastly, constraint (6.13e) restricts the sum of the variables controlled by the leader and although not strictly necessary, has proven to be an effective way of selecting branching variables during enumeration.

6.5.1 Algorithm

A depth-first implicit enumeration scheme centering on the leader's decision variables and applied to problem (6.13a)–(6.13f) is used to solve the 0-1 integer linear BLPP. The basic idea is to examine points that satisfy the constraints (6.13b)–(6.13f) in a systematic order, fix the x variables at their corresponding values, and then re-solve (6.13a)–(6.13f) to obtain a new point in the inducible region. By adjusting α and β at each iteration, the algorithm steadily decreases the leader's objective function until the problem becomes infeasible.

The notation introduced in Section 5.3.2 will be used to provide a structure for enumerating the x variables. Let $W = \{1, \dots, n\}$ and define at the k th iteration of the algorithm a path vector P_k of length $\ell = |W_k|$ (for $W_k \subseteq W$) corresponding to an assignment of either $x_j = 0$ or $x_j = 1$ for $j \in W_k$. The vector P_k identifies a partial solution for the variables controlled by the leader at the ℓ th level of the search tree. Here, $\ell = |W_k|$ is the length of the path from the root of the tree to the current node. The index set of assigned variables is denoted by W_k . The path vector indicates the order in which the variables in W_k have been assigned, as well as their binary state. For example, if $x_3 = 0$ and $x_2 = 1$ have been fixed (in that order) at iteration k , then $W_k = \{2, 3\}$, $\ell = 2$ and $P_k = (\underline{3}, 2)$.

Now let

$$\begin{aligned} S_k^+ &= \{j : j \in W_k \text{ and } x_j = 1\} \\ S_k^- &= \{j : j \in W_k \text{ and } x_j = 0\} \\ S_k^0 &= \{j : j \notin W_k\} \end{aligned}$$

A completion of W_k is an assignment of binary values to the free variables controlled by the leader. The latter constitute the index set S_k^0 . The algorithm never explicitly places restrictions on the follower's variables.

After k iterations, let IR^k be the set of points in the inducible region so far uncovered, and denote by \bar{F} the incumbent upper bound associated with the leader's objective

function. That is, $\bar{F} = \min\{F(\mathbf{x}, \mathbf{y}) : (\mathbf{x}, \mathbf{y}) \in IR^k\}$. At the start of the algorithm, all variables are free and $\bar{F} = \infty$.

Step 0 (Initialization) Set $k = 0$, $S_k^+ = \emptyset$, $S_k^- = \emptyset$, $S_k^0 = \{1, \dots, n\}$, $\alpha = \infty$, $\beta = 0$ and $\bar{F} = \infty$. This creates the root of the search tree.

Step 1 (General iteration) Set $x_j = 1$ for $j \in S_k^+$ and $x_j = 0$ for $j \in S_k^-$. For the current values of α and β , attempt to find a feasible solution to (6.13a)–(6.13f). If successful, put $k \leftarrow k + 1$, label the solution $(\mathbf{x}^k, \mathbf{y}^k)$ and go to Step 2; otherwise fathom the current node and go to Step 6.

Step 2 (Bounding) Fix \mathbf{x} at \mathbf{x}^k relax (6.13d) and (6.13e), and solve (6.13a,b,c,f) to get a point $(\mathbf{x}^k, \hat{\mathbf{y}}^k) \in IR$. Compute $F(\mathbf{x}^k, \hat{\mathbf{y}}^k)$ and update \bar{F} by putting $\bar{F} = \min\{\bar{F}, F(\mathbf{x}^k, \hat{\mathbf{y}}^k)\}$.

Step 3 Let $J = \{j \in S_{k-1}^0 : x_j^k = 1\}$. If $J = \emptyset$, set $S_k^+ = S_{k-1}^+$, $S_k^- = S_{k-1}^-$, $S_k^0 = S_{k-1}^0$, $P_k = P_{k-1}$ and go to Step 5; otherwise go to Step 4.

Step 4 (Branching) Create $|J|$ new nodes as follows: append $j \in J$ to P_{k-1} in ascending order, one by one, to obtain the new live nodes and the path P_k ; set $S_k^+ = S_{k-1}^+ \cup J$, $S_k^0 = S_{k-1}^0 \setminus J$ and $S_k^- = S_{k-1}^-$. The new path P_k extends the old path by length $|J|$ and leads from the root to the new current node.

Step 5 Let $\alpha = \bar{F} - 1$ and $\beta = 1 + |S_k^+|$. Go to Step 1.

Step 6 (Backtracking) If no live nodes exist, go to Step 7. Otherwise backtrack from the current node (this is the most recently created live node; denote the corresponding variable index by j'), branch on its complement by setting $x_{j'}^k = 0$, and update S_k^+ , S_k^- , S_k^0 and P_k as discussed below. Set $\beta = 0$ and go to Step 1.

Step 7 (Termination) If $\bar{F} = \infty$, there is no feasible solution to (6.12a)–(6.12f). Otherwise, declare the feasible point associated with \bar{F} an optimal solution.

Step 1 is designed to find a new point which is potentially bilevel feasible. The counter k is only incremented if the search is successful. Test results indicated that solving (6.13a)–(6.13f) to optimality at this step offers little advantage, primarily because the solution (call it $(\mathbf{x}^k, \mathbf{y}^k)$) is not necessarily in the inducible region. That is, another point might exist that satisfies (6.13b,c,e,f), but violates the cut (6.13d) and provides the follower with a smaller objective function value than $f(\mathbf{y}^k)$.

At Step 2, \mathbf{x} is fixed at \mathbf{x}^k , constraints (6.13d) and (6.13e) are relaxed, and the resultant subproblem is solved to get a point in the inducible region. If an improvement is realized, the incumbent is replaced. The set J in Step 3 identifies the branching

variables. If J is empty, control passes to Step 5 where β is set to one plus the number of elements in S_k^+ . The algorithm then returns to Step 1. Any feasible solution found at this point will be accompanied by a $J \neq \emptyset$.

Branching occurs at Step 4 where S_k^+ , S_k^0 and P_k are updated. In the process, all free \mathbf{x} variables equal to one in the solution at Step 1 are set to one, and a corresponding number of new nodes is created in the search tree. Contrary to the usual depth-first procedure where one new node is created at each iteration, it was found advantageous to extend the tree by several levels at a time.

At Step 5, α is set to the current best known value \bar{F} of (6.12a)–(6.12f) and decreased by one. This guarantees that when the algorithm returns to Step 1, if a feasible solution of (6.13a)–(6.13f) is found, say $(\mathbf{x}^k, \mathbf{y}^k)$, we will have $F(\mathbf{x}^k, \mathbf{y}^k) < \bar{F}$. Also at Step 5, β is set to the sum of the variables controlled by the leader and is incremented by one. This ensures that at least one element of the set S_k^0 at iteration k has a value of one, thus providing a branching variable at Step 4 in the next pass through Step 3. If constraint (6.13e) were not included, the same \mathbf{x} values might result when (6.13a)–(6.13f) is subsequently solved at Step 1. Should this occur, the algorithm is said to have stalled because no further progress is possible without introducing additional branching rules.

If problem (6.13a)–(6.13f) is infeasible at Step 1, the algorithm proceeds to Step 6 where backtracking takes place. Note that a live node is one associated with a subproblem defined by a combination of elements in S_k^+ and S_k^- that has yet to be fully explored or fathomed at Step 1 due to infeasibility. To facilitate bookkeeping, the path P_k in the search tree is represented by an ℓ -dimensional vector, where ℓ is the current depth of the tree. The order of the components is determined by their level. Indices only appear in the vector P_k if they are in S_k^+ or S_k^- , and appear underlined if they are in S_k^- . In the branching operation at Step 4, if more than one variable is selected, they are appended to the path vector in ascending order. Note that each P_k defines a unique node in the tree. Because the algorithm always branches to the left first ($x_j = 1$), backtracking is accomplished by finding the rightmost nonunderlined element of P_k , underlining it, and erasing all entries to its right. Though not explicitly stated in Step 6, a new node is added to the search tree each time the backtracking operation is performed. The newly underlined entry is deleted from S_k^+ and added to S_k^- ; the erased entries are deleted from S_k^- and added to S_k^0 . This leads to the following result.

Proposition 6.5.1 Let X^* be the set of optimal solutions for the leader, assuming an optimum exists. If $P(\mathbf{x})$ is single-valued for all $\mathbf{x} \in X^*$, or if $f(\mathbf{y})$ is unique for $(\mathbf{x}, \mathbf{y}) \in \{\mathbf{x} \in X^*, \mathbf{y} \in P(\mathbf{x})\}$, then the algorithm terminates with an optimal solution to the BLPP (6.12a)–(6.12f). Alternatively, a sufficient condition for the algorithm to terminate with an optimal solution is that the follower always selects a $\mathbf{y} \in P(\mathbf{x})$ that minimizes $F(\mathbf{x}, \mathbf{y})$ for \mathbf{x} fixed.

Proof: The algorithm implicitly looks at all combinations of \mathbf{x} at Steps 4 and 6; the subproblem solved at Step 2 guarantees that all incumbent solutions are in the inducible region. ■

Remark: The conditions of Proposition 6.5.1 are not readily verifiable so unless the subproblem (5.3) is solved, it cannot be determined whether the algorithm terminates with the optimum. However, by using ideas from goal programming and including the term $-\varepsilon F(\mathbf{x}, \mathbf{y})$ (where $\varepsilon > 0$ is an arbitrarily small constant) in the objective function (6.13a), the algorithm will yield a solution to the more general but conservative problem

$$\min_{(\mathbf{x}, \mathbf{y}) \in IR} \max_{\mathbf{y} \in P(\mathbf{x})} F(\mathbf{x}, \mathbf{y})$$

thus obviating the need to check for multiple optima (this solution strategy is commonly used in dynamic games; see [B14]). To see this, replace $f(\mathbf{y})$ with $f(\mathbf{y}) - \varepsilon F(\mathbf{x}, \mathbf{y})$ in (6.13a) and let $P'(\mathbf{x})$ be the set of solutions associated with problem (6.13a,b,c,f) for \mathbf{x} fixed. Because ε is arbitrarily small and \mathbf{y} is discrete, it follows that $P'(\mathbf{x}) \subseteq P(\mathbf{x})$. Hence, $(\mathbf{x}, \mathbf{y}') \subseteq IR$, where $\mathbf{y}' \in P'(\mathbf{x})$. Thus each time (6.13a,b,c,f) is solved with \mathbf{x} fixed at Step 2 to get a point in IR , the solution, call it \mathbf{y}' , minimizes $f(\mathbf{y})$ while concurrently maximizing $F(\mathbf{x}, \mathbf{y})$ over all points in $P(\mathbf{x})$.

Alternative Algorithm

Each time a feasible point $(\mathbf{x}^k, \mathbf{y}^k)$ is uncovered at Step 1, a subproblem is solved at Step 2 to obtain a point $(\mathbf{x}^k, \hat{\mathbf{y}}^k)$ in the inducible region. Because that additional effort is wasted whenever $\mathbf{y}^k = \hat{\mathbf{y}}^k$, it would be helpful to know when this condition is met. If (6.13a)–(6.13f) is solved to optimality, the following result can be used to provide a partial check.

Proposition 6.5.2 Let $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ solve problem (6.13a)–(6.13f). Then $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in IR$ if

$$c_1 \bar{\mathbf{x}} + \sum_{j=1}^m (d_{1j} - \min\{d_{1j}, 0\}) \leq \alpha \quad (6.14)$$

Proof: In order for $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ not to be in the inducible region, $\bar{\mathbf{y}} \in Y$ must fail to minimize $f(\mathbf{y})$ subject to $\mathbf{B}_2 \mathbf{y} \leq \mathbf{b}_2 - \mathbf{A}_2 \bar{\mathbf{x}}$. Thus there would exist a $\hat{\mathbf{y}} \in Y$ such that $\mathbf{B}_2 \hat{\mathbf{y}} \leq \mathbf{b}_2 - \mathbf{A}_2 \bar{\mathbf{x}}$, $\mathbf{B}_2 \hat{\mathbf{y}} \leq \mathbf{b}_2 - \mathbf{A}_2 \bar{\mathbf{x}}$ and $f(\hat{\mathbf{y}}) < f(\bar{\mathbf{y}})$. Since $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ solves problem (6.13a)–(6.13f), it follows that $c_1 \bar{\mathbf{x}} + \mathbf{d}_1 \bar{\mathbf{y}} > \alpha$. Now, by noting that the summation in (6.14) is equal to $\max_{\mathbf{y} \in Y} \{d_{1j}\}$, which of course must be greater than or equal to $d_1 \hat{\mathbf{y}}$, we see that it is impossible for $c_1 \bar{\mathbf{x}} + d_1 \hat{\mathbf{y}} > \alpha$. This contradiction proves the result. ■

In light of Proposition 6.5.2, an alternative computational scheme can be developed by modifying the first two steps of the algorithm in the following manner.

Step 1a (General iteration) Set $x_j = 1$ for $j \in S_k^+$ and $x_j = 0$ for $j \in S_k^-$. For the current values of α and β attempt to find an optimal solution to

(6.13a)–(6.13f). If successful, put $k \leftarrow k + 1$, label the solution $(\mathbf{x}^k, \mathbf{y}^k)$ and go to Step 2a; otherwise go to Step 6.

Step 2a (Bounding) Use Proposition 6.5.2 to try to verify that $(\mathbf{x}^k, \mathbf{y}^k)$ is in the inducible region. If so, go to Step 2b; otherwise fix \mathbf{x} at \mathbf{x}^k , relax (6.13d) and (6.13e), and solve (6.13a)–(6.13f) to get a point $(\mathbf{x}^k, \hat{\mathbf{y}}^k) \in IR$.

Step 2b Compute $F(\mathbf{x}^k, \hat{\mathbf{y}}^k)$ and put $\bar{F} = \min\{\bar{F}, F(\mathbf{x}^k, \hat{\mathbf{y}}^k)\}$.

This modification represents a tradeoff between the effort required at Step 1 to find a feasible solution to (6.13a)–(6.13f) versus the effort required at Step 1a to find an optimal solution to (6.13a)–(6.13f)) with the prospect of not having to solve the integer program at Step 2a. That is, at iteration k with $|S_k^0|$ free \mathbf{x} variables, the original algorithm first seeks a feasible solution to a zero-one program with $|S_k^0| + m$ variables and then (if successful) an optimal solution to a zero-one program with only m variables. The modified procedure immediately seeks an optimal solution to a 0-1 program with $|S_k^0| + m$ variables.

After some preliminary testing on problems with 20 leader variables, 20 follower variables and 16 constraints, Bard and Moore found that the original algorithm was anywhere from 2 to 10 times faster than the alternative. Additional investigation showed that the only time the latter is more efficient is when m is much greater than n . In this case, the subproblems at Step 2 are relatively burdensome so some advantage is gained if they don't have to be solved at each iteration.

Finally, it should be mentioned that there is nothing in either approach that limits the \mathbf{y} variables to be binary, or their corresponding functions to be linear. Integrality is required, though, if the α -cut (6.13d) is to be effective. The only constraining factor is the software used at Steps 1 and 2 to solve the respective subproblems. In addition, by substituting a branching rule for the β -cut (6.13e), the algorithm can be further generalized to include the case where the \mathbf{x} 's are unrestricted integer variables.

Example 6.5.1 A simple example for $\mathbf{x} \in R^1$ and $\mathbf{y} \in R^1$ is used to demonstrate how the algorithm works. It also illustrates two interesting characteristics of the solution.

$$\begin{aligned} \min_{x \geq 0} \quad & F(x, y) = x + y \\ \min_{y \geq 0} \quad & f(y) = -5x - y \\ \text{subject to} \quad & -x - y/2 \leq -2 \\ & -x/4 + y \leq 2 \\ & x + y/2 \leq 8 \\ & x - 2y \leq 4 \end{aligned}$$

The optimal solution of this problem without regard to integrality is $(x^*, y^*) = (8/9, 20/9)$, with $F = 28/9$. A graph of the feasible region, S , indicates that $x < 8$

and $y < 4$. Alternatively, the third constraint of the inner problem shows that $x \leq 8$ (with $x = 8$ requiring $y = 0$, which the fourth constraint forbids), while four times the second constraint plus the third constraint gives $4.5y \leq 16$, or $y < 4$. Therefore, if x and y are now restricted to integer values, a zero-one formulation can be obtained by letting $x = x_1 + 2x_2 + 4x_3$ and $y = y_1 + 2y_2$. This leads to

$$\begin{aligned} \min_{\mathbf{x} \in X} F(\mathbf{x}, \mathbf{y}) &= x_1 + 2x_2 + 4x_3 + y_1 + 2y_2 \\ \min_{\mathbf{y} \in Y} f(\mathbf{x}, \mathbf{y}) &= -5x_1 - 10x_2 - 20x_3 - y_1 - 2y_2 \\ \text{subject to } &-2x_1 - 4x_2 - 8x_3 - y_1 - 2y_2 \leq -4 \\ &-x_1 - 2x_2 - 4x_3 + 4y_1 + 8y_2 \leq 8 \\ &2x_1 + 4x_2 + 8x_3 + y_1 + 2y_2 \leq 16 \\ &x_1 + 2x_2 + 4x_3 - 2y_1 - 4y_2 \leq 4 \end{aligned}$$

The parameterized subproblem (6.13a)–(6.13f) is then

$$\begin{aligned} \min_{\mathbf{y} \in Y} f(\mathbf{y}) &= -y_1 - 2y_2 \\ \text{subject to } &-2x_1 - 4x_2 - 8x_3 - y_1 - 2y_2 \leq -4 \\ &-x_1 - 2x_2 - 4x_3 + 4y_1 + 8y_2 \leq 8 \\ &2x_1 + 4x_2 + 8x_3 + y_1 + 2y_2 \leq 16 \\ &x_1 + 2x_2 + 4x_3 - 2y_1 - 4y_2 \leq 4 \\ F(\mathbf{x}, \mathbf{y}) = &x_1 + 2x_2 + 4x_3 + y_1 + 2y_2 \leq \alpha \\ &x_1 + x_2 + x_3 \geq \beta \end{aligned}$$

The first few steps of the algorithm are outlined below. For conciseness, the integer representation of the binary points is used in the discussion. Note that the intermediate results are not unique but depend upon the algorithmic approach used to solve the zero-one subproblems at Steps 1 and 2.

After initialization at Step 0, the algorithm finds at Step 1 the point $(4,3)$, which is confirmed to be in the inducible region at Step 2. Thus, $\bar{F} = 7$, $S_1^+ = \{3\}$, $S_1^- = \emptyset$, $S_1^0 = \{1, 2\}$, $P_1 = (3)$, $\alpha = 6$ and $\beta = 2$. Solving problem (6.13a)–(6.13f) again at Step 1 yields the point $(5,1)$. At Step 2, $(\mathbf{x}^2, \hat{\mathbf{y}}^2) = (5, 3)$ and $F(\mathbf{x}^2, \hat{\mathbf{y}}^2) = 8$ so \bar{F} remains at 7. Updating gives $S_2^+ = \{1, 3\}$, $S_2^0 = \{2\}$, $P_2 = (3, 1)$, $\alpha = 6$ and $\beta = 3$. At Step 1 no feasible solution is found, so the algorithm fathoms the current node, goes to Step 6 and backtracks giving $S_2^+ = \{3\}$, $S_2^- = \{1\}$, $S_2^0 = \{2\}$, $P_2 = (3, 1)$, $\alpha = 6$ and $\beta = 0$. Returning to Step 1, the feasible point $(\mathbf{x}^3, \hat{\mathbf{y}}^3) = (4, 0)$ is found, and Step 2 again yields $(\mathbf{x}^3, \hat{\mathbf{y}}^3) = (4, 3)$ with $F(\mathbf{x}^3, \hat{\mathbf{y}}^3) = 7$. At Step 3, the fact that $S_2^0 = \{2\}$ and $x_2^3 = 0$ implies that $J = \emptyset$, and hence, $S_3^+ = S_2^+ = \{3\}$. The

parameter β is therefore set to $1 + |S_3^+| = 2$ at Step 5, and control passes to Step 1. The resultant subproblem is infeasible so the algorithm fathoms the current node and jumps to Step 6 and backtracks.

The calculations continue until convergence occurs. The optimal solution for the example is $(x^*, y^*) = (1, 2)$, with $F^* = 3$. The corresponding search tree is shown in Fig. 6.4, where the values of F beside the nodes are those found at Step 2. During execution, the algorithm returns to Step 1 thirteen times in an attempt to improve upon the incumbent. Subsequently, seven subproblems are solved at Step 2 to find new points in the inducible region. Once again, these results reflect the peculiarities of the specific zero-one code used. Finally, if the alternative algorithm is used and problem (6.13a)–(6.13f) is completely solved at Step 1, three of these seven subproblems are eliminated at Step 2. Here, the choice of zero-one code is immaterial.

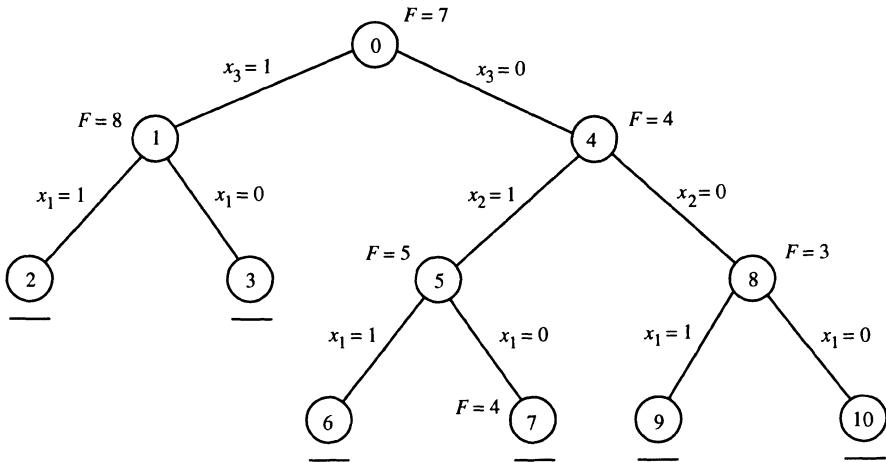


Figure 6.4 Search tree for Example 6.5.1

With regard to individual outcomes, note that the leader does better when the variables are restricted to integer, rather than continuous, values. This was similarly true in Example 6.5.1. Although this observation cannot be generalized, it arises here because the follower's interests are opposed to those of the leader in the sense that d_1 and d_2 have opposite signs, but the follower's feasible region $S(\mathbf{x})$ (and hence his freedom to advance those interests) is somewhat reduced by the integer requirements.

A second inference that can be drawn from the example concerns the nature of the solution. In particular, note that when compared to the optimum $(1, 2)$, the point $(2, 0)$ provides better outcomes for both players but is not in the inducible region. That is, if the leader plays $x = 2$, the follower chooses $y = 2$, thereby increasing

the leader's payoff by 1 unit giving $F = 4$. This verifies, for the integer case, the general fact that solutions to a BLPP are not necessarily Pareto-optimal. (If the first term in $f(x, y)$ is dropped, a slight alteration in the problem would be required to demonstrate this result.)

6.5.2 Computational Experience

The algorithm was tested on a series of randomly generated problems. In each instance, the constraints had a 0.5 probability of being \leq or \geq inequalities, and all right-hand-side values were positive. The coefficients of the \mathbf{A} and \mathbf{B} matrices ranged from -10 to $+28$ with approximately 40% of the nonzero entries being negative. The coefficients of the leader's and follower's objective functions similarly assumed both positive and negative values with the same probability. The number of constraints in each realization was set to 0.4 times the number of variables. Because problem difficulty often depends upon the density of the constraints, two separate cases were examined. The first was characterized by densities of approximately 0.475 for the \mathbf{A} and \mathbf{B} matrices, and the second by densities of approximately 0.245.

All computations were performed on an IBM 3081-D using the VS FORTRAN compiler. A Balas-type algorithm (see Section 3.2.5) was used to solve the 0-1 programs at Steps 1 and 2. In all, 10 problems were run for each case (a case is defined by the way the total number of variables is partitioned amongst the players). Performance measures included CPU time (seconds), the coefficient of variation (standard deviation/mean) for CPU time, the number of nodes in the search tree, the node at which the optimal solution was found, and the number of problems examined at Steps 1 and 2.

Table 6.6 presents the average results for the case where the matrix densities are 0.475. As expected, the CPU time grows exponentially with the size of the problem but seems to be independent of the way the variables are partitioned. Nevertheless, the averages given in the table do not reveal the large differences that were observed among problems of equivalent size. To some extent, this can be seen by examining the coefficient of variation – a measure of the dispersion of CPU time. As an example, the greatest range in CPU time appeared for those problems containing 45 variables where 15 are controlled by the follower. Here, the longest CPU time is 757 seconds while the shortest is less than 3 seconds; the standard deviation is 1.47 times the mean. Eliminating these two extremes and redoing the calculations gives an average of 89.6 seconds for the remaining eight problems with a standard deviation to mean ratio of 0.92.

In general, coefficient of variation values near 1 indicate the presence of outliers. This is illustrated by considering the following sample data set $\{1,1,1,9\}$. The mean for these four points is 3 and the coefficient of variation is 1.15. This type of pattern was typical of the CPU times accompanying the results. Each problem set contained

Table 6.6 Computational results for matrix densities of 0.475

No. of variables ($n + m$)	Followers variables (m)	CPU time (sec)	Coefficient of variation	No. of iterations		No. of nodes	Optimal solution (node)
				Step 1	Step 2		
15	5	0.06	0.71	26	14	23	13
20	5	0.10	0.58	24	11	25	15
20	20	0.37	1.03	47	29	34	18
25	5	0.51	0.77	39	18	39	26
25	10	1.87	0.79	141	80	119	50
30	5	0.98	0.89	47	19	53	44
30	10	1.23	0.95	44	21	46	24
30	15	3.11	1.21	47	25	43	26
35	5	5.10	1.36	36	14	42	31
35	10	4.97	1.30	49	23	50	41
35	15	13.49	1.15	86	49	71	54
40	5	14.06	1.25	47	15	61	47
40	10	31.87	2.10	52	22	58	45
40	15	18.58	1.22	47	21	51	31
40	20	21.78	1.01	44	21	44	28
45	10	23.08	1.08	20	8	22	16
45	15	147.66	1.47	50	20	7	48
45	20	106.86	0.83	124	64	117	106

one, and sometimes two, instances in which the CPU time was at least one order of magnitude greater than the majority. Similar observations were made concerning the number of times it was necessary to execute Steps 1 and 2. For problems with 50 or more variables (results not shown in Table 6.6), computation times often exceeded 900 seconds, the upper limit chosen for these runs. When the A and B matrix densities were set at approximately 0.245, both CPU time and the accompanying variation within a problem set were slightly greater than for the case where the matrix densities were 0.475. However, the order of magnitude of these two measures was the same. The places where strong differences arose were in the number of times Steps 1 and 2 were executed, the number of nodes in the search tree, and the node at which the optimal solution was found. These results are tabulated in [B13].

6.5.3 Assessment

After solving approximately 400 randomly generated problems, Bard and Moore concluded that a wide variation in algorithmic performance existed, even for problems of

the same class. And while the number of variables strongly affected overall computation time, little could be predicted in advance for a specific instance.

These observations, coupled with the fact that in most cases the algorithm quickly found good points in the inducible region, suggest a variety of heuristics for reducing the computational burden. These include: (1) stopping after a given number of nodes, (2) stopping after a given number of passes through Steps 1 or Step 2, or (3) stopping after a given amount of CPU time has elapsed. Justification follows from what appears to be a high correlation between the first and the last four columns in Table 6.6. A third possibility aimed at speeding convergence centers on the choice of branching rules. One alternative that was tried involved ordering the variables according to their coefficient values in the leader's objective function. On average, this produced a small degradation in performance. Likewise, branching on the variables one at a time increased the computational effort anywhere from 50 to 1000%.

If any real improvement in the basic algorithm is to be made, though, Step 1 holds the key. Step 1 yields an \mathbf{x}^k and a corresponding (i) $\mathbf{y}^k \in S(\mathbf{x}^k)$ for which $F(\mathbf{x}^k, \mathbf{y}^k) \leq \alpha$. What we would really like is an \mathbf{x}^k and (ii) a $\mathbf{y}^k \in P(\mathbf{x}^k)$ for which $F(\mathbf{x}^k, \mathbf{y}^k) \leq \alpha$. Although (i) is necessary for (ii), it provides only weak assurance, in general, that (ii) might hold. The modified algorithm offers a means of checking for (ii) but the underlying conditions proved to be too weak to be effective. Nevertheless, it might still be possible to sharpen Step 1 to increase the likelihood that \mathbf{y}^k is in $P(\mathbf{x}^k)$ without having to solve (6.13a)–(6.13f) to optimality. A plausible approach is to seek a test for $(\mathbf{x}^k, \mathbf{y}^k) \in IR$ based on the degree of harmony or opposition of the two players as measured by the relative sizes of \mathbf{d}_1 and \mathbf{d}_2 , or by the sign of the scalar product $\langle \mathbf{d}_1, \mathbf{d}_2 \rangle$.

CONVEX BILEVEL PROGRAMMING

7.1 INTRODUCTION

Of the algorithms presented in Chapter 5 for finding global minima of linear bilevel programs, the Kuhn-Tucker approach [B11], the variable elimination method [H1], and the complementarity approach [J4] are the most efficient and robust developed to date. These algorithms can be readily extended to solve the linear-quadratic case where the functions F , \mathbf{G} and \mathbf{g} are linear, and the function f is strictly convex quadratic.

Bilevel programs with linear constraints, nonlinear upper-level objective functions and strictly convex quadratic lower-level objectives are much more difficult to solve. To date, even for specific nonlinear instances of F , only a few enumerative procedures have been proposed to find a global minimum (e.g., see [A5, B6, E1]). Penalty function approaches ([A2, A3, I2]) and other descent algorithms [K6]) have also been developed but only guarantee local optimality.

In this chapter, we consider the convex version of the BLPP given by

$$\begin{aligned} & \min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}) \\ \text{subject to } & \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq 0 \\ & \min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \\ \text{subject to } & \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq 0 \end{aligned} \tag{7.1}$$

where $F, f : R^n \times R^m \rightarrow R^1$, $\mathbf{G} : R^n \times R^m \rightarrow R^p$, and $\mathbf{g} : R^n \times R^m \rightarrow R^q$ are continuous, twice differentiable convex functions. When convenient, it will be assumed that any nonnegativity restrictions on the variables are subsumed in the functions $\mathbf{G}(\mathbf{x}, \mathbf{y})$ and $\mathbf{g}(\mathbf{x}, \mathbf{y})$. In general, we will restrict ourselves to the case where $f(\mathbf{x}, \mathbf{y})$ is strictly convex in \mathbf{y} for \mathbf{x} fixed. This assures, along with a constraint qualification, that the solution to the follower's problem in (7.1) is unique, implying that the rational reaction set $P(\mathbf{x})$ is single-valued and that the inducible region, IR , can be replaced

by a unique response function, say, $\mathbf{y} = \Psi(\mathbf{x})$. Consequently, it can be shown that IR is continuous (see Proposition 8.1.2 and related discussion). This fact is exploited in the algorithm presented in Section 7.3.

If we now assume that a constraint qualification exists for the follower's problem for each $\mathbf{y} \in P(\mathbf{x})$; i.e., for each point in the rational reaction set, then Proposition 5.2.2 can be modified for the convex case. Accordingly, problem (7.1) can be written as the following single-level mathematical program

$$\min_{\mathbf{x}, \mathbf{y}, \mathbf{u}} F(\mathbf{x}, \mathbf{y}) \quad (7.2a)$$

$$\text{subject to } G(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \quad (7.2b)$$

$$\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) + \mathbf{u} \nabla_{\mathbf{y}} g(\mathbf{x}, \mathbf{y}) = \mathbf{0} \quad (7.2c)$$

$$\mathbf{u} g(\mathbf{x}, \mathbf{y}) = 0 \quad (7.2d)$$

$$g(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \quad (7.2e)$$

$$\mathbf{u} \geq \mathbf{0} \quad (7.2f)$$

where $\mathbf{u} \in R^q$ is a (row) vector of Kuhn-Tucker multipliers associated with the follower's subproblem for \mathbf{x} fixed.

The nonconvex nature of this formulation can be seen in the stationarity condition (7.2c) and in the complementarity condition (7.2d). The latter can be handled by implicit enumeration as proposed by Bard and Moore [B11]. The former may be highly nonlinear and hence a lot more troublesome. The following results from [B6] are for the case where f is quadratic.

Proposition 7.1.1 If $f(\mathbf{x}, \mathbf{y})$ is quadratic in (\mathbf{x}, \mathbf{y}) and the constraint region S is polyhedral then the inducible region is piecewise linear.

Proof: As \mathbf{x} is varied the solutions to the follower's problem in (7.1) either occur on a face of dimension $\leq q - 1$ of S or in its interior. In the latter case, we have $\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) = \mathbf{0}$ from (7.2c). The result then follows from Proposition 8.1.2. ■

Proposition 7.1.2 Let $F(\mathbf{x}, \mathbf{y})$ be strictly convex in (\mathbf{x}, \mathbf{y}) , $f(\mathbf{x}, \mathbf{y})$ be quadratic in (\mathbf{x}, \mathbf{y}) , and S be polyhedral. If $\mathbf{z}^1 = (\mathbf{x}^1, \mathbf{y}^1, \mathbf{u}^1)$ and $\mathbf{z}^2 = (\mathbf{x}^2, \mathbf{y}^2, \mathbf{u}^2)$ are distinct local solutions to problem (7.2a)–(7.2f) and both lie on the same face of S then that face cannot be in IR .

Proof: Let $\hat{\mathbf{z}} = (\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{u}}) = \alpha \mathbf{z}^1 + (1 - \alpha) \mathbf{z}^2$, $\alpha \in [0, 1]$ be a line on the face common to \mathbf{z}^1 and \mathbf{z}^2 and assume that $\hat{\mathbf{z}}$ satisfies constraints (7.2b)–(7.2f); i.e., $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in IR$. From strict convexity we have $F(\hat{\mathbf{z}}) < \alpha F(\mathbf{z}^1) + (1 - \alpha) F(\mathbf{z}^2)$ which, for α in the neighborhood of 0 or 1, contradicts the assumption that \mathbf{z}^1 and \mathbf{z}^2 are local optima. In particular, (7.2d) must be violated. ■

Example 7.1.1 Consider the convex BLPP for scalars x and y :

$$\min_{x \geq 0} F(x, y) = (x - 5)^2 + (2y + 1)^2$$

$$\text{subject to } \min_{y \geq 0} f(x, y) = (y - 1)^2 - 1.5xy$$

$$\text{subject to } -3x + y \leq -3$$

$$x - 0.5y \leq 4$$

$$x + y \leq 7$$

Figure 7.1 displays the BLPP constraint region S and the inducible region IR for Example 7.1.1. Notice that the latter is nonconvex, and unlike the case where all the functions are linear, does not lie wholly on the faces of S ; however, its piecewise linear nature can be observed. Nonconvexity foreshadows the existence of local solutions which are located at $(1, 0)$ and $(5, 2)$. Note that if these points are joined by adding the constraint $0.5x - y \leq 0.5$ to the example, Corollary 7.1.2 states that if they remain local optima (which they do) then the hyperplane $0.5x - y = 0.5$ cannot be in IR .

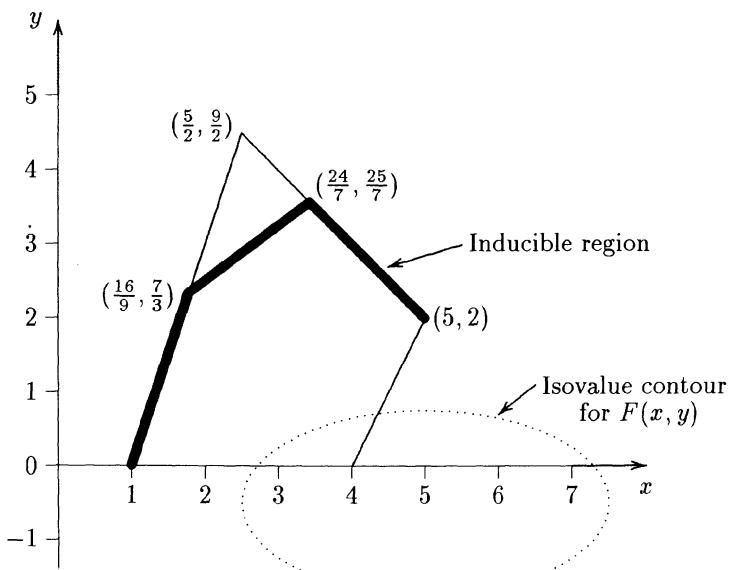


Figure 7.1 Geometry and inducible region for Example 7.1.1

In the remainder of this chapter we present a number of algorithms designed to find local and global solutions to several versions of the convex BLPP, the simplest being where F is strictly convex, G is convex, f is quadratic and g is affine. When f and g assume more complicated forms, problem (7.2a)–(7.2f) suggests that we will most likely have to be satisfied with local solutions.

7.2 DESCENT APPROACHES FOR THE QUADRATIC BLPP

In this section, two descent algorithms are presented for solving bilevel programs in which the upper-level objective function F is quadratic, the lower-level objective function f is strictly convex and quadratic, and the lower-level constraint set is polyhedral. The discussion closely follows that of Vicente, Savard and Júdice [V2]. The first algorithm tries to enforce direct movement along the inducible region by complementary pivoting in a simplex framework. The drawback of this strategy is that local optimality may not be achieved. When the upper-level objective function is concave, however, it is shown that the procedure always converges to a local minimum.

The second algorithm is a modification of the steepest descent approach of Savard and Guavín (see Section 8.5) which uses the sequential LCP method (see Section 5.3.3) to efficiently compute each steepest descent direction. This strategy is generally effective but may experience some difficulty when a local solution is at hand. (In fact, the authors prove that checking a point for (strict) local optimality in bilevel programming is NP-hard.) From an implementation point of view, new rules for computing exact stepsizes are introduced and a hybrid approach that combines both strategies is discussed.

Problem Definition and Properties

The quadratic bilevel programming problem (Q-BLPP) under investigation has the following form:

$$\begin{aligned} \min_{\mathbf{x} \geq 0} \quad & F(\mathbf{x}, \mathbf{y}) = \frac{1}{2} [\mathbf{x}^T, \mathbf{y}^T] \begin{bmatrix} \mathbf{C}_1 & \mathbf{C}_3 \\ \mathbf{C}_3^T & \mathbf{C}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} + c_1 \mathbf{x} + d_1 \mathbf{y} \\ \min_{\mathbf{y} \geq 0} \quad & f(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \mathbf{y}^T \mathbf{Q} \mathbf{y} + \mathbf{y}^T \mathbf{D} \mathbf{x} + d_2 \mathbf{y} \\ \text{subject to} \quad & \mathbf{A} \mathbf{x} + \mathbf{B} \mathbf{y} \leq \mathbf{b} \end{aligned}$$

where $\mathbf{c}_1 \in R^n$, $\mathbf{d}_1, \mathbf{d}_2 \in R^m$, $\mathbf{C}_1 \in R^{n \times n}$, $\mathbf{Q}, \mathbf{C}_2 \in R^{m \times m}$, $\mathbf{D}, \mathbf{C}_3^T \in R^{m \times n}$, $\mathbf{A} \in R^{q \times n}$, $\mathbf{B} \in R^{q \times m}$ and $\mathbf{b} \in R^q$.

We assume that $\mathbf{C} = \begin{bmatrix} \mathbf{C}_1 & \mathbf{C}_3 \\ \mathbf{C}_3^T & \mathbf{C}_2 \end{bmatrix}$ and \mathbf{Q} are respectively positive semidefinite and positive definite matrices. This implies that both the relaxed problem (RP) in the \mathbf{x} and \mathbf{y} variables

$$\begin{aligned} \min_{\mathbf{x} \geq 0, \mathbf{y} \geq 0} \quad & \frac{1}{2} [\mathbf{x}^T, \mathbf{y}^T] \begin{bmatrix} \mathbf{C}_1 & \mathbf{C}_3 \\ \mathbf{C}_3^T & \mathbf{C}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} + c_1 \mathbf{x} + d_1 \mathbf{y} \\ \text{subject to} \quad & \mathbf{A} \mathbf{x} + \mathbf{B} \mathbf{y} \leq \mathbf{b} \end{aligned}$$

and the lower-level problem (LLP(\mathbf{x})) in the \mathbf{y} variables

$$\begin{aligned} \min_{\mathbf{y} \geq 0} \quad & \frac{1}{2} \mathbf{y}^T \mathbf{Q} \mathbf{y} + \mathbf{y}^T \mathbf{D} \mathbf{x} + \mathbf{d}_2^T \mathbf{y} \\ \text{subject to} \quad & \mathbf{B} \mathbf{y} \leq \mathbf{b} - \mathbf{A} \mathbf{x} \end{aligned}$$

are convex quadratic programs. We also assume that the set $P(\mathbf{x}) = \{\mathbf{y} \in R^m : \mathbf{B} \mathbf{y} \leq \mathbf{b} - \mathbf{A} \mathbf{x}, \mathbf{y} \geq 0\}$ is nonempty for all values of \mathbf{x} . As a consequence, the lower-level problem LLP(\mathbf{x}) has a unique solution for each \mathbf{x} and Q-BLPP has a global optimum [E1]. The inducible region for Q-BLPP is the set $IR = \{(\mathbf{x}, \mathbf{y}) : \mathbf{x} \geq \mathbf{0}, \mathbf{y} \text{ is optimal for LLP}(\mathbf{x})\}$. Now, because LLP(\mathbf{x}) is a convex program in \mathbf{y} , the inducible region is defined by the following linear complementarity conditions:

$$\mathbf{Q} \mathbf{y} + \mathbf{D} \mathbf{x} + \mathbf{d}_2 + \mathbf{B}^T \boldsymbol{\gamma} - \boldsymbol{\beta} = \mathbf{0} \quad (7.3a)$$

$$\mathbf{A} \mathbf{x} + \mathbf{B} \mathbf{y} + \boldsymbol{\alpha} = \mathbf{b} \quad (7.3b)$$

$$\mathbf{x}, \mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma} \geq \mathbf{0} \quad (7.3c)$$

$$\boldsymbol{\alpha}^T \boldsymbol{\gamma} = \boldsymbol{\beta}^T \mathbf{y} = 0 \quad (7.3d)$$

where $\boldsymbol{\alpha}, \boldsymbol{\gamma} \in R^q$ and $\boldsymbol{\beta} \in R^m$. This result follows directly by replacing LLP(\mathbf{x}) with its Kuhn-Tucker conditions, which are sufficient by the convexity assumption.

Definition 7.2.1 The point $\mathbf{u} = (\mathbf{x}, \mathbf{y})$ is said to be in the extreme inducible region (*EIR*) if there exist $\boldsymbol{\alpha}, \boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ such that $(\mathbf{x}, \mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ is an extreme point of the polyhedral set defined by (7.3a)–(7.3c) and satisfies the complementarity conditions (7.3d). An *EIR* point is said to be nondegenerate if the values of the basic variables are all positive. Otherwise it is called degenerate.

From here on out, it is assumed that all points in *EIR* are nondegenerate.

As in linear programming, two points in *EIR* are said to be adjacent if their bases differ in exactly one column. It follows from Definition 7.2.1 that movement between two adjacent *EIR* points can be achieved with a pivot step that maintains complementarity (7.3d). All that is necessary to implement such a scheme is to prevent two complementary variables from being basic at the same time.

The next definition establishes a class of directions that plays an important role in the development of the complementary pivoting approach.

Definition 7.2.2 The vector $\mathbf{d} \in R^{n+m}$ is an extreme inducible region (*EIR*) direction if it lies along the edge connecting two adjacent points in *EIR*.

If an *EIR* point is not a local minimum of Q-BLPP, then there exists at least one descent *EIR* direction emanating from it. This result is stated in the next theorem and is used later to design a descent algorithm.

Theorem 7.2.1 Let \mathbf{u} be a point in EIR . If \mathbf{u} is not a local minimum of Q-BLPP then there is at least one descent EIR direction at \mathbf{u} .

Proof: The nonoptimality of \mathbf{u} implies the existence of at least one feasible descent direction in IR ; call it \mathbf{d} . From the piecewise linear property of the inducible region [B6], such a direction may be written as a convex combination of EIR directions, \mathbf{d}^i , $i = 1, \dots, \ell$:

$$\mathbf{d} = \sum_{i=1}^{\ell} \mu_i \mathbf{d}^i, \quad \text{where} \quad \sum_{i=1}^{\ell} \mu_i = 1, \quad \mu_i > 0 \quad (7.4)$$

If all directions \mathbf{d}^i , i, \dots, ℓ , satisfy

$$\nabla F(\mathbf{u})^T \mathbf{d}^i \geq 0$$

then

$$\sum_{i=1}^{\ell} \mu_i \nabla F(\mathbf{u})^T \mathbf{d}^i \geq 0$$

But from (7.4), this last condition implies that

$$\nabla F(\mathbf{u})^T \mathbf{d} \geq 0$$

which contradicts the assumption that \mathbf{d} is a descent direction. Therefore, at least one of the EIR directions $\mathbf{d}^1, \dots, \mathbf{d}^{\ell}$ is a descent direction. ■

7.2.1 An EIR Point Descent Algorithm

If $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is a nondegenerate EIR point then one of the following situations holds:

- (i) $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is a local minimum of Q-BLPP
- (ii) $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is not a local minimum and there exists an adjacent EIR point $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ and corresponding EIR direction satisfying:

$$F(\hat{\mathbf{x}}, \hat{\mathbf{y}}) < F(\bar{\mathbf{x}}, \bar{\mathbf{y}})$$

- (iii) $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is not a local minimum of Q-BLPP and

$$F(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \geq F(\bar{\mathbf{x}}, \bar{\mathbf{y}})$$

for all adjacent EIR points $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$. We call $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ a *local star inducible region (LSIR)* point.

It is important to note that if a $LSIR$ point $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is not a local minimum of Q-BLPP, then, by Theorem 7.2.1, there exists a descent EIR direction at $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$. This situation is illustrated in the following example.

Example 7.2.1 Consider the 3-variable Q-BLPP:

$$\min_{x_1, x_2} \frac{1}{2} \left(x_1 - \frac{4}{5} \right)^2 + \frac{1}{2} \left(x_2 - \frac{1}{5} \right)^2 + \frac{1}{2}(y - 1)^2$$

subject to $0 \leq x_1, x_2 \leq 1$

$$\min_y \frac{1}{2} y^2 + y - x_1 y + 2x_2 y$$

subject to $0 \leq y \leq 1$

The inducible region for this problem is the union of the following sets:

$$\{(x_1, x_2, y) \in R^3 : x_1 \leq 1, x_2 \geq 0, -x_1 + 2x_2 \leq 0, y = 1\}$$

$$\{(x_1, x_2, y) \in R^3 : -x_1 + 2x_2 + y = 1, 0 \leq y \leq 1\}$$

$$\{(x_1, x_2, y) \in R^3 : x_1 \geq 0, x_2 \leq 1, -x_1 + 2x_2 \geq 1, y = 0\}$$

The first set is depicted in Fig. 7.2 and consists of the triangle whose vertices are $V_1 = (1, 0, 1)$, $V_2 = (0, 0, 1)$ and $V_3 = (1, 1/2, 1)$. Although at the *EIR* point V_1 , $\vec{V}_1\vec{V}_2$ and $\vec{V}_1\vec{V}_3$ are descent *EIR* directions, the values for the upper-level objective function at the adjacent *EIR* points V_2 and V_3 are greater than the value at V_1 . Hence V_1 is an *LSIR* point.

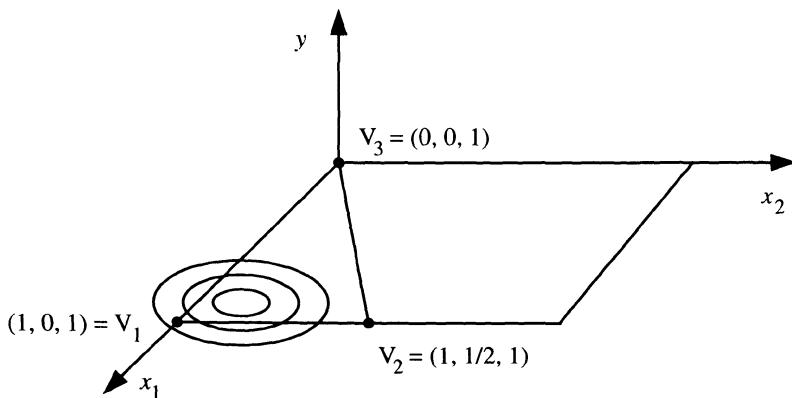


Figure 7.2 LSIR point with two descent EIR directions

It is easy to design an algorithm that finds at least an *LSIR* point for Q-BLPP. The first step is to find an initial point in *EIR*. At each iteration the current *EIR* point is either an *LSIR* point or a local minimum and the algorithm terminates, or an adjacent *EIR* point is obtained with a lower value of F . The procedure is then repeated. The steps of the algorithm are as follows:

Initial Step – Compute an initial *EIR* point \mathbf{u}^0 . Set $k = 0$.

General Step – Let \mathbf{D}_k be the set of descent *EIR* directions at \mathbf{u}^k :

$$\mathbf{D}_k = \{\mathbf{d} : \nabla F(\mathbf{u}^k)^T \mathbf{d} < 0 \text{ and } \mathbf{d} \text{ is an EIR direction}\}$$

(i) If $\mathbf{D}_k \neq \emptyset$, select $\mathbf{d}^k \in \mathbf{D}_k$ such that:

$$F(\mathbf{u}^{k+1}) < F(\mathbf{u}^k)$$

where \mathbf{u}^{k+1} is the *EIR* point adjacent to \mathbf{u}^k in the direction \mathbf{d}^k . Put $k \leftarrow k + 1$ and repeat this step. If such a direction does not exist Stop; \mathbf{u}^k is an *LSIR* point of Q-BLPP.

(ii) If $\mathbf{D}_k = \emptyset$, Stop; \mathbf{u}^k is a local minimum of Q-BLPP.

Each iteration of the algorithm consists of a pivot step that maintains complementarity, implying a low computational effort. Termination always occurs with an *LSIR* point \mathbf{u}^k , provided all the points in *EIR* are nondegenerate. Only the case where $\mathbf{D}_k = \emptyset$, however, assures that \mathbf{u}^k is a local minimum – a major drawback of the approach.

Another important issue is the computation of an initial point in *EIR*. Because the relaxed problem RP is a convex quadratic program and its constraint set is nonempty, an optimal solution $(\mathbf{x}_R, \mathbf{y}_R)$ exists and can be found in polynomial time. A point in *IR* can also be found in polynomial time by, say, setting $\mathbf{x} = \mathbf{x}_R$ and solving the lower-level quadratic program LLP(\mathbf{x}_R). Given that \mathbf{Q} is a symmetric positive definite matrix and $P(\mathbf{x}_R) \neq \emptyset$, this program has an unique solution, $\bar{\mathbf{y}}_R$, such that $(\mathbf{x}_R, \bar{\mathbf{y}}_R) \in \text{IR}$. Of course, if $(\mathbf{x}_R, \mathbf{y}_R)$ is in *IR* then it is a global minimum of Q-BLPP, but this is not normally the case.

Although an initial point in *IR* can be computed efficiently, such a point is not, in general, in *EIR* since it doesn't correspond to a basic solution of the system of linear constraints (7.3a)–(7.3c). Nevertheless, Murty [M18] provides a polynomial-time algorithm that can be used to generate a basic feasible solution. Because the number of positive variables is reduced at each iteration in that algorithm, only points in *IR* are visited so complementarity is satisfied. Hence an initial point in *EIR* can be found in polynomial time.

7.2.2 A Modified Steepest Descent Approach

The second approach to the Q-BLPP proposed by Vicente et al. [V2] makes use of the steepest descent algorithm introduced in [S2] and discussed in Section 8.5. A basic assumption in that work and imposed here is that the gradients of the active constraints at each point used by the algorithm are linearly independent.

At a given iteration k , the steepest descent direction $\mathbf{d}^k = (\mathbf{z}^k, \mathbf{w}^k)$ at an inducible region point $\mathbf{u}^k = (\mathbf{x}^k, \mathbf{y}^k)$ is found by solving the following linear-quadratic bilevel program denoted by LQ-BLPP $_k$:

$$\begin{aligned} & \min_{\mathbf{z}} (\mathbf{C}_1 \mathbf{x}^k + \mathbf{C}_3 \mathbf{y}^k + \mathbf{c}_1)^T \mathbf{z} + (\mathbf{C}_3^T \mathbf{x}^k + \mathbf{C}_2 \mathbf{y}^k + \mathbf{d}_1)^T \mathbf{w} \\ & \text{subject to } -1 \leq z_i \leq 1, \quad i = 1, \dots, n \\ & \min_{\mathbf{w} \geq 0} \mathbf{w}^T \mathbf{Q} \mathbf{w} + 2 \mathbf{w}^T \mathbf{D} \mathbf{z} \\ & \text{subject to } \mathbf{A}' \mathbf{z} + \mathbf{B}' \mathbf{w} \leq 0 \\ & \quad (-\phi^k \mathbf{A}')^T \mathbf{z} + (\mathbf{Q} \mathbf{y}^k + \mathbf{D} \mathbf{x}^k + \mathbf{d}_2)^T \mathbf{w} = 0 \end{aligned}$$

where $\mathbf{z} \in R^n$ and $\mathbf{w} \in R^m$. The matrices \mathbf{A}' and \mathbf{B}' contain all the rows of \mathbf{A} and \mathbf{B} corresponding to the active constraints at \mathbf{u}^k . Similarly, the vector \mathbf{w}' is a subvector of \mathbf{w} where only the indices i corresponding to zero variables (\mathbf{y}^k) are considered. Furthermore, ϕ^k is a (row) vector of multipliers associated to the active constraints at \mathbf{u}^k .

If the optimal value of LQ-BLPP $_k$ is greater than or equal to zero, then \mathbf{u}^k is a local minimum of Q-BLPP. Otherwise the optimal solution of LQ-BLPP $_k$ is the steepest descent direction $(\mathbf{z}^k, \mathbf{w}^k)$ and a new point in the inducible region is found from

$$(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}) = (\mathbf{x}^k, \mathbf{y}^k) + \sigma_k (\mathbf{z}^k, \mathbf{w}^k)$$

where σ_k is an appropriate stepsize.

If only a descent direction is required, there is no need to solve LQ-BLPP $_k$ until the final step of the algorithm. All values between zero and the (negative) optimal value of LQ-BLPP $_k$ correspond to descent directions [S2]. When trying to solve LQ-BLPP $_k$, it is important to take advantage of this property. We now show how the sequential LCP method discussed in Section 5.3.3 can be adopted for this purpose. We also describe how an exact stepsize can be computed in an efficient way. These two enhancements markedly improve the performance of the approach.

Use of Sequential LCP Method to Solve LQ-BLPP $_k$

Given that the lower-level problem in LQ-BLPP $_k$ is a convex program in \mathbf{w} , it can be replaced by its Kuhn-Tucker conditions. Thus LQ-BLPP $_k$ is equivalent to the following minimum linear complementarity problem:

$$\begin{aligned}
& \min (C_1 \mathbf{x}^k + C_3 \mathbf{y}^k + c_1)^T \mathbf{z} + (C_3^T \mathbf{x}^k + C_2 \mathbf{y}^k + d_1)^T \mathbf{w} \\
& \text{subject to } 2Q\mathbf{w} + 2D\mathbf{z} + B'^T \boldsymbol{\gamma}' - \boldsymbol{\beta}' + (Q\mathbf{y}^k + D\mathbf{x}^k + d_2)^T \boldsymbol{\xi} = 0 \\
& A'\mathbf{z} + B'\mathbf{w} + \boldsymbol{\alpha}' = 0 \\
& (-\phi^k A')^T \mathbf{z} + (Q\mathbf{y}^k + D\mathbf{x}^k + d_2)^T \mathbf{w} = 0 \\
& \boldsymbol{\alpha}'^T \boldsymbol{\gamma}' = \boldsymbol{\beta}'^T \mathbf{w}' = 0 \\
& -1 \leq z_i \leq 1, \quad i = 1, \dots, n \\
& \mathbf{w}', \boldsymbol{\alpha}', \boldsymbol{\beta}', \boldsymbol{\gamma}' \geq 0, \boldsymbol{\xi} \text{ unrestricted}
\end{aligned} \tag{7.5}$$

where the vectors $\boldsymbol{\alpha}'$ and $\boldsymbol{\gamma}'$ have dimension equal to the number of rows of B' , $\boldsymbol{\beta}'$ has the same dimension as \mathbf{w}' , and $\boldsymbol{\xi}$, which is the vector of dual variables associated with the follower's equality constraint in LQ-BLPP_k, has dimension m .

The sequential LCP method looks for a global minimum of (7.5) by solving a sequence of parameterized linear complementarity problems LCP(λ_i) of the form:

$$\begin{aligned}
& (C_1 \mathbf{x}^k + C_3 \mathbf{y}^k + c_1)^T \mathbf{z} + (C_3^T \mathbf{x}^k + C_2 \mathbf{y}^k + d_1)^T \mathbf{w} \leq \lambda_i \\
& 2Q\mathbf{w} + 2D\mathbf{z} + B'^T \boldsymbol{\gamma}' - \boldsymbol{\beta}' + (Q\mathbf{y}^k + D\mathbf{x}^k + d_2)^T \boldsymbol{\xi} = 0 \\
& A'\mathbf{z} + B'\mathbf{w} + \boldsymbol{\alpha}' = 0 \\
& (-\phi^k A')^T \mathbf{z} + (Q\mathbf{y}^k + D\mathbf{x}^k + d_2)^T \mathbf{w} = 0 \\
& \boldsymbol{\alpha}'^T \boldsymbol{\gamma}' = \boldsymbol{\beta}'^T \mathbf{w}' = 0 \\
& -1 \leq z_i \leq 1, \quad i = 1, \dots, n \\
& \mathbf{w}', \boldsymbol{\alpha}', \boldsymbol{\beta}', \boldsymbol{\gamma}' \geq 0, \boldsymbol{\xi} \text{ unrestricted}
\end{aligned}$$

where $\{\lambda_i\}$ is a strictly decreasing sequence.

The procedure stops when no solution to LCP(λ_i) can be found. In this case, the solution of the previous problem LCP(λ_{i-1}) is an ε -global solution to LQ-BLPP_k. The method works well to achieve an ε -global solution but faces difficulties in establishing that such a solution has been found. In fact, showing that a linear complementarity problem has no solution is much harder than finding its solution. Nevertheless, there is no need to actually find a global optimum to LQ-BLPP_k; any feasible point (\mathbf{z}, \mathbf{w}) of LCP(λ_i) with $\lambda_i < 0$ is a descent direction. Since the sequential LCP method solves a sequence of LCPs with strictly decreasing values of λ_i , it can be terminated whenever a solution of a LCP(λ_i) is found such that $\lambda_i < 0$. This overcomes the main drawback of the algorithm.

Now suppose that a local minimum \mathbf{u}^k of LQ-BLPP_k is at hand. Consequently, no descent directions exist emanating from \mathbf{u}^k so the optimal value of LQ-BLPP_k is

nonnegative. At this point, the sequential LCP method has to perform its last step to assure that a local minimum has been attained. This is difficult to do, however. It is known that checking local optimality in nonconvex quadratic programming is NP-hard [P4] implying the same for quadratic bilevel programming and, by specialization, the same for linear bilevel programming (see [V2] for the proofs).

Exact Stepsizes

Given a point $\mathbf{u}^k = (\mathbf{x}^k, \mathbf{y}^k)$ in the inducible region and a feasible *IR* direction $\mathbf{d}^k = (\mathbf{z}^k, \mathbf{w}^k)$, an efficient procedure needs to be developed to compute the largest feasible stepsize σ_{\max} in the solution of Q-BLPP. For polyhedral sets such a procedure is usually stated in terms of a minimum ratio rule. In the case of an inducible region, the boundaries are only implicitly defined so the problem of finding σ_{\max} is more complicated. This issue is now addressed.

Let η be the number of active constraints (including the nonnegativity conditions $\mathbf{y} \geq 0$) at $\mathbf{u}^k + \sigma \mathbf{d}^k$ in $\text{LLP}(\mathbf{x})$, where σ is a small positive number. If $\eta = 0$ the dual multipliers $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ in the stationarity condition (7.3a) are zero at $\mathbf{u}^k + \sigma \mathbf{d}^k$, hence:

$$Q(\mathbf{y}^k + \sigma \mathbf{w}^k) + \mathbf{D}(\mathbf{x}^k + \sigma \mathbf{z}^k) + \mathbf{d}_2 = 0$$

The feasible *IR* direction $\mathbf{d}^k = (\mathbf{z}^k, \mathbf{w}^k)$ should satisfy these conditions and consequently the stepsize σ_{\max} is the largest value of σ such that

$$\begin{aligned} \mathbf{A}(\mathbf{x}^k + \sigma \mathbf{z}^k) + \mathbf{B}(\mathbf{y}^k + \sigma \mathbf{w}^k) &\leq \mathbf{b} \\ \mathbf{x}^k + \sigma \mathbf{z}^k &\geq 0, \quad \mathbf{y}^k + \sigma \mathbf{w}^k \geq 0 \end{aligned}$$

Therefore, a minimum ratio rule is sufficient to compute σ_{\max} when $\eta = 0$. Consider now the case where $\eta > 0$. Let

$$\mathbf{r}_i^T \mathbf{x} + \mathbf{s}_i^T \mathbf{y} = t_i, \quad i = 1, \dots, \eta \quad (7.6)$$

be the η lower-level active constraints at $\mathbf{u}^k + \sigma \mathbf{d}^k$ in $\text{LLP}(\mathbf{x})$. The stationarity condition at $\mathbf{u}^k + \sigma \mathbf{d}^k$ can be written as follows:

$$Q(\mathbf{y}^k + \sigma \mathbf{w}^k) + \mathbf{D}(\mathbf{x}^k + \sigma \mathbf{z}^k) + \mathbf{d}_2 + \delta_1 \mathbf{s}_1 + \dots + \delta_\eta \mathbf{s}_\eta = 0 \quad (7.7)$$

where δ_i , $i = 1, \dots, \eta$, are the corresponding nonnegative multipliers (formerly $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ in (7.3a)). Because \mathbf{Q} is a nonsingular matrix this last condition implies

$$\mathbf{y}^k + \sigma \mathbf{w}^k = -\mathbf{Q}^{-1} \mathbf{D}(\mathbf{x}^k + \sigma \mathbf{z}^k) - \mathbf{Q}^{-1} \mathbf{d}_2 - \delta_1 \mathbf{Q}^{-1} \mathbf{s}_1 - \dots - \delta_\eta \mathbf{Q}^{-1} \mathbf{s}_\eta$$

Replacing this expression for $\mathbf{y}^k + \sigma \mathbf{w}^k$ in the active constraints (7.6), the following linear system in the multipliers δ_i , $i = 1, \dots, \eta$, is obtained:

$$\mathbf{Z}\boldsymbol{\delta} = \mathbf{q}' + \sigma \mathbf{q}'' \quad (7.8)$$

where

$$\begin{aligned} \mathbf{Z} &= \begin{bmatrix} -\mathbf{s}_1^T \mathbf{Q}^{-1} \mathbf{s}_1 & \cdots & -\mathbf{s}_1^T \mathbf{Q}^{-1} \mathbf{s}_\eta \\ \vdots & \ddots & \vdots \\ -\mathbf{s}_\eta^T \mathbf{Q}^{-1} \mathbf{s}_1 & \cdots & -\mathbf{s}_\eta^T \mathbf{Q}^{-1} \mathbf{s}_\eta \end{bmatrix}, \quad \boldsymbol{\delta} = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_\eta \end{bmatrix} \\ \mathbf{q}' &= \begin{bmatrix} t_1 - \mathbf{r}_1^T \mathbf{x}^k + \mathbf{s}_1^T \mathbf{Q}^{-1} \mathbf{Dx}^k + \mathbf{s}_1^T \mathbf{Q}^{-1} \mathbf{d}_2 \\ \vdots \\ t_\eta - \mathbf{r}_\eta^T \mathbf{x}^k + \mathbf{s}_\eta^T \mathbf{Q}^{-1} \mathbf{Dx}^k + \mathbf{s}_\eta^T \mathbf{Q}^{-1} \mathbf{d}_2 \end{bmatrix} \text{ and} \\ \mathbf{q}'' &= \begin{bmatrix} -\mathbf{r}_1^T \mathbf{z}^k + \mathbf{s}_1^T \mathbf{Q}^{-1} \mathbf{Dz}^k \\ \vdots \\ -\mathbf{r}_\eta^T \mathbf{z}^k + \mathbf{s}_\eta^T \mathbf{Q}^{-1} \mathbf{Dz}^k \end{bmatrix} \end{aligned}$$

By solving the two $\eta \times \eta$ linear systems

$$\mathbf{Z}\boldsymbol{\nu}' = \mathbf{q}' \quad \text{and} \quad \mathbf{Z}\boldsymbol{\nu}'' = \mathbf{q}'' \quad (7.9)$$

the following linear relationship among all the η multipliers and the σ parameter is obtained:

$$\delta_i = \nu'_i + \nu''_i \sigma, \quad i = 1, \dots, \eta$$

The following theorem states the computational effort involved in determining the vectors $\boldsymbol{\nu}'$ and $\boldsymbol{\nu}''$.

Theorem 7.2.2 The total number of systems required to compute the largest feasible stepsize σ_{\max} is $\eta + 2$ (i.e., η systems associated with the terms $\mathbf{Q}^{-1} \mathbf{s}_i$ in \mathbf{Z} , \mathbf{q}' and \mathbf{q}'' , and two systems associated with the matrix $\mathbf{Z} = [z_{ij}]_{\eta \times \eta}$ in (7.9), where $z_{ij} = -\mathbf{s}_i^T \mathbf{Q}^{-1} \mathbf{s}_j$).

Proof: After solving the η systems $\mathbf{Q}\boldsymbol{\zeta}_i = \mathbf{s}_i$ ($i = 1, \dots, \eta$) for $\boldsymbol{\zeta}_i$ only inner products involving $\boldsymbol{\zeta}_i$ and other vectors are needed to construct \mathbf{Z} , \mathbf{q}' and \mathbf{q}'' which are the data used in (7.8). The two systems associated with the matrix \mathbf{Z} in (7.9) must then be solved to find the vectors $\boldsymbol{\nu}'$ and $\boldsymbol{\nu}''$. ■

The fact that the matrix \mathbf{Q} is symmetric positive definite means that its Cholesky factors can be computed allowing it to be written as $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$, where \mathbf{L} is a lower triangular matrix with positive diagonal elements. It is important to note that this factorization only has to be done once during the entire steepest descent procedure. At each iteration we must solve 2η triangular systems and the two systems in (7.9).

After finding the vectors $\boldsymbol{\nu}'$ and $\boldsymbol{\nu}''$, the stepsize σ_{\max} is the largest value of σ such that

$$\mathbf{A}(\mathbf{x}^k + \sigma \mathbf{z}^k) + \mathbf{B}(\mathbf{y}^k + \sigma \mathbf{w}^k) \leqq \mathbf{b}$$

$$\begin{aligned} \mathbf{x}^k + \sigma \mathbf{z}^k &\geq \mathbf{0}, \quad \mathbf{y}^k + \sigma \mathbf{w}^k \geq \mathbf{0} \\ \nu'_i + \nu''_i \sigma &\geq 0, \quad i = 1, \dots, \eta \end{aligned}$$

Hence σ_{\max} can be computed by using a minimum ratio rule.

For computing the exact stepsize σ_k , consider the function

$$G(\sigma) = \frac{1}{2}(\mathbf{u}^k + \sigma \mathbf{d}^k)^T \mathbf{C}(\mathbf{u}^k + \sigma \mathbf{d}^k) + (\mathbf{c}_1, \mathbf{d}_1)(\mathbf{u}^k + \sigma \mathbf{d}^k), \quad \sigma > 0$$

Since \mathbf{C} is a symmetric positive semidefinite matrix, then G is a convex function. Thus the unconstrained minimizer σ'_k can be found by solving $G'(\sigma) = 0$:

$$\sigma'_k = \begin{cases} -\frac{(\mathbf{c}_1, \mathbf{d}_1)\mathbf{d}^k + (\mathbf{d}^k)^T \mathbf{C}\mathbf{u}^k}{(\mathbf{d}^k)^T \mathbf{C}\mathbf{d}^k} & \text{if } (\mathbf{d}^k)^T \mathbf{C}\mathbf{d}^k > 0 \\ +\infty & \text{if } (\mathbf{d}^k)^T \mathbf{C}\mathbf{d}^k = 0 \end{cases} \quad (7.10)$$

and the exact stepsize σ_k is computed as follows:

$$\sigma_k = \begin{cases} \sigma'_k & \text{if } 0 < \sigma'_k < \sigma_{\max} \\ \sigma_{\max} & \text{otherwise} \end{cases} \quad (7.11)$$

Example 7.2.2 To illustrate the computation of the exact stepsize σ_k , consider the following Q-BLPP:

$$\min_{x_1, x_2} \quad \frac{1}{2}(x_1 - 1)^2 + \frac{1}{2} \left(x_2 - \frac{2}{5} \right)^2 + \frac{1}{2} \left(y - \frac{4}{5} \right)^2$$

subject to $0 \leq x_1, x_2 \leq 1$

$$\min_y \quad \frac{1}{2}y^2 + y - x_1y + 3x_2y$$

subject to $0 \leq y \leq 1$

The inducible region for this simple 3-variable problem is the union of the following sets:

$$\{(x_1, x_2, y) \in R^3 : x_1 \leq 1, x_2 \geq 0, -x_1 + 3x_2 \leq 0, y = 1\}$$

$$\{(x_1, x_2, y) \in R^3 : -x_1 + 3x_2 + y = 1, 0 \leq y \leq 1\}$$

$$\{(x_1, x_2, y) \in R^3 : x_1 \geq 0, x_2 \leq 1, -x_1 + 3x_2 \geq 1, y = 0\}$$

Figure 7.3 depicts the first of these sets which is the triangle formed by the vertices $V_1 = (1, 0, 1)$, $V_2 = (0, 0, 1)$ and $V_3 = (3/2, 1/2, 1)$. If $\mathbf{u}^0 = V_1$ then the steepest descent direction \mathbf{d}^0 is $(0, 1/3, 0)$ and a new point $\mathbf{u}^1 = V_4$ in the inducible region is computed by $\mathbf{u}^1 = \mathbf{u}^0 + \sigma_0 \mathbf{d}^0$, where $\sigma_0 = \sigma_{\max} = 1/3$. In the second iteration, $\mathbf{d}^1 = (0, 1/15, -1/5)$ and $\mathbf{u}^2 = \mathbf{u}^1 + \sigma_1 \mathbf{d}_1 = (1, 2/5, 4/5) = V_5$, where $\sigma_1 = \sigma'_1 = \sqrt{10}/15$. The point \mathbf{u}^2 is a local minimum and the steepest descent algorithm terminates. Note that in the first iteration the stepsize σ'_0 is not feasible and the displacement in the direction \mathbf{d}^0 is made by the largest feasible stepsize σ_{\max} . This is not the case for the second iteration where the stepsize σ'_1 is feasible.

