Approximating the Lagrangian Dual of a Stochastic Integer Program via Fenchel Cuts

Rui Chen, James Luedtke

Department of Industrial and Systems Engineering, University of Wisconsin-Madison

Abstract

We present a decomposition method for solving large-scale stochastic integer programs. The proposed method is based on generation of Fenchel cuts, which combines the ideas of Benders decomposition and dual decomposition. We show that the strength of Fenchel cuts is equivalent to dual decomposition. We give practical approaches for generating strong Fenchel cuts by restricting the search space. Several computational enhancements have been proposed to improve the efficiency of our method. Computational results demonstrate that our method gives strong lower bounds at the root node and significantly reduces the size of the branch-and-cut tree for two classes of test problems.

1 Introduction

We introduce a new approach 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 decision makers have to make decisions before (first-stage decision) and after (second-stage decision) the revealing of the uncertain data with the objective to minimize the sum of the first-stage cost and the expected second-stage cost. Each realization of the uncertain data is called a scenario. Assuming a finite support of the uncertain data, a two-stage SIP can be formulated as follows:

$$\min_x \{c^T x + \sum_{s\in S} p_s Q_s(x) : Ax \geq b, x \in X\}, \tag{1}$$

where $c \in \mathbb{R}^n$, p_s denotes the probability of scenario s, $Q_s : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is the recourse function of scenario s with value

$$Q_s(x) = \begin{cases} \min\limits_y \{(q^s)^T y : \ W^s y \geq h^s - T^s x, y \in Y\}, & \text{if } Ax \geq b, x \in X, \\ +\infty, & \text{otherwise.} \end{cases} \tag{2}$$

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. Matrices W^s, T^s and vectors q^s, h^s are the uncertain data associated with scenario s.

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

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

From this perspective, SIPs are essentially large-scale integer programs (IPs) with special block structures. Directly solving (3) as an IP can be difficult when $|S|$ is large. So many studies of SIPs focus

on decomposition methods. Benders decomposition [1, 2] and dual decomposition [3, 4] are the two most commonly used decomposition methods for solving SIPs. Both methods solve a sequence of single scenario subproblems to generate cutting planes or lower bounds and have to be incorporated into a branch-and-bound framework to guarantee exact solution in general. 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 enforced to be equal to the master copy. These so-called nonanticipativity constraints are relaxed by Lagrangian relaxation. The Lagrangian dual function is then optimized by solving scenario IPs. A more detailed description of dual decomposition for SIPs is given in Section 2.2. In terms of strength, at the root node, the Benders decomposition approach converges to the LP relaxation bound of (3) while the Lagrangian dual bound obtained by dual decomposition converges to a bound equivalent to optimizing (3) with feasible region relaxed to be the intersection of its scenario integer hulls. Usually dual decomposition generates much stronger bounds than Benders decomposition but takes much longer to compute as it solves much harder subproblems. In this paper, we try to seek a place somewhere in between where we apply Fenchel cuts to the Benders model to approximate the strength of dual decomposition. Our work extends the work of Rahmaniani et al. [5] who proposed Benders Dual Decomposition with the same goal.

Apart from Benders decomposition and dual decomposition, some other interesting progress has been made in solving SIPs recently. We list a few related studies in stochastic integer programming here. Laporte and Louveaux [6] proposed integer L-shaped cuts for SIPs with pure binary first-stage variables. Lulli and Sen [7] developed a column generation based algorithm for solving multistage SIPs. Ntaimo [8] proposed the Fenchel decomposition decomposition approach for solving SIPs where Fenchel cuts are added in subproblems either in both first and second stages or in the second stage then lifted to the first stage. Bodur et al. [9] applied Benders decomposition with split cuts added to the scenario LP relaxation. Boland et al. [10] combined progressive hedging and the Frank-Wolfe method to calculate the Lagrangian dual bound. Zou et al. [11] applied strengthened Benders cuts and Lagrangian cuts to solve pure binary multistage SIPs. Li and Grossmann [12] developed a Benders-like decomposition algorithm implementing both Benders cuts and Lagrangian cuts for solving convex mixed 0-1 nonlinear stochastic programs. Rahmaniani et al. [5] applied strengthened Benders cuts and Lagrangian cuts to two-stage SIPs with continuous recourse. Although both use the term Fenchel cuts, Fenchel cuts in [8] are defined in the space of both first-stage and second-stage variables and added to scenario subproblems while Fenchel cuts in this paper are defined in the space of first-stage variables and second-stage cost variables and added to the master problem. In [11] and [5], the Lagrangian cuts derived from Lagrangian relaxation are essentially equivalent to an exact separation (13) of Fenchel cuts in this paper and the strengthened Benders cuts are special cases of Fenchel cuts.

The goal of this paper is to develop an effective way of generating Fenchel cuts that yield a strong lower bound while avoiding solving the hard exact separation problem. The main contributions of our work are summarized as follows.

1. We propose a normalization for generating Fenchel cuts similar to the one used in [13] for separating Benders cuts. This normalization can be used to construct a Fenchel cut separation problem different from the one (13) that generates Lagrangian cuts. It can also be applied to generate restricted Fenchel cuts.

2. We propose practical approaches for generating strong Fenchel cuts. We generate Fenchel cut by solving the cut generation problem in a restricted subspace. We also consider a mixed integer programming (MIP) approximation of the optimal subspace selection problem. Numerical results indicate that these approaches generate cuts that reduce integrality gaps quite efficiently.

3. We conduct an extensive numerical study on two classes of two-stage SIPs. We compare the impact of different parameter settings and the strength of different cut generation methods that

apply Fenchel cuts. Computational results are also given for using our method together with branch-and-cut as an exact solution method.

The paper is organized as follows. In Section 2, we define Fenchel cuts and illustrate its connection with dual decomposition. In Section 3, we propose a general framework to generate restricted Fenchel cuts. In Section 4, we present a numerical study on two types of SIP instances. We give concluding remarks in Section 5.

2 Preliminaries

2.1 Branch-and-cut based methods

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

$$\begin{aligned} \min_{x,\theta_s} \; & c^T x + \sum_{s\in S} p_s \theta_s \\ \text{s.t.} \; & \theta_s \geq Q_s(x), \qquad s \in S, \\ & Ax \geq b, \\ & x \in X. \end{aligned} \tag{4}$$

To solve (4), each recourse functions Q_s is replaced by a cutting-plane underestimate $\hat{Q}_s$ which creates a relaxation of (1) and transfers the problem to an IP with second-stage variables projected out. This cutting plane approximation is dynamically updated. In each iteration, the approximated problem is solved to obtain a candidate solution and a corresponding cut generation problem is then solved to update the cutting plane approximation. This process is repeated until convergence.

We introduce two types of valid inequalities for (4). The first collection of cuts is the famous Benders cuts [1, 2]. Benders cuts are often generated based on the LP relaxation of $Q_s(x)$. Given a candidate solution $\hat{x}$, we solve the LP relaxation of the recourse problem:

$$\begin{aligned} \max_{y} \; & (q^s)^T y \\ \text{s.t.} \; & W^s y \geq h^s - T^s \hat{x} \quad (\mu) \end{aligned}$$

Let μ be an optimal dual solution of the above problem. Based on LP duality, the following cut is a valid optimality cut:

$$\theta_s (\geq Q_s(x)) \geq -\mu^T T^s x + \mu^T h^s. \tag{5}$$

If the SIP has continuous recourse, i.e., $Y = \mathbb{R}^{n_y}$, then (5) is tight, i.e.,

$$Q_s(\hat{x}) = -\mu^T T^s \hat{x} + \mu^T h^s.$$

The cutting-plane model $\hat{Q}_s$ is often constructed by iteratively adding Benders cut until the lower bound converges to the LP relaxation bound. Benders cuts are sufficient to provide convergence for solving SIPs with continuous recourse using branch-and-cut.

Another useful family of cuts is the integer L-shaped cuts introduced in [6]. These cuts are valid only when the first-stage variables are binary, i.e., $X = \{0,1\}^n$. But integer L-shaped cuts do not require the second-stage problem to be an LP. In this case, given $\hat{x} \in \{0,1\}^n$, if we know a lower bound L_s on the recourse function value $Q_s(x)$ for all feasible x, then the following cut is a valid and tight optimality cut:

$$\theta_s (\geq Q_s(x)) \geq Q_s(\hat{x}) - (Q_s(\hat{x}) - L_s)\Big(\sum_{i:\hat{x}_i=1} (1 - x_i) + \sum_{i:\hat{x}_i=0} x_i \Big). \tag{6}$$

Integer L-shaped cuts often yield weak relaxations away from the points at which they are defined, so that a branch-and-cut algorithm using these cuts alone may require enumerating an impractically large number of branch-and-bound nodes.

Given the cutting-plane models $\hat{Q}_s$'s, a branch-and-cut algorithm using Benders cuts and integer L-shaped cuts is given in Algorithm 1.

Algorithm 1: Branch-and-cut for SIPs

Input: $\{\hat{Q}_s\}_{s\in S}$
Output: $(x^*, \{\theta_s^*\}_{s\in S})$
Initialize $\mathcal{L} \leftarrow \{0\}$, $\text{LP}_0 \leftarrow \min_{x,\theta_s}\{c^T x + \sum_{s\in S} p_s\theta_s : \theta_s \geq \hat{Q}_s(x), s \in S, Ax \geq b\}$, $\bar{z} \leftarrow +\infty$, $(x^*, \{\theta_s^*\}_{s\in S}) \leftarrow \emptyset$
while *Stopping condition not satisfied* **do**
  Choose a node $i \in \mathcal{L}$
  if *LP_i is feasible* **then**
    Solve LP_i to obtain optimal solution $(\hat{x}, \{\hat{\theta}_s\}_{s\in S})$ and optimal value $\hat{z}$
    if $\hat{z} < \bar{z}$ **then**
      if $\hat{x} \in X$ **then**
        for $s \in S$ **do**
          Evaluate $Q_s(\hat{x})$
          Add the Benders cut (5) to update $\hat{Q}_s$ if violated
          if $X = \{0,1\}^n$ **then**
            Add the integer L-shaped cut (6) to update $\hat{Q}_s$ if violated
          end
        end
        if $\hat{\theta}_s = Q_s(\hat{x})$ *for all* $s \in S$ **then**
          $\mathcal{L} \leftarrow \mathcal{L} \setminus \{i\}$
          $(x^*, \{\theta_s^*\}_{s\in S}) \leftarrow (\hat{x}, \{\hat{\theta}_s\}_{s\in S})$
          $\bar{z} \leftarrow \hat{z}$
        end
      else
        Branch on LP_i to construct $\text{LP}_{i_1}, \ldots, \text{LP}_{i_k}$ with feasible regions not containing $\hat{x}$
        $\mathcal{L} \leftarrow (\mathcal{L} \setminus \{i\}) \cup \{i_1, \ldots, i_k\}$
      end
    else
      $\mathcal{L} \leftarrow \mathcal{L} \setminus \{i\}$
    end
  end
end

In practice, Algorithm 1 can be implemented in a similar fashion using the LazyConstraint callback in modern IP solvers. Convergence is guaranteed in Algorithm 1 for SIPs with continuous recourse or pure binary first-stage variables. However, the efficiency of the algorithm can significantly depend 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 which results in a long solution time. The cutting-plane model $\hat{Q}_s$ is often initialized by Benders decomposition based on the LP relaxation of (3). Many cut generation methods (see, e.g., [9, 5]) were proposed to strengthen the model $\hat{Q}_s$ in order to accelerate the algorithm.

