# TUTORIALS IN OPERATIONS RESEARCH

**2005**

# Tutorials in Operations Research

## Emerging Theory, Methods, and Applications

Harvey J. Greenberg, Series Editor

J. Cole Smith, Tutorials Chair and Volume Editor

Presented at the INFORMS Annual Meeting, November 13–16, 2005

informs® *Institute for Operations Research and the Management Sciences*

**www.informs.org**

Copyright ©2005 by the Institute for Operations Research and the Management Sciences (INFORMS).

ISBN 1-877640-21-2

To order this book, contact:

INFORMS
7240 Parkway Drive, Suite 310
Hanover, MD 21076 USA
Phone: (800) 4-INFORMS or (443) 757-3500
Fax: (443) 757-3515
E-mail: informs@informs.org
URL: www.informs.org

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS
ISBN 1-877640-21-2

# Table of Contents

Foreword iv
Preface v
Acknowledgments viii

Chapter 1
Branch and Tree Decomposition Techniques for Discrete Optimization 1
*Illya V. Hicks, Arie M. C. A. Koster, and Elif Kolotoğlu*

Chapter 2
Stochastic Programming: Optimization When Uncertainty Matters 30
*Julia L. Higle*

Chapter 3
Network Models in Railroad Planning and Scheduling 54
*Ravindra K. Ahuja, Claudio B. Cunha, and Güvenç Şahin*

Chapter 4
Analyzing the Vulnerability of Critical Infrastructure to Attack and Planning Defenses 102
*Gerald G. Brown, W. Matthew Carlyle, Javier Salmerón, and Kevin Wood*

Chapter 5
Demand Selection and Assignment Problems in Supply Chain Planning 124
*Joseph Geunes, Yasemin Merzifonluoğlu, H. Edwin Romeijn, and Kevin Taaffe*

Chapter 6
An Introduction to Revenue Management 142
*Garrett J. van Ryzin and Kalyan T. Talluri*

Chapter 7
Decision Analysis = Decision Engineering 195
*James E. Matheson*

Chapter 8
Operations Research in Experimental Psychology 213
*J. Neil Bearden and Amnon Rapoport*

Chapter 9
Active Learning for Quantitative Courses 237
*James J. Cochran*

Chapter 10
CBC User Guide 257
*John Forrest and Robin Lougee-Heimer*

Contributing Authors 278

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS
ISBN 1-877640-21-2

# Foreword

This is the inaugural volume of the *Tutorials in Operations Research*, a book series of INFORMS, which I founded with much help and support from Frederic H. Murphy, Vice President of Publications. Building on the tutorials book I edited from the 2004 INFORMS meeting in Denver (published by Springer), we made this an annual series published by INFORMS. J. Cole Smith is our first Volume Editor, serving also as the Tutorials Chair for the 2005 INFORMS meeting. In forming policies and procedures, I had help from my Advisory Board: Erhan Erkut, J. Cole Smith, and David L. Woodruff.

Cole has done a great job as editor, recruiting a diverse set of tutorials for the INFORMS meeting and publishing some of those here. Each author is to be congratulated, and Cole has my gratitude for putting forth the extra work to produce this volume.

Having worked in academia, government, and industry (large and small), I have found that operations research is widely used but not so widely recognized. This "identity crisis" needs our attention because we, in the OR/MS profession, bring highly developed problem-solving skills to any table. Our strengths are modeling and analysis with concomitant strength in computation, both numerical and symbolic, including visualization. Our roots are in team efforts, and OR has been definitionally multidisciplinary. Our promise is to solve problems, and we draw upon many areas of mathematics, computer science, and economics to do our job well. In addition, we learn what is necessary for an application at hand, be it in production, finance, engineering, or science. OR is the exemplar of technology transfer.

In this volume, Cole has chosen tutorials that will help others learn areas of OR. His preface puts them into perspective, but it is worth emphasizing them. Note the mix of application-driven ("Network Models in Railroad Planning and Scheduling" and "Demand Selection and Assignment Problems in Supply Chain Planning") and method-driven ("Branch and Tree Decomposition Techniques for Discrete Optimization" and "Stochastic Programming: Optimization When Uncertainty Matters"), of traditional ("An Introduction to Revenue Management") and new ("Operations Research in Experimental Psychology"), of conceptual ("Decision Analysis = Decision Engineering") and computational ("CBC User Guide"), of classroom education ("Active Learning for Quantitative Courses") and field education ("Analyzing the Vulnerability of Critical Infrastructure to Attack and Planning Defenses"). I expect this volume to be very popular indeed.

Again, I thank Cole for all of his hard work and initiative and Fred Murphy for having the vision to produce this series.

Harvey J. Greenberg
University of Colorado at Denver
Denver, Colorado

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS
ISBN 1-877640-21-2

# Preface

One of the primary goals of the 2005 INFORMS meeting tutorials is to address the evolving challenges faced by our community, especially with respect to the visibility and impact of our profession. Indeed, communicating the role of operations research and management science (OR/MS) to industrial and governmental organizations often remains quite difficult, even to those organizations that can benefit the most from OR/MS. After all, the vast majority of people who navigate to and from work, plan their weekly budget, and decide when and at what price to buy airplane tickets never know that they are (heuristically) solving complex OR/MS problems. While the loss of five minutes on the way to work or paying 5% over the cheapest-possible airfare is of little consequence, it is well-documented that decision processes worth millions of dollars, or even those determining the difference between life and death, can unquestionably benefit from OR/MS principles.

INFORMS tutorials sessions provide an opportunity to witness the application of OR/MS to new problem domains, investigate new theoretical investigations from first principles, understand the foundations and procedures of new methods, and participate in the ongoing implementation of our findings in the classroom and in practice. However, they currently lack a certain outreach ability: Those who are not familiar with OR/MS will likely not attend the annual meeting in the first place, and even those present must budget their resources between subdivision meetings, parallel sessions, and other service responsibilities. Moreover, those who are in fact able to attend the tutorials sessions often do not have a written artifact of the topic to which they can refer after the conference. It is therefore the goal of this book to remedy these shortcomings and widen the visibility of new research and applications in our field. Last year's Tutorial Chair, Dr. Harvey Greenberg, compiled a volume consisting of eight tutorial chapters that complement the tutorial sessions presented at the 2004 INFORMS meeting in Denver. He has continued his efforts by establishing a series of tutorial compilations to be published by INFORMS, of which this book represents the first volume.

Illya Hicks, Arie Koster, and Elif Kolotoğlu begin in Chapter 1 by examining the concepts of branch and tree decomposition techniques in the context of combinatorial optimization. Although branch and tree decompositions are not new concepts, their usefulness in the realm of optimization is something that has only recently blossomed in the literature. While a firm background in graph theory concepts is useful to appreciate the details of their work, the reward is an exciting new approach to solving certain discrete optimization problems from an implicit enumeration perspective that is fundamentally different from most ad-hoc algorithms designed for hard optimization problems. The success of researchers that apply branch and tree decomposition techniques to combinatorial optimization problems (including the traveling salesman problem, synchronous optical network design problems, and frequency assignment problems) are truly eye-opening.

Julia Higle presents an introduction to stochastic programming in Chapter 2. Stochastic programming is rapidly becoming a hot area of research as the necessity of incorporating uncertainty into optimization models, rather than relying on sample means as deterministic input, has become evident. Higle demonstrates this concept with a simple example, illustrating the shortcomings of utilizing mean estimates and sensitivity analysis in optimization problems. One can then begin to imagine the myriad applications that can benefit from stochastic programming problems, such as electricity markets, telecommunication network design, and air transportation problems, where demand and system operability are

rarely known with certainty. This discussion recaps theory and solution techniques for basic two-stage problems, in which a decision must be made at the present time, followed by the realization of some stochastic event (e.g., actual demands or component failures), after which a set of *recourse decisions* must be made. For those readers already familiar with the stochastic programming, the closing sections offer new insights into stochastic decomposition methods for uniting statistical methods with classical decomposition techniques and multistage optimization models that challenge the limits of modern computing ability.

Multistage optimization is not limited to stochastic programming scenarios. While perhaps the best-known example of multistage optimization lies in air transportation, another less-researched area lies in railroad optimization, as discussed in Chapter 3. These problems involve a unique set of challenges with very significant costs. Ravindra K. Ahuja, Claudio Cunha, and Güvenç Sahin provide six classes of rail optimization problems, each of which is a large-scale NP-hard optimization problem. The authors provide specific details regarding the sizes of these problems, and they discuss the actual financial benefit of employing their methods. In addition to its interesting application of mathematical modeling and very large-scale neighborhood-based heuristics, this chapter is an exemplar of the importance of OR/MS in contemporary applications and the need for OR/MS professionals to seek out such applications.

Chapter 4 continues the network optimization theme, but in the context of modern security challenges that are faced in designing critical network infrastructures. Gerald Brown, Matthew Carlyle, Javier Salmerón, and Kevin Wood provide a comprehensive review of network interdiction problems and contemporary methods for their solution. The problems contained within this chapter are for the most part Stackelberg games, wherein an enemy may decide to disrupt a portion of the network after which the protagonist must make some recourse decision to mitigate the attack. The authors make a convincing case by repeated examples and illustrations that enough data exists in the public domain to justify the assumption that enemies are capable of acting with full knowledge of a network's capabilities. The authors discuss cases ranging from securing subway systems to airport security to electric grids in order to demonstrate the usefulness of their models and algorithms.

Critical networks and business processes can often be disrupted accidentally due to randomness in demands. In particular, the effectiveness of many supply chain applications depends very strongly on the quality of the forecast demands. In Chapter 5, Joseph Geunes, Yasemin Merzifonluoğlu, H. Edwin Romeijn, and Kevin Taaffe consider the impact of demand uncertainty for the case in which a supplier must select certain groups of demands (e.g., products) that they will try to satisfy. This chapter incorporates key concepts in discrete optimization and revenue management with traditional supply chain optimization research, and demonstrates another vital class of problems in which multiple stages of decisions must be addressed in an integrated fashion.

Revenue management is examined in Chapter 6 in more detail by Garrett van Ryzin and Kalyan T. Talluri. The depth and breadth of the field of revenue management is due in part to its successful application in a number of different industries, and to its importance as a subcomponent of other research fields (such as the study described in §5). This chapter examines methods for controlling capacity on a single resource (such as airline tickets) by various types of control strategies, such as levying booking limits or reserving capacity for classes of consumers. Based on these fundamentals, the authors then develop some of the basic theory and applications for revenue management in a self-contained manner.

While the first part of this book has focused almost entirely on quantitative methods for decision making, a large body of research exists in the field of decision analysis that merges quantitative research with decision making in practical settings. James Matheson makes this connection in Chapter 7 with an entertaining discussion on engineering the decision-making process. This chapter is not only valuable to the growing body of Decision Analysis Society members within INFORMS, but also to OR/MS researchers whose models include human

decision-making entities. Some of the most resonant contributions of Matheson's work are his description the decision hierarchy, accompanied by examples and illustrations of this structure, and a summary set of 10 commandments for decision analysis.

J. Neil Bearden and Amnon Rapoport approach human decision making in Chapter 8 from the rare perspective of behavioral psychologists with an expertise in optimization. They investigate several optimal stopping and triage-assignment problems, which are certainly interesting optimization problems in their own right. In addition to providing optimal solution techniques for these problems, Bearden and Rapoport have conducted a long series of behavioral experiments to judge how humans actually make decisions, and what sort of suboptimal behaviors humans exhibit in these problems. This study is of primary importance to the OR/MS community because while systems are often designed with the "rational" decision maker in mind, the findings presented in Chapter 8 demonstrate that humans tend to exhibit common suboptimal traits that should be anticipated in the design of human-in-the-loop systems. This chapter thus affords unique insights and opportunities to conduct new and more realistic game theory studies by better understanding specific characteristics of suboptimal human behavior.

The last two chapters of this volume discuss the implementation of these new ideas and classical concepts in the OR/MS classroom and in open-source computing environments. In Chapter 9, James Cochran provides a detailed discussion of active learning methods that are proven to enhance the understanding and retention of OR/MS material in the classroom. This chapter provides a unique contribution to the INFORMS audience for several reasons. One is that Cochran's message is not only valuable to academics within INFORMS, but to anyone in a leadership position responsible for communicating OR/MS principles. A second important and unique aspect of this chapter is that the discussion is geared toward the quantitative classroom, unlike many general-purpose teaching tips that do not necessarily translate to engineering classrooms. Finally, this chapter provides concrete examples of how active learning can be injected into the classroom, along with pointers to free software available to ease the transition.

The final chapter examines one facet of the Computational Infrastructure for Operations Research (COIN-OR) project, which is a collection of open-source optimization libraries that become stable and effective over time through repeated use and improvement by the optimization community. Chapter 10 contains a tutorial written by John Forrest and Robin Lougee-Heimer on how to use the COIN branch-and-cut (CBC) solver. (This chapter is perhaps a tutorial in the most literal sense, in that it provides specific source code on how to implement these algorithms.) While many OR/MS researchers have a vital interest in CBC in particular, this chapter also serves as a gateway to the many other libraries in the COIN-OR project.

These chapters accomplish more than recapping classical OR/MS methods; they provide an accessible discussion of evolving issues in theory, methodology, applications, and implementation. While each of these topics is very obviously distinct in their respective problem domains, they share the common thread that they are all of substantial importance, are new and emerging topics (some truly in their infancy), and are open to new ideas from the INFORMS community to further their development and impact.

J. Cole Smith
University of Arizona
Tucson, Arizona

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS
ISBN 1-877640-21-2

# Acknowledgments

First, I would like to express my sincere appreciation to each of the authors. The deadlines for completing the chapters of this book fell during a very busy period for virtually every contributor, and each of them made a substantial sacrifice of their already-full schedules in order to produce high-quality chapters in time to produce this book for the INFORMS 2005 meeting. Equally vital to this book were the support and selfless assistance of Harvey Greenberg, the 2004 Tutorials Chair and 2005 Series Editor, who provided me with invaluable direction (and reassurance) for almost an entire year while I organized the tasks required to publish this book. The tutorials book series is clearly the result of his tireless work and enthusiasm. I am also grateful to Jim Cochran for his efforts in originating the INFORMS 2005 meeting and for his continued support in producing this book series. Finally, the publications staff at INFORMS remained remarkably patient and helpful at every turn during the production of this book.

J. Cole Smith
University of Arizona
Tucson, Arizona

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS | ISBN 1-877640-21-2
DOI 10.1287/educ.1053.0017

# Branch and Tree Decomposition Techniques for Discrete Optimization

***Illya V. Hicks***
Department of Industrial Engineering, Texas A & M University, College Station, Texas 77843-3131, ivhicks@tamu.edu

***Arie M. C. A. Koster***
Zuse Institute Berlin (ZIB), Takustraße 7, D-14195 Berlin, Germany, koster@zib.de

***Elif Kolotoğlu***
Department of Industrial Engineering, Texas A & M University, College Station, Texas 77843-3131, elif@tamu.edu

**Abstract** This chapter gives a general overview of two emerging techniques for discrete optimization that have footholds in mathematics, computer science, and operations research: branch decompositions and tree decompositions. Branch decompositions and tree decompositions, along with their respective connectivity invariants, branchwidth and treewidth, were first introduced to aid in proving the graph minors theorem, a well-known conjecture (Wagner's conjecture [103]) in graph theory. The algorithmic importance of branch decompositions and tree decompositions for solving $\mathcal{NP}$-hard problems modeled on graphs was first realized by computer scientists in relation to formulating graph problems in monadic second-order logic. The dynamic programming techniques utilizing branch decompositions and tree decompositions, called branch decomposition- and tree decomposition-based algorithms, fall into a class of algorithms known as fixed-parameter tractable algorithms and have been shown to be effective in a practical setting for $\mathcal{NP}$-hard problems such as minimum domination, the traveling salesman problem, general minor containment, and frequency assignment problems.

**Keywords** branchwidth; treewidth; graph algorithms; combinatorial optimization

## 1. Introduction

The notions of branch decompositions and tree decompositions and their respective connectivity invariants, branchwidth and treewidth, are two emerging techniques for discrete optimization that also encompass the fields of graph theory, computer science, and operations research. The origins of branchwidth and treewidth are deeply rooted in the proof of the graph minors theorem, formally known as Wagner's conjecture [103]. Briefly, the *graph minors theorem* states that in an infinite list of graphs there would exist two graphs $H$ and $G$ such that $H$ is a minor of $G$. The algorithmic importance of the branch decomposition and tree decomposition was not realized until Courcelle [50] and Arnborg et al. [14] showed that several $\mathcal{NP}$-hard problems posed in monadic second-order logic can be solved in polynomial time using dynamic programming techniques on input graphs with bounded treewidth or branchwidth. A problem that is $\mathcal{NP}$-hard implies that as long as it is not proven that $\mathcal{P} = \mathcal{NP}$, we cannot expect to have a polynomial-time algorithm for the problem. These techniques are referred to as tree decomposition-based algorithms and branch decomposition-based algorithms, respectively. Branch decomposition- and tree decomposition-based algorithms are important in discrete optimization because they have

been shown to be effective for combinatorial optimization problems like the ring-routing problem (Cook and Seymour [47]), the traveling salesman problem (Cook and Seymour [48]), frequency assignment (Koster et al. [87]), general minor containment (Hicks [72]), and the optimal branch decomposition problem (Hicks [73]).

The procedure to solve an optimization problem with bounded branchwidth or treewidth involves two steps: (i) computation of a (good) branch/tree decomposition, and (ii) application of an algorithm that solves instances of bounded branchwidth/treewidth in polynomial time. Because the branchwidth or treewidth is considered to be a constant, not part of the input, this value may occur in the exponent of the complexity of both running time and space requirements. Hence, it is important to have a decomposition of width as small as possible. The problem of minimizing this quantity is, however, $\mathcal{NP}$-hard in itself.

Note that not every combinatorial problem defined on a graph of bounded branchwidth or treewidth can be solved in polynomial time. An example is the bandwidth minimization problem, which is $\mathcal{NP}$-hard even on ternary trees (every vertex has degree one or three) (Garey et al. [58] and Monien [95]). Even if the problem is polynomial on trees, the problem need not be polynomial on graphs of bounded treewidth: $L(2,1)$-coloring is $\mathcal{NP}$-complete for graphs with treewidth 2 (Fiala et al. [53]). For more information on $L(2,1)$-colorings, one is referred to the work of Chang and Kuo [42] and the work of Bodlaender and Fomin [30].

Besides using the theory of monadic second-order logic, whether or not the problem can be solved in polynomial time on graphs of bounded branchwidth or treewidth can be discovered by investigating characteristics of the solution. Given a vertex cut set, one has to answer the question of what impact the solution on one side of the cut set has on the solution on the other side. If the solutions only depend on the solution in the vertex cut, the problem likely can be solved with a dynamic programming algorithm specialized for the problem.

This chapter gives a general overview of branchwidth and treewidth along with their connections to structural graph theory, computer science, and operations research. Section 2 offers preliminary and relevant definitions in the subject area. Section 3 offers some interesting background on the graph minors theorem and its relation to branchwidth and treewidth. Section 4 describes algorithms to construct branch decompositions as well as a blueprint for branch decomposition-based algorithms. Section 5 offers similar results for treewidth with the addition of algorithms for computing relevant lower bounds to treewidth. Section 6 describes the extension of branchwidth and treewidth to matroids, and Section 7 describes relevant open problems in the area. It is our hope that this chapter will spark interest in this fascinating area of research.

## 2. Definitions

### 2.1. Graph Definitions

In this section we give basic definitions. The reader may skip this section and refer to it when necessary.

A *graph* is an ordered pair $(V, E)$ where $V$ is a nonempty set, called the set of *vertices* or *nodes*; $E$, the set of *edges*, is an unordered binary relation on $V$. A graph is called *complete* if all possible edges between the nodes of the graph are present in the graph. A *hypergraph* is an ordered pair $(V, E)$ of nodes and edges, and an incidence relationship between them that is not restricted to two ends for each edge. Thus, edges of hypergraphs, also called *hyperedges*, can have any number of ends.

A graph $\bar{G} = (\bar{V}, \bar{E})$ is a *subgraph* of the graph $G = (V, E)$ if $\bar{V} \subseteq V$ and $\bar{E} \subseteq E$. For a subset $V' \subseteq V$, $G[V']$ denotes the graph induced by $V'$, i.e., $G[V'] = (V', E \cap (V' \times V'))$. For a subset $E' \subseteq E$, the graph induced by these edges is denoted by $G[E']$. *Contraction* of an edge $e$ means deleting that edge and identifying the ends of $e$ into one node. Parallel edges are identified as well. A graph $H$ is a *minor* of a graph $G$ if $H$ can be obtained from a subgraph of $G$ by a series of contractions. A *subdivision* of a graph $G$ is a graph obtained from $G$ by replacing its edges by internally vertex disjoint paths.

The *degree* of a vertex is the number of edges incident with that vertex. A graph is *connected* if every pair of vertices can be joined by a path. The *connectivity* of a graph is the smallest number of vertices that can be removed to disconnect the graph. A graph that does not contain any cycles (*acyclic*) is called a *forest*. A connected forest is called a *tree*. The *leaves* of a tree are the vertices of degree 1.

A graph $G = (V, E)$ is *bipartite* if $V$ admits a partition into two classes such that every edge has its ends in different classes: Vertices in the same partition class must not be adjacent. A bipartite graph is *complete* if all possible edges between the nodes of the graph, while maintaining the restriction of the bipartition, are present in the graph. A graph is *planar* if it can be embedded in a plane such that no two edges cross. The *incidence graph* $I(G)$ of a hypergraph $G$ is the simple bipartite graph with vertex set $V(G) \cup E(G)$ such that $v \in V(G)$ is adjacent to $e \in E(G)$ if and only if $v$ is an end of $e$ in $G$. Seymour and Thomas [116] define a hypergraph $H$ as *planar* if and only if $I(H)$ is planar. Also, a hypergraph $G$ is called *connected* if $I(G)$ is connected. For an edge $e$, $\eta(e)$ is the number of nodes incident with $e$. The largest value $\eta(e)$ over all $e \in E$ is denoted by $\eta(G)$.

## 2.2. Branch Decompositions

Let $G = (V, E)$ be a hypergraph and $T$ be a ternary tree (a tree where every nonleaf node has degree 3) with $|E(G)|$ leaves. Let $\nu$ be a bijection (one-to-one and onto function) from the edges of $G$ to the leaves of $T$. Then, the pair $(T, \nu)$ is called a *branch decomposition* of $G$ (Robertson and Seymour [106]).

A *partial branch decomposition* is a branch decomposition without the restriction of every nonleaf node having degree 3. A *separation* of a graph $G$ is a pair $(G_1, G_2)$ of subgraphs with $G_1 \cup G_2 = G$ and $E(G_1 \cap G_2) = \varnothing$, and the *order* of this separation is defined as $|V(G_1 \cap G_2)|$. Let $(T, \nu)$ be a branch decomposition. Then, removing an edge, say $e$, from $T$ partitions the edges of $G$ into two subsets $A_e$ and $B_e$. The *middle set* of $e$, denoted $\text{mid}(e)$, is the set of vertices of G that are incident to the edges in $A_e$ and the edges in $B_e$, and the *width* of an edge $e$, denoted $|\text{mid}(e)|$, is the order of the separation $(G[A_e], G[B_e])$. The *width* of a branch decomposition $(T, \nu)$ is the maximum width among all edges of the decomposition. The *branchwidth* of $G$, denoted by $\beta(G)$, is the minimum width over all branch decompositions of $G$. A branch decomposition of $G$ with width equal to the branchwidth is an *optimal branch decomposition* of $G$. Figure 2 illustrates an optimal branch decomposition of the graph given in Figure 1.

Robertson and Seymour [106] characterized the graphs that have branchwidth $\leq 2$ and showed that $(n \times n)$-grid graphs have branchwidth $n$. Other known classes of graphs with known branchwidth are cliques whose branchwidth is $\lceil (2/3)|V(G)| \rceil$. For chordal graphs, the branchwidth of this class of graphs is characterized by $\lceil (2/3)\omega(G) \rceil \leq \beta(G) \leq \omega(G)$ where $\omega(G)$ is the maximum clique number of $G$ (Hicks [70] and Robertson and Seymour [106]).

FIGURE 1. Example graph.

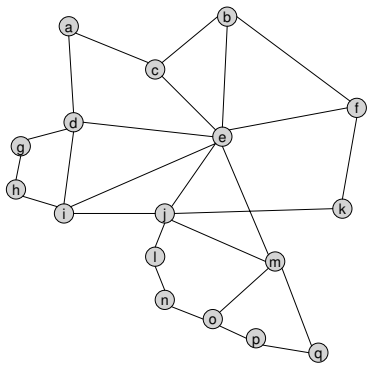
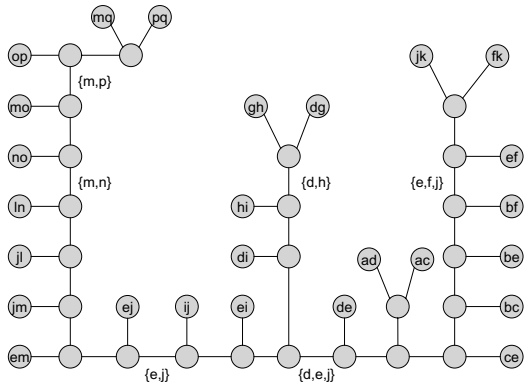


FIGURE 2. Branch decomposition of width 3 for the graph of Figure 1.




A *triangulated* or *chordal* graph is a graph in which every cycle of length of at least 4 has a chord. Related to chordal graphs, another connectivity invariant related to branchwidth called *strong branchwidth* was developed by Tuza [122].

## 2.3. Tangles

A *tangle* in $G$ of *order* $k$ is a set $\mathcal{T}$ of separations of $G$, each of order $< k$, such that

(T1) for every separation $(A, B)$ of $G$ of order $< k$, one of $(A, B)$, $(B, A)$ is an element of $\mathcal{T}$;

(T2) if $(A_1, B_1)$, $(A_2, B_2)$, $(A_3, B_3) \in \mathcal{T}$, then $A_1 \cup A_2 \cup A_3 \neq G$; and

(T3) if $(A, B) \in \mathcal{T}$, then $V(A) \neq V(G)$.

These are called the first, second, and third tangle axioms. The *tangle number* of $G$, denoted by $\theta(G)$, is the maximum order of tangles of $G$. Figure 3 shows the tangle of order 3 for the graph in Figure 1.

FIGURE 3. Tangle of order 3 for the graph $G$ of Figure 1.

**Separation of order 0**

$(\varnothing,\ G)$

**Separations of order 1**

$(v,\ G)\ \forall\ v\ \in\ V(G)$

**Separations of order 2**

$(\{v,\ w\},\ G)\ \forall\ v,\ w\ \in\ V(G)$

$(G[e],\ G[E(G) \setminus e])\ \forall\ e\ \in\ E(G)$

$(G[\{ac, ad\}],\ G[E(G) \setminus \{ac, ad\}])$

$(G[\{dg, gh\}],\ G[E(G) \setminus \{dg, gh\}])$

$(G[\{fk, jk\}],\ G[E(G) \setminus \{fk, jk\}])$

$(G[\{gh, hi\}],\ G[E(G) \setminus \{gh, hi\}])$

$(G[\{jl, lm\}],\ G[E(G) \setminus \{jl, lm\}])$

$(G[\{ln, no\}],\ G[E(G) \setminus \{ln, no\}])$

$(G[\{mq, pq\}],\ G[E(G) \setminus \{mq, pq\}])$

$(G[\{dg, gh, hi\}],\ G[E(G) \setminus \{dg, gh, hi\}])$

$(G[\{jl, ln, no\}],\ G[E(G) \setminus \{jl, ln, no\}])$

$(G[\{mq, op, pq\}],\ G[E(G) \setminus \{mq, op, pq\}])$

$(G[\{dg, di, gh, hi\}],\ G[E(G) \setminus \{dg, di, gh, hi\}])$

$(G[\{mo, mq, op, pq\}],\ G[E(G) \setminus \{mo, mq, op, pq\}])$

$(G[\{mo, mq, no, op, pq\}],\ G[E(G) \setminus \{mo, mq, no, op, pq\}])$

$(G[\{ln, mo, mq, no, op, pq\}],\ G[E(G) \setminus \{ln, mo, mq, no, op, pq\}])$

$(G[\{jl, ln, mo, mq, no, op, pq\}],\ G[E(G) \setminus \{jl, ln, mo, mq, no, op, pq\}])$

$(G[\{jl, jm, ln, mo, mq, no, op, pq\}],\ G[E(G) \setminus \{jl, jm, ln, mo, mq, no, op, pq\}])$

$(G[\{em, jl, jm, ln, mo, mq, no, op, pq\}],\ G[E(G) \setminus \{em, jl, jm, ln, mo, mq, no, op, pq\}])$

$(G[\{em, je, jl, jm, ln, mo, mq, no, op, pq\}],\ G[E(G) \setminus \{em, je, jl, jm, ln, mo, mq, no, op, pq\}])$

The relationship between tangle number and branchwidth is as follows:

**Theorem 1 (Robertson and Seymour [106]).** *For any hypergraph* $G$, $\max(\beta(G), \eta(G)) = \theta(G)$ *unless* $\eta(G) = 0$ *and* $V(G) \neq \varnothing$.

## 2.4. Tree Decompositions

The notions of a tree decomposition and treewidth were introduced by Robertson and Seymour [104] and measure the *tree-likeness* of a graph. A tree decomposition of a graph $G = (V, E)$ is a pair $(\{X_i, i \in I\}, T = (I, F))$ with $X_i \subseteq V$, $i \in I$ and $T = (I, F)$ a tree, such that

(TD1) $\bigcup_{i \in I} X_i = V$;

(TD2) for all $vw \in E$, there is an $i \in I$ with $v, w \in X_i$; and

(TD3) for all $v \in V$, $\{i \in I : v \in X_i\}$ forms a connected subtree of $T$.

The *width* of a tree decomposition $(\{X_i, i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$. The *treewidth* $\tau(G)$ of $G$ is the minimum width over all tree decompositions of $G$. The "$-1$" in the definition of the width of a tree decomposition has cosmetic reasons only: In this way the treewidth of a connected graph equals one if and only if it is a tree (or forest if the graph is unconnected). Figure 4 shows a tree decomposition of the graph in Figure 1. Property (TD3) is sometimes called the interpolation property. Furthermore, note that (TD1) is obsolete if $G$ has no isolated vertices.

The notions of branchwidth and treewidth are closely related to each other, as expressed in the following theorem.

**Theorem 2 (Robertson and Seymour [106]).** *Let* $G = (V, E)$ *be a graph with* $E \neq \varnothing$. *Then* $\max(\beta(G), 2) \leq \tau(G) + 1 \leq \max(\lfloor \frac{3}{2}\beta(G) \rfloor, 2)$.

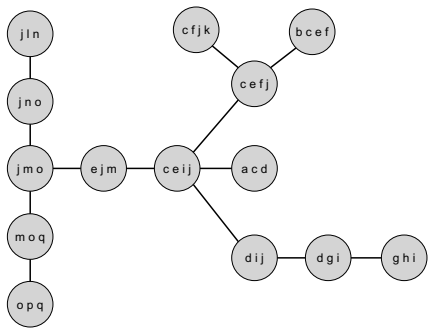
Several equivalent notions for treewidth have been studied over time, such as partial $k$ trees, dimension, and $k$-decomposability. A graph has treewidth of at most $k$ if and only if it is a partial $k$-tree, if and only if the dimension of $G$ is at most $k$, if and only if $G$ is $k$-decomposable. See Bodlaender [26] for further details.

If $T$ is restricted to be a path, we refer to $(\{X_i, i \in I\}, T = (I, F))$ as a *path decomposition* and the best width as the *pathwidth* of $G$. The pathwidth of a tree can be arbitrarily large. One is referred to survey papers by Bodlaender [26] and Bienstock [19] for a more thorough exposition of pathwidth.

## 2.5. Monadic Second-Order Logic

Every class of graphs, when interpreted as a logical object, can be defined by a logical statement. Furthermore, graph properties, such as the existence of disjoint paths, can be stated as a logical condition. The following theorem from Courcelle [49] showed that graph problems that can be stated in monadic second-order logic can be solved in polynomial time for graphs with bounded branchwidth or treewidth.

FIGURE 4. Tree decomposition with width 3 for the graph of Figure 1.



**Theorem 3 (Courcelle [49]).** *Let $\phi$ be a monadic second-order logic problem, and let $\mathcal{K}$ be a class of graphs with branchwidth or treewidth bounded above by $k$. For a graph $G$ in $\mathcal{K}$, it can be determined in polynomial time if $G$ satisfies $\phi$. If $G$ is given with a branch decomposition or tree decompositions with width less than or equal to $k$, then a linear time algorithm exists.*

Monadic second-order logic (MSOL) is an extension of first-order logic that includes vertex and edge sets and quantification (universal and existential) over these sets. In the context of graph theory, first-order logic consists of the logical connectives, $\wedge$, $\vee$, $\neg$, $\Rightarrow$ and $=$ (traditionally interpreted as "and," "or," "not," "implies," and "equals," respectively), variables (e.g., $x_1$, $y_1$, $a$, $b$, $z$), universal ($\forall$) and existential ($\exists$) quantifiers, and a symbol defined to represent the existence of edges between vertices. An example of a first-order formula $\phi$ is

$$\forall x\ \exists y\ [(\mathrm{edg}(x,y) \vee \mathrm{edg}(y,x)) \wedge \neg(x = y)],$$

which is read "for every $x$ there exists $y$ such that the graph in question contains an edge from $x$ to $y$ or the graph in question contains an edge from $y$ to $x$ and it is not true that $x$ equals $y$" or simply the graph contains no isolated vertices. First-order logic is limited in its expressiveness: It can only be used for local properties. Monadic second-order logic's set variables and set quantification allow for larger, more complex expressions.

For problems that involve optimization of a numerical evaluation over the sets of vertices or edges that are introduced in MSOL, we turn to the language of extended monadic second-order logic (EMSOL) and the extended monadic second-order extremum problem presented by Arnborg et al. [14]. These problems are also solvable in polynomial or linear time for graphs with bounded branchwidth or treewidth.

## 3. Graph Minors Theorem

In the 1930s, Kuratowski [90] proved that a graph $G$ is planar if and only if $G$ does not contain a subdivision of $K_5$, the complete graph on five vertices, or $K_{3,3}$, the complete bipartite graph with three vertices on each side of the bipartition. Later, Wagner [123] proved that a graph $G$ is planar if and only if it does not contain $K_5$ or $K_{3,3}$ as a minor of $G$.

Let $\mathcal{F}$ be a class of graphs. $\mathcal{F}$ is *minor closed* if all the minors of any member of $\mathcal{F}$ also belong to $\mathcal{F}$. Given a minor closed class of graphs $\mathcal{F}$, the *obstruction set* of $\mathcal{F}$ is the set of minor minimal graphs that are not elements of $\mathcal{F}$ (i.e., graphs that do not belong to $\mathcal{F}$, but all of their proper minors belong to $\mathcal{F}$). Clearly, any class of graphs embedable on a given surface is a minor closed class.

Recall that the obstruction set for the class of planar graphs was found to be $K_5$ and $K_{3,3}$. But what if we had surfaces other than the sphere? In 1979, Glover et al. [62] exhibited a list of 103 graphs in the obstruction set of projective-planar graphs, and then in 1980 Archdeacon [10] proved that this list is complete. In 1989, Archdeacon and Huneke [11] proved that the obstruction set for any nonorientable surface is finite. Finally, as a corollary of the graph minors theorem (GMT), formerly known as Wagner's conjecture, Robertson and Seymour [103] proved that every minor closed class of graphs has a finite obstruction set. The notions of branch decompositions, tangles, and tree decompositions were beneficial to the proof of GMT.

Given some surface $\Sigma$, an *antichain* for $\Sigma$ is a list of minor minimal graphs that cannot be embedded in $\Sigma$. This means that no member of an antichain is isomorphic to a minor of another. In the early 1960s, Wagner conjectured that *every surface has a finite antichain.* The proof of this conjecture would imply that the obstruction set for any minor closed class is finite (Robertson and Seymour [103]).

**Theorem 4: Graph Minors Theorem (Robertson and Seymour [109]).** *For every infinite sequence of graphs $G_1, G_2, \ldots$, there exists $i$, $j$ with $i < j$ such that $G_i$ is isomorphic to a minor of $G_j$.*

A class with a reflexive and transitive relation is a *quasi-order*. For example, the relation "$H$ is isomorphic to a subgraph of $G$" defines a quasi-order on the class of all graphs. A quasi-order, denoted by $(Q, \leq)$, is *well quasi-ordered* if for every countable sequence $q_1, q_2, \ldots$ of members of $Q$ there exist $1 \leq i < j$ such that $q_i \leq q_j$. Wagner's conjecture is equivalent to stating that the "minor" quasi-order, "$H$ is isomorphic to a minor of $G$," is well quasi-ordered.

One quasi-order that is not well quasi-ordered is the "subgraph" quasi-order stated earlier. This is true because a countable set of circuit graphs, one of each size, is an infinite antichain (Robertson and Seymour [103]). A graph $G$ *topologically contains* a graph $H$ if $G$ has a subgraph that is isomorphic to a subdivision of $H$. Topological containment is not well quasi-ordered either. The set of graphs formed by taking a circuit graph of each size and replacing each edge by two parallel edges is an infinite antichain (Robertson and Seymour [103]); however, Kruskal [89] proved that the class of all trees is well quasi-ordered under topological containment, one of two famous conjectures of Vázsonyi (see [89]). Robertson and Seymour [105] used this theorem to prove:

**Theorem 5.** *For any integer $k$, the class of all graphs with treewidth $\leq k$ is well quasi-ordered by minors.*

This theorem and many more like it proved beneficial for proving Theorem 3. The other conjecture of Vázsonyi [89] was that the class of all graphs with maximum degree of at most 3 is well quasi-ordered under topological containment (Robertson and Seymour [103]).

Another quasi-order is *immersion*. A pair of adjacent edges $ab$ and $bc$ is *lifted* if $ab$ and $bc$ are replaced by the edge $ac$. A graph $H$ is *immersed* in a graph $G$ if $H$ can be obtained from a subgraph of $G$ by lifting pairs of edges. Nash-Williams [96] conjectured that the "immersion" quasi-order is well quasi-ordered. This would imply both of Vázsonyi's conjectures (see [89]).

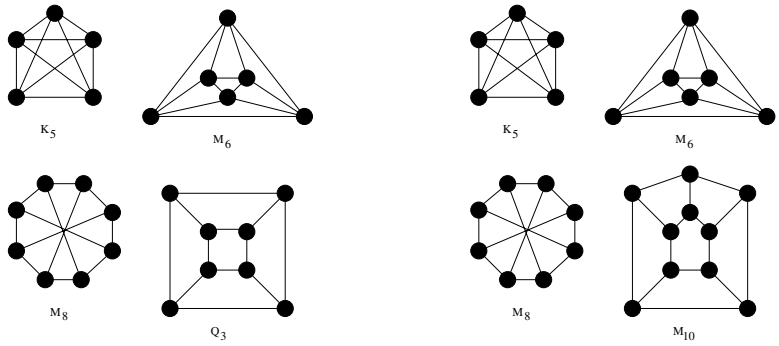
Tree decompositions were introduced in the proof of GMT because large-order tangles have a tree-like structure in their association with small-order tangles. Branch decompositions were introduced in Robertson and Seymour [106], where they studied the relationship between tree decompositions and tangles.

By a result of Robertson and Seymour [106], for a given integer $k$, the class of graphs with branchwidth of at most $k$ is a minor closed class, meaning that if $G$ is a graph that has branchwidth of at most $k$, any minor of $G$ has branchwidth of at most $k$. Hence, the class of graphs with branchwidth of at most $k$ and the class of graph with treewidth of at most $k$ have finite obstruction sets. The only completely known obstruction sets are for graphs with branchwidth or treewidth of 2 and 3. The obstruction set for both graphs with branchwidth of at most 2 and graphs with treewidth of at most 2 is $K_4$. Bodlaender and Thilikos [34] proved that a graph has branchwidth of at most 3 if and only if it does not have $K_5$, $Q_3$, $M_6$, and $M_8$ as minor; see Figure 5(a). Similarly, Arnborg et al. [15] and Satyanarayana and Tung [113] independently proved that the obstruction set for graphs with treewidth of at most 3 is $K_5$, $M_6$, $M_8$, and $M_{10}$; see Figure 5(b). The complete obstruction sets are not known for larger values of either treewidth or branchwidth; however, Hicks [70] did prove that the Petersen graph is a member of the obstruction set for graphs with branchwidth of at most 4. Note that the size of an obstruction for graphs with branchwidth or treewidth of at most $k$ probably grows exponentially with $k$; it may prove extremely difficult to characterize these obstructions sets for large $k$. We refer the reader to the work of Bienstock and Langston [20] for a more thorough survey of GMT.

## 4. Branch Decompositions and Algorithms

This section details constructing branch decompositions with width as small as possible and the use of branch decomposition-based algorithms for solving discrete optimization problems.

FIGURE 5. Obstruction set for graphs with branchwidth and treewidth at most 3.




(a) Obstruction set for graphs with $\beta(G) \leq 3$

(b) Obstruction set for graphs with $\tau(G) \leq 3$

## 4.1. Constructing Branch Decompositions

Because the complexity of branch decomposition-based algorithms is typically exponential in the given fixed width of an input branch decomposition (cf. §4.2), finding branch decompositions whose associated width is as small as possible is vital to the performance of a branch decomposition-based algorithm.

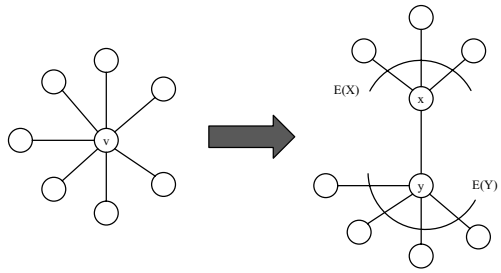
**4.1.1. Construction in Theory.** By a result of Seymour and Thomas [116], computing the branchwidth and finding an optimal branch decomposition of a general graph is $\mathcal{NP}$-hard. However, there is a polynomial-time algorithm in Robertson and Seymour [108] to approximate the branchwidth of a graph within a factor of 3. For example, the algorithm decides if a graph has branchwidth of at least 10 or finds a branch decomposition with width of at most 30. This algorithm and its improvements by Bodlaender [25], Bodlaender and Kloks [31], and Reed [101] are only of theoretical importance. Bodlaender and Thilikos [34] did give an algorithm to compute the optimal branch decomposition for any chordal graph of, with maximum clique size of at most 4, but the algorithm has been shown to be practical only for a particular type of three-tree. Bodlaender and Thilikos [33] also developed a tree decomposition-based linear time algorithm for finding an optimal branch decomposition, but it appears to be computationally impractical as well.

In terms of planar graphs, Fomin and Thilikos [54] proved that the branchwidth of any planar graph is asymptotically bounded by the square root of the graph's number of nodes $(2.122\sqrt{n})$ based on work on planar separators by Alon et al. [7]. This work also offered a complexity bound for the minimum dominating set problem on planar graphs smaller than any known complexity bound for the problem, including the work of Alber and Niedermeier [3] using tree decompositions. The authors have produced similar results for $\mathcal{NP}$-hard problems like the independent set problem, the longest-cycle problem and the bisection problem for planar graphs (Fomin and Thilikos [55]). Kloks et al. [80, 81] gave a polynomial-time algorithm to compute the branchwidth of interval graphs, but this algorithm has not been shown to be practical.

**4.1.2. Construction in Practice.**

***Tree Building.*** To construct a branch decomposition, start with a partial branch decomposition and refine this decomposition until the tree is ternary. The underlying structure used in constructing a branch decomposition is the separation; see §2.2. Without loss of generality, we only use separations $(G_1, G_2)$ such that $E(G_1)$ and $E(G_2)$ are nonempty. Separations are vital to the construction of a branch decomposition, because finding separations will help refine partial branch decompositions into branch decompositions. In this section, we may assume that the input graph $G$ is biconnected, because one can derive an optimal branch decomposition for a disconnected graph $G$ from the optimal branch decompositions of $G$'s

FIGURE 6. A one split.




connected components; one can also derive an optimal branch decomposition for a connected graph $H$ from the optimal branch decompositions of the biconnected components of $H$.

Given a partial branch decomposition, the studied refinements are *one splits* and *two splits*. Let $G$ represent our input graph and let $(T_1, \nu)$ be a partial branch decomposition of $G$. Let $v$ be a nonleaf node of $T_1$ with degree greater than 3 and denote $D_v$ as the set of edges incident with $v$. For a set $S \subseteq V(G)$, let he$(S)$ denote a hyperedge where the ends of the hyperedge are the elements in $S$. Define $H^v$ as the hypergraph constructed from the union of hyperedges he(mid$(e)$) for all $e \in D_v$. Therefore, if $T_1$ is a star, then $H^v$ would correspond to $G$ because $G$ is assumed to be biconnected. Let $(X, Y)$ be a separation of $H^v$. Create the tree $T_2$ by replacing $v$ with nodes $x$ and $y$ and the edge $xy$ where $x$ would be incident with the edges that correspond to $E(X)$ and $y$ would be incident with the edges that correspond to $E(Y)$. This procedure is called a *one split*. The middle set for the edge $xy$ would be $V(X) \cap V(Y)$; Figure 6 offers an illustration of a one split.

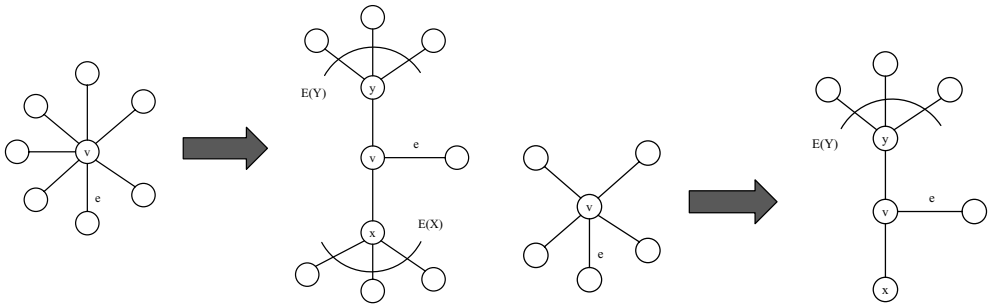
Let $G$, $(T_1, \nu)$, and $v$ be defined as in the previous paragraph. Let $e$ be an edge incident with $v$ and let he$(e)$ denote the hyperedge of $H^v$ that corresponds to $e$. Let $(X, Y)$ be a separation of the hypergraph $H^v \setminus \{\text{he}(e)\}$. Without loss of generality, assume that the cardinality of $E(X)$ is at most the cardinality of $E(Y)$. If the cardinality of $E(X)$ is greater than 1, create $T_2$ by adding new nodes $x$ and $y$ and edges $vx$ and $vy$ to $T_1$ with $x$ incident with the edges corresponding to $E(X)$ and $y$ incident with the edges corresponding to $E(Y)$. Otherwise, create $T_2$ by inserting a new node $y$ and edge $vy$ with $y$ incident with the edges corresponding to $E(Y)$. The middle sets of the new edges in either case would be

$$\text{mid}(vx) = (V(Y) \cup \text{mid}(e)) \cap V(X), \tag{1}$$

$$\text{mid}(vy) = (V(X) \cup \text{mid}(e)) \cap V(Y). \tag{2}$$

This procedure is called a *two split*. Figure 7 offers the two examples of a two split. Notice that a two split when $|E(X)| = 1$ is equivalent to a one split with $|E(X)| = 2$; otherwise, the two procedures do not yield the same results.

FIGURE 7. Two splits.




(a) $|E(X)| > 1$ (b) $|E(X)| = 1$

To build a branch decomposition, start with a partial branch decomposition whose tree is a star, and conduct a sequence of one and two splits to achieve a branch decomposition. The tree-building aspect of using only one splits is equivalent to the tree-building aspect developed by Cook and Seymour [47, 48], and the tree-building aspect of using only two splits is equivalent to the tree-building aspect developed by Robertson and Seymour [108].

A partial branch decomposition $(T, \nu)$ of a graph $G$ is called *extendible* given that $\beta(H^v) \leq \beta(G)$ for every nonleaf node $v \in V(T)$. This follows from the fact that if every $H^v$ had branchwidth of at most some number $k$, then one could use the optimal branch decompositions of the hypergraphs to build a branch decomposition of $G$ whose width is at most $k$. Even though a partial branch decomposition whose tree is a star is extendible, it is $\mathcal{NP}$-hard to check whether an arbitrary partial branch decomposition is extendible for general graphs. In contrast, this is not the case for planar graphs, as discussed later.

A separation is called *greedy* or *safe* (Cook and Seymour [47, 48]) if the next partial branch decomposition created by the use of the separation in conjunction with a one or two split is extendible if the previous partial branch decomposition was extendible. In particular, Cook and Seymour [47, 48] describe three types of safe separations; the first and more general type is called a *push*. For a hypergraph $H$ and $F$, a subset of nodes or edges, let $H[F]$ denote the subhypergraph of $H$ induced by $F$. The push separation is described in the following lemma.

**Lemma 1 (Cook and Seymour [47, 48]).** *Let $G$ be a graph with a partial branch decomposition $(T, \nu)$. Let $v \in V(T)$ have degree greater than 3, and let $D_v \subseteq E(T)$ be the set of edges incident with $v$. Also, let $H^v$ be the corresponding hypergraph for $v$. Suppose there exist $e_1, e_2 \in E(T)$ incident with $v$ such that $|(\text{mid}(e_1) \cup \text{mid}(e_2)) \cap \bigcup\{\text{mid}(f) : f \in D_v \setminus \{e_1, e_2\}\}| \leq \max\{|\text{mid}(e_1)|, |\text{mid}(e_2)|\}$. Let $h_{e_1}, h_{e_2} \in E(H^v)$ be the corresponding hyperedges for $e_1$ and $e_2$, respectively. Then the resulting partial branch decomposition after taking a one split using the separation $(H^v[\{h_{e_1}, h_{e_2}\}], H^v[E(H^v) \setminus \{h_{e_1}, h_{e_2}\}])$ is extendible if $T$ was extendible.*

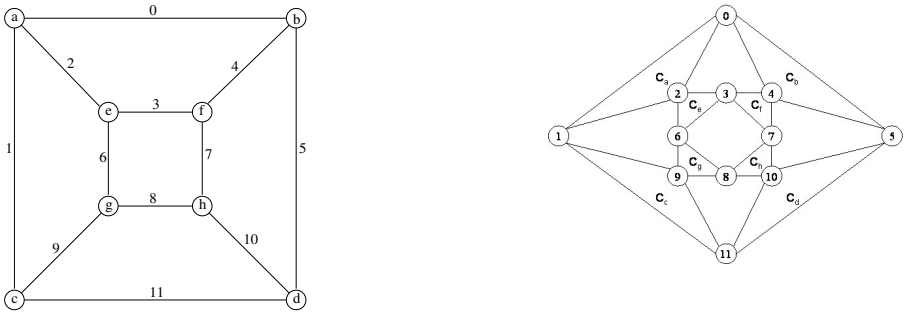
The other types of safe separations utilize two-separations and three-separations that satisfy some simple conditions. First, given a partial branch decomposition of a biconnected graph, if a separation $(X, Y)$ is found such that $|V(X) \cap V(Y)| = 2$, then $(X, Y)$ is safe. This is due to the fact that any two-separation is *titanic* in a biconnected graph (Robertson and Seymour [106]). All three-separations $(X, Y)$ are safe unless $V(X) \cap V(Y)$ corresponds to an independent set in $G$ and either $V(X) \setminus V(Y)$ or $V(Y) \setminus V(X)$ has cardinality 1; this is another result derived by Robertson and Seymour [106].

***Planar Graphs.*** For planar (hyper)graphs, there exists a polynomial-time algorithm called the *ratcatcher method* (Seymour and Thomas [116]) to compute the branchwidth. We briefly comment on the background behind the method and related results for computing the branchwidth of planar graphs.

Let $G$ be a graph with node set $V(G)$ and edge set $E(G)$. Let $T$ be a tree having $|V(G)|$ leaves in which every nonleaf node has degree 3. Let $\mu$ be a bijection between the nodes of $G$ and the leaves of $T$. The pair $(T, \mu)$ is called a *carving decomposition* of $G$. Notice that removing an edge $e$ of $T$ partitions the nodes of $G$ into two subsets $A_e$ and $B_e$. The *cut set* of $e$ is the set of edges that are incident with nodes in both $A_e$ and $B_e$ (also denoted $\delta(A_e)$ or $\delta(B_e)$). The *width* of a carving decomposition $(T, \mu)$ is the maximum cardinality of the cut sets for all edges in $T$. The *carvingwidth* for $G$, $\kappa(G)$, is the minimum width over all carving decompositions of $G$. A carving decomposition is also known as a *minimum-congestion routing tree*, and one is referred to Alvarez et al. [8] for a link between carvingwidth and network design. The ratcatcher method is really an algorithm to compute the carvingwidth for planar graphs. To show the relation between carvingwidth and branchwidth, we need another definition.

Let $G$ be a planar (hyper)graph and let $G$ also denote a particular planar embedding of the graph on the sphere. For every node $v$ of $G$, the edges incident with $v$ can be ordered in a clockwise or counterclockwise order. This ordering of edges incident with $v$ is the *cyclic order* of $v$. Let $M(G)$ be a graph with the vertex set $E(G)$. For a node $v \in V(G)$, define the cycle

FIGURE 8. $Q_3$ and its medial graph.




(a) $Q_3$ (b) $M(Q_3)$

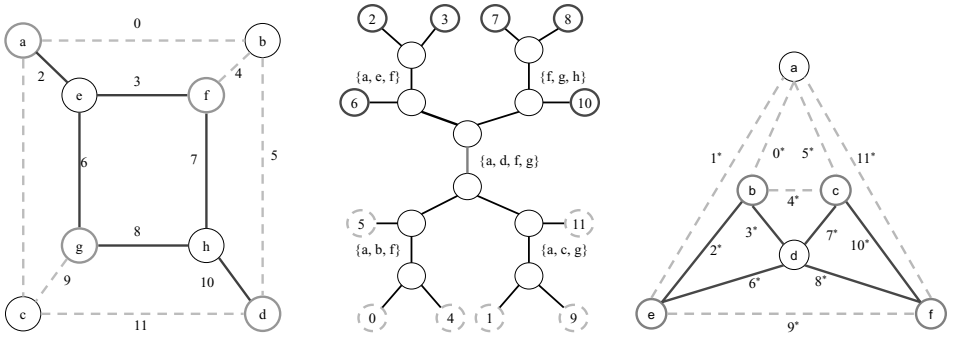
$C_v$ in $M(G)$ as the cycle through the nodes of $M(G)$ that correspond to the edges incident with $v$ according to $v$'s cyclic order in $G$; the edges of $M(G)$ is the union of cycles $C_v$ for all $v \in V(G)$. $M(G)$ is called a *medial graph* of $G$; see Figure 8. Notice that every connected planar hypergraph $G$ with $E(G) \neq \varnothing$ has a medial graph, and every medial graph is planar. In addition, notice that there is a bijection between the regions of $M(G)$ and the nodes and regions of $G$. Hence, one can derive, using the theory of Robertson and Seymour [107], that if a planar graph and its dual are both loopless, then they have the same branchwidth; see Hicks [70]. Figure 9 illustrates this result by presenting one branch decomposition for both $Q_3$ and $M_6$. For the relationship between branchwidth and carvingwidth, Seymour and Thomas [116] proved:

**Theorem 6 (Seymour and Thomas [116]).** *Let $G$ be a connected planar graph with $|E(G)| \geq 2$, and let $M(G)$ be the medial graph of $G$. Then the branchwidth of $G$ is half the carvingwidth of $M(G)$.*

Therefore, computing the carvingwidth of $M(G)$ gives us the branchwidth of $G$. Also, having a carving decomposition of $M(G)$, $(T, \mu)$, gives us a branch decomposition of $G$, $(T, \nu)$, such that the width of $(T, \nu)$ is exactly half the width of $(T, \mu)$. The ratcatcher method actually computes the carvingwidth of planar graphs. In addition, the ratcatcher method does not search for low cut sets in the medial graph, but for objects that prohibit the existence of low cut sets. These objects are called *antipodalities*; see Seymour and Thomas [116] for more details. The ratcatcher method has time complexity $O(n^2)$, but requires a considerable amount of memory for practical purposes. A slight variation that is more memory friendly was offered by Hicks [74] at the expense of the time complexity going up to $O(n^3)$.

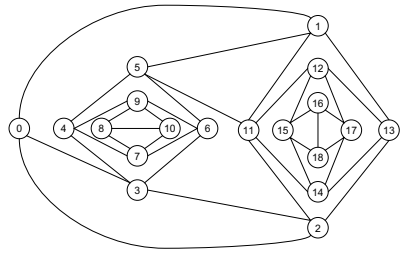
The original algorithm developed by Seymour and Thomas [116] to construct optimal branch decompositions had complexity $O(n^4)$ and used the ratcatcher method to find

FIGURE 9. $Q_3$ and $M_6$ have branchwidth 4.




(a) $Q_3$ (b) $(T, \nu)$ (c) Dual of $Q_3$: $M_6$

FIGURE 10. Tamaki's heuristic [119] gives a width bounded below by 6; the branchwidth is 3.




extendible separations. A practical improvement on this algorithm using a more thorough divide-and-conquer approach was offered by Hicks [75]. Recently, Gu and Tamaki [65] found an $O(n^3)$ time algorithm utilizing the ratcatcher method by bounding the number of calls to the ratcatcher method by $O(n)$. In addition, Tamaki [119] offered a linear time heuristic for constructing branch decompositions of planar graphs; the heuristic could find a branch decomposition of a 2,000-node planar graph in about 117 milliseconds on a 900 MHz Ultra SPARC-III. The heuristic uses the medial-axis tree of $M(G)$ derived from a breadth-first search tree of $M(G)^*$. Thus, the computed width is bounded below by the height of breadth-first search tree; the difference between this parameter (bounded below by the radius of the dual of the medial graph) and the branchwidth could be huge using a similar construction as in Figure 10. Figure 10 raises an interesting question: What characteristics of a planar graph $G$ guarantee that $\beta(G)$ will be equal to the radius of $M(G)^*$?

***General Graphs.*** For general graphs, most work has been done utilizing heuristics to actually construct branch decompositions. Cook and Seymour [47, 48] gave a heuristic algorithm to produce branch decompositions. Their heuristic is based on spectral graph theory and the work of Alon [6]. Moreover, Hicks [71] also found another branchwidth heuristic that was comparable to the algorithm of Cook and Seymour. This heuristic finds separations by minimal vertex separators between diameter pairs.

In addition, Hicks [73] has developed a branch decomposition-based algorithm for constructing an optimal branch decomposition based on the notion of a *tangle basis*. For an integer $k$ and hypergraph $G$, a *tangle basis* $\mathcal{B}$ of order $k$ is a set of separations of $G$ with order $<k$ such that

(B1) $(G[e], G[E(G) \setminus e]) \in \mathcal{B}$, $\forall\ e \in E(G)$ if $\eta(e) < k$;

(B2) $(C, D) \in \mathcal{B}$ and $\nexists e \in E(G)$ such that $G[e] = C$ if and only if $\exists (A_1, B_1)$, $(A_2, B_2) \in \mathcal{B}$ such that $A_1 \cup A_2 = C$ and $B_1 \cap B_2 = D$; and

(B3) $\mathcal{B}$ obeys the tangle axioms T2 and T3.

We will refer to B1, B2, and B3 as the tangle basis axioms. A tangle basis $\mathcal{B}$ in $G$ of order $k$ is *connected* if every separation $(A, B)$ of $\mathcal{B}$ has $A$ connected. Define the *connected tangle basis number* of $G$, denoted by $\theta'(G)$, as the maximum order of any connected tangle basis of $G$. The relationship between connected tangle bases and tangles is given in the following theorem:

**Theorem 7 (Hicks [73]).** *If hypergraph $G$ is connected such that $\beta(G) \geq \eta(G)$, then $\beta(G) = \theta'(G)$.*

The relationship shown in this theorem was used by Hicks [73] to construct a branch decomposition-based algorithm that tests whether a given connected graph has branchwidth of at most $k-1$ (where $k \geq 3$) to be used in a practical setting to compute the branchwidth and the optimal branch decomposition of a graph.

In the same vein of connected tangle bases, Fomin et al. [56] have developed the notion of connected branch decompositions for two-edge-connected graphs. Given a graph $G$ and

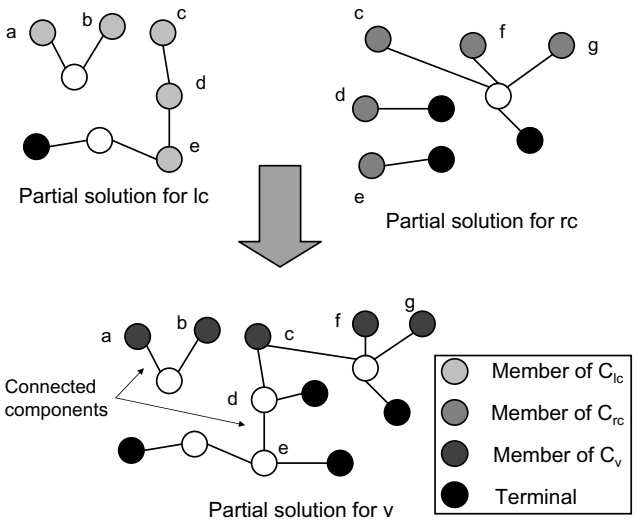
a branch decomposition $(T, \nu)$, $(T, \nu)$ is called *connected* if for every edge $e \in E(T)$, the two-edge induced subgraphs of $G$ corresponding to the two components of $T \setminus \{e\}$ are both connected. Fomin et al. [56] showed that, given a two-edge-connected graph $G$ and a branch decomposition of $G$ of width $k$, there also exists a connected branch decomposition of $G$ of width $k' \leq k$. The authors used this result to approximate in polynomial time the connected search strategy for a graph.

## 4.2. Branch Decomposition-Based Algorithms

The blueprint of a branch decomposition-based algorithm typically consists of two steps: transforming the tree of the branch decomposition of a graph $G$ into a rooted binary tree, and visiting the nodes of the tree in post depth-first search order to generate a solution for the problem of interest. The tree $T$ is transformed into a rooted binary tree by selecting an edge $ab$ and replacing $ab$ by a node $r$, the *root*, and edges $ar$ and $rb$. Because each leaf of $T$ corresponds to a particular edge of $G$, each rooted subtree of $T$ corresponds to a particular subgraph of $G$. Also, every node $v$ of $T$ that is not a leaf node and not the root of the tree is adjacent to three nodes: a parent node $p$, a left child $lc$, and a right child $rc$. For each tree node $v$, let $G_v$ denote the subgraph of $G$ induced by the edges corresponding to the leaves of the tree that are descendants of $v$. In addition, let $C_v$ denote the middle set of the parent edge $pv$. The algorithm then visits the tree nodes in post depth-first order and builds a set of partial solutions for each tree node. By visiting the tree in post depth-first order, all descendants of tree node $v$ are visited before $v$ is visited.

To illustrate further, some details for the Steiner tree problem in graphs are as follows. A partial solution for a tree node $v$ would be a forest $F$ of $G_v$ such that every connected component $D$ of $F$ has the property that $V(D) \cap C_v \neq \varnothing$ unless $F$ is a Steiner tree of $G$. For the case that the tree node $v$ is a leaf, there is one partial solution: $\nu(v)$ is an edge of the desired Steiner tree. For tree node $v$ that is not a leaf, the set of partial solutions of $v$ is contained in the union of the partial solutions of $v$'s children, $lc$ and $rc$, and the merger of partial solutions from both sets. Thus, every partial solution of $lc$ is merged with every partial solution of $rc$ on the node set $C_{lc} \cap C_{rc}$. Figure 11 illustrates this procedure where $C_{lc} = \{a, b, c, d, e\}$, $C_{rc} = \{c, d, e, f, g\}$, $C_v = \{a, b, c, f, g\}$, and the terminals are in black. The number of partial solutions for a tree node $v$ can grow very rapidly because the number of partial solutions for any tree node $v$ is exponential in the cardinality of $C_v$. Thus, the

FIGURE 11. Joining partial solutions for $lc$ and $rc$ to obtain a partial solution for $v$, for the Steiner tree problem.



time complexity of a branch decomposition-based algorithm is exponential in the width of the input branch decomposition, which explains the importance of having low width. In contrast, some partial solutions can be pruned away if they can never be transformed into a feasible Steiner tree. For example, if a partial solution containing a cycle is obtained by merging two partial solutions at a tree node, then that partial solution can be pruned away. Another example is if a partial solution of a tree node $v$ had a connected component $D$ that did not intersect with $C_v$ and all the terminals were not contained in $D$—then that partial solution can also be pruned.

A branch decomposition-based algorithm was offered by Cook and Seymour [47] for the ring-routing problem, which arises in the design of reliable cost-effective synchronous optical networks (SONET networks) and was incorporated into commercial software for Telcordia Technologies (formerly Bellcore). Cook and Seymour [48] also used a branch decomposition-based algorithm on a sparse graph generated by fractional solutions of linear programming (LP) relaxations of a TSP test instance to generate an optimal solution for the sparse graph and an upper bound for the test instance. This technique produced the best-known solutions for the 12 unsolved problems in TSPLIB95, a library of standard test instances for the TSP (Reinelt [102]). Hicks has also developed computationally efficient branch decomposition-based algorithms for minor containment (Hicks [72]) and optimal branch decompositions (Hicks [73]). One is also referred to the work of Christian [43] for other practical algorithms. Some examples of branch decomposition-based algorithms proposed in theory are the notable work of Fomin and Thilikos [54] and the work of Alekhnovich and Razborov [5], who used the branchwidth of hypergraphs to design a branch decomposition-based algorithm, in theory to solve satisfiability problems. Despite the aforementioned examples of research in branch decompositions offering glimpses of potentially rewarding research in the field, overall research in the area has been relatively ignored and vastly unexplored compared to research in tree decompositions.

## 5. Tree Decompositions and Algorithms

Besides branch decompositions, tree decompositions have also been used to solve discrete optimization problems. In this section, we consecutively discuss the construction of a tree decomposition with width as small as possible, lower bounds on the treewidth of a graph, and finally algorithms for optimization problems that are based on tree decompositions.

### 5.1. Constructing Tree Decompositions

The construction of an arbitrary tree decomposition is trivial: Just take a single-node tree and assign to this node a bag containing all vertices in the graph. This pair of a tree and a bag satisfies all three conditions of a tree decomposition and thus qualifies as a tree decomposition. Despite its formal status, it does not deserve the name tree *decomposition* from a practical perspective: The tree decomposition does not decompose anything and is not useful for solving combinatorial problems.

As with branch decompositions, the worth of a tree decomposition depends on its width. The smaller the size of the largest bag, the more promising a tree decomposition is for solving combinatorial optimization problems. Therefore, the construction of a tree decomposition with a small width is desirable. We first discuss the theoretical results. Afterwards, we review how tree decompositions are constructed in practice.

**5.1.1. Construction in Theory.** We are confronted with a difficult problem related to treewidth: Computing the treewidth of a graph is $\mathcal{NP}$-hard. Many theoretical assessments have been made concerning the difficulty of computing treewidth. The formal decision problem of treewidth asks whether there exists a tree decomposition with width at most $k$ for some integer $k > 0$. If $k$ is part of the input, $\mathcal{NP}$-completeness was proved by Arnborg et al. [13]. If $k$ may be considered as a constant, not part of the input, the best algorithm

has been given by Bodlaender [25], and checks in linear time whether or not a tree decomposition with width at most $k$ exists. The $O(n)$ notation for this algorithm, however, hides a huge constant coefficient that obstructs its practical computational value. An experimental evaluation by Röhrig [110] revealed that the algorithm is computationally intractable, even for $k$ as small as 4.

Graphs with treewidth of at most 4 can be characterized either directly or indirectly. As already pointed out in §2, $\tau(G) = 1$ if and only if $G$ is a forest. A graph $G$ has $\tau(G) \leq 2$ if and only if its biconnected components are series-parallel graphs (Bodlaender and Fluiter [35]). Arnborg and Proskurowski [12] gave six reduction rules that reduce $G$ to the empty graph if and only if $\tau(G) \leq 3$. Sanders [112] provided a linear time algorithm for testing $\tau(G) \leq 4$.

Besides forests and series-parallel graphs, the complexity of treewidth for some special classes of graphs are known (by presenting either a polynomial-time algorithm or an $\mathcal{NP}$-completeness proof). We refer the interested reader to two surveys on the topic by Bodlaender [24, 29]. Most remarkable in this context is that, so far, the complexity of treewidth for planar graphs is unknown, whereas for branchwidth a polynomial-time algorithm exists; see §4. Lapoire [91] and Bouchitté et al. [40] proved that the treewidth of a planar graph and of its geometric dual differ by at most 1.

As it is $\mathcal{NP}$-complete to decide whether the treewidth of a graph is at most $k$, a natural way to proceed is to consider polynomial-time approximation algorithms for the problem. Given a graph $G$ with $\tau(G) = k$, the best algorithms are given by Bouchitté et al. [41] and Amir [9], both providing a tree decomposition of width at most $O(k \log k)$ (i.e., an $O(\log k)$ approximation). So far, neither is a constant approximation algorithm known nor is it proven that no such algorithm exists.

If we insist on computing the treewidth exactly, unless $\mathcal{P} = \mathcal{NP}$, the only way to go is the development of an exponential time algorithm; see Woeginger [124] for a survey in this recent branch of algorithm theory. For treewidth, Arnborg et al. [13] gave an algorithm with running time $O(2^n \text{poly}(n))$, where $\text{poly}(n)$ is a polynomial in $n$. Fomin et al. [57] presented a $O(1.9601^n \text{poly}(n))$ algorithm. Whether these algorithms are of practical usefulness for computing treewidth is a topic of further research.

**5.1.2. Construction in Practice.** Most results presented in the previous subsection are of theoretical interest only: The computational complexity hides huge constant coefficients that make the algorithms impractical for actually computing treewidth. So far, only the reduction rules for treewidth of at most 3 have been proved to be of practical use in preprocessing the input graph. However, in all those cases where the treewidth is larger than 3, we have to turn to heuristics without any performance guarantee. Many of the results reviewed here have been tested on graphs of different origin, see TreewidthLIB [28] for a compendium.

***Preprocessing.*** The reduction rules of Arnborg and Proskurowski [12] not only reduce graphs of treewidth of at most 3 to the empty graph, but can also be used as a preprocessing technique to reduce the size of general graphs. In Bodlaender et al. [39], the rules have been adapted and extended so as to preprocess general graphs. Given an input graph $G$, a value low is maintained during the preprocessing such that $\max\{\text{low}, \tau(G')\} = \tau(G)$, where $G'$ is the (partly) preprocessed graph. If at any point no further preprocessing rules can be applied anymore, a tree decomposition of the preprocessed graph $G'$ is computed (see below). Finally, given a tree decomposition for $G'$, a tree decomposition for the input graph can be obtained by reversal of the preprocessing steps and adapting the tree decomposition appropriately. Computational experiments have shown that significant reductions in the graph size can be achieved by these rules.

The above-mentioned preprocessing rules emphasize the removal of vertices from the graph. Another way to reduce the complexity of finding a good tree decomposition is the splitting of the input graph into smaller graphs for which we can construct a tree decompo-

sition independently. In Bodlaender and Koster [32], so-called *safe separators* are introduced for this purpose. A separator $S$ is a set of vertices whose removal disconnects a graph $G$. Let $V^i$, $i=1,\ldots,p$ ($p\geq 2$) induce the connected components of $G-S$. On each of the connected components $G[V^i]$, a graph $G^i$ is defined as $G[V^i\cup S]\cup \text{clique}(S)$, where clique$(S)$ denotes a complete graph, or *clique*, on $S$. If $\tau(G)=\max_{i=1,\ldots,p}\tau(G^i)$, then $S$ is called *safe* for treewidth. In particular, clique separators (i.e., $S$ induces a clique) and almost clique separators (i.e., $S$ contains a $|S|-1$ clique) are safe. Experiments revealed that, roughly speaking, by applying a safe separator decomposition to a graph, it remains to construct a tree decomposition for the smaller graphs given by the decomposition.

***Exact Algorithms.*** Although treewidth is $\mathcal{NP}$-hard in general, there have been a couple of attempts to tackle the problem by exact approaches. Shoikhet and Geiger [117] implemented a modified version of the $O(n^{k+2})$ algorithm by Arnborg et al. [13]. A branch-and-bound algorithm based on vertex ordering has been proposed by Gogate and Dechter [63].

***Upper-Bound Heuristics.*** The operations research toolbox for constructing solutions to combinatorial optimization problems has been opened but not yet fully explored for computing the treewidth of a graph. Most heuristics are of a constructive nature: According to some principle, we construct a tree decomposition from scratch. Improvement heuristics as well as metaheuristics are less frequently exploited.

At first sight, condition (TD3) does not simplify the construction of good tree decompositions from scratch. However, an alternative definition of treewidth by means of graph triangulations reveals the key to constructive heuristics. A *triangulated* or *chordal* graph is a graph in which every cycle of length of at least 4 has a chord. A *triangulation* of a graph $G=(V,E)$ is a chordal graph $H=(V,F)$ with $E\subseteq F$.

**Lemma 2.** *Let $G$ be a graph, and let $\mathcal{H}$ be the set of all triangulations of $G$. Then, $\tau(G)=\min_{H\in\mathcal{H}}\omega(H)-1$, where $\omega(H)$ is the size of the maximum clique in $H$.*

Thus, if $G$ is triangulated, then $\tau(G)=\omega(G)-1$, otherwise we have to find a triangulation of $H$ with small maximum clique size. Several algorithms exist to check whether $G$ is triangulated, or to construct a triangulation of $G$. All are based on a special ordering of the vertices. A *perfect elimination scheme* of a graph $G=(V,E)$ is an ordering of the vertices $v_1,\ldots,v_n$ such that for all $v_i\in V$, $\delta_{G[v_i,\ldots,v_n]}(v_i)$ induce a clique.

**Lemma 3 (Gavril [59], Golumbic [64]).** *A graph $G$ is triangulated if and only if there exists a perfect elimination scheme.*

To check whether a graph is triangulated, it is thus enough to construct a perfect elimination scheme or to prove that no such scheme exists. The *lexicographic breadth first search* (LEX) recognition algorithm by Rose et al. [111] constructs in $O(n+m)$ time a perfect elimination scheme if such a scheme exists. The *maximum cardinality search* (MCS) by Tarjan and Yannakakis [120] does the same (with the same complexity in theory, but is faster in practice). Both algorithms can be adapted to find a triangulation $H$ if $G$ is not triangulated itself. With the help of Lemma 2, a tree decomposition can be constructed with width equal to the maximum clique size of $H$ minus one. The triangulated graph given by both algorithms is not necessarily minimal in the sense that there may not exist a triangulation $H'=(V,F')$ with $E\subset F'\subset F$. As unnecessarily inserted edges can increase the maximum clique size, it is desirable to find a minimal triangulation. For both algorithms there exist variants that guarantee the ability to find a minimal triangulation $H'$ of $G$, known as LEX–M (Rose et al. [111]) and MCS–M (Berry et al. [17]), respectively. See Koster et al. [84] for some experimental results for LEX–P, MCS, and LEX–M. Recently, Heggernes et al. [69] proposed a new algorithm to find a minimal triangulation. Alternatively, we can add as a postprocessing step to MCS and LEX–P an algorithm that turns a triangulation into a

minimal triangulation (Blair et al. [22], Dahlhaus [51], and Heggernes and Villanger [68]). Note that in case the input graph is chordal, the minimal triangulation is the graph itself, and the treewidth of the graph is computed exactly with all described algorithms.

The *minimal fill-in* problem is another problem that is studied in relation to triangulation of graphs. The minimum fill-in of a graph is the minimum number of edges to be added to a graph such that the resulting graph is chordal/triangulated. This problem is known to be $\mathcal{NP}$-hard (Yannakakis [126]), but it is not difficult to think of two heuristics. The first one is a greedy algorithm: Select repeatedly the vertex for which the fill-in among its neighbors is minimized, turn its neighbors into a clique, and remove that vertex. This algorithm is called *greedy fill-in* (GFI), or simply the minimum fill-in algorithm in some articles. The second algorithm does the same except that it selects the vertex according to the minimum degree. See Bachoore and Bodlaender [16] and Clautiaux et al. [44, 45] for computational experiments and fine-tuning of these algorithms.

Except for the algorithm that turns a triangulation into a minimal triangulation, all heuristics described so far are constructive. The algorithm described in Koster [83] can be viewed as an improvement heuristic, similar to the tree-building idea for branchwidth. Given a tree decomposition, it tries to replace the largest bag(s) by smaller ones, preserving all conditions of a tree decomposition. If the algorithm starts with the trivial tree decomposition consisting of a single node, the algorithm can be viewed as a constructive algorithm; if it starts with a tree decomposition constructed by another method, it can be considered an improvement heuristic as well.

Metaheuristics have been applied to treewidth as well. Clautiaux et al. [45] experimented with a tabu search algorithm. For a problem closely related to treewidth, Kjærulff [79] applies simulated annealing, whereas Larrañaga et al. [92] use a genetic algorithm.

***Branchwidth and Treewidth.*** As already pointed out in §2, the notions branchwidth and treewidth are closely related. Given a branch decomposition with width $k$, a tree decomposition with width at most $\lfloor 3/2k \rfloor$ can be constructed in polynomial time: Let $i$ be an internal node of the branch decomposition and let $j_1$, $j_2$, $j_3$ be its neighbors. Moreover, let $U_{j_1}$, $U_{j_2}$, $U_{j_3} \subseteq V$ be the vertex sets induced by edges corresponding to the leafs of the subtrees rooted at $j_1$, $j_2$, and $j_3$ respectively. Thus $\text{mid}(ij_1) := U_{j_1} \cap (U_{j_2} \cup U_{j_3})$, $\text{mid}(ij_2) := U_{j_2} \cap (U_{j_1} \cup U_{j_3})$, and $\text{mid}(ij_3) := U_{j_3} \cap (U_{j_1} \cup U_{j_2})$. Now, associate with node $i$ the bag $X_i := \text{mid}(ij_1) \cup \text{mid}(ij_2) \cup \text{mid}(ij_3)$. Because the union contains $U_j \cap U_k$, $j, k \in \{j_1, j_2, j_3\}$, $j \neq k$, twice, the size of $X_i$ is at most $\lfloor 3/2k \rfloor$. It is left to the reader to verify that $(\{X_i, i \in I\}, T = (I, F))$ satisfies all conditions of a tree decomposition.

## 5.2. Treewidth Lower Bounds

The heuristics for practical use described above do not generally guarantee a tree decomposition with width close to optimal. To judge the quality of the heuristics, lower bounds on treewidth are of great value. Moreover, obtaining good lower bounds quickly is essential for the performance of branch-and-bound algorithms (see Gogate and Dechter [63]), and the height of a treewidth lower bound is a good indication of the computational complexity of tree decomposition-based algorithms to solve combinatorial optimization problems.

In recent years, substantial progress on treewidth lower bounds has been achieved, both theoretically and practically. Probably the widest-known lower bound is given by the maximum clique size. This can be seen by Lemma 2: The maximum clique of $G$ will be part of a clique in any triangulation of $G$.

Scheffler [114] proved that every graph of treewidth of at most $k$ contains a vertex of degree at most $k$. Stated differently, the minimum degree $\delta(G)$ is a lower bound on the treewidth of a graph. Typically this lower bound is of no real interest, as the minimum degree can be arbitrarily small. Even if the preprocessing rules of the previous section have been applied before, only $\delta(G) \geq 3$ can be guaranteed.

Ramachandramurthi [99, 100] introduced the parameter

$$\gamma_R(G) = \min\Big(n-1, \min_{v,w \in V, v \neq w, \{v,w\} \notin E} \max(d(v), d(w))\Big)$$

and proved that this is a lower bound on the treewidth of $G$. Note that $\gamma_R(G) = n-1$ if and only if $G$ is a complete graph on $n$ vertices. If $G$ is not complete, then $\gamma_R(G)$ is determined by a pair $\{v, w\} \notin E$ with $\max(d(v), d(w))$ as small as possible. From its definition it is clear that $\gamma_R(G) \geq \delta_2(G) \geq \delta(G)$, where $\delta_2(G)$ is the second-smallest degree appearing in $G$ (note $\delta(G) = \delta_2(G)$ if the minimum-degree vertex is not unique). So, we have

$$\delta(G) \leq \delta_2(G) \leq \gamma_R(G) \leq \tau(G)$$

and all these three lower bounds can be computed in polynomial time.

One of the heuristics for constructing a (good) tree decomposition is the maximum cardinality search algorithm (MCS); see §5.1.2. Lucena [94] proved that with the same algorithm a lower bound on the treewidth can be obtained. The MCS visits the vertices of a graph in some order, such that at each step an unvisited vertex that has the largest number of visited neighbors becomes visited (note that the algorithm can start with an arbitrary vertex). An MCS ordering of a graph is an ordering of the vertices that can be generated by the algorithm. The visited degree of a vertex $v$ in an MCS ordering is the number of neighbors of $v$ that are before $v$ in the ordering. The visited degree of an MCS ordering $\psi$ of $G$ is the maximum visited degree over all vertices $v$ in $\psi$ and denoted by $mcslb_\psi(G)$.

**Theorem 8 (Lucena [94]).** *Let $G$ be a graph and $\psi$ an MCS ordering. Then, $mcslb_\psi(G) \leq \tau(G)$.*

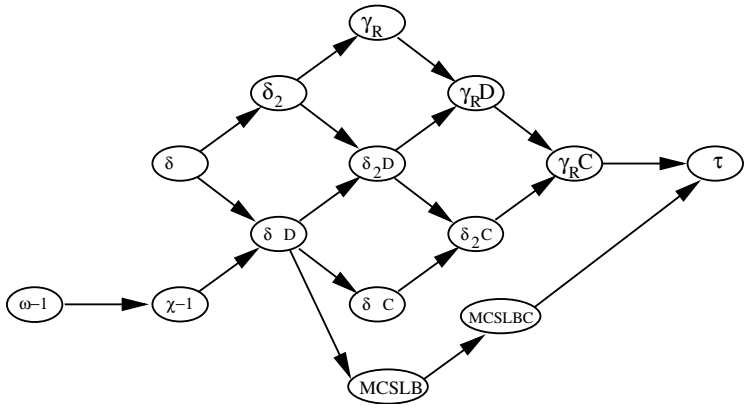
If we define the *maximum visited degree MCSLB*$(G)$ of $G$ as the maximum visited degree over all MCS orderings of graph $G$, then obviously $MCSLB(G) \leq \tau(G)$ as well. Bodlaender and Koster [32] proved that determining whether $MCSLB(G) \leq k$ for some $k \geq 7$ is $\mathcal{NP}$-complete and presented computational results by constructing MCS orderings using tiebreakers for the decisions within the MCS algorithm.

It is easy to see that every lower bound for treewidth can be extended by taking the maximum of the lower bound over all subgraphs or minors: Given an optimal tree decomposition for $G$ and $H$ a subgraph (minor) of $G$, then we can construct a tree decomposition with equal or better width for $H$ by removing vertices from the bags that are not part of the subgraph (minor) and replacing contracted vertices by their new vertex.

In Koster et al. [84], the minimum-degree lower bound has been combined with taking subgraphs. The maximum-minimum degree over all subgraphs, denoted by $\delta D(G)$, is known as the *degeneracy* of a graph $G$, and can be computed in polynomial time by repeatedly removing a vertex of minimum degree and recording the maximum encountered. Szekeres and Wilf [118] proved that $\delta D(G) \geq \chi(G) - 1$, and thus $\delta D(G) \geq \omega(G) - 1$. Hence, the degeneracy provides a lower bound no worse than the maximum clique size, and in addition it can be computed more efficiently. In Bodlaender and Koster [32] it is shown that $MCSLB(G) \geq \delta D(G)$.

Independently, Bodlaender et al. [37] and Gogate and Dechter [63] combined the minimum-degree lower bound with taking minors. The so-called *contraction degeneracy* $\delta C(G)$ is defined as the maximum-minimum degree over all minors of $G$. In Bodlaender et al. [37], it is proven that computing $\delta C(G)$ is $\mathcal{NP}$-hard and computational experiments are presented by applying tiebreakers to the following algorithm: Repeatedly contract a vertex of minimum degree to one of its neighbors and record the maximum encountered. Significantly better lower bounds than the degeneracy are obtained this way. In Wolle et al. [125], further results for contraction degeneracy are discussed, showing, for example, that $\delta C(G) \leq 5 + \gamma(G)$, where $\gamma(G)$ is the *genus* of $G$.

FIGURE 12. Degree-based treewidth lower bounds.




Also, the lower bounds $\delta_2(G)$, $\gamma_R(G)$, and $MCSLB(G)$ can be computed over all subgraphs or minors. In Bodlaender et al. [37] the combination of $MCSLB(G)$ and taking minors has been studied, whereas the combination of $\delta_2(G)$ and $\gamma_R(G)$ with taking subgraphs or minors is the topic of research in Koster et al. [88]. Whereas computing $\delta_2(G)$ over all subgraphs (denoted by $\delta_2 D(G)$) can be computed in polynomial time, surprisingly, computing $\gamma_R(G)$ over all subgraphs (denoted by $\gamma_R D(G)$) is already $\mathcal{NP}$-hard. A two-approximation for $\gamma_R D(G)$ is given by $\delta_2 D(G)$. Furthermore, $\delta_2 D(G) \leq \delta D(G) + 1$ and $\delta_2 C(G) \leq \delta C(G) + 1$, where $\delta_2 C(G)$ is the minor-taking variant of $\delta_2(G)$. Figure 12 shows an overview of the lower bounds for treewidth discussed so far. In practice, $\delta_2 C(G)$ and $\gamma_R C(G)$ are only marginal better (if at all) than the lower bounds computed for the contraction degeneracy.

Another vital idea to improve lower bounds for treewidth is based on the following result.

**Theorem 9 (Bodlaender [27]).** *Let $G = (V, E)$ be a graph with $\tau(G) \leq k$ and $\{v, w\} \notin E$. If there exist at least $k + 2$ vertex disjoint paths between $v$ and $w$, then $\{v, w\} \in F$ for every triangulation $H$ of $G$ with $\omega(H) \leq k$.*

Hence, if we know that $\tau(G) \leq k$ and there exist $k + 2$ vertex disjoint paths between $v$ and $w$, adding $\{v, w\}$ to $G$ should not hamper the construction of a tree decomposition with small width. Clautiaux et al. [44] explored this result in a creative way. First, they compute a lower bound $\ell$ on the treewidth of $G$ by any of the above methods (e.g., $\ell = \delta C(G)$). Next, they assume $\tau(G) \leq \ell$ and add edges $\{v, w\}$ to $G$ for which there exist $\ell + 2$ vertex disjoint paths in $G$. Let $G'$ be the resulting graph. Now, if it can be shown that $\tau(G') > \ell$ by a lower-bound computation on $G'$, our assumption that $\tau(G) \leq \ell$ is false. Hence, $\tau(G) > \ell$ or stated equally $\tau(G) \geq \ell + 1$: An improved lower bound for $G$ is determined. This procedure can be repeated until it is not possible anymore to prove that $\tau(G') > \ell$ (which of course does not imply that $\tau(G') = \ell$).

In Clautiaux et al. [44], $\delta D(G')$ is used to compute the lower bounds for $G'$. Because computing the existence of at least $\ell + 2$ vertex disjoint paths can be quite time consuming, a simplified version checks whether $v$ and $w$ have at least $\ell + 2$ common neighbors. In Bodlaender et al. [38] the above described approach is nested within a minor-taking algorithm, resulting in the best-known lower bounds for most tested graphs; see [28]. In many cases optimality could be proved by combining lower and upper bounds.

For graphs of low genus, in particular for planar graphs, the above described lower bounds typically are far from the real treewidth. For planar graphs, we can once more profit from Theorem 2. Treewidth is bounded from below by branchwidth, and branchwidth can be computed in polynomial time on planar graphs. Hence, a polynomial-time computable lower bound for treewidth of planar graphs is found. Further research in finding lower bounds (based on the concept of brambles (Seymour and Thomas [115])) for (near) planar graphs is underway (Bodlaender et al. [36]). One of these bounds is also a lower bound for branchwidth.

### 5.3. Tree Decomposition-Based Algorithms

All efforts to compute good tree decompositions (and lower bounds on treewidth) have two major reasons:

- Several practical problems in various fields of research are equivalent to treewidth on an associated graph.
- For many $\mathcal{NP}$-hard combinatorial problems that contain a graph as part of the input, polynomial-time algorithms are known in case the treewidth of the graph is bounded by some constant (as is the case for branchwidth).

For a long time, the second reason has been considered to be of theoretical value only, but (as with branchwidth) more and more practical work has been carried out in this direction.

Examples of the first reason can be found in VLSI design, Cholesky factorization, and evolution theory. We refer to Bodlaender [24] for an overview. In this context we also should mention that the control flow graph of goto-free computer programs written in common imperative programming languages like C or Pascal have treewidth bounded by small constants; see Thorup [121] and Gustedt et al. [66]. Recently, Bienstock and Ozbay [21] connected treewidth with the *Sherali-Adams operator* for 0/1 integer programs.

For many $\mathcal{NP}$-complete problems like Independent Set, Hamiltonian Circuit, Chromatic Index (Bodlaender [23]), or Steiner Tree (Korach and Solel [82]) it has been shown that they can be solved in polynomial time if defined on a graph of bounded treewidth. Typically there exists a kind of dynamic programming algorithm based on the tree decomposition. Because such algorithms follow a scheme similar to the branch decomposition-based algorithms described before, we leave out such a formal description (see, e.g., Bodlaender [24] for a description of the algorithm for the independent set problem, or Koster [83, 87] for frequency assignment).

Probably the first tree decomposition-based algorithm that has been shown to be of practical interest is given by Lauritzen and Spiegelhalter [93]. They solve the inference problem for probabilistic (or Bayesian belief) networks by using tree decompositions. Bayesian belief networks are often used in decision support systems. Applications of Bayesian belief networks can be found in medicine, agriculture, and maritime applications.

For problems where integer linear programming turns out to be troublesome, using a tree decomposition-based algorithm could be a good alternative. A demonstrative example in this context is a frequency assignment problem studied by Koster [83] (see also Koster et al. [86, 87]). In the so-called minimum interference frequency assignment problem, we have to assign frequencies to transmitters (base stations) in a wireless network such that the overall interference is minimized. For this purpose, let $G = (V, E)$ be a graph, and for every vertex $v \in V$, a set of radio frequencies $F_v$ is given. For every pair $\{v, w\}$ and every $f \in F_v$, $g \in F_w$, a penalty $p_{vfwg} \geq 0$ is defined. The penalties measure the interference caused by assigning two frequencies to the vertices. For $v$ and $w$, $\{v, w\} \in E$ if and only if at least one penalty $p_{vfwg} > 0$. In Koster et al. [85], a cutting-plane algorithm is shown to be effective only for $|F_v| \leq 6$. In practice, however, $|F_v| = 40$ on average. In Koster et al. [83, 87], a tree decomposition-based algorithm is developed for the problem. First, a tree decomposition is computed with the improvement heuristic described in §5.1.2. Next, the tree decomposition is used to run a dynamic programming algorithm to solve the problem. Several reduction techniques have been developed to keep the number of partial solutions to be maintained during the algorithm small. The algorithm is tested on frequency assignment problems that have been defined in the context of the CALMA project (see Aardal et al. [1, 2] for more information on the problems and overview of the results). It was indeed possible to solve 7 out of the 11 instances to optimality by this technique. For the other instances, the computer memory was exhausted before optimality of the best-known solution could be proven.

In Koster et al. [86] the algorithm is adapted to an interference lower-bound algorithm by considering subsets of the frequencies instead of the single frequencies. Step by step the

subsets are refined to improve the lower bound until either the best-known solution is proved to be optimal, or computer memory prohibits further computation.

In Koster et al. [87], this tree decomposition-based algorithm is discussed in the more general context of partial constraint satisfaction problems with binary relations. It is shown that the maximum satisfiability (MAX SAT) problem can be converted to a partial constraint satisfaction problem and computational results are presented for instances taken from the second DIMACS challenge on cliques, colorings, and satisfiability [78].

Other experimental work has been carried out for vertex covering and vertex coloring. Alber et al. [4] applied a tree decomposition-based algorithm for solving the vertex cover problem on planar graphs. Commandeur [46] experimented with an algorithm that solves the vertex coloring by first coloring the heaviest bag of a tree decomposition, and the remaining vertices afterward.

As already pointed out in the frequency assignment application, memory consumption is a major concern for tree decomposition-based algorithms. Recently, Betzler et al. [18] have proposed a technique for reducing the memory requirements of these algorithms.

Requests for computational assistance in the construction of tree decompositions for various graphs exemplify that applying treewidth approaches to various other combinatorial problems is gaining more and more interest in fields as different as bioinformatics, artificial intelligence, operations research, and (theoretical) computer science.

## 6. Branchwidth, Treewidth, and Matroids

### 6.1. Branchwidth of Matroids

It is only natural that branch decompositions can be extended to matroids. In fact, branch decompositions have been used to produce a matroid analogue of the graph minors theorem (Geelen et al. [60]). A formal definition for the branchwidth of a matroid is given below.

The reader is referred to the book by Oxley [98] if not familiar with matroid theory. Let $M$ be a matroid with finite ground set $S(M)$ and rank function $\rho$. The rank function of $M^*$, the dual of $M$, is denoted $\rho^*$.

A *separation* $(A, B)$ of a matroid $M$ is a pair of complementary subsets of $S(M)$, and the order of the separation, denoted $\rho(M, A, B)$, is defined to be following:

$$\rho(M, A, B) = \begin{cases} \rho(A) + \rho(B) - \rho(M) + 1 & \text{if } A \neq \varnothing \neq B, \\ 0 & \text{else,} \end{cases}$$

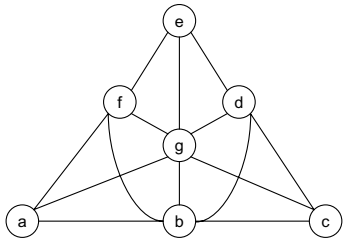
A *branch decomposition* of a matroid $M$ is a pair $(T, \mu)$ where $T$ is a tree having $|S(M)|$ leaves in which every nonleaf node has degree 3 and $\mu$ is a bijection from the ground set of $M$ to the leaves of $T$. Notice that removing an edge, say $e$, of $T$ partitions the leaves of $T$ and the ground set of $M$ into two subsets $A_e$ and $B_e$. The *order* of $e$ and of $(A_e, B_e)$, denoted $order(e)$ or $order(A_e, B_e)$, is equal to $\rho(M, A_e, B_e)$. The *width* of a branch decomposition $(T, \mu)$ is the maximum order of all edges in $T$. The *branchwidth* of $M$, denoted by $\beta(M)$, is the minimum width over all branch decompositions of $M$. A branch decomposition of $M$ is *optimal* if its width is equal to the branchwidth of $M$. For example, Figure 13 gives a Euclidean representation of a matroid and its optimal branch decomposition where all of the orders for the edges of the branch decomposition are provided.

Some results characterizing the branchwidth of matroids are given in the following lemma.

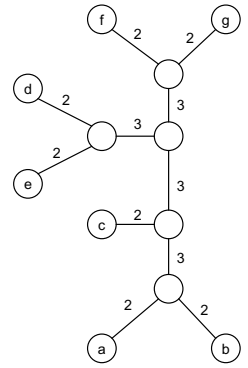
**Lemma 4 (Dharmatilake [52]).** *Let $M$ be a matroid. Then, $\beta(M) = \beta(M^*)$, and if $M'$ is a minor of $M$, then $\beta(M') \leq \beta(M)$.*

**Lemma 5 (Dharmatilake [52]).** *Let $M$ be a matroid. Then $\beta(M) \leq 1$ if and only if $M$ has no nonloop cycle. Moreover, $\beta(M) \leq 2$ if and only if $M$ is the cycle matroid of a series-parallel graph.*

FIGURE 13. Fano matroid $F_7$ with optimal branch decomposition $(T, \mu)$ of width 4.




(a) Euclidean representation of the Fano matroid




(b) Optical branch decomposition of the Fano matroid

The cycle matroid of graph $G$, denoted $M(G)$, has $E(G)$ as its ground set and the cycles of $G$ as the cycles of $M(G)$. For example, Figure 14 gives an optimal branch decomposition of the cycle matroid of the example graph given in Figure 1, where some of the orders for the edges of the branch decomposition are provided.

In addition, there is also the concept of matroid tangles, first offered by Dharmatilake [52]. Let $k$ be a positive integer, and let $M$ be a matroid. A *tangle* of order $k$ in $M$ is a set $\mathcal{T}$ of $<k$ separations of $M$ such that

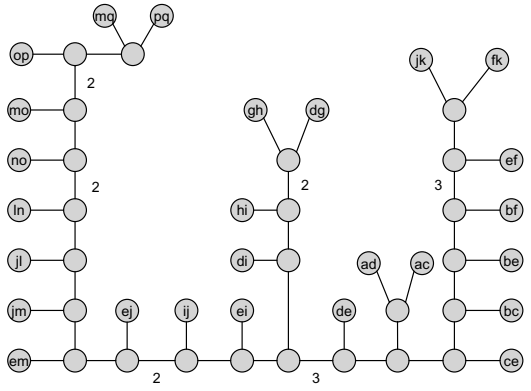
- for every separation $(A,B)$ of $M$ such that $\rho(M,A,B)<k$, either $(A,B)\in\mathcal{T}$ or $(B,A)\in\mathcal{T}$;
- if $(A_1,B_1)$, $(A_2,B_2)$, $(A_3,B_3)\in\mathcal{T}$, then $A_1\cup A_2\cup A_3 \neq S(M)$; and
- if $(A,B)\in\mathcal{T}$, then $\rho(A)<\rho(M)$.

They are referred to as the first, second, and third matroid tangle axioms. Likewise, the *tangle number* of a matroid $M$, denoted $\theta(M)$, is the largest order of a tangle for $M$. Notice that the third matroid tangle axiom offers instances where the tangle number could be infinity. This occurs when the matroid has a coloop because the coloop can always be on the second part of the separations of a tangle. In addition, Dharmatilake gave a min-max relationship between tangles of matroids and the branchwidth of matroids, given below.

**Theorem 10 (Dharmatilake [52]).** *Let $M$ be a matroid. Then $\beta(M)=\theta(M)$ if and only if $M$ has no coloop and $\beta(M)\neq 1$.*

It was conjectured by Geelen et al. [61] that the branchwidth of a graph and the branchwidth of the graph's cycle matroid are equal if the graph has a cycle of length of at least 2.

FIGURE 14. Optimal branch decomposition $(T, \mu)$ of width 3 for the cycle matroid of the graph in Figure 1.



Similar to the work on characterizing the classes of graphs with bounded branchwidth, there have been some developments in characterizing the classes of matroids with bounded branchwidth. Most notable is the work of Robertson and Seymour [106], characterizing the class of matroids with branchwidth of at most 2, and the work of Geelen et al. [60], proving that the size of excluded minors for the class of matroids with bounded branchwidth is finite. In addition, Hall et al. [67] showed that there are at most 14 excluded minors of the class of matroids with branchwidth of at most 3. Recently, Oum and Seymour [97] have derived a polynomial-time algorithm to approximate the branchwidth of any matroid, similar to the algorithm offered by Robertson and Seymour [108] for graphs. One is also referred to the work of Hliněný [76] for more detailed discussions on the branchwidth of matroids.

## 6.2. Treewidth of Matroids

In contrast to branchwidth, it is not straightforward to generalize the notion of treewidth to matroids. Where the width of a branch decomposition is defined on edges of the graph, the width of a tree decomposition is defined on graph vertex sets (the bags). The ground set of the cycle matroid corresponds with the edge set of the associated graph, whereas the vertices do not play a role in the cycle matroid.

Hliněný and Whittle [77], inspired by an unpublished work of Geelen, state a definition for treewidth of matroids and show that in the case of the cycle matroid of a graph this definition is equivalent to the graph-based definition, by providing an alternative characterization of the treewidth of graphs.

Let $M$ be a matroid on the ground set $S(M)$. A pair $(T, \mu)$, where $T$ is a tree and $\mu : S \to V(T)$ is an arbitrary mapping, is called a tree decomposition of $M$. For a node $x$ of $T$, denote the connected components of $T - x$ by $T_1, \ldots, T_d$, and set $F_i = \mu^{-1}(V(T_i))$. The *width of* $x$ is

$$\text{(MW)} \quad \sum_{i=1}^{d} \rho_M(S - F_i) - (d-1) \cdot \rho(M),$$

and the *width* of the decomposition $(T, \mu)$ is the maximum width over all nodes of $T$. The *treewidth* $\tau(M)$ of $M$ is the minimum width over all tree decompositions of $M$.

**Theorem 11 (Hliněný and Whittle [77]).** *Let $G$ be a graph with at least one edge, and let $M(G)$ be the cycle matroid of $G$. Then, $\tau(G) = \tau(M(G))$.*

Note that the mapping $\mu$ in the definition has nothing to do with the bags $X_i$ defined in the graph version of tree decompositions. The mapping $\mu$ guarantees that property (TD2) is fulfilled, whereas the interpolation property (TD3) is hidden in (MW). The width of a node $x$ of $T$ can be rewritten as

$$\rho(M) - \sum_{i=1}^{d} (\rho(M) - \rho_M(S - F_i)),$$

which can be seen as minimizing the rank defect of the branches of each node (here rank defect is defined as $\rho(M) - \rho_M(S - F)$ for a set $F \subseteq S$).

Besides the equivalence of graph and matroid treewidth, Hliněný and Whittle [77] also prove the relation between matroid branchwidth and matroid treewidth, similar to Theorem 2. They show that $\beta(M) \leq \tau(M) + 1 \leq \max(2\beta(M) - 1, 2)$ holds for all matroids $M$. Hence, all results for matroids of bounded branchwidth carry over to matroids of bounded treewidth and vice versa. Note that the upper bound is not as tight as in the case of graphs.

## 7. Open Problems

Related to branchwidth and treewidth, the following open questions are worth considering:

- Does there exist a polynomial-time algorithm to compute the branchwidth and optimal branch decomposition of a planar graph with complexity smaller than $O(n^3)$?
- Is the branchwidth of a graph and its cycle matroid equal if the graph has a cycle of length of at least 2?
- Do there exist polynomial-time algorithms to compute the branchwidth and optimal branch decomposition of graphs embedable on orientable surfaces other than the sphere, such as the torus or double torus?
- Do there exist (practical) integer-programming formulations for finding the branchwidth or treewidth of a graph?
- Does there exist a polynomial-time approximation algorithm for treewidth with constant approximation guarantee?
- Does there exist a polynomial-time algorithm for computing treewidth of a planar graph, or a proof that this problem is $\mathcal{NP}$-hard?
- Do heuristics other than MCS have a lower-bounding counterpart?
- How good can the contraction degeneracy $\delta C(G)$ be in general graphs? Is there a bound (different from treewidth) that limits this parameter?
- Do there exist computational efficient algorithms to compute the treewidth of general graphs? Are the exponential-time algorithms useful for practical computations?

Finally, the real application of the methodology to other combinatorial problems is of importance for both the operations research community and the algorithmic graph theory community. For operations researchers, it is good to have alternative methodologies other than integer linear programming that indeed solve real-world problems. For the algorithmic graph theorists, more applications will drive the research to improved lower and upper bounds for treewidth or related notions.

## 8. Acknowledgments

The authors would like to acknowledge the support of the National Science Foundation (Grant DMI-0217265), the DFG research group "Algorithms, Structure, Randomness" (Grant GR 883/9-3, GR 883/9-4), and the Netherlands Organization for Scientific Research (project "Treewidth and Combinatorial Optimization").

## References

[1] K. I. Aardal, C. A. J. Hurkens, J. K. Lenstra, and S. R. Tiourine. Algorithms for radio link frequency assignment: The CALMA project. *Operations Research* 50(6):968–980, 2003.

[2] K. I. Aardal, C. P. M. van Hoesel, A. M. C. A. Koster, C. Mannino, and A. Sassano. Models and solution techniques for the frequency assignment problem. *4OR* 1(4):261–317, 2003.

[3] J. Alber and R. Niedermeier. Improved tree decomposition based algorithms for domination-like problems. *Proceedings of the 5th Latin American Theoretical Informatics* (*LATIN 2002*), Lecture Notes in Computer Science, Vol. 2286. Springer-Verlag, Heidelberg, Germany, 613–627, 2002.

[4] J. Alber, F. Dorn, and R. Niedermeier. Experimental evaluation of a tree decomposition based algorithm for vertex cover on planar graphs. *Discrete Applied Mathematics* 145:210–219, 2004.

[5] M. Alekhnovich and A. Razborov. Satisfiability, branch-width and Tseitin tautologies. *43rd Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society, 593–603, 2002.

[6] N. Alon. Eigenvalues and expanders. *Combinatorica* 2:83–96, 1986.

[7] N. Alon, P. D. Seymour, and R. Thomas. Planar separators. *SIAM Journal on Discrete Mathematics* 7:184–193, 1994.

[8] C. Alvarez, R. Cases, J. Diaz, J. Petit, and M. Serna. Routing tree problems on random graphs. Technical Report LSI-01-10-R, Software Department, Universitat Politecnica de Catalunya, Barcelona, Spain, 2000.

[9] E. Amir. Efficient approximations for triangulation of minimum treewidth. *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, 7–15, 2001.

[10] D. Archdeacon. A Kuratowski theorem for the projective plane. Ph.D. thesis, Ohio State University, 1980.

[11] D. Archdeacon and P. Huneke. A Kuratowski theorem for non-orientable surfaces. *Journal of Combinatorial Theory Series B* 46(2):173–231, 1989.

[12] S. Arnborg and A. Proskurowski. Characterization and recognition of partial 3-trees. *SIAM Journal of Algebraic and Discrete Methods* 7:305–314, 1986.

[13] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a $k$-tree. *SIAM Journal of Algebraic and Discrete Methods* 8:277–284, 1987.

[14] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms* 12:308–340, 1991.

[15] S. Arnborg, A. Proskurowski, and D. Corneil. Forbidden minors characterization of partial 3-trees. *Discrete Mathematics* 80:1–19, 1990.

[16] E. Bachoore and H. L. Bodlaender. New upper bound heuristics for treewidth. S. Nikoletseas, ed. *WEA 2005: Workshop on Efficient and Experimental Algorithms*. Lecture Notes in Computer Science, Springer, 2005. Forthcoming.

[17] A. Berry, J. Blair, P. Heggernes, and B. Peyton. Maximum cardinality search for computing minimal triangulations of graphs. *Algorithmica* 39:287–298, 2004.

[18] N. Betzler, R. Niedermeier, and J. Uhlmann. Tree decompositions of graphs: Saving memory in dynamic programming. *CTW 2004: Cologne-Twente Workshop on Graphs and Combinatorial Optimization*, Villa Vigoni (CO), Italy, 56–60, 2004.

[19] D. Bienstock. Graph searching, path-width, tree-width and related problems (a survey). *Reliability of Computer and Communication Networks*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 5. American Mathematical Society, Providence, RI, 33–49, 1991.

[20] D. Bienstock and M. A. Langston. Algorithmic implications of the graph minor theorem. *Network Models, Handbook of Operations Research and Management Science*. North-Holland, Amsterdam, 481–502, 1995.

[21] D. Bienstock and N. Ozbay. Tree-width and the Sherali-Adams operator. *Discrete Optimization* 1:13–21, 2004.

[22] J. R. S. Blair, P. Heggernes, and J. Telle. A practical algorithm for making filled graphs minimal. *Theoretical Computer Science* 250:125–141, 2001.

[23] H. L. Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial $k$-trees. *Journal of Algorithms* 11:631–643, 1990.

[24] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica* 11:1–21, 1993.

[25] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing* 25:1305–1317, 1996.

[26] H. L. Bodlaender. A partial $k$-arboretum of graphs with bounded treewidth. *Theoretical Computer Science* 209:1–45, 1998.

[27] H. L. Bodlaender. Necessary edges in $k$-chordalizations of graphs. *Journal of Combinatorial Optimization* 7:283–290, 2003.

[28] H. L. Bodlaender. Treewidthlib. Working paper, Department of Computer Science, Utrecht University, Utrecht, The Netherlands, 2004.

[29] H. L. Bodlaender. Discovering treewidth. P. Vojtáš, M. Bieliková, B. Charron-Bost, and O. Sýkora, ed. *SOFSEM 2005: Theory and Practice of Computer Science*. Lecture Notes in Computer Science, Vol. 3381. Springer, 1–16, 2005.

[30] H. L. Bodlaender and F. Fomin. Equitable colorings of bounded treewidth graphs. Technical Report UU-CS-2004-010, Institute of Information and Computing Sciences, Utrecht University, The Netherlands, 2004.

[31] H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms* 21:358–402, 1996.

[32] H. L. Bodlaender and A. M. C. A. Koster. Safe separators for treewidth. *Proceedings 6th Workshop on Algorithm Engineering and Experiments ALENEX04*, 70–78, 2004.

[33] H. L. Bodlaender and D. Thilikos. Constructive linear time algorithms for branchwidth. P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, eds. *Proceedings of the 24th International Colloquium on Automata, Languages, and Programming*. Lecture Notes in Computer Science, Springer-Verlag, 627–637, 1997.

[34] H. L. Bodlaender and D. Thilikos. Graphs with branchwidth at most three. *Journal of Algorithms* 32:167–194, 1999.

[35] H. L. Bodlaender and B. van Antwerpen-de Fluiter. Parallel algorithms for series parallel graphs and graphs with treewidth two. *Algorithmica* 29:543–559, 2001.

[36] H. L. Bodlaender, A. Grigoriev, and A. M. C. A. Koster. Treewidth lower bounds with brambles. Working paper, 2005.

[37] H. L. Bodlaender, A. M. C. A. Koster, and T. Wolle. Contraction and treewidth lower bounds. S. Albers and T. Radzik, eds. *Proceedings 12th Annual European Symposium on Algorithms, ESA2004*. Lecture Notes in Computer Science, Vol. 3221. Springer, Berlin, Germany, 628–639, 2004.

[38] H. L. Bodlaender, A. M. C. A. Koster, and T. Wolle. Contraction and treewidth lower bounds. Technical Report UU-CS-2004-34, Department of Computer Science, Utrecht University, Utrecht, The Netherlands, 2004.

[39] H. L. Bodlaender, A. M. C. A. Koster, F. van den Eijkhof, and L. C. van der Gaag. Preprocessing for triangulation of probabilistic networks. J. Breese and D. Koller, eds. *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, 32–39, 2001.

[40] V. Bouchitté, F. Mazoit, and I. Todinca. Chordal embeddings of planar graphs. *Discrete Mathematics* 273:85–102, 2003.

[41] V. Bouchitté, D. Kratsch, H. Müller, and I. Todinca. On treewidth approximations. *Discrete Applied Mathematics* 136:183–196, 2004.

[42] G. Chang and D. Kuo. The $l(2,1)$-labeling problem on graphs. *SIAM Journal of Discrete Mathematics* 9:309–316, 1996.

[43] W. A. Christian. Linear-time algorithms for graphs with bounded branchwidth. Ph.D. thesis, Rice University, Houston, TX, 2003.

[44] F. Clautiaux, J. Carlier, A. Moukrim, and S. Négre. New lower and upper bounds for graph treewidth. J. D. P. Rolim, ed. *Proceedings International Workshop on Experimental and Efficient Algorithms, WEA 2003*. Lecture Notes in Computer Science, Vol. 2647. Springer-Verlag, 70–80, 2003.

[45] F. Clautiaux, A. Moukrim, S. Négre, and J. Carlier. Heuristic and meta-heuristic methods for computing graph treewidth. *RAIRO Operations Research* 38:13–26, 2004.

[46] M. Commandeur. Solving vertex coloring using tree decompositions. Master's thesis, Universiteit Maastricht, The Netherlands, 2004.

[47] W. Cook and P. D. Seymour. An algorithm for the ring-routing problem. Bellcore technical memorandum, Bellcore, Piscataway, NJ, 1994.

[48] W. Cook and P. D. Seymour. Tour merging via branch-decomposition. *INFORMS Journal on Computing* 15(3):233–248, 2003.

[49] B. Courcelle. Graph rewriting: An algebraic and logic approach. J. van Leeuwen, ed. *Handbook of Theoretical Computer Science*, Elsevier, New York, 194–242, 1990.

[50] B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation* 85:12–75, 1990.

[51] E. Dahlhaus. Minimal elimination ordering inside a given chordal graph. *Proceedings 23rd International Workshop on Graph-Theoretic Concepts in Computer Science WG'97*, Lecture Notes in Computer Science, Vol. 1335. Springer-Verlag, 132–143, 1997.

[52] J. S. Dharmatilake. A min-max theorem using matroid separations. *Contemporary Mathematics* 197:333–342, 1996.

[53] J. Fiala, P. Glovach, and J. Kratochvil. Distance constrained labelings of graphs of bounded treewidth. *Proceedings of 32nd International Colloquium on Autmata, Languages and Programming*, 2005. Forthcoming.

[54] F. Fomin and D. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. L. Caires et al., eds. *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*. Lecture Notes in Computer Sciences, Vol. 3580. Springer-Verlag, Berlin, Germany, 360–372, 2005.

[55] F. Fomin and D. Thilikos. A simple and fast approach for solving problems on planar graphs. *STACS 2004*, 2004.

[56] F. Fomin, P. Fraigniaud, and D. Thilikos. The price of connectedness in expansions. Technical Report LSI-04-28-R, Departament de Lenguatges i Sistemes Informàtics, Universitat Politécnica de Catalunya, Spain, 2004.

[57] F. V. Fomin, D. Kratsch, and I. Todinca. Exact (exponential) algorithms for treewidth and minimum fill-in. *Proceedings of the 31st International Colloquium on Automata, Languages and Programming*, 568–580, 2004.

[58] M. R. Garey, R. L. Graham, D. S. Johnson, and D. E. Knuth. Complexity results for bandwidth minimization. *SIAM Journal of Applied Mathematics* 34:477–495, 1978.

[59] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory Series B* 16:47–56, 1974.

[60] J. F. Geelen, A. M. H. Gerards, and G. Whittle. Branch width and well-quasi-ordering in matroids and graphs. *Journal of Combinatorial Theory Series B* 84:270–290, 2002.

[61] J. F. Geelen, A. M. H. Gerards, N. Robertson, and G. P. Whittle. On the excluded minors for the matroids of branch-width $k$. *Journal of Combinatorial Theory Series B* 88:261–265, 2003.

[62] H. Glover, P. Huneke, and C. S. Wang. 103 graphs that are irreducible for the projective plane. *Journal of Combinatorial Theory Series B* 27:332–370, 1979.

[63] V. Gogate and R. Dechter. A complete anytime algorithm for treewidth. *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence UAI-04*, UAI Press, Arlington, VA, 201–208, 2004.

[64] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[65] Q.-P. Gu and H. Tamaki. Optimal branch-decomposition of planar graphs in $O(n^3)$ time. Technical Report 2004-14, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, 2004.

[66] J. Gustedt, O. A. Mæhle, and J. A. Telle. The treewidth of Java programs. David M. Mount and Clifford Stein, eds. *Proceedings 4th International Workshop on Algorithm Engineering and Experiments*. Lecture Notes in Computer Science, Vol. 2409. Springer-Verlag, Berlin, Germany, 86–97, 2002.

[67] R. Hall, J. Oxley, C. Semple, and G. Whittle. On matroids of branch-width three. *Journal of Combinatorial Theory Series B* 86:148–171, 2002.

[68] P. Heggernes and Y. Villanger. Efficient implementation of a minimal triangulation algorithm. R. Möhring and R. Raman, eds. *Proceedings of the 10th Annual European Symposium on Algorithms, ESA'2002*. Lecture Notes in Computer Science, Vol. 2461. Springer-Verlag, 550–561, 2002.

[69] P. Heggernes, J. A. Telle, and Y. Villanger. Computing minimal triangulations in time $O(n^\alpha \log n) = o(n^{2.376})$. *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms SODA 2005*. SIAM, Philadelphia, PA, 905–916, 2005.

[70] I. V. Hicks. Branch decompositions and their applications. Ph.D. thesis, Rice University, Houston, TX, 2000.

[71] I. V. Hicks. Branchwidth heuristics. *Congressus Numerantium* 159:31–50, 2002.

[72] I. V. Hicks. Branch decompositions and minor containment. *Networks* 43(1):1–9, 2004.

[73] I. V. Hicks. Graphs, branchwidth, and tangles! Oh my! *Networks* 45(2):55–60, 2005.

[74] I. V. Hicks. Planar branch decompositions I: The ratcatcher. *INFORMS Journal on Computing*, 2005. Forthcoming.

[75] I. V. Hicks. Planar branch decompositions II: The cycle method. *INFORMS Journal on Computing*, 2005. Forthcoming.

[76] P. Hliněný. On the excluded minors for matroids of branch-width three. *Electronic Journal of Combinatorics* 9(Research Paper 32):13pp, 2002.

[77] P. Hliněný and G. Whittle. Matroid tree-width. Technical report, Technical University Ostrava, Ostrava, Czech Republic, 2003.

[78] D. S. Johnson and M. A. Trick, eds. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge* (October 11–13, 1993), AMS, Providence, RI, 1996.

[79] U. Kjærulff. Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing* 2:2–17, 1992.

[80] T. Kloks, J. Kratochvil, and H. Müller. New branchwidth territories. C. Meinel and S. Tison, eds. *STAC '99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 1999 Proceedings*, Springer-Verlag, Berlin, Germany, 173–183, 1999.

