Sequential Bounding Methods for Two-Stage Stochastic Programs

Alexander H. Gose

Graduate Program of Operations Research,

North Carolina State University, Raleigh, NC 27695, ahgose@ncsu.edu

Brian T. Denton

Department of Industrial and Operations Engineering,

University of Michigan, Ann Arbor, MI 48109, btdenton@umich.edu

October 17, 2014

Abstract

In rare situations, stochastic programs can be solved analytically. Otherwise, approximation is necessary to solve stochastic programs with a large or infinite number of scenarios to a desired level of accuracy. This involves statistical sampling or deterministic selection of a finite set of scenarios to obtain a tractable deterministic equivalent problem. Some of these approaches rely on bounds for primal and dual decision variables of the second stage. We develop new algorithms to improve these bounds and reduce the deterministic approximation error. Experiments were conducted to compare a sequential approximation approach with and without these new algorithms. Each algorithm is applied to a set of test instances for a problem of managing semiconductor inventory with downward substitutions, where random variables only appear in the right hand side of the second stage. Experiments were also conducted using the Sample Average Approximation (SAA) algorithm. The sequential approximation and SAA algorithm generate a feasible solution upon termination. We directly compare the quality of these solutions using a paired Student-t test.

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 [1]. Sampling-based methods have been used to estimate statistical confidence intervals on the optimality gap of stochastic programs by [2], [3]. 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 [4].

In this article, we describe methods to approximate the solution to two-stage stochastic programs with random variables appearing only in the right hand side of the second stage. We present ways to improve the bounds on the approximation error developed by [5] 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 [6], 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 paper 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 article is organized as follows. First, we review the relevant literature on bounding methods in Section 2. In Section 3 the general two-stage stochastic linear programming problem (2SLP) is given, and in Section 4 our sequential bounding algorithm is developed for this problem. In Section 5 we describe four approaches for finding bounds on the optimal second stage primal and dual decision variables. In Section 6 we describe an inventory planning problem with uncertain demand and downward substitutions. We present computational experiments comparing Sample Average Approximation and a variation of the sequential bounding algorithm. Conclusions are given in Section 7.

2 Literature Review

Early work on aggregation bounds was done in the context of deterministic linear programs. In [7, 8], 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 probability 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 [5], 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. This weighting function corresponds to the conditional joint probability distribution associated with the right hand side values, given they belong to a subset in a partition of the support set. Similarly, these weighting functions are used for aggregating variables that depend on the random variable values. [9] 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. [4] 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. [6]

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 [10]. 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 [6] 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. [5] 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. [11] used a related bounding approach, and is the only previous paper 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 paper 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 most cases, demonstrate that sequential bounding produces low cost solutions.

3 Two-stage Stochastic Linear Program

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

$$\begin{aligned} \text{Min} \quad & z(x) = cx + Q(x) \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0, \end{aligned} \tag{1}$$

where $Q(x) = E_{\xi}[Q(x, \xi(\omega))]$ and

$$
\begin{aligned}
Q(x, \xi(\omega)) = \text{Min} \quad & q(\omega)y \\
\text{s.t.} \quad & W(\omega)y = h(\omega) - T(\omega)x \\
& y \geq 0.
\end{aligned} \tag{2}
$$

Throughout this paper different functions are distinguished by name and number of arguments. Thus, $Q(\cdot)$ and $Q(\cdot, \cdot)$ are distinct. The letter A represents 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 is 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. Also, $\omega \in \Omega$ indexes the possible outcomes of ξ. The components of ξ are assumed to be $\mathcal{F}$-measurable with respect to the probability space $(\Omega, \mathcal{F}, \mu)$. The letters W and T represent the *recourse* and *technology* matrix, respectively. We assume Equation (2) 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 [12]), 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 to Equation (2). 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 paper. 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 of Equation (2) can be written as follows:

$$
\begin{aligned}
\text{Max} \quad & \pi(h(\omega) - T(\omega)x) \\
\text{s.t.} \quad & \pi W(\omega) \leq q(\omega) \\
& \pi \text{ unrestricted in sign.}
\end{aligned} \tag{3}
$$

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 (1), additional assumptions must be made (see [9]). 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:

$$\text{Max} \qquad vb + E_\xi[\pi(\omega)h(\omega)] \tag{4}$$

$$\text{s.t.} \quad vA + E_\xi[\pi(\omega)T(\omega)] \leq c \tag{5}$$

$$\pi(\omega)W(\omega) \leq q(\omega) \text{ a.s.}, \tag{6}$$

where v is a row vector of length m representing the first stage decision variables, and $\pi(\omega)$ is a row vector of length m_2 representing the second stage decision variables. Here, the constraints in Equations (5) and (6) are the first and second stage constraints respectively.

Applying the dual form of the results of [13] to Equation (4), an optimal solution, $v = v^*$ and $\pi = \pi^*$, 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) are bounded, and (4) π 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 π^* in Equation (4) are provided by [14]. 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 [15].

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.

4 Sequential Bounding

Sequential bounding involves partitioning the set Ξ into ν disjoint sets so 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:

$$\begin{aligned} \text{Min} \quad & \hat{z}^{\nu}(x, \{\hat{\xi}^k\}_{k=1}^{\nu}) = cx + \textstyle\sum_{k=1}^{\nu} p^k Q(x, \hat{\xi}^k) \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0. \end{aligned} \tag{7}$$

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:

$$\begin{aligned} \text{Min} \quad & z^{\nu}(x) = cx + Q^{\nu}(x) \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0, \end{aligned} \tag{8}$$

where $Q^{\nu}(x) = E_{\xi}[Q(x, \xi) | \xi \in \hat{\Xi}] = \sum_{k=1}^{\nu} p^k Q(x, \xi^k)$. 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 (4) is the dual linear program of Equation (8). We denote this dual optimal solution as $(v^{\nu *}, \{\pi^{k,\nu *}\}_{k=1}^{\nu})$.

We define each S^k as a hyper-rectangle aligned with the coordinate axes. In other words, for each $k = 1, \ldots, \nu$,

$$S^k = \Xi \cap \left([a_1^{(k)}, b_1^{(k)}] \times \ldots \times [a_J^{(k)}, b_J^{(k)}] \right), \tag{9}$$

where $J = m_2(1 + n + n_2) + n_2$ 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, we note that some of the values of $a_i^{(k)}$ and $b_i^{(k)}$ for $i = 1, \ldots, m_2$ can be $\pm\infty$; so, Ξ need not be bounded. In Section 5 we discuss how to compute lower and upper bounds on the optimal value of the second stage recourse decisions for each hyper-rectangle.

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 [9], but our proposition and proof allows for equality constraints in the primal problem and non-zero lower bounds for the second stage dual decision variable values, which lead to tighter bounds on $Q(x)$. [9] 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 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:

$$\hat{\xi}^{k,1} = Vec(h^k, T^k, \hat{W}^{k,1}, \hat{q}^{k,1}) \tag{10}$$

and

$$\hat{\xi}^{k,2} = Vec(\hat{h}^{k,2}, \hat{T}^{k,2}, \hat{W}^{k,2}, q^k). \tag{11}$$

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

$$\begin{aligned} 0 \leq y^{k,LB} \leq y^*(x,\xi) \leq y^{k,UB} \\ \pi^{k,LB} \leq \pi^*(x,\xi) \leq \pi^{k,UB} \end{aligned} \tag{12}$$

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. *Let $\hat{\xi}^{k,1}$ and $\hat{\xi}^{k,2}$ be defined as in Equations (10) and (11). Also, let $y^{k,UB}$,$y^{k,LB}$,$\pi^{k,UB}$, and $\pi^{k,LB}$ be defined as in Equation (12). 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

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

Proof. Proof: See Appendix A. □

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 Equation (2) 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 [16] and [17]. This establishes bounds on the objective function value associated with this first stage solution, since:

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

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 W and q are constant and the assumptions of Proposition 1 hold, then*

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

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

Proof. Proof: See Appendix B. □

We emphasize that the bounds of Proposition 1 and Corollary 1 are deterministically valid. This is in contrast to statistical confidence bounds such as those discussed in [2] for estimating the true objective function value of a feasible or optimal first stage solution. Notice that the expression defining the estimation error, ϵ_2^{ν}, is a summation of integrals. The integrands depend on a linear function of the difference between random variables in ξ and their conditional mean values, given that $\xi \in S^k$ for each $k = 1, \ldots, \nu$. 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. This is the case when the components of the random vector are independent or expressible as a linear transformation of independent random variables for example.

