A SCALABLE BOUNDING METHOD FOR MULTISTAGE
STOCHASTIC PROGRAMS*

BURHANEDDİN SANDIKÇI† AND OSPAN Y. ÖZALTIN‡

Abstract. Many dynamic decision problems involving uncertainty can be appropriately modeled
as multistage stochastic programs. However, most practical instances are so large and/or complex
that it is impossible to solve them on a single computer, especially due to memory limitations.
Extending the work of Sandıkçı et al. [39] on two-stage stochastic mixed-integer programs, this
paper considers general multistage stochastic programs and develops a bounding method based on
scenario decomposition. This method is broadly applicable, as it does not assume any problem
structure including convexity. Moreover, it naturally fits into a distributed computing environment.
Computational experiments with large-scale instances (with up to 100 million scenarios, about 1.5
billion decision variables—85% binary—and 800 million constraints) demonstrate that the proposed
method scales nicely with problem size and has immense potential to obtain high quality solutions
to practical instances within a reasonable time frame.

Key words. stochastic programming, multistage, mixed-integer, bounding, parallel computing

AMS subject classifications. 49M27, 68W10, 90C06, 90C11, 90C15, 90C25, 90C27, 90C59

1. Introduction. Many dynamic decision problems involving uncertainty can
be appropriately modeled as stochastic programs. Examples include problems in
energy [45], finance [23], manufacturing [3], telecommunication [16], transportation
and logistics [35], and health care [13, 34]. Despite their appeal as a flexible model-
ing framework, stochastic programs suffer exponential growth in their size with the
number of decision stages. Moreover, there is a lack of generally applicable efficient
solution algorithms for problems with integer decision variables. Therefore, practical
applications have generally been restricted to two-stage linear programs, truncating
or merging the problem data, and therefore, losing information that is otherwise
available. Two-stage programs, however, lack an ability to dynamically respond to
information that gradually becomes available to the decision maker. Furthermore, the
restriction to continuous variables hinders the applicability of stochastic programming
to a broad set of real life problems.

Several exact and approximation algorithms have been proposed in the literature
for solving stochastic programs (for reviews, see [5, 36, 41, 43]). Exact algorithms
typically rely on decomposition based methods for multistage models [1, 8, 27, 28,
46], however, they generally suffer from long running times, unless special problem
structure exists, and require discrete scenarios. To alleviate some of the difficulties
with exact algorithms, stochastic algorithms that return statistical bounds have been
developed. Examples include sample average approximation [25, 33] and stochastic
decomposition [21, 42]. In practice, however, using crude Monte-Carlo sampling in
stochastic algorithms has slow convergence, hence variance reduction techniques are
considered. Furthermore, there is no generally accepted approach for determining the
sample size to obtain good candidate solutions [22].

*Supported in part by National Science Foundation Grants CMMI-1435771 and CMMI-1436177.
An earlier version of this paper is available at SSRN: http://ssrn.com/abstract=2466650
†University of Chicago Booth School of Business (burhan@chicagobooth.edu). This author also
acknowledges the financial support from the University of Chicago Booth School of Business.
‡Edward P. Fitts Department of Industrial and Systems Engineering, North Carolina State Uni-
versity (oyozalti@ncsu.edu). This author also acknowledges support from Laboratory for Analytic
Sciences.
Bounding-based methods are proposed as an alternative way to address difficulties in solving stochastic programs. A common approach is to replace the true distribution of the random parameters by finitely supported approximate distributions. The most celebrated examples are Jensen’s inequality [24] for lower bounding and Edmundson-Madansky inequality [12, 29] for upper bounding, which have been extended in many dimensions [6, 10, 11, 15, 26].

The aforementioned bounds typically rely on some form of convexity assumption, and therefore, are not generally applicable. Our primary objective in this paper is to present a method for general multistage stochastic programs, which can generate tight bounds in a timely fashion through exploiting parallelism. In particular, we propose a scenario decomposition method that returns lower and upper bounds for the optimal objective value of the full problem. The proposed methodology is broadly applicable as it does not rely on restrictive assumptions such as convexity. Moreover, an inherent feature of this approach is its natural fit into a distributed computing environment, which makes it amenable to solving truly large-scale instances.

This paper extends our previous work on two-stage stochastic mixed-integer programs (SMIPs) [39] and builds on our unpublished work on multistage stochastic programs [40], which sparked a flurry of recent activity in the literature [7, 31, 32, 47]. Considering two-stage SMIPs with a discrete scenario set, Sandıkçi et al. [39] demonstrate how to generate bounds for the optimal objective value by solving smaller problems, called group subproblems, each formulated over a subset of scenarios of a given size $b$, and prove that these bounds monotonically improve with $b$. Formulating the group subproblems in [39] requires identifying a reference scenario to be included in each subproblem. These bounds are calculated by taking an expectation of group subproblem objective values over all subsets of size $b$, which may be computationally challenging for large scenario sets. In [40], we extend the bounds in [39] to multistage SMIPs and note that any partition of the scenario set is sufficient to produce legitimate bounds, which helps avoid the computational challenges encountered in [39].

Zenarosa et al. [47] also extend the bounds in [39] to multistage SMIPs by introducing a new class of group subproblems, called the cut-group subproblems, allowing for a set of multiple reference scenarios. Formulating a cut-group subproblem requires partitioning the scenario set and forming a group of scenarios by choosing a pre-set number of scenarios from each block of this partition. Taking an expectation over all such groups produces the bound in [47]. The authors identify a partition through a scenario-node cut defined as a subset of nodes in the scenario tree, where each node in this subset corresponds to a unique block of scenarios constituting the partition. They, however, do not offer any guidance on how to choose a good scenario-node cut. Furthermore, while this approach may allow for solving fewer number of group subproblems than that is required in [39], it can still be prohibitive for any but small scenario sets. As demonstrated by the computational experiments in [47], applying this approach in practice can also be rather cumbersome.

The rather obvious generalization of the bounds in [39] to multistage SMIPs is presented in [31]. Maggioni et al. [31] show that the monotonicity of bounds proven in [39] remains valid for multistage SMIPs and that the bounds get tighter as the set of reference scenarios expands. Maggioni and Pflug [32] further extend the scope of these bounds to multistage convex problems with risk functionals. However, neither of these studies [31, 32] are practically applicable for any but the smallest instances, because they both suffer from the same computational challenges as in [39].

Building on the partition-based bounds of Sandıkçı and Özaltın [40], Boland et al. [7] introduce the concept of a $b$-partition and prove that the average bound across
all \( b \)-partitions, called the expected partition bound, is monotone in \( b \). For a given positive integer \( b \), a \( b \)-partition is defined as a partition of the scenario set consisting of \( b \) blocks of size \( b \) or \( b-1 \). Such partitions always exist when \( b \) is sufficiently smaller than the number of scenarios \( N \) (i.e., \( b \leq \sqrt{N} \)). It is important to note that when \( N \) is perfectly divisible by \( b \), the expected partition bound equals the multistage extension of the bound in \([39]\), and their monotonicity proof relies on this equivalence. Calculating the expected partition bound across all \( b \)-partitions has the same level of computational burden as calculating the bound in \([39]\), therefore Boland et al. \([7]\) resort to sampling a small set of \( b \)-partitions and report the best bound.

In this paper, we establish that the bounds in \([39]\) can be extended to multistage stochastic programs with general functions and variables. Furthermore, we prove that these extended bounds can be derived from any blockset, i.e., a collection of \( b \)-subsets of the scenario set that covers the entire scenario set. At one extreme, for a given positive integer \( b \), a blockset can be composed of all \( b \)-subsets of the scenario set as in \([39]\), while at the other extreme it can simply be a partition of the scenario set. Allowing scenarios to appear an unequal number of times in a blockset can be viewed as a generalization of the reference scenario concept first introduced by Birge \([4]\). Such a flexibility in forming blocksets allows for designing custom scenario blocks to incorporate interactions among scenarios. We further establish that valid bounds can be generated from any multiset, which can be viewed as a generalized blockset that allows repetition of blocks. The effect of repeating blocks is to weigh the scenario interactions in the repeated blocks more heavily in the bound calculations. We also establish a rather strong dominance relationship between two multisets, which achieves the monotonicity results in \([31, 39]\) as a special case. Finally, unlike any of the previous studies, we use a modern computing cluster to illustrate the performance of our proposed method with a rich set of results on problems from the literature. The size of the instances we solve are truly large compared to what is reported in the literature. We observe immense potential to obtain high quality solutions to difficult large-scale multistage stochastic programs within a reasonable time frame.

The rest of this paper is organized as follows. We review multistage programs in Section 2. We present our decomposition approach in Section 3, followed in Section 4 by the bounds based on this decomposition. We present extensive numerical results in Section 5, and conclude in Section 6 with a summary and future research directions.

2. Multistage stochastic programs. We consider finite-horizon sequential decision making problems under uncertainty. Decisions are made at discrete stages, indexed by \( t = 1, 2, \ldots, T \), using the available information by that stage. A finite number of random events \( \omega_t \in \Omega_t \) may be realized in stage \( t \). The decision process starts with an initial decision \( x_1 \) in stage 1. In each subsequent stage \( t = 2, \ldots, T \), first an outcome of the random event \( \omega_t \) is observed and then a recourse action \( x_t \) is taken. Taking action \( x_t \) under the random event \( \omega_t \) in stage \( t \) contributes \( f_t(x_t, \omega_t) \) to the objective function. Furthermore, the set of permissible decisions \( X_t \subseteq \mathbb{R}^{n_t} \), where \( n_t \) is a positive integer, in stage \( t \) may depend on the history of decisions \( x_{1:t-1} := (x_1, \ldots, x_{t-1}) \) as well as the observed outcomes of the random events \( \omega_{2:t} := (\omega_2, \ldots, \omega_t) \).

We associate a random outcome vector \( \xi_t \in \mathbb{R}^{n_t} \) with each event \( \omega_t \in \Omega_t \), and drop the explicit dependence on \( \omega_t \) and hereafter write, for example, \( f_t(x_t, \xi_t) \) for the objective function contribution in stage \( t \). The overall objective is to minimize the total expected contributions, so a nested formulation is given by

\[
\min_{x_t \in X_t} f_t(x_t) + \mathbb{E}_{\xi_t} \left[ \min_{x_2 \in X_2} f_2(x_2, \xi_2) + \cdots + \mathbb{E}_{\xi_{t-1}} \left[ \min_{x_T \in X_T} f_T(x_T, \xi_T) \right] \right],
\]
where $E_{\xi_{t+1}\mid\xi_{2:t}}[\cdot]$ is the expectation with respect to $\xi_{t+1}$ conditioned on the realization $\xi_{2:t} := \{\xi_2, \ldots, \xi_t\}$, and $\mathcal{N}_{t} = \{x_t \in \mathbb{R}^{m_t} : g_t(x_{1:t}, \xi_{2:t}) = h_t(\xi_{2:t})\}$ is the set of permissible decisions in stage $t = 1, \ldots, T$, where $g_t(\cdot) : \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_t} \times \cdots \times \mathbb{R}^{n_T} \to \mathbb{R}^{m_t}$ and $h_t(\cdot) : \mathbb{R}^{r_2} \times \cdots \times \mathbb{R}^{r_T} \to \mathbb{R}^{m_t}$ for positive integers $n_t, r_t$, and $m_t$ for all $t$. Note that some or all components of the decision vector $x_t$ can be restricted to integers.

![Scenario tree representation of uncertainty.](image)

We assume that, for all stages $t$, the random vector $\tilde{\xi}_t$ is discrete with finite number of realizations. Thus, the stochastic process $\tilde{\xi} = \{\tilde{\xi}_2, \tilde{\xi}_3, \ldots, \tilde{\xi}_T\}$ has a finite support $\Xi = \{\xi^1, \xi^2, \ldots, \xi^N\}$, where $\xi^s = (\xi^s_1, \xi^s_2, \ldots, \xi^s_T)$ for $s \in \mathcal{S} := \{1, 2, \ldots, N\}$ is referred to as a scenario and is associated with a probability mass $p^s$. As a result, the uncertainty in the decision process can be represented by a scenario tree as in Figure 1. Each layer of the tree corresponds to a stage $t$ of the decision process. The single (root) node, indexed by 0, in stage 1 is due to the fact that the first set of decisions is made prior to observing the outcomes of any random events. Let $N_t$ denote the set of nodes in stage $t$, $N := \bigcup_{t=1}^{T} N_t$ the set of all nodes in the tree, $A_{m,n}$ the set of nodes on the path from node $m \in N$ to $n \in N$, $a(n)$ the parent node and $C_n$ the set of children nodes of node $n \in N$ (clearly, $a(0) = \emptyset$, $C_n = \emptyset$ for $n \in N_T$). Then, any node $n \in N_t$ for $t = 2, \ldots, T$ represents a particular realization of $\xi_{2:t}$, which can be interpreted as the state of the system in stage $t$. The (conditional) probability $\pi_n$ of node $n \in N_t$ is $P(\xi_t \mid \xi_{2:t-1})$, so $\sum_{n \in C_m} \pi_n = 1$ for any node $m \in N$. The unique path from the root node to a terminal node $n \in N_T$ corresponds to a scenario, and thus the number of terminal nodes $|N_T|$ equals the number of scenarios $N$. The probability associated with path $s \in \mathcal{S}$ from the root node to a terminal node is $p^s$, which equals the product of the conditional probabilities of all nodes that belong to path $s$. Let $S_n$ be the index set of scenarios passing through node $n$. Then the marginal probability of node $n \in N$ is given by $q_n = \sum_{s \in S_n} p^s$.

