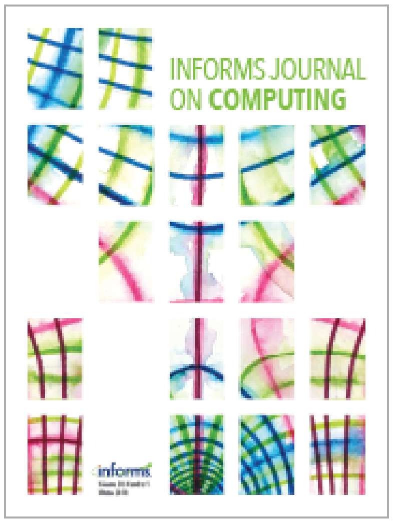


Sampling Scenario Set Partition Dual Bounds for Multistage Stochastic Programs

Ilke Bakir, Natashia Boland, Brian Dandurand, Alan Erera

http://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Vol. 32, No. 1, Winter 2020, pp. 145–163
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

Sampling Scenario Set Partition Dual Bounds for Multistage Stochastic Programs

Ilke Bakir,[a] Natashia Boland,[b] Brian Dandurand,[c] Alan Erera[b]

[a] Department of Operations, University of Groningen, 9712 CP Groningen, Netherlands; [b] H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332; [c] Department of Mathematical Sciences, Royal Melbourne Institute of Technology, Melbourne, Victoria 3000, Australia

Contact: i.bakir@rug.nl, https://orcid.org/0000-0002-3160-526X (IB); natashia.boland@isye.gatech.edu, https://orcid.org/0000-0002-6867-6125 (NB); b.dandurand@rmit.edu.au, https://orcid.org/0000-0003-2873-8878 (BD); alan.erera@isye.gatech.edu, https://orcid.org/0000-0001-8208-0830 (AE)

Received: December 9, 2016
Revised: August 7, 2017; July 30, 2018
Accepted: December 20, 2018
Published Online in Articles in Advance: July 18, 2019

https://doi.org/10.1287/ijoc.2018.0885

Abstract. We consider multistage stochastic programming problems in which the random parameters have finite support, leading to optimization over a finite *scenario set*. There has been recent interest in dual bounds for such problems, of two types. One, known as *expected group subproblem objective* (*EGSO*) *bounds*, require solution of a *group subproblem*, which optimizes over a subset of the scenarios, for all subsets of the scenario set that have a given cardinality. Increasing the subset cardinality in the group subproblem improves bound quality, (*EGSO* bounds form a hierarchy), but the number of group subproblems required to compute the bound increases very rapidly. Another is based on *partitions* of the scenario set into subsets. Combining the values of the group subproblems for all subsets in a partition yields a *partition bound*. In this paper, we consider partitions into subsets of (nearly) equal cardinality. We show that the expected value of the partition bound over all such partitions also forms a hierarchy. To make use of these bounds in practice, we propose random sampling of partitions and suggest two enhancements to the approach: sampling partitions that align with the multistage scenario tree structure and use of an auxiliary optimization problem to discover new best bounds based on the values of group subproblems already computed. We establish the effectiveness of these ideas with computational experiments on benchmark problems. Finally, we give a heuristic to save computational effort by ceasing computation of a partition partway through if it appears unpromising.

History: Accepted by Karen Aardal, Area Editor for Design and Analysis of Algorithms.
Supplemental Material: The online supplement is available at https://doi.org/10.1287/ijoc.2018.0885.

Keywords: stochastic programming • multistage stochastic programming • bounds • dual decomposition

1. Introduction

Stochastic programming provides an approach to optimized decision making that models uncertain parameters with probability distributions. The values of these parameters are typically revealed over time, in a multistage setting, and decisions made at each stage hedge against possible realizations of parameters revealed in future stages. Such multistage stochastic programming models are of enormous practical value and have been studied for many years now, with the literature offering a wealth of theoretical and algorithmic tools for solving them; see, for example, Ruszczynski and Shapiro (2003), Schultz (2003), Sahinidis (2004), Shapiro (2006), Powell (2007, 2014), Birge and Louveaux (2011), Sen and Zhou (2014), and Shapiro et al. (2014) and the many references therein.

Many important practical applications are effectively addressed with multistage stochastic programming models that include integer variables, leading to their formulation as multistage stochastic mixed integer programs (SMIPs). Solving multistage SMIPs is especially challenging, and decomposition approaches are typically required as the size of the scenario tree grows. Here we address the case in which the number of scenarios is small enough to enumerate explicitly, although still too large to permit direct solution of the deterministic equivalent formulation. This case arises often in applications reported in the literature and has been the focus of much research effort. [See, for example, the papers of Powell and Topaloglu (2003), Wallace and Fleten (2003), Nowak et al. (2005), Santoso et al. (2005), Huang and Ahmed (2009), Özaltin et al. (2011), Sandıkçı et al. (2013), Cheung et al. (2015), Erdogan et al. (2015), Veliz et al. (2015), Maggioni et al. (2016), and Sandıkçı and Özaltin (2017), which all address applications.] Computing dual bounds for measuring the quality of primal solutions remains a key challenge in this case.

A recent stream of research work has investigated the computation of dual bounds for SMIPs derived from the solution of *scenario group subproblems*, which each include variables and constraints for only a subset of scenarios. Sandıkçı et al. (2013), in a two-stage setting, explore dual bounds provided by the expected value over all groups of a fixed cardinality, q, of the group subproblem (including a reference scenario). For a fixed q, they name this the *expected group subproblem objective* bound, denoted as *EGSO(q)*. They prove that the $EGSO(q)$ bound is nondecreasing in q and give computational results showing that the bound is strong relative to that given by linear relaxations, even for small group sizes, and that as q increases, the bound quite rapidly approaches the optimal value. However, computing the $EGSO(q)$ bound can be challenging, as it requires solution of an SMIP with q scenarios, for every possible cardinality q subset of the set of all scenarios. This work has since been extended in several directions. Maggioni et al. (2014) generalize the *EGSO* bound idea to multistage linear stochastic programming problems with expectation objectives, while also allowing multiple reference scenarios. Maggioni and Pflug (2016) extend these results to multistage convex problems with concave risk functionals as objective. We discuss the extensions described by Maggioni et al. (2016) and Sandıkçı and Özaltin (2017) and the working paper of Zenarosa et al. (2014b), which are closest to our work, in Section 2.

In this paper, we consider dual bounds that we refer to as *partition bounds*. A single partition bound is obtained by combining the group subproblem values for each group that is a subset in a single fixed partition of the scenario set. Our first contribution is to prove that for partitions into subsets of nearly equal size, the expected value over all such partitions yields a hierarchy of bounds. By "nearly equal," we mean that all groups have size either q, or $q-1$; we call such a partition a *q-partition*. We denote the expected value of the partition bound over all q-partitions by $EP(q)$. Observing that the *EGSO* bound of Sandıkçı et al. (2013) readily extends from the two-stage to the multistage setting, we prove that the $EP(q)$ bound is equal to the $EGSO(q)$ bound in the case that q divides the number of scenarios, L, and interpolates between the $EGSO(q-1)$ and $EGSO(q)$ values otherwise. We thus obtain the result that the $EP(q)$ bound nondecreases monotonically in q.

Although this hierarchical property of the expected q-partition bounds is primarily of theoretical interest, q-partitions can provide bounds in practice. By solving $\lceil L/q \rceil$ SMIPs, each with q or $q-1$ scenarios, a single partition bound results. In empirical studies, we found that the distribution of such partition bounds for typical benchmark instances is not very far from symmetric, so the probability that a partition has above-average value is not small. Similar observations can be made from distributions provided in Sandıkçı and Özaltin (2017). Thus, calculating a partition bound for only a few, randomly sampled partitions and taking the best of these bounds is highly likely to result in a dual bound greater than the corresponding *EP* bound, and, hence, greater than the *EGSO* bound. We are thus motivated to seek partition bounds by random sampling of partitions.

Random sampling has been a valuable tool for tackling stochastic programs. In particular, the sample average approximation (SAA) method (see, e.g., Kleywegt et al. 2002, Ruszczynski and Shapiro 2003) has proved to be of great utility. In SAA, a set of scenarios is sampled, the sampled problem is solved, and the resulting solution evaluated using a, usually larger, sample of the scenarios. This process is repeated a number of times, allowing statistical bounds, including confidence intervals, to be computed (Norkin et al. 1998, Mak et al. 1999). In SAA, the expected value of the group subproblem is known to provide a dual bound (Kleywegt et al. 2002). Indeed, modulo replacement,[1] the average SAA group subproblem value can be interpreted as an estimate of the *EGSO* bound for groups of the same size.

Here, we propose to randomly *sample partitions* of the set of scenarios, with replacement, and to compute the best partition bound over the partitions sampled. The practical value of this idea is illustrated on benchmark instances, showing that randomly sampled q-partition bounds are better than those determined to be statistically likely using SAA with the same computational effort. The sampling approach is enhanced by leveraging the scenario tree structure and by constructing optimally recombined partitions from scenario subsets that are previously used in the algorithm. Empirical tests show that both these ideas can significantly improve the quality of the final bound. On benchmark problems, sampling as few as 30 partitions closes 94% of the wait-and-see gap, on average, for moderate group sizes (size 10), which is 79% more than that can be closed with the SAA estimates. By using the observation that the bound from a given partition can be heuristically estimated partway through its computation, we suggest a method to improve the efficiency of the approach. Strategies that compare the heuristically estimated partition bound with the best such bound found so far, to terminate the partition bound computation partway through, are tested empirically. In many cases, such strategies substantially reduce the computational effort with very little impact on the quality of the final bound.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of related literature. In Section 3, we introduce our notation and review the *EGSO* bounds and their extension to the multistage case. In Section 4, we introduce the *EP* bounds and prove that they yield a hierarchy of bounds. In Section 5, we present the random partition sampling approach and enhancements to it. In Section 6, we provide the results of computational experiments on two-stage and multistage benchmark problems. We give conclusions and discuss promising extensions to this work in Section 7.

2. Literature Review

Challenges in solving multistage stochastic programs (MSPs) with integer variables are well documented; see, for example, Ahmed (2010), Klein Haneveld and van der Vlerk (1999), Schultz (2003), and Sen and Zhou (2014). (A comprehensive list of relevant scholarly works published between 1996 and 2007 is provided by van der Vlerk 2007.) These challenges have spurred many recent algorithmic developments, with substantial effort focused on decomposition approaches.

One line of work employs stage-wise decomposition and convexification of the value function at each scenario tree node; see, for example, the work of Klein Haneveld et al. (2006), Sen and Higle (2005), Sen and Sherali (2006), and van der Vlerk (2010), and for a computational comparison of some alternative approaches, see Ntaimo and Sen (2008).

Another line of research, based on scenario-wise decomposition (Carøe and Schultz 1999), has been vigorously pursued in recent years, not least for its strong potential for parallel implementation. For example, Ahmed (2013) provides a scenario decomposition approach for 0–1 two-stage problems. The computational efficacy of a synchronous parallel implementation of this approach is demonstrated by Ahmed (2013), with algorithmic improvements, and an asynchronous parallel implementation is provided by Ryan et al. (2016). For two-stage stochastic mixed integer programming, Kim and Zavala (2018) develop software based on parallel implementation of a scenario decomposition method that uses Lagrangian relaxation to improve the dual bounds. Also solving a (stabilized) Lagrangian dual master problem, but exploiting its special structure to do so in parallel, is the method of Lubin et al. (2013). A related approach is progressive hedging (Rockafellar and Wets 1991), which has been used as an effective primal heuristic for SMIPs (Løkketangen and Woodruff 1996, Crainic et al. 2011, Watson and Woodruff 2011, Cheung et al. 2015, Veliz et al. 2015), including parallel implementations. Its connections with dual decomposition were recently exploited to derive hybrid approaches (Guo et al. 2015).

For an interesting study that compares stage-wise and scenario-wise decomposition for a class of problems with special structure, see Beier et al. (2015).

Scenario-wise decomposition can be generalized to decomposition into sets, or groups, of scenarios, with the subproblem for each group retaining all nonanticipativity constraints (NACs) between scenarios in the set, but the nonanticipativity constraints between scenarios in different groups relaxed. This idea was exploited by Escudero et al. (2010, 2012) and Aldasoro et al. (2013) in the context of branch-and-fix coordination algorithms, and by Escudero et al. (2013, 2016) with nonanticipativity constraints between groups (called "clusters" in this work) relaxed in a Lagrangian fashion. The groups form a partition of the set of all scenarios, the *scenario set*, that is induced by the subtrees corresponding to the nodes in one stage of the scenario tree. A hierarchy of bounds is observed by Escudero et al. (2016): for any Lagrangian multiplier vector, (and hence for the Lagrangian dual value), the Lagrangian relaxation dual bound is nonincreasing in the stage of the scenario tree used to induce the partition (the earlier the stage, the better the bound).