The use of 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 [18]. In [11], a less general version of Corollary 1 was applied to an appointment scheduling problem. Their version was used with iterative refinement

of the partition to produce a lower and upper bound on the optimal objective function value for 2SLP that converges to $z(x^*)$ as ν increases. During the ν^{th} refinement of the partition, ξ was restricted to the k^{th} cell, S^k, and the optimal values of y in the recourse problem of Equation (2), with $x = x^{\nu *}$ and $\xi \in S^k$, were restricted to a subset of the possible values associated with $\xi \in \Xi$. The authors used these restrictions 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 article, we generalize this algorithm to 2SLPs consistent with the assumptions of Section 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.

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 5.
4: Calculate $\epsilon_2^{\nu}(x^{\nu *}, \{\xi^k\}_{k=1}^{\nu})$ with the bounds calculated in Step 3. If $\epsilon_2^{\nu}(x^{\nu *}, \{\xi^k\}_{k=1}^{\nu}) \leq tolerance$, 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.

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 6, we will 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. If ξ is discretely distributed, then 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 (1), if $Q(x, \xi)$ 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 $\nu \rightarrow \infty$ (this is shown by [18]). In our computational experiments in Section 6, random variables only appear in h; so, convergence is guaranteed. Convergence is also possible with random variables appearing elsewhere. [19] provides such conditions, 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 for which cell of the partition to

split, which coordinate direction the split should be made, and which point along the coordinate direction to make the split. To decide on an appropriate approach for the experiments in Section 6, we considered the following instance of Equation (1):

$$\begin{aligned} \text{Min} \quad & z(x_1) = Q(x_1) \\ \text{s.t.} \quad & x_1 \geq 0, \end{aligned} \tag{13}$$

where $Q(x_1) = E_\xi[Q(x_1, \xi(\omega))]$, $q_1 \geq q_2 \geq 0$, and

$$\begin{aligned} Q(x, \xi(\omega)) = \ \text{Min} \quad & q_1 y_1 + q_2 y_2 \\ \text{s.t.} \quad & x_1 + y_1 - y_2 = h_1(\omega) \\ & y_1, y_2 \geq 0. \end{aligned} \tag{14}$$

This is equivalent to the famous Newsvendor Problem, and a special case of the inventory problem we introduce in Section 6. See [20, pg. 250] for a more detailed description of the Newsvendor Problem, including a formula for the optimal solution. The details and solution of this problem do not concern us here, but we use this special case to motivate our approach for sequential bounding. When $h_1(\omega)$ is a continuous random variable with a uniform probability distribution, the following proposition shows the best way to split cells in the partition of the random variable support.

Proposition 2. *For the Newsvendor Problem in Equation (13), let $h_1(\omega)$ be a continuous random variable with uniform probability distribution, $U[a, b]$. Also let $\pi_1^{k,UB} = q_1$ and $\pi_1^{k,LB} = -q_2$ be the upper and lower bound respectively on the dual decision variable of Equation (14) for $k = 1, \ldots, \nu$ and every iteration $\nu \geq 1$. At iteration ν the sequential bounding approximation error, $\epsilon_2^\nu(x^{\nu *}, \{\xi^k\}_{k=1}^\nu)$, is minimized by splitting the cell with the largest corresponding terms in the expression defining ϵ_2^ν in Proposition 1. In other words, the error is minimized by splitting the cell with index equal to:*

$$k^* = \underset{k \in \{1, \ldots, \nu'\}}{\text{argmax}} \left\{ q_1 \int_{h_1^k}^{b_1^{(k)}} \frac{(h_1 - h_1^k)}{b - a} dh_1 + q_2 \int_{a_1^{(k)}}^{h_1^k} \frac{(h_1^k - h_1)}{b - a} dh_1 \right\}$$

and replacing cell $\left[a_1^{(k^*)}, b_1^{(k^*)}\right]$ with cells $\left[a_1^{(k^*)}, \frac{a_1^{(k^*)} + b_1^{(k^*)}}{2}\right]$ and $\left[\frac{a_1^{(k^*)} + b_1^{(k^*)}}{2}, b_1^{(k^*)}\right]$ at each iteration $\nu' \leq \nu$.

Proof. Proof: See Appendix C. □

In Section 6 we use the results of Proposition 2 as a guide for partitioning during sequential bounding. In particular, at the end of iteration ν we select the cell index $k \in \{1, \ldots, \nu\}$ and row index $i \in \{1, \ldots, m_2\}$ with the maximum corresponding terms in the expression defining ϵ_2^ν in Proposition 1. In other words we

select the indices that maximize the following quantity:

$$\pi_i^{k,UB} \int_{S^k} (h_i - h_i^k) d\mu - \pi_i^{k,LB} \int_{S^k} (h_i^k - h_i) d\mu.$$

Next, we split cell k along the i^{th} coordinate direction. As was done in Proposition 2, we make the split at the conditional mean value of the cell. This partitioning scheme is not guaranteed to be the most effective for more general problems than the Newsvendor Problem with uniform demand, but we use this special case to motivate our approach for sequential partitioning. We refer to [21] for a more complete description of partitioning, including a number of alternative schemes.

If we use the cell partitioning scheme described in Proposition 2 for the Newsvendor Problem of Equation (13), then the approximation error will reduce as quickly as possible, compared to other paritioning schemes. The next proposition shows the rate of convergence is sublinear.

Proposition 3. *For the Newsvendor Problem in Equation (13), let $h_1(\omega)$ be a continuous random variable with uniform probability distribution, $U[a, b]$. Also let $\pi_1^{k,UB} = q_1$ and $\pi_1^{k,LB} = -q_2$ be the upper and lower bound respectively on the dual decision variable of Equation (14) for $k = 1, \ldots, \nu$ and every iteration $\nu \geq 1$. The sequential bounding approximation error $\epsilon_2^\nu(x^{\nu *}, \{\xi^k\}_{k=1}^\nu)$ converges sublinearly to zero as a function of ν.*

Proof. Proof: See Appendix D. □

Since the Newsvendor Problem is easily solved analytically, we might expect the sequential bounding algorithm to converge rapidly for this problem when we employ the best partitioning scheme possible. Proposition 3 demonstrates that this is not the case and suggests potential for improving sequential bounding beyond the partitioning scheme alone.

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 accommodated in the sequential bounding algorithm if we assume the first stage feasible region is bounded. In this case, 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 (8) 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.

4.1 Restricted Recourse Bounds

In this subsection we explain how to use restricted recourse to generate an upper bound on $z(x^{\nu *})$. We refer to [6] for a detailed description of restricted recourse. The upper bound is obtained 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, even though the optimal objective function value of the recourse problem is no greater than that for the restricted recourse problem. When this is the case, we use the smaller upper bound provided by the restricted recourse problem. This approach is used to improve the convergence of the sequential bounding algorithm. In particular, we add constraints associated with each cell of the random variable support partition if they reduce the upper bound on $z(x^{\nu *})$.

First, we define the restricted recourse problem:

$$\widetilde{Q}(x,\xi) = \text{Min} \quad qy \tag{15}$$

$$\text{s.t.} \quad Tx + Wy = h$$

$$\widetilde{W}y + \widetilde{V}y' = \widetilde{h} \tag{16}$$

$$y, y' \geq 0.$$

We see the model of Equation (15) is the same as Equation (2) with one or more additional constraints given by Equation (16). Since the vector of decision variables y' is not involved in the objective or constraints of Equation (2), the optimal objective function for Equation (15) is greater than or equal to that for Equation (2). We assume Equation (15) is feasible for all x and all $\xi \in \Xi$, but there are no restrictions on the number of constraints or new decision variables added. 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 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 $\xi \in \Xi$. So, 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 Equation (15). We define $\widetilde{\pi}^*(x,\xi)$ to be a vector of optimal dual decision variables for Equation (15) 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 optimal second stage cost for the PMVP with restricted recourse. The following corollary gives an upper bound for $z(x^{\nu *})$:

Corollary 2. *Let the restricted recourse problem be defined as in Equation (15), and let* $\widetilde{y}^*$, $\widetilde{\pi}^*$, *and* $\widetilde{Q}^\nu$ *be*

defined as in the preceding paragraph. If W and q are constant and the hypotheses of Proposition 1 hold, then

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

and

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

where

$$\begin{aligned}\widetilde{\epsilon}_2^{\nu}(x^{\nu*}) = \quad & \textstyle\sum_{k=1}^{\nu}\sum_{i=1}^{m_2} \widetilde{\pi}_i^{k,UB} \int_{S^k} \left[(h_i - h_i^k) - (T_{i,\cdot} - T_{i,\cdot}^k)x^{\nu*}\right]^+ d\mu \\ & - \textstyle\sum_{k=1}^{\nu}\sum_{i=1}^{m_2} \widetilde{\pi}_i^{k,LB} \int_{S^k} \left[-(h_i - h_i^k) + (T_{i,\cdot} - T_{i,\cdot}^k)x^{\nu*}\right]^+ d\mu\end{aligned}$$

