Strengthened Benders Cuts for Stochastic Integer Programs with Continuous Recourse

Merve Bodur*[1], Sanjeeb Dash†[2], Oktay Günlük‡[2], and James Luedtke**[3]

[1]Department of Mechanical and Industrial Engineering, University of Toronto
[2]Mathematical Sciences Department, IBM T.J. Watson Research Center
[3]Department of Industrial and Systems Engineering, University of Wisconsin-Madison

February 9, 2017

Abstract

With stochastic integer programming as the motivating application, we investigate techniques to use integrality constraints to obtain improved cuts within a Benders decomposition algorithm. We compare the effect of using cuts in two ways: (i) cut-and-project, where integrality constraints are 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 constraints are used to derive cuts directly in the Benders reformulation. For the case of split cuts, we demonstrate that although these approaches yield equivalent relaxations when considering a single split disjunction, cut-and-project yields stronger relaxations in general when using multiple split disjunctions. Computational results illustrate that the difference can be very large, and demonstrate that using split cuts within the cut-and-project framework can significantly improve the performance of Benders decomposition.

Keywords. Two-stage stochastic integer programs, Benders decomposition, Split cuts

1 Introduction

We study large-scale mixed-integer programs arising from two-stage stochastic integer programs (SIP) with continuous recourse (second-stage) variables and study techniques that use integrality constraints on the first-stage variables to strengthen Benders inequalities generated within the decomposition algorithm. We are interested in problems having the following block-structure form

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

*bodur@mie.utoronto.ca †sanjeebd@us.ibm.com ‡gunluk@us.ibm.com **jrluedt1@wisc.edu

where, $\mathcal{K}$ is a finite index set for scenarios, $c \in \mathbb{R}^n, b \in \mathbb{R}^l$, $0 \leq q \leq n$, and for each $k \in \mathcal{K}$, $p_k > 0$ (the probability of occurrence of scenario k), $h^k \in \mathbb{R}^m$, $d^k \in \mathbb{R}^t$, 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 as a mixed-integer program may be difficult due to its size. Therefore the stochastic programming literature has focused on methods to solve (1) via decomposition, in which the problem is decomposed into smaller subproblems, typically one per scenario. Two common decomposition approaches are *dual decomposition* [8, 9], and *Benders decomposition* – also called the L-shaped method [24]. Both methods solve a relaxation of (1) to obtain lower bounds and use these bounds in a branch-and-bound framework. In dual decomposition, a separate copy of the first-stage variables is created for each scenario and these copies are equated via new constraints. These so-called *nonanticipativity* constraints are then relaxed via Lagrangian relaxation, leading to a separate subproblem for each scenario. As these subproblems are mixed-integer programs, the bound obtained via Lagrangian relaxation is typically stronger than the bound obtained from the continuous relaxation of (1). Solving the Lagrangian dual, however, can be time-consuming due to the need to solve many mixed-integer programs. Benders decomposition can be used within an LP based branch-and-bound method where a *Benders reformulation* of (1) is solved by decomposing it into a master linear program consisting of the first-stage variables and LP subproblems corresponding to individual scenarios. More precisely, the Benders reformulation contains the first-stage decision variables x, and additional variables, z_k for $k \in \mathcal{K}$, that represent the expected cost of the second-stage problem. Cuts defining these variables are added by projecting the second-stage variables out of the formulation, which is accomplished by solving an LP for each scenario.

While Benders decomposition does not require solving MIP subproblems as in dual decomposition, a pure Benders decomposition algorithm may yield a weak relaxation, leading to a large branch-and-bound tree. Thus, an important question is how to use integrality constraints to obtain improved LP relaxations within the Benders decomposition framework. A natural approach (we call it *project-and-cut*) is to generate Benders cuts to approximate the feasible region of the Benders reformulation, and then use integrality information to derive strong valid inequalities for this formulation. In an alternative approach, which we refer to as *cut-and-project*, valid inequalities that use integrality constraints *of the first-stage decision variables* are derived and added to the second-stage subproblems. The resulting tightened linear programs are then used to obtain stronger Benders cuts. An advantage of this approach over deriving integrality-based cuts in the Benders reformulation is that cuts exploiting subproblem structure can easily be incorporated.

In this paper, we compare the strengths of relaxations obtained using these two approaches when the valid inequalities used are split cuts. We first consider a general setting where split cuts can either be added to improve a formulation in an extended variable space, or to the projection of this formulation. We demonstrate that the relaxations are equivalent when considering a single split disjunction, but when considering multiple split disjunctions, obtaining the cuts in the extended space can lead to a strictly better relaxation. We derive a linear program that can be used to generate cuts in the projected space that are as strong as those that can be obtained by working in the extended space. The size of this linear program grows linearly with the number of splits, so it does not appear to be a practical tool, but we still use it to measure the effect of cut-and-project.

For SIP problems, it is impractical to generate split cuts based on the extensive formulation of SIP. We therefore instead generate (rank-1) split cuts based on the constraints $T^k x + W^k y^k \geq h^k$ defining each scenario (and the constraints $Ax \geq b$ and the integrality of x) and augment

these constraints with the split cuts in an iterative fashion. These additional constraints lead to strengthened Benders cuts for the scenario. We compare this approach with the effect of (rank-1) split cuts based on (unstrengthened) Benders cuts approximating the LP relaxation of Benders reformulation. For an SIP with a single second-stage scenario, the previous paragraph implies that cut-and-project dominates project-and-cut, at least if split cuts are used and separated exactly; we use a heuristic generator (for rank-1 GMI cuts [12]) in both cases. In the presence of multiple scenarios, neither approach is dominant. Project-and-cut can derive split cuts based on Benders cuts from multiple scenarios; the scenario-based cut-and-project framework only derives split cuts based on individual scenario subproblems. On the other hand, the Benders cuts generated in project-and-cut may be an incomplete representation of the LP relaxation of the Benders reformulation.

We conduct an extensive numerical study on two different SIP problems to compare these two approaches. On the first test problem, a stochastic facility location problem, scenario-based split cuts in the extended space enable all instances to be solved efficiently, whereas using split cuts based on the (incomplete) master LP approximation of the Benders reformulation yields large optimality gaps after the time limit. On the second test problem, a stochastic network interdiction problem [20], the results were more mixed. Split cuts in the extended space significantly improved Benders decomposition when no other cuts are used. However, for this problem class we found that using cuts derived from the Benders reformulation was also very effective. Thus, we conclude either approach may be important, depending on the problem structure.

The cut-and-project approach has previously been used in [4, 8, 15, 19, 23, 25], though with different classes of valid inequalities, and not for continuous recourse problems. The papers [15] and [25] are most closely related to our work, and we provide a detailed comparison of our work to these papers in Section 3.4. The primary contribution of our work to this stream of literature is to provide understanding, through theoretical results and an extensive computational study, of the strength and limitations of the cuts generated using the cut-and-project approach in comparison to the project-and-cut approach. In our computational study, we demonstrate the applicability and effectiveness of the cut-and-project approach on instances with continuous recourse. In addition, the test instances we consider are larger than those considered in the papers mentioned above. Specifically, the network interdiction instances that we consider have up to 320 binary first-stage variables and over 450 scenarios, and our facility location instances have up to 50 binary first-stage variables and 1500 scenarios. The previous studies mentioned above except [23] and [4] consider at most 15 binary first-stage variables. [23] consider a set of test instances with up to 78 binary first-stage variables but only 23 scenarios, and [4] test on instances with up to 40 binary first-stage variables, but are only able to solve instances with 50 or fewer scenarios.

The remainder of this paper is organized as follows. In section 2 we present our general results comparing the use of split cuts in an extended vs. a projected variable space. We remark that the significance of these results is not restricted to their application in stochastic integer programming. In section 3 we describe Benders decomposition for SIP in more detail, and interpret the results of section 2 in this context. We describe the results of our numerical study in section 4.

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

In this section we first formally describe split cuts and then compare the effect of applying them to a polyhedral set with that of applying them to a formulation of this set in a higher dimensional space. In the two-stage stochastic programming context, this relates to comparing the effect of applying split cuts to the extensive formulation with that of applying them to the Benders reformulation. We discuss this more precisely in Section 3.

