Optimization Driven Scenario Grouping

Kevin Ryan\textsuperscript{1}, Shabbir Ahmed\textsuperscript{1}, Santanu S. Dey\textsuperscript{1}, and Deepak Rajan\textsuperscript{2}

\textsuperscript{1}School of Industrial & Systems Engineering, Georgia Institute of Technology
\textsuperscript{2}Lawrence Livermore National Laboratory

March 10, 2016

Abstract

Scenario decomposition algorithms for stochastic programs compute bounds by dualizing all nonanticipativity constraints and solving individual scenario problems independently. We develop an approach that improves upon these bounds by re-enforcing a carefully chosen subset of nonanticipativity constraints, effectively placing scenarios into ‘groups’. Specifically, we formulate an optimization problem for grouping scenarios that aims to improve the bound by optimizing a proxy metric based on information obtained from evaluating a subset of candidate feasible solutions. We show that the proposed grouping problem is NP-hard in general, identify a polynomially solvable case, and present a mixed integer programming formulation. We use the proposed grouping scheme as a preprocessing step for a particular scenario decomposition algorithm and demonstrate its effectiveness for solving standard test instances of two-stage 0-1 stochastic programs. Using this approach, we are able to prove optimality for all previously unsolved instances of a standard test set. Finally, the idea is extended to propose a finitely convergent algorithm for two-stage stochastic programs with a finite feasible region.

1 Introduction

Scenario decomposition approaches [1, 8, 12, 14, etc.] for stochastic programs dualize the nonanticipativity constraints and solve the scenario problems independently. Re-enforcing a subset of the nonanticipativity constraints creates scenario groups, sets of scenarios among which the nonanticipativity constraints are enforced. Re-enforcing these constraints leads to a tighter relaxation which provides stronger bounds. Given a bound on the group size, we consider the optimization problem of grouping scenarios so as to maximize the relaxation bound. To exactly evaluate the bound improvement corresponding a specific grouping would require solving the corresponding stochastic program to optimality, thereby defeating the purpose. Instead, we develop a proxy objective to \textit{a priori} estimate the bound improvement using information obtained from evaluating a subset of candidate solutions. Using this proxy objective, we formulate an optimal grouping problem that determines scenario groups. We show that the proposed grouping problem is NP-hard in general and identify a polynomially solvable case. Next, we develop a mixed integer programming formulation of the problem which can be solved by using off-the-shelf packages. We use the proposed grouping scheme as a preprocessing step within the asynchronous scenario decomposition algorithm of [15]. We present computational results on SSCH and SMKP instances from
SIPLIB [2]. This approach is able to prove optimality for several previously unsolved instances from the SSCH instance set. Finally, we extend the idea to propose a finitely convergent algorithm for two-stage stochastic programs with a finite feasible region. The algorithm iteratively solves the grouping problem and the resulting scenario groups for increasing maximum group size, either discovering new candidate solutions or improving the lower bound at each step.

In the remainder of this section, we review previous grouping and aggregation schemes for scenario decomposition algorithms as well as bounding schemes for stochastic programs. There have been many schemes proposed for scenario reduction in stochastic programming. We discuss only those scenario reduction schemes which use scenario optimal solution information to determine the grouping. Additionally, as our method requires bound estimation for scenario groups, we review prior schemes which improve or estimate bounds by scenario grouping.

The notion of solution driven grouping as a form of scenario tree reduction has been explored in two-stage stochastic programming. The authors of [9] use k-means clustering to group similar scenarios and propose a metric in order to group dissimilar scenarios. This approach defines scenarios to be similar or dissimilar based on two criteria, scenario optimal solutions or scenario right hand side values. In [19], a solution driven scenario aggregation approach for two-stage stochastic programs is proposed. The approach begins with a relaxation of the stochastic program and refines the scenario groups until an optimal solution is found. Our method differs from these previous scenario grouping methods as it uses candidate solution objective values only in order to determine the scenario groups.

Birge [6] proposes a succession of lower bounds for two-stage stochastic linear programs. Additionally, he observes that feasible solutions can be obtained from evaluating scenario optimal first-stage solutions. Ahmed [1] proposes conditions in which the scenario optimal solutions will likely be globally optimal solutions. A recent paper by Dey et al., [11], gives theoretical bounds on the nonanticipativity relaxation of two-stage packing or covering type mixed integer linear programs based on the number of groups. However, these bounds are entirely data-independent. In [16], the authors propose a hierarchy of bounds, known as expected group subproblem objective bounds (EGSO), for two-stage stochastic MIP. This work is extended to multi-stage problems in [17] and [20]. These bounds require enumerating all scenario groups of a certain size in order to calculate the bounds. In [7], Expected Partition (EP) scenario bounds are proposed. These bounds are shown to be computationally efficient as only a partition of the scenario set into groups is required to evaluate the bound. It is noted in [12] that grouping scenarios improves the convergence of their proposed lower bounds. Scenario grouping has also been recently discussed to improve nonanticipative relaxation bounds for chance constrained problems [3]. In each of these approaches, scenario groups of a given size are either randomly constructed or are completely enumerated. Our approach seeks to intelligently group the scenarios using known information.