The work of Sandıkçı et al. (2013) developing *EGSO*(q) bounds describes approaches for computing dual bounds for two-stage SMIPs via the solution of many scenario group subproblems. Papers by Sandıkçı and Özaltin (2017) and Maggioni et al. (2016) and the working paper by Zenarosa et al. (2014b) describe extensions of these ideas. Sandıkçı and Özaltin (2017, p. 1774) study bounds from collections of group subproblems (without reference scenario) for groups that cover the scenario set. Such a collection is called a "blockset." They prove that an appropriately weighted sum of the group subproblem values over all groups in a blockset gives a lower bound. They also show that if all groups in a blockset contain no more than b scenarios, each scenario appears in exactly m groups, and the blockset that gives the best possible lower bound with these requirements is used, then the bound from the $m = 1$ case is better than the others. This suggests that restricting attention to blocksets that form a partition of the set of all scenarios, rather than a cover, is beneficial. When the set of all scenarios is of size L, Sandıkçı and Özaltin (2017) consider partitions in which all groups have cardinality q, in the case that q divides L, and in which all groups but one have cardinality q, and one group has cardinality $L \bmod q$ otherwise. They provide computational results showing the distribution of the resulting dual bound over all partitions of a set of 16 scenarios, for each of $q = 1, 2, 4, 8$, and 16, showing how the dual bound improves with group problem size and computation time increases. The distribution of primal bounds, derived from solutions to the group subproblems, is also given. The suggestion to stop MIP solution of each group subproblem prior to proven optimality is explored and shown to have the potential to greatly decrease run times with relatively less impact on the quality of the bounds. Finally, a parallel implementation for a given partition is shown to be able to compute primal solutions and gaps for instances with an enormous number of scenarios in reasonable wall clock time.

Zenarosa et al. (2014b) generalize the Sandıkçı et al. (2013) *EGSO* bound to include multiple reference scenarios in each group subproblem, and the resulting dual bound is shown to be monotonically increasing in the subset size. Like Sandıkçı and Özaltin (2017), Zenarosa et al. (2014b, p. 11) consider collections of group subproblems, however, their collections are constructed from scenario-node cuts in the scenario tree. A given scenario-node cut in the scenario tree consists of a set of scenario tree nodes that induce a partition of the scenarios, with a subset of the partition for each node in the cut. For each node in the cut, a group subproblem is constructed, with a group of scenarios and a set of reference scenarios, both of which are subsets of the scenarios corresponding to that node. The group problem values are combined to compute what is called the value of the "cut-group subproblem." For a fixed cut and a fixed set of reference scenarios for each node in the cut, taking the expected value of the cut-group subproblem over all possible group subproblems yields a dual bound. This bound is shown to increase monotonically in the group size. By using solutions of cut-group subproblems, Zenarosa et al. (2014b) also derive primal bounds and prove monotonicity properties of their expected values. Computational results, including with a parallel implementation, show the utility of these ideas.

Maggioni et al. (2016) show how to generate dual bounds based on a set of reference scenarios and taking the expected value of the group subproblems over all groups of a fixed cardinality. They show that the resulting bound increases monotonically in both the cardinality of the groups and as the set of reference scenarios expands. Maggioni et al. (2016) also suggest ideas for upper bounds, and provide inequalities on their expected values, in a similar vein to some of those given in Zenarosa et al. (2014b). Maggioni et al. (2016) provide numerical tests based on a real logistics SMIP application.

Finally, we mention that there is another stream of research focusing on solving stochastic programs by using partitions of the scenario set (Birge 1985, Espinoza and Moreno 2014, Song and Luedtke 2015). Though very effective in practice, these methods are quite different from the approach we propose in this paper. For example, *partition-based approaches* are defined by Song and Luedtke (2015) as aggregation-based methods where scenario-specific constraints are aggregated according to partitions of the scenario set to obtain dual bounds, and the partitions are iteratively refined later.

3. Preliminaries and *EGSO* Bounds for Multistage Problems

We consider the multistage stochastic programming problem, with T stages, and with random data $\tilde{\xi}$ having finite support. In particular, the random data are defined on a probability space with discrete realization set $\Xi \subseteq \Xi^1 \times \cdots \times \Xi^T$, arranged according to a scenario tree. Each realization $\xi \in \Xi$ corresponds to a path in the scenario tree from its root node to a leaf node, and every such path uses T nodes. We use the notation $\xi_{[t]}$ for $(\xi_{t'})_{t'=1}^{t} = (\xi_1, \ldots, \xi_t)$, the realization, ξ, for stages $1, \ldots, t$. Because all scenarios share a single tree node in the first stage, it is assumed that $\tilde{\xi}_1$ is deterministic. We take the multistage stochastic program to have the following form:

$$z_{MSP} := \min\{f_1(x_1) + \mathbb{E}[g_2(x_1, \xi_{[2]})] \ : \ x_1 \in \mathscr{X}^1\},$$

where for each $t = 2, \ldots, T$, $g_t(x_{t-1}, \xi_{[t]})$ is defined as

$$g_t(x_{t-1}, \xi_{[t]}) = \min\big\{f_t(x_t, \xi_{[t]}) + \mathbb{E}[g_{t+1}(x_t, \xi_{[t+1]}) | \xi_{[t]}] \\ : x_t \in \mathscr{X}^t(x_{t-1}, \xi_{[t]})\big\},$$

where $g_{T+1} \equiv 0$, and stage t decision variables, x_t, are assumed to be of dimension n_t, so $x_t \in \mathbb{R}^{n_t}$ for each $t = 1, \ldots, T$. We allow any finite-valued functions f_t and any set-valued functions $\mathscr{X}^t$, provided (MSP) has an optimal solution, and provided the restriction to any proper subset of Ξ has an optimal solution. For practical implementation, we also require a solver that can handle problems in the form of (MSP).

Because $\tilde{\xi}$ has finite support, we may write $\Xi = \{\xi^1, \ldots, \xi^L\}$ for some positive integer L, and index the scenario set Ξ with $\mathcal{S} = \{1, \ldots, L\}$. Define $\mathscr{H}(t, s)$ to be the scenario tree node for the scenario with index s at stage t, for each $t = 1, \ldots, T$ and $s \in \mathcal{S}$. This permits us to write (MSP) in its *extensive form* as

$$\begin{aligned} z_{MSP} = \min \quad & \sum_{s \in \mathcal{S}} p_s \sum_{t=1}^{T} F_t^s(x_t^s) \\ \text{s.t.} \quad & (x_t^s)_{t=1}^T \in X^s, && \forall s \in \mathcal{S}, \\ & y_{\mathscr{H}(t,s)} = x_t^s, && \forall t = 1, \ldots, T, \ s \in \mathcal{S}, \end{aligned}$$

where p_s is the probability that ξ^s is realized, and it is assumed that $\sum_{s \in \mathcal{S}} p_s = 1$. We define $F_t^s(x_t^s) = f_t(x_t^s, \xi_{[t]}^s)$ and

$$X^s = \{(x_t)_{t=1}^T \ : \ x_1 \in \mathscr{X}^1, \ x_t \in \mathscr{X}^t(x_{t-1}, \xi_{[t]}^s)), \\ \forall t = 2, \ldots, T\}.$$

The term $y_h \in \mathbb{R}^{n_t}$ for each scenario tree node h at stage t is an auxiliary variable introduced to model the NACs, ensuring that decisions made at stage t do not depend on knowledge of realizations at later stages.

We now define the *group subproblem* for a subset $S \subseteq \mathscr{S}$ of the scenarios, denoted by $GR(S)$, to be

$$v_{GR}(S) = \frac{1}{\rho(S)} z_{GR}(S),$$

where

$$\begin{aligned} z_{GR}(S) = \quad \min \quad & \sum_{s\in S} p_s \sum_{t=1}^{T} F_t^s(x_t^s) \\ \text{s.t.} \quad & (x_t^s)_{t=1}^T \in X^s, \qquad \forall s \in S, \\ & y_{\mathcal{H}(t,s)} = x_t^s, \quad \forall t = 1,\dots,T,\ s \in S, \end{aligned}$$

and we define

$$\rho(S) = \sum_{s\in S} p_s.$$

Note that, as in Sandıkçı and Özaltin (2017), we use a simpler form of the group subproblem than that given by Sandıkçı et al. (2013), without a reference scenario. Note also that our assumption that (MSP) restricted to any proper subset of Ξ has an optimal solution ensures that all group subproblems have an optimal solution.

Observe that $v_{GR}(\mathscr{S}) = z_{GR}(\mathscr{S}) = z_{MSP}$, and that for each $s \in \mathscr{S}$, the value of the group subproblem for the set consisting of the single scenario, s, is simply

$$v_{GR}(\{s\}) = \min\left\{\sum_{t=1}^{T} F_t^s(x_t^s) \ :\ (x_t^s)_{t=1}^T \in X^s\right\};$$

we call this the *single-scenario subproblem*. The so-called wait-and-see solution has value

$$z_{WS} = \sum_{s\in\mathscr{S}} p_s v_{GR}(\{s\}) = \sum_{s\in\mathscr{S}} z_{GR}(\{s\}).$$

It is well known that this provides a dual (i.e., lower) bound on z_{MSP}, because it is precisely equivalent to solving (MSP) with all NACs omitted. Thus, $z_{WS} \le z_{MSP}$.

We now replicate the key result of Sandıkçı et al. (2013), adjusted and extended to our multistage context (see also Maggioni et al. 2014). The *expected group subproblem objective* for an integer q with $1 \le q \le L = |\mathscr{S}|$ is denoted by $EGSO(q)$ and defined by

$$EGSO(q) = \frac{1}{\binom{L-1}{q-1}} \sum_{S\in\Omega_q} \rho(S) v_{GR}(S) = \frac{1}{\binom{L-1}{q-1}} \sum_{S\in\Omega_q} z_{GR}(S),$$

where Ω_q denotes the set of all subsets of $\mathscr{S}$ of cardinality q. The result is as follows.

Theorem 1 (Sandıkçı et al. 2013, theorem 1).

$$z_{WS} = EGSO(1) \le EGSO(2) \le \cdots \le EGSO(L-1) \le EGSO(L) = z_{MSP}.$$

The proof of Theorem 1, as well as some necessary preliminary results, are given in Section 1 of the online supplement. The slight difference from the result in Sandıkçı et al. (2013), where $z_{WS} \le EGSO(1)$ rather than $z_{WS} = EGSO(1)$, is caused by the omission of the reference scenario.

4. Expected Partition Bounds: A Hierarchy of Dual Bounds

We use the notation Π to denote a partition of the scenario set, $\mathscr{S}$, so $\Pi = \{S_1,\dots,S_m\}$, for some m, with $S_i \subseteq \mathscr{S}$ for all $i = 1,\dots,m$ and $S_i \cap S_{i'} = \emptyset$ for $i' = 1,\dots,m$, with $i' \ne i$. It is well known that solving the group subproblem over all subsets in a partition yields a lower bound on (MSP).

Proposition 1. *Let Π be a partition of $\mathscr{S}$, with $\Pi = \{S_1,\dots,S_m\}$, for some positive integer m. Then, defining $z^P(\Pi)$ to be the total value of the problems over all subsets of the partition Π, we have*

$$z^P(\Pi) = \sum_{i=1}^{m} z_{GR}(S_i) \le z_{MSP}.$$

Thus a dual bound on z_{MSP} is obtained from a single partition. In practice, provided a solver is available for problems of the form of (MSP) and is computationally effective for problems with a modest number of scenarios, it can be applied in parallel to yield a single-partition bound for any partition in which all subsets are of modest size.

To obtain a hierarchy of bounds from partitions, we propose using the following particular type of partition, which, for a given integer q, partitions the scenario set into sets of size q or "almost" q, specifically, into sets of size q or $q-1$.

Definition 1. Given a positive integer q, the *q-partition set* of $\mathscr{S}$, denoted by Λ_q, is the set of all partitions of $\mathscr{S}$ into m subsets of size q and m' subsets of size $q-1$, where $m' = q\lceil \frac{L}{q} \rceil - L$, $m = \lceil \frac{L}{q} \rceil - m'$, and $L = |\mathscr{S}|$.