[81] T. Kloks, J. Kratochvil, and H. Müller. Computing the branchwidth of interval graphs. *Discrete Applied Mathematics* 145:266–275, 2005.

[82] E. Korach and N. Solel. Linear time algorithm for minimum weight Steiner tree in graphs with bounded treewidth. Technical Report 632, Computer Science Department, Technion—Israel Institute of Technology, Haifa, Israel, 1990.

[83] A. M. C. A. Koster. Frequency assignment—models and algorithms. Ph.D. thesis, University Maastricht, Maastricht, The Netherlands, 1999.

[84] A. M. C. A. Koster, H. L. Bodlaender, and S. P. M. van Hoesel. Treewidth: Computational experiments. H. Broersma, U. Faigle, J. Hurink, and S. Pickl, eds. *Electronic Notes in Discrete Mathematics*, Vol. 8. Elsevier, New York, 2001.

[85] A. M. C. A. Koster, C. P. M. van Hoesel, and A. W. J. Kolen. The partial constraint satisfaction problem: Facets and lifting theorems. *Operations Research Letters* 23(3–5):89–97, 1998.

[86] A. M. C. A. Koster, C. P. M. van Hoesel, and A. W. J. Kolen. Lower bounds for minimum interference frequency assignment problems. *Ricerca Operativa* 30(94–95):101–116, 2000.

[87] A. M. C. A. Koster, S. P. M. van Hoesel, and A. W. J. Kolen. Solving partial constraint satisfaction problems with tree decomposition. *Networks* 40:170–180, 2002.

[88] A. M. C. A. Koster, T. Wolle, and H. L. Bodlaender. Degree-based treewidth lower bounds. S. Nikoletseas, ed. *WEA 2005: Workshop on Efficient and Experimental Algorithms*. Lecture Notes in Computer Science, Vol. 3503. Springer-Verlag, Berlin, Germany, 101–112, 2005.

[89] J. Kruskal. Well-quasi-ordering, the tree theorem, and Vazsonyi's conjecture. *Transactions of the American Mathematical Society* 95:210–225, 1960.

[90] K. Kuratowski. Sur le probleme des courbes gauches en topologie. *Fundamenta Mathematicae* 15:271–283, 1930.

[91] D. Lapoire. Treewidth and duality for planar hypergraphs. Technical report, Laboratoire Bordelais de Recherche en Informatique, Talence cedex, France, 2002.

[92] P. Larrañaga, C. M. H. Kuijpers, M. Poza, and R. H. Murga. Decomposing Bayesian networks: triangulation of the moral graph with genetic algorithms. *Statistics and Computing (UK)* 7(1):19–34, 1997.

[93] S. J. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *The Journal of the Royal Statistical Society Series B (Methodological)* 50:157–224, 1988.

[94] B. Lucena. A new lower bound for tree-width using maximum cardinality search. *SIAM Journal of Discrete Mathematics* 16:345–353, 2003.

[95] B. Monien. The bandwidth minimization problem for caterpillars with hair length 3 is *NP*-complete. *SIAM Journal of Algebraic and Discrete Methods* 7:505–512, 1986.

[96] C. Nash-Williams. On well-quasi-ordering infinite trees. *Proceedings of the Cambridge Philosophical Society* 61:697–720, 1965.

[97] S.-I. Oum and P. D. Seymour. Approximating clique-width and branch-width. Technical report, Mathematics Department, Princeton University, 2005.

[98] J. G. Oxley. *Matroid Theory*. Oxford University Press, Oxford, UK, 1992.

[99] S. Ramachandramurthi. Algorithms for VLSI layout based on graph width metrics. Ph.D. thesis, Computer Science Department, University of Tennessee, Knoxville, TN, 1994.

[100] S. Ramachandramurthi. The structure and number of obstructions to treewidth. *SIAM Journal of Discrete Mathematics* 10:146–157, 1997.

[101] B. Reed. Tree width and tangles: A new connectivity measure and some applications. R. A. Bailey, ed. *Survey in Combinatorics, 1997*. Cambridge University Press, Cambridge, UK, 87–162, 1997.

[102] G. Reinelt. TSPLIB—A traveling salesman library. *ORSA Journal on Computing* 3:376–384, 1991.

[103] N. Robertson and P. D. Seymour. Graph minors: A survey. I. Anderson, ed. *Surveys in Combinatorics*. London Mathematical Society Lecture Note Series, Vol. 103. Cambridge University Press, Cambridge, UK, 153–171, 1985.

[104] N. Robertson and P. D. Seymour. Graph minors, II. Algorithmic aspects of tree-width. *Journal of Algorithms* 7:309–322, 1986.

[105] N. Robertson and P. D. Seymour. Graph minors, IV: Treewidth and well-quasi-ordering. *Journal of Combinatorial Theory Series B* 48:227–254, 1990.

[106] N. Robertson and P. D. Seymour. Graph minors, X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory Series B* 52:153–190, 1991.

[107] N. Robertson and P. D. Seymour. Graph minors, XI: Circuits on a surface. *Journal of Combinatorial Theory Series B* 60:72–106, 1994.

[108] N. Robertson and P. D. Seymour. Graph minors, XIII: The disjoint paths problem. *Journal of Combinatorial Theory Series B* 63:65–110, 1995.

[109] N. Robertson and P. D. Seymour. Graph minors, XX: Wagner's conjecture. *Journal of Combinatorial Theory Series B* 92(2):325–357, 2004.

[110] H. Röhrig. Tree decomposition: A feasibility study. Master's thesis, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1998.

[111] D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing* 5:266–283, 1976.

[112] D. Sanders. On linear recognition for tree-width at most four. *SIAM Journal of Discrete Mathematics* 9:101–117, 1996.

[113] A. Satyanarayana and L. Tung. A characterization of partial 3-trees. *Networks* 20:299–322, 1990.

[114] P. Scheffler. Die Baumweite von Graphen als ein Maß für die Kompliziertheit algorithmischer Probleme. Ph.D. thesis, Akademie der Wissenschaften der DDR, Berlin, Germany, 1989.

[115] P. D. Seymour and R. Thomas. Graph searching and a minmax theorem for tree-width. *Journal of Combinatorial Theory Series B* 58:239–257, 1993.

[116] P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica* 14(2):217–241, 1994.

[117] K. Shoikhet and D. Geiger. A practical algorithm for finding optimal triangulations. *Proceedings National Conference on Artificial Intelligence (AAAI '97)*, Morgan Kaufmann, 185–190, 1997.

[118] G. Szekeres and H. S. Wilf. An inequality for the chromatic number of a graph. *Journal of Combinatorial Theory* 4:1–3, 1968.

[119] H. Tamaki. A linear time heuristic for the branch-decomposition of planar graphs. Technical Report MPI-I-2003-1-010, Max-Planck-Institut für Informatik, 2003.

[120] R. E. Tarjan and M. Yannakakis. Simple linear time algorithms to test chordiality of graphs, test acyclicity of graphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing* 13:566–579, 1984.

[121] M. Thorup. All structured programs have small tree width and good register allocation. *Information and Computation* 142:159–181, 1998.

[122] Z. Tuza. Strong branchwidth and local transversals. *Discrete Applied Mathematics* 145:291–296, 2005.

[123] K. Wagner. Über eine Eigenschaft der ebenen Komplexe. *Mathematische Annalen* 115:570–590, 1937.

[124] G. J. Woeginger. Exact algorithms for *NP*-hard problems: A survey. *Combinatorial Optimization: "Eureka, You Shrink!"* Lecture Notes in Computer Science, Vol. 2570. Springer, Berlin, Germany, 185–207, 2003.

[125] T. Wolle, A. M. C. A. Koster, and H. L. Bodlaender. A note on contraction degeneracy. Technical Report UU-CS-2004-042, Institute of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, 2004.

[126] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal of Algebraic and Discrete Methods* 2:77–79, 1981.

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS | ISBN 1-877640-21-2
DOI 10.1287/educ.1053.0016

# Stochastic Programming: Optimization When Uncertainty Matters

***Julia L. Higle***
Department of Systems and Industrial Engineering, University of Arizona, Tucson, Arizona 85721, julie@sie.arizona.edu

**Abstract** Stochastic programming (SP) was first introduced by George Dantzig in the 1950s. Since that time, tremendous progress has been made toward an understanding of properties of SP models and the design of algorithmic approaches for solving them. As a result, SP is gaining recognition as a viable approach for large-scale models of decisions under uncertainty. In this paper, we present an introduction to stochastic programming models and methodology at a level that is intended to be accessible to the breadth of members within the INFORMS community.

**Keywords** stochastic programming; recourse models; decomposition methods

## 1. Introduction

Stochastic programming traces its roots to Dantzig [11], where the recourse model is first introduced, and Charnes and Cooper [9], where probabilistically constrained models are introduced. In many ways, a stochastic program (SP) might be thought of as one of the ultimate operations research models. When some of the data elements in a linear program (LP) are most appropriately described using random variables, a stochastic linear program (SLP) results. In this sense, stochastic programs involve an artful blend of traditional (deterministic) mathematical programs and stochastic models. In many cases, solution techniques for stochastic programs rely on statistical estimation and numerical approximation methods. In short, stochastic programming draws upon a wide variety of traditional operations research techniques.

In this paper, we provide a tutorial-level presentation of stochastic programming. Our goal is to provide an overview of the subject that is accessible to most members of the operations research community, with a particular focus on some of the details and terminology that potential users of SP methodology will want to know. We begin with an example illustrating why an SP model might be preferred to a deterministic model. Following that, we identify some of the essential features of a stochastic program that can be exploited by a solution method. Our focus in this paper will be on the class of recourse problems rather than the probabilistically constrained problems. This is primarily because the two classes of problems involve considerably different modeling and solution approaches. For a comprehensive review of probabilistic constraints, the reader is referred to Prékopa [36].

## 2. Does Uncertainty Matter?

As a general rule, stochastic programs are more difficult to formulate and solve than deterministic mathematical programs. Given the availability of postoptimality analysis, it can be tempting to ease the process by relying on sensitivity analysis to investigate the impact of uncertainty. In many cases, sensitivity analysis is not an appropriate tool for this investigation, as we shall see in the example that follows.

## 2.1. Sensitivity Analysis

We begin with a simple linear programming problem, which is a variation of a problem that appears in Winston and Venkataramanan [49], and also appears in Higle and Wallace [24].

The Dakota Furniture Company manufactures desks, tables, and chairs. The manufacture of each type of furniture requires lumber and two types of skilled labor: finishing and carpentry. The cost of each resource and the amount needed to make each type of furniture is given in Table 1.

TABLE 1. Dakota production/resource data.

| Resource | Cost ($) | Desk | Table | Chair |
|---|---|---|---|---|
| Lumber (bd. ft.) | 2 | 8 | 6 | 1 |
| Finishing (hrs.) | 4 | 4 | 2 | 1.5 |
| Carpentry (hrs.) | 5.2 | 2 | 1.5 | 0.5 |
| Demand | | 150 | 125 | 300 |

A desk sells for $60, a table sells for $40, and a chair sells for $10. In order to maximize total profit, how much of each item should be produced, and what are the corresponding resource requirements?

There are many ways to determine the production quantities. Perhaps the easiest method is a simple per-item profit analysis. A desk costs $42.40 to produce and sells for $60, for a profit of $17.60 per desk sold. Similarly, a table costs $27.80 to produce and sells for $40, for a profit of $12.20 per table sold. On the other hand, a chair costs $10.60 to produce, and sells for only $10, so that each chair sold represents a net loss of $0.60. Given the objective of maximizing profit, the information provided suggests that Dakota should meet the demand for desks and tables, but decline to produce any chairs, as indicated in Table 2. To determine the resource requirements, we only need to work with the data provided in Table 1. That is, based on the resource requirements per item, production of 150 desks and 125 tables requires the acquisition of 1,950 bd. feet of lumber, 850 labor hours for finishing, and 487.5 labor hours for carpentry. Based on these production and resource quantities, Dakota anticipates a profit of $4,165.

This problem does not require the power of linear programming for solution. Nonetheless, LP models are easily derived. For example, if we define

$x_b$ = number of bd. feet of lumber acquired
$x_f$ = number of labor hours acquired for finishing
$x_c$ = number of labor hours acquired for carpentry
$y_d$ = number of desks produced
$y_t$ = number of tables produced
$y_c$ = number of chairs produced

the problem can be formulated as follows:

$$\begin{aligned} \text{Max} \quad & 60y_d + 40y_t + 10y_c - 2x_b - 4x_f - 5.2x_c \\ \text{s.t.} \quad & y_d \le 150 \\ & y_t \le 125 \\ & y_c \le 300 \\ & 8y_d + 6y_t + y_c - x_b \le 0 \\ & 4y_d + 2y_t + 1.5y_c - x_f \le 0 \\ & 2y_d + 1.5y_t + 0.5y_c - x_c \le 0 \\ & y_d, y_t, y_c, x_b, x_f, x_c \ge 0. \end{aligned} \tag{1}$$

Of course, the solution to this problem is identical to the one presented in Table 2.

Table 2. Dakota production/resource solution.

| Item | Production cost ($) | Selling price ($) | Profit per item ($) | Demand | Production |
|---|---|---|---|---|---|
| Desk | 42.40 | 60.00 | 17.60 | 150 | 150 |
| Table | 27.80 | 40.00 | 12.20 | 125 | 125 |
| Chair | 10.60 | 10.00 | −0.60 | 300 | 0 |

Notice that even though the basic data might change, the structure of the problem remains the same. For example, if the selling price for a chair is increased from $10 to $11, it is only necessary to change the appropriate objective function coefficient in the model that is being used. This is the observation that gives rise to the postoptimality investigation of the solution known as *sensitivity analysis*. That is, knowing that the basic structure of the model remains the same, we can investigate the manner in which changes in individual data elements would impact an optimal solution. Notice that if nothing else were to change, and the selling price of a chair increased to $11, the production of chairs would become profitable, and the nature of the solution would change considerably. On the other hand, if the price of the chair were to remain at $10 but the demand for chairs changed from 300 to 200, there would be no impact on the nature of the solution.

Sensitivity analysis is used in an attempt to study the robustness of the solution to a linear programming model. If there is cause for concern regarding the accuracy of the data used, sensitivity analysis is undertaken to determine the manner in which the solution might change if the data were different. When the solution does not change (or when the nature of the solution does not change, as when the basis remains optimal), one breathes a sigh of relief, believing that the proposed solution is appropriate. Unfortunately, no such relief is available if the solution is sensitive to the data. How should you proceed if the solution, or the nature of the solution, varies when the data is changed? In the Dakota example, we saw that a small change in the selling price of a chair took the solution from one extreme, in which no chairs were produced, to the other, in which 300 chairs were produced! Surely, this would be cause for concern and a clear indication that additional effort should be expended toward a better understanding of the selling price of chairs.

Although sensitivity analysis offers some sense of security, it is important to recognize that, in many cases, this is really a false sense of security. If there is some uncertainty about the values of some data elements, it should be included in the model. How do we model problems when we *know* that some of the data elements are difficult to predict or estimate? This is precisely the situation for which stochastic programming models are intended.

## 2.2. Introducing Uncertainty to the LP Model

As a modeling tool, linear programming is intended for deterministic problems. That is, when we formulate a linear programming model, we act as if all data elements are known quantities. For example, in the Dakota example, Table 1 provides everything we need to know about each product's resource requirements. The demand for each product is specified, as are the per-unit selling prices and the per-unit resource costs. It seems a bit presumptuous to assume that we know *all* of these values! Any or all of them may be subject to uncertainty.

It is probably reasonable to assume that Dakota has a good handle on the resource requirements for each of the three items that they produce. However, the selling prices, resource costs, and product demands might all be reasonably assumed to be subject to uncertainty. We have already seen that the solution to the LP model is sensitive to the objective coefficients. To illustrate the power of stochastic programming as a modeling paradigm, we will focus on the demand, and explore the impact of uncertainty in these values. That is, suppose that for each product we are given three potential demand scenarios, which we identify as being the "low," "most likely," and "high" values, as indicated in Table 3.

TABLE 3. Dakota demand scenarios.

| | Low value | Most likely value | High value |
|---|---|---|---|
| Desks | 50 | 150 | 250 |
| Tables | 20 | 110 | 250 |
| Chairs | 200 | 225 | 500 |
| Probability | 0.3 | 0.4 | 0.3 |

Notationally, we will denote the probability that the set of low values occurs as $p_\ell$, and let $p_m$ and $p_h$ be similarly defined for the most likely and high values, respectively. That is, $p_\ell = 0.3$, $p_m = 0.4$, and $p_h = 0.3$. Notice that the collection of demand scenarios and the corresponding probabilities form a multivariate probability distribution that can be used to describe the demand that Dakota might face in the future.

Now that we have a basic understanding of the deterministic model, and we've been given a collection of demand scenarios and their corresponding probabilities, what should we do next? When faced with this question, most people seem to come up with one or both of the following suggestions:

- Use the *expected demand* and proceed.
- Solve the problem for *each* of the demand scenarios, and somehow combine the solutions.

If we follow this course, we first note that the expected demand is precisely the demand that we used initially: 150 desks, 125 tables, and 300 chairs. Because we have already solved the mean value problem, let us see what happens if we solve the scenario problems. Notice that this merely requires changing the right-hand-side values of the demand constraints in the original problem. The output of this exercise is summarized in Table 4.

There are many observations that one might make regarding Table 4. First, notice that *all* cases indicate that chairs should not be produced. This is to be expected, given our earlier comments regarding the profitability of the various items. Notice that the basic structure of the LP that we have used has not changed, and it still costs more to produce a chair than the revenue that we receive from the sale of a chair. As before, chairs are not profitable, and so none are produced. Next, notice that the resource and production quantities in the first column are simply the expected values of the resource and production quantities in each of the individual scenario columns. Again, this can be traced directly back to the original LP model. Recall that sensitivity analysis of the solution of this LP indicates that the same basis will be optimal for all of these demand configurations. In that case, it is easy to show that this situation must arise (i.e., the expected value of the optimal solutions obtained from the individual scenarios is an optimal solution to the expected demand problem[1]).

These two observations may offer some consolation. The basic structure of all of the solutions is the same—no chairs are produced. Moreover, there is a direct relationship between the collection of scenario solutions and the solution to the problem in which expected values are used. This would seem to suggest that if they are interested in maximizing the expected profit, Dakota should simply use the solution that appears in the first column.

Or should they? This solution would have Dakota produce 150 desks and 125 tables, and purchase exactly enough resource to accomplish this task. Notice that if 150 desks are produced, there is a 30% chance that Dakota will produce more than they are able to sell. Similarly, if 125 tables are produced, there is a 70% chance Dakota will produce more than they are able to sell. Notice that if Dakota plans for the expected demand, and the demand turns out to be low, there will be 100 desks and 105 tables produced that will not be sold.

[1] In the standard notation of linear programming, if $\tilde{b}$ is a right-hand-side vector in which some elements are random variables, and the same basis $B$ is optimal for all possible values of $\tilde{b}$, then $E[B^{-1}\tilde{b}] = B^{-1}E[\tilde{b}]$ is an optimal solution to the problem in which the random vector is replaced by its expected value.

TABLE 4. Output for individual demand scenarios.

| | Demand | | | |
|---|---|---|---|---|
| | Expected | Low | Most likely | High |
| Resource quantities | | | | |
| Lumber (bd. ft.) | 1,950 | 520 | 1,860 | 3,500 |
| Finishing labor (hours) | 850 | 240 | 820 | 1500 |
| Carpentry labor (hours) | 487.5 | 130 | 465 | 875 |
| Production Quantities | | | | |
| Desks | 150 | 50 | 150 | 250 |
| Tables | 125 | 20 | 110 | 250 |
| Chairs | 0 | 0 | 0 | 0 |
| Profit ($) | 4,165 | 1,124 | 3,982 | 7,450 |

Instead of the $4,165 profit that the model suggests they should expect, they will realize a net *loss* of $6,035... and there is a 30% chance that this painful situation will occur![2]

Why does this happen? This is exactly the result of the false sense of security that sensitivity analysis provides. The solutions that we have looked at so far are, on an individual basis, best for one particular demand scenario (i.e., expected, low, etc.). As Dakota embarks on this planning adventure, they do not know what the demand will be, and it is important to obtain a solution that balances the impact of the various scenarios that might be encountered. Because the model that we have used looks at only one demand scenario at a time, it is not able to provide this type of balance. We need a model that explicitly considers the various scenarios... not individually, but collectively. This is precisely the need that stochastic programming models fulfill.

## 2.3. A Variety of Modeling Choices

In the Dakota problem, there are two types of decisions that are made:

- Production: How many desks, tables, and chairs should be produced?
- Resource: How much lumber, finishing labor, and carpentry labor should be procured?

Between Tables 1 and 3, we have a full description of the data required to develop a more robust model of Dakota's problem. However, there is still a critical piece of information that has not been specified—When do the various decisions have to be made? In complicated problems, decisions and uncertainty are often interwoven over time. That is, we make a decision, we observe what happens, more decisions are made, we observe again, etc. Notice that neither the mean value nor the scenario problems are able to capture this interplay between decisions and uncertainty. Before a stochastic programming model can be developed, the timing of the decisions, relative to the resolution of uncertainty, must be specified.

It is important to recognize that the timing of the decisions is a property of the problem at hand. That is, the modeler must determine when the decisions are made, relative to the resolution of uncertainty. Decisions that can be delayed until after information about the uncertain data is available offer an opportunity to adjust or adapt to the information that is received. We often refer to this adaptation as *recourse*. Models in which some decisions are delayed until after some information about the uncertain data has been obtained are known as recourse models, and decision variables that are permitted to vary with the scenario are sometimes referred to as recourse variables. Decisions either can or cannot be delayed—this is typically beyond our control. However, when it is possible to do so, there is generally value associated with delaying a decision until after additional information is obtained.

[2] To see this, note that the resources will cost $9,835, but the income from the sale of 50 desks and 20 tables is only $3,800.

## 2.4. A Recourse Model

Recourse models result when some of the decisions must be fixed before information relevant to the uncertainties is available, while some of them can be delayed until afterward. In the Dakota problem, suppose that the resource quantities must be determined fairly early, but production quantities can be delayed until after the demand is known. In some sense, we may think of the production quantities as being *flexible* or *adaptive*, while the resource quantities are not.

To model the production quantities as being dependent on the demand scenario, the production variables that have been used thus far, $(y_d, y_t, y_c)$, will be replaced by $\{(y_{d,\omega}, y_{t,\omega}, y_{c,\omega})\}_{\omega\in\{l,m,h\}}$. For example, the variable $y_{t,h}$ will denote the number of tables that will be produced if the demand is "high," and the other variables are similarly defined. On the other hand, the resource variables do not vary with the scenario, and thus will remain as $x_b, x_l,$ and $x_c$. Notice that in this case, because production quantities are being determined *after* the demand scenario is known, Dakota will not run the risk of producing items that it is not able to sell.

To draw attention to the structure of the decision problem, we may state this version of Dakota's problem as follows:

$$\begin{aligned} \text{Max} \quad & -2x_b - 4x_f - 5.2x_c + E[h(x,\tilde{d})] \\ \text{s.t.} \quad & x_b, x_f, x_c \geq 0, \end{aligned} \tag{2}$$

where $\tilde{d} = \{\tilde{d}_d, \tilde{d}_t, \tilde{d}_c\}$ represents the uncertain demand for desks, tables, and chairs, and for each scenario $\omega \in \{l, m, h\}$,

$$\begin{aligned} h(x, d_\omega) = \text{Max} \quad & 60y_{d,\omega} + 40y_{t,\omega} + 10y_{c,\omega} \\ \text{s.t.} \quad & 8y_{d,\omega} + 6y_{t,\omega} + y_{c,\omega} \leq x_b \\ & 4y_{d,\omega} + 2y_{t,\omega} + 1.5y_{c,\omega} \leq x_f \\ & 2y_{d,\omega} + 1.5y_{t,\omega} + 0.5y_{c,\omega} \leq x_c \\ & y_{d,\omega} \leq d_{d,\omega} \\ & y_{t,\omega} \leq d_{t,\omega} \\ & y_{c,\omega} \leq d_{c,\omega} \\ & y_{d,\omega}, y_{t,\omega}, y_{c,\omega} \geq 0. \end{aligned}$$

Notice that in this case, the subproblem determines the optimal allocation of resources to products after the demand for products is known. Note also that the resource quantities $(x_b, x_f, x_c)$ are inputs to the subproblem and appear on the right-hand side of the constraints. For the sake of completeness, we present the problem in nondecomposed form as follows:

$$\begin{aligned} \text{Max} \quad & -2x_b - 4x_f - 5.2x_c + \sum_{\omega\in\{l,m,h\}} p_\omega * \{60y_{d,\omega} + 40y_{t,\omega} + 10y_{c,\omega}\} \\ \text{s.t.} \quad & -x_b + 8y_{d,\omega} + 6y_{t,\omega} + y_{c,\omega} \leq 0 \\ & -x_f + 4y_{d,\omega} + 2y_{t,\omega} + 1.5y_{c,\omega} \leq 0 \\ & -x_c + 2y_{d,\omega} + 1.5y_{t,\omega} + 0.5y_{c,\omega} \leq 0 \\ & y_{d,l} \leq 50 \\ & y_{d,m} \leq 150 \\ & y_{d,h} \leq 250 \end{aligned}$$

$$y_{t,l} \le 20$$
$$y_{t,m} \le 110$$
$$y_{t,h} \le 250$$
$$y_{c,l} \le 200$$
$$y_{c,m} \le 225$$
$$y_{c,h} \le 500$$
$$x_b, x_f, x_c, y_{i,\omega} \ge 0 \quad \forall\, i \in \{d, t, ch\}, \omega \in \{l, m, h\}.$$

So far, we have looked at three different models of the Dakota problem: one in which the random variables are replaced by their expected values, one in which each scenario is considered separately, and the recourse model. To facilitate comparisons and discussions, we summarize the output from each of these models in Table 5.

There are several interesting observations that Table 5 affords. Notice that in terms of the resource quantities, it is difficult to identify a meaningful connection between the solutions to the scenario problems and the solution to the recourse problem. This contrasts sharply with the solution to the mean value problem, which we have already noted is just the expected value of the individual scenario solutions.

The objective values that are reported in Table 5 are especially confusing. These values range from 1,124 for the low-demand scenario to 7,450 for the high-demand scenario. As a result, a naive modeler might be tempted to look favorably on the high-demand scenario. Recognizing that this scenario will occur with probability 0.3, it is important that the objective values that appear in Table 5 be understood within the context from which they are obtained. For example, the scenario solutions are based on the assumption of perfect information. If we knew for sure that demand would be low, Dakota could obtain a maximum profit of 1,124. Similarly, if we knew for sure that demand would be high, Dakota could obtain a maximum profit of 7,450. The scenario problems consider only one scenario at a time. They do not consider the impact of planning for one scenario and having another scenario materialize. Similarly, the mean value solution plans only for the average demand, and does not consider its impact under the various scenarios. Only the recourse model is designed to encourage a solution that balances the potential impact of the various scenarios. That is, the objective values for the mean value problem and the scenario problems are unrealistically optimistic—they do not include considerations of the impact of variability in

Table 5. Output for Dakota models.

| | | Scenario solutions | | | | | |
|---|---|---|---|---|---|---|---|
| | Mean value | low | m. likely | high | Recourse | | |
| Resource quantities | | | | | | | |
| Lumber (bd. ft.) | 1,950 | 520 | 1,860 | 3,500 | 1,300 | | |
| Fin. labor (hrs.) | 850 | 240 | 820 | 1,500 | 540 | | |
| Car. labor (hrs.) | 487.5 | 130 | 465 | 875 | 325 | | |
| | | | | | Demand | | |
| | | | | | low | m. likely | high |
| Production quantities | | | | | | | |
| Desks | 150 | 50 | 150 | 250 | 50 | 80 | 80 |
| Tables | 125 | 20 | 110 | 250 | 20 | 110 | 110 |
| Chairs | 0 | 0 | 0 | 0 | 200 | 0 | 0 |
| Obj. value | 4,165 | 1,124 | 3,982 | 7,450 | 1,730 | | |

TABLE 6. Expected profit associated with scenario solutions.

| Demand scenario | Profit suggested by scenario problem | Expected profit |
|---|---|---|
| Expected value | 4,165 | 1,545 |
| Low | 1,124 | 1,124 |
| Most likely | 3,982 | 1,702 |
| High | 7,450 | −2,050 |

*Recall that the optimal expected profit is 1,730.

the demand. Consequently, they fail to provide a meaningful representation of the expected profit.

Unlike the mean value and scenario problems, the objective values of the recourse problems correspond to *expected profit*. The flexibility that the recourse model affords is nicely illustrated by the structure of the solution. Notice that it is the only model that suggests that chairs should be made, and then only when the demand for Dakota's products is low. The solution to the recourse model purchases sufficient resources to satisfy demand at some of the higher levels. That is, enough lumber and labor are procured to enable Dakota to address, at least partially, the higher-demand scenarios. If it turns out that the demand is low, Dakota is able to recover much of their resource expense by producing chairs. In the low-demand scenario, some of the lumber and carpentry labor will go unused, but Dakota will make maximum use of the resources that they have purchased. In all cases, Dakota will be able to sell all items produced. Thus, we see that, in reality, chairs are useful as a fallback item. When demand is low, Dakota can cut its losses by producing and selling chairs, which is preferable to producing and *not* selling desks and tables!

We have already noted that the objective values associated with the scenario problems are overly optimistic. These problems are only appropriate if one truly has the ability to adapt fully to the demand once it is known. If they are used inappropriately (e.g., if demand is uncertain at the time that the resource decisions must be made), a mismatch between production and demand is likely to result, and the actual objective values will differ substantially from those that are suggested! In Table 6, we compare the objective value associated with the individual scenario problems and the corresponding expected profits in the recourse setting. Notice the extent to which the scenario objective values overestimate the expected profits.

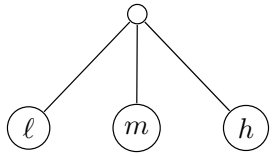
## 3. Recourse Problems

The formulation produced in §2.4 is known as a recourse problem. The term "recourse" refers to the opportunity to adapt a solution to the specific outcome observed. That is, $x$ corresponds to the resource levels to be acquired, and $\omega$ corresponds to a specific data scenario—in this case, the actual demands for the various products. The decision $y$ adapts to the specific combination of $x$ and $\omega$ obtained. In the event that the initial decision $x$ is coupled with a bad outcome, the variable $y$ offers an opportunity to recover to the fullest extent possible. Recourse problems are always presented as problems in which there are two or more decision stages.

### 3.1. Components of a Recourse Problem

Each recourse problem can be characterized by its scenario tree, its scenario problems, and its nonanticipativity constraints. A *scenario* is one specific, complete realization of the stochastic elements that might appear during the course of the problem (e.g., a possible sequence of demand over the duration of the problem). The *scenario tree* is a structured

FIGURE 1. Scenario tree for the Dakota problem.




distributional representation of the stochastic elements and the manner in which they may evolve over the period of time represented in the problem. A *scenario problem* is associated with a particular scenario and may be looked upon as a deterministic optimization problem (such as those whose solutions are displayed in Table 4). The scenario tree for the problem presented in §2.4 is very simple, involving only a root node and the three individual demand scenarios that are considered. We include it in Figure 1. In general, the nodes in the scenario tree correspond to a model of the manner in which information might become available over time. Although they are not typically depicted as such, nodes in a scenario tree are associated with a subset of decisions. The root node corresponds to the initial decision stage, during which no specific information regarding the random variables has been obtained. The leaf nodes correspond to the final decisions required, which are made after all available information has been obtained. In the Dakota problem, the resource decisions are associated with the root node while the production decisions are associated with the leaf nodes.

Depending on the manner in which the problem is formulated, it may be necessary to include specific conditions to ensure that the decision sequence honors the information structure associated with the scenario tree. These conditions are known as the *nonanticipativity constraints*, and impose the condition that scenarios that share the same history (of information) until a particular decision epoch should also make the same decisions. In reality, the nonanticipativity constraints ensure that the solutions obtained are implementable, i.e., that actions that must be taken at a specific point in time depend only on information that is available at that time. For that reason, the terms nonanticipativity and implementability are sometimes used interchangeably. These nonanticipativity constraints, which are derived from the scenario tree, are a distinguishing characteristic of stochastic programs—solution methods are typically designed to exploit their structure. The recourse formulation used to describe the Dakota problem implicitly honors the information structure. This is because the resource variables (which are associated with the root node in the scenario tree) do not vary with the scenario, while the production variables (which are associated with the leaf nodes in the scenario tree) are permitted to vary with the scenario. In §3.7 we will discuss formulations in which nonanticipativity constraints are specifically included.

Throughout this paper, $\tilde{\omega}$ will represent the uncertain data associated with the decision problem. In this sense, $\tilde{\omega}$ is a random variable defined on a probability space $(\Omega, \mathcal{A}, \mathcal{P})$. When $\tilde{\omega}$ is a discrete random variable, $p_\omega = \mathcal{P}\{\tilde{\omega} = \omega\}$ for each scenario $\omega \in \Omega$. Decision variables, matrices, etc, are defined in the same manner as for traditional linear programming problems. However, elements that might vary with the scenario will be identified by a subscript $\omega$. For example, a set of constraints

$$T_\omega x + W_\omega y_\omega = r_\omega$$

would indicate that the matrices ($T_\omega$ and $W_\omega$) and right-hand-side vector ($r_\omega$) contain elements that are specific to the scenario $\omega$. Additionally, the vector $y_\omega$ might vary with the specific scenario, while the vector $x$ cannot.

Finally, it is important to note that, in general, *randomness* within the data tends to be very sparse. That is, more often than not only a small number of elements in the matrices and vectors that are subject to uncertainty are represented as random variables—most elements are typically fixed. For example, in a max-flow problem, it may be the case that

arc capacities are modeled as random variables. However, the right-hand sides in flow conservation constraints remain fixed at zero and the data used to describe the flows (i.e., the incidence matrix) are similarly fixed.

### 3.2. Two-Stage Recourse Problems

The SLP formulated in §2.4 is an example of a two-stage problem, which has the general form:

$$\begin{aligned} &\text{Min} && cx + E[h(x,\tilde{\omega})] \\ &\text{s.t.} && Ax \geq b \\ & && x \geq 0 \end{aligned} \tag{3}$$

$$\text{where} \quad h(x,\omega) = \begin{aligned} &\text{Min} && g_\omega y \\ &\text{s.t.} && W_\omega y \geq r_\omega - T_\omega x \\ & && y \geq 0. \end{aligned} \tag{4}$$

In this case, $x$ is known as a first-stage decision (also known as the *here and now* decision). Note that in this model, $x$ does not respond to $\omega$—it is effectively determined before any information regarding the uncertain data has been obtained. On the other hand, $y$—the second-stage variable—is determined after observations regarding $\tilde{\omega}$ have been obtained. The problem in (4) is known variously as the second-stage problem, the subproblem, or the recourse subproblem. In essence, the goal of a two-stage model is to identify a first-stage solution that is well positioned against all possible observations of $\tilde{\omega}$. An optimal solution will tend to have the property that $x$ leaves the second-stage decision in a position to exploit advantageous outcomes of $\tilde{\omega}$ without excessive vulnerability to disadvantageous outcomes. Note that in this case, careful attention to the specific outcomes used to model the uncertainty is necessary. An overly coarse model may result in a failure to adequately represent outcomes that should influence the first-stage decision, leaving the second stage in an unmodeled state of vulnerability. An excessively fine model may result in increased computational burden.

As presented, without assuming any additional properties or structure on (4), we would describe the problem as having *general recourse.* In many cases, there is specific structure in the recourse subproblem that can be exploited for computational advantage. Some of the more commonly occurring types of structure are described in the following sections.

### 3.3. Simple Recourse

A special case of the recourse model, known as the simple recourse model, arises when the constraint coefficients in the second-stage problem, $W$, form an identity matrix so that

$$\begin{aligned} h(x,\omega) = \text{Min} \quad & g_\omega^+ y^+ + g_\omega^- y^- \\ \text{s.t.} \quad & Iy^+ - Iy^- = r_\omega - T_\omega x \\ & y^+, y^- \geq 0. \end{aligned} \tag{5}$$

For any right-hand-side vector, $r_\omega - T_\omega x$, a feasible solution to (5) is easily determined by simply noting the sign of the right-hand side of each constraint and setting $y^+$ and $y^-$ accordingly. Furthermore, provided that the $i$th component of $g_\omega^+ - g_\omega^-$ satisfies $g_{\omega i}^+ - g_{\omega i}^- > 0$ for each $i$, this feasible solution is also optimal.

It is worth noting that when a problem has simple recourse, the subproblem (5) can be equivalently represented as

$$h(x,\omega) = \sum_i h_i(x,\omega)$$

where

$$\begin{aligned} h_i(x,\omega) = \text{Min} \quad & g^+_{\omega i} y^+_i + g^-_{\omega i} y^-_i \\ \text{s.t.} \quad & y^+_i - y^-_i = r_{\omega i} - \{T_\omega x\}_i \\ & y^+_i, y^-_i \geq 0. \end{aligned}$$

That is, the second-stage problem is separable by row. As a result, only the marginal distributions of the right-hand-side vector, $r_{\tilde{\omega}} - T_{\tilde{\omega}} x$, are necessary to calculate the expected value of the second-stage objective function values, which eases their calculation considerably.

Simple recourse problems arise in numerous situations. For example, when *target values* can be identified, and a primary concern involves minimizing deviations from these target values (although these might be weighted deviations), a simple recourse problem results.

### 3.4. Fixed Recourse

Another case that often arises is a property that is known as *fixed recourse.* A fixed recourse problem is one in which the constraint matrix in the recourse subproblem is not subject to uncertainty (i.e., it is fixed). In this case, the recourse subproblem is given by:

$$\begin{aligned} h(x,\omega) = \text{Min} \quad & g_\omega y \\ \text{s.t.} \quad & W y \geq r_\omega - T_\omega x \\ & y \geq 0. \end{aligned}$$

Note that the simple recourse problem has fixed recourse. This representation of $h(x,\omega)$ is apparently not much different from (4). However, when the second-stage objective coefficients are also fixed, the dual representation of the recourse subproblem is given by

$$\begin{aligned} h(x,\omega) = \text{Max} \quad & \pi^\top (r_\omega - T_\omega x) \\ \text{s.t.} \quad & \pi^\top W \leq g^\top \\ & \pi \geq 0. \end{aligned} \tag{6}$$

In this case, the set of dual feasible solutions is fixed (i.e., does not vary with $\omega$), a property that can be exploited computationally while designing a solution method.

### 3.5. Complete Recourse

So far, our focus has been on properties that arise from the recourse problem data. The reader will note that our presentation of the recourse problems suggests a decomposition of the problem into a first and second-stage problem. Indeed, many solution procedures exploit this opportunity for decomposition. In this setting, a question arises that involves feasibility of a particular first-stage vector $x$. That is, what assurances are there that the recourse function $h(x,\omega)$ is necessarily finite?

Note that $E[h(x,\tilde{\omega})] < \infty$ as long as the recourse subproblem (4) is feasible for all $x$. A problem for which $Y(\omega,\chi) = \{y \mid W_\omega y \geq \chi\}$ is nonempty for any value of $\chi$ is said to have complete recourse. If a problem has complete recourse, the recourse function is necessarily finite. A slightly less strenuous property, which leads to the same result, is known as "relatively complete recourse." Relatively complete recourse results if $Y(\omega,\chi)$ is nonempty for all $\chi \in \{r_\omega - T_\omega x \mid (\omega,x) \in \Omega \times X\}$. That is, relatively complete recourse merely restricts the statement of complete recourse to those values of the right-hand-side vector that can be encountered.

Complete recourse and relatively complete recourse may sound like extremely difficult properties to ensure, but in fact it is quite easy to guarantee their existence while a model is

formulated. For example, by penalizing deviations from feasibility in the recourse subproblem as follows:

$$\begin{aligned} h(x,\omega) = \text{Min} \quad & g_\omega y + Me^\top z \\ \text{s.t.} \quad & Wy + z \geq r_\omega - T_\omega x \\ & y, z \geq 0 \end{aligned}$$

(where $M$ is a large constant and $e$ is an appropriately dimensioned vector of ones), the problem has complete recourse. This type of modeling technique is commonly employed by stochastic programmers. Note that penalizing deviations from the original model in (4) tends to promote feasibility in the first-stage decision. Perhaps more importantly, a formulation such as this does not promote solutions that are overly influenced by rare events with extreme values.

### 3.6. Scenario Formulations

There are several ways to formulate an SLP. Thus far, our focus has been on formulations that explicitly represent the information process (as modeled by the scenario tree) within the sequence of decisions that are made. An alternate, but equally valid, representation of the problem is one in which a problem is formulated for each possible scenario and constraints are added to ensure the information structure associated with the decision process is honored. In this case, we begin by representing all decision variables as if they were permitted to depend on the specific scenario encountered, which leads to the scenario problems for each $\omega \in \Omega$:

$$\begin{aligned} \text{Min} \quad & cx_\omega + g_\omega y_\omega \\ \text{s.t.} \quad & T_\omega x_\omega + W_\omega y_\omega \geq r_\omega \qquad (7) \\ & x_\omega, y_\omega \geq 0. \end{aligned}$$

Without the introduction of additional constraints, we obtain a situation in which $\{(x_\omega, y_\omega)\}_{\omega\in\Omega}$ vary freely in response to each specific scenario. This runs contrary to the notion that some decisions can respond to the specific scenario, while others cannot. We can remedy this by including constraints that ensure that the decision sequence honors the information structure present in the problem as follows:

$$\text{Min} \quad \sum_{\omega\in\Omega} (cx_\omega + g_\omega y_\omega) p_\omega \qquad (8)$$

$$\begin{aligned} \text{s.t.} \quad & T_\omega x_\omega + W_\omega y_\omega \geq r_\omega \\ & x_\omega - x = 0 \quad \forall \omega \in \Omega \qquad (9) \\ & x_\omega, y_\omega \geq 0. \end{aligned}$$

Recall that for each $\omega \in \Omega$, $p_\omega = P\{\tilde{\omega} = \omega\}$ so that the objective in (8) represents the expected value as in (3). Constraints such as (9) are known as nonanticipativity constraints and ensure that decisions honor the information structure of the problem. Note that in (9) we have used a free variable, $x$, to constrain the scenario-dependent, first-stage variables $\{x_\omega\}_{\omega\in\Omega}$ to be equal. There are numerous ways in which these constraints might be represented. For example, in (9) we might replace $x$ with $E[x_{\tilde{\omega}}] = \sum_{\omega\in\Omega} p_\omega x_\omega$, as in Dempster [12] and in Rockafellar and Wets [40]. Alternatively, one might consider a more sophisticated representation that results in sparser constraints, such as one finds in Mulvey and Ruszczyński [32]. In general, the precise manner in which the nonanticipativity constraints are represented depends on the analysis and/or solution methodology to be undertaken. We note that when an SP is explicitly presented in its full form, as in (8), it is sometimes referred to as the deterministic equivalent problem (DEP). Properties and characteristics of the DEP are discussed in Wets [48].

### 3.7. Multistage Recourse Problems

Our focus thus far has been on two-stage problems with recourse, in which an initial decision is made while the specific scenario to be obtained is unknown, followed by another decision that is made after this information is available. It is not difficult to envision situations in which this "decide-observe-decide..." pattern is repeated several times. This leads to a multistage recourse problem. Formulating a multistage recourse problem can become a delicate operation due to the manner in which decisions and observations are interspersed. In this section, we will simply introduce a scenario formulation and indicate a method for identifying the nonanticipativity constraints.[3]

To begin, for each scenario $\omega \in \Omega$, let $c_\omega$ represent the objective function coefficients corresponding to the scenario and let $X(\omega)$ denote the set of solutions that are feasible for the scenario. That is, if there were exactly one data scenario to consider, the problem would be represented as:

$$\begin{aligned} &\text{Min} \quad c_\omega x_\omega \\ &\text{s.t.} \quad x_\omega \in X(\omega). \end{aligned} \tag{10}$$

In general, the scenario constraints (10) are represented as multistage constraints:

$$\sum_{j=1}^{t} A_{tj} x_j = b_t \quad t = 1, \ldots, T$$

so that the actions taken at stage $t$ are constrained by actions taken earlier in the process. If $\mathcal{N}$ denotes the set of nonanticipative solutions, then a multistage problem can be expressed as:

$$\text{Min} \quad \sum_{\omega} p_\omega c_\omega x_\omega \tag{11}$$

$$\text{s.t.} \quad x_\omega \in X(\omega) \quad \forall\ \omega \in \Omega$$

$$\{x_\omega\}_{\omega \in \Omega} \in \mathcal{N}. \tag{12}$$

As we have mentioned previously, the nature of the nonanticipativity constraints in (12) depends on the specific structure of the scenario tree. Suppose that we have a scenario tree as depicted in Figure 2. Note that in this case we have depicted a tree for a four-stage problem. In general, each node in the scenario tree corresponds to a collection of scenarios at a specific stage. Consider the node marked $n$ in the scenario tree, and note that it corresponds to a stage in the problem, $t(n)$.[4] Let the set of scenarios that pass through node $n$ be denoted as $\mathcal{B}(n)$, as depicted by the darkened scenarios in Figure 2. In the second stage, these scenarios cannot be distinguished from each other—while it is possible to recognize that the data indicates that it corresponds to node $n$, it is not possible to recognize which of the scenarios in $\mathcal{B}(n)$ will ultimately result. For solutions to the problem to be implementable (i.e., a.k.a. nonanticipative), we must ensure that decision variables that are associated with node $n$ produce identical values. One way to do this is to include constraints of the following form:

$$x_{t(n)\omega} - x_n = 0 \quad \forall\ \omega \in \mathcal{B}(n).$$

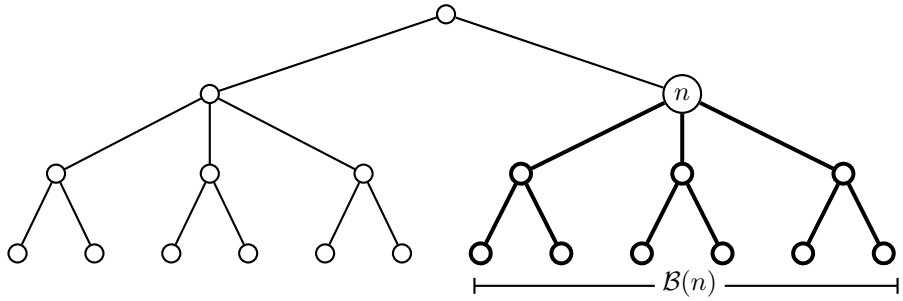
Note the similarity between this form of the constraint and (9). If we let $N$ denote the set of nonleaf nodes in the scenario tree, then we may represent the set of nonanticipative solutions as:

$$\mathcal{N} = \left\{ \{x_\omega\}_{\omega \in \Omega} \mid x_{t(n),\omega} - x_n = 0 \ \forall\ \omega \in \mathcal{B}(n),\ \forall\ n \in N \right\}.$$

[3] If we adopt a decision-stage formulation similar to (3), then $h(x, \omega)$ includes the expected "cost-to-go" function associated with later decision stages.

[4] In this case, $t(n) = 2$.

FIGURE 2. Bundles within a scenario tree.




Finally, as previously noted, the representation of the nonanticipativity constraints is not unique—there are any number of choices available. The specific choice selected is typically guided by the solution method to be used.

### 3.8. Solutions to Recourse Problems

Finally, it is necessary to comment on the nature of a solution to these problems, which involve multiple (i.e., two or more) decision stages. In deterministic linear programming, we are accustomed to specifying the entire solution vector—indicating a value (zero or otherwise) for each individual variable. If we consider this within the context of a two-stage problem, that would require reporting values for $x$ as well as for $\{y_\omega\}_{\omega\in\Omega}$—a task that can quickly become daunting. Note that if there are only 10 random variables within the data elements, and these are modeled as independent random variables with only three possible outcomes each (corresponding to high, medium, and low values), then $\Omega$ contains $3^{10} = 59,049$ separate data scenarios. For this reason, the reporting of stochastic programming solutions is typically restricted to the first-stage variables. Note that this is especially appropriate when considering the fact that this is the action that requires immediate commitment—all other decisions can be delayed until further information is postponed.

## 4. Does Uncertainty Matter?—A Quick Check

The Dakota example in §2.1 illustrates some of the ways in which deterministic models combined with investigations of solution sensitivity do not adequately represent opportunities to adapt to information obtained at intermediate stages of the decision sequence. The example also illustrates the manner in which a stochastic linear programming formulation might differ from a related deterministic linear programming formulation. For one thing, we see that the size of the problem increases, and we can easily imagine that solution difficulties increase as well. In fact, as the number of scenarios that must be considered increases, hopes of solving the resulting problem using general purpose, off-the-shelf LP solvers are quickly abandoned in favor of specialized solution methods. Prior to solving an SLP, it is useful to investigate the quality of the solution that can be obtained via the more easily solved deterministic LP.

We return to the general structure of the recourse problem (3)–(4),

$$\begin{aligned} &\text{Min} && cx + E[h(x,\tilde{\omega})] \\ &\text{s.t.} && Ax \geq b \\ & && x \geq 0 \end{aligned}$$

$$\text{where} \quad h(x,\omega) = \begin{aligned} &\text{Min} && g_\omega y \\ &\text{s.t.} && W_\omega y \geq r_\omega - T_\omega x \\ & && y \geq 0. \end{aligned}$$

Note that the function, $h(x,\omega)$, is defined as the value function of the second-stage linear program that appears in (4), and that the vector $x$ appears on the right-hand side of this minimization problem. The dual to (4) is given by

$$\begin{aligned} h(x,\omega) = \text{Max} \quad & \pi^\top (r_\omega - T_\omega x) \\ \text{s.t.} \quad & \pi^\top W_\omega \le g_\omega^\top \\ & \pi \ge 0. \end{aligned}$$

Using this dual representation of $h(x,\omega)$, it is a relatively simple exercise to verify that it is a piecewise linear convex function of the variable $x$. If the sample space of $\tilde{\omega}$ is countable, the expected value of this function, which appears in the objective of (3), is simply

$$E[h(x,\tilde{\omega})] = \sum_{\omega\in\Omega} h(x,\omega)p_\omega. \tag{13}$$

Convexity is preserved through this operation. In general, piecewise linearity is preserved as well.[5] When the problem has fixed recourse, so that uncertainty is not present in the second-stage constraint matrix, $W$, and the second-stage objective coefficients, $g$, are fixed as well, similar arguments will ensure that $h(x,\omega)$ is also a convex function of $\omega$.

Jensen's inequality, which involves convex functions of random variables, applies in this case and offers a simple method for bounding the objective value improvement that might be obtained via solution as an SLP. Jensen's inequality ensures that when $h(x,\omega)$ is convex in $\omega$ and $\tilde{\omega}$ is a random variable, then

$$h(x, E[\tilde{\omega}]) \le E[h(x,\tilde{\omega})].$$

Note that if $X = \{x : Ax \ge b,\ x \ge 0\}$ (i.e., all $x$ that satisfy the first-stage constraints in (3)), then

$$\begin{aligned} & cx + h(x, E[\tilde{\omega}]) \le cx + E[h(x,\tilde{\omega})] \quad \forall x \in X \\ \Rightarrow \quad & \operatorname*{Min}_{x\in X}\{cx + h(x, E[\tilde{\omega}])\} \le \operatorname*{Min}_{x\in X}\{cx + E[h(x,\tilde{\omega})]\}. \end{aligned} \tag{14}$$

Equation (14) indicates an ordering in the optimal objective function values for two distinct, yet related, problems. On the left-hand side, we have the case in which all random elements are replaced by their expected values—the so-called *mean value problem.* On the right-hand side, we have the SLP. Note that (14) indicates that the optimal objective value associated with the SLP is bounded by the optimal value of the mean value problem. Let

$$\begin{aligned} \bar{x} &\in \arg\min\{cx + h(x, E[\tilde{\omega}]) \mid x \in X\} \\ x^* &\in \arg\min\{cx + E[h(x,\tilde{\omega})] \mid x \in X\} \end{aligned}$$

and note that $c\bar{x} + h(\bar{x}, E[\tilde{\omega}]) \le cx^* + E[h(x^*,\tilde{\omega})]$. Note also that because $\bar{x} \in X$, we have that $cx^* + E[h(x^*,\tilde{\omega})] \le c\bar{x} + E[h(\bar{x},\tilde{\omega})]$. In combination, this yields

$$c\bar{x} + h(\bar{x}, E[\tilde{\omega}]) \le cx^* + E[h(x^*,\tilde{\omega})] \le c\bar{x} + E[h(\bar{x},\tilde{\omega})]. \tag{15}$$

The inequalities in (15) suggest a fairly straightforward method for quickly determining whether or not solving the problem as an SLP is worth the effort required:

*Step* 1. Solve $\text{Min}_{x\in X}\{cx + h(x, E[\tilde{\omega}])\}$ to obtain $\bar{x}$.

*Step* 2. Evaluate $E[h(\bar{x},\tilde{\omega})]$.

*Step* 3. If $E[h(\bar{x},\tilde{\omega})] - h(\bar{x}, E[\tilde{\omega}])$ is sufficiently small, accept $\bar{x}$ as an acceptable solution.

[5] In general, the expectation is calculated via integration. In some special cases when the random variables are absolutely continuous, the function is smooth rather than piecewise linear. However, convexity in $x$ is preserved nonetheless.

The gap identified in Step 3 is an upper bound on the loss of optimality associated with using $\bar{x}$ in lieu of identifying $x^*$. When this gap is sufficiently small, there is no need to invest further effort in pursuit of an optimal solution. Note that the precise evaluation of the expected value indicated in Step 2 may be difficult to undertake. In this case, statistical estimation techniques are suggested. For example, suppose that $\{\omega^t\}_{t=1}^N$ is a large number of randomly generated observations of $\tilde{\omega}$. Then $E[h(\bar{x},\tilde{\omega})]$ can be approximated using the sample mean, $(1/N)\sum_{t=1}^N h(\bar{x},\omega^t)$, and confidence statements regarding the accuracy of the estimated value are readily obtained.

For additional methods that can be used to estimate the potential value associated with solving the stochastic program (i.e., as compared to simply using the solution to the mean value problem), the reader is referred to Birge [4]. Note that (15) makes use of upper and lower bounds on the optimal SLP objective function value. The topic of upper and lower bounds in SLP has been extensively studied. The reader is referred to Birge and Wets [6], Edirisinghe and Ziemba [14], and Frauendorfer [15] for further comments on more involved bounding techniques.

## 5. Solution Approaches

By now, it is probably clear that when $\tilde{\omega}$ involves discrete random variables, a stochastic linear program is really a specially structured linear program. If the number of scenarios is small enough, the SLP can be solved using an off-the-shelf linear programming solver. In general, however, the number of scenarios can become explosively large. For example, when the random variables are independent, the number of scenarios is the product of the number of possible outcomes for each marginal random variable, which can lead to an explosive number of possible outcomes. When this occurs, it is necessary to use solution methods that are specifically designed to exploit the structural properties of the stochastic program. These methods typically involve a decomposition of the problem, and increasingly often use statistical estimation methods as well.

### 5.1. Decomposition

The first solution procedure proposed for two-stage stochastic linear programs with recourse is the L-shaped method (van Slyke and Wets [47]). The L-shaped method decomposes the problem by stage—the first-stage problem leads to a master problem and the second-stage problem leads to a subproblem. In reality, the method is simply an adaptation of Benders' decomposition [2] to the structure of the second-stage problem. Beginning with the problem statement as in (3)–(4), the second-stage objective function $E[h(x,\tilde{\omega})]$ is approximated using a piecewise linear convex function $\nu(x)$, where

$$\nu(x) = \text{Max}\{\alpha_t + \beta_t x \mid t = 1, ..., k\}.$$

The approximation is developed iteratively, and $\nu(x)$ is typically represented in a master program using a cutting-plane approximation:

$$\begin{aligned} \text{Min} \quad & cx + \nu \\ \text{s.t.} \quad & Ax \geq b \\ & \nu \geq \alpha_t + \beta_t x \quad t = 1, ..., k \\ & x \geq 0. \end{aligned} \tag{16}$$

The coefficients on these cutting planes are obtained from dual solutions to (4). That is,

$$\begin{aligned} h(x,\omega) &= \text{Min}\{g_\omega y \mid W_\omega y \geq r_\omega - T_\omega x,\ y \geq 0\} \\ &= \text{Max}\{\pi^\top (r_\omega - T_\omega x) \mid \pi^\top W \leq g^\top,\ \pi \geq 0\}. \end{aligned}$$

Let $\Pi = \{\pi \mid \pi^\top W \le g^\top, \pi \ge 0\}$, and note that for each $\pi \in \Pi$,

$$h(x,\omega) \ge \pi^\top (r_\omega - T_\omega x)$$

with equality holding when $\pi \in \arg\max\{\pi^\top (r_\omega - T_\omega x) \mid \pi \in \Pi\}$. Consequently, if $x^k$ is a solution to (16) obtained in the $k$th iteration, and $\pi(x^k,\omega) \in \arg\max\{\pi^\top (r_\omega - T_\omega x^k) \mid \pi \in \Pi\}$, then the next cut to be added to the piecewise linear convex approximation of $E[h(x,\tilde{\omega})]$ is given by:

$$\alpha_{k+1} + \beta_{k+1} x = \sum_{\omega \in \Omega} \pi^\top (x^k,\omega)^\top (r_\omega - T_\omega x) p_\omega.$$

In representing the cuts in this manner, a property of separability in the subproblem has been exploited. Formally, Benders' decomposition would define the subproblem as $E[h(x,\tilde{\omega})]$, a single problem involving all possible scenarios. However, because

$$\begin{aligned} E[h(x,\tilde{\omega})] &= \text{Min}\left\{ \sum_{\omega \in \Omega} p_\omega g_\omega y_\omega \;\middle|\; W_\omega y \ge r_\omega - T_\omega x,\ y_\omega \ge 0 \ \forall\ \omega \in \Omega \right\} \\ &= \sum_{\omega \in \Omega} p_\omega \text{Min}\left\{ g_\omega y_\omega \;\middle|\; W_\omega y \ge r_\omega - T_\omega x y_\omega \ge 0 \right\}, \end{aligned}$$

the L-shaped method is able to define cutting-plane coefficients from the individual (scenario) subproblems. This results in a substantial reduction in the size of the subproblems that are solved (although it increases the number of such problems that are solved in any iteration).

If the problem has either complete or relatively complete recourse, this presentation of the L-shaped method is sufficient to ensure that an optimal solution to the SLP can be identified. If the problem is not "blessed" with one of these useful properties, it is possible that for some values of $x^t$ one or more of the scenario subproblems are infeasible. This means that $x^t$ is not a feasible solution in the first place, and constraints must be added to the master program to eliminate it from further consideration. This is accomplished using extreme rays from the subproblem dual as in Benders' decomposition.

In many ways, the L-Shaped method is by now a classic technique for solving two-stage stochastic programs. Although it is well suited for an introduction to SLP solution methodology, it is no longer computationally effective for large-scale problems.[6] One of the first major improvements to this basic methodology involved the introduction of a regularizing term (a quadratic proximal term added to the objective, $\|x - x^k\|^2$, for example) in Ruszczyński [42]. The inclusion of this term imparts mathematical properties on the master problem that can be exploited computationally. Simultaneously, Ruszczyński [42] suggests a multicut representation of the second-stage objective approximation (i.e., one cut per scenario, rather than a single cut on the expected objective function), an adaptation of the basic cutting-plane methodology later investigated in Birge and Louveaux [5].

## 5.2. Statistically Based Methods

One of the major handicaps of the L-shaped method is the need to solve a subproblem for each scenario (i.e., for all $\omega \in \Omega$) for each cutting plane. In large-scale problems, the number of scenarios is much too high for this to be a reasonable approach, and the possibility of using statistical estimations of the recourse function becomes computationally attractive.

Conceptually, the simplest method for incorporating statistical approximations in the solution procedure is to replace the recourse function, $[h(x,\tilde{\omega})]$, by a sample mean approximation.

[6] In this sense, "large scale" typically refers to the number of scenarios in $\Omega$.

That is, if $\{\omega^t\}_{t=1}^n$ is a collection of independent and identically distributed observations of $\tilde{\omega}$, then one might consider undertaking the solution of *the sample mean problem*:

$$\begin{aligned} \text{Min} \quad & cx + \frac{1}{n}\sum_{t=1}^{n} h(x, \omega^t) \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0. \end{aligned} \tag{17}$$

Notice that (17) merely replaces the original distribution of $\tilde{\omega}$ with the empirical distribution associated with the sample. In other words, the sample mean problem is a stochastic program with an alternate distribution! When the sample size is significantly smaller than the number of scenarios in the original distribution, (17) will be much easier to solve than (3).

On the surface, this approach is quite appealing. However, we note that the solution obtained is dependent on the specific sample that was drawn. Consequently, it is subject to error in the same manner that the mean value solution is subject to error. In the absence of relatively complete recourse, it is possible that the solution obtained is actually infeasible! That is, there may be some scenarios that are not represented in the sample for which the solution to (17) is not feasible. For this reason, relatively complete recourse is a critical property for problems that are to be solved using statistical approximation schemes. A more subtle difficulty with this approach resides in the objective function values obtained from (17). In §4 we use Jensen's inequality to establish a lower bound on the optimal objective value. In [30], Mak et al. use similar logic to establish that the optimal value for (17) provides a biased estimator for the optimal value for (3). That is, the optimal value for (17) is *expected* to be lower than the optimal solution to (3). The reason for this may be fairly clear—solution of the mean value problem permits an optimal adjustment to a small subset of the scenarios in $\Omega$—while solution of the stochastic program requires that the entire set of scenarios be considered.

Based on results in Mak et al. [30], efforts have been made to eliminate this bias and potential error in the solution of the mean value problem. In essence, the suggestion is to undertake the solution of the mean value problem a number of times using a series of independent samples. That is, if $M$ separate samples are obtained (each with $n$ observations), then $M$ separate solutions are obtained, denoted as $\{x_{mv}^m\}_{m=1}^M$. These $M$ solutions are candidate solutions, from which the "apparently best" candidate is selected. To compare solutions, $cx_{mv}^m + E[h(x_{mv}^m, \tilde{\omega})]$ is estimated for $m = 1, 2, ..., M$. Of course, given that $\Omega$ involves a large collection of scenarios, these objective function values are estimated statistically based on a random sample that is independent from the samples used to obtain the candidate solutions. This process is referred to as the *sample average approximation method* (Shapiro [44] and Kleywegt et al. [28]), although it is related to *sample path optimization* (Robinson [37] and Fu et al. [35]). There is some evidence that when the objective function values are easy to estimate with only a small number of observations (i.e., when the variability in the objective function values is rather low), this replication can be undertaken without excessive computational burden—otherwise, the replication can become somewhat cumbersome (see, e.g., Higle and Zhao [25]).

### 5.3. Stochastic Decomposition

One method that specifically integrates elements of decomposition techniques and statistical approximation techniques is stochastic decomposition (SD) (Higle and Sen [21, 22]). Unlike the sample mean optimization in (17), which operates with a fixed sample size, SD operates with an adaptive sample size—increasing the sample size as iterations progress. Unlike the L-shaped method, which solves a subproblem for each scenario for each cutting plane constructed, SD uses recursive approximation methods based on previously solved problems

to bypass the solution of the vast majority of the subproblems that would otherwise be solved. The combination of adaptive sampling and subproblem approximations has proven to be quite powerful, especially when a regularized master program (as described in Higle and Sen [20]) is used (see, e.g., Higle and Zhao [25]).

### 5.4. Methods for Multistage Problems

It is probably obvious in §3.7 that solving multistage problems can be substantially more complex than solving two-stage problems. With multistage problems, it is especially important to exploit the opportunity to decompose the problem.

Early methods for solving multistage problems decomposed the problem by decision stage, bearing a strong similarity to the computation of the cost-to-go-function commonly employed in dynamic programming methods. Among these are nested decomposition (Birge [3]) and MSLiP (Gassmann [17]). Recall that the scenario tree is a critical construct in a stochastic programming model—and that each node in the scenario tree can be viewed as being associated with a linear program. In a stage decomposition of the problem, an approximation of the value function associated with the linear program is developed for each node in the scenario tree. For this reason, careful attention is paid to the manner in which nodes in the scenario tree are selected for improvement of the local approximation.

In many cases, it may be more advantageous to consider a scenario decomposition of the problem. Note that in (11)–(12), the nonanticipativity constraints are the only constraints that link scenarios together within the formulation. That is, in the absence of these nonanticipativity constraints, the multistage formulation naturally decomposes by scenario. As a result, a scenario decomposition can be undertaken via a Lagrangian relaxation of the nonanticipativity constraints as in the progressive hedging algorithm of Rockafellar and Wets [40] or the augmented Lagrangian decomposition method in Rosa and Ruszczyński [41], as well as the closely related diagonal quadratic approximation method in Mulvey and Ruszczyński [31, 32]. Alternatively, a scenario decomposition can be obtained via a dual representation of the multistage SLP, as discussed in Higle et al. [19]. It is interesting to note that Higle et al. [19] combine a scenario decomposition of the problem with statistical approximation methods.

## 6. Computational Illustration

To appreciate the scale of stochastic programming problems and the operating characteristics of some of the solution methodologies, a brief computational study is in order. A small collection of problem instances, which are well referenced in the stochastic programming literature and publicly available, are used. These problems are characterized as two-stage stochastic linear programs with general and complete recourse. The specific problems that we considered are:

*PGP2* is a small power generation planning problem described in Higle and Sen [20]. The first stage involves decisions regarding generator capacities, while the second-stage models generator operations to meet demand for power, which is modeled with random variables.

*20Term* is a motor freight scheduling problem described in Mak et al. [30]. The first-stage models the position of fleet vehicles at the start of the day, while the second-stage models decisions regarding shipments of freight by these vehicles. The point-to-point demand for goods is modeled using random variables.

*STORM* is a large-scale air freight scheduling model arising in military logistics described in Mulvey and Ruszczyński [32]. Demand for freight movement is modeled using random variables.

*SSN* is a large-scale telecommunication network planning problem described in Sen et al. [43]. The first stage involves decisions regarding network link capacities, while the

TABLE 7. Test problems.

| Problem | First stage Rows ($m_1$) | First stage Columns ($n_1$) | Second stage Rows ($m_2$) | Second stage Columns ($n_2$) | No. of RVs | No. of scenarios ($N_\Omega$) |
|---|---|---|---|---|---|---|
| PGP2 | 2 | 4 | 7 | 12 | 3 | 576 |
| SSN | 1 | 89 | 175 | 706 | 86 | $> 5^{86}$ |
| 20Term | 3 | 63 | 124 | 764 | 40 | $2^{40}$ |
| STORM | 59 | 128 | 526 | 1259 | 118 | $5^{118}$ |

second-stage models network operations (call routing) to meet demand, which is modeled using random variables.

To appreciate the size of the full representation of the SLP as in (3), the number of rows is $m_1 + N_\Omega m_2$ and the number of columns is $n_1 + N_\Omega n_2$. In this sense, it is clear that while PGP2 is a small problem, the remaining problems are quite large.

Prior to solving these problems, we begin with a quick check to determine whether or not it appears necessary to consider the stochastic elements of the problem. The results of these calculations appear in Table 8. The upper and lower bounds are calculated as described in (15). The relative error is calculated as the difference between the upper and lower bounds, relative to the lower bound. PGP2 involves a relatively small sample space, with $N_\Omega = 576$. Consequently, $E[h(\bar{x}, \tilde{\omega})]$ can be calculated precisely. Because the remaining problems involve sample spaces that are much too large to work with directly, the upper bounds are estimated statistically and are accurate to within 1% (with 95% confidence). Note that despite being the largest problem with regard to row, column, and scenario dimensions in Table 7, STORM should be considered to be the easiest problem among the group! The error associated with the mean value solution is estimated to be 0.2%. Given that the error in the upper bound estimate is 1%, this means that the estimated error is not significantly different from zero! It is clear that solving this particular problem as a stochastic program offers negligible value beyond the solution of the mean value problem. For this reason, we will omit it from further consideration and focus our attention on the remaining problems, for which uncertainty plays a more significant role.

In Table 9, we compare solutions obtained via the L-shaped method and those obtained via the regularized SD (see van Slyke and Wets [47], and Higle and Sen [22], respectively). A few comments about the manner in which these computations were performed are in order. SD is a statistically based decomposition method, and uses randomly generated observations of $\tilde{\omega}$. For this reason, the values reported are the averages over 10 independent replications, with standard deviations reported parenthetically. Within SD, the sample is increased in each iteration, so that the sample size depends on the number of iterations executed prior to termination of the method. For these computations, SD was terminated based on indications that the solution's objective value was nearly optimal, using the specific rules discussed in Higle and Zhao [25]. As noted, PGP2 is a small problem, and complete enumeration of the set of scenarios does not impose a significant computational burden. For this reason, PGP2 was solved exactly using the L-shaped method. For the remaining problems, precise solution

TABLE 8. Bounds based on mean value solutions, $\bar{x}$.

| Problem | Lower bound | Upper bound | Relative error |
|---|---|---|---|
| PGP2 | 428.5 | 505.3 | 0.179 |
| SSN | 0 | 73.4 | (zero-divide) |
| 20Term | 239273 | 279334 | 0.167 |
| STORM | 15,524,803 | 15,560,749 | 0.002 |

Table 9. Illustrative computational results.

| | L-shaped | | Stochastic decomposition | |
|---|---|---|---|---|
| Problem | Time | Objective value | Time | Objective value |
| PGP2 | 2.4 | 447.32 | 4.9 (4.3) | 448.7 (1.31) |
| SSN | 36,893 (5,962) | 10.02 (0.06) | 2,913 (1,391) | 10.26 (0.14) |
| 20Term | 11,757 (816.5) | 254,744 (57) | 250 (32) | 254,581 (79) |

using the L-shaped method is not possible, so a random sample was used to create a sample average approximation of the problem, as in (17). In these cases, the sample used was the same sample used by SD. In this fashion, it is possible to note the relative effort required by SD and L-shaped methods.

In Table 9, time is measured in seconds. Note that for the smaller problem, PGP2, the L-Shaped method performs quite well—there is no apparent need for a more sophisticated methodology for problems of this size. However, the time required to solve these problems using the L-shaped method on the sample average approximation of the problem is substantially larger than for the regularized SD, even though both methods use an identical set of observations! For SSN, the solution times for SD are less than 10% of the solution times for the L-shaped method, while for 20Term, the SD solution times are less than 2.5% of the solution times for the L-shaped method. Note also that the differences in the quality of the solutions produced are minimal. In short—methods designed specifically to exploit the structure of the problems can yield considerable computational advantage.

## 7. A Few Things That Were Omitted

In a tutorial chapter of this size, covering a field as dynamic as stochastic programming necessarily results in a variety of topics that must be omitted. For example, we have omitted discussion of any specific problem applications. For a variety of stochastic programming applications, the reader is referred to the book by Ziemba and Wallace [50]. In the remainder of this section, we include a few references to some of the topics that were omitted.

The recourse models that we focus on throughout this paper involve those for which optimality and feasibility are defined through expected values of relevant functions. An alternate set of models results when constraints must be satisfied with a given probability. These are referred to as models with probabilistic constraints. An excellent resource for the study of such models is the text by Prékopa [36].

Expected values are risk-neutral models, and do not always provide a satisfactory model for decisions under risk. Recent investigations of risk models that are amenable to large-scale computational efforts have lead to the development of *conditional value at risk*, a topic introduced by Rockafellar and Uryasev in [38] and [39]. An alternate measure of risk, which involves semideviations, was introduced in Ogryczak and Ruszczyński [33], and explored further in Ogryczak and Ruszczyński [34]. Modeling risk through dominance constraints is discussed in Dentcheva and Ruszczyński [13].

With regard to model development, we have focused on models involving discrete random variables. As a general rule, solution methods based on statistical approximations are applicable for both continuous and discrete distributions. The scenario tree is a critical element of the multistage model, and some level of care should be exercised when generating it. For an initial discussion of techniques that may be helpful in creating the scenario tree, see Hoyland and Wallace [26]. In cases where the scenario tree is extremely large, methods for aggregation and disaggregation of the tree can be extremely helpful, as discussed in Casey and Sen [8]. Finally, the models presented in this paper have been focused on continuous variables. Models and methods for stochastic integer programs form a very hot area of research. A stochastic integer programming bibliography can be found in van der Vlerk [46].

When problems are solved using partial information, as is the case when statistical approximations are used, questions arise regarding the validity of the solutions obtained. In some cases, these questions can be addressed directly via optimality tests incorporated within an algorithmic technique (as is the case with SD; Higle and Sen [22, 23]). In other cases, there is a need to investigate solution quality more directly. Methods designed to address such questions appear in Mak et al. [30] and Bayraksan and Morton [1]. Additionally, investigations of possible uses of variance reduction techniques with statistically motivated solutions methods are discussed in Infanger [27], Koivu [29], and Higle [18].

Some problem instances suitable for testing algorithms have been developed, but the choices available are very limited, especially within the realm of large-scale instances. To facilitate the sharing of problem data, the SMPS format has been created (Birge et al. [7]). For an excellent discussion of the format, which is related to the MPS format for mathematical programming problems, the reader is referred to Gus Gassmann's helpful website [16].

## 8. Resources on the Web

The stochastic programming community maintains a website with general information on the discipline [10]. A vast bibliography of stochastic programming references is maintained by Maarten van der Vlerk [46] (University of Groningen, The Netherlands). Maarten is always happy to accept information regarding updates to this website. A separate website is maintained by Werner Römisch (Humboldt University, Germany) for dissemination of recent (i.e., prepublication) results in stochastic programming [45].

## Acknowledgments

The author acknowledges the contributions of Stein Wallace for discussions leading to the example in §2.1, and Lei Zhao for his assistance with the computational work reported in this paper. This work was funded by Grant DMS-0400085 from the National Science Foundation.

## References

[1] G. Bayraksan and D. P. Morton. Assessing solution quality stochastic programs. `http://www.speps.info`, 2005.

[2] J. F. Benders. Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik* 4:238–252, 1961.

[3] J. R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research* 33(5):989–1007, 1985.

[4] J. R. Birge. The value of the stochastic solution in stochastic linear programs with fixed recourse. *Mathematical Programming* 31(1):25–41, 1982.

[5] J. R. Birge and F. V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research* 34:384–392, 1988.

[6] J. R. Birge and R. J.-B. Wets. Designing approximations schemes for stochastic optimization problems, in particular, for stochastic programs with recourse. *Mathematical Programming Study* 27:54–102, 1986.

[7] J. R. Birge, M. A. H. Dempster, H. I. Gassmann, E. A. Gunn, A. J. King, and S. W. Wallace. A standard input format for multiperiod stochastic linear programs. *COAL Newsletter* 17:1–19, 1987.

[8] M. Casey and S. Sen. The scenario generation algorithm for multi-stage stochastic linear programming. *Mathematics of Operations Research*, 2005. Forthcoming.

[9] A. Charnes and W. W. Cooper. Chance-constrained programming. *Management Science*, 6(1):73–79, 1959.

[10] Stochastic programming community website. `http://www.stoprog.org`.

[11] G. B. Dantzig. Linear programming under uncertainty. *Management Science* 1(3–4):197–206, 1955. Republished in the 50th anniversary issue of *Management Science* 50(12):1764–1769, 2004.

[12] M. A. H. Dempster. On stochastic programming. II. dynamic problems under risk. *Stochastics* 25(1):15–42, 1988.

[13] D. Dentcheva and A. Ruszczyński. Optimization with stochastic dominance constraints. *SIAM Journal on Optimization* 14(2):548–566, 2003.

[14] C. Edirisinghe and W. T. Ziemba. Bounds for two-stage stochastic programs with fixed recourse. *Mathematics of Operations Research* 19:292–313, 1994.

[15] K. Frauendorfer. Solving SLP recourse problems with aribtrary multivariate distributions. *Mathematics of Operations Research* 13:377–394, 1988.

[16] H. I. Gassmann. The SMPS format for stochastic linear programs. `http://www.mgmt.dal.ca/sba/profs/hgassmann/SMPS2.htm#RefBirge_etal` (last updated Feb. 2005).

[17] H. I. Gassmann. MSLiP: A computer code for the multistage stochastic linear programming problem. *Mathematical Programming* 47:407–423, 1990.

[18] J. L. Higle. Variance reduction and objective function evaluation in stochastic linear programs. *INFORMS Journal on Computing* 10(2):236–247, 1998.

[19] J. L. Higle, B. Rayco, and S. Sen. Stochastic scenario decomposition for multi-stage stochastic programs. `http://www.sie.arizona.edu/faculty/higle`, 2004.

[20] J. L. Higle and S. Sen. Finite master programs in regularized stochastic decomposition. *Mathematical Programming* 67:143–168, 1994.

[21] J. L. Higle and S. Sen. Stochastic decomposition: An algorithm for two stage linear programs with recourse. *Mathematics of Operations Research* 16(3):650–669, 1991.

[22] J. L. Higle and S. Sen. *Stochastic Decomposition: A Statistical Method for Large Scale Stochastic Linear Programming*. Kluwer Academic Publishers, Boston, MA, 1996.

[23] J. L. Higle and S. Sen. Statistical approximations for stochastic linear programs. *Annals of Operations Research* 85:173–192, 1999.

[24] J. L. Higle and S. W. Wallace. Sensitivity analysis and uncertainty in linear programming. *Interfaces* 33(4):53–60, 2003.

[25] J. L. Higle and L. Zhao. Adaptive and nonadaptive samples in solving stochastic linear programs: A computational approach. 2004. `http://www.sie.arizona.edu/faculty/higle.`

[26] K. Hoyland and S. W. Wallace. Generating scenario trees for multistage decision problems. *Management Science* 47(2):295–307, 2001.

[27] G. Infanger. Monte Carlo (importance) sampling within a benders decomposition algorithm for stochastic linear programs. *Annals of Operations Research* 39:69–95, 1992.

[28] A. J. Kleywegt, A. Shapiro, and T. Homem-de-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization* 12(2):479–502, 2001.

[29] M. Koivu. Variance reduction in sample approximation of stochastic programs. *Mathematical Programming* 103(3):463–485, 2005.

[30] W. K. Mak, D. P. Morton, and R. K. Wood. Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters* 24:47–56, 1999.

[31] J. M. Mulvey and A. Ruszczyński. A diagonal quadratic approximation method for large-scale linear programs. *Operations Research Letters* 12:205–215.

[32] J. M. Mulvey and A. Ruszczyński. A new scenario decomposition method for large-scale stochastic optimization. *Operations Research* 43(3):477–490, 1995.

[33] W. Ogryczak and A. Ruszczyński. From stochastic dominance to mean-risk models: Semideviations as risk measures. *European Journal of Operational Research* 116, 1999.

[34] W. Ogryczak and A. Ruszczyński. On consistency of stochastic dominance and mean-semideviation models. *Mathematical Programming* 89, 2001.

[35] B.-R. Fu, S. M. Robinson, E. L. Plambeck, and R. Suri. Sample-path optimization of convex stochastic performance functions. *Math. Programming B* 75:137–176, 1996.

[36] A. Prékopa. *Stochastic Programming*. Kluwer Academic Publishers Group, Boston, MA, 1995.

[37] S. M. Robinson. Analysis of sample-path optimization. *Mathematics of Operations Research* 21:513–528, 1996.

[38] R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk* 2(21), 2000.

[39] R. T. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, 26(7):1443–1471, 2002.

[40] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research* 16(1):119–147, 1991.

[41] C. H. Rosa and A. Ruszczyński. On augmented Lagrangian decomposition methods for multistage stochastic programs. *Annals of Operations Research* 64:289–309.

[42] A. Ruszczyński. A regularized decomposition method for minimizing a sum of polyhedral functions. *Mathematical Programming* 35:309–333, 1986.

[43] S. Sen, R. D. Doverspike, and S. Cosares. Network planning with random demand. *Telecommunications Systems* 3:11–30, 1994.

[44] A. Shapiro. Stochastic programming by Monte Carlo methods. `http://www.isye.gatech.edu/ashapiro/publications.html`, 2000.

[45] Stochastic programming e-print series. `http://www.speps.info`.

[46] M. van der Vlerk. Stochastic programming bibliography. `http://mally.eco.rug.nl/index.html?spbib.html`.

[47] R. van Slyke and R. J.-B. Wets. L-shaped programs with applications to control and stochastic programming. *SIAM J. on Applied Mathematics* 17:638–663, 1969.

[48] R. J.-B. Wets. Stochastic programs with fixed recourse: the equivalent deterministic program. *SIAM Review* 16:309–339, 1974.

[49] W. L. Winston and M. Venkataramanan. *Introduction to Mathematical Programming: Applications and Algorithms*. Duxbury Press, Belmont, CA, 2002.

[50] W. T. Ziemba and S. W. Wallace, eds. *Application of Stochastic Programming*. SIAM Series on Optimization. SIAM, Philadelphia, PA, 2005.

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS | ISBN 1-877640-21-2
DOI 10.1287/educ.1053.0013

# Network Models in Railroad Planning and Scheduling

***Ravindra K. Ahuja***
Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida 32611, ahuja@ufl.edu

***Claudio B. Cunha***
Department of Transportation Engineering, Escola Politécnica, University of São Paulo, São Paulo, SP, Brazil, cbcunha@usp.br

***Güvenç Şahin***
Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida 32611, guvencs@ufl.edu

**Abstract** The past few decades have witnessed numerous applications of operations research in logistics, and these applications have resulted in substantial cost savings. However, the U.S. railroad industry has not benefited from the advances, and most of the planning and scheduling processes do not use modeling and optimization. Indeed, most of the planning and scheduling problems arising in railroads, which involve billions of dollars of resources annually, are currently being solved manually. The main reason for not using OR models and methodologies is the mathematical difficulty of these problems, which prevented the development of decision tools that railroads can use to obtain implementable solutions. However, now this situation is gradually changing. We are developing cutting-edge operations research algorithms, by using state-of-the-art ideas from linear and integer programming, network flows, discrete optimization, heuristics, and very large-scale neighborhood (VLSN) search, that railroads have already started using and from which they have started deriving immense benefits. This chapter gives an overview of the railroad planning and scheduling problems, including the railroad blocking problem, train scheduling problem, yard location problem, train dispatching problem, locomotive scheduling problem, and crew scheduling problem. Some of these problems are very large-scale integer programming problems containing billions or even trillions of integer variables. We will describe algorithms that can solve these problems to near-optimality within one to two hours of computational time. We present computational results of these algorithms on the data provided by several U.S. railroads, demonstrating potential benefits from tens to hundreds of millions annually.

**Keywords** railroad scheduling; transportation; integer programming; networks; combinatorial optimization; heuristics; modeling

## 1. Introduction

Transportation is one of the most vital services in the modern economy. In this context, railroads play a significant role by providing efficient and cost-effective freight services for the transportation of products and goods. They also pose some of the most exciting and challenging opportunities for the application of operations research (OR) tools and methodologies.

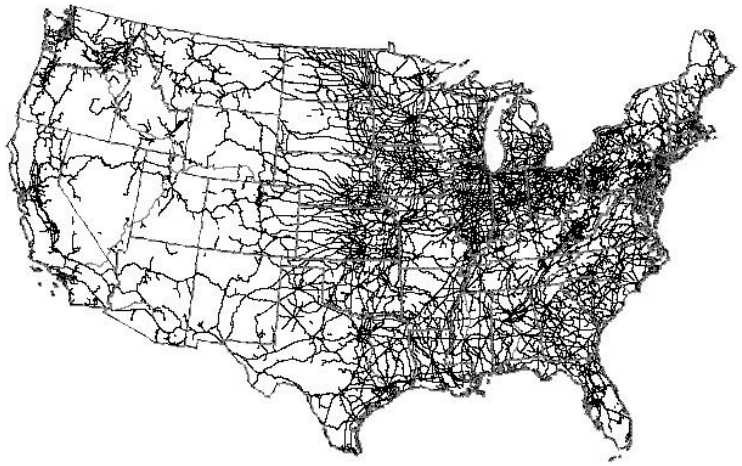
From the supply side, which involves all actions necessary to provide efficient and effective rail freight transportation services, railroad companies face a myriad of decisions and problems at different levels (strategic, tactical, and operational) that are, in general, much more complex than those found in competitive modes, especially trucking and air transportation.

One major difference is that railroads are responsible for building, operating, and maintaining all physical infrastructures related to their respective operations. It encompasses rail tracks on which trains travel and move; stations and yards where trains are formed and, railcars are loaded, unloaded, and switch trains; real-time dispatching, traffic control; and signalization. The number of decisions involved is enormous, usually interrelated, and very complex in nature. These include *strategic, capital investment decisions*, such as expanding the track network, duplicating and increasing capacities of lines; building, improving, and closing yards and stations, acquiring new locomotives, railcars, and other expensive equipments for many uses like loading and unloading; *tactical decisions*, related to train and locomotive schedules and maintenance plans; *operational decisions*, involving train dispatching, yard operations, empty railcar movement, locomotive scheduling, crew management, etc. Other modes of transportation, especially air transportation that has benefited significantly by the use of optimization methods (Yu [50]), do not face the same level of complexity as railroads in terms of its operational environment. Planning and scheduling problems arising in railroads are very large scale and quite complex. For a typical Class I U.S. railroad, its network comprises more than 10,000 miles of tracks, with several thousand weekly trains, and moving about 2,000–3,000 locomotives and no fewer than 80,000 railcars among about 200–300 yards, using some 5,000 train-crew members. Freight railroads operate every day of the year. Figure 1 depicts the U.S. railroad network, comprising over 140,000 miles of rail tracks over which seven major railroads (BNSF, CN, CP, CSX, FXE, KCS, NS, TFM and UP) operate.

From the demand side, each major railroad may receive some 4,000–7,000 new shipments (carloads) from its 5,000 customers every day, to be moved from some 1,000 origins to some other 2,000 destinations. With increasing global competitiveness and customer requirements, railroads must improve the planning and scheduling of their resources to provide fast and reliable services at competitive prices. Immediate actions are necessary for railroads not only to attract new customers, but also to maintain current customers, avoid losing them to truck companies due to poor services, inconsistent, and unreliable transit times, and uncompetitive increasing costs. Evidently, even in this challenging environment railroads also aim to increase their profitability.

This overall scenario is much more complex by far than the one faced by a trucking company or an airline carrier, even a major one. Thus, railroads desperately need computerized planning and scheduling tools so that they can improve their efficiency and profitability. However, the railroad industry has not benefited from the advances in operations research, and still relies mostly on simple simulation-based or manual methods for their planning

FIGURE 1. U.S. railroad network.



and scheduling. The research on railroad optimization problems has grown slowly; most recent contributions deal solely with simplified models or apply them only to unrealistic small instances. The majority of the models developed so far fail to incorporate the complex characteristics of real-life applications (Assad [7], Cordeau et al. [19], Haghani [26], Newman et al. [37]). More recently, the intense competition and the ever-increasing speed of computers have motivated the development and use of optimization models at various levels in railroad organizations. In addition, recently proposed models tend to exhibit an increased level of realism. As a result, there is a growing interest in utilizing optimization techniques for railroad problems. In the last few years, many advances in rail freight and passenger transportation have appeared in operations research literature. Recently, Canadian Pacific Railways was chosen as the winner of the *2003 Franz Edelman Award for Achievement in Operations Research and Management Sciences* for the project "Perfecting the Scheduled Railroad: Model-Driven Operating Plan Development" (Ireland et al. [30]). In this project, OR-based decision-support tools were developed and used to help create a more scheduled railroad, allowing cost savings of about US$ 170 million between 1999 and 2000.

It is important to give the reader a brief idea of what "running a scheduled railroad" means, because it seems to be an appealing approach to the challenging environment that railroads are currently confronted with, particularly increasing customer requirements. The schedule-based approach contrasts the traditional way railroads dispatch trains, known as the tonnage-based approach, which is based on the premise that reducing the number of trains operated, and creating longer trains, will minimize costs. In other words, with the tonnage-based approach trains can be delayed until they have sufficient tonnage to load them to capacity. In this practice, an operating plan may list a train that operates every day, but if there are insufficient railcars, the train can be delayed or even cancelled, with major prejudice to transit times. It causes unreliable services for the customers, may increase operating costs due to reduced utilization of rolling stock assets (locomotives and railcars), and increased idling and deadheading of equipments and crews. In contrast, in the scheduled-based approach trains run on time, as scheduled, even if they travel with light loads.

Though there is still controversy on the subject, this more disciplined approach is gaining favor in North American railroads to craft cost-effective and customer-effective operating plans, as pointed out by Ireland et al. [30]. It should be noted that the impacts in the operation practices are huge, because running lighter and more frequent trains with more rigid transit-time constraints may result in more trains being simultaneously assembled, dispatched, and running; more congested rail tracks and yards, potentially causing delays; more locomotives required to pull more trains; more empty railcars to be moved faster and in smaller batches in order to make them available for loading; and, as a result of all this, a risk of higher overall operating costs. To successfully implement these changes and cope with the increased level of synchronization in this challenging environment, railroads need effective tools to solve their planning and scheduling problems.

This chapter concerns the current state of the art of cutting-edge mathematical models and algorithms for solving some of the most important planning and scheduling problems faced by railroad companies that arise from the above issues in freight transportation. All problems considered here can be seen, and thus modeled, as some type of flow (shipments, trains, locomotives, railcars, crews) over an underlying network. Thus, the focus of this paper is on describing network-based models; the network structure allows us to have a better intuitive understanding of the problem and allows us to develop efficient and effective solution algorithms. A comprehensive review of the main concepts of network flow models and algorithms can be found in Ahuja et al. [4].

We have selected six of the most relevant planning and scheduling problems for railroads that could be effectively solved through network flow OR-based approaches. We give a detailed characterization of each problem, its context and importance to railroads, and the main issues, details, and constraints involved that make real instances extremely difficult

to be modeled and solved from the OR point of view. We also present the mathematical formulations, as well as cutting-edge algorithms to solve these instances arising in practice, and demonstrate significant savings in costs, some in excess of tens of millions of U.S. dollars annually.

The first and foremost planning problem to be solved in railroads is the *blocking problem*. This problem arises in the context that a railroad carries *millions* of shipments from their origins to their respective destinations. A typical shipment is composed of a set of individual *cars* that all share a common origin and destination. To reduce the handling of individual shipments as they travel, a set of shipments is *classified* (or grouped) together to create a *block*. The railroad blocking problem consists of identifying this blocking plan for all shipments in the railroad network so that the total transportation and handling cost is minimized.

The *yard location problem* is closely related to the railroad blocking problem. In this problem, we aim to find the best network configuration in terms of the number and location of yards where cars can be reclassified into new blocks and switch trains. Its importance is derived from the fact that yard locations have a major impact on blocking plans. However, in general, yard locations have been determined historically and have not changed much despite the significant cost implications, mostly fixed costs associated with equipment, infrastructure, and labor.

Given a blocking plan, developing a *train schedule* is perhaps the next-most important operational planning task faced by a railroad. The train scheduling problem is to determine train routes, their frequencies, and the days of operation of each train, aiming to minimize the cost of carrying blocks of cars from their origins to their destinations.

The *locomotive scheduling problem* consists of efficiently assigning different types of locomotives to the scheduled trains so they receive the desired pulling power while satisfying a variety of constraints, including fleet-size constraints on different locomotive types, and fueling and maintenance constraints.

The *train dispatching problem* aims to determine detailed train movements and timetables over a rail network under various operational constraints and restrictions in order to minimize both train deviations from the planned schedule and total delay as well. It is mostly motivated by the fact that trains traveling on single-line tracks can only overtake and cross each other at specific locations (sidings or meet-points), which are conveniently located at regular intervals along the line. Delays or deviations occur when trains traveling either in opposite directions or in the same direction meet, thus requiring one of the trains to be pulled over for the other to cross or overtake it.

The *crew scheduling problem* entails assigning crews to trains for each crew district, while complying with union rules, so that the crew costs are minimal and train delays due to crew unavailability are minimized.

In the following sections, each of the above problems is described in detail.

## 2. Railroad Blocking Problem

The *railroad blocking* problem is essentially a consolidation problem—how to consolidate a large number of shipments into *blocks* of shipments so as to reduce their individual handling as they travel from their origins to their destinations. Variants of this problem arise in several transportation sectors, including postal and package delivery companies, trucking companies, and the airline industry.

We will illustrate this through a problem faced by a large postal carrier such as the U.S. Postal Services (USPS). Consider, for example, the post office at a small city, which may receive several thousand envelopes every day to be mailed to thousands of other cities across the United States. It should be noted that handling each envelope individually—that is, sending it directly to its destination—would be highly inefficient because it would require

creating a separate bag for each of thousands of destinations and handling that many bags (some of which may be very small) as they travel to their destinations. Thus, the post office sorts (or consolidates) these individual envelopes into a small number of postal bags destined for different regions in the United States. For example, the post office may make a bag for Los Angeles (L.A.), and all envelopes headed for California can be placed into that bag. L.A. may receive several hundred bags from different cities in the United States that will be opened and resorted, and new bags will be created. At L.A., the post office may create a mailbag for Palo Alto; this bag will contain mail arriving at L.A. from several origins across the nation, all destined for Palo Alto.

Therefore, the purpose of the sorting process is to reduce the handling of the individual shipments to be moved. This sorting (or classification) process is a standard process in many situations where a large number of commodities travel over a network between several different origin-destination (OD) pairs of nodes. It arises in LTL (less than truckload) trucking industry where a large number of small shipments are consolidated into truckloads. An airline carrier's hub-and-spoke configuration also consolidates passenger traffic between thousands of origin-destination pairs into flight legs flown by different planes. It also arises in telecommunications where voice or data messages are consolidated and routed over the designed telecommunications network.

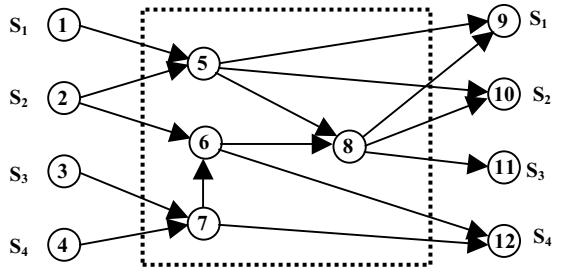
In railroads, this consolidation problem arises in the contexts of general carload (also known as general merchandise) and intermodal services. Carload services focus on the movement of small lots of railcars belonging to many different customers from a diverse set of origins to a diverse set of destinations. Intermodal transportation involves the movement of freight containers and over-the-road truck trailers on specially designed railcars between many different OD pairs. Both types of services are driven by the customer's need for shorter transit times and lower tolerances for service variances and delays. In these services, however, the amount of freight to be shipped from an origin station to a destination station does not usually allow the dispatching of a unit train, i.e., an entire train that is moved from the loading station to the unloading station without any intermediate handling, and usually containing a single product or a shipment for a single customer.

The number of shipments to be moved by railroads may be very high (about 50,000 in a month for a typical U.S. railroad). Given that it is physically and also economically impossible for a railroad to transport and handle each shipment individually, railroads have to consolidate shipments and respective railcars, route them across the network in blocks, through the rail yards and on the trains, in order to efficiently and effectively move them from their origins to their destinations. Due to this consolidation, some shipments may have to travel a longer distance with respect to their shortest-path direct routes from their respective origins to their destinations, as well as go through some intermediate handlings and switch trains (called *block swap*).

A railroad block is composed of several incoming and originating shipments that are grouped together. It is defined by an OD pair that may be different from the OD pairs of individual shipments in the block. Once a shipment is placed in a block, it will not be classified again until it reaches the destination of that block. Therefore, the blocks to be made are design arcs in a network representation. The railroad blocking problem is to construct the blocking network so that when all rail shipments are optimally routed over the blocking network, the total transportation cost, which is the weighted sum of the costs of distance traveled by cars and intermediate handlings, is minimum.

Figure 2 shows a sample blocking network, where three types of nodes can be identified: *origins* (nodes 1–4), where shipments $S_1$, $S_2$, $S_3$, and $S_4$ originate; *yards* (nodes 5–8), where shipments can be reblocked or reclassified; and *destinations* (nodes 9–12), where shipments $S_1$, $S_2$, $S_3$, and $S_4$ terminate. Each arc in the network represents a *block* with the origin at the tail of the arc and destination at the head of the arc. For example, arc (1, 5) signifies that a block will be made from station 1 to station 5. Shipments travel from their origins

FIGURE 2. An example of a blocking network.




to destinations over the arcs built in this blocking network. For example, a shipment $S_1$ that originates at node 1 and is destined for node 9 can be routed on two blocking paths: 1-5-9 and 1-5-8-9. The first blocking path flows on arcs (1, 5) and (5, 9) and involves one intermediate handling at node 5; the total cost is the sum of the flow costs on the two arcs with the handling cost at node 5. The second blocking path flows on arcs (1, 5), (5, 8), and (8, 9) and involves two intermediate handlings: at nodes 5 and 8. Another alternative, for example, would be to create a block arc (10, 9) linking node 10 to node 9. Similarly, a block arc (1, 8) would link node 1 directly to node 8 for all those shipments that flow on arc (1, 8), without intermediate handling at the yard located at node 5, even if this node is in the physical path between nodes 1 and 8. It should be noted that, in practice, yards can be origins as well as destinations, and nodes can send and receive shipments as well.

We shall now give the reader a rough estimate of the annual blocking cost a major U.S. railroad incurs and the potential savings that can be achieved. For a typical railroad, the flow cost per car per mile is \$0.50, and the average cost of reclassification is \$40 per car. A typical shipment may travel an average of 500 miles from its origin to its destination and may be classified 2.5 times. Assuming that the railroad ships 50,000 shipments per month with an average of 10 cars per shipment, we obtain 6 million cars per year traveling a total of 3 billion car-miles and 9 million reclassifications per year. This gives us a total annual cost of \$1.5 billion for car-miles traveled and \$1 billion in classifications. If these costs can be reduced by 2% to 5% by the use of an optimization-based algorithm, the annual savings will be of tens of millions of dollars for a single U.S. railroad and several hundred millions of dollars for all U.S. railroads.

## 2.1. Mathematical Formulation

Mathematically, the railroad blocking problem can be modeled as a *network design multicommodity flow problem*. In this problem, we have to select blocking arcs to build, and also decide how each shipment $k$ of a set $K$, consisting of $v_k$ railcars with the same origin $o(k)$ and the same destination $d(k)$, is grouped with other shipments in blocks and moved over these blocking arcs by different trains through some classification yards where blocking operations (i.e., the grouping of incoming cars for connection with outgoing trains) are performed.

Let $G = (N, A)$ be the blocking network, where $N$ is the set of all nodes denoting the stations where shipments originate, terminate, or switch trains and $A$ is the set of all potential blocking arcs in $N \times N$, i.e., $(i, j) \in A$ if a block can be built from node $i$ to node $j$. The cost of classifying a car at node $i$ is $h_i$ and the unit flow cost per car of shipping a block through arc $(i, j)$ is $m_{ij}$.

The blocking problem requires that all $v_k$ cars in each shipment $k \in K$ must be sent along a unique path in the blocking network. An efficient blocking solution needs to restrict the number of blocks made at a node $i \in N$ (given by $b_i$), as well as the number of cars it can handle (denoted by $d_i$). The maximum number of cars that can flow on block $(i, j) \in A$ is given by $u_{ij}$. These approximate estimates of yard capacities can be determined by its geometric configuration. There is a body of literature devoted exclusively to modeling yard

operations at a much more microscopic scheduling level, not taken into consideration in blocking modeling. We refer the interested reader to the survey work of Cordeau et al. [19].

The blocking problem has two sets of decisions: *design variables* $y_{ij} \in \{0,1\}$, concerning which links or arcs $(i,j) \in A$ to build; and *flow variables* $x_{ij}^k$, concerning how commodities should flow over the arcs in the network. In other words, $x_{ij}^k = v_k$ if shipment $k$ flows on the arc $(i,j) \in A$; 0 otherwise. Flow variables depend upon the design variables, as flow can take place only on the arcs that are built. The sets of arcs in $A$ entering and emanating from node $i$ are given by $I(i)$ and $O(i)$, respectively. The blocking problem can be formulated as the following mathematical programming problem.

$$\text{Minimize} \quad \sum_{k \in K} \sum_{(i,j) \in A} m_{ij} x_{ij}^k + \sum_{i \in N} \sum_{k \in K} \sum_{(i,j) \in O(i)} h_i x_{ij}^k \tag{1}$$

$$\text{subject to} \quad \sum_{(i,j) \in O(i)} x_{ij}^k - \sum_{(i,j) \in I(i)} x_{ij}^k = \begin{cases} v_k & \text{if } i = o(k) \\ 0 & \text{if } i \neq o(k) \text{ or } d(k), \\ -v_k & \text{if } i = d(k) \end{cases} \quad \forall k \in K \tag{2}$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij}, \quad \forall (i,j) \in A \tag{3}$$

$$\sum_{(i,j) \in O(i)} y_{ij} \leq b_i, \quad \forall i \in N \tag{4}$$

$$\sum_{k \in K} \sum_{(i,j) \in I(i)} y_{ij} \leq d_i, \quad \forall i \in N \tag{5}$$

$$y_{ij} \in \{0,1\}, \quad \forall (i,j) \in A \tag{6}$$

$$x_{ij}^k \in \{0, v_k\}, \quad \forall (i,j) \in A, \forall k \in K \tag{7}$$

In the above formulation, the objective function (1) seeks to minimize the weighted sum of flow costs on the arcs and classification costs at the nodes. The constraint (2), in conjunction with the constraint (7), ensure that each shipment flows on a single and unique path in the blocking network, and the constraint (3) ensures that shipments can flow on an arc only if it is built and the total flow on this arc is less than or equal to its flow capacity. The constraint (4) restricts the number of blocks created at a node and the constraint (5) restricts the number of car handlings at any node. Constraint (6) ensures that the arc design variables are binary variables.

The railroad blocking problem is a very large-scale network optimization problem. To illustrate this, let us consider a large U.S. railroad. It may have over 1,000 origins, over 300 yards, and over 2,000 destinations, and may send over 50,000 shipments in a month (a shipment may contain several cars usually ranging from 1 car to 50 cars). We can create a block from any origin to any yard, from any yard to any other yard, and from any yard to any destination. This gives a total of $1{,}000 \times 300$ (origins to yards) $+ 300 \times 300$ (yards to yards) $+ 300 \times 2{,}000$ (yards to destinations) $\approx 1$ million potential blocks that can be build. As there are over 50,000 shipments and about 1 million potential blocks, the number of decision variables for routing the shipments are in billions. Hence, this blocking problem contains about a million network design variables (either build or not build a block) and billions of flow variables (corresponding to the routing decisions of shipments over the blocking network).

Several attempts have been made in the past to model and develop an optimization-based algorithm to solve this problem (Barnhart et al. [8], Bodin et al. [9], Newton [38], and Newton et al. [39]). The largest problems solved so far are much below the size of real-world blocking problems found in practice. In addition, the optimization-based solution approaches described in the literature either do not produce high-quality solutions, or do not satisfy all the practical considerations necessary for implementability. To summarize, in spite of all achievements and advances in terms of optimization-based approaches, the

largest problems solved so far are much below the size of real-world blocking problems found in practice. However, a neighborhood search algorithm that has recently been proposed to solve the blocking problem appears to be very promising. We describe this algorithm next.

## 2.2. A Neighborhood Search Algorithm

A recent paper by Ahuja et al. [2] proposes very large-scale neighborhood (VLSN) search algorithms to solve real-life instances of the blocking problem. The VLSN search algorithms are neighborhood search algorithms where the size of the neighborhood is very large; in fact, it is so large that enumerating and evaluating all neighbors is prohibitively expensive. Using concepts from network flow theory (Ahuja et al. [4]), a VLSN search algorithm implicitly enumerates a vast neighborhood to identify an improved neighbor. We refer the reader to the paper by Ahuja et al. [5] for a detailed description of VLSN search algorithms.

In order to solve the railroad blocking problem, we make a simplifying assumption that makes it much easier to solve. We assume that the blocking problem needs to honor only the blocking capacities of nodes ($b_i$); the other capacity constraints—that is, car handling ($d_i$) and flows on arcs ($u_{ij}$)—can be ignored. These assumptions are based on the fact that in real-life blocking problems, even if these constraints are ignored, most nodes and arcs automatically satisfy them. Thus, in the absence of these capacity constraints, we need to honor only the constraint on the number of blocking arcs built at any node ($b_i$). Further, once we have decided what blocks will be built, all shipments can travel along their shortest paths from their origins to their respective destinations in the blocking network without violating any constraint. Hence, decision variables in the blocking problem reduce to what blocks should be built (i.e., which $y_{ij}$-variables equal 1), and thus flow variables $x_{ij}^k$ are automatically determined, because all shipments are always routed along their shortest paths in the blocking network built. The dropped constraints, i.e., flow capacity of blocking arcs and car-handling capacities at nodes, are handled by adding some Lagrangian relaxation-based penalties on their violation.

Figure 3 describes the VLSN search algorithm to solve the railroad blocking problem. It starts with a feasible solution of the blocking problem and iteratively improves the current blocking solution by replacing it by its neighbor until it cannot be improved. The neighborhood of a blocking solution is defined as consisting of all the blocking solutions that we can obtain by changing the blocks at one node of the blocking network. This algorithm has two important subroutines: constructing the initial feasible solution, and reoptimizing the blocking arcs emanating from a node. Each of them is described in detail below.

**2.2.1. Obtaining a Feasible Initial Solution.** We use a fairly simple construction heuristic to create the initial solution of the blocking network. This solution must ensure that the origin node for each shipment is connected to its destination node through a blocking path. To ensure this, a directed cycle that passes through all the yards exactly once and returns to the first yard is constructed (that is, a Hamiltonian cycle). Then, a blocking arc from each origin node (where some shipment starts) to the nearest yard is created.

FIGURE 3. The VLSN search algorithm for the railroad blocking problem.

```
algorithm VLSN-Blocking;
begin
    construct an initial blocking network and send all shipments along shortest paths in the
    blocking network;
    while the current solution is not locally optimal do
      for each node i ∈ N do {one pass}
          reoptimize the blocks emanating from node i;
          send all shipments along shortest paths in the updated blocking network;
      end; {one pass}
end;
```

Next, each destination node (where some shipment terminates) is connected to the nearest yard that still has some spare blocking capacity. After the blocking network is constructed, shipments are routed along their shortest paths over this network. In their computational experiments, Ahuja et al. [2] found that the proposed algorithm is fairly insensitive to the starting solution, and the effort to obtain a better initial solution does not have much impact on the final solution.

**2.2.2. Improvement Procedure.** An initial feasible solution is improved by exploring its neighborhood. The neighborhood of a blocking solution is defined as all the solutions that can be obtained by changing the blocks made at only one node and then rerouting all shipments along their updated shortest paths, which corresponds to the loop {*one pass*} of the algorithm in Figure 3. This neighborhood is very large, as there are $n$ ways to select the node for which we change the blocks emanating from it. Secondly, the number of ways we can select $k$ blocks out of a node with $p$ arcs emanating from it is ${}^{p}C_{k}$, which grows exponentially with $p$ and $k$. Clearly, enumerating all neighbors of a blocking solution and evaluating them explicitly is out of question.

Thus, we observe that the problem of identifying the best neighbor in the neighborhood (or a better neighbor than the current solution) is a difficult task. The formulation of this neighborhood problem as an integer programming problem could not be solved to optimality or near-optimality using state-of-the-art commercial integer programming software in several hours of computing time. This problem is believed to be NP-complete and, therefore, a heuristic solution is required. The paper Ahuja et al. [2] develops an algorithm that finds an improving neighbor heuristically. One pass of the improvement procedure for a selected node is composed of the following steps:

*Step* 1. Delete all blocking arcs emanating from the node.

*Step* 2. Reroute the traffic passing through the node along new shortest paths.

*Step* 3. Build new blocking arcs one at a time, causing maximum savings in the total cost.

*Step* 4. Reroute the shipments originating at (or passing through) this node over the shortest paths with the new blocking arcs.

As a result of Steps 1 and 2 of this algorithm being executed for a node $i$, all blocks made at this node are reoptimized, assuming that the blocks at other nodes do not change. The following *maximum savings rule* is considered in Steps 2 and 3: For every potential blocking arc that can be built at node $i$, it computes the savings in the total cost if that blocking arc is built and all shipments are rerouted to take advantage of this new blocking arc, and selects the blocking arc with the maximum savings. Step 4 reroutes all the shipments for which costs would decrease by using the newly built blocking arc.

**2.2.3. Handling Additional Practical Constraints.** The proposed VLSN search algorithm can also handle a variety of operational constraints and features that may arise due to historical restrictions, customer commitments, or trade-union-related reasons faced by railroads that cannot be easily incorporated into optimization-based formulations. These constraints and features include: (i) *blocks that must be made*, due to historical reasons and customer commitments; (ii) *blocks that must not be made* because they may be considered undesirable from crew, locomotive, or geographical-related issues; (iii) *some shipments must be sent on pre-specified blocks*; (iv) *not building small blocks*, because railroads do not like building small blocks when they travel large distances, as each block requires a certain amount of cost to build, and to subsequently handle it at yards, and small volume blocks do not justify this cost; (v) *give preference to some blocks* in order to avoid changing the blocking plan dramatically every month in response to variations in demand; (vi) *generate a blocking plan consistent with the train schedule*, that is, requiring that a block may not swap more than a certain number of trains, as each train swap requires some handling cost and delays the shipments in the block; (vii) *allow making incremental changes in a given*

*blocking solution*, because railroads may not be willing to make dramatic changes due to their associated risks in contrast with minor changes that can be easily implemented; (viii) *optimize a part of the blocking network*, which may be some selected nodes or a part of the railroad corridor.

Further details of the VLSN search algorithm can be found in Ahuja et al. [2]. We now describe the results of computational experiments for some real-world blocking problems.

## 2.3. Practical Experiments

The proposed VLSN search algorithm has been tested on real-world data provided by three major U.S. railroads: CSX Transportation, BNSF Railway, and NS Corporation. The results focus on comparing the blocking solutions of the algorithms with the blocking solutions currently used by the railroads. Two statistics are used to compare the solutions: average miles traveled by a car (which measures the cost of flow) and the number of reclassifications of cars (which measures the reclassification costs).

The first set of computational results compares VLSN search solutions with the corresponding railroads current solutions when it is allowed to make any block in the blocking network; it demonstrates, to some extent, the maximum savings possible due to changes in the blocking network. Table 1 gives these results for three railroads. The computational times given are taken on a 2.4 GHz Pentium IV computer with 2 GB RAM. We observe that the intermediate handlings can be dramatically reduced by using an optimized blocking plan.

We next report the computational results of the VLSN search algorithm when the blocking solution is restricted by the current train schedule, that is, requiring that a block may not swap more than a certain number of trains. In these tests, we prohibit those blocks that require more than two block swaps. Table 2 gives these results for two railroads.

The paper reports other results where incremental changes are made to a given blocking solution. The results are quite expressive. For instance, for CSX data, a change of only 5% in the current blocks results in 3.2% savings in car miles and 15.2% in reclassifications when solutions are unrestricted by the train schedule; and 2.9% and 12.6%, respectively, when solutions are restricted by the train schedule.

## 2.4. Concluding Remarks

The blocking problem is one of the most important railroad blocking problems, and its difficulty precluded the development of any optimal or near-optimal algorithm. As a result, almost all the railroads currently solve this problem manually. The VLSN search algorithm described here has a real potential to solve the blocking problem effectively and to be used in practice by railroads. It is also very flexible, and can easily incorporate a variety of practical considerations. The computational results of the algorithm are very impressive. For the data provided by the three major U.S. railroads, significant improvements have been obtained in short computing time. Another surprising outcome of the computational testing is that even a minor change in the blocking solution can lead to fairly dramatic improvement in reclassifications.

In the next section, we describe the yard location model, closely related to the railroad blocking problem because location and number of yards significantly affect how the blocking network is built, how and where blocks are made, and consequently, the detour of cars as

Table 1. Comparison of solutions unrestricted by the train schedule.

| | CSX | BNSF | NS |
|---|---|---|---|
| Savings in average car miles | 4.8% | 0.1% | 2.0% |
| Savings in reclassifications | 31.6% | 32.7% | 41.9% |
| Computational time | 2 hours | 5 minutes | 1 hour |

Table 2. Comparison of solutions restricted by the train schedule.

| | CSX | BNSF |
|---|---|---|
| Savings in average car miles | 4.4% | 5.2% |
| Savings in reclassifications | 23.2% | 19.8% |
| Savings in block swaps | 3.6% | 4.7% |
| Computational time | 2 hours | 5 minutes |

they travel from their origins to their destinations over the railroad network. In this problem, we aim to find the best network configuration in terms of the number and location of these yards.

## 3. Yard Location Problem

Yards play a vital role and can be considered the most crucial nodes of the blocking network because yards are where cars from inbound trains are reclassified and assembled into outbound blocks. Hence, these yards are usually referred as the nerve centers of the railroad system. From the definition of the blocking problem, it is immediately clear that the spatial configuration of yards is a very critical factor in identifying the resulting blocking network. Location and number of yards significantly affects the routes of cars and their detours as they travel in blocks from their origins to their destinations over the railroad network.

Yards are of three types: local, system, and regional. *Local yards* are small-scale yards where shipments from nonyard stations are assembled into blocks. *System* and *regional yards* are classified as hub yards (or hubs) that are occupied with large handling capacities. The yard location problem deals with the location of such yards. A major U.S. railroad has about 20–40 hub yards. Most of them were established several decades ago, considering the traffic pattern at that time. Due to changes in the traffic pattern through years, traffic growth, and railroad mergers, many yard locations are far from optimal. This implies that even the optimized blocking network with the current yard configuration would not give the best possible and least costly alternative to handle and route the shipments, unless the number of yards and their locations are reviewed. Costs involved in building a new yard, expanding the capacity of an existing yard, and operating the yards constitute a significant portion of capital investment. Hence, a railroad cannot afford to operate without analyzing the locations of its yards. This problem turns out to be the topmost strategic-level problem for a railroad in the process of finding the optimal way to ship goods from their origins to their destinations.

North American railroads have never paid attention to this strategic problem in the past. Even after the period of mergers, railroads have chosen to keep the yard locations as they are. As railroads have never studied the benefits of rearranging their yard configuration, the research on this particular issue has not attracted attention in the operations research literature. In this section, we discuss the first novel attempt to model and solve this problem.

Railroads potentially have much to gain if they can arrive at answers to the following questions:

(i) If some major yards need to be shut down, then what are the best yards to close with minimal impact on the transportation cost?

(ii) What are the trade-offs between the number of major yards and the transportation costs?

(iii) If some new yards are to be opened, then how many should be opened, and where?

(iv) If capacities of some yards can be increased, where should this capacity expansion should be done?

(v) What is the best network configuration? What are the impacts of operating a more centralized network in terms of fewer hub yards with increased capacities?

As railroads pay more attention to this strategic problem, they understand that some of the current major yards can be closed with little or no impact on the transportation cost, or some of the major hub yards can be relocated to significantly reduce the transportation costs. It is clear that any change in the yard locations would change the shipment plan of a railroad, because losing a current yard would invalidate the current blocking plan, or establishing a new yard location would make an improved blocking plan possible.

### 3.1. Mathematical Formulation

The yard locations are a major input to the railroad blocking problem, and the immediate effect of changing the yard locations will be changes in the blocking network. From this observation, it is obvious that yard location decisions should be considered along with their effects on the blocking network. Therefore, any mathematical formulation of the problem should include the blocking problem. The formulation approach we present here (Ahuja et al. [3]) is built on the same mathematical formulation of the blocking problem (Ahuja et al. [2]) presented in §2, with the addition of a high-level binary decision variable that identifies the selected yard locations in a given set of candidate yard locations. In general, similar to a simple facility location problem, the number of selected yard locations can be restricted either by a budget constraint related to the total cost of establishing and operating the yards, or by including these costs in the objective function. Here, we assume that the desired number of yards is known by the railroad and restricted to be less than that particular number ($p$). Without loss of generality, we assume a subset $\bar{N}$ of all yards $N$ that contains all candidate yard locations, and let $z_i$ equal 1 if we select a yard at node $i \in \bar{N}$ and 0 otherwise. Let $\bar{b}_i$ denote the blocking capacity of a candidate yard $i \in \bar{N}$ (that is, the maximum number of blocks that can be made at yard $i$) and $\bar{d}_i$ be the car-handling capacity of yard $i$—that is, the maximum number of cars that can pass through yard $i$. This yard location problem has the following formulation:

$$\text{Minimize} \quad \sum_{k \in K} \sum_{(i,j) \in A} m_{ij} x_{ij}^k + \sum_{i \in N} \sum_{k \in K} \sum_{(i,j) \in O(i)} h_i x_{ij}^k \tag{8}$$

$$\text{subject to} \quad \sum_{(i,j) \in O(i)} x_{ij}^k - \sum_{(i,j) \in I(i)} x_{ij}^k = \begin{cases} v_k & \text{if } i = o(k) \\ 0 & \text{if } i \neq o(k) \text{ or } d(k), \quad \forall k \in K \\ -v_k & \text{if } i = d(k) \end{cases} \tag{9}$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij} \quad \forall (i,j) \in A \tag{10}$$

$$\sum_{(i,j) \in O(i)} y_{ij} \leq b_i \quad \forall i \in N \backslash \bar{N} \tag{11}$$

$$\sum_{k \in K} \sum_{(i,j) \in I(i)} y_{ij} \leq d_i \quad \forall i \in N \backslash \bar{N} \tag{12}$$

$$\sum_{(i,j) \in O(i)} y_{ij} \leq b_i + \bar{b}_i z_i \quad \forall i \in \bar{N} \tag{13}$$

$$\sum_{k \in K} \sum_{(i,j) \in I(i)} x_{ij}^k \leq d_i + \bar{d}_i z_i \quad \forall i \in \bar{N} \tag{14}$$

$$\sum_{i \in \bar{N}} z_i \leq p \tag{15}$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A \tag{16}$$

$$x_{ij}^k \in \{0, v_k\} \quad \forall (i,j) \in A, \ \forall k \in K \tag{17}$$

$$z_i \in \{0,1\} \quad \forall i \in \bar{N}. \tag{18}$$

Among the existing constraints in the blocking formulation, the block-making (11) and car-handling (12) capacity constraints are now only defined for those locations that are not candidate yard locations. For candidate yard locations, constraints (13) and (14) ensure that the intermediate handlings can be performed only at those locations where yards have been located, and at other locations only originating or terminating shipments can be handled. Constraint (15) restricts the number of yards, and constraint (17) captures the integrality constraint for the new location variable. All other constraints are the same as in the blocking-problem formulation.