2.1 Split cuts

Let $P \subseteq \mathbb{Z}^n \times \mathbb{R}^q$ be a mixed integer set defined on variables (x, z) and let P^{LP} denote its linear relaxation. Let $\pi \in \mathbb{Z}^n$ be a row vector, $\gamma \in \mathbb{Z}$ and let

$$S = \{(x, z) \in \mathbb{R}^{n+q} : \gamma + 1 > \pi x > \gamma\},$$

be the split set associated with (π, γ) in $\mathbb{R}^{n+q}$. Then, an inequality $cx + dz \geq f$ is called a *split cut* for P^{LP} generated by S if it is valid for $\text{conv}(P^{LP} \setminus S)$. In other words, $cx + dz \geq f$ is a split cut if it is valid for both

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

The *split closure* of P is the collection of all points in P^{LP} that satisfy all split cuts generated by split sets S for all $\pi \in \mathbb{Z}^n$, and $\gamma \in \mathbb{Z}$. 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 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 [7]. 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} [6, 12, 14].

2.2 Projected split cuts

We next compare the effect of applying split cuts to a polyhedral set P^{LP} with that of applying them to an *extended formulation* of P^{LP}. More precisely, let $Q \subseteq \mathbb{Z}^n \times \mathbb{R}^q \times \mathbb{R}^t$ be a mixed integer set defined on variables (x, z, y) such that its continuous relaxation satisfies $P^{LP} = \text{Proj}_{(x,z)}\big(Q^{LP}\big)$. (Here $\text{Proj}_{(x,z)}\big(S\big)$ stands for the orthogonal projection of the set S in the space of (x, z) variables.) Q^{LP} is called an *extended formulation* of P^{LP}. Note that this also implies that $P = \text{Proj}_{(x,z)}\big(Q\big)$. The space of an extended formulation is referred to as "extended (variable) space" throughout the paper.

For $\pi \in \mathbb{Z}^n$ and $\gamma \in \mathbb{Z}$, let $S' = \{(x, z, y) \in \mathbb{R}^{n+q+t} : \gamma + 1 > \pi x > \gamma\}$ denote split sets associated with (π, γ) in $\mathbb{R}^{n+q+t}$. Note that the split set S' and the set S are generated by the same $\pi \in \mathbb{Z}^n$ and $\gamma \in \mathbb{Z}$, but in different spaces. Split cuts for Q^{LP} generated by S' and the split closure of Q are defined as in Section 2.1.

We next show that the projection of $Q^{LP} \setminus S'$ in the space of the (x, z) variables is the same as $P^{LP} \setminus S$.

Lemma 1. *Let $S \in \mathbb{R}^{n+q}$ and $S' \in \mathbb{R}^{n+q+t}$ be two split sets generated by the same $\pi \in \mathbb{Z}^n$ and $\gamma \in \mathbb{Z}$. Then, $P^{LP} \setminus S = \text{Proj}_{(x,z)}\big(Q^{LP} \setminus S'\big)$.*

Proof. 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 $\text{Proj}_{(x,z)}\big(Q^{LP} \setminus S'\big) \subseteq P^{LP} \setminus S$. □

Clearly, Lemma 1 implies that $\text{conv}(P^{LP} \setminus S) = \text{conv}(\text{Proj}_{(x,z)}\big(Q^{LP} \setminus S'\big))$ and therefore split cuts generated by (essentially) the same split set have the same effect in the projected space, 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 as we show in the next Lemma.

Lemma 2. *Let $S_1, \ldots, S_k$ be $k > 1$ split sets for P and let $S'_1, \ldots, S'_k$ be the corresponding split sets for Q. Then*

$$\bigcap_{i=1}^{k} \operatorname{conv}(P^{LP} \setminus S_i) \supseteq \operatorname{Proj}_{(x,z)}\Big(\bigcap_{i=1}^{k} \operatorname{conv}(Q^{LP} \setminus S'_i)\Big),$$

and the inclusion is strict in some cases.

Proof. Proof. We first show the inclusion "$\supseteq$" and then present an example when the inclusion is strict. Let $(\hat{x}, \hat{z}) \in \operatorname{Proj}_{(x,z)}\big(\bigcap_{i=1}^{k} \operatorname{conv}(Q^{LP} \setminus S'_i)\big)$, then there exists $\hat{y} \in \mathbb{R}^t$ such that $(\hat{x}, \hat{z}, \hat{y}) \in \operatorname{conv}(Q^{LP} \setminus S'_i)$ for $i = 1, \ldots, k$. Therefore, by Lemma 1, we have $(\hat{x}, \hat{z}) \in \operatorname{conv}(P^{LP} \setminus S_i)$ for $i = 1, \ldots, k$ 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} and Q^{LP} be defined as

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

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

(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 two split sets for P, and S'_1 and S'_2 be the corresponding split sets for Q.

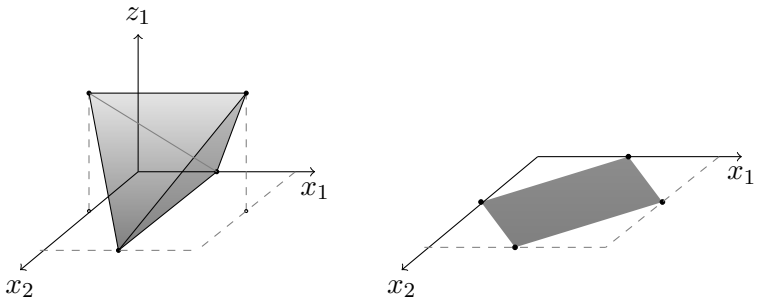

Figure 1: The set Q^{LP} and its projection P^{LP}

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

□

The split closure of the set P in the proof above is in fact equal to $\{(1/2, 1/2)\}$ [11] and Lemma 2 establishes that the projection of the split closure of Q is strictly contained in the split closure of P even though $\operatorname{Proj}_{(x,z)}\big(Q^{LP}\big) = P^{LP}$. Consequently, deriving split cuts in the extended space rather than in the projected space can yield a stronger relaxation. We note that a similar observation is made by [18] in the context of mixed-integer nonlinear programming but we are not aware of this observation having been made in the context of stochastic integer programming.

We note that the Lovász-Schrijver [17] lift-and-project operator $N(P)$ can be viewed as applying split cuts – based on split sets of the form $\{x \in \mathbb{R}^n : 0 < x_i < 1\}$ – to an extended formulation. In this setting, the y variables in Q represent linearized products of pairs of variables from x, z and the constraints $y_{ii} = y_{0i}$ in [17][pp. 169] (corresponding to a linearization of $x_i^2 = x_i$) can be obtained as split cuts in the extended space.

2.3 Working in the projected space

We next study how to solve the separation problem in the projected space when an explicit formulation for P^{LP} is not available. This is quite relevant in two-stage stochastic programming as the LP relaxation of the extensive formulation is given explicitly whereas the corresponding Benders reformulation is not. Let Q^{LP}, the extended formulation of P^{LP}, be 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 linear 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:

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

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

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. Proof. Let Q'_i denote $\mathrm{conv}(Q^{LP} \setminus S'_i)$. 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 $\hat{P} = \mathrm{Proj}_{(x,z)} (\bigcap_{i \in I} Q'_i))$. 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, a 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. In the stochastic programming context, this implies that split cuts for the Benders reformulation can be generated without computing the complete reformulation.

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 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*

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

Proof. Proof. Let Q'_i denote $\text{conv}(Q^{LP} \setminus S'_i)$ and assume that there exists a point $(\bar{x}, \bar{z}) \in \text{Proj}_{(x,z)} \big(\bigcap_{i \in \bar{I}} Q'_i \big)$. Then, by definition, there exist a point $(\bar{x}, \bar{z}, \bar{y}) \in \big(\bigcap_{i \in \bar{I}} Q'_i \big) \big)$. 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, $(\bar{x}, \bar{z}, \bar{y}) \in \big(\bigcap_{i \in I} Q'_i \big) \big)$ and consequently, $(\bar{x}, \bar{z}) \in \hat{P}$.

□

Therefore, removing the variables and constraints associated with split sets which exclude the given point (i.e. all indices in $I \setminus \bar{I}$) from the Separation LP (8) yields an equivalent formulation with reduced size.

3 Benders Decomposition and Cut-and-Project

In the previous section we established that applying splits cuts to an extended formulation before projecting it to a lower dimensional space yields a stronger relaxation than first projecting it to the lower dimensional space and then adding split cuts. In the context of two-stage stochastic programming, this means that adding split cuts to the extensive formulation yields better bounds than adding split cuts to the Benders reformulation. While working with the extensive formulation directly is typically not practical, cut-and-project can be applied it to each scenario separately in the Benders decomposition framework. We start with a slightly modified version of the extensive formulation described in Section 1:

$$\begin{aligned} \min \quad & cx + pz & & & \text{(9)} \\ \text{s.t.} \quad & Ax \geq b, & & x \in \mathbb{Z}^q_+ \times \mathbb{R}^{n-q}_+, \\ & z_k \geq d^k y^k, & & z_k \in \mathbb{R}_+ \text{ for all } k \in \mathcal{K}, \\ & T^k x + W^k y^k \geq h^k, & & y^k \in \mathbb{R}^t_+ \text{ for all } k \in \mathcal{K}, \end{aligned}$$

where the new variable z_k denotes the cost associated with the second-stage problem in scenario $k \in \mathcal{K}$. As before, variables x are the first-stage variables and y^k, for $k \in \mathcal{K}$, are the recourse variables.

