ABSTRACT

GOSE, ALEXANDER H. Sequential Bounding Methods for Stochastic Programming Models of Production Planning. (Under the direction of Brian Denton.)

This dissertation explores the use of stochastic programming for a variety of production planning problems. We develop methods for approximately solving these by generating representative sets of outcomes. We extend the approach first suggested by Birge (1985a), which we refer to as *sequential bounding*, where scenarios are generated in an iterative fashion based on a deterministic measure of the error in the approximation. This approach is extended to general two-stage and multi-stage stochastic programs, and methods are developed to improve convergence through a sequence of iterations. First, we present new sequential bounding approaches for two-stage stochastic programs and apply these approaches to a single-period production planning problem with downward product substitutions. Next, we present a multi-stage sequential bounding approach, and compare this with several other scenario generation approaches for a production planning problem with uncertain demand and constant work in process inventory. Finally, we develop a multi-stage stochastic program to model the evolution of demand forecasting for production planning with load dependent lead times. We compare a sequential bounding approach to solving this model with other models and solution methodologies.

Sequential Bounding Methods for Stochastic Programming Models of Production Planning

by
Alexander H. Gose

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Operations Research

Raleigh, North Carolina

2013

APPROVED BY:

Yahya Fathi	Karl Kempf
Negash Medhin	Reha Uzsoy

Brian Denton
Chair of Advisory Committee

DEDICATION

To my family for their love, support, and encouragement

and

the memory of my late Aunt Myriam

BIOGRAPHY

Alex Gose holds a BS in Physics and a BA in Mathematics from the University of Cincinnati. After graduating in 2003 with a Master of Operations Research from North Carolina State University, he worked in government contracting and consulting. He returned to NCSU to pursue a PhD in Operations Research in 2009. During this period he worked as a teaching assistant, a research assistant, and an Operations Research Analyst at Intel Corporation. His research interests include consulting, decision making under uncertainty, and semiconductor manufacturing.

ACKNOWLEDGEMENTS

I am deeply grateful for the help and careful guidance of Dr. Brian Denton, my thesis advisor. Our weekly one-on-one meetings over the years were often the highlight of my week. His professionalism, hard work, and positive attitude made working with him a pleasure. I admire him most for his creativity, infectious love for research, and ability to see the impact that a variety of ideas have on practical applications. The timely completion of this dissertation was only possible with his patience and dedication.

I am also thankful for the efforts of my committee members. Dr. Reha Uzsoy has been a tremendous source of inspiration, ideas, and connections to other areas of research. He pushed me to deeply analyse the results of computational experiments and offer plausible explanations for the outcomes, which has improved this dissertation a great deal. I am privileged to have worked under the direction of Dr. Karl Kempf, both as a student and employee at Intel Corporation. After working with him in person and attending weekly telephone meetings with him over the past several years, I hope that his exceptional skill at identifying opportunities, explaining the value of Operations Research to others, and implementing solutions has rubbed off on me. I am also very thankful for the efforts of Dr. Yahya Fathi, who encouraged me to communicate the ideas in this dissertation clearly. I had the privilege of taking several courses in Operations Research from him over the span of many years, and I enjoyed gaining new insights and learning from him and his research. I have benefited a great deal from conversations with Dr. Negash Medhin, who helped me understand stochastic programming from a deeper, mathematical perspective. I am also very grateful for his guidance, emotional support, and service as Co-Director of the Operations Research Program.

I also wish to thank my friends and the faculty, staff, and students at North Carolina State University. I will always have a special fondness for the Operations Research Program, the friendly environment, and the academic freedom that I have enjoyed so much. I regret that I cannot list the names of everyone who has contributed to my education while in the program.

There are many people who helped me and whose friendship I will always value, but I will acknowledge some individuals who have had a direct impact on my research. Many thanks to Dr. Amirhosein Norouzi and Dr. Erinc Albey, who sacrificed a number of hours over several weekends to help me understand their work and pore over printouts of scenario trees for problem instances from Chapter 5. I am also grateful to have worked with Dr. S. Ayca Erdogan, along with my advisor. Our work with online appointment scheduling for clinics in the health care industry was my first exposure to academic research. Thanks to Daniel Underwood, whose help with computational experiments on our shared Linux server was greatly appreciated. Last, but not least, I am sincerely grateful for the time, service, and persistence of Dr. Thom Hodgson, Co-Director, and Linda Smith, OR Program Assistant, in the face of several bureaucratic challenges.

Finally, I wish to thank my sister, my brother, and the rest of my family for their unconditional love and support over the years. My parents took a strong interest in my education from an early age, and my grandfather took an active role in shaping my love for mathematics and mathematical modeling. Their influence on the successful completion of my PhD cannot be overstated. I hope the completion of this dissertation will honor the memory of my late grandmother, who wanted me to pursue a PhD before I did, and my late Aunt Myriam who passed away shortly before my dissertation defense.

I was fortunate to receive financial support from North Carolina State University and Intel Corporation over the years. This dissertation is also based in part upon work supported by the National Science Foundation under Grant Number CMMI 1029706. Any opinions, findings, and conclusions or recommendations expressed here do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Overview

Stochastic programming is a branch of mathematical programming that deals with problems of decision making under uncertainty. An emphasis on mathematical programming techniques, such as linear, integer, mixed integer, and nonlinear programming sets stochastic programming apart from other methods for decision making under uncertainty. A broad variety of associated models, properties, and solution algorithms can be found in Birge and Louveaux (1997); Kall and Wallace (1994) and Shapiro et al. (2009).

In this dissertation, we consider multi-stage stochastic programming methodologies, specifically those dealing with linear costs and constraints and decision variables that are continuous or discrete. Throughout this document we refer to these problems as multi-stage stochastic linear programs (MSLPs). MSLPs model the choices that must be made by a decision maker in a finite sequence of decision epochs, or stages. In many applications these epochs correspond to decisions that must be made at equal intervals of time, but this is not a necessity. We assume that the decisions correspond to the choice of integer and/or continuous valued variables prior to the realization of random parameters at each decision epoch. We further assume that the decision maker benefits from knowing the outcome of previous stages' decisions and random

parameter outcomes.

If the random parameter outcomes for a given stage are independent of past decisions, then the MSLP is considered exogenous; otherwise it is referred to as endogenous. In any solution methodology, we must ensure that decisions are made knowing the distribution of random parameters at future stages, but not the outcomes. For example, in a production environment, a decision maker may need to determine the number of items to produce in a week, an integer decision variable, and the production start time, a continuous decision variable, after reviewing the amount of available inventory and considering uncertain future demand.

Most MSLPs utilize a discrete definition of the underlying random parameters, referred to as a scenario tree. Generating this scenario tree is an important aspect of multi-stage stochastic programming. In most practical problems, the number of possible outcomes of all the random parameters is very large or, in the case of continuous random variables, uncountably infinite. Most efficient solution algorithms rely on consideration of a much smaller subset of outcomes, or scenarios. In multi-stage stochastic programming, random parameters are associated with each stage. Since the decisions and outcomes may depend on decisions and outcomes of previous stages, a scenario tree structure is a natural representation of all outcomes under consideration. Deciding which scenarios to include involves balancing the conflicting objectives of a small problem size with accuracy in the model representation.

One important method for generating scenario trees uses a deterministic measure of the error associated with the scenario tree as an approximation of all possible random outcomes. The support of the random variables is partitioned into disjoint sets. Each set corresponds to a distinct scenario, and the probability that the random variables belong to the set is the probability of the scenario. The outcomes for each scenario are taken selected from the values in the corresponding set. For example, we often use the mean values of the random variables conditioned on those variables belonging to the set. Such algorithms can be adapted to a sequential approximation scheme which we refer to as *sequential bounding*. The approximate problem is solved iteratively and modified by refining the partition of the support for the random

variables until a desired level of accuracy or some other termination criterion is reached. Such an approach was first suggested by Birge (1985a).

In the following sections, we define the mathematical notation for an MSLP, including the special case of a two-stage stochastic linear program (2SLP). We also give an example of a multi-stage stochastic linear programming problem for production planning. More general nonlinear stochastic programming formulations are covered elsewhere (Birge and Louveaux, 1997). In the next chapter, we review solution algorithms and associated implementation issues with an emphasis on those methodologies that are likely to be most successful in solving large-scale problems in a reasonable amount of computation time.

1.2 Mathematical Formulations

The following is a mathematical description of a two-stage stochastic linear programming problem.

$$\begin{aligned} \text{Minimize} \quad & z = cx + Q(x) \\ \text{Subject To} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{1.1}$$

where $Q(x) = E_\xi[Q(x, \xi(\omega))]$ and $Q(x, \xi(\omega)) = \min\{q(\omega)^T y | W(\omega)y = h(\omega) - T(\omega)x, y \geq 0\}$.

The formulation for the general T-stage MSLP is as follows:

$$
\begin{aligned}
\text{Min} \quad & c^1x^1 + E_{\xi^2}[\min_{x^2(\omega^2)} c^2(\omega^2)x^2(\omega^2) \\
& + \ldots + E_{\xi^T|\xi^2,\ldots,\xi^{T-1}}[\min_{x^T(\omega^{[T]})} c^T(\omega^{[T]})x^T(\omega^{[T]})]\ldots] \\
\text{s.t.} \quad & W^1x^1 = h^1 \\
& T^2(\omega^2)x^1 + W^2(\omega^2)x^2(\omega^2) = h^2(\omega^2), \\
& \vdots \\
& T^T(\omega^{[T]})x^{T-1} + W^T(\omega^{[T]})x^T(\omega^{[T]}) = h^T(\omega^{[T]}) \\
& x^1, x^t(\omega^{[t]}) \geq 0, t = 2, \ldots, T. \qquad (1.2)
\end{aligned}
$$

We use W^t to denote an $m_t \times n_t$ matrix, and x^t is the t^{th} stage vector of decision variables of length n_t. In stage t, c^t is a row vector, h^t a column vector, and T^t a matrix of conformal dimensions. We use ξ^t to denote the vector of random variables associated with stage t, the entries of $T^t(\omega^{[t]})$, $h^t(\omega^{[t]})$, and $W^t(\omega^{[t]})$ for $t = 2, \ldots, T$. For stage 2 and higher, these random matrices and the decision variables, $x^t(\omega^{[t]})$, are indexed on $\omega^2 \in \Omega^2$ and $\omega^t \in \Omega^t(\omega^{[t-1]})$, the set of outcomes at stage 2 and all other stages $t = 3, \ldots, T$ respectively. Here we use the notation $\omega^{[t]} = (\omega^2, \ldots, \omega^t)$ to denote the vector of random outcomes prior to and including stage t. Similarly, the support for ξ^t is $\Xi^t(\omega^{[t-1]})$ for $t = 3, \ldots, T$. Sometimes it is convenient to refer to the vector of random problem parameters associated with every stage, $\xi = (\xi^2, \ldots, \xi^T)$. The realization of any such random vector is called a scenario, and the realization of any vector $\xi^{[t]} = (\xi^2, \ldots, \xi^t)$, with $2 \leq t \leq T$, is called a subscenario. The support set for all scenarios is denoted $\Xi = \{\xi(\omega^{[T]}) | \omega^2 \in \Omega^2, \ldots, \omega^T \in \Omega^T(\omega^{[T-1]})\}$.

The set of possible outcomes, as well as the conditional distribution of those outcomes, at any stage may depend on the outcomes at previous stages. If they also depend on the decisions made at previous stages, then ξ^t is endogenous; however, most of the stochastic programming literature assumes ξ^t is exogenous (Gupta and Grossmann, 2010; Jonsbraten and Woodruff, 1998). Reflecting the most common case, the expectations at each stage are conditioned on the

vectors of random variables of all the previous stages. This dependence, and any dependence that decision variables have on previous decisions, is otherwise implicit in the formulation. Of course, there is no dependence on the vectors of random variables associated with future stage outcomes or decisions. This requirement is commonly referred to as *non-anticipativity* in the stochastic programming literature.

Non-anticipativity can be defined more formally as follows. If we let $\Omega = \{\omega^{[T]}\}$ be the set of all outcomes in the sample space, then $(\Omega, \mathcal{F}, \mathcal{P})$ defines the probability space. Here, $\mathcal{F}$ is the sigma field generated from ξ^T, or $\sigma(\xi^T)$. If we let $\mathcal{F}_t$ be the sigma field generated by ξ^t, then we see that $\{\mathcal{F}_t\}_{t=2}^T$ is a filtration, with $\mathcal{F}_s \subseteq \mathcal{F}_t$ when $2 \leq s \leq t \leq T$. So, outcomes of the random variable ξ^s are $\mathcal{F}_t$-measurable whenever $s \leq t$. In other words, elements in $\mathcal{F}_t$ consist of subsets of the elements in Ω with the same history up to and including time t. See Ross and Peköz (2007) for an accessible introduction to measure theory and probability.

Equation 1.2 can be expressed more compactly as follows.

$$\begin{aligned} \text{Min} \quad & c^1 x^1 + Q^1(x^1) \\ \text{s.t.} \quad & W^1 x^1 = h^1 \\ & x^1 \geq 0. \end{aligned}$$

where the recursive relationship between the first stage functions is

$$\begin{aligned} Q^1(x^1) &= E_{\xi^2}[Q^1(x^1, \xi^2)] \\ Q^1(x^1, \xi^2(\omega^2)) &= \min_{x^2}\{c^2(\omega^2)x^2 + Q^2(x^2) | T^2(\omega^2)x^1 + W^2(\omega^2)x^2 = h^2(\omega^2); x^2 \geq 0\}, \end{aligned}$$

and the t^{th} stage recourse function is defined recursively as:

$$\begin{aligned} Q^t(x^t) &= E_{\xi^{t+1}|\xi^2 \ldots \xi^t}[Q^t(x^t, \xi^{t+1})], t = 2, \ldots, T-1 \\ Q^t(x^t, \xi^{t+1}(\omega^{[t+1]})) &= \min_{x^{t+1}} \{c^{t+1}(\omega^{[t+1]})x^{t+1} + Q^{t+1}(x^{t+1}) | \\ & \quad T^{t+1}(\omega^{[t+1]})x^t(\omega^{[t]}) + W^{t+1}(\omega^{[t+1]})x^{t+1} = h^{t+1}(\omega^{[t+1]}); x^{t+1} \geq 0\}, \\ & \quad t = 2, \ldots, T-1 \end{aligned}$$

Finally, we define the boundary condition at stage T as:

$$Q^T(x^T) \equiv 0.$$

The multi-stage formulation can be viewed as a two-stage formulation with the recourse function replaced by a multi-stage stochastic program with one less stage than the original. In the above formulations, there are no integer restrictions on the decision variables, but these can be added easily, generalizing the problem to a multi-stage stochastic integer program.

When the support set for all scenarios, Ξ, is finite, all of the random problem parameters have finite support. We can simplify the formulation in Equation 1.2 for this case by numbering all possible subscenarios $\xi^{[t]} = (\xi^2, \ldots, \xi^t)$ for $t = 2, \ldots, T$. We use the superscript index k for the decision variables and random problem parameters in accordance with this numbering scheme. We use $a(k)$ to denote the index of the parent subscenario associated with subscenario k. For example, if subscenario k is $(\xi^1, \xi^2, \xi^3) = (\hat{\xi}^1, \hat{\xi}^2, \hat{\xi}^3)$, then $a(k)$ is the index for subscenario $(\hat{\xi}^1, \hat{\xi}^2)$. If subscenario k is associated with the second stage, then we will use the convention that $a(k) = 0$. We also introduce the notation p^k to be the probability that the event associated with subscenario k occurs. The formulation that follows is often called the deterministic equivalent linear programming problem associated with the MSLP.

$$
\begin{aligned}
\text{Min} \quad & c^1 x^{1,0} + \textstyle\sum_{t=2}^{T} \sum_{k=1}^{K_t} p^k c^{t,k} x^{t,k} \\
\text{s.t.} \quad & W^1 x^{1,0} = h^1 \\
& T^{t,k} x^{t-1,a(k)} + W^{t,k} x^{t,k} = h^{t,k}, \text{ for all } k = 1, \ldots, K_t \text{ and } t = 2, \ldots, T. \\
& x^{1,0}, x^{t,k} \geq 0, \text{ for all } k = 1, \ldots, K_t, \text{ and } t = 2, \ldots, T.
\end{aligned} \tag{1.3}
$$

1.3 Production Planning Example

As an example of an MSLP, we consider a single-product, fixed lead-time, finite capacity production planning problem. At each of a finite number of decision epochs $t = 1, \ldots, T$, the decision maker must determine how many jobs X_t to release into the production system. These jobs will immediately become work in process inventory (WIP), and for simplicity, we assume that each job will become part of finished goods inventory (FGI) at the next decision epoch, provided a fixed production capacity constraint is not violated.

The first stage decision variables include:

$X_1 \equiv$ The number of jobs to be released initially at the first decision epoch,

$I_1 \equiv$ The initial FGI,

$B_1 \equiv$ The initial unmet demand,

$W_1 \equiv$ The initial WIP inventory,

$P_1 \equiv$ Number of finished goods produced between the first and second decision epochs.

The stage t decision variables include:

$X_t \equiv$ The number of jobs released by the decision maker at epoch t,

$I_t \equiv$ FGI at epoch t,

$B_t \equiv$ Cumulative unmet demand at epoch t,

$W_t \equiv$ WIP at epoch t,

$P_t \equiv$ Number of finished goods produced between epochs t and $t+1$.

We assume the time between successive decision epochs is sufficiently small that the WIP and FGI is roughly constant in the duration, and production can be used to fulfill demand during the same period. In the formulation below, WIP is always zero in any optimal solution. So, the W_t decision variables are not necessary, but we include them to illustrate how they relate to other decision variables in more complicated formulations.

I_1, B_1, and W_1 are assumed to be known prior to the first decision epoch. Other known parameters include:

$$\begin{aligned} C_t &\equiv \text{The maximum number of FGI that can be produced between epochs } t \text{ and } t+1. \\ \gamma &\equiv \text{The per period discount factor for future costs.} \end{aligned}$$

Between two consecutive decision epochs,

$$\begin{aligned} c_w &\equiv \text{Fixed per period unit cost of carrying WIP,} \\ c_h &\equiv \text{Fixed per period unit cost of carrying FGI,} \\ c_u &\equiv \text{Fixed per period unit cost of cumulative unmet demand, and} \\ c_p &\equiv \text{Fixed production cost per unit.} \end{aligned}$$

In addition to these deterministic parameters, there is a stochastic parameter.

$$D_t(\omega^t) \equiv \text{The demand for finished goods between epochs } t-1 \text{ and } t.$$

We assume knowledge of an underlying joint distribution for $\xi^{[t]}(\omega^{[t]}) = (D_2(\omega^2), \ldots, D_t(\omega^t))$ for $t = 1, \ldots, T$. The formulation follows.

$$\begin{aligned} \text{Min} \quad & c_h I_1 + c_u B_1 + c_w W_1 + c_p P_1 + Q^1(X_1, I_1, B_1, W_1, P_1) \\ \text{s.t.} \quad & P_1 \leq C_1 \\ & P_1 \leq W_1 + X_1 \\ & X_1, P_1 \geq 0 \end{aligned}$$

where

$$Q^1(X_1, I_1, B_1, W_1, P_1) \quad = \quad E_{\xi^2}[Q^1(X_1, I_1, B_1, W_1, P_1, \xi^2)],$$

and

$$\begin{aligned} Q^1(X_1, I_1, B_1, W_1, P_1, \xi^2(\omega^2)) \quad = \quad & \\ \text{Min} \quad & \gamma c_h I_2 + \gamma c_u B_2 + \gamma c_w W_2 + \gamma c_p P_2 \\ & + Q^2(X_2, I_2, B_2, W_2, P_2) \\ \text{s.t.} \quad & W_2 = W_1 + X_1 - P_1 \\ & I_2 - B_2 = I_1 - B_1 + P_1 - D_2(\omega^2) \\ & P_2 \leq W_2 + X_2 \\ & P_2 \leq C_2 \\ & X_2, I_2, B_2, W_2, P_2 \geq 0. \end{aligned}$$

The recourse function at stage t can be written as:

$$\begin{aligned} Q^t(X_t, I_t, B_t, W_t, P_t) \quad = \quad & E_{\xi^{t+1}|\xi^2 \ldots \xi^t}[Q^t(X_t, I_t, B_t, W_t, P_t, \xi^{t+1})] \\ \text{for } t = 2, \ldots, T-1, & \end{aligned}$$

where

$$
\begin{aligned}
Q^t(X_t, I_t, B_t, W_t, P_t, \xi^{t+1}(\omega^{t+1})) \quad = \quad & \\
\text{Min} \quad & \gamma^t c_h I_{t+1} + \gamma^t c_u B_{t+1} + \gamma^t c_w W_{t+1} + \gamma^t c_p P_{t+1} + \\
& Q^{t+1}(X_{t+1}, I_{t+1}, B_{t+1}, W_{t+1}, P_{t+1}) \\
\text{s.t.} \quad & W_{t+1} = W_t + X_t - P_t \\
& I_{t+1} - B_{t+1} = I_t - B_t + P_t - D_{t+1}(\omega^{t+1}) \\
& P_{t+1} \leq W_{t+1} + X_{t+1} \\
& P_{t+1} \leq C_{t+1} \\
& X_{t+1}, I_{t+1}, B_{t+1}, W_{t+1}, P_{t+1} \geq 0.
\end{aligned}
$$

for $t = 2, \ldots, T-1$.

The recourse function for stage T is:

$$Q^T(X_T, I_T, B_T, W_T, P_T) \quad \equiv \quad 0.$$

MSLPs such as the above example are, in general, difficult to solve. When the random problem parameters are independent, the two-stage version of MSLP is known to be #P-hard. Random variables for any given stage in MSLP are usually assumed to have probability distributions independent of decisions in previous stages. If we relax this assumption, MSLP is known to be PSPACE-hard (Dyer and Stougie, 2006). The computational complexity of MSLPs with exogenous random variables is still an open question, but Dyer and Stougie (2006) conjecture that it is also PSPACE-hard. If a polynomial algorithm is ever found for solving #P-hard or PSPACE-hard problems in polynomial time, this would imply the existence of a polynomial time algorithm for solving NP-hard problems. There is no known polynomial time algorithm to convert PSPACE-hard problems to #P-hard problems, but if PSPACE-hard problems can be solved in polynomial time, then so can #P-hard problems (Arora and Barak,

2009). This suggests that MSLPs may be more difficult to solve than similarly sized two-stage problems. MSLPs that include integer decision variables are also difficult to solve, since the recourse functions are not guaranteed to be convex. Even if all the random problem parameters are discrete, a relatively modest number of independent random variables results in a very large number of joint outcomes. Assuming independent variables at each stage, the total number of possible outcomes is obtained by multiplying the number of outcomes at each stage, causing the resulting scenario tree to grow exponentially in the number of stages. If any of the random parameters are continuous, then there is effectively an infinite number of possible outcomes. In practice, some representative sample of the possible outcomes must be obtained to make the problem tractable.

1.4 Organization of this Document

In this dissertation we present new sequential bounding approaches for two-stage stochastic programs and apply these approaches to a single-period production planning problem with downward product substitutions. We also present a multi-stage sequential bounding approach, and compare this with several other scenario generation approaches for a production planning problem with uncertain demand and constant work in process inventory. Finally, we consider a multi-stage stochastic program to model the evolution of demand forecasting for production planning with load dependent lead times. We compare a sequential bounding approach to solving this model with other solution methodologies.

In Chapter 2 we review the stochastic programming literature. Methods for solving MSLPs directly are generally limited to situations where the random problem parameters are restricted to a finite and relatively small support set. So, we review methods for approximating MSLPs with continuous or large support sets. In particular, we describe what a scenario tree is and how it is used to approximate the random process associated with an MSLP. We review adaptive and non-adaptive algorithms based on scenario trees. Finally, we consider applications of stochastic programming to supply chain planning problems.

In Chapter 3 we study sequential bounding for two-stage stochastic programs. We extend the approach developed by Denton and Gupta (2003) to more general two-stage stochastic problems. The structure of these problems is studied and methods for improving convergence are explored. In particular, we develop methods for tightening bounds on dual and primal decision variables and optimally selecting representative outcomes for each scenario. The sequential bounding approaches are compared to an existing Monte Carlo sampling-based approach for a single-period production planning problem with downward product substitutions.

In Chapter 4 we compare several approaches for scenario tree generation for multi-stage stochastic programming. These include a simple random sampling based approach and two previously published approaches using moment matching and expected value of perfect information (EVPI) to generate the scenario tree. We also propose and evaluate a new sequential bounding approach using dual information. We use a simulation model to evaluate the quality of the solutions for a supply chain management problem in which product demand is continuously distributed at each stage. We use a series of test instances to compare the scenario generation methods on the basis of solution quality, computation time, and difficulty in implementation.

In Chapter 5 we present a multi-stage stochastic program to model the evolution of demand forecasting for production planning subject to load dependent lead times. We use the sequential bounding approach developed in Chapter 4 to solve the MSLP. We compare experimental results with alternative models and solution approaches.

Finally, Chapter 6 covers the main conclusions that can be drawn from Chapters 3, 4, and 5 and describes opportunities for future research.

Chapter 2

Literature Review

2.1 Overview

In this chapter, we describe existing methods for efficiently solving two-stage stochastic programs and MSLPs. Methods for solving such problems directly are generally limited to situations where the random problem parameters are restricted to a finite and relatively small support set. Hence, we also review methods for approximating stochastic linear programming problems with continuous or large finite support sets. In particular, we describe what a scenario tree is and how it is used to approximate the random process associated with an exogenous stochastic linear programming problem. We also review methods for finding bounds on the optimal objective function value of stochastic programming problems, since these bounds are closely related to scenario tree approximations. Finally, we consider applications of stochastic programming to supply chain planning problems.

2.2 MSLP Methodologies

2.2.1 Deterministic Equivalent Linear Programs

As shown in Equation 1.3, when the support set of the random problem parameters is finite, we can formulate any multi-stage stochastic programming problem using a deterministic equivalent linear programming problem. This formulation is achieved by indexing decision variables associated with the second stage and later stages on the set of subscenarios, or the set of all possible past outcomes. Thus, the number of variables and constraints is multiplied by the number of possible outcomes. This can often result in problems that are too large to solve using standard linear programming algorithms, such as the simplex algorithm (Dantzig, 1963) or interior point methods. However, improvements can be made to standard interior point methods for solving two-stage deterministic equivalent linear programming problems (Birge and Holmes, 1992).

2.2.2 L-shaped Method and Nested Decomposition

In the special case of two stages, continuous decision variables, and discrete random variables, an efficient Benders decomposition approach, the L-shaped method (Van Slyke and Wets, 1969) has been developed based on an outer linearization of the recourse function. A relaxation of the problem is utilized, known as the master problem. This relaxation does not include any of the constraints associated with the second stage decision variables. The master problem is repeatedly modified and solved, progressively refining the outer linear approximation to the recourse function. The master problem solution is passed to subproblems corresponding to the recourse problem for each scenario. The solutions to the scenario subproblems, in turn, provide more information about the recourse function, which is passed to the master problem in the form of valid inequalities, known as feasibility and optimality cuts. The integer L-shaped method adds these valid inequalities within a branch and cut procedure in order to deal with binary integer decision variables (Laporte and Louveaux, 1993).

Nested Decomposition (Glassey, 1973) extended the use of Benders decomposition to MSLPs.

Similar to the L-shaped method, a master problem consisting of only those constraints and costs associated with the first stage of the problem is defined. However, here the subproblems are MSLPs. We can therefore use Benders decomposition for each subproblem, generating new nested subproblems. The solution process involves passing feasible solutions for a subproblem at one stage to its successor subproblems at the next stage. Those successor subproblems are solved to provide valid inequalities (optimality and feasibility cuts) for the parent problem. Degenerate subproblems are common when using nested decomposition, and in order to reduce runtime associated with degenerate solutions, a partitioning strategy was developed (Birge, 1985a). Within the nested decomposition framework, there are several implementation decisions that must be made, including how much information is passed between the master problem and subproblems before moving to a different stage. The fast forward and fast backward approach was developed and found to be among the most successful of such approaches (Wittrock, 1983; Gassmann, 1990).

2.2.3 Progessive Hedging

Progressive hedging (Rockafellar and Wets, 1991) is a heuristic method for solving MSLPs. This method involves the decomposition of the MSLP into scenario problems. These problems are based on relaxing non-anticipativity requirements. This is equivalent to introducing many new decision variables into the MSLP so that every decision variable is indexed on every scenario. Such an indexing scheme is only appropriate for the last stage decision variables in an MSLP with non-anticipativity. Progressive hedging seeks to iteratively solve this relaxed MSLP and combine decision variables to re-impose non-anticipativity. The convergence of the algorithm to a feasible solution is through the use of penalty costs which are based on the degree to which the non-anticipativity constraints are violated.

2.2.4 Endogenous MSLPs

When the outcome of random problem parameters associated with any particular stage of a stochastic programming problem depend on past decisions, the problem is said to be an endogenous stochastic programming problem. The literature on endogenous stochastic programming problems is limited (Gupta and Grossmann, 2010; Jonsbraten and Woodruff, 1998). If all endogenous random parameters depend linearly on past decisions, we can reformulate the problem as an exogenous MSLP. For example, consider the endogenous random variable $\xi^3 = M_1 x^1 + M_2 x^2 + \epsilon$, where ϵ is an independent random variable, x^t is a vector of decision variables, and M_t is a fixed matrix associated with stage t. We can define $y^2 = M_1 x^1$ and $y^3 = y^2 + M_2 x^2 + \epsilon$ to be vectors of decision variables at stage 2 and 3 respectively, replacing ξ^3 with y^3 everywhere in the formulation. In order to obtain an exogenous formulation, we can assume random parameters are linearly dependent on past decisions.

Throughout this thesis, we assume that either the underlying stochastic process for MSLP is exogenous, or a suitable exogenous approximation to an underlying endogenous process is possible.

2.3 Scenario Generation Methods

It is often impossible or impractical to solve problems with large or infinite support sets Ξ directly using the methods described in the last section. In such situations, we need some way to approximate the MSLP by another MSLP with a support set that is finite and relatively small. The process of generating this representative finite support set for the random problem parameters of a larger MSLP is called scenario tree generation. In this section, we describe several methods for generating and evaluating the quality of scenario trees.

We begin by defining a scenario tree. For exogenous MSLPs, the realization of random problem parameters may depend on past random variable outcomes, but they are always independent of past decisions. This fact allows us to model the stochastic process $\{\xi^t\}_{t=2}^T$ using a

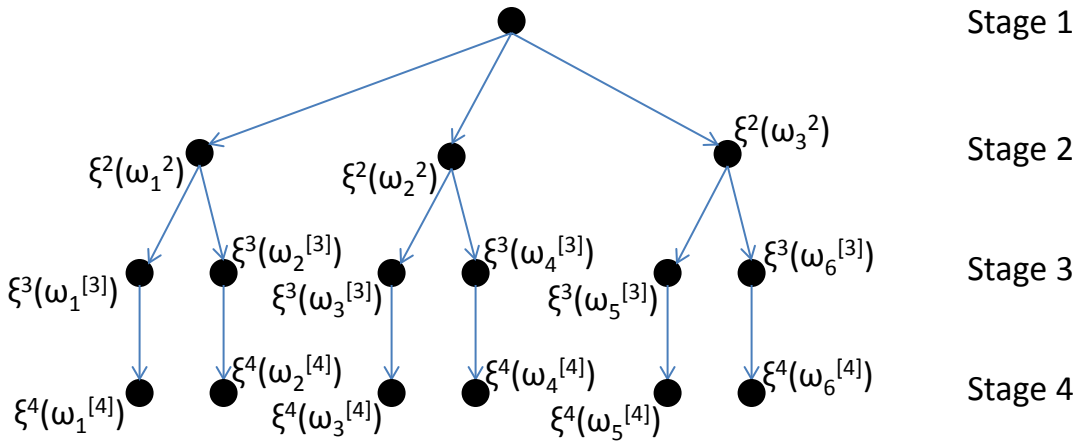

Figure 2.1: A scenario tree with each node representing the outcome of ξ^t at stage t.

tree data structure as follows. Every level of the tree corresponds to a stage in the MSLP, with the root node corresponding to the first stage and all leaf nodes corresponding to elements of Ξ^T, the support set for ξ^T. Every node, except for the root node, has exactly one predecessor node in the previous stage. We use arcs to show this parent-child relationship between nodes in the tree. From this definition, we see that any path from the root node to a node at level t corresponds to a subscenario $(\xi^2, \ldots, \xi^t)$. In particular, any path from the root node to a leaf node corresponds to a scenario.

We see that scenario trees are capable of capturing all the dependencies between random problem parameters. An intermediate node at level t between the root and leaf nodes in the tree represents a single outcome of ξ^t. Since $\xi^t = \xi^t(\omega^{[t]})$, may depend on the outcome associated with previous stages, this dependence is captured by the unique sequence of predecessor nodes at earlier stages of the tree.

We provide an example scenario tree in Figure 2.1. This scenario tree has six scenarios, including $(\xi^2(\omega_2^2), \xi^3(\omega_2^2, \omega_4^3), \xi^4(\omega_2^2, \omega_4^3, \omega_4^4))$ where $\omega_4^{[4]} = (\omega_2^2, \omega_4^3, \omega_4^4)$. This is a balanced tree since every node at the same level of the tree has the same number of child nodes below it. Although balanced trees are common, they are not required.

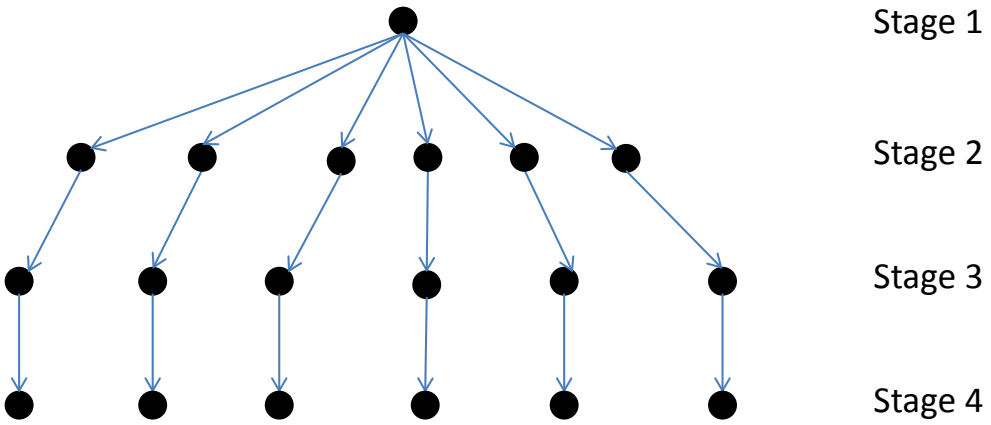

Figure 2.2: A fan-shaped scenario tree.

Next, we consider degeneracy in a scenario tree. Typically most nodes in a scenario tree, other than the root node, will have one or more sibling nodes connected to the same parent node of the previous stage. However, this is not always the case. When a node has only one child node, we will refer to it as degenerate, as in the nodes at stage 4 in Figure 2.1. Here, we see that the four stage stochastic programming problem may be viewed as a three stage problem, where the realization of the random vector ξ^4 is known with probability one when the realization of random vector ξ^3 is known.

A scenario $(\xi^2, \ldots, \xi^T)$ is considered degenerate if every node in that scenario, other than the root node, is degenerate. Similarly, we will say that a stage is degenerate if all the nodes at the corresponding level of the scenario tree are degenerate. An extreme case of degeneracy is the fan-shaped scenario tree of Figure 2.2. In this case, every scenario is considered degenerate because every node other than the root node is associated with only one scenario. Also, every stage is considered degenerate. Here, the four-stage stochastic programming problem can be viewed as a two-stage problem.

Degenerate nodes and nodes that have only a small number of child nodes are often used

in scenario trees. Such a node corresponding to a particular realization of $\xi^t(\omega^{[t]}) = \xi^t(\hat{\omega}^{[t]})$ is used even in situations where the number of possible realizations of $\xi^{t+1}((\hat{\omega}^2, \ldots, \hat{\omega}^t, \omega^{t+1}))$ is infinite, or large and finite. Since the number of variables and constraints of an MSLP is often proportional to the number of nodes in the underlying scenario tree, use of such nodes may be necessary to make the resulting MSLP tractable for solution by a computer. However, any model derived from a scenario tree with such nodes will necessarily fail to capture some of the inherent uncertainty associated with decision making.

In many cases, the relative importance of the stages diminishes over time. For example, future returns may be discounted according to an appropriate interest rate if we are trying to maximize total revenue. In these situations, with a limited computational budget, a scenario tree with fewer nodes in later stages may result in a more accurate approximation than one with the same number of nodes at every stage.

Next, we organize the literature on scenario generation into categories. First, we consider non-adaptive methods for generating scenario trees that produce a single scenario tree and often form the starting point for adaptive methods. We also review ideas related to evaluating the quality of scenario trees. These quality measures can also be used in adaptive algorithms, especially if they apply to subtrees of the scenario tree. Finally, we review adaptive methods that involve sequential modifications to a scenario tree in order to improve its accuracy in representing the original MSLP.

2.3.1 Non-adaptive Methods

Sampling Approach

Perhaps the most common approach for generating scenario trees for stochastic programs is sampling values of random parameters from known conditional distributions. This is also called the sample average approximation (SAA) method (Shapiro et al., 2009). A tree structure must be defined first. In other words, the number of nodes at every stage and how they are connected will be determined prior to using a random number generator. In this approach, we begin

by generating a random sample $\{\xi^2(\omega_i^2)\}_{i=1}^{K_1}$ of ξ^2. We will effectively replace the probability distribution associated with ξ^2 with the discrete distribution where each member of the sample set is equally likely. Each outcome of this sample set corresponds to a node at the second level of the scenario tree. Next, we generate samples of ξ^3 conditioned on the outcomes from the sample for stage 2. The size of these sample sets is simply the number of child nodes for each node corresponding to the outcome of ξ^2. We effectively replace the probability distribution of ξ^3 conditioned on the outcome of ξ^2 with a uniform discrete conditional distribution. The process of conditional sampling continues through each successive stage.

If random problem parameters are independent, then it is not necessary to generate random variables starting with the second stage. In that case, we use a random number generator for each independent random variable. When the distributions associated with random problem parameters depend on other parameters in the same or previous stage, then we must sample from a joint probability distribution function. One method for generating such random numbers is found in Cario and Nelson (1997).

We should point out that sampling described here is sometimes referred to as *external sampling* (Mak et al., 1999) because it is carried out prior to the solution procedure. This is in contrast to methods that use sampling during the course of the solution algorithm. See Pereira and Pinto (1991) for a description of the Stochastic Dual Dynamic Programming algorithm, which uses sampling to generate an outer approximation of the recourse function at every stage. See Higle and Sen (1991) for a description of the Stochastic Decomposition method for 2-stage stochastic programs, where sampling is used in a cutting plane algorithm. Also, see the related approach of Higle et al. (2010), Stochastic Scenario Decomposition, for multi-stage stochastic programming problems.

Let P be an MSLP with a large finite or continuous support set Ξ. We would like to determine the optimal solution x^* and optimal objective function z^* for P. Suppose that we have constructed a scenario tree using Monte Carlo sampling for P. After solving the approximating MSLP based on this scenario tree with smaller support set $\hat{\Xi}$, we obtain the optimal solution

$\hat{x}^*$ with optimal objective function value $\hat{z}^*$.

Although we usually cannot directly determine $|z^* - \hat{z}^*|$, we can repeatedly generate new and different scenario trees by resampling and resolving the approximating MSLPs. The average objective function value $\bar{\hat{z}} = \frac{1}{N_s}\sum_{i=1}^{N_s} \hat{z}_i^*$ for these approximating problems provides a biased estimate of the true optimal objective function value, since $E[\bar{\hat{z}}] \leq z^*$ (Shapiro et al., 2009). Chiralaksanakul and Morton (2004) use the $\hat{z}_i^*$ values to obtain a lower confidence bound on z^*. With finite second moment and relatively complete recourse assumptions, they also provide an upper confidence bound on z^* through repeated evaluation of feasible policies. When the random parameters between stages are independent, generating such feasible policies involves solving a two-stage stochastic programming problem at every stage. For most situations where random parameters depend on the random parameters of past stages, feasible policies are generated by solving a $(T-t+1)$-stage stochastic program for each stage $t = 2, \ldots, T$ in the original MSLP.

Variance reduction techniques, see Law (2006) for example, are generally useful for reducing the number of scenario trees that need to be generated for a single problem. For example, if the MSLP represents a problem of maximizing revenue subject to random demand at every stage, then antithetic variates might be used to generate a high demand scenario tree whenever a low demand scenario tree is generated. In this way, we hope to reduce the variance associated with $\bar{\hat{z}}$.

Importance sampling is among the most important variance reduction techniques for stochastic programming because it can be used to improve the accuracy associated with a single scenario tree. Notice that any instance of MSLP can be expressed as the maximization or minimization of a function of the form $\int f(x)p(x)dx$, where $p(x)$ is a joint probability distribution function for all components of the vector x. Importance sampling involves modifying the function f and replacing p with a different probability distribution function. In other words, we find an appropriate function q such that $\int f(x)p(x)dx = \int \frac{f(x)p(x)}{q(x)}q(x)dx$. Ideally, $\frac{f(x)p(x)}{q(x)}$ has a lower variance with respect to $q(x)$ than $f(x)$ with respect to $p(x)$. Thus, when evaluating the integral using random sampling, a smaller sample achieves the same accuracy in less time than if we

sampled $f(x)$ alone. See Dantzig and Glynn (1990) for a description of importance sampling in the context of MSLP.

