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

Scalable Branching on Dual Decomposition of Stochastic Mixed-Integer Programming Problems

Kibaek Kim · Brian Dandurand

Received: January 23, 2020 / Accepted: date

Abstract We present a scalable branching method for the dual decomposition of stochastic mixed-integer programming. Our new branching method is based on the branching method proposed by Carøe and Schultz that creates branching disjunctions on first-stage variables only. We propose improvements to the process for creating branching disjunctions, including (1) branching on the optimal solutions of the Dantzig-Wolfe reformulation of the restricted master problem and (2) using a more comprehensive (yet simple) measure for the dispersions associated with subproblem solution infeasibility. We prove that the proposed branching process leads to an algorithm that terminates finitely, and we provide conditions under which globally optimal solutions can be identified after termination. We have implemented our new branching method, as well as the Carøe-Schultz method and a branch-and-price method, in the open-source software package DSP. Using SIPLIB test instances, we present extensive numerical results to demonstrate that the proposed branching method significantly reduces the number of node subproblems and solution times.

Keywords stochastic mixed-integer programming · branch-and-bound · dual decomposition · parallel computing

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357. We gratefully acknowledge the computing resources provided on Bebop, a high-performance computing cluster operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

Kibaek Kim
Mathematics and Computer Science Division
Argonne National Laboratory 9700 South Cass Avenue, Lemont, IL 60439, USA
E-mail: kimk@anl.gov

Brian Dandurand
Mathematics and Computer Science Division
Argonne National Laboratory 9700 South Cass Avenue, Lemont, IL 60439, USA
E-mail: bdandurand@anl.gov

Mathematics Subject Classification (2000) 90-08, 90C06, 90C11, 90C15, 90C25, 90C26, 90C30, 90C46

1 Introduction

The Lagrangian dual decomposition (DD) has been successfully applied to large-scale stochastic mixed-integer programs (SMIPs) for finding solutions with tight bounds (e.g., [32,33,40,1,25,7,8,17]). In addition, the DD methods scale with the number of scenarios, capable of running on high-performance computing clusters. Various algorithmic approaches and extensive numerical results have been reported in recent literature [17,16,13]. In this paper, we consider the DD of the split variable deterministic reformulation of SMIPs (see, e.g., [6]) of the form

$$
\begin{aligned}
z = \min_{x_j, y_j} \quad & \sum_{j=1}^{N} p_j \left(c^T x_j + d_j^T y_j \right) && \text{(1a)}\\
\text{s.t.} \quad & \sum_{j=1}^{N} H_j x_j = 0, && \text{(1b)}\\
& (x_j, y_j) \in G_j, \quad \forall j \in \mathcal{J}, && \text{(1c)}
\end{aligned}
$$

where $x_j \in \mathbb{R}^{n_1}$ and $y_j \in \mathbb{R}^{n_2}$ are first- and second-stage decision variables, respectively, associated with scenario $j \in \mathcal{J} := \{1, \ldots, N\}$ with the corresponding probabilities p_j. Constraint (1b) is called the *nonanticipativity* constraint that represents $x_1 = x_2 = \cdots = x_N$ with the matrices $H_j \in \mathbb{R}^{N \cdot n_1 \times n_1}$ for $j \in \mathcal{J}$. The constraint sets $G := (G_j)_{j \in \mathcal{J}}$ are understood to be generic sets defined by linear constraints and mixed-integer restrictions. Their realizations will be node dependent within a branch-and-cut context. Of particular interest is the solution to the root node instance of problem (1), where the sets G_j, $j \in \mathcal{J}$, take realizations $G_j = G_j^{\mathcal{N}^0}$, where

$$
G_j^{\mathcal{N}^0} := \{(x, y) : Ax \geqslant b,\ T_j x + W_j y \geqslant h_j,\ x \in X,\ y \in Y\}
$$

where the first-stage constraint matrix and the right-hand side are given by $A \in \mathbb{R}^{m_1 \times n_1}$ and $b \in \mathbb{R}^{m_1}$, respectively, and the second-stage matrices and right-hand side are given by $T_j \in \mathbb{R}^{m_2 \times n_1}, W_j \in \mathbb{R}^{m_2 \times n_2}$, and $h_j \in \mathbb{R}^{m_2}$, respectively. The sets $X \subseteq \mathbb{R}^{n_1 - p_1} \times \mathbb{Z}^{p_1}$ and $Y \subseteq \mathbb{R}^{n_2 - p_2} \times \mathbb{Z}^{p_2}$ represent integer restrictions on some of the decision variables x and y.

The DD of (1) can be obtained by taking the Lagrangian relaxation of the nonanticipativity constraint (1b). Let $\lambda \in \mathbb{R}^{N \cdot n_1}$ be the dual variables corresponding to the nonanticipativity constraint. Relaxing the constraint, the Lagrangian dual function is given by

$$
D(\lambda) := \min_{x_j, y_j} \left\{ \sum_{j=1}^{N} p_j \left(c^T x_j + d_j^T y_j \right) - \lambda^T \sum_{j=1}^{N} H_j x_j,\ (x_j, y_j) \in G_j\ \forall j \in \mathcal{J} \right\}. \tag{2}
$$

The Lagrangian dual function provides a lower bound of the optimal objective value (i.e., $z \geqslant D(\lambda)$) for any $\lambda \in \mathbb{R}^{N \cdot n_1}$. Moreover, the Lagrangian dual function can be decomposed for each scenario $j \in \mathcal{J}$. Consequently, the Lagrangian dual bound can be obtained by solving

$$z_{LD} = \max_{\lambda \in \mathbb{R}^{N \cdot n_1}} \sum_{j=1}^{N} D_j(\lambda), \tag{3}$$

where for $j \in \mathcal{J}$,

$$D_j(\lambda) := \min_{(x_j, y_j) \in G_j} \left\{ p_j \left(c^T x_j + d_j^T y_j \right) - \lambda^T H_j x_j \right\}. \tag{4}$$

Note that $D(\lambda)$ is a piecewise concave function in λ, as are $D_j(\lambda)$ for $j \in \mathcal{J}$. Hence, we can solve the Lagrangian dual problem (3) by using (nonsmooth) convex optimization algorithms.

Branch-and-bound methods have been applied in various algorithmic frameworks for finding optimal solutions to SMIP [26,2,9,41,30]. The method proposed in [26] is based on the view of Dantzig-Wolfe decomposition of SMIP, where a branch-and-price method has been applied. Branch-and-price methods are available in existing open-source software packages (e.g., GCG [12] and DIP [38]). PIPS-SBB, developed in [30], solves the deterministic equivalent problem in distributed-memory architectures. In a recent work [4], the progressive hedging algorithm was used in the branch-and-bound framework for global optimality. The idea of this work [4] is similar to an earlier work by Carøe and Schultz [9], where the dual decomposition is used instead of progressive hedging. However, these methods [38,12,26,30,4] require the generation of branching constraints on integer variables in both first and second stages. Consequently, the size of the search tree is likely to increase with the number of scenarios. Of note is the pioneer work [9] in DD that presents the branching method that creates disjunctions on (both continuous and integer) first-stage variables only. This method is scalable because the search tree size is independent of the number of scenarios. The approach of [41] also applies branching only on first-stage variables within a generalized Benders' Decomposition framework. Other types of algorithms have also been proposed for certain classes of SMIP [2,1,40].

In this work, we present a new branching method for the DD framework. Our method is based most directly on the existing method in [9], and it is modified most significantly from [9] *to improve the choice of the branching variable* (i.e., the branching point and branching index). We prove that with our method of choosing the branching variable and its corresponding criteria for fathoming nodes, the modified algorithm preserves the finite termination property of the DD framework of [9]. Our approach provides two conditions under which a node may be fathomed by optimality. One is analogous to that given in [9]; the other, which follows from the innovation in the branching, is new and distinct, allowing for more opportunities to fathom nodes by optimality. For our numerical experiments, we implemented our proposed branching method as well as the branch-and-price method [26] and the Carøe-and-Schultz method [9]) in the open-source parallel software package DSP [17]. We tested SIPLIB instances on a 664-node computing cluster at Argonne National Laboratory. We demonstrate that our branching method outperforms the existing methods for all of the instances tested.

We remark that the dual decomposition method and DSP are applicable to all problems that have dual block angular structures, of which SMIPs are a special case. In fact, an early version of the branch-and-bound implementation in DSP was developed and used for a long-term planning problem [14], where the branching method was based on the branch-and-price method.

The contributions of this work are summarized as follows.

