Optimization Driven Scenario Grouping

Kevin Ryan[1], Shabbir Ahmed[1], Santanu S. Dey[1], and Deepak Rajan[2]

[1]School of Industrial & Systems Engineering, Georgia Institute of Technology
[2]Lawrence Livermore National Laboratory

May 4, 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, 15, 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 *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 [16]. 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, either discovering new candidate solutions, increasing maximum group size 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 [20], 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 [17], 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 [18] and [21]. 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 to evaluate as only a partition of the scenario set into groups is required to obtain this 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 ξ is a random vector with support Ξ, and the expectation is with respect to P, the distribution of ξ. 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 Ξ 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. [19]). Dualizing the nonanticipativity constraints using multipliers λ 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 λ, the value $z^*(\lambda)$ is a lower bound on z^*, and the dual problem of maximizing $z^*(\lambda)$ with respect to λ 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

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

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

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

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 λ, we have that

$$z^*_{\mathcal{M}}(\lambda) \geq z^*(\lambda),$$

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 straightforward upper bounding approach.

Proposition 1. *Given a 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 $k \in \mathcal{K}$, we have that*

$$0 \leq z^*_{\mathcal{M}}(\lambda) - z^*(\lambda) \leq \sum_{m=1}^{M} \theta_m \quad (6)$$

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 θ_m as an optimistic prediction of the bound improvement by grouping the scenarios $k \in \mathcal{K}_m$. To compute θ_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 $\operatorname{argmin}\{z^*_m(\lambda)\} \in \mathcal{S}$, then the predicted improvement θ_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 $\operatorname{argmin}\{z^*_k(\lambda)\} \in \mathcal{S}$.

Using the metric θ_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
>
> $$\max_{\mathcal{K}_1, \ldots, \mathcal{K}_M} \left\{ \sum_{m=1}^{M} \theta_m : \bigcup_{m=1}^{M} \mathcal{K}_m = \mathcal{K}, \mathcal{K}_m \cap K_n = \emptyset, |\mathcal{K}_m| \leq P, \forall m, n \in \mathcal{M} \right\}, \quad (8)$$
>
> where θ_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 group size* $P = 2$.

Proof. As every feasible solution to (8) includes in its objective value the term $-\sum_{k=1}^{K} z_k^*(\lambda)$, the optimal solution to (8) does not depend on the scenario optimal values, $z_k^*(\lambda)$. Thus, we may assume that $z_k^*(\lambda) = \min_{x \in \mathcal{S}} \{f_k(x) + \lambda^\top A_k x\}$ for every scenario. As a result, singleton groups have $\theta_m = 0$. For every pair of scenarios $\mathcal{K}_m = \{k_1, k_2\} \subseteq \mathcal{K}$, we can precompute the value of θ_m in time proportional to $|\mathcal{S}|(K)(K-1)$. There are $(K)(K-1)/2$ pairs of scenarios, with each pair requiring $|\mathcal{S}|$ comparisons of two term sums. 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 θ_m where $\mathcal{K}_m = \{k_1, k_2\}$. 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* $|\mathcal{S}| = K$.

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 $|\mathcal{S}| = K$, 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. □

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. θ_m: continuous variable representing estimated improvement due to group m.

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 $k \in \mathcal{K}$, let

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

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

$$V(\mathcal{S}, P) = \max_{\theta, y} \quad \sum_{m=1}^{M} \theta_m \tag{9}$$

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

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

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

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

Proof. First, given any solution of (8), we construct a solution of (10)-(13) of equal objective value. For each subset $\mathcal{K}_m$, if $k \in \mathcal{K}_m$, then $y_{km} = 1$, else $y_{km} = 0$. The constraints $\bigcup_{m=1}^{M} \mathcal{K}_m = \mathcal{K}$ and $\mathcal{K}_m \cap K_n = \emptyset$, $\forall m, n \in \mathcal{M}$ ensure the assignment constraint (11) holds, $|\mathcal{K}_m| \leq P$ ensures (12) holds. The set of constraints as part of (10) model the minimum of affine functions required to compute θ_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 $\mathcal{K}_m$ if $y_{km} = 1$. Using the same arguments as above, we can see that this produces a solution in (8) of equal objective value. □

