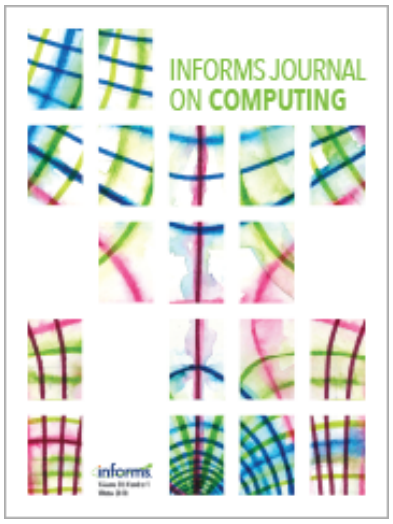


INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

On Generating Lagrangian Cuts for Two-Stage Stochastic Integer Programs

Rui Chen, James Luedtke

To cite this article:
Rui Chen, James Luedtke (2022) On Generating Lagrangian Cuts for Two-Stage Stochastic Integer Programs. INFORMS Journal on Computing

Published online in Articles in Advance 06 Apr 2022

. https://doi.org/10.1287/ijoc.2022.1185

http://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Articles in Advance, pp. 1–18
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

On Generating Lagrangian Cuts for Two-Stage Stochastic Integer Programs

Rui Chen,[a,*] **James Luedtke**[a]

[a] Department of Industrial and Systems Engineering, University of Wisconsin-Madison, Madison, Wisconsin 53706
*Corresponding author

Contact: rchen234@wisc.edu, https://orcid.org/0000-0002-8848-6118 (RC); jim.luedtke@wisc.edu, https://orcid.org/0000-0001-9265-7728 (JL)

Received: June 15, 2021
Revised: January 20, 2022; February 21, 2022
Accepted: February 24, 2022
Published Online in Articles in Advance: April 6, 2022

https://doi.org/10.1287/ijoc.2022.1185

Abstract. We investigate new methods for generating Lagrangian cuts to solve two-stage stochastic integer programs. Lagrangian cuts can be added to a Benders reformulation and are derived from solving single scenario integer programming subproblems identical to those used in the nonanticipative Lagrangian dual of a stochastic integer program. Although Lagrangian cuts have the potential to significantly strengthen the Benders relaxation, generating Lagrangian cuts can be computationally demanding. We investigate new techniques for generating Lagrangian cuts with the goal of obtaining methods that provide significant improvements to the Benders relaxation quickly. Computational results demonstrate that our proposed method improves the Benders relaxation significantly faster than previous methods for generating Lagrangian cuts and, when used within a branch-and-cut algorithm, significantly reduces the size of the search tree for three classes of test problems.

History: Accepted by Andrea Lodi, Area Editor for Design & Analysis of Algorithms—Discrete.
Funding: This work was supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research [Grant DE-AC02-06CH11357].
Supplemental Material: The online appendix is available at https://doi.org/10.1287/ijoc.2022.1185.

Keywords: two-stage stochastic integer programs • Lagrangian cuts • dual decomposition

1. Introduction

We study methods for solving two-stage stochastic integer programs (SIPs) with general mixed-integer first-stage and second-stage variables. Two-stage stochastic programs are used to model problems with uncertain data, where a decision maker first decides the values of first-stage decision variables, then observes the values of the uncertain data, and finally decides the values of second-stage decision variables. The objective is to minimize the sum of the first-stage cost and the expected value of the second-stage cost, where the expected value is taken with respect to the distribution of the uncertain data. Each realization of the uncertain data is called a scenario. Assuming the uncertainty is modeled with a finite set of scenarios S, a two-stage SIP can be formulated as follows:

$$z_{\text{IP}} = \min_x \left\{ c^\top x + \sum_{s\in S} p_s Q_s(x) : Ax \geq b, x \in X \right\}, \tag{1}$$

where $c \in \mathbb{R}^n$, and for each $s \in S$, p_s denotes the probability of scenario s and $Q_s : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is the recourse function of scenario s defined as

$$Q_s(x) = \min_y \{(q^s)^\top y : W^s y \geq h^s - T^s x, y \in Y\}. \tag{2}$$

Here $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^{n_y}$ denote the integrality restrictions on some or all variables of x and y, respectively. Sign restrictions and variable bounds, if present, are assumed to be included in the constraints $Ax \geq b$ or $W^s y \geq h^s - T^s x$. Matrices W^s, T^s and vectors q^s, h^s are the realization of the uncertain data associated with scenario $s \in S$. We do not assume relatively complete recourse; that is, it is possible that the recourse problem (2) is infeasible for some $s \in S$ and $x \in X$ satisfying $Ax \geq b$, in which case $Q_s(x) = +\infty$ by convention.

Two-stage SIPs can also be written in the extensive form:

$$\begin{aligned} \min_{x, y^s} \quad & c^\top x + \sum_{s\in S} p_s (q^s)^\top y^s \\ \text{s.t.} \quad & Ax \geq b, x \in X, \\ & W^s y^s \geq h^s - T^s x^s, y^s \in Y, s \in S. \end{aligned} \tag{3}$$

From this perspective, SIPs are essentially large-scale mixed-integer programs (MIPs) with special block structure. Directly solving (3) as a MIP can be difficult when $|S|$ is large. Therefore, significant research has been devoted to the study of decomposition methods for solving SIP problems. Benders decomposition (Benders 1962, Van Slyke and Wets 1969) and dual decomposition (Carøe and Schultz 1999) are the two most commonly used decomposition methods for solving SIPs. In Benders decomposition, the second-

stage variables are projected out from the Benders master model and linear programming (LP) relaxations of scenario subproblems are solved to add cuts in the master model. In dual decomposition, a copy of the first-stage variables is created for each scenario, and constraints are added to require the copies to be equal to each other. A Lagrangian relaxation is then formed by relaxing these so-called nonanticipativity constraints. Solving the corresponding Lagrangian dual problem requires solving single-scenario MIPs to provide function values and supergradients of the Lagrangian relaxation. A more detailed description of dual decomposition for SIPs is given in Section 2.2. Usually dual decomposition generates a much stronger bound than Benders decomposition, but the bound takes much longer to compute. Rahmaniani et al. (2020) proposed Benders dual decomposition (BDD) in which Lagrangian cuts generated by solving single-scenario MIPs identical to the subproblems in dual decomposition are added to the Benders formulation to strengthen the relaxation. Similarly, Li and Grossmann (2018) develop a Benders-like decomposition algorithm implementing both Benders cuts and Lagrangian cuts for solving convex mixed 0-1 nonlinear stochastic programs. Although representing an interesting hybrid between the Benders and dual decomposition approaches, the time spent generating the Lagrangian cuts remains a limitation in these approaches. In this paper, we extend this line of work by investigating methods to generate Lagrangian cuts more efficiently.

This work more broadly contributes to the significant research investigating the use of cutting planes to strengthen the Benders model. Laporte and Louveaux (1993) propose integer L-shaped cuts for SIPs with pure binary first-stage variables. Sen and Higle (2005) and Sen and Sherali (2006) apply disjunctive programming techniques to convexify the second-stage problems for SIPs with pure binary first-stage variables. Ntaimo (2013) investigates the use of Fenchel cuts added to strengthen the subproblem relaxations. Gade et al. (2014) and Zhang and Küçükyavuz (2014) demonstrate how Gomory cuts can be used within a Benders decomposition method for solving SIPs with pure integer first-stage and second-stage variables, leading to a finitely convergent algorithm. Qi and Sen (2017) use cuts valid for multiterm disjunctions introduced in Chen et al. (2011) to derive an algorithm for SIPs with mixed-integer recourse. van der Laan and Romeijnders (2020) propose a new class of cuts, scaled cuts, which integrates previously generated Lagrangian cuts into a cut-generation model, again leading to a finitely convergent cutting-plane method for SIPs with mixed-integer recourse. Bodur et al. (2017) compare the strength of split cuts derived in the Benders master problem space to those added in the scenario LP relaxation space. Zou et al. (2019) propose strengthened Benders cuts and Lagrangian cuts to solve pure binary multistage SIPs.

There are also other approaches for solving SIPs that do not rely on a Benders model. Lulli and Sen (2004) develop a column generation-based algorithm for solving multistage SIPs. Lubin et al. (2013) apply proximal bundle methods to parallelize the dual decomposition algorithm. Boland et al. (2018) and Guo et al. (2015) investigate the use of progressive hedging to calculate the Lagrangian dual bound. Kim and Dandurand (2022) propose a new branching method for dual decomposition of SIPs. Kim et al. (2019) apply asynchronous trust-region methods to solve the Lagrangian dual used in dual decomposition.

The goal of this paper is to develop an effective way of generating Lagrangian cuts to add to a Benders model. The main contributions of our work are summarized as follows.

1. We propose a normalization for generating Lagrangian cuts similar to the one used by Fischetti et al. (2010) for separating Benders cuts. This normalization can be used to construct a Lagrangian cut separation problem different from the one used by Zou et al. (2019) and Rahmaniani et al. (2020).

2. We propose methods for accelerating the generation of Lagrangian cuts, including solving the cut generation problem in a restricted subspace, and using a MIP approximation to identify a promising restricted subspace. Numerical results indicate that these approaches lead to significantly faster bound improvement from Lagrangian cuts.

3. We conduct an extensive numerical study on three classes of two-stage SIPs. We compare the impact of different strategies for generating Lagrangian cuts. Computational results are also given for using our method within branch-and-cut as an exact solution method. We find that this method has potential to outperform both a pure Benders approach and a pure dual decomposition approach.

Finally, although we focus on new techniques for generating Lagrangian cuts for use in solving two-stage SIPs, these techniques can be directly applied to multistage SIPs by integrating them within the stochastic dual dynamic integer programming (SDDIP) algorithm by Zou et al. (2019).

2. Preliminaries

2.1. Branch-and-Cut–Based Methods

Problem (1) can be reformulated as the following Benders model:

$$\min_{x,\,\theta_s}\left\{c^\top x+\sum_{s\in S}p_s\theta_s : (x,\theta_s)\in E^s, s\in S\right\}, \tag{4}$$

where for each $s\in S$, E^s contains the first-stage constraints and the epigraph of Q_s, that is,

$$E^s = \{(x, \theta_s) \in X \times \mathbb{R} : Ax \geq b, \theta_s \geq Q_s(x)\}. \quad (5)$$

In a typical approach to solve (4), each recourse function Q_s is replaced by a cutting-plane underestimate $\hat{Q}_s$ that creates a relaxation and is dynamically updated. In each iteration, an approximate problem defined by the current underestimate is solved to obtain a candidate solution and then a cut generation problem is solved to update the cutting plane approximation if necessary. This process is repeated until no cuts are identified.

We review two types of valid inequalities for E^s. The first collection of cuts are Benders cuts (Benders 1962, Van Slyke and Wets 1969). Benders cuts are generated based on the LP relaxation of Problem (2) defining $Q_s(x)$. Given a candidate solution $\hat{x}$, the LP relaxation of the recourse problem is solved:

$$\min_y \{(q^s)^\top y : W^s y \geq h^s - T^s \hat{x}\}. \quad (6)$$

Let μ be an optimal dual solution of Problem (6). Based on LP duality, the following *Benders cut* is valid for E^s:

$$\mu^\top T^s x + \theta_s \geq \mu^\top h^s. \quad (7)$$

If the SIP has continuous recourse, that is, $Y = \mathbb{R}^{n_y}$, then (7) is tight at $\hat{x}$, that is, $Q_s(\hat{x}) = -\mu^\top T^s \hat{x} + \mu^\top h^s$. The cutting-plane model $\hat{Q}_s$ is often constructed by iteratively adding Benders cuts until the lower bound converges to the LP relaxation bound. Benders cuts are sufficient to provide convergence for solving SIPs with continuous recourse (Kall et al. 1994).

Another useful family of cuts is the integer L-shaped cuts introduced by Laporte and Louveaux (1993). These cuts are valid only when the first-stage variables are binary, that is, $X = \{0,1\}^n$, but can be applied even when the second-stage includes integer decision variables. In this case, given $\hat{x} \in \{0,1\}^n$ and a value L_s such that $Q_s(x) \geq L_s$ for all feasible x, the following *integer L-shaped cut* is a valid inequality for E^s:

$$\theta_s \geq Q_s(\hat{x}) - (Q_s(\hat{x}) - L_s)\left(\sum_{i:\hat{x}_i=1}(1 - x_i) + \sum_{i:\hat{x}_i=0} x_i\right). \quad (8)$$

A standard branch-and-cut algorithm using Benders cuts and integer L-shaped cuts is described in Section S.1 of the online appendix. The branch-and-cut algorithm can be implemented using a lazy constraint callback in modern MIP solvers, which allows the addition of Benders or integer L-shaped cuts when the solver encounters a solution $(\hat{\theta}, \hat{x})$ with $\hat{x} \in X$ (i.e., $\hat{x}$ satisfies any integrality constraints) but for which $\hat{\theta}_s < Q_s(\hat{x})$ for some $s \in S$. These two classes of cuts are sufficient to guarantee convergence for SIPs with continuous recourse or pure binary first-stage variables. However, the efficiency of the algorithm depends significantly on the strength of the cutting-plane models $\hat{Q}_s$ s. Given poor relaxations of the recourse functions, the branch-and-bound search may end up exploring a huge number of nodes, resulting in a long solution time. As discussed in Section 1, many methods have been proposed to strengthen the model $\hat{Q}_s$ to accelerate the algorithm.