2.2 Dual decomposition

For two-stage SIPs, a strong lower bound can be essential for exact solution approaches. An approach for obtaining a strong bound is dual decomposition. In dual decomposition, a copy x^s of the first-stage variables x is created for each scenario which yields this nonanticipative reformulation of (1):

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

Then Lagrangian relaxation is applied to constraints $x = x^s, s \in S$ with multipliers $\lambda^s, s \in S$, which gives the following Lagrangian relaxation problem:

$$\begin{aligned} z(\lambda) = \min_{x,x^s,y^s} & \sum_{s\in S} p_s(c^T x^s + (q^s)^T y^s) + \sum_{s\in S} p_s(\lambda^s)^T(x^s - x), \\ \text{s.t.} \ & (x^s, y^s) \in K^s, \quad s \in S. \end{aligned} \tag{7}$$

Here $K^s = \{(x, y) : Ax \geq b, T^s x + W^s y \geq h^s, x \in X, y \in Y\}, s \in S$. We assume that K^s is nonempty and bounded for each $s \in S$ throughout the paper.

Note that (7) is a separable optimization problem given λ. And the Lagrangian dual problem $z_D = \max_\lambda z(\lambda)$ is a convex program which gives a lower bound on the optimal value of (1). Constraint $\sum_{s\in S} p_s\lambda^s = 0$ is implicitly enforced in $\max_\lambda z(\lambda)$ as $z(\lambda) = -\infty$ when $\sum_{s\in S} p_s\lambda^s \neq 0$. The following theorem gives a primal characterization of the bound z_D.

Theorem 1 (Carøe and Schultz [3])**.** *The following equality holds:*

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

Empirical evidence (e.g., [14, 15]) 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 to generate feasible solutions using heuristics (see, e.g., [3, 16]). However, the Lagrangian dual problem $z_D = \max_\lambda z(\lambda)$ can be hard to solve due to its large size (with dimension $n|S|$) and the difficulty of solving IP subproblems. We do not avoid solving IP subproblems in this paper but would like to decrease the size of the master problem so that we can avoid solving a huge number of integer subproblems to converge but still obtain a strong bound.

2.3 Conjugate and Fenchel cuts

We now investigate how cuts can be generated to the formulation in Section 2.1 to obtain a relaxation having the strength of dual decomposition, described in Section 2.2.

Fenchel duality (see, e.g., [17]) is a well-established tool from convex analysis. Given a function $f : \mathbb{R}^n \to \mathbb{R} \cup \{-\infty, +\infty\}$, the (convex) conjugate $f^* : \mathbb{R}^n \to \mathbb{R} \cup \{-\infty, +\infty\}$ of f is defined by

$$f^*(\pi) = \sup_x\{\pi^T x - f(x)\}.$$

Two useful properties of f^* are given below:

1. $\pi^T x \leq f(x) + f^*(\pi)$ for all x, π (Fenchel's inequality),

2. $\pi^T x = f(x) + f^*(\pi)$ if and only if $x \in \partial f^*(\pi)$.

For each scenario s, the conjugate Q_s^* of the recourse function Q_s is defined as

$$\begin{aligned} Q_s^*(\pi) &= \max_x \{\pi^T x - Q_s(x)\} \\ &= \max_{x,y} \; -(q^s)^T y + \pi^T x \\ &\quad \text{s.t. } Ax \geq b, \\ &\quad\quad T^s x + W^s y \geq h^s, \\ &\quad\quad x \in X, y \in Y. \end{aligned} \tag{8}$$

Note that Q_s^* is a piecewise linear convex function, and any optimal solution of (8) is a subgradient of Q_s^* at point π.

Based on Fenchel's inequality, for any $s \in S$, the following cut is valid for any $\pi \in \mathbb{R}^n$:

$$\theta_s (\geq Q_s(x)) \geq \pi^T x - Q_s^*(\pi).$$

We call a cut of this form a Fenchel cut in this paper. These Fenchel cuts are equivalent to the Lagrangian cuts in [11] and [5].

Based on the theory of conjugacy (see, e.g., [18]), Fenchel cuts can at most approximate Q_s^{**}, i.e., the pointwise maximum of all affine functions dominated by Q_s, which is equal to the lower semicontinuous convex envelope of Q_s. We show in the following theorem the strength of Fenchel cuts is equivalent to the relaxation used in dual decomposition.

Theorem 2. *The following equality holds:*

$$\min_{x,\theta_s} \{c^T x + \sum_{s\in S} p_s \theta_s : \theta_s \geq \pi^T x - Q_s^*(\pi) \text{ for all } \pi \in \mathbb{R}^n, s \in S\} \tag{9}$$

$$= \min_{x,y^s} \{c^T x + \sum_{s\in S} p_s (q^s)^T y^s : (x, y^s) \in \mathit{conv}(K^s), s \in S\}. \tag{10}$$

Proof. On the one hand, consider any $\bar{x}, \{\bar{y}^s\}_{s\in S}$ that satisfies

$$(\bar{x}, \bar{y}^s) \in \text{conv}(K^s), s \in S.$$

By definition of Q_s^*, for any $\pi \in \mathbb{R}^n, s \in S$, $\{\bar{\theta}_s := (q^s)^T \bar{y}^s\}_{s\in S}$ satisfies

$$\begin{aligned} \bar{\theta}_s - \pi^T \bar{x} =& (q^s)^T \bar{y}^s - \pi^T \bar{x} \\ \geq& \min_{x,y} \{(q^s)^T y^s - \pi^T x : (x, y^s) \in \text{conv}(K^s)\} \\ =& - Q_s^*(\pi). \end{aligned}$$

This implies that (9) $\leq$ (10).

On the other hand, note that (10) is equal to

$$\begin{aligned} &\min_{x,y^s,\theta_s} \{c^T x + \sum_{s\in S} p_s \theta_s : \theta_s \geq (q^s)^T y^s, (x, y^s) \in \text{conv}(K^s), s \in S\} \\ &\qquad = \min_{x,\theta_s} \{c^T x + \sum_{s\in S} p_s \theta_s : (x, \theta^s) \in U^s, s \in S\} \end{aligned} \tag{11}$$

where $U^s = \text{proj}_{(x,\theta_s)}\{(x, y^s, \theta_s) : \theta_s \geq (q^s)^T y^s, (x, y^s) \in \text{conv}(K^s)\}$. Consider now any $(\bar{x}, \bar{\theta}_s) \notin U^s$. Since K^s is a mixed integer linear set, we can assume $\text{conv}(K^s) = \{(x, y^s) : \bar{T}x + \bar{W}y^s \geq \bar{h}\}$ for some matrices $\bar{T}, \bar{W}$ and vector $\bar{h}$. Since $(\bar{x}, \bar{\theta}_s) \notin U^s$, there is no $\bar{y}^s$ such that

$$(q^s)^T \bar{y}^s \leq \bar{\theta}_s, \bar{W}\bar{y}^s \geq -\bar{T}\bar{x} + \bar{h}.$$

By the theorem of alternatives, there exists $u \geq 0, v \leq 0$ such that

$$q^s u + (\bar{W})^T v = 0,\ \bar{\theta}_s u + (-\bar{T}\bar{x} + \bar{h})^T v < 0.$$

We discuss the following two cases:

1. $u = 0$: Note that $v^T(-\bar{T}x + \bar{h}) = v^T(-\bar{T}x - \bar{W}y^s + \bar{h}) \geq 0$ is a valid inequality for $(x, y^s) \in \text{conv}(K^s)$ in this case. Thus for any $x \in \text{proj}_x(K^s)$, we have

   $$v^T\bar{T}(\bar{x} - x) \geq v^T\bar{T}\bar{x} - v^T\bar{h} > \bar{\theta}_s u = 0.$$

   Also note that K^s is compact. Then

   $$(\lambda\bar{T}^T v)^T\bar{x} - Q_s^*(\lambda\bar{T}^T\bar{v}) = \min_{x,y^s}\{(q^s)^T y + \lambda v^T\bar{T}(\bar{x} - x) : (x, y^s) \in K^s\} \to \infty \text{ as } \lambda \to \infty.$$

   Therefore, $(\bar{x}, \bar{\theta}_s)$ would violate the Fenchel cut

   $$\theta_s \geq (\lambda\bar{T}^T v)^T x - Q_s^*(\lambda\bar{T}^T v).$$

   for some large enough λ.

2. $u > 0$: In this case, we claim that $(\bar{x}, \bar{\theta}_s)$ violates the Fenchel cut

   $$\theta_s \geq (\bar{T}^T v/u)^T x - Q_s^*(\bar{T}^T v/u).$$

   Assume for contradiction that

   $$\bar{\theta}_s \geq (\bar{T}^T v/u)^T\bar{x} - Q_s^*(\bar{T}^T v/u) = (\bar{T}\bar{x})^T v/u - Q_s^*(\bar{T}^T v/u).$$

   Then since $\bar{\theta}_s u + (-\bar{T}\bar{x} + \bar{h})^T v < 0$,

   $$(\bar{T}\bar{x} - \bar{h})^T v > \bar{\theta}_s u \geq (\bar{T}\bar{x})^T v - uQ_s^*(\bar{T}^T v/u),$$

   i.e., $-\bar{h}^T v > -uQ_s^*(\bar{T}^T v/u)$. Note that since $q^s u + (\bar{W})^T v = 0$,

   $$\begin{aligned} -uQ_s^*(\bar{T}^T v/u) &= \min_{x,y}\{u(q^s)^T y - v^T\bar{T}x : \bar{T}x + \bar{W}y \geq \bar{h}\} \\ &= \max_{w}\{\bar{h}^T w : \bar{T}^T w = -\bar{T}^T v, \bar{W}^T w = -\bar{W}^T v, w \geq 0\}. \end{aligned} \tag{12}$$

   This produces a contradiction as $w = -v$ is feasible to (12), which implies that $-uQ_s^*(\bar{T}^T v/u) \geq -\bar{h}^T v$, but $-\bar{h}^T v > -uQ_s^*(\bar{T}^T v/u)$.

For both cases, $(\bar{x}, \bar{\theta}_s)$ can be cut off by some Fenchel cut. This implies that (10) = (11) $\leq$ (9). □

Directly solving (9) by adding all Fenchel cuts would be impractical but the separation from all Fenchel cuts can be formulated as a convex program. Given a candidate solution $(\hat{x}, \{\hat{\theta}_s\}_{s\in S})$, the separation problem for scenario s can be formulated as

$$\max_{\pi}\{\pi^T\hat{x} - Q_s^*(\pi)\} \tag{13}$$

where the conjugate function Q_s^* is convex but nonsmooth. The separation problem can be solved by bundle methods given an oracle of evaluating the function value and a subgradient of Q_s^*.

Evaluating the function value and a subgraident of Q_s^* at some point π can be accomplished by solving the scenario integer program:

$$\begin{aligned} \max_{x,y} \ & -(q^s)^T y + \pi^T x \\ \text{s.t. } & Ax \le b, \\ & T^s x + W^s y = h^s, \\ & x \in X, y \in Y. \end{aligned}$$

Given an optimal solution (x^*, y^*) of the above integer program, we have $Q_s^*(\pi) = -(q^s)^T y^* + \pi^T x^*$ and $x^* \in \partial Q_s^*(\pi)$.

However, even for medium-sized problems, applying bundle methods by solving integer programs as subproblems to solve (13) can be computationally inefficient as it requires solving potentially a long sequence of scenario integer programs to just separate one scenario solution $(\hat{x}, \hat{\theta}_s)$. Problem similar to (13) is proposed in [11] to generate the so-called Lagrangian cut while their computational results indicate that although Lagrangian cuts are very strong in general, adding these cuts would significantly increase the computation time and thus slow down the solution process. In [5], Rahmaniani et al. applied an inner approximation heuristic to (approximately) generate Lagrangian cuts.

In the next section, we propose solving a restricted version of (13). We restrict the searching space in order to accelerate the solution of the separation. Our approach is more computationally manageable while still yields cuts with good quality.

3 Restricted Separation of Fenchel Cuts

For each scenario s, the cut generated by exact separation (13) of Fenchel cuts is referred to as the Lagrangian cut in [11]. In general, the exact separation problem is hard to solve due to the difficulty of solving integer subproblems. However, given any $\pi \in \mathbb{R}^n$, evaluating $Q^*(\pi)$ gives a valid inequality

$$\theta_s \ge \pi^T x - Q_s^*(\pi). \tag{14}$$

By picking π properly, we show in the following proposition that very few simple Fenchel cuts can already give a reasonably good bound.