Importance sampling is especially useful when relatively infrequent events have a high impact on the objective function value. We can design q to assign a higher probability to those events. Other heuristic approaches for finding q include finding the best q subject to certain structural assumptions, such as additive separability with respect to the components of x; so, we assume the form $q(x) = \sum_{i=1}^{n} q_i(x_i)$.

Distribution Approximation Methods

Pennanen and Koivu (2002) uses quadrature rules for sampling to approximate the distribution function. So, if we cast the objective function of MSLP as a multi-variable integration problem $\int f(x)p(x)dx$, with p the joint distribution function, and x the vector of all decision variables from all stages, then this approach uses quadrature rules for p. The article points out that the objective function itself, f, is not taken into account in this approach, citing the difficulty in doing so.

Pflug (2001) takes a similar approach to solving MSLP by approximating p with a discrete distribution $\tilde{p}$. Given that f satisfies a Lipschitz condition, we can find an upper bound on the difference between the optimal objective function value associated with p and the optimal objective function value associated with $\tilde{p}$. This upper bound can be minimized by selecting the best $\tilde{p}$ that maintains a pre-determined scenario tree structure.

Moment Matching Approach

One approach to scenario tree generation uses moments of various order, or other relevant statistics like correlation coefficients, as the defining property shared among outcomes at each stage of the tree. Methods for improving accuracy in the tree include increasing the number of branches, and subsequent scenarios, and including higher order moments. This approach leads to a nonlinear programming problem with the objective of minimizing some measure of distance

between the statistical properties of the scenario tree and the underlying random process. These nonlinear programs often must be solved heuristically to determine the probabilities and values associated with the random problem parameters at each node of the scenario tree (Høyland and Wallace, 2001). Miller and Rice (1983) explain how moment matching can be implemented with Gaussian quadrature points. In a pension plan optimization problem, Kouwenberg (2001) found that the performance of a moment matching approach was substantially better than simple external sampling.

In many situations a probability distribution for modeling the random problem parameters is not available. This may be the result of insufficient data or if we fail to find an adequate probability distribution to fit existing data. The moment matching approach is particularly attractive in these cases, since we can limit ourselves to estimating moments and correlation coefficients rather than full joint probability distribution functions.

Optimal Design

Often scenarios that take specific requirements into account, such as the absence of arbitrage opportunities in the context of financial applications, must be enforced in the scenario tree generation process. Some properties of the data in the scenario tree can be enforced, such as adherence to pre-specified moments. In the choice of random problem parameters associated with the scenario tree nodes, as well as the probabilities of their occurrence, we may have considerable freedom in selecting these values.

A nonlinear programming approach, or an associated goal programming approach can be taken to find a sample for $\{\xi_t\}_{t=1}^{T}$ and the associated probabilities $\{P(\omega^{[t]})|\omega^{[t]} \in \Omega(\omega^{[t-1]})\}$. Høyland and Wallace (2001) builds on previous methods where the mean and variance of the outcomes were preserved. Dependencies between time periods are taken into account. The authors provide a general nonlinear programming formulation. The formulation includes a minimization of weighted squared deviations from the specified statistics. These statistics can be moments of various order, correlations, a single worst case outcome, or any other desired quantifiable

property. The choice of these statistics is problem specific (Kaut and Wallace, 2007).

Unbiasedness and Stability

In Kaut and Wallace (2007), the importance of unbiasedness and stability are emphasized when evaluating scenario trees. Biasedness is defined as the difference between the optimal objective function values of a sampled problem and the actual problem, or a sufficiently large representation of the actual problem. Stability refers to the difference in objective function values for two optimal solutions associated with two different scenario trees, either with respect to the original problem or with the corresponding sampled problems. Although these criteria can be used in evaluating deteministic methods for scenario tree generation, there must be some mechanism for generating different trees and making comparisons.

2.3.2 Adaptive Methods

Adaptive methods start with a non-adaptive method for scenario tree generation. The scenario tree is assessed according to some measure of quality, which often involves solving an MSLP instance based on the current scenario tree. The scenario tree is then modified in some way in the hope of improving the measure of quality. This process may continue until some termination criterion is reached.

Scenario and Stage Aggregation and Disaggregation

The influence of additional scenarios, including extreme events that may not be well represented otherwise, can be measured by the contamination method (Dupacova et al., 2001). In this method, a class of probability distributions (P_λ) formed from the convex combination of the probability distributions associated with the scenario tree before and after adding additional branches is utilized. Lower and upper bounds on the stochastic program associated with P_λ can be determined. Cluster analysis and combining scenarios that are similar, according to a similarity measure that gives greater weight to earlier stage outcomes, is also possible. A non-

homogeneous Markov process can be used to generate conditional outcomes of scenarios, when appropriate.

2.4 Constraint and Variable Aggregation

Stochastic programming problems with continuously distributed random problem parameters can be expressed as infinite dimensional linear programming problems. We can see this in Equation 1.2 from the fact that the constraints are indexed on $\omega^t \in \Omega^t$ for $t = 2, \ldots, T$. Also the decision variables at any stage, except for the first, depend on the outcome of random variables associated with both that stage and past stages. From this perspective, we see that $x^t(\omega^{[t]}) = x^t((\omega^2, \ldots, , \omega^t))$ in Equation 1.2 is a function of all past outcomes, and solving the MSLP requires us to find the optimal function for every $t = 2, \ldots, T$. Even if the support for the random problem parameters is finite, a relatively small number of possible outcomes associated with each stage can result in very large scale deterministic equivalent problems for multi-stage stochastic programs (see Equation 4.2).

In Zipkin (1980b,a), upper and lower bounds for the optimal objective function value of large-scale linear programming problems are found. These bounds are obtained by solving a smaller, but related, linear programming problem based on an aggregation of the constraints and variables of the original problem. Zipkin's results were established for pure linear programming problems, with a finite number of constraints and variables, and were made without any reference to stochastic programming. These bounds depend on establishing bounds on the optimal primal and dual decision variables for the original problem.

In Birge (1985a), the results of Zipkin are extended to MSLPs with continuously distributed random variables appearing in the right hand side of the constraints. In this paper, Birge does indicate that more general problems with other uncertainties are possible. In this approach, constraints are aggregated according to a weighting function. This weighting function corresponds to the conditional joint probability distribution associated with the right hand side values. Similarly, a weighting function is defined for aggregating variables. These weighting functions also

correspond to the joint conditional probability distribution functions for the outcomes on which the decision variables depend. Wright (1994) develops the results within a measure-theoretic probability framework, allowing for uncertainties in the constraint and cost coefficients.

In the case where random variables are independent and the objective function is a convex function of the random variables, bounds for stochastic programming problems can be found using the classic Edmundson-Madansky and Jensen's inequalities. The bounds can be made arbitrarily tight by sequentially partitioning the support of the random variables into a successively finer partition and applying the bounds to each new cell of this partition (Huang et al., 1977).

In Denton and Gupta (2003), the L-shaped method with sequential bounding is developed for a two-stage stochastic programming formulation of an appointment scheduling problem. The algorithm iteratively refines a partition of the support set of continuous random parameters associated with the recourse problem. The decision variables found at each iteration converge to an optimal solution, along with the upper and lower bound for the optimal objective function value. The bounds at each iteration are based on those developed by Birge (1985a). From the computational experience with this problem, the authors found that the upper bound converged more slowly to the optimal objective function value than the lower bound.

2.5 Applications

The literature on applications of stochastic programming is vast. In this section we focus on the literature for production planning. There are many other application papers for various industries, such as asset liability management (Cariño et al., 1994), portfolio optimization (Dantzig and Infanger, 1993), and energy planning (Pereira and Pinto, 1991) for example. See Di Domenica et al. (2009) for a long list of publications on the applications of stochastic programming to various industries. See the tutorial of Sen and Higle (1999) for an introduction to stochastic programming and its applications. We also refer to Dupacova (2002) for applications of multi-stage stochastic programs.

The literature for production planning under uncertainty is also vast. We refer to Mula et al. (2006) for a review of the literature and a classification scheme for the various models that have been developed over the years. They identify 6 research areas where analytical models, like stochastic programming, can be applied: hierarchical production planning, material requirement planning, capacity planning, manufacturing resource planning, inventory management, and supply chain planning. Of these research areas, perhaps stochastic programming has been applied most often to capacity planning (see Eppen et al., 1989; Morton and Wood, 1999; Huang and Ahmed, 2009). In this dissertation, we focus on manufacturing resource planning and inventory management. In particular, we assume capacity is known, and we determine production and inventory activities prior to the realization of customer demand. We do not consider distributing manufactured goods to multiple locations.

We provide some examples of papers where a real-life problem was used in the development of a stochastic programming problem for production planning. We cover papers where such a model was either developed for, or inspired by, a *particular* application.

Guan and Philpott (2011) use a multi-stage stochastic quadratic programming problem to model a simplified version of a supply chain planning problem for Fonterra, a New Zealand Dairy company. The authors use the Dynamic Outer Approximation Sampling Algorithm, that they developed, for solving this problem. The authors use simulation to evaluate the optimal policy found using their approach, and they compare this policy with one found by solving a related deterministic model. Although higher expected earnings were achieved with the authors' model, they feel that a more detailed model would provide a better estimate of the savings.

Zanjani et al. (2010) develop a multi-stage stochastic programming model inspired by sawmill production planning. The model incorporates uncertainties in yield as well as demand. The authors model the production of multiple products. The multi-stage model is compared with both a deterministic model and a two-stage stochastic programming model. The quality of the multi-stage model solutions were found to be significantly higher than those for the other two.

In Chapter 4, we consider a multi-stage stochastic programming problem developed in the semiconductor industry by Higle and Kempf (2011). The authors consider uncertainty in yields for a multi-stage production process subject to uncertain demand. A constant work in process inventory constraint is used to keep production smooth and inventory holding costs low.

2.6 Contributions of This Dissertation

This dissertation contributes to the literature for adaptive scenario generation approaches for solving two-stage and multi-stage stochastic programs. In particular, we use sequential bounding, where progress towards improving the scenario tree is based on deterministic bounds of the optimal objective function value. These bounds are obtained from variable and constraint aggregation. For two-stage stochastic programs, we also consider deterministic bounds based on restricted recourse (Morton and Wood, 1999). Most other adaptive approaches use statistical measures of quality or depend on a surrogate measure of improvement towards a lower average cost. We show how sequential bounding can be applied to a wide variety of two-stage and multi-stage stochastic programs, including those with continuously distributed random variables appearing in the constraint coefficients, right hand side values, and cost coefficients. We evaluate new ways to improve sequential bounding for two-stage stochastic programs based on improving decision variable bounds and optimally selecting outcomes to minimize an upper bound on the optimal objective value. We also develop a new multi-stage sequential bounding approach. We know of only one other sequential bounding approach for multi-stage problems in the literature (Casey and Sen, 2005). The approach developed in Chapter 4 is applicable to a wider class of stochastic programs and avoids lengthy calculations for refining the scenario tree. Extensive experimental results for multi-stage sequential bounding have never before been published. We compare our approach to several existing approaches for two different production planning problems. We also provide new insights for future improvements to sequential bounding based on our computational experiments.

Chapter 3

Sequential Bounding for Two Stage Stochastic Linear Programs

3.1 Introduction

For stochastic programs with a very large or an infinite number of scenarios, approximations must be employed. The most common approximation is Monte Carlo sampling, which was first used in the context of sampling-based decomposition by Dantzig and Glynn (1990). Sampling-based methods have been used to estimate statistical confidence intervals on the optimality gap of stochastic programs by Mak et al. (1999) and Glynn and Infanger (2011). The quality of the statistical bounds generated by sampling-based methods depend on the number of scenarios sampled, but larger numbers of scenarios generally result in longer computation times. The number of samples to best trade off computation time and accuracy is problem-specific.

It is also possible to compute a deterministic bound on the error in an approximation based on deterministic selection of scenarios. For example, the support of the random variables can be partitioned into disjoint sets. Each set corresponds to a distinct scenario, and the probability that the random variables belong to the set is the probability of the scenario. The outcomes for each scenario are the mean values of the random variables conditioned on those variables

belonging to the corresponding set. Such algorithms can be adapted to a sequential approximation scheme, where the approximate problem is iteratively solved and modified by refining the partition of the support for the random variables until a desired level of accuracy or some other termination criterion is reached. Such an approach was first suggested by Birge (1985a).

In this chapter, we describe methods to approximate the solution to two-stage stochastic programs. We present ways to improve the bounds on the approximation error developed by Birge (1985a) in the context of a deterministic sequential approximation algorithm, which we call *sequential bounding*. In particular, we consider the problem of finding tight bounds on optimal values of the second stage primal and dual decision variables. Although decision variable bounds are often readily apparent, we show how bounds on the optimal values can be improved for subsets of the random variable support, resulting in faster convergence of sequential bounding. We also consider improvements to the restricted recourse bounds of Morton and Wood (1999), which depend on such decision variable bounds. Finally, we consider a mathematical programming approach to find the smallest approximation error by selecting representative outcomes for subsets of the random variable support.

We present three algorithms for bounding the optimal values of primal and dual decision variables. The first algorithm uses linear programming complementary slackness conditions and improves bounds on either the set of primal or dual decision variables by utilizing bounds on the other set of variables. In the absence of primal degeneracy, we show that the tightest upper and lower bound on each primal and dual decision variable is the solution of a corresponding nonlinear program. A linear programming relaxation is utilized in the second algorithm for each of these problems. The third algorithm generates a simplicial cone of dominated dual solutions. This is used to generate a disjunctive linear program for each optimal dual decision variable bound. Finally, we identify special cases where bounds on the optimal decision variables can be found using simple recursion.

The algorithms developed in this chapter are utilized in comparing the performance of the sequential bounding algorithm with the Sample Average Approximation (SAA), a Monte Carlo

sampling-based approach. Computational experiments are based on a collection of test instances for an important stochastic program that arises in the context of semiconductor manufacturing. We compare the deterministic bounds obtained, using sequential bounding with each of the algorithms, to statistical confidence intervals on the optimality gap obtained by Monte Carlo sampling.

The remainder of this chapter is organized as follows. First, we review the relevant literature on bounding methods in Section 3.2. In Section 3.3 the general two-stage stochastic linear programming problem (2SLP) is given, and in Section 3.4 our sequential bounding algorithm is developed for this problem. In Section 3.5 we describe four approaches for finding bounds on the optimal second stage primal and dual decision variables. In Section 3.6 we describe an inventory planning problem with uncertain demand and downward substitutions. We present computational experiments comparing Sample Average Approximation and variations of the sequential bounding algorithm. Conclusions are given in Section 3.7.

3.2 Literature Review

Early work on aggregation bounds was done in the context of deterministic linear programs. In Zipkin (1980b,a), upper and lower bounds for the optimal objective function value of large-scale linear programs were developed. These bounds are obtained by solving a smaller, but related, linear program based on an aggregation of the constraints and variables of the original problem. Aggregate variables and constraints are generated through a linear combination of variables and constraints respectively. The multipliers used in the linear combinations can be viewed as a distribution over rows or columns of the problem parameters in the original linear program. The bounds also depend on establishing bounds on the optimal primal and dual decision variables of the original problem.

In Birge (1985a), the results of Zipkin were extended to stochastic programs with continuously distributed random variables appearing in the right hand side of the constraints. In this approach, constraints are aggregated according to a weighting function that corresponds to the

conditional joint probability distribution associated with the right hand side values. Similarly, a weighting function is defined for aggregating variables. These weighting functions also correspond to the joint conditional probability distribution functions for the outcomes on which the decision variables depend. Wright (1994) extended these bounds within a measure-theoretic probability framework, allowing for uncertainties in the constraint and cost coefficients.

In cases where the objective function is the expected value of a convex function, f, of independent random variables, upper and lower bounds for stochastic programs can be found. The classic Edmundson-Madansky inequality provides an upper bound. This is the expected value of a linear function that overestimates f. The classic Jensen's inequality provides a lower bound by evaluating f at the expected value of the random variables. Huang et al. (1977) show these bounds can be made arbitrarily tight by sequentially partitioning the support of the random variables into a successively finer partition and applying the bounds to each new cell of this partition. Many generalizations of these bounds and a variety of other bounding methods have been developed. Morton and Wood (1999) provide an excellent review of these methods.

Decision variable bounds based on problem structure have been previously discussed in the literature. Decision variable bounds for a production planning problem were discussed in Birge (1983). He used existing constraints to develop these bounds, such as non-negativity for primal decision variables and elastic penalty costs for dual variables. Other decision variable bounds are implied by existing constraints. Decision variable bounds were also developed by Morton and Wood (1999) for a network and semiconductor manufacturing problem. They reasoned that a unit increase in arc capacity results in no more than one unit of flow; so, the dual variable corresponding to each flow balance constraint is less than or equal to one. Birge (1985a) developed methods for decision variable bounds based on the sign of constraint coefficients and functions of these coefficients, but a limited number of problems possess the required structural properties. Denton and Gupta (2003) used a related bounding approach, and is the only previous publication we are aware of that improves the bounds on decision variables for different subsets of the random variable support. However, the approach is limited to a specific appointment

scheduling problem.

This chapter provides several novel contributions to the existing literature. First, we present several new methods to obtain bounds on optimal values for primal and dual decision variables that can be used to solve two-stage stochastic linear programs in a sequential bounding approach. We identify a class of special problem structures where optimal decision variable bounds can be found using simple recursion. We also develop a new mathematical programming approach to bound the approximation error by replacing random variables with decision variables. Finally, in our computational experiments we directly compare a sampling-based algorithm to sequential bounding, and in some cases, demonstrate that sequential bounding produces the lowest cost solution.

3.3 Two-stage Stochastic Linear Program

The following is a standard formulation for a two-stage stochastic linear program (2SLP):

$$\begin{aligned} \text{Min} \quad & z(x) = cx + Q(x) \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{3.1}$$

where $Q(x) = E_{\xi}[Q(x, \xi(\omega))]$ and $Q(x, \xi(\omega)) = \min\{q(\omega)y | W(\omega)y = h(\omega) - T(\omega)x, y \geq 0\}$. A is an $m \times n$ matrix of deterministic values, and $x \in \mathbb{R}^n$ is the vector of first stage decision variables. In the first stage constraints, b is a column vector and c a row vector of conformal dimensions. Here, $\xi(\omega) = Vec(h(\omega), T(\omega)), W(\omega), q(\omega))$, where $Vec(\cdot)$ is a vector whose entries match those of its arguments in columnwise order. In addition, $\omega \in \Omega$ indexes the possible outcomes of ξ. The components of ξ are assumed to be $\mathcal{F}$-measurable with respect to the probability space $(\Omega, \mathcal{F}, \mu)$. W and T are the *recourse* and *technology* matrices, respectively. We assume $Q(x, \xi(\omega))$ is feasible for any feasible first stage solution x and $\omega \in \Omega$. We use the

notation $Q(\cdot)$ to denote the *recourse function* (see Birge and Louveaux, 1997), which is the expectation over second stage recourse problems. The vector of second stage decision variables of length n_2 is denoted $y = y(x, \xi)$ to emphasize its dependence on x and ξ. We use $y^*(x, \xi)$ to denote an optimal solution of $Q(x, \xi)$. The matrix $W(\omega)$ is $m_2 \times n_2$, and the dimensions of the matrix $T(\omega)$, the row vector $q(\omega)$, and the column vector $h(\omega)$ associated with the second stage have conformal dimensions. Subscripts are used to denote entries of vectors and matrices throughout this chapter. For example, will refer to the i^{th} entry of $h(\omega)$ as $h_i(\omega)$ and the element in the i^{th} row and j^{th} column of $T(\omega)$ as $T_{i,j}(\omega)$.

The dual linear programming problem associated with the second stage recourse problem $Q(x, \xi(\omega))$ can be written as follows:

$$\begin{aligned} \text{Max} \quad & \pi(h(\omega) - T(\omega)x) \\ \text{s.t.} \quad & \pi W(\omega) \leq q(\omega) \\ & \pi \text{ unrestricted in sign} \end{aligned} \tag{3.2}$$

We use $\pi = \pi(x, \xi)$ for the dual decision variables to emphasize the dependence on x and the random variables ξ, and $\pi^*(x, \xi)$ is a corresponding optimal solution.

In order to define the dual stochastic program of 2SLP given in Equation 3.1, additional assumptions must be made (see Wright, 1994). First, we restrict $y(x, \xi(\cdot))$ to be in the Lebesgue space, $L^1(\Omega, \mathcal{F}, \mu; \mathbb{R}^{n_2})$, for any $x \geq 0$ such that $Ax = b$. Furthermore, we assume that there exists a feasible pair $(x, y(x, \xi(\omega)))$ for 2SLP so that $q(\omega)y(x, \xi(\omega))$ is integrable. The dual of 2SLP is as follows:

$$\text{Max} \qquad vb + E_{\xi}[w(\omega)h(\omega)] \tag{3.3}$$

$$\text{s.t.} \quad vA + E_{\xi}[w(\omega)T(\omega)] \leq c \tag{3.4}$$

$$w(\omega)W(\omega) \leq q(\omega), \text{ a.s.} \tag{3.5}$$

where v is a row vector of length m representing the first stage decision variables, and $w(\omega)$ is a row vector of length m_2 representing the second stage decision variables. Here, the constraints in Equations 3.4 and 3.5 are the first and second stage constraints respectively.

Applying the dual form of the results of Rockafellar and Wets (1976a) to Equation 3.3, an optimal solution, $v = v^*$ and $w = w^*$, exists if the following conditions are met: (1) the components of $h(\omega)$ are summable, but possibly unbounded, (2) all other components of ξ are bounded, (3) the set of feasible solutions (v, w) is bounded, and (4) w is measurable. For many applications, these conditions are not restrictive, since we can truncate distribution functions for random variables with infinite support with some possible loss of accuracy. Also, most realistic problems do not result in arbitrarily large optimal decision variable values. We will assume that these conditions hold, but alternative conditions to ensure the existence of v^* and w^* in Equation 3.3 are provided by Rockafellar and Wets (1976b). Those conditions rely on the existence of feasible solutions satisfying all the constraints with strict inequality. Conditions that do not rely on strict feasibility or a bounded feasible region are provided in Korf (2004).

In order to evaluate $Q(x)$ or solve an instance of 2SLP directly, generally the support set, Ξ, for the random variables, ξ, must be a finite set with reasonably small cardinality. In situations where this is not the case, an approximating problem can be solved where Ξ is replaced with a finite set, $\hat{\Xi}$. The following section describes a means for generating such a finite set so that a bound on the approximation error can be obtained.

3.4 Sequential Bounding

Sequential bounding involves partitioning the support set Ξ into disjoint sets S^k such that $\Xi = \cup_{k=1}^{\nu} S^k$. We also define $p^k = P\{\xi \in S^k\}$ and let $\hat{\xi}^k$ be an arbitrary vector in S^k for $k = 1, \ldots, \nu$. We define the finite support set $\hat{\Xi} = \{\hat{\xi}^k\}_{k=1}^{\nu}$, and $P\{\xi = \hat{\xi}^k\} = p^k$ for $k = 1, \ldots, \nu$. The 2SLP associated with this partitioning of Ξ, which we will refer to as the partitioned value problem (PVP) of 2SLP, can be formulated as follows:

$$\begin{aligned} \text{Min} \quad & \hat{z}^{\nu}(x, \{\hat{\xi}^k\}_{k=1}^{\nu}) = cx + \textstyle\sum_{k=1}^{\nu} p^k Q(x, \hat{\xi}^k) \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{3.6}$$

It will be convenient to define $\xi^k = E[\xi | \xi \in S^k] = Vec(h^k, T^k, W^k, q^k)$. If we set $\hat{\xi}^k = \xi^k$, we will refer to the resulting PVP as the partitioned mean value problem (PMVP) of 2SLP. The formulation follows:

$$\begin{aligned} \text{Min} \quad & z^{\nu}(x) = cx + Q^{\nu}(x) \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{3.7}$$

where $Q^{\nu}(x) = E_{\xi \in \hat{\Xi}}[Q(x, \xi)] = \sum_{k=1}^{\nu} p^k Q(x, \xi^k)$. We denote the optimal objective function value as $z^{\nu *}$, and the optimal solution as $(x^{\nu *}, \{y^{k,\nu *}\}_{k=1}^{\nu})$. Notice that the PMVP of 2SLP is a standard linear program, and the PMVP of Equation 3.3 is the dual linear program of Equation 3.7. We denote this dual optimal solution as $(v^{\nu *}, \{w^{k,\nu *}\}_{k=1}^{\nu})$.

We define each S^k as a hyper-rectangle aligned with the coordinate axes. In other words, for each $k = 1, \ldots, \nu$, $S^k = \Xi \cap \left([a_1^{(k)}, b_1^{(k)}] \times \ldots \times [a_J^{(k)}, b_J^{(k)}] \right)$, where $J = m_2(1 + n + 2n_2) + n_2$

is the number of components in ξ. So, for all $\omega \in \Omega$ such that $\xi(\omega) \in S^k$, $a_i^{(k)} \leq h_i(\omega) \leq b_i^{(k)}$, for $i = 1, \ldots, m_2$ and $a_{m_2(j)+i}^{(k)} \leq T_{ij}(\omega) \leq b_{m_2(j)+i}^{(k)}$, for $i = 1, \ldots, m_2$ and $j = 1, \ldots, n$. Subject to the assumptions of Section 3.3, we note that some of the a and b values can be $\pm\infty$; so, Ξ need not be bounded.

Since $x^{\nu *} \geq 0$ in the PMVP of 2SLP, we can define bounds $L_i^{(k)}$ and $U_i^{(k)}$, for $h_i(\omega) - \sum_{j=1}^{n} T_{i,j}(\omega) x_j^{\nu *}$ when $\xi(\omega) \in S^k$, as follows:

$$\begin{aligned} L_i^{(k)} &= a_i^{(k)} - \sum_{j=1}^{n} b_{m_2(j)+i}^{(k)} x_j^{(\nu)} \\ &\leq h_i(\omega) - \sum_{j=1}^{n} T_{ij}(\omega) x_j^{\nu *} \\ \leq b_i^{(k)} - \sum_{j=1}^{n} a_{m_2(j)+i}^{(k)} x_j^{(\nu)} &= U_i^{(k)}. \end{aligned} \tag{3.8}$$

In Section 3.5 we discuss how these bounds can be used to compute lower and upper bounds on the optimal value of the second stage recourse decisions.

The following result establishes bounds for the recourse function, $Q(x)$, using the PVP. For the special case where the PVP is the PMVP, a more compact version of this proposition and proof is found in Wright (1994), but our proposition and proof allows for equality constraints in the primal problem and non-zero $y^{k,LB}$ values, which lead to tighter bounds on $Q(x)$. Wright (1994) does not allow for arbitrary outcomes associated with each cell of the partition. We use the freedom to select these outcomes to develop tighter bounds on the recourse function in Section 3.4.2. The vectors $\hat{\xi}^{k,1}, \hat{\xi}^{k,2} \in S^k$ are arbitrary vectors with some of their components fixed to those for ξ^k:

$$\hat{\xi}^{k,1} = Vec(h^k, T^k, \hat{W}^{k,1}, \hat{q}^{k,1})$$

and

$$\hat{\xi}^{k,2} = Vec(\hat{h}^{k,2}, \hat{T}^{k,2}, \hat{W}^{k,2}, q^k).$$

We define $y^{k,UB}$,$y^{k,LB}$,$\pi^{k,UB}$, and $\pi^{k,LB}$ to be bounds on the second stage recourse decisions:

$$0 \leq y^{k,LB} \leq y^*(x,\xi) \leq y^{k,UB}$$
$$0 \leq \pi^{k,LB} \leq \pi^*(x,\xi) \leq \pi^{k,UB}$$

for all $\xi \in S^k$ and $k = 1,\ldots,\nu$. Here we use the notation $[\cdot]^+ = \max\{\cdot, 0\}$. Also $M_{i,\cdot}$ denotes the i^{th} row and $M_{\cdot,j}$ denotes the j^{th} column of any arbitrary matrix M.

Proposition 1. *Suppose x is feasible for the first stage constraints of 2SLP, $Ax = b$ and $x \geq 0$. Also assume that $y^*(x,\xi)$ and $\pi^*(x,\xi)$ are bounded for all $\xi \in \Xi$. Then*

$$\sum_{k=1}^{\nu} p^k Q(x,\hat{\xi}^{k,1}) - \epsilon_1^\nu(x,\{\hat{\xi}^{k,1}\}_{k=1}^\nu) \leq Q(x) \leq \sum_{k=1}^{\nu} p^k Q(x,\hat{\xi}^{k,2}) + \epsilon_2^\nu(x,\{\hat{\xi}^{k,2}\}_{k=1}^\nu),$$

where

$$\begin{aligned}
\epsilon_1^\nu(x,\{\hat{\xi}^{k,1}\}_{k=1}^\nu) = \; & \textstyle\sum_{k=1}^{\nu}\sum_{j=1}^{n_2}\Big[y_j^{k,UB}\int_{S^k}[\pi^*(x,\hat{\xi}^{k,1})(W_{\cdot,j}-\hat{W}_{\cdot,j}^{k,1}) - (q_j-\hat{q}_j^{k,1})]^+ d\mu \\
& -y_j^{k,LB}\textstyle\int_{S^k}[-\pi^*(x,\hat{\xi}^{k,1})(W_{\cdot,j}-\hat{W}_{\cdot,j}^{k,1}) + (q_j-\hat{q}_j^{k,1})]^+ d\mu\Big] \\
\epsilon_2^\nu(x,\{\hat{\xi}^{k,2}\}_{k=1}^\nu) = \; & \textstyle\sum_{k=1}^{\nu}\sum_{i=1}^{m_2}\pi_i^{k,UB}\int_{S^k}\Big[(h_i-\hat{h}_i^{k,2}) - (T_{i,\cdot}-\hat{T}_{i,\cdot}^{k,2})x \\
& -(W_{i,\cdot}-\hat{W}_{i,\cdot}^{k,2})y^*(x,\hat{\xi}^{k,2})\Big]^+ d\mu \\
& -\textstyle\sum_{k=1}^{\nu}\sum_{i=1}^{m_2}\pi_i^{k,LB}\int_{S^k}\Big[-(h_i-\hat{h}_i^{k,2}) + (T_{i,\cdot}-\hat{T}_{i,\cdot}^{k,2})x \\
& +(W_{i,\cdot}-\hat{W}_{i,\cdot}^{k,2})y^*(x,\hat{\xi}^{k,2})\Big]^+ d\mu
\end{aligned}$$

Proof.

$$
\begin{aligned}
Q(x) = & \int_{\Xi} q y^*(x,\xi) d\mu \\
= & \int_{\Xi} q y^*(x,\xi) d\mu - \sum_{k=1}^{\nu} \int_{S^k} \pi^*(x,\hat{\xi}^{k,1})[W y^*(x,\xi) - h + Tx] d\mu \\
\geq & \int_{\Xi} q y^*(x,\xi) d\mu - \sum_{k=1}^{\nu} \int_{S^k} \pi^*(x,\hat{\xi}^{k,1})[W y^*(x,\xi) - h + Tx] d\mu \\
& + \sum_{k=1}^{\nu} \int_{S_k} [\pi^*(x,\hat{\xi}^{k,1}) \hat{W}^{k,1} - \hat{q}^{k,1}] y^*(x,\xi) d\mu \\
= & \sum_{k=1}^{\nu} p^k Q(x,\hat{\xi}^{k,1}) - \sum_{k=1}^{\nu} \sum_{j=1}^{n_2} \int_{S^k} [\pi^*(x,\hat{\xi}^{k,1})(W_{\cdot,j} - \hat{W}_{\cdot,j}^{k,1}) - (q_j - \hat{q}_j^{k,1})] y_j^*(x,\xi) d\mu \\
\geq & \sum_{k=1}^{\nu} p^k Q(x,\hat{\xi}^{k,1}) - \sum_{k=1}^{\nu} \sum_{j=1}^{n_2} \Big[y_j^{k,UB} \int_{S^k} [\pi^*(x,\hat{\xi}^{k,1})(W_{\cdot,j} - \hat{W}_{\cdot,j}^{k,1}) - (q_j - \hat{q}_j^{k,1})]^+ d\mu \\
& - y_j^{k,LB} \int_{S^k} [-\pi^*(x,\hat{\xi}^{k,1})(W_{\cdot,j} - \hat{W}_{\cdot,j}^{k,1}) + (q_j - \hat{q}_j^{k,1})]^+ d\mu \Big] \\
= & \sum_{k=1}^{\nu} p^k Q(x,\hat{\xi}^{k,1}) - \epsilon_1^{\nu}(x, \{\hat{\xi}^{k,1}\}_{k=1}^{\nu})
\end{aligned}
$$

The first equality follows from the definition of $y^*(x,\xi)$. The next equality is due to the last summation being equal to zero as a result of $y^*(x,\xi)$ being feasible. The first inequality is due to $\pi^*(x,\hat{\xi}^{k,1})$ being dual feasible and $y^*(x,\xi) \geq 0$. The next equality results from rearranging terms. The last inequality follows from the definition of $y^{k,UB}$ and $y^{k,LB}$.

The second part of the proof is similar to the first.

$$
\begin{aligned}
Q(x) = & \int_{\Xi} \pi^*(x,\xi)[h - Tx]d\mu \\
\leq & \int_{\Xi} \pi^*(x,\xi)[h - Tx]d\mu + \sum_{k=1}^{\nu} \int_{S^k} (q - \pi^*(x,\xi)W)y^*(x,\hat{\xi}^{k,2})d\mu \\
= & \int_{\Xi} \pi^*(x,\xi)[h - Tx]d\mu + \sum_{k=1}^{\nu} \int_{S^k} (q - \pi^*(x,\xi)W)y^*(x,\hat{\xi}^{k,2})d\mu \\
& - \sum_{k=1}^{\nu} \int_{S^k} \pi^*(x,\xi)[\hat{h}^{k,2} - \hat{T}^{k,2}x - \hat{W}^{k,2}y^*(x,\hat{\xi}^{k,2})]d\mu \\
= & \sum_{k=1}^{\nu} p^k Q(x,\hat{\xi}^{k,2}) + \sum_{k=1}^{\nu} \sum_{i=1}^{m_2} \int_{S^k} \pi_i^*(x,\xi) \left[(h_i - \hat{h}_i^{k,2}) - (T_{i,\cdot} - \hat{T}_{i,\cdot}^{k,2})x \right. \\
& \left. -(W_{i,\cdot} - \hat{W}_{i,\cdot}^{k,2})y^*(x,\hat{\xi}^{k,2}) \right] d\mu \\
\leq & \sum_{k=1}^{\nu} p^k Q(x,\hat{\xi}^{k,2}) + \sum_{k=1}^{\nu} \sum_{i=1}^{m_2} \pi_i^{k,UB} \int_{S^k} \left[(h_i - \hat{h}_i^{k,2}) - (T_{i,\cdot} - \hat{T}_{i,\cdot}^{k,2})x \right. \\
& \left. -(W_{i,\cdot} - \hat{W}_{i,\cdot}^{k,2})y^*(x,\hat{\xi}^{k,2}) \right]^+ d\mu \\
& - \sum_{k=1}^{\nu} \sum_{i=1}^{m_2} \pi_i^{k,LB} \int_{S^k} \left[-(h_i - \hat{h}_i^{k,2}) + (T_{i,\cdot} - \hat{T}_{i,\cdot}^{k,2})x \right. \\
& \left. +(W_{i,\cdot} - \hat{W}_{i,\cdot}^{k,2})y^*(x,\hat{\xi}^{k,2}) \right]^+ d\mu \\
= & \sum_{k=1}^{\nu} p^k Q(x,\hat{\xi}^{k,2}) + \epsilon_2^{\nu}(x, \{\hat{\xi}^{k,2}\}_{k=1}^{\nu}).
\end{aligned}
$$

The first equality follows from the definition of $\pi^*(x,\xi)$. The next inequality follows from $\pi^*(x,\xi)$ being dual feasible and $y^*(x,\hat{\xi}^{k,2}) \geq 0$. The next equality follows from the last summation being equal to zero due to $y^*(x,\hat{\xi}^{k,2})$ being feasible. The next equality results from rearranging terms. The last inequality follows from the definition of $\pi^{k,LB}$ and $\pi^{k,UB}$, establishing the final result. □

In general, $y^{k,UB}$, $y^{k,LB}$, $\pi^{k,UB}$ and $\pi^{k,LB}$ can be difficult to determine efficiently. However, any valid bounds on $y^*(x,\xi)$ and $\pi^*(x,\xi)$ will yield valid bounds for $Q(x)$. Ideally we would find sufficiently tight bounds for these second stage decision variables associated with each cell of the partition without too much computational effort. Notice when $Q(x,\xi)$ has multiple optima, we may select any of the optimal solutions when defining $\pi^*(x,\xi)$ and $y^*(x,\xi)$. The choice of optimal solutions will in turn determine appropriate values for the decision variable bounds.

Proposition 1 is valid for any feasible first stage solution, x, such as one generated using the

Sample Average Approximation (SAA) described by Shapiro and Homem de Mello (1998) and Kleywegt et al. (2002). This establishes bounds on the true objective function value, since:

$$\hat{z}^{\nu}(x, \{\hat{\xi}^{k,1}\}_{k=1}^{\nu}) - \epsilon_1^{\nu}(x, \{\hat{\xi}^{k,1}\}_{k=1}^{\nu}) \leq cx + Q(x) = z(x)$$

and

$$z(x) \leq \hat{z}^{\nu}(x, \{\hat{\xi}^{k,2}\}_{k=1}^{\nu}) + \epsilon_2^{\nu}(x, \{\hat{\xi}^{k,2}\}_{k=1}^{\nu}).$$

The following corollary to Proposition 1 establishes bounds on the optimal objective function value of 2SLP, $z(x^*)$.

Corollary 1. *If the assumptions of Proposition 1 hold, then*

$$z^{\nu *} - \epsilon^{\nu} \leq z(x^*) \leq z(x^{\nu *}) \leq z^{\nu *} + \epsilon_2^{\nu}(x^{\nu *}, \{\xi^k\}_{k=1}^{\nu})$$

where x^* is an optimal first stage solution of 2SLP and

$$\begin{aligned} \epsilon^{\nu} = \quad & \textstyle\sum_{k=1}^{\nu} \sum_{j=1}^{n_2} y_j^{k,UB} \int_{S^k} [w^{k,\nu *}(W_{\cdot,j} - W_{\cdot,j}^k) - (q_j - q_j^k)]^+ d\mu \\ & - \textstyle\sum_{k=1}^{\nu} \sum_{j=1}^{n_2} y_j^{k,LB} \int_{S^k} [-w^{k,\nu *}(W_{\cdot,j} - W_{\cdot,j}^k) + (q_j - q_j^k)]^+ d\mu \end{aligned}$$

Proof. The proof of the first inequality follows from $z(x^*) = cx^* + Q(x^*)$ and Proposition 1 with $x = x^*$, $\hat{\xi}^{k,1} = \hat{\xi}^{k,2} = \xi^k$, and $\pi^*(x, \xi^k)$ replaced by $w^{k,\nu *}$. The second inequality follows from x^* being optimal and $x^{\nu *}$ being feasible in the first stage constraints. The last inequality is a direct consequence of Proposition 1. □

We emphasize that the bounds of Proposition 1 and Corollary 1 are 100% confidence bounds. This is in contrast to statistical confidence bounds such as those discussed in Mak et al. (1999) for estimating the true objective function value of a feasible or optimal first stage solution. Notice that the expressions defining the upper and lower estimation errors, ϵ^{ν} and ϵ_2^{ν}, are summations of integrals. The integrands depend on a linear function of the difference between

random variables in ξ and their conditional mean values. Intuitively, if the partition of the support for the random variables is refined, the cells in the partition become smaller, and these differences become negligible. If the partition is hyper-rectangular, and the joint distribution function decomposes into a product of single-variable functions, then the multi-dimensional integrals decompose into the product of one-dimensional integrals.

Iterative refinement of the partition $\{S^k\}_{k=1}^{\nu}$ to improve lower and upper bounds on the optimal objective function value for 2SLP has been used by Birge and Wets (1986). In Denton and Gupta (2003), a less general version of Corollary 1 was applied to an appointment scheduling problem. The result was used with iterative refinement of the partition $\{S^k\}_{k=1}^{\nu}$ to produce lower and upper bounds on the optimal objective function value for 2SLP that converge to $z(x^*)$ as ν increases. During the ν^{th} refinement of the partition, ξ is restricted to the k^{th} cell, S^k, and the optimal values of y in the recourse problem $Q(x^{\nu *}, \xi)$ with $\xi \in S^k$ are restricted to a subset of the possible values associated with $\xi \in \Xi$. The authors used these restrictions, the problem structure, and complementary slackness to find suitable bounds for $\pi_i^{k,LB}$ and $\pi_i^{k,UB}$ through a forward and backward recursion that exploited the special structure of the appointment scheduling problem. From the computational experience with this problem, the authors found that the upper bound converged more slowly to the optimal value, $z(x^*)$, than the lower bound. So, a methodology that provides tight bounds for the decision variables may help to improve convergence and the overall runtime to achieve a given level of accuracy for 2SLP. In this chapter, we generalize this algorithm to 2SLPs consistent with the assumptions of Section 3.3 and provide methods for improving the bounds on the second stage decision variables when $\xi \in S^k$. The sequential bounding algorithm for 2SLP is outlined in Algorithm 1.