- We develop an improved branching method combining the following qualities:
  1. The size of the search tree is independent of the number of scenarios (as with the method in [9]).
  2. The usage of information present in the solution process of the Dantzig-Wolfe decomposition of the SMIP is improved (as with the method in [26]) for informing the choice of branching variable.
- We prove that the finite termination property of the modified algorithm is preserved with the use of the new branching method and its corresponding criteria for fathoming. Furthermore, the improved usage of information from the Dantzig-Wolfe decomposition allows for an extra condition under which a node may be fathomed by optimality.
- We implement the new branching method, as well as two existing methods, in the software package DSP.
- We report global optimal solutions for SIPLIB instances that are possible with the new branching method.

The paper is organized as follows. In Section 2 we present a proximal bundle method for the Lagrangian dual problem. Note that other variants of the bundle method could be used in this work. In Section 3 we start by discussing two existing branching methods: the Carøe-and-Schultz method [9] (Section 3.1.1) and a branch-and-price method base on [26] (Section 3.1.2). We then present our new branching method (Section 3.2) that combines the main advantages of the previous two methods [9,26], and we show that the finite termination property of the resulting modified algorithm is preserved. In Section 5 we compare the numerical results of benchmarking the new branching method and the existing methods. In Section 6 we summarize our work and discuss directions for future extensions.

2 Proximal Bundle Method for the Lagrangian Dual Problem

We present a proximal bundle method adapted for solving the Lagrangian dual problem (3). The method is based on the cutting-plane method that iteratively approximates the piecewise linear functions $D_j(\lambda)$ by adding linear inequalities. Note that any convex nonsmooth optimization algorithm can potentially be used for solving the Lagrangian dual function $D(\lambda)$. We use the proximal bundle method because of the following reasons. First, the bundle information can be used for the branching method (as presented in Section 3.2). Second, the proximal term ensures that the Lagrangian dual problem is bounded, which is equivalent to the feasibility of the restricted master problem in the form of Dantzig-Wolfe decomposition (see Section 3.1.2). Third, the proximal bundle method regularizes the dual search space and may result in a fewer number of inequalities being generated. The proximal bundle method has been well studied in the literature (e.g., [19,20,11,36,35] and dually its proximal stabilized column generation analogs [3,21,8]). For completeness of the development of the algorithms, we present the application of the proximal bundle method to the dual decomposition in this section.

Given previously computed *trial points* λ^l and *stability centers* $\bar{\lambda}^l$, $l = 0, \ldots, k-1$, we describe the computation of the next trial point λ^k and stability center $\bar{\lambda}^k$. The model function representing the piecewise linear approximation of $D(\lambda)$ at iteration k is given by

$$m_k(\lambda) := \max_{\theta_j} \quad \sum_{j=1}^{N} \theta_j \tag{5a}$$

$$\text{s.t.} \quad \theta_j \leqslant D_j(\lambda^l) - (H_j x_j^l)^T(\lambda - \lambda^l) \quad \forall j \in \mathcal{J}^l,\ l = 0, 1, \ldots, k-1, \tag{5b}$$

where $-H_j x_j^l \in \partial D_j(\lambda^l)$. We define $\mathcal{J}^k \subseteq \mathcal{J}$, since the cutting planes (5b) may not be added at every iteration. Therefore, at iteration k, the proximal bundle master problem is given by

$$\max_{\lambda} \ m_k(\lambda) - \frac{u^k}{2} \left\| \lambda - \bar{\lambda}^{k-1} \right\|_2^2, \tag{6}$$

where u^k is a penalty parameter and $\bar{\lambda}^{k-1}$ is a stability center. The necessary and sufficient optimality condition in order for λ^k to be an optimal solution to (6) is given by

$$0 \in \partial m_k(\lambda^k) - u^k(\lambda^k - \bar{\lambda}^{k-1}). \tag{7}$$

Therefore, $u^k(\lambda^k - \bar{\lambda}^{k-1}) \in \partial m_k(\lambda^k)$, and for any $g^k \in \partial m_k(\lambda^k)$, we have

$$m_k(\lambda^k) + (g^k)^T(\lambda - \lambda^k) \geqslant m_k(\lambda) \geqslant D(\lambda) \tag{8}$$

for all $\lambda \in \mathbb{R}^{N \cdot n_1}$.

Convergence of proximal methods relies on how to update the proximal parameter u^k and stability center $\bar{\lambda}^k$ in (6)). We adapt the proximal bundle method developed in [19] for updating the parameters. A serious step is taken to update the stability center $\bar{\lambda}^k \leftarrow \lambda^k$ if

$$D(\lambda^k) \geqslant D(\bar{\lambda}^{k-1}) + m_L v^k, \tag{9}$$

where $m_L \in (0, 0.5)$ is a parameter and

$$v^k := m_k(\lambda^k) - D(\bar{\lambda}^{k-1}) \geqslant 0$$

is a predicted increase of $D(\cdot)$. Otherwise, the method takes a null step, at which the master problem is updated by adding a set of linear inequalities (5b). The method terminates if $v^k \leqslant \epsilon$ for some $\epsilon \geqslant 0$.

The penalty parameter needs to be carefully updated for accelerating performance. For instance, the method will take serious steps at iterations with marginal improvements if the penalty parameter value is too large. On the other hand, the method will take null steps at many iterations before finding a better lower bound if the penalty parameter value is too small. The former case can be identified by the test

$$D(\lambda^k) \geqslant D(\bar{\lambda}^{k-1}) + m_R v^k, \tag{10}$$

where $m_R \in (m_L, 1)$ is a parameter. If (10) holds, we update the penalty parameter as

$$u^{k+1} = \min\{\max\{h^k, 0.1u^k, u_{min}\}, u_{max}\}, \tag{11}$$

where $h^k \leqslant u^k$. In particular, we define h^k as in [19], which is motivated and stated as follows. Assuming temporarily that D is quadratic and strictly concave, $n_1 = 1$, and $k = 1$ so that $v^k = -\sum_{j\in\mathcal{J}}(H_j x_j^{k-1})^T(\lambda^k - \bar{\lambda}^{k-1})$ where x_j^{k-1}, $j \in \mathcal{J}$, are optimal solutions for problems (4) with $\lambda = \bar{\lambda}^{k-1}$, we have

$$\begin{aligned}
D(\lambda^k) &= D(\bar{\lambda}^{k-1}) + \partial D(\bar{\lambda}^{k-1})(\lambda^k - \bar{\lambda}^{k-1}) - \frac{h^k}{2}\|\lambda^k - \bar{\lambda}^{k-1}\|^2 \\
&= D(\bar{\lambda}^{k-1}) - \sum_{j\in\mathcal{J}}(H_j x_j^{k-1})^T(\lambda^k - \bar{\lambda}^{k-1}) - \frac{h^k}{2}\|\lambda^k - \bar{\lambda}^{k-1}\|^2 \\
&= D(\bar{\lambda}^{k-1}) + u^k\|\lambda^k - \bar{\lambda}^{k-1}\|^2 - \frac{h^k}{2}\|\lambda^k - \bar{\lambda}^{k-1}\|^2,
\end{aligned}$$

where the last equality holds because when $k = 1$ holds, both $-\sum_{j\in\mathcal{J}} H_j x_j^{k-1} \in \partial m_k(\lambda^k)$ and $-\sum_{j\in\mathcal{J}} H_j x_j^{k-1} \in \partial D(\bar{\lambda}^{k-1})$ hold, and since the former subgradient is uniquely $u^k(\lambda^k - \bar{\lambda}^{k-1})$ by (7), we have $-\sum_{j\in\mathcal{J}} H_j x_j^{k-1} = u^k(\lambda^k - \bar{\lambda}^{k-1})$. As a result, the Hessian $h^k > 0$ of D is given by

$$h^k = 2u^k\left(1 - \frac{D(\lambda^k) - D(\bar{\lambda}^{k-1})}{v^k}\right). \tag{12}$$

Note that $h^k \leqslant u^k$ for $m_R \in [0.5, 1)$, because

$$2\left(1 - \frac{D(\lambda^k) - D(\bar{\lambda}^{k-1})}{v^k}\right) \leqslant 2(1 - m_R) \leqslant 1.$$

The penalty parameter u^k is identified as too small if the linearization error

$$\bar{\delta}^k := D(\lambda^k) - (\sum_{j \in \mathcal{J}} H_j x_j^k)^T(\lambda^k - \bar{\lambda}^{k-1}) - D(\bar{\lambda}^{k-1}),$$

where x_j^k, $j \in \mathcal{J}$, are optimal solutions for problems (4) with $\lambda = \lambda^k$, is larger than the variation

$$V^k = \max\left\{D(\lambda) : |\lambda - \bar{\lambda}^{k-1}| \leqslant 1\right\} - D(\bar{\lambda}^{k-1}).$$

