




A joint decomposition method for global optimization of multiscenario nonconvex mixed-integer nonlinear programs

Emmanuel Ogbe¹ · Xiang Li¹ 

Received: 16 February 2018 / Accepted: 15 May 2019 / Published online: 21 May 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

This paper proposes a joint decomposition method that combines Lagrangian decomposition and generalized Benders decomposition, to efficiently solve multiscenario nonconvex mixed-integer nonlinear programming (MINLP) problems to global optimality, without the need for explicit branch and bound search. In this approach, we view the variables coupling the scenario dependent variables and those causing nonconvexity as complicating variables. We systematically solve the Lagrangian decomposition subproblems and the generalized Benders decomposition subproblems in a unified framework. The method requires the solution of a difficult relaxed master problem, but the problem is only solved when necessary. Enhancements to the method are made to reduce the number of the relaxed master problems to be solved and ease the solution of each relaxed master problem. We consider two scenario-based, two-stage stochastic nonconvex MINLP problems that arise from integrated design and operation of process networks in the case study, and we show that the proposed method can solve the two problems significantly faster than state-of-the-art global optimization solvers.

Keywords Generalized Benders decomposition · Dantzig–Wolfe decomposition · Lagrangian decomposition · Joint decomposition · Mixed-integer nonlinear programming · Global optimization · Stochastic programming

1 Introduction

Global optimization is a field of mathematical programming devoted to obtaining global optimal solutions; and it has over the years found enormous applications in Process Systems Engineering (PSE). Mixed-integer nonlinear programs are global optimization problems where some decision variables are integer while others are continuous. Discrete decisions and nonconvex nonlinearities introduce combinatorial behavior for such problems [1,2]. Various applications of mixed-integer nonlinear programming for PSE systems include natural gas network design and operation [3], gasoline blending and scheduling problems [4], expansion

✉ Xiang Li
xiang.li@queensu.ca

¹ Department of Chemical Engineering, Queen's University, Kingston, ON K7L 3N6, Canada

of chemical processes [5], reliable design of software [6,7], pump network problem [8,9], chemical process design synthesis [10], planning of facility investments for electric power generation [11], etc.

As adopted for mixed-integer linear programming (MILP), branch-and-bound has been employed for global optimization of nonconvex mixed-integer nonlinear programs (MINLP) [2,12,13]. The method entails systematically generating lower and upper bounds of the optimal objective function value over subdomains of the search space. The lower bounds can be generated via convex relaxations (such as McCormick relaxations [14]) or *Lagrangian relaxation* (or called *Lagrangian decomposition*) [15–17]. Ways of generating multipliers for the Lagrangian subproblem exist, including subgradient methods [18], cutting plane methods [15], and the Dantzig–Wolfe master problem (also known the restricted Lagrangian master problem) [19,20].

Branch-and-bound based strategies can be improved by incorporation of domain reduction techniques. Domain reduction entails eliminating portions of the feasible domain based on feasibility and optimality. Bound tightening or contraction [21], range reduction [22] and generation of cutting planes [23] are different domain reduction strategies that have been successful in solving nonconvex problems [7]. In bound contraction, the variable bounds are shrunk at every iteration by solving bound contraction subproblems [21]. In range reduction, the bounds on the variables are shrunk based on simple calculations using Lagrange multiplier information [22]. For cutting planes generation, Lagrangian relaxation information provides *cuts* that is used to cut-off portion of the feasible domain that does not contain the global optimum [24]. Current state-of-the-art commercial deterministic global optimization solvers embody branch-and-bound and enhancements such as tighter convex relaxations and domain reduction techniques, such as the Branch-And-Reduce Optimization Navigator (BARON) [2] and Algorithms for coNTinuous/Integer Global Optimization of Nonlinear Equations (ANTIGONE) [25]. They do provide rigorous frameworks for global optimization of Problem (P0).

Branch-and-bound based methods have been successful for global optimization, mostly for small to medium sized problems. However, when the size of the problem becomes large, the branch-and-bound steps needed for convergence can be prohibitively large. A typical example of large-scale nonconvex MINLP is the following multiscenario optimization problem:

$$\begin{aligned} \min_{x_0, v_1, \dots, v_s} \quad & \sum_{\omega=1}^s [f_{0,\omega}(x_0) + f_{\omega}(v_{\omega})] \\ \text{s.t.} \quad & g_{0,\omega}(x_0) + g_{\omega}(v_{\omega}) \leq 0, \quad \forall \omega \in \{1, \dots, s\}, \\ & v_{\omega} \in V_{\omega}, \quad \forall \omega \in \{1, \dots, s\}, \\ & x_0 \in X_0, \end{aligned} \tag{P0}$$

where x_0 links s subparts of the model that are indexed by ω , and it is called *linking variable* in the paper. We assume that at least one of the functions $f_{0,\omega} : X_0 \rightarrow \mathbb{R}$, $f_{\omega} : V_{\omega} \rightarrow \mathbb{R}$, $g_{0,\omega} : X_0 \rightarrow \mathbb{R}^m$, $g_{\omega} : V_{\omega} \rightarrow \mathbb{R}^m$ or one of the sets X_0 and V_{ω} is nonconvex, so Problem (P0) is a nonconvex MINLP, or a nonconvex nonlinear program (NLP) if no integer variables are involved. Clearly, (P0) is a large-scale problem when s is large. Problem (P0) has attracted more and more attention over the last 20 years in the field of PSE [26]. It usually arises from scenario-based two-stage stochastic programming [27,28], for which x_0 represents the first stage decisions that are made before the uncertainty is realized and v_{ω} represents second-stage decisions that are made after the uncertainty is revealed in scenario ω . Functions $f_{0,\omega}$ and f_{ω} represent probability times costs associated with x_0 and v_{ω} for every scenario ω . Problem

(P0) can also arise from integrated system design and operation problems which consider system operation over multiple time periods (but without uncertainties), such as for energy polygeneration plants [29] and electrical power distribution networks [30]). In this case, x_0 represents system design decisions and x_ω represents system operational decisions for time period (or scenario) ω , and $f_{0,\omega}$ and f_ω represent frequency of occurrence of time period ω times investment cost and operational cost, respectively. In this paper, we focus on how to efficiently solve Problem (P0) to global optimality, rather than how to generate scenarios and probabilities for stochastic programming or the time periods and their occurrence frequencies for multiperiod optimization.

It is well-known that Problem (P0) has a decomposable structure that could be exploited for efficient solution. *Benders decomposition* (BD) [31] (known as L-shaped method in the stochastic programming literature [27,28]) is one class of decomposition methods applied for MILPs. Geoffrion [32] generalized BD into *Generalized Benders Decomposition* (GBD), for solving convex MINLPs. Li et al. developed a further extension, called *Nonconvex Generalized Benders Decomposition* [4], for solving nonconvex MINLPs, but this method can guarantee global optimality only if the linking variable is fully integer. Karupiah and Grossmann applied a Lagrangian decomposition-based scheme to solve Problem (P0) [33]; in order to guarantee convergence to a global optimum, explicit branch-and-bound of linking variables is needed. They also presented bound contraction as an optional scheme in their Lagrangian-based branch-and-bound strategy. Shim et al. [34] proposed a method that combines Lagrangian decomposition and BD together with branch-and-bound (to ensure convergence), in order to solve a class of bilevel programs with an integer program in the upper-level and a complementarity problem in the lower-level. A more recent algorithm combining NGBD and Lagrangian decomposition was proposed by Kannan and Barton [35], and this algorithm also requires explicit branch-and-bound for convergence.

Efforts have been taken to achieve better computational efficiency by combining classical decomposition methods. In 1983, Van Roy proposed a *cross decomposition* method that combines Lagrangian decomposition and Benders decomposition [19] to solve MILP problems which do not have non-linking integer variables. Since then, a number of extensions and variants of cross decomposition have been developed [20,36–40]. All of these methods require that no nonconvexity comes from non-linking variables as otherwise finite convergence cannot be guaranteed.

The performance of branch-and-bound based solution methods depends heavily on the branching and node selection strategies, but what the best strategies are for a particular problem are usually unknown. In addition, branching and node selection strategies are not able to fully exploit the problem structure. Therefore, the goal of this paper is to develop a new decomposition method for global optimization of Problem (P0), which does not require explicit branch-and-bound. The new decomposition method was inspired by cross decomposition, and it follows a similar algorithm design philosophy, combining primarily generalized Benders decomposition and Lagrangian decomposition. However, its decomposition procedure is rather different in many details due to the nonconvexity it has to deal with, so we do not call it cross decomposition, but a new name *joint decomposition*. To the best of our knowledge, this is the first decomposition method that can solve Problem (P0) to global optimality without explicitly performing branch-and-bound (but the solution of nonconvex subproblems requires branch-and-bound based solvers).

The remaining part of the article is organized as follows. In Sect. 2, we give a brief introduction to generalized Benders decomposition and Lagrangian decomposition, using a reformulation of Problem (P0). Then in Sect. 3, we present the basic joint decomposition algorithm and the convergence proof. Section 4 discusses enhancements to the basic joint

decomposition algorithm, including domain reduction and use of extra convex relaxation subproblems. The joint decomposition methods are tested with two case study problems adapted from the literature, and the simulation results demonstrate the effectiveness and the computational advantages of the methods. The article ends with concluding remarks in Sect. 6.

2 Problem reformulation and classical decomposition methods

In order to bring up the joint decomposition idea, we reformulate Problem (P0) and briefly discuss how the reformulated problem can be solved via classical GBD and LD methods. The reformulation starts by separating the convex part and the nonconvex part of the problem, and it ends up in the following form:

$$\begin{aligned}
 & \min_{\substack{x_0, x_1, \dots, x_s \\ y_1, \dots, y_s}} \sum_{\omega=1}^s c_{\omega}^T x_{\omega} \\
 \text{s.t.} \quad & x_0 = H_{\omega} x_{\omega}, \quad \forall \omega \in \{1, \dots, s\}, \\
 & A_{\omega} x_{\omega} + B_{\omega} y_{\omega} \leq 0, \quad \forall \omega \in \{1, \dots, s\}, \\
 & x_0 \in X_0, \\
 & x_{\omega} \in X_{\omega}, \quad y_{\omega} \in Y_{\omega}, \quad \forall \omega \in \{1, \dots, s\},
 \end{aligned} \tag{P}$$

where set $X_{\omega} \subset \mathbb{R}^{n_x}$ is convex, set $Y_{\omega} \subset \mathbb{R}^{n_y}$ is nonconvex, and set $x_0 \subset \mathbb{R}^{n_0}$ can be either convex or nonconvex. The first group of equations in (P) are *nonanticipativity constraints* (NACs) [17,24,41], where matrix $H_{\omega} \in \mathbb{R}^{n_0} \times \mathbb{R}^{n_x}$ selects from x_{ω} the duplicated x_0 for scenario ω . The details of transforming (P0) to (P) are provided in “Appendix A”.

x_0 and y_{ω} are the two reasons why Problem (P) is difficult to solve. Linking variables x_0 couple different subparts of the model and they cause nonconvexity if set X_0 is nonconvex. Variables y_{ω} cause nonconvexity due to the nonconvexity of set Y_{ω} . If the values of x_0 and y_{ω} are fixed, the problem will be much easier to solve. Therefore, in this paper we call x_0 and y_{ω} *complicating variables*. In order to distinguish the two sets of variables, we also call x_0 *linking variables*, and y_{ω} *non-linking complicating variables*. We also call x_{ω} *non-complicating variables*.

The classical GBD method can be used to solve Problem (P) by treating x_0 and y_{ω} as complicating variables, while the LD method can be used to solve Problem (P) by dualizing NACs so that x_0 no long links different scenarios. In the next two subsections we briefly introduce GBD and LD for Problem (P), and we make the following assumptions for Problem (P) for convenience of discussion.

Assumption 1 X_0, X_{ω} and Y_{ω} for all $\omega \in \{1, \dots, s\}$ are non-empty and compact.

Assumption 2 After fixing (x_0, y_1, \dots, y_s) to any point in $X_0 \times Y_1 \times \dots \times Y_s$, if Problem (P) is feasible, it satisfies Slater condition.

Assumption 1 is a mild assumption, as for most real-world applications, the variables are naturally bounded and the functions involved are continuous. If a discontinuous function is involved, it can usually be expressed with continuous functions and extra integer variables. Assumption 2 ensures strong duality of convex subproblems that is required for GBD. If this assumption is not satisfied for a problem, we can treat the non-complicating variables that fail the Slater condition to be complicating variables, so that after fixing all complicating variables the Slater condition is satisfied.

2.1 Generalized Benders decomposition

At each GBD iteration l , fixing the complicating variables $x_0 = x_0^{(l)}, y_\omega = y_\omega^{(l)} (\forall \omega \in \{1, \dots, s\})$ results in an upper bounding problem that can be decomposed into the following Benders primal subproblem for each scenario ω :

$$\begin{aligned}
 obj_{\text{BPP}_\omega^{(l)}} &= \min_{x_\omega} c_\omega^T x_\omega \\
 \text{s.t.} \quad &x_0^{(l)} = H_\omega x_\omega, \\
 &A_\omega x_\omega + B_\omega y_\omega^{(l)} \leq 0, \\
 &x_\omega \in X_\omega,
 \end{aligned} \tag{BPP}_\omega^{(l)}$$

$obj_{\text{BPP}_\omega^{(l)}}$ is the optimal objective value of $(\text{BPP}_\omega^{(l)})$. For convenience, we indicate the optimal objective value of a problem in the above way for all subproblems discussed in this paper. Obviously, $\sum_{\omega=1}^s obj_{\text{BPP}_\omega^{(l)}}$ represents an upper bound for Problem (P). If $(\text{BPP}_\omega^{(l)})$ is infeasible for one scenario, then solve the following Benders feasibility subproblem for each scenario ω :

$$\begin{aligned}
 obj_{\text{BFP}_\omega^{(l)}} &= \min_{x_\omega, z_{1,\omega}^+, z_{1,\omega}^-, z_{2,\omega}} \|z_{1,\omega}^+\| + \|z_{1,\omega}^-\| + \|z_{2,\omega}\| \\
 \text{s.t.} \quad &x_0^{(l)} = H_\omega x_\omega + z_{1,\omega}^+ - z_{1,\omega}^-, \\
 &A_\omega x_\omega + B_\omega y_\omega^{(l)} \leq z_{2,\omega}, \\
 &x_\omega \in X_\omega, \quad z_{1,\omega}^+, z_{1,\omega}^-, z_{2,\omega} \geq 0,
 \end{aligned} \tag{BFP}_\omega^{(l)}$$

where $z_{1,\omega}^+, z_{1,\omega}^-$, and $z_{2,\omega}$ are slack variables. Note that $(\text{BFP}_\omega^{(l)})$ is always feasible according to Assumption 1. Solution of $(\text{BFP}_\omega^{(l)})$ provides a feasibility cut (that is described below), which prevents the generation of the same infeasible $x_0^{(l)}$ and $y_\omega^{(l)}$ [42].