We next describe Benders decomposition and how it is used to solve (9). We then discuss scenario-based cut-and-project for Benders decomposition and point out its limitations.

3.1 Benders decomposition

Benders decomposition solves a reformulation of (9) where the recourse variables y^k are projected out leading to a reformulation defined on the first stage variables only. 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

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

Letting $X = \{x \in \mathbb{Z}^q_+ \times \mathbb{R}^{n-q}_+ : Ax \geq b\}$, Benders decomposition is a method for solving (9) by solving the *Benders reformulation*

$$\min\ \left\{cx + pz\ :\ x \in X,\ (x, z_k) \in \text{Proj}_{(x,z_k)}\big(Q_k^{LP}\big) \text{ for } k \in \mathcal{K}\right\}. \tag{10}$$

When $|\mathcal{K}|$ is large, this reformulation reduces the variable space significantly. Although $\text{Proj}_{(x,z_k)}\big(Q_k^{LP}\big)$ 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.

Modern implementations of the Benders decomposition algorithm for a problem with integer first-stage variables work by building a single branch-and-bound tree where node relaxations are solved via Benders cuts. Let $P_k^{LP} := \text{Proj}_{(x,z_k)}\big(Q_k^{LP}\big)$ for $k \in \mathcal{K}$ and let $B_k \supseteq P_k^{LP}$ represent the polyhedron defined by the Benders cuts from scenario k currently in the master LP relaxation. Initially, $B_k = \mathbb{R}^{n+1}$. Given a node (including the root) relaxation solution $(\bar{x}, \bar{z})$, if $\bar{x}$ satisfies the integrality constraints, we determine if $(\bar{x}, \bar{z}_k) \in P_k^{LP}$ for $k \in \mathcal{K}$ by solving the second-stage LP subproblem for scenario $k \in \mathcal{K}$:

$$f_k(\bar{x}) = \min\ \{z : z \geq d^k y,\ W^k y \geq h^k - T^k \bar{x}, y \geq 0\}. \tag{11}$$

First consider the case when the LP in (11) has a feasible solution. In this case if $\bar{z}_k \geq f_k(\bar{x})$ then $(\bar{x}, \bar{z}_k) \in P_k^{LP}$. Otherwise, if $(1, \bar{\pi})$ is an optimal dual solution to the linear program (11), then $(\bar{x}, \bar{z})$ violates the Benders cut

$$z_k + \bar{\pi} T^k x \geq \bar{\pi} h^k. \tag{12}$$

If the LP in (11) is infeasible, then there exists a dual vector $(0, \bar{\pi}) \geq 0$ that certifies the infeasibility of the LP. In this case, $\bar{\pi} W^k = 0$ and a so-called feasibility cut of the form $\bar{\pi} T^k x \geq \bar{\pi} h^k$ can be constructed. In either case if a violated cut is found, it is added to the description of B_k and the node master LP is solved again. This process is repeated until either $(\bar{x}, \bar{z}_k) \in P_k^{LP}$ for all $k \in \mathcal{K}$, at which point $(\bar{x}, \bar{z}_k)$ is an optimal solution of the node LP, or $\bar{x}$ violates an integrality constraint, at which point we can either continue generating Benders cuts, or we can branch on a fractional integer variable.

3.2 Scenario-based cut-and-project

To motivate the scenario-based cut-and-project approach, we first compare the LP relaxation obtained by Benders decomposition with that obtained via dual decomposition. Let

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

and consider the problem

$$\min\; \left\{cx + pz \;:\; x \in X,\; (x, z_k) \in \text{Proj}_{(x,z_k)}\big(Q_k^{IP}\big) \text{ for } k \in \mathcal{K}\right\}, \tag{13}$$

where Q_k^{LP} in (10) is replaced with Q_k^{IP}. Clearly, (13) is equivalent to (9). However, the bound from the LP relaxation of (13), obtained by replacing X with its continuous relaxation X^{LP}, dominates that of (10). In the dual decomposition method, one solves the continuous relaxation of (13) to obtain lower bounds. This method leads to smaller enumeration trees compared to Benders decomposition due to stronger lower bounds, which are obtained at the expense of solving MIP subproblems.

The idea of scenario-based cut-and-project is to obtain stronger Benders cuts that ideally yield a relaxation closer to that of the dual decomposition relaxation, but without solving MIP subproblems. Specifically, we reformulate (10) by replacing Q_k^{LP} with a tighter set Q_k^S which satisfies $Q_k^{IP} \subseteq Q_k^S \subseteq Q_k^{LP}$. The polyhedron Q_k^S is obtained by augmenting Q_k^{LP} with valid inequalities for Q_k^{IP}, which are derived by using integrality information of first-stage variables. When the continuous relaxation of this new formulation is solved via Benders decomposition, one obtains stronger Benders cuts, i.e., cuts that are valid for $\text{Proj}_{(x,z_k)}\big(Q_k^{IP}\big)$ but are potentially not valid for $\text{Proj}_{(x,z_k)}\big(Q_k^{LP}\big)$. Ideally, one should use Q_k^{IP} instead of Q_k^S here, but characterizing the linear description of Q_k^{IP}, in general, is a difficult task.

The deterministic version of many SIP problems are well-studied integer programs, in which case one can use problem-specific structure to obtain Q_k^S. For instance, for capacitated network design problems, *cutset inequalities* can be used [5]. This approach would require implementation of different cut separation routines for different problem classes.

Alternatively, one can obtain Q_k^S by strengthening Q_k^{LP} with general purpose cutting-planes that do not use problem structure. In our computational study, presented in Section 4, we augment Q_k^{LP} with split cuts to obtain Q_k^S. In the integer programming literature [1, 2, 13], split cuts are known to be very effective at approximating the integer hulls of practical problems. In addition, there are practical heuristics [6, 12, 14] that find violated *Gomory mixed-integer* (GMI) cuts that form a subclass of split cuts.

We next discuss how we implement scenario-based cut-and-project. Suppose that we have solved a second-stage subproblem (11) for a scenario $k \in \mathcal{K}$ and obtained a solution (z^*, y^*). We then try to find a valid inequality of the form $ax + by^k \geq c$ for the set Q_k^{IP} that cuts off the solution $(\bar{x}, z^*, y^*)$. If such an inequality is found, then T^k, W^k and h^k are augmented by a, b and c respectively, and the inequality $by^k \geq c - a\bar{x}$ is added to the LP (11) which is re-solved to get updated values of z^*, y^*, but not $\bar{x}$. This process can be repeated as long as more violated inequalities are added to (11). At the end, a Benders cut, $z_k + \bar{\pi}T^k x \geq \bar{\pi}h^k$, or a feasibility cut, $\bar{\pi}T^k x \geq \bar{\pi}h^k$, is formed and added to the master LP relaxation if it cuts off the current relaxation solution $(\bar{x}, z^*)$. We emphasize that the matrices W^k, T^k and the vector h^k are not fixed as they are augmented with new rows corresponding to the inequalities added in this procedure. The dual vector $\bar{\pi}$ also increases in size.