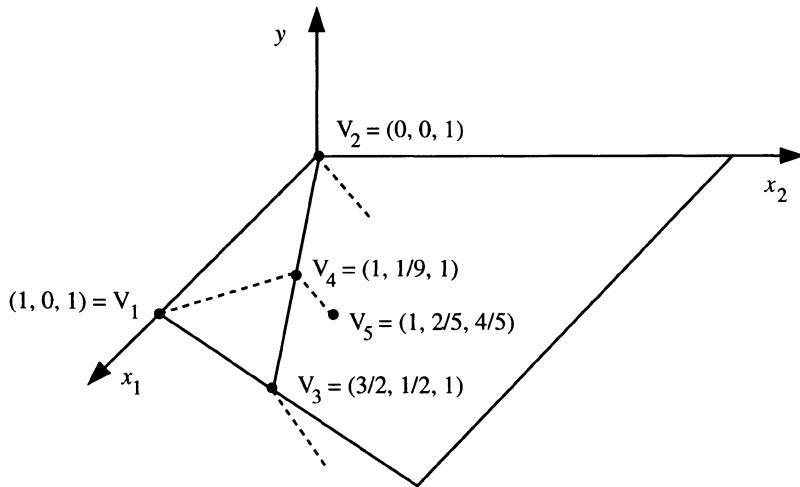


Figure 7.3 Computation of exact stepsizes σ_{\max} and σ'_k

It is easy to see that computing σ'_k is much less burdensome than finding the value of the stepsize σ_{\max} . An alternative approach that tries to skirt the computation of σ_{\max} at the expense of solving a strictly convex program is as follows:

- Compute σ'_k as specified in (7.10) and set $\bar{\mathbf{x}} = \mathbf{x}^k + \sigma'_k \mathbf{z}^k$.
- Solve the strictly convex quadratic program $\text{LLP}(\bar{\mathbf{x}})$ and let $\bar{\mathbf{y}}$ be its optimal solution. If $\bar{\mathbf{y}} = \mathbf{y}^k + \sigma'_k \mathbf{w}^k$ then we can consider the new point
$$\mathbf{u}^{k+1} = (\mathbf{x}^{k+1}, \mathbf{y}^{k+1}) = (\bar{\mathbf{x}}, \bar{\mathbf{y}})$$
- Otherwise σ'_k is not a local feasible stepsize and σ_{\max} has to be computed. Furthermore,

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \sigma_{\max} \mathbf{d}^k$$

The quadratic program $\text{LLP}(\bar{\mathbf{x}})$ can be solved in polynomial time, usually with a small number of iterations that are independent of the number η of active constraints. When

η is large the advantages of the above procedure may be significant; however, when σ'_k is not feasible we cannot avoid computing σ_{\max} .

7.2.3 Hybrid Approach and Concave Minimization

Comparing the *EIR* point algorithm with the steepest descent approach, we note that the former is much simpler but doesn't always converge to a local minimum of Q-BLPP. In this section, we present a hybrid algorithm designed to exploit the strengths of each.

Step 1 – Apply the descent *EIR* point algorithm. If it terminates with a local minimum, Stop. Otherwise let \mathbf{u}^k be the *LSIR* point obtained at the end of the procedure.

Step 2 – Solve LQ-BLPP $_k$ to get a descent direction \mathbf{d}^k . If the optimal value of this problem is greater than or equal to zero, Stop; \mathbf{u}^k is a local minimum of Q-BLPP. Otherwise compute σ_k as in (7.11) and set $\mathbf{u}^{k+1} = \mathbf{u}^k + \sigma_k \mathbf{d}^k$. Put $k \leftarrow k + 1$ and repeat Step 2.

The possibility of returning to Step 1 after several iterations of Step 2 might be a good alternative strategy. In that case, an initial *EIR* point can be found by using the procedure discussed in Section 7.2.1.

Consider again Q-BLPP and suppose that the upper-level objective function F is concave, that is, C is negative semidefinite. Then a well known result for concave programming also holds for the concave-quadratic BLPP.

Theorem 7.2.3 If $F(\mathbf{x}, \mathbf{y})$ is a strictly concave function then every local minimum of Q-BLPP is attained at an extreme inducible region point.

Proof: We start by showing that all the points on a given face of the inducible region form a polyhedral set. In fact, suppose that a given face is the set of η active constraints of the form (7.6). The stationarity condition in (7.7) represents a polyhedral set in $R^{n+m+\eta}$. The projection of this set onto R^{n+m} is also a polyhedral set (defined by at most m hyperplanes). The intersection of this last set with the given face in IR is a polyhedron.

Now let \mathbf{u} be a local minimum of Q-BLPP and recall that the inducible region of Q-BLPP is a finite union of polyhedral sets; i.e.,

$$IR = \bigcup_{i=1}^{\ell} P_i$$

where P_i is a polyhedral set, $i = 1, \dots, \ell$. Thus there exists at least one $k \in \{1, \dots, \ell\}$ such that $\mathbf{u} \in P_k$. Because \mathbf{u} is a local minimum over P_k the result follows immediately from concave minimization theory. ■

Suppose that the *EIR* point algorithm is applied to Q-BLPP when F is concave. If the last *EIR* point is nondegenerate, then it is an *LSIR* point and by Theorems 7.2.1 and 7.2.3 it is also a local minimum of Q-BLPP. Therefore, the modified steepest descent method is not required in this last case. When the last *EIR* point obtained by the *EIR* point algorithm is degenerate, though, there is no guarantee that such point is a local minimum.

As a final word in this section, we note that the possible occurrence of degeneracy in either descent algorithm has not been considered. Degeneracy is an important issue, especially for network-type models which are the basis of several bilevel programming applications in transportation and telecommunications. A second potential area of research is the extension of the descent algorithms to problems involving more general nonlinearities in $F(\mathbf{x}, \mathbf{y})$. Perhaps the first place to start is with the separable case where $F(\mathbf{x}, \mathbf{y}) = F_1(\mathbf{x}) + F_2(\mathbf{y})$.

7.3 BRANCH AND BOUND ALGORITHM

The first procedure for solving (7.1) when $f(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\mathbf{y}^T \mathbf{Q}\mathbf{y} + \mathbf{x}^T \mathbf{D}\mathbf{y} + \mathbf{d}_2\mathbf{y}$ and $\mathbf{g}(\mathbf{x}, \mathbf{y}) = \mathbf{A}_2\mathbf{x} + \mathbf{B}_2\mathbf{y} - \mathbf{b}_2$ was developed by Bard [B6]. As elaborated below, the basic idea was to first find a point in the inducible region and then iterate using an active set strategy to arrive at a local solution to (7.2a)–(7.2f). This was seen to furnish a good upper bound on F for fathoming nodes in a search tree. As in Section 5.3.2, let $W = \{1, \dots, q\}$ and P_k be the path vector at iteration k . Each path P_k in the tree corresponds to an assignment of either $u_i = 0$ or $g_i = 0$ for $i \in W_k \subseteq W$, thus identifying a partial solution. Specifically, let

$$\begin{aligned} S_k^+ &= \{i : i \in W_k \text{ and } u_i = 0\} \\ S_k^- &= \{i : i \in W_k \text{ and } g_i = 0\} \\ S_k^0 &= \{i : i \notin W_k\} = W \setminus W_k \\ J_k^- &= \{i : i \in S_k^-, g_i = 0, S_k^0 = \emptyset\} \end{aligned}$$

A completion of W_k is an assignment of either u_i or g_i for all i in the index set S_k^0 of free variables. Note that when $S_k^0 = \emptyset$ we are in the inducible region. The last set, J_k^- , identifies those surfaces which are incident to the current segment of the inducible region and, by Proposition 8.1.2, candidates for extending it in the direction of decreasing F . In all, the search tree contains $2^{q+1} - 1$ nodes. Upper and lower

bounds on F will be represented by \bar{F} and \underline{F} , respectively, while $S \setminus i$ will be used to denote $S \setminus \{i\}$ and $S \cup i$ to denote $S \cup \{i\}$ for any set S and element i .

Algorithm B&B

- Step 0 (Initialization) Solve $\min\{F(\mathbf{x}, \mathbf{y}) : \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}, \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}, \nabla_{\mathbf{y}}f(\mathbf{x}, \mathbf{y}) + \mathbf{u}\nabla_{\mathbf{y}}\mathbf{g}(\mathbf{x}, \mathbf{y}) = \mathbf{0}, \mathbf{u} \geq \mathbf{0}\}$ to get $(\mathbf{x}^0, \mathbf{y}^0, \mathbf{u}^0)$ and set $\underline{F} = F(\mathbf{x}^0, \mathbf{y}^0)$; set $\bar{F} = \infty$ and $k = 1$.
- Step 1 (Obtaining a feasible point) Fix \mathbf{x} at \mathbf{x}^0 and solve $\min_{\mathbf{y}}\{f(\mathbf{x}^0, \mathbf{y}) : \mathbf{g}(\mathbf{x}^0, \mathbf{y}) \leq \mathbf{0}\}$ to get a point $(\mathbf{x}^0, \hat{\mathbf{y}}^0)$ in IR . If $F(\mathbf{x}^0, \hat{\mathbf{y}}^0) = \underline{F}$ stop; otherwise determine the path P_k at $(\mathbf{x}^0, \hat{\mathbf{y}}^0)$ and fix the sets S_k^+ and S_k^- .
- Step 2 (Bounding) Solve $\min\{F(\mathbf{x}, \mathbf{y}) : \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}, \nabla_{\mathbf{y}}f(\mathbf{x}, \mathbf{y}) + \mathbf{u}\nabla_{\mathbf{y}}\mathbf{g}(\mathbf{x}, \mathbf{y}) = \mathbf{0}, g_i(\mathbf{x}, \mathbf{y}) = 0, i \in S_k^+, u_i = 0, i \in S_k^-\}$ to obtain $(\mathbf{x}^k, \mathbf{y}^k)$; put $\bar{F} = \min\{\bar{F}, F(\mathbf{x}^k, \mathbf{y}^k)\}$ and stop if $\bar{F} = \underline{F}$; if no solution exists fathom the node. Determine new J_k^- and go to Step 3.
- Step 3 (Advancing and fathoming) Select an $i \in J_k^-$ and call it i_1 . If none exists fathom all nodes on the path P_k such that $g_i = 0$ and go to Step 5. If the number of binding constraints is $< n + m$ then put $k \leftarrow k + 1$, $S_k^+ \leftarrow S_k^+ \cup i_1$, $S_k^- \leftarrow S_k^- \setminus i_1$ and $J_k^- \leftarrow J_k^- \setminus i_1$; otherwise select an $i \in S_k^+$, call it i_2 and put $k \leftarrow k + 1$, $S_k^+ \leftarrow S_k^+ \cup i_1 \setminus i_2$, $S_k^- \leftarrow S_k^- \cup i_2 \setminus i_1$, $J_k^- \leftarrow J_k^- \setminus i_1$. Go to Step 2.
- Step 4 (Backtracking) Select a new $i \in J_t^-$ ($t = 1, \dots, k - 1$) and call it i_1 . If none exists fathom all nodes on the paths P_t such that $g_i = 0$ and go to Step 6; otherwise select an $i \in S_t^+$, call it i_2 and put $k \leftarrow k + 1$, $S_k^+ \leftarrow S_t^+ \cup i_1 \setminus i_2$, $S_k^- \leftarrow S_t^- \cup i_2 \setminus i_1$, $J_k^- \leftarrow J_t^- \setminus i_1$. Go to Step 2.
- Step 5 (Branching) Select an unfathomed node on any of the paths P_k such that $i \in S_k^-$ and $g_i \neq 0$, call it i_3 ; if none exists go to Step 7; otherwise go to Step 6.
- Step 6 (Fathoming) Solve $\min\{F(\mathbf{x}, \mathbf{y}) : \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}, \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}, \nabla_{\mathbf{y}}f(\mathbf{x}, \mathbf{y}) + \sum_{i \in S_k^-} u_i \nabla_{\mathbf{y}}g_i(\mathbf{x}, \mathbf{y}) = \mathbf{0}, g_{i_3}(\mathbf{x}, \mathbf{y}) = 0\}$ to get a point $(\mathbf{x}^0, \mathbf{y}^0)$ in IR . If $F(\mathbf{x}^0, \mathbf{y}^0) \geq \bar{F}$, fathom all nodes on paths containing g_{i_3} and go to Step 4; otherwise put $k \leftarrow k + 1$ and go to Step 1.
- Step 7 (Degenerate case) Attempt to solve $\min\{F(\mathbf{x}, \mathbf{y}) : \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}, \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}, \nabla_{\mathbf{y}}f(\mathbf{x}, \mathbf{y}) = \mathbf{0}, u_i = 0, \forall i\}$ to get a point $(\mathbf{x}^0, \mathbf{y}^0)$ in IR . If a solution exists update \bar{F} . Go to Step 8.
- Step 8 (Termination) Declare the feasible point in IR associated with \bar{F} the optimal solution.

At Step 0 a relaxed problem is solved to get a lower bound on F . The resultant value of \mathbf{x} is then used at Step 1 to find a point in the inducible region and to test for global optimality. At Step 2 we minimize over the active constraint surface represented by S_k^+ to get $(\mathbf{x}^k, \mathbf{y}^k)$, and update the upper bound accordingly. This is an attempt to find a solution to (7.2). At Step 3, what might be termed an entering constraint is identified (i_1) which has the property that it is incident to the current segment of the inducible region; i.e., $g_{i_1}(\mathbf{x}^k, \mathbf{y}^k) = 0$ and $i_1 \notin S_k^+$. It is an empirical matter to decide how i_1 is chosen. If all the J_k^- have been exhausted, then, from the continuity of the inducible region (Proposition 8.1.2), we can conclude that a local optimum has been found. Finally, the choice of the leaving constraint (i_2) if necessary is problem dependent and thus fairly arbitrary. With certain qualifications the constraint associated with the smallest multiplier seems to produce good results.

At Step 3 when a local solution is reached all nodes along the paths where $\mathbf{g}(\mathbf{x}^k, \mathbf{y}^k) = 0$ can be fathomed, once again due to the continuity of IR and Corollary 7.1.2. This turns out to be an extremely powerful step, trimming the search tree by a factor as large as $2^{\bar{q}}$ (where \bar{q} is the number of active constraints). Backtracking at Step 4 is designed to return to a previously encountered point in the inducible region and to continue minimizing, but in a different direction. When all the potential paths are explored we branch at Step 5 and then solve a new subproblem at Step 6 to return to IR . The last problem to be solved at Step 7 corresponds to the case where all u_i are zero. The procedure terminates when every node has been fathomed.

Proposition 7.3.1 For F strictly convex in (\mathbf{x}, \mathbf{y}) , f quadratic in (\mathbf{x}, \mathbf{y}) , \mathbf{G} convex in (\mathbf{x}, \mathbf{y}) , and \mathbf{g} affine, Algorithm B&B terminates with a global optimum to problem (7.1).

Proof: By construction, the algorithm explores all nodes of the underlying search tree. And by hypothesis, all subproblems solved at Step 2 are convex, so for a given completion, no superior local optimum is overlooked. ■

Bookkeeping

The path P_k in the search tree can be concisely represented by an ℓ -dimensional vector, where ℓ is the current depth of the tree. The order of the components is determined by their level. Indices only appear in the vector P_k if they are in W_k , and appear underlined if the complementary condition has already been considered. To facilitate branching to any node in the tree the following notation is used. If $i \in W_k$, let it appear in P_k as

$$\begin{cases} i & \text{if } i \in S_k^+ \text{ and } u_i = 0 \text{ has not been considered} \\ \underline{i} & \text{if } i \in S_k^+ \text{ and } u_i = 0 \text{ has been considered} \\ -i & \text{if } i \in S_k^- \text{ and } g_i = 0 \text{ has not been considered} \\ -\underline{i} & \text{if } i \in S_k^- \text{ and } g_i = 0 \text{ has been considered} \end{cases}$$

When branching to say $g_t = 0$ from node ν_k we simply change P_k to (P_k, t) . In backtracking, the rightmost nonunderlined entry is underlined, its sign changed, and all entries to its right erased. If this procedure is followed branching between any two nodes can occur without the optimal solution being overlooked.

Example 7.1.1 (continued) To see how Algorithm B&B works let us return to the example in Fig. 7.1. Solving the relaxed problem at Step 1 yields the point $(4, 0)$ and a lower bound $F = 2$. Fixing x at 4 and solving the subproblem at Step 2 puts us on constraint 3 in the inducible region. Thus $S_1^+ = \{3\}$, $S_1^- = \{1, 2, 4\}$, $S_1^0 = \emptyset$, and $(x^0, \hat{y}^0) = (4, 3)$. Minimizing F over the current segment of the inducible region at Step 3 gives $(x^1, y^1) = (5, 2)$, an upper bound of $\bar{F} = 25$, and $J_1^- = \{2\}$.

The first path in the search tree as shown in Fig. 7.4 terminates at node 4, and may be represented by the vector $P_1 = (3, -1, -2, -4)$. At Step 4 we select $i_1 = 2$ and $i_2 = 3$ and return to Step 3 with $k = 2$ and $P_2 = (-3, -1, 2, -4)$. This path corresponds to node 8. An attempt to solve this problem fails due to infeasibility so the fathoming rule is applied. After eliminating all nodes on paths that contain $g_3 = 0$ and $g_2 = 0$ we backtrack to J_1^- which is now empty. Consequently, we branch to an unfathomed node at Step 6. The choices are those which are along the paths containing $g_1 = 0$ or $g_4 = 0$. Arbitrarily selecting the first and then solving the relaxed problem at Step 7 we arrive at the point $(1, 0)$ with $F = 17$. Setting $k = 3$ and returning to Step 2 we see that this point is in IR . Step 3 gives $\bar{F} = 17$ at node 11 in the tree with $P_3 = (-3, 1, -2, -4)$; Step 4 indicates that this is a local optimum, and proceeding, all paths containing $g_1 = 0$ and $g_4 = 0$ are fathomed.

Skipping to Step 8, the only subproblem remaining is associated with $u_i = 0$ for all i . The solution occurs at $(\frac{16}{9}, \frac{7}{3})$ in Fig. 7.1 with $F = 42.5$ and $P_4 = (-3, 1, -2, -4)$ in Fig. 7.4 so the corresponding node (13) is fathomed and the algorithm terminates. In all, 4 out of a possible 16 subproblems had to be solved, and 6 out of a possible 31 nodes had to be explored.

Comparative Results

To test the efficiency of the branch and bound algorithm with respect to the underlying search tree as well as other numerical procedures, three sets of sample problems were randomly generated and solved. Specifically, the algorithm was compared with the barrier method of Aiyoshi and Shimizu [A3] (see Section 8.3) and GRG2 [L1] applied directly to problem (7.2a)–(7.2f). The results are presented in Table 7.1. In all cases, nonnegativity of the variables was assumed and all upper-level constraints were a function of the leader's variables only; i.e., of the form $G(\mathbf{x}) \leq 0$.

Each problem set was characterized by the dimensions of \mathbf{x} and \mathbf{y} , and the number of independent and joint constraints. Ten cases were run for each set with performance being measured by the ‘average no. of iterations’, the ‘range of iterations’, and whether or not global optimality was achieved. Computation times are not reported

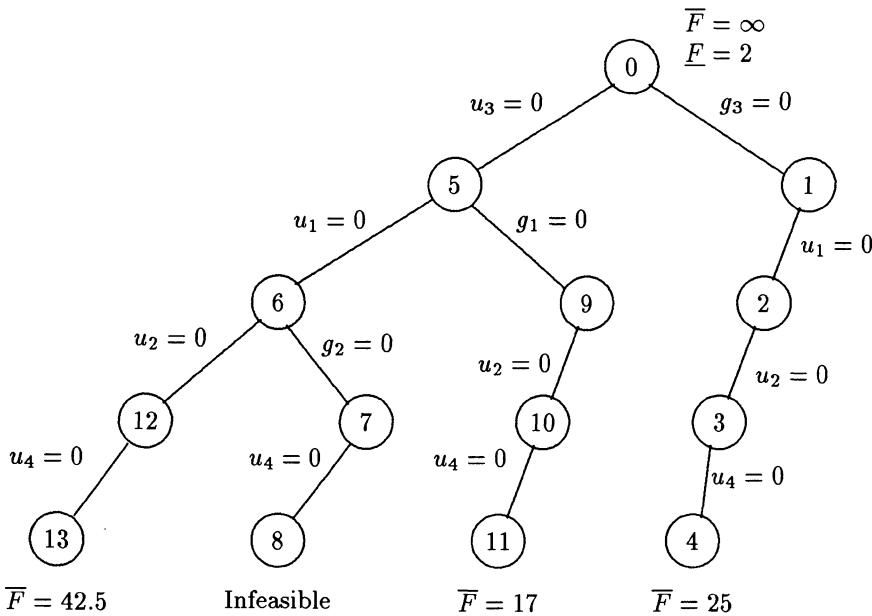


Figure 7.4 Search tree for Example 7.1.1

Table 7.1 Comparative results for branch and bound algorithm

Procedure	Problem size (n, p, m, q)	Average no. of iterations	Range of iterations	Problems solved ^a
B&B algorithm	(5,5,5,5)	24	2–84	100%
	(10,12,10,15)	77	21–174	100%
	(15,20,15,20)	209	62–743	100%
Barrier method	(5,5,5,5)	53	37–157	60%
	(10,12,10,15)	156	122–412	50%
	(15,20,15,20) ^b	—	—	—
GRG2	(5,5,5,5)	31	11–97	50%
	(10,12,10,15)	99	47–241	30%
	(15,20,15,20)	238	108–871	30%

^aGlobal optimum found.^bNo convergence for most problems after 10 minutes of CPU time.

because the analysis was done on different machines using codes embodying different levels of sophistication.

An iteration of the branch and bound algorithm is defined as the solution to one subproblem at a node in the search tree (these computations were performed with a reduced gradient code). Considering the first problem set where $m + q = 10$, the maximum number of nodes is $2^{11} - 1$. For the 10 problems investigated, 24 iterations on average were required indicating that only about 1% of the tree had to be explored. The results were not nearly as positive for the barrier method where an iteration is defined as a solution to problem (8.18) for a particular value of r . Here ϕ was taken to be logarithmic and GRG2 was used for the computations. The program was terminated when the objective function failed to improve by more than 2% in five consecutive iterations, or when the equivalent of 10 minutes of CPU time on an IBM 3081-D had elapsed. Notice that none of the problems in the third set met the convergence criterion within the allotted time, and for the first two sets global optimality proved elusive. For GRG2, an iteration is defined as the calculation of the reduced gradient and the solution to the corresponding line search problem. Although performance was relatively good, the global optimum was rarely uncovered, and in some cases, the code stopped with only a point in IR .

Discussion

The branch and bound algorithm presented in this section was designed to solve the BLPP when the leaders objective function and constraint set are convex, and the follower's problem is quadratic. By exploiting the properties of the inducible region an active set strategy permits the acquisition of a tight upper bound, thus reducing the number of subproblems that must be set up and solved. The numerical results bear this out; in all cases, only a small fraction of the nodes in the underlying search tree had to be examined. By way of comparison, the two other procedures investigated did not fare quite so well, primarily due to the sharp nonlinearities associated with the BLPP. In approximately 50% of the test cases they were unable to find global solutions which, in any case, could never be independently confirmed.

Additional testing by the author showed that the branch and bound algorithm can readily be modified to accommodate more general functional forms, but not without suffering some decline in efficiency. If the rational reaction set, $P(\mathbf{x})$, is single-valued, then the algorithm will also solve the linear BLPP with a minor modification to deal with the potential for multiple optima in problem (7.2a)–(7.2f). With regard to the nonlinear case, if the follower's problem is convex, it may be possible to replace it with its equivalent Lagrangian dual, write the first order stationarity conditions to get (7.2c), and then devise an iterative scheme that will converge, at least to a local optimum.

7.4 VARIABLE ELIMINATION ALGORITHM

Jaumard et al. [J1] extended the variable elimination algorithm presented in Section 5.3.4 to handle the case where $F(\mathbf{x}, \mathbf{y})$ and $\mathbf{G}(\mathbf{x}, \mathbf{y})$ are convex, $f(\mathbf{x}, \mathbf{y})$ is quadratic and $\mathbf{g}(\mathbf{x}, \mathbf{y})$ is affine. To begin, they present a series of necessary optimality conditions expressed in terms of the tightness of the constraints in the follower's subproblem; i.e., $\mathbf{A}_2\mathbf{x} + \mathbf{B}_2\mathbf{y} \leq \mathbf{b}_2$ and $\mathbf{y} \geq 0$. Associated with each such constraint is a binary variable α_i equal to 1 if the constraint is tight, and equal to 0 otherwise. The theorem below defines the logical relations among these variables by exploiting the monotonicity of the follower's objective function. These relations play a basic role in the branch and bound algorithm described presently.

The following sets are used in the development.

$$\begin{aligned}\Gamma_{j+} &= \{i \in \{1, \dots, q\} : y_j \text{ appears in } g_i(\mathbf{x}, \mathbf{y}) \text{ and } \nabla_{y_j} g_i(\mathbf{x}, \mathbf{y}) > 0, \text{ for all } \mathbf{x}, \mathbf{y} \\ &\quad \text{such that } \nabla_{y_j} f(\mathbf{x}, \mathbf{y}) < 0, \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq 0, \mathbf{x} \geq 0, \mathbf{y} \geq 0\} \\ \Gamma_{j-} &= \{i \in \{1, \dots, q\} : y_j \text{ appears in } g_i(\mathbf{x}, \mathbf{y}) \text{ and } \nabla_{y_j} g_i(\mathbf{x}, \mathbf{y}) < 0, \text{ for all } \mathbf{x}, \mathbf{y} \\ &\quad \text{such that } \nabla_{y_j} f(\mathbf{x}, \mathbf{y}) > 0, \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq 0, \mathbf{x} \geq 0, \mathbf{y} \geq 0\} \\ \Gamma_{j\pm} &= \{i \in \{1, \dots, q\} : y_j \text{ appears in } g_i(\mathbf{x}, \mathbf{y}) \text{ and } i \notin \Gamma_{j+} \cup \Gamma_{j-}\}\end{aligned}$$

The last set corresponds to the set of functions $\mathbf{g}(\mathbf{x}, \mathbf{y})$ that contains y_j and which are neither increasing nor decreasing with respect to y_j .

Theorem 7.4.1 For any point $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ in the inducible region of the convex BLPP (7.1) the tightness of the constraints in the follower's problem is such that

if f is decreasing with respect to y_j , i.e., $\nabla_{y_j} f(\mathbf{x}, \mathbf{y}) < 0$,

$$\text{then } \sum_{i \in \Gamma_{j+} \cup \Gamma_{j\pm}} \alpha_i \geq 1 \tag{7.12a}$$

if f is increasing with respect to y_j , i.e., $\nabla_{y_j} f(\mathbf{x}, \mathbf{y}) > 0$,

$$\text{then } \alpha_{q+j} + \sum_{i \in \Gamma_{j-} \cup \Gamma_{j\pm}} \alpha_i \geq 1 \tag{7.12b}$$

for $j = 1, \dots, m$.

Proof: Assume by contradiction that there exists some $j \in \{1, \dots, m\}$ such that $\nabla_{y_j} f(\mathbf{x}, \mathbf{y}) < 0$ and that (7.12a) does not hold for some rational solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$. Consider the vector $\mathbf{y} = \hat{\mathbf{y}} + \varepsilon e_j$ where e_j is the unit vector with the j th component equal to 1. Because f is continuous, there exists $\delta > 0$ such that $\nabla_{y_j} f(\mathbf{x}, \mathbf{y}) < 0$ for all $\varepsilon < \delta$. Moreover, as \mathbf{g} is a q -vector of continuous functions, there exists a δ' such that \mathbf{y} remains feasible for all $\varepsilon < \delta'$ in all constraints g_i with $i \in \Gamma_{j+} \cup \Gamma_{j\pm}$ (since

(7.12a) does not hold; i.e., constraints g_i with $i \in \Gamma_{j+} \cup \Gamma_{j\pm}$ are satisfied as strict inequalities). It is also easy to check that $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ satisfies the constraints with $i \in \Gamma_{j-}$, as the corresponding function is nonincreasing with respect to y_j . The feasibility is obvious for those g_i that do not contain y_j .

Consider now a vector \mathbf{y} with ε satisfying $0 < \varepsilon < \min\{\delta, \delta'\}$. Such a vector leads to a feasible solution with an objective function value lower than $f(\hat{\mathbf{x}}, \hat{\mathbf{y}})$, which is a contradiction. Similar reasoning for the relations in (7.12b) concludes the proof. ■

Corollary 7.4.1 For any optimal solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ of convex BLPP (7.1) the tightness of the constraints in the follower's problem is such that conditions (7.12a,b) are satisfied for all $j \in \{1, \dots, m\}$ such that $\nabla_{y_j} f(\hat{\mathbf{x}}, \hat{\mathbf{y}}) < 0$ and $\nabla_{y_j} f(\hat{\mathbf{x}}, \hat{\mathbf{y}}) > 0$, respectively.

Proof: By noting that optimal solutions of (7.1) are rational, the result follows immediately from Theorem 7.4.1. ■

7.4.1 Convex-Quadratic BLPP

Although the above theoretical results apply to the convex BLPP in general, the next algorithm we present is specialized to the convex-quadratic case. It may be written in the following general form

$$\begin{aligned} & \min_{\mathbf{x} \geq 0} F(\mathbf{x}, \mathbf{y}) \\ & \text{subject to } G(\mathbf{x}, \mathbf{y}) \leq 0 \\ & \min_{\mathbf{y} \geq 0} f(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \mathbf{y}^T \mathbf{Q} \mathbf{y} + \mathbf{x}^T \mathbf{D} \mathbf{y} + \mathbf{d} \quad (7.13) \\ & \text{subject to } \mathbf{A}_2 \mathbf{x} + \mathbf{B}_2 \mathbf{y} \leq \mathbf{b}_2 \end{aligned}$$

where \mathbf{Q} is an $m \times m$ symmetric positive semidefinite matrix and \mathbf{D} is an $n \times m$ matrix.

Separation Schemes

Jaumard et al. used three separation schemes in their branch and bound algorithm. The first is a binary separation on the 0-1 variables α_i ; i.e., the tightness of the constraints of the follower's problem. The second uses the logical relations obtained in Theorem 7.4.1. The third is a trichotomy based on the signs of the partial derivatives of the follower's objective function.

1. The first separation scheme (S1) involves fixing a subset of the binary variable α_i ($i = 1, \dots, q+m$) at 0 or 1. If $\alpha_i = 1$, the i th constraint of the follower's problem becomes an equality. It can then be used to eliminate one of the follower's

variables y_j . Additional logical relations of type (7.12a) or (7.12b) can then be derived. If $\alpha_i = 0$, the i th constraint in the follower's problem becomes a strict inequality, and from the complementary slackness theorem of convex programming, the corresponding dual variable in problem (7.2) must be equal to 0. This information is used in two different dual tests. The weakness of this separation scheme is that the optimal value of the relaxation always remains unchanged for one of the subproblems, independently of the branching rule selected.

2. The second separation scheme (S2) corresponds to multiple branching based on the logical relation, $r \triangleq \alpha_{i_1} + \alpha_{i_2} + \dots + \alpha_{i_p} \geq 1$. Separation is done progressively in the following manner: $\alpha_{i_1} = 1; \alpha_{i_1} = 0$ and $\alpha_{i_2} = 1; \alpha_{i_1} = 0$ and $\alpha_{i_2} = 0; \dots; \alpha_{i_1} = \alpha_{i_2} = \dots = \alpha_{i_{p-1}} = 0$ and $\alpha_{i_p} = 1$. Again, fixing an α_i at 1 leads to the elimination of one variable in the follower's problem while fixing an α_i at 0 leads to new information in the dual tests.

A useful observation when defining a branching rule is that if all slack variables of the constraints associated with the binary variables of the relation r have nonnegative value, all leader's relaxations of the new subproblems are improved (since in each branch there is always a binary variable fixed at 1, i.e., a slack variable fixed at 0). Finally, there are many possible orderings of the binary variables in the relation r .

3. The third separation scheme (S3) requires the sign of a partial derivative $\nabla_{y_j} f$ to be negative, positive, or equal to 0 for some $j \in \{1, \dots, m\}$. This separation scheme is of particular interest when the function f is quadratic, since the additional constraints (7.2c) are linear. When considering a branch in which $\nabla_{y_j} f(\mathbf{x}, \mathbf{y}) < 0$ or $\nabla_{y_j} f(\mathbf{x}, \mathbf{y}) > 0$, additional logical relations of the form (7.12a) or (7.12b) can be derived. When considering a branch in which $\nabla_{y_j} f(\mathbf{x}, \mathbf{y}) = 0$, one of the follower's variables can be eliminated. This separation scheme is considered only when the monotonicity of follower's objective function is unknown for some of the variables.

In practice, variable elimination can be performed by pivoting in a simplex tableau and fixing the eliminated variable, which leaves the basis at 0. This can be easily done when using an LP solver such as CPLEX or a nonlinear programming solver such as MINOS.

Subproblems

Assume that some of the follower's variables have been eliminated. Let the current vector of remaining second level variables be denoted by $\tilde{\mathbf{y}}$. The symbol \sim is used to identify vectors and matrices that have been modified accordingly. At each node of the branch and bound tree, the current bilevel subproblem can be written as follows:

$$\begin{aligned}
& \min_{\mathbf{x}, \mathbf{v}, \mathbf{u}} F(\mathbf{x}, \tilde{\mathbf{y}}) \\
& \text{subject to } \mathbf{G}(\mathbf{x}, \tilde{\mathbf{y}}) \leq \mathbf{0} \\
& \quad \left(\mathbf{u} \tilde{\mathbf{B}}_2 + \mathbf{v} \tilde{\mathbf{B}}_3 + \mathbf{x}^T \tilde{\mathbf{D}} + \mathbf{y}^T \tilde{\mathbf{Q}} \right)_i \geq -\tilde{d}_i \quad \text{for } i \in I_{DF} \\
& \quad \mathbf{x} \geq \mathbf{0}, \mathbf{u} \geq \mathbf{0}, \mathbf{v} \stackrel{\leq}{>} \mathbf{0}
\end{aligned} \tag{7.14}$$

where $\tilde{\mathbf{y}}$ is the solution of the lower-level problem defined by

$$\min_{\tilde{\mathbf{y}}} f(\mathbf{x}, \tilde{\mathbf{y}}) = \frac{1}{2} \tilde{\mathbf{y}}^T \tilde{\mathbf{Q}} \tilde{\mathbf{y}} + \mathbf{x}^T \tilde{\mathbf{D}} \tilde{\mathbf{y}} + \tilde{\mathbf{d}}^T \tilde{\mathbf{y}} \tag{7.15a}$$

$$\text{subject to } \mathbf{A}_2 \mathbf{x} + \tilde{\mathbf{B}}_2 \tilde{\mathbf{y}} \leq \mathbf{b}_2 \tag{7.15b}$$

$$\nabla_{y_i} f(\mathbf{x}, \tilde{\mathbf{y}}) > 0 \quad \text{for } i \in I^+ \tag{7.15c}$$

$$\nabla_{y_i} f(\mathbf{x}, \tilde{\mathbf{y}}) < 0 \quad \text{for } i \in I^- \tag{7.15d}$$

$$\nabla_{y_i} f(\mathbf{x}, \tilde{\mathbf{y}}) = 0 \quad \text{for } i \in I^0 \tag{7.15e}$$

$$\tilde{\mathbf{y}} \geq \mathbf{0} \tag{7.15f}$$

In the second constraint in (7.14), I_{DF} contains the indices of the dual feasibility constraints that have been introduced, and I^+ , I^- and I^0 contain the variable indices on which branching has taken place within the third separation scheme. The sign of the individual components of \mathbf{v} depends on which set i is in with regard to constraints (7.15c,d,e). The matrix $\tilde{\mathbf{B}}_3$ is derived from the constraints on the partial derivatives in (7.15c,d,e) and hence is a subset of the rows of $\tilde{\mathbf{Q}}$. That is, it has a row for each i in the sets I^+ , I^- and I^0 , and a column for each component of $\tilde{\mathbf{y}}$.

The current *leader's relaxation* (LR) is defined by (7.14) and (7.15b)–(7.15f); i.e., the second objective function (7.15a) is omitted. The current follower's relaxation (FR($\tilde{\mathbf{x}}$)) consists of (7.15a)–(7.15f) with \mathbf{x} fixed at $\tilde{\mathbf{x}}$. Observe that some of the constraints are strict inequalities (i.e., some of the constraints in (7.15b,f)) and all constraints (7.15c,d). These are difficult to handle in a primal algorithm so the following dual problem (DFR($\tilde{\mathbf{x}}$)) is considered instead.

$$\begin{aligned}
& \min \mathbf{u}(\mathbf{b}_2 - \mathbf{A}_2 \tilde{\mathbf{x}}) + \mathbf{v} \tilde{\mathbf{b}}_3 + \frac{1}{2} \tilde{\mathbf{y}}^T \tilde{\mathbf{Q}} \tilde{\mathbf{y}} \\
& \text{subject to } \mathbf{u} \mathbf{B}_2 + \mathbf{v} \tilde{\mathbf{B}}_3 + \tilde{\mathbf{x}}^T \tilde{\mathbf{D}} + \tilde{\mathbf{y}}^T \tilde{\mathbf{Q}} \geq -\tilde{\mathbf{d}} \\
& \quad \tilde{\mathbf{y}} \geq \mathbf{0}, \mathbf{u} \geq \mathbf{0}, \mathbf{v} \stackrel{\leq}{>} \mathbf{0}
\end{aligned} \tag{7.16}$$

where $\tilde{\mathbf{b}}_3$ corresponds to the constant terms in constraints (7.15c,d,e) for $\tilde{\mathbf{x}}$ fixed. That is, $\tilde{\mathbf{b}}_3$ has one component for each i in the sets I^+ , I^- and I^0 and the value of the i th component is $\tilde{d}_i + \tilde{\mathbf{x}}^T \tilde{\mathbf{D}}_i$ where $\tilde{\mathbf{D}}_i$ is the i th column of $\tilde{\mathbf{D}}$. Moreover, the strict inequalities in (7.15c,d) are dealt with in the dual (7.16) by setting the corresponding

values of v_i to zero. That is,

$$\begin{aligned} v_i &= 0 \quad \text{if } \alpha_i = 0, \quad i = 1, \dots, q \\ (\mathbf{u}B_2 + \mathbf{v}\tilde{B}_3 + \tilde{\mathbf{x}}^T \tilde{\mathbf{D}} + \mathbf{y}^T \tilde{\mathbf{Q}})_i &= -\tilde{d}_i \quad \text{if } \alpha_{q+i} = 0, \quad i = 1, \dots, m \end{aligned}$$

Finally, to be able to check that a given solution is rational for the original problem, we need to introduce the *dual follower problem*, DFP($\tilde{\mathbf{x}}$):

$$\begin{aligned} \min \quad & \mathbf{u}(\mathbf{b}_2 - \mathbf{A}_2 \tilde{\mathbf{x}}) + \frac{1}{2} \mathbf{y}^T \mathbf{Q} \mathbf{y} \\ \text{subject to} \quad & \mathbf{u}B_2 + \tilde{\mathbf{x}}^T \mathbf{D} + \mathbf{y}^T \mathbf{Q} \geq -\mathbf{d} \\ & \mathbf{y} \geq \mathbf{0}, \quad \mathbf{u} \geq \mathbf{0} \end{aligned} \tag{7.17}$$

Note that in the algorithm, (7.16) and (7.17) are solved for fixed values of \mathbf{x} and \mathbf{y} . This results in two linear programs denoted by DFR($\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$) and DFP($\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$), respectively, instead of two quadratic programs. This is a relaxation of the two dual problems but sufficient to guarantee convergence.

Algorithm SJX

- Step a (Initialization) Obtain an initial solution $(\mathbf{x}_h, \mathbf{y}_h) \in IR$ with a heuristic. Set incumbent solution $(\mathbf{x}_{opt}, \mathbf{y}_{opt})$ to $(\mathbf{x}_h, \mathbf{y}_h)$ and incumbent objective value F_{opt} to $F_{opt}(\mathbf{x}_{opt}, \mathbf{y}_{opt})$. If no heuristic solution can be found initialize $(\mathbf{x}_{opt}, \mathbf{y}_{opt})$ to arbitrary values and set F_{opt} to $+\infty$. Consider all logical variables α_i ($i = 1, \dots, q + m$) to be free. Set R , I^+ , I^- and I^0 to the empty set.
- Step b (First direct feasibility test) Solve LR: if infeasible go to Step j (backtracking).
- Step c (First direct optimality test) If LR is feasible, let $(\mathbf{x}_L^*, \mathbf{y}_L^*)$ denote the optimal solution. If $F_{LR}^* = F(\mathbf{x}_L^*, \mathbf{y}_L^*) \geq F_{opt}$, go to Step j.
- Step d (First direct solution test: Part 1) Solve the dual problem DFR($\mathbf{x}_L^*, \mathbf{y}_L^*$). If DFR($\mathbf{x}_L^*, \mathbf{y}_L^*$) is infeasible and
 - if all the dual feasible constraints are present go to Step j; otherwise choose the most violated dual feasibility constraint i , add it to I_{DF} and go to Step b.
 If DFR($\mathbf{x}_L^*, \mathbf{y}_L^*$) is feasible, check if $(\mathbf{x}_L^*, \mathbf{y}_L^*)$ is in the inducible region for the current subproblem.
 - If $f_{DFR}^* = f(\mathbf{x}_L^*, \mathbf{y}_L^*)$, then $(\mathbf{x}_L^*, \mathbf{y}_L^*)$ is rational. (Note that it is not possible to update the incumbent at this point because we only know that $(\mathbf{x}_L^*, \mathbf{y}_L^*)$ is rational for the current subproblem defined by (7.14) and (7.15a)–(7.15f), and not necessarily for the original problem.) Otherwise go to Step f.

- Step e (First direct solution test: Part 2) Consider again the optimal solution $(\mathbf{x}_L^*, \mathbf{y}_L^*)$ of LR. Check if it is rational for the original problem; i.e., solve DFP($\mathbf{x}_L^*, \mathbf{y}_L^*$). If $f_{DFP}^* = f(\mathbf{x}_L^*, \mathbf{y}_L^*)$, then $(\mathbf{x}_L^*, \mathbf{y}_L^*)$ is rational; otherwise go to Step f. Update the incumbent objective function value F_{opt} and solution $(\mathbf{x}_{opt}, \mathbf{y}_{opt})$ and go to Step j.
- Step f (Second direct optimality test) If the set R of logical relations of type (7.12a) or (7.12b) contains a relation $r_k = \sum_{j \in I_k} \alpha_j \geq 1$ such that $\alpha_j = 0$ for all $j \in I_k \subseteq \{1, \dots, q+m\}$, go to Step j.
- Step g (Relational optimality test) For all remaining y_j appearing in $f(\mathbf{x}, \mathbf{y})$ for which the partial derivatives $\nabla_{y_j} f$ is of constant sign, add to R the logical relations (7.12a) or (7.12b) on the α_i 's if they are nonredundant. Eliminate from R those relations that have become redundant.
- Step h (First conditional optimality test) If R contains a relation r_k such that $\alpha_j = 0$ for all $j \in I_k$ except for one index i , set the corresponding α_i to 1. Eliminate from the subproblem a variable y_j remaining in the i th constraint such that the fill-in is minimum and return to Step b.
- Step i (Branching) Apply the selected branching rule to choose either a free variable α_i (separation scheme S1), or a relation $r_k \in R$ (separation S2), or by imposing the sign on a partial derivative (separation S3). In the last case, update one of the sets I^+ , I^- or I^0 if necessary.
- Step j (Backtracking) Consider the last node where unexplored branches remain and investigate one of the subproblems according to the selected branching rule. If all the branches of all the nodes have been explored, stop. Otherwise, update the current subproblem and return to Step b.

It is important to verify that the algorithm correctly identifies a rational solution at Steps d and e. This is shown in the following lemma.

Lemma 7.4.1 If a solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is rational and a constraint qualification exists for all such points in the follower's problem then there exists an optimal solution $\hat{\mathbf{u}}$ of DFP($\hat{\mathbf{x}}, \hat{\mathbf{y}}$) with value f_{DFP}^* such that $f_{DFP}^* = f(\hat{\mathbf{x}}, \hat{\mathbf{y}})$.

Proof: Under the convexity assumption and constraint qualification assumption for the follower at $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$, the follower's problem is stable for a given value of $\hat{\mathbf{x}}$ (i.e., for \mathbf{x} in the neighborhood of $\hat{\mathbf{x}}$, the follower's solution does not change abruptly), and strong duality results hold. ■

Proposition 7.4.1 The variable elimination algorithm solves the convex-quadratic BLPP (7.13) in a finite number of iterations.

The proof is based on the equivalent Kuhn-Tucker formulation (7.2a)-(7.2f) and the fact that the accompanying complementarity conditions (7.2c) can only be satisfied in a finite number of ways. See Jaumard et al. [J1] for the details.

7.4.2 Computational Experience

The extended variable elimination algorithm was implemented by the authors in FORTRAN 77 and tested on a Sun Sparcstation 10. The first experiments conducted were aimed at investigating different branching rules and problem sizes. Next, an effort was made to determine the relationship between the computational effort and the degree of nonlinearity associated with the follower's objective function. The final tests were designed to compare performance with the Bard-Moore algorithm (see Section 5.3.2) for the linear-quadratic case. For the computations, all linear and nonlinear subproblems were solved with MINOS version 5.1. The test problems were randomly generated following closely the construction schemes used by Bard and Moore [B11] and Hansen et al. [H1] for linear bilevel programs, and by Pardalos [P3] for convex functions.

Comparison of Branching Rules and Initial Testing

Four different branching rules were tested. To begin, let s_i be the slack variable associated with the constraint (or bound) $g_i(\mathbf{x}, \mathbf{y}) \leq 0$ in the current LR subproblem and let u_i be the corresponding dual variable in DFP (or DFR if this problem is to be solved at the current node). Now, if the current solution is not rational then the complementary condition must not be satisfied, so we must have $s_i u_i > 0$ for at least one i . The proposed branching rules below use this information as a criterion to select the branching variables or relations.

- (BR1): (i) (Separation S2) Select the logical relation $r_k \in R$ that has cardinality two and has both corresponding slack variables s_i in the basis. If there is more than one such logical relation, select that which maximizes $\sum_{i \in I_k} s_i u_i$. Order the variables α_i in r_k by decreasing values of the product $s_i u_i$.
 - (ii) (Separation S1) If there is no such relation, select the α_i associated with the largest $s_i u_i$ and branch first on $\alpha_i = 1$.
- (BR2): Same as (BR1) except that in (ii), branch first on $\alpha_i = 0$.
- (BR3): (i) Select a variable y_i such that the monotonicity of follower's objective function is unknown and such that the corresponding product $s_i u_i$ is maximum. Define three branches by imposing the sign of the corresponding partial derivative $\nabla_{y_i} f$ to be respectively nonnegative, nonpositive and zero;
 - (ii) If there is no such variable, use (BR1).
- (BR4): Same as (BR3) except that in (ii), use (BR2).

These branching schemes are compared in Table 7.2, where n denotes the number of leader variables, m the number of follower variables, and q the number of follower

constraints; no first-level constraints were included in the test problems ($p = 0$). The output includes the ‘No. of nodes’ in the search tree and the ‘CPU times’ in seconds. The means (μ) and standard deviations (σ) are reported for either measure.

In all, 15 test problems were generated and solved for each of three combinations of (n, m, q) . The density of the A_2 and B_2 matrices was held fixed at 8%. In each instance, approximately 60% of the terms in the upper- and lower-level objective functions were nonlinear.

Table 7.2 Comparison of separation and branching schemes

n	m	q	Density	Branching scheme	No. of nodes		CPU time (sec)	
					μ	σ	μ	σ
40	30	28	8%	(BR1)	79.5	45.9	35.9	18.9
				(BR2)	198.2	63.1	75.5	23.3
				(BR3)	126.0	79.9	54.6	33.0
				(BR4)	948.6	745.8	361.2	285.5
40	40	30	8%	(BR1)	489.8	200.9	388.5	152.1
				(BR2)	842.2	339.6	716.7	286.4
				(BR3)	375.1	156.7	303.9	126.6
				(BR4)	818.8	251.2	712.6	218.2
50	50	40	8%	(BR1)	208.8	81.4	376.2	154.4
				(BR2)	608.0	195.8	1059.6	372.6
				(BR3)	345.3	172.9	625.3	327.9
				(BR4)	777.1	263.2	1382.5	506.3

From Table 7.2 we see that branching rules (BR1) and (BR3) give the best results; i.e., the best computational performance is achieved by first considering the branch where the logical variable α_i equals one in the search tree (the corresponding constraint is made an equality). Note that this behavior differs from that of the Bard–Moore algorithm which performs better when first fixing the dual variable to 0 (which is equivalent to fixing the logical variable α_i to 0). This phenomenon can be explained by the use of the dual test (Steps d and e) which detects more quickly a rational solution.

With regard to the separation schemes, those involving constraints on the sign of partial derivatives did not appear to be significantly more efficient. Consequently, the authors adopted (BR1) for all subsequent testing. Table 7.3 presents results for problems of increasing size and density where both objective functions are quadratic.

Table 7.3 Computational results for problems of increasing size

<i>n</i>	<i>m</i>	<i>q</i>	Density	No. of nodes		CPU time (sec)	
				μ	σ	μ	σ
30	20	20	40%	52.9	66.1	16.8	18.7
			17%	63.7	108.4	14.2	19.5
30	30	24	40%	127.5	93.6	68.1	50.5
			17%	175.1	199.2	96.5	111.8
40	30	28	17%	168.6	165.7	135.9	145.2
			8%	41.1	88.8	20.2	42.2
40	40	32	8%	193.6	186.0	147.5	143.1
50	40	40	8%	438.3	247.2	501.1	272.1
60	40	44	8%	197.0	109.4	259.6	141.9

Effect of Lower-Level Nonlinear Objective Function

Next the authors tried to determine how nonlinearities in the follower's objective function affected algorithmic performance. Table 7.4 shows the results obtained for problems in which the percentage of linear terms in $f(\mathbf{x}, \mathbf{y})$ is 100% (all linear), 50% and 0% (all quadratic). In all cases, the upper-level objective function contained approximately 60% quadratic terms. The results are for 15 problem instances in each category.

Table 7.4 Nonlinearity effect of follower's objective function

<i>n</i>	<i>m</i>	<i>q</i>	Density	Linear terms	No. of nodes		CPU time (sec)	
					μ	σ	μ	σ
30	30	24	17%	100 %	68.4	71.7	27.1	33.5
				50 %	107.8	128.6	49.6	60.4
				0 %	485.3	625.8	243.5	304.1
40	40	32	8%	100 %	35.4	8.0	25.8	5.6
				50 %	112.5	59.1	89.8	48.1
				0 %	215.4	94.7	207.9	93.1

As expected, the data in Table 7.4 indicate that the computational effort depends on the nonlinearity associated with $f(\mathbf{x}, \mathbf{y})$. The average CPU time required to solve problems with an all quadratic lower-level objective function was about 8 to 9 times greater than the time needed to solve problems of the same size and density but with a linear lower-level objective function.

Comparison with Bard–Moore Algorithm

When the leader's objective function and constraints in (7.13) are linear and the follower's problem is quadratic, the Bard–Moore algorithm (BM) can be used to solve the resultant BLPP (see 5.3.2). This allowed the variable elimination algorithm (JSX) to be compared directly with the former. Again, a series of test problems was generated for various combinations of (n, m, q) and matrix densities. Each contained 15 instances. The computational results are presented in Table 7.5.

Table 7.5 Comparison with Bard–Moore algorithm

<i>n</i>	<i>m</i>	<i>q</i>	Density	Algorithm	Number of nodes		CPU time (sec)	
					μ	σ	μ	σ
30	30	24	40%	JSX	175.8	229.9	70.7	90.4
				BM	365.4	299.3	85.8	75.8
			17%	JSX	121.0	145.6	38.3	42.7
				BM	267.7	333.8	38.6	38.8
			8%	JSX	27.7	37.4	7.6	9.7
				BM	173.4	258.4	10.5	14.4
				JSX	118.3	49.7	60.9	25.1
			40%	BM	93.8	21.3	28.1	6.3
				JSX	21.5	6.0	13.8	3.6
				BM	218.0	81.8	58.5	21.6
40	30	28	8%	JSX	29.8	11.4	9.1	3.1
				BM	463.2	258.8	51.6	32.6
				JSX	348.1	651.9	280.7	527.2
			17%	BM	1615.7	2490.7	437.6	546.6
				JSX	67.6	26.3	35.7	13.1
			40	BM	507.8	230.7	77.5	36.0
				JSX	266.4	88.5	220.2	74.7
				BM	8 of 15 problems not solved			
50	40	40	8%	JSX	312.7	181.2	374.0	224.5
				BM	10 of 15 problems not solved			

For the first three combinations of (n, m, q) , better performance was obtained by algorithm JSX except for the case where $n = 40, m = 30, q = 28$ and the matrix density was 40%. For a majority of the problem instances in the last two categories, algorithm BM was unable to find the optimal solution within 10,000 nodes. Moreover, it was observed that:

- (i) problem density has a larger effect on the performance of algorithm JSX than on algorithm BM. For test problems of identical size, the variable elimination algorithm was much more efficient on sparse problems;
- (ii) problems with a large number of lower-level variables are more difficult for both algorithms than problems with a large number of upper-level variables.

GENERAL BILEVEL PROGRAMMING

8.1 INTRODUCTION

Of the various types of mathematical two-level structures, the general bilevel programming problem is the most challenging. It was originally proposed as a model for a leader–follower game in which two players try to minimize their individual objective functions $F(\mathbf{x}, \mathbf{y})$ and $f(\mathbf{x}, \mathbf{y})$, respectively, subject to a series of interdependent constraints [S12, S13, S16]. Once again, the underlying assumptions are that full information is available, at least to the leader, and that cooperation is prohibited. This precludes the use of correlated strategies and side payments.

For completeness, we now formulate the general model. To begin, let us consider the lower-level problem for \mathbf{x} fixed; in particular, the follower must solve

$$\begin{aligned} & \min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \\ & \text{subject to } \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \end{aligned} \tag{8.1}$$

where $f : R^n \times R^m \rightarrow R^1$ and $\mathbf{g} : R^n \times R^m \rightarrow R^q$. For some “parameter” \mathbf{x} , denote the set of feasible solutions to (8.1) by $S(\mathbf{x}) \triangleq \{\mathbf{y} : \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}\}$ and the optimal solution set by $P(\mathbf{x}) \triangleq \{\mathbf{y} : \mathbf{y} \in \arg \min(f(\mathbf{x}, \hat{\mathbf{y}}) : \hat{\mathbf{y}} \in S(\mathbf{x}))\}$. Using the same terminology as in Chapter 5, $P(\mathbf{x})$ is called the rational reaction set and $\mathbf{y} \in P(\mathbf{x})$ a rational response. The set $IR = \{(\mathbf{x}, \mathbf{y}) : \mathbf{x} \text{ feasible}, \mathbf{y} \in P(\mathbf{x})\}$ is known as the inducible region. To avoid situations where (8.1) is not well posed, it is natural to assume that $S(\mathbf{x}) \neq \emptyset$ and $P(\mathbf{x}) \neq \emptyset$.

Next we formulate the leader’s problem with respect to the parameter \mathbf{x} .

$$\begin{aligned} & \min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}) \\ & \text{subject to } \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \end{aligned} \tag{8.2}$$

where $F : R^n \times R^m \rightarrow R^1$, $\mathbf{G} : R^n \times R^m \rightarrow R^p$, and $\mathbf{y} \in P(\mathbf{x})$. We call (8.2) the upper-level problem. Combining (8.1) and (8.2) gives the general bilevel program-

ming problem (BLPP) which takes the form of a static Stackelberg game. When the rational reaction set, $P(\mathbf{x})$, is not single-valued problem (8.2) is not well defined. This situation will be discussed presently. Alternatively, when the follower's solution \mathbf{y} is unique for a given \mathbf{x} , the BLPP can be expressed as:

$$\min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}(\mathbf{x})) \quad (8.3a)$$

$$\text{subject to } \mathbf{G}(\mathbf{x}, \mathbf{y}(\mathbf{x})) \leq \mathbf{0} \quad (8.3b)$$

$$f(\mathbf{x}, \mathbf{y}(\mathbf{x})) = \min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \quad (8.3c)$$

$$\text{subject to } \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \quad (8.3d)$$

where $P(\mathbf{x}) = \{\mathbf{y}(\mathbf{x})\}$ and $\mathbf{y}(\mathbf{x}) = \arg \min\{f(\mathbf{x}, \hat{\mathbf{y}}) : \hat{\mathbf{y}} \in S(\mathbf{x})\}$. From the leader's perspective, (8.3) can be viewed as a mathematical program with an implicitly defined constraint region given by the follower's problem (8.3c,d). Once the vector \mathbf{x} is chosen, though, the follower simply faces a standard optimization problem.

Before getting to the details of the algorithms developed over the last few years to solve particular instances of (8.3), it is important to reiterate the difficulties that can surface when $P(\mathbf{x})$ is not single-valued. As mentioned in Section 1.3.1, if all of the constraint functions in (8.3d) were linear, it is possible that $P(\mathbf{x})$ might consist of some nontrivial subset of a hyperplane. In this case, the follower would be indifferent to any point on that hyperplane; however, the leader might have a specific preference. When he plays \mathbf{x}^* , his best result might only be realized at a particular point in $P(\mathbf{x}^*)$ but there may be no way to induce the follower to select that point; call it $\mathbf{y}^* \in P(\mathbf{x}^*)$. It may further be true that if the leader chooses any point other than \mathbf{x}^* , his potential minimum cost will never be realized.

The following examples from [B7] illustrate the difficulties that often arise when $P(\mathbf{x})$ is multivalued and discontinuous.

Example 8.1.1 For $\mathbf{x} \in R^1$ and $\mathbf{y} \in R^2$, consider

$$\min_{\mathbf{x} \geq 0} F(\mathbf{x}, \mathbf{y}) = x + y_2$$

$$\text{subject to } 2 \leq x \leq 4$$

$$\min_{\mathbf{y} \geq 0} f(\mathbf{x}, \mathbf{y}) = 2y_1 + xy_2$$

$$\text{subject to } x - y_1 - y_2 \leq -4$$

The leader would of course like x and y_2 to be as small as possible, while the follower has the additional desire of minimizing y_1 . As long as $x > 2$,

$$P(x) = \{(y_1, y_2) : y_1 = 4 + x, y_2 = 0\}$$

At $x = 2$, though, we have

$$P(2) = \{(y_1, y_2) : y_1 + y_2 = 6, \mathbf{y} \geq \mathbf{0}\}$$

Now, given the sequence $x^k \rightarrow \bar{x} = 2$, $y_1^k \rightarrow 4 + \bar{x}$, $y_2^k \rightarrow 0$, and the point $\bar{\mathbf{y}} = (0, 6)$, we note that $\bar{\mathbf{y}} \in P(\bar{x})$ but that there does not exist an M such that, for $k \geq M$, $\mathbf{y}^k \rightarrow \bar{\mathbf{y}}$. Thus $P(x)$ is not open at $x = 2$, although it is closed for all $x \in [2, 4]$.

Proposition 8.1.1 If $P(\mathbf{x})$ is not single-valued for all permissible \mathbf{x} , the leader may not achieve his minimum objective.

To deal with this situation, three possibilities present themselves. The first would require replacing the ‘min’ with ‘inf’ in (8.3a) and define ε -optimal solutions. This would work for Example 8.1.1 as currently formulated but a slight change in the follower’s objective function to $f = 2y_1 + 2y_2$ would reintroduce the multivalued condition. The second approach argues for a conservative strategy that redefines problem (8.3) as

$$\begin{aligned} & \min_{\mathbf{x}} \max_{\mathbf{y}^*} F(\mathbf{x}, \mathbf{y}^*) \\ \text{subject to } & \mathbf{G}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{y}^* \in P(\mathbf{x}) \triangleq \left\{ \mathbf{y}^* \in S(\mathbf{x}) \mid \begin{array}{l} f(\mathbf{x}, \mathbf{y}^*) = \min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \\ \text{subject to } \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \end{array} \right\} \end{aligned} \quad (8.4)$$

Note that for the sake of simplicity, the constraint in (8.2) is replaced with (8.4) because constraint $\mathbf{G}(\mathbf{x}, \mathbf{y}^*) \leq \mathbf{0}$ may not hold for the follower’s actual choice of \mathbf{y}^* in $P(\mathbf{x})$ in light of the min-max solution.

If $P(\mathbf{x})$ is single-valued, however, we have:

Proposition 8.1.2 In general, if all the functions in (8.3a)–(8.3d) are twice continuously differentiable and all the solutions to the subproblem (8.3c,d) are unique for \mathbf{x} feasible, then the inducible region, IR , is continuous.

The basis for the proof can be found in Hogan [H7] (Corollary 8.1); the same result was established by Bard [B5] using duality arguments. For the linear case of the Stackelberg game we have the following result.

Proposition 8.1.3 The rational reaction set, $P(\mathbf{x})$, is closed for the case where all the functions in (8.3) are linear.

The proof follows from the observation that the subproblem (8.3c,d) is a right-hand-side perturbed linear program and that the accompanying optimal-value function $w(\mathbf{x}) = \min\{f(\mathbf{x}, \mathbf{y}) : \mathbf{y} \in S(\mathbf{x})\}$ is continuous, where $f(\mathbf{x}, \mathbf{y}) = \mathbf{c}_2 \mathbf{x} + \mathbf{d}_2 \mathbf{y}$ and $S(\mathbf{x}) = \{\mathbf{y} : \mathbf{B}_2 \mathbf{y} \leq \mathbf{b}_2 - \mathbf{A}_2 \mathbf{x}, \mathbf{y} \geq \mathbf{0}\}$. In fact, $w(\mathbf{x})$ is piecewise linear and convex. Additional properties of the linear BLPP are discussed in Section 5.2.

The next example shows that closeness of the rational reaction set for even the pure linear case does not guarantee that it is always single-valued. As a consequence, complications may still be present.

Example 8.1.2

$$\begin{aligned} \min_{\mathbf{x} \geq 0} F(\mathbf{x}, \mathbf{y}) &= -x + 10y_1 - y_2 \\ \text{subject to } \min_{\mathbf{y} \geq 0} f(\mathbf{x}, \mathbf{y}) &= -y_1 - y_2 \\ \text{subject to } x - y_1 &\leq 1 \\ x + y_2 &\leq 1 \\ y_1 + y_2 &\leq 1 \end{aligned}$$

Here, $P(\mathbf{x})$ is multivalued for all $x \neq 1$ but the leader can achieve his minimum cost ($F^* = 1$) by setting $x = 1$ which, in turn, forces the follower to play $\mathbf{y} = (0, 0)$. For $0 \leq x < 1$, $F^* = 1$ is realized only when the follower cooperates and picks the largest possible value for y_2 . Now, if $F = -x + 10y_1 - 2y_2$, the leader is faced with an ambiguous situation for all choices but one. Only at $x = 1$ is the follower's response, $\mathbf{y} = (0, 0)$, unique. Notice, however, that the point $\mathbf{x} = 0$, $\mathbf{y} = (0, 1)$ is most preferred by the leader giving $F = -2$, but may not be realized despite the fact that it is in the inducible region; that is, the follower might very well pick $(1, 0)$ giving $F = 10$.

With the objective function $F = -x + 10y_1 - 2y_2$, Example 8.1.2 suggests that without some incentive, the follower has no reason to select the point $\mathbf{y} = (0, 1)$ which would be best for the leader. The third option then for dealing with a multivalued $P(\mathbf{x})$ is to assume some level of cooperation among the players and rewrite the leaders problem (8.3a,b) as

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{y}^*} F(\mathbf{x}, \mathbf{y}^*) \\ \text{subject to } \mathbf{G}(\mathbf{x}, \mathbf{y}^*) \leq 0 \\ \mathbf{y}^* \in P(\mathbf{x}) \triangleq \left\{ \mathbf{y}^* \in S(\mathbf{x}) \mid \begin{array}{l} f(\mathbf{x}, \mathbf{y}^*) = \min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \\ \text{subject to } g(\mathbf{x}, \mathbf{y}) \leq 0 \end{array} \right\} \end{aligned} \tag{8.5}$$

The difficulty with this formulation though is that it violates the basic assumption of noncooperation. If the players were allowed to cooperate, then the preferable strategy would be to seek a Pareto-optimal solution. It is well known that the Stackelberg strategy is not necessarily Pareto-optimal.

In a few instances, however, it may be possible to justify (8.5). If limited cooperation is permitted ε -optimal solutions might be appropriate. Also, when the Stackelberg problem results from, say, a reformulation of a bilinear programming problem it makes sense to give the leader control of the rational reaction set because he is really the only decision maker.

8.1.1 Independence of Irrelevant Constraints

In this section, we describe a curious property of bilevel programs that is taken for granted when dealing with standard optimization problems. In particular, for single-level programs, an optimal solution remains optimal when an inactive (i.e., irrelevant) constraint is added to the formulation. This is called independence of irrelevant constraints (IIC). Macal and Hurter [M1] have shown that BLPs do not, in general, possess this property. Their results have important implications when it comes to formulating and interpreting solutions. It would be inappropriate, for example, to exclude nonnegativity restrictions on, say, some of the \mathbf{y} variables, solve the corresponding BLPP, and conclude that the optimal solution had been obtained if all excluded variables were nonnegative.

For exposition purposes, we now make the notation explicit with respect to the lower-level constraints. Denote problem (8.3a)–(8.3d) by $\text{BLP}(\mathbf{g})$ and let $(\mathbf{x}^*, \mathbf{y}^*)$ be its solution with optimal objective function value $F_g^* = F(\mathbf{x}^*, \mathbf{y}^*)$. Define the sets $\Delta_g = \{(\mathbf{x}, \mathbf{y}) : \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}, \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}\}$ and $\Delta_s = \{(\mathbf{x}, \mathbf{y}) : \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}, s(\mathbf{x}, \mathbf{y}) \leq 0\}$ for some arbitrary function $s : R^n \times R^m \rightarrow R^1$. In what follows, the notation $\mathbf{g} \cap s$ refers to the intersection of the sets Δ_g and Δ_s ; that is $\Delta_g \cap \Delta_s$.

Definition 8.1.1 A bilevel program $\text{BLP}(\mathbf{g})$ is independent of irrelevant constraints (IIC) if its solution $(\mathbf{x}^*, \mathbf{y}^*)$ is also a solution to the following augmented bilevel program $\text{BLP}(\mathbf{g} \cap s)$ for every set Δ_s that contains $(\mathbf{x}^*, \mathbf{y}^*)$;

$$\begin{aligned} F_{gs}^* &= \min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}) \\ \text{subject to } &\mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\ &\min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \\ \text{subject to } &\mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\ &s(\mathbf{x}, \mathbf{y}) \leq 0 \end{aligned} \quad \text{BLP}(\mathbf{g} \cap s)$$

IIC is an extremely desirable property for mathematical programs to have. Prior to solving a problem, it allows one to delete constraints that are suspected of not being active at the solution. Because almost all bilevel programming algorithms are exponential in the number of lower-level constraints, the removal of irrelevant constraints at this level could greatly improve their efficiency.

The IIC property can be defined analogously for the single-level program $\text{SP}(\mathbf{g})$ by removing the follower's objective function (8.3c) and adding the constraint $s(\mathbf{x}, \mathbf{y}) \leq 0$. (Recall that $\text{SP}(\mathbf{g})$ was termed the high point problem in Section 5.3.1.) Let $(\mathbf{x}^*, \mathbf{y}^*) \in \Delta_g$ be a solution to $\text{SP}(\mathbf{g})$ and denote by H_g^* the value of F at $(\mathbf{x}^*, \mathbf{y}^*)$. Suppose $s(\mathbf{x}^*, \mathbf{y}^*) \leq 0$ for the constraint s . The augmented problem $\text{SP}(\mathbf{g} \cap s)$ is as follows:

$$\begin{aligned}
 H_{gs}^* = \min_{\mathbf{x}, \mathbf{y}} & F(\mathbf{x}, \mathbf{y}) \\
 \text{subject to } & G(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\
 & g(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\
 & s(\mathbf{x}, \mathbf{y}) \leq 0
 \end{aligned} \tag{SP(g ∩ s)}$$

Analogously to the BLP case, a single-level program $SP(g)$ is independent of irrelevant constraints if its solution is also a solution to the problem $SP(g \cap s)$ for every set Δ_s , that contains $(\mathbf{x}^*, \mathbf{y}^*)$. Every single-level program $SP(g)$ has this IIC property by the following argument. Given that $(\mathbf{x}^*, \mathbf{y}^*)$ is feasible to $SP(g \cap s)$, we have $H_{gs}^* \leq H_g^*$. Also, because $\Delta_g \cap \Delta_s \subseteq \Delta_g$, it follows that $H_g^* \geq H_{gs}^*$. The same line of reasoning, however, does not apply to bilevel programs.

