A generalized Benders decomposition-based branch and cut algorithm for two-stage stochastic programs with nonconvex constraints and mixed-binary first and second stage variables

Can Li[a], Ignacio E. Grossmann[a,*]

[a]*Center for Advanced Process Decision-Making, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, United States*

Abstract

In this paper, we propose a generalized Benders decomposition-based branch and cut algorithm for solving two stage stochastic mixed-integer nonlinear programs (SMINLPs). At a high level, the proposed decomposition algorithm performs spatial branch and bound search on the first stage variables. Each node in the branch and bound search is solved with a Benders-like decomposition algorithm where both Lagrangean cuts and Benders cuts are included in the Benders master problem. The Lagrangean cuts are derived from Lagrangean decomposition. The Benders cuts are derived from the Benders subproblems, which are convexified by cutting planes, such as rank-one lift-and-project cuts. We prove that the proposed algorithm has convergence in the limit. We apply the proposed algorithm to a stochastic pooling problem, a crude selection problem, and a storage design problem. The performance of the proposed algorithm is compared with a Lagrangean decomposition-based branch and bound algorithm and solving the corresponding deterministic equivalent with the solvers including BARON, ANTIGONE, and SCIP.

Keywords: Stochastic programming; Nonconvex MINLP; Generalized Benders Decomposition; Cutting Planes

*Corresponding author.

E-mail address: grossmann@cmu.edu (I.E. Grossmann).

1 Introduction

Two-stage stochastic mixed-integer programs (SMIPs) is a framework to model decision-making uncertainty that involves discrete decisions. Traditional L-shaped method [41] can only be applied to linear SMIPs with continuous recourse, but cannot be applied to solve SMIPs with (mixed)-integer recourse. Applications for SMIPs with integer recourse include stochastic server location problem [28], dynamic capacity acquisition and allocation [3], stochastic unit commitment problem [35], etc. In the past two decades, there have been algorithmic developments in solving linear SMIPs with integer recourse. The pioneering work is done by Laporte and Louveaux [18] who propose integer cuts for two-stage SMIPs with pure binary first stage variables. Benders decomposition-based parametric cutting plane algorithms have also been proposed where the parametric cuts, such as disjunctive cuts [5, 33, 27, 29, 36], RLT [39, 38], Gomory mixed-integer cuts [12, 43], are used to convexify the (mixed)-integer subproblems. Carøe and Schultz [11] propose a dual decomposition-based branch and bound algorithm. For a review of recent advances in SMIPs, we refer to the tutorial by Küçükyavuz and Sen [17].

Although there have been algorithmic advances in linear SMIPs, the decomposition algorithms to address the nonlinear counterpart, stochastic mixed-integer nonlinear programs (SMINLPs), are few. For convex SMINLPs (the nonlinear functions in SMINLPs are all convex), Mijangos [25] proposes an algorithm based on Branch-and-Fix Coordination method [4] for convex problems with 0-1 mixed-integer variables in the first stage and only continuous variables in the second stage. Atakan and Sen [7] propose a progressive hedging-based branch-and-bound algorithm for convex SMINLP, which works well in the case of pure binary first stage variables and continuous recourse. Li and Grossmann [20] propose an improved L-shaped method where the Benders subproblems are convexified by rank-one lift-and-project and Lagrangean cuts are added to tighten the Benders master problem. Since the improved L-shaped method does not guarantee zero gap, Li and Grossmann propose a generalized Benders decomposition-based branch and bound (GBDBAB) algorithm [19] with finite ϵ-convergence for convex SMINLPs with mixed-binary first and second stage variables.

For nonconvex SMINLPs (the nonlinear functions in SMINLPs can be nonconvex), the nonconvexities in stage 2 are two-fold: first, we have nonlinear nonconvex functions in stage 2; second, some of the stage 2 variables need to satisfy integrality constraints. Li et al. [23] propose a nonconvex generalized Benders decomposition (NGBD) algorithm for problems with pure binary first stage variables where they relax the nonconvex functions in stage 2 by their convex relaxations in order to derive valid Benders-like cuts. The convergence of NGBD relies on using the integer cuts similar to the one proposed by Laporte and Louveaux [18]. NGBD and its variations have been applied to stochastic pooling problems [21, 24], integrated process design and operation problems [22], etc.

However, if the first stage variables are mixed-integer, the integer cuts used in Li et al. [23] are no longer applicable. For the more general case where the first stage variables can be mixed-integer, Ogbe and Li [30] propose a joint decomposition algorithm where they reformulate the two-stage problem so that all the nonconvexities are in the first stage. Therefore, the Benders subproblems in [30] are continuous and convex. The authors also add Lagrangean cuts to the Benders master problem to tighten the master problem. The advantage of [30] is that explicit spatial branch and bound can be avoided. However, a large-scale master problem may need to be solved by global solvers when the number of scenarios is large. Cao and Zavala [10] propose a perfect information-based branch and bound algorithm that solves nonconvex SMINLPs to global optimality. The

authors relax the nonanticipativity constraints (NACs) of the two stage problem to derive a valid bound at each node of the spatial branch and bound process where they only need to branch on the first stage variables. Kannan and Barton [15] propose a modified Lagrangean relaxation-based (MLR) branch and bound algorithm, which is similar to the algorithm proposed by Carøe and Schultz [11] except that MLR only dualizes the NACs corresponding to the continuous first stage variables. Therefore, the Lagrangean subproblems still have NACs corresponding to integer variables, which are solved by NGBD proposed by Li et al. [23]. The authors prove that MLR has finite ϵ-convergence.

This paper follows the lines of the work of Cao and Zavala [10] and Kannan and Barton [15]. We propose a new algorithm to solve nonconvex SMINLPs with mixed-binary variables in both first and second stage.

2 Problem Statement

We first define the two-stage stochastic programming problem (P) that we address in this paper,

$$(P) \quad \min \quad z = c^T x + \sum_{\omega \in \Omega} \tau_\omega d_\omega^T y_\omega \tag{1a}$$

$$A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{1b}$$

$$A_{1,\omega} x + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \quad \forall \omega \in \Omega \tag{1c}$$

$$x \in X, \quad X = \left\{ x : x_i \in \{0,1\}, \forall i \in I_1, \ 0 \leq x \leq x^{ub} \right\} \tag{1d}$$

$$y_\omega \in Y \quad \forall \omega \in \Omega, \quad Y = \left\{ y : y_j \in \{0,1\}, \forall j \in J_1, \ 0 \leq y \leq y^{ub} \right\} \tag{1e}$$

Here, x represents the first stage decisions. y_ω represents the second stage decisions in scenario ω. τ_ω represents the probability of scenario ω. g_0, $g_{1,\omega}$, can be smooth nonconvex functions. Both the first and the second stage decisions are mixed-binary. Let $I = \{1, 2, \cdots, n\}$ be the index set of all the first stage variables. $I_1 \subseteq I$ is the subset for indices of the binary first stage variables. Let $J = \{1, 2, \cdots, m\}$ be the index set of all the second stage variables. $J_1 \subseteq J$ is the subset for the indices of the binary second stage variables. x^{ub} is a vector that represents the upper bound of all the first stage variables. y^{ub} is a vector that represents the upper bound of all the second stage variables. We make the following assumptions about problem (P).

Assumption 1. *Problem (P) has relatively complete recourse, i.e., any solution x that satisfies the first stage constraints has feasible recourse decisions in the second stage.*

Assumption 2. *The feasible region of (P) is compact.*

Assumption 3. *x^{ub} and y^{ub} are finite, i.e., both the first and the second stage decisions are bounded.*

3 Motivation for a New Algorithm

In this paper, we address the problem of solving problem (P), i.e., nonconvex SMINLPs with mixed-binary first and second stage variables. Up untill now, the decomposition algorithms that explicitly consider the nonconvex subproblems are the branch and bound algorithms proposed

by Cao and Zavala [10] and Kannan and Barton [15] where the authors branch on the first stage variables and derive a valid bound at each node of the branch and bound process. The valid bound can come from Lagrangean relaxation or perfect information, which can be considered as a special case of Lagrangean relaxation with zero dual multipliers. The authors in [10] and [15] prove finite convergence of their algorithms. However, the branch and bound algorithms tend to have slow convergence computationally, especially when the number of first stage variables is large, i.e., a large number of nodes need to be solved in order to prove global optimality. In this paper, we explore the possibility of doing branch and cut instead of branch and bound.

The main challenge for doing branch and cut is how we can integrate cutting planes in the algorithm while still guaranteeing convergence. In the introduction section, we have reviewed the literature for using Benders-like decomposition to solve linear SMIPs where parametric cuts are used to convexify the subproblems. For nonconvex SMINLPs, cutting planes cannot be used directly to convexify the subproblems. However, if we replace the nonconvex functions in stage 2 by their convex relaxations or MILP relaxations, for example, we can relax a bilinear term by its McCormick envelope or piecewise McCormick envelope, the subproblems will become MILPs or convex MINLPs, which can be convexified by cutting planes. In principle, if the convexified subproblems are MILPs, parametric cuttting planes such as Gomory mixed-integer cut [9], lift-and-project cut [8], RLT [37] can be used. If the convexified subproblems become convex MINLPs, rank-one lift-and-project cuts can be used in the same way as in Li and Grossmann [20]. In this paper, only rank-one lift-and-project cuts for MILP subproblems are implemented. The framework can be easily extended to other types of cutting planes. Therefore, in order to integrate the cutting planes into the decomposition algorithm, the idea is to have a generalized Benders decomposition algorithm where the nonconvex subproblems are first relaxed to MILP problems and the MILP problems are convexified by cutting planes to derive valid Benders cuts. However, this will not guarantee global optimality.

