TECHNIQUES FOR INCORPORATING EXPECTED VALUE CONSTRAINTS INTO STOCHASTIC PROGRAMS

A DISSERTATION
SUBMITTED TO THE
DEPARTMENT OF MANAGMENT SCIENCE AND ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
ENGINEERING-ECONOMIC SYSTEMS AND OPERATIONS RESEARCH

By
Marty O'Brien
May 2000

Abstract

In this dissertation, techniques for solving a class of stochastic programs that are characterized by constraints on the expected value of some uncertain quantity are explored. These *expected value constraints* cause the structure of the problem to deviate from the dual angular structure required for using Benders decomposition, so other approaches must be considered.

One approach is to reformulate the problem to return it to a dual angular structure. Second-stage variables that are represented in the expected value constraints are converted to first-stage variables, and the expected value constraint is moved into the master problem. The result is a dual angular problem with many extra first-stage variables. Now classical Benders decomposition can be applied.

In many cases, this is still not sufficient to make the problem practical to solve. Modifications are made to the Benders decomposition algorithm that allow the original first-stage variables to be solved independently of the variables from the expected value constraints, and to simultaneously solve for the variables in the expected value constraints and the other second-stage variables. Subject to some conditions, the computational effort needed to apply the modified algorithm is comparable to the effort needed to solve similar problems without expected value constraints. Implementation of the algorithm is described in the context of DECIS, an existing software implementation of Benders decomposition.

Computation results from sample and real world applications are presented. The thesis concludes with a discussion of other applications for these techniques, and with suggestions for additional research.

Acknowledgements

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview of Stochastic Programming

The object of this thesis is to introduce a new technique for using stochastic programming to perform risk management.

A traditional stochastic programming formulation is a mathematical program in which one or more of the parameters of the problem is not known with certainty at the time the problem is solved. It is assumed that these uncertain parameters are at least known in distribution.

Some examples of multi-stage problems include:

Example 1.1 A fund manager must decide how to invest some money into some stocks in order to maximize his return on investment over the next 6 months. He must make his decisions without knowing exactly how each of the available stocks will perform in the future. /

Example 1.2 (From [6]) The manager of a reservoir must decide every month how much water to release downstream into a river. If he releases too much water and there is too little rainfall in that month, then the reservoir will not be able to meet the water needs of the surrounding community. If he releases too little water and there is too much rainfall that month, the community will be subject to flooding. He must make his decisions with only a set of uncertain weather forecasts to guide him. /

Example 1.3 A poker player in a game of 7-card stud must decide whether to raise (and if so, how much), call, or fold given that his own hand contains a pair of Kings, and he has incomplete information about the hands of his opponents. /

Example 1.4 A store manager must decide how much of each product sold in her store to order each week. If she orders too little of some product, she will have dissatisfied customers who will take their business to another store. If she orders too much, her store will incur higher holding costs. The demand for each product in the next week is uncertain. /

All of these are practical examples of the types of decisions that must be made under uncertainty.

1.1.1 Multi-stage stochastic programs

An important class of stochastic programming problems is the set of *multi-stage problems*. These are problems where some decision variables must be determined before some uncertain parameters are resolved (known with certainty), and some may be set after the uncertain parameters are known with more certainty. A period of time in which a set of parameters is resolved and a set of decision variables is specified are defined as a stage. The last three examples above are all examples of multi-stage problems.

The rest of this thesis concerns itself primarily with *linear* stochastic programs. Linear programs—in which the objective function and constraints on decision variables satisfy certain proportionality, additivity, and continuity assumptions (see Dantzig and Thapa [7], Section 1.2 for the axioms of linear programming)—are powerful enough to describe a great many useful models. These models also contain several desirable features that make them relatively easy to solve:

1. **Convexity**—Linear models are **convex**, assuring that any local optima that are discovered are also global optima.
2. **Concavity**—Linear models are also **concave**, meaning that the optimum solution must lie at an extreme point of the feasible region. For any finite linear program, there is a finite number of extreme points in the feasible set, and therefore a finite search space to find an optimum solution.

3. **Exactness**—Because the search space for the solution to a linear program is finite and discrete, an exact solution can almost always be found—a modeler need not be satisfied with a solution technique that will converge toward the solution at some rate, and terminate before an exact solution is found.

A *two-stage* linear model has the form:

$$\begin{array}{lrcrcl} \text{Minimize} & cx & + & fy & = & z \\ \text{subject to} & Ax & & & = & b \\ & -Bx & + & Dy & = & d \\ & x, & & y & \geq & 0. \end{array} \tag{1.1}$$

Here x and y are the n_1-dimensional vector of first-stage decision variables and the n_2-dimensional vector of second-stage decision variables, respectively. c and f are vectors of the first and second-stage cost coefficients. A is an m_1-by-n_1- dimensional constraint matrix for the first stage, also called the *technology matrix.* b is an m_1-dimensional vector of right-hand side data for the first-stage constraints. For the second stage, D is the m_2-by-n_2-dimensional constraint matrix, and d is the vector of right-hand side data. Finally, B is a *linking matrix* of dimension m_2-by-n_1, and it describes the relationship between the first and second stages of the problem.

To use this model to solve a stochastic program, the second stage is notated to reflect the uncertainty of the second-stage outcomes (which are revealed only after the first-stage decision variables have been set). Here is a formulation of a two-stage stochastic program in which there are 3 equally-likely possible outcomes, and the objective function seeks to minimize the sum of the first-stage costs and the expected value of the second-stage costs:

$$\begin{array}{lrcrcrcrcl} \text{Minimize} & cx & + & \frac{1}{3}f^1y^1 & + & \frac{1}{3}f^2y^2 & + & \frac{1}{3}f^3y^3 & = & z \\ \text{subject to} & Ax & & & & & & & = & b \\ & -B^1x & + & D^1y^1 & & & & & = & d^1 \\ & -B^2x & + & & & D^2y^2 & & & = & d^2 \\ & -B^3x & + & & & & & D^3y^3 & = & d^3 \\ & x, & & y^1, & & y^2, & & y^3 & \geq & 0, \end{array} \tag{1.2}$$

where y^i, f^i, B^i, D^i, and d^i refer to the variables and parameters associated with scenario i, $i \in \{1, 2, 3\}$.

For a general two-stage stochastic program, the formulation is:

$$\begin{array}{lrcrcll} \text{Minimize} & \hat{c}x & + & E_\omega(f^\omega y^\omega) & = & z & \\ \text{subject to} & \hat{A}x & & & = & \hat{b} & \\ & -B^\omega x & + & D^\omega y^\omega & = & d^\omega, & \forall \omega \in \Omega \\ & x, & & y^\omega & \geq & 0 & \forall \omega \in \Omega, \end{array} \tag{1.3}$$

where Ω represents the set of possible outcomes, and y^ω, f^ω, B^ω, D^ω, and d^ω refer to parameters associated with scenario $\omega \in \Omega$. Here the accented symbols $\hat{A}$, $\hat{b}$, and $\hat{c}$ are used to indicate that these parameters are fixed and known with certainty at the time the problem is solved.

The formulation can be further generalized to a *multi-stage* stochastic program:

$$\begin{array}{lrcrcrcccrcl} \text{min} & \hat{c}x & + & E_{\omega_2}(f_2^{\omega_2} y_2^{\omega_2}) & + & E_{\omega_3}(f_3^{\omega_3} y_3^{\omega_3}) & + & \cdots & + & E_{\omega_n}(f_n^{\omega_n} y_n^{\omega_n}) & = & z \\ \text{s/t} & \hat{A}x & & & & & & & & & = & \hat{b} \\ & -B_2^{\omega_2}x & + & D_2^{\omega_2} y_2^{\omega_2} & & & & & & & = & d_2^{\omega_2} \\ & -B_3^{\omega_3}x & - & D_{23}^{\omega_2\omega_3} y_2^{\omega_2} & + & D_3^{\omega_3} y_3^{\omega_3} & & & & & = & d_3^{\omega_3} \\ & \ldots & & & & & & & & & & \\ & -B_n^{\omega_n}x & - & D_{2n}^{\omega_2\omega_n} y_2^{\omega_2} & - & D_{3n}^{\omega_3\omega_n} y_3^{\omega_3} & + & \cdots & + & D_n^{\omega_n} y_n^{\omega_n} & = & d_n^{\omega_n} \\ & x, & & y_2^{\omega_2}, & & y_3^{\omega_3}, & & \cdots, & & y_n^{\omega_n}, & \geq & 0 \\ & & & \forall \omega_2 \in \Omega_2, & & \forall \omega_3 \in \Omega_3, & & \cdots, & & \forall \omega_n \in \Omega_n. & & \end{array} \tag{1.4}$$

Here, subscripts indicate the stage associated with a variable or a parameter, and superscripts refer to a particular scenario in that stage.

1.1.2 Solution of linear programs

Dantzig [7] developed the *simplex method* in 1947 as a way to solve linear programming problems efficiently. It is still the most widely used technique for solving linear programs.

With the first commercial-grade computer implementation by William Orchard-Hays in 1954, the simplex method became a powerful tool for solving linear programs of practical sizes. A number of software packages are available for solving linear programming problems (MINOS [12] and CPLEX [5], for example), and for modeling them with a computer (GAMS [3], AMPL [9]). As computers become more and more powerful, the size and importance of problems that can be solved with the simplex method are also becoming greater.

1.2 Overview of Decomposition Methods

And yet in the realm of stochastic programming, even linear problems can easily become too massive to solve directly. Consider the grocery store manager in Example 1.4. Suppose her store sells 100 different products, and she is planning the orders for the next week. One would expect the two-stage model for this problem to have 100 material balance constraints (one for each product) plus perhaps a store capacity constraint and a budget constraint. There could be on the order of 200 decision variables, corresponding to the order level of each of the products, the inventories of each product at the end of the week, plus some slack variables. So this problem has on the order of 100 variables and 200 constraints—a tiny problem for today's computer hardware and software.

If the manager must consider, say, 10 different demand scenarios for the next week, then the model becomes much larger. It must include a different balance equation for each product under each demand scenario—now the model has $\sim$1000 constraints. Because the final inventories will be different under each scenario, the model needs 1000 more decision variables corresponding to the final inventories of the 100 products under the 10 different scenarios. The 10-scenario stochastic version of this problem now contains 1000 constraints and over 1000 decision variables—about 10 times as large as before and more difficult to solve, but still quite manageable with today's computers.

Now suppose the manager is making plans for a 10-week period instead of a 1-week period, with product orders placed every week, and with 10 independent demand scenarios for each week. Then there are ultimately 10^{10} different possible scenarios that can unfold over the 10-week period, and the equivalent linear programming formulation contains $\sim 10^{12}$ constraints and 10^{12} decision variables (vs. the deterministic formulation of this program, which would have only 100 constraints and 100 decision variables). Problems of this size are well beyond the scope of modern computing power. And if anybody thinks that the current trend of increasingly powerful computers will soon make stochastic programs like this solveable, consider that there exists a real-world truckload freight transportation problem [10] with 10^{3359} scenarios! When discussing how to solve problems of this size, someone invariably speculates about whether the problem could be solved if every atom in the universe were a supercomputer and had been working on the problem since the Big Bang.

Decomposition refers to the strategy of breaking up a large, difficult-to-solve problem into two or more smaller, easier-to-solve problems, such that the solutions to the decomposed problems can be used to obtain the solution to the larger problem. Most stochastic programs lend themselves well to some decomposition method.

1.2.1 Benders Decomposition

Benders decomposition was derived in 1962 by J.F. Benders [1] as a technique for solving mixed integer programs. Van Slyke and Wets [14] realized in 1969 that Benders decomposition could be applied to large stochastic programs with a *dual angular structure* (as depicted in Figure 1.1), and they introduced what is called the *L-shaped method* to obtain exact solutions for these types of problems. In a dual angular structured problem, each second-stage decision variables is applicable to exactly one scenario.

A stochastic model with the dual angular "staircase" structure

$$\begin{array}{lrclcll} \text{Minimize} & cx & + & \sum_\omega f^\omega y^\omega & = & z & \\ \text{subject to} & Ax & & & = & b & \\ & -B^\omega x & + & D^\omega y^\omega & = & d^\omega & \forall\,\omega\in\Omega \\ & x, & & y^\omega & \geq & 0 & \forall\,\omega\in\Omega, \end{array} \tag{1.5}$$

is decomposed into a **master problem** and $|\Omega|$ independent **subproblems**. The form of the master problem is:

$$\begin{array}{lrclcl} \text{Minimize} & cx & + & \theta & = & z \\ \text{subject to} & Ax & & & = & b \\ & -Gx & + & \theta e & \geq & g \\ & -G'x & & & \geq & g' \\ & x & & & \geq & 0, \end{array} \tag{1.6}$$

where θ represents an estimate of the second-stage costs. The constraints $-Gx + \theta e \geq g$ and $-G'x \geq g'$ are *cuts* that describe the optimal objective value of the subproblems. The construction of $G, G', g,$ and g' is described below.

The independent subproblems, one for each $\omega \in \Omega$, look like:

$$\begin{array}{lrcll} \text{Minimize} & f^\omega y^\omega & = & Q(x,\omega) & \\ \text{subject to} & D^\omega y^\omega & = & d^\omega + B^\omega x & :\pi^\omega \\ & y^\omega & \geq & 0. & \end{array} \tag{1.7}$$

Here, x is not a set of decision variables, but a fixed vector of parameters specified by the solution to the master problem. $Q(x,\omega)$ is referred to as the *recourse function*, and represents the second-stage costs given some first-stage decision x and a second-stage scenario $\omega \in \Omega$. π^ω represents the marginal costs associated with the constraints of the subproblem. π^ω may also be obtained by solving the dual of the subproblem

(1.7):

$$\begin{array}{lccl} \text{Maximize} & \pi^\omega(d^\omega + B^\omega x) & = & Q(x,\omega) \\ \text{subject to} & \pi^\omega D^\omega & \geq & f^\omega. \end{array} \tag{1.8}$$

The procedure for using Benders decomposition is to solve the master problem and the subproblems iteratively. A solution x to the master problem is passed to all of the subproblems, and second-stage costs $Q(x,\omega)$ are computed for all $\omega \in \Omega$. If all of the subproblems have feasible solutions, then an *optimality cut* of the form

$$E_\omega[\pi^\omega(d^\omega + B^\omega x)] \leq \theta \tag{1.9}$$

is added to the master problem. This cut contributes to the *outer linearization* of the recourse function, help approximate second-stage costs in the master problem as a function of x, and produce a different value for x the next time the master problem is solved. If one of the subproblems is infeasible, then a *feasibility cut* with the form

$$\pi^\omega(d^\omega + B^\omega x) \leq 0 \tag{1.10}$$

is added to the problem. This constraint prevents the master problem from producing further solutions that may lead to infeasible subproblems.

At each iteration of the problem, the objective value of the solution to the master problem $c\hat{x} + \hat{\theta}$ provides a lower bound on the final optimal solution, since it is a relaxation of the full problem. Assuming there are feasible solutions to all of the subproblems, the objective function value of those solutions $c\hat{x} + E_\omega[f^\omega \hat{y}^\omega]$ is an upper bound, since it represents a feasible solution to the full problem. The algorithm proceeds until it converges to an optimal solution.

1.2.2 Dantzig-Wolfe Decomposition

Another useful type of decomposition is *Dantzig-Wolfe decomposition*, developed by Dantzig and Wolfe [8] in 1957. Dantzig-Wolfe decomposition is related to Benders decomposition in that it is equivalent to performing Benders decomposition on the

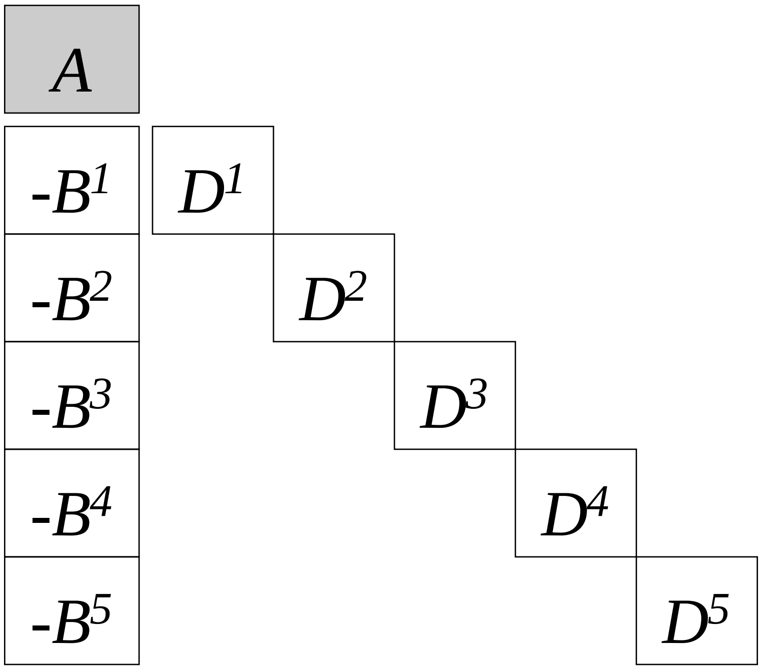

Figure 1.1: *Dual angular* structure of a two-stage stochastic program.

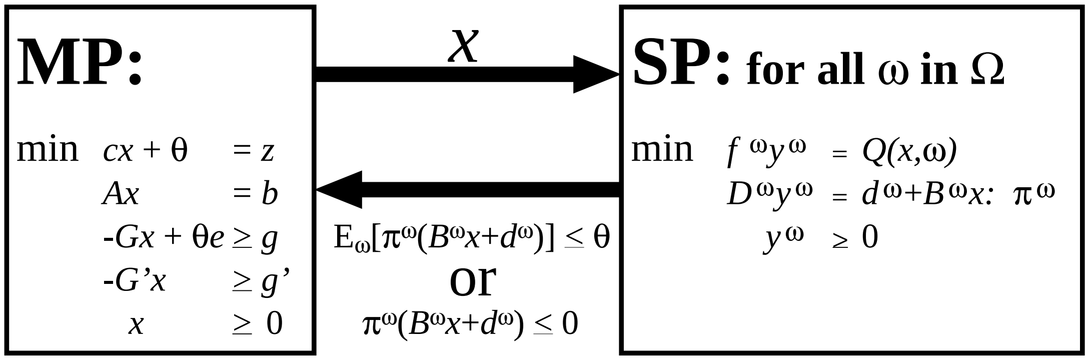

Figure 1.2: Description of Benders decomposition algorithm, showing flow of information between master problem and subproblems.

Investment option	"good" payoff	"bad" payoff	"ugly" payoff	expected payoff
high risk	150%	-10%	-95%	15%
low risk	15%	5%	-5%	5%
no risk	3%	3%	3%	3%

Table 1.1: Payoffs for the fund manager risk management examples (1.5-1.13).

dual of some linear program. As Benders decomposition is an iterative procedure in which a new row is added to the master problem after every iteration, Dantzig-Wolfe decomposition is an iterative procedure in which a new *column* is added to the master problem after every iteration. Dantzig-Wolfe decomposition can be applied to problems with *block angular* structure, as shown in Figure 1.3. See Dantzig and Thapa [7] for a deeper treatment of Dantzig-Wolfe decomposition.