2 Scenario Grouping

To determine our scenario grouping strategy, we first develop a metric for estimating potential bound improvement as a result of scenario grouping. This metric uses information recovered from evaluating a given set of candidate solutions. We then describe a grouping problem which optimizes this metric subject to an imposed maximum group size constraint. An analysis of the complexity of this problem is presented for different maximum group sizes and finally a mixed integer programming model for the grouping problem is proposed.
2.1 Preliminaries

We consider two-stage stochastic programs of the form

$$\min_x \{ \mathbb{E}_P[f(x, \xi)] : x \in \mathcal{X} \} \tag{1}$$

where \(x\) is a solution vector contained in the set \(\mathcal{X}\). Here \(\xi\) is a random vector with support \(\Xi\), and the expectation is with respect to \(P\), the distribution of \(\xi\). We do not make any assumption on the objective functions \(f\). Note that in the two-stage setting, these functions take the form

$$f(\xi, x) = cx + \min \{ \phi(y(\xi), \xi) : y(\xi) \in Y(\xi, x) \},$$

where \(y\) are the second stage variables. We assume that \(\Xi\) is a finite set consisting of \(K\) scenarios.

This leads to the following finite scenario reformulation of (1)

$$z^* = \min_x \left\{ \sum_{k=1}^{K} f_k(x) : x \in \mathcal{X} \right\}, \tag{2}$$

where \(f_k(x) = p_k f(\xi_k, x)\) and \(p_k\) is the probability that scenario \(k\) is realized. We denote the set \(\{1, \ldots, K\}\) as \(\mathcal{K}\).

Formulation (2) can be rewritten using copies of the variable \(x\) as

$$z^* = \min_{x^1, \ldots, x^K} \left\{ \sum_{k=1}^{K} f_k(x^k) : x^k \in \mathcal{X} \forall k \in \mathcal{K}, \sum_{k=1}^{K} A_k x^k = 0 \right\}, \tag{3}$$

where the the nonanticipativity constraints \(\sum_{k=1}^{K} A_k x^k = 0\) force the variables to be identical across scenarios (cf. [18]). Dualizing the nonanticipativity constraints using multipliers \(\lambda\) leads to the following Nonanticipative relaxation

$$z^*(\lambda) = \sum_{k=1}^{K} z_k^*(\lambda) \text{ where } z_k^*(\lambda) = \min_x \{ f_k(x) + \lambda^\top A_k x : x \in \mathcal{X} \} \forall k \in \mathcal{K}. \tag{4}$$

Note that for any \(\lambda\), the value \(z^*(\lambda)\) is a lower bound on \(z^*\), and the dual problem of maximizing \(z^*(\lambda)\) with respect to \(\lambda\) provides the greatest such bound.

2.2 Optimal scenario grouping

We consider grouping or partitioning the set of scenarios \(\mathcal{K}\) into \(M\) groups \(\{\mathcal{K}_1, \ldots, \mathcal{K}_M\}\) such that

$$\bigcup_{m=1}^{M} \mathcal{K}_m = \mathcal{K} \text{ and } \mathcal{K}_m \cap \mathcal{K}_n = \emptyset, \forall m, n \in \mathcal{M}$$

where \(\mathcal{M} = \{1, \ldots, M\}\) and \(M < K\). With some notational abuse, we will use \(\mathcal{M}\) also to denote the collection of groups. Given a grouping \(\mathcal{M}\), we have the following equivalent reformulation of (2)

$$z^* = \min_x \left\{ \sum_{m=1}^{M} \sum_{k \in \mathcal{K}_m} f_k(x) : x \in \mathcal{X} \right\},$$
and the associated nonanticipative relaxation is

\[ z^*_M(\lambda) = \sum_{m=1}^{M} z^*_m(\lambda) \quad \text{where} \quad z^*_m(\lambda) = \min_x \left\{ \sum_{k \in \mathcal{K}_m} (f_k(x) + \lambda^\top A_k x) : x \in \mathcal{X} \right\} \quad \forall m \in \mathcal{M}. \quad (5) \]

To see that (5) is indeed a valid nonanticipative relaxation, note that if the matrices \( A_1, \ldots, A_K \) are such that \( \sum_{k=1}^{K} A_k x^k = 0 \) if and only if \( x^1 = \cdots = x^K \), then we have that \( \sum_{m=1}^{M} (\sum_{k \in \mathcal{K}_m} A_k) x^m = 0 \) if and only if \( x^1 = \cdots = x^M \). Clearly, for any given \( \lambda \), we have that

\[ z^*_m(\lambda) \geq z^*(\lambda), \]

and we would like to construct a grouping \( \mathcal{M} \) such that the bound improvement is large. To compute the bound improvement, we need to compute the values \( z_m(\lambda)^* \) for all \( m \in \mathcal{M} \). As there are exponentially many groupings, we would like to use information already known to construct an estimate of these values without solving the actual grouped problems to optimality. We can use the information discovered from evaluating candidate solutions to construct this estimate. We consider the following easy-to-see upper bounding approach.

**Proposition 1.** Given a \( \mathcal{S} \) of feasible solutions, the values \( (f_k(x) + \lambda^\top A_k x) \) for all \( k \in \mathcal{K} \) and \( x \in \mathcal{S} \), and a set of scenario optimal values \( z_k^*(\lambda) \) for all \( k \in \mathcal{K} \), we have that

\[ 0 \leq z^*_M(\lambda) - z^*(\lambda) \leq \sum_{m=1}^{M} \theta_m \]

where

\[ \theta_m = \min_{x \in \mathcal{S}} \left\{ \sum_{k \in \mathcal{K}_m} (f_k(x) + \lambda^\top A_k x - z_k^*(\lambda)) \right\}. \quad (7) \]

We let \( \theta_m = 0 \) when \( \mathcal{K}_m = \emptyset \).