At the same iteration, the following Benders relaxed master problem is solved to yield a lower bound for Problem (P):

$$\begin{aligned}
 &\min_{\substack{x_0, \eta_0, \eta_1, \dots, \eta_s \\ y_1, \dots, y_s}} \eta_0 \\
 \text{s.t.} \quad &\eta_0 \geq \sum_{\omega=1}^s \eta_\omega \\
 &\eta_\omega \geq obj_{\text{BPP}_\omega^{(j)}} + (\lambda_\omega^{(j)})^T B_\omega (y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^T (x_0 - x_0^{(j)}), \\
 &\quad \forall \omega \in \{1, \dots, s\}, \quad \forall j \in T^{(l)}, \\
 &0 \geq obj_{\text{BFP}_\omega^{(j)}} + (\lambda_\omega^{(j)})^T B_\omega (y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^T (x_0 - x_0^{(j)}), \\
 &\quad \forall \omega \in \{1, \dots, s\}, \quad \forall j \in S^{(l)}, \\
 &x_0 \in X_0, \\
 &y_\omega \in Y_\omega, \quad \forall \omega \in \{1, \dots, s\},
 \end{aligned} \tag{BRMP}^{(l)}$$

where $\mu_\omega^{(l)}$ includes Lagrange multipliers for the first group of constraints in Problem $(\text{BPP}_\omega^{(l)})$ or $(\text{BFP}_\omega^{(l)})$, and $\lambda_\omega^{(l)}$ includes Lagrange multipliers for the second group of constraints in Problem $(\text{BPP}_\omega^{(l)})$ or $(\text{BFP}_\omega^{(l)})$. Set $T^{(l)}$ includes indices of Benders iterations at which only

(BPP $_{\omega}^{(l)}$) is solved, and set $S^{(l)}$ includes indices of Benders iterations at which (BFP $_{\omega}^{(l)}$) is solved. Note that Problem (BRMP $^{(l)}$) is used in the multicut BD or GBD, which is different from the one used in the classical single cut BD or GBD. The multicut version of the Benders master problem is known to be tighter than the single cut version [43,44], so it is considered in this paper.

Remark 1 The finite convergence property of GBD is stated and proved in [32]. In Sect. 3, we will provide more details in the context of our new decomposition method.

Remark 2 For (P), the relaxed master problem (BRMP $^{(l)}$) can still be very difficult as its size grows with the number of scenarios. However, if most variables in (P) are non-complicating variables, the size of (BRMP $^{(l)}$) is much smaller than that of (P), and then (BRMP $^{(l)}$) is much easier to solve than (P).

2.2 Lagrangian decomposition

We start discussing LD from the Lagrangian dual of Problem (P) that is constructed by dualizing the NACs of the problem:

$$obj_{DP} = \max_{\pi_1, \dots, \pi_s \geq 0} obj_{LS}(\pi_1, \dots, \pi_s), \tag{DP}$$

where $obj_{LS}(\pi_1, \dots, \pi_s)$ is the optimal objective value of the following Lagrangian subproblem with given (π_1, \dots, π_s) :

$$\begin{aligned} \min_{\substack{x_0, x_1, \dots, x_s \\ y_1, \dots, y_s}} & \sum_{\omega=1}^s [c_{\omega}^T x_{\omega} + \pi_{\omega}^T (x_0 - H_{\omega} x_{\omega})] \\ \text{s.t.} & \quad A_{\omega} x_{\omega} + B_{\omega} y_{\omega} \leq 0, \quad \forall \omega \in \{1, \dots, s\}, \\ & \quad x_0 \in X_0, \\ & \quad x_{\omega} \in X_{\omega}, y_{\omega} \in Y_{\omega}, \quad \forall \omega \in \{1, \dots, s\}. \end{aligned} \tag{LS}(\pi_1, \dots, \pi_s)$$

Due to weak duality, Problem (DP) or any Lagrangian subproblem is a lower bounding problem for Problem (P). Typically, the LD method is incorporated in a branch-and-bound framework that only needs to branch on linking variables x_0 to guarantee convergence to an ϵ -optimal solution. At each branch-and-bound node or LD iteration k , a set of multipliers $(\pi_1^k, \dots, \pi_s^k)$ are selected to construct a Lagrangian subproblem for (DP), and this subproblem can be naturally decomposed into $s + 1$ subproblems, i.e.,

$$\begin{aligned} obj_{LS_0^k} &= \min_{x_0} \sum_{\omega=1}^s (\pi_{\omega}^k)^T x_0 \\ \text{s.t.} & \quad x_0 \in X_0, \end{aligned} \tag{LS}_0^k$$

and

$$\begin{aligned} \min_{x_{\omega}, y_{\omega}} & \quad c_{\omega}^T x_{\omega} - (\pi_{\omega}^k)^T H_{\omega} x_{\omega} \\ \text{s.t.} & \quad A_{\omega} x_{\omega} + B_{\omega} y_{\omega} \leq 0, \\ & \quad x_{\omega} \in X_{\omega}, \quad y_{\omega} \in Y_{\omega}, \end{aligned} \tag{LS}_{\omega}^k$$

for all $\omega \in \{1, \dots, s\}$. Let obj_{LS^k} be the optimal objective value of the Lagrangian subproblem, then $obj_{LS^k} = \sum_{\omega=1}^s obj_{LS_{\omega}^k} + obj_{LS_0^k}$. Clearly, $obj_{LS^k} \leq obj_{DP}$ always holds. If $(\pi_1^k, \dots, \pi_s^k)$ happens to be an optimal solution of (DP), then $obj_{LS^k} = obj_{DP}$.

The upper bounds in the LD methods are typically generated by fixing x_0 to certain values. At each iteration k , an upper bounding problem, or called primal problem, is constructed via fixing $x_0 = x_0^k$ (which may be the solution of (LS_0^k)), and this problem can be separated into s primal subproblem in the following form:

$$\begin{aligned}
 obj_{PP^k_\omega} &= \min_{x_\omega, y_\omega} c_\omega^T x_\omega \\
 \text{s.t.} \quad &x_0^k = H_\omega x_\omega, \\
 &A_\omega x_\omega + B_\omega y_\omega \leq 0, \\
 &x_\omega \in X_\omega, \quad y_\omega \in Y_\omega,
 \end{aligned}
 \tag{PP^k_\omega}$$

Let obj_{PP^k} be the optimal objective value of the primal problem, then $obj_{PP^k} = \sum_{\omega=1}^s obj_{PP^k_\omega}$.

For generation of multipliers, we take the idea from Dantzig–Wolfe decomposition, which is essentially a special LD method. Consider the convex hull of nonconvex set Y_ω :

$$\tilde{Y}_\omega = \left\{ y_\omega \in \mathbb{R}^{n_y} : y_\omega = \sum_{i \in I} \theta_\omega^{[i]} y_\omega^{[i]}, \sum_{i \in I} \theta_\omega^{[i]} = 1, \theta_\omega^{[i]} \geq 0, \forall i \in I \right\},$$

where $y_\omega^{[i]}$ denotes a point in Y_ω that is indexed by i . The index set I may need to be an infinite set for \tilde{Y}_ω being the convex hull. Replace Y_ω with its convex hull for all ω in (P) , then we get the following Dantzig–Wolfe master problem, or called primal master problem in this paper:

$$\begin{aligned}
 \min_{\substack{x_0, \theta_1^{[i]}, \dots, \theta_s^{[i]} \\ x_1, \dots, x_s}} &\sum_{\omega=1}^s c_\omega^T x_\omega \\
 \text{s.t.} \quad &x_0 = H_\omega x_\omega, \quad \forall \omega \in \{1, \dots, s\}, \\
 &A_\omega x_\omega + B_\omega \sum_{i \in I} \theta_\omega^{[i]} y_\omega^{[i]} \leq 0, \quad \forall \omega \in \{1, \dots, s\}, \\
 &\sum_{i \in I} \theta_\omega^{[i]} = 1, \quad \theta_\omega^{[i]} \geq 0, \quad \forall i \in I, \quad \forall \omega \in \{1, \dots, s\}, \\
 &x_0 \in X_0, \\
 &x_\omega \in X_\omega, \quad \forall \omega \in \{1, \dots, s\}
 \end{aligned}
 \tag{PMP}$$

Clearly, Problem (PMP) is a relaxation of Problem (P) , and it is either fully convex or partially convex (as set X_0 can still be nonconvex). At LD iteration k , the following restriction of (PMP) can be solved:

$$\begin{aligned}
 \min_{\substack{x_0, \theta_1^{[i]}, \dots, \theta_s^{[i]} \\ x_1, \dots, x_s}} &\sum_{\omega=1}^s c_\omega^T x_\omega \\
 \text{s.t.} \quad &x_0 = H_\omega x_\omega, \quad \forall \omega \in \{1, \dots, s\}, \\
 &A_\omega x_\omega + B_\omega \sum_{i \in I^k} \theta_\omega^{[i]} y_\omega^{[i]} \leq 0, \quad \forall \omega \in \{1, \dots, s\}, \\
 &\sum_{i \in I^k} \theta_\omega^{[i]} = 1, \quad \theta_\omega^{[i]} \geq 0, \quad \forall i \in I^k, \quad \forall \omega \in \{1, \dots, s\}, \\
 &x_0 \in X_0, \\
 &x_\omega \in X_\omega, \quad \forall \omega \in \{1, \dots, s\},
 \end{aligned}
 \tag{RPMP^k}$$

where index set $I^k \subset I$ is finite. I^k may consist of indices of y_ω that are generated in the previously solved primal problems and Lagrangian subproblems. Replacing set I with set I^k is a restriction operation, so (RPMP^k) is a restriction of (PMP) . Since (PMP) is a relaxation of (P) , (RPMP^k) is neither a relaxation nor a restriction of (P) , so it does not yield an upper or a lower bound of (P) . The role of (RPMP^k) in joint decomposition is to generate multipliers for NACs to construct a Lagrangian subproblem, and to generate x_0^{k+1} to construct (PP_ω^{k+1}) . Problem (RPMP^k) can be solved by an optimization solver or by GBD.

Actually, we can construct a different Lagrangian dual of Problem (P) by dualizing both the NACs and the second group of constraints in the problem, as what we do for GBD in the last subsection. However, this Lagrangian dual is not as tight as Problem (DP) (as stated by the following proposition), so it is not preferred for a LD method. The following proposition follows from Theorem 3.1 of [17] and its proof is omitted here.

Proposition 1 Consider the following Lagrangian dual of Problem (P) :

$$\text{obj}_{\text{DP2}} = \max_{\substack{\mu_1, \dots, \mu_s \geq 0 \\ \lambda_1, \dots, \lambda_s \geq 0}} \text{obj}_{\text{LS2}}(\mu_1, \dots, \mu_s, \lambda_1, \dots, \lambda_s), \tag{DP2}$$

where

$$\begin{aligned} \text{obj}_{\text{LS2}} = \min_{\substack{x_0, x_1, \dots, x_s \\ y_1, \dots, y_s}} & \sum_{\omega=1}^s [c_\omega^T x_\omega + \mu_\omega^T (x_0 - H_\omega x_\omega) + \lambda_\omega^T (A_\omega x_\omega + B_\omega y_\omega)] \\ \text{s.t. } & x_0 \in X_0, \\ & x_\omega \in X_\omega, y_\omega \in Y_\omega, \quad \forall \omega \in \{1, \dots, s\}. \end{aligned}$$

The duality gap of (DP) is no larger than the duality gap of (DP2) .

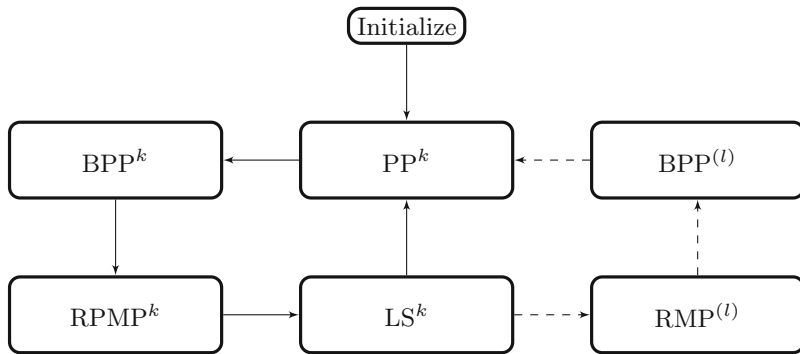
3 The joint decomposition method

3.1 Synergizing LD and GBD

In the LD method described in the last section, at each iteration the subproblems to be solved are much easier than the original problem (P) , as either the size of the subproblem is independent of number of scenarios, such as (PP_ω^k) , (LS_ω^k) , and (LS_ω^k) , or the subproblem is a MILP or convex MINLP that can be solved by existing optimization solvers or by GBD relatively easily, such as (RPMP^k) . However, without branching on the linking variables x_0 , LD cannot guarantee finding a global solution, and we do not always know how to exploit the problem structure to efficiently branch on x_0 and whether the branching can be efficient enough.

On the other hand, GBD can find a global solution, but it requires solving the nonconvex relaxed master problem $(\text{BRMP}^{(l)})$ at each iteration. The size of $(\text{BRMP}^{(l)})$ may be much smaller than the size of (P) if most variables in (P) are non-complicating variables, but $(\text{BRMP}^{(l)})$ can still be difficult to solve, especially considering that it needs to be solved at each iteration and its size grows with the number of iterations.

Therefore, there may be a way to combine LD and GBD, such that we solve as many LD subproblems and Benders primal subproblems as possible (as they are relatively easy to solve), but avoid solving many difficult Benders relaxed master problems $(\text{BRMP}^{(l)})$. This idea is similar to the one that motivates cross decomposition [19], but it leads to very different subproblems and a very different algorithmic procedure. The subproblems are very



$RPMP^k$: Restricted Primal Master Problem
 LS^k : Lagrangian subproblem, decomposed into (LS_0^k) and (LS_ω^k) ($\omega = 1, \dots, s$).
 $RMP^{(l)}$: Relaxed Master Problem, with extra cuts from LS^k and BPP^k .
 $BPP^{(l)}$: Benders Primal Problem, decomposed into $(BPP_\omega^{(l)})$ ($\omega = 1, \dots, s$).
 PP^k : Primal Problem, decomposed into (PP_ω^k) ($\omega = 1, \dots, s$).
 BPP^k : Benders Primal Problem, solved after PP^k is solved.

Fig. 1 The basic joint decomposition framework

different, because for problem (P), we prefer dualizing only NACs in LD in order to achieve the smallest possible dual gap (according to Proposition 1), but we have to dualize both the NACs and the second group of constraints in GBD. In addition, due to the different nature of the subproblems, the order in which the subproblems are solved and how often the problems are solved are different. Therefore, we do not name the proposed method cross decomposition, but call it *joint decomposition* (JD).

Figure 1 shows the basic framework of JD. Each JD iteration includes one LD iteration part, as indicated by the solid lines, and possibly one GBD iteration, as indicated by the dashed lines. In a JD iteration, the GBD iteration is performed only when the LD iteration improves over the previous LD iteration substantially. The GBD iteration is same to the one described in the last section, except that the relaxed master problem ($BRMP^{(l)}$) includes more valid cuts (which will be described later). The LD iteration is slightly different from the one described in the last section. One difference is that, after solving (PP_ω^k) at LD iteration k , a Benders primal problem (BPP^k) is constructed using x_0^k (which is used for constructing (PP_ω^k)) and (y_1, \dots, y_s) (which is from the optimal solution of (PP_ω^k)). The (BPP^k) is solved to generate a Benders cut that can be added to $(BRMP^{(l)})$. The other difference is that $(RPMP^k)$, (LS_0^k) , (LS_ω^k) (decomposed from (LS^k)) slightly differ from the ones described in the last section, and they will be described later.

Remark 3 The JD method requires that all subproblems can be solved using an existing optimization solver within reasonable time. If this requirement is not met, then JD does not work, or we have to further decompose the difficult subproblems into smaller, solvable subproblems.

3.2 Feasibility issues

According to Assumption 1, a subproblem in JD either has a solution or is infeasible. Here we explain how JD handles infeasibility of a subproblem.

First, if a lower bounding problem (LS^k) or ($BRMP^{(l)}$) is infeasible, then the original problem (P) is infeasible and JD can terminate.

Second, if (BPP^k) or ($BPP^{(l)}$) is infeasible, then JD will solve the corresponding Benders feasibility problem (BFP^k) or ($BFP^{(l)}$) to yield a feasibility cut. If (BFP^k) or ($BFP^{(l)}$) is infeasible, then (P) is infeasible and JD can terminate.

