Scenario Set Partition Dual Bounds for Multistage Stochastic Programming: A Hierarchy of Bounds and a Partition Sampling Approach

Ilke Bakir*[1], Natashia Boland†[1], Brian Dandurand‡[2], and Alan Erera§[1]

[1]Georgia Institute of Technology, USA
[2]Royal Melbourne Institute of Technology, Australia

Abstract

We consider multistage stochastic programming problems in which the random parameters have finite support, leading to optimization over a finite *scenario set*. We propose a hierarchy of bounds based on *partitions* of the scenario set into subsets of (nearly) equal cardinality. These *expected partition (EP) bounds* coincide with EGSO bounds provided by Sandıkçı et al. (2013) in the case that the subset cardinality divides evenly into the cardinality of the scenario set, and otherwise interpolate between the bounds for two consecutive cardinalities. Partition bounds have a substantial advantage for practical computation: solution of the group subproblem for all subsets in a *single* partition yields a dual bound. Furthermore, the best of even a very small number of such single partition bounds is very likely, in practice, to be better than the corresponding EP bound. By sampling partitions of the scenario set, we obtain an algorithm that provides exact dual bounds for the stochastic program. The practical value of these ideas is illustrated on benchmark problems, and the approach compared to Sample Average Approximation. Finally, we present three methods for improving the dual bounds obtained by the partition sampling algorithm: (i) Partition sampling using the scenario tree structure in multi-stage instances, (ii) generating optimally recombined partitions using the samples that were previously sampled in the algorithm, and (iii) truncating the partition bound calculations by ceasing computation of partitions that appear to be unpromising in terms of finding a good partition bound.

Key words: Stochastic programming

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, Birge and Louveaux (2011), Powell (2007, 2014), Ruszczynski and Shapiro (2003), Sahinidis (2004), Schultz (2003), Sen and Zhou (2014), Shapiro (2006), Shapiro et al. (2014) and the many references therein.

*ilkebakir@gatech.edu
†natashia.boland@isye.gatech.edu
‡b.dandurand@rmit.edu.au
§alan.erera@isye.gatech.edu

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. Computing dual bounds for measuring the quality of primal solutions is a key challenge.

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 *EGSO(q)* bound. They prove that the EGSO(q) bound is non-decreasing 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 a 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; the subsequent section covers the extensions described in Maggioni et al. (2016), and the working papers of Sandıkçı and Özaltin (2014), Zenarosa et al. (2014b).

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 $q-1$; such a partition is always possible provided q is not too large relative to the number of scenarios. We refer to the expected value of the partition bound over all partitions with groups of size nearly equal to q as EP(q). Observing that the EGSO bound of Sandıkçı et al. (2013) readily extends from the two-stage to the multi-stage setting, we prove that the expected value, over all partitions, of 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 increases monotonically in q.

This result has an important practical implication. Note that by solving (approximately) L/q SMIPs, each with q scenarios, a single partition bound results. We have observed in computational study that the distribution of partition bounds of typical benchmark instances is not very far from symmetric, and, in particular, the probability that a partition has above-average value is not small. Similar observations can be made from distributions provided in Sandıkçı and Özaltin (2014). 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. There is no need for all group subproblems of a given cardinality to be solved to compute high quality bounds. 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, for example, Ruszczynski and Shapiro (2003), Kleywegt et al. (2002)) has proven of great utility. In SAA, a set of scenarios is sampled, the corresponding group subproblem is solved, and the resulting solution evaluated using a, usually, larger, sample of the scenarios. In the case of a finite set of scenarios, which is the setting we consider here, it may be feasible to use all scenarios to evaluate the solution. 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

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

of the same size.

Here, we propose to *sample partitions* of the set of scenarios, with replacement, to ensure sample partitions are independent and identically distributed. This leads to an algorithm that provides exact dual bounds for the SMIP. The practical value of these ideas is illustrated on benchmark instances, showing that for far less computational effort, better bounds than the EGSO bounds can be found, and that partition bounds are better than those determined to be statistically likely using an SAA approach and the same computational effort.

The proposed method of obtaining exact dual bounds is enhanced by leveraging the scenario tree structure and constructing optimally recombined partitions from scenario subsets that are previously used in the algorithm. Furthermore, the possibility of improving the efficiency of the algorithm for providing the exact dual bound is explored using the observation that a heuristic estimate of the partition bound for a given partition can be obtained part-way through its computation, by using the values of the group subproblems computed so far. Strategies for terminating the solution of a partition part-way through, based on comparing the estimated partition bound with the best such bound found so far are tested computationally. We find that in many cases computational effort can be substantially reduced with very little impact on the quality of the final bound.

The remainder of this paper is organized as follows. Section 2 provides an overview of related literature. In Section 3, we introduce our notation, review the EGSO bounds, and extend these to the multistage case. In Section 4, we introduce the EP bounds, and prove that they yield a hierarchy of bounds. In Section 5, the algorithm based on independent sampling of partitions, that yields an exact dual bound, is presented. In Section 6, we provide the results of computational experiments demonstrating the practical value of these ideas on benchmark problems. We give our conclusions and discuss promising extensions to this work in Section 7.

2 Literature Review

Application domains for multistage SMIPs range broadly, from transportation and logistics (Arda et al. 2014, Maggioni et al. 2016, 2015, Powell and Topaloglu 2003, Santoso et al. 2005), to manufacturing (Huang and Ahmed 2009), mining (Ramazan and Dimitrakopoulos 2013), energy (Wallace and Fleten 2003, Nowak et al. 2005, Stoyan and Dessouky 2012, Cheung et al. 2015), forest management (Veliz et al. 2014), wildfire response planning, (Ntaimo et al. 2012), healthcare (Özaltin et al. 2011, Pérez et al. 2013, Zhang et al. 2014, Erdogan et al. 2015) and disaster preparedness (Mete and Zabinsky 2010). Substantial benefits when using multistage instead of two-stage models have been shown for many problems (Huang and Ahmed 2009). Other works demonstrate the benefits of solving over multiple stages, as opposed to the use of myopic strategies (Arda et al. 2014).

Challenges in solving multistage stochastic programs with integer variables are well documented; see, for example, Ahmed (2010), Klein Haneveld and van der Vlerk (1999), Schultz (2003), Sen and Zhou (2014). These challenges have spurred many recent algorithmic developments, with increasing interest in extensions to risk averse measures (Bruno et al. 2015) and to the use of decision rules Bertsimas and Georghiou (2015). However the most substantial effort to date has focused on decomposition approaches. One line of work has developed approaches based on 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), 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 vig-

orously 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, in which feasible first-stage solutions are iteratively eliminated by no-good constraints, allowing both the dual and primal bound available from solving all scenarios subproblems to be improved at each iteration, until optimality is proved. The computational efficacy of a synchronous parallel implementation of this approach is demonstrated in Ahmed (2013), with algorithmic improvements, and an asynchronous parallel implementation is provided in Ryan et al. (2015). For two-stage stochastic mixed integer programming, Kim and Zavala (2015) develop software based on parallel implementation of a scenario decomposition method that uses Lagrangian relaxation to improve the dual bounds, with an interior point method to solve the Lagrangian master problem, and the addition of Benders-type cuts to strengthen the scenario subproblems. 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), that has been used as an effective primal heuristic for SMIPs (Cheung et al. 2015, Crainic et al. 2011, Løkketangen and Woodruff 1996, Veliz et al. 2014, Watson and Woodruff 2011), including parallel implementations. Its connections with dual decomposition have been exploited recently 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 non-anticipativity constraints between scenarios in the set, but the non-anticipativity constraints between scenarios in different groups relaxed. This idea was exploited by Escudero et al. (2010, 2012), Aldasoro et al. (2013) in the context of branch-and-fix coordination algorithms, and by Escudero et al. (2013, 2016) with non-anticipativity 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 non-increasing 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 multistage SMIPs via the solution of many scenario group subproblems. Working papers of Sandıkçı and Özaltin (2014), Zenarosa et al. (2014b) and Maggioni et al. (2016) describe extensions of these ideas. Sandıkçı and Özaltin (2014) 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, and 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 (2014) 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 $L \mod q$, 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 (2014), Zenarosa et al. (2014b) 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 to include more scenarios. 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.

In this paper, we compare the best of a random sample of scenario partition bounds with bound estimates calculated using Sample Average Approximation (SAA) (e.g., Norkin et al. (1998), Mak et al. (1999), Ruszczynski and Shapiro (2003), Kleywegt et al. (2002)). The SAA approach is also known under various names, such as the stochastic counterpart method (Rubinstein and Shapiro 1990) and the sample path optimization method (Plambeck et al. 1996). A variety of convergence results are available, including for linear, pure integer and mixed integer cases (Schultz 1996, Shapiro and Homem-de Mello 2000, Ahmed and Shapiro 2002, Kleywegt et al. 2002). In combination with decomposition algorithms, SAA has been used to successfully solve stochastic linear programs of enormous size (Linderoth et al. 2006), and to tackle two-stage stochastic integer programs with a very large number of scenarios, as, for example, by Morton and Wood (1998), Ahmed and Shapiro (2002), Verweij et al. (2003), Schütz et al. (2009). SAA also forms the basis for a variety of approximations schemes for certain classes of stochastic combinatorial optimization problems, including for multistage problems; see, e.g., Bertsimas et al. (2011), Gupta et al. (2011), Swamy and Shmoys (2012), Romeijnders et al. (2014).

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 is 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 $1, \ldots, t$. Since 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 (MSP) to have the following form:

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

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

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

and where 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 $\mathcal{X}^t$, 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).

Since $\tilde{\xi}$ 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 $\mathcal{H}(t, s)$ 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 the (MSP) in its *extensive form* as

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

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 $F_t^s(x_t^s) = f_t(x_t^s, \xi_{[t]}^s))$,

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

and $y_h \in \mathbb{R}^{n_t}$ for each scenario tree node h at stage t is an auxilliary variable introduced to model the *non-anticipativity constraints (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 $S \subseteq \mathcal{S}$ of the scenarios, denoted by $GR(S)$ to be

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

where

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

and we define

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

Note that, as in Sandıkçı and Özaltin (2014), we use a simpler form of the group subproblem to 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 $v_{GR}(\mathcal{S}) = z_{GR}(\mathcal{S}) = z_{MSP}$ and that for each $s \in \mathcal{S}$, the value of the group subproblem for the set consisting of the single scenario, s, is simply

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

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

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

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

We now replicate the key result of Sandıkçı et al. (2013), adjusted to fit our context, and discuss how they extend to the multistage context.

The *expected group subproblem objective* for an integer q with $1 \leq q \leq L = |\mathcal{S}|$ 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 $\mathcal{S}$ of cardinality q.

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

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

The proof of Theorem 1, as well as some preliminary results leading to it are provided in Appendix A for completeness. Note the slight difference with the result in Sandıkçı et al. (2013), where $z_{WS} \leq EGSO(1)$ rather than $z_{WS} = EGSO(1)$: this is due to our 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, $\mathcal{S}$, so $\Pi = \{S_1, \ldots, S_m\}$, for some m, with $S_i \subseteq \mathcal{S}$ for all $i = 1, \ldots, m$ and $S_i \cap S_{i'} = \emptyset$ for $i' = 1, \ldots, m$ with $i' \neq i$. 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 $\mathcal{S}$, with $\Pi = \{S_1, \ldots, S_m\}$, 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*

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

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 positive integer q, the q-partition set of $\mathcal{S}$, denoted by Λ_q, is the set of all partitions of $\mathcal{S}$ 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 $L = |\mathcal{S}|$.*

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

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

where q, m', m and L are as defined in Definition 1. Furthermore, in the case that q divides evenly into L, it must be that $m' = 0$, and the q-partition set Λ_q partitions the scenario set into subsets of cardinality exactly q, and hence into subsets contained in Ω_q.

Note that if q is "too large" relative to $L = |\mathcal{S}|$, then it is possible that $m = \lceil \frac{L}{q} \rceil - (q\lceil \frac{L}{q} \rceil - L)$ could be negative, and hence Λ_q ill-defined. Note also that, to be of practical interest, partitions should consist of sets having size small relative to the entire set of scenarios, otherwise the benefits 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. This observation is straightforward to verify.