When there is a finite number of scenarios, problem (1) can be equivalently formulated as a large-scale deterministic program by duplicating the decision vector $x_t$ in stage $t$ for each scenario tree node $n \in N_t$. Let $x^n$ denote the decisions made at node $n \in N_t$ in stage $t$, and for $s \in S_n$ define $f^n(x^n) \equiv f_t(x_t, \xi^s_t)$,
\( g^n(\{x^i\}_{i \in \mathcal{A}_{0,n}}) \equiv g_t(x_{1,t}, \xi_{2,t}^n), \) and \( h^n \equiv h_t(\xi_{2,t}^n). \) The extensive form is given by:

\[
(2a) \quad z^* = \min \sum_{n \in \mathcal{N}} q_n f^n(x^n)
\]

\[
(2b) \quad \text{s.t.} \quad g^n(\{x^i\}_{i \in \mathcal{A}_{0,n}}) = h^n \quad n \in \mathcal{N},
\]

\[
(2c) \quad x^n \in \mathbb{R}^{n_t} \quad n \in \mathcal{N}_t, \quad t = 1, \ldots, T.
\]

Note that this formulation implicitly models the so-called nonanticipativity conditions, which require that if the realization of two scenarios \(s_1, s_2 \in \mathcal{S}\) are indistinguishable from each other up to stage \(t\), then the value of the decision vector \(x_t\) must not differ under these scenarios. In most applications, the extensive form formulation results in very large-scale mathematical programs and so it is not amenable to direct solutions with commercial solvers.

3. A group subproblem. Our goal in this section is to construct a deterministic mathematical program similar to problem (2), but smaller in size so that it can be solved with reasonable computational effort (e.g., by an off-the-shelf solver). To achieve this, we write problem (2) for a subset \(\Gamma \subseteq \mathcal{S}\) of scenarios, which we refer to as a block henceforth. Deriving valid bounds for \(z^*\) from such restricted problems requires adjusting the probability of each scenario \(s \in \Gamma\) in a structured manner.

Let \(\mathcal{P}(\mathcal{S})\) be the power set of \(\mathcal{S}\) excluding the empty set, and \(\mathcal{P}_b(\mathcal{S}) := \{\Gamma \in \mathcal{P}(\mathcal{S}) : |\Gamma| = b\} \) for all \(\Gamma\). A blockset \(\mathcal{G}\) is a set of blocks such that \(\mathcal{G} \subseteq \mathcal{P}(\mathcal{S})\) and \(\bigcup_{\Gamma \in \mathcal{G}} \Gamma = \mathcal{S}\). For any blockset \(\mathcal{G}\), let \(m_s(\mathcal{G})\) be the multiplicity of scenario \(s \in \mathcal{S}\), i.e.,

\[
m_s(\mathcal{G}) = \sum_{\Gamma \in \mathcal{G}} 1_{\Gamma}(s) \quad \text{for} \ s \in \mathcal{S}, \quad \text{where} \ 1_{\Gamma}(s) = 1 \text{ if } s \in \Gamma, \text{ and } 0 \text{ otherwise. Define}
\]

the adjusted probability of scenario \(s\) appearing in block \(\Gamma\) of a blockset \(\mathcal{G}\) as

\[
(3) \quad \hat{p}^s(\mathcal{G}, \Gamma) := \frac{p^s / m_s(\mathcal{G})}{\rho(\mathcal{G}, \Gamma)},
\]

where \(\rho(\mathcal{G}, \Gamma) := \sum_{s \in \Gamma} p^s / m_s(\mathcal{G})\). Also let \(\mathcal{N}^T \subseteq \mathcal{N}\) be the subset of scenario tree nodes that are associated with the scenarios in a given block \(\Gamma\), and \(\mathcal{N}_t^\Gamma\) denote those nodes of \(\mathcal{N}^T\) at stage \(t = 1, \ldots, T\). For \(\Gamma \in \mathcal{G}\), define the adjusted probability of node \(n \in \mathcal{N}^T\) as \(\hat{q}_n(\mathcal{G}, \Gamma) := \sum_{s \in \Gamma_n} \hat{p}^s(\mathcal{G}, \Gamma)\), where \(\Gamma_n\) is defined (similar to \(\mathcal{S}_n\)) as the set of scenarios in \(\Gamma\) that pass through node \(n\).

**Definition 1.** For any blockset \(\mathcal{G} \subseteq \mathcal{P}(\mathcal{S})\), the group subproblem for a block of scenario indices \(\Gamma \in \mathcal{G}\) is

\[
z(\mathcal{G}, \Gamma) = \min \sum_{n \in \mathcal{N}^T} \hat{q}_n(\mathcal{G}, \Gamma) \cdot f^n(x^n) \quad \text{(**GR(\mathcal{G}, \Gamma))**}
\]

\[
\text{s.t.} \quad g^n(\{x^i\}_{i \in \mathcal{A}_{0,n}}) = h^n \quad n \in \mathcal{N}^T,
\]

\[
x^n \in \mathbb{R}^{n_t} \quad n \in \mathcal{N}^T, \quad t = 1, \ldots, T.
\]

A group subproblem can also be defined by including a set of reference scenarios \(\mathcal{R} \subseteq \mathcal{S}\). The idea of using a reference scenario has appeared in [4, 30, 31, 39]. Including a reference scenario set into a group subproblem serves the purpose of tempering the effect of extreme scenarios so that group subproblem solutions are not too biased. While the choice of a (set of) reference scenario(s) may not be obvious in general, knowledge of problem specific features in a particular application may naturally produce such a set. Let \(p_\mathcal{R} = \sum_{s \in \mathcal{R}} p^s\) denote the total probability of \(\mathcal{R}\), and \(q^n_\mathcal{R} = \sum_{s \in \mathcal{R}_n} p^s\) with \(\mathcal{R}_n\) denoting the index set of reference scenarios passing through node \(n\).
DEFINITION 2. For \( \mathcal{R} \subseteq \mathcal{S} \) and any blockset \( \mathcal{G} \subseteq \mathcal{P}(\mathcal{S} \setminus \mathcal{R}) \), the group subproblem with the reference set \( \mathcal{R} \) for a block of scenario indices \( \Gamma \in \mathcal{G} \) is
\[
\begin{align*}
\zeta^\mathcal{R}(\mathcal{G}, \Gamma) &= \min_{n \in \mathcal{N}^\mathcal{R}} \sum_{n} q_n^{\mathcal{R}} \cdot f^n(x^n) + (1 - p_{\mathcal{R}}) \sum_{n \in \mathcal{N}^\mathcal{R}} g_n(\mathcal{G}, \Gamma) \cdot f^n(x^n) (G^R(\mathcal{G}, \Gamma)) \\
\text{s.t.} & \quad g^n\left(\{x^t\}_{t \in \mathcal{A}_n}\right) = h^n, \quad n \in \mathcal{N}_1^\mathcal{T} \cup \mathcal{N}_1^\mathcal{R}, \\
& \quad x^n \in \mathbb{R}^{m_n}, \quad n \in \mathcal{N}_1^T \cup \mathcal{N}_1^R, \quad t = 1, \ldots, T.
\end{align*}
\]

We illustrate both group subproblem definitions using a simple example in Figure 2. If \( \mathcal{R} = \emptyset \), we take \( p_{\mathcal{R}} = q_n^{\mathcal{R}} = 0 \), and \( \mathcal{N}^\mathcal{R} = \emptyset \), and therefore, the two definitions become identical. Given a nonempty reference set \( \mathcal{R} \subseteq \mathcal{S} \) and \( \mathcal{G} \subseteq \mathcal{P}(\mathcal{S} \setminus \mathcal{R}) \), one can construct a valid blockset \( \mathcal{G}' \subseteq \mathcal{P}(\mathcal{S}) \) by appending \( \mathcal{R} \) to each block \( \Gamma \in \mathcal{G} \). Both definitions incorporate the interaction between scenarios in \( \mathcal{R} \) and other scenarios in the group subproblem, but less weight is given to the influence of scenarios in \( \mathcal{R} \) in \( \zeta(\mathcal{G}', \Gamma) \) than in \( \zeta(\mathcal{G}, \Gamma) \). Consequently, although both subproblems use the same scenarios, \( \zeta(\mathcal{G}', \Gamma \cup \mathcal{R}) \) can be significantly different from \( \zeta(\mathcal{G}, \Gamma) \), but it is hard to predict which definition would yield a tighter bound.

The full problem depicted in (a) has \( T = 3 \) stages and \( N = 5 \) scenarios. Panel (b) depicts the scenario trees of group subproblems based on Definition 1 for blockset \( \mathcal{G}^1 = \{\Gamma^1, \Gamma^2, \Gamma^3\} \), where \( \Gamma^1 = \{1, 5\} \), \( \Gamma^2 = \{2, 4\} \), \( \Gamma^3 = \{2, 3\} \); thus \( m_1(\mathcal{G}^1) = 1 \) for \( s = 1, 3, 4, 5 \), and \( m_2(\mathcal{G}^1) = 2 \). Panel (c) depicts the scenario trees of group subproblems based on Definition 2 for blockset \( \mathcal{G}^2 = \{\Gamma^3, \Gamma^4\} \) where \( \Gamma^4 = \{1, 4, 5\} \cup \mathcal{R} \), \( \Gamma^5 = \{3, 4\} \cup \mathcal{R} \), and \( \mathcal{R} = \{2\} \); thus \( m_a(\mathcal{G}^2) = 1 \) for \( s = 1, 3, 5 \), and \( m_4(\mathcal{G}^2) = 2 \).

Definitions 1 and 2 allow flexibility in choosing different multiplicities for scenarios, whereas [31, 39] are restricted to using the same multiplicity for all scenarios. Such flexibility can be useful in practice. A user can exploit specific problem structure when designing a blockset, and include various (sets of) scenarios in a select set of group subproblems to incorporate custom scenario interactions, leading to unequal multiplicities. For a stochastic capacity expansion problem, Birge [4] and Birge and
Louveaux [5] suggest formulating group subproblems by including the worst-case scenario, defined as the one with the highest demand level. In our setting, such a choice is equivalent to increasing the multiplicity of the worst-case scenario while keeping the multiplicity of all other scenarios as 1, forming a blockset that is not a partition.

Such flexibility is also typically associated with significant computational savings when compared to using rigid blocksets of the form \( P_b(S) \) as in [31, 39]. Bounds based on \( P_b(S) \) require solving \( N \) group subproblems, whereas competing bounds from a partition can be obtained by solving (about) \( N/b \) group subproblems. While each subproblem is expected to have comparable solution times, bounds based on \( P_b(S) \) require solving approximately \( (N/b) \cdot (N-1) \) times as many group subproblems as required in any partition-based bound using blocks of \( b \) scenarios.

4. Bounds from the group subproblems. We now illustrate how one can generate bounds for \( z^* \) from a blockset \( G \). A unique feature of these bounds is that they can solve the group subproblems independently in parallel, which makes this method attractive to potentially address large-scale instances. The user has the flexibility of adjusting the block size according to the available computational resources.

4.1. A lower bound. Our main results in this section (Propositions 5 and 6) provide lower bounds for the optimal objective value \( z^* \) of the full problem (2) for any given blockset. For any blockset \( G \subseteq \mathcal{P}(S) \), let

\[
\mathcal{L}(G) := \sum_{\Gamma \subseteq G} \rho(G, \Gamma) \cdot z(G, \Gamma).
\]

Similarly, for a set of reference scenarios \( \mathcal{R} \subseteq S \) and a blockset \( G \subseteq \mathcal{P}(S \setminus \mathcal{R}) \), let

\[
\mathcal{L}^R(G) := \frac{1}{1 - p_{\mathcal{R}}} \sum_{\Gamma \subseteq G} \rho(G, \Gamma) \cdot z^R(G, \Gamma).
\]

**Lemma 3.** \( \sum_{\Gamma \subseteq G} \rho(G, \Gamma) = 1 - p_{\mathcal{R}} \) for any \( \mathcal{R} \subseteq S \) and \( G \subseteq \mathcal{P}(S \setminus \mathcal{R}) \).