1.3 Overview of Risk Management Techniques

A stochastic programming formulation such as model (1.3) or (1.4) allows for uncertain factors to be incorporated into the decision. However there are situations where one would want to use a more explicit consideration of risk in the formulation. Consider Example 1.1, the problem of the fund manager.

Example 1.5 The fund manager has three choices about how to invest $1000. He may invest it in a volatile, high-return stock, in a lower-return, lower-risk stock, or in a no-risk money market account. He considers three scenarios, denoted as "good", "bad", and "ugly", which he assumes are all equally likely to occur. The 6-month payoffs of the three options under those three scenarios are given in Table 1.1.

The model for the fund manager's problem can thus be formulated as:

$$
\begin{array}{lrcrcrcll}
\text{Max} & & & f(y^{good}, y^{bad}, y^{ugly}) & & & = & z & \\
\text{s.t.} & x_{high} & + & x_{low} & + & x_{no} & = & 1000 & \text{(budget constraint)} \\
& 1.50\,x_{high} & + & 0.15\,x_{low} & + & 0.03\,x_{no} & = & y^{good} & \text{(“good” payoff)} \\
& -0.10\,x_{high} & + & 0.05\,x_{low} & + & 0.03\,x_{no} & = & y^{bad} & \text{(“bad” payoff)} \\
& -0.95\,x_{high} & - & 0.05\,x_{low} & + & 0.03\,x_{no} & = & y^{ugly} & \text{(“ugly” payoff)} \\
& x_{high}, & & x_{low}, & & x_{no} & \geq & 0, &
\end{array}
\tag{1.11}
$$

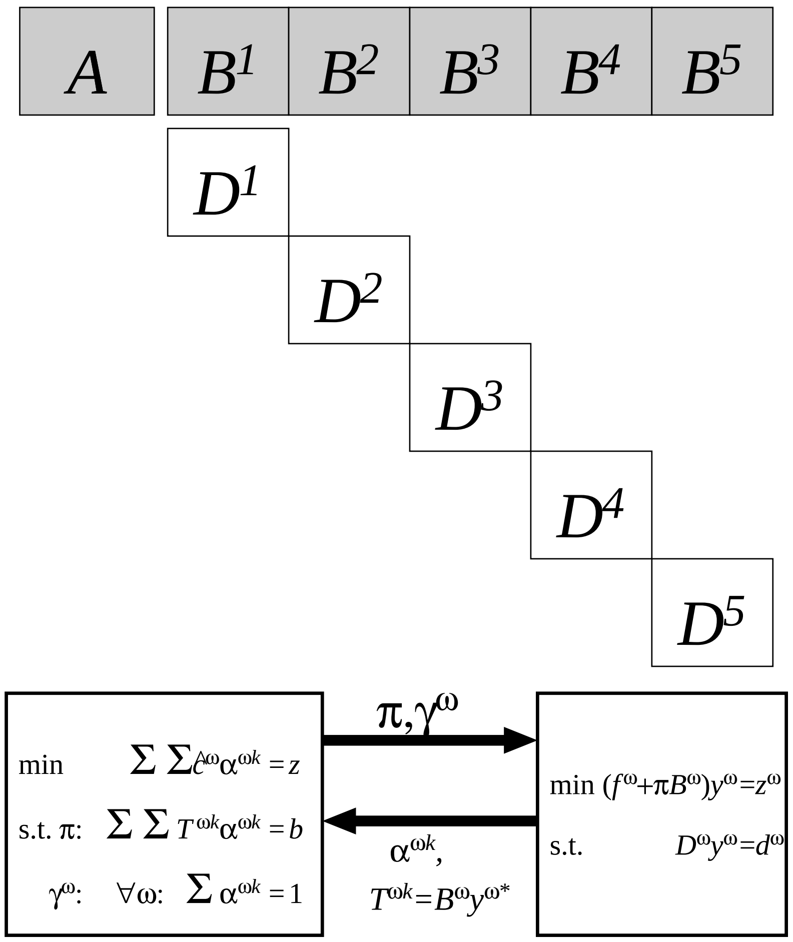

Figure 1.3: Top: Desirable structure for Dantzig-Wolfe decomposition. Bottom: Description of Dantzig-Wolfe decomposition, showing flow of information between master program and subprograms.

where the objective function is unspecified for now. /

If the objective of the fund manager is simply to maximize expected return, the objective function for the model should be

$$\text{Maximize} \quad z = \frac{1}{3}y^{good} + \frac{1}{3}y^{bad} + \frac{1}{3}y^{ugly}. \tag{1.12}$$

Given this objective function, the optimal solution is to invest all \$1000 of his funds into the high-risk stock—the investment with the greatest expected return. But note that for this solution, the manager has a $\frac{2}{3}$ probability of losing money, and a $\frac{1}{3}$ probability of losing 95% of his funds. Depending on the manager's tolerance for risk, those could be the characteristics of a very bad solution.

This example should be kept in mind as the following approaches to risk management are considered.

1.3.1 Adjusting the Objective Function

Risk may be accounted for explicitly in the objective function in several ways.

An expression that models the risk can be explicitly added to the objective function to penalize solutions that have too much risk, that is, outcomes with large deviations from the "average" outcome for a given solution.

Example 1.6 Quantify the risk for a particular solution and a particular scenario ω as the deviation from the average outcome of the outcome of that solution in that scenario. That is, let

$$r^{\omega} = \left| \frac{1}{3}(y^{good} + y^{bad} + y^{ugly}) - y^{\omega} \right| \tag{1.13}$$

for $\Omega = \{\text{good}, \text{bad}, \text{ugly}\}$. The objective function is adjusted to include this measure of risk:

$$\text{Maximize} \quad z = \alpha E_{\omega} y^{\omega} - (1-\alpha) E_{\omega} r^{\omega}, \tag{1.14}$$

where α is a number between 0 and 1 that reflects the relative importance of the objective[1] of maximizing return with respect to the objective of reducing risk, as it has been defined.
For $\alpha = 0$, that is, for the case where we are minimizing risk, the optimal solution is to invest all funds into the no-risk money market account for a guaranteed return of 3%.
For $\alpha = \frac{1}{2}$, where the objectives of maximizing return and minimizing risk are balanced, the optimal solution turns out to be to invest all funds into the low-risk stock. /

An extension to this approach is to just model the "downside risk"—outcomes that fall below the expected outcome.

Example 1.7 Define the downside risk for a scenario as

$$r'^{\omega} = \max[0, y^{\omega} - \frac{1}{3}y^{good} + y^{bad} + y^{ugly}]. \tag{1.15}$$

Now define the objective function as

$$\text{Maximize} \quad \alpha E_{\omega} y^{\omega} - (1-\alpha) E_{\omega} r'^{\omega} = z, \qquad \forall \omega \in \{\text{good}, \text{bad}, \text{ugly}\}, \tag{1.16}$$

where again α is a number between 0 and 1 reflecting the relative importance of the goals of maximizing return and minimizing downside risk.
Solving[2] for $\alpha = 0$ and $\alpha = \frac{1}{2}$, again the resulting solutions are to invest all funds in the no-risk money market account, and to invest all funds into the low-risk stock, respectively. /

Alternatively, the recourse function need not compute the expected value of the second-stage costs. The recourse function could be biased to reward solutions with low downsides. It may be helpful to have some prior knowledge about the problem to determine how the biasing should be accomplished.

Example 1.8 It is seen by inspection that the "ugly" scenario has the most downside risk associated with it. The "bad" scenario can potentially have some downside risk,

[1]Use of the absolute value sign in (1.13) may appear to make this a nonlinear formulation of the problem, but in this case, the model can be transformed into an equivalent linear formulation [7]. However, there would not be a valid transformation if the objective is to *maximize* risk.

[2]Again, the absolute value signs can be transformed away into an easy-to-solve linear model.

too. Instead of using (1.12) as the objective function, something like

$$\text{Maximize} \quad z = \frac{3}{12}y^{good} + \frac{4}{12}y^{bad} + \frac{5}{12}y^{ugly} \tag{1.17}$$

can be used. The solution to the model with this objective function is to invest all funds into the low-risk stock.

Alternatively, an objective function with even more bias can be used, such as

$$\text{Maximize} \quad z = \frac{1}{6}y^{good} + \frac{2}{6}y^{bad} + \frac{3}{6}y^{ugly}. \tag{1.18}$$

The solution to the model with this objective function is to invest all funds into the no-risk money market account. /

The objective function can be replaced by a *utility function*, which can better model the user's risk preferences. With a little ingenuity, a piecewise linear function can approximate many different types of utility functions, and the model is still be a linear program.

Example 1.9 Suppose that the funds being managed in the earlier examples were from the accounts of several widows and orphans. The fund manager has promised them a decent rate of return on their investment, but he is loath to expose them to a lot of risk. So he uses a utility function of

$$u(y) = 1 - \exp(-\frac{y}{500}) \tag{1.19}$$

with the linear approximation

$$\begin{aligned}
u^\omega &\leq 1.0585\frac{y^\omega}{500} + 0.3389 \\
u^\omega &\leq 0.6420\frac{y^\omega}{500} + 0.0274 \\
u^\omega &\leq 0.3894\frac{y^\omega}{500} + 0.0202 \\
u^\omega &\leq 0.2362\frac{y^\omega}{500} + 0.1356 \\
u^\omega &\leq 0.1433\frac{y^\omega}{500} + 0.2857 \\
y^\omega &\quad \text{free},
\end{aligned}$$

where y^ω is the return on the investment of \$1000 under scenario ω. Figure 1.4 demonstrates that this piecewise linear approximation of the utility function is fairly accurate and should be good enough for the fund manager's purposes. The fund manager now wishes to maximize expected utility of his return, rather than just maximizing the expected return.

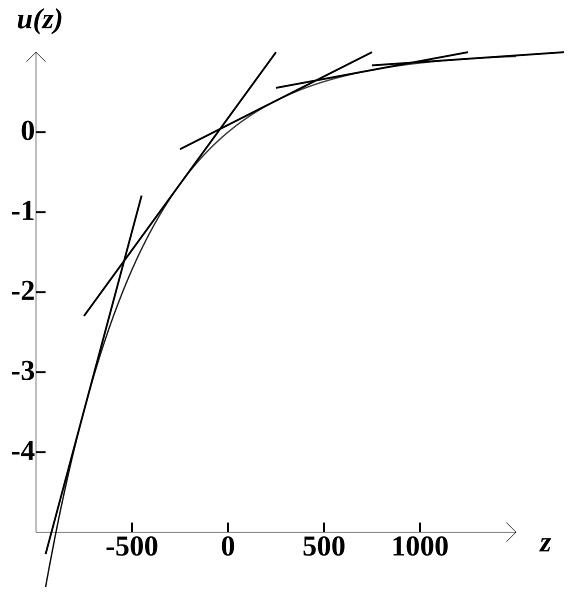

Figure 1.4: Utility function and its piecewise-linear approximation for Example 1.9.

In this example, the solution achieved by maximizing $E_\omega u^\omega$ is to invest \$887 into the low-risk stock, and the remaining \$113 into the no-risk money market account. /

1.3.2 Adjusting the Constraints

Risk can also be accounted for in the constraints of a stochastic program. Once an appropriate risk measure is determined and modeled, a constraint, which puts a bound on the risk measure can be added to the model.

Example 1.10 Recall that in the original model without risk management, the optimal solution was to invest all funds into the high-risk stock. The expected risk of this solution, using the definition in (1.13), was \$900. In order to perform risk management, the fund manager may wish to put an explicit upper bound on this measure of risk, say \$400. The constraint

$$r^{\omega} \leq 400 \quad \forall \omega \in \{\text{good}, \text{bad}, \text{ugly}\} \tag{1.20}$$

can be added to the original model[3] (1.11). The resulting optimal solution is to invest $\$166\frac{2}{3}$ into the high-risk stock, and $\$833\frac{1}{3}$ into the low-risk stock. /

Alternatively, downside risk, as opposed to total risk, can be constrained.

Example 1.11 Using the definition of downside risk in equation (1.15), the expected downside risk for the original optimal solution (1.11) is \$450. A constraint like

$$r'^{\omega} \leq 200 \quad \forall \omega \in \{\text{good}, \text{bad}, \text{ugly}\} \tag{1.21}$$

could be added to the original model. The resulting optimal solution is to invest $\$190\frac{10}{21}$ into the high-risk stock, with the remaining $\$809\frac{11}{21}$ being invested in the low-risk stock. /

Another approach is to add constraints based on the modeler's knowledge or insight about the problem of interest. With these contrived constraints, the modeler may be able to remove some of the more risky solutions from the feasible set.

Example 1.12 The fund manager knows from solving his problem without risk management that the high-risk stock is the riskiest investment available to him. So in order to reduce the risk of his investments, he restricts his investment in that stock to no more than 30% of his funds, or \$300.
He adds the constraint

$$x^{high} \leq 300 \tag{1.22}$$

or

$$x^{high} \leq 0.30(x^{high} + x^{low} + x^{no}) \tag{1.23}$$

to his original model (1.11). The resulting solution is to invest 30% of his funds (\$300) into the high-risk stock, and the remainder of his funds (\$700) into the next best investment, the low-risk stock. /

[3]Again, the problem can still be solved as a linear program if the constraint is an *upper bound* on risk, but would be problematic if it were a *lower bound* on risk.

Combinations of the above approaches can also be applied. For example, a manager may wish to adjust his model by playing with the cost coefficients in the objective function while also putting constraints on downside risk.

It should be noted in passing that in the preceding examples, when risk management approaches are focused on the objective function, the resulting solutions still tend to be extreme—the optimal solution was to invest all funds into one type of investment. When risk management is handled through constraints on the problem, the result is more diversified solutions—the optimal solution has the fund manager dividing his funds between two different investment options. This observation can be explained by the effect of the risk management techniques on the problem's feasible region. Without any risk management incorporated into the problem, the extreme points of the feasible region tend to be "extreme" solutions—in the case of the portfolio optimization problem, the extreme points of the feasible set correspond to allocating all resources into one type of investment. When a constraint is added to the problem, the feasible region becomes smaller, and the set of the extreme points of the feasible region changes. Now there is an opportunity for a diversified solution to be an optimal solution to the problem. If no constraints are added to the problem, but the linear objective function is modified, then the feasible region is not affected, and the optimal solution to the problem must still be one of the original "extreme" solutions.

It should also be stressed at this point that none of the above approaches to risk management is the best in all situations. The best approach to use depends on many factors, including the additional complexity that each approach adds to the existing model and the specific risk management needs of the user.

1.3.3 Chance-Constrained Programming

Charnes and Cooper [4] developed an approach they called *chance-constrained programming*, where one or more of the constraints are satisfied with some specified probability. Probabilistic methods are covered in Prékopa [13].

In general these problems require either some sort of nonlinear optimization, or some specialized analysis of the problem that is highly dependent on the particular probability distribution used. Both of these add an additional layer of complexity to the problem. Such complexity is beyond the scope of this thesis, so this approach will not be analyzed further.

1.4 Expected Value Constraints

In all of the above approaches to risk management, one is able to take a multi-stage stochastic program with a staircase structure (see Figure 1.1) and transform it into a problem that retains that staircase structure. This assures that any decomposition methods that could be used to solve the original problem can also be used to solve the risk-managed problem.

In Example 1.10, a constraint of the form

$$r^{\omega} \leq \hat{R} \tag{1.24}$$

was added to the model (1.11). This constraint places a bound on the risk *for each scenario*—that is, in every scenario, the risk is guaranteed ot to exceed some level—and does not change the dual angular structure of the underlying model (see Figure 1.5).

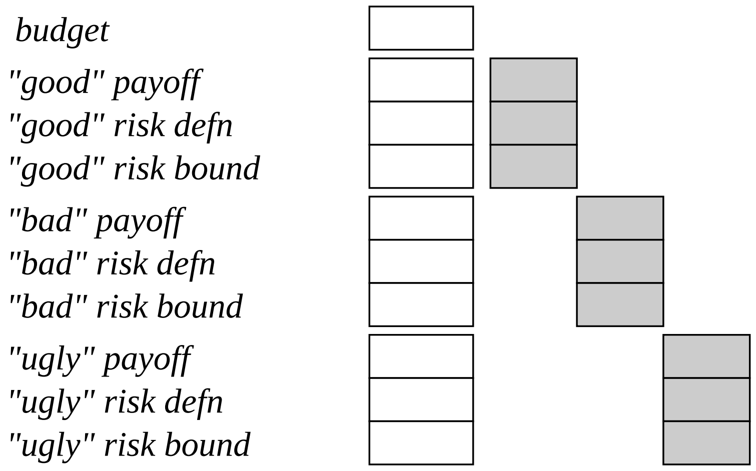

Figure 1.5: Structure of the fund manager problem (Example 1.10).

Let us consider another approach to managing risk. A single upper bound is put on the *expected risk*, rather than a separate bound on the risk of each scenario. That is, the risk associated with some scenarios may exceed the bound so long as the *average* risk does not exceed the bound. The form of this constraint is

$$E_\omega r^\omega = \frac{1}{3}r^{good} + \frac{1}{3}r^{bad} + \frac{1}{3}r^{ugly} \leq \hat{R}. \tag{1.25}$$

Example 1.13 The optimal solution of the original model (1.11) with the expected value constraint (1.25) with a value of $\hat{R} = \$200$ yields an optimal solution of investing $\$111\frac{1}{9}$ in the high-risk stock and $\$888\frac{8}{9}$ in the low-risk stock, with risk values of $\$116\frac{2}{3}$ in the "good" scenario, \$150 in the "bad" scenario, and $\$333\frac{1}{3}$ in the "ugly" scenario, for a total expected risk of \$200. /

A constraint of type (1.25) is called an *expected value constraint*, because it describes a bound on the expected value of some quantity.[4] The main features of the expected value constraint are that it encapsulates second-stage decision variables from more than one scenario into a single constraint, and that it wrecks the dual angular structure of the problem.

In principle, the model in Example 1.13 with constraint (1.25) is not any harder to solve than the model in Example 1.10 with constraint (1.24). In fact, the case could be made that the formulation of Example 1.13 is easier to solve than the formulation of Example 1.10, since the latter formulation adds 3 new constraints and 6 new nonzero elements to the problem, whereas the expected value constraint formulation adds only 1 constraint and 4 nonzero elements. But this alternative formulation does change the structure of the problem into one that does not have the staircase form represented in Figure 1.1. This problem, as shown in Figure 1.6, has one constraint that is relevant for every scenario. If this problem were large enough that Benders

[4]The term "expected value" suggests that the coefficient for each second-stage variable in an expected value constraint is related to the probability associated with that variables scenario, but this need not be the case. For example, one may be applying importance sampling, and assign higher weights to scenarios that are deemed to be "interesting". More formally and without loss of generality, the label *expected value constraint* is applied to any constraint with nonzero coefficients on second-stage variables from two or more scenarios.

decomposition might be needed to make the problem more manageable, then it could not be applied.