It is straightforward to verify that the q-partition set justifies its name, that is, that

$$mq + m'(q-1) = L.$$

Note that if q is "too large" relative to $L = |\mathscr{S}|$, then it is possible that m could be negative, and hence Λ_q could be ill defined. Note also that, to be of practical interest, partitions should consist of sets having sizes small relative to the entire set of scenarios, otherwise the benefit of being able to solve many smaller problems in parallel to construct a lower bound from the partition is lost. Fortunately, as the following observation indicates, q can be quite large relative to L without m dropping to zero. For example, $q \le \sqrt{L}$ suffices.

Lemma 1. *The q-partition set of $\mathcal{S}$, Λ_q, is well defined, that is, $m \geq 1$, if and only if*

$$q|L \quad or \quad L \geq (q-r)q + r, \tag{1}$$

where $r = L - q\lfloor \frac{L}{q} \rfloor$.

The expected partition bound that we propose to consider is obtained by taking the average of all q-partition single-partition bounds.

Definition 2. The *expected partition bound* for subsets of (almost) size q is denoted by $EP(q)$ and given by

$$EP(q) = \frac{1}{|\Lambda_q|} \sum_{\Pi \in \Lambda_q} z^P(\Pi).$$

In what follows, we repeatedly use the observation that q-partitions of a set S can be enumerated by enumerating all permutations of its $\ell = |S|$ elements and then taking each consecutive sequence of q (and then $q-1$) elements to form one of the subsets in the partition. However, many permutations of $1, \ldots, \ell$ give the same partition, because each sequence of length q (or $q-1$) can be permuted without changing the partition, and the m sequences themselves can be permuted without changing the partition. In general, the number of distinct partitions of a set of size $\ell = mq$ into m sets of size q is

$$\frac{\ell!}{(q!)^m m!}.$$

By similar arguments, each set in Ω_q appears in

$$\frac{(\ell - q)!}{(q!)^{m-1}(m-1)!}$$

distinct partitions of S. Both formulae are easily extended to q-partitions of $\mathcal{S}$, as given in the proof of the proposition below.

Proposition 2. *Let q be a positive integer satisfying (1), and define m and m' as in Definition 1, where $L = |\mathcal{S}|$. Then*

$$EP(q) = \frac{qm}{L} EGSO(q) + \frac{(q-1)m'}{L} EGSO(q-1).$$

Proof. Let $\ell = mq$ and $\ell' = m'(q-1)$, so $\ell + \ell' = L$. The number of distinct partitions of a set of size L into m sets of size q and m' sets of size $q-1$ is given by

$$|\Lambda_q| = \binom{L}{\ell} \frac{\ell!}{(q!)^m m!} \frac{\ell'!}{((q-1)!)^{m'} m'!},$$

which can easily be simplified to

$$|\Lambda_q| = \frac{L!}{(q!)^m m!((q-1)!)^{m'} m'!}.$$

Now, for P, a given subset of $\mathcal{S}$ of size q, any q-partition containing P must induce a partition of $\mathcal{S} \setminus P$ into two sets: S^1, the set of scenarios contained in subsets of size q of the partition (but not in P), and S^2, the set of scenarios contained in subsets of size $q-1$ of the partition. Obviously, $|S^1| = (m-1)q = \ell - q$ and $|S^2| = m'(q-1) = \ell'$. Thus, to construct all q-partitions that contain P, one may consider (i) all ways of choosing S^1 from $\mathcal{S} \setminus P$, (ii) all ways of partitioning S^1 into sets of size q, and (iii) all ways of partitioning S^2 into sets of size $q-1$, in combination. For case (i), there are $\binom{L-q}{\ell-q}$ such ways. For case (ii), there are $(\ell-q)!((q!)^{(m-1)}(m-1)!)$ ways. Similarly, for case (iii), there are $\ell'!/(((q-1)!)^{m'} m'!)$ ways. Putting these in combination, we see that each distinct subset of $\mathcal{S}$ of size q appears in

$$\eta_q := \binom{L-q}{\ell-q} \frac{(\ell-q)!}{(q!)^{(m-1)}(m-1)!} \frac{\ell'!}{((q-1)!)^{m'} m'!}$$

partitions. The above expression can easily be simplified to

$$\eta_q = \frac{(L-q)!}{(q!)^{(m-1)}(m-1)!((q-1)!)^{m'} m'!}.$$

Similarly, each distinct subset of $\mathcal{S}$ of size $q-1$ appears in

$$\eta'_q := \binom{L-q+1}{\ell} \frac{\ell!}{(q!)^m m!} \frac{(\ell'-q+1)!}{((q-1)!)^{(m'-1)}(m'-1)!}$$

partitions; the above expression can easily be simplified to

$$\eta'_q = \frac{(L-q+1)!}{(q!)^m m!((q-1)!)^{(m'-1)}(m'-1)!}.$$

Now

$$\begin{aligned}\frac{1}{|\Lambda_q|} \sum_{\Pi \in \Lambda_q} z^P(\Pi) &= \frac{1}{|\Lambda_q|} \sum_{\Pi \in \Lambda_q} \sum_{S \in \Pi} z_{GR}(S) \\ &= \frac{1}{|\Lambda_q|} \left(\eta_q \sum_{S \in \Omega_q} z_{GR}(S) + \eta'_q \sum_{S \in \Omega_{q-1}} z_{GR}(S) \right).\end{aligned} \tag{2}$$

But

$$\begin{aligned}\frac{\eta_q}{|\Lambda_q|} &= \frac{(L-q)!}{(q!)^{(m-1)}(m-1)!((q-1)!)^{m'} m'!} \\ &\quad \cdot \frac{(q!)^m m!((q-1)!)^{m'} m'!}{L!} \\ &= \frac{(L-q)!q!m}{L!} = \frac{qm}{L} \frac{(L-q)!(q-1)!}{(L-1)!} \\ &= \frac{qm}{L} \frac{1}{\binom{L-1}{q-1}}.\end{aligned}$$

Thus,

$$\frac{\eta_q}{|\Lambda_q|}\sum_{S\in\Omega_q} z_{GR}(S) = \frac{qm}{L}\frac{1}{\binom{L-1}{q-1}}\sum_{S\in\Omega_q} z_{GR}(S) = \frac{qm}{L}EGSO(q). \tag{3}$$

Similarly,

$$\begin{aligned}\frac{\eta'_q}{|\Lambda_q|} &= \frac{(L-q+1)!}{(q!)^m m!((q-1)!)^{(m'-1)}(m'-1)!}\\ &\quad\cdot\frac{(q!)^m m!((q-1)!)^{m'}m'!}{L!}\\ &= \frac{(L-q+1)!(q-1)!m'}{L!}\\ &= \frac{(q-1)m'}{L}\frac{(L-q+1)!(q-2)!}{(L-1)!}\\ &= \frac{(q-1)m'}{L}\frac{1}{\binom{L-1}{q-2}}\end{aligned}.$$

Thus,

$$\begin{aligned}\frac{\eta'_q}{|\Lambda_q|}\sum_{S\in\Omega_{q-1}} z_{GR}(S) &= \frac{(q-1)m'}{L}\frac{1}{\binom{L-1}{q-2}}\sum_{S\in\Omega_{q-1}} z_{GR}(S)\\ &= \frac{(q-1)m'}{L}EGSO(q-1).\end{aligned} \tag{4}$$

The result follows by substituting from (3) and (4) into (2). □

We now obtain our main result as a corollary.

Theorem 2. *If q divides evenly into $L = |\mathcal{S}|$, then*

$$EP(q) = EGSO(q),$$

and otherwise, provided that q satisfies (1),

$$EGSO(q-1) \le EP(q) \le EGSO(q),$$

with the first inequality strict unless $EGSO(q-1) = EGSO(q)$.

Proof. Let m and m' be as defined in Definition 1. Now if q divides evenly into L, then $m' = 0$, so $qm = L$, and the result follows by Proposition 2. Otherwise, suppose that q satisfies (1). Then, by Lemma 1, $m \ge 1$, obviously $\frac{qm}{L} > 0$ and $\frac{(q-1)m'}{L} \ge 0$, and, furthermore, $\frac{qm}{L} + \frac{(q-1)m'}{L} = 1$. Thus, by Proposition 2, it must be that $EP(q)$ is a convex combination of $EGSO(q-1)$ and $EGSO(q)$, with a strictly positive coefficient on the latter in the combination. Because $EGSO(q-1) \le EGSO(q)$, the result follows. □

This result shows that the $EP(q)$ bounds provide a sequence of dual bounds for z_{MSP} that are monotonically nondecreasing in q, and that coincide with the $EGSO$ bounds in the case where the subset cardinality divides evenly into the cardinality of the scenario set, and otherwise interpolates between the bounds for two consecutive cardinalities.

5. Scenario Set Partition Sampling

The practical value of the results in the previous section comes from the observation that there is a high likelihood that computing only a very small number of q-partition single-partition bounds will yield a better bound than $EP(q)$, provided the distribution of $z^P(\Pi)$ over $\Pi \in \Lambda_q$ is not highly right skewed. Specifically, given $\tilde{\Lambda}_q \subset \Lambda_q$, an independently sampled subset of the q-partitions, we have from Proposition 1 that

$$\max_{\Pi\in\tilde{\Lambda}_q} z^P(\Pi) \le z_{MSP}.$$

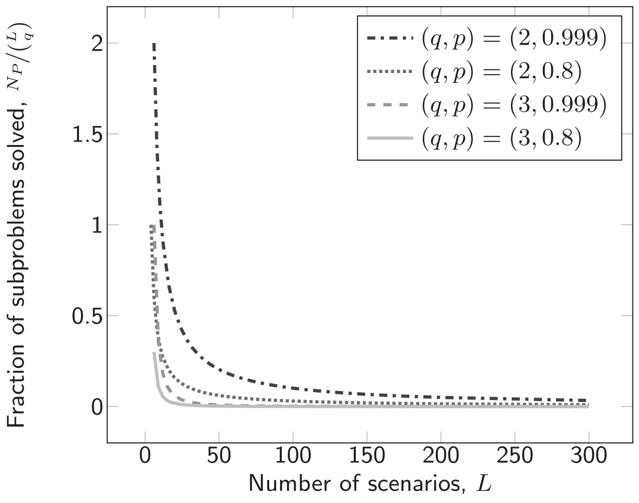
If, for example, the distribution of the values of $z^P(\Pi)$ over $\Pi \in \Lambda_q$ is symmetric about its average, then the probability that a single, randomly chosen $\Pi \in \Lambda_q$ will have value no greater than its average is 0.5. Because its average is exactly $EP(q)$, we have that

$$\begin{aligned}\Pr(\max_{\Pi\in\tilde{\Lambda}_q} z^P(\Pi) \ge EP(q)) &= 1 - \Pr(z^P(\Pi) < EP(q),\\ &\quad \forall\Pi\in\tilde{\Lambda}_q) \approx 1 - (0.5)^{|\tilde{\Lambda}_q|}.\end{aligned}$$

To illustrate the utility of this, consider a problem with $L = 24$ scenarios, and take $q = 3$. To compute $EP(3) = EGSO(3)$, we must compute the solution to $\binom{24}{3} = 2{,}024$ group subproblems, each with three scenarios. However, if we randomly sample 10 partitions, we need to solve only $\frac{L}{q} \times |\tilde{\Lambda}_q| = \frac{24}{3} \times 10 = 80$ such group subproblems, and will have found a bound at least as good with probability $1 - (0.5)^{10} > 0.999$. This idea is demonstrated in Figure 1 for a symmetric distribution of partition bounds. As the number of scenarios increases, the fraction of subproblems that need to be solved to find a better partition bound than the expected bound (EP) with a given probability gets smaller. Even in the case that the distribution of the values of $z^P(\Pi)$ over $\Pi \in \Lambda_q$ is somewhat right skewed, for example, say, $\Pr(z^P(\Pi) \ge EP(q)) = 0.2$ only, then an independent random sample of 31 partitions, requiring solution of only 248 group subproblems, is sufficient to ensure that the best bound generated by one of them is at least $EP(q)$ with probability at least 0.999. On benchmark instances, our observation echoes that made by Sandıkçı and Özaltin (2017): these distributions, in the case of q-partitions, are not highly skewed (see Section 2 of the online supplement).

This motivates Algorithm 1, which computes exact lower bounds from randomly sampled q-partitions. The stopping criterion enforces a maximum on the number of partitions sampled, K^{max}, which is a given parameter of the algorithm. Although the primary purpose of Algorithm 1 is to generate a good dual bound, there may also be an opportunity to generate

Figure 1. Best Partition vs. Expected Partition



Note. The term q represents the group size, and p is target probability of finding a better partition bound than $EP(q)$ after solving N_P subproblems; $N_P = \frac{L}{q} \times \lceil \log_2 \frac{1}{1-p} \rceil$.

upper bounds during the course of the algorithm, making use of the solution to each group subproblem. For example, Sandıkçı and Özaltin (2017) describe such a heuristic. Even though randomly sampled partitions provide effective results, as we discuss in Section 6, we explore several ways to select partitions more intelligently, to yield better partition bounds. These are described in the remainder of this section.