The validity of Lagrangian cuts does not require either continuous recourse or pure binary first-stage variables. However, outside of those settings, additional cuts or a specialized branching scheme would be required to obtain a convergent algorithm. We refer the reader to, Ahmed et al. (2004), Qi and Sen (2017), van der Laan and Romeijnders (2020), and Zhang and Küçükyavuz (2014) for examples of methods that can be used to obtain a finitely convergent algorithm in other settings. Lagrangian cuts could potentially be added to enhance any of these approaches.

2.2. Dual Decomposition

In dual decomposition (Carøe and Schultz 1999), a copy x^s of the first-stage variables x is created for each scenario $s \in S$, which yields a reformulation of (1):

$$\begin{aligned} \min_{x, x^s, y^s} \ & \sum_{s\in S} p_s (c^\top x^s + (q^s)^\top y^s) \\ \text{s.t.} \ & Ax^s \geq b, && s \in S, \\ & T^s x^s + W^s y^s \geq h^s, && s \in S, \\ & x^s \in X, y^s \in Y, && s \in S, \\ & x^s = x, && s \in S. \end{aligned}$$

Lagrangian relaxation is applied to the *nonanticipativity constraints* $x^s = x$ in this formulation with multipliers λ^s for $s \in S$, which gives the following Lagrangian relaxation problem:

$$\begin{aligned} z(\lambda) = \min_{x, x^s, y^s} \ & \sum_{s\in S} p_s (c^\top x^s + (q^s)^\top y^s) + \sum_{s\in S} p_s (\lambda^s)^\top (x^s - x), \\ \text{s.t.} \ & (x^s, y^s) \in K^s, \quad s \in S, \end{aligned} \quad (9)$$

where $K^s := \{x \in X, y \in Y : Ax \geq b, T^s x + W^s y \geq h^s\}$ for each $s \in S$. Throughout this paper, we assume that K^s is nonempty for each $s \in S$.

The nonanticipative Lagrangian dual problem is defined as $z_D = \max_\lambda z(\lambda)$. Constraint $\sum_{s\in S} p_s \lambda^s = 0$ is implicitly enforced in $\max_\lambda z(\lambda)$ because $z(\lambda) = -\infty$ when $\sum_{s\in S} p_s \lambda^s \neq 0$. Under this condition, the variables x in (9) can be eliminated, and hence (9) can be solved by solving a separate optimization for each scenario $s \in S$. The nonanticipative Lagrangian dual problem

$$z_D = \max_\lambda \left\{ z(\lambda) : \sum_{s\in S} p_s \lambda^s = 0 \right\}$$

is a convex program that gives a lower bound on the optimal value of (1). The following theorem provides a primal characterization of the bound z_D.

Theorem 1 (Carøe and Schultz 1999). *The following equality holds:*

$$z_D = \min_{x,y^s}\left\{c^\top x + \sum_{s\in S} p_s (q^s)^\top y^s : (x,y^s) \in \operatorname{conv}(K^s), s \in S\right\}.$$

Empirical evidence (Schütz et al. 2009, Solak et al. 2010) shows that the Lagrangian dual bound z_D is often a tight lower bound on the optimal objective value of (2). Such a bound can be used in a branch and bound algorithm and to generate feasible solutions using heuristics (Carøe and Schultz 1999, Kim and Dandurand 2022). However, the Lagrangian dual problem can be hard to solve because of its large size (with $n|S|$ variables) and the difficulty of solving MIP subproblems to evaluate $z(\lambda)$ and obtain supergradients of $z(\cdot)$ at a point λ.

2.3. Benders Dual Decomposition

The Benders dual decomposition (BDD) algorithm (Rahmaniani et al. 2020) is a version of Benders decomposition that uses strengthened Benders cuts and Lagrangian cuts to tighten the Benders model. In particular, they solve two types of scenario MIPs to generate optimality and feasibility cuts for E^s:

1. Given any $\lambda \in \mathbb{R}^n$, let $(\bar{x}^s_\lambda, \bar{y}^s_\lambda)$ be an optimal solution of

$$\min_{x,y}\{\lambda^\top x + (q^s)^\top y : (x,y) \in K^s\}.$$

Then the following *Lagrangian optimality cut* is valid for E^s:

$$\lambda^\top (x - \bar{x}^s_\lambda) + \theta_s \geq (q^s)^\top \bar{y}^s_\lambda. \tag{10}$$

2. Given any $\lambda \in \mathbb{R}^n$, let $(\hat{x}^s_\lambda, \hat{y}^s_\lambda, \hat{u}^s_\lambda, \hat{v}^s_\lambda)$ be an optimal solution of

$$\min_{x,y,u,v}\{\mathbb{1}^\top v + \mathbb{1}^\top u + \lambda^\top x : Ax + u \geq b, T^s x + W^s y + v \geq h^s, x \in X, y \in Y\}.$$

Then the following *Lagrangian feasibility cut* is valid for E^s:

$$\lambda^\top (x - \hat{x}^s_\lambda) \geq \mathbb{1}^\top \hat{v}^s_\lambda + \mathbb{1}^\top \hat{u}^s_\lambda. \tag{11}$$

The BDD algorithm generates both types of Lagrangian cuts by heuristically solving a Lagrangian cut generation problem. Numerical results from Rahmaniani et al. (2020) show that Lagrangian cuts are able to close significant gap at the root node for a variety of SIP problems. Our goal in this work is to provide new methods for quickly finding strong Lagrangian cuts.

3. Selection and Normalization of Lagrangian Cuts

We begin by introducing an alternative view of the Lagrangian cuts (10) and (11), which leads to a different cut generation formulation. Let $Q^*_s : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}$ be defined as

$$\begin{aligned} Q^*_s(\pi,\pi_0) &= \min_x\{\pi^\top x + \pi_0 Q_s(x) : Ax \geq b, x \in X\} \\ &= \min_{x,y}\{\pi^\top x + \pi_0 (q^s)^\top y : (x,y) \in K^s\} \end{aligned} \tag{12}$$

for $(\pi,\pi_0) \in \mathbb{R}^n \times \mathbb{R}_+$. According to (12), for any $(\pi,\pi_0) \in \mathbb{R}^n \times \mathbb{R}_+$, the following inequality is valid for set E^s defined in (5):

$$\pi^\top x + \pi_0 \theta_s \geq Q^*_s(\pi,\pi_0). \tag{13}$$

We next demonstrate that the set of all cuts of the form (13) is equivalent to set of Lagrangian cuts (10) and (11), and therefore we refer to cuts of the form (13) as Lagrangian cuts. For any $\Pi_s \subseteq \mathbb{R}^n \times \mathbb{R}_+$, let $P_s(\Pi_s)$ denote the set defined by all cuts (13) with coefficients restricted on Π_s, that is,

$$\begin{aligned} P_s(\Pi_s) := \{(x,\theta_s) : \pi^\top x + \pi_0 \theta_s \geq Q^*_s(\pi,\pi_0) \\ \text{for all } (\pi,\pi_0) \in \Pi_s\}. \end{aligned}$$

Proposition 1. *Let $\Pi_s \subseteq \mathbb{R}^{n+1}$ be a neighborhood of the origin and let $\Pi^*_s = \Pi_s \cap (\mathbb{R}^n \times \mathbb{R}_+)$. Define $\bar{P}_s := \{(x,\theta_s) :$ (10) and (11) hold for all $\lambda \in \mathbb{R}^n\}$. Then $P_s(\Pi^*_s) = \bar{P}_s$.*

Proof. We first show that $P_s(\Pi^*_s) \subseteq \bar{P}_s$. Let $(x,\theta_s) \in P_s(\Pi^*_s)$. We show that (x,θ_s) satisfies (10) and (11) for all $\lambda \in \mathbb{R}^n$. Let $\lambda \in \mathbb{R}^n$, and choose ρ^1_λ and ρ^2_λ large enough such that $(\lambda/\rho^1_\lambda, 1/\rho^1_\lambda), (\lambda/\rho^2_\lambda, 0) \in \Pi^*_s$, and thus (x,θ_s) satisfies (13) using these values for (π,π_0). Multiplying Inequality (13) with $(\pi,\pi_0) = (\lambda/\rho^1_\lambda, 1/\rho^1_\lambda) \in \Pi^*_s$ by ρ^1_λ gives

$$\begin{aligned} \lambda^\top x + \theta_s &\geq \rho^1_\lambda Q^*_s(\lambda/\rho^1_\lambda, 1/\rho^1_\lambda) = Q^*_s(\lambda, 1) \\ &= \lambda^\top \bar{x}^s_\lambda + (q^s)^\top \bar{y}^s_\lambda, \end{aligned}$$

where $(\bar{x}^s_\lambda, \bar{y}^s_\lambda)$ is as defined in (10) and where we have used the observation that the function Q^*_s is positively homogeneous. Subtracting from both sides yields that (x,θ_s) satisfies (10). Conversely, Inequality (13) with $(\pi,\pi_0) = (\lambda/\rho^2_\lambda, 0) \in \Pi^*_s$ implies

$$\begin{aligned} &(\lambda/\rho^2_\lambda)^\top x \\ &\geq Q^*_s(\lambda/\rho^2_\lambda, 0) \\ &= \min_{x,y}\{\lambda^\top x : Ax \geq b, T^s x + W^s y \geq h, x \in X, y \in Y\}/\rho^2_\lambda \\ &\geq \min_{x,y,u,v}\{\mathbb{1}^\top v + \mathbb{1}^\top u + \lambda^\top x : Ax + u \geq b, \\ &\qquad\qquad T^s x + W^s y + v \geq h^s, \\ &\qquad\qquad x \in X, y \in Y\}/\rho^2_\lambda \\ &= (\lambda^\top \hat{x}^s_\lambda + \mathbb{1}^\top \hat{v}^s_\lambda + \mathbb{1}^\top \hat{u}^s_\lambda)/\rho^2_\lambda. \end{aligned}$$

Multiplying both sides by ρ^2_λ and subtracting $\lambda^\top \hat{x}^s_\lambda$ from both sides gives Inequality (11). Because λ can be arbitrary, it implies $P_s(\Pi^*_s) \subseteq \bar{P}_s$.

Next let $(x,\theta_s)\in\bar{P}_s$. We aim to show $(x,\theta_s)\in P_s(\Pi_s^*)$. Let $(\pi,\pi_0)\in\Pi_s^*$. If $\pi_0>0$, then (13) is satisfied by scaling (10) with $\lambda=\pi/\pi_0$. We can similarly show that $\pi^\top x+\pi_0'\theta_s\geq Q_s^*(\pi,\pi_0')$ for any $\pi_0'>0$. Now assume $\pi_0=0$. By theorem 7.1 of Rockafellar (1970), Q_s^* is lower-semicontinuous, and hence,

$$\pi^\top x=\pi^\top x+\liminf_{\pi_0'\to 0}\pi_0'\theta_s\geq\liminf_{\pi_0'\to 0}Q_s^*(\pi,\pi_0')\geq Q_s^*(\pi,0),$$

that is, (13) is satisfied. Therefore, we have $\bar{P}_s\subseteq P_s(\Pi_s^*)$. □

Because Proposition 1 holds for any neighborhood Π_s of the origin, it allows flexibility to choose a normalization on the Lagrangian cut coefficients when generating cuts. We discuss the choice of such a normalization in Section 4.

