Noname manuscript No.
(will be inserted by the editor)

A finite ϵ-convergence algorithm for two-stage convex 0-1 mixed-integer nonlinear stochastic programs with mixed-integer first and second stage variables

Can Li · Ignacio E. Grossmann

Received: date / Accepted: date

Abstract In this paper, we propose a generalized Benders decomposition-based branch and bound algorithm, GBDBAB, to solve two-stage convex 0-1 mixed-integer nonlinear stochastic programs with mixed-integer variables in both first and second stage decisions. In order to construct the convex hull of the MINLP subproblem for each scenario in closed-form, we first represent each MINLP subproblem as a generalized disjunctive program (GDP) in conjunctive normal form (CNF). Second, we apply basic steps to convert the CNF of the MINLP subproblem into disjunctive normal form (DNF) to obtain the convex hull of the MINLP subproblem. We prove that GBD is able to converge for the problems with pure binary variables given that the convex hull of each subproblem is constructed in closed-form. However, for problems with mixed-integer first and second stage variables, we propose an algorithm, GBDBAB, where we may have to branch and bound on the continuous first stage variables to obtain the optimal solution. We prove that the algorithm GBDBAB can converge to ϵ-optimality in a finite number of steps. Since constructing the convex hull can be expensive, we propose a sequential convexification scheme that progressively applies basic steps to the CNF. Computational results demonstrate the effectiveness of the algorithm.

Keywords Stochastic programming · Integer recourse · Generalized Benders decomposition · Branch and bound

1 Introduction

Stochastic programming provides a mathematical framework to model decision-making under uncertainty [4]. The uncertain parameters are usually repre-

Ignacio E. Grossmann
Department of Chemical Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA
E-mail: grossmann@cmu.edu

sented by discrete probability distributions, which are assumed to be known *a priori*. Different realizations of the uncertain parameters are represented by the set of scenarios $\omega \in \Omega$, each of which has an associated probability τ_ω.

Two-stage stochastic programming problem, a special case of stochastic programming, can be computationally expensive when the number of scenarios becomes large. Therefore, decomposition algorithms, like (generalized) Benders decomposition [15; 33] and Lagrangean decomposition [16], and progressive hedging [26] have been proposed to solve the two-stage stochastic programs by decomposing the fullspace problem into scenarios. However, for two-stage mixed-integer stochastic programs with mixed-integer recourse, Benders decomposition cannot be applied directly given the nonconvex nature of the subproblems. Although Lagrangean decomposition [18] and progressive hedging [13; 36; 37] have been used to solve stochastic programs with mixed-integer recourse, they are not guaranteed to obtain the optimal solution because of the duality gap. Given that there are engineering applications that involve mixed-integer second stage decisions, such as unit commitment problem [17] and planning problems with discrete transportation costs [5], decomposition algorithms that can deal with mixed-integer recourse are of interest.

In the past two decades, some advances have been made in the algorithms to solve stochastic mixed-integer linear programs with (mixed) integer recourse. Laporte and Louveaux [20] propose a Benders-like algorithm with optimality cuts to solve two-stage stochastic programs with mixed-integer recourse and pure binary first stage variables. Ahmed et al. [1] propose a branch-and-bound algorithm for two-stage stochastic integer programs with mixed-integer first stage variables and pure integer second stage variables. Carøe and Schultz [8] propose a dual decomposition algorithm with branch-and-bound that is able to close the duality gap of Lagrangean relaxation. In order to apply Benders decomposition to solve the problems with mixed-integer recourse, decomposition algorithms with convexification schemes, such as lift-and-project cuts [1; 25; 29; 30], reformulation linearization technique (RLT) [31; 32], Gomory cuts [14], have also been proposed. For a more comprehensive review of the algorithmic advances in stochastic mixed-integer linear programs, we refer to Küçükyavuz and Sen [19].

For mixed-integer nonlinear stochastic programs, relatively little work has been done. Mijangos [24] proposes an algorithm based on Branch-and-Fix Coordination method for convex problems with 0-1 mixed-integer variables in the first stage and only continuous variables in the second stage. Atakan and Sen [2] propose a progressive hedging-based branch-and-bound algorithm for convex mixed-integer stochastic programs, which works well in the case of pure binary first stage variables and continuous recourse. Li et al. [22] propose a nonconvex generalized Benders decomposition (NGBD) algorithm for mixed-integer nonlinear stochastic programs with pure binary first stage variables.

In this paper, we propose a GBD-based algorithm that can solve two-stage convex 0-1 mixed-integer nonlinear stochastic programs with mixed-integer first and second stage variables with finite ϵ-convergence. This work is inspired by Sherali and Zhu [32] who propose a Benders decomposition algorithm to

solve two-stage mixed-integer linear stochastic programs having mixed-integer first and second stage variables. The authors use reformulation linearization technique (RLT) to convexify the original MILP subproblems. In this paper, we try to generalize the algorithm proposed by Sherali and Zhu [32] to convex nonlinear problems. Due to recent advances in generalized disjunctive programming (GDP), especially the work of Ruiz and Grossmann [27], we now have a procedure to obtain the convex hull of the feasible region of a convex MINLP. By applying the convexification scheme proposed in [27], we are able to develop a GBD-based algorithm with finite ϵ-convergence.

2 Background

In this section, we provide some background on disjunctive convex programming and basic steps [27], which are used to construct the convex hull of the MINLP subproblem in closed-form.

A convex set C can be defined as C=$\{x \in \mathbb{R}^n | \phi(x) \leq 0\}$, where $\phi(x) : \mathbb{R}^n \rightarrow \mathbb{R}^1$ is a convex function. Given a collection of sets such that C_j=$\{x \in \mathbb{R}^n | \phi_j(x) \leq 0\}$, $j \in M$, the union of the convex sets, which is an elementary disjunctive set, is defined as:

$$H = \bigcup_{j \in M} C_j = \{x \in \mathbb{R}^n | \bigvee_{j \in M} \phi_j(x) \leq 0\} \tag{1}$$

and the intersection of convex sets can be expressed as:

$$P = \bigcap_{j \in M} C_j = \{x \in \mathbb{R}^n | \bigwedge_{j \in M} \phi_j(x) \leq 0\} \tag{2}$$

A disjunctive set can be expressed in different forms. Two extreme cases are the conjunctive normal form (CNF) and the disjunctive normal form (DNF). The CNF is the intersection of some elementary disjunctive sets:

$$F_{CNF} = \bigcap_{i \in T} H_i \tag{3}$$

where each H_i is an elementary disjunctive set. The DNF is the union of some convex sets:

$$F_{DNF} = \bigcup_{i \in D} P_i \tag{4}$$

where each set P_i is a convex set defined by convex functions ϕ_j, $j \in M_i$. P_i can also be represented in a more succinct form $P_i = \{x \in \mathbb{R}^n | g_i(x) \leq 0\}$, where $g_i(x)$: $\mathbb{R}^n \rightarrow \mathbb{R}^m$.

A regular form (RF) between the CNF and DNF can be represented as:

$$F_{RF} = \bigcap_{k \in K} S_k \tag{5}$$

where $S_k = \cup_{i \in D_k} P_i$. Each S_k is called a conjunct, which is a disjunction over some convex sets P_i, $i \in D_k$. Balas [3] proposes an operation called basic step, which when applied to the regular form can reduce the number of conjuncts

by one in the linear case. The basic step is extended by Ruiz and Grossmann [27] to convex nonlinear case. Here, we rewrite Theorem 2.1 in [27], which is an extension of the theorem in Balas [3] to show how a CNF can be transformed into a DNF.

Theorem 1 *Let F_{RF} be a disjunctive set in regular form. Then F_{RF} can be brought to DNF by $|K|-1$ recursive applications of the following basic step which preserves regularity:*

For some $r, s \in K$, bring $S_r \cap S_s$ to DNF by replacing it with:
$S_{rs} = \bigcup_{i \in D_r, j \in D_s} (P_i \cap P_j)$

Theorem 1 will be used to construct the convex hull the MINLP subproblem in our proposed algorithm.

3 Overview of basic ideas

The goal of this paper is to design a decomposition algorithm that has finite ϵ-convergence for two-stage convex 0-1 mixed-integer nonlinear stochastic programs with mixed-integer first and second stage variables. While GBD cannot be applied to problems with mixed-integer recourse directly, it has finite ϵ-convergence for problems with convex constraints and continuous recourse variables as proved by Geoffrion [15]. Therefore, we propose to construct the convex hull for the integer-constrained subproblems by applying basic steps so that GBD can be applied to solve the relaxed problem. However, the relaxed problem is not equivalent to the original problem if we have mixed-integer first stage variables. Therefore, we first prove that the relaxed problem is equivalent to the original fullspace problem for the special case of pure binary first stage variables. Based on the theoretical results of the problems with pure binary variables, we propose a GBD-based branch and bound algorithm for the general problems with mixed-integer first and second stage variables where we may branch on the continuous first stage variables. The details of the algorithm are described in section 4.

However, constructing the convex hull by applying all the possible basic steps for each subproblem can be expensive, especially when we have a large number of binary variables in the second stage. Therefore, in section 5, we describe a sequential convexification scheme, which is a more limited way for convexifying the subproblem to avoid applying all the basic steps when certain sufficient conditions are satisfied. Computational results of the proposed GBDBAB algorithm are shown in section 6 with one small illustrative example and two process systems engineering applications.

4 GBD-based branch and bound algorithm

In this paper, we consider two-stage convex 0-1 mixed-integer nonlinear stochastic programs with mixed-integer first and second stage variables. The problem

is formally defined by (6)-(10).

$$(P) \quad \min \quad c^T x + \sum_{\omega \in \Omega} \tau_\omega d_\omega^T y_\omega \tag{6}$$

$$A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{7}$$

$$A_{1,\omega} x + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \quad \forall \omega \in \Omega \tag{8}$$

$$x \in X, \quad X = \left\{x : x_i \in \{0,1\}, \forall i \in I_1, \;\; 0 \leq x \leq x^{ub}\right\} \tag{9}$$

$$y_\omega \in Y \quad \forall \omega \in \Omega, \quad Y = \left\{y : y_j \in \{0,1\}, \forall j \in J_1, \;\; 0 \leq y \leq y^{ub}\right\} \tag{10}$$

Here, x represents the first stage decisions. y_ω represents the second stage decisions in scenario ω. g_0, $g_{1,\omega}$, are smooth convex functions. Both the first and the second stage decisions are mixed-integer. 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. In this paper, we assume 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. The feasible region of (P) is assumed to be compact.

