Split Cuts for Stochastic Integer Programming

Merve Bodur[1], Sanjeeb Dash[2], Oktay Günlük[2], and James Luedtke[1]

[1] University of Wisconsin-Madison, Madison, WI, USA
[2] IBM T.J. Watson Research Center, Yorktown Heights, NY, USA

Abstract. With stochastic integer programming as the motivating application, we investigate techniques to use integrality information to obtain improved cuts within a Benders decomposition algorithm. We consider two options: (i) cut-and-project, where integrality information is used to derive cuts in the extended variable space, and Benders cuts are then used to project the resulting improved relaxation, and (ii) project-and-cut, where integrality information is used to derive cuts directly in the projected space defined by Benders cuts. We analyze the use of split cuts in these two approaches, and demonstrate that although they yield equivalent relaxations when considering a single split, cut-and-project yields stronger relaxations in general when using multiple splits. Computational results illustrate that the difference can be very large, and demonstrate that using split cuts within the cut-and-project framework can significantly outperform other general purpose methods.

1 Introduction

In this paper, we study large scale mixed-integer programs arising from two-stage stochastic integer programs (SIP) and develop techniques that use integrality constraints on the first stage variables to strengthen Benders inequalities generated within the decomposition algorithm. The problems we are interested have the following block-structure form

$$\begin{aligned} &\min cx + \sum_{k\in\mathcal{K}} p_k d^k y^k && (1)\\ &Ax \geq b,\ x \in \mathbb{Z}^q_+ \times \mathbb{R}^{n-q}_+ \\ &T^k x + W^k y^k \geq h^k,\ y^k \in \mathbb{R}^t_+ \text{ for all } k \in \mathcal{K} \end{aligned}$$

where, $\mathcal{K}$ is a finite index set for scenarios, $c \in \mathbb{R}^n$, $b \in R^l$, $0 \leq q \leq n$, and for each $k \in \mathcal{K}$, $p_k > 0$, $h^k \in \mathbb{R}^m$, $d^k \in \mathbb{R}^t$, and T^k and W^k are matrices of appropriate size. In this formulation, variables x are called the *first-stage* variables and when they are fixed, the problem decomposes into subproblems, one for each scenario $k \in \mathcal{K}$. Variables y^k for $k \in \mathcal{K}$, on the other hand, are referred to as *recourse* variables. This formulation is also called the *extensive formulation* of the SIP.

When the set $\mathcal{K}$ is large, solving (1) directly may not be feasible due to its size. As a result, the stochastic programming literature has focused on methods for solving (1) via decomposition. Two common decomposition approaches are *dual decomposition* [1, 2], and *Benders decomposition*, which is also called the L-shaped method [3]. Both methods solve a relaxation of (1) to obtain lower bounds and use these bounds in a branch-and-bound framework. Dual decomposition yields stronger relaxations, but at the expense of solving integer programming subproblems. In this work we explore techniques for improving the relaxations obtained in Benders decomposition.

In Benders decomposition, the recourse variables y^k are projected out and replaced with a single variable per scenario, leading to a master problem in which cuts are added to accomplish the projection. In other words, if

$$Q_k^{LP} = \{(x, z_k, y^k) \in \mathbb{R}_+^n \times \mathbb{R} \times \mathbb{R}_+^t : z_k \geq d^k y^k,\ T^k x + W^k y^k \geq h^k\},$$

then a Benders cut is an inequality valid for

$$\mathrm{Proj}_{(x,z_k)}\big(Q_k^{LP}\big) = \{(x, z_k) \in \mathbb{R}_+^n \times \mathbb{R} : \exists y^k \text{ such that } (x, z_k, y^k) \in Q_k^{LP}\}$$

(here $\mathrm{Proj}_{(A)}\big(S\big)$ stands for the orthogonal projection of set S in the space of A variables). Letting $X = \{x \in \mathbb{Z}_+^q \times \mathbb{R}_+^{n-q} : Ax \geq b\}$, the Benders decomposition approach is a method for solving (1) by solving

$$\min\ \Big\{cx + pz\ :\ x \in X,\ (x, z_k) \in \mathrm{Proj}_{(x,z_k)}\big(Q_k^{LP}\big) \text{ for } k \in \mathcal{K}\Big\}. \qquad (2)$$

When $|\mathcal{K}|$ is large, this reformulation reduces the variable space significantly. Although $\mathrm{Proj}_{(x,z_k)}\big(Q_k^{LP}\big)$ is a polyhedron, it is usually impractical to explicitly compute it, and instead Benders decomposition implicitly approximates this projection in an iterative fashion using Benders cuts.

We explore two different techniques for using integrality information to obtain improved relaxations within Benders decomposition: (i) cut-and-project, where integrality information is used to derive cuts in the $(x, z_k, y^k,)$ space, and Benders cuts are then used to project the resulting improved relaxation, and (ii) project-and-cut, where integrality information is used to derive cuts directly for the relaxation in the (x, z_k) space defined by Benders cuts.

Specifically, in the cut-and-project approach, we replace Q_k^{LP} in (2) with a tighter set Q_k^S which satisfies $Q_k^{IP} \subseteq Q_k^S \subseteq Q_k^{LP}$, where

$$Q_k^{IP}\ =\ \mathrm{conv}\left\{(x, z_k, y^k)\ :\ x \in X,\ (x, z_k, y^k) \in Q_k^{LP}\right\}.$$

When we solve this new formulation via Benders decomposition, we obtain stronger Benders cuts, i.e., cuts that are valid for $\mathrm{Proj}_{(x,z_k)}\big(Q_k^{IP}\big)$ but are potentially not valid for $\mathrm{Proj}_{(x,z_k)}\big(Q_k^{LP}\big)$. The set Q_k^S can possibly be obtained by using problem specific structure as the deterministic version of most stochastic optimization problems are also well-studied optimization problems. Alternatively, Q_k^S can also be obtained by strengthening Q_k^{LP} with general purpose cutting-planes such as split cuts. Because of the importance of split cuts for solving deterministic integer programs [4–6], we focus on the second approach and search for violated split cuts using a heuristic for identifying violated *Gomory mixed-integer* (GMI) cuts.

In the project-and-cut approach, we directly search for valid inequalities for the sets $\mathrm{Proj}_{(x,z_k)}\big(Q_k^{IP}\big)$, $k \in \mathcal{K}$. One difficulty for using split cuts in the project-and-cut approach is that the set $\mathrm{Proj}_{(x,z_k)}\big(Q_k^{LP}\big)$ is not given explicitly, and instead is approximated via Benders cuts. However, we show that we can still use a linear program to generate split cuts for this set when the split is fixed. These cuts tighten the formulation in the projected space directly whereas the cut-and-project approach tightens the extended formulation and then projects it in the space of the variables of the master problem. We show that when considering multiple splits, using split cuts in the cut-and-project approach leads to stronger relaxations than in the project-and-cut approach. We also present computational experiments that demonstrate the strength of the cut-and-project approach on two SIP examples from the literature.