Let z_{LC} denote the bound we obtain after adding all Lagrangian cuts, that is,

$$z_{\mathrm{LC}}:=\min_{x,\theta_s}\left\{c^\top x+\sum_{s\in S}p_s\theta_s:(x,\theta_s)\in P_s(\mathbb{R}^n\times\mathbb{R}_+),s\in S\right\}.$$

Because Q_s^* is positively homogeneous and the set defined by an inequality is invariant to any positive scaling of the inequality, we also have

$$z_{\mathrm{LC}}=\min_{x,\theta_s}\left\{c^\top x+\sum_{s\in S}p_s\theta_s:(x,\theta_s)\in P_s(\Pi_s^*),s\in S\right\}$$

for any $\Pi_s^*=\Pi_s\cap(\mathbb{R}^n\times\mathbb{R}_+)$ such that Π_s is a neighborhood of the origin. Rahmaniani et al. (2020) showed that, for $|S|=1$, adding all Lagrangian cuts of the form (10) and feasibility cuts (11) yields a Benders master problem with optimal objective value equal to z_{IP}. This is not true in general when $|S|>1$. However, Li and Grossmann (2018) showed that $z_{\mathrm{LC}}\geq z_D$. We show in the following theorem that this is an equality, that is, $z_{\mathrm{LC}}=z_D$. Because $z_{\mathrm{IP}}=z_D$ according to Theorem 1 when $|S|=1$, this result generalizes the result by Rahmaniani et al. (2020).

Theorem 2. *The equality $z_{\mathrm{LC}}=z_D$ holds.*

Proof. Although $z_{\mathrm{LC}}\geq z_D$ has been shown in Li and Grossmann (2018), we show both directions for completeness.

By Theorem 1,

$$\begin{aligned}z_D&=\min_{x,y^s}\left\{c^\top x+\sum_{s\in S}p_s(q^s)^\top y^s:(x,y^s)\in\mathrm{conv}(K^s),s\in S\right\}\\&=\min_{x,\theta_s}\left\{c^\top x+\sum_{s\in S}p_s\theta_s:(x,\theta_s)\in U^s,s\in S\right\},\end{aligned}$$

where $U^s:=\mathrm{proj}_{(x,\theta_s)}\{(x,y^s,\theta_s):\theta_s\geq(q^s)^\top y^s,(x,y^s)\in\mathrm{conv}(K^s)\}$. Therefore, we only need to show that $U^s=P_s(\mathbb{R}^n\times\mathbb{R}_+)$ for each $s\in S$.

On the one hand, let $s\in S$ and $(\bar{x},\bar{\theta}_s)\in U^s$. Then there exists $\bar{y}^s$ such that $\bar{\theta}_s\geq(q^s)^\top\bar{y}^s$ and $(\bar{x},\bar{y}^s)\in\mathrm{conv}(K^s)$. Then by definition of Q_s^*, we have

$$\begin{aligned}\pi^\top\bar{x}+\pi_0\bar{\theta}_s&\geq\pi^\top\bar{x}+\pi_0(q^s)^\top\bar{y}^s\\&\geq\min_{x,y}\{\pi^\top x+\pi_0(q^s)^\top y:(x,y)\in\mathrm{conv}(K^s)\}\\&=Q_s^*(\pi,\pi_0)\end{aligned}$$

for all $(\pi,\pi_0)\in\mathbb{R}^n\times\mathbb{R}_+$, which implies that $(\bar{x},\bar{\theta}_s)\in P_s(\mathbb{R}^n\times\mathbb{R}_+)$.

On the other hand, let $s\in S$ and $(\bar{x},\bar{\theta}_s)\notin U^s$. Becauses U^s is a polyhedron, which is convex and closed, there exists a hyperplane strictly separating $(\bar{x},\bar{\theta}_s)$ from U^s. In other words, there exists $u\in\mathbb{R}^n$ and $v\in\mathbb{R}$ such that $u^\top\bar{x}+v\bar{\theta}_s<\min_{x,\theta_s}\{u^\top x+v\theta_s:(x,\theta_s)\in U^s\}$. We have $v\geq 0$ in this case because $\min_{x,\theta_s}\{u^\top x+v\theta_s:(x,\theta_s)\in U^s\}=-\infty$ if $v<0$. Then

$$\begin{aligned}u^\top\bar{x}+v\bar{\theta}_s&<\min_{x,\theta_s}\{u^\top x+v\theta_s:(x,\theta_s)\in U^s\}\\&=\min_{x,y}\{u^\top x+v(q^s)^\top y:(x,y)\in\mathrm{conv}(K^s)\}=Q_s^*(u,v),\end{aligned}$$

that is, $(\bar{x},\bar{\theta}_s)$ violates the Lagrangian cut $u^\top\bar{x}+v\bar{\theta}_s\geq Q_s^*(u,v)$. Therefore, $(\bar{x},\bar{\theta}_s)\notin P_s(\mathbb{R}^n\times\mathbb{R}_+)$. □

We next consider the problem of separating a point from $P_s(\Pi_s^*)$ using Lagrangian cuts. Given a candidate solution $(\hat{x},\hat{\theta}_s)$ and a convex and compact Π_s^*, the problem of separating $(\hat{x},\hat{\theta}_s)$ from $P_s(\Pi_s^*)$ can be formulated as the following bounded convex optimization problem:

$$\max_{\pi,\pi_0}\{Q_s^*(\pi,\pi_0)-\pi^\top\hat{x}-\pi_0^\top\hat{\theta}_s:(\pi,\pi_0)\in\Pi_s^*\}.\tag{14}$$

Problem (14) can be solved by bundle methods given an oracle for evaluating the function value and a supergradient of Q_s^*. The function value and a supergradient of Q_s^* can be evaluated at any $(\pi,\pi_0)\in\mathbb{R}^n\times\mathbb{R}_+$ by solving the single-scenario MIP (12). Let (x^*,y^*) be an optimal solution of (12). Then we have $Q_s^*(\pi,\pi_0)=\pi^\top x^*+\pi_0(q^s)^\top y^*$, and $(x^*,(q^s)^\top y^*)$ is a supergradient of Q_s^* at (π,π_0).

Although (14) is a convex optimization problem, the requirement to solve a potentially large number of MIP problems of the form (12) to evaluate function values and supergradients of Q_s^* to find a single Lagrangian cut can make the use of such cuts computationally prohibitive. Numerical results byin Zou et al. (2019) indicate that, although Lagrangian cuts can significantly improve the LP relaxation bound for many SIP problems, the use of exact separation of Lagrangian cuts does not improve overall solution time. In an attempt to reduce the time required to separate Lagrangian cuts, Rahmaniani et al. (2020) applied an inner approximation heuristic. In the next section, we propose an alternative approach for generating Lagrangian cuts more quickly.

4. Restricted Separation of Lagrangian Cuts

Observe that given any $(\pi, \pi_0) \in \mathbb{R}^n \times \mathbb{R}_+$, evaluating $Q_s^*(\pi, \pi_0)$ gives a valid inequality (13). By picking (π, π_0) properly, we show in the following proposition that a small number of Lagrangian cuts is sufficient to give the perfect information bound (Avriel and Williams 1970), defined as

$$z_{\mathrm{PI}} := \sum_{s \in S} p_s \min_{x,y} \{c^\top x + (q^s)^\top y : (x,y) \in K^s\}.$$

Although the perfect information bound can computed by solving a separate subproblem for each scenario, it is often better than the LP relaxation bound because each subproblem is a MIP.

Proposition 2. *The following equality holds:*

$$z_{\mathrm{PI}} = \min_{x,\theta_s} \left\{ c^\top x + \sum_{s \in S} p_s \theta_s : c^\top x + \theta_s \geq Q_s^*(c,1), s \in S \right\}. \tag{15}$$

Proof. We first show the "≤" direction. It follows by observing that if $(x, \{\theta_s\}_{s \in S})$ satisfies $c^\top x + \theta_s \geq Q_s^*(c,1)$ for all $s \in S$, then

$$\begin{aligned} c^\top x + \sum_{s \in S} p_s \theta_s &\geq c^\top x + \sum_{s \in S} p_s (-c^\top x + Q_s^*(c,1)) \\ &= \sum_{s \in S} p_s Q_s^*(c,1) \\ &= \sum_{s \in S} p_s \min_{x,y} \{c^\top x + (q^s)^\top y : (x,y) \in K^s\} \\ &= z_{\mathrm{PI}}. \end{aligned} \tag{16}$$

Inequality (15) follows from the fact that the first inequality in (16) must hold at equality for all optimal solutions of (15). This is because if $c^\top x + \theta_s > Q_s^*(c,1)$ for some $s \in S$, we can obtain a feasible solution with a lower objective value by decreasing the value of θ_s. □

From this example, we see that Lagrangian cuts with $(\pi, \pi_0) = (c, 1)$ can already yield the perfect information bound. One interpretation of this example is that useful Lagrangian cuts can be obtained even when searching in a heavily constrained set of coefficients, which motivates our study of restricted separation of Lagrangian cuts.

4.1. General Framework

We propose solving the following restricted version of (14) to search for a Lagrangian cut that separates a solution $(\hat{x}, \hat{\theta}_s)$:

$$\max_{\pi, \pi_0} \{Q_s^*(\pi, \pi_0) - \pi^\top \hat{x} - \pi_0 \hat{\theta}_s : (\pi, \pi_0) \in \Pi_s\}, \tag{17}$$

where Π_s can be any convex compact subset of $\mathbb{R}^n \times \mathbb{R}_+$, which is not necessarily defined by a neighborhood of the origin.

Although setting Π_s to be a neighborhood of the origin would give an exact separation of Lagrangian cuts, our proposal is to use a more restricted definition of Π_s with the goal of obtaining a separation problem that can be solved faster. As one might expect, the strength of the cut generated from (17) will heavily depend on the choice of Π_s. We propose to define Π_s to be the span of a small set of vectors. Specifically, we require

$$\pi = \sum_{k=1}^{K} \beta_k \pi^k$$

for some $\beta \in \mathbb{R}^K$, where $\{\pi^k\}_{k=1}^K$ are chosen vectors. If $\{\pi^k\}_{k=1}^K$ span $\mathbb{R}^n$, then this approach reduces to exact separation. However, for smaller K, we expect a trade-off between computation time and strength of cuts. A natural possibility for the vectors $\{\pi^k\}_{k=1}^K$, which we explore in Section 4.3, is to use the coefficients of Benders cuts derived from the LP relaxations.

4.2. Solution of (17)

We present a basic cutting-plane algorithm for solving (17) in Algorithm 1. The classical cutting-plane method used in the algorithm can be replaced by other bundle methods, for example, the level method (Lemaréchal et al. 1995). Within the solution process of (17), the function Q_s^* is approximated by an upper bounding cutting-plane model $\hat{Q}_s^* : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}$ defined by

$$\hat{Q}_s^*(\pi, \pi_0) = \min_{z, \theta_s^z} \{\pi^\top z + \pi_0 \theta_s^z : (z, \theta_s^z) \in \hat{E}^s\} \tag{18}$$

for $(\pi, \pi_0) \in \mathbb{R}^n \times \mathbb{R}_+$, where $\hat{E}^s$ is a finite subset of E^s. In our implementation, for each scenario s, all feasible first-stage solutions obtained from evaluating $Q_s^*(\pi, \pi_0)$ and the corresponding feasible second-stage costs are added into $\hat{E}^s$. Specifically, each time we solve (12) for some (π, π_0), we collect all optimal and suboptimal solutions (x, y) from the solver and store the first-stage solution $z = x$ and the corresponding feasible second-stage cost $\theta_s^z = (q^s)^\top y$. To guarantee finite convergence, we assume that at least one of the optimal solutions obtained when solving (12) is an extreme point of $\mathrm{conv}(K^s)$. Although we may solve Problem (17) associated with different $\hat{x}, \hat{\theta}_s$ and Π_s, the previous $\hat{Q}_s^*$ remains a valid overestimate of Q_s^*. Therefore, whenever we need to solve (12) with some new $\hat{x}, \hat{\theta}_s$ and Π_s, we use the currently stored $\hat{E}^s$ to initialize $\hat{Q}_s^*$. In our implementation, to avoid unboundedness of $\hat{Q}_s^*$, $\hat{E}^s$ is initialized by the perfect information solution $(z^s, (q^s)^\top y^s)$, where $(z^s, y^s) \in \mathrm{argmin}_{x,y} \{c^\top x + (q^s)^\top y : (x,y) \in K^s\}$.