As (P) has mixed-integer recourse, generalized Benders decomposition (GBD) [15] cannot be applied to solve it directly as in the case of convex continuous recourse. Therefore, we construct convex relaxations of (P) so that GBD can be applied to solve the problem.

We first define the set,

$$S_\omega = \left\{(x, y_\omega | A_{1,\omega} x + g_{1,\omega}(y_\omega) \leq b_{1,\omega}, y_\omega \in Y, 0 \leq x \leq x^{ub}\right\} \tag{11}$$

where the second stage constraints and the upper and lower bound constraints for the first stage decisions are included for each scenario ω. The fullspace problem with (x, y_ω) in the convex hull of S_ω for all $\omega \in \Omega$ is defined as (PC):

$$(PC) \quad \min \quad c^T x + \sum_{\omega \in \Omega} \tau_\omega d_\omega^T y_\omega \tag{12}$$

$$A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{13}$$

$$x \in X \tag{14}$$

$$(x, y_\omega) \in conv(S_\omega) \quad \forall \omega \in \Omega \tag{15}$$

Since the integrality constraints in the second stage are relaxed, (PC) is a relaxation of (P). Specifically, $conv(S_\omega)$ can be characterized by applying the hierarchy of relaxations for nonlinear convex generalized disjunctive programs proposed by Ruiz and Grossmann [27].

As a subset of the second stage variables y_ω are binary, i.e., $(y_\omega)_j \in \{0,1\}$, $\forall j \in J_1$, the set S_ω is equivalent to the disjunctions in (16).

$$\begin{bmatrix} A_{1,\omega}x + g_{1,\omega}(y_\omega) \le b_{1,\omega} \\ 0 \le x \le x^{ub} \\ 0 \le y_\omega \le y^{ub} \\ (y_\omega)_j = 1 \end{bmatrix} \vee \begin{bmatrix} A_{1,\omega}x + g_{1,\omega}(y_\omega) \le b_{1,\omega} \\ 0 \le x \le x^{ub} \\ 0 \le y_\omega \le y^{ub} \\ (y_\omega)_j = 0 \end{bmatrix} \quad \forall j \in J_1 \qquad (16)$$

We represent (16) as $S^1_{\omega j} \cup S^0_{\omega j}$, $\forall j \in J_1$. (16) is a conjunctive normal form (CNF) representation of S_ω where S_ω is represented as a conjunction over the disjunctions, i.e., $S_\omega = \cap_{j\in J_1}(S^1_{\omega j} \cup S^0_{\omega j})$. The hull relaxation of the CNF, $h - rel\big(\cap_{j\in J_1}(S^1_{\omega j} \cup S^0_{\omega j})\big)$, is equivalent to $\cap_{j\in J_1} conv(S^1_{\omega j} \cup S^0_{\omega j})$. We know that $\cap_{j\in J_1} conv(S^1_{\omega j} \cup S^0_{\omega j}) \subset conv(\cap_{j\in J_1} S^1_{\omega j} \cup S^0_{\omega j}) = conv(S_\omega)$. Therefore, the hull relaxation of the CNF is a relaxation of $conv(S_\omega)$. In order to construct $conv(S_\omega)$, we apply the basic step proposed by Balas [3] to the CNF (16). Based on the Theorem 1, the CNF can be converted to the following DNF shown in (17),

$$\vee_{r\in R} \begin{bmatrix} A_{1,\omega}x + g_{1,\omega}(y_\omega) \le b_{1,\omega} \\ 0 \le x \le x^{ub} \\ 0 \le y_\omega \le y^{ub} \\ (y_\omega)_j = e_{rj} \quad \forall j \in J_1 \end{bmatrix} \qquad (17)$$

where R is the set that corresponds to all the possible combinations of the binary variables $(y_\omega)_j$, $\forall j \in J_1$. For each $r \in R$, e_{rj} can be either zero or one. There are $2^{|J_1|}$ components in the set R. Ceria and Soares [9] proved that the convex hull of the DNF (17) can be expressed as the projection of a higher dimensional convex set where (x, y_ω) has to satisfy the following constraints:

$$\begin{aligned} & x = \sum_{r\in R} u^r_\omega \\ & y_\omega = \sum_{r\in R} v^r_\omega \\ & \sum_{r\in R} \gamma^r_\omega = 1, \quad 0 \le \gamma^r_\omega \le 1, \quad \forall r \in R \\ & A_{1,\omega}u^r_\omega + \gamma^r_\omega g_{1,\omega}(v^r_\omega/\gamma^r_\omega) \le b_{1,\omega}\gamma^r_\omega, \quad \forall r \in R \\ & 0 \le u^r_\omega \le x^{ub}\gamma^r_\omega, \quad \forall r \in R \\ & 0 \le v^r_\omega \le y^{ub}\gamma^r_\omega, \quad \forall r \in R \\ & (v_\omega)_j = e_{rj}\gamma^r_\omega \quad \forall j \in J_1, r \in R \end{aligned} \qquad (18)$$

According to Ruiz and Grossmann [27], (x, y_ω) is in $conv(S_\omega)$ if and only if it satisfies the constraints in (18). Since the perspective function of $g_{1,\omega}$ used in (18) can give rise to numerical difficulties at $\gamma^r_\omega = 0$, the approximation proposed by Furman et al. [12] is used in our implementation,

$$\gamma^r_\omega g_{1,\omega}(v^r_\omega/\gamma^r_\omega) \approx \big((1-\kappa)\gamma^r_\omega + \kappa\big) g_{1,\omega}\Big(\frac{v^r_\omega}{(1-\kappa)\gamma^r_\omega + \kappa}\Big) - \kappa g_{1,\omega}(0)(1-\gamma^r_\omega) \qquad (19)$$

where κ is a small number like 10^{-5}. We should note that this approximation is exact at $\gamma_\omega^r = 0$ and $\gamma_\omega^r = 1$. As we are able to construct $conv(S_\omega)$ for all $\omega \in \Omega$ in closed-form, GBD can be applied to solve (PC) to ϵ-optimality.

We briefly describe how GBD can be used to solve (PC), the reader can refer to Geoffrion [15] for details. The basic idea of GBD is to decompose (P) into a Benders master problem which only contains first stage decisions and Benders subproblems each of which only contains second stage decisions for a given scenario.

We first define the Benders master problem (MB^h) at iteration h of the GBD algorithm. (22) are Benders cuts that are generated up until iteration h, which are derived by solving the Benders subproblems (SB_ω^h). Benders master problem is a relaxation of (PC) projected on the x space. Therefore, a lower bound of (PC) is obtained by solving the MINLP Benders master problem:

$$(MB^h): \quad \min \quad z_{MB}^h = \sum_\omega \eta_\omega \tag{20}$$

$$s.t. \quad A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{21}$$

$$\eta_\omega \geq z_{SB,\omega}^{*h'} + (\lambda_\omega^{h'})^T (x - \tilde{x}^{h'}) + \tau_\omega c^T x \quad \forall \omega \in \Omega, h' \leq h \tag{22}$$

$$x \in X, \quad \eta_\omega \in \mathbb{R}^1, \quad \forall \omega \in \Omega \tag{23}$$

Assume that the solution to the Benders master problem at iteration h is $\tilde{x}^h$, x is fixed at $\tilde{x}^h$ in (PC). The rest of the problem can be decomposed into Benders subproblems (SB_ω^h).

$$(SB_\omega^h): \quad \min \quad z_{SB,\omega}^h = \tau_\omega d_\omega^T y_\omega \tag{24}$$

$$s.t. \quad x = \tilde{x}^h \tag{25}$$

$$(x, y_\omega) \in conv(S_\omega) \tag{26}$$

A Benders cut can be generated at iteration h:

$$\eta_\omega \geq z_{SB,\omega}^{*h} + (\lambda_\omega^h)^T (x - \tilde{x}^h) + \tau_\omega c^T x \tag{27}$$

where λ_ω^h are the optimal dual multipliers for constraints (25), $z_{SB,\omega}^{*h}$ is the optimal objective value of (SB_ω^h). A feasible solution to (PC) is obtained at the optimal solution of the Benders subproblem. After solving the Benders subproblems at iteration h, GBD will add the newly generated Benders cuts to the Benders master problem and keep iterating between the Benders master problem and the Benders subproblems until (PC) is solved to ϵ-optimality.

Although GBD can be applied to solve (PC), for the problem with both binary and continuous first stage variables, Sherali and Zhu [32] showed that in the MILP setting (PC) is not always equivalent to (P), since for fixed first stage decision $\bar{x}$, the extreme points of $conv(S_\omega) \cap \{(x, y) : x = \bar{x}\}$ will not always have binary values for $y_j, \forall j \in J_1$. As convex MINLP is a generalization of MILP, it is straightforward to show that (PC) is not always equivalent to (P).

However, it also shown in Sherali and Zhu [32] that in the MILP setting, under the restriction of pure binary first stage variables, (PC) and (P) are equivalent. Therefore, before we consider the proposed GBD-based algorithm to solve problem (P) with both binary and continuous first stage variables, the equivalence of (P) and the corresponding (PC) with pure binary first stage variables is proved in Proposition 1. The convergence of the GBD-based algorithm for the problem with mixed-integer first stage variables will be proved later based on Proposition 1 and Corollary 1.