**Proof.** \( \sum_{\Gamma \subseteq G} \rho(G, \Gamma) = \sum_{\Gamma \subseteq G} \frac{\rho^R}{m_s(\mathcal{R})} = \sum_{s \in \mathcal{S} \setminus \mathcal{R}} m_s(G) \, \frac{\rho^s}{m_s(G)} = 1 - p_{\mathcal{R}}. \)

**Lemma 4.** For any function \( h(\cdot) : \mathcal{S} \rightarrow \mathbb{R} \) and any blockset \( G \subseteq \mathcal{P}(S) \),

\[
\sum_{\Gamma \subseteq G} \sum_{s \in S} \frac{p^s}{m_s(G)} \cdot h(\{s\}) = \sum_{s \in S} p^s \cdot h(\{s\}).
\]

**Proof.** \( \sum_{\Gamma \subseteq G} \sum_{s \in S} \frac{p^s}{m_s(\mathcal{G})} \cdot h(\{s\}) = \sum_{s \in S} m_s(G) \frac{p^s}{m_s(G)} \cdot h(\{s\}) = \sum_{s \in S} p^s \cdot h(\{s\}). \)

**Proposition 5.** \( \mathcal{L}^R(G) \leq z^* \) for reference set \( \mathcal{R} \subseteq S \) and blockset \( G \subseteq \mathcal{P}(S \setminus \mathcal{R}) \).

**Proof.** Let \( (\tilde{z}^n)_{n \in \mathcal{N}} \) be an optimal solution to the full problem (2). Substituting this solution into the objective function of the problem \( GR^R(G, \mathcal{R}) \) for \( \Gamma \in G \), we find

\[
\begin{align*}
\tilde{z}^R(G, \mathcal{R}) &\leq \sum_{n \in \mathcal{N}} \eta_n \cdot f^n (\tilde{z}^n) + (1 - p_{\mathcal{R}}) \sum_{n \in \mathcal{N}} \bar{q}_n (G, \Gamma) \cdot f^n (\tilde{z}^n) \\
&= \sum_{n \in \mathcal{N}} \eta_n \cdot f^n (\tilde{z}^n) + (1 - p_{\mathcal{R}}) \sum_{n \in \mathcal{N}} \sum_{\Gamma \subseteq \mathcal{G}} \bar{q}^s (G, \Gamma) \cdot f^n (\tilde{z}^n) \\
&= \sum_{n \in \mathcal{N}} \eta_n \cdot f^n (\tilde{z}^n) + (1 - p_{\mathcal{R}}) \sum_{n \in \mathcal{N}} \sum_{\Gamma \subseteq \mathcal{G}} \frac{p^s}{\rho(G, \Gamma)} \cdot f^n (\tilde{z}^n) \quad \text{for } \Gamma \in G.
\end{align*}
\]
Multiplying this inequality with \( \frac{\rho(G, \Gamma)}{(1 - p_R)} \) and summing for \( \Gamma \in \mathcal{G} \), we find

\[
\mathcal{L}^R(G) \leq \left( \sum_{\Gamma \in \mathcal{G}} \frac{\rho(G, \Gamma)}{(1 - p_R)} \right) \left( \sum_{n \in \mathcal{N}} q_n^R f^n(\tilde{x}^n) + \sum_{n \in \mathcal{N}^R} \sum_{s \in (\mathcal{S} \setminus \mathcal{R})_n} p^s f^n(\tilde{x}^n) \right)
\]

(4a)

\[
= \sum_{n \in \mathcal{N}^R} \sum_{s \in \mathcal{R}_n} p^s f^n(\tilde{x}^n) + \sum_{n \in \mathcal{N}^R} \sum_{s \in (\mathcal{S} \setminus \mathcal{R})_n} p^s f^n(\tilde{x}^n)
\]

(4b)

\[
= \sum_{n \in \mathcal{N}^R} \sum_{s \in \mathcal{R}_n} p^s f^n(\tilde{x}^n)
\]

(4c)

where equality (4a) follows because, for \( \mathcal{G} \subseteq \mathcal{P}(\mathcal{S} \setminus \mathcal{R}) \), \( \sum_{\Gamma \in \mathcal{G}} \rho(G, \Gamma) = 1 - p_R \) (Lemma 3), and \( \bigcup_{\Gamma \in \mathcal{G}} \Gamma = \mathcal{S} \setminus \mathcal{R} \) and scenario \( s \in \mathcal{S} \setminus \mathcal{R} \) appears \( m_s(G) \) times in \( \mathcal{G} \).

**Proposition 6.** \( \mathcal{L}(G) \leq z^* \) for any blockset \( \mathcal{G} \subseteq \mathcal{P}(\mathcal{S}) \).

**Proof.** Similar to the proof of Proposition 5.

4.1.1. Choosing a blockset \( \mathcal{G} \). The quality of the lower bound \( \mathcal{L}(\mathcal{G}) \) (or, \( \mathcal{L}^R(\mathcal{G}) \)) for a given reference set \( \mathcal{R} \) clearly depends on the choice of the blockset \( \mathcal{G} \). For \( b \ll N \) denoting the size of the largest group subproblem the user can afford (or, is willing) to solve, we are interested in solving the following blockset selection problem:

\[
\max_{G \subseteq \mathcal{P}(\mathcal{S})} \mathcal{L}(G)
\]

s.t. \( |\Gamma| \leq b \) for \( \Gamma \in \mathcal{G} \).

(5a)

Solving problem (5) for an optimal blockset \( \mathcal{G}^* \) is, in general, very difficult. A major factor contributing to this difficulty is its large unstructured search space. To illustrate the computational intractability, we now provide exact formulas to count the number of blocksets. For \( N \) scenarios and \( b \) blocksize, the number of blocksets to consider in problem (5) is given by \( \mathbb{B}_{N,b} \), which can be found recursively:

\[
\mathbb{B}_{N,b} = 2\sum_{i=1}^{b} \binom{N}{i} - 1 - \sum_{i=1}^{N-1} \binom{N}{i} \mathbb{B}_{i,b},
\]

with \( \mathbb{B}_{i,b} = \mathbb{B}_{i,i} \) for \( i \leq b \leq N \), and \( \mathbb{B}_{i,1} = 1 \) for all \( i \leq N \). If blocksets are restricted to form partitions only, then the number of partitions is given by:

\[
\mathbb{P}_{N,b} = \sum_{i=0}^{b-1} \binom{N-1}{i} \mathbb{P}_{N-1-i,b},
\]

with \( \mathbb{P}_{N-1-i,b} = \mathbb{P}_{N-1-i,N-1-i} \) for \( N - 1 - i \leq b \leq N \), and \( \mathbb{P}_{0,b} = 1 \) for all \( b \leq N \). If each block of the partition is further restricted to include exactly \( b \) scenarios, which occurs only when \( b \) perfectly divides \( N \), then the number of partitions is given by:

\[
\mathbb{E}_{N,b} = \frac{1}{(N/b)!} \binom{N}{b, b, \ldots, b} = \frac{N!}{(N/b)!b!^{(N/b)}}
\]

These counts quickly become large even for small values of \( N \) and \( b \) (e.g., \( \mathbb{B}_{16,4} = 2.5 \times 10^{757}, \mathbb{P}_{16,4} = 6.6 \times 10^9 \), and \( \mathbb{E}_{16,4} = 2.6 \times 10^6 \)).
Influence of $b$. We classify blocksets with respect to the sizes of blocks they contain. A blockset $\mathcal{G}$ is $b$-primary if $|\Gamma| \leq b$ for all $\Gamma \in \mathcal{G}$ and $|\Gamma| = b$ for some $\Gamma \in \mathcal{G}$. A blockset $\mathcal{G}$ is $b$-maximal if it is $b$-primary and $|\Gamma_1 \cup \Gamma_2| > b$ for all $\Gamma_1, \Gamma_2 \in \mathcal{G}$. A blockset is $b$-split if it is $b$-primary but not $b$-maximal. Intuitively, any $b$-split blockset can be obtained from refining an appropriately selected $b$-maximal blockset. We establish in Proposition 7 that splitting any block $\Gamma \in \mathcal{G}$ for any $\mathcal{G} \in \mathcal{P}(\mathcal{S})$ into smaller blocks cannot improve the contribution of $\Gamma$ to the lower bound. An important implication of this result is that for each $b$-split blockset there exists at least one $b$-maximal blockset with a tighter lower bound. Therefore, when designing blocksets in pursuit of good lower bounds, blocksets that are $b$-maximal should be preferred over those that are $b$-split. Consequently, the search space for $\mathcal{G}^*$ is significantly reduced, because typically a small fraction of the blocksets are $b$-maximal (see, for example Figure 3).

Proposition 7. Given $\mathcal{G} \subseteq \mathcal{P}(\mathcal{S})$ and $\Gamma \in \mathcal{G}$. For any partition $\Pi$ of $\Gamma$,

$$\sum_{\Gamma' \in \Pi} \rho(\mathcal{G}^{\text{new}}, \Gamma') z(\mathcal{G}^{\text{new}}, \Gamma') \leq \rho(\mathcal{G}, \Gamma) z(\mathcal{G}, \Gamma),$$

where $\mathcal{G}^{\text{new}}$ is the new blockset obtained from $\mathcal{G}$ by partitioning the block $\Gamma \in \mathcal{G}$.

Proof. Observe that $m_s(\mathcal{G}^{\text{new}}) = m_s(\mathcal{G})$ for all $s$. Based on this observation, the rest of the proof follows similar to the proof of Proposition 5.

Fig. 3: Box-whisker plots for all partitions for a 16-scenario SMIP (Instance: MLS-2-5)

For each boxplot, horizontal line and diamond inside the box, respectively, denote the median and the mean; upper and lower endpoints of the box, respectively, denote 90th and 10th percentiles; and endpoints of the upper and lower whiskers, respectively, denote maximum and minimum. The total number of partitions evaluated is $10,480,142,147$, of which only $3.80\%$ are $b$-maximal. The set of $b$-split partitions is empty for $b = 1, 15, 16$, therefore no boxplots are drawn in these cases.

Increasing $b$ yields a better optimal objective value in problem (5) (i.e., a tighter lower bound $\mathcal{L}(\mathcal{G}^*)$), but this improved bound comes at the expense of solving larger group subproblems. While this monotonicity result is true for the optimal blockset $\mathcal{G}^*$, it does not necessarily apply to all blocksets as illustrated in Figure 3, which displays the distributions of lower bounds $\mathcal{L}(\mathcal{G})$ for a 16-scenario instance (details of the instance used are provided in §§5.1-5.2). We restrict the numerical demonstration in Figure 3 to partitions instead of general blocksets due to two reasons. First, the set of general blocksets is too large to enumerate — for $N = 16$ scenarios, there are a
total of $1.0 \times 10^{19728}$ general blocksets, but only $1.0 \times 10^{10}$ partitions. Secondly, the insights discussed below remain valid for partitions as well as general blocksets.

The best lower bound $\mathcal{L}(\mathcal{G}^*)$ in Figure 3 monotonically improves, as expected, when $b$ increases from 1 to 16. Furthermore, while the best lower bound from the class of $b$-maximal (or $b$-split) partitions is not necessarily monotone in $b$, the average lower bound within the class monotonically improves with $b$. This observation can indeed be proven in a similar fashion to the proof of Corollary 10 below, in which we establish the monotonicity of the average lower bound obtained from the set of all $b$-subsets of $\mathcal{S}$. However, significant overlaps between the boxplots within the class of $b$-maximal (or $b$-split) partitions when $b = 2, \ldots, 15$ demonstrate that a larger value for $b$ is, in general, not sufficient to obtain tighter lower bounds. The only exception to this phenomenon is when each block of a given partition is contained in one of the blocks of another partition, in which case the lower bound from the latter partition is guaranteed to be tighter (see Proposition 7). Consider, for example, $b = 1$ and $b = 16$ in Figure 3. Every block of the partition formed when $b = 1$—note that there is only one such partition—is necessarily contained by one of the blocks of any partition formed with $b \geq 2$, and therefore, it is not surprising that the lower bound is loosest when $b = 1$. Conversely, the only block of the partition formed when $b = 16$—note that this partition corresponds to the full problem—necessarily contains any block of any partition formed with $b < 16$, and therefore, the lower bound is tightest when $b = 16$. Sensitivity analysis results in §5.6 confirm similar observations when various distributions are used to model the random parameters in each instance.

Influence of scenarios’ multiplicities. Computing $\mathcal{L}(\mathcal{G})$ introduces flexibility in constructing group subproblems as it allows the scenarios to be repeated for an unequal number of times in $\mathcal{G}$, which may be a preferred strategy if some scenarios are known to have more influence on the optimal objective value $z^*$ than others. However, the flexibility gained by allowing blocksets that are not partitions of $\mathcal{S}$ (i.e., those with multiplicity $m_s > 1$ for some $s \in \mathcal{S}$) is offset by significant increases in the combinatorial burden of problem (5). At the same time, using such blocksets can influence the quality of the lower bound in an unstructured way. Consider the following example:

$$\min_{x \geq 0} \ 10x + E_ξ \left[ \min_{y \geq 0} \ -6y \right] \ \text{subject to} \ y \leq x, \ \frac{1}{2}ξ \leq y \leq ξ,$$

where $E = \{5, 10, 15, 20\}$ with equally likely scenarios. We solve problem (5) for this example with an additional constraint on multiplicity for each scenario and summarize the results in Table 1. In the base case, when $b = 2$ and the multiplicity vector $m = [1,1,1,1]$ is imposed, the best blockset $\mathcal{G}^1$ yields a lower bound value of 42.500.