Algorithm 1 (Solution of (17))
Input: $(\hat{x}, \hat{\theta}_s)$, Π_s, $\hat{E}^s$
Output: (π^*, π_0^*), $\hat{E}^s$

1. Initialize UB$\leftarrow +\infty$, LB$\leftarrow -\infty$
2. **while** *UB* > 0 *and UB-LB* $\geq \delta UB$ **do**
3. Solve UB $\leftarrow \max_{\pi,\pi_0}\{\hat{Q}_s^*(\pi,\pi_0) - \pi^\top \hat{x} - \pi_0\hat{\theta}_s : (\pi,\pi_0) \in \Pi_s\}$, and collect solution $(\pi,\pi_0) = (\hat{\pi},\hat{\pi}_0)$
4. Solve (12) to evaluate $Q_s^*(\hat{\pi},\hat{\pi}_0)$. Update $\hat{E}^s$ and $\hat{Q}_s^*$ with optimal and suboptimal solutions obtained while solving (12)
5. **if** $LB < Q_s^*(\hat{\pi},\hat{\pi}_0) - \hat{\pi}^\top\hat{x} - \hat{\pi}_0\hat{\theta}_s$ **then**
6. LB $\leftarrow Q_s^*(\hat{\pi},\hat{\pi}_0) - \hat{\pi}^\top\hat{x} - \hat{\pi}_0\hat{\theta}_s$
7. $(\pi^*,\pi_0^*) \leftarrow (\hat{\pi},\hat{\pi}_0)$
8. **end**
9. **end**

Although Algorithm 1 is for the most part a standard cutting-plane algorithm, an important detail is the stopping condition that uses a relative tolerance $\delta \in [0, 1)$. Convergence of Algorithm 1 follows standard arguments for a cutting-plane algorithm. Specifically, because the set of extreme-point optimal solutions from (12) is finite, there will eventually be an iteration where the optimal solution obtained when solving (12) is already contained in the set $\hat{E}^s$. When that occurs, after updating LB it will hold that UB = LB. At that point, it either holds that UB ≤ 0 or UB-LB $= 0 < \delta$UB and the algorithm terminates. If the optimal value of (17) is positive, then UB will be strictly positive throughout the algorithm. In that case, Algorithm 1 returns a violating Lagrangian cut with cut violation LB$\in ((1-\delta)\text{UB}, \text{UB}]$. The value of δ controls the tradeoff between the strength of the cut and the running time. Conversely, if the optimal value of (17) is nonpositive, then we can terminate as soon as UB ≤ 0, because in this case, we are assured a violated cut will not be found.

We next demonstrate that even when using $\delta > 0$, it is possible to attain the full strength of the Lagrangian cuts for a given set Π_s when applying them repeatedly within a cutting-plane algorithm for solving a relaxation of Problem (4). To make this result concrete, we state such a cutting-plane algorithm in Algorithm 2. At each iteration t, we approximate the true objective function by the cutting-plane model:

$$c^\top x + \sum_{s\in S} p_s \hat{Q}_s^t(x)$$
$$= c^\top x + \sum_{s\in S} p_s \min_{x,\theta_s}\{\theta_s : \bar{\pi}^\top x + \bar{\pi}_0\theta_s \geq \bar{\tau}, \forall(\bar{\pi}_0,\bar{\pi},\bar{\tau}) \in \Psi_s^t\},$$

where Ψ_s^t represents the Benders cuts and Lagrangian cuts associated with scenario s that have been added to the master relaxation up to iteration t. At iteration t, a master problem based on the current cutting-plane model is solved and generates candidate solution $(x^t, \{\theta_s^t\}_{s\in S})$. For each scenario s, the restricted Lagrangian cut separation problem (17) is solved using Algorithm 1 with Π_s restricted to be a (typically low-dimensional) set Π_s^t. The cutting-plane model is then updated by adding the violated Lagrangian cuts (if any) for each scenario.

We show in the next theorem that if we search on a static compact Π_s and keep adding Lagrangian cuts by solving (17) to a δ relative tolerance with $\delta \in [0, 1)$, the algorithm will converge to a solution that satisfies all the restricted Lagrangian cuts.

Theorem 3. *In Algorithm 2, let $\Pi_s^t \equiv \Pi_s$ be a compact set independent of t and assume that at each iteration of Algorithm 2, for each scenario $s \in S$, a Lagrangian cut (if a violated one exists) is generated by solving (17) to δ relative tolerance with $\delta \in [0, 1)$. Then for each scenario s, every accumulation point $(\bar{x}, \bar{\theta}_s)$ of the sequence $\{(x^t, \theta_s^t)\}_{t=1}^\infty$ satisfies $(\bar{x}, \bar{\theta}_s) \in P_s(\Pi_s)$.*

Algorithm 2 (Restricted Separation of Lagrangian Cuts)

Input: $\{\hat{Q}_s^0\}_{s\in S}$
Output: $\{\hat{Q}_s^t\}_{s\in S}$

1. Initialize $t \leftarrow 0$, cutfound $\leftarrow$ True
2. **while** *cutfound* = *True* **do**
3. cutfound $\leftarrow$ False;
4. Solve $\min_{x,\theta_s}\{c^\top x + \sum_{s\in S} p_s\theta_s : \theta_s \geq \hat{Q}_s^t(x), Ax \geq b\}$ to obtain solution $(x^t, \{\theta_s^t\}_{s\in S})$
5. **for** $s \in S$ **do**
6. Solve Benders subproblem (6) and obtain dual solution $\mu^{t,s}$;
7. **if** $(\mu^{t,s})^\top T^s x^t + \theta_s^t < (\mu^{t,s})^\top h^s$ **then**
8. Update $\hat{Q}_s^t$ using (7) defined by $\mu = \mu^{t,s}$ to obtain $\hat{Q}_s^{t+1}$;
9. cutfound $\leftarrow$ True;
10. **end**
11. **end**
12. **if** *coutfound* = *False* **then**
13. **for** $s \in S$ **do**
14. Generate $\Pi_s^t \subseteq \mathbb{R}^n$
15. Solve (17) using Algorithm 1 with $(\hat{x}, \hat{\theta}_s) = (x^t, \theta_s^t)$ and $\Pi_s = \Pi_s^t$, and collect solution $(\pi^{t,s}, \pi_0^{t,s}) := (\pi^*, \pi_0^*)$
16. **if** $\hat{Q}_s^*(\pi^{t,s}, \pi_0^{t,s}) - (\pi^{t,s})^\top x^t - \pi^{t,s}\theta_s^t > 0$ **then**
17. Update $\hat{Q}_s^t$ using cut (13) with $(\pi,\pi_0) = (\pi^{t,s}, \pi_0^{t,s})$ to obtain $\hat{Q}_s^{t+1}$;
18. cutfound $\leftarrow$ True;
19. **end**
20. **end**
21. **end**
22. $t \leftarrow t + 1$
23. **end**

Proof. Note that Q_s^* is a piecewise linear concave function. Therefore, Q_s^* is Lipschitz continuous. Assume $\{(x^{i_t}, \theta_s^{i_t})\}_{t=1}^\infty$ is any fixed subsequence of $\{(x^t, \theta_s^t)\}_{t=1}^\infty$ that converges to a point $(\bar{x}, \bar{\theta}_s)$. Assume for contradiction that

$$\max_{\pi,\pi_0}\{Q_s^*(\pi,\pi_0) - \pi^\top\bar{x} - \pi_0\bar{\theta}_s : (\pi,\pi_0) \in \Pi_s\} =: \epsilon > 0.$$

Because Π_s is bounded, there exists τ such that

$$\max_{\pi,\pi_0}\{|\pi^\top(x^{i_t}-\bar{x})+\pi_0(\theta_s^{i_t}-\bar{\theta}_s)|:(\pi,\pi_0)\in\Pi_s\} \le \epsilon/2 \text{ for all } t\ge\tau.$$

This implies that for all $t \ge \tau$,

$$\begin{aligned}
&\max_{\pi,\pi_0}\{Q_s^*(\pi,\pi_0)-\pi^\top x^{i_t}-\pi_0\theta_s^{i_t}:(\pi,\pi_0)\in\Pi_s\}\\
&=\max_{\pi,\pi_0}\{Q_s^*(\pi,\pi_0)-\pi^\top\bar{x}-\pi_0\bar{\theta}_s-\pi^\top(x^{i_t}-\bar{x})\\
&\quad-\pi_0(\theta_s^{i_t}-\bar{\theta}_s):(\pi,\pi_0)\in\Pi_s\}\\
&\ge\max_{\pi,\pi_0}\{Q_s^*(\pi,\pi_0)-\pi^\top\bar{x}-\pi_0\bar{\theta}_s\\
&\quad-|\pi^\top(x^{i_t}-\bar{x})+\pi_0(\theta_s^{i_t}-\bar{\theta}_s)|:(\pi,\pi_0)\in\Pi_s\}\\
&\ge\max_{\pi,\pi_0}\{Q_s^*(\pi,\pi_0)-\pi^\top\bar{x}-\pi_0\bar{\theta}_s:(\pi,\pi_0)\in\Pi_s\}\\
&\quad-\max_{\pi,\pi_0}\{|\pi^\top(x^{i_t}-\bar{x})+\pi_0(\theta_s^{i_t}-\bar{\theta}_s)|:(\pi,\pi_0)\in\Pi_s\}\\
&\ge\epsilon-\epsilon/2=\epsilon/2,
\end{aligned}$$

where the second "$\ge$" follows from $\max_x\{f(x)\}\ge\max_x\{f(x)+g(x)\}-\max_x\{g(x)\}$. Let $(\pi^{(i_t)},\pi_0^{(i_t)})$ denote the multiplier associated with the Lagrangian cut added at iteration i_t. For any $t'>t\ge\tau$,

$$Q_s^*\left(\pi^{(i_t)},\pi_0^{(i_t)}\right)-\left(\pi^{(i_t)}\right)^\top x^{i_{t'}}-\pi_0^{(i_t)}\theta_s^{i_{t'}}\le 0$$

because the Lagrangian cut associated with $(\pi^{(i_t)},\pi_0^{(i_t)})$ was added before iteration $i_{t'}$. Conversely,

$$\begin{aligned}
&Q_s^*\left(\pi^{(i_{t'})},\pi_0^{(i_{t'})}\right)-\left(\pi^{(i_{t'})}\right)^\top x^{i_{t'}}-\pi_0^{(i_{t'})}\theta_s^{i_{t'}}\\
&\ge(1-\delta)\max_{\pi,\pi_0}\{Q_s^*(\pi,\pi_0)-\pi^\top x^{i_{t'}}-\pi_0\theta_s^{i_{t'}}:(\pi,\pi_0)\in\Pi_s\}\\
&\ge(1-\delta)\epsilon/2.
\end{aligned}$$

Because the sequence $\{(x^{i_t},\theta_s^{i_t})\}_{t=1}^\infty$ is bounded and function Q_s^* is Lipschitz continuous, there exists $\rho>0$ such that $\|(\pi^{(i_t)},\pi_0^{(i_t)})-(\pi^{(i_{t'})},\pi_0^{(i_{t'})})\|\ge\rho$ for all $t'>t\ge\tau$. This contradicts with the fact that Π_s is bounded. □

In practice, we allow Π_s^t to change with iteration t, but Theorem 3 provides motivation for solving (17) to a δ relative tolerance. In Section 5.2.1, we present results of experiments demonstrating that, for at least one problem class, using δ much larger than zero can lead to significantly faster improvement in the lower bound obtained when using Lagrangian cuts. When allowing Π_s^t to change with iteration t, Theorem 3 no longer applies. Instead, in this setting, we follow common practice and terminate the cut-generation process after no more cuts are found or when it is detected that the progress in increasing the lower bound has significantly slowed (see Section 5.3 for an example stopping condition).

4.3. Choice of Π_s

The primary difference between the exact separation (14) and the proposed restricted separation problem (17) is the restriction constraint $(\pi,\pi_0)\in\Pi_s$. Ideally, we would like to choose Π_s so that (17) is easy to solve, in particular avoiding evaluating the function Q_s^* too many times, while also yielding a cut (13) that has a similar strength compared with the Lagrangian cut corresponding to the optimal solution of (14). Our proposal that aims to satisfy these properties is to define Π_s to be the span of past Benders cuts with some normalization. Specifically, we propose

$$\Pi_s=\left\{(\pi,\pi_0):\exists\beta\in\mathbb{R}^K \text{ s.t. } \pi=\sum_{k=1}^K\beta_k\pi^k,\ \alpha\pi_0+\|\pi\|_1\le 1,\pi_0\ge 0\right\}. \tag{19}$$

Here K and α are both predetermined parameters, and $\{\pi^k\}_{k=1}^K$ are the coefficients of the last K Benders cuts corresponding to scenario s. The choice of $\{\pi^k\}_{k=1}^K$ is important. The motivation for this choice is that the combination of cut coefficients from the LP relaxation could give a good approximation of the tangent of the supporting hyperplanes of Q_s at an (near-)optimal solution. The normalization used here is similar to the one proposed for the selection of Benders cuts by Fischetti et al. (2010).