Third, if (PP^k_ω) is infeasible, then JD will solve a feasibility problem that “softens” the second group of constraints: and this problem can be separated into s subproblems as follows:

$$\begin{aligned} & \min_{x_\omega, y_\omega, z_\omega} \quad \|z_\omega\| \\ \text{s.t.} \quad & x_0^k = H_\omega x_\omega, \\ & A_\omega x_\omega + B_\omega y_\omega \leq z_\omega, \\ & x_\omega \in X_\omega, \quad y_\omega \in Y_\omega, \quad z_\omega \geq 0. \end{aligned} \tag{FP^k_\omega}$$

If (FP^k_ω) is infeasible for one scenario ω , then (P) is infeasible and JD can terminate. If (FP^k_ω) is feasible for all scenarios, then JD can construct and solve a feasible Benders feasibility problem (BFP^k) to yield a Benders feasibility cut for ($BRMP^{(l)}$).

Finally, problem ($RPMP^k$) can actually be infeasible if none of the $(y_1^{[l]}, \dots, y_s^{[l]})$ in the problem is feasible for the original problem (P). To prevent this infeasibility, we can generate a point $(\hat{y}_1, \dots, \hat{y}_s)$ that is feasible for (P), by solving the following initial feasibility problem:

$$\begin{aligned} & \min_{\substack{x_0, x_1, \dots, x_s \\ y_1, \dots, y_s \\ z_1, \dots, z_s}} \quad \sum_{\omega=1}^s \|z_\omega\| \\ \text{s.t.} \quad & x_0 = H_\omega x_\omega, \quad \forall \omega \in \{1, \dots, s\}, \\ & A_\omega x_\omega + B_\omega y_\omega \leq z_\omega, \quad \forall \omega \in \{1, \dots, s\}, \\ & x_0 \in X_0, \\ & x_\omega \in X_\omega, \quad y_\omega \in Y_\omega, \quad z_\omega \geq 0, \quad \forall \omega \in \{1, \dots, s\}. \end{aligned} \tag{IFP}$$

Problem (IFP) is not naturally decomposable over the scenarios, but it can be solved by JD. When solving (IFP) using JD, the restricted primal master problem ($RPMP^k$) must have a solution (according to Assumption 1).

3.3 The tightened subproblems

The relaxed master problem described in Sect. 2 can be tightened with the solutions of previously solved subproblems in JD. The tightened problem, called joint decomposition relaxed master problem, can be written as:

$$\begin{aligned}
 & \min_{\substack{x_0, \eta_0, \eta_1, \dots, \eta_s \\ y_1, \dots, y_s}} \eta_0 \\
 \text{s.t. } & \eta_0 \geq \sum_{\omega=1}^s \eta_\omega, \\
 & \eta_\omega \geq \text{obj}_{\text{BPP}_\omega^{(j)}} + (\lambda_\omega^{(j)})^T B_\omega (y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^T (x_0 - x_0^{(j)}), \\
 & \quad \forall \omega \in \{1, \dots, s\}, \quad \forall j \in T^{(l)}, \\
 & 0 \geq \text{obj}_{\text{BFP}_\omega^{(j)}} + (\lambda_\omega^{(j)})^T B_\omega (y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^T (x_0 - x_0^{(j)}), \\
 & \quad \forall \omega \in \{1, \dots, s\}, \quad \forall j \in S^{(l)}, \\
 & \eta_\omega \geq \text{obj}_{\text{BPP}_\omega^j} + (\lambda_\omega^j)^T B_\omega (y_\omega - y_\omega^j) + (\mu_\omega^j)^T (x_0 - x_0^j), \\
 & \quad \forall \omega \in \{1, \dots, s\}, \quad \forall j \in T^k, \\
 & 0 \geq \text{obj}_{\text{BFP}_\omega^j} + (\lambda_\omega^j)^T B_\omega (y_\omega - y_\omega^j) + (\mu_\omega^j)^T (x_0 - x_0^j), \\
 & \quad \forall \omega \in \{1, \dots, s\}, \quad \forall j \in S^k, \\
 & \eta_0 \leq UBD, \\
 & \eta_0 \geq LBD, \\
 & \eta_\omega \geq \text{obj}_{\text{LS}_\omega^i} + (\pi_\omega^i)^T x_0, \quad \forall \omega \in \{1, \dots, s\}, \quad \forall i \in R^k, \\
 & x_0 \in X_0, \quad y_\omega \in Y_\omega, \quad \forall \omega \in \{1, \dots, s\},
 \end{aligned} \tag{JRMP}^{(l)}$$

where the index set $R^k = \{1, \dots, k\}$, UBD is the current best upper bound for (P), and LBD is the current best lower bound for (P).

Proposition 2 *Problem (JRMP)^(l) is a valid lower bounding problem for Problem (P).*

Proof Since it is already known that Problem (BRMP)^(l) is a valid lower bounding problem and UBD and LBD are valid upper and lower bounds, we only need to prove that the cuts from Lagrangian subproblems together with the Benders optimality cuts do not exclude an optimal solution. Let obj_P be the optimal objective value of (P), then

$$\text{obj}_P = \sum_{\omega=1}^s \text{obj}_{\text{PP}_\omega}(x_0),$$

where

$$\text{obj}_{\text{PP}_\omega}(x_0) = \min\{c_\omega^T x_\omega : x_0 = H_\omega x_\omega, A_\omega x_\omega + B_\omega y_\omega \leq 0, x_\omega \in X_\omega, y_\omega \in Y_\omega\}.$$

On the one hand, $\forall \pi_\omega^i, i \in R^k$,

$$\begin{aligned}
 & \text{obj}_{\text{PP}_\omega}(x_0) \\
 & \geq \min\{c_\omega^T x_\omega + (\pi_\omega^i)^T (x_0 - H_\omega x_\omega) : A_\omega x_\omega + B_\omega y_\omega \leq z_\omega, x_\omega \in X_\omega, y_\omega \in Y_\omega\} \tag{1} \\
 & = \text{obj}_{\text{LS}_\omega^i} + (\pi_\omega^i)^T x_0.
 \end{aligned}$$

On the other hand,

$$\text{obj}_{\text{PP}_\omega}(x_0) = \min_{y_\omega \in Y_\omega} v_\omega(x_0, y_\omega),$$

where $v_\omega(x_0, y_\omega) = \min\{c_\omega^T x_\omega : x_0 = H_\omega x_\omega, A_\omega x_\omega + B_\omega y_\omega \leq 0\}$. From weak duality, $\forall j \in T^{(l)}$,

$$\begin{aligned} v_\omega(x_0, y_\omega) &\geq \min\{c_\omega^T x_\omega + (\lambda_\omega^{(j)})^T(A_\omega x_\omega + B_\omega y_\omega) + (\mu_\omega^{(j)})^T(x_0 - H_\omega x_\omega) : x_\omega \in X_\omega\} \\ &= \text{obj}_{\text{BPP}_\omega^{(j)}} + (\lambda_\omega^{(j)})^T B_\omega(y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^T(x_0 - x_0^{(j)}). \end{aligned}$$

Therefore, $\forall y_\omega \in Y_\omega$,

$$\text{obj}_{\text{PP}_\omega}(x_0) \geq \text{obj}_{\text{BPP}_\omega^{(j)}} + (\lambda_\omega^{(j)})^T B_\omega(y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^T(x_0 - x_0^{(j)}). \tag{2}$$

Equations (1)–(2) indicate that the cuts from Lagrangian subproblems together with the Benders optimality cuts do not exclude an optimal solution of (P). \square

For convenience, we call the cuts from the Lagrangian subproblems, Lagrangian cuts. The Benders cuts and the Lagrangian cuts in (JRMP^(l)) imply that, $\forall i \in R^k$,

$$UBD \geq \eta_0 \geq \sum_{\omega=1}^s \eta_\omega \geq \sum_{\omega=1}^s \text{obj}_{\text{LS}_\omega^i} + \sum_{\omega=1}^s (\pi_\omega^i)^T x_0.$$

Therefore, from each iteration i we can construct the following valid cut

$$UBD \geq \sum_{\omega=1}^s \text{obj}_{\text{LS}_\omega^i} + \sum_{\omega=1}^s (\pi_\omega^i)^T x_0, \tag{*}$$

Since $x_0 = x_\omega$ in the original problem, the above cut is also valid if x_0 is replaced with x_ω . Consequently, problems (LS₀^k), (LS_ω^k), (RPMP^k) can be enhanced with the constraint as:

$$\begin{aligned} &\min_{x_\omega, y_\omega} c_\omega^T x_\omega - (\pi_\omega^k)^T H_\omega x_\omega \\ &\text{s.t. } A_\omega x_\omega + B_\omega y_\omega \leq 0, \\ &UBD \geq \sum_{\hat{\omega}=1}^s \text{obj}_{\text{LS}_{\hat{\omega}}^i} + \sum_{\hat{\omega}=1}^s (\pi_{\hat{\omega}}^i)^T x_\omega, \quad \forall i \in R^{k-1}, \\ &x_\omega \in X_\omega, y_\omega \in Y_\omega. \end{aligned} \tag{LS}_\omega^k$$

$$\begin{aligned} &\min_{x_0} \sum_{\omega=1}^s (\pi_\omega^k)^T x_0 \\ &\text{s.t. } UBD \geq \sum_{\hat{\omega}=1}^s \text{obj}_{\text{LS}_{\hat{\omega}}^i} + \sum_{\hat{\omega}=1}^s (\pi_{\hat{\omega}}^i)^T x_0, \quad \forall i \in R^{k-1}, \\ &x_0 \in X_0. \end{aligned} \tag{LS}_0^k$$

$$\begin{aligned}
 & \min_{\substack{x_0, \theta_1^{[1]}, \dots, \theta_s^{[1]} \\ x_1, \dots, x_s}} \sum_{\omega=1}^s c_{\omega}^T x_{\omega} \\
 \text{s.t. } & x_0 = H_{\omega} x_{\omega}, \quad \forall \omega \in \{1, \dots, s\}, \\
 & A_{\omega} x_{\omega} + B_{\omega} \sum_{i \in I^k} \theta_{\omega}^{[i]} y_{\omega}^{[i]} \leq 0, \quad \forall \omega \in \{1, \dots, s\}, \\
 & \sum_{i \in I^k} \theta_{\omega}^{[i]} = 1, \quad \theta_{\omega}^{[i]} \geq 0, \quad \forall i \in I^k, \quad \forall \omega \in \{1, \dots, s\}, \\
 & UBD \geq \sum_{\hat{\omega}=1}^s \text{obj}_{LS}^{\hat{\omega}} + \sum_{\hat{\omega}=1}^s (\pi_{\hat{\omega}}^i)^T x_{\omega}, \quad \forall i \in R^{k-1}, \\
 & x_0 \in X_0, \quad x_{\omega} \in X_{\omega}, \quad \forall \omega \in \{1, \dots, s\},
 \end{aligned} \tag{RPMP}^k$$

Note that the index set I^k includes indices for all constant points $y_{\omega}^{[i]}$ in Problem (RPMP)^k, and the constant points $y_{\omega}^{[i]}$ come from all previously solved PP, FP, LS and JRMP.

3.4 The basic joint decomposition algorithm

Table 1 shows the basic JD algorithm. As described in Sect. 3.1, a JD iteration always include a LD iteration and sometimes a GBD iteration as well. We index JD and LD iterations using k and GBD iterations using l . Whether a GBD iteration is performed at JD iteration k depends on whether LD iteration k improves over LD iteration $k - 1$ substantially, i.e., whether $\text{obj}_{LS}^k \geq \text{obj}_{LS}^{k-1} + \epsilon$. In the first JD iteration, subproblem (PP_ω) is constructed by fixing x_0 to its initial value, and in any subsequent JD iteration, (PP_ω) is constructed by fixing x_0 to the optimal value of x_0 for (RPMP) in the previous JD iteration. Note that the solution of (LS₀) is only used for constructing a valid lower bound, and the x_0 value in the solution of (LS₀) is not used for constructing (PP_ω). The JD algorithm has the following property.

Proposition 3 *The JD algorithm shown in Table 1 cannot perform an infinite number of LD iterations between two GBD iterations.*

Proof The initial point $(x_0^1, y_1^{[1]}, \dots, y_s^{[1]})$ that are feasible for Problem (P) can lead to a finite upper bound UBD . According to Assumption 1, all Lagrangian subproblems are bounded, so between two GBD iterations, the first LD iteration leads to a finite obj_{LS} , and the subsequent LD iterations increase obj_{LS} by at least $\epsilon > 0$ (because otherwise a GBD iteration has to be performed). Therefore, in a finite number LD iterations either obj_{LS} exceeds $UBD - \epsilon$ and the algorithm terminates with an ϵ -optimal solution, or a GBD iteration is performed. This completes the proof. □

Remark 4 If an initial feasible point for Problem (P) is not known, the initial feasibility problem (IFP) can be solved to get a feasible point for (P) or verify that Problem (P) is infeasible (when the optimal objective value of Problem (IFP) is positive). Note that it is easy to find a feasible point of Problem (IFP).

In the JD algorithm, we use k to index both a JD iteration and a LD iteration, as every JD iteration includes one LD iteration. We use l (together with '()') to index a GBD iteration, and usually $l < k$ because not every JD iteration includes one GBD iteration. We use i (together with '[]') to index the columns generated for constructing Problem (RPMP)^k. Next, we establish the finite convergence property of the JD algorithm.

Table 1 The basic joint decomposition algorithm

Initialization

(I.a) Select $x_0^1, y_1^{[1]}, \dots, y_s^{[1]}$ that are feasible for Problem (P).

(I.b) Give termination tolerance $\epsilon > 0$. Let index sets $T^1 = S^1 = R^1 = \emptyset, I^1 = \{1\}$, iteration counter $k = 1, i = 1, l = 1$, bounds $UBD = +\infty, LBD = -\infty$.

LD Iteration

(1.a) Solve Problem (PP_ω^k) . If Problem (PP_ω^k) is infeasible, solve Problem (FP_ω^k) . Let the solution obtained be (x_ω^k, y_ω^k) , and update $i = i + 1, I^k = I^k \cup \{i\}, (y_1^{[i]}, \dots, y_s^{[i]}) = (y_1^k, \dots, y_s^k)$.

(1.b) Solve Problem (BPP_ω^k) by fixing $(x_0, y_1, \dots, y_s) = (x_0^k, y_1^k, \dots, y_s^k)$. If (BPP_ω^k) is feasible for all ω , generate Benders optimality cuts with the obtained dual solution μ_ω^k and λ_ω^k , and update $T^{k+1} = T^k \cup \{k\}$. If $\sum_{\omega=1}^s obj_{PP_\omega^k} < UBD$, update $UBD = \sum_{\omega=1}^s obj_{PP_\omega^k}$, and incumbent solution $(x_0^*, x_1^*, \dots, x_s^*, y_1^*, \dots, y_s^*) = (x_0^k, x_1^k, \dots, x_s^k, y_1^k, \dots, y_s^k)$. If Problem (BPP_ω^k) is infeasible for at least one ω , solve Problem (BFP_ω^k) . Generate Benders feasibility cuts with the obtained dual solution μ_ω^k and λ_ω^k , and update $S^{k+1} = S^k \cup \{k\}$.

(1.c) Solve Problem $(RPMP^k)$. Let $x_0^{k+1}, \{\theta_\omega^{[i,k]}\}_{i \in I^k, \omega \in \{1, \dots, s\}}$ be the optimal solution obtained, and π_1^k, \dots, π_s^k be Lagrange multipliers for the NACs.

(1.d) Solve Problems (LS_ω^k) and (LS_0^k) . If $obj_{LS^k} = \sum_{\omega=1}^s obj_{LS_\omega^k} + obj_{LS_0^k} > LBD$, update $LBD = obj_{LS^k}$. Generate a Lagrangian cut and update $R^{k+1} = R^k \cup \{k\}$. Update $i = i + 1, I^{k+1} = I^k \cup \{i\}, (y_1^{[i]}, \dots, y_s^{[i]}) = (y_1^k, \dots, y_s^k)$.