There is a significant amount of recent work studying valid inequalities for SIP with integer *recourse* variables [1, 7–13]. In contrast, we are focused on problems with integer first-stage variables, but *continuous* recourse variables. While some of this work [9, 13] yields methods that can use integrality constraints on the first-stage decision variables to obtain stronger cuts, it appears our work is the first to consider using such an approach for a problem with continuous recourse variables. While this problem class is certainly easier than the case with integer recourse variables, we computationally show that strengthened Benders cuts can still make the difference between being able to solve an instance or not.

2 Project-and-cut vs. Cut-and-project for Split Cuts

In this section we study some properties of split cuts for polyhedral sets in general and then relate our observations to two-stage stochastic programming problems. We start with comparing the effect of applying split cuts to a polyhedral set in an extended space with that of applying split cuts to the projection of this set in a lower dimensional space. More precisely, let $P \subseteq \mathbb{Z}^n \times \mathbb{R}^q$ and $Q \subseteq \mathbb{Z}^n \times \mathbb{R}^q \times \mathbb{R}^t$ be mixed integer sets, defined on variables (x, z) and (x, z, y) respectively, and let P^{LP} and Q^{LP} denote their linear relaxation. Also assume that $P^{LP} = \mathrm{Proj}_{(x,z)}\big(Q^{LP}\big)$ and therefore $P = \mathrm{Proj}_{(x,z)}\big(Q\big)$.

Let $\pi \in \mathbb{Z}^n$ be a row vector, $\gamma \in \mathbb{Z}$ and let

$$S = \{(x, z) \in \mathbb{R}^{n+q} : \gamma+1 > \pi x > \gamma\} \text{ and } S' = \{(x, z, y) \in \mathbb{R}^{n+q+t} : \gamma+1 > \pi x > \gamma\}$$

be split sets associated with (π, γ) in $\mathbb{R}^{n+q}$ and $\mathbb{R}^{n+q+t}$, respectively.

Then, an inequality $cx + dz \geq f$ is called a *split cut* for P^{LP} generated by S if it is a valid inequality for $\mathrm{conv}(P^{LP} \setminus S)$. In other words, $cx + dz \geq f$ is a split cut if it is valid for both

$$P^{LP} \cap \{(x, z) \in \mathbb{R}^n \times \mathbb{R}^q \,:\, \pi x \leq \gamma\} \text{ and } P^{LP} \cap \{(x, z) \in \mathbb{R}^n \times \mathbb{R}^q \,:\, \pi x \geq \gamma + 1\}.$$

Split cuts for Q^{LP} generated by S' are defined similarly. All points in P satisfy every split cut for $P^{LP}$and multiple split cuts can be generated using the same split set. For a fixed split set a most violated split cut (if there is one) for a given point $(\bar{x}, \bar{z})$, can be found by solving a linear program. Even though separation is easy when the split set is fixed, it is NP-Hard to find a split set that leads to a violated cut [14]. Some of the practical algorithms that are currently available are heuristics that look for violated rank-1 Gomory mixed-integer (GMI) cuts using different bases of the simplex tableau associated with P^{LP} [15–17].

2.1 Projected split cuts

We next compare the effect of split cuts for P^{LP} generated by S with that of split cuts for Q^{LP} generated by S'.

Lemma 1. *$P^{LP} \setminus S = \text{Proj}_{(x,z)}\big(Q^{LP} \setminus S'\big)$ and consequently $\text{conv}(P^{LP} \setminus S) = \text{conv}(\text{Proj}_{(x,z)}\big(Q^{LP} \setminus S'\big))$.*

Proof. Let $(\hat{x}, \hat{z}) \in P^{LP} \setminus S$. As, $(\hat{x}, \hat{z}) \in P^{LP}$, there exists at least one point $(\hat{x}, \hat{z}, \hat{y}) \in Q^{LP}$. Furthermore, as $(\hat{x}, \hat{z}) \notin S$, we have $(\hat{x}, \hat{z}, \hat{y}) \notin S'$ and consequently $(\hat{x}, \hat{y}, \hat{z}) \in Q^{LP} \setminus S'$ implying $(\hat{x}, \hat{y}) \in \text{Proj}_{(x,z)}\big(Q^{LP} \setminus S'\big)$. This establishes that $P^{LP} \setminus S \subseteq \text{Proj}_{(x,z)}\big(Q^{LP} \setminus S'\big)$.

Similarly, if $(\hat{x}, \hat{z}) \in \text{Proj}_{(x,z)}\big(Q^{LP} \setminus S'\big)$, there is a point $(\hat{x}, \hat{z}, \hat{y}) \in Q^{LP} \setminus S'$. Therefore $(\hat{x}, \hat{z}) \in P^{LP}$. In addition, as $(\hat{x}, \hat{z}, \hat{y}) \notin S'$, we have $(\hat{x}, \hat{z}) \notin S$ and therefore $\text{Proj}_{(x,z)}\big(Q^{LP} \setminus S'\big) \subseteq P^{LP} \setminus S$. $\square$

In other words, in the projected space, split cuts generated by (essentially) the same split set have the same effect whether or not they are applied before or after the projection. This observation, however, does not hold when split cuts generated by multiple split sets are considered simultaneously.

Lemma 2. *Let S_1, S_2 be two split sets for P and let S'_1, S'_2 be the corresponding split sets for Q. Then*

$$\text{conv}(P^{LP} \setminus S_1) \cap \text{conv}(P^{LP} \setminus S_2) \supseteq \text{Proj}_{(x,z)}\big(\text{conv}(Q^{LP} \setminus S'_1) \cap \text{conv}(Q^{LP} \setminus S'_2)\big),$$

and the inclusion is strict in some cases.

Proof. We first show the inclusion "$\supseteq$" and then present an example when the inclusion is strict. Let $(\hat{x}, \hat{z}) \in \text{Proj}_{(x,z)}\big(\text{conv}(Q^{LP} \setminus S'_1) \cap \text{conv}(Q^{LP} \setminus S'_2)\big)$, then, there exists $\hat{y} \in \mathbb{R}^t$ such that $(\hat{x}, \hat{z}, \hat{y}) \in \text{conv}(Q^{LP} \setminus S'_1)$ and $(\hat{x}, \hat{z}, \hat{y}) \in \text{conv}(Q^{LP} \setminus S'_2)$. Therefore, by Lemma 1, we have $(\hat{x}, \hat{z}) \in \text{conv}(P^{LP} \setminus S_1)$ and $(\hat{x}, \hat{z}) \in \text{conv}(P^{LP} \setminus S_2)$ and consequently, the inclusion holds.