Proposition 3. *The following inequality hold:*

$$\begin{aligned} \min_x \{c^T x + \sum_{s\in S} p_s \theta_s : \theta_s \ge -c^T x - Q_s^*(-c), s \in S\} \\ \ge \sum_{s\in S} p_s \min_{x,y} \{c^T x + \sum_{s\in S} (q^s)^T y : (x,y) \in K^s, s \in S\} \end{aligned}$$

where the right-hand side is known as the perfect information bound [19].

Proof. The proof is straightforward by observing that

$$\begin{aligned} c^T x + \sum_{s\in S} p_s \theta_s \ge & c^T x + \sum_{s\in S} p_s(-c^T x - Q_s^*(-c)) \\ = & \sum_{s\in S} p_s(-Q_s^*(-c)) \\ = & \sum_{s\in S} p_s \min_{x,y} \{c^T x + \sum_{s\in S} (q^s)^T y : (x,y) \in K^s, s \in S\} \end{aligned}$$

for any solution $(x, \{\theta_s\}_{s\in S})$ feasible to the left-hand side. □

Motivated by this, we study restricted Fenchel cuts. We also argue computationally that with an appropriate basis, the strength of (9) can be well approximated by restricted Fenchel cuts.

3.1 General framework

We propose solving the following normalized and restricted version of (13) to separate a solution $(\hat{x}, \hat{\theta}_s)$:

$$\max_{\pi,\pi_0}\{-\pi_0\hat{\theta}_s + \pi^T\hat{x} - \bar{Q}_s^*(\pi,\pi_0) : \pi_0 \geq 0, (\pi,\pi_0) \in \Pi_s\} \tag{15}$$

where Π_s is a convex compact subset of $\mathbb{R}^{n+1}$ and $\bar{Q}_s^*$ is a piecewise linear convex function with

$$\begin{aligned}\bar{Q}_s^*(\pi,\pi_0) = \max_x \{\pi^T x - \pi_0 Q_s(x)\} = \max_{x,y}\ & \pi^T x - \pi_0 (q^s)^T y \\ \text{s.t. } & Ax \geq b, \\ & T^s x + W^s y \geq h^s, \\ & x \in X, y \in Y.\end{aligned} \tag{16}$$

Given a choice of coefficients (π^*, π_0^*), the following valid inequality can be added to the cutting-plane model:

$$-\pi_0^*\theta_s + (\pi^*)^T x - \bar{Q}_s^*(\pi^*,\pi_0^*) \leq 0. \tag{17}$$

Similar to (13), (15) is a convex program. But note that the objective function of (15) is positive homogeneous of degree 1 when $\pi_0 \geq 0$. If Π_s is a neighborhood of the origin, (e.g., $\Pi_s = \{(\pi,\pi_0) : |\pi_0| + \|\pi\|_1 \leq 1\}$,) then (15) exactly solves the Fenchel cut separation problem.

Proposition 4. *Assume Π_s is a neighborhood of the origin. Given any candidate solution $(\hat{x}, \hat{\theta}_s)$, if there exists $\bar{\pi}$ such that $\hat{\theta}_s < \bar{\pi}^T\hat{x} - Q_s^*(\bar{\pi})$, then*

$$\max_{\pi,\pi_0}\{-\pi_0\hat{\theta}_s + \pi^T\hat{x} - \bar{Q}_s^*(\pi,\pi_0) : \pi_0 \geq 0, (\pi,\pi_0) \in \Pi_s\} > 0.$$

Proof. Since Π_s is a neighborhood of 0, there exists $\rho > 0$ such that $(\tilde{\pi},\tilde{\pi}_0) := (\bar{\pi}/\rho, 1/\rho) \in \Pi_s$. And it follows that $\tilde{\pi}_0 \geq 0$ and

$$-\tilde{\pi}_0\hat{\theta}_s + \tilde{\pi}^T\hat{x} - \bar{Q}_s^*(\tilde{\pi},\tilde{\pi}_0) = [-\hat{\theta}_s + \bar{\pi}^T\hat{x} - Q_s^*(\bar{\pi})]/\rho > 0.$$

□

While setting Π_s to be a neighborhood of the origin would give an exact separation of Fenchel 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 (15) will heavily depend on the choice of Π_s. Based on our preliminary experiments, we find it effective to set Π_s to be a subset of the span of certain Benders cuts' coefficients. To be specific, we require

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

for some $\lambda \in \mathbb{R}^K$, where $\{\pi_k\}_{k=1}^K$ are coefficients of certain Benders cuts. If K is large enough and the Benders cuts span $\mathbb{R}^n$, then this approach reduces to exact separation of Fenchel cuts. However, for smaller K, we expect a trade-off between computation time and strength of cuts. The generation of Π_s is discussed in more detail in Section 3.2.2.

We call a solution x feasible to the SIP (1) if $Ax \geq b, x \in X$ and x has a feasible recourse solution y^s for each scenario s, i.e., $Q_s(x) < +\infty, s \in S$. Based on this definition of feasibility, we can classify (17) into optimality cuts and feasibility cuts.

Proposition 5. *Given any $\pi_0^* \geq 0, \pi^* \in \mathbb{R}^n$, (17) is a valid optimality cut if $\pi_0^* > 0$, and a valid feasibility cut if $\pi_0^* = 0$.*

Proof. Note that $\bar{Q}_s^*(\pi^*, \pi_0^*) = \pi_0^* Q_s^*(\pi^*/\pi_0^*)$ when $\pi_0^* > 0$. Therefore, if $\pi_0^* > 0$, (17) is equivalent to the Fenchel cut

$$\theta_s \geq (\pi^*/\pi_0^*)^T x - Q_s^*(\pi^*/\pi_0^*),$$

which is a valid optimality cut.

If $\pi_0^* = 0$, then (17) is equivalent to

$$\begin{aligned}(\pi^*)^T x \leq \bar{Q}_s^*(\pi^*, 0) = \max_{x,y} \ & (\pi^*)^T x \\ \text{s.t. } & Ax \geq b, \\ & T^s x + W^s y \geq h^s, \\ & x \in X, y \in Y,\end{aligned}$$

which is valid for any feasible solution x. □

It is worth mentioning that if $\{x : Ax \leq b\}$ is integral and (1) has relatively complete recourse, then any feasibility cut is redundant. And if $\text{span}(\{x : Ax \leq b\})$ is not full-dimensional and $Ax \leq b$ is enforced in the model, we only need to consider π that is perpendicular to the null space of $\{x : Ax \leq b\}$. This restriction may have practical benefits but we do not assume we have this information throughout the paper and our solution approach can be easily generalized to use this information.

Psuedocode for a cutting-plane algorithm which solves (15) to generate cuts is given in Algorithm 2. At each iteration t, we approximate the true objective function by the cutting-plane model:

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

where Ψ_s^t represents the past added Benders cuts and Fenchel cuts associated with scenario s at iteration t. The cutting-plane model is solved and generates candidate solution $(x^t, \{\theta_s^t\}_{s \in S})$. For each scenario s, the Fenchel cut separation problem (15) is solved with Π_s restricted to be a (typically low-dimensional) set Π_s^t. And the cutting-plane model is then updated by the Fenchel cut (if any) for all scenarios.

Algorithm 2: Restricted Fenchel Cut Separation

Input: $\{\hat{Q}_s^0\}_{s \in S}$
Initialize $t \leftarrow 0$
while *Stopping condition not satisfied* **do**
  Update $\{\hat{Q}_s^t\}_{s \in S}$ with Benders cuts (optional)
  Solve $x^t \in \arg\min_x \{c^T x + \sum_{s \in S} p_s \hat{Q}_s^t(x)\}$
  for $s \in S$ **do**
    $\theta_s^t \leftarrow \hat{Q}_s^t(x^t)$
  end
  for $s \in S$ **do**
    Generate $\Pi_s^t \subseteq \mathbb{R}^n$
    Solve (15) (approximately) with $(\hat{x}, \hat{\theta}_s) = (x^t, \theta_s^t)$ and $\Pi_s = \Pi_s^t$, and collect solution $(\pi, \pi_0) = (\pi^{t,s}, \pi_0^{t,s})$
    Update $\hat{Q}_s^t$ using cut (17) with $(\pi^*, \pi_0^*) = (\pi^{t,s}, \pi_0^{t,s})$ to obtain $\hat{Q}_s^{t+1}$
  end
  $t \leftarrow t + 1$
end

3.2 Computational enhancements

So far we have characterized a simple generic framework for adding cuts based on solving subproblems (15). We describe in this section some missing components and computational enhancements of the algorithm.

3.2.1 Solution of (15)

In Algorithm 2, we need to repeatedly solve problems of the form (15) with different $(\hat{\theta}_s, \hat{x})$. As $\bar{Q}_s^*(\pi, \pi_0)$ is a convex function of (π, π_0), problem (15) can be solved by a bundle method.

At iteration t, within the solution process of (15), the function $\bar{Q}_s^*$ can be approximated by a lower bounding cutting-plane model

$$\hat{Q}_{s,t}^*(\pi, \pi_0) = \max\{\pi^T z - \pi_0 \theta_s^z : (z, \theta_s^z) \in X^{s,t}\} \tag{18}$$

where $X^{s,t}$ is a set of stored feasible first-stage solutions z and second-stage costs θ_s^z satisfying $\theta_s^z \geq Q_s(z)$. In our implementation, for each scenario s, all feasible first-stage solutions obtained from evaluating $\bar{Q}_s^*(\pi, \pi_0)$ and the corresponding feasible second-stage costs are added into $X^{s,t}$. Specifically, each time we solve (16) for some (π, π_0), we collect all optimal and sub-optimal solutions (x, y) from the solver and store the first-stage solution $z = x$ and the feasible corresponding second-stage cost $\theta_s^z = (q^s)^T y$. Note that even though problem (15) is associated with different $\hat{x}, \hat{\theta}_s$ and Π_s at different iterations, $\hat{Q}_{s,t}^*$ remains a valid underestimate of $\bar{Q}_s^*$ at iteration $t+1$. Therefore, at each iteration $t+1$, $\hat{Q}_{s,t}^*$ can be used to initialize $\hat{Q}_{s,t+1}^*$, i.e., we can initialize $X^{s,t+1} = X^{s,t}$ at the beginning of iteration $t+1$. And in our implementation, to avoid unboundedness of $\hat{Q}_{s,0}^*$, $X^{s,0}$ is initialized by the perfect information solution $z^{s,0}$:

$$z^{s,0} \in \arg\min_x \{c^T x + Q_s(x)\}.$$

The basic version of the solution of (15) is described in Algorithm 3. The classical cutting-plane method used in the algorithm can be replaced by other bundle methods, e.g., the level method [20].