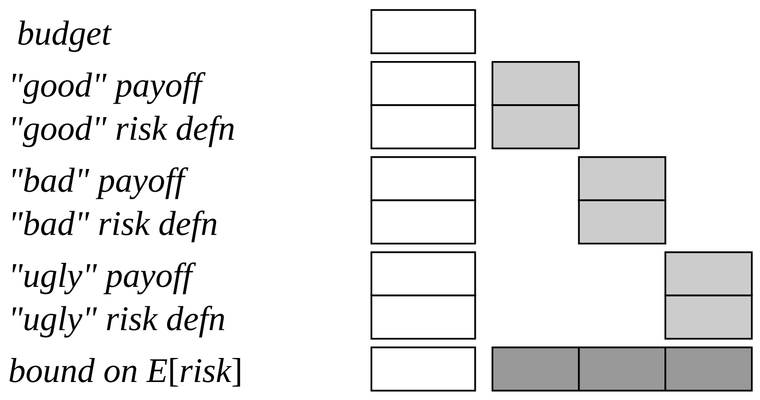

Figure 1.6: Structure of fund manager problem with expected value constraints (Example 1.13).

There is no single approach to risk management that is appropriate for all cases. The advantages and disadvantages of each risk management technique should be weighed before incorporating the approach into a problem. At this point it would appear that the approach using expected value constraints that was just introduced has a serious disadvantage in that it can not be used with Benders decomposition.

The rest of this thesis is organized as follows: Chapter 2 introduces an example and motivation for using expected value constraints—the advantages of that approach, if you will. A transformation is described to convert a two-stage problem with expected value constraints back into a problem with the desirable staircase structure, allowing Benders decomposition to be used to solve the problem. In Chapter 3, another algorithm is introduced with features that allow problems with expected value constraints to be solved efficiently, provided that certain conditions are satisfied. In many practical cases, it is demonstrated that problems with expected value constraints are not much more difficult to solve than similar stochastic programs without expected value constraints. Finally, Chapter 4 deals briefly with additional topics such as more generalized expected value constrained problems.

Chapter 2

Expected Value Constraints in Two-Stage Problems

This thesis is motivated by a supply-chain model and risk management technique in use at Hewlett-Packard (HP) Labs in Palo Alto, California. A simplified version of this model will be presented along with approaches used by HP for risk management. The drawbacks of those approaches will be shown for this application, along with the benefits of using expected value constraints. Discussion of how to solve stochastic programs with expected value constraints commences with a description of the intuitive structure of such a problem and continues with a reformulation of the problem. Numerical results of problems with the various risk management techniques will be presented.

2.1 Example and Motivation

2.1.1 Description of Hewlett-Packard Supply Chain Problem

Hewlett-Packard (HP) is involved in the production and sale of a wide variety of products with leading-edge technology and applications. HP was concerned about how to use components that were at the end of their life cycle. Components are said to be at the end of their life cycle when they are made obsolete by the introduction of better (newer, cheaper, etc.) components that perform the same function, or when the supplier discontinues the supply of that component. Once components are identified as being near the end of their life cycle, there is a relatively small window of opportunity to use those components by building them into a saleable product. If the opportunity is missed, HP will never have a chance to use those components in any future product, and will incur further holding or disposal costs.

Components are only "useful" to HP when they are built into finished products and sold to customers. If a component at the end of its life cycle can not become part of some product, or if that product is not ultimately sold to a customer, then HP must dispose of that component, receiving only a fraction of the previous value of the component, or perhaps even incurring some holding or disposal cost.

Many components are used in several different products. For example, a particular plastic case might be used for several different models of computers, or the same size bolt might be used in a disk drive, a cooling fan, and a laser printer. HP must decide which products to build in order to maximize consumption of their excess inventory of these special components.

2.1.2 Hewlett-Packard Supply Chain Problem Formulation

A **component** is any unit of hardware that can be quantified and stored in inventory. A component may be a discrete entity (a screw), it may be a **sub-assembly** that is composed of several smaller components (a cooling fan), or it may be an **end product** that may be sold to a customer (a laser printer).

Some components are produced at HP; other components are assembled at HP from other components; still other components are purchased from suppliers outside of HP. Assembled components are made in accordance with a **bill of materials** that describes how many of each component is used in order to assemble another component. Components that are purchased from outside suppliers have a **cost** to obtain those components. Components that are produced at HP have a **make cost** incurred during production. Every component also has a **sunk cost**, representing the total amount invested in order to obtain that component, and a **scrap value**, the value received when a component is **scrapped**, or disposed of without being incorporated into a saleable product. End products also have a **revenue** associated with their sale to a customer, and a stochastic **demand** that indicates how many products can be sold after a production run. Demands for products are realized from a (possibly infinite) set of outcomes Ω.

Some components have long **lead times**—the difference between the time that a component is ordered and the time that it is incorporated into a product. Often the process for procuring components with long lead times must begin long before product demand is realized, and long before HP will know how many of those components it will need to produce to satisfy demand. The problem is formulated as a two-stage problem by identifying those components with long lead times and requiring them to be procured before the demands are realized. That is, the purchase and assembly quantities of components with long lead times are the first-stage decision variables, and the purchase and assembly quantities of components with short lead times are second-stage variables. The amount of each end product to sell and the amount of each component to scrap are also second-stage decision variables.

Sets:

$\mathtt{I}$	set of all components
$\mathtt{B} \subseteq \mathtt{I}$	set of all components that can be **bought** by HP
$\mathtt{B}_1 \subseteq \mathtt{B}$	components that may be bought in the first-stage
$\mathtt{B}_2 \subseteq \mathtt{B}$	components that may be bought in the second-stage
$\mathtt{M} \subseteq \mathtt{I}$	set of all components that can be **made** by HP
$\mathtt{M}_1 \subseteq \mathtt{M}$	components that may be made in the first-stage
$\mathtt{M}_2 \subseteq \mathtt{M}$	components that may be made in the second-stage
$\mathtt{S} \subseteq \mathtt{I}$	set of end products, components that can be **sold**

Parameters:

$\hat{c}_j$	**cost** of procuring 1 unit of component $j \in \mathtt{B}$

$\hat{a}_j$	**make cost** of assembling 1 unit of component $j \in \mathtt{M}$
$\hat{e}_j$	initial **excess inventory** of component $j \in \mathtt{I}$
$\hat{v}_j$	**sunk cost** of each unit of component $j \in \mathtt{I}$
$\hat{q}_j$	**scrap value** of a unit of component $j \in \mathtt{I}$
$\hat{r}_j$	**revenue** for selling a unit of component $j \in \mathtt{S}$
$\hat{\alpha}_{ij}$	units of component $i \in \mathtt{I}$ needed to assemble one unit of $j \in \mathtt{M}$
$\tilde{d}_j$	demand for product $j \in \mathtt{S}$
p^ω	probability that scenario $\omega \in \Omega$ will be realized, $\sum_{\omega\in\Omega} p^\omega = 1$
γ_B	relative importance of procurement spending in the objective
γ_P	relative importance of profit in the objective
γ_R	relative importance of inventory reduction in the objective

A "hat" on a coefficient (e.g., $\hat{c}$) is used to denote that a coefficient is known with certainty at the time the problem is solved. A "tilde" over a coefficient (e.g., $\tilde{d}$) is used to indicate that a coefficient is known in distribution only at the time the problem is solved.

Decision variables:

b_{1j}	units of component $j \in \mathtt{B}_1$ bought in stage 1
m_{1j}	units of component $j \in \mathtt{M}_1$ made in stage 1
b_{2j}	units of component $j \in \mathtt{B}_2$ bought in stage 2
m_{2j}	units of component $j \in \mathtt{M}_2$ made in stage 2
s_{2j}	units of component $j \in \mathtt{S}$ sold in stage 2
w_{2j}	units of component $j \in \mathtt{I}$ scrapped in stage 2

The important constraints of this model are **market constraints** and **balance constraints**. Market constraints state that the number of end product components that are sold cannot exceed the demand for that product. That is,

$$s_{2j}^\omega \le \tilde{d}_j^\omega, \quad \forall\, \omega \in \Omega, j \in \mathtt{S}, \tag{2.1}$$

with the assumption that $\tilde{d}_j = 0$ for $j \notin \mathsf{S}$.

Balance constraints keep track of the first- and second-stage inventory, and assure that there is a nonnegative number of each component at all times. A component begins with $\hat{e}_j$ units in inventory. $m_{1j} + m_{2j} + b_{1j} + b_{2j}$ units of component j are acquired in the first and second-stages. $\sum_{i \in \mathtt{I}} \hat{\alpha}_{ji}(m_{1i} + m_{2i})$ units of component j are consumed in order to make other products. Finally, all components that are not consumed in the assembly of other products must ultimately either be sold (s_{2j} if $j \in \mathsf{S}$) or scrapped (w_{2j}). The second-stage balance constraint thus looks like:

$$\hat{e}_j + b_{1j} + m_{1j} + b^{\omega}_{2j} + m^{\omega}_{2j} = \sum_{i \in \mathsf{M}} \hat{\alpha}_{ji}(m_{1i} + m^{\omega}_{2i}) + s^{\omega}_{2j} + w^{\omega}_{j}, \qquad \omega \in \Omega, j \in \mathtt{I} \tag{2.2}$$

The first-stage balance constraint just seeks to assure that a nonnegative amount of each component is available at the end of the first-stage:

$$\hat{e}_j + b_{1j} + m_{1j} \geq \sum_{i \in \mathsf{M}} \hat{\alpha}_{ji} m_{1i}. \tag{2.3}$$

That is, the amount of component j consumed in making other components cannot exceed the initial inventory plus the amount that is either purchased or assembled during the first-stage.

Finally, there are three measureable *aggregate quantities* of interest in the model.

1. **Total profit**: HP is sometimes interested in the profit that can be made from their production run, and is tracked separately in each stage of the problem. Costs are incurred in two ways—from buying components from outside suppliers, and from production costs associated with assembling components at HP. The primary source of revenues is the sale of end products to customers, but HP can also receive the scrap value associated with components that cannot be sold.

$$P_1 = -\sum_{j \in \mathsf{B}_1} \hat{c}_j b_{1j} - \sum_{j \in \mathsf{M}_1} \hat{a}_j m_{1j} \tag{2.4}$$

$$P_2 = -\sum_{j \in \mathtt{B}_2} \hat{c}_j b_{2j} - \sum_{j \in \mathtt{M}_2} \hat{a}_j m_{2j} + \sum_{j \in \mathtt{S}} \hat{r}_j s_{2j} + \sum_{j \in \mathtt{I}} \hat{q}_j w_{2j}. \tag{2.5}$$

2. **Total spending**: HP is interested in the costs associated with their production run. Costs are incurred in two ways—from buying components from outside suppliers, and from production costs associated with assembling components at HP. Spending is tracked separately in both stages of the problem.

$$B_1 = \sum_{j \in \mathtt{B}_1} \hat{c}_j b_{1j} + \sum_{j \in \mathtt{M}_1} \hat{a}_j m_{1j} \tag{2.6}$$

and

$$B_2 = \sum_{j \in \mathtt{B}_2} \hat{c}_j b_{2j} + \sum_{j \in \mathtt{M}_2} \hat{a}_j m_{2j} \tag{2.7}$$

describe the procurement spending for the two stages of the problem.

3. **Inventory reduction**: HP wants to track how much of its inventory it can remove from the shelves and incorporate into saleable products. Since every component in inventory is either incorporated into a product and sold, or scrapped, the total inventory reduction can be measured by the expression

$$R_2 = \sum_{j \in \mathtt{I}} \hat{v}_j (\hat{e}_j - w_{2j}). \tag{2.8}$$

There is no meaningful expression for first-stage inventory reduction, since no components can be disposed of until the second-stage.

Any of these quantities could either be the focus of optimization (and accounted for in the objective function), constrained by the model (accounted for in an upper bound or lower bound), or simply measured.

The final form of the model is:

$$
\begin{array}{lrcl}
\text{Minimize} & \gamma_B(B_1 + E_\omega B_2^\omega) - \gamma_P(P_1 + E_\omega P_2^\omega) & & \\
 & -\gamma_R(E_\omega R_2^\omega) & = & z \\
\text{subject to} & & & \\
\quad\text{1st-stage budget defn.} & B_1 - \sum_{j\in \mathtt{B}_1} \hat{c}_j b_{1j} - \sum_{j\in \mathtt{M}_1} \hat{a}_j m_{1j} & = & 0 \\
\quad\text{1st-stage profit defn.} & P_1 + \sum_{j\in \mathtt{B}_1} \hat{c}_j b_{1j} + \sum_{j\in \mathtt{M}_1} \hat{a}_j m_{1j} & = & 0 \\
\quad\text{1st-stage balance} & \sum_{i\in \mathtt{I}} \hat{\alpha}_{ji} m_{1i} - m_{1j} - b_{1j} & \leq & \hat{e}_j \\
\text{And for all } \omega \in \Omega: & & & \\
\quad\text{2nd-stage budget defn.} & B_2^\omega - \sum_{j\in \mathtt{B}_2} \hat{c}_j b_{2j}^\omega - \sum_{j\in \mathtt{M}_2} \hat{a}_j m_{2j}^\omega & = & 0 \\
\quad\text{2nd-stage profit defn.} & P_2^\omega + \sum_{j\in \mathtt{B}_2} \hat{c}_j b_{2j}^\omega + \sum_{j\in \mathtt{M}_2} \hat{a}_j m_{2j}^\omega & & \\
 & - \sum_{j\in \mathtt{S}} \hat{r}_j s_{2j}^\omega - \sum_{j\in \mathtt{I}} \hat{q}_j w_{2j}^\omega & = & 0 \\
\quad\text{reduction definition} & R_2^\omega - \sum_{j\in \mathtt{I}} \hat{v}_j w_{2j}^\omega & = & \sum_{j\in \mathtt{I}} \hat{v}_j \hat{e}_j \\
\quad\text{2nd-stage balance} & w_{2j}^\omega + s_{2j}^\omega + \sum_{i\in \mathtt{I}} \hat{\alpha}_{ij} m_{2i}^\omega - m_{2i}^\omega - b_{2i}^\omega & = & \hat{e}_j + b_{1j} + m_{1j} \\
 & & & - \sum_{i\in \mathtt{I}} \hat{\alpha}_{ij} m_{1i} \\
\quad\text{demand constraint} & s_{2j}^\omega & \leq & \tilde{d}^\omega \\
 & b_{1j}, m_{1j}, b_{2j}^\omega, m_{2j}^\omega, s_{2j}^\omega, w_{2j}^\omega & \geq & 0 \\
 & & & \forall\, j \in \mathtt{I}, \omega \in \Omega,
\end{array}
$$

with potentially one or more of the following constraints for risk-management:

$$B_1 + B_2^\omega \quad \leq \quad \hat{B} \qquad \forall\, \omega \in \Omega \tag{2.9}$$

$$P_1 + P_2^\omega \quad \geq \quad \hat{P} \qquad \forall\, \omega \in \Omega \tag{2.10}$$

$$R_2^\omega \quad \geq \quad \hat{R} \qquad \forall\, \omega \in \Omega \tag{2.11}$$

$$B_1 + E_\omega B_2^\omega = B_1 + \sum_{\omega\in\Omega} p^\omega B_2^\omega \quad \leq \quad \hat{B} \tag{2.12}$$

$$P_1 + E_\omega P_2^\omega = P_1 + \sum_{\omega\in\Omega} p^\omega P_2^\omega \quad \geq \quad \hat{P} \tag{2.13}$$

$$E_\omega R_2^\omega = \sum_{\omega\in\Omega} p^\omega R^\omega \quad \geq \quad \hat{R}, \tag{2.14}$$

where $\hat{B}$, $\hat{P}$, and $\hat{R}$ are user-specified targets for procurement spending, profit, and inventory reduction, respectively. Constraints (2.9)–(2.11) were in use at HP, while constraints (2.12)–(2.14) are expected value constraints that also may be applied to perform risk management. In the remainder of this chapter, this model is used to demonstrate the technique for reformulating stochastic programs with expected value constraints.

2.1.3 Motivation for Expected Value Constraints

HP would typically use a constraint approach [one of constraints (2.9),(2.10), (2.11)] for risk management. A problem would be solved several times for several different values of the right-hand side of the risk-management constraint. For example, if the goal of a problem were to minimize procurement spending, subject to some minimum level of inventory reduction, then HP would solve the model (2.1.2) with constraint (2.11) several times for several different values of $\hat{R}$. The result would be a curve of Pareto-optimal solutions showing the trade-off between procurement spending and inventory reduction. Figure 2.1 depicts an example of such a curve. The management at HP could then make a decision about how best to proceed with the production run.

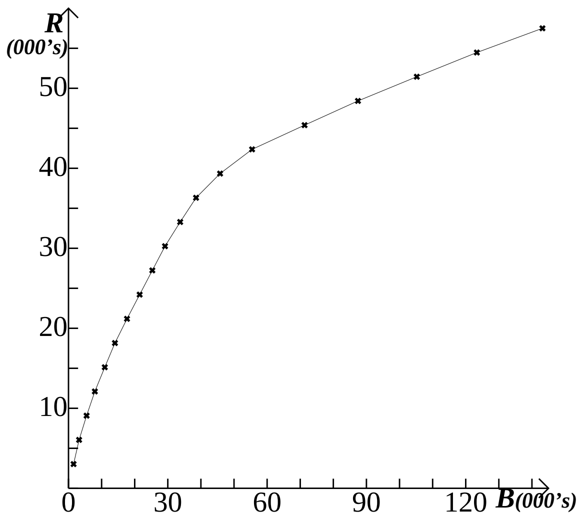

Figure 2.1: Tradeoff between procurement spending and inventory reduction for sample HP supply chain problem

This strategy was developed at a time when HP was using a deterministic model, so only one demand scenario would be used for each problem. When the strategy was extended to a stochastic model with multiple demand scenarios, some drawbacks to the approach emerged.

The strategy is particularly ill-suited for problems that have some "extreme" demand scenarios. Consider an HP supply chain problem with several demand scenarios, including one or more "very low" scenarios, with low levels of demand for all products. In the very low scenarios, very few end products can be sold, and therefore very little of the inventory can be reduced. The subproblem for this scenario is infeasible for any first-stage decision unless the lower bound on inventory reduction is set to be very low—low enough to be useless for the other scenarios. The examples below illustrate this issue.