To see that the inclusion is sometimes strict, consider the following example where $n = 2$, $q = 0$ and $t = 1$. Let

$$P^{LP} = \text{conv}\left\{(1/2, 0), (1/2, 1), (0, 1/2), (1, 1/2)\right\} \subseteq \mathbb{R}^2$$

and

$$Q^{LP} = \text{conv}\left\{(1/2, 0, 0), (1/2, 1, 0), (0, 1/2, 1), (1, 1/2, 1)\right\} \subseteq \mathbb{R}^3,$$

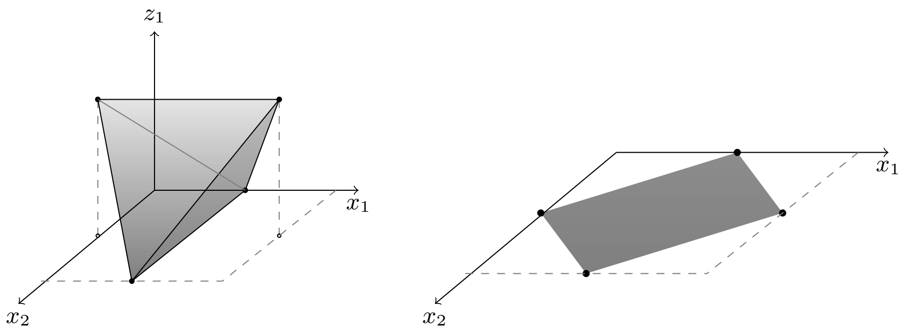

Fig. 1. The set Q^{LP} and its projection P^{LP}

(see Figure 1.) In addition, let $S_1 = \{(x_1, x_2) \in \mathbb{R}^2 \,:\, 1 > x_1 > 0\}$ and $S_2 = \{(x_1, x_2) \in \mathbb{R}^2 \,:\, 1 > x_2 > 0\}$ be the split sets for P, and S'_1 and S'_2 be the corresponding split sets for Q.

Notice that $\mathrm{conv}(P^{LP} \setminus S_1) = \mathrm{conv}\{(1/2, 0), (1/2, 1)\}$ and $\mathrm{conv}(P^{LP} \setminus S_2) = \mathrm{conv}\{(0, 1/2), (1, 1/2)\}$ and consequently, $\mathrm{conv}(P^{LP} \setminus S_1) \cap \mathrm{conv}(P^{LP} \setminus S_2) = \{(1/2, 1/2)\}$. However, $\mathrm{conv}(Q^{LP} \setminus S'_1) = \mathrm{conv}\{(1/2, 0, 0), (1/2, 1, 0)\}$ and $\mathrm{conv}(Q^{LP} \setminus S'_2) = \mathrm{conv}\{(0, 1/2, 1), (1, 1/2, 1)\}$ and $\mathrm{conv}(Q^{LP} \setminus S'_1) \cap \mathrm{conv}(Q^{LP} \setminus S'_2) = \emptyset$. $\square$

We also note that the split closure of P is in fact equal to $\{(1/2, 1/2)\}$ [18] and Lemma 2 establishes that the split closure of Q is strictly contained in the split closure of P even though $P^{LP} = \mathrm{Proj}_{(x,z)}(Q^{LP})$. We also note that a similar observation is made by [19] in the context of mixed-integer nonlinear programming.

2.2 Working in projected space

Let Q^{LP}, the extended formulation of P^{LP}, be explicitly given as follows:

$$Q^{LP} = \{(x, z, y) \in \mathbb{R}^n \times \mathbb{R}^q \times \mathbb{R}^t \,:\, Hx + Kz + Ly \ \geq \ g, \ x, y, z \geq 0\}.$$

Furthermore, let $S'_i = \{(x, z, y) \in \mathbb{R}^{n+q+t} \,:\, \gamma^i + 1 > \pi^i x > \gamma^i\}$ for $i \in I$ be a given collection of split sets. Then, it is possible to generate cuts for

$$\hat{P} \ = \ \mathrm{Proj}_{(x,z)} \Big(\bigcap_{i \in I} \mathrm{conv}(Q^{LP} \setminus S'_i) \Big) \tag{3}$$

using a single program.

Theorem 1. *The inequality*

$$cx + dz \geq f \tag{4}$$

is valid for the set $\hat{P}$ if and only if there exists a solution to the following set of inequalities:

$$c = \sum_{i \in I} c^i, \quad d = \sum_{i \in I} d^i, \quad 0 = \sum_{i \in I} h^i, \quad f = \sum_{i \in I} f^i \tag{5}$$

$$\left.\begin{array}{lll} c^i \geq \lambda_1^i H - \mu_1^i \pi^i & c^i \geq \lambda_2^i H + \mu_2^i \pi^i & \\ d^i \geq \lambda_1^i K & d^i \geq \lambda_2^i K & \\ h^i \geq \lambda_1^i L & h^i \geq \lambda_2^i L & \\ f^i \leq \lambda_1^i g - \mu_1^i \gamma^i & f^i \leq \lambda_2^i g + \mu_2^i (\gamma^i + 1) & \\ \lambda_1^i, \lambda_2^i \geq 0 & \mu_1^i, \mu_2^i \geq 0, & c^i, d^i, f^i \text{ free} \end{array}\right\} \forall i \in I \tag{6}$$

where c^i, d^i, λ_1^i and λ_2^i are row vectors of appropriate dimension, and f^i, μ_1^i and μ_2^i are real numbers for all $i \in I$.

Proof. Let Q'_i denote $\operatorname{conv}(Q^{LP} \setminus S'_i)$. Inequalities (6) ensure that for each $i \in I$ the inequality $c^i x + d^i z + h^i y \geq f^i$ is valid for Q'_i. Consequently, by inequality (5), $cx + dz \geq f$ is valid for $\bigcap_{i \in I} Q'_i$ and therefore valid for $\hat{P}$.

Conversely, assume that inequality (4) is valid for $\hat{P} = \operatorname{Proj}_{(x,y)} (\bigcap_{i \in I} Q'_i))$. Thus, $cx + dz \geq f$ is valid for $\bigcap_{i \in I} Q'_i$ and therefore it is a non-negative linear combination of valid inequalities for $\bigcap_{i \in I} Q'_i$. As any valid inequality for $\bigcap_{i \in I} Q'_i$ is a non-negative linear combination of valid inequalities for Q'_i, we have that $cx + dz \geq f$ can be obtained by a non-negative linear combination of valid inequalities for Q'_i. Consequently, for each $i \in I$ there exist a nonnegative weight w_i, and an inequality

$$c^i x + d^i z + h^i y \geq f^i \tag{7}$$

valid for Q'_i such that $c = \sum_{i \in I} c^i$, $d = \sum_{i \in I} d^i$, $0 = \sum_{i \in I} h^i$, and, $f = \sum_{i \in I} f^i$. Finally, note that if inequality (7) is valid for Q'_i, there must exist $\lambda_1^i, \lambda_2^i, \mu_1^i$ and μ_2^i that satisfy inequalities (6). □

Consequently, for a given point $(\bar{x}, \bar{z}) \notin \hat{P}$, a most violated valid inequality can be found by solving the following linear program:

$$\begin{aligned} \min \quad & z = c\bar{x} + d\bar{z} - f & \text{(8a)} \\ \text{subject to} \quad & ||\lambda_1||_1 + |||\lambda_2||_1 + ||\mu_1||_1 + ||\mu_2||_1 \leq 1 & \text{(8b)} \\ & \text{inequalities (5), (6).} & \text{(8c)} \end{aligned}$$

where inequality (8b) can be replaced with any other normalization constraint that truncates the cone defined by inequalities (5), (6).