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

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

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

Proof. Proof: This follows directly from the definition of the restricted recourse problem in Equation (15), Corollary 1 and the fact that $x^{\nu*}$ and $y^{k,\nu*}$, for $k = 1, \ldots, \nu$, are feasible and optimal for the PMVP with restricted recourse. □

In closing this subsection, notice that $\widetilde{\epsilon}_2^{\nu}$ in Corollary 2 is very similar to ϵ_2^{ν} in Corollary 1. In particular, bounds for feasible decisions in the dual of Equation (15) can be used as bounds for feasible decisions in Equation (3). Thus, if we define $\widetilde{\pi}^{UB}$ and $\widetilde{\pi}^{LB}$ such that

$$\widetilde{\pi}^{LB} \leq \widetilde{\pi} \leq \widetilde{\pi}^{UB},$$

for all feasible dual variables $\widetilde{\pi}$ corresponding to the primal constraints $Tx + Wy = h$ in Equation (15) and for all $\xi \in \Xi$, then setting $\pi^{k,LB} = \widetilde{\pi}^{LB}$ and $\pi^{k,UB} = \widetilde{\pi}^{UB}$ for all $k = 1, \ldots, \nu$ will produce a valid value for the expression defining ϵ_2^{ν} in Proposition 1. If we also set $\widetilde{\pi}^{k,UB} = \widetilde{\pi}^{UB}$ and $\widetilde{\pi}^{k,LB} = \widetilde{\pi}^{LB}$ for all $k = 1, \ldots, \nu$, then $\widetilde{\epsilon}_2^{\nu}(x^{\nu*}) = \epsilon_2^{\nu}(x^{\nu*}, \{\xi^k\}_{k=1}^{\nu})$ when W is constant in Equation (2). Methods for finding such bounds on the recourse decisions are the subject of Section 5.

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. Although it is possible to relax these assumptions somewhat, we keep them for ease of presentation.

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:

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

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 following 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 non-decreasing in r_i^k, the objective function is the sum of a linear term and single-variable functions that are convex in each of the decision variables. We can easily obtain a piecewise linear outer approximation of each function at evenly spaced intervals. 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 Section 1.3 [22] 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 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.

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 $k = 1, \ldots, \nu$. Since $x^{\nu *} \geq 0$ in the PMVP of 2SLP, we begin by defining 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:

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

Notice $a^{(k)}$ and $b^{(k)}$ are used for the limits of the hyper-rectangle S^k, defined in Equation (9).

When Equation (3) 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 $\xi \in S^k$:

$$\begin{aligned} \text{Max} \quad & \pi_\ell && (17) \\ \text{s.t.} \quad & L_i^{(k)} \leq r_i \leq U_i^{(k)}, \forall i \\ & W_{i,j}^{k,LB} \leq r_{i,j} \leq W_{i,j}^{k,UB}, \forall i, j \\ & q_j^{k,LB} \leq s_j \leq q_j^{k,UB}, \forall j \\ & \textstyle\sum_{j=1}^{n_2} r_{i,j} y_j = r_i, \forall i && (18) \\ & \textstyle\sum_{i=1}^{m_2} \pi_i r_{i,j} \leq s_j, \forall j && (19) \\ & \textstyle\sum_{j=1}^{n_2} s_j y_j \leq \sum_{i=1}^{m_2} \pi_i r_i && (20) \\ & y_j \geq 0, r_{i,j}, s_j, \pi_i, \text{ and } r_i \text{ unrestricted in sign } \forall i, j. \end{aligned}$$

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:

$$\begin{aligned}
\text{Min} \quad & \pi_\ell && (21)\\
\text{s.t.} \quad & L_i^{(k)} \le r_i \le U_i^{(k)}, \forall i \\
& W_{i,j}^{k,LB} \le r_{i,j} \le W_{i,j}^{k,UB}, \forall i,j \\
& q_j^{k,LB} \le s_j \le q_j^{k,UB}, \forall j \\
& \textstyle\sum_{j=1}^{n_2} r_{i,j} y_j = r_i, \forall i && (22)\\
& \textstyle\sum_{i=1}^{m_2} \pi_i r_{i,j} \le s_j, \forall j && (23)\\
& \textstyle\sum_{j=1}^{n_2} s_j y_j \le \sum_{i=1}^{m_2} \pi_i r_i, && (24)\\
& y_j \ge 0, r_{i,j}, s_j, \pi_i, \text{ and } r_i \text{ unrestricted in sign } \forall i,j.
\end{aligned}$$

Equations (18) and (22) are the primal constraints associated with the recourse problem in Equation (1). Equations (19) and (23) are the dual constraints associated with Equation (6). Equations (20) and (24) 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.

5.1 Global Bounds

Consistent with the assumptions of Section 3, we assume $\{\pi | \pi W \le q\}$ is a bounded set. Otherwise, it is possible to add constraints to $\pi W \le q$, since every optimal solution to Equation (6) is bounded. Assuming W and q are constant, we begin by describing how to obtain global bounds, $\pi_\ell^{k,LB}$ and $\pi_\ell^{k,UB}$, independent of the cell k in the partition. Since the feasible region remains the same in the dual of Equation (2) with $x = x^{\nu *}$ and for all values of $\xi \in \Xi$, we can establish an upper bound on the dual variable $\pi_\ell(x^{\nu *}, \xi)$ by

solving the following linear program:

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

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 [5], such global bounds on the decision variables are often readily apparent from examination of the constraints.

5.2 Linear Programming Relaxation

In this section, we use a linear programming relaxation 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.

The first step of the linear programming relaxation 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 [23] and is one of the techniques used in the more general Reformulation-Linearization Technique [see 24, and references therein].

Next, 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:

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

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:

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

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

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

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

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

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, Equation (2) with $x = x^{\nu *}$, 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 (17) 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:

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

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) 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 $F = \{j :$ constraint j is not active in Equation (3) $\}$.

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 complementary 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 $C = \{y | Wy \leq U^{(k)}, Wy \geq L^{(k)}, y \geq 0$ and $y_j = 0$ for all $j \in F\}$. Minimizing each variable y_j in turn over this set C will provide lower bounds $\{y_j^{LB,*}\}_{j=1}^{n_2}$ 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 2.

Algorithm 2 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 $\pi_\ell^{k,LB} = \min\{\pi_\ell | \pi W_{\cdot,j} \leq q_j \text{ for } j \notin E, \pi W_{\cdot,j} = q_j \text{ for } j \in E\}$ for all ℓ.
3: Set $\pi_\ell^{k,UB} = \max\{\pi_\ell | \pi W_{\cdot,j} \leq q_j \text{ for } j \notin E, \pi W_{\cdot,j} = q_j \text{ for } j \in E\}$ for all ℓ.
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.

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 1 for the case where there are only two second stage dual variables π_1 and π_2 associated with the recourse problem of Equation (3) with $x = x^{\nu *}$ and $\xi \in S^k$. The shaded square represents all possible cost vectors centered at π^{k*}, a second stage dual optimal solution of the PMVP. Recall the second stage dual problem for the PMVP is Equation (3) with $x = x^{\nu *}$ and $\xi = \xi^k$.

For any possible dual cost vector, we can graphically determine the optimal dual basic feasible solution in Figure 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 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