Observation 1. *Given that $r = L - q\lfloor \frac{L}{q} \rfloor$, the q-partition of $\mathcal{S}$, Λ_q, is well defined if and only if*

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

Proof. Define $p, r \in \mathbb{Z}_+$ so that $L = pq + r$, and $r < q$.

When $q|L$, $m' = 0$ and $m = p > 0$. When $q \nmid L$,

$$m' = q\left\lceil \frac{L}{q} \right\rceil - L = q(p+1) - (pq + r) = q - r$$
$$m = \left\lceil \frac{L}{q} \right\rceil - m' = p + 1 - (q - r)$$

The q-partition, Λ_q, is well-defined if and only if $m > 0$.

$$\begin{aligned} m \geq 1 \iff q|L \text{ or } p + 1 - (q-r) \geq 1 &\iff q|L \text{ or } p \geq q - r \\ &\iff q|L \text{ or } L \geq (q-r)q + r \end{aligned}$$

□

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 (EP) 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, since 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

$$|\Lambda_q| = \left(\begin{array}{c} L \\ \ell \end{array} \right) \frac{\ell!}{(q!)^m m!} \frac{\ell'!}{((q-1)!)^{m'} m'!},$$

which can easily be simplified to

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

Provided that $P \in \Omega_q$ denotes a subset of $\mathcal{S}$ of size q, a partition containing P can be defined with $S_q \subset \mathcal{S} \setminus P$, the set of scenarios contained in subsets of size q; and $S_{q-1} = \mathcal{S} \setminus (S_q \cup P)$, the set of scenarios contained in subsets of size $q-1$.

Each distinct subset P of $\mathcal{S}$ of size q appears in

$$\eta_q := \left(\begin{array}{c} L-q \\ \ell - q \end{array} \right) \frac{(\ell - q)!}{(q!)^{(m-1)}(m-1)!} \frac{\ell'!}{((q-1)!)^{m'} m'!}$$

partitions; where the first, second, and third terms represent the number of ways to choose the scenarios that will be in S_q, the number of ways to partition S_q into $m-1$ groups of size q, and the number of ways to partition S_{q-1} into m' groups of size $q-1$, respectively.

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

$$\eta'_q := \left(\begin{array}{c} L-q+1 \\ \ell \end{array} \right) \frac{\ell!}{(q!)^m m!} \frac{(\ell' - q + 1)!}{((q-1)!)^{(m'-1)}(m'-1)!}$$

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

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

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,

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

Thus

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

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 condition (1) holds,

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

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 condition (1) holds. Then $m \ge 1$, 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. Since $EGSO(q-1) \le EGSO(q)$, the result follows. □

This result substantiates our claim 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 that 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

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

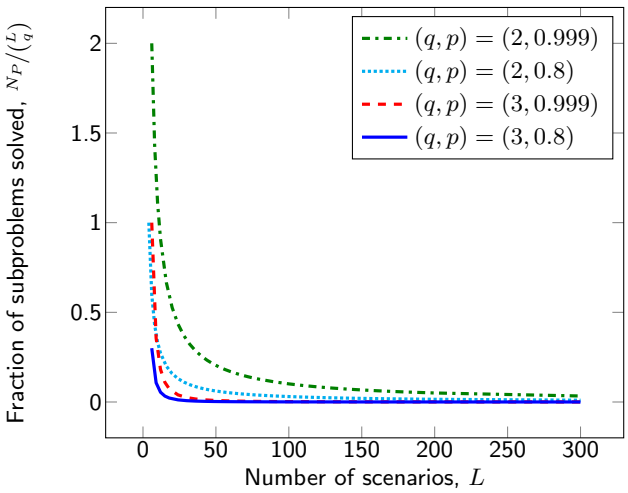


Figure 1: Best Partition vs. Expected Partition

Note. q: Group size, p: Target probability of finding a better partition bound than $EP(q)$ after solving N_P subproblems, $N_P = \frac{L}{q} \times \lceil \log_2 \frac{1}{1-p} \rceil$.

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. Since its average is exactly $EP(q)$, we have that

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

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 $\binom{24}{3} = 2024$ group subproblems, each with 3 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 increase, the fraction of subproblems that need to be solved in order 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 $Pr(z^P(\Pi) \geq EP(q)) = 0.2$ 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. Our observations on benchmark instances (presented in the online supplement) , which confirm the observations made by Sandıkçı and Özaltin (2014), suggest that these distributions are not highly skewed.

This motivates Algorithm 1, which computes exact lower bounds from randomly sampled q-partitions. The stopping criterion for the algorithm 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 upper bounds during the course of the algorithm, making use of the solution to each group subproblem. For example, Sandıkçı and Özaltin (2014) describe such a heuristic.

Algorithm 1 provides effective dual bounds, as we discuss in depth in Section 6. Even though randomly sampled partitions provide effective results, we explore several opportunities to improve the partition bounds by selecting partitions more intelligently. The remainder of this section is dedicated to describing these methods that can lead to better partition bounds without compromising the computational simplicity of the algorithm.

Algorithm 1 *Scenario Set Partition Sampling*

$LB := -\infty$, $k := 1$
while $k \leq K^{max}$ **do**
  $S := \mathcal{S}$
  for all $i = 1, \ldots, \lceil \frac{L}{q} \rceil$ **do**
    / Sample, without replacement, the next subset in the partition */*
    if $i \leq L - (q-1)\lceil \frac{L}{q} \rceil$ **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 */*
  $LB := \max\{LB, z_k\}$
  $k := k + 1$
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 to random partition sampling, due to less NACs being violated.

Algorithm 2 defines our method of generating q-partitions that groups scenarios with common nodes together, called the "tree-aligned" partitioning strategy. (For simplicity, $q|L$ is assumed.) It randomly selects a node j^* from the set of scenario tree nodes associated with the penultimate stage, $\mathcal{N}_{-1}$; and assigns randomly selected leaves originating from j^* into subsets $S_i, \quad i = 1, \ldots, \frac{L}{q}$. A few tree-aligned partitions on a small scenario tree are shown in Figure 2. Note that we use this method on 3-stage instances for our computational experiments, which means that all of the nodes in $\mathcal{N}_{-1}$ originate from the same root node. Therefore selecting nodes j^* randomly from $\mathcal{N}_{-1}$ is sensible. For problem instances with more than 3 stages, the algorithm can easily be modified to select nodes j^* by ensuring that they share common nodes in previous stages. This way, the least possible number of NACs will be violated.

To evaluate the outcomes of this method, we test the tree-aligned ("tree", in short) partitioning strategy against two others: "random" partitioning strategy, which randomly partitions the scenario set without considering the tree structure; and "misaligned" partitioning strategy, which keeps the scenarios with common nodes in separate groups as much as possible. The different partitioning strategies can be seen on a 3-stage example with 9 scenarios in Figure 3. Our computational results on benchmark instances, as presented in Section 6, suggest that the "tree" partitioning strategy provides considerable improvements in bound quality when tested against the other two strategies.

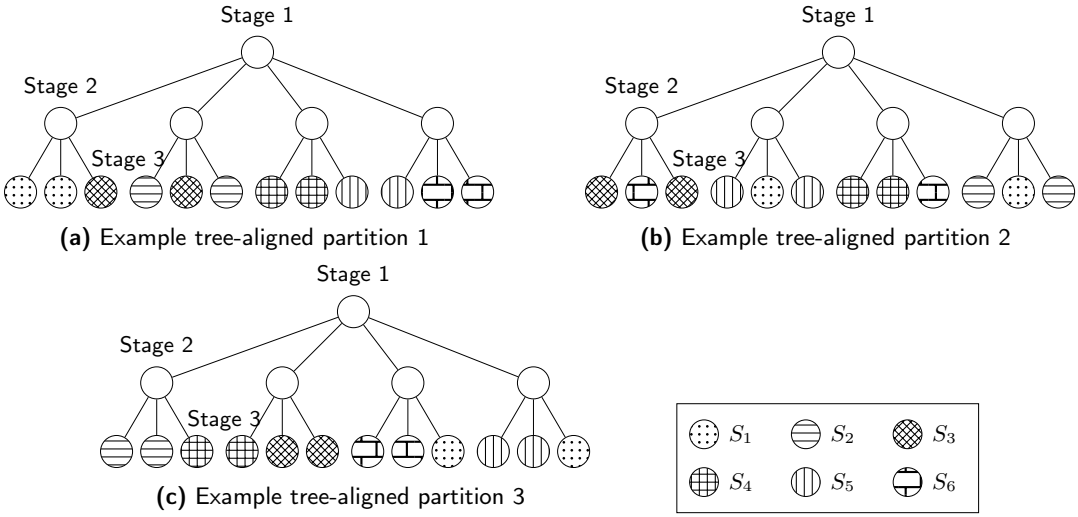


Figure 2: Example tree-aligned partitions

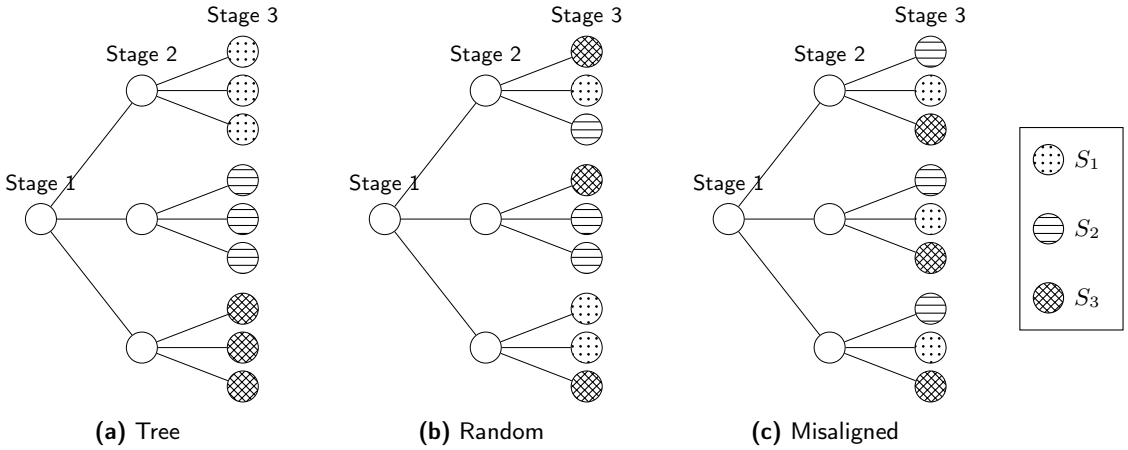


Figure 3: Methods to select partitions based on scenario tree structure

5.2 Optimally Recombined Partitions

As the scenario set partition sampling algorithm randomly samples partitions, it computes the group subproblem objectives, $z_{GR}(S)$, associated with scenario subsets, S, of these partitions. Once the algorithm has progressed sufficiently, 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 $\mathcal{C}(i)$ denote the collection of scenario subsets containing scenario i. An optimally recombined partition can be found by solving

$$\underset{x\in\{0,1\}^{|\mathcal{C}|}}{\arg\max}\{\sum_{S\in\mathcal{C}} z_{GR}(S)x_S : \sum_{S\in\mathcal{C}(i)} x_S = 1, \quad \forall i \in \mathcal{S}\}.$$

We incorporate the optimal recombination into the partition sampling algorithm by solving the set partitioning problem at certain iterations. If a recombined partition, which is different from the previously

Algorithm 2 *Partitioning Strategy, "Tree-Aligned"*

$\mathcal{N}_{-1}$: The set of scenario tree nodes associated with the stage before last
$\mathcal{N}_j$: The set of scenarios (leaves) originating from scenario tree node $j \in \mathcal{N}_{-1}$
$j^* :=$ Randomly picked node from $\mathcal{N}_{-1}$
$\mathcal{N}_{-1} := \mathcal{N}_{-1} \setminus \{j^*\}$
for all $i = 1, \ldots, \frac{L}{q}$ **do**
  $S_i := \emptyset$
  while $|S_i| < q$ **do**
    $k :=$ Randomly picked scenario from $\mathcal{N}_{j^*}$
    $S_i := S_i \cup \{k\}$
    $\mathcal{N}_{j^*} := \mathcal{N}_{j^*} \setminus \{k\}$
    if $\mathcal{N}_{j^*} = \emptyset$ **then**
      $j^* :=$ Randomly picked node from $\mathcal{N}_{-1}$
      $\mathcal{N}_{-1} := \mathcal{N}_{-1} \setminus \{j^*\}$
    end if
  end while
end for

sampled partitions and provides a better bound than the current best, is found, then we select that partition and update the current best accordingly. Otherwise, the algorithm continues to randomly 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, since 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 computational effort by using the group subproblem values calculated part-way 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.