As seen in the previous section, the blocking formulation cannot be solved to optimality for a real-life problem because it contains too many variables and constraints. It is clear that the yard location formulation with an additional binary decision will be even more difficult to solve using commercial software. Hence, we need to develop heuristic algorithms to solve this problem. We describe one such algorithm below.

### 3.2. Solution Method

The solution method for the yard location problem (Ahuja et al. [3]) is based on the solution algorithm of the blocking problem (Ahuja et al. [2]) presented in §2. The analysis requires using the blocking algorithm as a subroutine. This subroutine may be executed several thousand times to perform a complete analysis over the current set of yard locations of a railroad. Therefore, we first develop a simplified and specialized version of the blocking problem that can be solved very efficiently. Using this specific implementation for the blocking problem, we develop algorithms to make decisions about closing and opening yards.

In the simplified version of the blocking problem, the assumptions result in a significant reduction in the problem size without sacrificing the accuracy of the solution with respect to the effects of yard configuration. The algorithm to be used as a subroutine in yard location analysis solves this simplified version of the railroad blocking problem, but in essence would find the major blocking network structure given a set of yards, yard capacities, and shipment data.

For this simplified version of the blocking problem we make the following assumptions:

(i) If a shipment has its origin/destination at a customer location, its origin/destination is mapped to the nearest yard (system, regional, or local). Observe that after this mapping, all traffic originates and ends at yards (system, regional and local).

(ii) We allow intermediate block switching only at hub yards. Local yards can build, send, and receive traffic, but do not allow intermediate switching.

(iii) We build mandatory blocks between every pair of hub yards. In addition, each local yard sends and receives a block from the nearest hub yard. A local yard can also build a block to another hub yard, provided there is sufficient car volume or that building such a block will result in substantial savings in car miles.

It should be noted that a key feature of the solution approach is to allow railroads to assess possible benefits of decreasing the number of yards in their system. Decreasing the number of yards in the rail network reduces the costs associated with operating yards (which should be considered as fixed-charge costs per year or per lifetime of the yard), and simultaneously increases the costs associated with routing the shipments over the rail network (which should be considered as variable costs per shipment) as the shipments might have to take longer routes when there are fewer yards for blocks to be made or switched. Another desirable feature would be to evaluate the trade-off between the savings associated with operating fewer yards and the increase in transportation costs due to running trains on longer routes. Railroads do not necessarily want to solve the problem to optimal yard locations, but rather want to benefit from this approach as a "what-if" analysis tool. As the problem on hand is a highly strategic one, they want to analyze the effects of changes in the most important inputs to the problem, such as number of yards, yard operating and establishment costs, and yard capacities. Thus, the overall analysis will be capable of showing the results with

different number of yards and with different cost structure settings, and hence help the decision maker assess the best setting. In the following section, we discuss the results of:

(i) a *drop algorithm* that would consider eliminating each yard location one by one and drop the yard with minimal impact on the cost;

(ii) an *add-drop algorithm* that would first eliminate a set of yard locations, then consider adding several yard locations one by one and add the yard with the maximum savings in costs; and

(iii) an *exchange algorithm* that would consider swapping an existing yard location with a potential yard location and perform the swap with the maximum benefit.

The *drop algorithm* starts with solving the blocking problem and sorting the hub yards in the ascending order of their switching volumes. It then picks the first $k$ hub yards in this order, deletes each of these hub yards one by one, and considers the impact on the solution statistics. The hub yard whose deletion results in the best overall statistics is dropped. We again re-sort the hub yards in the ascending order of their updated switching volumes, consider the new set of the first $k$ yards for deletion, and drop the yard with least impact on the overall statistics.

In the *add-drop algorithm*, we consider eliminating several yards one after another and adding some of them back to eventually find a solution with fewer yards than the initial solution. We drop $p$ hub yards one by one and add back $q$ hub yards ($p > q$) one by one. We always drop the hub yard that worsens the overall statistics by the minimum amount and add the hub yard that improves the overall statistics by the maximum amount.

The *pairwise exchange* algorithm attempts to improve a solution by exchanging a current hub yard with a nonhub yard. This algorithm can be used to improve a solution with a particular number of yards obtained by either the drop algorithm or the add-drop algorithm.

### 3.3. Computational Experiments

The computational experiments are from a data set provided by a major U.S. railroad with 27 current yard locations. The data set comprises 77 days of traffic and contains about 1.2 million cars. The railroad wants to make an analysis in order to close some of its current yards, and hence finds the optimal number of yards to keep and their respective locations. In order to help the railroad management to find the set of yards to keep in their system, previously mentioned algorithms are implemented as follows:

(i) In each iteration of the drop algorithm we consider the deletion of a subset of current yards of size $k$.

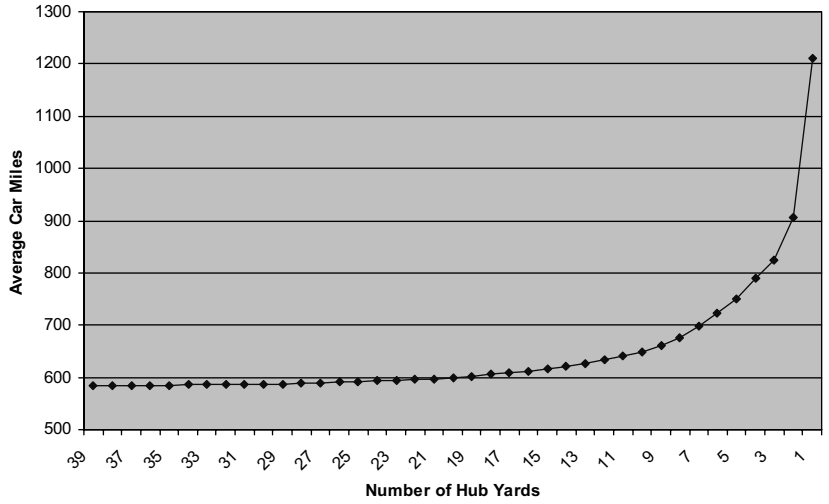
(ii) In the add-drop algorithm, we drop $p$ hub yards one by one and add back $q$ hub yards ($p > q$) one by one.

(iii) The pairwise exchange algorithm swaps a yard location with a nonyard location with the maximum saving in costs if any.

The implementations are tested on a data set of a major U.S. railroad that has specified a set of 39 candidate yard locations from which as many as 17 yards may be dropped, as specified by the railroad company. The add-drop algorithm is tested with $p = 2$ and $q = 1$, and $p = 3$ and $q = 2$. To the solutions obtained by this algorithm, we apply the pairwise exchange algorithm to improve the solution. We find that each of these algorithms give the same solution, which implies that the drop algorithm is quite robust. The running time of the drop algorithm is about six hours on a Pentium IV computer, which is actually a very reasonable time for such a huge strategic-level problem that is not solved on a frequent basis.

The results are evaluated based on two optimization criteria of the blocking problem: car miles and intermediate handlings. Charts of Figures 4 and 5 display average car miles and average intermediate handlings versus the number of yards. Apparently, as the number of hub yards decrease, the average car miles increases while the average intermediate handlings decreases (which is always one when there is only one hub yard in the system). The effect of working with more hubs is almost smooth for the average intermediate handlings. Although

FIGURE 4. Average car miles vs. number of hub yards.




the increase in average car miles is sharp when the number of hubs drops below 10, the change is almost negligible when the number of hubs is between 26 and 39. This shows that this railroad will be able to save a significant amount of money if 10 to 15 hub yards are eliminated, especially when the costs involved in operating yards are considered. When both terms are converted to monetary terms and combined into total transportation costs (see Figure 6), the lowest number of hub yards with almost negligible effect on the total transportation costs goes down to about 18. This analysis supports even more strongly the necessity of eliminating some yards in the system.

To understand the trade-off between deleting yards and the performance of the blocking network, we may combine the two performance criteria into one without even converting them into monetary terms. A general method accepted by the railroads to accomplish this is to find the weighted sum of car miles and intermediate handlings by mapping one unit of intermediate handling to a certain amount of car miles. Table 3 shows the average weighted car miles where one intermediate handling is assumed to be 100, 150, 200, 250, and 300 miles, along with the other figures already displayed in Figures 4, 5, and 6. With this type

FIGURE 5. Average car intermediate handlings vs. number of hub yards.

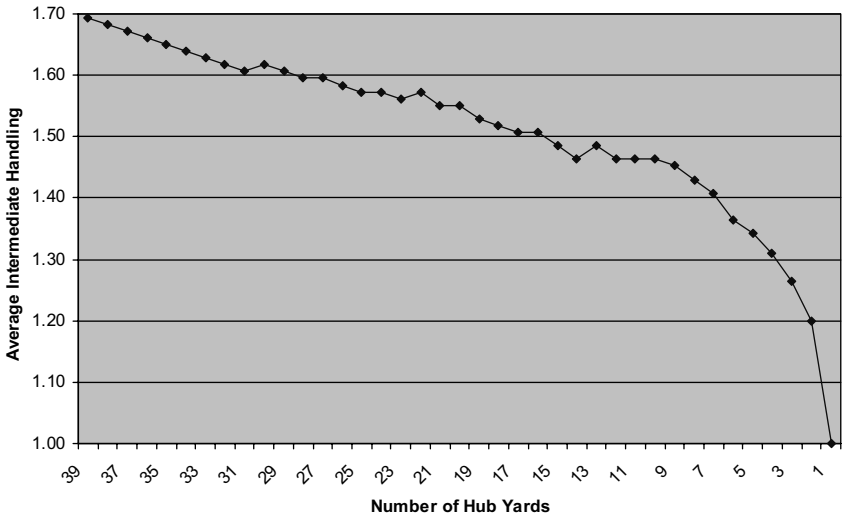


FIGURE 6. Total transportation costs vs. number of hub yards.

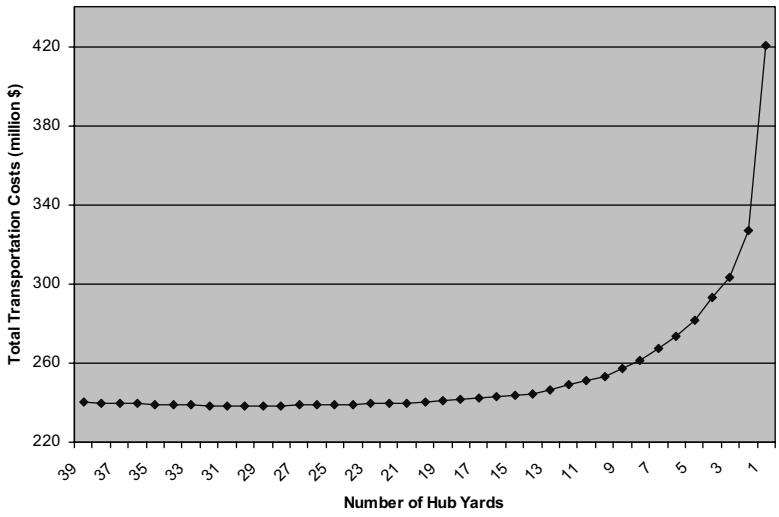



TABLE 3. Results with all performance criteria for different number of hub yards settings.

| Number of hubs | Average car miles | Average handling | Average weighted car miles | | | | | Total cost |
|---|---|---|---|---|---|---|---|---|
| | | | 100 | 150 | 200 | 250 | 300 | |
| 39 | 584.00 | 1.69 | 753.40 | 838.10 | 922.80 | 1,007.50 | 1,092.20 | 240,166,355 |
| 38 | 584.79 | 1.68 | 753.09 | 837.24 | 921.39 | 1,005.54 | 1,089.69 | 239,756,000 |
| 37 | 584.60 | 1.67 | 751.80 | 835.40 | 919.00 | 1,002.60 | 1,086.20 | 239,511,525 |
| 36 | 584.73 | 1.66 | 750.83 | 833.88 | 916.93 | 999.98 | 1,083.03 | 239,300,765 |
| 35 | 585.44 | 1.65 | 750.44 | 832.94 | 915.44 | 997.94 | 1,080.44 | 239,082,745 |
| 34 | 586.37 | 1.64 | 750.27 | 832.22 | 914.17 | 996.12 | 1,078.07 | 238,988,145 |
| 33 | 586.26 | 1.63 | 749.06 | 830.46 | 911.86 | 993.26 | 1,074.66 | 238,664,635 |
| 32 | 585.77 | 1.62 | **747.47** | 828.32 | 909.17 | 990.02 | 1,070.87 | **238,167,380** |
| 31 | 587.07 | 1.61 | 747.67 | **827.97** | 908.27 | 988.57 | 1,068.87 | 238,348,055 |
| 30 | 587.25 | 1.62 | 748.95 | 829.80 | 910.65 | 991.50 | 1,072.35 | 238,502,550 |
| 29 | 587.69 | 1.61 | 748.29 | 828.59 | 908.89 | 989.19 | 1,069.49 | 238,368,295 |
| 28 | 589.53 | 1.60 | 749.03 | 828.78 | 908.53 | 988.28 | 1,068.03 | 238,507,940 |
| 27 | 589.97 | 1.60 | 749.47 | 829.22 | 908.97 | 988.72 | 1,068.47 | 238,725,850 |
| 26 | 591.05 | 1.58 | 749.45 | 828.65 | 907.85 | 987.05 | 1,066.25 | 238,909,165 |
| 25 | 592.69 | 1.57 | 749.99 | 828.64 | **907.29** | 985.94 | 1,064.59 | 238,984,955 |
| 24 | 593.92 | 1.57 | 751.22 | 829.87 | 908.52 | 987.17 | 1,065.82 | 239,262,210 |
| 23 | 595.10 | 1.56 | 751.30 | 829.40 | 907.50 | 985.60 | 1,063.70 | 239,519,170 |
| 22 | 595.43 | 1.57 | 752.73 | 831.38 | 910.03 | 988.68 | 1,067.33 | 239,903,510 |
| 21 | 597.62 | 1.55 | 752.72 | 830.27 | 907.82 | 985.37 | 1,062.92 | 239,885,855 |
| 20 | 599.06 | 1.55 | 754.16 | 831.71 | 909.26 | 986.81 | 1,064.36 | 240,156,015 |
| 19 | 602.29 | 1.53 | 755.19 | 831.64 | 908.09 | **984.54** | 1,060.99 | 240,692,485 |
| 18 | 606.02 | 1.52 | 757.82 | 833.72 | 909.62 | 985.52 | 1,061.42 | 241,347,150 |
| 17 | 609.69 | 1.51 | 760.39 | 835.74 | 911.09 | 986.44 | 1,061.79 | 242,185,955 |
| 16 | 611.61 | 1.51 | 762.31 | 837.66 | 913.01 | 988.36 | 1,063.71 | 242,718,355 |
| 15 | 617.36 | 1.49 | 765.86 | 840.11 | 914.36 | 988.61 | 1,062.86 | 243,881,330 |
| 14 | 621.59 | 1.46 | 767.89 | 841.04 | 914.19 | 987.34 | **1,060.49** | 244,535,885 |
| 13 | 625.99 | 1.49 | 774.49 | 848.74 | 922.99 | 997.24 | 1,071.49 | 246,664,000 |
| 12 | 634.33 | 1.46 | 780.63 | 853.78 | 926.93 | 1,000.08 | 1,073.23 | 248,860,755 |
| 11 | 642.40 | 1.46 | 788.70 | 861.85 | 935.00 | 1,008.15 | 1,081.30 | 251,240,880 |
| 10 | 649.11 | 1.46 | 795.41 | 868.56 | 941.71 | 1,014.86 | 1,088.01 | 253,338,030 |

of analysis, railroads may see the best yard configuration setting (with the number of hubs and their locations) under different cost scenarios even though still not including the costs involved in operating (or even establishing) yards. A boldface number corresponds to the yard configuration with the minimum weighted car miles of its column. For example, if one intermediate handling is equivalent to 300 car miles, then the optimal solution corresponds to having 14 hub yards.

The results show that even if we ignore the savings of closing hub yards, we can always close some hub yards and reduce overall costs. As we close yards, the car miles increase, but intermediate handlings decrease in compensation, and there is very marginal impact to the overall statistics.

Further analysis is to be made by the railroad management by setting appropriate cost terms per car mile and per intermediate handling. Even the analysis without considering the savings due to yard closures reveals a significant amount of savings in total. Hence, it is apparent that the railroads should expect a huge amount of savings by reconsidering the number and locations of their hub yards.

### 3.4. Concluding Remarks

The yard location problem is one of the topmost strategic-level problems of the railroads, not only due to a high portion of costs involved, but also due to the fact that the blocking network with the yard locations forms the backbone of the transportation operations. Therefore, when the problem is not analyzed from an optimization point of view, the decisions will lead to deficiencies in railroad economies.

To our knowledge, the solution method presented here (although not very complicated once we have an efficient solution method for the blocking problem) is the first attempt in the operations research literature for the railroad yard location problem. Although this strategic problem does not require a real-time decision tool, the solution methods should be simple and fast enough to allow the railroad management to solve and analyze the problem under several scenarios (e.g., yard capacities, different set of candidate yard locations) and for different environmental parameters (e.g., different costs associated with establishing yards or capacity expansions, equivalence of one intermediate handling in car miles). Hence, the railroads should be able to assess such decisions more frequently and be able to respond to the changes in the transportation market and economy.

## 4. Train Scheduling Problem

Once a railroad has identified a blocking plan, it must design a train schedule so that trains can efficiently carry blocks from their origins to their destinations. The train schedule design problem, henceforth referred to as the *train scheduling* problem, determines: (i) how many trains to run; (ii) the origin, destination, and route of each train; (iii) the train arrival and departure times for each station at which it stops; (iv) the weekly operating schedule for each train; and (v) the assignment of blocks of cars to trains, so that the total cost of transportation (including distance, crew, locomotives, fuel, car hire, etc.) is the minimum possible. These decisions need to be made on a weekly basis, as a train schedule repeats every week.

Major U.S. railroads incur considerable expenses to maintain their operations. One of the largest American railroad companies, BNSF Railway, has approximately 400 long-distance trains. Each train runs an average of five days per week, with an average travel time of one day from its origin to its destination. Hence, a typical train runs about 52 (weeks/year) $\times$ 5 (trains per week) = 260 days. Daily operating costs per train include at least \$1,500 for the locomotive and about \$1,500 for the crew. Thus, the average annual cost to operate a single train is approximately \$0.8 million ($\$3,000 \times 260$ days). An optimization approach that eliminates 10 to 20 trains from the schedule may yield annual savings of \$12–\$20 million. In addition, an optimized train schedule may lead to decreased car flow times, thus reducing car hire cost and improving customer service.

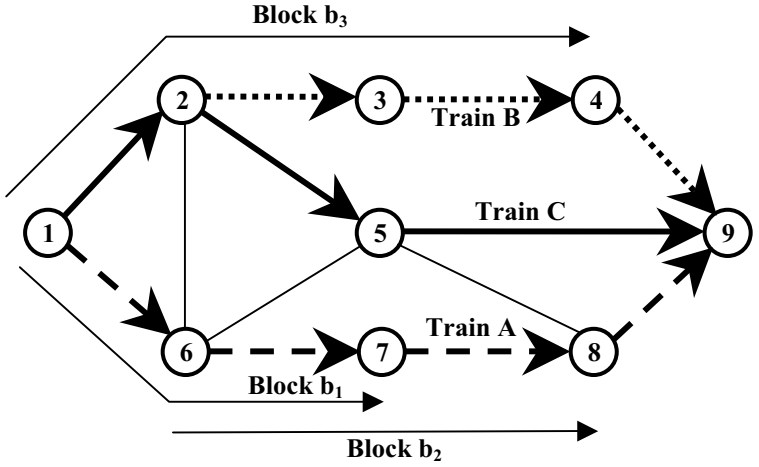
The train scheduling problem is a very large-scale network optimization problem with trillions of decision variables. Its difficulty precluded the development of train scheduling algorithms that are much needed in practical deployment. Some attempts have been made in the past to solve this problem. Early research on train scheduling includes papers by Assad [7], Crainic et al. [21], Crainic and Rosseau [20], Haghani [27], Keaton [31], Keaton [32], and Keaton [33]. These papers divide the train scheduling problem into two separate stages: the train design problem and the block routing problem, which are solved separately. The train design problem determines train routes and the block routing problem routes blocks over the trains formed. An iterative procedure solves each of the two stages in succession, using the solution from the other stage to guide the next iteration. A more recent paper by Gorman [25] considers the integrated train schedule problem, but solves it by a rather crude heuristic. None of these attempts has produced a solution procedure that railroads can use, because they are not scalable for realistically large train scheduling problems, or they ignore the practical realities necessary to generate implementable solutions.

The train scheduling problem comprises three entities: (i) the physical (railroad) network, (ii) trains that travel on the physical network, and (iii) blocks that travel on the trains. These entities are illustrated in Figure 7, which shows a simple example of a railroad network with nine nodes (denoting stations or yards) and the respective train paths and block paths. Train $A$ starts at node 1, follows the route 1-6-7-8-9, and terminates at node 9. Train $B$ follows the route 2-3-4-9, and train $C$ traverses the path 1-2-5-9. Three blocks of cars, $b_1$, $b_2$, $b_3$, travel on these trains. Block $b_1$ takes train $A$ over the train segments 1-6-7, while block $b_2$ takes the same train over the segments 6-7-8. So, as a train travels from its origin to its destination, it picks up blocks at various nodes it visits and may also drop off blocks at those nodes. Typically, several blocks ride on a train at any segment. Likewise, a block may travel on several trains as it goes from its origin to its destination. For example, block $b_3$ starts at node 1 and travels on train $C$ on the segment 1-2. It is off-loaded by train $C$ at node 2, where it is picked up by train $B$ and then carried from node 2 to node 4, its final destination. The transfer of a block from one train to another is called a *block swap*. Among the blocks in our example, only $b_3$ performs a block swap.

The preceding example provides a brief overview of the structural components of the train scheduling problem. There are five types of decisions to be made in this problem:

(i) What is the number of trains to run?
(ii) What are the origin, destination, and route of each train?
(iii) What are the arrival and departure times at each station that a train visits?

FIGURE 7. Illustrating the interplay between physical network, trains, and blocks.



(iv) What is the frequency of each train? In other words, how often should it run in a week, and on what days?

(v) How should blocks be assigned to trains?

A feasible solution of the train scheduling problem must honor the following constraints:

(i) The number of trains starting at any station within a specified time window (say, every four hours) is limited. This constraint also applies to the number of trains passing through or terminating at a specific station during a given period. It distributes arrivals and departures evenly throughout the day, thereby preventing congestion.

(ii) The number of trains passing through an arc of the physical network is limited. These constraints are called the *line capacity constraints*.

(iii) When blocks are assigned to trains, the number of cars traveling on any train segment cannot exceed a prespecified upper limit. In addition, the number of cars traveling on any segment must be at least a prespecified lower limit. These constraints are called the *maximum and minimum train size constraints*.

The goal of the train scheduling problem is to minimize the weighted sum of the following cost terms: (i) the total distance traveled by cars on trains, (ii) the total flow time of cars as they travel from their origins to their destinations, (iii) the total number of block swaps of cars between trains, (iv) the number of locomotives and crew required, and (v) the total number of trains formed during the week.

In a high-quality train scheduling solution, it is also desirable to minimize the number of train stops at stations, to maintain the consistency of blocks to trains on different days of the week, and to generate crew-friendly and locomotive-friendly train schedules.

The train scheduling problem arises in several areas in a railroad. The version of the problem we discuss here is called *zero-base train scheduling*. In this problem, we determine an entirely new train schedule for a railroad. Railroads typically solve the zero-base train scheduling problem once every several years, when railroad mergers take place or when a major shift in the traffic pattern renders a current train schedule obsolete. However, this task usually has to be performed manually, leaving much room for improvement. If a computerized optimization tool for train scheduling is available, then railroads can do zero-base train scheduling more often—perhaps twice every year. Regular (periodic) changes using the zero-base train scheduling will reduce operational costs and improve efficiency of railroads.

## 4.1. Mathematical Formulation

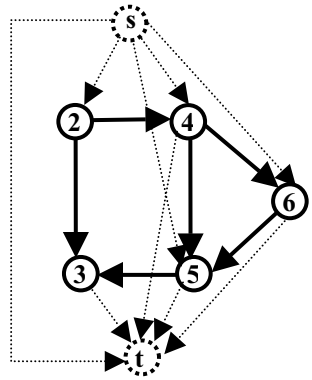
The train scheduling problem is a very large-scale integer programming problem. In this section, we consider a highly simplified version of the problem, in which we assume that each train runs every day of the week (which thus reduces the weekly problem to an equivalent and simpler daily problem), ignore the train timings, and show that even for this case the formulation contains billions of variables and millions of constraints. Figure 8 displays a sample physical network through which trains flow. Let $N$ denote the set of nodes in the network and $A$ denote the set of arcs. We have added a dummy node $s$ to represent train originations, and a dummy node $t$ to represent train terminations. We assume that at source node $s$, a unit flow of distinct commodities, each one representing a train, is available to be sent to sink node $t$. A commodity either flows through the network (implying that the corresponding train is formed), or flows through the arc $(s, t)$ (implying that the corresponding train is not formed).

Our formulation features two types of decision variables:

$y_a^k$ : 1 if train $k \in K$ traverses arc $a \in A$, and 0 otherwise; and

$x_{ka}^b$ : 1 if block $b \in B$ flows on arc $a \in A$ of train $k \in K$, and 0 otherwise.

FIGURE 8. Network representation of train paths.




The integer programming formulation for the train scheduling problem is as follows:

$$\text{Minimize} \quad \sum_{k\in K}\sum_{a\in A} f_a^k y_a^k + \sum_{b\in B}\sum_{k\in K}\sum_{a\in A} c_{ka}^b x_{ka}^b + \sum_{b\in B}\sum_{\substack{i\in N \\ i\neq o(b),\, i\neq d(b)}} s_i^b \left| \sum_{a\in O(i)} x_{ka}^b - \sum_{a\in I(i)} x_{ka}^b \right| \tag{19}$$

$$\text{subject to} \quad \sum_{a\in O(i)} y_a^k - \sum_{a\in I(i)} y_a^k = \begin{cases} 1, & \text{if } i = s \\ 0, & \text{if } i \neq s,t \\ -1, & \text{if } i = t \end{cases} \qquad \forall i \in N,\ \forall k \in K \tag{20}$$

$$\sum_{k\in K}\sum_{a=(s,\,i)} y_a^k \le T_i^+, \quad \forall i \in N \tag{21}$$

$$\sum_{k\in K}\sum_{a=(i,\,t)} y_a^k \le T_i^-, \quad \forall i \in N \tag{22}$$

$$\sum_{k\in K} y_k^a \ge T_a, \quad \forall a \in A \tag{23}$$

$$\sum_{k\in K}\sum_{a\in O(i)} x_{ka}^b - \sum_{k\in K}\sum_{a\in I(i)} x_{ka}^b = \begin{cases} 1 & \text{if } i = o(b) \\ 0 & \text{if } i \neq o(b) \text{ or } d(b), \\ -1 & \text{if } i = d(b) \end{cases} \quad \forall i \in N,\ \forall b \in B \tag{24}$$

$$\sum_{b\in B} x_{ka}^b \le M y_a^k, \quad \forall a \in A,\ \forall k \in K \tag{25}$$

$$L_a^k y_a^k \le \sum_{b\in B} v^b x_{ka}^b \le U_a^k y_a^k, \quad \forall k \in K,\ \forall a \in A \tag{26}$$

$$y_a^k \in \{0,1\}, \quad \forall k \in K,\ \forall a \in A \tag{27}$$

$$y_{ka}^b \in \{0,1\}, \quad \forall k \in K,\ \forall a \in A,\ \forall b \in B. \tag{28}$$

In the above formulation, $M$ denotes a large number, and the sets $I(i)$ and $O(i)$, respectively, denote the sets of incoming and outgoing arcs at node $i \in N$. Similarly, $o(b)$ and $d(b)$, respectively, denote the origin and destination of block $b$. The other notation is self-evident, and we will not describe it here for the sake of brevity. The objective function (19) is the sum of three cost terms: (i) the cost of running trains, (ii) the cost of flowing blocks, and (iii) the cost of block swaps. Constraint (20) states the flow balance constraints for trains.

Constraints (21) (or (22)) state that the number of trains originating (or terminating) at node $i$ per day is no more than $T_i^+$ (or $T_i^-$). Constraint (23) captures the restriction that the number of trains traversing arc $a$ per day does not exceed $T_a$. Constraint (24) states the flow balance constraints for blocks, and (25) ensures that blocks flow on an arc only if a train traverses that arc. Finally, constraints (26) and (27) represent the train-arc design variables and block-to-train assignment variables.

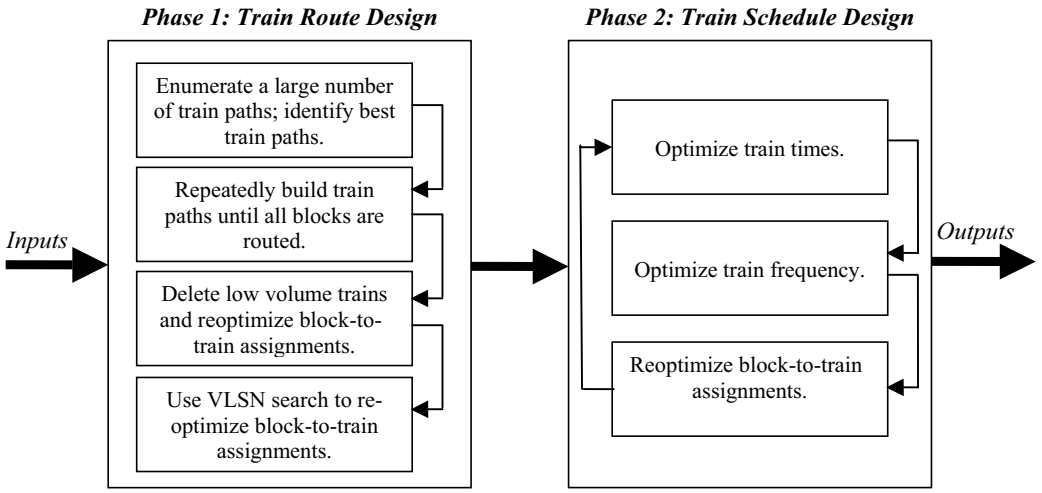
Let us now estimate the size of this IP formulation. For BNSF Railway, the physical network has about 2,000 nodes (where trains can originate, terminate or stop), 12,000 arcs, 1,300 blocks, and 400 trains. For this case, the above formulation contains about 5 million $y_a^k$ binary variables and about 7 billion $x_{ka}^b$ binary variables, as well as approximately 20 million constraints. Integer programming problems of this scale greatly exceed the capacity of existing commercial software.

## 4.2. Solution Approach

The integer programming formulation demonstrates that the train scheduling problem is a very large-scale discrete optimization problem containing billions of integer variables, even though we ignore some of the critical decision variables, such as train timings and train frequencies. If we consider the weekly scheduling problem instead of the daily scheduling problem, and consider train timings and frequencies, then the resulting integer programming formulation of the train scheduling problem will have trillions of integer decision variables. It is a common approach for problems of this magnitude to be solved by decomposition. In a decomposition approach, instead of determining the optimal values of all decision variables using a single large decision model, we solve a sequence of smaller decision models so that the solutions of these models, when taken collectively, provide a good solution to the integrated model. Decomposition allows us to obtain satisfactory solutions to intractable problems within acceptable running times. However, the method we use to decompose a large problem into a series of smaller problems determines the success of this approach. The approach we discuss here solves the problem in two phases. In the first phase, we determine the train network—i.e., train origins, destinations, and routes as well as block-to-train assignments—that embraces the blocking network; then train frequencies and timings are determined for this train network. The algorithm is summarized in Figure 9.

**4.2.1. Phase I: Train Route Design.** In this phase, we determine train origins, destinations, and routes, and assign blocks to trains assuming that each train runs every day

FIGURE 9. A schematic representation of the algorithm.



of the week. Our objective is to create a set of daily trains so that all the blocks can be feasibly carried by the train network. The following objective function terms are considered in determining this one-day train schedule:

(i) number of train starts,
(ii) total train miles,
(iii) average number of cars per train mile, and
(iv) number of block swaps.

The algorithm in this phase encompasses two heuristics: construction and improvement. The construction heuristic starts with a null train network and builds trains one by one until all the blocks are routed. The improvement heuristic considers the possible deletions of trains so that the associated objective function terms are improved while there are still sufficient trains to route the blocks. The main steps of these heuristics are summarized next. The steps of the construction heuristic are as follows:

*Step* 1. Map blocks on the shortest paths in physical network.

*Step* 2. Starting from each of the block destination, enumerate all the possible trains with the assumption that a train can only originate at a block origin.

*Step* 3. Attach blocks to possible trains, if the origin and destination of a block is on the route of the train, and a train path exists for a block from its origin to the station where it can take the train under consideration.

*Step* 4. Determine goodness of each possible train.

*Step* 5. Create a set of link-disjoint best trains from the set of possible trains.

*Step* 6. Continue the process till all the blocks are routed.

The steps of the improvement (delete) heuristic are as follows:

*Step* 1. Consider each train as a candidate for deletion if per-mile car volume is less than a threshold value.

*Step* 2. Search alternate train route for blocks that are assigned to the candidate train for deletion using the following criteria:

- There should not be more than $t$ trains in new train path for a block.
- A block should not be detoured on a path with distance more than $x$ % of shortest path distance.

*Step* 3. Delete the candidate train and reroute blocks if alternate train routes are available.

*Step* 4. Repeat the algorithm until no more deletion in Step 2 is possible.

**4.2.2. Phase II: Train Schedule Design.** The Phase II algorithm takes as an input the output of Phase I, the train origins, destinations, and routes, and block-to-train assignments; and determines the frequencies and train timings. The algorithm considers a seven-day train space-time network where blocks are routed over the network. We define *train leg* to be a train running on a particular day of the week. For example, if a train runs seven days a week, then it will have seven legs. The network consists of nodes representing physical stops of the trains. The arcs represent the path (more precisely, the path between two consecutive stops) of the trains. At any stop of the train, additional nodes are created to represent stops of individual train legs. These nodes are sorted in the order of departure time of the corresponding train legs. Arcs are introduced to connect these additional nodes in this sorted order.

Phase II consists of three subphases. The first subphase determines the initial train timings and assumes that each train runs every day of the week. The second subphase determines train frequencies, and the third phase reoptimizes train timings by considering the block-to-train assignments.

When we start Phase II, there are no trains added to the space-time network. In the first subphase, we add trains to the space-time network. There are typically two types of trains: those whose times have been specified and those whose times need to be determined. We add the first type of trains and route those blocks that can be completely routed using these types of trains. We next consider the second type of trains and arrange them in the

descending order of the total car miles on them. We then consider each train one by one in this order and determine its time. We use a simple enumeration approach to determine the train timings. We assume that each train runs by the hour, and thus the possible departure times of the train are 00, 01, 02, ..., 23 hours. We also assume that once the departure time of the train has been specified, all other arrival and departure times can be automatically determined. Among these 24 values of the departure time, we consider the departure time for which the total car flow time is minimum.

In the first subphase, we assume that each train runs every day, that is, it has seven legs. However, this may result in some train legs with small volumes. In the second subphase, we determine train frequencies by eliminating such train legs. We repeatedly delete the train legs with the smallest volume and reroute traffic riding on it onto trains on other days. We apply this until each train leg has sufficiently large volume (e.g., at least 25 cars). We consider a particular number of (say 10) train legs with smallest volume, consider each one of them for potential deletion and its impact on the total car flow time, and delete the train leg with the least impact on the total car flow time.

In the second subphase, we change train frequencies, which may make our train timings suboptimal. In the third subphase, we reoptimize train timings by reconsidering the block-to-train assignments. We consider each train one by one in some order and determine its optimal starting time, assuming all other trains do not change their starting times while allowing changes in block assignments. We consider the possible departure times of the train to be 00, 01, 02, ..., 23 hours, and select the time giving the minimum total car flow time.

**4.2.3. Incorporating Additional Practical Considerations.** Railroads often have some business rules, and they want their plans and schedules to honor those rules. Adherence to these business rules is essential, for otherwise these schedules will not be implemented. To satisfy these business rules and practices, we incorporate the following features in our algorithms:

(i) some trains must or must not be built;
(ii) some train segments or train routes may be given additional preference;
(iii) some train segments or train routes may be undesirable;
(iv) some blocks or traffic may be preassigned to some trains; and
(v) there may be restrictions on a shipment's value, height, and weight that trains must honor.

These features can easily be accommodated by the above heuristics.

## 4.3. Computational Results

The train scheduling algorithm is implemented to a data set of Norfolk Southern. The results for the performance criteria specified by the railroad are shown in Table 4. The improvements are calculated with respect to the railroad's current solution. The most significant outcome of these results is that the algorithm improves the solution for all criteria without having to consider the trade-offs based on the conflicts among these criteria. In addition, such improvement figures are found by the railroad to be very promising.

Table 4. Results of the algorithm compared to railroad's current solution.

| Statistics | % Improvement |
|---|---|
| Number of train starts | 20.2 |
| Total train miles | 12.4 |
| Total train travel time | 15.6 |
| Average car miles | 1.0 |
| Average car days | 1.8 |
| Average car block swap | 12.7 |

### 4.4. Concluding Remarks

Train scheduling is one of the most complex problems among the ones we discuss in this chapter. It is not only the size of the problem, but also integration of several issues that makes the problem complicated. The problem is involved with various decisions that are interrelated with each other, which increases the difficulty in modeling. In addition, an applicable solution targets several objectives to be achieved simultaneously, some of which are conflicting with each other, which increases the difficulty in solving the problem.

In the past, railroads have never been able to approach this problem from an optimization point of view because previous attempts brought poor-quality solutions. Solutions they produce cannot be implemented because they ignore some of the practical constraints. The algorithm we present here not only considers all realistic constraints, but also improves various performance criteria simultaneously. Solutions presented here evidently show that the railroads can improve their train schedules significantly and hence achieve substantial savings in their capital costs by running far fewer trains, and in operational costs by running these trains on shorter routes and with even less frequent block swaps.

## 5. The Locomotive Scheduling Problem

The *locomotive scheduling* problem is to optimally assign a set of locomotives (also known as a *consist*) to cover all scheduled trains in a given time horizon and to route all these locomotives over the network, while satisfying fleet-size constraints on different locomotive types, fueling and maintenance requirements of the locomotives, pulling-power requirements of all trains, and compatibility restrictions between locomotives and trains. Trains may have different weekly frequencies, i.e., some trains may run every day, while others may run less frequently. The overall aim is to improve the average locomotive utilization and thus reduce the number of locomotives needed.

Locomotive scheduling can be studied at two levels: planning or operational. At the *planning level*, we need to decide the set of locomotives to be assigned to each of the various trains considering the different types available (such as SD40, SD50, AC44), with different horsepower, pulling, and cost characteristics. Some locomotives may *deadhead* on trains, i.e., be pulled like railcars by a set of active locomotives from one place to another, instead of pulling the train. Deadheading allows extra locomotives to be moved as part of a train from locations where they are in surplus to other locations where they are in short supply. Locomotives also *light travel*, i.e., they travel on their own (i.e., not pulling any train or railcars) between different stations to reposition themselves between two successive assignments to trains. Light travel is different from deadheading because it is not limited by the train schedule; in addition, trains travel faster than deadheading. However, light travel is more costly, as a crew is required to operate the light-traveling locomotives, and this movement does not generate any revenue because no cars are attached.

At the *operational level*, we are also concerned with which specific units of the available fleet get assigned to each train. In other words, we need to assign locomotive *tail numbers* (which uniquely identify each individual locomotive) to trains. To do so, we need to take into account the fueling requirements of locomotives. For example, a locomotive may have enough fuel to travel 300 more miles, but a candidate train will not encounter any fueling station for the next 400 miles; clearly, we cannot assign this locomotive to the train. Another important issue is maintenance. Each locomotive needs maintenance at periodic intervals; for instance, regular maintenance every 3,000 miles and major maintenance every 10,000 miles. This maintenance can be done only at some specific locations, called *shops*. Thus, if a locomotive is due for maintenance, then we cannot assign it to a train that takes it too far from a shop, as it cannot return before its due maintenance. Another factor that might be considered in locomotive scheduling at the operational level is uncertainty in train times. Contrary to the planning level, in which it is assumed that all trains run on time, at the

operational level we take into consideration that, in practice, trains are often late. How to account for delayed trains and update locomotive assignments accordingly, with minimum cost impact to the locomotive-related costs, is an important issue that must be considered in an operational-level model.

To assign the available locomotives to trains, we need to account for *consist-busting*. Whenever a train arrives at its destination, its consist (i.e., its set of locomotives) is either assigned to an outbound train in its entirety, or it goes to the pool of locomotives where new consists are formed. In the first case, a *train-to-train connection* occurs between the inbound and the outbound trains, and no consist-busting takes place. In the second case, i.e., some one or more locomotives of a consist are assigned to a departing train while others are assigned to another departing train, consist-busting takes place. Consist-busting entails merging locomotives from inbound trains and regrouping them to make new consists for outbound trains. This may be undesirable for railroads for several reasons: First, it requires additional locomotive and crew time to execute the more complex and time-consuming moves, which requires decoupling and moving locomotives individually; second, it often results in outbound trains getting their locomotives from several inbound trains. If any of these inbound trains is delayed, the outbound train is also delayed, which may lead to further delays down the line. In an ideal schedule we try to maximize the train-to-train connections of locomotives, and thus minimize consist-busting. It should also be noticed that assigning a single, but more powerful, locomotive to a train is undesirable because if that locomotive breaks down, the train gets stranded on the track and blocks the movement of other trains.

Locomotive scheduling is among the most important problems in railroad scheduling, as it involves a very expensive asset. A lack of a planning and scheduling tool for this problem may result in a highly inefficient utilization of this resource. For example, CSX Transportation, a major U.S. railroad, has a fleet of about 3,600 locomotives. This translates into a capital investment of over $6 billion, in addition to over $700 million in yearly maintenance and operational costs. We have observed that the average utilization of locomotives (i.e., when they are actually pulling trains) is around 50%; at other times the locomotives are idling, deadheading, or light traveling. An improvement of 5% in average locomotive utilization may reduce the number of locomotives needed to pull the scheduled trains by 400 (from 3,600 to 3,200), which results in a savings of over $100 million per year in operational costs alone.

Due to its importance, there is a fair share of literature devoted to locomotive scheduling problems. The paper Cordeau et al. [19] presents an excellent survey of existing locomotive scheduling models and algorithms for this problem. Most existing models assume that there are multiple locomotive types available for assignment. However, some models assume that a train is assigned only one locomotive; we refer to such models as *single-locomotive assignment models*. Single-locomotive assignment models can be formulated as variants of integer multicommodity flow problems. Some papers on single-locomotive assignment models are due to Fischetti and Toth [22], Forbes et al. [24], and Wright [49]. Single-locomotive assignment models may be appropriate for some European railroad companies, but are not suited for U.S. railroad companies, because most trains are assigned multiple locomotives. We refer to the models in which multiple locomotives can be assigned to a train as *multiple-locomotive assignment models*. These models have been studied by Chih et al. [16], Florian et al. [23], Nou et al. [40], Smith and Sheffi [47], Ziarati et al. [52], and Ziarati et al. [51]. Most of the proposed models concern locomotive assignment for freight trains. There is little work on locomotive assignment for passenger trains (Cordeau et al. [18] and Ramani [41]).

Though there exist a few programming packages for locomotive scheduling problems, none of them are satisfactory to U.S. railroad companies. Some of the main reasons why existing models do not produce good usable schedules are that: (i) they assign locomotives, not consists, to the trains; (ii) they do not handle deadheading and light travel of locomotives

effectively; and (iii) they do not consider the constraints related to different levels of desirability or preferable assignments. In addition, the current models often produce solutions with excessive consist-busting. This causes delays and introduces costs that are not reflected in the objective function of these models.

### 5.1. Mathematical Formulation

We focus on the planning version of the locomotive scheduling model (LSM), in which we are to assign a set of locomotives to each train in a weekly train schedule, so that each train receives sufficient pulling power, horsepower, and speed. Delays, fueling, and maintenance needs are not considered. We aim to determine the active and deadheaded locomotives for each train, light-traveling locomotives, and the train-to-train connections, while minimizing the total costs (ownership, active, deadheaded, and light traveling of locomotives, penalty for single-locomotive assignments, and consist-busting).

We are given a weekly train schedule in terms of train routes and times; the set of all trains is given by $L$. We assume that the same train running on different days is represented by different trains in the set $L$, one for each day it runs. For each train $l \in L$, we are given its origin and destination stations, as well as the respective departure and arrival times. Let $T_l$ and $\beta_l$ be the tonnage requirement and the horsepower per tonnage needed for train $l$, and $E_l$ be the penalty for using a single-locomotive consist.

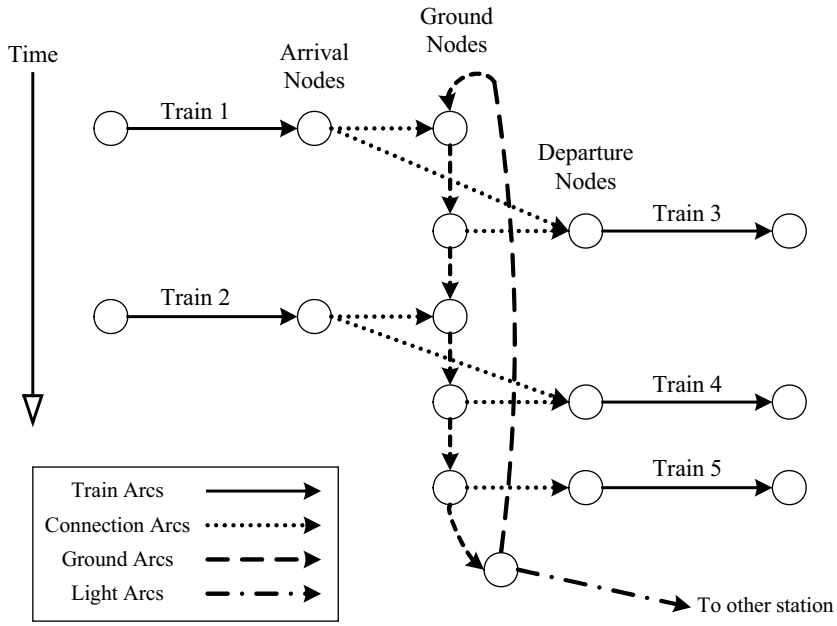
We are also given a set $K$ of all locomotive types that are available for pulling the trains. Let $h^k$ and $\lambda^k$ represent, respectively, the horsepower and the number of axles for each locomotive type $k \in K$. The weekly ownership cost of a locomotive of type $k$ is given by $G^k$, while $B^k$ represents its fleet size, i.e., the respective number of locomotives available for assignment. Each active locomotive of type $k$ assigned to train $l$ provides the tonnage pulling capability $t_l^k$.

We formulate the LSM as an integer multicommodity flow problem with side constraints. This mixed-integer programming (MIP) formulation represents the flow of locomotives on a *weekly space-time network*. In this network, arcs denote trains; nodes denote events (that is, arrivals and departures of trains and locomotives); and each different locomotive type defines a commodity.

We denote the weekly space-time network as $G = (N, A)$, where $N$ and $A$ represent the node and the arc sets, respectively. For each arrival event, we create an *arrival ground node*, and for each departure event we create a *departure ground* node. Each node $i \in N$ is associated with two attributes: place and time. The sets of nodes related to train departures and arrivals are denoted, respectively, by *DepNodes* and *ArrNodes*, and the ground nodes by *GrNodes*. Let $AllNodes = DepNodes \cup ArrNodes \cup GrNodes$.

This network contains four types of arcs, as shown in Figure 10. The set of *train arcs* (*TrArcs*) contains one arc for every scheduled train; its endpoints denote the location and time of the departure and arrival events. We connect each arrival node to the associated arrival-ground node by a directed arc called *arrival-ground connection arc*; similarly, we connect each departure-ground node to the associated departure node through a directed arc called *ground-departure connection arc*. These connection arcs belong to the set of connection arcs (*CoArcs*). We next sort all ground nodes at each station in chronological order by their time attributes and connect each ground node to the next ground node through directed arcs called *ground arcs*, denoted by the set *GrArcs* (we assume without any loss of generality that ground nodes at each station have distinct time attributes). These ground arcs allow inbound locomotives to stay in an inventory pool as they wait to be connected to the outbound trains. An inbound train that sends its entire consist to an outbound train is represented by a *train-to-train connection arc*, which directly links an arrival node to a departure node; these arcs also belong to the set *CoArcs*. Finally, we also allow the possibility of locomotives light travel. We create a *light arc* originating at a ground node (with a specific time and at a specific station) and also terminating at a ground node. Each light arc belongs to

FIGURE 10. Part of a weekly space-time network for a station.




the set *LiArcs* and has a fixed charge $F_l$ that denotes the fixed cost of sending a single locomotive with crew from the origin of the light arc to its destination. The light arc also has a variable cost that depends on the number of locomotives light traveling as a group. Let $AllArcs = TrArcs \cup CoArcs \cup GrArcs \cup LiArcs$, and $d_l^k$ be the cost of deadheading of locomotive type $k$ on a train arc $l \in TrArcs$; for every other arc, $d_l^k$ denotes the cost of traveling for a nonactive locomotive of type $k$ on arc. For each node $i$, the sets of incoming arcs into $i$ and outgoing arcs from $i$ are given by $I(i)$ and $O(i)$, respectively.

Our formulation has five sets of decision variables: (i) $x_l^k$ is the number of active locomotives of type $k$ allocated to train $l$; (ii) $y_l^k$ is the number of nonactive locomotives (deadheading, light-traveling, or idling) of type $k$ on all arcs; (iii) $z_l$ equals 1 if at least one locomotive flows on the arc $l \in LiArcs \cup CoArcs$, and 0 otherwise; (iv) $w_l$ equals 1 a single locomotive is assigned to train $l$, and 0 otherwise; and (v) $s^k$ indicates the number of unused locomotives of type $k$. The MIP formulation of the LSM is given by:

$$\text{Minimize} \quad \sum_{l \in TrArcs} \sum_{k \in K} c_l^k x_l^k + \sum_{l \in AllArcs} \sum_{k \in K} d_l^k y_l^k + \sum_{l \in LiArcs} F_l z_l + \sum_{l \in CB} V z_l + \sum_{l \in TrArcs} E_l w_l - \sum_{k \in K} G^k s^k \tag{29}$$

$$\text{subject to} \quad \sum_{k \in K} t_l^k x_l^k \geq T_l \qquad \forall\, l \in TrArcs \tag{30}$$

$$\sum_{k \in K} h_l^k x_l^k \geq \beta_l T_l \quad \forall\, l \in TrArcs \tag{31}$$

$$\sum_{k \in K} \lambda_l^k x_l^k \leq 24 \quad \forall\, l \in TrArcs \tag{32}$$

$$\sum_{k \in K} (x_l^k + y_l^k) \leq 12 \quad \forall\, l \in TrArcs \cup LiArcs \tag{33}$$

$$\sum_{l \in I(i)} (x_l^k + y_l^k) = \sum_{l \in O(i)} (x_l^k + y_l^k) \quad \forall\, i \in AllNodes,\ \forall\, k \in K \tag{34}$$

$$\sum_{k\in K} y_l^k \le 12 z_l \quad \forall\, l \in CoArcs \cup LiArcs \tag{35}$$

$$\sum_{l\in O(i)} z_l = 1 \quad \forall\, i \in ArrNodes \tag{36}$$

$$\sum_{l\in I(i)} z_l = 1 \quad \forall\, i \in DepNodes \tag{37}$$

$$\sum_{k\in K} (x_l^k + y_l^k) + w_l \ge 2 \quad \forall\, l \in TrArcs \tag{38}$$

$$\sum_{l\in S} (x_l^k + y_l^k) + s^k = B^k \quad \forall\, k \in K \tag{39}$$

$$x_l^k \in \mathrm{Z}^+ \quad \forall\, l \in TrArcs,\ \forall\, k \in K \tag{40}$$

$$y_l^k \in \mathrm{Z}^+ \quad \forall\, l \in TrArcs,\ \forall\, k \in K \tag{41}$$

$$z_l \in \{0,1\} \quad \forall\, l \in CoArcs \cup LiArcs \tag{42}$$

$$w_l \in \{0,1\} \quad \forall\, l \in TrArcs. \tag{43}$$

The objective function (29) contains six terms. The first term denotes the cost $c_l^k$ of actively pulling locomotives on train arcs. The second term captures the cost $d_l^k$ of deadheading locomotives on both train arcs and light travel arcs, as well as the cost of idling locomotives. The third term denotes the fixed cost $F_l$ of light-traveling locomotives. The fourth term denotes the fixed cost $V$ of consist-busting for the set $CB$ of all connection arcs linking arrival nodes to ground nodes. The fifth term denotes the penalty $E_l$ associated with single-locomotive consists, while the sixth term represents the savings $G^k$ accrued from not using all the available locomotives. Observe that we can obtain different levels of consist-busting by selecting different values of the consist-busting penalty cost $V$. The greater the value of $V$, the less the amount of consist-busting is in the optimal solution.

Constraints (30) and (31) ensure that the locomotives assigned to a train provide the required tonnage and horsepower, respectively. Constraint (32) enforces the maximum number of active axles assigned to a train, while constraint (33) ensures that each train arc and light arc is assigned at most 12 locomotives. Constraint (34) relates to the flow balance (number of incoming and outgoing locomotives of each type $k$) at every node. Constraint (35) captures the occurrence of a positive flow on a connection arc or a light arc; it also ensures that no more than 12 locomotives flow on any light arc. Constraints (36) and (37) ensure that either a train-to-train connection or a consist-busting occurs for each inbound and outbound train. The assignment of a single-locomotive consist to a train is represented by constraint (38). Finally, constraint (39) accounts for the total number of locomotives used in the week, given by the sum of the flow of locomotives on all the arcs crossing a time instant of the week when no event (arrival or departure of train) takes place, denoted by *CheckTime*; the difference between the number of locomotives available and the number of locomotives used equals the number of unused locomotives ($s^k$).

The locomotive scheduling problem is a very large combinatorial optimization problem and is also NP-complete (Ahuja et al. [3]). A large railroad has a train schedule that consists of several hundred trains with different weekly frequencies, which translates into several thousand train departures per week. Some trains run every day in a week, whereas others run less frequently. Many trains have long distances to travel and take several days to go from their origins to their destinations. To power these trains, several thousand locomotives of different types are available, with different costs, horsepower, and pulling capacities. To illustrate this, for a real-world problem, with 538 trains (each of which operated several days in a week) and 5 locomotive types, the size of the resulting formulation contains about 197,000 integer variables and 67,000 constraints. This problem is too large to be solved to optimality or near-optimality using existing commercial-level MIP software.

## 5.2. Solution Method

It should be noted that the above mathematical formulation (29)–(42) does not include other constraints found in practice. A railroad may want a solution that is *consistent* throughout the week in terms of the locomotive assignment and train-to-train connections. If a train runs five days a week, we want it to be assigned the same consist each day it runs. Consistency of the locomotive assignments and train-to-train connections are highly desirable from an operational point of view, because it translates into rule-based dispatching that makes the job of a locomotive dispatcher relatively easy (the dispatcher can remember the rule and enforce it). In addition, assignments of locomotive types to trains may have different degrees of preference.

In order to cope with these additional consistency constraints, which would make the formulation even more complex in terms of additional variables and constraints to be added, not to mention that the network size would be increased dramatically, Ahuja et al. [6] propose an efficient decomposition-based heuristic approach that allows near-optimal solutions for real instances of the LSM to be obtained in short to moderate computing times. This heuristic, which makes extensive use of the commercial software CPLEX 7.0, comprises two main stages:

(i) Modify the original problem so that all trains run seven days a week, and then solve the daily locomotive scheduling problem instead of the weekly scheduling problem.

(ii) Adjust the solution of the first stage (daily schedule) to solve the original weekly schedule problem in which trains do not all run seven days a week.

**5.2.1. Solving the Daily Locomotive Scheduling Problem.** This first stage of the solution approach is a simplification that reduces the weekly locomotive problem to a daily problem, thus reducing the size of the MIP substantially, and also helps satisfy the consistency constraints. A *daily space-time* network is created similarly to the weekly network described above and is about seven times smaller. In the simplified model it is assumed that (i) all trains that run $p$ days or more per week run *every* day of the week, and (ii) all trains that run less than $p$ days do not run at all. This assumption results in an approximation in the sense that it provides locomotives to trains that do not actually exist; similarly, it may not provide locomotives to trains that do exist. This simplification is based on the observation of real data, for which the vast majority of train arcs in the weekly space-time network correspond to the trains that run at least five days a week. Some minor modification is required in the mathematical formulation in order to account for the adjusted fleet size for the daily problem.

Though the daily space-time network is substantially smaller, it is still too large to be solved to optimality using commercial MIP software. To illustrate this, the daily space-time network equivalent to the weekly network described above contains 1,323 nodes and 30,034 arcs. Though the LP relaxation takes only a few seconds to solve, our experiments show that the MIP does not provide any feasible integer solution in 72 hours of running time. We conjecture that the biggest source of difficulty is the presence of the fixed-charge variables $z_l$ (for connection and light arcs). In this manner, to obtain high-quality feasible solutions and to maintain a relatively small total running time of the algorithm, the idea is to eliminate these fixed-charge variables using the following three-step sequential heuristic approach:

*Step* 1: *Determine Train-to-Train Connections.* In this step we consider that railroads usually specify some train-to-train connections, thus allowing us to eliminate inadmissible train-to-train connections. This can be done heuristically in the following way: (i) add all the candidate train-to-train connections; (ii) fix these train-to-train connections one by one and solve the LP relaxation of the daily MIP formulation to assess the impact of these connections; and (iii) keep those connections with small impact on the cost of assignment.

*Step* 2: *Determine Light Movements.* To allow light travel of locomotives from any station to any other station at any time of the day, it is necessary to add a large number of arcs

in the daily space-time network. Instead, we use heuristic procedures to create a small but potentially useful collection of light-travel arcs, and to select a subset of these arcs for light travel. To achieve this, we (i) add all the candidate light arcs, (ii) remove the candidate light arcs one by one and assess the impact of these removals using the LP relaxation of the MIP formulation, and (iii) remove the light movements with minimal impact on the cost of flow.

*Step* 3: *Determine the Assignment of Locomotives.* Once the fixed-charge variables are eliminated through the previous steps, we determine the remaining variables, which correspond to the arcs related to the movement of the active and deadheaded locomotives. To accomplish this, we solve the integer program for the daily locomotive assignment without the fixed-charge consist-busting and light-travel variables. Though CPLEX can provide a high-quality solution within 15 minutes of execution time, it does not terminate even when it is allowed to run for over 48 hours. On the other hand, this initial feasible solution can easily be improved by a modest modification. Thus, we use a neighborhood search algorithm to look for possible improvements with respect to this solution.

The above steps allow us to reach a near-optimal solution to the daily locomotive scheduling problem in short computing times.

**5.2.2. Solving the Weekly Locomotive Scheduling Problem.** When the daily scheduling solution is applied to the corresponding weekly scheduling problem (by repeating it every day of the week), some trains may not be assigned any consist (those that run less than $p$ days in a week) and some locomotives may be assigned to trains that do not exist (those that do not run every day). To transform this daily solution into a feasible solution for the original weekly problem, locomotives are taken from the trains that exist in the daily problem but do not exist in the weekly problem and assigned to the trains that do not exist in the daily problem but exist in the weekly problem. It should be noted that additional locomotives may eventually be required to meet these constraints.

A modified weekly space-time network is then constructed in the same manner as the daily space-time network, where each train running on different days is treated as a separate train. However, we make some changes for the train-to-train connection arcs. For a train-to-train connection arc in the daily space-time network, we have a corresponding train-to-train connection arc only on those days when both the associated trains run, and not otherwise.

Due to the size of the resulting modified MIP flow formulation that solves the weekly problem that is based on the solution of the daily problem, and thus requires excessive computing time, a heuristic approach is applied in which the locomotive schedule is determined for one locomotive type at a time. This can be accomplished by converting the multicommodity flow problem (with each locomotive type defining a commodity) into a sequence of single commodity flow problems with side constraints, one for each locomotive type. The aim is to determine the optimal flow of the current selected locomotive type while keeping the flow of other locomotive types intact, similar to the approach used for the daily scheduling problem. To do so, the different locomotive types are first arranged in decreasing order of their availability and then, following this order, considered one by one. Each of these subproblems is a (single-commodity) constrained minimum cost flow problem and can be solved very efficiently. After a feasible integer solution of the weekly locomotive scheduling problem is obtained, a neighborhood search algorithm is applied to improve it. The search is terminated when the solution value does not improve for any locomotive type. Finally, after the solution of the weekly locomotive scheduling problem is obtained, additional train-to-train connections are created by matching the locomotive assignments of inbound and outbound trains at each station if they have the same consist.

To summarize, the methodology for solving the planning version of the locomotive scheduling problem involves the following main steps: (i) modeling the weekly scheduling problem as a MIP multicommodity flow problem with side constraints; (ii) transforming this problem into a daily scheduling problem; (iii) heuristically solving a simplified version of the daily

Table 5. Comparison of heuristic algorithm and CSX solutions.

| Locomotive type | Proposed heuristic | CSX solution | Measure | Proposed heuristic (%) | CSX solution (%) |
|---|---|---|---|---|---|
| SD40 | 249 | 498 | Active time | 44.4 | 31.3 |
| SD50 | 160 | 171 | Deadheading time | 8.1 | 19.6 |
| C40 | 487 | 621 | Idling time | 46.7 | 49.1 |
| AC44 | 154 | 164 | Light-traveling time | 0.8 | 0 |
| AC60 | 160 | 160 | Consist-busting | 49.4 | 85.0 |
| Total | 1210 | 1614 | | | |

scheduling problem (in which some fixed-charge variables are previously set through greedy algorithms); and (iv) transforming the daily solution into a feasible and effective solution for the weekly problem.

### 5.3. Computational Results

In Table 5, we present the comparison of the results obtained by Ahuja et al. [6] with the ones obtained by the software developed in-house by CSX, a major U.S. railroad company. The algorithms were implemented in C++ and run and tested on a Pentium III PC with a speed of 750 MHz. The results indicate that whereas CSX software required 1,614 locomotives to satisfy the train schedule, the heuristic required only 1,210 locomotives, thereby achieving a savings of over 400 locomotives, or about 25% of the fleet. The locomotive utilization increased by 13.1%, while the deadhead and idling times decreased by 9.5% and 2.4%, respectively. The authors report similar improvements on other benchmark instances. We thus observe that this heuristic produces far superior solutions and demonstrates major savings potential.

### 5.4. Concluding Remarks

The results obtained so far by the heuristic algorithm to solve the planning version of the locomotive scheduling problem produces solutions that satisfy the constraints and business rules usually required by railroads, and offers considerable savings in cost, especially in terms of a significant reduction in the number of locomotives needed. Results are obtained in very short running times, usually less than one hour. The model is the first optimization approach to account for consist-busting, light travel, and consistency of the solution in a unified framework.

On the other hand, there is a fair amount of additional research that needs to be done before an optimization model can be used by railroads in their day-to-day locomotive planning and scheduling. It should consider the constraints that appear in the operational level, including the assignment of specific units of each locomotive type to each train, the fueling and maintenance needs of the locomotives, and train deviations from their scheduled times. Another area of potential interest for research is the assignment of locomotives for passenger trains, which may differ quite substantially from freight train problems in terms of its objective and constraints, though the ideas and algorithms presented here may be useful and applicable.

## 6. Train Dispatching Problem

The *train dispatching* problem, also known as the *train meet-and-pass* problem, aims to determine detailed train movements and timetables over a rail network under various operational constraints and restrictions in order to minimize train deviations from the planned schedule. This problem assumes that an ideal train schedule is given—i.e., for each scheduled train we know the planned departure and arrival times at the corresponding origin and

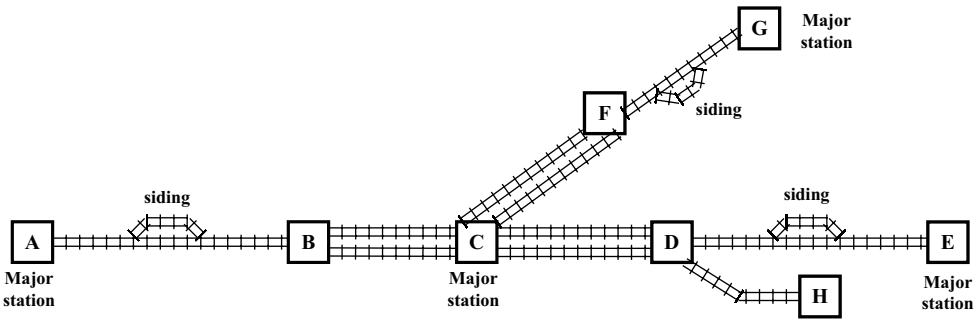
destination stations, as well as the travel times along the portions of its itinerary. This ideal train schedule results from the solution of the train scheduling problem that we discussed in §4. In the train scheduling problem we assign blocks to trains and identify a master schedule in terms of train paths and timings with respect to main stations and origin-destination points of the shipments. However, this master schedule does not consider the conflicts that occur as the trains meet along the tracks between these main stations.

This problem is particularly relevant for most freight railroads, ones that usually operate over extensive lengths of single-line tracks, in which a train has to be pulled over to cross or overtake another train, causing undesirable delays to freight and shipments, not to mention reductions in the productivity of locomotives and railcars. The related rail network may be composed of one or more track lines connecting major stations. Each track line may encompass several intermediate single- and double-line sections between intermediate stations (terminals or junction points). A single-line section is defined as a segment of track line that can accommodate only one train at a time. Similarly, a double-line section can accommodate up to two trains at a time, traveling either in the same direction or in opposite directions. A track section is delimited at each of its extreme endpoints by a siding or an intermediate station. A siding can accommodate at least one train waiting for crossing or overtaking, while an intermediate station serves as a yard or a terminal where passing trains can wait for crossing or overtaking and from which trains can depart or arrive. An example for a partial rail network is given in Figure 11, where stations A, C, E, and G can be considered as major stations. Track sections B-C, C-D, and C-F are double line. Sidings are located in sections A-B, D-E, and F-G.

The detailed dispatching plan consists of a timetable that comprises the departure and arrival times of each train in each track section of its route in such a way that all rail traffic constraints are met, especially track capacities, thus ensuring that all train crossings and overtakes happen at passing sidings or intermediate stations where there is enough room for the involved trains to be accommodated. Therefore, the resulting dispatching plan also encompasses the detailed information of all trains that are delayed due to conflicts with other trains, such as their locations and the start and end times of each nonservice stop where trains are pulled over solely for crossing or overtaking.

In addition to operational constraints that honor the rail traffic movement, some other practical constraints may arise. Insufficient track length or the nature of the freight being carried may restrict the trains that can be stopped at a location (e.g., a train of hazardous products may not be allowed to pull over near a populated area). Some trains may not be allowed to stop at some specific sidings because they may not have enough traction power to resume moving if they are pulled over, depending on the specific track slope ahead of the siding towards the direction the train is to move. Similarly, the lack of adequate infrastructure for crews may limit the maximum time a train can wait at a siding. Maximum allowed delays for individual trains and train types play a critical role in attaining consistent

FIGURE 11. Part of a rail network with single and double-line tracks.



schedules and dispatching plans. Other constraints found in practice may include meeting exact timetables for some trains, minimum and maximum departure and arrival headways for trains of the same type following the same route (to avoid unnecessary wait time by arriving too early to load or unload), and ensuring that these trains arrive on time and in the same sequence as they departed.

The train dispatching problem may arise in different contexts. In real-time scheduling, it aims to help the dispatchers who are controlling the traffic to make sound decisions about which trains to stop and where, as updated data about train positions become available. In tactical planning, it consists of finding the best master timetable schedule on a regular periodic basis (e.g., weekly, monthly, or quarterly). In strategic planning, it relates to investment decisions in railroad infrastructure like new sidings to be built, extension of current sidings to accommodate longer trains, and duplication of single-line segments. This problem has been receiving increasing attention lately as advanced control systems that provide real-time information on train position and velocity are being developed and implemented by several railroad companies.

This problem is a well-studied railroad optimization problem in the operations research literature. Several earlier research papers study analytical line models whose main aim is to estimate the delay to each train caused by interferences on a rail line as a function of dispatching policies, traffic distribution, and physical track topology. A good summary review of these papers can be found in Cordeau et al. [19]. Starting in the mid 80s, computerized train dispatching tools have been proposed. The papers Rivier and Tzieropoulos [42], Rivier and Tzieropoulos [43] and Sauder and Westerman [46] contain these studies. Particularly for European railroads that mostly run passenger trains, there are several studies that consider dispatching and track allocation problems (see Adenso-Diaz et al. [1], Carey and Lockwood [14], Kroon and Peeters [35]). Integer programming formulations are proposed in Brännlund et al. [10]; Caprara et al. [12]; Higgins et al. [28, 29]; and Kraay and Harker [34], as well as solution methods to solve these problems approximately. As the size of the problem increases, thus making it difficult to solve to optimality, not to mention the necessity of a tool to reach a solution quickly (specifically for real-time dispatching decisions), a number of heuristic algorithms have also been proposed (see Cai et al. [11] and Şahin [44]).

However, the algorithms developed so far rely on simplifications of the general problem to solve specific instances. Several papers deal with problems related to passenger trains, in which the objective function and the constraints may differ quite significantly from freight trains. The intractability of the formulations for freight train dispatching, some modeled even as nonlinear optimization problems, have led to simplifications to allow finding a feasible solution that sometimes is not even known to be applicable in practice. In general, the problems consider a railroad line linking two major points. None of them consider explicitly a rail network composed of different track lines and the interactions between different trains traveling from different origins to different destinations that share distinct parts of their routes. Decomposition approaches for such networks may lead to good, near-optimal partial solutions for each track line, but often result in poor global solutions due to the intrinsically greedy nature of this type of approach, which does not consider the impact of a train crossing the boundaries of a partial network on the traffic of the other trains in the entering portion of the network. Specific operational constraints such as those related to train size and type as mentioned before are considered neither in the formulations nor in the solutions proposed.

### 6.1. Modeling Approach

The paper in Şahin et al. [45] proposes a novel approach for modeling the train dispatching problem as a multicommodity flow problem with side constraints on a space-time network. Each train $t$ (belonging to a set $T$ of trains to be dispatched) functions as a commodity in the network. The space-time network representation allows the formulation to be easily adapted for intersecting complex rail networks instead of only a single major line, as well

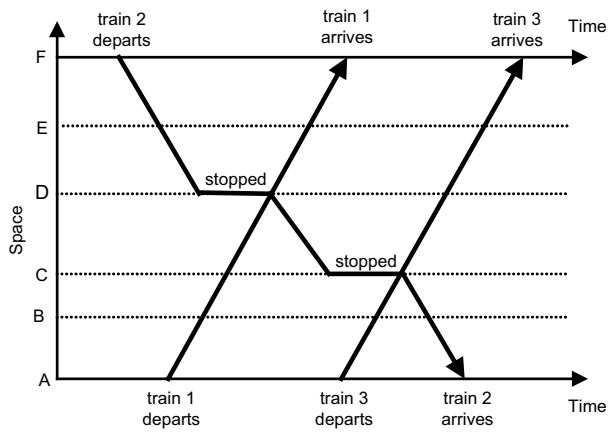
as for multitrack lines, as shown before in Figure 11. Most of the practical constraints that would make the problem size even larger with the previous formulation approaches found in the literature are handled within the network construction phase. The proposed formulation is flexible enough to incorporate most of the practical constraints without having to change its underlying structure.

**6.1.1. Network Representation.** For the sake of clarity on how the space-time network is built, we will consider a track line that links two major stations and is composed of single- or double-track sections; the generalization to a more complex structure involving multiple interconnecting lines is straightforward and requires no major change. The proposed network representation is based on the concept of a train graph diagram, which is a standard method for displaying the resulting train schedules, meet-points, and associated delays. Many freight railroads still rely on train schedules that are made manually by skilled dispatchers using this kind of diagram. Figure 12 illustrates a sample train graph schedule diagram for a single-line track that links terminal stations A and F. Intermediate meet-points are located at stations B to E. Vertical and horizontal axes represent space and time, respectively. There are two northbound trains that move from A to F, and one southbound train. The southbound train is delayed twice, at meet-points D and C, as a possible solution for the conflicts with the two northbound trains. In other words, these delays allow northbound trains to meet and pass.

Even in this very simple example, there are many possible combinations of sidings and times for trains to be pulled over to allow meets and passes, not to mention eventual changes in the departure times from the intermediate stations. Therefore, the train dispatching problem is a very large-scale combinatorial optimization problem. When several trains in both directions are scheduled, many conflicts may arise. Each conflict involves trains moving either in opposite directions or in the same direction. Depending on the chosen solution for a conflict involving two trains (i.e., which train pulls over for the other to pass), the location and the time of later conflicts may change, new conflicts at different locations and times may arise, and existing conflicts may cease to exist. Thus, the number of possible solutions to train conflicts can be very large.

On the line track linking the two major stations, let $S = \{0, \ldots, s\}$ represent the set of stations and sidings, numbered according to the order in which they appear along the track line (say from north to south). Each train $t \in T$ may depart from any station $s_1 \in S$ and arrive at any other station $s_2 \in S$ $(s_1 \neq s_2)$. We also define the set of sections (line segments) to represent the tracks between consecutive stations or sidings $s_i$ and $s_{i+1}$ as $L = \{0, \ldots, s-1\}$. $IS(t) \subset S$ denotes the set of intermediate stations or sidings $s \in S$ where train $t \in T$ is allowed to pull over to wait for crossing or overtaking while $P(t) \subset L$ denotes the ordered

FIGURE 12. A train graph schedule diagram.

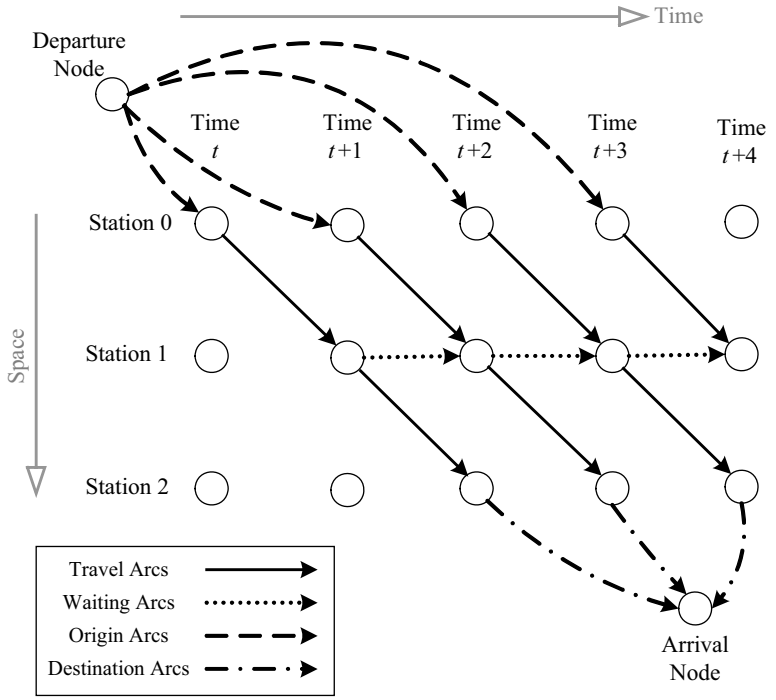


set of consecutive sections $(d^t - s_1, s_1 - s_2, \ldots, s_k - a^t)$ that compose the train path. For each train $t \in T$ we are given the origin $(d^t)$ and destination $(a^t)$ stations; the corresponding scheduled departure time $ed^t$ is known, as well as the travel time along track segment $i(f_i^t)$. The scheduled arrival time at destination $(sa^t)$ can be calculated by $sa^t = ed^t + \sum_{l \in P(t)} f_l^t$. For each train $t$, we also specify the maximum allowable delay $md^t$, which is based on business rules or operating principles. Accordingly, latest possible departure time for train $t$ $(ld^t)$ and latest possible arrival time for train $t(la^t)$ can be calculated as $ld^t = ed^t + f_i^t$ and $la^t = sa^t + f_i^t$.

We represent the space-time network as $G = (N, A)$, where $N$ and $A$ denote node and arc sets. The space-time network contains three types of nodes. *DepNodes* represents the set of artificial nodes in which the outflow generates departure of a train from its origin. There exists a departure node for each train $t \in T$ denoted as $Dep^t$. Therefore, the supply of each node in this set is one. The *ArrNodes* set corresponds to the set of artificial nodes in which the inflow represents the arrival of a train to its final destination. In a similar way, its demand is set to one and denoted as $Arr^t$, for all $t \in T$. *StatTimeNodes* is the set of nodes with two attributes: station/siding and time. Time is discretized with respect to equal-length time periods. For instance, if a daily schedule (24 hours) is discretized in 10-minutes periods, then $q = 144$ where the time period set is $Q = \{1, 2, \ldots, q\}$. Nodes of the set *StatTimeNodes* correspond to the copies of each station node $s \in S$ in each time period $k \in Q$. An element of this set is denoted as $i_k$, where $i \in S$ and $k \in Q$.

The arc set $A$ is composed of four subsets as shown in Figure 13. The arcs in the set of *origin arcs* emanate from nodes in *DepNodes*, and *destination arcs* enter the nodes in *ArrNodes*. To represent the departure of a train $t$ from its origin station $d^t$, we create arcs from node $Dep^t$ to nodes $d_k^t$ for each time period $k$ where $ed^t \le k \le ld^t$. Similarly, to model the arrival of a train $t$ at its destination station $a^t$, we create arcs from nodes $a_k^t$ to node $Arr^t$ for $sa^t \le k \le q'$, where $q^t = \min\{q, la^t\}$.

FIGURE 13. Part of a space-time network for a train.



The arcs of the space-time network are created on a train-by-train basis following the sequence of stations/sidings $s \in IS(t)$, and segments $l \in P(t)$ on the train's route, starting from its origin station $d^t$ through its destination station $a^t$. The *travel arcs* for a train are constructed between each pair of consecutive stations on its path for only the time periods that lie within the earliest and latest travel times for the section. The set of *waiting arcs* for train $t$ includes the arcs from node $i_k$ to node $i_{k+1}$, corresponding to the siding $s \in IS(t)$ where the train is allowed to wait. Whenever $i_k$ has an incoming arc that represents a possible movement of train $t$, we create consecutive waiting arcs on the same station level by considering the maximum possible delay of train $t$ at that station.

Practical constraints can be considered and thus incorporated into the model by adding new arcs or getting rid of certain arcs on the network. Therefore, for instance, if a train cannot wait at a particular siding, then the corresponding waiting arcs are not created. Another restriction might require that specific trains (typically passenger trains) be stopped at certain intermediate stations during certain time windows. We handle this by adding the waiting arcs and avoiding the associated travel arcs on the network. Other practical constraints are handled similarly during the network construction phase instead of explicitly expressing those constraints in the formulation.

All arc capacities are set to 1. The costs of waiting arcs are set to the length (in minutes) in which time is discretized, because each arc represents a train waiting at a particular siding from time $k$ to time $k+1$. Similarly, costs for origin arcs correspond to the respective lengths of the initial delays. All other arcs' costs equal 0. Delays can also be multiplied by specific train weights to reflect relative priority of certain trains.

**6.1.2. Integer Programming Formulation.** Our IP formulation has binary flow decision variables $x^t_{i_k j_l}$ representing the flow of train $t$ on arc $(i_k, j_l)$ where $t \in T$ and $(i_k, j_l) \in A$. Note that for a given train $t$, each binary flow variable is defined only for feasible moves corresponding to one of the four arc sets, as described above.

$$\text{Minimize} \quad \sum_{t \in T} \sum_{(i,j) \in A} c_{ij} x^t_{ij} \tag{44}$$

$$\text{subject to} \quad \sum_{ed^t \le k \le ld^t} x^t_{Dep^t d^t_k} = 1 \quad \forall t \in T \tag{45}$$

$$\sum_{sa^t \le k \le la^t} x^t_{a^t_k Arr^t} = 1 \quad \forall t \in T \tag{46}$$

$$\sum_{i \in StatTimeNodes} x^t_{ij} - \sum_{i \in StatTimeNodes} x^t_{ji} = 0 \quad \forall t \in T,\ \forall j \in StatTimeNodes \tag{47}$$

$$\sum_{t} x^t_{ij} \le 1 \quad \forall (i,j) \in A \tag{48}$$

$$\sum_{t \in T^{out}} \sum_{l \le k-1} \sum_{m \ge k} x^t_{i_l (i+1)_m} + \sum_{t \in T^{in}} \sum_{l \le k-1} \sum_{m \ge k} x^t_{(i+1)_l i_m} \le 1 \quad \forall i \in S,\ \forall k \in Q \tag{49}$$

$$x^t_{ij} \in \{0,1\} \quad \forall (i,j) \in A,\ \forall t \in T. \tag{50}$$

The objective function (44) minimizes the total delay of all trains. Constraint (45) ensures that for each train $t$ there is a unit outflow from its departure node to any node in *StatTimeNode* that corresponds to the origin station at a time within its departure time window. Similarly, constraint (46) handles the possible arrival of each train at its destination within its time window. Constraints (47) are flow conservation constraints for the nodes in *StatTimeNodes*, while constraint (48) restricts arc capacities. Constraint set (49) is the track capacity constraints. These constraints ensure that there is not more than one train traveling on a track segment at any time. This is accomplished by ensuring that at most one of

the arcs passing through the same time instant at a particular track segment carries a unit flow (Şahin et al. [45]). Note that in the generalized formulation, the flow variable on an arc is the sum of multicommodity flow variables on that arc, i.e., $x_{ij} = \sum_t x_{ij}^t$. The first term in constraint (49) represents the sum of flows on the arcs in the outbound direction (from station $i$ to station $i+1$) during time period $k$. The second term represents the flows on the arcs in the inbound direction (from station $i+1$ to station $i$) during time period $k$. In total, the number of such constraints is equal to the number of discrete time periods (i.e., $(|S|-1)*|Q|$).

We now briefly discuss the size of the problem for a typical railroad application. The number of stations and sidings $|S|$ in the network can reach 100. Proper time representation to accurately represent traffic may require time to be discretized in periods of 10 minutes or less. For a planning horizon of two days (48 hours), the resulting network has about 28,800 nodes. Fifty trains traveling along the network imply some 720,000 or more binary decision variables. Thus, even a moderate-size problem turns out to be very large to be solved to optimality using existing commercial-level IP software.

## 6.2. Solution Approach

Motivated by developing an algorithm to help dispatchers make decisions in real-time scheduling, which requires that effective solutions be obtained quickly (usually in no more than 5–10 minutes), Şahin et al. [45] proposed an IP-based heuristic to solve the train dispatching problem. This heuristic is based on observation of the performance of CPLEX version 8.1 to solve different instances of the train dispatching problem. It is observed that for moderately small instances, which are measured in terms of the number of sidings, the length of the time horizon, the number of trains, and the number of potential conflicts, CPLEX performs well, reaching the optimal solution in a few minutes. An important aspect of the formulation is that larger maximum delay allowances lead to larger problems due to the increase of the number of arcs in the network, representing potential alternative solutions to the conflicts. Thus, decreasing maximum delay allowances tightens the formulation without affecting the feasibility. As the size of the problem increases and/or as longer maximum train delays are allowed, CPLEX fails to quickly reach the optimal solution, though a feasible solution can be found in a reasonably short amount of time, usually in less than a minute.

It should be noted that real-time train traffic control is almost overwhelmingly dynamic; unexpected and unpredictable events occur frequently, bringing perturbations that cause trains to be delayed all the time. This need for real-time optimization suggests that a major effort to determine a global optimal solution for all trains over the whole scheduling horizon (typically ranging from 8–12 hours to up to 1–2 days) may not be worth the effort. These unpredictable events may make it quite unlikely for some trains to strictly follow their original schedules, particularly in the long term. On the other hand, not taking into consideration a sufficiently long scheduling horizon may lead to optimal or near-optimal decisions to resolve conflicts being made based solely in the short-term range. Additionally, these decisions made with only the short term in mind may even turn infeasible in the long term, not to mention that in real-time dispatching it may be undesirable and inconvenient to change the entire train schedule by reoptimizing the whole problem every time a train is delayed. All these issues led to the development of a heuristic that is based on the IP formulation, which relies on available, off-the-shelf optimization software to find effective solutions.

The basic idea in this method consists of gradually reducing the maximum total allowable delays ($md^t$) for all trains based on new feasible integer solutions that are progressively obtained by CPLEX. Initially, we divide the scheduling horizon $q$ into two periods, the short and the long terms. Let $q_s$ be the length of the short term ($q_s \leq q$); consequently $q_l = (q - q_s)$ defines the remaining length of the scheduling horizon (i.e., the long term). We define $\delta_s$

FIGURE 14. The IP-based heuristic for the train dispatching problem.

**algorithm** *IP-Based Heuristic for TDP*;
**begin**
find the first initial feasible solution for the IP formulation using CPLEX;
**while** the current solution is not locally optimal **or** CPU time limit is not reached **do**
**for** each train $t$
set $md^t = \min(d^t + \delta_1, md^t)$ for the short term (periods 1- $q_s$);
set $md^t = \min(d^t + \delta_2, md^t)$ for the long term (periods $q_s$+1, $q$);
**end**; {for each train $t$}
process CPLEX for $\sigma$ seconds and find the current best feasible solution;
**end;**
**end;**

and $\delta_l$ as increments to the train delays obtained in a feasible solution for the short and long terms, respectively. Suppose we find a feasible solution in which a train $t$ is delayed $d^t$ units of time ($d^t \leq md^t$). The idea is to make the mathematical formulation progressively tighter, to make it converge faster to a good solution. In this case, the maximum allowed delay for train $t$ is reduced from $md^t$ to the minimum of $(d^t + \delta_s)$ and $md^t$ for the time periods within the short term. Similarly, it is reduced from $md^t$ to the minimum of $(d^t + \delta_l)$ and $md^t$ in the long term where $\delta_l \leq \delta_s$. Note that the total delay along the whole scheduling horizon is still less than the maximum delay allowed for the short term. Therefore, a feasible solution is always guaranteed, and the problem is tighter. The optimization process is then restarted for this tighter problem and proceeds for a fixed time period $\sigma$. With this method, the chances that a better feasible solution is found in a shorter amount of time are increased. The same process is repeated until an optimal solution for the progressively tightened problem is found or a maximum total CPU time is reached. Figure 14 formalizes the IP-based heuristic for the train dispatching problem.

## 6.3. Computational Results

The paper Şahin et al. [45] reports extensive computational results for a set of generated test problems. These tests also consider two other heuristics proposed by the authors: a simulation-based construction heuristic and a greedy enumeration heuristic. The results show that the solution quality of the IP-based heuristic is very satisfactory and the solution times are competitive even when compared to the other simpler heuristics. In this section, we only discuss the results of real-life problem of a major Brazilian railroad transporting heavy-haul freight (iron ore unit trains with low-interval, frequent trains). The data is obtained from Leal et al. [36]. The problem comprises a single-line track that links two major stations with 23 intermediate stations or sidings where trains can be pulled over. 25 scheduled trains are to be dispatched over a 24-hour period. The trains run in both directions. The train routes, the travel times in each section, and the scheduled departure times are all known. Our objective is to minimize the total unweighted delay of all the trains. We set the maximum allowed delay equal to four hours for all the trains traveling between the two endpoints; for the remaining trains traveling between a pair of intermediate stations, we set the maximum allowed delay as proportional to this value based on the total travel time. Times are discretized in periods of only five minutes to produce a precise timetable for the trains.

Table 6 presents the results obtained for this real-world problem. The results show that the IP-based heuristic achieved the best result (0.06% above the optimal, corresponding to only five minutes of additional delay), significantly improving those obtained manually by the dispatchers, as well as the previous results of Leal et al. [36].

Table 6. Results for a real-world problem.

| | Total travel time (minutes) | Total delay (minutes) | Deviation from optimal (%) | CPU time (seconds) |
|---|---|---|---|---|
| Dispatchers solution | 14,119 | 6,311 | 71.35 | — |
| Optimal IP solution | 8,240 | 432 | — | 820 |
| IP-based heuristic | 8,245 | 437 | 0.06 | 173 |
| Heuristic #1* | 8,766 | 958 | 6.38 | Not available |
| Heuristic #2* | 8,755 | 947 | 6.25 | Not available |

*Obtained from Leal et al. [36].

### 6.4. Concluding Remarks

The novelty of the proposed IP formulation is based on the fact that it can easily handle several practical constraints without making the formulation more complex. The structure of the formulation does not change by adding these new constraints, because the variables are based on a space-time network representation rather than a conventional transformation of the variables. The complex railroad network infrastructures, such as double-lines and intersecting railroads, are easily handled with network representation, which avoids the complications in the formulation.

These approaches should be evaluated from two different perspectives: a master planning tool and a real-time decision-making tool. Our experiments indicate that with enough technology and advanced computational infrastructure that can support the most recent solver technology, the IP formulation provides satisfactory results as a master planning tool. The IP-based heuristic should also be considered a trustworthy master planning tool because the experiments confirm that it provides high-quality solutions with significantly less computational effort.

A real-time decision tool is a critical requirement for this problem because of unexpected delays and dynamically changing parameters that require train schedules to be constantly updated. Even the perfect master planning tools have no value unless they can support real-time decisions. The flexibility to change the parameters and the dynamic feature of the proposed approach make it a quick aid tool that the dispatcher can use to evaluate the effectiveness of real-time decisions.

## 7. Crew Scheduling Problem

Crew scheduling is a well known and widely studied problem in operations research, especially related to airlines and public transit systems. Railway crew scheduling problems have recently started to appear in operations research literature. The nature of these problems is highly dependent on the environmental factors (such as country, passenger/freight railroad, labor laws, and specific union agreements), and the majority of the studies conducted naturally focus on the specific characteristics related to the particular problem being studied, thus making it more difficult to apply it to a broad class of problems. In this section, we discuss the railroad crew scheduling problem according to the North American system, which is mainly composed of freight carrier operations.

Railroads employ thousands of units of manpower for the efficient transportation of the shipments (e.g., loading, unloading, yards, trains, traffic control, etc.). Hence, for a railroad, the manpower cost is a significant component of the overall transportation cost. A significant part of these employees, usually highly skilled, are involved in operating the trains on their planned timetable. Generally an engineer (who operates the locomotives) and a conductor (who coordinates the movement) are deployed in a train in its trip from its origin to its destination; this team is called a *crew/turn*. The *crew scheduling* (also called *crew planning* or *crew balancing*) problem is to schedule the crews in such a way that trains can be operated

at their schedule at minimum crew cost. In rail industry, the crew scheduling problem arises at two different levels:

(i) strategic level, in which decisions are made with respect to the number of crews required to operate the complete schedule of a long planning horizon, and

(ii) tactical level, in which for a given set of crews, assignments to trains are decided on a shorter time horizon (generally weekly).

In this section, we only focus on the tactical-level problem. Thus, the crew availability is given, and the aim is to determine crew assignment to operate trains in the schedule of a fairly short period of time (a week or shorter).

The two most common approaches in formulating and solving crew scheduling problems are based on column generation and set covering. A review of network models for crew scheduling in mass transportation systems can be found in Carraresi and Gallo [15]. In two recent studies on railroad crew scheduling problems, Chu and Chan [17] propose a network flow based algorithm for the Hong Kong Light Rail System and Caprara et al. [13] study a real-life case of the Italian railroad.

## 7.1. Characteristics of the Railroad Crew System

Similar to other applications of crew scheduling, railway crew scheduling is a complex problem. Not only the issues related with the rail traffic restrictions and/or requirements, but also contractual agreements and abiding laws should be taken into account in a feasible schedule. One particular characteristic of the North American railroads crew management system is the division of the rail network into *crew districts* that constitute a subset of terminals (nodes) and rail links (arcs). The crews are also grouped into *crew pools* or *crew types*. A crew type is usually characterized by a home terminal and a particular set of trains to operate. Every crew type has a terminal called the home terminal. The railroad does not incur any lodging cost when a crew member is at this terminal. All terminals other than the crew's home terminal are called *away terminals*. The rest of a crew member at any of these away terminals requires the railroad to take care of the necessary lodging arrangements, generally a hotel. In addition, further crew-type classification might be also based on a set of contractual rules or labor laws. For instance, two different crew types with the same home terminal who can operate the trains with same origin and destination are differentiated by their duty time period. While one crew type can be scheduled to operate the trains departing only before 6 pm, the other can be scheduled to work irrespective of the time.

*On-duty time* corresponds to the time at which a crew reports to be on duty. It does not necessarily correspond to the departure time of the train, because there may be some tasks to be performed by the crew members before the train departs. Similar to the tasks performed before the departure of the train, there are some tasks that are required to be performed upon the arrival of the train at the destination. Therefore, a crew will have to work after the arrival of the train. The time by which a crew finishes these tasks is called *tie-up time*. Accordingly, *duty period* of the crew starts at on-duty time and ends at tie-up time. The length of this period is also referred to as the journey length of the train.

*Crew deadheading* and *crew detention* are two special cases when the railroad is to pay the hourly rate to a crew member even while the crew is actually idle (i.e., not assigned to duty). Normally, a crew gets on to a train at the home terminal and takes it to the away terminal. After taking adequate rest at the away terminal, it is assigned to another train that is scheduled to another away terminal or back to the home terminal. However, either to meet the shortage of crew at a terminal or to improve the cost efficiency, a crew may be sent from one terminal to another by some other mode of transportation (e.g., via a taxi). All such idle travels of crews are called crew deadheadings. In each deadheading, the railroad incurs the cost of crew time traveling idle as well as the extra expenditure related to the movement of the crew (e.g., renting a taxi). Whenever a crew is released from duty at an away terminal, they stay in a hotel for an adequate rest period before being assigned to the

next train. After this minimum required rest period, there is a prespecified period in which crew can be called to duty at any time. However, if a crew will have to stay at an away terminal after this period, they are entitled to salary and, hence, the railroad has to incur extra cost for the rest of the crew. The period during which crew is paid salary while taking rest or idling at the away terminal is called *crew detention*. For example, let us assume that in a crew district, the minimum rest period at an away terminal is eight hours and the call period is eight hours. If a crew is not working for 18 hours at an away terminal before being assigned to a train, then the crew detention is 2 hours.

**7.1.1. Contractual Restrictions.** The problem of crew scheduling is apparently involved with workers. Hence, the work environment is bounded by the regulations imposed by the contractual agreements, labor regulations, and union agreements. Though such restrictions may vary slightly from one district to another or from one crew type to another, the regulations impose similar constraints to the work environment. Below, we list a set of restrictions that are commonly applicable to all crew districts and types:

(i) Duty period of a crew cannot exceed 12 hours. For an assignment to a train, the duty period includes the time elapsed between on-duty time and tie-up time, as mentioned before.

(ii) Whenever a crew member is released from duty at the home terminal or has been deadheaded to the home terminal, he/she can resume another duty only after 12 hours (10 hours rest followed by 2 hours call period) if the duty period is greater than 10 hours, and 10 hours (8 hours rest followed by 2 hours call period) if the duty period is less than or equal to 10 hours.

(iii) Irrespective of the duration of the stay of a crew at the home terminal, they are not paid any salary for the rest period.

(iv) Whenever a crew member is released from duty at the away terminal, he/she must go for a minimum of 8 hours of rest unless either of the following exceptions occurs. The first exception arises if the total time period corresponding to the recent travel time from the home terminal followed by a rest time less than 4 hours, plus travel time of a new assignment back home, is shorter than 12 hours. The second exception arises if the total time corresponding to the recent travel time from the home terminal plus travel time of a new assignment back home is less than 12 hours, and the rest time in-between the assignments is more than 4 hours. In both cases, there should be no other crew at the away terminal who is completely rested.

(iv) The deployment of crew must follow the first-in-first-out (FIFO) rule at both the home terminal and the away terminals. This rule implies that a crew with the maximum rest period among all the crews that are taking rest at the on-duty time of the next train should be assigned to that train.

The European railroad system runs strictly on schedule, mostly due to the fact that the schedule is composed of a mixture of freight and passenger trains, and the travel times are significantly shorter than those of the North American railroads. However, it is likely and probable to experience delays in the North American system for several reasons. Although this is highly unwanted, the system still should be capable of handling such situations by incurring appropriate costs. This is also an important factor from a quality-of-service perspective. One reason for departure delays is the unavailability of crew to run the train on schedule, which we refer to here as *crew delay*.

## 7.2. Mathematical Formulation

The railroad crew scheduling problem is to assign available crews to the trains over the planning horizon (generally weekly), such that the total costs of crew, deadheading, detention and crew delay are minimum while honoring the train schedule, the rail traffic constraints,

and contractual requirements. We first briefly describe the input, objective function, constraints, and decision variables for the railroad crew scheduling problem. We then develop a network representation of the problem and formulate a network flow model for this network. For simplicity, we formulate the problem for a single crew type of one district only. The home terminal of this crew type is either the origin or the destination of the trains considered in the schedule. All other terminals in the schedule are considered away terminals for this crew type. The network representation of the problem and the formulation can easily be extended to more crew types. Now, we briefly discuss the input requirements, decision variables, objective function, and constraints of the problem.

Input requirements for crew scheduling can be listed as follows:

*Train schedule*: This includes information about each train: its departure time at its origin and its arrival time at its destination. We also require the on-duty time and tie-up time for each train, and also the information.

*Crew availability*: This includes the name and type of each crew, its initial position (the terminal and time at which it finished its previous duty) at the beginning of the planning period, and the hours of duty done before.

*Cost parameters*: This includes per-hour salary of crew, per-hour taxi cost (deadhead), per-hour detention charge, and per-hour crew delay cost.

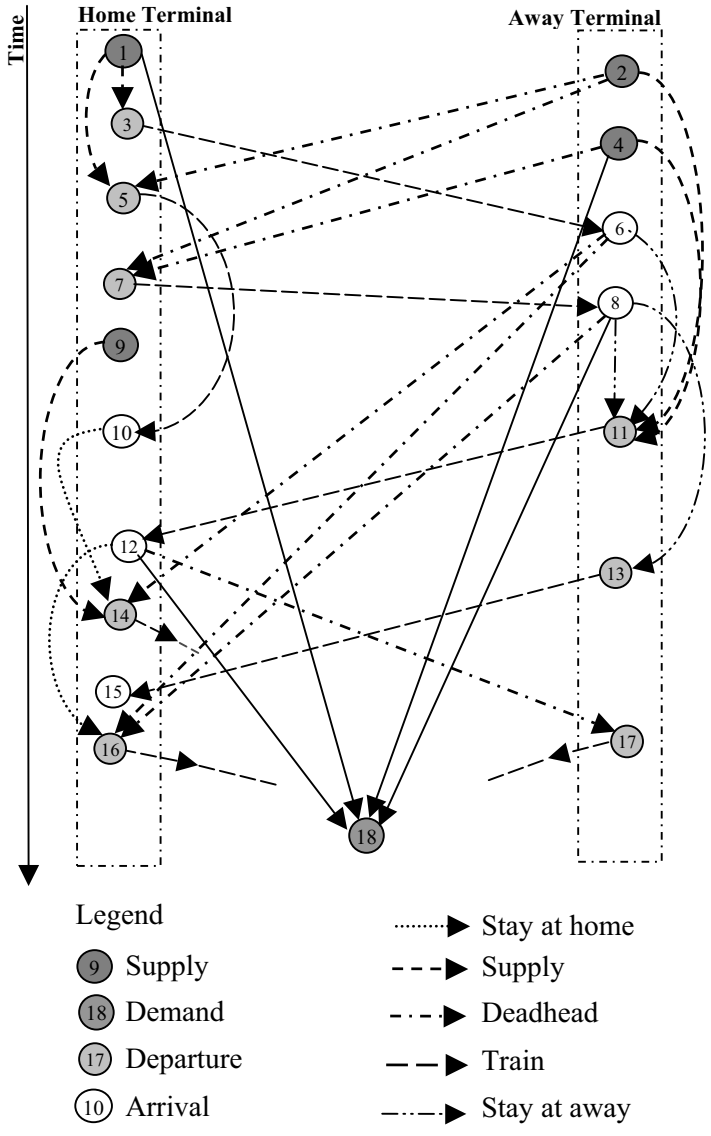
In the crew scheduling problem, we make decisions regarding the assignment of crews to trains, deadheading of crews, and crew delay of trains. The constraints can be considered in two groups: operational constraints and contractual requirements. The operational constraints ensure that every train gets a qualified crew to operate it, while a crew is not assigned to more than one train at the same time under the setting imposed by the crew districts and types. Therefore, only crews of certain crew types can be assigned to a certain train. Assignment of crews to trains must, in addition, satisfy the contractual requirements. The operational constraints of the model are handled by the network flow formulation, while the contractual restrictions are honored in the network construction phase. The objective of the crew scheduling problem is to minimize the total cost of salary to the crew, the cost of deadheading, the cost of crew detentions, and the cost of crew delay.

In this section, we discuss the network flow-based approach for the crew scheduling problem in Vaidyanathan et al. [48]. The problem is first represented by a space-time network in which (i) the nodes represent train origins and destinations and crew availability, (ii) the arcs are built based on possible and feasible assignments of crew to trains, and (iii) the cost parameters of the objective function are reflected in the network by the arc costs. Vaidyanathan et al. [48] then formulate the problem as a minimum-cost network flow problem in which each crew is represented by a commodity that flows.

**7.2.1. Constructing the Space-Time Network.** In the space-time network, each node has a time attribute attached to it in addition to its location attribute. We demonstrate the network representation with an example in Figure 15. To start with, the complete planning horizon is discretized in appropriately small time units. Then, the arrival and departure times of trains are mapped onto this discrete-time planning horizon.

Supply nodes represent the crew availability. Each crew member is denoted by a supply node whose time attribute is the time at which the crew is available, and whose location attribute corresponds to the terminal from where the crew can be released for duty. Because each supply node represents a crew member, the supply of the node is set to 1. We need only one demand node with a time attribute equal to the end of the planning horizon, and no specific location attribute. The demand of this node is set equal to the number of available crew members (i.e., the number of supply nodes). Departure nodes and arrival nodes correspond to the departures and arrivals of trains, respectively. The time attributes are attached by considering on-duty time (tie up time) for departure (arrival) nodes while the location attributes simply represent the origin (destination) terminals.

FIGURE 15. Crew scheduling network for a single district and a single crew type.




In Figure 15, arcs from supply nodes to departure nodes represent the possible first duty assignment of a crew to a train. Train arcs are constructed for each train from its departure node to its arrival node, and the arc costs are calculated based on the hourly salary and the travel time of the train. Deadhead arcs can be either from a supply node or arrival node to a train departure node representing the deadhead movement of a crew member with the associated deadheading cost. Stay arcs at the home (away) terminal are constructed from an arrival node to a departure node at the same terminal. While constructing the stay arcs, all possibilities of a crew member being assigned from an arriving train to a departing train should be considered, and the arc costs are calculated based on the length of the stay. Both deadhead arcs and stay arcs should comply with the contractual restrictions, hence, the network construction honors the associated constraints by creating only allowable arcs. Finally, the demand arcs are constructed from any of the arrival nodes and supply nodes to the single demand node by taking into account all possible last assignments of the crew members in the planning horizon.

FIFO policy cannot be easily handled while constructing the network. Figure 15 shows an example network where all rules except FIFO policy are considered as explained above. However, we will able to resolve this issue by adding a side constraint to the minimum-cost network flow formulation.

**7.2.2. Integer Programming Formulation.** The constructed network is used to formulate a minimum-cost network flow problem in which each crew corresponds to a commodity of the network. The formulation honors the operational constraints of the problem with flow balance and arc capacity constraints. In the following formulation, $N$ is the node set where $i \in N$ represents a node and $A$ is the arc set where $a \in A$ represents an arc; $I(i)$ $(O(i))$ is again the set of incoming (outgoing) arcs to (from) node $i$, and $n$ is the total number of crew members available for the planning horizon. Let decision variable $x_a$ represent the flow on arc $a$, and $c_a$ be the associated arc cost. To take care of FIFO constraints, we define $A_a$ as the set of arcs on which there cannot be positive flow if there is a flow on arc $a$.

$$\text{Minimize} \quad \sum_{a \in A} c_a x_a \tag{51}$$

$$\text{subject to} \quad \sum_{a \in O(i)} x_a = 1 \quad \forall i \in N \text{ and } i \text{ is a departure node} \tag{52}$$

$$\sum_{a \in O(i)} x_a = 1 \quad \forall i \in N \text{ and } i \text{ is a supply node} \tag{53}$$

$$\sum_{a \in I(i)} x_a = n \quad \forall i \in N \text{ and } i \text{ is the demand node} \tag{54}$$

$$\sum_{a \in I(i)} x_a = \sum_{a \in O(i)} x_a \quad \forall i \in N \text{ and } i \text{ is a departure or arrival node} \tag{55}$$

$$\sum_{a' \in A_a} x'_a \le M(1 - x_a) \quad \forall a \in A \tag{56}$$

$$x_a \in \{0, 1\} \quad \forall a \in A. \tag{57}$$

The objective function (51) minimizes the total arc costs, hence the total crew salary, deadheading, and detentions. Constraint (52) ensures that each train gets a crew member. Contraints (53) and (54) satisfy the supply and demand balance of the supply and demand nodes, respectively. Constraint (55) satisfies the flow balance for all remaining nodes. Constraint (56) represents the FIFO policy for the crew assignments based on the prohibited arc flows for each arc. Constraint (57) is the integrality constraint for the decision variable.

It should be noted that so far we have not considered the crew delay in our space-time network, and hence in the formulation. To induce the crew delay possibility, we also create those arcs that may violate the minimum-stay requirement, assuming that if there is any flow on these arcs, it will correspond to the delay of a train at its head node, such that the minimum-stay requirement is met. In addition to other costs, we put a penalty equivalent to the crew delay cost on these arcs.

## 7.3. Solution Method and Computational Results

Railroads are obviously more interested in having tools that can be used in real time and that can produce a reasonably good solution. The formulation without the FIFO constraints is a simple minimum-cost network flow formulation. However, FIFO constraints bring a challenge, especially in terms of the solution time. The experiments in Vaidyanathan et al. [48] show that the problem without the FIFO constraints could be solved within five seconds to optimality, using CPLEX 8.0, for real-life problem data consisting of 436 trains, 99 crews,

Table 7. Results for a real-life data of the crew scheduling problem.

| Criteria | Current solution | UF solution | Improvement (%) |
|---|---|---|---|
| Crew delay | 48 hours | 0 hours | 100 |
| Deadheadings | 322 hours | 200 hours | 37.9 |
| Crew detention | 88 hours | 53 hours | 39.8 |

and three operating terminals over a period of seven days. Hence, they propose a postprocessing procedure that repairs the solution of the problem without the FIFO constraints, and finds an approximate solution to the problem with FIFO constraints.

The postprocessing procedure starts by listing the crew assignments according to the arrival times of incoming trains and departure times of outgoing trains at all terminals. Starting from the beginning of the planning horizon, we search through the end of the list by detecting the assignments that violate FIFO constraints. For each pair of assignments that violates the constraints, we basically first swap the crew assignment for the pair and then update the future path of both crews accordingly by exchanging the future assignments between the two crews. Note that during the postprocessing the deadheading between two terminals is considered as a train between those two terminals and handled in the same way as the other trains are. In addition, both the crew delay and number of deadheads remain unchanged.

This algorithm is tested on the real-life data we mentioned previously. We compare the results in Vaidyanathan et al. [48] with the current solution used by the railroad company on three criteria: crew delay, deadheadings, and crew detention. As indicated by the values in Table 7, the solution in Vaidyanathan et al. [48] (UF Solution) outperforms the current solution of the railroad on all criteria.

### 7.4. Concluding Remarks

For the sake of simplicity and clarity, the formulation of the crew scheduling problem given in this section is for a single crew type and a single crew district. Indeed, the crew scheduling problem of a particular railroad is involved with multiple crew types of several crew districts. Some crew districts share a set of common trains, and different crew types of the same district can operate the same set of trains unless prohibited by the rules that particularly govern the crew type. Hence, the planning problem should be capable of handling multiple crew types and multiple crew districts. The network flow formulation we discuss here can easily be adapted for this case. The resulting network is a multicommodity flow network, where each crew type corresponds to a commodity type. Hence, the network flow formulation is a multicommodity network flow problem. Note that elimination of FIFO constraints and the associated postprocessing procedure is still valid to attain a feasible solution to the problem. The challenge here lies in finding an efficient solution method for the multicommodity network flow problem (without FIFO constraints), which is known to be NP-hard. However, the problems and formulations we discuss in other sections also face this challenge, and some efficient algorithms have overcome this difficulty by designing particular algorithms for each class of problems.

## 8. Conclusions

In this chapter, we deal with the six most important planning and scheduling problems for railroads, and describe the main achievements that have been obtained so far. We initially give a detailed characterization of each problem, its context and importance, followed by the mathematical formulation, which is based on a convenient network flow representation; in addition, we discuss the main issues involved that make real instances difficult to model and solve. We also present cutting-edge algorithms that effectively solve instances found

in practice by railroads, as well as the results obtained, thus allowing railroads to achieve substantial savings.

We hope to have successfully evidenced to the reader that railroads pose some of the most exciting and challenging opportunities for the application of operations research tools and methodologies. In addition, our aim is to have demonstrated that difficult, combinatorial, NP-hard problems that appear in practice can be effectively solved by the proper choice of efficient and effective techniques, cleverly designed to take advantage of the special characteristics and nature of each problem.

We also want to emphasize here that if operations research is to be acknowledged as *the science of better* and is effectively applied to solve real-world problems and to deliver premium value to the world's organizations, we must go beyond the current paradigms that drive the problems we choose to tackle, the way we model them, and how they are solved.

To summarize, if we want to develop and provide operations research tools that help organizations find near-optimal solutions that can be applied and used in practice, resulting in substantial savings, not to mention the benefits in terms of quality, improved level of service, and customer satisfaction and reliability, to name a few, we must be *innovative* in modeling and solving them. There are plenty of challenging opportunities involving the overall railroad business, given its intrinsic complexity and the interrelationships between problems, as we tried to demonstrate in this chapter.

## References

[1] B. Adenso-Diaz, M. O. Gonzalez, and P. Gonzalez-Torre. On-line timetable re-scheduling in regional train services. *Transportation Research B* 33:387–398, 1999.

[2] R. K. Ahuja, K. C. Jha, and J. Liu. Solving real-life railroad blocking problems. *Operations Research*, 2004. Forthcoming.

[3] R. K. Ahuja, J. Liu, and G. Sahin. Railroads yard location problem. Working paper, Innovative Scheduling Systems, Inc., Gainesville, FL, 2005.

[4] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, NJ, 1993.

[5] R. K. Ahuja, O. Ergun, J. B. Orlin, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics* 123:75–102, 2002.

[6] R. K. Ahuja, J. Liu, A. Mukherjee, J. B. Orlin, D. Sharma, and L. A. Shughart. Solving real-life locomotive scheduling problems. *Transportation Science*. Forthcoming.

[7] A. Assad. Models for rail transportation. *Transportation Research A* 14:205–220, 1980.

[8] C. Barnhart, H. Jin, and P. H. Vance. Railroad blocking: A network design application. *Operations Research* 48:603–614, 2000.

[9] L. D. Bodin, B. L. Golden, A. D. Schuster, and W. Romig. A model for the blocking of trains. *Transportation Research B* 14:115–120, 1980.

[10] U. Brännlund, P. O. Lindberg, A. Nõu, and J.-E. Nilsson. Railway timetabling using Lagrangian relaxation. *Transportation Science* 32(4):358–369, 1998.

[11] X. Cai, C. J. Goh, and A. I. Mees. Greedy heuristics for rapid scheduling of trains on a single track. *IIE Transactions* 30:481–493, 1998.

[12] A. Caprara, M. Fischetti, and P. Toth. Modeling and solving the train timetabling problem. *Operations Research* 50(5):851–861, 2002.

[13] A. Caprara, M. Fischetti, P. L. Guida, P. Toth, and D. Vigo. Solution of large-scale railway crew planning problems: The Italian experience. *Lecture Notes in Economics and Mathematical Systems*, Vol. 471, 1–18, 1999.

[14] M. Carey and D. Lockwood. A model, algorithms and strategy for train pathing. *Journal of the Operational Research Society* 46:988–1005, 1995.

[15] P. Carraresi and G. Gallo. Network models for vehicle and crew scheduling. *European Journal of Operational Research* 16:139–151, 1984.

[16] K. C. Chih, M. A. Hornung, M. S. Rothenberg, and A. L. Kornhauser. Implementation of a real time locomotive distribution system. T. K. S. Murthy, R. E. Rivier, G. F. List, and J. Mikolaj, eds. *Computer Applications in Railway Planning and Management*. Computational Mechanics Publications, Southampton, UK, 1990.

[17] S. C. K. Chu and E. C. H. Chan. Crew scheduling of light rail transit in Hong Kong: From modeling to implementation. *Computers and Operations Research* 25(11):887–894, 1998.

[18] J.-F. Cordeau, F. Soumis, and J. Desrosiers. A Benders' decomposition approach for the locomotive and car assignment problem. Technical Report G-98-35, GERAD, École de Hautes Études Commerciales de Montréal, Montréal, Quebec, Canada, 1998.

[19] J.-F. Cordeau, P. Toth, and D. Vigo. A survey of optimization models for train routing and scheduling. *Transportation Science* 32:988–1005, 1998.

[20] T. G. Crainic and J.-M. Rosseau. Multimode, multicommodity freight transportation: A general modelling and algorithmic framework for the service network design problem. *Transportation Research B* 20(3):225–242, 1986.

[21] T. G. Crainic, J. A. Ferland, J.-M. Rousseau. A tactical planning model for rail freight transportation. *Transportation Science* 18(2):165–184, 1984.

[22] M. Fischetti and P. Toth. A package for locomotive scheduling, Technical Report DEIS-OR-97-16, University of Bologna, Bologna, Italy, 1997.

[23] M. Florian, G. Bushell, J. Ferland, G. Guerin, and L. Nastansky. The engine scheduling problem in a railway network. *INFOR* 14:121–138, 1976.

[24] M. A. Forbes, J. N. Holt, and A. M. Watts. Exact solution of locomotive scheduling problems. *Journal of Operational Research Society* 42:825–831, 1991.

[25] M. F. Gorman. An application of genetic and tabu searches to the freight railroad operating plan problem. *Annals of Operations Research* 78:51–69, 1998.

[26] E. Haghani. Rail freight transportation: A review of recent optimization models for train routing and empty car distribution. *Journal of Advanced Transportation* 21:147–172, 1987.

[27] A. E. Haghani. Formulation and solution of a combined train routing and makeup, and empty car distribution model. *Transportation Research B* 23(6):433–452, 1989.

[28] A. Higgins, E. Kozan, and L. Ferreira. Optimal scheduling of trains on a single line track. *Transportation Research B* 30:147–161, 1996.

[29] A. Higgins, E. Kozan, and L. Ferreira. Heuristic techniques for single line tran scheduling. *Journal of Heuristics* 3:43–62, 1997.

[30] P. Ireland, R. Case, J. Fallis, C. Van Dyke, J. Kuehn, and M. Meketon. The Canadian Pacific Railway transforms operations research by using models to develop its operating plans. *Interfaces* 34(1):5–14, 2004.

[31] M. H. Keaton. Designing optimal railroad operating plans—Lagrangian-relaxation and heuristic approaches. *Transportation Research B* 23(6):415–431, 1989.

[32] M. H. Keaton. Service-cost tradeoffs for carload freight traffic in the United States rail industry. *Transportation Research A* 25(6):363–374, 1991.

[33] M. H. Keaton. Designing railroad operating plans—A dual adjustment method for implementing lagrangian-relaxation. *Transportation Science* 26(4):263–279, 1992.

[34] D. Kraay and P. Harker. Real-time scheduling of freight networks. *Transportation Research B* 29(3):213–229, 1994.

[35] L. G. Kroon and L.W. Peeters. A variable trip time model for cyclic railway timetabling. *Transportation Science* 37(2):198–212, 2003.

[36] J. E. Leal, A. C. Soares, and L. C. N. Nunes. A heuristic approach for the train scheduling problem on single railway tracks. *ANPET Congress*, Vol. 2. ANPET, Brazil, 945–954, 2004.

[37] M. Newman, L. Nozick, and C. A. Yano. Optimization in the rail industry. P. P. Pardalos and M. G. C. Resende, eds. *Handbook of Applied Optimization*. Oxford University Press, New York, 2002.

[38] H. N. Newton. Network design under budget constraints with application to the railroad blocking problem. Ph.D. thesis, Industrial and Systems Engineering Department, Auburn University, Auburn, AL, 1996.

[39] H. N. Newton, C. Barnhart, and P. M. Vance. Constructing railroad blocking plans to minimize handling costs. *Transportation Science* 32:330–345, 1998.

[40] A. Nou, J. Desrosiers, and F. Soumis. Weekly locomotive scheduling at Swedish State Railways. Technical Report G-97-35, GERAD, École des Hautes Études Commerciales de Montréal, Montréal, Quebec, Canada, 1997.