Proposition 1 *Consider a special case of (P) where the first stage variables are all binary. We assume that in the corresponding problem (PC), the convex hull of S_ω is expressed in the form of (18). Then (P) and (PC) are equivalent in the sense that they have the same optimal objective value and the optimal solution of (P) can always be obtained based on the optimal solution of (PC).*

Proof Note that (PC) is a relaxation of (P). Hence the optimal objective value of (PC) should be less than or equal to (P). Let $(x^*, y^*_{\omega_1}, y^*_{\omega_2}, \cdots, y^*_{\omega_{|\Omega|}})$ be the optimal solution of (PC). Now if we select any scenario ω, we have $(x^*, y^*_\omega) \in conv(S_\omega)$. For this scenario ω, we have $x^* = \sum_{r\in R} u^{r*}_\omega$ and $y^*_\omega = \sum_{r\in R} v^{r*}_\omega$. As we have assumed that x are pure binary variables and we have $0 \le u^{r*}_\omega \le e\gamma^{r*}_\omega$, for any $(x^*)_i = 1$, we must have $(u^{r*}_\omega)_i = \gamma^{r*}_\omega$, otherwise $\sum_{r\in R}(u^{r*}_\omega)_i$ will not sum up to 1. For any $(x^*)_i = 0$, $(u^{r*}_\omega)_i$ must be zero for all r. Therefore, for all $\gamma^{r*}_\omega > 0$, we have $u^{r*}_\omega/\gamma^{r*}_\omega = x^*$. Furthermore, for all $\gamma^{r*}_\omega > 0$, $d^T v^{r*}_\omega/\gamma^{r*}_\omega = d^T y^*_\omega$. Otherwise, there must exist r' and r'' such that $d^T v^{r'*}_\omega/\gamma^{r'*}_\omega > d^T y^*_\omega$ and $d^T v^{r''*}_\omega/\gamma^{r''*}_\omega < d^T y^*_\omega$. Then $v^{r''*}_\omega/\gamma^{r''*}_\omega$ satisfies all the constraints of (PC) and leads to a lower objective value than y^*_ω, which contradicts with the fact that y^*_ω is the optimal solution. Therefore, we can select any $\gamma^{r*}_\omega > 0$, and let $y_\omega = v^{r*}_\omega/\gamma^{r*}_\omega$ to construct a feasible solution that yields the same cost in the objective as y^*_ω. We can follow the same procedure to construct the optimal solution of every scenario $\omega \in \Omega$. Note that the solution that we construct satisfies all the constraints of (P) and yields the same objective value as the optimal solution to (PC). In that sense, (PC) and (P) are equivalent.

The following theorem can be proved based on Proposition 1.

Theorem 2 *GBD can be applied to obtain the ϵ-optimal solution of problem (P) with pure binary first stage variables by solving its relaxation (PC).*

Proof As we have assumed that the feasible region of (P) is nonempty compact and the problem has relatively complete recourse, $conv(S_\omega)$ is nonempty compact convex set for any fixed x, s.t. $x \in X$, $A_0 x \ge b_0$, $g_0(x) \le 0$, for every $\omega \in \Omega$. It follows directly from Theorem 2.5 of Geoffrion [15] that GBD has finite ϵ-convergence when applied to solve (PC). It is proved in Proposition 1 that (P) and (PC) are equivalent for the problem with pure binary first stage variables. Therefore, GBD can be applied to obtain the ϵ-optimal solution of problem (P) with pure binary first stage variables by solving its relaxation (PC).

Now we go back to focus on the problem (P) with both binary and continuous first stage variables. A corollary can be derived based on Proposition 1.

Corollary 1 *For (PC) with both binary and continuous first stage variables, if the optimal first stage variables x^* to (PC) are all at their upper or lower bound, i.e., $(x^*)_i = 0$ or $(x^*)_i = (x^{ub})_i$, $\forall i \in I$, then (PC) and its corresponding (P) are equivalent in the sense that they have the same optimal objective value.*

Proof The proof is similar to the case where we have pure binary first stage variables. As $(x^*)_i = 0$ or $(x^*)_i = (x^{ub})_i$, $\forall i$, we must have $u_\omega^{r*}/\gamma_\omega^r = x^*$ for all $\gamma_\omega^{r*} > 0$. Therefore, we can construct a feasible solution to (P) by letting $y_\omega = v_\omega^{r*}/\gamma_\omega^{r*}$ where $\gamma_\omega^{r*} > 0$ for all $\omega \in \Omega$. The feasible solution yields the same objective value as the optimal solution to (PC).

If the condition of Corollary 1 does not hold, the equivalence of (PC) and (P) cannot be guaranteed as we have discussed. In order to understand why (PC) and (P) are not equivalent in the general case, we can determine if the proof of Proposition 1 still holds for the problem with mixed-integer first stage variables. As we construct the convex hull of each scenario separately, if we take $u_\omega^{r*}/\gamma_\omega^{r*}$ for some $\gamma_\omega^{r*} > 0$ for each scenario ω as we did in Proposition 1 and Corollary 1, it is possible that we fail to obtain a unique first stage decisions for all the scenarios if some $(x^*)_i$ is not at its lower or upper bound. The reason why this occurs is that for component i of x^* such that $(x)_i$ does not lie at its upper or lower bound, there may exist one scenario ω for which we have $(x^*)_i = \sum_{r\in R}(u_\omega^{r*})_i$ but there exist r', r'' with $\gamma_\omega^{r'*} > 0$, $\gamma_\omega^{r''*} > 0$, $(u_\omega^{r'*}/\gamma_\omega^{r'*})_i < (x^*)_i$, $(u_\omega^{r''*}/\gamma_\omega^{r''*})_i > (x^*)_i$. As $(u_\omega^{r*}/\gamma_\omega^{r*})_i$, $\gamma_\omega^{r*} > 0$, is not always equal to $(x^*)_i$, the uniqueness of x obtained by taking $u_\omega^{r*}/\gamma_\omega^{r*}$ for some $\gamma_\omega^{r*} > 0$ for each scenario ω cannot be guaranteed. However, if we branch on the continuous first stage variables x, for example, by forcing $(x)_i \geq x_i^*$ at a given branch-and-bound node and furthermore in the construction of convex hull $(u_\omega^r)_i \geq (x)_i^*\gamma_\omega^r$ are added, the solution where $(u_\omega^{r'*}/\gamma_\omega^{r'*})_i < (x^*)_i$ will be cut off.

Inspired by the work of Sherali and Zhu [32], we propose a spatial branch-and-bound algorithm where we branch on the continuous first stage variables. At each node q of the branch-and-bound tree. The following problem $(PCBAB_q)$ is solved,

$$(PCBAB_q) \quad \min \quad c^T x + \sum_{\omega\in\Omega} \tau_\omega d_\omega^T y_\omega \tag{28}$$

$$A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{29}$$

$$x_q^{lb} \leq x \leq x_q^{ub} \tag{30}$$

$$x \in X \tag{31}$$

$$(x, y_\omega) \in conv(S_\omega^q) \tag{32}$$

$$S_\omega^q = \left\{(x, y_\omega | A_{1,\omega} x + g_{1,\omega}(y_\omega) \leq b_{1,\omega}, y_\omega \in Y, x_q^{lb} \leq x \leq x_q^{ub}\right\} \tag{33}$$

Note that the only difference between S_ω^q and S_ω is that the first stage decisions x is constrained in $[x_q^{lb}, x_q^{ub}]$ in S_ω^q. As we assume that problem (P) has relatively complete recourse, $conv(S_\omega^q)$ is a nonempty compact convex set for any fixed x, s.t. $x \in X$, $x_q^{lb} \leq x \leq x_q^{ub}$, $A_0 x \geq b_0$, $g_0(x) \leq 0$, for every $\omega \in \Omega$. Therefore, at each node q of the branch-and-bound tree, GBD can be applied to solve $(PCBAB_q)$ to ϵ-optimality in the same way that we solve (PC). We denote the optimal objective of $(PCBAB_q)$ as $v(PCBAB_q)$. The optimal solution to $(PCBAB_q)$ is $(x_q^*, y_{q\omega_1}^*, y_{q\omega_2}^*, \cdots, y_{q\omega_{|\Omega|}}^*)$. A heuristic algorithm can be applied to find a feasible solution to problem (P), while x has to satisfy the upper and lower bound constraints at node q.

Heuristic 1: Fix the first stage decisions at the optimal solution of $(PCBAB_q)$, x_q^*. Solve the subproblems with integrality constraints in parallel with some MINLP solvers, such as DICOPT [34], Pajarito [23], SBB [6], AlphaECP [38]. Note that we assume relatively complete recourse. Therefore, a feasible solution to (P) can be obtained at node q, which is denoted as $(x_q^f, y_{q\omega_1}^f, y_{q\omega_2}^f, \cdots, y_{q\omega_{|\Omega|}}^f)$. We denote the objective of the feasible solution found at node q as $V(PCBAB_q)$. Note that $v(PCBAB_q)$ and $V(PCBAB_q)$ are valid lower and upper bounds for the original problem (P), respectively, if the domain of x is restricted to $[x_q^{lb}, x_q^{ub}]$. In the GBD-based branch and bound algorithm, if $V(PCBAB_q) \leq v(PCBAB_q) + \epsilon$, node q can be fathomed by optimality. We denote the global lower bound as v, the global upper bound as V. If $v(PCBAB_q) \geq V$, node q is fathomed by bound. Otherwise, we select one component of the continuous first stage variables to branch on. The following branching rule is used to select one component of the continuous first stage variable to branch on.

Branching rule A:

At node q, calculate the distance of each component of the optimal continuous first stage variables to its bounds at node q.

$$\sigma_i = min\{(x_q^*)_i - (x_q^{lb})_i, (x_q^{ub})_i - (x_q^*)_i\}, \quad \forall i \in I \backslash I_1 \tag{34}$$

The variable with maximum distance to its bounds is selected to branch on.

$$p = \arg\max_{i \in I \backslash I_1} \sigma_i \tag{35}$$

The node q is partitioned into two new nodes q_1, q_2. Constraints $(x_q^{lb})_p \leq (x)_p \leq (x_q^*)_p$ and $(x_q^*)_p \leq (x)_p \leq (x_q^{ub})_p$ are added to node q_1 and q_2 respectively.

The steps of the GBD-based branch-and-bound algorithm are outlined as follows.

Algorithm GBDBAB

Step 0: Initialization Step. Initialize the global upper bound $V = +\infty$, the global lower bound $v = -\infty$, the iteration counter $k = 1$, the list of active nodes $L_k = \{q^0\}$. Set $x_{q^0}^{lb} = 0$, $x_{q^0}^{ub} = x^{ub}$. Let $\epsilon \geq 0$ be a selected optimality tolerance. Use GBD to solve the problem at the root node q^0 to ϵ-optimality. Let $v = v(PCBAB_{q^0})$. Apply a heuristic algorithm like heuristic 1 to obtain a feasible solution to (P) denoted as $(x_{q^0}^f, y_{q^0}^f)$. Let the objective value of the

feasible solution at node q^0 be V_{q_0}. Set $V = V_{q^0}$. If $V \leq v + \epsilon$, stop. Otherwise, go to step 1.