Example 2.1 Consider a simple HP supply chain problem [model (2.1.2) plus constraint (2.11)] with 8 components and with the specifications as given in Tables 2.1, 2.2, and 2.3. Component 2 comes from an outside supplier that has discontinued that component, so the only source of component 2 is the current excess inventory of that component—it cannot be bought or assembled from other components. Component lead times are such that components that must be bought (components 1 and 3) must be procured in the first stage, before demand uncertainty is resolved, while components that are made (components 4 through 8) may be assembled quickly enough so that production decisions about those components can be deferred until demand uncertainty is resolved. The problem is solved with the goal of minimizing the procurement spending needed to achieve a given level of inventory reduction under all demand scenarios. That is, the supply-chain model (2.1.2) with the risk management constraint (2.11) is solved. Figure 2.2 depicts the tradeoff curve of spending vs. inventory reduction for all valid values of $\hat{R}$ in constraint (2.11). /

Example 2.2 Consider Example (2.1) above but with a more "extreme" set of demands, as given in Table 2.5. The low demand levels in some of the scenarios make the constraint (2.11) more difficult to satisfy. As a result, the tradeoff curve for this problem (Figure 2.3) has a much smaller range than the curve in the previous Example (Figure 2.2). One would not expect this curve to be of much use to the decision makers at HP. /

The worst scenarios—those with a relatively small feasible set—have the greatest impact on the problem's solution. In the example HP supply chain problems above,

		$\mathtt{I}$	=	$\{1,2,3,4,5,6,7,8\}$
$\mathtt{B}_1$	=	$\mathtt{B}$	=	$\{1,3\}$
$\mathtt{M}_2$	=	$\mathtt{M}$	=	$\{4,5,6,7,8\}$
		$\mathtt{S}$	=	$\{7,8\}$

Table 2.1: Set specification for Examples 2.1 and 2.2.

Component	Excess	Buy cost	Make cost	Scrap value	Sunk cost	Revenue
1	0	\$1	—	\$0	\$10	—
2	150	—	—	\$0	\$10	—
3	10	\$1	—	\$0	\$10	—
4	20	—	\$1	\$0	\$30	—
5	30	—	\$1	\$0	\$10	—
6	40	—	\$1	\$0	\$20	—
7	0	—	\$1	\$0	\$40	\$49
8	0	—	\$1	\$0	\$40	\$51

Table 2.2: Cost and other specifications for components in Examples 2.1 and 2.2.

Component 1	Units of component 1 per unit of component 2	Component 2
1	1	4
2	2	4
2	1	5
2	1	6
3	1	6
4	1	7
5	1	7
5	2	8
6	1	8

Table 2.3: Bill of materials for assembled components in Examples 2.1 and 2.2.

Scenario	Probability	Demand for comp. 7	Demand for comp. 8
1	0.1	30	30
2	0.1	35	35
3	0.1	40	40
4	0.1	45	45
5	0.1	50	50
6	0.1	50	30
7	0.1	45	35
8	0.1	40	40
9	0.1	35	45
10	0.1	30	50

Table 2.4: Demand scenarios for Example 2.1, with moderate variance in demands.

Scenario	Probability	Demand for comp. 7	Demand for comp. 8
1	0.05	5	10
2	0.1	25	30
3	0.2	40	40
4	0.1	55	50
5	0.05	75	70
6	0.05	75	10
7	0.1	55	30
8	0.2	40	40
9	0.1	25	50
10	0.05	5	70

Table 2.5: Demand scenarios for Example 2.2, with high variance in demands.

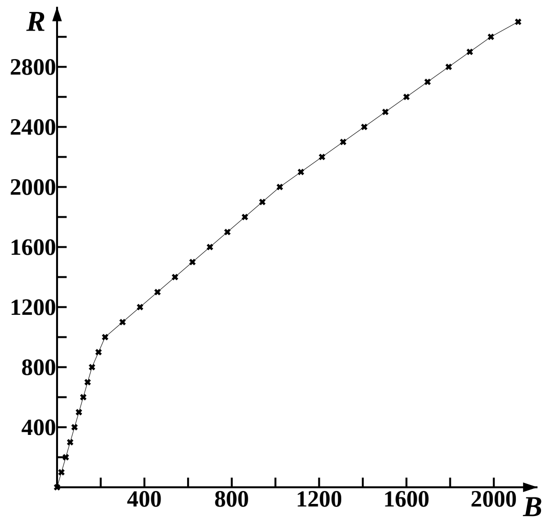

Figure 2.2: Tradeoff of procurement spending (B) vs. inventory reduction (R) for Example 2.1.

solutions are constructed in such a way that "the inventory reduction in *every* scenario is at least $\hat{R}$," even if those scenarios occur with a negligible probability.

A more flexible approach is to use expected value constraints. Then solutions are constructed in such a way that "the inventory reduction for some bad scenarios is less than $\hat{R}$, for good scenarios it is more than $\hat{R}$, but on average it is about $\hat{R}$." The solution is less susceptible to large influences from negligible scenarios.

Example 2.3 Using the same demand scenarios as Example (2.2), but including an expected value constraint on the inventory reduction [see (2.14)] rather than a bound on inventory reduction in all scenarios (2.11). The result is a much more meaningful tradeoff-curve, depicted in Figure 2.4. /

2.2 Reformulation of Expected Value Constrained Problems

Example 2.3 from the previous section is an expected value constrained problem. That problem is "small" in the sense that when the full stochastic program is specified as

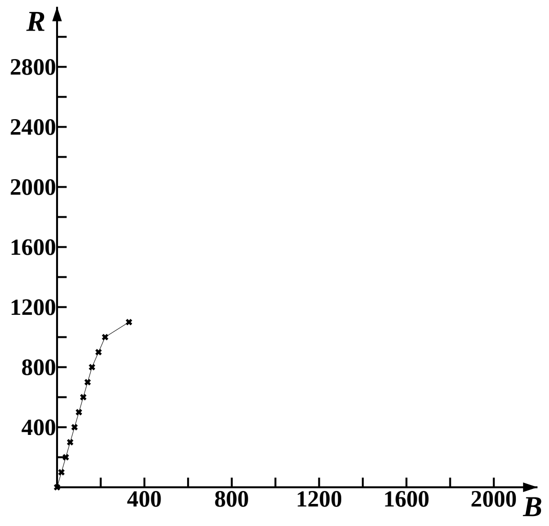

Figure 2.3: Tradeoff of B vs. R for Example 2.2. For $R >\sim 1200$, the problem is infeasible, so there are much fewer meaningful results for this example.

a single linear program, the size of the program does not overwhelm the available software packages for solving linear programs. The tableau for the fully-specified problem of Example 2.3 contains 122 equations and 193 decision variables, and it currently takes about 1.5 cpu seconds from a high-end Unix workstation to generate all of the data for Figure 2.4.

But many practical problems of interest are much larger than Example 2.3, and for all intents and purposes are not solvable without some sort of decomposition scheme. It was seen in Chapter 1 that a two-stage stochastic program with one or more expected value constraints does not have a dual-angular structure. Without the dual-angular structure, the subproblem cannot be decomposed into smaller independent subproblems, and Benders decomposition is still impractical for all but the smallest problems.

For most practical problems with expected value constraints, it is possible to reformulate the problem so that the desirable staircase structure reappears, and Benders decomposition can be practically applied. Let's consider a two-stage stochastic

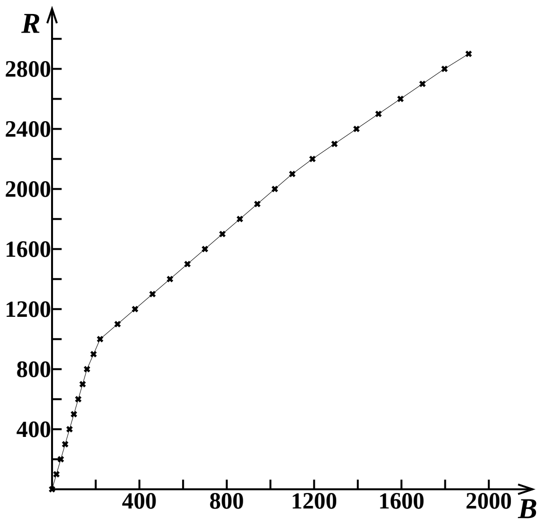

Figure 2.4: Tradeoff of B vs. R for model with high variance in demands, and a constraint on the expected value of R.

program with one or more expected value constraints:

$$\begin{array}{lrcrcll}
\text{Minimize} & cx & + & \sum_\omega f^\omega y^\omega & = & z & \\
\text{subject to} & Ax & & & = & b & \\
 & -B^\omega x & + & D^\omega y^\omega & = & d^\omega & \forall\, \omega \in \Omega \\
 & -Fx & + & \sum_\omega H^\omega y^\omega & = & h & \\
 & x, & & y^\omega & \geq & 0, &
\end{array} \qquad (2.15)$$

where x is the n_1-dimensional vector of first-stage decision variables, y^ω is an n_2-dimensional vector of second-stage decision variables, A is m_1-by-n_1, B^ω is m_2-by-n_1, D^ω is m_2-by-n_2, F is m_3-by-n_1, H^ω is m_3-by-n_2, b is m_1-dimensional, d^ω is m_2-dimensional, h is m_3-dimensional, and c and f^ω are the same dimensions as x and y^ω, respectively.

Figure 2.5 depicts a stochastic program with expected value constraints. The first stage consists of m_1 constraints and n_1 variables, each of the $|\Omega|$ subproblems of the second stage (ignoring the expected value constraints) consists of m_2 constraints and n_2 variables, and the expected value constraints constitute another m_3 constraints

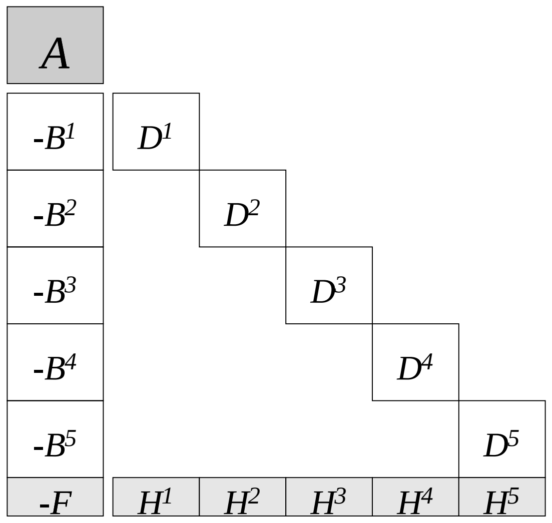

Figure 2.5: Block structure of stochastic program with expected value constraints

and $n_2 \times |\Omega|$ variables.

A large stochastic program with expected value constraints has a structure, which is unsuitable for either Benders decomposition or Dantzig-Wolfe decomposition, although it may be possible to use some nested decomposition scheme, as suggested by Figure 2.6. Instead, a reformulation of this problem is considered. The reformulation results in an equivalent problem that does have the desired staircase structure.

2.2.1 Identifying Problematic Second-Stage Variables

Expected value constraints are *problematic* in that they confound Benders decomposition schemes designed to make the problem more manageable. Second-stage variables that are represented in the expected value constraints are denoted as *problematic variables.*

In many practical cases, the expected value constraints are *sparse*[1]. This means that the number of problematic second-stage variables is typically rather small compared to the total number of second-stage variables. This is the case for the HP

[1]In other cases, they may be made sparse by the introduction of new variables. See section 4.2.

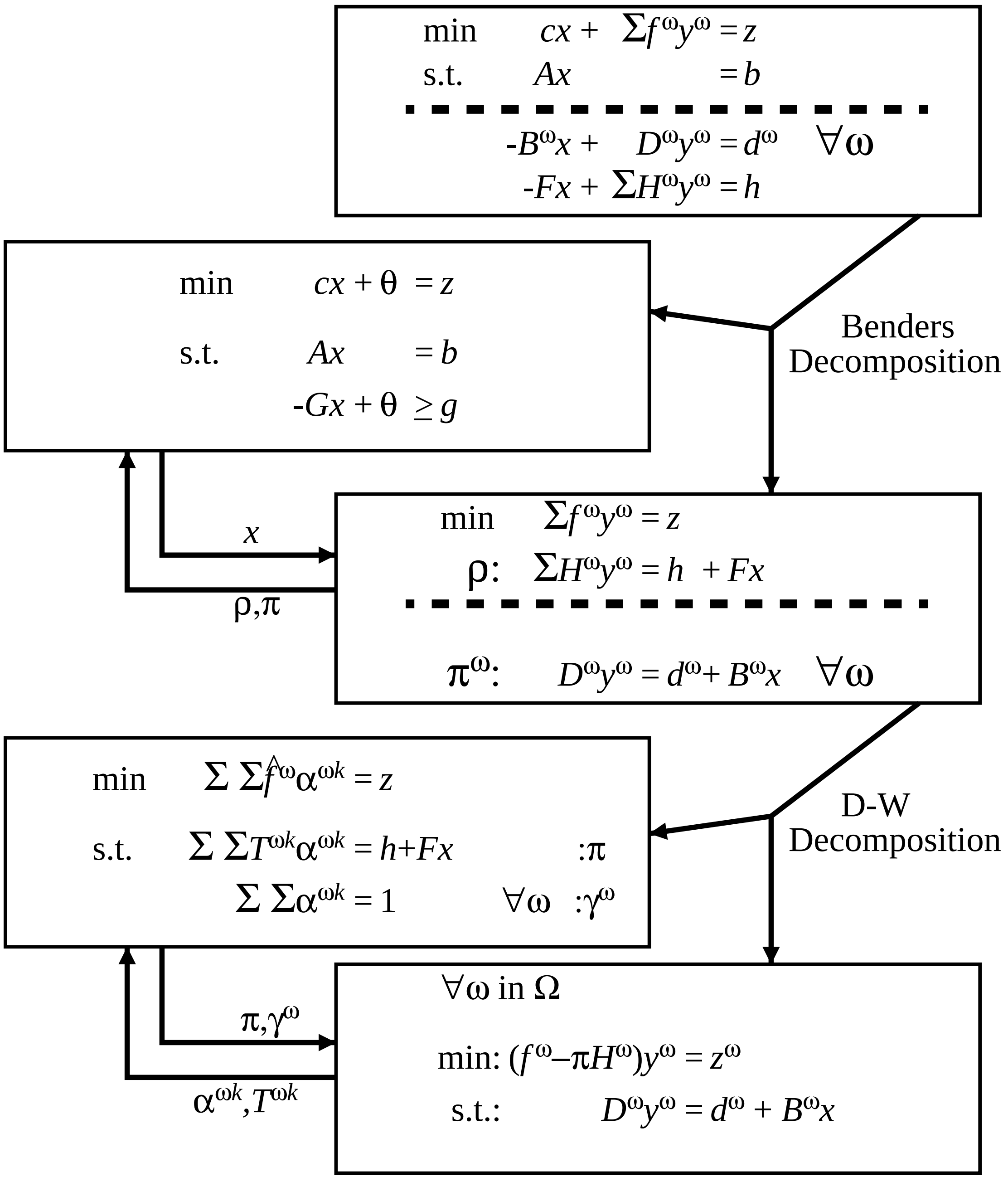
min $cx + \Sigma f^\omega y^\omega = z$
s.t. $Ax = b$
$-B^\omega x + D^\omega y^\omega = d^\omega \quad \forall\omega$
$-Fx + \Sigma H^\omega y^\omega = h$
Benders Decomposition
min $cx + \theta = z$
s.t. $Ax = b$
$-Gx + \theta \geq g$
x
ρ,π
min $\Sigma f^\omega y^\omega = z$
ρ: $\Sigma H^\omega y^\omega = h + Fx$
π^ω: $D^\omega y^\omega = d^\omega + B^\omega x \quad \forall\omega$
D-W Decomposition
min $\Sigma\Sigma \hat{f}^\omega \alpha^{\omega k} = z$
s.t. $\Sigma\Sigma T^{\omega k}\alpha^{\omega k} = h + Fx \quad :\pi$
$\Sigma\Sigma \alpha^{\omega k} = 1 \quad \forall\omega \quad :\gamma^\omega$
π,γ^ω
$\alpha^{\omega k}, T^{\omega k}$
$\forall\omega$ in Ω
min: $(f^\omega - \pi H^\omega)y^\omega = z^\omega$
s.t.: $D^\omega y^\omega = d^\omega + B^\omega x$

Figure 2.6: Suggestion of how Benders decomposition might be nested inside a Dantzig-Wolfe decomposition scheme to solve a problem with expected value constraints

supply-chain model with expected value constraints: the problematic variables are the procurement budget, profit, or inventory reduction in each scenario [from equations (2.12)–(2.14)]. There are only a few of these decision variables compared to the number of decision variables for second-stage production quantities, product sales, and component scrapping.

The first step in the reformulation is to identify these second-stage decision variables that are represented in the expected value constraints, and to distinguish them from the second-stage variables that are not represented in those constraints. Using y_1^ω to denote second-stage variables that are represented in expected value constraints, and y_2^ω for variables that are not represented in those constraints, a general two-stage stochastic program with expected value constraints can be expressed as:

$$\begin{array}{lrcrcrcl l}
\text{Minimize} & cx & + & \sum_\omega f_1^\omega y_1^\omega & + & \sum_\omega f_2^\omega y_2^\omega & = & z & \\
\text{subject to} & Ax & & & & & = & b & \\
\text{ev constr.:} & -Fx & + & \sum_\omega H^\omega y_1^\omega & & & = & h & \\
 & -B^\omega x & - & D_1^\omega y_1^\omega & + & D_2^\omega y_2^\omega & = & d^\omega & \forall\, \omega \in \Omega \\
 & x, & & y_1^\omega, & & y_2^\omega & \geq & 0. &
\end{array} \qquad (2.16)$$

2.2.2 Moving Problematic Variables to the First Stage

The next step is to look at the problem from a different point of view. It is natural to look at the problematic variables in the above formulation as second-stage variables, since they depend on some uncertainty that will not be realized until the second stage of the problem, and the number of these variables is proportional to the number of scenarios in the model.

But suppose the problem is partitioned so that the expected value constraints are a part of the master problem. In this case, problem (2.16) does have a dual angular structure and may be decomposed such that the master problem is

$$\begin{array}{lrclcll} \text{Minimize} & cx & + & \sum_\omega f_1^\omega y_1^\omega & + \ \theta & = & \zeta \\ \text{subject to} & Ax & & & & = & b \\ & -Fx & + & \sum_\omega H^\omega y_1^\omega & & = & h \\ \text{cuts:} & -G_0 x & - & \sum_\omega G_1^\omega y_1^\omega & + \ \theta & \geq & g \\ & -G_0' x & - & \sum_\omega {G_1'}^\omega y_1^\omega & & \geq & g' \\ & x, & & y_1^\omega & & \geq & 0 \qquad \forall\, \omega \in \Omega, \end{array} \tag{2.17}$$

with independent subproblems for all $\omega \in \Omega$:

$$\begin{array}{lrcl} \text{Minimize} & f_2^\omega y_2^\omega & = & \theta(x, y_1^\omega, \omega) \\ \text{subject to} & D_2^\omega y_2^\omega & = & d^\omega + B^\omega x + D_1^\omega y_1^\omega \\ & y_2^\omega & \geq & 0. \end{array} \tag{2.18}$$

Then the problem is in a form suitable for use with independent subproblems in Benders decomposition. Figure 2.7 depicts how to change one's perspective on a two-stage program with expected value constraints to obtain a two-stage program that is suitable for Benders decomposition.