Bilevel Programs and the IIC Property

To see why BLPs are not necessarily IIC, compare $BLP(g)$ and $BLP(g \cap s)$. Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a solution to $BLP(g)$, and suppose that $(\mathbf{x}^*, \mathbf{y}^*) \in \Delta_s$. Denote the optimal objective function values of $BLP(g)$ and $BLP(g \cap s)$ by F_g^* and F_{gs}^* , respectively. Then $(\mathbf{x}^*, \mathbf{y}^*) \in \Delta_g \cap \Delta_s \subseteq \Delta_g$ so that $F_{gs}^* \leq F_g^*$. But it is not necessarily true that $F_{gs}^* \geq F_g^*$ because the inducible region associated with $BLP(g \cap s)$ is not necessarily contained in the inducible region associated with $BLP(g)$. This is illustrated in the example below.

Example 8.1.3 Consider the following unconstrained BLP for $x, y \in R^1$:

$$\min_x F(x, y) = (x - 1)^2 + (y - 1)^2 \tag{8.6a}$$

$$\min_y f(x, y) = 0.5y^2 + 500y - 50xy \tag{8.6b}$$

The function F is convex in x and y , and f is convex in y for every x . Replacing (8.6b) by its first-order condition yields the equivalent problem:

$$\begin{aligned}
 \min_{x,y} & (x - 1)^2 + (y - 1)^2 \\
 \text{subject to } & y - 50x + 500 = 0
 \end{aligned} \tag{8.7}$$

where the rational reaction set $P(x) = \{y : y = 50x - 500\}$ is a one-to-one mapping. The unique solution to (8.7) is $(x^*, y^*) = (50102/5002, 4100/5002) = (10.02, 0.82)$ which yields $F(x^*, y^*) = 81.33$. Now consider the same BLP augmented by a nonnegativity constraint on y :

$$\min_x (x - 1)^2 + (y - 1)^2 \tag{8.8a}$$

$$\begin{aligned}
 \min_y & 0.5y^2 + 500y - 50xy \\
 \text{subject to } & y \geq 0
 \end{aligned} \tag{8.8b}$$

Replacing (8.8b) by its Kuhn-Tucker conditions yields the equivalent problem (see Fig. 8.1):

$$\begin{aligned} \min_{x,y} \quad & (x - 1)^2 + (y - 1)^2 \\ \text{subject to} \quad & y - 50x + 500 \geq 0 \\ & y(y - 50x + 500) = 0 \\ & y \geq 0 \end{aligned} \tag{8.9}$$

Note that (x^*, y^*) is feasible to (8.9) and $y^* > 0$ is strictly interior to the nonnegativity constraint. However, consider the point $(x, y) = (1, 0)$, which is feasible to (8.9) but is not feasible to the original unconstrained problem (8.7). Also, $F(1, 0) = 1 < F(x^*, y^*) = 81.33$. Thus taking $\Delta_g = \{(x, y) : (x, y) \in R^2\}$, $s(x, y) = y \geq 0$, and $\Delta_{g \cap s} = \{(x, y) : y \geq 0\}$ shows that problem (8.6a,b) is not IIC.

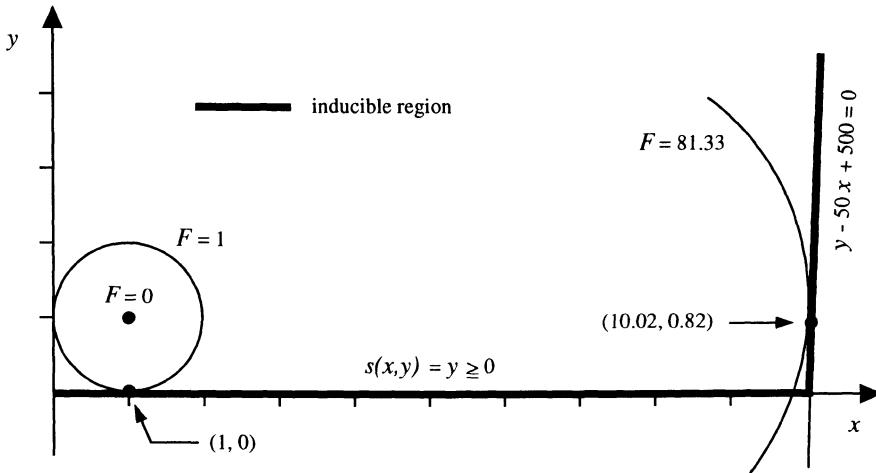


Figure 8.1 Bilevel program that is not independent of irrelevant constraints

This example illustrates that adding a constraint which includes follower variables alters the inducible region of the BLPP. The resulting inducible region is not necessarily a subset of the original inducible region, unlike the case of any single-level mathematical program. Referring to Fig. 8.1, the original inducible region for BLP(g) is $IR_g = \{(x, y) : y = 50x - 500\}$ and the inducible region for $BLP(g \cup s)$ is $IR_{gs} = \{(x, y) : y = 50x - 500 \text{ for } 50x - 500 \geq 0, y = 0 \text{ for } 50x - 500 \leq 0\}$. Thus $IR_{gs} \not\subseteq IR_g$ and by adding the inactive constraint $y \geq 0$, the objective function of the augmented BLP at the optimum actually decreases.

Interpretation of BLP Solutions

The result that bilevel programs are not IIC leads to a somewhat different interpretation of BLP solutions than solutions to standard mathematical programs. This point

is illustrated by the following situation. Suppose that a central planner in a firm seeks to minimize overall production costs. Let the lower level in the model represent a decentralized division of the firm. In minimizing costs, it might be assumed that a resource such as labor is unlimited because an ample workforce is available. Suppose now that the solution of the cost-minimizing BLP shows a labor requirement of 100. Some time later there is a strike and labor availability is reduced to 101. If unit labor costs have not changed, it is natural to conclude that since the previous solution of 100 is within the new limit of 101, the original solution remains optimal. Surprisingly, this may be an incorrect conclusion. The reduction in labor availability might alert the central planner to the fact that costs could be reduced further, possibly dramatically, by using a smaller workforce. The only way to resolve this issue is to solve the BLP again with the new labor availability constraint.

Conditions for Bilevel Programs to be IIC

We now present necessary and sufficient conditions for BLPs to be IIC. The proof can be found in [M1]. The following definitions are needed for the exposition.

Definition 8.1.2 The *unconstrained level-two problem* of a bilevel program $\text{BLP}(g)$ consists of the lower-level problem (8.1) without the constraints $g(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}$. That is, the unconstrained level-two problem for a given \mathbf{x} is $\min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$.

Definition 8.1.3 A bilevel program $\text{BLP}(g)$ is *degenerate* if a solution to its associated single-level program $\text{SP}(g)$ is feasible to the unconstrained level-two problem of $\text{BLP}(g)$; that is, if $\nabla_{\mathbf{y}} f(\mathbf{x}_s^*, \mathbf{y}_s^*) = \mathbf{0}$ where $(\mathbf{x}_s^*, \mathbf{y}_s^*)$ is a solution to $\text{SP}(g)$. A *nondegenerate* BLP is one that does not have this property.

A degenerate bilevel program has the same solution whether or not any of the constraints are included in the lower-level problem. In effect, $\text{BLP}(g)$ degenerates into a single-level program consisting of the upper-level problem (8.2) and the first-order conditions associated with the lower-level objective function included as equality constraints. Therefore, a degenerate BLP is IIC.

Theorem 8.1.1 A nondegenerate bilevel program $\text{BLP}(g)$ is independent of any constraint $s(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}$ if and only if there is a solution to the associated single-level program $\text{SP}(g)$ that is in the inducible region of $\text{BLP}(g)$, under the assumption that the Kuhn-Tucker conditions for the lower-level problem (8.1) with \mathbf{x} fixed are necessary and sufficient for an optimal solution to that problem.

Theorem 8.1.1 requires that the leader's optimal solution to the single-level program coincide with the follower's optimal solution in the corresponding bilevel program for the particular value of \mathbf{x} that is optimal to the single-level program. This is an extremely strong condition which is not likely to hold in real-world applications. For this condition to be satisfied, it must happen that both players have the same solution at one point, regardless of the differences in their objective functions. The implication

is that the vast majority of BLPs encountered in practice will depend on irrelevant constraints.

From Theorem 8.1.1 we observe that the solution to the single-level program $SP(g)$, denoted by $(\mathbf{x}_s, \mathbf{y}_s)$, satisfies the following condition if and only if the bilevel program $BLP(g)$ is independent of all irrelevant constraints $s(\mathbf{x}, \mathbf{y}) \leq 0$.

Condition 8.1 Let $(\mathbf{x}_s, \mathbf{y}_s)$ be the solution of the single-level program $SP(g)$ and let $I = \{i : g_i(\mathbf{x}_s, \mathbf{y}_s) = 0\}$ be the corresponding set of binding constraints. Then there exists a $u_i \geq 0$ for all $i \in I$ such that the following conditions are satisfied:

$$\frac{\partial f}{\partial y_j}(\mathbf{x}_s, \mathbf{y}_s) + \sum_{i \in I} u_i \left(\frac{\partial g}{\partial y_j}(\mathbf{x}_s, \mathbf{y}_s) \right) = 0, \quad j = 1, \dots, m$$

BLPs satisfying this condition have the property that the solution to the associated SP is on the boundary of the feasible region defined by the constraints $\mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}$. In this case, any constraint that is added to the BLP produces a feasible region that is contained in the original feasible region defined by $\mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}$; therefore, a new constraint does not negate the optimality of a solution to the original BLP.

To test for IIC, first solve the associated single-level program, which is generally much easier to solve than the bilevel program under investigation. As discussed by Macal and Hurter, four different cases may arise at this point indicating whether or not the IIC property holds. Concerning feasibility, it is straightforward to show that a BLP has a feasible solution if and only if its associated SP has a feasible solution; therefore, a feasible solution to the associated SP implies the BLP is also feasible.

8.1.2 Preview of Algorithms

We now move on to algorithms for the general BLPP. In this regard, penalty methods have played a key role beginning in the early 1980s. Taking the lead, Aiyoshi and Shimizu [A3, S8] approximated the original two-level problem (8.3) by a series of (single-level) nonlinear programs and proved that the sequence of approximate solutions converges to an optimal solution of the Stackelberg problem.

A second approach, already discussed in Chapters 5 and 7, is based on solving the nonlinear program obtained by replacing the lower-level problem with its Kuhn-Tucker conditions. In this category, Bard (7.3) applied an active constraint strategy and replaced the lower-level problem with its stationarity conditions. Fortuny-Amat and McCarl [F7] developed a computational method by transforming the original problem into a mixed-integer program, and Edmunds and Bard [E1] developed a branch and bound algorithm based on an implicit enumeration of the complementarity conditions associated with the follower's problem (8.1) for \mathbf{x} fixed. The latter is an extension of the Bard-Moore algorithm proposed for the linear BLPP discussed in Section 5.3.2.

The effectiveness of these approaches is mainly limited to the case where the leader's objective function is linear and the follower's is convex-quadratic. Shimizu and Lu [S11] extended the Kuhn-Tucker formulation to cover situations where the upper-level problem consists of convex functions or differences of two convex functions, but where the lower-level problem is still limited to be convex and quadratic. Loridan and Morgan [L5] developed general convergence properties for related computational methods. Al-Khayyal, Horst and Pardalos [A5] addressed the case of a concave upper-level problem coupled with a linear lower-level problem. This is discussed in Section 8.4.

In a different vein, arguments common to sensitivity analysis in parametric nonlinear programming can provide information on the gradient and the directional derivative of the optimal solution $\mathbf{y}^*(\mathbf{x})$ of the lower-level problem. Using $\nabla \mathbf{y}^*(\mathbf{x})$ or the directional derivative $D\mathbf{y}^*(\mathbf{x}; \mathbf{d})$, it is possible to derive optimality conditions for (8.3). The theoretical details are discussed in [D2, O3, S2]. One of the first efforts to implement this idea was undertaken by Kolstad and Lasdon [K6]. Building on that work, Falk and Liu [F1] developed a bundle method that exploits subgradient information to compute directional derivatives. The specifics are presented in Section 8.6.

Alternatively, let us consider the optimal-value function of the lower-level problem (8.1); i.e.,

$$w(\mathbf{x}) \triangleq f(\mathbf{x}, \mathbf{y}^*) = \min_{\mathbf{y} \in S(\mathbf{x})} f(\mathbf{x}, \mathbf{y}) \quad (8.10)$$

where $\mathbf{y}^* \in P(\mathbf{x})$ is an arbitrarily chosen solution vector of the follower. Then the BLPP (8.5) is equivalent to

$$\min_{\mathbf{x}, \mathbf{z}} F(\mathbf{x}, \mathbf{z}) \quad (8.11a)$$

$$\text{subject to } \mathbf{G}(\mathbf{x}, \mathbf{z}) \leq \mathbf{0} \quad (8.11b)$$

$$\mathbf{g}(\mathbf{x}, \mathbf{z}) \leq \mathbf{0} \quad (8.11c)$$

$$f(\mathbf{x}, \mathbf{z}) - w(\mathbf{x}) = 0 \quad (8.11d)$$

$$w(\mathbf{x}) = \min_{\mathbf{y} \in S(\mathbf{x})} f(\mathbf{x}, \mathbf{y}) \quad (8.11e)$$

where $w(\mathbf{x})$ is the optimal-value function defined by (8.10). In (8.11), $\mathbf{z} \in R^m$ serves as a substitute for the solution of the lower-level problem $\mathbf{y}^* \in P(\mathbf{x})$ and is an artificial variable newly introduced as an upper-level decision variable. Note that constraint (8.11d) has been added. By viewing (\mathbf{x}, \mathbf{z}) and \mathbf{y} as the upper-level and the lower-level decision variables, respectively, problem (8.11) can be considered a special case of the parameter design problem studied by Shimizu et al. [S10] (Chapter 12).

One can exploit the above formulation in the development of algorithms to solve the original BLPP (8.3). However, conventional nonlinear programming techniques cannot be applied even if F , \mathbf{G} and \mathbf{g} possess "nice" properties such as convexity or linearity, because constraint (8.11d) is neither convex nor differentiable. Problem (8.11) is a single-level nonlinear program with a nondifferentiable equality constraint

(8.11d). Thus using nonsmooth optimization theory, optimality conditions can be derived in terms of an expression or an estimate of the generalized gradients of $w(\mathbf{x})$. This type of approach was developed by Chen and Florian [C3]. Recently, global optimization methods have been applied to problem (8.11) to obtain a global optimum of problem (8.3); e.g., see [A5, S11, T5].

Variants of the Stackelberg problem extend to hierarchical decentralized systems where the lower level consists of several semi-autonomous subsystems. Such problems, which include important applications related to general resource allocation, are highlighted in [A2, S7]. Two-level multiobjective decision problems have also been studied in [I1, I3, S8] as well as the edited volume by Anandalingam and Friesz [A7] which contains a variety of algorithms, applications and theoretical advances.

8.2 BRANCH AND BOUND ALGORITHM

Most constrained optimization problems can be analyzed from a combinatorial point of view. If it were known beforehand which inequalities were binding at the optimum, it would be a simple matter of setting them equal to zero and solving the resultant equality constrained optimization problem. When all the functions are continuously differentiable this is equivalent to solving a series of $n + m$ nonlinear equations in $n + m$ variables (n problem variables and m Lagrange multipliers). In the presence of inequality constraints, though, nonlinear programming theory tells us that the situation is a lot more complicated. In this section, we combine the ideas of implicit enumeration with standard nonlinear programming to develop an algorithm for the general bilevel programming problem:

$$\min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}) \quad (8.12a)$$

$$\text{subject to } \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \quad (8.12b)$$

$$\min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \quad (8.12c)$$

$$\text{subject to } \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \quad (8.12d)$$

We restrict our attention to BLPPs with the following properties.

- (i) f and \mathbf{g} are twice continuously differentiable in \mathbf{y} for all $\mathbf{y} \in S(\mathbf{x}) = \{\mathbf{y} : \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}\}$;
- (ii) f is strictly convex in \mathbf{y} for all $\mathbf{y} \in S(\mathbf{x})$;
- (iii) $S(\mathbf{x})$ is a compact convex set; and
- (iv) F is continuous and convex in \mathbf{x} and \mathbf{y} .

Under assumptions (i)~(iii), the rational reaction set $P(\mathbf{x}) = \{\mathbf{y} : \mathbf{y} \in \arg \min[f(\mathbf{x}, \hat{\mathbf{y}})] : \hat{\mathbf{y}} \in S(\mathbf{x})\}$ is a continuous point-to-point map. Hence, assumption (iv) implies

that $F(\mathbf{x}, P(\mathbf{x}))$ is continuous. The additional fact that $P(\mathbf{x})$ is closed under the first three assumptions implies that the inducible region $IR = \{(\mathbf{x}, \mathbf{y}) : \mathbf{x} \text{ feasible}, \mathbf{y} \in P(\mathbf{x})\}$ is compact. Thus the leader minimizes a continuous function over a compact set. It is well known that the solution to such a problem is guaranteed to exist.

Rather than addressing (8.12a)–(8.12d) directly, we create once again an alternative representation by replacing the follower's subproblem (8.12c,d) with his Kuhn-Tucker conditions and append the resultant system to the leader's problem. This gives rise to a traditional nonlinear program of the form:

$$\min_{\mathbf{x}, \mathbf{y}, \mathbf{u}} F(\mathbf{x}, \mathbf{y}) \quad (8.13a)$$

$$\text{subject to } \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \quad (8.13b)$$

$$\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) + \mathbf{u} \nabla_{\mathbf{y}} \mathbf{g}(\mathbf{x}, \mathbf{y}) = \mathbf{0} \quad (8.13c)$$

$$\mathbf{u} \mathbf{g}(\mathbf{x}, \mathbf{y}) = \mathbf{0} \quad (8.13d)$$

$$\mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \quad (8.13e)$$

$$\mathbf{u} \geq \mathbf{0} \quad (8.13f)$$

where $\mathbf{u} \in R^q$ is a (row) vector of Kuhn-Tucker multipliers for the follower's subproblem for \mathbf{x} fixed. As an aside we note that by virtue of the fact that (8.13a)–(8.13f) is a single-level program it is always independent of irrelevant constraints, whether or not the original BLP is IIC. Adding another constraint to the above formulation is equivalent to adding a constraint to the upper-level problem (8.2). This simply reduces the original inducible region to a subset of itself.

If the follower's objective function f is strictly convex and $S(\mathbf{x})$ is a compact convex set for all allowable choices of the leader variables, then $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{u}^*)$ solves (8.13) if and only if $(\mathbf{x}^*, \mathbf{y}^*)$ solves (8.12a)–(8.12d) (see [S13]). Of course, this equivalent single-level program is nonconvex so local minima may exist. Even if all the functions were linear, the complementarity term (8.13d) remains quadratic and presents enormous difficulties for virtually all nonlinear solvers. Most of the algorithms discussed in Chapter 5 address this issue by first relaxing the complementarity requirements in (8.13) and then trying to reintroduce them through some form of enumeration. Such an approach was taken by Edmunds and Bard [E1] who extended the Bard-Moore algorithm (Section 5.3.2) for the linear BLPP by allowing for a wider range of functional forms and permitting greater flexibility in the generation of the search tree. Their approach is highlighted below. The general idea is to relax the complementary slackness conditions and solve the corresponding optimization problem. If the solution violates one or more of these constraints, a combination of depth-first and breadth-first branch and bound is used to implicitly enumerate all possibilities.

In the depth-first search, one of the violated constraints (say, the i th) is selected and two subproblems are set up (see Section 3.2). The first corresponds to the case where $u_i = 0$ and the second to the case where $g_i = 0$. In both instances, the i th complementarity constraint will be satisfied. A solution to one of the two subproblems

is obtained and the procedure is repeated until all of the constraints in (8.13d) are satisfied. The algorithm terminates when all subproblems have either been solved or are known to have solutions which are either suboptimal or infeasible.

In the breadth-first search, one or more of the violated complementary slackness conditions is selected and two or more subproblems are set up and solved. In each instance, all of the selected complementary slackness conditions are satisfied by the addition of various combinations of $u_i = 0$ and $g_i = 0$ constraints. If three complementary slackness conditions are selected, for example, eight subproblems are set up and solved. At the next expansion step (major iteration), one of these 8 subproblems, which still has violated complementary slackness conditions, is selected and the process is repeated.

The branch and bound procedure may be viewed in terms of a rooted binary tree. The nodes correspond to subproblems in which various combinations of $u_i = 0$ and $g_i = 0$ constraints are enforced. Arcs connect the nodes and define relationships among the subproblems. More formally, let us write (8.13d) equivalently as q individual equations, $u_i g_i = 0$, and let $W = \{1, 2, \dots, q\}$ be the corresponding index set. Denote the incumbent lower and upper bound on the leader's objective function as \underline{F} and \bar{F} , respectively. Conceptually, each path vector, P^k , at node k in the search tree corresponds to an assignment of either $u_i = 0$ or $g_i = 0$ for $i \in W^k \subseteq W$. If $i \in W^k$ and constraint $u_i = 0$ is included in subproblem k , then the element $+i$ appears in the path vector P^k . If $i \in W^k$ and constraint $g_i = 0$ is included in the subproblem then the element $-i$ appears in P^k . The path vector is a q -dimensional column vector and is initially set to $[0]$.

In the actual implementation, the authors start with a breadth-first search and switch over to a depth-first search when the number of live nodes becomes excessive. A flowchart of the algorithm is displayed in Fig. 8.2. The notation in Table 8.1 is employed in the description of the algorithm.

Algorithm

- Step 1 (Initialization) Assign values to n_e and n_{max} . Set $k = 0$, $P = [0, 0]$, $S = [0, 0]$, $V = [0]$ and $L = \emptyset$.
- Step 2 (Upper bound) Formulate and solve problem (8.13) without complementarity constraint (8.13d). Label the solution $(\mathbf{x}^0, \mathbf{y}^0)$. If $(\mathbf{x}^0, \mathbf{y}^0) \in IR$ stop; otherwise, fix $\mathbf{x} = \mathbf{x}^0$, solve follower's subproblem (8.12c,d) in \mathbf{y} . Label this solution $(\mathbf{x}^*, \mathbf{y}^*)$ and set upper bound $\bar{F} = F(\mathbf{x}^*, \mathbf{y}^*)$.
- Step 3 (Lower bound) Assign vector $S[-, 1] = [\mathbf{x}^0, \mathbf{y}^0, \mathbf{G}^0, \mathbf{g}^0, F(\mathbf{x}^0, \mathbf{y}^0)]$; set $L = \{1\}$, $j' = 1$ and go to Step 5.
- Step 4 (Update lower bound) Find the column index j' such that $j' \in L$ and $S[n_T, j'] \leq S[n_T, j]$ for all $j \in L$. Set lower bound to $\underline{F} = S[n_T, j']$.

Table 8.1 Notation for branch and bound algorithm

Parameter	Definition
n_e	number of violated complementary slackness conditions to satisfy in the next expansion step
n_{max}	maximum number of subproblems to retain (maximum number of live nodes)
n_v	total number of original variables in (8.12a)–(8.12d); $n_v = n + m$
n_f	total number of constraints in (8.12a)–(8.12d); $n_f = p + q$
n_T	$n_v + n_f + 1$
q	number of follower inequality constraints, including variable bounds
L	set of live nodes; elements in L reference columns in arrays P and S
n_L	number of elements in the set L
P	array of dimension $q \times n_{max}$ holding path vectors of live nodes
S	array of dimension $n_T \times n_{max}$ holding solutions of subproblems at live nodes (objective function values are stored in row n_T of each column)
V	vector of length n_e holding indices of the most violated complementary slackness conditions

Step 5 (Objective tolerance) If $[\bar{F} - \underline{F}]$ is within tolerance, terminate with an ε -optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$.

Step 6 (Expansion check) If $2^{n_e} > n_L$, go to Step 12.

Step 7 (Expansion: first subproblem) Choose the n_e indices from the set $i \in W$ corresponding to the n_e largest values of the expression $|u_i^{j'} g_i^{j'}|$, and place them in the array V . Find r , where r is the smallest value of index i such that $P[i, j'] = 0$. Append the V array obtained above to the following partition of the P array:

$$[P(1, j'), P(2, j'), \dots, P((r-1), j')]^T$$

Then put $k \leftarrow k + 1$ and attempt to solve the subproblem at node k defined by the path vector $P[-, j']$.

Step 8 (Update bound and fathom) If the subproblem is infeasible, set $L \leftarrow L \setminus \{j'\}$ and go to Step 10; otherwise, label the solution $(\mathbf{x}^k, \mathbf{y}^k, \mathbf{u}^k)$. If $F(\mathbf{x}^k, \mathbf{y}^k) \leq \bar{F}$ and $u_i^k g_i(\mathbf{x}^k, \mathbf{y}^k) = 0$ for all $i \in W$, then put $(\mathbf{x}^*, \mathbf{y}^*) \leftarrow (\mathbf{x}^k, \mathbf{y}^k)$ and $\bar{F} = F(\mathbf{x}^*, \mathbf{y}^*)$. If $F(\mathbf{x}^k, \mathbf{y}^k) > \bar{F}$, set $L \leftarrow L \setminus \{j'\}$ and go to Step 10.

Step 9 (Objective tolerance) If $[\bar{F} - \underline{F}]$ is within tolerance, terminate with an ε -optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$.

Step 10 (Backtrack) If $V \leqq [0]$ then go to Step 4.

Step 11 (Next subproblem) Find an index $j \leq n_{max}$ such that $j \notin L$ and $j \neq j'$. Set $P[-, j] = P[-, j']$, $L \leftarrow L \cup \{j\}$ and $j' = j$. Find s , where s is the largest index i such that $V_i > 0$. Put $V_s \leftarrow -V_s$ and $V_i \leftarrow |V_i|$ for all $i > s$. Put $k \leftarrow k + 1$ and append this V array to a partition of the path array: $[P(1, j'), P(2, j'), \dots, P((r - 1), j')]^T$. Solve subproblem defined by path array $P[-, j']$ and go to Step 8.

Step 12 (Depth-first search) Pass path vector $P[-, j']$ to depth-first branch and bound subroutine and implicitly enumerate all subproblems in the tree rooted at node $P[-, j']$. Go to Step 4.

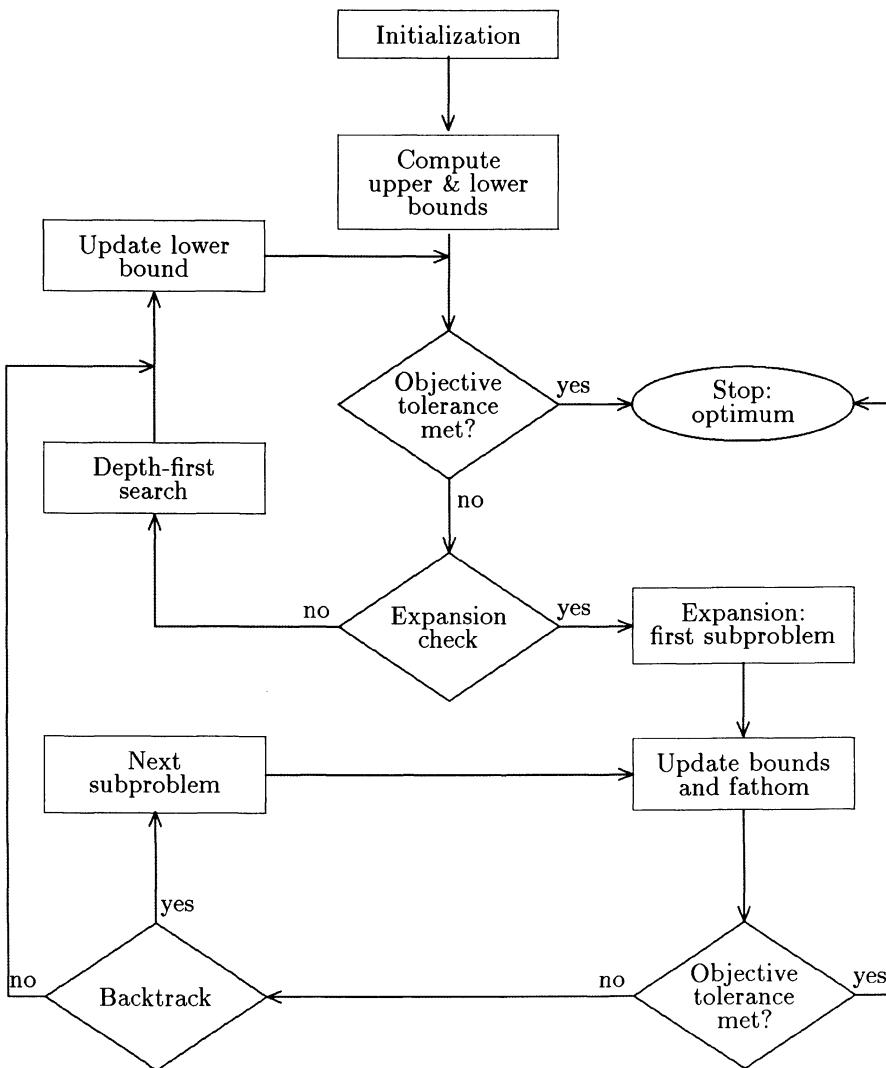
Parameters are initialized at Step 1, the counter k is set to zero, and the arrays P , S and V are filled with zeros. At Step 2, an upper bound on the objective function (8.13a) is obtained by finding a point in the inducible region. First we solve (8.13) without the complementary slackness conditions to get $(\mathbf{x}^0, \mathbf{y}^0)$. If this point is in IR we are finished. If not, we fix the leader variables at \mathbf{x}^0 and solve the follower's problem. This yields an upper bound on the global optimum.

The solution to a relaxation of (8.13) is given by $(\mathbf{x}^0, \mathbf{y}^0, \mathbf{u}^0)$. Hence $F(\mathbf{x}^0, \mathbf{y}^0)$ is a lower bound on the global minimum as indicated in Step 3. The solution vector is loaded into column 1 of the array S (in general, $S[-, j]$ refers to the j th column of S). The variables are loaded into the first n_v rows and the function values are loaded into the next n_f rows of this array, with the objective function value in row $n_v + n_f + 1$.

The set L identifies columns in the P and S arrays corresponding to live nodes. Thus $L = \{1\}$ implies that the path in column 1 of the P array corresponds to a live node with solution given in column 1 of the S array. Initially, the P array is filled with zeros, indicating that the path vector in column 1 is the zero vector and consequently that it is the root node of the search tree.

The live node with the best (smallest) objective function value is selected and the lower bound is updated in Step 4. At Step 5, the objective function value of the solution to (8.13) is known to lie between $[\bar{F} - \underline{F}]$. If this range is less than a prescribed tolerance, ε , the search is terminated. The number of live nodes that will exist after the next expansion is computed in Step 6. If this number exceeds n_{max} , control is passed to Step 12 where a depth-first search begins. Implicit enumeration then reduces the number of live nodes by one, so an expansion step may be possible at the next iteration.

At Step 7, the first of 2^{n_e} new subproblems is created by expanding the tree n_e levels starting from the best live node j' . The n_e indices associated with the largest complementary slackness violations are stored in V . The path vector of the new subproblem is generated by appending the array V to the path vector accompanying node j' .

**Figure 8.2** Hybrid branch and bound algorithm

For example, let the path vector in column 1 of the P array be $P[-, 1] = [-2, 1, 0, 0, 0]$, let $n_{max} = 8$ and expand by $n_e = 2$ levels to form the next subproblem. Assume the set of live nodes is $L = \{1\}$ and the index with minimum objective function value is $j' = 1$. Further assume that complementary slackness conditions three and four are

the most violated, so the array $V = [4, 3]$. The purpose of finding the row index r is to identify the smallest nonzero element in the path vector. In this example, $r = 3$. The array V is appended to the path vector in column j' of P at the index r to obtain the path vector associated with the next subproblem. This new path vector is stored in column j' of the P array and is equal to $P[-, 1] = [-2, 1, 4, 3, 0]$. Thus constraints $g_2 = 0$, $u_1 = 0$, $u_4 = 0$ and $u_3 = 0$ are to be included in the next subproblem.

After the subproblem in Step 7 is solved, a number of checks are made at Step 8. If the problem is infeasible or its solution is above \bar{F} , its index is removed from L . Alternatively, if the solution is feasible and satisfies all complementary slackness conditions, a point in IR has been found and the index is likewise removed from the set of live nodes. Finally, \bar{F} is updated and a termination check is performed at Step 9.

At Step 10, a check for backtracking is made to determine whether or not all subproblems generated by the expansion step have been examined. If the V array consists of nonpositive entries then no active candidates remain and another live node is selected for expansion. Subsequently, control is passed to Step 4. The algorithm backtracks at Step 11 and solves the next subproblem generated by the expansion step. Bookkeeping operations require changing the sign of the rightmost positive entry, s , in the vector V and making all entries to the right of s positive.

In our example, $L = \{1\}$, $j' = 1$, $V = [4, 3]$ and $P[-, 1] = [-2, 1, 4, 3, 0]$ so we choose index $j = 2$ to store the next path vector and solution. We set $L = \{1, 2\}$, $P[-, 2] = P[-, 1]$ and $s = 2$. The sign of element V_2 is switched and the new array is given by $V = [4, -3]$. The latter is then appended to the partition of the path array defined by the index $r = 3$ to obtain a new path vector $P[-, 2] = [-2, 1, 4, -3, 0]$.

If storage requirements for the solution vectors in the S array exceed computer memory, the algorithm reverts to a depth-first search at Step 12. The live node with minimum objective function value is used for branching. After the depth-first search is completed, control is passed to Step 4 where the lower bound is updated.

Computational Experience

Edmunds and Bard coded their algorithm in vs FORTRAN and ran a series of test problems on an IBM 3081-D mainframe. A successive quadratic programming (SQP) code was used to solve the subproblems at the nodes in the tree. After some preliminary testing, n_e was set at 2 and n_{max} at 8 as a compromise between storage requirements and the prospect of early fathoming. The majority of the experimental results were aimed at evaluating the effectiveness of generating an initial upper bound at Step 2, and comparing the hybrid approach with a pure depth-first search.

The test problems themselves were randomly generated to meet conditions required for global convergence. In particular, the objective functions variously comprised

convex quadratic, separable convex exponential, and separable convex quadratic functions plus linear terms, while the constraints were all linear. The second-level objective function took the following form:

$$f(\mathbf{x}, \mathbf{y}) = \mathbf{d}_2 \mathbf{y} + \mathbf{x}^T \mathbf{Q}_1 \mathbf{y} + \frac{1}{2} \mathbf{y}^T \mathbf{Q}_2 \mathbf{y} + \sum_{i=1}^m f_i(y_i)$$

where \mathbf{Q}_1 is an $n \times m$ matrix, \mathbf{Q}_2 is an $m \times m$ symmetric positive semidefinite matrix, and $f_i(y_i)$ is convex. Each problem set consisted of 5 instances, with dimensions given in Table 8.2.

Table 8.2 Test problem dimensions

Problem set	No. leader variables	No. follower variables	No. inequality constraints	No. variables bounds
1	5	5	5	10
2	10	5	10	15
3	10	5	5	15
1a	5	5	5	0
2a	10	5	10	0
3a	10	5	5	0
4	10	10	10	20
5*	10	10	10	20

*Only quadratic terms in the objective functions

Two techniques were examined in an effort to obtain a tight initial upper bound on the BLPP. The first, and the one finally chosen, is described in the discussion of Step 2 of the algorithm. The second involved an attempt to solve problem (8.13) with the complementary slackness conditions (8.13d) included. Problem sets 1, 2 and 3 were solved for each of these cases. The results are summarized in Table 8.3, where

- Total nodes = number of subproblems solved, averaged over problems in the set
- Optimal node = number of subproblems solved before optimum was found, averaged over problems in the set
- Function calls = number of times each of the functions in the subproblem is evaluated, averaged over problems in the set
- Derivative calls = number of times partial derivatives in the subproblem are evaluated, averaged over problems in the set

The data show that the first upper bounding technique yields the minimum average run time. The failure of the second approach to provide a tight upper bound can be attributed to the fact that in most of the test problems, no feasible point was found. As a consequence, the upper bound was fixed at $+\infty$ so the time spent in this

Table 8.3 Comparison of upper bound techniques

First technique (fix $\mathbf{x} = \mathbf{x}^0$, then solve follower's problem)					
Problem set	CPU (sec)	Total nodes	Optimal node	Function calls	Derivative calls
1	15.2	15	13	501	3660
2	45.7	15	13	475	5537
3	38.6	1	8	485	4680
<i>mean</i>	33.2	14	11	487	4626
Second technique (solve (8.13) with (8.13d))					
Problem set	CPU (sec)	Total nodes	Optimal node	Function calls	Derivative calls
1	79.2	19	16	999	5635
2	95.9	15	14	531	5768
3	84.0	12	9	751	5268
<i>mean</i>	86.4	15	13	760	5557
No upper bound technique used ($\bar{F} = \infty$)					
Problem set	CPU (sec)	Total nodes	Optimal node	Function calls	Derivative calls
1	37.3	19	17	1225	6635
2	42.7	14	13	393	4977
3	59.3	12	9	655	5064
<i>mean</i>	46.4	15	13	758	5559

unsuccessful search was wasted. It should be mentioned that this difficulty persisted when GRG2 (see [L1]) was substituted for SQP; however, more favorable results were obtained when lower bounds on the follower's variables were removed. In problem sets 1a, 2a and 3a, the follower variables are unrestricted. The second upper bounding technique found the global minimum of (8.13) in 14 out of 15 cases.

A pure depth-first search was also compared with the hybrid algorithm where up to 2^{n_c} subproblems are set up and solved during branching. Problem sets 4 and 5 were solved with both techniques. The results are summarized in Table 8.4 and demonstrate the superiority of the latter. On average, the depth-first approach required 33.9% more CPU time than did the hybrid algorithm.

The conclusions drawn from these experiments were that a combination of breadth-first and depth-first search works best and that the effectiveness of obtaining an initial upper bound is dependent on whether or not follower variable bounds are included in the original problem. Modest size nonlinear BLPPs with 10 leader variables, 10 follower variables, 10 inequality constraints, and bounded variables were solved in less than 400 CPU seconds. Repeated testing suggested that global optimality, though not guaranteed, was achieved in each case. From a practitioners point of view, these

Table 8.4 Comparison of depth-first and hybrid algorithms

Depth-first search					
Problem set	CPU (sec)	Total nodes	Optimal node	Function calls	Derivative calls
4	514.3	30	15	2091	31,970
5	526.3	27	25	1374	26,600
<i>mean</i>	520.3	29	20	1733	29,285
Hybrid search ($n_e = 2$, $n_{max} = 8$)					
Problem set	CPU (sec)	Total nodes	Optimal node	Function calls	Derivative calls
4	399.1	26	23	1373	26,690
5	378.3	20	18	1153	20,870
<i>mean</i>	388.7	23	21	1263	23,780

problems are quite small so there still exists a need to solve much larger instances. It should be noted that the performance of any algorithm designed for this purpose strongly depends on the efficiency of the nonlinear solver used in Step 2.

8.3 DOUBLE PENALTY FUNCTION METHOD

In this section, we present a solution method for the general BLPP based on the use of penalty functions, an approach commonly employed in nonlinear programming. The idea is to transform the two-level problem into a sequence of single-level unconstrained problems [A3, I2]. Specifically, we wish to solve

$$\begin{aligned} & \min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) \\ & \text{subject to } \mathbf{G}(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) \leq \mathbf{0} \\ & \quad f(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) = \min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \\ & \quad \text{subject to } \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \end{aligned} \tag{8.14}$$

where $\mathbf{y}^*(\mathbf{x})$, once again, denotes an optimal solution to lower-level problem for a given \mathbf{x} .

Assumption 8.3.1 For any fixed \mathbf{x} , the optimal solution $\mathbf{y}^*(\mathbf{x})$ is uniquely determined.

In [S8], a penalty function approach was proposed in which the follower's problem in (8.14) was replaced by an unconstrained problem with an augmented objective

function; i.e.,

$$p(\mathbf{x}, \mathbf{y}; r) = f(\mathbf{x}, \mathbf{y}) + r\phi(g(\mathbf{x}, \mathbf{y})), \quad r > 0 \quad (8.15)$$

where ϕ is a continuous interior penalty function on the negative domain of R^m , r is a parameter, and

$$\begin{aligned} \phi(g(\mathbf{x}, \mathbf{y})) &> 0 && \text{if } \mathbf{y} \in \text{int}S(\mathbf{x}) \\ \phi(g(\mathbf{x}, \mathbf{y})) &\rightarrow +\infty && \text{if } \mathbf{y} \rightarrow \text{bd}S(\mathbf{x}) \end{aligned} \quad (8.16)$$

Here, $S(\mathbf{x})$ is $\{\mathbf{y} : g(\mathbf{x}, \mathbf{y}) \leq 0\}$. Then the original problem (8.14) was transformed into

$$\begin{aligned} &\min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}^*(\mathbf{x}; r)) \\ \text{subject to } &G(\mathbf{x}, \mathbf{y}^*(\mathbf{x}; r)) \leqq 0 \\ &p(\mathbf{x}, \mathbf{y}^*(\mathbf{x}; r), r) = \min_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}; r) \end{aligned} \quad (8.17)$$

In [S8], it is proved that the sequence $\{(\mathbf{x}^k, \mathbf{y}^*(\mathbf{x}^k; r^k))\}$ of optimal solutions to problem (8.17), in response to a parameter sequence of r converging to zero, converges to the solution of problem (8.14). However, in solving (8.17) for a fixed value of the parameter r , it is necessary to solve the follower's unconstrained problem in (8.17) every time the trial value of the leader's variables are updated. This greatly slows convergence. A more efficient "double penalty" function approach is developed below.

In particular, the same augmented objective function (8.15) and the penalty function (8.16) are used for the follower's problem, but now the latter is replaced by its stationarity condition $\nabla_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}; r) = \mathbf{0}$. Thus the original two-level problem is approximated by a single-level problem

$$\begin{aligned} &\min_{\mathbf{x}, \mathbf{y}} F(\mathbf{x}, \mathbf{y}) \\ \text{subject to } &G(\mathbf{x}, \mathbf{y}) \leqq 0 \\ &\nabla_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}; r) = \mathbf{0} \\ &g(\mathbf{x}, \mathbf{y}) < 0 \end{aligned} \quad (8.18)$$

where the last constraint in (8.18) prescribes the domain of the function p .

The next step is to solve (8.18) for a given penalty parameter r . In so doing, we adopt the interior-exterior mixed penalty function approach, and introduce a new overall (leader's) augmented objective function

$$\begin{aligned} Q(\mathbf{x}, \mathbf{y}; t, s, r) &= F(\mathbf{x}, \mathbf{y}) + t\Phi(G(\mathbf{x}, \mathbf{y})) \\ &\quad + r\phi(g(\mathbf{x}, \mathbf{y})) + s\Psi(\|\nabla_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}; r)\|) \end{aligned} \quad (8.19)$$

Here Φ is a continuous interior penalty (barrier) function on the negative domain of R^p , and

$$\begin{aligned} \Phi(G(\mathbf{x}, \mathbf{y})) &> 0 && \text{if } (\mathbf{x}, \mathbf{y}) \in \text{int}\{(\mathbf{x}, \mathbf{y}) : G(\mathbf{x}, \mathbf{y}) \leqq 0\} \\ \Phi(G(\mathbf{x}, \mathbf{y})) &\rightarrow +\infty && \text{if } (\mathbf{x}, \mathbf{y}) \rightarrow \text{bd}\{(\mathbf{x}, \mathbf{y}) : G(\mathbf{x}, \mathbf{y}) \leqq 0\} \end{aligned} \quad (8.20)$$

Ψ is a continuous and monotone increasing exterior penalty function on $[0, +\infty)$ such that

$$\begin{aligned}\Psi(\|\nabla_{\mathbf{y}} p(\mathbf{x}, \mathbf{y})\|) &= 0 \quad \text{if } (\mathbf{x}, \mathbf{y}) \in \{\bigcup_{\mathbf{x} \in R^n} \{\mathbf{x} \times \text{int}S(\mathbf{x})\} : \nabla_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}; r) = \mathbf{0}\} \\ \Psi(\|\nabla_{\mathbf{y}} p(\mathbf{x}, \mathbf{y})\|) &> 0 \quad \text{if } (\mathbf{x}, \mathbf{y}) \in \{\bigcup_{\mathbf{x} \in R^n} \{\mathbf{x} \times \text{int}S(\mathbf{x})\} : \nabla_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}; r) \neq \mathbf{0}\}\end{aligned}\tag{8.21}$$

Hence, we can approximate the original problem (8.14) by the following single-level unconstrained problem in which the overall augmented objective function (8.19) is minimized with respect to \mathbf{x} and \mathbf{y} jointly; that is,

$$\min_{\mathbf{x}, \mathbf{y}} Q(\mathbf{x}, \mathbf{y}; t, s, r).\tag{8.22}$$

To solve problem (8.18) in the traditional manner using the function Q , one has to let the penalty parameters associated with Q go to their limits for a fixed penalty parameter r in p , and then one must repeat this process every time r in problem (8.18) is updated to solve the original problem. A critical theoretical result, however, states that all the penalty parameters can be updated simultaneously as part of the doubly penalized approach. This markedly increases the efficiency of this approach when compared to those in [A2, S8]. Further extensions can be found in [I2].

We now make the following assumptions.

Assumption 8.3.2 $\text{int}S(\mathbf{x})$ which is given by $\{\mathbf{y} : \mathbf{g}(\mathbf{x}, \mathbf{y}) < \mathbf{0}\}$ is not empty and its closure is $S(\mathbf{x})$.

Assumption 8.3.3 $\text{int}\{(\mathbf{x}, \mathbf{y}) : \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}\}$, which is given by $\{(\mathbf{x}, \mathbf{y}) : \mathbf{G}(\mathbf{x}, \mathbf{y}) < \mathbf{0}\}$ is not empty and its closure becomes $\text{int}\{(\mathbf{x}, \mathbf{y}) : \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}\}$.

Assumption 8.3.4 The functions F , \mathbf{G} , f and \mathbf{g} are continuous with respect to their arguments, and in particular, f and \mathbf{g} are differentiable in \mathbf{y} and ϕ is differentiable in \mathbf{g} . Furthermore, $\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$, $\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$ and $\nabla \phi$ are continuous with respect to their arguments.

Assumption 8.3.5 The set $\{(\mathbf{x}, \mathbf{y}) : \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}, \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}\}$ is compact.

It can be easily proven that under Assumptions 8.3.2~8.3.5 there exists a solution to problem (8.22) for every positive penalty parameter on $\{(\mathbf{x}, \mathbf{y}) : \mathbf{G}(\mathbf{x}, \mathbf{y}) < \mathbf{0}, \mathbf{g}(\mathbf{x}, \mathbf{y}) < \mathbf{0}\}$. Also, from some standard results of the penalty function method (e.g, see Chapter 3 in [S10]), we have

Lemma 8.3.1 Let \mathbf{x} be given arbitrarily such that $\mathbf{G}(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) < \mathbf{0}$ for the corresponding $\mathbf{y}^*(\mathbf{x})$, and $\{\mathbf{y}^k\}$ be a sequence of optimal solutions to $\min_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}; r^k)$ corresponding to a positive sequence $\{r^k\}$ converging to zero. Then, for any positive sequence $\{s^k\}$ diverging to infinity and any positive sequence $\{t^k\}$ converging to zero

$$\lim_{k \rightarrow \infty} Q(\mathbf{x}, \mathbf{y}^k; t^k, s^k, r^k) = F(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))\tag{8.23}$$

We now add the following four assumptions.

Assumption 8.3.6 The set $\{\mathbf{x} : \mathbf{G}(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) < \mathbf{0}\}$ is not empty and its closure is $\{\mathbf{x} : \mathbf{G}(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) \leq \mathbf{0}\}$.

Assumption 8.3.7 The function ϕ is additively separable: $\phi(\mathbf{g}) = \sum_{i=1}^q \varphi(g_i)$, where φ is a monotone increasing continuously differentiable function on $(-\infty, 0)$.

Assumption 8.3.8 Let $I(\mathbf{x}, \mathbf{y}) = \{i : g_i(\mathbf{x}, \mathbf{y}) = 0\}$ for any fixed (\mathbf{x}, \mathbf{y}) . Then $\nabla_{\mathbf{y}} g_i(\mathbf{x}, \mathbf{y})$, $i \in I(\mathbf{x}, \mathbf{y})$ are linearly independent.

Assumption 8.3.9 The functions f and \mathbf{g} are convex in \mathbf{y} for every fixed \mathbf{x} .

Lemma 8.3.2 Let $\{(\mathbf{x}^k, \mathbf{y}^k)\}$ be a sequence of optimal solutions to problem (8.22) corresponding to positive sequences $\{r^k\}$ and $\{t^k\}$ converging to zero and a positive sequence $\{s^k\}$ diverging to infinity. If Assumptions 8.3.1 ~ 8.3.9 are satisfied and there exists an accumulation point $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ of $\{(\mathbf{x}^k, \mathbf{y}^k)\}$, then $\tilde{\mathbf{y}}$ solves the follower's problem for a given $\tilde{\mathbf{x}}$.

Proof: See [A3] or [S10]. In light of the above lemmas, we now prove the main theorem needed for convergence.

Theorem 8.3.1 Let $\{(\mathbf{x}^k, \mathbf{y}^k)\}$ be a sequence of optimal solutions of problem (8.22) corresponding to positive sequences $\{t^k\}$ and $\{r^k\}$ converging to zero, and a positive sequence $\{s^k\}$ diverging to infinity. If Assumptions 8.3.1~8.3.9 are satisfied then the sequence $\{(\mathbf{x}^k, \mathbf{y}^k)\}$ has accumulation points, any one of which solves problem (8.3.1).

Proof: Since $\{(\mathbf{x}^k, \mathbf{y}^k)\}$ belongs to the compact set of Assumption 8.3.5, it has an accumulation point. Denote any one of the accumulation points by $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ and a newly convergent sequence of $\{(\mathbf{x}^k, \mathbf{y}^k)\}$ to $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ by $\{(\hat{\mathbf{x}}^k, \hat{\mathbf{y}}^k)\}$.

Feasibility of accumulation points: Lemma 8.3.2 says that $\tilde{\mathbf{y}}$ solves the follower's problem for $\tilde{\mathbf{x}}$ fixed. Since $\mathbf{y}^*(\tilde{\mathbf{x}})$ is unique under Assumption 8.3.1, we have $\tilde{\mathbf{y}} = \mathbf{y}^*(\tilde{\mathbf{x}})$. Furthermore, since $\mathbf{G}(\mathbf{x}^k, \mathbf{y}^k) < \mathbf{0}$, the continuity of \mathbf{G} implies that $\mathbf{G}(\mathbf{x}^k, \mathbf{y}^k) \leq \mathbf{0}$. Thus, $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ is feasible for problem (8.14).

Optimality of accumulation points: Suppose that $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ does not solve problem (8.14). Then there exists an $\hat{\mathbf{x}}$ such that

$$\mathbf{G}(\hat{\mathbf{x}}, \mathbf{y}^*(\hat{\mathbf{x}})) \leq \mathbf{0} \quad (8.24)$$

$$F(\hat{\mathbf{x}}, \mathbf{y}^*(\hat{\mathbf{x}})) < F(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = F(\tilde{\mathbf{x}}, \mathbf{y}^*(\tilde{\mathbf{x}})) \quad (8.25)$$

Because $\mathbf{y}^*(\mathbf{x})$ is continuous at any \mathbf{x} under Assumptions 8.3.1, 8.3.2, 8.3.4 and 8.3.5, as proved by Hogan [H6], $F(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))$ also becomes continuous at any \mathbf{x} under the

continuity of F . Therefore, consider an open ball $B(\hat{\mathbf{x}}; \delta)$ around $\hat{\mathbf{x}}$ with radius δ ; then there exists a number $\delta > 0$ such that $F(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) < F(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ for $\forall \mathbf{x} \in B(\hat{\mathbf{x}}; \delta)$. Furthermore, (8.24) and Assumption 8.3.6 imply that there exists another point $\hat{\tilde{\mathbf{x}}} \in \{\mathbf{x} : \mathbf{G}(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) < 0\} \cap B(\hat{\mathbf{x}}; \delta)$. That is, there exists an $\hat{\tilde{\mathbf{x}}}$ such that

$$\mathbf{G}(\hat{\tilde{\mathbf{x}}}, \mathbf{y}^*(\hat{\tilde{\mathbf{x}}})) < 0 \quad (8.26)$$

$$F(\hat{\tilde{\mathbf{x}}}, \mathbf{y}^*(\hat{\tilde{\mathbf{x}}})) < F(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \quad (8.27)$$

and let

$$F(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) - F(\hat{\tilde{\mathbf{x}}}, \mathbf{y}^*(\hat{\tilde{\mathbf{x}}})) = 2\varepsilon, \quad \varepsilon > 0. \quad (8.28)$$

On the other hand, considering the solution $\mathbf{y}^*(\hat{\tilde{\mathbf{x}}})$ to the follower's problem with $\hat{\tilde{\mathbf{x}}}$ and the solution $\hat{\tilde{\mathbf{y}}}^k (= \mathbf{y}^*(\hat{\tilde{\mathbf{x}}}; r^k)) \in \text{int}S(\hat{\tilde{\mathbf{x}}})$ to the problem $\min_{\mathbf{y}} p(\hat{\tilde{\mathbf{x}}}, \mathbf{y}; r^k)$, we have $\hat{\tilde{\mathbf{y}}}^k \rightarrow \mathbf{y}^*(\hat{\tilde{\mathbf{x}}})$ as $k \rightarrow \infty$, based on the standard penalty function method. This convergence, the continuity of \mathbf{G} , and (8.26) imply the existence of a positive integer K such that

$$\mathbf{G}(\hat{\tilde{\mathbf{x}}}, \hat{\tilde{\mathbf{y}}}^k) < 0 \quad \forall k > K. \quad (8.29)$$

Together with the fact that $\hat{\tilde{\mathbf{y}}}^k \in \text{int}S(\hat{\tilde{\mathbf{x}}})$, (8.29) implies that $(\hat{\tilde{\mathbf{x}}}, \hat{\tilde{\mathbf{y}}}^k)$ belongs to the domain of definition of Q for all $k > K$.

Now, Lemma 8.3.1 implies that $\lim_{k \rightarrow \infty} Q(\hat{\tilde{\mathbf{x}}}, \hat{\tilde{\mathbf{y}}}^k; t^k, s^k, r^k) = F(\hat{\tilde{\mathbf{x}}}, \mathbf{y}^*(\hat{\tilde{\mathbf{x}}}))$. That is, there exists a positive integer \tilde{K} such that

$$|Q(\hat{\tilde{\mathbf{x}}}, \hat{\tilde{\mathbf{y}}}^k; t^k, s^k, r^k) - F(\hat{\tilde{\mathbf{x}}}, \mathbf{y}^*(\hat{\tilde{\mathbf{x}}}))| < \varepsilon \quad \forall k > \tilde{K} \quad (8.30)$$

for ε in (8.28). Also, from $(\mathbf{x}^k, \mathbf{y}^k) \rightarrow (\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ and continuity of F , we have the existence of a positive integer \tilde{K} such that

$$|F(\mathbf{x}^k, \mathbf{y}^k) - F(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})| < \varepsilon \quad \forall k > \tilde{K} \quad (8.31)$$

for the same ε . Using (8.28), (8.30), (8.31) and positiveness of the penalty term, in turn, we have the following relations for all values of the integer $k > \max(\tilde{K}, \tilde{K})$

$$Q((\hat{\tilde{\mathbf{x}}}, \hat{\tilde{\mathbf{y}}}^k; t^k, s^k, r^k) < F(\hat{\tilde{\mathbf{x}}}, \mathbf{y}^*(\hat{\tilde{\mathbf{x}}}) + \varepsilon = F(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) - \varepsilon < F(\mathbf{x}^k, \mathbf{y}^k) \leq Q(\mathbf{x}^k, \mathbf{y}^k; t^k, s^k, r^k) \quad (8.32)$$

This contradicts the fact that $(\mathbf{x}^k, \mathbf{y}^k)$ is an optimal solution of problem (8.22) for all positive integer $k > \max(K, \tilde{K}, \tilde{K})$. This completes the proof. ■

The theorem says that a limit point of $\{(\mathbf{x}^k, \mathbf{y}^k)\}$ generated by a series of problems (8.22) corresponding to a sequence $\{(r^k, s^k, t^k)\}$ for $r^k \rightarrow 0$, $s^k \rightarrow +\infty$, $t^k \rightarrow 0$, if it exists, is a solution to the bilevel program (8.14). Consequently, if positive

numbers r^k , s^k and t^k are given, the problem can be solved easily with unconstrained optimization techniques.

The following examples demonstrate the convergence properties stated in Theorem 8.3.1.

Example 8.3.1 Consider

$$\begin{aligned} \min_x \quad & x^2 + (y^*(x) - 10)^2 \\ \text{subject to} \quad & -x + y^*(x) \leq 0, \quad 0 \leq x \leq 15 \\ & (x + 2y^*(x) - 30)^2 = \min_y (x + 2y - 30)^2 \\ & \text{subject to} \quad x + y \leq 20 \\ & \quad 0 \leq y \leq 20 \end{aligned}$$

For this problem, SUMT-type penalty functions were used in the subsidiary and overall augmented objective functions, and solved $\min_{(x,y)} Q(x, y; t^k, s^k, r^k)$ with a Fletcher-Reeves conjugate gradient method. The computational results are summarized in Table 8.5.

Table 8.5 Computational results for Example 8.3.1

r^k	s^k	t^k	x^k	y^k	Q^k	F^k	f^k
100.	0.01	100.	8.650	5.708	212.1	93.2	91.8
10.	0.1	10.	9.348	8.676	125.4	89.1	10.90
1.0	1.0	1.0	9.875	9.587	103.6	97.7	0.903
0.1	10.	0.1	9.929	9.826	100.0	98.6	0.1751
0.01	100.	0.01	9.977	9.912	99.8	99.5	0.0399
0.001	1000.	0.001	9.980	9.917	517.6	99.6	0.0343
true values			10.000	10.000	—	100.0	0.000
$Q^k = Q(x^k, y^k; t^k, s^k, r^k), F^k = F(x^k, y^k), f^k = f(x^k, y^k)$							

Example 8.3.2 For $\mathbf{x} = (x_1, x_2)$ and $\mathbf{y} = (y_1, y_2)$ consider

$$\begin{aligned}
& \min_{\mathbf{x}} 2x_1 + 2x_2 - 3y_1^*(\mathbf{x}) - 3y_2^*(\mathbf{x}) - 60 \\
& \text{subject to } x_1 + x_2 + y_1^*(\mathbf{x}) - 2y_2^*(\mathbf{x}) - 40 \leq 0 \\
& \quad 0 \leq x_1 \leq 50, \quad 0 \leq x_2 \leq 50 \\
& \quad (y_1^*(\mathbf{x}) - x_1 + 20)^2 + (y_2^*(\mathbf{x}) - x_2 + 20)^2 \\
& \quad = \min_{\mathbf{y}} (y_1 - x_1 + 20)^2 + (y_2 - x_2 + 20)^2 \\
& \text{subject to } 2y_1 - x_1 + 10 \leq 0, \quad 2y_2 - x_2 + 10 \leq 0 \\
& \quad -10 \leq y_1 \leq 20, \quad -10 \leq y_2 \leq 20
\end{aligned}$$

The computational results are shown in Table 8.6.

Table 8.6 Computational results for Example 8.3.2

r^k	s^k	t^k	x_1^k	x_2^k	y_1^k	y_2^k	Q^k	F^k	f^k
100.	0.01	100.	12.44	17.84	-2.650	0.622	96.31	6.651	31.81
10.	0.1	10.	19.38	25.58	1.524	6.047	20.75	7.206	4.826
1.0	1.0	1.0	22.94	28.28	3.115	8.147	10.47	8.654	0.0495
0.1	10.	0.1	23.06	28.20	3.028	8.274	9.36	8.613	7.44×10^{-3}
0.01	100.	0.01	24.52	29.73	4.518	9.667	5.99	5.932	3.77×10^{-3}
0.001	1000.	0.001	24.52	29.71	4.515	9.693	5.91	5.825	1.80×10^{-4}
0.0001	10000.	0.0001	24.52	29.70	4.516	9.698	5.82	5.786	2.0×10^{-6}
true values			25.00	30.00	5.000	10.000	—	5.000	0.000
$Q^k = Q(x^k, y^k; t^k, s^k, r^k), \quad F^k = F(x^k, y^k), \quad f^k = f(x^k, y^k)$									

This output demonstrates the convergence property and applicability of the double penalty method. However, it should be mentioned that the authors experienced computational difficulties associated with ill-conditioning in the vicinity of the boundary of the feasible region when r and t were very small and s was large. It is to be seen whether this ill-conditioning can be overcome with existing techniques. An unsettled issue is that the original problem (8.14) as well as the transformed problem (8.22) are both nonconvex programs. Therefore, it is desirable that (8.22) be solved by a solution technique producing a global optimum. If one applies a standard gradient method, it is advisable to resolve problem (8.22) for several different initial points.

8.4 RECTANGULAR PARTITIONING

When the follower's optimality conditions are both necessary and sufficient, the non-linear BLPP (8.3a)–(8.3d) can be solved using global optimization techniques. The complementary slackness condition (8.13d) in the single-level formulation is usually the complicating constraint in such problems. Al-Khayyal et al. [A5] show how this constraint can be replaced by an equivalent system of convex separable quadratic constraints when the follower's problem is a linear program (actually, their approach should work when the follower's problem is a quadratic program as well). For the case where the leader's objective function is concave, they propose two different methods for finding the global minimum of a concave function subject to quadratic separable constraints. The first method is based on rectangular partitions of an outer approximation of the feasible region to obtain upper and lower bounds which are then used in an implicit enumeration scheme. We limit our discussion to this method.

In the development, assume that F is concave, \mathbf{G} is convex, and f and \mathbf{g} are convex in \mathbf{x} and affine in \mathbf{y} for problem (8.13). Then the follower's problem for \mathbf{x} fixed is a linear program and the feasible region of (8.13) is convex when the complementarity term $ug(\mathbf{x}, \mathbf{y}) = 0$ is removed. The first step taken by Al-Khayyal et al. is to show that this term can be expressed as a simple separable quadratic constraint augmented by a set of differentiable convex inequalities.

To see this, note that the complementary slackness constraint (8.13d) may be written as $\sum_i u_i g_i(\mathbf{x}, \mathbf{y}) = 0$. Given that $(u_i, -g_i) \geq (0, 0)$ for all i , this constraint is equivalent to

$$\begin{aligned} 0 &= \sum_i \min\{u_i, -g_i(\mathbf{x}, \mathbf{y})\} \\ &= \sum_i (u_i - \max\{0, u_i + g_i(\mathbf{x}, \mathbf{y})\}) \end{aligned}$$

which, in turn, is equivalent to

$$0 = \sum_i (u_i^2 - z_i) \tag{8.33a}$$

$$z_i \geq (\max\{0, u_i + g_i(\mathbf{x}, \mathbf{y})\})^2 \quad \forall i \tag{8.33b}$$

when the objective function to be minimized is taken to be $F(\mathbf{x}, \mathbf{y}) + \sum_i z_i$, since at an optimal solution of this objective, the inequalities (8.33b) are binding. The desired system is obtained by expressing the equation (8.33a) as two inequalities. Notice that for each i we have

$$u_i + g_i < 0 \Rightarrow z_i = 0 \Rightarrow u_i^2 = 0 \Rightarrow g_i < 0$$

and

$$\begin{aligned} u_i + g_i > 0 \Rightarrow z_i &= (u_i + g_i)^2 \Rightarrow u_i^2 - (u_i + g_i)^2 = 0 \\ &\Rightarrow g_i(u_i + 2u_i) = 0 \Rightarrow g_i = 0 \Rightarrow u_i > 0 \end{aligned}$$

The above reformulation (8.33a,b) is similar to that proposed by Bard and Falk [B10] who introduced a piecewise linear separable convex equation and a set of auxiliary linear equalities as replacement for the complementarity term (see eq. (5.15a,b)).

For the new formulation, the separable quadratic reverse convex inequality constraint obtained by writing (8.33a) as two inequalities is the most difficult element to deal with; namely,

$$\sum_i (z_i - u_i^2) \leq 0$$

Primarily for this reason, the authors restrict themselves to implementable procedures for minimizing concave functions over bounded nonconvex sets defined by separable quadratic constraints. More precisely, their goal is to determine the global minimum of problems taking the form

$$\begin{aligned} & \text{global min } F(\mathbf{x}) \\ & \text{subject to } g_i(\mathbf{x}) \triangleq \sum_{j=1}^n \left(\frac{1}{2} p_{ij} x_j^2 + q_{ij} x_j + r_{ij} \right) \leq 0, \quad i = 1, \dots, m \\ & \quad a_j \leq x_j \leq b_j, \quad j = 1, \dots, n \end{aligned} \quad (8.34)$$

where p_{ij} , q_{ij} , r_{ij} , a_j , b_j ($i = 1, \dots, m$ and $j = 1, \dots, n$) are given real numbers, $F(\mathbf{x})$ is a real-valued concave function defined on an open convex set containing the rectangle $M_0 = \{\mathbf{x} : a_j \leq x_j \leq b_j, \forall j\}$. Now, denote by D the feasible set of problem (8.34). Because D is compact and F is continuous on D , the global minimum of F over D exists.

There has been little work done on nonconvex problems with quadratic constraints of the form (8.34). Even for the case where $F(\mathbf{x})$ is linear, the problem has been shown to be NP-hard. The branch and bound algorithm proposed by Al-Khayyal et al. consists of

- increasingly refined rectangular partitions of the initial rectangle M_0 ,
- lower bounds $\beta(M) \leq \min F(M)$ associated with each partition element (rectangle) M generated by the procedure,
- upper bounds $\alpha_r \geq \min F(D)$ determined in each step r of the algorithm,
- certain deletion rules by which some partition sets M are deleted when it is known that $M \cap D = \emptyset$ or that $\min F(D)$ cannot be attained in M .

Each of these aspects will be discussed briefly before presenting the algorithmic steps. Proofs not given can be found in the original work [A5].

Subdivision of Rectangles

Definition 8.4.1 Let $M_i \subset R^n$ be an n -dimensional rectangle (n -rectangle) and let I be a finite set of indices. A set $\{M_i : i \in I\}$ of n -rectangles $M_i \subset M$ is said to be a rectangular partition of M if we have

$$M = \bigcup_{i \in I} M_i, \quad M_i \cap M_j = \partial M_i \cap \partial M_j, \quad \text{for all } i, j \in I, i \neq j$$

where ∂M_i denotes the boundary of M_i .

Definition 8.4.2 Let $\{M_q\}$ be an infinite decreasing (nested) sequence of rectangular partition sets generated by the algorithm. The underlying subdivision procedure of rectangles is called exhaustive if the sequence of diameters $d(M_q)$ of M_q satisfies

$$\lim_{q \rightarrow \infty} d(M_q) = 0$$

Note that an n -rectangle $M = \{\mathbf{x} : \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}\}$, $\mathbf{a}, \mathbf{b} \in R^n$, $\mathbf{a} < \mathbf{b}$, is uniquely determined by its lower left vertex \mathbf{a} and its upper right vertex \mathbf{b} . Each of the 2^n vertices of M is of the form $\mathbf{a} + \boldsymbol{\theta}$ where $\boldsymbol{\theta}$ is a vector with components 0 or $(b_i - a_i)$, $i = 1, \dots, n$, and for the diameter $d(M)$ of M we have $d(M) = \|\mathbf{a} - \mathbf{b}\|$, where $\|\cdot\|$ denotes the Euclidean norm in R^n .

Bisection is a very simple procedure for subdividing an n -rectangle $M = \{\mathbf{x} : \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}\}$ into two n -rectangles by a cutting hyperplane through $(\mathbf{a} + \mathbf{b})/2$, perpendicular to the longest edge of M . It is well known that bisection is exhaustive in the sense of Definition 8.4.2.

Lower and Upper Bounds

Let M be an n -rectangle and denote by $V(M)$ the vertex set of M . Then we know that by the concavity of F that $\min F(M) = \min F(V(M))$ and for the lower bound $\beta(M)$ we set

$$\beta(M) = \min F(V(M))$$

Now, whenever $D \cap M \neq \emptyset$, we have $\beta(M) \leq \min F(D \cap M)$.

For the upper bounds $\alpha_r \geq \min F(D)$ in step r of the algorithm, the authors always choose

$$\alpha_r = \min F(S_r)$$

where S_r is the set of feasible points calculated prior to step r ; that is, $\alpha_r = F(\mathbf{x}^r)$ where \mathbf{x}^r is the best feasible point determined so far. If feasible points are not available; i.e., $S_r = \emptyset$, set $\alpha_r = \infty$.

Deletion

A partition of M may be deleted whenever $\beta(M) > \alpha_s$ for some iteration step s . In this case, it should be clear that the global minimum on D cannot be attained

in M . A more difficult question is that of properly deleting infeasible partition sets M ; i.e., sets satisfying $M \cap D = \emptyset$. Usually, M is known by its vertex set $V(M)$ and $V(M) \cap D = \emptyset$ does not imply $M \cap D = \emptyset$. Therefore, from the information at hand, it is not appropriate to delete all partition sets M satisfying $M \cap D = \emptyset$, so it is necessary to apply a “deletion by infeasibility” rule that is “certain in the limit” in the sense of fathoming “enough” infeasible sets to guarantee convergence of the algorithm to a global minimum of F on D .