Algorithm 3: Solution of (15) at iteration t

Input: $(\hat{x}, \hat{\theta}_s)$, Π_s, $X^{s,t-1}$
Output: $(\pi^{t,s}, \pi_0^{t,s})$, $X^{s,t}$
Initialize $X^{s,t} \leftarrow X^{s,t-1}$, UB$\leftarrow +\infty$, LB$\leftarrow -\infty$
while *UB-LB*$\geq \delta\, UB$ **do**
  Solve $UB \leftarrow \max_{\pi,\pi_0}\{-\pi_0\hat{\theta}_s + \pi^T\hat{x} - \hat{Q}_{s,t}^*(\pi, \pi_0) : \pi_0 \geq 0, (\pi, \pi_0) \in \Pi_s\}$ to obtain solution $(\hat{\pi}, \hat{\pi}_0)$
  Evaluate $\bar{Q}_s^*(\hat{\pi}, \hat{\pi}_0)$ by solving (16), and update $X^{s,t}$ and $\hat{Q}_{s,t}^*$
  if $LB < -\hat{\pi}_0\hat{\theta}_s + \hat{\pi}^T\hat{x} - \bar{Q}_s^*(\hat{\pi}, \hat{\pi}_0)$ **then**
    $LB \leftarrow -\hat{\pi}_0\hat{\theta}_s + \hat{\pi}^T\hat{x} - \bar{Q}_s^*(\hat{\pi}, \hat{\pi}_0)$
  end
end

Note that if

$$-\pi_0^*\hat{\theta}_s + (\pi^*)^T\hat{x} - \bar{Q}_s^*(\pi^*, \pi_0^*) > 0,$$

then (17) is a valid inequality that cuts off the solution $(\hat{\theta}_s, \hat{x})$. Therefore, we solve (15) only to a δ relative tolerance. The value of δ plays the role of a trade-off between the depth of the cut and the running time of each iteration. For numerical reasons we also terminate if the upper bound is small enough ($< 10^{-6}(|\hat{\theta}_s| + 1)$) or two adjacent multipliers are too close to each other (infinity norm of the difference $< 10^{-10}$) in our implementation.

Let $P_s(\Pi_s)$ denote the feasible set defined by all normalized Fenchel cuts restricted on Π_s:

$$P_s(\Pi_s) = \{(x, \theta_s) : -\pi_0\theta_s + \pi^T x - \bar{Q}_s^*(\pi, \pi_0) \leq 0 \text{ for any } (\pi, \pi_0) \in \Pi_s \text{ with } \pi_0 \geq 0\}.$$

Note that $P_s(\Pi_s)$ is a polyhedron if Π_s is a polytope since $\bar{Q}_s^*$ is a polyhedral function. Also since $\bar{Q}_s^*$ is positive homogeneous, $P_s(\Pi_s) = P_s(C_{\Pi_s})$ where C_{Π_s} is the conic hull of Π_s. We show in the next theorem that if we use a static compact Π_s and keep adding Fenchel cuts by solving (15) to δ optimality with $\delta \in [0, 1)$, the algorithm will converge to solutions that satisfy all restricted Fenchel cuts.

Theorem 6. *Assume that at each iteration of Algorithm 2, for each scenario s a Fenchel cut (if a violated one exists) is generated by solving (15) to δ optimality with $\delta \in [0, 1)$ and $\Pi_s^t \equiv \Pi_s$ is a compact set independent of t. 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 $(\bar{x}, \bar{\theta}_s) \in P_s(\Pi_s)$.*

Proof. Note that $\bar{Q}_s^*$ is a piecewise linear convex function. Therefore, $\bar{Q}_s^*$ is Lipshitz 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

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

Since Π_s is bounded, there exists τ such that

$$\max_{\pi,\pi_0}\{|-\pi_0(\theta_s^{i_t} - \bar{\theta}_s) + \pi^T(x^{i_t} - \bar{x})| : (\pi, \pi_0) \in \Pi_s, \pi_0 \geq 0\} \leq \epsilon/2 \text{ for all } t \geq \tau.$$

This implies that

$$\max_{\pi,\pi_0}\{-\pi_0\theta_s^{i_t} + \pi^T x^{i_t} - \bar{Q}_s^*(\pi, \pi_0) : (\pi, \pi_0) \in \Pi_s, \pi_0 \geq 0\} \geq \epsilon/2 \text{ for all } t \geq \tau.$$

Let $(\pi^{(i_t)}, \pi_0^{(i_t)})$ denote the multiplier associated with the Fenchel cut added at iteration i_t. For any $t' > t \geq \tau$,

$$-\pi_0^{(i_t)}\theta_s^{i_{t'}} + (\pi^{(i_t)})^T x^{i_{t'}} - \bar{Q}_s^*(\pi^{(i_t)}, \pi_0^{(i_t)}) \leq 0$$

since the Fenchel cut associated with $(\pi^{(i_t)}, \pi_0^{(i_t)})$ was added before iteration $i_{t'}$. One the other hand,

$$\begin{aligned}
&-\pi_0^{(i_{t'})}\theta_s^{i_{t'}} + (\pi^{(i_{t'})})^T x^{i_{t'}} - \bar{Q}_s^*(\pi^{(i_{t'})}, \pi_0^{(i_{t'})}) \\
\geq &(1-\delta)\max_{\pi,\pi_0}\{-\pi_0\theta_s^{i_{t'}} + \pi^T x^{i_{t'}} - \bar{Q}_s^*(\pi, \pi_0) : (\pi, \pi_0) \in \Pi_s, \pi_0 \geq 0\} \geq (1-\delta)\epsilon/2.
\end{aligned}$$

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

In Section 4.3.1, we experiment on the effect of the parameter δ.

Another computational issue that might occur when solving (15) is that typically we would solve the subproblem (16) as an integer program. If $\pi_0 = 0$, the solution of (16) may give an incorrect information about $Q_s(x^*)$ since the coefficients associated with the y variables are equal 0. In that case, if we store the "optimal" second-stage cost $(q^s)^T y^*$ and use it together with x^* to define a cut in the cutting-plane model (18), this second-stage cost value $(q^s)^T y^*$ can be very far away from $Q_s(x^*)$ and lead to a very loose cut. So in our implementation, when π_0 is small ($< 10^{-4}$), given x^* being the optimal solution of $\max_x\{\pi^T x - \pi_0 Q_s(x)\}$, we reevaluate $Q_s(x^*)$ by solving the scenario subproblem (2) with $x = x^*$ fixed and use this value as $\theta_s^{x^*}$.

3.2.2 Choice of Π_s

As discussed in Section 3.2.1, one of the major differences between the exact separation (13) and our proposed problem (15) is the restriction constraint $(\pi, \pi_0) \in \Pi_s$. The following two properties are desired for a good choice of Π_s:

1. Π_s should be chosen so that problem (15) is easy to solve, in particular avoiding evaluating the function $\bar{Q}_s^*$ too many times.
2. Π_s should be chosen so that cut (17) has a similar strength compared with the Fenchel cut corresponding to the optimal solution of (13).

To satisfy these two properties, the default setting of Π_s in this paper is defined by the span of past Benders cuts with some normalization. To be specific,

$$\Pi_s = \{(\pi, \pi_0) : \exists \lambda \in \mathbb{R}^K \text{ s.t. } \pi = \sum_{k=1}^{K} \lambda_k \pi^k, \alpha|\pi_0| + \|\pi\|_1 \leq 1\}. \tag{19}$$

Here K and α are both predetermined parameters, and $\{\pi_k\}_{k=1}^K$ are the last K Benders cuts' coefficients. Preliminary experiments indicate that the choice of $\{\pi_k\}_{k=1}^K$ is important. For this choice of $\{\pi_k\}_{k=1}^K$, the hope is that 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 in [13].

Our second proposal is to use

$$\Pi_s = \{(\pi, \pi_0) : \exists \lambda \in \mathbb{R}^K \text{ s.t. } \pi = \sum_{k=1}^{K} \lambda_k \pi^k, \alpha|\pi_0| + \|\lambda\|_1 \leq 1\}. \tag{20}$$

This normalization imposes a constraint on $\|\lambda\|_1$ which may prefer to choose a sparse λ.

Next, we consider choosing $\{\pi_k\}_{k=1}^K$ from a set of candidate vectors. We propose using (20) with an MIP approximation for optimally selecting $\{\pi_k\}_{k=1}^K$ from a scenario coefficient pool $V^s = \{v^k\}_{k \in I_s}$ which consists of finitely many vectors. Using (20), the optimal selection of $\{\pi_k\}_{k=1}^K$ which maximizes the normalized violation can be determined by the following convex integer program:

$$\max_{\mu_k, z_k, \pi, \pi_0} \quad -\pi_0 \hat{\theta}_s + \pi^T \hat{x} - \bar{Q}_s^*(\pi, \pi_0) \tag{21}$$

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

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

$$-z_k \leq \mu_k \leq z_k, \qquad k \in I_s, \tag{24}$$

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

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

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

The above problem is computationally challenging due to the implicit function $\bar{Q}_s^*$. However, if we replace $\bar{Q}_s^*$ by its cutting-plane approximation $\hat{Q}_{s,t}^*$, then the problem can be solved as an integer linear program. Specifically, the MIP approximation of (21)-(27) at iteration t is given by:

$$\begin{aligned} \max_{\mu_k, z_k, \pi, \pi_0, \tau} \quad & -\pi_0 \hat{\theta}_s + \pi^T \hat{x} - \tau \\ \text{s.t.} \quad & \tau \geq \pi^T z - \pi_0 \theta_s^z, \qquad (z, \theta_s^z) \in X^{s,t}, \\ & (22) - (27). \end{aligned} \tag{28}$$

Vectors v^k with $z_k = 1$ would be chosen to formulate the basis. The basis size in this case can be strictly smaller than K. Additionally the optimal value of (28) gives an upper bound of (21)-(27). Therefore, if the optimal objective value of (28) is too small, we can skip this scenario for generating Fenchel cuts.

Compared with (20), (19) does not have an integer program approximation for the optimal coefficients selection without the use of a big-M formulation, although (20) could be sensitive to the scaling of π^k. In (19) and (20), we call $\{\pi_k\}_{k=1}^K$ the basis of Π_s and K the basis size of Π_s. In Section 4.3.2, we study the relationship between the basis size K and the strength of generated Fenchel cuts based on normalization (19) or (20).

4 Computational Study

We conduct numerical experiments to study the computational performance of different cut generation strategies. We consider in total six implementations of Fenchel cuts:

1. *StrBen*: Strengthened Benders cuts from [11], i.e., (14) with multiplier π equal to the Benders cut coefficients each time a Benders cut is generated.
2. *BDD*: BDD_3 from [5]. This algorithm uses a multiphase implementation that first generates Benders cuts and strengthened Benders cuts, and then generates Fenchel cuts based on a regularized inner approximation.
3. *Exact*: Exact separation of Fenchel cuts of the form (15) with $\Pi_s = \{(\pi, \pi_0) : \alpha\pi_0 + \|\pi\|_1 \leq 1\}$.
4. *Rstr1*: Restricted separation of Fenchel cuts (15) with Π_s defined in (19) and $\{\pi_k\}_{k=1}^K$ being the last K Benders cuts' coefficients.
5. *Rstr2*: Restricted separation of Fenchel cuts (15) with Π_s defined in (20) and $\{\pi_k\}_{k=1}^K$ being the last K Benders cuts' coefficients.
6. *RstrMIP*: Restricted separation of Fenchel cuts (15) with Π_s defined in (20) and $\{\pi_k\}_{k=1}^K$ determined by solving the integer program approximation (28) with $V^s = \{v^k\}_{k \in I_s}$ being the set of all Benders' cuts coefficients that have been generated.

4.1 Test problems

We test our algorithm on two types of problems, namely the stochastic server location problem (SSLP) introduced in [21], and the stochastic network interdiction problem (SNIP) from [22].