The two blocksets $\mathcal{G}^2$ and $\mathcal{G}^3$, obtained when $m = [1,1,2,2]$ and $m = [2,2,1,1]$, respectively, contain $\mathcal{G}^1$ and another block of two scenarios. It can be argued that these latter blocksets use more scenario information than $\mathcal{G}^1$ since they capture interactions between those scenarios in the additional block that does not exist in $\mathcal{G}^1$. Interestingly, however, the base case bound is improved with $\mathcal{G}^2$, while it is worsened with $\mathcal{G}^3$. The deleterious effect of increasing the multiplicities of scenarios 1 and 2 on the lower bound persists even when the blocksize is relaxed to 3 as seen in $\mathcal{L}(\mathcal{G}^3)$.

Constructing an optimal blockset $\mathcal{G}^*$ is still challenging even when multiplicities of scenarios are given and the optimal objective values for all relevant blocks are somehow known. It can be shown that, under these assumptions, the blockset selection problem reduces to a generalized set partitioning problem (GSPP), which is NP-hard. Thus, when the blocksize and multiplicity of each scenario are given, if one can afford
Table 1: An example illustrating the effect of scenario multiplicity on lower bound

<table>
<thead>
<tr>
<th>b</th>
<th>$m_1(G)$</th>
<th>$m_2(G)$</th>
<th>$m_3(G)$</th>
<th>$m_4(G)$</th>
<th>Best blockset ($G^*$)</th>
<th>Bound ($L(G^*)$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$G^1 = {{1,4},{2,3}}$</td>
<td>42.500</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>$G^2 = {{1,4},{2,3},{3,4}}$</td>
<td>43.750</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>$G^3 = {{1,4},{2,3},{1,2}}$</td>
<td>35.000</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>$G^4 = {{1,3,4},{2,3,4}}$</td>
<td>47.500</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>$G^5 = {{1,2,4},{1,2,3}}$</td>
<td>40.625</td>
</tr>
</tbody>
</table>

Solving all group subproblems associated with each scenario block, then an optimal blockset can be formed using the approaches developed for GSPP [37]. However, this optimization problem will eventually get intractable as the numbers of scenarios and scenario blocks increase.

As discussed above, the effect of scenarios’ multiplicities is hard to predict and depends on specific problem characteristics. Therefore, in our computational study, we focus on using partitions instead of general blocksets. There is growing interest in studying solution driven partitioning of the scenario set in the recent two-stage stochastic programming literature. Focusing on problems with fixed recourse, Song and Luedtke [44] study partitioning in the context of scenario aggregation, where scenarios in each block of the given partition are aggregated into a single representative scenario in the spirit of scenario reduction. For two-stage stochastic linear programs with simple recourse, they show the existence of a small sufficient partition—a partition that has a small number of blocks (so that the resulting aggregated problem is small in size) and that yields an $\epsilon$-tight bound for the optimal objective value $z^*$ for a small $\epsilon$. Furthermore, they propose an adaptive method that iteratively refines a given partition using the solution from the previous iteration until an optimal solution is found, and report encouraging numerical results using stochastic linear programming instances. However, the effectiveness of this method for more general stochastic programs (e.g., multistage, mixed-integer recourse) is unknown.

Ryan et al. [38] offer a mixed-integer programming (MIP) reformulation of problem (5) when blocksets are restricted to partitions. Relying on the information obtained from evaluating a given set of candidate solutions, this reformulation maximizes a metric that estimates the potential bound improvement from a particular grouping of scenarios. They incorporate this reformulation in a finitely convergent algorithm to find the optimal objective value $z^*$ and numerically demonstrate its effectiveness on two-stage 0-1 stochastic programs. While this framework is feasible for the size of instances considered in their setting, the associated MIP reformulation would quickly render it intractable for large-scale instances. For those cases, we present practical approaches that might yield partitions with good lower bounds.

**Strategies for selecting a partition.** As one of the major goals of this paper is to numerically demonstrate the effectiveness of the proposed bounding method, we consider simple to implement representative strategies for selecting partitions. In so doing, we focus on forming scenario blocks by grouping the leaf nodes of the scenario tree. However, it should be noted that the proposed bounding method can also be employed by grouping non-leaf nodes. The grouping (of scenarios or non-leaf nodes) can also be informed by elaborate methods such as clustering (see, for example, [9]) and scenario tree reduction (see, for example, [18, 19, 20]) that take the information structure of the tree into account.
Given a fixed value \( b \) for the blocksize parameter, we consider the following approaches for selecting a partition with \( \lceil \frac{N}{b} \rceil \) blocks:

1. **Sequential.** Construct the blocks of the partition by grouping \( b \) scenarios in the order of their indexes as they appear in the scenario tree. This strategy attempts to group similar scenarios together — similar in the sense that a large fraction of the scenario tree nodes on the path to each scenario in the block are common.

2. **Random.** Construct the blocks of the partition by randomly sampling \( b \) scenarios without replacement. This strategy attempts to increase within block heterogeneity to capture interactions among scenarios from a broad spectrum.

3. **Common history.** Construct the blocks of the partition such that each group subproblem adheres to the nonanticipativity constraints up to and including time stage \( h \in \mathcal{H} = \{1, \ldots, t \mid |N_t| \leq b\} \). A necessary feature of partitions in this family is then to include scenario tree nodes \( N_1 \cup \cdots \cup N_h \) in each group subproblem formulation. See Figure 4 for an illustrative example.

![Fig. 4: Block formation using the Common history strategy.](image)

A 4-stage problem with 2 branches in each stage. Each block of a partition formed under this strategy must include at least one scenario from each dashed box, and scenarios can be selected randomly without replacement until the total number of scenarios selected equals the given blocksize \( b \). Observe that Random strategy is a special case of the Common history strategy with \( h = 1 \).

The sequential strategy is expected to have the shortest running time per block, because the scenario subtree for a group subproblem induced by this strategy, on average, contains relatively fewer nodes. Due to the same reason, the common history strategy would require progressively longer running times per block as \( h \) increases.

It is, however, less clear which strategy would produce tighter lower bounds. To gain insight into this problem, we numerically compare these strategies using \( b = 10^i \) for \( i = 0, 1, 2, 3 \), for three scenario tree structures (fat-and-short, thin-and-long, and midsize) and two problem classes (MCAP and MLS, details are provided in §5.1). Given \( b \) and \( h \in \mathcal{H} \), when assessing the common history strategy we randomly generate 30 partitions and evaluate the lower bound for each partition. Table 2 summarizes the lower bounds obtained from 2,664 different partitions employed in this experiment, which demonstrate that the lower bound from the common history strategy is insensitive to parameter \( h \). Moreover, while the lower bounds from the common history strategy are consistently worse than those from the sequential strategy by about 2-4% in MLS instances, there are negligible differences in MCAP instances. Similar observations are made in the sensitivity analysis results of §5.6. Therefore, we conclude that the observed differences are not large enough to support the use of the more sophisticated and laborious common history strategy over the sequential strategy.
Table 2: Comparing strategies for selecting a partition: Lower bound

<table>
<thead>
<tr>
<th>Instance</th>
<th>Seq. bound</th>
<th>Common history (h)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocksize (b)</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

MLS-50-4 (“fat-and-short”)        
1   100  0.00     
10  100  -3.69    
100 100  -4.08  -4.10     
1000 100  -3.94  -3.97   
MLS-3-10 (“midsize”)             
1   100  0.00     
10  100  -2.55  -2.56  -2.85     
100 100  -2.49  -2.50  -2.53  -2.61  -2.77   
1000 100  -2.80  -2.79  -2.79  -2.80  -2.83  -2.88  -3.23   
MLS-2-15 (“thin-and-long”)       
1   100  0.00     
10  100  -1.75  -1.75  -1.79  -1.93     
100 100  -1.59  -1.59  -1.60  -1.59  -1.58  -1.69   
1000 100  -2.11  -2.11  -2.10  -2.11  -2.11  -2.11  -2.13  -2.15  -2.19  -2.28   
MCAP-50-4-2-3 (“fat-and-short”)  
1   100  0.00     
10  100  -0.19    
100 100  -0.16  -0.17     
1000 100  -0.08  -0.08   
MCAP-3-10-2-3 (“midsize”)        
1   100  0.00     
10  100  -0.04  -0.03  -0.05     
100 100  +0.05  +0.05  +0.05  +0.05  +0.04   
1000 100  +0.00  -0.01  -0.01  -0.02  -0.01  -0.01  -0.09   
MCAP-2-15-2-3 (“thin-and-long”)  
1   100  0.00     
10  100  +0.04  +0.05  +0.03  +0.02     
100 100  +0.02  +0.03  +0.03  +0.03  +0.03  +0.02  +0.01   
1000 100  +0.02  +0.02  +0.02  +0.02  +0.02  +0.02  +0.02  +0.02  +0.02  +0.02  

* For each blocksize, lower bounds are reported after normalizing them with respect to the corresponding lower bound for the Sequential strategy.

‡ For any blocksize b, h ≤ 1 + log b, where r is the number of scenario tree branches emanating from each non-terminal scenario tree node. For each h, values are reported from averaging across 30 different partitions based on common history strategy.

4.1.2. Monotone lower bounds. Proving monotonicity is an inherently theoretical interest, but recognizing such structures can also help design better blocksets in applications of the proposed bounding method. As we demonstrate in §4.1.1, not only the set of all blocksets is formidable large but also lower bounds across this entire set lack desirable monotonicity property. However, in this section, we show that there exists particular classes of blocksets that exhibit monotone lower bounds.

Consider, for instance, the class of blocksets $P_b(S)$ for positive integers $b \leq N$. The monotonicity of the EGSO($b$) bound offered in [39] for two-stage SMIPs relies on $P_b(S)$. The rather obvious extension of EGSO($b$) to multistage SMIPs in [31], called $MEGSO(b, \cdot)$ bound, and its monotonicity in $b$, similarly relies on $P_b(S)$. In this section, considering a large class of blocksets that includes $P_b(S)$, we prove a general result, which subsumes the monotonicity of the $MEGSO(b, \cdot)$ bounds.

For this purpose, we generalize the blockset definition to include multisets that allow a block $\Gamma \subseteq S$ to appear more than once. Formally, we define a multiset $M$ as a pair $(G, c)$, where $G \subseteq P(S)$ is an ordinary blockset, called the core of $M$, and $c : G \rightarrow \mathbb{Z}_+^{P(S)}$ is the cardinality function denoting the numbers of replicas of each $\Gamma \in G$ appearing in $M$. To obtain a valid lower bound from a multiset $M = (G, c)$, first note that the group subproblem Definitions 1 and 2 require inputting the multiplicity of scenario $s \in S$, which can be easily obtained by $m_s(M) = \sum_{\Gamma \in G} c_{\Gamma} \cdot 1_{\Gamma}(s)$, where $c_{\Gamma}$
denotes the \( i \)th entry in \( c \). Furthermore, the lower bounding formula \( \mathcal{L}(\hat{\mathcal{G}}) \) introduced in §4.1 for ordinary blocksets \( \mathcal{G} \subseteq \mathcal{P}(\mathcal{S}) \) must be adjusted to account for the cardinality of each block in the given multiset \( \mathcal{M} = (\hat{\mathcal{G}}, c) \); that is,

\[
\hat{\mathcal{L}}(\mathcal{M}) := \sum_{\Gamma \in \hat{\mathcal{G}}} c_{\Gamma} \cdot \rho(\mathcal{M}, \Gamma) \cdot z(\mathcal{M}, \Gamma)
\]

\( \hat{\mathcal{L}}(\mathcal{M}) \) reduces to \( \mathcal{L}(\mathcal{G}) \) for any \( \mathcal{M} = (\hat{\mathcal{G}}, c) \) when \( c \leq e \), where \( e \) is a vector of ones.

The main result of this section (Proposition 9) states a (weak) dominance relationship between multisets \( \mathcal{M} = (\hat{\mathcal{G}}, c) \) and \( \hat{\mathcal{M}} = (\hat{\mathcal{G}}, \hat{c}) \) in the sense that \( \hat{\mathcal{L}}(\mathcal{M}) \leq \mathcal{L}(\hat{\mathcal{M}}) \), where the core of \( \hat{\mathcal{M}} \) is composed of blocks of size \( \hat{b} \) and the core of \( \mathcal{M} \) is composed of blocks of size \( b < \hat{b} \). The generalization to multisets is necessary to establish this dominance relationship, because a block \( \Gamma \in \mathcal{G} \) can potentially be covered by more than one block \( \hat{\Gamma} \in \hat{\mathcal{G}} \) (e.g., \( \mathcal{G} \ni \Gamma = \{1, 2, 3\} \) and \( \hat{\mathcal{G}} \ni \hat{\Gamma} = \{1, 2, 3, 4\}, \{1, 2, 3, 5\} \)), in which case ordinary blocksets will fail to maintain the relationship \( \mathcal{L}(\mathcal{G}) \leq \hat{\mathcal{L}}(\mathcal{G}) \).