In the case of a single split set, an interesting consequence of Lemma 1 and Theorem 1 is that, split cuts for the projected set P^{LP} can be separated even when an explicit characterization of this set is not available.

Corollary 1. *Given a split set S, split cuts for P^{LP} can be found by the Separation LP (8) that uses inequalities defining Q^{LP} only.*

Therefore, when P^{LP} is defined as a projection of a higher dimensional set (as it is the case in decomposition based approaches for stochastic programming) it is still possible to generate split cuts for this set without an explicit knowledge of the inequalities defining it. We next present an observation that shows that only a subset of the split sets are needed when solving the Separation LP (8).

Lemma 3. *Let $(\bar{x}, \bar{z}) \in \mathbb{R}^{n+q}$ be a given a point such that $(\bar{x}, \bar{z}) \in S_i$ for $i \in \bar{I}$ and $(\bar{x}, \bar{z}) \notin S_i$ for $i \in I \setminus \bar{I}$. If $(\bar{x}, \bar{z}) \notin \hat{P}$, then*

$$(\bar{x}, \bar{z}) \;\notin\; \operatorname{Proj}_{(x,z)} \big(\textstyle\bigcap_{i \in \bar{I}} \operatorname{conv}(Q^{LP} \setminus S_i') \big).$$

Proof. Let Q_i' denote $\operatorname{conv}(Q^{LP} \setminus S_i')$ and assume that there exists a point $(\bar{x}, \bar{z}) \in \operatorname{Proj}_{(x,z)} \big(\bigcap_{i \in \bar{I}} Q_i' \big)$. Then, by definition, there exist a point $(\bar{x}, \bar{z}, \bar{y}) \in \big(\bigcap_{i \in \bar{I}} Q_i' \big)$. As $(\bar{x}, \bar{z}, \bar{y}) \in Q^{LP}$ and $(\bar{x}, \bar{z}) \notin S_i$ for $i \in I \setminus \bar{I}$, we have $(\bar{x}, \bar{z}, \bar{y}) \in Q_i'$ for $i \in I \setminus \bar{I}$. Therefore, $(\bar{x}, \bar{z}, \bar{y}) \in \big(\bigcap_{i \in I} Q_i' \big)$ and consequently, $(\bar{x}, \bar{z}) \in \hat{P}$. $\square$

Corollary 2. *It suffices to solve the Separation LP (8) with variables and constraints associated with $i \in \bar{I}$ only.*

2.3 Benders decomposition and integrality-based cuts

We first describe how Benders decomposition is used within a branch-and-cut algorithm for solving (2). To simplify exposition we assume that $\{x \in \mathbb{R}^n_+ : Ax \geq b\} \supseteq \operatorname{Proj}_{(x)} \big(Q_k^{LP} \big)$ for all $k \in \mathcal{K}$ (see [11]). We also define $P_k^{LP} := \operatorname{Proj}_{(x,z_k)} \big(Q_k^{LP} \big)$ for $k \in \mathcal{K}$.

The algorithm is a standard enumeration algorithm where node relaxations consist of the first-stage variables x and z, together with inequalities $Ax \leq b$ and globally valid Benders cuts that have been already generated. Let $B_k \supseteq P_k^{LP}$ represent the polyhedron defined by the Benders cuts currently in the master LP relaxation for scenario k. Given a node relaxation solution $(\bar{x}, \bar{z})$, we determine if $(\bar{x}, \bar{z}_k) \in P_k^{LP}$ for $k \in \mathcal{K}$ by solving the linear program:

$$f_k(\bar{x}) = \min \; \{z : z \geq d^k y, \; W^k y \geq h^k - T^k \bar{x}, y \geq 0\}. \tag{9}$$

If $\bar{z}_k \geq f_k(\bar{x})$ then $(\bar{x}, \bar{y}) \in P_k^{LP}$. Otherwise, if $\bar{\pi}$ is an optimal dual solution to the linear program (9), then $(\bar{x}, \bar{z})$ violates the Benders cut $z_k \geq \bar{\pi} h^k - \bar{\pi} T^k x$, and so this cut is added to the description of B_k.

Note that for the purpose of obtaining a valid lower bound at a node of the enumeration tree, it is not necessary for the current solution of the LP relaxation to satisfy all Benders cuts. For computational efficiency, it is sometimes preferable not to generate all Benders cuts at all tree nodes. If the solution is integral, however, one has to make sure that it satisfies all Benders cuts.

Cut-and-project. In this approach, we identify valid inequalities for the set Q_k^{IP}, leading to an approximation Q_k^S which satisfies $Q_k^{IP} \subseteq Q_k^S \subseteq Q_k^{LP}$. To implement this approach, suppose we having solved a subproblem (9) for a scenario $k \in \mathcal{K}$ and obtained a solution (z^*, y^*). We then attempt to find one or more valid inequalities for the set Q_k^{IP} that cut off the solution $(\bar{x}, z^*, y^*)$. If cuts are found, they are added to the LP (9) (i.e. T^k, W^k and h^k are updated) and the LP is re-solved. This process can be repeated. At the end, a Benders cut, $z_k \geq \bar{\pi} h^k - \bar{\pi} T^k x$, is formed and added to the master LP relaxation.

One important advantage of this approach when using split cuts is that the node LP relaxations can be superior to those obtained using project-and-cut instead. In addition, moving beyond split cuts, inequalities derived based on the structure of the set Q_k^{IP} can easily be integrated into the algorithm. In this way, the proposed framework can take advantage of any polyhedral results known for the deterministic version of a problem to help solve the stochastic version. Finally, in contrast to the dual decomposition, the subproblems remain LPs.

Project-and-cut. Given a solution $(\bar{x}, \bar{z}_k)$ of the master LP relaxation, this option searches for valid inequalities for the set $\text{Proj}_{(x,z_k)}(Q_k^{IP})$ that cut off this solution, and adds such inequalities to the master LP relaxation. A primary disadvantage of this approach is that the set Q_k^{IP} may have structure that is amenable to the use of certain classes of valid inequalities, which may be "hidden" in the projected space.

Another disadvantage of this approach is that the strength of the cuts is limited by the cuts used in the current approximation B_k of P_k^{LP}. However, for the case of split cuts defined by a split S, an inequality valid for $\text{conv}(P_k^{LP} \backslash S)$ can be found using S as the only split in the LP (8), thus overcoming this limitation. The resulting LP simultaneously identifies valid inequalities in the extended space *and* projects this to the space of master problem variables. A potential advantage of this approach is that it does not require an iterative procedure for generating split cuts in the extended space before projecting, or storing a list of valid split cuts in the extended space for each scenario. Unfortunately, as demonstrated in Lemma 2, this approach can yield weaker relaxations if we solve (8) for a single split at a time. While considering multiple splits simultaneously can overcome this weakness, this does not appear to be a computationally viable option due the large size of the resulting LP.

3 Numerical illustration

We used two problem sets from the literature in our computational experiments. Detailed description of these problems are provided in the appendix.