In order to derive such a rule, note that each constraint function g_i is Lipschitzian on M , i.e., there is a constant $L_i = L_i(M) > 0$ such that

$$|g_i(\mathbf{z}) - g_i(\mathbf{x})| \leq L_i \|\mathbf{z} - \mathbf{x}\|, \quad \text{for all } \mathbf{x}, \mathbf{z} \in M$$

An upper bound for L_i is given by any number A_i satisfying

$$A_i \geq \max \{\|\nabla g_i(\mathbf{y})\| : \mathbf{y} \in M\}$$

Let $M = \{\mathbf{x} : a_j \leqq x_j \leqq b_j; j = 1, \dots, n\}$. Using monotonicity and separability, we see that

$$A_i = A_i^* = \max \{\|\nabla g_i(\mathbf{y})\| : \mathbf{y} \in M\}$$

is given by

$$\begin{aligned} A_i^* &= \left[\sum_{j=1}^n \left(\max_{a_j \leqq y_j \leqq b_j} |p_{ij}y_j + q_{ij}| \right)^2 \right]^{1/2} \\ &= \left[\sum_{j \in N_i^1} (p_{ij}a_j + q_{ij})^2 + \sum_{j \in N_i^2} (p_{ij}b_j + q_{ij})^2 \right]^{1/2} \end{aligned} \quad (8.35)$$

where

$$N_i^1 = \left\{ j \mid -\frac{q_{ij}}{p_{ij}} \geqq \frac{a_j + b_j}{2} \right\}, \quad N_i^2 = \left\{ j \mid -\frac{q_{ij}}{p_{ij}} < \frac{a_j + b_j}{2} \right\},$$

Let $V'(M)$ be any nonempty subset of the vertex set $V(M)$; e.g., $V'(M) = \{\mathbf{a}, \mathbf{b}\}$ if $M = \{\mathbf{x} : \mathbf{a} \leqq \mathbf{x} \leqq \mathbf{b}\}$. Denote again by $d(M) = \|\mathbf{a} - \mathbf{b}\|$ the diameter of M . Al-Khayyal et al. propose the following:

Deletion Rule (DR)

Delete a partition set M whenever there is an $i \in \{1, \dots, m\}$ satisfying

$$\max\{g_i(\mathbf{x}) : \mathbf{x} \in V'(M)\} - A_i d(M) \geqq 0 \quad (8.36)$$

where $A_i \geqq L_i$ (see (8.35)).

Lemma 8.4.1 Let the subdivision procedure be exhaustive and apply the deletion rule (DR). Then every infinite decreasing sequence $\{M_q\}$ of partition sets generated by the algorithm satisfies

$$M_q \xrightarrow[q \rightarrow \infty]{} \{\bar{x}\}, \quad \bar{x} \in D$$

To see that the deletion rule only fathoms infeasible sets M , note that by $A_i \geq L_i$, we have $g_i(z) \geq g_i(x) - L_i \|z - x\| \geq g_i(x) - A_i d(M)$ for all $x, z \in M$ which implies along with (8.36) that $g_i(z) > 0$ for all $z \in M$.

Algorithm

Step 0 Let $M_0 = M$, choose $S_{M_0} \subset D$ (possibly empty) and determine $\beta(M_0) = \min F(V(M))$, $\alpha_0 = \min F(S_{M_0})$ ($\alpha_0 = \infty$ if $S_{M_0} = \emptyset$). Let $I_0 = \{M_0\}$ and $\beta_0 = \beta(M_0)$. If $\alpha_0 < \infty$, choose $x^0 \in \arg \min F(S_{M_0})$ (i.e., $F(x^0) = \alpha_0$). If $\alpha_0 - \beta_0 = 0$ ($\leq \varepsilon > 0$), then stop; $\alpha_0 = \beta_0 = \min F(D)$ ($\alpha_0 - \beta_0 \leq \varepsilon$), x^0 is an ε -approximate solution. Otherwise, set $r = 1$ and go to Step r.

Step r At the beginning of this step, the current rectangular partition I_{r-1} of a subset of M_0 is still under consideration. Furthermore, for every $M \in I_{r-1}$ we have $S_M \subset M \cap D$ and bounds $\beta(M)$, $\alpha(M)$ satisfying

$$\beta(M) = \min F(M) \leq \alpha(M)$$

Moreover, the current lower and upper bounds β_{r-1} and α_{r-1} satisfy

$$\beta_{r-1} = \min F(D) \leq \alpha_{r-1}$$

Finally, if $\alpha_{r-1} < \infty$, we have a point $x^{r-1} \in D$ satisfying $F(x^{r-1}) = \alpha_{r-1}$ (the best feasible point obtained so far).

- r1 Delete all $M \in I_{r-1}$ satisfying $\beta(M) \geq \alpha_{r-1}$. Let R_r be the collection of the remaining rectangles in the partition I_{r-1} .
- r2 Select a nonempty collection of sets $P_r \subset R_r$ satisfying

$$\arg \min \{\beta(M) : M \in I_{r-1}\} \subset P_r$$

and subdivide every member of P_r by bisection (or any other exhaustive subdivision yielding rectangular partitions). Let P'_r be the collection of all new partition elements.

- r3 Delete any $M \in P'_r$ satisfying the deletion rule (DR). Let I'_r be the collection of all remaining members of P'_r .

r4 Assign to each $M \in I'_r$ the set $S_M \subset M \cap D$ of feasible points in M known so far and

$$\beta(M) = \min F(V(M)), \quad \alpha(M) = \min F(S_M) \quad (\alpha(M) = \infty \text{ if } S_M \neq \emptyset).$$

r5 Set $I_r = (R_r - P_r) \cup I'_r$. Compute

$$\begin{aligned}\alpha_r &= \inf\{\alpha(M) : M \in I_r\} \\ \beta_r &= \min\{\beta(M) : M \in I_r\}\end{aligned}$$

If $\alpha_r \leq \infty$, let $\mathbf{x}^r \in D$ such that $F(\mathbf{x}^r) = \alpha_r$.

r6 If $\alpha_r - \beta_r = 0$ ($\leq \varepsilon$), then stop; \mathbf{x}^r is an ε -approximate solution. Otherwise, put $r \leftarrow r + 1$ and go to step r.

The convergence of the algorithm is based on the following theorem.

Theorem 8.4.1

(i) The sequence of lower bounds β satisfies

$$\beta := \lim_{r \rightarrow \infty} \beta_r = \min F(D)$$

(ii) Assume that we have $S_M \neq \emptyset$ for all partition sets M . Then

$$\beta = \lim_{r \rightarrow \infty} \beta_r = \min F(D) = \lim_{r \rightarrow \infty} \alpha_r =: \alpha$$

holds and every accumulation point of the sequence $\{\mathbf{x}^r\}$ solves (8.34).

For problem (8.34), it may be difficult to obtain feasible points such that $S_M \neq \emptyset$ for all partition elements M . In that case, the authors propose to consider the iterative sequence $\{\tilde{\mathbf{x}}^r\}$ defined by $F(\tilde{\mathbf{x}}^r) = \beta_r$. Note that $\tilde{\mathbf{x}}^r$ is not necessarily feasible to (8.34) but it can be proven that every accumulation point of $\{\tilde{\mathbf{x}}^r\}$ is, indeed, a solution.

8.5 STEEPEST DESCENT DIRECTION

A number of algorithms have been developed using gradient information to find good search directions for the leader's objective function. In this regard, Dempe [D2] was one of the first to derive necessary optimality conditions for the BLPP. His approach was based on the nonexistence of a descent direction in a combinatorial system but no suggestions were offered for verifying the existence or the nonexistence of the given direction. In this section, following the work of Savard and Gauvin [S2], we give

alternative necessary optimality conditions for the BLPP based on the directional derivative of the rational reaction function $\mathbf{y}(\mathbf{x})$. The advantage of these conditions is that it can be verified efficiently. In addition, we characterize the steepest descent direction and propose a descent method to determine a local optimum. In the developments, the lower-level problem is viewed as a parametric program where the optimal solution $\mathbf{y}(\mathbf{x})$ depends on the parameter \mathbf{x} .

Hogan [H6] and Tanino and Ogawa [T1] have proposed the use of the directional derivative for the solution of programs with implicitly defined constraints. Their work relates to the specific case in which the optimal value of the lower-level problem

$$w(\mathbf{x}) = \min\{f(\mathbf{x}, \mathbf{y}) : \mathbf{y} \in S(\mathbf{x})\},$$

is used instead of the optimal solution

$$\mathbf{y}(\mathbf{x}) = \arg \min\{f(\mathbf{x}, \mathbf{y}) : \mathbf{y} \in S(\mathbf{x})\}$$

in the upper-level objective, where $S(\mathbf{x})$ is the feasible region (8.3d). The resultant problems are convex under standard convexity assumptions on the upper- and lower-level problems; the steepest descent direction can be found by solving a linear program. This structure appears frequently in the decomposition of mathematical programs and is discussed at length in Chapters 6–8 in [S10]. Under the same assumptions, however, the general bilevel problem is not convex. Nevertheless, it is shown presently that the steepest descent direction can be found by solving a linear-quadratic bilevel program.

8.5.1 Necessary Optimality Conditions

For convenience, we rewrite the nonlinear bilevel programming problem (8.3a)–(8.3d) in the following form:

$$\begin{aligned} & \min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}) \text{ where } \mathbf{y} \text{ solves} \\ & \quad \min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \\ & \quad \text{subject to } g_i(\mathbf{x}, \mathbf{y}) \leq 0, \quad i \in I \\ & \quad \quad g_i(\mathbf{x}, \mathbf{y}) = 0, \quad i \in J \end{aligned} \tag{8.37}$$

Here, I, J are finite sets of indices and $S(\mathbf{x}) = \{\mathbf{y} : g_i(\mathbf{x}, \mathbf{y}) \leq 0, i \in I \text{ and } g_i(\mathbf{x}, \mathbf{y}) = 0, i \in J\}$ is the lower-level feasible region.

Assumption 8.5.1 The problem is well-posed, i.e., that for any \mathbf{x} , the optimal solution $\mathbf{y}(\mathbf{x})$ of the lower-level problem is unique.

Assumption 8.5.2 If $I(\mathbf{x}) = \{i \in I : g_i(\mathbf{x}, \mathbf{y}(\mathbf{x})) = 0\}$ denotes the set of indices for the binding inequality constraints at $(\mathbf{x}, \mathbf{y}(\mathbf{x}))$, then the vectors

$$\nabla_{\mathbf{y}} f_i(\mathbf{x}, \mathbf{y}(\mathbf{x})), \quad i \in I(\mathbf{x}) \cup J$$

are linearly independent. This implies the existence of a unique Kuhn-Tucker vector $\mathbf{u}(\mathbf{x})$ corresponding to the optimal solution $\mathbf{y}(\mathbf{x})$. We denote by

$$T(\mathbf{x}) = \{\mathbf{v} \in R^m : \nabla_{\mathbf{y}} g_i(\mathbf{x}, \mathbf{y}(\mathbf{x}))\mathbf{v} = 0, i \in I(\mathbf{x}) \cup J\}$$

the tangent subspace at $\mathbf{y}(\mathbf{x})$.

Assumption 8.5.3 Second-order sufficiency condition:

$$\mathbf{v}^T \nabla_{\mathbf{y}}^2 \mathcal{L}(\mathbf{y}(\mathbf{x}); \mathbf{u}(\mathbf{x}))\mathbf{v} > 0, \quad \forall \mathbf{v} \in T(\mathbf{x}), \mathbf{v} \neq 0$$

where $\mathcal{L}(\mathbf{x}, \mathbf{y}; \mathbf{u}) = f(\mathbf{x}, \mathbf{y}) + \sum_{i \in I(\mathbf{x}) \cup J} \mathbf{u}_i g_i(\mathbf{x}, \mathbf{y})$ is the Lagrangian corresponding to the lower-level problem.

Assumption 8.5.1 assures that the bilevel program has a solution, as discussed in Section 8.1. Assumption 8.5.2 imposes a first-order necessary condition on the lower-level problem. Taken together with Assumption 8.5.3, they guarantee at least Lipschitzian behavior of the function $\mathbf{y}(\mathbf{x})$.

Remark 1 At the cost of a slightly more complicated formulation of the quadratic program defined below, the second assumption can be replaced by the Mangasarian-Fromowitz constraint qualification. In that case, the Kuhn-Tucker vectors would not necessarily be unique. The third assumption cannot be significantly relaxed.

Under Assumptions 8.5.1~8.5.3, we have the following first-order necessary optimality conditions.

Theorem 8.5.1 Let $(\mathbf{x}^*, \mathbf{y}(\mathbf{x}^*))$ be an optimal solution to the BLPP (8.37). Then for any upper-level direction $\mathbf{d} \in R^n$ at \mathbf{x}^* , the directional derivative of the objective function of the upper-level problem satisfies:

$$F'(\mathbf{x}^*, \mathbf{y}(\mathbf{x}^*); \mathbf{d}) = \nabla_{\mathbf{x}} F(\mathbf{x}^*, \mathbf{y}(\mathbf{x}^*))\mathbf{d} + \nabla_{\mathbf{y}} F(\mathbf{x}^*, \mathbf{y}(\mathbf{x}^*))\mathbf{w}(\mathbf{x}^*; \mathbf{d}) \geq 0$$

where $\mathbf{w}(\mathbf{x}^*; \mathbf{d}) \in R^m$ is the optimal solution for $\mathbf{x} = \mathbf{x}^*$ of the quadratic program (QP):

$$\begin{aligned} & \min_{\mathbf{w}} (\mathbf{d}^T, \mathbf{w}^T) \nabla_{\mathbf{x}\mathbf{y}}^2 \mathcal{L}(\mathbf{x}, \mathbf{y}(\mathbf{x}); \mathbf{u}(\mathbf{x}))(\mathbf{d}, \mathbf{w}) \\ & \text{subject to } \nabla_{\mathbf{y}} g_i(\mathbf{x}, \mathbf{y}(\mathbf{x}))\mathbf{w} \leq -\nabla_{\mathbf{x}} g_i(\mathbf{x}, \mathbf{y}(\mathbf{x}))\mathbf{d}, \quad i \in I(\mathbf{x}) \\ & \quad \nabla_{\mathbf{y}} g_i(\mathbf{x}, \mathbf{y}(\mathbf{x}))\mathbf{w} = -\nabla_{\mathbf{x}} g_i(\mathbf{x}, \mathbf{y}(\mathbf{x}))\mathbf{d}, \quad i \in J \\ & \quad \nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}(\mathbf{x}))\mathbf{w} = -\nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}(\mathbf{x}))\mathbf{d} + \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{y}(\mathbf{x}); \mathbf{u}(\mathbf{x}))\mathbf{d} \end{aligned} \tag{8.38}$$

Proof: At a local optimum $(\mathbf{x}^*, \mathbf{y}(\mathbf{x}^*))$ of (8.37), we have for any direction $\mathbf{d} \in R^n$ and for some $t_0 > 0$,

$$F(\mathbf{x}^*, \mathbf{y}(\mathbf{x}^*)) \leq F(\mathbf{x}^* + t\mathbf{d}, \mathbf{y}(\mathbf{x}^* + t\mathbf{d})), \quad t \in (0, t_0)$$

where $\mathbf{y}(\mathbf{x}^* + t\mathbf{d})$ is the optimal solution of the lower-level problem for $\mathbf{x}^* + t\mathbf{d}$ fixed. It follows that

$$F'(\mathbf{x}^*, \mathbf{y}(\mathbf{x}^*); \mathbf{d}) = \lim_{t \downarrow 0} [F(\mathbf{x}^* + t\mathbf{d}, \mathbf{y}(\mathbf{x}^* + t\mathbf{d})) - F(\mathbf{x}^*, \mathbf{y}(\mathbf{x}^*))]/t \geq 0.$$

By the chain rule, we obtain the first-order necessary optimality condition

$$\nabla_{\mathbf{x}} F(\mathbf{x}^*, \mathbf{y}(\mathbf{x}^*))\mathbf{d} + \nabla_{\mathbf{y}} F(\mathbf{x}^*, \mathbf{y}(\mathbf{x}^*))\mathbf{y}'(\mathbf{x}^*; \mathbf{d}) \geq 0$$

where

$$\mathbf{y}'(\mathbf{x}^*; \mathbf{d}) = \lim_{t \downarrow 0} [\mathbf{y}(\mathbf{x}^* + t\mathbf{d}) - \mathbf{y}(\mathbf{x}^*)]/t$$

is the directional derivative of the optimal solution $\mathbf{y}(\mathbf{x}^* + t\mathbf{d})$ at $t = 0^+$ in the direction \mathbf{d} . According to a result about the directional derivative of this optimal solution originally stated in [J3], it happens that the quadratic program (8.38) has, under the given assumptions, a unique optimal solution $\mathbf{w}(\mathbf{x}^*; \mathbf{d})$ which turns out to be the directional derivative $\mathbf{y}'(\mathbf{x}^*; \mathbf{d})$. ■

The computation of the steepest descent direction for (8.37) is now straightforward. For a nonoptimal point $(\mathbf{x}, \mathbf{y}(\mathbf{x}))$, a direction of descent at \mathbf{x} is a $\mathbf{d} \in R^n$ such that

$$\nabla_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}(\mathbf{x}))\mathbf{d} + \nabla_{\mathbf{y}} F(\mathbf{x}, \mathbf{y}(\mathbf{x}))\mathbf{w}(\mathbf{x}; \mathbf{d}) < 0$$

Therefore the steepest descent direction at \mathbf{x} for the BLPP is given by any optimal solution of the following *linear-quadratic bilevel program*:

$$\begin{aligned} & \min_{\mathbf{d}} \quad \nabla_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}(\mathbf{x}))\mathbf{d} + \nabla_{\mathbf{y}} F(\mathbf{x}, \mathbf{y}(\mathbf{x}))\mathbf{w}(\mathbf{x}; \mathbf{d}) \\ & \text{subject to} \quad -1 \leq d_j \leq 1, \quad j = 1, \dots, n \end{aligned} \tag{8.39}$$

where $\mathbf{w}(\mathbf{x}; \mathbf{d})$ solves the QP (8.38).

The quadratic bilevel program (8.39) can be solved by several efficient algorithms; e.g., see Sections 5.3.2 and 5.3.4.

8.5.2 Overview of Descent Method

The algorithm below shows how the results of Theorem 8.5.1 can be used to develop a descent method for (8.37). Let $(\mathbf{x}^k, \mathbf{y}(\mathbf{x}^k))$ denote the solution at iteration k .

Step 0 (Initial rational point) Find an initial point in the inducible region by, say, solving the relaxed mathematical program derived from (8.37) by removing the follower's objective function. Fix \mathbf{x} at the solution to the relaxed program and solve the lower-level problem in (8.37). The solution is a rational point in IR .

Step 1 (Steepest descent direction) Solve the quadratic bilevel programming problem (8.39) to obtain the steepest descent direction $\mathbf{d}^k \in R^n$ with corresponding direction $\mathbf{w}(\mathbf{x}^k; \mathbf{d}^k)$ for the lower-level problem. If it happens that the optimal value of (8.39) is nonnegative then stop; $(\mathbf{x}^k, \mathbf{y}(\mathbf{x}^k))$ satisfies the necessary optimality conditions. Otherwise go to Step 2.

Step 2 (Step length computation) Compute a step length t_k such that

$$F(\mathbf{x}^k + t_k \mathbf{d}^k, \mathbf{y}(\mathbf{x}^k + t_k \mathbf{d}^k)) < F(\mathbf{x}^k, \mathbf{y}(\mathbf{x}^k))$$

Return to Step 1 with the feasible point $(\mathbf{x}^{k+1}, \mathbf{y}(\mathbf{x}^{k+1})) = (\mathbf{x}^k + t_k \mathbf{d}^k, \mathbf{y}(\mathbf{x}^k + t_k \mathbf{d}^k))$ and put $k \leftarrow k + 1$.

If convergence is obtained, the steepest descent algorithm will usually yield a local optimum for the BLPP. A global optimization scheme is then needed to ensure global optimality.

Remark 2 At Step 2, it should be noted that whenever $F(\mathbf{x}^k + t \mathbf{d}^k, \mathbf{y}(\mathbf{x}^k + t \mathbf{d}^k))$ is required, the lower-level problem for fixed $(\mathbf{x}^k + t \mathbf{d}^k)$ must be solved. Assumption 8.5.1 assures that there exists a $t^* > 0$ such that for all $t \in (0, t^*]$, the feasible set for the follower $S(\mathbf{x}^k + t \mathbf{d}^k)$ is nonempty and his optimal solution $\mathbf{y}^k = \mathbf{y}(\mathbf{x}^k + t \mathbf{d}^k)$ exists (see [G3]). Hence the use of an approximate line search technique adapted for a nondifferentiable function is justified (see [L3]).

Remark 3 In the implementation of the algorithm, it is not necessary to solve (8.39) exactly. Any rational solution with negative objective value yields a descent direction.

The following example from [S2] is presented to illustrate the computations. The geometry is shown in Fig. 8.3. Because the feasible set is a polyhedron, the maximum step length allowed will be one that maintains the feasibility of $(\mathbf{x}^k + t \mathbf{d}^k, \mathbf{y}^k + t \mathbf{w}^k)$.

Example 8.5.1

$$\begin{aligned} \min_x \quad & (x - 1)^2 + 2y_1^2 - 2x \\ \min_{y_1, y_2} \quad & (2y_1 - 4)^2 + (2y_2 - 1)^2 + xy_1 \\ \text{subject to} \quad & 4x + 5y_1 + 4y_2 \leq 12 \\ & -4x - 5y_1 + 4y_2 \leq -4 \\ & 4x - 4y_1 + 5y_2 \leq 4 \\ & -4x + 4y_1 + 5y_2 \leq 4 \\ & x \geq 0, y_i \geq 0, \quad i = 1, 2 \end{aligned}$$

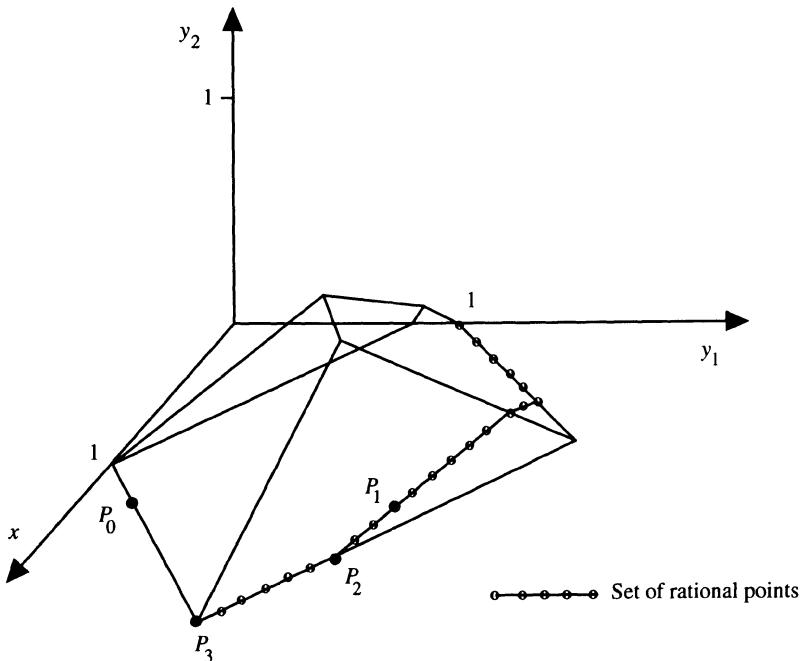


Figure 8.3 Geometry of Example 8.5.1

For the example, we have:

$$\begin{aligned}\mathcal{L}(x, \mathbf{y}; \mathbf{u}) &= (2y_1 - 4)^2 + (2y_2 - 1)^2 + xy_1 + u_1(4x + 5y_1 + 4y_2 - 12) \\ &\quad + u_2(-4x - 5y_1 + 4y_2 + 4) + u_3(4x - 4y_1 + 5y_2 - 4) \\ &\quad + u_4(-4x + 4y_1 + 5y_2 - 4) - u_5x - u_6y_1 - u_7y_2\end{aligned}$$

$$\begin{aligned}\nabla_{xy} \mathcal{L}(x, \mathbf{y}; \mathbf{u}) &= (y_1 + 4u_1 - 4u_2 + 4u_3 - 4u_4 - u_5, \\ &\quad 4(2y_1 - 4) + x + 5u_1 - 5u_2 - 4u_3 + 4u_4 - u_6, \\ &\quad 4(2y_2 - 1) + 4u_1 + 4u_2 + 5u_3 + 5u_4 - u_7)\end{aligned}$$

$$\nabla_{xy}^2 \mathcal{L}(x, \mathbf{y}; \mathbf{u}) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 8 & 0 \\ 0 & 0 & 8 \end{pmatrix}$$

$$\nabla_{xy} F(x, \mathbf{y}) = (2(x - 1) - 2, 4y_1, 0)$$

$$\nabla_{xy} f(x, \mathbf{y}) = (y_1, 4(2y_1 - 4) + x, 4(2y_2 - 1))$$

The individual steps of the procedure are as follows.

Step 0: The initial rational point $(x, y_1, y_2)^1 = (4/3, 158/123, 5/82)$ (P_1 in Fig. 8.3) is computed by first considering the following relaxed problem where the lower-level objective function is omitted:

$$\begin{aligned} \min \quad & (x - 1)^2 + 2y_1^2 - 2x \\ \text{subject to} \quad & 4x + 5y_1 + 4y_2 \leq 12 \\ & -4x - 5y_1 + 4y_2 \leq -4 \\ & 4x - 4y_1 + 5y_2 \leq 4 \\ & -4x + 4y_1 + 5y_2 \leq 4 \\ & x \geq 0, y_i \geq 0, \quad i = 1, 2 \end{aligned}$$

The optimal solution is $(4/3, 1/3, 0)$ (P_0 in Fig. 8.3). By checking the optimality of the lower-level problem for x fixed at $4/3$, we obtain our first rational point $(x, y_1, y_2)^1 = (4/3, 158/123, 5/82)$, with $I(x) = \{1\}$ and $u_1 = 72/82$.

Step 1: For the current point, problem (8.39) is:

$$\begin{aligned} \min_d \quad & -4/3 d + 632/123 w_1 \\ \min_{w_1, w_2} \quad & 2dw_1 + 8w_1^2 + 8w_2^2 \\ \text{subject to} \quad & 5w_1 + 4w_2 \leq -4d \\ & -5w_1 - 4w_2 = 4d \\ & -1 \leq d \leq 1 \end{aligned}$$

The optimal solution is $(d, w_1, w_2)^1 = (1, -22/41, -27/82)$ with a negative value.

Step 2: With this direction, we compute the step length

$$\min_{t>0} \{F((x, y_1, y_2)^1 + t(d, w_1, w_2)^1) : (x, y_1, y_2)^1 + t(d, w_1, w_2)^1 \text{ feasible}\}$$

and find $t^* = 5/27$ to obtain the second point $(x, y_1, y_2)^2 = (41/27, 32/27, 0)$ (P_2 in Fig. 8.3) with $I(x) = \{1, 7\}$, $u_1 = 4$ and $u_7 = 0$. We return to Step 1.

Step 1: For the current point, problem (8.39) is:

$$\begin{aligned}
& \min_d -26/27 d + 128/27 w_1 \\
& \quad \min_{w_1, w_2} 2dw_1 + 8w_1^2 + 8w_2^2 \\
& \text{subject to} \quad 5w_1 + 4w_2 \leq -4d \\
& \quad -5w_1 - 4w_2 = 4d \\
& \quad -1 \leq d \leq 1, w_2 \geq 0
\end{aligned}$$

The optimal solution is $(d, w_1, w_2)^2 = (1, -4/5, 0)$ with objective value $-642/135 \leq 0$.

Step 2: With this direction, we compute the step length

$$\min_{t>0} \{F((x, y_1, y_2)^2 + t(d, w_1, w_2)^2) : (x, y_1, y_2)^2 + t(d, w_1, w_2)^2 \text{ feasible}\}$$

and find $t^* = 10/27$ to obtain the third point $(x, y_1, y_2)^3 = (17/9, 8/9, 0)$ (P_3 in Fig. 8.3), with $I(x) = \{1, 3, 7\}$, $u_1 = 1.4$, $u_3 = 0$ and $u_7 = 1.6$. We return to Step 1.

Step 1: For the current point, problem (8.39) is:

$$\begin{aligned}
& \min_d -2/9 d + 32/9 w_1 \\
& \quad \min_{w_1, w_2} 2dw_1 + 8w_1^2 + 8w_2^2 \\
& \text{subject to} \quad 5w_1 + 4w_2 \leq -4d \\
& \quad -4w_1 + 5w_2 \leq -4d \\
& \quad -7w_1 - 4w_2 = -8/9 d \\
& \quad -1 \leq d \leq 1, w_2 \geq 0
\end{aligned}$$

The optimal solution is $(d, w_1, w_2)^3 = (0, 0, 0)$ with objective value = 0 so the point $(x, y_1, y_2)^3 = (17/9, 8/9, 0)$ satisfies the necessary optimality conditions. As it turns out, this point is the global solution.

8.6 SUBGRADIENT DESCENT – BUNDLE METHOD

As shown in the previous section, an alternative way of tackling the general BLPP (8.3a)–(8.3d) is to view the upper-level problem as a standard nonlinear program (NLP) and use a descent approach to iteratively reduce the value of the leader's objective function (see [D3], [K6], [S8]). To compute a search direction, gradient information is obtained from the lower-level problem. In this section, we present the related background and one of two algorithms proposed by Falk and Liu [F1] for

implementing this idea. Their algorithms build directly on the theoretical results of Savard and Gauvin [S2].

The motivation for this work came from a number of implied and practical difficulties that have hampered existing approaches. In particular, for the Kuhn-Tucker (KT) approach presented in Section 8.2, a first major difficulty is that the KT conditions of the lower-level problem include complementary slackness which makes the constraints of the converted single nonlinear program very hard to solve. For the current class of descent methods, a second major difficulty stems from parametric nonlinear programming theory which tells us that even if the lower-level problem possesses highly satisfactory properties for each \mathbf{x} , such as strict convexity of the objective function, the strong second-order sufficient condition, and linear independence of the binding constraints, the feasible set of the converted nonlinear program is an arc defined by the KT conditions of the lower-level problem and generally will not be differentiable with respect to \mathbf{x} . This phenomenon might seriously degrade the efficiency of a gradient-based nonlinear solver. As for penalty-based methods, such as the one discussed in Section 8.3, slow convergence is the norm, especially when the problems are highly nonlinear. Hence these methods may have considerable difficulty in solving relatively large nonlinear problems.

Existing algorithms based on gradient approaches, though seemingly more efficient, assume the following properties:

- (1) $\mathbf{y}(\mathbf{x})$ is locally unique for each fixed \mathbf{x} ;
- (2) $\mathbf{y}(\mathbf{x})$ is continuously differentiable.

To ensure these properties hold, nondegenerate conditions consisting of the strong second-order sufficient condition, the linear independence condition, and strict complementary slackness are essentially needed at $\mathbf{y}(\mathbf{x})$ for each \mathbf{x} (see [F4]). It should be noted that the need to impose these conditions at $\mathbf{y}(\mathbf{x})$ for every \mathbf{x} is reinforced by the work of Springarn [S15] who showed that, loosely speaking, the nondegenerate conditions are of a generic type; i.e., the set of \mathbf{x} such that nondegenerate conditions do not hold at $\mathbf{y}(\mathbf{x})$ is a first category set of measure zero. As observed by deSilva and McCormick [D3] and Kolstad and Lasdon [K6], though, the nondifferentiability of $\mathbf{y}(\mathbf{x})$ is unavoidable but generally occurs at a limited number of points. Nevertheless, no convergence results have yet been obtained.

As mentioned, the implicitly defined function $\mathbf{y}(\mathbf{x})$ may not be everywhere differentiable; however, under the standard assumptions for the superlinear convergence of general nonlinear programming algorithms, i.e., the strong second-order sufficient condition and the linear independence condition, $\mathbf{y}(\mathbf{x})$ is piecewise differentiable and its generalized Jacobian has a rather desirable structure. A key fact observed by Falk and Liu was that the calculation of an element of its generalized Jacobian is computationally implementable after solving the lower-level problem, and that it can in turn be used to assist in the search for a solution to the upper-level problem.

This motivated their first algorithm called the *leader predominate algorithm*, which is an application of a bundle method of Schramm and Zowe [S3] to the upper-level problem. Under appropriate conditions, they show that the algorithm converges to regular points of the BLPP.

They further observed that if it is possible to determine a priori whether or not $\mathbf{y}(\mathbf{x})$ is twice continuously differentiable near a particular \mathbf{x} by introducing a second-order differentiability detector, then a quasi-Newton method can be employed to accelerate the search for a solution in the upper-level problem. Based on the idea of a second-order differentiability detector they propose a hybrid method, called the *adaptive leader predominate algorithm*, for solving the BLPP. The latter includes a quasi-Newton method within a bundle method that is adaptively called when the local structure of the current estimated solution warrants. The algorithm is shown to be globally convergent and locally superlinearly convergent to regular points of the BLPP provided that the nondegenerate conditions hold at these points. Nevertheless, Falk and Liu only consider local solutions in their work presented below. Obtaining global solutions can be a challenging task even for the unconstrained nonlinear BLPP.

8.6.1 Preliminaries

In this subsection, we introduce basic concepts and notation, and briefly review some known sensitivity results that will be used in the developments. Several optimality criteria for the BLPP are also stated. In the presentation, constraints of the form $G(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}$, $H(\mathbf{x}, \mathbf{y}) = \mathbf{0}$ are omitted from model (8.2) and $h(\mathbf{x}, \mathbf{y}) = \mathbf{0}$ from model (8.1), but their inclusion is straightforward.

Definition 8.6.1 We say that a point (\mathbf{x}, \mathbf{y}) is a *semi-local solution* (or *semi-solution*) to the BLPP if $(\mathbf{x}, \mathbf{y}) \in S = \{(\mathbf{x}, \mathbf{y}) : g(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}\}$ and \mathbf{y} is a local solution (or global solution) to (8.1) with \mathbf{x} fixed. We say that a point \mathbf{y} is a local solution (or solution) with respect to \mathbf{x} if \mathbf{y} is a local solution (or global solution) to (8.1) with \mathbf{x} fixed.

Definition 8.6.2

1. A point $(\mathbf{x}^*, \mathbf{y}^*)$ is said to be a *(strict) local solution* to the BLPP if
 - (a) $(\mathbf{x}^*, \mathbf{y}^*)$ is a semi-local solution;
 - (b) there exists a neighborhood N of $(\mathbf{x}^*, \mathbf{y}^*)$ such that $(F(\mathbf{x}^*, \mathbf{y}^*) < F(\mathbf{x}, \mathbf{y}))$ $F(\mathbf{x}^*, \mathbf{y}^*) \leq F(\mathbf{x}, \mathbf{y})$ for all semi-local solutions $(\mathbf{x}, \mathbf{y}) \in N$.
2. A point $(\mathbf{x}^*, \mathbf{y}^*)$ is said to be a *solution* to the BLPP if
 - (a) $(\mathbf{x}^*, \mathbf{y}^*)$ is a semi-solution;
 - (b) $F(\mathbf{x}^*, \mathbf{y}^*) \leq F(\mathbf{x}, \mathbf{y})$ for all semi-solutions $(\mathbf{x}, \mathbf{y}) \in S$.

Definition 8.6.3 A point $(\mathbf{x}^*, \mathbf{y}^*)$ is said to be an *isolated local solution* to the BLPP if there exists a neighborhood N of $(\mathbf{x}^*, \mathbf{y}^*)$ such that it is the unique local solution to the BLPP in N .

Define for each $u \in R^1$, $u^+ \triangleq \max\{u, 0\}$ and $u^- \triangleq \min\{u, 0\}$. For a fixed \mathbf{x} , the Lagrangian associated with (8.1) based on normal maps of Robinson [R2] is defined by

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{u}) = f(\mathbf{x}, \mathbf{y}) + \mathbf{u}^+ g(\mathbf{x}, \mathbf{y})$$

where $\mathbf{u} \in R^q$ is a row vector of multipliers. If \mathbf{y} is a local solution of (8.1) with fixed \mathbf{x} and an appropriate constraint qualification holds at $\mathbf{y}(\mathbf{x})$, then it follows that the Kuhn-Tucker conditions hold at \mathbf{y} ; i.e., there exists a $\mathbf{u} \in R^q$ such that $(\mathbf{x}, \mathbf{y}, \mathbf{u})$ satisfies the following normal equations:

$$\begin{aligned}\nabla_{\mathbf{y}} \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{u}) &= \mathbf{0} \\ g(\mathbf{x}, \mathbf{y}) - \mathbf{u}^- &= \mathbf{0}\end{aligned}\tag{8.40}$$

As we know from Chapter 4, system (8.40) plays a key role in mathematical programming. It readily follows from the definition that a necessary condition for $(\mathbf{x}^*, \mathbf{y}^*)$ to be a local solution to the BLPP is that there exists a $\mathbf{u} \in R^q$ such that $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{u}^*)$ is feasible to the problem (see [B10])

$$\begin{array}{ll}\min_{\mathbf{x}, \mathbf{y}, \mathbf{u}} & F(\mathbf{x}, \mathbf{y}) \\ \text{subject to} & \nabla_{\mathbf{y}} \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{u}) = \mathbf{0} \\ & g(\mathbf{x}, \mathbf{y}) - \mathbf{u}^- = \mathbf{0}\end{array}$$

Denote the active (inequality) set of indices at (\mathbf{x}, \mathbf{y}) by $I(\mathbf{x}, \mathbf{y})$; i.e., $I(\mathbf{x}, \mathbf{y}) = \{i \in \{1, \dots, q\} : g_i(\mathbf{x}, \mathbf{y}) = 0\}$. For each $\mathbf{u} \in R^q$, define $I_+(\mathbf{u}) = \{i : u_i > 0\}$, $I_0(\mathbf{u}) = \{i : u_i = 0\}$, and $I_-(\mathbf{u}) = \{i : u_i \leq 0\}$. For fixed \mathbf{x} , the following regularity and second-order conditions for (8.1) will be used.

- (a) The *linear independence* condition (LI) holds at \mathbf{y} if $\nabla_{\mathbf{y}} g_i(\mathbf{x}, \mathbf{y})$, $i \in I(\mathbf{x}, \mathbf{y})$ are linearly independent.
- (b) The *strict complementary slackness* condition (SCS) holds at \mathbf{y} with respect to \mathbf{u} if $u_i > 0$ for every $i \in I(\mathbf{x}, \mathbf{y})$.
- (c) The *strong second-order sufficient condition* (SSOSC) holds at \mathbf{y} with \mathbf{u} if $\mathbf{d}^T \nabla_{\mathbf{y}}^2 \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{u}) \mathbf{d} > 0$, for all $\mathbf{d} \neq 0$, such that $\mathbf{d}^T \nabla_{\mathbf{y}} g_i(\mathbf{x}, \mathbf{y}) = 0$ if $u_i > 0$.

Sensitivity Results

Several results from sensitivity theory, as adapted for bilevel programming, are presented below. The first says that under the nondegenerate conditions the local optimal strategy of the follower is locally unique and continuously differentiable with respect to the decision taken by the leader.

Proposition 8.6.1 ([F3]) Suppose KT, SSOSC, SCS and LI hold at \mathbf{y}^0 with multipliers \mathbf{u}^0 for (8.1) with $\mathbf{x} = \mathbf{x}^0$, and that the problem functions f and \mathbf{g} are C^3 in a neighborhood of $(\mathbf{x}^0, \mathbf{y}^0)$. Then

- (a) for \mathbf{x} in a neighborhood of \mathbf{x}^0 , there exists a unique twice continuously differentiable function $\mathbf{z}(\mathbf{x}) = (\mathbf{y}(\mathbf{x}), \mathbf{u}(\mathbf{x}))$ satisfying KT, SSOSC, SCS and LI at $\mathbf{y}(\mathbf{x})$ with $\mathbf{u}(\mathbf{x})$ for (8.1) such that $\mathbf{z}(\mathbf{x}^0) = (\mathbf{y}^0, \mathbf{u}^0)$ and $\mathbf{y}(\mathbf{x})$ is a unique local solution of (8.1) at \mathbf{x} ;
- (b) the Jacobian $Q(\mathbf{x})$ of the following system

$$\begin{aligned}\nabla_{\mathbf{y}} \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{u}) &= \mathbf{0} \\ g(\mathbf{x}, \mathbf{y}) - \mathbf{u}^- &= \mathbf{0}\end{aligned}\tag{8.41}$$

with respect to (\mathbf{y}, \mathbf{u}) evaluated at \mathbf{x} is locally nonsingular and

$$\nabla_{\mathbf{x}} \mathbf{z}(\mathbf{x}) = Q(\mathbf{x})^{-1} J(\mathbf{x})\tag{8.42}$$

where $-J(\mathbf{x})$ is the Jacobian of the system (8.41) with respect to \mathbf{x} .

Definition 8.6.4 We say that a function θ from a finite-dimensional space X into another finite dimensional space Z is twice directionally differentiable at a point \mathbf{z}^0 along a direction $\mathbf{d} \neq \mathbf{0}$ if the following limit exists

$$D^2\theta(\mathbf{z}^0; \mathbf{d}) = \lim_{t \downarrow 0} \frac{1}{t} [D\theta(\mathbf{z}^0 + t\mathbf{d}; \mathbf{d}) - D\theta(\mathbf{z}^0; \mathbf{d})]$$

where $D\theta(\mathbf{z}^0 + t\mathbf{d}; \mathbf{d})$ and $D\theta(\mathbf{z}^0; \mathbf{d})$ are the first directional derivatives of θ at $\mathbf{z}^0 + t\mathbf{d}$ and \mathbf{z}^0 along \mathbf{d} , respectively.

A main result derived by Liu [L4] is that under some appropriate conditions, the perturbed solution vector $\mathbf{z}(\mathbf{x})$ can be continuously selected locally from a set of finite C^1 functions $\{\mathbf{z}^k\}$; i.e., there exists a neighborhood U of \mathbf{x} such that for any $\mathbf{x} \in U$, $\mathbf{z}(\mathbf{x}) = \mathbf{z}'(\mathbf{x})$ for some $\mathbf{z}' \in \{\mathbf{z}^k\}$. It is said that $\{\mathbf{z}^k\}$ is a *selection base* of \mathbf{z} at \mathbf{x}^0 if for any $\mathbf{z}' \in \{\mathbf{z}^k\}$, there is a sequence $\{\mathbf{x}^j\}$ such that $\mathbf{z}(\mathbf{x}^j) = \mathbf{z}'(\mathbf{x}^j)$ and $\mathbf{x}^j \rightarrow \mathbf{x}^0$, $\mathbf{x}^j \neq \mathbf{x}^0$.

The following proposition specializes several results from sensitivity theory to the bilevel programming problem. The significant improvement over Proposition 8.6.1 is that SCS is relaxed.

Proposition 8.6.2 ([J3], [L4]) Suppose KT, SSOSC and LI hold at \mathbf{y}^0 with multipliers \mathbf{u}^0 for (8.1) with $\mathbf{x} = \mathbf{x}^0$, and that the problem functions f and \mathbf{g} are C^3 in a neighborhood of $(\mathbf{x}^0, \mathbf{y}^0)$. Then

- (a) for \mathbf{x} in a neighborhood of \mathbf{x}^0 , there exists a unique continuous function $\mathbf{z}(\mathbf{x}) = (\mathbf{y}(\mathbf{x}), \mathbf{u}(\mathbf{x}))$ satisfying KT, SSOSC, and LI at $\mathbf{y}(\mathbf{x})$ with $\mathbf{u}(\mathbf{x})$ for (8.1) such that $\mathbf{z}(\mathbf{x}^0) = (\mathbf{y}^0, \mathbf{u}^0)$ and $\mathbf{y}(\mathbf{x})$ is a locally unique local solution of (8.1);

- (b) $\mathbf{z}(\cdot)$ is locally Lipschitz near \mathbf{x}^0 and directionally differentiable at \mathbf{x}^0 along any direction $\mathbf{d} \neq \mathbf{0}$; furthermore, there exists a selection base $\{\mathbf{z}^k\}$ for \mathbf{z} at \mathbf{x}^0 and each \mathbf{z}^k is continuously differentiable at \mathbf{x}^0 and is locally defined as the solution of the following system

$$\begin{aligned}\nabla_{\mathbf{y}} \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{u}^I) &= \mathbf{0} \\ \mathbf{g}^I(\mathbf{x}, \mathbf{y}) &= \mathbf{0} \\ \mathbf{g}^{I^c}(\mathbf{x}, \mathbf{y}) - \mathbf{u}^{I^c} &= \mathbf{0}\end{aligned}\tag{8.43}$$

for some index set I such that $I_+(\mathbf{u}^0) \subset I \subset \{1, \dots, q\}$, where $I^c = \{1, \dots, q\} \setminus I$, and $\mathbf{g}^I, \mathbf{u}^I$ consist of functions g_i , variables u_i , $i \in I$, respectively;

- (c) the function $\mathbf{z}(\cdot)$ is twice directionally differentiable at \mathbf{x}^0 along any direction $\mathbf{d} \neq \mathbf{0}$; furthermore, the first and second directional derivatives $D\mathbf{z}(\mathbf{x}^0; \mathbf{d})$ and $D^2\mathbf{z}(\mathbf{x}^0; \mathbf{d})$ are Lipschitzian as functions of the direction \mathbf{d} .

8.6.2 Optimality Conditions and Stability of Local Solutions

In the presence of nonsingularity assumptions common to nonlinear programming, Falk and Liu have shown that the BLPP can be implicitly converted to a nonsmooth unconstrained minimization problem. Necessary and sufficient conditions for local solutions of the BLPP then follow as do some preliminary stability and sensitivity results. The relevant assumptions and theorems are presented below. Proofs are contained in the original paper.

Differentiability Assumption (A1): The function F is in C^2 , and functions f and \mathbf{g} are C^3 .

Proposition 8.6.2 says that for a fixed strategy \mathbf{x}^0 of the leader if SSOSC and LI hold at a local optimal strategy \mathbf{y}^0 of the follower, then for each strategy \mathbf{x} near \mathbf{x}^0 there is a locally unique optimal strategy \mathbf{y} near \mathbf{y}^0 . The invocation of SSOSC and LI has played a key role in nonlinear programming. Actually, they are the standard assumptions imposed when studying the convergence of most algorithms designed to solve (8.1). In the remainder of this section we shall make the following blanket assumption.

Strong Regularity Assumption (A2): For each fixed \mathbf{x} , problem (8.1) has at least a local solution, and SSOSC and LI are satisfied at all local solutions to (8.1) at \mathbf{x} .

Now suppose the leader makes a decision \mathbf{x}^0 and \mathbf{y}^0 is the local rational response of the follower. It follows from Proposition 8.6.2 that there exists a $\mathbf{y}(\mathbf{x})$ defined in the

neighborhood U of \mathbf{x}^0 such that $\mathbf{y}(\mathbf{x}^0) = \mathbf{y}^0$ and $\mathbf{y}(\mathbf{x})$ is the local optimal strategy of the follower with respect to the strategy \mathbf{x} in U . Hence we can project the set of semi-local solutions into \mathbf{x} -space and define $PF(\mathbf{x}) \triangleq F(\mathbf{x}, \mathbf{y}(\mathbf{x}))$ for $\mathbf{x} \in U$; then the BLPP locally reduces to the following unconstrained minimization problem

$$\min_{\mathbf{x}} PF(\mathbf{x})$$

What is important is that if $(\mathbf{x}^0, \mathbf{y}^0)$ is not yet optimal, the above optimization problem could tell us how to proceed locally towards optimality. From Proposition 8.6.2 we know that even if the function PF is not differentiable, it is locally Lipschitz and its subdifferential has a simple structure. Also, the first- and second-order directional derivatives of $PF(\mathbf{x})$ at \mathbf{x}^0 along any direction $\mathbf{d} \neq \mathbf{0}$ are always available. Simple calculation yields, that for any direction $\mathbf{d} \neq \mathbf{0}$,

$$DPF(\mathbf{x}^0; \mathbf{d}) = \frac{\partial F}{\partial \mathbf{x}} \mathbf{d} + \frac{\partial F}{\partial \mathbf{y}} D\mathbf{y}(\mathbf{x}^0; \mathbf{d})$$

and

$$\begin{aligned} D^2 PF(\mathbf{x}^0; \mathbf{d}) &= \mathbf{d}^T \frac{\partial^2 F}{\partial \mathbf{x}^2} \mathbf{d} + 2\mathbf{d}^T \frac{\partial^2 F}{\partial \mathbf{x} \partial \mathbf{y}} D\mathbf{y}(\mathbf{x}^0; \mathbf{d}) \\ &\quad + D\mathbf{y}(\mathbf{x}^0; \mathbf{d})^T \frac{\partial^2 F}{\partial \mathbf{y}^2} D\mathbf{y}(\mathbf{x}^0; \mathbf{d}) + \frac{\partial F}{\partial \mathbf{y}} D^2 \mathbf{y}(\mathbf{x}^0; \mathbf{d}) \end{aligned}$$

Consequently, the following optimality conditions can be deduced.

Theorem 8.6.1 (Necessary conditions) If $(\mathbf{x}^*, \mathbf{y}^*)$ is a local solution to the BLPP, then

1. $\mathbf{0} \in \partial PF(\mathbf{x}^*)$;
2. For any direction $\mathbf{d} \neq \mathbf{0}$ we have
 - (a) $DPF(\mathbf{x}^*; \mathbf{d}) \geq 0$;
 - (b) if $DPF(\mathbf{x}^*; \mathbf{d}) = 0$, then $D^2 PF(\mathbf{x}^*; \mathbf{d}) \geq 0$.

Analogous to the case of smooth programming, we define the regular points of the BLPP to be the points that satisfy the necessary first-order optimality conditions.

Definition 8.6.5 Suppose $(\mathbf{x}^*, \mathbf{y}^*)$ is a semi-local solution to the BLPP. We say that $(\mathbf{x}^*, \mathbf{y}^*)$ is a stationary point or a *regular point of first category* to the BLPP if $\mathbf{0} \in \partial PF(\mathbf{x}^*)$, and a *regular point of second category* to the BLPP if for any direction $\mathbf{d} \neq \mathbf{0}$, $DPF(\mathbf{x}^*; \mathbf{d}) \geq 0$.

A regular point of second category must be a regular point of first category but not vice versa in general. To see this, recall [S10] that for a locally Lipschitz function $c : R^t \rightarrow R^1$ at $\mathbf{z} \in R^t$ one has

1. the generalized directional derivative $c^0(\mathbf{z}; \mathbf{v}) = \max\{\boldsymbol{\xi}^T \mathbf{v} : \boldsymbol{\xi} \in \partial c(\mathbf{z})\}$ for all directions $\mathbf{v} \in R^t$;
2. $c^0(\mathbf{z}; \mathbf{v}) \leq Dc(\mathbf{z}; \mathbf{v})$ for all directions $\mathbf{v} \in R^t$.

Even a regular point of second category to the BLPP is not necessarily a local solution. To ensure optimality, additional conditions must be imposed.

Theorem 8.6.2 (Sufficient conditions) Let $(\mathbf{x}^*, \mathbf{y}^*) \in S$. Assume that for any direction $\mathbf{d} \neq \mathbf{0}$ we have

- (a) $DPF(\mathbf{x}^*; \mathbf{d}) > \mathbf{0}$ or
- (b) $DPF(\mathbf{x}^*; \mathbf{d}) = \mathbf{0}$ and $D^2PF(\mathbf{x}^*; \mathbf{d}) > 0$.

Then $(\mathbf{x}^*, \mathbf{y}^*)$ is a strict local solution to the BLPP. Furthermore, there exist a neighborhood N of $(\mathbf{x}^*, \mathbf{y}^*)$ and a $\lambda > 0$ such that if $(\mathbf{x}, \mathbf{y}) \in N$ and (\mathbf{x}, \mathbf{y}) is a semi-local solution, then

$$F(\mathbf{x}, \mathbf{y}) \geq F(\mathbf{x}^*, \mathbf{y}^*) - \lambda \|(\mathbf{x}, \mathbf{y}) - (\mathbf{x}^*, \mathbf{y}^*)\|^2$$

We now turn our attention to the stability of the BLPP. Consider the following parametric bilevel programming problem $BLPP(\epsilon)$:

$$\begin{aligned} \min_{\mathbf{x}} \quad & F(\mathbf{x}, \mathbf{y}, \epsilon) \\ \min_{\mathbf{y}} \quad & f(\mathbf{x}, \mathbf{y}, \epsilon) \\ \text{subject to} \quad & (\mathbf{x}, \mathbf{y}) \in S(\epsilon) \end{aligned}$$

and the feasible set is defined by $S(\epsilon) = \{(\mathbf{x}, \mathbf{y}) : \mathbf{g}(\mathbf{x}, \mathbf{y}, \epsilon) \leq \mathbf{0}\}$, $F, f : R^n \times R^m \times R^t \rightarrow R^1$, $\mathbf{g} : R^n \times R^t \rightarrow R^q$. For some $\epsilon^* \in R^t$, $F(\mathbf{x}, \mathbf{y}, \epsilon^*) = F(\mathbf{x}, \mathbf{y})$, $f(\mathbf{x}, \mathbf{y}, \epsilon^*) = f(\mathbf{x}, \mathbf{y})$, and $\mathbf{g}(\mathbf{x}, \mathbf{y}, \epsilon^*) = \mathbf{g}(\mathbf{x}, \mathbf{y})$. It is assumed that the functions F, f and \mathbf{g} as well as their partial derivatives are continuous in $(\mathbf{x}, \mathbf{y}, \epsilon)$, and that for each ϵ assumptions (A1) and (A2) hold for the $BLPP(\epsilon)$.

The theorem below states that if the sufficient conditions hold at a local solution of the BLPP, then the perturbed BLPP is locally solvable and the perturbed solution map is upper semicontinuous.

Theorem 8.6.3 Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a local solution to the $BLPP(\epsilon^*)$ and assume that sufficient conditions hold at $(\mathbf{x}^*, \mathbf{y}^*)$ for $BLPP(\epsilon^*)$. Then there exist a neighborhood N of $(\mathbf{x}^*, \mathbf{y}^*)$ and a neighborhood E of ϵ^* such that if we define the localized local solution map as $\text{sol}_N(\epsilon) \triangleq \{(\mathbf{x}, \mathbf{y}) \in \text{cl}(N) : (\mathbf{x}, \mathbf{y}) \text{ is a local solution to } BLPP(\epsilon)\}$ for $\epsilon \in E$, where $\text{cl}(\cdot)$ is the closure, then $\text{sol}_N(\cdot) \neq \emptyset$ for each $\epsilon \in E$ and is upper semicontinuous in E .

In light of Theorem 8.6.3, we can define a restricted optimal-value function that is associated with an appropriate choice of a neighborhood N of $(\mathbf{x}^*, \mathbf{y}^*)$ as follows:

$$w_N(\varepsilon) \triangleq \begin{cases} \min \{F(\mathbf{x}, \mathbf{y}, \varepsilon) : (\mathbf{x}, \mathbf{y}) \in \text{sl}(\varepsilon) \cap \text{cl}(N)\}, & \text{if } \text{sl}(\varepsilon) \neq \emptyset \\ +\infty, & \text{if } \text{sl}(\varepsilon) = \emptyset \end{cases}$$

where $\text{sl}(\varepsilon)$ is the set of semi-local solutions to the BLPP(ε). It is interesting that in the current setting the restricted optimal-value function $w_N(\varepsilon)$ is actually differentiable with respect to the perturbation parameter ε .

Theorem 8.6.4 Under the previously stated conditions, assume that the functions involved are partially continuously differentiable with respect to ε . Then $w_N(\varepsilon)$ is differentiable at ε^* and its derivative is

$$\nabla_{\varepsilon} w_N(\varepsilon^*) = \nabla_{\varepsilon} F(\mathbf{x}^*, \mathbf{y}^*, \varepsilon^*).$$

8.6.3 Leader Predominate Algorithm

As mentioned, Falk and Liu proposed two algorithms for solving the BLPP in the above setting. The leader predominate algorithm essentially employs a trust region bundle method for the upper-level problem using subgradient information from the lower-level problem. The adaptive leader predominate algorithm is a hybrid method for the upper-level problem designed to combine the strong points of bundle methods and quasi-Newton methods. It tries to make the best use of available gradient information from the lower-level problem based on the idea of a twice differentiability detector. The authors give convergence results for both algorithms and discuss stability-related issues. The first algorithm will now be presented. The interested reader is referred to the original paper for the proofs as well as the details of the second algorithm.

The idea of leader predominate algorithm for solving bilevel programs is as follows. Suppose at the k th iterative step, the leader's strategy is \mathbf{x}^k and the follower's response is \mathbf{y}^k ; i.e., \mathbf{y}^k is a local solution with respect to \mathbf{x}^k . Assume that the role of the leader is predominate in the sequential decision making process. This means that whatever strategy \mathbf{x} the leader takes, the follower has to respond immediately to it with a strategy that is a local solution with respect to \mathbf{x} . In these circumstances, the leader may be able to utilize the information available near the semi-local solution $(\mathbf{x}^k, \mathbf{y}^k)$ to choose a better strategy \mathbf{x}^{k+1} such that the value of the cost function F decreases sufficiently. If the value of F can be decreased from iteration to iteration, the process should reach a locally optimal pair, at least in the limit.

To motivate the algorithm, suppose we have reached a semi-local solution $(\mathbf{x}^k, \mathbf{y}^k)$ at the k th iteration. Recall the blanket assumptions (A1) and (A2). If the leader moves

from \mathbf{x}^k along a direction \mathbf{d} , then the rate of change of the value of the leader's cost function is

$$DPF(\mathbf{x}^k; \mathbf{d}) = \frac{\partial F}{\partial \mathbf{x}} \mathbf{d} + \frac{\partial F}{\partial \mathbf{y}} D\mathbf{y}(\mathbf{x}^k; \mathbf{d})$$

since it is assumed that the follower must respond to the leader's move immediately. Therefore, a possible strategy is to move forward along the steepest descent direction, i.e., choose \mathbf{d}^k such that it solves

$$\min_{\|\mathbf{d}\| \leq 1} \left\{ DPF(\mathbf{x}^k; \mathbf{d}) = \frac{\partial F}{\partial \mathbf{x}} \mathbf{d} + \frac{\partial F}{\partial \mathbf{y}} D\mathbf{y}(\mathbf{x}^k; \mathbf{d}) \right\}$$

where the minimization is taken with respect to some norm $\|\cdot\|$. Due to the non-differentiability nature of PF , however, this strategy may not work well since it may get stuck near a nonoptimal kink (see [L3]).

As an alternative, one of the most promising ways to handle a nondifferentiable function in an optimization problem is the so called bundle method. All such methods bear two distinctive features (e.g., see [S3]):

- (1) They exploit previous iterations by gathering the subgradient information into a bundle, build up a model of the function in question based on the current bundle, and compute an estimate of the solution;
- (2) If, due to the kinky structure of the nondifferentiable function, the current model is not yet good enough, more subgradient information is collected to produce a better model.

Because $PF(\mathbf{x})$ is locally Lipschitz, all bundle methods are theoretically applicable to the optimization of PF . Their adaptation, though, depends on whether or not the computation of a subgradient $\xi \in \partial PF(\mathbf{x})$ can be carried out efficiently. Based on Proposition 8.6.2, the function $\mathbf{y}(\mathbf{x})$, and in turn, $PF(\mathbf{x})$, indeed have a desirable structure which will be investigated first.

Let $G : R^t \rightarrow R^1$, $G_i : R^t \rightarrow R^1$, $i = 1, \dots, r$ be functions in C^0 and C^1 , respectively. Define an index mapping Λ such that for any $\mathbf{w} \in R^t$ the set $\Lambda(\mathbf{w})$ is a nonempty subset of $\{1, \dots, r\}$ satisfying

$$\begin{aligned} G_k(\mathbf{w}) &= G_i(\mathbf{w}) \text{ for any } k, i \in \Lambda(\mathbf{w}), \\ G_k(\mathbf{w}) &\neq G_j(\mathbf{w}) \text{ for any } k \in \Lambda(\mathbf{w}), \text{ any } j \notin \Lambda(\mathbf{w}) \end{aligned}$$

We say that G is a piecewise smooth function if for any $\mathbf{w} \in R^t$

$$G(\mathbf{w}) = G_i(\mathbf{w}) \text{ for some } i \in \Lambda(\mathbf{w}).$$

Such an index mapping $\Lambda(\cdot)$ is called a *selection* of G . For a piecewise smooth function G , since it is locally Lipschitz, we can define a mapping Λ^B from R^t to a subset of

$\{1, \dots, r\}$ as follows:

$$\begin{aligned}\Lambda^B(\mathbf{w}) = & \{i : \text{there exist } \mathbf{w}^j \rightarrow \mathbf{w}, \mathbf{w}^j \in D_G, \\ & \text{such that } G(\mathbf{w}^j) = G_i(\mathbf{w}^j), \nabla G(\mathbf{w}^j) = \nabla G_i(\mathbf{w}^j)\}\end{aligned}$$

where D_G denotes the set of points where G is differentiable. For any $\mathbf{w} \in R^t$, let

$$B^\delta(\mathbf{w}) = \{\lim \nabla G(\mathbf{w}^j) : \mathbf{w}^j \rightarrow \mathbf{w}, \mathbf{w}^j \in D_G\},$$

which is the so-called *B-subdifferential* of G at \mathbf{w} . It is well known that Clarke's subdifferential $\partial G(\mathbf{w}) \triangleq \text{conv} B^\delta(\mathbf{w})$.

Lemma 8.6.1 In the above setting, we have $B^\delta(\mathbf{w}) = \{\nabla G_i(\mathbf{w}) : i \in \Lambda^B(\mathbf{w})\}$.

Note that the first two equations in the system (8.43) are actually the KT conditions of the following equality constrained program parameterized in \mathbf{x}

$$\begin{aligned}\min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \\ \text{subject to } (\mathbf{x}, \mathbf{y}) \in S^I\end{aligned}$$

where $S^I = \{(\mathbf{x}, \mathbf{y}) : g^I(\mathbf{x}, \mathbf{y}) = 0\}$. Given Assumptions (A1) and (A2), this program uniquely defines a C^1 solution function $\mathbf{y}^I(\mathbf{x})$. Therefore, if we label the set of C^1 functions

$$\{\mathbf{y}^I(\cdot) : \text{for any } I \text{ such that } I \subset \{1, \dots, q\}\}$$

by an index set $\{1, \dots, r\}$; i.e., $y_i(\cdot)$ is labeled $\mathbf{y}^I(\cdot)$ for some I , and define

$$\Lambda(\mathbf{x}) = \{i : \mathbf{y}(\mathbf{x}) = y_i(\mathbf{x}), i = 1, \dots, r\}$$

then $\Lambda(\cdot)$ is a selection of $\mathbf{y}(\cdot)$. Because $\mathbf{y}(\mathbf{x})$ is piecewise differentiable, by the chain rule $PF(\mathbf{x})$ is also piecewise differentiable. This leads to the essential problem of how to compute a subgradient of PF .

Computing a subgradient of a general Lipschitz function is by any measure an extremely difficult task. Nevertheless, the structure of $\mathbf{y}(\cdot)$ makes these calculations practical for PF . Note that the main advantage of bundle methods is that they do not require the complete set of subdifferentials; only a subgradient at a trial point is needed. In what follows, it will be shown how to calculate a sub-Jacobian of $\mathbf{y}(\cdot)$ at a point \mathbf{x}^0 . A subgradient of PF at \mathbf{x}^0 can then be obtained by the chain rule. Let $(\mathbf{y}^0, \mathbf{u}^0)$ be a solution of (8.1) at $\mathbf{x} = \mathbf{x}^0$. Take a direction \mathbf{d} , and perturb \mathbf{x} at \mathbf{x}^0 along \mathbf{d} . It is well known that the directional derivative $Dz(\mathbf{x}^0; \mathbf{d})$ of $z(\cdot)$ at \mathbf{x}^0 in direction \mathbf{d} is the solution of the following quadratic program

$$\begin{aligned}\min_{\mathbf{w}} & \mathbf{w}^T \nabla_{\mathbf{y}}^2 \mathcal{L} \mathbf{w} + 2\mathbf{w}^T \nabla_{\mathbf{y}}^2 \mathcal{L} \mathbf{d} \\ \text{subject to } & \nabla_{\mathbf{y}} g_i \mathbf{w} + \nabla_{\mathbf{x}} g_i \mathbf{d} = 0, \quad i \in I_+(\mathbf{u}^0) \\ & \nabla_{\mathbf{y}} g_i \mathbf{w} + \nabla_{\mathbf{x}} g_i \mathbf{d} \leq 0, \quad i \in I_-(\mathbf{u}^0)\end{aligned}\tag{8.44}$$

Let the solution to (8.44) be $(\mathbf{w}, \boldsymbol{\mu})$ where $\boldsymbol{\mu}$ is the vector of KT multipliers. The following first-order approximation of \mathbf{z} at \mathbf{x}^0 along \mathbf{d} is

$$\mathbf{z}(\mathbf{x}^0 + t\mathbf{d}) = \mathbf{z}(\mathbf{x}^0) + t(\mathbf{w}, \boldsymbol{\mu}) + o(t). \quad (8.45)$$

Thus

$$\mathbf{u}(\mathbf{x}^0 + t\mathbf{d}) = \mathbf{u}^0 + t\boldsymbol{\mu} + o(t) \quad (8.46)$$

Given that $\mathbf{z}(\cdot)$ is comprised of pieces of C^0 functions, we are particularly interested in the multipliers \mathbf{u} because they tell us in which piece $\mathbf{z}(\mathbf{x}^0 + t\mathbf{d})$ is and whether or not $\mathbf{z}(\cdot)$ is differentiable at $\mathbf{x}^0 + t\mathbf{d}$. Let $I(\mathbf{d}) = I_+(\mathbf{u}^0) \cup \{i : \mu_i > 0\}$. Note that for any \mathbf{d} , LI and SSOSC always hold at the solution $(\mathbf{w}, \boldsymbol{\mu})$ of (8.44). Suppose in addition that SCS holds at $(\mathbf{w}, \boldsymbol{\mu})$. Since (8.44) is homogeneous, if we replace the parameter \mathbf{d} with $t\mathbf{d}$ in (8.44), the resultant program has a solution $t(\mathbf{w}, \boldsymbol{\mu})$. Also, SCS still holds at $t(\mathbf{w}, \boldsymbol{\mu})$ for (8.44) with $t\mathbf{d}$. Now let $t \downarrow 0$. We only consider the binding constraints at \mathbf{x}^0 because only they may cause $\mathbf{z}(\cdot)$ to be locally nondifferentiability. For any $i \in I_0(\mathbf{u}^0)$, if $0 \in \text{cl}\{t > 0 : g_i(\mathbf{x}^0 + t\mathbf{d}) = 0\}$, then the i th constraint in (8.44) must be binding. By SCS we have $\mu_i > 0$ and then from (8.46), $u_i(\mathbf{x}^0 + t\mathbf{d}) > 0$ for t small enough. This means that SCS holds at $\mathbf{z}(\mathbf{x}^0 + t\mathbf{d})$ for sufficiently small t , so $\mathbf{z}(\cdot)$ is differentiable at $\mathbf{x}^0 + t\mathbf{d}$ for such t . Therefore, we have $I(\mathbf{d}) \in \Lambda^B(\mathbf{x}^0)$, so that $\nabla \mathbf{y}_{I(\mathbf{d})}(\mathbf{x}^0) \in B^0(\mathbf{x}^0)$. Also, from Proposition 8.6.1 one has

$$\nabla \mathbf{z}_{I(\mathbf{d})}(\mathbf{x}^0) = (Q_{I(\mathbf{d})}(\mathbf{x}^0))^{-1} J_{I(\mathbf{d})}(\mathbf{x}^0) \quad (8.47)$$

where $Q_{I(\mathbf{d})}(\mathbf{x}^0)$ and $-J_{I(\mathbf{d})}(\mathbf{x}^0)$ are the Jacobians of the following system

$$\nabla_{\mathbf{y}} \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{u}^{I(\mathbf{d})}) = \mathbf{0}$$

$$g^{I(\mathbf{d})}(\mathbf{x}, \mathbf{y}) - \mathbf{u}^{I(\mathbf{d})} = \mathbf{0}$$

with respect to $(\mathbf{y}, \mathbf{u}^{I(\mathbf{d})})$ and \mathbf{x} , respectively. Given that only a sub-Jacobian of $\mathbf{y}(\cdot)$ at \mathbf{x}^0 is needed, the matrices in (8.47) can be partitioned into blocks and the calculation simplified. Thus we can obtain a sub-Jacobian of $\mathbf{y}(\cdot)$ at \mathbf{x}^0 .

On the other hand, if SCS does not hold at $(\mathbf{w}, \boldsymbol{\mu})$, another direction \mathbf{d} can be chosen and the above procedure repeated. Because the set of directions for which SCS is not satisfied for (8.44) at its solution has measure zero, normally just one trial will result in a subgradient of PF . This resolves the problem of computing a subgradient of PF . For more detail, interested readers may consult [P2] which presents an indepth study of the structure of piecewise smooth functions.

A sketch of the leader predominate algorithm is given below. In the presentation, it is assumed that the following routines are in hand.

NLP: A standard nonlinear programming solver that is based on the sequential quadratic technique using an exact penalty function (see [F6]). This routine will

be used to solve (8.1) for a fixed \mathbf{x} . Specifically, if we input \mathbf{x} to NLP , it will produce a solution $(\mathbf{y}(\mathbf{x}), \mathbf{u}(\mathbf{x}))$ for (8.1).

SGF: A subgradient–finding routine that will compute a subgradient of PF at a point \mathbf{x} according to the procedure stated above for computing $\partial\mathbf{y}(\cdot)$ coupled with the chain rule for subdifferentiation. The first trial direction is chosen to be the current direction in the bundle method.