As a corollary to Proposition 9, we retrieve the monotonicity results in [31, 39] when blocksets of the form \( \mathcal{P}_b(\mathcal{S}) \) are used for lower bounding. We first need Lemma 8 to establish that all scenarios would have the same multiplicity when \( \mathcal{P}_b(\mathcal{S}) \) is used for lower bounding. This feature of using \( \mathcal{P}_b(\mathcal{S}) \) essentially washes away the effect of scenarios’ multiplicities in the group subproblems, and therefore, the \( \text{EGSO}(b) \) and \( \text{MEGSO}(b, \cdot) \) bounds can be viewed as an average across all blocks \( \Gamma \in \mathcal{P}_b(\mathcal{S}) \).

**Lemma 8.** For \( \Gamma \subseteq \mathcal{S} \), \( m_\mathcal{a}(\mathcal{P}_b(\Gamma)) = \binom{|\Gamma| - 1}{b - 1} \) for \( s \in \Gamma \) and positive integer \( b \leq |\Gamma| \).

**Proof.** For any \( \Gamma \subseteq \mathcal{S} \) and \( b \leq |\Gamma| \), every block \( \hat{\Gamma} \in \mathcal{P}_b(\Gamma) \) has \( b \) elements. Given \( |\Gamma| \) scenario indices to construct a \( b \)-set, if we fix one of the elements of the \( b \)-set to a particular index value, we are left with \( |\Gamma| - 1 \) indices from which to choose for the remaining \( b - 1 \) positions in the \( b \)-set. As a result, each scenario index \( s \in \Gamma \) appears \( \binom{|\Gamma| - 1}{b - 1} \) times in \( \mathcal{P}_b(\Gamma) \).

**Proposition 9.** Given positive integers \( b \leq \hat{b} \leq N \), and a multiset \( \hat{\mathcal{M}} = (\hat{\mathcal{G}}, \hat{c}) \) with \( \hat{\mathcal{G}} \subseteq \mathcal{P}_b(\mathcal{S}) \) and \( \hat{c} \in Z_{\mathcal{P}_b(\mathcal{S})}^\mathcal{S} \), \( \hat{\mathcal{L}}(\mathcal{M}) \leq \mathcal{L}(\hat{\mathcal{M}}) \) for any \( \mathcal{M} = (\hat{\mathcal{G}}, c) \) such that \( \hat{\mathcal{G}} = \bigcup_{\Gamma \in \hat{\mathcal{G}}} \mathcal{P}_b(\hat{\Gamma}) \), and \( c_{\Gamma} = \sum_{\hat{\Gamma} \in \hat{\mathcal{G}}} \hat{c}_{\hat{\Gamma}} \cdot 1_{\hat{\Gamma}}(\Gamma) \) for \( \Gamma \in \mathcal{G} \) and 0 elsewhere.

**Proof.** Consider any two multisets \( \mathcal{M} = (\hat{\mathcal{G}}, c) \) and \( \hat{\mathcal{M}} = (\hat{\mathcal{G}}, \hat{c}) \) as specified in the statement of the proposition. Let \( \hat{x}_n \) be an optimal solution to the group subproblem \( GR(\hat{\mathcal{M}}, \hat{\Gamma}) \) for \( \hat{\Gamma} \in \hat{\mathcal{G}} \). It follows that \( \hat{x}_n \) is a feasible solution to the group subproblem \( GR(\mathcal{M}, \Gamma) \) for \( \Gamma \in \mathcal{P}_b(\hat{\Gamma}) \), \( \hat{\Gamma} \in \hat{\mathcal{G}} \), because \( \mathcal{N}_b^\mathcal{S} \subseteq \mathcal{N}_b^\mathcal{G} \). Therefore, for \( \hat{\Gamma} \in \hat{\mathcal{G}} \), \( \Gamma \in \mathcal{P}_b(\hat{\Gamma}) \), and \( b \leq \hat{b} \),

\[
(9) \quad z(\mathcal{M}, \Gamma) \leq \sum_{n \in \mathcal{N}_b} \hat{q}_n(\mathcal{M}, \Gamma) f^n(\hat{x}_n) = \sum_{n \in \mathcal{N}_b} \sum_{s \in \Gamma_n} \frac{p^s / m_\mathcal{a}(\mathcal{M})}{\rho(\mathcal{M}, \Gamma)} f^n(\hat{x}_n).
\]

Multiplying (9) for \( \Gamma \in \mathcal{P}_b(\hat{\Gamma}) \) with \( \hat{c}_{\hat{\Gamma}} \cdot \rho(\mathcal{M}, \Gamma) \) and summing over all \( \Gamma \in \mathcal{P}_b(\hat{\Gamma}) \), we find, for \( \hat{\Gamma} \in \hat{\mathcal{G}} \) and \( b \leq \hat{b} \),

\[
(10a) \quad \sum_{\Gamma \in \mathcal{P}_b(\hat{\Gamma})} \hat{c}_{\Gamma} \cdot \rho(\mathcal{M}, \Gamma) \cdot z(\mathcal{M}, \Gamma) \leq \sum_{\Gamma \in \mathcal{P}_b(\hat{\Gamma})} \sum_{n \in \mathcal{N}_b} \hat{c}_{\hat{\Gamma}} \left( \sum_{s \in \Gamma_n} \frac{p^s}{m_\mathcal{a}(\mathcal{M})} \right) f^n(\hat{x}_n)
\]

\[
= \hat{c}_{\hat{\Gamma}} \sum_{n \in \mathcal{N}_b} \left( \sum_{s \in \Gamma_n} \frac{b - 1}{b - 1} \frac{p^s}{m_\mathcal{a}(\mathcal{M})} \right) f^n(\hat{x}_n)
\]

(10b)
\[ \tilde{c}_\Gamma \sum_{n \in \mathcal{N}^T} \left( \sum_{s \in \mathcal{F}_n} \frac{p^s}{m_s(\mathcal{M})} \right) f^n(\tilde{x}^n), \]

(10c)

\[ \tilde{c}_\Gamma \rho(\tilde{M}, \tilde{\Gamma}) \sum_{n \in \mathcal{N}^T} \hat{q}_n(\tilde{M}, \tilde{\Gamma}) f^n(\tilde{x}^n), \]

(10d)

\[ \tilde{c}_\Gamma \cdot \rho(\tilde{M}, \tilde{\Gamma}) \cdot z(\tilde{M}, \tilde{\Gamma}), \]

(10e)

where (10b) follows, because \( \bigcup_{\Gamma \in \mathcal{P}_b(\hat{\Gamma})} \mathcal{N}^T = \mathcal{N}^\hat{\Gamma} \) and scenario \( s \in \hat{\Gamma} \) appears in exactly \((b-1)\) blocks of \( \mathcal{P}_b(\hat{\Gamma}) \) (Lemma 8), and (10c) follows, because \( m_s(\mathcal{M}) = \binom{b-1}{b-1} m_s(\tilde{\mathcal{M}}) \) for \( s \in \hat{\Gamma} \). Summing over all \( \hat{\Gamma} \in \hat{\mathcal{G}} \), we find, for \( b \leq \hat{b} \), the right hand-side of (10e) as

\[ \sum_{\Gamma \in \hat{\mathcal{G}}} \tilde{c}_\Gamma \cdot \rho(\tilde{M}, \tilde{\Gamma}) \cdot z(\tilde{M}, \tilde{\Gamma}) = \hat{L}(\tilde{M}), \]

(11)

and, since \( \bigcup_{\Gamma \in \hat{\mathcal{G}}} \mathcal{P}_b(\hat{\Gamma}) = \hat{\mathcal{G}} \), the left hand-side of (10e) as

\[ \sum_{\Gamma \in \hat{\mathcal{G}}} \tilde{c}_\Gamma \cdot \rho(\mathcal{M}, \Gamma) \cdot z(\mathcal{M}, \Gamma) = \sum_{\Gamma \in \hat{\mathcal{G}}} \tilde{c}_\Gamma \cdot 1_{\hat{\Gamma}}(\Gamma) \cdot \rho(\mathcal{M}, \Gamma) \cdot z(\mathcal{M}, \Gamma)
\]

(12a)

\[ = \sum_{\Gamma \in \hat{\mathcal{G}}} \tilde{c}_\Gamma \cdot \rho(\mathcal{M}, \Gamma) \cdot z(\mathcal{M}, \Gamma) = \hat{L}(\mathcal{M}), \]

(12b)

which completes the proof.

Corollary 10. \( L(\mathcal{P}_b(\mathcal{S})) \leq L(\mathcal{P}_{b+1}(\mathcal{S})) \) for \( b = 1, 2, \ldots, N - 1 \).}

Proof. For any \( b < N \), in Proposition 9, set \( \tilde{\mathcal{M}} = (\mathcal{P}_{b+1}(\mathcal{S}), \tilde{c}) \) with \( \tilde{c}_\Gamma = 1 \) for \( \hat{\Gamma} \in \mathcal{P}_{b+1}(\mathcal{S}) \) and 0 elsewhere. Therefore, \( \hat{L}(\mathcal{M}) = L(\mathcal{P}_{b+1}(\mathcal{S})) \). Furthermore, \( \bigcup_{\Gamma \in \mathcal{P}_{b+1}(\mathcal{S})} \mathcal{P}_b(\hat{\Gamma}) = \mathcal{P}_b(\mathcal{S}) \) and, for \( \Gamma \in \mathcal{P}_b(\mathcal{S}) \), \( c_\Gamma = \sum_{\Gamma \in \mathcal{P}_{b+1}(\mathcal{S})} \tilde{c}_\Gamma \cdot 1_{\hat{\Gamma}}(\Gamma) = \sum_{\Gamma \in \mathcal{P}_{b+1}(\mathcal{S})} 1_{\hat{\Gamma}}(\Gamma) = N - b \), which yields \( \mathcal{M} = (\mathcal{P}_b(\mathcal{S}), \tilde{c}) \) with \( c_\Gamma = N - b \) for \( \Gamma \in \mathcal{P}_b(\mathcal{S}) \) and 0 elsewhere. Because the core of \( \mathcal{M} \) is the set of all \( b \)-subsets of \( \mathcal{S} \) and the cardinalities of all blocks in the core are identical, we find \( \hat{L}(\mathcal{M}) = \hat{L}(\mathcal{M}^\prime) \), where \( \mathcal{M}^\prime = (\mathcal{P}_b(\mathcal{S}), \frac{1}{N-b} \tilde{c}) \), and therefore, \( \hat{L}(\mathcal{M}) = L(\mathcal{P}_b(\mathcal{S})) \).}

Similar to Proposition 9, we now characterize a set of ordinary blocksets dominated by any given partition. Another major difference between Propositions 9 and 11 is that the former requires each multiset to be composed of same size blocks, while the latter relaxes such a restriction, making Proposition 11 somewhat more flexible.

Proposition 11. \( L(\mathcal{G}) \leq L(\hat{\mathcal{G}}) \) for any partition \( \hat{\mathcal{G}} \) and \( \mathcal{G} = \left( \hat{\mathcal{G}} \setminus \hat{\mathcal{G}} \right) \cup \left( \bigcup_{\Gamma \in \hat{\mathcal{G}}} \mathcal{P}_b(\hat{\Gamma}) \right) \), where \( b(\hat{\Gamma}) \leq |\hat{\Gamma}| \) for \( \hat{\Gamma} \in \hat{\mathcal{G}} \), \( \mathcal{G} \subseteq \hat{\mathcal{G}} \).

Proof. Observe that \( m_s(\mathcal{G}) = 1 \) for all \( s \in \mathcal{S} \), and \( m_s(\mathcal{G}) = \binom{|\hat{\Gamma}| - 1}{b(\hat{\Gamma}) - 1} \) for \( s \in \hat{\Gamma} \), \( \hat{\Gamma} \in \hat{\mathcal{G}} \subseteq \hat{\mathcal{G}} \) and \( m_s(\mathcal{G}) = 1 \) for \( s \in \mathcal{S} \setminus \hat{\mathcal{G}} \). The rest of the proof follows similar to the proof of Proposition 9.

4.2. An upper bound. Lower bounds serve as certificates for the quality of a given feasible solution to the full problem. There may be numerous approaches to obtaining feasible solutions to the full problem, one of which is through completing a given group subproblem solution. Let \( \hat{x}(\mathcal{G}, \Gamma) \) denote an optimal solution to the group subproblem \( GR(\mathcal{G}, \Gamma) \). If the residual problem obtained after substituting \( \hat{x}(\mathcal{G}, \Gamma) \) into
the full problem is feasible, then a feasible solution for the full problem can be found, and the objective value associated with that solution is an upper bound $U(G, \Gamma)$ for $z^*$. When multiple blocks are used to generate upper bounds, the best one is picked:

$$U(G) = \min_{\Gamma \in G} U(G, \Gamma).$$

(See Figure 5 for an illustration.) This approach generalizes the one that was first proposed in Birge [4] for 2-stage problems.