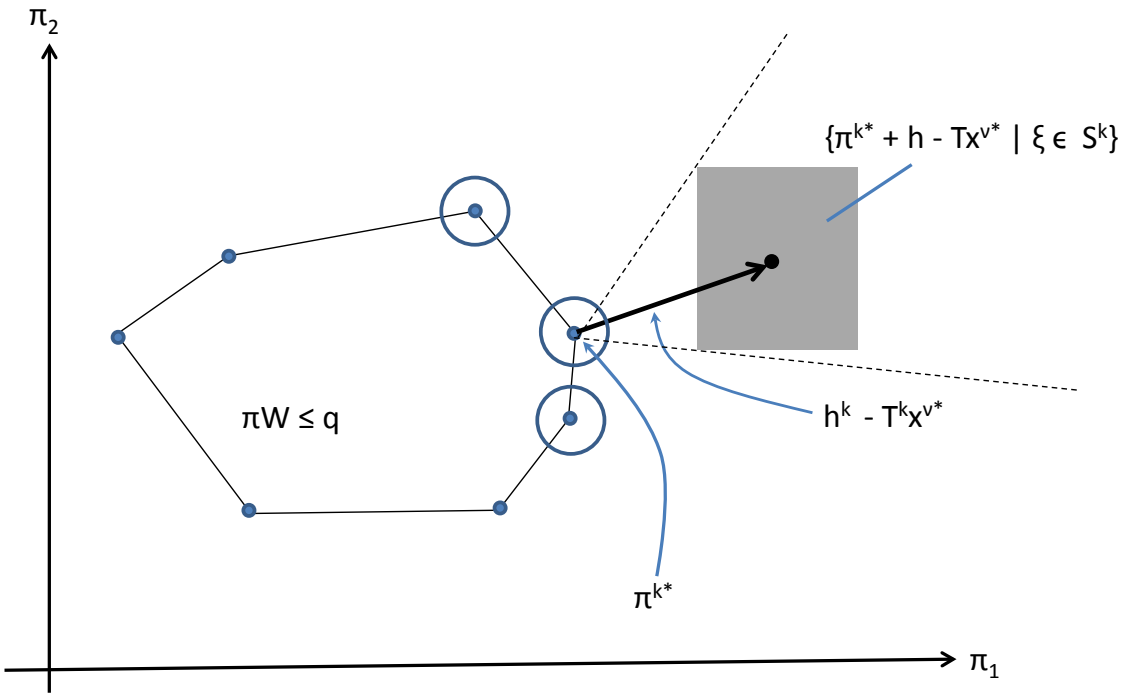

Figure 1: A 2-dimensional example of the dual problem given by Equation (3) with $x = x^{\nu *}$ and $\xi \in S^k$. From the possible dual cost vectors, we see that only the circled basic feasible solutions need to be considered when determining bounds for π_1 and π_2.

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 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 the optimal value of π_ℓ found by solving the corresponding DBLP is improved by solving the following disjunctive linear programming problem:

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

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

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

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 [25], 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.

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. We also assume the i^{th} column of any basis will be from column i or column $m_2 + i$ from W.

A number of important two-stage stochastic linear programs have this special structure, including the simple recourse problem described in [26], as well as appointment and operating room scheduling problems (see [11], [27], and [28]). Tandem queueing networks described in [29] 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. 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:

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

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

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:

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

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

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:

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

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

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

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

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 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 5 as well as restricted recourse and optimized outcomes described in Section 4.

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 manufacturing 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 the work done by [30] and [31], 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:

$$\min \quad \sum_{i=1}^{N}(c_i x_i + K_i z_i) + E_\xi \left[\sum_{i=1}^{N}(h_i u_i^+(\omega) + p_i u_i^-(\omega)) + \sum_{i=1}^{N}\sum_{j=i}^{N} s_{i,j} w_{i,j}(\omega)\right] \tag{25}$$

$$\text{s.t.} \quad x_i \leq M z_i, \text{ for } i = 1, \ldots, N; \omega \in \Omega \tag{26}$$

$$\sum_{i=1}^{j} w_{i,j}(\omega) + u_j^-(\omega) = \xi_j(\omega), \text{ for } j = 1, \ldots, N; \omega \in \Omega \tag{27}$$

$$\sum_{j=i}^{N} w_{i,j}(\omega) + u_i^+(\omega) = x_i, \text{ for } i = 1, \ldots, N; \omega \in \Omega \tag{28}$$

$$x_i, w_{i,j}, u_i^+(\omega), u_j^-(\omega) \geq 0; z_i \in \{0, 1\} \text{ for } i, j = 1, \ldots, N; i \leq j \text{ and } \omega \in \Omega.$$

Here, M is a sufficiently large quantity to ensure that Equation (26) is a redundant constraint when $z_i = 1$. Equation (27) is the set of N second stage demand balance constraints, and Equation (28) is the set of N second stage supply balance constraints. Since this problem has a total of $2N$ second stage constraints and $2N + \frac{N(N+1)}{2}$ second stage decision variables, the recourse matrix does not have the special structure described in Section 5.5. Therefore, the approach of that section can not be used to solve this problem.

6.2 Implementation of Restricted Recourse Bounds

Proposition 4 below was used to develop a restricted recourse problem to apply the bound of Section 4.1, providing an alternative upper bound for the sequential bounding algorithm.

Proposition 4. *The recourse linear program for the inventory planning problem with downward substitutions and N products is:*

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

Let ℓ be a fixed integer such that $1 \leq \ell < N$, and consider the following restricted recourse linear program:

$$\begin{aligned} \widetilde{Q}_N(x,\xi) = \min \quad & \textstyle\sum_{i=1}^{N}(h_i u_i^+ + p_i u_i^-) + \sum_{i=1}^{N}\sum_{j=i}^{N} s_{i,j} w_{i,j} \\ s.t. \quad & \textstyle\sum_{i=1}^{j} w_{i,j} + u_j^- = \xi_j, \ \textit{for } j = 1,\ldots,N \\ & \textstyle\sum_{j=i}^{N} w_{i,j} + u_i^+ = x_i, \ \textit{for } i = 1,\ldots,N \\ & w_{i,j} = 0 \ \textit{for all } i \leq \ell \ \textit{and } j > \ell \qquad (29) \\ & w_{i,j}, u_i^+, u_j^- \geq 0 \ \textit{for } i,j = 1,\ldots,N \ \textit{and } i \leq j. \end{aligned}$$

Notice that the restricted recourse linear program is the recourse linear program with additional constraints in Equation (29). The restricted recourse linear program for N products decomposes into one recourse problem with ℓ products and another with $N-\ell$ products: $\widetilde{Q}_N(x,\xi) = Q_\ell(x,[\xi_1,\ldots,\xi_\ell]^\intercal) + Q_{N-\ell}(x,[\xi_{\ell+1},\ldots,\xi_N]^\intercal)$.

Proof. Proof: No product with index less than or equal to ℓ will ever be substituted for a product with index greater than ℓ in the restricted recourse linear program. 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 restricted 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 the problem of Equation (25), corresponding to the problem of Equation (2) in the general case, we obtained a valid restricted recourse problem, as described in Section 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 4, 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 [23], described in Section 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.

6.3 Computational Experiments

We used SAA for comparison with sequential bounding. SAA is a Monte Carlo sampling-based approach where demand outcomes are randomly sampled and given equal probability of occurrence. This sample is used to construct a linear program that is an approximation of the stochastic program. Following the approach of [2], we repeatedly sampled demand outcomes and solved such a linear program. The optimal objective function values of these linear programs are used to generate a statistical lower bound on the optimal objective function value of the original stochastic program. We refer to the collection of linear programs as the lower bounding problems. After a feasible first stage solution is selected, we repeatedly generated random demand outcomes and solved the second stage recourse linear program associated with the first stage solution and demand outcomes. The optimal objective function values of these second stage recourse linear programs are used to generate a statistical upper bound on the optimal objective function value for the original stochastic program.

In order to calculate these upper and lower statistical bounds, four values must be specified: the number of scenarios for each lower bounding problem, the number of lower bounding problems, a feasible first stage solution, and the number of demand outcomes used to generate the statistical upper bound. For comparison

purposes, we wished to obtain a feasible first stage solution and calculate upper and lower statistical bounds within a given time limit. We used an iterative approach described by [32], where a feasible first stage solution and statistical bounds are obtained at the first iteration. In each subsequent iteration the statistical lower bound is modified, and the first stage solution and statistical upper bound either remain the same or are both modified.

We solved one new sampled instance of the lower bounding problem at each iteration. Iterations continued until the time limit was reached. Thus, the total number of iterations at the end of the time limit is equal to the number of lower bounding problems solved. 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 new feasible first stage solution, and solved for a statistical estimate of an upper bound on the optimality gap. Following the choices made by [32], we selected 20 scenarios for the lower bounding problem and 1000 demand outcomes for the upper bounding problem. In addition to 20 scenarios for the lower bounding problem (labeled "SAA 20"), we also considered 100 scenarios for the lower bounding problem (labeled "SAA 100") 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 [32]. 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 8 threads (2.93 GHz) Intel Core i7 processor.