The variation can be bounded above by

$$\begin{aligned} V^k &\leqslant m_k(\lambda^k) + \max_{|\lambda - \bar{\lambda}^{k-1}| \leqslant 1} \left\{(g^k)^T(\lambda - \lambda^k)\right\} - D(\bar{\lambda}^{k-1}) \\ &= m_k(\lambda^k) + (g^k)^T(\bar{\lambda}^{k-1} - \lambda^k) + \max_{|\lambda - \bar{\lambda}^{k-1}| \leqslant 1} \left\{(g^k)^T(\lambda - \bar{\lambda}^{k-1})\right\} - D(\bar{\lambda}^{k-1}) \\ &= \delta^k(\bar{\lambda}^{k-1}) + |g^k|, \end{aligned}$$

for $g^k \in \partial m_k(\lambda^k)$ with $\delta^k(\lambda) := m_k(\lambda^k) + (g^k)^T(\lambda - \lambda^k) - D(\lambda)$ and where the inequality holds due to (8). As a result, we use the test

$$\bar{\delta}^k > \max\{\delta^k(\bar{\lambda}^{k-1}) + |g^k|, 10v^k\} \tag{13}$$

for determining the decrease of u^k.

We present the detailed algorithmic steps in Algorithm 1. The algorithm starts at $k = 0$ by initializing parameters, solving the Lagrangian dual with $\lambda = \lambda^0$, and constructing and adding cuts (5b) to the master problem (5) (lines 1–3). (Note that each solve of the Lagrangian dual subproblem (line 2 and lines 8–10) can be solved for each scenario $j \in \mathcal{J}$ in parallel as in [17, 13, 16].) At iterations $k > 0$, the algorithm first solves the master problem in line 6 to obtain an updated trial λ^k, followed by solving the Lagrange dual subproblem with $\lambda = \lambda^k$ (lines 8–10). Next, serious step tests (9) determining the stability center $\bar{\lambda}^k$ update are taken. If the test holds, the algorithm takes the serious step (lines 14–25), where the stability center and penalty parameter are updated. Otherwise, the null step is taken (lines 26–38). In either case, the master problem is updated by adding cuts (5b). Counter i_u^k is updated for the number of consecutive serious steps (lines 21–25) or null steps (lines 33–37), thereby allowing for additional tuning of the penalty parameter updates. At the end of each iteration, the model is updated to m^{k+1} by constructing and adding cuts (5b). The method repeats the steps with new trial points until the predicted increase v^k is within a given tolerance ϵ.

The convergence of Algorithm 1 has been proved, for example, in [19].

3 Branch-and-Bound Methods with Dual Decomposition

The optimal value of the root node Lagrangian dual problem (3) with $G = G^{\mathcal{N}^0}$ and the optional value of the root node problem (1) with $G = G^{\mathcal{N}^0}$ may have a positive duality gap. In this case a branch-and-bound (BB) method

Algorithm 1 Proximal Bundle Method for the Lagrangian Dual Problem

```
1: Initialize $\lambda^0$, $\bar{\lambda}^0 \leftarrow \lambda^0$, $u^1 > u_{min}$, $i_u^1 \leftarrow 0$, and $k \leftarrow 0$
2: For each $j \in \mathcal{J}$, solve problem (4) with $\lambda = \lambda^0$
3: Add cuts for $j \in \mathcal{J}$ of form (5b) with $l = 0$ to the model (5) to obtain $m^1$

4: repeat
5:    $k \leftarrow k + 1$
6:    Solve the master (6) to find $\lambda^k$ and the value $m^k(\lambda^k)$
7:    Compute $g^k \leftarrow u^k(\lambda^k - \bar{\lambda}^{k-1}) \in \partial m_k(\lambda^k)$
8:    For each $j \in \mathcal{J}$, solve problem (4) with $\lambda = \lambda^k$
9:          for $D_j(\lambda^k)$ and optimal solution $x_j^k$
10:         compute $\bar{g}^k \leftarrow -\sum_{j\in\mathcal{J}} H_j x_j^k \in \partial D(\lambda^k)$
11:   Compute $\bar{v}^k \leftarrow D(\lambda^k) - D(\bar{\lambda}^{k-1})$,   $v^k \leftarrow m_k(\lambda^k) - D(\bar{\lambda}^{k-1})$
12:      $\bar{\delta}^k \leftarrow \bar{v}^k + (\bar{g}^k)^\top(\lambda^k - \bar{\lambda}^{k-1})$,   $\delta^k \leftarrow v^k + (g^k)^\top(\lambda^k - \bar{\lambda}^{k-1})$
13:      $h^k \leftarrow 2u^k\left(1 - \frac{\bar{v}^k}{v^k}\right)$,
14:   if $\bar{v}^k \geqslant m_L v^k$ then
15:      $\bar{\lambda}^k \leftarrow \lambda^k$
16:      if $\bar{v}^k \geqslant m_R v^k$ and $i_u^k > 0$ then
17:         $u^{k+1} \leftarrow \max\{h^k, 0.1u^k, u_{min}\}$
18:      else if $i_u^k > 3$ then
19:         $u^{k+1} \leftarrow \max\{0.5u^k, u_{min}\}$
20:      end if
21:      if $u^{k+1} = u^k$ then
22:         $i_u^{k+1} \leftarrow \max\{i_u^k + 1, 1\}$
23:      else
24:         $i_u^{k+1} \leftarrow 1$
25:      end if
26:   else
27:      $\bar{\lambda}^k \leftarrow \bar{\lambda}^{k-1}$
28:      if $\bar{\delta}^k > \max\{\delta^k + |g^k|, 10v^k\}$ and $i_u^k < -3$ then
29:         $u^{k+1} \leftarrow \min\{h^k, 10u^k\}$
30:      else
31:         $u^{k+1} \leftarrow 10u^k$
32:      end if
33:      if $u^{k+1} = u^k$ then
34:         $i_u^{k+1} \leftarrow \min\{i_u^k - 1, -1\}$
35:      else
36:         $i_u^{k+1} \leftarrow -1$
37:      end if
38:   end if
39:   Add cuts for $j \in \mathcal{J}$ of form (5b) with $l = k$ to the model (5) to obtain
      $m^{k+1}$
40: until $v^k \leqslant \epsilon$
41: $\bar{\lambda}^* \leftarrow \bar{\lambda}^k$, $z_{LD}^* \leftarrow D(\bar{\lambda}^*)$, $(\breve{x}, \breve{y}) \leftarrow (x^k, y^k)$
42: Recover the optimal dual multipliers $\hat{\alpha}^l$ corresponding to the cuts (5b) for
    $l = 0, \ldots, k$
43: $(\hat{x}, \hat{y}) \leftarrow \sum_{l=0}^k \hat{\alpha}_l (x^l, y^l)$
44: return $\bar{\lambda}^*$, $z_{LD}^*$, $(x_j^l, y_j^l)_{l=1,\ldots,k}$, $(\hat{\alpha}^l)_{l=0,\ldots,k}$, $(\breve{x}, \breve{y})$, $(\hat{x}, \hat{y})$
```

can be used to close the duality gap and under certain conditions identify a globally optimal solution. In this section, we present a BB method in the DD framework, where each BB node $\mathcal{N}$ lower bound is computed by applying Algorithm 1 to problem (1) with $G = G^{\mathcal{N}}$. The resulting optimal value is actually the optimal value for the Lagrangian dual problem (3) with $G = G^{\mathcal{N}}$; hence it is a lower bound on the optimal value of problem (1) with $G = G^{\mathcal{N}}$.

We present the existing branching methods and a new branching method. Let $\mathcal{T}$ be the BB search tree in which each node is denoted by $\mathcal{N}$. The node $\mathcal{N}$ specific realizations $G^{\mathcal{N}} := (G_j^{\mathcal{N}})_{j\in\mathcal{J}}$ of the constraint sets G are given by

$$G_j^{\mathcal{N}} := G_j^{\mathcal{N}^0} \cap C_j^{\mathcal{N}}, \tag{14}$$

where $C_j^{\mathcal{N}} \subset \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$ is the node $\mathcal{N}$ specific constraint set (due to cuts and bounds) on (x, y).

3.1 Existing Branching Methods

We present two existing methods that can be used in the DD framework.