In order 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 *portion of the group subproblems are solved, the heuristic estimate of* $z^P(\Pi)$ *remains lower than* γ *times the best lower bound, for the next* r' *group subproblems solved, where* r' *is at least* β *portion 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 elimination rule parametrized by (α, β, γ), we characterize a partition Π as "unpromising" if

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

provided $r \geq \lceil \alpha m \rceil$ and $r' = \lceil \beta(m-r) \rceil - 1$. Note that eliminating a partition, i.e., 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. The higher the values of α and β, and the lower the value of γ, the more conservative the 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 towards promising partitions, resulting in an improved partition bound.

6 Computational Results

Computational experiments are performed for three different classes of problems, namely, stochastic capacitated facility location problem (CAP), stochastic server location problem (SSLP), and dynamic capacity acquisition and allocation problem (DCAP). The former two classes of problems are 2-stage, and the latter is 3-stage. Computations are run on a heterogeneous cluster of machines with Xeon E5520, Xeon E5-2670, E5-2650v3 and Xeon E5430 processors using 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. For each of those values, random q-partitions are sampled and the resulting partition bounds are observed. Partition bounds are compared against Sample Average Approximation (SAA) estimates found by solving the same size and number of group subproblems and confidence intervals around those estimates. The distribution of partition bounds are examined for spread and symmetry. Furthermore, the methods described in Sections 5.1, 5.2, and 5.3 to improve the partition sampling algorithm are implemented and their effects are studied.

6.1 Test Instances

The stochastic capacitated facility location problem (CAP) considered in this paper is described in 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. (2014). Each instance has 5000 scenarios. Here, we use the first 240 of them. Group sizes $q = 5, 10, 15, 20$ are used for CAP experiments.

Partiton bounds for reasonable group sizes can easily be computed for the CAP instances with 5000 scenarios. Our experiments reveal that a partition bound can be computed in around 57 hours with $q = 5$, and 77 hours with $q = 10$. This corresponds to around 3.5 minutes and 9 minutes per group subproblem for group sizes 5 and 10, respectively. So, especially in a parallel computing environment, finding partition bounds for the 5000-scenario CAP instances is possible. However, directly solving the problem instances (for comparison purposes) is not possible with reasonable computing power. Therefore, in order to have an optimal objective value for reference, we chose to use first 240 scenarios. The computational results we obtain from the 240-scenario instances clearly apply to larger problems.

The stochastic server location problem (SSLP) considered in this paper is described in 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 assignment of clients to servers. Group sizes $q = 2, 5, 10, 25$ are used in experiments on SSLP instances.

The dynamic capacity acquisition and allocation problem (DCAP) is described in Ahmed and Garcia

(2003), and the 3-stage instances used in computational experiments of this paper are 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 as "DCAP $n_i n_j 2$ 1xn_2xn_3", where n_i and n_j represent the number 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 $q = 2, 5, 10, 20, 30, 40, 50$.

6.2 Comparing Partition Sampling with SAA

To compare partition bounds found by sampling with Sample Average Approximation (SAA) estimates, for each instance and each q value tested, we run Algorithm 1 with $K^{max} = 30$, requiring solution of $n' = 30L/q$ 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 as $z_{SA}(S)$ is found by solving the following problem:

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

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 associated with the sample average is $\hat{s}_{SAA}^2 = \frac{1}{n'-1}\sum_{i=1}^{n'}\left(z_{SA}(S_i) - \hat{z}_{n'}^{SAA}\right)^2$ and the confidence interval with a level of confidence α is $\hat{z}_{n'}^{SAA} \pm t_{\frac{\alpha}{2},n'-1}\sqrt{\frac{\hat{s}_{SAA}^2}{n'}}$.

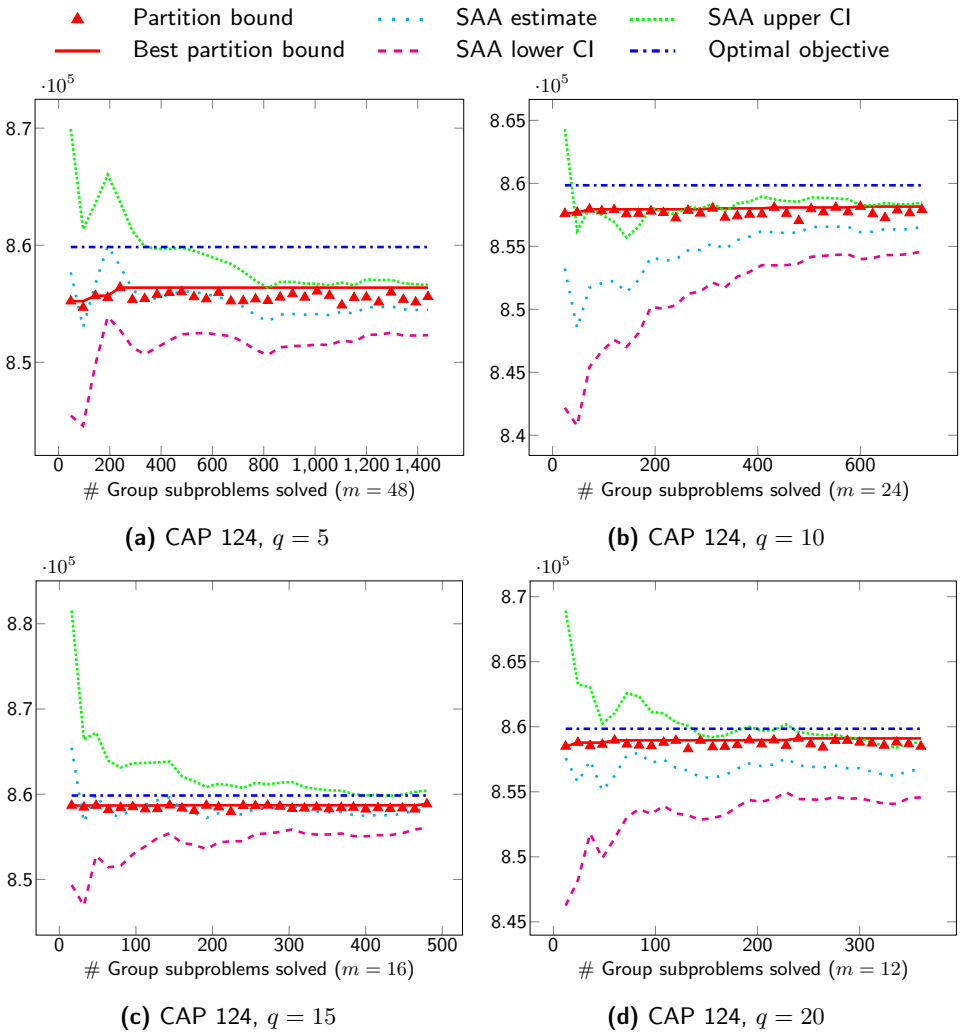
Figure 4 illustrates, using a typical CAP instance, how partition bounds compare to SAA estimates and 95% confidence intervals around SAA estimates. Partition bounds associated with 30 independent partitions are presented, along with the SAA estimates calculated by solving the same size and number of 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 higher 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 3-stage DCAP instance, when the "tree" partition selection 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 at the very beginning of the algorithm, while SAA estimates require a considerably long warm-up phase.

Similar behavior is observed in all CAP, SSLP, and DCAP instances. Tables 1 and 2 present

- ***(i)*** the gap obtained from the best partition bound after $k = 30$ q-partitions , Δ_k^P (Equation 5),
- ***(ii)*** 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} (Equation 6), and
- ***(iii)*** the proportion of the latter gap that is closed by the best partition bound, $\delta_k^{SAA,P}$ (Equation 7).

The gaps are calculated respective to the wait-and-see solution, z_{WS}. Since the test instances have a considerable amount of "sunk cost", which is the cost that has to be incurred regardless of solution quality,



(a) CAP 124, $q = 5$ **(b)** CAP 124, $q = 10$

(c) CAP 124, $q = 15$ **(d)** CAP 124, $q = 20$

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

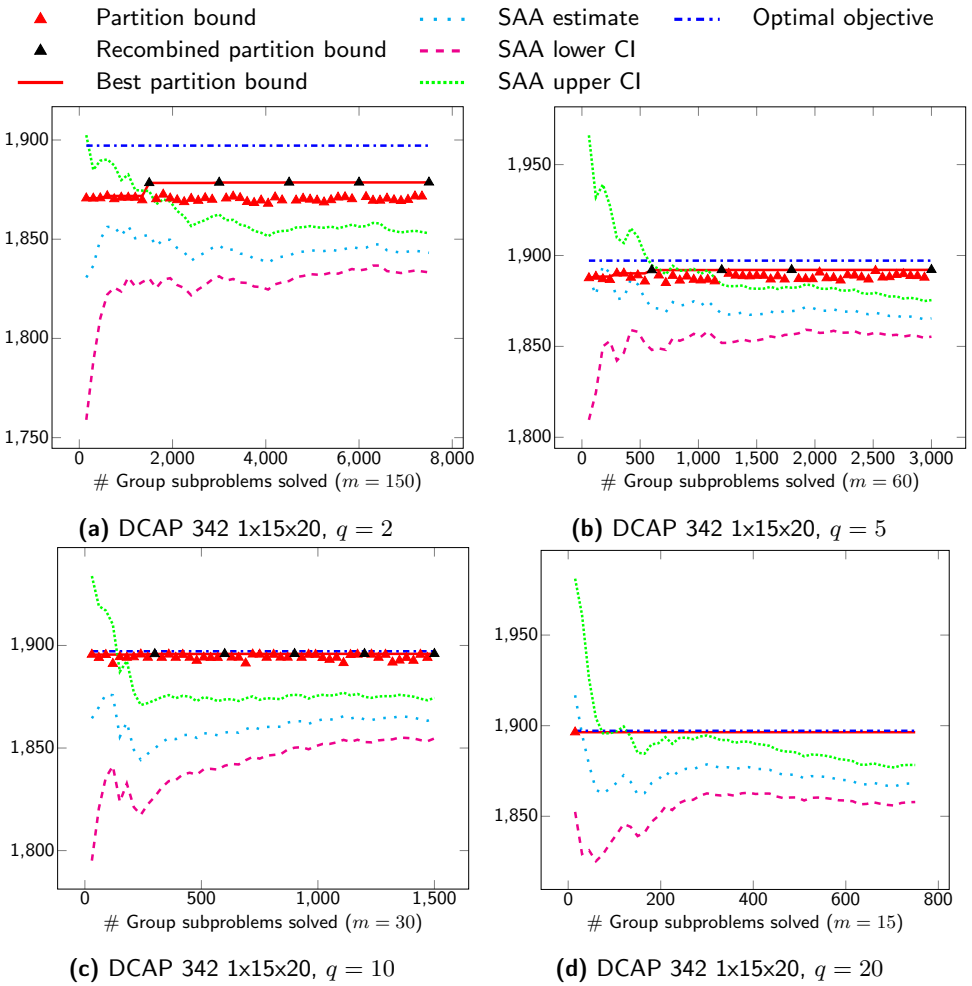
the objective values associated with different solutions do not seem very different from each other. In order to be able to objectively compare different solutions in terms of quality, we subtract the lowerbound z_{WS} from the objective values in our reporting. Provided that OPT represents the the optimal value over all L scenarios, the gaps are calculated as follows:

$$\Delta_k^P = \frac{(OPT - z_{WS}) - (max_{i=1,\dots,k}\{z^{P_i}\} - z_{WS})}{OPT - z_{WS}} \tag{5}$$

$$\Delta_k^{SAA} = \frac{(OPT - z_{WS}) - \left(\hat{z}_{km}^{SAA} - t_{0.025,km-1}\sqrt{\frac{\hat{s}_{SAA}^2}{km}} - z_{WS}\right)}{OPT - z_{WS}}, \text{ where } m = \frac{L}{q} \tag{6}$$

$$\delta_k^{SAA,P} = \frac{\Delta_k^{SAA} - \Delta_k^P}{\Delta_k^{SAA}} \tag{7}$$

Table 1 demonstrates the gaps Δ_{30}^P, Δ_{30}^{SAA}, $\delta_{30}^{SAA,P}$ for the SSLP instances with different group sizes q, and Table 2 gives the means and standard deviations of Δ_{30}^P, Δ_{30}^{SAA} and $\delta_{30}^{SAA,P}$, 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 Appendix B that in the majority of instances, the best partition bound is attained within 10 or 20 partitions.