When $\hat{x}(G, \Gamma)$ is substituted into the full problem, the residual problem becomes smaller in size and possibly decomposes into independent subproblems, which can be solved in parallel. Furthermore, while general purpose solvers can be used for solving the residual problems, in practical applications, problem specific features may warrant using special solution approaches or even heuristics, as we do in §5.3.

Fig. 5: Creating upper bounds for the example in Figure 2.

After substituting $\hat{x}(G, \Gamma^1)$ in Figure 2(a), the residual problem is composed only of those decision variables and constraints that are associated with nodes 4, 5, and 6 as depicted in Figure 5(a). If all three subproblems of this residual problem has a feasible solution, then we conclude with an upper bound $U(G, \Gamma^1)$. Panels (b) and (c) illustrate two alternative upper bounds that can be obtained through using $\Gamma^2$ and $\Gamma^3$, respectively.

$$U(G) = \min \left\{ U(G, \Gamma^1), U(G, \Gamma^2), U(G, \Gamma^3) \right\}.$$

5. Computational study. We test the numerical performance of the proposed bounding method using two problems from the literature. For implementation, we use C++ along with the callable library of CPLEX 12.5 in single thread mode to solve subproblems, and openMPI 1.6 for parallelization. We use the sandyb partition within the Midway high performance computing cluster at the University of Chicago Research Computing Center for all numerical tests. Each compute node at sandyb has two eight-core 2.6GHz Intel ‘Sandy Bridge’ processors and 32GB of memory.

5.1. Test Problems. Although our bounding method is applicable to general stochastic programs with recourse, we focus on multistage SMIPs in this computational study. Our first test set is a multistage dynamic capacity acquisition problem, which seeks to determine a minimum cost schedule of acquiring capacity for a set $R$ of resources to satisfy uncertain processing requirements for a set $T$ of tasks. We extend the two-stage model of [2] to a multistage setting to allow revising the capacity decisions as uncertainty unfolds gradually. A multistage SMIP formulation is:

$$\begin{align}
(13a) \quad & \text{[MCAP]} \quad \min \sum_{n \in N} q_n \left( \sum_{i \in R} [f_n^i u_n^i + g_n^i x_n^i] + \sum_{j \in T} \left[ s_n^j z_n^j + \sum_{i \in R} c_n^{ij} y_n^{ij} \right] \right) \\
(13b) \quad & \text{s.t.} \quad x_n^i \leq \kappa_n^i u_n^i, \quad \quad \quad \quad n \in N, i \in R.
\end{align}$$
where \( u_{ni}^i, g_{ni}^j \), and \( z_{ni}^j \) are, respectively, the indicator variables for expanding capacity of resource \( i \), assigning resource \( i \) to task \( j \), and not completing task \( j \), and \( x_n^i \) is the amount of capacity expansion for resource \( i \) for \( n \in \mathcal{N}, i \in \mathcal{R}, j \in \mathcal{T} \). The parameters \( f_n^i \) and \( g_n^i \), respectively, denote the fixed and variable costs of expanding capacity of resource \( i \), \( c_{nj}^i \) is the cost of assigning \( i \) to \( j \), \( s_n^i \) is the penalty cost of not completing task \( j \), \( \kappa_n^i \) is the upper limit on the capacity expansion of resource \( i \), and \( d_n^j \) is the processing requirement of task \( j \) for \( n \in \mathcal{N}, i \in \mathcal{R}, j \in \mathcal{T} \). Objective function (13a) minimizes the total expected cost of capacity acquisition and resource assignment to tasks. Constraints (13b) enforce the capacity limit for each resource, (13c) limit the total assigned capacity of a resource by the acquired capacity of that resource, and (13d) identify if task \( j \) is assigned to any resource or not.

Our second test set is the multistage single-item stochastic lot-sizing problem studied in [17]. The objective of this problem is to determine a minimum cost production and inventory holding schedule for a product so as to satisfy its stochastic demand over a planning horizon. A multistage SMIP formulation is:

\[
\begin{align*}
\text{(14a)} & \quad [\text{MLS}] \quad \min \sum_{n \in \mathcal{N}} q_n (\alpha_n x_n + \beta_n y_n + h_n s_n) \\
\text{(14b)} & \quad \text{s.t. } s_{n(0)} + x_n = d_n + s_n, \quad n \in \mathcal{N}, \\
\text{(14c)} & \quad x_n \leq M_n y_n, \quad n \in \mathcal{N}, \\
\text{(14d)} & \quad x_n, s_n \geq 0, \quad y_n \in \{0, 1\}, \quad n \in \mathcal{N}, \\
\text{(14e)} & \quad s_{n(0)} = 0,
\end{align*}
\]

where \( x_n, s_n, \) and \( y_n \) denote production level, inventory level, and setup indicator variables, and the parameters \( \alpha_n, \beta_n, h_n, d_n, \) and \( M_n \) denote production cost, setup cost, inventory holding cost, demand, and production capacity, respectively, for scenario tree node \( n \in \mathcal{N} \). Objective function (14a) minimizes the total expected inventory holding, production, and setup costs. Constraints (14b) enforce inventory balance, (14c) enforce production capacity limits, and (14e) enforces no initial inventory.

### 5.2. Instance generation.

For each problem class, we randomly generate multiple instances using different initial seeds and symmetric scenario trees. Two parameters common to all instances are the number of stages \( T \), and the number of scenario tree branches \( r \) emanating from each non-terminal node. An instance with \( T \) stages and \( r \) branches per scenario tree node has \( r^{T-1} \) scenarios and \( (r^{T-1})/(r-1) \) scenario tree nodes. In particular, the largest instance solved in §5.5 contains 100M(illion) scenarios and over 111M scenario tree nodes.

We populate data for MCAP instances similar to the two-stage instances of [2], which solves instances up to 3 resources, 4 tasks, and 500 scenarios using a decomposition based branch-and-bound algorithm. In particular, we set \( \kappa_n^i = 1, f_n^i \sim \mathcal{U}[25, 50], \) \( g_n^i \sim \mathcal{U}[5, 10], \) \( s_n^i \sim \mathcal{U}[500, 1000], \) \( c_{nj}^i \sim \mathcal{U}[5, 10], \) and \( d_n^j \sim \mathcal{U}[0.5, 1.5] \) for all \( i \in \mathcal{R}, j \in \mathcal{T}, n \in \mathcal{N} \). We label the MCAP instance with \( |\mathcal{R}| \) resources, \( |\mathcal{T}| \) tasks, \( T \) stages, and \( r \) branches as MCAP-\( r \)-T-\(|\mathcal{R}|\)-\(|\mathcal{T}|\).
We populate data for MLS instances as in [17], which solves instances with up to 128 scenarios using a tailored cutting-plane algorithm. In particular, \(d_n\) is uniformly distributed between 0 and 100, denoted \(d_n \sim U[0, 100]\), \(M_n \sim U[40, 60]\), \(h_n \sim U[40, 60]\), \(h_n \sim U[0, 10]\), \(\alpha_n \sim U[0, 8, 1.2] (\alpha_n / h_n)\) and \(\beta_n \sim U[0, 8, 1.2] (\beta_n / h_n)\), for \(n \in \mathbb{N}\), where the production-to-holding-cost ratio \(\alpha / h = 4\), and the setup-to-holding-cost ratio \(\beta / h = 400\). We label the MLS instance with \(T\) stages and \(r\) branches as MLS-r-T.

### 5.3. Practical considerations. Lower bounding.

Some preliminary work is required for choosing an appropriate blocksize \(b\). Using a larger \(b\) leads to fewer but relatively harder group subproblems for lower bounding. An additional consideration when choosing \(b\) is its impact on the memory requirements for solving the subproblems. Furthermore, for any partition \(\mathcal{G}\), it may be desirable to suboptimally terminate a group subproblem \(GR(\mathcal{G}, \Gamma)\) and use its best known lower bound for \(z(\mathcal{G}, \Gamma)\) when computing the lower bound \(L(\mathcal{G})\). Such early termination strategies can provide significant runtime savings without severely hurting the quality of the lower bound.

![Fig. 6: Impact of suboptimally terminating subproblems on lower bound](image)

Each point displays performance averaged over \(n = 10\) instances except where noted. In each line plot, points from left to right correspond to using an early termination threshold (i.e., relative MIP gap tolerance) of 50%, 25%, 10%, 5%, 4%, 3%, 2%, and 1% when solving group subproblems. At termination thresholds of 5% or less in Figure 6(c) (indicated by the shaded region), most of the 10 instances failed to produce any lower bound due to exceeding the available memory.

The impact of these considerations are illustrated in Figure 6. For each instance, the sequential strategy is used to generate a partition \(\mathcal{G}\), and each group subproblem induced by a block in \(\mathcal{G}\) is solved using an early termination threshold (i.e., relative MIP gap tolerance) varied from 50% to 1%. As expected, for any \(b\), reducing the termination threshold improves the quality of the lower bound at the expense of longer running times. It is, however, important to note that the marginal gain in quality decreases, while the marginal cost in running time increases at a much faster rate. Furthermore, observe that solution time per block increases significantly with \(b\), particularly at lower termination thresholds, and, more strikingly, when \(b = 1000\), the size of the branch-and-bound tree grows so large that it exceeds the available memory for many group subproblems. Insights gleaned from these experiments are further supported by similar experiments with MCAP instances (results not presented).
Upper bounding. For any partition $\mathcal{G}$, one need not solve $U(\mathcal{G}, \Gamma)$ for all $\Gamma \in \mathcal{G}$, which may indeed be preferred in practice to trade off a little bit of upper bound quality with significant runtime savings. Moreover, using fast heuristics as opposed to general purpose solvers may be desirable for solving residual problems. For instance, preliminary experimentation with the large-scale instances considered in this paper revealed that solving the residual problem as an MIP takes unreasonably long runtimes. Therefore, we employ the following tailored heuristics to solve the residual problems:

- **MCAP heuristic:** For each scenario tree node $n$, trying resources in increasing order of their expansion cost $f^i_n$, tentatively expand the capacity of resource $i$ by $k^i_n$, and re-assign tasks using the assignment rule described below. If this expansion is cost-saving, then the increase in the capacity of resource $i$ is made permanent before considering the next resource.

  **Assignment rule.** Given capacity levels of resources at scenario tree node $n$, and considering resources in increasing order of $f^i_n$ for $i \in R$, assign all unassigned tasks $j \in T$ to resource $i$ in decreasing order of $s^j_n - c^j_i$ until no other task can be assigned.

- **MLS heuristic:** At a scenario tree node, production takes place only if the beginning inventory is not enough to satisfy current demand, where the amount of production equals that node’s requirement (i.e. demand minus beginning inventory), plus the expected demand over a look-ahead period.

![Figure 7: Performance comparison of solving the upper bounding residual problem with CPLEX versus heuristic algorithm](image)

Each boxplot summarizes $n$ data points obtained from ten instances. For each instance, the sequential strategy is used with $b = 1,000$ for MCAP instances and $b = 100$ for MLS instances to generate a partition $\mathcal{G}$, and the group subproblem induced by block $\Gamma \in \mathcal{G}$ is solved with a termination threshold of 2%. For $\Gamma \in \mathcal{G}$, the upper bounding residual problem is re-solved twice – once with CPLEX 12.5 as an MIP and a second time with the appropriate heuristic described in this section. When solving the residual problem with CPLEX, a termination threshold of 5% is applied to avoid extra-long runtimes, and furthermore, for MLS instances, the residual scenario tree is split into subtrees with at most 500 nodes to prevent CPLEX from running out of memory.

Figure 7 illustrates the tradeoffs involved in solving the upper bounding residual problems with CPLEX versus the heuristics described above. These results demon-
strate that the heuristics produce competitive upper bounds in significantly shorter durations compared to those obtained from solving the residual problems as an MIP. In particular, the MCAP heuristic, on average, produces an upper bound that is within 5% of the upper bound produced by CPLEX and runs about 100× faster. Similarly, the MLS heuristic consistently produces high quality upper bounds, especially for shorter time stages, and runs, on average, about 400× faster. The relatively smaller savings in runtime for instances with 4 time stages is due to significant speedup in the runtime of CPLEX, not due to any major slowdown of heuristics.