Step 1: Branching Step. Select the node q in the active node list with the minimum lower bound $v(PCBAB_q)$. Branch on first stage variable $(x)_p$ according to the Branching rule A. Create two new nodes q_1, q_2 and add constraints $(x_q^{lb})_p \leq (x)_p \leq (x_q^*)_p$ and $(x_q^*)_p \leq (x)_p \leq (x_q^{ub})_p$ to node q_1 and q_2 respectively. Let $k = k + 1$. Delete node q in L_k and add node q_1, q_2 to L_k. Go to step 2.

Step 2: Bounding Step. Solve the two problems $(PCBAB_{q_1})$ and $(PCBAB_{q_2})$ using generalized Benders decomposition to ϵ-optimality respectively. Update the global lower bound v with $v = min_{q \in L_k} v(PCBAB_q)$. A heuristic algorithm is applied to obtain feasible solutions at node q_1 and q_2 that yield objective values of $V(PCBAB_{q_1})$ and $V(PCBAB_{q_2})$ respectively. Update the global upper bound V if $V(PCBAB_{q_1}) \leq V$ or $V(PCBAB_{q_2}) \leq V$. Go to Step 3.

Step 3: Fathoming Step. For the two new nodes q_i, $i = 1, 2$, fathom the node if $V(PCBAB_{q_i}) \leq v(PCBAB_{q_i}) + \epsilon$, $i = 1, 2$. Fathom any node q with $v(PCBAB_q) + \epsilon \geq V$. If the node list L_k becomes empty, stop. Otherwise go to step 1.

In order to prove the finite ϵ-convergence of our proposed GBDBAB algorithm we make the following assumption.

Assumption 1

$\forall \epsilon > 0, \exists \delta_\epsilon > 0$, such that for any node q with $\sigma_i \leq \delta_\epsilon, \forall i \in I \backslash I_1, V(PCBAB_q) \leq v(PCBAB_q) + \epsilon$, i.e., the node is fathomed by optimality if $\sigma_i \leq \delta_\epsilon$, $\forall i \in I \backslash I_1$. Here σ_i, $\forall i \in I \backslash I_1$, is defined in Branching rule A. $V(PCBAB_q)$ is obtained by heuristic 1.

As we have assumed that problem (P) is bounded, the change of the first stage variables can only have finite change of value in the objective. As $\delta_\epsilon \to 0$, node q can be fathomed by optimality based on Corollary 1. Therefore, Assumption 1 always holds for problems that are bounded.

Proposition 2 *If Assumption 1 holds, the algorithm GBDBAB has finite ϵ-convergence.*

Proof First we prove that for a given node q and any continuous first stage variables $(x)_i$ in node q that are constrained in the interval $\left[(x_q^{lb})_i, (x_q^{ub})_i\right]$ where $(x_q^{ub})_i - (x_q^{lb})_i \leq \delta_\epsilon$, variable i will not be branched on again if branching rule A is used. Assuming that variable i is branched on again when branching rule A is used, we have $\sigma_{i'} \leq \sigma_i \leq \delta_\epsilon, \forall i' \in I \backslash \{I_1 \cup i\}$. However, by Assumption 1, any node q with $\sigma_i \leq \delta_\epsilon$, $\forall i \in I \backslash I_1$,is fathomed by optimality. We reach a contradiction. Therefore, in the worst case, the domain of each continuous first stage variable will be partitioned into intervals with length δ_ϵ. There will be a finite number of hyperrectangle partitions for the domain of the continuous first stage variables $[0, x^{ub}]$, i.e., a finite number of nodes in the branch-and-bound tree. In each node, GBD can converge in a finite number of iterations

according to Geoffrion [15]. Thus, the algorithm GBDBAB is able to converge to ϵ-optimum in a finite number of iterations.

Remark 1 Here, in order to prove the convergence of the algorithm, we assume that the convex hull of S_ω^q is constructed by the hierarchy of relaxations proposed by Ruiz and Grossmann [27]. If there are only a few binary variables in the second stage, the representation of the convex hull is tractable. However, if the number of binary variables in the second stage is large, it is computationally expensive to represent the convex hull since there can be $2^{|J_1|}$ disjuncts. Therefore, a more limited sequential convexification approach should be applied, which will be discussed in section 5.

Remark 2 Note that at each branching step, the child nodes generated are restrictions of their parent node. The Benders cuts in the master problem of the parent node can by inherited by the child nodes, which will make the GBD algorithm for the child node require fewer iterations to converge compared to running it from scratch.

An illustrative example
The following problem is solved by GBDBAB as an illustrative example.

$$\min \quad x_1+x_2+3x_3+3x_4+\sum_{\omega=\omega_1,\omega 2} \tau_\omega\big(y_{1\omega}-12y_{2\omega}+100y_{3\omega}+3y_{4\omega}-3y_{5\omega}\big) \tag{36}$$

$$x_1 \le 4x_3, \quad x_2 \le 2x_4, \tag{37}$$

$$x_1, x_2 \ge 0 \quad x_3, x_4 \in \{0,1\} \tag{38}$$

$$y_{1\omega} \le x_1, \quad y_{2\omega} \le x_2 \quad \forall \omega = \omega_1, \omega_2 \tag{39}$$

$$(y_{1\omega}-3)^2 + (y_{2\omega}-2)^2 \le 1 + 16(1-y_{4\omega}) \quad \forall \omega = \omega_1, \omega_2 \tag{40}$$

$$(y_{1\omega}-1)^2 + y_{2\omega}^2 \le 1 + 16y_{4\omega} \quad \forall \omega = \omega_1, \omega_2 \tag{41}$$

$$y_{1\omega}^2 + (y_{2\omega}-1)^2 \le 1 + 16(1-y_{5\omega}) \quad \forall \omega = \omega_1, \omega_2 \tag{42}$$

$$(y_{1\omega}-4)^2 + (y_{2\omega}-1)^2 \le 1 + 16y_{5\omega} \quad \forall \omega = \omega_1, \omega_2 \tag{43}$$

$$y_{1\omega} + y_{2\omega} + y_{3\omega} \ge d_\omega \quad \forall \omega = \omega_1, \omega_2 \tag{44}$$

$$y_{1\omega}, y_{2\omega}, y_{3\omega} \ge 0, \quad y_{4\omega}, y_{5\omega} \in \{0,1\} \quad \forall \omega = \omega_1, \omega_2 \tag{45}$$

where x_1, x_2, x_3, x_4 are the first stage variables, $y_{1\omega}, y_{2\omega}, y_{3\omega}, y_{4\omega}, y_{5\omega}$ are the second stage variables. Both the first and the second stage variables are mixed-integer. $\tau_{\omega_1} = \tau_{\omega_2} = 0.5$. d_ω is the uncertain parameter. $d_{\omega_1} = 1.5$, $d_{\omega_2} = 2$. Here we describe how this problem can be solved by GBDBAB. Recall that at each node q of the branch and bound tree, GBD is applied to solve problem $(PCBAB_q)$.

For this problem, three nodes are visited in order to solve the problem to a relative optimality gap within 0.1% (see Figure 1). At each node, the lower bound $v(PCBAB_q)$ is the lower bound obtained by the GBD algorithm, i.e., the objective value of the Benders master problem in the last iteration. The upper bound $V(PCBAB_q)$ is obtained by using Heuristic 1.

$v(PCBAB_q)$, $V(PCBAB_q)$, the optimal solution at each node are shown in Table 1. The Benders cuts that are generated at all the nodes are shown in Tables 9-11 in Appendix 1. At the root node, 15 iterations are needed for the GBD algorithm to converge. However, the relative optimality gap of the global upper and lower bound that can be obtained at the root node is not within 0.1%. Therefore, we select one continuous first stage variable to branch on according to branching rule A. x_1 is branched on and two new nodes 1 and 2 are created. GBD is applied to solve the problem at nodes 1 and 2, respectively. Both nodes 1 and 2 inherit the Benders cuts of the root node because the problems solved at this two nodes are restrictions of the root node and the Benders cuts of the root node are still valid. As a result, only 3 and 2 iterations are needed for the GBD to converge at nodes 1 and 2, respectively (see Tables 10, 11). The upper and the lower bounds of nodes 1 and 2 are within 0.1% optimality gap. Nodes 1 and 2 can be fathomed by optimality and GBDBAB terminates. An optimal value of -6.02080 is obtained.

However, if the hull relaxation of the CNF is used, which is a relaxation of $conv(S^q_\omega)$, the lower bound that can be obtained at the root node is -12.5767. The DNF yields a much tighter lower bound than the CNF at the root node, which shows the impact of applying the basic step.

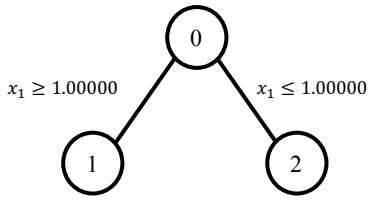

Fig. 1 Branch and bound tree for the illustrative example

Table 1 Optimal solution, upper and lower bound at each node of the BAB tree

Node	$v(PCBAB_q)$	$V(PCBAB_q)$	x_1^*	x_2^*	x_3^*	x_4^*	Branching constraints
0	-6.04451	-5.98613	1.00000	1.03467	1	1	-
1	-6.02092	-6.02072	1.00005	1.00003	1	1	$x_1 \geq 1.00000$
2	-6.02082	-6.02080	1.00000	1.00000	1	1	$x_1 \leq 1.00000$

5 A sequential convexification scheme to solve ($PCBAB_q$)

In solving problem ($PCBAB_q$), we assume that the convex hulls of S^q_ω, $\forall \omega \in \Omega$ are constructed by applying the basic steps and converting each GDP from a CNF to a DNF. However, the number of disjuncts can be large if we have a

large number of binary variables in the second stage decisions. The number of variables that are needed to represent the convex hull grows exponentially with the number of binary variables in the second stage decisions. Therefore, we propose a sequential convexification scheme, which progressively applies the basic steps to the CNF representation of S^q_ω. As we will see later, in some cases, it is not necessary to convert the CNF to DNF by applying all the basic steps. Namely, in some cases applying part of but not all the basic steps is sufficient to solve $(PCBAB_q)$. The partial application of all the possible basic steps can be regarded as a sequential convexification scheme to solve $(PCBAB_q)$, which will be discussed next.