We can interpret \( \theta_m \) as an optimistic prediction of the bound improvement by grouping the scenarios \( k \in \mathcal{K}_m \). To compute \( \theta_m \), the set \( \mathcal{S} \) can be the set of scenario optimal solutions or can be generated using any heuristic. Regardless of the method used to construct \( \mathcal{S} \), each solution in \( \mathcal{S} \) must be evaluated. If \( \arg\min \{ z_m^*(\lambda) \} \in \mathcal{S} \), then the predicted improvement \( \theta_m \) is equal to the actual improvement \( z^*_m(\lambda) - \sum_{k \in \mathcal{K}_m} z_k^*(\lambda) \). It follows that \( \theta_m = 0 \) if \( \mathcal{K}_m = \{k\} \) and \( \arg\min \{ z_k^*(\lambda) \} \in \mathcal{S} \).

Using the metric \( \theta_m \) to measure the quality of a grouping, we consider the following optimization problem to find a grouping \( \mathcal{M} \) that maximizes \( \sum_{m \in \mathcal{M}} \theta_m \). 

**Scenario grouping problem:** Given a finite set \( \mathcal{S} \) of feasible solutions, the values \( (f_k(x) + \lambda^\top A_k x) \) for all \( k \in \mathcal{K} \) and \( x \in \mathcal{S} \), and a set of scenario optimal values \( z_k^*(\lambda) \) for all \( k \in \mathcal{K} \), the scenario grouping problem with maximum group size \( P \leq K \) is

\[ \max_{\mathcal{K}_1, \ldots, \mathcal{K}_M} \left\{ \sum_{m=1}^{M} \theta_m : \mathcal{K}_m = \mathcal{K}, \mathcal{K}_m \cap \mathcal{K}_n = \emptyset, |\mathcal{K}_m| \leq P, \forall m, n \in \mathcal{M} \right\}, \quad (8) \]

where \( \theta_m \) is defined as in (7).

With no restriction on the number of scenarios placed into the individual groups \( (P = K) \), clearly \( \mathcal{K}_1 = \mathcal{K} \) is an optimal solution. As we seek a computationally tractable alternative to solving the extensive formulation, we will typically limit the size of the individual groups by setting \( P \ll K \).
2.3 Complexity

We provide two complexity results for the scenario grouping problem (8).

**Proposition 2.** There exists a polynomial time algorithm for the scenario grouping problem (8) when the maximum subset size $P = 2$ and the solution set size $|S|$ polynomial in $K$.

**Proof.** For every pair of scenarios $\mathcal{K}_m = \{k_1, k_2\} \subseteq \mathcal{K}$, we can precompute the value of $\theta_m$ in time proportional to $|S|(K)(K - 1)$. There are $(K)(K - 1)/2$ pairs of scenarios, with each pair requiring $|S|$ comparisons of two term sums. This is polynomial in $K$ as $|S|$ is polynomial in $K$. Using this information, we define a complete graph $G = (V, E)$ with vertices corresponding to scenarios. The weight of an edge $(k_1, k_2)$ is equal to the $\theta_m$ where $\mathcal{K}_m = \{k_1, k_2\}$. Recall that $\theta_m = 0$ when $\mathcal{K}_m = \{k\}$ for all $k \in \mathcal{K}$. It then follows that the optimal maximum weight matching on this graph corresponds to an optimal solution for (8), where nodes which have been matched represent scenarios that should be grouped together.

**Proposition 3.** The scenario grouping problem (8) is NP-hard, even for $P = 3$ and solution set size $|S| = K$.

**Proof.** We will show this by polynomially reducing an arbitrary instance of Graph Partition into Triangles, shown to be NP-complete in [13], to an instance of (8). We define an instance of Graph Partition into Triangles from [13] as

**Partition into Triangles:** Given a graph $G = (V, E)$, with $|V| = 3q$, for some integer $q$. Can the vertices of $G$ be partitioned into $q$ disjoint sets $V_1, \ldots, V_q$, each containing exactly 3 vertices, such that for each $V_i = \{u_i, v_i, w_i\}$, $1 \leq i \leq q$, all three of the edges $\{u_i, v_i\}$, $\{u_i, w_i\}$ and $\{v_i, w_i\}$ belong to $E$?

Let $\mathcal{K} = V$. Fix $P = 3$, $\lambda = 0$ and $z_k^*(\lambda) = 0$ for all $k \in \mathcal{K}$. As $|S| = K$, there will be one solution for each scenario. Let $\mathcal{S} = V$. We define the values $f_k(x^s)$ for all $k \in \mathcal{K}, x^s \in \mathcal{S}$ as follows.