[41] K. V. Ramani. An information system for allocating coach stock on Indian Railways. *Interfaces* 11(3):44–51, 1981.

[42] R. E. Rivier and P. Tzieropoulos. Interactive Graphic Models for Railway Operational Planning. M. Florian, ed. *The Practice of Transportation Planning*. Elsevier Science Publishers, Amsterdam, 1984.

[43] R. E. Rivier and P. Tzieropoulos. Computer-aided planning of railway networks, lines and stations. T. K. S. Murthy, L. S. Lawrence, and R. E. Rivier, eds. *Computers in Railway Management*. Computational Mechanics Publications, Berlin, Germany, 1987.

[44] İ. Şahin. Railway traffic control and train scheduling based on inter-train conflict management. *Transportation Research B* 33:511–534, 1999.

[45] G. Şahin, R. K. Ahuja, and C. B. Cunha. New approaches for the train dispatching problem. Working paper, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, 2004.

[46] R. L. Sauder and W. Westerman. Computer aided train dispatching: Decision support through optimization. *Interfaces* 13(6):24–37, 1993.

[47] S. Smith and Y. Sheffi. Locomotive scheduling under uncertain demand. *Transportation Research Record* 1251:45–53, 1988.

[48] B. Vaidyanathan, R. K. Ahuja, and K. C. Jha. A network flow approach for railroad crew scheduling. Working paper, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, 2004.

[49] M. B. Wright. Applying stochastic algorithms to a locomotive scheduling problem. *Journal of Operational Research Society* 40:187–192, 1989.

[50] G. Yu. *Operations Research in the Airline Industry*. Kluwer Academic Publishers, Boston, MA, 1998.

[51] K. Ziarati, F. Soumis, J. Desrosiers, and M. M. Solomon. A branch-first, cut-second approach for locomotive assignment. *Management Science* 45:1156–1168, 1999.

[52] K. Ziarati, F. Soumis, J. Desrosiers, S. Gelinas, and A. Saintonge. Locomotive assignment with heterogeneous consists at CN North America. *European Journal of Operational Research* 97:281–292, 1997.

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS | ISBN 1-877640-21-2
DOI 10.1287/educ.1053.0018

# Analyzing the Vulnerability of Critical Infrastructure to Attack and Planning Defenses

***Gerald G. Brown, W. Matthew Carlyle, Javier Salmerón, and Kevin Wood***
Operations Research Department, Naval Postgraduate School, Monterey, California 93943
{gbrown@nps.edu, mcarlyle@nps.edu, jsalmero@nps.edu, kwood@nps.edu}

**Abstract** We describe new bilevel programming models to (1) help make the country's critical infrastructure more resilient to attacks by terrorists, (2) help governments and businesses plan those improvements, and (3) help influence related public policy on investment incentives, regulations, etc. An intelligent attacker (terrorists) and defender (us) are key features of all these models, along with information transparency: These are Stackelberg games, as opposed to two-person, zero-sum games. We illustrate these models with applications to electric power grids, subways, airports, and other critical infrastructure. For instance, one model identifies locations for a given set of electronic sensors that minimize the worst-case time to detection of a chemical, biological, or radiological contaminant introduced into the Washington, D.C. subway system. The paper concludes by reporting insights we have gained through forming "red teams," each of which gathers open-source data on a real-world system, develops an appropriate attacker-defender or defender-attacker model, and solves the model to identify vulnerabilities in the system or to plan an optimal defense.

**Keywords** critical infrastructure protection; bilevel program; mixed-integer program; homeland security

## The Problem

What is critical infrastructure? The National Strategy for Homeland Security deems 13 infrastructure sectors critical to the United States; see Table 1 (DHS [18]). These include sectors such as "Government" and "Public Health," but a number, such as "Transportation" and "Information and Telecommunications," comprise physical systems that connect components of our economy: In essence, they enable the transfer and distribution of our economy's life forces. We focus on defending this type of infrastructure from attacks by terrorists, but we believe almost any type of critical infrastructure deserves analysis with the techniques we describe.

Any critical infrastructure system represents a huge investment of our nation's wealth, and minor disruptions to such a system's components—these disruptions can be random or deliberate—can severely degrade its performance as well as the performance of dependent systems. For instance, a massive power outage can result from the failure of just a few key lines and protective circuit breakers (U.S.-Canada Power System Outage Task Force [39]). The direct effect is to interrupt the energy supply to residential and industrial customers, but all other infrastructure systems listed in Table 1 will be affected if the power outage lasts long enough. So, how do we carry out a "vulnerability analysis" when terrorist attacks are the key concern? That is, how do we analyze the vulnerability of a critical infrastructure system to a terrorist attack, or set of coordinated attacks, and make informed proposals for reducing that vulnerability?

Most infrastructure systems are engineered to handle disruptions that result from accidents, or from random acts of nature, with little or no degradation in performance. Real-time reliability assessment of an electric power grid pronounces the system robust if no crippling "single point of failure" exists (e.g., Wood and Wollenberg [44]). Analysts of transportation

TABLE 1. Thirteen infrastructure sectors critical to the United States, as defined by the Department of Homeland Security [18].

| | |
|---|---|
| Agriculture | Food |
| Water | Public Health |
| Emergency Services | Government |
| Defense Industrial Base | Information and Telecommunications |
| Energy | Transportation |
| Banking and Finance | Chemical Industry |
| Postal and Shipping | |

systems, power plants, and other infrastructure often use fault trees to assess vulnerability (Roberts et al. [34]). Such an assessment helps identify minimal sets of events, or "cutsets," that are most likely to disrupt the system, and pronounce the system robust if their combined probability is sufficiently low. This assessment can suggest changes to the system to improve robustness, and the overall methodology can be used to evaluate alternative system configurations proposed by the analyst.

However, infrastructure that resists single points of random failure, or whose cutsets have low occurrence probabilities, may not survive a malicious, intelligent attack. For example, a lone attacker with a high-powered rifle could gravely damage an entire electric power grid by targeting highly reliable components at just a few key substations. (We reach this conclusion from our own analyses of electric power grids and from reports of gunfire disabling a substation; see Wallace [41].) Also, cutsets that are likely to occur due to random causes may not share any similarities to the cutsets that an attacker will likely find. An analyst might attempt a fault-tree assessment of a system subject to attack by guessing at the probability that each individual component might be attacked. In fact, such analysis is practiced (Garcia [22]), but the results must be classified as a guesses. We require a new paradigm for vulnerability analysis.

The new paradigm must account for an adversary's ability to collect information about an infrastructure system and use that information to identify weak spots in the system's architecture. A captured Al Qaeda training manual (Department of Justice [19]) advises: "Using public sources openly and without resorting to illegal means, it is possible to gather at least 80% of information about the enemy." We interpret that statement to mean: "It is possible to gather, from public sources, at least 80% of the information needed to plan a highly disruptive attack on an infrastructure system." Our experience indicates that one can often find *all* the information necessary to plan such an attack.

Our backgrounds compel us ask how a military analyst, faced with an intelligent enemy, would approach vulnerability analysis for military infrastructure. First, the analyst would assume that our infrastructure will be attacked and that we must take steps to protect it, i.e., harden the infrastructure or improve its active defenses. The budget for hardening or actively defending infrastructure will always be limited. So, typically, the analyst would be instructed to create a prioritized list of "defended assets" most in need of protection, along with a list of potential defensive measures, and deliver those lists to higher-level decision makers. The latter parties would make the final decisions after balancing costs, effectiveness, and intangibles, and after determining the true budget (which may be monetary or may be the number of aerial sorties, cruise missiles, tanks, etc., that can be spared for defensive purposes). Table 2 shows the doctrinal components that the U.S. Army uses to guide the prioritization of its defended assets (as well as its enemies').

Any person who has had a course in discrete optimization understands the fundamental flaw in the concept and use of a prioritized list. In addition to that shortcoming of the nominal military approach, we see that the civilian problem itself differs from the military one:

- almost every civilian U.S. asset is susceptible to surveillance or attack, and is thus vulnerable;

Table 2. Criteria for prioritizing defended assets (Department of the Army [20, 21]).

| | |
|---|---|
| Criticality | How essential is the asset? |
| Vulnerability | How susceptible is the asset to surveillance or attack? |
| Reconstitutability | How hard will it be to recover from inflicted damage, considering time, special repair equipment, and manpower required to restore normal operation? |
| Threat | How probable is an attack on this asset? |

- no matter how hard it is to recover from inflicted damage, we will, eventually, reconstitute and recover; and
- military planners have vast experience in determining the likelihood of alternative attacks; homeland-security planners do not. Thus, we must plan for what is possible, rather than what subjective assessments indicate is likely.

In fact, normally, we do not try to measure the importance, or value, of an asset directly. Rather, we model a complete infrastructure system, its value to society, and how losses of the system's components reduce that value, or how improvements in the system mitigate against lost value. The exact meaning of value will depend on the system under investigation: It may mean economic output, time to detection of a toxic substance, etc., and sometimes cost, the converse of value, will be a more convenient yardstick.

Al Qaeda teaches as its primary mission "overthrow of godless regimes (by) gathering information about the enemy, the land, the installations, and the neighbors, ... blasting and destroying the places of amusement, ... embassies, ... vital economic centers, ... bridges leading into and out of cities, ...." (Department of Justice [19]). Al Qaeda may not have a perfect model of a particular infrastructure system, but its operatives are instructed to gather (widely available) information about it. Clearly, that information is being used to plan the worst attacks Al Qaeda can devise. Consequently, prudence dictates the assumption that Al Qaeda, or any other terrorist organization, will use its limited offensive assets to maximize damage to the infrastructure system it decides to attack, and has all the data necessary to do this.

Our paradigm of an *attacker-defender model* does address criticality, vulnerability, reconstitutability, and threat, but in a very different way than military planners might. We incorporate reconstitutability by modeling how system components are repaired over time and how a repaired component contributes to improved system value (Salmerón et al. [36]). Unless strictly defended or hardened, every system component is assumed to be vulnerable. We address "threat" by positing different levels of offensive resources for the terrorists. At the end of our analysis, we can determine the criticality of a *group* of system components, i.e., the value of protecting them, hardening them, or the value of adding new components into the system for purposes of redundancy. Another paradigm, discussed later, directly identifies an optimal defense plan: This is the *defender-attacker model.*

To understand our approach, the reader must understand the basics of the next two sections. However, a reader not interested in the mathematics may feel free to skim those details.

## Attacker-Defender Models

The core of an attacker-defender model is an optimization model of an infrastructure system whose objective function represents the system's value to society while it operates, or the cost to society when the system loses functionality. For instance, the maximum throughput

of an oil pipeline system might measure that system's value, while power-generation costs, plus economic losses resulting from unmet demand, might measure the full cost of operating an electric power grid.

To set the mathematical context, we assume that the defender operates a system so as to minimize cost that can be represented by a linear function. The defender's problem is

$$\min_{\mathbf{y}\in Y} \mathbf{cy}, \tag{1}$$

where (i) $\mathbf{y}$ represents system operating decisions or activities, (ii) $\mathbf{c}$ defines the corresponding vector of costs (and/or penalties), and (iii) the set $Y$ represents constraints on that operation and the requirements to be met, e.g., road capacities in a road network, the number of commuters wishing to travel between various points in that network, etc. Of course, by including auxiliary variables in $\mathbf{y}$, and auxiliary constraints in $Y$, we can also represent certain nonlinear cost functions.

We note that "defender" is actually a misnomer in these models, because the models do not directly represent defensive actions; better terms might be "system user" or "system operator." However, our ultimate goal is to help identify defensive actions for the system user, so we feel justified in the slight abuse of terminology.

Now, our model posits that an attacker wishes to maximize the defender's optimal (minimal) operating cost, and will do so by restricting actions $\mathbf{y}$. Let $x_k = 0$ if the attacker *interdicts* the defender's $k$th asset, let $x_k = 0$ otherwise, and let $\mathbf{x}$ denote the vector of interdiction decisions. "Interdicting an asset" may be viewed as interdicting some component of the defender's infrastructure system. For simplicity in this paper, we assume that if $x_k = 1$, then $y_i = 0$ for any activity $i$ that requires asset $k$. That is, interdiction of an asset stops the defender from carrying on activities that depend on that asset. We note that defender-attacker models often exhibit a one-to-one relationship between assets and activities; for example, interdiction of a pipeline segment between cities $a$ and $b$ stops the single activity that can occur on that segment, "flow from $a$ to $b$."

Binary restrictions on $\mathbf{x}$, and some reasonable set of resource limitations on the attacker's resources, are represented by $\mathbf{x} \in X$. We represent the defender's set of feasible actions, restricted by interdictions $\mathbf{x}$, as $Y(\mathbf{x})$. Thus, the attacker solves this problem to guide his attacks:

$$\text{(MAX-MIN)} \quad \max_{\mathbf{x}\in X} \min_{\mathbf{y}\in Y(\mathbf{x})} \mathbf{cy}. \tag{2}$$

MAX-MIN is a type of bilevel program (e.g., Moore and Bard [29]), which is an instance of a Stackelberg game (von Stackelberg [40]). The attacker leads with an attack and the defender follows with a response, hence the standard phrases *leader* and *follower*, for attacker and defender, respectively. The key assumption here is that the attacker has a perfect model of how the defender will optimally operate his system, and the attacker will manipulate that system to his best advantage. That is a strong but prudent assumption for the defender: He can suffer no worse if the attacker plans his attacks using a less-than-perfect model of the defender's system. We find no difficulties in assuming that the defender will operate his system optimally, but a simple adjustment to the objective function can account for certain types of inefficiencies. (More general models of inefficiency seem unsupportable. For instance, one might be able to model a defender who always operates his system at a random point along, say, the "90%-efficiency frontier," but such a model would be hard to solve and, more importantly, hard to justify.)

One can devise many supportable generalizations of MAX-MIN including attacks that increase costs rather than limiting activities, or attacks that reduce the capacity of an asset less than 100%. We will cover some of these generalizations after establishing basic results.

Naturally, the defender may also lack perfect knowledge of the attacker's capabilities. That is, the defender may be guessing at the interdiction-resource constraints contained within $\mathbf{x} \in X$. However, the defender can solve the model over a range of posited interdiction resources and use those results to guide system improvements.

## Solving an Attacker-Defender Model

For many situations, a linear program (LP) will provide an adequate model of the defender's system and its operations. For instance, the electric power industry commonly employs linearized optimal power-flow models for security analysis (Wood and Wollenberg [44]). Therefore, we express the optimal operation of the defender's system as

$$\begin{aligned} \text{(D0)} \qquad & \min_{\mathbf{y}} \ \mathbf{cy} && (3)\\ & \text{s.t.} \quad A\mathbf{y} = \mathbf{b} && (4)\\ & \qquad F\mathbf{y} \le \mathbf{u} && (5)\\ & \qquad \mathbf{y} \ge 0. && (6) \end{aligned}$$

Constraints (4) correspond to general system-operations constraints (e.g., balance of flow in a transportation network), and constraints (5) represent capacity limitations for asset $k \in K$ (e.g., maximum flow across the $k$th network link, per unit of time). Assets might include power lines, pipelines, roads, ports, communications hubs, etc.

The attacker's interdictions might affect the system in any number of ways, but let us assume that only "assets" are in danger of being interdicted, and that interdiction of asset $k$ causes the loss of all its capacity $u_k$. Thus, the full attacker-defender model is

$$\begin{aligned} \text{(AD0)} \qquad & \max_{\mathbf{x}\in X} \min_{\mathbf{y}} \ \mathbf{cy}\\ & \text{s.t.} \quad A\mathbf{y} = \mathbf{b}\\ & \qquad F\mathbf{y} \le U(\mathbf{1}-\mathbf{x})\\ & \qquad \mathbf{y} \ge \mathbf{0} \end{aligned}$$

where $U = \text{diag}(\mathbf{u})$. We assume that the inner LP has been constructed to be feasible for any $\mathbf{x}$, because we expect the system to operate in some degraded fashion for any conceivable attack. This may require the use of auxiliary variables that are not susceptible to interdiction.

A natural approach to reformulating this problem fixes $\mathbf{x}$ temporarily, takes the dual of the inner linear program, and then releases $\mathbf{x}$. Unfortunately, an unappealing, nonlinear, mixed-integer program results. That model can be linearized in some instances (e.g., Wood [43], Salmerón et al. [35]), but an alternative model comes to mind: Change the paradigm of capacity interdiction to "cost interdiction," and then take the dual of the inner problem. (See Cormican et al. [16] for the mathematical details.) Specifically, let $-\mathbf{p}$ strictly bound the set of dual variables associated with $F\mathbf{y} \le U(\mathbf{1}-\mathbf{x})$, taken over all possible values of $\mathbf{x} \in X$. Thus, $p_k$ bounds the value of a unit of asset $k$'s capacity for the defender. Because we assume that AD0 is feasible even when asset $k$ has been interdicted and has no capacity, it must be possible to set a cost on asset $k$'s capacity that makes it too costly to use: $p_k$ is just that cost. This is the standard approach to formulating an "elastic model"; see Brown et al. [8] for more discussion.

Thus, AD0 is equivalent to

$$\begin{aligned} \text{(AD1)} \qquad & \max_{\mathbf{x}\in X} \min_{\mathbf{y}} (\mathbf{c} + \mathbf{x}^T PF)\mathbf{y} \qquad \text{[Dual vars. for fixed } \mathbf{x}\text{]}\\ & \text{s.t.} \quad A\mathbf{y} = \mathbf{b} \ [\boldsymbol{\theta}]\\ & \qquad F\mathbf{y} \le \mathbf{u} \ [\boldsymbol{\beta}]\\ & \qquad \mathbf{y} \ge \mathbf{0}, \end{aligned}$$

where $P = \text{diag}(\mathbf{p})$. (Actually, nonstrict bounds $\mathbf{p}$ are also valid for identifying an optimal $\mathbf{x}$; see Cormican et al. [16].)

Now, when we take the dual of the inner minimization, a mixed-integer linear program (MILP) results:

$$\begin{aligned} \text{(AD1-MILP)} \qquad & \max_{\mathbf{x},\boldsymbol{\theta},\beta} \mathbf{b}^T\boldsymbol{\theta} + \mathbf{u}\boldsymbol{\beta} \\ & \text{s.t.} \quad A^T\boldsymbol{\theta} + F^T\boldsymbol{\beta} - F^T P\mathbf{x} \le \mathbf{c} \\ & \qquad\qquad\qquad\qquad\qquad\quad \mathbf{x} \in X \\ & \qquad\qquad\qquad\qquad\qquad\quad \boldsymbol{\beta} \le \mathbf{0}. \end{aligned}$$

We can solve this model directly, or by using Benders decomposition [4]. In fact, the standard Benders approach for integer $\mathbf{x}$ begins by taking the dual of AD1-MILP with $\mathbf{x}$ fixed, which obviously yields AD1. Thus, the max-min formulation of AD1 is a natural representation of the interdiction problem for application of Benders decomposition.

To illustrate with a concrete, albeit simplified, example, consider the following model of a crude-oil pipeline network:

**Data**

$A$ node-arc incidence matrix for the pipeline network

$\mathbf{b}$ vector of supplies and demands: $b_i > 0$ defines a supply of $b_i$ million barrels per day (mmbbl/day) at node $i$, $b_i < 0$ defines a demand of $b_i$ mmbbl/day at $i$, and $b_i = 0$ implies $i$ is a transshipment node (pumping station)

$\mathbf{c}_1$ vector of shipping costs by pipeline segment, i.e., arc ($/mmbbl/day)

$\mathbf{c}_2$ vector of penalties for not taking available supply ("take-or-pay penalties") ($/mmbbl/day)

$\mathbf{c}_3$ vector of penalties for unmet demand (e.g., spot-market cost) ($/mmbbl/day)

$\hat{I}_2$ incomplete diagonal matrix with a 1 for each supply node, but 0 elsewhere

$\hat{I}_3$ incomplete diagonal matrix with a 1 for each demand node, but 0 elsewhere

**Variables**

$\mathbf{y}_1$ flows on pipelines (mmbbl/day)

$\mathbf{y}_2$ unused supply (mmbbl/day)

$\mathbf{y}_3$ unmet demand (mmbbl/day)

**Formulation**

$$\begin{aligned} (\mathrm{D_P}0) \qquad & \min_{\mathbf{y}} \ \mathbf{c}_1\mathbf{y}_1 + \mathbf{c}_2\mathbf{y}_2 + \mathbf{c}_3\mathbf{y}_3 \\ & \text{s.t.} \quad A\mathbf{y}_1 - \hat{I}_2\mathbf{y}_2 + \hat{I}_3\mathbf{y}_3 = \mathbf{b} \qquad (7) \\ & \qquad\qquad I\mathbf{y}_1 \le \mathbf{u} \qquad (8) \\ & \qquad\qquad \text{all variables} \ge 0. \end{aligned}$$

Constraints (7) are elastic flow-balance constraints, and constraints (8) represent pipeline capacities. For simplicity, we

1. have ignored the oil's purchase price,
2. will assume $\mathbf{c}_2 = \mathbf{0}$ and $\mathbf{c}_1 > \mathbf{0}$,
3. set all unmet demand penalties equal, i.e., $\mathbf{c}_3 = (c_3\ c_3 \ldots c_3)$, and
4. assume that only pipeline segments can be interdicted (not, say, pumping stations).

Now we proceed directly to create a cost-interdiction model in the form of AD1. Let $x_k = 1$ if the attacker interdicts asset $k$, let $x_i = 0$ otherwise, and let $\mathbf{x} \in X$ denote the binary restrictions on $\mathbf{x}$ along with some plausible resource constraints. For example, intelligence indicates that the attacker can form at most $T$ squads to carry out simultaneous attacks, so $X = \left\{x_i \in \{0,1\}\ \forall\, i \in I \,\middle|\, \sum_{i\in I} x_i \le T\right\}$. We further note that $p = c_3$ exceeds the penalty

incurred by not supplying one mmbbl/day (because $\mathbf{c}_1 > \mathbf{0}$). Thus, letting $\mathbf{p} = (c_3\, c_3 \cdots c_3)$ and $P = \text{diag}(\mathbf{p})$, the max-min interdiction model is

$$\begin{aligned} (\text{AD}_\text{P}1) \qquad \max_{\mathbf{x}\in X} \min_{\mathbf{y}} & (\mathbf{c}_1 + \mathbf{x}^T P)\mathbf{y}_1 + \mathbf{c}_2\mathbf{y}_2 + \mathbf{c}_3\mathbf{y}_3 \\ \text{s.t.} \quad & A\mathbf{y}_1 - \hat{I}_2\mathbf{y}_2 + \hat{I}_3\mathbf{y}_3 = \mathbf{b} \\ & I\mathbf{y}_1 \le \mathbf{u} \\ & \text{all variables} \ge 0. \end{aligned}$$

We leave it to the reader to take the dual of the inner minimization to create ADP1-MILP, but a caveat is in order: The quality of the LP relaxation of that MILP will depend directly on how small the penalties $p_k$ are, and the modeler may need to expend some effort in identifying small, valid values. For instance, each $p_i$ in ADP1 can be validly reduced to $p_k - c_{1,\min} + \varepsilon$ where $c_{1,\min}$ is the smallest shipping cost a demand might incur while being satisfied, and where $\varepsilon$ is some small, positive constant.

In some instances, a cost-interdiction model like AD1 can actually be a more natural paradigm than AD0. In such cases, the analyst can avoid the AD0-to-AD1 transition and will not have to worry about bounds on dual variables. For instance, suppose D0, with constraints (5) eliminated, corresponds to a shortest-path problem on a road network. In some situations, we may replace the capacity constraints by modeling the interdiction of a link $k$ in the network as a delay $d_k$ on the nominal length $c_k$ (transit time). Thus, this model becomes:

$$\begin{aligned} (\text{AD}_\text{R}1) \qquad \max_{\mathbf{x}\in X} \min_{\mathbf{y}} & (\mathbf{c} + \mathbf{x}^T D)\mathbf{y} \\ \text{s.t.} \quad & A\mathbf{y} = \mathbf{b} \\ & \mathbf{y} \ge \mathbf{0}, \end{aligned}$$

where $D = \text{diag}(\mathbf{d})$, with $\mathbf{d}$ being the vector of delays $d_k$. See Israeli and Wood [25] for details on this model and solution techniques for it. We note that $\text{AD}_R1$ also fits into the framework of defender-attacker models, described next; § 4.4 provides an example.

## Defender-Attacker Models

By solving an attacker-defender model, we identify a set of most critical components for an infrastructure system. This leads to some obvious heuristics for solving an "optimal defense problem," i.e., identifying the best possible defense plan given a limited defense budget. We prefer truly optimal solutions, however.

In theory, one merely embeds the bilevel attacker-defender model in a trilevel defender-attacker-defender model such as

$$\min_{\mathbf{z}\in Z} \max_{\mathbf{x}\in X(\mathbf{z})} \min_{\mathbf{y}\in Y(\mathbf{x})} \mathbf{c}\mathbf{y}. \tag{9}$$

Here, $\mathbf{z}$ denotes a binary vector of defense decisions ($z_k = 1$ if asset $k$ is hardened and made invulnerable, say, and $z_k = 0$, otherwise); $\mathbf{z} \in Z$ denotes the binary restrictions on $\mathbf{z}$ together with budgetary (and possibly other) constraints; and the inner max-min problem simply represents an attacker-defender model with a restricted set of attack strategies, $X(\mathbf{z})$. Thus, the defender seeks to identify a defense plan $\mathbf{z}^*$ so that when the attacker solves

$$\max_{\mathbf{x}\in X(\mathbf{z}^*)} \min_{\mathbf{y}\in Y(\mathbf{x})} \mathbf{c}\mathbf{y}, \tag{10}$$

the benefit the attacker sees, i.e., the damage the attacker can guarantee to inflict, is minimized.

Unfortunately, these trilevel problems solve only with extreme difficulty, and no conversion to an MILP appears possible, in general. (See Israeli and Wood [25] for more details, and for the description of one special-case solution technology.)

Fortunately, certain optimal-defense problems lend themselves to easier bilevel, defender-attacker models. The defender becomes the leader in this new Stackelberg game, so we essentially reverse the meanings of $\mathbf{x}$ and $\mathbf{y}$, and make the following definitions:

**Indices**

$k$ asset the defender might want to defend, and the attacker might want to attack (this simple defender-attacker model assumes a one-to-one relationship between potentially attacked and potentially defended assets)

**Data**

$c_k$ value to the attacker of attacking undefended asset $k$ (vector form $\mathbf{c}$)

$p_k$ reduction in value of attacking the defender's $k$th asset if that asset is defended, i.e., the attacker receives benefit $c_k + p_k$, $p_k \le 0$, by attacking defended asset $k$ (vector form $\mathbf{p}$)

**Variables**

$$x_k = \begin{cases} 1 & \text{if the defender defends his } k\text{th asset} \\ 0 & \text{otherwise} \end{cases}$$

$$y_k = \begin{cases} 1 & \text{if the attacker attacks the defender's } k\text{th asset} \\ 0 & \text{otherwise} \end{cases}$$

$\mathbf{x}, \mathbf{y}$ vector forms of $y_k$ and $x_k$, respectively

**Constraints**

$\mathbf{x} \in X$ resource constraints and binary restrictions on the defender's defense plan, e.g., $X = \{\mathbf{x} \in \{0,1\}^n \mid G\mathbf{x} \le \mathbf{f}\}$

$\mathbf{y} \in Y$ resource constraints and binary restrictions on the attacker's attack plan, e.g., $Y = \{\mathbf{y} \in \{0,1\}^n \mid A\mathbf{y} = \mathbf{b}\}$

**Formulation**

$$\text{(DA1)} \qquad \min_{\mathbf{x} \in X} \max_{\mathbf{y}} (\mathbf{c} + \mathbf{x}^T P)\mathbf{y}$$
$$\text{s.t. } \mathbf{y} \in Y.$$

A simplified example illustrates. Suppose intelligence reports indicate that a terrorist organization, "the attacker," intends to send out $b$ teams to attack $b$ different subway stations in a city encompassing $M > b$ total stations. Municipal authorities, "the defender," have $m$ teams, $m < M$, with which to defend stations; a defended station becomes invulnerable to attack. The value to the defender of station $k$ is $c_k > 0$, and we assume the attacker assigns the same values. (If not, the defender's optimal defense plan may perform better than predicted.) Let $p_k = -c_k$; thus, if station $k$ is defended, the attacker will gain no benefit by attacking it. This "subway-defense problem" may be formulated as

$$\text{(DA1}_{\text{SUB}}\text{)} \qquad \min_{\mathbf{x} \in X} \max_{y} \sum_{k=1}^{M} (c_k + x_k p_k) y_k \tag{11}$$

$$\text{s.t.} \quad \sum_{k=1}^{M} y_k = b \tag{12}$$

$$y_k \in \{0,1\} \quad \forall k \tag{13}$$

where $X = \{\mathbf{x} \in \{0,1\}^M \mid \sum_{k=1}^{M} x_k = m\}$.

In general, the model DA1 and instances like $\text{DA1}_{\text{SUB}}$ are difficult to solve because the inner minimization is not an LP. Thus, no general transformation exists to convert DA1 into an MILP as we converted AD1 into AD1-MILP. This situation can be resolved in several ways:

1. We decide that continuous attack effort represents a reasonable approximation of reality and convert $Y$ to $Y_{CONT} = \{\mathbf{y} \in R^n_+ \mid A\mathbf{y} = \mathbf{b}\}$ (Golden [23]).

2. The LP relaxation of $Y$, $Y_{LP} = \{\mathbf{y} \in R^n_+ \mid A\mathbf{y} = \mathbf{b}\}$, yields intrinsically binary solutions, so a conversion from DA1 into "DA1-MILP" is, in fact, possible. This is the situation with $\text{DA}_{\text{SUB}}1$, and we invite the reader to work out the details. See Brown et al. [9] for an example involving theater ballistic-missile defense.

3. Or, neither of the cases above pertains, and we really must include restriction $\mathbf{y} \in \{0,1\}^n$ in the definition of $Y$.

Case 3 requires special techniques to solve, but solution methods better than brute-force enumeration do exist (e.g., Israeli and Wood [25], Brown et al. [10]). This paper focuses on the simpler case, Case 2.

## What We Have Done

A terrorist organization can learn just about everything it needs to know to plan a perfect attack on our critical infrastructure. This key insight leads us to apply attacker-defender and defender-attacker models to problems of protecting this infrastructure. This section describes a number of these models (the first embedded in a complete decision-support system), along with applications.

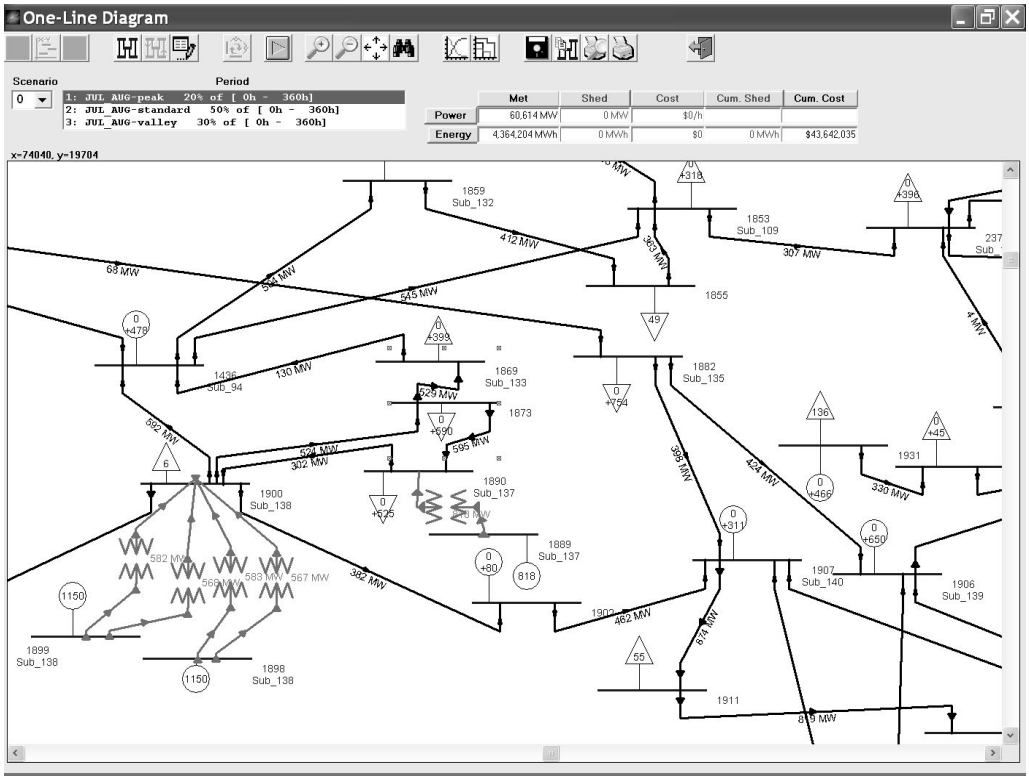
These models reflect our experience as military planners who have been asked to help target enemy infrastructure and defend our own infrastructure, such as road, communication, electric power, and pipeline networks. Most of the models have been derived in the course of our research and/or our students'. We have been fortunate to be able to test many of these models by (i) defining a hypothetical but real-world scenario; (ii) assembling a "red team" of well-trained, military officer-students to gather scenario data from strictly public sources; (iii) guiding the team in building, instantiating, and running an appropriate model, and then analyzing the results.

The results are always interesting, and usually lead to valuable insights. We find cases in which a given set of attackers can do more damage than we would have predicted, or less; and sometimes the attacks do not target the "obvious" components revealed in single-point-of-failure analyses. An anecdote illustrates this last point. Suppose that a terrorist organization wants to attack and close down the operations of a specific airline, at a single airport, for the purpose of disrupting the airline's finances. Based on passenger-revenue data obtained from the Internet, a red-team analysis indicates that "City A" is the most damaging airport to strike for one large U.S. airline. If the terrorists can afford two strikes, Cities B and C would be best (Brown et al. [7]).

### Electric Power Grids: An Attacker-Defender Model

We have produced a decision-support system called the Vulnerability of Electric Grids Analyzer (VEGA) (Salmerón et al. [37]), which identifies an optimal or near-optimal attack (i.e., a set of coordinated attacks) on an electric power grid. VEGA also animates the system operator's optimal response to that attack. Given a scenario extracted from an electric grid database and an assessment of the level of effort needed for an attacker to target each component, VEGA determines, and illustrates graphically, which equipment-loss patterns lead to maximal damage measured in terms of load (demand for power) that must be shed (dropped). Figure 1 depicts one of VEGA's many interface screens. We note that VEGA has been built with the intention of analyzing regional, bulk-power transmission systems as opposed to local distribution systems, but it could certainly be used in the latter case.

FIGURE 1. A screen shot of the "one-line diagram" view in VEGA.




*Note.* The icons depict generators, transformers, buses, transmission lines, and customer demands. For any given level of attacker capability, VEGA finds an optimal or near-optimal target set of vulnerable components (Salmerón et al. [36].)

In VEGA, an "optimal DC power-flow model" comprises D0, the inner, minimizing LP. This model incorporates elastic current-balance (flow-balance) constraints along with linearized admittance constraints for AC lines. This power-flow model approximates the "true" active power flows and disregards reactive power flows, but the electric power industry normally deems this approximation adequate for analyzing system security.

When an electric grid possesses sufficient generating and transmission capacity to meet all demand, the power-flow model reflects how a system operator would set generating levels to minimize cost. When capacity is insufficient, as after an attack, the model reflects how the operator will react to minimize the amount of load shed, while using generation cost as a secondary criterion.

Given a fixed attack plan, VEGA must solve a sequence of power-flow models. This is true because we normally model long-term unmet demand for energy (amount of load shed, integrated over time), taking into account (i) differing repair times for components, as well as (ii) daily demand variations ("load duration curves"), and (iii) seasonal demand variations. Modeling restoration is crucial because damaged transmission lines might be repaired in a few days, other components might be repaired in a week or two, but a damaged transformer might take many months to replace. Transformers pose special difficulties because they are big, heavy, and expensive; few spares exist; and a replacement might have to be ordered from, built by, and shipped from an overseas manufacturer.

An attacker-defender model can be embedded in a formal trilevel model to optimize the upgrading or hardening of a system against terrorist attack (Israeli and Wood [25]); see also §3 in this chapter). Such models exist for electric grids, but real-world instances are impossible to solve at this time (Salmerón et al. [35]). Consequently, we use heuristic

procedures as illustrated here. We consider a small section of the U.S. grid containing roughly 5,000 buses, 500 generators, 3,000 loads, 5,000 lines, 1,000 transformers, 500 substations, a total reference load of 60 gigawatts (GW), and a total generating capacity of 70 GW.

We posit a group of 10 terrorists: A single terrorist can destroy a line, which takes 48 hours to repair; two terrorists can destroy a transformer or a bus, which takes 168 hours to repair; and three terrorists can destroy a substation, which takes 360 hours to repair. (These repair times are likely to be optimistic and serve for purposes of illustration only.) Three hundred and sixty hours also represents the study's time horizon because the system can be fully repaired in that time.

We employ a load-duration curve (a staircase function) that states: The actual load is 100% of the reference load 20% of the time ("peak load"), 70% of the reference 50% of the time ("standard load"), and 45% of the reference 30% of the time ("valley load"). This load-duration curve implies a total demand for energy, over the course of the study, of about 15,000 gigawatt-hours (GWh). For simplicity, we set all generation costs to \$10 per megawatt-hour (MWh) and set the cost of any unmet demand to \$1,000 per MWh.

VEGA identifies a near-optimal interdiction plan for the terrorists in about 30 minutes on a 3 gigahertz personal computer. The plan interdicts three substations and one line, which results in 356 GWh of energy being shed over the study period, and a peak unmet load of 2.8 GW. These values are small percentagewise, but 2.8 GW represents the requirements of nearly three million residential customers. The economic effects of this attack would be substantial.

From these results, it is clear that protecting substations must be a priority. Therefore, we assume utility companies will spend enough money on increased security at the three hypothetically attacked substations to make them invulnerable to such attacks. We rerun VEGA with this information and find that total unmet demand reduces to less than 160 GWh and peak unmet load decreases to 1.4 GW. Once again, the terrorists attack three substations and one line.

We have reduced the disruption that the 10 terrorists can cause by about 50%, but suppose the defense budget enables us to harden the three substations attacked in the second round, plus one more: We choose one that seems to be important in a model variant that allows 15 terrorists. In the ensuing third round of attacks, the 10 terrorists attack three substations and one line, but this attack results in total unmet demand for energy of only 90 GWh and a peak unmet load of less than 600 GW. Thus, we can substantially reduce the vulnerability of this power grid by improving security at only seven substations, from a total of roughly 500. This may be deemed cost effective by utility planners.

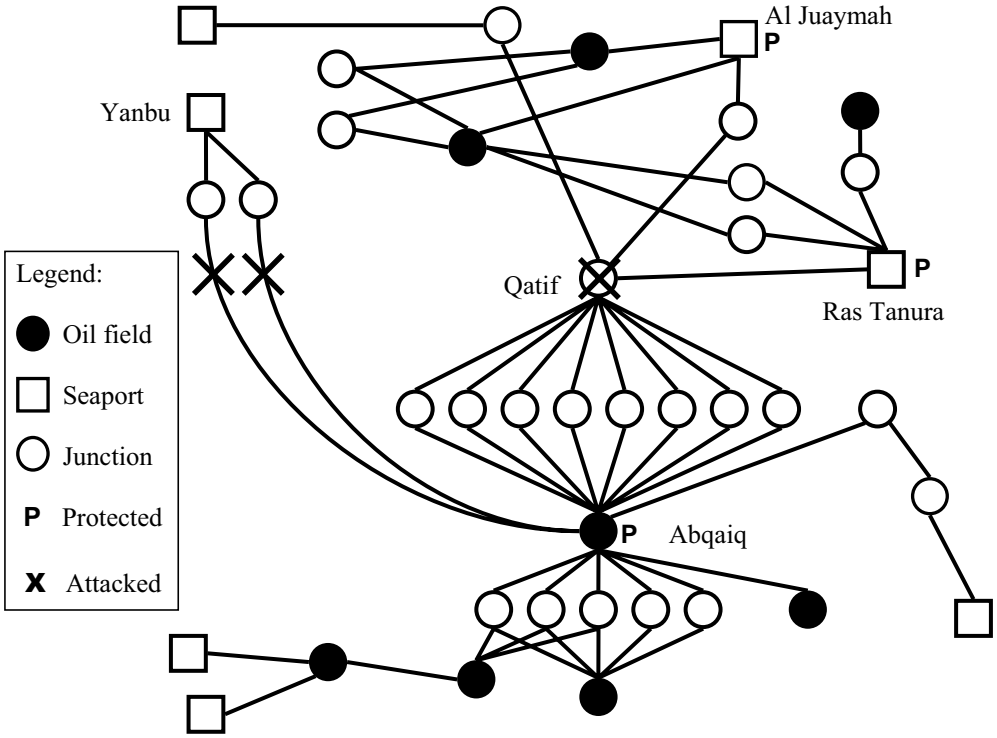
VEGA has been funded, in part, by the U.S. Department of Homeland Security, Office of Domestic Preparedness, and by the Department of Energy. It uses an Intel-based computer, a Microsoft operating system, and modeling software, all of which costs about five thousand dollars per seat.

## Oil Pipelines: An Attacker-Defender Model

Pipeline systems for crude oil and refined petroleum products (and natural gas) are sparsely connected because of the enormous expense required to acquire rights of way, lay pipe, build pumping stations and maintain the system once it is complete. For instance, consider Figure 2, which is a schematic of the crude-oil pipeline network in Saudi Arabia (found, with all capacity data, through a simple Internet search). This network is clearly sparse, although our experience indicates that it is more densely connected than the typical gas or oil pipeline in the United States (e.g., Avery et al. [2]). In fact, the Saudi network may have substantial redundant capacity (Bremmer [6]) and, consequently, may be more resilient to attack than pipeline networks elsewhere.

An enormous security force guards the Saudi pipeline network (Sparshott [38]), but the network covers a huge area that cannot be patrolled completely. Where should the Saudi

FIGURE 2. Three attacks on the Saudi Arabian crude-oil pipeline system reduce capacity by 3.7 mmbbl/day.




*Note.* The Saudi Arabian oil pipeline network has some heavily protected, invulnerable components, indicated by "P," but most of the network is hard to defend and vulnerable to attack. Assuming insurgents have only enough resources to attack three different facilities, the three attacks shown maximally reduce Saudi capacity, even after the pipeline operator optimally redirects flows to use reserve capacity. The reduced output here exceeds a breakpoint estimated to cause a worldwide economic recession (Andrews et al. [1]).

government concentrate its security efforts? A full answer to that question would require a more detailed study than we can undertake. However, one of our red teams has found an "Achilles cutset," so to speak, whose components ought to be considered closely for potential protection.

For purposes of analysis, we play the part of a terrorist organization. First, what is our goal? Well, analysts at Morgan Stanley (Chaney and Berner [14]) report that a reduction in Saudi crude-oil output to 4 mmbbl/day (million barrels per day), from a current 8 to 9 mmbbl/day, would cause worldwide economic distress. The loss would only amount to about 5% of world demand, but Chaney and Berner estimate the price of oil would jump to \$80/bbl from a 2004 price of \$40/bbl. Furthermore, this jump could lead to a global recession if damaged facilities could not be repaired in a few months. So, taking the lead from Morgan Stanley, we set a goal of reducing Saudi oil output to 4 mmbbl/day or less.

Naturally, we would like to implement a coordinated strategy that requires as few individual attacks as possible. What is the minimum number necessary to reach our goal? We assume that the largest oil field at Abqaiq is well protected, i.e., invulnerable to attack, as are the two seaports on the Persian Gulf, Ras Tanamura and Al Juaymah. However, all other system components, pipeline segments, and junctions are potential targets.

We can solve this problem via the max-flow interdiction model of Wood [43], which minimizes maximum flow given a fixed amount of interdiction resource. (Thus, we must solve a min-max attacker-defender model rather than paradigmatic max-min model, AD0.) Each seaport in Figure 2 is connected to a supersink, with the arc's capacity equaling the

port's capacity. Similarly, each oil field is connected by an arc to a supersource, with the arc's capacity equaling the production capacity of the field. Pipeline arcs are assigned their known capacities, and junctions are split into arcs, as required, to represent limited pumping capacity.

The best single attack targets the junction at Qatif. Worldwide oil prices spike on the news, but moderate quickly when it is learned that maximum output has only been reduced to 8.7 mmbbl/day, in a system whose current total capacity is about 10 mmbbl/day, with current output around 9 mmbbl/day. (Exact values for these numbers would depend on when the hypothetical attack occurs. These values are close to current numbers, but Saudi Arabia may add capacity in the near future, and demand could increase or decrease.)

The best attack on two targets adds one of the pipelines connecting Abqaiq and Yanbu, and reduces maximum output to 5.8 mmbbl/day. The world gets really worried. The best attack on three targets adds the second Abqaiq-Yanbu pipeline, Saudi output drops to 3.7 mmbbl/day, our goal has been reached, and worldwide oil prices shoot skyward.

This situation might not last for long—pipelines can usually be repaired fairly quickly—but at the very least, a painful spike in oil prices would result. The three targeted pipeline-system components need security measures reviewed at the very least. At first glance, it seems that a reasonable strategy to mitigate such attacks would add a third Abqaiq-Yanbu pipeline, parallel but not collocated for obvious reasons. However, this pipeline would extend 1,200 kilometers and, estimating from other pipeline construction projects around the world, might cost one billion dollars (Pipeline & Gas Journal [32]). Clearly, other options require exploration.

## The D.C.-Metro System: A Defender-Attacker Model

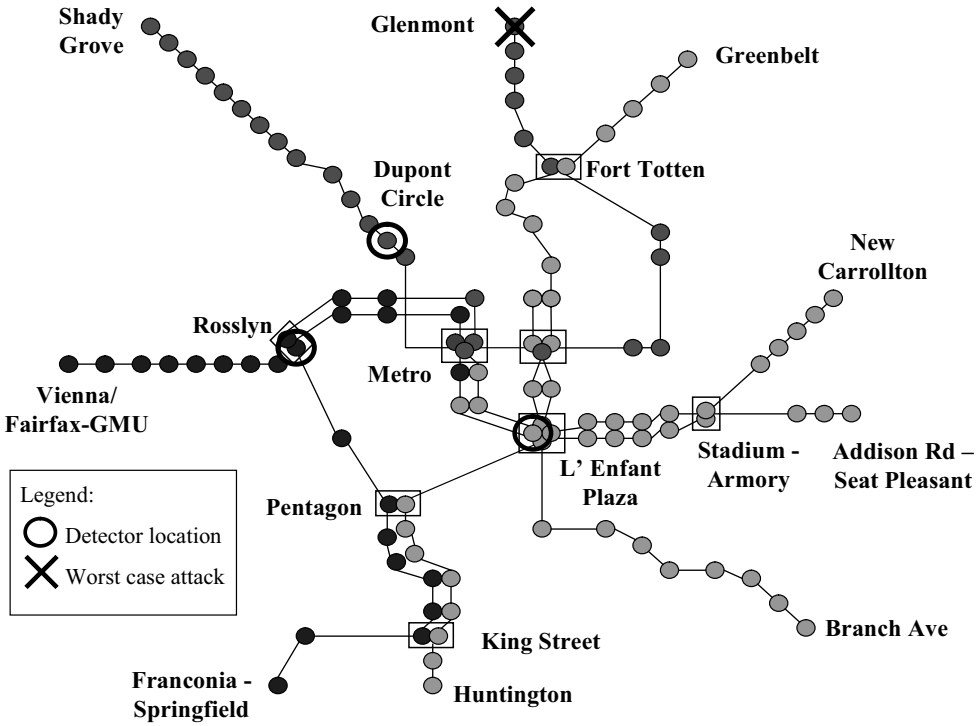
Terrorists have certainly considered the possibility of attacking the United States with nuclear, biological, or chemical (NBC) agents. In likely scenarios, terrorists contaminate a civilian population with a chemical or biological agent, or with radioactive debris from a "dirty bomb." Subway systems in metropolitan areas seem to be attractive targets for this purpose, because their efficiency in moving large numbers of people, quickly, over large distances, would also spread a contaminant among large numbers of people, quickly, over large distances. Consequently, authorities have already begun to install NBC sensors in the Washington, D.C., subway system ("D.C. Metro") and in other transportation facilities around the country (Chang [15]). NBC sensors are expensive, so given limited budgets, how should these detectors be deployed? Figure 3 displays a diagram of the D.C. Metro, and depicts optimal locations given a supply of three sensors. "Optimal" implies that the locations minimize the worst-case time to detection (i.e., no matter where a terrorist might strike). By minimizing detection time, trains could be stopped as quickly as possible after an attack and hazardous-material response teams called in to help reduce casualties. The detection-time objective function only takes transit times and interplatform transfer times into account, but it could certainly account for passenger volumes, if desired.

We will not provide details of this min-max defender-attacker model, but we note that related models have been studied for detecting malevolent contamination of a municipal water system (e.g., Berry et al. [5]).

Figure 4 shows the value of the optimal solution for varying numbers of detectors. This diagram leads to the key insight for policymakers: casualties versus dollars.

Before leaving this topic, we must add a caveat, lest the reader be lulled into a false sense of security. At this stage in the development of NBC detectors, especially biological detectors, noxious substances cannot be quickly and reliably identified. Such detectors may be able to identify a "suspicious" substance instantaneously, but verification may take many hours; sensitivity must be increased and false positives decreased if such technology is to prove useful. A Defense Science Board report states "... in fact, a technological breakthrough is needed" (Defense Science Board [17]).

FIGURE 3. Locations of NBC detectors in the D.C. Metro (subway) system to minimize maximum time to detect an attack.




*Note.* Using public Metro maps and schedules, we model the circulation of an NBC agent throughout the network. The solution installs detectors at Dupont Circle, L'Enfant Plaza, and Rosslyn. Observing this, an optimizing attacker would choose Glenmont to maximize the time to first detection: 31 minutes (Avital et al. [3]).

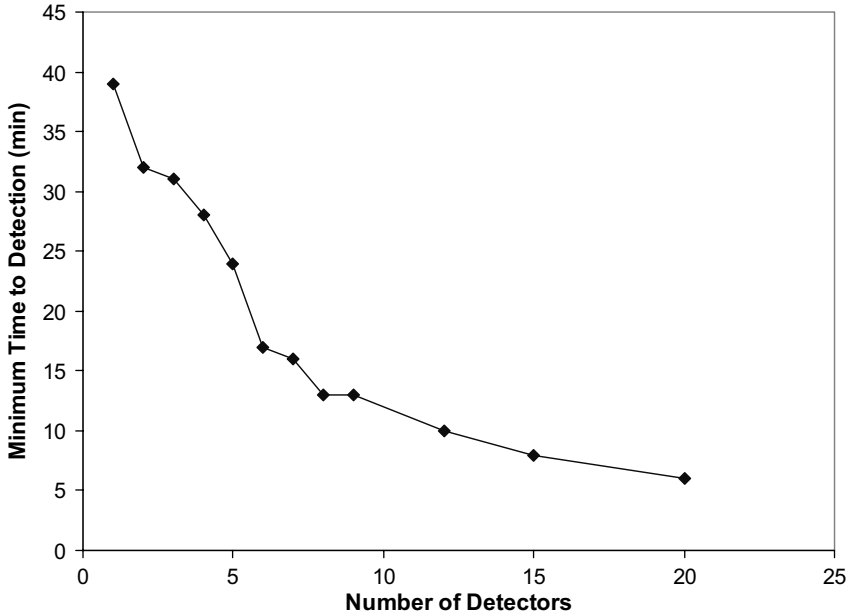
## Improving Airport Security: A Defender-Attacker Model

Airport security has received much attention in recent years, mostly regarding the effectiveness, or ineffectiveness, of personnel and equipment at security checkpoints (Miller [28]). However, the system aspects of airport security deserve the attention of OR analysts. Here, we investigate techniques to improve the probability of detecting a terrorist who is trying to: infiltrate Terminal 1 at the Los Angeles International Airport (LAX); reach an airline gate; and hijack or sabotage an airplane. For simplicity, we consider only a single terrorist, or "infiltrator," who moves along the standard paths that legitimate passengers use.

Figure 5 shows a map of Terminal 1, along with a skeleton of the "infiltration network" that describes the paths that an infiltrator could take from "curbside" into the terminal, through a check-in procedure, through one or more security checkpoints, and finally out to the airline gates. (The full network contains too many arcs to depict.) We shall represent the airport's administration: Our goal is to spend a limited "defense budget" on screening devices and procedures that increase detection probabilities on individual arcs, with the purpose of maximizing the overall detection probability. The options for changing procedures include, for instance, simply closing off certain ingress routes, or performing a physical search of, say, every third passenger, rather than every tenth. In addition to improving standard screening equipment, the red team analyzing this scenario (Landon et al. [26]) also includes the potential installation of advanced imaging devices now undergoing field tests (Levine [27]).

*Probability of nondetection* proves to be a more convenient concept with which to describe a defender-attacker model for this problem. For simplicity, we assume every arc $k$ in the

FIGURE 4. Minimum detection time of an NBC attack on the D.C. Metro as a function of the number of detectors.




*Note.* This is a display for policymakers: The horizontal axis converts to investment of millions of dollars, and the vertical axis converts to the number of people exposed to NBC contaminants, i.e., potential casualties. An analysis like this frequently reveals sharp break points (say, at six detectors here) that may affect policy making (Avital et al. [3]).

network possesses some nominal probability of nondetection, $q_k > 0$. This is the probability that the infiltrator will not be detected if he traverses arc $k$. If we spend exactly $c_k$ dollars at arc $k$, a new device will be installed, or a new procedure implemented, and the nondetection probability becomes $\bar{q}_k > 0$, with $\bar{q}_k < q_k$. (Notes: (i) The model extends easily to handle multiple options for reducing nondetection probability on an arc, (ii) completely closing off a route can be handled by setting $\bar{q}_k$ arbitrarily close to 0, and (iii) every artificial arc $k$ connecting $t$ in $\mathcal{G}$ has $q_k = \bar{q}_k = 1$.) Our overall task is to expend a total budget of $c'$ dollars so as to maximize the minimum probability of nondetection along any path the infiltrator might take. Assuming independence of detection events, this model can be formulated as follows (see the related stochastic-programming model in Pan et al. [31]):

**Indices and Structural Data**

$i \in \mathcal{N}$ nodes of the infiltration network
$k \in \mathcal{A}$ directed arcs of the infiltration network
$\mathcal{G} = (\mathcal{N}, \mathcal{A})$ infiltration network

**Variables**

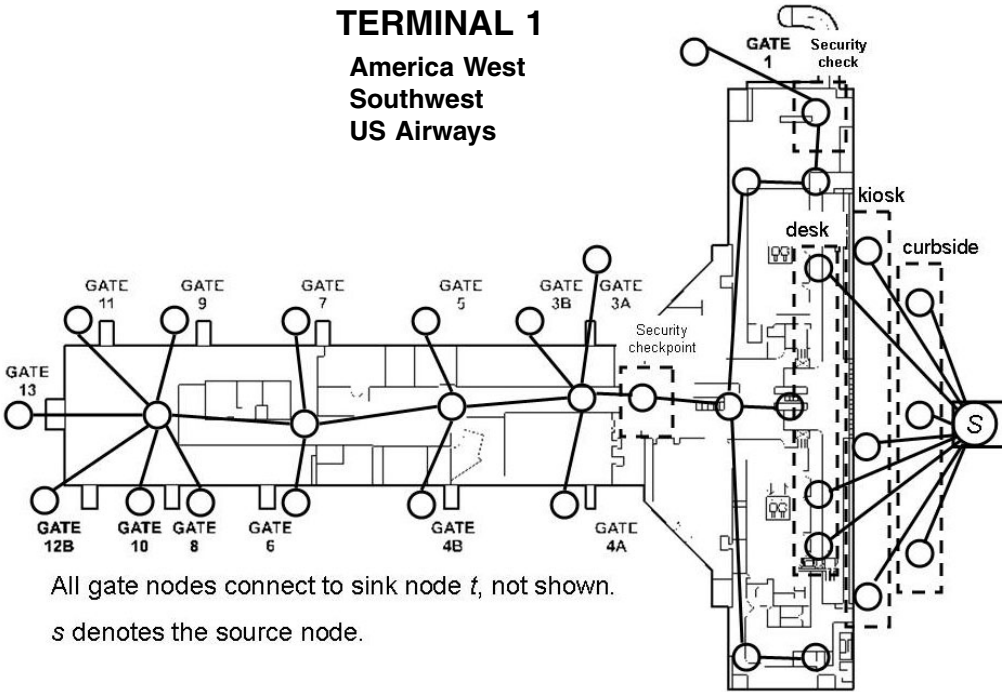
$$x_k = \begin{cases} 1 & \text{if the defender upgrades security on arc } k \\ 0 & \text{otherwise} \end{cases}$$

$$y_k = \begin{cases} 1 & \text{attacker traverses arc } k \text{ when } x_k = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\bar{y}_k = \begin{cases} 1 & \text{attacker traverses arc } k \text{ when } x_k = 1 \\ 0 & \text{otherwise} \end{cases}$$

$\mathbf{x}, \mathbf{y}, \bar{\mathbf{y}}$ vector forms of $x_k, y_k$, and $\bar{y}_k$, respectively

FIGURE 5. A limited security budget can be optimally allocated to protect Los Angeles International Airport (LAX) Terminal 1.




*Note.* This figure displays a map of the terminal along with a skeleton of the "infiltration network" that represents infiltration routes for terrorists (and the routes the legitimate passengers use). Arcs not shown represent movements from check-in desks or automated check-in kiosks to screening stations, through screening stations, through physical-search stations, and also artificial arcs connecting each gate node to a single sink node $t$ (Landon et al. [26].)

**Data**

$A$ node-arc incidence matrix corresponding to $\mathcal{G}$
$\mathbf{b}$ node-length vector with $b_s = 1$, $b_t = -1$ and $b_i = 0$ for all $i \in \mathcal{N} - s - t$
$q_k$ nominal probability of nondetection on arc $k$ when $x_k = 0$ ($q_k > 0$, vector form $\mathbf{q}$)
$\bar{q}_k$ probability of nondetection on arc $k$ when $x_k = 1$ ($q_k > \bar{q}_k > 0$, vector form $\bar{\mathbf{q}}$)
$d_k$ $\ln q_k$ (natural log of $q_k$) (vector form $\mathbf{d}$, matrix form $D = \text{diag}(\mathbf{d})$)
$\bar{d}_k$ $\ln \bar{q}_k$ (vector form $\bar{\mathbf{d}}$, matrix form $D = \text{diag}(\bar{\mathbf{d}})$)
$c_k$ cost, in dollars, to upgrade security on arc $k$ (vector form $\mathbf{c}$)
$c'$ total budget, in dollars, for upgrading security

**Formulation**

$$(\text{DA}_{\text{LAX}}1) \quad \min_{\mathbf{x}\in X} \max_{\mathbf{y},\bar{\mathbf{y}}} \prod_{k\in A} q_k^{(1-x_k)y_k} \bar{q}_k^{x_k \bar{y}_k} \tag{14}$$

$$\text{s.t.} \quad A\mathbf{y} + A\bar{\mathbf{y}} = \mathbf{b} \tag{15}$$

$$\mathbf{y}, \bar{\mathbf{y}} \in \{0,1\}^{|\mathcal{A}|} \tag{16}$$

where $X = \{\mathbf{x} \in \{0,1\}^{|\mathcal{A}|} \mid \mathbf{cx} \le c'\}$.

Constraints (15) and (16) ensure that one unit of "unsplittable flow," representing the infiltrator, moves from $s$ to $t$. Constraints (15) are standard flow-balance constraints, just like those one could use to model a shortest-path problem in $\mathcal{G}' = (\mathcal{N}, \mathcal{A} \cup \mathcal{A})$, which is simply $\mathcal{G}$ with each arc duplicated.

The standard reformulation technique for this model takes a logarithm of the objective function, say the natural logarithm. This leads to the essentially equivalent model, $\text{D2}_{\text{LAX}}$,

below. It is clear then that we can replace constraints (16) with simple nonnegativity restrictions, because the constraint matrix (15) is totally unimodular, and for fixed $\mathbf{x}$, the model defines a straightforward shortest-path problem on $\mathcal{G}'$ (if we multiply the nonpositive objective function by $-1$, and switch the inner maximization to a minimization). Note also that the infiltrator's objective can only worsen by putting flow around a cycle, so no difficulty analogous to negative-length cycles in a shortest-path problem can arise here.

$$\begin{aligned}(\mathrm{DA2_{LAX}}) \qquad & \min_{\mathbf{x}\in X} \max_{\mathbf{y},\bar{\mathbf{y}}} (\mathbf{1}+\mathbf{x})^T D\mathbf{y} + \mathbf{x}^T \bar{D}\bar{\mathbf{y}} \\ & \text{s.t.} \quad A\mathbf{y} + A\bar{\mathbf{y}} = \mathbf{b} \\ & \qquad\qquad \mathbf{y}, \bar{\mathbf{y}} \geq \mathbf{0}\end{aligned}$$

Clearly, this model converts easily into an MILP.

Before reporting computational results, we note that modifying security equipment and procedures can both increase delays for legitimate passengers, or decrease them. For instance, increasing the percentage of people receiving physical searches on an arc will certainly increase the detection probability for an infiltrator traversing that arc, but it will also raise the average passenger's delay there. On the other hand, adding parallel metal detectors, parallel imaging devices, and parallel personnel to oversee this equipment will reduce average delays. $\mathrm{DA2_{LAX}}$ can be modified to incorporate constraints that limit, at least approximately, the average delay for a legitimate passenger. However, for simplicity, we simply report the changes in delay that result from changes in security procedures and equipment, under the pessimistic assumption that passengers do not adjust their routes to reduce delay for themselves.

For obvious reasons, our red team can only make educated guesses about the cost of, and improved detection probabilities for, these devices. The team must also make similar guesses regarding the delay that new imaging devices will cause passengers. Therefore, the absolute statistics reported by the team cannot be taken literally. However, the relative results are believable, and the methodology can accept any system-describing parameters, which field testers and manufacturers should eventually be able to provide. We summarize the red team's computation results below. Note that "Risk" reflects probability of detection only as a relative value, and the expenditures are probably optimistic and the delay values are probably pessimistic:

1. Baseline, Scenario 1, no security improvements: Budget = \$0, "Risk" (to the infiltrator) = 10, Delay (incremental) = 0 hours, Actions = {}.

2. Scenario 2: Budget = \$100,000, Risk = 126, Average Delay = 1.5 hours, Actions = {Add two imaging devices, screen 1 in 10 at two locations, close three check-in kiosks}. (Note: Closing an automated kiosk increases the reliability of identification checks.)

3. Scenario 3: Budget = \$250,000, Risk = 249, Average Delay = 2.5 hours, Actions = {Add 15 imaging devices, screen 1 in 3 at all locations security checkpoints, close three check-in kiosks}.

## Supply Chains

Supply chains, i.e., physical distribution systems, are a key infrastructure of private-sector companies that manufacture and/or distribute goods. "Supply chains" do not appear on the list of critical infrastructure systems shown in Table 1, but they are certainly critical to our nation's well-being.

Strategic supply-chain design has a long and successful record in the United States, reducing costs and increasing service levels. Unfortunately, efficient supply chains are fragile. In fact, after scrupulously investing exactly the right amount of money in a supply chain, on exactly the right bottlenecks, the resulting product-flow patterns resemble one or more spanning trees. However, as any OR analyst knows, a spanning tree is maximally fragile: Breaking any link disconnects the network.

To address supply-chain vulnerability, we have teamed with Prof. Terry Harrison of Pennsylvania State University and Dr. Jeffrey Karrenbauer, President of INSIGHT, Inc., a company devoted to supply-chain optimization for over 25 years (INSIGHT [24]). Together, we have analyzed detailed corporate supply-chain data for many companies, including the majority of the FORTUNE 50. Also, we have developed new features for INSIGHT's supply-chain optimization tools to evaluate and mitigate supply-chain vulnerability. Many key results have already been presented by Brown et al. [11, 12, 13], so we provide only an overview here.

The first key "result" is an observation: We still encounter considerable confusion in the private sector between random acts of nature—these have been studied by insurance actuaries for centuries—and belligerent acts of intelligent terrorists who observe defensive preparations and act to maximize damage. We strongly suggest remedying this confusion before proceeding with any analysis.

On occasion, one can reduce vulnerability substantially with simple planning and with only a modest investment in new physical infrastructure. Sometimes, just strategically relocating surge capacity can provide benefit at virtually no cost. This contrasts with the high cost of adding redundant capacity, or hardening components, in other types of infrastructure such as pipelines and electric power grids.

We have learned that labor unions and competitors can be just as clever and determined as terrorists, and have similar goals: maximize damage inflicted (to market share, profit, reputation, etc.). The denial of access to West Coast ports in the United States in 2002 due to a labor dispute was no less damaging than the anthrax attacks of 2001 that closed postal and shipping services on the East Coast.

We have presented our findings to numerous companies, with enthusiastic responses to even simple discoveries. American companies now have senior executives focused on "preserving corporate continuity." These positions were originally motivated by threats to information systems, and thus back-up computer facilities and doubly backed-up data have become ubiquitous. Now, these same companies are coming to realize that they must also back up their physical operations to handle attacks on their own infrastructure (e.g., equipment, warehouses) as well as on public infrastructure they use (e.g., roads, communications networks).

## Other Systems

Our work on critical infrastructure protection represents just one aspect of a research program that has also led to new military and diplomatic planning models; two have already been incorporated into comprehensive decision-support systems. One system helps plan theater ballistic-missile defense (Brown et al. [9]). The embedded defender-attacker model optimally locates antimissile platforms (ships or ground-based units supplied with antimissile missiles), while assuming the attacker can see some or all of our defensive preparations. The other system identifies optimal actions (e.g., embargoes of key materials, economic sanctions, military strikes) to delay a covert nuclear weapons program (Brown et al. [10]). This is an attacker-defender model where we, for a change, are the attacker. As with the missile-defense model, analysis can be carried out under different assumptions regarding the adversary's (defender's) ability to observe our actions. This model applies to any complex industrial project that can be delayed by a competitor.

A key insight from these military and diplomatic exercises is that deception and secrecy can make huge contributions to successful defense of our critical assets, or to successful attacks on an adversary's critical assets. (The techniques of two-person game theory can also be useful here; for example, see Owen [33].) Secrecy is already becoming an important (and debated) issue in the general area of homeland security.

Although this work is all relatively new, there is already an emerging body of unclassified publications including about fifty case studies, over twenty graduate theses, open-literature

Table 3. Case studies of these topics have evaluated roles of both attacker and defender.

| | |
|---|---|
| Electric grids | Weapons of mass destruction (WMDs) at |
| Road networks | Reliant Stadium, Houston, Texas |
| Strategic rail networks | Washington, D.C., Metro |
| Domestic water systems | Melbourne, Australia |
| Sea lanes, canals, restricted straits | Changi Naval Base, Singapore |
| Multicommodity supply chains | Manhattan |
| Petroleum distribution networks for | Norfolk, Virginia |
| U.S. Southwest | Insurgent incursions |
| Northern California | Economic warfare |
| Defense Fuel Supply System, Japan | WMD development project |
| Theater ballistic missile attacks | |

publications, and a number of prototypic decision-support tools. Table 3 shows some of the topics studied. We are working with certain institutions that address these threats and welcome inquiries from others. We also provide classified products to planners when the need arises.

## What We Have Learned

We have discovered much through our own mathematical modeling of critical-infrastructure protection, and from applications and red-team studies. We have also learned from reading the literature, attending conferences, and speaking with colleagues, clients, and students. This section summarizes the lessons we have gleaned from all these sources.

*The attacker has the advantage.* This is the reverse of classical military theory and accrues from the asymmetric nature of this conflict: The defender must protect a huge, dispersed target set, while the attacker need only focus on a small set of targets chosen to maximize damage.

*Some systems are naturally robust, while others are not.* It turns out that our road systems are remarkably robust, fuel-distribution systems are highly fragile, and most other systems lie somewhere in between.

*Hardening an infrastructure system from attack can be expensive.* However, if you understand what the most damaging attacks must look like, you can better improve the system's robustness against attack for a given budget.

Critical infrastructure has been built to be "cost-effective" with little concern for coordinated, belligerent attacks. Consequently, these systems are fragile with respect to such attacks, and even four years after September 11th, private owners of infrastructure have few economic incentives to spend large sums of money to reduce this fragility. This calls for (i) government subsidies, changes to tax codes, and regulatory reform, and/or (ii) proving the secondary economic benefit of these expenditures, if such exist (for example, spare electric transmission capacity could provide new, profitable trading opportunities).

*The data are out there, and if we can get them, anybody can.* "Sunshine laws" in the United States require that our governments, federal to local, conduct their affairs with transparency to the public. As a result government agencies have produced lots of excellent websites with lots of useful information for terrorists based anywhere in the world. Many websites have been redesigned in recent years to reduce access to potentially dangerous information, but we find stunning exceptions. We advise owners of public websites associated with infrastructure to appoint an independent "red team" to analyze the website with intent to cause harm to the owners or to the users of the infrastructure.

*The answers are not always obvious.* The most damaging coordinated attacks, or the most effective defenses, can be nonintuitive. Key U.S. infrastructure systems are huge, and analysis at large scale deserves rigorous, purpose-built, optimizing decision-support tools to

formalize the notion of a transparent, two-sided conflict. Heuristics have their place here but, preferably, not for identifying worse-case attacks. What is an "approximately worse-case attack"?

*Malicious, coordinated attacks can be much more damaging than random acts of nature.* Our audiences usually arrive with the opposite point of view. However a, skillful, small-scale attack can inflict more damage than a major hurricane or earthquake.

*Reliability is not the answer.* We must protect the most *critical* components in our infrastructure systems, rather than backing up the least reliable components. Many infrastructure owners still think that a "reliable system," i.e., a system that fails rarely due to random events, will be a "robust system" in the face of malicious, coordinated attacks. However, common sense (for a terrorist) dictates: Destroy the most reliable components. After all, they have been made most reliable because they are most critical for system operations.

*The right redundancy may be the answer.* For any given level of investment, there is usually a dominant set of incremental changes to infrastructure that returns maximal immediate benefit. For some types of infrastructure—e.g., supply chains—benefit can be achieved at relatively modest cost by adding a few alternate shipment paths or by installing excess capacity at just the right locations, etc.

*Secrecy and deception may be valuable.* Our military applications of attacker-defender and defender-attacker models have shown that much can be gained from secrecy and/or deception. For instance, hiding the location of a defensive asset can cause the attacker to strike a target that is essentially invulnerable. Clearly, in the world of suicide terrorists and physical infrastructure, such an outcome could be desirable.

*Worst-case analysis using optimization is key to a credible assessment of infrastructure vulnerability, and to reducing that vulnerability.* We cannot depend on standard reliability analyses to protect us adequately because we cannot assume that attacks occur randomly. We face a determined, intelligent enemy who seeks to do us maximal harm.

## Acknowledgments

Salmerón and Wood thank the U.S. Department of Homeland Security and the U.S. Department of Energy for research support. Brown and Wood are grateful for sustaining research support from the Office of Naval Research (Division of Mathematical Sciences), the Air Force Office of Scientific Research (Optimization and Discrete Mathematics Program), and the Joint Warfare Analysis Center. Brown, Carlyle, and Wood also thank the National Security Agency for support. At various times, the authors have also worked with and received support from every U.S. uniformed military service, and they thank every one.

In late 2001, the authors approached INSIGHT, Inc., to help discover what private companies could do to fortify their operations against hostile threats. (Note: Brown and Wood have worked with INSIGHT on private-sector business optimization problems for decades.) INSIGHT has granted unfettered use of its supply-chain design software, devoted extensive development effort, provided data from a host of private-sector clients (scrubbed of proprietary identification and confidential data), and arranged direct access to its clients. The authors are grateful for INSIGHT's assistance.

## References

[1] C. Andrews, K. Cason, A. Godfrey, and M. Revor. Optimal defense of Saudi Arabia's pipelines against terrorist attack. Red Team Report, OA4202 Network Flows and Graphs, Naval Postgraduate School, Monterey, CA, 2003.

[2] W. Avery, G. G. Brown, J. Rosenkranz, and R. K. Wood. Optimization of purchase, storage and transmission contracts for natural gas utilities. *Operations Research* 40:446–462, 1992.

[3] I. Avital, K. Cormican, and K. Wood. Where to install contamination detectors in a subway system. Working paper, 2005.

[4] J. F. Benders. Partioning procedures for solving mixed integer variables programming problems. *Numerische Mathematik* 4:238–252, 1962.

[5] J. Berry, L. K. Fleischer, W. E. Hart, and C. A. Phillips. Sensor placement in municipal water networks. *World Water and Environmental Resources Congress*, SAND2003-0918C, Philadelphia, PA (June 23–26) 2004.

[6] I. Bremmer. The Saudi paradox. *World Policy Journal* 21(3):23–30, 2004.

[7] E. Brown, M. Hakola, M. Raffetto, and T. Yanik. Effects of terrorist attacks at U.S. airports. Red Team Report, OA4202 Network Flows and Graphs, Naval Postgraduate School, Monterey, CA, 2003.

[8] G. G. Brown, R. F. Dell, and R. K. Wood. Optimization and persistence. *Interfaces* 27:15–37, 1997.

[9] G. Brown, M. Carlyle, D. Diehl, J. Kline, and K. Wood. A two-sided optimization for theater ballistic missile defense. *Operations Research* 53(5), 2005.

[10] G. Brown, M. Carlyle, R. Harney, E. Skroch, and K. Wood. Interdicting a nuclear weapons project. Working paper, 2005.

[11] G. Brown, M. Carlyle, T. Harrison, J. Salmerón, and K. Wood. How to attack a linear program. Plenary address, Military Operations Research Society, Quantico, VA (June 10) 2003a.

[12] G. Brown, M. Carlyle, T. Harrison, J. Salmerón, and K. Wood. Tutorial: How to build a robust supply chain or harden the one you have. *INFORMS Annual Meeting*, Atlanta, GA (October 19), 2003b.

[13] G. Brown, M. Carlyle, T. Harrison, J. Salmerón, and K. Wood. Designing robust supply chains and hardening the ones you have. *INFORMS Conference on OR/MS Practice*, Cambridge, MA (April 26–27), 2004.

[14] E. Chaney and R. Berner. Global: oil price update: Still higher and more uncertain. *Global Economic Forum*, Morgan Stanley (May 4) `http://www.morganstanley.com/GEFdata/digests/20040507-fri.html`, 2004.

[15] K. Chang. Ideal sensors for terror attack don't exist yet. *New York Times* (April 1):F1, 2003.

[16] K. J. Cormican, D. P. Morton, and R. K. Wood. Stochastic network interdiction. *Operations Research* 46:184–197, 1998.

[17] Defense Science Board. Defense Science Board 2003 summer study on DoD roles and missions in homeland security. *Supporting Roles, Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics*, Vol. II-A. Washington, D.C. (May) 2004.

[18] Department of Homeland Security (DHS). National strategy for homeland security. `http://www.whitehouse.gov/homeland/book/`, 2002.

[19] Department of Justice. Al Qaeda training manual. `http://www.usdoj.gov/ag/trainingmanual.htm`, 2004.

[20] Department of the Army. ADA employment principles, guidelines, and priorities, Appendix A. *Army Field Manual FM 3-01.11.* `http://www.globalsecurity.org/military/library/policy/army/fm/`, 2000a.

[21] Department of the Army. Fundamentals of army air and missile defense operations, Ch. 4. *Army Field Manual FM 44-100.* `http://www.globalsecurity.org/military/library/policy/army/fm/`, 2000b.

[22] M. L. Garcia. *The Design and Evaluation of Physical Protection Systems.* Butterworth-Heinemann, 2001.

[23] B. Golden. A problem in network interdiction. *Naval Research Logistics Quarterly* 25:711–713, 1978.

[24] INSIGHT. Strategic analysis of integrated logistics systems (SAILS). Manassas, VA. `http://www.insight-mss.com`, 2004.

[25] E. Israeli and K. Wood. Shortest-path network interdiction. *Networks* 40:97–111, 2002.

[26] C. Landon, K. Takagi, and K. Watts. Protection of restricted areas from infiltration. Red Team Report, OA4202 Network Flows and Graphs, Naval Postgraduate School, Monterey, CA (November) 2004.

[27] S. Levine. Software and Smart Card. *U.S. News & World Report* (March 14) `http://www.usnews.com/usnews/news/articles/050314/14airport.htm`, 2005.

[28] L. Miller. Airport security hasn't improved. *Associated Press* (April 16), 2005.

[29] J. T. Moore and J. F. Bard. The mixed integer linear bilevel programming problem. *Operations Research* 38:911–921, 1990.

[30] NORTHCOM. `http://www.northcom.mil/`, 2004.

[31] F. Pan, W. S. Charlton, and D. P. Morton. A stochastic program for interdicting smuggled nuclear material. D. L. Woodruff, ed. *Network Interdiction and Stochastic Integer Programming.* Kluwer Academic Publishers, Boston, MA, 1–20, 2003.

[32] *Pipeline & Gas Journal*. Pipeline construction scorecard. (January):82–85, 2005.

[33] G. Owen. *Game Theory*, 3rd ed. Academic Press, San Diego, CA, 2001.

[34] N. H. Roberts, W. E. Vesely, D. F. Haasl, and F. F. Goldberg. *Fault Tree Handbook.* NUREG-0492, U.S. Nuclear Regulatory Commission, Washington, D.C., 1981.

[35] J. Salmerón, R. K. Wood, and R. Baldick. Optimizing an electric grid under asymmetric threat (II). Technical Report NPS-OR-04-001, Naval Postgraduate School, Monterey, CA. `http://library.nps.navy.mil/uhtbin/hyperion/NPS-OR-04-001.pdf`, 2004a.

[36] J. Salmerón, K. Wood, and R. Baldick. Analysis of electric grid security under terrorist threat *IEEE Transactions on Power Systems* 19-2:905–912, 2004b.

[37] J. Salmerón, R. K. Wood, and R. Baldick. VEGA. `http://www.nps.navy.mil/orfacpag/resumePages/projects/VEGA.htm`, 2005.

[38] J. Sparshott. Saudi's security for oil extensive. *The Washington Times* (June 22) 2004.

[39] U.S.-Canada Power System Outage Task Force. Causes of the August 14th blackout in the United States and Canada. Interim report, `http://www.nerc.com/~filez/blackout.html`, 2003.

[40] H. von Stackelberg. *The Theory of the Market Economy.* William Hodge & Co., London, UK, 1952.

[41] B. Wallace. Three-year term for Army vet who fired on power plant. *San Francisco Chronicle* (April 24):A21, 1998.

[42] *The Washington Post*. Sniper shootings. `http://www.washingtonpost.com/wp-srv/metro/daily/oct02/snipershootings.htm`, 2002.

[43] R. K. Wood. Deterministic network interdiction. *Mathematical and Computer Modelling* 17:1–18, 1993.

[44] A. J. Wood and B. F. Wollenberg. *Power Generation, Operation and Control*, 2nd ed. John Wiley and Sons, New York, 1996.

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS | ISBN 1-877640-21-2
DOI 10.1287/educ.1053.0014

# Demand Selection and Assignment Problems in Supply Chain Planning

***Joseph Geunes, Yasemin Merzifonluoğlu, H. Edwin Romeijn***
Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida 32611 {geunes@ise.ufl.edu, yasemin@ufl.edu, romeijn@ise.ufl.edu}

***Kevin Taaffe***
Department of Industrial Engineering, Clemson University, Clemson, South Carolina 29634, taaffe@clemson.edu

**Abstract** Effective demand planning has recently become recognized as an important source of competitive advantage from both marketing and operations perspectives. The operations literature has often focused on pricing (revenue management) and information-based approaches (e.g., improved forecasting, mechanisms to elicit advance demand information) for integrated demand and production management. We can think of pricing, for example, as an implicit demand selection mechanism that determines the scope and scale of demands a supplier will ultimately serve, and that therefore drives the production requirements (as well as costs and revenues) that the supplier will face. In contrast, a variety of planning and design contexts involve a supplier or supply chain stage explicitly selecting a subset of demands from a collection of potential downstream demand sources. Such decisions on the demands a supply chain stage will serve may arise out of necessity (due to supply capacity limits), may be based strictly on economic considerations, or may be a component of the (assignment) decisions in a larger supply chain network design problem. Incorporating such dimensions of demand selection flexibility within supply chain planning models allows a supplier to best match available resources with downstream requirements, which can create opportunities for enhancing profitability. This chapter discusses a class of optimization models for addressing varying degrees of demand selection flexibility in integrated production and demand planning.

**Keywords** production planning; demand management; revenue management; economic lot sizing; newsvendor problem

## 1. Introduction

Recent trends in supply chain management recognize the importance of effective demand management for suppliers. Research on demand management in the operations literature has broadly focused on the mechanisms a supplier can use to influence demand, and how a supplier can best utilize these mechanisms to provide a good match between capacity and demand. In this chapter, we discuss a class of integrated operations planning and demand management problems, for which we provide corresponding planning models. This class of models addresses questions regarding the best levels of demand for a supplier. We begin with single-stage planning problems where the stage has varying degrees of flexibility in determining the set of demands to which it will respond. In addition, we consider multifacility problems that determine the best allocation (or assignment) of downstream demands to upstream facilities. As we will see, the single-stage planning models, while interesting in their own right as combined demand and operations planning models, also serve as subproblems in a decomposition strategy applied to the multifacility problems.

Recent demand management literature in operations discusses mechanisms for affecting both capacity and demand to increase profit. Crandall and Markland [6] studied service industry firms and classified several demand management approaches, including capacity management and demand-influencing strategies. Moodie [24] examined pricing and lead-time negotiation strategies for maximizing long-run net revenue when customers are time and price sensitive and capacity is fixed. Keskinocak et al. [18] considered models for coordinating scheduling and lead-time quotation. In these models, revenues from customers are sensitive to lead times, and the scheduling problem therefore contains a demand management dimension. Iyer et al. [17] use postponement (and an associated customer reimbursement) as a mechanism for managing demand surges under limited capacity. These demand management problems exploit knowledge of customer delivery-timing flexibility to effectively match supply and demand, but do not consider economies of scale often associated with operations costs. Lee et al. [20], on the other hand, consider the benefits of customer delivery-timing flexibility within a multiperiod production-planning context with fixed production setup costs.

Integrated operations and demand-planning problems often draw from two related streams of literature involving yield (or revenue) management and combined inventory planning and pricing models. Yield management is often associated with perishable goods such as airline seats or hotel stays (see, for example, Weatherford and Bodily [36] for a thorough discussion of this line of research), while integrated pricing and inventory planning models often consider influencing uncertain demand for durable physical goods. Gallego and van Ryzin [9] and Petruzzi and Dada [26] provide excellent examples of research in this area, as well as discussions of the broader related literature. In this problem class, pricing decisions serve as a mechanism for effectively setting the best demand levels over time.

Although much of the integrated pricing and inventory literature focuses on using pricing to influence uncertain demand, pricing decisions have also been considered in make-to-order and other effectively deterministic contexts. Here the emphasis lies in managing economies of scale in production and/or capacity in conjunction with demand. Thomas [33] considered the classical uncapacitated economic lot-sizing (UELS) problem with price-sensitive demands and proposed a dynamic programming approach for setting prices in every period. For the problem of setting a single price over the entire horizon in UELS, Kunreuther and Schrage [19] proposed a heuristic method, while Gilbert [13] subsequently provided an exact algorithm. More recently, van den Heuvel and Wagelmans [34] provided a new exact algorithm that applies to a more general set of cost and demand parameters. We also note that Biller et al. [3] incorporated pricing decisions within a capacitated lot-sizing problem, but without fixed production setup costs.

While pricing serves as a key mechanism for influencing demand, the single-stage models discussed in this chapter consider a more direct way of influencing demand through explicit demand selection decisions. That is, these models assume that a set of candidate demands exists from which the supplier is free to select. Such demand selection decisions may be practically relevant in several contexts. For example, if capacity limits exist at a production facility, and total demand outstrips this capacity, a supplier may be forced to turn away some subset of customer demands. As another example, suppose that a firm wishes to determine a set of initial markets to penetrate in the roll-out of a new product, and demand characteristics vary among markets. In both of these examples, the supplier selectively determines some subset of demands it will satisfy to maximize its net profit after operations costs.

Past literature on models that directly select demands from a candidate set is limited, with a few notable exceptions. Charnsirisakskul et al. [5] considered a (deterministic) capacitated order selection model in which customers express acceptable delivery lead times, and the supplier incurs a tardiness cost for late deliveries. They focused on the positive impact of increased customer lead-time flexibility on profits in a capacitated setting without economies of scale in production costs (all of the deterministic models we will consider contain a

fixed-charge or general concave production cost structure, resulting in production economies of scale). In a stochastic setting, Carr and Lovejoy [4] proposed an inverse newsvendor model, in which the capacity to satisfy demand is a random variable with known mean and variance. The supplier has available multiple prioritized customer classes, each consisting of a set of customer clusters, and must determine what fraction of each customer cluster within each class it will pursue. The objective is to determine a demand distribution from the opportunity set of available distributions that maximizes expected profit. In the stochastic version of the model that we will consider, capacity is a decision variable, and customer demands must be completely accepted or rejected.

Demand selection serves as the common thread that links the single-stage models we will discuss in this chapter. These models generalize several classical production and inventory planning models to incorporate demand selection. In certain cases, where appropriate, we illustrate a very natural analogous pricing model corresponding to the demand selection version of the model. In addition to the selection dimension of demand management, several of our models consider elements of customer delivery-timing flexibility. We also discuss a set of models in which production is capacity constrained. Such models can serve as valuable benchmarking tools when a supplier has a good understanding of the costs of demand fulfillment as well as customer delivery-timing flexibility.

This chapter brings together a number of recent works with a demand selection theme. The broad scope of problems we will discuss, and space limitations, preclude providing the complete set of our results for all of the models we discuss. We will therefore provide the primary results for those problems we discuss, and cite the sources containing detailed results where appropriate. The remainder of this chapter is organized as follows. Section 2 discusses the generalization of the classical economic order quantity (EOQ) and newsvendor problems to allow for demand selection flexibility. Somewhat surprisingly, the resulting optimization problems are structurally identical, but lead to different interpretations. Section 3 then considers classical dynamic, deterministic production planning models that allow for demand selection. We briefly discuss interesting results on capacity-constrained versions of the problem, including the case in which capacitated overtime and subcontracting options are available. Section 4 provides a framework for multifacility customer assignment problems, for which the models in §§2 and 3 can serve as subproblems. Section 5 concludes with a summary and possible directions for future research.

## 2. Static Single-Stage Models with Demand Selection

We first study, in §2.1, a generalization of the well-known EOQ model (Harris [15]). In the presence of demand selection flexibility, instead of having a fixed demand rate $\lambda$, we have a choice of different markets we can serve, each with its own demand rate and net revenue. Then, in §2.2, we consider a single-stage problem with stochastic market demands in a single-period context, which we call the *selective newsvendor problem*, or SNP. For more details on the models discussed in this section, please see Geunes et al. [12] and Taaffe et al. [31].

### 2.1. EOQ Model with Market Selection

**2.1.1. Basic Model.** We begin with the base assumptions of the EOQ model, with a single commodity produced at a single stage, a fixed ordering cost of $K$, a per-unit order cost of $c$, and a per-unit holding cost of $h$ per unit time (see Nahmias [25] or Silver et al. [29] for a full discussion of the EOQ model). A set of potential markets $I$ exists (indexed by $i$), and the producer can choose to supply any subset of markets. Market $i$ provides a net revenue per unit of $r_i$ (net of any variable delivery costs), where we assume without loss of generality that $r_i > 0$ for all $i \in I$. Market $i$ has a deterministic demand rate equal to $\lambda_i$ (units per unit time). We preindex markets in *decreasing net revenue* (DNR) order,

which implies that $r_i > r_j$ for $i > j$ (where we assume without loss of generality that no two markets have identical net revenue values).

Letting $y_i$ denote a binary decision variable equal to 1 if we satisfy the demand of market $i$, and 0 otherwise, the demand rate the supplier faces equals $\sum_{i \in I} \lambda_i y_i \equiv \lambda^\top y$ (where $\lambda$ and $y$ are the vectors of $\lambda_i$ and $y_i$ variables). Thus, the optimal time between orders, $T^*(y)$, is equal to $\sqrt{2K/h\lambda^\top y}$. Substituting this into the minimum average (per unit time) setup and holding-cost equation for the EOQ model results in $\sqrt{2Kh\lambda^\top y}$. The market-selection version of the EOQ problem maximizes net revenue less average holding and ordering costs per unit time, based on order selection decisions, which leads to the following optimization problem.

$$[\textbf{EOQMC}] \qquad \text{maximize} \quad \sum_{i \in I} r_i \lambda_i y_i - \sqrt{2Kh \sum_{i \in I} \lambda_i y_i}$$

$$\text{subject to:} \quad y_i \in \{0, 1\} \quad \text{for all } i \in I.$$

Shen et al. [28] showed that problems containing this structure can be solved in $\mathcal{O}(n \log n)$ time (where $n = |I|$) using the following property. If we sort items in nonincreasing order of the ratio of the coefficient of $y_i$ in the linear term to the coefficient of $y_i$ in the square root term, then if $j > i$ and an optimal solution selects item $j$, an optimal solution exists that selects item $i$. This indexing approach applied to our EOQMC problem results in the DNR index ordering that we used to preindex the markets in this problem. Therefore, after sorting items in this order, we simply need to evaluate the cost of $n + 1$ ordered solutions containing markets $1, \ldots, j$, for $j = 0, \ldots, n$ (with $j = 0$ corresponding to selecting no markets at all). Interestingly, the attractiveness of a market in this context is purely determined by the per-unit net revenue value and is independent of the market demand level. Geunes et al. [12] exercise this model to provide expressions useful in managerial decisions, such as the minimum revenue and demand levels required to enter a new market. They also show that the DNR indexing approach continues to hold under a finite production rate and under market-specific holding-cost parameters, after a holding-cost-adjusted revenue index ordering is applied.

**2.1.2. Capacity Constraints.** In addition to using a finite production rate to reflect capacity limits, a supplier might also face limits on its total output per unit time, or on the batch sizes it can produce in a production run (due to, for example, storage space limitations). When total output per unit time is limited by some bound $B$, we must add the following constraint to the [EOQMC] formulation.

$$\sum_{i \in I} \lambda_i y_i \leq B \tag{1}$$

With this additional constraint, the resulting problem becomes a nonseparable, nonlinear, and convex 0–1 knapsack problem. Geunes et al. [12] provide a polynomial-time algorithm for solving the linear relaxation of this problem, using a combination of the DNR index-ordering algorithm and a simple continuous knapsack problem solution. They also provide an asymptotically optimal algorithm (in the number of markets, for a problem class in which the capacity grows linearly in the number of markets) for solving the 0–1 version of the problem. These results continue to hold under a finite production rate assumption.

When a limit of $B_L$ exists on the maximum lot size, the problem becomes slightly trickier. We consider the following constraint:

$$Q \leq B_L.$$

Let $I^s$ denote a set of selected markets (i.e., those for which $y_i = 1$), and let $\lambda^s$ denote the corresponding demand rate (i.e., $\lambda^s = \sum_{i \in I^s} \lambda_i$). Because the average cost per unit time is

convex in $Q$ for a given set of markets, if $Q^*(I^s) = \sqrt{2K\lambda^s/h} \leq B_L$ holds, then this is the optimal order quantity for the set of markets $I^s$. If $Q^*(I^s) \geq B_L$, then setting $Q(I^s) = B_L$ provides the best order quantity for the set $I^s$.

We approach this problem by considering two separate subproblems: one that admits values of $y$ such that the corresponding EOQ value is feasible ($\leq B_L$; we will call these EOQ-feasible solutions), and one that admits all values of $y$ but requires a lot size of $Q = B_L$. In the first subproblem, we use the objective function of the EOQMC problem (which assumes an EOQ value is used for the selected markets) and add the constraint

$$\sqrt{2K\lambda^\top y/h} \leq B_L \equiv \sum_{i \in I} \lambda_i y_i \leq \frac{hB_L^2}{2K},$$

which takes the same form as constraint (1), and we can therefore handle this problem in the same manner discussed at the beginning of this section. We next need to solve the following problem for the case in which $Q = B_L$:

$$\text{maximize} \quad \sum_{i \in I} \left( r_i - \frac{K}{B_L} \right) \lambda_i y_i - \frac{hB_L}{2} : \qquad y_i \in \{0,1\}, \quad i = 1, \ldots, n.$$

The optimal solution to the above problem simply selects all markets $i$ such that $r_i \geq K/B_L$, where $K/B_L$ is the per-unit setup cost (this results because the holding cost is a constant as a result of fixing $Q = B_L$). Let $I^s(B_L)$ denote the selected markets in the above problem. If $Q(I^s(B_L)) \leq B_L$, then the best solution with $Q = B_L$ is dominated by the best EOQ-feasible solution; otherwise, the solution $I^s(B_L)$ provides a candidate solution that we compare to the best EOQ-feasible solution obtained by evaluating the DNR ordered solutions.

## 2.2. The Selective Newsvendor Problem

**2.2.1. Basic Model.** In this section we consider a single-stage problem with stochastic market demands in a single-period context, which we call the *selective newsvendor problem*, or SNP. For a more in-depth discussion of the SNP, please see Taaffe et al. [31]. The newsvendor problem has a long history in the research literature, particularly in supply chain management applications in the past 20 years (see Hadley and Whitin [14] and Porteus [27]). In line with the basic newsvendor model, we consider a supplier with a single centralized stocking point for a single item with per-unit cost $c$, a salvage value $v < c$, and shortage cost $e$. In addition to these basic newsvendor assumptions, we consider a set $I$ of potential markets that the supplier may choose to serve. Market $i \in I$ provides the supplier with a per-unit net revenue $r_i$, and serving a market requires incurring a fixed market entry cost of $S_i$.

Demand during the selling season in market $i$ is denoted by the random variable $D_i$, which has probability density function (pdf) $f_i(D_i)$, cumulative distribution function (cdf) $F_i(D_i)$, mean $\mu_i$, and variance $\sigma_i^2$. We make two assumptions on market demand probability distributions for analytical tractability. We approximate demand in each market using a normal distribution and assume market demands are statistically independent. Note that the assumption of (approximate) normality is reasonable because we deal with market demands rather than individual orders. Moreover, a vast body of past research employs these assumptions in order to obtain insightful structural results for optimal solutions that can be applied more broadly (see, for example, Eppen [7] and Aviv [2]).

The sequence of events occurs as follows. Prior to the selling season, the supplier simultaneously determines the markets it will enter, along with the order quantity $Q$, which arrives at the beginning of the selling season. Market demands are then realized; if $Q$ is insufficient to satisfy all demands in the selected markets, then a shortage cost of $e$ is incurred per unit short. This unit shortage cost $e$ may correspond to either an expediting cost for units

obtained immediately from an external supplier, or to a backlogging cost for units satisfied at the end of the period through an additional replenishment made to the supplier after realizing demand (for example, at the beginning of the period, the supplier observes demand and schedules a later replenishment to cover any shortfall, which arrives at the end of the period). In either case, the entire demand is ultimately satisfied in each selected market. We will later discuss the implications of assuming that shortages result in lost sales.[1]

As in the previous EOQMC model, we let $y$ denote our binary market selection vector, with $i$th element $y_i$, and define $\mu$ and $\nu$ as corresponding vectors of market demand means and variances. The distribution of demand faced by the supplier is normal with mean $\mu^\top y$ and variance $\nu^\top y$. Let $F_y(\cdot)$ denote the cdf of total demand, and observe that the order quantity that minimizes expected cost is given by $Q^*(y) = F_y^{-1}(\rho)$, where $\rho = (e-c)/(e-v)$. Because market demands are normally distributed, we can express the optimal order quantity as $Q^*(y) = \mu^\top y + z_\rho\sqrt{\nu^\top y}$, where $z_\rho$ is the standard normal variate corresponding to the fraction $\rho$. Letting $L(z_\rho)$ denote the standard normal loss function (see Nahmias [25]), and defining $k(c,v,e) = (c-v)z_\rho + (e-v)L(z_\rho)$, we can write the minimum expected cost at $Q^*(y)$, which we denote by $C(Q^*(y))$, as

$$c\mu^\top y + k(c,v,e)\sqrt{\nu^\top y}.$$

Observe that if we select market $i$, then our expected net revenue from this market equals $(r_i - c)\mu_i - S_i \equiv \tilde{r}_i$, and we define $\tilde{r}$ as the vector of expected net revenue values; the supplier's total expected net revenue therefore equals $\tilde{r}^\top y$. Taking expected net revenue less expected cost leads to an expected profit equation of $\tilde{r}^\top y - k(c,v,e)\sqrt{\nu^\top y}$. We therefore solve the following optimization problem to maximize expected net profit.

$$\begin{aligned} [\mathbf{SNP}] \qquad \text{maximize} \quad & \sum_{i\in I} \tilde{r}_i y_i - k(c,v,e)\sqrt{\sum_{i\in I}\sigma_i^2 y_i} \\ \text{subject to:} \quad & y_i \in \{0,1\} \quad \text{for all } i \in I. \end{aligned}$$

Interestingly, the SNP model is structurally identical to the EOQMC model, although these two models employ drastically different assumptions. In the SNP, the square root term corresponds to an uncertainty-pooling term, while in the EOQMC problem it has a setup-cost-pooling interpretation. We can, of course, employ the same solution approach, although this again leads to a different interpretation. Indexing markets in nonincreasing order of the coefficient of $y_i$ in the linear term to its coefficient in the square root term leads to a ratio of the form $\tilde{r}_i/\sigma_i^2$, which we call the net revenue to uncertainty (NRU) ratio. To optimally solve [SNP], we therefore sort items in decreasing NRU (DNRU) order[2] and evaluate the expected profit of solution $j$ (containing markets $1,\ldots,j$) for $j = 0,\ldots,n$, retaining the best among these solutions as the optimal solution. The NRU ratio provides an indication of the attractiveness of a market in the selective newsvendor context, and captures the trade-off between revenue and uncertainty. We can think of this ratio as a net-revenue term adjusted for the implicit cost of uncertainty in the market.

Observe that if we assume that shortages result in lost sales, then the resulting critical fractile value $\rho$ becomes a function of the markets we select, and the problem becomes substantially more complex as a result. In the case that all market revenue terms ($\tilde{r}_i$s) are identical, however, then again the resulting $\rho$ does not depend on the selected markets, and by adjusting the $k(c,v,e)$ cost term properly, we can obtain an exact expression for expected

[1] Note that the results we obtain here are equally valid for the stationary demand, revenue, and cost data case with an infinite horizon and backlogging, assuming periodic review and zero fixed order cost.

[2] We can show that we can assume without loss of generality that no two markets exist with the same NRU ratio value; see Taaffe et al. [31].

profit that is structurally identical to the SNP (although this reduces to a somewhat trivial case in which markets are indexed in decreasing variance order). If, for example, market revenue parameters are not substantially different, or the fraction $\rho$ is close to one, we can use a similar approach to the lost-sales case with identical revenues to approximate the expected profit equation under lost sales.

Taaffe et al. [31] explore the managerial implications of the SNP model and results by examining the minimum revenue and demand levels required to make a market attractive (all else being equal), as well as the maximum level of uncertainty that would allow a new market to be attractive to the supplier.

**2.2.2. Impact of Market Effort.** In this section we consider the impacts of efforts by the supplier to influence market demands through some sort of additional market effort, such as advertising, or by providing some additional value-added service or product at no cost to customers. We generically refer to this mechanism as "market effort" and let $a_i$ denote the market effort applied to market $i$ at a per-unit cost of $t_i$. We assume for this case that markets require some market effort if they are selected, and the minimum demand in market $i$ under no market effort, $\underline{\mu}_i$, provides insufficient revenue to cover the fixed market entry cost, i.e., $S_i > (r_i - c)\underline{\mu}_i$. Here we consider the case in which expected market $i$ demand is a function of market effort $a_i$, and demand variance is independent of market effort (see Taaffe et al. [31] for an analysis that allows demand variance to be influenced by market effort). In line with much of the marketing literature (see Lilien and Rangaswamy [21]), we assume that expected demand follows an S-shaped curve as a function of market effort. That is, a small amount of market effort can create a sharp increase in market demand, while this growth begins to taper off after a large amount of market effort. Let $\mu_i(a_i)$ denote the expected market $i$ demand as a function of market effort $a_i$. This function is characterized as follows

$$\mu_i(a_i) = \begin{cases} \mu_i^{(1)}(a_i), & 0 \le a_i \le \alpha_i, \\ \mu_i^{(2)}(a_i), & a_i \ge \alpha_i, \end{cases}$$

where $\mu_i^{(1)}(a_i)$ is convex and increasing, $\mu_i^{(2)}(a_i)$ is concave and nondecreasing, $\mu_i^{(1)}(0) = \underline{\mu}_i$, and $\mu_i^{(1)}(\alpha_i) = \mu_i^{(2)}(\alpha_i)$. We assume here that $\mu_i^{(1)}$ and $\mu_i^{(2)}$ are both everywhere differentiable for $a_i \ge 0$ (see Taaffe et al. [31] for analysis of more general nondifferentiable cases).

The selective newsvendor problem with market effort (SNPM) can be formulated as follows.

$$\begin{aligned} [\textbf{SNPM}] \qquad \text{maximize} \quad & \sum_{i \in I} [(r_i - c)\mu_i(a_i) - t_i a_i - S_i] y_i - \sqrt{\sum_{i \in I} \sigma_i^2 y_i} \\ \text{subject to:} \quad & y_i \in \{0,1\} \quad \text{for all } i \in I, \\ & a_i \ge 0 \qquad\quad \text{for all } i \in I \end{aligned}$$

For any given market-selection vector $y$, the market effort decisions decompose by market. Finding the optimal amount of market effort in a market requires solving

$$\begin{aligned} \text{maximize} \quad & (r_i - c)\mu_i(a_i) - t_i a_i - S_i \\ \text{subject to:} \quad & a_i \ge 0. \end{aligned}$$

Let $\hat{a}_i^{(2)}$ denote a value of $a_i \ge \alpha_i$ such that $d\mu_i^{(2)}(a_i)/da_i = t_i/(r_i - c)$. Taaffe et al. [31] show that two candidates exist for an optimal solution to the above market $i$ effort problem: $a_i = 0$ and $a_i = \hat{a}_i^{(2)}$ (if no such value of $\hat{a}_i^{(2)}$ exists, then $a_i = 0$ is the only candidate solution). Letting $a_i^*$ denote the optimal value of $a_i$ for the market $i$ effort subproblem, and defining $\tilde{r}_i^a = (r_i - c)\mu_i(a_i^*) - t_i a_i^* - S_i$, we can replace the coefficient of $y_i$ in the SNPM formulation with this value of $\tilde{r}_i^a$, drop the last constraint set, and the resulting problem is identical to

the original SNP. Our DNRU ratio follows for assessing the attractiveness of markets, where the net-revenue parameter $\tilde{r}_i^a$ is now adjusted to reflect the optimal level of market effort and corresponding expected demand level.

Taaffe et al. [31] consider the implications of an overall marketing budget, which may not allow applying the optimal (unconstrained) market effort in each market. The resulting problem becomes a nonlinear integer programming model for which they provide a branch-and-bound algorithm, which they showed can quickly solve problems involving up to 50 markets.

## 3. Dynamic Single-Stage Models with Demand Selection

Manufacturers often face an environment in which demands for a good arrive in the form of discrete orders for a good. When production capacity is limited, a manufacturer may be forced to turn some orders away. In other contexts, it may be in the supplier's best interest to reject orders regardless of capacity levels, depending on the economics of production and how much the customer is willing to pay for the good. This section introduces a class of single-product finite-horizon planning models to handle such order selection and rejection decisions under production economies of scale and different customer reservation prices (in each case we consider a horizon of length $T$, where time periods are indexed by $t$). In certain cases, as we discuss in the following subsection, the resulting problem is equivalent to a pricing problem, where a (possibly) unique price is set in every period, which determines the aggregate demand level for the period. We first consider contexts in which capacity is effectively unlimited and customers require delivery in a prespecified period, if the supplier elects to satisfy the order. Section 3.2 generalizes this model to situations in which customers allow delivery-time flexibility (in a manner similar to the lot-sizing problem with delivery-time windows discussed in Lee et al. [20]). In §3.3, we consider the model under time-invariant production capacities, and §3.4 generalizes this model to allow for capacity-constrained overtime options and subcontracting. For each of these cases we cite our primary results and refer the reader to additional works that study the problems in greater detail.

### 3.1. Uncapacitated Order Selection Problem

Consider a supplier to whom customers submit orders for production of a product. The $j$th order in period $t$ requests $d_{jt}$ units of the good, and prior to the production horizon, the supplier must decide whether or not to accept the order, which provides a per-unit revenue of $r_{jt}$. Defining $J(t)$ as the number of orders in period $t$ and letting $y_{jt}$ denote the fraction of order $j$ in period $t$ that the supplier chooses to satisfy, the supplier's total revenue over the horizon becomes $\sum_{t=1}^{T}\sum_{j=1}^{J(t)} r_{jt}d_{jt}y_{jt}$. As in the majority of prior production-planning literature, we assume that the supplier faces a fixed production setup cost of $S_t$ if a production setup is performed in period $t$, and that a variable production cost of $c_t$ is incurred per unit produced. Let $z_t$ denote a binary variable equal to 1 if the supplier sets up in period $t$, and 0 otherwise. A holding cost of $h_t$ is incurred for each unit remaining in inventory at the end of period $t$, where $I_t$ denotes the inventory at the end of period $t$. Letting $D(t,T) = \sum_{s=t}^{T}\sum_{j=1}^{J(t)} d_{st}$ and denoting $x_t$ as the total production amount in period $t$, we can formulate the supplier's *uncapacitated order selection problem* (UOSP) as follows.

$$[\textbf{UOSP}] \quad \text{maximize} \quad \sum_{t=1}^{T}\left(\sum_{j=1}^{J(t)} r_{jt}d_{jt}y_{jt} - S_t z_t - c_t x_t - h_t I_t\right)$$

$$\text{subject to:} \quad \sum_{j=1}^{J(t)} d_{jt}y_{jt} + I_t = x_t + I_{t-1}, \quad t = 1,\ldots,T, \tag{2}$$

$$x_t \le D(t,T)z_t, \quad t = 1,\ldots,T, \tag{3}$$

$$
\begin{aligned}
& x_t, I_t \geq 0, && t = 1, \ldots, T, \\
& 0 \leq y_{jt} \leq 1, && t = 1, \ldots, T,\ j = 1, \ldots, J(t), \\
& z_t \in \{0, 1\}, && t = 1, \ldots, T
\end{aligned}
$$

The objective function maximizes total revenue less production and holding costs, while constraint (2) ensures inventory balance. Constraint (3) forces production to zero if no setup occurs; otherwise, production is limited only by the maximum remaining demand.

It is straightforward to show that an optimal solution for the UOSP exists such that (i) all orders are completely accepted or rejected (i.e., $y_{jt} = 0$ or 1 for all $j$ and $t$) and (ii) the zero-inventory production property holds (i.e., $I_{t-1}x_t = 0$ for all $t$). Using these properties, Geunes et al. [11] illustrate how to solve the UOSP using an acyclic longest-path graph structure that is nearly identical to the one used to solve the well-known Wagner and Whitin [35] economic lot-sizing problem. This graph contains $T + 1$ nodes and an arc emanating from each node to every higher-numbered node. The profit of an arc $(t, t')$ equals the maximum profit possible if we consider using the setup in period $t$ to satisfy demands in periods $t, \ldots, t' - 1$. An order's net profit will be included in this profit value if (i) its unit revenue exceeds the associated variable production and holding cost incurred, and (ii) the total net profit (total revenue less variable production and holding costs) on the arc exceeds the setup cost in period $t$ (otherwise the arc is not profitable and receives a zero profit value). The profit value of an arc $(t, t')$ then equals the maximum between the total net profit minus $S_t$, and zero. Finding the longest path in this graph solves the UOSP. If $J_{\max} = \max_{t=1,\ldots,T} J(t)$, then the worst-case complexity of this longest-path approach is $\mathcal{O}(J_{\max}T^2)$.

Geunes et al. [11] consider an analogous pricing problem that can be described as follows. Because orders in a period will be selected in nondecreasing order of their revenue ($r_{jt}$) values instead of viewing the total revenue as the sum of individual revenues from orders, we can view it as a market's piecewise-linear and concave total revenue curve. In other words, if we index orders in each period in nondecreasing revenue order, then a solution that selects orders $1, \ldots, k$ in period $t$ provides a corresponding total revenue of $\sum_{j=1}^{k} r_{jt}d_{jt}$, and the total revenue graphed as a function of the total demand satisfied in the period provides a piecewise-linear revenue curve. We might alternatively view this curve as the total revenue available in the market at a unit price of $p_{kt} = \sum_{j=1}^{k} r_{jt}d_{jt} / \sum_{j=1}^{k} d_{jt}$. Using this interpretation, $d_{jt}$ corresponds to the width of the $j$th segment of the revenue curve, and $r_{jt}$ corresponds to the slope. Thus, the UOSP is equivalent to a pricing problem when market revenue can be approximated by nondecreasing piecewise-linear concave function of the total demand satisfied (which corresponds to a demand curve that is decreasing in price). The fact that an optimal solution exists for the UOSP in which each order is either fully satisfied or rejected implies that the optimal amount of demand satisfied in the pricing problem will occur at one of the breakpoints of the piecewise-linear revenue curve. When a single price is offered to the entire market, the resulting model is equivalent to the one proposed by Thomas [33]. Geunes et al. [11] also consider a general concave revenue function, different market response curves in each period, and piecewise-linear concave production cost functions.

### 3.2. Uncapacitated Order Selection with Delivery-Timing Flexibility

Instead of requiring a fixed delivery period, in certain contexts customers might permit flexibility in the time period in which an order is delivered (Lee et al. [20] and Charnsirisakskul et al. [5] considered this type of delivery-timing flexibility in production planning contexts). For this case, we let $J$ denote the total set of orders available to the supplier over the horizon, where order $j$ requests $d_j$ units. Full delivery of order $j$ in period $t$ provides a total

revenue of $R_{jt}$, and delivery of any fraction $y_{jt}$ of the total quantity requested provides proportional revenue of $R_{jt}y_{jt}$. We formulate the uncapacitated order selection problem with delivery-timing flexibility ($\text{UOSP}_{\text{DTF}}$) as follows.

$$
\begin{aligned}
[\mathbf{UOSP_{DTF}}] \quad \text{maximize} \quad & \sum_{t=1}^{T}\left(\sum_{j=1}^{J} R_{jt}y_{jt} - S_t z_t - c_t x_t - h_t I_t\right) \\
\text{subject to:} \quad & \sum_{j=1}^{J} d_{jt}y_{jt} + I_t = x_t + I_{t-1}, \quad t = 1,\ldots,T, \\
& x_t \le D(1,T)z_t, \qquad t = 1,\ldots,T, \\
& \sum_{t=1}^{T} y_{jt} \le 1, \qquad j = 1,\ldots,J, \\
& x_t, I_t, y_{jt} \ge 0, \qquad t = 1,\ldots,T,\ j = 1,\ldots,J, \\
& 0 \le y_{jt} \le 1, \qquad t = 1,\ldots,T,\ j = 1,\ldots,J, \\
& z_t \in \{0,1\}, \qquad t = 1,\ldots,T
\end{aligned}
$$

Merzifonluoğlu and Geunes [22] show that an optimal solution for the $\text{UOSP}_{\text{DTF}}$ exists where (i) all orders are either completely satisfied or rejected, (ii) production for each satisfied order occurs within only one period, and (iii) full delivery of each accepted order occurs in a single period (in addition to the zero-inventory production property). Despite these properties, they show that when the revenue ($R_{jt}$) parameters can take arbitrary values, the $\text{UOSP}_{\text{DTF}}$ generalizes the uncapacitated facility location problem, which implies that the $\text{UOSP}_{\text{DTF}}$ is NP-hard (see Garey and Johnson [10]).

In practical contexts, however, we might expect that each customer order would have a preferred delivery period, and that the supplier's revenues might be nondecreasing in time prior to this preferred period, and nonincreasing thereafter. To reflect this possibility, suppose that order $j$ has an associated preferred delivery period of $t_j^p$, and that the supplier receives a revenue of $r_j$ per unit, regardless of the delivery period. In addition, suppose the supplier incurs a per-unit revenue loss of $h_{jt}^c$ for each unit delivered in a period $t < t_j^p$ (this lost revenue may reflect a customer holding cost for early delivery). Similarly, the supplier loses a per-unit revenue equal to $b_{jt}^c$ for units delivered in a period $t > t_j^p$ (this parameter reflects a unit backlogging cost). In addition, we make the following cost assumptions:

(1) $c_t + h_t \ge c_{t+1}$, for $t = 1,\ldots,T-1$.
(2) $h_{jt}^c \ge h_{j,t+1}^c$, for all $j \in J$ and $t < t_j^p$.
(3) $c_t + b_{jt}^c \ge c_{t-1} + b_{j,t-1}^c$, for all $j \in J$ and $t > t_j^p$.

The above assumptions imply that if we select order $j$, it is always beneficial to use either the latest setup prior to period $t_j^p$, or the earliest setup following period $t_j^p$. These assumptions are often referred to as equivalent to *nonspeculative motives* for holding inventory (or backlogging; see, for example, Lee et al. [20]). Under these assumptions, we can solve $\text{UOSP}_{\text{DTF}}$ in polynomial time using an acyclic longest-path approach similar to the one discussed in the prior section. Instead of having $T+1$ nodes, the graph now contains $T+2$ nodes, where an arc of the form $(0,t')$ implies that no setup is performed prior to period $t'$. Given an arc $(t,t')$ in the graph, for each order $j : t \le t_j^p \le t'$, we consider the net revenue obtained by assigning the order to the setups in periods $t$ and $t'$. If the higher of these two values exceeds the associated variable cost, then the order's net contribution to profit is included in the arc's profit value (otherwise, the order is "rejected" if the longest path in the graph traverses this arc). Unlike the approach in the previous section, the profit value of an arc equals the maximum variable profit (revenue less production, holding, and backorder costs) minus the setup cost in period $t$ (whereas in the previous section we took the maximum between this value and zero), which allows us to accurately account for the total setup cost

incurred on any path in the graph. The resulting worst-case complexity of this algorithm is $\mathcal{O}(|J|T^2)$. Merzifonluoğlu and Geunes [22] also provide a dual-based heuristic solution approach for the general case in which cost and revenue parameters do not follow these specialized nonspeculative motive assumptions.

### 3.3. Equal-Capacity Order Selection Problem

This section briefly discusses the implications of finite production capacity levels in dynamic deterministic order selection problems. In particular, we return to the model of §3.1 under time-invariant production capacities. For a discussion of heuristic approaches for the model under time-varying capacities, please see Taaffe and Geunes [30]. The model we consider is the same as the UOSP, except that we replace constraint (3) with the constraint

$$x_t \leq Cz_t, \quad t = 1, \ldots, T,$$

where $C$ denotes the production capacity in every period. We refer to the resulting problem as the equal-capacity order selection problem (ECOSP). Our solution approach for this problem builds on the results obtained by Florian and Klein [8], who provided a polynomial-time algorithm for the lot-sizing problem with equal capacities. Observe that for any set of $y_{jt}$ values, which determine the sequence of demands the supplier will satisfy, the resulting problem is an equal-capacity lot-sizing problem (ECLSP). The structural properties of optimal solutions for the ECLSP will thus apply to optimal solutions of the ECOSP for any given choice of $y_{jt}$ values. The remaining difficulty then lies in determining a manageable characterization of the orders that will be selected in an optimal solution.

Our approach for this problem relies on the optimality of *capacity-constrained sequences* for the ECLSP. Given a *regeneration interval* (i.e., a sequence of periods $\tau, \ldots, \tau'$ such that $I_\tau = I_{\tau'} = 0$ and $I_t > 0$ for $t = \tau + 1, \ldots, \tau' - 1$), a capacity-constrained sequence is one in which production output $x_t$ equals either 0 or $C$ in all periods from $\tau + 1$ to $\tau' - 1$, except for at most one of these periods. If we are required to satisfy a total demand of $D(\tau, \tau')$ in periods $\tau + 1, \ldots, \tau' - 1$, then for the period in the capacity-constrained sequence in which we produce neither 0 nor $C$, we must produce $\epsilon$, where $\epsilon = D(\tau, \tau') (\text{mod } C)$. Given $D(\tau, \tau')$, we also know that the total number of setups that must occur in periods $\tau + 1, \ldots, \tau' - 1$ must equal $k + 1$, where $k = \lfloor D(\tau, \tau')/C \rfloor$. Florian and Klein [8] show that given $D(\tau, \tau')$, the subproblem to determine the optimal production plan for regeneration interval $(\tau, \tau')$ can be solved in $\mathcal{O}(T^2)$ time.

Geunes et al. [11] provide results that allow considering a limited number of demand values for every potential regeneration interval. That is, given a potential regeneration interval $(\tau, \tau')$, efficiently solving ECOSP requires consideration of a manageable number of possible $D(\tau, \tau')$ values. To do this, we define an adjusted revenue parameter, $\rho_{jt} = r_{jt} + \sum_{s=t}^{T} h_s$. Given a regeneration interval $(\tau, \tau')$, Geunes et al. [11] show that if $\rho_{jt} \geq \rho_{it'}$ (with both $t$ and $t'$ between $\tau$ and $\tau'$), then if an optimal regeneration interval solution exists with $y_{it'} > 0$, an optimal solution exists with $y_{jt} = 1$ (equivalently, if $y_{jt} < 1$, then $y_{it'} = 0$). This implies that within any regeneration interval we will have at most one order that is partially satisfied. In addition, Geunes et al. [11] show that when an order is partially satisfied within a regeneration interval, then all production levels must be at 0 or $C$. Similarly, when one period within a regeneration interval exists in which production is at neither 0 nor $C$, all orders within the regeneration interval will be either fully satisfied or completely rejected.