The SSLP problem is a two-stage SIP with pure binary first-stage and mixed-binary second-stage variables. All SSLP instances are named of the form sslp$k$-m-n-$|S|$ where m denotes the number of sites, n denotes number of clients, $|S|$ is the number of scenarios and k is the instance number. In this problem, the decision maker has to choose from m sites to allocate servers with some cost in the first stage. Then in the second stage, the availability of each client i would be observed and every available client must be served at some site. The first-stage variables are denoted by x with $x_j = 1$ if and only if a server is located at site j. The second-stage variables are denoted by y and y_0 where $y_{ij}^s = 1$ if and only if client i is served at site j in scenario s and y_{0j}^s is the amount of resource shortage at at site j in scenario s. Allocating a server costs c_j at site j. Each allocated server can provide up to u units of resource. Client i uses d_{ij} units of resource and generate revenue q_{ij} if served at site j. Each unit of resource shortage at site j incurs a penalty q_{0j}. The client availability in scenario s is represented by h^s with $h_i^s = 1$ if and only if client i is present in scenario s. The extensive formulation of the problem

is as follows:

$$
\begin{aligned}
\min_{x,y^s,y_0^s} \ & \sum_{j=1}^{m} c_j x_j + \sum_{s\in S} p_s \Big(\sum_{j=1}^{m} q_{0j} y_{0j}^s - \sum_{i=1}^{n}\sum_{j=1}^{m} q_{ij} y_{ij}^s\Big) & \\
\text{s.t.} \ & \sum_{i=1}^{n} d_{ij} y_{ij}^s - y_{0j}^s \le u x_j, & j=1,\dots,m,\ s\in S,\\
& \sum_{j=1}^{m} y_{ij}^s = h_i^s, & i=1,\dots,n,\ s\in S,\\
& x_j \in \{0,1\}, & j=1,\dots,m,\\
& y_{ij}^s \in \{0,1\}, & i=1,\dots,n,\ j=1,\dots,m,\ s\in S,\\
& y_{0j}^s \ge 0, & j=1,\dots,m,\ s\in S.
\end{aligned}
$$

All SSLP instances are generated with $(m,n) \in \{(20,100),(30,70),(40,50),(50,40)\}$ and $|S| \in \{50,200\}$. For each size $(m,n,|S|)$, three instances are generated with $k \in \{1,2,3\}$. The instances we tested on have 20/30/40/50 first-stage binary variables, roughly 2000 second-stage mixed-binary (most binary) variables per scenario, and either 50 or 200 scenarios. Details of the data generation for the instances can be found in Appendix.

The SNIP problem is a two-stage SIP with pure binary first-stage and continuous second-stage variables. In this problem the defender interdicts arcs on a directed network by installing sensors in the first stage to maximize the probability of finding an attacker. Then in the second stage, giving a random origin u^s and a random destination v^s, the attacker travels along the maximum reliability path from u^s to v^s in scenario s. Let N and A denote the node set and the arc set of the network and let $D \subseteq A$ denote the set of interdictable arcs. The first-stage variables are denoted by x with $x_a = 1$ if and only if the defender installs a sensor on arc $a \in D$. The second-stage variables are denoted by π where π_i^s denotes the maximum probability of reaching destination v^s undetected from node i in scenario s. The budget for installing sensors is b, and the cost of installing a sensor on arc a is c_a for each arc $a \in D$. For each arc $a \in A$, the probability of traveling on arc a undetected is r_a if the arc is not interdicted, or q_a if the arc is interdicted. Finally, let u_j^s denote the maximum probability of reaching the destination undetected from node j when no sensors are installed. The extensive formulation of the problem is as follows:

$$
\begin{aligned}
\min_{x,\pi^s} \ & \sum_{s\in S} p_s \pi_{u^s}^s & \\
\text{s.t.} \ & \sum_{a\in A} c_a x_a \le b, & \\
& \pi_i^s - r_a \pi_j^s \ge 0, & a=(i,j)\in A\setminus D,\ s\in S,\\
& \pi_i^s - r_a \pi_j^s \ge -(r_a - q_a) u_j^s x_a, & a=(i,j)\in D,\ s\in S,\\
& \pi_i^s - q_a \pi_j^s \ge 0, & a=(i,j)\in D,\ s\in S,\\
& \pi_{v^s}^s = 1, & s\in S,\\
& x_a \in \{0,1\}, & a\in D.
\end{aligned}
$$

All of the SNIP instances we used in this paper are from [22]. We consider instances with snipno = 3, and budget $b \in \{30,50,70,90\}$. All instances have 320 first-stage binary variables, 2586 second-stage continuous variables per scenario and 456 scenarios.

4.2 Implementation details

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

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. Except for the initialization, the Benders cuts are obtained via L-shaped method by the classical cutting-plane method (Kelley's algorithm [23]) for SNIP instances, and via L-shaped method with regularization by the level method [20] for SSLP instances. These Benders cuts are added if the violation is at least $10^{-4}(|\hat{\theta}_s| + 1)$. Since we find the QP solver not stable enough when subspace restrictions are added to the problem, all convex programs (15) are solved by Kelley's algorithm, except for the exact separation of Fenchel cuts which is solved by the level method. The normalization coefficient α in (19) or (20) is set to be 1 for tests on SSLP instances, and 0.1 for tests on SNIP instances. In the root node experiments, the stopping condition for all tests is that either the algorithm hits the 1 hour time limit or there is no cut that can be found by the algorithm (with a 10^{-6} relative tolerance).

When implementing strengthened Benders cuts (see Section 4.3), similar Benders decomposition is applied first to obtain the LP relaxation bound. Then strengthened Benders cuts are added based on the optimal solution obtained from the current cutting-plane model.

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

For the *BDD* tests on SNIP and SSLP instances, we basically follow the implementation of [5] except that we scale the regularization coefficient $\delta_0 = 0.01$ to $\delta_0 = 100$ for SNIP instances which leads to better bounds. For *BDD* tests on SSLP instances, we still use the default $\delta_0 = 0.01$.

We have also tested our instances using the general purpose SIP solver DSP [24, 25, 16]. 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 IP solver.

4.3 Tests with the root node

We first compare results obtained from different implementations of Fenchel cuts on SSLP and SNIP test instances and study the impact of different parameters in the algorithm.

On certain instances, we present the convergence profiles representing the increase of the lower bound over time using different methods or different parameter settings. To compare the overall performance of the methods visually, we also 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 at the root node by all 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 $t^\gamma_{p,m}$ being 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 $t^\gamma_{p,m} = \infty$. The γ-gap-closed profile is a figure representing the cumulative growth (distribution function) of the γ-gap-closed ratio $\rho^\gamma_m(\tau)$'s over time τ where

$$\rho^\gamma_m(\tau) = \frac{|\{p \in P : t^\gamma_{p,m} \leq \tau\}|}{|P|}.$$

In this paper, P is either the set of all SSLP test instances or all SNIP test 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 at the root node. And we set parameter γ equal to either 0.75 or 0.95 for summarizing the information regarding time needed for obtaining a reasonably good bound ($\gamma = 0.75$) or a strong bound ($\gamma = 0.95$).

4.3.1 Relative gap tolerance δ

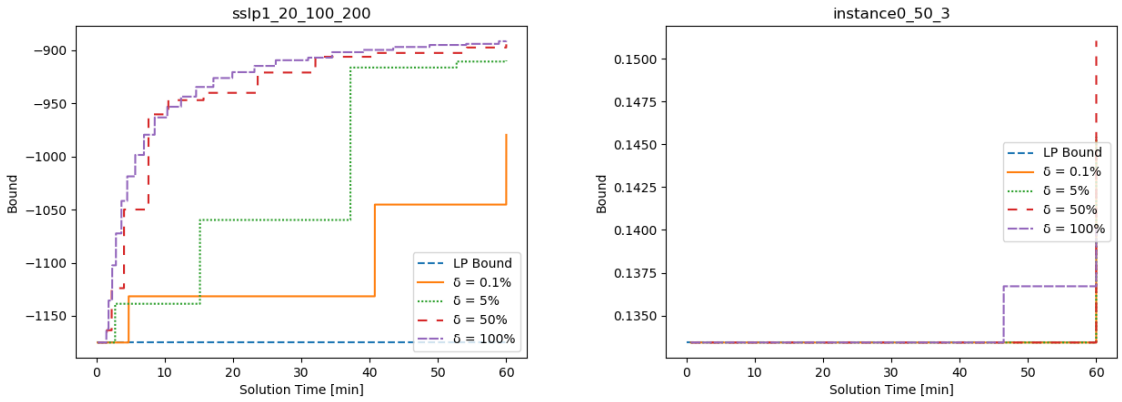

Figure 1: Convergence profile of *Exact* on an SSLP instance (left) and an SNIP instance (right) with varying δ values

We present numerical results with varying relative gap tolerance δ. Experiments are run on SSLP and SNIP instances with parameter δ being $0.1\%, 5\%, 50\%$ and 100%, respectively. A small δ value (e.g., 0.1%) means solving (15) to almost optimality, which may generate strong cuts but can be time-consuming. Using large δ values might sacrifice some strength of the cut 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 SNIP instance (instance0_50_3). For SNIP instances, due to its high dimensions of first-stage variables, *Exact* often cannot finish one iteration of separation within an hour. For SSLP instances, a large δ value usually leads to faster separation, which helps close the gap faster, compared with a small δ value. Plots providing the same comparison for *Rstr1* with $K = 10$ are provided in Appendix.

We plot the 0.75-gap-closed profile and 0.95-gap-closed profile with different delta values in Figure 2 and 3. The gap-closed profiles for SNIP instances with *Exact* are omitted as no useful bound is obtained by *Exact* for any SNIP instance. For SSLP instances, it is preferable to use a large δ value as both a reasonable bound and a strong bound are obtained earlier when δ is large. For SNIP instances, there is less clear preference whereas small δ values, especially $\delta = 0.1\%$, perform slightly better than large delta values. This is because the time spent on each iteration of separation does not vary too much by changing the δ value for SNIP instances.

The optimal choice of δ can generally depend on the instance. But we find a relatively large δ, e.g., $\delta = 50\%$, has a stable performance on most instances.

4.3.2 Basis size K

We next investigate the impact of the choice of the basis size K in *Rstr1*, *Rstr2* and *RstrMIP*. In Figure 4 and 5, we plot 0.75-gap-closed profile and 0.95-gap-closed profile with different K values for SSLP instances and SNIP instances, respectively. Convergence profiles for *Rstr1*, *Rstr2* and *RstrMIP* with different K values on one SNIP instance and one SSLP instance can be found in Appendix.

As seen in Figure 4 and 5, the trends are quite different for SSLP instances and SNIP instances when applying *Rstr1*. For SSLP instances, smaller K performs better for obtaining a reasonably good bound faster since the restricted separation is much easier to solve when K is small. However, for SNIP instances, as the difference in the solution time of one restricted separation is not significant

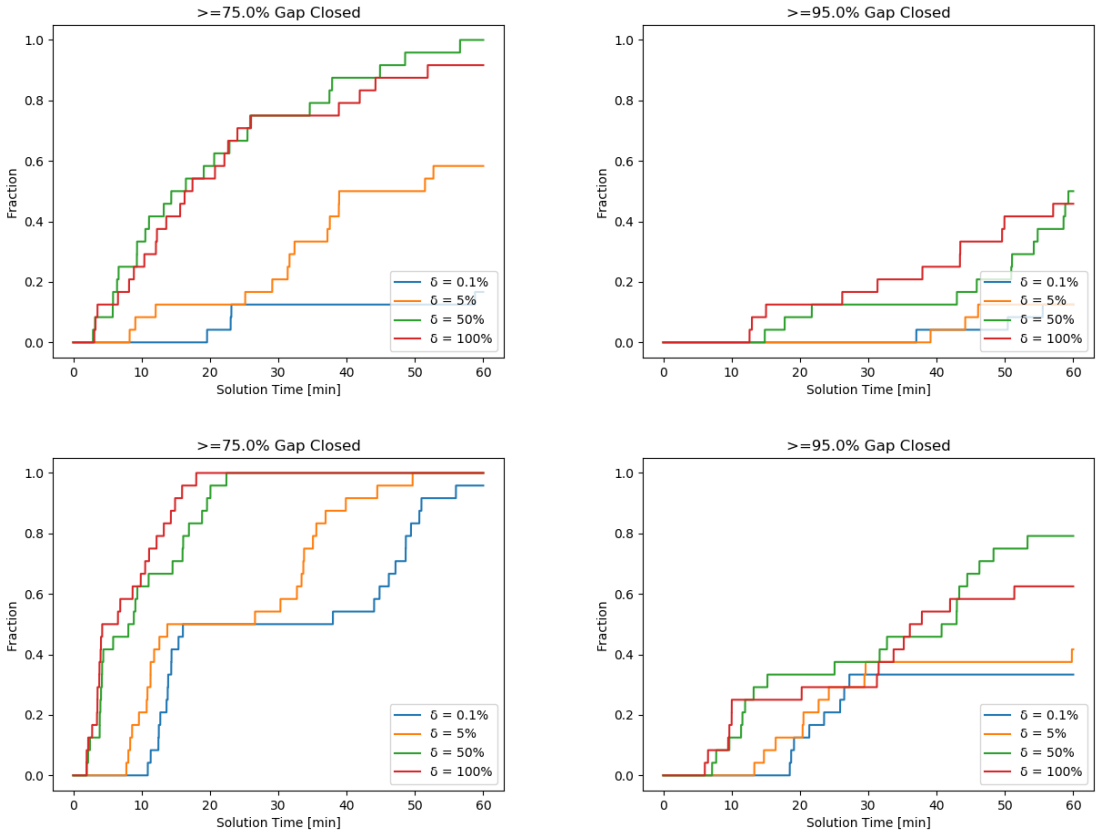

Figure 2: γ-gap-closed profile for SSLP instances obtained by *Exact* (top) and *Rstr1* (bottom) with $\gamma = 0.75$ (left) or $\gamma = 0.95$ (right) and varying δ values