A sequential approach to defining the partition begins with $\nu = 1$, and $S^1 = \Xi$. We generate a two cell partition, $\nu = 2$, by splitting the first cell into two. The process continues, splitting one of the cells in the partition to generate two new cells at every iteration. Many different approaches to refining the partition are possible. In our implementation in Section 3.6, we will

Algorithm 1 Sequential Bounding for 2SLP

1: Let ν index the iteration. Set $\nu = 0$.
2: Set $\nu = \nu + 1$. Solve the PMVP for the partition $\{S^k\}_{k=1}^{\nu}$. Let $(x^{\nu *}, \{y^{k,\nu *}\}_{k=1}^{\nu})$ be the optimal solution.
3: Determine upper and lower bounds for optimal primal and dual second stage decision variables using one of the methods described in Section 3.5.
4: Calculate ϵ^{ν} and $\epsilon_2^{\nu}(x^{\nu *})$ using Proposition 1 with the bounds calculated in Step 3. If $\max\{|\epsilon^{\nu}|, |\epsilon_2^{\nu}(x^{\nu *})|\} \leq tolerance$, then stop. Otherwise, go to Step 5.
5: Refine the current partition $\{S^k\}_{k=1}^{\nu} \rightarrow \{S^k\}_{k=1}^{\nu+1}$ and return to Step 2.

assume that cells are split perpendicular to one of the coordinate axes, say the i^{th} axis, and parallel to all the others to maintain hyper-rectangular cells. The cell is always split at the conditional mean value, ξ_i^k. We avoid splitting cells with only a single value, $S^k = \{\xi^k\}$, since there is no difference between PMVP and 2SLP if every cell is a singleton.

A few comments about the convergence of sequential bounding are in order. In reference to Equation 3.1, Birge and Wets (1986) show that $Q(x, \xi)$ is a convex function of ξ, which is the case when random variables only appear in T and h, then sequential bounding will converge as long as $\max_k\{P[\xi \in S^k]\} \rightarrow 0$ as $\nu \rightarrow \infty$. In our computational experiments in Section 3.6, random variables only appear in h; so, convergence is guaranteed. Convergence is also possible with random variables appearing elsewhere. Pennanen (2009) provides convergence conditions for one such case, which rely on the existence of feasible solutions satisfying all the constraints with strict inequality. In the case where all random variables are discretely distributed, for example, sequential bounding will always terminate after a finite number of iterations.

In any implementation of sequential bounding, a decision must be made as to which cell of the partition to split and in which coordinate direction the split should be made. One approach is to select the cell based on one or more of the error terms ϵ^{ν}, ϵ_1^{ν}, and ϵ_2^{ν}. If we let $\tilde{W}$ be an upper bound on each component of $w^{k,\nu *}$, then an upper bound on one term of ϵ^{ν} follows:

$$\begin{aligned} & y_j^{k,UB} \int_{S^k} [w^{k,\nu *}(W_{\cdot,j} - W_{\cdot,j}^k) - (q_j - q_j^k)]^+ d\mu \\ \leq \; & \sum_{i=1}^{m_2} y_j^{k,UB} \tilde{W} \int_{S^k} |W_{i,j} - W_{i,j}^k| d\mu + y_j^{k,UB} \int_{S^k} |q_j - q_j^k| d\mu \end{aligned}$$

A similar procedure can be carried out for the terms of ϵ_1^ν and ϵ_2^ν. Each term in the upper bound corresponds to a different random variable component of ξ. So, in our implementation we select the term with the largest contribution to the upper bound to determine the cell and coordinate direction along which the cell splitting will take place.

Although we did not specify any of the components of the first stage decision variable vector, x, to be discrete in 2SLP, this can be accomodated in the sequential bounding algorithm. Since all the decision variables are bounded, discrete components of x can only take a finite number of possible values. The bounds in Proposition 1 and Corollary 1 remain valid if we fix the integer variables in x. When convergence is guaranteed, the set of fixed integer decision variables corresponding to an optimal solution, x^*, will have a smaller cost than all non-optimal sets of integers. At each iteration of sequential bounding, solving the PMVP of Equation 3.7 with some of the x variables restricted to integer values corresponds to using the same partitioning scheme for every possible set of fixed decision variables.

3.4.1 Restricted Recourse Bounds

In this subsection we explain how to use restricted recourse, the approach of Morton and Wood (1999), to generate an upper bound on $z(x^{\nu *})$. This is achieved by adding constraints or fixing decision variables in the recourse problem. Such restrictions change the recourse problem, but any upper bound for the restricted recourse problem is valid for the original recourse problem. The upper bound on $z(x^{\nu *})$ provided by Corollary 1 can be less for the restricted recourse problem than the unrestricted problem. This approach is used to improve the convergence of the sequential bounding algorithm by adding constraints associated with each cell of the random variable support partition.

First, we define the restricted recourse problem:

$$\begin{aligned} \widetilde{Q}(x,\xi) = \text{Min} \quad & qy \\ \text{s.t.} \quad & Tx + Wy = h \\ & \widetilde{W}y + \widetilde{V}y' = \widetilde{h} \\ & y, y' \geq 0 \end{aligned}$$

We see that the definition of $\widetilde{Q}(x,\xi)$ is the same as $Q(x,\xi)$ with one or more additional constraints. We assume $\widetilde{Q}(x,\xi)$ is feasible for all x and all $\xi \in \Xi$. The new constraints are added after iteration ν of the sequential bounding algorithm so that the matrices $\widetilde{W}$ and $\widetilde{V}$ as well as the vector $\widetilde{h}$ are constant for all values of ξ in the k^{th} cell, S^k, of the random variable support partition for $k = 1, \ldots, \nu$. Furthermore, we assume that there exists a $y' \geq 0$ such that $\widetilde{W}y^{k,\nu *} + \widetilde{V}y' = \widetilde{h}$ for every $k = 1, \ldots, \nu$ and $\xi \in \Xi$. This ensures that the new constraints do not cut off the second stage optimal solution of the PMVP.

We define $\widetilde{y}^*(x,\xi)$ to be a vector of optimal y variables in $\widetilde{Q}(x,\xi)$. We define $\widetilde{\pi}^*(x,\xi)$ to be a vector of optimal dual decision variables for $\widetilde{Q}(x,\xi)$ corresponding to the primal constraints $Tx + Wy = h$. We also define $\widetilde{Q}^\nu(x) = \sum_{k=1}^{\nu} p^k \widetilde{Q}(x,\xi^k)$ to be the restricted recourse of the PMVP. The following corollary gives an upper bound for $z(x^{\nu *})$:

Corollary 2. *If the assumptions and hypotheses of Proposition 1 hold, then*

$$cx^{\nu *} + \widetilde{Q}^\nu(x^{\nu *}) = z^{\nu *}$$

and

$$z^{\nu *} - \widetilde{\epsilon}^\nu \leq z(x^*) \leq z(x^{\nu *}) \leq z^{\nu *} + \widetilde{\epsilon}_2^\nu(x^{\nu *})$$

where

$$\begin{aligned}
\widetilde{\epsilon}^{\nu} = \quad & \sum_{k=1}^{\nu}\sum_{j=1}^{n_2}\Big[\widetilde{y}_j^{k,UB}\int_{S^k}[\pi^*(x^{\nu*},\xi^k)(W_{\cdot,j}-W_{\cdot,j}^k)-(q_j-q_j^k)]^+d\mu \\
& -\widetilde{y}_j^{k,LB}\int_{S^k}[-\pi^*(x^{\nu*},\xi^k)(W_{\cdot,j}-W_{\cdot,j}^k)+(q_j-q_j^k)]^+d\mu\Big] \\
\widetilde{\epsilon}_2^{\nu}(x^{\nu*}) = \quad & \sum_{k=1}^{\nu}\sum_{i=1}^{m_2}\widetilde{\pi}^{k,UB}\int_{S^k}\Big[(h_i-h_i^k)-(T_{i,\cdot}-T_{i,\cdot}^k)x^{\nu*} \\
& -(W_{i,\cdot}-W_{i,\cdot}^k)y^*(x^{\nu*},\xi^k)\Big]^+d\mu \\
& -\sum_{k=1}^{\nu}\sum_{i=1}^{m_2}\widetilde{\pi}^{k,LB}\int_{S^k}\Big[-(h_i-h_i^k)+(T_{i,\cdot}-T_{i,\cdot}^k)x^{\nu*} \\
& +(W_{i,\cdot}-W_{i,\cdot}^k)y^*(x^{\nu*},\xi^k)\Big]^+d\mu
\end{aligned}$$

and we define $\widetilde{y}^{k,UB}$,$\widetilde{y}^{k,LB}$,$\widetilde{\pi}^{k,UB}$, *and* $\widetilde{\pi}^{k,LB}$ *to be bounds on the second stage restricted recourse decisions:*

$$\begin{aligned}
0 \le \widetilde{y}^{k,LB} \le \widetilde{y}^*(x^{\nu*},\xi) \le \widetilde{y}^{k,UB} \\
\widetilde{\pi}^{k,LB} \le \widetilde{\pi}^*(x^{\nu*},\xi) \le \widetilde{\pi}^{k,UB}
\end{aligned}$$

for all $\xi \in S^k$ *and* $k = 1,\ldots,\nu$.

Proof. This follows directly from the definition of the recourse problem, $\widetilde{Q}(x,\xi)$, Corollary 1 and the fact that $x^{\nu*}$ and $y^{k,\nu*}$, for $k = 1,\ldots,\nu$, is feasible and optimal for the PMVP with restricted recourse. □

In closing this subsection, notice that Corollary 2 is very similar to Proposition 1 applied to the PMVP. In particular, if the bounds on the restricted recourse decisions do not depend on the cell of the random variable support partition and are valid bounds for the decisions in $Q(x,\xi)$ for all $\xi \in \Xi$, then the restricted recourse bounds are identical to those from Proposition 1. Methods for finding such bounds on the optimal recourse decisions are the subject of Section 3.5.

3.4.2 Bounds from Optimized Outcomes

In this subsection, we describe a method to generate an upper bound on $z(x^*)$ by selecting representative outcomes for each cell of the random variable support partition. We assume that random variables only appear in the right hand side vector, h, of the recourse problem and all of these random variables are independent.

In general, the upper bound for $Q(x)$ in Proposition 1 holds for a variety of vectors $\hat{\xi}^{k,2} \in S^k$ for $k = 1, \ldots, \nu$. This suggests replacing each random variable h_i with a decision variable r_i and minimizing the upper bound using the following mathematical program:

$$
\begin{aligned}
\text{Min} \quad & cx + \textstyle\sum_{k=1}^{\nu} p^k q^k y^k + \sum_{k=1}^{\nu} \sum_{i=1}^{m_2} \left(\pi_i^{k,UB} \int_{S^k} \left[h_i - r_i^k \right]^+ d\mu - \pi_i^{k,LB} \int_{S^k} \left[r_i^k - h_i \right]^+ d\mu \right) \\
\text{s.t.} \quad & Ax = b \\
& x \geq 0 \\
& Tx + Wy^k = r^k, \text{ for all } k = 1, \ldots, \nu \\
& y^k \geq 0, \text{ for all } k = 1, \ldots, \nu \\
& a^{(k)} \leq r^k \leq b^{(k)}, \text{ for all } k = 1, \ldots, \nu
\end{aligned}
$$

In general, the objective function for this mathematical program is nonlinear in the decision variables. However, suppose F is the cumulative distribution function of h_i. Taking the derivative of the objective function with respect to r_i^k yields the expression:

$$(\pi_i^{k,UB} - \pi_i^{k,LB}) F(r_i^k) - \pi_i^{k,UB} F(b_i^{(k)}) + \pi_i^{k,LB} F(a_i^{(k)}).$$

Since this is an increasing function in r_i^k, the objective function is a sum of single-variable functions that are convex in each of the decision variables. Each such function can be approximated with a piecewise linear function of evenly spaced points lying on the graph of the function. In the computational experiments of Section 3.6, we used 9 evenly spaced points for each function.

These approximating functions are greater than or equal to the original convex functions and are incorporated in the mathematical program with additional decision variables and linear constraints (see Bertsimas and Tsitsiklis, 1997, for example).

In general, the bounds on the dual decision variables, $\pi_i^{k,UB}$ and $\pi_i^{k,LB}$, can depend on the first stage decisions, x. In general, optimal dual recourse decision variables are nonlinear and non-convex functions of x; so, the bounds on these variables can be as well. To make the mathematical program tractable, we assume $\pi_i^{k,UB}$ and $\pi_i^{k,LB}$ are the same value for every k and all values of x. Such global bounds are discussed in more detail in Section 3.5.1. The resulting mathematical program is a standard linear program whose optimal solution is an upper bound for the optimal objective of 2SLP, $z(x^*)$. The optimal first stage decisions from the linear program are used to calculate tighter bounds on the dual decision variables and a tighter bound on $z(x^*)$. Methods for tighter bounds on the dual decision variables are discussed in the next section.

3.5 Methods for Bounding Recourse Variables

In order to use Proposition 1 and sequential bounding for 2SLP, we need to determine decision variable bounds, $y^{k,UB}$,$y^{k,LB}$,$\pi^{k,UB}$, and $\pi^{k,LB}$ for $k = 1, \ldots, \nu$. When Equation 3.2 does not have multiple optimal solutions, we can find the smallest possible value for the ℓ^{th} component of $\pi^{k,UB}$ by solving the nonlinear program below. Otherwise, this nonlinear program provides an upper bound and value for $\pi_\ell^{k,UB}$. Here, $W_{i,j}^{k,LB}$ and $W_{i,j}^{k,UB}$ are the lower and upper bounds on the random variable $W_{i,j}$, respectively, defined by the cell of the partition when $\xi \in S^k$. Similarly, $q_j^{k,LB}$ and $q_j^{k,UB}$ are bounds on the random variable q_j when $\xi \in S^k$:

$$\text{Max} \quad \pi_\ell \tag{3.9}$$

$$\text{s.t.} \quad L_i^{(k)} \leq r_i \leq U_i^{(k)}, \forall i$$

$$W_{i,j}^{k,LB} \leq r_{i,j} \leq W_{i,j}^{k,UB}, \forall i,j$$

$$q_j^{k,LB} \leq s_j \leq q_j^{k,UB}, \forall j$$

$$\textstyle\sum_{j=1}^{n_2} r_{i,j} y_j = r_i, \forall i \tag{3.10}$$

$$\textstyle\sum_{i=1}^{m_2} \pi_i r_{i,j} \leq s_j, \forall j \tag{3.11}$$

$$\textstyle\sum_{j=1}^{n_2} s_j y_j \leq \sum_{i=1}^{m_2} \pi_i r_i \tag{3.12}$$

$$y_j \geq 0, r_{i,j}, s_j, \pi_i, \text{ and } r_i \text{ unrestricted in sign } \forall i,j$$

A similar nonlinear program can be used to calculate a lower bound for $\pi_\ell^{k,LB}$ by minimizing the objective function instead of maximizing:

$$\text{Min} \quad \pi_\ell \tag{3.13}$$

$$\text{s.t.} \quad L_i^{(k)} \leq r_i \leq U_i^{(k)}, \forall i$$

$$W_{i,j}^{k,LB} \leq r_{i,j} \leq W_{i,j}^{k,UB}, \forall i,j$$

$$q_j^{k,LB} \leq s_j \leq q_j^{k,UB}, \forall j$$

$$\textstyle\sum_{j=1}^{n_2} r_{i,j} y_j = r_i, \forall i \tag{3.14}$$

$$\textstyle\sum_{i=1}^{m_2} \pi_i r_{i,j} \leq s_j, \forall j \tag{3.15}$$

$$\textstyle\sum_{j=1}^{n_2} s_j y_j \leq \sum_{i=1}^{m_2} \pi_i r_i, \tag{3.16}$$

$$y_j \geq 0, r_{i,j}, s_j, \pi_i, \text{ and } r_i \text{ unrestricted in sign } \forall i,j$$

Equations 3.10 and 3.14 are the primal constraints associated with the recourse problem in Equation 3.1. Equations 3.11 and 3.15 are the dual constraints associated with Equation 3.5.

Equations 3.12 and 3.16 correspond to the strong duality condition for the recourse linear program. The collection of all these nonlinear programs for all $\ell = 1, \ldots, m_2$, as well as the nonlinear programs obtained by minimizing and maximizing over each component of y, will be referred to as the bounds nonlinear programs (BNPs).

The BNPs are not necessarily convex optimization problems. For example $\sum_{i=1}^{m_2} \pi_i r_i - s_j$ is not a convex function in the decision variables; so, finding optimal solutions to the BNPs may require a prohibitive amount of computation time in general. After solving the PMVP to obtain $x^{\nu *}$ and $\{y^{k*}\}_{k=1}^{\nu}$ and the corresponding optimal second stage dual decision variables, $\{\pi^{k*}\}_{k=1}^{\nu}$, a feasible solution to the BNPs is readily available by setting $r = h^k - T^k x^{\nu *}$, $r_{i,j} = W_{i,j}^k$, $s_j = q_j^k$, $y = y^{k*}$, and $\pi = \pi^{k*}$. However, we need upper bounds for the maximization problems and lower bounds for the minimization problems, which are not provided by sub-optimal feasible solutions. Approximation methods for these difficult to solve BNPs are the subject of the remainder of this section.

3.5.1 Global Bounds

Consistent with the assumptions of Section 3.3, we assume $\{\pi | \pi W \leq q\}$ is a bounded set. Otherwise, it is possible to add constraints to $\pi W \leq q$, since every optimal solution to Equation 3.5 is bounded. Assuming W and q are constant, we begin by describing how to obtain global bounds, $\pi_i^{k,LB}$ and $\pi_i^{k,UB}$, independent of the cell k in the partition. Since the feasible region remains the same in the dual of $Q(x^{\nu *}, \xi)$ for all values of $\xi \in \Xi$, we can establish an upper bound on the dual variable $\pi_i(x^{\nu *}, \xi)$ by solving the following linear program:

$$\begin{aligned} \text{Max} \quad & \pi_i \\ \text{s.t.} \quad & \pi W \leq q \end{aligned}$$

A lower bound is found by minimizing instead of maximizing, and we will refer to all of these associated linear programs as the dual bounds linear programs (DBLPs). Although this

involves the solution of $2m_2$ linear programming problems, the bounds are valid for each value of k and ν; so, the problems only need to be solved once before the first partition refinement of the first iteration of the algorithm.

Although this method for finding bounds on the second stage dual variables does not take advantage of the fact that ξ is restricted to S^k for cell k, and therefore are not likely to be very tight, they provide a default set of a priori bounds which can be used to find better bounds. As pointed out by Birge (1985a), such global bounds on the decision variables are often readily apparent from examination of the constraints.

3.5.2 Linear Programming Relaxations

In this section, we use linear programming relaxations of the BNPs. Unlike the global bounds on the dual decision variables, the bounds found in this section take advantage of individual cells in the partition of the support set. We also relax the assumption that W and q are constant.

We describe two ways to obtain a linear program relaxation of the BNPs. The first approach is to replace all products of decision variables appearing in the constraints with new decision variables. Constraints involving the new decision variables are derived from range constraints on the original decision variables. This approach is described in Al-Khayyal and Falk (1983) and is one of the techniques used in the more general Reformulation-Linearization Technique (see Sherali and Adams, 1998, and references therein). This is the approach we use in Section 3.6.

We illustrate the technique for a typical constraint in the BNPs. Let η_i be a decision variable of a BNP for $i = 1, \ldots, N$. Let C, A_i, L_i, U_i, and $B_{i,j}$ be fixed values for $i, j = 1, \ldots, N$. Consider the following nonlinear constraint and range constraints on the decision variables:

$$\sum_{i=1}^{N} A_i \eta_i + \sum_{i=1}^{N} \sum_{j=1}^{N} B_{i,j} \eta_i \eta_j \leq C$$
$$L_i \leq \eta_i \leq U_i, \text{ for } i = 1, \ldots, N$$

A linear program relaxation is obtained by replacing $\eta_i\eta_j$ with a new decision variable $v_{i,j}$. We obtain four additional constraints involving $v_{i,j}$ by multiplying the four range constraints involving η_i and η_j. Finally, we replace the nonlinear constraint in the BNP with the following linear constraints:

$$\sum_{i=1}^{N} A_i\eta_i + \sum_{i=1}^{N}\sum_{j=1}^{N} B_{i,j}v_{i,j} \leq C$$

$$(\eta_i - L_i)(\eta_j - L_j) = v_{i,j} - L_i\eta_j - L_j\eta_i + L_iL_j \geq 0$$

$$(\eta_i - L_i)(U_j - \eta_j) = U_j\eta_i - L_iU_j - v_{i,j} + L_i\eta_j \geq 0$$

$$(U_i - \eta_i)(\eta_j - L_j) = U_i\eta_j - v_{i,j} - U_iL_j + L_j\eta_i \geq 0$$

$$(U_i - \eta_i)(U_j - \eta_j) = U_iU_j - U_j\eta_i - U_i\eta_j + v_{i,j} \geq 0$$

A second approach for obtaining a linear program relaxation is described next. Here, we transform the nonlinear programs in two ways. First, we substitute $\pi = \pi^+ - \pi^-$ and add nonnegativity constraints: $\pi^+, \pi^- \geq 0$. Next, we linearize the nonlinear constraints in Equation 3.9, utilizing bounds on $r_{i,j}, r_i$, and s_j given in the constraints. The relaxed feasible region, P, is defined by the following set of constraints:

$$L_i^{(k)} \leq r_i \leq U_i^{(k)}, \forall i$$

$$\sum_{j=1}^{n_2} W_{i,j}^{k,LB} y_j \leq r_i, \forall i$$

$$\sum_{j=1}^{n_2} W_{i,j}^{k,UB} y_j \geq r_i, \forall i$$

$$\sum_{i=1}^{m_2} [\pi_i^+ W_{i,j}^{k,LB} - \pi_i^- W_{i,j}^{k,UB}] \leq q_j^{k,UB}, \forall j$$

$$\sum_{j=1}^{n_2} q_j^{k,LB} y_j \leq \sum_{i=1}^{m_2} [\pi_i^+ U_i^{(k)} - \pi_i^- L_i^{(k)}]$$

$$y_j, \pi_i^+, \pi_i^- \geq 0, \text{ and } r_i \text{ u.r.s. } \forall i, j$$

Consistent with the assumptions of Section 3.3, we assume every component of π^+ and π^- is bounded above in P. The detailed description is given in Algorithm 2 for a fixed cell index k and iteration index ν.

Algorithm 2 LP Relaxation for a fixed cell index k at iteration ν of Sequential Bounding

Require: Iteration ν is complete for Algorithm 1.
Maximize and minimize each component of π^+ and π^- subject to the constraints in P. Denote these optimal values as $\pi^{+,max}, \pi^{+,min}, \pi^{-,max}, \pi^{-,min}$
if $\pi_i^{+,max} = 0$ **then**
$\pi_i^{k,UB} = -\pi_i^{-,min}$ and $\pi_i^{k,LB} = -\pi_i^{-,max}$.
else if $\pi_i^{-,max} = 0$ **then**
$\pi_i^{k,UB} = \pi_i^{+,max}$ and $\pi_i^{k,LB} = \pi_i^{+,min}$.
else
$\pi_i^{k,UB} = \pi_i^{+,max}$ and $\pi_i^{k,LB} = -\pi_i^{-,max}$
end if

3.5.3 Complementary Slackness Improvements

In this section, we use complementary slackness conditions on the second stage recourse problem to iteratively improve bounds on the primal and dual decision variables. As in the last subsection, the bounds on the dual decision variables in this subsection take advantage of individual cells in the partition of the support set. Throughout this section, we assume that W and q are constant.

For the recourse problem, $Q(x^{\nu*}, \xi)$, the optimal second stage primal decision variables associated with cell k can vary depending on the value of $\xi \in S^k$. We begin by pointing out that nonlinear programs similar to the BNP of Equation 3.9 can be used to find bounds on the primal decision variables of the recourse problem. For example, the following nonlinear program provides the largest optimal value of y_j associated with cell k:

$$
\begin{aligned}
\text{Max} \quad & y_j \\
\text{s.t.} \quad & L^{(k)} \leq r \leq U^{(k)} \\
& Wy = r \\
& \pi W \leq q \\
& qy \leq \pi r \\
& y \geq 0, \pi \text{ and } r \text{ unrestricted in sign}
\end{aligned}
$$

Due to the computational difficulties in solving the above nonlinear program, we focus on finding upper bounds for problems that maximize y_j and lower bounds for problems that minimize y_j. The following linear program provides such an upper bound for y_j by removing constraints from the above nonlinear program:

$$
\begin{aligned}
\text{Max} \quad & y_j \\
\text{s.t.} \quad & L^{(k)} \leq r \leq U^{(k)} \\
& Wy = r \\
& y \geq 0, r \text{ unrestricted in sign}
\end{aligned}
$$

A lower bound for y_j is found by changing this from a maximization problem to a minimization problem, and we refer to all these linear programs as the primal bounds linear programs (PBLPs). Observe that, unlike the DBLPs, the PBLPs allow us to use the restriction $\xi \in S^k$ for cell k.

For $i = 1, \ldots, m_2$, let l_i and u_i be the lower and upper bounds for the decision variables in Equation 3.2 found using the DBLPs. We may find that some of the constraints $\pi W \leq q$ are

redundant. This is the case for constraint j when

$$\sum_{i \in W_{\cdot j}^{+}} W_{ij} u_i + \sum_{i \in W_{\cdot j}^{-}} W_{ij} l_i < q_j,$$

where we define $W_{\cdot j}^{+} = \{i : W_{ij} \geq 0\}$ and $W_{\cdot j}^{-} = \{i : W_{ij} < 0\}$. Let $F = \{j : \text{constraint } j \text{ is not active in Equation 3.2 }\}$.

After solving the DBLPs, if the set F is not empty, we can potentially improve the bounds on the primal second stage problem by modifying the PBLPs. By complimentary slackness, we know that $y_j = 0$ for any $j \in F$. We can modify the constraint set of the PBLPs to obtain a new set $C = \{y | Wy \leq U^{(k)}, Wy \geq L^{(k)}, y \geq 0 \text{ and } y_j = 0 \text{ for all } j \in F\}$. Minimizing each variable y_i in turn over this set C will provide lower bounds $\{y_j^{LB,*}\}_{j=1}^{n}$ for the primal second stage variables when $\xi \in S^k$. We refer to all of these linear programming problems as the improved primal bounds linear programs (IPBLPs). As we will see in the next section, these bounds can be used to improve the bounds on the dual second stage variables.

After solving the IPBLPs and finding bounds on the primal second stage variables when $\xi \in S^k$, we define $E = \{j : y_j^{LB,*} > 0\}$. Returning to the DBLPs, we impose the complementary slackness condition that $\sum_{i=1}^{m_2} \pi_i W_{ij} = q_j$ for all $j \in E$. We refer to these modified DBLPs as the improved dual bounds linear programs (IDBLPs). If the bounds are improved to such a degree that some new constraints are left redundant in the constraints $\pi W \leq q$, we may add new elements to the set F in the IPBLPs. This has the potential to improve the bounds for the primal decision variables, adding to the set E in the IDBLPs. This process of going back and forth between the improved dual and improved primal second stage problems can be repeated to improve the bounds. This iterative process eventually terminates because there are only a finite number of constraints in the dual problem that can be redundant, and a finite number of variables in the primal problem that can be greater than zero. Details are outlined in Algorithm 3.

Algorithm 3 Complementary Slackness for cell k at iteration ν of sequential bounding

Require: Iteration ν is complete for Algorithm 1.

1: Set $E, F = \emptyset$.
2: Set $\pi_i^{k,LB} = \min\{\pi_i | \pi W_{\cdot,j} \leq q_j \text{ for } j \notin E, \pi W_{\cdot,j} = q_j \text{ for } j \in E\}$ for all i.
3: Set $\pi_i^{k,UB} = \max\{\pi_i | \pi W_{\cdot,j} \leq q_j \text{ for } j \notin E, \pi W_{\cdot,j} = q_j \text{ for } j \in E\}$ for all i.
4: Set $F = F \cup \{j | \sum_i \pi_i^{k,UB} \max\{W_{i,j}, 0\} - \sum_i \pi_i^{k,LB} \max\{-W_{i,j}, 0\} < q_j\}$.
5: Set $y_j^{LB,*} = \min\{y_j | Wy \leq U^{(k)}, Wy \geq L^{(k)}, y_j \geq 0 \text{ for } j \notin F, y_j = 0 \text{ for } j \in F\}$ for all j.
6: Set $E = E \cup \{j | y_j^{LB,*} > 0\}$. If E does not change, then stop and return $\pi^{k,LB}$ and $\pi^{k,UB}$. Otherwise, go to Step 2.

3.5.4 Disjunctive Linear Programs

In this section we show how disjunctive linear programs can improve bounds on the optimal second stage dual decision variables beyond those found using the DBLPs or the complementary slackness approach of the last section. As in the last section, we assume here that W and q are constant.

The potential for improving the bounds is illustrated by the example in Figure 3.1 for the case where there are only two second stage dual variables π_1 and π_2 associated with the recourse problem $Q(x^{\nu*}, \xi)$ with $\xi \in S^k$. The shaded square represents all possible cost vectors centered at π^{k*}, the dual optimal solution corresponding to the primal problem $Q(x^{\nu*}, \xi^k)$ associated with the PMVP.

For any possible dual cost vector, we can graphically determine the optimal dual basic feasible solution in Figure 3.1. It is clear that only the circled basic feasible solutions in the feasible set $\{\pi | \pi W \leq q\}$ can be optimal solutions when ξ is restricted to the set S^k. All other basic feasible solutions are dominated by the solution π^{k*} for all possible dual cost vectors. Thus, tighter bounds for the optimal dual decision variables exist than those based solely on bounds obtained through consideration of the feasible region alone. Although we would not be able to improve the upper bounds in this case, we can improve the lower bounds substantially.

In the two-dimensional case, each line segment of the feasible region corresponds to an active constraint. Not all of the optimal basic feasible solutions, those that are circled, are located on

the same line. This means that none of the constraints in the dual problem are active for every optimal solution associated with $\xi \in S^k$. However, the complementary slackness approach in Section 3.5.3 depends on finding dual constraints that are active for all possible optimal solutions when $\xi \in S^k$. Otherwise, the set E in the IDBLPs will always be empty, and we would only be capable of finding bounds based solely on consideration of the feasible region alone, i.e. bounds found by solving the DBLPs.

Adding a disjunctive constraint to the DBLPs allows us to reduce the dual feasible region $\{\pi|\pi W \leq q\}$ to a smaller set, where dual feasible solutions dominated by π^{k*} have been removed. Suppose that every feasible solution in the set $\{\pi|\pi v_k \leq \pi^{k*} v_k \text{ for } k = 1, \ldots, m_2\}$ has a lower objective function value than, and is therefore dominated by, π^{k*}. For now, we will assume that $\{v_k\}_{k=1}^{m_2}$ is given, but for the case illustrated in Figure 3.1, observe that the set of any two vectors pointing in the direction of the dotted lines has the desired property. The upper bound on optimal values of π_i found by solving the corresponding DBLP is improved by solving the following disjunctive linear programming problem:

$$\begin{aligned} \max \quad & \pi_i \\ \text{s.t.} \quad & \pi W \leq q \\ & \pi \in \cup_{k=1}^{m_2} \{\pi|\pi v_k \geq \pi^{k*} v_k\} \end{aligned}$$

We reformulate this as the following mixed integer linear programming problem:

$$\begin{aligned} \max \quad & \pi_i \\ \text{s.t.} \quad & \pi W \leq q \\ & \pi v_k \geq \pi^{k*} v_k - M(1 - \delta_k) \text{ for all } k = 1, \ldots, m_2 \\ & \textstyle\sum_{k=1}^{m_2} \delta_k = 1 \\ & \delta_k \in \{0, 1\} \text{ for all } k = 1, \ldots, m_2 \end{aligned}$$

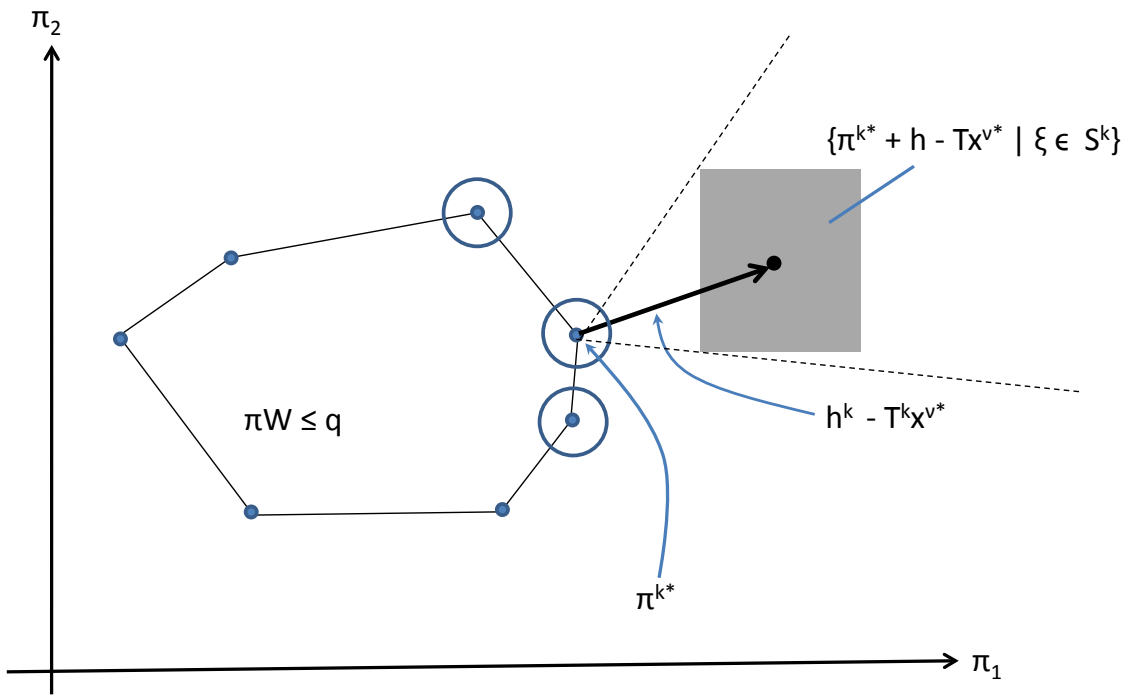

Figure 3.1: A 2-dimensional example of the dual problem for $Q(x^{\nu*}, \xi)$ with $\xi \in S^k$. From the possible dual cost vectors, we see that only the circled basic feasible solutions need to be considered when determining bounds for π_1 and π_2.

where M is a sufficiently large number.

There are a variety of methods for computing the set of vectors $\{v_k\}_{k=1}^{m_2}$ for use in the above mixed integer programming formulation. The method that we use generates the extreme rays of a simplicial cone containing the set of dual cost vectors $K = \{h - Tx^{\nu,*} | \xi \in S^k\}$. From Theorem 2.13.4.2.1 and Section 2.13.1.1 in Dattorro (2005), if $K \subset cone(\{v_i\}_{i=1}^{m_2})$, then $\{\pi | \pi v_i \leq 0 \text{ for all } i = 1, \ldots, m_2\} \subset K^*$, where K^* is the polar cone of K and $cone(\cdot)$ represents the conical hull of a set of vectors. This means that $\pi^{k*} + \{\pi | \pi v_i \leq 0 \text{ for all } i = 1, \ldots, m_2\} \subset \pi^{k*} + K^*$, or equivalently, $\{\pi | \pi v_i \leq \pi^{k*} v_i \text{ for all } i = 1, \ldots, m_2\} \subset \pi^{k*} + K^*$. Since every vector in $\pi^{k*} + K^*$ is dominated by π^{k*}, every feasible solution in the set $\{\pi | \pi v_i \leq \pi^{k*} v_i \text{ for all } i = 1, \ldots, m_2\}$ is also dominated by π^{k*}; so, the set of vectors $\{v_k\}_{k=1}^{m_2}$ are valid for constructing mixed integer programs.

There are many ways to generate the extreme rays of a simplicial cone containing the hyper-rectangle K. One approach involves constructing a matrix, $\mathcal{T}$, that transforms each of the unit coordinate direction vectors into the desired set of vectors $\{v_k\}_{k=1}^{m_2}$. We can build $\mathcal{T}$ by constructing its inverse, which transforms every vector in K into a set of vectors contained in the positive quadrant $\{\pi | \pi \geq 0\}$. For our computational experiments, $\mathcal{T}^{-1}$ was constructed as a product of invertible sparse matrices. These sparse matrices included orthogonal transformations, each of which is an identity matrix with one of the entries changed from $+1$ to -1. Each of the remaining sparse matrices is an identity matrix with one of the off-diagonal terms changed from 0 to a non-zero value λ. Each of these sparse linear transformations leaves all but one of the dimensions of its argument unchanged and is easily invertible; so, no more than m_2 of each type of matrix is necessary. Each orthogonal matrix generates no more than m_2 sign changes, and each of the remaining matrices involve no more than m_2 additions and multiplications. So, $\mathcal{T}^{-1}$ can be constructed in $O(m_2^2)$ time.

3.5.5 Exact Solutions to BNPs

In this subsection we discuss a special case of stochastic programs whose associated BNPs can be solved efficiently due to special structure. We will consider 2SLP with random variables only appearing in the right hand side vector h and such that $W = [\mathcal{L}|\mathcal{R}]$ is a partitioned matrix structured so that both $\mathcal{L}$ and $\mathcal{R}$ are lower triangular matrices in $\mathbb{R}^{m_2 \times m_2}$. We also assume that corresponding diagonal entries have opposite signs: $sign(\mathcal{L}_{i,i}) = -sign(\mathcal{R}_{i,i}) \neq 0$ for all $i = 1, \ldots, m_2$. Without loss of generality, we assume that $\mathcal{L}$ contains all columns with positive diagonal entries and $\mathcal{R}$ contains those with negative entries.

A number of important two-stage stochastic linear programs have this special structure, including the simple recourse problem described in Wets (1983), as well as appointment and operating room scheduling problems (see Denton and Gupta, 2003; Batun et al., 2011; Erdogan et al., 2011). Tandem queuing networks described in Chan and Schruben (2008) with random inter-arrival times and service times are also included in this category.