- $f_k(x^s) = 1 : \{s, k\} \in E$.
- $f_k(x^s) = 0 : \{s, k\} \notin E$
- $f_k(x^s) = -K : s = k$

We now show that a partition of vertices into triangles is possible if and only if there exists a partition of scenarios with objective function value $(K/3)(-K + 2)$. First note that for some subset of scenarios $\mathcal{K}_m$, any solution $x^s$ which achieves the equality $\theta_m = \sum_{k \in \mathcal{K}_m} f_k(x^s)$ must satisfy $s \in \mathcal{K}_m$. This is clear as $f_k(x^s) = -K$ when $s = k$. So, the maximum value of $\theta_m = (-K + 2)$, exactly when the vertices in $\mathcal{K}_m$ form a triangle (clique of size 3) in the graph $G$. It follows that $\sum_{m=1}^{M} \theta_m = (K/3)(-K + 2)$ for any partition of the vertices, $\mathcal{K}_1, \ldots, \mathcal{K}_m$, which results in a triangle cover, as there must be exactly $(K/3)$ groups in any triangle cover.

With maximum group size $P = 3$, there must be at least $(K/3)$ groups in any partition. If any partition contains subgroups $\mathcal{K}_m$, with $|\mathcal{K}_m| \leq 2$, then there must be at least $(K/3 + 1)$ groups, each with $\theta_m \leq (-K + 2)$, and it follows that $\sum_{m=1}^{M} \theta_m \leq (K/3 + 1)(-K + 2) < (K/3)(-K + 2)$. If some group $\mathcal{K}_m$, with $|\mathcal{K}_m| = 3$ corresponds to vertices that are not a clique, then $\theta_m^* \leq (-K + 1)$ and it follows that $\sum_{m=1}^{M} \theta_m \leq (K/3 - 1)(-K + 2) + (-K + 1) < (K/3)(-K + 2)$. So, we have shown that any partition which does result in a partition into triangles has $\sum_{m=1}^{M} \theta_m < (K/3)(-K + 2)$. If a partition into triangles exists, the optimal value of (8) is $(K/3)(-K + 2)$, otherwise it is strictly less. 

\[ \square \]
2.4 MIP formulation

In this section we develop a mixed integer programming (MIP) formulation of the scenario grouping (8) using the variables below.

1. $y_{km}$: 0-1 variable indicating whether we place scenario $k$ into group $m$.
2. $\theta_m$: continuous variable representing estimated improvement due to set $m$

Given a finite set $S$ of feasible solutions, the values $(f_k(x) + \lambda^\top A_k x)$ for all $k \in K$ and $x \in S$, and a set of scenario optimal values $z^*_k(\lambda)$ for all $k \in K$, let

$$w_{ks} = f_k(x) + \lambda^\top A_k x - z^*_k(\lambda), \quad \forall x \in S$$

**Proposition 4.** Given a finite set of solutions $S$, values $w_{ks}$, and a maximum scenario group size $P$, a mixed integer programming formulation for the scenario grouping problem (8) is as follows:

$$V(S, P) = \max_{\theta, y} \sum_{m=1}^{M} \theta_m$$

s.t. $\theta_m \leq \sum_{k=1}^{K} w_{ks} y_{km}, \quad \forall x \in S, \forall m \in M$ (10)

$$\sum_{m=1}^{M} y_{km} = 1, \quad \forall k \in K$$ (11)

$$\sum_{k=1}^{K} y_{km} \leq P, \quad \forall m \in M$$ (12)

$$y_{km} \in \{0, 1\}, \quad \theta_m \in \mathbb{R}, \quad \forall m \in M, \forall k \in K.$$ (13)

**Proof.** First, given any solution of (8), we construct a solution of (10)-(13) of equal objective value. For each subset $K_m$, if $k \in K_m$, then $y_{km} = 1$, else $y_{km} = 0$. The constraints $\bigcup_{m=1}^{M} K_m = K$ and $K_m \cap K_n = \emptyset, \forall m, n \in M$ ensure the assignment constraint (11) holds, $|K_m| \leq P$ ensures (12) holds. The set of constraints as part of (10) model the minimum of affine functions required to compute $\theta_m$ accurately for a given solution. Next, we take a solution of (9) and create a solution of equal objective value in (8) by placing scenario $k$ into group $K_m$ if $y_{km} = 1$. Using the same arguments as above, we can see that this produces a solution in (8) of equal objective value. \[ \square \]

Note that in any feasible solution to (10)-(13), each scenario is assigned to exactly one set. This means that the value $-\left(\sum_{k=1}^{K} z^*_k(\lambda)\right)$ is included in the objective value of every feasible solution. We do not require the scenario optimal solution values in order to compute the optimal solution partition and as a result do not need to solve the individual scenario problems to optimality. However, for finite convergence of our algorithm in Section 3.2, we require the scenario optimal values to be known.

3 Algorithmic Approaches

We describe how to use the scenario grouping problem in two different ways. It can be incorporated into an existing scenario decomposition algorithm, to be solved after a round of scenario solves in order to create bound improving groups. Or it can form the basis of a finitely convergent algorithm for solving stochastic programs with a finite feasible region.
3.1 Grouping Within Scenario Decomposition

The scheme proceeds as follows.

1. Solve scenario problems, saving $z^*_k(\lambda)$ and solutions $x^k$ for each $k \in \mathcal{K}$
2. Let $\mathcal{S}$ be the set of collected candidate solutions
3. Compute $f_k(x) + \lambda^\top A_k x$ for $x \in \mathcal{S}$ and $k \in \mathcal{K}$
4. Choose maximum group size $P$, solve the MIP (9)-(13)
5. Group scenarios based on optimal partition
6. Solve grouped problems to obtain stronger bounds.