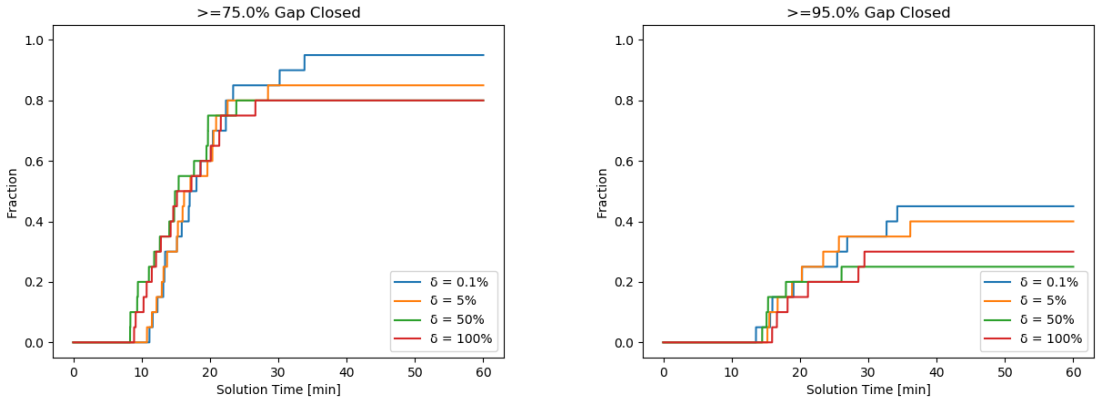

Figure 3: γ-gap-closed profile for SSLP instances obtained by *Rstr1* with $\gamma = 0.75$ (left) or $\gamma = 0.95$ (right) and varying δ values

while $K = 20$ gives much stronger cuts for closing the gap, $K = 20$ seems to dominate $K = 5$ and

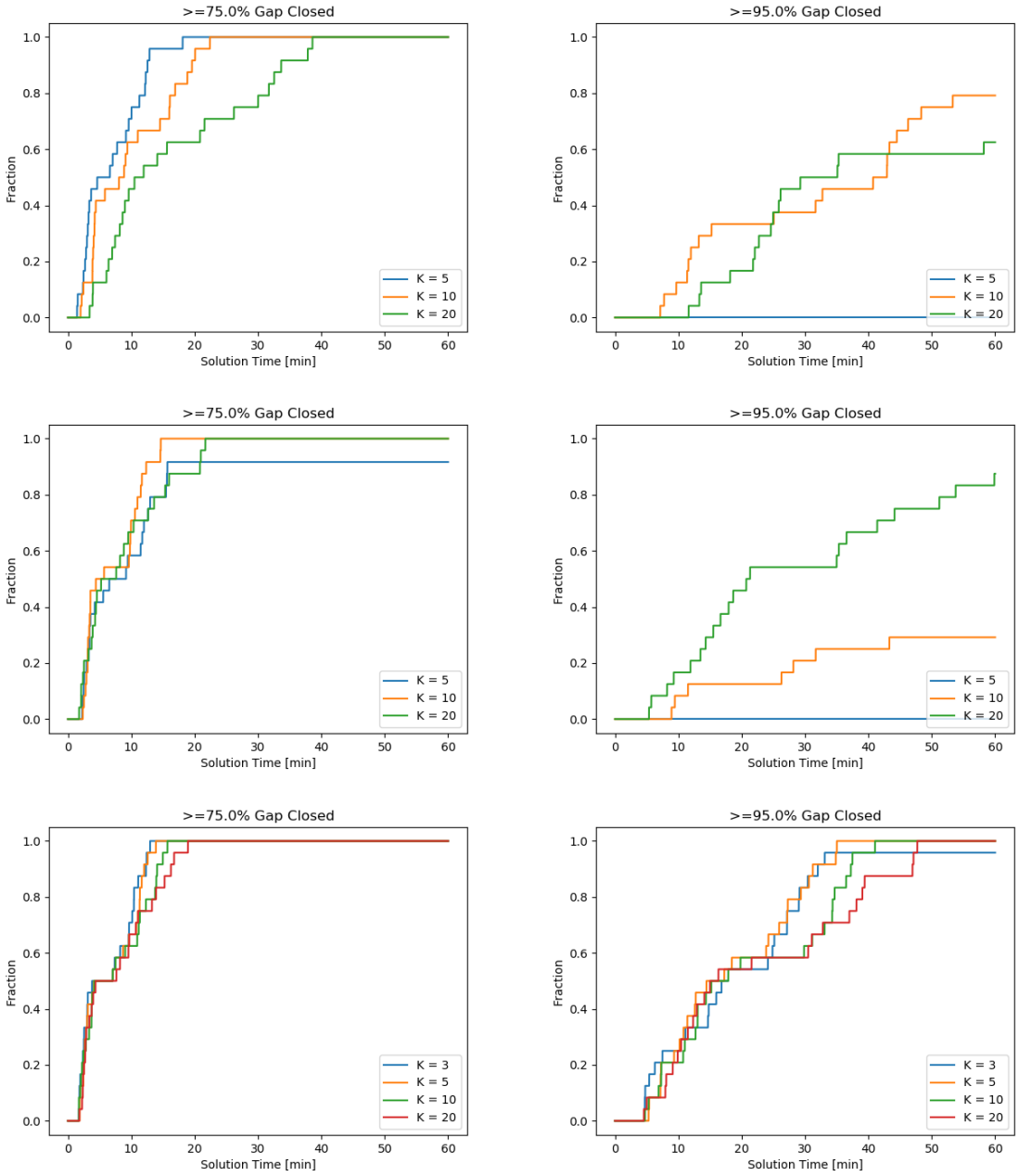

Figure 4: γ-gap-closed profile for SSLP instances obtained by *Rstr1* (top), *Rstr2* (middle) and *RstrMIP* (bottom) with $\gamma = 0.75$ (left) or $\gamma = 0.95$ (right) and varying K values

$K = 10$. Comparing *Rstr1* and *Rstr2*, *Rstr2* appears to be less sensitive to K in terms of obtaining a reasonably good bound for SSLP instances. On the other hand, the performance of *Rstr2* is almost identical to the performance of *Rstr1* for SNIP instances. For SSLP instances with *Rstr1* or *Rstr2*, we find that larger K often leads to longer solution time for obtaining a reasonable bound while small K ($K = 5$) is incapable of getting a strong bound. We observe that *RstrMIP* is fairly robust to the choice

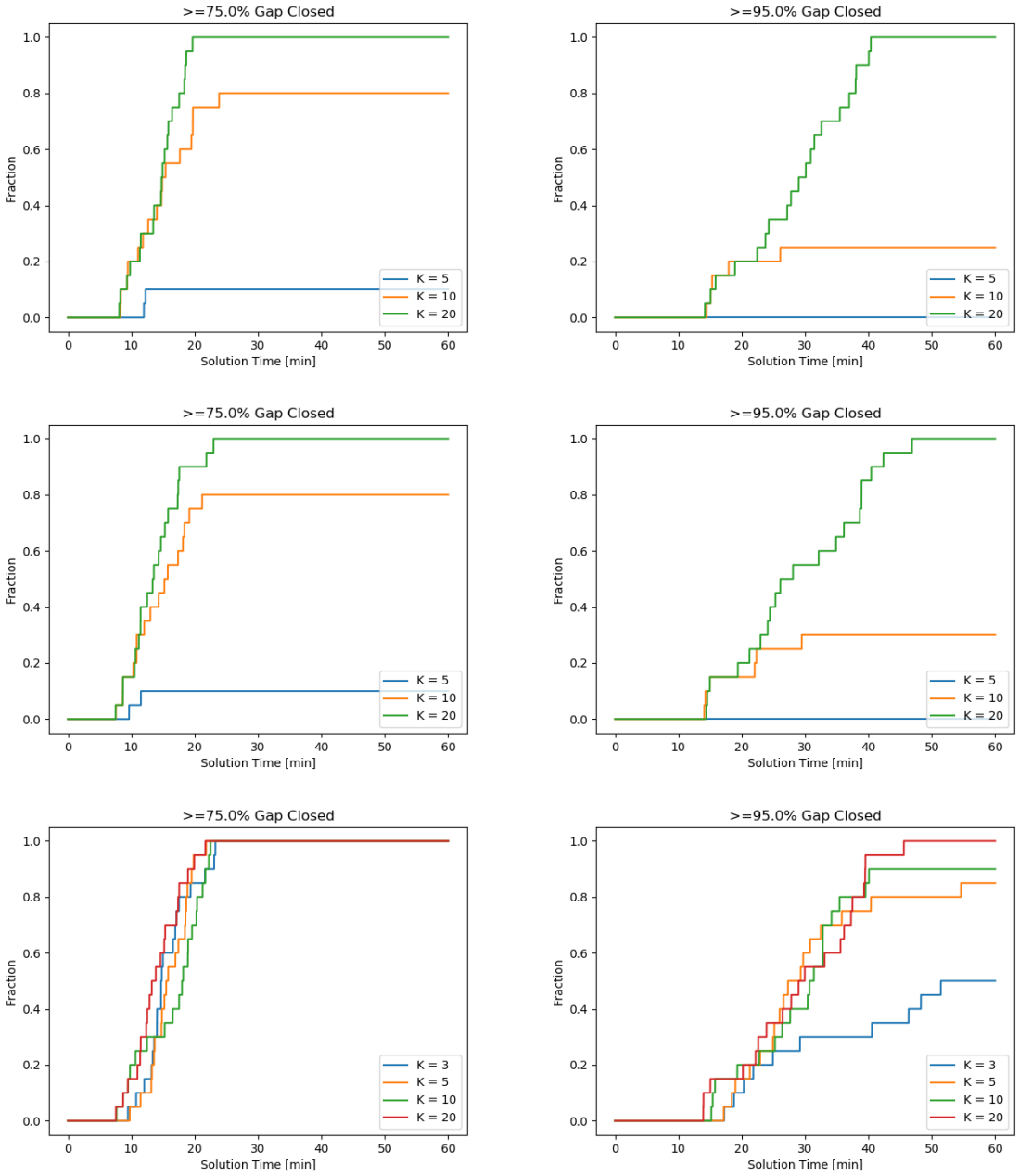

Figure 5: γ-gap-closed profile for SNIP instances obtained by *Rstr1* (top), *Rstr2* (middle) and *RstrMIP* (bottom) with $\gamma = 0.75$ (left) or $\gamma = 0.95$ (right) and varying K values

of the basis size K for both SSLP and SNIP instances. One particular reason is that the constraint $\sum_{k \in I_s} z_k \leq K$ in (28) is often loose in early stages of the algorithm for relatively large K's as the constraint $\alpha\pi_0 + \|\mu\|_1 \leq 1$ tends to select a sparse μ and thus a sparse z. For the same instance, different basis sizes result in very close bounds at the end of the run.