2.2.3 Comparison of Reformulated Problem

Expected value constraints contain second-stage variables and may have coefficients that are not known with certainty at the time the problem is solved. For this reason, the intuitive way of formulating and decomposing the problem places the expected value constraints in the subproblem (as shown in Figure 2.5). In this case, the problem does not have a dual-angular structure, and the subproblem cannot be further decomposed into independent subproblems. For large problems, Benders decomposition does still not make the problem practical to solve.

In the reformulated problem, the expected value constraints are moved into the master problem (2.17). The master problem of the reformulated problem thus has one more constraint (not including the cuts) and several more variables—the number of extra variables is proportional to the number of scenarios being considered—than

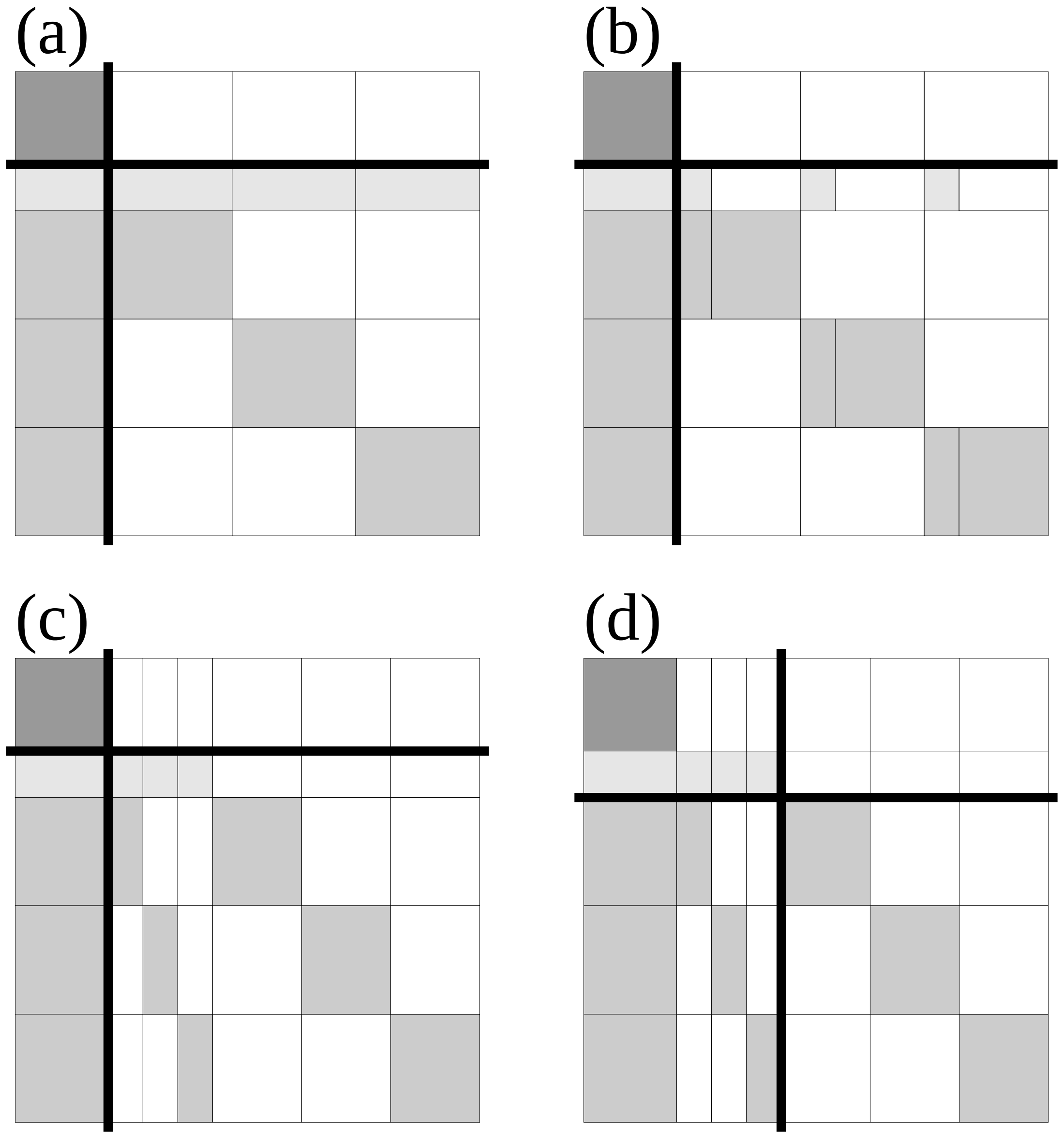

Figure 2.7: (a) Two-stage problem with expected value constraint. Dark box shows "natural" choice for first stage. (b) Distinguishing problematic variables. (c) New perspective of problem structure. (d) New choice of how to divide the problem into stages.

the master problem of the original problem. But the reformulated problem does have dual angular structure. So Benders decomposition can be applied to expected value constrained problems, and based on past experience, it should make the problem tractable.

2.3 Numerical Results

A variety of supply-chain problems from HP were solved under many different demand distributions, with both the conventional risk management constraints (2.9)–(2.11) and expected value constraints (2.12)–(2.14). Benders decomposition was applied to the problems—using the reformulated model (2.17) and (2.18) for the expected value constrained case—and the problems were solved using Gerd Infanger's DECIS (Benders DEComposition and Importance Sampling) [11] software package, an implementation of Benders decomposition.

Results are presented in Table 2.6, including the characteristic size of the model, the number of demand scenarios used, the number of times the master problem and independent subproblems were solved to achieve a solution, and the CPU time needed to solve the problem on a high-end Silicon Graphics workstation.

2.4 Discussion of Results

From the previous section, it can be seen that reformulation often does allow an expected value constrained problem to be solved with Benders decomposition. But in some cases, particularly in cases where many demand scenarios ($|\Omega|$) are used, a large amount of iterations and computational effort is required to solve the problems with expected value constraints compared to the effort needed to solve problems with the conventional risk management constraints.

The number of columns in the master problem is often one of the most significant factors in determining how much effort it takes to solve a stochastic problem. As shown in Section 2.2, the expected value constrained problem does have one extra variable in the master problem for each scenario that is considered. But in some cases,

		Conventional model		EV constrained model	
Problem†	number of scenarios	solution time	MP/SP	solution time	MP/SP
tom_{1K}	5	0.08	2/10	0.08	2/10
tom_{1K}	25	0.36	13/109	0.53	8/200
tom_{1K}	100	0.88	13/409	7.40	32/3200
tom_{10K}	5	0.31	22/58	0.42	28/75
tom_{10K}	25	0.68	28/220	3.89	93/1802
tom_{10K}	100	2.26	31/1120	598.47	2094/2.1E5
tom_{29K}	5	0.37	25/73	0.70	51/185
tom_{29K}	25	0.97	33/393	18.49	343/7639
tom_{29K}	100	2.88	33/1518	1431.93	5615/5.5E5
bap_{10K}	5	122.72	425/2121	146.02	448/2230
bap_{10K}	25	168.16	423/1.1E4	923.42	999/2.5E4
bap_{10K}	100	310.03	420/4.2E4	6698*	2335/2.2E5
bap_{1M}	5	283.41	647/3231	497.76	940/4690
bap_{1M}	25	347.09	647/1.6E4	1980.37	1902/4.6E4
bap_{1M}	100	565.15	647/6.5E4	††	††/††
bap_{5M}	5	346.57	673/3361	524.89	900/4490
bap_{5M}	25	473.53	852/2.1E4	3243*	2532/6.0E4
bap_{5M}	100	893.57	925/9.2E4	††	††/††
dlaw_{5K}	5	5377*	4081/2.0E4	11542*	6102/3.0E4
dlaw_{5K}	25	7684*	4292/1.1E5	††	††/††
dlaw_{200K}	5	22297*	1.4E4/6.8E4	††	††/††
dlaw_{200K}	25	30075*	1.5E4/3.7E5	††	††/††

Table 2.6: Results for HP Supply Chain Models with conventional risk management constraints and with expected value constraints. Except where noted, solution time refers to CPU seconds of a single processor needed to solve the problem.

† Characteristic size of these problems:

Problem	**MP size**	**SP size**	**Deterministic size** with $\|\Omega\| = 100$
tom	39×39	163×149	16339×14939
bap	265×265	548×817	54965×81965
dlaw	378×383	2844×4980	284748×498382

†† A solution was not obtained for this problem in fewer than 25 hours or 30000 iterations.

* Solution time is reported in real seconds, not CPU seconds — this is a minor overestimation of the CPU time needed to solve the problem.

the effort needed to solve a reformulated expected value constrained problem was 10 times or more that the effort needed to solve the conventionally formulated problem. The few extra variables in the master problem is not a sufficient explanation for the extra effort required.

Rather, there is something fundamentally *pathological* about the structure of the reformulated expected value constrained problem with Benders decomposition. For the problematic variables that are moved from the second stage of the problem to the first stage, each variable has an effect on the subproblem objective value for only one scenario, and support for those variables as provided by the feasibility and optimality cuts is very slow to develop. Consequently, it takes many iterations of Benders decomposition before the values of those problematic variables start to converge toward their optimal values.

The multi-cut technique of Birge and Louvaeux [2] adds a cut to the master problem every time an independent subproblem is solved. This technique may help build support for the problematic variables more quickly, since it would add several cuts to the problem in each iteration. But this technique suffers from its own scalability issues – it generates cuts in proportion to the number of random outcomes under consideration, and in general it cannot be used with sampling random outcomes from a continuous or otherwise large sample space. The drawbacks are serious enough to reject this technique out of hand as a worthwhile approach to dealing with expected value constraints. Instead, other approaches that work in the context of conventional Benders decomposition should be explored. This is the focus of the next chapter.

Chapter 3

A Modified Benders Decomposition Algorithm

In the previous chapter, it was demonstrated that in many practical cases a two-stage problem with one or more expected value constraints could be reformulated into a program with dual-angular structure, so the L-shaped method of Van Slyke and Wets [14] can be applied to solve a large stochastic program. The number of iterations of Benders decomposition needed to solve such a reformulated problem was very large compared to what one would expect from a typical stochastic program of the same size. In this chapter, other approaches for solving expected-value constrained problems are discussed, with emphasis on techniques that reduce the overall computational effort needed to achieve a solution. A modified Benders decomposition algorithm is presented, which efficiently solves expected-value constrained problems that satisfy certain conditions, along with some numerical results that demonstrate the tractability of many expected-value constrained problems.

3.1 Nested Solution Approaches

The structure of expected value constrained problems suggests that they could be solved with a nested decomposition approach. There are at least two ways to set up and solve the problem with a nested decomposition approach.

3.1.1 Nested Dantzig-Wolfe Decomposition

Figure 2.6 in the previous chapter suggests how one might use Dantzig-Wolfe decomposition inside Benders decomposition. First, one could partition the problem (2.15) into a master problem and a single, large subproblem. The solution to the master problem would determine a first-stage solution x that would be passed to the subproblem. The subproblem has a block angular structure, and it may be solved with Dantzig-Wolfe decomposition to determine the second-stage solutions y^ω that satisfy the independent scenario subproblems and the expected value constraint.

3.1.2 Nested Benders Decomposition

When the expected value constraints are sparse, another type of nested decomposition is suggested. The expected value constrained problem (2.15) can again be partitioned into a master problem and a single, large subproblem. The solution to the master problem would determine a first-stage solution x to be passed to the subproblem. By distinguishing the second-stage variables y_1^ω that are represented in the expected value constraints from the second-stage variables y_2^ω that are not, the large subproblem can be decomposed further. The large subproblem contains a master problem consisting of the expected value constraint. Solution of the master problem would lead to a second-stage solution for y_1^ω. The remainder of the large subproblem could be decomposed into independent scenario subproblems, and solutions for y_2^ω can be obtained by applying the L-shaped method of Van Slyke and Wets. When an expected value constrained problem is partitioned this way, it may be thought of as a *three-stage* problem: values for x are determined in the first stage, values for y_1^ω are determined in the second stage, and values for y_2^ω are determined in the third stage. Figure 2.7 from the previous chapter is a graphical depiction of this type of nested decomposition.

Either of the above two nested decomposition schemes is necessarily more complex than a single decomposition scheme. If, for example, the master problem above took an average of 100 iterations to converge on an optimal first-stage solution, and the large subproblem, whether using Benders or Dantzig-Wolfe decomposition, also took about 100 iterations to determine an optimal second-stage solution, the corresponding nested decomposition scheme would require each independent subproblem to be solved 10,000 times to achieve a solution. With the additional complexity and effort required to solve such a problem, a nested decomposition solution can hardly be expected to be much more useful than applying a single layer of decomposition over the reformulated problem from the previous chapter.

3.2 Problem Partitioning Schemes

Nested decomposition schemes are not considered further in this chapter, so attention must shift to other ideas for handling problem (2.16). But between the problem reformulation from Chapter 2 and the nested Benders decomposition scheme from the previous section, the issue of how best to partition the problem arises. A couple of mutually exclusive partitioning suggestions will now be explored more deeply.

3.2.1 Scheme 1

Figure 2.7 suggests one way to partition the problem into a master problem and a subproblem. Under this partitioning scheme, the master problem looks like

$$\begin{array}{lrcrcrcl}
\text{Minimize} & c_1x_1 & + & c_2x_2 & + & \theta_3 & = & z \\
\text{subject to} & A_{11}x_1 & & & & & = & b_1 \\
 & -A_{12}x_1 & + & A_{22}x_2 & & & = & b_2 \\
 & -G_1x_1 & - & G_2x_2 & + & \theta_3 & \geq & g \\
 & x_1, & & x_2 & & & \geq 0, &
\end{array} \tag{3.1}$$

where x_1 denotes the original first-stage decision variables, x_2 are the problematic second-stage variables (second-stage variables represented in the expected value constraints), and the number of elements in x_2 is related to the number of random outcomes under consideration $|\Omega|$. Letting x_3^ω denote the non-problematic (remaining) second-stage variables, the independent subproblems look like

$$\begin{array}{rrcll} \text{Minimize} & c_3^\omega x_3^\omega & = & z_3^\omega & \\ \text{subject to} & A_{33}^\omega & = & b_3^\omega + A_{13}^\omega x_1 + A_{23}^\omega x_2 & \quad : \pi^\omega \\ & x_3^\omega & \geq & 0. & \end{array} \tag{3.2}$$

Due to the nature of expected value constraints, the A_{23}^ω matrices are constructed in such a way that every nonzero column of A_{23}^ω is associated with the same scenario, so that if a particular column of A_{23}^ω has nonzero elements, then that column of $A_{23}^{\omega'}$ must be zero for all $\omega' \neq \omega$.

The cuts in the master problem (3.1) are constructed from the dual solution to the subproblem as

$$\begin{array}{rcl} G_1 & = & \sum_\omega p_\omega \pi^\omega A_{13}^\omega \\ G_2 & = & \sum_\omega p_\omega \pi^\omega A_{23}^\omega \\ g & = & \sum_\omega p_\omega \pi^\omega b_3^\omega. \end{array} \tag{3.3}$$

3.2.2 Scheme 2

A second partitioning scheme that is a little more intuitive places the expected value constraint back into the subproblem. In this case the master problem is

$$\begin{array}{rrcccl} \text{Minimize} & c_1 x_1 & + & \theta_2 & = & z \\ \text{subject to} & A_{11} x_1 & & & = & b_1 \\ & -G' x_1 & + & \theta_2 & \geq & g' \\ & x_1 & \geq & 0, & & \end{array} \tag{3.4}$$

the subproblem is

$$\begin{array}{rrcrcll} \text{Minimize} & c_2 x_2 & + & \sum_\omega p_\omega c_3^\omega x_3^\omega & = & z_2 & \\ \text{subject to} & A_{22} x_2 & & & = & b_2 + A_{12} x_1 & : \rho \\ & -A_{23}^\omega x_2 & + & A_{33}^\omega x_3^\omega & = & b_3^\omega + A_{13}^\omega x_1 & : \pi^\omega \qquad \forall\, \omega \in \Omega \\ & x_2, & & x_3^\omega & \geq & 0, & \end{array} \tag{3.5}$$

and the cuts are formulated from the dual solution to the subproblem as

$$\begin{array}{rcccl} G' & = & \rho A_{12} & + & \sum_\omega p_\omega \pi^\omega A_{13}^\omega \\ g' & = & \rho b_2 & + & \sum_\omega p_\omega \pi^\omega b_3^\omega. \end{array} \tag{3.6}$$

3.2.3 Comparison of the partitioning schemes

There are some substantial differences between these two partitioning schemes, and it is worthwhile to examine these differences more closely. Both of these schemes have some desirable features and some undesirable features.

- **Master problem** In scheme 1, the expected value constraint is included in the master problem, and so some of the decision variables in the master problem are related to the stochastic outcome. Scheme 2 has a more intuitive master problem because only the original first-stage decision variables x_1 are determined. The master problem for scheme 1 (3.1) contains more decision variables than the master problem for scheme 2 (3.4), and one would expect the master problem for scheme 1 to be a little more difficult to solve than the master problem from scheme 2.

- **Cuts** The master problem from scheme 1 seeks to estimate the cost contributions of all the non-problematic second-stage variables. The cuts contain coefficients for both x_1 and x_2, but the expressions used to determine these coefficients (3.3) are in a familiar form. The previous chapter demonstrates that these cuts provide very poor support for x_2, so one would expect a Benders decomposition procedure that uses partitioning scheme 1 to take a very large number of iterations to solve. In scheme 2, only x_1 is determined in the master problem, so only x_1 is represented in the cuts. But the expression for the cut (3.6) contains unfamiliar ρ terms, which are associated with the dual prices of the expected value constraint.

- **Subproblem** In scheme 1, only the non-problematic variables x_3^ω are determined in the subproblem. The subproblem can be further decomposed into independent "scenario" subproblems, with one subproblem associated with each possible outcome under consideration. This is a very desirable feature for solving large problems. In contrast, scheme 2 has the expected value constraint in the subproblem, which wrecks the dual angular structure of the problem and prevents the subproblem from being further decomposed into independent subproblems. The subproblem with the expected value constraint is denoted as the "large" subproblem to indicate that it requires all scenarios to be solved for simultaneously and cannot be broken up into several smaller problems.

Each scheme has its advantages and disadvantages. In a problem without expected value constraints, the master problem resembles (3.4), the cuts are likely to provide good support, and the problem is likely to be solved in a reasonable number of iterations. The subproblem can be decomposed into independent subproblems like (3.2), allowing the subproblems to be solved very efficiently. For problems with expected value constraints, these same features are desirable. The cuts for the master problem should provide good support for the recourse function. The subproblem should be able to be decomposed into independent "scenario" subproblems, or should otherwise be able to be solved efficiently.

3.3 Solving the "large" subproblem

The linear program specified by (3.5) arises as a single system of equations in the context of decomposing the full problem (cf. 2.16). Because of the expected value constraint, this system cannot be decomposed further into independent subproblems, and so (3.5) is referred to as the "large" subproblem.