The above approach can be used as a preprocessing approach or after any round of scenario solves in a scenario decomposition algorithm. In our preprocessing approach, we solve the scenario problems independently and use the scenario optimal solutions to construct $\mathcal{S}$. This is not a requirement of the preprocessing algorithm, and any solution may be added to the set $\mathcal{S}$, as long as it has been evaluated. Note that if the values $f_k(x)$ are known for all $k \in \mathcal{K}$ for a given $x$, computing the values with the penalty term $\lambda^\top A_k x$ requires trivial computational time.

3.2 A Finite Algorithm

Here we present a Scenario Grouping Algorithm for solving stochastic programs with finite feasible region $\mathcal{X}$. In this algorithm, we fix $\lambda = 0$. We let the number of groups equal to the number of scenarios, i.e. $(M = K)$. Additionally, let $LB_{NA}$ be equal to the value of the nonanticipativity relaxation bound of (3). Denote $z^*_m = z^*_m(0)$.

**Algorithm 1 Scenario Grouping**

1. Step 0: $\mathcal{S} = \emptyset$, $\mathcal{K}_m = \{m\}$, $\forall m \in \mathcal{M}$, $UB = \infty$, $LB = -\infty$, $P = 2$
2. while $(UB > LB)$ do
3.   if $(\mathcal{S} \neq \emptyset)$ then
4.     Compute $w_{k_x}$ for all $k \in \mathcal{K}$ and $x \in \mathcal{S}$
5.     Solve the MIP (9)-(13), Record the optimal value $V(\mathcal{S}, P)$, Update $\mathcal{K}_m$ for all $m \in \mathcal{M}$
6.   end if
7.   for $m=1$ to $M$ do
8.     Compute $z^*_m$ and collect an optimal solution $x^m$
9.     $\mathcal{S} \leftarrow \mathcal{S} \cup x^m$
10. end for
11. $UB \leftarrow \min_{x \in \mathcal{S}} \left( UB, \sum_{k=1}^{K} f_k(x) \right)$
12. $LB \leftarrow \sum_{m=1}^{M} z^*_m$
13. if $(LB - LB_{NA} == V(\mathcal{S}, P))$ then
14.   $P \leftarrow P + 1$
15. end if
16. end while
Proposition 5. Algorithm 1 is finitely convergent to $z^*$

Proof. If $S = \mathcal{X}$, then we have evaluated every solution in $\mathcal{X}$ and thus have solved the problem via enumeration. Our algorithm begins with $S = \emptyset$ and $P = 2$. We next show that in Algorithm 1, for each iteration defined by the while loop on line 2, one of three things happens: we terminate with optimality, increase the size of $S$, or increase the value of $P$ (until $P = K$). Consider any given iteration of Algorithm 1 where we do not terminate with optimality. We consider two cases, $P < K$ and $P = K$.

Case 1: $P < K$ We may assume that $(LB - LB_{NA} < V(S, P))$, as otherwise we increase $P$.

If $(LB - LB_{NA} < V(S, P))$, then there must exist some partition $\mathcal{K}_m$ with objective value $z^*_m$ which satisfies

$$z^*_m - \sum_{k \in \mathcal{K}_m} z^*_k < \min_{x \in S} \left( \sum_{k \in \mathcal{K}_m} w_{ks} \right)$$

and thus there must exist a solution $x^m \notin S$ that is optimal for the partition $\mathcal{K}_m$, such that

$$\sum_{k \in \mathcal{K}_m} (f_k(x^m) - z^*_k) < \min_{x \in S} \left( \sum_{k \in \mathcal{K}_m} w_{ks} \right)$$

If the conditions above hold, this solution will be discovered and added to $S$, increasing the size of $S$.

Case 2: $P = K$

We know from the previous case that if $(LB - LB_{NA} < V(S, P))$, then we increase the size of the set $S$. Thus we may assume that $(LB - LB_{NA} = V(S, P))$ and we have not terminated with optimality. We use these assumptions to come to a contradiction. If $P = K$, then the partition $\mathcal{K}_1 = \mathcal{K}, \mathcal{K}_m = \emptyset, m \neq 1$, is feasible, and thus $V(S, P) \geq UB - LB_{NA}$. We know that

$$LB - LB_{NA} = V(S, P) \geq UB - LB_{NA}$$

and it follows that $LB \geq UB$, contradicting the assumption that we did not prove optimality. Thus we have shown that in each iteration, we either prove optimality or we increase the size of $S$ or $P$. As both the sets $S$ and $P$ are finite, this proves finite convergence. \qed

4 Computational Results

We test our scenario grouping approach as a preprocessing step before applying the scenario decomposition algorithm from [15] to two-stage stochastic programs. In this approach, we first relax the nonanticipativity constraints ($\lambda = 0$) and solve each scenario problem independently. After optimality is proven for each scenario problem or a time limit is reached, we collect scenario incumbent first-stage solutions. We then solve the scenario grouping MIP to determine scenario groups. The scenario decomposition algorithm from [15] is then applied to the grouped problem.