An illustrative example of the partition of the random variable support set is provided in Figure 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 labeled 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 the upper and lower bounds on the optimal objective function value are plotted at each iteration of the sequential bounding algorithm. The difference is a bound on 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 [31]. 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

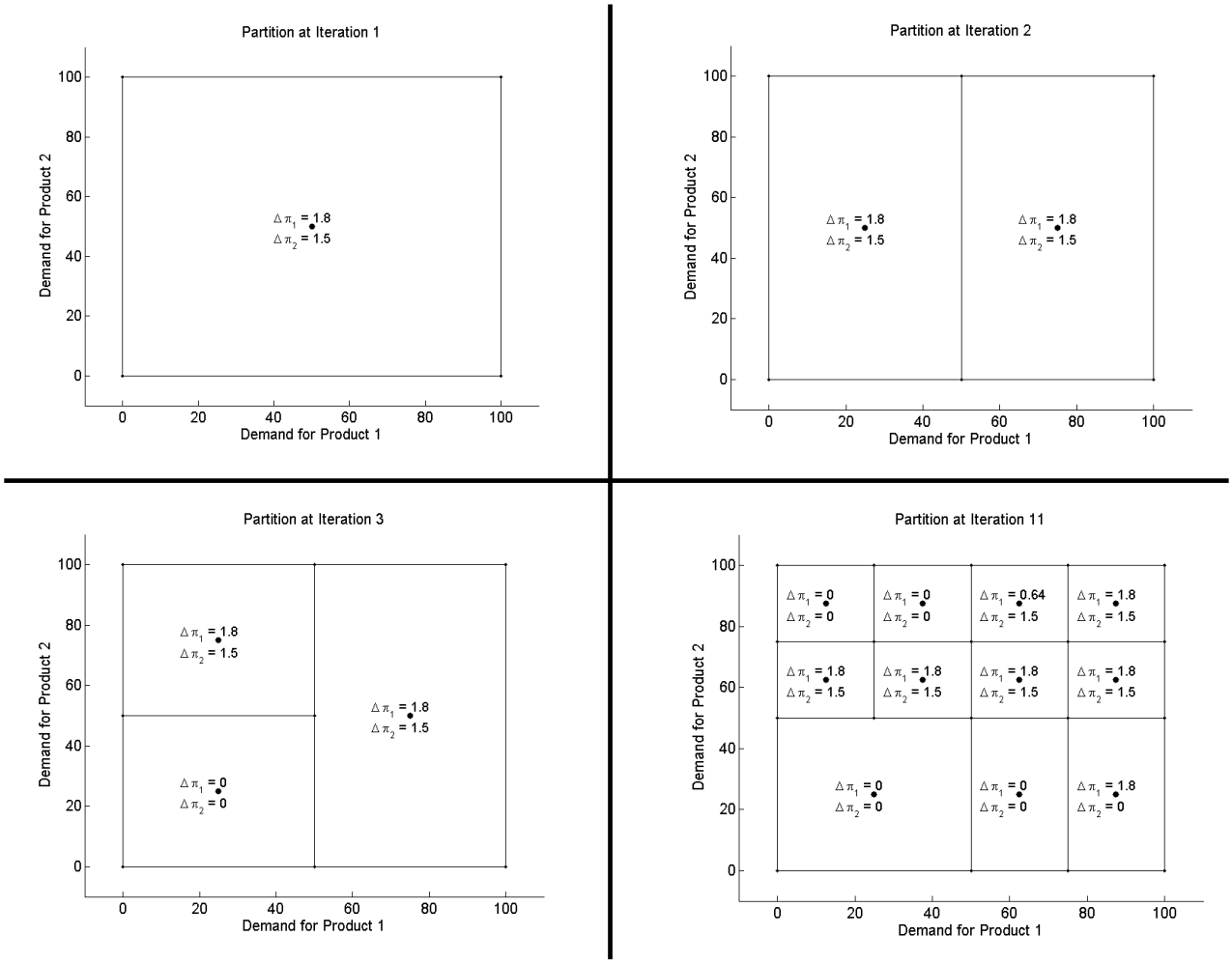

Figure 2: A 2-dimensional example of the partition of the random variable support set at iterations 1, 2, 3, and 11 of the LP-relaxation version of the sequential bounding algorithm. Conditional mean values for each cell are labeled with a black dot. The quantities $\Delta\pi_1$ and $\Delta\pi_2$ are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

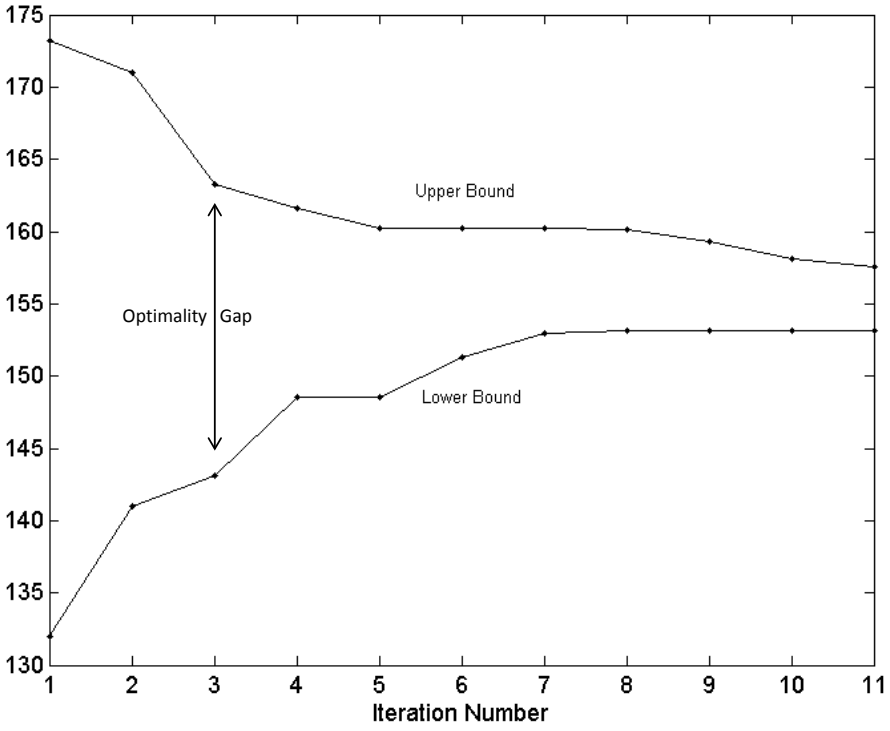

Figure 3: Upper and lower bounds on the optimal objective function value vs. iteration number for the LP-relaxation version of the sequential bounding algorithm. The difference between the bounds is the optimality gap.

Case	N	Demand	Costs
1	5	Norm[100,10]	Low
2	5	Norm[100,10]	High
3	5	Unif[10,100]	Low
4	5	Unif[10,100]	High
5	5	Unif[10,1000]	Low
6	5	Unif[10,1000]	High
7	5	LNorm[100,10]	Low
8	5	LNorm[100,10]	High
9	10	Norm[100,10]	Low
10	10	Norm[100,10]	High
11	10	Unif[10,100]	Low
12	10	Unif[10,100]	High
13	10	Unif[10,1000]	Low
14	10	Unif[10,1000]	High
15	10	LNorm[100,10]	Low
16	10	LNorm[100,10]	High
17	25	Norm[100,10]	Low
18	25	Norm[100,10]	High
19	25	Unif[10,100]	Low
20	25	Unif[10,100]	High
21	25	Unif[10,1000]	Low
22	25	Unif[10,1000]	High
23	25	LNorm[100,10]	Low
24	25	LNorm[100,10]	High

Table 1: Test cases used for computational experiments

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 independent normal random variables; so, integrals appearing in Proposition 1 for ϵ_1 and ϵ_2 simplify after a change of variables. The instances we chose for test problems are given in Table 1. Three different numbers of products were used: 5, 10, and 25 with normally, uniformly, and lognormally distributed demand. We included lognormally distributed demand to include a non-symmetric distribution. Cost parameters associated with the most and least expensive products considered by [31] were used in the tests.

The computational results for the inventory planning problem with downward substitutions are shown in Table 2 for the algorithms described in Section 5 as well as the restricted recourse approach of Section 4.1. Results for the method of optimized outcomes, described in Section 4.2, are presented in Table 3. The method of optimized outcomes could be used to calculate an upper bound at each iteration of sequential bounding. However, we took a two phase approach. In the first phase we conduct sequential bounding with a 900 second time limit. We used two different sequential bounding algorithms: one using global bounds on the optimal dual decision variables and the other using the restricted recourse approach described in Section 4.1. In the second phase we solve the linear program described in Section 4.2 to obtain an improved upper bound. 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.

The run times reported in Table 3 include the total time required for both phases. In this table two results are reported for each case, corresponding to the two sequential bounding algorithms used in the first phase. For each case, the maximum of the two run times was used as the time limit for the algorithms in Table 2. 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 4.2. The results are reported as the percent error, relative to the lower bound, after the last iteration. The total number of iterations is reported in parentheses.

Results for the SAA are shown in Table 4. The maximum run time for both algorithms in Table 3 was used as the time limit for both algorithms in Table 4. 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. 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 approach infinity. The first stage solution associated with the lowest optimal objective value of all lower bounding problems is returned upon termination of the SAA.

The percent error for the sequential bounding algorithms in Table 2 are surprisingly similar. The difference in percent error between algorithms for each case is never more than 6.1 percentage points. The difference in percent error between the algorithms in the table tends to decrease as the number of products, N, increases. In every case the restricted recourse approach of Section 4.1 yields the lowest percent error.