These results imply that there are $\mathcal{O}(J_{\max}T)$ candidate demand vectors that we must consider within any regeneration interval. Given that we can determine the optimal production plan for a given demand vector and regeneration interval in $\mathcal{O}(T^2)$ time, the time required to solve a regeneration interval subproblem (considering all candidate demand vector values) is bounded by $\mathcal{O}(J_{\max}T^3)$. The number of possible regeneration intervals is $\mathcal{O}(T^2)$, which implies that the ECOSP can be solved in $\mathcal{O}(J_{\max}T^5)$ time in the worst case. Note that

this result improves the worst-case complexity found in Geunes et al. [11] by an order of magnitude, based on a recent analysis and improvement of the algorithm.

The analogous pricing problem follows from our discussion in §3.1. That is, within each period we sort orders in nonincreasing order of revenue values ($r_{jt}$s), and take these values as the slopes of the piecewise-linear and concave total revenue function (as a function of total demand satisfied). The width of the $j$th segment of the revenue function in period $t$ equals $d_{jt}$. Observe that, unlike the UOSP pricing analog, under capacity constraints it is possible that an optimal solution exists such that the optimal amount of total demand satisfied in a period occurs at a value that is between breakpoints of the piecewise-linear revenue curve.

### 3.4. ECOSP with Capacitated Overtime and Subcontracting

Firms with internal production capacity limits often have options for exceeding that capacity through, for example, the use of overtime and/or subcontracting. Such options in order selection contexts may permit a supplier to accept a greater amount of total demand to increase total revenue. In this section, we generalize the ECOSP to address the trade-off between the additional capacity costs associated with overtime and subcontracting, and the potential for increasing revenue by using these additional sources of capacity. Atamtürk and Hochbaum [1] considered the dynamic economic lot-sizing problem with subcontracting options, and our approach is structurally similar to theirs, although they did not consider an order selection (or the analogous pricing problem) dimension, or the availability of overtime as a source of additional capacity.

Overtime options are often viewed as an alternative form of subcontracting from a modeling perspective, although in practice, distinct differences typically exist. For example, a firm can choose to completely subcontract without using any internal capacity, whereas overtime is only used after exhausting regular internal capacity. Moreover, if regular-time production capacity is limited, then some associated limit on overtime capacity will also likely exist, and the costs of regular-time and overtime production will contain a similar structure (with the overtime cost rate being higher than the regular-time cost rate). The model we present in this section captures these subtle but important differences between overtime and subcontracting costs.

We generalize the cost model used in the ECOSP by allowing the production cost function in period $t$, which we denote by $p_t(x_t)$, to be concave and nondecreasing in $x_t$ (as opposed to the fixed-charge structure we employed in the previous section), where $x_t$ is now defined as the total production from internal sources (regular and overtime production) in period $t$. Letting $v_t$ and $\varepsilon_t$ denote the regular and overtime production output in period $t$, respectively, we have $x_t = v_t + \varepsilon_t$. We define the function $o_t(\varepsilon_t)$ as an *incremental* cost function for units produced during overtime, where $o_t(\varepsilon_t)$ is also concave and nondecreasing. The amount of *overtime capacity* in a period in this model equals $C$, and the amount of *regular-time capacity* equals $\kappa C$, where $\kappa$ is a positive integer. The amount of available overtime capacity is therefore some fraction $1/\kappa$ of the regular capacity in a period (the model and results presented here also apply when available overtime capacity is a positive integer multiple of regular capacity).

The total production cost is therefore equal to $p_t(x_t)$ when $x_t \leq \kappa C$, and $p_t(x_t) + o_t(\varepsilon_t)$ when $C < x_t \leq \kappa C$. The regular production cost function $p_t(x_t)$ accounts for the basic production cost structure for using internal capacity, while the incremental overtime cost function $o_t(\varepsilon_t)$ accounts for the incremental cost of units produced during overtime hours. Thus, the structure of the total production cost function in a period depends on the amount of regular internal capacity $C$ (Merzifonluoğlu et al. [23] consider additional model extensions in which additional regular capacity is available to the supplier at a cost, and this capacity level therefore becomes a decision variable). In effect, because economies of scale

in production exist, the total cost of a unit of overtime production depends on how much regular-time production is available.

In addition to overtime options, we assume that an external subcontractor is available for producing the good, and define $g_t(s_t)$ as the subcontracting cost function in period $t$, where $s_t$ is the amount subcontracted in the period (we assume that no limit exists on the amount of subcontracting available in any period). We formulate the ECOSP with overtime and subcontracting options ($\text{ECOSP}_{\text{OS}}$) as follows.

$$
\begin{aligned}
[\textbf{ECOSP}_{\textbf{OS}}] \quad \text{minimize} \quad & \sum_{t=1}^{T}(p_t(v_t+\varepsilon_t)+o_t(\varepsilon_t)+g_t(s_t)-h_tI_t)-\sum_{t=1}^{T}\sum_{j=1}^{J(t)} r_{jt}d_{jt}y_{jt} \\
\text{subject to:} \quad & \sum_{j=1}^{J(t)} d_{jt}y_{jt}+I_t = v_t+\varepsilon_t+s_t+I_{t-1}, \quad t=1,\ldots,T, \\
& v_t \le \kappa C, \qquad t=1,\ldots,T, \\
& \varepsilon_t \le C, \qquad t=1,\ldots,T, \\
& 0 \le y_{jt} \le 1, \qquad t=1,\ldots,T,\ j=1,\ldots,J(t), \\
& v_t,\varepsilon_t,s_t,I_t \ge 0, \qquad t=1,\ldots,T
\end{aligned}
$$

The $\text{ECOSP}_{\text{OS}}$ minimizes a concave function over a polyhedron, which implies that an extreme-point optimal solution exists. We next provide a characterization of the extreme points of the above polyhedron, which allows us to provide a polynomial-time solution approach. The following lemma paves the way for solving a set of shortest-path subproblems in order to determine the optimal order selection and production-planning decisions for a regeneration interval.

**Lemma 1 (Merzifonluoğlu et al. [23]).** *In an extreme-point solution for* $[ECOSP_{\text{OS}}]$*, a regeneration interval can have at most one period* $t$ *with* $0 < v_t < \kappa C$ *(fractional regular-time production), at most one period* $t$ *with either* $0 < \varepsilon_t < C$ *(fractional overtime production) or* $s_t > 0$ *(subcontracting), or at most one period with* $0 < y_{jt} < 1$ *(partial demand satisfaction), but not any of these simultaneously.*

Lemma 1 implies that given a potential regeneration interval $(t,t')$, we need to consider four types of solutions:

(1) One period containing fractional regular production with overtime production always at 0 or $C$, zero subcontracting, and all orders fully satisfied or rejected;

(2) One period containing fractional overtime production with regular production always at 0 or $\kappa C$, zero subcontracting, and all orders fully satisfied or rejected;

(3) One period containing subcontracting, with regular-time (overtime) production always at 0 or $\kappa C$ (0 or $C$), and all orders fully satisfied or rejected;

(4) One period containing a fractionally satisfied order with regular-time (overtime) production always at 0 or $\kappa C$ (0 or $C$), and zero subcontracting.

For each of these types of regeneration intervals, and given any potential regeneration interval $(t,t')$, we can determine the optimal regeneration interval solution of each type in polynomial time. This solution relies on using the adjusted revenue parameters $\rho_{jt}$, described in the previous section, to measure the relative preferences of orders within any regeneration interval. For regeneration interval types (1)–(3), because we know that all orders are either fully satisfied or rejected, we have $\mathcal{O}(J_{\max}T)$ candidate demand vectors for the regeneration interval demand $D(t,t')$. Given any one of these candidate demand vectors, for each type of regeneration interval, we can determine a discrete number of possible cumulative production quantities throughout the regeneration interval. We use a layered graph with $t'-t$ layers (one for each period within the regeneration interval, plus a start node), where a node exists in each layer for every possible cumulative production value. Arcs connect each

layer to the next-highest layer based on feasible cumulative production transitions between periods. We then solve a shortest-path problem on this network to determine the best regeneration interval solution for each type. Merzifonluoğlu et al. [23] show that each of these shortest-path graphs is solvable in polynomial time, and we must solve $\mathcal{O}(J_{\max}T)$ of these shortest-path problems within each possible regeneration interval. For regeneration interval type (4), although at most one order may be satisfied at a fractional value, we know that cumulative production in a regeneration interval must be an integer multiple of $C$. Given an integer multiple of $C$, we can then determine which orders (demand) to satisfy based on filling the capacity with orders in nonincreasing order of the $\rho_{jt}$ values. Merzifonluoğlu et al. [23] show that the worst-case complexity for solving the regeneration interval subproblem is $\mathcal{O}(\kappa^2 J_{\max}T^4)$, which leads to the following theorem.

**Theorem 1 (Merzifonluoğlu et al. [23]).** *A polynomial-time algorithm exists for the $ECOSP_{OS}$ with worst-case complexity of $\mathcal{O}(\kappa^2 J_{\max}T^6)$.*

## 4. Multifacility Demand Assignment Models

Our focus thus far has dealt exclusively with single-stage models. This section illustrates the role these models might play in a larger tactical distribution network design context. We consider two-echelon problems containing an upstream supply echelon and a downstream demand echelon. The upstream echelon might, for example, consist of production/distribution (P/D) facilities, while the downstream echelon might correspond to retail sites. Our model takes the supplier perspective, considering upstream production and inventory holding costs, plus any additional costs (such as transportation costs) associated with meeting retailer demands. We assume the supplier has a number of geographically dispersed P/D facilities, each of which can produce a given product. The supplier faces demands from a number of retail locations for the product and wishes to determine which source facility will serve each retail site. Our approach assumes that the supplier uses a *single-sourcing* strategy, with each retail site being served by (or assigned to) a single upstream P/D facility. While some distribution systems might employ transhipments in extraordinary situations, as a primary strategy, this single-sourcing approach is often found in practice because it reduces coordination complexity and decreases information systems integration requirements. Our model therefore applies to contexts where such single-sourcing is the primary operating approach. Due to space limitations, we provide a sketch of our overall solution approach for these problems. For more details on this approach, and in particular how it can be applied using a cost model of the form discussed in §§2.1 and 2.2, please see Huang et al. [16] and Taaffe et al. [32].

### 4.1. Model Formulation

Our modeling approach begins with a general set-partitioning formulation that partitions a set of $n$ retailers into $m$ subsets, where $m$ is the number of P/D facilities. After formulating this set-partitioning problem, we discuss a column generation approach in which the column-pricing problem brings us back to the single-stage models that have served as the primary focus of this chapter. We begin by letting $K_i$ denote the number of subsets of retailers that can be assigned to facility $i$, for $i = 1, \ldots, m$. We define $\gamma_{i\cdot}^k$ as an indicator vector corresponding to the $k$th feasible subset of retailers at facility $i$, where $\gamma_{ij}^k = 1$ if retailer $j$ is included in facility $i$'s $k$th feasible subset. The function $H_i(\gamma_{i\cdot}^k)$ (where $H_i : \{0,1\}^n \rightarrow \mathbb{R} \cup \{+\infty\}$) provides the cost of assigning the $k$th feasible subset to facility $i$, and we let $y_i^k$ equal 1 if this assignment is made, and 0 otherwise. Constraints may be incorporated into this cost function ($H_i$) by assigning a cost of $+\infty$ to any infeasible assignment. At this point we do not provide any further specification of the $H_i(\cdot)$ functions, although we will return

to this later when we discuss the pricing problem. For now, we simply assume that given $\gamma_{i\cdot}^k$, we can compute $H_i(\gamma_{i\cdot}^k)$. We formulate our set-partitioning (SP) formulation as follows.

$$
\begin{aligned}
[\mathbf{SP}] \quad \text{minimize} \quad & \sum_{i=1}^{m}\sum_{k=1}^{K_i} H_i(\gamma_{i\cdot}^k) y_i^k \\
\text{subject to:} \quad & \sum_{i=1}^{m}\sum_{k=1}^{K_i} \gamma_{ij}^k y_i^k = 1, \quad j=1,\ldots,n, \\
& \sum_{k=1}^{K_i} y_i^k = 1, \qquad i=1,\ldots,m, \\
& y_i^k \in \{0,1\}, \qquad i=1,\ldots,m,\ k=1,\ldots,K_i
\end{aligned}
$$

The objective function minimizes the sum of the assignment costs at each facility, while the constraints ensure that every retailer is assigned to a facility and that some subset is assigned to each facility. Our solution approach begins with some subset $\widehat{K}_i \subset K_i$ $(i = 1,\ldots,m)$ that admits a feasible solution to the linear programming relaxation of [SP], which we call LP(SP). After solving LP(SP) with the subset of columns, we then solve a pricing problem for each facility $i$. If the optimal solution value to the pricing problem for facility $i$ is negative, the resulting new column is added to the set $\widehat{K}_i$, and we repeat the process. If all $m$ pricing problems produce nonnegative solution values, then we have an optimal solution to LP(SP). If this solution is integral, then we have an optimal solution for [SP]. Otherwise, we apply a branch-and-price algorithm to solve [SP] to optimality (see Huang et al. [16] for details on this branch-and-price algorithm). The next section discusses the pricing problem for LP(SP).

### 4.2. The Pricing Problem

Let $u_j^*$ and $v_i^*$ denote optimal dual vectors corresponding to the constraints in LP(SP) for some set of $\widehat{K}_i$ values. The pricing problem for facility $i$ is written as

$$
\begin{aligned}
[\mathrm{PP}_i] \quad \text{minimize} \quad & H_i(z) - \sum_{j=1}^{n} u_j^* z_j + v_i^* \\
\text{subject to:} \quad & z_j \in \{0,1\}, \quad j=1,\ldots,n,
\end{aligned}
$$

where an optimal solution $z^* = (z_1^*,\ldots,z_n^*)^\top$ provides a new member for the set $\widehat{K}_i$ when $H_i(z^*) - \sum_{j=1}^{n} u_j^* z_j^* + v_i^* < 0$. If we compare the pricing problem formulation $[\mathrm{PP}_i]$ to the [EOQMC] formulation of §2.1 or the [SNP] formulation of §2.2, we see that through a proper definition of the $H_i(\cdot)$ function, the pricing problem becomes equivalent to these problems. Taken in the context of these models, the $z_j$ variables correspond to market (or retailer) selection variables, and the dual price $u_j^*$ provides either an additional revenue term for selecting market (retailer) $j$ (if $u_j^* > 0$) or a cost penalty (if $u_j^* < 0$). Thus, if the facility cost (and revenue) structures are consistent with the [EOQMC] or [SNP] model, we have efficient methods for solving the pricing problem within a larger distribution network design problem. Huang et al. [16] directly address the case in which facility costs take the EOQ cost structure, and provide computational test results using the branch-and-price procedure, which they show can solve problems of reasonably large size (up to 10 facilities and 50 retailers) in acceptable computing time (1–4 minutes). Taaffe et al. [32] provide motivating contexts and a solution approach when supplier cost structures are consistent with those of the [SNP] model.

Because we have defined these cost functions $H_i(\cdot)$ very generally, and by incorporating the proper cost structures and constraints in the definition of these functions, the pricing

problem for each facility $i$ can in principle also incorporate the class of dynamic order selection problems covered in §3. In this case, instead of the downstream demand echelon consisting of facilities, it might consist of a pool of time-phased (or time-flexible) orders that the supplier firm must satisfy, where facility cost structures are consistent with those defined in §3. One caveat applies, however, when considering the order selection models of §3 in general. In the absence of production capacities, we know that all orders will be either fully accepted or rejected by a facility. Our capacity-constrained models, however, allowed for partial satisfaction of orders. If orders must be completely accepted or rejected by facilities under capacity constraints (which would likely be the case in the distribution network design problem), the resulting complexity of the pricing problems (which then generalize the NP-hard knapsack problem), might preclude the column generation approach (which then needs to be embedded in a branch-and-price algorithm) as a viable solution method. However, the results in Huang et al. [16] indicate that high-quality solutions can often still be found in reasonable time in such cases. Further research is required to determine situations in which capacity-constrained facilities with all-or-nothing order satisfaction requirements would be amenable to this solution method.

## 5. Concluding Remarks and Future Research Directions

This chapter provided an overview of a collection of related models that address a supplier's ability to influence the collective demands to which it will respond. The majority of these models apply to single-stage, single-product contexts, and generalize classical inventory models to address elements of demand selection. While we posed the problem in the context of market (or order) selection decisions, we can view such selection decisions as an explicit mechanism for shaping demand characteristics. For certain cases, we discussed a different interpretation of the models in which pricing served as an implicit mechanism for effectively selecting demand levels. As the previous section illustrated, these demand selection problems also arise as pricing subproblems in distribution network design problems. The solution approaches we have developed for the single-stage models can therefore serve as a subroutine for solving these larger network design models.

As we noted in the previous section, the dynamic models discussed in §3 can be, in principle incorporated as subproblems in a distribution network design problem that accounts for economies of scale at facilities that must collectively respond to a firm's set of orders over multiple periods. Exploring models under which this solution method is efficient for this class of problems serves as a potential avenue for further research. When the context requires that all orders are either accepted in full or completely rejected, further methodologies must be explored that can handle the added complexity this implies. Our distribution network design approach can also be extended to address more comprehensive supply chain cost structures, where economies of scale exist in transportation between echelons, and inventory costs at both echelons of the supply chain are incorporated.

## Acknowledgments

This research was partially supported by NSF Grants DMI-0355533 and DMI-0322715.

## References

[1] A. Atamtürk and D. S. Hochbaum. Capacity acquisition, subcontracting, and lot sizing. *Management Science* 47(8):1081–1100, 2001.

[2] Y. Aviv. The effect of collaborative forecasting on supply chain performance. *Management Science* 47(10):1326–1343, 2001.

[3] S. Biller, L. M. A. Chan, D. Simchi-Levi, and J. Swann. Dynamic pricing and the direct-to-customer model in the automotive industry. *Electronic Commerce Research* 5(2):309–334, 2005.

[4] S. Carr and W. Lovejoy. The inverse newsvendor problem: Choosing an optimal demand portfolio for capacitated resources. *Management Science* 46(7):912–927, 2000.

[5] K. Charnsirisakskul, P. M. Griffin, and P. Keskinocak. Order selection and scheduling with leadtime flexibility. *IIE Transactions* 36(7):697–707, 2004.

[6] R. E. Crandall and R. E. Markland. Demand management—Today's challenge for service industries. *Production & Operations Management* 5(2):106–120, 1996.

[7] G. Eppen. Effects of centralization on expected costs in a multi-location newsboy problem. *Management Science* 25(5):498–501, 1979.

[8] M. Florian and M. Klein. Deterministic production planning with concave costs and capacity constraints. *Management Science* 18(1):12–20, 1971.

[9] G. Gallego and G. van Ryzin. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Science* 40(8):999–1020, 1994.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, 1979.

[11] J. Geunes, H. E. Romeijn, and K. Taaffe. Requirements planning with pricing and order selection flexibility. *Operations Research*. Forthcoming.

[12] J. Geunes, Z.-J. Shen, and H. E. Romeijn. Economic ordering decisions with market choice flexibility. *Naval Research Logistics* 51(1):94–112, 2004.

[13] S. M. Gilbert. Coordination of pricing and multi-period production for constant priced goods. *European Journal of Operational Research* 114:330–337, 1999.

[14] G. Hadley and T. M. Whitin. *Analysis of Inventory Systems*. Prentice Hall, Englewood Cliffs, NJ, 1963.

[15] F. W. Harris. How many parts to make at once. *Factory, The Magazine of Management* 10:135–136,152, 1913.

[16] W. Huang, H. E. Romeijn, and J. Geunes. The continuous-time single-sourcing problem with capacity constraints and expansion opportunities. *Naval Research Logistics* 52(3):193–211, 2005.

[17] A. V. Iyer, V. Deshpande, and Z. Wu. A postponement model for demand management. *Management Science* 49(8):983–1002, 2003.

[18] P. Keskinocak, R. Ravi, and S. Tayur. Scheduling and reliable lead time quotation for orders with availability intervals and lead-time sensitive revenues. *Management Science* 47(2):264–279, 2001.

[19] H. Kunreuther and L. Schrage. Joint pricing and inventory decisions for constant priced items. *Management Science* 19(7):732–738, 1973.

[20] C. Y. Lee, S. Çetinkaya, and A. P. M. Wagelmans. A dynamic lot sizing model with demand time windows. *Management Science* 47(10):1384–1395, 2001.

[21] G. L. Lilien and A. Rangaswamy. *Marketing Management and Strategy: Marketing Engineering Applications*, 2nd ed. Prentice Hall, Englewood Cliffs, NJ, 2003.

[22] Y. Merzifonluoğlu and J. Geunes. Uncapacitated production and location planning models with demand fulfillment flexibility. *International Journal of Production Economics*. Forthcoming.

[23] Y. Merzifonluoğlu, J. Geunes, and H. E. Romeijn. Production planning models with variable demand and capacity. Working paper, University of Florida, Gainesville, FL, 2005.

[24] D. R. Moodie. Demand management: The evaluation of price and due date negotiation strategies using simulation. *Production & Operations Management* 8(2):151–162, 1999.

[25] S. Nahmias. *Production and Operations Analysis*, 5th ed. McGraw-Hill Irwin, Boston, MA, 2005.

[26] N. P. Petruzzi and M. Dada. Pricing and the newsvendor problem: A review with extensions. *Operations Research* 47(2):183–194, 1999.

[27] E. L. Porteus. Stochastic inventory theory, Ch. 2. D. P. Heyman and M. J. Sobel, eds. *Stochastic Models. Handbooks in Operations Research and Management Science*, Vol. 2. North-Holland, Amsterdam, The Netherlands, 1990.

[28] Z.-J. Shen, C. Coullard, and M. Daskin. A joint location-inventory model. *Transportation Science* 37(1):40–55, 2003.

[29] E. A. Silver, D. F. Pyke, and R. Peterson. *Inventory Management and Production Planning and Scheduling*, 3rd ed. John Wiley & Sons, New York, 1998.

[30] K. Taaffe and J. Geunes. Models for integrated order selection and requirements planning decisions under limited production capacity. P. M. Pardalos, A. Migdalas, and G. Baourakis, eds. *Supply Chain and Finance*. Series on Computers and Operations Research, Vol. 2. World Scientific, River Edge, NJ, 2004.

[31] K. Taaffe, J. Geunes, and H. E. Romeijn. Integrated marketing and operational decisions: The selective newsvendor problem. Working paper, University of Florida, Gainesville, FL, 2005.

[32] K. Taaffe, J. Geunes, and H. E. Romeijn. Capacity acquisition and stochastic customer demand assignment in a network of facilities. *Proceedings of the 2005 Annual IE Research Conference*, Atlanta, GA, 2005.

[33] J. Thomas. Price-production decisions with deterministic demand. *Management Science* 16(11):747–750, 1970.

[34] W. van den Heuvel and A. P. M. Wagelmans. A polynomial time algorithm for a deterministic joint pricing and inventory model. *European Journal of Operational Research*. Forthcoming.

[35] H. Wagner and T. Whitin. Dynamic version of the economic lot size model. *Management Science* 5:89–96, 1970.

[36] L. R. Weatherford and S. E. Bodily. A taxonomy and research overview of perishable-asset revenue management: Yield management, overbooking and pricing. *Operations Research* 40(5):831–844, 1992.

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS | ISBN 1-877640-21-2
DOI 10.1287/educ.1053.0019

# An Introduction to Revenue Management

***Garrett J. van Ryzin***
Graduate School of Business, Columbia University, 3022 Broadway, New York, New York 10027, gjv1@columbia.edu

***Kalyan T. Talluri***
Department of Economics and Business, Universitat Pompeu Fabra, Jaume I Building, Ramon Trias Fargas, 25–27, 08005 Barcelona, Spain, kalyan.talluri@upf.edu

**Abstract** *Revenue management* (RM) refers to the collection of strategies and tactics firms use to scientifically manage demand for their products and services. It has gained attention recently as one of the most successful application areas of operations research (OR). The practice has grown from its origins as a relatively obscure practice among a handful of major airlines in the postderegulation era in the United States (circa 1978) to its status today as a mainstream business practice with a growing list of industry users, ranging from Walt Disney Resorts to National Car Rental. The economic impact of RM is significant, with increases in revenue of 5% or more reported in several industry applications of RM systems. Professional practice and research in the area is also expanding. There are now several major industry RM conferences each year, and published research on the methodology of RM has been growing rapidly. This chapter provides an introduction to this increasingly important subfield of OR. It is based on excerpts from our book *The Theory and Practice of Revenue Management* [75].

**Keywords** revenue management; dynamic pricing; optimization; demand modeling; demand management; capacity control; tutorial

## 1. Introduction

Every seller of a product or service faces a number of fundamental decisions. A child selling lemonade outside her house has to decide on which day to have her sale, how much to ask for each cup, and when to drop the price (if at all) as the day rolls on. A homeowner selling a house must decide when to list it, what the asking price should be, which offer to accept, and when to lower the listing price—and by how much—if no offers come in. Anyone who has ever faced such decisions knows the uncertainty involved. You want to sell at a time when market conditions are most favorable, but who knows what the future might hold? You want the price to be right—not so high that you put off potential buyers and not so low that you lose out on potential profits. You would like to know how much buyers value your product, but more often than not you must just guess at this number.

Businesses face even more complex selling decisions. For example, how can a firm segment buyers by providing different conditions and terms of trade that profitably exploit their different buying behavior or willingness to pay? How can a firm design products to prevent cannibalization across segments and channels? Once it segments customers, what prices should it charge each segment? If the firm sells in different channels, should it use the same price in each channel? How should prices be adjusted over time, based on seasonal factors and the observed demand to date for each product? If a product is in short supply, to which segments and channels should it allocate the products? How should a firm manage the pricing and allocation decisions for products that are complements (seats on two connecting airline flights) or substitutes (different car categories for rentals)?

RM is concerned with such *demand-management* decisions[1] and the methodology and systems required to make them. It involves managing the firm's "interface with the market," as it were—with the objective of *increasing revenues*. RM can be thought of as the complement of *supply chain management* (SCM), which addresses the *supply decisions* and processes of a firm with the objective (typically) of *lowering the cost* of production and delivery.

Other roughly synonymous names have been given to the practice over recent years—*yield management* (the traditional airline term), *pricing and revenue management*, *pricing and revenue optimization*, *revenue process optimization*, *demand management*, *demand-chain management* (favored by those who want to create a practice parallel to supply chain management)—each with its own nuances of meaning and positioning. However, we use the more standard term revenue management to refer to the wide range of techniques, decisions, methods, processes, and technologies involved in demand management.

## 1.1. Demand-Management Decisions

RM addresses three basic categories of demand-management decisions:

- Structural decisions: Which selling format to use (such as posted prices, negotiations, or auctions); which segmentation or differentiation mechanisms to use (if any); which terms of trade to offer (including volume discounts and cancellation or refund options); how to bundle products; and so on.
- Price decisions: How to set posted prices, individual-offer prices, and reserve prices (in auctions); how to price across product categories; how to price over time; how to mark down (discount) over the product lifetime; and so on.
- Quantity decisions: Whether to accept or reject an offer to buy; how to allocate output or capacity to different segments, products, or channels; when to withhold a product from the market and sale at later points in time; and so on.

Which of these decisions is most important in any given business depends on the context. The timescale of the decisions varies as well. Structural decisions about which mechanism to use for selling and how to segment and bundle products are normally strategic decisions taken relatively infrequently. Firms may also have to commit to certain price or quantity decisions, for example, by advertising prices in advance or deploying capacity in advance, which can limit their ability to adjust price or quantities on a tactical level. The ability to adjust quantities may also be a function of the technology of production—the flexibility of the supply process and the costs of reallocating capacity and inventory. For example, the use of capacity controls as a tactic in airlines stems largely from the fact that the different "products" an airline sells (different ticket types sold at different times and under different terms) are all supplied using the same homogeneous seat capacity. This gives airlines tremendous quantity flexibility, so quantity control is a natural tactic in this industry. Retailers, in contrast, often commit to quantities (initial stocking decisions) but have more flexibility to adjust prices over time. The ability to price tactically, however, depends on how costly price changes are, which can vary depending on the channel of distribution, such as online versus catalog.

Whether a firm uses quantity-based or price-based RM controls varies even across firms within a given industry. For instance, while most airlines commit to fixed prices and tactically allocate capacity, low-cost carriers tend to use price as the primary tactical variable.

Broadly speaking, RM addresses all three categories of demand-management decisions—structural, pricing, and quantity decisions. We qualify RM as being either *quantity-based RM* or *price-based RM* if it uses (inventory- or) capacity-allocation decisions or prices as the

[1] These can be referred to as either *sales decisions* (we are making decisions on where and when to sell and to whom and at what price) or *demand-management decisions* (we are estimating demand and its characteristics and using price and capacity control to "manage" demand). We use the latter consistently and use the shorter *demand management* whenever appropriate.

primary tactical tool, respectively, for managing demand. Both the theory and practice of RM differ depending on which control variable is used, and hence we use this dichotomy as necessary.

## 1.2. What's New About RM?

In one sense, RM is a very old idea. Every seller in human history has faced RM-type decisions. What price to ask? Which offers to accept? When to offer a lower price? And when to simply "pack up one's tent" as it were and try selling at a later point in time or in a different market. In terms of business practice, the problems of RM are as old as business itself.

In terms of theory, at a broad level the problems of RM are not new either. Indeed, the forces of supply and demand and the resulting process of price formation—the "invisible hand" of Adam Smith—lie at the heart of our current understanding of market economics. They are embodied in the concept of the "rational" (profit-maximizing) firm, and define the mechanisms by which market equilibria are reached. Modern economic theory addresses many advanced and subtle demand-management decisions, such as nonlinear pricing, bundling, segmentation, and optimizing in the presence of asymmetric information between buyers and sellers.

What *is* new about RM is not the demand-management decisions themselves, but rather *how* these decisions are made. The true innovation of RM lies in the *method* of decision making—a technologically sophisticated, detailed, and intensely operational approach to making demand-management decisions.

This new approach is driven by two complementary forces. First, scientific advances in economics, statistics, and operations research now make it possible to model demand and economic conditions, quantify the uncertainties faced by decision makers, estimate and forecast market response, and compute optimal solutions to complex decision problems. Second, advances in information technology provide the capability to automate transactions, capture and store vast amounts of data, quickly execute complex algorithms, and then implement and manage highly detailed demand-management decisions. This combination of science and technology applied to age-old demand management is the hallmark of modern RM.

Also, both the science and technology used in RM are quite new. Much of the science used in RM today (demand models, forecasting methods, optimization algorithms) is less than 50 years old, most of the information technology (large databases, personal computers, Internet) is less than 20 years old, and most of the software technology (Java, object-oriented programming) is less than 5 years old. Prior to these scientific developments, it would have been unthinkable to accurately model real-world phenomena and demand-management decisions. Without the information technology, it would be impossible to *operationalize* this science. These two capabilities *combined* make possible an entirely new approach to decision making—one that has profound consequences for demand management.

The first consequence is that science and technology now make it possible to manage demand on a *scale and complexity* that would be unthinkable through manual means (or would require a veritable army of analysts to achieve). A modern large airline, for example, can have thousands of flights a day and provide service between hundreds of thousands of origin-destination pairs, each of which is sold at dozens of prices—and this entire problem is replicated for hundreds of days into the future!

The second consequence of science and technology is that they make it possible to improve the *quality* of demand-management decisions. The management tasks that are involved—quantifying the risks and rewards in making demand-management decisions under uncertainty; working through the often subtle economics of pricing; accurately interpreting market conditions and trends and reacting to this information with timely, accurate, and consistent real-time decisions; optimizing a complex objective function subject to many constraints and business rules—are tasks that most humans, even with many years of experience, are simply not good at.

Of course, even with the best science and technology, there will always be decisions that are better left to human decision makers. Most RM systems recognize this fact and parse the decision-making task, with models and systems handling routine demand-management decisions on an automated basis and human analysts overseeing these decisions and intervening (based on flags or alerts from the system) when extraordinary conditions arise. Such man-machine interaction offers a firm the best of both human and automated decision making.

The process of managing demand decisions with science and technology—implemented with disciplined processes and systems, and overseen by human analysts (a sort of "industrialization" of the entire demand-management process)—defines modern RM.

### 1.3. The Origins of RM

Where did RM come from? In short, the airline industry. There are few business practices whose origins are so intimately connected to a single industry. Here we briefly review the history of airline RM and then discuss the implications of this history for the field.

The starting point for RM was the Airline Deregulation Act of 1978. With this act, the U.S. Civil Aviation Board (CAB) loosened control of airline prices, which had been strictly regulated based on standardized price and profitability targets. Passage of the act led to rapid change and a rash of innovation in the industry. Established carriers were now free to change prices, schedules, and service without CAB approval. At the same time, new low-cost and charter airlines entered the market. Because of their lower labor costs, simpler (point-to-point) operations, and no-frills service, these new entrants were able to profitably price much lower than the major airlines. They tapped into an entirely new and vast market for discretionary travel—families on a holiday, couples getting away for the weekend, college students visiting home—many of whom might otherwise have driven their cars or not traveled at all.

The potential of this market was embodied in the rapid rise of PeopleExpress, which started in 1981 with cost-efficient operations and fares 50% to 70% lower than the major carriers. By 1984, its revenues were approaching $1 billion, and for the year 1984 it posted a profit of $60 million, its highest profit ever (Cross [21]). These developments resulted in a significant migration of price-sensitive discretionary travelers to the new, low-cost carriers and the cumulative losses in revenue from the shift in traffic were badly damaging the profits of major airlines.

A strategy to recapture the leisure passenger was needed. However, for the majors, a head-to-head, across-the-board price war with the upstarts was deemed almost suicidal. Robert Crandall, American Airline's vice president of marketing at the time, is widely credited with the breakthrough in solving this problem. He recognized that his airline was already producing seats at a marginal cost near zero because most of the costs of a flight (capital costs, wages, fuel) are fixed. As a result, American could in fact afford to compete on cost with the upstarts using its surplus seats.

However, two problems had to be solved to execute this strategy. First, American had to have some way of identifying the surplus seats on each flight. Second, they had to ensure that American's business customers did not switch and buy the new low-price products it offered to discretionary, leisure customers.

American solved these problems using a combination of *purchase restrictions* and *capacity-controlled fares*. First, they designed discounts that had significant restrictions for purchase: They had to be purchased 30 days in advance of departure, were nonrefundable, and required a seven-day minimum stay. These restrictions were designed to prevent most business travelers from utilizing the new low fares. At the same time, American limited the number of discount seats sold on each flight: They *capacity-controlled* the fares. This combination provided the means to compete on price with the upstart airlines without damaging their core business-traveler revenues.

Initially, American's capacity controls were based on setting aside a fixed portion of seats on each flight for the new low-fare products. However, as they gained experience with its Super-Saver fares, American realized that not all flights were the same. Flights on different days and at different times had very different patterns of demand. A more intelligent approach was needed to realize the full potential of capacity-controlled discounts. American therefore embarked on the development of what became known as the *Dynamic Inventory Allocation and Maintenance Optimizer* system (DINAMO). These efforts on DINAMO represent, in many ways, the first large-scale RM system development in the industry.

DINAMO was implemented in full in January 1985 along with a new fare program entitled Ultimate Super-Saver Fares, which matched or undercut the lowest discount fares available in every market American served. DINAMO made all this possible. American could now be much more aggressive on price. It could announce low fares that spanned a large swath of individual flights, confident in its capability to accurately capacity-control the discounts on each individual departure. If a rival airline advertised a special fare in one of American's markets, American could immediately match the offer across the board, knowing that the DINAMO system would carefully control the availability of this fare on the thousands of departures affected by the price change. This feature of pricing aggressively and competitively at an aggregate, market level, while controlling capacity at a tactical, individual-departure level, still characterizes the practice of RM in the airline industry today.

The effect of this new capability was dramatic. PeopleExpress was especially hard-hit as American repeatedly matched or beat their prices in every market it served. PeopleExpress's annual profit fell from an all-time high in 1984 (the year prior to implementation of DINAMO) to a loss of over $160 million by 1986 (one year after DINAMO was implemented). It soon went bankrupt as a result of mounting losses, and in September 1986 the company was sold to Continental Airlines.

Donald Burr, CEO of PeopleExpress, summarized the reasons behind the company's failure (Cross [21]):

> We were a vibrant, profitable company from 1981 to 1985, and then we tipped right over into losing $50 million a month. We were still the same company. What changed was American's ability to do widespread Yield Management in every one of our markets. . . . We did a lot of things right. But we didn't get our hands around Yield Management and automation issues. . . . [If I were to do it again,] the number one priority on my list every day would be to see that my people got the best information technology tools. In my view, that's what drives airline revenues today more than any other factor—more than service, more than planes, more than routes.

This story was played out in similar fashion throughout the airline industry in the decades following deregulation, and airlines that did not have similar RM capabilities scrambled to get them.

As a result of this history, the practice of RM in the airline industry today is both pervasive and mature, and RM is viewed as critical to running a modern airline profitably. For example, American Airlines estimates that its RM practices generated $1.4 billion in additional incremental revenue over a three-year period starting around 1988 (Smith et al. [69]). Many other carriers also attribute similar improvements in their revenue due to RM.

## 1.4. Consequences of the Airline History

The intimate connection of RM to the airline industry is both a blessing and a curse for the field of RM. The blessing is that RM can point to a major industry in which the practice of RM is pervasive, highly developed, and enormously effective. Indeed, a large, modern airline today would just not be able to operate profitably *without* RM. By most estimates, the revenue gains from the use of RM systems are roughly comparable to many airlines'

total profitability in a good year (about 4% to 5% of revenues).[2] In addition, the scale and complexity of RM at major airlines is truly mind-boggling. Therefore, the airline success story validates both the economic importance of RM and the feasibility of executing it reliably in a complex business environment. This is the good-news story for the field from the airline experience.

The bad news—the curse if you will—of the strong association of RM with airlines is that it has created a certain myopia inside the field. Many practitioners and researchers view RM solely in airline-specific terms, and this has at times tended to create biases that have hampered both research and implementation efforts in other industries.

A second problem with the airline-specific association of RM is that airline pricing has something of a bad reputation among consumers. While on the one hand customers love the very low fares made possible by RM practices, the fact that fares are complex, are available one minute and gone the next, and can be drastically different for two people sitting side by side on the same flight, has led to a certain hostility toward the way airlines price. As a result, managers outside the industry are at times, quite naturally, somewhat reluctant to try RM practices for fear of engendering a similar hostile reaction among their customers. However, the reality is that, in most cases, applying RM does *not* involve radically changing the structure of pricing and sales practices; rather, it is a matter of making more intelligent decisions.

## 1.5. A Conceptual Framework for RM

So, if airlinelike conditions aren't strictly necessary for RM, then exactly where *does* it apply? A short answer is: in any business where tactical demand management is important and the technology and management culture exists to implement it. More specifically, the following conditions generally favor the application of RM techniques:

- *Customer heterogeneity.* If all customers value a product identically and exhibit similar purchase behavior, there is less potential to exploit variations in willingness to pay, variations in preference for different products, and variations of purchase behavior over time. Therefore, the more heterogeneity in customers, the more potential there is to exploit this heterogeneity strategically and tactically to improve revenues.
- *Demand variability and uncertainty.* The more demand varies over time (due to seasonalities, shocks, and so on) and the more uncertainty one has about future demand, the more difficult the demand-management decisions become. Hence, the potential to make bad decisions rises, and it becomes important to have sophisticated tools to evaluate the resulting complex trade-offs.
- *Production inflexibility.* Joint production constraints and costs complicate the demand-management problem. If a firm can "absorb" variations in *demand* easily and costlessly through variations in *supply*, then the complexity of managing demand diminishes; you just supply enough to meet demand. However, the more inflexible the production—the more delays involved in producing units, the more fixed costs or economies of scale involved in production, the more the switch-over costs, the more capacity constraints—the more difficult or costly it becomes to match demand variations with supply variations. As a result, inflexibility leads to more interaction in the demand management at different points in time, between different segments of customers, across different products of a product line, and across different channels of distribution. The complexity increases and the consequences of poor decisions become more acute. Hence, RM becomes more beneficial.
- *Data and information systems infrastructure.* To operationalize RM requires data to accurately characterize and model demand. It also requires systems to collect and store the

[2] Many skeptics point to Southwest Airlines as a counterexample, but Southwest does use RM systems. However, because its pricing structure is simpler than most other airlines, the use of RM is less obvious to consumers and casual observers.

data and to implement and monitor the resulting real-time decisions. In most industries it is usually feasible—in theory, at least—to collect and store demand data and automate demand decisions. However, attempting to apply RM in industries that do not have databases or transactions systems in place can be a time consuming, expensive, and risky proposition. RM, therefore, tends to be more suited to industries where transaction-processing systems are already employed as part of incumbent business processes.

• *Management culture.* RM is a technically complex and demanding practice. There is a risk, therefore, that a firm's management may simply not have sufficient familiarity with—or confidence in—science and technology to make implementing an RM system a realistic prospect. The culture of the firm may not be receptive to innovation or may value more intuitive approaches to problem solving. This is often due to the culture of the industry and its managers: their educational backgrounds, their professional experiences and responsibilities en route to leadership positions, and the skills required to succeed in the industry.

## 1.6. Industry Adopters Beyond the Airlines

What do these conditions imply for adopters of RM technology? The production-inflexibility characteristics of airlines are shared by many other service industries, such as hotels, cruise ship lines, car rental companies, theaters and sporting venues, and radio/TV broadcasters, to name a few. Indeed, RM is strongly associated with service industries.

Retailers have recently begun to adopt RM, especially in the fashion apparel, consumer electronics, and toy sectors. Retail demand is highly volatile and uncertain, consumers' valuations change rapidly over time, and with short selling seasons and long production and distribution lead times, supply is quite inflexible. On the technology front, the introduction of bar codes and point-of-sale (POS) technology has resulted in a high degree of automation of sales transactions for most major retailers.

The energy sector has been a recent adopter of RM methods as well, principally in the area of managing the sale of pipeline capacity for gas transportation. Again, energy demands are volatile and uncertain, and the technology for generating and transmitting electricity and gas can be inflexible. Also, thanks to deregulation in the industry, there has been a lot of experimentation and innovation in the pricing practices of energy, gas, and transmission markets.

Manufacturing is potentially a vast market for RM methods, although to date relatively few instances of the practice have been documented. Enterprise resource planning (ERP), supply chain management (SCM), and customer relationship management (CRM) systems are commonplace in the industry, and most manufacturers have huge amounts of data and heavily automated business processes, which could form the foundations for RM. For example, in the auto industry, Ford Motor Corporation recently completed a high-profile implementation of RM technology (Coy [20]).

What about future adopters of RM? Given the criteria outlined above, one can argue that many industries are potential candidates. Almost all businesses must deal with demand variability, uncertainty, and customer heterogeneity. Most are subject to some sort of supply or production inflexibility. Finally, thanks largely to the wave of enterprise software and e-commerce innovation of late, many firms have now automated their business processes. All of these factors bode well for the future of RM.

Nevertheless, as with any technological and business-practice innovation, the case for RM ultimately boils down to a cost-benefit analysis for each individual firm. For some, the potential benefit will simply never justify the costs of implementing RM systems and business processes. However, we believe that for the majority of firms, RM will eventually be justified once the technology and methodology in their industry matures. Indeed, the history of RM in industries such as airlines, hotels, and retail suggests that once the technology gains a foothold in an industry, it spreads quite rapidly. As a result, we would not be surprised to see RM systems (or systems performing RM functions under a different label) become as ubiquitous as ERP, SCM, and CRM systems are today.

### 1.7. Overview of Topics

The number of topics the field spans is too large to cover adequately in a single chapter like this. Our book, *The Theory and Practice of Revenue Management* (Kluwer 2004) provides in-depth coverage of both quantity- and price-based RM as well as supporting topics such as demand modeling, economics, forecasting, and system implementation. Here, we only give a sample of two such topics: single-resource capacity control and deterministic dynamic pricing problems. While incomplete, this sampling serves to illustrate the types of modeling ideas and solution methods found in revenue management.

## 2. Single-Resource Capacity Control

In this section, we examine some basic results on the problem of quantity-based revenue management for a single resource; specifically, optimally allocating capacity of a resource to different classes of demand. Two prototypical examples are controlling the sale of different fare classes on a single flight leg of an airline and the sale of hotel rooms for a given date at different rate classes. This is to be contrasted with multiple-resource—or network—problems, in which customers require a bundle of different resources (such as two connecting flights or a sequence of nights at the same hotel). In reality, many quantity-based RM problems are network RM problems, but in practice, they are still frequently solved as a collection of single-resource problems (treating the resources independently). For this reason, it is important to study single-resource RM models. Moreover, single-resource models are useful as building blocks in heuristics for the network case.

We assume that the firm sells its capacity in $n$ distinct classes[3] that require the same resource. In the airline and hotel context, these classes represent different discount levels with differentiated sale conditions and restrictions. In the early parts of this section, we assume that these products appeal to distinct and mutually exclusive segments of the market: The conditions of sale segment the market perfectly into $n$ segments—one for each class. Customers in each segment are eligible for or can afford only the class corresponding to their segment. Later in the section, we look at models that do not assume that customers are perfectly segmented, but instead that they choose among the $n$ classes.

The units of capacity are assumed to be homogeneous, and customers demand a single unit of capacity for the resource. The central problem of this section is how to optimally allocate the capacity of the resource to the various classes. This allocation must be done dynamically as demand materializes and with considerable uncertainty about the quantity or composition of future demand. The remainder of the section focuses on various models and methods for making these capacity-allocation decisions.

### 2.1. Types of Controls

In the travel industry, reservation systems provide different mechanisms for controlling availability. These mechanisms are usually deeply embedded in the software logic of the reservation system and, as a result, can be quite expensive and difficult to change. Therefore, the control mechanisms chosen for a given implementation are often dictated by the reservation system. Here, we focus on the control mechanisms themselves.

**2.1.1. Booking Limits.** *Booking limits* are controls that limit the amount of capacity that can be sold to any particular class at a given point in time. For example, a booking limit of 18 on class 2 indicates that at most 18 units of capacity can be sold to customers in class 2. Beyond this limit, the class would be "closed" to additional class 2 customers. This limit of 18 may be less than the physical capacity. For example, we might want to protect capacity for future demand from class 1 customers.

[3] In the case of airlines, these are called *fare classes*. Terms like *rate products*, *rate classes*, *revenue classes*, *booking classes*, and *fare products* are also used. We shall use the generic term *class* in this section.

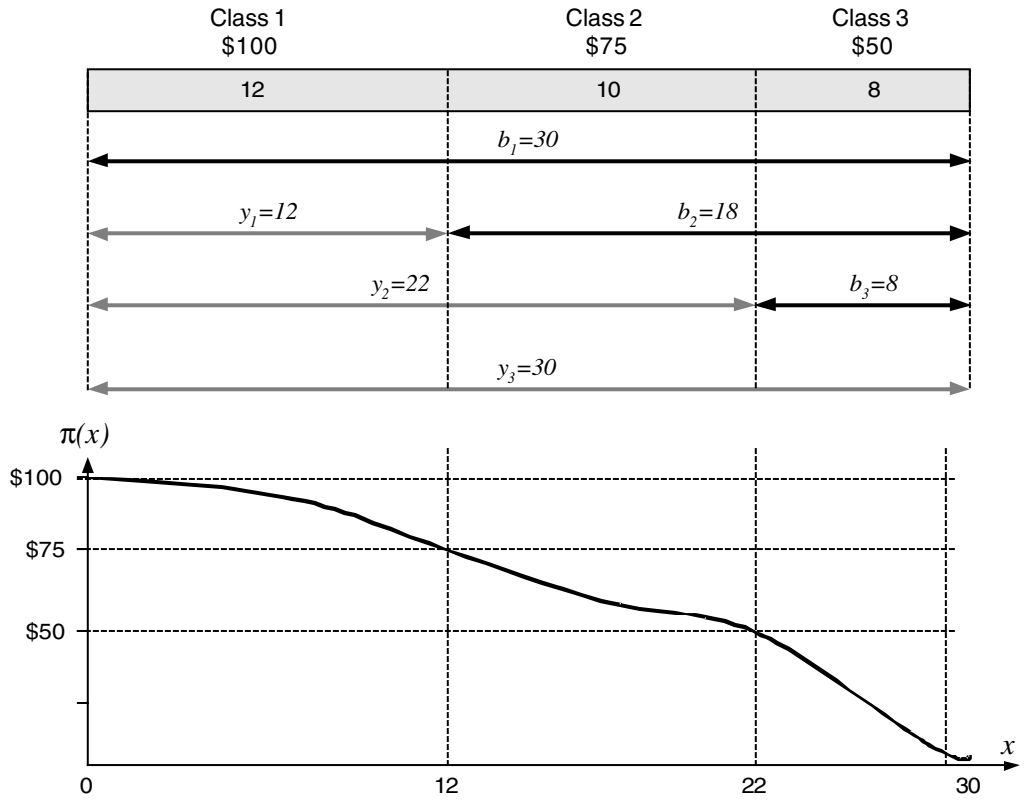
Booking limits are either *partitioned* or *nested*: A *partitioned booking limit* divides the available capacity into separate blocks (or *buckets*)—one for each class—that can be sold only to the designated class. For example, with 30 units to sell, a partitioned booking limit may set a booking limit of 12 units for class 1, 10 units for class 2, and 8 units for class 3. If the 12 units of class 1 capacity are used up, class 1 would be closed regardless of how much capacity is available in the remaining buckets. This could be undesirable if class 1 has higher revenues than do classes 2 and 3 and the units allocated to class 1 are sold out.

With a *nested booking limit*, the capacity available to different classes overlaps in a hierarchical manner—with higher-ranked classes having access to all the capacity reserved for lower-ranked classes (and perhaps more). Let the nested booking limit for class $j$ be denoted $b_j$. Then $b_j$ is the maximum number of units of capacity we are willing to sell to classes $j$ and lower. So, in Figure 1, the nested booking limit on class 1 and lower (all classes) would be $b_1 = 30$ (the entire capacity), the nested booking limit on classes 2 and 3 combined would be $b_2 = 18$, and the nested booking limit on class 3 alone would be $b_3 = 8$. We would accept at most 30 bookings for classes 1, 2, and 3; at most 18 for classes 2 and 3 combined; and at most 8 for class 3 customers. Effectively, this logic simply allows any capacity "left over" after selling to low classes to become available for sale to higher classes.

Nesting booking limits in this way avoids the problem of capacity being simultaneously unavailable for a high class yet available for lower classes. Most reservations systems that use booking-limit controls quite sensibly use nested rather than partitioned booking limits for this reason.

**2.1.2. Protection Levels.** A *protection level* specifies an amount of capacity to reserve (protect) for a particular class or set of classes. Again, protection levels can be *nested* or *partitioned*. A partitioned protection level is trivially equivalent to a partitioned booking

FIGURE 1. The relationship between booking limits $b_j$, protection levels $y_j$, and bid prices $\pi(x)$.



limit; a booking limit of 18 on class 2 sales is equivalent to protecting 18 units of capacity for class 2.

In the nested case, protection levels are again defined for sets of classes—ordered in a hierarchical manner according to class order. Suppose class 1 is the highest class, class 2 the second highest, and so on. Then the protection level $j$, denoted $y_j$, is defined as the amount of capacity to save for classes $j, j-1, \ldots, 1$ combined—that is, for classes $j$ and higher (in terms of class order). Continuing our example, we might set a protection level of 12 for class 1 (meaning 12 units of capacity would be protected for sale only to class 1), a protection level of 22 for classes 1 and 2 combined, and a protection level of 30 for classes 1, 2, and 3 combined (although frequently no protection level is specified for this last case because it is clear that all the capacity is available to at least one of the classes).

Figure 1 shows the relationship between protection levels and booking limits. The booking limit for class $j$, $b_j$ is simply the capacity minus the protection level for classes $j-1$ and higher. That is,

$$b_j = C - y_{j-1}, \quad j = 2, \ldots, n,$$

where $C$ is the capacity. For convenience, we define $b_1 = C$ (the highest class has a booking limit equal to the capacity) and $y_n = C$ (all classes combined have a protection level equal to capacity).

**2.1.3. Standard vs. Theft Nesting.** The standard process for using booking limits or nested protection levels proceeds as follows. Starting with $C$ units of capacity, we begin receiving bookings. A booking for class $j$ is accepted provided (1) there is capacity remaining and (2) the total number of requests accepted for class $j$ to date is less than the booking limit $b_j$ (equivalently, the current capacity remaining is more than the protection level $y_{j-1}$ for classes higher than $j$). This is called *standard nesting*, and it is the most natural and common way to implement nested-capacity controls.

Another alternative, which is less prevalent although still encountered occasionally in practice, is called *theft nesting*. In theft nesting, a booking in class $j$ not only reduces the allocation for class $j$, but also "steals" from the allocation of all lower classes. Therefore, when we accept a request for class $j$, not only is the class $j$ allocation reduced by one, but so are the allocations for classes $j+1, j+2, \ldots, n$. This is equivalent to keeping $y_j$ units of capacity protected for *future* demand from class $j$ and higher. In other words, even though we just accepted a request for class $j$, under theft nesting we continue to reserve $y_j$ units for class $j$ and higher, and to do so requires reducing the allocation for classes $j+1, j+2, \ldots, n$. Under standard nesting, in contrast, when we accept a request from class $j$ we effectively reduce by one the capacity we protect for future demand from class $j$ and higher.

The rationale for standard nesting is that the capacity protected for, say, class 1 is based on a forecast of future demand for class 1. Once we observe some demand for class 1, we then *reduce* our estimate of future demand—and hence the capacity we protect for class 1. Standard nesting does this by reducing the capacity protected for future class 1 demand on a one-for-one basis after each arriving request is accepted (and similarly for other classes as well). To illustrate, suppose in our example that demand for class 1 is deterministic and equal to the protection level $y_1 = 12$. Then if we receive 5 requests for class 1, we know for certain that future demand for class 1 will be only 7, and hence that it makes sense to reduce the capacity we protect for future demand from 12 to 7, which is precisely what standard nesting does. Theft nesting, in contrast, intuitively corresponds to an assumption of "memorylessness" in demand. In other words, it assumes the demand to date for class 1 does not affect our estimate of *future* demand for class 1. Therefore, we continue to protect $y_1$ units of capacity for class 1 (and hence must reduce the allocation for classes $2, 3, \ldots, n$).

The two forms of nesting are in fact equivalent if demand arrives strictly in low-to-high class order; that is, the demand for class $n$ arrives first, followed by the demand for class $n-1$,

and so on.[4] This is what the standard (static) single-resource models assume, so for these static models, the distinction is not important. However, in practice demand rarely arrives in low-to-high order, and the choice of standard versus theft nesting matters. With mixed order of arrivals, theft nesting protects more capacity for higher classes (equivalently, allocates less capacity to lower classes). Again, however, standard nesting is the norm in RM practice.

**2.1.4. Bid Prices.** What distinguishes bid-price controls from both booking limits and protection levels is that they are revenue-based, rather than class-based, controls. Specifically, a bid-price control sets a threshold price (which may depend on variables such as the remaining capacity or time), such that a request is accepted if its revenue exceeds the threshold price and rejected if its revenue is less than the threshold price. Bid-price controls are, in principle, simpler than booking-limit or protection-level controls because they require storing only a single threshold value at any point in time—rather than a set of capacity numbers, one for each class. However, to be effective, bid prices must be updated after each sale—and possibly also with time as well—and this typically requires storing a table of bid-price values indexed by the current available capacity, current time, or both.

Figure 1 shows how bid prices can be used to implement the same nested-allocation policy as booking limits and protection levels. The bid price $\pi(x)$ is plotted as a function of the remaining capacity $x$. When there are 12 or fewer units remaining, the bid price is over \$75 but less than \$100, so only class 1 demand is accepted. With 13 to 22 units remaining, the bid price is over \$50 but less than \$75, so only classes 1 and 2 are accepted. With more than 22 units of capacity available, the bid price drops below \$50, so all three classes are accepted.

Bid-price control is criticized by some as being "unsafe"—the argument being that having a threshold price as the only control means that the RM system will sell an unlimited amount of capacity to any class whose revenues exceed the bid-price threshold. However, this is true only if the bid price is not updated. As shown in Figure 1, if the bid price is a function of the current remaining capacity, then it performs exactly like a booking limit or protection level, closing off capacity to successively higher classes as capacity is consumed. Without this ability to make bid prices a function of capacity, however, a simple static threshold is indeed a somewhat dangerous form of control.

One potential advantage of bid-price controls is their ability to discriminate based on revenue rather than class. Often a number of products with different prices are booked in a single class. RM systems then use an average price as the price associated with a class. However, if actual revenue information is available for each request, then a bid-price control can selectively accept only the higher revenue requests in a class, whereas a control based on class designation alone can only accept or reject all requests of a class. Of course, if the exact revenue is not observable at the time of reservation, then this advantage is lost.

## 2.2. Displacement Cost

While the mathematics of optimal capacity controls can become complex, the overriding logic is simple. First, capacity should be allocated to a request if and only if its revenue is greater than the value of the capacity required to satisfy it. Second, the value of capacity should be measured by its (expected) *displacement cost*—or *opportunity cost*—which is the expected loss in future revenue from using the capacity now rather than reserving it for future use.

Theoretically, the displacement-cost idea is captured by using a *value function*, $V(x)$, that measures the optimal expected revenue as a function of the remaining capacity $x$. The displacement cost then is the difference between the value function at $x$ and the value function at $x-1$, or $V(x)-V(x-1)$. Much of the theoretical analysis of the capacity controls boils down to analyzing this value function, but conceptually, the logic is simply to compare revenues to displacement costs to make the accept or deny decision.

[4] It is easy to convince oneself of this fact by tracing out the accept/deny decisions under both forms of nesting, and doing so is an instructive exercise.

## 2.3. Static Models

In this section, we examine one of the first models for quantity-based RM, the so-called *static*[5] single-resource models.

The static model makes several assumptions that are worth examining in some detail. The first is that demand for the different classes arrives in nonoverlapping intervals in the order of increasing prices of the classes.[6] In reality, demand for the different classes may overlap in time. However, the nonoverlapping-intervals assumption is a reasonable approximation (for example, advance-purchase discount demand typically arrives before full-fare coach demand in the airline case). Moreover, the optimal controls that emerge from the model can be applied—at least heuristically—even where demand comes in arbitrary order (using either bid prices or the nesting policies, for example). As for the strict low-before-high assumption, this represents something of a worst-case scenario; for instance, if high-revenue demand arrives before low-revenue demand, the problem is trivial because we simply accept demand first come, first serve.

The second main assumption is that the demands for different classes are independent random variables. Largely, this assumption is made for analytical convenience, because to deal with dependence in the demand structure would require introducing complex state variables on the history of observed demand. We can make some justification for the assumption by appealing to the forecast inputs to the model. That is, to the extent that there are systematic factors affecting *all* demand classes (such as seasonalities), these are often reflected in the forecast and become part of the *explained* variation in demand in the forecasting model (for example, as the differences in the forecasted means and variance on different days). The randomness in the single-resource model is then only the residual, unexplained variation in demand. So, for example, the fact that demand for all classes may increase on peak flights does not in itself cause problems, provided the increase is predicted by the forecasting method. Still, one has to worry about possible residual dependence in the *unexplained variation* in demand, and this is a potential weakness of the independence assumption.

A third assumption is that demand for a given class does not depend on the capacity controls; in particular, it does not depend on the availability of other classes. Its only justification is if the multiple restrictions associated with each class are so well designed that customers in a high-revenue class will not buy down to a lower class, and if the prices are so well separated that customers in a lower class will not buy up to a higher class if the lower class is closed. However, neither is really true in practice. There is considerable porousness (imperfect segmentation) in the design of the restrictions, and the price differences between the classes are rarely that dispersed. The assumption that demand does not depend on the capacity controls is therefore a weakness, although in §2.5 we look at models that handle imperfect segmentation.

Fourth, the static model suppresses many details about the demand and control process within each of the periods. This creates a potential source of confusion when relating these models to actual RM systems. In particular, the static model assumes that an aggregate quantity of demand arrives in a single stage and the decision is simply how much of this demand to accept. However, in a real reservation system, we typically observe demand sequentially over time, or it may come in batch downloads. The control decision has to be made knowing only the demand observed to date and is usually implemented in the form of prespecified controls uploaded to the reservation system. These details are essentially ignored in the static model. However, fortunately (and perhaps surprisingly), the form of the

[5] The term *static* is somewhat of a misnomer here because demand does arrive sequentially over time, albeit in stages ordered from low-revenue to high-revenue demand. However, this term is now standard and helps distinguish this class of models from *dynamic* models that allow arbitrary arrival orders.

[6] Robinson [67] generalizes the static model to the case where demand from each class arrives in nonoverlapping intervals, but the order is not necessarily from low to high revenue.

optimal control is not sensitive to this assumption and can be applied quite independently of how the demand is realized within a period (all at once, sequentially, or in batches). The simplicity and robustness of the optimal control is, in fact, a central result of the theory for this class of models.

A fifth assumption of the model is that either there are no groups, or if there are group bookings, they can be partially accepted.

Finally, the static models assume risk neutrality. This is a reasonable assumption in practice, because a firm implementing RM typically makes such decisions for a large number of products sold repeatedly (for example, daily flights, daily hotel room stays, and so on). Maximizing the average revenue, therefore, is what matters in the end. While we do not cover this case here, some researchers have recently analyzed the single-resource problem with risk-averse decision makers (Feng and Xiao [30]).

We start with the simple two-class model to build some basic intuition, and then examine the more general $n$-class case.

**2.3.1. Littlewood's Two-Class Model.** The earliest single-resource model for quantity-based RM is due to Littlewood [55]. The model assumes two product classes, with associated prices $p_1 > p_2$. The capacity is $C$, and we assume there are no cancellations or overbooking. Demand for class $j$ is denoted $D_j$, and its distribution is denoted by $F_j(\cdot)$. Demand for class 2 arrives first. The problem is to decide how much class 2 demand to accept before seeing the realization of class 1 demand.

The two-class problem is similar to the classic *newsboy problem* in inventory theory, and the optimal decision can be derived informally using a simple marginal analysis: Suppose that we have $x$ units of capacity remaining and we receive a request from class 2. If we accept the request, we collect revenues of $p_2$. If we do not accept it, we will sell unit $x$ (the marginal unit) at $p_1$ if and only if demand for class 1 is $x$ or higher. That is, if and only if $D_1 \geq x$. Thus, the expected gain from reserving the $x$th unit for class 1 (the *expected marginal value*) is $p_1 P(D_1 \geq x)$. Therefore, it makes sense to accept a class 2 request as long as its price exceeds this marginal value or, equivalently, if and only if

$$p_2 \geq p_1 P(D_1 \geq x). \tag{1}$$

Note that the right-hand side of (1) is decreasing in $x$. Therefore, there will be an optimal protection level, denoted $y_1^*$, such that we accept class 2 if the remaining capacity exceeds $y_1^*$ and reject it if the remaining capacity is $y_1^*$ or less. Formally, $y_1^*$ satisfies

$$p_2 < p_1 P(D_1 \geq y_1^*) \quad \text{and} \quad p_2 \geq p_1 P(D_1 \geq y_1^* + 1).$$

If a continuous distribution $F_1(x)$ is used to model demand (as is often the case), then the optimal protection level $y_1^*$ is given by the simpler expressions

$$p_2 = p_1 P(D_1 > y_1^*), \quad \text{equivalently,} \quad y_1^* = F_1^{-1}\left(1 - \frac{p_2}{p_1}\right), \tag{2}$$

which is known as *Littlewood's rule*. Setting a protection level of $y_1^*$ for class 1 according to Littlewood's rule is an optimal policy. Equivalently, setting a booking limit of $b_2^* = c - y_1^*$ on class 2 demand is optimal. Alternatively, we can use a bid-price control with the bid price set at $\pi(x) = p_1 P(D_1 > x)$.

We omit a rigorous proof of Littlewood's rule because it is a special case of a more general result proved below. However, to gain some insight into it, consider the following example:

**Example 1.** Suppose $D_1$ is normally distributed with mean $\mu$ and standard deviation $\sigma$. Then by Littlewood's rule, $F_1(y_1^*) = 1 - p_2/p_1$, which implies that the optimal protection level can be expressed as

$$y_1^* = \mu + z\sigma,$$

where $z = \Phi^{-1}(1 - p_2/p_1)$ and $\Phi(\cdot)^{-1}$ denotes the inverse of the standard normal c.d.f. Thus, we reserve enough capacity to meet the mean demand for class 1, $\mu$, plus or minus a factor that depends both on the revenue ratio and the demand variation $\sigma$. If $p_2/p_1 > 0.5$, the optimal protection level is less than the mean demand; and if $p_2/p_1 < 0.5$, it is greater than the mean demand. In general, the lower the ratio $p_2/p_1$, the more capacity we reserve for class 1. This makes intuitive sense because we should be willing to take very low prices only when the chances of selling at a high price are lower.

**2.3.2. $n$-Class Models.** We next consider the general case of $n > 2$ classes. Again, we assume that demand for the $n$ classes arrives in $n$ stages, one for each class, with classes arriving in increasing order of their revenue values. Let the classes be indexed so that $p_1 > p_2 > \cdots > p_n$. Hence, class $n$ (the lowest price) demand arrives in the first stage (stage $n$), followed by class $n-1$ demand in stage $n-1$, and so on, with the highest price class (class 1) arriving in the last stage (stage 1). Because there is a one-to-one correspondence between stages and classes, we index both by $j$. Demand and capacity are most often assumed to be discrete, but occasionally we model them as continuous variables when it helps simplify the analysis and optimality conditions.

*Dynamic programming formulation.* This problem can be formulated as a dynamic program in the stages (equivalently, classes), with the remaining capacity $x$ being the state variable. At the start of each stage $j$, the demand $D_j, D_{j-1}, \ldots, D_1$ has not been realized. Within stage $j$, the model assumes that the following sequence of events occurs:

(1) The realization of the demand $D_j$ occurs, and we observe its value.

(2) We decide on a quantity $u$ of this demand to accept. The amount accepted must be less than the capacity remaining, so $u \le x$. The optimal control $u^*$ is therefore a function of the stage $j$, the capacity $x$, and the demand $D_j$, $u^* = u^*(j, x, D_j)$, although we often suppress this explicit dependence on $j$, $x$, and $D_j$ in what follows.

(3) The revenue $p_j u$ is collected, and we proceed to the start of stage $j-1$ with a remaining capacity of $x - u$.

This sequence of events is assumed for analytical convenience; we derive the optimal control $u^*$ "as if" the decision on the amount to accept is made *after* knowing the value of demand $D_j$. In reality, of course, demand arrives sequentially over time, and the control decision has to be made *before* observing all the demand $D_j$. However, it turns out that optimal decisions do not use the prior knowledge of $D_j$ as we show below. Hence, the assumption that $D_j$ is known is not restrictive.

Let $V_j(x)$ denote the value function at the start of stage $j$. Once the value $D_j$ is observed, the value of $u$ is chosen to maximize the current stage $j$ revenue plus the revenue to go, or

$$p_j u + V_{j-1}(x - u),$$

subject to the constraint $0 \le u \le \min\{D_j, x\}$. The value function entering stage $j$, $V_j(x)$, is then the expected value of this optimization with respect to the demand $D_j$. Hence, the Bellman equation is[7]

$$V_j(x) = E\left[\max_{0 \le u \le \min\{D_j, x\}} \{p_j u + V_{j-1}(x - u)\}\right], \tag{3}$$

with boundary conditions

$$V_0(x) = 0, \quad x = 0, 1, \ldots, C.$$

The values $u^*$ that maximize the right-hand side of (3) for each $j$ and $x$ form an optimal control policy for this model.

[7] Readers familiar with dynamic programming may notice that this Bellman equation is of the form $E[\max\{\cdot\}]$ and not $\max E[\cdot]$ as in many standard texts. Essentially, however, the $\max E[\cdot]$ form can be recovered by considering the demand $D_j$ to be a state variable along with $x$. While the two forms can be shown to be equivalent, the $E[\max\{\cdot\}]$ is simpler to work with in many RM problems. In our case, this leads to the modeling assumption that we optimize "as if" we observed $D_j$.

*Optimal policy: Discrete demand and capacity.* We first consider the case where demand and capacity are discrete. To analyze the form of the optimal control in this case, define

$$\Delta V_j(x) \equiv V_j(x) - V_j(x-1).$$

$\Delta V_j(x)$ is the *expected marginal value of capacity* at stage $j$—the expected incremental value of the $x$th unit of capacity. A key result concerns how these marginal values change with capacity $x$ and the stage $j$:

**Proposition 1.** *The marginal values $\Delta V_j(x)$ of the value function $V_j(x)$ defined by* (3) *satisfy $\forall x, j$:*

(i) $\Delta V_j(x+1) \le \Delta V_j(x)$ *and*
(ii) $\Delta V_{j+1}(x) \ge \Delta V_j(x)$.

That is, at a given stage $j$ the marginal value is decreasing in the remaining capacity, and at a given capacity level $x$ the marginal value increases in the number of stages remaining. These two properties are intuitive and greatly simplify the control. To see this, consider the optimization problem at stage $j+1$. From (3) and the definition of $\Delta V_j(x)$, we can write

$$V_{j+1}(x) = V_j(x) + E\left[\max_{0\le u\le \min\{D_{j+1},\,x\}}\left\{\sum_{z=1}^{u}(p_{j+1} - \Delta V_j(x+1-z))\right\}\right],$$

where we take the summation above to be empty if $u=0$. Because $\Delta V_j(x)$ is decreasing in $x$ by Proposition 1(i), it follows that the terms in the sum $p_{j+1} - \Delta V_j(x+1-z)$ are decreasing in $z$. Thus, it is optimal to increase $u$ (keep adding terms) until the terms $p_{j+1} - \Delta V_j(x+1-z)$ become negative or the upper bound $\min\{D_{j+1}, x\}$ is reached, whichever comes first.

The resulting optimal control can be expressed in terms of optimal protection levels $y_j^*$ for $j, j-1, \ldots, 1$ (class $j$ and higher in the revenue order) by

$$y_j^* \equiv \max\{x : p_{j+1} < \Delta V_j(x)\}, \quad j = 1, \ldots, n-1. \tag{4}$$

(Recall the optimal protection level $y_n^* \equiv C$ by convention.) The optimal control at stage $j+1$ is then

$$u^*(j+1, x, D_{j+1}) = \min\{(x - y_j^*)^+, D_{j+1}\}, \tag{5}$$

where the notation $z^+ = \max\{0, x\}$ denotes the positive part of $z$. The quantity $(x - y_j^*)^+$ is the remaining capacity in excess of the protection level, which is the maximum capacity we are willing to sell to class $j+1$. The situation is shown in Figure 2.

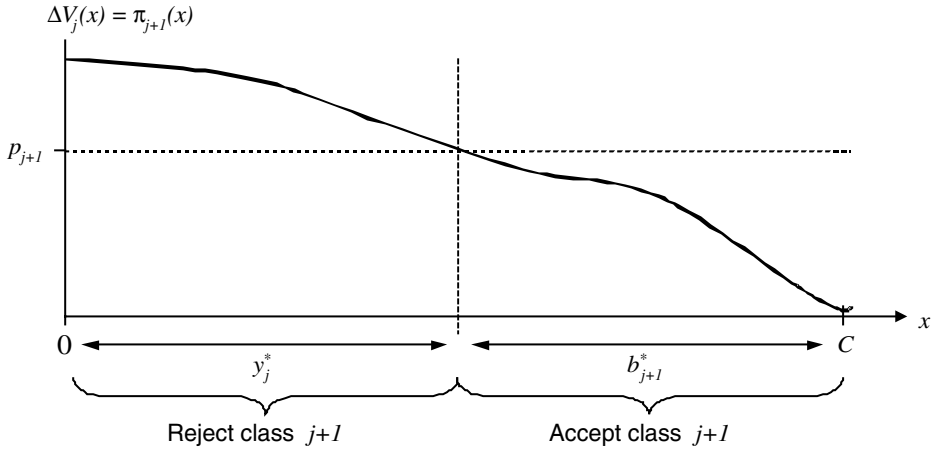
In practice, we can simply post the protection level $y_j^*$ in a reservation system and accept requests first come, first serve until the capacity threshold $y_j^*$ is reached or the stage ends, whichever comes first. Thus, the optimal protection-level control at stage $j+1$ requires no information about the demand $D_{j+1}$, yet it produces the same optimal decision "as if" we knew $D_{j+1}$ exactly at the start of stage $j+1$. The reason for this is that knowledge of $D_{j+1}$ does not affect the future value of capacity, $V_j(x)$. Deciding to accept or reject each request simply involves comparing current revenues to the marginal cost of capacity, and this comparison does not depend on how many stage-$(j+1)$ requests there are in total.

Proposition 1(ii) implies the nested protection structure

$$y_1^* \le y_2^* \le \cdots \le y_n^*.$$

This fact is easily seen from Figure 2. If $p_{j+1}$ increases with $j$ and the curve $\Delta V_j(x)$ decreases with $j$, then the optimal protection level $y_j^*$ will shift to the left (decrease). Together, this ordering produces the nested protection-level structure.

FIGURE 2. The optimal protection level $y_j^*$ in the static model.




One can also use booking limits in place of protection levels to achieve the same control. Optimal nested booking limits are defined by

$$b_j^* \equiv C - y_{j-1}^*, \quad j = 2, \ldots, n, \tag{6}$$

with $b_1^* \equiv C$. The optimal control in stage $j+1$ is then to accept

$$u^*(j+1, x, D_{j+1}) = \min\{(b_{j+1} - (C-x))^+, D_{j+1}\}.$$

Note that $C - x$ is the total capacity sold prior to stage $j+1$ and $b_{j+1}$ is the booking limit for class $j+1$, so $(b_{j+1} - (C-x))^+$ is the remaining capacity available for class $j+1$. The optimal booking limit is also shown in Figure 2.

Finally, the optimal control can also be implemented through a table of bid prices. Indeed, if we define the stage $j+1$ bid price by

$$\pi_{j+1}(x) \equiv \Delta V_j(x), \tag{7}$$

then the optimal control is

$$u^*(j+1, x, D_{j+1}) = \begin{cases} 0 & \text{if } p_{j+1} < \pi_{j+1}(x) \\ \max\{z : p_{j+1} \geq \pi_{j+1}(x-z)\} & \text{otherwise.} \end{cases}$$

In words, we accept the $z$th request in stage $j+1$ if the price $p_{j+1}$ exceeds the bid price value $\pi_{j+1}(x-z)$ of the $z$th unit of capacity that is allocated. In practice, we can store a table of bid prices and process requests by sequentially comparing the price of each product to the table values corresponding to the remaining capacity.

We summarize these results in the following theorem:

**Theorem 1.** *For the static model defined by* (3)*, the optimal control can be achieved using either*

(i) *nested protection levels defined by* (4)*,*
(ii) *nested booking limits defined by* (6)*, or*
(iii) *bid-price tables defined by* (7)*.*

*Optimality conditions for continuous demand.* Next, consider the case where capacity is continuous and demand at each stage has a continuous distribution. In this case, the dynamic program is still given by (3); however, $D_j$, $x$, and $u$ are now continuous quantities. The analysis of the dynamic program is slightly more complex than it is in the discrete-demand case, but many of the details are quite similar. Hence, we briefly describe only the key differences.

The main change is that the marginal value $\Delta V_j(x)$ is now replaced by the derivative of $V_j(x)$ with respect to $x$, $\frac{\partial}{\partial x}V_j(x)$. This derivative is still interpreted as the marginal expected value of capacity, and an argument nearly identical to that in the proof of Proposition 1 shows that the marginal value $\frac{\partial}{\partial x}V_j(x)$ is decreasing in $x$ (equivalently, $V_j(x)$ is concave in $x$).

Therefore, the optimal control in stage $j+1$ is to keep increasing $u$ (keep accepting demand) as long as

$$p_{j+1} \geq \frac{\partial}{\partial x}V_j(x-u)$$

and to stop accepting once this condition is violated or the demand $D_{j+1}$ is exhausted, whichever comes first. Again, this decision rule can be implemented with optimal protection levels, defined by

$$y_j^* \equiv \max\left\{x : p_{j+1} < \frac{\partial}{\partial x}V_j(x)\right\}, \quad j = 1, \ldots, n-1.$$

One of the chief virtues of the continuous model is that it leads to simplified expressions for the optimal vector of protection levels $\mathbf{y}^* = (y_1^*, \ldots, y_n^*)$. We state the basic result without proof (see Brumelle and McGill [17] for a proof).

First, for an arbitrary vector of protection levels $\mathbf{y}$ and vector of demands $\mathbf{D} = (D_1, \ldots, D_n)$, define the following $n-1$ *fill events*

$$B_j(\mathbf{y}, \mathbf{D}) \equiv \{D_1 > y_1, D_1 + D_2 > y_2, \ldots, D_1 + \cdots + D_j > y_j\}, \quad j = 1, \ldots, n-1. \tag{8}$$

$B_j(\mathbf{y}, \mathbf{D})$ is the event that demand to come in stages $1, 2, \ldots, j$ exceeds the corresponding protection levels. A necessary and sufficient condition for $\mathbf{y}^*$ to be an optimal vector of protection levels is that it satisfy the $n-1$ equations

$$P(B_j(\mathbf{y}^*, \mathbf{D})) = \frac{p_{j+1}}{p_1}, \quad j = 1, 2, \ldots, n-1. \tag{9}$$

That is, the $j$th fill event should occur with probability equal to the ratio of class $(j+1)$ revenue to class 1 revenue. As it should, this reduces to Littlewood's rule (2) in the $n = 2$ case, because $P(B_1(\mathbf{y}^*, \mathbf{D})) = P(D_1 > y_1^*) = p_2/p_1$.

Note that

$$B_j(\mathbf{y}, \mathbf{D}) = B_{j-1}(\mathbf{y}, \mathbf{D}) \cap \{D_1 + \cdots + D_j > y_j\},$$

so the event $B_j(\mathbf{y}, \mathbf{D})$ can occur only if $B_{j-1}(\mathbf{y}, \mathbf{D})$ occurs. Also, if $y_j = y_{j-1}$ then $B_j(\mathbf{y}, \mathbf{D}) = B_{j-1}(\mathbf{y}, \mathbf{D})$. Thus, if $p_j < p_{j-1}$, we must have $y_j^* > y_{j-1}^*$ to satisfy (9). Thus, the optimal protection levels are strictly increasing in $j$ if the revenues are strictly decreasing in $j$.

**2.3.3. Heuristics.** As we have seen, computing optimal controls for the static single-resource model is not particularly difficult. Despite this fact, exact optimization models are not widely used in practice. Indeed, most single-resource airline RM systems use one of several heuristics to compute booking limits and protection levels.

There are two main reasons for this state of affairs. The first is simply a case of practice being one step ahead of the underlying theory. As mentioned, in the airline industry the practice of using capacity controls to manage multiple classes quickly gained popularity following deregulation in the mid-1970s. However, this predates the theory of optimal controls

by more than a decade. The only known optimal controls in the 1970s were Littlewood's results for the two-class problem. As a result, heuristics were developed for the general $n$-class problem. During the decade following deregulation, RM software embedded these heuristics, and people grew accustomed to thinking in terms of them. The inertia generated from this early use of the heuristics is one reason for their continued popularity today.

Heuristics are also widely used because they are simpler to code, quicker to run, and generate revenues that in many cases are close to optimal. Indeed, many practitioners in the airline industry simply believe that even the modest effort of computing optimal controls is not worth the benefit they provide in improved revenue performance. Proponents of heuristics argue that the potential improvement from getting better revenue data and improving demand forecasts swamps the gains from using optimal controls—reflecting the philosophy that it is better to be "approximately right" than it is to be "precisely wrong."

While these points are well taken, such criticisms are somewhat misdirected. For starters, using optimal controls does not mean one has to give up on improvements in other areas, such as forecasting. These activities are not mutually exclusive, although an understaffed development group might very well consider refining optimization modules a low-priority task. Still, given the very modest cost of coding and computing optimal controls, the strong objections to the use of optimal controls are often not entirely rational.

Regardless of one's view on the use of heuristics, it is important to understand them. They remain widely used in practice and can also help develop useful intuition.

We next look at the two most popular heuristics: EMSR-a and EMSR-b, both of which are attributed to Belobaba [3, 4, 5]. Both heuristics are based on the $n$-class, static, single-resource model defined above in §2.3. They differ only in how they approximate the problem. Static model assumptions apply: Classes are indexed so that $p_1 > p_2 > \cdots > p_n$, $F_j(x)$ denotes the c.d.f. of class $j$ demand, and low-revenue demand arrives before high-revenue demand in stages that are indexed by $j$ as well. Moreover, for ease of exposition we assume that capacity and demand are continuous and that the distribution functions $F_j(x)$, $j = 1, \ldots, n$, are continuous as well, although these assumptions are easily relaxed.

*EMSR-a.* EMSR-a (*expected marginal seat revenue–version a*) is the most widely publicized heuristic for the single-resource problem. Despite this fact, it is less popular in practice than its close cousin, EMSR-b, which surprisingly is not well documented in the literature. Generally, EMSR-b provides better revenue performance, and it is certainly more intuitive, although EMSR-a is important to know just the same.

EMSR-a is based on the idea of adding the protection levels produced by applying Littlewood's rule to successive pairs of classes. Consider stage $j+1$, in which demand of class $j+1$ arrives with price $p_{j+1}$. We are interested in computing how much capacity to reserve for the remaining classes, $j, j-1, \ldots, 1$; that is, the protection level, $y_j$, for classes $j$ and higher. To do so, let us consider a single class $k$ among the remaining classes $j, j-1, \ldots, 1$ and compare $k$ and $j+1$ *in isolation.* Considering only these two classes, we would use Littlewood's rule (2) and reserve capacity $y_k^{j+1}$ for class $k$, where

$$P(D_k > y_k^{j+1}) = \frac{p_{j+1}}{p_k}. \tag{10}$$

Repeating for each future class $k = j, j-1, \ldots, 1$, we could likewise compute how much capacity to reserve for each such class $k$ in isolation. The idea of EMSR-a, then, is simply to add up these individual protection levels to approximate the total protection level $y_j$ for classes $j$ and higher. That is, set the protection level $y_j$ as

$$y_j = \sum_{k=1}^{j} y_k^{j+1}, \tag{11}$$

where $y_k^{j+1}$ is given by (10). One then repeats this same calculation for each stage $j$.

EMSR-a is certainly simple and has an intuitive appeal. For a short while it was even believed to be optimal, but this notion was quickly dispelled once the published work on optimal controls appeared.

*EMSR-b.* EMSR-b is again based on an approximation that reduces the problem at each stage to two classes, but in contrast to EMSR-a, the approximation is based on aggregating *demand* rather than aggregating *protection levels*. Specifically, the demand from future classes is aggregated and treated as one class with a revenue equal to the weighted-average revenue.

Consider stage $j+1$, in which we want to determine protection level $y_j$. Define the aggregated future demand for classes $j, j-1, \ldots, 1$ by

$$S_j = \sum_{k=1}^{j} D_k,$$

and let the weighted-average revenue from classes $1, \ldots, j$, denoted $\bar{p}_j$, be defined by

$$\bar{p}_j = \frac{\sum_{k=1}^{j} p_k E[D_k]}{\sum_{k=1}^{j} E[D_k]}. \tag{12}$$

Then the EMSR-b protection level for class $j$ and higher, $y_j$, is chosen by Littlewood's rule (2) so that

$$P(S_j > y_j) = \frac{p_{j+1}}{\bar{p}_j}. \tag{13}$$

It is common when using EMSR-b to assume that demand for each class $j$ is independent and normally distributed with mean $\mu_j$ and variance $\sigma_j^2$, in which case

$$y_j = \mu + z_\alpha \sigma,$$

where $\mu = \sum_{k=1}^{j} \mu_k$ is the mean and $\sigma^2 = \sum_{k=1}^{j} \sigma_k^2$ is the variance of the aggregated demand to come at stage $j+1$ and $z_\alpha = \Phi^{-1}(1 - p_{j+1}/\bar{p}_j)$ (recall $\Phi^{-1}(x)$ is the inverse of the standard normal c.d.f.). Again, one repeats this calculation for each $j$.

In practice EMSR-b is more popular and generally seems to perform better than EMSR-a, although studies comparing the two have at times shown mixed results. Belobaba [6] reports studies in which EMSR-b is consistently within 0.5% of the optimal revenue, whereas EMSR-a can deviate by nearly 1.5% from the optimal revenue in certain cases, although with mixed order of arrival and frequent reoptimization, he reports that both methods perform well. However, another recent study by Polt [66] using Lufthansa airline data showed more mixed performance, with neither method dominating the other.

## 2.4. Dynamic Models

Dynamic models relax the assumption that the demand for classes arrives in a strict low-to-high revenue order. Instead, they allow for an arbitrary order of arrival, with the possibility of interspersed arrivals of several classes. While at first this seems like a strict generalization of the static case, the dynamic models require the assumption of Markovian (such as Poisson) arrivals to make them tractable. This puts restrictions on modeling different levels of variability in demand. Indeed, this limitation on the distribution of demand is the main drawback of dynamic models in practice. In addition, dynamic models require an estimate of the pattern of arrivals over time (called the *booking curve*), which may be difficult to estimate in certain applications. Thus, the choice of dynamic versus static models essentially comes down to a choice of which set of approximations is more acceptable and what data are available in any given application.

Other assumptions of the static model are retained. Demand is assumed to be independent between classes and over time and also independent of the capacity controls. The firm is again assumed to be risk neutral. The justifications (or criticisms) for these assumptions are the same as in the static-model case.

**2.4.1. Formulation and Structural Properties.** In the simplest dynamic model, we have $n$ classes as before, with associated prices $p_1 \geq p_2 \geq \cdots \geq p_n$. There are $T$ total periods and $t$ indexes the periods, with the time index running forward ($t = 1$ is the first period, and $t = T$ is the last period; this is in contrast to the static dynamic program, where the stages run from $n$ to 1 in the dynamic programming recursion). Because there is no longer a one-to-one correspondence between periods and classes, we use separate indices—$t$ for periods and $j$ for classes.

In each period we assume, by a sufficiently fine discretization of time, that at most one arrival occurs.[8] The probability of an arrival of class $j$ in period $t$ is denoted $\lambda_j(t)$. The assumption of at most one arrival per period implies that we must have

$$\sum_{j=1}^{n} \lambda_j(t) \leq 1.$$

In general, the periods need not be of the same duration. For example, early in the booking process when demand is low, we might use a period of several days, whereas during periods of peak booking activity we might use a period of less than an hour. Note also that the arrival probabilities may vary with $t$, so the mix of classes that arrive may vary over time. In particular, we do not require lower classes to arrive earlier than higher classes.

*Dynamic program.* As before, let $x$ denote the remaining capacity and $V_t(x)$ denote the value function in period $t$. Let $R(t)$ be a random variable, with $R(t) = p_j$ if a demand for class $j$ arrives in period $t$, and $R(t) = 0$ otherwise. Note that $P(R(t) = p_j) = \lambda_j(t)$. Let $u = 1$ if we accept the arrival (if there has been one), and $u = 0$ otherwise. (We suppress the period subscript $t$ of the control as it should be clear from the context.) We want to maximize the sum of current revenue and the revenue to go, or

$$R(t)u + V_{t+1}(x - u).$$

The Bellman equation is therefore

$$\begin{aligned} V_t(x) &= E\Big[\max_{u \in \{0,1\}} \{R(t)u + V_{t+1}(x-u)\}\Big] \\ &= V_{t+1}(x) + E\Big[\max_{u \in \{0,1\}} \{(R(t) - \Delta V_{t+1}(x))u\}\Big], \end{aligned} \tag{14}$$

where $\Delta V_{t+1}(x) = V_{t+1}(x) - V_{t+1}(x-1)$ is the expected marginal value of capacity in period $t+1$. The boundary conditions are[9]

$$V_{T+1}(x) = 0, \quad x = 0, 1, \ldots, C,$$

and

$$V_t(0) = 0, \quad t = 1, \ldots, T.$$

*Optimal policy.* An immediate consequence of (14) is that if a class $j$ request arrives, so that $R(t) = p_j$, then it is optimal to accept the request if and only if

$$p_j \geq \Delta V_{t+1}(x).$$

Thus, the optimal control can be implemented using a bid-price control where the bid price is equal to the marginal value,

$$\pi_t(x) = \Delta V_t(x). \tag{15}$$

[8] The assumption of one arrival per period can be generalized as shown by Lautenbacher and Stidham [51], but it is a convenient assumption both theoretically and computationally.

[9] The second boundary condition can be eliminated if we use the control constraint $u \in \{0, \min\{1, x\}\}$ instead of $u \in \{0, 1\}$. However, it is simpler conceptually and notationally to use the $x = 0$ boundary conditions instead.

Revenues that exceed this threshold are accepted; those that do not are rejected.

As in the static case, an important property of the value function is that it has decreasing marginal value $\Delta V_t(x) = V_t(x) - V_t(x-1)$.

**Proposition 2.** *The increments $\Delta V_t(x)$ of the value function $V_t(x)$ defined by (14) satisfy $\forall x, t$:*

(i) $\Delta V_t(x+1) \le \Delta V_t(x)$, *and*
(ii) $\Delta V_{t+1}(x) \le \Delta V_t(x)$.

This theorem is natural and intuitive because one would expect the value of additional capacity at any point in time to have a decreasing marginal benefit and the marginal value at any given remaining capacity $x$ to decrease with time (because as time elapses, there are fewer opportunities to sell the capacity).

As a consequence, the optimization on the right-hand side of (14) can also be implemented as a nested-allocation policy, albeit one that has time-varying protection levels (or booking limits). Specifically, we can define time-dependent optimal protection levels

$$y_j^*(t) = \max\{x : p_{j+1} < \Delta V_{t+1}(x)\}, \quad j = 1, 2, \ldots, n-1 \tag{16}$$

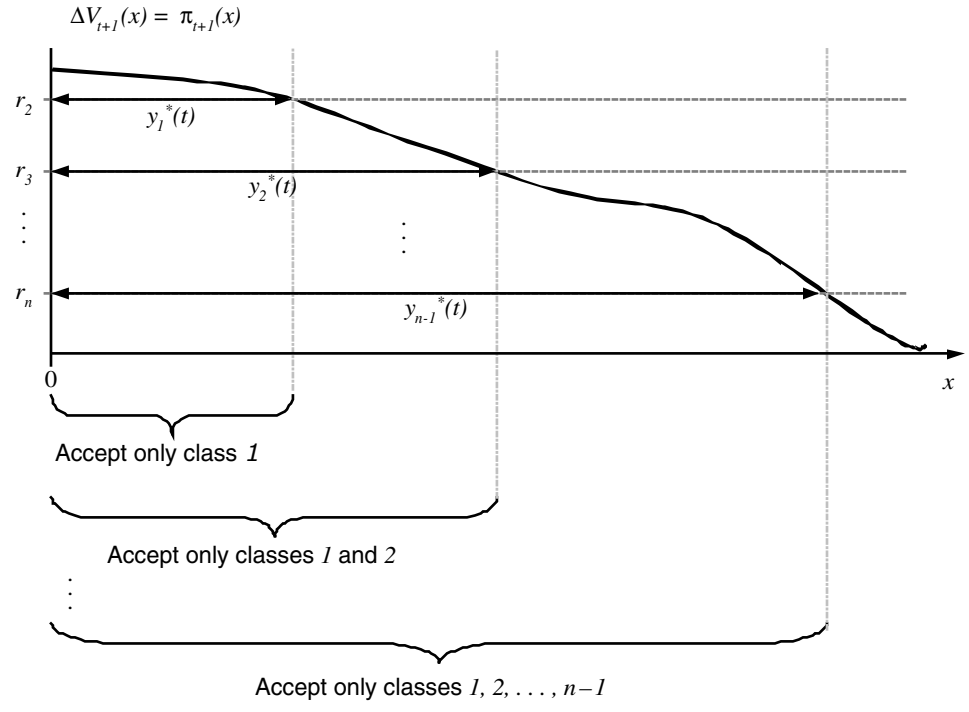
that have the usual interpretation that $y_j^*(t)$ is the capacity we protect for classes $j, j-1, \ldots, 1$. Then the protection levels are nested, $y_1^*(t) \le y_2^*(t) \le \cdots \le y_{n-1}^*(t)$, and it is optimal to accept class $j$ if and only if the remaining capacity exceeds $y_{j-1}^*(t)$. The situation is illustrated in Figure 3.

Time-dependent nested booking limits can also be defined as before by

$$b_j^*(t) \equiv C - y_{j-1}^*(t), \quad j = 2, \ldots, n, \tag{17}$$

That the booking limits and protection levels depend on time in this case essentially stems from the fact that the demand to come varies with time in the dynamic model. The change

FIGURE 3. Optimal protection level $y_j^*(t)$ in the dynamic model.



in demand to come as time evolves affects the opportunity cost, and therefore the resulting booking limit and protection levels.

As a practical matter, because the value function is not likely to change much over short periods of time, fixing the protection levels or booking limits computed by a dynamic model and updating them periodically (as is done in most RM systems in practice) is usually close to optimal. Still, the time-varying nature of the protection levels remains a key distinction between static and dynamic models.

We summarize these results in the following theorem:

**Theorem 2.** *For the dynamic model defined by* (14)*, the optimal control can be achieved using either:*

(i) *time-dependent nested protection levels defined by* (16)*,*
(ii) *time-dependent nested booking limits defined by* (17)*, or*
(iii) *bid-price tables defined by* (15)*.*

## 2.5. Customer-Choice Behavior

A key assumption in the models that we have described thus far is that demand for each of the classes is completely independent of the capacity controls being applied by the seller. That is, it is assumed that the likelihood of receiving a request for any given class does not depend on which other classes are available at the time of the request. Needless to say, this is a somewhat unrealistic assumption. For example, in the airline case the likelihood of selling a full-fare ticket may very well depend on whether a discount fare is available at the same time, and the likelihood that a customer buys at all may depend on the lowest available fare. When customers buy a higher fare when a discount is closed it is called *buy-up* (from the firm's point of view, this is also called *sell-up*); when they choose another flight when a discount is closed it is called *diversion*.

Clearly, such customer behavior could have important RM consequences and ought to be considered when making control decisions. We next look at some heuristic and exact methods for incorporating customer-choice behavior in single-resource problems.

**2.5.1. Buy-Up Factors.** One approach to modeling customer-choice behavior that works with the two-class model is to include buy-up probabilities—also called *buy-up factors*—in the formulation.

The approach works as follows. Consider the simple two-class static model, and recall that Littlewood's rule (2) (slightly restated) is to accept demand from class 2 if and only if

$$p_2 \geq p_1 P(D_1 > x), \tag{18}$$

where $x$ is the remaining capacity—that is, if the revenue from accepting class 2 exceeds the marginal value of the unit of capacity required to satisfy the request. Now suppose that there is a probability $q$ that a customer for class 2 will buy class 1 if class 2 is closed. The net benefit of accepting the request is still the same, but now rather than losing the request when we reject it, there is some chance the customer will buy up to class 1. If so, we earn a net benefit of $p_1 - p_1 P(D_1 > x)$ (the class 1 revenue minus the expected marginal cost). Thus, it is optimal to accept class 2 now if $p_2 - p_1 P(D_1 \geq x) \geq q p_1 (1 - P(D_1 > x))$ or, equivalently, if

$$p_2 \geq (1-q) p_1 P(D_1 > x) + q p_1. \tag{19}$$

Note that the right-hand side of the modified rule (19) is strictly larger than the right-hand side in Littlewood's rule (18), which means that the modified rule (19) is more likely to reject class 2 demand. This is intuitive because with the possibility of customers upgrading to class 1, we should be more eager to close class 2.

The difficulty with this approach is that it does not extend to more than two classes—at least not in an exact way—because the probability that a customer buys class $i$ given

that class $j$ is closed depends not only on $i$ and $j$, but also on which other classes are also available. In other words, with more than two classes the customer faces a *multinomial* choice rather than a *binary* choice.

However, one can at least heuristically extend the buy-up factor idea to EMSR-a or EMSR-b, because these heuristics approximate the multiclass problem using the two-class model.

For example, EMSR-b can be extended to allow for a buy-up factor by modifying the equation for determining the protection level $y_j$, (13), as follows:

$$p_{j+1} = (1 - q_{j+1})\bar{p}_j P(S_j > y_j) + q_{j+1}\hat{p}_{j+1}, \tag{20}$$

where $q_{j+1}$ is the probability that a customer of class $j+1$ buys up to one of the classes $j, j-1, \ldots, 1$; $\bar{p}_j$ is the weighted-average revenue from these classes as defined by (12); and $\hat{p}_{j+1} > p_{j+1}$ is an estimate of the average revenue received given that a class $j+1$ customer buys up to one of the classes $j, j-1, \ldots, 1$ (for example, $\hat{p}_{j+1} = p_j$ if customers are assumed to buy up to the next-highest price class). Again, the net result of this change is to increase the protection level $y_j$ and close down class $j+1$ earlier than one would do under the traditional EMSR-b rule.[10]

While this modification to EMSR-b provides a simple heuristic way to incorporate choice behavior, it is a somewhat ad hoc adjustment to an already heuristic approach to the problem. Beyond the limitations of the model and its assumptions, there are some serious difficulties involved in estimating the buy-up factors. Indeed, in current applications of the model, they are often simply made-up, reasonable-sounding numbers. Moreover, the assumptions of the model can clash with unconstraining and recapture procedures that are subsequently applied, resulting in double counting of demand. Despite these limitations, buy-up factors have proved useful as a rough-cut approach for incorporating choice behavior in practice.

**2.5.2. Discrete-Choice Models.** We next look at a single-resource problem in which customer-choice behavior is explicitly modeled using a general discrete-choice model. In contrast to the heuristic approach of buy-up factors, this model provides a more theoretically sound approach to incorporating choice behavior. It also provides insights into how choice behavior affects the optimal availability controls. The theory is first developed for the general choice model case and then applied to some special demand models.

*Model definition.* As in the traditional dynamic model of §2.4, time is discrete and indexed by $t$, with the indices running forward in time ($t = T$ is the period of resource usage). In each period there is at most one arrival. The probability of arrival is denoted by $\lambda$, which we assume, for ease of exposition, is the same for all time periods $t$. There are $n$ classes, and we let $\mathcal{N} = \{1, \ldots, n\}$ denote the entire set of classes. We let choice index 0 denote the no-purchase choice; that is, the event that the customer does not purchase any of the classes offered. Each class $j \in \mathcal{N}$ has an associated price $p_j$, and without loss of generality we index classes so that $p_1 \geq p_2 \geq \cdots \geq p_n \geq 0$. We let $p_0 = 0$ denote the revenue of the no-purchase choice.

Customer purchase behavior is modeled as follows. In each period $t$, the seller chooses a subset $S_t \subseteq \mathcal{N}$ of classes to offer. When the set of classes $S_t$ is offered in period $t$, the probability that a customer chooses class $j \in S_t$ is denoted $P_j(S_t)$. $P_0(S_t)$ denotes the no-purchase probability.

The probability that a sale of class $j$ is made in period $t$ is therefore $\lambda P_j(S_t)$, and the probability that no sale is made is $\lambda P_0(S_t) + (1 - \lambda)$. Note that this last expression reflects the fact that having no sales in a period could be due either to no arrival at all or an

[10] That it increases the protection level about the usual EMSR-b value can be seen by noting that $p_{j+1} = \bar{R}_j P(S_j > y_j)$ in the usual EMSR-b case and $\hat{p}_{j+1} > p_{j+1}$; thus, $y_j$ has to increase to satisfy the equality (20).

TABLE 1. Fare-product revenues and restrictions for Example 2.

| Fare product (class) | SA stay | 21-day adv. | Revenue |
|---|---|---|---|
| $Y$ | No | No | \$800 |
| $M$ | No | Yes | \$500 |
| $K$ | Yes | Yes | \$450 |

TABLE 2. Segments and their characteristics for Example 2.

| | | Qualifies for restrictions? | | Willing to buy? | |
|---|---|---|---|---|---|
| Segment | Prob. | SA stay | 21-day adv. | $Y$ class | $M$ class |
| Bus. 1 | 0.1 | No | No | Yes | Yes |
| Bus. 2 | 0.2 | No | Yes | Yes | Yes |
| Leis. 1 | 0.2 | No | Yes | No | Yes |
| Leis. 2 | 0.2 | Yes | Yes | No | Yes |
| Leis. 3 | 0.3 | Yes | Yes | No | No |

arrival that does not purchase. This leads to an incomplete-data problem when estimating the model.

The only condition we impose on the choice probabilities $P_j(S)$ is that they define a proper probability function. That is, for every set $S \subseteq \mathcal{N}$, the probabilities satisfy

$$P_j(S) \geq 0, \quad \forall\, j \in S$$
$$\sum_{j \in S} P_j(S) + P_0(S) = 1.$$

This includes most choice models of practical interest (see Ben-Akiva and Lerman [8]) and even some rather pathological cases.[11] The following running example will be used to illustrate the model and analysis:

**Example 2.** An airline offers three fare products—$Y, M$, and $K$. These products differ in terms of revenues and conditions, as shown in Table 1. The airline has five segments of customers—two business segments and three leisure segments. The segments differ in terms of the restrictions that they qualify for and the fares they are willing to pay. The data describing each segment are given in Table 2. The second column of Table 2 gives the probability that an arriving customer is from each given segment.

Given this data for Example 2, the first four columns of Table 3 give the choice probabilities that would result.[12]

This particular method of generating choice probabilities is only for illustration. Other choice models could be used, and in general any proper set of probabilities could be used to populate Table 3.

*Formulation.* As before, let $C$ denote the total capacity, $T$ the number of time-periods, $t$ the current period, and $x$ the number of remaining inventory units. Define the value function

[11] For example, some psychologists have shown that customers can be overwhelmed by more choices, and they may become more reluctant to purchase as more options are offered (see Iyengar and Lepper [43]). Such cases would be covered by a suitable choice of $P_j(S)$ that results in the total probability of purchase, $\sum_{j \in S} P_j(S)$, being decreasing in $S$.

[12] To see how the probabilities in Table 3 are derived, consider the set $S = \{Y, K\}$. If $S = \{Y, K\}$ is offered, segments Business 1 and Business 2 buy the $Y$ fare because they cannot qualify for both the SA stay and 21-day advance-purchase restrictions on $K$, so $P_Y = 0.1 + 0.2 = 0.3$. Similarly, Leisure 1 cannot qualify for the SA stay restriction of $K$ and is not willing to purchase $Y$, so these customers do not purchase at all. Segments Leisure 2 and 3, however, qualify for both restrictions on $K$ and purchase $K$. Hence, $P_K = 0.2 + 0.3 = 0.5$. Class $M$ is not offered, so $P_M = 0$. The other rows of Table 3 are filled out similarly.

TABLE 3. Choice probabilities $P_j(S)$, probability of purchase $Q(S)$, and expected revenue $R(S)$ for Example 2.

| $S$ | $P_Y(S)$ | $P_M(S)$ | $P_K(S)$ | $Q(S)$ | $R(S)$ | *Efficient?* |
|---|---|---|---|---|---|---|
| $\{Y\}$ | 0.3 | 0 | 0 | 0.3 | 240 | Yes |
| $\{M\}$ | 0 | 0.4 | 0 | 0.4 | 200 | No |
| $\{K\}$ | 0 | 0 | 0.5 | 0.5 | 225 | No |
| $\{Y,M\}$ | 0.1 | 0.6 | 0 | 0.7 | 380 | No |
| $\{Y,K\}$ | 0.3 | 0 | 0.5 | 0.8 | 465 | Yes |
| $\{M,K\}$ | 0 | 0.4 | 0.5 | 0.9 | 425 | No |
| $\{Y,M,K\}$ | 0.1 | 0.4 | 0.5 | 1 | 505 | Yes |

*Note.* Efficient sets are defined in §2.5.2.

$V_t(x)$ as the maximum expected revenue obtainable from periods $t, t+1, \ldots, T$ given that there are $x$ inventory units remaining at time $t$. Then the Bellman equation for $V_t(x)$ is

$$\begin{aligned} V_t(x) &= \max_{S\subseteq\mathcal{N}}\left\{\sum_{j\in S}\lambda P_j(S)(p_j + V_{t+1}(x-1)) + (\lambda P_0(S) + 1 - \lambda)V_{t+1}(x)\right\} \\ &= \max_{S\subseteq\mathcal{N}}\left\{\sum_{j\in S}\lambda P_j(S)(p_j - \Delta V_{t+1}(x))\right\} + V_{t+1}(x), \end{aligned} \tag{21}$$

where $\Delta V_{t+1}(x) = V_{t+1}(x) - V_{t+1}(x-1)$ denotes the marginal cost of capacity in the next period, and we have used the fact that for all $S$,

$$\sum_{j\in S} P_j(S) + P_0(S) = 1.$$

The boundary conditions are

$$V_{T+1}(x) = 0, \quad x = 0, 1, \ldots, C \tag{22}$$

$$V_t(0) = 0, \quad t = 1, \ldots, T. \tag{23}$$

Note one key difference in this formulation compared to our analysis of the traditional independent-class models of §§2.3.2 and 2.4—we assume the seller *precommits* to the open set of classes $S$ in each period, while in the traditional models, we assume the seller observes the class of the request and then makes an accept or deny decision based on the class. The reason for the difference is that in the traditional models the class of an arriving request is completely independent of the controls, so it does not matter whether we precommit to the set of open classes or not. However, in the choice-based model, the class that an arriving customer chooses depends (through the choice model $P_j(S)$) on which classes $S$ we report as being open. Hence, the formulation (21) reflects this fact (we are taking $\max E[\cdot]$ in 21 instead of $E[\max(\cdot)]$); we must choose $S$ prior to seeing the realization of the choice decision.

*Structure of the optimal policy.* The problem (21) at first seems to have very little structure, but a sequence of simplifications provides a good characterization of the optimal policy. The first simplification is to write (21) in more compact form as

$$V_t(x) = \max_{S\subseteq\mathcal{N}}\{\lambda(R(S) - Q(S)\Delta V_{t+1}(x))\} + V_{t+1}(x), \tag{24}$$

where

$$Q(S) = \sum_{j\in S} P_j(S) = 1 - P_0(S)$$

denotes the total probability of purchase, and

$$R(S) = \sum_{j \in S} P_j(S) p_j$$

denotes the total expected revenue from offering set $S$. Table 3 gives the values $Q(S)$ and $R(S)$ for our Example 2. For theoretical purposes, we also consider allowing the seller to randomize over the sets $S$ that are offered at the beginning of each time period, but this relaxation is not strictly needed because there is always at least one set $S$ that achieves the maximum in (24).

The second simplification is to note that not all $2^n - 1$ subsets need to be considered when maximizing the right-hand side of (24). Indeed, the search can be reduced to only those sets that are efficient as defined below:

**Definition 1.** A set $T$ is said to be *inefficient* if there exist probabilities $\alpha(S), \forall S \subseteq \mathcal{N}$ with $\sum_{S \subseteq \mathcal{N}} \alpha(S) = 1$ such that

$$Q(T) \geq \sum_{S \subseteq \mathcal{N}} \alpha(S) Q(S) \quad \text{and} \quad R(T) < \sum_{S \subseteq \mathcal{N}} \alpha(S) R(S).$$

A set is said to be *efficient* if no such probabilities $\alpha(S)$ exist.

In words, a set $T$ is inefficient if we can use a randomization of other sets $S$ to produce an expected revenue that is strictly greater than $R(T)$ with no increase in the probability of purchase $Q(T)$.

The significance of inefficient sets is that they can be eliminated from consideration:

**Proposition 3.** *An inefficient set is never an optimal solution to* (21).

The proof is omitted, but the fact that such sets should be eliminated from consideration is quite intuitive from (24); an inefficient set $T$ provides strictly less revenue $R(T)$ than do other sets and incurs at least as high a probability of consuming capacity $Q(T)$ (and hence incurs at least as high an opportunity cost $Q(S)\Delta V_{t+1}(x)$ in (24)).

For Example 2, Table 3 shows which sets are efficient—namely, the sets $\{Y\}$, $\{Y, K\}$, and $\{Y, K, M\}$. That these sets are efficient follows from inspection of Figure 4, which shows a scatter plot of the values $Q(S)$ and $R(S)$ for all subsets $S$. Note from this figure and Definition 1 that an efficient set is a point that is on the "efficient frontier" of the set of points $\{Q(S), R(S)\}, S \subseteq \mathcal{N}$. Here, "efficiency" is with respect to the trade-off between expected revenue $R(S)$ and probability of sale $Q(S)$.

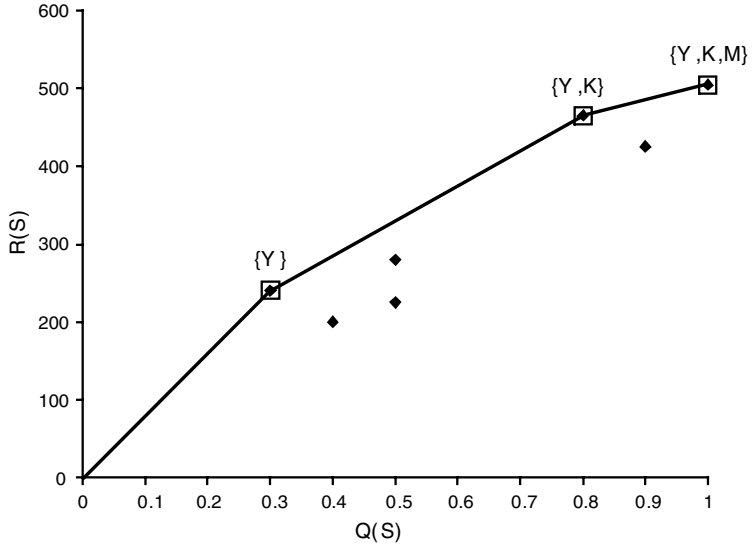
The third simplification is to note that the efficient sets can be easily ordered. Indeed, let $m$ denote the number of efficient sets. These sets can be indexed $S_1, \ldots, S_m$ such that both the revenues and probabilities of purchase are monotone increasing in the index. That is, if the collection of $m$ efficient sets is indexed such that $Q(S_1) \leq Q(S_2) \leq \cdots \leq Q(S_m)$, then $R(S_1) \leq R(S_2) \leq \cdots \leq R(S_m)$ as well. The proof of this fact is again omitted, but it is easy to see intuitively from Figure 4. Note from Table 3 that there are $m = 3$ efficient sets $\{Y\}$, $\{Y, K\}$, and $\{Y, K, M\}$. These can be ordered $S_1 = \{Y\}$, $S_2 = \{Y, K\}$, and $S_3 = \{Y, K, M\}$, with associated probabilities of purchase $Q_1 = 0.3$, $Q_2 = 0.8$, and $Q_3 = 1$ and prices $p_1 =$ \$240, $p_2 =$ \$465, and $p_3 =$ \$505 as claimed.

Henceforth, we assume the efficient sets are denoted $S_1, \ldots, S_m$ and are indexed in increasing revenue and probability order. Also, to simplify notation we let $R_k = R(S_k)$ and $Q_k = Q(S_k)$ and note that $R_k$ and $Q_k$ are both increasing in $k$. Therefore, the Bellman equation can be further simplified to

$$V_t(x) = \max_{k=1,\ldots,m} \{\lambda(R_k - Q_k \Delta V_{t+1}(x))\} + V_{t+1}(x). \tag{25}$$

The final simplification is to show that when expressed in terms of the sequence $S_1, \ldots, S_m$ of efficient sets, the optimal policy has a simple form as stated in the following theorem:

FIGURE 4. Scatter plot of $Q(S)$ and $R(S)$ for Example 2 (efficient points are enclosed in squares and labeled).




**Theorem 3.** *An optimal policy for* (21) *is to select a set* $k^*$ *from among the* $m$ *efficient, ordered sets* $\{S_k : k = 1, \ldots, m\}$ *that maximizes* (25)*. Moreover, for a fixed* $t$*, the largest optimal index* $k^*$ *is increasing in the remaining capacity* $x$*, and for any fixed* $x$*,* $k^*$ *is increasing in time* $t$*.*

The proof of this theorem is involved, but derives from the fact that the marginal value $\Delta V_{t+1}(x)$ is decreasing in $x$ and the fact that the optimal index $k^*$ is decreasing in this marginal value.

This characterization is significant for several reasons. First, it shows that the optimal sets can be reduced to only those that are efficient, which in many cases significantly reduces the number of sets we need to consider. Moreover, it shows that this limited number of sets can be sequenced in a natural way, and that the more capacity we have (or the less time remaining), the higher the set we should use in this sequence.

For example, applying Theorem 3 to Example 2, we see that the efficient sets $S_1 = \{Y\}$, $S_2 = \{Y, K\}$, and $S_3 = \{Y, K, M\}$ would be used as follows. With very large amounts of capacity remaining, $S_3$ is optimal: All three fare classes are opened. As capacity is consumed, at some point we switch to only offering $S_2$: Class $M$ is closed, and only $Y$ and $K$ are offered. As capacity is reduced further, at some point we close class $K$ and offer only class $Y$ (set $S_1$ is used).

Note what is odd here; it can be optimal to offer the highest fare $Y$ and the lowest fare $K$, but not the middle fare $M$. This is because opening $M$ causes some buy-down from $Y$ to $M$, whereas $K$ is sufficiently restricted to prevent buy-down. Only when capacity is plentiful is $M$ opened.

*Optimality of nested-allocation policies.* The optimization results above also have important implications for the optimality of nested-allocation policies. Indeed, Definition 1 and Theorem 3 can be used to provide a complete characterization of cases in which nested-allocation policies are optimal. They also can be used to provide conditions under which the optimal nesting is by revenue order.

We begin with a precise definition of a nested-allocation policy in the context of the choice model:

**Definition 2.** A control policy is called a *nested policy* if there is an increasing family of subsets $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_m$ and an index $k_t(x)$ that is increasing in $x$, such that set $S_{k_t(x)}$ is chosen at time $t$ when the remaining capacity is $x$.

Although this is a somewhat abstract definition of a nested policy, it is in fact a natural generalization of nested allocations from the traditional single-resource models of §§2.3.2 and 2.4 and implies an ordering of the classes based on when they first appear in the increasing sequence of sets $S_k$. That is, class $i$ is considered "higher" than class $j$ in the nesting order if class $i$ appears earlier in the sequence. Returning to Example 2, we see that the efficient sets are indeed nested according to this definition because $S_1 = \{Y\}$, $S_2 = \{Y, K\}$, and $S_3 = \{Y, K, M\}$ are increasing. Class $Y$ would be considered the highest in the nested order, followed by class $K$ and then class $M$.

If the optimal policy is nested in this sense, then we can define optimal protection levels $y_k^*(t), k = 1, \ldots, m$, such that classes lower in the nesting order than those in $S_k$ are closed if the remaining capacity is less than $y_k^*(t)$, just as in the traditional single-resource case. The optimal protection levels for $k = 1, 2, \ldots, m-1$ are defined by

$$y_k^*(t) = \max\{x : R_k - Q_k \Delta V_{t+1}(x) > R_{k+1} - Q_{k+1} \Delta V_{t+1}(x)\}.$$

Nested booking limits can also be defined in the usual way, $b_k(t) = C - y_{k-1}(t)$.

We again return to Example 2 to illustrate this concept. Table 4 shows the objective function value $R_k - Q_k \Delta V_{t+1}(x)$ for each of the three efficient sets $k = 1, 2, 3$, for a particular marginal value function $\Delta V_{t+1}(x)$, which we assume is given in this example. Capacities are in the range $x = 1, 2, \ldots, 20$. The last column of Table 4 gives the index, $k_t^*(x)$, of the efficient set that is optimal for each capacity $x$.

Note that for capacities 1, 2, and 3, the set $S_1 = \{Y\}$ is the optimal set, so class $Y$ is the only open fare. Once we reach four units of remaining capacity, set $S_2 = \{Y, K\}$ becomes optimal and we open class $K$ in addition to class $Y$. When the remaining capacity reaches 13, set $S_3 = \{Y, K, M\}$ becomes optimal, and we open $M$ in addition to $Y$ and $K$. As a result, the optimal protection level for set $S_1$, is $y_1^* = 3$, and the protection level for set $S_2$ is $y_2^* = 12$. $S_3$ has a protection level equal to capacity.

Table 4. Illustration of nested policy for Example 2.

| | | $R_k - Q_k \Delta V_{t+1}(x)$ | | | |
|---|---|---|---|---|---|
| $x$ | $\Delta V_{t+1}(x)$ | $k=1$ | $k=2$ | $k=3$ | $k_t^*(x)$ |
| 1 | 780.00 | 6.00 | −159.00 | −275.00 | 1 |
| 2 | 624.00 | 52.80 | −34.20 | −119.00 | 1 |
| 3 | 520.00 | 84.00 | 49.00 | −15.00 | 1 |
| 4 | 445.71 | 106.29 | 108.43 | 59.29 | 2 |
| 5 | 390.00 | 123.00 | 153.00 | 115.00 | 2 |
| 6 | 346.67 | 136.00 | 187.67 | 158.33 | 2 |
| 7 | 312.00 | 146.40 | 215.40 | 193.00 | 2 |
| 8 | 283.64 | 154.91 | 238.09 | 221.36 | 2 |
| 9 | 260.00 | 162.00 | 257.00 | 245.00 | 2 |
| 10 | 240.00 | 168.00 | 273.00 | 265.00 | 2 |
| 11 | 222.86 | 173.14 | 286.71 | 282.14 | 2 |
| 12 | 208.00 | 177.60 | 298.60 | 297.00 | 2 |
| 13 | 195.00 | 181.50 | 309.00 | 310.00 | 3 |
| 14 | 183.53 | 184.94 | 318.18 | 321.47 | 3 |
| 15 | 173.33 | 188.00 | 326.33 | 331.67 | 3 |
| 16 | 164.21 | 190.74 | 333.63 | 340.79 | 3 |
| 17 | 156.00 | 193.20 | 340.20 | 349.00 | 3 |
| 18 | 148.57 | 195.43 | 346.14 | 356.43 | 3 |
| 19 | 141.82 | 197.45 | 351.55 | 363.18 | 3 |
| 20 | 135.65 | 199.30 | 356.48 | 369.35 | 3 |

## 3. Dynamic Pricing

In this section, we look at settings in which prices rather than quantity controls are the primary variables used to manage demand. While the distinction between quantity and price controls is not always sharp (for instance, closing the availability of a discount class can be considered equivalent to raising the product's price to that of the next-highest class), the techniques we look at here are distinguished by their explicit use of price as the control variable and their explicit modeling of demand as a price-dependent process.