(1.e) If $UBD \leq LBD + \epsilon$, terminate and return the incumbent solution as an ϵ -optimal solution. If $obj_{LS^k} \geq obj_{LS^{k-1}} + \epsilon, k = k + 1$, go to step (1.a); otherwise $k = k + 1$ and go to step (2.a);

GBD Iteration

(2.a) Solve Problem $(JRMP^{(l)})$, and let the obtained solution be $(x_0^{(l)}, y_1^{(l)}, \dots, y_s^{(l)})$. Update $i = i + 1, I^k = I^k \cup \{i\}, (y_1^{[i]}, \dots, y_s^{[i]}) = (y_1^{(l)}, \dots, y_s^{(l)})$. If $obj_{JRMP^{(l)}} > LBD$, update $LBD = obj_{JRMP^{(l)}}$.

(2.b) Solve Problem $(BPP_\omega^{(l)})$ by fixing $(x_0, y_1, \dots, y_s) = (x_0^{(l)}, y_1^{(l)}, \dots, y_s^{(l)})$. If $(BPP_\omega^{(l)})$ is feasible for all ω , generate Benders optimality cuts with the dual solution μ_ω^k and λ_ω^k , and update $T^{(l+1)} = T^{(l)} \cup \{l\}$. If $\sum_{\omega=1}^s obj_{BPP_\omega^{(l)}} < UBD$, update $UBD = obj_{BPP_\omega^{(l)}}$ and the incumbent solution $(x_0^*, x_1^*, \dots, x_s^*, y_1^*, \dots, y_s^*) = (x_0^{(l)}, x_1^{(l)}, \dots, x_s^{(l)}, y_1^{(l)}, \dots, y_s^{(l)})$. If Problem $(BPP_\omega^{(l)})$ is infeasible for at least one ω , solve Problem $(BFP_\omega^{(l)})$. Generate Benders feasibility cuts with the obtained dual solution μ_ω^l and λ_ω^l , and update $S^{(l+1)} = S^{(l)} \cup \{l\}$.

(2.c) If $UBD \leq LBD + \epsilon$, terminate and return the incumbent solution as an ϵ -optimal solution; otherwise $l = l + 1$ and go to step (1.a).

Proposition 4 *If set X_ω is polyhedral $\forall \omega \in \{1, \dots, s\}$, the JD algorithm shown in Table 1 cannot perform an infinite number of GBD iterations.*

Proof In this case, the GBD part of the algorithm reduces to BD, and BD is known to have finite termination property [31,42]. The finite termination property results from:

- (a) The Benders master problem $(BRMP^{(l)})$ (and therefore $(JRMP^{(l)})$ as well) requires only a finite number of Benders cuts to equal Problem (P), due to linear duality theory;
- (b) A same Benders cut cannot be generated twice before the optimality gap is closed.

□

Proposition 5 *If $X_0 \times Y_1 \times \dots \times Y_s$ is a finite discrete set, the JD algorithm shown in Table 1 cannot perform an infinite number of GBD iterations.*

Proof This result comes from the fact that a point in $X_0 \times Y_1 \times \dots \times Y_s$ cannot be generated twice before the optimality gap is closed. For more details readers can see Theorem 2.4 of [32]. \square

Proposition 6 *The JD algorithm shown in Table 1 cannot include an infinite number of GBD iterations at which the Benders primal problem BPP is feasible.*

Proof A similar proposition has been proved in the context of GBD in [32] (as Theorem 2.5). The central idea of the proof can be used here for JD.

Suppose the JD algorithm includes an infinite number of GBD iterations at which the Benders primal problem BPP is feasible. Let superscript (n) index these GBD iterations, $\{(\eta_0^{(n)}, x_0^{(n)}, y_1^{(n)}, \dots, y_s^{(n)})\}$ be the sequence of optimal solutions of JRMP and $\{(\mu_\omega^{(n)}, \lambda_\omega^{(n)})\}$ be the sequence of dual solutions of BPP. Since $\{\eta_0^{(n)}\}$ is nondecreasing and is bounded from above, so a subsequence of it converges to a finite value, say η_0^* . Due to the compactness of X_0, Y_1, \dots, Y_s , a subsequence of $\{(x_0^{(n)}, y_1^{(n)}, \dots, y_s^{(n)})\}$, say, $\{(x_0^{(n_i)}, y_1^{(n_i)}, \dots, y_s^{(n_i)})\}$, converges to $(x_0^*, y_1^*, \dots, y_s^*) \in X_0 \times Y_1 \times \dots \times Y_s$. Solving BPP in this subsequence of GBD iterations can be viewed as point-to-set mappings from points in $X_0 \times Y_1 \times \dots \times Y_s$ to the relevant Lagrange multiplier sets. From Lemma 2.1 of [32] and Assumption 2, such a mapping is uniformly bounded in some open neighborhood of the point it maps from. Let such open neighborhood of $(x_0^*, y_1^*, \dots, y_s^*)$ be $N(x_0^*, y_1^*, \dots, y_s^*)$, then $\exists t$ such that $\forall n_i > t, (x_0^{(n_i)}, y_1^{(n_i)}, \dots, y_s^{(n_i)}) \in N(x_0^*, y_1^*, \dots, y_s^*)$, and then the relevant subsequence of Lagrange multipliers is bounded, which must contain a subsequence converging to $\{\mu_\omega^*, \lambda_\omega^*\}$. Therefore, there exists a subsequence of $\{(\eta_0^{(n)}, x_0^{(n)}, y_1^{(n)}, \dots, y_s^{(n)}, \mu_\omega^{(n)}, \lambda_\omega^{(n)})\}$, say, $\{(\eta_0^{(m)}, x_0^{(m)}, y_1^{(m)}, \dots, y_s^{(m)}, \mu_\omega^{(m)}, \lambda_\omega^{(m)})\}$, which converges to $\{(\eta_0^*, x_0^*, y_1^*, \dots, y_s^*, \mu_\omega^*, \lambda_\omega^*)\}$.

Consider any GBD iteration $m > 1$ in this convergent subsequence. Let UBD and LBD be the upper and lower bounds after this GBD iteration, then

$$\begin{aligned} obj_{BPP^{(m-1)}} &\geq UBD, \\ LBD &\geq \eta^{(m)}, \end{aligned}$$

and that the JD algorithm does not terminate after GBD iteration m implies

$$UBD > LBD + \epsilon,$$

therefore

$$obj_{BPP^{(m-1)}} > \eta^{(m)} + \epsilon. \tag{3}$$

According to how JRMP is constructed,

$$\begin{aligned} \eta^{(m)} &\geq obj_{BPP^{(m-1)}} + \\ &\sum_{\omega=1}^s \left[(\lambda_\omega^{(m-1)})^T B_\omega (y_\omega^{(m)} - y_\omega^{(m-1)}) + (\mu_\omega^{(m-1)})^T (x_0^{(m)} - x_0^{(m-1)}) \right]. \end{aligned} \tag{4}$$

Equations (3) and (4) imply that

$$0 > \sum_{\omega=1}^s \left[(\lambda_\omega^{(m-1)})^T B_\omega (y_\omega^{(m)} - y_\omega^{(m-1)}) + (\mu_\omega^{(m-1)})^T (x_0^{(m)} - x_0^{(m-1)}) \right] + \epsilon. \tag{5}$$

However, when m is sufficiently large, $y_\omega^{(m)} - y_\omega^{(m-1)}$ and $x_0^{(m)} - x_0^{(m-1)}$ are sufficiently close to 0 while $\mu_\omega^{(m-1)}$ and $\lambda_\omega^{(m-1)}$ are sufficiently close to limit points μ_ω^* and λ_ω^* , so the right-hand-side of Eq. (5) is a positive value (as $\epsilon > 0$). This contradiction implies that the JD algorithm cannot include an infinite number of GBD iterations at which BPP is feasible. \square

Theorem 1 *With an initial feasible point, the JD algorithm shown in Table 1 terminates in a finite number of iterations with an ϵ -optimal solution, if one the following three conditions is satisfied:*

- (a) *Set X_ω is polyhedral $\forall \omega \in \{1, \dots, s\}$.*
- (b) *Set $X_0 \times Y_1 \times \dots \times Y_s$ is finite discrete.*
- (c) *There are only a finite number of GBD iterations at which the Benders primal problem BPP is infeasible.*

Proof From Proposition 3, the JD algorithm can only include a finite number of LD iterations. From Propositions 4 and 5, when condition (a) or (b) is satisfied, the JD algorithm can only include a finite number of BD iterations. From Proposition 6, the JD algorithm can only have a finite number of GBD iterations at which the Benders primal problem BPP is feasible, and together with condition (c), it implies that the JD algorithm can only include a finite number of BD iterations. Therefore, if one of the three conditions is satisfied, the JD algorithm can only include a finite number LD and BD iterations before termination.

On the other hand, according to Proposition 2, the JD algorithm never excludes an optimal solution. This together with the termination criterion ensures that the solution returned is ϵ -optimal. \square

Remark 5 Condition (c) in Theorem 1 is actually not a restrictive condition, because we can always “soften” the complicating constraints in Problem (P) (i.e., penalize the violation of these constraints in the objective function) so that Problem (BPP $_\omega^{(l)}$) is always feasible.

4 Enhancements to joint decomposition

The solution of Problem (JRMP $^{(l)}$) is the bottleneck of the JD algorithm, even considering that the problem is solved only when necessary. Problem (JRMP $^{(l)}$) is challenging due to two major reasons. One is that the number of complicating variables in Problem (JRMP $^{(l)}$) is dependent on the number of scenarios, so the size of Problem (JRMP $^{(l)}$) is large (although smaller than the original problem). The other is that the number of constraints in the problem grows with the JD iteration; in other words, Problem (JRMP $^{(l)}$) becomes more and more challenging as JD progresses. In this section, we introduce two ways to mitigate the difficulty in solving Problem (JRMP $^{(l)}$):

1. To solve a convex relaxation of Problem (JRMP $^{(l)}$) before solving Problem (JRMP $^{(l)}$). If the solution of the convex relaxation can improve the lower bound, then skip solving Problem (JRMP $^{(l)}$).
2. To perform domain reduction iteratively in JD in order to keep reducing the ranges of the complicating variables. This way, the convex relaxation of Problem (JRMP $^{(l)}$) is progressively tightened and Problem (JRMP $^{(l)}$) itself does not become much harder as the algorithm progresses.

In addition, domain reduction for the complicating variables can make other nonconvex JD subproblems easier, including Problems (LS $_\omega^k$) and (PP $_\omega^k$). Domain reduction for the

linking variables can also tighten the Lagrangian relaxation gap [41]; in extreme cases, the Lagrangian relaxation gap can diminish and there is no need to solve Problem (JRMP^(l)) in JD to close the optimality gap. Note that we do not perform domain reduction for non-complicating variables, because normally reducing ranges on these variables do not help much to tighten convex relaxations and ease the solution of nonconvex subproblems.

4.1 Convex relaxation and domain reduction

The convex relaxation of Problem (JRMP^(l)) is a valid lower bounding problem for Problem (JRMP^(l)) and consequently for Problem (P) as well. It can be written as:

$$\begin{aligned}
 & \min_{\substack{x_0, \eta_0, \eta_1, \dots, \eta_s \\ y_1, \dots, y_s}} \eta_0 \\
 \text{s.t. } & \eta_0 \geq \sum_{\omega=1}^s \eta_\omega, \\
 & \eta_\omega \geq \text{obj}_{\text{BPP}_\omega^{(j)}} + (\lambda_\omega^{(j)})^T B_\omega (y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^T (x_0 - x_0^{(j)}), \\
 & \qquad \qquad \qquad \forall \omega \in \{1, \dots, s\}, \quad \forall j \in T^{(l)}, \\
 & 0 \geq \text{obj}_{\text{BFP}_\omega^{(j)}} + (\lambda_\omega^{(j)})^T B_\omega (y_\omega - y_\omega^{(j)}) + (\mu_\omega^{(j)})^T (x_0 - x_0^{(j)}), \\
 & \qquad \qquad \qquad \forall \omega \in \{1, \dots, s\}, \quad \forall j \in S^{(l)}, \\
 & \eta_\omega \geq \text{obj}_{\text{BPP}_\omega^j} + (\lambda_\omega^j)^T B_\omega (y_\omega - y_\omega^j) + (\mu_\omega^j)^T (x_0 - x_0^j), \\
 & \qquad \qquad \qquad \forall \omega \in \{1, \dots, s\}, \quad \forall j \in T^k, \\
 & 0 \geq \text{obj}_{\text{BFP}_\omega^j} + (\lambda_\omega^j)^T B_\omega (y_\omega - y_\omega^j) + (\mu_\omega^j)^T (x_0 - x_0^j), \\
 & \qquad \qquad \qquad \forall \omega \in \{1, \dots, s\}, \quad \forall j \in S^k, \\
 & \eta_0 \leq UBD, \\
 & \eta_0 \geq LBD, \\
 & \eta_\omega \geq \text{obj}_{\text{LS}_\omega^i} + (\pi_\omega^i)^T x_0, \quad \forall \omega \in \{1, \dots, s\}, \quad \forall i \in R^k, \\
 & x_0 \in \hat{X}_0, \quad y_\omega \in \hat{Y}_\omega, \quad \forall \omega \in \{1, \dots, s\}.
 \end{aligned} \tag{JRMPR}^{(l)}$$

Here \hat{X}_0 and \hat{Y}_ω denote the convex relaxations of X_0 and Y_ω . Let $\text{obj}_{\text{JRMPR}^{(l)}}$ be the optimal objective of Problem (JRMPR^(l)).

Since Problem (JRMPR^(l)) is also a valid convex relaxation of Problem (P), the solution of Problem (JRMPR^(l)) can be exploited to eliminate the parts of variable ranges that cannot include an optimal solution of Problem (P), using marginal based domain reduction method. This method was first proposed in [22] (and it was called range reduction therein). The following proposition lays the foundation of marginal based domain reduction for complicating variables y_ω in JD, which results directly from Theorem 2 in [22].

Proposition 7 Consider the following bounds on $y_{\omega,j}$ ($\forall \omega \in \{1, \dots, s\}, \quad \forall j \in \{1, \dots, n_y\}$):

$$\begin{aligned}
 & y_{\omega,j} - y_{\omega,j}^{up} \leq 0, \\
 & y_{\omega,j}^{lo} - y_{\omega,j} \leq 0,
 \end{aligned}$$

whose Lagrange multipliers obtained at the solution of Problem (JRM $PR^{(l)}$) are $u_{\omega,j}, v_{\omega,j}$. Let $\mathbb{J}_{1,\omega}^{(l)}$ include indices of upper bounds whose $u_{\omega,j}$ are nonzero, and $\mathbb{J}_{1,\omega}^{(2)}$ include indices of lower bounds whose $v_{\omega,j}$ are nonzero, then the following constraints do not exclude an optimal solution of (P):

$$y_{\omega,j} \geq y_{\omega,j}^{up} - \frac{(UBD - obj_{JRM\mathit{P}R^{(l)}})}{u_{\omega,j}}, \quad \forall j \in \mathbb{J}_{1,\omega}^{(l)}, \quad \forall \omega \in \{1, \dots, s\},$$

$$y_{\omega,j} \leq y_{\omega,j}^{lo} + \frac{(UBD - obj_{JRM\mathit{P}R^{(l)}})}{v_{\omega,j}}, \quad \forall j \in \mathbb{J}_{2,\omega}^{(l)}, \quad \forall \omega \in \{1, \dots, s\}.$$

The following proposition states a similar result for the linking variables x_0 :

Proposition 8 Consider the following bounds on $x_{0,j}$ ($\forall j \in \{1, \dots, n_0\}$):

$$x_{0,j} - x_{0,j}^{up} \leq 0,$$

$$x_{0,j}^{lo} - x_{0,j} \leq 0,$$

whose Lagrange multipliers obtained at the solution of Problem (JRM $PR^{(l)}$) are $u_{0,j}, v_{0,j}$. Let $\mathbb{J}_{1,0}^{(l)}$ include indices of upper bounds whose $u_{0,i}$ are nonzero, and $\mathbb{J}_{2,0}^{(l)}$ include indices of lower bounds whose $v_{0,i}$ are nonzero, then the following constraints do not exclude an optimal solution of (P):

$$x_{0,j} \geq x_{0,j}^{up} - \frac{(UBD - obj_{JRM\mathit{P}R^{(l)}})}{u_{0,j}}, \quad \forall j \in \mathbb{J}_{1,0}^{(l)}$$

$$x_{0,j} \leq x_{0,j}^{lo} + \frac{(UBD - obj_{JRM\mathit{P}R^{(l)}})}{v_{0,j}}, \quad \forall j \in \mathbb{J}_{2,0}^{(l)}$$

According to Propositions 7 and 8, the bounds of nonconvex and linking variables can be updated via the following range reduction calculation:

$$y_{\omega,j}^{up} = \min \left\{ y_{\omega,j}^{up}, y_{\omega,j}^{lo} + \frac{G^{(l)}}{u_{\omega,j}} \right\}, \quad \forall j \in \mathbb{J}_{1,\omega}^{(l)}, \quad \forall \omega \in \{1, \dots, s\},$$

$$y_{\omega,j}^{lo} = \max \left\{ y_{\omega,j}^{lo}, y_{\omega,j}^{up} - \frac{G^{(l)}}{v_{\omega,j}} \right\}, \quad \forall j \in \mathbb{J}_{2,\omega}^{(l)}, \quad \forall \omega \in \{1, \dots, s\},$$

$$x_{0,j}^{up} = \min \left\{ x_{0,j}^{up}, x_{0,j}^{lo} + \frac{G^{(l)}}{u_{0,j}} \right\}, \quad \forall j \in \mathbb{J}_{1,0}^{(l)},$$

$$x_{0,j}^{lo} = \max \left\{ x_{0,j}^{lo}, x_{0,j}^{up} - \frac{G^{(l)}}{v_{0,j}} \right\}, \quad \forall j \in \mathbb{J}_{2,0}^{(l)},$$

(MDR $^{(l)}$)

where $G^{(l)} = UBD - obj_{RM\mathit{P}CR^{(l)}}$.