3.3.1 Characterization of the "large" subproblem

If an efficient method for solving the "large" subproblem (3.5) were available, then the second partitioning scheme from the previous section could be applied to solve the expected value constrained problem in a reasonable number of iterations. Depending on the characteristics of the large subproblem, there may be such a method. In this section a modified Benders decomposition algorithm is presented that allows the large subproblem to be solved efficiently when the following conditions hold:

1. **There is a one-to-one mapping between the elements of x_2 and the elements of Ω.** That is, A_{23}^{ω} has at most one nonzero column for each $\omega \in \Omega$.

   Let a_ω denote the unique nonzero column vector in A_{23}^{ω} that is associated with scenario ω. Then the subproblem constraints $A_{33}^{\omega}x_3^{\omega} = b_3^{\omega} + A_{23}^{\omega}x_2 + A_{13}^{\omega}x_1$ can be expressed as

   $$A_{33}^{\omega}x_3^{\omega} = b_3^{\omega} + a_\omega x_{2\omega} + A_{13}^{\omega}x_1. \tag{3.7}$$

2. **There is exactly one expected value constraint.** That is, the expression $A_{22}x_2 = b_2 + A_{12}x_1$ consists of exactly one equation. b_2 is a scalar, and A_{12} and A_{22} are row-vectors.

3. **The expected value constraint is an equality.**

From items 1 and 2 above, there is a coefficient α_ω[1] in the row-vector A_{22} that is uniquely associated with scenario $\omega \in \Omega$. The single expected value constraint can then be written as

$$\sum_{\omega \in \Omega} \alpha_\omega x_{2\omega} = b_2 + a_{12}x_1. \tag{3.8}$$

Without loss of generality, it is assumed that $\alpha_\omega \geq 0$ for all ω. If α_ω is negative, then the problem may be reformulated with $-x_{2\omega}$ instead of $x_{2\omega}$, with the constraint $x_{2\omega} \geq 0$ being relaxed without consequence.

To summarize, the specification of the expected value constrained problem to be solved is

[1]If (3.8) is a true expected value constraint, then $\alpha_\omega = p_\omega$, the probability associated with scenario ω for all $\omega \in \Omega$. But this need not be the case—for example, the modeler may wish to bias the model for a few "extreme" outcomes, in order to obtain a solution that performs well under some worst-case scenarios. In this case, the coefficients associated with those scenarios may be weighted more heavily.

$$\begin{array}{lrcrcrcll}
\text{Minimize} & c_1x_1 & + & c_2x_2 & + & \sum_\omega p_\omega c_3^\omega x_3^\omega & = & z & \\
\text{subject to} & A_{11}x_1 & & & & & = & b_1 & \\
 & -a_{12}x_1 & + & \sum_\omega \alpha_\omega x_{2\omega} & & & = & b_2 & \\
 & -A_{13}^\omega x_1 & - & A_{23}^\omega x_2 & + & A_{33}^\omega x_3^\omega & = & b_3^\omega & \forall\,\omega \in \Omega \\
 & & & \alpha_\omega x_{2\omega} & & & \geq & 0 & \forall\,\omega \in \Omega \\
 & x_1, & & & & x_3^\omega & \geq & 0, &
\end{array} \tag{3.9}$$

where A_{23}^ω has at most one nonzero column for each scenario $\omega \in \Omega$.

3.3.2 Algorithm to solve the large subproblem

If conditions 1–3 above hold for the expected value constrained problem, then the following algorithm leads to an optimal solution (x_2, x_3^ω) to the large subproblem (3.5).

Algorithm 3.1 Solution procedure for the large subproblem (3.5).

Step 0. Determine an initial allocation x_2 such that the expected value constraint $\sum_{\omega\in\Omega} \alpha_\omega x_{2\omega} = b_2 + a_{12}x_1, x_2 \geq 0$ is satisfied. Define $\Omega' := \emptyset$.

Step 1. Define the *resource price function for scenario* ω as

$$\psi_\omega(x_{2\omega}) \;,\; \frac{c_{2\omega}}{\alpha^\omega} + \pi^\omega(x_{2\omega})a_\omega \frac{p_\omega}{\alpha^\omega}. \tag{3.10}$$

The first term of this function describes the direct effect of cost coefficient $c_{2\omega}$ on the objective function value for (3.5) for a change in $x_{2\omega}$. The derivation of the second term is discussed later, but for now it can be thought of as the effect of a change in

$x_{2\omega}$ on the objective function value of the scenario subproblem

$$\begin{array}{llcll} \text{Minimize} & c_3^\omega x_3^\omega & = & z_3^\omega(x_{2\omega}) & \\ \text{subject to} & A_{33}^\omega x_3^\omega & = & b_3^\omega + A_{13}^\omega x_1 + a_\omega x_{2\omega} & : \pi^\omega. \end{array} \tag{3.11}$$

Fix x_2 with the values determined in Step 0 and solve the independent scenario subproblems (3.11) for all $\omega \in \Omega$ to determine $\pi^\omega(x_{2\omega})$ and $\psi_\omega(x_{2\omega})$. Record the values of ψ_ω.

If the scenario subproblem associated with ω is infeasible, then it should be determined why the subproblem is infeasible. If an increase in $x_{2\omega}$ would allow the problem to become feasible, then assign $\psi_\omega := -\infty$. If a decrease in $x_{2\omega}$ would allow the problem to become feasible, then assign $\psi_\omega := +\infty$. If no value of $x_{2\omega}$ makes the scenario subproblem feasible, then note the large subproblem (3.5) as infeasible and stop.

Step 2. Determine $j \in \Omega \setminus \Omega' : \psi_j \geq \psi_i \qquad \forall\, i \in \Omega \setminus \Omega'$

$k \in \Omega : \qquad \psi_k \leq \psi_i \qquad \forall\, i \in \Omega,$

where the meaning of Ω' is discussed below. If $\Omega \setminus \Omega' = \emptyset$ or if $\psi_j = \psi_k$, then record the current solution (x_2, x_3^ω) as the optimal solution and stop.

Step 3. Reallocate resources from scenario j to scenario k as follows:
Determine $\Delta x_{2j} \geq 0$ and $\Delta x_{2k} := \Delta x_{2j}\dfrac{\alpha_j}{\alpha_k}$ such that for some small value of $\delta > 0$, either

$$\begin{array}{lclcl} \psi_k(x_{2k}) & < & \psi_k(x_{2k} + \Delta x_{2k}) & \leq & \psi_j(x_{2j} - \Delta x_{2j}) \\ & & \psi_k(x_{2k} + \Delta x_{2k} + \dfrac{\delta}{\alpha_k}) & > & \psi_j(x_{2j} - \Delta x_{2j} - \dfrac{\delta}{\alpha_j}) \end{array} \tag{3.12}$$

or

$$\begin{aligned} \psi_k(x_{2k}) \;=\; \psi_k(x_{2k} + \mathtt{M}x_{2k}) &< \psi_j(x_{2j} - \mathtt{M}x_{2j}) \\ \psi_k(x_{2k} + \mathtt{M}x_{2k} + \frac{\delta}{\alpha_k}) &\geq \psi_j(x_{2j} - \mathtt{M}x_{2j} - \frac{\delta}{\alpha_j}) \end{aligned} \tag{3.13}$$

is satisfied. If $\mathtt{M}x_{2j}$ and $\mathtt{M}x_{2k}$ can be increased without bound and without either condition (3.12) or (3.13) being satisfied, then report the large subproblem as being unbounded with extreme vector $-\mathtt{M}x_{2j}e_j + \mathtt{M}x_{2k}\frac{\alpha_k}{\alpha_j}e_k$.

Step 4. Reallocate x_{2j} and x_{2k} as follows:

If condition (3.12) was satisfied at the end of the previous step, then remove k from Ω', if $k \in \Omega'$. If condition (3.13) was satisfied at the end of the previous step, then add j to Ω'.

Let $x_{2j} := x_{2j} - \mathtt{M}x_{2j}$; $x_{2k} := x_{2k} + \mathtt{M}x_{2k}$. Return to Step 2. /

A few of these steps require some additional explanation. x_2 can be thought of as a vector of resource allocations that are distributed among the scenarios. The resource price functions ψ_ω can be thought of as the relative value of allocating additional resources to scenario ω. In Step 2, the scenarios with the highest and lowest values of these resource price functions are selected, and in Step 3 resources are removed from the scenario with the low function value and added to the scenario with the high function value, in such a way that the expected value constraint (3.8) is always satisfied. The amount of resources to be transferred is governed by the point at which the resource price functions of the two scenarios become equal. This point can be determined with a simple bisection line search, but every iteration of the line search requires two extra scenario subproblems (3.11) to be solved.

In the next subsection, it is shown that the resource price functions ψ_ω are increasing step functions. In Step 3, as $\mathtt{M}x_{2j}$ and $\mathtt{M}x_{2k}$ increase, ψ_k necessarily increases, and ψ_j necessarily decreases. If $\mathtt{M}x_{2j}$ and $\mathtt{M}x_{2k}$ can be increased without bound and without the condition $\psi_k \geq \psi_j$ ever being satisfied, then resources can be transferred without bound from scenario k to scenario j, with an improvement in the objective function related to $\psi_k - \psi_j$ for every unit of resources transferred. The large subproblem can then be identified as having an unbounded solution.

3.3.3 Properties of the algorithm

It was previously stated that Algorithm 3.1 leads to an optimal solution of the large subproblem (3.5). This assertion will now be proved.

The scenario subproblem (3.11) is a *parametric linear program* in $x_{2\omega}$. Dantzig and Thapa [7] cite some useful properties of the objective function value for parametric linear programs of this type. The optimal objective function value $z_3^\omega(x_{2\omega})$ of this problem is necessarily

1. continuous over all valid values of $x_{2\omega}$;
2. piecewise-linear; and
3. convex in $x_{2\omega}$, that is $z_3^\omega(\lambda x_{2\omega} + [1-\lambda]x'_{2\omega}) \geq \lambda z_3^\omega(x_{2\omega}) + [1-\lambda]z_3^\omega(x'_{2\omega})$ for all $0 \leq \lambda \leq 1$, $x_{2\omega}$, and $x'_{2\omega}$.

An additional useful property is that the piecewise-linear function has a finite number of “pieces”. This can be seen because each “piece” of the function corresponds to a different optimal basis of A_{33}^ω. There is only a finite number of bases in A_{33}^ω.

The first derivative of z_3^ω with respect to $x_{2\omega}$ is necessarily an increasing step function, with a discontinuity wherever $z_3^\omega(x_{2\omega})$ moves from one “piece” of the function to another. This step function also has a finite number of pieces. The optimal solution (x_3^ω, π^ω) of (3.11) can also be thought of as a parametric function in $x_{2\omega}$, and by the duality of linear programs, the following relation must hold:

$$z_3^\omega(x_2) = z_3^\omega(x_{2\omega}) = z_3^\omega((x_3^\omega, \pi^\omega)(x_{2\omega})) = \pi^\omega(x_{2\omega})(b_3^\omega + A_{13}^\omega x_1 + a_\omega x_{2\omega}). \tag{3.14}$$

That is, for a given $x_{2\omega}$ and optimal solution (x_3^ω, π^ω) to (3.11), the slope of $z_3^\omega(x_{2\omega})$ with respect to $x_{2\omega}$ is $\pi^\omega(x_{2\omega})a_\omega$. The objective function value of the large subproblem (3.5) may be rewritten as

$$z_2 = c_2 x_2 + \sum_\omega p_\omega z_3^\omega(x_2), \tag{3.15}$$

and in light of the previous discussion, it is apparent that for any $i \in \Omega$,

$$\begin{aligned} \frac{\partial z_2}{\partial x_{2i}} &= c_{2i} + \textstyle\sum_\omega p_\omega \frac{\partial z_3^\omega}{\partial x_{2i}} \\ &= c_{2i} + p_i \frac{\partial z_3^i}{\partial x_{2i}} \\ &= c_{2i} + p_i \pi^i a_i. \end{aligned} \tag{3.16}$$

When an infinitesimal amount of resources $(-\mathtt{M}x_{2j}, \mathtt{M}x_{2k})$ are transferred between two scenarios, the corresponding change in the objective function value (3.15) is thus

$$\mathtt{M}x_{2j} \frac{\partial z_2}{\partial x_{2k}} - \mathtt{M}x_{2k} \frac{\partial z_2}{\partial x_{2j}}. \tag{3.17}$$

If the expected value constraint (3.8) is to be satisfied, then $\mathtt{M}x_{2j} = \mathtt{M}x_{2k}\frac{\alpha_k}{\alpha_j}$, and

$$\begin{aligned} \mathtt{M}x_{2k}\frac{\partial z_2}{\partial x_{2k}} - \mathtt{M}x_{2j}\frac{\partial z_2}{\partial x_{2j}} &= \mathtt{M}x_{2k}(\frac{\partial z_2}{\partial x_{2k}} - \frac{\alpha_k}{\alpha_j}\frac{\partial z_2}{\partial x_{2j}} \\ &= \mathtt{M}x_{2k}(c_{2k} + p_k\pi^k a_k - \frac{\alpha_k}{\alpha_j}(c_{2j} + p_j\pi^j a_j) \\ &= \mathtt{M}x_{2k}\alpha_k[\frac{1}{\alpha_k}(c_{2k} + p_k\pi^k a_k) - \frac{1}{\alpha_j}(c_{2j} + p_j\pi^j a_j)] \\ &= \mathtt{M}x_{2k}\alpha_k(\psi_k - \psi_j) \end{aligned} \tag{3.18}$$

where ψ_j and ψ_k have the form described by (3.10). Hence (3.10) captures the notion of the relative value of changing the allocation of resources to some scenario. These properties are useful in proving the termination and optimality of Algorithm 3.1.

It is now appropriate to state some properties of Algorithm 3.1.

Lemma 3.1 *Let $\psi_{\min} := \min_{\omega\in\Omega}\psi_\omega(x_{2\omega})$ denote the smallest price resource function value at any point in the algorithm. As the algorithm proceeds, $\psi_{\min}$ never decreases.*

Proof. By construction the condition $\psi_k(x_{2k} + \mathtt{M}x_{2k}) \le \psi_j(x_{2j} - \mathtt{M}x_{2j})$ always holds at the end of Step 2, whether condition (3.12), condition (3.13), or neither is satisfied. $\psi_k(x_{2k})$ is an increasing function of x_{2k}, so $\psi_k(x_{2k} + C) \ge \psi_k(x_{2k})$ for all $C \ge 0$. By construction, $\mathtt{M}x_{2k} \ge 0$, so $\psi_k(x_{2k}) \le \psi_k(x_{2k} + \mathtt{M}x_{2k})$. Hence at the end of Step 4, after resources have been transferred from scenario j to scenario k, the condition $\psi_{\min} \leq \psi_k \leq \psi_j$ holds. Since no other resource price function values are affected by this step, no value of $\psi_i, i \in \Omega$ has changed such that $\psi_i < \psi_{\min}$ could be true. ∎

Lemma 3.2 *Let* $\psi_{\min} := \min_{\omega \in \Omega} \psi_\omega(x_{2\omega})$ *denote the smallest resource price function value at any point in the algorithm. Within* $2|\Omega| - 1$ *iterations, the algorithm either terminates, or* $\psi_{\min}$ *increases.*

Proof. There are $|\Omega|$ scenarios. Let Ω_1 denote the set of scenarios for which $\psi_i = \psi_{\min}, i \in \Omega$, and let Ω_2 denote $\Omega - (\Omega_1 \cup \Omega')$. If follows that $|\Omega_1| + |\Omega_2| \leq |\Omega|$ and that $\psi_i > \psi_{\min}$ for $i \in \Omega_2$.

In each iteration of the algorithm, a scenario $j \in \Omega_2$ is selected to transfer resources to a scenario $k \in \Omega_1$. If the algorithm does not terminate in Step 2 or in Step 3, then one of the following occurs in Step 4:

If (3.12) was satisfied, it means that $\psi_k(x_{2k} + \mathsf{M}x_{2k}) > \psi_{\min}$, and scenario k is removed from Ω_1. But scenario k is also added to set Ω_2. If k was the last element of Ω_1, then $\psi_{\min} < \psi_i$ for all $i \in \Omega$, and $\psi_{\min}$ must be redefined to an increased value.

If (3.13) was satisfied, it means that some resources were removed from scenario j such that for the new values of ψ_j and ψ_k, $\psi_j > \psi_k$; but if any more resources are removed from scenario j, then it would be true that $\psi_j \leq \psi_k$. In this case, j is removed from Ω_2. If j was the last element of Ω_2, then in the next iteration Ω_2 is an empty set, and the algorithm terminates.

So in each iteration of the algorithm, either an element is removed from Ω_1 and added to Ω_2, an element is removed from Ω_2, $\psi_{\min}$ increases, or the algorithm terminates. The first of these events can only happen $|\Omega_1|$ times before $\psi_{\min}$ must be increased. The second of these can happen at most $|\Omega_1| + |\Omega_2|$ times (once for each element of Ω_2, and once for each element of Ω_1 that may be moved into Ω_2) before the algorithm must terminate. If all of the scenarios are members of Ω_1, then $\psi_j = \psi_k$ in Step 2 and the algorithm must terminate, so as a worst case if the algorithm is to proceed, then the largest possible size of the set Ω_1 is $|\Omega| - 1$. Therefore the most iterations that can proceed before the algorithm either terminates or produces an increase in $\psi_{\min}$ is $2|\Omega| - 1$. ∎

The meaning of the set Ω' in the algorithm is now clear. The set Ω' is the set of scenarios whose resource price function values are greater than or equal to $\psi_{\min}$, but to whom resources cannot be removed without causing their resource price function values to fall below $\psi_{\min}$. From a graphical perspective, Ω' represents the set of scenarios $\{\omega \in \Omega : z_3^{\omega}(x_{2\omega})$ is known to be at[2] a vertex$\}$, and any decrease in the allocation of resources to scenario ω would result in a decrease in the resource price function below $\psi_{\min}$. Figure 3.1 provides an illustration of what it means for a scenario to be in Ω'. It can now be proved that the algorithm terminates in a finite number of iterations.

Theorem 3.1 *If Algorithm 3.1 is applied to a problem with a finite number of scenarios* $(|\Omega| < \infty)$*, then the algorithm terminates in a finite number of iterations.*

Proof. The resource price function for each scenario is an increasing step function with a finite number of pieces. From Lemma 3.2, for a given $\psi_{\min} := \min_{\omega} \psi_{\omega}(x_{2\omega})$ at a particular point in the algorithm, if the algorithm has not terminated after $2|\Omega|$ iterations, then $\psi_{\min}$ has increased. Since $|\Omega|$ is finite and since each resource price function is composed of a finite number of pieces, $\psi_{\min}$ can only increase a finite number of times before the algorithm must terminate. ■

Theorem 3.2 *Let* $(x_2^{\delta}, x_3^{\omega\delta})$ *denote the terminal solution of Algorithm 3.1 when a given value of* δ *is used in Step 3 of the algorithm. Then*

$$\lim_{\delta \to 0} (x_2^{\delta}, x_3^{\omega\delta})$$

is the optimal solution to the large subproblem (3.5).