For fixed values of ξ and x, the optimal solution to the recourse problem can be found using recursion. Notice that every basis will be lower triangular, and the i^{th} column for any basis will either be from column i or column $m_2 + i$ in W. We denote the i^{th} entry of $h - Tx$ as $(h - Tx)_i$. Assuming $q \geq 0$, the optimal solution is:

$$y_1 = \begin{cases} \frac{(h-Tx)_1}{\mathcal{L}_{1,1}}, \text{ if } (h - Tx)_1 \geq 0 \\ 0, \text{ otherwise,} \end{cases}$$

$$y_{m_2+1} = \begin{cases} \frac{(h-Tx)_1}{\mathcal{R}_{1,1}}, \text{ if } (h - Tx)_1 < 0 \\ 0, \text{ otherwise.} \end{cases}$$

For $i = 2, \ldots, m_2$, the optimal solution is:

$$y_i = \begin{cases} \frac{1}{\mathcal{L}_{i,i}}((h - Tx)_i - \sum_{k=1}^{i-1}(\mathcal{L}_{i,k}y_k + \mathcal{R}_{i,k}y_{m_2+k})), \\ \quad \text{if } (h - Tx)_i \geq \sum_{k=1}^{i-1}(\mathcal{L}_{i,k}y_k + \mathcal{R}_{i,k}y_{m_2+k}) \\ 0, \text{ otherwise,} \end{cases}$$

$$y_{m_2+i} = \begin{cases} \frac{1}{\mathcal{R}_{i,i}}((h - Tx)_i - \sum_{k=1}^{i-1}(\mathcal{L}_{i,k}y_k + \mathcal{R}_{i,k}y_{m_2+k})), \\ \quad \text{if } (h - Tx)_i < \sum_{k=1}^{i-1}(\mathcal{L}_{i,k}y_k + \mathcal{R}_{i,k}y_{m_2+k}) \\ 0, \text{ otherwise.} \end{cases}$$

For simplicity, in the rest of this section we drop the superscript index k denoting the cell number. When x is fixed, consider values of ξ such that $L \leq h - Tx \leq U$. The upper and lower bounds for the optimal values of y are denoted y^{UB} and y^{LB} respectively. These bounds are the solutions to the corresponding BNPs when maximizing and minimizing each primal decision variable. We use the notation $[\cdot]^+ \equiv \max\{0, \cdot\}$. The bounds are:

$$y_1^{UB} = \begin{cases} \frac{U_1}{\mathcal{L}_{1,1}}, \text{ if } U_1 > 0 \\ 0, \text{ otherwise.} \end{cases}$$

$$y_1^{LB} = \begin{cases} \frac{L_1}{\mathcal{L}_{1,1}}, \text{ if } L_1 > 0 \\ 0, \text{ otherwise.} \end{cases}$$

$$y_{m_2+1}^{UB} = \begin{cases} \frac{L_1}{\mathcal{R}_{1,1}}, \text{ if } L_1 < 0 \\ 0, \text{ otherwise.} \end{cases}$$

$$y_{m_2+1}^{LB} = \begin{cases} \frac{U_1}{\mathcal{R}_{1,1}}, \text{ if } U_1 < 0 \\ 0, \text{ otherwise.} \end{cases}$$

For $i = 2, \ldots, m_2$, the bounds are:

$$y_i^{UB} = \begin{cases} \frac{1}{\mathcal{L}_{i,i}}(U_i - \phi_i^{LB}), \text{ if } U_i > \phi_i^{LB} \\ 0, \text{ otherwise,} \end{cases}$$

$$y_i^{LB} = \begin{cases} \frac{1}{\mathcal{L}_{i,i}}(L_i - \phi_i^{UB}), \text{ if } L_i > \phi_i^{UB} \\ 0, \text{ otherwise,} \end{cases}$$

$$y_{m_2+i}^{UB} = \begin{cases} \frac{1}{\mathcal{R}_{i,i}}(L_i - \phi_i^{UB}), \text{ if } L_i < \phi_i^{UB} \\ 0, \text{ otherwise,} \end{cases}$$

$$y_{m_2+i}^{LB} = \begin{cases} \frac{1}{\mathcal{R}_{i,i}}(U_i - \phi_i^{LB}), \text{ if } U_i < \phi_i^{LB} \\ 0, \text{ otherwise,} \end{cases}$$

where we define:

$$\phi_i^{UB} = \sum_{k=1}^{i-1} \left([\mathcal{L}_{i,k}]^+ y_k^{UB} - [-\mathcal{L}_{i,k}]^+ y_k^{LB} + [\mathcal{R}_{i,k}]^+ y_{m_2+k}^{UB} - [-\mathcal{R}_{i,k}]^+ y_{m_2+k}^{LB} \right)$$

$$\phi_i^{LB} = \sum_{k=1}^{i-1} \left([\mathcal{L}_{i,k}]^+ y_k^{LB} - [-\mathcal{L}_{i,k}]^+ y_k^{UB} + [\mathcal{R}_{i,k}]^+ y_{m_2+k}^{LB} - [-\mathcal{R}_{i,k}]^+ y_{m_2+k}^{UB} \right).$$

Using the bounds for the optimal primal decision variables, we construct upper and lower bounds π^{UB} and π^{LB}, for optimal dual recourse variables. This is done using backwards recursion and the complementary slackness conditions. The bounds are:

$$\pi_{m_2}^{UB} = \begin{cases} \frac{q_{2m_2}}{\mathcal{R}_{m_2,m_2}} \text{ if } y_{2m_2}^{LB} > 0 \\ \frac{q_{m_2}}{\mathcal{L}_{m_2,m_2}}, \text{ otherwise,} \end{cases}$$

$$\pi_{m_2}^{LB} = \begin{cases} \frac{q_{m_2}}{\mathcal{L}_{m_2,m_2}}, \text{ if } y_{m_2}^{LB} > 0 \\ \frac{q_{2m_2}}{\mathcal{R}_{m_2,m_2}} \text{ otherwise.} \end{cases}$$

For $i = 1, \ldots, m_2 - 1$, the bounds are:

$$\pi_i^{UB} = \begin{cases} \frac{1}{\mathcal{R}_{i,i}}(q_{m_2+i} - \psi_i^{UB}), \text{ if } y_{m_2+i}^{LB} > 0 \\ \frac{1}{\mathcal{L}_{i,i}}(q_i - \theta_i^{LB}), \text{ otherwise,} \end{cases}$$

$$\pi_i^{LB} = \begin{cases} \frac{1}{\mathcal{L}_{i,i}}(q_i - \theta_i^{UB}), \text{ if } y_i^{LB} > 0 \\ \frac{1}{\mathcal{R}_{i,i}}(q_{m_2+i} - \psi_i^{LB}), \text{ otherwise,} \end{cases}$$

where we define:

$$\theta_i^{UB} = \sum_{k=i+1}^{m_2} \left([\mathcal{L}_{k,i}]^+ \pi_k^{UB} - [-\mathcal{L}_{k,i}]^+ \pi_k^{LB} \right)$$

$$\theta_i^{LB} = \sum_{k=i+1}^{m_2} \left([\mathcal{L}_{k,i}]^+ \pi_k^{LB} - [-\mathcal{L}_{k,i}]^+ \pi_k^{UB} \right)$$

$$\psi_i^{UB} = \sum_{k=i+1}^{m_2} \left([\mathcal{R}_{k,i}]^+ \pi_k^{UB} - [-\mathcal{R}_{k,i}]^+ \pi_k^{LB} \right)$$

$$\psi_i^{LB} = \sum_{k=i+1}^{m_2} \left([\mathcal{R}_{k,i}]^+ \pi_k^{LB} - [-\mathcal{R}_{k,i}]^+ \pi_k^{UB} \right)$$

3.6 Computational Experiments

In this section we describe a model for an important semiconductor manufacturing application that we use as the basis for computational experiments. We show how the restricted recourse approach described in Section 3.4 is integrated with sequential bounding for this problem. We present the results of computational experiments for sequential bounding using the various algorithms described in Section 3.5 as well as restricted recourse and optimized outcomes described in Section 3.4.

3.6.1 Inventory Planning with Downward Substitutions

The problem of inventory planning with downward substitutions involves a decision maker determining production quantities for a family of products in the first stage. In the second stage of the problem, substitutions can be made after the random demand for each product is known. Any product can be substituted with another having a lower numbered index but not vice versa. This is often the case in the semiconductor industry, for example, when faster microprocessors can be substituted for slower processors. The goal of the decision maker is to minimize production costs as well as the expected penalty and holding costs for not meeting demand and having excess inventory after demand is met. Following Bassok et al. (1999) and Rao et al. (2004), we describe the stochastic programming formulation for this problem for N products. We begin with a description of the first stage decision variables:

$z_i \equiv 1$ if product i is produced, 0 otherwise, for $i = 1, \ldots, N$;

$x_i \equiv$ the production quantity for product i, for $i = 1, \ldots, N$.

We let $\xi_j(\omega)$ denote the random demand for product $j = 1, \ldots, N$. The second stage decision variables are:

$w_{i,j}(\omega) \equiv$ the quantity of product i that will be used as a substitute for product j, where $i \leq j$ and $i, j = 1, \ldots, N$;

$u_j^-(\omega) \equiv$ The quantity of unmet demand for product $j = 1, \ldots, N$;

$u_j^+(\omega) \equiv$ The quantity of excess inventory for product $j = 1, \ldots, N$.

Next, we describe the cost parameters:

$K_i \equiv$ the setup cost for product $i = 1, \ldots, N$;

$c_i \equiv$ the unit production cost for product $i = 1, \ldots, N$;

$h_i \equiv$ the unit holding cost for excess inventory after demand is met for product $i = 1, \ldots, N$;

$p_i \equiv$ the unit penality cost for not meeting demand for product $i = 1, \ldots, N$;

$s_{i,j} \equiv$ the unit cost of substituting product i for product j when $i \leq j$ and $i, j = 1, \ldots, N$.

The complete two-stage stochastic programming formulation is as follows:

$$\begin{aligned}
\min \quad & \textstyle\sum_{i=1}^{N}(c_i x_i + K_i z_i) + E_\xi \left[\sum_{i=1}^{N}(h_i u_i^+(\omega) + p_i u_i^-(\omega)) + \sum_{\substack{i,j=1 \\ i\le j}}^{N} s_{i,j} w_{i,j}(\omega)\right] \\
\text{s.t.} \quad & x_i \le M z_i, \text{ for } i = 1,\ldots,N; \omega \in \Omega \qquad (3.17) \\
& \textstyle\sum_{i=1}^{j} w_{i,j}(\omega) + u_j^-(\omega) = \xi_j(\omega), \text{ for } j = 1,\ldots,N; \omega \in \Omega \qquad (3.18) \\
& \textstyle\sum_{j=i}^{N} w_{i,j}(\omega) + u_i^+(\omega) = x_i, \text{ for } i = 1,\ldots,N; \omega \in \Omega \qquad (3.19) \\
& x_i, w_{i,j}, u_i^+(\omega), u_j^-(\omega) \ge 0; z_i \in \{0,1\} \text{ for } i,j = 1,\ldots,N; i \le j \text{ and } \omega \in \Omega
\end{aligned}$$

Here, M is a sufficiently large quantity to ensure that Equation 3.17 is a redundant constraint when $z_i = 1$. Equation 3.18 is the set of demand balance constraints, and Equation 3.19 is the set of supply balance constraints.

3.6.2 Implementation of Restricted Recourse Bounds

Proposition 2 below was used to develop a restricted recourse problem to apply the bound of Section 3.4.1, providing an alternative upper bound for the sequential bounding algorithm.

Proposition 2. *Let $Q_N(x,\xi)$ be the following recourse linear program for the inventory planning problem with downward substitutions and N products:*

$$\begin{aligned}
\min \quad & \textstyle\sum_{i=1}^{N}(h_i u_i^+ + p_i u_i^-) + \sum_{\substack{i,j=1 \\ i\le j}}^{N} s_{i,j} w_{i,j} \\
\textit{s.t.} \quad & \textstyle\sum_{i=1}^{j} w_{i,j} + u_j^- = \xi_j, \textit{ for } j = 1,\ldots,N \\
& \textstyle\sum_{j=i}^{N} w_{i,j} + u_i^+ = x_i, \textit{ for } i = 1,\ldots,N \\
& w_{i,j}, u_i^+, u_j^- \ge 0 \textit{ for } i,j = 1,\ldots,N \textit{ and } i \le j
\end{aligned}$$

Let ℓ be a fixed integer such that $1 \le \ell < N$. If $w_{i,j} = 0$ for all $i \le \ell$ and $j > \ell$, then the recourse problem for N products decomposes into one recourse problem with ℓ products and another with $N - \ell$ products: $Q_N(x,\xi) = Q_\ell(x, [\xi_1,\ldots,\xi_\ell]^\intercal) + Q_{N-\ell}(x, [\xi_{\ell+1},\ldots,\xi_N]^\intercal)$.

Proof. The hypotheses of this proposition imply that no product with index less than or equal

to ℓ will ever be substituted for a product with index greater than ℓ. After eliminating all $w_{i,j}$ decision variables that are known to be equal to zero, every constraint either involves decision variables with indices less than or equal to ℓ or indices greater than ℓ, but not both. The recourse linear program therefore separates into two linear programs. □

At iteration ν of sequential bounding, suppose there exists a $k \leq N$ and ℓ such that $1 \leq \ell < N$ and $w_{i,j}^{k,\nu*} = 0$ for all $i \leq \ell$ and $j > \ell$ in the PMVP optimal solution, then we temporarily added the following constraints to the recourse problem:

$$\widetilde{W}_{i,j}(\omega) w_{i,j} = 0, \text{ for all } i \leq \ell, j > \ell$$

where

$$\widetilde{W}_{i,j}(\omega) = \begin{cases} 1, & \text{if } \xi(\omega) \in S^k, i \leq \ell, j > \ell \\ 0, & \text{otherwise.} \end{cases}$$

Note that there may be more than one such value of ℓ for a given value of k where these constraints can be added. In those cases, we can further decompose one of the smaller recourse problems into two even smaller recourse problems. We repeat this process as many times as possible.

After adding these constraints to $Q(x, \xi)$, we obtained a valid restricted recourse problem, as described in Section 3.4.1, and we can use the bound of Corollary 2 instead of the bound of Corollary 1 in the sequential bounding routine. The upper bound of Corollary 2 requires bounds on the optimal dual decision variables of the restricted recourse problem. From Proposition 2, optimal decision variables of the restricted recourse problem are equivalent to decision variables for smaller instances of the unrestricted recourse problem. We used the linear program relaxation approach of Al-Khayyal and Falk (1983), described in section 3.5.2, for solving the BNPs to obtain optimal dual decision variable bounds $\widetilde{\pi}^{k,UB}$ and $\widetilde{\pi}^{k,LB}$ for unrestricted recourse problems with less than N products. These bounds are often tighter than those found for the unrestricted recourse problem with N products, the decision variable bounds used in Corollary 1. We always

used the smaller upper bound from Corollary 1 and 2 for each iteration of the sequential bounding approach. Note that we remove these added constraints prior to the next iteration of the algorithm.

3.6.3 Computational Experiments

We used SAA, a Monte Carlo sampling-based approach, for comparison with sequential bounding. A description of SAA is provided by Santoso et al. (2005). SAA generates a number of stochastic programs, where demand outcomes are randomly sampled and given equal probability of occurrence. These new stochastic programs are easily solved using linear programming. The optimal objective function values of these new stochastic programs are used to generate a statistical lower bound on the optimal objective function value of the original stochastic program. After a feasible first stage solution is selected, SAA repeatedly generates a random demand outcome and solves the recourse linear program for the first stage solution and demand outcome. The optimal objective function values of these recourse linear programs are used to generate a statistical upper bound on the optimal objective function for the original stochastic program.

There are three parameters that must be specified for SAA: the number of scenarios for each lower bounding problem, the number of lower bounding problems, and the number of demand outcomes used to generate the statistical upper bound. We solved a new sampled instance of the lower bounding problem at each iteration until the time limit of 1800 seconds was reached. Whenever an instance of the lower bounding problem yielded an optimal objective function value that was lower than any found in previous iterations, we used that solution as the best solution found so far, and solved for a statistical estimate of an upper bound on the optimality gap. Following Santoso et al. (2005), we selected 20 scenarios for the lower bounding problem and 1000 demand outcomes for the upper bounding problem. We also considered 100 scenarios for the lower bounding problem in our computational experiments. No other attempts were made to optimize these algorithm control parameters. We did not use Benders decomposition

to solve each of the sampled lower bound problems as was done by Santoso et al. (2005). We also avoided using Benders decomposition for each iteration of the sequential bounding algorithm. It is not clear if either of the two algorithms would benefit more than the other if decomposition was used.

The sequential bounding algorithms, as well as the SAA algorithm, were implemented using the C++ programming language and a commerical mixed integer programming solver, the IBM ILOG CPLEX 12.4 callable library. The test cases were run on a 64-bit Windows machine with an 8 core (2.93 GHz) Intel Core i7 processor.

An illustrative example of the partition of the random variable support set is provided in Figure 3.2. Each of the four diagrams correspond to the partition at iterations 1, 2, 3, and 11 of the LP-relaxation version of the sequential bounding algorithm. A two-product instance was used to generate the figure. The demand for each product is a continuous uniform random variable taking values between 0 and 100. For each partition, every cell has the conditional mean values labelled with a black dot. We also include the difference in the bounds on the first two dual decision variables, $\Delta\pi_1$ and $\Delta\pi_2$, when the random variables are restricted to each cell of the partition. In Figure 3.3 the upper and lower bounds on the optimal objective function value are plotted at each iteration of the sequential bounding algorithm. The difference in these bounds is the optimality gap for the two-stage stochastic program.

A total of 24 test cases were used in the computational experiments. The instance parameters were based on those considered by Rao et al. (2004). In that paper, a large number of test problems were considered. The test problems include gamma and correlated-normal distributed demand. We did not consider these two particular types of demand, but these cases can be handled with sequential bounding. In particular, correlated-normal random variables are equal to an affine transformation of uncorrelated random variables; so, integrals appearing in Proposition 1 for ϵ_1 and ϵ_2 simplify after a change of variables. Three different numbers of products were used: 5, 10, and 25 with normally (N), uniformly (U), and lognormally (LN) distributed demand. We added lognormally distributed demand to include a non-symmetric distribution.

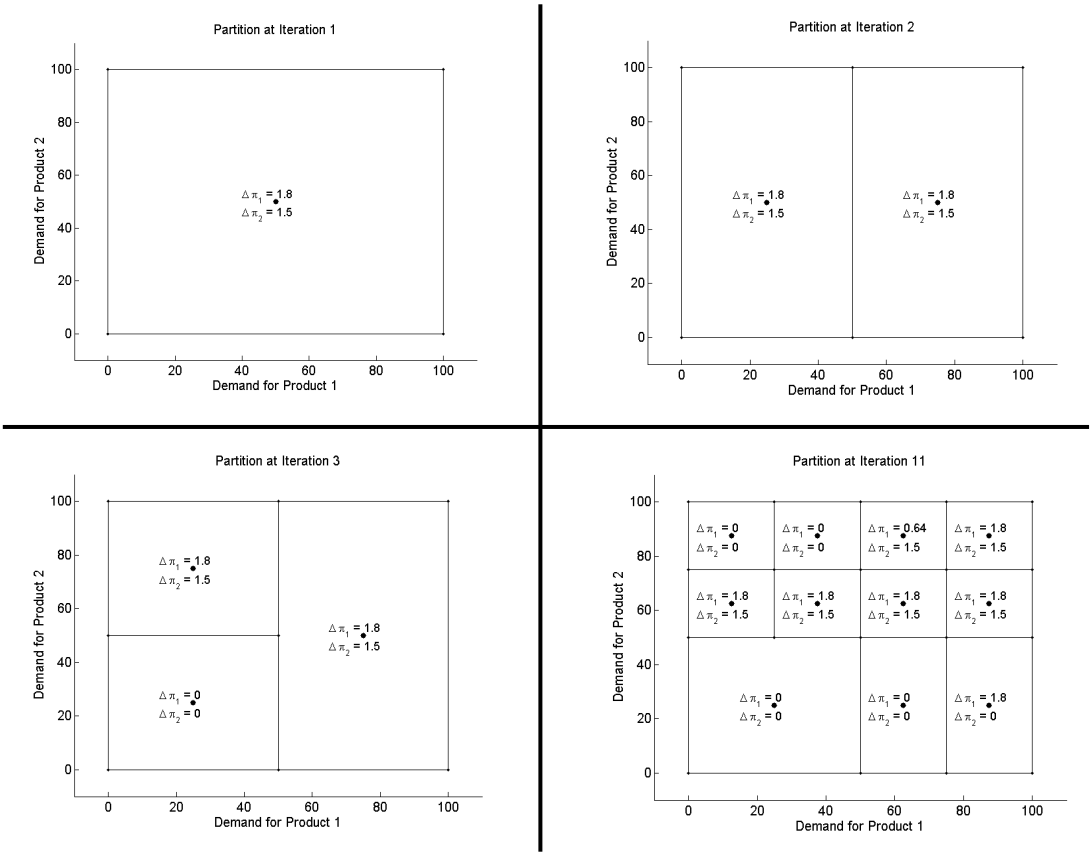

Figure 3.2: A 2-dimensional example of the partition of the random variable support set at iterations 1, 2, 3, and 11 of the LP-relaxation version of the sequential bounding algorithm. Conditional mean values for each cell are labelled with a black dot. The quantities $\Delta\pi_1$ and $\Delta\pi_2$ are the differences in the bounds on the first two dual decision variables when the random variables are restricted to each cell.

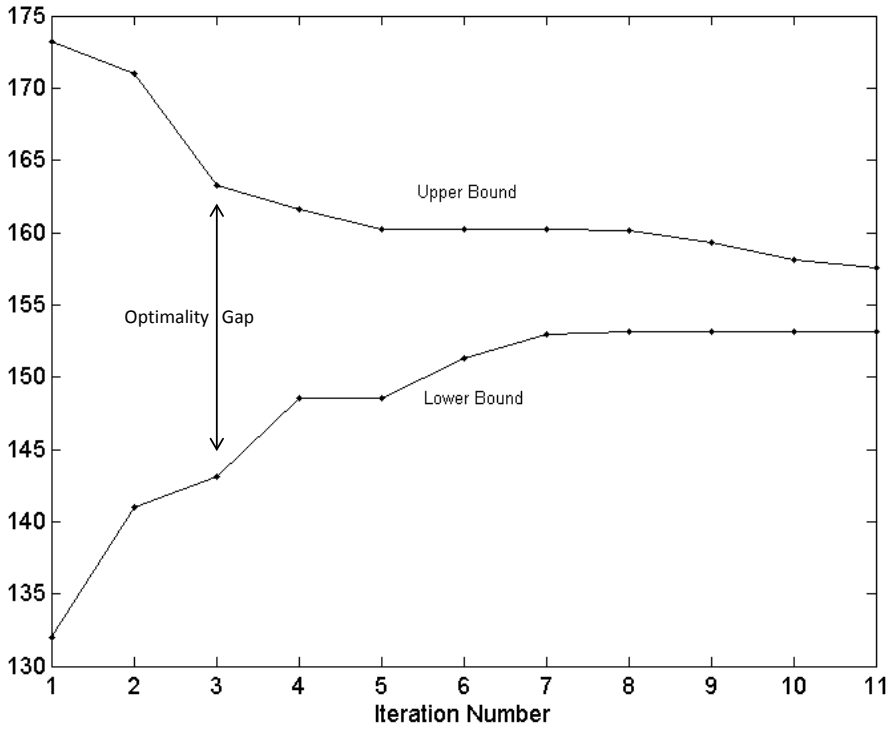

Figure 3.3: Upper and lower bounds on the optimal objective function value vs. iteration number for the LP-relaxation version of the sequential bounding algorithm for a two product instance of the inventory problem with downward substitutions. The difference between the bounds is the optimality gap.

Cost parameters associated with the most and least expensive products considered by Rao et al. (2004) were used in the tests. The mean and standard deviations for both the normal and log-normal demand are 100 and 10 respectively. Demand uniformly distributed over the intervals $[10, 100]$ and $[10, 1000]$ were also considered.

We began the computational experiments with a modified version of the inventory planning problem. First, we prevented all product substitutions and reduced all the fixed costs to zero. In other words, $w_{i,j} = 0$ and $K_i = 0$ for all i and j. Next, we added a capacity constraint to the first stage:

$$\sum_{i=1}^{N} x_i \leq \sum_{i=1}^{N} E[\xi_i]$$

Computational results for this modified problem are shown in Figure 3.4, comparing sequential bounding with the various algorithms described in Section 3.5 as well as the restricted recourse approach of Section 3.4.1. All 24 test cases were run until the last iteration finished after 900 seconds. From the figure, we see that restricted recourse consistently has the lowest percent error. We also observe that the percent error increases with increasing demand variance and number of products.

Computational results for the modified inventory planning problem using the method of optimized outcomes described in Section 3.4.2 are shown in Figure 3.5. A mixed integer program was solved to obtain an upper bound after sequential bounding completed. The additional time required was less than a minute for all test cases. We present results for optimized outcomes for two sequential bounding algorithms: one using global bounds on the optimal dual decision variables and the other using the restricted recourse approach. We see that optimized outcomes with restricted recourse tends to outperform the approach with global bounds. For the unmodified inventory planning problem described below, we give an explanation for why the effect of the decision variable bounding approaches diminishes as the number of products increase, and we feel this explanation applies here as well. We also point out that for the two 25-product cases where demand is uniformly distributed between 10 and 100, optimized outcomes with global bounds slightly outperforms the approach with restricted recourse bounds. This appears

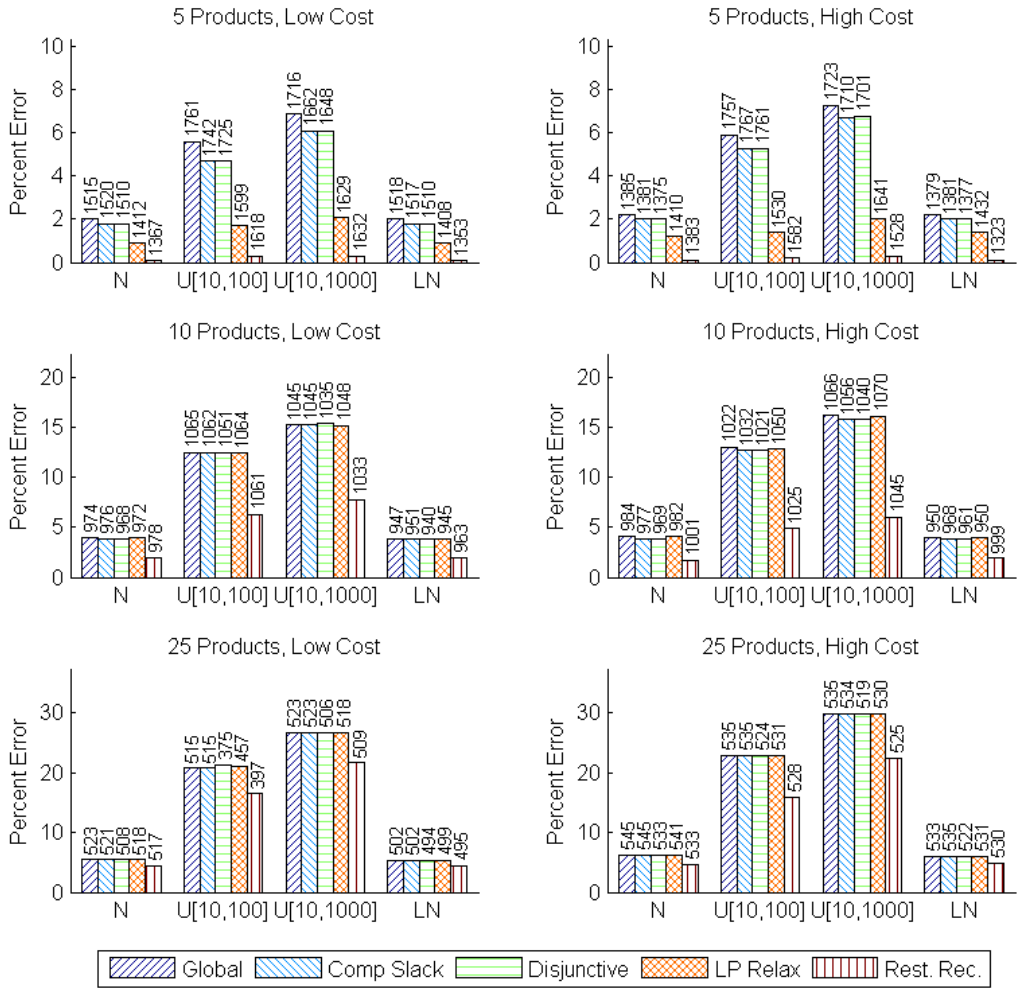

Figure 3.4: Experimental results for the modified inventory planning problem, reported as percent error. Results are shown for each of 5 sequential bounding algorithms, 4 demand distributions, 3 numbers of products, and 2 cost structures. The number of iterations are shown above each bar.

to be due to the extra time required to generate restricted recourse bounds. This means that fewer iterations of sequential bounding are completed, and the percent error is not reduced as much. How much of a reduction in the percent error occurs at each iteration is difficult to say in advance, but the reduction tends to be largest in the earlier iterations, and these two 25-product cases stopped after fewer iterations than most of the other test cases. The lognormally distributed demand cases ended after fewer iterations than when demand was uniform on $[10, 100]$, but there is less of a difference in the number of iterations for those cases between optimized outcomes with global and restricted recourse bounds.

The computational results for the inventory planning problem with downward substitutions are shown in Figure 3.6 for the algorithms described in Section 3.5 as well as the restricted recourse approach of Section 3.4.1. The results are reported as the percent error, relative to the lower bound, after the last iteration. The total number of iterations appears above each bar in the figure. Results for the method of optimized outcomes, described in Section 3.4.2, are presented in Figure 3.7. This approach could be used to calculate an upper bound at each iteration of sequential bounding. However, we solved the mixed integer linear program upon completion of sequential bounding. We present results for optimized outcomes for two sequential bounding algorithms: one using global bounds on the optimal dual decision variables and the other using the restricted recourse approach. The optimal first stage decision variables found from solving the upper bounding problem, rather than the first stage solution from sequential bounding, were used in comparisons with the SAA algorithm. We used 9 evenly spaced points to create a piecewise linear outer approximation of each convex function appearing in the objective function of the mathematical program in Section 3.4.2. The run times for these results include 900 seconds of run time for sequential bounding plus the time required for solving the upper bounding problem. The maximum of the two run times for each test case was used as the run time for all the algorithms in Figure 3.6. All of these run times, as well as the run times for SAA, are shown in Figure 3.8.

The time required to solve the upper bounding problem was particularly long for the case

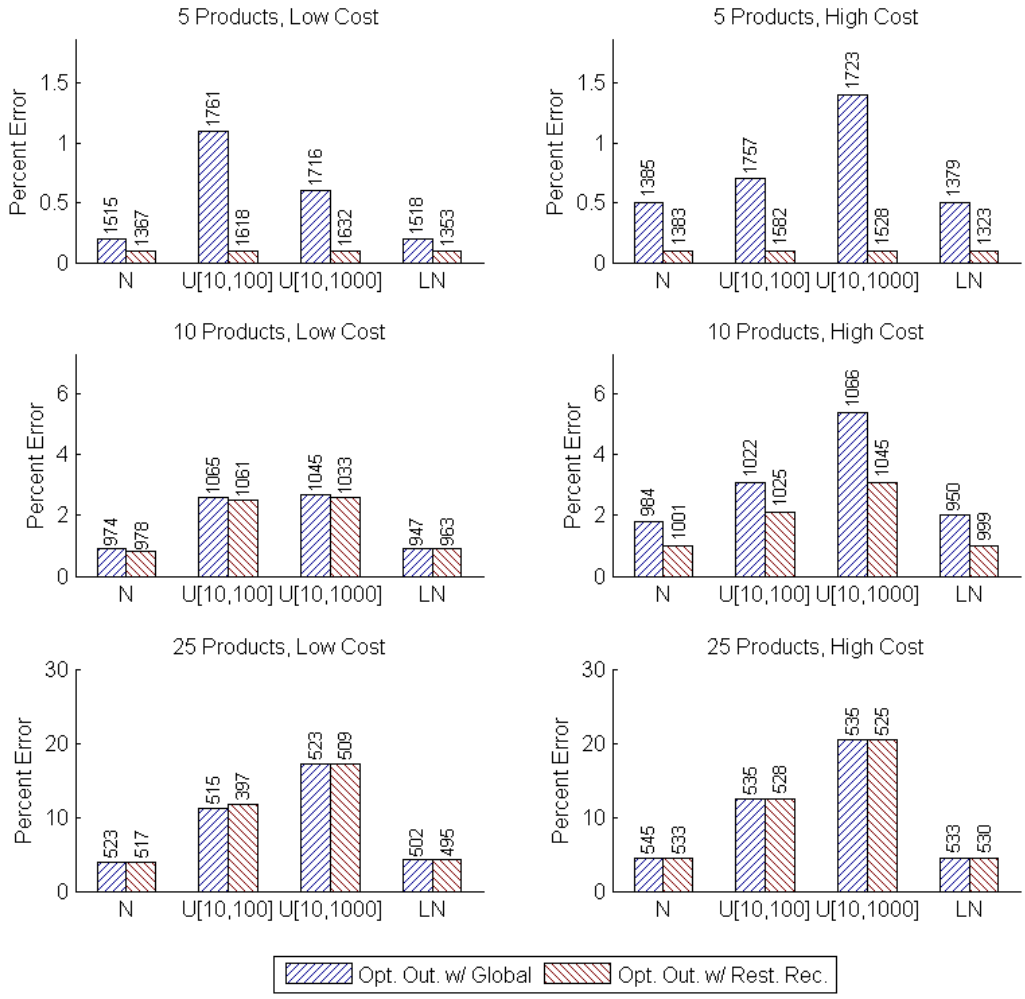

Figure 3.5: Experimental results for sequential bounding with optimized outcomes for the modified inventory planning problem, reported as percent error. Results are shown for each of 2 sequential bounding algorithms with optimized outcomes, 4 demand distributions, 3 numbers of products, and 2 cost structures. The number of iterations are shown above each bar.

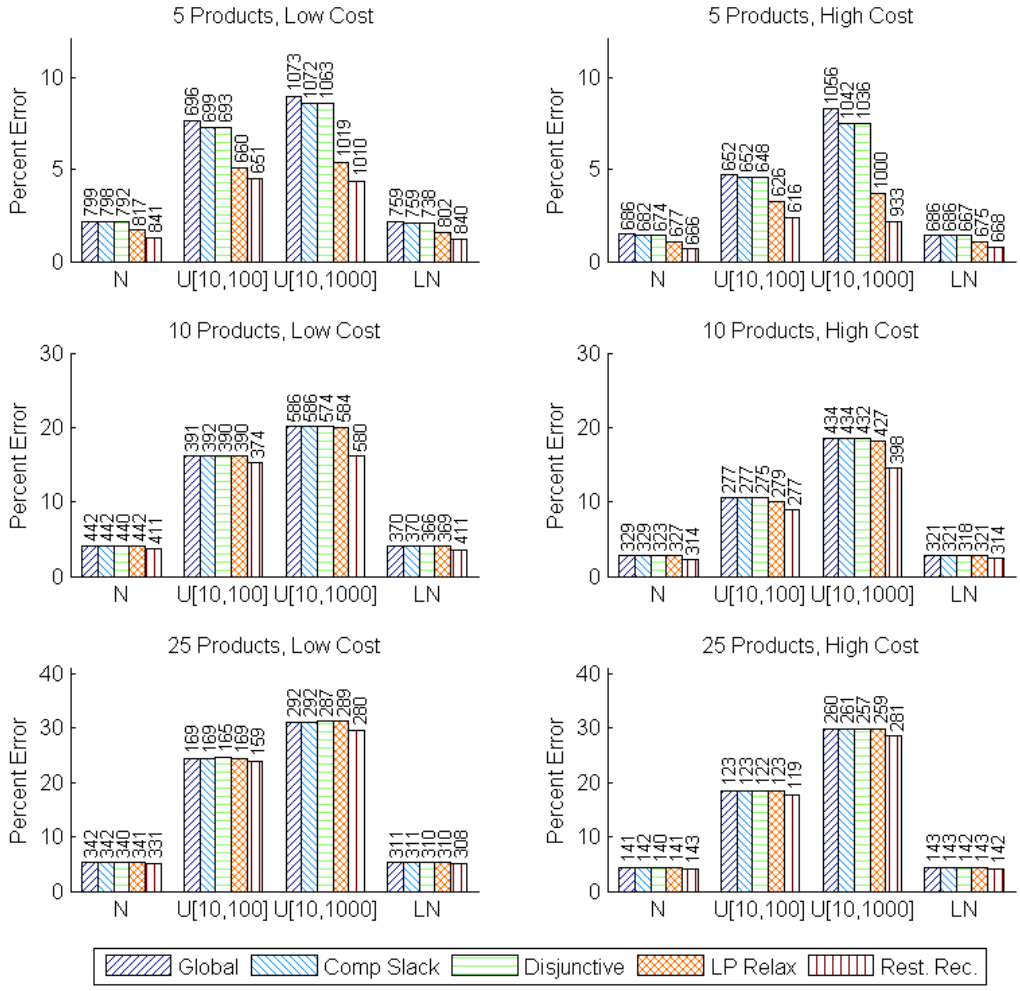

Figure 3.6: Experimental results for sequential bounding, reported as percent error. Results are shown for each of 5 sequential bounding algorithms, 4 demand distributions, 3 numbers of products, and 2 cost structures. The number of iterations are shown above each bar.

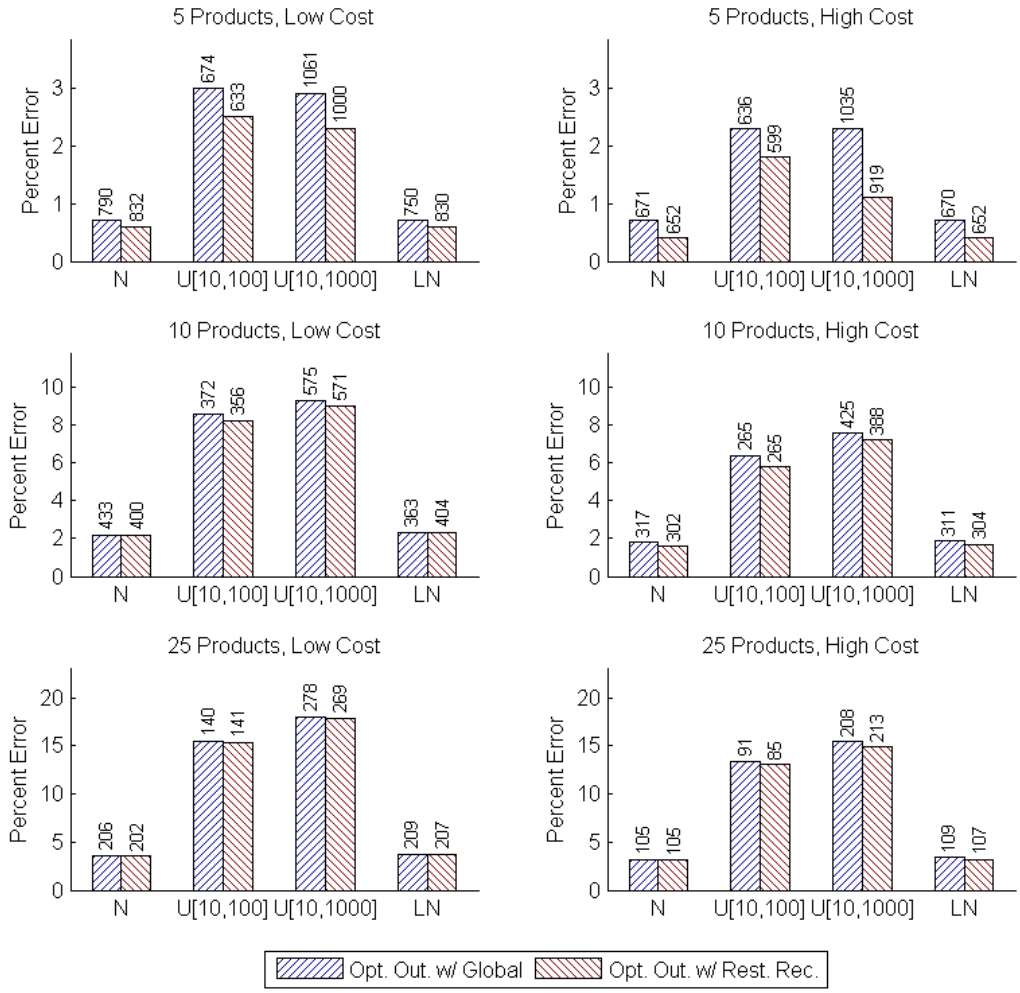

Figure 3.7: Experimental results for sequential bounding with optimized outcomes, reported as percent error. Results are shown for each of 2 sequential bounding algorithms with optimized outcomes, 4 demand distributions, 3 numbers of products, and 2 cost structures. The number of iterations are shown above each bar.

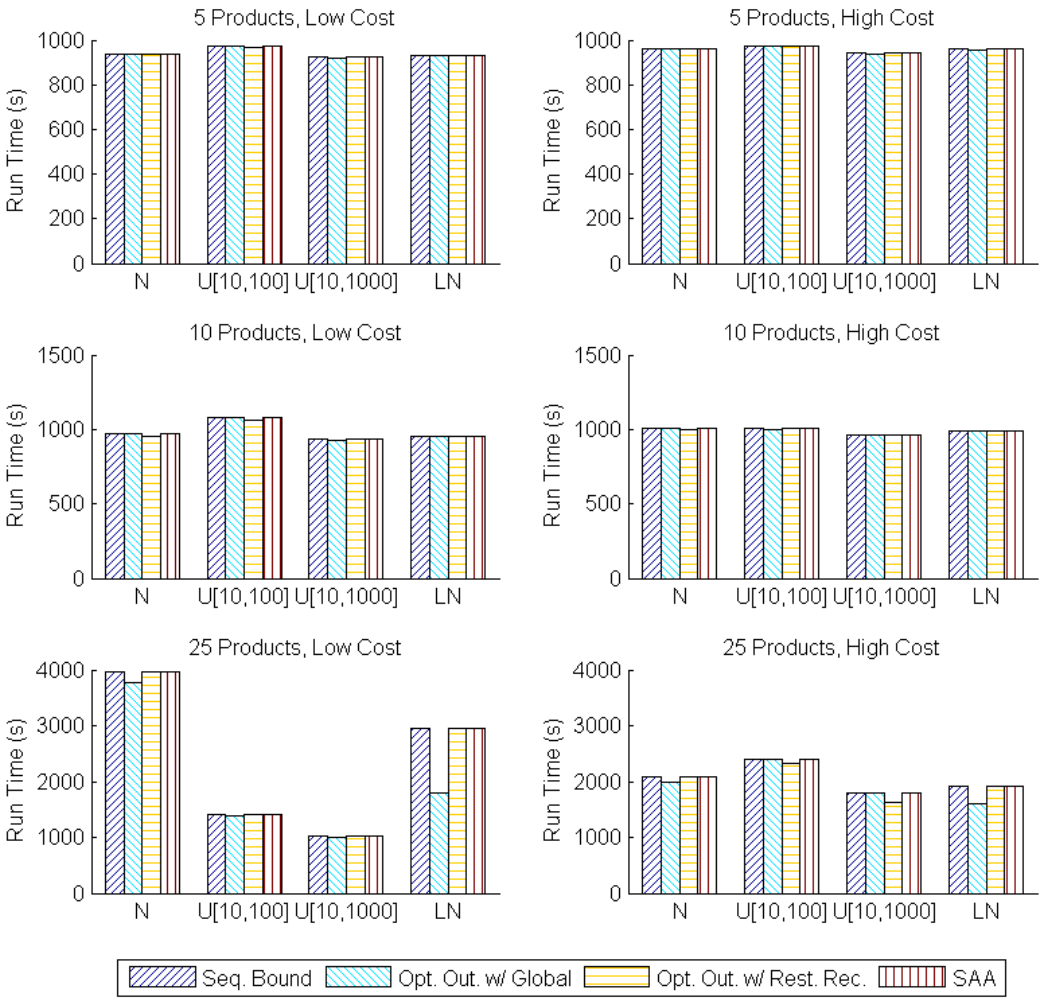

Figure 3.8: Run time in seconds for the sequential bounding algorithms, both variations of optimized outcomes, and the SAA algorithms. Run times are reported for each of 4 demand distributions, 3 numbers of products, and 2 cost structures.

with 25 products, low costs, and low demand variance. This is most likely due to the size of the resulting mixed integer program and the fact that low costs and demand variance result in smaller total cost differences between various first stage production setup configurations. Thus, more branch and bound iterations are required to solve the upper bounding problem for this case. This explains the relatively long run times for the last row in Figure 3.8. We also see that higher demand variance for the 25 product cases can result in shorter run times for the upper bounding problem when the costs are lower. So, lower costs make the total cost more sensitive to variations in demand, which improves the overall run time of branch and bound for solving the upper bounding problem.