In order to solve the problem to global optimality, we add Lagrangean cuts to the Benders master problem as well. Lagrangean cuts are proved to be valid for the Benders master problem [30, 20]. If we add Lagrangean cuts to the Benders master problem, the lower bound obtained by the Benders master problem is at least as tight as using Lagrangean decomposition (see Proposition 2 in [20]). Therefore, we propose a generalized Benders decomposition (GBD) based branch and cut algorithm where we have both Benders cuts and Lagrangean cuts in the Benders master problem and branch on the first stage variables similar to Kannan and Barton [15]. We are able to prove that the proposed algorithm has convergence in the limit.

The paper is organized as follows: In section 4, we describe the decomposition algorithm. In section 5, we prove the convergence of the proposed algorithm. In section 6, we describe the implementation details. The computational results are shown in section 7. We draw the conclusion in section 8.

4 The Proposed Algorithm

4.1 Overview of the Proposed Algorithm

The major steps of the proposed algorithm are shown in Figure 1. There are two major components in the proposed algorithm: (1) spatial branch and bound (described by the rounded rectangles in Figure 1). (2) the generalized Benders decomposition-based algorithm to solve a

single node in the branch and bound process (described by the regular rectangles in Figure 1).

At a high level, the algorithm performs spatial branch and bound search on the first stage variables. We need to specify the node selection rule and branching rule for the spatial branch and bound algorithm. The details are discussed in subsection 4.3.

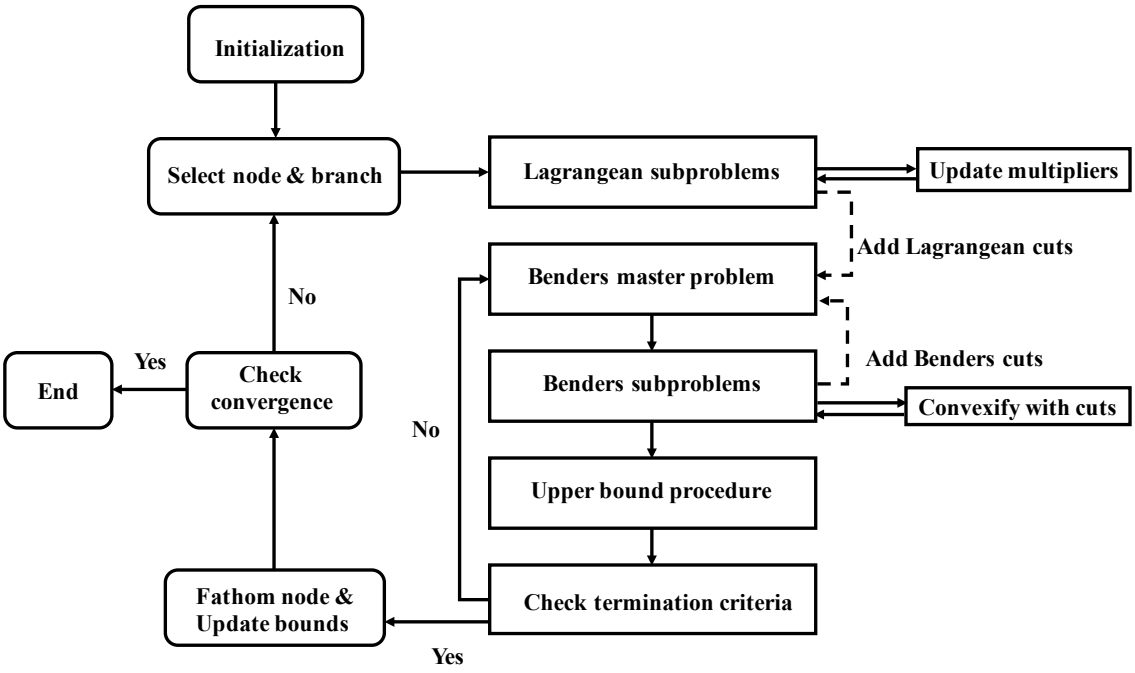

Figure 1. Flowchart of the proposed algorithm

Note that we only branch on the first stage variables. Therefore, each node in the branch and bound process has a decomposable structure. We solve each node by the generalized Benders decomposition-based algorithm described by the regular rectangles in Figure 1. We have two types of valid inequalities including Benders cuts and Lagrangean cuts in the Benders master problem. The Lagrangean cuts come from solving the Lagrangean subproblems iteratively. We update the Lagrangean multipliers after each Lagrangean iteration. Then we add all the Lagrangean cuts to the Benders master problem and start the Benders iterations. Note that the Benders iterations are *independent* of the Lagrangean iterations. In the Benders iterations, we solve the Benders master problem and the Benders subproblems iteratively. The Benders subproblems are convexified by cutting planes. An upper bound can be obtained at each Benders iteration using some heuristics. We check some termination criteria in each Benders iteration. The details of the decomposition algorithm to solve a single node will be described in subsection 4.2.

4.2 Decomposition Algorithm to Solve a Single Node

Before we go to the branch and bound process, we first describe the decomposition algorithm to solve a single node since the branching rules of branch and bound process depend the solutions of the decomposition algorithm.

At a given node q of the branch-and-bound tree, we solve the following problem (P_q) using

generalized-Benders decomposition with Benders cuts and Lagrangean cuts.

$$(P_q) \quad \min \quad c^T x + \sum_{\omega \in \Omega} \tau_\omega d_\omega^T y_\omega \tag{2a}$$

$$A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{2b}$$

$$A_{1,\omega} x + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \quad \forall \omega \in \Omega \tag{2c}$$

$$x \in X_q, \quad X_q = \left\{ x : x_i \in \{0,1\}, \forall i \in I_1, \ x_q^{lb} \leq x \leq x_q^{ub} \right\} \tag{2d}$$

$$y_\omega \in Y \quad \forall \omega \in \Omega, \quad Y = \left\{ y : y_j \in \{0,1\}, \forall j \in J_1, \ 0 \leq y \leq y^{ub} \right\} \tag{2e}$$

where x_q^{ub} and x_q^{lb} are the upper and lower bounds for the first-stage decisions at node q, respectively. The only difference between problem (P_q) and problem (P) is that we have branched on the first stage variables x. The domain of x at node q is represented by set X_q.

Before we go to the steps of the proposed decomposition algorithm, we define the subproblems that are used in the proposed decomposition algorithm, i.e., the regular rectangles in Figure 1. We have both Benders iterations and Lagrangean iterations, which are solved separately. We use letter k to denote the Benders iteration number. We use letter l to denote the Lagrangean iteration number. Note that the Benders iteration number is *independent* of the Lagrangean iteration number, which will become clear when we describe the steps of the decomposition algorithm in Algorithm 1.

Lagrangean subproblems

In order to define the Lagrangean subproblems, problem (P_q) is first reformulated as $(PNAC_q)$ by adding nonanticipativity constraints (NACs) in equation (3d).

$$(PNAC_q) \quad \min \quad c^T x + \sum_{\omega \in \Omega} \tau_\omega d_\omega^T y_\omega \tag{3a}$$

$$A_0 x_\omega \geq b_0, \quad g_0(x_\omega) \leq 0 \tag{3b}$$

$$A_{1,\omega} x_\omega + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \quad \forall \omega \in \Omega \tag{3c}$$

$$x_{\omega 1} = x_{\omega 2}, x_{\omega 1} = x_{\omega 3}, \cdots, x_{\omega 1} = x_{\omega |\Omega|} \tag{3d}$$

$$x_\omega \in X_q, \quad y_\omega \in Y \quad \forall \omega \in \Omega \tag{3e}$$

The $NACs$ can be dualized by multiplying π_ω^l to the constraints, $x_{\omega 1} = x_{\omega+1}, \omega = \omega_1, \omega_2, \cdots, \omega_{|\Omega|-1}$. The Lagrangean subproblem at iteration l of Lagrangean decomposition at node q is defined as follows:

$$(SL_\omega^l)^q : \quad \min \quad z_{SL,\omega}^{l,q} = \tau_\omega (c^T x_\omega + d_\omega^T y_\omega) + \mu_\omega^l x_\omega \tag{4a}$$

$$s.t. \quad A_0 x_\omega \geq b_0, \quad g_0(x_\omega) \leq 0 \tag{4b}$$

$$A_{1,\omega} x_\omega + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \tag{4c}$$

$$x_\omega \in X_q \tag{4d}$$

$$y_\omega \in Y \tag{4e}$$

where $\mu_{\omega 1}^l = \sum_{\omega=\omega 1}^{\omega_{|\Omega|-1}} \pi_\omega^l$, $\mu_{\omega+1}^l = -\pi_\omega^l$, $\omega = \omega_1, \omega_2, \cdots, \omega_{|\Omega|-1}$. In this paper, the Lagrangean multipliers are updated using the subgradient method [31].