Leader Predominate Algorithm (application of bundle method)

- Step 0 (Initialization) Choose a starting semi-local solution $(\mathbf{x}^1, \mathbf{y}^1)$, a value for the parameter $\varepsilon > 0$, and an upper bound $J_{\max} \geq 3$ for $|J_k|$. Compute $PF(\mathbf{x}^1)$, $\xi^1 \in \partial PF(\mathbf{x}^1)$ and put $\mathbf{w}^1 \leftarrow \mathbf{x}^1$, $J_1 \leftarrow \{1\}$ and $k \leftarrow 1$.
- Step 1 (Inner iteration) Computer semi-local solution $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1})$ with NLP and ξ^{k+1} with *SGF* as in (3.3) of Schramm and Zowe [S3], or realize that \mathbf{x}^{k+1} is ε -optimal (in which case stop).
- Step 2 (Bundle size check) If $|J_k| = J_{\max}$, go to Step 3; otherwise put $J \leftarrow J_k$ and go to Step 4.
- Step 3 (Reset) Choose $J \subset J_{\max}$ with $|J| \leq J_{\max} - 2$ and do subgradient aggregation.
- Step 4 (Linearization error update) Update linearization error $\alpha_{k+1,i}$.

In Step 0, a starting point and initial parameter values are set. In the full algorithm, additional parameters must be specified. The parameter J_{\max} is used to control the bundle size, i.e., the number of elements in the current bundle denoted by J_k . In Step 1, a quadratic program is solved to compute a possible descent direction and an estimate of the leader's objective function value in that direction over the trust region. If the estimate results in a large enough improvement, then a *serious step* is taken and we go to the next iteration; otherwise a *null step* is taken to enhance the bundle. Null steps are repeated until a desirable estimate is obtained. Step 2 controls the size of the bundle according to the prescribed bound J_{\max} . Step 3 uses the so called subgradient aggregation strategy to aggregate the past subgradient information. Step 4 computes $\alpha_{k+1,i}$ which measures the quality of the i th subgradient linearization at the $k + 1$ st iteration.

A convergence result for the algorithm is now stated. In so doing, one additional assumption is needed. Also, for simplicity we work with the stopping parameter value $\varepsilon = 0$ which ordinarily would be a small finite number.

Boundedness Assumption (A3): For a given $(\mathbf{x}^0, \mathbf{y}^0)$, the level set $\{(\mathbf{x}, \mathbf{y}) : F(\mathbf{x}, \mathbf{y}) \leq F(\mathbf{x}^0, \mathbf{y}^0)\}$ is bounded.

Theorem 8.6.5 Assume (A1), (A2), and (A3). Let $(\mathbf{x}^k, \mathbf{y}^k)$ be the sequence generated by leader predominate algorithm for an arbitrary starting semi-local solution $(\mathbf{x}^1, \mathbf{y}^1)$. Then the sequence $(\mathbf{x}^k, \mathbf{y}^k)$ has at least one accumulation point, say $(\mathbf{x}^*, \mathbf{y}^*)$, which is a regular point of first category to the BLPP.

The leader predominate algorithm can be regarded as an extension of conjugate gradient methods to nondifferentiable optimization problems. Hence it may not be more than linearly convergent. To obtain faster convergence one has to exploit second-order information. Note that Proposition 8.6.2 says that PF is twice directionally differentiable along any direction. But when PF is not twice differentiable, incorporating second-order directional information in the above algorithm, though possibly helpful, is extremely difficult.

Notice, however, that $\mathbf{y}(\cdot)$ is piecewise differentiable and that nondifferentiability occurs only on the boundary of the various pieces. Thus the likelihood that $\mathbf{y}(\cdot)$ is twice differentiable at an arbitrary point may be very high. Spingarn [S15] suggests that if the twice differentiability of $\mathbf{y}(\mathbf{x})$ is independent of the optimizing sequence $\{\mathbf{x}^k\}$, then $\mathbf{y}(\cdot)$ is twice differentiable at \mathbf{x}^k with probability one. Moreover, whether or not $\mathbf{y}(\cdot)$ is twice differentiable near a particular point can be known a priori, and once the local twice differentiability is confirmed, one can utilize the second-order approximation to accelerate the convergence by employing a Newton-type method until the possible failure of twice differentiability has been detected. It must be stressed that the Newton-type methods might get stuck due to the failure of twice differentiability. With this in mind, Falk and Liu introduced the idea of a twice differentiability detector and developed a second algorithm that exploits the accompanying information. No computational experience has been reported for either algorithm so it is not possible to judge their efficacy.

8.7 TRANSFORMATION TO CONCAVE PROGRAM

The methods previously discussed in this chapter cannot, in general, guarantee a global optimum for all forms of the nonlinear BLPP. Based on the work of Shimizu and Lu [S11], the purpose of this section is to present a global optimization method for more general BLPPs.

We have already seen in Section 8.1 that our problem (8.3) or (8.5) can be equivalently expressed as

$$\begin{aligned} & \min_{\mathbf{x}, \mathbf{z}} F(\mathbf{x}, \mathbf{z}) \\ & \text{subject to } \mathbf{G}(\mathbf{x}, \mathbf{z}) \leq \mathbf{0} \end{aligned} \tag{8.48}$$

$$\begin{aligned} \mathbf{g}(\mathbf{x}, \mathbf{z}) &\leq \mathbf{0} \\ f(\mathbf{x}, \mathbf{z}) - w(\mathbf{x}) &= 0 \end{aligned}$$

where

$$w(\mathbf{x}) \triangleq f(\mathbf{x}, \mathbf{y}^*) = \min_{\mathbf{y} \in S(\mathbf{x})} f(\mathbf{x}, \mathbf{y}) \quad (8.49)$$

subject to $\mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}$

Under certain conditions, this problem can be transformed into a nonlinear (nonconvex) program whose objective and constraints are convex functions. Then, by using an exterior penalty function method, we can obtain an auxiliary problem having only inequality constraints. We prove that a solution to the auxiliary problem converges to a global optimum of the transformed problem as the penalty parameter goes to infinity. We also show that the auxiliary problem can be equivalently transformed into a concave program whose global optimum can be found.

The proposed method is applicable to a broad class of Stackelberg games whose upper-level functions are convex or the differences of two convex functions, and whose lower-level functions are convex. The transformed problem is a concave program which is globally solvable.

Problem (8.48) appears to be an ordinary nonlinear programming problem. However, it is a nondifferentiable since $w(\mathbf{x})$ is not differentiable in general. Effective computational methods do not exist for optimization problems with nondifferentiable equality constraints.

To proceed, let us assume the following.

Assumption 8.7.1 The functions $f(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$ are convex in $R^n \times R^m$.

Proposition 8.7.1 The optimal-value function $w(\mathbf{x})$ is convex when Assumption 8.7.1 holds.

From Proposition 8.7.1 there exist subgradients of $w(\mathbf{x})$. Suppose further the following.

Assumption 8.7.2 The functions $F(\mathbf{x}, \mathbf{y})$, $\mathbf{G}(\mathbf{x}, \mathbf{y})$ are convex in $R^n \times R^m$.

Under Assumptions 8.7.1 and 8.7.2, problem (8.48) becomes a nonlinear program whose objective and inequality constraints are convex functions and whose equality constraint is the difference of two convex functions. By introducing an auxiliary variable t_1 , this problem can be transformed into the following problem in which all functions are convex.

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{z}, t_1} \quad & F(\mathbf{x}, \mathbf{z}) \\ \text{subject to} \quad & \mathbf{G}(\mathbf{x}, \mathbf{z}) \leq \mathbf{0} \end{aligned}$$

$$\begin{aligned} \mathbf{g}(\mathbf{x}, \mathbf{z}) &\leq \mathbf{0} \\ f(\mathbf{x}, \mathbf{z}) - t_1 &= 0 \\ w(\mathbf{x}) - t_1 &= 0 \end{aligned} \tag{8.50}$$

If a global optimum $(\mathbf{x}^*, \mathbf{z}^*, t_1^*)$ to (8.50) is found by some algorithm, then $(\mathbf{x}^*, \mathbf{z}^*)$ solves the BLPP (8.48). In what follows, we develop a global optimization method for solving problem (8.50) instead of solving the original problem (8.3) or (8.5) directly.

Remark 1 Problem (8.50) is not a convex program since it has nonlinear equality constraints. Therefore, obtaining a global optimum with conventional mathematical programming techniques is problematic.

Remark 2 By applying the approach proposed in Section 3.2 of [L6], even when $F(\mathbf{x}, \mathbf{y})$ and $\mathbf{G}(\mathbf{x}, \mathbf{y})$ are the difference of two convex functions and $f(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$ are convex functions, we can transform problem (8.48) into a problem such as (8.50).

We now consider applying an exterior penalty function method as discussed in Section 4.4.2 to (8.50). By adding the two equality constraints in (8.50) to the objective function, we obtain the following auxiliary problem:

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{z}, t_1} \quad & F(\mathbf{x}, \mathbf{z}) - s(f(\mathbf{x}, \mathbf{z}) + w(\mathbf{x}) - 2t_1) \\ \text{subject to} \quad & \mathbf{G}(\mathbf{x}, \mathbf{z}) \leq \mathbf{0} \\ & \mathbf{g}(\mathbf{x}, \mathbf{z}) \leq \mathbf{0} \\ & f(\mathbf{x}, \mathbf{z}) - t_1 \leq 0 \\ & w(\mathbf{x}) - t_1 \leq 0 \end{aligned} \tag{8.51}$$

Before considering the relationship between problem (8.51) and problem (8.50), we state the following result.

Proposition 8.7.2 The functions $F(\mathbf{x}, \mathbf{y})$, $\mathbf{G}(\mathbf{x}, \mathbf{y})$, $f(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$ are continuous at any (\mathbf{x}, \mathbf{y}) , and $w(\mathbf{x})$ is continuous at any \mathbf{x} .

This proposition derives from Assumptions 8.7.1, 8.7.2, Proposition 8.7.1, and the continuity of convex functions.

Assumption 8.7.3 The set $\{\mathbf{x} : \mathbf{G}(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) \leq \mathbf{0}\}$ is not empty.

Assumption 8.7.4 The function $F(\mathbf{x}, \mathbf{y})$ has a lower bound.

Theorem 8.7.1 Let Assumptions 8.7.1~8.7.4 hold and assume that problem (8.51) has a global optimum for any $s > 0$. Let $\{(\mathbf{x}^k, \mathbf{z}^k, t_1^k)\}$ be a sequence of such solutions to problem (8.51) corresponding to a sequence of positive numbers $\{s^k\}$ monotonically

diverging to infinity. Then any accumulation point of the sequence $\{(\mathbf{x}^k, \mathbf{z}^k, t_1^k)\}$ is a global optimum to problem (8.50) and the following relation holds:

$$\lim_{k \rightarrow \infty} s^k (f(\mathbf{x}^k, \mathbf{z}^k) + w(\mathbf{x}^k) - 2t_1^k) = 0$$

Proof: See [S10] or [S11].

Theorem 8.7.1 shows that a sequence of global optimum to the auxiliary problem (8.51) converges to a global optimum of the transformed problem (8.50).

Global Optimization of the Auxiliary Problem

The auxiliary problem (8.51) is an inequality constrained optimization problem where the objective is the difference of two convex functions and the constraint functions are convex. We now consider the following problem related to (8.51).

$$\begin{aligned} & \min_{\mathbf{x}, \mathbf{z}, t_1, t_2} t_2 - s(f(\mathbf{x}, \mathbf{z}) + w(\mathbf{x}) - 2t_1) \\ & \text{subject to } F(\mathbf{x}, \mathbf{z}) - t_2 \leq 0 \\ & \quad G(\mathbf{x}, \mathbf{z}) \leq 0 \\ & \quad g(\mathbf{x}, \mathbf{z}) \leq 0 \\ & \quad f(\mathbf{x}, \mathbf{z}) - t_1 \leq 0 \\ & \quad w(\mathbf{x}) - t_1 \leq 0 \end{aligned} \tag{8.52}$$

This is a concave program with a concave objective function and a convex constraint set. The following theorem holds in regard to the equivalence relation between problem (8.51) and problem (8.52).

Theorem 8.7.2 Let the penalty parameter $s = s^k$ be given.

- (i) If $(\mathbf{x}^k, \mathbf{z}^k, t_1^k)$ solves problem (8.51), then there exists a $t_2^k \in R^1$ such that $(\mathbf{x}^k, \mathbf{z}^k, t_1^k, t_2^k)$ solves problem (8.52).
- (ii) If $(\mathbf{x}^k, \mathbf{z}^k, t_1^k, t_2^k)$ solves problem (8.52), then $(\mathbf{x}^k, \mathbf{z}^k, t_1^k)$ solves problem (8.51).

Proof: See [S10] or [S11].

By this theorem we can obtain a global optimum to problem (8.51) by solving the concave program (8.52). This leads to the following procedure for finding a global optimum to problem (8.50) which is equivalent to the original BLPP (8.3) or (8.5).

(Main Algorithm)

Step 1 Let $\delta > 0$ be a termination scalar. Choose a penalty parameter $s^1 > 0$ and a scalar $\beta > 1$. Set $k \leftarrow 1$.

Step 2 Solve the concave program (8.52) with $s = s^k$ to obtain $(\mathbf{x}^k, \mathbf{z}^k, t_1^k, t_2^k)$, a global optimum. Go to Step 3.

Step 3 If $-s^k (f(\mathbf{x}^k, \mathbf{z}^k) + w(\mathbf{x}^k) - 2t_1^k) < \delta$, then take $(\mathbf{x}^k, \mathbf{z}^k, t_1^k)$ as the global optimum to (8.50) and terminate. Otherwise, set $s^{k+1} \leftarrow \beta s^k$, $k \leftarrow k + 1$, and go to Step 2.

We now explain how to solve the concave program (8.52) with $s = s^k$ given. Generally, there are several local optima in a concave program. Each is attained at extreme points of the constraint set because of the concavity of the objective function and the convexity of the constraint set (see Theorem I.1 in [H10]). Therefore, we only need to search the set of extreme points of the constraint set for finding a global optimum to a concave program. This is achieved in Step 2 as follows. In general, there are two main approaches for obtaining a global optimum to a concave program: the outer approximation method by cutting planes and the branch-and-bound method. Horst and Tuy [H10] presented a survey of these approaches. Here we apply the former to solve (8.52) with $s = s^k$.

Let the feasible set

$$S = \{(\mathbf{x}, \mathbf{z}, t_1, t_2) : F(\mathbf{x}, \mathbf{z}) - t_2 \leq 0, \mathbf{G}(\mathbf{x}, \mathbf{z}) \leq \mathbf{0}, \mathbf{g}(\mathbf{x}, \mathbf{z}) \leq \mathbf{0}, f(\mathbf{x}, \mathbf{z}) - t_1 \leq 0, w(\mathbf{x}) - t_1 \leq 0\}$$

be enclosed in a polytope $S_1 \supset S$. Instead of solving problem (8.52), we solve the relaxed problem

$$\min_{(\mathbf{x}, \mathbf{z}, t_1, t_2) \in S_1} t_2 - s^k (f(\mathbf{x}, \mathbf{z}) + w(\mathbf{x}) - 2t_1) \quad (8.53)$$

Let $(\mathbf{x}^{k,1}, \mathbf{z}^{k,1}, t_1^{k,1}, t_2^{k,1})$ be a global optimum of (8.53). Then $(\mathbf{x}^{k,1}, \mathbf{z}^{k,1}, t_1^{k,1}, t_2^{k,1})$ solves problem (8.52) if $(\mathbf{x}^{k,1}, \mathbf{z}^{k,1}, t_1^{k,1}, t_2^{k,1}) \in S$. Otherwise, we can find a hyperplane $\ell_1(\mathbf{x}, \mathbf{z}, t_1, t_2) = 0$ separating S and $(\mathbf{x}^{k,1}, \mathbf{z}^{k,1}, t_1^{k,1}, t_2^{k,1})$ in the sense that

$$\ell_1(\mathbf{x}^{k,1}, \mathbf{z}^{k,1}, t_1^{k,1}, t_2^{k,1}) > 0 \text{ and } \ell_1(\mathbf{x}, \mathbf{z}, t_1, t_2) \leq 0 \text{ for all } (\mathbf{x}, \mathbf{z}, t_1, t_2)$$

The linear constraint $\ell_1(\mathbf{x}, \mathbf{z}, t_1, t_2) \leq 0$ is added to the system of inequalities defining S_1 . We cut off the point $(\mathbf{x}^{k,1}, \mathbf{z}^{k,1}, t_1^{k,1}, t_2^{k,1})$ and determine a new polytope S_2 that provides a tighter approximation to S than S_1 . We now replace S_1 with S_2 and repeat the procedure.

Since the functions $F(\mathbf{x}, \mathbf{y})$, $\mathbf{G}(\mathbf{x}, \mathbf{y})$, $f(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$ and $w(\mathbf{x})$ are all convex, the maximal component function

$$\begin{aligned} p(\mathbf{x}, \mathbf{z}, t_1, t_2) &= \max\{F(\mathbf{x}, \mathbf{z}) - t_2, G_1(\mathbf{x}, \mathbf{z}), \dots, G_p(\mathbf{x}, \mathbf{z}), \\ &\quad g_1(\mathbf{x}, \mathbf{z}), \dots, g_q(\mathbf{x}, \mathbf{z}), f(\mathbf{x}, \mathbf{z}) - t_1, w(\mathbf{x}) - t_1\} \end{aligned}$$

is convex and possesses subgradients. Further, one can compute a subgradient of an optimal-value function or a maximal component function by using Proposition 6.7.1 in [S10] (alternatively, see [C5]). However, there might exist a vector $\bar{\mathbf{x}} \in \{\mathbf{x} : (\mathbf{x}, \mathbf{z}, t_1, t_2) \in S_j\}$ ($j = 1, 2, \dots$) such that the set $\{\mathbf{y} : \mathbf{g}(\bar{\mathbf{x}}, \mathbf{y}) \leq 0\}$ is empty, that is, no subgradient of $w(\bar{\mathbf{x}})$ will be available. To cope algorithmically with this situation, we take $S_1 \subseteq \{(\mathbf{x}, \mathbf{z}, t_1, t_2) : \mathbf{g}(\mathbf{x}, \mathbf{z}) \leq 0\}$.

Assuming that an initial polytope $S_1 \supset S$ is given, at each iteration j define the linear function $\ell_j(\mathbf{x}, \mathbf{z}, t_1, t_2)$ as follows:

$$\ell_j(\mathbf{x}, \mathbf{z}, t_1, t_2) = (\mathbf{q}^j)^T \begin{pmatrix} \mathbf{x} - \mathbf{x}^{k,j} \\ \mathbf{z} - \mathbf{z}^{k,j} \\ t_1 - t_1^{k,j} \\ t_2 - t_2^{k,j} \end{pmatrix} + p(\mathbf{x}^{k,j}, \mathbf{z}^{k,j}, t_1^{k,j}, t_2^{k,j})$$

where \mathbf{q}^j is a subgradient of the function p at the trial point $(\mathbf{x}^{k,j}, \mathbf{z}^{k,j}, t_1^{k,j}, t_2^{k,j})$. Note that p is convex.

Accordingly, a procedure for solving the concave program (8.52) using an outer approximation by cutting planes can be stated as follows. Note that the set of extreme points of a polytope is its vertex set.

(Partial Algorithm for Step 2)

Step 2.1 Find a polytope S_1 so that $S_1 \supset S$ and $S_1 \subseteq \{(\mathbf{x}, \mathbf{z}, t_1, t_2) : \mathbf{g}(\mathbf{x}, \mathbf{z}) \leq 0\}$. Set $j \leftarrow 1$.

Step 2.2 To obtain a new trial point $(\mathbf{x}^{k,j}, \mathbf{z}^{k,j}, t_1^{k,j}, t_2^{k,j})$, solve

$$\min_{(\mathbf{x}, \mathbf{z}, t_1, t_2) \in V(S_j)} t_2 - s^k (f(\mathbf{x}, \mathbf{z}) + w(\mathbf{x}) - 2t_1) \quad (8.54)$$

where $V(S_j)$ is the vertex set of the polytope S_j . If $(\mathbf{x}^{k,j}, \mathbf{z}^{k,j}, t_1^{k,j}, t_2^{k,j}) \in S$, take $(\mathbf{x}^{k,j}, \mathbf{z}^{k,j}, t_1^{k,j}, t_2^{k,j})$ as a global optimum to (8.52) and terminate Step 2; otherwise, go to Step 2.3.

Step 2.3 Add the following linear constraint

$$\ell_j(\mathbf{x}, \mathbf{z}, t_1, t_2) = (\mathbf{q}^j)^T \begin{pmatrix} \mathbf{x} - \mathbf{x}^{k,j} \\ \mathbf{z} - \mathbf{z}^{k,j} \\ t_1 - t_1^{k,j} \\ t_2 - t_2^{k,j} \end{pmatrix} + p(\mathbf{x}^{k,j}, \mathbf{z}^{k,j}, t_1^{k,j}, t_2^{k,j}) \leq 0$$

to the system of inequalities defining S_j , where \mathbf{q}^j is a subgradient of p at the trial point $(\mathbf{x}^{k,j}, \mathbf{z}^{k,j}, t_1^{k,j}, t_2^{k,j})$. Set $S_{j+1} \leftarrow S_j \cap \{(\mathbf{x}, \mathbf{z}, t_1, t_2) : \ell_j(\mathbf{x}, \mathbf{z}, t_1, t_2) \leq 0\}$, $j \leftarrow j + 1$ and go to Step 2.2.

In Step 2.2, we need to find the vertex set $V(S_j)$ of the polytope S_j in order to solve problem (8.54). Procedures for doing this can be found in Section II.4 of [H10]. Also, it can be shown (e.g., Theorem II.1 of [H10]) that every accumulation point of the sequence $(\mathbf{x}^{k,j}, \mathbf{z}^{k,j}, t_1^{k,j}, t_2^{k,j})$ is global optimum to problem (8.52).

Example 8.7.1 Consider the problem

$$\begin{aligned} & \min_x 16x^2 + 9y^*(x)^2 \\ & \text{subject to } -4x + y^*(x) \leq 0, \quad x \geq 0 \\ & \qquad \left(x + y^*(x) - 20 \right)^4 = \min_y (x + y - 20)^4 \\ & \qquad \text{subject to } 4x + y - 50 \leq 0, \quad y \geq 0 \end{aligned} \tag{8.55}$$

The feasible set consists of the line segments \overline{AB} and \overline{BC} as shown in Fig. 8.4.

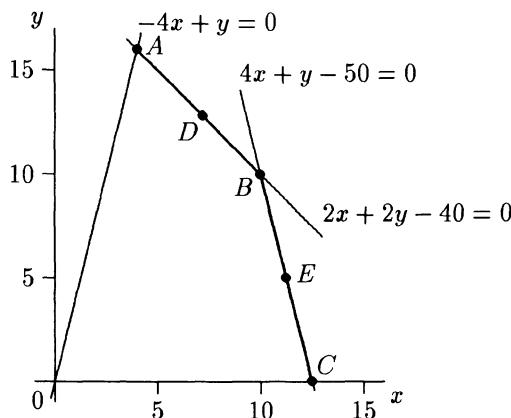


Figure 8.4 Geometry of Example 8.7.1

The set \overline{ABC} is not convex. Furthermore, this problem has two local solutions: $D(7.2, 12.8)$ and $E(11.25, 5)$. The global solution is $E(11.25, 5)$.

Another aspect of problem (8.55) is that the objective function in the lower level is not quadratic, so achieving global optimality is difficult. However, when the objective function in the lower level is quadratic, such problems can be solved by using the method described in [L6].

Using the method proposed in this section, we have a concave program that can be written in the form of (8.52) as follows:

$$\begin{aligned}
 & \min_{x,z,t_1,t_2} t_2 - s\{(x+z-20)^4 + w(x) - 2t_1\} \\
 & \text{subject to } 16x^2 + 9z^2 - t_2 \leq 0 \\
 & \quad 4x + z \leq 0 \\
 & \quad 4x + z - 50 \leq 0 \\
 & \quad (x+z-20)^4 - t_1 \leq 0 \\
 & \quad w(x) - t_1 \leq 0 \\
 & \quad x, z \geq 0
 \end{aligned} \tag{8.56}$$

where $w(x)$ is the optimal-value function of the lower-level problem. That is,

$$w(x) = \min_y \{(x+y-20)^4 : 4x+y-50 \leq 0, y \geq 0\}$$

Table 8.7 summarizes the computational results obtained by the proposed method. Each row shows the results of problem (8.56) with $s = s^k$. Note that k is the iteration number and $F^k = F(x^k, z^k)$, $H^k = f(x^k, z^k) + w(x^k) - 2t_1^k$. The data shown in Table 8.7 verifies that the global optimum to problem (8.55) was obtained.

Table 8.7 Computational results for Example 8.7.1

k	s^k	(x^k, z^k)	(t_1^k, t_2^k)	F^k	$s^k H^k$	$F^k - s^k H^k$
1	0.1	(5.546, 10.48)	(248.6, 1481.2)	1481.2	24.86	1506.1
2	1.0	(5.899, 10.49)	(170.7, 1546.4)	1546.4	170.7	1717.2
3	10.0	(6.646, 11.58)	(9.826, 1914.3)	1914.3	98.26	2012.6
4	100.0	(11.25, 4.999)	(197.8, 2250.0)	2250.0	7.252×10^{-4}	2250.0
True values		(11.25, 5)	—	2250	—	—

8.8 ASSESSMENT OF ALGORITHMS

An assessment of the algorithms presented in this chapter for the nonlinear BLPP and those in Chapters 5, 6 and 7 for the linear, discrete and convex BLPP, respectively, indicates that exact solutions can only be guaranteed for problem instances with up to a few hundred variables and constraints, and then only for the linear case. When nonlinear (nonconvex) functions are included in the model, virtually all algorithms stumble in the presence of more than a handful of variables and constraints. With a few exceptions, such as the network design paper by Ben-Ayed et al. [B19], the work by Bard [B8] on government support for biofuel production (see Chapter 12), and applications mentioned in several unpublished manuscripts, the examples contained

in the above references are merely illustrative. Our ability for formulate problems far outstrips our capacity to solve them optimally.

When faced with the problem of actually having to provide solutions to large-scale BLPPs, researchers have inevitably fallen back on heuristics and ad hoc procedures. Anandalingam et al. [A8] have taken the lead in this regard by devising simulated annealing and genetic algorithm-based approaches to the linear BLPP. As we shall see in Chapter 9, they were able to confirm global optimality in many instances; however, run times were less than satisfactory. More impressive results have been obtained by Gendreau et al. [G5] who proposed a tabu search heuristic, also for the linear case. For a series of test problems ranging from 40 to 200 variables and from 20 to 200 constraints, they were able to obtain the optimal solution 74% of the time. In the problem faced by Bard, because there was only a small number of leader variables, it was possible to impose a grid over their range and repeatedly solve the follower's problem for every decreasing grid step sizes. Convergence was achieved in all cases examined.

The conclusion that can be drawn from these observations and from ongoing experience with various applications ranging from digital filter design, government regulation, and decentralized control is that there is still an enormous need for efficient algorithms. It is perhaps for this reason that realistic applications lag considerably behind theory and code development.

HEURISTICS

9.1 INTRODUCTION

As emphasized throughout the text, the main difficulty in solving bilevel programs stems from their inherent nonconvexity and nondifferentiability. This is true even for the simplest of forms where all the functions are linear and all the variables are continuous. Although exact approaches, such as those based on the Kuhn-Tucker conditions of the follower's problem or on variable elimination, handle the nonconvexity in different ways, their success has been limited. We are still only able to solve moderate size problems. Heuristics, and what have become known as global optimization techniques, offer additional possibilities.

A number of global optimization techniques have been proposed for solving more general classes of nonconvex programs [H10]. They are primarily based on traditional operations research methods, including vertex enumeration, cutting planes or domain partitioning [G8], Lagrangian relaxation [A4], and branch and bound. In the associated work, the slant has been on convergence analysis rather than on exploiting the tremendous power that computers offer in solving difficult problems. Convergence analysis is important in assessing the worst-case complexity of an algorithm and in getting an understanding of how it can be expected to perform. It may not, however, provide much insight into how an algorithm will fair on practical applications or on specific classes of problems.

In this chapter, we present the most recent developments on heuristics for solving bilevel programs. For the most part, work in this area has been restricted to the linear case leaving the nonlinear case to the next generation of researchers. We begin with two approaches that have roots in artificial intelligence: genetic algorithms [G11] and simulated annealing [K3]. The latter operates in a manner similar to pure random search; however, new solutions need not always be improvements. Inferior solutions are accepted at random with nonzero probability according to a specific parameterized distribution, usually called the temperature. Genetic algorithms (GAs)

base the search for better solutions on Darwinian principles of looking for fitter genes, where the genes are analogous to admissible solutions and ‘fitness’ is the objective function value. Unlike simulated annealing which maintains a single solution at each stage of the computations, GAs maintain a population of solutions. When going from one iteration to the next, only the best overall population is considered.

The third heuristic discussed is based on tabu search [G10] which can be thought of as a higher level procedure designed to guide other methods (or their component processes) to escape the trap of local optimality. Tabu search has three basic features: (i) a flexible memory structure to permit search information to be exploited more thoroughly than by rigid or memoryless systems; (ii) conditions for strategically constraining and freeing the search process embodied in tabu restrictions and aspiration criteria; and (iii) memory functions of varying time spans for intensifying and diversifying the search. The method can be viewed as a post-processor that employs a set of moves for transferring one solution into another. To do this, it requires an evaluation function for measuring the attractiveness of these moves. Examples of moves are changing the value assigned to a variable, adding or deleting an element from a set, and executing a pivot step. Tabu search has its origins in combinatorial procedures applied to nonlinear covering problems [G9]. Its greatest successes have been on structured integer programs where high quality solutions have been obtained with modest computational effort. This has spurred interest in applying the technique to a variety of continuous and nondifferentiable optimization problems.

9.2 ARTIFICIAL INTELLIGENCE-BASED APPROACHES

In this section, we consider the adaptation of genetic algorithms and simulated annealing for solving BLPPs. Both techniques belong to the generate-and-test paradigm of artificial intelligence, capturing the gross features of the solution domain before homing in on the optimal (or near-optimal) solution. This ‘domain specific knowledge’ is essential for the techniques to solve optimization problems, and hence differentiate them from the more traditional OR techniques previously discussed.

In the development of the algorithms, we work with the linear BLPP (5.1) of the form

$$\min_{\mathbf{x} \geq 0} F(\mathbf{x}, \mathbf{y}) = c_1 \mathbf{x} + d_1 \mathbf{y} \quad (9.1a)$$

$$\min_{\mathbf{y} \geq 0} f(\mathbf{y}) = d_2 \mathbf{y} \quad (9.1b)$$

$$\text{subject to } g(\mathbf{x}, \mathbf{y}) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{b} \leq 0 \quad (9.1c)$$

The material below reflects the work of Anandalingam et al. [A8] and Mathieu et al. [M10] who developed and evaluated a genetic algorithm and a simulated annealing

heuristic for the linear BLPP. Extensions to the nonlinear case are straightforward. We begin with an overview of the two approaches and discuss how they can be used to solve the problem at hand. The individual steps are then outlined followed by comparisons with Bard's grid search procedure [B2]. Computational results indicate that the AI-based approaches have some promise but are in need of considerably more streamlining if they are to be applied to problem instances too large to be solved by exact methods.

9.2.1 Overview of Genetic Algorithms

Adaptive procedures, of which the genetic algorithm is one, involve the progressive modification of the structure of the process to give better performance for a particular problem environment [H8]. In a genetic algorithm, the structure is modified according to the general principles of genetics. GAs are used for global search, but instead of maintaining a single structure (hereafter "solution" to the problem) at any point in time, a set of structures (hereafter "population" of solutions) are maintained. The initial population of solutions is usually generated at random. Each succeeding population is created from its predecessors using randomized mechanisms that seek to preserve and combine the positive characteristics of the more attractive members of the preceding population.

Each solution in a genetic algorithm is represented as a string of characters from some alphabet, usually ones and zeroes from the binary alphabet. This string is analogous to a chromosome in biological systems and is meant to be sufficient to capture (encode) all the characteristics of the solution it represents. Two genetic operations are typically performed on strings: crossover replaces part of one string with the corresponding part of another, and mutation arbitrarily changes the value of a character on a string to another member of its alphabet (another of its possible values).

Each candidate solution of each generation is assigned a quantity known as its fitness which is the objective function value in optimization problems. Creation of the succeeding generation from its parent generation occurs as follows: To create a member of the succeeding generation, first, a member of the parent generation is selected at random with probability proportional to its fitness divided by the generation average. Then crossover occurs with a certain probability (a parameter of the GA). If crossover did not occur, the single selected member of the parent generation is copied into the succeeding generation intact, at least temporarily. Finally, whether or not crossover occurred, the new member of the succeeding generation is allowed to undergo mutation with (typically low) probability equal to the mutation probability (another parameter of the GA). The entire process repeats until the succeeding generation has been "procreated" from the parent generation.

From an optimization perspective, the genetic algorithm works as follows: The first generation of solutions, generated at random, is uniformly distributed over the search space. Succeeding generations, due to the “survival of the fittest” of their ancestors, tend to be increasingly localized around the best modes in the search space. The final generation is usually concentrated around one, or at most, a few very good modes of the search space. (The number of generations, as well as the generation size, are exogenous parameters sufficiently large for the preceding to occur.)

The general genetic algorithm can be described as follows [G12]:

```

 $k = 0$ 
Initialize  $\Pi(k)$  (population at generation  $k$ )
Evaluate  $\Pi(k)$ 
While termination condition not satisfied, do
Begin
     $k \leftarrow k + 1$ 
    Select  $\Pi(k)$ 
    Recombine  $\Pi(k)$ 
End

```

It should be noted that GA-based optimization procedures may be better described as heuristics rather than as algorithms, especially in the vernacular of operations researchers. We will continue to use algorithm since it is standard practice to do so in the GA field and because it also fits well with the use of the word by computer scientists.

9.2.2 GABBA

The Genetic Algorithm Based Bilevel programming Algorithm (GABBA) is best described as an adaptive reproductive plan based on principles common to genetic algorithms. To solve the linear bilevel program (9.1a)–(9.1c), the leader’s decision vector \mathbf{x} is reproduced according to a modification of the GA, and the follower’s decision vector \mathbf{y} is obtained by solving the lower-level linear program (9.1b,c). The fitness test involves a modification of traditional GA tests. GABBA is described in detail in the next subsection. We now highlight its basic parameters.

GABBA’s reproductive plan is controlled by the operators, population size, number of structures in the previous generation’s population to reproduce, percentage of alleles in each structure to reproduce, number of new structures to produce randomly every generation, and selection strategy. It does not exploit crossover. It could be argued that crossover is a special case of mutation, since it could be reduced to multiple mutations. Many assert that crossover is essential in GA, as it is responsible for propagating the characteristics of best structures, like real-life genetic processes.

Using mutations does not allow each successive generation to be strongly connected to the parent generation. This is not true in the algorithm proposed by Mathieu et al., though, because mutations are used to change only part of the parent characteristics.

Contrary to what is commonly done in most pure genetic algorithms, GABBA does not encode its structures as 0-1 bit strings, but rather as a string of base-10 digits. This alternative is more intuitive for adapting reproductive plans to mathematical programming.

In GABBA, the *population size* N is the number of components of the overall solution vector $\mathbf{z} = (\mathbf{x}, \mathbf{y})$. In relation to the definition of the bilevel programming problem, $\mathbf{z} \in IR \subset R^{n+m}$, where IR is the inducible region (see Definition 5.1.1e). Individual members in the population (which are called *structures*), are represented as points in $(n + m)$ -dimensional space. The population at generation (iteration) k is represented by $\Pi(k)$. As in traditional genetic algorithms, each component of the structure is called an *allele*. The following example is used to illustrate the notation.

Example 9.2.1 Suppose we have the population:

Population #	x_1	x_2
1	2.2	3.1
2	5.6	3.7
3	7.8	10.9

which consists of 3 structures. For structure #2, the alleles are 5.6 and 3.7 which are represented by the 2-dimensional vector (5.6, 3.7).

In GABBA, all structures are defined by alleles of base-10 numbers. The base-10 adaptive process is fundamental to GABBA. In the encoding, each allele is defined to have a head and a tail. The point that separates the head from the tail is determined by a parameter SCALE, where

$$\text{SCALE} = 10^\eta, \eta \in I$$

and I is the set of positive real integers. For example, if our allele is 12345.67, and $\text{SCALE} = 10$, the head of the allele is 1234*.**, and the tail of the allele is ****5.67.

When optimizing a function using this base-10 scheme, the information to be passed on to the next generation is contained in the head. When creating a new species from an old species, the head of the previous generation is kept and a random number, generated using a uniform distribution on the interval (0, SCALE) becomes the new tail. For example, if 1234*.** is passed to the new generation, and a random tail

of 9.81 is generated, the new species becomes 12349.81. Thus tails are continually generated at a given SCALE level until “good” solutions are obtained. After it is determined what are good solutions based on the problem objective, the SCALE value is modified so that new solutions are more precise than the old ones. The procedure employed by Mathieu et al. to revise the SCALE parameters after the determination of good solutions is:

$$(\text{new}) \text{ SCALE} = (\text{old}) \text{ SCALE}/10 \quad (9.2)$$

The population is generated within feasible lower and upper bounds. Thus the *alphabet* for each allele is all base-10 numbers between the lower and upper bounds. This is different from traditional uses of GAs where the alphabet is made up of the binary code (0,1). If the SCALE is changed according to eq. (9.2), we will have as many macro-iterations in GABBA as the order of magnitude that the alphabet spans.

The *selection strategy* is based on obtaining at each generation, the N most fit structures (i.e., solutions with the smallest value of $F(\mathbf{x}, \mathbf{y})$, the leader’s objective). Candidates for selection include the N structures from the previous generation, NP offspring structures newly created by mutation of structures in the previous generation, and NR new structures produced randomly. Note that not all NR candidates for mutation will produce feasible solutions. Accordingly, there will be slightly less than $(N + NP + NR)$ candidates from which to select the N best for the next generation.

Algorithm

We now present the specifics of GABBA using the following notation:

I	set of positive real numbers
$U[a, b]$	uniform distribution between a and b
$\mathbf{x}(k), \mathbf{y}(k)$	leader and follower decision vectors at generation k
x_i	ith component of the leader’s decision vector $\mathbf{x} = (x_1, \dots, x_{n_1})$
y_i	ith component of the follower’s decision vector $\mathbf{y} = (y_1, \dots, y_{n_2})$
$\mathbf{x}^j, \mathbf{y}^j$	jth structure of leader and follower decision vectors
\mathbf{x}_i^j and \mathbf{y}_i^j	ith component of jth structure of vectors \mathbf{x} and \mathbf{y}
$F^j(k)$	value of the leader’s objective given by (9.1a) for $\mathbf{x} = \mathbf{x}^j, \mathbf{y} = \mathbf{y}^j$ at the k th generation
$\Pi(k)$	population at generation k : $\Pi(k) = \{\mathbf{x}(k), \mathbf{y}(k)\}$

Note that when the generation index k is suppressed, it should be understood that the respective decision vectors or components are being considered at their appropriate generation. It should also be noted that, as given by Definition 5.1.1 in Section 5.1, whenever we refer to the follower’s rational reaction set $P(\mathbf{x})$, we mean that the following linear program is solved for a given \mathbf{x} , (say $\widehat{\mathbf{x}}$), using the simplex method.

$$\begin{aligned} & \min && d_2 \mathbf{y} \\ & \text{subject to} && \mathbf{B}\mathbf{y} \leq \mathbf{b} - \mathbf{A}\widehat{\mathbf{x}} \\ & && \mathbf{y} \geq 0 \end{aligned} \quad (9.3)$$

Step 0 (Initialization)

Let $k = 0$, $F^*(-1) = \infty$, and set parameters:

(a) N = population size

(b) NP = number of current solutions (i.e., structures) in population $\hat{P}(k)$ to undergo mutation

(c) NX = number of leader decision variables (i.e., alleles) to undergo mutation

(d) NR = number of new random solutions created during each iteration

(e) ε = degree of accuracy required

Step 1 (Set bounds)

Generate an upper bound (and lower bound) for each component x_i by solving the following problem for $i = 1, \dots, n$

$$\begin{aligned} & \max (\min) \quad x_i \\ & \text{subject to} \quad Ax + By \leq b \\ & \quad x \geq 0, \quad y \geq 0 \end{aligned}$$

Set SCALE = 10^η , $\eta \in I$ such that SCALE $\geq \max\{x_1, \dots, x_{n_1}\}$.

Step 2 (Generate initial population)

$j = 1$

Generate population $\hat{P}(k)$, which contains N vectors $\mathbf{z}^j = (\mathbf{x}^j, \mathbf{y}(\mathbf{x}^j))$,

$j = 1, \dots, N$, as follows:

(2.1) $x_i^j \sim U[\min x_i^j, \max x_i^j]$, $i = 1, \dots, n$

(2.2) Solve (9.2.3)

(a) If feasible, store rational reaction $\mathbf{y}(\mathbf{x}^j)$;

If $j = N$, go to Step 3;

Else put $j \leftarrow j + 1$ and go to Step 2.1

(b) If infeasible, discard \mathbf{x}^j and go to Step 2.1

Step 3 (SCALE modification)

Sort array $\Pi(k)$ according to level 1 objective:

$$F^j(k) = c_1 \mathbf{x}^j + d_1 \mathbf{y}^j, \quad j = 1, \dots, N$$

$$(\text{i.e., } F^j(k) < F^{j-1}(k), \forall j)$$

Let $\mathbf{z}^j = (\mathbf{x}^j, \mathbf{y}^j) = \arg \{F^j(k), j = 1, \dots, N\}$.

Store $F^*(k) = \min F^j(k)$ and $(\mathbf{x}^*(k), \mathbf{y}^*(k)) = \arg F^*(k)$.

Let $F_\alpha(k) = \frac{1}{\alpha N} \times \sum_{j=1}^{\alpha N} F^j(k)$.

If $F^*(k) = F^*(k - 1)$ and if $F_{0.6}(k) \leq 0.15 \times F^*(k)$,
then SCALE = SCALE/10.

Step 4 (Stopping criterion)

If $\text{SCALE} < \varepsilon$, then STOP; satisfactory solution is $(\mathbf{x}^*(k), \mathbf{y}^*(k))$.

Otherwise, go to Step 5.

Step 5 (Mutated structures)

Let $\mathbf{z}_1^j = (\mathbf{x}^j, \mathbf{y}^j)$, $j = 1, \dots, NP$ (Note $NP < N$, and $\{\mathbf{z}_1^j\} \subset \{\mathbf{z}^j\}$).

Mutate the NP vectors \mathbf{z}_1^j as follows (let $j = 0$):

(5.1) $j = j + 1$

Obtain $\mathbf{x}^{j'}$ by mutating NX randomly selected alleles of \mathbf{x} such that

$\mathbf{x}_i^{j'} = \text{int}(\mathbf{x}_i^j / \text{SCALE}) \times \text{SCALE} + \omega$, where $\omega \sim U[0, \text{SCALE}]$.

(5.2) Solve problem (9.3)

(i) If solution is feasible, store $\mathbf{y}(\mathbf{x}^j)$;

If $j = NP$, go to Step 6; else go to Step 5.1

(ii) If solution is infeasible, discard \mathbf{x}^j and go to Step 5.1

Step 6 (New random structures)

Let $\mathbf{z}_2^j = (\mathbf{x}^j, \mathbf{y}(\mathbf{x}^j))$, $j = 1, \dots, NR < N$.

Generate NR vectors \mathbf{z}_2^j as follows (let $j = 0$):

(6.1) $j = j + 1$

$\mathbf{x}_i^j \sim U[\min \mathbf{x}_i^j, \max \mathbf{x}_i^j]$, $i = 1, \dots, n$

(6.2) Solve problem (9.3)

(i) If feasible, store $\mathbf{y}(\mathbf{x}^j)$.

If $j = NR$, go to Step 7; else go to Step 6.1

(ii) If infeasible, discard \mathbf{x}^j and go to Step 6.1

Step 7 (Selection)

From the N $\{\mathbf{x}, \mathbf{y}(\mathbf{x})\}$ structures from $\Pi(k)$, from the NP or less $\{\mathbf{x}^j, \mathbf{y}(\mathbf{x}^j)\}$ structures from mutation, and from the NR $\{\mathbf{x}^{j'}, \mathbf{y}(\mathbf{x}^{j'})\}$ population from random structures, select the N structures that have smallest values of $F(\cdot)$ to form $\Pi(k + 1)$.

Put $k \leftarrow k + 1$ and go to Step 3.

Convergence and Stopping Rules

Many probabilistic search techniques have been proposed in the operations research literature (e.g., see [R1]). All these methods consider the problem solved if, for some $\varepsilon > 0$, an element of the following set has been identified:

$$\begin{aligned} A_z(\varepsilon) &= \{\mathbf{z} \in IR : \| \mathbf{z} - \mathbf{z}^* \| \leq \varepsilon\} \\ A_F(\varepsilon) &= \{\mathbf{z} \in IR : |F(\mathbf{z}) - F(\mathbf{z}^*)| \leq \varepsilon\} \end{aligned}$$

In Step 4 of GABBA, the set $A_Z(\varepsilon)$ is used for each SCALE level of the algorithm. Clearly at very large values, SCALE dominates ε and we move to the next level only when \mathbf{z} exactly equals \mathbf{z}^* . At smaller values of SCALE, the stopping value ε tends to dominate.

Generally, one cannot guarantee absolutely that probabilistic methods will provide a solution $\mathbf{z} \in A_Z(\varepsilon)$ or $\mathbf{z} \in A_F(\varepsilon)$. Nevertheless, under conditions on the sampling distributions and the curvature of $F(\cdot)$, it can be proven that an element of $A_Z(\varepsilon)$ or ($A_F(\varepsilon)$) is sampled *almost surely* as the sample size increases. One option to guarantee success absolutely would be to use the probabilistic technique in the first stage for locating regions where good local optima exist, and to use traditional operations research techniques at the second stage to find the actual optimum.

9.2.3 Grid Search Technique

Bard [B2] proposed a technique for solving linear BLPs that makes use of parametric programming. His algorithm is a heuristic in the sense that it is guaranteed to find a global optimum to (9.1) only if one exists that happens to be an efficient point of the vector optimization problem

$$\min\{(F(\mathbf{x}, \mathbf{y}), f(\mathbf{y})) : \mathbf{Ax} + \mathbf{By} \leq \mathbf{b} \leq 0, \mathbf{x} \geq 0, \mathbf{y} \geq 0\} \quad (9.4)$$

Mathieu et al. chose to compare GABBA with the grid search technique because the latter generally has low computational requirements and hence converges quickly.

To understand Bard's algorithm, consider the standard linear program

$$\begin{array}{ll} \min_{\mathbf{x} \geq 0, \mathbf{y} \geq 0} & \lambda(\mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y}) + (1 - \lambda)\mathbf{d}_2 \mathbf{y} \\ \text{subject to} & \mathbf{Ax} + \mathbf{By} \leq \mathbf{b} \end{array} \quad (9.5)$$

where $0 \leq \lambda \leq 1$ is a scalar parameter. Call the solution $(\mathbf{x}^*(\lambda), \mathbf{y}^*(\lambda))$. It is well known that for $\lambda \in (0, 1)$ any solution to (9.5) is an efficient point of (9.4). Now let λ_{\max} be the largest value of λ for which \mathbf{y}^* is in the rational reaction set $P(\mathbf{x}^*) = \{\mathbf{y} : \mathbf{y} \in \arg \min\{f(\hat{\mathbf{y}}) : \mathbf{B}\hat{\mathbf{y}} \leq \mathbf{b} - \mathbf{A}\mathbf{x}^*, \hat{\mathbf{y}} \geq 0\}\}$ and hence solves the follower's problem. If such a solution $(\mathbf{x}^*(\lambda_{\max}), \mathbf{y}^*(\lambda_{\max}))$ exists then it solves (9.1).

Bard's algorithm iteratively solves problem (9.5) for varying values of λ . Sensitivity analysis or parametric linear programming [M18] is used to expedite the computations. Mathieu et al. used a modified grid search (MGS) algorithm that follows Bard's general principle but employs upper and lower bounds on λ at each iteration so as to converge to λ_{\max} quickly. The computations are halted when decreasing λ no longer effects a basis change.

9.2.4 Comparison of GABBA and Grid Search

To test GABBA against other algorithms, Mathieu et al. constructed a random BLPP generator. The problems investigated were of size (3,7,4), (5,10,6), (6,14,8) and (8,17,10), where the triplet (n, m, q) represents the number of the level 1 decision variables, the number of the level 2 decision variables, and the number of constraints, respectively. The problems contained both positive and negative coefficients in all data elements. The coefficient matrices were kept dense by allowing only a small percentage of zeros. In all, 15 random instances were generated for each combination. The runs were made on an AT&T PC6300 microcomputer equipped with an Intel 80286 microprocessor and an 80287 math coprocessor.

Performance was evaluated on the basis of computational efficiency measured by CPU time, and solution quality measured by two factors: (i) the percentage of time that one of the algorithms succeeded in producing a better solution (i.e., a smaller level 1 objective at the end of its execution), and (ii) the sum of absolute deviation of the best solutions of each defined by

$$\Delta = \sum_{\ell=1}^L \{F_\ell^*(\text{MGS}) - F_\ell^*(\text{GABBA})\} \quad (9.6)$$

where F_ℓ^* is the best value of F obtained for the ℓ th problem, and L is the total number of problems. Note that if the absolute deviation Δ is positive, it means that the solutions produced by the modified grid search technique were better than those produced by GABBA in an overall sense. Conversely, if Δ was negative, then GABBA was better overall than the MGS.

For two classes of problems, GABBA was run for varying settings of NR (number of new randomly obtained decision vectors) and NX (number of alleles to undergo mutation). The results are shown in Tables 9.1 and 9.2 and the parameter settings are shown in Table 9.3. The settings of NP (number of current structures to undergo mutation), N (population size), and the SCALE modification heuristic (see Step 1 above) were held constant. Specifically, $N = 100$ and $NP = 40$ were used. Also, the stopping criterion ε was set to 10^{-4} . From the results in Tables 9.1 and 9.2, it is not clear what the best version of GABBA is. When measured in terms of average CPU time (in seconds), GABBA1 seems to be best in both problem classes. Compared to

the grid search technique, however, GABBA1 only yielded a better solution 33.3% of the time for problem size (3,7,4), and 50% of the time for problem size (5,10,6). GABBA5 yielded the better solution 67% of the time for problem size (5,10,6) but had almost a 70% higher CPU time compared to GABBA1. The authors decided that the probability of reaching a near-optimal solution was more important than CPU time. Hence, GABBA5, with $N = 100$, $NR = 0.5N$, $NX = 0.5n$, and $NP = 0.4N$, was chosen for all subsequent analyses.

It should be noted that although the authors experimented with some of the other parameter settings in GABBA, they did not attempt to optimize them in the sense described by Grefenstette [G12]. Parameter optimization is an extremely expensive activity which is often very sensitive to problem characteristics and size. This undermines the idea of obtaining optimal settings.

Table 9.1 GABBA results for different parameter settings – problem size (3,7,4)

Algorithm	Avg (CPU)	Var (CPU)	% Better	Δ
Grid Search	4.24	11.34	n.a.	n.a.
GABBA1	17.13	56.31	33.3 %	0.556
GABBA2	26.61	135.30	38.0 %	0.435
GABBA3	26.56	134.630	38.0 %	0.435
GABBA4	20.13	105.40	44.0 %	0.160
GABBA5	24.81	176.70	40.0 %	0.181
GABBA6	24.81	176.79	40.0 %	0.181
GABBA7	24.88	105.41	44.4 %	0.160
GABBA8	24.81	177.58	40.0 %	0.181
GABBA9	24.76	176.95	40.0 %	0.181

As can be seen from the output in Table 9.4, the genetic algorithm provided reasonably good results relative to MGS. Although the grid search algorithm took less time to solve the problems, GABBA yielded a better solution in most instances. For example, for 11 of the 15 problem instances (73%) in the class (6,14,8), GABBA proved superior to the grid search algorithm. This means that GABBA would be more likely to reach a global optimum than would grid search. As in most pseudo-random search techniques, though, computation times accompanying GABBA were very long. This underscores a major disadvantage of these methods.

One reason for the lengthy run times could be due to the absence of crossover in the procedure. In addition, the stopping rule used for GABBA was more stringent than the one used for MGS. It remains to be seen whether algorithmic improvements will lead to better performance.

Table 9.2 GABBA results for different parameter settings – problem size (5,10,6)

Algorithm	Avg (CPU)	Var (CPU)	% Better	Δ
Grid Search	13.68	72.89	n.a	n.a
GABBA1	47.76	388.17	50.0 %	-0.029
GABBA2	67.191	1531.58	38.0 %	0.144
GABBA3	82.68	1562.97	38.0 %	0.295
GABBA4	52.87	764.01	44.0 %	0.153
GABBA5	80.48	2092.73	67.0 %	-0.057
GABBA6	82.551	1880.62	57.0 %	0.075
GABBA7	70.02	1071.84	40.4 %	0.247
GABBA8	99.83	2832.03	57.0 %	-0.026
GABBA9	110.77	3163.03	57.0 %	0.063

Table 9.3 Parameter settings for different algorithms

Algorithm	NR	NX
GABBA1	$0.2N$	$0.2n$
GABBA2	$0.2N$	$0.5n$
GABBA3	$0.2N$	$0.8n$
GABBA4	$0.5N$	$0.2n$
GABBA5	$0.5N$	$0.5n$
GABBA6	$0.5N$	$0.8n$
GABBA7	$0.8N$	$0.2n$
GABBA8	$0.8N$	$0.5n$
GABBA9	$0.8N$	$0.8n$

Table 9.4 GABBA versus grid search algorithm

Problem size	Grid search Avg (CPU)	GABBA Avg (CPU)	GABBA	
			dominates grid search (%)	Δ
(3, 7, 4)	4.23	24.81	40.0 %	0.181
(5, 10, 6)	13.69	80.48	66.6 %	-0.057
(6, 14, 8)	27.27	699.457	73.0 %	-0.135
(8, 17, 10)	63.99	3281.59	60.0 %	-0.593

9.2.5 Simulated Annealing Algorithm (SABBA)

Simulated annealing was derived from statistical mechanics with the aim of finding (near) optimal solutions to large-scale problems. It generalizes hill climbing methods (in the case of maximization) and eliminates their main disadvantage: dependence of the solution on the starting point. Moreover, it statistically promises to deliver a globally optimal solution in the limit. This is achieved by introducing a probability ρ of acceptance (i.e., the replacement of the current point by a new point): $\rho = 1$ if the new point provides a better value of the objective function. In general, $\rho > 0$ depends on the values of the objective function evaluated at the current and the new points, and an additional control parameter known as the *temperature*, denoted by T . The lower the temperature, the smaller the chances for acceptance of the new point. During the execution of the algorithm, T is lowered in steps. Termination occurs for some small value of T for which virtually no changes are accepted anymore. Most of the applications of simulated annealing have targeted combinatorial optimization problems but the technique has been adapted to nonlinear programming problems as well (e.g., see [M13]).

Anandalingam et al. [A8] developed a Simulated Annealing Based Bilevel programming Algorithm (SABBA) in addition to GABBA for solving linear BLPs. SABBA makes use of the fact that for a given \mathbf{x} , the follower's rational reaction $\mathbf{y}(\mathbf{x})$ can be obtained by solving the linear program (9.3). This implies that only components of the vector \mathbf{x} need to be generated randomly. The authors use the same procedures as in Step 1 of GABBA to generate the admissible range for the x_i variables, and then generate specific values from a uniform distribution over this range. The main differences relate to the selection criterion and the tests of acceptability.

Notation

F	objective of the leader as defined by eq. (9.1a) for current solution
F^*	best value of leader's objective function at current iteration
\mathbf{z}^*	array that stores the solution associated with F^*
k^*	maximum allowable iterations with no change in optimal solution
α	temperature reduction parameter
T^*	minimum temperature allowed for annealing schedule

SABBA

- Step 0 (Initialize) Set $F^* = \infty$, temperature $T = T_{max}$, and let $k = 0$.
- Step 1 Generate $x_i \sim U[\min x_i, \max x_i]$ ($i = 1, \dots, n$) in Step 2 of GABBA.
- Step 2 Solve (9.3) for \mathbf{x} generated in Step 1; obtain $\mathbf{x}, \mathbf{y}(\mathbf{x})$) and current value of F .
- Step 3 Compute $\Delta F = F^* - F$:
If $\Delta F > 0$, then $F^* \leftarrow F$, and $\mathbf{z}^* = (\mathbf{x}, \mathbf{y}(\mathbf{x}))$.

If $\Delta F < 0$ let $\mathbf{z}^* = (\mathbf{x}, \mathbf{y}(\mathbf{x}))$ with probability $\exp(-\Delta F/T)$.

If $\Delta F = 0$, put $k \leftarrow k + 1$.

If $k > k^*$, go to Step 4; else go to Step 1.

Step 4 Set $k = 0$ and lower temperature, $T \leftarrow \alpha T$.

If $T < T^*$ or other termination criterion met, STOP with solution \mathbf{z}^* ; else go to Step 1.

The second stopping criterion used in Step 4 is similar to the one used in Step 4 of GABBA except that it is based on $A_F(\varepsilon)$. In either case, global optimality is not guaranteed.

9.2.6 Comparison of SABBA and Grid Search

The same test problems and computer resources that were used in Section 9.2.4 were used to compare SABBA with Bard's grid search algorithm. Hence, the results indirectly provide a comparison of SABBA and GABBA. In the case of SABBA, the authors experimented with a number of values for the parameters k^* , α and T^* . In making the selection, it was necessary to strike a balance between distance from the optimum and time to converge. They finally settled on $k^* = 5$ (no change in F^* for 5 iterations before decreasing the cooling temperature), $T^* = 0.01$ degrees (the lowest permissible temperature), and $\alpha = 0.1$.

The results in Table 9.5, like those in Table 9.4 for GABBA, show that SABBA dominates the grid search algorithm for the larger problems with respect to the leader's objective function value. The average time taken to achieve satisfactory solutions, though, was again longer. The last column in Table 9.5 was computed from eq. (9.6) by replacing the second term with the values of $F_t^*(\text{SABBA})$. Looking at Tables 9.4 and 9.5, we see that GABBA dominates SABBA with respect to both solution quality and CPU time (sec) for all problem classes.

Table 9.5 SABBA versus grid search algorithm

Problem size	Grid search Avg (CPU)	SABBA Avg (CPU)	SABBA dominates grid search (%)		Δ
			grid search (%)	Δ	
(3, 7, 4)	4.23	508.74	33.3 %	0.137	
(5, 10, 6)	13.69	645.42	51.2 %	-0.009	
(6, 14, 8)	27.27	6923.00	64.4 %	-0.111	
(8, 17, 10)	63.99	3879.25	62.2 %	-0.504	

9.2.7 Assessment

The development of GABBA and SABBA represents the first step in circumventing the difficulties associated with nonconvexity and nondifferentiability that plague standard mathematical programming techniques when applied to bilevel programs. Nevertheless, the results are not particularly satisfying when compared to exact methods such as vertex enumeration (Section 5.3.1), the Kuhn-Tucker approach (Sections 5.3.2 – 5.3.4), or the penalty method (Section 5.3.5). As can be seen from the results reported in Table 5.5 and elsewhere in Section 5.4, GABBA is more than two orders of magnitude slower than any of these algorithms.

Currently, exact methods for the linear BLPP can solve instances with up to 250 leader variables, 150 follower variables, and 150 follower constraints in less than an hour of CPU time. In this same amount of time, Mathieu et al. were only able to solve problems with 8 leader variables, 17 follower variables, and 10 constraints. But even for these relatively small instances, no guarantee of global (or even local) optimality could be made. This is the nature of heuristics. The densities of the coefficient matrices and the hardware used in the computations may account for a portion of this imbalance; however, unless the accompanying CPU times can be drastically reduced, the use of these heuristics cannot be justified for solving linear BLPPs. Looking at the results associated with the nonlinear BLP branch and bound algorithm in Section 8.2, a similar conclusion can be drawn even when nonlinearities are present.

9.3 HYBRID TABU-DESCENT ALGORITHM

Tabu search is a heuristic methodology developed by Fred Glover [G9, G10] to find and improve upon feasible solutions to integer programming problems. Although it is not guaranteed to yield a global optimum, a central component of the methodology is an adaptive mechanism designed to overcome local optimality. To date, there have been hundreds of successful implementations aimed at solving the most challenging combinatorial problems but there have been few implementations associated with nonlinear models. In this section, we present the work of Gendreau et al. [G5] who developed a tabu search algorithm for the linear BLPP.

Their algorithm is composed of three main building blocks: a *startup phase* designed to produce a good initial solution, a *local descent* phase, and a *tabu* phase whose aim is to move away from and improve upon the current local optimum. Each of these phases is described below with respect to the BLPP given by (9.1a)–(9.1c).

9.3.1 Initialization Procedure

Recall that for a given \mathbf{x} in the rational reaction set $P(\mathbf{x})$, a vector \mathbf{y} is optimal for the lower-level problem if and only if it satisfies, together with a dual vector \mathbf{u} :

$$\begin{aligned} \mathbf{B}\mathbf{y} &\leq \mathbf{b} - \mathbf{A}\mathbf{x} && \text{primal} \\ \mathbf{y} &\geq \mathbf{0} && \text{feasibility} \\ \mathbf{u}\mathbf{B} &\geq -\mathbf{d}_2 && \text{dual} \\ \mathbf{u} &\geq \mathbf{0} && \text{feasibility} \\ (\mathbf{u}\mathbf{B} + \mathbf{d}_2)\mathbf{y} &= \mathbf{0} && \text{complementarity} \\ \mathbf{u}(\mathbf{b} - \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{y}) &= \mathbf{0} && \text{slackness} \end{aligned}$$

Taking the Kuhn-Tucker approach discussed in Section 5.3.2, if the above primal-dual optimality conditions are substituted for the lower-level problem, one obtains the equivalent single-level formulation:

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{y}, \mathbf{u}} \quad & c_1\mathbf{x} + d_1\mathbf{y} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b} \\ & \mathbf{x}, \mathbf{y} \geq \mathbf{0} \\ & \mathbf{u}\mathbf{B} \geq -\mathbf{d}_2 \\ & \mathbf{u} \geq \mathbf{0} \\ & (\mathbf{u}\mathbf{B} + \mathbf{d}_2)\mathbf{y} = \mathbf{0} \\ & \mathbf{u}(\mathbf{b} - \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{y}) = \mathbf{0}. \end{aligned}$$

Following the idea of Anandalingam and White [A9], the complementarity slackness terms can be placed in the objective function as penalty terms to obtain the linearly constrained single-level program parameterized in M :

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{y}, \mathbf{u}} \quad & c_1\mathbf{x} + d_1\mathbf{y} + M[(\mathbf{u}\mathbf{B} + \mathbf{d}_2)\mathbf{y} + \mathbf{u}(\mathbf{b} - \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{y})] \\ & = c_1\mathbf{x} + d_1\mathbf{y} + Md_2\mathbf{y} + Mu(\mathbf{b} - \mathbf{A}\mathbf{x}) \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b} \\ & \mathbf{u}\mathbf{B} \geq -\mathbf{d}_2. \\ & \mathbf{x}, \mathbf{y}, \mathbf{u} \geq \mathbf{0} \end{aligned} \tag{9.7}$$

Whenever M is sufficiently large, the penalty is exact in the sense that problem (9.7) and the original BLPP (9.1) admit the same solution sets. If an optimal dual vector \mathbf{u} were known a priori, (9.7) would reduce to a standard linear program. The proposed heuristic estimates \mathbf{u} in the simplest fashion by setting it, for a given upper-level vector \mathbf{x} , to an optimal dual solution of the lower-level problem. The resultant problem is then solved with respect to the \mathbf{x} and \mathbf{y} variables. If the corresponding solution vector \mathbf{y} is not in $P(\mathbf{x})$, the penalty parameter M is increased and the procedure repeated. The rationale behind this strategy is to generate a sequence of

points converging to $(\mathbf{x}^0, \mathbf{y}^0)$ that satisfies the condition $\mathbf{y}^0 \in P(\mathbf{x}^0)$, while favoring the upper level's objective. This parametric scheme is reminiscent of Bard's efficient point algorithm [B2] which solves the weighted problem

$$\begin{aligned} & \min_{\mathbf{x} \geq 0, \mathbf{y} \geq 0} \quad c_1 \mathbf{x} + d_1 \mathbf{y} + M d_2 \mathbf{y} \\ & \text{subject to } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b} \end{aligned}$$

for increasing values of the parameter M until a rational point $(\mathbf{x}^0, \mathbf{y}^0)$ is identified (note that the above problem is an alternative representation of (9.5)). This strategy, however, only finds Pareto-optimal minima which may be not be globally optimal (see [H4] for further discussion). The difficulty can be traced to the absence of the correcting term $M\mathbf{u}(\mathbf{b} - \mathbf{Ax})$ in (9.7), and may lead to poor feasible solutions.

The initialization algorithm, called Algorithm_INIT, is now described. In Step 1, S is the BLPP constraint region (see Definition 5.1.1) and in Step 2, D is the set of dual feasible solutions defined above.

Algorithm_INIT

- Step 0 Select a parameter ΔM and set M to 0.
- Step 1 Let $(\mathbf{x}(M), \mathbf{y}(M)) \in \arg \min \{c_1 \mathbf{x} + d_1 \mathbf{y} + M[d_2 \mathbf{y} + \mathbf{u}(M)(\mathbf{b} - \mathbf{Ax})] : (\mathbf{x}, \mathbf{y}) \in S\}$.
- Step 2 Let $\mathbf{u}(M) \in \arg \min \{\mathbf{u}(\mathbf{b} - \mathbf{Ax}(M)) : \mathbf{u} \in D\}$.
- Step 3 If $d_2 \mathbf{y}(M) + \mathbf{u}(M)(\mathbf{b} - \mathbf{Ax}(M)) = 0$ then go to Step 4;
Else put $M \leftarrow M + \Delta M$ and go to Step 1.
- Step 4 Output the solution $(\mathbf{x}(M), \mathbf{y}(M))$.

At Step 2, the following linear program

$$\begin{aligned} & \min_{\mathbf{y} \geq 0} \quad d_2 \mathbf{y} \\ & \text{subject to } \mathbf{B}\mathbf{y} \leq \mathbf{b} - \mathbf{Ax}(M) \end{aligned}$$

is solved to obtain an optimal primal solution $\mathbf{y}(\mathbf{x}(M))$ as well as the optimal dual vector $\mathbf{u}(M)$.

Several variants of Algorithm_INIT were investigated by the authors but proved less effective. For example, they tried outputting the solution $(\mathbf{x}(M), \mathbf{y}(\mathbf{x}(M)))$ which yielded the smallest value of the function $c_1 \mathbf{x}(M) + d_1 \mathbf{y}(\mathbf{x}(M))$ but this solution did not always correspond to the largest value of the penalty parameter M . They also tried, at a somewhat higher computational cost, to solve the bilinear program associated with a given value of M . This problem, though, is as difficult theoretically as the original bilevel program implying that a local minimum is as much as can be

expected. One way to obtain a local minimum is by iteratively solving linear programs with respect to the vectors (\mathbf{x}, \mathbf{y}) and \mathbf{u} in the manner of Gauss-Seidel.

The initialization phase is completed with a local search that loops through the first three steps. In particular, pivots in the (\mathbf{x}, \mathbf{y}) -space are performed in an effort to improve the leader's objective function while preserving the rationality of \mathbf{y} . Termination occurs when no (local) improvement can be found. In the implementation, the procedure might halt before a local minimum is actually identified because degenerate pivots are not considered. To guarantee local optimality, it would be necessary to explore all neighbors of the current vertex, which are exponential in number.

Remark: The lower-level problem is, by nature, highly degenerate. This follows because if there is an optimal solution to a BLP there is one that occurs at an extreme point of S . Consequently, a basic solution vector \mathbf{y} of the lower-level problem must have at least k zero components, where k is the number of nonzero components of the upper-level decision vector \mathbf{x} , including the slack variables associated with the constraints $\mathbf{Ax} + \mathbf{By} \leq \mathbf{b}$.