Results for the SAA are shown in Figure 3.9. We report the estimated percent error as the width of the 99% confidence intervals for the optimality gap divided by the estimated lower bound on the optimal solution objective value. The number of iterations for the SAA algorithm is the total number of times that the lower bounding problem was solved. These are reported above each of the bars in the figure. By the central limit theorem, the accuracy of these confidence bounds is asymptotically valid as the number of iterations and sample size for the upper bound problem approaches infinity. There are test cases where the number of iterations is less than 30, such as those shown in the bottom right chart in Figure 3.9. So, the estimated percent error reported for those cases should not be considered accurate. The first stage solution associated with the lowest optimal objective value of all lower bounding problems is returned upon termination of the SAA. Similar to what was observed in Figures 3.6 and 3.7, we see that the estimated percent error increases for increasing demand variance in Figure 3.9.

The percent error for the sequential bounding algorithms in Figure 3.6 are surprisingly similar. The difference in percent error between algorithms for each case is never more than 6.1 percentage points, but the restricted recourse approach of Section 3.4.1 yields the lowest percent error in every case. The difference in percent error between the algorithms in the table diminishes as the number of products, N, increases. Preliminary experiments with very small instances led us to to develop the methods described in Section 3.5. For example, analysis of

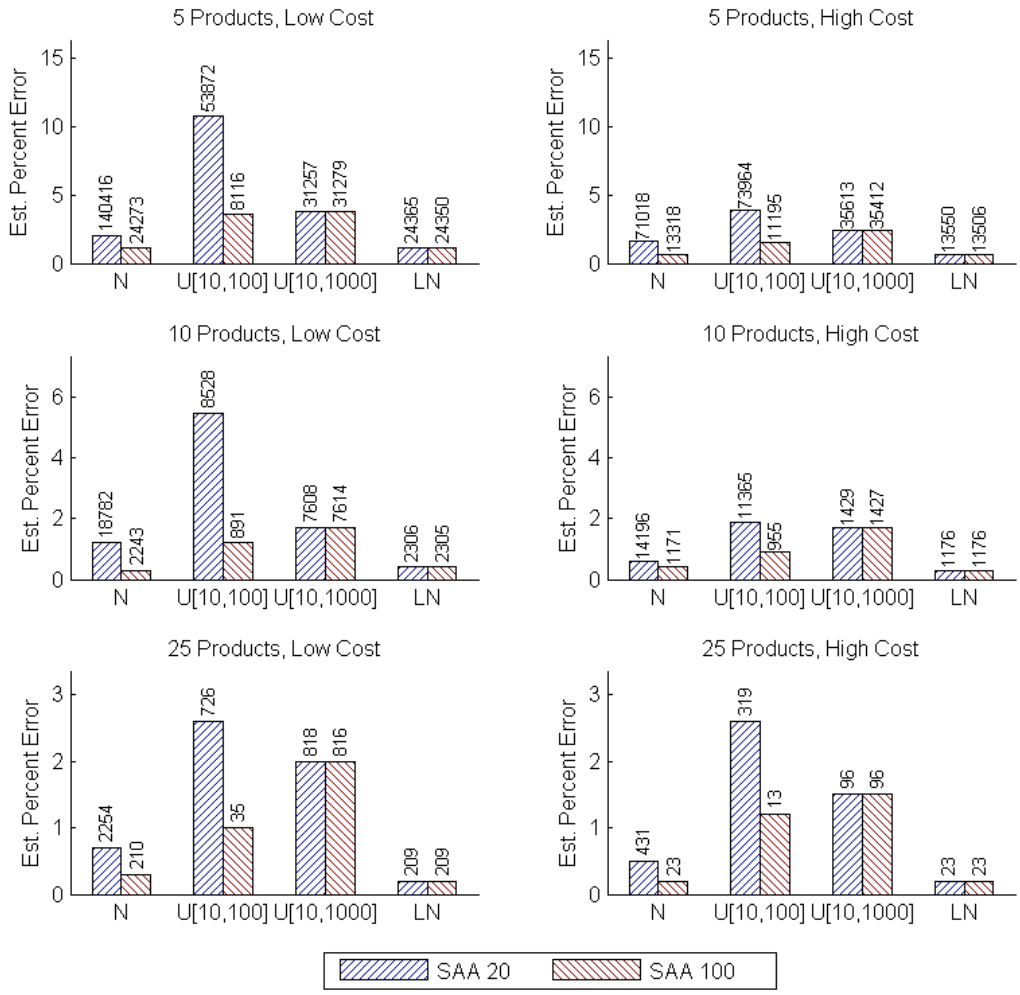

Figure 3.9: Experimental results for SAA, reported as estimated percent error. Results are shown for 20 and 100 scenarios for the lower bounding problem, 4 demand distributions, 3 numbers of products, and 2 cost structures. The number of iterations are shown above each bar.

the instance with two random variables depicted in Figure 3.1 led to the development of the disjunctive bounding method described in Section 3.5.4. Our success with small instances of this problem did not translate to success for larger instances.

We offer an explanation for the lack of variation in percent error between the different algorithms for instances with higher numbers of products in Figure 3.6. Global decision variable bounds are not affected by restrictions on the set of possible random variable outcomes, but the decision variable bounding methods of Section 3.5 and 3.4.1 depend on such restrictions. For this problem, the extent to which decision variable bounds can be improved beyond the global bounds appears to depend on most or all of the random variables being restricted to a set of outcomes that is a proper subset of their support sets. Unfortunately, as the number of products increases, the number of iterations required for the cells to have such restrictions grows exponentially. For example, in order to obtain N cells with every random variable restricted to a proper subset of its support set, we must perform 2^N iterations. This is achieved by splitting every cell with respect to one of the random variables, then splitting each of those cells with respect to the next random variable, etc. This suggests that sequential bounding will probably be most successful, in combination with methods for improving decision variable bounds, for problems where the bounds are only influenced by a small subset of all the random variables and their range of possible outcomes.

Next we compare Figures 3.6 and 3.7. The optimized outcome approach of section 3.4.2 outperforms all the other sequential bounding algorithms, with the exception of the case where demand is uniformly distributed over $[10, 1000]$, with 5 products and high costs. For every case, optimized outcomes with the restricted recourse algorithm provides the lowest percent error of all the approaches represented in both tables. The difference between the percent error for the two optimized outcome algorithms tends to be smaller for larger numbers of products. This suggests that for larger numbers of products, the improvements from optimized outcomes is overwhelmingly more important than the approaches presented in Section 3.5 and Section 3.4.1.

We provide an explanation for the relative success of the optimized outcomes approach over those described in Section 3.5 and 3.4.1, which focus on improving decision variable bounds. These other approaches always divide the support of the random variables into cells and use the conditional mean values of the random variables for each cell as representative outcomes. This particular approach for approximating continuous distribution functions with discrete distributions was shown by Miller and Rice (1983) to underestimate the variance of the underlying continuous distribution. In contrast to these methods, the optimized outcomes approach seeks to find representative outcomes for each cell that minimizes a particular upper bound on the optimal objective function value. So, the variance of representative outcomes can be larger and closer to the true variance. This explains why the relative success of optimized outcomes is greatest for the test cases with higher variance in the demand distribution.

We also feel that the results in Figures 3.6 and 3.7 provide insight into the inner workings of the sequential bounding algorithm. We feel this insight may prove useful in future efforts to improve the performance of the algorithm and may help practitioners understand which applications are more likely to be successful. In particular, notice that higher costs result in a lower percent error when the demand distribution and number of products is fixed. This appears to be due to higher set up costs for producing products in the first stage, making it more attractive to set up for fewer products, but at a higher expected cost of substituting products in the second stage. Solutions with fewer setups are more robust because inventory is higher for the products that are produced. These products also tend to be substitutable. Inventory for such products can be used to guard against unexpectedly high demand for the product as well as demand for higher indexed products. The risk of not meeting customer demand for a low indexed product, but having excess inventory for a higher indexed product, is therefore reduced. Because of this risk pooling effect, when we decide to produce product i, but not product $i+1$, scenarios where product i has high demand and product $i+1$ has low demand can be combined with those where demand is high for product $i+1$ and low for product i. So, potentially fewer representative scenarios would be needed to obtain good solutions if this

effect can be exploited when partitioning the support of the random variables.

Of the two SAA algorithms, the one with 100 scenarios for the lower bounding problem has the lowest estimated percent error for every test case. In comparing SAA to sequential bounding below, we selected the 100 scenario and optimized outcomes with restricted recourse approaches respectively.

Bounds on the solution found at the final iteration of both algorithms are reported in Figure 3.10. These numbers were generated using restricted recourse with optimized outcomes and a fixed first stage solution. Recall that deterministic bounds on the recourse problem for a fixed first stage solution were given in Proposition 1. We compare the solution from solving the linear program described in Section 3.4.2 after sequential bounding with restricted recourse to the solution from SAA with 100 scenarios for the lower bounding problem. The optimized outcomes approach produces a lower cost solution relative to the SAA for the cases with normally and lognormally distributed demand, high costs, and 25 products. In all the other cases, the bounds overlap. So, we cannot say which algorithm produces the best solution with certainty in those cases.

Figure 3.11 presents 99% confidence bounds on the objective function value of the solution at the final iteration of both algorithms. These confidence bounds are based on sampling and solving the recourse problem 10,000 times. We see that the confidence interval for restricted recourse with optimized outcomes is below that of SAA for 3 of the 25 product cases and one of the 10 product cases. The confidence interval for SAA is below that for restricted recourse with optimized outcomes for 5 of the 25 product cases and 4 of the 10 product cases. For the remainder of the cases, the confidence intervals overlap.

We close this section with some further observations associated with Figure 3.11. For the 10 and 25 product cases, we notice that SAA most likely produces the lowest cost solutions for the instances with the highest demand variance, where demand is uniformly distributed between 10 and 1000. We pointed out previously that the method of optimized outcomes tended to outperform sequential bounding without optimized outcomes for these same cases. We offered

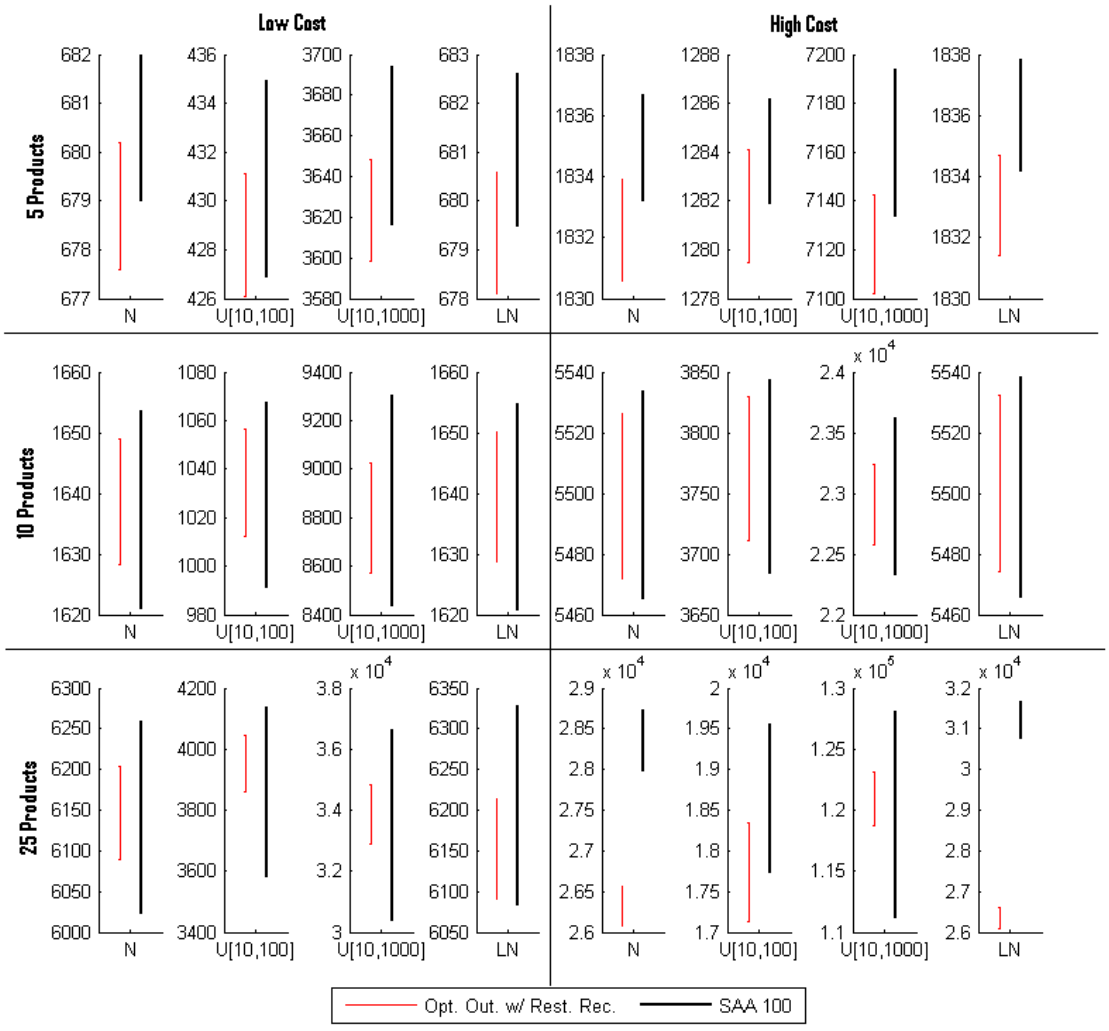

Figure 3.10: Deterministic bounds for the optimal objective function value. Results are shown for the optimized outcomes approach with restricted recourse and SAA with 100 scenarios for the lower bounding problem. The test instances include 4 different demand distributions, 3 numbers of products, and 2 cost structures.

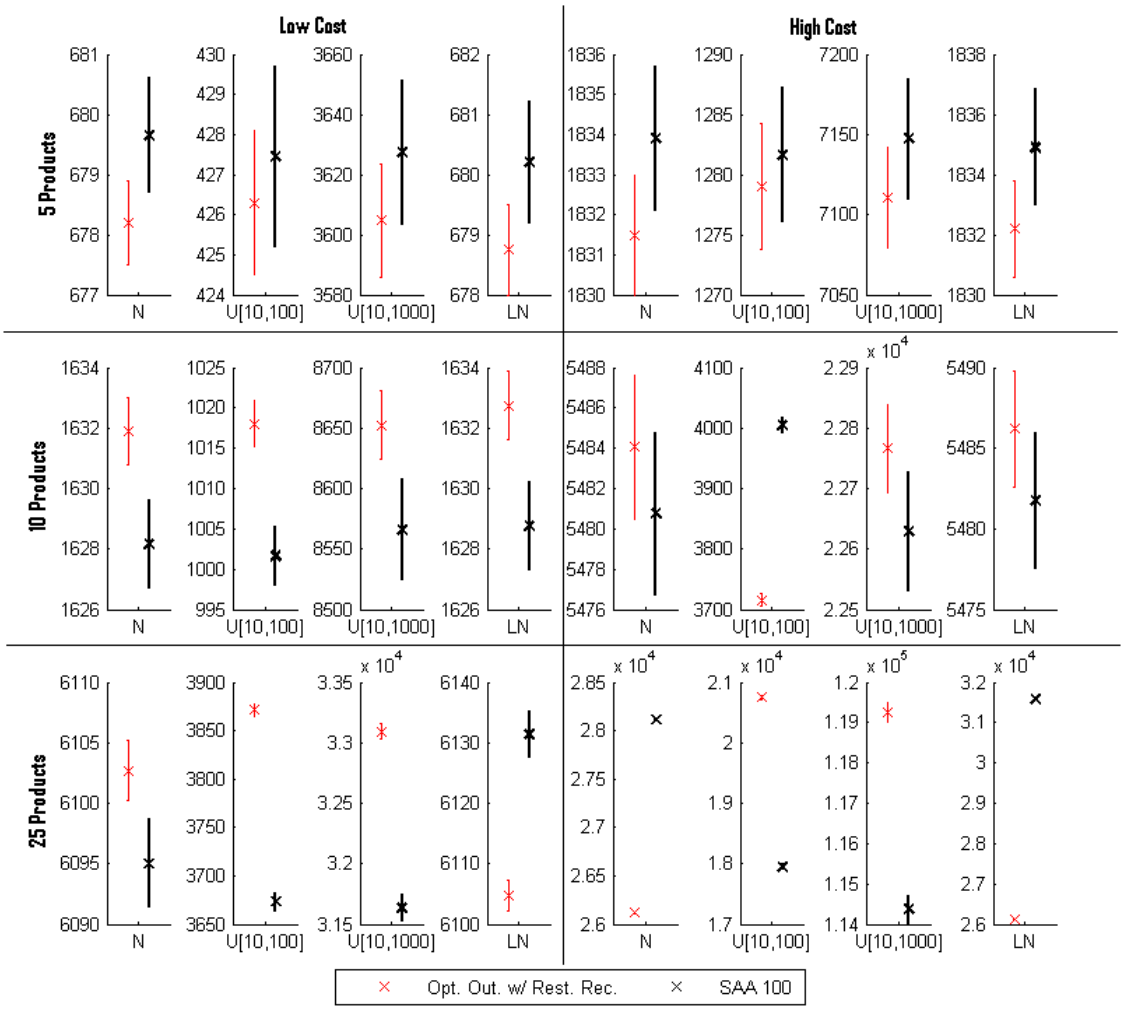

Figure 3.11: Estimate and 99 % confidence bounds for the optimal objective function value at the final iteration. The estimated objective value is marked with an "x". Results are shown for the optimized outcomes approach with restricted recourse and SAA with 100 scenarios for the lower bounding problem. The test instances include 4 different demand distributions, 3 numbers of products, and 2 cost structures.

explanations for this phenomenon for larger numbers of products and higher demand variance. Although optimized outcomes improved the results for these cases, it appears that there is still room for improvement. It also appears that the greatest potential for further improving the sequential bounding algorithm lies in finding effective ways to more accurately approximate continuous distributions with a finite and discrete set of outcomes, especially when the continuous distributions have high variance and a large number of random variables are present in the problem.

3.7 Conclusions

We presented new algorithms for improving optimal decision variable bounds for two-stage stochastic programs. We further demonstrated how bounds can be obtained efficiently using simple recursion for certain types of problems. The new approaches were incorporated into a sequential bounding algorithm, first proposed by Birge (1985b), for solving two-stage stochastic programs. These approaches also extend the restricted recourse bounds developed by Morton and Wood (1999) to more effectively bound recourse problems. We incorporated these bounds into a sequential bounding approach as well. We also developed a new mathematical programming approach to obtain an upper bound on the optimal objective function value of 2SLP based on substituting random variables with decision variables to obtain optimized outcomes. A wide variety of two-stage stochastic programs can be solved in a bounding and parititioning framework. This includes situations where the second stage problem has random right hand side values as well as random cost, technology and recourse matrix coefficients. In Section 3.3 we made relatively few assumptions, even allowing for unbounded support in the random right hand side values.

We found that the sequential bounding algorithm is capable of generating solutions for challenging two-stage stochastic programming problems with continuous support. For this problem, we found that a combination of restricted recourse, described in Section 3.4.1, decision variable bounds based on a linear programming relaxation of the BNPs, described in Section 3.5.2, and

the approach of optimized outcomes, described in Section 3.4.2, are consistently better than the global bounds approach for sequential bounding. Although the sequential bounding algorithms developed in this chapter require more time per iteration for the sequential bounding algorithm than the version with global decision variable bounds, we found that the extra time per iteration was worthwhile. In addition to a deterministic measure of the optimality gap, our computational experiments provide new evidence that sequential bounding can produce lower cost solutions than those found using the SAA for some cases. Thus it may be a promising heuristic for 2SLPs that are difficult or impossible to solve exactly.

We compared the sequential bounding algorithms with two versions of a Monte Carlo sampling based approach. Sampling based approaches cannot produce a measure of the optimality gap with certainty. So, using sequential bounding, we generated deterministic bounds on the solution found at the last iteration for each type of algorithm. In 2 out of 24 test cases we found that the restricted recourse version of sequential bounding with optimized outcomes produced a lower cost solution than that found by the Sample Average Approximation with 100 scenarios used for the lower bounding problem. In all other cases, the deterministic bounds overlap. We also generated 99% confidence bounds on the solution found at the last iteration of each algorithm. In many cases, these confidence intervals overlapped; however, the SAA confidence interval was below that of the sequential bounding algorithm in 9 of the cases. The sequential bounding algorithm's confidence interval was below that of the SAA for 4 of the cases.

There are many opportunities for future research. We provided some insights into future directions for improving the performance of sequential bounding. Additional computational results for other two-stage stochastic programming problems would help determine for which applications sequential bounding is most effective. There may be other special cases or other types of stochastic programs to which the methods we described can be extended. For example, in Chapter 4, we developed a multi-stage version of the sequential bounding algorithm similar to the two-stage version described in this chapter.

Chapter 4

Scenario Generation for Multi-Stage Stochastic Programming and Production Planning

4.1 Introduction

In this chapter we discuss scenario generation methods for multi-stage stochastic linear programs (MSLPs). MSLPs are commonly used to model decision processes with a finite number of stages. Decisions are made sequentially prior to the realization of random parameters after each stage. The decisions at each stage are subject to a set of linear constraints based on previous decisions as well as deterministic and random model parameters. The goal of the decision maker is to maximize or minimize the expected value of a linear function of decision variables. See Di Domenica et al. (2009) for a long list of publications on the applications of multi-stage stochastic programming to finance, supply chain management, transportation, telecommunications, environmental conservation, and energy production planning.

Although MSLPs have advantages when compared to linear programming and two-stage stochastic programming, there are barriers to their adoption in practice. Like two-stage stochas-

tic programs, MSLPs make full use of probability distributions for random variables, eliminating the need to replace random problem parameters with mean values. MSLPs also provide for a sequence of system observations and recourse decisions, whereas two-stage stochastic programming is limited to a single recourse stage after all the random problem parameters have been revealed. For example, Huang and Ahmed (2009) use an MSLP to model a semiconductor tool planning problem. They use a linear programming relaxation to approximately solve the problem, producing much better solutions than those found using a two-stage stochastic programming model. Zanjani et al. (2010) has also demonstrated the value of a multi-stage model for a saw mill production planning problem. Despite these potential benefits, we are not aware of any real-life use of MSLP models for supply chain management.

Failure to adopt MSLPs in practice appears to be due, in part, to the size of such models and the associated computational challenges. MSLPs are too difficult to solve directly unless a small number of joint random variable outcomes are possible at each stage. In most practical applications the MSLP is approximated by a smaller MSLP. A major barrier to the application of such models is the design of the scenario tree that approximates the multi-stage stochastic nature of random factors, such as product demand and yield losses associated with manufacturing processes.

In this chapter we propose a new algorithm to generate a multi-stage scenario tree. This algorithm uses dual information as well as column and row aggregation to calculate deterministic bounds on the optimal objective function value. The algorithm refines a partition of the random variable support set and seeks to improve the scenario tree at every iteration. We compare the performance of our algorithm with three other well known algorithms: a simple random sampling method, a moment matching approach similar to Høyland and Wallace (2001) based on Miller and Rice (1983), and expected value of perfect information (EVPI) importance sampling developed by Dempster and Thompson (1998).

We compare the three approaches using a rolling horizon simulation model to evaluate the quality of the solutions in which product demand and production yields are continuously

distributed at each stage. We use a series of test instances to compare the scenario generation methods on the basis of solution quality, computation time, and difficulty of implementation. In many of the experiments, we did not detect a statistically significant difference between any of the algorithms tested. In those cases where a statistically significant difference in the optimal solution value was found, none of the algorithms consistently outperformed all the others. This supports the view that scenario tree generation can be successfully automated.

The remainder of this chapter is organized as follows. In Section 4.2, we begin by reviewing the literature on scenario tree generation and supply chain planning in the context of multi-stage stochastic programming. We also review previously published work on bounding and solution methodologies for these problems. In Section 4.3 we review the sampling, moment matching, and EVPI methods for generating scenario trees. We also describe the new scenario generation approach. In Section 4.4, we cover the notation and model formulations that will be used for computational experiments. We compare the quality of the solutions found for all four algorithms using computational experiments in Section 4.5. Concluding remarks are made in Section 4.6.

4.2 Literature Review

In this literature review we focus on MSLPs and scenario generation methods. We provide a more thorough review of scenario generation methods than that presented in Chapter 2.

MSLPs are generally considered difficult to solve. They often involve optimizing the expected value of a function that depends on many random variables. Evaluating such a function involves calculating a high-dimensional integral. As explained in Dantzig and Infanger (1993), Monte Carlo methods are usually suggested for computing such integrals. In Shapiro et al. (2009) difficulties associated with random sampling, also known as Monte Carlo sampling or the sample average approximation method, are investigated. Under certain assumptions, the number of scenarios required to guarantee a specific level of accuracy in the solution of a t-stage stochastic program is the square of the number of scenarios required in a $t-1$-stage stochastic program.

Hence, the number of scenarios required is exponential in the number of stages. However, the authors point out that this analysis does not guarantee exponential complexity in solving MSLP models in general.

Scenario trees have been used to develop approximate solutions to difficult MSLPs for over twenty years (Mulvey and Vladimirou, 1989). Scenario trees are directed graphs consisting of nodes and arcs without cycles. We will refer to the arcs as branches. An example of a scenario tree is given in Figure 4.1. Nodes at each level of the tree correspond to all possible decision points that can be reached at the corresponding stage. Consider a branch from a node at level t to a node at level $t+1$ in the tree. This branch corresponds to a single set of outcomes for all those random variables whose values are unknown at stage t but revealed at stage $t+1$. A scenario tree representation of an MSLP is only possible when the random problem parameters are limited to a finite number of possible outcomes. Kaut and Wallace (2007), Dupacova et al. (2001), and Kall and Wallace (1994) review a variety of methods for generating and evaluating the quality of scenario trees.

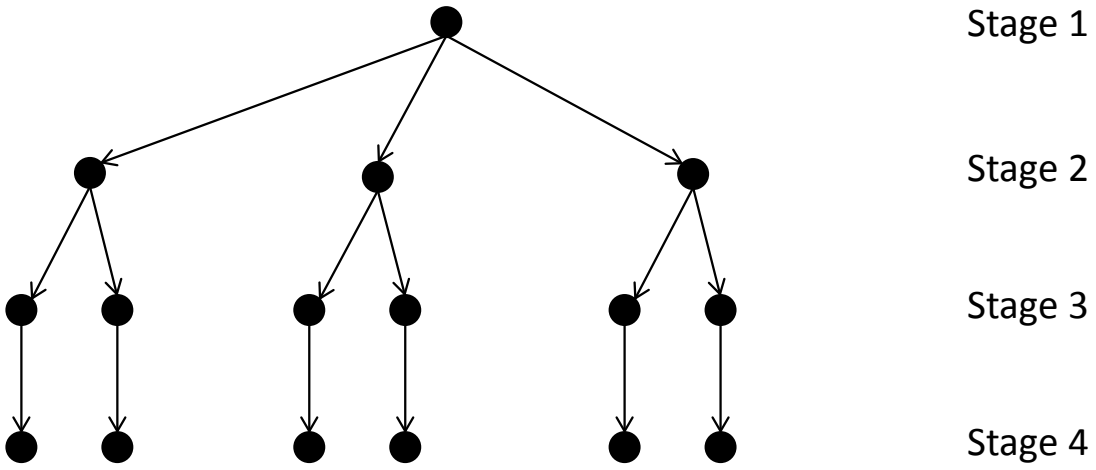

Figure 4.1: A scenario tree with 6 scenarios

Next, we review existing methods for generating scenario trees. We hope to generate random

variable outcomes and associated probabilities while maintaining certain statistical properties associated with the underlying joint probability distribution for the random variables. For example, we can match the first several moments and covariances or minimize any measure of distance between the chosen values and the underlying distribution. Høyland and Wallace (2001) show how to model the problem of determining these values as a nonlinear programming problem. The problem must be formulated carefully, or else too many matching constraints may make the problem infeasible. After the nonlinear program is formulated, it may not be possible to find an optimal solution in a reasonable amount of time, and a heuristic method must be used. Miller and Rice (1983) describe a way to generate N probabilities and corresponding random variable outcomes so that the first $2N - 1$ moments match those of the underlying probability distribution. This approach is based on Gaussian quadrature and was used for our implementation of the moment matching algorithm. Kouwenberg (2001) reports significant benefits of the moment matching approach compared to random sampling.

The EVPI importance sampling approach to scenario generation was developed by Dempster and Thompson (1998) and Dempster (2006). Unlike the previously mentioned approaches, this algorithm involves repeatedly refining the scenario tree and solving the associated MSLP. Refinements are based on the expected value of perfect information (EVPI) associated with each node. This quantity can be thought of as the amount that a risk-neutral decision maker should be willing to pay for perfect knowledge of all future outcomes. Relatively higher values of EVPI at a node result in adding more branches, and lower values result in removing branches. In this manner we hope to gain a more accurate characterization of the random outcomes when there is more value to be gained from knowing future outcomes.

Some scenario tree generation approaches have been developed based on lower and upper bounds on the optimal objective function of a stochastic program. We review such approaches next. In the case where random variables are independent and the objective function is convex in the random variables, bounds can be found using the classic Edmundson-Madansky and Jensen's inequalities (see Birge and Louveaux, 1997, Chapt. 9). Huang et al. (1977) shows that

these bounds can be made arbitrarily tight by sequentially partitioning the support of the random variables into a successively finer set of cells and applying the bounds to each new cell. This approach requires evaluating a function at the extreme points of each cell, involving 2^n function evaluations when the cell is a hyper-rectangular region and n the number of random variables. Hence, this approach is limited to problems where n is relatively small.

In Birge (1985a), the work of Zipkin (1980b,a) is extended to obtain deterministic bounds for the optimal objective function value of multi-stage stochastic programming problems with continuously distributed random variables appearing in the right hand side of the constraints. Constraints are aggregated according to a weighting function. Decision variables are similarly aggregated. These weighting functions are the joint conditional probability distribution functions for the random variable outcomes on which the decision variables depend. Wright (1994) develops a measure-theoretic probability framework for the results of Birge (1985a), allowing for uncertainties in the constraint and cost coefficients.

The bounding approach of Birge (1985a) has been used to solve two-stage stochastic programs. In Denton and Gupta (2003), an L-shaped method with sequential bounding is developed for an appointment scheduling problem. The decision variables computed at each iteration converge to an optimal solution, along with the upper and lower bound for the optimal objective function value. From the computational experience with this problem, the authors found that the upper bound converged more slowly for this minimization problem than the lower bound.

Casey and Sen (2005) use the bounding approach of Birge (1985a) and Wright (1994) to develop a scenario tree generation algorithm for multi-stage stochastic programs with random parameters appearing in the right hand side of the constraints. Their procedure is similar to that proposed in this chapter. Both approaches repeatedly solve the MSLP problem associated with a scenario tree and update the scenario tree by splitting one of the nodes in the tree. The node splitting corresponds to a refinement of the partition of the support set of random variables. The two approaches for generating a scenario tree differ in how this node is selected. The method of Casey and Sen (2005) depends on developing bounds on the expected salvage value for each

node in the scenario tree, which contributes to an overall measure of the optimality gap. The expectation is taken over all scenarios that pass through the node. This reduces multi-stage problems to solving bounds on a two-stage problem for every node at every iteration. The resulting two-stage problems may still be difficult to solve. The authors did not develop or provide any details for a practical computer implementation for this method. We avoid such calculations by splitting the node that contributes most to the difference in upper and lower bounds on the optimal objective function value. Our approach also allows for random parameters appearing in the cost and constraint coefficients.

The contributions of this chapter are as follows. First, we propose a new algorithm for generating scenario trees using bounding and partitioning. The algorithm provides deterministic bounds on the optimal objective function value of an MSLP problem that tend to improve as the algorithm progresses. We also present experimental evidence suggesting that the algorithm proposed in this chapter is as effective as algorithms that require tuning and problem-specific modifications.

4.3 Scenario Generation Methods

Before describing the methods we use to solve MSLP models, for convenience, we reproduce the following formulation and description of the general MSLP problem previously given in Equation 1.2:

$$
\begin{aligned}
\max \quad & c^1x^1 + \textstyle\sum_{t=2}^{T} E_{\xi^{[t]}}\left[c^t(\omega^{[t]})x^t(\omega^{[t]})\right] \\
\text{s.t.} \quad & W^1x^1 \leq h^1 \\
& \mathcal{T}^2(\omega^2)x^1 + W^2(\omega^2)x^2(\omega^2) \leq h^2(\omega^2), \text{ for all } \omega^2 \in \Omega^2 \\
& \mathcal{T}^t(\omega^{[t]})x^{t-1}(\omega^{[t-1]}) + W^t(\omega^{[t]})x^t(\omega^{[t]}) \leq h^t(\omega^{[t]}), \\
& \text{for all } \omega^{[t]} \in \Omega^2 \times \ldots \times \Omega^t \text{ and } t = 3, \ldots, T \\
& x^1, x^t(\omega^{[t]}) \geq 0, \text{ for all } \omega^{[t]} \in \Omega^2 \times \ldots \times \Omega^t \text{ and } t = 2, \ldots, T
\end{aligned} \tag{4.1}
$$

A detailed description of the notation used in this formulation follows. The t^{th} stage column vector of decision variables of length n_t is x^t. In stage t, the problem parameters include the $m_t \times n_t$ matrix W^t, the row vector c^t, the column vector h^t, and the matrix $\mathcal{T}^t$ (note that we use the caligraphic font to distinguish this matrix from the number of stages, T). All of these matrices and vectors have conformal dimensions, and starting with the second stage, may contain random variable entries. For $t = 2, \ldots, T$, the components of ξ^t consist of all the random variable problem parameters associated with stage t . Random matrices, vectors, and decision variables are indexed on $\omega^2 \in \Omega^2$ and $\omega^t \in \Omega^t(\omega^{[t-1]})$, the set of outcomes at stage 2 and $3 \leq t \leq T$ respectively. Here we use the notation $\omega^{[t]} = (\omega^2, \ldots, \omega^t)$ to denote the vector of random outcomes prior to and including stage t. The support for the random vector ξ^2 is Ξ^2. Similarly, the support for ξ^t is $\Xi^t(\omega^{[t-1]})$ for $t = 3, \ldots, T$. Sometimes it is convenient to refer to the vector of random problem parameters associated with every stage, $\xi = (\xi^2, \ldots, \xi^T)$. The realization of any such random vector is called a scenario, and the realization of any vector $\xi^{[t]} = (\xi^2, \ldots, \xi^t)$, with $2 \leq t \leq T$, is called a subscenario. The support set for all scenarios is denoted $\Xi = \{\xi(\omega^{[T]}) | \omega^2 \in \Omega^2, \ldots, \omega^T \in \Omega^T(\omega^{[T-1]})\}$, and the support set for all stage t subscenarios is denoted $\Xi^{[t]} = \{\xi^{[t]}(\omega^{[t]}) | \omega^{[t]} \in \Omega^2 \times \ldots \times \Omega^t\}$.

The MSLP formulation given in Equation 4.1 reduces to a linear program when Ξ is finite. This formulation is commonly referred to as the *deterministic equivalent* linear program

(Charnes and Cooper, 1963). Since all random problem parameters are limited to a finite number of possible outcomes, this corresponds to a scenario tree. This formulation is shown below in Equation 4.2.

$$
\begin{aligned}
\max \quad & c^1 x^{1,1} + \textstyle\sum_{t=2}^{T} \sum_{k=1}^{K_t} p^{t,k} c^{t,k} x^{t,k} \\
\text{s.t.} \quad & W^1 x^{1,1} = h^1 \\
& T^{t,k} x^{t-1,a_t(k)} + W^{t,k} x^{t,k} = h^{t,k}, \text{ for all } k = 1, \ldots, K_t \text{ and } t = 2, \ldots, T \\
& x^{1,1}, x^{t,k} \geq 0, \text{ for all } k = 1, \ldots, K_t, \text{ and } t = 2, \ldots, T
\end{aligned} \tag{4.2}
$$

A detailed description of the notation used in this formulation follows. First, we number all possible subscenarios $\xi^{[t]} = (\xi^2, \ldots, \xi^t)$ for $t = 2, \ldots, T$. These subscenarios correspond to paths from the root node to a node at level t in the scenario tree. We use the superscript index k for the decision variables and random problem parameters in accordance with this numbering scheme. $a_t(k)$ is defined to be the number of the subscenario at stage $t-1$ that is the parent of subscenario k at stage t. So, if subscenario 20 is given by $(\xi^2, \xi^3, \xi^4) = (\hat{\xi}^2, \hat{\xi}^3, \hat{\xi}^4)$, then $a_3(10)$ is the number for subscenario $(\hat{\xi}^2, \hat{\xi}^3)$. We use the convention that $a_2(k) = 1$ for all $k = 1, \ldots, K_2$. We also define $p^{t,k}$ to be the probability that subscenario k occurs at stage t.

In most realistic situations the support set for the random variables, Ξ, is either infinite or too large for the linear programming problem of Equation 4.2 to be solved with a limited amount of time and memory. Therefore, we must approximate the true MSLP model with one where Ξ is replaced with a smaller, finite set $\hat{\Xi}$. We then hope that the result will be close to an optimal solution for the original MSLP model. Since $\hat{\Xi}$ is finite, it corresponds directly to a scenario tree. In the rest of this section, we describe methods for generating such a scenario tree.

4.3.1 Monte Carlo Sampling

The Monte Carlo sampling algorithm, which goes by other names such as sample average approximation (SAA) (Shapiro, 2001), generates a scenario tree using a random number generator. A description is given in Algorithm 4. Notice that the algorithm allows random numbers to depend on previously generated random numbers when random problem parameters associated with a stage depend on those of previous stages. Once the algorithm generates the tree, the corresponding deterministic equivalent linear programming problem is solved. The time and memory required to solve this problem depends on the size and structure of the scenario tree. The number of nodes and branches per node at each level of the tree must be known before the algorithm starts. In our description of the algorithm, we assume that the number of branches per node is the same at each level of the tree and given by the quantity, $b(t)$, at level t of the tree. So, for example, if $b(1) = 5$, there will be five branches from the root node of the tree. Ideally, the tree structure is based on the computational experience of the user and knowledge of the specific application. So, success with this approach often depends on experience and intuition about the problem.

Algorithm 4 Monte Carlo Sampling

Require: $b(t)$ = The number of branches per node at level t
  for $t = 2$ to T **do**
    for all Nodes at level t **do**
      for $i = 1$ to $b(t)$ **do**
        Retrieve $\xi^2, \ldots, \xi^{t-1}$ values for the current node if necessary
        Use the random number generator to generate values for ξ^t
        Assign the generated values to branch i from the current node
      end for
    end for
  end for

4.3.2 Method of Matching Moments

Like Monte Carlo sampling, the method of matching moments involves generating a scenario tree and solving the corresponding MSLP. Both algorithms begin by defining a specific tree structure which corresponds to the number of outcomes of random variables associated with each stage of the MSLP. We again assume the number of branches per node $b(t)$ is the same at each level of the tree. We also assume that the number of outcomes, $O_i(t)$, for the i^{th} component of the random variable vector ξ^t has been determined, and $\prod_{i=1}^{M} O_i(t) = b(t)$. So, for example, if ξ^2 has two components, we may decide to have 3 outcomes for the first component, $O_1(2) = 3$, and 2 outcomes for the second, $O_2(2) = 2$, giving us a total of 6 branches for every node, $b(2) = 6$, at the second level of the tree.

As described in Section 4.2, whenever we need to generate N outcomes for a random variable, we use a method based on Gaussian quadrature described by Miller and Rice (1983). This ensures that the first $2N - 1$ moments match those for the underlying distribution and avoids solving a nonlinear programming problem. This method must be carried out for each random variable. Values for the probabilities and outcomes for uniform and normally distributed random variables are conveniently tabulated in Stroud and Secrest (1966). We simply include a branch in the scenario tree for each combination of random variable outcomes and assign the branch a probability equal to the product of the probabilities associated with the outcomes. The approach is presented in detail in Algorithm 5.