3.3 Limitations of scenario-based cut-and-project

Let the LP relaxation of the extensive formulation of the SIP be

$$Q^{LP} = \left\{(x, z, y) \,:\, x \in X^{LP},\ (x, z_k, y^k) \in Q_k^{LP} \text{ for all } k \in \mathcal{K}\right\}$$

and let

$$P^{LP} = \left\{(x, z) \,:\, x \in X^{LP},\ (x, z_k) \in P_k^{LP} \text{ for all } k \in \mathcal{K}\right\}$$

denote its projection in the space of first-stage variables. In addition, let $SC(\cdot)$ denote the split closure of a polyhedral set. So far, we established that $\text{Proj}_{(x,z)}\big(SC(Q^{LP})\big)$ is always contained in $SC(P^{LP})$ and that the containment is sometimes strict. The scenario-based cut-and-project approach, however, does not work with $\text{Proj}_{(x,z)}\big(SC(Q^{LP})\big)$, but instead works with an approximation of the set $\text{Proj}_{(x,z)}\big(Q^S\big)$ where

$$Q^S = \left\{(x, z, y) \,:\, x \in X^{LP},\ (x, z_k, y^k) \in SC(Q_k^{LP}) \text{ for all } k \in \mathcal{K}\right\}.$$

Clearly, in some cases we have

$$\text{Proj}_{(x,z)}\big(SC(Q^{LP})\big) = \text{Proj}_{(x,z)}\big(Q^S\big) \subset SC(P^{LP})$$

for example when $|K| = 1$ and $X^{LP} = \mathbb{R}^n$. We next present an example to show that it is also possible to have

$$SC(P^{LP}) \subset \text{Proj}_{(x,z)}\big(Q^S\big).$$

Consider the following set

$$Q^{LP} = \left\{(x, z) \in \mathbb{R}^2 \times \mathbb{R}^2 \,:\, (x, z_1) \in Q_1^{LP},\ (x, z_2) \in Q_2^{LP}\right\}$$

where $Q_1^{LP} = \text{conv}\{(0,0,0), (1,0,0), (0,1,0), (0,0,1)\}$, and $Q_2^{LP} = \text{conv}\{(0,0,0), (1,1,0), (0,1,0), (0,1,1)\}$ (see Figure 2). Clearly, both Q_1^{LP} and Q_2^{LP} are integral, and consequently, $SC(Q_1^{LP}) = Q_1^{LP}$ and $SC(Q_2^{LP}) = Q_2^{LP}$, implying $\text{Proj}_{(x,z)}\big(Q^S\big) = \text{Proj}_{(x,z)}\big(Q^{LP}\big)$. However, $P^{LP} = \text{conv}\{(0,0), (0,1), (1/2, 1/2)\}$ and $SC(P^{LP}) = \text{conv}\{(0,0), (0,1)\}$ and therefore $SC(P^{LP})$ is equal to the integer hull of P^{LP}.

In conclusion, while the cut-and-project approach with split cuts improves upon the basic Benders decomposition approach, split cuts in the projected space can still be useful. In section 4, we discuss this further in a computational setting.

3.4 Related literature

There is a significant amount of recent work studying valid inequalities for SIP having integer or mixed-integer recourse variables [8, 10, 15, 19, 21, 22, 23, 25]. Much of the focus of this work has been the investigation of methods that use cutting planes to approximate the value function of the second-stage program, with the goal of obtaining a finitely convergent algorithm.

In contrast, we are focused on problems with (mixed-)integer first-stage variables, but *continuous* recourse variables. For such problems, a finitely convergent algorithm can be obtained using standard Benders cuts, but obtaining stronger cuts can still be important for improving computational performance. The cut-and-project approach can also be applied to improve the relaxations of stochastic integer programs with integer recourse variables. In fact, cut-and-project provides a unified view of some of the previous work on valid inequalities for SIP with integer recourse

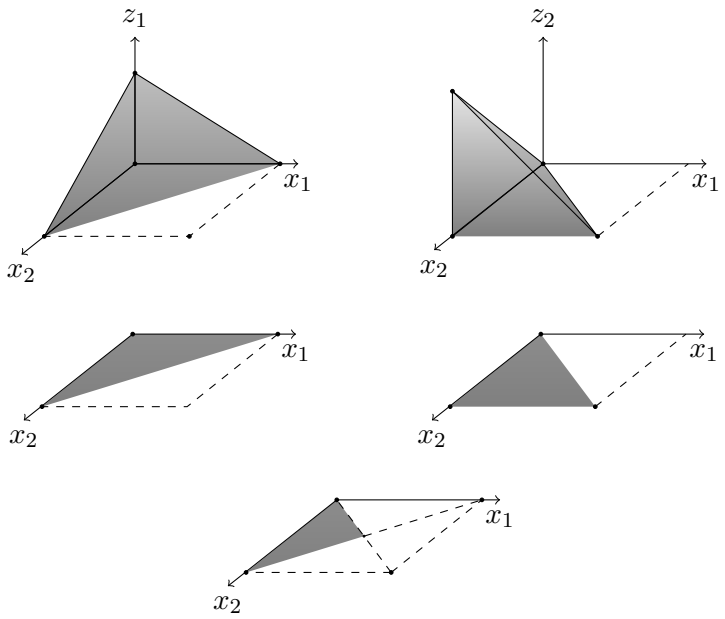

Figure 2: Sets Q_1^{LP}, Q_2^{LP} on top, their projections P_1^{LP}, P_2^{LP} in the middle and $P_1^{LP} \cap P_2^{LP}$ on the bottom.

variables, where different studies have used different classes of cuts in the subproblem space. In particular, lift-and-project split cuts (split cuts using only single variable disjunctions) were used by [8] and extended and implemented by [23]. [19] used the so-called *Fenchel cuts*, which are inequalities obtained through an explicit use of the equivalence between optimization and separation, while [4] extended them to *scenario-wise Fenchel cuts.*

The papers by [15] and [25] are most closely related to our particular implementation of cut-and-project, since they also use Gomory mixed-integer (GMI) cuts. [15] and [25] refer to their cuts as *parametric* Gomory cuts, but they can also be viewed as standard Gomory cuts which are derived from an equation that is valid for the set Q_k^{LP}. The key difference between our use of GMI cuts and that proposed in these papers is in the method for obtaining a basis from which to derive the GMI cuts. In [15], the method for constructing a basis is restricted to the case when a binary first-stage solution $\bar{x}$ has been obtained. In that approach, the basis is constructed using the basis obtained from solving the second-stage subproblem, and then extending that to a basis in the (x, y^k) space by treating all the x variables as nonbasic variables. In our use of GMI cuts and that of [25], it is not required that all first-stage variables be binary, and cuts can be applied at fractional first-stage solutions $\bar{x}$. In [25], a basis in the (x, y^k) is constructed using information from the master and the subproblem bases. We use the method of [12] to obtain a feasible basis of the the set Q_k^{LP}. This approach offers more flexibility in the choice of basis from which to derive a cut, potentially yielding stronger cuts. Another difference is that our approach always finds a basis of the original LP relaxation Q_k^{LP}, i.e., without any cuts added, so that the cuts we generate are all rank-1 GMI cuts, whereas the cuts used in [25] can be based on previously generated cuts, and thus have higher rank.

While the work described in the previous paragraphs fits within the cut-and-project framework, it appears that our work is the first to consider using such an approach for a problem with continuous recourse variables.

4 Computational Experiments

In this section, our goal is to quantify the benefits of scenario-based cut-and-project where the cutting planes used are split cuts. We compare the root bound improvement of this method over the bound obtained by solving the LP relaxation of the extensive formulation (1). Furthermore, we compare this approach to a project-and-cut approach, where we add split cuts to the master problem approximating the Benders reformulation. We also study how these approaches compare when solving SIP instances to optimality in a branch-and-bound setting, both with and without general-purpose cuts provided by a commercial solver.

The problems we consider satisfy $X^{LP} = \{x \in \mathbb{R}^n_+ : Ax \geq b\} \subseteq \text{Proj}_{(x)}\big(Q_k^{LP}\big)$ for all $k \in \mathcal{K}$, i.e., there always exists a feasible recourse decision for every feasible continuous solution to the first-stage constraints. This is a slightly stronger property than *relatively complete recourse*. Consequently, in our computational experiments we do not need to consider "feasibility cuts" and only need Benders cuts.

4.1 Computational approach

Our primary implementation uses a rank-1 GMI heuristic (called FEAS in [12]) to separate split cuts in each case. We discuss our adaptation of this heuristic in more detail in Section 4.2. We compare the following decomposition methods for solving the root node of (1):

- "BEN": Solve the linear programming relaxation using only Benders cuts.
- "SP": Add split cuts in x and y^k variables to subproblem k before generating Benders cuts from that subproblem. This is what we call the scenario-based cut-and-project approach.
- "MP": Add split cuts based on the master LP approximation of the Benders reformulation (in the *projected space*).

We also compare these decomposition methods to directly solving (1) with an MIP solver – we call this "EXT".