Proof. Let $(x_2', {x_3'}^{\omega}) := \lim_{\delta\to 0}(x_2^{\delta}, x_3^{\omega\delta})$ denote the solution obtained at termination of Algorithm 3.1. If $(x_2', {x_3'}^{\omega})$ is not optimal, then there must exist a $\mathtt{M}x_2 \neq 0$ such

[2] Or, for a practical implementation, $x_{2\omega}$ is within $\dfrac{\delta}{\alpha_{\omega}}$ of a vertex

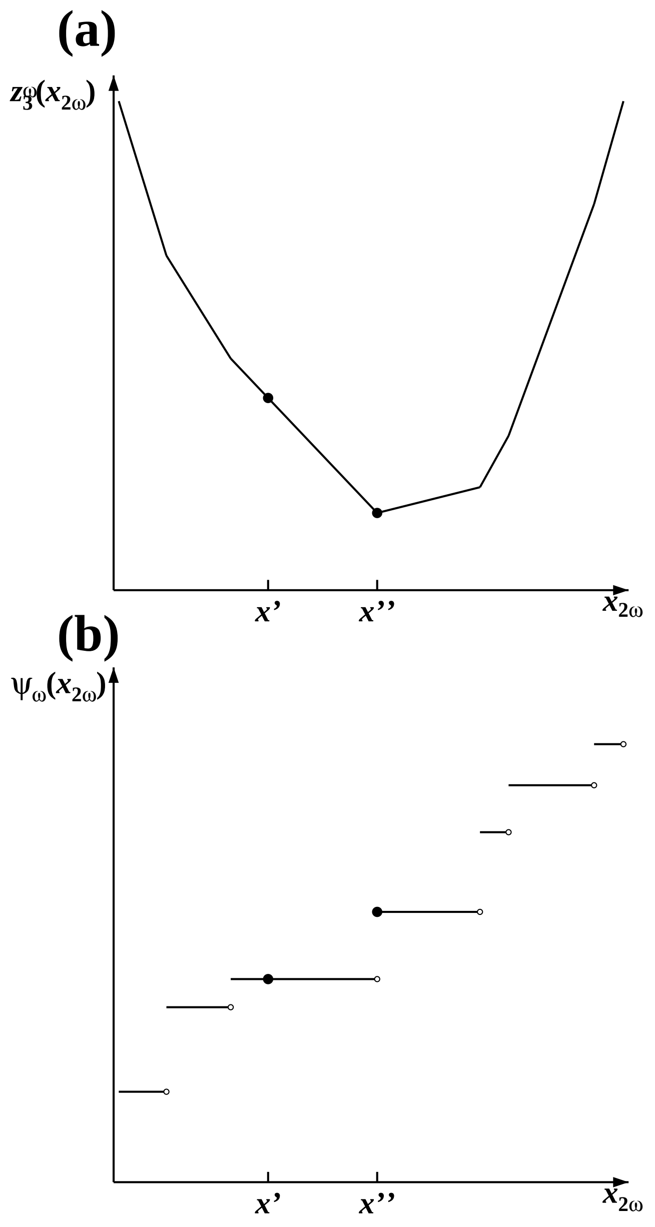

Figure 3.1: **(a)** Sample optimal objective function value as a function of $x_{2\omega}$. **(b)** Resource price function associated with (a). The point at x' represents a point that is not at a vertex of $z_3^\omega(x_{2\omega})$. x' can be decreased without decreasing the resource price function value. The point at x'' is at a vertex of $z_3^\omega(x_{2\omega})$. If x'' is decreased, then the resource price function must decrease. If $x_{2\omega} = x'$, then $\omega \notin \Omega'$. If $x_{2\omega} = x''$, then depending on the current value of $\psi_{\min}$, ω might be $\in \Omega'$.

that

$$\begin{array}{rcl} \alpha \mathsf{M} x_2 & = & 0 \\ c_2 \mathsf{M} x_2 + \sum_\omega p_\omega [z_3^\omega(x_2' + \mathsf{M} x_2) - z_3^\omega(x_2')] & < & 0. \end{array} \tag{3.19}$$

$z_3^\omega(x_2' + \mathsf{M}x_2)$ is piecewise-linear and convex in $\mathsf{M}x_2$. $c_2\mathsf{M}x_2$ is also linear (and thus convex) in $\mathsf{M}x_2$, so $c_2\mathsf{M}x_2 + \sum_w p_\omega[z_3^\omega(x_2' + \mathsf{M}x_2) - z_3^\omega(x_2')]$ is also convex in $\mathsf{M}x_2$. By this convexity property, if $c_2\mathsf{M}x_2 + \sum_w p_\omega[z_3^\omega(x_2' + \mathsf{M}x_2) - z_3^\omega(x_2')] > 0$, then it must also be true that $c_2\lambda\mathsf{M}x_2 + \sum_w p_\omega[z_3^\omega(x_2' + \lambda\mathsf{M}x_2) - z_3^\omega(x_2')] > 0$ for any λ such that $0 < \lambda \leq 1$, including infinitesimally small values of λ. In that case the expression

$$\sum_w p_\omega[z_3^\omega(x_2' + \lambda\mathsf{M}x_2) - z_3^\omega(x_2')] \tag{3.20}$$

may be replaced with the appropriate dual values. If $\mathsf{M}x_{2\omega} > 0$, then

$$z_3^\omega(x_{2\omega} + \mathsf{M}x_{2\omega}) - z_3^\omega(x_{2\omega}) = \mathsf{M}x_{2\omega}\pi^\omega a_\omega = \mathsf{M}x_{2\omega}\psi_\omega(x_{2\omega}). \tag{3.21}$$

If $\mathsf{M}x_{2\omega} < 0$, then

$$z_3^\omega(x_{2\omega} + \mathsf{M}x_{2\omega}) - z_3^\omega(x_{2\omega}) = \mathsf{M}x_{2\omega}\pi^\omega a_\omega = \mathsf{M}x_{2\omega}\psi_\omega(x_{2\omega} - \mathsf{M}x_{2\omega}). \tag{3.22}$$

(Recall that ψ_ω indicates the value of increasing the allocation to scenario ω. To evaluate what happens when resources are deallocated, the resource price function ψ_ω must be evaluated at its proposed final allocation instead of its initial allocation.) And of course if $\mathsf{M}x_{2\omega} = 0$ then

$$z_3^\omega(x_{2\omega} + \mathsf{M}x_{2\omega}) - z_3^\omega(x_{2\omega}) = 0. \tag{3.23}$$

For the given solution $\mathsf{M}x_{2\omega}$ to (3.19), let $S_1 := \{\omega \in \Omega : \mathsf{M}x_{2\omega} > 0\}$ and let $S_2 := \{\omega \in \Omega : \mathsf{M}x_{2\omega} < 0\}$. $\sum_\omega \alpha_\omega \mathsf{M}x_{2\omega} = 0$ and $\mathsf{M}x_2 \neq 0$, so both S_1 and S_2 must be non-empty with

$$\sum_{\omega\in S_1} |\alpha_{2\omega}\mathtt{M}x_{2\omega}| = \sum_{\omega\in S_2} |\alpha_{2\omega}\mathtt{M}x_{2\omega}|. \tag{3.24}$$

Now it will be shown that if any $\mathtt{M}x_2$ exists that satisfies (3.19) and (3.24), then a solution must exist where S_1 and S_2 each contain exactly one element. Such a solution $\mathtt{M}x_2'$ can be constructed as follows:

For any solution $\mathtt{M}x_2$ to (3.19) and (3.24), define $\varphi > 0$ as

$$\varphi \texttt{,}\ \ c_2\mathtt{M}x_2 + \sum_\omega p_\omega(z_3^\omega(x_2 + \mathtt{M}x_2) - z_3^\omega(x_2)). \tag{3.25}$$

Choose any $i \in S_1$ and $j \in S_2$. Let $\mathtt{M}x_{2i}' := \max[\mathtt{M}x_{2i}, -\mathtt{M}x_{2j}\frac{\alpha_j}{\alpha_i}]$ and let $\mathtt{M}x_{2j}' := \min[\mathtt{M}x_{2j}, -\mathtt{M}x_{2i}\frac{\alpha_i}{\alpha_j}]$. Also let $\mathtt{M}x_{2k}' := 0$ for all $k \in \Omega \,\mathtt{r}\, \{i,j\}$. By construction, $\mathtt{M}x_2'$ satisfies $\alpha\mathtt{M}x_2' = 0$.

Let $\varphi' := c_2\mathtt{M}x_2' + \sum_\omega p_\omega(z_3^\omega(x_2' + \mathtt{M}x_2') - z_3^\omega(x_2'))$. If $\varphi' > 0$, then $\mathtt{M}x_2'$ satisfies (3.19) and (3.24), and also satisfies the desired condition of having exactly one element in each of S_1 and S_2.

If $\varphi' \leq 0$, then let $\mathtt{M}x_2 := \mathtt{M}x_2 - \mathtt{M}x_2'$ and $\varphi := \varphi - \varphi'$. Now $\mathtt{M}x_2$ still satisfies (3.19) and (3.24). The new solution $\mathtt{M}x_2$ has at least one fewer nonzero element than the old solution, but it still must have at least one element in S_1 and S_2. This procedure may be repeated until either a $\varphi' > 0$ is found, or until there are exactly two nonzero elements of $\mathtt{M}x_2$ remaining. In the latter case, φ must still be greater than 0, so this $\mathtt{M}x_2$ is a valid two-element solution to (3.19) and (3.24), with $\mathtt{M}x_{2i} > 0$ for exactly one element of $i \in \Omega$ and $\mathtt{M}x_{2j} < 0$ for exactly one element of $j \in \Omega$. Since this solution satisfies (3.19),

$$\mathtt{M}x_{2i} = -\mathtt{M}x_{2j}\frac{\alpha_j}{\alpha_i}. \tag{3.26}$$

Using an infinitesimal positive multiple of $\mathtt{M}x_2$, it follows that

$$\begin{aligned}
& c_2\mathtt{M}x_2 + \textstyle\sum_\omega p_\omega(z_3^\omega(x_2+\mathtt{M}x_2) - z_3^\omega(x_2)) \\
= \;& c_{2i}\mathtt{M}x_{2i} + c_{2j}\mathtt{M}x_{2j} + p_i z_3^i(x_{2i}+\mathtt{M}x_{2i}) - p_i z_3^i(x_{2i}) \\
& \quad + p_j z_3^j(x_{2j}+\mathtt{M}x_{2j}) - p_j z_3^j(x_{2j}) \\
= \;& (c_{2i} + p_i\pi^i(x_{2i})a_i)\mathtt{M}x_{2i} + (c_{2j} + p_j\pi^j(x_{2j} - |\mathtt{M}x_{2j}|)a_j)\mathtt{M}x_{2j} \\
= \;& \alpha_i\left[\frac{c_{2i}+p_i\pi^i(x_{2i})a_i}{\alpha_i}\mathtt{M}x_{2i} + \frac{c_{2j}+p_j\pi^j(x_{2j}-|\mathtt{M}x_{2j}|)a_j}{\alpha_i}\mathtt{M}x_{2j}\right] \\
= \;& \alpha_i\left[\frac{c_{2i}+\pi^i(x_{2i})a_i}{\alpha_i}\mathtt{M}x_{2i} - \frac{c_{2j}+\pi^j(x_{2j}-|\mathtt{M}x_{2j}|)a_j}{\alpha_j}\mathtt{M}x_{2i}\right] \\
= \;& \alpha_i\mathtt{M}x_{2i}\left[\psi_i(x_{2i}) - \psi_j(x_{2j}-|\mathtt{M}x_{2j}|)\right].
\end{aligned} \tag{3.27}$$

$\alpha_i\mathtt{M}x_{2i}$ is positive. At the termination of the algorithm, by construction $\psi_i(x_{2i}) \geq \psi_{\min}$ and $\psi_j(x_{2j} - |\mathtt{M}x_{2j}|) \leq \psi_{\min}$, so the quantity in (3.19) is necessarily nonnegative. Thus a contradiction exists, and at the termination of the algorithm, there can be no solution $\mathtt{M}x_2$ to (3.19) where $\mathtt{M}x_{2i} > 0$ for exactly one $i \in \Omega$ and $\mathtt{M}x_{2j} < 0$ for exactly one $j \in \Omega$. Then there can be no solution $\mathtt{M}x_2 \neq 0$ to (3.19). And therefore the solution at the end of the algorithm is the optimal solution. ■

Corollary 3.1 *Let $\psi_{\max}$ and $\psi_{\min}$ denote the maximum and minimum resource price function values at the termination of Algorithm 3.1. Then for any choice of $\delta > 0$ in Step 3, the terminal objective function value $z_2(x_2^\delta, x_3^{\omega\delta})$ is optimal for (3.5) to within $|\Omega|\delta(\psi_{\max} - \psi_{\min})$.*

Proof. Corollary (3.1) can be seen to be true by the following argument. At the termination of the algorithm,

$$\psi_{\min} \leq \quad \psi_\omega(x_{2\omega}) \quad \leq \psi_{\max} \tag{3.28}$$

$$\psi_\omega(x_{2\omega} - \delta_\omega) \quad \leq \psi_{\min}, \tag{3.29}$$

where $\delta_\omega := \frac{\delta}{\alpha_\omega}$, and for all $\omega \in \Omega$. Let us find two scenarios i and j such that $\psi_i = \psi_{\max}$ and $\psi_j = \psi_{\min}$. The allocation for scenario i, x_{2i}, is "near" (within $\delta_i := \dfrac{\delta}{\alpha_i}$) but not necessarily "at" a vertex of $z_3^i(x_{2i})$. The optimal objective function value may be improved by moving the allocation of scenario i to a vertex.

Suppose the allocation for scenario i lies exactly $\delta_i' \leq \delta_i$ from a vertex such that $\psi_i(x_{2i} - \hat{\delta}) \geq \psi_{\min}$ for $\hat{\delta} < \delta_i'$. Then the amount to deallocate from scenario i is exactly δ_i' units of resources. To maintain the feasibility of the expected value constraint, this deallocation must be accompanied by a reallocation of $\delta_j' := \delta_i' \dfrac{\alpha_i}{\alpha_j}$ resources to scenario j.

The resulting change in the objective function after this transfer of resources is $\mathtt{M}\, z_2(x_2) := \alpha_i \delta_i' \psi_i(x_{2i}) - \alpha_j \delta_j' \tilde{\psi}_j$. Here, $\tilde{\psi}_j$ represents a linear combination of all the resource price function values for scenario j that occur between x_{2j} and $x_{2j} + \delta_j'$. $\psi_j(x_{2j}) = \psi_{\min}$, and $\psi_j(x_{2j} + \delta_j') \geq \psi_{\min}$, so by a convexity argument $\tilde{\psi}_j \geq \psi_{\min}$. $\psi_i(x_{2i}) = \psi_{\max}$ and $\delta' := \alpha_i \delta_i' = \alpha_j \delta_j' \leq \delta$, so therefore

$$\mathtt{M}\, z_2(x_2) \leq \delta(\psi_{\max} - \psi_{\min}). \tag{3.30}$$

After the transfer of resources, the allocation for scenario i is now at a vertex, and no further improvement is possible for that scenario. These additional sub-delta transfers can occur no more than $|\Omega|$ times before all scenario allocations are at vertices and absolutely no further improvement is possible. ■

3.4 A Modified Benders Decomposition Algorithm

Given an efficient technique for solving the large subproblem (3.5) that simultaneously solves for (x_2, x_3^ω), the technique can be included into a Benders decomposition algorithm that can solve the whole expected value constrained problem (3.9). The following modified Benders decomposition algorithm performs that task.

Algorithm 3.2 Modified Benders decomposition for solving the full problem (3.9)
Step 1. Obtain a feasible solution (x_1) to the relaxed master problem

$$\begin{aligned} A_{11}x_1 &= b_1 \\ x_1 &\geq 0. \end{aligned} \tag{3.31}$$

Step 2. Using the solution x_1 obtained in the previous step, solve the large subproblem

$$\begin{array}{llcrcll} \text{Minimize} & c_2x_2 & + & \sum_\omega p_\omega c_3^\omega x_3^\omega & = & z_2 - c_1x_1 & \\ \text{subject to} & A_{22}x_2 & & & = & b_2 + A_{12}x_1 & : \rho \\ & -A_{23}^\omega x_2 & + & A_{33}^\omega x_3^\omega & = & b_3^\omega + A_{13}^\omega x_1 & : \pi^\omega \qquad \forall\, \omega \in \Omega \\ & x_2, & & x_3^\omega & \geq & 0 & \end{array} \tag{3.32}$$

using Algorithm 3.1. By choosing an appropriate value for δ in Step 3 of Algorithm 3.1, the user can use the results of corollary (3.1) to solve the large subproblem to arbitrary accuracy. If an unbounded solution was returned, then mark the problem as unbounded and stop. If a feasible solution was returned, record the dual solution (ρ, π^ω) and the optimal objective value z_2 to this problem.

Step 3. If the preceding large subproblem was infeasible, find one scenario $\omega \in \Omega$ for which the scenario subproblem (3.2) cannot be made feasible. Record the extreme dual vector (π^ω) and add the feasibility cut

$$\pi^\omega A_{13}^\omega x_1 + \pi^\omega(b_3^\omega + A_{23}^\omega x_2) \leq 0 \tag{3.33}$$

to the master problem. If the subproblem was feasible, record the optimal dual values (ρ, π^ω) and add the optimality cut

$$-\left[\sum_\omega (p_\omega \pi^\omega) A_{13}^\omega + \rho A_{12}\right] x_1 + \theta \geq \sum_\omega (p_\omega \pi^\omega) b_3^\omega + \rho b_2 \tag{3.34}$$

to the master problem.

Step 4. Solve the master problem

$$\begin{array}{llcrcll} \text{Minimize} & c_1x_1 & + & \theta & = & z_1 & \\ \text{subject to} & A_{11}x_1 & & & = & b_1 & \\ & -Gx_1 & + & \theta & \geq & g & \text{optimality cuts} \\ & -G'x_1 & & & \geq & g' & \text{feasibilty cuts} \\ & x_1 & & & \geq & 0 & \end{array} \tag{3.35}$$

to obtain a new intermediate optimal solution (x_1, θ) and intermediate objective function value z_1. If the master problem is infeasible, then record the problem as infeasible and stop. If the master problem was feasible with $z_1 = z_2$, then record (x_1, x_2, x_3^ω) as the optimal solution to the problem and stop. If the master problem was feasible with $z_1 > z_2$, then return to Step 2 with the new intermediate first-stage solution x_1.
/

Some additional comments about this algorithm are in order. In Step 3, a feasibility cut may be needed to restrict the feasible region of x_1 so that a feasible second-stage solution might be found. When the feasibility cut

$$\pi^\omega A_{13}^\omega x_1 + \pi^\omega(b_3^\omega + A_{23}^\omega x_2) \leq 0 \tag{3.36}$$

is made, a current value for x_2 that produced the infeasible scenario subproblem should be substituted in place of the x_2 in the above expression. An explicit x_2 term should *not* appear in the feasibility cut in the master problem.