Different normalization leads to different cuts. Our second proposal is to use

$$\Pi_s=\left\{(\pi,\pi_0):\exists\beta\in\mathbb{R}^K \text{ s.t. } \pi=\sum_{k=1}^K\beta_k\pi^k,\ \alpha\pi_0+\|\beta\|_1\le 1,\pi_0\ge 0\right\}, \tag{20}$$

which replaces $\|\pi\|_1$ in (19) by $\|\beta\|_1$. This normalization may tend to lead to solutions with sparse β. An advantage of this normalization is that it leads to a MIP approximation for optimally choosing π_k's from a set of candidate vectors, which we discuss next.

Let $V^s=\{v^k\}_{k\in I_s}$ be a given (potentially large) collection of coefficient vectors for a scenario $s\in S$. We construct a MIP model for selecting a subset of these vectors of size at most K, to then use in (20). Specifically, we introduce a binary variable z_k for each $k\in I_s$, where $z_k=1$ indicates we select v^k as one of the π^k vectors to use in (20). Using these decision variables, the problem of selecting a subset of at most K vectors that maximizes the normalized violation can be formulated as the following mixed-integer nonlinear program:

$$\max_{\beta_k,z_k,\pi,\pi_0} Q_s^*(\pi,\pi_0)-\pi^\top\hat{x}-\pi_0\hat{\theta}_s \tag{21}$$

$$\text{s.t. } \pi=\sum_{k\in I_s}\beta_k v^k, \tag{22}$$

$$\alpha\pi_0+\sum_{k\in I_s}|\beta_k|\le 1, \tag{23}$$

$$-z_k \le \beta_k \le z_k, \qquad k \in I_s, \tag{24}$$

$$\sum_{k \in I_s} z_k \le K, \tag{25}$$

$$z_k \in \{0, 1\}, \qquad k \in I_s, \tag{26}$$

$$\pi_0 \ge 0. \tag{27}$$

Objective and Constraints (22) and (23) match Formulation (20), whereas Constraint (25) limits the number of selected vectors to at most K, and (24) enforces that if a vector is not selected then the weight on that vector must be zero. This problem is computationally challenging because of the implicit function Q_s^*. However, if we replace Q_s^* by its current cutting-plane approximation $\hat{Q}_s^*$, then the problem can be solved as a MIP. Specifically, the MIP approximation of (21)–(27) is given by

$$\max_{\beta_k, z_k, \pi, \pi_0, \tau} \tau - \pi^\top \hat{x} - \pi_0 \hat{\theta}_s$$
$$\text{s.t. } \tau \le \pi^\top z + \pi_0 \theta_s^z,$$
$$(22)\text{–}(27). \tag{28}$$

After solving this approximation, we choose the vectors $\{\pi_k\}_{k=1}^{K'}$ to be the vectors v_k with $z_k = 1$. Given this basis, we then solve (17) with Π_s defined in (20) to generate Lagrangian cuts. The basis size K' in this case can be strictly smaller than K. Observe that the optimal value of (28) gives an upper bound of (21)–(27). Therefore, if the optimal objective value of (28) is small, this would indicate we would not be able to generate a highly violated Lagrangian cut for this scenario, and hence, we can skip this scenario for generating Lagrangian cuts.

5. Computational Study

We report our results from a computational study investigating different variants of our proposed methods for generating Lagrangian cuts and also comparing these to existing approaches. In Section 5.2, we investigate the bound closed over time by just adding cuts to the LP relaxation. In Section 5.3, we investigate the impact of using our cut-generation strategy to solve our test instances to optimality within a simple branch-and-cut algorithm.

We conduct our study on three test problems. The first two test problems are standard SIP problems that can be found in literature: the stochastic server location problem (SSLP, binary first-stage and mixed-integer second-stage) introduced by Ntaimo and Sen (2005) and the stochastic network interdiction problem (SNIP, binary first-stage and continuous second-stage) by Pan and Morton (2008). We also create a variant of the stochastic server location problem (SSLPV) that has mixed-binary first-stage variables and continuous second-stage variables. Our test set includes 24 SSLP instances with 20–50 first-stage variables, 2,000–2,100 second-stage variables, and 50–200 scenarios. The SSLPV test instances are the same as for SSLP, except the number of first-stage variable is doubled, with half of them being continuous. We use 20 instances of the SNIP problem having 320 first-stage variables, 2,586 second-stage variables, and 456 scenarios. More details of the test problems and instances can be found in Section S.2 of the online appendix. All problem instances and code for running the experiments are available at https://github.com/rchen234/SIP_LAG.

We consider in total six implementations of Lagrangian cuts:

1. *StrBen*: Strengthened Benders cuts from Zou et al. (2019), that is, (10) with λ equal to the negative of the Benders cut coefficients each time a Benders cut is generated.
2. *BDD*: BDD_3 is from Rahmaniani et al. (2020). This algorithm uses a multiphase implementation that first generates Benders cuts and strengthened Benders cuts and then generates Lagrangian cuts based on a regularized inner approximation for separation.
3. *Exact*: Exact separation of Lagrangian cuts of the form (17) with $\Pi_s = \{(\pi, \pi_0) : \alpha\pi_0 + \|\pi\|_1 \le 1, \pi_0 \ge 0\}$.
4. *Rstr1*: Restricted separation of Lagrangian cuts (17) with Π_s defined in (19) and $\{\pi_k\}_{k=1}^K$ being the coefficients of the last K Benders cuts corresponding to scenario s.
5. *Rstr2*: Restricted separation of Lagrangian cuts (17) with Π_s defined in (20) and $\{\pi_k\}_{k=1}^K$ being the coefficients of the last K Benders cuts corresponding to scenario s.
6. *RstrMIP*: Restricted separation of Lagrangian cuts (17) with Π_s defined in (20) and $\{\pi_k\}_{k=1}^K$ determined by solving the MIP approximation (28) with $V^s = \{v^k\}_{k \in I_s}$ being the set of all Benders cuts coefficients that have been generated for scenario s.

5.1. Implementation Details

All experiments are run on a Windows laptop with 16 GB RAM and an Intel Core i7-7660U processor running at 2.5 GHz. All LPs, MIPs, and convex quadratic programs (QPs) are solved using the optimization solver Gurobi 8.1.1 for SNIP and SSLP instances and are solved using Gurobi 9.0.3 for SSLPV instances.

In Algorithm 1, we also terminate if the upper bound is small enough ($< 10^{-6}(|\hat{\theta}_s| + 1)$) or two consecutive multipliers are too close to each other (infinity norm of the difference $< 10^{-10}$). A computational issue that can occur when solving (17) is that if we obtain a solution with $\pi_0 = 0$ (or very small), the solution of (x^*, y^*) of (12) does not provide information about $Q_s(x^*)$ because the coefficients associated with the y variables are equal zero. In that case, if we store the "optimal" second-stage cost $(q^s)^\top y^*$ and use it together with x^* to define a cut in the cutting-plane model (18),

this second-stage cost value $(q^s)^\top y^*$ can be very far away from $Q_s(x^*)$ and lead to a very loose cut. Therefore, in our implementation, when π_0 is small ($< 10^{-4}$), after finding (x^*, y^*) from solving (12), we reevaluate $Q_s(x^*)$ by solving the scenario subproblem (2) with $x = x^*$ fixed, and use this value as $\theta_s^{x^*}$.

For adding Benders cuts in Algorithm 2, the initialization of $\hat{Q}_s^0$ is obtained by one iteration of Benders cuts. Specifically, to avoid unboundedness, we solve $\min_x\{0 : Ax \geq b\}$ to obtain a candidate solution and add the corresponding Benders cuts for all scenarios. As described in Algorithm 2, lines 2–11, Benders cuts from the LP relaxation are added using the classical cutting-plane method (Kelley 1960). This implementation was used for the SNIP test problem. For the SSLP test problem, we found this phase took too long to converge, so the phase of adding Benders cuts as described in lines 2–11 was replaced by the level method (Lemaréchal et al. 1995). Benders cuts are only added if the violation is at least $10^{-4}(|\hat{\theta}_s| + 1)$. We store the coefficients of all Benders cuts that have been added to the master problem. For each $s \in S$, we define Π_s in the definitions of *Rstr1* and *Rstr2* based on the coefficients of the K Benders cuts that have been most recently found by the algorithm for scenario s.

We use Algorithm 1 for solving the separation problem, with one exception. When doing exact separation of Lagrangian cuts (i.e., without constraining Π_s to be low-dimensional set), we implement a level method for solving (17), because in that case we found Algorithm 1 converged too slowly. Based on preliminary experiments, the normalization coefficient α in (19) or (20) is set to be 1 for tests on SSLP and SSLPV instances and 0.1 for tests on SNIP instances.

In all SSLP, SSLPV, and SNIP instances, the set $\{x : Ax \leq b\}$ is integral, and we have relatively complete recourse. Therefore, no feasibility cuts are necessary. Only Lagrangian cuts with $\pi_0 \geq 10^{-6}$ are added in our tests.

For the *BDD* tests, we follow the implementation of Rahmaniani et al. (2020) and set the regularization coefficient as $\delta_0 = 0.01$ for SSLP and SSLPV instances and as $\delta_0 = 100$ for SNIP instances. These values are chosen based on preliminary experiments as producing the best improvement in bound over time.

We have also tested our instances using the general purpose SIP solver DSP (Kim and Zavala 2018, Kim et al. 2019, Kim and Dandurand 2022). All DSP experiments are run on an Ubuntu virtual machine built on the same Windows machine using the binary version of DSP 1.1.3 with CPLEX 12.10 as the MIP solver.

5.2. LP Relaxation Results

We first compare results obtained from different implementations of Lagrangian cuts on our test problems and study the impact of different parameters in the algorithm. The stopping condition of Algorithm 2 for all root node experiments is that either the algorithm hits the one-hour time limit or there is no cut that can be found by the algorithm (with a 10^{-6} relative tolerance).

To compare the overall performance of the methods visually across many test instances, we present results in the form of a γ-gap-closed profile. Given a set of problem instances P and a set of cut generation methods M, let $\bar{g}_p$ denote the largest gap closed by any of the methods (compared with the LP bound) in M for instance $p \in P$. The γ-gap-closed profile is defined with respect to certain threshold $\gamma \in (0, 1]$. Given the value of γ, we define $t_{p,m}^\gamma$ as the earliest time of closing the gap by at least $\gamma\bar{g}_p$ for problem $p \in P$ with method $m \in M$. If method m did not close a gap at least $\gamma\bar{g}_p$ before termination, we define $t_{p,m}^\gamma = \infty$. The γ-gap-closed profile is a figure representing the cumulative growth (distribution function) of the γ-gap-closed ratio $\rho_m^\gamma(\tau)$ s over time τ, where

$$\rho_m^\gamma(\tau) = \frac{|\{p \in P : t_{p,m}^\gamma \leq \tau\}|}{|P|}.$$

In this paper, P is either the set of all SSLP test instances, all SNIP test instances, or all SSLPV instances, and M is the set of all methods with all parameters we have tested. Therefore, for each instance p, $\bar{g}_p$ is defined to be the largest gap we have been able to close with Lagrangian cuts. We set parameter γ equal to either 0.75 or 0.95 for summarizing the information regarding time needed for obtaining what we refer to as a "reasonably good bound" ($\gamma = 0.75$) or a "strong bound" ($\gamma = 0.95$).

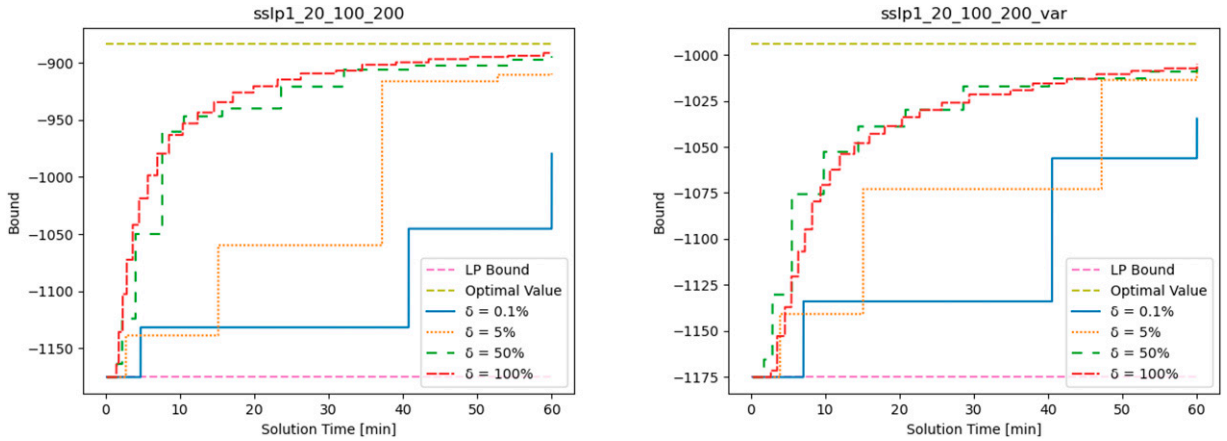
To illustrate the behavior of these algorithms, we also present the lower bound obtained over time using different cut generation schemes on some individual instances.