(a) DCAP 342 1x15x20, $q = 2$ **(b)** DCAP 342 1x15x20, $q = 5$

(c) DCAP 342 1x15x20, $q = 10$ **(d)** DCAP 342 1x15x20, $q = 20$

Figure 5: DCAP 342 1x15x20 Partition (with "tree" partitioning strategy and optimal recombination) vs. SAA ($K^{max} = 50$)

Table 1: Best partition bound vs. lower limit of the 95% SAA confidence interval on SSLP instances.

Instance	L	q	Δ_{30}^{P}	Δ_{30}^{SAA}	$\delta_{30}^{SAA,P}$
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 2: Best partition bound vs. lower limit of the 95% SAA confidence interval: summary for CAP instances.

L	q	Δ_{30}^{P}		Δ_{30}^{SAA}		$\delta_{30}^{SAA,P}$	
		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%

6.3 Improvements with Algorithmic Enhancements

For problem instances with more than 2 stages, the method of sampling partitions based on scenario tree structure is described in Section 5.1. The results of our comprehensive computational experiments with 3-stage DCAP instances are reported in Tables 8 and 9 of Appendix B. It can be clearly seen that the "tree" partitioning strategy, which violates less NACs than the other partitioning strategies, provides the best partition bounds. Therefore, from this point on in this paper, we only report the "tree" partition bounds associated with the DCAP instances.

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 that provides a better bound than the current best bound can be obtained. Optimal recombination appears to be particularly beneficial for smaller group sizes.

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

Figure 6 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 less computations. Savings in computational effort comes at a cost of reduced bound quality in some cases, while in other cases bound quality remains the same. Figure 7(c) 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.

Figures 7(a) and 7(b) show the tradeoff between computational savings and reduction in bound quality for different truncation strategies on an SSLP instance. Using the conservative strategy, up to about 20% 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 towards considering more partitions and therefore possibly improving the partition bound. To demonstrate this idea, we performed experiments on an SSLP and a DCAP instance, where new partitions are explored using the saved computational effort due to the conservative truncation strategy. The results of these experiments are presented in detail in Figure 8. Figure 8(a) demonstrates an instance where 30 partitions ($30 \times 10 = 300$ group subproblems) are attempted originally, but the conservative elimination strategy results in solving

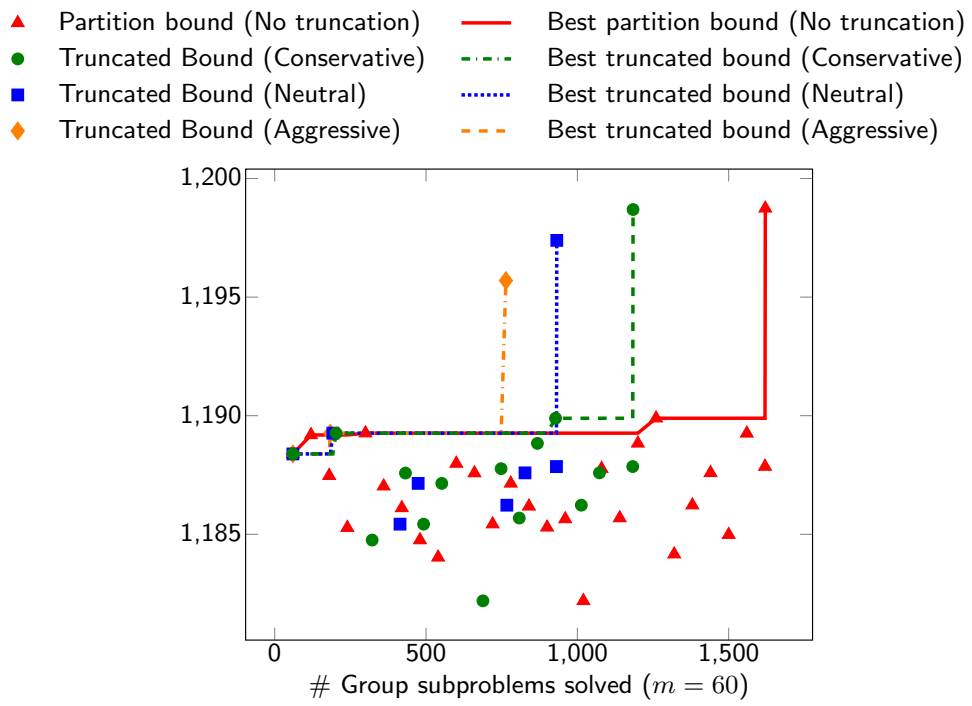
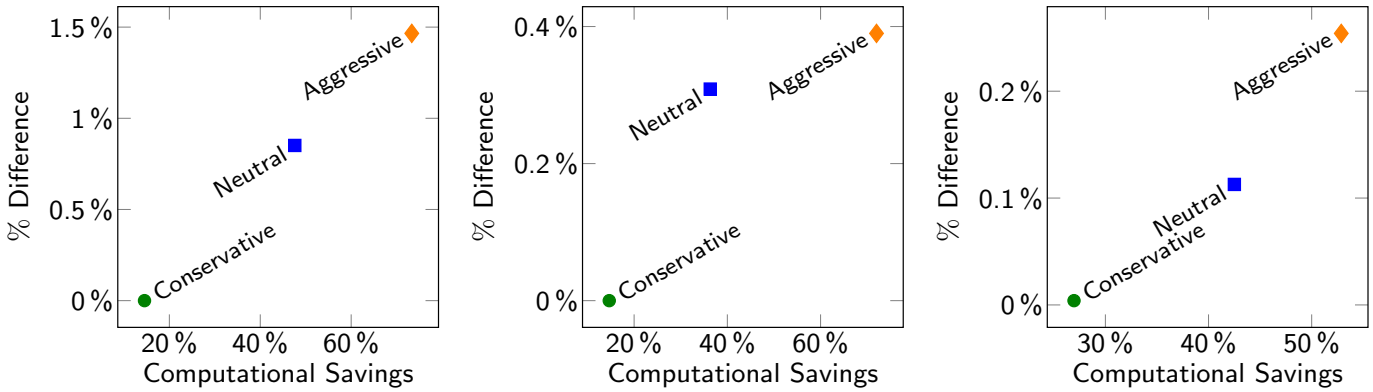


Figure 6: Partition bounds with different truncation strategies. DCAP 332 1x15x20, $q = 5$.

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.

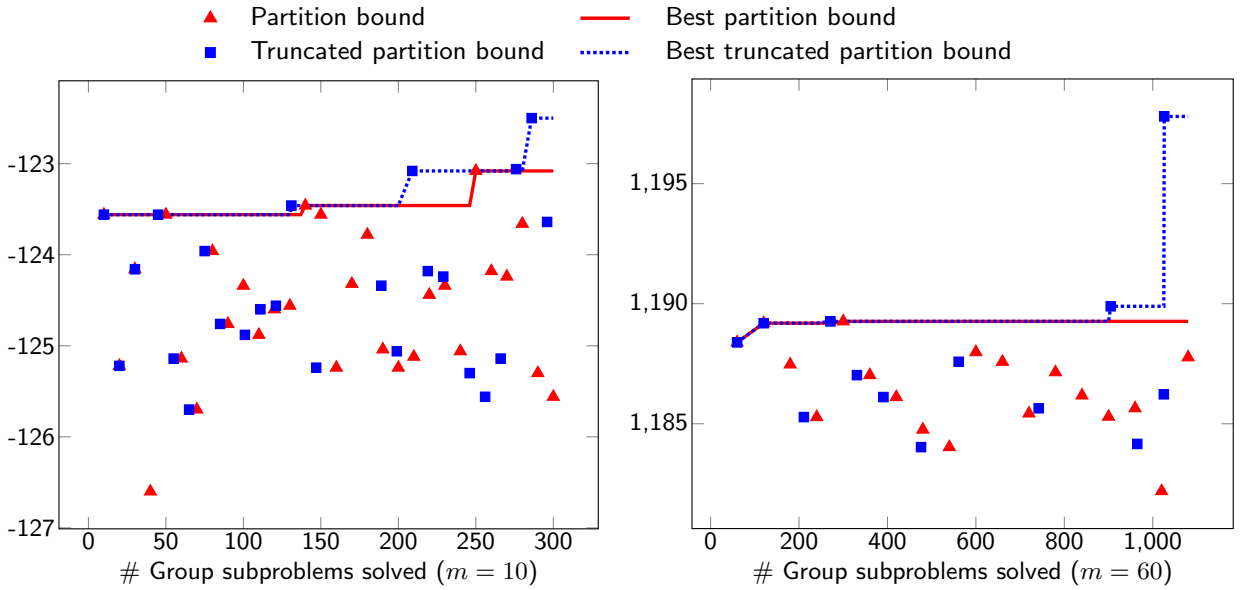
only 256 group subproblems. The remaining computational effort, corresponding to 44 group subproblems, is used towards solving for 4 new partitions, one of which yielded a better partition bound than the best partition bound. A similar result can be observed in Figure 8(b). Clearly, it is not guaranteed 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 yield better partition bounds.



(a) SSLP 5-25-50, $q = 2$ **(b)** SSLP 5-25-50, $q = 5$ **(c)** DCAP 332 1x15x20, $q = 5$

Figure 7: Eliminating partitions.

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



(a) SSLP 5-25-50, $q = 5$ **(b)** DCAP 332 1x15x20, $q = 5$

Figure 8: No elimination vs. conservative elimination (SSLP 5-25-50, $q = 5$)

Note. The last truncated partition bound on Figure (b) belongs to an optimally recombined partition.

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. 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.

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 zero

Lagrange multipliers. Any method that provides better Lagrange multipliers for the partition can be used to improve the partition bound, and yield better bounds. 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 re-sampling approaches could be explored. These might be designed, for example, so that new partitions re-use some previously solved group subproblems, while still exploring re-groupings 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.

Appendix A Proofs of Lemmas and Theorems

Lemma 1 (Sandıkçı et al. (2013), Lemma 1)**.** *Given integers q_1 and q_2 with $1 \leq q_1 \leq q_2 \leq L = |\mathcal{S}|$ and and a set $S \subseteq \Omega_{q_2}$,*

$$\sum_{S' \in \Omega_{q_1} \cap 2^S} \rho(S') = \tbinom{q_2-1}{q_1-1} \rho(S).$$

Lemma 2 (Sandıkçı et al. (2013), Lemma 2)**.** *Given integers q_1, q_2 and q_3 with $1 \leq q_1 \leq q_2 \leq q_3 \leq L = |\mathcal{S}|$, a set $S \subseteq \Omega_{q_3}$, and a function $v(\cdot) : \Omega_{q_1} \cap 2^S \to \mathbb{R}$,*

$$\sum_{S' \in \Omega_{q_2} \cap 2^S} \sum_{S'' \in \Omega_{q_1} \cap 2^{S'}} \rho(S'')v(S'') = \begin{pmatrix} q_3 - q_1 \\ q_2 - q_1 \end{pmatrix} \sum_{\hat{S} \in \Omega_{q_1} \cap 2^S} \rho(\hat{S})v(\hat{S}).$$

Proof of the first two lemmas from Sandıkçı et al. (2013) follow in a straightforward way from counting arguments. For example, if $\gamma(s)$ denotes any quantity dependent on scenario $s \in \mathcal{S}$, and $\mathcal{C}$ is any collection of subsets of $\mathcal{D} \subseteq \mathcal{S}$, then

$$\sum_{S \subseteq \mathcal{C}} \sum_{s \in S} \gamma(s) = \sum_{s \in \mathcal{D}} \eta(\mathcal{C}, s)\gamma(s),$$

where $\eta(\mathcal{C}, s)$ is the number of sets in the collection $\mathcal{C}$ that contain s. In the case that $\mathcal{C} = \Omega_q$ and $\mathcal{D} = \mathcal{S}$, then $\eta(\mathcal{C}, s) = \binom{L-1}{q-1}$ for all s, since for each s, this is the number of ways that the remaining $q-1$ scenarios needed to form a subset of size q containing s can be chosen from the L scenarios other than s. From such arguments and from definitions the following results hold. Note that 2^S denotes the power set of S, i.e., the set of all subsets of S.