Algorithm 1 Scenario Set Partition Sampling (S, L, q, K^{max})

```
Inputs: Set of scenarios $S$, $L = |S|$, group size $q$, number of partitions $K^{max}$
$LB := -\infty$, $k := 1$
while $k \le K^{max}$ do
  $S := \mathcal{S}$
  for all $i = 1, \dots, \lceil \frac{L}{q} \rceil$ do
    /* Sample, without replacement, the next subset
       in the partition */
    if $i \le L - (q-1)\lceil \frac{L}{q} \rceil$ then
      $S_{ki}$ := a random sample of $q$ scenarios from $S$
    else
      $S_{ki}$ := a random sample of $q-1$ scenarios from $S$
    end if
    /* Solve the multistage SP over scenarios in the
       current subset */
    Solve $GR(S_{ki})$ to get optimal value $z_{GR}(S_{ki})$
    $S := S \setminus S_{ki}$
  end for
  $z_k := \sum_{i=1}^{\lceil L/q \rceil} z_{GR}(S_{ki})$
  /* Update exact lower bound */
  $LB := \max\{LB, z_k\}$
  [Optional]: seek an optimally recombined partition (Section 5.2) to improve $LB$
  $k := k + 1$
end while
```

5.1. Partition Sampling Based on Scenario Tree Structure

When using partition sampling on multistage problem instances, taking advantage of the scenario tree structure can potentially result in improved partition bounds. Selecting partitions that group together scenarios with as many common nodes in the scenario tree as possible can provide better partition bounds compared with random partition sampling, because of fewer NACs being violated. On the other hand, partitions that keep "similar" scenarios (scenarios with common nodes in the scenario tree) in different groups also have the potential to provide better bounds than randomly sampled partitions, because of each group subproblem being a more accurate representation of the original problem. Here, we explore both strategies.

Algorithm 2 defines our method of generating q-partitions that groups scenarios with common nodes together, called the "tree-aligned" partitioning strategy. The algorithm takes the scenario tree, $\mathcal{H}$, as input and first generates $\mathcal{N}_j$, given in (5), a priori for every scenario tree node, j. Here, $\mathcal{N}_j$ represents the set of scenarios that have node j on their scenario path, that is, the set of leaves of a subtree rooted at node j, so $\mathcal{N}_j$ is given by

$$\mathcal{N}_j = \{s \in S : j \in \mathcal{P}_{\mathcal{H}}(s)\}, \tag{5}$$

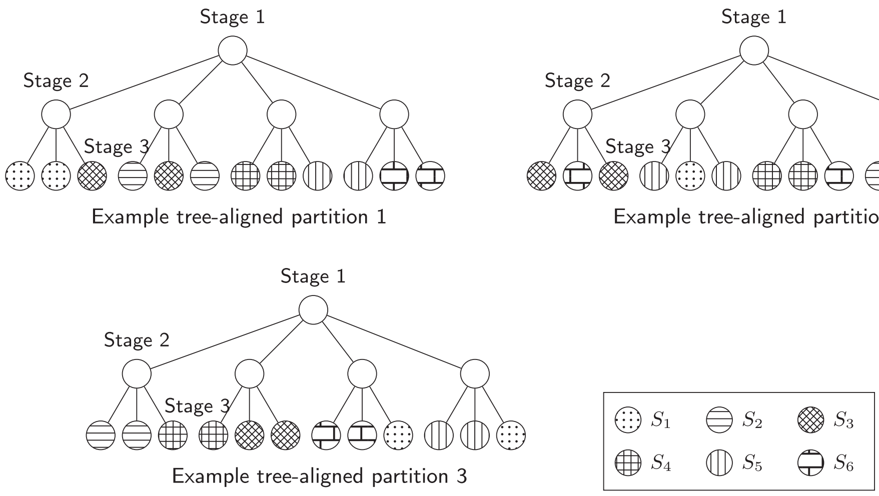
where $\mathcal{P}_{\mathcal{H}}(s)$ denotes the path in scenario tree $\mathcal{H}$ from the root node to leaf node s.

Algorithm 2 starts with any scenario tee node j^* in the penultimate stage, $T-1$, and randomly selects scenarios, $k \in \mathcal{N}_{j^*}$, to add to the first partition subset S_1. To keep track of which scenarios have been assigned a partition subset, the selected scenario k is removed from $\mathcal{N}_{j^*}$, and is removed from $\mathcal{N}_j$ for every $j \in \mathcal{P}_{\mathcal{H}}(k)$.

This is repeated until either the current subset, S_i, is full or there are no more scenarios that include node j^* ($\mathcal{N}_{j^*}$ is empty). In the former case, a new subset is started (i is incremented), and the process continues. In the latter case, a new tree node, j^*, with nonempty $\mathcal{N}_{j^*}$, having the maximum possible number of tree nodes in common with the previous j^* is found, and again the process continues. Selection of the new j^* is done by recursively checking the parent nodes j' of the current j^*, that is, $j' = a(j^*)$, $j' = a(a(j'))$, and so forth, until a nonempty $\mathcal{N}_{j'}$ is found. Once such a node j' is identified, a random $(T-1)$st stage descendant j'' of j' with nonempty $\mathcal{N}_{j''}$ is selected. This j'' is set as the new j^*. To see the steps of Algorithm 2 on an example, the reader is referred to Section 4 of the online supplement.

This strategy ensures that the resulting partition solution satisfies as many NACs as possible. The concept of a tree-aligned partition is illustrated in Figure 2, which shows three tree-aligned partitions on a small,

Figure 2. Example Tree-Aligned Partitions for a Tree with $|\Xi^1 \times \Xi^2 \times \Xi^3| = 1 \times 4 \times 3$ and $q = 2$


Stage 1
Stage 2
Stage 3
Example tree-aligned partition 1
Stage 1
Stage 2
Stage 3
Example tree-aligned partition 2
Stage 1
Stage 2
Stage 3
Example tree-aligned partition 3
S_1
S_2
S_3
S_4
S_5
S_6

three-stage scenario tree, with $|\Xi^1 \times \Xi^2 \times \Xi^3| = 1 \times 4 \times 3$ and $q = 2$. Note that, provided $q \neq |\Xi^t \times \Xi^{t+1} \times \cdots \times \Xi^T|$ for some t, in which case there is a unique tree-aligned partition, the number of tree-aligned partitions is still large relative to the total number of partitions. For example, a $1 \times 2 \times 3$ tree with $q = 2$ has 9 distinct tree-aligned partitions out of a total 90 distinct partitions, and a $1 \times 3 \times 2$ tree with $q = 3$ has 6 distinct tree-aligned partitions out of a total 20.

Algorithm 2 Partitioning Strategy, Tree Aligned $(S, L, q, \mathcal{H})$

```
*Inputs:* Set of scenarios $S$, $L = |S|$, group size $q$, scenario tree $\mathcal{H}$
**for all** $j$ in nodes of scenario tree $\mathcal{H}$ **do**
    $\mathcal{N}_j := \{s \in S : j \in \mathcal{P}_{\mathcal{H}}(s)\}$
end for
$j^* :=$ any scenario tree node in the penultimate stage, $T - 1$
**for all** $i = 1, \ldots, \lceil \frac{L}{q} \rceil$ **do**
    $S_i := \emptyset$
    **While** $(i \leq L - (q-1)\lceil \frac{L}{q} \rceil$ and $|S_i| < q)$ or $(i > L - (q-1) \cdot \lceil \frac{L}{q} \rceil$ and $|S_i| < q - 1)$ **do**
        **If** $\mathcal{N}_{j^*} \neq \emptyset$ **then**
            $k :=$ randomly picked scenario from $\mathcal{N}_{j^*}$
            $S_i := S_i \cup \{k\}$
            **for all** $j \in \mathcal{P}_{\mathcal{H}}(k)$ **do**.
                $\mathcal{N}_j := \mathcal{N}_j \setminus \{k\}$
            **end for**
        **end if**
        **while** $\mathcal{N}_{j^*} = \emptyset$ **do**
            $j^* :=$ its parent node
        **end while**
        $j^* :=$ randomly picked $(T - 1)$st stage descendant node $j$ of $j^*$ with $\mathcal{N}_j \neq \emptyset$
    **end while**
**end for**
```

To evaluate this method, we test the tree-aligned ("tree," for short) partitioning strategy against two others: a "random" strategy, which randomly partitions the scenario set without considering the tree structure, and a "misaligned" strategy, which keeps the scenarios with common nodes in separate groups as much as possible. These three different partitioning strategies are illustrated on a three-stage example with nine scenarios in Figure 3.

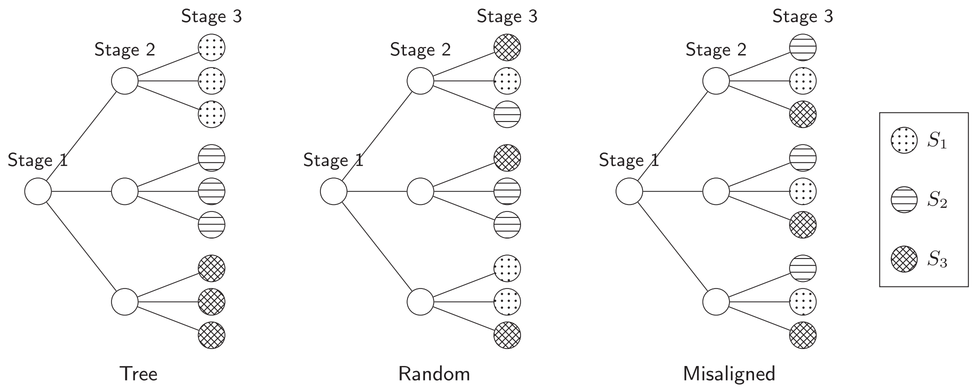
An extensive computational study on the quality of partition bounds attained by each of the three partitioning strategies is conducted, and the results are discussed in Section 6. Computational results reveal that the tree-aligned partitioning strategy provides considerable improvements in bound quality over the other two strategies on the three-stage instance class used in our computational study. However, the performance of different partitioning strategies depends on specific characteristics of the problem instance under consideration; therefore, it is not always obvious which strategy will result in better partition bounds. In Section 6, we present a discussion on the effects of some instance-specific characteristics that affect bound quality in tree-aligned and misaligned partitioning strategies.

5.2. Optimally Recombined Partitions

As the scenario set partition sampling procedure randomly samples partitions, it computes the group subproblem objectives, $z_{GR}(S)$, associated with scenario subsets, S, of these partitions. Once a sufficient number of partitions are sampled, it may be possible to recombine the previously sampled subsets into a new and potentially better partition. Thus, we seek optimally recombined partitions by solving a set partitioning problem over the previously sampled subsets, S, and the corresponding group subproblem objectives.

Recalling that $\mathcal{S}$ represents the set of scenario indices, let $\mathcal{C}$ represent the collection of previously sampled scenario subsets, and let $\mathcal{C}(i)$ denote the collection of

Figure 3. Three Alternative Methods to Construct Partitions Based on the Scenario Tree Structure



scenario subsets containing scenario i. An optimally recombined partition can be found by solving

$$\arg\max_{x\in\{0,1\}^{|\mathscr{C}|}}\left\{\sum_{S\in\mathscr{C}} z_{GR}(S)x_S : \sum_{S\in\mathscr{C}(i)} x_S = 1, \quad \forall i \in \mathscr{S}\right\}.$$

We incorporate optimal recombination into the partition sampling procedure by solving the set partitioning problem at certain iterations. If a recombined partition that is different from the previously sampled partitions provides a better bound than the current best, then we update the current best bound accordingly. Otherwise, we continue to sample partitions.

5.3. Truncated Partition Bound Calculation

During the progression of the partition sampling algorithm, only a small fraction of partitions actually improve the best partition bound. (This is to be expected, because if k partitions have been solved so far, the probability that the next partition yields a better value is $1/(k+1)$.) This suggests the possibility of reducing wasted computational effort by using the group subproblem values calculated partway through a partition to heuristically estimate the final partition bound, and to decide whether a partition is promising or not, before all the group subproblems of the partition are solved.