Similarly to (16), we define the disjunction that specifies the jth binary variable as $S^{q1}_{\omega j} \cup S^{q0}_{\omega j}$, $j \in J_1$. The CNF of S^q_ω can be denoted as $\cap_{j\in J_1}(S^{q1}_{\omega j} \cup S^{q0}_{\omega j})$. In the sequential convexification scheme to solve $(PCBAB_q)$, we start with the hull relaxation of the CNF, and progressively apply basic steps to construct tighter relaxations if the problem $(PCBAB_q)$ is not solved.

Before we describe the details of the sequential convexification scheme, we first define some notation. A partial application of basic steps to the CNF $\cap_{j\in J_1}(S^{q1}_{\omega j} \cup S^{q0}_{\omega j})$ can be represented as $S^q_\omega = \cap_{t\in T^q_\omega}\big(\cap_{j\in D^q_{\omega t}}(S^{q1}_{\omega j} \cup S^{q0}_{\omega j})\big)$, where T^q_ω is the set of conjuncts, $D^q_{\omega t}$ is the set of the indices of the binary variables specified by conjunct t. $D^q_{\omega t}$ are disjoint sets and $\cup_{t\in T^q_\omega} D^q_{\omega t} = J_1$, i.e., the value of each binary variable $j \in J_1$ has to be specified in one and only one of the conjuncts. For each $t \in T^q_\omega$, $\cap_{j\in D^q_{\omega t}}(S^{q1}_{\omega j} \cup S^{q0}_{\omega j})$ can be represented in DNF by applying the basic steps, $\cap_{j\in D^q_{\omega t}}(S^{q1}_{\omega j} \cup S^{q0}_{\omega j}) = \cup_{r\in R^q_{\omega t}} S^q_{\omega tr}$. $R^q_{\omega t}$ is the set of disjuncts for the DNF representation. In each $S^q_{\omega tr}$, the values of all $(y_\omega)_j$, $j \in D^q_{\omega t}$, are specified to 0 or 1. Note that the dimension of $R^q_{\omega t}$ is $2^{|D^q_{\omega t}|}$, which corresponds to all possible combinations of the binary variables in $D^q_{\omega t}$. For the CNF, $|T^q_\omega| = |J_1|$ and $|D^q_{\omega t}| = 1$. If we apply all the possible basic steps to convert the CNF to the corresponding DNF, $|T^q_\omega| = 1$ and $|D^q_{\omega t}| = |J_1|$. The hull relaxation of $\cap_{t\in T^q_\omega}(\cup_{r\in R^q_{\omega t}} S^q_{\omega tr})$ is given in (46)-(53),

$$x = \sum_{r\in R^q_{\omega t}} u^r_{\omega t}, \quad \forall t \in T^q_\omega \tag{46}$$

$$y_\omega = \sum_{r\in R^q_{\omega t}} v^r_{\omega t}, \quad \forall t \in T^q_\omega \tag{47}$$

$$\sum_{r\in R^q_{\omega t}} \gamma^r_{\omega t} = 1, \quad \forall t \in T^q_\omega \tag{48}$$

$$\gamma^r_{\omega t} \geq 0, \quad \forall t \in T^q_\omega, r \in R^q_{\omega t} \tag{49}$$

$$A_{1,\omega} u^r_{\omega t} + \gamma^r_{\omega t} g_{1,\omega}(v^r_{\omega t}/\gamma^r_{\omega t}) \leq b_{1,\omega}\gamma^r_{\omega t}, \quad \forall t \in T^q_\omega, r \in R^q_{\omega t} \tag{50}$$

$$x^{lb}_q \gamma^r_{\omega t} \leq u^r_{\omega t} \leq x^{ub}_q \gamma^r_{\omega t}, \quad \forall t \in T^q_\omega, r \in R^q_{\omega t} \tag{51}$$

$$0 \leq v^r_{\omega t} \leq y^{ub}\gamma^r_{\omega t}, \quad \forall t \in T^q_\omega, r \in R^q_{\omega t} \tag{52}$$

$$(v^r_{\omega t})_j = (e^r_{\omega t})_j \gamma^r_{\omega t}, \quad \forall t \in T^q_\omega, r \in R^q_{\omega t}, j \in D^q_{\omega t} \tag{53}$$

Instead of solving $(PCBAB_q)$ directly, we can solve a relaxation of $(PCBAB_q)$ where $(x, y_\omega) \in h-rel\big(\cap_{t\in T^q_\omega}(\cup_{r\in R^q_{\omega t}} S^q_{\omega tr})\big)$. Problem $(PCBAB_q)$ is solved in iterations where the number of iteration is denoted by l. In the first iteration, $l = 1$, the problem with $(x, y_\omega) \in h - rel(\cap_{j\in J_1}(S^{q1}_{\omega j} \cup S^{q0}_{\omega j})$, $\forall\omega \in \Omega$, is solved. We denote the relaxation of $(PCBAB_q)$ that is solved at iteration l as $(PCBAB^l_q)$. The hull relaxation of $\cap_{t\in T^q_\omega}(\cup_{r\in R^q_{\omega t}} S^q_{\omega tr})$ at iteration l for scenario ω is denoted as $h - rel(S^q_{\omega l})$. $(PCBAB^l_q)$ is formally defined by (54)-(57).

$$(PCBAB^l_q) \quad \min \quad c^T x + \sum_{\omega\in\Omega} \tau_\omega d^T_\omega y_\omega \tag{54}$$

$$A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{55}$$

$$x^{lb}_q \leq x \leq x^{ub}_q \tag{56}$$

$$(x, y_\omega) \in h - rel(S^q_{\omega l}), \quad \forall\omega \in \Omega \tag{57}$$

$(PCBAB^l_q)$ can be solved using GBD to ϵ-optimality in a finite number of steps. The optimal value of $(PCBAB^l_q)$ is denoted as $v(PCBAB^l_q)$. The optimal solution is denoted as $(x^{q*}_l, y^{q*}_{l\omega_1}, y^{q*}_{l\omega_2}, \cdots, y^{q*}_{l\omega_{|\Omega|}})$. If we can prove that $(x^{q*}_l, y^{q*}_{l\omega_1}, y^{q*}_{l\omega_2}, \cdots, y^{q*}_{l\omega_{|\Omega|}})$ satisfies all the constraints of $(PCBAB_q)$, then $(x^{q*}_l, y^{q*}_{l\omega_1}, y^{q*}_{l\omega_2}, \cdots, y^{q*}_{l\omega_{|\Omega|}})$ solves $(PCBAB_q)$. More specifically, if $(x^{q*}_l, y^{q*}_{l\omega}) \in conv(S^q_\omega)$, $\forall\omega \in \Omega$, $(x^{q*}_l, y^{q*}_{l\omega_1}, y^{q*}_{l\omega_2}, \cdots, y^{q*}_{l\omega_{|\Omega|}})$ solves $(PCBAB_q)$.

However, deciding whether $(x^{q*}_l, y^{q*}_{l\omega})$ is in $conv(S^q_\omega)$ is not an easy problem as we do not want to use the closed-form representation of $conv(S^q_\omega)$, since this representation could be expensive when the number of second stage binary variables is large. Here, we provide a quick way to prove $(x^{q*}_l, y^{q*}_{l\omega}) \in conv(S^q_\omega)$ when some conditions are satisfied, which is only a sufficient condition for $(x^{q*}_l, y^{q*}_{l\omega})$ to be in $conv(S^q_\omega)$. Note that it is easy to check if a point is in S^q_ω just by checking if this point satisfies all the constraints that define S^q_ω. Therefore, the basic idea of this procedure is that if we could find that $(x^{q*}_l, y^{q*}_{l\omega})$ can be expressed as convex combination of some points that are in S^q_ω, then $(x^{q*}_l, y^{q*}_{l\omega})$ is in $conv(S^q_\omega)$. More specifically, note that after we solve $(PCBAB^l_q)$, we also obtain $v^{r*}_{\omega t}$, $\gamma^{r*}_{\omega t}$, $\forall\omega \in \Omega$, $t \in T^q_\omega$, $r \in R^q_{\omega t}$, where $y^{q*}_{l\omega} = \sum_{r\in R^q_{\omega t}} v^{r*}_{\omega t}$, $\forall\omega \in \Omega$, $t \in T^q_\omega$. We can establish a sufficient condition on $(x^{q*}_l, y^{q*}_{l\omega}) \in conv(S^q_\omega)$ by inspecting the values of $v^{r*}_{\omega t}$, $\gamma^{r*}_{\omega t}$ in Proposition 3.

Proposition 3 *For a given scenario ω, if $\exists t' \in T^q_\omega$ such that $(v^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'})_j$, $\forall r \in R^q_{\omega t'}, \gamma^{r*}_{\omega t'} > 0, j \in J_1$, are 0 or 1, i.e., $v^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'}$ satisfy the integrality constraints in S^q_ω, we have $(x^{q*}_l, y^{q*}_{l\omega}) \in conv(S^q_\omega)$.*

Proof By construction, $(u^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'}, v^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'})$, $\forall r \in R^q_{\omega t'}, \gamma^{r*}_{\omega t'} > 0$ already satisfies the continuous constraints that define S^q_ω. Therefore, $(u^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'}, v^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'})$, $\forall r \in R^q_{\omega t'}, \gamma^{r*}_{\omega t'} > 0$ are in S^q_ω. $(x^{q*}_l, y^{q*}_{l\omega})$ can be expressed as convex combination of $(u^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'}, v^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'})$, $\forall r \in R^q_{\omega t'}, \gamma^{r*}_{\omega t'} > 0$, i.e., $(x^{q*}_l, y^{q*}_{l\omega}) \in conv(S^q_\omega)$.

Note that Proposition 3 is only a sufficient condition for $(x_l^{q*}, y_{l\omega}^{q*}) \in conv(S_\omega^q)$. If we can prove that $(x_l^{q*}, y_{l\omega}^{q*}) \in conv(S_\omega^q)$, $\forall \omega \in \Omega$, $(PCBAB_q)$ is solved. Otherwise, for the scenarios where we cannot prove that $(x_l^{q*}, y_{l\omega}^{q*}) \in conv(S_\omega^q)$, we apply basic steps by updating T_ω^q, $D_{\omega t}^q$ to construct tighter relaxations until we are able to prove $(x_l^{q*}, y_{l\omega}^{q*}) \in conv(S_\omega^q)$, $\forall \omega \in \Omega$. Here, we propose two heuristics on how to apply the basic steps.