Basis size K yields trade-off 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* while the performance of *RstrMIP* is robust to the choice of K.

4.3.3 Comparison between methods

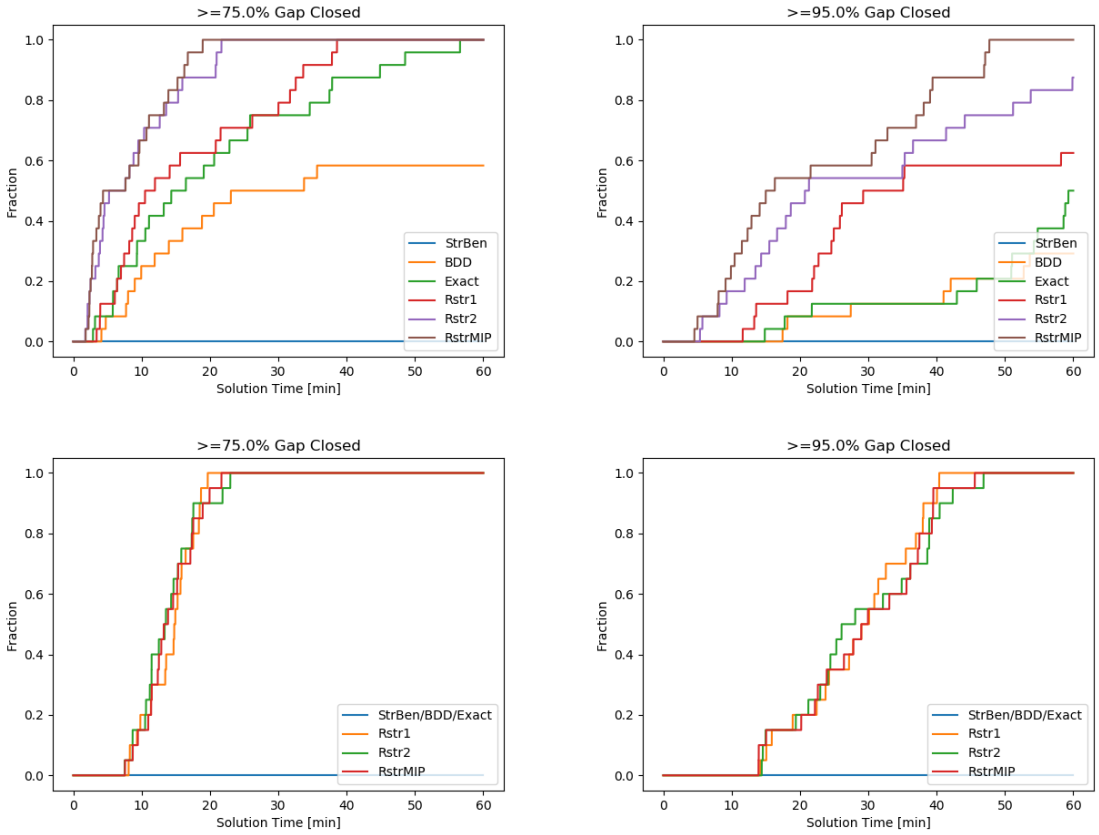

Figure 6: γ-gap-closed profile for SSLP instances (top) and SNIP instances (bottom) with $\gamma = 0.75$ (left) or $\gamma = 0.95$ (right)

We compare results obtained by strengthened Benders cut (*StrBen*), the Benders dual decomposition (*BDD*) and different variants of our restricted Fenchel cut separation (*Exact*, *Rstr1*, *Rstr2* and *RstrMIP*). For all variants of restricted Fenchel 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 profile and 0.95-gap-closed profile for these methods in Figure 6.

Apparently, *RstrMIP* has the best performance on SSLP instances in terms of the gap closed over time. On SNIP instances, the performances of *Rstr1*, *Rstr2* and *RstrMIP* are almost identical while *RstrMIP* is much more robust in the choice of K as observed in Section 4.3.2. 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 and one SNIP instance can be found in Appendix. Since our methods provide stronger bounds than *BDD* in shorter time, we do not test branch-and-cut on top of *BDD*.

4.4 Tests for solving to optimality

Finally, we study Fenchel cuts integrated with branch-and-cut (FBC) to solve SIPs to optimality, where the branch-and-cut algorithm is described in Algorithm 1 with Benders cuts and integer L-shaped cuts added using Gurobi callback when a new MIP incumbent is found. We consider only *RstrMIP* with $\delta = 50\%$ and $K = 10$ as the implementation of Fenchel cuts. To reduce the time spent on Fenchel cut generation at the root node, an early termination rule is applied. We stop generating Fenchel cuts if the gap closed in the last 5 iterations is less than 1% of the total gap closed so far. For comparison, we also experiment with Benders cuts integrated with branch-and-cut (BBC) and the DSP solver [24] on the same instances. A two-hour time limit is set for all branch-and-cut 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.) For each instance class with different sizes, we report the number of instances solved and average solution time for each algorithm, and average gap, average branch-and-cut time and average number of nodes explored for FBC and BBC. Here for each instance, the gap 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 branch-and-cut algorithm at the end of each run.

Table 1: Number of solved instances and average solution time obtained by different methods for SSLP instances

$(m, n, \lvert S\rvert)$	# solved			Avg soln time		
	FBC	BBC	DSP	FBC	BBC	DSP
(20,100, 50)	3/3	0/3	3/3	**852**	≥7200	974
(20,100,200)	3/3	0/3	3/3	**2806**	≥7200	8757
(30, 70, 50)	3/3	0/3	3/3	1506	≥7200	**1133**
(30, 70,200)	3/3	0/3	3/3	**4288**	≥7200	7531
(40, 50, 50)	3/3	0/3	3/3	**1689**	≥7200	2189
(40, 50,200)	3/3	0/3	3/3	**5518**	≥7200	14676
(50, 40, 50)	3/3	0/3	3/3	1937	≥7200	**738**
(50, 40,200)	3/3	0/3	3/3	4780	≥7200	**4414**

We compare the results obtained by FBC, BBC and DSP on SSLP instances in Table 1 and Table 2. All SSLP instances are solved to optimality by FBC within the two-hour time limit with at most a few thousand nodes explored. On the other hand, none of the SSLP instances are solved by BBC before hitting the time limit, leaving a 10% to 35% gap. SSLP instances are hard to solve by BBC since neither Benders cuts or 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, FBC solves most instances faster than DSP. The performance of FBC is relatively stable compared with DSP in the sense that FBC 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 FBC on average solution time, no branching is needed for closing the dual gap.

In Table 3 and Table 4, we report results obtained from SNIP instances. We observe that no SNIP instance can be solved within two hours by DSP. This is because solving the nonanticipative dual problem of SNIP instances is too hard. BBC takes significantly shorter time to solve SNIP instances than FBC as the time spent of cut generation at the root node is much shorter.

It is observed in [9] that the IP solver cuts have a large impact on SNIP instances which can lead to a much smaller root gap. To study the real strength of the cuts and its impact on branch-and-cut,

Table 2: Average gap, branch-and-cut time, number of nodes obtained by FBC and BBC for SSLP instances

$(m, n, \|S\|)$	Avg gap (%)		Avg B&C time		Avg # nodes	
	FBC	BBC	FBC	BBC	FBC	BBC
(20,100, 50)	0.0	10.5	12	≥7200	207.3	≥12350.0
(20,100,200)	0.0	15.5	28	≥7200	325.0	≥ 4451.0
(30, 70, 50)	0.0	18.1	17	≥7200	476.7	≥13328.0
(30, 70,200)	0.0	25.1	215	≥7189	1853.7	≥ 4007.3
(40, 50, 50)	0.0	22.8	64	≥7200	2241.3	≥14493.7
(40, 50,200)	0.0	33.3	391	≥7183	3352.3	≥ 3439.0
(50, 40, 50)	0.0	21.2	6	≥7200	175.0	≥15060.0
(50, 40,200)	0.0	34.3	17	≥7200	112.7	≥ 3917.3

Table 3: Number of solved instances and average solution time obtained by different methods for SNIP instances

b	# solved			Avg soln time		
	FBC	BBC	DSP	FBC	BBC	DSP
30	5/5	5/5	0/5	1316	**98**	≥7200
50	5/5	5/5	0/5	2665	**162**	≥7200
70	5/5	5/5	0/5	2485	**164**	≥7200
90	5/5	5/5	0/5	2865	**230**	≥7200

Table 4: Average gap, branch-and-cut time, number of nodes obtained by FBC and BBC for SNIP instances

b	Avg gap (%)		Avg B&C time		Avg # nodes	
	FBC	BBC	FBC	BBC	FBC	BBC
30	0.0	0.0	34	65	509.0	1582.0
50	0.0	0.0	70	117	1106.2	3473.0
70	0.0	0.0	65	112	1259.4	2635.6
90	0.0	0.0	115	171	2047.8	5700.0

we also report results obtained by FBC and BBC with presolving and Gurobi cuts turned off. It turns out that presolving and Gurobi cuts are quite essential for solving SNIP instances when using BBC. The efficiency of BBC depends a lot on presolving and Gurobi cuts generated at the root node. For the harder SNIP instances with budget $b \geq 70$, turning off presolving and Gurobi cuts for BBC can make originally easily solvable instances unsolvable. In these cases, presolving and Gurobi cuts significantly improves the LP bounds for BBC at the root node which is essential for limiting the size of the branch-and-cut tree. FBC is much more robust to this difference and solves all the SNIP instances in both

Table 5: Comparison of FBC and BBC for SNIP instances with presolving and Gurobi cuts turned off

b	# solved		Avg soln time		Avg gap (%)		Avg B&C time		Avg # nodes	
	FBC	BBC	FBC	BBC	FBC	BBC	FBC	BBC	FBC	BBC
30	5/5	5/5	1323	**390**	0.0	0.0	41	357	1094.2	510760.6
50	5/5	4/5	**2694**	≥3535	0.0	0.6	99	≥3490	5262.4	≥3182018.8
70	5/5	0/5	**2529**	≥7200	0.0	2.2	108	≥7148	2953.2	≥4723076.8
90	5/5	0/5	**2992**	≥7200	0.0	8.9	242	≥7141	7635.8	≥3279800.8

cases with little difference in solution time. This can be explained by the fact that the LP bound is already strong enough after applying Fenchel cuts at the root node.

5 Conclusion

We have developed a generic approach for adding Fenchel cuts to solve two-stage stochastic integer programs. Extensive numerical experiments were conducted on two classes of SIP instances to study the performance of the proposed method with varying parameters and computational enhancements. We also compared our method with the open-source SIP solver DSP. Our method solves one problem class that is unsolvable to DSP and performs modestly better on the other problem class, demonstrating the robustness of our approach.

Empirically we found that using Benders cuts' coefficients as the basis for restricted separation of Fenchel cuts is promising for obtaining strong lower bounds. However, this heuristic choice of the basis does not necessarily guarantee to obtain the strongest theoretical bound, i.e., the nonanticipative Lagrangian dual bound. We tried a few heuristics for augmenting the basis, but did not find any that improve the performance nontrivially. Therefore, one direction for future work could be understanding the strength of different basis choices and developing basis generating methods for obtaining better bounds at the root node.

Appendix

Generation of SSLP instances

For each SSLP instance, the data are generated as:

- $p_s = 1/|S|$, $s \in S$.
- $c_j \sim$Uniform$(\{40, 41, \ldots, 80\})$, $j = 1, \ldots, m$.
- $d_{ij} = q_{ij} \sim$Uniform$(\{0, 1, \ldots, 25\})$, $i = 1, \ldots, n$, $j = 1, \ldots, m$.
- $q_{0j} = 1000$, $j = 1, \ldots, m$.
- $u = \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij}/m$.
- $h_i^s \sim$Uniform$(\{0, 1\})$, $i = 1, \ldots, n$, $s \in S$.