In terms of business practice, varying prices is often the most natural mechanism for revenue management. In most retail and industrial trades, firms use various forms of dynamic pricing—including personalized pricing, markdowns, display and trade promotions, coupons, discounts, clearance sales, and auctions and price negotiations (request for proposals and request for quotes—RFP/RFQ processes) to respond to market fluctuations and uncertainty in demand. Exactly how to make such price adjustments in a way that maximizes revenues (or profits, in the case where variable costs are involved) is the subject of this section.

Dynamic pricing is as old as commerce itself. Firms and individuals have always resorted to price adjustments (such as haggling at the bazaar) in an effort to sell their goods at a price that is as high as possible, yet acceptable to customers. However, the last decade has witnessed an increased application of scientific methods and software systems for dynamic pricing, both in the estimation of demand functions and the optimization of pricing decisions.

### 3.1. Price-Based vs. Quantity-Based RM

Some industries use price-based RM (retailing), whereas others use quantity-based RM (airlines). Even in the same industry, firms may use a mixture of price- and quantity-based RM. For instance, many of the RM practices of the new low-cost airlines more closely resemble dynamic pricing than the quantity-based RM of the traditional carriers. What explains these differences?

It is hard to give a definitive answer, but in essence it boils down to a question of the extent to which a firm is able to vary quantity or price in response to changes in market conditions. This ability, in turn, is determined by the commitments a firm makes (to price or quantity), its level of flexibility in supplying products or services, and the costs of making quantity or price changes.

Consider airlines, for example. While arguably less true today than in the past, airlines normally commit to prices for their various fare products in advance of taking bookings. This is due to advertising constraints (such as the desire to publish fares in print media and fare tariff books), distribution constraints, and a desire to simplify the task of managing prices. For these marketing and administrative reasons, most airlines advertise and price fare products on an aggregate origin-destination market level for a number of flights over a given interval of time, and do not price on a departure-by-departure basis. This limits their ability to use price to manage the demand on any given departure, demand that varies considerably by flight and is quite uncertain at the time of the price posting. At the same time, the supply of the various classes is almost perfectly flexible between the products (subject to the capacity constraint of the flight) because all fare products sold in the same cabin of service share a homogeneous seat capacity. It is this combination of price commitments together with flexibility on the supply side that make quantity-based RM an attractive tactic in the airline industry. Hotels, cruise ships, and rental cars—other common quantity-based RM industries—share many of these same attributes.

In other cases, however, firms have more price flexibility than quantity flexibility. In apparel retailing, for example, firms commit to order quantities well in advance of a sales season—and may even commit to certain stocking levels in each store. Often, it is impossible (or very costly) to reorder stock or reallocate inventory from one store to another. At the

same time, it is easier (though not costless) for most retailers to change prices, as this may require only changing signage and making data entries into a point-of-sale system. Online retailers in particular enjoy tremendous price flexibility because changing prices is almost costless. Business-to-business sales are often conducted through a RFP/RFQ process, which allows firms to determine prices on a transaction-by-transaction basis. In all these situations, price-based RM is therefore a more natural practice. Of course, the context could dictate a different choice even in these industries. For example, if a retailer commits to advertised prices in different regional markets yet retains a centralized stock of products, it might then choose to manage demand by tactically allocating its supply to these different regions—a quantity-based RM approach.

However, given the choice between price- and quantity-based RM, one can argue that price-based RM is the preferred option. The argument is as follows (see Gallego and van Ryzin [35]). Quantity-based RM operates by rationing the quantity sold to different products or to different segments of customers. However, rationing, by its very nature, involves reducing sales by *limiting* supply. If one has price flexibility, however, rather than reducing sales by *limiting supply*, we can reduce sales by *increasing price*. This achieves the same quantity-reducing function as rationing, but does it more profitably because by increasing price we both reduce sales *and* increase revenue at the same time. In short, price-based "rationing" is simply a more profitable way to limit sales than quantity-based rationing.

In practice, of course, firms rarely have the luxury of choosing price and quantity flexibility. Therefore, practical business constraints dictate which tactical response—price- or quantity-based RM (or a mixture of both)—is most appropriate in any given business context.

## 3.2. Industry Overview

To give a sense of the scope of activity in the area of dynamic pricing, we next review pricing innovations in a few industries.

**3.2.1. Retailing.** Retailers, especially in apparel and other seasonal-goods sectors, have been at the forefront in deploying science-based software for pricing, driven primarily by the importance of pricing decisions to retailers' profits. For example, Kmart alone wrote off $400 million due to markdowns in one quarter of 2001, resulting in a 40% decline in its net income (Friend and Walker [33]).

Several software firms specializing in RM in retailing have recently emerged. Most of this software is currently oriented toward optimizing markdown decisions. Demand models fit to historical point-of-sale data, together with data on available inventory, serve as inputs to optimization models that recommend the timing and magnitude of markdown decisions.

Major retailers—including Gymboree, J. C. Penney, L. L. Bean, Liz Claiborne, Safeway, ShopKo, and Walgreen's—are experimenting with this new generation of software (Friend and Walker [33], Girard [38], Johnson [45], and Merrick [59]). Many have reported significant improvements in revenue from using pricing models and software. For example, ShopKo reported a 24% improvement in gross margins as a result of using its model-based pricing software (Johnson [45]), and other retailers report gains in gross margins of 5% to 15% (Friend and Walker [33]). Academic studies based on retail data have also documented significant improvements in revenues using model-based markdown recommendations (Bitran et al. [15] and Heching et al. [40]).

**3.2.2. Manufacturing.** Scientific approaches to pricing are gaining acceptance in the manufacturing sector as well. For example, Ford Motor Co. reported a high-profile implementation of pricing-software technology to support pricing and discounts for its products (Coy [20]). The project, which started in 1995, focused on identifying features that customers were most willing to pay for and changing salesforce incentives to focus on profit margins rather than unit-sale volumes. Ford then applied pricing models developed by an

outside consulting firm to optimize prices and dealer and customer incentives across its various product lines. In 1998, Ford reported that the first five U.S. sales regions using this new pricing approach collectively beat their profit targets by $1 billion, while the 13 that used their old methods fell short of their targets by about $250 million (Coy [20]).

**3.2.3. E-business.** E-commerce has also had a strong influence on the practice of pricing (van Ryzin [79]). Companies such as eBay and Priceline have demonstrated the viability of using innovative pricing mechanisms that leverage the capabilities of the Internet. E-tailers can discount and mark down on the fly based on customer loyalty and click-stream behavior. Because a large e-tailer like Amazon.com has to make a large number of such pricing decisions based on real-time information, automating decision making is a natural priority. The success of these e-commerce companies—inconsistent and volatile as it may appear at times—is at least partly responsible for the increased interest among traditional retailers in using more innovative approaches to pricing.

On the industrial side, e-commerce pricing has been influenced by the growth of business-to-business (B2B) exchanges and other innovations in using the Internet to gain trading efficiencies. While this sector too has had its ebbs and flows, it has produced an astounding variety of new pricing and trading mechanisms, some of which are used regularly for the sale products such as raw materials, generic commodity items, and excess inventory. For example, Freemarkets has had significant success in providing software and service for industrial-procurement auctions, and as of this writing claims to have facilitated over $30 billion in trade since its founding in 1999. Covisint—an exchange jointly funded by Daimler-Chrysler, Ford Motor Company, and General Motors—while slow to develop, looks nevertheless to become a permanent feature of the auto-industry procurement market. Most infrastructure software for B2B exchanges—sold by firms such as Ariba, i2, IBM, and Commerce One—also has various forms of dynamic pricing capabilities built in.

For all these reasons, e-commerce has given price-based RM a significant boost in recent years.

## 3.3. Examples of Dynamic Pricing

We next examine three specific examples of dynamic pricing and the qualitative factors driving price changes in each case.

**3.3.1. Style-Goods Markdown Pricing.** Retailers of style and seasonal goods use markdown pricing to clear excess inventory before the end of the season. This type of price-based RM is most prevalent in apparel, sporting goods, high-tech, and perishable-foods retailing. The main incentive for price reductions in such cases is that goods perish or have low salvage values once the sales season is over; hence, firms have an incentive to sell inventory while they can, even at a low price, rather than salvage it.

However, apart from inventory considerations, there are other proposed explanations for markdown pricing. One explanation, proposed by Lazear [52] and investigated empirically in Pashigan [63] and Pashigan and Bowen [64], is that retailers are uncertain about which products will be popular with customers. Therefore, firms set high prices for all items initially. Products that are popular are the ones for which customers have high reservation prices, so these sell out at the high initial price. The firm then identifies the remaining items as low-reservation-price products and marks them down. In this explanation, markdown pricing serves as a form of demand learning.

A second explanation for markdowns is that customers who purchase early have higher willingness to pay, either because they can use the product for a full season (a bathing suit at the start of summer) or because there is some cache to being the first to own it (a new dress style or electronic gadget). Markdown pricing then serves as a segmentation mechanism to separate price-insensitive customers from those price-sensitive customers willing to defer consumption to get a lower price.

Warner and Barsky [80] give yet another explanation, with empirical evidence, for markdown pricing. On holidays and during peak shopping periods (such as before Christmas), customers can search for the lowest prices more efficiently because they are actively engaged in the search, making many shopping trips over a concentrated period of time. Even those customers who normally do not spend much time searching for the best price change their behavior during these peak shopping periods and become more vigilant. The result is that demand during peak periods is more price sensitive and retailers respond by running "sales" during these periods.

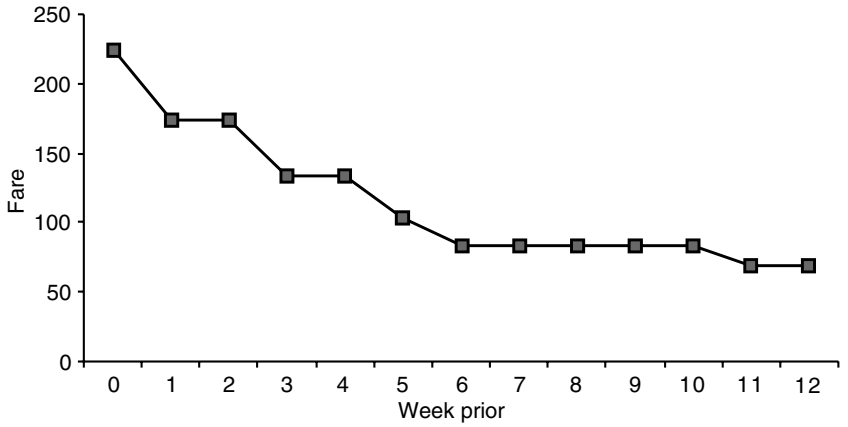
**3.3.2. Discount Airline Pricing.** Not all dynamic pricing involves price reductions, however. As we mentioned earlier, discount airlines use primarily price-based RM, but with prices often going up over time. These airlines (some examples are easyJet and Ryanair in Europe and jetBlue in the United States) typically offer only one type of ticket on each flight, a nonrefundable, one-way fare without advance-purchase restrictions. However, they offer these tickets at different prices for different flights, and moreover, during the booking period for each flight, vary prices dynamically based on capacity and demand for that specific departure. To quote from one practitioner of this type of dynamic pricing (Easyjet website, 2003):

> The way we structure our fares is based on supply and demand, and prices usually increase as seats are sold on every flight. So, generally speaking, the earlier you book, the cheaper the fare will be. Sometimes, however, due to market forces our fares may be reduced further. Our booking system continually reviews bookings for all future flights and tries to predict how popular each flight is likely to be.

Figure 5 shows the evolution of prices for a particular European discount airline flight as a function of the number of weeks prior to departure. Note that prices are highest in the last few weeks prior to departure.

There are some fundamental differences between air travel and style- and seasonal-goods products that explain this increasing price pattern. For one, the value of air travel to customers does not necessarily go down as the deadline approaches. Conversely, the value of a ticket earlier on is lower for customers as customers multiply the value by the probability that they will indeed use the ticket (especially for a nonrefundable ticket). Somewhat related to these points, additionally, although customers purchase tickets at different points of time, all customers consume the product (fly the flight) at the same time. Therefore, two factors come into play. Customers who purchased early may get upset to see prices drop while they are still holding a reservation; indeed, many airlines give a price guarantee to refund the

FIGURE 5. Prices as a function of weeks prior to departure at a European low-cost discount air carrier.



difference if there is a price drop (to encourage passengers to book early), making it costly for the firms to lower prices. Also, in the travel business, high-valuation high-uncertainty customers tend to purchase closer to the time of service. Hence, demand is less price sensitive close to the time of service.

**3.3.3. Consumer Packaged-Goods Promotions.** In contrast to markdown and discount airline pricing, promotions are short-run, temporary price reductions. Promotions are the most common form of price-based RM in the consumer packaged-goods (CPG) industry (soap, diapers, coffee, yogurt, and so on).

The fact that customers purchase CPG products repeatedly has important implications for pricing and promotions. Specifically, customers are aware of past prices and past promotions, so running promotions too frequently may condition customers to view the brand as a frequently discounted product, cutting into brand equity in the long run. Because customers are aware of past prices, promotions impact their subjective "reference price"—or sense of the "fair" price—for products. Also, customers may *stockpile* products, so short-run increases in demand due to promotions may come at the expense of reduced future demand.

The institutional structure of promotions is also more complicated. There are three parties involved—manufacturers, retailers, and end customers. Promotions are run either by a manufacturer as discounts to retailers (trade promotions), which may or may not be passed on to the customers by the retailers (retailer pass-thru), or by retailers (retail promotions or consumer promotions). In some forms of promotion (e.g., mail-in coupons) manufacturers give a discount directly to the end customer.

The motivations of the manufacturer and the retailer are different as well. While a manufacturer is interested in increasing sales or profits for its brand, retailers are interested in overall sales or profits for a category constituting multiple brands from multiple manufacturers. For a retailer, discounting a particular brand may increase sales for that brand but dilute overall category profits as customers switch from high-margin brands to the discounted brand. Therefore, in designing optimal promotions structures, one has to consider complex incentive compatibility constraints.

## 3.4. Modeling Dynamic Price-Sensitive Demand

Any dynamic-pricing model requires a model of how demand—either individual or aggregate—responds to changes in price. The basic theory of consumer choice and the resulting market response models from economics and marketing are used here. However, in dynamic-pricing problems some additional factors must be considered. The first concerns how individual customers behave over time—what factors influence their purchase decisions and how sophisticated their decision-making process is, and so on. The second concerns the state of market conditions—specifically the level of competition and the size of the customer population. We next look at each of these assumptions qualitatively.

**3.4.1. Myopic-Customer vs. Strategic-Customer Models.** One important demand-modeling assumption concerns the level of sophistication of customers. Most of the models we consider in this section assume *myopic customers*—those who buy as soon as the offered price is less than their willingness to pay. Myopic customers do not adopt complex buying strategies, such as refusing to buy in the hope of lower prices in the future. They simply buy the first time the price drops below their willingness to pay. Models that incorporate *strategic customers*, in contrast, allow for the fact that customers will optimize their own purchase behavior in response to the pricing strategies of the firms.

Of course, the strategic-customer model is more realistic. However, such a demand model makes the pricing problem essentially a strategic game between the customers and the firm, and this significantly complicates the estimation and analysis of optimal pricing strategies—often making the problem intractable. In contrast, the myopic-customer model is much more tractable, and hence is more widely used. The issue in practice is really a matter of how "bad"

the myopic assumption is in any given context. In many situations, customers are sufficiently spontaneous in making decisions that one can ignore their strategic behavior. Moreover, customers often do not have sufficient time or information to behave very strategically. However, the more expensive and durable the purchase, the more important it becomes to model strategic-customer behavior (for example, automobile buyers waiting to purchase at the end of a model year).

One common defense of the myopic assumption is the following. The forecasting models that use observations of past customer behavior in a sense reflect the effects of our customers' strategic behavior. For example, if the customers who are most price sensitive tend to adopt a strategy of postponing their purchases until end-of-season clearance sales, then the estimated price sensitivity in these later periods will tend to appear much higher than in earlier periods. Therefore, even though we do not model the strategic behavior directly, our forecasting models indirectly capture the correct price response.

This view is plausible if the pricing strategies obtained from a model are roughly similar to past policies, so that they can be viewed as "perturbations" or "fine-tuning" of a historical pricing strategy—a strategy that customers have already factored into their behavior. On the other hand, if optimized pricing recommendations are radically different in structure from past pricing strategies, then it is reasonable to expect that customers will adjust their buying strategies in response. If this happens, the predictions of myopic models that are fit to historical data may be very bad indeed.

Yet even when the myopic approach works (in the sense of correctly predicting price responses), it runs the risk of reinforcing "bad equilibrium" pricing strategies. For example, a myopic model fit to past data may reconfirm the "optimality" of lowering prices significantly at the end of a sales season or running periodic holiday sales because it estimates, based on historical data, that demand is especially price sensitive in these periods. However, this price sensitivity may be due to the fact that customers have learned not to buy at other times because they know prices will be cut at the end of the season or during holidays. If the firm was to adopt a constant price strategy—and customers were convinced that the firm was sticking to this strategy—then the observed price sensitivity might shift. The resulting equilibrium might be more profitable, but it is one that the firm would not discover using a myopic-customer model.

Despite these limitations and potential pitfalls of the myopic model, it is practical, is widely used, and provides useful insight into dynamic pricing. We therefore focus on the myopic case for the most part in this section. However, we consider strategic customers in §3.7.2.

**3.4.2. Infinite-Population vs. Finite-Population Models.** Another important assumption in demand modeling is whether the population of potential customers is finite or infinite. Of course, in reality, every population of customers is finite; the question is really a matter of whether the number and type of customers that have already bought changes one's estimate of the number or type of future customers.

In an infinite-population model, we assume that we are sampling *with replacement* when observing customers. As a result, the distribution of the number of customers and the distribution of their willingness to pay is not affected by the past history of observed demand. This is often termed the *nondurable-goods assumption* in economics because we can view this as a case where customers immediately consume their purchase and then reenter the population of potential customers (say, for a can of Coke). This assumption is convenient analytically because one does not need to retain the history of demand (or a suitable sufficient statistic) as a state variable in a pricing-optimization problem.

The finite-population model assumes a random process *without replacement*. That is, there is a finite (possibly random) number of customers with heterogeneous willingness-to-pay values. If one of the customers in the population purchases, the customer is removed from

the population of potential customers, and therefore future purchases only occur from the remaining customers. This is termed the *durable-goods assumption* in economics because we can consider it as a case where the good being purchased is consumed over a long period of time (for example, an automobile), and hence once a customer purchases, he effectively removes himself from the population of potential customers.

For example, suppose we assume that a price $p(t)$ is offered in period $t$ and all customers who value the item at more than $p(t)$ purchase in period $t$ (myopic behavior). Then, under a finite-population model, we know that after period $t$, the remaining customers all have valuations less than $p(t)$. In particular, the future distribution of willingness to pay is conditioned on the values being less than $p(t)$. As a result, in formulating a dynamic-pricing problem, we have to keep track of past pricing decisions and their effect on the residual population of customers.

Which of these models is most appropriate depends on the context. While often the infinite-population model is used simply because it is easier to deal with analytically, the key factors in choosing one model over the other are the number of potential customers relative to the number that actually buy and the type of good (durable versus nondurable). Specifically, the infinite-population model is a reasonable approximation when there is a large population of potential customers and the firm's demand represents a relatively small fraction of this population, because in such cases the impact of the firm's past sales on the number of customers and the distribution of their valuations is negligible. It is also reasonable for consumable goods. However, if the firm's demand represents a large fraction of the potential pool of customers or if the product is a durable good, then past sales will have a more significant impact on the statistics of future demand, and the finite-population assumption is more appropriate.

Qualitatively, the two models lead to quite different pricing policies. Most notably, finite-population models typically lead to *price skimming* as an optimal strategy, in which prices are lowered over time in such a way that high-valuation customers pay higher prices earlier, while low-valuation customers pay lower prices in later periods. Effectively, this creates a form of second-degree price discrimination, segmenting customers with different values for the good and charging differential prices over time. In infinite-population models, there is no such price-skimming incentive. Provided the distribution of customer valuations does not shift over time, the same price that yields a high revenue in one period will yield a high revenue in later periods, and thus a firm has no incentive to deviate from this revenue-maximizing price.

**3.4.3. Monopoly, Oligopoly, and Perfect-Competition Models.** Another key assumption in dynamic-pricing models concerns the level of competition the firm faces. Many pricing models used in RM practice are *monopoly models*, in which the demand a firm faces is assumed to depend only on its own price and not on the price of its competitors. Thus, the model does not explicitly consider the competitive reaction to a price change. Again, one makes this assumption primarily for tractability, and it is not always realistic.

As with the myopic-customer model, the monopoly model can be partly justified on empirical grounds—namely, that an observed historical price response has embedded in it the effects of competitors' responses to the firm's pricing strategy. So, for instance, if a firm decides to lower its price, the firm's competitors might respond by lowering their prices. With market prices lower, the firm and its competitors see an increase in demand. The observed increase in demand is then measured empirically and treated as the "monopoly" demand response to the firm's price change in a dynamic-pricing model—even though competitive effects are at work.

Again, while such a view is pragmatic and reflects the conventional wisdom behind the pricing models used in practice, there are some dangers inherent in it, paralleling those of the myopic-customer model. The price-sensitivity estimates may prove wrong if the optimized

strategy deviates significantly from past strategies because then the resulting competitive response may be quite different from the historical response. Also, the practice runs the risk of reinforcing "bad" equilibrium responses. Despite these risks, monopoly models have still proved to be valuable for decision support.

It is worth noting that oligopoly models, in which the equilibrium-price response of competitors is explicitly modeled and computed, also have their pitfalls. Most notably, the assumption that firms behave rationally (or quasi-rationally, if heuristics are used in place of optimal strategies) may result in a poor predictor of their actual price response. These potential modeling errors, together with the increased complexity of analyzing oligopoly models and the difficulty in collecting competitor data to estimate the models accurately have made them less popular in practice. Shugan [68] provides a good summary of this point of view; he notes that "the strong approximating assumption of no competitive response is sometimes better than the approximating assumption of pre-existing optimal behavior." However, properly designed and validated, oligopoly models can provide valuable insights on issues of pricing strategy.

Finally, one can also consider perfectly competitive models—in which many competing firms supply an identical commodity. The output of each firm is assumed to be small relative to the market size, and this, combined with the fact that each firm is offering identical commodities, means that a firm cannot influence market prices.[13] Therefore, each firm is essentially a *price taker*—able to sell as much as it wants at the prevailing market price, but unable to sell anything at higher prices. Despite the importance of perfect-competition models in economic theory, the assumption that firms have no pricing power means that the results are not that useful for price-based RM. Nevertheless, they do play a role in quantity-based RM. For example, one can interpret the capacity-control models of Section 2 as stemming from competitive, price-taking models; firms take the price for their various products as given (set by competitive market forces), and control only the quantity they supply (the availability or allocation) at these competitive prices. As our focus in this section is on price-based RM, we do not consider this model of competition further in this section.

### 3.5. Basic Single-Product Dynamic Pricing Without Replenishment

The first problem we look at is dynamic pricing of a single product over a finite sales horizon given a fixed inventory at the start of the sales horizon. We assume that the firm is a monopolist, customers are myopic, and there is no replenishment of inventory.

The models are representative of the type used in style and seasonal-goods retail RM. For such retailers, production and ordering cycles are typically much larger than the sales season, and the main challenge is to determine the price path of a particular style at a particular store location, given a fixed set of inventory at the beginning of the season.

At one level, such models are simplistic: They consider only a single product in isolation and assume customers are myopic, and therefore demand is a function solely of time and the current price (although other factors such as inventory depletion are sometimes included). They therefore ignore competition, the impact of substitution, and the possible strategic behavior of customers over time. Despite these simplifications, the models provide good rough-cut approximations and are useful in practice. In addition, by decomposing the problem and treating products independently, it is possible to solve such models efficiently even when there are hundreds of thousands of product-location combinations. Finally, even with the simplifying assumptions, the analysis can still become complex if we allow stochastic demand and put constraints on prices.

[13] This is in contrast to the Cournot model of quantity competition, in which there is only a small number of firms whose quantity decisions do affect the market price. Roughly speaking, Cournot competition approaches perfect competition, as the number of firms in the industry tends to infinity.

Because we consider only a single product, there is a single (scalar) price decision at each time $t$, denoted $p(t)$, which induces a unique (scalar) demand rate $d(t,p)$. The set of allowable prices is denoted $\Omega_p$, and $\Omega_d$ denotes the set of achievable demand rates. We assume that these functions satisfy the following regularity conditions:

- The demand functions are continuously differentiable and strictly decreasing, $d'(t,p) < 0$, on $\Omega_p$. Hence, they have an inverse, denoted $d(t,p)$.
- The demand functions are bounded above and below and tend to zero for sufficiently high prices—namely,

$$\inf_{p\in\Omega_p} d(t,p) = 0.$$

- The revenue functions $r(t,p) = pd(t,p)$ (equivalently $r(t,d) = dp(t,d)$) are finite for all $p \in \Omega_p$ and have a finite maximizer interior to $\Omega_p$.
- The marginal revenue as a function of demand, $d$, defined by

$$J(t,d) \equiv \frac{\partial}{\partial d} r(t,d) = p(t,d) + dp'(t,d),$$

is strictly decreasing in $d$.

The demand function can also be expressed as $d(t,p) = N_t(1 - F(t,p))$, where $N_t$ is the market-size parameter and $F(t,p)$ is the fraction of the market with willingness to pay less than $p$. We let $x(t)$ denote the inventory at time $t = 1,\ldots,T$, where $T$ is the number of periods in the sale horizon. The initial inventory is $x(0) = C$.

**3.5.1. The Model.** The simplest deterministic pricing model is formulated in discrete time as follows. Given an initial inventory $x(0) = C$, select a sequence of prices $p(t)$ (inducing demand rates of $d(t,p(t))$) that maximize total revenues. Formulating the problem in terms of the demand rates $d(t)$, the optimal rates $d^*(t)$ must solve

$$\begin{aligned} \max \quad & \sum_{t=1}^{T} r(t,d(t)) \\ \text{s.t.} \quad & \sum_{t=1}^{T} d(t) \le C \\ & d(t) \ge 0. \end{aligned} \tag{26}$$

Let $\pi^*$ be the Lagrange multiplier on the inventory constraint, and recall that $J(t,d) = \frac{\partial}{\partial d} r(t,d)$ denotes the marginal revenue. Then the first-order necessary conditions for the optimal rates $d^*(t)$ and multiplier $\pi^*$ are

$$J(t,d^*(t)) = \pi^*, \tag{27}$$

subject to the complementary slackness condition

$$\pi^* \left( C - \sum_{t=1}^{T} d^*(t) \right) = 0 \tag{28}$$

and the multiplier nonnegativity constraint $\pi^* \ge 0$. Assuming that $J(t,d)$ is decreasing in $d$, $r(t,d)$ is concave; hence, these conditions are also sufficient.

The optimality conditions are quite intuitive. The Lagrange multiplier $\pi^*$ has the interpretation as the marginal opportunity cost of capacity. The condition $J(t,d^*(t)) = \pi^*$ says that the marginal revenue should equal the marginal opportunity cost of capacity in each period. This makes sense, because if marginal revenues and costs are not balanced, we can increase revenues by reallocating sales (by adjusting prices) from a period of low marginal

TABLE 5. Allocations of capacity between periods 1 and 2 and the marginal values and total revenue.

| $d_1$ | $d_2$ | $J(1,d_1)$ | $J(2,d_2)$ | $r$ |
|---|---|---|---|---|
| 22 | 18 | 56 | 42 | 2634 |
| 23 | 17 | 54 | 43 | 2646.5 |
| 24 | 16 | 52 | 44 | 2656 |
| 25 | 15 | 50 | 45 | 2662.5 |
| 26 | 14 | 48 | 46 | 2666 |
| **27** | **13** | **46** | **47** | **2666.5** |
| 28 | 12 | 44 | 48 | 2664 |
| 29 | 11 | 42 | 49 | 2658.5 |
| 30 | 10 | 40 | 50 | 2650 |
| 31 | 9 | 38 | 51 | 2638.5 |
| 32 | 8 | 36 | 52 | 2624 |
| 33 | 7 | 34 | 53 | 2606.5 |

revenue to a period of higher marginal revenue. Finally, the complementary slackness condition says that the opportunity cost cannot be positive if there is an excess of stock. If the opportunity cost is zero ($\pi^* = 0$), then if we maximize revenue without a constraint in every period (pricing to the point where marginal revenue is zero), we will still not exhaust the supply. This means it can be optimal—even in the absence of any costs for capacity—not to sell all the available supply.

Note that this problem is essentially equivalent to the problem of optimal third-degree price discrimination if we consider customers in each period $t$ to be different segments who are offered discriminatory prices $p(t)$. Another way of viewing the above argument is that the firm, faced with a capacity constraint, decides how much to sell in each period, and its optimal allocation of capacity occurs when the marginal revenue in all the periods is the same. The following example illustrates the idea:

**Example 3.** Consider a two-period selling horizon, where during the first period demand is given by $d_1 = -p_1 + 100$ and in period 2 demand is given by $d_2 = -2p_2 + 120$. (Customers in the second period are more price sensitive than those in the first period.) Purchase behavior is assumed to be myopic. Considered separately, the revenue-maximizing price for the first period (maximizing $r_1 = p_1(-p_1 + 100)$) is given by $p_1^* = 50$ and $d_1^* = 50$, and in the second period by $p_2^* = 30, d_2^* = 60$ (maximizing $r_2 = p_2(-2p_2 + 120)$).

Intertemporal effects come into play if the firm has only a limited number of items to sell (less than 50+60). Suppose the firm's capacity is 40. How should it divide the sale between the two periods?

Note that here, $J(1,d_1) = -2d_1 + 100$ and $J(2,d_2) = -d_2 + 60$. Consider the table of marginal values, Table 5, at various allocations and the corresponding revenues. The total revenue is maximized at the point where the marginal values for the two periods are approximately the same (when $d_1 = 27, d_2 = 13$), conforming to our intuition; if they were not equal, the firm would reallocate capacity to the higher marginal-value period.

To see qualitatively how prices will change over time, we can write the optimality condition (27) as

$$\frac{p^*(t) - \pi^*}{p^*(t)} = \frac{1}{|\epsilon(t,p^*)|},$$

where $\epsilon(t,p)$ is the elasticity of demand in period $t$, defined by

$$\epsilon(t,p) \equiv \frac{p}{d(t,p)} \frac{\partial d(t,p)}{\partial p}.$$

Thus, more elastic demand in period $t$ implies a lower optimal price $p^*(t)$.

For example, if customers that buy toward the end of the sales horizon are more price sensitive than those that buy early, then optimal prices will decline over time. If customers early on are price sensitive, and those buying later are less price sensitive, then optimal prices will increase over time. This observation offers one explanation for why in some industries (such as apparel retailing) prices tend to decline over time, while in others (such as airlines) prices increase over time.

*Discrete Prices.* Often, in practice, we would like to choose prices from a discrete set. For example, prices close to convenient whole dollar amounts (such as \$24.99 or \$149.99), or fixed percentage markdowns (such as 25% off or 50% off) are often used because they are familiar to customers and easy to understand. In such cases, it may be desirable as a matter of policy to constrain prices to a finite set of $k$ discrete price points, so that $p(t) \in \Omega_p$, where $\Omega_p = \{p_1, \ldots, p_k\}$. Equivalently, the sales rate $d(t)$ is constrained to a discrete set $d(t) \in \Omega_d(t)$ (time varying in this case if the demand function is time varying), where $\Omega_d(t) = \{d_1(t), \ldots, d_k(t)\}$, and $d_i(t) = d(t, p_i)$ denotes the sales rate at time $t$ when using the price $p_i$.

The discreteness of the prices imposes technical complications when attempting to solve the dynamic pricing problem (26) because the problem is no longer continuous or convex. However, one can overcome this difficulty by relaxing the problem to allow the use of convex combinations of the discrete prices (or demand rates). In most periods, the optimal solution will be to use only one of the discrete prices; in the remaining periods, the solution has the interpretation of allocating a fraction of time to each of several prices.

To see this, define a vector of new variables $\alpha_i(t)$ for each $t$, $\boldsymbol{\alpha}(t) = (\alpha_1(t), \ldots, \alpha_k(t))$, which represent convex weights: They are nonnegative and sum to one. Next, in each period replace the variable $d(t)$ with the convex combination

$$d(t) = \sum_{i=1}^{k} \alpha_i(t) d_i(t),$$

and replace the constraint $d(t) \in \Omega_d(t)$ with the constraint

$$\boldsymbol{\alpha}(t) \in W \equiv \left\{ \boldsymbol{\alpha} \in \Re^k : \sum_{i=1}^{k} \alpha_i = 1, \alpha \geq 0 \right\}.$$

The optimization problem is then

$$\begin{aligned} &\max_{\boldsymbol{\alpha}(t) \in W} \sum_{t=1}^{T} \sum_{i=1}^{k} r_i(t) \alpha_i(t) \\ &\text{s.t.} \quad \sum_{t=1}^{T} \sum_{i=1}^{k} \alpha_i(t) d_i(t) \leq C, \end{aligned} \tag{29}$$

where $r_i(t) = p_i d_i(t)$ is the revenue rate at price $p_i$. This is a linear program in the variables $\boldsymbol{\alpha}(t)$, so it is easy to solve numerically.

To relate the solution to the unconstrained price case, introduce a dual variable $\pi^*$ on the capacity constraint as before. The optimal solution $\boldsymbol{\alpha}^*(t)$ in each period is then characterized by solving

$$\max_{\boldsymbol{\alpha}(t) \in W} \left\{ \sum_{i=1}^{k} \alpha_i(t)(r_i(t) - \pi^* d_i(t)) \right\}, \tag{30}$$

where $\pi^* \geq 0$ and $\boldsymbol{\alpha}^*(t)$ are convex weights satisfying the complementary slackness condition

$$\pi^* \left( \sum_{t=1}^{T} \sum_{i=1}^{k} \alpha_i^*(t) d_i(t) - C \right) = 0. \tag{31}$$

Because the objective function of (30) is linear in $\alpha(t)$, if there is a unique index $i^*$ for which $r_{i^*}(t) - \pi^* d_{i^*}(t)$ is greatest, then the optimal solution is simply $\alpha_{i^*}(t) = 1$, which corresponds to using the discrete price $p_{i^*}$. If there is more than one such value $i^*$, then there will be multiple solutions to (30), and determining which is optimal can be resolved by appealing to the complementary slackness condition (31). Of course, such a choice could result in a fractional solution in which $\alpha_i(t) > 0$ for two or more values $i$. However, this can be interpreted as saying that we should use the price $i$ for a fraction $\alpha_i(t)$ of period $t$. Hence, the solution of (29) can be converted in practice into a discrete-price recommendation.

*Inventory-depletion effect.* Another practical factor affecting dynamic pricing in many retailing contexts is the adverse effects of low inventory levels. This is sometimes referred to in retailing as a *broken-assortment effect*. For example, if the inventory-pricing model is applied at an aggregate item level, where an item contains several SKUs—such as color-size combinations in apparel retailing—then when inventories run low, certain SKUs may be out of stock even though there is a positive inventory for the item as a whole (for example, if a color or size runs out). The resulting reduction in alternatives naturally reduces the sales rate at any given price. Indeed, empirical studies have confirmed a positive correlation between inventory levels and sales rates (Bhat [12]).

These inventory-depletion effects can be modeled by making the demand rate a function of inventory as well as of price and time, so that the demand rate becomes $d(t, p(t), x(t))$. We can use a variety of functional forms to represent this inventory-depletion effect. For example, one proposed model is the following multiplicative form (Smith and Achabal [70]):

$$\hat{d}(t, x(t)) = d(t)g(x(t)), \tag{32}$$

where $g(\cdot)$ is a depletion-effect term. We will call $d(t)$ the *unadjusted sales rate* (the rate of sales if inventory were unlimited) and $\hat{d}(t, x(t))$ the *adjusted sales rate* (the rate adjusted for inventory-depletion effects). One choice for $g$ is

$$g(x) = 1 - \gamma \max\{0, 1 - x/x_0\},$$

where $x_0$ is the minimum *full-fixture inventory* and $0 \le \gamma \le 1$ is a sensitivity parameter. Both $x_0$ and $\gamma$ can be estimated from historical data. Note that $g(x)$ is concave in $x$.

Another possible form is

$$g(x) = e^{-\gamma \max\{0, 1 - x/x_0\}},$$

where $\gamma$ and $x_0$ have the same interpretation (see Smith and Achabal [70]).

For this model with inventory depletion one must keep track of the inventory at each time $t$ in the optimization problem. For example, assuming the multiplicative inventory-depletion model of (32) and formulating the problem in terms of the unadjusted sales rate $d(t)$, the inventory evolves according to the state equation

$$x(t+1) = x(t) - d(t)g(x(t)),$$

and the revenue-maximization problem can be formulated as

$$\begin{aligned} \max_{d(t) \ge 0} \ & \sum_{t=1}^{T} r(t, d(t)) g(x(t)) \\ \text{s.t.} \ & x(t+1) = x(t) - d(t)g(x(t)), \quad t = 1, \ldots, T \\ & x(T) \ge 0, \\ & x(0) = C, \end{aligned} \tag{33}$$

where $r(t, d(t)) = p(t, d(t))d(t)$ is the unadjusted revenue-rate function.

While somewhat more complex than the case without inventory-depletions effects, this is still a relatively simple nonlinear program to solve because the objective function is separable and the constraints are linear. (The objective function, however, is not necessarily jointly concave even if $r(t, d(t))$ and $g(x)$ are both concave.)

One qualitative impact of this inventory-depletion phenomenon is that optimal prices may decline over time even though the unadjusted revenue-rate function is time invariant. (Recall that in the problem without inventory-depletion effects, a time-invariant revenue-rate function implied a time-invariant optimal price.) For example, Smith and Achabal [70] show, for the continuous-time version of this model, that if the unadjusted revenue-rate function is constant and the inventory-depletion effect is multiplicative, then optimal prices decline over time in such a way that the adjusted sales rate $g(x(t))d(t)$ is constant; that is, as inventory depletion reduces demand, the optimal prices fall to exactly compensate for the drop in sales due to inventory depletion.

## 3.6. Multiproduct, Multiresource Pricing

Multiproduct, multiresource—or network—versions of dynamic-pricing problems arise in many applications. Two fundamental factors typically link the pricing decisions for multiple products. First, demand for products may be correlated. For example, when products are substitutes or complements, the price charged for one product affects the demand for other related products. Then, a firm jointly managing the pricing of a family of such products must consider these cross-elasticity effects when determining its optimal pricing policy. Second, products may be linked by joint capacity constraints. For example, two products may require the same resource, which is available in limited supply. Even if there are no cross-elasticity effects between the two products, the pricing decision for one product will need to account for the joint effect on demand for the other product that uses the limited resource.

As in the case of capacity controls, most problems in real life are multiproduct problems, either because of cross-elasticity effects or because of joint capacity constraints, or both. For example, a grocery store that is pricing brands in a food category—say, salty snacks—needs to consider the cross-elasticity effects of its pricing decision for all products in the category. An increase in the price of a packet of potato chips will not just cause a drop in demand for potato chips, but will likely also increase the demand for corn chips. At the same time, these products may occupy the same limited shelf space, so stocking more of one product may require stocking less (or none) of other products.

We can model such situations using multiproduct demand functions and joint capacity constraints on resources. However, like the network problems of capacity control, such formulations quickly become difficult to analyze and solve, which is the reason that many commercial applications of dynamic-pricing models make the simplifying assumption of unrelated products and independent demands and solve a collection of single-product models as an approximation.

However, in cases where cross-elasticity or resource-constraint effects are strong—for example, when products are only slightly differentiated, customers are very price sensitive, or joint capacity constraints are tight—then ignoring multiproduct effects can be severely suboptimal. In such cases, we must solve a pricing problem incorporating these effects—or at least approximating them in some fashion. In this section, we look as such multiproduct, multiresource models and methods.

**3.6.1. A Basic Deterministic Model Without Replenishment.** Under a deterministic demand assumption, it is relatively straightforward to formulate a multiproduct, multiresource version of dynamic pricing similar to those described in §3.5. There are $n$ products, indexed by $j$, and $m$ resources, indexed by $i$. There is a horizon of $T$ periods, with each period indexed by $t$. Let $\mathbf{d} = (d_1, \ldots, d_n)$ denote the demand rate for the $n$ products and $\mathbf{p}(t, \mathbf{d})$ denote the inverse-demand function in period $t$. We further assume that the revenue-rate function $r(t, \mathbf{d})$ is bounded and jointly concave in $\mathbf{d}$.

Product $j$ uses a quantity $a_{ij}$ of resource $i$. The matrix $\mathbf{A} = [a_{ij}]$ therefore describes the *bill of materials* for all $n$ products. We assume there are limited capacities $\mathbf{C} = (C_1, \ldots, C_m)$ of the $m$ resources.

The dynamic-pricing problem can then be formulated as finding a sequence of demand vectors $d^*(t)$ that maximizes the firm's total revenue subject to the capacity constraints $C$:

$$\begin{aligned} \max \ & \sum_{t=1}^{T} r(t, \mathbf{d}(t)) \\ \text{s.t.} \ & \sum_{t=1}^{T} \mathbf{A}\mathbf{d}(t) \le \mathbf{C} \\ & \mathbf{d}(t) \ge 0, \quad t = 1, \ldots, T. \end{aligned} \tag{34}$$

We assume $r(t, \mathbf{d})$ is concave in $\mathbf{d}$, and therefore, the following Kuhn-Tucker conditions are necessary and sufficient for characterizing an optimal solution $\mathbf{d}^*(t)$ to (34):

$$J(t, \mathbf{d}^*(t)) = \mathbf{A}^\top \boldsymbol{\pi}^* \tag{35}$$

$$\boldsymbol{\pi}^{*\top} \left( \mathbf{C} - \sum_{t=1}^{T} \mathbf{A}\mathbf{d}(t) \right) = 0 \tag{36}$$

$$\boldsymbol{\pi}^* \ge 0, \tag{37}$$

where $J(t, \mathbf{d}) = \nabla_d r(t, \mathbf{d})$ is the marginal-value vector and $\boldsymbol{\pi}^*$ is the optimal dual price on the joint-capacity constraints, having the usual interpretation as the vector of marginal opportunity costs (marginal values) for the $m$ resources. Condition (35) says that at the optimal sales rate, the marginal revenue for each product $j$ should equal the marginal opportunity cost of the resources used by product $j$, or $\boldsymbol{\pi}^{*\top} \mathbf{A}_j$. Condition (36) says that the marginal opportunity cost of resource $i$ can be positive only if the corresponding capacity constraint for resource $i$ is binding. Finally, (37) requires that the marginal opportunity costs be nonnegative.

The nonlinear program (34) is relatively easy to solve numerically because the objective function is concave and the constraints are linear. (See Bertsekas [9, 10] for specific techniques.)

**Example 4.** Consider the six-node airline network shown in Figure 6. Nodes 2 and 3 are "hub" nodes. (Leg seat capacities are as indicated in the figure.) For a given path $j$ on the network, the revenue function is time homogeneous and log linear

$$d_j(p_j) = a_j e^{-\epsilon_j (p_j / \bar{p}_j - 1)},$$

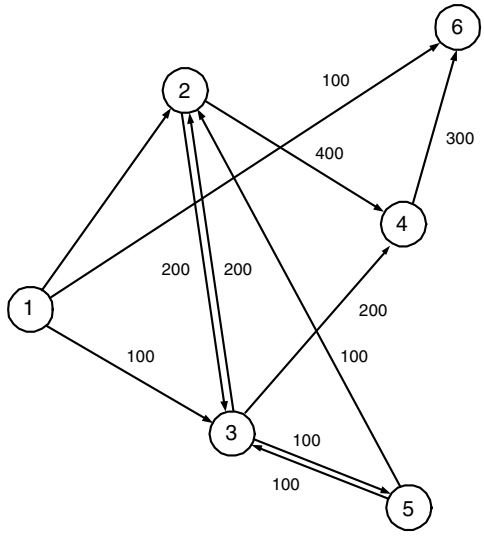
where $\bar{p}_j$ is interpreted as a reference price for itinerary $j$, $a_j$ is the demand rate at the reference price, and $\epsilon_j$ is the magnitude of the elasticity of demand at the reference price. Demand-function parameters for all O-D pairs are shown in Table 6, along with the path (itinerary) used by each O-D pair.

Because the demand functions are time homogeneous, optimal prices are constant over time. The optimal O-D prices and demand are shown in the last two columns in Table 6. The solution gives a total revenue of \$661,200 across all O-D pairs.

**3.6.2. Action-Space Reductions.** One simplification that is useful for multiproduct dynamic-pricing problems is to express the problem in terms of resource-consumption rates rather than the demand rates $\mathbf{d}$. This yields an equivalent formulation with an often greatly reduced dimensionality that can be much easier to solve. The approach is due to Maglaras and Meissner [56].

To illustrate the main idea, consider the case of the deterministic model (34), where there is only $m = 1$ resource but $n > 1$ products. For example, this could be a situation similar

FIGURE 6. A six-node, two-hub airline network.




to the traditional single-resource problem of §2, but one in which we control the demand for each product $j$, $d_j$, by adjusting its price $p_j$. The deterministic problem (34) in this case is then

$$\begin{aligned} \max \ & \sum_{t=1}^{T} r(t, \mathbf{d}(t)) \\ \text{s.t.} \ & \sum_{t=1}^{T} \sum_{j=1}^{n} d_j(t) \le C \\ & \mathbf{d}(t) \ge 0, \quad t = 1, \ldots, T. \end{aligned} \tag{38}$$

TABLE 6. Demand-function parameters, itineraries, and optimal solution for Example 4.

| Market | | Demand function | | | | Optimal solution | |
|---|---|---|---|---|---|---|---|
| O | D | $a_j$ | $\epsilon_j$ | $\bar{p}_j$ | Path | $d_j^*$ | $p_j^*$ |
| 1 | 2 | 300 | 1.0 | 220 | 1–2 | 135 | \$396.62 |
| 1 | 3 | 300 | 1.2 | 220 | 1–3 | 67 | \$495.86 |
| 1 | 4 | 300 | 2.0 | 400 | 1–2–4 | 165 | \$520.11 |
| 1 | 5 | 300 | 1.0 | 250 | 1–3–5 | 33 | \$752.04 |
| 1 | 6 | 300 | 0.8 | 200 | 1–6 | 100 | \$525.58 |
| 2 | 3 | 300 | 1.0 | 230 | 2–3 | 168 | \$364.28 |
| 2 | 4 | 300 | 0.9 | 200 | 2–4 | 143 | \$365.74 |
| 2 | 5 | 300 | 2.0 | 200 | 2–3–5 | 32 | \$423.79 |
| 2 | 6 | 300 | 1.0 | 200 | 2–4–6 | 92 | \$436.80 |
| 3 | 2 | 300 | 1.0 | 200 | 3–2 | 200 | \$281.76 |
| 3 | 4 | 300 | 2.0 | 230 | 3–4 | 131 | \$325.30 |
| 3 | 5 | 300 | 2.0 | 120 | 3–5 | 35 | \$249.51 |
| 3 | 6 | 300 | 2.0 | 150 | 3–4–6 | 14 | \$378.60 |
| 4 | 6 | 300 | 1.0 | 150 | 4–6 | 162 | \$243.30 |
| 5 | 2 | 300 | 1.0 | 200 | 5–2 | 100 | \$420.39 |
| 5 | 3 | 300 | 2.0 | 150 | 5–3 | 47 | \$289.90 |
| 5 | 4 | 300 | 1.0 | 160 | 5–3–4 | 21 | \$585.20 |
| 5 | 6 | 300 | 1.0 | 230 | 5–3–4–6 | 32 | \$748.50 |

To reduce the dimensionality of this problem, we express the problem in terms of the aggregate-demand rate rather than the individual demand rates $\mathbf{d}$. To this end, define the aggregate-demand rate

$$\hat{d} = \sum_{j=1}^{n} d_j,$$

and for a given $\hat{d}$ define the maximized revenue-rate function by

$$\begin{aligned} \hat{r}(t,\hat{d}) = \max\ & r(t,\mathbf{d}) \\ \text{s.t.}\ & \sum_{j=1}^{n} d_j = \hat{d} \\ & \mathbf{d} \geq 0. \end{aligned} \tag{39}$$

That is, $\hat{r}(t,\hat{d})$ is the instantaneous maximum revenue rate given that the total demand rate (equivalently, the resource *consumption rate*) is constrained to be $\hat{d}$. It is easy to show that if $r(t,\mathbf{d})$ is jointly concave in $\mathbf{d}$, then $\hat{r}(t,\hat{d})$ will be concave in $\hat{d}$.

Using these new variables, we can then formulate (38) as

$$\begin{aligned} \max\ & \sum_{t=1}^{T} \hat{r}(t,\hat{d}(t)) \\ \text{s.t.}\ & \sum_{t=1}^{T} \hat{d}(t) \leq C \\ & \hat{d}(t) \geq 0 \quad t = 1,\ldots,T. \end{aligned} \tag{40}$$

Note that this is now a problem that is equivalent to a single-product pricing problem of the same form as (26) with a scalar demand rate $\hat{d}$ and revenue-rate functions $\hat{r}(t,\hat{d})$. Once we solve for the optimal demand rates $\hat{d}^*(t)$, we can then convert these into optimal vectors of demand rates $\mathbf{d}^*(t)$ by inserting $\hat{d}^*(t)$ into the optimization problem (39). Thus, the solution proceeds in two steps: First, solve (40) to determine the optimal aggregate-sales rate, and then solve (39) at each time $t$ to disaggregate this optimal aggregate rate into an optimal vector of sales rates (equivalently prices) for each product. This same action-space-reduction approach also works for stochastic versions of this problem, and it extends to the general multiproduct ($m > 1$), multiresource problem (34) as well.

## 3.7. Finite-Population Models and Price Skimming

We next consider what effect a finite-population assumption has on an optimal dynamic-pricing policy. Recall that a finite-population model assumes that we sample customers without replacement from a finite number of potential customers. Thus, the history of demand (how many customers have purchased, how much they paid, and so on) affects the distribution of both the number and valuations of the remaining customers.

Because the finite-population assumption is more complex, we focus on deterministic models of this situation. However, we consider both a myopic and strategic customer version of the problem.

**3.7.1. Myopic Customers.** Recall that a myopic customer is assumed to purchase the first time the current price $p(t)$ drops below his valuation $v$. Combined with the finite-population assumption, this behavior can be exploited by the firm to achieve *price skimming*—a version of classical second-degree price discrimination.

Assume for simplicity that there is a finite population size $N$ and that customers in this population have valuations $v$ that are uniformly distributed on the interval $[0,\bar{v}]$. As

an approximation, we assume that sales can occur in fractions, so the population can be regarded as continuous. The important point to note is that the fraction of customers who purchased until time $t$ leave the population of customers for the remaining sale period.

As a result of the myopic-customer assumption, if the firm offers a price $p$, $N(1-\bar{v}/p)$, customers will buy. Also, by the finite-population assumption, there will then be $N\bar{v}/p$ remaining customers, with valuations uniformly distributed on the interval $[0,p]$.

Now, consider a firm that sells a fixed capacity $C$ of a product to this population over $T$ time periods. The firm is free to set different prices in each period. What is the optimal pricing strategy?

First, it is not hard to see that the optimal prices are decreasing over time, because (by the myopic-customer assumption) the only customers left at time $t$ are those with values less than the minimum price offered in periods $1,\ldots,t-1$. Hence, the firm will sell nothing if it posts a price in period $t$ that is higher than the minimum price offered in the past. This observation, applied inductively, shows that the optimal prices must decline over time. Moreover, note that if $p(t)\le p(t-1)$ for all $t$, the revenue generated in period $t$ is given by

$$p(t)\frac{N}{\bar{v}}(p(t-1)-p(t)),$$

where we define $p(0)=\bar{v}$. This is because $(N/\bar{v})(p(t-1)-p(t))$ is the number of customers with valuations greater than $p(t)$ but less than the lowest previous price $p(t-1)$.

To see the effect the decreasing price schedule has on the optimal pricing policy, assume for simplicity that $C>N$, so the capacity constraint is never binding. In this case, the firm must solve

$$\max \sum_{t=1}^{T} \frac{N}{\bar{v}} p(t)(p(t-1)-p(t)) \tag{41}$$

$$\text{s.t. } p(t)\le p(t-1), \quad t=1,\ldots,T \tag{42}$$

$$p(0)=\bar{v}, \tag{43}$$

$$p(t)\ge 0. \tag{44}$$

Note that the objective function is jointly concave in $p(t), t=1,\ldots,T$. It is not hard to see that the constraints (42) are redundant, because the objective function (41) will penalize the use of a price $p(t)>p(t-1)$. Therefore, ignoring constraints (42) and defining $p(T+1)=0$, the first-order conditions imply that the optimal unconstrained solution must satisfy

$$p(t)=\frac{p(t-1)-p(t+1)}{2}, \quad t=1,\ldots,T.$$

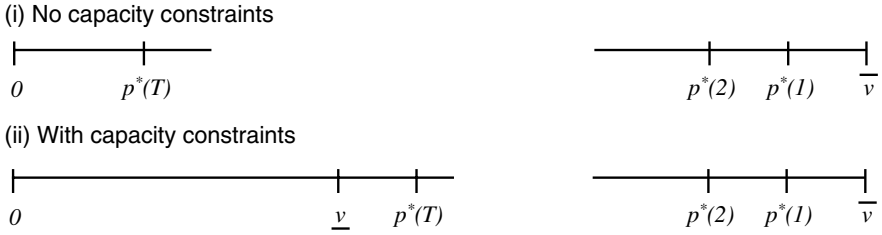
One can easily verify that the solution

$$p^*(t)=\bar{v}\left(1-\frac{t}{T+1}\right) \tag{45}$$

satisfies these first-order conditions. Because the optimization problem (41–d) is strictly concave and (45) satisfies the inequality constraints $p(t)\le p(t-1)$ for all $t$, it is in fact the unique optimal solution for (41–d). This solution is illustrated in Figure 7(i).

The optimal pricing strategy effectively exploits the myopic behavior of customers to segment them into $T+1$ groups based on their valuations, and then price discriminates based on this segmentation. Specifically, as shown in Figure 7, segment $t$ consists of those customers whose valuations are in the range $[p^*(t),p^*(t-1)]$, and these segments pay a declining price $p^*(t)$ given by (45). Segment $T+1$ has values in the range $[0,\bar{v}/(T+1)]$ and is not served at all.

FIGURE 7. Optimal price-skimming solution for myopic customers: (i) no capacity constraints, (ii) with capacity constraints.




There are several interesting observations about this solution. First, note that we can write the optimal price in period $t$ as

$$p^*(t) = \frac{p^*(t-1)}{2} + \frac{\bar{v}}{2}\left(1 - \frac{t}{T+1}\right).$$

The first term on the right, $p^*(t-1)/2$, is simply the single-period revenue-maximizing price, which follows from the fact that the remaining customers in period $t$ have values uniformly distributed on $[0, p^*(t-1)]$. Therefore, the optimal price in period $t$ is higher than the single-period revenue-maximizing price for period $t$ (except in the last period $t = T$, where they are equal). Intuitively, this occurs because there is an additional benefit to the firm of raising its price in period $t$ in the multiperiod setting; namely, it will have more customers to sell to in the future.

Second, note that the price changes over time not because the distribution of valuations changes over time—as in the infinite-population model of demand—but because the firm seeks to price discriminate among the finite population of customers. For example, in an equivalent infinite-population model (essentially, the model of §3.5.1 with a linear demand function), the distribution of values of customers is unaffected by past demand, and hence the distribution would still be uniform over $[0, \bar{v}]$ in each period. In this case, the optimal price to charge in each period would be a constant $\bar{v}/2$ rather than the declining price (45). Therefore, a finite population of customers creates an incentive to offer dynamically decreasing prices to achieve price discrimination, an incentive that is not present in infinite-population models.

Finally, note that if the number of periods $T$ increases, the firm's revenues increase because one can show (after some algebra) that the optimal total revenue for $T$ periods is

$$\sum_{t=1}^{T} p^*(t)\frac{N}{\bar{v}}(p^*(t) - p^*(t-1)) = \frac{N\bar{v}}{2}\left(\frac{T}{T+1}\right).$$

Indeed, as $T$ tends to infinity, the firm achieves perfect price discrimination and captures the entire consumer surplus $N\bar{v}/2 = \int_0^{\bar{v}} (N/\bar{v})dv$; each customer ends up paying a price arbitrarily close to his valuation. In particular, a continuous-time model of this problem can achieve perfect price discrimination because the firm can continuously lower prices from $\bar{v}$ down to zero over the interval $[0, T]$. A number $dp(N/\bar{v})$ of customers with values $[p, p+dp]$ will buy when the price is $p$, so the firm achieves a revenue of $\int_0^{\bar{v}} p(N/\bar{v})dp = N\bar{v}/2$, which is the entire consumer surplus.

**3.7.2. Strategic Customers.** One might question why customers would behave myopically when faced with a price-skimming strategy. Indeed, knowing that prices will decline over time, rational customers could do better (increase their net utility) by deviating from myopic behavior and delaying purchase until the price is much lower than their valuation. Such behavior is quite plausible and is a valid criticism of the myopic-customer model, but it complicates the analysis of the firm's optimal-pricing policy considerably. Here we focus

on the effect of strategic customers on the price-skimming strategy alone. Throughout this section we consider only the case where the firm has no capacity constraint ($C > N$).

To proceed, one first has to make assumptions about whether the firm can credibly commit to a schedule of prices over time or whether the firm must follow a subgame-perfect equilibrium-pricing strategy. In our case, requiring a subgame-perfect equilibrium means that the strategy for the firm at each time $t$ has to be an equilibrium for the residual revenue-maximization game over the horizon $t, t+1, \ldots, T$, given whatever state the firm and customers were in period $t$.

For example, if the firm can commit to a price schedule, then a rational customer will simply look at the schedule of prices and (assuming no discounting of utility) decide to purchase in the period with the lowest price, and only customers with valuations above this lowest price will decide to purchase. So, effectively, it is only the lowest price among the $T$ periods that matters to customers. Given this fact (and ignoring capacity constraints), the firm will then set this minimum price as the single-period revenue-maximizing price, which, in the case where customer valuations are uniformly distributed on $[0, \bar{v}]$, is just $\bar{v}/2$. The firm will then set arbitrary but higher prices in the other periods. Which period the firm chooses for the minimum price does not matter unless revenues are discounted, in which case the firm would prefer collecting revenues sooner rather than later and would choose period 1. The total revenue the firm receives is then $N\bar{v}^2/4$, which is just the product of the price $\bar{v}/2$ and the number of customers willing to pay that price, $N\bar{v}/2$. One can formalize this reasoning and show that this is indeed the equilibrium strategy in the case where the firm has to commit to a price schedule.

Note that the fact that customers are rational has eliminated the ability of the firm to price discriminate; the firm is forced to offer a single uniform price to all customers. Moreover, the firm's revenue is strictly worse under this model. This is to be expected; the firm ought to do worse when customers are "smarter."

However, the single-period strategy outlined above is not always subgame-perfect. To see why, suppose this lowest price $\bar{v}/2$ occurs in period 1. Then in period 2, there will be a population of customers with values less than $\bar{v}/2$ who have not purchased. If the firm has any remaining supply after period $t$, it would rather sell the remaining stock at some positive price than let it go unsold. Thus, it has an incentive to lower the price in period 2 to capture some of the remaining customers. However, rational customers realize the firm faces this temptation after period 1 and, anticipating the price drop, do not purchase in period 1, so offering the lowest price in period 1 cannot be a subgame-perfect equilibrium.

Besanko and Winston [11] analyze the subgame-perfect pricing strategy. The equilibrium is for the firm to lower prices over time, similar to the price-skimming strategy of §3.7.1. In the case where revenues are not discounted, this equilibrium results in the firm setting a declining sequence of prices, where the price in the last period $T$ is simply the single-period optimal price $\bar{v}/2$; all customers buy only in the last period. This case is essentially equivalent to the case where the firm can commit to a schedule of prices, with the exception that the firm is forced to offer the lowest price only in the last period.

The situation is somewhat more interesting if revenues and customer utility are discounted at the same rate. In this case, the subgame-perfect equilibrium has customers with high values buying in the early periods and those with lower values buying in later periods, again, as in the price-skimming case of §3.7.1. However, unlike the price-skimming case, the equilibrium price in each period is *lower* than the single-period revenue-maximizing price for the customers remaining in that period. In particular, in period 1 the equilibrium price is less than $\bar{v}/2$, and the equilibrium price declines in subsequent periods. Thus, the firm is strictly worse off than when it can commit to a price schedule. This is because when the firm can commit to its price schedule, it can force all customers to purchase in period 1 by simply offering very high prices in periods $t > 1$ while setting a price of exactly $\bar{v}/2$ in period 1. All customers will then buy in period 1 at a price of $\bar{v}/2$.

Besanko and Winston [11] show that with strategic customers, the firm is always better off with fewer periods; that is, the firm's equilibrium revenue is decreasing in the number of periods. This is because the inability of the firm to commit to prices in later periods hurts it, and the more periods, the more often the firm falls victim to the temptation to lower prices. That is, it discounts early and often. This is to be contrasted with the case of myopic customers, where the firm's revenues are increasing in the number of periods. Thus, although the strategy looks like price skimming, rational customers create a qualitatively different situation for the firm than do myopic customers.

## 4. Summary and Conclusions

The notion that a firm's demand should be actively managed—and that scientific methods can help improve demand decisions—is the essence of revenue management. In this chapter, we have given an overview of the field, its origins and applications, and a sampling of the models and methods used. Still, there is much more to the subject than what we have presented here. Estimating and forecasting market response, for example, is a vast and important element of revenue management on which we have not touched. There are also many relevant concepts from economics on related topics such as price discrimination, peak-load pricing, mechanism design, and oligopoly pricing that augment the operational theory surveyed here. System implementation is also a vital part of RM practice. These and other topics are covered in our book, *The Theory and Practice of Revenue Management* (Kluwer 2004), for those readers interested in more depth and detail. What makes the subject fascinating is that it is in many ways the quintessential OR topic, combining a vitally important business application with sophisticated techniques and concepts from economics, statistics, and optimization.

In addition, the future of revenue management looks equally bright. The domain of industrial application is spreading rapidly beyond the transportation and hospitality industry. Retailing is already a major industry user of RM. Manufacturing, advertising, energy, and financial services applications are growing. With each new industry application, one encounters new challenges in modeling, forecasting, and optimization, so research in the area is also blossoming. It is an exciting field to follow and be a part of, and will likely stay that way for many years to come.

## Appendix: Notes and Sources

### Introduction and Overview Articles

The 1997 book by Robert Cross, *RM: Hard Core Tactics for Market Domination* (Cross [21]) was influential in popularizing the story of airline RM and introducing the concept of RM to the general business community. Several other books on RM have been published recently; Ingold et al. [42] focuses primarily on the hotel industry, and Daudel and Vialle [24] focuses on air transportation. The book by Nagel and Holden [61] provides a comprehensive overview of many managerial issues involved in pricing, and is useful reading.

Several survey articles provide general coverage of RM. The *Handbook of Airline Economics* edited by Jenkins [44] provides several good practice-oriented articles on RM in the airline industry. Kimes [49] provides a conceptual introduction to RM with a hotel RM focus. Smith et al. [69] provide a nice description of the practice of RM at American Airlines and the DINAMO system.

As for guides to the research literature, Weatherford and Bodily [81] propose a taxonomy for classifying the sets of assumptions used in many traditional RM models, although the taxonomy itself is little used. McGill and van Ryzin [58] provide a comprehensive overview and annotated bibliography of the published academic literature in the field through 1998. Elmaghraby and Keskinocak [27] provide a survey on research in the area of dynamic pricing.

### Single-Resource Capacity Control

The notion of theft versus standard nesting is not well-documented and is part of the folklore of RM practice. Our understanding, however, greatly benefited from discussions with our colleagues Peter Belobaba, Sanne de Boer, and Craig Hopperstad.

The earliest paper on the static models of §2.3 is Littlewood [55]. Another early applied paper is Bhatia and Parekh [13]. However, there are close connections to earlier work on the stock-rationing problem in the inventory literature by Kaplan [47] and Topkis [78]; see also Gerchak and Parlar [36], Gerchak et al. [37], and Ha [39]. Indeed, Topkis's [78] results can be used to show the optimality of nested-allocation policies.

Optimal policies for the $n > 2$ case were obtained in close succession (using slightly different methods and assumptions) in papers by Brumelle and McGill [17], Curry [22], Robinson [67], and Wollmer [84]. See also McGill's thesis [57]. Robinson [67] also analyzed the case where the order of arrival is not the same as the revenue order. Brumelle et al. [19] analyzed a two-class static model with dependent demand.

The dynamic model of §2.4 was first analyzed by Lee and Hersh [53]. Lautenbacher and Stidham [51] provide a unified analysis of both the static and dynamic single-resource models. Walczak and Brumelle [18] relate this problem to a dynamic-pricing problem using a Markov model of demand that allows for partial information on the revenue values or customer types. See Liang [54] for an analysis of a continuous-time version of the dynamic model.

The EMSR-a and EMSR-b heuristics are both due to Belobaba. The most detailed coverage of EMSR-a is contained in Belobaba's 1987 thesis [3], but see also the published articles from it (Belobaba [4, 5]). EMSR-b was introduced in Belobaba [6]; see also Belobaba and Weatherford [7].

The buy-up heuristics in §2.5.1 are due to Belobaba [3, 4, 5]. See also Belobaba and Weatherford [7], Weatherford et al. [82], and the simulation study of Bohutinsky [16]. See Titze and Griesshaber [77] for a discussion of passenger behavior in the simple two-class model. The material on choice-based models in §2.5.2 is from Talluri and van Ryzin [74]; see also Algers and Besser [1] and Andersson [2] for an application of discrete-choice models at SAS. For a good reference on discrete-choice modeling, see Ben-Akiva and Lerman [8]. De Boer [25] is another recent work that addresses customer choice in a single-resource problem.

**Dynamic Pricing**

The book by Nagle [61] provides a good general-management overview of pricing decisions. Elmaghraby and Keskinocak [27] provide a survey on research in the area of dynamic pricing. As for the connection between pricing- and capacity-allocation decisions, see Walczak and Brumelle [18].

Smith and Achabal [70] study a continuous-time version of the problem with inventory-depletion effect, as in §3.5.1. Heching et al. [40] provide revenue estimates based on a regression test of this same type of deterministic model on data from an apparel retailer.

Gallego and van Ryzin [34] analyzed a continuous-time, time-homogeneous stochastic model, providing monotonicity properties of the optimal price, an exact solution in the exponential demand case, and proving the asymptotic optimality of the deterministic policy. Bitran and Mondschein [14] analyze a discrete-time model of this problem and test it on apparel retail data. Zhao and Zheng [87] analyze the continuous-time model with a time-varying demand function and provide an alternative proof of monotonicity of the marginal values. See also Kincaid and Darling [50] and Stadje [71]. Das Varmand and Vettas [23] analyze the problem of selling a finite supply over an infinite horizon with discounted revenues, where the discounting provides an incentive to sell items sooner rather than later and there is no hard deadline on the sales season.

Stochastic models with discrete price changes are analyzed in the continuous-time case in a series of papers by Feng and Gallego [28, 29] and Feng and Xiao [31, 32]. The problems differ in terms of whether there are two prices or more than two prices, whether the price changes are reversible or one-way changes. Feng and Gallego [29] extend the analysis also to the interesting case where demand is Markovian and may depend on the current inventory level—for example, as in the classical Bass model of new-product diffusion. The notion of the *maximum concave envelope* of prices is due to Feng and Xiao [31]. See also You [85] for a discrete-time analysis of the problem.

There is an extensive literature on production-pricing problems, which we have not covered in this chapater. Eliashberg and Steinberg [26] provide of review of joint pricing and production models. Single-period, convex-cost problems under demand uncertainty are analyzed by Karlin and Carr [48], Mills [60], and the early paper of Whitin [83]. The literature on single-period pricing under demand uncertainty (the price-dependent newsvendor problem) is surveyed by Petruzzi and Dada [65]. Multiperiod, convex-cost models are analyzed by Hempenius [41], Thowsen [76], and Zabel [86].

Multiproduct, multiresource dynamic-pricing problems were analyzed in Gallego and van Ryzin [35], including bounds on the relationship between the stochastic and deterministic versions of the

problem. The action-space-reduction approach is a recent result due to Maglaras and Meissner [56]. A related network pricing we have omitted is congestion pricing for communications service; see, for example, Pashalidis and Tsitiklis [62].

Stokey [72] analyzes a model of intertemporal price discrimination similar to that presented in §3.7.1. See also Kalish [46]. Stokey [73] analyzes a price-skimming model with rational customers under the assumption that the firm can commit to a price schedule. The material in §3.7.2 on the subgame-perfect pricing equilibrium for a firm faced with strategic customers is from Besanko and Winston [11].

## Acknowledgment

This tutorial chapter was published originally as *The Theory and Practice of Revenue Management* by Kalyan T. Talluri and Garrett J. van Ryzin [75]. Portions of the original are reprinted here with permission of Springer Science + Business Media.

## References

[1] S. Algers and M. Besser. Modeling choice of flight and booking class: A study using stated preference and revealed preference data. *International Journal of Services Technology and Management* 2:28–45, 2001.

[2] S. E. Andersson. Operational planning in airline business—Can science improve efficiency? Experiences from SAS. *European Journal of Operations Research* 43:3–12, 1989.

[3] P. P. Belobaba. Air travel demand and airline seat inventory management. Ph.D. thesis, Flight Transportation Laboratory, MIT, Cambridge, MA, 1987.

[4] P. P. Belobaba. Airline yield management: An overview of seat inventory control. *Transportation Science* 21:63–73, 1987.

[5] P. P. Belobaba. Application of a probabilistic decision model to airline seat inventory control. *Operations Research* 37:183–197, 1989.

[6] P. P. Belobaba. Optimal vs. heuristic methods for nested seat allocation. *ORSA/TIMS Joint National Meeting*, San Francisco, CA (November) 1992.

[7] P. P. Belobaba and L. R. Weatherford. Comparing decision rules that incorporate customer diversion in perishable asset revenue management situations. *Decision Sciences* 27:343–363, 1996.

[8] M. Ben-Akiva and S. Lerman. *Discrete-Choice Analysis: Theory and Application to Travel Demand*. MIT Press, Cambridge, MA, 1985.

[9] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, Belmont, MA, 1996.

[10] D. P. Bertsekas. *Nonlinear Programming*, 2nd ed. Athena Scientific, Belmont, MA, 1999.

[11] D. Besanko and W. L. Winston. Optimal price skimming by a monopolist facing rational consumers. *Management Science* 36:555–567, 1990.

[12] R. R. Bhat. *Managing the Demand for Fashion Items*. UMI Research Press, Ann Arbor, MI, 1985.

[13] A. V. Bhatia and S. C. Parekh. Optimal allocation of seats by fare. Presentation to AGIFORS Reservations Study Group, Trans World Airlines, 1973.

[14] G. R. Bitran and S. V. Mondschein. Periodic pricing of seasonal products in retailing. *Management Science* 43:61–79, 1997.

[15] G. R. Bitran, R. Caldentey, and S. V. Mondschein. Coordinating clearance markdown sales of seasonal products in retail chains. *Operations Research* 46:609–624, 1998.

[16] C. H. Bohutinsky. The sell-up potential of airline demand. Master's thesis, Flight Transportation Lab, MIT, Cambridge, MA, 1990.

[17] S. L. Brumelle and J. I. McGill. Airline seat allocation with multiple nested fare classes. *Operations Research* 41:127–137, 1993.

[18] S. Brumelle and D. Walczak. Dynamic airline revenue management with multiple semi-Markov demand. *Operations Research* 51:137–148, 2003.

[19] S. L. Brumelle, J. I. McGill, T. H. Oum, K. Sawaki, and M. W. Tretheway. Allocation of airline seat between stochastically dependent demands. *Transportation Science* 24:183–192, 1990.

[20] P. Coy. The power of smart pricing: Companies are fine-tuning their price strategies—and it's paying off. *Business Week* (April 10):160–164, 2000.

[21] R. G. Cross. *Revenue Management: Hardcore Tactics for Market Domination*. Broadway Books, New York, 1997.

[22] R. E. Curry. Optimal airline seat allocation with fare classes nested by origins and destinations. *Transportation Science* 24:193–204, 1990.

[23] G. Das Varma and N. Vettas. Optimal dynamic pricing with inventories. *Economics Letters* 72:335–340, 2001.

[24] S. Daudel and G. Vialle. *Yield Management: Applications to Air Transport and Other Service Industries*. Les Presses de L'Institut du Transport Aerien, Paris, France, 1994.

[25] S. V. de Boer. Advances in airline revenue management and pricing. Ph.D. thesis, Sloan School of Management, MIT, Cambridge, MA, 2003.

[26] J. Eliashberg and R. Steinberg. Marketing-production joint decision making. J. Eliashberg and J. D. Lilien, eds. *Management Science in Marketing, Handbooks in Operations Research and Management Science*. North Holland, Amsterdam, The Netherlands, 1991.

[27] W. J. Elmaghraby and P. Keskinocak. Dynamic pricing in the presence of inventory considerations: Research overview, current practices, and future directions. *Management Science* 49:1287–1309, 2003.

[28] Y. Feng and G. Gallego. Optimal starting times for end-of-season sales and optimal stopping times for promotional fares. *Management Science* 41:1371–1391, 1995.

[29] Y. Feng and G. Gallego. Perishable asset revenue management with Markovian time dependent demand intensities. *Management Science* 46:941–956, 2000.

[30] Y. Feng and B. Xiao. Maximizing revenue of perishable assets with a risk factor. *Operations Research* 47:337–341, 1999.

[31] Y. Feng and B. Xiao. A continuous-time yield management model with multiple prices and reversible price changes. *Management Science* 46:644–657, 2000.

[32] Y. Feng and B. Xiao. Optimal policies of yield management with multiple predetermined prices. *Operations Research* 48:332–343, 2000.

[33] S. C. Friend and P. H. Walker. Welcome to the new world of merchandising. *Harvard Business Review* 79 (November) 2001.

[34] G. Gallego and G. J. van Ryzin. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Science* 40:999–1020, 1994.

[35] G. Gallego and G. J. van Ryzin. A multi-product dynamic pricing problem and its applications to network yield management. *Operations Research* 45:24–41, 1997.

[36] Y. Gerchak and M. Parlar. A single period inventory problem with partially controlled demand. *Computers and Operations Research* 14:1–9, 1987.

[37] Y. Gerchak, M. Parlar, and T. K. M. Ye. Optimal rationing policies and production quantities for products with several demand classes. *Canadian Journal of Administration Science* 2:161–176, 1985.

[38] G. Girard. Revenue management: The price can't be right if the tools aren't. Technical report, AMR Research Inc., Boston, MA (September) 2000.

[39] A. Y. Ha. Stock-rationing policy for a make-to-stock production system with two priority classes and backordering. *Naval Research Logistics* 44:457–472, 1997.

[40] A. Heching, G. Gallego, and G. J. van Ryzin. Markdown pricing: An empirical analysis of policies and revenue potential at an apparel retailer. *Journal of Pricing and Revenue Management* 1:139–160, 2002.

[41] A. L. Hempenius. *Monopoly with Random Demand*. Rotterdam University Press, Rotterdam, The Netherlands, 1970.

[42] A. Ingold, U. McMahon-Beattie, and I. Yeoman, eds. *Yield Management: Strategies for the Service Sector*, 2nd ed. Continuum, London, UK, 2000.

[43] S. S. Iyengar and M. Lepper. When choice is demotivating: Can one desire too much of a good thing? *Journal of Personality and Social Psychology* 76:995–1006, 2000.

[44] D. Jenkins, ed. *Handbook of Airline Economics*. McGraw-Hill, New York, 1995.

[45] C. A. Johnson. Retail revenue optimization: Timely and rewarding. Technical report, Forrester Research Inc., Cambridge, MA (July 23) 2001.

[46] C. Kalish. Monopolist pricing with dynamic demand and production costs. *Marketing Science* 2:135–159, 1983.

[47] A. Kaplan. Stock rationing. *Management Science* 15:260–267, 1969.

[48] S. Karlin and C. R. Carr. Prices and optimal inventory policies. K. J. Arrow, S. Karlin, and H. Scarf, eds. *Studies in Applied Probability and Management Science*. Stanford University Press, California, CA, 1962.

[49] S. E. Kimes. Yield management: A tool for capacity-constrained service firms. *Journal of Operations Management* 8:348–363, 1989.

[50] W. M. Kincaid and D. Darling. An inventory pricing problem. *Journal of Mathematical Analysis and Applications* 7:183–208, 1963.

[51] C. J. Lautenbacher and S. J. Stidham. The underlying Markov decision process in the single-leg airline yield management problem. *Transportation Science* 34:136–146, 1999.

[52] E. P Lazear. Retail pricing and clearance sales. *American Economic Review* 76:14–32, 1986.

[53] T. C. Lee and M. Hersh. A model for dynamic airline seat inventory control with multiple seat bookings. *Transportation Science* 27:252–265, 1993.

[54] Y. Liang. Solution to the continuous time dynamic yield management model. *Transportation Science* 33:117–123, 1999.

[55] K. Littlewood. Forecasting and control of passenger bookings. *Proceedings of the 12th Annual AGIFORS Symposium*, Nathanya, Israel, 1972.

[56] C. Maglaras and J. Meissner. Dynamic pricing strategies for multi-product revenue management problems. Decision, Risk and Operations Research Division Working Paper DRO-2003-10, 2003.

[57] J. I. McGill. Optimization and estimation problems in airline yield management. Ph.D. thesis, Faculty of Commerce and Business Administration, University of British Columbia, Vancouver, Canada, 1989.

[58] J. I. McGill and G. J. van Ryzin. Revenue management: Research overview and prospects. *Transportation Science* 33:233–256, 1999.

[59] A. Merrick. Priced to move: Retailers attempt to get a leg up on markdowns with new software. *Wall Street Journal* (April 7), 2001.

[60] E. S. Mills. Uncertainty and price theory. *Quarterly Journal of Economics* 73:117–130, 1959.

[61] N. T. Nagle and R. K. Holden (contributor). *The Strategy and Tactics of Pricing: A Guide to Profitable Decision Making*, 2nd ed. Prentice-Hall, Englewood Cliffs, NJ, 1994.

[62] I. C. Paschalidis and J. N. Tsitiklis. Congestion-dependent pricing of network services. *IEEE/ACM Transactions on Networking* 8:171–184, 2000.

[63] P. P. B. Pashigan. Demand uncertainty and sales. *American Economic Review* 78:936–953, 1988.

[64] P. P. B. Pashigan and B. Bowen. Why are products sold on sale? Explanations of pricing regularities. *Quarterly Journal of Economics* 106:1015–1038, 1991.

[65] N. C. Petruzzi and M. Dada. Pricing and the newsvendor problem: A review with extensions. *Operations Research* 47:183–194, 1999.

[66] S. Polt. Back to the roots: New results on leg optimization. *1999 AGIFORS Reservations and Yield Management Study Group Symposium*, London, UK, 1999.

[67] L. W. Robinson. Optimal and approximate control policies for airline booking with sequential nonmonotonic fare classes. *Operations Research* 43:252–263, 1995.

[68] S. M. Shugan. Editorial: Marketing science, models, monopoly models, and why we need them. *Marketing Science* 21:223–228, 2002.

[69] B. C. Smith, J. F. Leimkuhler, and R. M. Darrow. Yield management at American Airlines. *Interfaces* 22:8–31, 1992.

[70] S. A. Smith and D. D. Achabal. Clearance pricing and inventory policies for retail chains. *Management Science* 44:285–300, 1998.

[71] W. Stadje. A full information pricing problem for the sale of several identical commodities. *Zeitschrift für Operations Research* 34:161–181, 1990.

[72] N. Stokey. Intertemporal price descrimination. *Quarterly Journal of Economics* 94:355–371, 1979.

[73] N. Stokey. Rational expectations and durable goods pricing. *Bell Journal of Economics* 12:112–128, 1981.

[74] K. T. Talluri and G. J. van Ryzin. Revenue management under a general discrete choice model of consumer behavior. *Management Science* 50:15–33, 2004.

[75] K. T. Talluri and G. J. van Ryzin. *The Theory and Practice of Revenue Management*. Springer Science + Business Media, Berlin, Germany, 2004.

[76] G. T. Thowsen. A dynamic, nonstationary inventory problem for a price/quantity setting firm. *Naval Research Logistics* 22:461–476, 1975.

[77] B. Titze and R. Griesshaber. Realistic passenger booking behaviors and the simple low-fare/high-fare seat allotment model. *Proceedings of the 23rd Annual AGIFORS Symposium*, 1983.

[78] D. M. Topkis. Optimal ordering and rationing policies in a nonstationary dynamic inventory model with $n$ demand classes. *Management Science* 15:160–176, 1968.

[79] G. J. van Ryzin. The brave new world of pricing. Survey: Mastering management. *Financial Times* (October 16) 2000.

[80] E. J. Warner and R. B. Barsky. The timing and magnitude of retail store markdowns: Evidence from weekends and holidays. *Quarterly Journal of Economics* 110:321–352, 1995.

[81] L. R. Weatherford and S. E. Bodily. A taxonomy and research overview of perishable-asset revenue management: Yield management, overbooking, and pricing. *Operations Research* 40:831–844, 1992.

[82] L. R. Weatherford, S. E. Bodily, and P. E. Pfeifer. Modeling the customer arrival process and comparing decision rules in perishable asset revenue management situations. *Transportation Science* 27:239–251, 1993.

[83] T. M. Whitin. Inventory control and price theory. *Management Science* 2:61–68, 1955.

[84] R. D. Wollmer. An airline seat management model for a single leg route when lower fare classes book first. *Operations Research* 40:26–37, 1992.

[85] P. S. You. Dynamic pricing in airline seat management for flights with multiple legs. *Transportation Science* 34:192–206, 1999.

[86] Zabel, E. Muti-Period Monopoly Under Uncertainty. *Journal of Economic Theory* **5** 524–546, 1972.

[87] W. Zhao and Y.-S. Zheng. Optimal dynamic pricing for perishable assets with nonhomogeneous demand. *Management Science* 46:375–388, 2000.

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS | ISBN 1-877640-21-2
DOI 10.1287/educ.1053.0015

# Decision Analysis = Decision Engineering

***James E. Matheson***
Department of Management Science and Engineering, Stanford University,
Stanford, California 94305-4026, jmatheson@smartorg.com

**Abstract** Decision analysis is an engineering approach to helping decision makers reach rational decisions. It incorporates systems engineering and decision theory, enhanced by the ability of modern computation to build value-based models and carry out computations needed to deal with complexity in the number of factors, uncertainties, and dynamics. It also includes the processes for reaching good decisions with real people (and real organizations) and gaining their commitment to carry them out. Decision analysis is not a single approach, but a discipline with underlying principles and procedures that are adapted to diverse situations. Decision analysis uses tools like decision theory, influence diagrams, system dynamics, game theory, etc. to "engineer" good decisions in new product development, business strategy, space system safety, etc. The paper ends with the "10 commandments of decision analysis." You can guess the first.

**Keywords** decision analysis; decision engineering; tutorial

## Decision Engineering

Suppose we were to ask an experienced and well-qualified engineer to specify the design of a bridge over a river crossing. Would we expect the bridge expert to contemplate for a while and then tell us the specification for a sound bridge? Of course not! The expert would examine the conditions where the bridge is to be built, the loads it must carry, the funds available for construction, etc. He would probably do a preliminary design of several feasible bridge types, calculate stresses and strains on the structural elements of the bridge, and determine which design would allow the construction of the best bridge for this purpose. He would examine the effects of climate, such as flooding and winds. He would use many different skills in assessing the environment of the bridge and the needs of its users in developing possible bridges and in coming to a choice of the best one. He would be likely to consult experts in many specialties to refine his understanding of the needs and desires of the users as he goes through this engineering process of coming to a recommendation. He will need to explain to laymen why he made or recommends certain choices, and he may need to obtain the sponsors' trade-offs on matters such as cost versus aesthetics.

In contrast, executives are often asked to make critical choices based on gut feelings. Even an experienced engineer would not attempt to build a sound bridge this way, and the executive, even one with a great education and lots of experience, should not be expected to arrive at sound decisions this way either. A new top executive of an organization told his subordinates not to waste time on decision making, but rather to bring their hard decisions to him because he was a decision maker, and he would call the shots for them. Wall Street often plays into this concept by expecting changes in top management to instantly turn around poorly performing organizations. Like the bridge design, a good decision needs to be properly engineered; it is naive to guess the answers directly.

The term "decision analysis" was defined by Ron Howard in a 1965 presentation of a paper, "Decision Analysis: Applied Decision Theory" (Howard [7]). Here he defined decision analysis as "a formal procedure for the analysis of decisions." By procedure he did not mean

decision theory. Ron Howard defined his original procedure as three phases: a deterministic one, where alternatives are defined and an initial formal model is developed; a probabilistic phase, where probabilities are assessed on the crucial uncertainties, risk attitude is encoded (if necessary), and tentative recommendations are made; and a post mortem phase (later called an informational or appraisal phase) to determine what iterative steps should be taken before acting on a recommendation. This decision analysis procedure is the outline of an engineering process for designing a good decision, analogous to the bridge designer's process for specifying a good bridge. In the paper, he did apply elements of decision theory, particularly the philosophy that probability captures an individual's state of mind about an uncertainty, but his use of decision theory is to fit it into a framework of an engineering approach to decision making.

As a historical note, in the mid-1960s Ron Howard and I were walking with Howard Raiffa on the Harvard campus debating what would be the best name for this profession. Howard Raiffa seemed opposed to the term decision analysis, as it seemed too analytical and academic. Ron Howard actually had similar reservations, but he discarded the term "decision engineering" because engineering has a second meaning of manipulation. A while after this debate, Ron and I were both astonished to discover that Howard Raiffa had changed the name of his new book to *Decision Analysis* (Raiffa [18]). Between the two of them, they cast this term in concrete. However, in the good sense of the word, the field is really about "decision engineering."

## The Discipline of Decision Analysis

Why the word "discipline"? I am using the term discipline in the sense of a branch of knowledge or area of teaching. Major universities will organize their engineering activities around disciplines such as mechanical engineering, electrical engineering, chemical engineering, etc. These disciplines gather together all of the specialized knowledge relevant to their discipline and show how knowledge from other specialties fits in as well. Their knowledge base increases over time, but the charter of the discipline remains relatively constant.

Returning to the bridge example, it would be silly to suggest that we design a bridge without using mechanical engineering, to ask, "what are alternate methods we might use to design bridges?," with the idea of testing which method works best. The problem is that mechanical engineering is not a technique, but the field concerned with all of the techniques and knowledge about bridge building. The question does not compare apples with oranges, it compare apples with the bowl holding the apples and oranges.

Similarly, we are often asked to compare "decision analysis" with some other technique. From a decision-engineering perspective, this is the same kind of silly question. One problem is that many people think decision analysis is the technique of decision trees. They might really be asking if can we compare decision trees with some other method such as influence diagrams, or even if we can compare normative recommendations with descriptive behavior. These are appropriate questions within the field of decision analysis. Furthermore, our own profession gets caught up in similar questions, such as comparing decision analysis with game theory, system dynamics, or multiattribute utility. From a decision-engineering perspective, all of these items are tools within the discipline, and it is absurd to compare them with the whole discipline.

When I use the term decision analysis in the remainder of this paper, I mean it in this sense of the field of decision engineering.

## When to Use Decision Analysis

Decision analysis is for when you do not "know" what to do. You may know what to do from long experience with a repetitive class of decisions. In many of these repetitive situations the human ability to recognize patterns, and in emergency situations to act on them quickly, is the heart of the decision problem. Gary Klein and others have developed a whole field

of "naturalistic decision making" by studying these sorts of situations (Klein [12]). The currently popular book *Blink* looks at similar situations where human pattern recognition dominates, but in the latter part of the book Gladwell shows many instances where these judgments go awry (Gladwell [4]). In an example I studied, the operators dealing with the Three Mile Island nuclear plant incident focused early on an incorrect accident scenario, then rejected all evidence to the contrary: It took a new operator who walked into the room hours later, without this prejudgment, to recognize the real situation and stop the accident. In high-stakes situations, a rapid decision analysis look at the situation might uncover mistakes. Bayes' rule might have led to the correct diagnosis. Another situation of "knowing" involves decisions close to basic value judgments, such as a decision of whom to marry, or whether to have a child. While I have seen decision analysis used in both of these situations, many people are guided by inner values and "know" what they want. Here introspection often trumps analysis.

Decision analysis is most powerful when focused on high-value decisions dominated by uncertainty, dynamics, and complexity. It is usually more important to uncover better alternatives, cut through the complexity, and assess the biggest uncertainties, than it is to finely optimize decision variables. While there is no clear boundary, decision analysis situations and optimization situations are dominated by different issues and emphasize the use of different tools. In organizational settings, there are many reasons to use decision analysis, even if you think you know what to do. The first is to double-check your intuition and possibly hone it further. Ask if this is a situation surrounded by "fog." Even the best-trained and experienced pilots cannot fly level in fog without instruments: They are likely to turn their plane upside down and crash. Second, you may need to explain the basis of your decision to others: superiors to justify your actions or recommendations, and subordinates to convey the understanding necessary to execute the appropriate actions and adapt to new information. Third, you may want to bring many participants into the decision process. Often there are experts available in many aspects of the situation. They provide information needed to model and assess various factors. When responsibility for the decision is diffuse, a group may use the explicit structure of decision analysis to work together to a single recommendation. Decision analysis is often used to get a wide range of participants to agree on the wisdom of the decision and to gain the knowledge and motivation to implement it well.

## Philosophy

Decisions are a way to gain a degree of control over a reality too complex to understand fully. The decision analysis approach explicitly treats only a bounded portion of this reality. However, it includes tools for understanding or testing if the bounds on the problem are inappropriately set. Limiting reality to the natural bounds (scope or frame) of a particular decision (or set of decisions) helps people to temporarily focus on issues relevant to this decision and motivates them to treat this decision carefully and to reach closure. This focus helps prevent mental and organizational attention from wandering from one thing to another without resolving anything.

### Declaring Decisions

For some problems, just formulating a clear decision is a big step. In educating youth, the Decision Education Foundation (DEF), a nonprofit focused primarily on teenagers, has found that getting individuals to see that they have a choice, other then simply reacting to situations, opens up a whole new way of controlling their future (Abbas et al. [1]). They have a hallway poster that says, "stop, think, and decide." Organizations too often do (or do not do) the same thing, carrying on reactively with the same old habits without recognizing that they have important and valuable decisions to make. Unless something breaks down, organizations are not faced with the need for decisions, yet decisions, such as getting the jump

on the competition, need to happen a long time before competitive breakdown. Declaring times when decisions need to be made (and focusing the appropriate effort on making them) is a critical responsibility of organizational leadership. Many decisions that will ultimately determine the fate of an organization fall in Covey's important, but not critical, quadrant (Covey [2]). Leadership needs to continually set and monitor the decision agenda for their organization to assure that key decisions are "declared" and deliberately made with sound processes.

## Bounding the Problem

The world is too complex to comprehend in one frame or model. Therefore, we bound our perspective on the world in different ways for different situations. An electrical engineer would not use quantum mechanics or Maxwell's equations to design a circuit, but he might use them to design a component or a communications system. We can only grasp a small portion of reality at a time, but if we get it right we are able to explain our universe better and better and to develop a whole host of new products and ways of organizing our businesses and our lives. Once we declare a decision we need to place appropriate bounds on the piece of reality we need to capture, and to determine what degree of verisimilitude we need in our logic and models to reach a good or satisfactory decision. We would not use the same degree of precision to order a dinner as we would to decide on a critical medical treatment.

If we picture the situation as a foggy mess, we need to put a circle around the piece of this mess we intend to capture in our decision-making process. Inside these bounds we will need to model the situation to whatever degree of approximation is required to reach the decision (or class of decisions) at hand. We follow Einstein's sage advice to use the simplest model that adequately treats the situation. Otherwise, we get into the game of trying to model the world, which is impossible even within our problem bounds. Simplicity yields more insight and motivation than complexity.

Bounds need to be tested and revised. When solving the problem gets technically difficult, for example by requiring detailed optimization or difficult assessments, I take that as a sign that the bounds need to be revisited. In strategic settings constraints are rarely fixed, even if people initially claim that they are inviolable. Pricing constraints often leads to simpler solutions, and adjusting prices gives insight into the value of letting go of the perceived constraint. The interfaces of the problem bounds with the real world are probability assessments on what might happen "out there," our perception and creation of a range of possible alternatives, and higher-level value assessments. For example, public corporations are presumably run to benefit the shareholder-owners. Therefore, we might be tempted to include shareholder preferences, the way shareholders change through market transactions, price formation in the stock market, and so forth. However, if we are designing a new product, we would typically leave out all of this detail and use maximizing the net present value of cash flow as a surrogate. In this case, we interface the bounded problem with the larger reality through assessment of a discount rate and a risk attitude. In another situation, we might be concerned about the price of energy-intensive raw materials and be tempted to model the energy industry, but it is probably wiser to directly assess the future cost of raw materials, and, if this is not sufficient for our purposes (reaching a specific decision), expand to directly assessing future energy price and how our raw material costs (and other important costs) depend on it. We adjust the problem boundaries to fit the problem structure and peoples' ability to assess value, uncertainties, and alternatives at the interface.

A great example of models going wrong is given by the well-documented saga of a Wall Street hedging firm called Long-Term Capital Management (Lowenstein [13]). The founders included Nobel Prize winners, who set out to make money from "riskless" hedging transactions by finding sophisticated arbitrage opportunities in the financial markets. Over time they came to believe their market models' predictions and to take more and more "naked" risk. In the end the market did not behave as their models predicted, and they came very

close to bringing down the whole U.S. and international financial systems. Economists like to attribute these problems to market imperfection, but they are really due to model imperfection combined with hubris. They came to believe that their model was reality and that departures from their model were defects of the world—not defects of their model. Physics learned and lamented this lesson when the theory of relativity displaced Newton's laws, but if you listen to their talk today—for example, about multiple worlds, they forget this is just an interpretation of a theory, not reality. The lesson learned over and over is that the only perfect model of reality is reality itself, and our model is only useful to describe a piece of reality for a specific purpose—in our case, to reach a specific decision or a small class of decisions. Bounding the problem correctly, and reexamining those bounds carefully, is critical (Matheson [14]).

## Admitting and Treating Uncertainty

Some people think that they should guess the future and then make decisions consistent with that guess. Corporate staff members have characterized one of their jobs as telling top management what the future will bring, so that management can make decisions assuming that future. For example, if the staff projects demand growth, management can easily figure out when to add capacity. However, if they were to admit uncertainty, management might have to work out ways of hedging their capacity additions, for example, using contract manufacturing for a period until they are sure demand will be sufficient to fill a new plant. The kinds of alternatives to consider under certainty are more limited than the kinds needed to deal with uncertainty. So, admitting uncertainty is essential to developing a good set of alternatives from which to choose. In addition, with uncertainty comes the pain of risk taking, and we need to consider how to trade off risk versus return.

Once decision makers admit they are facing uncertainty, their choice could have both good and bad outcomes. An excellent decision could have a bad outcome, while a poor decision might have a good one. This means that the decision maker cannot logically be responsible for the actual outcome, but only for making good decisions. Furthermore, because the decision is made in advance of its outcome, judging the quality of the decision is most useful at the time it is being made! Therefore, we need a definition of a good decision at the time it is made, to assure decision makers that they are making good decisions and to allow others to audit the quality of decisions that have been made.

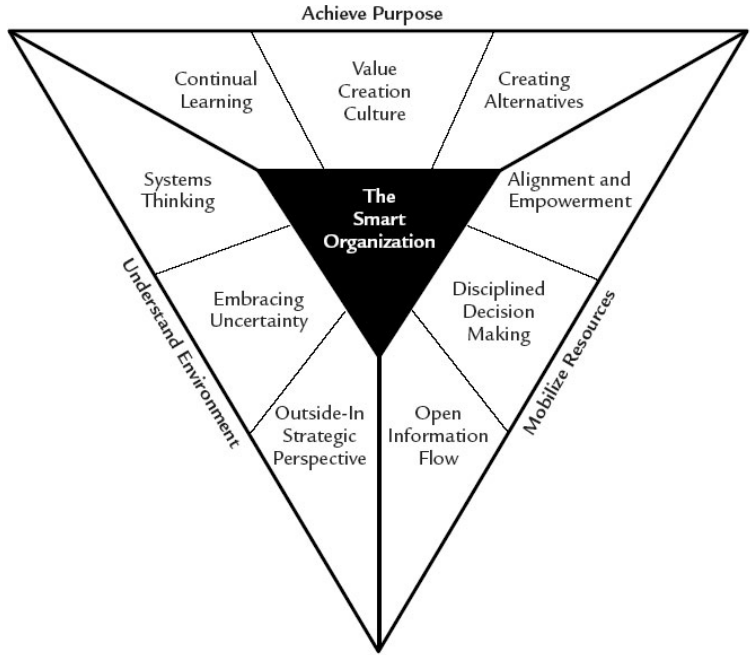
The whole decision analysis process is itself the definition of a good decision! One must ask: Has the process been applied well, were the right participants and experts brought into the process, were the right tools applied correctly, etc.? If not, the decision is suspect. The need to distinguish the quality of the decision from the quality of the outcome is the genesis of decision analysis.

## The Nine Principles of the Smart Organization

In 1998, David Matheson and I introduced "nine principles of the smart organization," illustrated here in Figure 1 (Matheson and Matheson [15]). Each principle represents a philosophy and behaviors that contribute to the support of high-quality decision making. Together they create a culture that is receptive to the decision analysis approach and that motivates its use. They are organized in triads around:

- achieving purpose—continual learning about what is valuable and finding alternatives to create and capture more value;
- understanding the environment—building an outside-in strategic perspective and capturing all of our information and insights with uncertain and dynamic thinking and formal models;
- mobilizing resources—developing disciplined decision-making processes that are mobilized through open information flow and that align and empower all participants in the process.

FIGURE 1. Nine principles of the smart organization.




Space does not allow for an explanation of these powerful principles and their ramifications, so please see the book for further detail.

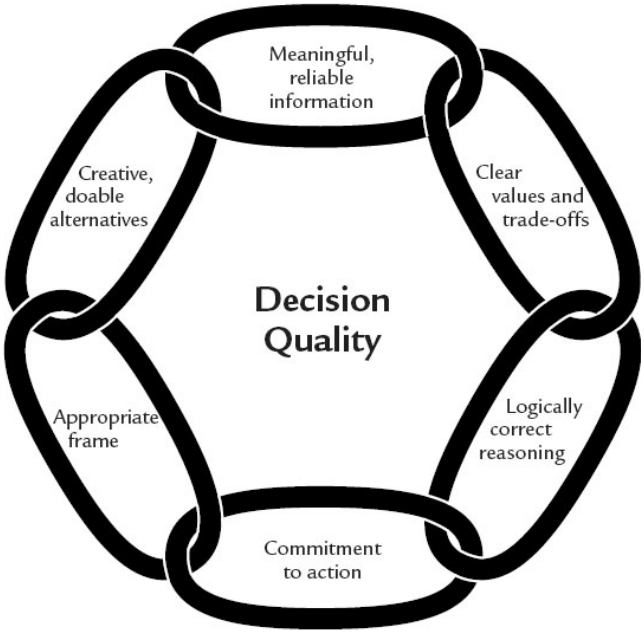
When we wrote the book, we devised an "organizational IQ" test that measures an organization's conformance to these nine principles. However, we could not prove that these principles created more value for the organizations that embraced them. After publication we administered thousands of organizational IQ tests, which to our pleasant surprise showed that organizations in the top quartile of organizational IQ approach a 50% chance of also being in the top quartile of performance, while organizations in the bottom quartile of organizational IQ have only a 10% chance of being in the top quartile of performance (Matheson and Matheson [16]). *The "high-IQ" organization has about five times the chance of being a top performer than its "low-IQ" counterpart.*

## Decision Quality

So, what is a high-quality decision anyway? The ideal time to determine the quality of a decision is when you make it. A retrospective, after the results are in, may shed light on something that should not have been overlooked in the decision-making process, but usually they are used to "punish the innocent" for a bad outcome. Of course, the only time you can correct a poor-quality decision is before you commit the resources. Over the years we have evolved six elements of decision quality and arranged them into a chain as in Figure 2 (Spetzler and Keelin [20]). To determine which link is most important, think of hanging over the edge of a precipice by this chain—the most important link is the weakest one. On the other hand, links that are overbuilt have wasted resources that might have gone into reinforcing the weaker links, but it is too late now.

This leads us to a spider diagram version, as in Figure 3. When reviewing the status of a decision-making process, we should review each element for sufficiency to support a good decision. A common mistake is for decision makers to spend excessive effort gathering information without taking stock of the appropriate frame or considering the range of alternatives they might pursue—information becomes a security blanket to the detriment of a balanced decision analysis.

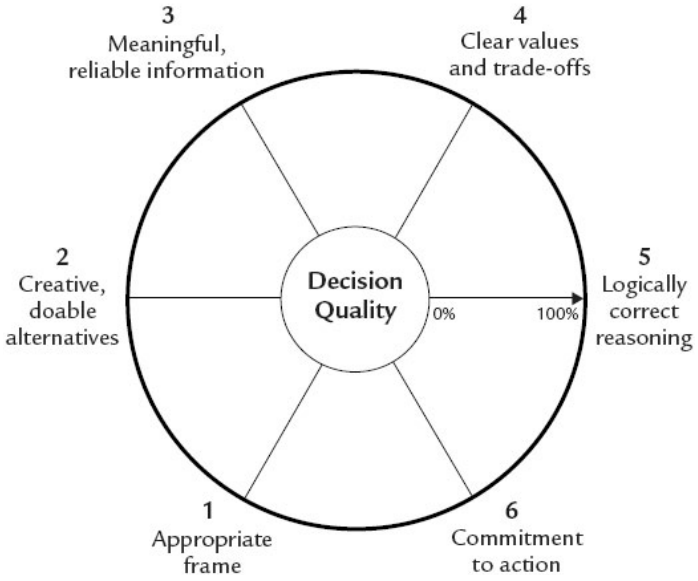
FIGURE 2. The decision quality chain.




The first link is framing, or bounding the problem, which we have discussed above. More problems fail here than elsewhere. The next three links, called the "decision basis" (alternatives, information and values), are the building blocks of the model of the situation and the assessments that support it. The fifth link is sound reasoning to derive the logical conclusions and insights indicating a correct course of action. Lastly, there should be commitment to action. We have all experienced personal "decisions" to diet, quit smoking, etc., which fail through the lack of will to carry them out.

This brings us to a philosophical question of whether a decision is a commitment to action or the action itself. For example, we buy a house or a car by signing legal papers. Here it

FIGURE 3. The decision quality spider diagram.



seems sensible to call that act the decision. However, in some states you have three days to change your mind on such decisions, so is the real decision not revoking the original commitment over the three-day period? In organizational settings the link between the commitment and the action becomes more tenuous. For example, an executive may decide to build a new factory. This decision results in many more decisions to pick the best site, hire contractors, buy equipment, hire and train employees to run the factory, etc. It is impossible to identify any one irrevocable commitment to build the factory. In fact, the executive decision makers almost never take the irrevocable actions that implement their decisions—they never write the checks themselves.

So, in an organizational setting it is often useful to separate the decision from the actions that implement that decision. A decision maker commits his or her organization to carry out the required actions. This distinction creates the possibility of implementation failure for many possible reasons: lack of will, resources becoming unavailable, new economic conditions, subordinates not following through, etc. A good organizational decision-making process includes mechanisms for increasing commitment to action—increasing the probability that the consequent actions will actually be carried out.

## Processes and Tools

A process often used in reaching high-level business decisions is called the dialogue decision process (DDP), which was developed by the Strategic Decisions Group (SDG) and evolved through many client engagements. It is targeted at large-scale organizationally intensive decisions involving both cross-functional teams and several layers of management. It is designed to bring the right people into the process to obtain the necessary knowledge and creativity to understand the decision situation. The process is also designed to motivate and commit these people to carry out the committed decision.

Figure 4 shows a decision process "snaking" between a decision team, the people who bring corporate perspective and the power to execute the decisions, and a strategy team, the people who bring content information and who are likely to participate in implementation—carrying out the "commitment to action." I have avoided calling the decision team the executive team because executives and managers may serve on both teams, and on different teams for different decisions. Sometimes a few decision team members even change hats and serve as content experts for the strategy team.

Figure 4. The dialogue decision process: Deciding.

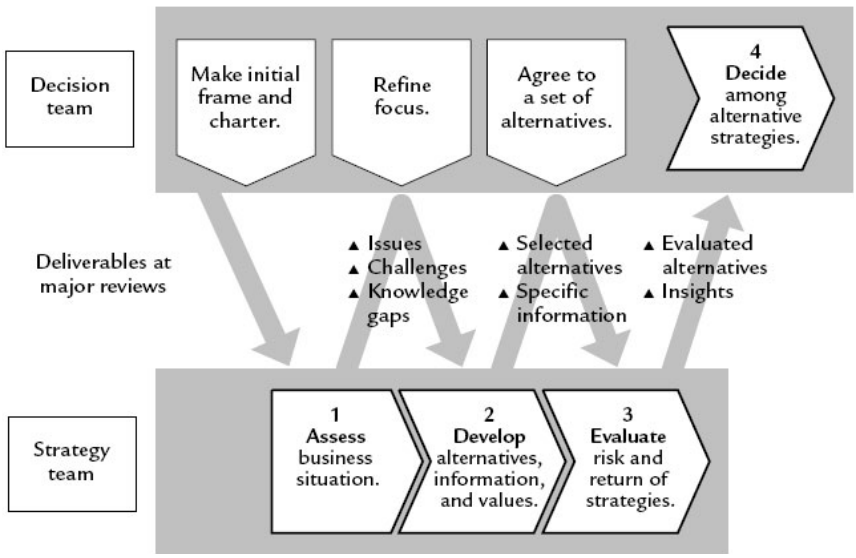
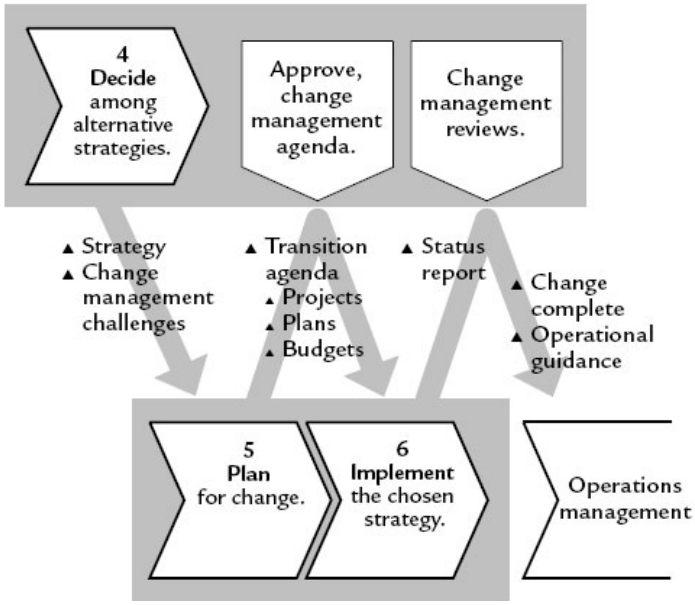


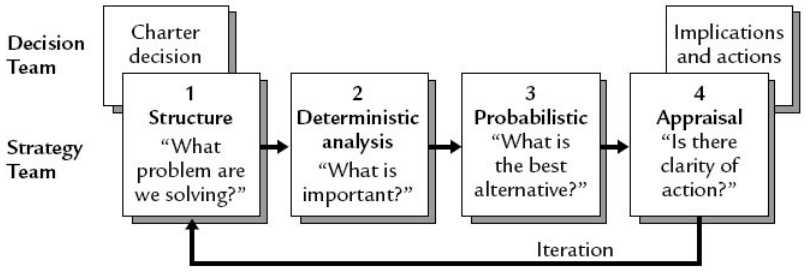
FIGURE 5. The dialogue decision process: Implementing.




In this design, the decision team members participate in the decision-making process from the beginning, and are not simply judges who review the output of a staff analysis. They are told that if they withhold issues or information during the process, they are out of order bringing them up at the end. On a decision team, they cannot use the common ploy of absenting themselves from the process while sitting in judgment at the conclusion. These two groups carry out an orderly decision process, which goes through four major steps before reaching the decision—commitment to action—at the end of Step 4. To insure the committed action is actually carried out, a second phase with two additional steps is devoted to planning and implementing, shown in Figure 5.

Another process is the classical decision analysis cycle of Figure 6, which is used in this form for problems that are less organizationally intensive. This process emphasizes bounding and framing the problem in Step 1, understanding the important factors and relationships in the deterministic analysis of Step 2, formally dealing with uncertainty and risk attitude to reach recommendations in Step 3, and appraising whether to act now or refine aspects of the analysis further in Step 4. In a major DDP, a complete first pass through the decision analysis cycle might be included in the DDP assessment phase and a second pass included in the next two DDP phases leading up to recommendations. These two processes are different ways of adapting to various organizational settings as well as different ways to look at the decision analyst's job.

FIGURE 6. The four-phase decision analysis cycle.



I will now review a few of the key tools in the framework of the decision analysis cycle. My goal is to show how they fit into the decision-engineering process, rather than to explain the tools in depth. Good overviews and case studies are available in (Matheson and Matheson [15]) and the tools and methods are detailed in *Decision Analysis for the Professional* (McNamee and Celona [17]) as well as in extensive professional literature.

## The Decision Hierarchy

A failure mode is to try to solve too many levels of a decision problem simultaneously. For example, many corporate executives are highly skilled at dealing with operation problems, where detailed planning is critical. However, detailed planning sabotages strategy because it severely limits the number of strategies that can be explored. Many so-called corporate strategies are nothing more than multiyear plans, usually with no consideration of uncertainty. Detailed planning is carried out in the second phase of a DDP (look back at Figure 5) after the major strategic direction has been set. However, for setting the strategic decision we need visions of potential landscapes, not detail.

The decision hierarchy is a great tool for setting and communicating the frame of a decision. Figure 7 illustrates the decision hierarchy for a manufacturing plant modernization problem. We take as given the statement called policy decision at the top of the hierarchy, here the policy that we will continue to manufacture—we will not be questioning that decision. However, if all strategies turn out to be poor, we might return to question that policy and revise our frame. In addition, from the bottom of the diagram we see that we will not be concerned with the design of products to be manufactured at this facility, detailed manufacturing operations or making marketing plans, as these are all tactical (or implementation) decisions. All of these items will be optimized after the strategic decisions are reached. However, we may need to put some placeholder tactical decisions in place to evaluate the strategies. We are asserting that the placeholder tactics are sufficient for selecting the best strategies, an assumption we can and should test with sensitivity analysis. The concerns of this decision analysis are the items listed in the center—the strategic decisions—the basic configuration and capabilities of the plant along with high-level strategy for marketing the products it produces.

The decision hierarchy specifies the strategic decision areas that are to be detailed by one or more columns in a strategy table, as illustrated in Figure 8. In Figure 9 we develop four potential strategies by the selection of one option from each column. A real application typically would have 10 to 30 columns, possibly specified in a hierarchy of strategy tables. It would not make logical sense to try to enumerate and model all of the strategies that could

FIGURE 7. A decision hierarchy.

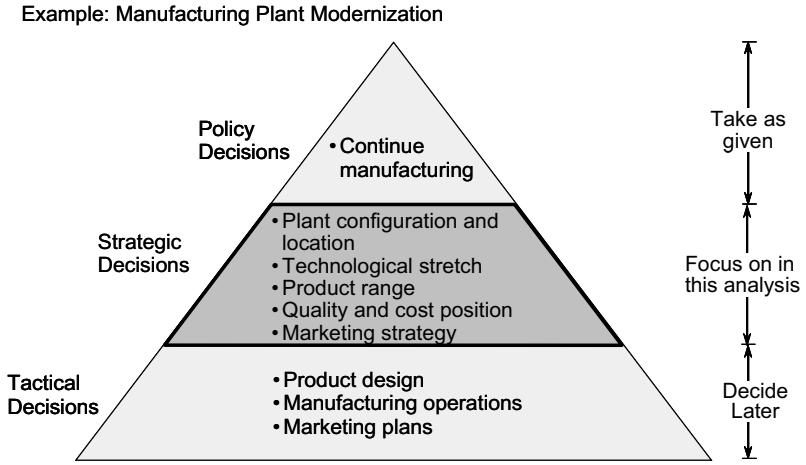


FIGURE 8. The decision hierarchy determines decision columns in a strategy table.

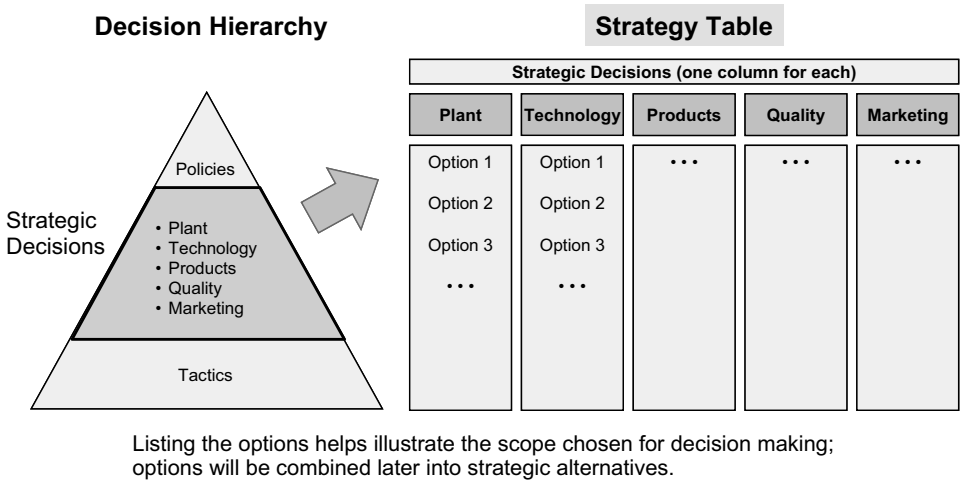
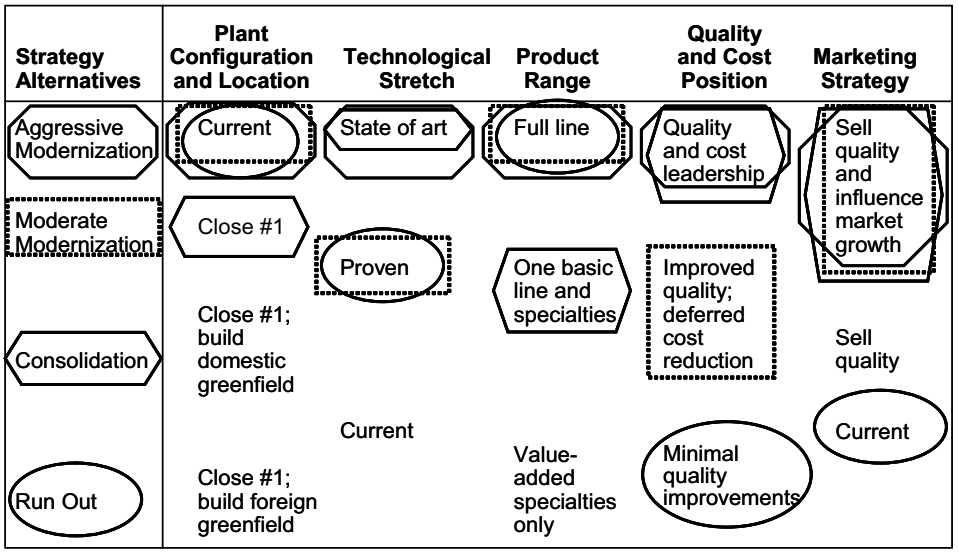



FIGURE 9. Example strategy table.