Our first test problem, 'CAP', is a stochastic version of the capacitated facility location problem, as described by Louveaux [20]. In this problem, binary facility opening decisions must be made before observing the realizations of random customer demands. Then, in the second stage the customer demands are fractionally allocated to the facilities. Data for this test problem is obtained by starting with deterministic 'CAP' instances from the OR-Library [21] that have 50 customers and 25-50 potential facilities. We construct the demands of the stochastic instances by sampling either 250 or 500 scenarios from a normal distribution with mean taken from the corresponding deterministic instance.

The second test problem is the stochastic network interdiction problem (SNIP) described in [22]. All instances have 456 scenarios that contain 320 binary first-stage decision variables, corresponding to which arc to interdict subject to a budget constraint. The second-stage problems are network-structured LPs with 5290 constraints and 830 decision variables each. We focused our experiments on the more difficult instances which we call snipno=3 and snipno=4. These are the instances reported in Tables 3 and 4 in [22] where time is measured in minutes.

3.1 Computational Environment and Implementation Details.

We implemented all algorithms in C++ and solved all LPs and IPs using IBM ILOG CPLEX 12.4. All experiments were run using a single thread on a Linux workstation with 2.33 GHz Intel Xeon CPUs and 32 GB memory. We used a time limit of 4 hours. The extensive formulation was solved with default Cplex settings, and presolve features were turned off for the Benders algorithms. Rank-1 GMI cuts were obtained using the FEAS heuristic of Dash and Goycoolea [15].

3.2 Computational experiments with the root bound

We first investigate the root relaxation gap given by standard Benders decomposition and three different methods that improve the relaxation with split cuts:

– CGLP: Use the CGLP (8) with a *single* split at a time to separate cuts directly in the (x, z_k) space for each scenario k, and only use *simple splits sets* of the form $S = \{\gamma < x_j < \gamma + 1\}$ that involve a single variable.
– CGLPSP: Use a standard CGLP to add split cuts in the *extended space*, as in the cut-and-project approach, and limit to simple split sets.
– GMISP: Use the Rank-1 GMI heuristic to add split cuts in the extended space.

For CGLP and CGLPSP we continue to violated add cuts as long as we can find them and therefore we optimize over the *closure* of simple disjunctions. Table 1 reports the average percentage root gap (defined in the usual way) obtained by each of these methods over sets of CAP and SNIP instances having similar properties. We find that for the CAP instances, CGLP yields a modest improvement in gap over standard Benders. On the other hand, CGLPSP yields a dramatic improvement over CGLP, indicating that the difference in strength of the relaxations suggested by Lemma 2 can sometimes be very large. Finally, the GMI heuristic closes a similar amount of gap as the CGLPSP simple split closure. For the SNIP instances, we were not able to compute the CGLPSP gap within a day of computation time, and so we only report the gaps obtained by Benders, CGLP, and GMISP. We observe in this case that both CGLP and GMISP yield significantly smaller gaps than Benders, but GMISP does not close as much gap as on the CAP instances. In addition, in this case, the difference between working in the projected space (CGLP) and the extended space (GMISP) is not so significant. In both instances, the solution times for obtaining the CGLP and CGLPSP gaps were not practical because no early termination criterion was applied for these. On the other hand, the Benders and GMISP times were reasonable (29 sec. and 310 sec. on average for the CAP instances, respectively).

Table 1. Average % root gap obtained by different methods.

CAP#	Ben	CGLP	CGLPSP	GMISP	SNIP Budget	Ben	CGLP	GMISP
101-104	22.4	18.0	0.11	0.07	30	22.5	5.8	7.8
111-114	8.7	7.8	0.22	0.39	50	27.5	10.4	12.0
121-124	18.9	16.3	1.15	0.99	70	28.9	14.8	12.5
131-134	25.2	21.6	1.46	0.30	90	33.1	18.8	17.6

3.3 Computational experiments with the branch-and-bound tree

We next explore the use of GMI cuts in the cut-and-project approach within Benders decomposition. We refer to this method as Ben+GMI. We compare Ben+GMI to two alternative general-purpose approaches. The first method (Ext) is to solve the extensive formulation using default Cplex, and the second method (Ben) is to use a pure Benders decomposition algorithm. Details of the implementation will be provided in the full version of the paper.

Table 2. Comparison of algorithms for CAP instances. None of the instances were solved by Benders within the time limit, so those times are not reported. Nearly all instances were solved with Ben+GMI, so the Avg Gap is not reported for that method.

		Avg Time (# unsolved)		Avg Gap (%)		Avg # Nodes		
K	CAP #	Ext	+GMI	Ext	Ben	Ext	Ben	+GMI
250	101-104	258.1	56.2	0.00	14.08	0	7863.5	17.5
	111-114	2359.0	644.0	0.00	6.94	0	4640.1	1028.0
	121-124	3252.3 (2)	1223.4	0.43	15.62	0	4383.3	984.8
	131-134	4150.8 (1)	294.8	0.19	22.23	0	5653.2	164.2
500	101-104	1170.5	113.4	0.00	15.69	0	3975.0	16.9
	111-114	10787.1 (3)	1994.2 (1)	2.00	7.54	0	1853.7	969.0
	121-124	10935.4 (3)	2420.0	3.12	15.89	0	1913.6	583.9
	131-134	9512.0 (3)	737.3	1.44	23.09	0	2539.7	157.6

Tables 2 and 3 present a summary of the results obtained by the three methods on the CAP and SNIP instances, respectively. Each row is the average of several instances having similar characteristics. For solution time and nodes, the geometric mean is reported, where the times used in the average were truncated at the four hour time limit. The number of instances that are not solved by a method is reported in parentheses in the time column. Thus, if not all instances in a set were solved all within the time limit, the reported mean time and nodes are lower bounds on what would be required to solve the instances to optimality. We also report the arithmetic average optimality gap over the instances after the time limit, where gap is calculated as (UB-LB)$\times$100/UB where UB and LB are the upper and lower bound, respectively. Instances that were solved to optimality are included as a zero in calculating this average.

Table 3. Comparison of algorithms for SNIP instances. None of the instances were solved by Ext in the time limit, so those times are not reported.

		Avg Time (# unsolved)		Avg Gap (%)			Avg # Nodes		
snipno	Budget	Ben	+GMI	Ext	Ben	+GMI	Ext	Ben	+GMI
3	30	1139	426	18.0	0.00	0.00	0	394343	15346
	50	11838 (3)	2158	26.1	1.95	0.00	0	2695041	209329
	70	14400 (5)	8242 (1)	27.1	3.25	0.08	0	2747827	989760
	90	14400 (5)	13425 (3)	30.7	6.40	1.50	0	2456388	1627738
4	30	695	412	22.4	0.00	0.00	0	161141	7764
	50	4966 (1)	1107	29.1	0.88	0.00	0	866078	60123
	70	9554 (2)	1597	39.0	0.34	0.00	0	1137123	69672
	90	9641 (3)	1475	58.7	0.82	0.00	0	794407	63824

For the CAP instances, we find that the extensive formulation is able to solve many of the instances with 250 scenarios, but fails frequently with 500 scenarios. On the other hand, Benders decomposition fails to solve any of the instances, and yields large optimality gaps after the time limit. This is not surprising given the results in Table 1 which indicate that Benders yields weak LP relaxations. Finally, we observe that the Ben+GMI algorithm is able to solve nearly all of these instances in the time limit, and does so significantly faster than the extensive formulation. For the SNIP instances, the role of the extensive formulation and Benders algorithm is reversed: the extensive formulation is unable to solve any of the instances, and yields a very large average optimality gap after the time limit, whereas Benders decomposition is able to solve some of the instances. This is caused by the relatively larger size of these instances, which prevents the extensive formulation from even being able to finish processing the root node within the time limit, whereas Benders decomposition is able to process a large number of nodes. Once again, we find that Ben+GMI yields the best performance by far, which is attributed to the significant reduction in the number of nodes processed. Finally, we remark that the results obtained with this general purpose method are comparable to those obtained in [22] using problem-specific valid inequalities.