The effectiveness of marginal based domain reduction relies on how many bounds are active, the magnitude of Lagrange multipliers of active bounds at the solution of JRM $PR^{(l)}$, and how often JRM $PR^{(l)}$ is solved. In order to achieve effective domain reduction more consistently, we also introduce optimization based domain reduction in JD. Optimization based domain reduction, or called bound contraction or bound tightening [21,45], is to maximize or minimize a single variable over a convex relaxation of the feasible set of the original problem.

For example, if we are to estimate the upper bound of a linking variable $x_{0,j}$ at JD iteration k , we can solve the following optimization problem:

$$\begin{aligned}
 & \max_{\substack{x_0, x_1, \dots, x_s \\ y_1, \dots, y_s}} x_{0,i} \\
 \text{s.t. } & x_0 = H_\omega x_\omega, \quad \forall \omega \in \{1, \dots, s\}, \\
 & A_\omega x_\omega + B_\omega y_\omega \leq 0, \quad \forall \omega \in \{1, \dots, s\}, \\
 & \sum_{\omega=1}^s c_\omega^T x_\omega \leq UBD, \\
 & x_0 \in X_0^k, \\
 & x_\omega \in X_\omega, \quad y_\omega \in \hat{Y}_\omega^k, \quad \forall \omega \in \{1, \dots, s\}.
 \end{aligned}
 \tag{ODRStd_i^k}$$

The third group of constraints in Problem (ODRStd_i^k) utilizes the known upper bound of (P) to tighten the convex relaxation, but it cannot be included in Problem (ODRStd_i^k) when UBD is not available (e.g., before a feasible solution of (P) is known). We now index sets X_0, \hat{Y}_ω with the JD iteration number k , as these sets may change after the domain reduction calculations.

Problem (ODRStd_i^k) represents the standard optimization based domain reduction formulation, but it can be further enhanced in the JD algorithm, via the incorporation of valid cuts derived from other JD subproblems. First, we can add the following constraint:

$$\sum_{\omega=1}^s c_\omega^T x_\omega \geq LBD.$$

This constraint is redundant in the classical branch-and-bound based global optimization, as LBD is obtained via convex relaxation as well. In JD, LBD is obtained via Lagrangian subproblems and JD relaxed master problems, which may be tighter than convex relaxations of the original problem, so this constraint may enhance Problem (ODRStd_i^k). Second, we can include constraints (*) (that are derived from Problem (JRMP^(l))). Therefore, we can write the enhanced optimization based domain reduction formulation as:

$$\begin{aligned}
 & \min_{\substack{x_0, x_1, \dots, x_s \\ y_1, \dots, y_s}} / \max_{\substack{x_0, x_1, \dots, x_s \\ y_1, \dots, y_s}} x_{0,i} \\
 \text{s.t. } & x_0 = H_\omega x_\omega, \quad \forall \omega \in \{1, \dots, s\}, \\
 & A_\omega x_\omega + B_\omega y_\omega \leq 0, \quad \forall \omega \in \{1, \dots, s\}, \\
 & \sum_{\omega=1}^s c_\omega^T x_\omega \leq UBD, \\
 & \sum_{\omega=1}^s c_\omega^T x_\omega \geq LBD, \\
 & UBD \geq \sum_{\omega=1}^s obj_{LS_\omega}^i + \sum_{\omega=1}^s (\pi_\omega^i)^T x_0, \quad \forall i \in R^k, \\
 & x_0 \in X_0^k, \\
 & x_\omega \in X_\omega, \quad y_\omega \in \hat{Y}_\omega^k, \quad \forall \omega \in \{1, \dots, s\}.
 \end{aligned}
 \tag{ODR_i^k}$$

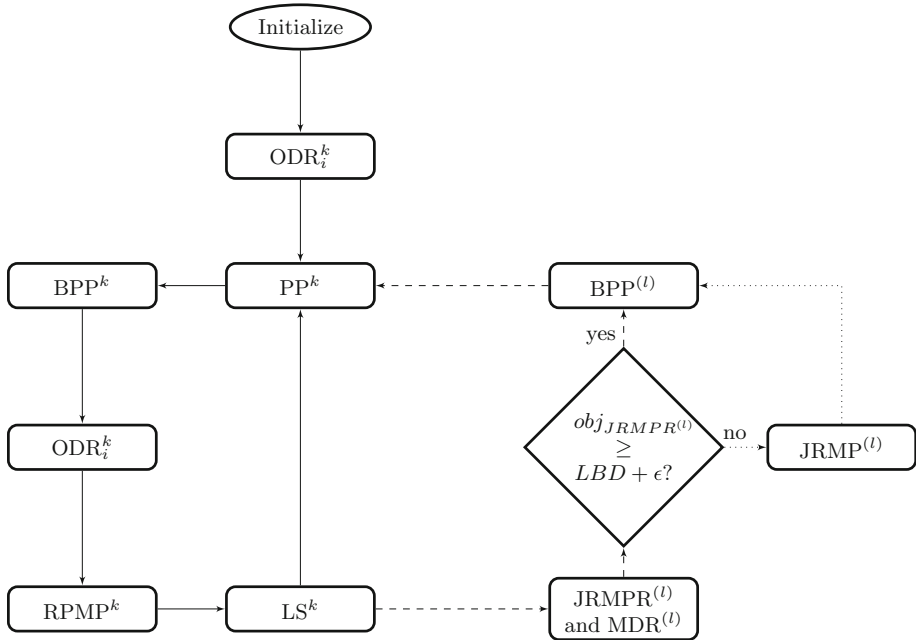


Fig. 2 The enhanced joint decomposition framework

If we are to estimate an upper bound, then Problem (ODR_i^k) is a maximization problem; otherwise, Problem (ODR_i^k) is a minimization problem.

Although Problem (ODR_i^k) is convex, it can have a very large size because its size grows with the number of scenarios. Therefore, we proposed to solve Problem (ODR_i^k) for x_0 but not for y_ω . Actually, we can see in the case study section that optimization based domain reduction is time consuming even when we only solve Problem (ODR_i^k) for x_0 .

4.2 The enhanced joint decomposition method

Figure 2 shows the framework of the JD method that includes solving convex relaxation, Problem $(JRMPR^{(l)})$, bound tightening for x_0 and the domain reduction calculations. In this framework, optimization based domain reduction is performed at the beginning of the algorithm and in every LD iteration (right before the solution of nonconvex Lagrangian subproblems). Convex relaxation, Problem $(JRMPR^{(l)})$, is solved before solving Problem $(JRMP^{(l)})$, and after solving Problem $(JRMPR^{(l)})$, marginal based domain reduction is performed. Problem $(JRMP^{(l)})$ is not solved if Problem $(JRMPR^{(l)})$ can improve the lower bound significantly; this strategy can postpone solving Problem $(JRMP^{(l)})$ to a later time, so that the ranges of x_0 can be reduced as much as possible when a Problem $(JRMP^{(l)})$ has to be solved. The detailed algorithm for the enhanced JD is shown in Table 2.

Theorem 2 *The decomposition algorithm described in Table 2 terminates in a finite number of steps with an ϵ -optimal solution of Problem (P), if one the following three conditions is satisfied:*

- (a) Set X_ω is polyhedral $\forall \omega \in \{1, \dots, s\}$.

Table 2 Enhanced joint decomposition method—enhancement is in bold font

Initialization

- (I.a) Select $x_0^1, y_1^{[1]}, \dots, y_s^{[1]}$ that are feasible for Problem (P).
- (I.b) Give termination tolerance $\epsilon > 0$. Let index sets $T^1 = S^1 = R^1 = \emptyset, I^1 = \{1\}$, iteration counter $k = 1, i = 1, l = 1$, bounds $UBD = +\infty, LBD = -\infty$.
- (I.c) **Solve Problem (ODR_i^k) to update bounds of all $x_{0,i}$.**

LD Iteration

- (1.a) Solve Problem (PP_ω^k). If Problem (PP_ω^k) is infeasible, solve Problem (FP_ω^k). Let the solution obtained be $(x_ω^k, y_ω^k)$, and update $i = i + 1, I^k = I^k \cup \{i\}, (y_1^{[i]}, \dots, y_s^{[i]}) = (y_1^k, \dots, y_s^k)$.
- (1.b) Solve Problem (BPP_ω^k) by fixing $(x_0, y_1, \dots, y_s) = (x_0^k, y_1^k, \dots, y_s^k)$. If (BPP_ω^k) is feasible for all ω , generate Benders optimality cuts with the obtained dual solution $\mu_ω^k$ and $\lambda_ω^k$, and update $T^{k+1} = T^k \cup \{k\}$. If $\sum_{\omega=1}^s obj_{PP_ω^k} < UBD$, update $UBD = \sum_{\omega=1}^s obj_{PP_ω^k}$, and incumbent solution $(x_0^*, x_1^*, \dots, x_s^*, y_1^*, \dots, y_s^*) = (x_0^k, x_1^k, \dots, x_s^k, y_1^k, \dots, y_s^k)$. If Problem (BPP_ω^k) is infeasible for at least one ω , solve Problem (BFP_ω^k). Generate Benders feasibility cuts with the obtained dual solution $\mu_ω^k$ and $\lambda_ω^k$, and update $S^{k+1} = S^k \cup \{k\}$.
- (1.c) **Solve Problem (ODR_i^k) to update bounds of all $x_{0,i}$.**
- (1.d) Solve Problem (RPMP^k). Let $x_0^{k+1}, \{\theta_{\omega}^{[i,k]}\}_{i \in I^k, \omega \in \{1, \dots, s\}}$ be the optimal solution obtained, and π_1^k, \dots, π_s^k be Lagrange multipliers for the NACs.
- (1.e) Solve Problems (LS_ω^k) and (LS₀^k). If $obj_{LS^k} = \sum_{\omega=1}^s obj_{LS_ω^k} + obj_{LS_0^k} > LBD$, update $LBD = obj_{LS^k}$. Generate a Lagrangian cut and update $R^{k+1} = R^k \cup \{k\}$. Update $i = i + 1, I^{k+1} = I^k \cup \{i\}, (y_1^{[i]}, \dots, y_s^{[i]}) = (y_1^k, \dots, y_s^k)$.
- (1.f) If $UBD \leq LBD + \epsilon$, terminate and return the incumbent solution as an ϵ -optimal solution. If $obj_{LS^k} \geq obj_{LS^{k-1}} + \epsilon, k = k + 1$, go to step (1.a); otherwise $k = k + 1$ and go to step (2.a);

GBD Iteration

- (2.a) **Solve Problem (JRMPR^(l)), and then perform marginal based domain reduction (MDR^(l)). If $obj_{JRMPR^{(l)}} \geq LBD + \epsilon$, let the obtained solution be $(x_0^{(l)}, y_1^{(l)}, \dots, y_s^{(l)})$, update $LBD = obj_{JRMPR^{(l)}}, i = i + 1, I^{k+1} = I^k \cup \{i\}, (y_1^{[i]}, \dots, y_s^{[i]}) = (y_1^{(l)}, \dots, y_s^{(l)})$, go to step (2.c). Otherwise, go to set (2.b).**
- (2.b) Solve Problem (JRMP^(l)), and let the obtained solution be $(x_0^{(l)}, y_1^{(l)}, \dots, y_s^{(l)})$. Update $i = i + 1, I^{k+1} = I^k \cup \{i\}, (y_1^{[i]}, \dots, y_s^{[i]}) = (y_1^{(l)}, \dots, y_s^{(l)})$. If $obj_{RMP^{(l)}} > LBD$, update $LBD = obj_{RMP^{(l)}}$.
- (2.c) Solve Problem (BPP_ω^(l)) by fixing $(x_0, y_1, \dots, y_s) = (x_0^{(l)}, y_1^{(l)}, \dots, y_s^{(l)})$. If (BPP_ω^(l)) is feasible for all ω , generate Benders optimality cuts with the dual solution $\mu_ω^k$ and $\lambda_ω^k$, and update $T^{(l+1)} = T^{(l)} \cup \{l\}$. If $\sum_{\omega=1}^s obj_{BPP_ω^{(l)}} < UBD$, update $UBD = obj_{BPP^{(l)}}$ and the incumbent solution $(x_0^*, x_1^*, \dots, x_s^*, y_1^*, \dots, y_s^*) = (x_0^{(l)}, x_1^{(l)}, \dots, x_s^{(l)}, y_1^{(l)}, \dots, y_s^{(l)})$. If Problem (BPP_ω^(l)) is infeasible for at least one ω , solve Problem (BFP_ω^(l)). Generate Benders feasibility cuts with the obtained dual solution $\mu_ω^l$ and $\lambda_ω^l$, and update $S^{(l+1)} = S^{(l)} \cup \{l\}$.
- (2.d) If $UBD \leq LBD + \epsilon$, terminate and return the incumbent solution as an ϵ -optimal solution; otherwise $l = l + 1$, go to step (1.a).

- (b) Set $X_0 \times Y_1 \times \dots \times Y_s$ is finite discrete.
- (c) There are only a finite number of GBD iterations at which the Benders primal problem BPP is infeasible.

Proof This can be proved by showing that, solving Problem (JRMPR^(l)) in every GBD iteration in JD and including domain reduction calculations do not invalidate the finite termination to an ϵ -optimal solution.

First, we can show that there cannot be an infinite number of GBD iterations at which Problem (JRMPR^(l)) is solved but Problem (JRMP^(l)) is not solved. Consider a GBD iteration at which Problem (JRMPR^(l)) is solved but Problem (JRMP^(l)) is not solved, then Problem (JRMPR^(l)) is not unbounded (because otherwise Problem (JRMP^(l)) needs to be solved) and the lower bound LBD is finite. The upper bound UBD is also finite (because an initial feasible solution exists). Therefore, it is not possible that LBD can be improved by $\epsilon > 0$ for an infinite number of GBD iterations, so there cannot be an infinite number of GBD iterations at which Problem (JRMPR^(l)) is solved but Problem (JRMP^(l)) is not solved. According to the proof of Theorem 1, JD can only include a finite number of LD iterations, and a finite number of GBD iterations at which Problem (JRMP^(l)) is solved, if one of the three listed conditions are satisfied.