Note that in any feasible solution to (10)-(13), each scenario is assigned to exactly one set. This means that the value $-(\sum_{k=1}^{K} z_k^*(\lambda))$ 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: $\mathcal{S} = \emptyset$, $K_m = \{m\}$, $\forall m \in \mathcal{M}$, $UB = \infty$, $LB = -\infty$, $P = 2$
2: **while** $(UB > LB)$ **do**
3: **if** $(\mathcal{S} \neq \emptyset)$ **then**
4: Compute w_{ks} 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 $\mathcal{K}_m \forall m \in \mathcal{M}$
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 $\mathcal{S} = \mathcal{X}$, then we have evaluated every solution in $\mathcal{X}$ and thus have solved the problem via enumeration. Our algorithm begins with $\mathcal{S} = \emptyset$ 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 $\mathcal{S}$, 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 $(LB - LB_{NA} < V(\mathcal{S}, P))$, as otherwise we increase P.

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

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

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

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

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

Case 2: P = K

We know from the previous case that if $(LB - LB_{NA} < V(\mathcal{S}, P))$, then we increase the size of the set $\mathcal{S}$. Thus we may assume that $(LB - LB_{NA} = V(\mathcal{S}, P))$ 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 $V(\mathcal{S}, P) \geq UB - LB_{NA}$. We know that

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

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 $\mathcal{S}$ or P. As both the sets $\mathcal{S}$ and P are finite, this proves finite convergence. □

4 Computational Results

We test our scenario grouping approach as a preprocessing step before applying the scenario decomposition algorithm from [16] 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, 'Asynch++' from [16] is then applied to these scenario groups. For the reminder of the paper we refer to 'Asynch++' as 'Asynch'.

Six approaches are considered

1. CPLEX on the extensive formulation
2. The parallel asynchronous scenario decomposition algorithm (Asynch) from [16]
3.,4. Randomly grouping scenarios into groups of size at most two or four as a preprocessing step followed by Asynch
5.,6. Our grouping scheme defined in Section 3.1 using groups of size two or four as a preprocessing step followed by Asynch

Due to our choice of Asynch from [16] as our scenario decomposition algorithm, we restrict our experiments to two-stage stochastic programs with 0-1 first-stage variables from SIPLIB [2]: Stochastic Supply Chain Planning Problems (SSCH), Stochastic Server Location Problems (SSLP) and Stochastic Multiple Knapsack Problems (SMKP).

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. To account for performance variability, five runs were conducted with different random seeds per run for each instance and the presented results are averages over these five runs.

Each instance is solved on one node, using 24GB of memory and only six cores. When solving the extensive formulation, CPLEX uses six threads. When applied, Asynch uses one master and five 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 [16], additional first-stage solutions may be added to the set $\mathcal{S}$ 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 $\mathcal{S}$. 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. The scenario grouping problem is given a random grouping as an initial feasible solution in order to ensure a feasible grouping result. 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.

When the maximum group size is set to two, an alternate integer program is used to determine the partition, following the methodology described in Proposition 2 of Section 2.3. This integer program uses precomputed edge weights and ensures a proper matching through integrality requirements. This approach was chosen over the blossom algorithm for general matching as it was trivial to implement and solved very quickly.

4.1 SSCH Results

The Stochastic Supply Chain (SSCH) instance set was originally proposed in [4] and can be found as part of SIPLIB. These instances have 0-1 first-stage variables and mixed binary second-stage variables. In this set, there are nine 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.

In Tables 1 and 2, we compare times to solve the instances to optimality as well as average absolute gap after three hours. For our proposed grouping approach, the total time required includes; time required to generate the set $\mathcal{S}$, 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 record the absolute gap of the algorithm at three hours. Previously unsolved instances are marked in bold. In Table 1, the number in parenthesis represents the number of runs solved to optimality out of five.