5.2.1. Relative Gap Tolerance δ.

We first examine the impact of varying the relative gap tolerance δ. Experiments are run with parameter δ being 0.1%, 5%, 50%, and 100%. A small δ value (e.g., 0.1%) means solving (17) to near optimality, which may generate strong cuts but can be time-consuming to generate each cut. Using large δ values might sacrifice some strength of the cut in an iteration for an earlier termination.

For different δ values, in Figure 1, we compare the bounds obtained by *Exact* over time. As the trend is quite similar for instances within the same problem class, we only report the convergence profiles for one SSLP instance (sslp1_20_100_200) and one SSLPV instance (sslp1_20_100_200_var). For the SNIP instances, which have many more first-stage variables, *Exact* often cannot finish one iteration of separation within

Figure 1. (Color online) Convergence Profile of *Exact* on an SSLP Instance (Left) and SSLPV Instance (Right) with Varying δ Values



an hour, so we put the convergence profile of *Exact* for one SNIP instance in Section S.3 of the online appendix. For the SSLP and SSLPV instances, a large δ value tends to close the gap faster than a small δ value. Plots providing the same comparison for *Rstr1* with $K = 10$ are provided in Section S.3 of the online appendix. For *Rstr1*, a large δ value helps improve the bound efficiently for SSLP and SSLPV instances,

Figure 2. (Color online) A γ-Gap-Closed Profile for SSLP Instances Obtained by *Exact* (Top) and *Rstr1* ($K = 10$; Bottom) with $\gamma = 0.75$ (Left) or $\gamma = 0.95$ (Right) and Varying δ Values

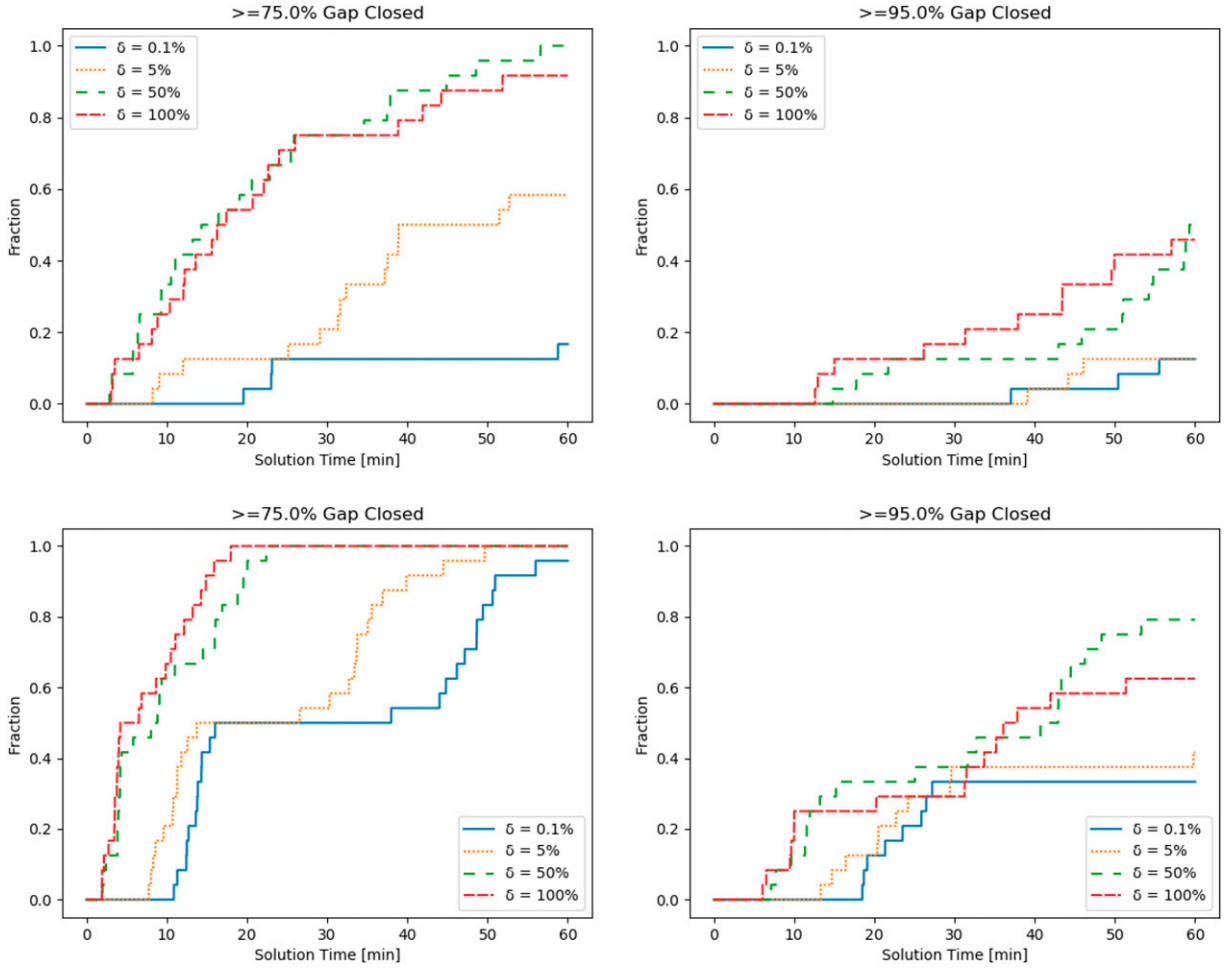
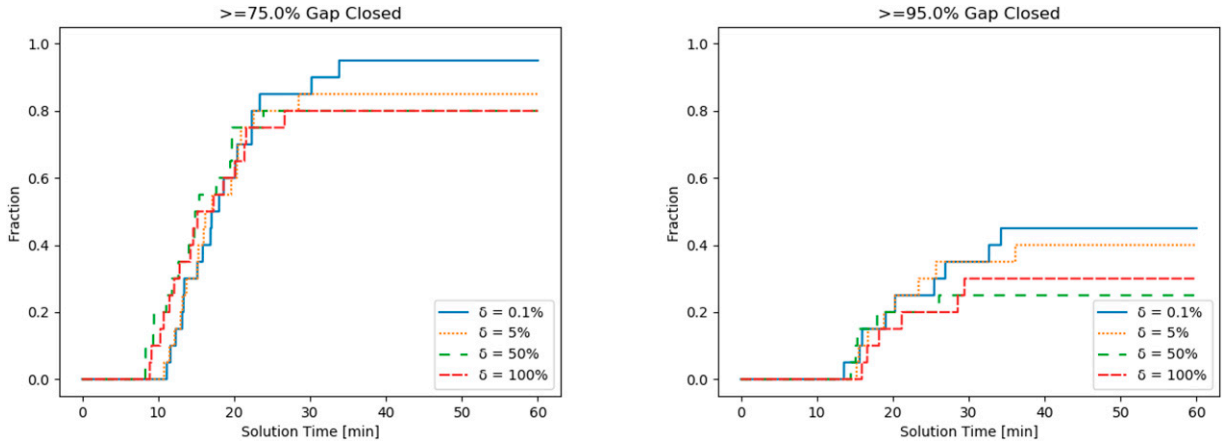


Figure 3. (Color online) A γ-Gap-Closed Profile for SNIP Instances Obtained by *Rstr1* with $\gamma = 0.75$ (Left) or $\gamma = 0.95$ (Right), $K = 10$, and Varying δ Values



whereas the choice of δ has little impact on the performance of the SNIP instances.

To summarize the impact of δ over the full set of test instances, we plot the 0.75-gap-closed profile and 0.95-gap-closed profile with different δ values for SSLP and SNIP instances in Figures 2 and 3. We omit the gap-closed profiles for the SSLPV instances because they are similar to those for the SSLP

Figure 4. (Color online) A γ-Gap-Closed Profile for SSLP Instances Obtained by *Rstr1* and *Rstr2* (Top) and *RstrMIP* (Bottom) with $\gamma = 0.75$ (Left) or $\gamma = 0.95$ (Right), $\delta = 50\%$, and Varying K Values

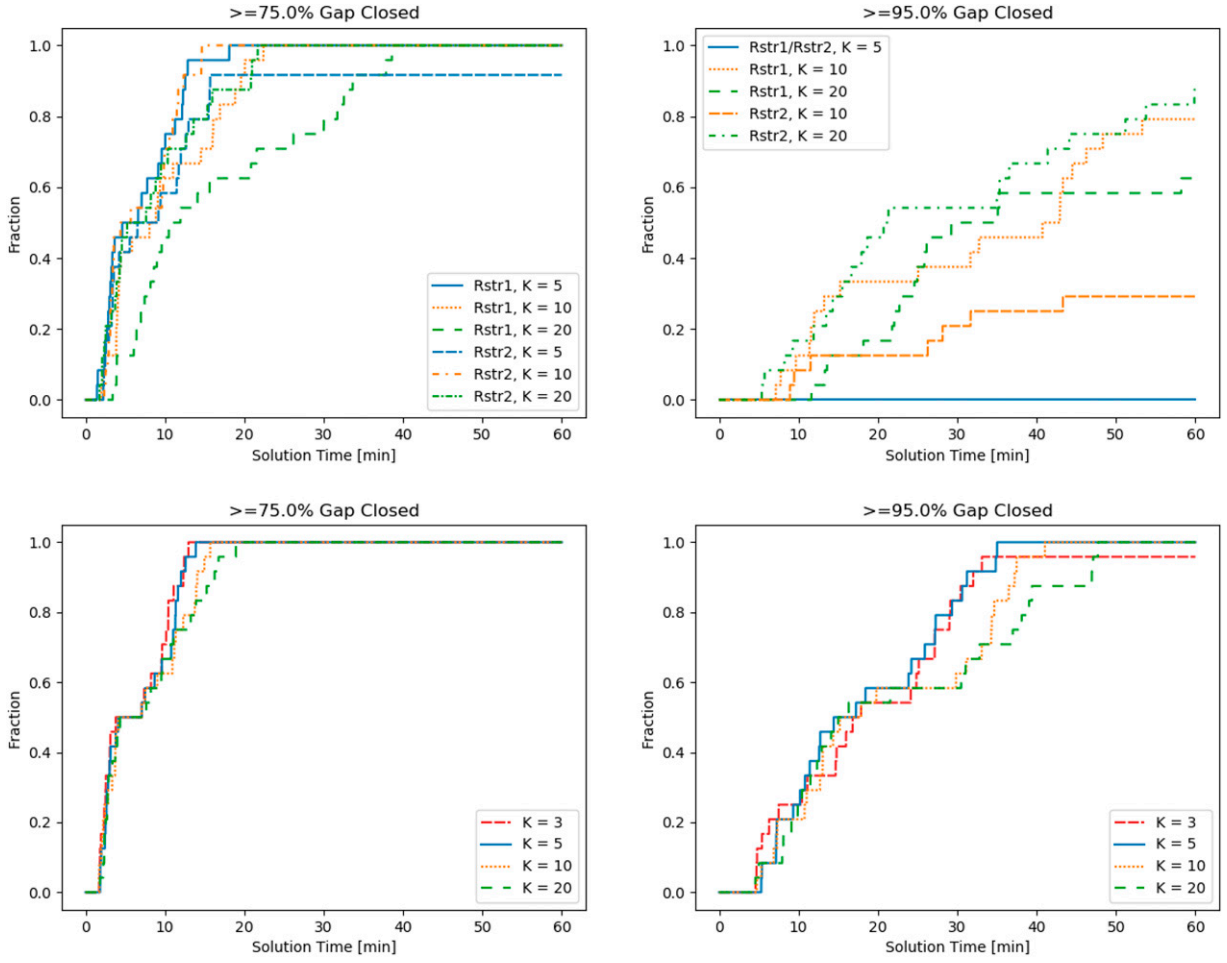
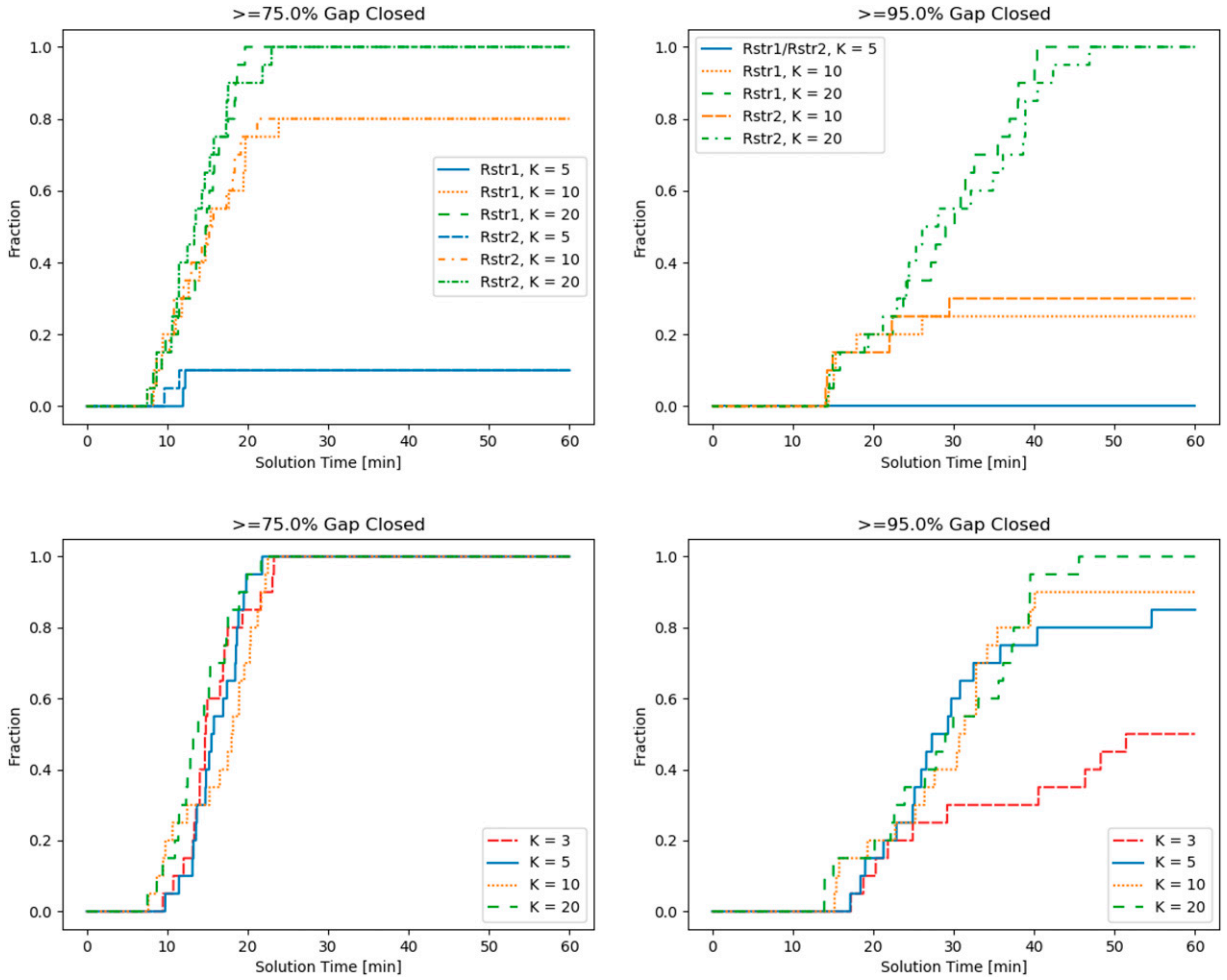