Second, domain reduction reduces the ranges of x_0 and y_1, \dots, y_s but does not exclude any optimal solution from the reduced ranges. So the Lagrangian relaxation problems and JD relaxation master problems are still valid lower bounding problems and they cannot cut off any optimal solution. \square

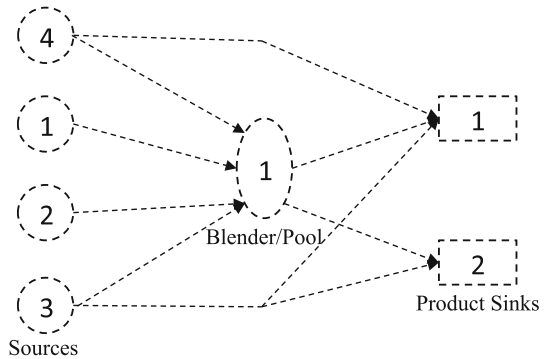
5 Case studies

The purpose of the case studies is to demonstrate the potential computational advantages of the proposed joint decomposition method for problems exhibiting the decomposable structure of (P0), especially when off-the-shelf solvers cannot effectively exploit the problem structure. We consider two case study problems here, which are both scenario-based two-stage stochastic nonconvex MINLPs arising from integrated design and operation under uncertainty.

5.1 Case study problems

Case Study A—This problem is a variant of the stochastic Haverly pooling problem [3], which was originally developed based on the classical Haverly pooling problem [46,47]. Figure 3 shows the superstructure of the pooling system to be developed. The circles denote four sources that supply intermediate gasoline products with different sulfur percentages and costs, the ellipse denotes a blender (or called a pool) at which some intermediate products can be blended, and the rectangles denote product sinks at which the final products are blended. The goal of optimization is to minimize the negative profit of the system by determining: (1) Whether the pool and the two product sinks are to be developed in the system; (2) The capacities of the sources and the pipelines. The stochastic pooling model of the problem can be found in “Appendix B”. Two uncertain parameters, percentage of sulfur in source 4 and upper limit on the demand at sink 1, were considered. They were assumed to follow independent normal distributions, with means of 2.5 and 180 and standard deviations of 0.08 and 10, respectively. Other parameters used in the problem can be found in [3]. For this problem, x_0 contains 3 binary variables and 13 continuous variables, x_ω contains $7s$ continuous variables and y_ω contains $14s$ continuous variables, where s stands for the total number of scenarios. In the case study, each uncertain parameter was sampled for 5, 6, 7,

Fig. 3 Superstructure of case study A problem



8, 9 and 10 scenario values, via the sampling rule described in [3], and this led to problem instances with 25, 36, 49, 64, 81 and 100 scenarios.

Case Study B—This problem is a variant of the Sarawak Gas Production System (SGPS) design problem [48], and the original form of the design problem appeared in [3]. Figure 4 shows the superstructure of the SGPS system under consideration, where the circles represent gas fields (sources), ellipses represent offshore gas platforms (pools) at which gas flows from different gas fields are mixed and split, rectangles represent onshore liquefied natural gas (LNG) plants (product terminals). Symbols with solid lines represent the part of the system that is already developed, and symbols with dashed lines represent the superstructure of the part of the system that needs to be designed in the problem. The goal of optimization is to maximize expected net present value while satisfying specifications for gas qualities at the LNG plants in the presence of uncertainty. There are two uncertain parameters, i.e., the quality of CO₂ at gas field M1 and upper limit on the demand at LNG plant 2. They were assumed to follow independent normal distributions with means of 3.34% and 2155 Mmol/day and standard deviations of 1% and 172.5 Mmol/day, respectively. In the case study, each uncertain parameter was sampled for 5, 6, 7, 8, 9 and 10 scenario values, via the same sampling rule described in [3], which led to problem instances with 25, 36, 49, 64, 81 and 100 scenarios. The problem was also formulated following the new stochastic pooling model provided in “Appendix B”. In the resulting formulation, x_0 contains 5 binary variables and 29 continuous variables. The 5 binary variables are to determine whether gas fields HL, SE, M3, M1 and JN are to be developed, and the 29 continuous variables are the capacities of other units to be developed. x_ω contains $8s$ variables and y_ω contains $85s$ variables, where s stands for the total number of scenarios.

5.2 Solution approaches and implementation

The case studies were run on a virtual machine allocated with a 3.2GHz CPU. The virtual machine ran Linux operating system (Ubuntu 16.04) with 6 GB of memory. Four solution approaches were compared in the case studies: Monolith, GBD, JD1, JD2. Monolith refers to solving the problem using an off-the-shelf, general-purpose global optimization solver, GBD refers to generalized Benders decomposition, JD1 refers to the basic JD algorithm, and JD2 refers to the enhanced JD algorithm. The case study problems and the subproblems required in GBD, JD1 and JD2 were all modeled on GAMS 24.7.4 [49], but GBD, JD1 and JD2 algorithms were programmed on MATLAB 2014a [50]. Data exchange between MATLAB and GAMS was realized via GAMS GDXMRW facility [51].

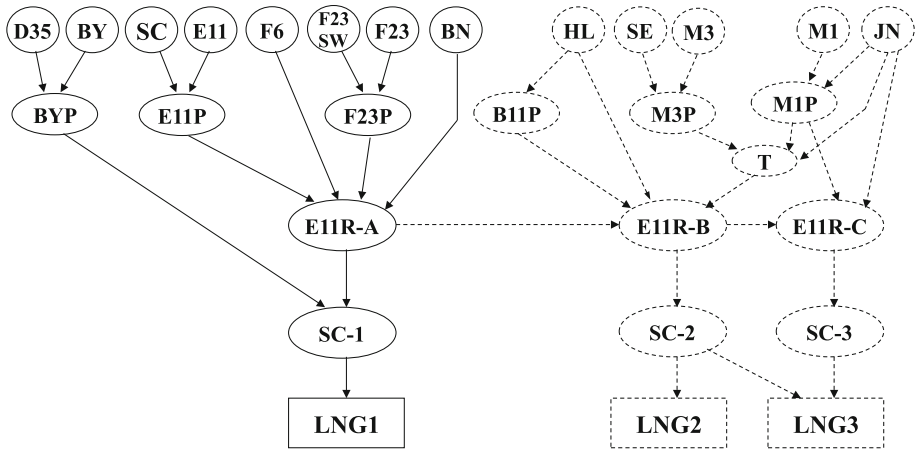


Fig. 4 Superstructure of case study B problem

The monolith approach solved the problems using three global optimization solvers, i.e., ANTIGONE 1.1 [25], BARON 16.8 [2], SCIP 3.2 [52]. ANTIGONE 1.1 and BARON 16.8 adopted CONOPT 3 [53] as its NLP solver and CPLEX 12.6 [54] as its LP/MILP solver, and SCIP 3.2 used its default solvers for the subproblems. GBD, JD1 and JD2 solved the problems by using CPLEX 12.6 for the LP/MILP subproblems and ANTIGONE 1.1 for the nonconvex NLP/MINLP subproblems.

The construction of Problems (ODR_i^k) and $(JRMPr^{(l)})$ in JD2 requires the convex relaxation of nonconvex sets X_0 and Y_ω . In the case studies, X_0 was a mixed integer set defined by linear constraints, and it was relaxed into a polyhedral set via continuous relaxation. Y_ω was a nonconvex continuous set defined with bilinear functions, and it was relaxed into a polyhedral set via standard McCormick relaxation [14]. The relative and absolute termination tolerances for Case Study A were set to 10^{-3} , and for Case study B were set to 10^{-2} . GBD, JD1 and JD2 started with all design decisions being 0 (which is a trivial feasible solution for both case study problems).

During the execution of JD1 and JD2, large computing overhead may be incurred due to frequent model generation in GAMS and data exchange between GAMS and MATLAB. So both “Total solver time” and “Total run time” were recorded for the simulation studies, which refer to the total time for the subproblem solvers to solve each individual subproblem and the wall time for the entire solution procedure, respectively. The computing overhead could have been significantly reduced if JD1 and JD2 had been implemented using general-purpose programming languages, such as C++. For the monolith approach, the computing overhead was much less, as seen from the results in the next subsection.

5.3 Results and discussion

Summary of the results for case study A is presented on Tables 3, 4, 5 and 6. Table 3 shows the results for the monolith approach using the three global optimization solvers. It can be seen that ANTIGONE was the fastest among the three solvers, but its solution time increased quickly with the problem size. BARON could also solve small problem instances quickly, but it could not find the desired 10^{-3} -optimal solution (i.e., a solution with a relative gap no larger than 0.1%) for larger problem instances within the 1 h run time limit. SCIP was

Table 3 Results for case study A—Monolith (unit for time: s)

Number of scenarios	25	36	49	64	81	100
ANTIGONE 1.1						
Objective val. (\$)	− 532.1	− 530.6	− 531.2	− 531.5	− 531.1	− 531.1
Relative gap	≤ 0.1%	≤ 0.1%	≤ 0.1%	≤ 0.1%	≤ 0.1%	≤ 0.1%
Total solver time	12	30	95	242	548	1470
Total run time	13	35	112	284	645	1703
BARON 16.8						
Objective val. (\$)	− 532.1	− 530.6	− 233.17	− 397.7	− 163.2	− 427.8
Relative gap	≤ 0.1%	≤ 0.1%	63.6%	25.5%	69.6%	22.2%
Total solver time	18	30	_a	_a	_a	_a
Total run time	20	37	_a	_a	_a	_a
SCIP 3.2						
Objective val. (\$)	− 532.1	− 530.6	− 531.2	− 531.5	− 531.1	− 531.1
Relative gap	≤ 0.1%	≤ 0.1%	0.58%	2%	3.7%	0.13%
Total solver time	134	1226	_b	_b	_b	_b
Total run time	163	1470	_b	_b	_b	_b

^aSolver terminated after the 1 h time limit, without finding the optimal solution

^bSolver obtained the optimal solution after the 1 h time limit, but did not reduce the gap to the set tolerance (10^{-3})

the slowest of the three solvers; but unlike BARON, it happened to find the 10^{-3} -optimal solution within 1 h for all problem instances (but could not verify the optimality for large problem instances). Table 4 shows that GBD could not find a nontrivial feasible solution for any problem instance within the 1 h time limit (and the zero objective value is from the initial trivial solution). On the other hand, Tables 4 and 5 show that both JD1 and JD2 could solve all problem instances fairly quickly. JD1 was not as fast as ANTIGONE or BARON for small problem instances, but its solution time increased more slowly than that of ANTIGONE or BARON. This was primarily because the number of JD1 iterations did not vary much with the number of scenarios. The nonconvex relaxed master problem ($JRMP^{(l)}$) was the major contributor to JD1 solution time, and sometimes it dominated the solution time (as in the 64 scenario case). In JD2 where the relaxation of ($JRMP^{(l)}$) (i.e., ($JRMPR^{(l)}$)) is solved, the number of ($JRMP^{(l)}$) needed to be solved was significantly reduced, and each ($JRMP^{(l)}$) was much easier to solve due to extensive domain reduction. The price for reducing the ($JRMP^{(l)}$) solution time was the time spent on optimization based domain reduction ODR^k , but the resulting total solution time still decreased for most cases, so JD2 generally outperformed JD1 and it scaled with the number of scenarios in a more consistent way. Note that Tables 4 and 5 do not include the times to solve easy LP and MILP subproblems like Problem ($BPP_{\omega}^{(l)}$), ($BFP_{\omega}^{(l)}$), (LS_0^k) and ($JRMPR^{(l)}$), because those times were very small compared to the total solution time.

Tables 7, 8, 9 and 10 present the results for case study B. ANTIGONE actually found the desired 10^{-2} -optimal solution, but it cannot reduce the gap to 1% within the 24 h run time limit; for the 25-scenario instance, it mistakenly terminated before the run time limit without reducing the gap to 1%. BARON had the similar problem; it obtained the 10^{-2} -optimal solution for most problem instances but could not reduce the gap to 1% for any problem instance. SCIP performed better than ANTIGONE and BARON for case study B, but it

Table 4 Results for case study A—GBD (unit for time: s)

Number of scenarios	25	36	49	64	81	100
Optimal obj. (\$)	0	0	0	0	0	0
Num. of iterations	5	8	5	5	5	5
Total solver time	> 3600	> 3600	> 3600	> 3600	> 3600	> 3600

For all cases, the solver terminated after 1 h time limit without finding a nontrivial feasible solution. The zero objective value is from the initial point where all variables are zero

Table 5 Results for case study A—JD1 (unit for time: s)

Number of scenarios	25	36	49	64	81	100
Optimal obj. (\$)	− 532.1	− 530.6	− 531.2	− 531.5	− 531.1	− 531.1
Relative gap	≤ 0.1%	≤ 0.1%	≤ 0.1%	≤ 0.1%	≤ 0.1%	≤ 0.1%
Num. of iterations	8	13	10	14	10	12
Num. of JRMP ^(l) solved	4	5	5	7	5	6
Time for JRMP ^(l)	6	8	11	519	122	202
Time for LS _ω ^k	49	128	108	188	179	262
Time for PP ^k	7	25	18	124	45	66
Total solver time	63	168	141	840	352	540
Total run time	139	479	318	1223	677	1020

Table 6 Results for case study A—JD2 (unit for time: s)

Number of scenarios	25	36	49	64	81	100
Objective val. (\$)	− 532.1	− 530.5	− 531.2	− 531.5	− 530.7	− 530.7
Relative gap	≤ 0.1%	≤ 0.1%	≤ 0.1%	≤ 0.1%	≤ 0.1%	≤ 0.1%
Num. of iterations	10	10	10	8	10	10
Num. of JRMPR ^(l) solved	7	5	6	4	4	5
Num. of JRMP ^(l) solved	3	1	3	1	1	2
Time for JRMP ^(l)	1	2	2	2	3	5
Time for LS _ω ^k	51	71	103	110	165	190
Time for PP ^k	10	22	24	47	66	37
Time for ODR ^k	35	44	55	61	104	140
Total solver time	100	142	192	283	345	391
Total run time	210	308	406	549	739	968

could only solve the 25 scenario and 36 scenario problem instances successfully. Table 8 shows that GBD could not return a nontrivial feasible solution within the 24 h time limit. Table 9 shows that JD1 achieved an optimal solution for the 36 and 64 scenario cases, but it could not close the optimality gap within the 24 h time limit for the 25 and 49 scenario cases, and it suffered from insufficient memory for the 81 and 100 scenario cases. Either the insufficient solution time limit or the insufficient memory problem results from the fact that subproblem (JRMP^(l)) is too difficult if its domain is not sufficiently reduced according to the

Table 7 Results for case study B—Monolith (unit for time: s)

Number of scenarios	25	36	49	64	81	100
ANTIGONE 1.1						
Objective val. (Billion \$)	−33.87	−33.67	−33.81	−33.76	−33.78	−33.79
Relative gap.	1.4%	2.1%	1.7%	1.8%	1.8%	1.7%
Total solver time	51,465 ^a	_b	_b	_b	_b	_b
Total run time	58,522 ^a	_b	_b	_b	_b	_b
BARON 16.8						
Objective val. (Billion \$)	−33.87	−33.91	−33.90	−33.31	−33.91	−33.79
Relative gap.	1.4%	1.3%	1.3%	3.6%	1.3%	1.6%
Total solver time	40,530 ^a	59,965 ^a	58,460 ^a	_b	_b	_b
Total run time	68,060 ^a	69,520 ^a	70,196 ^a	_b	_b	_b
SCIP 3.2						
Objective val. (Billion \$)	−33.92	−33.91	−33.81	−33.76	−33.78	−33.77
Relative gap.	≤1%	≤1%	1.52%	1.69%	1.69%	1.75%
Total solver time	54,337	11,952	_b	_b	_b	_b
Total run time	61,365	13,316	_b	_b	_b	_b

^aSolver terminated with a nonzero exit code within 24 h, and the relative gap was larger than the set tolerance (10^{-2})

^bSolver terminated after the 24 h time limit, with a relative gap larger than the set tolerance (10^{-2})

Table 8 Results for case study B—GBD (unit for time: s)

Number of scenarios	25	36	49	64	81	100
Optimal obj. (\$)	0	0	0	0	0	0
Num. of iterations	205	145	117	101	111	121
Total solver time	> 86,400	> 86,400	> 86,400	> 86,400	> 86,400	> 86,400

For all cases, the solver terminated after the 24 h time limit without finding a nontrivial feasible solution. The zero objective value is from the initial point where all variables are zero

information from previous solved subproblems. Table 7 shows that JD2 solved all problem instances successfully, and its solution time scaled well with the number of scenarios. This is because the total number of JD2 iterations did not vary significantly with the number of scenarios, and the times for JRMP^(l) and domain reduction did not increase greatly with the number of scenarios. It can be seen that for this problem, domain reduction, primarily (ODR_i^k), dominated the total solution time, so a more efficient way to perform domain reduction could have been able to effectively reduce the solution time. This case study problem indicates that, general-purpose global optimization solvers may not be able to effectively exploit the structure of a complex nonconvex MINLP and solve the problem efficiently enough, and this is when one might consider the use of a tailored decomposition strategy like the one proposed in this paper.