To determine how promising a partition is after solving only a subset of the group subproblems, we define a heuristic estimate for the partition bound $z^P(\Pi)$. When group subproblems associated with only the first r scenario subsets $\{S_1,\ldots,S_r\}$ of partition Π are solved, the heuristic estimate $\hat{z}_r^P(\Pi)$ is defined as

$$\hat{z}_r^P(\Pi) = \frac{1}{\sum_{i=1}^r \rho(S_i)}\sum_{i=1}^r z_{GR}(S_i).$$

We consider the following rule for eliminating unpromising partitions, parameterized by the triple (α,β,γ) with $\alpha,\beta\in[0,1]$ and $\gamma>0$: *Eliminate partition Π if, after at least α fraction of the group subproblems are solved, the heuristic estimate of $z^P(\Pi)$ remains lower than γ times the best bound for the next r' group subproblems solved, where r' is at least β fraction of the remaining groups.* In other words, when m and $z^P(\Pi^*)$ denote the number of groups in partition Π and the best partition bound found so far, respectively, under the truncation rule parametrized by (α,β,γ), we characterize a partition Π as unpromising if

$$\hat{z}_i^P(\Pi) \le \gamma z^P(\Pi^*), \qquad \forall i = r,\ldots,r+r',$$

provided $r \ge \lceil \alpha m\rceil$ and $r' = \lceil \beta(m-r)\rceil - 1$. Note that eliminating a partition, that is, ceasing computation of the value of the group subproblems in the partition, will save the computation time of at least $(1-\beta)(m-r)$ group subproblems. However, as the estimate is only a heuristic, it may also eliminate a partition giving a new best bound. The higher the values of α and β, and the lower the value of γ, the more conservative the truncation rule. A more conservative rule will save less computing time, at less risk of eliminating a partition yielding a better bound.

In our computational experiments, we explore the extent of computational savings that can be obtained from truncated bound calculation. Furthermore, we present examples in which the computational effort saved by eliminating unpromising partitions is used toward promising partitions, resulting in an improved partition bound.

6. Computational Results

Computational experiments are performed for four different classes of problems: the stochastic capacitated facility location problem (CAP), the stochastic server location problem (SSLP), the dynamic capacity acquisition and allocation problem (DCAP), and the stochastic capacity expansion problem (PLTEXP). CAP and SSLP instances are two-stage instances, DCAP instances are three-stage instances, and PLTEXP instances are three-, four-, and five-stage instances. Computations were run on a heterogeneous cluster of machines with Xeon E5520, Xeon E5-2670, Xeon E5-2603, Xeon

E5-2650v3, and Xeon E5430 processors using the Gurobi 5.6.3 Python interface (with Python 2.7). Different values for the group size, q, are considered, while ensuring that q divides the number of scenarios, L, evenly, for simplicity.

6.1. Test Instances

The stochastic capacitated facility location problem considered here was described by Louveaux (1986). The first-stage decisions determine which facilities to open, and the second-stage decisions give the fraction of customer demand to be satisfied by each open facility. The instances we use come from Bodur et al. (2016). Each instance has 5,000 scenarios. Here, we use the first 240 of them and experiment with group sizes $q = 5, 10, 15, 20$.

The stochastic server location problem considered in this paper was described by Ntaimo and Sen (2005), and the instances used in experiments come from Ahmed et al. (2015). The first-stage decisions concern installation of servers at possible locations, and the second-stage decisions define the assignment of clients to servers. Group sizes $q = 2, 5, 10, 25$ are used in experiments on SSLP instances with 50 and 100 scenarios.

The dynamic capacity acquisition and allocation problem was described by Ahmed and Garcia (2003), and the three-stage instances used in computational experiments of this paper were acquired from Zenarosa et al. (2014a). The capacity acquisition decisions are made at stages 1 and 2, and based on the acquired capacities of the resources, allocation decisions are made at stages 2 and 3. The test instances are named "DCAP $n_in_j2\ 1{\times}n_2{\times}n_3$," where n_i and n_j represent the numbers of resources and tasks, respectively; n_2 represents the number of second-stage nodes in the scenario tree; and n_3 represents the number of third-stage nodes originating from every second-stage node. The instances used in this study have 200, 300, and 500 scenarios. Computational experiments are conducted using group sizes $q = 2, 5, 10, 20, 30, 40, 50$.

The multistage capacity expansion problem was described by Sims (1992) and Holmes (1994), and the multistage problem instances were acquired from Holmes (2018). Three-, four-, and five-stage problem instances, with six scenario tree nodes at each stage, were used in computational experiments. The model considers capacity expansion and allocation decisions in a flexible manufacturing system subject to uncertain demand, with the objective of maximizing profit. At each stage, regular and overtime capacity allocation are modeled, as well as the capacity acquisition/expansion decisions. The instances used in the computational study have 36, 216, and 1,296 scenarios, and group sizes of $q = 2, 4, 6, 12, 18$ are used in computational experiments.

Details of these test problems are given in Section 1 of the online supplement.

6.2. Computation Time

As briefly mentioned in Section 4, partition bounds provide a computationally efficient way to obtain dual bounds for problems that have too many scenarios to allow direct solution of the deterministic equivalent, or even to permit the model to be loaded without encountering memory issues. In this section, we investigate the computational burden of partition bounds by providing solution times associated with different numbers of scenarios (L) and group sizes (q) for a typical capacitated facility location problem instance (CAP 101) using a Xeon E5-2603 processor.

We experiment with $L = 240, 480$, and 960 scenarios and various group sizes. For each number of scenarios and group size, 10 partitions are randomly sampled, and the solution times are reported. Table 1A presents the total solution time (averaged over 10 partitions) to represent the case when all group subproblems are solved sequentially, and Table 1B presents the maximum solution time among all group subproblems (averaged over 10 partitions) to represent the case when the group subproblems are solved in parallel. To better understand the trade-off between solution time and bound quality, best partition bounds (among the 10 partitions) associated with each (L, q) pair are reported in Table 2, and the fractions of series/parallel computation time required to achieve the partition bounds in Table 2 are presented in Table 3.

It is clear that in each one of the $L = 240, 480$, and 960 cases, using moderate group sizes such as $q = 5$, 10, and 20, allows computing partition bounds in very reasonable times even when parallel implementation is not possible, as opposed to using very large group sizes or directly solving the original problem. It is also important to note that when trying to directly solve the original problem, memory restrictions must be taken into consideration, in addition to computation time issues. For example, directly solving the CAP 101 instance with 960 scenarios is not possible

Table 1. Solution Times (in Seconds) for CAP 101

	q					
L	5	10	20	40	80	$q = L$
			Panel A. Total solve time			
240	52.71	88.97	242.15	329.19	466.04	5,585.30
480	98.04	162.74	484.67	778.17	944.44	15,351.15
960	223.83	240.61	579.96	1,515.23	1,871.99	87,683.96
			Panel B. Maximum solve time			
240	2.08	10.07	49.47	121.75	168.80	5,585.30
480	2.70	9.99	58.79	161.05	234.98	15,351.15
960	5.01	8.70	38.09	162.28	184.90	87,683.96

Notes. The values in both panels are averages of 10 solutions. For $q < L$, 10 partitions are randomly sampled, and the average total (or maximum) solution time is reported. For $q = L$, the deterministic equivalent problem with L scenarios is solved 10 times, and the average solution time is reported.

Table 2. Best Partition Bounds (Among 10 Randomly Sampled Partitions) for CAP 101

	q					
L	5	10	20	40	80	$q = L$
240	732,005.22	733,311.23	733,967.62	734,158.09	734,302.79	734,354.25
480	729,194.92	730,530.73	731,161.12	731,434.18	731,604.76	731,631.30
960	729,929.05	731,283.14	731,888.98	732,184.13	732,345.42	732,490.10

for moderate memory sizes, as it uses over 50 GB of memory just to load the model.

Because larger group sizes provide better partition bounds, it is desirable to use the largest group size allowed by the processing power, memory, and parallel implementation capability. But even when the computing platform at hand has moderate processing power and memory, finding partition bounds with small group sizes is a computationally efficient way of obtaining good dual bounds, especially if parallel computing is available.

6.3. Comparing Partition Sampling with SAA

Partition bounds are compared against sample average approximation estimates found by solving the same size and number of group subproblems and then calculating confidence intervals around the resulting estimates. The lower end of the confidence interval is taken to be the SAA lower bound estimate. For each instance and each q value tested, we run Algorithm 1 with $K^{max} = 30$ or 50, requiring solution of $n' = K^{max}L/q$ group subproblems, each of size q. We then apply SAA to the instance, using similar computational effort: we take n' independent samples of size q, generated (with replacement) based on the probability distribution associated with scenario occurrences. For each scenario sample, S, the sample average value, denoted by $z_{SA}(S)$, is found by solving the following problem (see Kleywegt et al. 2002):

$$z_{SA}(S) = \min\left\{\frac{1}{q}\sum_{s\in S}\sum_{t=1}^{T} F_t^s(x_t^s)\ :\ (x_t^s)_{t=1}^T \in X^s,\ \forall s \in S,\right.$$
$$\left. y_{\mathcal{H}(t,s)} = x_t^s, \forall t = 1,\ldots,T,\ s \in S\right\}.$$

After n' subproblems are solved, for samples $S_1, S_2, \ldots, S_{n'}$, the corresponding SAA estimate is $\hat{z}_{n'}^{SAA} = \frac{1}{n'}\sum_{i=1}^{n'} z_{SA}(S_i)$. Then the sample standard deviation is $\hat{s}_{SAA}^2 = \frac{1}{n'-1}\sum_{i=1}^{n'}\big(z_{SA}(S_i) - \hat{z}_{n'}^{SAA}\big)^2$, and the confidence interval with a level of confidence α is $\hat{z}_{n'}^{SAA} \pm t_{\frac{\alpha}{2},n'-1}\sqrt{\frac{\hat{s}_{SAA}^2}{n'}}$.

Figure 4 illustrates, using a typical CAP instance, how partition bounds compare with SAA estimates and 95% confidence intervals around SAA estimates. Partition bounds associated with 30 independent partitions are presented, along with the SAA estimates calculated by solving the same size and number of group subproblems. For comparability of results in terms of computational burden, SAA estimates on the plots are updated only when a partition bound is fully computed. It can be observed that the best partition bound is significantly greater than the lower limit of the 95% confidence interval around the SAA estimate. Also, in most cases, the best partition bound exceeds the SAA estimate. Usefully, the best partition bound improves quite rapidly, so good bounds are available very early in the sampling process.

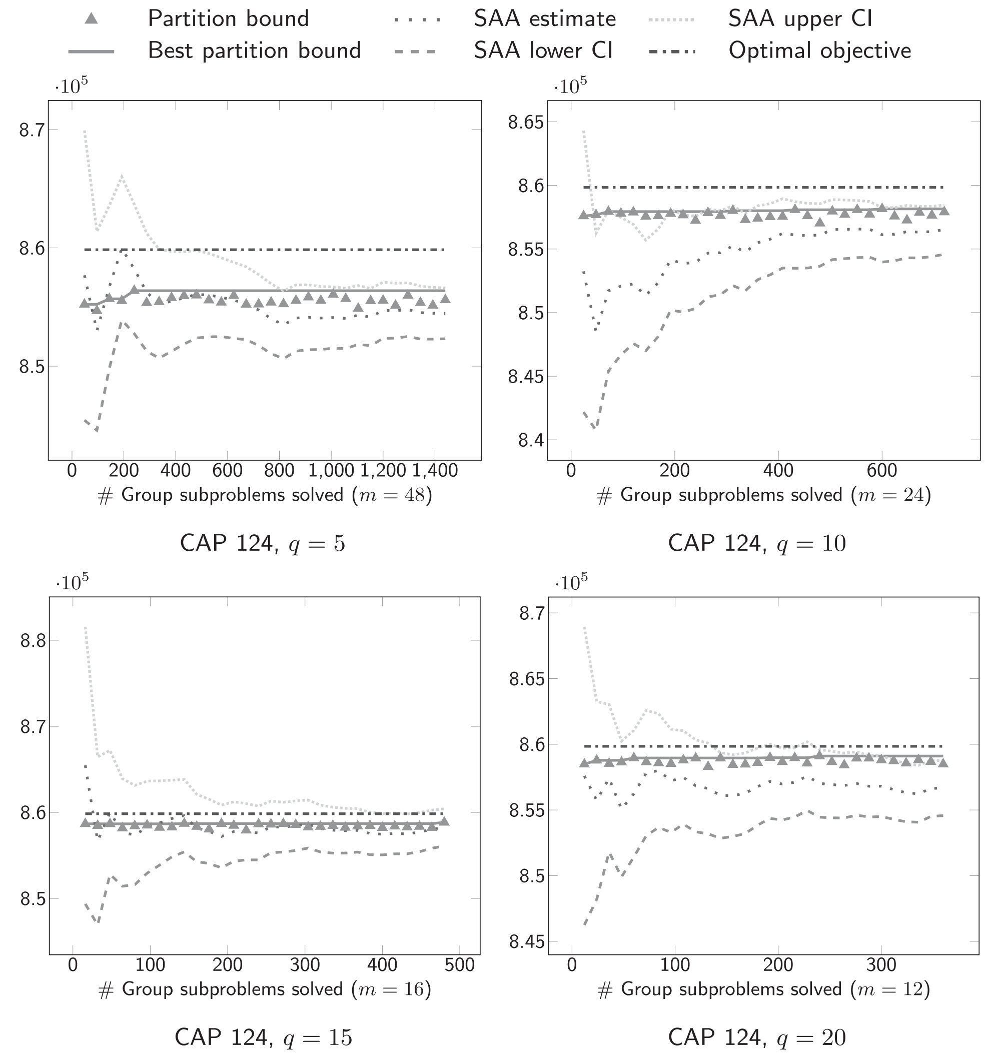
Figure 5 shows the progression of partition bounds and SAA estimates on a three-stage DCAP instance when the tree-aligned partitioning strategy is used (as described in Section 5.1) and an optimally recombined partition is attempted (as described in Section 5.2) every 10 iterations. As in the previous example, the best partition bound exceeds the lower limit of the 95% confidence interval around the SAA estimate. Additionally, good partition bounds are obtained very early in the partition sampling process,