3.1.1 Carøe-and-Schultz Branching Method

We first discuss the branching method used in [9]. A key idea of this branching method is to compromise the nonanticipativity constraints (1b) by branching on first-stage variables only. To create branches, the branching method uses the average point of the first-stage variable values $\check{x}$ returned by Algorithm 1, denoted by

$$\bar{x} := \sum_{j\in\mathcal{J}} p_j \check{x}_j, \tag{15}$$

where $\check{x}_j$ is taken from an optimal solution $(\check{x}_j, \check{y}_j)$ for each scenario j subproblem (4) with $G = G^{\mathcal{N}}$ and $\lambda = \bar{\lambda}^*$, where $\bar{\lambda}^*$ is an optimal dual solution generated (within tolerance) by Algorithm 1. Note that the BB node subproblem is fathomed due to optimality with feasible first-stage solution $\check{x}_j$, if the first-stage variable values are the same for all $j \in \mathcal{J}$ (i.e., $\check{x}_1 = \check{x}_2 = \cdots = \check{x}_N$). We note that these constraints must be met exactly for fathoming by optimality, in the sense that they are met exactly when, for example, $\bar{x}_i \in \{x_i^{LB}, x_i^{UB}\}$ for all $i = 1, \ldots, n_1$. (See Proposition 2 given later.) That is, this requirement is a practical condition that is realistically met for some of the node subproblems $\mathcal{N}$. Otherwise, the first-stage variable values $\check{x}_j$, $j \in \mathcal{J}$ could be averaged and rounded as part of a primal heuristic to provide feasible solutions to (1). Such heuristics are described in Section 4 and applied in the experiments in Section 5.

If $\bar{x}$ violates integrality for some variable $\bar{x}_i$ (i.e., $\lfloor \bar{x}_i \rfloor < \bar{x}_i < \lceil \bar{x}_i \rceil$), two nodes $\mathcal{N}^L$ and $\mathcal{N}^R$ with corresponding subproblems are created from the parent node $\mathcal{N}$ by setting, respectively,

$$C_j^{\mathcal{N}^L} := C_j^{\mathcal{N}} \cap \{(x,y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \ : \ x_i \leqslant \lfloor \bar{x}_i \rfloor\} \tag{16a}$$
$$C_j^{\mathcal{N}^R} := C_j^{\mathcal{N}} \cap \{(x,y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \ : \ x_i \geqslant \lceil \bar{x}_i \rceil\}, \tag{16b}$$

for $j \in \mathcal{J}$. (See (14).) If $\bar{x}$ satisfies integrality but the $\check{x}_j$, $j \in \mathcal{J}$, violate the nonanticipativity constraints, two nodes $\mathcal{N}^L$ and $\mathcal{N}^R$ with corresponding subproblems are created from the parent node $\mathcal{N}$ by setting, respectively,

$$C_j^{\mathcal{N}^L} := C_j^{\mathcal{N}} \cap \{(x,y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \ : \ x_i \leqslant \bar{x}_i - \epsilon_{\text{BB}}\} \tag{17a}$$
$$C_j^{\mathcal{N}^R} := C_j^{\mathcal{N}} \cap \{(x,y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \ : \ x_i \geqslant \bar{x}_i + \epsilon_{\text{BB}}\}, \tag{17b}$$

for $j \in \mathcal{J}$, where $\epsilon_{\text{BB}} \geqslant 0$ has the purpose of forcing disjunction of the resulting node subproblem feasible sets. Note that the disjunction created by (16) may be stronger than that created by (17) when $\epsilon_{\text{BB}} < 1/2$. Also note that disjunctions created with a large value of ϵ_{BB} can potentially cut away optimal solutions.

For the continuous branching disjunctions of form (17), the branching index i is chosen based on some measure of violation, in this case dispersion, of the nonanticipativity constraints. To measure the dispersion of the first-stage solution $\check{x}$ returned from Algorithm 1, we use the dispersion measure σ_i, for $i = 1, \ldots, n_1$, specified in Section 4 of [9] given by

$$\sigma_i := \max_j \check{x}_i^j - \min_j \check{x}_i^j, \tag{18}$$

where $\check{x}_j := (\check{x}_{ij})_{i=1,\ldots,n_1}$, and $\bar{x}$ are defined the same as for equation (15). The use of positive convergence tolerance $\epsilon > 0$ is necessary for finite termination. Note that z_{UB} can be updated without the application of a primal heuristic only if $\max_{i=1,\ldots,n_1} \sigma_i = 0$.

We highlight that *this branching method is scalable because the number of possible branching steps is not affected by the number of scenarios.* But it has the main drawback that the dispersion (18) used to inform the continuous disjunctive branching (17) takes into account only the last vertex $(\check{x}_j, \check{y}_j)_{j \in \mathcal{J}}$ generated from solving the node $\mathcal{N}$ subproblems (1) with $G = G^{\mathcal{N}}$. In actuality, some of the other vertices generated from the previous solves account for dispersion also.

Algorithm 2 summarizes the BB method presented in [9]. The BB method is initialized with an empty node tree $\mathcal{T}$, best bounds z_{UB}, and z_{LB} and a positive nonanticipativity tolerance $\epsilon_{\text{BB}} > 0$ (line 1) and creates a root node (line 2–4). Note that for each $z^{\mathcal{N}}$ computed at each node $\mathcal{N}$, and z_{LD} optimal for problem (3) with $G = G^{\mathcal{N}}$, we have $z_{\text{UB}} \geqslant z^{\mathcal{N}} = z_{\text{LD}} \geqslant z_{\text{LB}}$ and also that $z_{\text{LB}} = z_{\text{LD}}$ at the root node $\mathcal{N}^0$. In lines 6–7 we choose a BB node from the tree, for which a number of search strategies can be used (e.g., depth-first search, best-bound search). For each BB node $\mathcal{N}$, we create the node

Algorithm 2 Carøe-and-Schultz BB Method

```
1: Initialize $\mathcal{T} \leftarrow \emptyset$, $z_{\text{UB}} \leftarrow \infty$, $z_{\text{LB}} \leftarrow -\infty$, $\epsilon_{\text{BB}} > 0$, $\epsilon \geqslant 0$
2: Create a root node $\mathcal{N}^0$
3:         and the corresponding node subproblem (1) with $G = G^{\mathcal{N}^0}$
4: $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{N}^0\}$
5: repeat
6:     Choose a node $\mathcal{N} \in \mathcal{T}$
7:     $\mathcal{T} \leftarrow \mathcal{T} \backslash \{\mathcal{N}\}$
8:     Apply Algorithm 1 to problem (1) with $G_j = G_j^{\mathcal{N}}$ for each $j \in \mathcal{J}$,
9:             which returns optimal value $z^{\mathcal{N}} \leftarrow z_{LD}^*$, optimal solution $\bar{\lambda}^*$,
10:            and problem (4) (with $\lambda = \lambda^*$) optimal solutions $(\breve{x}, \breve{y})$
11:    Compute $\bar{x} \leftarrow \sum_{j \in \mathcal{J}} \breve{x}_j$ as in equation (15)
12:    $\sigma_i \leftarrow \max_j \breve{x}_i^j - \min_j \breve{x}_i^j$ for $i = 1, \ldots, n_1$
13:    if $\max_{i=1,\ldots,n_1} \sigma_i = 0$ then
14:        $z_{\text{UB}} \leftarrow \min\{z_{\text{UB}}, z^{\mathcal{N}}\}$
15:    else if $\max_{i=1,\ldots,n_1} \sigma_i > \epsilon$ and $z^{\mathcal{N}} < z_{\text{UB}}$ then
16:        if $\bar{x}$ is fractional then
17:            Choose integer variable index $i$ such that $\lfloor \bar{x}_i \rfloor < \bar{x}_i < \lceil \bar{x}_i \rceil$
18:            Create two nodes $\mathcal{N}^L$ and $\mathcal{N}^R$ via (16)
19:        else
20:            Choose continuous variable index $i \in \text{argmax}_i \, \sigma_i$
21:            Create two nodes $\mathcal{N}^L$ and $\mathcal{N}^R$ via (17)
22:        end if
23:        $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{N}^L, \mathcal{N}^R\}$
24:    end if
25:    Update $z_{LB}$
26: until $\mathcal{T} = \emptyset$
```

subproblem (1) with $G = G^{\mathcal{N}}$ and apply Algorithm 1 to it (lines 8–10). Note that we can terminate Algorithm 1 if $D(\bar{\lambda}^k) \geqslant z_{\text{UB}}$. If the nonanticipativity constraints (1b) are satisfied via the condition in line 13, the best upper bound z_{UB} may be updated (line 14). If $z^{\mathcal{N}} \geqslant z_{\text{UB}}$, the node is fathomed. Otherwise, we perform branching (lines 15–22). We choose the most fractional variable in line 17. Similarly, we choose the continuous variable with the largest dispersion σ_i in line 20. Note, however, that one can devise and use other metrics to choose branching variables for the steps in lines 17 and 20. The best lower bound is updated such that z_{LB} is the smallest value of $z^{\mathcal{N}}$ for all leaf nodes $\mathcal{N}$ that are not fathomed yet (line 25). We repeat the steps (i.e., lines 5–26) until the search tree $\mathcal{T}$ is empty.