Some convergence profiles

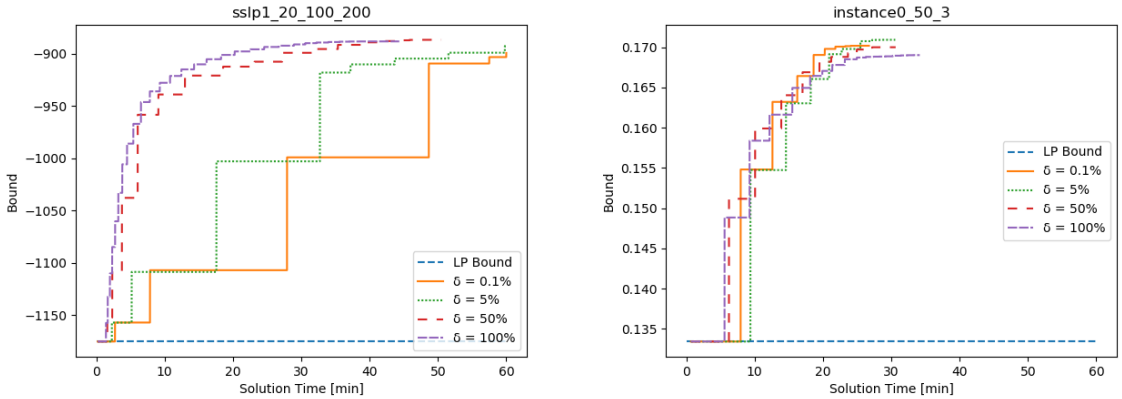

Figure 7: Convergence profile of *Rstr1* on an SSLP instance (left) and an SNIP instance (right) with varying δ values

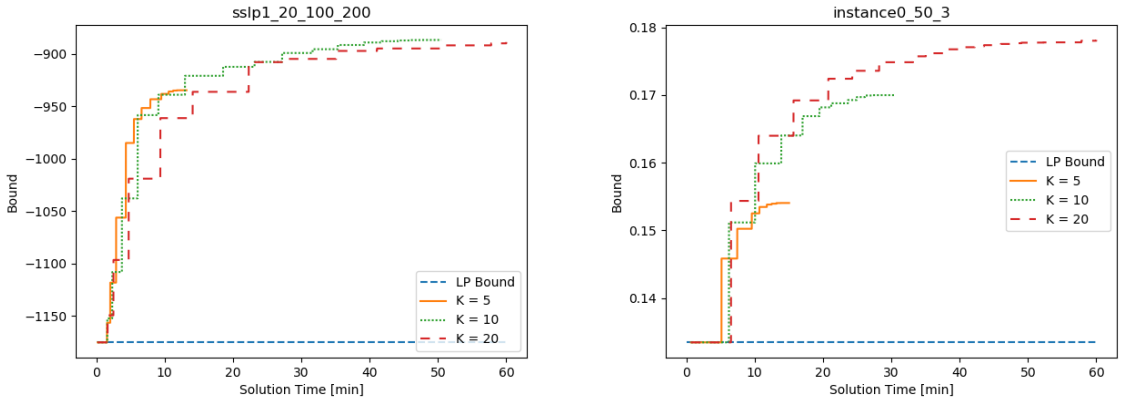

Figure 8: Convergence profile of *Rstr1* on an SSLP instance (left) and an SNIP instance (right) with varying K values

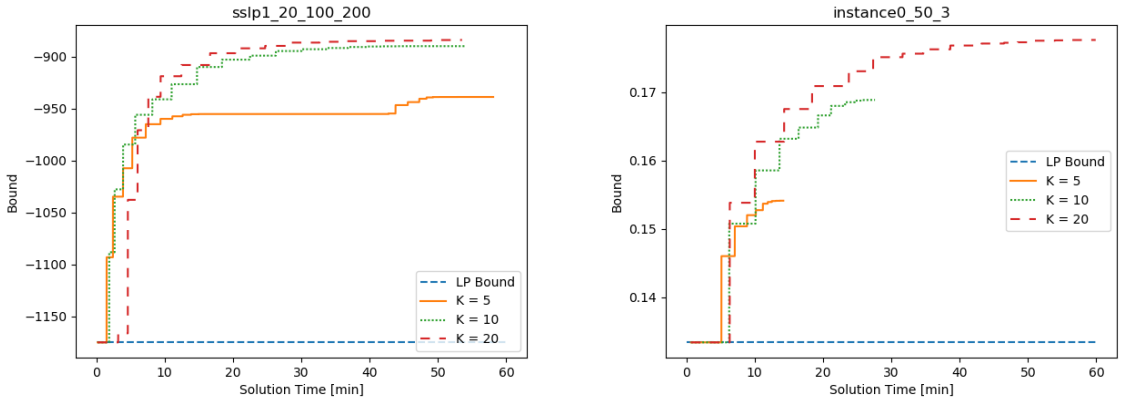

Figure 9: Convergence profile of *Rstr2* on an SSLP instance (left) and an SNIP instance (right) with varying K values

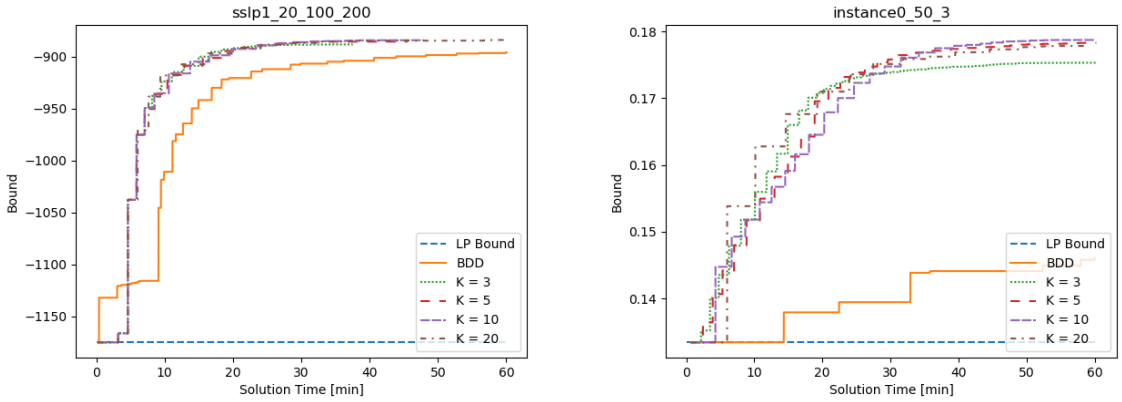

Figure 10: Convergence profile of *RstrMIP* on an SSLP instance (left) and an SNIP instance (right) with varying K values and comparison with *BDD*

For the SSLP instance, we observe that *BDD* does not make much progress in the first 10 minutes. This is because *BDD* is in the phase of generating strengthened Benders cuts which are not strong enough to substantially improve the bound except for the first iteration. However, skipping the strengthened Benders cut phase would kill the algorithm as this phase is essential for building an inner approximation of the exact separation in the next phase.

References

[1] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, pp. 238–252, 1962.

[2] R. M. Van Slyke and R. Wets, "L-shaped linear programs with applications to optimal control and stochastic programming," *SIAM Journal on Applied Mathematics*, vol. 17, no. 4, pp. 638–663, 1969.

[3] C. C. Carøe and R. Schultz, "Dual decomposition in stochastic integer programming," *Operations Research Letters*, vol. 24, no. 1-2, pp. 37–45, 1999.

[4] M. Lubin, K. Martin, C. G. Petra, and B. Sandıkçı, "On parallelizing dual decomposition in stochastic integer programming," *Operations Research Letters*, vol. 41, no. 3, pp. 252–258, 2013.

[5] R. Rahmaniani, S. Ahmed, T. G. Crainic, M. Gendreau, and W. Rei, "The benders dual decomposition method," *Operations Research*, 2020.

[6] G. Laporte and F. V. Louveaux, "The integer l-shaped method for stochastic integer programs with complete recourse," *Operations research letters*, vol. 13, no. 3, pp. 133–142, 1993.

[7] G. Lulli and S. Sen, "A branch-and-price algorithm for multistage stochastic integer programming with application to stochastic batch-sizing problems," *Management Science*, vol. 50, no. 6, pp. 786–796, 2004.

[8] L. Ntaimo, "Fenchel decomposition for stochastic mixed-integer programming," *Journal of Global Optimization*, vol. 55, no. 1, pp. 141–163, 2013.

[9] M. Bodur, S. Dash, O. Günlük, and J. Luedtke, "Strengthened benders cuts for stochastic integer programs with continuous recourse," *INFORMS Journal on Computing*, vol. 29, no. 1, pp. 77–91, 2017.

[10] N. Boland, J. Christiansen, B. Dandurand, A. Eberhard, J. Linderoth, J. Luedtke, and F. Oliveira, "Combining progressive hedging with a frank–wolfe method to compute lagrangian dual bounds in stochastic mixed-integer programming," *SIAM Journal on Optimization*, vol. 28, no. 2, pp. 1312–1336, 2018.

[11] J. Zou, S. Ahmed, and X. A. Sun, "Stochastic dual dynamic integer programming," *Mathematical Programming*, vol. 175, no. 1-2, pp. 461–502, 2019.

[12] C. Li and I. E. Grossmann, "An improved l-shaped method for two-stage convex 0–1 mixed integer nonlinear stochastic programs," *Computers & Chemical Engineering*, vol. 112, pp. 165–179, 2018.

[13] M. Fischetti, D. Salvagnin, and A. Zanette, "A note on the selection of benders' cuts," *Mathematical Programming*, vol. 124, no. 1-2, pp. 175–182, 2010.

[14] P. Schütz, A. Tomasgard, and S. Ahmed, "Supply chain design under uncertainty using sample average approximation and dual decomposition," *European Journal of Operational Research*, vol. 199, no. 2, pp. 409–419, 2009.

[15] S. Solak, J.-P. B. Clarke, E. L. Johnson, and E. R. Barnes, "Optimization of r&d project portfolios under endogenous uncertainty," *European Journal of Operational Research*, vol. 207, no. 1, pp. 420–433, 2010.

[16] K. Kim and B. Dandurand, "Scalable branching on dual decomposition of stochastic mixed-integer programming problems." Preprint on webpage at `http://www.optimization-online.org/DB_HTML/2018/10/6867.html`.

[17] R. T. Rockafellar, *Convex analysis*. No. 28, Princeton university press, 1970.

[18] R. T. Rockafellar and R. J.-B. Wets, *Variational analysis*, vol. 317. Springer Science & Business Media, 2009.

[19] M. Avriel and A. Williams, "The value of information and stochastic programming," *Operations Research*, vol. 18, no. 5, pp. 947–954, 1970.

[20] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov, "New variants of bundle methods," *Mathematical programming*, vol. 69, no. 1-3, pp. 111–147, 1995.

[21] L. Ntaimo and S. Sen, "The million-variable "march" for stochastic combinatorial optimization," *Journal of Global Optimization*, vol. 32, no. 3, pp. 385–400, 2005.

[22] F. Pan and D. P. Morton, "Minimizing a stochastic maximum-reliability path," *Networks: An International Journal*, vol. 52, no. 3, pp. 111–119, 2008.

[23] J. E. Kelley, Jr, "The cutting-plane method for solving convex programs," *Journal of the society for Industrial and Applied Mathematics*, vol. 8, no. 4, pp. 703–712, 1960.

[24] K. Kim and V. M. Zavala, "Algorithmic innovations and software for the dual decomposition method applied to stochastic mixed-integer programs," *Mathematical Programming Computation*, vol. 10, no. 2, pp. 225–266, 2018.

[25] K. Kim, C. G. Petra, and V. M. Zavala, "An asynchronous bundle-trust-region method for dual decomposition of stochastic mixed-integer programming," *SIAM Journal on Optimization*, vol. 29, no. 1, pp. 318–342, 2019.

[26] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical programming*, vol. 91, no. 2, pp. 201–213, 2002.