5.4. Experiments with small instances. The proposed bounding approach is intended for large-scale instances. However, to gain insight into numerical performance of these bounds, we first explore small but nontrivial instances for which optimal objective function values may be obtained. Figure 8 displays normalized optimal values and relative values of lower and upper bounds with respect to the optimal value for several MCAP and MLS instances. Two sets of scenario trees are considered: the first fixing the number of branches \( r = 2 \) to illustrate the sensitivity of bounds to the number of time stages \( T \), and the second fixing \( T = 4 \) to illustrate sensitivity to \( r \). The results in Figure 8 demonstrate that lower bounds for MCAP are very tight, and those for MLS are about 5% away from the optimal value. Overall, lower bounds are loosened by increases in \( T \) or \( r \), but \( T \) has more pronounced effect than \( r \); however, this negative impact is negligibly small for MCAP instances, while it is slightly more in effect for MLS instances. This degradation in lower bound quality as \( T \) or \( r \) increases is explained by the fact that the group subproblems constitute a smaller portion of the overall scenario tree with increased \( T \) or \( r \), since blocksize is fixed. The degradation disappears when the blocksize is set so that group subproblems constitute the same fraction of the overall tree (results not presented).

Fig. 8: Quality of lower and upper bounds

To obtain the optimal objective function values for each instance, extensive forms are solved using CPLEX 12.5 with 1 hour time limit. The best known objective value at 1 hour is used for MCAP-2-10 (optimality gap=0.03%), MLS-2-8 (0.7%), MLS-2-9 (1.5%), MLS-2-10 (2.4%), and MLS-8-4 (0.4%). To obtain lower and upper bounds for each instance, the sequential strategy is used with \( b = 5 \) to generate a partition \( \mathcal{G} \). Group subproblems are optimized using CPLEX 12.5 and the upper bounding residual problems are solved using the appropriate heuristic from §5.3.

The upper bounding MLS heuristic is relatively insensitive to changes in \( r \), but it is highly negatively affected by an increase in \( T \), witnessing a steep linear degradation. This degradation is mainly due to the myopic nature of the heuristic decisions, and it persists even after controlling for the size of the group subproblems relative to the
overall tree. An increase in $T$ or $r$ can slightly degrade the quality of the upper bound obtained by MCAP heuristic, but this effect is often small. While Figure 8 displays results for $b = 5$ only, similar patterns have been observed for $b = 1, 2, \ldots, 50$ (see Figure S4 in the Online Supplement).

In the next set of tests, we use moderate size instances to compare the performance of the proposed bounding approach to CPLEX 12.5. To ensure fair comparisons, we execute our approach in serial mode on a single CPU with 2GB memory, and subsequently execute CPLEX in single thread mode with 1 hour time limit for the same instance on the same CPU. Results for a sample of MCAP and MLS instances are summarized in Table 3. The average optimality gap obtained with our approach for MCAP and MLS instances is 5.4% and 9.4%, respectively, whereas the average CPLEX optimality gap during the same time frame is 88.3% and 81.7%, respectively.

<table>
<thead>
<tr>
<th>Problem size*</th>
<th>VSS lower bound (%)</th>
<th>Our approach†</th>
<th>CPLEX Gap1,†</th>
<th>CPLEX with extra time†</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(r, T) = (</td>
<td>R</td>
<td>,</td>
<td>T</td>
<td>)$</td>
</tr>
<tr>
<td>MCAP instances</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(30, 4)-(2, 3)</td>
<td>313.8</td>
<td>108</td>
<td>6.8</td>
<td>87.0 (12.9x)</td>
</tr>
<tr>
<td>(6, 6)-(2, 3)</td>
<td>304.6</td>
<td>39</td>
<td>5.3</td>
<td>89.8 (16.9x)</td>
</tr>
<tr>
<td>(6, 6)-(3, 5)</td>
<td>341.2</td>
<td>91</td>
<td>3.8</td>
<td>85.1 (22.6x)</td>
</tr>
<tr>
<td>(6, 6)-(5, 10)</td>
<td>814.8</td>
<td>1,137</td>
<td>3.8</td>
<td>85.1 (22.6x)</td>
</tr>
<tr>
<td>(5, 7)-(2, 3)</td>
<td>279.7</td>
<td>80</td>
<td>9.7</td>
<td>80.2 (8.2x)</td>
</tr>
<tr>
<td>(3, 10)-(2, 3)</td>
<td>280.3</td>
<td>132</td>
<td>4.6</td>
<td>94.5 (20.5x)</td>
</tr>
<tr>
<td>(2, 15)-(2, 3)</td>
<td>248.1</td>
<td>157</td>
<td>4.0</td>
<td>96.0 (24.1x)</td>
</tr>
</tbody>
</table>

Mean 368.9 249 5.4 88.3 (16.3x) >2,614 (10.5x) 15.6

| MLS instances | | | | | |
|---------------|----------------------|---------------|--------------|------------------------|
| (50, 4)-♦, ♦ | 37.1 | 194 | 4.7 | 37.9 (8.0x) | 1,288 (6.6x) | 32.7$^*$ |
| (18, 5)-♦, ♦ | 49.5 | 185 | 5.2 | 80.1 (15.4x) | >3,600 (19.5x) | 31.7 |
| (6, 7)-♦, ♦ | 71.2 | 128 | 8.8 | 87.1 (9.9x) | >3,600 (28.1x) | 34.0 |
| (3, 15)-♦, ♦ | 94.1 | 108 | 11.1 | 92.3 (8.3x) | 3,429 (31.8x) | 11.0 |
| (2, 15)-♦, ♦ | 116.2 | 193 | 13.1 | 95.9 (7.3x) | 3,318 (17.2x) | 13.0 |
| (2, 17)-♦, ♦ | 118.8 | 745 | 13.7 | 96.7 (7.0x) | 736 (1.0x) | 96.7$^*$ |

Mean 81.1 259 9.4 81.7 (8.0x) >2,662 (10.3x) 36.5

---

* $r = \text{number of branches per non-terminal scenario tree node}$, $T = \text{number of time stages}$, $|R| = \text{number of resources}$, $|T| = \text{number of tasks}$; parameters $|R|$ and $|T|$ are only relevant for MCAP instances. There are $r^{|T|-1}$ scenarios; 2 constraints and 3 variables per scenario tree node for each MLS instance; and 2 $|R|$ + $|T|$ constraints and 2 $|R|$ + $|T|$ + $|R||T|$ variables per scenario tree node for each MCAP instance. The largest instance in this table has 186,620 constraints and 653,170 variables (93% binary).

† For each instance, the sequential strategy is used with $b = 50$ to generate a partition $G$, and each group subproblem is solved using CPLEX with 2% relative MIP gap tolerance and upper bounds are computed using the appropriate heuristic from §5.3 for randomly selected 10 blocks in $G$.

1 The optimality gap reported by CPLEX when it is allowed to run as long as our approach.
2 The degree of looseness as measured by the ratio of CPLEX gap to the gap obtained by our approach.
3 The amount of extra time CPLEX took to reduce its gap to the level obtained by our approach.
4 All times are reported as seconds; all gaps are reported as percentages.
5 CPLEX terminated due to exceeding memory limit before closing the gap.

We have also included in Table 3 a lower bound estimate of the value of the stochastic solution (VSS) to help assess the quality of the optimality gaps reported. We compute this estimate using the difference between the expected result of the solution obtained when random parameters are stage-wise replaced with their mean values [14] and the objective function value of the best known feasible solution to the
extensive form (i.e., VSS lower bound = expected result of the mean value solution / value of the incumbent−1). The average VSS across all MCAP and MLS instances is at least 368.9% and 81.1%, respectively. The relatively large VSS values reassure the significance of the tighter optimality gaps obtained by our approach.

The significantly smaller gaps from our approach are also obtained fairly fast — average run time is 249 seconds and 259 seconds, respectively, for MCAP and MLS. The question of how long it would take CPLEX to achieve as tight of an optimality gap as our bounding approach is answered in the last three columns of Table 3. CPLEX is able to obtain optimality gaps as tight as our approach only for the smallest 4 instances at the expense of about 9× to 32× longer runtimes. On the other hand, for 3 instances, CPLEX exhausts the available memory before closing the optimality gap to the level obtained by our approach. In the remaining 6 instances, CPLEX terminates due to the one-hour time limit, yet still leaving large optimality gaps.

All of these results suggest that even in serial mode the proposed bounding approach is competitive with CPLEX. Furthermore, the inherent parallelizability of the proposed approach suggests an immense potential for obtaining high quality solutions to large-scale instances within a reasonable time frame.

### 5.5. Experiments with large-scale instances.

In this section, we test the performance of our bounding approach on large-scale MCAP and MLS instances. The sizes of some of these instances are several orders of magnitude larger than what has been previously reported in the literature for multistage SMIPs. CPLEX 12.5 with 32GB memory and 36 hours time limit, which compose the global limits for each compute node set by our computing environment, fails to optimize even the linear relaxations of some of these instances.

In our computations, to maximize the number of cores available to us, we use 2GB memory per core and, therefore, 720 cores —one reserved for a master processor distributing jobs and the rest for workers solving subproblems. We use the sequential strategy to generate a partition $\mathcal{G}$ with blocksize $b = 100$ for MLS and $b = 1,000$ for MCAP. For each block in $\mathcal{G}$, considering the tradeoff in running time and lower bound quality observed in Figure 6, we solve group subproblems using CPLEX with an early termination threshold of 2%. We solve the upper bounding residual problems for up to 500 randomly selected blocks in $\mathcal{G}$ using the MCAP heuristic for MCAP instances and the MLS heuristic with a lookahead period of 2 stages for MLS instances. Preliminary experiments revealed that 500 is a sufficiently large sample size to cover the range of the distribution of upper bounds from all blocks in $\mathcal{G}$.

Results are summarized in Tables 4 and 5 for MCAP and MLS, respectively. The first eight instances in both tables fix $r = 10$. Group subproblems in these instances are solved rather quickly —average solution times for MCAP and MLS are $\leq 7.9$ seconds and $\leq 1.0$ second, respectively— and the solution times are fairly insensitive to increases in $T$. However, solution times for upper bounding increase linearly by a factor of about 10 with $T$, which is expected, because the heuristics employed for upper bounding sequentially process each node of increasingly larger scenario trees as $T$ increases. The resulting optimality gap for these MCAP instances is around 5.5%, which remains relatively insensitive to changes in $T$—the nonzero optimality gaps reported for instances (10,2)-(2,3), (10,3)-(2,3), and (10,4)-(2,3) are caused by the early termination of the group subproblems. On the other hand, the optimality gap for MLS instances monotonically increases with $T$. We argue that this deterioration is explained by —as previously observed in §5.4— a degradation in the performance of the MLS heuristic, which makes myopic production decisions.
Table 4: Problem sizes for MCAP instances, solution times, and optimality gaps

| Problem size* \((r, T) - (|R|, |T|)\) | Running time† | Bound quality‡ |
|-------------------------------------|----------------|----------------|
|                                     | \(LP\) \(\mathcal{L}(G)\) \(U(G)\) | \(LP\) Gap (%) |
| \((10, 2) - (2, 3)\)               | 1              | < 0.1 < 0.1 < 0.1 | 0.55 1.3 |
| \((10, 3) - (2, 3)\)               | 1              | < 0.1 < 0.1 < 0.1 | 0.55 0.8 |
| \((10, 4) - (2, 3)\)               | 1              | 0.1 3.6 0.1      | 0.56 1.5 |
| \((10, 5) - (2, 3)\)               | 10             | 1.7 5.9 4.2      | 0.57 5.6 |
| \((10, 6) - (2, 3)\)               | 100            | 43.7 7.9 46.3    | 0.57 5.5 |
| \((10, 7) - (2, 3)\)               | 1,000          | 3,082.8 7.5 469.4 | 0.58 5.4 |
| \((10, 8) - (2, 3)\)               | 10,000         | — 7.5 4,711.8    | — 5.4 |
| \((10, 9) - (2, 3)\)               | 100,000        | — 7.6 47,254.4   | — 5.3 |
| \((25, 4) - (2, 3)\)               | 16             | 3.4 4.1 6.3      | 0.55 6.3 |
| \((50, 4) - (2, 3)\)               | 125            | 81.3 4.7 52.6    | 0.54 6.6 |
| \((200, 4) - (2, 3)\)              | 8,000          | — 5.1 3,365.6    | — 6.7 |
| \((300, 4) - (2, 3)\)              | 27,000         | — 5.5 11,382.3   | — 6.7 |
| \((50, 5) - (2, 3)\)               | 6,250          | — 5.9 2,659.5    | — 6.1 |
| \((25, 6) - (2, 3)\)               | 9,766          | — 6.7 4,256.1    | — 5.9 |
| \((15, 7) - (2, 3)\)               | 11,391         | — 7.4 5,149.4    | — 5.5 |
| \((11, 8) - (2, 3)\)               | 19,488         | — 8.3 9,085.9    | — 5.5 |
| \((5, 10) - (2, 3)\)               | 1,954          | 54,158.0 10.0 1,050.0 | 0.56 4.7 |
| \((3, 15) - (2, 3)\)               | 4,783          | — 17.3 3,182.0   | — 4.5 |
| \((2, 25) - (2, 3)\)               | 16,778         | — 52.5 15,700.9  | — 4.5 |
| \((100, 4) - (2, 3)\)              | 1,000          | 2,770.5 4.8 420.6 | 0.53 6.6 |
| \((100, 4) - (3, 5)\)              | 1,000          | 11,099.3 48.8 668.7 | 0.47 5.3 |
| \((100, 4) - (5, 10)\)             | 1,000          | — 328.5 1,612.9  | — 9.5 |
| \((100, 4) - (5, 20)\)             | 1,000          | — 411.5 2,857.7  | — 12.6 |

- indicates our computing environment including 32GB memory is not sufficient to solve the linear relaxation of this instance.