Five approaches are considered

1. CPLEX on the extensive formulation
2. The parallel asynchronous scenario decomposition algorithm (Asynch++) from [15]
3. Randomly grouping scenarios into pairs \((P = 2)\) as a preprocessing step followed by Asynch++

4. Our grouping scheme defined in Section 3.1 using maximum group sizes \(P = 2\) as a preprocessing step followed by Asynch++

5. Same as approach 4 above but with \(P = 4\)

Due to our choice of [15] as our scenario decomposition algorithm, we restrict our experiments to instances with 0-1 first-stage variables from SIPLIB [2]: Stochastic Supply Chain Planning Problems (SSCH) and Stochastic Multiple Knapsack Problems (SMKP). The SSCH and SMKP instances are chosen for two reasons. Individual scenario problems require nontrivial computational time to solve and the instances require multiple iterations of Asynch++ to prove optimality. Results from the SSLP instance set from [2] are not included as individual scenarios require trivial computational time to solve and require a limited number of master iterations for [15] to prove optimality. Essentially, these instances are too easy for [15] to solve and the improved bounds from grouping do not justify the extra computational time required to solve the grouping integer program.

All computations were performed on the Sierra Cluster at Lawrence Livermore National Laboratory. The cluster consists of 1,944 nodes, with nodes connected using InfiniBand Interconnect. Each individual node consists of 2 Intel 6-core Xeon X5660 processors and 24GB of memory. The asynchronous implementation was written in C using MVAPICH 2 version 1.7 for parallelism. Optimization problems were solved using CPLEX 12.5.

Each instance is solved on one node, using 24GB of memory and only 6 cores. When solving the extensive formulation, CPLEX uses 6 threads. When applied, Asynch++ uses 1 master and \((ncores - 1)\) worker processes, with each worker running single threaded CPLEX. If grouping is performed, then enough workers are idled to ensure that the number of workers is no more than the number of grouped scenarios. In keeping with the design of the asynchronous scenario decomposition algorithm from [15], additional first-stage solutions may be added to the set \(S\) from scenario resolves which are sent to workers while waiting for all scenarios to be solved to optimality.

A total time limit of 10,800 seconds (three hours) is used for all experiments. In the grouping approach, a time limit of 120 seconds is set for the initial maximum scenario solution time. If the time limit is reached, the incumbent first-stage solution for the scenario problem is added to the set \(S\). Additionally, a time limit of 120 seconds is set for the scenario grouping MIP. If the time limit is reached, then the best known feasible solution is used to create the partition. We experimented with a variety of time limits for the initial maximum scenario solution time and the scenario grouping MIP solution time. For both instance sets, time limits of 60 seconds and 240 seconds produced very similar results.

4.1 SSCH Results

The Stochastic Supply Chain (SSCH) instance set was originally proposed in [4] and can be found as part of [2]. These instances have 0-1 first-stage variables and mixed binary second-stage variables. In this set, there are 9 instances (we could not access c9 online). Each instance contains 23 scenarios and 67-78 first-stage binary variables. The second-stage problems contain 36 binary variables and approximately 3,000 continuous variables. To account for performance variability, three runs were conducted for each instance and the presented results are averages over these three runs.
<table>
<thead>
<tr>
<th>Prob</th>
<th>Opt Value</th>
<th>CPLEX</th>
<th>Asynch</th>
<th>Rand P=2</th>
<th>Part P=2</th>
<th>Part P=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>184,438</td>
<td>(16,752)</td>
<td>(16,590)</td>
<td>(4,263)</td>
<td>(2,468)</td>
<td>5,275</td>
</tr>
<tr>
<td>c2</td>
<td>0</td>
<td>43</td>
<td>(16,889)</td>
<td>(6,085)</td>
<td>276</td>
<td>90</td>
</tr>
<tr>
<td>c3</td>
<td>230,268</td>
<td>(15,949)</td>
<td>(15,806)</td>
<td>(2,999)</td>
<td>(666)</td>
<td>1,301</td>
</tr>
<tr>
<td>c4</td>
<td>201,454</td>
<td>(17,003)</td>
<td>(13,529)</td>
<td>(728)</td>
<td>(692)</td>
<td>247</td>
</tr>
<tr>
<td>c5</td>
<td>0</td>
<td>2,039</td>
<td>(8,215)</td>
<td>228</td>
<td>247</td>
<td>298</td>
</tr>
<tr>
<td>c6</td>
<td>231,369</td>
<td>(13,466)</td>
<td>(12,611)</td>
<td>(742)</td>
<td>9,910</td>
<td>117</td>
</tr>
<tr>
<td>c7</td>
<td>140,180</td>
<td>164</td>
<td>(10,418)</td>
<td>(770)</td>
<td>4,504</td>
<td>65</td>
</tr>
<tr>
<td>c8</td>
<td>100,523</td>
<td>(13,270)</td>
<td>(5,803)</td>
<td>(366)</td>
<td>2,392</td>
<td>5,342</td>
</tr>
<tr>
<td>c10</td>
<td>139,739</td>
<td>(1,066)</td>
<td>(2,670)</td>
<td>25</td>
<td>52</td>
<td>72</td>
</tr>
</tbody>
</table>

Table 1: SSCH: Solution Time/(Absolute Gap at Three Hours)