3.1.2 Branching Method Based on Dantzig-Wolfe Decomposition

DD can be viewed as the dual of the Dantzig-Wolfe decomposition (DWD), which allows DD to use the branching methods developed for DWD. The branch-and-bound method in DWD (i.e., branch-and-price method) has been applied for SMIP in [26]. For simplification, we consider the dual of the master problem (6) without the quadratic proximal term. The corresponding dual at node $\mathcal{N}$ is given by

$$\begin{aligned}
\min_{\alpha_j^l} \quad & \sum_{l=0}^{k} \sum_{j \in \mathcal{J}^l} p_j \left(c^T x_j^l + d_j^T y_j^l\right) \alpha_j^l && (19a)\\
\text{s.t.} \quad & \sum_{l=0}^{k} \sum_{j \in \mathcal{J}^l} H_j x_j^l \alpha_j^l = 0 && (19b)\\
& \sum_{l \in \mathcal{L}^j} \alpha_j^l = 1 \quad \forall j \in \mathcal{J}, && (19c)\\
& \alpha_j^l \geqslant 0 \quad \forall j \in \mathcal{J}^l,\ l = 0,1,\ldots,k, && (19d)
\end{aligned}$$

where $w_j^l := (x_j^l, y_j^l)$ is a minimizer of (4) for given λ^l so that $w_j^l \in \operatorname{conv}(G_j^{\mathcal{N}})$ for all $j \in \mathcal{J}^l$, $l = 0,1,\ldots,k$, and $\mathcal{L}^j := \{l : j \in \mathcal{J}^l,\ l = 0,1,\ldots,k\}$. Note that α_j^l are the dual multipliers corresponding to the constraints (5b). Constraints (19c) and (19d) construct the convex combination of the minimizers (x_j^l, y_j^l) for $l = 1,\ldots,k$. Therefore, the dual (19) is equivalent to

$$\begin{aligned}
\min_{x_j, y_j} \quad & \sum_{j \in \mathcal{J}} p_j \left(c^T x_j + d_j^T y_j\right) && (20a)\\
\text{s.t.} \quad & \sum_{j \in \mathcal{J}} H_j x_j = 0, && (20b)\\
& (x_j, y_j) \in \operatorname{conv}\left(\bigcup_{l=1}^{k} (x_j^l, y_j^l)\right) \subseteq \operatorname{conv}(G_j^{\mathcal{N}}), \quad \forall j \in \mathcal{J}, && (20c)
\end{aligned}$$

where $\operatorname{conv}(G_j)$ is the convex hull of set G_j so that $G_j \subseteq \operatorname{conv}(G_j)$. In fact, for each node $\mathcal{N}$, the application of Algorithm 1 to problem (1) with $G = G^{\mathcal{N}}$ *does not solve* problem (1) with $G = G^{\mathcal{N}}$ in general. It actually solves the convexified problem (20) with the optimal solution $(\hat{x}_j, \hat{y}_j)$ returned by Algorithm 1 being optimal for (20) (e.g., Theorem 6.2 of [31]). Hence, for any $j \in \mathcal{J}$, $(\hat{x}_j, \hat{y}_j) = \sum_{l \in \mathcal{L}^j} \hat{\alpha}_j^l (x_j^l, y_j^l)$ with an optimal dual multiplier $\hat{\alpha}_j^l$ associated with the constraint (5b). With that justification, we apply Algorithm 1 in place of a specialized DWD implementation for the subproblem solutions required in the next two algorithms. In particular, while $\hat{w} := (\hat{x}_j, \hat{y}_j)_{j \in \mathcal{J}}$ satisfies the nonanticipativity constraints present in both problems (1) and (20), it does not generally satisfy the mixed-integer restrictions in the constraints of problem (1).

Suppose that $\hat{w}_{ij}$ is fractional for some element $i \in \{1,\ldots,n_1+n_2\}$ and some $j \in \mathcal{J}$. To ensure an integer-feasible solution, we can apply the branching method that creates two nodes $\mathcal{N}^L$ and $\mathcal{N}^R$ with corresponding subproblems created from the parent node $\mathcal{N}$ by setting, respectively,

$$C_j^{\mathcal{N}^L} := C_j^{\mathcal{N}} \cap \{w \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \ : w_i \leqslant \lfloor \hat{w}_{ij} \rfloor\} \tag{21a}$$

$$C_j^{\mathcal{N}^R} := C_j^{\mathcal{N}} \cap \{w \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \ : w_i \geqslant \lceil \hat{w}_{ij} \rceil\} \tag{21b}$$

for all scenarios $j \in \mathcal{J}$ if $i \leqslant n_1$ (first-stage branching) but for only one scenario $j \in \mathcal{J}$ if $n_1 < i \leqslant n_1 + n_2$ (second-stage branching), while for all other scenarios $j' \neq j$, we set $C_{j'}^{\mathcal{N}} = C_{j'}^{\mathcal{N}^L} = C_{j'}^{\mathcal{N}^R}$. That is, first-stage branching via (21) applies the same way to all scenario $j \in \mathcal{J}$ subproblems, but only to one scenario $j \in \mathcal{J}$ subproblem for second-stage branching. Therefore, the first-stage variable branching, having a stronger cutting effect, may have priority for branching over the second-stage variable branching. Also from the above observations, we note that the size of the search tree in this branching method is significantly increased with the number of scenarios and second-stage integer variables.

Algorithm 3 Branch-and-Price Method

```
 1: Initialize $\mathcal{T} \leftarrow \varnothing$, $z_{\text{UB}} \leftarrow \infty$, $z_{\text{LB}} \leftarrow -\infty$
 2: Create a root node $\mathcal{N}^0$
 3:        and the corresponding node subproblem (1) with $G = G^{\mathcal{N}^0}$
 4: $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{N}^0\}$
 5: **repeat**
 6:    Choose a node $\mathcal{N} \in \mathcal{T}$
 7:    $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\mathcal{N}\}$
 8:    Apply Algorithm 1 to the node subproblem (1) with $G = G^{\mathcal{N}}$,
 9:          which returns $z^{\mathcal{N}} \leftarrow z^*_{LD}$ and $\hat{w}_j \leftarrow (\hat{x}_j, \hat{y}_j)$ for each $j \in \mathcal{J}$
10:    **if** $\hat{w}_j$ are integer feasible for all $j$ **then**
11:       $z_{\text{UB}} \leftarrow \min\{z_{\text{UB}}, z^{\mathcal{N}}\}$
12:    **else if** $z^{\mathcal{N}} < z_{\text{UB}}$ **then**
13:       Choose integer variable index $i \in \{1, \ldots, n_1 + n_2\}$
14:             (preferring indices $i \leqslant n_1$) such that $\lfloor \hat{w}_i \rfloor < \hat{w}_i < \lceil \hat{w}_i \rceil$
15:       Create two nodes $\mathcal{N}^L$ and $\mathcal{N}^R$ via (21)
16:       $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{N}^L, \mathcal{N}^R\}$
17:    **end if**
18:    Update $z_{LB}$
19: **until** $\mathcal{T} = \varnothing$
```

We summarize the algorithmic steps in Algorithm 3 that uses the branching hyperplanes (21). The solutions $\hat{w}_j$ computed in lines 8–9 are optimal for the DWD problem (20). If $\hat{w}_j$ are integer feasible for all j, $\hat{w}$ is also optimal for problem (1) with $G = G^{\mathcal{N}}$; and the algorithm may update the best upper bound z_{UB} and fathom the node $\mathcal{N}$ (line 11). If $z^{\mathcal{N}} < z_{\text{UB}}$ for fractional $\hat{w}_j$, the branching constraint sets (21) are used to create two node subproblems (lines 13–16). We choose the most fractional variable for branching in line 13 after preferring first-stage variables. We repeat the steps of lines 5–19 until no node subproblem is available in tree $\mathcal{T}$.