Figure 5. (Color online) A γ-Gap-Closed Profile for SNIP Instances Obtained by *Rstr1* and *Rstr2* (Top) and *RstrMIP* (Bottom) with $\gamma = 0.75$ (Left) or $\gamma = 0.95$ (Right), $\delta = 50\%$, and Varying K Values



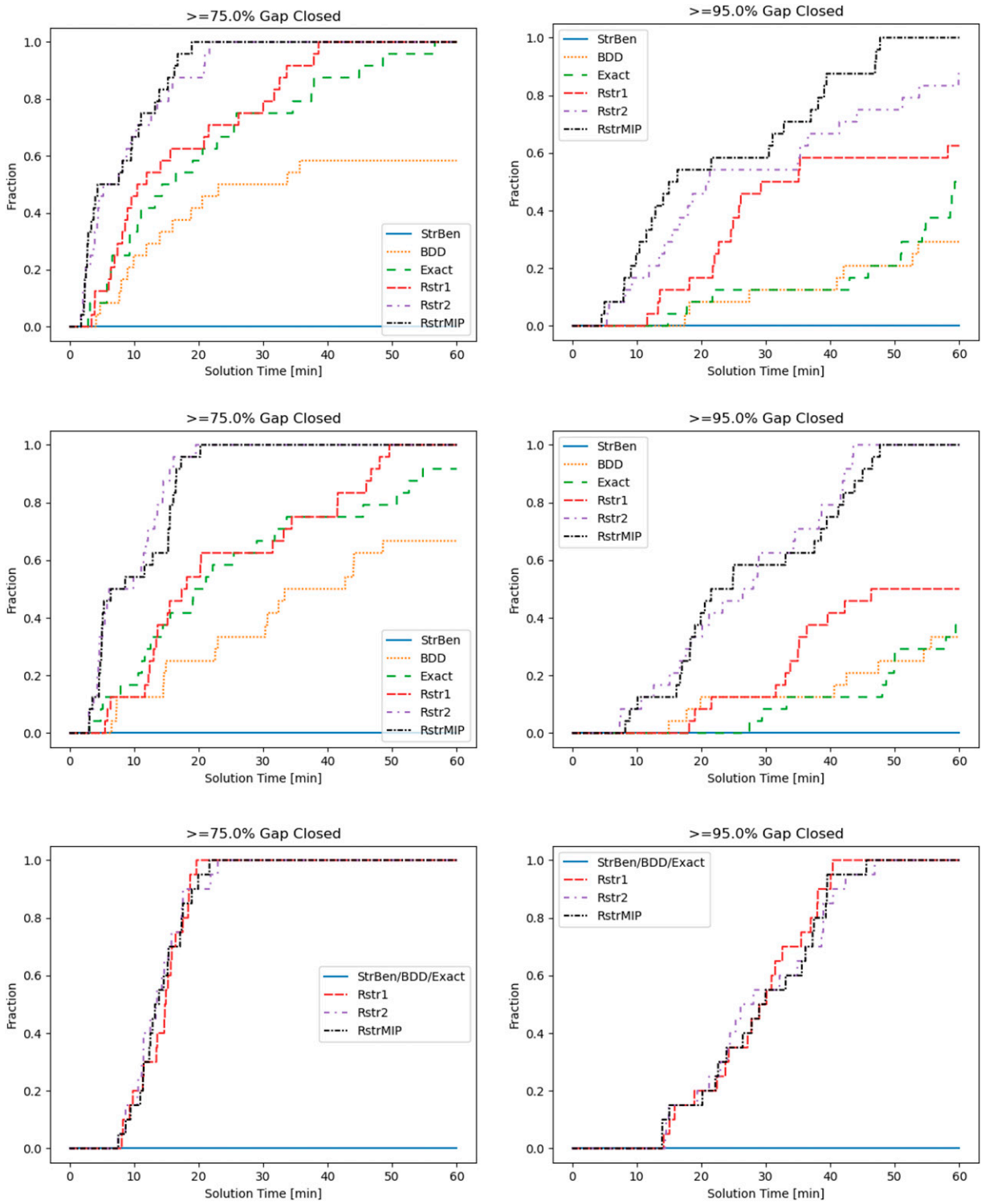
instances. The gap-closed profiles for SNIP instances with *Exact* are omitted as no useful bound is obtained by *Exact* for any of the SNIP instances. For the SSLP and SSLPV instances, it is preferable to use a large δ value as both a reasonable bound (0.75 gap closed) and a strong bound (0.95 gap closed) are obtained earlier when δ is large. For the SNIP instances, there is less clear preference although small δ values perform slightly better. This is because the time spent on each iteration of separation does not vary too much by changing the δ value for the SNIP instances.

Depending on the problem, the choice of δ can have either a mild or big impact. However, we find a relatively large δ, for example, $\delta = 50\%$, has a stable performance on our instance classes.

5.2.2. Basis Size *K*. We next investigate the impact of the choice of the basis size K in *Rstr1*, *Rstr2*, and *RstrMIP*. In Figures 4 and 5, we plot the 0.75-gap-closed profiles and the 0.95-gap-closed profiles with different K values for SSLP and SNIP instances, respectively. The results for SSLPV instances are omitted as they are similar to the results for SSLP instances. Plots showing the evolution of the lower bound obtained using *Rstr1*, *Rstr2*, and *RstrMIP* with different K values on an example SNIP instance; an example SSLP instance and an example SSLPV instance are given in Section S.3 of the online appendix.

As seen in Figures 4 and 5, the trends are quite different for different problem classes when applying *Rstr1*. For SSLP and SSLPV instances, smaller K performs better for obtaining a reasonably good bound faster because the restricted separation is much easier to solve when K is small. However, for SNIP instances, $K = 20$ seems to dominate $K = 5$ and $K = 10$. We believe this is because the time to generate a cut is not significantly impacted by K for the SNIP instances, but $K = 20$ gives much stronger cuts for closing the gap. Comparing *Rstr1* and *Rstr2*, *Rstr2* appears to be less sensitive to K in terms of obtaining a reasonably

Figure 6. (Color online) A γ-Gap-Closed Profile for SSLP Instances (Top), SSLPV Instances (Middle), and SNIP Instances (Bottom) with $\gamma = 0.75$ (Left) or $\gamma = 0.95$ (Right)



good bound for SSLP instances. Conversely, the performance of *Rstr2* is almost identical to the performance of *Rstr1* for SNIP instances. For SSLP and SSLPV instances with *Rstr1* or *Rstr2*, we find that larger K often leads to longer solution time for obtaining a reasonable bound, whereas small K ($K = 5$) is incapable of getting a strong bound. We observe that *RstrMIP* is fairly robust to the choice of the basis size K. One

Table 1. Integrality Gap Closed by Lagrangian Cuts Generated *RstrMIP*

	LP gap (%)			Gap after *RstrMIP* cuts (%)		
Class	Minimum	Maximum	Average	Minimum	Maximum	Average
SSLP	32.9	58.0	44.4	0.0	3.6	0.7
SSLPV	18.2	27.8	23.8	0.0	2.1	0.8
SNIP	22.1	34.2	28.0	0.8	5.9	3.0

particular reason is that the constraint $\sum_{k\in I_s} z_k \le K$ in (28) is often loose in early stages of the algorithm for relatively large Ks as the constraint $\alpha\pi_0 + \|\beta\|_1 \le 1$ tends to select a sparse β and thus a sparse z. For the same instance, different basis sizes result in very close bounds at the end of the run.

The basis size K yields a tradeoff between the strength of cuts and computation time. We find the optimal choice of K for *Rstr1* depends on the instance. Slightly larger K ($K = 20$) often works better for *Rstr2*, whereas the performance of *RstrMIP* is much more robust to the choice of K.

5.2.3. Comparison Between Methods. We present results obtained by strengthened Benders cut (*StrBen*), the Benders dual decomposition (*BDD*), and different variants of our restricted Lagrangian cut separation (*Exact*, *Rstr1*, *Rstr2*, and *RstrMIP*). For all variants of restricted Lagrangian cut separation, we choose relative gap tolerance $\delta = 50\%$. For basis sizes, we set $K = 20$ for *Rstr1*, *Rstr2*, and *RstrMIP*. We plot 0.75-gap-closed profiles and 0.95-gap-closed profiles for these methods in Figure 6.

Apparently, *RstrMIP* has the best performance on SSLP instances in terms of the gap closed over time. On SSLPV instances, the performances of *Rstr2* and *RstrMIP* are similar and much better than other methods. On SNIP instances, the performance of *Rstr1*, *Rstr2*, and *RstrMIP* is almost identical, whereas *RstrMIP* is much more robust in the choice of K as observed in Section 5.2.2. The curves for *BDD* are flat on the bottom plots of Figure 6 as *BDD* fails to close 75% of the gap for all SNIP instances. Convergence profiles for *BDD* on one SSLP instance, one SSLPV instance, and one SNIP instance can be found in Section S.3 of the online appendix.

RstrMIP has the best overall performance and, in particular, is more robust to the choice of K.

5.2.4. Integrality Gaps. In Table 1, we report statistics (minimum, maximum, and average) of the LP integrality gap and the integrality gap obtained after 60 minutes of Lagrangian cut generation by *RstrMIP* with $K = 20$ for each problem class. The integrality gaps are calculated as $(z_{IP} - \text{LB})/|z_{IP}|$, where LB is the lower bound obtained from the Benders model before or after adding the generated Lagrangian cuts. The results show that *RstrMIP* closes the vast majority of the gap to the optimal value. Because the resulting bound closely approximates the optimal value, and the exact Lagrangian dual must lie between this bound and the optimal value, these results imply that, on these test instances, our method approximates the optimal value of the Lagrangian dual and that the Lagrangian dual approximates the optimal value. We emphasize, however, that this experiment is just to illustrate the potential for this method to close a majority of the gap when given enough time: We recommend using an early stopping condition to stop the cut generation process earlier when it is detected that the bound improvement has slowed significantly.