Heuristic A Note that in Proposition 3, in order to prove $(x_l^{q*}, y_{l\omega}^{q*}) \in conv(S_\omega^q)$ we need to find one disjunction t' such that $(v_{\omega t'}^{r*}/\gamma_{\omega t'}^{r*})_j$, $\forall j \in J_1$, $\gamma_{\omega t'}^{r*} > 0$, satisfy the integrality constraints. Therefore, in this heuristic, for a given scenario ω and for each disjunction $t \in T_\omega^q$, we first identify the most fractional $(v_{\omega t}^{r*}/\gamma_{\omega t}^{r*})_j$, $\forall r \in R_{\omega t}^q, \gamma_{\omega t}^{r*} > 0, j \in J_1$ and denote it as f_t.

$$f_t = \max_{r \in R_{\omega t}^q, \gamma_{\omega t}^{r*} > 0, j \in J_1} \min\{1 - (v_{\omega t}^{r*}/\gamma_{\omega t}^{r*})_j, (v_{\omega t}^{r*}/\gamma_{\omega t}^{r*})_j\} \tag{58}$$

We also denote the index of the most fractional $(v_{\omega t}^{r*}/\gamma_{\omega t}^{r*})_j$ as g_t,

$$g_t = \operatorname*{arg\,max}_{j: r \in R_{\omega t}^q, \gamma_{\omega t}^{r*} > 0, j \in J_1} \min\{1 - (v_{\omega t}^{r*}/\gamma_{\omega t}^{r*})_j, (v_{\omega t}^{r*}/\gamma_{\omega t}^{r*})_j\} \tag{59}$$

In the sufficient condition given by Proposition 3, we try to find disjunction t' such that $f_{t'}$ is close to zero, i.e., all the $(v_{\omega t'}^{r*}/\gamma_{\omega t'}^{r*})_j$, $\forall j \in J_1$, $\gamma_{\omega t'}^{r*} > 0$ satisfy the integrality constraints. Here, we slightly abuse notation and denote the disjunction t with the least f_t as t',

$$t' = \operatorname*{arg\,min}_{t \in T_\omega^q} f_t \tag{60}$$

If $f_{t'}$ is close to zero, we can prove $(x_l^{q*}, y_{l\omega}^{q*}) \in conv(S_\omega^q)$ by Proposition 3. Otherwise, we apply a basic step between t' and the disjunction that specifies $g_{t'}$. The idea is to make the most fractional $(v_{\omega t'}^{r*}/\gamma_{\omega t'}^{r*})_j$ in disjunction t' satisfy the integrality constraint in the next iteration.

Remark 3 For the problem with a large number of binary variables in the second stage, the sequential convexification scheme could potentially reduce the computational time by avoiding applying unnecessary basic steps. While the sequential convexification scheme is less expensive than using the DNF representation directly, solving $(PCBAB_q^l)$ in the first iteration with the CNF representation can also be expensive. Therefore, one may start with the NLP relaxation of S_ω^q and start using the CNF representation when the lower bound cannot be improved by the NLP relaxation anymore. In fact, this strategy is used when we solve the planning problem using GBDBAB with sequential convexification scheme in section 6.

6 Computational results

In this section, computational results of three different problems are presented to demonstrate the effectiveness of GBDBAB. The tolerance for the relative

optimality gap for GBDBAB is set to 0.1%. The GBDBAB algorithm to solve the three problems with increasing number of scenarios is implemented in JuMP/Julia [11]. All the problems are solved on the 12 processors of an Intel Xeon (2.67GHz) machine with 64 GB RAM. Depending on the computational efficiency on different problems, different NLP solvers including Knitro [7], and IPOPT [35] are used to solve the NLP subproblems in parallel. The choice of NLP solvers for each problem are discussed in detail in the subsequent subsections. CPLEX [10] is used to solve the MILP master problems. The corresponding deterministic equivalent formulations (DEFs) are implemented in GAMS and solved using convex MINLP solvers including DICOPT, AlphaECP, and SBB.

6.1 Computational results of the illustrative example with quadratic constraints

In section 4, an illustrative example with 2 scenarios is solved using GBDBAB. The convex nonlinear constraints in the model are all quadratic as shown in (40)-(44). d_ω is the only uncertain parameter in the problem. In this section, we generate stochastic convex MINLP problems with increasing number of scenarios by uniformly sampling d_ω in the interval $[0, 3]$. We assume that all the scenarios are of equal probability. The time limit for all DEFs of this small problem is set to 10,000 secs. The sizes of the DEFs, the total running time and the relative optimality gap of each solver are shown in Table 2. Although AlphaECP and DICOPT are able to solve the 20-scenario problem to optimality within 10 secs, the computational time increases dramatically with the increase in the number of scenarios. For instance, for the DEF with 300 scenarios, neither SBB nor DICOPT is able to obtain the optimal solution within the time limit. AlphaECP solves the problem in 917 secs.

Table 2 Computational statistics of the DEFs of the illustrative problem

Scenarios	Linear Constr	Nonlinear Constr	Binary Var	Continuous Var	SBB sec(gap)	AlphaECP sec(gap)	DICOPT sec(gap)
20	62	80	42	62	Timed out(7%)	9	2
60	182	240	122	182	Timed out(234%)	60	10
150	452	600	302	452	Timed out(245%)	254	685
300	902	1200	602	902	Timed out(247%)	917	Timed out(9%)

GBDBAB is also implemented to solve this illustrative problem. Since there are only two binary variables in the second stage, $conv(S^q_\omega)$ can be constructed easily with the DNF representation. The NLP subproblems are solved using IPOPT in parallel. The upper bound at each node is obtained by applying Heuristic 1 where the upper bound subproblems are solved using Pajarito. The total wall time, the wall time corresponding to solving the master problems, subproblems and upper bound subproblems, and the number of nodes

visited are shown in Table 3. In all the cases, only three nodes are visited in order to solve the problems to optimality due to the tight relaxations given by the DNF representation. It is easy to observe that the wall time for solving the subproblems is the most significant part of the total wall time. For the problems with 150 and 300 scenarios, GBDBAB performs better than using the solvers to solve the DEFs.

Table 3 Computational statistics of solving the illustrative problem using GBDBAB

Scenarios	Time (s)	Master (s)	Subproblem (s)	UB subproblem (s)	Nodes
20	80	3	59	10	3
60	76	3	58	5	3
150	111	10	81	9	3
300	121	11	81	11	3

6.2 Computational results of the constrained layout problem

The constrained layout problem under price uncertainty described in Appendix 2 is tested with 3 rectangles and 2 areas. Since the model is formulated as a GDP, the DEFs of the model are converted to MINLP with the big-M and the hull reformulation respectively. The time limit is set to 10,000 secs. The sizes of the DEFs with the big-M and the hull reformulation are shown in Table 4 and Table 5 respectively. Although all the solvers can solve the DEFs with 3 and 9 scenarios within the time limit, they all fail to obtain the optimal solution for the problems with 36 and 100 scenarios.

Table 4 Computational statistics of the big-M reformulation of the constrained layout problem

Scenarios	Linear Constr	Nonlinear Constr	Binary Var	Continuous Var	SBB sec(gap)	AlphaECP sec(gap)	DICOPT sec(gap)
3	54	72	30	36	14	120	3
9	108	216	66	84	4142	1478	43
36	351	864	228	300	Timed out(94%)	Timed out*	Timed out(84%)
100	927	2400	612	812	Timed out(98%)	Timed out*	Timed out(97%)

GBDBAB is also implemented to solve the constrained layout problems with increasing number of scenarios. In order to obtain $conv(S^q_\omega)$, constraint (64) and the constraints setting the upper and lower bound of all the variables are added to each disjunct shown in (65). We further apply basic steps to convert the CNF to DNF and reformulate the DNF with hull relaxation, which gives us $conv(S^q_\omega)$. The NLP subproblems are solved using IPOPT in parallel.

*No feasible solution found in 10,000 secs time limit.

Table 5 Computational statistics of the hull reformulation of the constrained layout problem

Scenarios	Linear Constr	Nonlinear Constr	Binary Var	Continuous Var	SBB sec(gap)	AlphaECP sec(gap)	DICOPT sec(gap)
3	360	72	30	180	38	32	6
9	702	216	66	372	5626	638	49
36	2241	864	228	1236	Timed out(52%)	Timed out (46%)	Timed out*
100	5889	2400	612	3284	Timed out(82%)	Timed out (67%)	Timed out*

The statistics of GBDBAB to solve these problems are shown in Table 6. For the problems with 3, 36, and 100 scenarios, optimality is proved at the root node. For the problem with 9 scenarios, 3 nodes are visited in order to prove optimality. Most of the computational time is still in solving the NLP subproblems.

Table 6 Computational statistics of solving the constrained layout problem with GBDBAB

Scenarios	Time (s)	Master (s)	Subproblem (s)	UB subproblem (s)	Nodes
3	137	40	63	8	1
9	276	65	17	17	3
36	444	117	277	20	1
100	537	85	363	39	1

6.3 Computational results of the planning problem

The planning problem reported in Li and Grossmann [21] is tested with 2 suppliers, 2 plants, 2 customers, and the number of scenarios from 3 to 81. The time limit for solving the DEFs is set to 50,000 secs. For the 81-scenario problem, none of the solvers is able to obtain the optimal solution within the time limit.

Table 7 Computational statistics of the DEFs of the planning problem

Scenarios	Linear Constr	Nonlinear Constr	Binary Var	Continuous Var	AlphaECP sec(gap)	SBB sec(gap)	DICOPT sec(gap)
3	332	12	32	338	6	49	3
9	980	36	80	998	81	Timed out(2%)	9
27	2,924	108	224	2,978	3530	Timed out(19%)	96
81	8,756	324	656	8,918	Timed out(2%)	Timed out(40%)	Timed out(0.2%)

GBDBAB is also applied to solve the planning problem. The NLP subproblems are solved with Knitro in this case since it behaves best for this problem among the NLP solvers available in JuMP. GBDBAB is able to solve all the planning problems at the root node. Sequential convexification scheme

with Heuristic A is applied and the maximum and the minimum number of basic steps that are applied in all the scenarios are shown in the last column of Table 8. In each of the four cases, the number of basic steps needed varies for different scenarios. For some of the scenarios, we are able to prove the solution (x, y_ω) is in $conv(S_\omega^{q^0})$ by Proposition 3 and therefore no more basic steps need to be applied. The computational statistics of GBDBAB are shown in Table 8. The problems with increasing number of scenarios can all be solved to optimality by GBDBAB within 10,000 secs.