Table 3. Expected Fraction of Series/Parallel Computation Time Needed to Achieve the Partition Bounds Shown in Table 2

	q				
L	5	10	20	40	80
240	0.094/0.0004	0.159/0.0018	0.434/0.0089	0.589/0.0218	0.834/0.0302
480	0.064/0.0002	0.106/0.0007	0.316/0.0038	0.507/0.0105	0.615/0.0153
960	0.026/0.0001	0.027/0.0001	0.066/0.0004	0.173/0.0019	0.213/0.0021

Notes. The computation times under consideration are for computing 10 partition bounds, the best ones among which are presented in Table 2. The reported fractions represent the computation times required for solving all group subproblems in series and in parallel, respectively, divided by the computation time required for solving the original problem with $q = L$.

Figure 4. CAP 124 Partition Sampling vs. SAA ($K^{max} = 30$)



Note. CI, Confidence interval.

whereas SAA estimates require a considerably long warm-up phase.

Similar behavior is observed in all CAP, SSLP, DCAP, and PLTEXP instances. Detailed results for all problem instances are presented in Section 2 of the online supplement. Tables 4 and 5 present summary statistics for CAP and SSLP instances, more specifically,

1. the gap obtained from the best partition bound after $k = 30$ q-partitions, Δ_k^P, given by Equation (6),
2. the gap obtained from the lower limit of the SAA 95% confidence interval after a computational effort equivalent to $k = 30$ q-partitions, Δ_k^{SAA}, given by Equation (7), and
3. the proportion of the latter gap that is closed by the best partition bound, $\delta_k^{SAA,P}$, given by Equation (8).

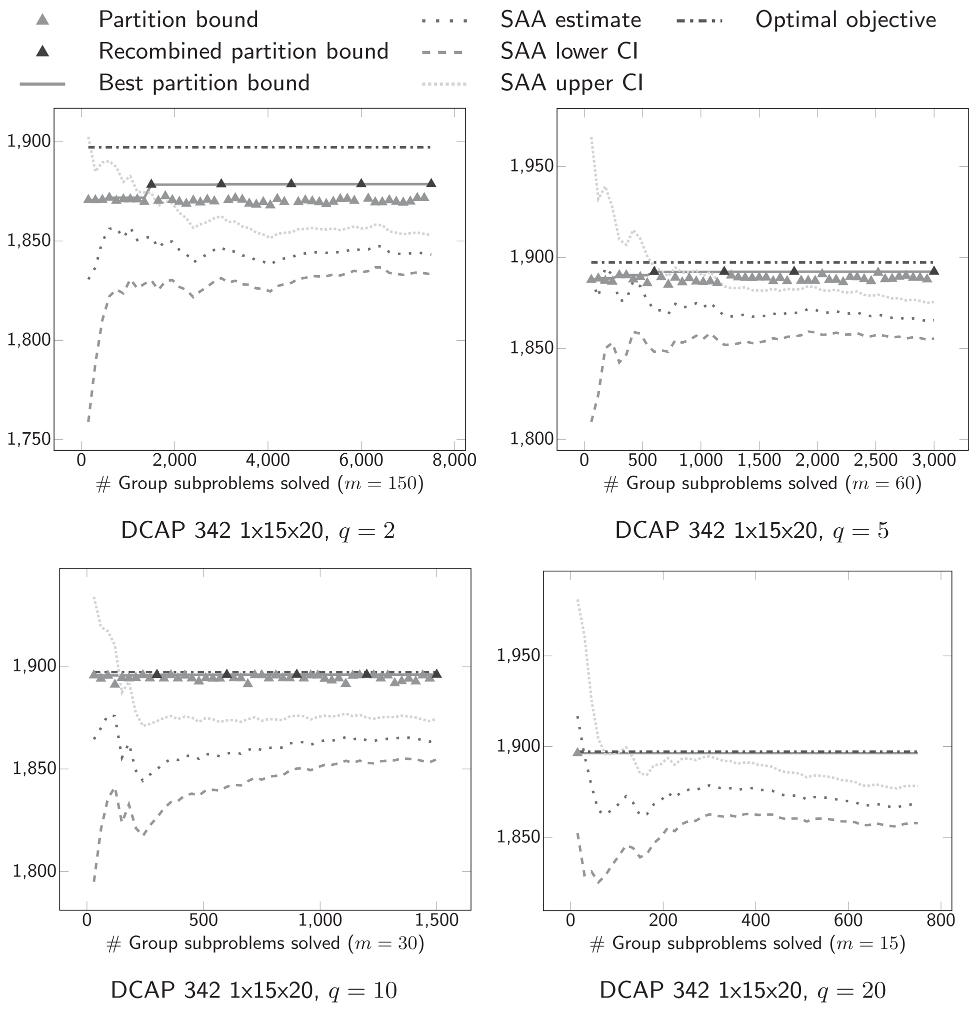
The gaps are calculated with respect to the wait-and-see solution, z_{WS}. Because the test instances have a considerable amount of sunk cost, which is the cost that has to be incurred regardless of solution quality, the objective values associated with different solutions do not seem very different from each other. To be able to objectively compare different solutions in terms of quality, we subtract the lower bound z_{WS} from the objective values in our reporting. Provided that OPT represents the optimal value over all L scenarios, the gaps are calculated as follows:

$$\Delta_k^P = \frac{(OPT - z_{WS}) - (\max_{i=1,\dots,k}\{z^{P_i}\} - z_{WS})}{OPT - z_{WS}}, \tag{6}$$

$$\Delta_k^{SAA} = \frac{(OPT - z_{WS}) - \left(\hat{z}_{km}^{SAA} - t_{0.025,km-1}\sqrt{\frac{\hat{s}^2_{SAA}}{km}} - z_{WS}\right)}{OPT - z_{WS}}, \quad \text{where } m = \frac{L}{q}, \tag{7}$$

$$\delta_k^{SAA,P} = \frac{\Delta_k^{SAA} - \Delta_k^P}{\Delta_k^{SAA}}. \tag{8}$$

Figure 5. DCAP 342 1×15×20 Partition Sampling (with Tree-Aligned Partitioning Strategy and Optimal Recombination) vs. SAA (K^{max} = 50)



Note. CI, Confidence interval.

Table 4 demonstrates the gaps Δ^{P}_{30}, Δ^{SAA}_{30}, and $\delta^{SAA,P}_{30}$ for the SSLP instances with different group sizes q, and Table 5 gives the means and standard deviations of Δ^{P}_{30}, Δ^{SAA}_{30}, and $\delta^{SAA,P}_{30}$, taken over all 16 CAP instances with different values of q. It can be seen in both tables that after 30 partitions, the gaps from the best partition bound are noticeably tighter than those from the SAA confidence interval lower limit for the same computational effort. Furthermore, it can be seen in Section 2 of the online supplement that in the majority of instances, the best partition bound is attained within 10 or 20 partitions.

6.4. Partition Bound Sampling Enhancements

For problem instances with more than two stages, the method of sampling partitions based on scenario tree structure is described in Section 5.1. We conduct a computational study for testing the bound quality for tree-aligned, random, and misaligned partitioning strategies in DCAP and PLTEXP instances. It is observed that in all problem instances belonging to these instance classes, when partitions are generated with the tree-aligned strategy, the resulting partition bounds are better than those for random and misaligned partitioning strategies. In DCAP instances, with 30 partitions, the tree-aligned strategy closes, on average, 85% of the wait-and-see gap with group size of only 5, and

Table 4. Best Partition Bound vs. Lower Limit of the 95% SAA Confidence Interval on SSLP Instances

Instance	L	q	Δ^{P}_{30} (%)	Δ^{SAA}_{30} (%)	$\delta^{SAA,P}_{30}$ (%)
SSLP 5-25-50	50	2	41.60	73.36	43.29
		5	11.62	19.97	41.83
		10	0.00	30.50	100.00
		25	0.00	21.19	100.00
SSLP 10-50-100	100	2	45.29	67.90	33.30
		5	9.01	25.51	64.67

Table 5. Best Partition Bound vs. Lower Limit of the 95% SAA Confidence Interval: Summary for CAP Instances

		Δ_{30}^{P}		Δ_{30}^{SAA}		$\delta_{30}^{SAA,P}$	
L	q	Mean (%)	Std. dev. (%)	Mean (%)	Std. dev. (%)	Mean (%)	Std. dev. (%)
240	5	14.87	4.53	32.35	8.34	52.23	14.08
240	10	6.45	3.37	25.71	8.44	71.07	16.96
240	15	3.53	2.34	19.17	8.80	80.23	14.27
240	20	2.40	1.87	17.14	10.16	83.54	12.88

Note. Std. dev., standard deviation.

96% with size 10. As reported in detail in Section 2 of the online supplement, the tree-aligned strategy obtains significantly better bounds than those found using other partitioning strategies, even with group sizes many times larger. The tree-aligned partitions close up to 3.5 times (1.9 times, on average, for DCAP instances) the wait-and-see gap that can be closed by the random partitions, which are constructed without taking the scenario tree structure into account. Therefore, in the remainder of this paper, we report only the tree-aligned partition bounds associated with multistage problem instances (unless stated otherwise).

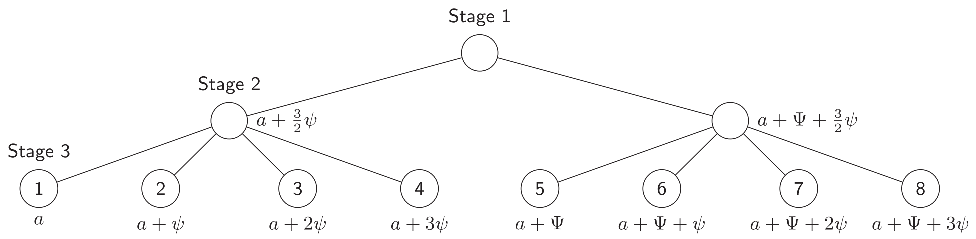
Clearly, there are potential advantages and disadvantages in both the tree-aligned and the misaligned partitioning strategies. The tree-aligned strategy violates as few NACs as possible, but may fail to represent the entire scenario set accurately in each group subproblem. The misaligned strategy, on the other hand, represents a better portion of the entire scenario set in every group subproblem, but violates many NACs. To understand the characteristics of the scenario tree that may determine which of these two partitioning strategies result in better bounds, we experiment with a small DCAP instance.

We conjecture that when the major distinction between the scenario tree nodes is observed in initial stages (closer to the root node), better representing the scenario set in group subproblems is critical, so the misaligned partitioning strategy provides better bounds. Similarly, when the major distinction between the nodes is observed in final stages (closer to the leaves), both partitioning strategies would represent the scenario set in group subproblems well, and therefore violating fewer NACs becomes more important in bound quality. In that case, the tree-aligned partitioning strategy has a potential advantage in providing better bounds.