Corollary 1 (Sandıkçı et al. (2013), Corollary 1)**.** *Given integers q_1, q_2 with $1 \leq q_1 \leq q_2 \leq L = |\mathcal{S}|$, a set $S \subseteq \Omega_{q_2}$, and a function $v(\cdot) : \Omega_1 \cap 2^S \to \mathbb{R}$,*

$$\sum_{S' \in \Omega_{q_1} \cap 2^S} \sum_{s \in S'} p_s v(\{s\}) = \begin{pmatrix} q_2 - 1 \\ q_1 - 1 \end{pmatrix} \sum_{s \in S} p_s v(\{s\}).$$

The following result requires the observation that if $((x_t^s)_{t=1}^T)_{s \in S}$ is feasible for $GR(S)$, where $S \subseteq \mathcal{S}$, then $((x_t^s)_{t=1}^T)_{s \in S'}$ is feasible for $GR(S')$ for any proper subset $S' \subseteq S$. This is obviously true in the multistage case by inspection of the definition of the group subproblem.

Lemma 3 (Sandıkçı et al. (2013), Lemma 3)**.** *Given an integer q with $1 \leq q \leq L = |\mathcal{S}|$ and a set $S \subseteq \Omega_q$,*

$$(q-1)\rho(S)v_{GR}(S) \geq \sum_{\hat{S} \in \Omega_{q-1} \cap 2^S} \rho(\hat{S})v_{GR}(\hat{S}).$$

Proof. Let $((\tilde{x}_t^s)_{t=1}^T)s \in S$ denote an optimal solution to $GR(S)$. Then it is obvious from the group subproblem definition that, for any $S' \in \Omega_{q-1} \cap 2^S$, $((\tilde{x}_t^s)_{t=1}^T)s \in S'$ is feasible for $GR(S')$ and thus

$$\frac{1}{\rho(S')} \sum_{s \in S'} p_s \sum_{t=1}^{T} F_t^s(\tilde{x}_t^s) \geq z_{GR}(S').$$

Multiplying both sides by $\rho(S')$ and summing over all $S' \in \Omega_{q-1} \cap 2^S$, we obtain

$$\sum_{S' \in \Omega_{q-1} \cap 2^S} \sum_{s \in S'} p_s \sum_{t=1}^{T} F_t^s(\tilde{x}_t^s) \geq \sum_{S' \in \Omega_{q-1} \cap 2^S} \rho(S') z_{GR}(S').$$

Applying Corollary 1 to the left-hand side of this inequality, with $q_2 = q$, $q_1 = q-1$, and $v(\{s\}) = \sum_{t=1}^{T} F_t^s(\tilde{x}_t^s)$, we obtain

$$(q-1) \sum_{s \in S} p_s \sum_{t=1}^{T} F_t^s(\tilde{x}_t^s) = (q-1)\rho(S) v_{GR}(S) \geq \sum_{S' \in \Omega_{q-1} \cap 2^S} \rho(S') z_{GR}(S'),$$

as required. □

Proof. ***[of Theorem 1.]*** $z_{WS} = EGSO(1)$ and $EGSO(L) = z_{MSO}$ follow immediately from definitions and earlier observations. Now consider any integer q with $1 \leq q \leq L-1$. For any $S \in \Omega_{q+1}$, we have, from Lemma 3, that

$$q\rho(S) v_{GR}(S) \geq \sum_{\hat{S} \in \Omega_q \cap 2^S} \rho(\hat{S}) v_{GR}(\hat{S}).$$

Summing over all $S \in \Omega_{q+1}$ yields

$$q \sum_{S \in \Omega_{q+1}} \rho(S) v_{GR}(S) \geq \sum_{S \in \Omega_{q+1}} \sum_{\hat{S} \in \Omega_q \cap 2^S} \rho(\hat{S}) v_{GR}(\hat{S}),$$

which, by the definition of $EGSO(q+1)$, implies

$$q \binom{L-1}{q} EGSO(q+1) \geq \sum_{S \in \Omega_{q+1}} \sum_{\hat{S} \in \Omega_q \cap 2^S} \rho(\hat{S}) v_{GR}(\hat{S}).$$

Applying Lemma 2 with $q_3 = L$, $q_2 = q+1$ and $q_1 = q$ to the right-hand side of this inequality, we obtain

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

which, noting $L - q \geq 1$, is precisely $EGSO(q+1) \geq EGSO(q)$, as required. □

Appendix B Results of Computational Experiments

K^{max} partitions are solved for every instance, and every partition requires solving m group subproblems, where m equals total number of scenarios, L, divided by group size, q. (K^{max} is 30 for CAP and SSLP instances, and 50 for DCAP instances.) OPT denotes the optimal objective over all scenarios. Δ_k^P and Δ_k^{SAA} denote optimality gaps associated with the partition bound and the lower end of the 95% confidence interval around the SAA estimate, respectively, after k partitions are completed. The optimality gaps are calculated relative to the wait-and-see solution, $z_{WS} = \sum_{s \in \mathcal{S}} z_{GR}(\{s\})$.

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

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

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

Table 3: Partition bound vs. lower end of the 95% SAA confidence interval, CAP instances, $q = 5$.

				# Cont.	# Bin.	# Cons-	$k = 10$			$k = 20$			$k = 30$		
Instance	L	OPT	z_{WS}	Vars.	Vars.	traints	Δ_{10}^P	Δ_{10}^{SAA}	$\delta_{10}^{SAA,P}$	Δ_{20}^P	Δ_{20}^{SAA}	$\delta_{20}^{SAA,P}$	Δ_{30}^P	Δ_{30}^{SAA}	$\delta_{30}^{SAA,P}$
CAP 101	240	734,354	720,186	6,250	25	376	14.23%	49.14%	**71.05%**	14.23%	58.77%	**75.79%**	14.23%	44.26%	**67.86%**
CAP 102	240	788,961	769,943	6,250	25	376	18.30%	56.15%	**67.41%**	18.30%	35.97%	**49.13%**	18.30%	36.83%	**50.32%**
CAP 103	240	823,775	808,880	6,250	25	376	21.61%	73.22%	**70.48%**	21.61%	50.25%	**56.99%**	21.61%	43.59%	**50.42%**
CAP 104	240	841,151	831,881	6,250	25	376	5.77%	65.31%	**91.16%**	5.77%	46.72%	**87.65%**	5.65%	40.89%	**86.18%**
CAP 111	240	763,265	749,675	12,500	50	501	9.12%	42.35%	**78.46%**	8.97%	36.02%	**75.09%**	8.97%	28.06%	**68.03%**
CAP 112	240	840,190	815,795	12,500	50	501	19.61%	37.55%	**47.77%**	18.55%	33.21%	**44.15%**	17.33%	26.38%	**34.32%**
CAP 113	240	916,658	890,418	12,500	50	501	20.87%	38.89%	**46.32%**	19.89%	42.89%	**53.62%**	19.89%	33.54%	**40.69%**
CAP 114	240	1,024,707	1,003,174	12,500	50	501	17.17%	25.94%	**33.82%**	15.53%	23.37%	**33.56%**	15.53%	23.02%	**32.54%**
CAP 121	240	720,520	707,503	12,500	50	501	12.54%	41.20%	**69.57%**	11.39%	21.98%	**48.18%**	11.12%	17.83%	**37.61%**
CAP 122	240	777,722	761,099	12,500	50	501	16.28%	28.06%	**41.97%**	15.01%	24.51%	**38.76%**	15.01%	34.40%	**56.37%**
CAP 123	240	824,148	805,682	12,500	50	501	18.90%	43.72%	**56.77%**	17.31%	32.23%	**46.30%**	17.16%	32.03%	**46.43%**
CAP 124	240	859,846	838,223	12,500	50	501	16.03%	34.52%	**53.56%**	16.03%	39.01%	**58.91%**	16.03%	34.82%	**53.96%**
CAP 131	240	723,212	709,403	12,500	50	501	13.93%	67.40%	**79.33%**	13.93%	47.31%	**70.55%**	13.93%	38.62%	**63.93%**
CAP 132	240	778,244	763,527	12,500	50	501	17.46%	42.68%	**59.08%**	16.10%	29.32%	**45.10%**	16.10%	32.77%	**50.87%**
CAP 133	240	815,298	796,921	12,500	50	501	19.52%	36.75%	**46.87%**	19.52%	35.40%	**44.85%**	19.52%	34.70%	**43.75%**
CAP 134	240	849,237	836,685	12,500	50	501	7.52%	28.35%	**73.48%**	7.52%	7.60%	**1.12%**	7.52%	15.81%	**52.46%**
Mean							15.55%	44.45%	61.70%	14.98%	35.29%	51.86%	14.87%	32.35%	52.23%
Std. dev.							4.75%	14.32%	15.86%	4.56%	12.58%	20.18%	4.53%	8.34%	14.08%

Note. 240-scenario instances of CAP 101-104 have 300,000 continuous variables, 25 binary variables, and 18,001 constraints; and 240-scenario instances of CAP 111-134 have 600,000 continuous variables, 50 binary variables, and 24,001 constraints.

Table 4: Partition bound vs. lower end of the 95% SAA confidence interval, CAP instances, $q = 10$.

				# Cont.	# Bin.	# Cons-	$k = 10$			$k = 20$			$k = 30$		
Instance	L	OPT	z_{WS}	Vars.	Vars.	traints	Δ_{10}^{P}	Δ_{10}^{SAA}	$\delta_{10}^{SAA,P}$	Δ_{20}^{P}	Δ_{20}^{SAA}	$\delta_{20}^{SAA,P}$	Δ_{30}^{P}	Δ_{30}^{SAA}	$\delta_{30}^{SAA,P}$
CAP 101	240	734,354	720,186	12,500	25	751	7.07%	15.54%	**54.48%**	6.98%	17.35%	**59.78%**	6.98%	19.68%	**64.55%**
CAP 102	240	788,961	769,943	12,500	25	751	8.06%	37.20%	**78.34%**	8.06%	24.51%	**67.12%**	8.06%	20.36%	**60.42%**
CAP 103	240	823,775	808,880	12,500	25	751	12.72%	32.83%	**61.27%**	12.72%	29.87%	**57.42%**	12.72%	27.21%	**53.27%**
CAP 104	240	841,151	831,881	12,500	25	751	0.00%	60.04%	**100.00%**	0.00%	34.36%	**100.00%**	0.00%	31.64%	**100.00%**
CAP 111	240	763,265	749,675	25,000	50	1,001	3.45%	17.56%	**80.35%**	3.06%	25.93%	**88.19%**	3.06%	17.21%	**82.20%**
CAP 112	240	840,190	815,795	25,000	50	1,001	7.65%	18.80%	**59.30%**	7.65%	26.42%	**71.04%**	7.65%	27.86%	**72.53%**
CAP 113	240	916,658	890,418	25,000	50	1,001	10.83%	34.05%	**68.20%**	10.51%	26.79%	**60.78%**	10.51%	26.63%	**60.54%**
CAP 114	240	1,024,707	1,003,174	25,000	50	1,001	6.68%	50.18%	**86.69%**	6.53%	38.80%	**83.17%**	6.45%	34.01%	**81.03%**
CAP 121	240	720,520	707,503	25,000	50	1,001	4.08%	39.59%	**89.70%**	4.08%	33.67%	**87.89%**	4.08%	27.16%	**84.99%**
CAP 122	240	777,722	761,099	25,000	50	1,001	6.06%	40.30%	**84.97%**	6.06%	29.98%	**79.80%**	5.95%	30.65%	**80.58%**
CAP 123	240	824,148	805,682	25,000	50	1,001	7.82%	30.77%	**74.60%**	6.98%	23.32%	**70.09%**	6.98%	16.49%	**57.70%**
CAP 124	240	859,846	838,223	25,000	50	1,001	8.82%	44.03%	**79.96%**	8.15%	28.68%	**71.58%**	7.82%	24.32%	**67.85%**
CAP 131	240	723,212	709,403	25,000	50	1,001	5.34%	25.47%	**79.05%**	5.34%	26.96%	**80.21%**	5.27%	28.87%	**81.73%**
CAP 132	240	778,244	763,527	25,000	50	1,001	7.36%	21.64%	**65.98%**	7.36%	18.73%	**60.70%**	7.12%	12.85%	**44.58%**
CAP 133	240	815,298	796,921	25,000	50	1,001	11.03%	29.69%	**62.86%**	10.05%	23.92%	**58.01%**	10.05%	18.72%	**46.35%**
CAP 134	240	849,237	836,685	25,000	50	1,001	0.58%	71.30%	**99.18%**	0.58%	62.12%	**99.06%**	0.58%	47.67%	**98.78%**
Mean							6.72%	35.56%	76.56%	6.51%	29.46%	74.68%	6.45%	25.71%	71.07%
Std. dev.							3.49%	15.43%	13.75%	3.38%	10.28%	14.16%	3.37%	8.44%	16.96%

Table 5: Partition bound vs. lower end of the 95% SAA confidence interval, CAP instances, $q = 15$.