Table 8 Computational statistics of solving the planning problem using GBDBAB

Scenarios	Time (s)	Master (s)	Subproblem (s)	UB subproblem (s)	Basic step (max, min)
3	705	59	491	16	(7,2)
9	1,221	92	861	23	(5,1)
27	1,859	216	1,403	17	(0,0)
81	7,994	1,534	5,091	83	(1,0)

Remark 4 The computational results in this section demonstrate that GBDBAB can outperform the convex MINLP solvers in solving the DEFs with a large number of scenarios. However, for the first small problem and the planning problem, some of the solvers can obtain solutions to the DEFs with small optimality gaps. The effect of using GBDBAB is most significant for the constrained layout problem. First of all, it is relatively easy to construct the convex hull of each subproblem as there are only 3 disjunctions in each subproblem. Second, since this problem is highly nonlinear, it would take solvers that use linearization strategies like DICOPT and AlphaECP many iterations to converge. Third, the constrained layout problem also has poor NLP relaxation. Solvers that are sensitive to the quality of NLP relaxation of the original problem like SBB would need more iterations to converge. Therefore, GBDBAB is most effective in solving highly nonlinear problems with poor relaxations but with small number of binary variables in the second stage decisions.

7 Conclusion

In this paper, a generalized Benders decomposition-based branch-and-bound algorithm, GBDBAB, is proposed to solve two-stage convex mixed integer nonlinear stochastic programs with mixed-integer variables in both first and second stage decisions. In order to obtain the convex hull of the mixed-integer nonlinear subproblems in closed-form, each subproblem is first represented in conjunctive normal form (CNF), and we apply basic steps to transform CNF to disjunctive normal form (DNF). The convex hull of each subproblem can be represented by the hull relaxation of the DNF. For the problems with pure binary first stage variables, we are able to prove that GBD has finite

ϵ-convergence if the convex hull of each subproblem is constructed in closed-form. For the problems with mixed-integer first stage variables, we also prove that in order to have finite ϵ-convergence, we may need to branch on the first stage continuous variables. A sequential convexification scheme is proposed to adaptively apply basic steps for the subproblems with a large number of binary variables.

Three problems with increasing number of scenarios are solved with GBD-BAB to within 0.1% optimality gap. Convex MINLP solvers including SBB, AlphaECP, and DICOPT are used to solve the corresponding DEFs to benchmark the proposed algorithm. It is shown that the proposed algorithm outperforms the solvers for the problems with a large number of scenarios. Especially, the algorithm is preferable for highly nonlinear problems with poor relaxations but with small number of binary variables in the second stage decisions like the constrained layout problem.

Acknowledgements The authors gratefully acknowledge financial support from the Center of Advanced Process Decion-making at Carnegie Mellon University and from the Department of Energy as part of the IDAES Project.

8 Appendix 1: Benders cuts for the illustrative example

The Benders cuts that are generated in each node of the branch-and-bound tree for the illustrative example in section 4 are shown in Tables 9-11.

9 Appendix 2: Constrained layout problem under price uncertainty

The constrained layout problem is adapted from the PhD thesis of Sawaya [28]. In this problem, we need to decide the layout of some units represented by rectangles with known lengths and widths. The units have to be manufactured before we install them at certain locations. In the manufacturing process, we need to provide the relative position of the units in order to produce the pipes that connect those units. The first stage decisions include the designed relative position of the units. After the units are manufactured, they can be installed at several locations. In order to install some unit(s) at a given location, we need to purchase a circled area around the center of that location. The rectangle(s) installed at that location has to be constrained within the circle that is purchased. The price of each area is uncertain and the designed units are installed according to the prices.

The sets, parameters, and variables that are used in the model is defined as follows:

Sets

$i \in N$ = rectangles (units)

$q \in Q$ = areas

$\omega \in \Omega$ = scenarios

Table 9 Benders cuts that are generated at the root node

Iteration	Benders cuts for ω_1	Benders cuts for ω_2
1	$\eta_{\omega_1} \geq -218.536x_1 - 1.44E+02x_2 - 16.1135x_3 - 16.1135x_4 + 73.5$	$\eta_{\omega_2} \geq -244.5x_1 - 1.60E+02x_2 - 18.0179x_3 - 18.0179x_4 + 98.5$
2	$\eta_{\omega_1} \geq 4.97E-01x_1 - 3.67E+04x_2 + 3790.08x_3 - 3782.6x_4 - 3715.16$	$\eta_{\omega_2} \geq 0.496464x_1 - 3.15E+04x_2 + 3245.59x_3 - 3237.48x_4 - 3145.69$
3	$\eta_{\omega_1} \geq -7.01E+04x_1 + 0.49604x_2 - 14326.1x_3 + 14672.1x_4 - 14597.2$	$\eta_{\omega_2} \geq -1.31E+05x_1 + 0.495977x_2 - 14554.5x_3 + 13995.4x_4 - 13895.5$
4	$\eta_{\omega_1} \geq 4.96E-01x_1 - 33738.8x_2 + 3526.15x_3 + 3525.62x_4 - 6975.37$	$\eta_{\omega_2} \geq 4.99E-01x_1 - 42390.2x_2 + 9538.87x_3 + 9537.98x_4 - 18975.4$
5	$\eta_{\omega_1} \geq 0.499992x_1 - 541.837x_2 + 62.1838x_3 + 62.2324x_4 - 50.4293$	$\eta_{\omega_2} \geq 0.499999x_1 - 541.843x_2 + 105.178x_3 + 111.603x_4 - 117.794$
6	$\eta_{\omega_1} \geq -49x_1 - 55.5x_2 + 9.52529x_3 + 9.51962x_4 + 57.4551$	$\eta_{\omega_2} \geq -49x_1 - 55.5x_2 + 8.33968x_3 + 12.4331x_4 + 80.7273$
7	$\eta_{\omega_1} \geq 1.72E-05x_1 + 7.79833x_2 + 10.0592x_3 + 10.0902x_4 - 38.3289$	$\eta_{\omega_2} \geq -0.0291935x_1 + 2.37213x_2 + 4.42782x_3 + 4.47516x_4 - 16.1181$
8	$\eta_{\omega_1} \geq 0.5x_1 - 122.173x_2 + 27.6631x_3 + 22.5481x_4 + 9.92204$	$\eta_{\omega_2} \geq 0.5x_1 - 122.173x_2 + 22.0681x_3 + 13.3055x_4 + 49.7594$
9	$\eta_{\omega_1} \geq 0.5x_1 - 6x_2 + 3.00259x_3 + 2.99245x_4 - 3.74505$	$\eta_{\omega_2} \geq 0.5x_1 - 68.4353x_2 + 15.7236x_3 + 15.6932x_4 + 31.8559$
10	$\eta_{\omega_1} \geq -8.58E+01x_1 + 0.5x_2 + 11.6499x_3 + 11.8356x_4 + 22.7981$	$\eta_{\omega_2} \geq -85.7842x_1 + 0.5x_2 + 11.6177x_3 + 14.9939x_4 + 44.672$
11	$\eta_{\omega_1} \geq 1.69E-05x_1 + 0.5x_2 + 3.94807x_3 + 3.95063x_4 - 11.4815$	$\eta_{\omega_2} \geq -55.6337x_1 + 0.5x_2 + 10.0374x_3 + 8.64236x_4 + 33.0624$
12	$\eta_{\omega_1} \geq 0.5x_1 - 0.999963x_2 + 3.61146x_3 + 3.58604x_4 - 9.72527$	$\eta_{\omega_2} \geq 0.5x_1 - 1.02808x_2 + 3.42646x_3 + 3.4266x_4 - 9.32503$
13	$\eta_{\omega_1} \geq 1.71E-05x_1 + 0.5x_2 + 3.86146x_3 + 4.98608x_4 - 12.4304$	$\eta_{\omega_2} \geq -0.0292157x_1 + 0.5x_2 + 4.01615x_3 + 4.15695x_4 - 11.644$
14	$\eta_{\omega_1} \geq 0.268432x_1 - 0.305251x_2 + 3.73107x_3 + 3.72861x_4 - 10.4742$	$\eta_{\omega_2} \geq -49.3943x_1 + 0.5x_2 + 11.471x_3 + 18.4473x_4 + 15.9746$
15	$\eta_{\omega_1} \geq 0.38848x_1 - 0.665399x_2 + 3.69908x_3 + 3.69815x_4 - 10.1589$	$\eta_{\omega_2} \geq -49.001x_1 + 0.5x_2 + 12.1317x_3 + 13.1548x_4 + 20.2145$

Table 10 Benders cuts that are generated at node 1

Iteration	Benders cuts for ω_1	Benders cuts for ω_2
1	$\eta_{\omega_1} \geq -23.8893x_1 + 0.5x_2 + 1.96244x_3 + 3.67593x_4 + 14.7302$	$\eta_{\omega_2} \geq -30.2523x_1 + 0.499996x_2 + 4.9294x_3 + 4.93241x_4 + 16.8905$
2	$\eta_{\omega_1} \geq -0.0221633x_1 + 0.5x_2 + 3.1919x_3 + 3.19356x_4 - 9.88416$	$\eta_{\omega_2} \geq -0.030201x_1 + 0.499998x_2 + 3.28529x_3 + 3.28533x_4 - 10.0405$
3	$\eta_{\omega_1} \geq -1.2564x_1 + 0.499999x_2 + 3.02764x_3 + 3.03329x_4 - 8.32534$	$\eta_{\omega_2} \geq -1.61672x_1 + 0.491218x_2 + 3.46615x_3 + 3.52462x_4 - 8.86527$

Table 11 Benders cuts that are generated at node 2

Iteration	Benders cuts for ω_1	Benders cuts for ω_2
1	$\eta_{\omega_1} \geq 6.41636x_1 + 0.499997x_2 + 5.16831x_3 + 5.17416x_4 - 20.2796$	$\eta_{\omega_2} \geq 5.9259x_1 + 0.499993x_2 + 5.16145x_3 + 5.1623x_4 - 19.7497$
2	$\eta_{\omega_1} \geq 0.631399x_1 + 0.499976x_2 + 5.06953x_3 + 5.12023x_4 - 14.3419$	$\eta_{\omega_2} \geq -0.299457x_1 - 9.43408x_2 + 4.11646x_3 + 4.11753x_4 - 1.50045$