To test these ideas, we use a DCAP instance with only 12 scenarios, as depicted in Figure 6. The only uncertain parameter in the DCAP instances is the capacity consumption of each task on each resource. To better quantify the distinction between various scenario tree nodes, we design the problem instance so that the capacity consumption of every task on every resource is the same for a given scenario. For example, in scenario 1, all capacity consumptions are equal to a, whereas in scenario 8, all capacity consumptions are equal to $a + \Psi + 3\psi$. The difference between the two penultimate stage nodes and the difference between leaf nodes originating from the same penultimate stage node are determined by Ψ and ψ, respectively. Based on our conjecture, we expect that increasing values of Ψ would favor the misaligned partitioning strategy, and increasing values of ψ would favor the tree-aligned partitioning strategy.

We create three settings based on the overall capacity consumption: low, medium, and high. In each of these settings, we experiment with three Ψ values and three ψ values. For each (Ψ, ψ) pair, the value of a is calculated based on the total capacity consumption, A, with $a = \frac{A-4\Psi-12\psi}{8}$. For every setting and every (Ψ, ψ) pair, we generate all tree-aligned and all misaligned partitions, compute the corresponding partition bounds, and compare the average tree-aligned bound with the average misaligned bound. The results obtained from the described experimental

Figure 6. Example Scenario Tree with $|\Xi^1 \times \Xi^2 \times \Xi^3| = 1 \times 2 \times 4$ for DCAP Instances



Note. The expression below each node represents the value of every uncertain parameter at the given stage and node.

setting are presented in Table 6. Values in the table represent the differences of average tree-aligned and average misaligned partition bounds, divided by the optimal objective. More specifically, $\delta^{T,M} = \frac{1}{OPT}\left(\frac{1}{|\Lambda^T|}\sum_{\Pi\in\Lambda^T} z^P(\Pi) - \frac{1}{|\Lambda^M|}\sum_{\Pi\in\Lambda^M} z^P(\Pi)\right)$, where Λ^T and Λ^M denote the collection of tree-aligned and misaligned partitions, respectively. Details of the percentage values in Table 6 are presented in Tables 9, 10, and 11 of the online supplement.

Table 6 verifies that larger differences between leaf nodes (small Ψ, large ψ) provide an advantage for the tree-aligned partitioning strategy, and larger differences between earlier tree nodes (large Ψ, small ψ) provide an advantage for the misaligned partitioning strategy. Unfortunately, in the extant benchmark instances, the level of the scenario tree at which the major distinction between the scenarios occurs is not always clear. Indeed, the dependence, if any, of uncertainty parameter probability distributions on the tree node at which they are sampled is not clearly articulated in the problem descriptions. Therefore, it is not easy to decide, a priori, which partitioning strategy would provide the best bounds for a specific problem instance. However, in the full-sized DCAP instances (L = 200, 300, and 500) and PLTEXP instances (L = 36, 216, and 1,296) used in our experimental study, there is no evidence pointing to a major distinction between the earlier nodes of the scenario tree. The capacity consumption realizations at each leaf node seem to be independent of their parent node. Under these circumstances, as expected, the tree-aligned partitioning strategy provides better partition bounds.

The method of recombining previously used scenario subsets into new, and possibly better, partitions is described in Section 5.2. Figure 5 depicts the results of a computational experiment in which optimal recombination is attempted after every 10 partitions on a DCAP instance. It can be observed that the best partition bound is improved whenever a recombined partition can be obtained.

Even though optimal recombination of partitions has great potential to provide improved partition bounds, it is important to note that it may not always be possible to feasibly recombine the groups of existing partitions into a new partition. Clearly, the tree-aligned partitioning strategy provides some benefits in this area, because it inherently divides the scenario set into smaller subsets based on the scenario tree structure. (The successful recombinations done while using a tree-aligned partitioning strategy can be observed in Figure 5.) However, in other partitioning strategies, such as random sampling with no consideration of the scenario tree structure, the possibility to recombine the groups of existing partitions into a new partition may not be very likely. It is intuitive that smaller group sizes provide more opportunities for recombination, so the likelihood of obtaining a new recombined partition from the existing partitions is greater for small group sizes. We provide support for this intuition in Section 3 of the online supplement.

In cases where it is possible to generate recombined partitions from existing partitions, optimally recombined partition bounds tend to provide more improvement to the incumbent bound when the group size is small. This phenomenon can be observed in Figure 5 and explained with the observation that the distribution of partition bounds have higher variability for smaller group sizes.

As mentioned in Section 5.3, to further improve the efficiency of partition sampling, we suggest a truncated bound calculation strategy, where we cease the bound calculation for unpromising partitions partway and start with a new partition.

Figure 7 demonstrates how bound truncation strategies affect the progression of the partition sampling algorithm on a DCAP instance with group size $q = 5$. It can be clearly seen that more aggressive truncation strategies result in fewer computations. Savings in computational effort comes at a cost of reduced bound quality in some cases, whereas in other cases bound quality remains the same. Figure 8 plots the savings in computational effort against the sacrifice in bound quality for different truncation strategies, where sacrifice in bound quality is expressed as the percentage difference between the bounds found using truncation strategies and the best partition bound without truncation.

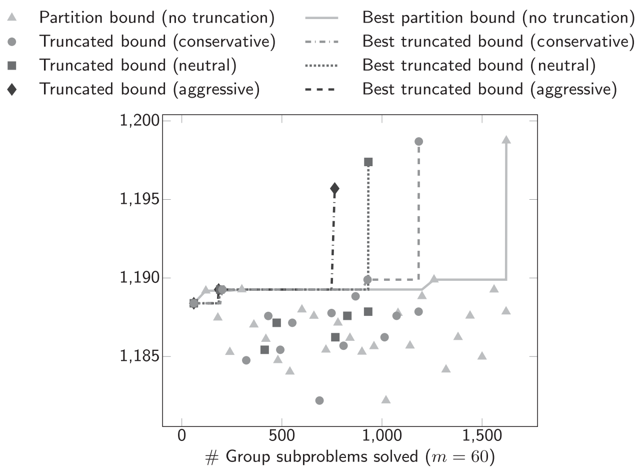
Figure 8(a)–(c) shows the trade-off between computational savings and loss in bound quality for different

Table 6. Difference of Tree-Aligned and Misaligned Partition Bounds in a Small DCAP 332 Instance

Low consumption (A = 2.7)				Medium consumption (A = 4.6)				High consumption (A = 9.0)			
	ψ				ψ				ψ		
Ψ	0.01	0.05	0.09	Ψ	0.01	0.11	0.21	Ψ	0.01	0.15	0.3
0	0.02%	0.08%	0.16%	0	0.01%	0.14%	0.22%	0	0.01%	1.20%	1.67%
0.2	−11.87%	−8.86%	−4.92%	0.25	−15.35%	−10.02%	−2.04%	0.7	−0.08%	0.13%	1.21%
0.4	−23.11%	−22.77%	−22.45%	0.5	−20.43%	−17.69%	−7.89%	1.3	−2.13%	−1.30%	0.76%

Notes. Plotted values, $\delta^{T,M} = \frac{1}{OPT}\left(\frac{1}{|\Lambda^T|}\sum_{\Pi\in\Lambda^T} z^P(\Pi) - \frac{1}{|\Lambda^M|}\sum_{\Pi\in\Lambda^M} z^P(\Pi)\right)$, are the percentage differences of average tree-aligned and misaligned partition bounds. Negative values indicate that the misaligned strategy gives the better bound.

Figure 7. Partition Bounds with Different Truncation Strategies (DCAP 332 1×15×20, $q = 5$)



Note. The last partition bound of every truncation strategy belongs to an optimally recombined partition, and (α, β, γ) values of (0.3, 0.02, 1.0), (0.35, 0.03, 0.985), and (0.4, 0.04, 0.96) are used to demonstrate *aggressive*, *neutral*, and *conservative* truncation strategies, respectively.

truncation strategies on an SSLP instance and a DCAP instance, respectively. Using the conservative strategy, more than 14% can be saved in computational effort while losing very little or nothing in terms of solution quality. Over 70% computational savings can be attained by eliminating partitions aggressively, while losing no more than 1.5% in bound quality. Similar results hold for other CAP, SSLP, and DCAP instances.

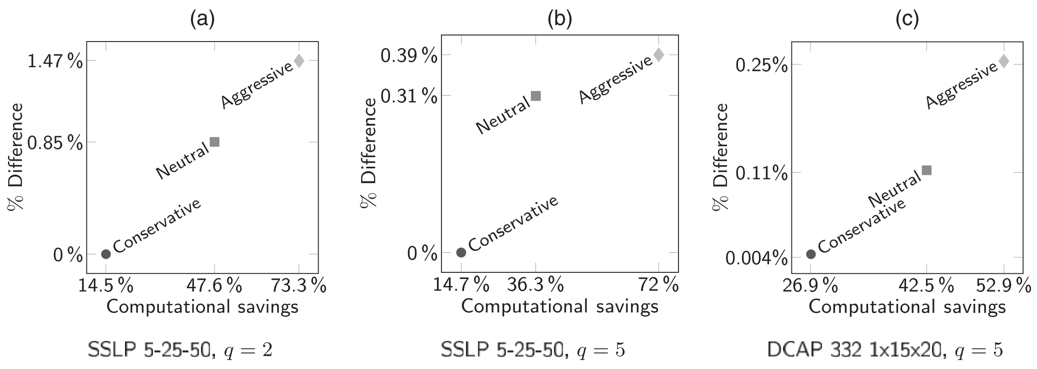
When truncation strategies are used, the saved computational effort can be used toward considering more partitions and therefore possibly improving the partition bound. To demonstrate this idea, we conduct experiments on some test instances, where new partitions are explored using the computational effort saved by using a conservative truncation strategy. The results of these experiments are presented in detail in Figure 9 for an SSLP and a DCAP instance. Figure 9(a) demonstrates an instance where 30 partitions ($30 \times 10 = 300$ group subproblems) are attempted originally, but the conservative truncation strategy results in solving only 256 group subproblems. The remaining computational effort, corresponding to 44 group subproblems, is used toward solving for four new partitions, one of which yields a better partition bound than the best partition bound. A similar result can be observed in Figure 9(b). Clearly, it is not guaranteed that the truncation strategies will not eliminate a partition that would provide a better bound than the current best, or that the new partition bounds calculated using the saved computational effort will result in an improved partition bound. But the examples we provide substantiate the potential of the truncation approach to save computational effort or to yield better partition bounds.

7. Conclusions and Future Work

We have shown that random sampling of partition bounds can be a simple yet effective approach to calculating dual bounds for SMIPs. On benchmark instances, it performs better than other sampling approaches for the same computational effort, and has the added benefit of providing a guaranteed bound, rather than one that is statistically likely. In practice, only a few partitions need to be sampled before the value of the best partition exceeds the expected value of group subproblems. It would be interesting to characterize classes of partitions or attributes of stochastic programs that ensure the distribution of partition bounds has the "right shape," that is, is sufficiently away from being right skewed, to ensure sampling is an attractive approach. This seems a quite difficult question to answer, but may be one that future work could shed some light on.

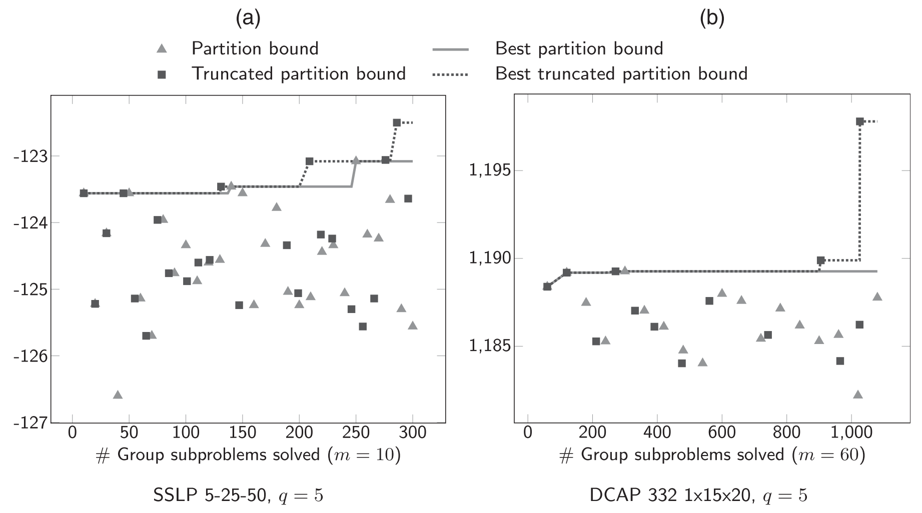
The ideas presented here have the potential to be extended in several directions. For any partition, the partition bound can be viewed as the special case of a bound obtained by Lagrangian relaxation of the NACs between scenarios from different subsets of the partition: the partition bound is simply the case with

Figure 8. Computational Savings vs. Loss in Bound Quality for Different Truncation Strategies



Note. The (α, β, γ) values (0.2, 0.02, 1.05), (0.4, 0.04, 0.99), and (0.5, 0.06, 0.95) are used to demonstrate *aggressive*, *neutral*, and *conservative* strategies for SSLP instances.