				# Cont.	# Bin.	# Cons-	$k = 10$			$k = 20$			$k = 30$		
Instance	L	OPT	z_{WS}	Vars.	Vars.	traints	Δ_{10}^{P}	Δ_{10}^{SAA}	$\delta_{10}^{SAA,P}$	Δ_{20}^{P}	Δ_{20}^{SAA}	$\delta_{20}^{SAA,P}$	Δ_{30}^{P}	Δ_{30}^{SAA}	$\delta_{30}^{SAA,P}$
CAP 101	240	734,354	720,186	18,750	25	1,126	4.41%	28.97%	**84.76%**	4.41%	30.28%	**85.42%**	3.91%	26.99%	**85.51%**
CAP 102	240	788,961	769,943	18,750	25	1,126	3.74%	32.03%	**88.32%**	3.74%	15.64%	**76.09%**	3.74%	14.89%	**74.89%**
CAP 103	240	823,775	808,880	18,750	25	1,126	9.33%	21.34%	**56.29%**	9.33%	26.76%	**65.14%**	9.33%	17.13%	**45.54%**
CAP 104	240	841,151	831,881	18,750	25	1,126	0.00%	45.52%	**100.00%**	0.00%	25.66%	**100.00%**	0.00%	20.04%	**100.00%**
CAP 111	240	763,265	749,675	37,500	50	1,501	1.39%	45.33%	**96.93%**	1.34%	32.17%	**95.82%**	1.34%	30.42%	**95.58%**
CAP 112	240	840,190	815,795	37,500	50	1,501	5.36%	35.52%	**84.92%**	5.36%	31.96%	**83.24%**	5.36%	27.57%	**80.57%**
CAP 113	240	916,658	890,418	37,500	50	1,501	6.77%	44.97%	**84.95%**	6.77%	26.51%	**74.46%**	6.12%	19.05%	**67.88%**
CAP 114	240	1,024,707	1,003,174	37,500	50	1,501	2.70%	23.50%	**88.52%**	2.65%	18.51%	**85.68%**	2.65%	16.43%	**83.86%**
CAP 121	240	720,520	707,503	37,500	50	1,501	2.14%	14.28%	**85.03%**	2.14%	-0.85%	**N/A**	2.14%	7.71%	**72.27%**
CAP 122	240	777,722	761,099	37,500	50	1,501	2.50%	28.79%	**91.33%**	2.50%	19.34%	**87.09%**	2.50%	16.10%	**84.49%**
CAP 123	240	824,148	805,682	37,500	50	1,501	2.88%	29.87%	**90.35%**	2.88%	19.21%	**85.00%**	2.88%	21.39%	**86.53%**
CAP 124	240	859,846	838,223	37,500	50	1,501	5.30%	25.78%	**79.44%**	5.30%	20.53%	**74.19%**	4.66%	17.38%	**73.19%**
CAP 131	240	723,212	709,403	37,500	50	1,501	2.94%	57.95%	**94.92%**	2.94%	51.73%	**94.31%**	2.85%	38.91%	**92.68%**
CAP 132	240	778,244	763,527	37,500	50	1,501	4.54%	56.09%	**91.91%**	4.14%	22.39%	**81.51%**	3.87%	16.78%	**76.96%**
CAP 133	240	815,298	796,921	37,500	50	1,501	7.04%	26.77%	**73.70%**	5.18%	22.55%	**77.02%**	5.18%	14.26%	**63.66%**
CAP 134	240	849,237	836,685	37,500	50	1,501	0.15%	5.32%	**97.27%**	0.15%	-2.23%	**N/A**	0.00%	1.73%	**100.00%**
Mean							3.82%	32.63%	86.79%	3.68%	22.51%	83.21%	3.53%	19.17%	80.23%
Std. dev.							2.53%	14.35%	10.61%	2.41%	12.61%	9.49%	2.34%	8.80%	14.27%

Table 6: Partition bound vs. lower end of the 95% SAA confidence interval, CAP instances, $q = 20$.

				# Cont.	# Bin.	# Cons-	$k = 10$			$k = 20$			$k = 30$		
Instance	L	OPT	z_{WS}	Vars.	Vars.	traints	Δ_{10}^{P}	Δ_{10}^{SAA}	$\delta_{10}^{SAA,P}$	Δ_{20}^{P}	Δ_{20}^{SAA}	$\delta_{20}^{SAA,P}$	Δ_{30}^{P}	Δ_{30}^{SAA}	$\delta_{30}^{SAA,P}$
CAP 101	240	734,354	720,186	25,000	25	1,501	3.98%	36.21%	**89.00%**	3.46%	15.63%	**77.84%**	3.28%	12.60%	**73.95%**
CAP 102	240	788,961	769,943	25,000	25	1,501	3.31%	29.87%	**88.92%**	3.28%	12.86%	**74.48%**	3.20%	9.96%	**67.84%**
CAP 103	240	823,775	808,880	25,000	25	1,501	7.88%	40.66%	**80.61%**	7.88%	18.96%	**58.42%**	6.90%	15.53%	**55.59%**
CAP 104	240	841,151	831,881	25,000	25	1,501	0.00%	69.85%	**100.00%**	0.00%	40.50%	**100.00%**	0.00%	27.85%	**100.00%**
CAP 111	240	763,265	749,675	50,000	50	2,001	1.06%	35.61%	**97.03%**	0.93%	20.45%	**95.44%**	0.26%	20.70%	**98.72%**
CAP 112	240	840,190	815,795	50,000	50	2,001	4.86%	43.12%	**88.74%**	4.86%	25.16%	**80.70%**	4.86%	16.29%	**70.19%**
CAP 113	240	916,658	890,418	50,000	50	2,001	3.69%	25.64%	**85.59%**	3.69%	11.75%	**68.56%**	3.69%	21.29%	**82.65%**
CAP 114	240	1,024,707	1,003,174	50,000	50	2,001	1.07%	26.31%	**95.95%**	1.07%	20.55%	**94.81%**	1.07%	16.13%	**93.39%**
CAP 121	240	720,520	707,503	50,000	50	2,001	1.64%	55.38%	**97.04%**	1.64%	38.48%	**95.74%**	1.64%	30.19%	**94.56%**
CAP 122	240	777,722	761,099	50,000	50	2,001	1.46%	10.72%	**86.42%**	1.46%	3.64%	**59.99%**	1.46%	5.68%	**74.38%**
CAP 123	240	824,148	805,682	50,000	50	2,001	1.51%	22.25%	**93.23%**	1.51%	14.07%	**89.30%**	1.51%	13.05%	**88.45%**
CAP 124	240	859,846	838,223	50,000	50	2,001	4.12%	30.08%	**86.31%**	3.43%	24.96%	**86.25%**	3.43%	24.39%	**85.93%**
CAP 131	240	723,212	709,403	50,000	50	2,001	1.45%	60.04%	**97.58%**	1.45%	30.46%	**95.23%**	1.45%	25.41%	**94.28%**
CAP 132	240	778,244	763,527	50,000	50	2,001	2.28%	36.82%	**93.82%**	2.28%	32.14%	**92.92%**	2.28%	27.70%	**91.78%**
CAP 133	240	815,298	796,921	50,000	50	2,001	3.39%	22.59%	**85.01%**	3.39%	20.72%	**83.66%**	3.39%	18.15%	**81.34%**
CAP 134	240	849,237	836,685	50,000	50	2,001	0.00%	4.03%	**100.00%**	0.00%	-3.91%	**N/A**	0.00%	-10.72%	**N/A**
Mean							2.61%	34.32%	91.58%	2.52%	20.40%	83.55%	2.40%	17.14%	83.54%
Std. dev.							2.04%	17.19%	6.00%	2.00%	11.78%	13.33%	1.87%	10.16%	12.88%

Table 7: Best partition bound vs. lower limit of the 95% SAA confidence interval on SSLP instances.

Instance	L	OPT	z_{WS}	q	# Cont. Var.	# Bin. Var.	# Cons-traints	$k=10$ Δ_{10}^{P}	$k=10$ Δ_{10}^{SAA}	$k=10$ $\delta_{10}^{SAA,P}$	$k=20$ Δ_{20}^{P}	$k=20$ Δ_{20}^{SAA}	$k=20$ $\delta_{20}^{SAA,P}$	$k=30$ Δ_{30}^{P}	$k=30$ Δ_{30}^{SAA}	$k=30$ $\delta_{30}^{SAA,P}$
SSLP 5-25-50	50	-121.6	-134.3	2	50	255	65	48.82%	106.99%	**54.37%**	41.60%	80.94%	**48.60%**	41.60%	73.36%	**43.29%**
				5	125	630	155	15.38%	28.86%	**46.69%**	14.60%	18.54%	**21.27%**	11.62%	19.97%	**41.83%**
				10	250	1255	305	1.26%	73.90%	**98.30%**	0.00%	38.38%	**100.00%**	0.00%	30.50%	**100.00%**
				25	625	3130	755	0.00%	52.13%	**100.00%**	0.00%	29.61%	**100.00%**	0.00%	21.19%	**100.00%**
SSLP 10-50-100	100	-354.2	-371.4	2	100	1010	130	46.74%	80.53%	**41.95%**	45.29%	74.21%	**38.97%**	45.29%	67.90%	**33.30%**
				5	250	2510	310	13.49%	38.40%	**64.87%**	9.01%	26.56%	**66.07%**	9.01%	25.51%	**64.67%**

Note. The full SSLP 5-25-50 instance have 1250 continuous variables, 6255 binary variables, and 1505 constraints; and the full SSLP 10-50-100 instance have 5000 continuous variables, 50,010 binary variables, and 6010 constraints.

Table 8: Best partition bound vs. lower limit of the 95% SAA confidence interval on DCAP 332 instances.