If an optimality cut is needed, then the dual value ρ of the expected value constraint in the large subproblem needs to be obtained. This value is never determined explicitly in the context of solving the large subproblem with the simplex method. But its value can readily be obtained by realizing that ρ represents the increase in the objective function that would be obtained by increasing the right-hand side of the expected value constraint $b_2 + A_{12}x_1$. Keep in mind the metaphor of using the expected value constraint to decide how to allocate some resource among the different scenarios. The right-hand side of the expected value constraint represents the total amount of resources available for allocation to the scenarios. Then ρ is simply a measure of the change in the objective function value per change in the total amount of resources to allocate. If additional resources did become available (i.e., if b_2 were to increase), then the sensible thing to do would be to allocate the new resources to the scenario(s) such that it would have the greatest beneficial effect on the objective function value. These scenarios can be identified by their resource price function (3.10) values, and hence the value for ρ must be

$$\rho = \min_{\omega \in \Omega} \psi_\omega. \tag{3.37}$$

3.4.1 Implementation in DECIS

The DECIS software package [11] used to produce the results in the previous chapter is designed for two-stage linear stochastic optimization problems that may be cleanly decomposed into a master problem such as (3.1) or (3.4) and independent subproblems such as (3.2). DECIS required some modifications in order to handle problems that are decomposed into a master problem (3.4) and a large subproblem such as (3.5). This section describes the nature of these modifications.

Partitioning the problem

For expected value constrained problems that satisfy the required conditions for Algorithm 3.1, the input files to DECIS are configured such that the master problem resembles (3.1) and the independent subproblems resemble (3.2). That is, the same input files are used to solve the problem with the modified Benders decomposition algorithm (3.2) as are used to solve the reformulated expected value constrained problem. While solving the relaxed master problem in the first iteration and the master problem with cuts in subsequent iterations, DECIS determines a solution for both x_1, the original first-stage decision variables, and x_2, the problematic variables represented in the expected value constraint.

Implementing Algorithm 3.1

Program control then moves to the modified portion of DECIS that solves the large subproblem. The values of x_1 from the solution of the master problem are passed to this section of the program. The values for x_2 from the solution of the master problem are discarded in favor of the initial allocation

$$x_{2\omega} = \frac{b_2 + A_{12}x_1}{\sum_\omega \alpha_w} \qquad \forall\, \omega \in \Omega. \tag{3.38}$$

This allocation is a feasible for the expected value constraint, and it usually provides a better starting point for the algorithm than the solution for x_2 generated by DECIS.

To continue with the implementation of Algorithm 3.1, the program obtains dual solutions to all the subproblems and computes ψ_ω for all scenarios. The program proceeds to determine candidates repeatedly for transferring resources and calculates how resources are to be transferred until the termination conditions of the algorithm are reached. With the given value of x_1 and the optimal value of x_2 for that choice of x_1, all of the subproblems are solved again in order to obtain the optimal dual values π^ω. The resource price functions are evaluated one more time to help determine $\rho = \min_{\omega\in\Omega} \psi_\omega$.

Adjusting the cuts

Some further adjustments are necessary to apply an optimality cut to the master problem. As far as DECIS is concerned, intermediate values for x_2 are decided in the master problem, and those variables should be represented in the cuts. But for the purposes of the modified Benders decomposition algorithm, the values for x_2 in the master problem are ignored except to guarantee that values for x_1 are chosen that allow the expected value constraint to be feasible.

Once an optimal solution $(x_2, x_3^\omega, \rho, \pi^\omega)$ has been determined for the large subproblem, the following adjustments must be made to create a cut suitable for application to the master problem in DECIS:

1. As DECIS iterates through the scenarios for the final time, it keeps a running

total of the quantities $\sum_\omega p_\omega \pi^\omega b_3^\omega$, $\sum_\omega p_\omega \pi^\omega A_{13}^\omega$, and $\sum_\omega p_\omega \pi^\omega A_{23}^\omega$. The last of these three quantities can be ignored.

2. When $\rho := \min_\omega \psi_\omega$ has been determined, the scalar quantity ρb_2 and vector quantity ρA_{12} must be calculated.

3. The cut that is finally added to the master problem has the form

$$-(\rho A_{12} + \sum_\omega p_\omega \pi^\omega A_{13}^\omega) x_1 + \theta \geq \rho b_2 + \sum_\omega p_\omega \pi^\omega b_3^\omega. \tag{3.39}$$

Once the cut has been applied to the master problem, a new iteration begins and the master problem is solved again.

3.5 Numerical Results

To demonstrate the practical application of the modified Benders decomposition algorithm, a variety of supply-chain problems from HP with different risk management techniques were solved under many different demand distributions. These problems were solved and with Problems were solved with a modified version of DECIS [11] that implements the modified Benders decomposition algorithm from the previous section. Results of these experiments are presented below, including the characteristic size of the model, the number of demand scenarios used, the number of Benders iterations required to achieve a solution, the number of scenario subproblems that were solved to achieve a solution, and the CPU time in seconds needed to solve the problem on a high-end Silicon Graphics workstation.

In the modified Benders decomposition implementation, the value for *delta* (see Step 3 of Algorithm 3.1) was selected to be 0.01 times the probability of the least likely stochastic outcome. This value was thought to represent a good tradeoff between accuracy (using higher values of *delta* makes the solutions to the large subproblems less accurate) and computational effort (a lower value of *delta* requires extra subproblems to be solved, increasing computational effort).

	Conventional model		EV constrained model			
			Reformulated soln.		Modified Benders	
Problem no. of scen.	soln. time	MP/SP	soln. time	MP/SP	soln. time	MP/SP
tom_{1K}–5	0.08	2/10	0.08	2/10	0.21	2/71
tom_{1K}–25	0.36	13/109	0.53	8/200	0.85	2/392
tom_{1K}–100	0.88	13/409	7.40	32/3200	7.76	12/2917
tom_{10K}–5	0.31	22/58	0.42	28/75	2.10	16/1052
tom_{10K}–25	0.68	28/220	3.89	93/1802	8.09	13/4344
tom_{10K}–100	2.26	31/1120	598.47	2094/2.1E5	33.99	14/1.7E4
tom_{29K}–5	0.37	25/73	0.70	51/185	1.14	17/514
tom_{29K}–25	0.97	33/393	18.49	343/7639	16.29	18/8997
tom_{29K}–100	2.88	33/1518	1431.93	5615/5.5E5	58.88	21/3.1E4
bap_{10K}–5	122.72	425/2121	146.02	448/2230	132.47	421/4206
bap_{10K}–25	168.16	423/1.1E4	923.42	999/2.5E4	240.56	425/2.1E4
bap_{10K}–100	310.03	420/4.2E4	6698*	2335/2.2E5	530.09	425/8.5E4
bap_{1M}–5	283.41	647/3231	497.76	940/4690	335.61	649/6486
bap_{1M}–25	347.09	647/1.6E4	1980.37	1902/4.6E4	507.45	648/3.2E4
bap_{1M}–100	565.15	647/6.5E4	††	††/††	952.93	649/1.3E5
bap_{5M}–5	346.57	673/3361	524.89	900/4490	992*	665/7460
bap_{5M}–25	473.53	852/2.1E4	3243*	2532/6.0E4	799.95	867/4.4E4
bap_{5M}–100	893.57	925/9.2E4	††	††/††	1512.79	960/2.0E5
dlaw_{5K}–5	5377*	4081/2.0E4	11542*	6102/3.0E4	6069*	4125/4.1E4
dlaw_{5K}–25	7684*	4292/1.1E5	††	††/††	9793*	4199/2.1E5
dlaw_{200K}–5	22297*	1.4E4/6.8E4	††	††/††	52143*	1.7E4/6.2E5
dlaw_{200K}–25	30075*	1.5E4/3.7E5	††	††/††	78341*	1.5E4/1.5E6

Table 3.1: Results for HP Supply Chain Models with conventional risk management constraints, expected value constraints solved by reformulation and conventional Benders decomposition, and expected value constraints solved with the modified Benders decomposition algorithm. Except where noted, solution time refers to CPU seconds of a single processor needed to solve the problem.

† Characteristic size of these problems:

Problem	**MP size**	**SP size**	**Deterministic size** with $\lvert\Omega\rvert = 100$
tom	39 × 39	163 × 149	16339 × 14939
bap	265 × 265	548 × 817	54965 × 81965
dlaw	378 × 383	2844 × 4980	284748 × 498382

†† A solution was not obtained for this problem in fewer than 25 hours or 25000 iterations.

* Solution time is reported in real seconds, not CPU seconds — this is a minor overestimation of the CPU time needed to solve the problem.

3.6 Discussion of Numerical Results

The previous section provides a demonstration that the modified Benders decomposition algorithm presented in this chapter can be used to solve expected value constrained problems. Where a direct comparison can be made between using the modified Benders decomposition of this chapter and the reformulated problem of the previous chapter to solve an expected value constrained problem, the number of iterations and amount of computational effort needed to solve a problem are typically much smaller for the modified Benders decomposition.

Using the modified Benders decomposition on the expected value constrained problem takes about the same number of iterations as regular Benders decomposition on the conventionally constrained problem. But the modified algorithm typically needs to solve many more subproblems. In the conventional problem, the ratio of the number of subproblems solved to the number of master problems solved is related to the number of scenarios. In the modified Benders decomposition, that ratio is typically much higher because the scenario subproblems are all solved many times in the context of solving the large subproblem (3.5).

The computational effort needed to solve the expected value constrained problem with the modified Benders decomposition is larger than the effort needed to solve a similar conventionally constrained problem, but the effort is comparable. In most cases, if the conventional problem can be solved in a reasonable amount of time, then the expected value constrained problem can be, too. With the modified Benders decomposition algorithm, many useful expected value constrained problems are now practical to solve.

Chapter 4

Extensions and Future Research

The algorithm that was presented in Chapter 3 was applicable to only a subset of stochastic programs with expected value constraints, namely, those with a single equality expected value constraint, and with a single representative from each scenario in the expected value constraint. This class of expected value constrained problems contains many useful examples that can be solved by the method of the previous chapter. But other interesting expected value constrained problems do not satisfy these conditions, and other techniques must be considered.

This short chapter discusses how some of the conditions from the previous chapter may be relaxed. Finally, the new knowledge about expected value constraints is summarized.

4.1 Inequalities in Expected Value Constraints

The case where the expected value constraint is an inequality is straightforward enough to deal with. The inequality can be converted to an equality with the introduction of an appropriate slack variable; that is,

$$\sum_{i \in \Omega} \alpha_{2i} x_{2i} \leq \hat{x}_2 \tag{4.1}$$

becomes

$$x_{20} + \sum_{i \in \Omega} \alpha_{2i} x_{2i} = \hat{x}_2. \tag{4.2}$$

The problematic variable $x_{20} \geq 0$ can be thought of as the resources that are allocated to an artificial scenario that has no effect on the subproblem objective function value. All that is left is to define an appropriate resource price function for this artificial scenario. Since transferring resources to and from this artificial scenario has no direct effect on the large subproblem objective function value, an appropriate function is

$$\psi_0(x_{20}) = \begin{cases} \infty & x_{20} < 0 \\ 0 & x_{20} \geq 0. \end{cases} \tag{4.3}$$

$x_{20} < 0$ is not a feasible value for this slack variable, so there is a substantial penalty for choosing $x_{20} \geq 0$ (that is, a strong incentive to add resources to artificial scenario 0 when $x_{20} < 0$).

If (4.1) were a $\geq$ constraint rather than a $\leq$ constraint, the appropriate artificial resource price function would be

$$\psi_0(x_{20}) = \begin{cases} -\infty & x_{20} > 0 \\ 0 & x_{20} \leq 0. \end{cases} \tag{4.4}$$

In this case whenever $x_{20} > 0$, there is a substantial incentive for the algorithm to remove resources from the artificial scenario 0.

4.2 Expected value constraints with more than one representative from some scenarios

It may be that there is more than one nonzero coefficient from some scenarios in the expected value constraint.

Example 4.1 A contrived example of this situation is the HP supply chain problem where a lower bound is placed on the sum of the expected profit and the expected inventory reduction. That is, the constraint

$$\sum_{\omega \in \Omega} p_\omega (P^\omega + R^\omega) \geq \hat{Q} \tag{4.5}$$

is added to the model (2.1.2). Or more generally,

$$\sum_{\omega \in \Omega} \alpha_P^\omega P^\omega + \alpha_R^\omega R^\omega \geq \hat{Q}. \tag{4.6}$$

/

In this instance there are two resource price functions for scenario i—one indicates the relative value of increasing P^i, and the other indicates the relative value of increasing R^i. Conceivably, one could simultaneously increase the profit target for scenario i and reduce the inventory reduction target for scenario i, and hence improve the subproblem objective function value just by transferring resources within scenario i.

A workaround is to define a set of artificial problematic variables and to remove the other variables from the expected value constraint. In the contrived example, one may define

$$Q^\omega := \alpha_P^\omega P^\omega + \alpha_R^\omega R^\omega. \tag{4.7}$$

Then the expected value constraint (4.6) may be rewritten as

$$\sum_{\omega \in \Omega} Q^\omega \geq \hat{Q}, \tag{4.8}$$

and the additional constraint

$$\alpha_P^\omega P^\omega + \alpha_R^\omega R^\omega = Q^\omega \tag{4.9}$$

can be added to the independent subproblems. Now the expected value constraint and the independent subproblems have the necessary form for the application of the modified Benders decomposition algorithm from the previous chapter.

4.3 More than one expected value constraint

Some of the concepts of the modified Benders decomposition algorithm from the previous chapter can still be applied to problems where the conditions for using the algorithm do not hold. For example, in the previous chapter two scenarios were selected for a transfer of resources. An optimization problem was solved to determine how many resources to transfer. After the transfer, the expected value constraint was still satisfied, and the transfer resulted in an improved objective function value for the large subproblem.

In the case where there is more than one expected value constraint, all of these concepts are still applicable. The only problematic issue is with the first step, where two scenarios are singled out for a transfer of resources. When there is only a single expected value constraint in a problem, then there is a unique resource price function for each scenario. The scenarios with the highest and the lowest values of this function are selected for transferring resources. When there is more than one expected value constraint, then each scenario has more than one resource price function. Alternatively, one could say that the resource price function maps a set of decision variables x_2 to an m_2-dimensional vector, where m_2 is the number of expected value constraints in the problem.

For ordered tuples of higher order than one, there is not necessarily a logical ordering for the function values of each scenario, and there may not be an obvious method for selecting those scenarios. Furthermore, there is not necessarily a way to transfer resources between two scenarios while still satisfying more than one expected value constraint. It may be necessary to select and transfer resources between as many as $m_2 + 1$ scenarios to simultaneously satisfy m_2 expected value constraints.

In any case, if a suitable set of scenarios is determined, an optimization problem can be constructed whose solution determines how to transfer resources between these scenarios. The problem would consist of all of the scenario subproblems for the chosen set of scenarios. There would also be an additional constraint for each expected value constraint—these constraints would assert that each expected value constraint remains satisfied after a transfer of resources. As a brute force algorithm, one could try every possible permutation consisting of $m_2 + 1$ scenarios and attempt to move resources among those scenarios. After a sequence in which every possible permutation was tried without improving the large subproblem objective, it could be said that the optimal objective solution to the large subproblem had been achieved. More research is necessary to discover if there is a better way of determining the best order in which to attempt to transfer resources, or if a general optimality condition exists that would allow the algorithm to terminate without trying all possible permutations of the scenarios.

As a last resort for the most general of expected value constrained problems, the nested Benders decomposition or nested Dantzig-Wolfe decomposition algorithms presented in the previous chapter may be applied.

4.4 Summary

Expected value constraints provide another way to apply risk management to stochastic programs. They have advantages over other constraint-based methods in that the feasible region of an expected value constrained problem is not as dependent on the worst-case scenarios as other types of constraints. Their main disadvantage is that an expected value constraint wrecks the dual-angular structure of a stochastic program, so large problems cannot make full use of Benders decomposition in order to be solved. A reformulation of expected value constrained problems exists that does have a dual-angular structure, but Benders decomposition tends to perform poorly on those problems, so many useful expected value constrained problems are not practical to solve that way.

A modified Benders decomposition algorithm exists that has many advantages over the reformulation of the problem. The algorithm may be applied under certain conditions, including a limit of one expected value constraint in the problem. In this algorithm, the expected value constraint is solved for simultaneously with the independent scenario subproblems, and typically requires many fewer iterations than the conventional Benders decomposition with the reformulated problem. This algorithm makes many larger expected value constrained problems practical to solve. The computational effort needed to solve problems with this approach is often not much more than the effort needed to solve the problems without the expected value constraint.

More research can be done on extending the algorithm to cover a larger class of problems, especially problems with two or more expected value constraints. But the work presented in this thesis demonstrates that problems with expected value constraints are useful, and in many cases practical to solve.

Bibliography

[1] Benders, J.F., “Partitioning procedures for solving mixed-variables programming problems,” *Numerische Mathematik* **4**:238–252 (1962).

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

[3] Brooke, A., Kendrick, D., and Meeraus, A., *GAMS : Release 2.25 : A User’s Guide*, Scientific Press, 1992.

[4] Charnes, A. and Cooper, A.A., “Chance Constrained Programming”, *Management Science* **6** 73–79 (1959).

[5] CPLEX Optimization, Incline Village. *Using the CPLEX Callable Library and CPLEX Mixed Integer Library*, 1993.

[6] Dantzig, G.B.and Infanger, G., “Intelligent control and optimization under uncertainty with application to hydro power,” *European Journal of Operational Research* **97**:396–407 (1997).

[7] Dantzig, G.B.and Thapa, M.N., *Linear Programming I: Introduction*, Springer-Verlag, 1997.

[8] Dantzig, G.B.and Wolfe, P., “Decomposition principle for linear programs.” *Operations Research*, **8**:101–111, 1960.

[9] Fourer, R., Gay, D.M., and Kernighan, B.W., *AMPL: A Modeling Language for Mathematical Programming*, Duxbury Press / Brooks/Cole Publishing Co., 1993.

[10] Hackney, H.N. (1997). *Solution Techniques for a Class of Large-Scale Stochastic Programs* Ph.D. Dissertation, Dept. of Engineering-Economic Systems and Operations Research, Stanford University.

[11] Infanger, G., DECIS User's Guide (preliminary), Dr. Gerd Infanger, 1590 Escondido Way, Belmont, California 94002 (1997).

[12] Murtagh, B.A., and Saunders, M.A., MINOS 5.4 User's Guide, Report SOL 83-20R, Department of Operations Research, Stanford University (Revised February 1995).

[13] Prékopa, A., "Numerical Solution of Probabilistic Constrained Programming Models", in Ermoliev, Y. and Wets., R.J.-B. (eds.): *Numerical Techniques for Stochastic Optimization*, Springer Verlag, Berlin, 123–139, (1988).

[14] Van Slyke, R.M., and Wets, R.J.-B., "L-shaped linear programs with application to optimal control and stochastic programming," *SIAM Journal on Applied Mathematics* **17**:638–663 (1969).