Let $z^{*l,q}_{SL,\omega}$ be the optimal objective value of the Lagrangean subproblem $(SL^l_\omega)^q$. $\sum_\omega z^{*l,q}_{SL,\omega}$ yields a lower bound of problem (P_q). Lagrangean cuts $\eta_\omega \geq z^{*l,q}_{SL,\omega} - \mu^l_\omega x$ can be added to the Benders master problem after each Lagrangean iteration as proved by Ogbe and Li [30], Li and Grossmann [20].

Let set L be the set of Lagrangean iteration numbers, i.e., we solve the Lagrangean subproblems for $|L|$ iterations. Note that the Lagrangean iterations are separate from the Benders iterations. The only interaction between Lagrangean and Benders is that Lagrangean cuts are added to the Benders master problem. We will not start the Benders iterations until the Lagrangean subproblems are solved for $|L|$ iterations.

Benders master problem

The Benders master problem at the kth Benders iteration at node q is defined as follows,

$$(MB_k)^q: \quad \min \quad z^{k,q}_{MB} = \sum_\omega \eta_\omega \tag{5a}$$

$$s.t. \quad A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{5b}$$

$$\eta_\omega \geq z^{*l,q}_{SL,\omega} - \mu^l_\omega x \quad \forall \omega \in \Omega, l \in L \tag{5c}$$

$$\eta_\omega \geq z^{*k',q}_{SB,\omega} + (\lambda^{k'}_\omega)^T (x - \tilde{x}^{k'}) + \tau_\omega c^T x \quad \forall \omega \in \Omega, k' < k \tag{5d}$$

$$x \in X_q, \quad \eta_\omega \in \mathbb{R} \quad \forall \omega \in \Omega \tag{5e}$$

where (5c) are Lagrangean cuts, (5d) are Benders cuts. Note that in each Benders iteration k, the Lagrangean cuts are added for all Lagrangean iterations $l \in L$ for all scenario ω. The Lagrangean cuts are derived by solving the Lagrangean subproblems as described before. The Benders cuts are added for all Benders iterations $k' < k$ all scenario $\omega \in \Omega$. The Benders cuts are derived by solving the Benders subproblems, which will be described next.

Let $z^{*k,q}_{MB}$ be the optimal objective value of the Benders master problem. $z^{*k,q}_{MB}$ is a valid lower bound of problem (P_q). Let $\tilde{x}^k$ be the optimal solution of problem $(MB_k)^q$.

Benders subproblems

As we have discussed in the introduction section, a valid Benders cut cannot be derived by simply fixing x at $\tilde{x}^k$ and solve the rest of the subproblems because of the nonconvexities in stage 2. The nonconvexities in the second stage lies in the nonconvex functions $g_{1,\omega}$ and the integrality constraints on some of the y_ω variables. In order to derive Benders cuts, we need to have convex relaxations for the second stage constraints. $g_{1,\omega}(y_\omega)$ can be relaxed by some convex function $\tilde{g}_{1,\omega}(y_\omega, t_\omega)$ where t_ω are the additional variables that are introduced for the convex relaxations. Note that the variables t_ω can be mixed-binary. For example, we may relax a blinear term by its piecewise McCormick envelope where both continuous and binary variables need to be introduced. Without loss of generality, we assume that J_2 is the index set where $(t_\omega)_j$, $j \in J_2$ are the binary variables in (t_ω). If (t_ω) are all continuous, set J_2 is an empty set.

After $g_{1,\omega}$ is convexified with $\tilde{g}_{1,\omega}$, the rest of the problem may still have integrality constraints, i.e., a convex MINLP or an MILP. In order to have a continuous convex relaxation, one alternative is to simply relax the integrality constraints. Another alternative is to use the convexification schemes for convex MINLP/MILP subproblems. MILP subproblems can be convexified by parametric cuttting planes such as Gomory mixed-integer cut [9], lift-and-project cut [8], RLT [37]. Convex MINLP subproblems can be convexified by rank-one lift-and-project cuts [16, 20].

In this paper, only rank-one lift-and-project cuts for MILP subproblems are implemented. Here, we assume that constraints $\tilde{g}_{1,\omega}$ are linear. For lift-and-project cuts for convex nonlinear

constraints, we refer the readers to [20]. To simplify notation, we denote the constraints $A_{1,\omega}x + \tilde{g}_{1,\omega}(y_\omega, t_\omega) \leq b_{1,\omega}$, $0 \leq y \leq y^{ub}$, $x_q^{lb} \leq x \leq x_q^{ub}$ as $\tilde{A}_q \begin{pmatrix} x \\ y_\omega \\ t_\omega \end{pmatrix} \geq \tilde{b}_q$. Suppose $\tilde{x}^k$ is the optimal solution of the Benders master problem at iteration k. $\tilde{y}_\omega^k$, $\tilde{t}_\omega^k$ is the optimal solution of Benders subproblem before adding the lift-and-project cuts generated at iteration k. If $(\tilde{y}_\omega^k)_j, j \in J_1$ is fractional, rank-one lift-and-project cuts are generated by solving the following cut generating linear program $(CGLP_j)^q$ [8].

$$(CGLP_j)^q \quad \min \quad \alpha^x \tilde{x}^k + \alpha^y \tilde{y}_\omega^k + \alpha^t \tilde{t}_\omega^k - \beta \tag{6a}$$

$$\alpha = u^T \tilde{A}_q - u_0 e_{j+n}, \quad \alpha = v^T \tilde{A}_q + v_0 e_{j+n} \tag{6b}$$

$$\beta = u^T \tilde{b}_q, \quad \beta = v^T \tilde{b}_q + v_0 \tag{6c}$$

$$u^T e + v^T e + u_0 + v_0 = 1 \tag{6d}$$

$$u, v, u_0, v_0 \geq 0 \tag{6e}$$

Note that we can also generate lift-and-project cuts corresponding to the binary variables in t_ω. We only need to change equation (6b) to

$$\alpha = u^T \tilde{A}_q - u_0 e_{j+n+m}, \quad \alpha = v^T \tilde{A}_q + v_0 e_{j+n+m} \tag{7}$$

to generate the rank-one cut corresponding to $(t_\omega)_j$, $j \in J_2$. The newly generated cuts are then added to the Benders subproblem $(SB_\omega^k)^q$ at iteration k for scenario ω defined as follows,

$$(SB_\omega^k)^q : \quad \min \quad z_{SB,\omega}^k = \tau_\omega d^T y_\omega \tag{8a}$$

$$x = \tilde{x}^k \tag{8b}$$

$$A_{1,\omega}x + \tilde{g}_{1,\omega}(y_\omega, t_\omega) \leq b_{1,\omega} \tag{8c}$$

$$\alpha_{k'c}^x x + \alpha_{k'c}^y y_\omega + \alpha_{k'c}^t t_\omega \geq \beta_{k'c} \quad \forall k' \leq k, c \in C^{k'} \tag{8d}$$

$$0 \leq y \leq y^{ub} \tag{8e}$$

where (8d) are the cutting planes that are generated to convexify the subproblem. Set $C^{k'}$ represents the set of cuts that are generated at iteration k'. Note that we accumulate the cuts generated in all iterations $k' \leq k$.

Upper bound procedure

There are multiple ways to obtain a feasible solution to the original problem. A simple way would be fixing the first stage decisions and solve the upper bound subproblem (UB_ω) for each scenario ω separately,

$$(UB_\omega) : \quad \min \quad z_{UB,\omega} = \tau_\omega d_\omega^T y_\omega \tag{9a}$$

$$s.t. \quad g_{1,\omega}(y_\omega) \leq b_{1,\omega} - A_{1,\omega}\tilde{x} \tag{9b}$$

$$g_{2,\omega}(y_\omega) \leq b_{2,\omega} \tag{9c}$$

$$y_\omega \in Y \tag{9d}$$

where the value of the first stage decisions is fixed at $\tilde{x}$. The choice of $\tilde{x}$ decides the upper bound that is obtained from (UB_ω). Here we propose three heuristics to obtain a "good" $\tilde{x}$.

- Let $\tilde{x} = \tilde{x}^k$, i.e., be the optimal solution that is obtained in the Benders master problem $(MB_k)^q$ at iteration k.
- Let $x^{*l,q}_{\omega 1}, x^{*l,q}_{\omega 2}, \cdots, x^{*l,q}_{\omega|\Omega|}$ be the optimal solution for the Lagrangean subproblems $(SL^l_\omega)^q$ at iteration l. We define

$$x^{*l,q}_{avg} = \sum_{\omega\in\Omega} \tau_\omega x^{*l,q}_\omega$$

as the probability weighted average of the optimal first stage solutions over all scenarios. Then we select scenario ω^*, such that

$$\omega^* = \arg\min_{\omega\in\Omega} \sum_{i\in I} \left(\frac{(x^{*l,q}_{avg})_i - (x^{*l,q}_\omega)_i}{(x^{ub})_i} \right)^2$$

ω^* is the scenario where the optimal solution $x^{*l,q}_\omega$ has the smallest distance with the weighted average x^{*l}_{avg}. Note that in calculating the distances all the first stage variables are scaled by the distances of their original upper and lower bounds (the original lower bounds are assumed to be zero in problem (P)). We can fix $\tilde{x}$ at $x^{*l,q}_{\omega^*}$ and solve (UB_ω) to obtain a feasible solution.

- Randomly select one scenario ω and fix the first stage decisions at the optimal value of the Lagrangean subproblem corresponding to this scenario, i.e., $x^{*l,q}_\omega$.