Instance	L	OPT	z_{WS}	q	# Cont. Var.	# Bin. Var.	Partition Selection	# Cons-traints	$k=10$ Δ_{10}^{P}	$k=10$ Δ_{10}^{SAA}	$k=10$ $\delta_{10}^{SAA,P}$	$k=20$ Δ_{20}^{P}	$k=20$ Δ_{20}^{SAA}	$k=20$ $\delta_{20}^{SAA,P}$	$k=30$ Δ_{30}^{P}	$k=30$ Δ_{30}^{SAA}	$k=30$ $\delta_{30}^{SAA,P}$
DCAP 332 1x10x20	200	918.69	850.83	2	24	60	Tree	78.0	63.73%		**51.32%**	63.46%		**45.28%**	63.42%		**40.75%**
							Random	59.8	67.31%	130.91%	**48.58%**	59.27%	115.97%	**48.89%**	57.93%	107.03%	**45.87%**
							Misaligned	57.0	74.44%		**43.13%**	73.71%		**36.44%**	73.68%		**31.16%**
				5	60	150	Tree	240.0	25.34%		**76.04%**	25.10%		**75.31%**	25.05%		**71.85%**
							Random	174.3	62.77%	105.77%	**40.66%**	61.86%	101.66%	**39.16%**	61.86%	89.00%	**30.50%**
							Misaligned	156.0	70.04%		**33.78%**	70.04%		**31.11%**	70.02%		**21.32%**
				10	120	300	Tree	510.0	5.06%		**94.43%**	4.99%		**95.06%**	4.96%		**94.14%**
							Random	392.8	51.17%	90.97%	**43.75%**	51.17%	101.09%	**49.38%**	51.17%	84.57%	**39.50%**
							Misaligned	321.0	69.94%		**23.12%**	69.94%		**30.82%**	69.93%		**17.31%**
				20	240	600	Tree	1050.0	1.25%		**98.36%**	1.25%		**98.12%**	1.25%		**97.89%**
							Random	884.7	41.86%	76.70%	**45.43%**	40.05%	66.81%	**40.05%**	40.05%	59.46%	**32.63%**
							Misaligned	861.0	48.86%		**36.29%**	47.89%		**28.32%**	47.89%		**19.46%**
				40	480	1200	Tree	2109.0	0.06%		**99.94%**	0.05%		**99.94%**	0.05%		**99.93%**
							Random	1943.1	24.02%	96.20%	**75.03%**	23.39%	82.44%	**71.63%**	23.39%	69.54%	**66.37%**
							Misaligned	1941.0	30.25%		**68.55%**	27.40%		**66.76%**	27.40%		**60.59%**
DCAP 332 1x15x20	300	1212.54	1130.14	2	24	60	Tree	78.0	59.65%		**53.99%**	59.45%		**47.07%**	59.41%		**39.98%**
							Random	59.6	78.04%	129.65%	**39.80%**	68.83%	112.34%	**38.73%**	64.29%	98.99%	**35.05%**
							Misaligned	57.4	86.04%		**33.63%**	84.17%		**25.08%**	82.78%		**16.38%**
				5	60	150	Tree	240.0	19.56%		**73.07%**	16.75%		**79.01%**	16.65%		**79.29%**
							Random	168.7	67.48%	72.65%	**7.12%**	67.48%	79.77%	**15.41%**	67.27%	80.38%	**16.31%**
							Misaligned	156.0	73.46%		**N/A**	73.46%		**7.91%**	73.06%		**9.10%**
				10	120	300	Tree	510.0	3.42%		**95.96%**	3.34%		**94.95%**	3.32%		**94.29%**
							Random	371.5	52.20%	84.61%	**38.31%**	52.20%	66.09%	**21.02%**	52.20%	58.14%	**10.23%**
							Misaligned	332.8	60.13%		**28.93%**	60.09%		**9.08%**	59.87%		**N/A**
				20	240	600	Tree	1050.0	2.11%		**96.96%**	2.11%		**96.55%**	2.11%		**96.27%**
							Random	832.4	41.82%	69.51%	**39.84%**	41.82%	61.19%	**31.66%**	41.82%	56.59%	**26.10%**
							Misaligned	767.4	50.07%		**27.97%**	50.07%		**18.18%**	50.07%		**11.53%**
				30	360	900	Tree	1569.0	0.48%		**99.38%**	0.44%		**99.22%**	0.43%		**99.23%**
							Random	1333.0	34.45%	78.45%	**56.09%**	34.20%	55.81%	**38.73%**	34.01%	55.87%	**39.13%**
							Misaligned	1296.0	41.53%		**47.07%**	41.53%		**25.60%**	41.53%		**25.68%**
DCAP 332 1x25x20	500	1691.62	1646.88	2	24	60	Tree	78.0	61.05%		**48.60%**	60.49%		**44.27%**	60.42%		**45.45%**
							Random	58.9	78.66%	118.79%	**33.79%**	72.05%	108.55%	**33.62%**	68.64%	110.77%	**38.03%**
							Misaligned	57.1	82.33%		**30.69%**	81.08%		**25.30%**	80.48%		**27.34%**
				5	60	150	Tree	240.0	25.90%		**79.95%**	24.14%		**73.65%**	23.89%		**69.39%**
							Random	163.7	68.89%	129.13%	**46.65%**	68.89%	91.61%	**24.80%**	68.89%	78.05%	**11.73%**
							Misaligned	156.0	72.05%		**44.20%**	72.05%		**21.35%**	71.69%		**8.15%**
				10	120	300	Tree	510.0	10.89%		**90.63%**	10.73%		**89.08%**	10.67%		**87.67%**
							Random	353.8	55.77%	116.22%	**52.01%**	55.29%	98.26%	**43.73%**	54.42%	86.56%	**37.14%**
							Misaligned	325.2	59.10%		**49.15%**	59.10%		**39.86%**	59.10%		**31.73%**
				20	240	600	Tree	1050.0	6.53%		**79.92%**	6.53%		**89.73%**	6.53%		**90.10%**
							Random	775.1	45.27%	32.51%	**N/A**	45.17%	63.59%	**28.96%**	44.89%	65.96%	**31.94%**
							Misaligned	693.5	52.95%		**N/A**	52.74%		**17.07%**	52.74%		**20.05%**
				50	600	1500	Tree	2628.0	2.36%		**97.00%**	2.22%		**95.88%**	2.21%		**95.85%**
							Random	2226.5	32.53%	78.62%	**58.62%**	31.51%	53.90%	**41.54%**	31.43%	53.16%	**40.88%**
							Misaligned	2166.0	38.24%		**51.36%**	36.79%		**31.75%**	36.79%		**30.79%**

Note. The full DCAP 332 1x10x20 instance has 2,400 continuous variables, 6,000 binary variables, and 10,581 constraints; the full DCAP 332 1x15x20 instance has 3,600 continuous variables, 9,000 binary variables, and 15,876 constraints; and the full DCAP 332 1x25x20 instance has 6,000 continuous variables, 15,000 binary variables, and 26,466 constraints.

Table 9: Best partition bound vs. lower limit of the 95% SAA confidence interval on DCAP 342 instances.

Instance	L	OPT	z_{WS}	q	# Cont. Var.	# Bin. Var.	Partition Selection	# Constraints	$k=10$ Δ_{10}^{P}	$k=10$ Δ_{10}^{SAA}	$k=10$ $\delta_{10}^{SAA,P}$	$k=20$ Δ_{20}^{P}	$k=20$ Δ_{20}^{SAA}	$k=20$ $\delta_{20}^{SAA,P}$	$k=30$ Δ_{30}^{P}	$k=30$ Δ_{30}^{SAA}	$k=30$ $\delta_{30}^{SAA,P}$
DCAP 342 1x10x20	200	1817.92	1780.54	2	24	76	Tree	86.0	39.31%		**81.16%**	38.25%		**77.40%**	38.21%		**71.40%**
							Random	65.2	65.93%	208.63%	**68.40%**	50.05%	169.22%	**70.43%**	42.53%	133.61%	**68.17%**
							Misaligned	61.0	85.26%		**59.13%**	85.02%		**49.76%**	84.93%		**36.43%**
				5	60	190	Tree	266.0	6.24%		**95.19%**	3.87%		**97.58%**	3.85%		**96.94%**
							Random	187.8	70.81%	129.79%	**45.44%**	68.39%	159.91%	**57.23%**	68.39%	125.94%	**45.70%**
							Misaligned	166.0	84.10%		**35.20%**	83.80%		**47.59%**	83.80%		**33.46%**
				10	120	380	Tree	566.0	1.42%		**98.78%**	1.26%		**98.69%**	1.20%		**98.65%**
							Random	426.1	51.54%	116.45%	**55.74%**	51.54%	96.67%	**46.69%**	51.54%	88.84%	**41.99%**
							Misaligned	341.0	83.54%		**28.26%**	83.51%		**13.62%**	83.49%		**6.03%**
				20	240	760	Tree	1166.0	0.29%		**99.84%**	0.29%		**99.70%**	0.29%		**99.66%**
							Random	967.6	36.60%	183.77%	**80.08%**	35.60%	95.09%	**62.56%**	35.60%	84.30%	**57.77%**
							Misaligned	941.0	46.89%		**74.48%**	45.47%		**52.18%**	45.47%		**46.06%**
				40	480	1520	Tree	2341.0	0.24%		**99.69%**	0.24%		**99.68%**	0.24%		**99.59%**
							Random	2143.3	18.37%	78.42%	**76.58%**	16.24%	76.95%	**78.89%**	16.24%	59.67%	**72.78%**
							Misaligned	2141.0	19.96%		**74.55%**	13.21%		**82.83%**	13.21%		**77.85%**
DCAP 342 1x15x20	300	1897.17	1852.91	2	24	76	Tree	86.0	42.49%		**71.92%**	41.95%		**72.80%**	41.85%		**71.00%**
							Random	64.0	67.14%	151.28%	**55.62%**	53.67%	154.19%	**65.19%**	49.18%	144.34%	**65.93%**
							Misaligned	61.3	77.51%		**48.76%**	76.56%		**50.35%**	74.93%		**48.08%**
				5	60	190	Tree	266.0	11.73%		**89.44%**	11.51%		**87.48%**	11.40%		**87.93%**
							Random	181.3	67.93%	111.04%	**38.83%**	67.93%	91.94%	**26.12%**	67.93%	94.45%	**28.08%**
							Misaligned	166.0	77.20%		**30.48%**	77.17%		**16.06%**	77.17%		**18.30%**
				10	120	380	Tree	566.0	2.98%		**98.14%**	2.85%		**97.32%**	2.83%		**97.06%**
							Random	402.7	59.76%	160.18%	**62.69%**	59.76%	106.20%	**43.73%**	58.23%	96.00%	**39.34%**
							Misaligned	354.8	73.23%		**54.29%**	72.57%		**31.66%**	72.40%		**24.58%**
				20	240	760	Tree	1166.0	1.76%		**98.66%**	1.76%		**97.84%**	1.76%		**98.02%**
							Random	906.8	43.67%	131.29%	**66.74%**	43.67%	81.79%	**46.61%**	43.67%	88.81%	**50.83%**
							Misaligned	830.5	59.77%		**54.47%**	58.61%		**28.34%**	58.61%		**34.01%**
				30	360	1140	Tree	1741.0	1.14%		**98.84%**	1.13%		**98.76%**	1.11%		**98.66%**
							Random	1458.2	38.17%	98.36%	**61.19%**	36.91%	91.27%	**59.55%**	36.80%	82.95%	**55.63%**
							Misaligned	1416.0	47.07%		**52.14%**	44.36%		**51.40%**	44.36%		**46.52%**
DCAP 342 1x25x20	500	1675.73	1636.25	2	24	76	Tree	86.0	47.77%		**56.85%**	47.28%		**57.89%**	47.21%		**57.56%**
							Random	63.2	70.84%	110.69%	**36.00%**	64.19%	112.27%	**42.82%**	57.18%	111.23%	**48.59%**
							Misaligned	61.2	80.10%		**27.63%**	79.01%		**29.63%**	77.94%		**29.93%**
				5	60	190	Tree	266.0	11.37%		**90.56%**	9.90%		**90.99%**	9.46%		**89.97%**
							Random	174.9	74.46%	120.48%	**38.20%**	74.07%	109.92%	**32.62%**	74.07%	94.30%	**21.46%**
							Misaligned	166.0	79.93%		**33.66%**	79.93%		**27.28%**	79.93%		**15.24%**
				10	120	380	Tree	566.0	1.95%		**98.28%**	1.84%		**98.20%**	1.79%		**98.04%**
							Random	379.4	68.19%	113.02%	**39.66%**	68.00%	102.34%	**33.55%**	67.82%	90.97%	**25.45%**
							Misaligned	346.2	77.93%		**31.05%**	77.44%		**24.33%**	77.44%		**14.88%**
				20	240	760	Tree	1166.0	0.56%		**99.59%**	0.56%		**99.53%**	0.56%		**99.38%**
							Random	837.1	58.22%	137.77%	**57.74%**	57.25%	118.77%	**51.80%**	57.25%	90.66%	**36.85%**
							Misaligned	741.5	71.65%		**48.00%**	71.40%		**39.89%**	71.40%		**21.25%**
				50	600	1900	Tree	2916.0	0.37%		**99.65%**	0.31%		**99.54%**	0.31%		**99.54%**
							Random	2438.9	39.31%	104.49%	**62.38%**	37.04%	68.00%	**45.53%**	37.04%	66.90%	**44.63%**
							Misaligned	2366.0	48.16%		**53.92%**	47.22%		**30.57%**	46.42%		**30.62%**

Note. The full DCAP 342 1x10x20 instance has 2,400 continuous variables, 7,600 binary variables, and 11,741 constraints; the full DCAP 342 1x15x20 instance has 3,600 continuous variables, 11,400 binary variables, and 17,616 constraints; and the full DCAP 342 1x25x20 instance has 6,000 continuous variables, 19,000 binary variables, and 29,366 constraints.