Figure 9. Bound Progression with Conservative Truncation vs. No Truncation (Same Computational Effort)



Note. The last truncated partition bound in panel (b) belongs to an optimally recombined partition.

zero Lagrange multipliers. Any method that provides better Lagrange multipliers for the partition can be used to improve the partition bound. This suggests that sampling of partitions may be combined with Lagrangian relaxation methods to alter both the partition and the Lagrange multipliers in tandem.

The calculation of partition bounds is naturally amenable to parallel implementation, and effective parallel codes could be developed in the future. These would be particularly helpful for cases in which the number of scenarios is very large.

Finally, we mention that more systematic resampling approaches could be explored. These might be designed, for example, so that new partitions reuse some previously solved group subproblems, while still exploring regroupings of the scenarios via randomness. Another idea would be to record the degree of NAC violation in partition bound solutions, by scenario pair, and seek to group scenarios that exhibit large violations when assigned to different group subproblems in a partition.

Acknowledgments

The authors thank the associate editor, area editor, and three anonymous referees for their constructive comments that led to substantial improvements in the paper.

Endnote

[1] SAA samples scenarios with replacement, allowing the same scenario to be sampled more than once, whereas the *EGSO* bound assumes that all scenarios in a group are distinct.

References

Ahmed S (2010) Two-stage stochastic integer programming: A brief introduction. Cochran JJ, Cox, Jr. LA, Keskinocak P, Kharoufeh JP, Smith JC, eds. *Wiley Encyclopedia of Operations Research and Management Science* (Wiley, Hoboken, NJ).

Ahmed S (2013) A scenario decomposition algorithm for 0–1 stochastic programs. *Oper. Res. Lett.* 41(6):565–569.

Ahmed S, Garcia R (2003) Dynamic capacity acquisition and assignment under uncertainty. *Ann. Oper. Res.* 124(1–4):267–283.

Ahmed S, Garcia R, Kong N, Ntaimo L, Parija G, Qui F, Sen S (2015) SIPLIB: A stochastic integer programming test problem library. Accessed December 21, 2015, http://www.isye.gatech.edu/~sahmed/siplib.

Aldasoro U, Escudero LF, Merino M, Pérez G (2013) An algorithmic framework for solving large-scale multistage stochastic mixed 0–1 problems with nonsymmetric scenario trees. Part II: Parallelization. *Comput. Oper. Res.* 40(12):2950–2960.

Beier E, Venkatachalam S, Corolli L, Ntaimo L (2015) Stage-and scenario-wise Fenchel decomposition for stochastic mixed 0-1 programs with special structure. *Comput. Oper. Res.* 59:94–103.

Birge JR (1985) Aggregation bounds in stochastic linear programming. *Math. Programming* 31(1):25–41.

Birge JR, Louveaux F (2011) *Introduction to Stochastic Programming* (Springer Science & Business Media, Berlin).

Bodur M, Dash S, Günlük O, Luedtke J (2016) Strengthened benders cuts for stochastic integer programs with continuous recourse. *INFORMS J. Comput.* 29(1):77–91.

Carøe CC, Schultz R (1999) Dual decomposition in stochastic integer programming. *Oper. Res. Lett.* 24(1–2):37–45.

Cheung K, Gade D, Silva-Monroy C, Ryan SM, Watson JP, Wets RJB, Woodruff DL (2015) Toward scalable stochastic unit commitment. *Energy Systems* 6(3):417–438.

Crainic TG, Fu X, Gendreau M, Rei W, Wallace SW (2011) Progressive hedging-based metaheuristics for stochastic network design. *Networks* 58(2):114–124.

Erdogan SA, Gose A, Denton BT (2015) On-line appointment sequencing and scheduling. *IIE Trans.* 47(11):1267–1286.

Escudero LF, Garín MA, Unzueta A (2016) Cluster Lagrangean decomposition in multistage stochastic optimization. *Comput. Oper. Res.* 67:48–62.

Escudero LF, Garín MA, Merino M, Pérez G (2010) On BFC-MSMIP strategies for scenario cluster partitioning, and twin node family branching selection and bounding for multistage stochastic mixed integer programming. *Comput. Oper. Res.* 37(4):738–753.

Escudero LF, Garín MA, Merino M, Pérez G (2012) An algorithmic framework for solving large-scale multistage stochastic mixed 0–1 problems with nonsymmetric scenario trees. *Comput. Oper. Res.* 39(5):1133–1144.

Escudero LF, Garín MA, Pérez G, Unzueta A (2013) Scenario cluster decomposition of the Lagrangian dual in two-stage stochastic mixed 0–1 optimization. *Comput. Oper. Res.* 40(1):362–377.

Espinoza D, Moreno E (2014) A primal-dual aggregation algorithm for minimizing conditional value-at-risk in linear programs. *Comput. Optim. Appl.* 59(3):617–638.

Guo G, Hackebeil G, Ryan SM, Watson JP, Woodruff DL (2015) Integration of progressive hedging and dual decomposition in stochastic integer programs. *Oper. Res. Lett.* 43(3):311–316.

Holmes D (1994) A collection of stochastic programming problems. Technical Report 94-11, Department of Operations Engineering, University of Michigan, Ann Arbor.

Holmes D (2018) A (PO)rtable (S)tochastic programming (T)est (S)et (POSTS). Accessed February 10, 2018, http://users.iems.northwestern.edu/~jrbirge/html/dholmes/post.html.

Huang K, Ahmed S (2009) The value of multistage stochastic programming in capacity planning under uncertainty. *Oper. Res.* 57(4):893–904.

Kim K, Zavala VM (2018) Algorithmic innovations and software for the dual decomposition method applied to stochastic mixed-integer programs*Math. Programming Comput.*10(2):225–266.

Klein Haneveld WK, van der Vlerk MH (1999) Stochastic integer programming: General models and algorithms. *Ann. Oper. Res.* 85(0):39–57.

Klein Haneveld WK, Stougie L, van der Vlerk MH (2006) Simple integer recourse models: Convexity and convex approximations. *Math. Programming* 108(2–3):435–473.

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

Løkketangen A, Woodruff DL (1996) Progressive hedging and tabu search applied to mixed integer (0, 1) multistage stochastic programming. *J. Heuristics* 2(2):111–128.

Louveaux FV (1986) Discrete stochastic location models. *Ann. Oper. Res.* 6(2):21–34.

Lubin M, Martin K, Petra CG, Sandıkçı B (2013) On parallelizing dual decomposition in stochastic integer programming. *Oper. Res. Lett.* 41(3):252–258.

Maggioni F, Pflug GC (2016) Bounds and approximations for multistage stochastic programs. *SIAM J. Optim.* 26(1):831–855.

Maggioni F, Allevi E, Bertocchi M (2014) Bounds in multistage linear stochastic programming. *J. Optim. Theory Appl.* 163(1):200–229.

Maggioni F, Allevi E, Bertocchi M (2016) Monotonic bounds in multistage mixed-integer linear stochastic programming. *Comput. Management Sci.* 13(3):423–457.

Mak W-K, Morton DP, Wood RK (1999) Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Oper. Res. Lett.* 24(1–2):47–56.

Norkin VI, Pflug GC, Ruszczyński A (1998) A branch and bound method for stochastic global optimization. *Math. Programming* 83(1–3):425–450.

Nowak MP, Schultz R, Westphalen M (2005) A stochastic integer programming model for incorporating day-ahead trading of electricity into hydro-thermal unit commitment. *Optim. Engrg.* 6(2):163–176.

Ntaimo L, Sen S (2005) The million-variable "march" for stochastic combinatorial optimization. *J. Global Optim.* 32(3):385–400.

Ntaimo L, Sen S (2008) A comparative study of decomposition algorithms for stochastic combinatorial optimization. *Comput. Optim. Appl.* 40(3):299–319.

Özaltin OY, Prokopyev OA, Schaefer AJ, Roberts MS (2011) Optimizing the societal benefits of the annual influenza vaccine: A stochastic programming approach. *Oper. Res.* 59(5):1131–1143.

Powell WB (2007) *Approximate Dynamic Programming: Solving the Curses of Dimensionality* (Wiley, New York).

Powell WB (2014) Clearing the jungle of stochastic optimization. *INFORMS TutORials in Operations Research*, 109–137.

Powell WB, Topaloglu H (2003) Stochastic programming in transportation and logistics. Ruszczynski A, Shapiro A, eds. Stochastic Programming. *Handbooks in Operations Research and Management Science*, vol. 10 (Elsevier, Amsterdam), 555–635.

Rockafellar RT, Wets RJB (1991) Scenarios and policy aggregation in optimization under uncertainty. *Math. Oper. Res.* 16(1):119–147.

Ruszczynski AP, Shapiro A (2003) *Stochastic Programming*, Handbooks in Operations Research and Management Science, vol. 10 (Elsevier, Amsterdam).

Ryan K, Rajan D, Ahmed S (2016) Scenario decomposition for 0-1 stochastic programs: Improvements and asynchronous implementation. *2016 IEEE Internat. Parallel Distributed Processing Sympos. Workshops (IPDPSW)* (Institute of Electrical and Electronics Engineers Computer Society, Piscataway, NJ), 722–729.

Sahinidis NV (2004) Optimization under uncertainty: State-of-the-art and opportunities. *Comput. Chemical Engrg.* 28(6–7):971–983.

Sandıkçı B, Özaltin OY (2017) A scalable bounding method for multistage stochastic programs. *SIAM J. Optim.* 27(3):1772–1800.

Sandıkçı B, Kong N, Schaefer AJ (2013) A hierarchy of bounds for stochastic mixed-integer programs. *Math. Programming* 138(1–2): 253–272.

Santoso T, Ahmed S, Goetschalckx M, Shapiro A (2005) A stochastic programming approach for supply chain network design under uncertainty. *Eur. J. Oper. Res.* 167(1):96–115.

Schultz R (2003) Stochastic programming with integer variables. *Math. Programming* 97(1–2):285–309.

Sen S, Higle JL (2005) The C^3 theorem and a D^2 algorithm for large scale stochastic mixed-integer programming: Set convexification. *Math. Programming* 104(1):1–20.

Sen S, Sherali HD (2006) Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Math. Programming* 106(2):203–223.

Sen S, Zhou Z (2014) Multistage stochastic decomposition: A bridge between stochastic programming and approximate dynamic programming. *SIAM J. Optim.* 24(1):127–153.

Shapiro A (2006) On complexity of multistage stochastic programs. *Oper. Res. Lett.* 34(1):1–8.

Shapiro A, Dentcheva D, Ruszczynski A (2014) *Lectures on Stochastic Programming: Modeling and Theory*, vol. 16 (SIAM, Philadelphia).

Sims MJ (1992) Use of a stochastic capacity planning model to find the optimal level of flexibility for a manufacturing system. Working paper, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor.

Song Y, Luedtke J (2015) An adaptive partition-based approach for solving two-stage stochastic programs with fixed recourse. *SIAM J. Optim.* 25(3):1344–1367.

van der Vlerk MH (2007) Stochastic integer programming bibliography. Accessed February 14, 2019, http://www.eco.rug.nl/mally/biblio/sip.html.

van der Vlerk MH (2010) Convex approximations for a class of mixed-integer recourse models. *Ann. Oper. Res.* 177(1):139–150.

Veliz FB, Watson JP, Weintraub A, Wets RJB, Woodruff DL (2015) Stochastic optimization models in forest planning: A progressive hedging solution approach. *Ann. Oper. Res.* 232(1):259–274.

Wallace SW, Fleten S-E (2003) Stochastic programming models in energy. Ruszczynski A, Shapiro A, eds. *Stochastic Programming*, Handbooks in Operations Research and Management Science, vol. 10 (Elsevier, Amsterdam), 637–677.

Watson JP, Woodruff DL (2011) Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Comput. Management Sci.* 8(4):355–370.

Zenarosa GL, Prokopyev OA, Schaefer AJ (2014a) M-SMIPLIB: A multistage stochastic mixed-integer programming test set library. Accessed October 17, 2016, http://www.cs.cmu.edu/~gzen/m-smiplib/.

Zenarosa GL, Prokopyev OA, Schaefer AJ (2014b) Scenario-tree decomposition: Bounds for multistage stochastic mixed-integer programs. Working paper, University of Pittsburgh, Pittsburgh.