5.3. Tests for Solving to Optimality

Finally, we study the use of our approach for generating Lagrangian cuts within a branch-and-cut method to solve SIPs to optimality, where Benders cuts and integer L-shaped cuts are added using Gurobi lazy constraint callback when a new solution satisfying the integrality constraints is found. We refer to this method as LBC. We consider only *RstrMIP* with $\delta = 50\%$ and $K = 10$ as the implementation of Lagrangian cuts. To reduce the time spent on Lagrangian cut generation at

Table 2. Comparison of Algorithms for Solving SSLP Instances to Optimality

	On $\|S\| = 50$ instances			On $\|S\| = 200$ instances		
	LBC	BBC	DSP	LBC	BBC	DSP
No. solved	12/12	0/12	12/12	12/12	0/12	7/12
Average solution time	1,496	≥ 7,200	1,259	4,348	≥ 7,200	≥ 5,365
Average gap (%)	0.0	18.1	—	0.0	27.0	—
Average B&C time	25	≥ 7,200	—	163	≥ 7,200	—
Average no. of nodes	775	13,808	—	1,411	3,954	—

Table 3. Comparison of Algorithms for Solving SSLPV Instances to Optimality

	On $\|S\| = 50$ instances			On $\|S\| = 200$ instances		
	LBC	BBC	DSP	LBC	BBC	DSP
No. solved	11/12	0/12	3/12	7/12	0/12	2/12
Average solution time	≥2,537	≥7,200	—	≥5,894	≥7,200	—
Average gap (%)	0.1	18.4	—	0.3	20.8	—
Average B&C time	≥605	≥7,197	—	≥1,156	≥7,191	—
Average no. of nodes	≥7,101	≥22,970	—	≥6,300	≥8,945	—

the root node, an early termination rule is applied. We stop generating Lagrangian cuts if the gap closed in the last five iterations is less than 1% of the total gap closed thus far. For comparison, we also experiment with Benders cuts integrated with branch-and-cut (BBC; algorithm described in Section S.1 of the online appendix) and the DSP solver (Kim and Zavala 2018) on the same instances. A two-hour time limit is set for these experiments. Gurobi heuristics are turned off for the branch-and-cut master problem to avoid the bug in Gurobi 8.1.1 that the MIP solution generated by Gurobi heuristics sometimes cannot trigger the callback. (This bug is supposed to be fixed in Gurobi 9.1.)

We present the results obtained by LBC, BBC, and DSP on SSLP instances in Table 2. We report the number of instances at sample size $|S| = 50$ or $|S| = 200$ each algorithm solved within the time limit (No. solved), the total average solution time of each algorithm (Average solution time) including time to generate cuts if relevant, the average ending optimality gap (Average gap (%)), average time spent solving the problem with branch-and-cut *not including the initial cut generation time* (Average B&C time), and the average number of branch-and-bound nodes explored (Average no. of nodes'). The ending optimality gap of a method on an instance is calculated as $(\text{UB} - \text{LB})/\max\{|\text{UB}|, |\text{LB}|\}$, where UB and LB are the upper and lower bounds obtained from the method when the time limit was reached.

All SSLP instances are solved to optimality by LBC within the two-hour time limit with at most a few thousand nodes explored. Conversely, none of the SSLP instances are solved by BBC before hitting the time limit, leaving a relatively large optimality gap. SSLP instances are hard to solve by BBC because neither Benders cuts nor integer L-shaped cuts are strong enough to cut off infeasible points efficiently in the case when second-stage variables are discrete. Comparing with DSP, LBC solves more $|S| = 200$ SSLP instances than DSP withinin the time limit. The performance of LBC is relatively stable compared with DSP in the sense that LBC scales well as $|S|$ increases. The efficiency of DSP heavily depends on the strength of the nonanticipative dual bound. For all instances where DSP outperforms LBC on solution time, no branching is needed for closing the dual gap. We observe that LBC spends most of the solution time on generating Lagrangian cuts for SSLP instances. The branch-and-cut time required after generating Lagrangian cuts is very small.

In Table 3, we report results obtained by these methods for the SSLPV instances. We do not report the average solution time of DSP because for many of the SSLPV instances DSP fails at a very early stage by outputting an infeasible solution. Although the performances of our approaches at the root node is similar for SSLPV as SSLP, we find the SSLPV instances are far more difficult to solve to optimality than the SSLP instances. As with the SSLP instances, BBC is unable to solve any of the SSLPV instances because the Benders cuts are not strong enough. Within the two-hour time limit, only a small fraction of the SSLPV instances can be solved by DSP correctly. On the other hand, LBC solves the majority of the SSLPV instances, and the ending optimality gaps are small for those unsolved instances, showing its capability and stability in dealing with different problem classes.

In Table 4, we report results using these methods to solve the SNIP instances. We observe that no SNIP instance can be solved within two hours by DSP. This

Table 4. Comparison of Algorithms for Solving SNIP Instances to Optimality

	LBC	BBC	DSP
No. solved	20/20	20/20	0/20
Average solution time	2,333	163	≥7,200
Average gap (%)	0.0	0.0	—
Average B&C time	71	116	—
Average no. of nodes	1,231	3,348	—

Table 5. Comparison of LBC and BBC on SNIP Instances with Gurobi Presolve and Cuts Turned Off

	LBC	BBC
No. solved	20/20	9/20
Average solution time	2,385	$\geq 4,981$
Average gap (%)	0.0	2.9
Average B&C time	71	$\geq 4,534$
Average no. of nodes	4,236	$\geq 2,923,914$

is because solving the nonanticipative dual problem of SNIP instances is not completed in two hours. BBC solves the SNIP instances faster than LBC as the time spent on cut generation at the root node is much shorter and the Benders cuts, when combined with Gurobi's presolve and cut generation procedures, are sufficient for keeping the number of nodes small.

To better understand the impact of the Lagrangian cuts versus Benders cuts on their own, we also report results obtained by LBC and BBC with Gurobi presolve and cuts disabled in Table 5. We see that Gurobi presolve and cuts are quite essential for solving SNIP instances when using only Benders cuts. In this case, BBC is unable to solve many of the instances within the time limit. This is because Gurobi presolve and cuts significantly improve the LP bounds for BBC. LBC still solves all SNIP instances, demonstrating the strength of the Lagrangian cuts generated.

6. Conclusion

We derive a new method for generating Lagrangian cuts to add to a Benders formulation of a two-stage SIP. Extensive numerical experiments on three SIP problem classes indicate that the proposed method can improve the bound more quickly than alternative methods for using Lagrangian cuts. When incorporating the proposed method within a simple branch-and-cut algorithm for solving the problems to optimality we find that our method is able to solve one problem class within a time limit that is is not solved by the open-source solver DSP, and performs modestly better on the other problem classes.

A direction of future work is to explore the integration of our techniques for generating Lagrangian cuts with the use of other classes of cuts for two-stage SIPs and also for multistage SIPs via the SDDIP algorithm proposed by Zou et al. (2019).

Acknowledgments

The authors thank Simge Küçükyavuz for helpful feedback on a draft of this paper.

References

Ahmed S, Tawarmalani M, Sahinidis NV (2004) A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Math. Programming* 100(2):355–377.

Avriel M, Williams A (1970) The value of information and stochastic programming. *Oper. Res.* 18(5):947–954.

Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. *Numerical Math.* 4:238–252.

Bodur M, Dash S, Günlük O, Luedtke J (2017) Strengthened Benders cuts for stochastic integer programs with continuous recourse. *INFORMS J. Comput.* 29(1):77–91.

Boland N, Christiansen J, Dandurand B, Eberhard A, Linderoth J, Luedtke J, Oliveira F (2018) Combining progressive hedging with a Frank–Wolfe method to compute Lagrangian dual bounds in stochastic mixed-integer programming. *SIAM J. Optim.* 28(2):1312–1336.

Carøe CC, Schultz R (1999) Dual decomposition in stochastic integer programming. *Oper. Res. Lett.* 24(1-2):37–45.

Chen B, Küçükyavuz S, Sen S (2011) Finite disjunctive programming characterizations for general mixed-integer linear programs. *Oper. Res.* 59(1):202–210.

Fischetti M, Salvagnin D, Zanette A (2010) A note on the selection of Benders' cuts. *Math. Programming* 124(1-2):175–182.

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

Guo G, Hackebeil G, Ryan SM, Watson JP, Woodruff DL (2015) Integration of progressive hedging and dual decomposition in stochastic integer programs. *Oper. Res. Lett.* 43(3):311–316.

Kall P, Wallace SW, Kall P (1994) *Stochastic Programming* (Springer, Berlin).

Kelley JE Jr (1960) The cutting-plane method for solving convex programs. *J. Soc. Industry Appl. Math.* 8(4):703–712.

Kim K, Dandurand B (2022) Scalable branching on dual decomposition of stochastic mixed-integer programming problems. *Math. Programming Comput.* 14(1):1–41.

Kim K, Zavala VM (2018) Algorithmic innovations and software for the dual decomposition method applied to stochastic mixed-integer programs. *Math. Programming Comput.* 10(2): 225–266.

Kim K, Petra CG, Zavala VM (2019) An asynchronous bundle-trust-region method for dual decomposition of stochastic mixed-integer programming. *SIAM J. Optim.* 29(1):318–342.

Laporte G, Louveaux FV (1993) The integer L-shaped method for stochastic integer programs with complete recourse. *Oper. Res. Lett.* 13(3):133–142.

Lemaréchal C, Nemirovskii A, Nesterov Y (1995) New variants of bundle methods. *Math. Programming* 69(1-3):111–147.

Li C, Grossmann IE (2018) An improved L-shaped method for two-stage convex 0–1 mixed integer nonlinear stochastic programs. *Comput. Chemical Engrg.* 112:165–179.

Lubin M, Martin K, Petra CG, Sandikçi B (2013) On parallelizing dual decomposition in stochastic integer programming. *Oper. Res. Lett.* 41(3):252–258.

Lulli G, Sen S (2004) A branch-and-price algorithm for multistage stochastic integer programming with application to stochastic batch-sizing problems. *Management Sci.* 50(6):786–796.

Ntaimo L (2013) Fenchel decomposition for stochastic mixed-integer programming. *J. Global Optim.* 55(1):141–163.

Ntaimo L, Sen S (2005) The million-variable "march" for stochastic combinatorial optimization. *J. Global Optim.* 32(3):385–400.

Pan F, Morton DP (2008) Minimizing a stochastic maximum-reliability path. *Networks* 52(3):111–119.

Qi Y, Sen S (2017) The ancestral Benders' cutting plane algorithm with multi-term disjunctions for mixed-integer recourse decisions in stochastic programming. *Math. Programming* 161(1-2): 193–235.

Rahmaniani R, Ahmed S, Crainic TG, Gendreau M, Rei W (2020) The Benders dual decomposition method. *Oper. Res.* 68:878–895.

Rockafellar RT (1970) *Convex Analysis*, vol. 36 (Princeton University Press, Princeton, NJ).

Schütz P, Tomasgard A, Ahmed S (2009) Supply chain design under uncertainty using sample average approximation and dual decomposition. *Eur. J. Oper. Res.* 199(2):409–419.

Sen S, Higle JL (2005) The *C*3 theorem and a *D*2 algorithm for large scale stochastic mixed-integer programming: Set convexification. *Math. Programming* 104(1):1–20.

Sen S, Sherali HD (2006) Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Math. Programming* 106(2):203–223.

Solak S, Clarke JPB, Johnson EL, Barnes ER (2010) Optimization of R&D project portfolios under endogenous uncertainty. *Eur. J. Oper. Res.* 207(1):420–433.

van der Laan N, Romeijnders W (2020) A converging Benders' decomposition algorithm for two-stage mixed-integer recourse models Accessed December 17, 2020, http://www.optimization-online.org/DB_HTML/2020/12/8165.html.

Van Slyke RM, Wets R (1969) L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM J. Appl. Math.* 17(4):638–663.

Zhang M, Küçükyavuz S (2014) Finitely convergent decomposition algorithms for two-stage stochastic pure integer programs. *SIAM J. Optim.* 24(4):1933–1951.

Zou J, Ahmed S, Sun XA (2019) Stochastic dual dynamic integer programming. *Math. Programming* 175(1-2):461–502.