* \(r\) = number of branches per non-terminal scenario tree node, \(T\) = number of time stages, \(|R|\) = number of resources, \(|T|\) = number of tasks. There are \(r^T - 1\) scenarios. Each scenario tree node contains 2 constraints and 3 variables for MLS instances, \(2|\mathcal{R}| + |\mathcal{T}|\) constraints and \(2|\mathcal{R}| + |\mathcal{T}| + |\mathcal{R}| \cdot |\mathcal{T}|\) variables for MCAP instances. The largest instances in Tables 4 and 5 have 100,000,000 scenarios, which corresponds to \(777,777,777\) constraints, and 1,444,444,443 variables (85% binary) for MCAP, and \(222,222,222\) constraints and \(333,333,333\) variables (1/3 binary) for MLS.

† All times are reported in seconds. \(LP\) is the CPLEX running time for the linear relaxation of the extensive form problem. \(\mathcal{L}(G)\) and \(U(G)\) are the average running time per block solved for our lower and upper bounds, respectively.

‡ \(LP\) is the lower bound from linear relaxation of the extensive form as a fraction of our lower bound \(\mathcal{L}(G)\). Gap (%) is the optimality gap between \(\mathcal{L}(G)\) and \(U(G)\).

The next set of instances in both tables fix \(T = 4\) and change \(r\). Overall, there is a slight but practically insignificant increase in the optimality gap of these instances for both problem classes — average gap is about 6.6% for MCAP and 4.0% for MLS.

The third set in both tables contains instances that increase \(T\) while decreasing \(r\) to retain large but manageable scenario trees. We observe the convolution of the previous two effects in the optimality gaps reported for these instances. The gap for MCAP decreases slightly as the scenario tree goes from fat-and-short to thin-and-long.

However, the gap for MLS increases noticeably as the scenario tree changes in the same fashion, owing mostly due to the degradation in the quality of the upper bound.

746 The next set of instances in both tables fix \(T = 4\) and change \(r\). Overall, there is a slight but practically insignificant increase in the optimality gap of these instances for both problem classes — average gap is about 6.6% for MCAP and 4.0% for MLS.

749 The third set in both tables contains instances that increase \(T\) while decreasing \(r\) to retain large but manageable scenario trees. We observe the convolution of the previous two effects in the optimality gaps reported for these instances. The gap for MCAP decreases slightly as the scenario tree goes from fat-and-short to thin-and-long.

However, the gap for MLS increases noticeably as the scenario tree changes in the same fashion, owing mostly due to the degradation in the quality of the upper bound.
Table 5: Problem sizes for MLS instances, solution times, and optimality gaps

| Problem size \( (r, T) \) | \( |\mathcal{G}| \) | Running time Bound quality | Bound quality |
|--------------------------|----------------|--------------------------|----------------|
|                          | \( L(\mathcal{G}) \) | \( U(\mathcal{G}) \) | \( LP \) | Gap (%) |
| (10, 2)                  | 1              | < 0.1                    | < 0.1         | 0.80   | 1.1   |
| (10, 3)                  | 1              | < 0.1                    | < 0.1         | 0.72   | 1.7   |
| (10, 4)                  | 10             | < 0.1                    | 0.3           | 0.71   | 2.9   |
| (10, 5)                  | 100            | 0.1                      | 1.0           | 0.2    | 4.4   |
| (10, 6)                  | 1,000          | 2.7                      | 1.0           | 2.2    | 6.6   |
| (10, 7)                  | 10,000         | 71.8                     | 0.5           | 22.3   | 8.7   |
| (10, 8)                  | 100,000        | 3,277.0                  | 0.5           | 226.9  | 10.3  |
| (10, 9)                  | 1,000,000      | —                        | 0.4           | 2,295.4 | 11.3 |
| (25, 4)                  | 157            | 0.2                      | 0.4           | 0.71   | 3.3   |
| (50, 4)                  | 1,250          | 2.7                      | 0.5           | 2.3    | 3.5   |
| (100, 4)                 | 10,000         | 48.9                     | 0.3           | 18.4   | 4.3   |
| (200, 4)                 | 80,000         | 1,863.6                  | 0.3           | 145.9  | 4.3   |
| (300, 4)                 | 270,000        | —                        | 0.3           | 498.2  | 4.2   |
| (50, 5)                  | 62,500         | 1,222.1                  | 0.4           | 120.5  | 5.3   |
| (25, 6)                  | 97,657         | 2,861.8                  | 0.5           | 198.4  | 6.8   |
| (15, 7)                  | 113,907        | 3,744.1                  | 0.5           | 244.3  | 5.6   |
| (11, 8)                  | 194,872        | —                        | 0.5           | 439.0  | 9.9   |
| (5, 10)                  | 19,532         | 274.7                    | 0.9           | 51.9   | 10.7  |
| (3, 15)                  | 47,830         | 2,026.0                  | 1.6           | 163.8  | 13.3  |
| (2, 25)                  | 167,773        | —                        | 2.4           | 869.0  | 15.0  |

* See Table 4 for description of each column.

The size of the problem in each node of the scenario tree is fixed in the previous three sets of instances. The last set of MCAP instances in Table 4, however, are such that the size of the scenario tree is fixed, while the size of each node varies with \(|\mathcal{R}|\) and \(|\mathcal{T}|\). Not surprisingly, the average runtimes for the lower and upper bounding subproblems as well as the optimality gap increases with the size of the node. We argue that the deterioration in the optimality gaps is mostly due to the performance of the assignment rule used in the MCAP heuristic for upper bounding, which degrades with the increased number of combinations for larger values of \(|\mathcal{R}|\) and \(|\mathcal{T}|\).

Finally, to assess the performance of the lower bound \( L(\mathcal{G}) \), an entire compute node with 32GB memory is dedicated to CPLEX 12.5 to solve the linear relaxation of the extensive form for each instance. CPLEX fails for the largest 12 MCAP instances. In the remaining 11 MCAP instances, the ratio of the linear relaxation bound to our lower bound ranges between 0.47 to 0.58, illustrating the superiority of our lower bounds. Furthermore, the inferiority of the linear relaxation bounds is exacerbated by significant runtime requirements (e.g., the linear relaxation of MCAP-5-10-2-3 takes over 54,000 seconds to solve, but the resulting bound is only 56% of our lower bound). Similar observations can be made for MLS instances in Table 5.

In summary, these numerical experiments demonstrate that our approach can produce notably tighter lower bounds than the linear relaxation bounds in a significantly shorter duration, owing to its inherent parallelizability. Furthermore, the quality of our lower bounds is observed to be robust with respect to problem size.
5.6. Sensitivity analysis. We test the robustness of our computational findings to the distributions used in modeling the random parameters in each instance. In doing so, we regenerate each MCAP and MLS instance by replacing the Uniform distribution used in §5.2 with a new distribution. We choose six levels for the new distribution, obtained by two levels for distribution family (i.e., Normal and Gamma) and three levels for coefficient of variation (CV) (i.e., baseline, low CV, and high CV). In particular, when regenerating an instance, say, for baseline-Normal, we replace the Uniform distribution used for any random parameter in §5.2 with a Normal distribution whose mean and standard deviation (s.d.) are kept the same as those of the replaced distribution. For low CV and high CV, respectively, we set the s.d. of the new distribution to 50% and 150% of the s.d. of the replaced distribution.

All results from this sensitivity analysis are summarized in the Online Supplement. Figures S2 and S3 confirm our previous discussion surrounding Figure 3—in particular, a larger block size is simply not sufficient to guarantee tighter lower bounds. Furthermore, it is also important to note that the lower bound distribution across partitions may not be symmetric as demonstrated in Figures S2 and S3, which renders the partition selection problem even more difficult.

The sensitivity of partition selection heuristics to parameter distributions is summarized, similar to Table 2, in Table S1. These results confirm our earlier conclusion that the lower bound from the common history strategy is insensitive to parameter $h$. Furthermore, neither the sequential nor the common history strategy can be identified as a clear-cut winner. Sequential strategy is preferred in some instances, while common history strategy is preferred in others, but the differences in the lower bounds are practically insignificant. The only exception to this statement is for the MLS-50-4 instance generated using high CV-Gamma distributions. For this isolated instance, the lower bounds from the common history strategy can be up to 186% better than those from the sequential strategy. Nevertheless, considering the negligible differences in the vast majority of cases, we conclude that the sequential partitioning strategy should be preferred over the more sophisticated and laborious common history strategy.

Figure S5 displays, similar to Figure 8, the sensitivity of the quality of our lower and upper bounds to parameter distributions. Compared to Figure 8, the high quality of our lower bounds for MCAP instances remain virtually unchanged across all distributions tested including the high CV cases, and tighter upper bounds are obtained using the MCAP heuristic. For MLS instances, both lower and upper bounds for the baseline cases are comparable for all three distribution families tested (i.e., Uniform, Normal, and Gamma), but both bounds deteriorate with increases in CV. The deterioration in lower bounds is more pronounced for instances with $r \geq 3$ and Gamma distributed random parameters. Overall, these results demonstrate the potential sensitivity of our bounds to high variance situations.

Tables S2 and S3 summarize the sensitivity results for large-scale MCAP and MLS instances, respectively. To avoid long-winded discussions, we only present, as in the “LP” column in Tables 4 and 5, the ratio of the linear relaxation bound to our lower bound. Table S2 demonstrates that CPLEX 12.5 with 32GB memory fails to solve the linear relaxations for the largest 72 MCAP instances. In the remaining 66 MCAP instances, this ratio ranges between 0.30 and 0.58, confirming the superiority of our lower bounds. Our lower bounds are even more favorable when Normal or Gamma distributions are used instead of Uniform. Surprisingly, however, there is no identifiable pattern with respect to changes in CV—as the CV increases from low to high, the linear relaxation bound relative to our lower bound improves (deteriorates) marginally for instances with Normally (Gamma) distributed random parameters.
For MLS, CPLEX fails for the largest 24 instances, while the ratio of the linear relaxation bound to our lower bound ranges between 0.44 and 4.33 for the remaining 120 instances. This ratio is strictly smaller than 1.0 for all MLS instances except for high CV-Gamma cases. For such instances, the range is between 0.44 and 4.33, illustrating the occasional inferiority of our lower bounds. In general, our lower bound for these high variance cases are relatively loser for smaller time stages $T$ and it improves as $T$ increases. On the other hand, as the scenario tree gets fatter for these high variance cases, our bound deteriorates relative to the linear relaxation bound.

Comparing the run times (not reported in the tables) between the two lower bounding approaches, we find the results immensely in favor of our lower bounding approach. The average (maximum) run time per group subproblem when generating our lower bounds for MCAP and MLS instances is 48.5 (412.1) seconds and 0.4 (2.8) seconds, respectively, while the average (maximum) linear relaxation solution times are 7,127 (121,225) seconds and 3,650 (129,491) seconds, respectively.

In summary, the extensive sensitivity results of this section confirm that the findings summarized in earlier sections of the paper are generally robust to changes in the distributional assumptions made for modeling the random parameters. Furthermore, these results provide a wealth of numerical evidence for the high quality of our (lower) bounds, and exemplify that our lower bounds can occasionally be inferior to the linear relaxation bounds in some high variance cases.

6. Concluding remarks. We develop a bounding framework for general multistage stochastic programs based on scenario decomposition, extending our previous work on two-stage SMIPs [39]. This framework does not make convexity assumptions, rather it builds directly on the power of parallelism and uses out-of-the-box modern solvers. As a result, this approach stands out as a good candidate for use in large-scale implementations of stochastic programming models.

Multistage stochastic programs with integer decision variables are known for their computational difficulty, but this paper provides evidence that there is hope to obtain high quality solutions to practical instances in a reasonable time. We have documented initial numerical experience for our bounding method using large-scale SMIP instances with up to 1.5 billion decision variables (85% binary). While these initial results are very encouraging, they should not be taken as all encompassing tests, but rather as an invitation for further research investment in the proposed bounding approach.

Some aspects of this framework remain topics for further research. For example, our scenario decomposition-based lower bounding method can be used in conjunction with the advanced cutting plane methods in a global branch-and-cut algorithm to obtain exact solutions. Furthermore, we have demonstrated the importance of choosing a blockset on the quality of the bounds, but a more detailed investigation of choosing a good blockset is open. Finally, applying this framework to nonlinear stochastic programs, possibly with integer decision variables, is of future interest.

Acknowledgments. Completed in part with resources provided by the University of Chicago Research Computing Center (RCC). We are particularly thankful for the technical assistance of the RCC staff Robin Weiss. We also thank John R. Birge, Kipp Martin, and Miles Lubin for valuable discussions and helpful comments.

References.