4.3.3 EVPI Importance Sampling

We outline the EVPI importance sampling algorithm from Dempster and Thompson (1998); Dempster (2006). This algorithm starts by generating a relatively small scenario tree using the Monte Carlo Sampling Algorithm. The corresponding deterministic equivalent linear programming problem is solved and the scenario tree is modified at every iteration of the algorithm. Modifications to the scenario tree are based on the EVPI associated with each decision node. This value is the maximum amount that a risk-neutral decision maker should be willing to pay

Algorithm 5 Moment Matching

Require: $b(t)$ = The number of branches per node at level t.
Require: $O_i(t)$ = the number of outcomes to assign to the i^{th} component of ξ^t so that $O_1(t) \cdot \ldots \cdot O_M(t) = b(t)$
  for $t = 1$ to $T - 1$ **do**
    for all Nodes at level t **do**
      Retrieve $\xi^2, \ldots, \xi^t$ values for the current node if necessary
      for $i = 1$ to M **do**
        Look up $O_i(t)$ sets of values and associated probabilities for the i^{th} component of ξ^{t+1} conditioned on outcomes for components $1, \ldots, i-1$ of ξ^{t+1} and all components of $\xi^2, \ldots, \xi^t$
      end for
      Assign each unique combination of random variable outcomes to each of the branches from the current node, and set the probability of the branch equal to the product of the probabilities of all the outcomes associated with the branch
    end for
  end for

for a perfectly accurate prediction of all future outcomes.

Next, we describe how the nodes in the scenario tree are modified by the EVPI algorithm. The user of the algorithm must specify a threshold EVPI tolerance, *tol*. Nodes with an EVPI below *tol* will either be re-sampled or have branches and subsequent nodes removed from the scenario tree. A low EVPI value may simply be the result of sampling low probability outcomes. The chances of this taking place are higher when there are only a few branches from the node in question. So, sampling is carried out several times before removing branches in the subtree corresponding to these nodes. Nodes with an EVPI above *tol* will have branches and nodes added. This allows the tree to grow in portions where uncertainty is likely to have a greater effect on decisions being made. The user must also specify termination criteria for the algorithm, the maximum execution time *max_time* and the maximum number of iterations *max_iter*. See Algorithm 6 for a detailed description.

Besides the algorithm control parameters previously described, the EVPI algorithm can be adjusted in other ways. For example, the number of groups of nodes with an EVPI above *tol* can be modified, and the number of branches added to these nodes can also be changed. For nodes

Algorithm 6 EVPI Importance Sampling

Require: TREE = A small scenario tree generated by the Monte Carlo Sampling Algorithm
Require: max_time, max_iter, and tol to be defined
  $t \leftarrow 2$
  while $i \leq max\_iter$ AND $time \leq max\_time$ **do**
    Solve the deterministic equivalent linear programming model from TREE
    Calculate $EVPI(n)$ = the local EVPI for each node n in TREE
    for all nodes n at level t with $EVPI(n) \leq tol$ **do**
      if n has 2 or fewer children and has been re-sampled 2 or fewer times **then**
        Re-sample the subtree at node n
      else
        Collapse the subtree at node n into a single scenario
      end if
    end for
    Sort and divide nodes at level t with EVPI$> tol$ into 3 groups by EVPI value
    Add 1, 2, and 3 branches to nodes in groups 1, 2, and 3 respectively
    if all three groups are empty **then**
      If $t = 1$, then the algorithm is done. Otherwise, $t \leftarrow \max\{1, t-2\}$
    else
      If $t = T$, then $t \leftarrow 2$. Otherwise, $t \leftarrow t+1$
    end if
    $i \leftarrow i+1$
  end while

with EVPI below *tol*, we can also adjust the maximum number of resamplings before collapsing the subtree. The EVPI algorithm does not give as much control to the user as the Monte Carlo Sampling Algorithm in specifying the final scenario tree, but just as with the Monte Carlo algorithm, success with this approach will depend on the experience and judgement of the user.

4.3.4 A New Scenario Tree Generation Algorithm

The following is the dual of MSLP given in Equation 4.1.

$$
\begin{aligned}
\min \quad & \pi^1 h^1 + \textstyle\sum_{t=2}^{T} E_{\xi^{[t]}} \left[\pi^t(\omega^{[t]}) h^t(\omega^{[t]}) \right] \\
\text{s.t.} \quad & \pi^1 W^1 + E_{\xi^2}[\pi^2(\omega^2)\mathcal{T}^2(\omega^2)] \geq c^1 \\
& \pi^{t-1}(\omega^{[t-1]}) W^{t-1}(\omega^{[t-1]}) + E_{\xi^t|\xi^{[t-1]}}[\pi^t(\omega^{[t]})\mathcal{T}^t(\omega^{[t]})] \geq c^{t-1}(\omega^{[t-1]}), \\
& \text{for all } \omega^{[t-1]} \in \Omega^2 \times \ldots \times \Omega^{t-1} \text{ and } t = 3, \ldots, T \\
& \pi^T(\omega^{[T]}) W^T(\omega^{[T]}) \geq c^T(\omega^{[T]}), \text{ for all } \omega^{[T]} \in \Omega^2 \times \ldots \times \Omega^T \\
& \pi^1, \pi^t(\omega^{[t]}) \geq 0, \text{ for all } \omega^{[t]} \in \Omega^2 \times \ldots \times \Omega^t \text{ and } t = 1, \ldots, T
\end{aligned} \tag{4.3}
$$

By simply replacing all random problem parameters in the general MSLP formulation of Equation 4.1 with their mean values, we can obtain a deterministic linear programming problem like that given in Equation 4.2. So, ξ^2 is replaced with $\xi^{2,1} = E[\xi^2]$, and ξ^t is replaced with $\xi^{t,1} = E_{\xi^t|\xi^{[t-1]}}[\xi^t|\xi^{[t-1]} = (\xi^{2,1}, \ldots \xi^{t-1,1})]$ for all $t = 3, \ldots, T$. This corresponds to a degenerate scenario tree with a single branch at every node, and the branch from the node at level t corresponds to $\xi^{t+1,1}$ for $t = 1, \ldots, T-1$. The resulting deterministic linear program is commonly referred to as the *mean value problem* for the MSLP of Equation 4.1, see page 139 of Birge and Louveaux (1997) for example.

By partitioning the support set Ξ and replacing random variables with their conditional mean values, we can generate a scenario tree of any desired structure from the general MSLP formulation of Equation 4.1. We begin by describing how to obtain the first and second levels

of the scenario tree. If the number of branches at the root node is K_2, then we begin by partitioning the support set for Ξ^2 into K_2 disjoint, non-empty subsets $\{\Xi_k^2\}_{k=1}^{K_2}$ such that $\Xi_{k_1}^2 \cap \Xi_{k_2}^2 = \emptyset$ for all $k_1 \neq k_2$, and $\cup_{k=1}^{K_2} \Xi_k^2 = \Xi^2$. For convenience, we also assume that each Ξ_k^2 is the intersection of Ξ^2 with a full-dimensional hyper-rectangular cell. The k^{th} branch from the root node corresponds to $\xi^{2,k} = E[\xi^2 | \xi^2 \in \Xi_k^2]$ for $k = 1, \ldots, K_2$. The components of each vector in $\{\xi^{2,k}\}_{k=1}^{K_2}$ supply the second stage problem parameters in Equation 4.2.

We generate the subsequent levels of the scenario tree in a similar manner. Let $D_t(k)$ denote the set of indices for all nodes that are descendants of node k at level t in the scenario tree. Given the partition of the second stage support set $\Xi^2 = \cup_{k=1}^{K_2} \Xi_k^2$, we recursively define a partition for every subscenario support set $\Xi^{[t]}$ for $t = 2, \ldots, T$ as follows. Let $\Xi_k^{[2]} \equiv \Xi_k^2$ for $k = 1, \ldots, K_2$. For $t = 3, \ldots, T$ and $k_{t-1} = 1, \ldots, K_{t-1}$, let $\{\Xi_k^{[t]}\}_{k \in D_{t-1}(k_{t-1})}$ be a partition of $\{\xi^{[t]} \in \Xi^{[t]} | \xi^{[t-1]} \in \Xi_{k_{t-1}}^{[t-1]}\}$ into disjoint non-empty subsets. For convenience, we define Ξ_k^t so that $\Xi_k^{[t]} = \Xi_{k_{t-1}}^{[t-1]} \times \Xi_k^t$. If $a_t(k) = k_{t-1}$, then the branch from node k_{t-1} at level $t-1$ to node k at level t corresponds to $\xi^{t,k} = E_{\xi^t | \xi^{[t-1]}}[\xi^t | \xi^{t-1} = \xi^{t-1,a_t(k)}]$. The components of each vector in $\{\xi^{t,k}\}_{k=1}^{K_t}$ supply the stage t problem parameters in Equation 4.2.

After constructing this scenario tree, we can apply the result of Wright (1994) to establish bounds on the optimal objective function value z^* of the MSLP model of Equation 4.1. Our version of the result allows for non-zero lower bounds on the optimal dual and primal decision variables. This results in tighter bounds for z^*. We include the result below as Proposition 1. The subscenario partition $\{\Xi_k^{[t]}\}_{k=1}^{K_t}$ is used to aggregate stage t constraints and decision variables in the MSLP. For the corresponding deterministic linear programming problem of Equation 4.2, let the optimal objective function value be $\tilde{Z}$. For $k = 1, \ldots, K_t$ and $t = 2, \ldots, T$, let $\tilde{X}^{t,k}$ be the optimal primal solution. Similarly, let $\tilde{W}$ be the optimal dual objective function value and $\tilde{\Pi}^{t,k}$ the associated optimal dual solution to Equation 4.2.

Proposition 1. *Assuming strong duality holds for the MSLP in Equation 4.1,* $\tilde{Z} - \epsilon^1 \leq z^* \leq$

$\tilde{W} + \epsilon^2$, *where*

$$
\begin{aligned}
\epsilon^1 = & \sum_{i=1}^{m_1} \overline{\pi}_i^{1,1}(h_i^1 - W_{i,\cdot}^1 \tilde{X}^{1,1}) \\
& + \sum_{t=2}^{T} \sum_{k=1}^{K_t} \sum_{i=1}^{m_t} \overline{\pi}_i^{t,k} \int_{\Xi_k^{[t]}} \left[-h_i^t + W_{i,\cdot}^t \tilde{X}^{t,k} + \mathcal{T}_{i,\cdot}^t \tilde{X}^{t-1,a_t(k)} \right]^+ dF(\xi^{[t]}) \\
& - \sum_{t=2}^{T} \sum_{k=1}^{K_t} \sum_{i=1}^{m_t} \underline{\pi}_i^{t,k} \int_{\Xi_k^{[t]}} \left[h_i^t - W_{i,\cdot}^t \tilde{X}^{t,k} - \mathcal{T}_{i,\cdot}^t \tilde{X}^{t-1,a_t(k)} \right]^+ dF(\xi^{[t]}) \\
\epsilon^2 = & \sum_{j=1}^{n_1} \overline{x}_j^{1,1} \left[c_j^1 - \tilde{\Pi}^{1,1} W_{\cdot,j}^1 - \sum_{k=1}^{K_2} \int_{\Xi_k^2} \tilde{\Pi}^{2,k} \mathcal{T}_{\cdot,j}^2 dF(\xi^2) \right]^+ \\
& - \sum_{j=1}^{n_1} \underline{x}_j^{1,1} \left[-c_j^1 + \tilde{\Pi}^{1,1} W_{\cdot,j}^1 + \sum_{k=1}^{K_2} \int_{\Xi_k^2} \tilde{\Pi}^{2,k} \mathcal{T}_{\cdot,j}^2 dF(\xi^2) \right]^+ \\
& + \sum_{t=3}^{T} \sum_{k=1}^{K_{t-1}} \sum_{j=1}^{n_t} \overline{x}_j^{t-1,k} \int_{\Xi_k^{[t-1]}} \left[c_j^{t-1} - \tilde{\Pi}^{t-1,k} W_{\cdot,j}^{t-1} \right. \\
& \qquad \left. - \sum_{d \in D_{t-1}(k)} \int_{\Xi_d^t} \tilde{\Pi}^{t,d} \mathcal{T}_{\cdot,j}^t dF(\xi^t | \xi^{[t-1]}) \right]^+ dF(\xi^{[t-1]}) \\
& - \sum_{t=3}^{T} \sum_{k=1}^{K_{t-1}} \sum_{j=1}^{n_t} \underline{x}_j^{t-1,k} \int_{\Xi_k^{[t-1]}} \left[-c_j^{t-1} + \tilde{\Pi}^{t-1,k} W_{\cdot,j}^{t-1} \right. \\
& \qquad \left. + \sum_{d \in D_{t-1}(k)} \int_{\Xi_d^t} \tilde{\Pi}^{t,d} \mathcal{T}_{\cdot,j}^t dF(\xi^t | \xi^{[t-1]}) \right]^+ dF(\xi^{[t-1]}) \\
& + \sum_{k=1}^{K_T} \sum_{j=1}^{n_T} \overline{x}_j^{T,k} \int_{\Xi_k^{[T]}} \left[c_j^T - \tilde{\Pi}^{T,k} W_{\cdot,j}^T \right]^+ dF(\xi^{[T]}) \\
& - \sum_{k=1}^{K_T} \sum_{j=1}^{n_T} \underline{x}_j^{T,k} \int_{\Xi_k^{[T]}} \left[-c_j^T + \tilde{\Pi}^{T,k} W_{\cdot,j}^T \right]^+ dF(\xi^{[T]})
\end{aligned}
$$

Proof. See Wright (1994). □

The notation used in Proposition 1 is described next. Here, we use subscripts to indicate vector components. $\underline{\pi}_i^{t,k}$ and $\overline{\pi}_i^{t,k}$ can be any lower and upper bounds on the optimal dual decision variable, $\pi_i^{t,*}(\omega^{[t]})$, from Equation 4.3 when $\xi^{[t]}(\omega^{[t]}) \in \Xi_k^{[t]}$. Similarly, $\underline{x}_i^{t,k}$ and $\overline{x}_i^{t,k}$ can be any bounds on the optimal primal decision variable, $x_i^{t,*}(\omega^{[t]})$, from Equation 4.1 when $\xi^{[t]}(\omega^{[t]}) \in \Xi_k^{[t]}$. We outline a process in Section 4.3.4 for calculating such bounds on the primal and dual decision variables. We will use $A_{i,\cdot}$ to denote the i^{th} row and $A_{\cdot,j}$ to denote the j^{th} column of any matrix A. $[x]^+ \equiv \max\{0, x\}$ for any value x, $F(\xi^{[t]})$ is the joint distribution function for $\xi^{[t]}$, and $F(\xi^t | \xi^{[t-1]})$ is the conditional distribution of ξ^t given $\xi^{[t-1]}$.

We offer a useful interpretation of Proposition 1. Most of the terms for ϵ^1 and ϵ^2 involve an integral of the form $\int_{\Xi_k^{[t]}} \left[g(\xi^t) \right]^+ dF(\xi^t)$, where $g(\xi^t)$ is a linear function in ξ^t. After identifying the region of integration over $\Xi_k^{[t]}$ where $g(\xi^t) \geq 0$, we can distribute the integral over all

the linear terms of $g(\xi^t)$. The integral of each term can be viewed as the contribution each random variable component of ξ^t makes to ϵ^1 and ϵ^2. Furthermore, since Proposition 1 includes a term for every value of $k = 1, \ldots, K_t$ and $t = 1, \ldots, T$, we can identify the contribution that each random variable makes to each level and node of the corresponding scenario tree. In our algorithm, this interpretation is used to identify nodes for splitting.

Next, we describe how to split a node in a scenario tree. First we select a node in the scenario tree, say node k at level $t > 1$. At node k, the support for ξ^t is Ξ_k^t, a hyper-rectangular cell of the partition. Next, we identify a random variable component of ξ^t. The support of this random variable corresponding to Ξ_k^t is an interval $[a, b]$. Other cells of the partition may also include the interval $[a, b]$ for the selected random variable. We divide all such cells into two new cells, one with the interval $[a, \frac{a+b}{2}]$ and the other with $[\frac{a+b}{2}, b]$ for the random variable. In the scenario tree, this corresponds to replacing node k with two new nodes. An identical copy of the subtree representing all descendant nodes and branches of node k is used as a subtree for each new node. At level $t-1$, node $a_t(k)$ becomes the parent of each of the new nodes. As before, every branch corresponds to the mean values of the random variables restricted to subsets of the support set. A visual representation of a node splitting is shown in Figure 4.2.

We can now describe the new algorithm. The algorithm begins with the construction of a degenerate scenario tree, one with a single node at every level of the tree. We solve the corresponding mean value problem and calculate $\epsilon = \epsilon^1 + \epsilon^2$ using the expressions in Proposition 1. If a time limit, *max_time*, or an iteration limit, *max_iter*, is reached or ϵ is below a specific tolerance, then the algorithm terminates. Otherwise, we identify the node and random variable that contributed most to the calculation of ϵ. We perform a splitting at this node in the scenario tree, dividing one or more of the cells in the partition of the support set. Conditional mean values are calculated for every branch in the modified portion of the scenario tree. We solve the deterministic linear programming problem associated with the new tree, re-calculate ϵ, and check the termination criteria. Node splittings and scenario tree refinements continue until one of the termination criteria are met. See Algorithm 7 for a pseudocode description.

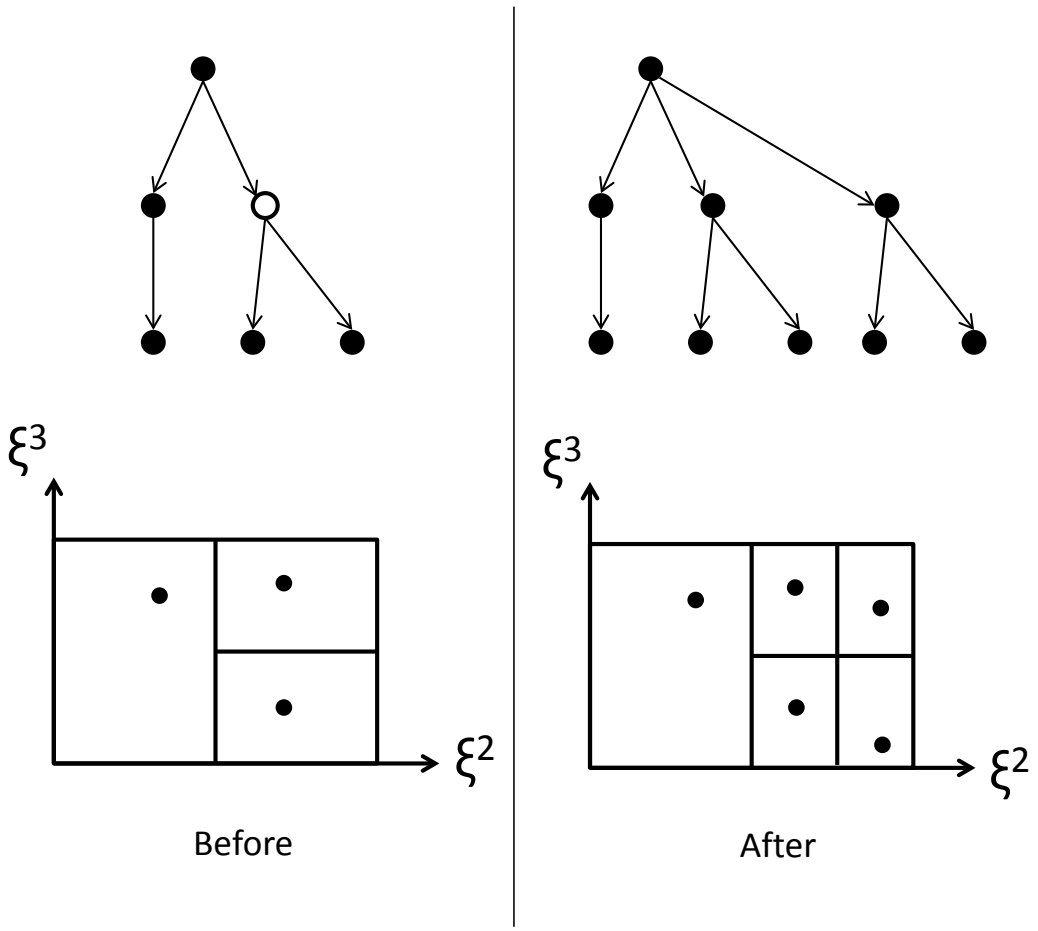

Figure 4.2: The effect of splitting the white node on the scenario tree and corresponding support set partition. Black dots in the support set represent mean values over the respective cells.

Primal and Dual Decision Variable Bounds

We now describe a method for obtaining bounds on the dual decision variables of Equation 4.3. We assume the matrix W^t is deterministic for all t, i.e., the MSLP has fixed recourse. We also assume that the random constraint coefficients are stage-wise independent, but this is not required for other random problem parameters. We further assume that the feasible regions of Equation 4.1 and 4.3 are bounded. In particular, there exists $\underline{\pi}^{T,k}$ and $\overline{\pi}^{T,k}$ such that $0 \leq \underline{\pi}^{T,k} \leq \pi \leq \overline{\pi}^{T,k}$ for all π where $\pi W_T \geq \min_{\xi^{[T]} \in \Xi_k^{[T]}} \{c_T\}$ and $k = 1, \ldots, K_T$.

Algorithm 7 A New Scenario Tree Generation Algorithm

Require: *max_time*, *max_iter*, and *tol* to be defined
  Set $TREE$ to a degenerate scenario tree with mean value outcomes
  while $i \leq max\_iter$ AND $time \leq max\_time$ **do**
    Solve the deterministic equivalent linear programming model from $TREE$
    Calculate $\epsilon = \epsilon^1 + \epsilon^2$
    If $\epsilon < tol$, then the algorithm is done. Otherwise, continue.
    Identify the node and random variable contributing most to ϵ
    Split the node and modify $TREE$ accordingly
    $i \leftarrow i + 1$
  end while

Having established bounds on the stage T dual decision variables, we provide a backwards recursion for calculating bounds on the remaining dual variables. For $k = 1, \ldots, K_t$ and $t = 2, \ldots, T$, we assume $\{\pi | \pi W^{t-1} + \int_{\Xi^t} \pi^t \mathcal{T}^t dF(\xi^t | \xi^{[t-1]}) \geq c^{t-1}\}$ is bounded and we can find $\underline{\pi}^{t-1,k}$ and $\overline{\pi}^{t-1,k}$ such that $0 \leq \underline{\pi}^{t-1,k} \leq \pi \leq \overline{\pi}^{t-1,k}$ for all π where

$$\pi W^{t-1} \geq \min_{\xi^{[t-1]} \in \Xi_k^{[t-1]}} \{c^{t-1}\} - \Psi^{t-1,k}$$

and the j^{th} component of the vector $\Psi^{t-1,k}$ is defined as

$$\Psi_j^{t-1,k} = \sum_{d \in D_{t-1}(k)} \sum_{i=1}^{m_t} \overline{\pi}_i^{t,d} \int_{\Xi_d^t} [\mathcal{T}_{i,j}^t]^+ dF(\xi^t) - \sum_{d \in D_{t-1}(k)} \sum_{i=1}^{m_t} \underline{\pi}_i^{t,d} \int_{\Xi_d^t} [-\mathcal{T}_{i,j}^t]^+ dF(\xi^t).$$

At each step of the recursion, upper and lower bounds are found by solving a linear programming problem to maximize and minimize each component of π with respect to the given constraints. Examples of such linear programs are given in Equation 4.4, which is used to find $\overline{\pi}_i^{t,k}$, and Equation 4.5, which is used to find $\underline{\pi}_i^{t,k}$ for $t < T$.

$$\begin{aligned} \max \quad & \pi_i \\ \text{s.t.} \quad & \pi W^t \geq \min_{\xi^{[t]} \in \Xi_k^{[t]}} \{c^t\} - \Psi^{t,k} \\ & \pi \geq 0 \end{aligned} \tag{4.4}$$

$$\begin{aligned} \min \quad & \pi_i \\ \text{s.t.} \quad & \pi W^t \geq \min_{\xi^{[t]} \in \Xi_k^{[t]}} \{c^t\} - \Psi^{t,k} \\ & \pi \geq 0 \end{aligned} \tag{4.5}$$

We can determine bounds for the primal decision variables of Equation 4.1 using forward recursion. Assume $\{x|W^1x \leq h^1\}$ is a bounded polyhedron. In particular, there exists $\underline{x}^{1,1}$ and $\overline{x}^{1,1}$ such that $0 \leq \underline{x}^{1,1} \leq x \leq \overline{x}^{1,1}$ for all x where $W^1x \leq h^1$. For $k = 1, \ldots, K_t$ and $t = 2, \ldots, T$, assume that $\{x^t|\mathcal{T}^t x^{t-1} + W^t x^t \leq h^t\}$ is bounded and there exists $\underline{x}^{t,k}$ and $\overline{x}^{t,k}$ such that $0 \leq \underline{x}^{t,k} \leq x \leq \overline{x}^{t,k}$ for all x where

$$W^t x \leq \max_{\xi^{[t]} \in \Xi_k^{[t]}} \{h^t\} - \Theta^{t,k}$$

and where we define the i^{th} component of $\Theta^{t,k}$ as

$$\Theta_i^{t,k} = \sum_{j=1}^{n_{t-1}} \left[\min_{\xi^t \in \Xi_k^t} \{\mathcal{T}_{i,j}^t\} \right]^+ \underline{x}_j^{t-1,a_t(k)} - \sum_{j=1}^{n_{t-1}} \left[- \min_{\xi^t \in \Xi_k^t} \{\mathcal{T}_{i,j}^t\} \right]^+ \overline{x}_j^{t-1,a_t(k)}.$$

At each step of the recursion, linear programming problems can be used to find the bounds on each variable in a manner similar to that used for the dual decision variable bounds.

The bounds on the optimal dual and primal decision variables can be tightened using complementary slackness conditions. For example, if $\underline{\pi}_i^{t,k} > 0$, then the corresponding constraint

$\mathcal{T}^t_{i,\cdot}x^{t-1} + W^t_{i,\cdot}x^t \leq h^t_i$ must be active for all $\xi^{[t]} \in \Xi^{[t]}_k$ for a particular $k \in \{1,\ldots,K_t\}$ and $t \in \{2,\ldots,T\}$. So, the constraint $W_{i,\cdot}x \geq \min_{\xi^{[t]}\in\Xi^{[t]}_k}\{h^t_i\} - \tilde{\Theta}^{t,k}_i$ must hold, where we define

$$\tilde{\Theta}^{t,k}_i = \sum_{j=1}^{n_{t-1}} \left[\max_{\xi^t\in\Xi^t_k}\{\mathcal{T}^t_{i,j}\}\right]^+ \overline{x}_j^{t-1,a_t(k)} - \sum_{j=1}^{n_{t-1}} \left[-\max_{\xi^t\in\Xi^t_k}\{\mathcal{T}^t_{i,j}\}\right]^+ \underline{x}_j^{t-1,a_t(k)}.$$

This constraint is added to the existing constraints $W^t x \leq \max_{\xi^{[t]}\in\Xi^{[t]}_k}\{h^t\} - \Theta^{t,k}$ when carrying out the forward recursion to calculate $\underline{x}^{t,k}$ and $\overline{x}^{t,k}$. The dual variable bounds $\underline{\pi}^{t,k}$ and $\overline{\pi}^{t,k}$ may be improved in a similar manner when $\underline{x}^{t,k}_j > 0$ for some j.

4.4 Supply Chain Model Formulation

In this chapter we use a MSLP formulation based on a semiconductor supply chain to illustrate the use of our scenario tree generation method. The model uses continuous decision variables and is based on Higle and Kempf (2011). The detailed formulation is given in Equation 4.6. We start with a description of the decision variables at stage t:

$x^t_0 \equiv$ Quantity of goods starting production in period t;

$x^t_P \equiv$ Quantity of finished goods newly produced in period t;

$y^t_j \equiv$ Quantity of demand from j periods ago met in period t;

$w^t_k \equiv$ Quantity of finished goods produced k periods ago and used to meet demand in period t;

$u^t_j \equiv$ Quantity of demand that has been unmet for j periods for $j = 1,\ldots,L$; Demand is lost after L periods;

$s^t_k \equiv$ Quantity of finished goods produced k periods ago for $k = 1,\ldots,R$;

$r^t \equiv$ Quantity of finished goods scrapped after R periods in inventory.

In the first period, the only decision variable is x^1_0. We assume that the work in process inventory levels,$\{x^t_0\}^0_{t=-P+1}$, the unmet demand, $\{u^1_j\}^L_{j=1}$, and the finished goods inventory, $\{s^1_k\}^R_{k=1}$, are known.

Next, we describe the problem parameters used in the the formulation. The deterministic

parameters for the problem include the unit gross profit of meeting demand, c^t_{GP}, the unit cost of scrapping finished goods inventory, c^t, and the unit cost of unmet demand, c^t_j. C is the work in process inventory capacity and is also a fixed, known quantity. We assume that finished goods undergo a testing procedure after the final stage of production and defective goods are removed from inventory. $\lambda^t(\omega^t)$ is a random variable representing the fraction of goods that pass in stage t. $d^t(\omega^t)$ is a random variable representing the quantity of demand in period t.

$$\max \quad E\left[\sum_{t=2}^{T-1}\sum_{j=0}^{L}\sum_{k=0}^{R}\left(c^t_{GP}y^t_j(\omega^{[t]}) + c^t r^t(\omega^{[t]}) + c^t_j u^t_j(\omega^{[t]})\right)\right] \tag{4.6}$$

$$\text{s.t.} \qquad x^t_P = \lambda^t(\omega^t)x^{t-P}_0, \text{ for all } t > 1 \tag{4.7}$$

$$\sum_{i=0}^{P-1} x^{t-i}_0 \leq C, \text{ for all } t \tag{4.8}$$

$$u^t_0 = d^t(\omega^t) - y^t_0, \text{ for all } t > 1 \tag{4.9}$$

$$u^t_{j+1} = u^{t-1}_j - y^t_{j+1}, \text{ for all } t > 1 \text{ and } j < L \tag{4.10}$$

$$s^t_0 = x^t_P - w^t_0, \text{ for all } t > 1 \tag{4.11}$$

$$s^t_{k+1} = s^{t-1}_k - w^t_{k+1}, \text{ for all } t > 1 \text{ and } k < R \tag{4.12}$$

$$r^t = s^t_R, \text{ for all } t > 1 \tag{4.13}$$

$$x^1_0, x^t_p, y^t_j, w^t_k, u^t_j, s^t_k, r^t \geq 0, \text{ for all } j, k, p, \text{ and } t \tag{4.14}$$

Equation 4.7 gives the quantity of newly produced finished goods as a function of the quantity of goods that began production P periods ago. Equation 4.8 gives the constant work in process inventory constraint. Equations 4.9-4.10 give the balance equations for unmet demand from current and past customer orders. Equations 4.11-4.13 give the balance equations for finished goods inventory for goods produced in the current and previous periods.

Applying the methods of Section 4.3.4 to the formulation given in Equation 4.6, we cannot find finite bounds for all the dual decision variables. Although every instance of the problem has a finite optimal solution, the feasible region of the dual formulation is not bounded. So,

we develop bounds on the values of the decision variables that cannot be determined from the constraints alone. First, let $\pi_C^t \geq 0$ be the dual variable associated with Equation 4.8. This quantity represents the profit associated with increasing the WIP capacity by one unit in period t. Since this change can increase total throughput of the production system by at most one unit, the resulting change in profit can be no more than the gross profit for selling one additional item minus the costs associated with backordered demand. An upper bound on π_C^t is therefore $\max\{c_{GP}^t\} - (L+1)\max\{c_j^t\}$. This is also an upper bound for the dual variable $\pi_{x_P}^t$, associated with Equation 4.7, since adding one unit of inventory at the first production stage in period $t - P$ can increase total production by at most one unit. We also see that c^t is an appropriate lower bound for both π_C^t and $\pi_{x_P}^t$, since increasing WIP capacity or total production by one unit may result in having to scrap that unit. By similar reasoning, we see that $(L+1)\max\{c_j^t\}$ and $\max\{c_{GP}^t\} - (L+1)\max\{c_j^t\}$ are lower and upper bounds, respectively, on the dual variables $\pi_{u_j}^t$ associated with Equations 4.9-4.10. This corresponds to the maximum change in profit due to increasing demand by no more than one unit. The lower and upper bounds for $\pi_{s_k}^t$, the dual variables associated with Equations 4.11-4.12, are 0 and $\max\{c_{GP}^t\} - (L+1)\max\{c_j^t\}$ respectively. This corresponds to the maximum change in profit due to increasing the supply of finished goods by no more than one unit. Finally, we point out that the dual variable associated with Equation 4.13 is bounded above by zero, since no profit can be generated from scrapping an additional unit of inventory. In other words, we could replace the constraint with $r^t \geq s_R^t$.

4.5 Numerical Experiments

In this section, we summarize results based on numerical experiments to compare all four methods covered in Section 4.3: Monte Carlo sampling, moment matching, EVPI importance sampling, and the new scenario generation algorithm. All experiments are conducted on a Windows 7 PC with an 8 core 2.93 GHz Intel i7 Processor and 16 GB of RAM. All algorithms are implemented in the C++ programming language with Matlab scripting used for calling and evaluating the algorithms. The C Callable Library for IBM ILOG CPLEX 12 is used to solve

the deterministic equivalent linear programming problems for scenario trees.

4.5.1 Problem Instances

A total of 24 test instances of the formulation given in Section 4.4 are used for the experiments. In all the test instances $P = 2$ and $L = 2$. The initial conditions at the beginning of the first period are those for an empty production system with no finished goods, work in process inventory, or backordered demand. Two types of demand distributions are used: the uniform distribution, $U[a, b]$, and the truncated normal distribution, $N[\mu, \sigma]$, where the demand is never less than zero or greater than $\mu+6\sigma$. Here, μ and σ are the parameters prior to truncation. λ^t is uniformly distributed, $U[0.95, 1]$. A discount factor of 0.99 for future earnings per period is used. See Table 4.1 for the parameters used in each test instance. We use a labelling scheme for each test instance with a prototype label $TXCZ$. Here, the letter T represents the number of periods, either the number 10 or 15. The letter X represents either the letter S if the salvage cost, c^t, is higher or P if the penalty cost for unmet demand, c_j^t, is higher. The letter C represents the capacity, which is either the number 5000 or 1000. The letter Y represents the demand distribution, which is either U for the uniform distribution, $N500$ for the normal distribution with mean 500, or $N1000$ for the normal distribution with mean 1000.

Considerable variation is possible in implementing the Monte Carlo, moment matching, and EVPI algorithms. In the case of Monte Carlo sampling and the method of matching moments, the tree structure does not change during execution of the algorithm. The structure is chosen so that the branches per node, starting with the root node, are 7, 5, 4, 4, 2, and 2 for the first six levels of the tree. All subsequent levels have one branch per node. The initial scenario tree for the EVPI algorithm has 7, 2, 2, 2, 2, and 2 branches per node at the first six levels and one branch per node at all subsequent levels. Both initial tree structures are non-increasing sequences of numbers, and were chosen on a trial and error basis during preliminary testing as the first tree with the time required to solve corresponding MSLPs not too long. The initial scenario tree for the EVPI algorithm should have smaller numbers than those for the Monte

Table 4.1: MSLP test instances

Instance	Label	T	$c^t_{GP}/c^t/c^t_j$	d^t	C
1	10S5000U	10	10/5/1	U[1,500]	5000
2	10S1000U	10	10/5/1	U[1,500]	1000
3	10S5000N500	10	10/5/1	N[500,100]	5000
4	10S1000N500	10	10/5/1	N[500,100]	1000
5	10S5000N1000	10	10/5/1	N[1000,250]	5000
6	10S1000N1000	10	10/5/1	N[1000,250]	1000
7	10P5000U	10	10/1/5	U[1,500]	5000
8	10P1000U	10	10/1/5	U[1,500]	1000
9	10P5000N500	10	10/1/5	N[500,100]	5000
10	10P1000N500	10	10/1/5	N[500,100]	1000
11	10P5000N1000	10	10/1/5	N[1000,250]	5000
12	10P1000N1000	10	10/1/5	N[1000,250]	1000
13	15S5000U	15	10/5/1	U[1,500]	5000
14	15S1000U	15	10/5/1	U[1,500]	1000
15	15S5000N500	15	10/5/1	N[500,100]	5000
16	15S1000N500	15	10/5/1	N[500,100]	1000
17	15S5000N1000	15	10/5/1	N[1000,250]	5000
18	15S1000N1000	15	10/5/1	N[1000,250]	1000
19	15P5000U	15	10/1/5	U[1,500]	5000
20	15P1000U	15	10/1/5	U[1,500]	1000
21	15P5000N500	15	10/1/5	N[500,100]	5000
22	15P1000N500	15	10/1/5	N[500,100]	1000
23	15P5000N1000	15	10/1/5	N[1000,250]	5000
24	15P1000N1000	15	10/1/5	N[1000,250]	1000

Carlo algorithm in order to complete more than one iteration in the same amount of time.

For the Method of Matching Moments, we used Gaussian quadrature nodes and weights to generate probabilities and random variable outcomes associated with each branch of the scenario tree (Miller and Rice, 1983). When $\mathcal{O}$ outcomes are generated, this approach will generate the outcomes and associated probabilities such that the first $2\mathcal{O}-1$ moments match those of the continuous distribution. For example, since we produce 7 outcomes for the first period demand, the first $2 \cdot 7 - 1 = 13$ moments will match the moments of the continuous demand distribution. We use the average values for λ^t at each stage, which is equivalent to matching the first moment for these random variables. Although we could have chosen to match fewer moments of the demand distribution in order to match more moments of λ^t, variation in demand tends to have a greater influence on the problem than variation in production.

Several of the parameters for controlling the EVPI algorithm were fixed. We followed the algorithm described in Dempster (2006), but within this description there is room for variation. For our implementation, if the local EVPI of a node falls below 0.1% of the root node EVPI after three successive resamplings, all scenarios involving that node are collapsed into a single scenario. New branches are never added to the root node. Whenever new branches are added to a node at a certain level of the tree, the number of branches per node at subsequent levels of the tree are in accordance with the initial tree structure. An exception to this rule is made when all subsequent levels of the tree have only one node per branch. When adding a branch to a node in this situation, we add two branches to the first child of the node. Otherwise, there would be no opportunity for more than one branch per node beyond level 6 of the tree. We did not attempt to tune these parameters to the test instances.

Next we describe the termination criteria used for the experiments. Short of adjusting the size of the scenario tree, there are no parameters to control the runtime of the Monte Carlo or moment matching algorithms. Of course, predicting run time based on tree size alone is difficult. In situations where a limited time is available, we would expect a user to generate smaller scenario trees than necessary in practice to avoid reaching the time limit without generating

a feasible first stage solution. The EVPI algorithm was typically only able to complete one iteration in the same time as the Monte Carlo approach. Since the EVPI algorithm continues to refine the scenario tree until one of the termination criteria is met and a feasible first stage solution is available at every iteration, we would expect the typical runtime for this algorithm to be longer than the Monte Carlo algorithm. So, we allowed the EVPI algorithm to continue to the end of the second iteration in our experiments. Finally, we observed that the new scenario generation algorithm was typically able to complete tens of iterations in the runtime required for the EVPI algorithm. We allowed the new algorithm to complete the last iteration after reaching the total runtime required for the EVPI algorithm.

4.5.2 Rolling Horizon Simulation

Before describing how we measure the effectiveness of the three algorithms presented in Section 4.3, we describe how such algorithms might be used in practice. Solving the deterministic equivalent linear program associated with a T-stage scenario tree will always produce a set of feasible first stage decision variables. Unfortunately, the random outcomes that are revealed after the first stage do not necessarily correspond to any of the outcomes selected for the scenario tree. Hence, the decision variables associated with subsequent stages in the deterministic equivalent linear program may not be appropriate or even feasible. At the start of the second stage, we must generate a new $T-1$-stage scenario tree with the second stage of the previous tree becoming the first stage of the new tree. The state of the system, including initial inventory levels, should be updated prior to generating this scenario tree. In practice, this process is repeated for all subsequent periods in the planning horizon.

To evaluate the relative performance of the scenario generation methods, we make use of the rolling horizon simulation method described in Kouwenberg (2001). A diagram illustrating this approach is shown in Figure 4.3. The process begins by calling the algorithm to generate a scenario tree and solving the corresponding deterministic linear program. This produces a feasible first stage decision for production starts, x_0^1. Next, we simulate the demand and production

yields for the first period using a random number generator. Although we use the same probability distributions to generate these quantities as those used to construct the scenario tree, both processes are carried out independently. We update the finished goods inventory (FGI), work in process (WIP) inventory, and unmet demand levels based on the quantities simulated after the first stage. The process is repeated when we call the algorithm again to generate a scenario tree for the new planning horizon starting at stage 2 instead of stage 1. This process continues until we reach the final stage of the planning horizon.