Next we compare Tables 2 and 3. The optimized outcome approach of Section 4.2 outperforms all the other sequential bounding algorithms, with the exception of Case 6, compare the "Rest. Rec." column in Table 2 and the "Global" column of Table 3 . 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 5 and Section 4.1.

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.

We conducted a paired Student-t test to compare SAA and optimized outcomes. We refer to [33] for an example of how to evaluate the quality of two or more stochastic programming solutions using the Student-t test. Let (x^*_{SAA}, z^*_{SAA}) and (x^*_{OO}, z^*_{OO}) denote the first stage solutions found at the final iteration of each algorithm respectively. The paired difference in the objective function value for a particular value of the

Table 2: Experimental results reported as percent error (number of iterations)

Case	Run Time (s)	Global	Comp Slack	Disjunctive	LP Relax	Rest. Rec.
1	935.9	2.2 % (799)	2.2 % (798)	2.2 % (792)	1.7 % (817)	1.3 % (841)
2	959.6	1.5 % (686)	1.4 % (682)	1.4 % (674)	1.1 % (677)	0.7 % (666)
3	972.8	7.7 % (696)	7.3 % (699)	7.3 % (693)	5.1 % (660)	4.5 % (651)
4	970.4	4.7 % (652)	4.6 % (652)	4.6 % (648)	3.3 % (626)	2.4 % (616)
5	922.6	9.0 % (1073)	8.6 % (1072)	8.6 % (1063)	5.4 % (1019)	4.4 % (1010)
6	944.7	8.3 % (1056)	7.5 % (1042)	7.5 % (1036)	3.7 % (1000)	2.2 % (933)
7	933.6	2.2 % (759)	2.1 % (759)	2.1 % (738)	1.6 % (802)	1.2 % (840)
8	962.1	1.4 % (686)	1.4 % (686)	1.4 % (667)	1.1 % (675)	0.8 % (668)
9	973.5	4.0 % (442)	4.0 % (442)	4.0 % (440)	4.0 % (442)	3.7 % (411)
10	1008.2	2.8 % (329)	2.8 % (329)	2.8 % (323)	2.8 % (327)	2.3 % (314)
11	1078.4	16.1 % (391)	16.1 % (392)	16.1 % (390)	16.1 % (390)	15.3 % (374)
12	1012.0	10.5 % (277)	10.5 % (277)	10.5 % (275)	10.0 % (279)	8.9 % (277)
13	935.7	20.1 % (586)	20.1 % (586)	20.1 % (574)	20.0 % (584)	16.2 % (580)
14	966.6	18.5 % (434)	18.5 % (434)	18.5 % (432)	18.2 % (427)	14.5 % (398)
15	950.8	4.0 % (370)	4.0 % (370)	4.0 % (366)	4.0 % (369)	3.6 % (411)
16	992.7	2.8 % (321)	2.8 % (321)	2.8 % (318)	2.8 % (321)	2.4 % (314)
17	3953.2	5.2 % (342)	5.2 % (342)	5.2 % (340)	5.2 % (341)	5.1 % (331)
18	2092.0	4.4 % (141)	4.4 % (142)	4.4 % (140)	4.4 % (141)	4.2 % (143)
19	1403.9	24.5 % (169)	24.5 % (169)	24.6 % (165)	24.5 % (169)	24.0 % (159)
20	2387.6	18.5 % (123)	18.5 % (123)	18.5 % (122)	18.5 % (123)	17.7 % (119)
21	1025.0	31.1 % (292)	31.1 % (292)	31.2 % (287)	31.2 % (289)	29.7 % (280)
22	1794.3	29.9 % (260)	29.9 % (261)	29.9 % (257)	29.9 % (259)	28.5 % (281)
23	2955.9	5.2 % (311)	5.2 % (311)	5.2 % (310)	5.2 % (310)	5.1 % (308)
24	1917.8	4.4 % (143)	4.4 % (143)	4.4 % (142)	4.4 % (143)	4.2 % (142)
Max	3953.2	31.1 % (1073)	31.1 % (1072)	31.2 % (1063)	31.2 % (1019)	29.7 % (1010)
Avg	1377.1	10.0 % (472)	9.9 % (472)	9.9 % (466)	9.3 % (466)	8.5 % (461)

Case	Global	Run Time (s)	Rest. Rec.	Run Time (s)
1	0.7 % (790)	934.7	0.6 % (832)	935.8
2	0.7 % (671)	958.3	0.4 % (652)	959.3
3	3.0 % (674)	973.1	2.5 % (633)	964.9
4	2.3 % (636)	970.0	1.8 % (599)	970.4
5	2.9 % (1061)	916.9	2.3 % (1000)	922.6
6	2.3 % (1035)	936.1	1.1 % (919)	944.7
7	0.7 % (750)	930.0	0.6 % (830)	933.6
8	0.7 % (670)	956.7	0.4 % (652)	962.1
9	2.2 % (433)	973.5	2.2 % (400)	955.4
10	1.8 % (317)	1008.2	1.6 % (302)	994.0
11	8.6 % (372)	1078.4	8.2 % (356)	1063.7
12	6.4 % (265)	997.6	5.8 % (265)	1012.0
13	9.3 % (575)	930.4	9.0 % (571)	935.7
14	7.6 % (425)	966.6	7.2 % (388)	963.5
15	2.3 % (363)	948.8	2.3 % (404)	950.8
16	1.9 % (311)	987.0	1.7 % (304)	992.7
17	3.6 % (206)	3782.0	3.5 % (202)	3953.2
18	3.2 % (105)	1993.2	3.1 % (105)	2092.0
19	15.5 % (140)	1387.1	15.3 % (141)	1403.9
20	13.4 % (91)	2387.6	13.1 % (85)	2324.5
21	18.0 % (278)	991.8	17.9 % (269)	1025.0
22	15.4 % (208)	1794.3	14.9 % (213)	1626.5
23	3.7 % (209)	1793.5	3.7 % (207)	2955.9
24	3.4 % (109)	1601.3	3.2 % (107)	1917.8
Max	18.0 % (1061)	3782.0	17.9 % (1000)	3953.2
Avg	5.4 % (446)	1299.9	5.1 % (435)	1365.0

Table 3: Experimental results for sequential bounding with optimized outcomes, reported as percent error (number of iterations)

Case	Run Time (s)	SAA 20	SAA 100
1	935.9	2.0 % (140416)	1.1 % (24273)
2	959.6	1.6 % (71018)	0.6 % (13318)
3	972.8	10.8 % (53872)	3.6 % (8116)
4	970.4	3.9 % (73964)	1.5 % (11195)
5	922.6	3.8 % (31257)	3.8 % (31279)
6	944.7	2.4 % (35613)	2.4 % (35412)
7	933.6	1.1 % (24365)	1.1 % (24350)
8	962.1	0.6 % (13550)	0.6 % (13506)
9	973.5	1.2 % (18782)	0.3 % (2243)
10	1008.2	0.6 % (14196)	0.4 % (1171)
11	1078.4	5.5 % (8528)	1.2 % (891)
12	1012.0	1.9 % (11365)	0.9 % (955)
13	935.7	1.7 % (7608)	1.7 % (7614)
14	966.6	1.7 % (1429)	1.7 % (1427)
15	950.8	0.4 % (2306)	0.4 % (2305)
16	992.7	0.3 % (1176)	0.3 % (1176)
17	3953.2	0.7 % (2254)	0.3 % (210)
18	2092.0	0.5 % (431)	0.2 % (23)
19	1403.9	2.6 % (726)	1.0 % (35)
20	2387.6	2.6 % (319)	1.2 % (13)
21	1025.0	2.0 % (818)	2.0 % (816)
22	1794.3	1.5 % (96)	1.5 % (96)
23	2955.9	0.2 % (209)	0.2 % (209)
24	1917.8	0.2 % (23)	0.2 % (23)
Max	3953.2	10.8 % (140416)	3.8 % (35412)
Avg	1377.1	2.1 % (21430)	1.2 % (7527)

Table 4: Experimental results for SAA, reported as estimated percent error (number of iterations)