<table>
<thead>
<tr>
<th>Prob</th>
<th>Non-Ant Gap</th>
<th>% of NonAnt Gap Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Rand (P=2)</td>
</tr>
<tr>
<td>c1</td>
<td>18,145</td>
<td>70%</td>
</tr>
<tr>
<td>c2</td>
<td>19,448</td>
<td>58%</td>
</tr>
<tr>
<td>c3</td>
<td>20,928</td>
<td>68%</td>
</tr>
<tr>
<td>c4</td>
<td>15,627</td>
<td>77%</td>
</tr>
<tr>
<td>c5</td>
<td>9,618</td>
<td>100%</td>
</tr>
<tr>
<td>c6</td>
<td>18,082</td>
<td>78%</td>
</tr>
<tr>
<td>c7</td>
<td>13,762</td>
<td>70%</td>
</tr>
<tr>
<td>c8</td>
<td>7,132</td>
<td>82%</td>
</tr>
<tr>
<td>c10</td>
<td>5,667</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 2: SSCH: Nonanticipativity Gap Closed

<table>
<thead>
<tr>
<th>Prob</th>
<th>Non-Ant Gap</th>
<th>Pred (P=2)</th>
<th>Act (P=2)</th>
<th>Pred (P=4)</th>
<th>Act (P=4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>18,145</td>
<td>14,770</td>
<td>14,735</td>
<td>18,145</td>
<td>17,694</td>
</tr>
<tr>
<td>c2</td>
<td>19,448</td>
<td>19,082</td>
<td>19,082</td>
<td>19,448</td>
<td>19,448</td>
</tr>
<tr>
<td>c3</td>
<td>20,928</td>
<td>16,951</td>
<td>16,885</td>
<td>20,928</td>
<td>20,674</td>
</tr>
<tr>
<td>c4</td>
<td>15,627</td>
<td>12,529</td>
<td>12,549</td>
<td>15,627</td>
<td>15,627</td>
</tr>
<tr>
<td>c5</td>
<td>9,618</td>
<td>9,618</td>
<td>9,618</td>
<td>9,618</td>
<td>9,618</td>
</tr>
<tr>
<td>c6</td>
<td>18,082</td>
<td>14,998</td>
<td>14,998</td>
<td>18,082</td>
<td>18,082</td>
</tr>
<tr>
<td>c7</td>
<td>13,762</td>
<td>11,499</td>
<td>11,497</td>
<td>13,762</td>
<td>13,762</td>
</tr>
<tr>
<td>c8</td>
<td>7,132</td>
<td>7,132</td>
<td>7,095</td>
<td>7,132</td>
<td>6,944</td>
</tr>
<tr>
<td>c10</td>
<td>5,667</td>
<td>5,667</td>
<td>5,667</td>
<td>5,667</td>
<td>5,667</td>
</tr>
</tbody>
</table>

Table 3: SSCH: Prediction Accuracy
In Table 1, we compare times to solve the instances to optimality. For our proposed grouping approach, the total time required includes; time required to generate the set \(S\), solve the grouping problem (9) and the time required to solve the problem using Asynch++ on the grouped problem. If optimality is not proven by the time limit of three hours, we present the absolute gap in parenthesis, (Absgap), of the algorithm at three hours. Previously unsolved instances are marked in bold.

Random grouping, while improving the initial relaxation bound, only solves three out of the nine instances to optimality within the time limit. These three instances were previously solved. Our grouping approach with \(P = 2\) is able to solve 6 instances, including 1 previously unsolved instance. For the instances which are not solved to optimality, the final absolute gap is stronger than the bound provided by random grouping. Our grouping approach with maximum set size \(P = 4\) is able to solve all of the instances to optimality within the three hour time limit, and solves all four previously unsolved instances to optimality.

In Table 2, we compare the lower bound produced by the nonanticipativity relaxation with the lower bound generated by each grouping approach. The value of the nonanticipativity gap is computed using the best known feasible solution value and the nonanticipativity bound. The gap closed columns represent the percentage of the nonanticipativity gap closed by each grouping approach.

Randomly grouping does close a large percentage of the nonanticipativity gap. However, we see that the grouping with maximum group size \(P = 2\) using the scenario grouping MIP closes a larger percentage of the gap than random grouping in every instance and more than 80% of the nonanticipativity gap for every instance. Increasing the maximum group size to \(P = 4\) closes more than 97% of the nonanticipativity gap for every instance.

In Table 3 we compare the predicted bound improvement given by the optimal value of the grouping problem (9), \(V(S, P)\), with the actual improvement in the lower bound due to grouping. The first column labeled ‘Non-Ant Gap’ displays the value of the absolute gap computed using the best known feasible solution value and the nonanticipativity bound. The remaining four columns compare the predicted and actual gap improvements from our grouping approaches with maximum group sizes \(P = 2\) and \(P = 4\). Solutions with a * indicate that the partition problem was not solved to optimality, instead terminating at the time limit.

For this instance set, the improvement generated from the scenario grouping MIP is exactly predicted by the value \(V(S, P)\) for 6 instances when \(P = 2\) and 6 instances when \(P = 4\). When the prediction is not exact, it is within 3% of the actual improvement for all instances, regardless of maximum group size \(P\).