Pseudocode for the three decomposition methods is given in Figure 3. In this figure, steps which are labeled with [SP] and [MP] are executed only in the SP and MP methods, respectively. In all of the decomposition methods, we work with a multi-cut version, i.e., for each scenario k, there is a variable z_k denoting the objective function contribution of the y^k variables, rather than just one variable for the second-stage objective function. Each Benders cut is derived from a single scenario (say k) in the variables x and z_k. We initialize the master problem LP (or master LP) to

$$\min\{cx + \sum_{k\in\mathcal{K}} p_k z_k \ : \ Ax \geq b,\ x \in \mathbb{R}^n_+, z \geq z^{LB}\}, \tag{14}$$

where

$$z_k^{\text{LB}} := \min_{x,y}\{d^k y \ : \ Ax \geq b,\ T^k x + W^k y \geq h^k,\ x \in \mathbb{R}^n_+,\ y \in \mathbb{R}^t_+\} \text{ for } k \in \mathcal{K}.$$

Of course, in the first solution $(\bar{x}, \bar{z})$ to the master LP, $\bar{z}_k = z_k^{LB}$. For the pure Benders algorithm BEN, at each subsequent iteration, for the current $(\bar{x}, \bar{z})$ and for each $k \in \bar{\mathcal{K}} \subseteq \mathcal{K}$, we solve the scenario subproblem (11), and obtain a Benders cut as described after equation (11). We discuss how the set $\bar{\mathcal{K}}$ is chosen in Section 4.3. Each violated Benders cut is added to the master LP (we call this process "one round of cuts"), which is re-solved to update $(\bar{x}, \bar{z})$. This process ends when no Benders cut can be found, at which point the objective value equals the LP relaxation value of (1). In the SP method, after solving the scenario subproblem as in BEN to obtain a solution $\bar{y}^k$, we invoke a GMI separation heuristic once to find split cuts violated by $(\bar{x}, \bar{y}^k)$. Note that these cuts have split-rank 1 as they are based on the original constraints. The constraints associated

with the scenario k are augmented with these cuts (thus T^k, W^k and h^k are updated), and the updated scenario subproblem is re-solved. At this point a strengthened Benders cut is generated, and is added to the master problem if it is violated by $(\bar{x}, \bar{z})$. In the MP method, in each outer iteration (where we solve the master LP to obtain $(\bar{x}, \bar{z})$) we generate violated (unstrengthened) Benders cuts as in BEN; separately, we use a GMI heuristic to find violated split cuts derived from the current master problem constraints.

```
Initialize Master LP as (14);
repeat
    Solve current Master LP to obtain solution (x̄, z̄);
    for k ∈ K̄ do
        Solve scenario k subproblem LP (11) to obtain solution ȳ^k;
        [SP] Search for split cuts that cuts off (x̄, ȳ^k);
        [SP] If violated split cuts are found, augment LP (11) for scenario k and re-solve;
        Form Benders cut (12) and add it to Master LP if it is violated by (x̄, z̄_k);
    end
    [MP] Search for split cuts based on current master problem, and add to master LP if
     violated by (x̄, z̄);
until No violated cut added to Master LP or early termination criterion detected;
```

Figure 3: Pseudocode of decomposition methods for solving the root node. Parts of the code labeled with [SP] and [MP] are executed only for the SP and MP methods, respectively.

As we use a heuristic split cut separation procedure, it may be difficult to conclude that one approach yields stronger relaxations than the other. For example, we may generate cuts based on different split disjunctions in the SP and MP methods. We therefore also consider the following methods where we exactly separate split cuts (via cut generation LPs) based on *simple split sets* of the form $S = \{\gamma < x_j < \gamma + 1\}$ involving a single variable:

- "SP-CGLP": Instead of heuristic GMI cuts in the SP method, generate split cuts for Q_k^{IP} and augment scenario subproblems.
- "PR-CGLP": Use the cut generation LP (8) with a *single* split at a time to separate cuts in the (x, z_k) space that are valid for scenario k.

Specifically, SP-CGLP is the same as SP, except that when searching for split cuts in the subproblem space, we perform exact separation for each simple split set, rather than using the GMI heuristic. Thus SP-CGLP is an exact variant of SP, but for a fixed set of disjunctions. Aside from restricting to simple split disjunctions, PR-CGLP also differs from MP in that it derives split cuts based on all Benders cuts derivable from one scenario at a time, whereas MP derives split cuts based on Benders cuts from multiple scenarios simultaneously.

4.2 Separating split cuts

Our precise approach to finding split cuts for Q_k^{IP} that cut off the solution $(\bar{x}, \bar{z}, \bar{y})$ is based on the FEAS heuristic for rank-1 GMI cuts in [12]. We assume that $\bar{x}$ is a basic solution to $Ax \geq b$ and some Benders cuts. For $\bar{x}$, we solve (11) (in equality form) with the simplex method and get an optimal basis B and associated solution $(\bar{y}, \bar{s})$, where s stands for the slack variables in (11). Let (y_B, s_B) stand for the basic y, s variables in this solution. Consider the IP

$$\min\{cx + p_k d^k y : Ax \geq b, T^k x + W^k y - s = h^k, y, s \geq 0, x \in \mathbb{Z}_+^q \times \mathbb{R}_+^{n-q}\}. \tag{15}$$

The basic variables (y_B, s_B) augmented with the basic x variables contain a basic solution of the LP-relaxation of (15) (see [12]). We search for a basic solution from among these variables (if a basic $\bar{x}$ is not available, we use the positive x variables as a proxy for basic variables) by setting the remaining variables to zero in the LP-relaxation of (15) and solving it. We take the new optimal basis, and generate GMI cuts and keep those that are violated by $(\bar{x}, \bar{z}, \bar{y})$. We depict this process in Figure 4, where the polyhedron represents the LP relaxation of (15), and, for convenience we have dropped the constraints $Ax \geq b$ and assume $x \in \mathbb{R}^1, y \in \mathbb{R}^1$, and $z = y$. The thick vertical line inside the polyhedron depicts $W^k y \geq h^k - T^k \bar{x}, y \geq 0$. The above process corresponds to looking for a vertex of (15) (marked by a circle) on the face of (15) where $\bar{y}$ lies, and generating a GMI cut from this vertex denoted by a horizontal dashed line (and based on the split set between the vertical dashed lines). GMI cuts in MP are also obtained using FEAS; we look for a subset of

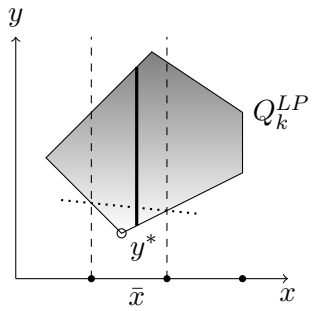

Figure 4: $Q_k^{LP} \in \mathbb{R}^2$, $z = y$, and Q_k^{LP} for $x = \bar{x}$ (as the solid line), and an associated split cut obtained as a rank-1 GMI cut from a vertex of Q_k^{LP}.

variables basic in $\bar{x}$ that define a basis of the original constraints $Ax \geq b$.

4.3 Implementation details

We solve all LPs and IPs using IBM ILOG CPLEX 12.4. All experiments are run using a single thread on a Linux workstation with 2.33 GHz Intel Xeon CPUs and 32 GB memory. When using our algorithms to obtain root bounds only, we do not impose a time limit. For all branch-and-cut runs, we set a time limit of 4 hours, and the relative MIP optimality tolerance to 0.1%. We solve the extensive formulation with default Cplex settings, and turn off presolve features for the Benders-decomposition based algorithms.

We implement our branch-and-cut algorithm using callback features of Cplex. The formulation at the end of root node consists of the initial constraints and all Benders cuts found at the root node. Subsequently, we use the LazyConstraintCallback to add *lazy* (Benders) cuts violated by an integer solution of the current formulation (found by a Cplex heuristic or returned by the node LP), and the UserConstraintCallback to add *user* cuts cutting off fractional solutions. We only generate user cuts at nodes with depth up to 10 in the branch-and-bound tree. Moreover, at every node except the root node, we generate at most one round of user cuts. We add lazy cuts only if their relative violation (the absolute violation of the cut divided by the 2-norm of the cut coefficients) is at least 10^{-5}, whereas user cuts are added if their relative violation is at least 10^{-4} at the root node and at least 10^{-3} at other nodes in the tree. For BEN and MP, only unstrengthened Benders (user) cuts are added in the tree, whereas strengthened Benders cuts are added in SP. A flow chart of our branch-and-cut algorithm is provided in the Online Supplement (available as supplemental material at http://pubsonline.informs.org/doi/10.1287/ijoc.2016.0717).

To reduce the root node solution time, we introduce an early stopping condition. We stop adding Benders cuts if the master LP objective value does not change by 0.05% in five iterations. Moreover, we limit the number of scenarios used to generate Benders cuts in a round of cuts (equivalently, we let $\bar{\mathcal{K}}$ in Section 4.1 to a strict subset of $\mathcal{K}$). We start with $\bar{\mathcal{K}} := \mathcal{K}$. For each scenario, we record the ratio of the number of iterations in which we successfully generate a violated Benders cut to the number of times we attempt to do so, and update this "success" ratio in every iteration. If in some iteration, fewer than half the scenarios return violated Benders cuts, in each subsequent iteration (including in the non-root nodes) we only consider the top 5% of scenarios (thus $|\bar{\mathcal{K}}| = 0.05|\mathcal{K}|$) by success ratio. The success ratios are updated in each iteration, so the selected scenarios can change over iterations or cut-generating rounds.