Case	SAA100 - R.R. w/ O.O.	SAA100
1	1.5 (1.2, 1.7)	679.5
2	2.5 (2.1, 3.0)	1834.1
3	1.3 (0.7, 1.9)	427.6
4	2.8 (2.3, 3.3)	1281.2
5	24.7 (17.7, 31.6)	3629.2
6	36.7 (27.7, 45.6)	7152.4
7	1.5 (1.2, 1.8)	680.1
8	2.9 (2.4, 3.3)	1835.1
9	-4.0 (-4.5, -3.6)	1627.6
10	-3.4 (-4.3, -2.6)	5480.6
11	-15.3 (-16.7, -13.9)	1004.8
12	-11.8 (-14.8, -8.8)	3709.6
13	-71.0 (-89.7, -52.3)	8604.1
14	-127.6 (-168.1, -87.2)	22691.4
15	-4.3 (-4.8, -3.8)	1628.1
16	-4.7 (-5.6, -3.8)	5481.6
17	-32.8 (-34.5, -31.0)	6068.2
18	1994.0 (1989.2, 1998.9)	28106.7
19	-196.5 (-201.0, -192.0)	3676.2
20	-303.9 (-320.8, -287.1)	16873.8
21	-1436.4 (-1488.9, -1383.9)	31671.1
22	-4840.3 (-5022.3, -4658.3)	114439.7
23	26.6 (24.9, 28.4)	6129.4
24	-28.0 (-36.5, -19.5)	26083.2

Table 5: 99% confidence bounds for the paired differences in objective function values at the final iteration between SAA100 and Restricted Recourse with Optimized Outcomes. Data is in the form of "mean, (lb,ub)", where lb and ub are the lower and upper bounds for the confidence interval, and mean is the average of paired differences in objective function values for the solution returned by each algorithm over 10,000 sampled demand outcomes. For reference, we include the the average SAA100 objective function value in the final column.

random variable vector ξ is given below:

$$\sum_{i=1}^{N}[c_i(x^*_{SAA,i} - x^*_{OO,i}) + K_i(z^*_{SAA,i} - z^*_{OO,i})] + Q_N(x^*_{SAA}, \xi) - Q_N(x^*_{OO}, \xi).$$

This quantity was calculated for $10,000$ sampled outcomes of the random variable vector ξ using JuMP, a mathematical programming language described by [34] and implemented in the Julia computer programming language, see [35]. We used Clp, an open-source linear programming solver described by [36], for every evaluation of the function Q_N. The p-value was calculated based on the null hypothesis that the paired differences have a mean of zero. In every case we rejected the null hypothesis with a p-value below 0.001. In Table 5 we report the 99.9% confidence bounds on the paired difference for each case. We also include the average objective function value for the SAA solution over all $10,000$ sampled outcomes for the random variable vector ξ.

From Table 5 we see that the Restricted Recourse with Optimized Outcomes approach always does best for the 5 product cases, but it only does better than SAA for two of the 25 product cases. Although SAA is

always the best algorithm in the 10 product cases, we see that average difference in the objective values of the two algorithms, relative to the objective value of SAA, is always less than 2 percent for all of the 5 and 10 product cases. In the 25 product cases, we see more differentiation. Here, SAA is best in the cases with uniformly distributed demand. These are also the cases with the greatest variance in demand. We see that the Optimized Outcomes approach never underperforms by more than 6 percent from the SAA solution's objective function value in the 25 product cases. SAA underperforms by more than 7 percent in case 18. This suggests that SAA is probably the best overall algorithm, especially when the number of products and variance in demand is high, but Optimized Outcomes remains competitive, if not better, when the variance in demand or the number of products is low.

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 [4], for solving two-stage stochastic programs. These approaches also extend the restricted recourse bounds developed by [6] 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 when random variables only appear in the right hand side of the constraints in Equation (3). 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 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. We found that a combination of restricted recourse, described in Section 4.1, decision variable bounds based on a linear programming relaxation of the BNPs, described in Section 5.2, and the approach of optimized outcomes, described in Section 4.2, were consistently better than the global bounds approach for sequential bounding. Although the sequential bounding algorithms developed in this paper required 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 low cost solutions relative to 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. Most of the approaches that we considered did not significantly improve the performance of sequential bounding compared to the implementation in which global bounds were used. Because the global bounds approach is easier to implement, it should be considered in practice.

We compared the restricted recourse with optimized outcomes approach with a Monte Carlo sampling based approach using a paired Student-t test. In 10 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, Sample Average Approximation produced the lowest cost. Overall our results suggest that sequential bounding is capable of producing solutions that are close to, or better, than those generated with Monte Carlo sampling, provided the number of products and demand variance are not both high. However, SAA has a clear tendency to produce the lowest cost solution when the number of products is above 5. It is notable that the bounds obtained from the sequential bounding algorithm are deterministic as opposed to the statistical confidence intervals obtained using SAA.

There are many opportunities for future research. Additional computational results for other two-stage stochastic programming problems would help in evaluating whether or not sequential bounding is a viable solution approach. There may be other special cases or other types of stochastic programs to which the methods we described can be extended. For example, the authors are currently developing a multi-stage version of the sequential bounding algorithm similar to the two-stage version described in this paper.

Acknowledgements

The authors are grateful for the comments of two anonymous referees. The authors thank Karl Kempf and Reha Uzsoy for discussions related to the computational experiments. The authors would also like to thank Reha Uzsoy for discussions during the development of the method described in Section 4.2. This article 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 in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

A Proof of Proposition 1

For ease of presentation, we drop the dependence that every random variable has on the outcome $\omega \in \Omega$. Here, every component of the various ξ, W, q, h, and T vectors and matrices are random variables.

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

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.

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

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.

B Proof of Corollary 1

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 $\epsilon_1^\nu = 0$ when W and q are constant. 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.

C Proof of Proposition 2

The validity of q_1 and $-q_2$ as bounds on the dual decision variable are a direct result of the constraints in the dual of Equation (14).

Since cell k at iteration ν can be different from cell k at iteration $\nu+1$, we will add a superscript ν to variables when the distinction is not otherwise apparent.

At any arbitrary iteration ν, the k^{th} cell $S^{k,\nu} = [a_1^{(k),\nu}, b_1^{(k),\nu}]$. Since h_1 is uniformly distributed, $\xi^{k,\nu} = h_1^{k,\nu} = \frac{a^{(k),\nu}+b^{(k),\nu}}{2}$. The approximation error for sequential bounding is below and follows from the definition of ϵ_2^ν in Proposition 1:

$$\epsilon_2^\nu(x^{\nu *}, \{\xi^{k,\nu}\}_{k=1}^\nu) \quad = \sum_{k=1}^\nu \left\{ q_1 \int_{S^{k,\nu}} \left[h_1 - h_1^{k,\nu}\right]^+ d\mu + q_2 \int_{S^{k,\nu}} \left[h_1^{k,\nu} - h_1\right]^+ d\mu \right\}.$$

The summation is taken over the cell indices, and the sum of exactly two integrals make up the contribution to the approximation error from cell k.

For iteration $\nu+1$, suppose we split cell $S^{k,\nu} = [a_1^{(k),\nu}, b_1^{(k),\nu}]$ at some point t, where $a_1^{(k),\nu} \leq t \leq b_1^{(k),\nu}$. Without loss of generality, we assume all the other cell indices remain the same, but the two new cells will have indices k and $\nu+1$. So, $S^{k,\nu+1} = [a_1^{(k),\nu}, t]$ and $S^{\nu+1,\nu+1} = [t, b_1^{(k),\nu}]$. The change in the approximation

error, $\Delta\epsilon_2^{\nu+1} = \epsilon_2^{\nu+1}(x^{\nu+1*}, \{\xi^{k,\nu+1}\}_{k=1}^{\nu+1}) - \epsilon_2^{\nu}(x^{\nu*}, \{\xi^{k,\nu}\}_{k=1}^{\nu})$, is given below:

$$\begin{aligned}
\Delta\epsilon_2^{\nu+1} &= q_1 \int_{S^{k,\nu+1}} \left[h_1 - h_1^{k,\nu+1}\right]^+ d\mu + q_2 \int_{S^{k,\nu+1}} \left[h_1^{k,\nu+1} - h_1\right]^+ d\mu \\
&+ q_1 \int_{S^{\nu+1,\nu+1}} \left[h_1 - h_1^{\nu+1,\nu+1}\right]^+ d\mu + q_2 \int_{S^{\nu+1,\nu+1}} \left[h_1^{\nu+1,\nu+1} - h_1\right]^+ d\mu \\
&- q_1 \int_{S^{k,\nu}} \left[h_1 - h_1^{k,\nu}\right]^+ d\mu - q_2 \int_{S^{k,\nu}} \left[h_1^{k,\nu} - h_1\right]^+ d\mu \\
&= q_1 \int_{h_1^{k,\nu+1}}^{t} \left[h_1 - h_1^{k,\nu+1}\right] \tfrac{dh_1}{b-a} + q_2 \int_{a_1^{(k),\nu}}^{h_1^{k,\nu+1}} \left[h_1^{k,\nu+1} - h_1\right] \tfrac{dh_1}{b-a} \\
&+ q_1 \int_{h_1^{\nu+1,\nu+1}}^{b_1^{(k),\nu}} \left[h_1 - h_1^{\nu+1,\nu+1}\right] \tfrac{dh_1}{b-a} + q_2 \int_{t}^{h_1^{\nu+1,\nu+1}} \left[h_1^{\nu+1,\nu+1} - h_1\right] \tfrac{dh_1}{b-a} \\
&- q_1 \int_{h_1^{k,\nu}}^{b_1^{(k),\nu}} \left[h_1 - h_1^{k,\nu}\right] \tfrac{dh_1}{b-a} - q_2 \int_{a_1^{(k),\nu}}^{h_1^{k,\nu}} \left[h_1^{k,\nu} - h_1\right] \tfrac{dh_1}{b-a}.
\end{aligned}$$

Since $h_1^{k,\nu+1}$ and $h_1^{\nu+1,\nu+1}$ are both linear with respect to t, the expression above for $\Delta\epsilon_2^{\nu+1}$ reduces to a convex quadratic expression with respect to t. To find the point t that minimizes $\Delta\epsilon_2^{\nu+1}$, solve the equation $\frac{d\Delta\epsilon_2^{\nu+1}}{dt} = 0$ for t. The resulting solution $t = \frac{a_1^{(k),\nu} + b_1^{(k),\nu}}{2}$ is a point inside $S^{k,\nu}$, and is therefore the split point that results in the greatest decrease in the approximation error.