Random grouping with maximum group size two, while improving the initial relaxation bound, is only able to solve four out of the nine instances to optimality within the time limit and is unable to prove optimality in all five runs for any given instance. These four instances were previously solved. Our grouping approach with maximum group size of two is able to prove optimality for six instances, including one previously unsolved instance. For the instances which are not solved to optimality, the final absolute gap from our grouping approach is stronger than the final gap from random grouping. Random grouping with maximum group size four is able to prove optimality for four instances, but again is unable to prove optimality in all five runs for any instance. Our grouping approach with maximum set size four is able to solve all of the instances for all runs to optimality within the three hour time limit, and solves all four previously unsolved instances to optimality.

Table 1: SSCH: Solution Time

	Seconds (instances solved/5)					
Prob	CPLEX	Asynch	Rand P=2	Part P=2	Rand P=4	Part P=4
c1	**- (0)**	**- (0)**	**- (0)**	**- (0)**	**- (0)**	**2,683 (5)**
c2	41 (5)	- (0)	422 (1)	182 (5)	- (0)	150 (5)
c3	**- (0)**	**- (0)**	**- (0)**	**- (0)**	**3,067 (1)**	**723 (5)**
c4	**- (0)**	**- (0)**	**- (0)**	**- (0)**	**- (0)**	**2,036 (5)**
c5	2,029 (5)	- (0)	228 (1)	256 (5)	- (0)	523 (5)
c6	**- (0)**	**- (0)**	**- (0)**	**7,928 (4)**	**2,154 (1)**	**237 (5)**
c7	181 (5)	- (0)	- (0)	5,222 (5)	- (0)	68 (5)
c8	- (0)	- (0)	3,111 (1)	3,780 (4)	3,111 (1)	4,747 (5)
c10	- (0)	- (0)	273 (3)	78 (5)	84 (3)	73 (5)

Table 2: SSCH:Absolute Gap at Three Hours

		Average AbsGap at Three Hours					
Prob	OptVal	CPLEX	Asynch	Rand P=2	Part P=2	Rand P=4	Part P=4
c1	**184,438**	**17,420**	**16,681**	**6,847**	**2,492**	**2,846**	**0**
c2	0	0	16,997	5,414	0	2,153	0
c3	**230,268**	**14,276**	**16,301**	**6,427**	**1,504**	**2,743**	**0**
c4	**201,454**	**16,571**	**13,606**	**5,778**	**732**	**2,681**	**0**
c5	0	0	8,218	3,955	0	2,082	0
c6	**231,369**	**12,606**	**14,667**	**4,661**	**336**	**1,664**	**0**
c7	140,180	202	10,890	3,865	0	2,129	0
c8	100,523	13,788	5,786	1,636	8	322	0
c10	139,739	1,979	3,212	749	0	212	0

Table 3: SSCH: Nonanticipativity Gap Closed

		% of NonAnt Gap Closed			
Prob	NonAnt Gap	Rand P=2	Part P=2	Rand P=4	Part P=4
c1	**18,145**	**58%**	**81%**	**82%**	**99%**
c2	19,448	65%	98%	73%	100%
c3	**20,928**	**56%**	**80%**	**80%**	**99%**
c4	**15,627**	**49%**	**80%**	**76%**	**99%**
c5	9,618	49%	100%	71%	100%
c6	**18,082**	**57%**	**83%**	**79%**	**100%**
c7	13,762	54%	84%	69%	100%
c8	7,132	68%	96%	70%	99%
c10	5,667	70%	84%	89%	100%

Table 4: SSCH: Prediction Accuracy

Prob	NonAnt Gap	Pred P=2	Act P=2	Pred P=4	Act P=4
c1	**18,145**	**14,770**	**14,734**	**18,145**	**18,036**
c2	19,448	19,082	19,082	19,448	19,448
c3	**20,928**	**16,951**	**16,771**	**20,928**	**20,701**
c4	**15,627**	**12,529**	**12,548**	**15,627**	**15,416**
c5	9,618	9,618	9,618	9,618	9,618
c6	**18,082**	**15,024**	**15,017**	**18,082**	**18,082**
c7	13,762	11,499	11,498	13,762	13,762
c8	7,132	7,132	6,810	7,132	7,032
c10	5,667	5,667	4,732	5,667	5,667

In Table 3, 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 remaining 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 our grouping approach with maximum group size of two 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 four closes more than 98% of the nonanticipativity gap for every instance and again significantly improves over random grouping.