4.4 Capacitated facility location problem

Our first test problem, 'CAP', is a stochastic version of the capacitated facility location problem, as described in [16]. In this problem, facility opening decisions (yes or no) must be made before observing the realizations of random customer demands. Then, in the second stage the customer demands are fractionally allocated to the open facilities. Let $\mathcal{I}$ be the set of potential facilities, $\mathcal{J}$ be the set of customers and $\mathcal{K}$ be the set of scenarios. The first-stage facility opening decision variables are denoted by x ($x_i = 1$ if and only if the ith facility is opened), while the second-stage recourse 'flow' variables are denoted by y (y_{ij} is the amount of the jth customer's demand satisfied by facility i). We denote the opening cost and capacity of facility i, respectively, by f_i and s_i, and the cost of sending a unit of demand (or flow) from facility i to customer j by q_{ij}. The demand of customer j under scenario k is denoted by λ_j^k. We use the following formulation for this problem:

$$\min \sum_{i\in\mathcal{I}} f_i x_i + \frac{1}{|K|}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} q_{ij} y_{ij}^k \tag{16a}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}} y_{ij}^k \geq \lambda_j^k \ , \qquad j\in\mathcal{J},\ k\in\mathcal{K}, \tag{16b}$$

$$\sum_{j\in\mathcal{J}} y_{ij}^k \leq s_i x_i \ , \qquad i\in\mathcal{I},\ k\in\mathcal{K}, \tag{16c}$$

$$\sum_{i\in\mathcal{I}} s_i x_i \geq \max_{k\in\mathcal{K}} \sum_{j\in\mathcal{J}} \lambda_j^k, \tag{16d}$$

$$x\in\{0,1\}^{|I|}, y\in\mathbb{R}_+^{|I||J||K|}. \tag{16e}$$

This formulation differs slightly from the 'standard formulation' in [16] where decision variables $\bar{y}_{ij}^k = y_{ij}^k/\lambda_j^k$ give the *fraction* of customer i's demand served by facility j. We use the above formulation because there is no uncertainty in the *technology matrix* (the coefficients multiplying x_i in the second-stage constraints). For each scenario, the inequalities (16b) make sure that customer demand is satisfied, while inequalities (16c) enforce the condition that customers can only be served by open facilities. Finally, the inequalities (16d) enforce relatively complete recourse by ensuring that the total capacity of open facilities is sufficient to satisfy the total customer demand in each scenario.

We take the deterministic 'CAP' instances from J. E. Beasley's OR-Library [3] that have 50 customers and 25-50 potential facilities, and create stochastic instances from them. Let $\bar{\lambda}_j$ be the deterministic demand of customer j in the data. We construct the demands of the stochastic instances by sampling 250, 500 or 1500 scenarios from a normal distribution with mean taken from the corresponding deterministic instance. 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 and generate stochastic demands $\lambda_j \sim N(\mu_j, \sigma_j)$.

Computational experiments with the root bound. We first investigate the gap given by solving the root node of the master problem by the methods BEN, SP, MP, SP-CGLP and PR-CGLP. Table 1 reports the average percentage root gap (calculated as $(\nu_{IP} - \nu_{LP}) * 100/\nu_{IP}$ where ν_{IP} denotes the optimal value of the MIP and ν_{LP} denotes the master LP bound) obtained by each of above methods over sets of CAP instances having similar properties. We only report the case for 250 scenarios; the results with 500 and 1500 scenarios are very similar.

We find that PR-CGLP yields a modest improvement in gap over BEN. On the other hand, SP-CGLP yields a dramatic improvement over PR-CGLP, indicating that the difference in the strength of the relaxations suggested by Lemma 2 can sometimes be very large. Finally, SP yields a root gap similar to SP-CGLP. The MP heuristic does slightly worse than PR-CGLP. These results indicate that for this problem class, using split cuts in the extended space has far more impact on the root bound than using split cuts in the projected space. In the remainder of the paper, we do not experiment with SP-CGLP and PR-CGLP as these are computationally very expensive (they do not always terminate in a day).

Computational experiments with the branch-and-bound tree. We next compare the different branch-and-cut methods defined earlier. Due to the early stopping conditions for cut-generation used in branch-and-bound, we may obtain different root gaps than those in Table 1.

Table 1: Average % root gap obtained by different methods for CAP instances with 250 scenarios. At each CAP # row, the averages of four instances are reported.

CAP #	BEN	MP	PR-CGLP	SP	SP-CGLP
101-104	22.40	20.74	18.00	0.07	0.11
111-114	8.72	8.01	7.79	0.41	0.22
121-124	18.92	18.26	16.32	0.99	1.15
131-134	25.23	24.18	21.56	0.30	1.46
Mean	18.82	17.80	15.92	0.44	0.74

In addition, after adding Benders cuts to the master problem formulation and starting the branch-and-bound phase, at the root node Cplex adds its own cuts that can potentially further improve the LP relaxation. In Table 2, we report the root gaps obtained when our early stopping conditions are in effect, and with Cplex cuts turned off and on. For these instances, Cplex cuts yield a very slight improvement in the root gap. Also, the loss incurred due to stopping early is not significant.

Table 2: Effect of Cplex cuts on average % root gap for CAP instances with 250 scenarios. At each CAP # row, the averages of four instances are reported.

	Without Cplex Cuts			With Cplex Cuts		
CAP #	BEN	MP	SP	BEN	MP	SP
101-104	22.40	21.58	0.07	16.91	16.69	0.07
111-114	8.72	8.28	0.41	7.84	7.79	0.41
121-124	18.97	20.04	1.02	17.99	19.57	1.02
131-134	25.24	25.30	0.33	20.15	21.68	0.33
Mean	18.83	18.80	0.46	15.72	16.43	0.46

In order to better understand the effect of the GMI cuts as used in either the extended space (SP) or projected space (MP), we first test our branch-and-cut algorithms with Cplex cuts turned off. We leave Cplex cuts turned on for the extensive formulation. Table 3 presents a summary of the branch-and-cut results for CAP instances with 250, 500 and 1500 scenarios obtained with EXT, BEN, SP and MP.

Table 3: Comparison of algorithms for CAP instances when Cplex cuts are off. Times are given in seconds. At each (K,CAP #) row, the averages of four instances are reported.

		Avg Time (# unsolved)				Avg Gap (%)			
K	CAP #	EXT	BEN	MP	SP	EXT	BEN	MP	SP
250	101-104	258	(4)	(4)	64	0.00	14.93	15.10	0.00
	111-114	2359	(4)	(4)	586	0.00	8.02	8.12	0.00
	121-124	4718(2)	(4)	(4)	1690	0.43	16.11	16.47	0.00
	131-134	4151(1)	(4)	(4)	477	0.19	22.22	23.23	0.00
	Mean	1858	-	-	417	0.16	15.32	15.73	0.00
500	101-104	1171	(4)	(4)	175	0.00	16.75	16.37	0.00
	111-114	10787(3)	(4)	(4)	2967(1)	2.00	8.51	8.00	0.05
	121-124	10935(3)	(4)	(4)	6442(2)	3.12	16.39	18.00	0.13
	131-134	9512(3)	(4)	(4)	1239	1.44	22.92	24.01	0.00
	Mean	6020	-	-	1426	1.64	16.14	16.60	0.04
1500	101-104	8583(1)	(4)	(4)	815	3.76	19.10	18.30	0.00
	111-114	(4)	(4)	(4)	7813(2)	15.74	9.14	8.36	0.45
	121-124	(4)	(4)	(4)	10182(3)	22.23	19.08	19.88	0.69
	131-134	(4)	(4)	(4)	7944	44.15	25.28	26.72	0.00
	Mean	12653	-	-	4764	21.47	18.15	18.32	0.28

We report geometric means of solution times, where the times are 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. If all instances in a set are not solved within the time limit, the reported mean time is a lower bound on the time to solve 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 bounds reported by the solver at the end of the run. Instances that were solved to optimality are included as a zero in calculating this average.

EXT is able to solve to optimality many of the instances with 250 scenarios. It fails to solve a number of 500 scenario instances but terminates with a small optimality gap. However, for the

1500 scenario instances, the optimality gaps are quite large. On the other hand, BEN and MP fail to solve any of the instances, and yield large optimality gaps after the time limit. This is not surprising given the results in Table 2, which indicate that BEN and MP yield weak LP relaxations. Finally, we observe that the SP algorithm is able to solve most of these instances in the time limit, and does so significantly faster than the extensive formulation. When it does not solve within the time limit, the optimality gaps are much smaller than the other methods, especially for the larger instances. These results indicate that on this problem class, adding split cuts in the extended space rather than in the projected space is essential. We obtain very similar results when we turn on Cplex cuts, and so provide that summary table in the Online Supplement.