The steps of the decomposition algorithm to solve a single node

With the definitions of the subproblems, we outline the steps of the decomposition algorithm to solve a single node q in Algorithm 1. The decomposition algorithm is defined as a *solvenode*(q) function. First, Lagrangean subproblems $(SL^l_\omega)^q$ are solved for all Lagrangean iterations $l \in L$. We add all the Lagrangaen cuts (5c) to the Benders master problem $(MB_1)^q$ (the Benders master problem in the first iteration at node q). Then we iteratively solve the Benders master problem $(MB_k)^q$ and the Benders subproblems $(SB^k_\omega)^q$ with cutting planes. After each Benders iteration, we add Benders cuts (5d) to the Benders master problem. The proposed upper bound procedure can be used to obtain feasible solutions based on the optimal solutions of the Lagrangean subproblems or the Benders master problem. We can terminate if the node is solved to optimality. Otherwise, the algorithm terminates by setting the maximum Lagrangean and Benders iterations to $|L|$ and $|K|$, respectively. The decomposition algorithm always returns an upper bound UB_q and a lower LB_q at node q.

4.3 Branch and bound

If we apply the function *solvenode*(q) described in Algorithm 1 to the rootnode (problem (P)), we cannot guarantee to solve it to global optimality because of the duality gap. That is why we need to branch and bound on the first stage decisions as in Caroe and Schultz[11], Cao and Zavala [10], Kannan and Barton [15]. The spatial branch and bound corresponds to the rounded rectangles in Figure 1. Most importantly, we need to define the node selection rule and the branching rule used in the branch and bound process. Let Γ be the set of active nodes.

Node selection rule

Algorithm 1: Algorithm to solve a single node

Function *solvenode(q)*
  `/* Lagrangean iterations */`
  for $l = 1, 2, \cdots, L$ **do**
    for $\omega = \omega_1, \omega_2, \cdots, \omega_{|\Omega|}$ **do**
      Solve Lagrangean subproblems $(SL^l_\omega)^q$. Let $z^{*l,q}_{SL,\omega}$ be the optimal objective value
    end
    Update $LB_q = \max(LB_q, \sum_\omega z^{*l,q}_{SL,\omega})$.
    Update UB_q using the upper bound procedure.
    if $UB_q - LB_q \leq \epsilon$ **then**
      return q
    end
    Update Lagrangean multipliers μ^l_ω with the subgradient method
  end
  Add Lagrangean cuts (5c) to Benders master problem $(MB_1)^q$.
  `/* Benders iterations */`
  for $k = 1, 2, \cdots, K$ **do**
    Solve $(MB_k)^q$. Let $z^{*k,q}_{MB}$ be the optimal objective value of $(MB_k)^q$. Set $LB_q = z^{*k,q}_{MB}$
    for $\omega = \omega_1, \omega_2, \cdots, \omega_{|\Omega|}$ **do**
      Solve Benders subproblems $(SB^k_\omega)^q$. Let y^*_ω, t^*_ω be the optimal value.
      if *there is fractional* $(y^*_\omega)_j$ *for* $j \in J_1$, $(t^*_\omega)_j$ *for* $j \in J_2$ **then**
        Generate cutting planes (8d) by solving $(CGLP_j)^q$ and add to $(SB^k_\omega)^q$.
        Solve $(SB^k_\omega)^q$ again with the newly generated cutting planes.
      end
      Generate Benders cuts (5d) and add to Benders master problem $(MB_k)^q$.
    end
    Update UB_q with the upper bound procedure.
    if $UB_q - LB_q \leq \epsilon$ **then**
      return q
    end
  end
  return q.
end

Select node q such that $q = \arg\min_{q \in \Gamma} LB_q$.

After a node is selected, we branch on one of the first stage variables of this node. The branching rules are defined according to the bounds of the first stage variables and the solutions obtained by the proposed decomposition algorithm (Algorithm 1) at this node.

Branching rules

1. Select the first stage variable with the largest normalized relative diameter,

$$i^* = \arg\max_{i \in I} \frac{(x_q^{ub})_i - (x_q^{lb})_i}{(x_{q_0}^{ub})_i - (x_{q_0}^{lb})_i} \delta_i$$

q_0 represent the root node. δ_i is a normalization factor for variable i. Two new nodes q_1 and q_2 are then created.

$$(x_{q_j}^{ub})_i = (x_q^{ub})_i, \quad (x_{q_j}^{lb})_i = (x_q^{lb})_i \quad j \in \{1,2\}, i \neq i^*$$

$$(x_{q_1}^{lb})_{i^*} = (x_q^{lb})_{i^*}, \quad (x_{q_1}^{ub})_{i^*} = \frac{1}{2}\Big((x_q^{lb})_{i^*} + (x_q^{ub})_{i^*}\Big)$$

$$(x_{q_2}^{lb})_{i^*} = \frac{1}{2}\Big((x_q^{lb})_{i^*} + (x_q^{ub})_{i^*}\Big), \quad (x_{q_2}^{ub})_{i^*} = (x_q^{ub})_{i^*},$$

where the domain of variable i^* is bisected.

2. Let $(x_q^*, y_{q\omega 1}^*, y_{q\omega 2}^*, \cdots, y_{q\omega|\Omega|}^*)$ be the best feasible solution found at node q. Then we select variable i^* where best feasible first stage variable has the largest normalized distance to its bounds,

$$i^* = \arg\max_{i \in I} \frac{\min\left\{(x_q^{ub})_i - (x_q^*)_i, (x_q^*)_i - (x_q^{lb})_i\right\}}{(x_{q_0}^{ub})_i - (x_{q_0}^{lb})_i} \delta_i$$

Two new nodes q_1 and q_2 are then created.

$$(x_{q_j}^{ub})_i = (x_q^{ub})_i, \quad (x_{q_j}^{lb})_i = (x_q^{lb})_i \quad j \in \{1,2\}, i \neq i^*$$

$$(x_{q_1}^{lb})_{i^*} = (x_q^{lb})_{i^*}, \quad (x_{q_1}^{ub})_{i^*} = (x_q^*)_{i^*}$$

$$(x_{q_2}^{lb})_{i^*} = (x_q^*)_{i^*}, \quad (x_{q_2}^{ub})_{i^*} = (x_q^{ub})_{i^*},$$

where the domain of $(x)_{i^*}$ is divided based on the best feasible solution found on node q.

3. Let x_q^{avg} be the weighted average of first stage decisions obtained from Lagrangean subproblems in the iteration where the tightest lower bound can be found. Then we select variable i^* such that

$$i^* = \arg\max_{i \in I} \frac{\min\left\{(x_q^{ub})_i - (x_q^{avg})_i, (x_q^{avg})_i - (x_q^{lb})_i\right\}}{(x_{q_0}^{ub})_i - (x_{q_0}^{lb})_i} \delta_i$$

Similarly, two new nodes q_1 and q_2 are then created.

$$(x_{q_j}^{ub})_i = (x_q^{ub})_i, \quad (x_{q_j}^{lb})_i = (x_q^{lb})_i \quad j \in \{1,2\}, i \neq i^*$$

$$(x_{q_1}^{lb})_{i^*} = (x_q^{lb})_{i^*}, \quad (x_{q_1}^{ub})_{i^*} = (x_q^{avg})_{i^*}$$

$$(x_{q_2}^{lb})_{i^*} = (x_q^{avg})_{i^*}, \quad (x_{q_2}^{ub})_{i^*} = (x_q^{ub})_{i^*},$$

With the node selection rule and the branching rules, we outline the steps of the proposed

Algorithm 2: Generalized-Benders decomposition-based branch and cut algorithm

```
   /* Initialization                                                    */
 1 Initialize UB = +∞, LB = −∞ Create root node q_0. Set X_{q_0} = X. Set all the Lagrangean
     multipliers μ_ω to zero. solvenode(q_0). Create a node list Γ = {q_0}.
 2 while Γ ≠ ∅ do
 3 |   Select node q such that q = arg min_{q∈Γ} LB_q.
 4 |   Apply one of the proposed branching rules and create two child node of q, denoted as q_1,
   |     q_2. Branching rule 1 should be applied at least once after a finite number of iterations.
   |     Let Γ = Γ\{q}
 5 |   solvenode(q_1); solvenode(q_2).
 6 |   Let UB = min_{q∈Γ} UB_q, LB = min_{q∈Γ} LB_q.
 7 |   for q ∈ Γ do
 8 |   |   if UB_q − LB_q ≤ ε then
 9 |   |   |   Γ = Γ\{q}   /* Fathom by optimality                       */
10 |   |   end
11 |   |   if LB_q − UB ≥ ε then
12 |   |   |   Γ = Γ\{q}   /* Fathom by bound                           */
13 |   |   end
14 |   end
15 end
```

generalized-Benders decomposition-based branch and cut algorithm in Algorithm 2.

The steps of generalized-Benders decomposition-based branch and cut algorithm

At a high level, the proposed algorithm is performing spatial branch and bound search that includes node selection and branching, node fathom. The steps of the spatial branch and bound are described in Algorithm 2. The branch and bound algorithm selects node with the tightest lower bound. Different branching rules described above can be applied. However, branching rule 1 must be applied at least once after a finite number of iterations to have the theoretical guarantee of convergence, which will be discussed in detail in section 5. Each node in the branch and bound process is solved with the generalized Benders decomposition-based algorithm described in Algorithm 1. The flowchart of the algorithm is shown in Figure 1 where the rounded rectangles describe the high-level spatial branch-and-bound algorithm (Algorithm 2); the regular rectangles describe the decomposition algorithm that solves a single node (Algorithm 1).