### 4.2 SMKP Results

The Stochastic Multiple Knapsack (SMKP) instance set was originally proposed in [5] and can be found as part of [2]. Each of the thirty instances contains 240 binary first-stage variables and 20 scenarios, with each scenario containing 120 binary second-stage variables. There are 50 first-stage only constraints and 5 second-stage constraints per scenario. We divide the instances into 5 groups, based on the number of approaches which solve the instances to optimality. Instances (1-7) are labelled ‘Easy’, (9-10) as ‘Medium’, (8,11,13,16,18) as ‘Hard’ and (12, 14-15,17,19-29) as ‘Very Hard’. None of five approaches finds a lower bound within the defined time limit and given optimality tolerance for smkp30, and so it is omitted. As before, the time to solve an instance is the average of three independent runs. Solution times and absolute gaps for a given group of
problems, such as the 'Easy' group, are computed using these averages which are then averaged over all instances in the group.

This instance set is challenging for scenario decomposition approaches when compared to stagewise decomposition approaches as individual scenarios take significant time to solve. As individual scenarios are challenging to solve, scenario groups which include more than two scenarios require excessive computational effort for this instance set. We do not test our grouping approach using the scenario grouping MIP with $P = 4$ as a preprocessing step for this set of instances due to these challenges.

Table 4 presents the solutions times and follows the same conventions as Table 1. Using our scenario grouping MIP to perform grouping, we are able to solve 14 of the instances to optimality in three hours, as opposed to 9 using random grouping and 7 using our original scenario decomposition method. For instances which no method is able to prove optimality, our partition based grouping terminates with a better optimality gap than any of the other approaches. This is likely due to the stronger initial relaxation.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Opt Value</th>
<th>(Absgap)/Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CPLEX</td>
</tr>
<tr>
<td>Easy (7 instances)</td>
<td>9,145.67</td>
<td>(12.68)</td>
</tr>
<tr>
<td>Medium (2 instances)</td>
<td>9,278.20</td>
<td>(21.59)</td>
</tr>
<tr>
<td>Hard (5 instances)</td>
<td>9,128.21</td>
<td>(14.19)</td>
</tr>
<tr>
<td>Very Hard (15 instances)</td>
<td>9,432.68</td>
<td>(16.77)</td>
</tr>
</tbody>
</table>

Table 4: SMKP: Solution Time/(Absolute Gap at Three Hours)

<table>
<thead>
<tr>
<th>Prob</th>
<th>Non-Ant Gap</th>
<th>% of Non-Ant Gap Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rand (P=2)</td>
<td>Part (P=2)</td>
</tr>
<tr>
<td>Easy (7 instances)</td>
<td>5.31</td>
<td>58 %</td>
</tr>
<tr>
<td>Medium (2 instances)</td>
<td>9.68</td>
<td>53 %</td>
</tr>
<tr>
<td>Hard (5 instances)</td>
<td>11.70</td>
<td>41 %</td>
</tr>
<tr>
<td>Very Hard (15 instances)</td>
<td>14.23</td>
<td>40 %</td>
</tr>
</tbody>
</table>

Table 5: SMKP: Nonanticipativity Gap Closed

<table>
<thead>
<tr>
<th>Prob</th>
<th>Non-Ant Gap</th>
<th>Pred (P=2)</th>
<th>Act (P=2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy (7 instances)</td>
<td>5.31</td>
<td>5.3</td>
<td>4.4</td>
</tr>
<tr>
<td>Medium (2 instances)</td>
<td>9.68</td>
<td>8.8</td>
<td>7.4</td>
</tr>
<tr>
<td>Hard (5 instances)</td>
<td>11.70</td>
<td>10.9</td>
<td>9.0</td>
</tr>
<tr>
<td>Very Hard (15 instances)</td>
<td>14.23</td>
<td>12.6</td>
<td>8.5</td>
</tr>
</tbody>
</table>

Table 6: SMKP: Prediction Accuracy
Table 5 displays the percentage of the original nonanticipativity gap closed by the different grouping approaches and follows the same conventions as Table 2. We see that for all classes of instances, grouping based on the scenario grouping MIP provides a stronger bound than randomly grouping. As each scenario solve is very time consuming, this tighter bound improves overall solution time as it reduces the total number of scenario solves required before proving optimality.

Table 6 displays the prediction accuracy of the scenario grouping MIP and follows the same conventions as Table 3. For this instance set, most of the gap is predicted to be closed and the prediction is within 25% of the actual gap closed for Easy, Medium and Hard instances and within 50% of the actual gap closed for the Very Hard instances.

5 Conclusion

In this work, we propose an optimization driven scenario grouping technique for stochastic programs that leads to stronger nonanticipative relaxation bounds. We show that the grouping problem is NP-Hard in general and give conditions for which a polynomial time algorithm exists. We develop a mixed integer formulation of the grouping problem and show how to incorporate this scheme into any general scenario decomposition scheme. Finally, we propose a finitely convergent algorithm for solving two-stage stochastic integer programs with a finite first-stage feasible region. Our approach is simple to implement and provides significantly stronger initial relaxation bounds when compared with random grouping. We believe that this new approach to scenario grouping can yield significant improvements and should be considered anytime scenario decomposition algorithms are used. Extension of the proposed approach to scenario decomposition of risk averse [10] and chance constrained stochastic programs [3] could be an interesting direction of research.

Acknowledgements

This research has been supported in part by the Office of Naval Research Grants N00014-15-1-2078 and 127307. Some of this work is performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344.

References