Table 9 Results for case study B—JD1 (unit for time: s)

Number of scenarios	25	36	49	64	81	100
Objective val. (Billion \$)	− 33.49 ^a	− 33.58	− 33.5 ^a	− 33.56	0 ^b	− 33.50 ^b
Relative gap	1.3%	≤ 1%	2.1%	≤ 1%	–	2.8%
Num. of iterations	15	25	14	21	6	17
Num. of JRMP ^(l) solved	9	17	8	12	4	8
Time for JRMP ^(l)	> 86,400	25,066	> 86,400	4452	797	1975
Time for LS _ω ^k	251	4418	323	4642	172	1137
Time for PP ^k	61	146	229	206	118	199
Total solver time	> 86,400	29,713	> 86,400	9440	1105	3401
Total run time	> 86,400	33,913	> 86,400	14,127	2051	6944

^aSolver terminated while solving JRMP because of the 24 h limit (86,400s). The relative gap was larger than the set tolerance (10^{-2})

^bSolver terminated because of insufficient memory

Table 10 Results for case study B—JD2 (unit for time: s)

Number of scenarios	25	36	49	64	81	100
Objective val. (Billion \$)	− 33.58	− 33.57	− 33.77	− 33.71	− 33.57	− 33.55
Relative gap	≤ 1%	≤ 1%	≤ 1%	≤ 1%	≤ 1%	≤ 1%
Num. of iterations	27	24	30	25	23	23
Num. of JRMP ^(l) solved	21	17	23	17	16	14
Num. of JRMP ^(l) solved	17	10	15	7	8	6
Time for JRMP ^(l)	948	696	3547	1617	3948	5651
Time for LS _ω ^k	5676	3820	14,279	2734	2188	2814
Time for PP ^k	155	443	560	509	388	1000
Time for ODR ^k	7203	9247	19,020	22,661	21,137	30,961
Total solver time	14,028	14,288	37,832	27,702	27,893	40,769
Total run time	16,431	16,482	44,525	32,150	33,271	47,483

6 Concluding remarks

Two joint decomposition methods, JD1, and JD2, are developed in this paper for efficient global optimization of Problem (P). JD1 is a basic joint decomposition approach, which follows the notions of classical decomposition methods as well as convex relaxation, in order to solve (P) via solving a sequence of relatively easy subproblems. JD2 is an enhanced version of JD1 that integrates several domain reduction techniques. It has been proved that both methods can terminate in a finite number of iterations with an ϵ -optimal solution if some mild conditions are satisfied.

We considered two case study problems that come from integrated design and operation under uncertainty, in order to demonstrate the potential computational advantages of joint decomposition. For the first problem which is smaller and easier, both JD1 and JD2 outperformed state-of-the-art global solvers when the number of scenarios was large, and JD2 generally outperformed JD1. For the second problem which is larger and more difficult, JD2 outperformed state-of-the-art global solvers and JD1 (which could not close the gap for most

cases). The case study results indicate that, when joint decomposition can effectively exploit the problem structure, the total number of iterations it requires does not increase significantly with the number of scenarios, and consequently the solution time increases slowly with the problem size compared to the general-purpose global optimization solvers. On the other hand, like all decomposition methods, joint decomposition uses existing solvers to solve its subproblems, so its computational performance does rely on the advances in general-purpose local and global optimization solvers.

In this paper, we only consider domain reduction for the linking variables in x_0 . In the future, we will also consider domain reduction for some key non-linking complicating variables in y_ω that influence the convergence rate the most, and investigate how to find out these key variables. This can effectively tighten the convex relaxation of Problem (JRMP^(l)), and therefore reduce the number of JRMP^(l) to be solved and accelerate the solution of each JRMP^(l). In addition, in the current JD method subproblem (RPMP^k) is used for generating Lagrangian multipliers (as well as x_0), but this subproblem may be time-consuming when a large number of JD iterations is required. In the future, we would like to use a sub-gradient based method (such as a bundle method [55]) to generate Lagrangian multipliers in the JD framework, and assess the performance of this method in comparison to the (RPMP^k) approach.

Acknowledgements The authors are grateful to the discovery grant (RGPIN 418411-13) and the collaborative research and development grant (CRDPJ 485798-15) from Natural Sciences and Engineering Research Council of Canada (NSERC).

Appendix A: Reformulation from (P0) to (P)

From Problem (P0), We first separate the convex part and the nonconvex part of the problem. Specifically, let $v_\omega = (v_{c,\omega}, v_{nc,\omega})$, where $v_{c,\omega}$ includes variables that are only involved in convex functions and restricted by convex constraints/sets, and $v_{nc,\omega}$ includes the variables that are involved in a nonconvex function and/or restricted by a nonconvex constraint/set. In addition, we introduce duplicate variables $v_{0,1}, \dots, v_{0,s}$ for variable x_0 , to express the relation among all scenarios using NACs. We then rewrite Problem (P0) as:

$$\begin{aligned}
 & \min_{\substack{x_0, v_{0,1}, \dots, v_{0,s} \\ v_{c,1}, \dots, v_{c,s} \\ v_{nc,1}, \dots, v_{nc,s}}} \sum_{\omega=1}^s [f_{0,\omega}(v_{0,\omega}) + f_{c,\omega}(v_{c,\omega}) + f_{nc,\omega}(v_{nc,\omega})] \\
 & \text{s.t. } x_0 = v_{0,\omega}, \quad \forall \omega \in \{1, \dots, s\}, \\
 & \quad g_{0,\omega}(v_{0,\omega}) + g_{c,\omega}(v_{c,\omega}) + g_{nc,\omega}(v_{nc,\omega}) \leq 0, \quad \forall \omega \in \{1, \dots, s\}, \\
 & \quad x_0 \in X_0, \\
 & \quad v_{0,\omega} \in \hat{X}_0, \quad v_{c,\omega} \in V_{c,\omega}, \quad v_{nc,\omega} \in V_{nc,\omega}, \quad \forall \omega \in \{1, \dots, s\}.
 \end{aligned} \tag{A.1}$$

In the above formulation, set $X_0 \subset \mathbb{R}^{n_0}$ is either convex or nonconvex, set $V_{c,\omega} \subset \mathbb{R}^{n_c}$ is convex, set $V_{nc,\omega} \subset \mathbb{R}^{n_{nc}}$ is either convex or nonconvex. Functions $f_{c,\omega} : V_{c,\omega} \rightarrow \mathbb{R}$ and $g_{c,\omega} : V_{c,\omega} \rightarrow \mathbb{R}^{m_c}$ are convex. Functions $f_{nc,\omega} : V_{nc,\omega} \rightarrow \mathbb{R}$, $g_{nc,\omega} : V_{nc,\omega} \rightarrow \mathbb{R}^{m_{nc}}$, $f_{0,\omega}$, and $g_{0,\omega}$ are either convex or nonconvex. Set $\hat{X}_0 \in \mathbb{R}^{n_0}$ is a convex relaxation of X_0 (and it is same to X_0 if X_0 is convex). The restriction $z_{0,\omega} \in \hat{X}_0$ is actually redundant with the presence of NACs; however, it tightens the problem when the NACs are dualized. Note that in order to generate a convex relaxation of X_0 , extra variables may be introduced [56], so the

dimension of the relaxation may be larger than that of X_0 . Here \hat{X}_0 can be understood as the projection of the relaxation set on the \mathbb{R}^{n_0} space. For simplicity of notation, in this paper we always express a convex relaxation (of a set or a function) on the original variable space and do not explicitly show the extra variables needed for constructing the relaxation.

Define new variables $t_\omega, \alpha_{c,\omega}, \alpha_{nc,\omega}, \beta_{c,\omega}, \beta_{nc,\omega}$, such that Problem (A.1) can be written as:

$$\begin{aligned}
 & \min \sum_{\omega=1}^s t_\omega \\
 \text{s.t. } & x_0 = v_{0,\omega}, \quad \forall \omega \in \{1, \dots, s\}, \\
 & \beta_{c,\omega} + \beta_{nc,\omega} \leq 0, \quad \forall \omega \in \{1, \dots, s\}, \\
 & t_\omega \geq \alpha_{c,\omega} + \alpha_{nc,\omega}, \quad \forall \omega \in \{1, \dots, s\}, \\
 & \alpha_{c,\omega} \geq f_{c,\omega}(v_{c,\omega}), \quad \forall \omega \in \{1, \dots, s\}, \\
 & \alpha_{nc,\omega} \geq f_{0,\omega}(v_{0,\omega}) + f_{nc,\omega}(v_{nc,\omega}), \quad \forall \omega \in \{1, \dots, s\}, \\
 & \beta_{c,\omega} \geq g_{0,\omega}(v_{0,\omega}) + g_{c,\omega}(v_{c,\omega}), \quad \forall \omega \in \{1, \dots, s\}, \\
 & \beta_{nc,\omega} \geq g_{nc,\omega}(v_{nc,\omega}), \quad \forall \omega \in \{1, \dots, s\}, \\
 & x_0 \in X_0, \\
 & v_{0,\omega} \in \hat{X}_0, \quad v_{c,\omega} \in V_{c,\omega}, \quad v_{nc,\omega} \in V_{nc,\omega}, \quad \forall \omega \in \{1, \dots, s\}.
 \end{aligned} \tag{A.2}$$

Define $x_\omega = (v_{0,\omega}, v_{c,\omega}, t_\omega, \alpha_{c,\omega}, \beta_{c,\omega})$, $y_\omega = (v_{nc,\omega}, \alpha_{nc,\omega}, \beta_{nc,\omega})$, then the above formulation can be written as the following Problem (P):

$$\begin{aligned}
 & \min \sum_{\omega=1}^s c_\omega^T x_\omega \\
 \text{s.t. } & x_0 = H_\omega x_\omega, \quad \forall \omega \in \{1, \dots, s\}, \\
 & A_\omega x_\omega + B_\omega y_\omega \leq 0, \quad \forall \omega \in \{1, \dots, s\}, \\
 & x_0 \in X_0, \\
 & x_\omega \in X_\omega, \quad y_\omega \in Y_\omega, \quad \forall \omega \in \{1, \dots, s\},
 \end{aligned} \tag{P}$$

where the matrices

$$c_\omega = \begin{bmatrix} 0 \\ 0 \\ I \\ 0 \\ 0 \end{bmatrix}, \quad H_\omega = [I \ 0 \ 0 \ 0 \ 0], \quad A_\omega = \begin{bmatrix} 0 & 0 & 0 & 0 & I \\ 0 & 0 & -I & I & 0 \end{bmatrix}, \quad B_\omega = \begin{bmatrix} 0 & 0 & I \\ 0 & I & 0 \end{bmatrix},$$

and the sets

$$\begin{aligned}
 X_\omega &= \{(v_{0,\omega}, v_{c,\omega}, t_\omega, \alpha_{c,\omega}, \beta_{c,\omega}) : v_{0,\omega} \in \hat{X}_0, \quad v_{c,\omega} \in V_{c,\omega}, \\
 & \quad \alpha_{c,\omega} \geq f_{c,\omega}(v_{c,\omega}), \quad \beta_{c,\omega} \geq g_{0,\omega}(v_{0,\omega}) + g_{c,\omega}(v_{c,\omega})\}, \\
 Y_\omega &= \{(v_{nc,\omega}, \alpha_{nc,\omega}, \beta_{nc,\omega}) : v_{nc,\omega} \in V_{nc,\omega}, \quad \alpha_{nc,\omega} \geq f_{0,\omega}(v_{0,\omega}) + f_{nc,\omega}(v_{nc,\omega}), \\
 & \quad \beta_{nc,\omega} \geq g_{nc,\omega}(v_{nc,\omega})\}.
 \end{aligned}$$

The “0” and “I” in the matrices represent zero and identity matrices, and their dimensions are conformable to the relevant variables. According to the convexity/nonconvexity of the functions and the sets stated before, set X_ω is convex and set Y_ω is nonconvex.

Appendix B: The stochastic pooling problem with mixed-integer first-stage decisions

The two-stage stochastic pooling problem from Li et al. [3] is modified here to address continuous design (first-stage) decisions. The nomenclature used in [3] is adopted to describe the model, in which the scenarios are indexed by h (rather than ω).

In the modified model, the design decisions on sources, pools, product terminals, denoted by y_i^S, y_j^P, y_k^T , can be continuous, integer, or mixed integer. If $y_i^S \in \{0, 1\}$, then the design decision is to determine whether source i is to be developed, and the related parameter Z_i^{UB} represents the fixed capacity of the source. If y_i^S is continuous and $y_i^S \in [0, 1]$, then it is a capacity design decision, specifically it represents the ratio of source i capacity to the maximum allowed capacity of the source (denoted by Z_i^{UB}). The design decisions on the pipelines among sources, pools, and terminals are all continuous, denoted by $y_{i,j}^{SP}, y_{i,k}^{ST}, y_{j,j^-}^{PP}, y_{j,k}^{PT} \in [0, 1]$. They represents the ratios of the pipeline capacities to the maximum allowed capacities (denoted by $F_{i,j}^{SP,UB}, F_{i,k}^{ST,UB}, F_{j,j^-}^{PP,UB}, F_{j,k}^{PT,UB}$).

All design and operational decision variables are nonnegative, and we do not impose other lower bounds on these variables in order to simplify discussion. The new stochastic pooling model consists primarily of three submodels, for the sources, pools, and product terminals, respectively.