Acknowledgements: The authors are grateful to David Morton and Feng Pan for providing the instances used in [22]. This research has been supported in part by the National Science Foundation under grant CMMI-1130266.

References

1. Carøe, C.C.: Decomposition in Stochastic Integer Programming. PhD thesis, Department of Operations Research, University of Copenhagen, Denmark (1998)
2. Carøe, C.C., Schultz, R.: Dual decomposition in stochastic integer programming. Oper. Res. Lett. (1999) 37–45
3. Van Slyke, R., Wets, R.J.: L-shaped linear programs with applications to optimal control and stochastic programming. SIAM J. Appl. Math **17** (1969) 638–663
4. Balas, E., Ceria, S., Cornuejols, G., Natraj, N.: Gomory cuts revisited. Oper. Res. Lett. **19** (1996) 1–9
5. Balas, E., Saxena, A.: Optimizing over the split closure. Math. Program. **113** (2008) 219–240
6. Dash, S., Günlük, O., Lodi, A.: Mir closures of polyhedral sets. Math. Program. **121** (2010) 33–60
7. Carøe, C.C., Tind, J.: L-shaped decomposition of two-stage stochastic programs with integer recourse. Math. Program. (1998) 451–464
8. Gade, D., Küçükyavuz, S., Sen, S.: Decomposition algorithms with parametric gomory cuts for two-stage stochastic integer programs. Math. Program. (2012) 10.1007/s10107-012-0615-6.
9. Tanner, M., Ntaimo, L.: Computations with disjunctive cuts for two-stage stochastic mixed 0-1 integer programs. J. Global. Opt. **58** (2008) 365–384
10. Ntaimo, L.: Fenchel decomposition for stochastic mixed-integer programming. J. Global. Opt. **55** (2013) 141–163
11. Sen, S., Higle, J.L.: The C^3 theorem and a D^2 algorithm for large scale stochastic mixed-integer programming: set convexification. Math. Program. **104** (2005) 1–20
12. Sen, S., Sherali, H.: Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. Math. Program. **106** (2006) 203–223
13. Zhang, M., Küçükyavuz, S.: Finitely convergent decomposition algorithms for two-stage stochastic pure integer programs. (2013)
14. Caprara, A., Letchford, A.: On the separation of split cuts and related inequalities. Math. Program. **94** (2003) 279–294
15. Dash, S., Goycoolea, M.: A heuristic to separate rank-1 GMI cuts. Math. Program. Comput. **2** (2010) 231–257
16. Fischetti, M., Salvagnin, D.: A relax-and-cut framework for gomory's mixed-integer cuts. Math. Program. Comput. **3** (2011) 79–102
17. Bonami, P.: On optimizing over lift-and-project closures. Math. Program. Comput. **4** (2012) 151–179
18. Cornuéjols, G., Li, Y.: On the rank of mixed 0,1 polyhedra. Math. Program. **91** (2002) 391–397
19. Modaresi, S., Kilinc, M., Vielma, J.P.: Split cuts for conic programming (2012) Poster presented at MIP2012, July 16, UC-Davis, Davis, CA.
20. Louveaux, F.V.: Discrete stochastic location models. Ann. Oper. Res. **6** (1986) 23–34
21. Beasley, J.: OR-Library: Distributing test problems by electronic mail. J. Oper. Res. Soc. **41** (1990) 1069–1072 http://people.brunel.ac.uk/ mastjjb/jeb/orlib/capinfo.html.
22. Pan, F., Morton, D.P.: Minimizing a stochastic maximum-reliability path. Networks **52**(3) (2008) 111–119

Appendix

Detailed description of test problems

We first describe the details of the capacitated facility location ('CAP') test problem. Let $\mathcal{I}$ be the set of potential facilities, $\mathcal{J}$ be the set of customers and $\mathcal{K}$ be the set of scenarios. For the cardinalities of the sets, we use $|\mathcal{I}| = I,\ |\mathcal{J}| = J,\ |\mathcal{K}| = K$. The first stage facility opening decision variables are denoted by x, while second stage flow variables are denoted by y. The parameter f corresponds to facility opening cost, q corresponds to allocation cost, and s corresponds to facility capacity. The demand of customer j under scenario k is represented by λ_j^k. We use the following formulation for this problem:

$$\min \sum_{i\in\mathcal{I}} f_i x_i + \frac{1}{K}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} q_{ij} y_{ij}^k \tag{10a}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}} y_{ij}^k \geq \lambda_j^k\ , \qquad j\in\mathcal{J},\ k\in\mathcal{K} \tag{10b}$$

$$\sum_{j\in\mathcal{J}} y_{ij}^k \leq s_i x_i\ , \qquad i\in\mathcal{I},\ k\in\mathcal{K} \tag{10c}$$

$$\sum_{i\in\mathcal{I}} s_i x_i \geq \max_{k\in\mathcal{K}} \sum_{j\in\mathcal{J}} \lambda_j^k \tag{10d}$$

$$x \in \{0,1\}^I, y \in \mathbb{R}_+^{IJK} \tag{10e}$$

This formulations differ slightly from the "standard formulation" from [20]. In this formulation the recourse variables assign flows from facilities to customers, whereas the the formulation in [20] uses decision variables $\bar{y}_{ij}^k = y_{ij}^k/\lambda_j^k$ to represent *fraction* of a customer's demand served by a facility. We use the above formulation because it results in a problem with no uncertainty in the *technology matrix* (the coefficients multiplying x_i in the second-stage constraints). Also, we introduce constraint (10d) in order to have relatively complete recourse. In other words, it makes the second stage feasible for each fixed feasible first stage solution.

For the deterministic data, we use CAP instances from J E Beasley's OR-Library [21]. Let $\bar{\lambda}_j$ be the deterministic demand of customer j in the data. Since we reformulated constraints (10b) and (10c), we first scale allocation costs with the reciprocal of provided demand values. We use normal distribution to generate stochastic demands $\lambda_j \sim N(\mu_j, \sigma_j)$ where we take for each $j \in \mathcal{J},\ \mu_j = \bar{\lambda}_j$ and $\sigma_j \in U(0.1\bar{\lambda}_j, 0.3\bar{\lambda}_j)$, where U represents uniform distribution.