Remark 1. The Benders cuts and Lagrangean cuts in a parent node can be inherited by its child nodes. The parametric cutting planes in the Benders subproblems of a parent node can also be inherited by its child nodes. Therefore, the solution process of the child nodes can be warm-started by inheriting those cuts.

Remark 2. After branching, the Lagrangean multipliers of a child node can be either initialized to zero or initialized to the multipliers that give the tightest lower bound for its parent node.

5 Convergence of the proposed algorithm

Cao and Zavala [10] and Kannan and Barton [15] have proved the convergence of their algorithms, both of which are Lagrangean decomposition-based branch and bound algorithm. Both proofs rely on Chapter IV of Horst and Tuy [13]. Since we also have Lagrangean cuts in the Benders master problem, the lower bound that we can obtain at each node would be at least as tight as Lagrangean decomposition as proved by Li and Grossmann (Proposition 2 of [20]). Therefore, it would be straightforward to prove convergence based on previous work.

In order to make this paper self-contained, we briefly describe the convergence proof for our proposed algorithm. We first define some notations for the branch and bound algorithm. D represents the feasible region of the first stage decisions. $D = \{x|\exists(x, y_{\omega 1}, y_{\omega 2}, \cdots, y_{\omega|\Omega|}) \text{ feasible for problem } (P)\}$. Since we have assumed that the problem has relatively complete recourse, we can let $D = \{x|A_0 x \geq b_0, g_0(x) \leq 0, x \in X\}$. The domain of the first stage decisions at node q is defined as X_q in equation (2d). X_{q_0} is the domain of x at the root node. X_{q_p} denotes the domain of x of a node at level p of the tree. A path in the branch and bound tree from the rootnode to one leaf node has one node in each level p of the tree, which can be represented as $\{X_{q_p}\}$. q_{p+1} is a child of q_p and $X_{q_{p+1}} \subset X_{q_p}$. $\delta(X_q)$ represents the diameter of X_q. $\delta(X_q) = ||x_q^{ub} - x_q^{lb}||_\infty$. UB_q and LB_q are upper and lower bound obtained at node q, respectively. In order to prove convergence, we need to prove $\lim\limits_{q\to\infty} UB_q = \lim\limits_{q\to\infty} LB_q = z^*$.

Definition 1. *A subdivision is called exhaustive if* $\lim\limits_{p\to\infty} \delta(X_{q_p}) = 0$*, for all decreasing subsequences* X_{q_p} *generated by the subdivision (Definition IV.10 in [13]).*

Lemma 1. *The subdivision process of the proposed algorithm is exhaustive.*

Proof. Since branching rule 1 is applied at least once after a finite number of iterations and the bounds for each variable at the rootnode and the normalization factors δ_i are all finite, the variable i with the largest diameter is divided by half once after a finite number of iterations. Therefore, $\delta(X_{q_p}) = ||x_{q_p}^{ub} - x_{q_p}^{lb}||_\infty$ is divided by half once after a finite number of iterations. $\lim\limits_{p\to\infty} \delta(X_{q_p}) = 0$ (see Lemma 3.5.1 in [15]). □

Definition 2. *A selection operation is said to be bound improving if, after a finite number of steps, at least one partition element where the actual lower bounding is attained is selected for further partition. (Definition IV.6 in [13])*

Lemma 2. *The selection operation of the proposed algorithm is bound improving.*

Proof. This is obvious from the node selection rule used by the proposed algorithm. □

Definition 3. *The "deletion by infeasibility" rule throughout a branch and bound procedure is called certain in the limit if, for every infinite decreasing sequence $\{X_{q_p}\}$ of successively refined partition elements with limit $\bar{X}$, we have $\bar{X} \cap D \neq \emptyset$. (Definition IV.8. in [13])*

Lemma 3. *Deletion by infeasibility is certain in the limit in the proposed algorithm.*

Proof. Since branching rule 1 is applied at least once after a finite number of iterations in the proposed branch and bound algorithm, any infinite decreasing sequence $\{X_{q_p}\}$ would converge to a point $\bar{x}$. We prove this by contradiction. Suppose $\bar{x} \notin D$. Since D is compact, there exist $r > 0$, such that $\{x| \ ||x - \bar{x}||_2 \leq r\} \cap D = \emptyset$. Therefore, $\exists p_0$, such that $\forall p \geq p_0$, $X_{q_p} \cap D = \emptyset$. $X_{q_{p_0}} \cap D = \emptyset$, which means the sequence $\{X_{q_p}\}$ should have been fathomed at $X_{q_{p_0}}$. We complete the proof by contradiction. (Similar proof in Lemma 2 of [10]) □

Definition 4. *A lower bounding operation is called strongly consistent if, at every iteration, any undeleted partition set can be further refined and if any infinite decreasing sequence $\{X_{q_p}\}$ successively refined partition elements contains a sub-sequence $\{X_{q_{p'}}\}$ satisfying $\bar{X} \cap D \neq \emptyset$, $\lim_{p\to\infty} LB_{q_p} = z^*(\bar{X} \cap D)$, where $\bar{X} = \cap_p X_{q_p}$. (Definition IV.7. in [13])*

Lemma 4. *The proposed algorithm is strongly consistent.*

Proof. Since the division is exhaustive, the sequence $\{X_{q_p}\}$ would converge to a singleton $\bar{x}$. From Lemma 3, $\bar{x} \in D$. Note that the lower bound LB_{q_p} is at least as tight as $\sum_\omega z^{*l,q_p}_{SL,\omega}$.

$$\begin{aligned}\lim_{p\to\infty} LB_{q_p} \geq \lim_{p\to\infty} \sum_{\omega\in\Omega} z^{*l,q_p}_{SL,\omega} = \quad & \sum_{\omega\in\Omega} \min \quad \tau_\omega(c^T\bar{x}_\omega + d^T_\omega y_\omega) + \mu^l_\omega \bar{x}_\omega \\ & s.t. \quad A_{1,\omega}\bar{x}_\omega + g_{1,\omega}(y_\omega) \leq b_{1,\omega}, \quad y_\omega \in Y \quad \forall\omega \in \Omega \\ & = z^*(\bar{X} \cap D)\end{aligned}$$

The last equality holds because the $NACs$ are satisfied. Therefore, $\lim_{p\to\infty} LB_{q_p} = z^*(\bar{X} \cap D)$ □

Now we are ready to prove the convergence of the proposed algorithm.

Theorem 1. *The propose algorithm is convergent, i.e., $\lim_{q\to\infty} LB_q = \lim_{q\to\infty} UB_q = z^*$.*

Proof. Since we have proved that the subdivision process is exhaustive (Lemma 1), "deletion by infeasibility" is certain in the limit (Lemma 3), and the lower bounding operation is strongly consistent (Lemma 4), we have that the lower bounding operation is consistent according to Lemma IV.5 in [13]. Since we have proved that the selection operation is bound improving, the branch and bound procedure is convergent according to Theorem IV.3 in [13]. □

6 Implementation

The proposed algorithm is implemented as part of the PlasmoAlgorithms package in Julia programming language. PlasmoAlgorithms is package for decomposition algorithms that use PlasmoGraph [14] as input. PlasmoGraph is a graph based data structure based on JuMP/Julia. Each node in a PlasmoGraph contains an optimization model written in JuMP. In this case, in order to access our algorithm, the user only needs write the first and second stage problems in different

nodes in PlasmoGraph and connect the first and second stage nodes with linking constraints. A brief tutorial is provided in [1].

The default solver for the Lagrangean subproblems, upper bound subproblems, which are MINLPs or NLPs, is BARON. The default solver for the MILP Benders master problem and the LP Benders subproblems are CPLEX.

Rank-one lift-and-project cuts are implemented as the routine to convexify the MILP subproblems after the nonconvex functions in stage 2 are replaced by their convex relaxations. Numerical issues can occur if the rank-one cuts are not implemented properly especially for the problems that are not scaled well. To avoid the numerical instability, we do not generate all the theoretically valid cuts. We only generate a rank-one cut when the optimal solution of the binary variables in solving the subproblems are between 0.01 and 0.99. A cut will not be added if the absolute value of any coefficient in the optimal solution of α in $(CGLP_j)^q$ is nonzero but less than 10^{-6}. A cut will not be added if the any of the dual variables correspond to equation (6c) is less than 0.01.

7 Computational results

The proposed algorithm is tested with three problems of this category. A brief description and the computational results for the three problems are described in the subsections below. In all the three problems, we solve the deterministic equivalent of the stochastic programs with three different global solvers, BARON [40], ANTIGONE [26], and SCIP [2]. All the problems are solved using one processor of an Intel Xeon (2.67GHz) machine with 64 GB RAM. The computational results of solving the deterministic equivalent are shown in Tables 1, 3, 5. We also use three different decomposition algorithms to solve each problem shown in Tables 2, 4, and 6. GBD(with cuts) + L represents the proposed algorithm where each node of the branch and bound tree is solved by generalized Benders decomposition with Lagrangean cuts and Benders cuts coming from the subproblems convexified by cutting planes. GBD+L is similar to the proposed algorithm but cutting planes are not used to convexified the Benders subproblems. LD represents the Lagragean decomposition-based branch and bound algorithm proposed by Carøe and Schultz [11]. The tolerance of the relative optimality gap is set to 0.1%. The walltime limit is set to $10,000$ seconds.