Model for the sources

The following group of constraints (B.1) represents the submodel for the sources. Equations (B.1a–B.1c) are same to Eqs. (12–14) in [3], except that the lower flow bounds are not imposed. Equations (B.1d–B.1f) are developed in place of the topology constraints Eqs. (15–16) (which are invalid for continuous design decisions). Equations (B.1d–B.1e) limit the capacity of a pipeline by the capacity of the source it connects. If $y_i^S = 0$, then there cannot exist a pipeline connecting it, in other words, the capacity of a pipeline connecting it has to be zero. Equation (B.1f) requires that the total capacity of all pipelines connecting to a source should be no less than the capacity of the source. This is to ensure enough pipeline capacity to move all materials generated in the source to other parts of the system in real-time.

$$\sum_{j \in \Theta_i^{SP}} f_{i,j,h}^{SP} + \sum_{k \in \Theta_i^{ST}} f_{i,k,h}^{ST} \leq y_i^S Z_i^{UB}, \tag{B.1a}$$

$$f_{i,j,h}^{SP} \leq y_{i,j}^{SP} F_{i,j}^{SP,UB}, \tag{B.1b}$$

$$f_{i,k,h}^{ST} \leq y_{i,k}^{ST} F_{i,k}^{ST,UB}, \tag{B.1c}$$

$$y_{i,j}^{SP} F_{i,j}^{SP,UB} \leq y_i^S Z_i^{UB}, \tag{B.1d}$$

$$y_{i,k}^{ST} F_{i,k}^{ST,UB} \leq y_i^S Z_i^{UB}, \tag{B.1e}$$

$$y_i^S Z_i^{UB} \leq \sum_{j \in \Theta_i^{SP}} y_{i,j}^{SP} F_{i,j}^{SP,UB} + \sum_{k \in \Theta_i^{ST}} y_{i,k}^{ST} F_{i,k}^{ST,UB},$$

$$\forall i \in \{1, \dots, n\}, \forall j \in \Theta_i^{SP}, \forall k \in \Theta_i^{ST}, \forall h \in \{1, \dots, b\}. \tag{B.1f}$$

Model for the pools

The following group of constraints (B.2) represents the submodel for the pools. Equations (B.2a–B.1e) are same to Eqs. (17–21) in [3], except that the lower flow bounds are not imposed. Equations (B.2f–B.2k) are developed in place of the topology constraints (23–26) in [3]. The interpretation of Eqs. (B.2f–B.2k) is similar to that of Eqs. (B.1d–B.1f) and therefore omitted.

$$f_{j,k,w,h}^{PT} = s_{j,k,h}^{PT} \left(\sum_{i \in \Omega_j^{SP}} f_{i,j,h}^{SP} U_{i,w,h} + \sum_{j^+ \in \Omega_j^{PP+}} f_{j^+,j,w,h}^{PP} \right), \tag{B.2a}$$

$$f_{j,j^-,w,h}^{PP} = s_{j,j^-,h}^{PP} \left(\sum_{i \in \Omega_j^{SP}} f_{i,j,h}^{SP} U_{i,w,h} + \sum_{j^+ \in \Omega_j^{PP+}} f_{j^+,j,w,h}^{PP} \right), \tag{B.2b}$$

$$\sum_{j^- \in \Omega_j^{PP-}} s_{j,j^-,h}^{PP} + \sum_{k \in \Omega_j^{PT}} s_{j,k,h}^{PT} = 1, \quad s_{j,j^-,h}^{PP}, s_{j,k,h}^{PT} \geq 0, \tag{B.2c}$$

$$y_{j,j^-}^{PP} F_{j,j^-}^{PP,UB} \leq \sum_{w \in \{1, \dots, l\}} f_{j,j^-,w,h}^{PP} \leq y_{j,j^-}^{PP} F_{j,j^-}^{PP,UB}, \tag{B.2d}$$

$$y_{j,k}^{PT} F_{j,k}^{PT,UB} \leq \sum_{w \in \{1, \dots, l\}} f_{j,k,w,h}^{PT} \leq y_{j,k}^{PT} F_{j,k}^{PT,UB}, \tag{B.2e}$$

$$y_j^P Z_j^{P,UB} \geq y_{i,j}^{SP} F_{i,j}^{SP,UB}, \tag{B.2f}$$

$$y_j^P Z_j^{P,UB} \geq y_{j^+,j}^{PP} F_{j^+,j}^{PP,UB}, \tag{B.2g}$$

$$y_j^P Z_j^{P,UB} \geq y_{j,j^-}^{PP} F_{j,j^-}^{PP,UB}, \tag{B.2h}$$

$$y_j^P Z_j^{P,UB} \geq y_{j,k}^{PT} F_{j,k}^{PT,UB}, \tag{B.2i}$$

$$y_j^P Z_j^{P,UB} \leq \sum_{j^+ \in \Omega_j^{PP+}} y_{j^+,j}^{PP} F_{j^+,j}^{PP,UB} + \sum_{i \in \Omega_j^{SP}} y_{i,j}^{SP} F_{i,j}^{SP,UB}, \tag{B.2j}$$

$$y_j^P Z_j^{P,UB} \leq \sum_{j^- \in \Omega_j^{PP-}} y_{j,j^-}^{PP} F_{j,j^-}^{PP,UB} + \sum_{k \in \Omega_j^{PT}} y_{j,k}^{PT} F_{j,k}^{PT,UB},$$

$$\forall j \in \{1, \dots, r\}, \forall j^- \in \Omega_j^{PP-}, \forall k \in \Omega_j^{PT}, \forall w \in \{1, \dots, l\}, \forall h \in \{1, \dots, b\}. \tag{B.2k}$$

Model for the product terminals

The following group of constraints (B.3) represents the submodel for the terminals. Equations (B.3a–B.3b) are same to Eq. (27–28) in [3], except that the lower flow bounds and content bounds are not imposed. Again, Eqs. (B.3c–B.3e) are developed in place of the old topology constraints that are invalid for continuous design decisions (i.e., Eqs. (23–26) in [3]).

$$\sum_{j \in \Pi_k^{PT}} \sum_{w \in \{1, \dots, l\}} f_{j,k,w,h}^{PT} + \sum_{i \in \Pi_k^{ST}} f_{i,k,h}^{ST} \leq y_k^T D_{k,h}^{UB}, \tag{B.3a}$$

$$\sum_{j \in \Pi_k^{PT}} f_{j,k,w,h}^{PT} + \sum_{i \in \Pi_k^{ST}} f_{i,k,h}^{ST} U_{i,w,h} \leq \left(\sum_{j \in \Pi_k^{PT}} \sum_{w \in \{1, \dots, l\}} f_{j,k,w,h}^{PT} + \sum_{i \in \Pi_k^{ST}} f_{i,k,h}^{ST} \right) V_{k,w}^{UB} \tag{B.3b}$$

$$y_k^T D_k^{UB} \geq y_{i,k}^{ST} F_{i,k}^{ST,UB} \tag{B.3c}$$

$$y_k^T D_k^{UB} \geq y_{j,k}^{PT} F_{j,k}^{PT,UB}, \tag{B.3d}$$

$$y_k^T D_k^{UB} \leq \sum_{i \in \Pi_k^{ST}} y_{i,k}^{ST} F_{i,k}^{ST,UB} + \sum_{k \in \Pi_k^{PT}} y_{j,k}^{PT} F_{j,k}^{PT,UB} \tag{B.3e}$$

$$\forall k \in \{1, \dots, m\}, \forall w \in \{1, \dots, l\}, \forall h \in \{1, \dots, b\}.$$

The modified stochastic pooling model can be stated as:

minimize objective

s.t. Eq. (B.1a-B.1f), Eq. (B.2a-B.2k), Eq. (B.3a-B.3e),

$$y_i^S, y_j^P, y_k^T \in \{0, 1\} \text{ or } [0, 1],$$

$$y_{i,j}^{SP}, y_{i,k}^{ST}, y_{j,j-}^{PP}, y_{j,k}^{PT} \in [0, 1],$$

all flow rates are nonnegative,

redundant constraints for accelerating global optimization (Eqs. (38–39) in [3]).

The objective can be negative net present value, or negative annualized profit, as specified in [3].

References

1. Floudas, C.A.: *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. Oxford University Press, Oxford (1995)
2. Tawarmalani, M., Sahinidis, N.V.: Global optimization of mixed-integer nonlinear programs: a theoretical and computational study. *Math. Program.* **99**, 563–591 (2004)
3. Li, X., Armagan, E., Tomasgard, A., Barton, P.I.: Stochastic pooling problem for natural gas production network design and operation under uncertainty. *AIChE J.* **57**, 2120–2135 (2011)
4. Li, X., Tomasgard, A., Barton, P.I.: Nonconvex generalized Benders decomposition for stochastic separable mixed-integer nonlinear programs. *J. Optim. Theory Appl.* **151**, 425–454 (2011)
5. Sahinidis, N., Grossmann, I.: Convergence properties of generalized Benders decomposition. *Comput. Chem. Eng.* **15**(7), 481–491 (1991)
6. Berman, O., Ashrafi, N.: Optimization models for reliability of modular software systems. *IEEE Trans. Softw. Eng.* **19**, 1119–1123 (1993)
7. Floudas, C.A., Pardalos, P.M., Adjiman, C.S., Esposito, W.R., Gumus, Z., Harding, S.T., Klepeis, J.L., Meyer, C.A., Schweiger, C.A.: *Handbook of Test Problems for Local and Global Optimization*. Kluwer Academic Publishers, Dordrecht (1999)
8. Adjiman, C.S., Androulakis, I.P., Floudas, C.A.: Global optimization of mixed-integer nonlinear problems. *AIChE J.* **46**(9), 1769–1797 (2000)
9. Westerlund, T., Pettersson, F., Grossmann, I.E.: Optimization of pump configurations as a MINLP problem. *Comput. Chem. Eng.* **18**(9), 845–858 (1994)
10. Duran, M., Grossmann, I.: A mixed-integer nonlinear programming algorithm for process systems synthesis. *AIChE J.* **32**, 592–606 (1986)
11. Bloom, J.: Solving an electricity generating capacity expansion problem by generalized benders’ decomposition. *Oper. Res.* **31**, 84–100 (1983)

12. Falk, J., Soland, R.: An algorithm for seperable nonconvex programming problems. *Manag. Sci.* **15**, 550–569 (1969)
13. Soland, R.: An algorithm for seperable nonconvex programming problems II: nonconvex constraints. *Manag. Sci.* **17**, 759–772 (1971)
14. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: part I—convex underestimating problems. *Math. Program.* **10**, 147–175 (1976)
15. Fisher, M.: Lagrangian relaxation methods for solving integer programming problems. *Manag. Sci.* **27**, 1–18 (1981)
16. Barnhart, C., Johnson, E.: Branch and price: column generation for solving huge integer programs. *Oper. Res.* **46**, 316–329 (1998)
17. Guignard, M., Kim, S.: Lagrangean decomposition: a model yielding stronger lagrangean bounds. *Math. Program.* **39**, 215–228 (1987)
18. Held, M., Wolfe, P., Crowder, H.: Validation of subgradient optimization. *Math. Program.* **6**, 62–88 (1974)
19. Van Roy, T.J.: Cross decomposition for mixed integer programming. *Math. Program.* **25**(1), 46–63 (1983)
20. Ogbe, E., Li, X.: A new cross decomposition method for stochastic mixed-integer linear programming. *Eur. J. Oper. Res.* **256**, 487–499 (2017)
21. Zamora, J., Grossmann, I.E.: A branch and contract algorithm for problems with concave univariate, binear and linear fractional terms. *J. Glob. Optim.* **14**, 217–249 (1999)
22. Ryoo, H.S., Sahinidis, N.V.: A branch-and-reduce approach to global optimization. *J. Glob. Optim.* **8**, 107–138 (1996)
23. Misener, R., Floudas, C.A.: A framework for globally optimizing mixed-integer signomial programs. *J. Optim. Theory Appl.* **161**, 905–932 (2014)
24. Karuppiah, R., Grossmann, I.E.: A Lagrangean based branch-and-cut algorithm for global optimization of nonconvex mixed-integer nonlinear programs with decomposable structures. *J. Glob. Optim.* **41**, 163–186 (2008)
25. Misener, R., Floudas, C.A.: ANTIGONE: algorithms for continuous/integer global optimization of non-linear equations. *J. Glob. Optim.* **59**, 503–526 (2014)
26. Sahinidis, N.: Optimization under uncertainty: state-of-the-art and opportunities. *Comput. Chem. Eng.* **28**, 971–983 (2004)
27. Birge, J., Louveaux, F.: *Introduction to Stochastic Programming*. Springer, New York (2010)
28. Van Slyke, R.M., Wets, R.: L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM J. Appl. Math.* **17**, 638–663 (1969)
29. Chen, Y., Adams, T.A., Barton, P.I.: Optimal design and operation of static energy polygeneration systems. *Ind. Eng. Chem. Res.* **50**(9), 5099–5113 (2011)
30. Frank, S.M., Rebennack, S.: Optimal design of mixed AC–DC distribution systems for commercial buildings: a nonconvex generalized benders decomposition approach. *Eur. J. Oper. Res.* **242**(3), 710–729 (2015)
31. Benders, J.: Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.* **4**, 238–252 (1962)
32. Geoffrion, A.M.: Generalized Benders decomposition. *J. Optim. Theory Appl.* **10**(4), 237–260 (1972)
33. Karuppiah, R., Grossmann, I.: A Lagrangean based branch-and-cut algorithm for global optimization of nonconvex mixed-integer nonlinear programs with decomposable structures. *J. Glob. Optim.* **41**, 163–186 (2008)
34. Shim, Y., Fodstad, M., Gabriel, S.A., Tomsgard, A.: A branch-and-bound method for discretely-constrained mathematical programs with equilibrium constraints. *Ann. Oper. Res.* **210**(1), 5–31 (2013)
35. Kannan, R.: Algorithms, analysis and software for the global optimization of two-stage stochastic programs. Ph.D. thesis, Massachusetts Institute of Technology (2018)
36. Van Roy, T.J.: A cross decomposition algorithm for capacitated facility location. *Oper. Res.* **34**, 145–163 (1986)
37. Holmberg, K.: On the convergence of cross decomposition. *Math. Program.* **47**, 269–296 (1990)
38. Holmberg, K.: Mean value cross decomposition applied to integer programming problems. *Eur. J. Oper. Res.* **97**, 124–138 (1997)
39. Deep, N., Shahidehpour, S.: Cross decomposition for multi-area optimal reactive power planning. *IEEE Trans. Power Syst.* **8**, 1539–1544 (1993)
40. Mitra, S., Garcia-Herreros, P., Grossmann, I.E.: A cross-decomposition scheme with integrated primal-dual multi-cuts for two-stage stochastic programming investment planning problems. *Math. Program.* **157**(1), 95–119 (2016)
41. Carøe, C.C., Schultz, R.: Dual decomposition in stochastic integer programming. *Oper. Res. Lett.* **24**(1–2), 37–45 (1999)
42. Lasdon, L.: *Optimization Theory for Large Systems*, 1st edn. Macmillian, Toronto (1970)

43. Birge, J., Louveaux, F.V.: A multicut algorithm for two-stage stochastic linear programs. *Eur. J. Oper. Res.* **34**(3), 384–392 (1988)
44. Ogbe, E., Li, X.: Multicolumn-multicut cross decomposition for stochastic mixed-integer linear programming. *Comput. Aided Chem. Eng.* **37**, 737–742 (2015)
45. Maranas, C., Floudas, C.A.: Global optimization in generalized geometric programming. *Comput. Chem. Eng.* **21**(4), 351–369 (1997)
46. Haverly, C.: Studies of the behaviour of recursion for the pooling problem. *ACM SIGMAP Bull.* **25**, 29–32 (1978)
47. Haverly, C.: Behaviour of recursion model—more studies. *ACM SIGMAP Bull.* **26**, 22–28 (1979)
48. Selot, A., Kuok, L.K., Robinson, M., Mason, T.L., Barton, P.I.: A short-term operational planning model for natural gas production systems. *AIChE J.* **54**(2), 495–515 (2008)
49. Brook, A., Kendrick, D., Meeraus, A.: Gams, a user's guide. *SIGNUM Newsl.* **23**(3–4), 10–11 (1988)
50. MATLAB, version 8.3 (R2014a) The MathWorks Inc., Natick, Massachusetts (2014)
51. Ferris, M., Dirkse, S., Ramakrishnan, J.: GDXMRW: Interfacing GAMS and MATLAB. <http://research.cs.wisc.edu/math-prog/matlab.html> (2016). Accessed August
52. Achterberg, T.: SCIP: solving constraint integer programs. *Math. Program. Comput.* **1**(1), 1–41 (2009)
53. Drud, A.S.: CONOPT—a large-scale GRG code. *ORSA J. Comput.* **6**, 207–216 (1994)
54. IBM: IBM ILOG CPLEX OPTIMIZER: high-performance mathematical programming engine (2014)
55. Kiwiel, K.C.: A proximal-projection bundle method for Lagrangian relaxation, including semidefinite programming. *SIAM J. Optim.* **17**(4), 1015–1034 (2006)
56. Gatzke, E.P., Tolsma, J.E., Barton, P.I.: Construction of convex relaxations using automated code generation technique. *Optim. Eng.* **3**, 305–326 (2002)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.