In Table 4 we compare the predicted bound improvement given by the optimal value of the grouping problem (9), $V(\mathcal{S}, P)$, 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 $P = 4$.

For this instance set, the improvement generated from the scenario grouping MIP is exactly predicted by the value $V(\mathcal{S}, P)$ for two instances when the maximum group size is two and five instances when it is four. When the prediction is not exact, it is within 17% of the actual improvement for all instances, regardless of maximum group size.

4.2 SMKP Results

The Stochastic Multiple Knapsack (SMKP) instance set was originally proposed in [5] and can be found as part of SIPLIB. 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 five groups, roughly 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 the six 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 five 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. The number of instances solved is a total over all runs for 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 any grouping approach with maximum group size four as a preprocessing step for this set of instances due to these challenges.

Table 5 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 fourteen of the instances to optimality in three hours, as opposed to eleven using random grouping and seven 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 5: SMKP: Solution Time/(Absolute Gap at Three Hours)

		Seconds (instances solved)			
Problem	OptVal	CPLEX	Asynch	Rand (P=2)	Part (P=2)
Easy (7 inst)	9,145.67	- (0/35)	4,678 (35/35)	2,395 (35/35)	2,177 (35/35)
Medium (2 inst)	9,278.20	- (0/10)	- (0/10)	5,734 (10/10)	4,507 (10/10)
Hard (5 inst)	9,128.21	- (0/25)	- (0/25)	9,548 (4/25)	7,271 (23/25)
Very Hard (15 inst)	9,432.68	- (0/75)	- (0/75)	- (0/75)	- (0/75)

Table 6: SMKP: Absolute Gap at Three Hours

		Average AbsGap at Three Hours			
Problem	OptVal	CPLEX	Asynch	Rand (P=2)	Part (P=2)
Easy (7 inst)	9,145.67	12.71	0	0	0
Medium (2 inst)	9,278.20	20.90	2.18	0	0
Hard (5 inst)	9,128.21	15.44	8.43	2.71	0
Very Hard (15 inst)	9,432.68	16.40	11.96	7.62	4.88

Table 7: SMKP: Nonanticipativity Gap Closed

		% of NonAnt Gap Closed	
Prob	Non-Ant Gap	Rand (P=2)	Part (P=2)
Easy (7 inst)	5.31	55%	82%
Medium (2 inst)	9.68	49%	76%
Hard (5 inst)	11.70	42%	78%
Very Hard (15 inst)	13.78	37%	60%

Table 8: SMKP: Prediction Accuracy

Prob	Non-Ant Gap	Pred (P=2)	Act (P=2)
Easy (7 inst)	5.31	5.30	4.34
Medium (2 inst)	9.68	8.77	7.36
Hard (5 inst)	11.70	10.86	9.16
Very Hard (15 inst)	13.78	12.57	8.24

Table 7 displays the percentage of the original nonanticipativity gap closed by the different grouping approaches and follows the same conventions as Table 3. 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 8 displays the prediction accuracy of the scenario grouping MIP and follows the same conventions as Table 4. 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.

4.3 SSLP Results

The Stochastic Server Location Problem is described in detail in [14]. It is a common benchmark problem for 0-1 stochastic programs. In this problem set, the first stage 0-1 variables represent placement of servers at given locations. The uncertainty is in potential customer demand for the service. Second stage recourse variables represent a decision to serve a particular customer from a given server or to not serve the customer at all. Instances can be found as part of SIPLIB and are written in the form $n_1\_n_2\_K$, where n_1 is the number of first stage variables, n_2 is the number of second stage variables per scenario and K is the number of scenarios. We combine the solution time and absolute gap tables for these instances as only a few CPLEX runs are unable to prove optimality. Results presented are solution time in seconds with the exception of certain CPLEX runs, where absolute gap at three hours is presented in parenthesis. For the sets of instances which CPLEX solves a fraction of the runs to optimality, we report solution time only and omit information regarding absolute gap at three hours.