3.2 New Branching Method

We present a new scalable branching method that uses the optimal restricted master solution $\hat{w}_j$ to generate the Carøe-and-Schultz (CS) branching constraints as via (16) and (17). The key idea is to address the following drawbacks in the existing branching methods. One can easily see that the branch-and-price method (Algorithm 3) does not scale with the number of scenarios for SMIP. Specifically, the number of branching candidates (i.e., integer variables) by the branch-and-price method increases as the number of scenarios increases. On the other hand, the CS branching method (Algorithm 2) does not make comprehensive use of the information generated during the application of Algorithm 1. For example, the dispersions computed in (18) take into account only the last vertex $(\check{x}, \check{y})$ returned from Algorithm 1 against the average $\bar{x}$. Furthermore, for $j \in \mathcal{J}$, the $(\check{x}_j, \check{y}_j)$ are typically not even the *unique* optimal solutions to problem (4) for $\lambda = \bar{\lambda}^*$. In fact, any (x_j^l, y_j^l), for $l = 0, \ldots, k$, returned from Algorithm 1 for which $\hat{\alpha}_j^l > 0$ is also optimal for (4), $\lambda = \bar{\lambda}^*$.

To address these drawbacks, we propose a new branching criterion that implicitly makes use of all column solution information (x_j^l, y_j^l), for $j \in \mathcal{J}$ and $l = 0, \ldots, k$, returned from Algorithm 1 so as to inform the choice of branching point. To this end, we first define the dispersion measure for all first-stage variables $i = 1, \ldots, n_1$,

$$\rho_i := \sum_{j \in \mathcal{J}} \sum_{l=0}^{k} p_j \hat{\alpha}_j^l |x_{ij}^l - \hat{x}_i|, \tag{22}$$

where $\hat{\alpha}_j^l$ and $\hat{x}_i$ are defined as in Section 3.1.2. We derive the upper bound of the dispersion measure ρ_i in Proposition 1, which is necessary in order to show the finite termination of our branching process. In preparation, we state and prove Lemma 1.

Lemma 1 *Let* $\xi^l \in [\xi^{LB}, \xi^{UB}] \subset \mathbb{R}$, $l = 0, \ldots, k$, *and* $\hat{\xi} := \sum_{l=0}^{k} \alpha^l \xi^l$, *where* $\alpha_l > 0$, $l = 0, \ldots, k$, *are multipliers such that* $\sum_{l=0}^{k} \alpha^l = 1$. *Then*

$$\sum_{l=0}^{k} \alpha^l |\xi^l - \hat{\xi}| \leqslant 2 \min\{|\xi^{LB} - \hat{\xi}|,\ |\xi^{UB} - \hat{\xi}|\}. \tag{23}$$

Proof Denote $L^- := \{l = 0, \ldots, k\ :\ \xi^l \leqslant \hat{\xi}\}$ and $L^+ := \{l = 0, \ldots, k\ :\ \xi^l > \hat{\xi}\}$. Furthermore, denote

$$\alpha^- := \sum_{l \in L^-} \alpha^l \quad \text{and} \qquad \alpha^+ := \sum_{l \in L^+} \alpha^l \tag{24a}$$

$$\xi^- := \frac{1}{\alpha^-} \sum_{l \in L^-} \alpha^l \xi^l \quad \text{and} \qquad \xi^+ := \frac{1}{\alpha^+} \sum_{l \in L^+} \alpha^l \xi^l. \tag{24b}$$

Note that $\hat{\xi} = \alpha^- \xi^- + \alpha^+ \xi^+$. Now compute

$$\sum_{l=0}^{k} \alpha^l |\xi^l - \hat{\xi}| = \sum_{l \in L^-} \alpha^l (\hat{\xi} - \xi^l) + \sum_{l \in L^+} \alpha^l (\xi^l - \hat{\xi}) \tag{25a}$$

$$= 2\alpha^- (\hat{\xi} - \xi^-), \tag{25b}$$

where the last equality follows from rearranging terms, substituting $\alpha^+ = 1 - \alpha^-$, and substituting out ξ^+ via the equality $\hat{\xi} = \alpha^- \xi^- + \alpha^+ \xi^+$. Analogously, we compute

$$\sum_{l=0}^{k} \alpha^l |\xi^l - \hat{\xi}| = 2\alpha^+ (\xi^+ - \hat{\xi}). \tag{26}$$

Thus,

$$\sum_{l=0}^{k} \alpha^l |\xi^l - \hat{\xi}| \leqslant 2 \min\{(\hat{\xi} - \xi^-),\ (\xi^+ - \hat{\xi})\} \tag{27a}$$

$$\leqslant 2 \min\{|\xi^{LB} - \hat{\xi}|,\ |\xi^{UB} - \hat{\xi}|\}. \tag{27b}$$

□

Proposition 1 *For all $i = 1, \ldots, n_1$, we have*

$$\rho_i \leqslant 2 \min\{|x_i^{LB} - \hat{x}_i|, |x_i^{UB} - \hat{x}_i|\}, \tag{28a}$$

where x_i^{LB} and x_i^{UB} are, respectively, the node subproblem specific lower and upper bounds on the first stage variables x_i for all $i = 1, \ldots, n_1$.

Proof Compute

$$\rho_i = \sum_{j \in \mathcal{J}} \sum_{l=0}^{k} p_j \alpha_j^l |x_{ji}^l - \hat{x}_i| \leqslant 2 \min\{|x_i^{LB} - \hat{x}_i|,\ |x_i^{UB} - \hat{x}_i|\}, \tag{28b}$$

where the inequality (28b) follows from Lemma 1 since $\sum_{j \in \mathcal{J}} \sum_{l=0}^{k} p_j \hat{\alpha}_j^l = 1$.

□

Note that the bounds x_i^{LB} and x_i^{UB}, $i = 1, \ldots, n_1$, are typically specified in $C^{\mathcal{N}}$, as referred to in (14). They are the same for all scenarios $j \in \mathcal{J}$ because the branching indices represent the first-stage variables $i = 1, \ldots, n_1$.

Now we develop the new termination criterion for our branching method. Similar to σ_i used for Algorithm 2, let us define a new dispersion measure as follows: for $i = 1, \ldots, n_1$,

$$\hat{\sigma}_i := \max_j x_{ij}^{l(j)} - \min_j x_{ij}^{l(j)},$$

where

$$l(j) := \arg \max_{l=0,\ldots,k} \hat{\alpha}_j^l. \tag{29}$$

Note that the scenario subproblem solution $(x_j^{l(j)}, y_j^{l(j)})_{j \in \mathcal{J}}$ generated at each node subproblem is integer feasible, since it is an optimal solution to the Lagrangian linear mixed-integer subproblem (2) for a given Lagrangian multiplier λ. Furthermore, when $\max_{i=1,\dots,n_1} \hat{\sigma}_i < \epsilon$, then $(x_j^{l(j)}, y_j^{l(j)})_{j \in \mathcal{J}}$ satisfies the nonanticipativity constraints within tolerance ϵ.

In the case that $\max_{i=1,\dots,n_1} \hat{\sigma}_i \geqslant \epsilon$, we choose $i' \in \operatorname{argmax}_{i=1,\dots,n_1} \rho_i$ as the branching index. We justify this branching index selection in the following propositions.

Lemma 2 *The number of vertices of* $\operatorname{conv}(G_j^{\mathcal{N}})$ *is bounded from above by some positive integer* V *for all scenarios* $j \in \mathcal{J}$ *and all nodes* $\mathcal{N}$*.*

Proof Denote the ordered tuplet $\iota \in \mathbb{Z}^{n_1+n_2}$, and define the sets

$$\mathcal{I}^{\text{int}} := \{i = 1, \dots, n_1 + n_2 : \; w_i \text{ has integral restriction in (1)}\}$$

and

$$G_j^{\mathcal{N},\iota} := \left\{ w \in G_j^{\mathcal{N}} \; : w_i = \iota_i \text{ for } i \in \mathcal{I}^{\text{int}} \right\}.$$

Since $G_j^{\mathcal{N}} \subseteq G_j^{\mathcal{N}^0}$ for all nodes $\mathcal{N}$ and $G_j^{\mathcal{N}^0}$ is assumed to be bounded, then $G_j^{\mathcal{N},\iota}$ is nonempty for a finite number of ι. Denote this bound by $\kappa \in \mathbb{Z}_+$. Also note that $G_j^{\mathcal{N}} = \bigcup_\iota G_j^{\mathcal{N},\iota}$, and each $G_j^{\mathcal{N},\iota}$ is a convex polydedral set, so that by the Upper Bound Theorem [28,29], the maximum number of vertices of $G_j^{\mathcal{N},\iota}$ depends only on the integer parameters $m_1, m_2, n_1, n_2, p_1, p_2$ associated with the formulation of the SMIP (1), and not on the particular instantiation of the bounds specific to node $\mathcal{N}$. Call this number Δ. Then, independent of $j \in \mathcal{J}$ and the node $\mathcal{N}$, we have the number of vertices of $\operatorname{conv}(G_j^{\mathcal{N}})$ being bounded from above by $V := \kappa\Delta$ due to the vertices of $\operatorname{conv}(G_j^{\mathcal{N}})$ being the union over the ι of the vertices of $G_j^{\mathcal{N},\iota}$.