References

Ahmed, S. (2010). Two stage stochastic integer programming: a brief introduction. *Wiley Encyclopedia of Operations Research and Management Science*.

Ahmed, S. (2013). A scenario decomposition algorithm for 0–1 stochastic programs. *Operations Research Letters*, 41(6):565–569.

Ahmed, S. and Garcia, R. (2003). Dynamic capacity acquisition and assignment under uncertainty. *Annals of Operations Research*, 124(1-4):267–283.

Ahmed, S., Garcia, R., Kong, N., Ntaimo, L., Parija, G., Qui, F., and Sen, S. (2015). Siplib: A stochastic integer programming test problem library. `http://www.isye.gatech.edu/~sahmed/siplib`. Accessed: 2015-12-21.

Ahmed, S. and Shapiro, A. (2002). A sample average approximation algorithm for stochastic programs with integer recourse. Technical report, ISyE, Georgia Tech.

Aldasoro, U., Escudero, L. F., Merino, M., and Pérez, G. (2013). An algorithmic framework for solving large-scale multistage stochastic mixed 0–1 problems with nonsymmetric scenario trees. part ii: Parallelization. *Computers & Operations Research*, 40(12):2950–2960.

Arda, Y., Crama, Y., Kronus, D., Pironet, T., and Van Hentenryck, P. (2014). Multi-period vehicle loading with stochastic release dates. *EURO Journal on Transportation and Logistics*, 3(2):93–119.

Beier, E., Venkatachalam, S., Corolli, L., and Ntaimo, L. (2015). Stage-and scenario-wise fenchel decomposition for stochastic mixed 0-1 programs with special structure. *Computers & Operations Research*, 59:94–103.

Bertsimas, D. and Georghiou, A. (2015). Design of near optimal decision rules in multistage adaptive mixed-integer optimization. *Operations Research*.

Bertsimas, D., Goyal, V., and Sun, X. A. (2011). A geometric characterization of the power of finite adaptability in multistage stochastic and adaptive optimization. *Mathematics of Operations Research*, 36(1):24–54.

Birge, J. R. and Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

Bodur, M., Dash, S., Günlük, O., and Luedtke, J. (2014). Strengthened benders cuts for stochastic integer programs with continuous recourse. Technical report, Technical report. available as Optimization Online 2014-03-4263.

Bruno, S., Ahmed, S., Shapiro, A., and Street, A. (2015). Risk neutral and risk averse approaches to multistage renewable investment planning under uncertainty. *European Journal of Operational Research*.

Carøe, C. C. and Schultz, R. (1999). Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1):37–45.

Cheung, K., Gade, D., Silva-Monroy, C., Ryan, S. M., Watson, J.-P., Wets, R. J.-B., and Woodruff, D. L. (2015). Toward scalable stochastic unit commitment. *Energy Systems*, pages 1–22.

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

Erdogan, S. A., Gose, A., and Denton, B. T. (2015). On-line appointment sequencing and scheduling. *IIE Transactions*, (just-accepted):00–00.

Escudero, L. F., Garín, M. A., Merino, M., and 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. *Computers & Operations Research*, 37(4):738–753.

Escudero, L. F., Garín, M. A., Merino, M., and Pérez, G. (2012). An algorithmic framework for solving large-scale multistage stochastic mixed 0–1 problems with nonsymmetric scenario trees. *Computers & Operations Research*, 39(5):1133–1144.

Escudero, L. F., Garín, M. A., Pérez, G., and Unzueta, A. (2013). Scenario cluster decomposition of the lagrangian dual in two-stage stochastic mixed 0–1 optimization. *Computers & Operations Research*, 40(1):362–377.

Escudero, L. F., Garín, M. A., and Unzueta, A. (2016). Cluster lagrangean decomposition in multistage stochastic optimization. *Computers & Operations Research*, 67:48–62.

Guo, G., Hackebeil, G., Ryan, S. M., Watson, J.-P., and Woodruff, D. L. (2015). Integration of progressive hedging and dual decomposition in stochastic integer programs. *Operations Research Letters*, 43(3):311–316.

Gupta, A., Pál, M., Ravi, R., and Sinha, A. (2011). Sampling and cost-sharing: Approximation algorithms for stochastic optimization problems. *SIAM Journal on Computing*, 40(5):1361–1401.

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

Kim, K. and Zavala, V. M. (2015). Algorithmic innovations and software for the dual decomposition method applied to stochastic mixed-integer programs. *Optimization Online*.

Klein Haneveld, W. K., Stougie, L., and Van der Vlerk, M. H. (2006). Simple integer recourse models: convexity and convex approximations. *Mathematical programming*, 108(2-3):435–473.

Klein Haneveld, W. K. and van der Vlerk, M. H. (1999). Stochastic integer programming:general models and algorithms. *Annals of Operations Research*, 85(0):39–57.

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

Linderoth, J., Shapiro, A., and Wright, S. (2006). The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142(1):215–241.

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

Louveaux, F. (1986). Discrete stochastic location models. *Annals of Operations research*, 6(2):21–34.

Lubin, M., Martin, K., Petra, C. G., and Sandıkçı, B. (2013). On parallelizing dual decomposition in stochastic integer programming. *Operations Research Letters*, 41(3):252–258.

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

Maggioni, F., Potra, F. A., and Bertocchi, M. (2015). Stochastic versus robust optimization for a transportation problem. *Optimization Online*, 2015-03-4805.

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

Mete, H. O. and Zabinsky, Z. B. (2010). Stochastic optimization of medical supply location and distribution in disaster management. *International Journal of Production Economics*, 126(1):76–84.

Morton, D. P. and Wood, R. K. (1998). On a stochastic knapsack problem and generalizations. In Woodruff, D., editor, *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search: Interfaces in Computer Science and Operations Research*, pages 149–168. Kluwer Academic Publishers.

Norkin, V. I., Pflug, G. C., and Ruszczyński, A. (1998). A branch and bound method for stochastic global optimization. *Mathematical programming*, 83(1-3):425–450.

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

Ntaimo, L., Arrubla, J. A. G., Stripling, C., Young, J., and Spencer, T. (2012). A stochastic programming standard response model for wildfire initial attack planning. *Canadian Journal of Forest Research*, 42(6):987–1001.

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

Ntaimo, L. and Sen, S. (2008). A comparative study of decomposition algorithms for stochastic combinatorial optimization. *Computational Optimization and Applications*, 40(3):299–319.

Özaltin, O. Y., Prokopyev, O. A., Schaefer, A. J., and Roberts, M. S. (2011). Optimizing the societal benefits of the annual influenza vaccine: A stochastic programming approach. *Operations research*, 59(5):1131–1143.

Pérez, E., Ntaimo, L., Malavé, C. O., Bailey, C., and McCormack, P. (2013). Stochastic online appointment scheduling of multi-step sequential procedures in nuclear medicine. *Health care management science*, 16(4):281–299.

Plambeck, E. L., Fu, B.-R., Robinson, S. M., and Suri, R. (1996). Sample-path optimization of convex stochastic performance functions. *Mathematical Programming*, 75(2):137–176.

Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons.

Powell, W. B. (2014). Clearing the jungle of stochastic optimization. *Informs TutORials*.

Powell, W. B. and Topaloglu, H. (2003). Stochastic programming in transportation and logistics. *Handbooks in operations research and management science*, 10:555–635.

Ramazan, S. and Dimitrakopoulos, R. (2013). Production scheduling with uncertain supply: a new solution to the open pit mining problem. *Optimization and Engineering*, 14(2):361–380.

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

Romeijnders, W., Stougie, L., and van der Vlerk, M. H. (2014). Approximation in two-stage stochastic integer programming. In *Surveys in Operations Research and Management Science*, volume 19, pages 17–33. Elsevier.

Rubinstein, R. Y. and Shapiro, A. (1990). Optimization of static simulation models by the score function method. *Mathematics and Computers in Simulation*, 32(4):373–392.

Ruszczynski, A. P. and Shapiro, A. (2003). *Stochastic programming*, volume 10. Elsevier Amsterdam.

Ryan, K., Rajan, D., and Ahmed, S. (2015). Scenario decomposition for 0-1 stochastic programs: Improvements and asynchronous implementation. *Optimization Online*.

Sahinidis, N. V. (2004). Optimization under uncertainty: state-of-the-art and opportunities. *Computers & Chemical Engineering*, 28(6):971–983.

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

Sandıkçı, B. and Özaltin, O. Y. (2014). A scalable bounding method for multi-stage stochastic integer programs. *Optimization Online*.

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

Schultz, R. (1996). Rates of convergence in stochastic programs with complete integer recourse. *SIAM Journal on Optimization*, 6(4):1138–1152.

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

Schütz, P., Tomasgard, A., and Ahmed, S. (2009). Supply chain design under uncertainty using sample average approximation and dual decomposition. *European Journal of Operational Research*, 199(2):409–419.

Sen, S. and Higle, J. L. (2005). The c 3 theorem and a d 2 algorithm for large scale stochastic mixed-integer programming: set convexification. *Mathematical Programming*, 104(1):1–20.

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

Sen, S. and Zhou, Z. (2014). Multistage stochastic decomposition: a bridge between stochastic programming and approximate dynamic programming. *SIAM Journal on Optimization*, 24(1):127–153.

Shapiro, A. (2006). On complexity of multistage stochastic programs. *Operations Research Letters*, 34(1):1–8.

Shapiro, A., Dentcheva, D., et al. (2014). *Lectures on stochastic programming: modeling and theory*, volume 16. SIAM.

Shapiro, A. and Homem-de Mello, T. (2000). On the rate of convergence of optimal solutions of monte carlo approximations of stochastic programs. *SIAM journal on optimization*, 11(1):70–86.

Stoyan, S. J. and Dessouky, M. M. (2012). A stochastic mixed-integer programming approach to the energy-technology management problem. *Computers & Industrial Engineering*, 63(3):594–606.

Swamy, C. and Shmoys, D. B. (2012). Sampling-based approximation algorithms for multistage stochastic optimization. *SIAM Journal on Computing*, 41(4):975–1004.

Van der Vlerk, M. H. (2010). Convex approximations for a class of mixed-integer recourse models. *Annals of Operations Research*, 177(1):139–150.

Veliz, F. B., Watson, J.-P., Weintraub, A., Wets, R. J.-B., and Woodruff, D. L. (2014). Stochastic optimization models in forest planning: a progressive hedging solution approach. *Annals of Operations Research*, pages 1–16.

Verweij, B., Ahmed, S., Kleywegt, A. J., Nemhauser, G., and Shapiro, A. (2003). The sample average approximation method applied to stochastic routing problems: a computational study. *Computational Optimization and Applications*, 24(2-3):289–333.

Wallace, S. W. and Fleten, S.-E. (2003). Stochastic programming models in energy. *Handbooks in operations research and management science*, 10:637–677.

Watson, J.-P. and Woodruff, D. L. (2011). Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, 8(4):355–370.

Zenarosa, G. L., Prokopyev, O. A., and Schaefer, A. J. (2014a). M-smiplib: A multistage stochastic mixed-integer programming test set library. `http://www.cs.cmu.edu/~gzen/m-smiplib/`. Accessed: 2016-10-17.

Zenarosa, G. L., Prokopyev, O. A., and Schaefer, A. J. (2014b). Scenario-tree decomposition: bounds for multistage stochastic mixed-integer programs. *Optimization Online*.

Zhang, Z., Xie, X., and Geng, N. (2014). Dynamic surgery assignment of multiple operating rooms with planned surgeon arrival times. *Automation Science and Engineering, IEEE Transactions on*, 11(3):680–691.