We next describe the stochastic network interdiction problem (SNIP) test problem, which is based on [22]. In this problem, the interdictor selects the arcs to install sensors on subject to a capacity constraint among a given subset of arcs in the network. Then, the evader selects a maximum-reliability path. However, the evader's origin-destination pair is known to the interdictor only through a probability distribution.

Let N and A represent the set of nodes and the set of arcs in a given directed network, respectively. D is defined as the subset of A on which sensors are allowed to be replaced. b is total budget for installations. c_{ij} is cost of installing a sensor on arc $(i, j) \in D$. r_{ij} is the probability that the evader traverses uninterdicted arc (i, j) undetected, whereas q_{ij} is the undetection probability of traversing interdicted arc (i, j). Scenario k is represented by origin-destination pair (s^k, t^k) whose realization probability is denoted by p^k. Lastly, ψ_j^k is used for the value of a maximum-reliability path from j to t^k in the case that no sensors are placed. Note that ψ_j^k can be calculated by solving a shortest path problem.

First stage binary variable x_{ij} is 1 if a sensor is installed on arc (i, j). Second stage variable π_i corresponds to the conditional probability that evader can travel from i to t^k undetected. We directly describe the formulation in terms of the master and subproblem. The master problem is

$$\min \{pz : cx \leq b, z_k \geq f_k(x), k \in \mathcal{K}, x \in \{0, 1\}^{|D|}\}$$

where for each $k \in \mathcal{K}$

$$\begin{aligned} f_k(x) = \min\ & \pi_{s^k} \\ \text{s.t. } & \pi_{t^k} = 1 \\ & \pi_i - q_{ij}\pi_j \geq 0\ , && (i,j) \in D \\ & \pi_i - r_{ij}\pi_j \geq 0\ , && (i,j) \in A \setminus D \\ & \pi_i - r_{ij}\pi_j \geq -(r_{ij} - q_{ij})\psi_j^k x_{ij}\ , && (i,j) \in D \\ & \pi_i \geq 0, && (i,j) \in A \end{aligned}$$

Detailed Computational Results

Table 4 provides the gap closed by the different methods for all 250 scenario CAP instances in our test set. These results are summarized in Table 1 in the main paper. Table 5 provides the relaxation gaps for all the SNIP instances we have this data for, those with snipno=3 (corresponding to the instances in Table 3 of [22]). The results from this table for budget levels of 30,50,70,and 90 are summarized in Table 1 of the main paper.

Table 6 presents the results for solving each individual instance of the CAP test set with the different methods. These results are summarized in Table 2 in the main paper. In the columns under the Time(sec)/Gap heading, if the instance is solved within the time limit the time in seconds is reported. Otherwise, the endinging optimality gap is reported in bold.

We have a couple of outliers, such as CAP114 for 250 scenarios, in which the Benders+GMI implementation processes many more branch-and-bound nodes than the other algorithms. The reason for this behavior is our choice of implementation setting. Specifically, we have some settings that attempt to pick a subset of scenarios on which to attempt to generate cuts, which go into effect after a certain warm-up phase. In these outlier cases, the setting switches to the

Table 4. Relaxation gap (%) for CAP instances with 250 scenarios.

CAP#	Ben	CGLP	CGLPSP	GMISP
101	17.45	13.64	0.00	0.03
102	21.88	17.49	0.00	0.11
103	24.45	20.01	0.11	0.10
104	25.81	20.86	0.33	0.04
111	9.02	8.06	0.05	0.04
112	8.98	8.04	0.20	0.33
113	8.50	7.52	0.11	0.36
114	8.38	7.54	0.53	0.81
121	15.89	13.65	0.03	0.18
122	19.03	16.47	0.60	0.78
123	20.14	17.32	1.38	1.11
124	20.64	17.84	2.59	1.89
131	20.55	17.35	0.01	0.02
132	25.22	21.65	0.94	0.22
133	27.13	23.22	1.87	0.36
134	28.03	24.04	3.03	0.60

reduced effort setting too quickly, and so the algorithm becomes much less aggressive in generating cuts at that point. As a result, the GMI version reaches less that one percent optimality gap very quickly (e.g. in CAP114 for 250 scenarios in less than 160 seconds) but then switches to the less aggressive cut generation scheme. At that point, it solves far fewer subproblems and is able to quickly explore many branch-and-bound nodes. Unfortunately, it has a hard time to find integer solutions, and therefore it explores many nodes. On the other hand, in these instances, pure Benders version takes a very long time to switch to the less aggressive cut generation setting and thus spends a lot of time adding cuts to the master problem. As a result, the pure Benders method explores many fewer nodes than the method with GMI cuts for these outlier instances.

Tables 7 and 8 present the results for solving each individual instance of the SNIP test set with snipno=3 and snipno=4, respectively. These results are partially summarized in Table 3 of the main paper, which includes only summary results for budget levels 30,50,70 and 90.

Table 5. Relaxation gap (%) for SNIP instances with snipno=3.

Budget	Instance	Ben	CGLP	GMISP
30	0	22.27	5.69	7.99
	1	22.10	5.61	7.92
	2	22.48	5.36	7.00
	3	22.63	6.28	8.11
	4	22.88	6.03	7.81
40	0	25.25	8.38	10.98
	1	26.74	9.85	12.25
	2	25.45	8.14	9.80
	3	26.65	9.44	11.20
	4	27.01	9.73	12.10
50	0	26.86	9.81	11.96
	1	28.53	11.99	13.52
	2	27.02	9.62	11.07
	3	25.85	8.27	9.64
	4	29.42	12.42	13.70
60	0	26.90	9.63	11.42
	1	28.59	15.30	13.24
	2	28.63	15.79	12.30
	3	27.15	12.91	11.73
	4	29.56	15.70	13.50
70	0	29.02	16.48	12.65
	1	28.60	14.35	12.39
	2	28.81	14.86	12.15
	3	28.38	12.87	11.94
	4	29.72	15.29	13.59
80	0	28.90	12.85	12.07
	1	31.35	13.81	14.31
	2	30.90	14.54	14.92
	3	31.67	13.77	15.27
	4	31.54	14.58	15.57
90	0	31.04	15.81	14.95
	1	34.15	18.40	18.54
	2	33.00	19.44	18.43
	3	33.40	19.33	17.33
	4	33.78	20.79	18.65

Table 6. Comparison of algorithms to solve stochastic capacitated facility location problem.