Let $\Delta_k^{\nu} = b_1^{(k),\nu} - a_1^{(k),\nu}$ be the length of the interval of cell k at iteration ν. We can express the sequential bounding approximation error entirely in terms of the cell lengths:

$$\epsilon_2^{\nu}(x^{\nu*}, \{\xi^{k,\nu}\}_{k=1}^{\nu}) \quad = \sum_{k=1}^{\nu} \tfrac{(q_1+q_2)(\Delta_k^{\nu})^2}{8(b-a)}.$$

Notice that each cell contributes one term to the total error, and the term is proportional to the square of the cell's length. Furthermore, the proportionality constant is the same for every cell. So, if we split cell k with length Δ_k^{ν} into two cells, each with length $\frac{\Delta_k^{\nu}}{2}$, then the approximation error will change by the following quantity:

$$\begin{aligned}
\Delta\epsilon_2^{\nu+1} = \quad & \tfrac{(\frac{\Delta_k^{\nu}}{2})^2}{8(b-a)}(q_1+q_2) + \tfrac{(\frac{\Delta_k^{\nu}}{2})^2}{8(b-a)}(q_1+q_2) - \tfrac{(\Delta_k^{\nu})^2}{8(b-a)}(q_1+q_2) \\
= \quad & -\tfrac{(\Delta_k^{\nu})^2}{16(b-a)}(q_1+q_2) \qquad (30)
\end{aligned}$$

In other words, the approximation error is reduced by half of a cell's contribution prior to it being split, and the contribution from all other cells remains unchanged. Thus, we achieve the greatest reduction in the approximation error after one iteration if we split the cell with the largest contribution, which is the cell with the longest length. New cells are always at least as small as the cell from which they were derived. So, there is no advantage to splitting any cell but the longest at each iteration, since splitting a cell with a shorter interval will never produce a longer cell at future iterations.

D Proof of Proposition 3

The validity of q_1 and $-q_2$ as bounds on the dual decision variable are a direct result of the constraints in the dual of Equation (14).

Since cell k at iteration ν can be different from cell k at iteration $\nu+1$, we will add a superscript ν to variables when the distinction is not otherwise apparent.

Let $\Delta_k^\nu = b_1^{k,\nu} - a_1^{k,\nu}$ be the length of the interval of the k^{th} cell $S^{k,\nu}$ at iteration ν. Recall from the proof of Proposition 2 that the sequential bounding approximation error can be expressed in terms of the cell lengths:

$$\epsilon_2^\nu(x^{\nu*}, \{\xi^{k,\nu}\}_{k=1}^\nu) \quad = \sum_{k=1}^\nu \tfrac{(q_1+q_2)(\Delta_k^\nu)^2}{8(b-a)}.$$

The approximation error is a summation over each cell. Each term is proportional to the squared length of the cell interval, and the proportionality constant is independent of the cells. In the proof of Proposition 2 we also determined that the greatest reduction in the error is achieved by splitting the cell with the largest interval length into two cells of equal length. We can deduce that the error is reduced by $\frac{1}{2^k}$ after every 2^k iterations, for any value of k, since every cell's contribution to the error is reduced by half after it is split. For any value of k the error will be reduced by equal amounts for each iteration between iterations 2^k and $2^{k+1}-1$. Below we use the convention where lg is the logarithm to the base 2. Since the error after the first iteration is $\epsilon_2^1(x^{1*}, \{\frac{a+b}{2}\}) = \frac{(q_1+q_2)(b-a)}{8}$, the approximation error at all other iterations can be expressed as follows:

$$\epsilon_2^\nu(x^{\nu*}, \{\xi^{k,\nu}\}_{k=1}^\nu) = \left(1 - \frac{\nu - 2^{\lfloor \lg \nu \rfloor}}{2^{\lfloor \lg \nu \rfloor + 1}}\right) \frac{(q_1+q_2)(b-a)}{2^{\lfloor \lg \nu \rfloor + 3}}.$$

If $\nu+1$ is a power of 2, then the ratio of successive approximation errors is:

$$\frac{\epsilon_2^{\nu+1}(x^{\nu+1*}, \{\xi^{k,\nu+1}\}_{k=1}^{\nu+1})}{\epsilon_2^\nu(x^{\nu*}, \{\xi^{k,\nu}\}_{k=1}^\nu)} = \frac{1 - \frac{1}{2}}{1 - (2^{\lfloor \lg \nu \rfloor} - 1)/(2^{\lfloor \lg \nu \rfloor + 1})}.$$

This ratio converges to 1 as $\nu \to \infty$. If $\nu+1$ is not a power of 2, then the ratio of successive approximation errors is:

$$\frac{\epsilon_2^{\nu+1}(x^{\nu+1*}, \{\xi^{k,\nu+1}\}_{k=1}^{\nu+1})}{\epsilon_2^\nu(x^{\nu*}, \{\xi^{k,\nu}\}_{k=1}^\nu)} = \frac{1 - (\nu + 1 - 2^{\lfloor \lg \nu \rfloor})/(2^{\lfloor \lg \nu \rfloor + 1})}{1 - (\nu - 2^{\lfloor \lg \nu \rfloor})/(2^{\lfloor \lg \nu \rfloor + 1})}.$$

This ratio converges to 1 as $\nu \to \infty$. Therefore, the approximation error converges to zero sublinearly.

References

[1] George Dantzig and Peter Glynn. Parallel processors for planning under uncertainty. *Annals of Operations Research*, 22(1):1–21, December 1990.

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

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

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

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

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

[7] Paul H. Zipkin. Bounds on the effect of aggregating variables in linear programs. *Operations Research*, 28(2):403–418, March 1980.

[8] Paul H. Zipkin. Bounds for Row-Aggregation in linear programming. *Operations Research*, 28(4):903–916, July 1980.

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

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

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

[12] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer, New York, New York, 1997.

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

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

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

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

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

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

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

[20] Steven Nahmias. *Production and Operations Analysis, 4th ed.* McGraw-Hill, 2001.

[21] N.C.P. Edirisinghe and G.-M. You. Second-order scenario approximation and refinement in optimization under uncertainty. *Annals of Operations Research*, 64:143–178, 1996.

[22] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization.* Athena Scientific Belmont, 1997.

[23] Faiz A. Al-Khayyal and James E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.

[24] Hanif D Sherali and Warren P Adams. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, volume 31. Springer, 1998.

[25] J. Dattorro. *Convex optimization and Euclidean distance geometry.* Meboo Publishing USA, 2005.

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

[27] Sakine Batun, Brian T Denton, Todd R Huschka, and Andrew J Schaefer. Operating room pooling and parallel surgery processing under uncertainty. *INFORMS Journal on Computing*, 23(2):220–237, 2011.

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

[29] Wai Kin Victor Chan and Lee Schruben. Optimization models of discrete-event system dynamics. *Operations Research*, 56(5):1218–1237, 2008.

[30] Yehuda Bassok, Ravi Anupindi, and Ram Akella. Single-Period multiproduct inventory models with substitution. *Operations Research*, 47(4):632–642, July 1999.

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

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

[33] David P Morton and Elmira Popova. A bayesian stochastic programming approach to an employee scheduling problem. *Iie Transactions*, 36(2):155–167, 2004.

[34] Miles Lubin and Iain Dunning. Computing in operations research using julia, 2013.

[35] Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing, 2012.

[36] John Forrest, David de la Nuez, and Robin Lougee-Heimer. Clp user guide, 2004.