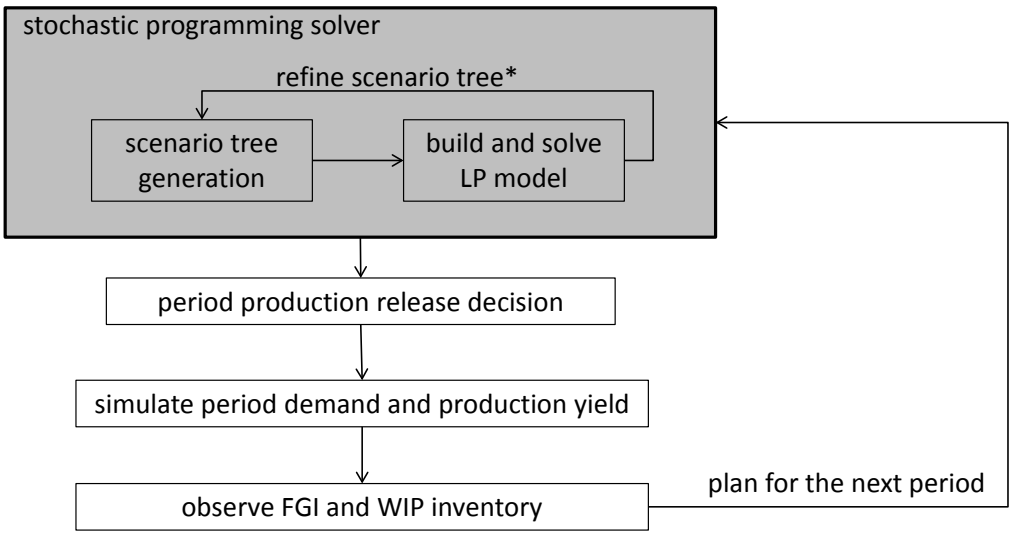

Figure 4.3: Flow chart for the rolling horizon simulation approach used to evaluate effectivness of the algorithms

The measure of effectiveness for each algorithm is the average net profit. In one simulation experiment, the total net profit is calculated. Due to computational requirements, we limited the experiments to 100 replications for each test case and algorithm. The average net profit is obtained by averaging the total net profit for all 100 replications. We also report the standard deviation, σ, of the net profit for these replications. For a given replication of the simulation

experiment, common random numbers were used for each algorithm. This was done to reduce the variance when comparing the effectiveness of the different algorithms. We used the Matlab *lillietest* function to verify that the paired differences in total net profit between each pair of algorithms was normally distributed. This allowed us to test if the difference in average net profit between any pair of algorithms was statistically significant using the paired t-test at the 5% level of significance.

4.5.3 Comparison of Methods

Table 4.2 shows deterministic bounds on the optimal solution for the MSLP model. These bounds are generated from the new algorithm, and cannot be generated from the other algorithms. These bounds are calculated during the first iteration of the first replication of the rolling horizon simulation. In the first iteration, the scenario tree represents outcomes starting with the first period. So, a full rolling horizon simulation is not needed to produce these bounds, since they are generated upon completion of the algorithm for the start of the planning horizon. No random numbers were generated when calculating these bounds, and unlike the full rolling horizon simulation itself, multiple replications of the calculation were not needed. In other words, these values are based on solving a single MSLP instance.

The sequential algorithm results in Table 4.2 represent bounds on the total average profit from the first period to the last period in the planning horizon. Because a rolling horizon simulation was not used, the run times for these results were less than the average run times for a single replication of the sequential algorithm rolling horizon simulation results in Table 4.3. We also performed a limited set of experiments where we allowed the new sequential algorithm to run for an hour for each case. This was done to see if the bounds on the optimal objective function value would improve. We only saw a small improvement, despite an order of magnitude increase in the number of scenario tree nodes.

The results of the rolling horizon simulation experiments are shown in Table 4.3. The average runtime and 95% confidence intervals for the average net profit are shown for each of the four

Table 4.2: Bounds on z^* obtained from the new sequential bounding algorithm

Case	Label	z^* Lower Bound	z^* Upper Bound
1	10SU5000	-40241	23292
2	10SU1000	12067	23256
3	10SN5005000	-3845	46786
4	10SN5001000	37933	42897
5	10SN10005000	53229	93559
6	10SN10001000	20048	32314
7	10PU5000	-98483	22377
8	10PU1000	3505	22284
9	10PN5005000	-54725	44806
10	10PN5001000	22037	33806
11	10PN10005000	10191	89599
12	10PN10001000	-52168	-21993
13	15SU5000	-34387	34284
14	15SU1000	19712	34278
15	15SN5005000	14989	68726
16	15SN5001000	52449	60603
17	15SN10005000	88714	137423
18	15SN10001000	22490	42270
19	15PU5000	-100348	33371
20	15PU1000	639	33312
21	15PN5005000	-41940	66746
22	15PN5001000	33013	52448
23	15PN10005000	33477	133463
24	15PN10001000	-89217	-40597

algorithms in order of the Monte Carlo (MC), Moment Matching (MM), EVPI, and the new sequential bounding algorithm (NEW). The new algorithm has the highest average net profit in 9 out of the 24 cases. The Monte Carlo algorithm had the highest average net profit in 9 different cases. The EVPI and Moment Matching algorithm each achieved the highest average net profit in 3 out of 24 cases. However, for each test case and pair of algorithms we did not always detect a statistically significant difference in net profit. Those cases where a difference was detected at the 5% level are shown with a connecting line in Figure 4.4. Here, we present results for the uniform distribution (U), normal distribution with mean 500 (N500), and normal distribution with mean 1000 (N1000).

Next, we describe the statistical analysis of the data associated with Table 4.3 that was used to generate Figure 4.4. In all cases except Case 21 we failed to reject the null hypothesis that the paired differences are normally distributed. We also performed a test for normality of the total net profit values over all replications for each case and algorithm. We were unable to reject the null hypothesis that the net profit values are normally distributed in at least 20 of the 24 cases for each of the four algorithms at the 5% level of significance. We therefore assume that these values are normally distributed and adding and subtracting 1.96 standard deviations to the mean values yields the 95% confidence bounds on the average net profit for each algorithm shown in Table 4.3.

Comparing the deterministic bounds in Table 4.2 with the 95% confidence bounds on the average net profit for each algorithm in Table 4.3, we see that in most cases the deterministic lower bound for z^* is lower than the 95% confidence lower bound. However, in Cases 16, 22, and 24 this is not true. In several cases the deterministic lower bounds are larger than the 95% confidence upper bounds. This occurs with Cases 4, 6, and 10. When this occurs, we can be reasonably confident that the algorithm in question does not generate an optimal solution for that instance.

From Figure 4.4, only half of the 15-period horizon cases have a significant difference between any of the algorithms. This is due to the high variance associated with these problems. From

Table 4.3, we see that the width of the confidence interval is almost always wider for any particular algorithm and case with a 15-period horizon compared to the same case with 10 periods. There are only three situations, out 48 possible, where this is not true. Wider confidence intervals are expected, since the mean and variance of net profit for corresponding periods in the planning horizon should be comparable. In other words, we would expect the same mean and variance in net profit for the first period of a 10-period horizon as we would for the first of a 15-period horizon. Problems with more periods will therefore have a higher average and variance for net profit when aggregated over all periods. Unfortunately, every algorithm is outperformed by another algorithm for at least one of the 15-period horizon problems. This suggests using the algorithm that is easiest to implement, the Monte Carlo approach, in these cases. However, when the variance in demand, production capacity, and penalty cost for unmet demand are all relatively high, then one might consider using the new algorithm, since it outperforms the Monte Carlo approach in that case. We will offer an explanation for the new algorithm's success for this case below.

We also observe that the new algorithm is outperformed by at least one other algorithm in 5 out of 12 of the 15-period cases. One explanation follows from the fact that the new algorithm uses the mean values of intervals for the random variable support partition representing demand for each period. Miller and Rice (1983) showed that any such scheme for approximating a probability distribution will underestimate the variance of the true distribution. Underestimating demand variance tends to result in insufficient safety stock levels, unless the details of the problem instance discourage safety stock from accumulating. For example, if the cost associated with excess inventory is high relative to the penalty cost for unmet demand, then we would expect the effect of underestimating demand variance to be reduced. Similarly, since excess inventory can not be sold after the last period in the horizon, safety stock is not as important for short horizon problems. Thus, the new algorithm tends to perform better for shorter horizon problems with a relatively high excess inventory cost. This explains why the new algorithm is never outperformed by another algorithm in the second row of Figure 4.4, and

never outperforms another algorithm in row 3.

Comparing cases across the first and last row of Figure 4.4, the new algorithm is outperformed by the moment matching approach when demand is normally distributed with mean 500 and production capacity is high. However, the new algorithm has the highest average net profit, and outperforms at least one other algorithm at the 95% significance level, when the demand is normally distributed with mean 1000. The explanation for this phenomenon appears to be due to the new algorithm's tendency to underestimate demand variance in combination with other factors. Since new production starts require two periods to become finished goods, the new algorithm has a tendency to produce insufficient quantities to meet demand. Once demand is realized and this becomes apparent, the new algorithm must increase the level of production to meet the unmet demand as well as future demand. Production yields are uncertain; so, this has a tendency to make production quantities higher than they would be for other algorithms. When demand variance is sufficiently high, this effect is cancelled out by the conflicting tendency of the new algorithm to build insufficient safety stock as a result of underestimating demand variance. However, when demand variance is low, the effects of yield uncertainty dominate, and safety stocks become higher than necessary.

Table 4.3: Results of the numerical experiments for Monte Carlo (MC), moment matching (MM), EVPI, and new sequential bounding (NEW) algorithms based on the rolling horizon simulation

Case	Label	Avg. Runtime (s)				Avg. Net Profit (95% confidence interval)			
		MC	MM	EVPI	NEW	MC	MM	EVPI	NEW
1	10SU5000	12.2	7.0	19.3	20.0	19991 ± 8717	20393 ± 9111	19579 ± 9925	21568 ± 8282
2	10SU1000	11.7	7.1	19.4	13.6	19685 ± 8947	19938 ± 8094	20232 ± 8495	21241 ± 9197
3	10SN5005000	12.8	7.1	20.1	20.5	42535 ± 7227	41858 ± 7685	41129 ± 8597	40447 ± 7156
4	10SN5001000	12.1	11.4	19.2	13.1	29924 ± 2762	29819 ± 2313	29653 ± 2630	29831 ± 2928
5	10SN10005000	13.3	7.9	24.0	18.4	85391 ± 20452	83532 ± 21248	84505 ± 21051	87254 ± 19039
6	10SN10001000	10.2	11.9	18.4	12.4	15195 ± 4940	15351 ± 4562	14774 ± 4224	15371 ± 4223
7	10PU5000	11.4	7.6	20.1	20.6	17671 ± 7779	18839 ± 7022	19032 ± 7295	18082 ± 7736
8	10PU1000	10.0	7.0	19.2	13.3	18363 ± 7709	18101 ± 7101	17034 ± 7865	18220 ± 7056
9	10PN5005000	12.4	6.7	19.6	20.0	37160 ± 5915	38071 ± 5497	37747 ± 4924	38147 ± 4885
10	10PN5001000	11.3	10.0	18.1	12.9	3313 ± 12655	2059 ± 13604	1566 ± 11962	1940 ± 11979
11	10PN10005000	12.5	7.5	21.0	16.0	73905 ± 12703	75787 ± 11038	73748 ± 13631	77068 ± 14710
12	10PN10001000	10.1	11.7	16.5	11.3	-67877 ± 21770	-70302 ± 22218	-71102 ± 23559	-70179 ± 23691
13	15SU5000	32.8	18.0	44.3	45.7	31158 ± 11447	31665 ± 10760	31432 ± 10227	32662 ± 11248
14	15SU1000	33.4	19.0	45.6	47.3	31856 ± 10664	31746 ± 12470	31065 ± 10698	30986 ± 11441
15	15SN5005000	35.7	18.6	81.5	82.9	63847 ± 7884	64346 ± 8456	64401 ± 8614	62823 ± 8924
16	15SN5001000	39.9	42.8	85.8	53.6	52779 ± 3510	52067 ± 3618	52690 ± 3947	49550 ± 5125
17	15SN10005000	38.4	20.6	86.1	72.9	128504 ± 23627	129020 ± 20868	127747 ± 20987	127930 ± 20760
18	15SN10001000	34.0	36.2	67.8	65.5	30688 ± 5337	30471 ± 6057	30414 ± 4335	29951 ± 5419
19	15PU5000	31.3	18.8	46.2	47.3	29066 ± 9286	28889 ± 8804	29276 ± 10542	28966 ± 10277
20	15PU1000	28.3	19.9	43.1	42.4	28041 ± 9351	29029 ± 8019	28731 ± 7640	28899 ± 9687
21	15PN5005000	33.8	18.1	52.5	53.6	59592 ± 5922	60260 ± 6647	59810 ± 6422	59382 ± 5223
22	15PN5001000	42.5	35.4	116.9	85.2	14643 ± 18306	14316 ± 21046	13944 ± 20597	7829 ± 19554
23	15PN10005000	35.1	19.5	61.5	53.8	118451 ± 14460	119999 ± 15293	118756 ± 15164	121308 ± 16929
24	15PN10001000	32.7	36.2	53.8	54.4	-96784 ± 26418	-96956 ± 30103	-97627 ± 26916	-95656 ± 30120

4.6 Conclusions

The new sequential algorithm proposed in this chapter provides deterministic bounds on the optimal objective function value after solving a single instance of the MSLP. These bounds are generated without replications or the need for a rolling horizon simulation. We see that the bounds are considerably wider than those presented in Chapter 3 for a two-stage stochastic program. Since the size of an MSLP instance depends exponentially on the number of stages, we would expect exponentially longer run times for MSLPs to achieve the same level of accuracy as two-stage stochastic programs.

The algorithm proposed in this chapter was compared to three other algorithms: Monte Carlo sampling, moment matching, and EVPI importance sampling. This comparison was conducted using 100 replications of a rolling horizon simulation. The 95% confidence bounds generated for each algorithm and instance are wide, but with a few exceptions, tighter than the deterministic bounds that were generated with the new sequential bounding algorithm. However, a simulation approach means that the run time required to generate these confidence bounds was roughly 100 times greater than that required to generate the deterministic bounds.

Based on the experiments of this chapter, we see that there is always at least one case where any algorithm performs better than any other algorithm at the 95% level of significance. Thus, there is no single algorithm that dominates the others and the best choice of algorithms depends on the problem instance. The new sequential algorithm did obtain the highest average net profit as often as any of the other algorithms, and there are many cases where it performs better than the other algorithms in a one-on-one comparison. However, there are many cases where the new sequential algorithm did not perform as well in a one-on-one comparison. So, the new sequential algorithm appears to be competitive with existing approaches. The sequential algorithm also has the benefit of requiring less expert judgement and experience with the specific problem. In situations where an initial scenario tree structure can be chosen appropriately, we feel that the Monte Carlo approach is probably a reasonable approach for practical applications, since it is much easier to implement.

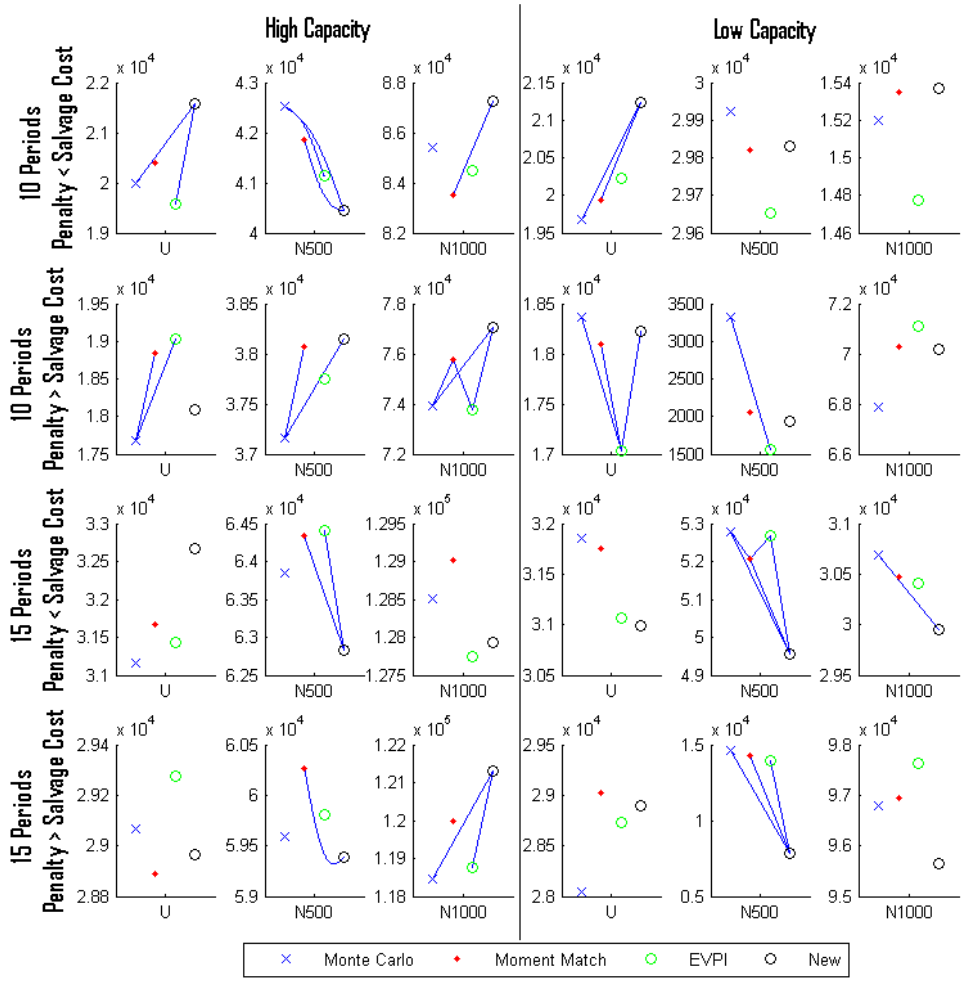

Figure 4.4: Average net profit from the rolling horizon simulation for each of the four algorithms. Results are shown for a total of 24 test cases, with 3 demand distributions, 2 cost structures, 2 capacity levels, and 2 planning horizon durations. A line connecting the average net profit between two different algorithms indicates that the difference is statistically significant at the 95% confidence level.

Areas of future work include finding suitable conditions where strong duality holds and Proposition 1 holds true. We used 100 replications in simulation experiments to compare the effectiveness of the various algorithms for each case. In many cases, we were unable to find a statistically significant difference in the effectiveness of the algorithms. The reason for this is clear when we examine the results in Table 4.3. The confidence intervals for total net profit are wide. More replications of the rolling horizon simulation would improve these confidence bounds, and allow us to distinguish between algorithms, but the additional computation time would be very long. Since replications of the simulation can be run independently, it may be possible to more clearly distinguish between the algorithms if a parallel computing environment with a very large number of processors is utilized.

Chapter 5

A Comparison of Multi-stage Stochastic Programming and Chance Constrained Programming for Semiconductor Manufacturing

5.1 Introduction

In the semiconductor manufacturing industry, successful production planning depends on accurate models of both production lead times and uncertain demand. Production lead times are dependent on the loading of the factory: the work in process inventory (WIP) relative to available capacity. Demand forecasts are updated regularly and evolve over time as customers adjust the quantity and timing of their orders.

Correlation plays an important role in forecasting demand for production planning subject to capacity constraints. If forecasted demand is negatively correlated between periods, then an increase in forecasted demand in one period is followed by a decrease in the next. Fewer resources need to be diverted for producing safety stock in these situations compared to the same

increase in forecasted demand with zero correlation. On the other hand, if demand is positively correlated, then safety stock levels will need to be higher to maintain adequate service levels for capacitated production planning.

Our goal is to determine the number of orders to release for production in the factory to achieve the lowest possible average total cost. In this chapter, we explore the following question: is a chance constrained model (CCM) or multi-stage stochastic linear program (MSLP) more appropriate for achieving this goal? Except in rare situations, MSLPs are impossible to solve exactly. So, approximate solution approaches must be used. We also wish to know if average total cost varies significantly for different solution approaches for MSLPs.

In this chapter, we review a CCM and MSLP for semiconductor manufacturing. In particular, we review models that consider the dependence of factory load on lead times as well as the evolution of correlated demand forecasts as customer orders are modified over time. We present an MSLP formulation based on Norouzi (2012, Chap. 4). We compare several MSLP solution approaches and compare it with an existing CCM and MSLP solution approach. The CCM model can be viewed as an approximation of the MSLP. In particular, an exact solution approach for the MSLP would yield solutions with lower total average cost than the CCM model.

In our numerical experiments, we solve the MSLP presented in this chapter using three approaches: moment matching, Monte Carlo sampling and the sequential bounding approach outlined in Chapter 4. The Monte Carlo and sequential bounding approaches that we use are general MSLP solution approaches that can be applied to a wide variety of models, including those not related to semiconductor manufacturing. We also consider variations of the sequential bounding approach. We conclude the chapter with insights and areas for future research to improve the general approach of sequential bounding.

This chapter is organized as follows. In Section 5.2, we review existing models of demand forecast evolution and load dependent lead times and explain how this chapter contributes to the existing literature. In Section 5.3 we cover the various models and approaches used in

our computational experiments, including assumptions and approximations. In Section 5.4, we present numerical experiments for the various models and solution approaches. In Section 5.5, we provide conclusions and areas for future research.

5.2 Literature Review

In addition to modeling correlation in demand forecasting, the Martingale Model of Forecast Evolution (MMFE) describes the evolution of forecasts over time in a manner consistent with a wide variety of forecasting models (see Heath and Jackson, 1994). The model assumes long-term forecasts are known, but it does not specify how this input data is generated. So, expert judgement, or any other long-term forecasting technique, can be used.

Graves et al. (1986) developed the additive MMFE to model the evolution of demand forecasts. Here, information revealed from one period to the next can be modelled as a vector of random variables representing changes in demand forecasts for each future period. This vector also includes the difference between forecasted and actual demand for the most recently observed period. Heath and Jackson (1994) extend the model so that each period's random variable vector is independent from the others, but the random variables within each vector have a correlated multivariate normal distribution. Although the random variable vectors are independent, this model allows for the distribution governing demand observed in one period to depend on previously observed demand quantities.

Next, we provide a more detailed mathematical description of the MMFE of Heath and Jackson (1994). Let $D_{s,t}$ be the forecasted demand for some future period t at the end of period $s \leq t$. If $s = t$, then $D_{s,t}$ is the demand observed to have taken place in period s. Forecasts are not updated for periods that are more than H periods in the future. So, if $t > s + H$, we assume $D_{s,t} = \mu$. In this paper, we assume this quantity is the same for all periods beyond the next H periods in the future, but this is not essential. For a given period t where $s < t \leq s+H$, $D_{s,t}$ will be adjusted as s increases and we get closer to period t. The equation governing this

evolution is:

$$D_{s+1,t} = D_{s,t} + \xi^t_{t-s}, \text{ for } t = s+1, \ldots, s+H$$

Here, ξ^t_i, for $i = 1, \ldots, H$ are random variables that are assumed to be distributed according to a multivariate correlated normal distribution with mean zero and variance-covariance matrix Σ. The vector whose components consist of these random variables is denoted ξ^t. We assume that ξ^t and ξ^{t+1} are pairwise independent.

The problem of workload-dependent lead times in production planning has been studied by many authors. Lead times have a nonlinear dependence on WIP, which in turn depends on the history of factory work orders. The majority of models in the literature either treat lead times as exogenous random variables, ignore the effects of WIP on lead time, or require iterative solution of simulation and optimization models. We refer to Missbauer and Uzsoy (2011) and the references therein for a review of such models.

Clearing functions are defined to be the expected output quantity per period, X, of a production resource as a function of the WIP, W, at that resource. So, $X = X(W)$. They were first proposed by Graves (1986) to model workload dependent lead times and are easily incorporated into linear programs and MSLPs. Although clearing functions do not explicitly include lead time, from Little's Law, $X(W) = \frac{W}{L(W)}$, where $L(W)$ is the average lead time as a function of W. Assuming the primary source of variation in output is attributable to the WIP level, we treat the average output as the actual output for each period. Clearing functions can be determined empirically, and their effectiveness has been demonstrated in several studies (Kacar, 2012, Chapt. 1).

Citing few production planning approaches that consider forecast evolution and workload-dependent lead times, Norouzi (2012, Chapt. 4) develops two solution approaches. The first approach is based on a chance constrained model, where a simple base stock policy is assumed to be implemented. Although the chance constraint can be used to obtain a pre-specified level of service, the service level is viewed as a parameter, along with the base stock level. The lowest base stock level and highest service level consistent with the lowest cost are determined,

assuming the probability distribution of demand and planned shortfall from the base stock level is known. The cost and problem structure is used to find an approximate deterministic equivalent to the chance constraint. In particular, a discrete-time, continuous state, and high traffic queue approximation is used, based on backorder costs being high relative to holding costs, so that high service levels result in lower overall costs. The remaining random demand parameters in the chance constrained model are replaced with their mean values, yielding a linear program approximation. The second approach is based on a multi-stage stochastic program model. Although demand outcomes for this model can be generated in accordance with the MMFE, forecasted demand quantities were not used in the solution approach used for computational experiments. A scenario tree was generated (see Chapter 2) using the moment matching approach of Miller and Rice (1983). The structure of the scenario tree was chosen with five branches per node, and the same unconditional demand distribution was used to generate outcomes for every level and branch of the tree. Both solution approaches can be viewed as approximate methods for solving the same MSLP model.

This chapter contributes to the literature in several ways. First, we reformulate the MSLP of Norouzi (2012, Chapt. 4) without reference to a scenario tree structure. This formulation is convenient for adaptive scenario tree generation approaches for solving MSLPs, like the sequential bounding approach of Chapter 4, where the scenario tree structure changes during the course of the algorithm and is independent of the model. We use the general MSLP solution approach of Monte Carlo sampling and sequential bounding developed in Chapter 4 to solve instances of this problem. These approaches do not make problem or cost structure assumptions, nor do they assume a base stock policy is implemented. These approaches also do not replace random demand with mean values or replace the MMFE with independent and identically distributed demand in every period. We present computational experiments comparing these approaches to the approaches of Norouzi (2012, Chapt. 4). We also compare several variations of sequential bounding on the basis of average total cost. We conclude with some insights and future areas of research to improve the general solution approach.

5.3 Production Planning Models

In this section, we provide the chance constrained model and a reformulation of the MSLP model of Norouzi (2012, Chap. 4) for production planning. In both models, the objective of the decision maker is to determine the quantity of material to release into a factory to minimize the average total cost of production. Demand for finished goods will be back ordered at a unit cost per period until it is met. Excess inventory incurs a unit holding cost per period.

We define stage t to be one of a consecutively numbered set of points in time when the decision maker determines the quantity of material to release for $t \in \{0, \ldots, T+1\}$. Stage 1 is the current point in time when a release quantity must be determined, and stage 0 is the point in the past where the most recent release quantity was determined. We define period t to be the span of time between stage t and stage $t+1$ for $t = 0, \ldots, T$.

In the models of this section, we use a piecewise linear concave clearing function (see Missbauer and Uzsoy, 2011) to model production capacity as a function of work in process (WIP) inventory. A prototype 3-point clearing function is shown in Figure 5.1. We assume α_i and β_j are defined in accordance with this figure for $i, j = 1, 2$. We require $\alpha_1 \geq \beta_1$ in order to ensure that the quantity produced never exceeds available WIP. In order to maintain a concave clearing function, we require the slopes of each linear segment to be decreasing: $\frac{\beta_1}{\alpha_1} \geq \frac{\beta_2 - \beta_1}{\alpha_2 - \alpha_1} \geq 0$, where we assume $\alpha_2 > \alpha_1 > 0$. In the experimental results presented in Section 5.4, we use the special cases of clearing functions with one and two line segments.

5.3.1 Chance Constrained Model

Chance constraints were introduced by Charnes et al. (1958) and Charnes and Cooper (1959). They are constraints that must be satisfied at or above some given probability. For example, we may specify that the quantity of backorders in a given period should be equal to zero with probability greater than or equal to 90%. Such constraints have the advantage of being easy to understand from a decision maker's perspective. In general they involve multidimensional integrals that may be impossible to express in a closed form expression. However, after mak-

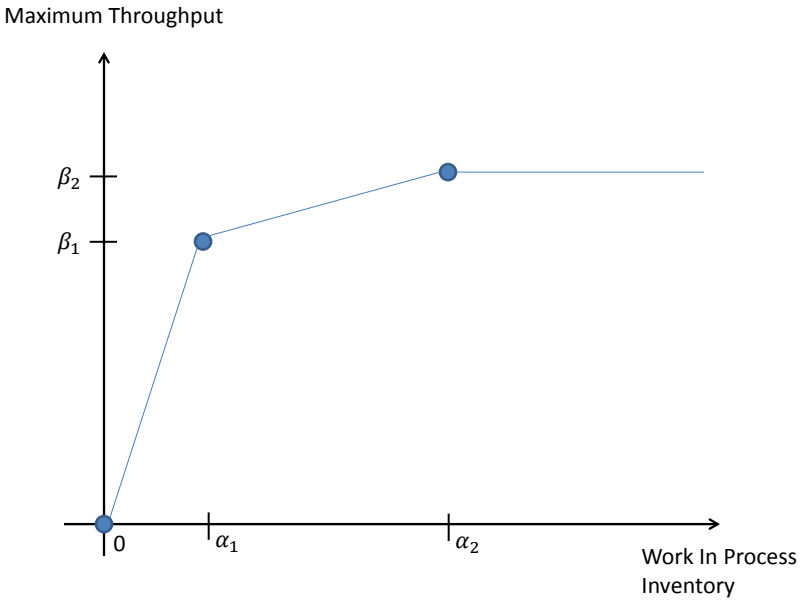

Figure 5.1: A piecewise linear clearing function defined by three points $(0, 0)$, (α_1, β_1), and (α_2, β_2).

ing suitable approximations, the chance constrained model in this section is translated into a deterministic linear program.

The chance constrained model has the following decision variables associated with stage t:

$W^t \equiv$ Quantity of WIP inventory, measured in units of finished goods.

$I^t \equiv$ Quantity of finished goods inventory (FGI). (defined for $t > 1$)

$B^t \equiv$ Quantity of back ordered demand. (defined for $t > 1$)

$R^t \equiv$ Quantity of released material, measured in units of finished goods.

$X^t \equiv$ Quantity of finished goods to produce in the next period.

$Y^t \equiv$ Planned shortfall quantity.

We assume that the initial quantities, I^1 and B^1 are known. The following cost parameters are also assumed to be known:

$h_W \equiv$ Unit cost per period for WIP inventory.

$h_I \equiv$ Unit cost per period for FGI.

$b \equiv$ Unit cost per period for back orders.

We assume a base stock policy is implemented with newly produced goods $X^t = S - Y^t - I^t + B^t$, where S is the base stock level. We define D^t to be a random variable representing the quantity of demand for period t, based on the information that we have at the current stage. The chance constrained formulation follows:

$$\begin{aligned}
\min \quad & E[\textstyle\sum_{t=1}^{T} \left(h_W W^t + h_I I^t + bB^t\right)] \\
\text{s.t.} \quad & X^t \leq \tfrac{\beta_1}{\alpha_1} W^t, \text{ for } t = 1, \ldots, T+1 && (5.1) \\
& X^t \leq \tfrac{\beta_2 - \beta_1}{\alpha_2 - \alpha_1}(W^t - \alpha_1) + \beta_1, \text{ for } t = 1, \ldots, T+1 && (5.2) \\
& X^t \leq \beta_2, \text{ for } t = 1, \ldots, T+1 && (5.3) \\
& W^t = W^{t-1} + R^t - X^{t-1}, \text{ for } t = 1, \ldots, T+1 && (5.4) \\
& I^t - B^t = I^{t-1} - B^{t-1} + X^{t-1} - D^{t-1}, \text{ for } t = 2, \ldots, T+1 && (5.5) \\
& Y^t = Y^{t-1} - X^t + D^{t-1}, \text{ for } t = 1, \ldots, T+1 && (5.6) \\
& P\{S - Y^t \geq D^t\} \geq \alpha, \text{ for } t = 1, \ldots, T && (5.7) \\
& X^t, R^t, W^t, I^t, B^t, Y^t \geq 0 \text{ for all } t.
\end{aligned}$$

Equations 5.1-5.3 are constraints defining the clearing function. Equation 5.4 is the WIP balance constraint. Equation 5.5 is the FGI balance constraint. Equation 5.6 is the planned shortfall balance constraint. Equation 5.7 is the chance constraint requiring that the the base stock level, S, less the planned shortfall, Y^t, be sufficiently high to meet demand for period t, D^t, with a probability of at least α. Although forecasted demand quantities are not explicit in this formulation, random demand associated with each period can be generated so as to fully capture the MMFE. We also point out that removing the chance constraint in Equation 5.7, would result in a lower cost solution, since the feasible region would include more solutions. In the next subsection, we present an MSLP formulation that does not include a chance constraint. So, if a decision maker is primarily interested in minimizing cost, this CCM can be viewed as an approximation of that MSLP.

We first make the approximation of replacing the random demands appearing in Equations 5.5 and 5.6 with their mean values. Each mean demand value is the forecasted demand given the information available at the current period, i.e., the conditional expectation of demand

given the available information up to that point in time. If $G_t(\cdot)$ is the cumulative distribution function of the sum of the demand and planned shortfall quantity in period t, then we can rewrite Equation 5.7 as the following deterministic constraint:

$$S \geq G_t^{-1}(\alpha).$$

Substituting for S, this constraint becomes:

$$X^t + Y^t + I^t - B^t \geq G_t^{-1}(\alpha).$$

We also assume that α is set equal to the value that minimizes the expected backorder and holding costs associated with period t. In other words, we wish to find the value of S that minimizes the following costs:

$$\int_{-\infty}^{S} h_I(S-x)g_t(x)dx + \int_{S}^{\infty} b(x-S)g_t(x)dx$$

Here, $g_t(\cdot)$ is the derivative of $G_t(\cdot)$. Since this expression is convex in S, we can find the minimum by taking the derivative with respect to S and setting the expression equal to zero. So, we must solve the following expression for S:

$$bG_t(S) - h_I(1 - G_t(S)) = 0$$

This leads to $G_t(S) = \frac{h_I}{b+h_I}$. From this, we see that $\alpha = \frac{h_I}{b+h_I}$ gives the lowest cost. A lowest cost solution can also be found by ignoring service level considerations and removing the chance constraint from the formulation. However, we see that the chance constraint with this value of α is consistent with the lowest cost solution. The chance constraint serves the purpose of increasing the base stock level, allowing for sufficient safety stock, and guarding against stock outs. Since backorder costs are usually higher than holding costs, this results in a lower cost solution than we would expect if we simply replaced the random variables for demand with

their mean values and did not include the chance constraint.

With these assumptions, this formulation can be translated into the following deterministic linear program:

$$
\begin{aligned}
\min \quad & \textstyle\sum_{t=1}^{T} \left(h_W W^t + h_I I^t + b B^t\right) \\
\text{s.t.} \quad & X^t \leq \tfrac{\beta_1}{\alpha_1} W^t, \text{ for } t = 1, \ldots, T+1 \\
& X^t \leq \tfrac{\beta_2 - \beta_1}{\alpha_2 - \alpha_1}(W^t - \alpha_1) + \beta_1, \text{ for } t = 1, \ldots, T+1 \\
& X^t \leq \beta_2, \text{ for } t = 1, \ldots, T+1 \\
& W^t = W^{t-1} + R^t - X^{t-1}, \text{ for } t = 1, \ldots, T+1 \\
& I^t - B^t = I^{t-1} - B^{t-1} + X^{t-1} - D^t_{1,t-1}, \text{ for } t = 2, \ldots, T+1 \\
& Y^t = Y^{t-1} - X^t + D^t_{1,t-1}, \text{ for } t = 1, \ldots, T+1 \\
& I^t - B^t + X^t + Y^t \geq G_t^{-1}(h_I/(h_I + b)), \text{ for } t = 1, \ldots, T \\
& X^t, R^t, W^t, I^t, B^t, Y^t \geq 0 \text{ for all } t.
\end{aligned}
\tag{5.8}
$$

In Norouzi (2012, Chap. 4), a lengthy derivation shows how the right hand side quantity of Equation 5.8 can be approximated when utilization is high, backorder costs are large compared to holding costs, and a base stock policy is implemented. We also assume that $D^t_{1,t-1}$ is a known value and equal to the demand forecasted for period $t-1$, given the information at the current period. The subscripts match the notation used in Heath and Jackson (1994).

5.3.2 MSLP Model Formulation

In this model, demand for goods in future periods is in accordance with the additive demand forecasting model from Heath and Jackson (1994). At stage $t = 1, \ldots, T+1$, we observe demand in the previous period and forecast demand for the next $H-1$ periods based on forecasts made at stage $t-1$. We assume $H < T$.

The decision variables for stage $t = 1, \ldots, T+1$ are:

$W^t \equiv$ Quantity of WIP inventory, measured in units of finished goods.

$I^t \equiv$ Quantity of finished goods inventory (FGI). (defined for $t > 1$)

$B^t \equiv$ Quantity of back ordered demand. (defined for $t > 1$)

$R^t \equiv$ Quantity of released material, measured in units of finished goods.

$X^t \equiv$ Quantity of finished goods to produce in the next period.

$D^t_{t-1,s} \equiv$ Forecast at stage t of customer orders for finished goods during period $s = t, \ldots, t + H - 2$. (defined for $t > 1$)

$D^t_{t-1,t-1} \equiv$ Quantity of customer orders for finished goods realized during period $t-1$. (defined for $t > 1$)

Notice that two subscripts are used with the "D" variables to be consistent with the notation used by Heath and Jackson (1994). The decision variables W^t and X^t are defined for $t = 0$, but we assume they are known quantities at that stage. We also assume the following quantities are known:

$I^1 \equiv$ Initial quantity of FGI.

$B^1 \equiv$ Initial quantity of back ordered demand.

$D^1_{0,s} \equiv$ Initial forecast at stage 1 of demand during period $s = 1, \ldots, H$.

$D^t_{t-1,t+H-1} \equiv$ Initial forecast at stage $t = 2, \ldots, T - H + 1$ of demand during period $t + H - 1$.(Note: we assume an initial forecast is known for every period at stage 1, but forecasted demand is only updated when it is less than H periods in the future.)

The following cost parameters are also known:

$h_W \equiv$ Unit cost per period for WIP inventory.

$h_I \equiv$ Unit cost per period for FGI.

$b \equiv$ Unit cost per period for back orders.

$\gamma \equiv$ Period discount factor for costs incurred in future periods.

In accordance with the additive demand forecast model described in Heath and Jackson (1994), we define ξ^t to be a vector of H normally distributed random variables defined on the probability space (Ω, σ_t, μ), with zero mean and symmetric positive definite covariance matrix Σ, for $t = 2, \ldots, T+1$. Here, $\{\sigma_t\}_{t=2}^{T+1}$ is a filtration; so, $\sigma_t \subset \sigma_s$ for $2 \leq t \leq s \leq T+1$. See Ross and Peköz (2007, Chap. 3) for an accessible introduction to filtrations and probability theory. The value of ξ^t is known at stage t.

The multi-stage stochastic program is:

$$\begin{aligned}
\min \quad & h_W W^1 + h_I I^1 + bB^1 + E_{\xi^2}[\min \gamma(h_W W^2 + h_I I^2 + bB^2) + \ldots \\
& + E_{\xi^{T+1}}[\min \gamma^T (h_W W^{T+1} + h_I I^{T+1} + bB^{T+1})] \ldots] \\
\text{s.t.} \quad & X^t \leq \tfrac{\beta_1}{\alpha_1} W^t, \text{ for } t = 1, \ldots, T+1 && (5.9) \\
& X^t \leq \tfrac{\beta_2 - \beta_1}{\alpha_2 - \alpha_1}(W^t - \alpha_1) + \beta_1, \text{ for } t = 1, \ldots, T+1 && (5.10) \\
& X^t \leq \beta_2, \text{ for } t = 1, \ldots, T+1 && (5.11) \\
& W^t = W^{t-1} + R^t - X^{t-1}, \text{ for } t = 1, \ldots, T+1 && (5.12) \\
& I^t - B^t = I^{t-1} - B^{t-1} + X^{t-1} - D^t_{t-1,t-1}, \text{ for } t = 2, \ldots, T+1 && (5.13) \\
& D^t_{t-1,s} = D^{t-1}_{t-2,s} + \xi^t_{s-t+2}, \text{ for } s = t-1, \ldots, t+H-2; \\
& \qquad t = 2, \ldots, T+1 && (5.14) \\
& X^t, R^t, W^t, I^t, B^t, D^t_{t-1,s} \geq 0 \text{ for all } t, s.
\end{aligned}$$

Notice that $t-1$ is the exponent of γ for stage t in the objective function and not a superscript index. All other superscripts in the formulation are indices for the stage number. The constraints above are assumed to hold almost surely. All random variables and decision variables associated

with stage $t \geq 2$ implicitly depend on each outcome $\omega \in \Omega$, where Ω is the sample space. All decision variables associated with stage t are measurable with respect to the sigma algebra σ_t for $t = 2, \ldots, T+1$. The decision variables are non-anticipative at stage t; i.e., they are functions of $(\xi^2, \ldots, \xi^t)$, but not $(\xi^{t+1}, \ldots, \xi^{T+1})$ for $t = 2, \ldots, T$.