9.3.2 Tabu Phase

Starting at a point $(\mathbf{x}^0, \mathbf{y}^0)$ in the inducible region, the aim of the tabu phase of the algorithm is to determine another point $(\mathbf{x}^+, \mathbf{y}^+)$ in the inducible region such that

$$c_1 \mathbf{x}^+ + d_1 \mathbf{y}^+ = c_1 \mathbf{x}^0 + d_1 \mathbf{y}^0$$

This is illustrated in Fig. 9.1. The corresponding primal-dual nonconvex feasibility problem is

$$\begin{aligned} c_1 \mathbf{x}^+ + d_1 \mathbf{y}^+ &= c_1 \mathbf{x}^0 + d_1 \mathbf{y}^0 \\ \mathbf{Ax}^+ + \mathbf{By}^+ &\leq \mathbf{b} \\ \mathbf{x}^+, \mathbf{y}^+, \mathbf{u}^+ &\geq 0 \\ \mathbf{u}^+ \mathbf{B} &\geq -\mathbf{d}_2 \\ \mathbf{u}^+ (\mathbf{b} - \mathbf{Ax}^+) &= -\mathbf{d}_2 \mathbf{y}^+ \\ (\mathbf{x}^+, \mathbf{y}^+) &\neq (\mathbf{x}^0, \mathbf{y}^0) \end{aligned} \tag{9.8}$$

By introducing the *gap function*

$$g(\mathbf{x}, \mathbf{y}) = \max_z \{ d_2(\mathbf{y} - z) : \mathbf{Bz} \leq \mathbf{b} - \mathbf{Ax}, z \geq 0 \}$$

(9.8) can be rewritten as

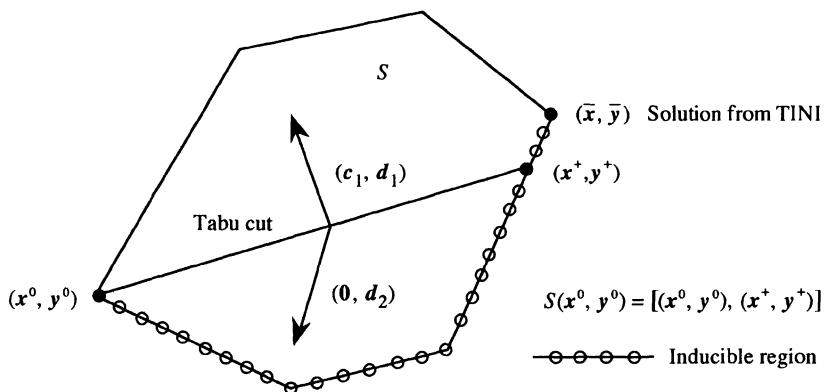


Figure 9.1 Example of a tabu cut

$$\begin{aligned}
 0 &= \text{global min}_{\mathbf{x}, \mathbf{y}} g(\mathbf{x}, \mathbf{y}) \\
 \text{subject to} \quad & c_1 \mathbf{x} + d_1 \mathbf{y} = c_1 \mathbf{x}^0 + d_1 \mathbf{y}^0 \\
 & (\mathbf{x}, \mathbf{y}) \in S \\
 & (\mathbf{x}, \mathbf{y}) \neq (\mathbf{x}^0, \mathbf{y}^0)
 \end{aligned} \tag{9.9}$$

Notice that unless $(\mathbf{x}^0, \mathbf{y}^0)$ is already globally optimal for the BLPP, there exists at least one solution to the system (9.8). The generic procedure is described below.

Algorithm.TABU

Step 1 (Moving away from the current solution $(\mathbf{x}^0, \mathbf{y}^0)$)

Through a pivot sequence, generate a point $(\mathbf{x}', \mathbf{y}')$ which is

- (i) “far” from $(\mathbf{x}^0, \mathbf{y}^0)$, and
- (ii) achieves a “high” value of the gap function g .

Record the relevant tabu information.

Step 2 (Searching for a point in the inducible region)

Starting from $(\mathbf{x}', \mathbf{y}')$, solve problem (9.9) to get $(\mathbf{x}^+, \mathbf{y}^+) \in IR$.

Elaborating on the implementation, at Step 1 a random move is made to an adjacent vertex $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ of $(\mathbf{x}^0, \mathbf{y}^0)$ in the polyhedron

$$S(\mathbf{x}^0, \mathbf{y}^0) = \{(\mathbf{x}, \mathbf{y}) \in S : c_1 \mathbf{x} + d_1 \mathbf{y} = c_1 \mathbf{x}^0 + d_1 \mathbf{y}^0\}$$

Next, the squared distance

$$\|(\mathbf{x}, \mathbf{y}) - (\mathbf{x}^0, \mathbf{y}^0)\|^2 \tag{9.10}$$

is maximized by performing one simplex pivot on the linear program resulting from the linearization of the convex function (9.10) at the point $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$. This process is repeated until a local maximum is reached. Because the distance function (9.10) is convex, a finite sequence of vertices of $S(\mathbf{x}^0, \mathbf{y}^0)$ is generated by the calculations.

The tabu methodology comes into play at Step 2. Three types of tabu tags are maintained throughout this phase: (1) tags on entering variables, (2) tags on leaving variables, and (3) tags on pivots, i.e., pairs of entering and leaving variables. The first two sets of tags prevent the reversal of past moves, while the third tag inhibits pivot repetitions. All remain active for a fixed number of iterations whose values are generated randomly and uniformly within the intervals $[\underline{\theta}_{in}, \bar{\theta}_{in}]$, $[\underline{\theta}_{out}, \bar{\theta}_{out}]$ and $[\underline{\theta}_{piv}, \bar{\theta}_{piv}]$, respectively.

Each iteration of the tabu phase involves a move from a vertex of $S(\mathbf{x}^0, \mathbf{y}^0)$ to an adjacent vertex by means of a pivot operation. Only a subset of possible moves is considered at each iteration. The candidate list is made up of the k_1 most promising nonbasic variables obtained at the previous iteration, and is completed by selecting k_2 variables according to a cyclic management scheme. Hence, at each iteration, the length of the candidate list is at most $k = k_1 + k_2$. For each of these variables, a “merit score” defined as the sum of two terms: the gap value that would result if the pivot were actually carried out and a penalty factor related to the tabu status of the corresponding move, is computed. For a pivot involving r as the entering variable and s as the leaving variable, the penalty is given by the formula

$$\Pi(r, s) = \alpha([t_{in}(r) - t]^+ + [t_{out}(s) - t]^+ + 2[t_{piv}(r, s) - t]^+),$$

where α is a penalty weight factor whose value decreases from one to zero as the number of tabu iterations already performed increases, t is the current iteration index, $t_{in}(r)$ the instant (iteration index) at which the variable r is to be removed from the tabu list associated with the entering variables, $t_{out}(s)$ the instant at which the variable s is to be removed from the tabu list associated with the leaving variables, $t_{piv}(r, s)$ the instant at which the pair (r, s) is removed from the pivot tabu list, and $[\cdot]^+$ denotes the maximum function: $[u]^+ = \max\{0, u\}$. Note that at the beginning of the tabu step, the instants $t_{in}(r)$, $t_{out}(s)$ and $t_{piv}(r, s)$ are set to zero for all indices r and s .

While scanning the list of candidate pivots, the first (r, s) pivot that satisfies either of the following conditions is implemented

1. (r, s) is not tabu and decreases the gap, or
2. (r, s) is tabu and significantly decreases the gap (aspiration criterion). More precisely, (r, s) is implemented if the gap resulting from the (r, s) pivot is less than

$$\frac{1}{2} \times \text{current gap} \times \left(1 - \frac{\text{maxit} - t}{\text{maxit}}\right)$$

where ‘maxit’ is the maximum number of iterations allowed in the tabu phase and t the current iteration index.

According to standard terminology, this second condition is called an “aspiration criterion” in the sense that the tabu status of a pivot can be overridden if such action would allow the search to reach a particularly promising solution. If no pivot meets the previous two requirements, then a pivot is selected that minimizes the previously defined merit score.

If, during the tabu phase the starting point $(\mathbf{x}^0, \mathbf{y}^0)$ is re-encountered, the parameters $[\underline{\theta}_{in}, \bar{\theta}_{in}]$, $[\underline{\theta}_{out}, \bar{\theta}_{out}]$ and $[\underline{\theta}_{piv}, \bar{\theta}_{piv}]$ are increased and the process restarted. In a symmetric fashion, if the gap value increases on two consecutive iterations these parameters are decreased.

Once the tabu phase is successfully completed, an attempt is made to improve on $(\mathbf{x}^+, \mathbf{y}^+)$ using the procedure TINI described below. This algorithm is simply a reverse implementation of Algorithm_INIT.

Algorithm_TINI

- Step 0 Select a parameter ΔM .
Set M to some suitably large value and let $\mathbf{u}(M)$ be an optimal dual vector corresponding to the primal lower-level vector $(\mathbf{x}^+, \mathbf{y}^+)$.
- Step 1 Let $(\mathbf{x}(M), \mathbf{y}(M)) \in \arg \min \{c_1 \mathbf{x} + d_1 \mathbf{y} + M[d_2 + \mathbf{y}\mathbf{u}(M)(\mathbf{b} - \mathbf{Ax})] : (\mathbf{x}, \mathbf{y}) \in S\}$.
- Step 2 Let $\mathbf{u}(M) \in \arg \min \{\mathbf{u}(\mathbf{b} - \mathbf{Ax}(M)) : \mathbf{u} \in D\}$.
- Step 3 If $d_2 \mathbf{y}(M) + \mathbf{u}(M)(\mathbf{b} - \mathbf{Ax}(M)) \neq 0$ then go to Step 4;
Else set $M = M - \Delta M$ and go to Step 1.
- Step 4 Output the feasible solution $(\mathbf{x}(M + \Delta M), \mathbf{y}(M + \Delta M))$.

At Step 0, “suitably large” means that M is an exact penalty parameter implying that any solution of the optimization problem (9.7) achieves a null gap value. Like Algorithm_INIT, most of the effort in Algorithm_TINI is spent trying to achieve local optimality. The point $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ in Fig. 9.1 represents a local solution obtained from TINI. The overall procedure is illustrated in Fig. 9.2.

9.3.3 Numerical Results

The complete algorithm was programmed in FORTRAN using the LP code XMP [M6] to solve the linear subproblems encountered throughout. All computations were done on an HP730 UNIX workstation. For the smaller test problems, comparisons were

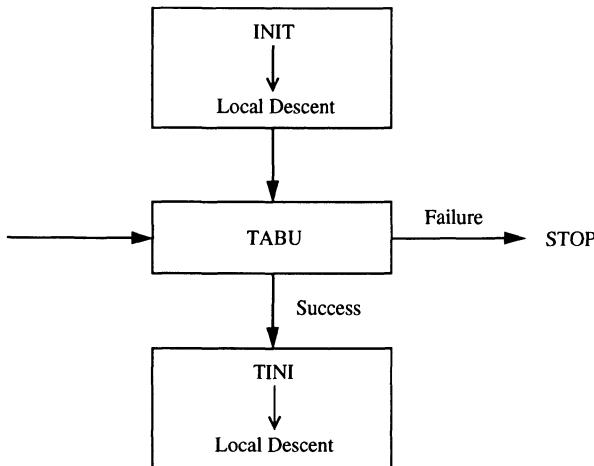


Figure 9.2 Hybrid tabu–descent algorithm

made with the results obtained from the exact algorithm of Hansen, Jaumard and Savard (see Section 5.3.4 or [H1]).

The test problems examined ranged in size from 40 to 200 variables and 20 to 200 constraints, with matrix densities between 0.4 and 1. Specific instances were generated using a modification of the Bard-Moore generator [B11] with ‘max’ rather than ‘min’ objective functions. The experimental results are presented in Tables 9.6 to 9.11. The values in Tables 9.6 to 9.9 are averaged over groups of 10 problem instances.

The meaning of the column headings is given below:

HTA:	hybrid tabu–descent algorithm
EXACT:	exact algorithm HJS
<i>n</i> :	number of upper–level variables
<i>m</i> :	number of lower–level variables
<i>q</i> :	number of constraints
DENS:	density of the constraint matrix $[A \mid B]$
$\bar{\theta}$:	common value of the parameters $\bar{\theta}_{in}$, $\bar{\theta}_{out}$, $\bar{\theta}_{piv}$
$\underline{\theta}$:	common value of the parameters $\underline{\theta}_{in}$, $\underline{\theta}_{out}$, $\underline{\theta}_{piv}$
maxit:	maximum number of iterations allowed in each tabu phase
NS:	number of successful tabu phases
NOPT:	number of problems solved to optimality
WORST:	worst heuristic-to-optimal ratio
AVG:	average heuristic-to-optimal ratio

CPU0:	CPU time (sec) associated with Algorithm.INIT (including local search phase)
AVG:	average heuristic-to-optimal ratio
CPU0:	CPU time (sec) associated with Algorithm.INIT (including local search phase)
CPU1:	CPU time (sec) associated with the tabu-descent algorithm
CPU2:	CPU time (sec) associated with the exact algorithm HJS
k :	length of the list of candidate variables, sorted with respect to their respective merit scores
k_1 :	number of most promising variables kept from the previous iteration in tabu phase
OBJ:	value of upper-level objective
OBJ0:	value of upper-level objective achieved by Algorithm.INIT (including local descent phase)
HEUR:	value of upper-level objective computed by HJS's initialization procedure
UP:	initial upper bound on the upper-level (maximization) objective

The numerical experiments were conducted in two stages. First, a series of tests was performed to determine good values for the various parameters involved in the tabu phase of the algorithm. Problems with 50 upper-level variables, 55 lower-level variables, 42 constraints and matrix densities of 100% were used for these tests. The resulting problems were of size and difficultly that one may expect to encounter in several practical situations; i.e., the BLP formulation of the nonparametric discriminant analysis problem leads to 100% dense constraint matrices (see [M5]). In the second stage, full-scale testing was done on a wide range of problems.

The first observation was that in Algorithm.INIT, the penalty became exact for values of M between 2 to 45. In Algorithm.TINI, the parameter M was initially set to 5 and increased by increments of 0.5 until the penalty became exact. In both cases, the value of ΔM was set to 0.5.

In Table 9.6, an assessment is given on the influence of the parameter 'maxit' on the performance of the algorithm. As expected, the quality of the solution improves as maxit and CPU1 increase. Nevertheless, it can be seen that increasing maxit above the value 110 (twice the number of second-level variables) led to minute improvements in the solution while considerably increasing the computing time. In subsequent tests, maxit was set to twice the number q of lower-level variables.

Next, a strategy was determined for restricting the number of allowable pivots by introducing two parameters. The first, k , governs the number of variables kept from the previous tabu iteration, sorted with respect to their merit scores; and the second,

Table 9.6 Influence of the parameter “maxit”

Problem parameters						
$n = 50, m = 55, q = 42$						
DENS=1.0, $\underline{\theta} = 11, \bar{\theta} = 15$						
$k = 55, k_1 = 10$						
maxit	NS	NOPT	WORST	AVG	CPU1	CPU2/CPU1
55	3	6	97.59	99.37	12.82	22.88
110	6	7	98.16	99.71	28.70	10.22
220	6	9	98.00	99.80	64.16	4.57

k_1 , controls the number of variables from the previous list of k variables to be evaluated in priority as entering variables. For each of these parameters, 3 different values were considered. The results are shown in Tables 9.7 and 9.8. Based on these findings, the values $k = m$ and $k_1 = 10$ were retained. Note that the algorithm proved quite insensitive to the value of the parameter k_1 . Also, the authors claim that no benefit was gained by using the full list of variables. As k was increased from 55 to 105, the computing time increased and the solution deteriorated.

Table 9.7 Influence of parameter k

Problem parameters						
$n = 50, m = 55, q = 42$						
DENS=1.0, $\underline{\theta} = 11, \bar{\theta} = 15$						
$k_1 = 10, \text{maxit}=110$						
k	NS	NOPT	WORST	AVG	CPU1	CPU2/CPU1
27	3	5	97.59	99.20	16.85	17.41
55	6	7	98.16	99.71	28.70	10.22
105	4	6	97.84	99.43	39.99	7.33

A sensitivity analysis was also conducted with respect to the length of the tabu list. The results evidenced little or no sensitivity to this parameter primarily because of the adaptive nature of the algorithm coupled with the use of an aspiration criterion.

The main results are presented in Table 9.9 where it is observed that the computing time CPU1 of the tabu-descent algorithm increases superlinearly with respect to the problem size, albeit at a much slower rate than the computing time CPU2 of the exact procedure HJS. With the exception of the smaller instances, CPU1 is noticeably less than CPU2. An exact solution was obtained by the tabu search methodology in 184 out of 250 problems, corresponding to a success rate of 73.6%. Even more impressive was the fact that the relative error was, on average, less than 0.5% and seldom greater

Table 9.8 Influence of parameter k_1

Problem parameters						
$n = 50, m = 55, q = 42$						
DENS=1.0, $\theta = 11, \bar{\theta} = 15$						
$k = 55, \text{maxit}=110$						
k_1	NS	NOPT	WORST	AVG	CPU1	CPU2/CPU1
0	5	7	98.15	99.71	31.90	9.20
10	6	7	98.16	99.71	28.70	10.22
20	4	6	97.59	99.37	26.39	11.12

to 5%. With respect to the computation burden, these results were achieved in times 6 to 20 faster than those of the exact algorithm for all but the smallest problems. In several instances, the initialization algorithm INIT produced an optimal solution. In the next-to-last series of test problems ($n = 30, m = 70, q = 56$), the performance of the heuristic was actually superior to that of the exact algorithm. This was due to one extremely nasty instance of a 70×70 problem, which had to be halted when CPU2 exceeded the 15,000 sec time limit.

Table 9.10 contains a detailed account of the test series corresponding to the parameters $n = 60, m = 65, q = 50$ and DENS= 0.4. The aggregated results for these problems are contained near the bottom of Table 9.9. In five instances, the best solution was achieved at the end of the initialization phase, and two of these solutions were optimal. In the other five instances, where the tabu phase improved on the initial solution, the optimal solution was reached. In the two cases where the exact algorithm's computing time CPU2 was less than CPU1, Algorithm INIT produced either an optimal or a near-optimal (within 0.03% of optimality) solution. Computation times for INIT are reported in the 3rd column of Table 9.10. These values are very low except for the 4th instance where the lower-level problem was highly degenerate. Finally, we note that the coefficient of variation associated with CPU1 is very small, confirming the stability of the methodology.

Table 9.11 contains the experimental results for 10 large problems for which a time limit of one hour was imposed on the exact algorithm HJS. In all but one instance, the hybrid tabu-descent procedure produced the best solutions, and in less than 13 minutes. In the other instance (problem 9), the solutions were identical and the running times comparable. In five instances, the best solution was obtained at the end of the initialization phase.

Table 9.9 Main results

Problem parameters $\underline{\theta} = \lceil m/5 \rceil$, $\bar{\theta} = \underline{\theta} + 4$ $k = m$, $k_1 = 10$, maxit=2m										
<i>n</i>	<i>m</i>	<i>q</i>	DENS	maxit	NS	NOPT	WORST	AVG	CPU1	CPU2
30	30	24	0.4	60	3	7	93.67	99.23	3.77	7.19
30	30	24	0.7	60	6	8	85.14	98.35	3.26	7.94
30	30	24	1.0	60	4	9	99.50	99.95	3.15	6.94
30	35	26	0.4	70	3	8	89.61	98.91	4.45	11.65
30	35	26	0.7	70	2	7	78.58	97.75	4.40	29.36
30	35	26	1.0	70	6	10	100.00	100.00	5.88	31.77
40	40	32	0.4	80	4	7	95.19	99.17	7.68	29.34
40	40	32	0.7	80	5	8	97.41	99.66	8.31	109.12
40	40	32	1.0	80	3	8	96.86	99.48	9.16	80.08
40	45	34	0.4	90	6	6	93.29	99.31	15.45	87.40
40	45	34	0.7	90	4	8	97.11	99.66	13.24	64.42
40	45	34	1.0	90	4	5	89.08	97.41	11.23	97.65
50	50	40	0.4	100	7	6	98.70	99.75	24.16	167.77
50	50	40	0.7	100	4	7	91.50	98.36	22.71	355.31
50	50	40	1.0	100	2	8	98.56	99.77	16.94	91.52
50	55	42	0.4	110	13	7	98.68	99.74	45.91	330.48
50	55	42	0.7	110	2	9	99.96	99.99	24.55	199.76
50	55	42	1.0	110	5	7	98.16	99.71	28.70	293.34
60	60	48	0.4	120	4	8	97.60	99.74	41.89	508.87
60	60	48	0.7	120	6	7	96.49	99.53	49.63	1680.99
60	60	48	1.0	120	3	9	96.89	99.69	35.97	585.62
60	65	50	0.4	130	6	7	98.01	99.64	69.86	1699.93
60	65	50	0.7	130	2	7	93.83	99.17	100.61	659.36
70	70	56	0.4	140	6	7	99.12	100.29	88.56	1728.66
80	70	60	0.4	140	3	5	94.35	99.00	95.80	1163.10

9.3.4 Discussion

The hybrid tabu-descent heuristic performed quite well on a large number of difficult test problems involving up to 200 variables and 200 constraints. Optimal solutions were identified for the vast majority of instances while relative errors for the remainder were generally quite small. Moreover, computation times proved to be low and stable from one problem to the next.

A surprisingly effective component of the algorithm was the primal-dual startup procedure INIT which produced optimal or near-optimal solutions a vast majority of the time. When optimality was not achieved, the tabu phase was successful in improving

Table 9.10 Detailed results

#	OBJ0	CPU0	HTA		CPU1	HEUR	EXACT		CPU2
			NS	OBJ			UP	OBJ	
1	60.63	10.11	0	60.63	62.93	33.11	61.62	60.63	338.63
2	75.85	3.43	0	75.85	67.76	37.05	82.12	76.17	1257.61
3	75.16	5.20	2	76.38	96.97	39.96	81.61	76.38	793.65
4	64.17	80.08	1	65.28	142.84	33.65	68.10	65.28	579.28
5	79.00	7.64	1	79.93	54.95	63.25	82.14	79.93	551.09
6	79.47	1.79	0	79.47	31.07	43.05	79.84	79.47	6.00
7	62.24	3.53	1	62.26	81.79	44.71	62.38	62.26	47.27
8	84.18	8.95	0	84.18	52.75	34.82	91.15	85.20	11350.11
9	71.53	5.96	1	71.97	57.29	49.69	75.01	71.97	115.76
10	85.18	1.39	0	85.18	50.28	63.14	90.88	86.92	1959.94

Table 9.11 Results for large problems

#	OBJ0	HTA		CPU1	HEUR	EXACT			CPU2
		NS	OBJ			UP	OBJ	CPU2	
1	78.72	1	78.92	453.32	60.01	81.41	77.73	3600	
2	118.02	0	118.02	274.72	89.28	118.37	116.67	3600	
3	98.36	0	98.36	200.36	46.24	110.23	95.91	3600	
4	80.96	0	80.96	331.11	53.00	89.91	77.87	3600	
5	71.37	1	71.42	358.74	59.04	72.70	71.42	3600	
6	92.85	1	96.90	752.22	68.03	100.87	93.17	3600	
7	69.14	1	70.15	604.44	31.68	73.07	69.54	3600	
8	92.54	0	92.54	183.21	60.41	94.40	80.50	3600	
9	92.99	0	92.99	257.59	83.62	93.59	92.99	228.21	
10	120.01	1	122.55	679.77	82.58	124.77	107.70	3600	

upon the initial solutions. In most cases where the startup procedure did not find an optimal solution, the tabu phase did.

The results in this section demonstrate the superiority of the proposed methodology over the simulated annealing and genetic algorithm implementations presented earlier in the chapter. A logical next step would be to develop and evaluate the capabilities of the hybrid approach with respect to nonlinear bilevel programs, beginning perhaps with formulations containing bilinear and convex objective functions, and linear constraints. A second avenue of research might involve the implementation of more advanced features such as strategic oscillation and clustering analysis, as well as the development of probabilistic and reactive extensions to the basic tabu methodology.

PART III

APPLICATIONS

TRANSPORTATION NETWORK DESIGN

10.1 INTRODUCTION

One of the first applications of bilevel programming (BLP) centered on the design of transportation networks. This chapter deals with the construction of such a model for use in optimizing the investment in the inter-regional highway network of Tunisia, a developing country in North Africa. It is based on the work of Ben-Ayed et al. [B19].

Tunisia is a 164,000 square kilometer country bounded on the north and east by the Mediterranean sea, on the southeast by Libya and on the west by Algeria. In the 1980s, the population was about 7 million with 47% rural and less than 15% living in the south. The per capita income was about US \$1,100 and the inflation rate averaged nearly 10.5% per year. At the time of the study, the highway network consisted of 18,142 kilometers (km) of roads of which 53% were paved. The average width of a paved road was 5.5 meters (m); 17% were narrower than 4.5 m. The total number of vehicles in the country was estimated at 385,600.

Budget allocations among road links in developing countries are often based on simple rules, such as the comparison of recent increases of daily traffic on the links to be improved or the comparison of the differences between benefits and costs for the projected improvements. Network-based approaches are essentially unknown. The high cost of highway improvements motivates the use of more sophisticated analytic techniques to better allocate the available budget and to take into account the specific characteristics of the transportation networks in those countries. Such characteristics are very important, especially in rural networks where the difference between developed and developing is so significant that the models elaborated for one category can rarely be applied to the other. For example, models measuring capacity in number of lanes do not make sense in a developing country where it is uncommon to find a rural highway with more than two lanes; moreover, a high proportion of the roads are not paved.

While a great deal of effort was made by the authors to make the presentation as realistic as possible, some liberties were taken with the data, both to fill in gaps and to simplify the model. A mixture of old and new data was used, and for some parts of the model no data existed at all. In such a case, either information obtained from other countries was adapted or careful subjective estimates were made.

10.2 RURAL HIGHWAY NETWORK

The highway network in developing countries basically consists of two-lane paved and unpaved roads. Two-lane roads are by definition undivided, meaning that the passing of slower vehicles requires the use of the opposing lane when sight distance and gaps in the opposing traffic stream permit. The traffic flow in one direction influences the flow in the other direction, which requires the inclusion of the total flow in both directions in all travel time functions.

A significant part of the analysis is built on data from the Highway Capacity Manual (HCM) [T4] which uses the concept of level of service (LOS), a qualitative measure describing operational conditions within a traffic stream and their perception by the travelers. Six levels of service are defined from A to F; level of service A represents the best operating conditions (nearly free flow) and level of service F, the worst (forced or breakdown flow). Level of service E represents operating conditions at or near capacity.

The HCM provides maximum values of the ratio of flow to capacity measured under ideal conditions (ideal capacity). Such conditions, as defined by the HCM, are nonrestrictive geometric, traffic, or environmental conditions. Specifically, they state that (1) design speed is greater than or equal to 60 miles per hour (97 km/h), (2) roadway width is greater than or equal to 24 feet (7.3 m), (3) width of both usable shoulders is greater than or equal to 12 feet (3.7 m), (4) “no passing zones” are not used on the highway, (5) all vehicles are passenger, (6) the directional split of traffic is 50/50, (7) there are no impediments to through traffic due to traffic control or turning vehicles, and (8) the terrain is level.

Whenever ideal conditions are not satisfied, adjustments are made using the following relationship:

$$X_i = 2800 \times R_i \times D \times W \times H \quad (10.1)$$

Here, X_i is the service flow in both directions for the prevailing roadway and traffic conditions for level of service i in passenger cars units (PCU) per hour. Service flow is the maximum flow attainable for the assumed roadway and traffic conditions. In (10.1), R_i is the ratio of service flow to ideal capacity for level of service i , obtained from Table 8-1 in the HCM; D is an adjustment factor for directional distribution of traffic obtained from Table 8-4 in the HCM; W is an adjustment factor for narrow

lanes and restricted shoulder widths obtained from Table 8-5; and H is an adjustment factor for the presence of heavy vehicles in the traffic stream. H is computed as follows:

$$H = [1 + p_T(e_T - 1) + p_R(e_R - 1) + p_B(e_B - 1)]^{-1}$$

where p_T , p_R and p_B are the proportions of trucks, recreational vehicles and buses in the traffic stream, respectively. Also, e_T is the PCU equivalent of one truck (passenger cars refer to all vehicles having exactly four wheels contacting the road, including light vans and pickups truck), e_R is the PCU equivalent to one recreational vehicle, and e_B is the PCU equivalent to one bus; e_T , e_R and e_B are obtained from Table 8-6 in the HCM. Under ideal conditions and level of service E, relationship (10.1) gives an ideal service flow of 2800 PCU; for different conditions, the maximum service flow can be found by application of the formula.

The data suggested by the HCM can be applied to developing countries; however, some extensions are necessary. Table 8-5 does not consider roadways narrower than 5.5 m, whereas 4 m roadways are very common in developing countries. Moreover, the HCM ignores the condition of the road surface, even though the surface has an important impact on many factors such as capacity and operating costs. Surfaces can be classified into three categories: asphaltic concrete overlay (hereafter referred to as asphaltic), surface treatment (hereafter referred to as treatment), and unpaved. Every road in each surface category can be ranked as good, fair or poor, which gives a total of nine surface types.

The surface of the shoulders as well as the roadway can be included in a capacity analysis. The surface of the shoulders is almost always of lower quality; when it is the same as the roadway, the road is considered to consist of a broader roadway with no shoulders. This is usually the case for unpaved roads. According to Table 8-5 in the HCM, widening of shoulders beyond 12 feet (3.7 m) has no effect on increasing capacity. This result is applicable to thoroughfares with wide roadways (more than 5.5 m); however, when roads with narrower roadways are considered, the effect of wider shoulders is significant. The quality of the surface can be given by two other tables, one for the roadway and the other for the shoulders. It is important to notice that the effect of the surface of the shoulders depends on their width: the narrower the shoulders, the less significant the effect of their surface. Relationship (10.1) can be modified to become:

$$X_i = 2800 \times R_i \times D \times W \times H \times P \times S \quad (10.2)$$

where P is an adjustment factor for the effect of the quality of the roadway and S is an adjustment factor for the effect of the shoulder quality. An empirical analysis is necessary to extend Table 8-5 in the HCM to narrower widths and to provide realistic estimates for the coefficients P and S . Two more ideal conditions need to be added to those listed in the HCM: (9) the surface of the roadway is asphaltic concrete (or similar quality) and in good condition, and (10) the shoulders are treatment and in good condition.

10.3 DECISION VARIABLES

The two decision variables that are related to every link in the network are the *flow* and the *added capacity*. The former is usually measured with respect to the design hour. The selection of an appropriate hour for design purposes is a compromise between providing an adequate level of service for every (or almost every) hour of the year and economic efficiency. Customary practice is to use an hour between the 10th and the 50th highest hour of the year. For rural highways, the 30th highest hour is the common choice. The flow during the design hour is not a constant; however, it is often agreed that for rural highways the flow during the 30th highest hour is about 12% of the average daily flow. This coefficient, called the *design hour ratio*, is used to convert daily flow, and vice versa.

The capacity is assumed to be continuous to permit small link improvements. One convenient unit of measure of capacity is the PCU per hour. Because the traffic involves other vehicles such as buses and large trucks, those heavy vehicles are converted to equivalent PCU. Capacity refers to the maximum PCU allowed on the road in both directions without leading to heavily congested flow (high delays and low speeds). Likewise, the added capacity is evaluated in PCU as is the flow.

For the study of the Tunisian highway network given the available data, the authors found it suitable to partition Tunisia into 19 regions. Each region is both an origin and a destination, and is represented by a central node labeled from 1 to 19. The network also includes 39 intermediate nodes labeled from 20 to 58. There were 112 two-way links included in the study defined by their end nodes, length, terrain (1 for level, 2 for rolling, and 3 for mountainous), roadway width, shoulder width, roadway surface quality and shoulder quality (1 for asphaltic-good, 2 for asphaltic-fair, 3 for asphaltic-poor, 4 for treatment-good, 5 for treatment-fair, 6 for treatment-poor, 7 for unpaved-good, 8 for unpaved-fair, and 9 for unpaved-poor). The terrain data were given by the Army Map Service, whereas most of the other data were obtained from the publications of the Direction de l'Entretien et de l'Exploitation Routière. Nevertheless, little information was available about the width and quality of the roadway surface and no information was available about the shoulders. The following assumptions were made: (1) if the road is national (referred to as GP), its width is 6.5 m or wider; if it is regional (referred to as RVE), its width is less than 4.5 m; (2) the real width of the roadway is proportional to the width of the road on the detailed maps and is also proportional to its average flow; (3) the sum of the roadway and shoulders widths of all roads is at least 10 m; (4) the pavement is asphaltic when the roadway is at least 8 m in width – otherwise, these roads have treatment surface; (5) the surface quality of the pavement of all links in a given region is equal to the average quality of all pavements of that region, rounded to the nearest integer; (6) if a link connects two regions, its quality is the average of their qualities; (7) the quality of the surface of all unpaved roads is fair; (8) the qualities of the shoulders is unpaved-fair for asphaltic and treatment roads; (9) there are no shoulders for unpaved roads.

10.4 BLP FORMULATION

Traditionally, investment costs are controlled and allocated in an optimal way from the system's perspective, but travel costs depend on traffic flows which are determined by individual user route choice. Because users are assumed to make decisions to maximize their individual utility functions, their actions do not necessarily coincide with what is best for the system as a whole. In fact, they may be in conflict. The system can influence users' choices, however, by improving some links to make them more attractive than others. In deciding on these improvements, the system tries to influence the users' preferences with the aim of minimizing total system costs. The partition of control over the decision variables between two ordered levels requires the formulation of the network design problem (NDP) as a bilevel program in which the system is the upper-level decision maker and the user is the lower-level decision maker.

LeBlanc and Boyce [L2] first gave an explicit formulation of the NDP as a BLP. Their model, however, required the assumption of linear improvement cost functions which may not be realistic. Ben-Ayed et al. [B20] gave a similar, but more general, formulation that allows convex as well as concave improvement cost functions. The model presented here can incorporate any piecewise linear function, including those that are nonconvex and nonconcave.

Recall that a piecewise linear function of Z has the form:

$$f(Z) = b_m Z + d_m \quad \text{for } q_{m-1} \leq Z \leq q_m, \quad m = 1, \dots, J$$

Because f is continuous, we must have for all m :

$$b_m q_m + d_m = b_{m+1} q_m + d_{m+1} \quad \text{or} \quad -q_m(b_{m+1} - b_m) = d_{m+1} - d_m$$

For any m , this implies

$$\begin{aligned} (b_1 Z + d_1) + \sum_{j=1}^{m-1} (b_{j+1} - b_j)(Z - q_j) &= Z \left(b_1 + \sum_{j=1}^{m-1} (b_{j+1} - b_j) \right) \\ &+ d_1 + \sum_{j=1}^{m-1} (d_{j+1} - d_j) = b_m Z + d_m = f(Z) \end{aligned}$$

Note that the above sums are for j from 1 only to $m - 1$. Now, defining $W_j = \max\{Z - q_m, 0\}$, we have

$$f(z) = (b_1 Z + d_1) + \sum_{j=1}^{J-1} (b_{j+1} - b_j) W_j \triangleq R(Z, W)$$

for any $q_0 \leq Z \leq q_J$, since the additional terms in the summation corresponding to $q_j > Z$ will all be zero.

If we want to use $f(Z)$ in the upper-level objective function of a BLPP, we include $R(Z, W)$ in the upper-level objective, require that the lower-level objective make the W_j as small as possible, and include $W_j \geq Z - q_j$ and $W_j \geq 0$ in the problem constraints. Now, using the above concepts for improvement cost functions, the following formulation of the NDP as a BLPP can be obtained.

$$\min_Z \sum_a \left\{ C_a + \tau \left[(b_{1a}Z_a + d_{1a}) + \sum_{m=1}^{J_a-1} (b_{m+1,a} - b_{ma})W_{ma} \right] \right\}$$

where $C_a, \tilde{C}_a, W_{ma}, X_a, X_{ad}$ solve

$$\min_{C, W, X} \sum_a \left\{ (\tilde{C} + \varepsilon C_a) + \sum_{m=1}^{J_a-1} W_{ma} \right\}$$

such that for each node n and each destination d :

$$\sum_{a \in A_n} X_{ad} - \sum_{a \in B_n} X_{ad} = u_{nd}$$

and for each link a :

$$C_a - r_{ma}X_a + g_{ma}Z_a \geq S_{ma}, \quad m = 1, \dots, M_a$$

$$\tilde{C}_a - \tilde{r}_{ma}X_a + \tilde{g}_{ma}Z_a \geq \tilde{s}_{ma}, \quad m = 1, \dots, \tilde{M}_a$$

$$W_{ma} - Z_a \geq -q_{ma}, \quad m = 1, \dots, J_a-1$$

$$\sum_d X_{ad} - X_a = 0$$

$$Z_a, C_a, \tilde{C}_a, W_{ma}, X_a, X_{ad} \geq 0, \quad \forall a, m, d$$

where Z_a is the number of units of capacity added to link a ; X_{ad} is the flow on link a with destination d ; C_a is the approximation of the system travel cost function on link a ; \tilde{C}_a is the approximation of the cumulative user cost function on link a ; d_{1a} is the intercept of the first piece of the improvement cost function; b_{ma} is the slope of the piece delimited by $q_{m-1,a}$ and $q_{ma} = \max\{Z_a - q_{ma}, 0\}$; τ is a factor to convert improvement cost to the same units as travel cost; M_a, \tilde{M}_a and J_a are the numbers of segments in the approximations of system travel cost, cumulative user travel cost, and improvement cost in link a , respectively; ε is a positive scalar sufficiently small so that the optimum to the above problem is the same as the one with ε equal to zero. A_n and B_n are the sets of links pointing out of and into node n , respectively; u_{nd} is the required number of trips between node n and destination d ; r_{ma} and \tilde{r}_{ma} are the slopes of the linear pieces in the travel cost approximations; s_{ma} and \tilde{s}_{ma} are the intercepts of the same pieces; and g_{ma} and \tilde{g}_{ma} are the effects of improvement of reducing s_{ma} and \tilde{s}_{ma} , respectively.

It should be emphasized that the NDP is always concerned with the cost of the system to the community as a whole; the upper-level objective to be minimized is the total system cost consisting of (1) the system travel cost as a function of the flows and the link improvements, and (2) the improvement cost as a function of the links' improvements. The lower-level objective consists of the cumulative user travel cost; i.e., the integral of the average travel cost with respect to flow, and the sum of the W_{ma} required by the nonconvex improvement functions.

10.5 OBJECTIVE FUNCTIONS

10.5.1 Travel Time Functions

Because all travel time functions in the literature are functions of directional flow and therefore implicitly intended for divided highways only, no analytical representation is available for two-lane highways. Consequently, an empirical travel time function was proposed for this study based on data provided by the HCM. Table 8-1 in the HCM gives the minimum average speeds versus the maximum values of the ratio of flow to capacity. For example, under ideal conditions the speed at level of service A is greater than or equal to 93 km/h. Equivalently, the travel time spent per km is less than or equal to 0.64 minutes ($60/93$). On the other hand, the ratio of flow to ideal capacity is less than or equal to 0.15, which means that the service flow is less than or equal to 420 PCU (recall that ideal capacity equals 2800). Therefore, a flow less than or equal to 420 PCU corresponds to a travel time less than or equal to 0.64 minute per km. It seems reasonable to assume equality holds for these relationships. Similarly, for the other levels of service a flow equal to 756 PCU corresponds to a travel time of 0.68 min/km, 1204 corresponds to 0.72, 1792 corresponds to 0.75 and 2800 corresponds to 0.83. A sixth point can also be obtained: free flow corresponds to the design speed, or equivalently, travel time is 0.62 min/km when speed is 97 km/h. For any other value of the flow between 0 and capacity, a linear interpolation is used; an average travel time function is obtained by connecting the six points, as shown in Fig. 10.1.

For every link, an average travel time function can be obtained by applying relationship (10.2) to the corresponding entries of Table 8-1. For rolling and mountainous terrain, the design speeds need to be known to find the travel time corresponding to free flow. Given any geometric, traffic, environmental, and surface conditions, the average travel time obtained using the above procedure is monotonically increasing with respect to the flow.

Level of service F has not been considered thus far. This level is distinguished by its high density, or number of vehicles occupying a given length of roadway. The density corresponding to the capacity is called the critical density; level of service F occurs

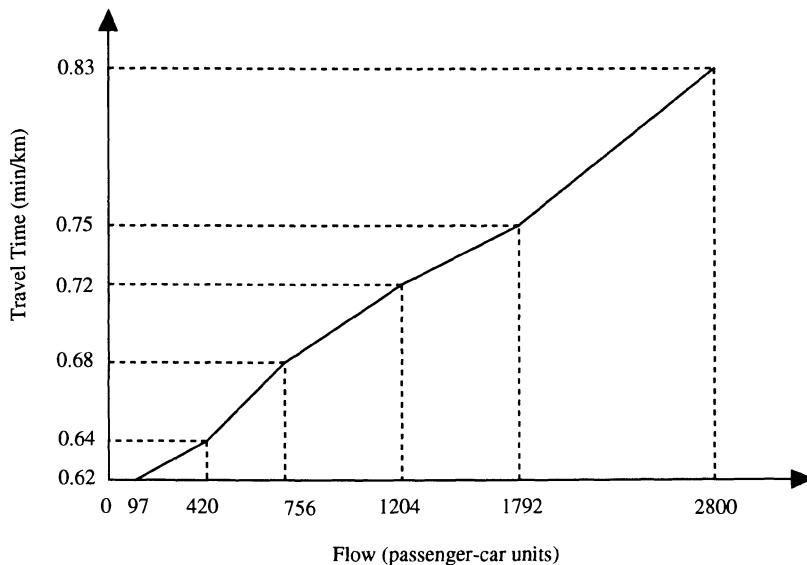


Figure 10.1 Average travel time function

when the density exceeds the critical density. At this level, queues are formed which are characterized by stop-and-go movement. Because at level of service F the vehicles are delayed, the flow is small. A small flow can correspond to either a high travel time if the flow is unstable (congested stop-and-go traffic) or to a low travel time if the flow is stable. Therefore, each flow can correspond to two travel times, which violates the definition of a function. This difficulty can be overcome by assuming that the flow can exceed the capacity, but for those flows greater than the capacity, the travel time increases sharply.

For the case study here, the following assumptions are added (in some cases because of missing data): (1) the data in Tables 8-1, 8-4, 8-5 and 8-6 in the HCM can be applied to Tunisia; (2) the directional split is 60/40 for all links; (3) the percentages of no passing zones are 20% in level terrain, 40% in rolling terrain, and 60% in mountainous terrain; (4) the design speeds are 95 km/h for level terrain, 90 km/h for rolling terrain, and 75 km/h for mountainous terrain (personal experience); (5) additional data can be obtained from any table by using either interpolation or extrapolation; interpolation is linear while extrapolation is based on regression analysis; (6) the vehicle mix for all links consists of 83.5% passenger cars, 13.9% trucks, and 2.6% buses; (7) the factors P and S in Tables 10.3 and 10.4 below can be applied to Tunisia; (8) when the flow exceeds the capacity, the flow of the average travel time is an arbitrarily high number that is the same for all links. Based on these assumptions, Tables 10.1 and 10.2 have been constructed as extensions of the original Tables 9-1 and 8-5 in the HCM.

For each link, an average travel time function is found by applying the procedure explained at the beginning of this section and using the new tables. It is also possible to estimate a system travel time function and a cumulative user travel time function; the cumulative user travel time at a flow X is equal to the area below the average travel time and above the abscissa, and delimited by 0 and X .

In developing countries, travel time cannot be the only component of the travel cost function because of the low value of time of travelers. Other factors such as operating costs and accident cost must be included.

Table 10.1 Maximum ratio R_i of service flow to ideal capacity (both directions)

Flow (LOS)	0 flow	LOS A	LOS B	LOS C	LOS D	LOS E
Level terrain						
Average speed in km/m	95	92	87	82	79	71
Ratio R_i	0	0.12	0.24	0.39	0.62	1.00
Rolling terrain						
Average speed in km/h	90	86	81	77	74	60
Ratio R_i	0	0.07	0.19	0.35	0.52	0.92
Mountainous terrain						
Average speed in km/h	75	70	68	61	56	44
Ratio R_i	0	0.04	0.13	0.23	0.40	0.82

10.5.2 Operating Costs

Operating costs include those associated with the ownership and running of vehicles. The former, such as purchase costs, insurance, and economic aging costs can be neglected because they are independent of the decision variables. Thus only running costs, such as fuel, grease, oil, tires, and depreciation (technological aging) are considered. Economists have emphasized that only the real social costs are involved, and transfers in the form of taxes are not relevant. This means that as far as the social objective function is concerned, the economic values must be considered without taking into account the taxes paid by the individual. Of course, when the objective of user equilibrium is concerned, the entire price paid by the individual must be considered.

Running costs are functions of the link flows. The formulation of the running costs in Tunisia is based on the data provided by SETEC [S5]. Table 10.5 gives the average running costs for 1980 in TD/100 km of an average passenger car in Tunisia when the

Table 10.2 Adjustment factor W for combined effect of roadway and shoulder width

Width of shoulders (meters)	Width of both lanes (meters)							
	10.0		7.5		7.0		6.0	
	LOS A-D	LOS E	LOS A-D	LOS E	LOS A-D	LOS E	LOS A-D	LOS E
≥ 4	1.28	1.28	1.02	1.02	0.97	0.97	0.82	0.85
2.5	1.20	1.25	0.94	0.99	0.89	0.95	0.75	0.83
1.0	1.04	1.19	0.81	0.94	0.76	0.89	0.65	0.78
0	1.00	1.10	0.72	0.90	0.67	0.85	0.57	0.74

Width of shoulders (meters)	Width of both lanes (meters)							
	5.5		5.0		4.5		4.0	
	LOS A-D	LOS E	LOS A-D	LOS E	LOS A-D	LOS E	LOS A-D	LOS E
≥ 7.0	0.70	0.76	0.65	0.71	0.60	0.66	0.54	0.60
4.0	0.70	0.76	0.62	0.70	0.53	0.62	0.42	0.53
2.5	0.66	0.74	0.58	0.68	0.48	0.60	0.38	0.52
1.0	0.56	0.70	0.50	0.63	0.41	0.55	0.32	0.47
0	0.49	0.66	0.43	0.59	0.36	0.52	0.28	0.43

Table 10.3 Adjustment factors P for the surface of the roadway

Quality	Good	Fair	Poor
Asphalt	1.0	0.8	0.5
Treatment	0.9	0.7	0.4
Unpaved	0.5	0.4	0.2

Table 10.4 Adjustment factors S for the quality of the shoulders

Shoulder quality	Treatment good	Treatment fair	Unpaved good	Unpaved fair
≥ 4 m width shoulder	1.00	0.95	0.97	0.90
0 m width shoulder	1.00	1.00	1.00	1.00

speed is 80 km/h, the terrain is level, and the surface is asphaltic-fair or treatment-fair. Tables 10.6 and 10.7 provide the adjustment factors for the effects of the surface conditions and the terrain, respectively. The running costs (tax excluded) for speeds

ranging from 24 km/h to 112 km/h are given in Table 10.8. The tables need some modifications to better fit the year of the study. The following assumptions were added: (1) running costs on treatment are the same as on asphaltic; (2) the economical value of 1 TD in a given year is the same as 1.1 TD in the next year (based on the inflation rate); (3) taxes in 1990 are the same as in 1980; (4) running costs, tax excluded, in 1990 at a given speed are equal to running costs, tax excluded, in 1980 at the same speed, multiplied by a constant; (5) when flow exceeds capacity, running costs increase as sharply as travel time.

Table 10.5 Average running cost in TD per 100 km for 1980

Tax	Fuel	Grease-Oil	Tires	Depreciation	Total
Excluded	0.8623	0.0299	0.1503	0.4139	1.4564
Included	1.7500	0.0447	0.2322	0.4967	2.5236

Table 10.6 Adjustment factors for the effect of the state of the surface on running cost (paved means asphaltic or treatment)

State of the surface	Fuel	Grease-oil	Tires	Depreciation
Paved-good	0.96	0.94	0.92	0.80
Paved-fair	1.00	1.00	1.00	1.00
Paved-poor	1.04	1.06	1.08	1.20
Unpaved-good	1.20	1.11	3.73	1.68
Unpaved-fair	1.26	1.25	3.75	2.68
Unpaved-poor	1.37	1.29	5.06	1.96

Table 10.7 Adjustment factors for the effect of terrain on fuel consumption

Terrain	Level	Rolling	Mountainous
Adjustment	1.000	1.022	1.041

Let RCTE be the running costs in TD/100 km, tax excluded, in 1990 at 80 km/h; RCTE is obtained by multiplying the values of the first four entries of the first row of Table 10.5 by the inflation factor $(1.1)^{10}$, then multiplying the obtained values by the corresponding row of Table 10.6 depending on the surface. The fuel entry also needs to be multiplied by the same inflation factor for the terrain from Table 10.7; the sum of the calculated values of the components gives RCTE. Let RCTI be the running costs in TD/100 km, tax included, in 1990 at 80 km/h. Every component of RCTE is multiplied by the corresponding tax, obtained from the original Table 10.5,

Table 10.8 Running costs in TD per 100 km as function of speed in km/h for 1980

Speed	24	32	40	48	56	64
Cost	1.241	1.204	1.148	1.174	1.200	1.291
Speed	72	80	86	96	104	112
Cost	1.344	1.456	1.551	1.171	1.903	2.241

and the sum gives RCTI. The running costs as a function of the flow are computed from Table 10.8; after the speed is substituted with its equivalent travel time, the inverse function of the average travel time function is applied to convert the travel times into the corresponding flows; columns with negative flows are discarded. Also, the costs corresponding to unstable flows are replaced by higher values; the values in the original Table 10.8 were measured for test speeds and therefore are not applicable to unstable flows. An adjustment factor, AF, is obtained by dividing RCTE by the entry of the second row of Table 10.8 corresponding to the 80 km/h column. The average running costs, tax excluded, as a function of the flow are obtained after the multiplication of the second row of Table 10.8 corresponding to the 80 km/h column. The average running costs, tax excluded, as a function of the flow are obtained after the multiplication of the second row of Table 10.8 by the calculated adjustment factor AF and the connection of the points. To have the same function with tax included, we just multiply the costs, tax included, by (RCTI/RCTE).

For plots of the average running cost function, tax excluded, of the road described above, see [B19]; this function is the basis of the system running cost functions. In contrast, the cumulative user running costs function is based on the average running costs, tax included.

10.5.3 Accident Costs

Road accident costs consist of material damage to vehicles and the environment as well as the social costs associated with bodily damage leading to hospitalization or death. Hence, they are included in the system objective function but not in the user objective, since these costs are not perceived by individuals. The number and severity of accidents depends on many factors, such as congestion (flow), signalization, width, surface (capacity), illumination, time (day, night), condition of the car, the driver, and speed. Some of these factors are related to the decision variables of the model and hence can be controlled by the model. Others are beyond the scope of those variables because of their independence of the flow and capacity.

The number of accidents is usually defined with respect to a unit of length. Depending on the available data, accidents are classified into different categories, and a function is evaluated to relate the costs of each category to the causal factors included in the decision variables of the optimization problem. A natural way to find the relationships is a regression model. Its coefficient of determination R^2 is not expected to be very high since only the factors controlled by the optimization problem are included in the regression model.

The accident data available for this study pertained to 20 regions in Tunisia, comprising for each region the average number of accidents in 1983 (without any details on their severity), the average width of asphaltic or treatment roads, the average quality of the surface of the roadway, and the length of the included links. Four independent variables were chosen: (1) average flow per hour, (2) width of asphaltic or treatment roads in meters, (3) surface condition of the roadway, and (4) capacity in PCU using the relationship (10.2). Surface condition is quantified by introducing a scale measure (10 for good, 5 for fair, and 0 for poor). One observation having an exceptionally high number of accidents was discarded to avoid misleading results. Based on the remaining nineteen observations, the regression analysis showed that accidents depend mainly on flow and width. However, those two variables are significantly correlated ($R = 0.69$). Since the number of accidents depends more on flow ($R = 0.88$) than on width ($R = 0.56$), the following relationship was obtained:

$$\text{Number of accidents per 100 km} = 0.2677 \text{ (flow per hour)} - 5.6466$$

The adjusted coefficient of determination R^2 is 76%.

The insignificant correlation between the number of accidents and the remaining two variables (conditions of roadway surface and capacity) may reflect the fact that good roads attract more vehicles, which results in more accidents due to congestion. Nevertheless, more accurate data that include unpaved roads may give different results. Nineteen observations are too few to generate a reliable regression model. Moreover, those observations are averages for each region; they are not specific to any link of the network.

10.5.4 Improvement and Maintenance Costs

Link improvement has two different aspects, the first is its cost and the second is its effect on reducing travel costs. For this reason, the choice of the improvement cost functions is critical in the model. Unfortunately, no accurate data were available about improvement costs for Tunisia. Subjective estimates were therefore used.

The improvements considered did not include adding new lanes or widening the roadway beyond the existing combined width of the roadway and its shoulders. This restriction was made to avoid consideration of the costs of land acquisition and earth-

work. Land acquisition costs are known to vary considerably from one region to another but no specific cost data are available. The solution of the optimization problem may suggest further study of some links to investigate the possibility of adding more lanes or even upgrading the road to a divided highway. In this study, however, the types of improvements were limited to maintenance costs, roadway widening, roadway surface improvement, and shoulder surface improvement, which excludes some major periodic costs, such as the construction of the road bed and bridges.

The costs related to each type of improvement depend on several factors other than the length of the improved road. The cost of resurfacing per unit of length, for example, depends on the width of the roadway, the type of the new surface, and the condition of the existing one. For each link, it is necessary to have one specific improvement cost depending on the specific conditions of that link and the extent of improvement to shoulders or roadway from the existing state to any other feasible state. Widening of the roadway is considered to be an improvement of a portion of the shoulder because the widening of the roadway is always made at the expense of the shoulders. For the same reason the effect of the terrain is ignored since the costs of resurfacing are not noticeably affected by terrain.

Table 10.9 gives the cost (including taxes) of the six possible surface improvements and roadway widening considered. These costs apply if the surface to be improved is at least 4 m wide. If the width to be added is only 0.5 m, the widening cost per m^2 is assumed to be twice the value given in the table. To add widths between 0.5 and 4 m, a linear interpolation is used. It is usually required that widening of the roadway be accompanied by resurfacing. To avoid double counting fixed costs, it was assumed that the cost of widening and resurfacing at the same time is 90% the sum of their costs if done separately. It was also assumed that if the existing surface is good and if the new surface is in the same category (asphaltic, treatment or unpaved), the cost of resurfacing is 30% of the cost of upgrading from poor to good in that category.

An ideal improvement cost function for the network design problem is one that gives the cost of every PCU of added capacity. However, the capacity can be increased in many ways with different costs and effects on travel costs. Resurfacing the roadway, for example, may increase the capacity by the same amount as improving the shoulders, but the two alternatives do not necessarily have the same cost nor the same effect on reducing the travel cost. Even for the same improvement, the effect varies from one flow to another. The formulation of the investment function, therefore, involves serious difficulties because of the large number of decision variables, which is equal to the number of possible improvement and yield the cost and the effect of each added PCU.

The analysis of the maintenance costs was based on the data provided by SETEC. These annual costs, given in Table 10.10, depend on the type of road. Here, 'paved' means asphaltic or treatment. Based on the following assumptions, Tables 10.9 and

Table 10.9 Costs of surface improvement and roadway widening in TD per m²

Existing surface	Asphaltic good	Treatment good	Unpaved good
Asphaltic-fair	6.0	—	—
Asphaltic-poor	9.0	—	—
Treatment-good	9.5	—	—
Treatment-poor	11.0	5.5	—
Unpaved-good	12.0	6.5	—
Unpaved-poor	13.0	7.5	2.5

10.10 can be used after updating the values of the costs to 1990 and deducting the taxes: (1) Table 10.9 applies to Tunisia for the year 1987; (2) taxes on investment and maintenance costs are 20% of the total cost (SETEC); (3) the data about earth surface apply to all unpaved roads.

Table 10.10 Annual fixed maintenance costs in TD per km for 1981

Type of road	Earth	Paved 4-5 m	Paved 6-7 m	Paved 9-10 m
Annual costs	120	220	260	330

10.5.5 Additivity of Cost Functions

The upper- and lower-level objective functions for each link are obtained by adding the components of the system cost functions and the components of the cumulative user costs functions, respectively. Nevertheless, such additivity is possible only when the components are expressed in the same units and for the same period of time. The authors choose the unit to be 1000 TD and the period to be one year. It was assumed that the goal was to allocate the optimal inter-regional highway investment for a five-year period from 1988 to 1992; the year 1990 was assumed to be an average year. The choice of the specific unit and period is not relevant to the model because changing them is equivalent to multiplying the objective functions by a positive constant. This would not affect the solution of the optimization problem.

In contrast, the choice of the conversion factor is significant. The value of time is provided by SETEC. This value was estimated to be 0.25 TD/h in 1986 with an increase of 2.8% per year, which gives a value of 0.28 TD/h in 1990. However, since the average number of passengers per vehicle in Tunisia is 4.38, this value becomes

1.22 TD. For the costs of accidents, no Tunisian data were found; therefore, the cost per accident in 1985 provided by the National Safety Council for the U.S. was applied to Tunisia. To take into account the differences in the standards of living, the costs of wage loss, in medical expense and insurance administration (\$29.3 billion) is multiplied by the ratio of the Tunisian gross national product per capita to the U.S. gross national product per capita (1,420/12,820 according to the World Bank). To convert the value to 1990 it was multiplied by $(1.028)^5$, which corresponds to the increase of salaries in Tunisia during that period. Next, the motor-vehicle property damage (19.3 billions of dollars) is summed and, multiplied by the inflation factor $(1.1)^5$. The computed total cost is then divided by the total number of accidents in the U.S. in 1985 (19,300,000). This leads to a cost of \$1,804 or 1,455 TD per accident. The number of accidents in 1990 was assumed to be the same as in 1983.

The travel time function has units of time per hour because flow is defined with respect to the hour. To convert those functions to an annual basis, it is necessary to divide by 0.12 to obtain the travel time per day, as explained in Section 10.5.3, and then multiply by 365. It is assumed that the flow is uniform during the 365 days of the year. On the other hand, improvement costs cover a period that is usually much longer than one year. Let C_1 be the actual expenditure in the first year, n the lifetime of the investment, and C_n the equivalent annual expenditure over the n -year period. Assuming the end of year convention C_n is given in [M16]:

$$C_n = [C_1 i(1 + i)^n]/[(1 + i)^n - 1]$$

where i is the interest rate. The interest rate used for Tunisia is 10% (SETEC). The lifetime of each investment (Table 10.11) is assumed to be 1.5 times the period given by SETEC for what they call periodical maintenance costs.

Table 10.11 Lifetime of investment

Type of road	Earth	Treatment	Asphaltic
Lifetime (yrs)	7.5	10.5	18

Using the above information, the total system travel cost and the total cumulative user costs for the illustrative link used in the study can be obtained (see [B19] for more detail). Those functions are basically composed of two pieces corresponding to stable and unstable flows. This result was confirmed for all other links in the network. There are two special cases: the first segment must start with the 0 improvement point and the third segment must end with the last point, the most expensive improvement. It is required that the approximated cost given by the first segment at zero improvement be nonnegative and no greater than twice the actual (maintenance) cost; when this condition is not satisfied for a given approximation the exact value of that point is imposed. For each of the 112 links in the study, the best approximation consisted of a three-piece nonconvex function. This greatly increased the complexity of the BLP

formulation. The other alternative, which is the convex and/or linear formulation, is much more efficient from a computational point of view but is often much less accurate; the convex formulation increases the error for 10.71% of the links by more than 12 times as compared to the nonconvex approximation. Fortunately, however, there are other links for which a convex improvement-cost function is nearly as accurate as the nonconvex one. In dealing with the tradeoff between accuracy and computational efficiency, the authors decided to use concave approximation only when it decreased the error by more than 50% as compared to convex approximation of the improvement cost. This compromise permitted the number of nonconvex improvement cost functions to be reduced to 36 (all 36 were three-piece nonconvex and nonconcave). The remaining 76 consisted of 8 linear and 68 two-piece convex functions. For the illustrative link, the best approximation is a three-piece nonconvex-nonconcave function (Fig. 10.2). This approximation was selected after considering 16,816 combinations. With regard to the piecewise linear total travel cost functions, the first piece is used to represent the flow in the stable range (level of service A-E). When flow is in the unstable range, the second piece was defined taking into account capacity additions.

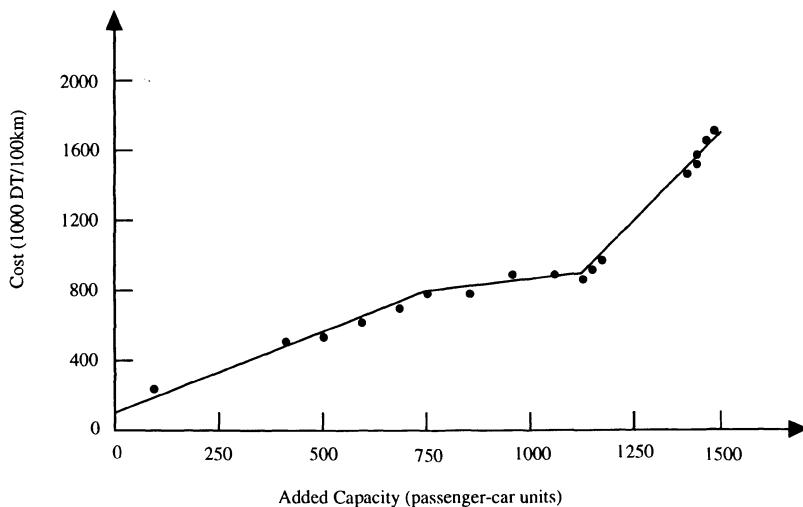


Figure 10.2 Example of improvement cost function

10.6 CONSERVATION OF FLOW CONSTRAINTS

One way to state the conservation of flow conditions is

$$\sum_{r \in R_{od}} X_{r,od} = u_{od} \quad \text{for each origin-destination pair } od \quad (10.3)$$

where R_{od} is the set of all routes r going from origin node o to destination node d , $X_{r,od}$ is the flow of passengers (per unit time) from o to d using route r , and u_{od} is an element of the trip-matrix giving the required number of vehicle trips (per unit time) from o to d .

An equivalent formulation of those conditions is

$$\sum_{a \in A_n} X_{ad} - \sum_{a \in B_n} X_{ad} = u_{nd} \quad (10.4)$$

for each destination d and each node n other than d , where X_{ad} is the flow on link a with destination d , and u_{nd} is the required number of trips between node n and destination d .

A choice has to be made between relationships (10.3) and (10.4) in order to minimize the number of constraints and variables involved in the formulation of the conservation of flow conditions. Since the network consists of 19 origin-destination nodes, 39 intermediate nodes, and 224 directed links, relationship (10.4) requires 4256 variables (19×224) and 1083 equality constraints ($19 \times (39 + 19 - 1)$). Relationship (10.3), however, requires at most 342 equality constraints (19×18), but many more variables. Besides its fewer number of constraints, relationship (10.3) has a very important advantage: the number of variables and constraints can be considerably decreased, since there are origin-destination pairs with zero entries in the trip matrix. If an origin-destination pair has a zero entry in the trip matrix, it does not belong to the set of od pairs. Consequently, there is no reason to have any variable or constraint associated with it.

In attempting to test the efficiency of such a property in the use of relationship (10.3), the number of variables had to be limited. The following three assumptions were made: (1) every traveler going from origin o to destination d chooses his route from among the first 100 shortest routes from o to d , (2) no traveler chooses a route where there is a node visited more than once, and (3) no traveler chooses a route that is more than twice as long as the shortest route. Based on these assumptions, for every origin-destination pair, the 100 shortest paths were computed and then all routes with nodes visited more than once and all routes with length more than twice that of the first shortest path were excluded. This resulted in 11,900 variables in the BLP model.

The links included in the study are used by inter-regional flow as well as by intra-regional flow. Intra-regional flow includes all trips between intermediate nodes or between places inside the regions. Those flows have to be considered by the model because of their effect in congesting the roads and thereby increasing the travel costs to the inter-regional flow. A convenient way to include intra-regional flows is to assume that they reduce the capacity of the roads used by the inter-regional flow. For each region, it was assumed that the intra-regional flow is 40% of the existing capacity.

SETEC provided a 29×29 annual trip matrix for 1977 and predicted trips for 1986. The same rates of increase were retained to find the predicted numbers of required trips between each origin-destination pair for 1990. Some subregions had to be grouped to obtain the 19×19 trip matrix; flows between subregions belonging to the same region were discarded. The entries of the matrix were converted into average trips per hour (in PCU) to be consistent with the definition of the flow. The trip matrix obtained is symmetric. All origin nodes are also destination nodes (they are referred to as origin-destination nodes). Among the 342 entries in the matrix, 24 have a value of zero, which means that the 342 constraints required by relationship (10.3) can be decreased to 318. Among the 11,900 variables, 1006 (corresponding to those 24 *od* pairs with zero entry) are redundant, thereby decreasing the number of variables to 10,894. Those numbers can be decreased even more when the following two facts are recognized.

First, in many cases all trips from origin o to destination d must go through a third origin-destination node x . In such a case, the trip matrix can be modified to include zero trips from o to d . Let t , m and n denote the required number of trips from o to d , o to x , and x to d , respectively. If all routes from o to d include the node x , then it is equivalent to say that $m + t$ are required to go from o to x , $n + t$ are required to go from x to d , and zero trips are required to go from o to d . By applying this fact to the trip matrix, the number of entries with zero value is increased to 102, which decreases the number of *od* constraints to 240 and the number of routes to 7655.

The second fact is based on the symmetry of the trip matrix and is even more important because both objective functions depend on the sum of the flows in both directions for all links. Symmetry means that the trip matrix has an identical (50/50) directional split. However, the travel cost and conservation of flow are the same if we consider $2t$ trips going from o to d and zero trips from d to o , instead of t trips from o to d and t trips from d to o . This involves changing the entries of the upper right triangle of the matrix to zero and doubling the values of the entries of the lower left triangle. This alone, increases the number of zero entries to 183.

When both facts are used, the number of zero entries becomes 222 (Table 10.12), yielding 3824 variables and 120 constraints. These numbers clearly dominate those obtained by relation (10.4), which requires 4256 variables and 1083 constraints. Therefore, (10.3) is used for the formulation of the conservation of flow conditions. Furthermore, many *od* pairs are far away from each other, resulting in a large number of routes and a small number of required trips between the *od* pairs. In other words, those pairs are introducing a huge number of variables that can be discarded without having a considerable effect on the problem. In the study, the number of routes corresponding to each *od* pair was limited to twice the number of required trips between the nodes of that pair. This imposition led to the elimination of 1729 additional routes, leaving a final number of flow variables equal to 2095.

Table 10.12 Equivalent trip matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	total
2	1832																		1832
3	982	102																	1084
4	292	14	50																356
5	838	16	80	98															1032
6	2660	120	56	8	26														2870
7	232	56	30	6	10	40													374
8	1056	80	38	36	12	170	80												1472
9				6	4			352											362
10							3260	324											3584
11	172	2	6	18	8	36	34	454	48										778
12	124	4	44	70	4	8	24	52	6	46									382
13	48	6	8	74	4	8	2		8	44	16								218
14	502	46	6	12	10	66	10	134	340	62	74	6	28						1296
15	18		4	2	2		2		10	4	64	8	16	162					292
16	60	2	4	4	6		6		6	4	28	8	40	18	50				236
17	126	6	8	2	4	6	2	28	12	8	6	2	4	84	20	56		374	
18																108		108	
19															10	112	24	2	148
total	8942	454	334	336	90	334	160	4280	754	78	262	40	88	274	70	168	132	2	16798

10.7 SOLUTION OF EMPIRICAL PROBLEM

There are two reasons for having a BLP formulation: (1) the user optimized flow requirement (user-equilibrium), and (2) the nonconvex improvement functions. The proposed algorithm was designed to take advantage of the fact that the lower-level problem can be decomposed into two distinct problems which can be solved separately. At each iteration, an attempt is made to find a better compromise with the user, while including the smallest possible number of nonconvex improvement functions to get the exact solution with minimum computational effort. The algorithm is an iterative procedure that tries to reduce the gap between an ideal solution and the incumbent. The procedure terminates when the gap falls below a desired accuracy or when the number of iterations exceeds a fixed limit.

Approximately 15.25 minutes of CPU time and 1.4 million words (64 bits per word) of disk space were required to solve the problem. The computations were conducted

on a CRAY X-MP/24 at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign. Despite the size and complexity of the problem, an accurate solution was obtained with a gap of 2.56% between the upper and lower bounds. The complete presentation of the solution is provided in Appendix E of [B17]. Almost all link improvements are on the roads connecting Tunis, the capital, to its major neighboring cities. A similar recommendation was made by SETEC stating that the roads corresponding to the exits of Tunis will be highly congested by the year 2000 if no improvements are made. In contrast, the entries of the trip matrix corresponding to regions in the south are very low, which results in no improvement even for unpaved roads.

Table 10.13 gives for each link to be improved the inter-regional flow, the existing capacity available to this flow (60% of the total existing capacity), the added capacity, its cost, and maximum added capacity allowed by the formulation. The improvement required for link 14 is very small; a slight surface improvement of this link gives it exactly the same capacity as the adjacent link 79, which does not need improvement although it is closer to the capital. All other improvements are more significant, and three of them, namely those on links 3, 20 and 26, require the maximum added capacity allowed by the formulation. For each of the links, the flow is higher than the new capacity; more detailed study is needed to include the possibility of upgrading them to divided four-lane highways. Link 90 is also congested; however, the solution for the system does not add more capacity even though it has the possibility to do so. The reason is that the congestion on those roads is less expensive than the addition of more capacity.

An analysis of the results obtained (see [B17]) draws attention to several possible improvements of the empirical formulation. First, the travel times at the different levels of service A to E are so close, according to the HCM, that the resulting system and cumulative user costs at stable flow turned out to be one-piece linear functions that do not depend on the added capacity. The only instrument in the model for the system to influence a user's choice of routes is by adding capacity if flow is unstable. Therefore, the ability of the system to affect the user-optimized equilibrium is almost negligible. More detailed empirical data about travel costs are needed to represent the effects of congestion on user costs.

Second, it would have been beneficial to include land purchase costs in the study. Their unavailability meant that new links could not be considered. This restricted investments to improvement of existing links by not allowing the added width to go beyond the existing shoulder. This limitation resulted in low improvement costs which are about 50 times smaller than travel costs. The introduction of a BLP formulation to account for the investment costs did not lead to much saving as compared to simpler LP formulations because the costs are already low according to the problem data.

Table 10.13 Links to be improved

Link	Flow	Existing capacity	Added capacity	Improvement cost	Limit on added capacity
1	2144	1370	774	592	887
3	2537	1637	899	394	899
4	2112	1115	997	297	1083
6	2015	1055	960	615	1070
10	1980	814	1166	484	1460
12	1052	448	604	94	2046
14	306	217	35	22	1354
19	1056	447	609	89	2046
20	2275	1173	1101	463	1101
26	2537	1637	899	245	899
28	2010	1370	640	362	887
30	1517	1370	147	60	887
73	2112	862	1250	394	1496
77	1772	1055	717	156	1070
90	2058	1370	687	179	887

10.8 CONCLUSIONS

The issues investigated in this chapter represent an attempt to go beyond theoretical formulations and small illustrative examples and address much more interesting real-world problems. Much of the effort was devoted to the construction of a bilevel linear programming formulation, a step in the formal optimization of the inter-regional highway network of a developing country. The reliability of any formulation is conditioned on the accuracy of its components and the tractability of the resulting problem to be solved. The proposed bilevel programming model is much more realistic than single-level models considered in the past in two respects: (i) both societal and individual decision levels are explicitly represented; (ii) investment functions with economies of scale are incorporated. Furthermore, the complexity of the formulation is not a barrier to its solution although greater computational effort is required. In today's computing environment, this may be viewed as an opportunity rather than a drawback.