Parameters
L_i = length of rectangle i
H_i = width of rectangle i
$xbar_q$ = the x coordinate of area q
$ybar_q$ = the y coordinate of area q
τ_ω = probability of scenario ω
$d_{q\omega}$ = unit price of area q in scenario ω
First stage decisions
x_i = the designed x coordinate of the center of rectangle i
y_i = the designed y coordinate of the center of rectangle i
$delx_{ij}$ = the designed distance of the center of rectangle i and the center of rectangle j along the x axis
$dely_{ij}$ = the designed distance of the center of rectangle i and the center of rectangle j along the y axis
Z^1_{ij}, Z^2_{ij}, Z^3_{ij}, $Z^4_{ij} \in \{True, False\}$ decides the relative position of rectangle i and rectangle j
Second stage decisions
$xs_{i\omega}$ = the installed x coordinate of the center of rectangle i in scenario ω
$ys_{i\omega}$ = the installed y coordinate of the center of rectangle i in scenario ω
$W_{qi\omega} = \{True, False\}$ whether to position rectangle i in area q or not
$S_{q\omega}$ = the size of area q that is purchased in scenario ω

Constraint (62) relates the designed relative distance of rectangle i and rectangle j in both x and y direction to the designed x and y coordinate of rectangle i and rectangle j. There are variable costs associated with the relative designed distances, which corresponds to the pipes that connect those units. Constraint (63) prevents rectangle i and rectangle j from overlapping with each other. After the design decisions are made, the actual position to install each unit $i \in N$ are decided. Constraint (64) enforces that the actual installed positions of rectangle i and rectangle j in each scenario $\omega \in \Omega$ have to coincide with the designed relative position. Moreover, in each scenario, each rectangle i has to be constrained in exactly one circle $q \in Q$ that is purchased, which is enforced by constraint (65). The objective also includes the expected cost of the purchase of circles. Note that the problem is expressed as a GDP and can be reformulated with big-M or hull reformulation as a convex MINLP, which is a two-stage stochastic program with mixed integer variables in both first and second stage.

$$\min \quad \sum_i \sum_{j,j>i} c_{ij}(delx_{ij} + dely_{ij}) + \sum_{\omega\in\Omega} \tau_\omega \sum_{q\in Q} (d_{q\omega} S_{q\omega}) \tag{61}$$

$$\begin{aligned} & s.t. \\ & delx_{ij} \geq x_i - x_j \quad \forall i,j \in N, i<j \\ & delx_{ij} \geq x_j - x_i \quad \forall i,j \in N, i<j \\ & dely_{ij} \geq y_i - y_j \quad \forall i,j \in N, i<j \\ & dely_{ij} \geq y_j - y_i \quad \forall i,j \in N, i<j \end{aligned} \tag{62}$$

$$\begin{bmatrix} Z_{ij}^1 \\ x_i + L_i/2 \leq x_j - L_j/2 \end{bmatrix} \vee \begin{bmatrix} Z_{ij}^2 \\ x_j + L_j/2 \leq x_i - L_i/2 \end{bmatrix} \vee \begin{bmatrix} Z_{ij}^3 \\ y_i + H_i/2 \leq y_j - H_j/2 \end{bmatrix} \vee \begin{bmatrix} Z_{ij}^4 \\ y_j + H_j/2 \leq y_i - H_i/2 \end{bmatrix}$$
$$\forall i, j \in N, i < j \tag{63}$$

$$\begin{aligned} xs_{i\omega} - xs_{j\omega} &= x_i - x_j \quad \forall i, j \in N, i < j, \omega \in \Omega \\ ys_{i\omega} - ys_{j\omega} &= y_i - y_j \quad \forall i, j \in N, i < j, \omega \in \Omega \end{aligned} \tag{64}$$

$$\vee_{q \in Q} \begin{bmatrix} W_{qi\omega} \\ (xs_{i\omega} - L_i/2 - xbar_q)^2 + (ys_{i\omega} + H_i/2 - ybar_q)^2 \leq S_{q\omega} \\ (xs_{i\omega} - L_i/2 - xbar_q)^2 + (ys_{i\omega} - H_i/2 - ybar_q)^2 \leq S_{q\omega} \\ (xs_{i\omega} + L_i/2 - xbar_q)^2 + (ys_{i\omega} + H_i/2 - ybar_q)^2 \leq S_{q\omega} \\ (xs_{i\omega} + L_i/2 - xbar_q)^2 + (ys_{i\omega} - H_i/2 - ybar_q)^2 \leq S_{q\omega} \end{bmatrix} \quad \forall i \in N, \omega \in \Omega \tag{65}$$

$$delx_{ij}, dely_{ij}, S_{q\omega} \in \mathbb{R}_+^1, Z_{ij}^1, Z_{ij}^2, Z_{ij}^3, Z_{ij}^4, W_{qi\omega} \in \{True, False\} \quad \forall i, j \in N, i < j, q \in Q, \omega \in \Omega \tag{66}$$

References

1. Angulo, G., Ahmed, S., Dey, S.S.: Improving the integer L-shaped method. INFORMS Journal on Computing **28**(3), 483–499 (2016)
2. Atakan, S., Sen, S.: A progressive hedging based branch-and-bound algorithm for stochastic mixed-integer programs. Optimization Online (2017). URL http://www.optimization-online.org/DB_HTML/2017/05/6008.html
3. Balas, E.: Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. SIAM Journal on Algebraic Discrete Methods **6**(3), 466–486 (1985)
4. Birge, J.R., Louveaux, F.: Introduction to stochastic programming. Springer Science & Business Media (2011)
5. Brunaud, B., Bassett, M.H., Agarwal, A., Wassick, J.M., Grossmann, I.E.: Efficient formulations for dynamic warehouse location under discrete transportation costs. Computers & Chemical Engineering (2017). DOI https://doi.org/10.1016/j.compchemeng.2017.05.011
6. Bussieck, M.R., Drud, A.: SBB: A new solver for mixed integer nonlinear programming. Talk, OR (2001). URL http://ftp.gamsworld.org/presentations/present_sbb.pdf
7. Byrd, R.H., Nocedal, J., Waltz, R.A.: Knitro: An integrated package for nonlinear optimization. In: Large-scale Nonlinear Optimization, pp. 35–59. Springer (2006)
8. Carøe, C.C., Schultz, R.: Dual decomposition in stochastic integer programming. Operations Research Letters **24**(1), 37–45 (1999)
9. Ceria, S., Soares, J.: Convex programming for disjunctive convex optimization. Mathematical Programming **86**(3), 595–614 (1999)

10. CPLEX, I.I.: V12. 1: User's manual for CPLEX. International Business Machines Corporation **46**(53), 157 (2009)
11. Dunning, I., Huchette, J., Lubin, M.: JuMP: A modeling language for mathematical optimization. SIAM Review **59**(2), 295–320 (2017)
12. Furman, K.C., Sawaya, N., Grossmann, I.: A computationally useful algebraic representation of nonlinear disjunctive convex sets using the perspective function. Optimization Online (2016). URL http://www.optimization-online.org/DB_FILE/2016/07/5544.pdf
13. Gade, D., Hackebeil, G., Ryan, S.M., Watson, J.P., Wets, R.J.B., Woodruff, D.L.: Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs. Mathematical Programming **157**(1), 47–67 (2016)
14. 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)
15. Geoffrion, A.M.: Generalized Benders decomposition. Journal of Optimization Theory and Applications **10**(4), 237–260 (1972)
16. Guignard, M.: Lagrangean relaxation. Top **11**(2), 151–200 (2003)
17. Jiang, R., Guan, Y., Watson, J.P.: Cutting planes for the multistage stochastic unit commitment problem. Mathematical Programming **157**(1), 121–151 (2016)
18. Kim, K., Zavala, V.M.: Algorithmic innovations and software for the dual decomposition method applied to stochastic mixed-integer programs. Mathematical Programming Computation pp. 1–42 (2017)
19. Küçükyavuz, S., Sen, S.: An introduction to two-stage stochastic mixed-integer programming. In: Leading Developments from INFORMS Communities, pp. 1–27. INFORMS (2017)
20. 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)
21. 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)
22. 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)
23. Lubin, M., Yamangil, E., Bent, R., Vielma, J.P.: Extended Formulations in Mixed-Integer Convex Programming, pp. 102–113. Springer International Publishing, Cham (2016)
24. Mijangos, E.: An algorithm for two-stage stochastic mixed-integer nonlinear convex problems. Annals of Operations Research **235**(1), 581–598 (2015)
25. 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)

26. Rockafellar, R.T., Wets, R.J.B.: Scenarios and policy aggregation in optimization under uncertainty. Mathematics of Operations Research **16**(1), 119–147 (1991)
27. Ruiz, J.P., Grossmann, I.E.: A hierarchy of relaxations for nonlinear convex generalized disjunctive programming. European Journal of Operational Research **218**(1), 38–47 (2012)
28. Sawaya, N.: Reformulations, relaxations and cutting planes for generalized disjunctive programming. Ph.D. thesis, Carnegie Mellon University (2006)
29. Sen, S., Higle, J.L.: The C-3 theorem and a D-2 algorithm for large scale stochastic mixed-integer programming: set convexification. Mathematical Programming **104**(1), 1–20 (2005)
30. 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)
31. 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)
32. 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)
33. 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)
34. Viswanathan, J., Grossmann, I.E.: A combined penalty function and outer-approximation method for MINLP optimization. Computers & Chemical Engineering **14**(7), 769–782 (1990)
35. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming **106**(1), 25–57 (2006)
36. Watson, J.P., Woodruff, D.L.: Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. Computational Management Science **8**(4), 355–370 (2011)
37. Watson, J.P., Woodruff, D.L., Hart, W.E.: PySP: modeling and solving stochastic programs in Python. Mathematical Programming Computation **4**(2), 109–149 (2012)
38. Westerlund, T., Lundqvist, K.: Alpha-ECP, version 5.01: An interactive MINLP-solver based on the extended cutting plane method. Åbo Akademi (2001)