Proposition 2 *For each* $i = 1, \dots, n_1$*, there exists a fixed* $M > 0$ *for which*

$$2 \min\{|x_i^{LB} - \hat{x}_i|, |x_i^{UB} - \hat{x}_i|\} \geqslant \rho_i \geqslant M\hat{\sigma}_i \tag{30}$$

for every node $\mathcal{N}$*.*

Proof Computing $l(j)$ as in (29) for each $j \in \mathcal{J}$ guarantees that $\hat{\alpha}_j^{l(j)} \geqslant 1/V_j^{\mathcal{N}}$, where $V_j^{\mathcal{N}}$ is the number of extreme points of $\operatorname{conv}(G_j^{\mathcal{N}})$, which, in light of the polyhedral nature of $\operatorname{conv}(G_j^{\mathcal{N}})$, must be finite. Thus, $\hat{\alpha}_j^{l(j)} \geqslant 1/V_j^{\mathcal{N}}$ follows by noting that $\sum_{l=0,\dots,k} \alpha_j^l = 1$ for each $j \in \mathcal{J}$ and that $\hat{\alpha}_j^{j(l)} = \max_{l=0,\dots,k} \alpha_j^l$. By Lemma 2, there exists a finite positive integer V for which $V_j^{\mathcal{N}} \leqslant V$ for all $j \in \mathcal{J}$ and all nodes $\mathcal{N}$. Thus, $\hat{\alpha}_j^{l(j)} \geqslant 1/V$ for all $j \in \mathcal{J}$ and all $\mathcal{N}$. Then, for each i, we have

$$\begin{aligned}
\rho_i &= \max_i \sum_j \sum_{l=0,\dots,k} p_j \alpha_j^l |x_{ij}^l - \hat{x}_i| \\
&\geqslant \sum_j p_j \hat{\alpha}_j^{l(j)} |x_{ij}^{l(j)} - \hat{x}_i| \\
&\geqslant \min_j \left\{ \frac{p_j}{V} \right\} \sum_j |x_{ij}^{l(j)} - \hat{x}_i| \\
&\geqslant \min_j \left\{ \frac{p_j}{V} \right\} \max_j |x_{ij}^{l(j)} - \hat{x}_i| \\
&\geqslant \min_j \left\{ \frac{p_j}{2V} \right\} \max_{j^1, j^2} \left\{ x_{ij^1}^{l(j^1)} - x_{ij^2}^{l(j^2)} \right\} \\
&\geqslant \min_j \left\{ \frac{p_j}{2V} \right\} \left(\max_j x_{ij}^{l(j)} - \min_j x_{ij}^{l(j)} \right) \\
&= \min_j \left\{ \frac{p_j}{2V} \right\} \hat{\sigma}_i .
\end{aligned}$$

Thus, $\rho_i \geqslant \min_j\{\frac{p_j}{2V}\}\hat{\sigma}_i$. By Proposition 1, we have $2\min\{|x_i^{UB} - \hat{x}_i|, |\hat{x}_i - x_i^{LB}|\} \geqslant \rho_i$. Thus, defining $M := \min_j\{\frac{p_j}{2V}\}$, which is fixed and positive, we have $\min\{|x_i^{UB} - \hat{x}_i|, |\hat{x}_i - x_i^{LB}|\} \geqslant M\hat{\sigma}_i$ as intended. □

Based on the dispersion measure $\hat{\sigma}_i$, we define a fathoming criterion as

$$\max_{i=1,\dots,n_1} \hat{\sigma}_i \leqslant \epsilon, \tag{31}$$

whose purpose is to guarantee finite termination. Otherwise, fathoming by optimality is justified when either the integer feasibility of $(\hat{x}_j, \hat{y}_j)$ is met, as used in Algorithm 3, or under certain conditions that guarantee $\max_{i=1,\dots,n_1} \sigma_i = 0$ exactly, such as when $\hat{x}_{ij} \in \{x_i^{LB}, x_i^{UB}\}$ for all $i = 1, \dots, n_1$ and $j \in \mathcal{J}$. The latter condition has its analog in Algorithm 2, while the former condition is an innovation to Algorithm 2.

We summarize the algorithmic steps of our new branching method in Algorithm 4. The algorithm is based on Algorithm 2 with modifications to use the optimal restricted master solutions $\hat{x}$ and their dispersions with respect to column solutions $(x_j^l, y_j^l)_{j\in\mathcal{J}}$ for $l = 0, \dots, k$, to determine branching. Hence, after applying Algorithm 1 to the node subproblem (1) with $G = G^{\mathcal{N}}$, we obtain $(\hat{x}_j, \hat{y}_j)_{j\in\mathcal{J}}$ at line 9. Note that the first-stage variable values $\hat{x}_j$ satisfy the nonanticipativity constraints; that is, $\hat{x}_j = \hat{x}_{j'}$ for any $j \neq j' \in \mathcal{J}$. If $(\hat{x}_j, \hat{y}_j)$ are integer feasible for all $j \in \mathcal{J}$, then we have an optimal solution to the node subproblem (1) with $G = G^{\mathcal{N}}$, so that node $\mathcal{N}$ is fathomed by optimality with an update of the best known upper bound z_{UB} (line 13) if appropriate. On the other hand, if $\max_i \hat{\sigma}_i \leqslant \epsilon$, we compute a candidate solution $(x_j^{l(j)}, y_j^{l(j)})_{j\in\mathcal{J}}$ that satisfies integrality constraints (by construction) and the nonanticipativity constraints within ϵ tolerance, as in Algorithm 2. In the case that exact satisfaction of $\max_i \hat{\sigma}_i = 0$ may be inferred, node $\mathcal{N}$ is fathomed by optimality, and z_{UB} is updated as warranted. Otherwise, a candidate primal solution may

Algorithm 4 Improved CS BB Method

```
1: Initialize $\mathcal{T} \leftarrow \emptyset$, $z_{\mathrm{UB}} \leftarrow \infty$, $z_{\mathrm{LB}} \leftarrow -\infty$, $\epsilon_{\mathrm{BB}} \geqslant 0$, $\epsilon \geqslant 0$
2: Create a root node $\mathcal{N}^0$ and the root node subproblem (1) with $G = G^{\mathcal{N}^0}$
3: $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{N}^0\}$
4: repeat
5:    Choose a node $\mathcal{N} \in \mathcal{T}$
6:    $\mathcal{T} \leftarrow \mathcal{T} \backslash \{\mathcal{N}\}$
7:    Apply Algorithm 1 to the node subproblem (1) with $G = G^{\mathcal{N}}$,
8:        which returns $z^{\mathcal{N}} \leftarrow z^*$
9:        and $(\hat{x}_j, \hat{y}_j)$ and $(\hat{\alpha}_j^l, x_j^l, y_j^l)_{l=0,\dots,k}$ for each $j \in \mathcal{J}$
10:   Compute $l(j) \leftarrow \arg\max_{l=0,\dots,k} \hat{\alpha}_j^l$ for each $j \in \mathcal{J}$
11:   $\hat{\sigma}_i \leftarrow \max_j x_{ij}^{l(j)} - \min_j x_{ij}^{l(j)}$ for $i = 1, \dots, n_1$
12:   if $(\hat{x}_j, \hat{y}_j)_{j \in \mathcal{J}}$ are integer feasible or $\max_{i=1,\dots,n_1} \hat{\sigma}_i = 0$ then
13:      $z_{\mathrm{UB}} \leftarrow \min\{z_{\mathrm{UB}}, z^{\mathcal{N}}\}$
14:   else if $\max_{i=1,\dots,n_1} \hat{\sigma}_i > \epsilon$ and $z^{\mathcal{N}} < z_{\mathrm{UB}}$ then
15:      Set $\mathcal{F} := \{i \in 1, \dots, n_1 :\ x_i$ integer but $\hat{x}_i$ fraction valued$\}$
16:      if $\mathcal{F}$ is nonempty ($\hat{x}$ is fractional) then
17:         Choose branching index $i' \leftarrow \mathrm{argmax}_{i \in \mathcal{F}}\{\rho_i\}$
18:         Create two nodes $\mathcal{N}^L$ and $\mathcal{N}^R$ via (16)
19:      else
20:         Choose branching index $i' \leftarrow \mathrm{argmax}_{i \in 1,\dots,n_1}\{\rho_i\}$
21:         Create two nodes $\mathcal{N}^L$ and $\mathcal{N}^R$ via (17)
22:      end if
23:      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{N}^L, \mathcal{N}^R\}$
24:   end if
25:   Update $z_{LB}$
26: until $\mathcal{T} = \emptyset$
```