The elaboration of the formulation to depict the network design problem more realistically, as well as the success in solving the resulting optimization problem (to a specified degree of accuracy) do not necessarily guarantee its credibility. As in any mathematical programming application, the realism and availability of the required data ultimately determine the acceptability of the results. In the application described here, an extensive data base would be required for a full-scale implementation. Cost functions which do not exist must be estimated; moreover, they must be estimated for

unrealistic as well as realistic decision alternatives in order for these to be systematically compared. In particular, careful and detailed theoretical and empirical studies are needed to give more representative data, including better travel cost functions and better improvement cost functions for two-lane and unpaved highways.

Finally, the fixed demand, user-equilibrium route choice formulation was somewhat artificial. A follow-on study might include stochastic route choice and estimation of travel demand. It should also be noted that transportation improvements often have a considerable impact on regional location decisions, potentially implying income redistribution effects of considerable importance. It is hardly surprising that motives of income redistribution have a way of appearing implicitly or explicitly in the rationale for many public transportation investments. In fact, transportation investments can result in improved productivity and expanded employment over the long run; *i.e.*, transportation infrastructure can affect ways in which regions and communities develop. To account for this phenomenon, the current study could be expanded to optimize societal objectives, such as a desire to balance economic growth among the regions. A simple way to include these concepts in the bilevel model is to introduce higher trip matrix entries to the less developed regions. This would result in more of the budget being allocated to roads in those regions thus helping influence their development.

PRODUCTION PLANNING WITH INEXACT CUSTOMER DEMAND

11.1 INTRODUCTION

The vast majority of research in production planning takes demand as given and attempts to find an optimal production schedule by balancing inventory and manufacturing costs with system constraints (e.g. see [M11, V4]). Efforts to account for uncertainty in demand have primarily focused on the relationship between consumer behavior and advertising [A10, N4]. In most organizations, marketing is responsible for collecting and analyzing the relevant data. Advertising strategies are then devised and implemented through a series of allocation decisions aimed at maximizing total sales. The planning department is usually at the receiving end of these decisions and must respond with a master production schedule designed to meet customer demand at minimum cost. Experience shows that this can be a difficult and frustrating task due to the physical and operational constraints of the system. Even with such techniques as capacity requirements planning, lengthy trial and error procedures may still be needed to find good feasible solutions.

The crux of the problem is one way communications. In many large organizations, little coordination exists between departments so critical information is often lost or ignored by those persons who could benefit most. An effective way to correct this situation is with an integrated approach that addresses the concerns of marketing, manufacturing and planning in a single framework. In this chapter, we develop such an approach based on the work of Bard and Moore [B12] who revised the standard dynamic inventory model to include a functional relationship between advertising and demand. In particular, they introduced a customer who, at each point in time, reacts to the cumulative effects of advertising by specifying how much of each product he will consume. As a consequence, management is able to influence but not control demand through a deliberate adjustment of the advertising budget.

These ideas are elaborated in the next section where the notation is defined and a mathematical statement of the problem is given. A series of modifications are

proposed to account for customer behavior. This leads to a bilevel programming formulation. An application reflecting the costs and market interaction of a manufacturer of electric power supplies is presented in Section 11.3 where the computations are highlighted. This is followed in Section 11.4 by a discussion of the practicality of the approach and some recommendations for implementation.

11.2 MATHEMATICAL DEVELOPMENTS

In reality, the underlying parameters of a model are rarely known with certainty but have to be estimated from what little data are available. If some of these parameters are random variables, chance-constrained or stochastic programming techniques could prove effective in the analysis. More likely, though, it may only be possible to provide upper and lower bounds on parameter values. In this case, an inexact representation of the problem may be more appropriate (e.g., see [S14]). Note that the ideas of inexact mathematical programming have been subsumed in the field of robust optimization.

Bard and Chatterjee [B9] formulate the production planning problem along these lines and show, in general, how to obtain objective function bounds under worst-case conditions. Building on this work, we now consider a manufacturer of who wishes to determine a production schedule that will maximize his profits over a fixed period of time subject to resources constraints and an inexact demand forecast. Subcontracting and the use of advertising to influence customer choice will also be included in the model.

11.2.1 Formulation as a Bilevel Program

The basic components of this scenario define a dynamic inventory model. In reformulating the model to take into account advertising, we get a sequential game. Consider the situation where a manufacturer of n products wishes to determine a production mix, $\mathbf{x} \in R^{n \times T}$, that maximizes his profits over a finite horizon T in the face of q resource constraints. Assume that the demand at time t , $\mathbf{d}_t \in R^n$, is inexact; i.e., known only to lie in the set D_t whose boundaries are subject to the influences of advertising. For example, D_t might specify upper and lower bounds on the demand for each product in period t . We now suppose the existence of a sole customer whose demand (requirements) for the n products is a function of the manufacturer's expenditure on advertising, $\mathbf{v} \in R^{n \times T}$, where 'advertising' is intended to include all activities such as marketing, the use of extended warranties, and temporary on-site trouble shooting that might sway the customer's purchase decisions. (The particular application we have in mind centers on the manufacture of specialized assemblies for an auto maker by a supplier new to the market. It is not unusual for a startup

firm producing specialized parts to have a single customer in its first few years of operation.)

In the model, the manufacturer is assigned the role of leader and begins by selecting a production mix and advertising strategy for each point in time. It is assumed that the customer is rational and will react to these choices by meeting his demand at minimum cost. The customer can therefore be viewed as a follower in the game whose purchase decisions come after the manufacturer announces his full set of plans for the current period. The nature of the market rules out cooperation. An additional assumption is that all demand must be met. As a consequence, the customer effectively controls inventory and shortages. This can be seen from the material balance equation (11.1f) below which indicates that once the production and advertising decisions are taken, demand must be satisfied from a combination of inventory and subcontracting. This raises a number of modeling issues, depending upon whether subcontracting is permitted in a period when adequate production capacity and inventory exist. Only allowing subcontracting at times when shortages occur has some practical justification because some uncertainty usually surrounds future demand. Nevertheless, building inventory is compatible with the situation where the manufacturer can anticipate the customer's decision.

In the developments, we make use of the following notation.

Parameters

a_{ijt}	amount of resource i required to make a unit of product j in period t
b_{it}	amount of resource i available in period t
p_{jt}	selling price of product j in period t
c_{jt}	unit cost of manufacturing product j in period t
h_{jt}	unit holding cost of product j in period t
s_{jt}	unit cost of subcontracting product j in period t
r_t	cost of renting warehouse space in period t
f_{jt}	setup cost for product j in period t
B_j	relative measure of space occupied by product j normalized to product 1
\bar{I}_t	maximum amount of free inventory space available in period t
M	large constant

Sets

V_j	set constraining advertising expenditures for product j
$D_t(\mathbf{v})$	set in which the n -dimensional vector \mathbf{d}_t lies in period t

Decision variables

x_{jt}	amount of product j manufactured in period t (leader variable)
v_{jt}	amount spent on advertising product j in period t (leader variable)
S_{jt}	amount of shortage of product j in period t (follower variable)
I_{jt}	amount of inventory of product j at the end of period t ; it is assumed that the initial inventory I_{j0} is 0 (follower variable)
d_{jt}	leader's (inexact) demand forecast (alternatively, the customer's demand) for product j in period t (follower variable)
y_t	binary variable equal to 1 if inventory space is rented in period t ; 0 otherwise
z_{jt}	binary variable equal to 1 if a setup cost is incurred for product j in period t ; 0 otherwise

Given the above assumptions, and including setup costs and upper limits on inventory, we get the following mixed-integer linear bilevel programming problem (BLPP).

$$\max_{\mathbf{x}, \mathbf{v}, \mathbf{z}} F = \sum_{j=1}^n \sum_{t=1}^T p_{jt} d_{jt} - \sum_{j=1}^n \sum_{t=1}^T [c_{jt} x_{jt} + h_{jt} I_{jt} + s_{jt} S_{jt} + v_{jt}] - \sum_{t=1}^T r_t y_t - \sum_{j=1}^n \sum_{t=1}^T f_{jt} z_{jt} \quad (11.1a)$$

$$\text{subject to } \sum_{j=1}^n a_{ijt} x_{jt} \leq b_{it}, \quad \forall i, t \quad (11.1b)$$

$$M z_{jt} - x_{jt} \geq 0, \quad \forall j, t \quad (11.1c)$$

$$x_{jt} \geq 0, \quad v_j = (v_{j1}, \dots, v_{jT}) \in V_j, \quad z_{jt} \in \{0, 1\}, \quad \forall j, t \quad (11.1d)$$

$$\min_{\mathbf{d}, \mathbf{I}, \mathbf{S}, \mathbf{y}} \sum_{j=1}^n \sum_{t=1}^T p_{jt} d_{jt} \quad (11.1e)$$

$$\text{subject to } I_{jt} - I_{j,t-1} + d_{jt} - S_{jt} = x_{jt}, \quad \forall j, t \quad (11.1f)$$

$$M y_t + \bar{I}_t - \sum_{j=1}^n B_t I_{jt} \geq 0, \quad \forall t \quad (11.1g)$$

$$S_{jt} \geq 0, \quad I_{jt} \geq 0, \quad I_{j0} = 0, \quad \forall j, t \quad (11.1h)$$

$$\mathbf{d}_t = (d_{1t}, \dots, d_{nt}) \in D_t(\mathbf{v}), \quad \forall t \quad (11.1i)$$

The manufacturer's objective (11.1a) is to maximize profits, i.e., the difference between total revenues and the following costs: production, inventory, shortage, advertising, warehouse rental, and setup. Conversely, the follower is simply trying to minimize his costs given in (11.1e). Prices, p_{jt} , are assumed to be fixed and outside

the control of the manufacturer. Constraint (11.1b) places certain restrictions on the use of resources, while limits on advertising are implicitly established in (11.1d) and affect demand through (11.1i). If product j is produced in period ($x_{jt} > 0$), the manufacturer incurs a setup cost denoted by f_{jt} . This is enforced by inequality (11.1c). Equation (11.1f) accounts for material balance. Initial inventory levels are taken as zero (11.1h) but may assume arbitrary values. Of course, if advertising is omitted from the model and the demand d_{jt} is known exactly, then D_t is a singleton, and problem (11.1) reduces to a standard mixed-integer linear program.

When production capacity is insufficient to meet demand in a given period, two options are assumed to be available to the manufacturer. He can fill orders from stock or he can buy additional product from a subcontractor. (Note that the exclusion of the backlogging option does not affect the generality of the results.) Depending upon the marginal costs, it may be advantageous to over order in the current period, holding the surplus for later distribution. As the need arises, the manufacturer is permitted to rent additional space. To keep the formulation simple, it is assumed that at most one supplementary storage facility will be required, and that it must be rented in its entirety. Modeling this situation is accomplished with T binary variables, y_t , one for each period. Similarly, setup costs are handled with the $n \times T$ binary variables, z_{jt} . Constraint (11.1g) requires the manufacturer to rent additional space in period t whenever total inventory exceeds \bar{I}_t . He pays a fixed fee, r_t , for the whole warehouse. If space can be rented in incremental units, it would be necessary to redefine y_t as an integer variable and modify (11.1g) accordingly.

When setup costs and the option to rent additional warehouse space are excluded, (11.1a)–(11.1i) simplifies to a continuous BLPP:

$$\begin{aligned} & \max_{\mathbf{x}, \mathbf{v}} F \\ & \text{subject to eqs. (11.1b,c)} \\ & \min_{\mathbf{d}, \mathbf{I}, \mathbf{S}} \sum_{j=1}^n \sum_{t=1}^T p_{jt} d_{jt} \quad (11.2) \\ & \text{subject to eqs. (11.1f,h,i)} \end{aligned}$$

11.2.2 Interpretation and Technical Considerations

In problem (11.1a)–(11.1i), once the leader chooses \mathbf{x} , \mathbf{v} and \mathbf{z} , the follower must select \mathbf{d} , \mathbf{I} , \mathbf{S} and \mathbf{y} to minimize the cost function (11.1e). This point is emphasized by placing the leader's variables, x_{jt} , on the right-hand side of eq. (11.1f). The follower's subproblem is equivalent to a zero-one mixed-integer linear program when V_j and D_t are polyhedral.

If subcontracting is only allowed in periods when a shortage exists, the addition of the following complementarity constraint to problem (11.1) is required: $I_{jt}S_{jt} = 0$, for all j and t . This nonlinearity is best dealt with by placing it in the customer's objective function as a penalty term. The modified objective function becomes

$$\min_{\mathbf{d}, \mathbf{I}, \mathbf{S}, \mathbf{y}} \sum_{j=1}^n \sum_{t=1}^T p_{jt}d_{jt} + M \sum_{j=1}^n \sum_{t=1}^T I_{jt}S_{jt} \quad (11.3)$$

where M is a sufficiently large constant to assure that the second term in (11.3) is zero in the optimal solution.

If subcontracting is permitted in any period without regard to need, a situation may arise where the customer is indifferent to any one of a number of optimal responses. This occurs because the BLP formulation gives the customer rather than the manufacturer control over inventory and subcontracting. When a shortage is in view, the customer can either subcontract in the current period or fill orders from stock. Because neither choice affects (11.3), he may be indifferent between the two. Only one of these decisions, though, may be optimal for the leader. And without cooperation, it may be impossible for the leader to realize his maximum gain. Such a situation may be disturbing from a theoretical point of view, but is easy to deal with at the algorithmic level (the equivalent of problem (5.3) can be solved to check for multiple optimal solutions).

In problem (11.1), setup costs are controlled by the manufacturer but the option to rent additional space, y_t , is a customer decision. As with inventory and shortages, if the reverse were true, the manufacturer would be able to artificially constrain demand by restricting his rental decision when the economics proved advantageous. Of course, if hard upper bounds exist on inventory, and subcontracting is not permitted, it is possible to model these conditions by placing excessively high costs on rental space, and by adding a backorder term to the manufacturer's objective function. Finally, note that in the above formulation the manufacturer is allowed to meet shortages in a given period by ordering product from a subcontractor at any prior time. To preclude this option, a penalty term must be added to the customer's objective function (11.1e), as in (11.3).

11.3 APPLICATION ASSOCIATED WITH ELECTRIC MOTOR PRODUCTION

The computations are illustrated with data gathered from a manufacturer of transformers, electric generators and motors. Major customers include the U.S. auto makers, the home appliance industry, and a handful of secondary personal computer manufacturers. Recently, the company has expanded its product line and now offers

equipment rated anywhere from 50 watts up to 25 kilowatts. The primary means of advertising is through trade journals and direct mail.

In the last five years, orders have been steadily growing but at an uneven and erratic pace. Swings of over 50% have been common from one year to the next. As a consequence, management has decided against expansion. It feels that it is better to keep production at a relatively fixed level and to meet excess demand through subcontracting. Accordingly, agreements have been made with three local job shops to pick up the workload when orders exceed capacity.

We begin the analysis with an investigation of problem (11.2), and then discuss the solution of the more complicated version given in (11.1). In each instance, the number of products is 2, the number of periods is 4, and the number of resource constraints is 6, implying that $n = 2$, $q = 6$, and $T = 4$. The cost data used in the study are displayed in Table 11.1. A majority of these data were obtained from the accounting department after lengthy discussions; however, some of it had to be estimated from incomplete records, and therefore should only be construed as illustrative in nature. In fact, the company has over 50 products, and has yet to fully assess the individual cost components of each. With these words of caution, notice that values for p_{jt} and c_{jt} suggest a seasonal pattern, and that in a given period it is always better to produce an item than to subcontract it out. There are times, though, when it may be more economical to acquire product through subcontracting and hold it in stock, than to produce it in subsequent periods. For example, consider the subcontracting and holding costs for item 2 in period 2 ($s_{22} + h_{22} = \$170 + \25) versus its production cost ($c_{23} = \$200$) in period 3.

Table 11.1 Problem data

Coefficient	Period			
	1	2	3	4
P_{1t}	75	115	160	90
P_{2t}	115	200	300	160
c_{1t}	50	75	100	60
c_{2t}	75	95	200	100
h_{1t}	25	25	25	25
h_{2t}	25	25	25	25
s_{1t}	80	110	140	95
s_{2t}	110	170	140	95
f_{1t}	4300	4300	4300	4300
f_{2t}	3200	3200	3200	3200
r_t	3000	3000	3000	3000

Also, $\bar{I}_t = 225$ in each period, $B_1 = 1$ and $B_2 = 1.5$. For the six resource constraints, the \mathbf{A} matrix and right-hand-side vector are given below. The first four columns of \mathbf{A} index time for product 1 and second four index time for product 2.

$$\mathbf{A} = \begin{bmatrix} 3 & 3 & 3 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 2 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 \end{bmatrix}$$

$$\mathbf{b}^T = [2750, 2000, 2700, 2700, 2700, 2700]$$

With regard to advertising, the marketing department has established a policy that limits expenditures for the two products under consideration to no more than \$10,000 over the planning horizon. As a general rule, however, no less than \$1,500 should be spent on a single product in any two consecutive periods. These requirements generate seven linear constraints, which, in effect, define sets V_j . Letting $V = \cup_j V_j = \{\mathbf{v} : \mathbf{Ev} \leq \mathbf{e}\}$, the corresponding matrix, \mathbf{E} , and right-hand-side vector, \mathbf{e} , associated with the variables, v_{jt} , are shown below. The first constraint is \leq and the other six are \geq .

$$\mathbf{E} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\mathbf{e}^T = [10000, 1500, 1500, 1500, 1500, 1500, 1500]$$

Using regression analysis, it was possible to able to obtain information about the relationship between advertising and sales. When the above strategy is followed, it is expected that at least 100 units of each product will be sold. However, for each dollar spent on advertising in one period, a certain amount of demand is subsequently generated; i.e., product demand is related to the amount of advertising done before and during a period. This relationship is clarified in Tables 11.2 and 11.3.

Based on historical trends, the customer's requirements for product 1 are as follows. For each unit bought in period 1, two-thirds of a unit will be purchased in period 2; for each unit bought in period 2, one-half of a unit will be purchased in period 3; and

Table 11.2 Demand per dollar of advertising for product 1

Advertising period	Demand per period			
	1	2	3	4
1	0.050	0.030	0.015	0.005
2	–	0.325	0.020	0.020
3	–	–	0.0325	0.0175
4	–	–	–	0.037

Table 11.3 Demand per dollar of advertising for product 2

Advertising period	Demand per period			
	1	2	3	4
1	0.080	0.020	0.015	0.005
2	–	0.040	0.020	0.010
3	–	–	0.031	0.019
4	–	–	–	0.039

for each unit purchased in period 3, three-fourths of a unit will be bought in period 4. A similar set of relationships exists for product 2. In all, this information taken with the data presented in Tables 11.2 and 11.3, defines the sets D_t :

$$d_{11} + d_{12} + d_{13} + d_{14} \geq 100$$

$$d_{21} + d_{22} + d_{23} + d_{24} \geq 100$$

$$2d_{11} - 3d_{12} \leq 0$$

$$d_{12} - 2d_{13} \leq 0$$

$$3d_{13} - 4d_{14} \leq 0$$

$$d_{21} - 2d_{22} \leq 0$$

$$d_{22} - 4d_{23} \leq 0$$

$$3d_{23} - 5d_{24} \leq 0$$

$$100d_{11} - 5v_{11} \geq 0$$

$$100d_{12} - 3v_{11} - 3.5v_{12} \geq 0$$

$$100d_{13} - 1.5v_{11} - 2v_{12} - 3.25v_{13} \geq 0$$

$$100d_{14} - 0.5v_{11} - v_{12} - 1.75v_{13} - 3.7v_{14} \geq 0$$

$$\begin{aligned}
100d_{21} - 8v_{21} &\geq 0 \\
100d_{22} - 2v_{21} - 4v_{22} &\geq 0 \\
100d_{23} - 1.5v_{21} - 2v_{22} - 3.1v_{23} &\geq 0 \\
100d_{24} - 0.5v_{21} - v_{22} - 1.9v_{23} - 3.9v_{24} &\geq 0
\end{aligned}$$

Note that the above set of inequalities implies that customer response is linear with respect to advertising. While empirical evidence suggests that this is true over a limited range of expenditures, diminishing returns usually set in quickly [A10]. Therefore, a nonlinear concave response function would be more appropriate if we were operating near the point of saturation. Such an addition could be readily accommodated in the formulation but would require a different code than the one we actually used to find solutions.

11.3.1 Solution to Continuous BLP Model

We first address problem (11.2) which contains no integer variables. To obtain solutions we follow the Kuhn-Tucker approach described in Section 5.3.2 and replace the customer's problem with the corresponding Kuhn-Tucker conditions. As a consequence, the manufacturer now controls all the variables and is faced with a linear program, save for a single equality of the form $\sum_i u_i g_i = 0$ (where $u_i \geq 0$, is the Lagrange multiplier associated with the affine constraint $g_i \geq 0$, for all i in the reformulated version of (11.2)). This term is handled in the computations through implicit enumeration in the context of branch and bound using the Bard-Moore BLP code.

For the given set of data, the two players control 16 and 24 variables apiece. In its original form, the problem contains 37 structural constraints and 40 nonnegativity constraints. When recast using the Kuhn-Tucker approach, 25 additional constraints and 40 additional variables are required. The solution to problem (11.2) is contained in Table 11.4. As can be seen, production takes place in all but the third period where manufacturing costs are excessive. In this case, demand is met from inventory. Apparently, there is no need to subcontract because sufficient capacity exists in each period to satisfy customer demand, and setup costs are low enough to avoid carrying costs in all but period 2. Following this strategy, the manufacturer realizes a profit of \$68,317 and the customer a cost of \$173,273. The branch and bound algorithm used to solve this problem required 12.8 seconds of CPU time on a mainframe IBM 3081. The optimal solution was found at vertex 118, but was not confirmed until 267 vertices were explored. Since the schedule in Table 11.4 does not call for subcontracting, the same results apply for the more restrictive case given by the equivalent of (11.3).

Table 11.4 Optimal solution to continuous BLPP

Variable	Period			
	1	2	3	4
<i>Product 1</i>				
Production	75.0	121.3	0.0	53.4
Advertising	1500.0	0.0	1500.3	0.0
Demand	75.0	50.0	71.3	53.4
Inventory	0.0	71.3	0.0	0.0
Subcontracting	0.0	0.0	0.0	0.0
<i>Product 2</i>				
Production	440.0	349.0	0.0	77.4
Advertising	5500.0	0.0	1500.0	0.0
Demand	440.0	220.0	129.0	77.4
Inventory	0.0	129.0	0.0	0.0
Subcontracting	0.0	0.0	0.0	0.0

11.3.2 Noncooperative Implications of the Model

Two of the basic assumptions in the formulation of problems (11.1) and (11.2) are that the decisions are made sequentially and that cooperation among the players is prohibited. Consequently, it is natural to ask: is there another strategy that returns a higher profit to the manufacturer and still meets the customer's demand without increasing his costs? The answer is affirmative as shown in Table 11.5. This new solution yields a profit of \$73,705 for the manufacturer, while the customer's costs remain at \$173,273. The difference in results can be attributed to a reduction in advertising, accompanied by a redistribution of demand away from period 3 where production costs are highest.

Unfortunately, in order to achieve this solution which is Pareto-optimal, the players must negotiate under circumstances that strongly favor the leader. As it now stands, if the manufacturer announces the production and advertising strategy given in Table 11.5, the customer, in an effort to minimize his costs, responds with the decisions displayed in Table 11.5. The manufacturer is forced to carry more inventory in some periods than actually needed so his profits drop by about 28% to \$52,805. The customer's costs decline to \$162,686. The fact that the negative change in profit is significantly greater than the cost improvements implies that there is little incentive for the leader to cooperate. At most, he can hope to obtain a \$5,388 advantage over the original solution to problem (11.2).

Table 11.5 Pareto-optimal solution to continuous BLPP

Variable	Period			
	1	2	3	4
<i>Product 1</i>				
Production	89.5	59.7	30.0	70.5
Advertising	0.0	1500.0	0.0	1500.0
Demand	89.0	57.7	30.0	70.5
Inventory	0.0	0.0	0.0	0.0
Subcontracting	0.0	0.0	0.0	0.0
<i>Product 2</i>				
Production	370.2	443.1	0.0	53.2
Advertising	0.0	1500.0	1016.6	483.4
Demand	370.2	354.5	88.6	53.2
Inventory	0.0	88.8	0.0	0.0
Subcontracting	0.0	0.0	0.0	0.0

Table 11.6 Customer's response to strategy in Table 11.5

Variable	Period			
	1	2	3	4
<i>Product 1</i>				
Demand	78.7	52.5	30.0	88.5
Inventory	10.8	18.0	18.0	0.0
Subcontracting	0.0	0.0	0.0	0.0
<i>Product 2</i>				
Demand	370.2	185.1	61.5	249.7
Inventory	0.0	258.0	196.5	0.0
Subcontracting	0.0	0.0	0.0	0.0

11.3.3 Solution to Mixed–Integer BLP Model

When setup costs and upper bounds on inventory are included in the model, the manufacturer controls 24 variables (8 of which are binary), and the customer 28 (4 binary). The overall problem contains 45 constraints. From an algorithmic point of view, the introduction of zero–one variables in the formulation provides enormous complications. It was therefore necessary to develop a new set of routines to deal with

the discrete nature of the problem. The basic algorithm and its variants are discussed in Section 6.4.

The solution to problem (11.1) is displayed in Table 11.7 and yields a profit of \$47,430. The presence of setup and rental costs accounts for the decline, as well as a revision in strategy. The manufacturer now finds it beneficial to overproduce in period 1, and to place the surplus in inventory. This requires the rental of additional storage space in periods 1 and 2. Once again, demand for period 3 is filled out of inventory but now the economics favor subcontracting in period 4. As in the solution of the continuous version (11.2), though, no outside orders are placed when inventory is being held. This alleviates the need to solve the modified version of problem (11.1) which restricts $\sum_{jt} I_{jt} S_{jt}$ to zero.

Table 11.7 Optimal solution to mixed-integer BLPP

Variable	Period			
	1	2	3	4
<i>Product 1</i>				
Production	166.5	0.0	0.0	0.0
Advertising	881.2	618.8	1500.0	0.0
Demand	44.1	48.1	74.3	55.8
Inventory	122.4	74.3	0.0	0.0
Subcontracting	0.0	0.0	0.0	55.8
<i>Product 2</i>				
Production	789.0	0.0	0.0	0.0
Advertising	5500.0	0.0	1500.0	0.0
Demand	440.0	220.0	129.0	77.4
Inventory	349.0	129.0	0.0	0.0
Subcontracting	0.0	0.0	0.0	77.4

11.4 DISCUSSION OF RESULTS

Problem (11.1) was solved in slightly less than an hour on an IBM 3081 with the Moore-Bard mixed-integer code. The sharp increase in CPU time relative to problem (11.2) is typical of what occurs when discrete variables are added to a continuous formulation. A closer look at the mixed-integer BLP algorithm evidences a need to solve a subproblem equivalent in complexity to problem (11.2) at each iteration. This is the principal reason for the increase. In all, Bard and Moore examined a wide range of problems that included up to 200 decision variables and 80 structural constraints.

This was more than adequate for the company under investigation. Solution times averaged 15 minutes for the larger, continuous models.

From a planning perspective, the primary advantage of the bilevel approach is that it incorporates a feedback loop in the analysis. This provides management with both a coordination mechanism and the ability to gauge directly the impact of advertising on manufacturing. Strategies that are apt to overburden the production facility can be reevaluated before promises to customers exceed delivery capabilities. Also, as feasibility becomes less of an issue the job of constructing a master production schedule becomes more routine. If these benefits are to be realized, though, it is essential to maintain a complete and comprehensive data base. The utility of the proposed model is ultimately linked to a clear understanding of customer demand and the accompanying cost relationships.

Finally, it should be mentioned that in the development it was assumed that prices were fixed, and that customer demand remained unaffected by other sources of supply in the market. At best, this is only true in a completely competitive economy operating in equilibrium. In the more general case, prices fluctuate with both supply and demand, and customer behavior is influenced by the presence of other vendors. Nevertheless, if the relationships between p_{jt} and d_{jt} were known they could incorporated into problem (11.1) as additional constraints. The impact of additional suppliers might be taken into account by extending the current model along the lines discussed in [B3]. In that formulation, there is one leader and multiple followers. The general idea of a Stackelberg solution is maintained between the levels but solutions at the lower level, once the leader makes his decision, are defined as Pareto-optimal. This implies that economic efficiency is achieved among the followers.

DETERMINING PRICE SUPPORT LEVELS FOR BIOFUEL CROPS

12.1 INTRODUCTION

One of the first applications of bilevel programming dealt with the Mexican government's efforts to establish agricultural development policies [C1]. As leader, the government's objective was to maximize the sum of consumer and producer surplus. This was done by simulating the behavior of an atomistic market with profit-maximizing producers. Two sets of variables were defined. The first, called impact variables, included employment, farm income, the level of maize production, the level of wheat production, and the size of the government budget. Maize and wheat were singled out because of their role as basic food grains. The second set comprised the control variables and included fertilizer subsidies, subsidies on the interest rate charged against irrigation investment loans, purchases of maize and wheat at varying support prices levels, and water tax possibilities.

The lower-level component of the model described an irrigated agricultural region in northwestern Mexico covering around 250,000 hectares. Decision variables included: (a) 28 production activities for 7 crops; (b) 259 segmented demand activities (37 per crop); (c) 2 irrigation investment options; (c) 5 factor supply activities for long-term credit, short-term credit, agricultural chemicals, water, and machinery services; and (e) 15 risk transformation options. The constraints included input and production balance equations, availability constraints on land, water and labor, demand constraints, bounds on investments in irrigation, and limitations on choices of risk strategies.

In this chapter, a similar bilevel programming model is presented with the aim of helping decision makers arrive at a rational policy for cutting agricultural subsidies and improving air quality through reductions in automobile emissions. Taking the lead, the French government set out to determine what financial incentives were needed to motivate the petro-chemical industry to produce biofuels from farm crops. Examples of crops that can be used for this purpose are wheat, corn, rapeseed and sunflower among others. The stumbling block to such a policy is that industry's costs for pro-

ducing fuels from hydrocarbon-based raw materials is significantly less than it is for producing biofuels. Without incentives in the form of tax credits, industry will not buy farm output for conversion.

To state the problem succinctly, the government must determine the level of tax credits for each final product or biofuel that industry can produce while minimizing public outlays. A secondary objective is to realize some predefined level of land usage for nonfood crops. Industry is assumed to be neutral in this scenario and will produce any biofuel that is profitable. In the analysis, the agricultural sector is represented by a subset of farms in an agriculturally intensive region of France and is a profit maximizer. It uses the land available for nonfood crops only as long as the revenue generated from that activity exceeds the difference between the set-aside payments now received directly from the government and the maintenance costs incurred under the support program. If a farmer leaves a plot of land fallow and wishes to qualify for direct aide, he must plant a cover crop during the growing season to enrich the soil. Currently, 15% of the arable land (with some exceptions for very small farmers) must remain fallow and is thus eligible for this type of aide.

The conflict inherent in the problem is that the government wants to minimize its costs subject to a given level of land usage for nonfood crops while the agricultural sector wishes to maximize its profits subject to the technological constraints of production and certain agronomic constraints that are part of the regulatory program. A typical agronomic constraint is that no more than 47.6% of the arable land on each farm may be allocated to wheat production for both food and nonfood crops.

Government regulation is a good example of a classic leader-follower game in which the former sets the rules and the latter reacts, sometimes with unforeseen consequences. In public situations such as this, the decisions at the upper level are often complicated by special interest groups trying to impose their point of view on the regulatory agency. When those who pay the bills are not the direct beneficiaries of the policy, disagreement is inevitable. Accordingly, the problem can be investigated from several different points of view. In the developments to date [B8], it has been assumed that the government is the leader and fixes policy by specifying tax credit levels. Industry then sets the prices it is willing to pay at the farm gate for nonfood crops. Given these prices, the farm sector devises a usage plan for both food and nonfood crops, and land to be set aside.

We now describe the modeling details and proposed solution algorithms. The first is based on the idea of imposing a grid on the leader's variable set and solving the follower's problem, which turns out to be a linear program, for each point enumerated. The second is a more traditional nonlinear programming approach that assumes a standard model in which all functions are once continuously differentiable. SQP [F2] is used to find solutions. The results in either case are seen to be almost identical.

12.2 MATHEMATICAL MODEL

The following notation is used to describe the subsidy model under investigation.

Units

ha	hectare (10,000 square meters)
hl	hectoliter (100 liters)
t	metric tonne (1000 kilograms)
FF	French francs

Indices and sets

c	index for food crops; $c \in C = \{1, \dots, n_c\}$ (c' is index for sugar beets)
d	index for nonfood crops; $d \in D \subset C$
f	index for farms; $f \in F = \{1, \dots, n_f\}$
b	index for biofuels; $b \in B = \{1, \dots, n_b\}$
k	index for agronomic constraints; $k \in K = \{1, \dots, n_k\}$
$C(f)$	subset of food crops grown on farm f
$D(f)$	subset of nonfood crops grown on farm f
$D(b)$	subset of nonfood crops that can be used to make biofuel b
$L(b)$	subset of nonfood crops subject to capacity limitations in the production of biofuel b (ethanol only); $L(b) \subseteq D(b)$
$B(d)$	subset of biofuels that can be made from crop d
$G(f, k)$	set of food crops grown on farm f associated with agronomic constraint k
$H(f, k)$	set of nonfood crops grown on farm f associated with agronomic constraint k

Parameters

m_{cf}	gross margin (income) for food crop c grown on farm f (FF/ha)
r_{df}	yield of nonfood crop d grown on farm f (tonnes/ha)
α_{db}	factor for converting one tonne of nonfood crop d to biofuel b (hl/tonne)
β_{db}	cost of converting one unit of nonfood crop d to biofuel b (FF/hl)
c_{df}	production cost for nonfood crop d on farm f (FF/ha)
σ_f	total arable land available on farm f (ha)
σ'_f	land available on farm f for sugar beets (for sugar) (ha)
s_d	subsidy paid to farmers for nonfood crop d (FF/ha)
π_b	profit expected by industry for one unit of biofuel b (FF/hl)
v_b	market price for biofuel b (FF/hl)
o_{db}	market price of co-products associated with production of one unit of biofuel b from nonfood crop d (FF/hl)
w_f	multiplier used to scale up arable land of farm f

- L_b limitations on production of biofuel b from certain nonfood crops
 $L(b)$ (currently 3,000,000 hl/yr of ethanol from corn, wheat, and sugar beets only)
 ϕ_k maximum fraction of land permitted for crops included in agronomic constraint k
 δ_d indicator parameter equal to 1 if no subsidy is paid for nonfood crop d grown on land set aside; 0 otherwise
 θ fraction of arable land that must be set aside but could be used for production of nonfood crops (currently, $q = 0.15$)
 γ set-aside payment for fallow land (currently 1600 FF/ha)
 ρ fraction of set-aside land targeted by government for nonfood crop production (currently 0.20)
 η fraction of cake produced from one tonne of either rapeseed or sunflower (approximately 0.56 for both crops)
 μ large constant

Decision variables

- x_{cf} area allocated to food crop c on farm f (ha)
 xn_{df} area allocated to nonfood crop d on farm f (ha)
 xf_f area set aside on farm f (ha)
 p_d price at farm gate paid by industry for nonfood crop d (FF/tonne)
 τ_b government tax credit given to industry for biofuel b (FF/hl)

Government Model (leader)

$$\min \sum_{b \in B} \sum_{f \in F} \sum_{d \in D(f)} \alpha_{db} r_{df} xn_{df} \tau_b - \gamma \sum_{f \in F} \sum_{d \in D(f)} \delta_d xn_{df} \quad (12.1a)$$

$$\text{subject to } \sum_{f \in F} \sum_{d \in D(f)} xn_{df} \geq \rho \theta \sum_{f \in F} w_f \sigma_f \quad (12.1b)$$

$$\sum_{f \in F} \sum_{d \in D(f) \cap L(b)} \alpha_{db} r_{df} xn_{df} \leq L_b \quad \forall b \in B \quad (12.1c)$$

$$(\text{Industry Model}) \quad p_d \leq (\tau_b + v_b - \beta_{db} - \pi_b + o_{db}) \alpha_{db} \quad \forall d \in D, b \in B \quad (12.1d)$$

$$p_d \geq 0, \quad \tau_b \geq 0 \quad \forall d \in D, b \in B \quad (12.1e)$$

Agricultural Sector Model (follower)

$$\max \sum_{f \in F} \sum_{c \in C(f)} m_{cf} x_{cf} + \sum_{f \in F} \sum_{d \in D(f)} (p_d r_{df} + s_d - c_{df}) xn_{df} + \gamma \sum_{f \in F} xf_f \quad (12.1f)$$

$$\text{subject to } \sum_{c \in C(f)} x_{cf} + \sum_{d \in D(f)} xn_{df} + xf_f \leq w_f \sigma_f \quad \forall f \in F \quad (12.1g)$$

$$\sum_{d \in D(f)} xn_{df} + xf_f = \theta w_f \sigma_f \quad \forall f \in F \quad (12.1h)$$

$$x_{c'f} \leq w_f \sigma'_f \quad \forall f \in F \quad (12.1i)$$

$$\sum_{c \in C(f)} x_{cf} + \sum_{d \in D(f)} xn_{df} \leq \phi_k w_f \sigma_f \quad \forall f \in F, k \in K \quad (12.1j)$$

$$x_{cf} \geq 0, \quad xn_{df} \geq 0, \quad xf_f \geq 0 \quad \forall c \in C, d \in D, f \in F \quad (12.1k)$$

In the model, the government assumes the role of leader with the objective (12.1a) of minimizing the total value of tax credits given to the petro-chemical industry. The second term in (12.1a) reflects the savings to the government for not having to pay a premium, γ , to farmer f who grows nonfood crop d on land that would ordinarily be set aside. Currently, this applies only to sugar beets. When farmers grow sugar beets and only sugar beets on land set aside, they lose the subsidy per hectare on that land. But if sugar beet farms produce, say, wheat for ethanol or rapeseed for ester on land set aside, they keep the subsidy. This policy is dictated by Brussels in the larger context of the European Union.

The arable land available to farmer f in the sample group is denoted by σ_f . To account for the bias in the farm sample and to improve the fidelity of the model, each value of σ_f is scaled up by a unique multiplier denoted by w_f . This value reflects the yield, soil fertility, climate, and surface area of the specific farm with respect to a predetermined norm. The summation on the right-hand side of constraint (12.1b) is an approximation of the total arable land available in the region as a function of these factors. The parameter θ , currently equal to 0.15, is the proportion of arable land that must be set aside. If it is maintained properly while fallow, it is eligible for a direct government payment denoted by γ . Such agreements are common in most western countries.

As an alternative for the land set aside, the farmer is permitted to grow certain crops that can be converted to fuels by the petro-chemical industry. The requirement in (12.1b) that at least $\rho \times 100\%$ of the fallow land be used for the purpose of growing nonfood crops forces the government to grant tax credits since the cost to industry for producing fuels from biomass is currently greater than the cost of producing fuels from hydrocarbon sources. Note that ρ is a policy variable and is treated as a parameter in the model. Without incentives, industry would not buy any farm output for conversion. The price industry is willing to pay at the farm gate for nonfood crop d , denoted by p_d , must be high enough to offset the direct payment, γ , when subsidy

payments, s_d , and costs, c_{df} , are taken into account. The tradeoff at the farm level is reflected in the follower's objective function (12.1f) which will be discussed presently.

On the set-aside land, the agricultural sector can grow a variety of nonfood crops for sale to industry for biofuel production. Current installed capacity limits output in certain instances. Constraint (12.1c) restricts the production of biofuel b to L_b hectoliters per year.

In the model, it is assumed that industry is a neutral player whose only objective is to make a profit. Constraint (12.1d) assures that if crop d is purchased by industry to produce biofuel b , the tax credit τ_b is sufficiently high to cover costs β_{db} given the market price v_b , an expected profit π_b , as well as any revenues from co-products o_{db} . The neutrality assumption implies that the government will control the choices of p_d . A more elaborate model might include a third level where industry's production decisions were taken into account in more detail.

The objective (12.1f) of the agricultural sector, the second level in the model, is to maximize profits. It is assumed that as long as farm revenues exceed costs the set-aside land will be used for nonfood production. The problem for farmer f then is to decide how much of his land should be devoted to food crop c (x_{cf}), how much to nonfood crop d (xn_{cf}), and how much should remain fallow (xf_{cf}). It should be noted that a subsidy negotiated by the European Union is paid to all farmers for each crop. For food crop c , this value is taken into account in the calculation of the gross margin, m_{cf} , appearing in the first term of (12.1f); for nonfood crop d , this value, denoted by s_d , appears explicitly in the second term of (12.1f). The third term represents income from direct payments received by farmers for leaving a portion of their land fallow.

Constraint (12.1g) limits production by farm with the agricultural sector deciding the best use of the available land given the price p_d for nonfood crop d and gross margin m_{cf} for food crop c . Equation (12.1h) assures that $\theta \times 100\%$ of the arable land is either set aside or used to grow nonfood crops. Inequality (12.1i) enforces an upper limit on sugar beet production (c' is the corresponding index). Because sugar beets have the highest price supports they are one of the most profitable crops and must be treated separately in the model. The final set of constraints (12.1j) reflects agronomic considerations. The index k includes individual crops and groups of crops such as cereals. Each constraint limits output of the referenced crops on a particular farm f . For example, the output of oil crops, which comprise rapeseed, sunflower, and peas in the food category, and rapeseed and sunflower in the nonfood category, must not exceed 43.45% of the arable land.

12.3 DESCRIPTION OF ALGORITHMS

Model (12.1) is a nonlinear bilevel program and not easy to solve even for small instances. The only complicating nonlinearity, though, appears in the leader's objective function as the cross-product term, $xn_{df} \times \tau_b$ (surface area allocated to nonfood crop d on farm f \times tax credit for biofuel b). The cross-product term in the follower's objective function, $p_d \times xn_{df}$, (price at farm gate of nonfood crop d \times surface area allocated to nonfood crop d on farm f) is of little consequence because once p is chosen by the leader, all that the follower must do is solve a linear program. The additional fact that the follower's constraint region is independent of the leader's decision variables simplifies the overall solution process.

The data set used in the study comprises 393 farms (Appendix A contains parameter values). Each farmer can grow up to 7 food crops and 5 nonfood crops. The available options are defined in sets C and D .

$$\begin{aligned} C &= \{\text{wheat, barley, corn, sugar beet, rapeseed, sunflower, peas}\} \\ D &= \{\text{wheat, corn, sugar beet, rapeseed, sunflower}\} \end{aligned}$$

At present, the two types of biofuels being considered – ethanol and ester – give rise to the following conversion sets:

$$\begin{aligned} D(\text{ethanol}) &= \{\text{wheat, corn, sugar beet}\} \\ D(\text{ester}) &= \{\text{rapeseed, sunflower}\} \end{aligned}$$

Today about a half-dozen computer codes exist for solving the linear bilevel programming problem (see [B11, B24, H1, J4] or Chapter 5). At best, they can handle 200 leader variables, 100 follower variables and 50 constraints. When nonlinearities are present, the manageable problem size shrinks by nearly an order of magnitude (e.g., see [E1, S11, T2] or Chapters 7 and 8). Our problem has 7 leader variables (level of tax credits for ethanol and ester plus prices for the 5 nonfood crops), 3628 follower variables, 7 leader constraints and 3230 constraints in the agricultural sector model (12.1f)–(12.1k). This is much too big for any standard algorithm to solve. We have therefore taken an ad hoc approach and developed two distinct procedures that are shown to work well for the given application and accompanying data.

The basic idea in either case is to exploit the fact that once the biofuel tax credits are specified, the prices at the farm gate for nonfood crops can be readily computed from (12.1d). Given these prices, the agricultural sector model which was formulated and coded in GAMS [B27], reduces to a linear program (LP). We note that GAMS is used to solve this LP at the first iteration of our solution algorithms only. OSL [O2] is used independently of GAMS to solve all subsequent linear programs.

12.3.1 Industry Model

The industry sector in model (12.1) is used to determine prices at the farm gate for the nonfood crops. The computations depend on industry's expected profit, conversion costs, and market prices for biofuels and co-products. For algorithmic purposes, we can rewrite (12.1d) as

$$p_d = \max \{(\tau_b + v_b - \beta_{bd} - \pi_b + o_{db})\alpha_{db} : b \in B(d)\} \quad \forall d \in D \quad (12.1d')$$

where $D = \{\text{wheat, corn, sugar beet, rapeseed, sunflower}\}$; $B(\text{wheat}) = B(\text{corn}) = B(\text{sugar_beet}) = \{\text{ethanol}\}$ and $B(\text{rapeseed}) = B(\text{sunflower}) = \{\text{ester}\}$ in the current data set. Thus there is a unique relationship between crop conversion and biofuels; that is, $|B(d)| = 1$ for all $d \in D$. This means that the 'max' operator in eq. (12.1d') can be ignored. If a particular nonfood crop d could be converted into more than one biofuel b , then the 'max' operator would have to be used to compute the value of p_d .

12.3.2 Grid Search Algorithm (GSA)

In model (12.1), the leader (government) has control over τ and p ; however, once the values of τ_b are chosen p_d can be computed from (12.1d') so it is possible to view p as a function of τ , that is, $p = p(\tau)$. This relationship and the fact that $B = \{\text{ester, ethanol}\}$, implies that there are only two independent variables, a small enough number to impose a grid over their defined ranges and solve the government and farm models sequentially. Once values for nonfood crop prices are found and the surface area allocated to each such crop, the government's two constraints (12.1b) and (12.1c) can be evaluated to determine whether or not the solution is feasible to the overall problem.

The basic steps of the algorithm are presented below. The range of the grid search for τ_{ester} and τ_{ethanol} is as follows.

$$0 \leq \tau_{\text{ester}} \leq 230 \text{ FF/hl}$$

$$0 \leq \tau_{\text{ethanol}} \leq 330 \text{ FF/hl}$$

Step 1 We begin with a step size of 25 (denoted by *STEP*) and, for each point on the grid, compute the prices for corn, wheat, sugar beets, rapeseed, and sunflower using eq. (12.1d'). Next we solve the farm model (12.1f)–(12.1k) with these prices and compute the values for the government objective (12.1a), surface constraint (12.1b) denoted by *surf*, and ethanol production limit (12.1c) denoted by *ethlim* (there is no practical limit on ester production). All points that are feasible with respect to the two constraints are stored in a table and sorted in ascending order by the value of the government objective (*gobj*).

- Step 2 A candidate list is then constructed by marking first the best point on the list (first), and second the best point in each succeeding group of 10 points.
- Step 3 The step size is reduced from 25 to 5 and a new grid is constructed around each point on the candidate list. The grid is centered at the current value of τ_b , $b \in \{\text{ethanol, ester}\}$, and is defined over the interval $[\tau_b - \text{STEP_OLD}, \tau_b + \text{STEP_OLD}]$. Points are enumerated sequentially and evaluated as in Step 1 above. Solutions that satisfy constraints (12.1b) and (12.1c) are added to the table of feasible grid points.
- Step 4 Steps 2 and 3 are repeated with $\text{STEP} = 1$. The algorithm then terminates.
- Step 5 The full table of feasible points is re-sorted by g_{obj} and printed to file ‘grid.out’.
- Step 6 All points generated (feasible and infeasible) together with the government objective and constraint values, and attendant prices are written to file ‘oslmpls.out’ in a format suitable for spreadsheet importation.

The main advantage of the grid search procedure is that it avoids the difficulties resulting from the nondifferentiability of the relationship between the LP decision variables (x_{n_d}) and the nonfood crop prices (p_d). Any substantial increase in problem scale with respect to the number of biofuels or the number of nonfood crops would likely induce a combinatorial explosion. This would drastically reduce the efficiency of the procedure. In the next section, we present an alternative approach for dealing with this shortcoming.

12.3.3 Nonlinear Programming Approach

Recall that in the formulation of the model, the government is given control over the decision variables $\boldsymbol{\tau} \in T \subseteq R^{n_1}$ and $\mathbf{p} \in P \subseteq R^{n_2}$, while the agricultural sector collectively controls the vector $\mathbf{x} \in X \subseteq R^m$. The government goes first and attempts to minimize its objective function $F(\boldsymbol{\tau}, \mathbf{p}, \mathbf{x})$ over a feasible region defined by a set of functions in all problem variables. Because the government’s objective function also depends on the farm sector’s decisions, the former must anticipate each response or reaction of the follower before selecting a policy. Once the government makes a decision, the agricultural sector is faced with a traditional optimization problem of maximizing its objective function $f(\boldsymbol{\tau}, \mathbf{p}, \mathbf{x})$ over a feasible region $\{\mathbf{x} \in X : \mathbf{g}(\boldsymbol{\tau}, \mathbf{p}, \mathbf{x}) \leq 0\}$ which is partially defined by $\boldsymbol{\tau}$ and \mathbf{p} . In our case, though, the constraint set of the follower is independent of the leader’s policy variables so $\mathbf{g}(\boldsymbol{\tau}, \mathbf{p}, \mathbf{x}) = \mathbf{g}(\mathbf{x})$.

In simple mathematical terms, problem (12.1) can be written as

$$\min_{\tau \in T, p \in P} F(\tau, p, x) \quad (12.2a)$$

$$\text{subject to } G(\tau, p, x) \leq 0 \quad (12.2b)$$

$$\max_{x \in X} f(\tau, p, x) \quad (12.2c)$$

$$\text{subject to } g(x) \leq 0 \quad (12.2d)$$

where T , P and X place additional restrictions such as bounds on the decision variables, $G : R^{n_1+n_2+m} \rightarrow R^p$, and $g : R^m \rightarrow R^q$. For the current data set, $p = 7$ and $q = 3628$.

In bilevel programming, it is customary to view the follower's problem as parameterized in the leader's variables. We can thus write $x = x(\tau, p)$ and rewrite the leader's objective and constraint functions as $F(\tau, p, x(\tau, p))$ and $G(\tau, p, x(\tau, p))$, respectively. Unfortunately, the vector $x(\tau, p)$ which is returned from the solution of the follower's problem (12.2c)–(12.2d), is not necessarily differentiable everywhere or even continuous in τ and p . This makes it difficult to apply nonlinear programming theory to the BLPP (12.2) directly, hence the development of the grid search algorithm. Nevertheless, some progress can be made if we are willing to sacrifice theoretical rigor in favor of an “engineering” approach.

In particular, we note that although the government's problem depends on the decisions of individual farmers to allocate surface area to specific crops, the real dependence is on crop prices, which we have seen are determined from the biofuel tax credits; i.e., $p = p(\tau)$. Going one step further, the farm sector LP may be viewed as a function or subroutine that maps crop prices into allocation decisions. This dependence can be expressed as $x = x(p)$ without explicit refer to τ . The government model may then be formulated as a standard nonlinear program (NLP) with functions of the form $F(\tau, x(p))$ and $G(\tau, x(p))$; i.e.,

$$\min_{\tau, p} F(\tau, x(p)) \quad (12.3a)$$

$$\text{subject to } G_1(\tau, x(p)) \geq 0 \quad (12.3b)$$

$$G_2(\tau, x(p)) \geq 0 \quad (12.3c)$$

$$p_d - p_d(\tau_b) = 0 \quad \forall b \in B, d \in D(b) \quad (12.3d)$$

$$p_d \geq 0, 0 \leq \tau_b \leq \tau_{\max} \quad \forall d \in D, b \in B \quad (12.3f)$$

where (12.3b) and (12.3c) correspond to the government's constraints (12.1b) and (12.1c), and (12.3d) is equivalent to (12.1d'). This formulation has only 7 decision variables (τ_b , $b \in \{\text{ethanol, ester}\}$; p_d , $d \in D = \{\text{wheat, corn, sugar beet, rapeseed, sunflower}\}$) and 7 constraints so it can be handled easily by any NLP solver.

General Issues

All widely-used NLP solvers assume that all problem functions possess continuous first derivatives with respect to the decision variables. These derivatives are used to determine directions of movement and whether the optimality conditions are satisfied. As mentioned, $F(\tau, \mathbf{x}(p))$ and $\mathbf{G}(\tau, \mathbf{x}(p))$ are not continuously differentiable in τ and \mathbf{x} . These variables depend on the crop prices in a discontinuous manner, varying as the farm sector LP changes bases. For example, when a particular p_d is zero or small, output of nonfood crop d will be zero. As p_d increases beyond some threshold, output will jump to a positive level on a subset of farms. This jump corresponds to a basis change and demonstrates the discontinuous nature of $\mathbf{x}(p)$.

To apply an NLP code to model (12.3), the simplest approach is to ignore the discontinuities until they become too troublesome. Such codes generally default to estimating first derivatives by finite differences, computing an average rate of change in each function over a small change in each variable. The best that we can hope for in our case is to compute an average rate of change in the components of \mathbf{x} for a change in the components of \mathbf{p} that involves at least one basis change. Such estimates may or may not be strong enough to drive an NLP solver to a local optimum with any reliability. Here “reliability” is taken to mean that for different instances of the same model (different data) a point acceptably close to a local optimum will be found. Indication that the derivative estimates are well-behaved would be that the solver makes reasonably steady progress and routinely terminates when the fractional change in the objective value is below some predetermined threshold. When progress has stalled or the line search algorithm repeatedly fails, the natural conclusion would be that the derivative information is no longer useful.

To solve (12.3) with an NLP code, gradient information is needed with respect to the decision variables τ and \mathbf{p} . Two options are available for obtaining this information from the farm sector LP:

1. Determine a single finite difference perturbation factor for the prices that will induce at least one basis change. This is relatively simple to implement since many NLP codes allow the user to set the perturbation factor; in others the value is easily set in the source code.
2. A more precise method would be to exploit the sensitivity information in the LP solution by determining for each price, the minimum change required to induce a basis change. This approach would require that the differentiation routine in most NLP solvers be modified to allow a different perturbation step for each variable.

If neither of these methods results in reliable performance on the part of the NLP solver, it is reasonable to conclude that the inherent discontinuities in the problem cannot be ignored. In our implementation we took the first approach and exper-

mented with the standard codes GRG2 and SQP as the NLP solver. The latter turned out to be more suitable for our problem.

12.3.4 QP Formulation for Follower's Problem

In light of the uncertainty and limitations associated with obtaining derivative estimates for SQP with the LP follower problem, the question arises as to whether a more reliable and robust procedure might be developed. To address this issue, we note that the gradient of F with respect to (τ, \mathbf{p}) is given by

$$\nabla F(\tau, \mathbf{x}(\mathbf{p})) = [\nabla_\tau F(\tau, \mathbf{x}(\mathbf{p})), \mathbf{0}] + [0, \nabla_{\mathbf{x}} F(\tau, \mathbf{x}(\mathbf{p})) \nabla_{\mathbf{p}} \mathbf{x}(\mathbf{p})]$$

where $\nabla_{\mathbf{p}} \mathbf{x}(\mathbf{p})$ is the Jacobian of $\mathbf{x}(\mathbf{p})$, denoted by $J(\mathbf{p})$, and the ∇ operator produces a row vector. As mentioned, $\mathbf{x}(\mathbf{p})$ is not generally differentiable with respect to \mathbf{p} , or even known, so finding $J(\mathbf{p})$ is problematic [K5]. To circumvent this difficulty, we can add the smoothing term $-\varepsilon \|\mathbf{x} - \mathbf{x}_0\|^2$ to the follower's objective function, where \mathbf{x}_0 is a fixed reference point and $\varepsilon > 0$ is a small constant. This leads to a *regularized* version of the follower's problem in which we would have to find

$$\mathbf{x}(\mathbf{p}) = \arg \max \{f(\mathbf{x}, \mathbf{p}) - \varepsilon \|\mathbf{x} - \mathbf{x}_0\|^2 : \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \mathbf{x} \in X\}$$

and an element of $J(\mathbf{p})$. An "easy" case arises when strict complementarity holds and the active constraints are linearly independent, i.e., $\mathbf{x}(\cdot)$ is differentiable at \mathbf{p} . Then $J(\mathbf{p})$ can be computed via the implicit function theorem from the Kuhn-Tucker conditions for the associated equality constrained quadratic program (EQP) subproblem in which the inactive constraints are ignored and the active ones are treated as equalities. The remaining "hard" case is messy but again an engineering approach can be used. Specifically, the EQP subproblem may be derived by ignoring constraints with null Lagrange multipliers and those linearly dependent on the remaining ones. Then $J(\mathbf{p})$ can be computed from EQP as before.

An attempt was made to implement some of these ideas by augmenting the existing LP objective with a quadratic penalty term giving

$$\max (\mathbf{c}\mathbf{x} - \varepsilon \|\mathbf{x} - \mathbf{x}_0\|^2) \quad (12.4)$$

where \mathbf{x}_0 is the previous point at which the QP was solved. This was done by exploiting OSLs facility for *layering* a QP on top of an existing LP model. Specifically, that facility allows the user to solve QPs of the form

$$\begin{aligned} \max \quad & \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c} \mathbf{x} \\ \text{subject to} \quad & \mathbf{A} \mathbf{x} \leq \mathbf{b} \end{aligned}$$

by specifying the quadratic matrix \mathbf{Q} and then invoking the QP solution algorithm. We specified \mathbf{Q} to be $-2\varepsilon \mathbf{I}$, computed the coefficients for the linear terms, and then

added them to the corresponding coefficients in the LP objective. As an aside, we note that Falk and Liu (see Section 8.6) provide a related solution approach. They first propose a method to compute directional derivatives for \boldsymbol{x} with respect to \boldsymbol{p} and $\boldsymbol{\tau}$ from the follower's problem. This information is used to compute subgradients of F with respect to \boldsymbol{p} and $\boldsymbol{\tau}$. A bundle algorithm is then devised to solve the leaders problem. They show that the overall methodology converges to a regular point of the general BLPP under appropriate conditions.

12.4 IMPLEMENTATION

In working with large mathematic programs it is common to use an algebraic modeling language to generate a representation of the problem suitable for input to an optimization routine. In this project, the agricultural sector model (12.1f)–(12.1k) was originally formulated and coded in GAMS. This provided the starting point. All LPs arising in the computations were solved with OSL. It should be noted, however, that the methodology has no dependency on the use of any particular modeling language or solver.

In this section we describe the interface between GAMS and OSL as well as the design, structure, and use of the overall system herein referred to as BIOFUEL. The intent is to provide enough information for the reader to understand the logic, data flows and computations performed by the various modules and subroutines. All programs were written in FORTRAN and designed to be portable. Initial computations have been carried out on a Sun Sparcstation 10.

12.4.1 Overall System Design and Components

At the center of BIOFUEL is a GAMS LP model developed by the Institut National de la Recherche Agronomique that maximizes farm sector profit for a given set of nonfood crop prices. These prices are a function of the biofuel tax credits. The outputs from this LP are the farm level decisions as to how much land to devote to each crop and how much land to set aside. These decisions are inputs to the government problem which seeks to minimize out-of-pocket tax credits to the petrochemical industry subject to a set of constraints that reflects public policy and the physical limitations of production.

Since any procedure that attempts to solve the government problem must employ the farm sector model as a subsystem, we chose to extract the farm model from its GAMS representation rather than attempt to build a solution algorithm around the GAMS model. This approach allows freer experimentation with existing NLP solvers and

facilitates the development of customized algorithms to solve the government problem – the upper component of the bilevel programming problem given in (12.1a)–(12.1e).

Because BIOFUEL is a research code, most design decisions were made (1) to provide flexibility and ease of modification, (2) to facilitate the quick generation of results, (3) to minimize ‘hardcoding’ of problem characteristics, and (4) to allow for easy incorporation into larger systems. In the following subsections, we comment on these four issues and explain how they were handled.

Farm Model MPS File

Given that the farm model is a linear program, it is straightforward to instruct the LP solver invoked by GAMS to generate an MPS file for the farm model problem. This MPS file may then be supplied as input to whatever LP solver is incorporated in the larger solution system to solve the government problem. As mentioned, we have chosen OSL as our LP solver.

It is essential that the GAMS link to OSL be used to generate any MPS file for BIOFUEL. The MPS file produced by OSL contains the exact problem representation fed to OSL by GAMS. MPS files generated by other LP solvers are likely to vary and so may not match the required format. For example, the CPLEX MPS file contains a problem representation that reflects certain presolve operations and is no longer identical to the problem generated by GAMS. This prevents straightforward use of knowledge of the GAMS model structure to modify the LP problem represented by the MPS file.

GAMS Model Coefficient Updating

In the discussion that follows, the term ‘base case’ refers to the LP defined by the GAMS farm model (together with fixed prices, yields, and other specific data). This model takes as inputs the prices of the five nonfood crops under consideration and produces as output the number of hectares dedicated to each crop on each farm, as well as the amount of land set aside. These crop prices appear only in the objective function (12.1f) and correspond to $\tau_{\text{ester}} = 230$ and $\tau_{\text{ethanol}} = 330$. Hence, to solve the farm model for a new set of prices, the only changes required are to the objective coefficients that depend on the price(s) which have changed from the base case represented by the MPS file. Moreover, since only objective coefficients change between cases, if the LP has an optimal solution for the base case, it will have an optimal solution for any other set of prices.

The individual objective coefficient values ($p_d r_{df} + s_d - c_{df}$) in the second term in (12.1f) express the net unit profit (revenue – costs) associated with each nonfood crop d on each farm f . The revenue terms are computed as (price \times yield + subsidy). To appropriately modify the LP objective function to reflect a new set of prices, we must know which objective coefficients correspond to each combination of crop and

farm (i.e., the expansion of the GAMS model expressions involving the crop and farm sets) as well as the price for the base case and the yield for the crop/farm combination represented by each term. A new price will be reflected by computing ‘base_coefficient_value – old_price × yield + new_price × yield’ for each objective term which depends on that crop price.

To perform these computations, we need to know the exact structure of the objective function. The only source of this information is the GAMS listing file which contains the expansion of each function in the model into its elementary terms. For the current data set, the first few lines of that expansion are given below.

```
OBJ.. 4557*X(F1,WHEAT) + 4331*X(F1,BARLEY) + 3948*X(F1,SUNFLOWER)
      + 6123*X(F1,SUGAR_BEET) + 6411*X(F1,PEAS) + 4995*X(F2,WHEAT)
      + 4057*X(F2,BARLEY) + 2241*X(F2,CORN) + 12262*X(F2,SUGAR_BEET) + ...
```

Consider the highlighted first term. Here, **4557** is the base case coefficient for the decision variable set X (we only extract terms for variable set XS which represents allocation of area to nonfood crops), **WHEAT** the crop for this allocation decision, and **F1** the index for farm 1. The first module in BIOFUEL (see Fig. 12.1) processes the GAMS listing file together with the data tables used as input to the GAMS model. The final output of this module is a file containing the base case coefficient, corresponding index, crop, farm, and yield for each term involving an XS variable. This file is read into the farm model subsystem (see Fig. 12.2) to assist in the generation of modified LPs for arbitrary price changes. (The obvious disadvantage associated with extracting coefficient information from the GAMS listing file is the dependency placed on the exact layout and construction of that file. If subsequent releases of GAMS change the format, our code will have to be modified accordingly. A more stable procedure would be to extract the needed information through calls to GAMS functions which directly access the internal data structures and symbol table. At present, however, there is no convenient way to do this.)

Overview of System

In what we call the preprocessing stage, the GAMS model is run for the base case and a listing is produced. The input data for the GAMS model are contained in six files. The listing file is then read by the program **PARSEOBJ** and the objective function coefficients for the XN variables are extracted. A second program called **MAPOBJ** is used to merge yield data and objective coefficient information. Finally, an MPS file is produced by OSL.

The data files generated during the preprocessing stage are used as input by the calling system of the main program. This is shown in Fig. 12.2. A separate set of subprograms has been written to implement the sequence of operations listed below which leads to a solution of the government problem. The central calling system

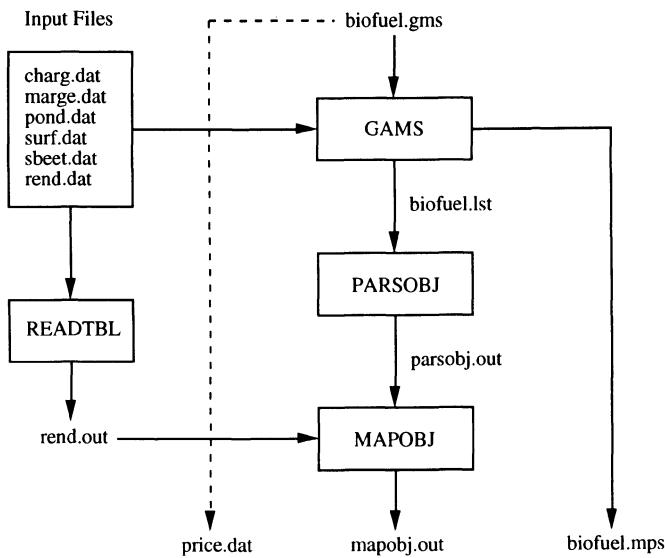


Figure 12.1 BIOFUEL preprocessing data flow

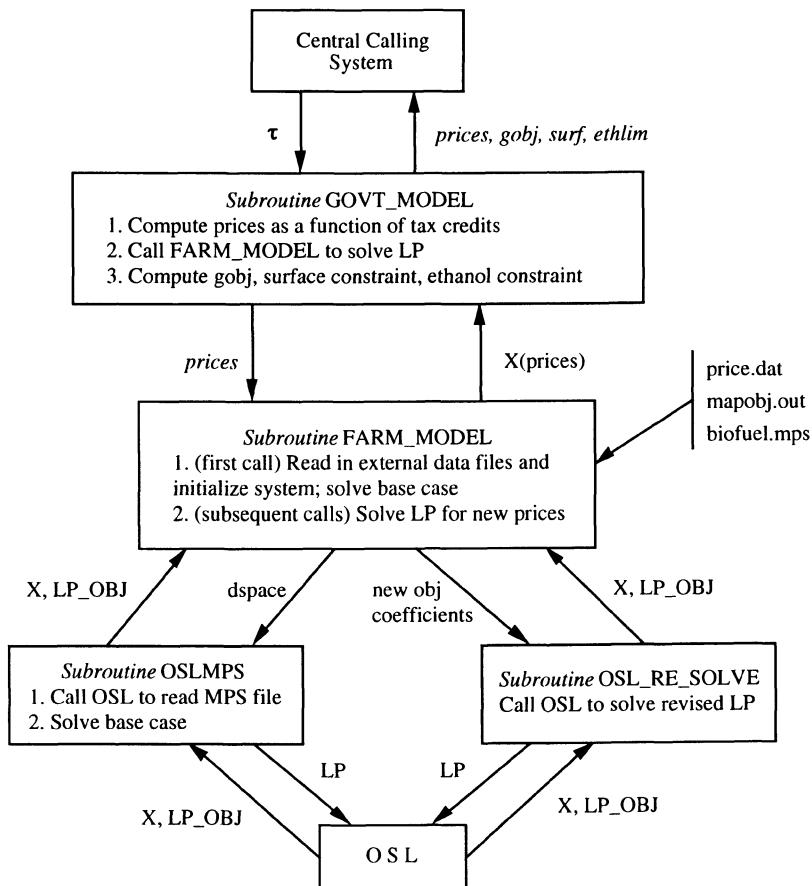
should be thought of as either the grid search algorithm highlighted in Fig. 12.3 or the SQP-based algorithm depicted in Fig. 12.4. The generic steps of the system are:

1. Accept as input a set of tax credits for nonfood crops.
2. Compute a set of nonfood crop prices based on these tax credits.
3. Apply the revised prices to the farm model objective function.
4. Solve the farm model LP.
5. Evaluate and return to the calling system the values associated with the government objective function and constraints.

12.4.2 GAMS Model Structure Determination

As discussed above, a sequence of programs is used to extract the structure of the terms in the objective function as expanded by GAMS and written to the GAMS listing file. These programs are bound into one module. The GAMS model processing goes as follows:

1. The yield values are contained in the GAMS include file ‘rend.dat’. This is a table with columns for each crop and rows for each farm. Program **READTBL**



- Notes:
- LP denotes the farm sector LP presented to OSL
 - dspace is the workspace in which OSL builds the problem
 - LP_OBJ denotes the optimal LP objective function value
 - X denotes the optimal land allocations in the LP solution

Figure 12.2 BIOFUEL module structure and data flow

reads in ‘rend.dat’, converts the data into the triplets <farm><crop><yield>, then writes each triplet (one to a record) to file ‘rend.out’.