We see in Table 10 that the scenario grouping approach is able to close more of the nonanticipativity gap than random grouping in every instance. However, the overhead required for computing the optimal partition leads to increased computational time for the scenario grouping approach when compared with the random grouping approach. The overhead involved includes the additional time required to evaluate first-stage solutions, as optimality based shortcuts to terminate first-stage solution evaluation early cannot be employed. Overhead also includes the time required to solve the optimal scenario grouping problem. Additionally, the SSLP instance set has few first-stage feasible solutions and thus it is easy for the Asynch algorithm to improve the bound via no-good cuts. These disadvantages and the computational benefits demonstrated on the SSCH and SMKP instance sets suggest that the scenario grouping approach we propose is best used for instances where it is challenging to improve upon the nonanticipativity bound using traditional approaches for stochastic programs.

For the scenario grouping approach using the integer programming formulation (9) with group size four, the 10_50_2000 instance cannot be solved due to insufficient memory. The same instance with group size two can be solved as the matching based formulation requires significantly less memory to solve. Additionally, the integer programming formulation (9) with group size four is unable to improve beyond the random bunching solution for all runs of instances 10_50_500 and 10_50_1000. This suggests that for instances with a large number of scenarios and group size greater than two an approximation of the partitioning problem (8) may be required.

Table 9: SSLP: Solution Time

		Seconds/(AbsGap) (instances solved/5)					
Prob	OptVal	CPLEX	Asynch	Rand P=2	Part P=2	Rand P=4	Part P=4
10_50_50	364.64	98 (5)	8 (5)	11 (5)	18 (5)	12 (5)	18 (5)
10_50_100	345.19	4,222 (3)	15 (5)	16 (5)	29 (5)	21 (5)	136 (5)
10_50_500	349.14	(0.65) (0)	61 (5)	56 (5)	171 (5)	76 (5)	303 (5)
10_50_1000	351.71	(0.74) (0)	117 (5)	107 (5)	444 (5)	152 (5)	614 (5)
10_50_2000	347.26	(0.69) (0)	252 (5)	206 (5)	1,294 (5)	630 (5)	OOM*
		Seconds/(AbsGap) (instances solved/25)					
15_45_15(a-e)	230.56	1,932 (25)	230 (25)	88 (25)	61 (25)	52 (25)	71 (25)
15_45_20(a-e)	246.81	3,930 (16)	73 (25)	47 (25)	55 (25)	146 (25)	56 (25)
15_45_25(a-e)	246.82	3,833 (16)	106 (25)	67 (25)	87 (25)	232 (25)	162 (25)

Table 10: SSLP: Nonanticipativity Gap Closed

		% of NonAnt Gap Closed			
Prob	NonAnt Gap	Rand P=2	Part P=2	Rand P=4	Part P=4
10_50_50	14.3	53%	87%	83%	99%
10_50_100	17.2	49%	87%	76%	99%
10_50_500	16.5	50%	92%	78%	80%**
10_50_1000	15.6	51%	94%	83%	80%**
10_50_2000	15.6	54%	67%**	82%	OOM*
15_45_15(a-e)	19.7	50%	82%	84%	97%
15_45_20(a-e)	19.5	50%	85%	82%	99%
15_45_25(a-e)	20.1	47%	82%	71%	99%

$*$: the grouping problem ran out of memory before it could evaluate a feasible solution or determine a bound.
$**$: the grouping problem did not improve over the random bunching solution MIPstart in at least one run

Table 11: SSLP: Prediction Accuracy

Prob	NonAnt Gap	Pred P=2	Act P=2)	Pred P=4	Act P=4
10_50_50	14.3	13.2	12.5	14.3	14.3
10_50_100	17.2	15.2	14.9	17.0	17.0
10_50_500	16.5	15.2	15.2	12.9	12.9
10_50_1000	15.6	14.6	14.6	12.3	12.3
10_50_2000	15.6	10.5	10.5	OOM*	OOM*
15_45_15(a-e)	19.7	16.6	15.9	19.7	19.1
15_45_20(a-e)	19.5	17.2	16.5	19.9	19.3
15_45_25(a-e)	20.1	16.8	16.5	20.1	20.0

$*$: the grouping problem ran out of memory before it could evaluate a feasible solution or determine a bound.