		Time(sec) / **Gap(%)**			# Nodes		
K	CAP #	Ext	Ben	+GMI	Ext	Ben	+GMI
250	101	104.2	**10.39%**	48.9	0	6539	14
	102	246.7	**12.80%**	63.1	0	6820	18
	103	358.3	**16.88%**	55.4	0	9602	17
	104	481.9	**16.26%**	58.6	0	8929	22
	111	1003.5	**8.31%**	196.3	0	4410	30
	112	2869.2	**6.65%**	311.7	0	4200	484
	113	3320.1	**5.92%**	481.0	0	4800	1322
	114	3239.6	**6.88%**	5842.4	0	5214	58191
	121	624.7	**15.11%**	264.3	0	4695	43
	122	3824.4	**15.76%**	881.2	0	4680	803
	123	**0.88%**	**16.39%**	1371.5	0	4000	1796
	124	**0.85%**	**15.20%**	7012.7	0	4200	15170
	131	631.0	**21.82%**	207.4	0	6836	38
	132	3221.1	**23.74%**	289.6	2	5800	263
	133	10142.5	**22.82%**	380.7	78	5200	322
	134	**0.76%**	**20.54%**	330.1	7	4954	226
500	101	431.7	**13.56%**	94.1	0	3949	16
	102	1082.3	**14.51%**	120.8	0	2733	26
	103	1837.4	**17.63%**	115.6	0	5064	18
	104	2186.5	**17.06%**	126.0	0	4568	11
	111	4534.5	**9.22%**	455.2	0	1668	33
	112	**0.12%**	**7.53%**	792.1	0	1703	403
	113	**3.38%**	**6.63%**	**0.24%**	0	1920	32800
	114	**4.49%**	**6.77%**	3045.8	0	2165	2021
	121	4789.1	**15.39%**	633.5	0	2075	59
	122	**1.73%**	**16.81%**	2512.4	0	1951	869
	123	**4.93%**	**16.37%**	3741.6	0	1880	1569
	124	**5.82%**	**15.01%**	5759.2	0	1762	1445
	131	2741.6	**22.23%**	460.3	0	3327	42
	132	**0.12%**	**25.52%**	670.3	0	2600	133
	133	**1.70%**	**23.04%**	1215.7	0	2186	913
	134	**3.95%**	**21.56%**	787.7	0	2200	121

Table 7. Comparison of algorithms to solve stochastic network interdiction problem for snipno=3.

		Time(sec) / **Gap(%)**			# Nodes		
Budget	Instance	Ext	Ben	+GMI	Ext	Ben	+GMI
30	0	**18.21**	701.4	358.2	0	235235	9260
	1	**17.24**	1154.6	410.0	0	346106	9720
	2	**18.75**	1098.0	472.6	0	401498	57307
	3	**16.37**	1613.0	480.0	0	599573	13970
	4	**19.35**	1335.3	421.3	0	486556	11812
40	0	**21.29**	3253.9	562.4	0	1099438	23292
	1	**22.45**	**1.90**	2795.5	0	3025236	354326
	2	**20.95**	**1.50**	2961.9	0	5196073	605957
	3	**23.94**	12693.7	6457.3	0	6755231	1299021
	4	**36.56**	13604.9	1774.6	0	4666531	172624
50	0	**23.54**	10207.5	798.0	0	2259458	41493
	1	**20.55**	**3.25**	5701.8	0	2571959	600973
	2	**23.02**	**3.18**	2993.5	0	3811724	449364
	3	**25.96**	7628.9	643.1	0	2209044	59091
	4	**37.65**	**3.30**	5347.2	0	2905580	607007
60	0	**24.39**	7572.4	887.1	0	1461985	60200
	1	**19.13**	**2.38**	2759.5	0	2316265	255021
	2	**25.47**	**3.58**	**1.14**	0	3494470	2049624
	3	**25.02**	**3.44**	2520.6	0	3678437	318494
	4	**24.66**	**1.48**	2185.8	0	2526100	147437
70	0	**23.02**	**3.81**	7585.1	0	2434317	981786
	1	**20.30**	**2.97**	11723.7	0	2641550	1407500
	2	**26.87**	**4.45**	**0.42**	0	3076319	2145250
	3	**29.29**	**4.28**	3544.8	0	3050410	401734
	4	**36.04**	**0.73**	8381.1	0	2596101	797565
80	0	**25.12**	**0.68**	2757.3	0	2672975	289731
	1	**26.33**	**4.29**	**0.28**	0	2452384	1359834
	2	**30.95**	**5.73**	**1.25**	0	2348300	2209948
	3	**23.59**	**6.26**	**0.75**	0	1657439	2060831
	4	**38.39**	**2.17**	5175.5	0	4489205	577516
90	0	**25.77**	**3.63**	13884.7	0	2337641	1780947
	1	**26.97**	**8.84**	**4.56**	0	1927200	1680441
	2	**32.93**	**7.93**	**2.09**	0	2339400	2045585
	3	**27.75**	**6.42**	10517.7	0	2522921	1268873
	4	**40.20**	**5.18**	**0.83**	0	3363349	1471001

Table 8. Comparison of algorithms to solve stochastic network interdiction problem for snipno=4

		Time(sec) / **Gap(%)**			# Nodes		
Budget	Instance	Ext	Ben	+GMI	Ext	Ben	+GMI
30	0	**19.44**	486.3	327.0	0	112618	2997
	1	**20.05**	770.4	375.0	0	156449	5497
	2	**24.88**	1201.7	548.3	0	368805	44408
	3	**26.09**	695.7	417.5	0	147239	6505
	4	**21.74**	518.2	420.7	0	113560	5928
40	0	**22.57**	653.6	410.8	0	116204	6374
	1	**26.51**	4040.4	702.6	0	937433	32904
	2	**22.89**	5192.8	1295.2	0	1453346	252051
	3	**29.41**	4035.5	1485.0	0	955481	150434
	4	**27.22**	3886.0	944.4	0	859117	63763
50	0	**26.65**	1563.5	798.4	0	222326	51085
	1	**26.72**	5234.9	927.1	0	886194	25801
	2	**28.91**	**4.42**	5109.5	0	3236377	717781
	3	**30.81**	3641.8	445.1	0	713954	21950
	4	**32.15**	7035.8	989.7	0	1070376	37831
60	0	**29.34**	6689.7	1915.8	0	893467	121026
	1	**25.93**	8391.6	1396.4	0	1247026	78818
	2	**31.50**	**3.11**	5695.5	0	2843086	576198
	3	**33.31**	2079.4	698.2	0	327560	36760
	4	**31.54**	**1.26**	1481.9	0	1980090	57193
70	0	**43.35**	5854.2	922.4	0	609612	23398
	1	**31.64**	6081.8	837.8	0	687282	44647
	2	**44.28**	**1.40**	4989.8	0	1990577	424925
	3	**32.13**	10784.9	1386.6	0	1390593	46720
	4	**43.68**	**0.27**	1941.8	0	1639345	79158
80	0	**40.99**	6160.8	1165.0	0	611669	44207
	1	**35.29**	2605.7	553.3	0	208022	12637
	2	**39.96**	7809.5	1243.5	0	1013707	59022
	3	**31.78**	2239.0	680.0	0	190454	4332
	4	**100.00**	3236.0	712.4	0	296933	23032
90	0	**46.86**	4904.8	554.1	0	367686	12991
	1	**48.88**	5686.4	973.6	0	414750	46672
	2	**50.27**	**0.58**	2337.1	0	1507747	133492
	3	**47.68**	**1.54**	1398.6	0	1164660	57569
	4	**100.00**	**1.99**	3963.4	0	1181471	227296