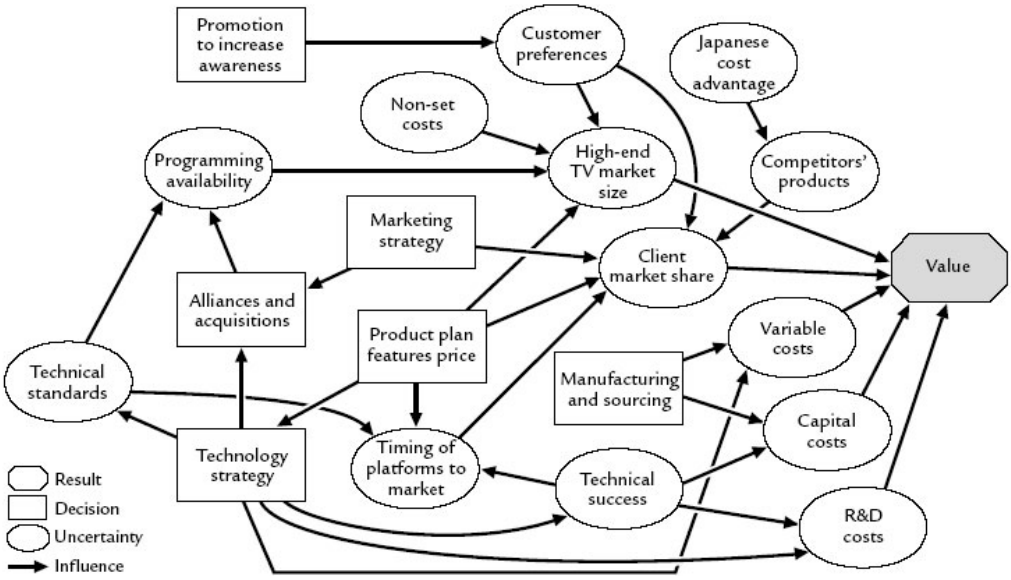

be specified in such a table. Rather, we ask the participants to develop about three to six significantly different alternatives that make up coherent sets of selections to evaluate and test—we can add others based on the insights provided by the initial set.

It is critical that each option in the strategy table represents an actionable, although high-level, specification of direction that the participants could plan in detail if it were selected—so we often ask participants to write an essay of a few paragraphs describing each one. The decision model will contain cost and other features of each possible choice.

## The Influence Diagram

The influence diagram furnishes a transition from more qualitative framing to quantitative modeling. Figure 10 is an example influence diagram. On the surface it looks like just another block diagram, but it actually can be used to precisely specify an entire decision model and all of its assessments (Howard and Matheson [9]). Each rectangle represents a decision, which is informed by arrows from any nodes preceding it. You can think of the alternatives as inside the rectangle, to be viewed by opening the box (double-clicking). Each

FIGURE 10. Influence diagram.




oval represents an uncertainty, containing a set of probability distributions conditioned by the graphical nodes immediately preceding it. A special eight-sided node represents a value model. Sometimes deterministic nodes are added, usually indicated by a double oval as a degenerate form of an uncertainty, which allows easy specification of model relationships and even time series. Influence diagrams are a breakthrough representation for capturing uncertain decision situations. On the surface they are a simple picture of the elements of the situation, but they enable automatic model generation, specification of decision trees, and new algorithms for directly solving influence diagrams themselves. They have become a ubiquitous tool for decision analysis, artificial intelligence, and other fields as well.

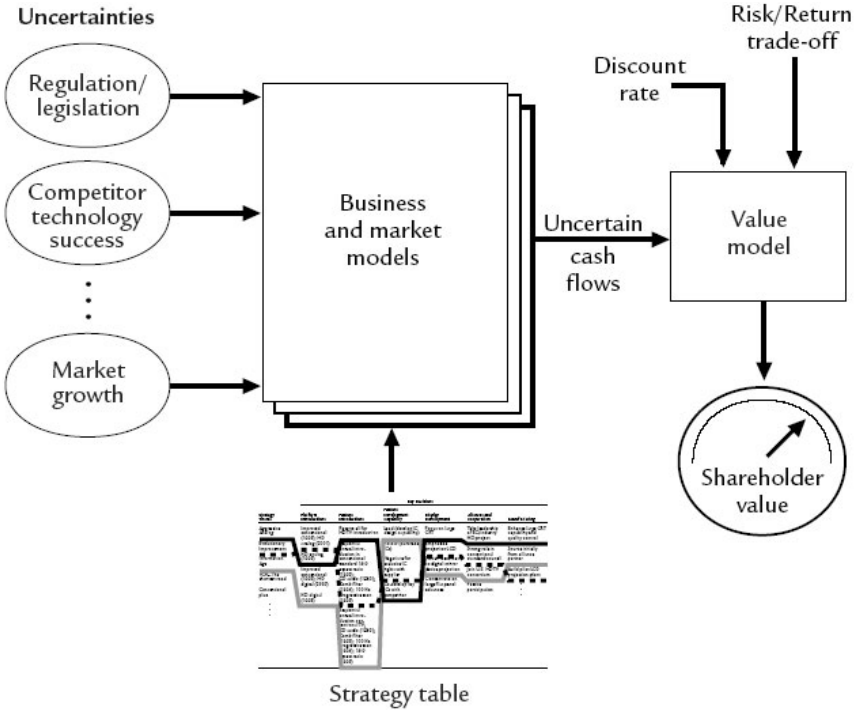
## The Decision Model

Usually after relationships are specified and understood in an influence diagram, an economic model is either automatically generated or manually constructed to calculate first deterministic, then probabilistic results. An example strategic business model is illustrated in Figure 11, which we sometimes refer to as a "boiler diagram." The center is a deterministic model of the business, markets, competitors, dynamics, etc.—"the boiler"—often implemented as an Excel spreadsheet. Feeding into it from the bottom are all of the decisions we might make, here shown as paths through a strategy table. Feeding into it from the left are all of the uncertain variables, such as those specified by the ovals in the influence diagram. Out of the boiler come results, in business usually in the form of uncertain cash flows, that get discounted to the present, adjusted for risk, and measured by the shareholder value meter. In principle, we just select the decision that maximizes the reading on this meter. However, in practice, we proceed in steps, learning and refining the model and assessments as we go along.

## The Tornado Diagram

In the deterministic phase, we assess the continuous uncertainties by estimating the range of probability, most often expressed as the 10-50-90 percentiles. Early in decision analysis practice we would quickly assess these ranges and return to more carefully assess them in the probabilistic phase. Today, we recommend that particular care be taken to remove biases from these range assessments by the same procedures that have been established

FIGURE 11. Key elements of a strategic business model.

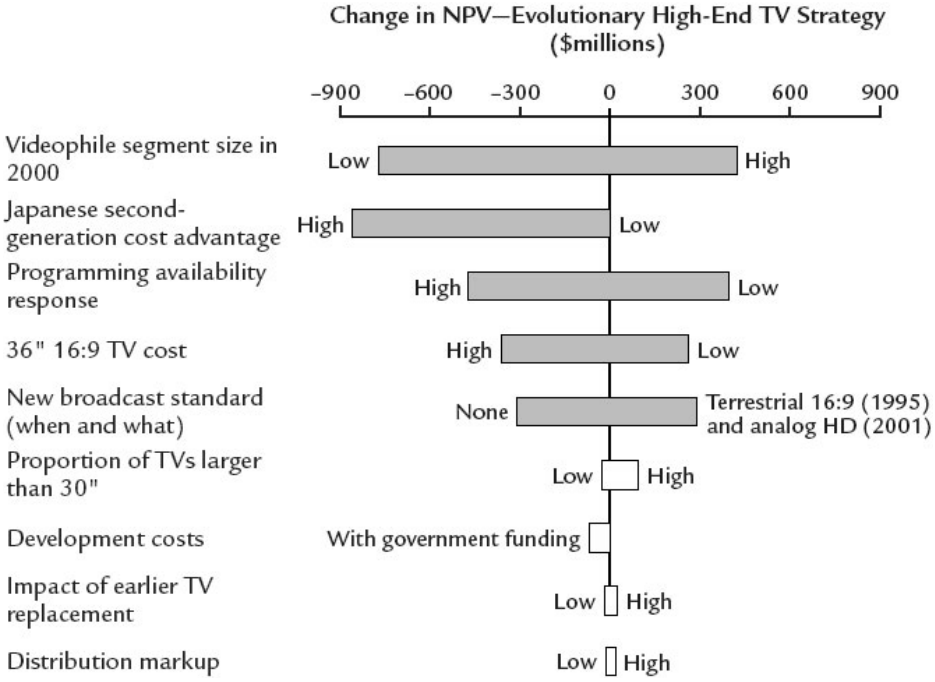



for complete probability assessment. There are two reasons for this advice: First, we have learned that biases can be so strong that even deterministic sensitivity can be untrustworthy; and second, in rapid analysis these ranges are often used to create the initial probabilistic assessment—for example, by assigning them discrete probabilities of 0.25, .05, and 0.25, so it is important that they be trustworthy assessments. For more information on biases, see the original *Science* article of Kahneman and Tversky [5], the probability assessment procedure developed with their help (Spetzler and Staël van Holstein [21]), and the more popular book *Decision Traps* (Russo and Shoemaker [19]).

Once the uncertainties have been assessed in consistent 10-50-90 ranges, the deterministic model is used to vary each variable over its range while holding the other variables at their medians, and record the value measure, typically the NPV of cash flow. If these sensitivities are monotonic, the high and low values occur at the end of each range. For each variable we record the difference between the high and low values (the swing) and order the variables in descending order of these differences. If we plot these results as a bar graph, we end up with a characteristic tornado shape, as shown in the example tornado diagram of Figure 12. This figure and the next are taken from Chapter 9 of (Matheson and Matheson [15]), which provides additional discussion of this case.

Looking at the tornado diagram, we see that the first few variables contribute most of the uncertainty. In a simple additive independent model the width of the overall uncertainty bar would be the square root of the sum of the squares of these individual bars, amplifying the dominating contribution of the first few bars. Usually we find the decision-making team has been neglecting one of the first few variables and wasting a lot of time on the last few, in this case the second-to-last bar. While we intend to proceed to a probabilistic phase, we can make important observations by studying the tornado. For example, if we can easily get information about the first few bars, and if that information would change our decisions, we should probably go get it—it has high value of information. If we have a way of favorably controlling an uncertainty, say by purchasing a technology or even a small company, then

FIGURE 12. Sensitivity analysis: Tornado diagram.

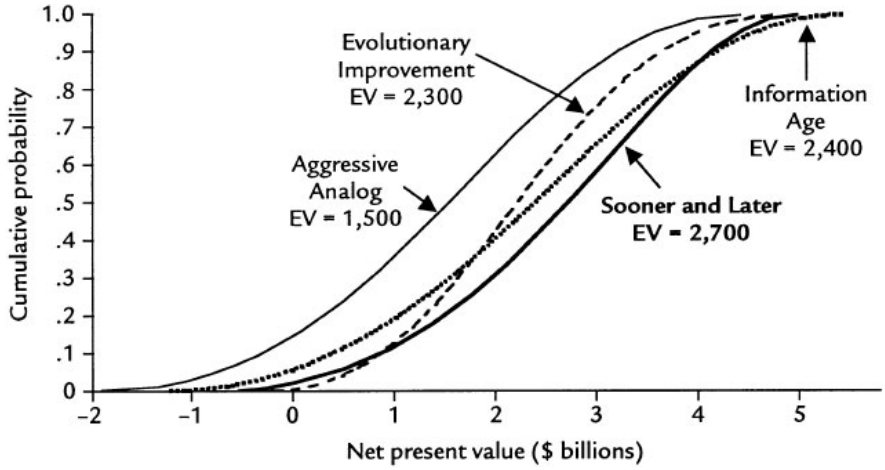



we should seriously consider doing it—it has high value of control. If there is no way to get information about one of these top variables, then quit worrying about it—it is existential uncertainty you have to live with, just like your own time of death.

Proceeding to the probabilistic phase, we carry out a complete probabilistic analysis, which will automatically include more combinations of these variables than in the individual sensitivities of the tornado diagram. If we have been quick and dirty in making the range assessments or if we see ways to get better information on the top variables, for example, by interviewing an easily available expert, we should carry out that reassessment. However, if we have been careful assessing and debiasing the 10-50-90 ranges, we may wish to convert these ranges into probability distributions for a first cut at probabilistic analysis.

A set of final probability distributions for the HDTV example is shown in Figure 13. The original three strategies were aggressive analog, evolutionary improvement, and infor-

FIGURE 13. Evaluation of several strategies



mation age, which presented a risk-return dilemma. However, with the insights of the first analysis a new hybrid strategy, sooner and later, was invented, which captured the low end of evolutionary improvement and the high end of information age. This alternative almost stochastically dominates the other alternatives and was accepted as the new corporate strategy, without the need to deal further with risk attitude. It is surprising how often this is the case in business problems with big, discrete alternatives, but of course in problems where you are deciding how much to invest in a venture, or how to bid, risk attitude usually is a critical input.

### Value-Based Modeling

Coming from the business decision-making context, we have developed an approach called value-based decision making. The first principle is to make decisions based on monetary value today. If you are spending money today, then you need to explicitly or implicitly justify this expenditure as being worth this value. Implicit judgments, or vague assessments like point-scoring systems, easily lead you astray and into the realm of social/political pressures and fuzzy thinking. Measuring monetary value provides a depersonalized standard of objectivity. A Dilbert cartoon character asked, "Why don't we just figure out what creates the most value and do that?" This is what we advise!

How do we get to this measure? If we first think deterministically, we often model scenarios that produce a cash flow over time (before or after taxes, free cash flow, etc.). We ask the decision maker, "If you had certain title to this cash flow, what would you trade for in cash today?" We call his answer the present equivalent of the cash flow stream. The answer does not have to be a linear function of the cash flows or stable over time, and I have seen cases in private companies where it is not. However, in most public corporations you will get the answer, "Just discount the cash flow by the corporate cost of capital (risk free) and call that the present equivalent." This is how we end up using net present value (NPV) as the present equivalent. Also, in the presence of a linear bank it can be shown that this procedure creates the most wealth—it creates the biggest pie.

Now what about uncertainty? First we reduce time streams to present equivalents (or NPVs), so our decision maker is now facing lotteries on present money, an easier situation to contemplate. We then ask the decision maker, "If you faced a particular lottery, how much would you trade it for in a certain payment?" We call this amount the certain equivalent. (A historical note: The reason we avoid the East Coast "certainty equivalent" is that presently equivalent does not make grammatical sense, so we chose parallel terms.) Like the linear discounting case, there are many good reasons to capture risk attitude by an exponential utility function (constant risk attitude), at least approximately. In a dynamic decision situation and using these approximations, we would roll back a decision tree by discounting over time to present equivalents, reducing lotteries to certain equivalents, discounting again, and repeating this process until we work back to the present. For consistency with utility theory, one must increase future risk tolerances by the compounding rate used in the NPV.

What about nonmonetary considerations? First, usually nonmonetary results must be paid for, whether it is keeping employment in the local town, or maintaining a safe environment. Usually these items can be treated by willingness to pay. In a corporate setting, where money is being invested to create value, willingness to pay added to the monetary cash flow seems to work better than multiattribute utility theory. It gives a clearer line of sight to value creation.

## Instant Decision Analysis

In the modern world of the Internet and hyperactivity, everyone wants and expects instant gratification. There is little time to stop and think. Can decision analysis deliver results in this new paradigm?

Yes and no! Deep complex problems, especially ones with high organizational (social-political) involvement take time to work through. One the other hand, computer power and the Internet have put analysis and information retrieval on a new footing. In repetitive classes of problems, where participants can be trained to understand their roles and expectations in the decision-making process, there is great potential for rapid high-quality decision making. Repetitive decision classes allow formulation of reusable decision models, quick links to appropriate information sources, and standing interrelationships among people who know their roles.

Importantly, the customer has changed over the years. When I began doing decision analysis in 1964, few managers or executives had even heard of decision or probability theory, and computers were still huge mysterious machines in large air-conditioned rooms. These executives needed training and convincing every step of the way. Today, most executives have been exposed to decision and probability theory, use their own computers every day, use the Internet fluently, and are predisposed to accept decision analysis methods and results if delivered on their terms. Tom Friedman has just published a provocative book on the revolutionary changes taking place, called *The World is Flat* (Friedman [3]).

However, the new paradigm is very different from the old decision-consulting paradigm. In brief, some of the key shifts are:

- from decision doctor to capability builder,
- from client to customer,
- from consultant knows best to customer knows best,
- from managing large projects to empowering teams,
- from "custom tailored" to "off the rack",
- from time out for a DDP to decision quality in the work flow, and
- from empowered consultants to empowered customers.

For the last five years, I have been part of a software and training startup, SmartOrg, Inc., which has been learning how to deliver into this new paradigm. We have been successful in helping many organizations with dozens of innovation decisions (e.g., R&D, product development) and with balancing portfolios of decisions. The new paradigm can work, but delivering on this promise is very demanding, and the customers are just learning to differentiate between data-rich operational systems and more judgmentally oriented strategic systems. However, the early adopters will attain a new source of competitive advantage: rapid high-quality decision making.

## The Professional Decision Analyst

### Whom Do You Work For?

In theory, a decision analyst serves a decision maker. On occasion, you might be engaged by an individual allocating his or her own resources to ends you consider ethical. This single individual has the choice of making all of the assessments or of delegating some of them to others. The axioms and principles of decision analysis demand that he ultimately accepts inputs from others as his own, such as probability and value assessments. For example, if he engages an expert to assess a probability distribution on sales of a new product given a certain pricing policy, he should review the expert's thinking and assessments and then assert from everything he knows, including his confidence in the expert and the decision analyst who made the assessment, that he believes these assessments represent his own judgments as he makes this decision. On occasion, decision makers do reject expert assessments.

However, when you are engaged by an organization, who is the decision maker? If you are lucky, there will be a single fiduciary responsible for the decision at hand. Here the situation is similar to the personal one, except that both you and the decision maker are ethically obligated to serve the organization who has engaged you, not decision maker self-

interests. Because of bounded rationality, you may not have a clear line of sight to the total organizational situation, but you and the decision maker should be aligned in this purpose. If the decision maker is furthering his or her own agenda at the expense of the organization, I advise you not to take the assignment.

My philosophy is to always work toward the best interest of the organization that engages me. When the single decision maker is not so clear, I make sure the team is working together in the interests of their organization, and that they are willing to make their arguments through a logical decision analysis rather than organizational politics. The arguments should be about the structure and assessments, not directly about picking the alternative they may initially advocate. Sensitivity analysis often shows that what seem to be major differences lead to identical decisions. It directs them to explore the most important difference, often leading to gathering additional information.

The ultimate limitation is the desire of people to be rational. If that desire is not sufficiently present, this analysis is likely to be professionally and ethically unsatisfactory.

### Scope of Responsibility

Another concern is the responsibility of the decision analysis for the content of the analysis. Should a decision analyst provide some of the content, or vouch for the content? When a decision analyst recommends a decision, most decision makers do not clearly distinguish a logical recommendation,—"this is the best you can do with your information"—from an absolute recommendation—"I recommend that this action is best for you." I have had many intense discussions with colleagues on this issue.

Personally, I come down with mixed advice. Providing his own detailed assessments leads the decision analyst away from objectivity. Once you provide inputs, you have a credibility need to defend them, and you might spend too much time on your own area of responsibility rather than conducting a balanced analysis. However, being generally knowledgeable about the field in which you are working helps with communication and rapid analysis, as long as you can and do still ask the "dumb" rhetorical questions that lead to new thoughts. An expert usually cannot do that.

However, when I make a recommendation, I have moved from the position of logical recommendation to a holistic one. A decision analyst is not just an observer, but also an integral part of bounding the problem, developing a sound model, evoking assessments, etc. I am willing to say that having conducted a decision analysis, as far as I know we have appropriately framed and modeled the problem, considered an adequate set of alternatives, captured reliable information from trustworthy and knowledgeable sources, come to well-reasoned recommendations, and provided a setting to achieve commitment to action. The ultimate test in a business situation is that given what I now know (and the decision analyst often ends up with the most complete perspective on the situation), would I be willing to bet my own money this way? If I have truly engineered a good decision, I should be eager to make this bet.

## The 10 Commandments of Decision Analysis

1. Decision analysis is the one master discipline of decision engineering—do not get lost following fads.

2. A single decision maker is the fiduciary responsible for declaring the decision and accepting delegated analyses and assessments—work for the decision maker.

3. Strive to construct a single value measure, in monetary terms—nobody ever put a score in the bank.

4. Have no regret—produce the biggest pie for the organization for whom you are working—do not covet the pie you did not get or the other guy's pie.

5. Beware of internal rates of return—nobody ever put a rate of return in the bank either.

6. Beware of difference lotteries—they never happen.

7. Beware of triage—you can probably add more value to the big "no-brainers" than to the borderline cases where the choice has little impact on value.

8. If the problem is hard, change your frame—avoid picking constraints as problem bounds.

9. Start simple and iterate—use the simplest model that gives insight and decision clarity.

10. Change with the times and keep up with the new paradigm—the world is flat, after all.

## References

[1] Ali Abbas, Dave Reiter, Carl Spetzler, and Steve Tani. Teaching teens how to make good decisions. *OR/MS Today* 31(4), 2004.

[2] Stephen R. Covey. *The Seven Habits of Highly Effective People.* Covey Leadership Center, Fireside, NY, 1989.

[3] Thomas L. Friedman. *The World Is Flat: A Brief History of the Twenty-First Century.* Farrar, Straus and Giroux, New York, 2005.

[4] Malcolm Gladwell. *Blink: The Power of Thinking Without Thinking.* Little, Brown & Co., New York, 2005.

[5] D. Kahneman and A. Tversky. Judgment under uncertainty: Heuristics and biases. *Science* 185:1124–31, 1974.

[6] D. Kahneman and A. Tversky. Prospect theory: An analysis of decision under risk. *Econometrica* 47:263–291, 1979.

[7] Ronald A. Howard. Decision analysis: Applied decision theory. D. B. Hertz and J. Melese, eds. *Proceedings of the Fourth International Conference on Operational Research*, Wiley-Interscience, New York, 55–71, 1966.

[8] Ronald A. Howard. On making life and death decisions. R. C. Schwing and W. A. Albers, eds. *Societal Risk Assessment, How Safe Is Save Enough?* General Motors Research Laboratories, Plenum Press, New York, 89–113, 1980.

[9] Ronald A. Howard and James E. Matheson. Influence diagrams. Department of Engineering-Economic Systems, Stanford University, Stanford, CA, 1979.

[10] Ronald A. Howard and James E. Matheson, eds. *Readings on the Principles and Applications of Decision Analysis.* Strategic Decision Group, Menlo Park, CA, 1983.

[11] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives.* John Wiley & Sons, New York, 1976.

[12] Gary Klein. *Sources of Power.* MIT Press, Cambridge, MA, 1998.

[13] Roger Lowenstein. *When Genius Failed: The Rise and Fall of Long-Term Capital Management.* Random House, New York, 2000.

[14] David Matheson. When should you reexamine your frame? Ph.D. dissertation, Department of Engineering-Economic Systems, Stanford University, Stanford, CA, 1990.

[15] David Matheson and Jim Matheson. *The Smart Organization: Creating Value Through Strategic R&D.* Harvard Business School Press, Boston, MA, 1998.

[16] David Matheson and James E. Matheson. Smart organizations perform better. *Research Technology Management* 44(4):49–54, 2001.

[17] Peter McNamee and John Celona. *Decision Analysis for the Professional*, 3rd ed. SmartOrg, Inc., Menlo Park, CA, 2001.

[18] Howard Raiffa. *Decision Analysis.* Addison Wesley, 1968.

[19] J. Edward Russo and Paul J. H. Schoemaker. *Decision Traps: The Ten Barriers to Brilliant Decision-Making and How to Overcome Them.* Fireside, New York, 1989.

[20] Carl S. Spetzler and Tom Keelin. Decision quality: Opportunity for leadership in total quality management. Strategic Decision Group, 1992.

[21] Carl S. Spetzler and Carl-Axel S. Staël van Holstein. Probability encoding in decision analysis. *Management Science* 22:340–358, 1975.

[22] J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, Princeton, NJ, 1944.

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS | ISBN 1-877640-21-2
DOI 10.1287/educ.1053.0011

# Operations Research in Experimental Psychology

***J. Neil Bearden, Amnon Rapoport***
Department of Management and Policy, University of Arizona, Tucson, Arizona 85719
{jneilb@gmail.com, amnon@u.arizona.edu}

**Abstract** This chapter reviews some of the uses of operations research methods in experimental psychology. We begin by describing some basic methodological issues that arise in the study of human decision making. Next, we describe in more detail research in experimental psychology that has used methods common to operations research—such as dynamic programming—to understand sequential observation and selection behavior. We then suggest some ways in which experimental psychology and operations research can each provide the other with new research questions.

**Keywords** experimental psychology; experimental methodology; dynamic programming

## Introduction

Theories of decision making are typically classified in one of three ways. *Normative* theories provide a framework for understanding how decisions should be made. Implicitly at least, these theories quite often rely on rational agents who have unlimited computational and cognitive capacities. *Descriptive* theories help explain how actual—rather than ideal—agents make decisions. Typically these theories emerge from experimental studies of human decision makers. *Prescriptive* studies of decision making are aimed at determining how actual decision makers (DMs) could behave more in accord with the dictates of normative theories with some systematic reflection (see Raiffa [42] for a classic treatment). To determine appropriate prescriptions, it is necessary to understand how it is that decision making goes wrong—that is, how actual decision making departs from normative decision making (von Winterfeldt and Edwards [68]).

The normative theories of decision making most often emerge from two fields: economic theory (including game theory) and operations research (OR) (Larichev [34]). Utility theory, as developed by Neumann and Morgenstern [66], and later extended by Savage [54] to accomodate subjective probabilities, has received considerable attention by experimental psychologists and, more recently, by experimental economists. Just about any review of the field of behavioral decision theory—particularly those from the 1960s, 1970s, and 1980s—dedicates considerable space to discussions of utility theories of different sorts and how actual human decision making compares to them (e.g., Rapoport and Wallsten [49], Slovic et al. [59]). Camerer [12] should be consulted for an up-to-date review of experimental tests of game-theoretic predictions of behavior. More complex decision problems that involve solving optimization problems of one sort or the other have received less experimental attention. In the area of dynamic decision making, normative theories that are standard in operations research have been relatively neglected by experimental psychologists.

One might wonder why psychologists should be concerned with research in OR. If asked, a number of answers can be given. First, coming from an applied field, OR problems tend to have some correspondence to the kinds of problems likely to be faced by actual DMs. Second, OR problems tend to be clearly formulated, with assumptions and predictions fully specified, thereby increasing their testability. Quite often, little is required to turn an abstract OR

problem into one that can be tested in the laboratory. Finally, optimality results from OR can be useful in understanding the decision policies employed by human DMs.

Before proceeding, a brief comment on the logic of using optimal normative theories to understand decision behavior is in order. Because the assumptions of a normative theory entail the predictions of the theory, comparing empirical decision data to normative predictions can be informative. First, consider the case in which the decision data are incompatible with the predictions. Supposing that one's experiment is designed and run properly (issues that we discuss in the next paragraph), simple *modus tollens* reasoning allows one to conclude that at least one assumption underlying the normative theory is violated in the human decision process. More formally, suppose a theory $T$ is composed of a conjunction of assumptions $T = \{A_1 \wedge \ldots \wedge A_k\}$, and that $T$ entails some outcome $O$. Then, if we observe $\neg O$, we know that $\neg A_i$ is true for at least one assumption $A_i$ of the theory. One can then begin to get an idea of the nature of the difference between the normative model and the actual decision behavior by modifying the assumptions of the former to derive predictions consistent with the latter. One need not take the position that human decision behavior *ought* to be consistent with the dictates of normative theory for the theory to be useful. Without taking a stance on the ought issue, one can simply use the normative theory as a reference point—a benchmark—for evaluating behavior. Similar arguments have been offered elsewhere in favor of the use of optimality theory in theoretical biology (e.g., Hutchinson and McNamara [30], Parker and Smith [39]).

However, there is an asymmetry: When behavior is consistent with normative predictions, fewer inferences are permissible. One cannot conclude that subjects are, for example, performing the calculations needed to arrive at the normative predictions—say, by solving a linear programming problem. Rather, one can only say that their decision behavior is consistent with the theory.

Of course, a number of auxiliary assumptions go into experimentally testing normative theories in the laboratory (or any theory, for that matter; see, e.g., Hempel [25], Quine [41]). One must assume that the subjects fully understand the decision problems they face, that they are motivated, and that basic protocols of good experimental procedure are followed (Hertwig and Ortmann [26]). To ensure that subjects fully understand a task, instructions should clearly explain the task and provide a number of examples demonstrating the "rules of the game." By offering subjects nonnegligible sums of money contingent on their performance, one can ensure that they approach laboratory problems with a level of seriousness at least close to that with which they approach problems outside the laboratory. If subjects behave suboptimally because they do not really care about the outcomes of their decisions, then little is learned. If, however, they exhibit departures from normative theory when real money is on the line, something interesting may be going on. Good basic experimental protocols are too numerous to name and largely depend on the nature of the experimental task. We do not discuss these here.

Below, we review some of the work in experimental psychology that has relied on computational procedures from operations research. In particular, we focus on multistage decision problems that can be solved by dynamic programming. We do so for two reasons. First, this will provide a certain cohesion. Second, much of our own work involves experimental investigations of behavior in dynamic decision problems. Hence, when discussing these problems, we can give an insider's view of ways in which experimentalists think about the OR methods and how we think these methods can inform us about human cognition.

Although we restrict this review to experimental studies of sequential observation and selection behavior, OR researchers should be aware that other areas of dynamic decision making have been brought into the laboratory. Toda [63] pioneered the study of multistage decision behavior more than forty years ago. He devised a one-person game called the "fungus-eater" game, in which subjects were asked to control the sequential search of a robot that attempts to maximize the amount of a valuable resource ("uranium") it collects,

while ensuring that it has sufficient fuel. All of this happens on a hypothetical planet on which uranium and fuel are distributed randomly. Other dynamic decision problems that were studied experimentally in the last 40 years, in which dynamic programming has been used to compute the optimal decision policy, include control of explosive systems (Rapoport [43, 44]), portfolio selection over discrete time and multistage gambling (Ebert [19], Rapoport et al. [50]), vehicle navigation (Anzai [1], Jagacinkski and Miller [31]), health management (Kleinmuntz and Thomas [33]), inventory control (Rapoport [45], Sterman [61]), and fire-fighting (Brehmer and Allard [10]). For cumulative reviews of this literature, see Rapoport [46], Brehmer [9], Sterman [62], Kerstholt and Raaijmakers [32], and, more recently, Busemeyer [11] and Diederich [17].

## Optimal and Empirical Results on Optimal Stopping

The problems that we will discuss in this chapter involve sequential search, selection, and assignment. Problems of this sort are referred to in different literatures by different names. Sometimes they are dubbed *optimal* stopping problems; at other times, one sees them referred to as *optional* stopping problems. Depending on their precise formulation and on the literature in which they appear (e.g., OR, economics, psychology), the problems are sometimes referred to as job-search problems, rent-seeking problems, secretary problems, etc. All of the problems in this class share an important feature with a number of interesting real-world choice problems, namely the choice alternatives are encountered *sequentially*. Given their multistage structure, dynamic programming (DP) methods are often invoked to find optimal decision policies for these problems.

### Full-Information Problems

Suppose the DM can observe as many as $n$ observations $X_j$ $(j = 1, \ldots, n)$, but for each observation she must pay a fixed cost $c \geq 0$. The observations are drawn independently according to the density function $f(x)$, which is known to the DM. The DM's objective is to maximize the expected value of her selected observation minus her total costs; she cannot return to an observation once she has rejected it, and if she reaches the $n$th one, she must accept it. Because the DM knows the distribution from which the observations are sampled, this problem is known as a *full-information* optimal stopping problem.

Under the optimal policy, the DM should stop on observation $j$ whenever the value of the draw $x_j$ exceeds the expected value of moving to the next stage (i.e., to $j+1$) and behaving optimally thereafter, denoted $V^*_{j+1}$. Because the DM is forced to accept the $n$th observation, if reached, $V^*_n = \int_{-\infty}^{\infty} x f(x) dx - c$. Hence, at stage $n-1$ the optimal DM sets her *cutoff* to $s^*_{n-1} = V^*_n$. The cutoffs for each stage determine the observation values that the optimal DM finds acceptable at each stage; specifically, the DM stops on observation $j$ whenever $x_j \geq s^*_j$. Thus, more generally, $V^*_{j+1} = s^*_j$. The optimal cutoffs for each stage $j = n-1, \ldots, 1$ can be obtained by computing the recurrence (Sakaguchi [51])

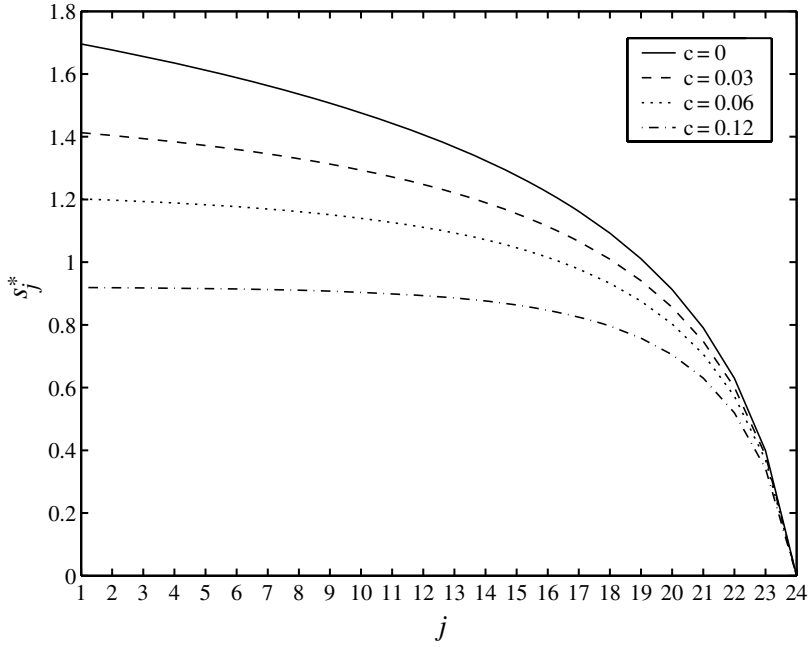
$$s^*_j = \int_{s^*_{j+1}-c}^{\infty} \left[x - \left(s^*_{j+1} - c\right)\right] f(x)\, dx + s^*_{j+1} - c. \tag{1}$$

Optimal policies for observations from a standard normal distribution with $n = 24$ are shown in Figure 1 for various values of $c$. These well-known results form the classical basis for optimal stopping problems. We present them here merely to set the stage for a discussion of experimental studies of optimal stopping, which we turn to next.

### Experimental Studies of Full-Information Optimal Stopping

Rapoport and Tversky [48] conducted an experimental test of full-information optimal stopping. One difficulty in testing the full-information problem is that it requires that the DM

FIGURE 1. Optimal cutoffs $s_j^*$ for various costs for the standard normal distribution.




know the distribution from which the observations are taken (cf. Rapoport and Tversky [47]). Simply telling a naïve subject that observations are taken from, for example, "a normal distribution with mean 100 and standard deviation 10" is obviously problematic, as the subject is unlikely to have a good grasp of precisely what this means. To get a strong test of the predictions of the optimal policy, one must ensure that the subjects have a good sense of the distribution. To guarantee that their subjects understood the nature of the distribution from which observations were taken, Rapoport and Tversky had them first perform a signal detection task in which observations were taken from two normal distributions $A$ and $B$ with a common standard deviation (167), but with different means (1,630 and 1,797, respectively). The subjects were required to judge whether a sample was from $A$ or $B$. Over a six-week period, five times a week, subjects observed a total of 7,800 samples from each of the two distributions. Hence, presumably, the subjects ultimately had a good sense of the nature of the distributions.

Once the distributions were learned, the subjects performed optimal stopping tasks over several sessions. In each session, the subjects were told whether the values would be sampled from $A$ or from $B$, and that $n = 24$. They were also told the cost $c$ for each sampled observation, which was held constant within a trial. The cost values used in the study were 0, 5, 10, or 20. (Rapoport and Tversky studied optimal stopping problems both with and without recall, but for brevity we will only discuss results from the latter.)

The main results are summarized in Table 1. For each cost condition, Rapoport and Tversky observed that, on average, subjects tended to sample fewer observations than predicted by the application of the optimal policy. Rapoport and Tversky suggested that the stopping results were reasonably consistent with subjects using a fixed-threshold cutoff rule. Under this decision rule, the subject stops on an observation $x_j$ whenever $x_j \geq s$, where $s$ is fixed for all $j$ $(j = 1, \ldots, n)$. Thus, assuming this is the correct model for the choice behavior, the subjects were insensitive to the search horizon, and sought a single "target" value.

Other experimental studies of full-information optimal search have examined the effects of allowing the recall of previously encountered observations (Rapoport and Tversky [47]); others (e.g., Hey [27, 28, 29]) have more closely examined the factors that influence decisions

TABLE 1. Mean number of observations for each cost condition from Rapoport and Tversky [48].

| | $c=0$ | $c=5$ | $c=10$ | $c=20$ |
|---|---|---|---|---|
| Average number of draws | 9.51 | 9.61 | 8.44 | 4.11 |
| Expected number of draws | 10.42 | 11.25 | 10.80 | 4.50 |

*Note.* The expected number of observations are not monotonically decreasing in cost because the values shown are based on the application of the optimal policy to the empirically sampled observation values, not on the limiting expectation under random sampling.

to stop, such as the influence of the history of observation values (e.g., whether the sequence has been increasing or decreasing).

In our view, the main shortcoming of full-information optimal stopping problems—when taken as models of actual human search problems—is the assumption that $f(x)$ is known to the DM. This assumption is critical for testing the optimal search model because the values of $s_j^*$ are located at the right tail of the distribution $f(x)$, and are, therefore, very sensitive to deviations from them. Perhaps in many situations DMs do have a "good sense" of the operative distribution; in many others, we suspect, this condition is not met. Next, we turn to *no-information* search problems that do not require that the DM have any knowledge about the distribution from which observations are sampled.

## No-Information Problems

The standard no-information stopping problem is the "Secretary Problem." To contrast it with other no-information problems that we discuss later, we will refer to the most common formulation of the problem as the *classical secretary problem* (CSP). It can be stated as follows:

1. There is a fixed and known number $n$ of applicants for a single position who can be ranked in terms of quality from best to worst with no ties.

2. The applicants are interviewed sequentially in a random order (with all $n!$ orderings occurring with equal probability).

3. For each applicant $j$ the DM can only ascertain the *relative rank* of the applicant, that is, how valuable the applicant is relative to the $j-1$ previously viewed applicants.

4. Once rejected, an applicant cannot be recalled. If reached, the $n$th applicant must be accepted.

5. The DM earns a payoff of 1 for selecting the applicant with *absolute rank* 1 (i.e., the overall best applicant in the population of $n$ applicants) and 0 otherwise.

The payoff-maximizing strategy for the CSP, which simply maximizes the probability of selecting the best applicant, is to interview and reject the first $t^*-1$ applicants and then accept the first applicant thereafter with a relative rank of 1 (Gilbert and Mosteller [23]). The optimal cutoff can be obtained by

$$t^* = \min\left\{t \geq 1 : \sum_{k=t+1}^{n} \frac{1}{k-1} \leq 1\right\}. \tag{2}$$

Interestingly, $t^*$ converges to $ne^{-1}$ as $n \to \infty$, and the policy selects the best applicant with probability $e^{-1}$.

A historical review of the CSP can be found in Ferguson [20] and in Samuels [52]. Numerous generalizations of the CSP have been proposed. For example, Corbin [15] and Yang [69] presented procedures for computing optimal policies for secretary problems in which options can be recalled with probabilistic success; Pressman and Sonin [40] discussed problems in which the number of applicants $n$ is itself unknown, but the DM does know the distribution from which $n$ is sampled.

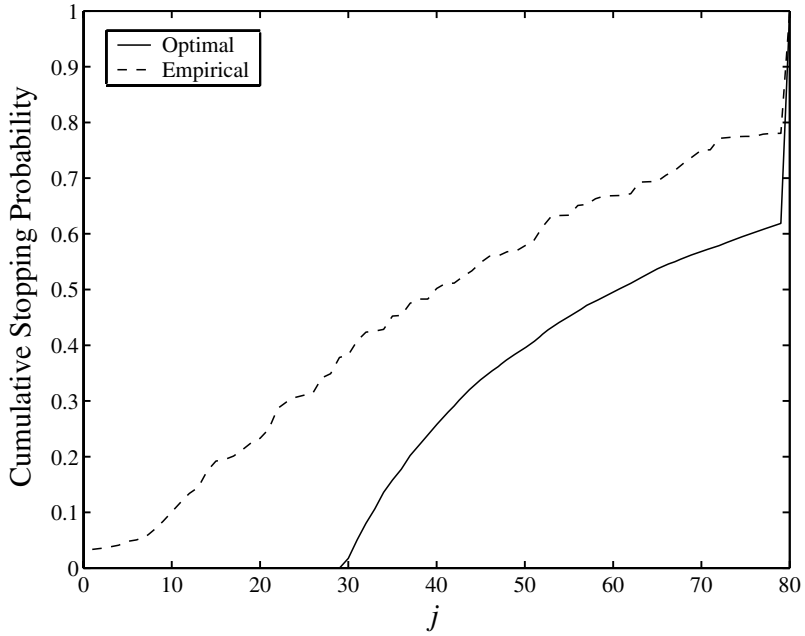
## Experimental Studies of the CSP

Seale and Rapoport [55] tested two versions of the CSP, one with $n = 40$ and another with $n = 80$. Using a between-subjects experimental design, each subject in their experiment was either in the $n = 40$ or $n = 80$ condition. Each subject played a total of 100 independent instances (trials) of the CSP in a computer-controlled environment. The subjects were first given a cover story that described their task as one of hiring applicants for a position. Each of the trials proceeded as follows. The *relative rank* of the first applicant was displayed (which, by definition, was always 1). The subjects could then choose to select (hire) the applicant or proceed (interview) to the next applicant. Whenever the subject chose to proceed from applicant $j$ to applicant $j + 1$, the computer displayed the relative ranks of all applicants up to $j + 1$. Once a subject selected an applicant, the *absolute ranks* of all $n$ applicants were displayed. If the subject selected the best overall applicant, the computer informed her that she had made a correct selection and added her payoff for that trial to her cumulative earnings. For the $n = 40$ condition, a subject earned \$0.30 each time she selected an applicant with absolute rank 1; for the $n = 80$ condition, the corresponding payoff was \$0.50.

The probability of stopping on applicant $j$ or sooner under the optimal policy if $n = 80$ is displayed in Figure 2. Figure 2 also shows the proportion of times that subjects in Seale and Rapoport's $n = 80$ condition stopped on applicant $j$ or sooner. Note that the empirical curve is shifted considerably to the left of the optimal curve, demonstrating a propensity of the subjects to stop earlier than is predicted by the optimal policy.

From the results displayed in Figure 2, we can see that the subjects are behaving suboptimally; however, we cannot infer how it is that the subjects are making their decisions: We cannot infer their actual decision policies. Seale and Rapoport tried to get a handle on the DM's underlying (unobservable) decision policies by competitively testing three different single-parameter decision policies or heuristics. The particular policies studied were chosen for their psychological plausibility: A priori they seemed like policies that people might reasonably use. We will consider each of these in turn and will describe some of the properties of the policies that were later derived by Stein et al. [60].

FIGURE 2. Optimal and empirical cumulative stopping probabilities from the $n = 80$ condition in Seale and Rapoport [55].



**The Cutoff Rule (CR).** Do not accept any of the first $t-1$ applicants; thereafter, select the first encountered *candidate* (i.e., an applicant with relative rank 1). This rule has as a special case the optimal policy for the CSP for which $t^*$ is obtained by Equation (1). In addition to the optimal policy, the CR can be set to begin accepting candidates earlier in the sequence ($t < t^*$), or later ($t > t^*$).

**Candidate Count Rule (CCR).** Select the $h$th encountered candidate. Note that this rule does not necessarily skip any applicants; it only considers how many candidates have been observed, not how deep the DM is in the applicant sequence.

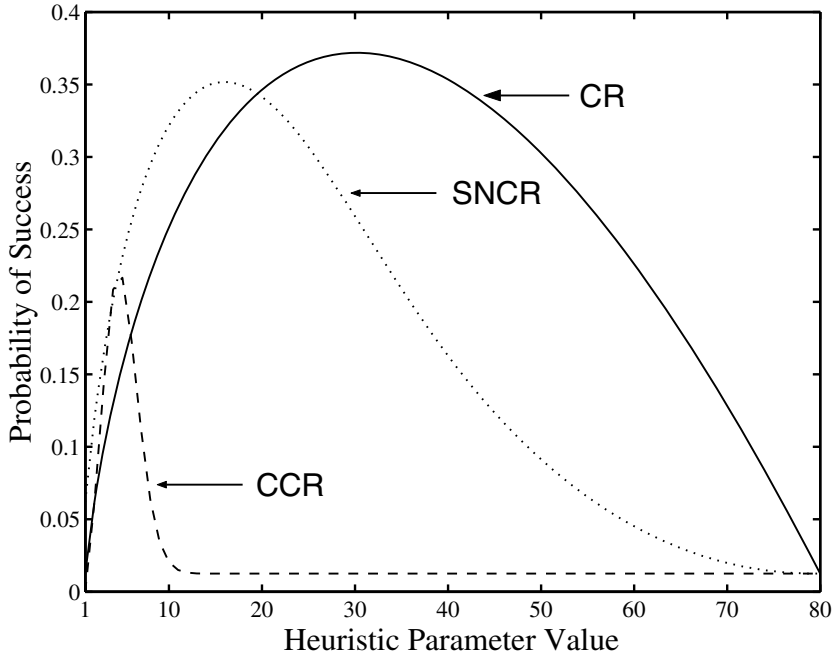
**Successive Noncandidate Rule (SNCR).** Select the first encountered candidate after observing $g$ successive noncandidates (i.e., applicants with relative rank $> 1$).

Each of these three heuristic decision policies can be represented by a single parameter ($t$, $h$, and $g$). Before returning to the results of Seale and Rapoport [55], let us examine the theoretical performance of the heuristics. We know that no heuristic can outperform the CR with $t = t^*$. But how well can the others do?

Results for each the heuristics for $n = 80$ CSP are displayed in Figure 3. The horizontal axis corresponds to the values of each heuristic's single parameter ($t$, $h$, and $g$ for the CR, CCR, and SNCR, respectively). There are a number of interesting features of these results. First, the CR and SNCR heuristics strongly outperform the CCR heuristic. In addition, when properly tuned, the SNCR can perform nearly as well as the optimal policy (i.e., the CR with $t = t^*$), with the former earning about 95% of what is earned by the latter. Finally, the CR is relatively robust to misparameterization. Whenever $0.63t^* \leq t \leq 1.47t^*$, the DM can expect to earn at least 90% of what is earned under the optimal policy. From sensitivity analyses, the flatness of payoff functions for a number of problems studied by behavioral decision theorists has often been noted (see, e.g., Rapoport [46], von Winterfeldt and Edwards [67]).

Seale and Rapoport [55] fit each of the three heuristics just described to each subject's choice data. For each subject and each heuristic, they used a *brute force* search procedure to find the heuristic's parameter that minimized the number of incorrect stops (or violations)

FIGURE 3. Probability of success for three CSP heuristics.




*Note.* These results are based on the derivations presented in Stein et al. [60].

predicted by the heuristic. Put differently, if a subject stopped on applicant $j$ on a given trial and the heuristic predicted that the subject should stop on applicant $j' \neq j$, then a violation was obtained. Hence, a particular parameterization of a heuristic could produce between 0 and 100 (the total number of experimental trials) violations for a given subject.

The results for both the $n = 40$ and the $n = 80$ conditions were consistent; for brevity, we will restrict discussion to those from the latter condition. The most important result is that the subjects' decision data were best characterized by the CR heuristic. For 21 of 25 subjects, the CR best fit the data; the SNCR best fit the choices for 8 of the 25 subjects; and the CCR fit best for only 1 subject. (The number of best-fitting heuristics sums to more than 25 because of ties. For some subjects, the best-fitting CR and SNCR heuristics produced the same number of violations.) The authors compared the subjects' CR cutoffs ($t$ values) to the optimal cutoff, and found that $t < t^*$ for 21 subjects, $t > t^*$ for 3 subjects, and $t = t^*$ for only 1 subject.

This illustrates one logic that can be used to draw inferences from experimental data about the decision policies employed by actual decision makers. One might be tempted to argue that the three heuristics tested by Seale and Rapoport were arbitrarily chosen, and that any number of other heuristics *could* have been tested. The latter is certainly true. However, it is up to the critic to propose specific alternatives for the former argument to have force. Further, all explanations are going to be underdetermined (see, e.g., Van Fraassen [65]), even those that come from our most successful scientific theories. We believe, however, that there is another criticism of the procedures used by Seale and Rapoport that should be carefully considered. Although the CR heuristic best accounted for most subjects' data, it did not do so perfectly. For most subjects, the best-fitting heuristic could only explain about 60%–70% of the stopping decisions. What drove the other 30%–40% of the stopping decisions? This question led Bearden and Murphy [3] to formulate a stochastic version of the CR and fit it to Seale and Rapoport's data; however, they remained agnostic on the source of the threshold variability. A fuller account of the decision behavior in the CSP should address this issue.

Several extensions of the CSP have been studied experimentally. Seale and Rapoport [56], for example, looked at a problem in which the DMs do not know $n$, but only its distribution. Compared with the optimal policy, the subjects tended not to search enough. Zwick et al. [71] examined the effect of search cost and probabilistic recall of previously seen applicants. The most important finding from this study is that the subjects tended to search for too long with positive search costs, whereas they did not search enough when search costs were set at 0.

The nothing-but-the-best payoff structure of the CSP seems patently unrealistic. One can imagine few situations in which selecting the best option from a pool yields positive utility, and selecting any other option yields no utility. What, then, might we have learned about actual human decision making in environments in which options are encountered sequentially from the experimental studies of the CSP? Might it be the case that the observed early stopping is simply a consequence of the CSP's special payoff function? That is, perhaps we now know that actual DMs cannot perfectly solve the CSP probability puzzle, but we have learned nothing about how people might actually go about selecting secretaries. To address this issue, we studied an extension of the CSP in which the DM's objective is to find a *good*—not necessarily the best—alternative. Next, we describe the formal problem, how to compute its optimal policy, and some experimental studies of decision making in this more realistic context.

## The Generalized Secretary Problem

Consider a variant of the secretary problem in which the DM earns a positive payoff $\pi(a)$ for selecting an applicant with absolute rank $a$, and assume that $\pi(1) \geq \cdots \geq \pi(n)$. Mucci [38] proved that the optimal search policy for this problem has the same threshold form as that of the CSP. Specifically, the DM should interview and reject the first $t_1^* - 1$ applicants, then

between applicant $t_1^*$ and applicant $t_2^* - 1$ she should only accept applicants with relative rank 1; between applicant $t_2^*$ and applicant $t_3^* - 1$ she should accept applicants with relative ranks 1 or 2; and so on. As she gets deeper into the applicant pool, her standards relax and she is more likely to accept applicants of lower quality. The values of the $t^*$ depend on the operative payoffs and on the number of applicants $n$.

We obtain what we call the *generalized secretary problem* (GSP) by replacing 5 in the CSP with the more general payoff function:

5′. The DM earns a payoff of $\pi(a)$ for selecting an applicant with absolute rank $a$ where $\pi(1) \geq \cdots \geq \pi(n)$.

Clearly, the CSP is a special case of the GSP in which $\pi(1) = 1$ and $\pi(a) = 0$ for all $a > 1$. Results for other special cases of the GSP have appeared in the literature. For example, Moriguti [37] examined a problem in which a DM's objective is to minimize the expected rank of the selected applicant. This problem is equivalent to maximizing earnings in a GSP in which $\pi(a)$ increases linearly as $(n - a)$ increases.

## Finding Optimal Policies for the GSP

We begin by introducing some notation. The orderings of the $n$ applicants' *absolute ranks* is represented by a vector $\mathbf{a} = (a_1, \ldots, a_n)$, which is some random permutation of the integers $1, \ldots, n$. The *relative rank* of the $j$th applicant, denoted $r_j$, is the number of applicants from $1, \ldots, j$ whose absolute rank is smaller than or equal to $a_j$. A *policy* is a vector $\mathbf{s} = (s_1, \ldots, s_n)$ of nonnegative integers in which $s_j \leq s_{j+1}$ for all $1 \leq j < n$. The policy dictates that the DM stop on the first applicant for which $r_j \leq s_j$. Therefore, the probability that the DM stops on the $j$th applicant, conditional on reaching this applicant, is $Q(s_j) = s_j/j$. A DM's *cutoff* for selecting an applicant with a relative rank of $r$, denoted $t_r$, is the smallest value $j$ for which $r \leq s_j$. Hence, a policy $\mathbf{s}$ can also be represented by a vector $\mathbf{t} = (t_1, \ldots, t_n)$. Sometimes, the cutoff representation will be more convenient.

Optimal thresholds can be computed straightforwardly by combining numerical search methods with those of dynamic programming. We will describe below a procedure for doing so. Lindley [36] described a similar method, and another was briefly sketched in Yeo and Yeo [70]. A fuller treatment is presented in Bearden and Murphy [3].

The probability that the $j$th applicant out of $n$ whose relative rank is $r_j$ has an absolute (overall) rank of $a$ is given by

$$\Pr(A = a | R = r_j) = \frac{\binom{a-1}{r-1}\binom{n-a}{j-r}}{\binom{n}{j}} \tag{3}$$

when $r_j \leq a \leq r_j + (n - j)$, otherwise, $\Pr(A = a | R = r_j) = 0$. Thus, the expected payoff for selecting an applicant with relative rank $r_j$ is

$$\mathrm{E}(\pi_j | r_j) = \sum_{a=r_j}^{n} \Pr(A = a | R = r_j)\pi(a). \tag{4}$$

The expected payoff for making a selection at stage $j$ for some stage $j$ policy $s_j > 0$ is

$$\mathrm{E}(\pi_j | s_j) = (s_j)^{-1} \sum_{i=1}^{s_j} \mathrm{E}(\pi_j | r_j = i); \tag{5}$$

otherwise, when $s_j = 0$, $\mathrm{E}(\pi_j | s_j) = 0$. Now, denoting the expected payoff for starting at stage $j + 1$ and then following a fixed-threshold policy $(s_{j+1}, \ldots, s_n)$ thereafter by $V_{j+1}$, the value of $V_j$ for any $s_j \leq j$ is simply

$$V_j = Q(s_j)\mathrm{E}(\pi_j | s_j) + [1 - Q(s_j)]V_{j+1}. \tag{6}$$

As with the optimal policy for the full-information optimal stopping problem described above, the optimal policy for the GSP entails stopping on an applicant with relative rank $r_j$ at stage $j$ whenever the expected payoff for doing so exceeds the expected payoff for proceeding and playing optimally thereafter. This follows directly from the principle of optimality (Bellman [8]). Given that the expected earnings of the optimal policy at stage $n$ are $V_n^* = n^{-1}\sum_{a=1}^{n}\pi(a)$, we can easily find an $s_j^*$ for each $j$ $(j = n-1,\ldots,1)$ by backward induction. Because the last applicant must be selected, $s_n^* = n$; then, for $j = n-1,\ldots,1$,

$$s_j^* = \min\{s \in \{0,\ldots,s_{j+1}^*\} : V_j \geq V_{j+1}^*\}. \tag{7}$$

The expected payoff for following a feasible policy $\mathbf{s}$ is

$$\mathrm{E}(\pi|\mathbf{s}) = \sum_{j=1}^{n}\left[\prod_{i=0}^{j-1}[1-Q(s_i)]\right]Q(s_j)\mathrm{E}(\pi_j|s_j) = V_1, \tag{8}$$

where $Q(s_0) = 0$. Denoting the applicant position at which the search is terminated by $m$, the expected stopping position under the policy is

$$\mathrm{E}(m) = 1 + \sum_{j=1}^{n-1}\left[\prod_{i=1}^{j}[1-Q(s_i)]\right]. \tag{9}$$

Equation (9) can be useful in evaluating the results of actual DMs in GSPs (see, e.g., Bearden et al. [7]).

Optimal cutoffs for several GSPs are presented in Table 2. In the first column, we provide a shorthand for referring to these problems. The first one, GSP1, corresponds to the CSP with $n = 40$. The optimal policy dictates that the DM should search through the first 15 applicants without accepting any and then accept the first one thereafter with a relative rank of 1. GSP2 corresponds to another CSP with $n = 80$. In both, the DM should search through roughly the first 37% and then take the first encountered applicant with a relative rank of 1. GSPs 3 and 4 were discussed in Gilbert and Mosteller [23], who presented numerical solutions for a number of problems in which the DM earns a payoff of 1 for selecting either the best or second-best applicant and nothing otherwise. GSPs 5 and 6 correspond to those studied by Bearden et al. [7] in Experiments 1 and 2, respectively. In the first, the DM searches through the first 20 applicants without accepting any; then between 21 and 42 she stops on applicants with relative rank of 1; between 43 and 52, she stops on applicants with relative rank 1 or 2; etc.

## Experimental Studies of the GSP

Recall that Seale and Rapoport [55] found that subjects playing the CSP (GSPs 1 and 2, actually) tended to search insufficiently into the applicants before making a selection—i.e., on average they stopped too soon. Above, we suggested that the CSP might be quite

Table 2. Several GSPs and their optimal policies.

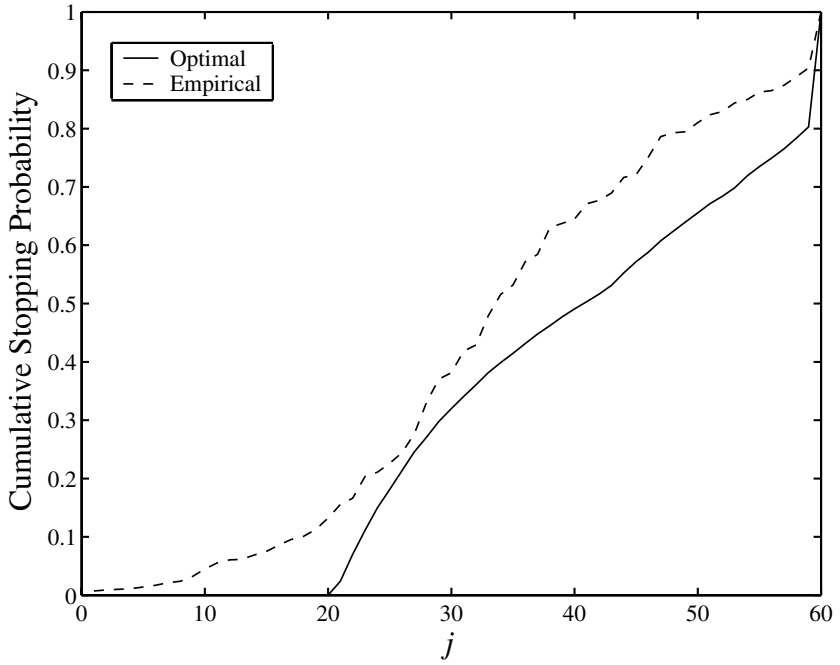
| GSP | $n$ | $\pi = (\pi(1),\ldots,\pi(n))$ | $\mathbf{t}^* = (t_1^*,\ldots,t_n^*)$ | $\mathrm{E}(\pi\text{—}\mathbf{s}^*)$ | $\mathrm{E}(m)$ |
|---|---|---|---|---|---|
| 1 | 40 | $(1,0,\ldots,0)$ | $(16,40,\ldots,40)$ | 0.38 | 30.03 |
| 2 | 80 | $(1,0,\ldots,0)$ | $(30,80,\ldots,80)$ | 0.37 | 58.75 |
| 3 | 20 | $(1,1,0,\ldots,0)$ | $(8,14,20,\ldots,20)$ | 0.69 | 14.15 |
| 4 | 100 | $(1,1,0,\ldots,0)$ | $(35,67,100,\ldots,100)$ | 0.58 | 68.47 |
| 5 | 60 | $(25,13,6,3,2,1,0,\ldots,0)$ | $(21,43,53,57,58,59,60,\ldots,60)$ | 12.73 | 41.04 |
| 6 | 40 | $(15,7,2,0,\ldots,0)$ | $(14,29,37,40,\ldots,40)$ | 6.11 | 27.21 |

contrived and that this result could possibly be an artifact of the narrow payoff scheme of the CSP. Bearden et al. [7] experimentally tested this using two variants of the GSP, GSPs 5 and 6. In Experiment 1, each subject played 60 instances of GSP5. (They were paid in cash for two randomly selected trials.) It is difficult to say what a realistic payoff function is, but it seems certain that the one for GSP5 is more realistic than the one for the CSPs. For the GSP5, the DM gets a substantial payoff for selecting the best applicant ($25 in the experiment), considerably less—but still significant—for the second best ($13), and so on. This captures the payoff properties of problems in which the DM would like to get a "really good" applicant, which seems like a desire that DMs might often have. Using this more plausible payoff scheme, we find that the early stopping result from studies of the CSP persists. Figure 4 shows the cumulative stopping results for the optimal policy and also those from the experimental subjects for the GSP5. We found that the subjects tend to terminate their searches too soon relative to the optimal policy. Comparing Figures 2 and 4, we can see that the tendency to stop searching too soon is less pronounced in the GSP than in the CSP, but drawing strong inferences from data gathered in two different experimental settings is problematic.

Bearden, Rapoport, and Murphy competitively tested multiparameter generalizations of the three heuristics examined by Seale and Rapoport [55]. The generalized cutoff rule, for example, had thresholds for each of the relative ranks that could entail positive payoffs ($r \leq 6$). Under this rule, the DM stops on applicant $j$ with relative rank $r_j$ whenever $j \geq t_{r_j}$. The candidate count rule and the successive noncandidates rule were extended in the same fashion. Consistent with the findings of Seale and Rapoport [55], the results were best captured by the cutoff rule with cutoffs shifted toward early stopping.

In a second experiment, Bearden, Rapoport, and Murphy experimentally studied the GSP6. In all regards the choice results captured the qualitative properties of those found in the data from the first experiment, including the superiority of the cutoff rule in accounting for the data. After subjects performed 60 trials of the GSP6, they were then asked to

FIGURE 4. Optimal and empirical cumulative stopping probabilities from Experiment 1 in Bearden et al. [7].



perform a probability estimation task. They were shown the relative ranks $r_j$ of applicants in different positions $j$ and asked to estimate the probability that the applicant had various absolute ranks $a$. For example, a subject might have been asked: "What is the probability that applicant 10 (of 40), whose relative rank is 2, has an absolute rank of 2?" Or: "What is the probability that applicant 10 (of 40), whose relative rank is 2, has an absolute rank of 3?" (It is 0.06 for $a = 2$ and 0.09 for $a = 3$.) We only asked the subjects about absolute ranks that could entail positive payoffs, i.e., about $a \leq 3$. We used a strictly proper scoring rule to encourage the subjects to give accurate estimates. Put simply, the subjects made more money as their estimates were closer to the true probabilities, which can be obtained from Equation (3), and there was no incentive for them to misrepresent their true estimates (i.e., to respond strategically).

The results were informative. Overwhelmingly, the subjects tended to overestimate the true probabilities, particularly for applicants early in the sequence. In fact, the estimates were quite often subadditive: For a given applicant position $j$ and relative rank $r_j$, the sum of the probability estimates exceeded 1. Taking these results seriously, the subjects were often *supercertain* that they would obtain positive payoffs for selecting applicants. These findings are consistent with Tversky and Koehler's [64] *support theory*, an empirically successful theory of subjective probability. Under support theory, the subjective probability assigned to an event $E$, $p(E)$, is a function of the support (evidence) one gives to $E$ and its complement $\neg E$; specifically,

$$p(E) = \frac{s(E)}{s(E) + s(\neg E)}, \tag{10}$$

where $s(\cdot) > 0$ is a real-valued measure of the strength of the evidence one can find for $E$. The evidence can be arrived at through a process of memory search or by other means (Bearden and Wallsten [4]). Under the theory, the *focal* event $E$ receives more support than the *nonfocal* event $\neg E$. This aspect of the theory is consistent with considerable evidence from psychology that shows that human information processing is biased toward confirmation seeking (i.e., toward searching for evidence for $E$, rather than for evidence of $\neg E$, which is also normatively relevant). In short, according to the theory, when evaluating the probability that an applicant has a particular absolute rank $a$, one is likely to give insufficient weight to the possibility that the applicant *does not* have an absolute rank $a$, focusing (disproportionately) instead on the event that the applicant's absolute rank *is* $a$.

Perhaps the probability estimation results—at least in part—explain the early stopping finding. If a DM believes it to be certain that stopping early will produce a good payoff, then stopping early is sensible. Put differently, one possible reason for subjects not behaving in accord with the optimal policy is that their representation of the search problem is distorted. Constructing the optimal policy requires the computation of Equation (3), but the estimation data show that subjects' intuitive probability judgments depart quite radically from the true probabilities. This demonstrates the strength of the assumptions required for one to presume that the optimal policy is a priori a reasonable predictor of human decision making in this class of problems.

What, then, of questions regarding the rationality of decision making in sequential search problems? Can we say that people are irrational if they cannot intuit the probabilities that fall out of Equation (3)? Of course not. However, although it is implausible to presume that people should behave in accord with the optimal policy, the optimal policy still provides some basis for evaluating the experimental decision data. While it may be important in economic contexts to anticipate that people will not search optimally, we are still left wondering what underlying psychology drives sequential search decisions. Fortunately, the optimal policy can and has served as a starting point for understanding *how it is* that people make these decisions. As described above, the cutoff rule—of which the optimal policy is a special case—best accounts for the stopping results. Likewise, comparing the subjects' probability

estimates to the true probabilities was informative. Hence, knowledge of the optimal policy can be useful for explanatory purposes.

The GSP captures important features of a number of dynamic decision problems likely to be encountered in the wild (i.e., the so-called "real world"). However, there are, of course, many problems that it does not capture. Quite often we are faced with decision problems in which we must make trade-offs among the attributes of decision alternatives. An academic job may offer a good salary but an undesirable teaching load; a house may be close to one's office (minimizing commute time) but in a poor school district; etc. One can argue that the GSP sidesteps these kinds of problems by collapsing the multiattribute utility of options into a single ranking. Because we are primarily interested in how people actually make decisions, we do not want to assume away this interesting problem of making trade-offs among attributes. We would like to know how people do so, particularly in situations in which options are encountered sequentially. Next, we describe a multiattribute extension of the GSP, describe its solution, and present some experimental findings.

## The Multiattribute Secretary Problem

The *Multiattribute Secretary Problem* (MASP) further generalizes the GSP to applicants with multiple features or attributes. Formally, it is defined as follows:

1. There is a fixed and known number $n$ of applicants for a single position. The applicants differ along $k$ different dimensions or attributes. Within a given attribute, the applicants can be ranked from best (1) to worst ($n$) with no ties. The attributes are uncorrelated.
2. The applicants are interviewed sequentially in a random order (with all $n!$ orderings occurring with equal probability).
3. For each applicant $j$ the DM can only ascertain the *relative ranks* of the applicant's $k$ attributes.
4. Once rejected, an applicant cannot be recalled. If reached, the $n$th applicant must be accepted.
5. For each attribute $i$ of the selected applicant, the DM earns a payoff of $\pi^i(a^i)$, where $a^i$ is the selected applicant's absolute rank on attribute $i$ and $\pi^i(1) \geq \ldots \geq \pi^i(n)$.

Before describing the optimal policy for the MASP, we must introduce some notation. The *absolute rank* of the $j$th applicant on the $i$th attribute, denoted $a^i_j$, is simply the number of applicants in the applicant pool, including $j$, whose $i$th attribute is at least as good as the $j$th applicant's. The $j$th applicant's set of absolute ranks can therefore be represented by a vector $\mathbf{a}_j = (a^1_j, \ldots, a^k_j)$. The *relative rank* of the $j$th applicant on the $i$th attribute, $r^i_j$, is the number of applicants from 1 to $j$ whose $i$th attribute is at least as good as the $j$th's. Similar to the GSP, when interviewing an applicant, the DM observes $\mathbf{r}_j = (r^1_j, \ldots, r^k_j)$, and must make her selection decision on the basis of this information.

Though she only observes relative ranks $\mathbf{r}_j$, the DM's payoff for selecting the $j$th applicant, denoted $\pi_j$, is based on the applicant's absolute ranks $\mathbf{a}_j$; specifically,

$$\pi_j = \sum_{i=1}^{k} \pi^i(a^i_j). \tag{11}$$

An optimal policy for the MASP is one that maximizes the expected value of the selected applicant.

Some related problems have appeared in the OR literature. Gnedin [24] presented the solution to a multiattribute CSP in which the attributes are independent, and the DM's objective is to select an applicant who is best on at least one attribute. Ferguson [21] generalized the problem presented by Gnedin by allowing dependencies between the attributes, and showed that the optimal policy has the same threshold form as the standard single-attribute CSP. Samuels and Chotlos [53] extended the rank-minimization problem of Chow

et al. [13]. They sought an optimal policy for minimizing the sum of two ranks for independent attributes. The rank-sum minimization problem they studied is equivalent to the MASP in which $\pi^1(a) = \pi^2(a) = n - a$. The MASP is more general than these previous problems, as it only constrains the payoff functions to be nondecreasing in the quality of the selected applicant's attributes.

## Finding Optimal Policies for the MASP

The probability that the $i$th attribute of the $j$th applicant whose relative rank on that attribute is $r$ has an absolute (overall) rank of $a$ is given by Equation (3). To simplify matters, we assume that the $k$ attributes are pairwise independent, that is, $\Pr(a^i = a \wedge a^{i'} = a') = \Pr(a^i = a)\Pr(a^{i'} = a')$ for any pair of attributes $i$ and $i'$. (Based on work by Ferguson [21], introducing arbitrary correlations $\rho$ among attributes would likely make the determination of the appropriately corresponding Equation (3) intractable, as for any $j$ it would depend—in complicated ways—on $(\mathbf{r}_1, \ldots, \mathbf{r}_j)$, i.e., on the entire history of relative ranks.) Consequently, the expected payoff for selecting the $j$th applicant is

$$\mathrm{E}(\pi_j|\mathbf{r}_j) = \sum_{i=1}^{k} \sum_{a=r_j^i}^{n} \Pr(A = a|R = r_j^i)\pi^i(a). \tag{12}$$

At each stage $j$ of the decision problem, the DM must decide to accept or reject an applicant knowing only the applicant's relative ranks $\mathbf{r}_j$. We represent a decision policy for each stage $j$ as a set of acceptable $\mathbf{r}_j$ for that stage $\mathbf{R}_j$. Under the stage policy $\mathbf{R}_j$, the DM stops on an applicant with relative ranks $\mathbf{r}_j$ if and only if $\mathbf{r}_j \in \mathbf{R}_j$. The global policy is just the collection of stage policies $\mathbf{R} = \{\mathbf{R}_1, \ldots, \mathbf{R}_n\}$. By Bellman's [8] principle of optimality, for an optimal (global) policy $\mathbf{R}^*$, each subpolicy $\{\mathbf{R}_j, \ldots, \mathbf{R}_n\}$ from stage $j$ to $n$ must also be optimal. Given this property, we can find the optimal policy using straightforward dynamic programming methods by working backward from stage $n$ to stage 1. A procedure for constructing optimal stage policies $\mathbf{R}_j^*$ follows from Proposition 1, which we present below. To simplify exposition, we first make the following assumption:

**Assumption 1.** *When the expected value of stopping at stage $j$ equals the expected value of continuing to stage $j+1$ and behaving optimally thereafter, the optimal DM stops at $j$.*

Because the DM must accept the $n$th applicant, if reached, $V_n^* = n^{-1} \sum_{i=1}^{k} \sum_{a=1}^{n} \pi^i(a)$. And, for stages $j < n$, we have

$$V_j^* = Q(\mathbf{R}_j^*)\mathrm{E}(\pi_j|\mathbf{R}_j^*) + [1 - Q(\mathbf{R}_j^*)]V_{j+1}^*, \tag{13}$$

where $\mathrm{E}(\pi_j|\mathbf{R}_j^*) = |\mathbf{R}_j^*|^{-1} \sum_{\mathbf{r}\in\mathbf{R}_j^*} \mathrm{E}(\pi_j|\mathbf{r})$ is the expected payoff for stopping at stage $j$ under the optimal stage $j$ policy, and $Q(\mathbf{R}_j^*) = |\mathbf{R}_j^*|/k^j$ is the probability of stopping under the optimal stage $j$ policy. Given $V_n^*$, a method for constructing $\mathbf{R}^*$ is entailed by the following proposition:

**Proposition 1.** $\mathbf{r} \in \mathbf{R}_j^* \Leftrightarrow \mathrm{E}(\pi_j|\mathbf{r}) \geq V_{j+1}^*$.

Invoking Assumption 1, the proof of Proposition 1, presented in Bearden and Murphy [3], follows directly from the principle of optimality (Bellman [8]).

**Proposition 2.** $\mathbf{r} \in \mathbf{R}_j^* \Rightarrow \mathbf{r} \in \mathbf{R}_{j+1}^*$.

Proposition 2 follows from Corollary 2.1b in Mucci [38]. Stated simply, Proposition 2 tells us that if it is optimal to stop at stage $j$ when one observes $\mathbf{r}$, then it is optimal to stop when one observes $\mathbf{r}$ in the next stage; by induction, then, it is optimal to stop given $\mathbf{r}$ in *all* subsequent stages. This property allows us to represent the optimal policies rather compactly by specifying for each feasible $\mathbf{r}$ the smallest $j$ for which $\mathbf{r} \in \mathbf{R}^*$.

TABLE 3. MASP example problem.

| $a$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Payoff values | | | | | | |
| $\pi^1(a)$ | 6 | 5 | 4 | 3 | 2 | 1 |
| $\pi^2(a)$ | 5 | 4 | 3 | 2 | 0 | 0 |

| Applicant ($j$) | 1 | 2 | **3** | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Example applicant sequence | | | | | | |
| $a_j^1$ | 2 | 4 | **3** | 6 | 5 | 1 |
| $a_j^2$ | 5 | 2 | **1** | 3 | 6 | 4 |
| $r_j^1$ | 1 | 2 | **2** | 4 | 4 | 1 |
| $r_j^2$ | 1 | 1 | **1** | 3 | 5 | 4 |
| Optimal policy and payoffs | | | | | | |
| $V_{j+1}^*$ | 7.82 | 7.67 | **7.37** | 6.83 | 5.83 | — |
| $\mathrm{E}(\Pi_j \mid \mathbf{r}_j)$ | 5.83 | 5.73 | **7.55** | 2.93 | 1.83 | 8.00 |
| $\Pi_j$ | 5.00 | 7.00 | **9.00** | 4.00 | 2.00 | 8.00 |

## An Example of an MASP and the Application of Its Optimal Policy

Let us consider how the optimal policy would be applied in an MASP. Table 3 contains an example of an instance of an MASP with $n = 6$ and $k = 2$. The top panel contains the payoffs for each of the attributes. Absolute and relative ranks for each of the 6 applicants are shown in the center panel. We see that applicant 1 has absolute ranks of 2 and 5 on attributes 1 and 2, respectively; her relative ranks are, of course, 1 for both attributes. Applicant 2 has absolute ranks of 4 and 2, and therefore relative ranks of 2 and 1, for attributes 1 and 2, respectively, etc. The bottom panel displays the value of the optimal policy for each applicant position and the expected payoffs for selecting each applicant. Because applicant 3 is the first applicant for which $\mathrm{E}(\pi_j \mid \mathbf{r}_j) \geq V_{j+1}^*$, she is selected.

## Experimental Studies of the MASP

Bearden et al. [5] tested actual DMs on the MASP in two experiments. Using $n = 30$ (up to 30 applicants) and $k = 2$ (2 attributes), the experiments were identical in all regards except for their payoff schemes. Experiment 1 tested an MASP with symmetric payoffs, where $\pi^1(a) = \pi^2(a)$ for all $a$. Experiment 2 implemented an asymmetric payoff scheme in which attribute 1 was considerably more important (i.e., contributed more to the payoff) than attribute 2. The actual payoff schemes for both experiments are displayed in Table 4. In each condition, each subject played 100 random instances of the MASP. Payoffs were based on a single randomly selected trial; hence, those in Experiment 1 could earn up to \$50 for the one-hour session, whereas those in Experiment 2 could earn up to \$40. As with our

TABLE 4. MASP payoffs for Experiments 1 and 2 in Bearden et al [5].

| | | | | | | |
|---|---|---|---|---|---|---|
| Experiment 1 | | | | | | |
| $a$ | 1 | 2 | 3 | 4 | 5 | 6–30 |
| $\pi^1(a)$ | 25 | 12 | 8 | 4 | 2 | 0 |
| $\pi^2(a)$ | 25 | 12 | 8 | 4 | 2 | 0 |
| Experiment 2 | | | | | | |
| $a$ | 1 | 2 | 3 | 4 | 5 | 6–30 |
| $\pi^1(a)$ | 25 | 12 | 8 | 4 | 2 | 0 |
| $\pi^2(a)$ | 15 | 8 | 4 | 2 | 1 | 0 |

TABLE 5. Optimal policies for Experiments 1 and 2 from Bearden et al. [5].

| | $r_2=1$ | $r_2=2$ | $r_2=3$ | $r_2=4$ | $r_2=5$ | $r_2=6$ |
|---|---|---|---|---|---|---|
| Experiment 1 (symmetric) | | | | | | |
| $r_1=1$ | 7 | 12 | 14 | 15 | 16 | 16 |
| $r_1=2$ | 12 | 19 | 22 | 24 | 25 | 26 |
| $r_1=3$ | 14 | 22 | 25 | 27 | 27 | 28 |
| $r_1=4$ | 15 | 24 | 27 | 28 | 29 | 29 |
| $r_1=5$ | 16 | 25 | 27 | 29 | 30 | 30 |
| $r_1=6$ | 16 | 26 | 28 | 29 | 30 | 30 |
| Experiment 2 (asymmetric) | | | | | | |
| $r_1=1$ | 7 | 11 | 12 | 13 | 13 | 13 |
| $r_1=2$ | 14 | 19 | 22 | 23 | 24 | 24 |
| $r_1=3$ | 17 | 23 | 25 | 26 | 27 | 27 |
| $r_1=4$ | 19 | 25 | 27 | 28 | 29 | 29 |
| $r_1=5$ | 20 | 26 | 28 | 29 | 30 | 30 |
| $r_1=5$ | 21 | 27 | 29 | 30 | 30 | 30 |

*Notes.* For a given pair of relative ranks $(r_1, r_2)$, the table entry is the smallest $j$ for which applicants with relative ranks $(r_1, r_2)$ are selected. This representation is analogous to the cutoff representation $\mathbf{t}$ of optimal policies for the GSP.

previous experiments on the GSP, we used a hiring cover story, and instructed the subjects that the attributes were uncorrelated (in language they could easily understand).

Threshold representations of the optimal policies for the two MASPs examined in Experiments 1 and 2 are shown in Table 5. The cell entries for a pair of relative ranks $(r_1, r_2)$ correspond to the applicant position at which the optimal DM should begin to accept applicants with those relative ranks. For example, in the symmetric case (Experiment 1), the DM should begin accepting applicants with relative ranks of 1 on both attributes $(1, 1)$ at applicant position 7; applicants with a relative rank of 1 on one attribute and 2 on the other ($(1, 2)$ or $(2, 1)$) should be accepted starting at position 12; etc. For the asymmetric payoffs (Experiment 2), the DM should also begin accepting applicants with relative ranks of 1 on both attributes at position 7. Applicants with relative rank 1 on attribute 1 and relative rank 2 on attribute 2 should be accepted starting at position 11. In contrast, when the second attribute (the less important attribute) has relative rank 1 and the first (the more important attribute) has relative rank 2, the DM should wait until applicant position 14 to start accepting applicants with this profile.

In both conditions, we observed that the subjects examined fewer options on average than predicted by the optimal policy. For the MASP studied in Experiment 1, the expected length of search is 20.09 (by Equation (9)); the empirical average length of search was 15.89. For Experiment 2, the expected search length under the optimal is 19.45; the empirical average was 15.90. These findings are consistent with the previous results on the CSP and GSP that we reported above. Fortunately, the data from the MASP lend themselves to additional analyses that are quite informative.

Recall that Seale and Rapoport [55] tested several heuristics using the number of incorrect predictions ("violations" in their language) as their objective to minimize. To fit the cutoff rule, for example, they found the value of $t$ that minimized the number of incorrect stopping decisions for a given subject. To get a deeper understanding of the underlying decision policies subjects employ in the MASP, we used similar methods to estimate subjects' policies. We restricted ourselves to policies of the same form as the optimal policy, viz., to sets of thresholds for each stage $j$ that lead to decisions to stop the search. Given that each of the heuristics tested by Seale and Rapoport could take on at a maximum of 80 different values, they could use brute force search without any computational difficulty. Fitting MASP policies is more challenging due to the large feasible set of policies. To get around this

TABLE 6. Difference in location of the median empirical and the optimal cutoff locations for Experiments 1 and 2 in Bearden et al. [5].

| | $r_2=1$ | $r_2=2$ | $r_2=3$ | $r_2=4$ | $r_2=5$ | $r_2=6$ |
|---|---|---|---|---|---|---|
| Experiment 1 (symmetric) | | | | | | |
| $r_1=1$ | 0 | −4 | −5 | −3 | −3 | 2 |
| $r_1=2$ | −4 | −9 | −10 | −9 | −7 | −1 |
| $r_1=3$ | −5 | −10 | −9 | −8 | −5 | −2 |
| $r_1=4$ | −3 | −9 | −8 | −6 | −4 | 0 |
| $r_1=5$ | −3 | −7 | −5 | −4 | −1 | 0 |
| $r_1=6$ | 2 | −1 | −2 | 0 | 0 | 0 |
| Experiment 2 (asymmetric) | | | | | | |
| $r_1=1$ | 0 | −4 | 0 | −1 | 8 | 8 |
| $r_1=2$ | −4 | −6 | −5 | −3 | 1 | 1 |
| $r_1=3$ | −3 | −3 | −3 | −1 | −2 | 0 |
| $r_1=4$ | −2 | −1 | −2 | −3 | −1 | −1 |
| $r_1=5$ | 2 | 0 | 0 | 0 | 0 | 0 |
| $r_1=6$ | 4 | 3 | 1 | 0 | 0 | 0 |

*Notes.* For a given pair of relative ranks $(r_1, r_2)$, the table entry is the smallest $j$ for which applicants with relative ranks $(r_1, r_2)$ are selected. Negative differences indicate that cutoffs are shifted toward stopping too early.

problem, we used a heuristic procedure to find (probably) best-fitting policies for each subject. Specifically, we used Dueck and Scheuer's [18] *threshold-accepting* algorithm, which is an easy-to-implement cousin of simulated annealing. The details of our treatment of the optimization problem can be found in Bearden et al. [5].

Using the median cutoff (taken over subjects) of the estimated policies for each pair of relative ranks, we examined the difference between the optimal and empirical cutoffs to look for any systematic departures of the empirical policies from the optimal ones. Thus, when the difference is negative for a pair of relative ranks $(r_1, r_2)$, the empirical policy tends to stop earlier than the optimal one. The difference values for policies derived from both experiments are shown in Table 6. In most cases, for both experiments, we observe that the differences are negative, revealing a tendency (or bias) to accept applicants earlier than is optimal. More interesting, the bias is strongest for small relative rank pairs (e.g., $(1,2)$, $(2,2)$, $(2,3)$, etc.). There is also a bias to stop later on pairs for which one attribute guarantees 0 payoff (i.e., $r \geq 6$) and the other does not (i.e., $r < 6$). For example, in the symmetric case, the subjects tended to pass up applicants with relative ranks 1 and 6 even when stopping had a greater expectation under the optimal policy.

How should we interpret the policy results in Table 6? One possibility is that the subjects were using a policy consistent with Herbert Simon's notion of *satisficing* (Simon [58]). According to Simon, given the bounds on the capacities of actual agents—in contrast to ideal agents—we should not expect optimizing behavior. Instead, he suggested that agents might search for options that are "good enough," rather than those that are optimal. These good-enough options are the ones that *satisfice*, i.e., that meet the agent's aspirations on *all* (relevant) attributes. Because our subjects tended to stop quite early on applicants with small pairs of relative ranks and to avoid stopping on those with one good relative rank (e.g., 1) and one poor one (e.g., 16), we might suppose that their policy is of the satisficing sort: They want options that are sufficiently good on both attributes, and they do not want an option that is too poor on any single attribute as these do not satisfice.

One possible line of future inquiry is to look at optimal satisficing strategies in the MASP. Suppose the DM wants to guarantee (or maximize the probability) that she selects an applicant that is acceptable on all attributes. Perhaps problems with this objective more

closely match those that actual DMs face. Thus, optimal satisficing MASP policies may have both theoretical and practical importance.

The problems we have focused on up to now have involved selecting an option from a set of options presented sequentially. Next, we look at a class of problems in which the DM must assign each of the sequentially encountered options to open positions.

## Sequential Assignment

Derman et al. [16] considered the following problem: A DM observes a sequence of $n$ jobs for which there are $m$ machines available. Each job $j$ has a *value*, which is a random variable $X_j$ that takes on the value $x_j$. The cost of assigning job $j$ to machine $i$ is $c_i x_j$, and the DM's objective is to minimize her expected costs. In the simplest case, the jobs are sampled i.i.d. according to the distribution function $f(x)$, which is known by the DM.

Chun and Sumichrast [14] extended the problem presented by Derman et al. to scenarios in which the payoffs are determined only on the basis of the ranks of the jobs. Under this formulation, we need not assume a priori that the DM has full distributional information on the jobs. Chun and Sumichrast's sequential selection and assignment problem (SSAP) can be described as follows:

1. There are $n$ applicants for $m$ positions. Each applicant can be ranked in terms of quality with no ties. (For brevity and with no loss of generality (see Chun and Sumichrast [14]), we will only consider cases in which $n = m$.) Associated with each position $i$ is a cost $c_i$, where $c_1 \le c_2 \le \cdots \le c_m$.

2. The applicants are interviewed sequentially in a random order (with all $n!$ orderings occurring with equal probability).

3. For each applicant $j$ the DM can only ascertain the *relative rank* of the applicant, that is, how valuable the applicant is relative to the $j-1$ previously viewed applicants.

4. Each applicant must be assigned to an open position. Once assigned to a position $i$, the applicant cannot be reassigned to another position.

5. The total cost for assigning the $n$ applicants is $\sum_j a_j c_i$, where $a_j$ is the absolute rank of the $j$th applicant and $c_i$ is the cost of the position to which $j$ is assigned. The DM's objective is to minimize her total assignment costs.

## Computing Optimal Policies for the SSAP

Chun and Sumichrast [14] presented a procedure for determining optimal assignment policies for the SSAP. Interestingly, the optimal policy does not depend on the values of the position costs $c_i$; only the rank ordering of the costs matters. The optimal policy is expressed as sets of *critical relative ranks* $r^*_{i,k}$ for each stage $k = n - j + 1$ (where $k$ is simply the number of remaining to-be-interviewed applicants, including the current one). The critical ranks work as follows: Assign an applicant with relative rank $r_k$ at stage $k$ to the $i$th position if $r^*_{i-1,k} < r_k < r^*_{i,k}$. The critical ranks are computed by recursively solving

$$r^*_{i,k} = \begin{cases} 0 & \text{for } i = 0 \\ \dfrac{1}{n-k+3} \displaystyle\sum_{r_{k-1}}^{n-k+2} \min\{\max\{r_{k-1}, r^*_{i-1,k-1}t\}, r^*_{i,k-1}\} & \text{for } 1 \le i < k \\ \infty & \text{for } i = k. \end{cases} \tag{14}$$

Critical ranks for a problem with $n = 9$ are shown in Table 7. The first applicant ($j = 1$ or $k = 9$) should always be assigned to position 5 because she will always have a relative rank 1 which is between $r^*_{4,9}$ and $r^*_{5,9}$. If the second applicant's ($j = 2$ or $k = 8$) relative rank is 1, she should be assigned to position 3; otherwise, she should be assigned to position 6 (because her relative rank will therefore be 2).

TABLE 7. Critical relative ranks for the SSAP with $n = 9$.

| State | $k=9$ | $k=8$ | $k=7$ | $k=6$ | $k=5$ | $k=4$ | $k=3$ | $k=2$ | $k=1$ |
|---|---|---|---|---|---|---|---|---|---|
| $i=0$ | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| $i=1$ | 0.50 | 0.75 | 1.00 | 1.33 | 1.75 | 2.31 | 3.11 | 4.50 | $\infty$ |
| $i=2$ | 0.64 | 0.96 | 1.42 | 1.92 | 2.59 | 3.50 | 4.89 | $\infty$ | |
| $i=3$ | 0.75 | 1.25 | 1.78 | 2.50 | 3.41 | 4.69 | $\infty$ | | |
| $i=4$ | 0.92 | 1.50 | 2.22 | 3.08 | 4.25 | $\infty$ | | | |
| $i=5$ | 1.08 | 1.75 | 2.58 | 3.67 | $\infty$ | | | | |
| $i=6$ | 1.25 | 2.04 | 3.00 | $\infty$ | | | | | |
| $i=7$ | 1.36 | 2.25 | $\infty$ | | | | | | |
| $i=8$ | 1.50 | $\infty$ | | | | | | | |
| $i=9$ | $\infty$ | | | | | | | | |

TABLE 8. Absolute and relative ranks for nine patients.

| Patient number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| $a_j$ | 6 | 2 | 5 | 3 | 8 | 1 | 9 | 7 | 4 |
| $r_j$ | 1 | 1 | 2 | 2 | 5 | 1 | 7 | 6 | 4 |

## An Example of an SSAP and the Application of Its Optimal Policy

To fully illustrate this implementation, let us consider a problem with $n = 9$ in which the applicants are patients who must be assigned to hospital beds. The absolute and relative ranks of the 9 patients are displayed in Table 8. We assume that the absolute ranks correspond to the severity of the patient's malady, with lower ranks representing more serious cases. Each patient must be assigned to a bed in one of three hospitals that differ in terms of their costs. Table 9 shows the configuration of 9 beds that are distributed across three hospitals. Hospital A is a high-cost hospital that should be used for severely injured patients; Hospital B is for intermediate cases; and Hospital C is for the least-severe cases. Further, within each hospital the beds can be ranked in terms of costs. The most costly bed is Bed $A_1$ in Hospital A, and the least costly bed is Bed $C_4$ in Hospital C. For purposes of optimal assignment, all that matters is the cost of a particular bed $c_i$, where $c_1 \geq c_2 \geq \ldots \geq c_9$. Applying the assignment dictated by the critical ranks shown in Table 7, we determine the assignments shown in Table 10.

TABLE 9. Hospital bed positions for example. Bed numbers are shown in parentheses.

| Hospital A (high cost) | Hospital B (med. cost) | Hospital C (low cost) |
|---|---|---|
| Bed $A_1$ (1) | Bed $B_1$ (3) | Bed $C_1$ (6) |
| Bed $A_2$ (2) | Bed $B_2$ (4) | Bed $C_2$ (7) |
| — | Bed $B_3$ (5) | Bed $C_3$ (8) |
| — | — | Bed $C_4$ (9) |

TABLE 10. Optimal assignments for the example based on the ranks in Table 8 for the positions in Table 9.

| Hospital A (high cost) | Hospital B (med. cost) | Hospital C (low cost) |
|---|---|---|
| 1 | 2 | 5 |
| 4 | 3 | 7 |
| — | 6 | 9 |
| — | — | 8 |

## Experimental Studies of the SSAP

Bearden et al. [6] conducted three experiments on the SSAP. In each experiment, the subjects were asked to make triage decisions in a computer-controlled task. The SSAP was described to the subjects in simple language. They were asked to imagine that they had to assign patients to hospital beds after a mass casualty event, and were told that they would be paid based on the quality of their assignments. The costs within a hospital were constant, but hospitals differed from one another in their costs.

The three experiments differed in the number of to-be-assigned patients (12 in Experiment 1, and 24 in Experiments 2 and 3), and the number of bed positions in each of four hospitals. (For our $n = 9$ example, we used the configuration in Table 9; however, we could have used other configurations, such as 3 beds per hospital. The particular configurations used do not affect the optimal assignment, but we suspected they might make a psychological difference.) In each experiment, each subject played a total of 60 instances of the SSAP.

The SSAP is a relatively complex task and produces data that can be analyzed in a number of ways. Due to space considerations, we focus on what we consider to be the most interesting finding from these experiments. The original paper (Bearden et al. [6]) can be consulted for a fuller treatment

For a given instance of an SSAP, we can determine the god's-eye (or a priori) optimal assignment. This, however, is of little use in evaluating the assignments of an experimental subject. What we need to do for a given subject and a given problem instance is determine what is *conditionally optimal*, that is, what the subject should do—were she wishing to behave optimally—given what she has done up to that point. For example, the god's-eye optimal policy might dictate that patient $j$ be assigned to Hospital B, but if B is full by the time this patient is observed, then this assignment is impossible. What we need to determine for $j$ is how she should be assigned given the assignments that have previously been made (from 1 to $j-1$). What we report next is based on this notion of conditionally optimal assignment.

Using the experimental data, for each applicant position we can determine the probability that a patient will be assigned to each of the four hospitals under conditionally optimal assignment. Likewise, for each of the $n$ patient positions we can get the empirical probabilities (the proportions) of assignments to each of the four hospitals. When we do so, we find a systematic difference between the empirical and optimal assignment probabilities. Table 11 shows the proportion of times that the empirical probabilities exceeded the optimal ones.

Across the three experiments, the results are unambiguous: The subjects tended to assign patients to the intermediate hospitals with greater probability than was optimal. The results reveal a tendency to try to keep open beds in the extreme (really good (highest-cost) and really poor (lowest-cost)) hospitals. It seems that the subjects wished to reserve positions for the really injured and the not-so-injured patients, when doing so was suboptimal.

The triage task used in the experiments is quite artificial. However, it is not implausible to suppose that civilians who might be called on to make triage-type decisions in times of true mass casualty situations (say, a low-yield nuclear explosion in NYC) might demonstrate similar types of biases. (There are well-defined triage procedures for trained professionals,

Table 11. Proportion of times patients were assigned to each hospital with probability greater than predicted by the optimal assignment policy.

| | Hospital A | Hospital B | Hospital C | Hospital D |
|---|---|---|---|---|
| Experiment 1 | 0.50 | 0.92 | 0.92 | 0.25 |
| Experiment 2 | 0.25 | 0.92 | 0.92 | 0.20 |
| Experiment 3 | 0.31 | 0.93 | 0.78 | 0.20 |

*Notes.* Hospitals are ordered with respect to their costs. Hospital A is the highest-cost hospital and D is the lowest-cost one. Data taken from Bearden et al. [6].

but, unfortunately, one can easily imagine situations in which this relatively small group of individuals would be overwhelmed and might have to rely on civilians for help.)

## Conclusion

Our primary goal in writing this chapter is to stir the interests of OR researchers in the way *actual* DMs tend to solve the sort of problems faced by *ideal* agents in the platonic OR world. Not surprisingly, actual DMs do not always behave in accord with the dictates of optimal decision policies, which (quite often) must be determined by computationally intensive (at least with respect to the capacities of normal humans) procedures. *That* humans do not make optimal decisions is not particularly interesting. *How* it is that they do, in fact, make decisions should, we think, be of considerable interest to OR researchers. There are a number of reasons for this belief.

The first reason is that consideration of actual human cognition can lead to new research questions. Given what we know about the bounds of human cognition—or simply making reasonable assumptions regarding these bounds—we can formulate new sets of optimization problems. We could then ask: How well do humans perform relative to the appropriate *constrained optimal*? This is different from the standard question regarding optimality, which is: How well do humans perform relative to an *unconstrained optimal*? Determining the appropriate constraints on particular problems and studying the resulting constrained version can, we think, lead to interesting problems. See Shuford [57] for a nice example of this approach.

Bearden and Connolly [2], for example, compared the behavior of subjects in a full-information multiattribute optimal stopping task under two different conditions. In the *unconstrained* condition, subjects observed the actual values of the attributes of the encountered options. (A complete description of the problem faced by the subjects and the design of the experimental procedure can be found in the original paper. A procedure for computing optimal policies for the problem can be found in Lim et al. [35].) The *constrained* condition forced the subjects to use a policy of the same form as satisficing, as proposed by Simon [58]. Specifically, the subjects set aspiration levels or cutoffs for each attribute, and were then simply shown whether the attribute values were above (satisfactory) or below (unsatisfactory) their aspiration levels. Based on the procedures developed by Lim et al., Bearden and Connolly found optimal aspiration levels for the problem faced by the subjects; that is, they found optimal satisficing decision policies. One of the more interesting results from this study is that the subjects who were forced to satisfice (or to play the constrained problem) tended to set their aspiration levels too high, which caused them to accumulate higher-than-expected search costs and, consequently, to obtain lower-than-expected net earnings. Going beyond what can be concluded legitimately, one might say that even if people do tend to satisfice when they search, they do not do so optimally because they set their aspiration levels too high. Surprisingly, little has been said (in psychology or elsewhere) about the possibility of optimal satisficing. Research that examines the theoretical performance of relatively easy-to-implement heuristics, such as those studied by Stein et al. [60], can be of considerable importance to experimental psychologists and experimental economists, in additional to the typical readers of OR journals (see Gigerenzer et al. [22], particularly pp. 287–308, for some interesting examples of work along these lines).

Another reason why OR researchers should consider behavioral studies of decision making is that this work may impact more traditional OR problems. Perhaps by understanding the ways in which human behavior compares to optimality, one can design solutions to problems that are robust to human shortcomings. Whether one is determining optimal facility layouts or how fires should best be fought, understanding the properties of the actual agents who will be on the ground working in these facilities and making decisions when fires erupt must have some utility.

There are a number of areas in experimental psychology that we have not discussed in which OR methods play a valuable role (such as the study of problems involving optimal control; see (Busemeyer [11]) for an overview), and there are many more areas in which OR methods may serve a useful purpose. Unfortunately, much lip service is given to the need for "interdisciplinary research," but it seems that little is done. Very few of our colleagues in psychology have ever heard of INFORMS. Likewise, few of our colleagues in OR have heard of the Society for Judgment and Decision Making, the main research society for behavioral decision research. We can vouch for the fruitfulness of cross-discipline collaboration, and hope that in the future there is more cross-fertilization between experimental psychology and OR.

## Acknowledgments

The authors gratefully acknowledge financial support by Contract F49620-03-1-0377 from the AFOSR/MURI to the Department of Systems and Industrial Engineering and the Department of Management and Policy at the University of Arizona. The authors would also like to thank J. Cole Smith and Ryan O. Murphy for their feedback.

## References

[1] Y. Anzai. Cognitive control of real-time event driven systems. *Cognitive Science* 8:221–254, 1984.

[2] J. N. Bearden and T. Connolly. Satisficing in sequential search. *Organizational Behavior and Human Decision Processes*, 2005. Forthcoming.

[3] J. N. Bearden and R. O. Murphy. On generalized secretary problems. *Theory and Decision*, 2005. Forthcoming.

[4] J. N. Bearden and T. S. Wallsten. MINERVA-DM and subadditive frequency judgments. *Journal of Behavioral Decision Making* 26:349–363, 2004.

[5] J. N. Bearden, R. O. Murphy, and A. Rapoport. A multi-attribute extension of the secretary problem: Theory and experiments. *Journal of Mathematical Psychology*, 2004.

[6] J. N. Bearden, A. Rapoport, and R. O. Murphy. Assigning patients to hospitals in times of disaster: An experimental test of sequential selection and assignment. 2005. Forthcoming.

[7] J. N. Bearden, A. Rapoport, and R. O. Murphy. Sequential observation and selection with rank-dependent payoffs: An experimental test. *Management Science*, 2005. Forthcoming.

[8] R. Bellman.*Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[9] B. Brehmer. Dynamic decision making: Human control of complex systems. *Acta Psychologica* 81:211–241, 1992.

[10] B. Brehmer and R. Allard. Real-time dynamic decision making: Effects of task complexity and feedback delays. J. Rasumussen, B. Brehmer, and J. Leplat, eds. *Distributed Decision Making: Cognitive Models for Cooperative Work*. Wiley, Chinchester, UK, 1991.

[11] J. R. Busemeyer. Dynamic decision making. N. J. Smelser and P. B. Baltes, eds. *International Encyclopedia of the Social & Behavioral Sciences*. Elsevier, Oxford, UK, 3903–3908, 2001.

[12] C. Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton University Press, Princeton, NJ, 2003.

[13] Y. S. Chow, S. Moriguti, H. Robbins, and S. M. Samuels. Optimal selection based on relative rank (the "secretary problem"). *Israel Journal of Mathematics* 2:81–90, 1964.

[14] Y. H. Chun and R. T. Sumichrast. A rank-based approach to the sequential selection and assignment problem. *European Journal of Operational Research*. Forthcoming.

[15] R. M. Corbin. The secretary problem as a model of choice. *Journal of Mathematical Psychology* 21:1–29, 1980.

[16] C. Derman, G. J. Lieberman, and S. M. Ross. A sequential stochastic assignment problem. *Management Science* 18:349–355, 1972.

[17] A. Diederich. Sequential decision making. N. J. Smelser and P. B. Baltes, eds. *International Encyclopedia of the Social & Behavioral Sciences*. Elsevier, Oxford, UK, 13917–13922, 2001.

[18] G. Dueck and T. Scheuer. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics* 90:161–175, 1990.

[19] R. J. Ebert. Human control of a two-variable decision system. *Organizational Behavior and Human Performance* 7:237–264, 1972.

[20] T. S. Ferguson. Who solved the secretary problem? *Statistical Science* 4:282–296, 1989.

[21] T. S. Ferguson. Best-choice problems with dependent criteria. T. S. Ferguson and S. Samuels, eds. *Strategies for Sequential Search and Selection in Real Time*. American Mathematical Society, Providence, RI, 135–151, 1992.

[22] G. Gigerenzer, P. M. Todd, and ABC Group. *Simple Heuristics That Make Us Smart*. Oxford University Press, Oxford, England, 1999.

[23] J. Gilbert and F. Mosteller. Recognizing the maximum of a sequence. *Journal of the American Statistical Association* 61:35–73, 1966.

[24] A. V. Gnedin. A multicriteria problem of optimal stopping of a selection process. *Automation and Remote Control* 42:981–986, 1981.

[25] C. Hempel. *Philosophy of Natural Science*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[26] R. Hertwig and A. Ortmann. Experimental practices in economics: A methodological challenge for psychologists? *Behavioral and Brain Sciences* 23:383–403, 2001.

[27] J. D. Hey. Are optimal search rules reasonable? And vice versa? *Journal of Economic Behavior and Organization* 2:47–70, 1981.

[28] J. D. Hey. Search for rules of search. *Journal of Economic Behavior and Organization* 3:65–81, 1982.

[29] J. D. Hey. Still searching. *Journal of Economic Behavior and Organization* 8:137–144, 1987.

[30] J. M. C. Hutchinson and J. M. McNamara. Ways to test stochastic dynamic programming models empirically. *Animal Behavior* 59:656–676, 2000.

[31] R. J. Jagacinski and R. A. Miller. Describing the human operator's internal model of a dynamic system. *Human Factors* 20:425–433, 1978.

[32] J. H. Kerstholt and J. G. W. Raaijmakers. Decision making in dynamic task environments. R. Raynard, W. R. Crozier, and O. Svenson, eds. *Decision Making: Cognitive Models and Explanations*. Routledge, London, UK, 205–217, 1997.

[33] D. Kleinmuntz and J. Thomas. The value of action and inference in dynamic decision making. *Organizational Behavior and Human Decision Processes* 39:341–364, 1987.

[34] O. Larichev. Normative and descriptive aspects of decision making. T. Gal, T. Stewart, and T. Hanne, eds. *Multicritia Decision Making: Advances in MCDM Models, Algorithms, Theory and Applications.* Kluwer Academic Publishing, Boston, MA, 5.1–5.24, 1999.

[35] C. Lim, J. N. Bearden, and J. C. Smith. Sequential search with multi-attribute options. *Decision Analysis*, 2005. Forthcoming.

[36] D. V. Lindley. Dynamic programming and decision theory. *Applied Statistics* 10:39–51, 1961.

[37] S. Moriguti. Basic theory of selection of relative rank with cost. *Journal of Operations Research Society of Japan* 36:46–61, 1993.

[38] A. G. Mucci. Differential equations and optimal choice problems. *Annals of Statistics* 1:104–113, 1973.

[39] G. A. Parker and J. M. Smith. Optimality theory in evolutionary biology. *Nature* 348:27–33, 1990.

[40] E. L. Pressman and I. M. Sonin. The best choice problem for a random number of objects. *Theory of Probability and Its Applications* 17:657–668, 1972.

[41] W. V. Quine. *From a Logical Point of View*. Harvard University Press, Cambridge, MA, 1952.

[42] H. Raiffa. *Decision Analysis*. Addison-Wesley, Reading, MA, 1968.

[43] A. Rapoport. Sequential decision-making in a computer-controlled task. *Journal of Mathematical Psychology* 1:351–374, 1964.

[44] A. Rapoport. A study of human control in a stochastic multistage decision task. *Behavioral Science* 11:18–32, 1966.

[45] A. Rapoport. Dynamic programming models for multistage decision making. *Journal of Mathematical Psychology* 4:48–71, 1967.

[46] A. Rapoport. Research paradigms for studying dynamic decision behavior. H. Jungermann and G. De Zeeuw, eds. *Utility, Probability, and Human Decision Making.* Riedal, Dordrecht, Holland, 349–369, 1975.

[47] A. Rapoport and A. Tversky. Cost and accessibility of offers as determinants of optional stopping. *Psychonomic Science* 4:145–146, 1966.

[48] A. Rapoport and A. Tversky. Choice behavior in an optimal stopping task. *Organizational Behavior and Human Performance* 5:105–120, 1970.

[49] A. Rapoport and T. S. Wallsten. Individual decision behavior. *Annual Review of Psychology* 23:131–176, 1972.

[50] A. Rapoport, L. V. Jones, and J. P. Kahan. Gambling behavior in multiple-choice multistage betting games. *Journal of Mathematical Psychology* 7:12–36, 1970.

[51] M. Sakaguchi. Dynamic programming of some sequential sampling design. *Journal of Mathematical Analysis and Applications* 71:680–683, 1961.

[52] S. M. Samuels. Secretary problems. B. K. Ghosh and P. K. Sen, eds. *Handbook of Sequential Analysis*. Marcel Dekker, New York, 381–405, 1991.

[53] S. M. Samuels and B. Chotlos. A multiple criteria optimal selection problem. J. Van Ryzin, ed. *Adaptive Statistical Procedures and Related Topics: Proceedings of a Symposium in Honor of Herbert Robbins*. Brookhaven National Laboratory, Upton, NY, 1987.

[54] L. J. Savage. *The Foundations of Statistics*. Wiley, New York, 1954.

[55] D. A. Seale and A. Rapoport. Sequential decision making with relative ranks: An experimental investigation of the secretary problem. *Organizational Behavior and Human Decision Processes* 69:221–236, 1997.

[56] D. A. Seale and A. Rapoport. Optimal stopping behavior with relative ranks: The secretary problem with unknown population size. *Journal of Behavioral Decision Making* 13:391–411, 2000.

[57] E. H. Shuford, Jr. Some Bayesian learning processes. M. W. Shelly and G. L. Bryan, eds. *Human Judgment and Optimality*. Wiley, New York, 1964.

[58] H. A. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics* 69:99–118, 1955.

[59] P. Slovic, B. Fischoff, and S. Lichtenstein. Behavioral decision theory. *Annual Review of Psychology* 28:1–39, 1977.

[60] W. E. Stein, D. A. Seale, and A. Rapoport. Analysis of heuristic solutions to the best choice problem. *European Journal of Operational Research* 51:140–152, 2003.

[61] J. D. Sterman. Misperceptions of feedback in dynamic decision making. *Organizational Behavior and Human Decision Processes* 43:301–335, 1989.

[62] J. D. Sterman. Learning in and about complex systems. *System Dynamics and Review* 10:291–330, 1994.

[63] M. Toda. The design of the fungus eater: A model of human behavior in an unsophisticated environment. *Behavioral Science* 7:164–183, 1962.

[64] A. Tversky and D. Koehler. Support theory: A non-extensional representation of subjective probability. *Psychological Review* 101:547–567, 1994.

[65] B. van Fraassen. *The Scientific Image*. Clarendon Press, Oxford, England, 1980.

[66] J. von Neumann and O. Morgenstern. *The Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ, 1944.

[67] D. von Winterfeldt and W. Edwards. Flat maxima in linear optimization models. Technical report, Engineering Psychology Laboratory, University of Michigan, 1973.

[68] D. von Winterfeldt and W. Edwards. *Decision Analysis and Behavioral Research*. Cambridge University Press, Cambridge, England, 1986.

[69] M. C. K. Yang. Recognizing the maximum of a random sequence based on relative rank with backward solicitation. *Journal of Applied Probability* 11:504–512, 1974.

[70] A. J. Yeo and G. F. Yeo. Selecting satisfactory secretaries. *Australian Journal of Statistics* 36:185–198, 1994.

[71] R. Zwick, A. Rapoport, A. K. C. Lo, and A. V. Muthukrishnan. Consumer sequential search: Not enough of too much? *Marketing Science* 22:503–519, 2003.

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS | ISBN 1-877640-21-2
DOI 10.1287/educ.1053.0012

# Active Learning for Quantitative Courses

***James J. Cochran***
Department of Marketing and Analysis, College of Administration and Business,
Louisiana Tech University, Ruston, Louisiana 71272, jcochran@cab.latech.edu

**Abstract** This tutorial provides an overview of active learning for quantitative courses such as operations research, management science, production and operations management, probability, and statistics. The potential risks and benefits of active learning exercises are discussed, types of active learning exercises are reviewed, and several interesting and creative active learning exercises are described.

**Keywords** operations research education; active learning; cases; games

## Introduction

*Learning* is a naturally active process—true understanding is developed through student interaction with concepts, the instructor, and other students. This view is the basis of *active learning*, an approach that has been embraced throughout history by such notable and progressive educator/philosophers as Confucius, Socrates, and John Dewey. However, somewhere the lessons of these great educators have been largely abandoned at colleges and universities as lecturing has become the dominant pedagogical method; college students are now routinely subjected to long lectures in which only the transfer of information is accomplished. Indeed, one could consider lecturing to be a process by which ideas are transferred directly from the speaker's notes to the students' notes without interference from intermediary thought!

A well-delivered lecture has value; it is an extremely efficient means for demonstrating concepts or transferring information. While lecturing is an important and useful tool, it generally induces passivity of thought and does not engage students in the concepts and ideas being presented (Hartley and Cameron [46], McLeish [59]). Holding the attention of a student audience throughout a long lecture is extremely difficult (McKeachie [56]). Hartley and Davies [47] conclude that the typical student's level of attention increases during the first 10 minutes of lecture and diminishes rapidly thereafter.

Continuous dependence on lecturing creates a stagnant and lethargic learning environment; students are not compelled to question and discuss ideas, listen actively, think critically, form opinions, or experiment with concepts. This results in a superficial understanding of concepts and ideas that frustrates both the instructor and the student (Russell et al. [68]). Furthermore, student attention levels can vacillate wildly and somewhat independently as the lecture proceeds; the differences in the students' attention spans and the independence of the vacillations in their attention levels only further exacerbates the problem, creating ever-changing pockets of disinterest and indifference. Fortunately, several research studies (as summarized by Chilcoat [13]) suggest that an instructor can recapture students' attention by changing the classroom environment during a lecture. While a simple break from the lecture can be used to achieve this change, such a break can be even more effective if augmented by a well-crafted active learning exercise (Brown and Atkins [8], Campbell and Smith [9], Cashin [10]). Chickering and Gamson [12] summarize this point

very well:

> Learning is not a spectator sport. Students do not learn much just by sitting in class listening to teachers, memorizing prepackaged assignments, and spitting out answers. They must talk about what they are learning, write about it, relate it to past experiences, apply it to their daily lives. They must make what they learn part of themselves.

There are many reasons why active learning exercises effectively help students "make what they learn a part of themselves." Student participation in such activities allows them to deal with concepts on their own terms and work through ambiguities and misconceptions. Active learning can be a particularly effective means for redistributing the responsibility for learning, reducing the emphasis on lecturing, and increasing the students' role in their education. Active learning is also effective over a wider range of learning styles relative to other classroom approaches.

In quantitative methods courses, the problem of holding students' attention is aggravated by their apprehension of the course material. Many students are intimidated by quantitative material because they are poorly prepared, have little prior knowledge or understanding of quantitative methods, and are anxious about the required mathematics. Fortunately, active learning exercises can also be used to address this problem; student apprehension can be eased by allowing for classroom opportunities to work with and process quantitative concepts and methods. This results in a deeper understanding and appreciation of quantitative material, which in turn builds students' self-confidence and tolerance for the frustration of studying relatively difficult concepts, which leads back to a deeper understanding and appreciation of quantitative material (Garfield [41], Giraud [43], Gnanadesikan et al. [44], Keeler and Steinhorst [52], Magel [55], Rumsey [67]). This developmental cycle is the ultimate goal all instructors should have for their students—successful learning propagates successful learning!

This tutorial provides guidance for and encouragement to instructors who wish to introduce or increase the emphasis on active learning exercises in their courses. It also includes a discussion of some common, effective, and interesting active learning exercises. Although the focus is active learning in quantitative courses, most of the content of this tutorial is general and could be applied in any course.

## What *Is* Active Learning? What *Isn't* Active Learning?

*Active learning* is an elusive term, and several definitions are provided in the literature and on websites. McKinney [58] states that "Active learning refers to techniques where students do more than simply listen to a lecture. Students are DOING something including discovering, processing, and applying information." Bonwell and Eison [6] consider active learning to be "... instructional activities involving students in doing things and thinking about what they are doing." ActiveLearning Online [1] defines active learning as "... a multi-directional learning experience in which learning occurs teacher-to-student, student-to-teacher, and student-to-student."

While each of these definitions is unique, they have several commonalities. Each stresses (implicitly or explicitly) that

— students take a *participatory* role in their education,
— this participation should involve students in *higher-order* (not rote) thinking tasks, and
— *interaction* is a critical component.

The definition utilized in this tutorial synthesizes the following common traits.

*Active learning*: Instructional strategies that engage students through their interactive participation in exercises that involve them in higher-order thinking tasks such as analysis, synthesis, and critical evaluation of course material.

Given this definition, almost any learning exercise *could* be called active learning. This is not an incorrect characterization; it depends solely on how the exercise is conducted. Are

students engaged and taking a participatory role in their learning? Are they involved in higher-order thinking tasks? Is there substantial interaction? If the answers to these three questions are *yes*, the learning exercise in question is an active learning exercise.

Several other terms are frequently used interchangeably or in conjunction with active learning—these include the following.

*Experiential learning*: This is a process through which a learner develops knowledge and understanding from direct experience. Models of experiential learning are generally comprised of three distinct phases: an initial experience or problem situation, a reflective phase in which the learner reviews the experience looking for insights and understanding, and a testing phase in which the new insights and understanding are applied to a new problem situation or experience and assessed. Osland et al. [61] refer to these phases as *concrete experience*, *reflective observation and abstract conceptualization*, and *active experimentation*. Cooperative education, apprenticeships, internships, and projects are examples of experiential learning activities that can (and should) involve active learning at some level.

*Problem-based learning*: This approach is advocated by Barrows [2] and Stepien and Gallagher [74], among others. The students' knowledge, skills, and understanding are developed through their efforts to solve a real and messy problem. The problem is chosen by the instructor with specific pedagogical goals in mind; it should compel the student to acquire necessary or critical knowledge, enhance her/his problem-solving proficiency, and develop the ability to learn independently. The instructor acts as a facilitator and gives guidance to the students without interceding in their attempts to solve a problem. Students are ultimately evaluated primarily on how well they solve the problem.

*Cooperative or collaborative learning*: Small teams, each comprised of students with different backgrounds and/or levels of ability, work together to enhance their understanding of a subject. This technique aims to create an atmosphere of cooperation and achievement by making team members responsible for their teammates' understanding and learning.

Each of these approaches engages students and requires them to take a participatory role in their learning; each involves students in higher-order thinking tasks, and interaction is a vital component of each. Thus, they are specific types of active learning exercises.

## The Case for Active Learning

Several studies have demonstrated that active learning techniques can have a profound impact on student learning by enhancing both understanding and retention (Bonwell and Eison [6], George [42]). Many of these research studies have further demonstrated that active learning techniques can be both as effective as lectures in promoting mastery of content and superior to lectures in promoting the development of students' analytic and critical thinking skills (Felder and Brent [39], Johnson et al. [51]). Use of active learning exercises has also been shown to improve student attitudes toward studying and learning. Furthermore, some cognitive research (Claxton and Murrell [22], Felder [35, 36], Haller et al. [45]) has shown that a substantial portion of students have learning styles that are best served by pedagogical techniques other than lecturing (such as active learning).

Given the seemingly overwhelming evidence of the effectiveness of active learning, why is not its use more widespread? There are several possible and somewhat interrelated reasons.

### Barriers to Implementation of Active Learning—The Student Perspective

While many students will welcome a break from the classroom routine and be curious about what the instructor has planned, some may initially resist the shift to a stronger active learning orientation (Kelley et al. [53]). Felder and Brent [38], and Woods [77] draw provocative analogies between the phases psychologists associate with trauma/grief and the

stages a student may go through when required to assume more responsibility for her/his own learning:

*Shock and astonishment*—The student does not believe or will not accept that the instructor will deviate from exclusive reliance on lectures and assigned readings.

*Denial and refusal*—The student refuses to earnestly participate in the exercises and is certain the instructor will return to an exclusive reliance on traditional teaching methods.

*Defiance and withdrawal*—The student becomes more passive as s/he becomes convinced the instructor is committed to this new approach; s/he may even drop the class in the hope that a section taught in a more traditional style will be offered later.

*Capitulation and resignation*—Realizing that the instructor is committed to the new approach, the student is resigned to remain in the class and grudgingly begins to (somewhat earnestly) participate.

*Struggle and exploration*—The student observes other students succeeding with and enjoying the alternative approach and decides to reconsider her/his initial reaction.

*Return of/increase in confidence*—The student gains a better understanding of concepts through her/his participation and begins to feel positively about the alternative approach.

*Acceptance and success*—The student appreciates the value of the experience and fully supports the instructor's efforts to modify the classroom environment.