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 grouping approach into any general scenario decomposition scheme. When scenario grouping is used as a preprocessing step, the combined scheme is able to solve four previously unsolved standard test instances. 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

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

[2] AHMED, S., GARCIA, R., KONG, N., NTAIMO, L., PARIJA, G., QIU, F., AND SEN, S. A stochastic integer programming test library. `http://www2.isye.gatech.edu/~sahmed/siplib/`, 2015.

[3] AHMED, S., LUTEDKE, J., SONG, Y., AND XIE, W. Nonanticipative duality, relaxations, and formulations for chance-constrained stochastic programs. *Optimization Online* (2015).

[4] ALONSO-AYUSO, A., ESCUDERO, L., GARIN, A., ORTUNO, M., AND PEREZ, G. An approach for strategic supply chain planning under uncertainty based on stochastic 0-1 programming. *Journal of Global Optimization 26*, 1 (2003), 97–124.

[5] ANGULO, G., AHMED, S., AND DEY, S. Improving the integer L-shaped method. *INFORMS Journal on Computing* (2016). To appear.

[6] BIRGE, J. R. The value of the stochastic solution in stochastic linear programs with fixed recourse. *Mathematical Programming 24*, 1, 314–325.

[7] BOLAND, N., BAKIR, I., DANDURAND, B., AND ERERA, A. Scenario set partition dual bounds for multistage stochastic programming: A hierarchy of bounds and a partition sampling approach. *Optimization Online* (2016).

[8] CAROE, C. C., AND SCHULTZ, R. Dual decomposition in stochastic integer programming. *Operations Research Letters 24* (1999), 37–45.

[9] CRAINIC, T. G., HEWITT, M., AND REI, W. Scenario grouping in a progressive hedging-based meta-heuristic for stochastic network design. *Computers and Operations Research 43* (2014), 90 – 99.

[10] DENG, Y., AHMED, S., AND SHEN, S. Parallel scenario decomposition of risk averse 0-1 stochastic programs. *Optimization Online* (2016).

[11] DEY, S. S., MOLINARO, M., AND WANG, Q. Analysis of sparse cutting-plane for sparse ips with applications to stochastic ips. *arXiv:1601.00198* (2016).

[12] GADE, D., HACKEBEIL, G., RYAN, S. M., WATSON, J.-P., WETS, R. J. B., AND WOODRUFF, D. L. Obtaining lower bound from the progressive hedging algorithm for stochastic mixed-integer programs.

[13] GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[14] NTAIMO, L., AND SEN, S. The million-variable march for stochastic combinatorial optimization. *Journal of Global Optimization 32*, 3 (2005), 385–400.

[15] ROCKAFELLAR, R. T., AND WETS, R. J.-B. W. Scenario and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research 16* (1991), 119–147.

[16] RYAN, K., RAJAN, D., AND AHMED, S. Scenario decomposition for 0-1 stochastic programs: Improvements and asynchronous implementation. In *Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2016 IEEE International* (May 2016). To appear.

[17] SANDIKÇI, B., KONG, N., AND SCHAEFER, A. J. A hierarchy of bounds for stochastic mixed-integer programs. *Mathematical Programming 138*, 1 (2012), 253–272.

[18] SANDIKÇI, B., AND OZALTIN, O. A scalable bounding method for multi-stage stochastic integer programs. *Optimization Online* (2014).

[19] SHAPIRO, A., DENTCHEVA, D., AND RUSZCZYNSKI, A. *Lectures on Stochastic Programming: Modeling and Theory*, second ed. SIAM, 2014.

[20] SONG, Y., AND LUEDTKE, J. An adaptive partition-based approach for solving two-stage stochastic programs with fixed recourse. *SIAM Journal on Optimization 25*, 3 (2015), 1344–1367.

[21] ZENAROSA, G. L., PROKOPYEV, O. A., AND SCHAEFER, A. J. Scenario-tree decomposition: Bounds for multistage stochastic mixed-integer programs. *Optimization Online* (2014).