4.5 Stochastic network interdiction problem

The second test problem is the stochastic network interdiction problem (SNIP) described in [20]. In this problem, the interdictor installs sensors to some of the arcs of a given network in order to maximize the expected probability of catching an intruder. Let N and A denote the set of nodes and the set of arcs of the network and let $D \subseteq A$ be the subset of arcs on which sensors are allowed to be placed. In the first stage the interdictor installs the sensors knowing the probability of the intruder avoiding detection with and without a sensor on each arc (denoted by r_{ij} and q_{ij}). A scenario k corresponds to the origin and destination of the intruder denoted by s^k and t^k and the realization probability of the scenario is denoted by p_k. Cost of installing a sensor on arc $(i,j) \in D$ is denoted by c_{ij} and b is total budget of the interdictor. In the second stage the intruder chooses a maximum-reliability path from s^k to t^k that maximizes the probability of avoiding detection. Lastly, ψ_j^k is the value of a maximum-reliability path from j to t^k when no sensors are placed. ψ_j^k can be calculated by solving a shortest path problem. First stage binary variables x_{ij} denote if a sensor is installed on arc (i,j). Second stage variables π_i^k give the probability that the evader can travel from i to t^k undetected. The formulation is as follows:

$$
\begin{aligned}
\min \ & \sum_{k\in\mathcal{K}} p_k \pi^k_{s^k} \\
\text{s.t.} \ & \sum_{(i,j)\in D} c_{ij} x_{ij} \le b, \\
& \pi^k_{t^k} = 1, && k \in \mathcal{K}, \\
& \pi^k_i - q_{ij}\pi^k_j \ge 0\ , && (i,j) \in D, k \in \mathcal{K}, \\
& \pi^k_i - r_{ij}\pi^k_j \ge 0\ , && (i,j) \in A \setminus D, k \in \mathcal{K}, \\
& \pi^k_i - r_{ij}\pi^k_j \ge -(r_{ij}-q_{ij})\psi^k_j x_{ij}, && (i,j) \in D, k \in \mathcal{K}, \\
& \pi^k_i \ge 0, \quad i \in N,\ k \in \mathcal{K},\ x \in \{0,1\}^{|D|}.
\end{aligned}
$$

The first constraint above ensures that the total cost of installing sensors does not exceed the budget b. The remaining constraints ensure that π_i^k is at least the probability that the evader can travel from i to j undetected times the probability that the evader can travel from j to t^k undetected.

All instances have 456 scenarios, contain 320 binary first-stage decision variables, and use the same network consisted of 783 nodes and 2586 arcs. We focus 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 [20], where time is measured in minutes and MIPs are solved to 1% optimality.

Computational experiments with the root bound. Table 4 reports the average percentage root gap obtained with BEN, MP and SP over sets of SNIP instances having similar properties (the

early stopping condition is turned off). We observe that both MP and SP yield significantly smaller gaps than BEN, but SP does not close as much gap as on the CAP instances.

Table 4: Average % root gap obtained by different methods for SNIP instances. At each Budget row, the averages of five instances are reported.

	snipno = 3			snipno = 4		
Budget	BEN	MP	SP	BEN	MP	SP
30	22.5	13.5	7.9	25.1	13.7	8.3
40	26.2	18.0	11.3	28.4	17.0	11.4
50	27.5	18.5	12.0	30.5	19.7	13.2
60	28.2	19.1	12.5	32.1	22.3	14.7
70	28.9	20.4	12.6	32.6	23.8	13.9
80	30.9	21.9	14.5	33.3	24.6	14.3
90	33.1	23.6	17.6	36.2	29.2	18.4
Mean	28.2	19.3	12.6	31.2	21.5	13.5

We do not report PR-CGLP and SP-CGLP gaps for SNIP instances as these methods take too much time. However, preliminary results (from lower bounds at the end of one day of computing time for some instances) indicate that PR-CGLP yields gaps similar to or smaller than SP; e.g., for snipno=3, PR-CGLP gaps range between 10.0 and 15.8%. Thus, project-and-cut seems as effective as cut-and-project for these instances.

Computational experiments with the branch-and-bound tree. We next compare the different branch-and-cut methods defined earlier. In Table 5, we report the root gaps obtained when Cplex cuts are turned off and on (and with the early stopping conditions turned on). In the absence

Table 5: Effect of Cplex cuts on average % root gap obtained by different methods for SNIP instances. At each (snipno,Budget) row, the averages of five instances are reported.

		Without Cplex Cuts			With Cplex Cuts		
snipno	Budget	BEN	MP	SP	BEN	MP	SP
3	30	23.2	13.3	12.7	5.18	4.64	3.25
	40	27.2	17.0	15.7	7.37	6.47	5.50
	50	27.8	18.3	16.5	6.58	6.15	5.27
	60	28.4	18.5	16.8	6.12	5.46	4.98
	70	29.0	19.2	17.5	5.42	4.82	4.31
	80	31.2	20.7	18.3	6.93	6.07	5.18
	90	33.1	23.8	20.9	8.15	7.85	7.26
	Mean	28.6	18.7	16.9	6.53	5.92	5.11
4	30	25.8	14.2	13.5	4.97	4.75	3.08
	40	29.2	17.7	16.1	7.49	6.39	4.42
	50	31.4	19.3	18.0	8.13	6.71	5.53
	60	32.5	22.0	19.4	8.11	7.84	6.14
	70	32.9	22.5	19.0	6.38	6.29	4.70
	80	33.4	23.7	18.6	4.54	6.21	3.02
	90	36.8	26.7	22.4	7.24	9.19	5.67
	Mean	31.7	20.9	18.1	6.69	6.77	4.65

of Cplex cuts, MP and SP gaps are close to each other and both are much better than BEN gaps, though the difference between MP and SP root gaps is smaller than in the Table 4. As also observed in Table 4, applying GMI cuts in both extended and projected spaces is beneficial for SNIP instances. More interestingly, when Cplex cuts are also applied, all algorithms obtained similar root gaps, which are all much smaller than the ones obtained when Cplex cuts are off. Nevertheless, SP gives better bounds than MP with and without Cplex cuts. These results suggest that, the ability to use information from multiple scenarios to derive cuts in the Benders reformulation (as Cplex cuts do) may be important for achieving improved bounds in this problem.

Next, in Table 6 we present a summary of the branch-and-cut results for SNIP instances. In order to understand the effect of GMI cuts, we first turn Cplex cuts off in all methods except the extensive formulation. We provide the number of unsolved instances (if there are any) for each category in parentheses in the time column.

Table 6: Comparison of algorithms for SNIP instances without Cplex cuts. Times are in seconds. At each (snipno="no",Budget="b") row, the averages of five instances are reported.

		Avg Time (# unsolved)				Avg Gap (%)			
no	b	EXT	BEN	MP	SP	EXT	BEN	MP	SP
3	30	(5)	639	487	442	18.0	0.00	0.00	0.00
	40	(5)	7915(3)	6265(1)	2253	25.0	0.67	0.17	0.00
	50	(5)	8626(3)	5339(1)	2328	26.1	1.27	0.47	0.00
	60	(5)	10599(4)	6885(2)	2425(1)	23.7	2.38	1.03	0.14
	70	(5)	- (5)	- (5)	4435(1)	27.1	3.90	1.66	0.22
	80	(5)	- (5)	11709(3)	10096(4)	28.9	3.94	2.44	0.96
	90	(5)	- (5)	- (5)	13283(4)	30.7	6.14	4.16	1.76
	Mean	-	7536	5977	3188	25.7	2.61	1.42	0.44
4	30	(5)	601	485	485	22.4	0.00	0.00	0.00
	40	(5)	1802	2312	774	25.7	0.00	0.00	0.00
	50	(5)	3442(1)	3579	1252	29.0	0.39	0.00	0.00
	60	(5)	4917(1)	6494(2)	1441	30.3	0.54	0.61	0.00
	70	(5)	6815(2)	12621(3)	1097	39.0	1.21	1.59	0.00
	80	(5)	4503	10494(2)	796	49.6	0.00	1.43	0.00
	90	(5)	5948(2)	13376(4)	1630	58.7	1.08	5.70	0.00
	Mean	-	3188	4639	995	36.4	0.46	1.33	0.00

Compared to CAP results, for the SNIP instances, the role of EXT and BEN is reversed: EXT is unable to solve any of the instances, and yields a very large average optimality gap after the time limit, whereas BEN is able to solve some of the instances. This is caused by the relatively larger size of these instances, which prevents EXT from even being able to finish processing the root node within the time limit, whereas BEN is able to process a large number of nodes. MP and SP perform significantly better than BEN. They solve more instances and end up with smaller optimality gaps for the unsolved instances. SP is able to solve more instances, including all of snipno=4 instances, and converges significantly faster than MP does.