Not all students will experience all (or any) of these stages, but several may, and so the instructor must be prepared to deal with each stage. The keys to alleviating students' apprehension as they progress through these stages are (i) establishment of an open and sympathetic environment, (ii) maintenance of structure (that can be relaxed as students' apprehension diminishes), (iii) clear communication of the instructor's intentions and the goals of the exercise, and (iv) guidance that helps students understand and successfully complete the exercise.

## Barriers to Implementation of Active Learning—The Instructor Perspective

Students are not alone in their apprehension over altering an established learning environment. Instructors also have several legitimate reasons for feeling anxious about such a change (Felder and Brent [38]). Reasons for this anxiety include:

*Faculty mimic their instructors—because most faculty learned from instructors who used lectures as their primary mode, they have not been exposed to or seriously considered other options. Thus, lecturing is a large part of many faculty members' self-defined role as an instructor; it has become a self-propagating academic tradition.*

I believe there are three distinct potential transition points in an instructor's career:

- the instructor realizes that s/he can teach differently than those from whom s/he learned;
- the instructor realizes that s/he has a learning style that differs from most of her/his students; and
- the instructor realizes that s/he can recover from *almost* any minor misstep in the classroom.

Not all instructors experience these transitions, nor do they need to in order to be effective teachers. Furthermore, an instructor can achieve these realizations in any order. Obviously, an instructor who is considering the use of an alternative approach to teaching has reached at least one of these transition points and is looking beyond her/his self-defined role as an instructor.

*Faculty are apprehensive about trying something new or different in the classroom—many instructors who do not use active learning exercises are already successful in the classroom (some very much so). These instructors may view potential changes to their classroom approaches as risky. The alternate approach may not work as intended; students may not*

*earnestly cooperate/participate; the instructor may doubt her/his ability to use the alternate approach.*

There is certainly risk inherent in trying a new instructional approach such as active learning, but this risk can be mitigated. The instructor should make her/his first attempt at using active learning in a receptive class in which s/he has built a solid rapport with the students. S/he should think through the active learning exercises to be used. (What are the pedagogical goals? What could go wrong and how could the situation be handled? How long should the exercise take? How and where does this activity fit into the lecture? How will s/he re-engage the lecture at the conclusion of the exercise?) The instructor can start with a few simple active learning exercises (several potential exercises are provided later in this tutorial) and should explain to the students that s/he is going to give them a short break from the lecture and utilize a teaching approach that has been shown to improve student understanding and retention. The offer of a small inducement (i.e., credit for a missed quiz) for student participation may help. Students will generally appreciate the short break from the lecture and will be curious about what the instructor is going to do during this break. My experience also suggests that, if the instructor is well prepared, students will be very tolerant of small miscues in her/his early attempts at active learning.

*Instructors often fail to comprehend that* (i) *many* (*most?*) *of their students learn differently than they do and* (ii) *they* (*the instructors*) *are generally far more interested in their disciplines than are students. This intense interest usually dates back to the instructor's initial contact with the discipline; it is fueled by latent ability and fosters a high level of achievement. Students who eventually become college instructors generally experience less difficulty with the basics of her/his discipline, and as instructors they often expect* (*perhaps subconsciously*) *their students to understand the discipline with similar ease.*

Upon hearing such an explanation, an instructor may experience an epiphany—why expect students to learn the same way s/he learns? This sudden realization often results in a ready willingness by the instructor to incorporate active learning into her/his courses.

*Many instructors already feel they do not have adequate time to meet their responsibilities, and active learning exercises take time to prepare and time to execute. These instructors feel rushed and do not feel they have adequate time to achieve the expected coverage of the material. A few minutes taken from each of several class meetings will represent a substantial portion of the course. What is to be sacrificed to facilitate active learning exercises?*

Several studies have found that student comprehension and retention of difficult concepts is enhanced when the instructor pauses for a few minutes at 10-minute intervals throughout a lecture (Hartley and Davies [47]). Filling these short breaks with relevant and interesting active learning exercises can amplify these benefits (Brown and Atkins [8], Campbell and Smith [9], Cashin [10]). Even if the use of active learning exercises does result in a small loss of lecture time, is it not preferable to cover slightly less material at a higher level of student comprehension and retention?

*Administrations generally provide little support for the development of alternative instructional approaches. The tenure process often encourages research productivity at the expense of all else. Furthermore, instructors who are considering the use of active learning exercises many fear their colleagues' scorn and ridicule* (*s/he plays games in her/his classroom!*).

If used properly, active learning exercises can enhance student comprehension and retention at a very low cost (especially once the instructor has established several exercises that s/he likes to use). This does not require an immediate and absolute change—the instructor can integrate active learning exercises into her/his courses gradually, minimizing both her/his risk and the cost. Administrators and colleagues certainly cannot argue with a tremendous classroom enhancement that ultimately takes little time away from a faculty member's research efforts.

Thus, each of these barriers can be overcome with a small commitment and a little creativity!

### Mischaracterization of the Nature of the Instructor/Student Relationship

While it is true that a student is an end user of her/his education, defining and dealing with a student as a *traditional* customer distorts and demeans the nature of the instructor/student relationship. A student who thinks of her/himself in this manner is more likely to expect to be "served" her/his education. Unfortunately, this type of instructor-student relationship does not result in effective education—the student must be engaged to learn.

The coproduction model proposed by Whitaker [76] for service industries and later applied by Chappell [11] to public education provides a far more appropriate paradigm for the relationship between students and instructors. In this model:

— the end user/customer (i.e., the student) has a physical or virtual presence and supplies some labor;

— information flow is facilitated through interaction with the service provider (i.e., the instructor) *and* other customers (i.e., classmates); and

— the end user makes indispensable intellectual contributions to the process.

The goal of this model is *customer efficiency*, i.e., customers who use *less* of the service provider's resources while accomplishing *more* for themselves. Factors that influence customer efficiency include:

— internal factors such as the individual customer's ability, preferences, background, etc.;

— external factors such as the design of the service delivery process, distribution channels for service delivery, etc.; and

— the customer peer group.

The insights into the role of service providers (i.e., instructors) is clear—they are primarily responsible for facilitating improvement of customer (i.e., student) efficiency through the design of service delivery process and service delivery distribution channels. Instructors can use active learning exercises to enhance the service delivery process and ultimately improve student efficiency. Furthermore, an instructor who gives assignments to be completed by student teams also has some control over the customer peer group in her/his classroom. This control can be used to ultimately improve student efficiency.

## Integrating Active Learning into the Quantitative Courses

Once an instructor has decided to use active learning exercises in her/his course, s/he is faced with the tasks of finding or designing active learning exercises and integrating these exercises into the course. Careful thought must be undertaken at this stage to ensure the exercises will be effective and constructive. Whether the instructor is designing a new active learning exercise or using a preexisting exercise, s/he should consider the following.

*The audience*: The students' prior training, education, and experience are critical considerations. The active learning exercise must require students to consider concepts at an appropriate level. If the exercise is too simple, students will quickly lose interest; an overly demanding or complex exercise will frustrate students. In either of these cases, students will conclude that the instructor is wasting their time! While it is certainly important to consider the students' formal education and prior exposure to relevant concepts, it is also important to consider their prior classroom experiences. For example, extra care must be taken when assigning cases to students who have no prior case-analysis experience.

*His/Her personality*: The instructor must stay within her/himself. An instructor with an outgoing personality will likely be comfortable with and enjoy exercises that encourage instructor-student interaction, while an introverted instructor will likely be more at ease with exercises that stress student-student interaction.

*Course material*: Once the instructor has decided what concept(s) to emphasize with active learning, s/he must design or select exercises that are appropriate. The exercise must

be suitable for the concept(s) the instructor wants to emphasize, and it must be executed at a point in the class meeting when it will be most effective.

*Pedagogical objectives*: The instructor must decide if s/he wants to use the active learning exercises proactively to encourage students to consider a concept prior to a class discussion, reactively to encourage students to give deeper consideration to concepts that have already been discussed in class, or intra-actively to help students understand a concept in lieu of a classroom discussion. Depending on when it is executed, an active learning exercise can be used to help students gain deeper insight into an important concept, integrate several concepts, or perceive similarities and differences in several concepts.

*The classroom space*: Some classrooms are designed in a manner that allows students to easily work in groups of four or five; some have tables in the front of the room; some have chairs and desks that are anchored to the floor. Classroom characteristics such as these can limit or enhance an active learning exercise.

*Available equipment and materials*: Again, this will vary across schools (and even across classrooms at a particular school). For example, I ask a student to collect responses and build a bar chart on the chalk board as the class plays the classroom version of *Who Wants To Be A Millionaire*. If I had access to a classroom with PDAs, I would certainly use them to electronically collect and summarize student responses.

*Available class time*: While most active learning exercises can be fully executed in less than 10 minutes, a few can take substantially longer. The instructor must weigh the benefits (in terms of enhanced student understanding and retention) and costs (in terms of class time) when designing or selecting an active learning exercise.

An instructor who is new to the active learning approach should consider using a simple preexisting exercise; several active learning exercises are presented in INFORM-*ED* (INFORMS forum for education—http://education.forum.informs.org/) sponsored sessions at INFORMS annual conferences, and several more are published in *INFORMS Transactions on Education* (http://ite.pubs.informs.org/). These exercises can often be used off the shelf or can easily be retrofit to meet the individual instructor's needs.

The instructor should prepare students for the active learning exercise by announcing what s/he intends to do in advance (perhaps in the class meeting prior to the use of the exercise). The instructor should explain clearly that s/he is going to use the exercise to give the class a short break from the lecture and reinforce one or more important concepts. Students should be made to understand that this is a serious assignment; a small inducement (i.e., credit for a missed quiz) may help. Finally, the instructor should acknowledge that this deviation from the usual classroom approach may cause students to feel somewhat uneasy (the instructor is not the only person taking a risk!)—this will help establish an intellectually and emotionally supportive environment in which the students feel more free to participate.

The active learning exercise can be integrated into the class in several ways. Perhaps the least risky approach is to modify the lecture (Penner [63]). The simplest modification is the *pause procedure*; simply pause for a few minutes to allow students to consolidate their notes at a few points during a lecture. Research (Ruhl et al. [65]) indicates that students learn significantly more information in classes that use this approach.

Another simple and effective way to engage students during a lecture is to follow a short demonstration or writing assignment with a brief discussion period; the discussion can involve the entire class, or the class can be broken into small discussion groups. Similarly, in the *feedback lecture approach* a lecture can be split into shorter segments by brief breaks during which student groups discuss what has been covered. The *guided lecture* is somewhat more radical; the instructor gives a relatively short lecture (perhaps 30 minutes) during which the students do not take notes. After the lecture, students are given five minutes to write what they remember from the lecture. In the remaining class time, students meet in small groups to clarify and elaborate on the lecture (as the instructor moves from one group to another, answering questions, and providing some guidance).

Note that these techniques achieve increased interaction (and the resulting enhancements to understanding and retention) through in-class discussion in reaction to the material covered in a lecture. Alternatively, proactive techniques can be used to encourage students to consider concepts prior to a lecture. In the *jigsaw approach* student teams are formed in advance; each team member is made responsible for reviewing a specific part of a topic or concept that will be discussed in the next lecture. At the beginning of the next class meeting, the students are divided into groups according to the specific part of the topic they were assigned. After these groups each discuss their specific findings, the students return to their original teams and integrate their results. Although not traditionally thought of in this manner, even *cases* can be used in a similar fashion; the case can be assigned prior to discussion of some or all of the relevant concepts, and the instructor can present student teams with a series of questions or issues to be considered when analyzing the case. Again, each of these questions or issues is assigned to a specific team member; on the day the case is to be discussed, a jigsaw-type approach can be used to help student groups reach some consensus. This is followed by a full class discussion that enables the instructor to (i) clear up any remaining misunderstandings, (ii) address any of the students' persistent concerns, and (iii) discuss issues that the student groups have overlooked. Even *group projects* can form the basis of an active learning classroom environment—as the project advances, each group can present regular progress reports in class, and the student groups can provide mutual support by asking questions and suggesting alternate approaches.

Although these active learning exercises are relatively simple, they can be extremely effective. As McKeachie et al. [57] found, discussion/interaction is more effective than lecturing for promoting understanding and retention of information, developing critical thinking skills, encouraging students to apply concepts and ideas in different circumstances, and motivating students toward further learning. Of course, there are much more elaborate active learning approaches that can be used once the instructor is comfortable using active learning exercises. These exercises are the topic of the next section of this tutorial.

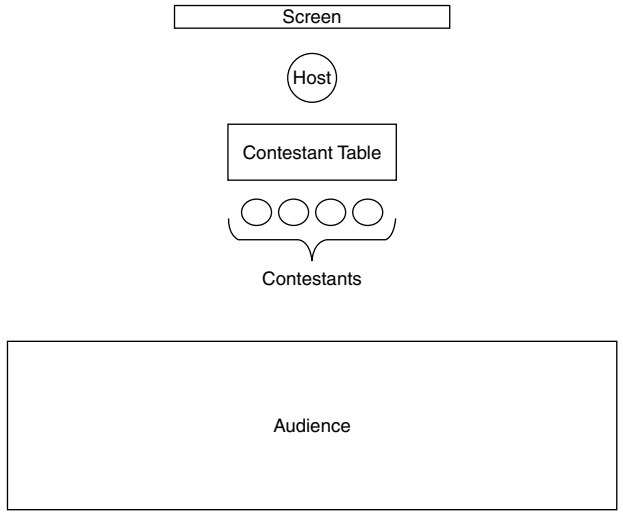
## Some Active Learning Activities

Each of the previously discussed active learning approaches has several common traits: They engage students through their interactive participation in exercises; they involve students in higher-order thinking tasks such as analysis, synthesis, and evaluation of course material; and they are all built around a break from a traditional lecture. There are more elaborate activities, such as games and cases, that can be used in the quantitative classroom. These active learning exercises have the potential for extremely high return; thus, as the instructor becomes more comfortable with active learning exercises, s/he may wish to integrate some of these more elaborate exercises into the classroom.

## Games

Games can be an important tool in the quantitative instructor's arsenal (Chlond and Toase [19], Cochran [28], Sniedovich [71]). Several authors have reported successful use of classroom games, including Biesterfeld [4], Bosch [7], Chlond [14, 15, 16, 17, 18], Chlond et al. [21], Chlond and Toase [20], Cochran [27, 30], Eckert [32], Erkut [33], Feldman and Morgan [40], Letavec and Ruggiero [54], Myers [60], Pendegraft [62], Rump [66], Sniedovich [71, 72, 73], Trick [75], and Yeomans [78]. Each has found games (and puzzles and paradoxes) to be very effective methods for developing their students' understanding of complex concepts. Although these games do not expose students to "realistic" problems, they are complex and require the students to engage in higher-order thinking about the concept(s) in question, which in turn does prepare students to attack practical problems they may encounter in later studies or in their career. An overview of a few classroom games follows.

FIGURE 1. Classroom layout for *Who Wants To Be A Millionaire-The Classroom Edition*.




## *Who Wants To Be A Millionaire-The Classroom Edition (WWTBAM-TCE)*

*WWTBAM-TCE* (Cochran [27]) is a Microsoft PowerPoint-based classroom version of the internationally popular television game show *Who Wants To Be A Millionaire*. At an appropriate breaking point in a lecture, a student or team of students is invited to participate as "contestants" in the game. All other students in the class are instructed to put away their note-taking materials for the duration of the game (this enhances the perception that the class is taking a break). The contestants are seated in front of the class facing the screen with their backs to the audience (so that they cannot receive any unsolicited assistance), as depicted in Figure 1.

The contestants are presented with a question and four potential responses that appear on a screen image similar to that which is used on the television show *Who Wants To Be A Millionaire*; music from the show also plays at appropriate junctures of the game. The team of student contestants must collaborate, reach a consensus, and give a single response to the question. Ultimately, they may respond in one of three ways:

— Refuse to answer the question, quit the game, and retain all credit earned to that point. The team of student contestants is initially awarded three-fourths of a quiz credit for participating and receives one-fourth of a quiz credit for each correct answer, so a maximum of two quiz credits are earned if all four questions are correctly answered;

— Answer the question and continue to play if correct or forfeit their right to continue playing and lose all credit earned if incorrect. However, once the team of student contestants has earned one quiz credit (answered the first question correctly) they cannot lose that credit; or

— Use a lifeline (*Ask a Friend, Poll the Class*, or *50/50*). Each team of student contestants is only allowed a single use of one lifeline.

If the team of student contestants answers a question correctly, they receive the quarter quiz credit and are permitted to proceed to the next question. Because the game requires use of multiple choice questions with short answers, a four-question version (which can be executed in less than 10 minutes) gives students a short break from the lecture and provides a review of a few concepts while maintaining some class momentum. *WWTBAM-TCE* can be downloaded from *INFORMS Transactions on Education* at http://ite.pubs.informs.org/Vol1No3/Cochran/Cochran.php. This game takes approxi-

mately 10 minutes to prepare using *WWTBAM-TCE* as a template, and prepared sets of questions can certainly be recycled over several academic terms.

### *Jeopardy!*

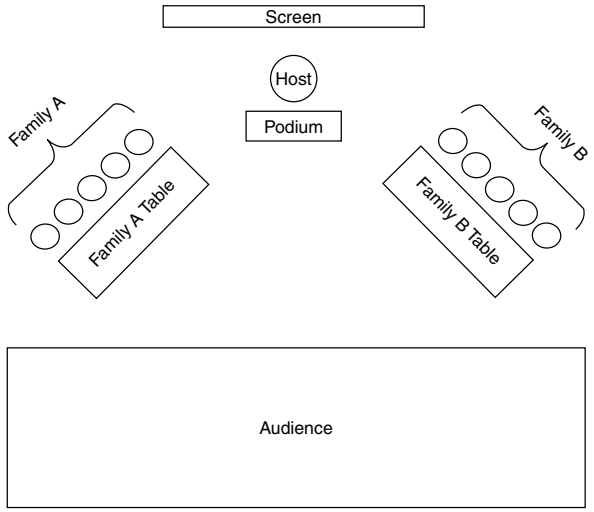
*Jeopardy!*, the television game show in which contestants compete for cash by providing the correct *question* when given an *answer* (responses not given in the form of a question are considered incorrect!), is the basis of a classroom game developed by Benek-Rivera and Mathews [3]. The authors divide a class into teams of three to five students and draw the *Jeopardy!* game board on the blackboard (using four category titles and five ascending point values for items in each category). One team is allowed to select the first *Jeopardy!* category and point value; when an item (a combination of category and point value) is selected, the instructor/host erases it from the board to indicate that the item is no longer available for play. S/he then reads the corresponding answer from an index card or script to the class. The first student to indicate s/he wishes to respond for his or her team is called on to play that item; that student is responsible for responding for his or her team. If the student gives the correct question within 10 seconds, her/his team is awarded the points associated with the question. If the student gives an incorrect question or fails to respond within 10 seconds, her/his team is penalized the points associated with the answer; at this juncture, any player on one of the other teams is allowed to indicate s/he wishes to respond for his or her team. The authors suggest a limit of two "attempts" per item to allow time for the instructor to provide further explanations (this is particularly important when an item is not answered correctly).

As with the television version of *Jeopardy!*, answers with higher values are considered more difficult, and the team that most recently provided the correct question controls the board and is allowed to choose the next item to play. The authors also suggest ways to incorporate the hidden "Daily Doubles" and the "Final Jeopardy" round. They estimate that the classroom *Jeopardy!* game takes 30 to 40 minutes to prepare and 30 to 60 minutes to execute, and they use the game for examination review sessions. Again, prepared sets of answers can be recycled over several academic terms. Linda Leon, Zbigniew Przasnyky, and Kala Seal of Loyola Marymount University have also developed a very good PowerPoint version of this game that works in a manner similar to *WWTBAM-TCE* (Seal [70]).

### *Family Feud-The Classroom Edition*

In yet another electronic version of a television game show, *Family Feud* has been adapted for the classroom in a PowerPoint format with images and music from the television show. The actual premise of the *Family Feud* game show is simple—a survey is administered and responses to a question are collected. The responses to this question are then listed on the screen in order by their frequency and hidden from view of the contestants. The game is initiated when the first contestant from each the two families approaches the podium. The host reveals and reads a question, then waits for the two contestants standing at the podium to respond. The contestant who gives the answer associated with the greater frequency of survey responses wins control of the board for her/his family; they can then choose to play or pass. If they choose to play, individual family members (in order) continue to guess the survey responses (and accumulate points equal to the number of survey responses associated with their guess) until they either guess all responses on the list (and are awarded the points they have accumulated) or give three incorrect guesses. If they give three incorrect guesses, the opposing family is given one opportunity to guess at remaining unrevealed survey responses: If they correctly guess one of the remaining survey responses, they "steal" the points accumulated on this round by their opponent; if they fail to correctly guess one of the remaining unrevealed survey responses, their opponent retains the points they have accumulated in this round. Rounds are played until one family accumulates a predetermined

FIGURE 2. Classroom layout for *Family Feud-The Classroom Edition.*




number of points, is declared the winner, and is allowed to move to the "Lightning Round" (a timed, solo version of the game played by two members of the family for higher stakes).

The classroom version is similar. The instructor initially selects two sets (or "families") of five students, and each family stands behind a table that is angled toward the audience in a fashion that still allows the family members to see the overhead screen (see Figure 2). A podium is situated between the two tables.

Student responses to open-ended quiz questions from previous sections of the course are used. Two families (sets of five students) are selected from the audience. Instead of playing until one family reaches a predetermined point total, the class plays up to three rounds (each round takes approximately three to five minutes). If a team correctly guesses all of the survey answers or successfully steals, they must still (as a team) identify the correct answer to win the round—even if the correct answer has not been uncovered! If they do not identify the correct answer, the points accumulated during the round are not awarded to either team. A team also must win two rounds to win the game; if neither team wins at least two of the three rounds, the game is declared a stalemate. Also note that no lightning round is played.

Because students are attempting to guess the most common answers, *Family Feud-The Classroom Edition* (i) allows students who do not know the correct answer to more fully participate and (ii) forces students to identify alternative (albeit incorrect) answers to questions. The nature of the game also allows for student consideration of answers of varying degrees of correctness. The instructor can and should briefly discuss the question and various answers at the end of each round. Because the questions and answers are taken from real surveys that have been administered to students in prior sections, students generally pay very close attention. Once the surveys have been administered and the data collected, preparation of this game takes 10 to 15 minutes, and prepared sets of questions can be recycled over several academic terms.

### *Trump or Monkey?*

This game is a PowerPoint version of a routine often used on the *Late Night with David Letterman* television talk show. *Trump or Monkey?* is used to encourage discussion and insight into the value of random sampling. As Peter Gabriel's 1982 hit song *Shock the Monkey* plays in the background, students are informed that they are about to see portions of three images (two of monkeys and one of Donald Trump). From this information, they

are to guess which image is of Donald Trump. The top 20% (approximately) of the images are revealed, the class is polled, and the responses are tallied and summarized. At this point, the instructor uncovers the remainder of the image that received the most votes (and it is most often not the Donald Trump image). The instructor then shows the students the same images with a randomly selected 20% of each exposed and asks them if the new information is more insightful than the originally exposed portions of the images. Of course, it is—the random samples expose parts of noses, eyes, skin (or fur), ears, etc., and the value of a random sample over a convenience sample is established. Later in the course the information can be used further to support discussions on the binomial and multinomial distributions as well as estimation and inference for proportions.

## Cases

Use of cases in quantitative methods courses can be problematic (particularly at the introductory level). While many disciplines use cases exclusively in certain courses (particularly in MBA programs), students in quantitative methods courses usually rely heavily on the instructor's explanations (i.e., lectures) and would be extremely uncomfortable if the course were entirely case-based (there are, of course, several notable exceptions). This is aggravated by differences in the nature of quantitative cases and cases from other disciplines; cases from other disciplines are generally managerially oriented, while the quantitative courses are problem oriented. Finally, there are relatively few quantitative cases available.

Despite these obstacles, cases can be an extremely effective active learning tool in quantitative methods courses—even at the undergraduate introductory level (Cliff and Curtin [23] and Cochran [26]). As a part of an active learning environment, they can be used to supplement lectures, provide opportunity for students to apply methods and integrate concepts, and review topics prior to examinations (Feinstein and Veenendall [34], Richards et al. [64], Herreid [48, 49]).

In the undergraduate introductory courses I teach, I generally give two interim exams and a final exam; written two-page analyses of two cases are due the class period prior to each exam. The class meeting immediately prior to each exam is devoted to discussion of the cases that are due, and students are instructed to come to class prepared to participate in the discussion; this forms the basis of a very revealing, effective, and somewhat unusual review session. I select and assign pairs of cases that reinforce different concepts, and the students are wholly responsible for determining how to complete their case analyses.

I have used this approach in classes with as many as 300 students; while managing the discussion is more challenging with large audiences, it can be done effectively. A powerful voice or a cordless microphone will enable the instructor to move around the classroom and encourage participation. Grading can also be a consideration—cases analyses can take a long time to grade. However, the task can be mitigated by

— putting students into groups and requiring a single analysis of a case from each group;
— requiring a subset of the students or groups to submit each case analysis; and/or
— grading the cases on a pass/fail basis (I was surprised that the quality of cases improved when he adopted this case-grading policy—apparently the threat of receiving no credit is a tremendous motivator).

I also consider the audience and its experience with cases as I grade cases; my expectations for seniors and master's-level students are much higher than my expectations for students in introductory undergraduate courses, and my expectations increase as the academic term progresses.

## Case Discussion Formats

Once an instructor has committed to using cases and has chosen appropriate cases, s/he must consider potential class discussion formats. Because the class will eventually be expected,

as a group, to reach some conclusion (with some possible dissention), consideration of various group approaches to strategic decision making is appropriate. Schweiger et al. [69] provide, compare, and contrast three common group approaches to strategic decision making (consensus, devil's advocacy, and dialectical inquiry) that are described briefly below:

*Consensus decision making*—The instructor works with the student groups throughout the discussion to help students reach a mutually agreeable conclusion or solution to the case. This type of discussion is easiest for the instructor to implement and maintain. Because the risk of being incorrect is spread throughout the class, it is also least intimidating for the students. Thus, the consensus decision-making discussion is best suited to instructors and/or students who are relatively inexperienced with the case method.

*Devil's advocacy*—An individual or subgroup is chosen to critique a proposed course of action and identify problems to consider before a final decision on the case problem is reached. Instructors who use this approach will often assign presentation of the case to one (or a few) student groups and appoint the remaining student groups to act as the devil's advocates. This approach is more demanding on the instructor and more intimidating to students, and so is better suited to instructors and/or students who have some experience with the case method.

*Dialectical inquiry*—The class approaches a decision from two (or more) opposing perspectives and debates conflicting views. When two (or more) student groups take opposing perspectives on a case, this approach can generate great creativity and insight. Of course, the instructor cannot rely on the emergence of two opposing perspectives during a case discussion; if s/he wants to establish this type of discussion, s/he must be prepared to take on the role of devil's advocate. This discussion approach is the most demanding on the instructor and most intimidating to students, and so is best suited to instructors and/or students who are relatively experienced with the case method.

Of course, these group discussion formats are not mutually exclusive—it is easily conceivable that an instructor may wish to interject or encourage some of each behavior in a single case discussion. Nor are they collectively exhaustive—several other models for group discussion and decision making exist and are worthy of consideration.

## Facilitating a Case Discussion

These are several approaches to facilitating case discussions. Some common or interesting techniques include:

*Standard (give-and-take) discussion*—All students or student groups complete analyses and come to class prepared to discuss the case. The discussion spans all students or student groups.

*Presentation-centered discussion*—A student group is given responsibility for presenting and defending its case analysis in front of its classmates, providing the basis for the case discussion. Student groups rotate primary responsibility for presenting results (and providing the basis for the discussion).

*Anchored instruction (or role playing)*—Anchored instruction puts students into the context of a case. Each member of one selected student group assumes the role of a character in the case, and these students act out the case scenario for their classmates. The class then breaks into their student groups and discusses the case presentation. This is followed by a second performance with a few critical differences—a member of the audience may stop the performance at any time to

— ask any character a question or

— take the place of any of the student "actors" and assume the role of her/his character.

I was somewhat skeptical when I first heard of this approach. However, after participating (both as an actor and an audience member), I was amazed at how much insight I quickly developed into the potential motivations of the various characters in the case.

*Box-and-one method*—This approach is named for the special defense devised by college basketball coach Jim Valvano. Students are divided into teams of four or five; all teams are assigned the case and told to prepare to present the case for the next class meeting. The instructor further explains that the next class meeting will be somewhat unusual, and the students will participate in a case-related exercise that will enhance their abilities to "think on their feet" and react quickly to ideas presented by competing teams.

The students arrive at the next class meeting to find that separate projection stations (each with projection equipment and a screen) are set up in each corner of the room. Work stations with tables, chairs, blank transparencies, and pens are also set up and labeled for each student team, and a lone stool (hopefully one that swivels) sits in the middle of the room.

The instructor enters the room, sits on the stool in the center of the classroom, and instructs the students to begin presenting. At first, the students are confused. Who should present? Which team should speak? In what order? The instructor encourages any team to go to any projection station and start the discussion by explaining the case scenario, and also encourages the other teams to go to the other projection stations when they have additional points to make or take issue with something presented by another team. Within a few minutes, the projection stations are occupied by teams debating case issues and approaches, while the remaining teams are waiting for a turn at the projection stations. Teams race back to their work stations to create new slides (either electronically or manually, depending on the available projection equipment), using Excel to make calculations and create slides and tables. Order and energy emerge out of the initial anarchy. As with all case discussions, the instructor monitors (and sometime redirects) the discussion and ensures that all teams have ample opportunity to participate; these tasks are more demanding when using the box-and-one approach to facilitating case discussions.

The instructor must assume responsibility for maintaining decorum in the classroom and insisting that students treat each other with respect throughout the case discussion. S/he must also make decisions about how much time to spend investigating particular points raised by a student group and when to direct a wayward discussion back onto a more desirable path. Finally, the instructor must work hard to ensure the students do not feel ambushed at the onset of the first execution of this approach; s/he must again assure students that the approach has been adopted to enhance their abilities to think on their feet and react quickly to ideas presented by competing teams (critical business meeting skills that are difficult to develop or practice).

## Example Cases

The three cases described in this section demonstrate how effective quantitative cases can be found in very accessible and mundane places. *You've Got Cocaine in Your Wallet* is based on a newspaper article, *Bowie Kuhn's Worst Nightmare* is drawn from my personal interest (baseball), and *Dynamo Automobile Alarms* was derived from a consulting project in which I participated.

*You've Got Cocaine in Your Wallet.* The *You've Got Cocaine in Your Wallet* case (Cochran [25]) is based on a newspaper article entitled "You're Carrying Cocaine in Your Wallet—Drugs in Britain: Special Report" (Dodd [31]) that first appeared in October of 1999 in *The Guardian*, a daily newspaper published in Great Britain. It eventually was sent out by a news service, and was first read by the author in the *Dayton Daily News* (a daily central Ohio newspaper) later that month. The article opens with the provocative claim that Prime Minister Tony Blair, Parliamentarian Jack Straw, and Sir Paul Condon (Britain's top policeman at the time) are almost certainly carrying cocaine-tainted bills in their wallets. The article then proceeds to explain that the BBC Newsroom South-East commissioned a study in which a sample of 500 British bank notes (supplied by the Bank of England's returned note centre) was tested for the presence of cocaine and ecstasy. After explaining

that traces of a drug can be left on bills either by direct contact with the drug or transference through skin oils of users, the article reports some findings, including the following.

— More than 99% of the notes tested were contaminated by cocaine.

— Of the 500 notes, 4 tested were "clean."

— Nearly 1 in 20 tested notes were found to have high levels of cocaine (suggesting they had been in direct contact with the illegal drug).

— Of the tested notes, 4% contained traces of ecstasy.

— One in 100 notes tested positive for high levels of both cocaine and ecstasy.

Students must assess whether the results suggest that Britain has a serious drug problem (as is implied by the article). While presence of cocaine on 99.2% of the 500 bills sampled initially seems to imply a very serious problem, a closer look belies this conclusion. Suppose that 1% of the British population is using cocaine and could contaminate a bill; the distribution of the number of times a bill could be contaminated in its life is approximately binomial with a probability of success (contamination) of 0.01 on each trial, where the number of trials is the number of times a bill is passed throughout its circulation.

The class is actually interested in the probability that a bill is contaminated at least one time in a given number of trials. This can most easily be calculated by subtracting the probability of the complementary event (a bill is never contaminated in a given number of trials) from 1.0. Once a student decides on what s/he feels is a reasonable number of times a bill will be passed from one person to another during its circulation, it is easy to calculate this probability—for 1,000 trials under these conditions, the probability is approximately 0.0000432! Thus, given these assumptions, the results (if reliable) provide no evidence for (or against) the claim. Of course, solutions will vary across students, as they will each make their own assumptions about the number of times a bill will be passed in its circulation and the probability that it will be contaminated in any one of these passes. Students must also consider the reliability of the data: The observations may not be independent; the article does not indicate whether a probability sample was taken; presence of cocaine on a bill is likely a very poor proxy for usage; and cocaine could be spread to notes by bank sorting machines or contact with dirty notes during storage.

The author uses this case in undergraduate introductory statistics classes. It provides students with an opportunity to work with and develop a deeper understanding of basic probability and discrete probability distributions, and it requires them to make and explain assumptions in modeling this problem.

*Bowie Kuhn's Worst Nightmare.* In *Bowie Kuhn's Worst Nightmare* (Cochran [29]), the 1981 Major League Baseball (MLB) season provides the backdrop for a case that the author uses to help students simultaneously develop a broad understanding of integer programming and Simpson's Paradox. MLB players went out on strike after the completion of approximately one-third of the 1981 season. After another one-third of the MLB season had passed, the players and owners approved a new collective bargaining agreement. Both sides also agreed that games cancelled during the strike would not be rescheduled and the 1981 season would be divided into two "halves" (prestrike and poststrike). In each division (East and West) of each league (National and American), the team with the best prestrike record would play the team with the best poststrike record for the division championship. If the same team had both the best pre- and poststrike records within a division, that team would play the team with the second-best poststrike record for the division championship.

After devising their scheme for dealing with the playoffs, Baseball Commissioner Bowie Kuhn and the 26 franchise owners had to be concerned about whether either of two potentially embarrassing circumstances could arise:

— A team finishes with the best overall (combined pre- and poststrike) record in its division, yet fails to qualify for its divisional playoffs, or

— A team finishes first in its division in both halves of the split season but does not have the best combined (pre- and poststrike) record in its division.

Occurrence of either of these situations would further erode the fan base's diminished confidence in MLB.

Students are provided with the prestrike win-loss records and number of poststrike games scheduled for each team. As students use these data to answer both questions with relatively simple integer programming models, they have a few interesting issues to face:

— What is the criterion for comparing teams? One could use *games behind* (which is actually used by MLB and reported by newspapers) or individual team *win/loss percentages* (which is also reported by newspapers and better understood by many fans). It is important to note that these two approaches give the same results if the teams play an equal number of games.

— What happens if a team does not play all of its poststrike games (because of poor weather, etc.)?

It is easy to envision how a team could finish with the best overall (combined pre- and poststrike) record in its division, yet not qualify for its divisional playoffs (this actually happened in two divisions). On the other hand, it is much more difficult to foresee how a team could finish first in its division in both halves of the split season, but not have the best combined (pre- and poststrike) record in its division. If the student uses the games behind criterion, s/he will eventually prove that this cannot happen no matter how many poststrike games the teams play. However, if the student uses the win-loss percentage criterion, s/he will find that this can happen if there is a large discrepancy in the number of poststrike games the teams play—these students have used integer programming to demonstrate the potential for Simpson's Paradox and have gained tremendous insight into this phenomenon! This case is useful and appropriate for undergraduate and MBA introductory operations research courses (although the undergraduate students may need some guidance).

*Dynamo Automobile Alarms.* The *Dynamo Automobile Alarms* case is based on a small consulting project on which the author worked several years ago. The heart of the problem is a simple production-mix formulation; Dynamo Corporation produces a single product—automobile alarms—and must decide how many to produce during each quarter of the upcoming fiscal year. Dynamo's current inventory, per unit cost of maintaining inventory, and forecasted quarterly demand for the fiscal year are given in the case. The case also provides an explanation that, due to the competitive nature of the automobile alarm industry, Dynamo's customers will not place backorders. Dynamo's quarterly regular and overtime production capacities and per unit costs are provided. Finally, the case indicates that Dynamo's current labor agreement with the Automobile Alarm Workers' Union (AAWU) limits the amount of overtime labor (and overtime production) over two consecutive quarters, and the number of automobile alarms produced using overtime labor during the last quarter of the previous year is provided.

The case narrative then explains that Dynamo has the option of reopening its labor contract negotiations. Because Dynamo management believes that labor's limit on the amount of overtime production will constrict profit in the upcoming fiscal year, the company would like to negotiate a relaxation on the restriction on overtime in consecutive quarters. Students are instructed to use this information to provide Dynamo's executives with some guidelines for their labor negotiations, i.e., the amount they should be willing to pay per unit produced with overtime labor and how many units they should be willing to manufacture at that rate. This case, which can be used in undergraduate and MBA introductory operations research courses, requires the students to consider and use shadow prices to develop a strategy. It is particularly interesting because the nature of shadow prices allows for the development of several different stair-step negotiation strategies.

## Miscellaneous

Several other active learning exercises defy categorization but are worthy of consideration. Descriptions of two such exercises follow.

*Legos.* One of the most well-known and popular active learning exercises for the operations research classroom is Pendegraft's [62] Lego exercise. The class is divided into small teams, each of which is given a bag containing eight small and six large Legos; these pieces represent the two inputs necessary to manufacture tables and chairs. Instructions on how to construct a table or chair from the Lego pieces are provided; tables are built from two large and two small Legos, while chairs are built from one large and two small Legos. After the teams are told that tables sell for \$16 and chairs sell for \$10, they are then given a few minutes to attempt to find a product mix that maximizes profits from the available resources (the Lego pieces).

If this game is played prior to coverage of linear programming, it serves as an excellent proactive exercise; although many student teams can find the optimal mix through trial and error, some teams will arrive at some suboptimal solution. The entire class can then work through the formulation, which then can provide the basis for discussions on graphical, algebraic, and/or spreadsheet approaches to solving the problem to optimality. Students quickly develop a basic understanding of linear programming and generally agree on its value. Of course, the game can be extended—the results provide a fertile and concrete example of how the marginal values of the various resources change as the class moves from the current extreme point to an adjacent extreme point on the feasible region. The game also provides a natural lead-in to integer programming.

*Monty Hall (or Three Door) Problem.* The Monty Hall problem is a well-known and interesting puzzle that can be used to teach several basic probability concepts. Monty Hall, the host of the popular television game show *Let's Make A Deal*, offers a contestant a choice of three doors. The contestant will receive the prize that is concealed behind the chosen door. Behind one door is a very desirable prize, while the other two doors conceal undesirable prizes. After the contestant selects a door, Monty opens one of the remaining two doors and exposes an undesirable prize (because there are two undesirable prizes and Monty knows their locations, he can always do this). Monty then offers to trade the remaining unopened door for the door the contestant originally chose. The contestant's optimal decision is to swap doors with Monty—her/his probability of winning the desirable prize increases from one out of three to two out three.

To use the Monty Hall problem in class, the instructor begins by dividing a deck of playing cards into sets of three; each set should contain exactly two cards with identical face values. S/he asks the students to find a partner and distributes one set of the cards to each student pair. One member of each student pair acts as Monty Hall and the other acts as the contestant, and the student pairs use the cards to repeatedly play the game while keeping track of how often the contestant wins when he (i) keeps his original card and (ii) swaps with Monty. After several executions (perhaps 20 times), the students reverse roles and play again for several executions. After both students in each pair have had an opportunity to act as the contestant, each student pair summarizes their results and writes a paragraph justifying what they believe to be the contestant's optimal decision.

This drill enables students to intuitively understand the solution to the Monty Hall problem, and by doing so also gain deeper insight into several basic probability concepts. The instructor uses this exercise in undergraduate introductory statistics courses.

## Conclusions

Lectures are generally far more effective when they are periodically interrupted; a break can effectively recapture the students' attention and interest. When punctuated with an interesting and relevant active learning exercise, the break becomes even more effective. These activities do not have to be elaborate to be successful; a simple exercise can reveal subtleties, emphasize key points, expose similarities in and differences between concepts, and provide transitions to new topics. Such an exercise can also encourage students to think about a

concept in an engaged manner, help them integrate ideas, and promote greater retention of course material. This can all be accomplished at a relatively small cost to both the instructor and the student—if they are willing to put forth a small amount of effort, the rewards can be tremendous. For lists of further readings on active learning, see the INFORM-*ED* website on active learning (http://education.forum.informs.org/active.htm), Charles C. Bonwell's *The Active Learning Site* [5], or Richard M. Felder's *Active and Cooperative Learning* Site [37].

## References

[1] Active Learning Online. http://www.acu.edu/cte/activelearning/.

[2] H. Barrows. *Designing a Problem Based Curriculum for the Pre-Clinical Years*. Springer, New York, 1985.

[3] J. Benek-Rivera and V. E. Mathews. Active learning with *Jeopardy*: Students ask the questions. *Journal of Management Education* 28(1):104–118, 2004.

[4] A. Biesterfeld. The price (or probability) is right. *Journal of Statistics Education* 9(3): 2001. http://www.amstat.org/publications/jse/v9n3/biesterfeld.html.

[5] C. C. Bonwell. *The Active Learning Site*. http://www.active-learning-site.com/.

[6] C. C. Bonwell and J. A. Eison. Active learning: Creating excitement in the classroom. ASHE-ERIC Higher Education Report No.1. George Washington University, Washington, D.C., 1991.

[7] R. Bosch. Big Mac attack. *ORMS Today* 20(4):30–31, 1993.

[8] G. Brown and M. Atkins. *Effective Teaching in Higher Education*, Methuen, London, 1988.

[9] W. E. Campbell and K. A. Smith. *New Paradigms for College Teaching*. Jossey-Bass, San Francisco, CA, 1995.

[10] W. E. Cashin. Improving lectures. IDEA Paper No. 14. Kansas State University Center for Faculty Evaluation and Development, Manhattan, KS, 1985.

[11] R. T. Chappell. Can TQM in public education survive without co-production? *Quality Progress* 27(7):41–44, 1994.

[12] A. W. Chickering and Z. F. Gamson. Seven principles for good practice. *AAHE Bulletin* 39:3–7, 1987.

[13] G. W. Chilcoat. Instructional behaviors for clearer presentations in the classroom. *Instructional Science* 18:289–314, 1989.

[14] M. J. Chlond. Unconstrained peg solitaire. *INFORMS Transactions on Education* 2(3), 2002a. http://ite.pubs.informs.org/Vol2No3/Chlond/.

[15] M. J. Chlond. The traveling space telescope problem. *INFORMS Transactions on Education* 3(1), 2002b. http://ite.pubs.informs.org/Vol3No1/Chlond/.

[16] M. J. Chlond. A new kind of IP. *INFORMS Transactions on Education* 4(2), 2004a. http://ite.pubs.informs.org/Vol4No2/Chlond/.

[17] M. J. Chlond. Fantasy OR. *INFORMS Transactions on Education* 4(3), 2004b. http://ite.pubs.informs.org/Vol4No3/Chlond/.

[18] M. J. Chlond. Classroom exercises in IP modeling: Su Doku and the log pile. *INFORMS Transactions on Education* 5(2), 2005. http://ite.pubs.informs.org/Vol5No2/Chlond/.

[19] M. J. Chlond and C. M. Toase. IP modeling of chessboard placements and related puzzles. *INFORMS Transactions on Education* 2(2), 2002. http://ite.informs.org/vol2no2/ChlondToase/.

[20] M. J. Chlond and C. M. Toase. IP modeling and the logical puzzles of Raymond Smullyan. *INFORMS Transactions on Education* 3(3), 2003. http://ite.pubs.informs.org/Vol3No3/ChlondToase/.

[21] M. J. Chlond, R. C. Daniel, and S. Heipcke. Fiveleapers a-leaping. *INFORMS Transactions on Education* 4(1), 2003. http://ite.pubs.informs.org/Vol4No1/ChlondDanielHeipcke/.

[22] C. S. Claxton and P. H. Murrell. Learning styles: Implications for improving educational practice. ASHE-ERIC Higher Education Report No. 4. George Washington University, Washington, D.C., 1987.

[23] W. H. Cliff and L. N. Curtin. The directed case method. *Journal of College Science Teaching* 30(1):64–66, 2000.

[24] J. J. Cochran. A case methodology for introductory statistics & MS courses in undergraduate business programs. *INFORMS Annual Meeting*, Seattle, WA (October) 1998.

[25] J. J. Cochran. *You've Got Cocaine in Your Wallet: A Simple and Effective Probability Case. INFORMS Annual Meeting*, Salt Lake City, UT, 2000.

[26] J. J. Cochran. Successful use of cases in introductory undergraduate business college operations research courses. *The Journal of the Operational Research Society* 51(12):1378–1386, 2000.

[27] J. J. Cochran. *Who Wants To Be A Millionaire—The Classroom Edition. INFORMS Transactions on Education* (1)3, 2001a. http://ite.informs.org/Vol1No3/Cochran/.

[28] J. J. Cochran. Probability, statistics, and "playing games." *ORMS Today* 28(2):14, 2001b.

[29] J. J. Cochran. *Bowie Kuhn's Worst Nightmare. INFORMS Transactions on Education* 5(1), 2004. http://ite.pubs.informs.org/Vol5No1/Cochran/.

[30] J. J. Cochran. Can you really learn basic probability by playing sports board games? *The American Statistician* 59(3):266–272, 2005.

[31] V. Dodd. You're carrying cocaine in your wallet—Drugs in Britain: Special report. *The Guardian* (October 4) 1999.

[32] S. Eckert. Teaching hypothesis testing with playing cards: A demonstration. *Journal of Statistics Education* 2(1), 1994. http://www.amstat.org/publications/jse/v2n1/eckert.html.

[33] E. Erkut. Big Mac attack revisited. *ORMS Today* 21(3):50–52, 1994.

[34] M. C. Feinstein and T. L. Veenendall. Using the case study method to teach interpersonal communication. *Inquiry: Critical Thinking Across the Disciplines* 9(3):11–14, 1992.

[35] R. M. Felder. Reaching the second tier: Learning and teaching styles in college science education. *Journal of College Science Teaching* 235:286–290, 1993.

[36] R. M. Felder. Cooperative learning in a sequence of engineering courses: A success story. *Cooperative Learning and College Teaching Newsletter* 5(2):10–13, 1995.

[37] R. M. Felder. Active and cooperative learning site. http://www.ncsu.edu/felder-public/Cooperative_Learning.html.

[38] R. M. Felder and R. Brent. Navigating the bumpy road to student-centered instruction. *College Teaching* 44(2):43–47, 1996.

[39] R. M. Felder and R. Brent. Learning by doing. *Chemical Engineering Education* 37(4):282–283, 2003.

[40] L. Feldman and F. Morgan. The pedagogy and probability of the dice game HOG. *Journal of Statistics Education* 11(2), 2003. www.amstat.org/publications/jse/v11n2/feldman.html.

[41] J. Garfield. Teaching statistics using small-group cooperative learning. *Journal of Statistics Education* 1(1), 1993. http://www.amstat.org/publications/jse/v1n1/garfield.html.

[42] P. G. George. The effectiveness of cooperative learning strategies in multicultural university classrooms. *Journal of Excellence in College Teaching* 51:21–30, 1994.

[43] G. Giraud. Cooperative learning and statistics instruction. *Journal of Statistics Education* 5(3), 1997. http://www.amstat.org/publications/jse/v5n3/giraud.html.

[44] M. Gnanadesikan, R. L. Scheaffer, A. E. Watkins, and J. A. Witmer. An activity-based statistics course. *Journal of Statistics Education* 5(2), 1997. http://www.amstat.org/publications/jse/v5n2/gnanadesikan.html.

[45] C. R. Haller, V. J. Gallagher, T. L. Weldon, and R. M. Felder, Dynamics of peer education in cooperative learning workgroups. *Journal Engineering Education* 89(3):285–293, 2000.

[46] J. Hartley and A. Cameron. Some observations on the efficiency of lecturing. *Educational Review* 20:30–37, 1967.

[47] J. Hartley and I. K. Davies. Note-taking: A critical review. *Programmed Learning and Educational Technology* 15:207–224, 1978.

[48] C. F. Herreid. Structured controversy: A case study strategy. *Journal of College Science Teaching* 26(2):95–101, 1996.

[49] C. F. Herried. The case study: Can case studies be used to teach critical thinking? *Journal of College Science Teaching* 33(6):12–15, 2004.

[50] INFORM-*ED* website on Active Learning. http://education.forum.informs.org/active.htm.

[51] D. W. Johnson, R. T. Johnson, and K. A. Smith. *Active Learning: Cooperation in the College Classroom.* Interaction Book Co., Edina, MN, 1991.

[52] C. M. Keeler and R. K. Steinhorst. Using small groups to promote active learning in the introductory statistics course: A report from the field. *Journal of Statistics Education* 3(2), 1995. http://www.amstat.org/publications/jse/v3n2/keeler.html.

[53] S. M. Kelley, K. M. Shemberg, B. S. Cowell, and B. J. Zinnbaur. Coping with students resistance to critical thinking: What the psychotherapy literature can tell us. *College Teaching* 43(4):140–145, 1995.

[54] C. Letavec and J. Ruggiero. The *n*-queens problem. *INFORMS Transactions on Education* 2(3), 2002. http://ite.pubs.informs.org/Vol2No3/LetavecRuggiero/.

[55] R. C. Magel. Using cooperative learning in a large introductory statistics class. *Journal of Statistics Education* 6(3), 1998. http://www.amstat.org/publications/jse/v6n3/magel.html.

[56] W. J. McKeachie. *Teaching Tips*, 10th ed. Houghton Mifflin Co., Boston, MA, 1999.

[57] W. J. McKeachie, P. R. Pintrich, Y. Lin, and D. A. F. Smith. Teaching and learning in the college classroom: A review of the research literature. Regents of The University of Michigan, Ann Arbor, MI, 1986.

[58] K. McKinney. Active learning. http://www.cat.ilstu.edu/teaching_tips/handouts/newactive.shtml.

[59] J. McLeish. *The Lecture Method*. Cambridge Monographs on Teaching Methods, Cambridge Institute of Education, Cambridge, England, 1968.

[60] D. Myers. LETGO, Inc. *INFORMS Transactions on Education* 3(1), 2002. http://ite.informs.org/Vol3No1/Myers/.

[61] J. S. Osland, D. A. Kolb, and I. M. Rubin. *Organizational Psychology: An Experiential Approach*, 7th ed. Prentice-Hall, Englewood Cliffs, NJ, 2001.

[62] N. Pendegraft. Lego of my simplex. *ORMS Today* 24(1):8, 1997.

[63] J. G. Penner. *Why Many College Teachers Cannot Lecture.* Charles C. Thomas, Springfield, IL, 1984.

[64] L. G. Richards, M. Gorman, W. T. Scherer, and R. D. Landel. Promoting active learning with cases and instructional modules. *Journal of Engineering Education* 84(4):375–381, 1995.

[65] K. L. Ruhl, C. A. Hughes, and P. J. Schloss. Using the pause procedure to enhance lecture recall. *Teacher Education and Special Education* 10:14–18, 1987.

[66] C. M. Rump. Who wants to see a $million error? *INFORMS Transactions on Education* 1(3), 2001. http://ite.informs.org/vol1no3/Rump/.

[67] D. J. Rumsey. A cooperative teaching approach to introductory statistics. *Journal of Statistics Education* 6(1), 1998. http://www.amstat.org/publications/jse/v6n1/rumsey.html.

[68] I. J. Russell, W. D. Hendricson, and R. J. Herbert. Effects of lecture information density on medical student achievement. *Journal of Medical Education* 59:881–889, 1984.

[69] D. M. Schweiger, W. R. Sandberg, and J. W. Ragan. Group approaches for improving strategic decision making: A comparative analysis of dialectical inquiry, devil's advocacy, and consensus. *Academy of Management Journal* 29(1):51–71, 1986.

[70] K. Seal. Teaching with technology. *INFORMS Teaching of Management Science Workshop*, Marlborough, MA (July) 2004.

[71] M. Sniedovich. OR/MS games: 1. A neglected educational resource. *INFORMS Transactions on Education* 2(3), 2002a. http://ite.informs.org/Vol2No3/Sniedovich/.

[72] M. Sniedovich. OR/MS games: 2. Towers of Hanoi. *INFORMS Transactions on Education* 3(1), 2002b. http://ite.pubs.informs.org/Vol3No1/Sniedovich/.

[73] M. Sniedovich. OR/MS games: 3. Counterfeit coin problem. *INFORMS Transactions on Education* 3(2), 2003. http://ite.pubs.informs.org/Vol3No2/Sniedovich/.

[74] W. J. Stepien and S. A. Gallagher. Problem-based learning: As authentic as it gets. *Educational Leadership* 50(7):25–28, 1993.

[75] M. Trick. Building a better game through dynamic programming: A flip analysis. *INFORMS Transactions on Education* 2(1), 2001. http://ite.informs.org/vol2no1/trick/.

[76] G. Whitaker. Co-production: Citizen participation in service delivery. *Public Administration Review* (May–June):240–242, 1980.

[77] D. R. Woods. 1996. *Problem-Based Learning: How to Gain the Most from PBL.* Donald R. Woods, Waterdown, ON, 1996.

[78] J. Yeomans. Solving "Einstein's riddle" using spreadsheet optimization. *INFORMS Transactions on Education* 3(2), 2003. http://ite.pubs.informs.org/Vol3No2/Yeomans/.

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

ISBN 1-877640-21-2
DOI 10.1287/educ.1053.0020

# CBC User Guide*

***John Forrest, Robin Lougee-Heimer***
Department of Mathematical Sciences, IBM T. J. Watson Research Center, IBM Research,
1101 Kitchawan Road, Yorktown Heights, New York 10598
{jjforre@us.ibm.com, robinlh@us.ibm.com}

**Abstract** The Computational Infrastructure for Operations Research (COIN-OR) branch-and-cut solver (CBC) is an open-source mixed-integer program (MIP) solver. The performance of branch-and-cut algorithms can vary greatly with problem-specific customization, such as dictating that the order nodes in the search tree are traversed. CBC provides operations research professionals with a well-tested, robust, reusable code base for experimenting with advanced customizations of branch-and-cut algorithms. The CBC design makes the most commonly desired customizations readily possible: (a) dynamically selecting the next node in the search tree for processing, (b) using specialized criteria for determining which variable(s) to branch on, (c) calling tailormade heuristics to generate MIP-feasible solutions quickly, (d) including standard (or user-provided) cut generation in solving the linear program (LP) relaxations of the MIP, and (e) invoking customized subproblem solvers.

CBC is written in C++ and is intended to be used primarily as a callable library. CBC requires a linear program (LP) solver. CBC uses the COIN-OR open solver interface (OSI) to communicate with the user's choice of LP solver. CBC can use any LP solver with an OSI. The LP solver expected to be used most commonly is the freely available COIN-OR LP solver (CLP). CBC can be used as a branch-and-bound solver or as a branch-and-cut solver. For cut generators, CBC relies on the COIN-OR Cut Generation Library (CGL). CBC can use any cut generator written to CGL standards.

CBC is an active open-source project led by John Forrest. The full CBC source code is available under the Common Public License for industrial and academic use at www.coin-or.org.

This chapter introduces CBC and illustrates how to implement a variety of common branch-and-cut customizations in CBC. The chapter assumes familiarity with C++, fundamentals of mixed-integer programming, and basic knowledge of OSI.

**Keywords** software; branch and bound; cutting planes; mixed-integer programming; open-source software

## 1. Introduction

The COIN[1] branch-and-cut solver (CBC) is an open-source mixed-integer program (MIP) solver written in C++. CBC is intended to be used primarily as a callable library to create customized branch-and-cut solvers. A basic, standalone executable version is also available. CBC is an active open-source project led by John Forrest at www.coin-or.org.

### 1.1. Prerequisites

The primary users of CBC are expected to be developers implementing customized branch-and-cut algorithms in C++, using CBC as a library. Consequently, this chapter assumes a

* ©International Business Machines Corporation 2005. Reproduced by permission of International Business Machines, Armonk, NY.

[1] The complete acronym is "COIN-OR" which stands for the Compuational Infrastructure for Operations Research. For simplicity (and in keeping with the directory and function names) we will simply use "COIN."

working knowledge of C++, including basic object-oriented programming terminology, and familiarity with the fundamental concepts of linear programming (LP) and mixed-integer programming (MIP).

CBC relies on other parts of the COIN repository. CBC needs an LP solver and relies on the COIN open solver interface (OSI) to communicate with the user's choice of solver. Any LP solver with an OSI can be used with CBC. The LP solver expected to be used most commonly is COIN's native linear program solver, CLP. For cut generators, CBC relies on the COIN Cut Generation Library (CGL). Any cut generator written to CGL standards can be used with CBC. Some of the cut generators in CGL rely on other parts of COIN, e.g., CGL's Gomory cut generator rely on the factorization functionality of `CoinFactorization`. This chapter assumes basic familiarity with OSI and CGL.

Technically speaking, CBC accesses the solver (and sometimes the model and data it contains) through an `OsiSolverInterface`. For the sake of simplicity, we will refer to the `OsiSolverInterface` as "the solver" in this chapter, rather than "the standard application programming interface to the solver." We hope any confusion caused by blurring this distinction will be mitigated by the shorter sentences.

In summary, readers should have the following prerequisites:

- C++ knowledge,
- LP and MIP fundamentals, and
- OSI familiarity.

Unless otherwise stated, we will assume the problem being optimized is a minimization problem. The terms "model" and "problem" are used synonymously.

## 1.2. Branch-and-Cut Overview

Before examining CBC in more detail, we tersely describe the basic branch-and-cut algorithm by way of example (which should really be called branch-and-cut-and-bound) and show the major C++ class(es) in CBC related to each step. The major CBC classes, labelled (A) through (F), are described in Table 1.

*Step* 1: *Bound.* Given an MIP model to minimize where some variables must take on integer values (e.g., 0, 1, or 2), relax the integrality requirements (e.g., consider each "integer"

Table 1. Associated classes.

| Note | Class name | Description |
|---|---|---|
| (A) | `CbcBranch...` | These classes define the nature of MIP's discontinuity. The simplest discontinuity is a variable that must take an integral value. Other types of discontinuities exist, e.g., lot-sizing variables. |
| (B) | `CbcNode` | This class decides which variable/entity to branch on next. Even advanced users will probably only interact with this class by setting `CbcModel` parameters (e.g., priorities). |
| (C) | `CbcTree` | All unsolved models can be thought of as being nodes on a tree where each node (model) can branch two or more times. The user should not need to be concerned with this class. |
| (D) | `CbcCompare...` | These classes are used to determine which of the unexplored nodes in the tree to consider next. These classes are very small simple classes that can be tailored to suit the problem. |
| (E) | `CglCutGenerators` | Any cut generator from CGL can be used in CBC. The cut generators are passed to CBC with parameters that modify when each generator will be tried. All cut generators should be tried to determine which are effective. Few users will write their own cut generators. |
| (F) | `CbcHeuristics` | Heuristics are very important for obtaining valid solutions quickly. Some heuristics are available, but this is an area where it is useful and interesting to write specialized ones. |

variable to be continuous with a lower bound of 0.0 and an upper bound of 2.0). Solve the resulting linear model with an LP solver to obtain a lower bound on the MIP's objective function value. If the optimal LP solution has integer values for the MIP's integer variables, we are finished. Any MIP-feasible solution provides an upper bound on the objective value. The upper bound equals the lower bound; the solution is optimal.

*Step* 2: *Branch.* Otherwise, there exists an "integer" variable with a nonintegral value. Choose one nonintegral variable (e.g., with value 1.3) (A)(B) and branch. Create two nodes, one with the branching variable having an upper bound of 1.0, and the other with the branching variable having a lower bound of 2.0. Add the two nodes to the search tree.

While (search tree is not empty)

*Step* 3: *Choose Node.* Pick a node off the tree (C)(D).

*Step* 4: *Reoptimize LP.* Create an LP relaxation and solve.

*Step* 5: *Bound.* Interrogate the optimal LP solution, and try to prune the node by one of the following.

- LP is infeasible, prune the node.
- Else, the optimal LP solution value of the node exceeds the current upper bound, prune the node.
- Else, the optimal LP solution of the node does not exceed the current upper bound and the solution is feasible to the MIP. Update the upper bound, and the best-known MIP solution, and prune the node by optimality.

*Step* 6: *Branch.* If we were unable to prune the node, then branch. Choose one nonintegral variable to branch on (A)(B). Create two nodes and add them to the search tree.

This is the outline of a branch-and-bound algorithm. If in optimizing the linear programs, we use cuts to tighten the LP relaxations (E)(F), then we have a branch-and-cut algorithm. (Note, if cuts are only used in Step 1, the method is called a cut-and-branch algorithm.)

There are a number of resources available to help new CBC users get started. This chapter is designed to be used in conjunction with the files in the Samples subdirectory of the main CBC directory (`COIN/Cbc/Samples`). The samples illustrate how to use CBC and may also serve as useful starting points for user projects. In the event that either this chapter or the available Doxygen content conflicts with the observed behavior of the source code, the comments in the header files, found in `COIN/Cbc/include`, are the ultimate reference.

## 2. The CBC Model Class

The main class in CBC is `CbcModel`. The `CbcModel` class is where most of the parameter setting is done. The absolute minimum number of actions taken with `CbcModel` is two:

- `CbcModel(OsiSolverInterface & linearSolver)` as constructor and
- `branchAndBound()` for solving the problem.

### 2.1. Simple Branch-and-Bound Example

The first sample program shows how to perform simple branch and bound with CBC. This program is short enough to present in full. Most of the remaining examples will take the form of small code fragments. The complete code for all the examples in this chapter can be found in the CBC Samples directory, `COIN/Cbc/Samples`.

**Example 1.** `minimum.cpp`

```
// Copyright (C) 2005, International Business Machines
// Corporation and others.  All Rights Reserved.

#include "CbcModel.hpp"

// Using CLP as the solver
#include "OsiClpSolverInterface.hpp"
```

```
int main (int argc, const char *argv[]) {
  OsiClpSolverInterface solver1;

  // Read in example model in MPS file format
  // and assert that it is a clean model
  int numMpsReadErrors = solver1.readMps("../../Mps/Sample/p0033.mps","");
  assert(numMpsReadErrors==0);

  // Pass the solver with the problem to be solved to CbcModel
  CbcModel model(solver1);

  // Do complete search
  model.branchAndBound();

  /* Print the solution.  CbcModel clones the solver so we
     need to get current copy from the CbcModel */
  int numberColumns = model.solver()->getNumCols();

  const double * solution = model.bestSolution();

  for (int iColumn=0;iColumn<numberColumns;iColumn++) {
    double value=solution[iColumn];
    if (fabs(value)>1.0e-7&&model.solver()->isInteger(iColumn))
      printf("%d has value %g\n",iColumn,value);
   }
  return 0;
}
```

The program in Example 1 creates a `OsiClpSolverInterface` solver interface (i.e., `solver1`) and reads an MPS file. If there are no errors, the program passes the problem to `CbcModel`, which solves the problem using the branch-and-bound algorithm. The part of the program that solves the problem is very small—one line!—but before that one line, the LP solver (i.e., `solver1`) had to be created and populated with the problem. After that one line, the results were printed out.

## 2.2. The Relationship Between OSI and CBC

The program in Example 1 illustrates the dependency of CBC on the `OsiSolverInterface` class. The constructor of `CbcModel` takes a pointer to an `OsiSolverInterface` (i.e., a solver). The `CbcModel` clones the solver, and uses its own instance of the solver. The `CbcModel`'s solver and the original solver (e.g., `solver1`) are not necessarily in sync unless the user synchronizes them. The user can always access the `CbcModel`'s solver through the `model()` class. To synchronize the two solvers, explicitly refreshing the original, e.g.,

```
solver1 = model.solver();
```

`CbcModel`'s method `solver()` returns a pointer to CBC's cloned solver.

For convenience, many of the OSI methods to access problem data have identical method names in `CbcModel`. (It is just more convenient to type `model.getNumCols()` rather than `model.solver()->getNumCols()`.) The `CbcModel` refreshes its solver at certain logical points during the algorithm. At these points, the information from the `CbcModel` model will match the information from the `model.solver()`. Elsewhere the information may vary. For instance, the method `CbcModel::bestSolution()` will contain the best solution so far, while the OSI method `getColSolution()` may not. In this case, it is safer to use `CbcModel::bestSolution()`.

While all the OSI methods used in `minimum.cpp` have equivalent methods in `CbcModel`, there are some OSI methods that do not. For example, if the program produced a lot of

undesired output, one might add the line

```
model.solver()->setHintParam(OsiDoReducePrint,true,OsiHintTry);
```

to reduce the output. There is no `setHintParam()` method in `CbcModel`.

### 2.3. Getting Solution Information and Impacting the Solution Process

Optimality can be checked through a call to `model.isProvenOptimal()`. Also available are `isProvenInfeasible()`, `isSolutionLimitReached()`, `isNodeLimitReached()`, or the feared `isAbandoned()`. There is also `int status()`, which returns 0 if finished (which includes the case when the algorithm is finished because it has been proved infeasible), 1 if stopped by user, and 2 if difficulties arose.

In addition to these `CbcModel` methods, solution values can be accessed via OSI methods (see Table 2). The OSI methods pick up the current solution in the `CbcModel`. The current solution will match the best solution found so far if called after `branchAndBound()` and a solution was found.

Most of the parameter setting in CBC is done through `CbcModel` methods. The most commonly used set and get methods are listed in Table 3.

`CbcModel` is extremely flexible and customizable. The class structure of CBC is designed to make the most commonly desired customizations of branch and cut possible. These include:

- selecting the next node to consider in the search tree,
- determining which variable to branch on,
- using heuristics to generate MIP-feasible solutions quickly,
- including cut generation when solving the LP-relaxations, and
- invoking customized subproblem solvers.

To enable this flexibility, `CbcModel` uses other classes in CBC (some of which are virtual and may have multiple instances). Not all classes are created equal. Tables 4 and 5 list in alphabetical order the classes used by `CbcModel` that are of most interest and of least interest. There is not much about the classes listed in Table 5 that the average user needs to know about.

## 3. Selecting the Next Node in the Search Tree

The order in which the nodes of the search tree are explored can strongly influence the performance of branch-and-cut algorithms. CBC gives users complete control over the search order. The search order is controlled via the `CbcCompare...` class. CBC provides an abstract

TABLE 2. Methods for getting solution information from OSI.

| Purpose | Name | Notes |
|---|---|---|
| Primal-column solution | `const double * getColSolution()` | The OSI method will return the best solution found thus far, unless none has been found. It is safer to use `CbcModel` version, `CbcModel::bestSolution()`. |
| Dual-row solution | `const double * getRowPrice()` | Identical `CbcModel` version available, `CbcModel::getRowPrice()`. |
| Primal-row solution | `const double * getRowActivity()` | Identical `CbcModel` version available, `CbcModel::getRowActivity()`. |
| Dual-column solution | `const double * getReducedCost()` | Identical `CbcModel` version available, `CbcModel::getReducedCost()`. |
| Number of rows in model | `int getNumRows()` | Identical `CbcModel` version available, `CbcModel::getNumRows()`. Note: the number of rows can change due to cuts. |
| Number of columns in model | `int getNumCols()` | Identical `CbcModel` version available, `CbcModel::getNumCols()`. |

TABLE 3. Useful set and get methods in `CbcModel`.

| Method(s) | Description |
|---|---|
| `bool setMaximumNodes(int value)`<br>`int getMaximumNodes() const`<br>`bool setMaximumSeconds(double value)`<br>`double getMaximumSeconds()`<br>`bool setMaximumSolutions(double value)`<br>`double getMaximumSolutions() const` | These set methods tell CBC to stop after a given number of nodes, seconds, or solutions is reached. The get methods return the corresponding values. |
| `bool setIntegerTolerance(double value)`<br>`const double getIntegerTolerance() const` | An integer variable is deemed to be at an integral value if it is no further than this *`value`* (tolerance) away. |
| `bool setAllowableGap(double value)`<br>`double getAllowableGap() const`<br>`bool setAllowablePercentageGap (double value)`<br>`double getAllowablePercentageGap() const`<br>`bool setAllowableFractionGap(double value)`<br>`double getAllowableFractionGap() const` | `CbcModel` returns if the gap between the best known solution and the best possible solution is less than this *`value`*, as a percentage or as a fraction. |
| `void setNumberStrong(double value)`<br>`int numberStrong() const` | These methods set or get the maximum number of candidates at a node to be evaluated for strong branching. |
| `void setPrintFrequency(int value)`<br>`int printFrequency() const` | Controls the number of nodes evaluated between status prints. Print frequency has a very slight overhead, if *`value`* is small. |
| `int getNodeCount() const` | Returns number of nodes evaluated in the search. |
| `int numberRowsAtContinuous() const` | Returns number of rows in the problem when handed to the solver (i.e., before cuts were added). Commonly used in implementing heuristics. |
| `int numberIntegers() const`<br>`const int * integerVariable() const` | Returns number of integer variables and an array specifying them. |
| `bool isBinary(int colIndex) const`<br>`bool isContinuous(int colIndex) const`<br>`bool isInteger(int colIndex) const` | Returns information on variable *`colIndex`*. OSI methods can be used to set these attributes (before handing the model to `CbcModel`). |
| `double getObjValue() const` | This method returns the best objective value so far. |
| `double getCurrentObjValue() const` | This method returns the current objective value. |
| `const double * getObjCoefficients() const` | This method returns the objective coefficients. |
| `const double * getRowLower() const`<br>`const double * getRowUpper() const`<br>`const double * getColLower() const`<br>`const double * getColUpper() const` | These methods return the lower and upper bounds on row and column activities. |
| `const CoinPackedMatrix * getMatrixByRow() const` | This method returns a pointer to a row copy of matrix stored as a `CoinPackedMatrix`, which can be further examined. |
| `const CoinPackedMatrix * getMatrixByCol() const` | This method returns a pointer to a column copy of matrix stored as a `CoinPackedMatrix`, which can be further examined. |
| `CoinBigIndex getNumElements() const` | Returns the number of nonzero elements in the problem matrix. |
| `void setObjSense(double value)`<br>`double getObjSense() const` | These methods set and get the objective sense. The parameter *`value`* should be $+1$ to minimize and $-1$ to maximize. |

*Notes.* The method `numberStrong` (and some of the others) does not follow the "get" convention. The convention has changed over time, and there are still some inconsistencies to be cleaned up. Also, `CoinBigIndex` is a `typedef`, which in most cases is the same as int.

base class, `CbcCompareBase`, and several commonly used instances that are described in Table 6.

It is relatively simple for a user to create new compare class instances. The code in Example 2 describes how to build a new comparison class and the reasoning behind it. The complete source can be found in `CbcCompareUser.hpp` and `CbcCompareUser.cpp`, located in the CBC Samples directory (see §7). The key method in `CbcCompare` is `bool test(CbcNode* x, CbcNode* y))`, which returns *`true`* if node *`y`* is preferred over node *`x`*. In the `test()`

TABLE 4. Classes used by `CbcModel`—Most useful.

| Class name | Description | Notes |
|---|---|---|
| `CbcCompareBase` | Controls which node on the tree is selected. | The default is `CbcCompareDefault`. Other comparison classes in `CbcCompareActual.hpp` include `CbcCompareDepth` and `CbcCompareObjective`. Experimenting with these classes and creating new compare classes is easy. |
| `CbcCutGenerator` | A wrapper for `CglCutGenerator` with additional data to control when the cut generator is invoked during the tree search. | Other than knowing how to add a cut generator to `CbcModel`, there is not much the average user needs to know about this class. However, sophisticated users can implement their own cut generators. |
| `CbcHeuristic` | Heuristic that attempts to generate valid MIP-solutions leading to good upper bounds. | Specialized heuristics can dramatically improve branch-and-cut performance. As many different heuristics as desired can be used in CBC. Advanced users should consider implementing custom heuristics when tackling difficult problems. |
| `CbcObject` | Defines what it means for a variable to be satisfied. | Used in branching. Virtual class. CBC's concept of branching is based on the idea of an "object." An object has (i) a feasible region, (ii) can be evaluated for infeasibility, (iii) can be branched on, e.g., a method of generating a branching object, which defines an up branch and a down branch, and (iv) allows comparison of the effect of branching. Instances of objects include `CbcSimpleInteger`, `CbcSimpleIntegerPseudoCosts`, `CbcClique`, `CbcSOS` (Type 1 and 2), `CbcFollowOn`, and `CbcLotsize`. |
| `OsiSolverInterface` | Defines the LP solver being used and the LP model. Normally a pointer to the desired `OsiSolverInterface` is passed to `CbcModel` before branch-and-cut. | Virtual class. The user instantiates the solver interface of their choice, e.g., `OsiClpSolverInterface`. |

method, information from `CbcNode` can easily be used. Table 7 lists some commonly used methods to access information at a node.

The node desired in the tree is often a function of how the search is progressing. In the design of CBC, there is no information on the state of the tree. The CBC is designed so that the method `newSolution()` is called whenever a solution is found, and the method `every1000Nodes()` is called every 1,000 nodes. When these methods are called, the user has the opportunity to modify the behavior of `test()` by adjusting their common variables (e.g., `weight_`). Because `CbcNode` has a pointer to the model, the user can also influence the search through actions such as changing the maximum time CBC is allowed, once a solution has been found (e.g., `CbcModel::setMaximumSeconds(double value)`). In `CbcCompareUser.cpp` of the `COIN/Cbc/Samples` directory, four items of data are used:

(1) the number of solutions found so far;

(2) the size of the tree (defined to be the number of active nodes);

(3) a weight, `weight_`, which is initialized to $-1.0$; and

(4) a saved value of weight, `saveWeight_` (for when weight is set back to $-1.0$ for a special reason).

TABLE 5. Classes used by `CbcModel`—Least useful.

| Class name | Description | Notes |
|---|---|---|
| `CbcBranchDecision` | Used in choosing which variable to branch on, however, most of the work is done by the definitions in `CbcObject`. | Defaults to `CbcBranchDefaultDecision`. |
| `CbcCountRowCut` | Interface to `OsiRowCut`. It counts the usage so cuts can gracefully vanish. | See `OsiRowCut` for more details. |
| `CbcNode` | Controls which variable/entity is selected to be branched on. | Controlled via `CbcModel` parameters. Information from `CbcNode` can be useful in creating customized node selection rules. |
| `CbcNodeInfo` | Contains data on bounds, basis, etc., for one node of the search tree. | Header is located in `CbcNode.hpp`. |
| `CbcTree` | Defines how the search tree is stored. | This class can be changed, but it is not likely to be modified. |
| `CoinMessageHandler` | Deals with message handling. | The user can inherit from `CoinMessageHandler` to specialize message handling. |
| `CoinWarmStartBasis` | Basis representation to be used by solver. | |

The full code for the `CbcCompareUser::test()` method is given in Example 2.

**Example 2.** `CbcCompareUser::test()`

```
// Returns true if y better than x
bool CbcCompareUser::test (CbcNode * x, CbcNode * y) {
  if (weight_==-1.0) {
    // before solution
    if (x->numberUnsatisfied() > y->numberUnsatisfied())
      return true;
    else if (x->numberUnsatisfied() < y->numberUnsatisfied())
      return false;
    else
      return x->depth() < y->depth();
  } else {
    // after solution.
    // note: if weight_=0, comparison is based
    //       solely on objective value
    double weight = CoinMax(weight_,0.0);
    return x->objectiveValue()+ weight*x->numberUnsatisfied() >
      y->objectiveValue() + weight*y->numberUnsatisfied();
  }
}
```

Initially, `weight_` is $-1.0$ and the search is biased toward depth first. In fact, `test()` prefers $y$ if $y$ has fewer unsatisfied variables. In the case of a tie, `test()` prefers the node with the greater depth in the tree.

Once a solution is found, `newSolution()` is called. The method `newSolution()` interacts with `test()` by means of the variable `weight_`. If the solution was achieved by branching, a calculation is made to determine the cost per unsatisfied integer variable to go from the continuous solution to an integer solution. The variable `weight_` is then set to aim at a slightly better solution. From then on, `test()` returns true if it seems that $y$ will lead to a better solution than $x$. This source for `newSolution()` is given in Example 3.

Table 6. Compare classes provided.

| Class name | Description |
|---|---|
| `CbcCompareDepth` | This will always choose the node deepest in the tree. It gives minimum tree size, but may take a long time to find the best solution. |
| `CbcCompareObjective` | This will always choose the node with the best objective value. This may give a very large tree. It is likely that the first solution found will be the best and the search should finish soon after the first solution is found. |
| `CbcCompareDefault` | This is designed to do a mostly depth-first search until a solution has been found. It then uses estimates that are designed to give a slightly better solution. If a reasonable number of nodes have been explored (or a reasonable number of solutions found), then this class will adopt a breadth-first search (i.e., making a comparison based strictly on objective function values) unless the tree is very large, in which case it will revert to depth-first search. |
| `CbcCompareEstimate` | When pseudo costs are invoked, they can be used to guess a solution. This class uses the guessed solution. |

Table 7. Information available from `CbcNode`.

| Class name | Description |
|---|---|
| `double objectiveValue() const` | Value of objective at the node. |
| `int numberUnsatisfied() const` | Number of unsatisfied integers (assuming branching object is an integer—otherwise it might be number of unsatisfied sets). |
| `int depth() const` | Depth of the node in the search tree. |
| `double guessedObjectiveValue() const` | If user was setting this (e.g., if using pseudo costs). |
| `int way() const` | The way in which branching would next occur from this node (for more advanced use). |
| `int variable() const` | The branching "variable" (associated with the `CbcBranchingObject`—for more advanced use). |

**Example 3.** `CbcCompareUser::newSolution()`

```
// This allows the test() method to change behavior by resetting weight_.
// It is called after each new solution is found.
void CbcCompareUser::newSolution(CbcModel * model,
                   double objectiveAtContinuous,
                   int numberInfeasibilitiesAtContinuous)
{
  if (model->getSolutionCount()==model->getNumberHeuristicSolutions())
    return; // solution was found by rounding so ignore it.

  // set weight_ to get close to this solution
  double costPerInteger =
    (model->getObjValue()-objectiveAtContinuous)/
    ((double) numberInfeasibilitiesAtContinuous);
  weight_ = 0.98*costPerInteger;
  saveWeight_=weight_;
  numberSolutions_++;
  if (numberSolutions_>5)
    weight_ =0.0; // comparison in test() will be
                  // based strictly on objective value.
}
```

As the search progresses, the comparison can be modified. If many nodes (or many solutions) have been generated, then `weight_` is set to 0.0, leading to a breadth-first search. Breadth-first search can lead to an enormous tree. If the tree size exceeds 10,000, it may be desirable to return to a search biased toward depth first. Changing the behavior in this manner is done by the method `every1000Nodes`, shown in Example 4.

**Example 4.** `CbcCompareUser::every1000Nodes()`

```
// This allows the test() method to change behavior every 1000 nodes
bool CbcCompareUser::every1000Nodes(CbcModel * model, int
numberNodes) {
  if (numberNodes>10000)
    weight_ =0.0; // compare nodes based on objective value
  // get size of tree
  treeSize_ = model->tree()->size();
  if (treeSize_>10000) {
    // set weight to reduce size most of time
    if (treeSize_>20000)
      weight_=-1.0;
    else if ((numberNodes%4000)!=0)
      weight_=-1.0;
    else
      weight_=saveWeight_;
  }
  return numberNodes==11000; // resort if first time
}
```

## 4. Getting Good Bounds in CBC

In practice, it is very useful to get a good solution reasonably fast. Any MIP-feasible solution produces an upper bound, and a good bound will greatly reduce the run time. Good solutions can satisfy the user on very large problems where a complete search is impossible. Obviously, heuristics are problem dependent, although some do have more general use. At present there is only one heuristic in CBC itself, `CbcRounding`. Hopefully, the number will grow. Other heuristics are in the `COIN/Cbc/Samples` directory. A heuristic tries to obtain a solution to the original problem so that it only needs to consider the original rows and does not have to use the current bounds. CBC provides an abstract base class `CbcHeuristic` and a rounding heuristic in CBC.

This chapter describes how to build a greedy heuristic for a set-covering problem, e.g., the miplib problem fast0507. A more general (and efficient) version of the heuristic is in `CbcHeuristicGreedy.hpp` and `CbcHeuristicGreedy.cpp`, located in the `COIN/Cbc/Samples` directory (see §7).

The greedy heuristic will leave all variables taking value 1 at this node of the tree at value 1, and will initially set all other variables to value 0. All variables are then sorted in order of their cost divided by the number of entries in rows that are not yet covered. (We may randomize that value a bit so that ties will be broken in different ways on different runs of the heuristic.) The best one is choosen, and set to 1. The process is repeated. Because this is a set-covering problem (i.e., all constraints are $\geq$), the heuristic is guaranteed to find a solution (but not necessarily an improved solution). The speed of the heuristic could be improved by just redoing those affected, but for illustrative purposes we will keep it simple. (The speed could also be improved if all elements equal 1.)

The key `CbcHeuristic` method is `int solution(double & solutionValue, double * betterSolution)`. The `solution()` method returns 0 if no solution found, and returns 1 if a solution is found, in which case it fills in the objective value and primal solution. The code in `CbcHeuristicGreedy.cpp` is a little more complicated than this following example. For instance, the code here assumes all variables are integer. The important bit of data is a copy of the matrix (stored by column) before any cuts have been made. The data used are bounds, objective, and the matrix, plus two work arrays.

**Example 5.** Data

```
OsiSolverInterface * solver = model_->solver();
  // Get solver from CbcModel
const double * columnLower = solver->getColLower(); // Column Bounds
const double * columnUpper = solver->getColUpper();
const double * rowLower = solver->getRowLower();
      // We know we only need lower bounds
const double * solution = solver->getColSolution();
const double * objective = solver->getObjCoefficients();
      // In code we also use min/max
double integerTolerance =
  model_->getDblParam(CbcModel::CbcIntegerTolerance);
double primalTolerance;
solver->getDblParam(OsiPrimalTolerance,primalTolerance);
int numberRows = originalNumberRows_;
      // This is number of rows when matrix was passed in
// Column copy of matrix (before cuts)
const double * element = matrix_.getElements();
const int * row = matrix_.getIndices();
const CoinBigIndex * columnStart = matrix_.getVectorStarts();
const int * columnLength = matrix_.getVectorLengths();

// Get solution array for heuristic solution
int numberColumns = solver->getNumCols();
double * newSolution = new double [numberColumns];
// And to sum row activities
double * rowActivity = new double[numberRows];
```

The `newSolution` is then initialized to the rounded-down solution.

**Example 6.** Initialize `newSolution`

```
for (iColumn=0;iColumn<numberColumns;iColumn++) {
  CoinBigIndex j;
  double value = solution[iColumn];
  // Round down integer
  if (fabs(floor(value+0.5)-value)<integerTolerance)
    value=floor(CoinMax(value+1.0e-3,columnLower[iColumn]));
  // make sure clean
  value = CoinMin(value,columnUpper[iColumn]);
  value = CoinMax(value,columnLower[iColumn]);
  newSolution[iColumn]=value;
  if (value) {
    double cost = objective[iColumn];
    newSolutionValue += value*cost;
    for (j=columnStart[iColumn];
         j<columnStart[iColumn]+columnLength[iColumn];j++) {
      int iRow=row[j];
      rowActivity[iRow] += value*element[j];
    }
  }
}
```

At this point, some row activities are below their lower bound. To correct the infeasibility, the variable that is cheapest in reducing the sum of infeasibilities is found and updated, and the process repeats. This is a finite process. (The implementation could be faster, but is kept simple for illustrative purposes.)

**Example 7.** Create Feasible `newSolution` from Initial `newSolution`

```
while (true) {
  // Get column with best ratio
  int bestColumn=-1;
  double bestRatio=COIN_DBL_MAX;
```

```
  for (int iColumn=0;iColumn<numberColumns;iColumn++) {
    CoinBigIndex j;
    double value = newSolution[iColumn];
    double cost = direction * objective[iColumn];
    // we could use original upper rather than current
    if (value+0.99<columnUpper[iColumn]) {
      double sum=0.0; // Compute how much we will reduce infeasibility by
      for (j=columnStart[iColumn];
           j<columnStart[iColumn]+columnLength[iColumn];j++) {
        int iRow=row[j];
        double gap = rowLower[iRow]-rowActivity[iRow];
        if (gap>1.0e-7) {
          sum += CoinMin(element[j],gap);
        if (element[j]+rowActivity[iRow]<rowLower[iRow]+1.0e-7) {
          sum += element[j];
    }
  }
      if (sum>0.0) {
    double ratio = (cost/sum)*(1.0+0.1*CoinDrand48());
    if (ratio<bestRatio) {
          bestRatio=ratio;
          bestColumn=iColumn;
        }
      }
    }
  }
  if (bestColumn<0)
    break; // we have finished
  // Increase chosen column
  newSolution[bestColumn] += 1.0;
  double cost = direction * objective[bestColumn];
  newSolutionValue += cost;
  for (CoinBigIndex j=columnStart[bestColumn];
       j<columnStart[bestColumn]+columnLength[bestColumn];j++) {
    int iRow = row[j];
    rowActivity[iRow] += element[j];
  }
}
```

A solution value of `newSolution` is compared to the best solution value. If `newSolution` is an improvement, its feasibility is validated.

**Example 8.** Check Solution Quality of `newSolution`

```
returnCode=0; // 0 means no good solution
if (newSolutionValue<solutionValue) { // minimization
  // check feasible
  memset(rowActivity,0,numberRows*sizeof(double));
  for (iColumn=0;iColumn<numberColumns;iColumn++) {
    CoinBigIndex j;
    double value = newSolution[iColumn];
    if (value) {
  for (j=columnStart[iColumn];
       j<columnStart[iColumn]+columnLength[iColumn];j++) {
    int iRow=row[j];
    rowActivity[iRow] += value*element[j];
      }
    }
  }
  // check was approximately feasible
  bool feasible=true;
  for (iRow=0;iRow<numberRows;iRow++) {
    if(rowActivity[iRow]<rowLower[iRow]) {
```

```
    if (rowActivity[iRow]<rowLower[iRow]-10.0*primalTolerance)
      feasible = false;
      }
    }
    if (feasible) {
      // new solution
      memcpy(betterSolution,newSolution,numberColumns*sizeof(double));
      solutionValue = newSolutionValue;
      // We have good solution
      returnCode=1;
    }
  }
```

## 5. Branching

CBC's concept of branching is based on the idea of an "object." An object has (i) a feasible region; (ii) can be evaluated for infeasibility; (iii) can be branched on, e.g., a method of generating a branching object, which defines an up branch and a down branch; and (iv) allows comparison of the effect of branching. Instances of objects include

(1) `CbcSimpleInteger`,
(2) `CbcSimpleIntegerPseudoCosts`,
(3) `CbcClique`,
(4) `CbcSOS` (Type 1 and 2),
(5) `CbcFollowOn`, and
(6) `CbcLotsize`.

In this section we give examples of how to use existing branching objects.

### 5.1. Pseudo Cost Branching

If the user declares variables as integer but does no more, then CBC will treat them as simple integer variables. In many cases, the user would like to do some more fine-tuning. This section shows how to create integer variables with pseudo costs. When pseudo costs are given, then it is assumed that if a variable is at 1.3, then the cost of branching that variable down will be 0.3 times the down pseudo cost and the cost of branching up would be 0.7 times the up pseudo cost. Pseudo costs can be used both for branching and for choosing a node. The full code is in `longthin.cpp`, located in the CBC Samples directory (see §7).

The idea is simple for set-covering problems. Branching up gets us much closer to an integer solution, so we will encourage that direction by branching up if variable value is greater than one-third. The expected cost of going up obviously depends on the cost of the variable. The pseudo costs are chosen to reflect that fact.

**Example 9.** `CbcSimpleIntegerPseudoCosts`

```
int iColumn;
int numberColumns = solver3->getNumCols();
// do pseudo costs
CbcObject ** objects = new CbcObject * [numberColumns];
// Point to objective
const double * objective = model.getObjCoefficients();
int numberIntegers=0;
for (iColumn=0;iColumn<numberColumns;iColumn++) {
  if (solver3->isInteger(iColumn)) {
    double cost = objective[iColumn];
    CbcSimpleIntegerPseudoCost * newObject =
      new CbcSimpleIntegerPseudoCost(&model,numberIntegers,iColumn,
                                     2.0*cost,cost);
    newObject->setMethod(3);
    objects[numberIntegers++]= newObject;
  }
```

```
}
// Now add in objects (they will replace simple integers)
model.addObjects(numberIntegers,objects);
for (iColumn=0;iColumn<numberIntegers;iColumn++)
  delete objects[iColumn];
delete [] objects;
```

The code in Example 9 also tries to give more importance to variables with more coefficients. Whether this sort of thing is worthwhile should be the subject of experimentation.

## 5.2. Follow-on Branching

In crew scheduling, the problems are long and thin. A problem may have a few rows, but many thousands of variables. Branching a variable to 1 is very powerful as it fixes many other variables to 0, but branching to 0 is very weak, as thousands of variables can increase from 0. In crew-scheduling problems, each constraint is a flight leg, e.g., JFK airport to DFW airport. From DFW there may be several flights the crew could take next—suppose one flight is the 9:30 flight from DFW to LAX airport. A binary branch is that the crew arriving at DFW either takes the 9:30 flight to LAX or they do not. This "follow-on" branching does not fix individual variables. Instead this branching divides all the variables with entries in the JFK-DFW constraint into two groups—those with entries in the DFW-LAX constraint and those without entries.

The full sample code for follow-on branching is in `crew.cpp`, located in the CBC Samples directory in §7. In this case, the simple integer variables are left, which may be necessary if other sorts of constraints exist. Follow-on branching rules are to be considered first, so the priorities are set to indicate that the follow-on rules take precedence, where Priority 1 is the highest priority.

**Example 10.** `CbcFollowOn`

```
int iColumn;
int numberColumns = solver3->getNumCols();
/* We are going to add a single follow-on object but we
   want to give low priority to existing integers
   As the default priority is 1000 we don't actually need to give
   integer priorities but it is here to show how.
*/
// Normal integer priorities
int * priority = new int [numberColumns];
int numberIntegers=0;
for (iColumn=0;iColumn<numberColumns;iColumn++) {
  if (solver3->isInteger(iColumn)) {
    priority[numberIntegers++]= 100; // low priority
  }
}
/* Second parameter is set to true for objects,
   and false for integers. This indicates integers */
model.passInPriorities(priority,false);
delete [] priority;
/* Add in objects before we can give them a priority.
   In this case just one object
   - but it shows the general method
*/
CbcObject ** objects = new CbcObject * [1];
objects[0]=new CbcFollowOn(&model);
model.addObjects(1,objects);
delete objects[0];
delete [] objects;
// High priority
int followPriority=1;
model.passInPriorities(&followPriority,true);
```

## 6. Advance Solver Uses

CBC uses a generic `OsiSolverInterface` and its `resolve` capability. This does not give much flexibility, so advanced users can inherit from their interface of choice. This section illustrates how to implement such a solver for a long thin problem, e.g., fast0507 again. As with the other examples in this chapter, the sample code is not guaranteed to be the fastest way to solve the problem. The main purpose of the example is to illustrate techniques. The full source is in `CbcSolver2.hpp` and `CbcSolver2.cpp` located in the CBC Samples directory (see §7).

The method `initialSolve` is called a few times in CBC, and provides a convenient starting point. The `modelPtr_` derives from `OsiClpSolverInterface`.

**Example 11.** `initialSolve()`

```
// modelPtr_ is of type ClpSimplex *
modelPtr_->setLogLevel(1); // switch on a bit of printout
modelPtr_->scaling(0); // We don't want scaling for fast0507
setBasis(basis_,modelPtr_); // Put basis into ClpSimplex
// Do long thin by sprint
ClpSolve options;
options.setSolveType(ClpSolve::usePrimalorSprint);
options.setPresolveType(ClpSolve::presolveOff);
options.setSpecialOption(1,3,15); // Do 15 sprint iterations
modelPtr_->initialSolve(options); // solve problem
basis_ = getBasis(modelPtr_); // save basis
modelPtr_->setLogLevel(0); // switch off printout
```

The `resolve()` method is more complicated than `initialSolve()`. The main pieces of data are a counter `count_` (which is incremented each solve), and an integer array `node_` (which stores the last time a variable was active in a solution). For the first few times, the normal dual simplex is called and `node_` array is updated.

**Example 12.** First Few Solves

```
if (count_<10) {
  OsiClpSolverInterface::resolve(); // Normal resolve
  if (modelPtr_->status()==0) {
    count_++; // feasible - save any nonzero or basic
    const double * solution = modelPtr_->primalColumnSolution();
    for (int i=0;i<numberColumns;i++) {
  if (solution[i]>1.0e-6||modelPtr_->getStatus(i)==ClpSimplex::basic) {
    node_[i]=CoinMax(count_,node_[i]);
    howMany_[i]++;
  }
    }
  } else {
    printf("infeasible early on\n");
  }
}
```

After the first few solves, only those variables that took part in a solution in the last so many solves are used. As fast0507 is a set-covering problem, any rows that are already covered can be taken out.

**Example 13.** Create Small Subproblem

```
  int * whichRow = new int[numberRows]; // Array to say which rows used
  int * whichColumn = new int [numberColumns]; // Array to say which
      columns used
  int i;
  const double * lower = modelPtr_->columnLower();
  const double * upper = modelPtr_->columnUpper();
  setBasis(basis_,modelPtr_); // Set basis
  int nNewCol=0; // Number of columns in small model
```

```
// Column copy of matrix
const double * element = modelPtr_->matrix()->getElements();
const int * row = modelPtr_->matrix()->getIndices();
const CoinBigIndex * columnStart =
    modelPtr_->matrix()->getVectorStarts();
const int * columnLength = modelPtr_->matrix()->getVectorLengths();

int * rowActivity = new int[numberRows];
            // Number of columns with entries in each row
memset(rowActivity,0,numberRows*sizeof(int));
int * rowActivity2 = new int[numberRows];
            // Lower bound on row activity for each row
memset(rowActivity2,0,numberRows*sizeof(int));
char * mark = (char *) modelPtr_->dualColumnSolution();
            // Get some space to mark columns
memset(mark,0,numberColumns);
for (i=0;i<numberColumns;i++) {
  bool choose = (node_[i]>count_-memory_&&node_[i]>0);
            // Choose if used recently
  // Take if used recently or active in some sense
  if ((choose&&upper[i])
  ||(modelPtr_->getStatus(i)!=ClpSimplex::atLowerBound&&
        modelPtr_->getStatus(i)!=ClpSimplex::isFixed)
  ||lower[i]>0.0) {
    mark[i]=1; // mark as used
whichColumn[nNewCol++]=i; // add to list
    CoinBigIndex j;
    double value = upper[i];
    if (value) {
      for (j=columnStart[i];
           j<columnStart[i]+columnLength[i];j++) {
        int iRow=row[j];
        assert (element[j]==1.0);
        rowActivity[iRow] ++; // This variable can cover this row
      }
      if (lower[i]>0.0) {
        for (j=columnStart[i];
             j<columnStart[i]+columnLength[i];j++) {
          int iRow=row[j];
          rowActivity2[iRow] ++; // This row redundant
        }
      }
    }
  }
}
int nOK=0; // Use to count rows which can be covered
int nNewRow=0; // Use to make list of rows needed
for (i=0;i<numberRows;i++) {
  if (rowActivity[i])
    nOK++;
  if (!rowActivity2[i])
    whichRow[nNewRow++]=i; // not satisfied
  else
    modelPtr_->setRowStatus(i,ClpSimplex::basic); // make slack basic
}
if (nOK<numberRows) {
  // The variables we have do not cover rows - see if we can find
    any that do
  for (i=0;i<numberColumns;i++) {
    if (!mark[i]&&upper[i]) {
      CoinBigIndex j;
      int good=0;
```

```
        for (j=columnStart[i];
             j<columnStart[i]+columnLength[i];j++) {
          int iRow=row[j];
          if (!rowActivity[iRow]) {
            rowActivity[iRow] ++;
            good++;
          }
        }
        if (good) {
          nOK+=good; // This covers - put in list
          whichColumn[nNewCol++]=i;
        }
      }
    }
  }
  delete [] rowActivity;
  delete [] rowActivity2;
  if (nOK<numberRows) {
    // By inspection the problem is infeasible - no need to solve
    modelPtr_->setProblemStatus(1);
    delete [] whichRow;
    delete [] whichColumn;
    printf("infeasible by inspection\n");
    return;
  }
  // Now make up a small model with the right rows and columns
  ClpSimplex *temp=
      new ClpSimplex(modelPtr_,nNewRow,whichRow,nNewCol,whichColumn);
```

If the variables cover the rows, then the problem is feasible (no cuts are being used). (If the rows were equality constraints, then this might not be the case. More work would be needed.) After the solution, the reduced costs are checked. If any reduced costs are negative, the code goes back to the full problem and cleans up with primal simplex.

**Example 14.** Check Optimal Solution

```
temp->setDualObjectiveLimit(1.0e50);
            // Switch off dual cutoff as problem is restricted
temp->dual(); // solve
double * solution = modelPtr_->primalColumnSolution();
    // put back solution
const double * solution2 = temp->primalColumnSolution();
memset(solution,0,numberColumns*sizeof(double));
for (i=0;i<nNewCol;i++) {
  int iColumn = whichColumn[i];
  solution[iColumn]=solution2[i];
  modelPtr_->setStatus(iColumn,temp->getStatus(i));
}
double * rowSolution = modelPtr_->primalRowSolution();
const double * rowSolution2 = temp->primalRowSolution();
double * dual = modelPtr_->dualRowSolution();
const double * dual2 = temp->dualRowSolution();
memset(dual,0,numberRows*sizeof(double));
for (i=0;i<nNewRow;i++) {
  int iRow=whichRow[i];
  modelPtr_->setRowStatus(iRow,temp->getRowStatus(i));
  rowSolution[iRow]=rowSolution2[i];
  dual[iRow]=dual2[i];
}
// See if optimal
double * dj = modelPtr_->dualColumnSolution();
// get reduced cost for large problem
// this assumes minimization
```

```
memcpy(dj,modelPtr_->objective(),numberColumns*sizeof(double));
modelPtr_->transposeTimes(-1.0,dual,dj);
modelPtr_->setObjectiveValue(temp->objectiveValue());
modelPtr_->setProblemStatus(0);
int nBad=0;

for (i=0;i<numberColumns;i++) {
  if (modelPtr_->getStatus(i)==ClpSimplex::atLowerBound
      &&upper[i]>lower[i]&&dj[i]<-1.0e-5)
    nBad++;
}
// If necessary clean up with primal (and save some statistics)
if (nBad) {
  timesBad_++;
  modelPtr_->primal(1);
  iterationsBad_ += modelPtr_->numberIterations();
}
```

The array `node_` is updated for the first few solves. To give some idea of the effect of this tactic, the problem fast0507 has 63,009 variables, but the small problem never has more than 4,000 variables. In only about 10% of solves was it necessary to re-solve, and then the average number of iterations on full problem was less than 20.

***Quadratic MIP.*** To give another example—again only for illustrative purposes—it is possible to do quadratic MIP with CBC. In this case, we make `resolve` the same as `initialSolve`. The full code is in `ClpQuadInterface.hpp` and `ClpQuadInterface.cpp`, located in the CBC Samples directory (see §7).

**Example 15.** Solving a Quadratic MIP

```
// save cutoff
double cutoff = modelPtr_->dualObjectiveLimit();
modelPtr_->setDualObjectiveLimit(1.0e50);
modelPtr_->scaling(0);
modelPtr_->setLogLevel(0);
// solve with no objective to get feasible solution
setBasis(basis_,modelPtr_);
modelPtr_->dual();
basis_ = getBasis(modelPtr_);
modelPtr_->setDualObjectiveLimit(cutoff);
if (modelPtr_->problemStatus())
  return; // problem was infeasible
// Now pass in quadratic objective
ClpObjective * saveObjective  = modelPtr_->objectiveAsObject();
modelPtr_->setObjectivePointer(quadraticObjective_);
modelPtr_->primal();
modelPtr_->setDualObjectiveLimit(cutoff);
if (modelPtr_->objectiveValue()>cutoff)
  modelPtr_->setProblemStatus(1);
modelPtr_->setObjectivePointer(saveObjective);
```

## 7. More Samples

The CBC distribution includes a number of `.cpp` sample files. Users are encouraged to use them as starting points for their own CBC projects. The files can be found in the `COIN/Cbc/Samples/` directory. For the latest information on compiling and running these samples, please see the file `COIN/Cbc/Samples/INSTALL`. Most of them can be built by

```
make DRIVER=name,
```

which produces an executable `testit`. Tables 8 and 9 provide lists of some of the most useful sample files with a short description for each file.

## 8. Messages

Messages and codes passed by CBC are listed in the tables below. For a complete list, see `COIN/Cbc/CbcMessages.cpp`. The notation used is the same as for the `printf` in the C programming language.

- `%s` is a string
- `%d` is an integer
- `%g` or `%f` is a floating point value

There are several log levels. Setting the log level to be $i$ produces the log messages for level $i$ and all levels less than $i$.

- Log Level 0: Switches off all CBC messages but one (see Table 10).
- Log Level 1: The default (see Table 11).
- Log Level 2: Substantial amount of information, e.g., message 15 is generated once per node. Can be useful when the evaluation at each node is slow (see Table 12).
- Log Level 3: Tremendous amount of information, e.g., multiple messages per node (see Table 13).

Table 8. Basic samples.

| Source file | Description |
|---|---|
| `minimum.cpp` | This is a CBC "Hello, world" program. It reads a problem in MPS file format and solves the problem using simple branch and bound. |
| `sample2.cpp` | This is designed to be a file that a user could modify to get a useful driver program for his or her project. In particular, it demonstrates the use of CGL's preprocess functionality. It uses `CbcBranchUser.cpp`, `CbcCompareUser.cpp`, and `CbcHeuristicUser.cpp` with corresponding `*.hpp` files. |

Table 9. Advanced samples.

| Source file | Description |
|---|---|
| `crew.cpp` | This sample shows the use of advanced branching and a use of priorities. It uses `CbcCompareUser.cpp` with corresponding `*.hpp` files. |
| `longthin.cpp` | This sample shows the advanced use of a solver. It also has coding for a greedy heuristic. The solver is given in `CbcSolver2.hpp` and `CbcSolver2.cpp`. The heuristic is given in `CbcHeuristicGreedy.hpp` and `CbcHeuristicGreedy.cpp`. It uses `CbcBranchUser.cpp` and `CbcCompareUser.cpp` with corresponding `*.hpp` files. |
| `qmip.cpp` | This solves a quadratic MIP. It is to show advanced use of a solver. The solver is given in `ClpQuadInterface.hpp` and `ClpQuadInterface.cpp`. It uses `CbcBranchUser.cpp` and `CbcCompareUser.cpp` with corresponding `*.hpp` files. |
| `sos.cpp` | This artificially creates a special ordered set problem. |
| `lotsize.cpp` | This artificially creates a lot-sizing problem. |

Table 10. CBC messages passed at Log Level 0.

| Code | Text and notes |
|---|---|
| 3007 | `No integer variables - nothing to do` |

Table 11. CBC messages passed at or above Log Level 1.

| Code | Text and notes |
|---|---|
| 1 | `Search completed - best objective %g, took %d iterations, and %d nodes` |
| 3 | `Exiting on maximum nodes` |
| 4 | `Integer solution of %g found after %d iterations and %d nodes` |
| 5 | `Partial search - best objective %g (best possible %g), took %d iterations, and %d nodes` |
| 6 | `The LP relaxation is infeasible or too expensive` |
| 9 | `Objective coefficients multiple of %g` |
| 10 | `After %d nodes, %d on tree, %g best solution, best possible %g` |
| 11 | `Exiting as integer gap of %g less than %g or %g%%` |
| 12 | `Integer solution of %g found by heuristic after %d iterations and %d nodes` |
| 13 | `At root node, %d cuts changed objective from %g to %g in %d passes` |
| 14 | `Cut generator %d (%s) - %d row cuts (%d active), %d column cuts %? in %g seconds - new frequency is %d` |
| 16 | `Integer solution of %g found by strong branching after %d iterations and %d nodes` |
| 17 | `%d solved, %d variables fixed, %d tightened` |
| 18 | `After tightenVubs, %d variables fixed, %d tightened` |
| 19 | `Exiting on maximum solutions` |
| 20 | `Exiting on maximum time` |
| 23 | `Cutoff set to %g - equivalent to best solution of %g` |
| 24 | `Integer solution of %g found by subtree after %d iterations and %d nodes` |
| 26 | `Setting priorities for objects %d to %d inclusive (out of %d)` |
| 3008 | `Strong branching is fixing too many variables, too expensively!` |

Table 12. CBC messages passed at or above Log Level 2.

| Code | Text and notes |
|---|---|
| 15 | `Node %d Obj %g Unsat %d depth %d` |
| 21 | `On closer inspection node is infeasible` |
| 22 | `On closer inspection objective value of %g above cutoff of %g` |
| 23 | `Allowing solution, even though largest row infeasibility is %g` |

Table 13. CBC messages passed at or above Log Level 3.

| Code | Text and notes |
|---|---|
| 7 | `Strong branching on %d (%d), down %g (%d) up %g (%d) value %g` |
| 25 | `%d cleanup iterations before strong branching` |

# Appendix

## Frequently Asked Questions

**Q:** What is CBC?
**A:** The COIN-OR branch-and-cut code is designed to be a high-quality mixed-integer code provided under the terms of the Common Public License. CBC is written in C++, and is primarily intended to be used as a callable library (though a rudimentary standalone executable exists).

**Q:** What are some of the features of CBC?
**A:** CBC allows the use of any CGL cuts and the use of heuristics and specialized branching methods.

**Q:** How do I obtain and install CBC?
**A:** Please see the COIN-OR FAQ at www.coin-or.org for details on how to obtain and install COIN-OR modules.

**Q:** Is CBC reliable?
**A:** CBC has been tested on many problems, but more testing and improvement is needed before it can get to Version 1.0.

**Q:** Is there any documentation for CBC?
**A:** A list of CBC class descriptions generated by Doxygen is available. The latest user guide is available at www.coin-or.org.

**Q:** Is CBC as fast as CPLEX or Xpress?
**A:** No. However, its design is much more flexible, so advanced users will be able to tailor CBC to their needs.

**Q:** When will Version 1.0 of CBC be available?
**A:** It is expected that Version 1.0 will be released in time for the 2005 INFORMS annual meeting.

**Q:** What can the community do to help?
**A:** People from all around the world are already helping. There are probably 10 people who do not always post to the discussion mail list but are constantly "improving" the code by demanding performance or bug fixes or enhancements, and there are others posting questions to discussion groups.

A good start is to join the coin-discuss mailing list where CBC is discussed. See www.coin-or.org/mail.html. Some other possibilities include:

- Comment on the design.
- Give feedback on the documentation and FAQs.
- Break the code, or better yet—mend it.
- Tackle any of the "to-dos" listed in the Doxyen documentation and contribute back to COIN-OR.

## Doxygen

There is Doxygen content for CBC available online at http://www.coin-or.org/Doxygen/Cbc/index.html. A local version of the Doxygen content can be generated from the CBC distribution. To do so, in the directory COIN/Cbc, enter `make doc`. The Doxygen content will be created in the directory `COIN/Cbc/Doc/html`. The same can be done for the COIN core, from the `COIN/Coin` directory.

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS
ISBN 1-877640-21-2

# Contributing Authors

**Ravindra K. Ahuja** ("Network Models in Railroad Planning and Scheduling") is a professor in the Department of Industrial and Systems Engineering at the University of Florida, Gainesville, Florida. He received his Ph.D. from the Indian Institute of Technology (IIT), Kanpur, in 1982. He conducts research on network flows, combinatorial optimization, and applications of these techniques in airline and railroad scheduling. He is coauthor of the well-known text and reference book *Network Flows: Theory, Algorithm, and Applications*, which won the prestigious Lanchester prize in 1993.

**J. Neil Bearden** ("Operations Research in Experimental Psychology") is currently a postdoctoral fellow at the Brain Imaging and Analysis Center, Duke University. Prior to coming to Duke, he was a research associate at the University of Arizona jointly in the Department of Management and Policy and the Department of Systems and Industrial Engineering. He received his Ph.D. from the University of North Carolina at Chapel Hill in 2003. His research interests include behavioral decision theory, optimization, and epistemology.

**Gerald G. Brown** ("Analyzing the Vulnerability of Critical Infrastructure to Attack and Planning Defenses") is a distinguished professor of operations research at the Naval Postgraduate School. His optimization research and decision support software have been used by every uniformed service, and by the majority of the Fortune 50, in dealing with problems ranging from supply chain design to capital planning. He receives research support from the Air Force Office of Scientific Research and the Office of Naval Research.

**W. Matthew Carlyle** ("Analyzing the Vulnerability of Critical Infrastructure to Attack and Planning Defenses") is an associate professor in the Operations Research Department at the Naval Postgraduate School. He received his Ph.D. in operations research from Stanford University in 1997. His research interests include network optimization, integer programming, and network interdiction. Applications of this research include attack and defense of critical infrastructure, delaying large industrial projects and weapons programs, theater ballistic missile defense, sensor mix and deployment, naval logistics, network diversion, underground mining, and semiconductor manufacturing.

**James J. Cochran** ("Active Learning for Quantitative Courses") received a Ph.D. in quantitative analysis from the University of Cincinnati in 1997. Currently an assistant professor at Louisiana Tech University, he was a visiting scholar with Stanford University's Statistics Department and has previously been on the faculties of Wright State University, Drexel University, Miami University, and the University of Cincinnati. He teaches courses in statistical methods, linear models, experimental design, multivariate statistics, categorical data analysis, mathematical statistics, operations research methods, and mathematical programming.

**Claudio B. Cunha** ("Network Models in Railroad Planning and Scheduling") is an assistant professor and graduate coordinator in the Department of Transportation Engineering at the Polytechnic Engineering School, University of Sao Paulo, in Brazil. He received his Ph.D. in transportation engineering at the University of Sao Paulo in 1997. He instructs in the areas

of operations research, transportation, and logistics. His current research interests involve developing efficient algorithms for combinatorial problems in logistics and transportation, particularly in location and network design, railroad planning and scheduling, and vehicle routing and scheduling.

**John Forrest** ("CBC User Guide") is a member of the Mathematical Sciences Department at the IBM T. J. Watson Research Center, where he served four years as the benevolent dictator of the COIN-OR initiative until a much-welcome coup by the nonprofit COIN-OR Foundation, Inc. John is well known for his contributions to the IBM OSL product and his open-source software projects, including the COIN-OR Linear Programming Solver, the Branch and Cut Solver, and the Cut Generation Library.

**Joseph Geunes** ("Demand Selection and Assignment Problems in Supply Chain Planning") conducts research that focuses on optimization-based models for practical production and inventory planning and control problems under various dimensions of planning flexibility. He is Co-Director of the Supply Chain and Logistics Engineering (SCALE) Center at the University of Florida. He received his Ph.D. and MBA from Pennsylvania State University's Smeal College of Business, and he holds a B.S. in electrical engineering from Drexel University.

**Illya V. Hicks** ("Branch and Tree Decomposition Techniques for Discrete Optimization") received his Ph.D. from Rice University in 2000. He is an assistant professor in the Department of Industrial Engineering at Texas A&M University. His research interests are in combinatorial optimization, graph theory, and integer programming. Some applications of interest are network design, intelligence analysis, cancer treatment, and logistics. His current research is focused on using branch decomposition techniques to solve NP-hard problems. Dr. Hicks is also currently the President of the Minority Issues Forum of INFORMS.

**Julie Higle** ("Stochastic Programming: Optimization When Uncertainty Matters") is a professor of systems and industrial engineering at the University of Arizona. She joined the faculty at UA after completing her Ph.D. in industrial and operations engineering (University of Michigan, 1985). She has been involved in stochastic programming research since the late 1980s, primarily through her collaboration with Suvrajeet Sen. She currently serves in various editorial positions, and is a member of The Committee on Stochastic Programming, a standing committee of MPS.

**Elif Kolotoğlu** ("Branch and Tree Decomposition Techniques for Discrete Optimization") is currently a graduate student in the Department of Industrial Engineering at Texas A&M University. She received her B.S. in industrial engineering from Bilkent University in Turkey. Her research interests are in combinatorial optimization, graph theory, and integer programming with an emphasis in network design.

**Arie M. C. A. Koster** ("Branch and Tree Decomposition Techniques for Discrete Optimization") received his Ph.D. from the University of Maastricht in 1999. He is a researcher at the Zuse Institute Berlin (ZIB). His interests are directed toward the application of advanced mathematical optimization techniques, including tree decomposition-based algorithms on real-world operations research problems. Special focuses are on algorithms for computing treewidth and on optimization problems in telecommunication network design (e.g., frequency assignment and optical network planning).

**Robin Lougee-Heimer** ("CBC User Guide") is a research staff member at the IBM T. J. Watson Research Center in Yorktown Heights, New York. She is Cocreator of the Computation Infrastructure for Operations Research (COIN-OR, www.coin-or.or) and a

board member of the educational nonprofit COIN-OR Foundation, Inc. Her research interests include integer programming and using analytics to provide solutions to challenging business problems.

**James E. Matheson** ("Decision Analysis = Decision Engineering") is Chairman and CFO of SmartOrg, Inc., and was awarded the Ramsey Medal in recognition of leadership in the development and application of decision analysis. He is also a fellow of INFORMS and Consulting Professor of Management Science and Engineering at Stanford University. He received a BS from the Carnegie Institute of Technology, and an MS and Ph.D. from Stanford University. He coauthored *The Smart Organization: Creating Value Through Strategic R&D* (Harvard Business School Press).

**Yasemin Merzifonluoğlu** ("Demand Selection and Assignment Problems in Supply Chain Planning") completed her undergraduate studies at Bilkent University, Turkey, in industrial engineering in 2002. She recently received her master's degree in industrial and systems engineering at the University of Florida, where she is currently a Ph.D. student. Her Ph.D. thesis focuses on modeling and solving production planning problems with variable demand and supply capacity. She has also been working on a joint research project with Merck & Co. since 2004.

**Amnon Rapoport** ("Operations Research in Experimental Psychology") is the Karl Eller Professor of Management and Policy in the Eller College of Management at the University of Arizona, and he is an adjunct professor of economics at the Hong Kong University of Science and Technology. He received his Ph.D. from the University of North Carolina in 1965. His research interests include individual and interactive decision making. He has published 5 books, and 200 journal articles and book chapters on psychology, political science, and economics.

**H. Edwin Romeijn** ("Demand Selection and Assignment Problems in Supply Chain Planning") is an associate professor in the Department of Industrial and Systems Engineering at the University of Florida. His research interests are in optimization theory and applications to problems in supply chain management and medicine. He received his Ph.D. in operations research and his M.S. in econometrics from Erasmus University, The Netherlands.

**Güvenç Şahin** ("Network Models in Railroad Planning and Scheduling") obtained BS and MS degrees in industrial engineering from the Middle East Technical University in Ankara, Turkey, in 2000 and 2002, respectively. He is currently pursuing his Ph.D. at the University of Florida. His main area of research is operations research, and his topics of special interest are mathematical modeling, integer programming, logistics network design, and facility location. He has been working on railroad optimization problems for the last three years.

**Javier Salmerón** ("Analyzing the Vulnerability of Critical Infrastructure to Attack and Planning Defenses") is Research Assistant Professor in the Operations Research Department at the Naval Postgraduate School. At NPS since 2000, he teaches linear and nonlinear programming. He has worked on deterministic and stochastic models with applications to civilian and military problems, including railroad freight transportation, electric power generation, resource allocation, and network interdiction.

**Kevin Taaffe** ("Demand Selection and Assignment Problems in Supply Chain Planning") conducts research examining how producers can select demand sources when faced with issues such as constrained production systems and uncertain order requests. In addition to

his interests in optimizing production and inventory management systems, he is engaged in transportation and logistics systems research. He received his Ph.D. in industrial and systems engineering from the University of Florida, and he holds B.S. and M.S. degrees in industrial engineering from the University of Illinois, Urbana-Champaign.

**Kalyan T. Talluri** ("An Introduction to Revenue Management") is an associate professor at the Universitat Pompeu Fabra in Barcelona in the Economics and Business Department. He received his master's from Purdue University and Ph.D. in operations research from MIT (1991). He was a visiting professor at the Kellogg Graduate School of Management of Northwestern University and at INSEAD, Fontainebleau. From 1992 until 1995 he worked at USAir (USAirways) in the areas of scheduling, revenue management, and pricing.

**Garrett J. van Ryzin** ("An Introduction to Revenue Management") is the Paul M. Montrone Professor of Decision, Risk and Operations at the Columbia University Graduate School of Business. He is a leading authority on analytical pricing methods and revenue management. He has published over 30 scientific papers, several book chapters, and is coauthor of the book *The Theory and Practice of Revenue Management* (Kluwer Academic Publishers, 2004), the leading scientific text on revenue management. Garrett's research has been supported by grants from the National Science Foundation and major corporations, and he has served as a consultant to leading companies in the area of revenue management. He received his B.S.E.E. degree (academic honors with distinction) from Columbia University, and his S.M. (electrical engineering and computer science) and Ph.D. (operations research) from MIT.

**Kevin Wood** ("Analyzing the Vulnerability of Critical Infrastructure to Attack and Planning Defenses") is Professor of Operations Research at the Naval Postgraduate School. At NPS since 1982, he has taught courses in networks and optimization and has studied problems of network reliability, deterministic and stochastic mathematical programming, network algorithms, and interdiction. His 1993 paper, "Deterministic Network Interdiction," has spurred interest in the application of bilevel programming to military and civilian problems, including critical-infrastructure protection. ONR, AFOSR, and DOE support his research.