7.1 Stochastic pooling problem with contract selection

The stochastic pooling problem with contract selection is an extension of the stochastic pooling problem studied by Li et al [21]. The first stage decisions are design decisions, which include the selection and capacity of feeds and pools. Note that a binary variable is used to model whether a feed or pool is selected or not. Continuous variables are used to represent the capacity of the feeds and pools. The second stage decisions are mainly operating decisions including the mass flow rates that are subject to product quality specifications. Moreover, we consider purchasing the feeds with three different types of contracts, i.e., fixed price, discount after a certain amount, and bulk discount [32]. A binary variable is used to model which contract to select for each feed. The details of the mathematical formulation are described in Appendix 1.

The uncertainties in this problem include demands of the products, prices of the feeds, and selling prices of the products. We assume that all the three uncertain parameters can have high, medium, or low realizations. We create instances of stochastic programs with 3, 9, and 27, scenarios. In the 3-scenario problem, only the demands of the products are uncertain. In the 9

scenario-problem, we consider the Cartesian product of the demands of the feeds and the prices of the feeds. In the 27-scenario problem, the Cartesian product of all the three uncertain parameters are considered.

This problem has 9 binary variables, 9 continuous variables, and 18 linear constraints in stage 1. There are 35 binary variables, 112 continuous variables, 116 linear constraints, 22 nonlinear constraints in stage 2 per scenario. The nonlinear constraints correspond to the bilinear terms in the pooling problem.

The computational results of the deterministic deterministic equivalent are shown in Table 1. While BARON and ANTIGONE can solve the problems with 3 and 9 scenarios to optimality within the time limit, all the solvers fail to solve the problem with 27 scenarios to optimality within the time limit.

Table 1. Walltime and relative optimality gap of solving the deterministic equivalent of the stochastic pooling problem with 3, 9, 27, scenarios

#Scenarios	3	9	27
BARON 18.5.8	5/0.1%	3005/0.1%	10^4/8.7%
ANTIGONE 1.1	16/0.1%	251/0.1%	10^4/1.4%
SCIP 5.0	10^4/54.4%	10^4/100.0%	10^4/100.0%

Table 2. Walltime, relative optimality gap, and number of nodes by using different decomposition algorithms to solve the stochastic pooling problem with 3, 9, 27, scenarios

#Scenarios	3	9	27
GBD(with cuts)+L	152/0.1%/1	502/0.1%/1	2113/0.1%/1
GBD+L	10^4/0.1%/381	10^4/0.8%/39	10^4/1.3%/7
LD	10^4/0.2%/363	10^4/7.1%/43	10^4/12.2%/9

We also apply the three decomposition algorithms, GBD(with cuts)+L, GBD+L, LD, to solve this problem. The walltime, the relative optimality gap, and the number of nodes in the branch and bound tree are shown in Table 2 separated by "/". The proposed algorithm, GBD(with cuts)+L, can solve all the problems to optimality at the root node within the time limit. GBD+L can solve the 3 scenario problem with 381 nodes. However, for the problems with 9 and 27 scenarios, GBD+L cannot close the gap but can provide solutions with reasonably good optimality gap, namely 0.8% and 1.3%, respectively. LD runs out of time for all the problems. For the 3 scenario problem, LD provides 0.2% gap by solving 363 nodes. For the 9 and 27 scenario problem, LD is only able to solve 43 and 9 nodes, respectively, within the time limit. Therefore, the gaps are quite large for LD in large problems.

7.2 Crude selection and refinery optimization under uncertainty

The crude selection and refinery optimization problem comes from the stochastic programming library in GOSSIP (decomposition software for the Global Optimization of nonconvex two-Stage Stochastic mixed-Integer nonlinear Programs) [15]. The problem was first proposed by Yang and Barton [42]. In their model, crudes are purchased in certain predefined discrete amounts in the

first stage. Kannan and Barton [15] extend this work by considering the purchased crudes as 'semi-continuous' variables i.e., the crude purchase quantity can either be zero if the decision maker decides not to purchase that particular crude, or it must lie between some prespecified lower and upper bounds. The second stage decisions correspond to the operating conditions of the refinery, such as the mass flow rates and the split fractions. The decisions need to be subject to mass balances, quality specifications, market demands, and capacity and supply restriction. The uncertainties are in the crude oil qualities and yields. The nonlinearities in the model come from the bilinear terms in the quality specifications in stage 2, which is similar to the formulation in a pooling problem.

This problem has 10 binary variables, 10 continuous variables, and 21 linear constraints, in stage 1. There are 142 continuous variables, 85 linear constraints, 26 nonlinear constraints in stage 2 per scenario.

The two-stage stochastic program for this problem is solved with 5, 10, 20, 40, and 120 scenarios. The computational results of the deterministic equivalent are shown in Table 3. The deterministic equivalent with less than 40 scenarios can be solved to optimality within the time limit with all the solvers where ANTIGONE has the best performance. For the largest instance, the 120 scenario problem, the solvers can obtain good gaps, i.e., 0.4% for SCIP.

Table 3. Walltime and relative optimality gap of solving the deterministic equivalent of the crude selection and refinery optimization under uncertainty with 5,10,20,40,120 scenarios

#Scenarios	5	10	20	40	120
BARON 18.5.8	24/0.1%	5092/0.1%	10^4/0.4%	10^4/0.1%	10^4/0.9%
ANTIGONE 1.1	5/0.1%	26/0.1%	322/0.1%	10^4/0.1%	10^4/0.9%
SCIP 5.0	4/0.1%	60/0.1%	10^4/0.2%	10^4/0.1%	10^4/0.4%

Table 4. Walltime, relative optimality gap, and number of nodes by using different decomposition algorithms to solve the crude selection and refinery optimization under uncertainty with 5, 10, 20, 40, 120 scenarios

#Scenarios	5	10	20	40	120
GBD(with cuts)+L	10^4/1.1%/13	10^4/0.9%/9	10^4/1.0%/5	10^4/0.9%/3	10^4/0.9%/1
GBD+L	10^4/1.8%/155	10^4/0.8%/85	10^4/1.1%/37	10^4/0.9%/17	10^4/0.9%/7
LD	10^4/11.7%/165	10^4/7.8%/81	10^4/16.2%/33	10^4/80.7%/17	10^4/145.3%/7

We also use the three decomposition algorithms to solve this problem. For GBD(with cuts)+L, piecewise McCormick relaxations are used for the bilinear terms in stage 2. Binary variables are introduced to describe piecewise McCormick relaxations, which are convexified by rank-one lift-and-project cuts in the convexifed subproblems. The results are shown in Table 4. It turns out that the decomposition algorithms are not competitive compared with solving the deterministic equivalent directly with the solvers. The gaps remain to be around 1.0% for GBD(with cuts)+L and GBD+L at the time limit. However, from this computational experiment, we can observe that the gap of the proposed algorithm is smaller than LD, especially for the large instances where LD can only solve relatively few nodes.

7.3 Storage design for a multi-product plant under uncertainty

The storage design for a multi-product plant under uncertainty problem also comes from the library in GOSSIP. The case study considers a chemical plant that uses a single reactor to produce products and stores them in storage tanks. The stage 1 decisions are design decisions, i.e., the product tank sizes. In stage 2, binary variables are used to represent the assignment of products to campaigns. Continuous decisions include the production amount in each campaign, the duration of the campaigns, etc. The uncertainties considered are the demands of the products. The details can be found in Rebennack et al [34].

This problem has 3 continuous variables, and 1 nonlinear constraint, in stage 1. The nonlinearity corresponds to a univariate signomial term, which is used to calculate the tank investment cost. There are 9 binary variables, 41 continuous variables, 85 linear constraints, 26 nonlinear constraints in stage 2 per scenario. These nonlinear constraints come from the bilinear terms that are used to calculate the variable inventory costs.

The deterministic equivalent of the stochastic programs is solved with 2, 3, 4, 5, 9, and 27 scenarios. From the results in Table 5, one can observe that BARON and SCIP can solve the small problems to optimality within the time limit but fail to solve the problem to optimality with more than 4 scenarios. On the other hand, ANTIGONE can solve the 9 and 27 scenario problem, but fails to solve the small problems to optimality within the time limit.

Table 5. Walltime and relative optimality gap of solving the deterministic equivalent of the storage design for a multi-product plant under uncertainty with 2,3,4,5,9,27 scenarios

#Scenarios	2	3	4	5	9	27
BARON 18.5.8	1117/0.1%	10^4/2.2%	10^4/7.0%	10^4/11.0%	10^4/13.0%	10^4/13.0%
ANTIGONE 1.1	10^4/2.8%	10^4/8.0%	10^4/11.0%	10^4/13.5%	3/0.1%	87/0.1%
SCIP 5.0	404/0.1%	7960/0.1%	10^4/1.0%	10^4/2.4%	10^4/4.6%	10^4/12.2%

Table 6. Walltime, relative optimality gap, and number of nodes by using different decomposition algorithms to solve the storage design for a multi-product plant under uncertainty with 2,3,4,5,9,27 scenarios

#Scenarios	2	3	4	5	9	27
GBD(with cuts)+L	127/0.1%/47	239/0.1%/63	761/0.1%/141	365/0.1%/59	1250/0.1%/99	8681/0.1%/159
GBD+L	74/0.1%/47	277/0.1%/149	353/0.1%/141	188/0.1%/59	549/0.1%/99	4108/0.1%/159
LD	61/0.1%/47	214/0.1%/149	270/0.1%/141	152/0.1%/59	436/0.1%/99	3219/0.1%/165

Similarly, the three decomposition algorithms are used to solve this problem. The results are shown in Table 6. All the decomposition algorithms can solve this problem to optimality within the time limit. However, in this case LD performs better than GBD(with cuts)+L and GBD+L. The results indicate that the Benders cuts are dominated by the Lagrangean cuts since in most of the instances all the three algorithms have to solve same number of nodes in order to prove optimality. Only in the case of 3 and 27 scenarios, GBD(with cuts)+L requires fewer nodes than LD, which means that the Benders cuts with cutting planes can help to solve the problems.