Moreover, in Table 7 we compare the four methods when Cplex cuts are turned on. BEN and SP solve all of the instances while MP fails to solve a few instances but ends up with small optimality gaps. In terms of solution time, for snipno=4 instances, which are easier than snipno=3 instances, BEN performs better than SP. However, SP converges slightly faster in the more difficult large capacity instances of snipno=3. The main reason that BEN performs well is that Cplex

Table 7: Comparison of algorithms for SNIP instances with Cplex cuts. Times are in seconds. At each (snipno="no",Budget="b") row, the averages of five instances are reported.

		Avg Time (# unsolved)				Avg Gap (%)			
no	b	EXT	BEN	MP	SP	EXT	BEN	MP	SP
3	30	(5)	183	280	415	18.0	0	0	0
	40	(5)	784	1739	830	25.0	0	0	0
	50	(5)	512	1134	867	26.1	0	0	0
	60	(5)	906	1666	1121	23.7	0	0	0
	70	(5)	1402	1891	1389	27.1	0	0	0
	80	(5)	1938	4729(2)	1579	28.9	0	0.45	0
	90	(5)	4794	9214(3)	4050	30.7	0	0.60	0
	Mean	-	980	1855	1169	25.7	0	0.15	0
4	30	(5)	183	304	443	22.4	0	0	0
	40	(5)	266	668	661	25.7	0	0	0
	50	(5)	429	1161	720	29.0	0	0	0
	60	(5)	468	1697	799	30.3	0	0	0
	70	(5)	499	2198	934	39.0	0	0	0
	80	(5)	214	1815	485	49.6	0	0	0
	90	(5)	327	8859(3)	533	58.7	0	1.55	0
	Mean	-	320	1460	633	36.4	0	0.22	0

cuts dramatically improve the root bounds. We note that to obtain this improvement while using Cplex, it is essential to solve the master LP outside of the branch-and-cut algorithm, and include the Benders cuts as part of the formulation passed to the branch-and-cut algorithm. If we add root Benders cuts via cut callback routines, all algorithms worsen, and BEN and MP are much worse; they fail to solve the majority of the instances and on average their solution times are larger than the solution times of SP by a factor of 3.6 and 4.8 for snipno=3 and snipno=4 instances, respectively.

To summarize, for SNIP instances deriving split cuts in the extended space and the projected space are both useful. In the absence of Cplex cuts, SP is the best method. However, as Cplex cuts reduce root gaps of all our branch-and-cut algorithms to almost the same level, the results with Cplex cuts are mixed. SP is a computationally intensive method, and the extra time spent solving the progressively harder subproblems and generating cuts is not compensated by the gap improvement for relatively small or easy instances.

Finally, we remark that our results with general purpose cuts are comparable to those obtained in [20] using problem-specific valid inequalities (even though they solve MIPs to 1% optimality in [20], unlike our 0.1% tolerance).

5 Concluding Remarks

We analyzed two approaches for using integrality constraints to improve the LP relaxation within a Benders decomposition algorithm. In project-and-cut, Benders cuts are used to approximate the Benders reformulation, and then integrality constraints are used to derive cuts directly for that formulation. In cut-and-project, integrality constraints are used to derive cuts in the extended formulation, and this improved relaxation is projected via Benders cuts. We found that when using split cuts, these approaches are equivalent for a single split, but cut-and-project can yield stronger

relaxations when using multiple splits. In addition, our computational results on a stochastic facility location problem indicated that this difference can be very significant. On the other hand, we observed that for a stochastic integer program, cuts derived directly from the Benders reformulation can be stronger due to the ability to use information from multiple scenarios. Our computational results on a second test problem indicated that this effect can also be important.

Our result on the strength of split cuts in a projected vs. extended space are applicable to MIP in general. An interesting idea raised by this result is to consider using or building extended formulations of a MIP problem for the purpose of obtaining a stronger relaxation using a fixed class of valid inequalities or a fixed procedure for separating valid inequalities. In stochastic integer programming, an interesting future direction is to investigate the use of problem-specific cuts within the cut-and-project framework.

Acknowledgments. The authors are grateful to David Morton and Feng Pan for providing the instances used in [20]. This research has been supported by the National Science Foundation under grant CMMI-1130266 and by the Office of Naval Research under award N00014-15-1-2268.

References

[1] E. Balas, S. Ceria, G. Cornuejols, and N. Natraj. Gomory cuts revisited. *Oper. Res. Lett*, 19:1–9, 1996.

[2] E. Balas and A. Saxena. Optimizing over the split closure. *Math. Program.*, 113:219–240, 2008.

[3] J.E. Beasley. OR-Library: Distributing test problems by electronic mail. *J. Oper. Res. Soc.*, 41:1069–1072, 1990. `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/capinfo.html`.

[4] E. Beier, S. Venkatachalam, L. Corolli, and L. Ntaimo. Stage-and scenario-wise Fenchel decomposition for stochastic mixed 0-1 programs with special structure. *Comput. and Oper. Res.*, 59:94–103, 2015.

[5] D. Bienstock and O. Günlük. Capacitated network design – polyhedral structure and computation. *INFORMS J. Comput.*, 8:243–259, 1994.

[6] P. Bonami. On optimizing over lift-and-project closures. *Math. Program. Comput.*, 4:151–179, 2012.

[7] A. Caprara and A. Letchford. On the separation of split cuts and related inequalities. *Math. Program.*, 94:279–294, 2003.

[8] C. C. Carøe. *Decomposition in Stochastic Integer Programming.* PhD thesis, Department of Operations Research, University of Copenhagen, Denmark, 1998.

[9] C. C. Carøe and R. Schultz. Dual decomposition in stochastic integer programming. *Oper. Res. Lett*, pages 37–45, 1999.

[10] C. C. Carøe and J. Tind. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Math. Program.*, pages 451–464, 1998.

[11] G. Cornuéjols and Y. Li. On the rank of mixed 0,1 polyhedra. *Math. Program.*, 91:391–397, 2002.

[12] S. Dash and M. Goycoolea. A heuristic to separate rank-1 GMI cuts. *Math. Program. Comput.*, 2:231–257, 2010.

[13] S. Dash, O. Günlük, and A. Lodi. MIR closures of polyhedral sets. *Math. Program.*, 121:33–60, 2010.

[14] M. Fischetti and D. Salvagnin. A relax-and-cut framework for gomory's mixed-integer cuts. *Math. Program. Comput.*, 3:79–102, 2011.

[15] D. Gade, S. Küçükyavuz, and S. Sen. Decomposition algorithms with parametric Gomory cuts for two-stage stochastic integer programs. *Math. Program.*, 144(1-2):39–64, 2014.

[16] F. V. Louveaux. Discrete stochastic location models. *Ann. Oper. Res.*, 6:23–34, 1986.

[17] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM J. Opt.*, 1:166–190, 1991.

[18] S. Modaresi, M.R. Kılınç, and J.P. Vielma. Split cuts and extended formulations for mixed integer conic quadratic programming. *Oper. Res. Lett*, 43(1):10–15, 2015.

[19] L. Ntaimo. Fenchel decomposition for stochastic mixed-integer programming. *J. Global. Opt.*, 55:141–163, 2013.

[20] Feng Pan and David P Morton. Minimizing a stochastic maximum-reliability path. *Networks*, 52(3):111–119, 2008.

[21] S. Sen and H.D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Math. Program.*, 106:203–223, 2006.

[22] S. Sen and J.L. Higle. The C^3 theorem and a D^2 algorithm for large scale stochastic mixed-integer programming: set convexification. *Math. Program.*, 104:1–20, 2005.

[23] M.W. Tanner and L. Ntaimo. Computations with disjunctive cuts for two-stage stochastic mixed 0-1 integer programs. *J. Global. Opt.*, 58:365–384, 2008.

[24] R. Van Slyke and R. J.-B. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM J. Appl. Math*, 17:638–663, 1969.

[25] M. Zhang and S. Küçükyavuz. Finitely convergent decomposition algorithms for two-stage stochastic pure integer programs. *SIAM J. Opt.*, 24:1933–1951, 2014.