Equations 5.9, 5.10, and 5.11 define the maximum capacity as a function of WIP in accordance with the clearing function. Constraints in Equation 5.12 are the WIP inventory balance constraints. Equation 5.13 defines the FGI balance constraints. Equation 5.14 defines the demand forecast evolution constraints.

The MSLP formulation presented here uses the $D^t_{t-1,s}$ decision variables to represent actual and forecasted demand. The random variable problem parameters, given by ξ^t, represent additive updates for these decision variables, incorporating all aspects of the MMFE. In the MSLP models appearing in Norouzi (2012, Chapt. 3 and 4) demand for each period appears as a problem parameter in constraints that balance supply with demand. These demand parameters can be generated in accordance with the MMFE, but these parameters were assumed to be independent and identically distributed (i.i.d.) in the computational experiments presented in Norouzi (2012, Chapt. 4). We also point out that exact solutions of this MSLP will have lower average total costs than the MSLP with i.i.d. demand and CCM when forecast evolution is in accordance with the MMFE and the load dependent nature of lead times is in accordance with clearing functions.

5.4 Numerical Experiments

In this section, we present results of numerical experiments for the model presented in Section 5.3.2. Since there are an infinite number of possible demand forecast evolution scenarios for this MSLP, we must solve the problem using approximation. We use Monte Carlo sampling and the sequential bounding approach with global bounds on the optimal dual decision variables presented in Chapter 4. To carry out Monte Carlo sampling, we constructed scenario trees with 16 branches per node. No other scenario tree structures were considered. The sequential

bounding approach iteratively constructs and modifies a scenario tree based on deterministic bounds for the optimal objective function value; so, a scenario tree structure does not need to be specified.

We compare the experimental results using sequential bounding with experiments using the chance constraint model and the MSLP model with i.i.d. demand and a moment matching scenario tree. The chance constraint model results are from Norouzi (2013). The moment matching scenario tree was constructed using the approach of Miller and Rice (1983), with five branches at every node of the scenario tree. Experiments for this model, the Monte Carlo approach, and the sequential bounding approach were conducted on a Windows 7 PC with an 8 core (2.93 GHz) Intel i7 Processor and 16 GB of RAM. All algorithms are implemented in the C++ programming language and use a commerical linear programming solver, the IBM ILOG CPLEX 12 Callable Library, to solve deterministic equivalent linear programming problems for the scenario trees. The user interface for the computer program for sequential bounding is described in Appendix A. Matlab was used for calling the algorithms in a rolling horizon simulation.

5.4.1 Test Instances

A total of 16 test cases were considered, and following Norouzi (2012, Chapt. 4), were evaluated using a rolling horizon simulation. A 250 period planning horizon was used for the chance constraint and moment matching models. These models determined factory material release quantities at the beginning of each period after solving the model for a 4-period rolling horizon problem. Due to run time requirements, a shorter 100 period planning horizon was used for Monte Carlo sampling and all variations of the sequential bounding approach. An example illustration of a 7 period planning horizon with a 2 period rolling horizon is shown in Figure 5.2.

All problem parameters were chosen to conform to Norouzi (2013). The following problem parameters remained fixed for each of the test instances. The forecast planning horizon, H, was set equal to 3 periods. An initial demand of 100 units was assumed for any period beyond the

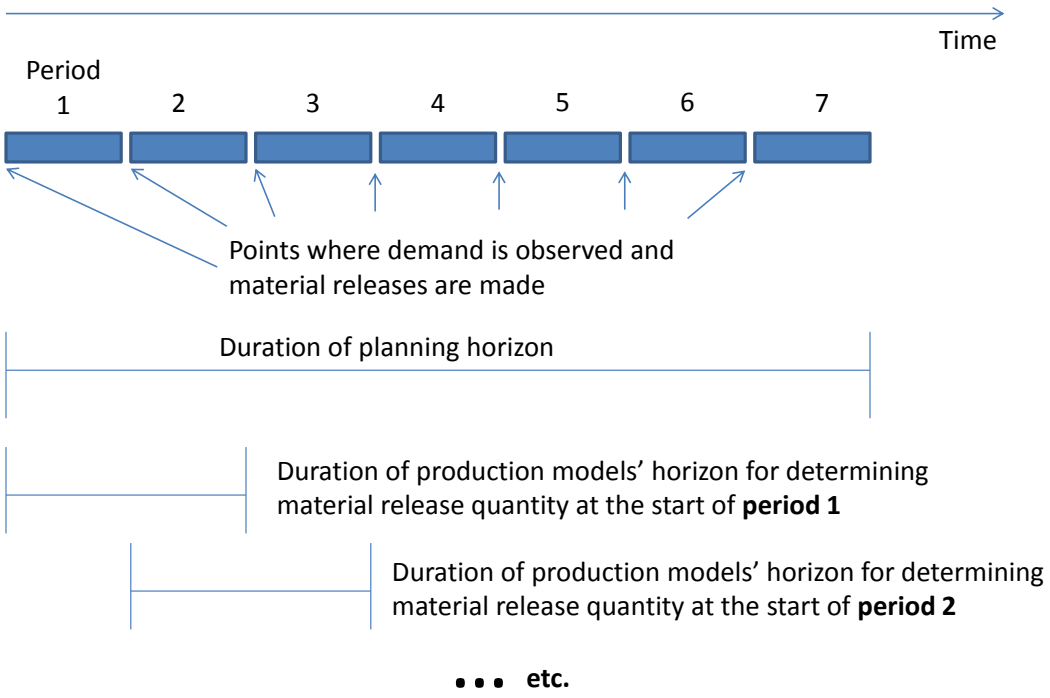

Figure 5.2: Example illustration of a 7 period planning horizon with a 2 period rolling horizon

forecast horizon, i.e., four or more periods into the future. The per period holding cost was set to 20 dollars per unit of FGI and WIP inventory, and no per period discounting was used, $\gamma = 1$. There were no initial backorders, but an initial FGI and WIP of 50 and 200 units of inventory were assumed respectively. A diagonal variance-covariance matrix, corresponding to independent demand for each period, was used. The entries along the diagonal of the matrix are 324, 196, and 105.

Our test instances consisted of a variety of inventory capacities (Cap.) and per period backorder costs (b) for each unit of unmet demand. We considered unlimited production capacity as well as 111.11, 112, and 115 units of total capacity. In keeping with the clearing function model of Section 5.3.2, we assumed that production was limited to half of the WIP inventory as well as the fixed capacity level. Four different backorder costs were considered for each of these capacity levels: 122.86, 180, 313.33, and 980.

5.4.2 Computational Results

In all of the experiments, we report the average total backorder and holding costs for the simulated planning horizon. We also report the service level in parentheses, defined to be one minus the fraction of periods where at least one unit of demand was unmet. Table 5.1 presents the results of the chance constrained model (CCM), which always runs in a matter of seconds for each period, and the MSLP model from Norouzi (2012, Chapt. 4) with i.i.d. demand and a moment matched (MM) scenario tree, which never requires more than a minute for each period. The table also presents results from the MSLP model presented in Section 5.3.2 with the Monte Carlo (MC) sampling approach. Although this approach requires more run time than the Moment Matching approach because the corresponding scenario tree is larger, it never required more than a minute of run time for each period. This MSLP was also solved using the sequential bounding (SB) approach when 1 and 5 minutes of run time are used for each period.

In Table 5.1, we see that CCM has the lowest average total cost of all the cases, except where the backorder cost is lowest with the two lowest capacity levels. In these cases MM has the lowest

Table 5.1: Experimental results reported as average total cost (service level)

Case	Cap.	b	CCM	MM	MC	SB 1 min.	SB 5 min.
1	111.11	128.86	5016 (0.91)	4999 (0.85)	5487 (0.74)	6598 (0.38)	6598 (0.38)
2	111.11	180	5210 (0.95)	5230 (0.91)	5824 (0.81)	7713 (0.40)	7712 (0.40)
3	111.11	313.33	5495 (0.97)	5524 (0.92)	6422 (0.86)	10311 (0.41)	10310 (0.41)
4	111.11	980	5989 (0.98)	6177 (0.97)	8304 (0.90)	22624 (0.41)	22642 (0.41)
5	112	128.86	4969 (0.90)	4966 (0.86)	5415 (0.76)	6461 (0.38)	6461 (0.38)
6	112	180	5144 (0.95)	5197 (0.91)	5759 (0.80)	7530 (0.39)	7530 (0.39)
7	112	313.33	5418 (0.96)	5461 (0.93)	6303 (0.86)	9983 (0.43)	9983 (0.43)
8	112	980	5896 (0.98)	6063 (0.97)	8021 (0.90)	21829 (0.43)	21828 (0.43)
9	115	128.86	4837 (0.89)	4869 (0.88)	5190 (0.78)	6087 (0.41)	6087 (0.41)
10	115	180	4975 (0.93)	5054 (0.90)	5513 (0.79)	7013 (0.41)	7013 (0.41)
11	115	313.33	5211 (0.96)	5332 (0.94)	5959 (0.87)	9137 (0.42)	9137 (0.42)
12	115	980	5600 (0.98)	5648 (0.97)	7346 (0.90)	19326 (0.46)	19330 (0.45)
13	Unlim.	128.86	4579 (0.86)	4786 (0.97)	4580 (0.84)	5017 (0.54)	5017 (0.54)
14	Unlim.	180	4639 (0.88)	4806 (0.97)	4652 (0.89)	5425 (0.54)	5425 (0.54)
15	Unlim.	313.33	4720 (0.94)	4854 (0.97)	4772 (0.92)	6378 (0.54)	6378 (0.54)
16	Unlim.	980	4884 (0.98)	5093 (0.97)	5295 (0.93)	11139 (0.54)	11139 (0.54)

total cost. Since the approximation used for the chance constraint in CCM depends on high backorder costs, the higher total costs associated with CCM seems reasonable for these cases. MM has a lower service level for these cases. This suggests that a low backorder cost causes CCM to effectively overestimate demand variance, produce more safety stock than necessary, and achieve higher service levels. However, the chance constraint approximation also depends on high utilization, which is achieved at low capacity levels. The fact that the capacity is low for these cases implies that the chance constraint approximation effectively underestimates the variance in demand at high capacity levels, generating insufficient safety stock and achieving relatively low service levels. This explanation is consistent with MM having a higher service level than CCM when capacity is unlimited. The overestimation of demand variance for low backorder costs is cancelled out by the underestimation of demand variance for high capacity levels enough for CCM to have the lowest cost in this case.

Next, notice that MC tends to have higher costs and lower service levels than the algorithm with the lowest total cost for every case. The case with unlimited capacity and backorder cost of 180 is the only case where the service level for MC is higher. This suggests that MC is

underestimating the variance in demand and not producing sufficient safety stock. This seems reasonable when we consider the fact that MC generates 16 demand outcomes for each period to approximate a multivariate continuous normal distribution. Since the expected value of the variance of a sample of size n is $\frac{n-1}{n}$ times the variance of the underlying distribution, we see that MC underestimates the true demand variance by about 94%.

We also observe that sequential bounding has considerably higher costs and lower service levels than the Monte Carlo sampling approach. Sequential bounding also has considerably higher costs and lower service levels than both the chance constrained model and MSLP model with moment matched scenario tree from Norouzi (2012, Chapt. 4). In general, both of these models tended to build and maintain higher levels of safety stock than the sequential bounding approach. Increasing the run time for sequential bounding by a factor of 5 improved its performance only by a very small amount, if at all. In case 12, the added run time resulted in a slightly higher cost. We can conclude that additional run time is unlikely to improve the performance of sequential bounding. We also observe that the chance constrained model does best, relative to sequential bounding, when backorder costs are high. This is consistent with the approximation used to translate the chance constrained model into a linear program, which relied on the backorder cost being large compared to holding costs.

Due to poor performance in several test instances, we considered four modifications to the sequential bounding algorithm. First, we ignored the effects of forecast evolution and used the sequential bounding approach with i.i.d. in a manner similar to Norouzi (2012, Chapt. 4). This involves removing Equation 5.14 and treating each $D^t_{t-1,t-1}$ as an i.i.d. random variable instead of a decision variable in Equation 5.13. For the second approach we increased the length of the rolling horizon to 20 periods, instead of 4. Here, we used the sequential bounding approach with 5 minutes of run time for each period. All the other results are based on 1 minute of run time for each period. The third approach uses sequential bounding, but we forced the scenario tree to have 16 branches at every node. Sequential bounding is based on a partition of the random variables' support and selects a cell in this partition to split into two new cells at every

Table 5.2: Experimental results for sequential bounding modifications reported as average total cost (service level)

Case	Cap.	b	i.i.d. Demand	20 Periods	16 Branches	Random Select
1	111.11	128.86	5353 (0.84)	6596 (0.38)	5540 (0.71)	5660 (0.69)
2	111.11	180	5639 (0.88)	7711 (0.41)	5899 (0.75)	6138 (0.69)
3	111.11	313.33	6155 (0.89)	10314 (0.41)	6695 (0.82)	7176 (0.73)
4	111.11	980	7669 (0.91)	22625 (0.41)	10148 (0.86)	12744 (0.75)
5	112	128.86	5302 (0.87)	6461 (0.38)	5457 (0.74)	5563 (0.69)
6	112	180	5567 (0.88)	7532 (0.39)	5815 (0.76)	6032 (0.72)
7	112	313.33	6019 (0.90)	9981 (0.43)	6578 (0.82)	7007 (0.74)
8	112	980	7344 (0.92)	21880 (0.43)	9723 (0.86)	12178 (0.77)
9	115	128.86	5123 (0.85)	6087 (0.41)	5210 (0.76)	5318 (0.71)
10	115	180	5321 (0.90)	7013 (0.41)	5484 (0.79)	5765 (0.73)
11	115	313.33	5678 (0.90)	9139 (0.42)	6096 (0.81)	6651 (0.75)
12	115	980	6541 (0.93)	19335 (0.45)	8173 (0.88)	10311 (0.80)
13	Unlim.	128.86	4690 (0.95)	5017 (0.54)	4595 (0.88)	4594 (0.80)
14	Unlim.	180	4775 (0.98)	5425 (0.54)	4651 (0.89)	4716 (0.80)
15	Unlim.	313.33	4907 (0.99)	6378 (0.54)	4751 (0.94)	5003 (0.80)
16	Unlim.	980	5134 (0.99)	11139 (0.54)	4966 (0.98)	5289 (0.93)

iteration of the algorithm. The algorithm selects the cell with the largest contribution to the scenario tree approximation error. For the fourth and final modification, we tried an alternative cell selection approach. Cells are randomly selected, and each cell's probability of selection is weighted according to its contribution to the scenario tree approximation error. The results of these four approaches are shown in Table 5.2.

In Table 5.2, although the modifications did improve the results for sequential bounding, the chance constrained model has a lower average total cost for every test case. We see that extending the length of the rolling horizon to 20 periods generally resulted in only a small change in the sequential bounding results, if at all. However, random selection of cells improved the sequential bounding results considerably. When a 16 branch per node scenario tree was used, sequential bounding tended to perform even better than when random cell selection was used. Finally, we see that the best sequential bounding modification was achieved by considering

i.i.d. demand, and ignoring forecasted demand quantities.

There are several possible explanations for the relatively poor performance of sequential bounding. First, the global decision variable bounds may not be tight enough to generate good bounds on the optimal objective function value and select promising cells of the partition of the random variable support. However, a second possible cause seems more likely. When the scenario trees generated at the final iteration of sequential bounding were examined, we saw that the random variables corresponding to updates in forecasted demands for future periods tended to be replaced with their mean values. This resulted in underestimating the effects of demand variance and preventing the level of safety stock from rising to levels necessary to achieve high service levels. Since each iteration of sequential bounding affects a single random variable associated with a single stage, large scale problems are vulnerable to this issue. When we used i.i.d. demands, we effectively reduce the number of random variables in the formulation; so, this explanation is consistent with the fact that sequential bounding performed best with i.i.d. demand.

5.5 Conclusions

In this chapter, we formulated an MSLP model that integrated the full MMFE developed by Heath and Jackson (1994) with clearing functions developed by Graves (1986) to model the evolution of demand forecasts for production planning subject to load-dependent lead times. We solved this model using two general MSLP solution approaches, Monte Carlo sampling and the sequential bounding method described in Chapter 4. Neither of these two approaches require any instance or problem specific assumptions to be made. We compared the results to two solution approaches used for computational experiments in Norouzi (2012, Chapt. 4). The first of these approaches depends on assumptions associated with the structure of particular instances. Both approaches use a simplified version of the MMFE by either assuming mean values or i.i.d random variables for demand.

For this particular problem and set of test instances, we found that the chance constrained

model generally outperforms the MSLP with moment matching from Norouzi (2012) as well as the MSLP presented in this chapter when Monte Carlo sampling and sequential bounding were used. The chance constrained approach continued to outperform sequential bounding after several modifications. This indicates that the chance constrained approach may be competitive or superior to the MSLP approaches in many applications. We feel that these results support the use of the chance constrained model in practice, especially given its speed of execution. However, caution should be exercised when using this approach for instances with cost structures that differ significantly from the instances reported here. Whether or not this conclusion holds for multi-echelon and multi-product production planning problems also remains to be seen.

This chapter also provides an excellent benchmark of the sequential bounding approach for a challenging MSLP. We feel that this study can help inform future development of general MSLP solution methods. Sequential bounding tended to generate scenario trees where random variables corresponding to updates in forecasted demands for future periods were replaced with their mean values. This suggests that improvements in sequential bounding may be possible by modifying the outcomes associated with these random variables in a way that allows higher safety stocks to be built. An approach similar to the sequential bounding approach with optimized outcomes developed in Chapter 3 for two-stage stochastic programs might help to achieve this, since the outcomes and first stage decision variables are jointly determined in that approach.

Chapter 6

Conclusions and Areas for Future Research

In this dissertation we reviewed the stochastic programming literature in Chapter 2, focusing on those methods that efficiently solve multi-stage stochastic programs (MSLPs). We reviewed a variety of methods for producing scenario trees to approximate a large or infinite set of possible scenarios. Most of these approaches require some level of problem-specific tuning or computational experience for successful implementation.

We presented new sequential bounding approaches for two-stage stochastic programs and applied these approaches to a single-period production planning problem with downward product substitutions in Chapter 3. The approach is a general one, which can be applied to problems with random variables appearing in the cost coefficients as well as the constraints of the formulation. The linear programming relaxation approach, combined with restricted recourse and an upper bounding linear program with optimized outcomes resulted in the best performance of sequential bounding. We found that the solution of the upper bounding linear program was the primary contributor to improvements in the algorithm. We found test cases where the new algorithm outperforms the Sample Average Approximation (SAA) with certainty, based on deterministic bounds on the objective function values, and we found a number of test cases where

the 99% confidence intervals for total average cost were below those for SAA. However, relative performance generally appears to depend on the particular instance under consideration.

We also presented a multi-stage sequential bounding approach, and compared this with three other scenario generation approaches for a production planning problem with uncertain demand and constant work in process inventory in Chapter 4. The alternative approaches included Monte Carlo sampling, EVPI importance sampling, and moment matching. Every one of the approaches had some test cases where it produced the highest average profit at the 5% level of significance. While showing that sequential bounding is a competitive approach for such problems, we also observed that the relative success of the algorithm depends on the particular instance under consideration. Sequential bounding has the advantage of requiring less problem-specific tuning, since it automatically generates and refines a scenario tree and progress is based on a deterministic measure of the approximation error. It is also a very general approach in the sense that it can be used to solve problems with random variables appearing in the cost and constraint coefficients, as well as the right hand sides of the constraints at each stage of the formulation. We also feel that the Monte Carlo sampling approach, with its relative ease of implementation and good performance, should be considered in practical applications.

Finally, we developed a multi-stage stochastic program to model the evolution of demand forecasting for production planning with load dependent lead times in Chapter 5. In addition to being a multi-stage problem, there were multiple random variables associated with forecasting at each stage of the problem, making this a challenging problem for benchmarking the general sequential bounding approach. We considered previously developed models and approaches: a chance constrained model and an MSLP model based on independent and identically distributed demand and moment matched scenario tree. These approaches, with their problem-specific simplifying assumptions, outperformed sequential bounding, even with several modifications. We feel that this supports the use of the chance constrained approach when the instance parameters are similar to those used in our computational experiments, but additional models and experiments should be conducted to determine if this success will carry over to multi-echelon

and multi-product environments. We also feel that the results of our study provide insights into developing improved approaches for solving more general multi-stage stochastic programs.

From our analysis of the computational results in Chapters 3-5, we observed that the sequential bounding algorithm consistently suffers from underestimating variance. As pointed out by Miller and Rice (1983), this is a problem with any technique that uses mean values for intervals of the support set of a random variable. For two-stage problems, we were able to overcome this issue by selecting values other than the conditional mean values for each interval with the method of optimized outcomes. It may be possible to use more than one outcome for each interval, and not just the conditional mean value, to improve the performance of sequential bounding for multi-stage problems. In Chapter 5, we observed that the tendency to underestimate variance can also affect Monte Carlo sampling approaches to solving MSLPs. So, this issue is an important one that should be addressed in future research.

Throughout Chapters 3-5, we also observed that the particular instance has an influence on the relative performance of sequential bounding. The cost structure and the need to stock out of finished goods at the end of the planning horizon appeared to have the most influence in the model considered in Chapter 4, but we found that the random production yield can also be significant when demand variance is low. Because the problem of Chapter 5 had a very long rolling horizon, the need to stock out at the last period did not influence the relative performance of the algorithms. In all of the problems, the influence of the cost structure and demand distribution had a strong influence, due to the consistent underestimation of demand variance of sequential bounding. When backorder costs were high, the need for sufficient safety stock became more important and this effect was most prominent.

In summary, we see that the success of sequential bounding, either for two or multi-stage problems, depends on the particular problem and instance under consideration. However, since sequential bounding iteratively builds and improves upon the scenario tree based on a deterministic measure of the error in the approximation, it generally does not require problem-specific tuning or computational experience. This is particularly important if stochastic programming

is to be used successfully by people who are not expert practitioners. We feel that this dissertation has provided insight for future development of more automated and general stochastic programming solution methods.

We feel that there are promising areas for future research. For two-stage problems, it may be possible to build on the success of the optimized outcome approach in Chapter 3. In particular, it may be possible to optimize the boundries of cells used in the random variables' support partition, as well as the outcomes being used to represent each cell. It may also be possible to extend the method of optimized outcomes to problems with random variables appearing in places other than the right hand sides of the constraints. The primary challenge to these extensions is determining whether or not a meaningful upper bounding linear program can be constructed. If not, then perhaps a nonlinear convex program is possible.

We also feel that the optimized outcome approach of Chapter 3 is a promising one for multi-stage stochastic programs. The challenge here is in extending the results for two-stage problems to an upper bounding problem for multi-stage problems. However, overcoming this challenge may be worthwhile. Many of the random variables appearing in the multi-stage problem of Chapter 5 were simply replaced with their mean values. An approach that allows alternative representative outcomes, other than conditional mean values, may improve the performance of sequential bounding for such problems, allowing safety stock levels to rise in accordance with higher demand and forecast variance. Such an approach also may improve the relative performance of sequential bounding compared to other approaches presented in Chapter 4.

We also feel that there are many opportunities for improving the sequential bounding approach for multi-stage stochastic programs. In particular, the success of one modification to the algorithm, random selection of cells, indicates that heuristic approaches for selecting a cell of the random variable support partition for splitting at each iteration may be worthwhile. This was not the most successful modification to the algorithm in that chapter, but it does not rely on details of the problem structure; so, it can be adapted to other problems. Preliminary experiments with the two-stage stochastic program of Chapter 3 showed degraded performance

of sequential bounding when cells were selected randomly. So, there appear to be limits to such approaches.

It may also be possible to build on the success of the chance constrained approach in Chapter 5 for more general MSLPs. The approach of adding chance constraints to an MSLP formulation and converting them into deterministic constraints consistent with a low cost solution may be a successful one for problems other than those for production planning. In order to apply this approach, it is necessary to find good approximations to distribution functions that may be impossible to determine analytically. However, the effort required to find these approximations may be rewarded with higher quality solutions than can be produced with any other available MSLP solution approach.

Building on the exact bounds for dual decision variables described in Chapter 3, it may be possible to use sequential bounding to obtain exact solutions to instances of certain problems, such as the appointment scheduling problem presented in Denton and Gupta (2003). However, several very challenging obstacles need to be overcome before this can happen. The number of possible optimal second stage dual bases, and therefore optimal second stage dual solutions, appears to be polynomial in the number of appointments for this problem. At each iteration of sequential bounding, each optimal second stage dual solution corresponds to a region of the random variable support corresponding to a simplicial cone. If sequential bounding can be modified to use simplicial cones instead of hyper-rectangular cells to partition the support of the random variables, then we may be able to define the cells to correspond to each region of the random variable support where the optimal second stage dual solution is constant. This would result in deterministic lower and upper bounds being equal, since the lower and upper decision variable bounds would be equal in every cell.

It may also be possible to extend the variable bounding approaches of Chapter 3 to some types of two-stage stochastic programs with integer recourse. For example, the appointment scheduling problem presented in Denton and Gupta (2003) can be extended to a problem with integer recourse when more than one server is utilized. The associated bounds nonlinear

programs of Chapter 3 would become bounds nonlinear mixed integer programs in this case.

Finally, the ability of stochastic programs to capture the transient nature of queues has been relatively unexplored. As Chan and Schruben (2008) demonstrated for tandem queues, two-stage stochastic programs can sometimes be used to find key output statistics faster than simulation models. Successfully capturing the dynamics of more complicated queueing networks embedded in a stochastic program would help to model uncertainties in production systems. This approach might also contribute to modelling and optimization of projects, or other problems where jobs have a network dependency structure.

REFERENCES

Al-Khayyal, F. A. and Falk, J. E. (1983). Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286.

Arora, S. and Barak, B. (2009). *Computational complexity: a modern approach*. Cambridge University Press.

Bassok, Y., Anupindi, R., and Akella, R. (1999). Single-Period multiproduct inventory models with substitution. *Operations Research*, 47(4):632–642.

Batun, S., Denton, B. T., Huschka, T. R., and Schaefer, A. J. (2011). Operating room pooling and parallel surgery processing under uncertainty. *INFORMS Journal on Computing*, 23(2):220–237.

Bertsimas, D. and Tsitsiklis, J. N. (1997). *Introduction to linear optimization*. Athena Scientific Belmont.

Birge, J. (1983). Aggregation bounds in stochastic production problems. Technical Report 83-13, University of Michigan, Department of Industrial and Operations Engineering, Ann Arbor, MI.

Birge, J. (1985a). Aggregation bounds in stochastic linear programming. *Mathematical Programming*, 31(1):25–41.

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

Birge, J. R. and Holmes, D. F. (1992). Efficient solution of two-stage stochastic linear programs using interior point methods. *Computational Optimization and Applications*, 1(3):245–276.

Birge, J. R. and Louveaux, F. (1997). *Introduction to Stochastic Programming*. Springer, New York, New York.

Birge, J. R. and Wets, R. J.-B. (1986). Designing approximation schemes for stochastic optimization problems. *Mathematical Programming Study*, 27:54102.

Cariño, D. R., Kent, T., Myers, D. H., Stacy, C., Sylvanus, M., Turner, A. L., Watanabe, K., and Ziemba, W. T. (1994). The Russell-Yasuda kasai model: An Asset/Liability model for a japanese insurance company using multistage stochastic programming. *Interfaces*, 24(1):29–49.

Cario, M. and Nelson, B. (1997). Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix. *Northwestern University, IEMS Technical Report*, 50:100150.

Casey, M. S. and Sen, S. (2005). The scenario generation algorithm for multistage stochastic linear programming. *Mathematics of Operations Research*, 30(3):615631.

Chan, W. K. V. and Schruben, L. (2008). Optimization models of discrete-event system dynamics. *Operations Research*, 56(5):1218–1237.

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

Charnes, A. and Cooper, W. W. (1963). Deterministic equivalents for optimizing and satisficing under chance constraints. *Operations Research*, 11(1):18–39.

Charnes, A., Cooper, W. W., and Symonds, G. H. (1958). Cost horizons and certainty equivalents: an approach to stochastic programming of heating oil. *Management Science*, 4(3):235–263.

Chiralaksanakul, A. and Morton, D. (2004). Assessing policy quality in multi-stage stochastic programming. *Stochastic Programming E-Print Series*, 12:4.

Dantzig, G. and Glynn, P. (1990). Parallel processors for planning under uncertainty. *Annals of Operations Research*, 22(1):1–21.

Dantzig, G. and Infanger, G. (1993). Multi-stage stochastic linear programs for portfolio optimization. *Annals of Operations Research*, 45(1):59–76.

Dantzig, G. B. (1963). *Linear programming and extensions*. Princeton University Press, Princeton, N.J.

Dattorro, J. (2005). *Convex optimization and Euclidean distance geometry*. Meboo Publishing USA.

Dempster, M. A. H. (2006). Sequential importance sampling algorithms for dynamic stochastic programming. *Journal of Mathematical Sciences*, 133(4).

Dempster, M. A. H. and Thompson, R. T. (1998). Evpi-based importance sampling solution procedures for multistage stochastic linear programmes on parallel mimd architectures. *Annals of Operations Research*, 90.

Denton, B. and Gupta, D. (2003). A sequential bounding approach for optimal appointment scheduling. *IIE Transactions*, 35(11):1003–1016.

Di Domenica, N., Lucas, C., Mitra, G., and Valente, P. (2009). Scenario generation for stochastic programming and simulation: a modelling perspective. *IMA Journal of Management Mathematics*, 20(1):1–38.

Dupacova, J. (2002). Applications of stochastic programming: achievements and questions. *European Journal of Operational Research*, 140(2):281–290.

Dupacova, J., Consigli, G., and Wallace, S. W. (2001). Scenarios for multistage stochastic programs. *Annals of Operations Research*, 100:25–30.

Dyer, M. and Stougie, L. (2006). Computational complexity of stochastic programming problems. *Mathematical Programming*, 106(3):423–432.

Eppen, G. D., Martin, R. K., and Schrage, L. (1989). A scenario approach to capacity planning. *Operations Research*, 37(4):517–527.

Erdogan, S. A., Gose, A. H., and Denton, B. T. (2011). On-line appointment sequencing and scheduling. *working paper*.

Gassmann, H. I. (1990). Mslip: A computer code for the multistage stochastic linear programming problem. *Mathematical Programming*, 47:407–423.

Glassey, C. R. (1973). Nested decomposition and multi-stage linear programs. *Management Science*, 20(3):282–292.

Glynn, P. and Infanger, G. (2011). Simulation-based confidence bounds for two-stage stochastic programs (submitted for publication).

Graves, S. C. (1986). A tactical planning model for a job shop. *Operations Research*, 34(4):522–533.

Graves, S. C., Meal, H. C., Dasu, S., and Qui, Y. (1986). Two-stage production planning in a dynamic environment. In Axster, S., Schneeweiss, C. A., and Silver, E. A., editors, *Multi-stage production planning and inventory control*, Lecture notes in economics and mathematical systems ; 266. Springer-Verlag, Berlin.

Guan, Z. and Philpott, A. B. (2011). A multistage stochastic programming model for the New Zealand dairy industry. *International Journal of Production Economics*, 134(2):289–299.

Gupta, V. and Grossmann, I. E. (2010). Solution strategies for multistage stochastic programming with endogenous uncertainties. *Computers and Chemical Engineering*, In Press, Corrected Proof.

Heath, D. C. and Jackson, P. L. (1994). Modeling the evolution of demand forecasts with application to safety stock analysis in production/distribution systems. *IIE transactions*, 26(3):17–30.

Higle, J. L. and Kempf, K. G. (2011). Production planning under supply and demand uncertainty: A stochastic programming approach. In Infanger, G., editor, *Stochastic Programming*, volume 150 of *International Series in Operations Research and Management Science*, pages 297–315. Springer New York.

Higle, J. L., Rayco, B., and Sen, S. (2010). Stochastic scenario decomposition for multistage stochastic programs. *IMA Journal of Management Mathematics*, 21(1):3966.

Higle, J. L. and Sen, S. (1991). Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of operations research*, page 650669.

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

Huang, C. C., Ziemba, W. T., and Ben-Tal, A. (1977). Bounds on the expectation of a convex function of a random variable: With applications to stochastic programming. *Operations Research*, 25(2):315–325.

Huang, K. and Ahmed, S. (2009). The value of multi-stage stochastic programming in capacity planning under uncertainty. *Operations Research*, 57:893–904.

Jonsbraten, T. W., W. R. J. B. and Woodruff, D. L. (1998). A class of stochastic programs with decision dependent random elements. *Annals of Operations Research*, 82:83–106.

Kacar, N. B. (2012). *Fitting Clearing Functions to Empirical Data: Simulation Optimization and Heuristic Algorithms*. PhD thesis, North Carolina State University.

Kall, P. and Wallace, S. W. (1994). *Stochastic Programming*. John Wiley & Sons.

Kaut, M. and Wallace, S. W. (2007). Evaluation of scenario-generation methods for stochastic programming. *Pacific Journal of Optimization*, 3(2):257–271.

Kleywegt, A., Shapiro, A., and Homem-de Mello, T. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502.

Korf, L. (2004). Stochastic programming duality: $\mathcal{L}^\infty$ multipliers for unbounded constraints with an application to mathematical finance. *Mathematical programming*, 99(2):241–259.

Kouwenberg, R. (2001). Scenario generation and stochastic programming models for asset liability management. *European Journal of Operational Research*, 134(2):279–292.

Laporte, G. and Louveaux, F. V. (1993). The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142.

Law, A. M. (2006). *Simulation modeling and analysis*. McGraw-Hill series in industrial engineering and management science. McGraw-Hill,, Boston, 4th ed. edition.

Mak, W. K., Morton, D. P., and Wood, R. K. (1999). Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24(1-2):47–56.

Miller, A. C. and Rice, T. R. (1983). Discrete approximations of probability distributions. *Management Science*, 29(3):352362.

Missbauer, H. and Uzsoy, R. (2011). Optimization models of production planning problems. In Kempf, K. G., Keskinocak, P., and Uzsoy, R., editors, *Planning Production and Inventories in the Extended Enterprise*, number 151 in International Series in Operations Research & Management Science, pages 437–507. Springer US.

Morton, D. P. and Wood, R. K. (1999). Restricted-recourse bounds for stochastic linear programming. *Operations Research*, 47(6):943–956.

Mula, J., Poler, R., Garcia-Sabater, J., and Lario, F. (2006). Models for production planning under uncertainty: A review. *International journal of production economics*, 103(1):271–285.

Mulvey, J. and Vladimirou, H. (1989). Stochastic network optimization models for investment planning. *Annals of Operations Research*, 20(1):187–217.

Norouzi, A. (2012). *The Effect of Forecast Evolution on Production Planning with Resources Subject to Congestion*. PhD thesis, North Carolina State University.

Norouzi, A. (2013). Independent demand. 16 May e-mail from the author.

Pennanen, T. (2009). Epi-convergent discretizations of multistage stochastic programs via integration quadratures. *Mathematical Programming*, 116(1):461–479.

Pennanen, T. and Koivu, M. (2002). Integration quadratures in discretization of stochastic programs. *Stochastic programming e-print series*, 11.

Pereira, M. V. F. and Pinto, L. M. V. G. (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52(1):359–375.

Pflug, G. (2001). Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical Programming*, 89(2):251–271.

Rao, U. S., Swaminathan, J. M., and Zhang, J. (2004). Multi-product inventory planning with downward substitution, stochastic demand and setup costs. *IIE Transactions*, 36(1):59–71.

Rockafellar, R. and Wets, R.-B. (1976a). Stochastic convex programming: basic duality. *Pacific Journal of Mathematics*, 62(1):173–195.

Rockafellar, R. and Wets, R.-B. (1976b). Stochastic convex programming: relatively complete recourse and induced feasibility. *SIAM Journal on Control and Optimization*, 14(3):507–522.

Rockafellar, R. T. and Wets, R. J. B. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147.

Ross, S. and Peköz, E. (2007). *A second course in probability*. ProbabilityBookstore. com, Boston, MA.

Santoso, T., Ahmed, S., Goetschalckx, M., and Shapiro, A. (2005). A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167(1):96–115.

Sen, S. and Higle, J. L. (1999). An introductory tutorial on stochastic linear programming models. *Interfaces*, 29(2):33–61.

Shapiro, A. (2001). Monte carlo simulation approach to stochastic programming. In *Simulation Conference, 2001. Proceedings of the Winter*, volume 1, pages 428 –431 vol.1.

Shapiro, A., Dentcheva, D., and Ruszczyski, A. (2009). *Lectures on stochastic programming: modeling and theory*, volume 9. Society for Industrial Mathematics.

Shapiro, A. and Homem de Mello, T. (1998). A simulation-based approach to two-stage stochastic programming with recourse. *Mathematical Programming*, 81(3):301–325.

Sherali, H. D. and Adams, W. P. (1998). *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, volume 31. Springer.

Stroud, A. H. and Secrest, D. (1966). *Gaussian Quadrature Formulas*. Prentice-Hall.

Van Slyke, R. and Wets, R.-B. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal of Applied Mathematics*, 17:638–663.

Wets, R. J. B. (1983). Solving stochastic programs with simple recourse. *Stochastics: An International Journal of Probability and Stochastic Processes*, 10(3-4):219–242.

Wittrock, R. J. (1983). Advances in a nested decomposition algorithm for solving staircase linear programs. Technical Report SOL 83-2, Stanford University, Stanford, CA.

Wright, S. E. (1994). Primal-Dual aggregation and disaggregation for stochastic linear programs. *Mathematics of Operations Research*, 19(4):893–908.

Zanjani, M. K., Nourelfath, M., and Ait-Kadi, D. (2010). A multi-stage stochastic programming approach for production planning with uncertainty in the quality of raw materials and demand. *International Journal of Production Research*, 48(16):47014723.

Zipkin, P. H. (1980a). Bounds for Row-Aggregation in linear programming. *Operations Research*, 28(4):903–916.

Zipkin, P. H. (1980b). Bounds on the effect of aggregating variables in linear programs. *Operations Research*, 28(2):403–418.

APPENDIX

Appendix A

User Interface for an MSLP

In this appendix, we describe the user interface to the computer program for the new MSLP model presented in Chapter 5, and we provide a small example instance. The software reads a specially formatted text file. Each line in the text file contains a parameter name, a colon, and the parameter value. The parameters must appear in order. Requirements for each parameter include the type, "int" for integer and "double" for double precision floating point number, and any bounds. The data are described below:

`T:` the value of T (int; > 0).

`H:` the value of H (int; > 0; $\leq T$).

`W0-X0:` the value of $W^0 - X^0$ (double; ≥ 0).

`I1:` the value of I^1 (double; ≥ 0).

`B1:` the value of B^1 (double; ≥ 0).

`D1[`s`]:` the value of $D^1_{0,s}$ (double; ≥ 0), with an entry for each value of $s = 1, \ldots, H$.

`Dt[`$t + H - 1$`]:` the value of $D^t_{t-1,t+H-1}$ (double; ≥ 0), with an entry for each value of $t = 2, \ldots, T - H + 1$.

`alpha1:` the value of α_1 (double; > 0; $\geq \beta_1$; $< \alpha_2$).

`alpha2:` the value of α_2 (double; > 0; $> \alpha_1$).

`beta1:` the value of β_1 (double; ≥ 0; $\leq \alpha_1$; $\geq \beta_2 \frac{\alpha_1}{\alpha_2}$).

`beta2:` the value of β_2 (double; ≥ 0; $\geq \beta_1$; $\leq \beta_1 \frac{\alpha_2}{\alpha_1}$).

`hW:` the value of h_W (double; ≥ 0).

`hI:` the value of h_I (double; ≥ 0).

`b:` the value of b (double; ≥ 0).

`gamma:` the value of γ (double; > 0; ≤ 1).

`S[`i`][`j`]:` the value of $S_{i,j}$ (double). The value in row i and column j of the orthogonal matrix S, whose columns are eigenvectors for the covariance matrix Σ. There is an entry for each value of $i, j = 1, \ldots, H$.

`eigval[`i`]:` the value of λ_i (double; > 0), the i^{th} eigenvalue of the covariance matrix Σ. These entries should match the eigenvector entries given above. There is an entry for each $i = 1, \ldots, H$.

`maxIter:` the value for the maximum number of iterations (int; > 0) used for the sequential bounding algorithm.

`maxError:` the value for the maximum error (double; > 0) used for the sequential bounding algorithm.

`maxRuntime:` the value for the maximum runtime (double; > 0) in seconds for the sequential bounding algorithm.

`outputFilename:` the full path and filename (char[]) where the output data will be written. This file should have a ".dat" extension. Spaces are permitted in directory and filenames.

An example input text file for a small problem instance is given below:

```
T:4
H:2
W0-X0:500
I1:0
B1:0
D1[1]:2000
D1[2]:3000
D2[3]:2000
D3[4]:2000
alpha1:2000
alpha2:4000
beta1:2000
beta2:3000
hW:1
hI:1
b:9
gamma:.99
S[1][1]:-0.8817
S[1][2]:0.4719
S[2][1]:0.4719
S[2][2]:0.8817
eigval[1]:197
eigval[2]:380
maxIter:500
maxError:1e-5
maxRuntime:600
```

```
outputFile:smallTest.dat
```