8 Conclusion

In this paper, we have proposed a generalized-Benders decomposition-based branch and cut algorithm for two stage stochastic programs with nonconvex constraints and mixed-binary first and second stage variables. We include both Lagrangean cuts and Benders cuts in the Benders master problem. The convergence of the proposed algorithm relies on adding Lagrangean cuts to the Benders master problem and performing a branch-and-bound search on the first stage variables, which follows from the convergence proof of general branch and bound by Horst and Tuy [13]. The Benders cuts are derived from the convexified subproblems where cutting planes can be added. Our computational results show that adding cutting planes can potentially accelerate the convergence of the branch-and-bound procedure, especially in the stochastic pooling problem with contract selection. However, in the storage design for a multi-product plant under uncertainty problem, adding rank-one lift-project-cut does not help the branch-and-bound procedure.

For the problems with tight Lagrangean relaxation, the corresponding Benders cuts derived from the Benders subproblem convexified by cutting planes would be dominated by the Lagrangean cuts (storage design problem in subsection 7.3). On the other hand, for the problems with weak Lagrangean relaxation (stochastic pooling problem in subsection 7.1 , and crude selection problem in subsection 7.2), adding Benders cuts may be able to close the duality gap significantly. Therefore, in order to predict whether it is preferable to add cutting planes, heuristics need to be proposed to decide cut generation.

While the proposed algorithm can solve the test cases to small optimality gaps, sovling stochastic nonconvex MINLPs is still challenging. It would be desirable to come up with tighter convex relaxations for the nonconvex terms in stage 2 in order to generate tight Benders cuts. Tighter convex relaxations are in general more difficult to solve. For example, SDP+RLT relaxations [6] are reported to be tight for problems with nonconvex quadratic constraints but are computationally expensive. Therefore, there is a tradeoff between using tight convex relaxations and doing more branching. We propose to have more computational studies in the future to learn the appropriate convex relaxations based on the features of each problem to make the proposed algorithm more efficient.

9 Appendix 1: Mathematical Formulation of Stochastic Pooling Problem with Contract Selection

Sets
i=feeds
j=products
l=pools
p=qualities
c=contracts
ω=scenarios
T_X=(i,l) pairs for which input to pool connection allowed
T_Y= (l,j) pairs for which pool to output connection allowed
T_Z=(i,j) pairs for which input to output connection allowed

Parameters

A_i^U=maximum available flow
A_i^L=minimum available flow
S_l^U=maximum pool size
S_l^L=minimum pool size
D_j^U=maximum product demand
C_{ik}=feed concentration
P_{jk}^U=maximum allowable product concentration
P_{jk}^L=minimum allowable product concentration
α_l=fixed cost parameter for pools
β_l=variable cost parameter for pools
δ_i=fixed cost for feed storage
ξ_i=variable cost for feed storage
τ_ω=probability of scenario ω
$\psi_{i\omega}^c$=unit price for feed i under contract c in scenario ω
$d_{j\omega}$=product unit price in scenario ω
σ_i^c=minimum purchasable amount of feed i under contract c

First stage variables

binary variables:
λ_i=whether feed i exists
θ_l=whether pool l exists
continuous variables:
S_l=capacity of pool l
A_i=capacity of pool i

Second stage variables

binary variables:
$u_{i\omega}^c$=whether contract c is selected in purchasing feed i in scenario ω
continuous variables:
$y_{lj\omega}$=flow from pool l to product j in scenario ω
$z_{ij\omega}$=flow of feed i to product j in scenario ω
$q_{il\omega}$=proportion of flow from input i to pool l in scenario ω
$CT_{i\omega}$=cost of purchasing feed i in scenario ω
$CT_{i\omega}^c$=cost of purchasing feed i under contract c in scenario ω
$B_{i\omega}^c$=the amount of feed i purchased under contract c in scenario ω

We study pooling problem with purchase contracts under price uncertainty. The first stage decisions include whether to install the feeds and pools, the capacity of the feeds and pools, whether the feeds exist.

$$A_i^L \lambda_i \leq A_i \leq A_i^U \lambda_i \quad \forall i \tag{10}$$

$$S_l^L \theta_l \leq S_l \leq S_l^U \theta_l \quad \forall l \tag{11}$$

After the pools are installed, the uncertainties in the prices of the feeds and products are realized. We need to decide the material flows just as in a standard pooling problem. Equations (12)-(20) are the equations in the pq-formulation of the pooling problem.

$$\sum_{l:(i,l)\in T_X} \sum_{j:(l,j)\in T_Y} q_{il\omega} y_{lj\omega} + \sum_{j:(i,j)\in T_Z} z_{ij\omega} \leq A_i \quad \forall i, \omega \tag{12}$$

$$\sum_{j:(l,j)\in T_Y} y_{lj\omega} \le S_l \quad \forall l,\omega \tag{13}$$

$$\sum_{l:(l,j)\in T_Y} y_{lj\omega} + \sum_{i:(i,j)\in T_Z} z_{ij\omega} \le D^U_{j\omega} \quad \forall j,\omega \tag{14}$$

$$\sum_{i:(i,l)\in T_X} q_{il\omega} = \theta_l \quad \forall i,\omega \tag{15}$$

$$\begin{aligned} P^L_{jk}\Big(\sum_{l:(l,j)\in T_Y} y_{lj\omega} + \sum_{i:(i,j)\in T_Z} z_{ij\omega}\Big) \le \sum_{l:(l,j)\in T_Y}\sum_{i:(i,l)\in T_X} C_{ik} q_{il\omega} y_{lj\omega} + \sum_{i:(i,j)\in T_Z} C_{ik} z_{ij\omega} \\ \le P^U_{jk}\Big(\sum_{l:(l,j)\in T_Y} y_{lj\omega} + \sum_{i:(i,j)\in T_Z} z_{ij\omega}\Big) \quad \forall j,k,\omega \end{aligned} \tag{16}$$

$$0 \le q_{il\omega} \le \lambda_i \quad \forall (i,l)\in T_X,\omega \tag{17}$$

$$0 \le y_{lj\omega} \le \min\{S_l, D^U_j, \sum_{i:(i,l)\in T_X} A^U_i\} \quad \forall (l,j)\in T_Y \tag{18}$$

$$0 \le z_{ij\omega} \le \min\{A^U_i, D^U_j\} \quad \forall (i,j)\in T_Z,\omega \tag{19}$$

$$\sum_{i:(i,l)\in T_X} q_{il\omega} y_{lj\omega} = y_{lj\omega} \quad \forall l,j,\omega \tag{20}$$

In this case study, we also consider different types of purchase contracts when the prices are realized. The formulation of contracts is based Park et al. [32]. (21)-(24) decides that the total amount of feed i purchased can come from exactly one contract.

$$\sum_{l:(i,l)\in T_X}\sum_{j:(l,j)\in T_Y} q_{il\omega} y_{lj\omega} + \sum_{j:(i,j)\in T_Z} z_{ij\omega} = \sum_c B^c_{i\omega} \quad \forall i,\omega \tag{21}$$

$$A^L_i u^c_{i\omega} \le B^c_{i\omega} \le A^U_i u^c_{i\omega} \quad \forall i,\omega,c \tag{22}$$

$$\sum_c u^c_{i\omega} \le \lambda_i \quad \forall i,\omega \tag{23}$$

$$CT_{i\omega} = \sum_c CT^c_{i\omega} \quad \forall i,\omega \tag{24}$$

(25) describes fixed price contract.

$$CT^f_{i\omega} = \psi^f_{i\omega} B^f_{i\omega} \quad \forall i,\omega \tag{25}$$

(26)-(31) describe discount after a certain amount contract.

$$CT^d_{i\omega} = \psi^{d1}_{i\omega} B^{d1}_{i\omega} + \psi^{d2}_{i\omega} B^{d2}_{i\omega} \quad \forall i,\omega \tag{26}$$

$$B^d_{i\omega} = B^{d1}_{i\omega} + B^{d2}_{i\omega} \quad \forall i,\omega \tag{27}$$

$$B^{d1}_{i\omega} = B^{d11}_{i\omega} + B^{d12}_{i\omega} \quad \forall i, \omega \tag{28}$$

$$0 \leq B^{d11}_{i\omega} \leq \sigma^{d}_{i\omega} u^{d1}_{i\omega} \quad \forall i, \omega \tag{29}$$

$$B^{d12}_{i\omega} = \sigma^{d}_{i\omega} u^{d2}_{i\omega} \quad \forall i, \omega \tag{30}$$

$$0 \leq B^{d2}_{i\omega} \leq A^{U}_{i} u^{d2}_{i\omega} \quad \forall i, \omega \tag{31}$$