2. Program PARSEOBJ reads in the name of the GAMS listing file to be processed from data file ‘parseobj.dat’. It then reads that file until the marker for the objective function text (namely, OBJ..) is encountered. At that point, it begins to read each line of the expanded objective function text from the listing file, breaks it into its component terms (and counts the terms for indexing pur-

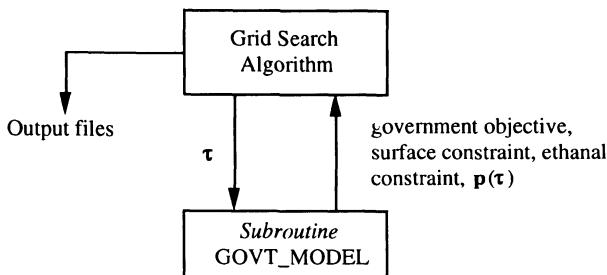


Figure 12.3 Central calling structure for grid search algorithm

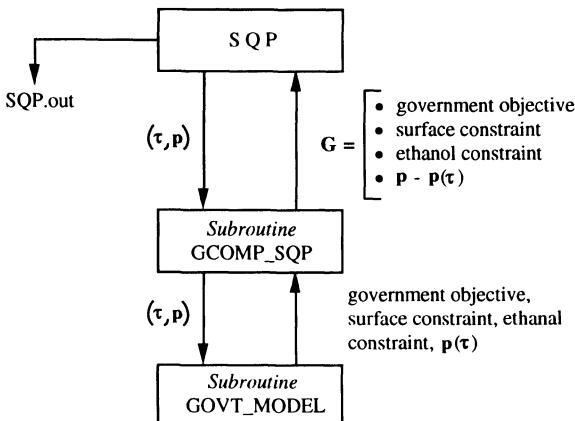


Figure 12.4 Central calling structure for SQP

poses), and parses or separates out the elements of each term. The elements associate with the XS variables (surface allocation for nonfood crop on set-aside land) are written to file ‘parseobj.out’ one term per line. For example, the term 2030*XS(WHEAT,E200) will generate a record in ‘parseobj.out’ of the form: [nnnn 2030 XS WHEAT E200] where nnnn is the index of the location of this term in the function (e.g., nnnn = 2134 indicates that we are at term number 2134 counting from the left).

3. Program **MAPOBJ** merges the yield values generated by **READTBL** with the objective function information generated by **PARSEOBJ**. The file ‘rend.out’ is first read in and then each record of ‘parseobj.out’ is read one at a time. The yield value corresponding to the crop and farm for that objective term is retrieved and the value is appended to the objective term information. The resultant records are written to file ‘mapobj.out’.

The file ‘mapobj.out’ is used as input to the farm model subroutine to supply the yield values for each relevant objective term. It also supplies the base case coefficient values for checking purposes to ensure that the coefficients (and yields) correspond to the coefficients in the MPS file.

12.4.3 Model Evaluation Subsystem

Subroutine **FARM_MODEL**

The original design goal was to have all input data related to the GAMS model enter the system through subroutine **FARM_MODEL** which is called by subroutine **GOVT_MODEL**. This is the way things are currently done although it may be advantageous to modify the calling sequence in the future. The main input to **FARM_MODEL** is a data file whose name is stored in character variable ‘model_file’. The value of ‘model_file’ is passed to subroutine **FARM_MODEL** by its calling program; ‘model_file’ is organized as follows:

- Record 1: name of MPS file
- Record 2: max or min (sense of optimization to be sent to OSL)
- Record 3: name of coefficient/yield file (from program **MAPOBJ**)
- Record 4: name of file containing base case prices and τ values
- Record 5: debug print level
- Record 6: debug case number (switch to higher debug level at case nnnn)
- Record 7: debug case level (print level to switch to at case ##)

Base Case Processing

The major modes of subroutine **FARM_MODEL** are controlled by **status_flag** which is an input argument. If **status_flag** = INIT, **FARM_MODEL** initializes a new run. The initialization call performs the following operations:

1. Reads in and stores the information in ‘model_file’.
2. Calls subroutine **OSLMPS** to read in the MPS file and solve the base case. The name of the MPS file (OSL reads from FORTRAN unit 98) and the sense of optimization are passed to **OSLMPS**. This subroutine also extracts the objective row coefficients (and column indices) and returns these to **FARM_MODEL**.
3. Stores the base case objective coefficients in ‘base_obj_coefficients’ and copies these values to the appropriate locations (mapped by array ‘colnbrs’) in the OSL objective area of ‘dspace’ (the OSL working array pointers into ‘dspace’ are obtained through calls to **EKKNGET**).
4. GAMS represents the LP objective in the form ‘max Z subject to $f(X) - Z = 0$, where $f(X)$ is the actual objective function and Z is a free variable. Consequently, the objective vector in the MPS file produced by OSL will consist of

all zeros with the exception of a ‘-1’ in the position for Z . The actual values are stored in the LP coefficient matrix as row 1 (this is the way GAMS formulates a problem so it is not advisable to alter this convention). To facilitate easy modifications of objective coefficient values (OSL allocates a dense vector for the objective coefficients) and to avoid the need to extract data from the coefficient matrix, the base case coefficients are copied into ‘dspace’ and the coefficient for Z is set to 0 (in ‘base_obj_coefficients’ as well). As a result, Z becomes a true free variable, row 1 is ignored, and the objective coefficients are stored in their proper positions in ‘dspace’.

5. Calls subroutine **CHECKOBJ** to read in the objective coefficient file generated by **MAPOBJ**, stores the yield values for each XS objective term, and checks the coefficients for each XS against the corresponding values stored in ‘dspace’. If any discrepancies are detected, error messages are issued and execution terminated.
6. Calls subroutine **GET_PRICES** to read in the base case prices and τ values (government biofuel tax credits).
7. Calls **EKKIGET/EKKISET** to set OSL integer control variable 2 to 88. This will redirect all OSL output to unit 88 (opened as ‘oslmsg.out’) to avoid cluttering the standard output (unit *).

When all base case processing is completed, **status_flag** is set to **SOLVE** and control returns to the calling program, **GOVT_MODEL**. At that time, **FARM_MODEL** returns the crop names, base case prices, problem size parameters, farm index set, and XS index set (if **xn_loaded(i) = .true.** then objective term i belongs to the variable set XS).

*‘Solve’ Mode for Subroutine **FARM_MODEL***

When **FARM_MODEL** is called with **status_flag = SOLVE**, the following sequence of operations occurs:

1. Array ‘obj_coefficients’ is reset to the values from ‘base_obj_coefficients’.
2. The set of crop prices sent to **FARM_MODEL** is examined to determine which prices differ from the base case values. The objective terms that depend on the changed prices are flagged. Subroutine **MOD_OBJ_ROW** is then called to compute the new objective coefficients for those terms.
3. The values in ‘obj_coefficients’ are copied to the OSL objective vector area in ‘dspace’.
4. Subroutine **OSL_RE_SOLVE** is called to restart OSL with the new objective coefficients.
5. Subroutine **FARM_MODEL** returns the LP objective value in **OBJ_OUT** and the LP decision variable values in **X_OUT**.

Subroutine GOVT_MODEL

Subroutine **GOVT_MODEL** sits on top of **FARM_MODEL** and calculates an instance of the overall bilevel model for an arbitrary set of τ (biofuel tax credit) values. The first time it is called, all the data structures are set up and the base case is solved. Subsequent iterations are controlled by the particular calling program.

Model Initialization

The first time **GOVT_MODEL** is called **status_flag** is initialized to **INIT** in a data statement so it will call **FARM_MODEL** for an initialization run. The sequence of operations performed for **status_flag = INIT** is as follows:

1. **FARM_MODEL** is called with **status_flag = INIT** (see section above) to solve the base case and set up the appropriate data structures.
2. Subroutine **INDICES** is called to set symbolic indices for crops and biofuels. This allow for more readable code (i.e., expressions of the form ‘price(wheat) = ...’ may be written).
3. Subroutine **SET_PARMS** is called to set values for other parameters that are needed for price computation (such as α , β , and π defined in Appendix A). These data are hardcoded but should be moved to an external data file to facilitate data changes without recompilation (and a more intimate linkage to the GAMS model data). As a general design principle, all numeric data should be specified in only one place and passed to all other places where it is needed by standard exchange methods.
4. Subroutine **GET_EXTERNAL_DATA** is called to read σ_f (farm surface area) and w_f (farm scaling factor) values from the GAMS include files ‘surf.dat’ and ‘pond.dat’, respectively.
5. Array ‘colnbrs_inv’ is constructed. This array is the inverse of the ‘colnbrs’ index set that is returned from OSL for the row 1 coefficients. When **colnbrs(j) = k**, the j th coefficient in the LP row under consideration (in this case row 1) is in LP column k ; **colnbrs_inv(j) = k** implies that LP column j corresponds to row 1 coefficient (and hence objective term) k .
6. Base case values for the government objective function and constraints are computed and returned to the called system (see Steps 3 - 8 below).

Noninitialization Calls to GOVT_MODEL

The first call to **GOVT_MODEL** solves the base case and sets up the data structures. In all subsequent calls, the following sequence of operations occurs. Note that all government inequality constraints are assumed to be written in the form $G(\mathbf{x}) \geq 0$ so nonnegative values indicate feasibility.

1. The values of τ passed through the argument list are used to compute the nonfood crop prices. An input parameter, `price_floor`, determines whether negative prices are allowed. For the grid search algorithm `price_floor = .true.` so all negative prices are set to 0. To avoid additional discontinuities when `GOVT_MODEL` is called as a function evaluation routine from the NLP algorithm, `price_floor` may be set to `.false.` In that case, negative prices that result are left as is because they won't affect the production decisions.
2. `FARM_MODEL` is called with `status_flag = SOLVE` to solve the LP model for the new set of prices.
3. The government objective value (12.1a) is computed and returned as $g(1)$.
4. The surface feasibility constraint (12.1b) value is computed and returned as $g(2)$.
5. The ethanol production constraint (12.1c) value is computed and returned as $g(3)$. The right-hand side, L_{ethanol} , is currently hardcoded as 3,000,000.
6. The LP objective value is returned as $g(4)$.
7. The prices are returned in '`price_out`'.
8. Various terms in the government model, represented in the GAMS model as free variables, are returned in $g(5)$ through $g(9)$.

12.5 COMPUTATIONAL RESULTS

All algorithms have been coded in FORTRAN 77. The grid search and initial SQP runs with the LP follower problem were made on a Sun Sparcstation 10. The SQP runs with the LP follower problem and the corresponding runs with the QP follower problem were made on a Pentium 100 system using the WATCOM FORTRAN 77 v10.6 compiler.

12.5.1 Grid Search Solutions

GSA takes approximately 30 minutes to run on the Sun. A high level diagram of the basic routines is given in Fig. 12.3 where the call to the `GOVT_MODEL` refers to Fig. 12.2. At each iteration, the LP farm model with 3628 variables and 3230 constraints is reoptimized with OSL. The best solution obtained for a grid size of 25 occurs at $\tau_{\text{ester}} = 125$, $p_{\text{rapeseed}} = 400.5$, $p_{\text{sunflower}} = 347.8$, $\tau_{\text{ethanol}} = p_{\text{wheat}} = p_{\text{corn}} = p_{\text{sugar_beet}} = 0$ with $g_{\text{obj}} = 2.11 \times 10^7$, $\text{surf} = 4786.0$ and $\text{ethlim} = 3 \times 10^6$. The last value implies that ethanol is not produced.

After a series of refinements to the grid, first cutting the step size to 5 and then to 1 as indicated in Steps 3 and 4 of Section 12.3.2, we get the best solution at $\tau_{\text{ester}} = 117$, $p_{\text{rapeseed}} = 365$, $p_{\text{sunflower}} = 310$, $\tau_{\text{ethanol}} = p_{\text{wheat}} = p_{\text{corn}} = p_{\text{sugar-beet}} = 0$ with $gobj = 2.06 \times 10^7$, $surf = 48.8$ and $ethlim = 3 \times 10^6$. (In fact, the output indicates that $\tau_{\text{ethanol}} = 5$, but since there is no ethanol production we can interpret this as zero.) Table 12.1 presents a sampling of output in the neighborhood of this solution. The rows are sorted by the government objective value. As can be seen, the first two entries are not feasible because not enough surface area is being farmed. In addition, there is no ethanol production until the tax credit reach about 260 FF/hl.

A final point to note about the results is that the base case, with $\tau_{\text{ester}} = 230$, $\tau_{\text{ethanol}} = 330$, $p_{\text{rapeseed}} = 873$, $p_{\text{sunflower}} = 841$, $p_{\text{wheat}} = 511$, $p_{\text{corn}} = 581$, and $p_{\text{sugar-beet}} = 140$, yields $gobj = 1.05 \times 10^9$, $surf = 4.96 \times 10^5$ and $ethlim = -2.17 \times 10^5$ which is not feasible. That is, ethanol is overproduced by a significant amount. Also, the corresponding cost to the government in about 50 times higher than the best solution found.

Table 12.1 Sample output from grid search algorithm

τ_{ethanol} (FF/hl)	τ_{ester} (FF/hl)	Gov_obj (10^5)	Farm_obj (10^6)	Surface (ha)	Ethlim (10^6)
0	115	188.49	1918.688	-770	3.00
5	116	193.82	1918.854	-560	3.00
5	117	206.32	1919.026	49	3.00
260	115	209.96	1918.709	-499	2.99
5	118	224.75	1919.205	971	3.00
5	119	241.99	1919.403	1857	3.00
265	115	242.28	1918.768	-131	2.98
0	120	250.36	1919.608	2212	3.00
5	121	260.89	1919.822	2639	3.00
260	120	271.04	1919.628	2435	2.99
5	122	273.59	1920.041	3203	3.00
5	123	283.69	1920.267	3593	3.00
270	115	307.97	1918.931	635	2.96

12.5.2 Output from SQP

A high level diagram of the SQP-based code is given in Fig. 12.4 where the call to the GOVT_MODEL refers to Fig. 12.2. We examined the performance of this system from three starting points for τ_{ester} : (i) $\tau_{\text{ester}} = 10.0$ (far below the grid search optimum of 117); (ii) $\tau_{\text{ester}} = 125.0$ (in the neighborhood of 117); and (iii) $\tau_{\text{ester}} = 230.0$ (far above 117). Each employed the same initial prices ($p_{\text{wheat}} = 5.0$, $p_{\text{corn}} = 5.0$,

$p_{\text{sugar-beet}} = 5.0$, $p_{\text{rapeseed}} = 400.5$, $p_{\text{sunflower}} = 347.8$). The first three prices are a considerable distance from the optimum; the latter two are much closer. In any real-world instance, the current and past prices would be known, so any systematic examination of algorithmic performance for prices far away from the optimum or some known base point is difficult to justify. The second variable, τ_{ethanol} , was started at 250 in each case.

Because of the large differences in magnitudes of the functions involved, we scaled the government objective by 10^{-9} and the surface and ethanol constraints by 10^{-4} . After some experimentation, we settled on an SQP penalty weight of 10.0 and a derivative perturbation step of 0.01. The results were encouraging though not definitive. The SQP runs generated a solution path that moves monotonically towards feasibility, obtains feasibility, then monotonically decreases the objective until several successive line searches fail and the algorithm terminates on the criterion ‘all remedies have failed to find a better point’. This behavior is about as much as can be expected. The process eventually runs out of usable derivative information and is unable to compute a direction of descent. Table 12.2 gives the results from each starting point for the SQP runs with the LP follower problem. In each case, the value of τ_{ethanol} converged to around 190 which is well below the trigger point of 260 for ethanol production. This implies that τ_{ethanol} can be taken as zero.

Table 12.2 SQP runs with LP follower formulation

τ_{ester} (initial)	τ_{ester} (final)	Gov_obj (10^8)	SQP iterations	CPU time (sec)
10.0	117.67	2.08078	26	240
125.0	117.06	2.06415	20	191
230.0	117.05	2.06408	22	209

We also ran SQP using the QP follower problem formulation over the same cases for three different values of the penalty weight $\varepsilon \in \{10^{-10}, 10^{-6}, 10^{-4}\}$ in model (12.4). For the first two values of ε , the results were identical to those in Table 12.2. The results for $\varepsilon = 10^{-4}$ are given in Table 12.3. The corresponding CPU times are somewhat exaggerated as we had difficulties with OSL’s warm start option for its QP solver. For unknown reasons the QP warm starts periodically locked up the Pentium, so all QP runs were cold started (including those solved during finite difference derivative computations). Fixing this problem should result in at least an order of magnitude improvement in CPU time, although the approach would still be slower by a factor of 2 to 4.

When started from $\tau_{\text{ester}} = 230$, both formulations yielded values almost identical to the best values obtained by the grid search; a similar statement can be made for the first formulation for $\tau_{\text{ester}} = 125$. In either case, the runs started from $\tau_{\text{ester}} = 10.0$

Table 12.3 SQP runs with QP follower formulation for $\epsilon = 10^{-4}$

τ_{ester} (initial)	τ_{ester} (final)	Gov_obj (10^8)	SQP iterations	CPU time (sec)
10.0	117.67	2.08078	26	10820
125.0	117.06	2.08060	17	7162
230.0	117.05	2.06408	22	9381

terminated at a slightly inferior point. Notably, both formulations yielded identical results for the first and third starting points. If the QP formulation is generating better derivative information, it cannot be inferred from these runs since the LP formulation attains what appears to be the optimum with no greater algorithmic difficulties (e.g., line search failures) than the QP formulation. Examination of the detailed output confirmed that SQP followed the same path in either case.

12.6 DISCUSSION

The main advantage of the grid search procedure is that it avoids the difficulties resulting from the nondifferentiability of the relationship between the LP decision variables (x_{n_d}) and the nonfood crop prices (p_d). Any substantial increase in problem scale with respect to the number of biofuels or the number of nonfood crops would likely induce a combinatorial explosion. This would drastically reduce the efficiency of the procedure. Nevertheless, for our problem instance characterized by a small number of upper level variables and constraints and a large number of lower level variables and constraints, the approach proved remarkably reliable.

With regard to SQP, we note that it is not a feasible path algorithm so it may move into and out of the feasible region. For a problem such as this where the Kuhn-Tucker optimality test is not relevant and the solution process is expected to terminate by satisfying the fractional objective change criterion or with the ‘all remedies failed’ condition, it is quite probable that the final point will not be feasible. It is essential therefore to store and update the best feasible point visited by the solver and ultimately report the incumbent as the solution.

A final and somewhat unsettling point about the NLP approach is that it has a number of weaknesses that cannot be brushed aside. Although we were able to find what appeared to be the optimal solution, this may not always be the case. It is quite possible that for different data or a different LP subproblem the algorithm will get bogged down immediately with bad derivative information and not be able to

proceed. Should this occur, starting the NLP solver at different point might provide a fix. More experimentation is needed to gain a greater understanding of this issue.

APPENDIX A

Data Set for Subsidy Model

Definition of data elements

$\alpha(d, b)$ = factor for converting one tonne of nonfood crop d to biofuel b (hl/tonne)

$\beta(d, b)$ = cost of converting nonfood crop d to biofuel b (FF/hl)

$\phi(d, b)$ = market price of co-products associated with production of one unit of biofuel b from nonfood crop d (FF/hl)

$\pi(b)$ = profit expected by industry for one unit of biofuel b (FF/hl)

$v(b)$ = market price for one unit of biofuel b (FF/hl)

$\tau(b)$ = government tax credit given to industry for biofuel b (FF/hl)

$D = \{\text{wheat, corn, sugar beet, rapeseed, sunflower}\}$ = set of nonfood crops

$B = \{\text{ester, ethanol}\}$ = set of biofuels

Data for model ('no' indicates that conversion is not possible)

$\alpha(\text{wheat, ester}) = \text{no}$

$\alpha(\text{corn, ester}) = \text{no}$

$\alpha(\text{sugar_beet, ester}) = \text{no}$

$\alpha(\text{rapeseed, ester}) = 4.5 \text{ hl/tonne}$

$\alpha(\text{sunflower, ester}) = 4.7 \text{ hl/tonne}$

$\alpha(\text{wheat, ethanol}) = 3.5 \text{ hl/tonne}$

$\alpha(\text{corn, ethanol}) = 3.8 \text{ hl/tonne}$

$\alpha(\text{sugar_beet, ethanol}) = 1 \text{ hl/tonne}$

$\alpha(\text{rapeseed, ethanol}) = \text{no}$

$\alpha(\text{sunflower, ethanol}) = \text{no}$

$\beta(\text{wheat, ester}) = \text{no}$

$\beta(\text{corn, ester}) = \text{no}$

$\beta(\text{sugar_beet, ester}) = \text{no}$

$\beta(\text{rapeseed, ester}) = 168 \text{ FF/hl}$

$\beta(\text{sunflower, ester}) = 168 \text{ FF/hl}$

$\beta(\text{wheat, ethanol}) = 207 \text{ FF/hl}$

$\beta(\text{corn, ethanol}) = 207 \text{ FF/hl}$

$$\beta(\text{sugar_beet, ethanol}) = 130 \text{ FF/hl}$$

$$\beta(\text{rapeseed, ethanol}) = \text{no}$$

$$\beta(\text{sunflower, ethanol}) = \text{no}$$

$$o(\text{wheat, ester}) = \text{no}$$

$$o(\text{corn, ester}) = \text{no}$$

$$o(\text{sugar_beet, ester}) = \text{no}$$

$$o(\text{rapeseed, ester}) = 120 \text{ FF/hl}$$

$$o(\text{sunflower, ester}) = 105 \text{ FF/hl}$$

$$o(\text{wheat, ethanol}) = 83 \text{ FF/hl}$$

$$o(\text{corn, ethanol}) = 90 \text{ FF/hl}$$

$$o(\text{sugar_beet, ethanol}) = 0$$

$$o(\text{rapeseed, ethanol}) = \text{no}$$

$$o(\text{sunflower, ethanol}) = \text{no}$$

$$\pi(\text{ester}) = 60 \text{ FF/hl}$$

$$\pi(\text{ethanol}) = 120 \text{ FF/hl}$$

$$v(\text{ester}) = 72 \text{ FF/hl}$$

$$v(\text{ethanol}) = 60 \text{ FF/hl}$$

Range on tax credits

$$0 \leq \tau(\text{ester}) \leq 230 \text{ FF/hl}$$

$$0 \leq \tau(\text{ethanol}) \leq 330 \text{ FF/hl}$$

REFERENCES

- [A1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ (1993).
- [A2] E. Aiyoshi and K. Shimizu, "Hierarchical Decentralized Systems and Its New Solution by a Barrier Method," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. SMC-11, No. 6, pp. 444–449 (1981).
- [A3] E. Aiyoshi and K. Shimizu, "A Solution Method for the Static Constrained Stackelberg Problem via Penalty Method," *IEEE Trans. Automatic Control*, Vol. AC-29, No. 12, pp. 1111–1114 (1984).
- [A4] F.A. Al-Khayyal, "Minimizing a Quasiconcave Function over a Convex Set: A Case Solvable by Lagrangian Duality," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 661–663, Tucson, AZ (1985).
- [A5] F.A. Al-Khayyal, R. Horst and P.M. Pardalos, "Global Optimization of Concave Functions subject to Quadratic Constraints: An Application in Nonlinear Bilevel Programming," *Annals of Operations Research*, Vol. 34, No. 1-4 pp. 125–147.
- [A6] G. Anandalingam and V. Apprey, "Multi-Level Programming and Conflict Resolution," *European Journal of Operational Research*, Vol. 51, pp. 233–247 (1991).
- [A7] G. Anandalingam and T.L. Friesz (eds.), "Hierarchical Optimization," *Annals of Operations Research*, Vol. 34, No. 1-4 (1992).
- [A8] G. Anandalingam, R. Mathieu, L. Pittard and R. Sinha, "Artificial Intelligence Based Approaches for Hierarchical Optimization," in R. Sharda et al. (eds.), *Impact of Recent Computer Advances in Operations Research*, North-Holland, New York (1989).
- [A9] G. Anandalingam and D.J. White, "A Solution Method for the Linear Stackelberg Problem Using Penalty Functions," *IEEE Trans. Automatic Control*, Vol. 35, No. 10, pp. 1170–1173 (1990).
- [A10] A. Arsham and D. Dianich, "Consumer Buying Behavior and Optimal Advertising Strategy: The Quadratic Profit Function Case," *Computers & Operations Research*, Vol. 15, No. 4, pp. 299–310 (1988).

- [B1] A. Bachem and M. Grötschel, “New Aspects of Polyhedral Theory,” in B. Korte (ed.), *Modern Applied Mathematics, Optimization and Operations Research*, pp. 51–106, North-Holland, Amsterdam (1982).
- [B2] J.F. Bard, “An Efficient Point Algorithm for a Linear Two-Stage Optimization Problem,” *Operations Research*, Vol. 31, No. 4, pp. 670–684 (1983).
- [B3] J.F. Bard, “Coordination of a Multidivisional Firm through Two Levels of Management,” *Omega*, Vol. 11, No. 5, pp. 457–465 (1983).
- [B4] J.F. Bard, “An Investigation of the Linear Three Level Programming Problem,” *IEEE Trans. Systems, Man, and Cybernetics*, Vol. SMC-14, No. 5, pp. 711–717 (1984).
- [B5] J.F. Bard, “Optimality Conditions for the Bilevel Programming Problem,” *Naval Research Logistics Quarterly*, Vol. 31, pp. 13–26 (1984).
- [B6] J.F. Bard, “Convex Two-Level Optimization,” *Mathematical Programming*, Vol. 40, pp. 15–27 (1988).
- [B7] J.F. Bard, “Some Properties of the Bilevel Programming Problem,” *J. of Optimiz. Theory & Appl.*, Vol. 68, No. 2, pp. 371–378 (1991).
- [B8] J.F. Bard, “Solution Algorithms for the Government–Agriculture Bilevel Programming Model,” Progress Report, prepared for Institut National de la Recherche Agronomique (INRA), Thiverval-Grignon, France, by Department of Mechanical Engineering, University of Texas, Austin (1996).
- [B9] J.F. Bard and S. Chatterjee, “Objective Function Bounds for the Inexact Linear Programming Problem with Generalized Cost Coefficients,” *Computers & Operations Research*, Vol. 12, No. 5, pp. 483–491 (1985).
- [B10] J.F. Bard and J.E. Falk, “An Explicit Solution to the Multi-Level Programming Problem,” *Computers & Operations Research*, Vol. 9, No. 1, pp. 77–100 (1982).
- [B11] J.F. Bard and J.T. Moore, “A Branch and Bound Algorithm for the Bilevel Programming Problem,” *SIAM Journal of Scientific and Statistical Computing*, Vol. 11, No. 2, pp. 281–292 (1990).
- [B12] J.F. Bard and J.T. Moore, “Production Planning with Variable Demand,” *Omega*, Vol. 18, No. 1, pp. 35–42 (1990).
- [B13] J.F. Bard and J.T. Moore, “An Algorithm for the Discrete Bilevel Programming Problem,” *Naval Research Logistics*, Vol. 39, pp. 419–435 (1992).
- [B14] T. Baser and G.J. Olsder, *Dynamic Noncooperative Game Theory*, Academic Press, London (1982).

- [B15] M.S. Bazaraa, J.J. Jarvis and H.D. Sherali, *Linear Programming and Network Flows*, Second Edition, John Wiley & Sons, New York (1990).
- [B16] M.S. Bazaraa, H.D. Sherali and C.M. Shetty, *Nonlinear Programming: Theory and Algorithms*, Second Edition, John Wiley & Sons, New York (1993).
- [B17] O. Ben-Ayed, "Bilevel Linear Programming: Analysis and Application to the Network Design Problem," Ph.D. Dissertation, University of Illinois, Urbana-Champaign, IL (1988).
- [B18] O. Ben-Ayed and C.E. Blair, "Computational Difficulties of Bilevel Linear Programming," *Operations Research*, Vol. 38, No. 3, pp. 556–560 (1990).
- [B19] O. Ben-Ayed, C.E. Blair, D.E. Boyce and L.J. LeBlanc, "Construction of a Real-World Bilevel Programming Model of the Highway Network Design Problem," *Annals of Operations Research*, Vol. 34, No. 1-4, pp. 219–254 (1992).
- [B20] O. Ben-Ayed, D.E. Boyce and C.E. Blair, "A General Bilevel Linear Programming Formulation of the Network Design Problem," *Transportation Research*, Vol. B22, pp. 311–318 (1988).
- [B21] D.P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York (1982).
- [B22] D. Bertsimas and J.N. Tsitsiklis, *Introduction to Linear Optimization*, Athena Scientific, Belmont, MA (1997).
- [B23] W.F. Bialas and M.H. Karwan, "On Two-Level Optimization," *IEEE Trans. Automatic Control*, Vol. AC-27, No. 1, pp. 211–214 (1982).
- [B24] W.F. Bialas and M.H. Karwan "Two-Level Linear Programming," *Management Science*, Vol. 30, No. 8, pp. 1004–1020 (1984).
- [B25] R.E. Bixby, "Progress in Linear Programming," *ORSA Journal on Computing*, Vol. 6, No. 1, pp. 15–22 (1994).
- [B26] J. Bracken and J.T. McGill, "Mathematical Programs with Optimization Problems in the Constraints," *Operations Research*, Vol. 21, No. 1 pp. 37–44 (1973).
- [B27] A. Brooke, D. Kendrick and A. Meerhaus, *GAMS: A User's Guide*, Boyd & Frazer, Danvers, MA (1992).
- [C1] W. Candler and R. Norton, "Multi-Level Programming and Development Policy," Working Paper No. 258, World Bank, Washington DC (May 1977).
- [C2] W. Candler and R. Townsley, "A Linear Two-Level Programming Problem," *Computers & Operations Research*, Vol. 9, No. 1, pp. 59–76 (1982).

- [C3] Y. Chen and M. Florian, "The Nonlinear Bilevel Programming Problem: A General Formulation and Optimality Conditions," Research Report, CRT-794, Centre de Recherche sur les Transports, Université de Montréal, Québec (1991).
- [C4] I. C. Chow, C.L. Monma and D.F. Shanno, "Further Development of a Primal-Dual Interior Point Method," *ORSA Journal on Computing*, Vol. 2, No. 4, pp. 304–311 (1990).
- [C5] F.H. Clarke, *Optimization and Nonsmooth Analysis*, John Wiley & Sons, New York (1983).
- [D1] J.M. Danskin, "The Theory of Max-Min with Applications," *SIAM J. of Applied Mathematics*, Vol. 14, No. 4, pp. 641–664 (1966).
- [D2] S. Dempe, "A Necessary and a Sufficient Optimality Condition for Bilevel Programming Problems," *Optimization*, Vol. 25, pp. 341–354 (1992).
- [D3] A.H. deSilva and G.P. McCormick, "Implicitly Defined Optimization Problems," *Annals of Operations Research*, Vol. 34, No. 1-4, pp. 107–124 (1992).
- [D4] Y.M.I. Dirickx and L.P. Jennergren, *Systems Analysis by Multilevel Methods with Applications to Economics and Management*, John Wiley & Sons, New York (1979).
- [E1] T.A. Edmunds and J.F. Bard, "Algorithms for Nonlinear Bilevel Mathematical Programs," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. SMC-21, No. 1, pp. 83–89 (1991).
- [E2] T.A. Edmunds and J.F. Bard, "An Algorithm for the Mixed-Integer Nonlinear Bilevel Programming Problem," *Annals of Operations Research*, Vol. 34, pp. 149–162 (1992).
- [F1] J.E. Falk and J. Liu, "On Bilevel Programming, Part 1: General Nonlinear Cases," *Mathematical Programming*, Vol. 70, No. 1, pp. 47–72 (1995).
- [F2] Y. Fan, S. Sarkar and L. Lasdon, "Experiments with Successive Quadratic Programming Algorithms," *J. of Optimiz. Theory & Appl.*, Vol. 56, No. 3, pp. 359–383 (1988).
- [F3] A.V. Fiacco, *Introduction to Sensitivity Analysis in Nonlinear Programming*, Academic Press, New York (1983).
- [F4] A.V. Fiacco and J. Liu, "Degeneracy in NLP and the Development of Results Motivated by its Presence," *Annals of Operations Research*, Vol. 46, pp. 61–80 (1993).
- [F5] A.V. Fiacco and G.P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Technique*, John Wiley & Sons, New York (1968).

- [F6] R. Fletcher, *Practical Methods of Optimization*, Second Edition, John Wiley & Sons, New York (1987).
- [F7] J. Fortuny-Amat and B. McCarl, “A Representation and Economic Interpretation of a Two-Level Programming Problem,” *Journal of the Operational Research Society*, Vol. 32, pp. 783–792 (1981).
- [F8] J.R. Freeland and N. Baker, “Goal Partitioning in a Hierarchical Organization,” *Omega*, Vol. 3, No. 6, pp. 673–688 (1975).
- [G1] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York (1979).
- [G2] S.I. Gass, *Linear Programming: Methods and Applications*, Fifth Edition, McGraw Hill, New York (1985).
- [G3] J. Gauvin and R. Janin, “Directional Behaviour of Optimal Solutions in Nonlinear Mathematical Programming,” *Mathematics of Operations Research*, Vol. 13, No. 4, pp. 629–649 (1988).
- [G4] K.R. Gehner, “Necessary and Sufficient Optimality Conditions for the Fritz John Problem with Linear Equality Constraints,” *SIAM J. of Control*, Vol. 12, No. 1, pp. 140–149 (1974).
- [G5] M. Gendreau, P. Marcotte and G. Savard, “A Hybrid Tabu–Ascent Algorithm for the Linear Bilevel Programming Problem,” *Journal of Global Optimization*, Vol. 8, No. 3, pp. 217–233 (1996).
- [G6] A.M. Geoffrion, “Primal Resource–Directive Approaches for Optimizing Nonlinear Decomposable Systems,” *Operations Research*, Vol. 18, No. 3, pp. 375–403 (1970).
- [G7] P.E. Gill, W. Murray and M.H. Wright, *Practical Optimization*, Academic Press, London (1981).
- [G8] F. Glover, “Convexity Cuts and Cut Search,” *Operations Research*, Vol. 21, pp. 123–134 (1973).
- [G9] F. Glover, “Tabu Search – Part I,” *ORSA Journal on Computing*, Vol. 1, No. 3, pp. 190–206 (1989).
- [G10] F. Glover, “Tabu Search: A Tutorial,” *Interfaces*, Vol. 20, No. 4, pp. 74–94 (1990).
- [G11] D.E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, Reading, MA (1989).
- [G12] J. Grefenstette, “Optimization of Control Parameters for Genetic Algorithms,” *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 16., No. 1, pp. 122–128 (1986).

- [G13] M. Grötschel and M.W. Padberg, “Polyhedral Theory,” in E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds.), *The Traveling Salesman Problem: A guided Tour of Combinatorial Optimization*, Chapter 8, pp. 251–305, John Wiley & Sons, New York (1985).
- [G14] M. Grötschel and M.W. Padberg, “Polyhedral Computations,” in E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds.), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Chapter 9, pp. 307–360, John Wiley & Sons, New York (1985).
- [H1] P. Hansen, B. Jaumard and G. Savard, “New Branch-and-Bound Rules for Linear Bilevel Programming,” *SIAM Journal of Scientific and Statistical Computing*, Vol. 13, No. 5, pp. 1194–1217 (1992).
- [H2] P.T. Harker and J.S. Pang, “Existence of Optimal Solutions to Mathematical Programs with Equilibrium Constraints,” *Operations Research Letters*, Vol. 7, pp. 61–64 (1988).
- [H3] P.M.J. Harris, “Pivot Selection Methods for the Devex LP Code,” *Mathematical Programming*, Vol. 5, pp. 1–28 (1973).
- [H4] A. Haurie, G. Savard and D.J. White, “A Note on: An Efficient Point Algorithm for a Linear Two-Stage Optimization Problem,” *Operations Research*, Vol. 38, No. 3, pp. 553–555.
- [H5] K.L. Hoffman and M.W. Padberg, “Solving Airline Crew Scheduling Problems by Branch-and-Cut,” *Management Science*, Vol. 39, No. 6, pp. 657–682 (1993).
- [H6] W.W. Hogan, “Directional Derivatives for Extremal-Value Functions with Application to the Completely Convex Case,” *Operations Research*, Vol. 21, pp. 188–209 (1973).
- [H7] W.W. Hogan, “Point-to-Set Maps in Mathematical Programming,” *SIAM Review*, Vol. 15, pp. 591–603 (1973).
- [H8] H. Holland, “Adaptive Algorithms for Discovering and Using General Patterns in Growing Knowledge Bases,” *Int'l J. Policy Analysis and Information Systems*, Vol. 4, No. 3, pp. 345–268 (1980).
- [H9] J. N. Hooker, “Karmarkar’s Linear Programming Algorithm,” *Interfaces*, Vol. 16, No. 4, pp. 75–90 (1986).
- [H10] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*, Third Edition, Springer-Verlag, Berlin (1995).
- [I1] Y. Ishizuka, “Optimality Conditions for Directionally Differentiable Multi-Objective Programming Problems,” *J. of Optimiz. Theory & Appl.*, Vol. 72, No. 1, pp. 91–111 (1992).

- [I2] Y. Ishizuka and E. Aiyoshi, "Double Penalty Method for Bilevel Programming Problems," *Annals of Operations Research*, Vol. 34, No. 1-4, pp. 73–88 (1992).
- [I3] Y. Ishizuka and K. Shimizu, "Necessary and Sufficient Conditions for the Efficient Solutions of Nondifferentiable Multi-Objective Problems," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. SMC-14, No. 4, pp. 624–629 (1984).
- [J1] B. Jaumard, G. Savard and X. Xiong, "An Exact Algorithm for Convex Bilevel Programming," Working paper G-95-33, École des Hautes Études Commerciales, École Polytechnique de Montréal, Québec (1995).
- [J2] R.G. Jeroslow, "The Polynomial Hierarchy and a Simple Model for Competitive Analysis," *Mathematical Programming*, Vol. 32, pp. 146–164 (1985).
- [J3] K. Jittorntrum, "Solution Point Differentiability without Strict Complementarity in Nonlinear Programming," *Mathematical Programming Study* 21, pp. 127–138, North-Holland, Amsterdam (1984).
- [J4] J.J. Júdice and A.M. Faustino, "A Sequential LCP Method for Bilevel Linear Programming," *Annals of Operations Research*, Vol. 34, No. 1-4, pp. 89–106 (1992).
- [J5] J.J. Júdice and A.M. Faustino, "A Linear–Quadratic Bilevel Programming Problem," *INFOR*, Vol. 32, No. 2, pp. 87–98 (1994).
- [K1] B. Kalantari and J.B. Rosen, "Penalty for Zero–One Equivalent Problem," *Mathematical Programming*, Vol. 24, pp. 229–232 (1982).
- [K2] N. Karmarkar, "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica*, Vol. 4, pp. 373–395 (1984).
- [K3] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598, pp. 671–680 (1983).
- [K4] K.C. Kiwiel, *Methods of Descent for Nondifferentiable Optimization*, Lecture Notes in Mathematics, Vol. 1133, Springer-Verlag, Berlin (1985).
- [K5] K.C. Kiwiel, *Private communications*, Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland (Oct. 1995).
- [K6] C.D. Kolstad and L.S. Lasdon, "Derivative Evaluation and Computational Experience with Large Bilevel Mathematical Programs," *J. of Optimiz. Theory & Appl.*, Vol. 65, pp. 485–499 (1990).
- [K7] G. Kontoravdis and J.F. Bard, "Implementing Branch-and-Cut for the Vehicle Routing Problem with Time Windows," Working paper, Department of Mechanical Engineering, University of Texas, Austin (1997).

- [K8] J. Kyparisis, “Sensitivity Analysis for Nonlinear Programs and Variational Inequalities with Nonunique Multipliers,” *Mathematics of Operations Research*, Vol. 15, No. 2, pp. 286–298, (1990).
- [L1] L. Lasdon, J. Plummer and A. Warren, “Nonlinear Programming,” in M. Avriel and B. Golany (eds.), *Mathematical Programming for Industrial Engineers*, Chapter 6, pp. 385–485, Marcel Dekker, New York (1996).
- [L2] L.J. LeBlanc and D.E. Boyce, “A Bilevel Programming Algorithm for Exact Solution of the Network Design Problem with User–Optimal Flows,” *Transportation Research*, Vol. 20B, pp. 259–265 (1986).
- [L3] C. Lemaréchal, “Nondifferentiable Optimization,” in G.L. Nemhauser, A.H.G. Rinnooy Kan, and M. J. Todd (eds.), *Handbooks in Operations Research and Management Science*, Vol. 1, *Optimization*, North-Holland, Amsterdam, Chapter VII, pp. 529–572 (1989).
- [L4] J. Liu, “Sensitivity Analysis in Nonlinear Programs and Variational Inequalities via Continuous Selections,” *Journal on Control and Optimization*, Vol. 33, pp. 1040–1060 (1995).
- [L5] P. Loridan and J. Morgan, “New Results on Approximate Solutions in Two–Level Optimization,” *Optimization*, Vol. 20, pp. 819–836 (1989).
- [L6] M. Lu and K. Shimizu, “A Global Optimization Technique for Solving Nonlinear Programming Problem with Equality and Inequality Constraints and Its Application to the Bilevel Programming Problem,” (in Japanese), *Trans. of the Society of Instrument and Control Engineers*, Vol. 28, No. 7, pp. 879–886 (1992).
- [L7] D.G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Second Edition, Addison Wesley, Reading, MA (1984).
- [L8] I.J. Lustig, R.E. Marsten and D.F. Shanno, “Computational Experience with a Primal–Dual Interior Point Method,” *Linear Algebra and Its Applications*, Vol. 152, pp. 191–222 (1991).
- [M1] C.M. Macal and A.P. Hurter, “Dependence of Bilevel Mathematical Programs on Irrelevant Constraints,” *Computers & Operations Research*, Vol. 24, No. 12, pp. 1129–1140 (1997).
- [M2] T.M. Magee and F. Glover, “Integer Programming,” in M. Avriel and B. Golany (eds.), *Mathematical Programming for Industrial Engineers*, Chapter 3, pp. 123–269, Marcel Dekker, New York (1996).
- [M3] P. Marcotte, “Network Design Problem with Congestion Effects: A Case of Bilevel Programming,” *Mathematical Programming*, Vol. 34, pp. 142–162 (1986).

- [M4] P. Marcotte and G. Savard, "A Note on the Pareto Optimality of Solutions to the Linear Bilevel Programming Problem," *Computers & Operations Research*, Vol. 18, No. 4, pp. 355–359 (1991).
- [M5] P. Marcotte and G. Savard, "Novel Approaches to the Discrimination Problem," *Zeitschrift für Operations Research*, Vol. 36, pp. 517–545 (1992).
- [M6] R.E. Marsten, "The Design of the XMP Linear Programming Library," *ACM Transactions on Mathematical Software*, Vol. 7, No. 4, pp. 481–497 (1981).
- [M7] R.E. Marsten, M.J. Saltzman, D.F. Shanno, G.S. Pierce and J.F. Ballintijn, "Implementation of a Dual Affine Interior Point Algorithm for Linear Programming," *ORSA Journal on Computing*, Vol. 1, No. 4, pp. 287–297 (1989).
- [M8] R.E. Marsten, R. Subramanian, M. Saltzman, I. Lustig and D. Shanno, "Interior Point Methods for Linear Programming: Just Call Newton, Lagrange, and Fiacco and McCormick," *Interfaces*, Vol. 20, No. 4, pp. 105–116 (1990).
- [M9] T.H. Matheiss and D.S. Rubin, "A Survey and Comparison of Methods for Finding all Vertices of a Convex Polyhedral Sets," *Operations Research*, Vol. 5, No. 2, pp. 167–185 (1980).
- [M10] R. Mathieu, L. Pittard and G. Anandalingam, "Genetic Algorithm Based Approach to Bi-Level Linear Programming," *recherche opérationnelle*, Vol. 28, No. 1, pp. 1–21 (1994).
- [M11] W. Maxwell, J.A. Muckstadt, L.J. Thomas and J. VanderEechen, "A Modeling Framework for Planning and Control of Production in Discrete Parts Manufacturing and Assembly Systems," *Interfaces*, Vol. 13, No. 6, pp. 92–104 (1983).
- [M12] P.G. McKeown, "A Vertex Ranking Procedure for Solving the Linear Fixed Charge Problem," *Operations Research*, Vol. 23, No. 6, pp. 1183–1191 (1975).
- [M13] Z. Michalewicz, "Evolutionary Computation Techniques for Nonlinear Programming Problems," *International Transactions in Operational Research*, Vol. 1, No. 2, pp. 223–240 (1994).
- [M14] A. Migdalas, P.M. Pardalos and P. Värbrand (eds.), *Multilevel Optimization: Algorithms and Applications*, Kluwer Academic Publishers, Boston (1998).
- [M15] J.T. Moore and J.F. Bard, "The Mixed Integer Linear Bilevel Programming Problem," *Operations Research*, Vol. 38, No. 5, pp. 911–921 (1990).
- [M16] E.K. Morlok, *Introduction to Transportation Engineering and Planning*, McGraw-Hill, New York (1978).
- [M17] K.G. Murty, "Adjacency on Convex Polyhedra," *SIAM Review*, Vol. 13, No. 3, pp. 377–386 (1971).

- [M18] K.G. Murty, *Linear Programming*, John Wiley & Sons, New York (1983).
- [M19] K.G. Murty, *Linear Complementarity, Linear and Nonlinear Programming*, Heldermann Verlag, Berlin (1988).
- [N1] S.G. Nash and A. Sofer, “A Barrier Method for Large-Scale Constrained Optimization,” *ORSA Journal on Computing*, Vol. 5, No. 1, pp. 40–53 (1993).
- [N2] S.G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw Hill, New York (1996).
- [N3] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York (1988).
- [N4] D. Nguyen, “An Analysis of Optimal Advertising under Uncertainty,” *Management Science*, Vol. 31, No. 5, pp. 622–633 (1985).
- [O1] W. Ogryczak, “A Note on Modeling Multiple Choice Requirements for Simple Mixed Integer Programming Solvers,” *Computers & Operations Research*, Vol. 23, No. 2, pp. 199–205 (1996).
- [O2] OSL, *Optimization Subroutine Library: Guide and Reference*, IBM Corp., Dept. 55JA, Poughkeepsie, NY (1995).
- [O3] J.V. Outrata, “Necessary Optimality Conditions for Stackelberg Problems,” *J. of Optimiz. Theory & Appl.*, Vol. 76, No. 2, pp. 305–320 (1993).
- [P1] M.W. Padberg and G. Rinaldi, “A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems,” *SIAM Review*, Vol. 33, No. 1, pp. 60–100 (1991).
- [P2] J.S. Pang and D. Ralph, “Piecewise Smoothness, Local Invertibility, and Parametric Analysis of Normal Maps,” Preprint Series No. 26, Department of Mathematics, University of Melbourne, Australia (1993).
- [P3] P.M. Pardalos, “Construction of Test Problems in Quadratic Bivalent Programming,” *ACM Transactions on Mathematical Software*, Vol. 17, No. 1, pp. 74–87 (1991).
- [P4] P.M. Pardalos and G. Schnitger, “Checking Local Optimality in Constrained Quadratic Programming is NP-hard,” *Operations Research Letters*, Vol. 7, pp. 33–35 (1988).
- [R1] A.H.G. Rinnooy and G.T. Timmer, “Stochastic Global Optimization Methods, Part I: Clustering Methods and Part II: Multi-Level Methods,” *Mathematical Programming*, Vol. 39, No. 1, pp. 27–78 (1987).
- [R2] S.M. Robinson, “Normal Maps Induced by Linear Transformations,” *Mathematics of Operations Research*, Vol. 17, No. 2, pp. 691–714 (1992).

- [R3] G.M. Roodman, “Postoptimality Analysis in Zero–One Programming by Implicit Enumeration,” *Naval Research Logistics Quarterly*, Vol. 19, No. 3, pp. 435–447 (1972).
- [R4] T. Ruefli, “A Generalized Goal Decomposition Model,” *Management Science*, Vol. 17, No. 9, pp. B505–B518 (1971).
- [S1] H.M. Salkin and K. Mathur, *Foundations of Integer Programming*, North-Holland, Amsterdam (1989).
- [S2] G. Savard and J. Gauvin, “The Steepest Descent Direction for the Nonlinear Bilevel Programming Problem,” *Operations Research Letters*, Vol. 15, pp. 265–272 (1994).
- [S3] H. Schramm and J. Zowe, “A Version of the Bundle Idea for Minimizing a Nonsmooth Function: Conceptual Idea, Convergence Analysis, Numerical Results,” *SIAM Journal of Optimization*, Vol. 2, pp. 121–152 (1992).
- [S4] J.K. Sengupta, *Decision Methods in Stochastic Programming: Operations Methods of Decision Making Under Uncertainty*, North-Holland, Amsterdam (1982).
- [S5] SETEC Economie et SOTINFOR, Plan Directeur Routier, Rapport Général Tomes 1 et 2, République Tunisienne, Ministère de l’Equipement, Direction des Ponts et Chausées, Sous-Direction des Etudes (1982).
- [S6] H.D. Sherali, “A Multiple Leader Stackelberg Model and Analysis,” *Operations Research*, Vol. 32, No. 2, pp. 390–404 (1984).
- [S7] K. Shimizu, “Large-Scale and Two-Level Mathematical Programming,” in H. Tamura and T. Yoshikawa (eds.), *Large-Scale Systems Control and Decision Making*, Chapter 6, pp. 151–200, Marcel Dekker, New York (1990).
- [S8] K. Shimizu and E. Aiyoshi, “New Computational Method for Stackelberg and Min-Max Problems by Use of a Penalty Method,” *IEEE Trans. Automatic Control*, Vol. AC-26, No. 2, pp. 460–466 (1981).
- [S9] K. Shimizu and Y. Ishizuka, “Optimality Conditions and Algorithms for Parameter Design Problems with Two-Level Structure,” *IEEE Trans. Automatic Control*, Vol. AC-30, No. 10, pp. 986–993 (1985).
- [S10] K. Shimizu, Y. Ishizuka and J.F. Bard, *Nondifferentiable and Two-Level Mathematical Programming*, Kluwer Academic Publishers, Boston (1997).
- [S11] K. Shimizu and M. Lu, “A Global Optimization Method for the Stackelberg Problem with Convex Functions via Problem Transformations and Concave Programming,” *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 25, No. 12, pp. 1635–1640 (1995).

- [S12] M. Simaan, “Stackelberg Optimization of Two-Level Systems,” *IEEE Trans. Systems, Man, and Cybernetics*, Vol. SMC-7, No. 4, pp. 554–556 (1977).
- [S13] M. Simaan and J.B. Cruz, Jr., “On the Stackelberg Strategy in Nonzero-Sum Games,” *J. of Optimiz. Theory & Appl.*, Vol. 11, No. 5, pp. 535–555 (1973).
- [S14] A. L. Soyster, “Inexact Linear Programming with Generalized Resource Sets,” *European Journal of Operational Research*, Vol. 3, No. 4, pp. 316–321 (1979).
- [S15] J.E. Spingarn, “Second-Order Conditions that are Necessary with Probability One,” in A.V. Fiacco (ed.), *Mathematical Programming with Data Perturbations*, Vol. 1, Marcel Dekker, New York (1983).
- [S16] H. Von Stackelberg, *The Theory of the Market Economy*, Oxford University Press, Oxford (1952).
- [S17] U.H. Suhl and L.M. Suhl, “Computing Sparse LU Factorizations for Large-Scale Linear Programming Bases,” *ORSA Journal on Computing*, Vol. 2, No. 4, pp. 325–335 (1994).
- [T1] T. Tanino and T. Ogawa, “An Algorithm for Solving Two-Level Convex Optimization Problems,” *Int'l J. of Systems Science*, Vol. 15, No. 2, pp. 163–174 (1984).
- [T2] B. Tolwinski, “Closed-Loop Stackelberg Solution to Multi-Stage Linear-Quadratic Game,” *J. of Optimiz. Theory & Appl.*, Vol. 34, No. 4, pp. 485–501 (1981).
- [T3] D.M. Topkis and A.F. Veinott, “On the Convergence of Some Feasible Direction Algorithms for Nonlinear Programming,” *SIAM J. of Control*, Vol. 5, pp. 268–279 (1967).
- [T4] Transportation Research Board, *Highway Capacity Manual*, Special Report 209, National Research Council, Washington, DC (1985).
- [T5] H. Tuy, A. Migdalas and P. Värbrand, “A Global Optimization Approach for the Linear Two-Level Program,” *Journal of Global Optimization*, Vol. 3, pp. 1–23 (1993).
- [V1] R.J. Vanderbei, *Linear Programming: Foundations and Extensions*, Kluwer Academic Publishers, Boston (1996).
- [V2] L.N. Vicente, G. Savard and J.J. Júdice, “Descent Approaches for Quadratic Bilevel Programming,” *J. of Optimiz. Theory & Appl.*, Vol. 81, No. 2, pp. 379–399 (1994).
- [V3] L.N. Vicente, G. Savard and J.J. Júdice, “The Discrete Linear Bilevel Programming Problem,” *J. of Optimiz. Theory & Appl.*, Vol. 89, No. 3, pp. 597–614 (1996).

- [V4] S.K. Vickery and R.E. Markland, "Multistage Lot Sizing in a Serial Production System," *International Journal of Production Research*, Vol. 24, No. 3, pp. 517–534 (1986).
- [W1] U.P. Wen and W.F. Bialas, "The Hybrid Algorithm for Solving the Three-Level Linear Programming Problem," *Computers & Operations Research*, Vol. 13, No. 4, pp. 367–377 (1986).
- [W2] U.P. Wen and Y.H. Yang, "Algorithms for Solving the Mixed Integer Two-Level Linear Programming Problem," *Computers & Operations Research*, Vol. 17, No. 2, pp. 133–142 (1990).
- [W3] R.E. Wendell and A.P. Hurter, Jr., "Minimization of a Non-separable Objective Function Subject to Disjoint Constraints," *Operations Research*, Vol. 24, No. 4, pp. 643–656 (1976).
- [W4] D.J. White and G. Anandalingam, "A Penalty Function for Solving Bi-Level Linear Programs," *Journal of Global Optimization*, Vol. 3, pp. 397–419 (1993).
- [W5] P. Wolfe, "Finding the Nearest Point in a Polytope," *Mathematical Programming*, Vol. 11, No. 2, pp. 128–149 (1976).
- [Y1] P.L. Yu, "Cone Convexity, Cone Extreme Points and Nondominated Solutions in Decision Problems with Multiple Objectives," *J. of Optimiz. Theory & Appl.*, Vol. 14, No. 3, pp. 319–377 (1974).

INDEX

Affine hull, 122
Affine independence, 122, 124
Algorithms
 0-1 BLPP, 259
 Bard and Moore (BM) —, 206
 Benders decomposition, 127, 130
 branch and bound, 91, 102, 213, 258
 comparisons, 190, 222, 287, 299, 359
 complementarity approach, 209
 conjugate gradient —, 178
 convex BLPP, 284, 290
 extreme point ranking, 50, 203
 genetic —, 363
 gradient projection method, 183
 grid search, 435
 implementation, 96, 204, 250, 259,
 286, 296, 440
 Kth-best algorithm, 203, 222
 Kuhn-Tucker approach, 204, 247,
 311
 linear BLPP, 202
 method of integer forms, 110
 mixed-integer linear BLPP, 247, 250
 nonlinear BLPP, 311, 320, 327, 339,
 352, 435–436
 nonlinear programming, 181
 penalty function approach, 218, 320
 penalty methods, 185
 primal all-integer cuts, 114
 rectangular partitioning, 327
 sequential LCP method, 210, 228,
 277
 sequential quadratic programming,
 188
 tabu search, 375
 variable elimination method, 213,
 216, 290

Zoutendijk's method, 182
Applications, 8
 advertising, 414
 bilevel programming, 391, 414, 428
 government regulation, 428
 network design, 391
Artificial intelligence, 362
Assignment problem, 134
Backtracking, 91, 96, 104, 107–108,
 207, 287
Benders decomposition, 127
BFGS method, 180, 190
Bilevel programming
 0-1 —, 233, 259
 applications, 8, 391, 414, 428
 branch and bound, 204, 213, 250,
 259, 311
 complementarity approach, 209
 convex —, 269, 311
 discrete —, 232
 discrete variables, 258
 general —, 301
 geometry, 197, 271
 irrelevant constraints, 305
 Kth-best algorithm, 203
 Kuhn-Tucker approach, 204, 247,
 312
 linear —, 195
 linear-quadratic —, 212
 mixed-integer —, 245, 425
 nonlinear —, 269, 301, 311, 320,
 327, 339, 428
 notation, 196
 NP-hard, 279
 optimality conditions, 344
 order of play, 12

- penalty function approach, 218, 229, 320
- quadratic —, 272, 291, 439
- random problem generation, 222, 254, 266
- relaxation(s), 245, 293, 315
- terminology, 7, 196
- theoretical properties, 198
- variable elimination method, 213
- Bimatrix game, 11
- Binary variables, 233
- Branch and bound, 87, 106, 206, 248, 258, 284, 290, 311, 356
 - backtracking, 108, 207
 - bookkeeping, 207, 250, 286, 313
 - bounding theorems, 249
 - branching rules, 217, 225, 251, 296
 - breadth-first search, 97
 - data structures, 106, 206, 313
 - depth-first search, 96
 - fathoming rules, 212, 245–246, 285, 314
 - separation schemes, 291
- Breadth-first search, 97, 313
- Bundle method, 339, 440
- Complementarity conditions, 273
- Complementary pivoting heuristic, 209, 259
 - Complementary pivoting, 272, 274
 - Complementary slackness, 48, 66, 72, 204, 219, 312, 327
 - theorem, 64, 219–220
- Concave programming, 352
- Conjugate direction method, 177
- Conjugate gradient method, 325, 352
- Convergence
 - penalty methods, 187
 - vector —, 146
- Convex bilevel programming, 269
- Convex combination, 121
- Convex functions, 168
- Convex hull, 86, 121
- Convex programming, 167
 - definition, 140
- Convexity
 - definition, 152
- CPLEX, 214, 292
- Cutting planes, 83, 85, 109
 - all-integer cuts, 114
 - unit coefficients, 118
 - valid inequalities, 121
- Data structures, 106, 206, 313
- Decision making, 3
- Definitions
 - affine independence, 122
 - continuous functions, 147
 - convex polyhedron, 26
 - convexity, 152
 - differentiable functions, 147
 - directional derivative, 148
 - extreme homogeneous solution, 45
 - fathoming, 89
 - global minimum, 155
 - gradient, 147
 - Hessian, 147
 - homogeneous solution, 44
 - inducible region, 196
 - Jacobian, 148
 - linear independence, 121
 - local minimum, 155
 - order of convergence, 154
 - polyhedron, 123
 - positive (semi)definite matrix, 144
 - rate of convergence, 152
 - rational reaction set, 196
 - regular point, 159, 164, 345
 - stationary point, 149
 - totally unimodular, 133
 - unimodularity, 133
- Depth-first search, 96, 313
- Directional derivative, 148, 334
- Discrete variables, 232
- Double penalty function method, 320
- Duality
 - dual simplex method, 72
 - economic interpretation, 66
 - fundamental theorem, 63, 219
 - gap, 218, 378

- primal–dual pair, 59, 61
shadow prices, 67
theorems, 61
weak duality theorem, 62
Eigenvalues, 144, 177
Eigenvectors, 144
Enumerative methods
 see branch and bound, 87
Euclidean norm, 146
Existence of solutions, 11, 303
Extreme point, 25, 49
Extreme-value function, 7
Fathoming, 88
Fibonacci search, 172
Fixed-charge problem, 81
Game theory, 6
GAMS, 434, 440, 443
Gauss-Jordan elimination, 31
Generalized reduced gradient (GRG)
 method, 185
Genetic algorithms, 363
Global minimum
 definition, 155
Goal programming, 4
Golden section, 170
Gradient descent, 332, 339
Gradient projection method, 183
Gradient, 147
Hessian, 147, 156, 165
 — of the Lagrangian, 163, 166, 181
 quasi-Newton methods, 179
Heuristics, 361
 comparisons, 370, 374
 genetic algorithms, 363
 grid search, 369
 simulated annealing, 373
 tabu search, 375
Hierarchical decentralized systems, 7
Hierarchical optimization, 3
Implicit enumeration, 83, 91, 102, 258
Implicit function theorem, 150
Inducible region, 196, 199, 271, 301,
 303, 312
Integer programming, 76
method of integer forms, 110
monotone variables, 100
penalties, 98
primal all-integer cuts, 114
special ordered sets, 101
zero-one —, 102
Integer variables, 232
Inventory model, 414
Irrelevant constraints, 305
Jacobian, 148, 340, 343, 439
 K th-best algorithm, 203
Kuhn-Tucker approach, 247, 312, 423
 irrelevant constraints, 312
Kuhn-Tucker conditions, 164, 277
 linear programming, 57
Kuhn-Tucker formulation, 218, 270
Lagrangian, 57, 161, 163, 166, 189,
 334, 342
Limit inferior, 146
Limit point, 146
Limit superior, 146
Linear complementarity problem
 (LCP), 57, 209
Linear complementarity, 278
Linear independence, 121
Linear programming, 17
 alternative optima, 48
 artificial variables, 51
 basic feasible solution, 32
 basic solution, 22, 30
 basis, 22
 bounded variables, 54
 boundedness, 29
 convex properties, 25
 cycling, 37
 degeneracy, 22, 37
 dual feasible basis, 73
 duality theorems, 61
 duality, 59
 fundamental theorem, 23, 51
 geometry, 41
 homogeneous solution, 44–45
 inexact —, 414
 interior point methods, 17

- leaving variable, 32, 56
- minimum ratio test, 33, 39, 44, 56
- phase 1, 51
- phase 2, 53
- pivoting, 30
- pricing out, 35
- reduced cost coefficients, 33, 48
- relative cost coefficients, 33
- sensitivity analysis, 66
- simplex method, 21, 29
- slack and surplus variables, 58
- stalling, 40
- standard form, 18
- tableau, 30
- unboundedness, 44
- Lipschitz function, 330, 344
- Local minimum
 - definition, 155
- Matrix
 - positive (semi)definite, 144, 168, 177, 272
 - projection —, 185
 - rank, 121
 - totally unimodular, 133
 - unimodular, 133
- Max-min problem, 8, 198
- Mean value theorem, 150, 156
- Method of false position, 174
- Method of integer forms, 110
- Monotone variables, 100
- Multidimensional search techniques, 175
- Multiple objective programming, 7, 311
- Network design, 391
 - accident costs, 402
 - BLP model, 396
 - decision variables, 394
 - flow constraints, 407
 - improvement costs, 403
 - maintenance costs, 403
 - objective function, 397
 - operating costs, 399
 - travel time functions, 397
- Newton's method, 173, 176
- Nonlinear programming, 137
 - algorithms, 181
 - equality constraints, 159
 - inequality constraints, 164
 - multidimensional search techniques, 175
 - nonnegative variables, 157
 - one-dimensional search, 170
 - optimality conditions, 155
 - search techniques, 169
 - unconstrained problems, 156
- Notation
 - bilevel programming, 196
 - branch and bound, 248, 259
 - nonlinear programming, 143
- NP-hard, 198, 272, 279
- One-dimensional search, 170
- Opportunity cost, 68
- Optimal-value function, 8, 303
- Optimality conditions (also see Kuhn-Tucker conditions), 57
- Optimality conditions
 - equality constraints, 161
 - inequality constraints, 164
 - necessary —, 333
 - nonlinear programming, 155
 - nonnegative variables, 158
 - NP-hard, 279
 - unconstrained problems, 156
- Order of convergence
 - definition, 154
- OSL, 214, 434, 440
- Outer approximation method, 356
- Pareto optimality, 197, 266, 304, 424
- Penalties, 98, 215, 226
- Penalty approach, 229
- Penalty function, 233, 237, 240, 320, 376
- Penalty methods, 185, 218
 - convergence, 187
- Penalty parameter, 238, 244
- Piecewise linear function, 395
- Polyhedral theory, 121

- Polyhedron, 123
 - dimension, 123
- Production planning, 414
- Projection matrix, 185
- Quadratic objective, 201, 270, 318
- Quadratic programming, 439
 - definition, 139
- Quadratic programs, 273
- Quasi-Newton methods, 179
- Quasiconvex functions, 170
- Rate of convergence
 - definition, 152
- Rational reaction set, 196, 218, 301, 303
- Rectangular partitioning, 327
- Regular point, 345
 - definition, 159, 164
- Search techniques
 - conjugate direction method, 177
 - Fibonacci, 172
 - golden section, 170
 - method of false position, 174
 - multidimensional —, 175
 - Newton's method, 173, 176
 - one-dimensional —, 170
 - quasi-Newton methods, 179
 - steepest descent, 176
 - unconstrained problems, 169
- Sensitivity analysis
 - cost coefficients, 68
 - linear programming, 66
 - matrix coefficients, 72
 - ranging, 68
 - right-hand-side coefficients, 70
- Sequential LCP method, 210, 228
- Sequential linear programming, 188
- Sequential quadratic programming, 188
- Sets
 - closed, 146
 - compact, 146
 - open, 146
- Simplex method, 29
- Simulated annealing, 373
- SLCP algorithm, 210, 228
- SOS constraints, 80, 101
- Special ordered sets, 80, 101
- Stackelberg game, 3, 301, 352
- Stalling, 40, 261
- Stationary point, 175
- Steepest descent, 176, 276, 332
- Stepsize, 279
- Subgradients, 339, 353, 440
- Successive quadratic programming, 317
- Tabu search, 375
- Tangent plane, 160
- Taylor series expansion, 150, 188
- Terminology, 7
- Three-level programming, 237, 240–241, 245
- Totally unimodular, 133
- Transportation network, 391
- Unimodal functions, 170
- Unimodularity, 77, 133
- Valid inequalities, 121
- Variable elimination algorithm, 216
- Vectors
 - conjugate, 177
 - Q-orthogonal, 177
- Vertex enumeration, 17
 - adjacency, 42, 47–48
- Weierstrass theorem, 21, 149
- Zero-one variables, 102, 233
- Zigzagging, 176, 182, 185
- Zoutendijk's method, 182

Nonconvex Optimization and Its Applications

1. D.-Z. Du and J. Sun (eds.): *Advances in Optimization and Approximation*. 1994.
ISBN 0-7923-2785-3
2. R. Horst and P.M. Pardalos (eds.): *Handbook of Global Optimization*. 1995
ISBN 0-7923-3120-6
3. R. Horst, P.M. Pardalos and N.V. Thoai: *Introduction to Global Optimization* 1995
ISBN 0-7923-3556-2; Pb 0-7923-3557-0
4. D.-Z. Du and P.M. Pardalos (eds.): *Minimax and Applications*. 1995
ISBN 0-7923-3615-1
5. P.M. Pardalos, Y. Siskos and C. Zopounidis (eds.): *Advances in Multicriteria Analysis*. 1995
ISBN 0-7923-3671-2
6. J.D. Pintér: *Global Optimization in Action*. Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications. 1996
ISBN 0-7923-3757-3
7. C.A. Floudas and P.M. Pardalos (eds.): *State of the Art in Global Optimization*. Computational Methods and Applications. 1996
ISBN 0-7923-3838-3
8. J.L. Higle and S. Sen: *Stochastic Decomposition*. A Statistical Method for Large Scale Stochastic Linear Programming. 1996
ISBN 0-7923-3840-5
9. I.E. Grossmann (ed.): *Global Optimization in Engineering Design*. 1996
ISBN 0-7923-3881-2
10. V.F. Dem'yanov, G.E. Stavroulakis, L.N. Polyakova and P.D. Panagiotopoulos: *Quasidifferentiability and Nonsmooth Modelling in Mechanics, Engineering and Economics*. 1996
ISBN 0-7923-4093-0
11. B. Mirkin: *Mathematical Classification and Clustering*. 1996
ISBN 0-7923-4159-7
12. B. Roy: *Multicriteria Methodology for Decision Aiding*. 1996
ISBN 0-7923-4166-X
13. R.B. Kearfott: *Rigorous Global Search: Continuous Problems*. 1996
ISBN 0-7923-4238-0
14. P. Kouvelis and G. Yu: *Robust Discrete Optimization and Its Applications*. 1997
ISBN 0-7923-4291-7
15. H. Konno, P.T. Thach and H. Tuy: *Optimization on Low Rank Nonconvex Structures*. 1997
ISBN 0-7923-4308-5
16. M. Hajdu: *Network Scheduling Techniques for Construction Project Management*. 1997
ISBN 0-7923-4309-3
17. J. Mockus, W. Eddy, A. Mockus, L. Mockus and G. Reklaitis: *Bayesian Heuristic Approach to Discrete and Global Optimization*. Algorithms, Visualization, Software, and Applications. 1997
ISBN 0-7923-4327-1
18. I.M. Bomze, T. Csendes, R. Horst and P.M. Pardalos (eds.): *Developments in Global Optimization*. 1997
ISBN 0-7923-4351-4
19. T. Rapcsák: Smooth Nonlinear Optimization in R^n . 1997
ISBN 0-7923-4680-7
20. A. Migdalas, P.M. Pardalos and P. Värbrand (eds.): *Multilevel Optimization: Algorithms and Applications*. 1998
ISBN 0-7923-4693-9
21. E.S. Mistakidis and G.E. Stavroulakis: *Nonconvex Optimization in Mechanics*. Algorithms, Heuristics and Engineering Applications by the F.E.M. 1998
ISBN 0-7923-4812-5

Nonconvex Optimization and Its Applications

- 22. H. Tuy: *Convex Analysis and Global Optimization*. 1998 ISBN 0-7923-4818-4
- 23. D. Cieslik: *Steiner Minimal Trees*. 1998 ISBN 0-7923-4983-0
- 24. N.Z. Shor: *Nondifferentiable Optimization and Polynomial Problems*. 1998 ISBN 0-7923-4997-0
- 25. R. Reemtsen and J.-J. Rückmann (eds.): *Semi-Infinite Programming*. 1998 ISBN 0-7923-5054-5
- 26. B. Ricceri and S. Simons (eds.): *Minimax Theory and Applications*. 1998 ISBN 0-7923-5064-2
- 27. J.-P. Crouzeix, J.-E. Martinez-Legaz and M. Volle (eds.): *Generalized Convexity, Generalized Monotonicity: Recent Results*. 1998 ISBN 0-7923-5088-X
- 28. J. Outrata, M. Kočvara and J. Zowe: *Nonsmooth Approach to Optimization Problems with Equilibrium Constraints*. 1998 ISBN 0-7923-5170-3
- 29. M.A. Ridley: *Lowering the Cost of Emission Reduction: Joint Implementation in the Framework Convention on Climatic Change*. 1998 ISBN 0-7923-4914-8
- 30. J.F. Bard: *Practical Bilevel Optimization. Algorithms and Applications*. 1999 ISBN 0-7923-5458-3