(32)-(37) describe bulk discount contract.

$$CT^{b}_{i\omega} = \psi^{b1}_{i\omega} B^{b1}_{i\omega} + \psi^{b2}_{i\omega} B^{b2}_{i\omega} \quad \forall i, \omega \tag{32}$$

$$B^{b}_{i\omega} = B^{b1}_{i\omega} + B^{b2}_{i\omega} \quad \forall i, \omega \tag{33}$$

$$0 \leq B^{b1}_{i\omega} \leq \sigma^{b}_{i\omega} u^{b1}_{i\omega} \quad \forall i, \omega \tag{34}$$

$$\sigma^{b}_{i\omega} u^{b2}_{i\omega} \leq B^{b2}_{i\omega} \leq A^{U}_{i} u^{b2}_{i\omega} \quad \forall i, \omega \tag{35}$$

$$u^{b1}_{i\omega} + u^{b2}_{i\omega} = u^{b}_{i\omega} \quad \forall i, \omega \tag{36}$$

Because of the decisions on purchase contracts, there are binary variables in the second stage decisions.

$$u^{c}_{i\omega}, u^{d1}_{i\omega}, u^{d2}_{i\omega}, u^{b1}_{i\omega}, u^{b2}_{i\omega} \in \{0, 1\} \tag{37}$$

The objective includes the fixed and variable cost of installing the pools and fixed cost of the feeds, purchase of feeds, and sales of final products.

$$\min \sum_{l} \big(\alpha_l \theta_l + \beta_l S_l\big) + \sum_{i} \big(\delta_i \lambda_i + \xi_i A_i\big) + \sum_{\omega} \tau_\omega \Big(\sum_{i} CT_{i\omega} - \sum_{j} d_j \big(\sum_{l:(l,j)\in T_Y} y_{lj\omega} + \sum_{i:(i,j)\in T_Z} z_{ij\omega} \big) \Big) \tag{38}$$

References

[1] Source code of PlasmoAlgorithms. https://github.com/bbrunaud/PlasmoAlgorithms.jl. Accessed: 2018-11-21

[2] Achterberg, T.: SCIP: solving constraint integer programs. Mathematical Programming Computation **1**(1), 1–41 (2009)

[3] Ahmed, S., Garcia, R.: Dynamic capacity acquisition and assignment under uncertainty. Annals of Operations Research **124**(1-4), 267–283 (2003)

[4] Alonso-Ayuso, A., Escudero, L.F., Ortuno, M.T.: Bfc, a branch-and-fix coordination algorithmic framework for solving some types of stochastic pure and mixed 0–1 programs. European Journal of Operational Research **151**(3), 503–519 (2003)

[5] Angulo, G., Ahmed, S., Dey, S.S.: Improving the integer L-shaped method. INFORMS Journal on Computing **28**(3), 483–499 (2016)

[6] Anstreicher, K.M.: Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. Journal of Global Optimization **43**(2-3), 471–484 (2009)

[7] Atakan, S., Sen, S.: A progressive hedging based branch-and-bound algorithm for mixed-integer stochastic programs. Computational Management Science pp. 1–40 (2018)

[8] Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0–1 programs. Mathematical programming **58**(1-3), 295–324 (1993)

[9] Balas, E., Ceria, S., Cornuéjols, G., Natraj, N.: Gomory cuts revisited. Operations Research Letters **19**(1), 1–9 (1996)

[10] Cao, Y., Zavala, V.M.: A scalable global optimization algorithm for stochastic nonlinear programs. Under Review (2017)

[11] Carøe, C.C., Schultz, R.: Dual decomposition in stochastic integer programming. Operations Research Letters **24**(1-2), 37–45 (1999)

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

[13] Horst, R., Tuy, H.: Global optimization: Deterministic approaches. Springer Science & Business Media (2013)

[14] Jalving, J., Abhyankar, S., Kim, K., Hereld, M., Zavala, V.M.: A graph-based computational framework for simulation and optimisation of coupled infrastructure networks. IET Generation, Transmission & Distribution **11**(12), 3163–3176 (2017)

[15] Kannan, R.: Algorithms, analysis and software for the global optimization of two-stage stochastic programs. Ph.D. thesis, Massachusetts Institute of Technology (2018)

[16] Kılınç, M.R., Linderoth, J., Luedtke, J.: Lift-and-project cuts for convex mixed integer nonlinear programs. Mathematical Programming Computation **9**(4), 499–526 (2017)

[17] Küçükyavuz, S., Sen, S.: An introduction to two-stage stochastic mixed-integer programming. In: Leading Developments from INFORMS Communities, pp. 1–27. INFORMS (2017)

[18] Laporte, G., Louveaux, F.V.: The integer L-shaped method for stochastic integer programs with complete recourse. Operations Research Letters **13**(3), 133–142 (1993)

[19] Li, C., Grossmann, I.E.: A finite ϵ-convergence algorithm for two-stage convex 0-1 mixed-integer nonlinear stochastic programs with mixed-integer first and second stage variables. Optimization Online (2018)

[20] Li, C., Grossmann, I.E.: An improved L-shaped method for two-stage convex 0-1 mixed integer nonlinear stochastic programs. Computers & Chemical Engineering **112**, 165 – 179 (2018)

[21] Li, X., Armagan, E., Tomasgard, A., Barton, P.I.: Stochastic pooling problem for natural gas production network design and operation under uncertainty. AIChE Journal **57**(8), 2120–2135 (2011)

[22] Li, X., Chen, Y., Barton, P.I.: Nonconvex generalized benders decomposition with piecewise convex relaxations for global optimization of integrated process design and operation problems. Industrial & Engineering Chemistry Research **51**(21), 7287–7299 (2012)

[23] Li, X., Tomasgard, A., Barton, P.I.: Nonconvex generalized benders decomposition for stochastic separable mixed-integer nonlinear programs. Journal of Optimization Theory and Applications **151**(3), 425 (2011)

[24] Li, X., Tomasgard, A., Barton, P.I.: Decomposition strategy for the stochastic pooling problem. Journal of Global Optimization **54**(4), 765–790 (2012)

[25] Mijangos, E.: An algorithm for two-stage stochastic mixed-integer nonlinear convex problems. Annals of Operations Research **235**(1), 581–598 (2015)

[26] Misener, R., Floudas, C.A.: Antigone: algorithms for continuous/integer global optimization of nonlinear equations. Journal of Global Optimization **59**(2-3), 503–526 (2014)

[27] Ntaimo, L.: Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. Operations research **58**(1), 229–243 (2010)

[28] Ntaimo, L., Sen, S.: The million-variable march for stochastic combinatorial optimization. Journal of Global Optimization **32**(3), 385–400 (2005)

[29] Ntaimo, L., Tanner, M.W.: Computations with disjunctive cuts for two-stage stochastic mixed 0-1 integer programs. Journal of Global Optimization **41**(3), 365–384 (2008)

[30] Ogbe, E., Li, X.: A joint decomposition method for global optimization of multiscenario nonconvex mixed-integer nonlinear programs. arXiv preprint arXiv:1802.07342 (2018)

[31] Oliveira, F., Gupta, V., Hamacher, S., Grossmann, I.E.: A lagrangean decomposition approach for oil supply chain investment planning under uncertainty with risk considerations. Computers & Chemical Engineering **50**, 184–195 (2013)

[32] Park, M., Park, S., Mele, F.D., Grossmann, I.E.: Modeling of purchase and sales contracts in supply chain optimization. Industrial & Engineering Chemistry Research **45**(14), 5013–5026 (2006)

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

[34] Rebennack, S., Kallrath, J., Pardalos, P.M.: Optimal storage design for a multi-product plant: A non-convex minlp formulation. Computers & chemical engineering **35**(2), 255–271 (2011)

[35] Ryan, S.M., Wets, R.J.B., Woodruff, D.L., Silva-Monroy, C., Watson, J.P.: Toward scalable, parallel progressive hedging for stochastic unit commitment. In: Power and Energy Society General Meeting (PES), 2013 IEEE, pp. 1–5. IEEE (2013)

[36] Sen, S., Sherali, H.D.: Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. Mathematical Programming **106**(2), 203–223 (2006)

[37] Sherali, H.D., Adams, W.P.: A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM Journal on Discrete Mathematics **3**(3), 411–430 (1990)

[38] Sherali, H.D., Fraticelli, B.M.: A modification of Benders decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. Journal of Global Optimization **22**(1-4), 319–342 (2002)

[39] Sherali, H.D., Zhu, X.: On solving discrete two-stage stochastic programs having mixed-integer first-and second-stage variables. Mathematical Programming **108**(2), 597–616 (2006)

[40] Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Mathematical Programming **103**, 225–249 (2005)

[41] Van Slyke, R.M., Wets, R.: L-shaped linear programs with applications to optimal control and stochastic programming. SIAM Journal on Applied Mathematics **17**(4), 638–663 (1969)

[42] Yang, Y., Barton, P.I.: Integrated crude selection and refinery optimization under uncertainty. AIChE journal **62**(4), 1038–1053 (2016)

[43] Zhang, M., Küçükyavuz, S.: Finitely convergent decomposition algorithms for two-stage stochastic pure integer programs. SIAM Journal on Optimization **24**(4), 1933–1951 (2014)