be taken as the average point as in (15), that is, $\sum_{j \in \mathcal{J}} p_j x_j^{l(j)}$ and $(y_j^{l(j)})_{j \in \mathcal{J}}$. If the fathoming criterion $\max_{i=1,\dots,n_1} \hat{\sigma}_i \leqslant \epsilon$ is not met and $z^{\mathcal{N}} < z_{\mathrm{UB}}$, two child node subproblems are created by branching in lines 16–22. The branching index is chosen by $i' \in \arg\max_{i=1,\dots,n_1} \rho_i$. If i' corresponds to an integer variable, we create branching constraints $x_i \leqslant \lfloor \hat{x}_i \rfloor$ and $x_i \geqslant \lceil \hat{x}_i \rceil$. Otherwise, we create branching constraints $x_i \leqslant \hat{x}_i - \epsilon_{\mathrm{BB}}$ and $x_i \geqslant \hat{x}_i + \epsilon_{\mathrm{BB}}$ on $\hat{x}_i$. We may choose a branching index with higher preference to integer variables over continuous variables. When the set

$$\mathcal{F} := \{i = 1, \dots, n_1 :\ x_i \text{ integer but } \hat{x}_i \text{ fraction valued}\}. \tag{32}$$

is nonempty, we may choose $i' \in \mathrm{argmax}_{i \in \mathcal{F}} \rho_i$ as the branching index. Note, however, that we need not consider integer variables first for branching. We found in our numerical experiments that $i' \in \arg\max_{i=1,\dots,n_1} \rho_i$ always satisfied $i' \in \mathcal{F}$ whenever $\mathcal{F}$ was not empty. After processing a node, the best lower bound z_{LB} may be updated in line 25. We repeat the steps (i.e., lines 4–26) until the tree $\mathcal{T}$ is empty.

We show that Algorithm 4 terminates after solving a finite number of node subproblems for a sufficiently small feasibility tolerance $\epsilon > 0$.

Proposition 3 *The Algorithm 4 applied to problem* (1) *with* $\epsilon_{BB} = 0$ *and* $\epsilon > 0$ *must terminate after processing a finite number of node subproblems.*

Proof Without loss of generality, we assume that the first-stage variables are all continuous. since the finiteness argument is trivial for integrality branching. We show that for $\epsilon > 0$, the length of the bounding interval at each of the new branching nodes created in lines 18 and 21 is diminished by at least ϵM from the corresponding interval length of the parent node, where M is the positive fixed constant defined in the proof of Proposition 2.

Suppose that there exists a choice of variable index i' for branching (found in line 16) such that $\sigma_{i'} > \epsilon$. Note that we fathom the current node (by line 12) if such i' does not exist, which is not the case of interest in this proof. From Proposition 1, observe that both $\epsilon M \leqslant |x_{i'}^{LB} - \hat{x}_{i'}|$ and $\epsilon M \leqslant |x_{i'}^{UB} - \hat{x}_{i'}|$, because $\min\{|x_{i'}^{LB} - \hat{x}_{i'}|, |x_{i'}^{UB} - \hat{x}_{i'}|\} \geqslant M\sigma_{i'} > \epsilon M$ by Proposition 2. Then, using $\epsilon M \leqslant |x_{i'}^{LB} - \hat{x}_{i'}|$, we have

$$\begin{aligned} |x_{i'}^{UB} - x_{i'}^{LB}| &= |x_{i'}^{LB} - \hat{x}_{i'}| + |x_{i'}^{UB} - \hat{x}_{i'}| && \text{(33a)} \\ &\geqslant \epsilon M + |x_{i'}^{UB} - \hat{x}_{i'}|, && \text{(33b)} \end{aligned}$$

and analogously, using $\epsilon M \geqslant |x_{i'}^{UB} - \hat{x}_{i'}|$, we have

$$\begin{aligned} |x_{i'}^{UB} - x_{i'}^{LB}| &= |x_{i'}^{LB} - \hat{x}_{i'}| + |x_{i'}^{UB} - \hat{x}_{i'}| && \text{(33c)} \\ &\geqslant \epsilon M + |x_{i'}^{LB} - \hat{x}_{i'}|. && \text{(33d)} \end{aligned}$$

In both cases of branching on variables $x_{i'}$, each of the lengths $|x_{i'}^{LB} - \hat{x}_{i'}|$ and $|x_{i'}^{UB} - \hat{x}_{i'}|$ in the children nodes must be smaller by at least the fixed amount $\epsilon M > 0$ than the corresponding bound interval length $|x_{i'}^{UB} - x_{i'}^{LB}|$ of the parent node. Because we assume finite bounds on x_i for all $i = 1, \ldots, n_1$, we conclude that branching can occur only along each x_i for each $i = 1, \ldots, n_1$ a finite number of times. □

We remark that the fathoming by the optimality condition that either $(\hat{x}_j, \hat{y}_j)$ satisfies integrality or $\max_{i=1,\ldots,n_1} \sigma_i = 0$ is not necessary for the finite termination of Algorithm 4 but is practical for numerical experiments for obtaining optimality gaps $z_{\text{UB}} - z_{\text{LB}}$ and thus assessing the optimality of the incumbent solution. In particular, we found that the condition that $(\hat{x}_j, \hat{y}_j)$ satisfies integrality was always met (even before the other conditions were met) for all node subproblems in our numerical experiments.

4 Primal Heuristics

Finding a good primal feasible solution is critical for reducing the size of the search tree in branch-and-bound methods. The primal heuristic methods [5] developed for the traditional branch-and-bound method may be applicable to our decomposition setting. In this section, we describe two heuristic algorithms that are specific to the branch-and-bound method of our decomposition method.

4.1 Fixing-First Heuristic

The first heuristic is motivated by the fact that the second-stage recourse function can be evaluated in parallel once the first-stage nonanticipative variables are fixed. We call this algorithm a fixing-first heuristic; the algorithmic steps are described as follows.

We refer to x^{ref} as the first-stage primal solution obtained from Algorithms 2, 3, or 4. Specifically, $x^{\text{ref}} = \bar{x}$ for Algorithm 2, and $x^{\text{ref}} = \hat{x}$ for Algorithms 3 and 4. Note that $x^{\text{ref}} = \sum_{j\in\mathcal{J}} p_j x_j^{l(j)}$ when $\max_{i=1,\ldots,n_1} \hat{\sigma}_i < \epsilon$ for Algorithm 4. We suppose that x^{ref} is a mixed-integer variable vector and violates the integrality constraint for some i. We then round all of the fractional values of the integer variables and fix the rounded values as well as the continuous variable values. For the fixed first-stage variable value, the dual decomposition simply evaluates the individual scenario recourse function values. That is, the heuristic solves

$$\forall j \in \mathcal{J}: \quad \min_{y_j} \left\{ d_j^T y_j : W_j y_j \geqslant h_j - T_j \lfloor x^{\text{ref}} \rceil,\ y_j \in Y \right\}, \tag{34}$$

where $\lfloor\cdot\rceil$ is defined as a rounding operator for integer variables.

In our numerical experiments, we round fractional values to the nearest integer values. One can also use other ways such as randomly rounding. Note that this heuristic always finds an upper bound for complete recourse problems.

4.2 Simple Rounding Heuristic

The other heuristic we develop here is based on a simple rounding method on integer variables in both the first and second stages. We use x^{ref} as defined in the preceding section. We also define y_j^{ref} as $\check{y}_j$ for Algorithm 2 and $\hat{y}_j$ for Algorithms 3 and 4. Similarly, we use $y_j^{\text{ref}} = y_j^{l(j)}$ when $\max_{i=1,\ldots,n_1} \hat{\sigma}_i < \epsilon$ for Algorithm 4. We define $\mathcal{I}_x$ and $\mathcal{I}_y$ as the sets of indices for first- and second-stage integer variables, respectively.

By rounding and fixing all the integer variable values, the heuristic solves the dual decomposition of the resulting problem.

$$\min_{(x_j,y_j)\in G_j} \quad \sum_{j\in\mathcal{J}} p_j \left(c^T x_j + d_j^T y_j\right) \tag{35a}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{J}} H_j x_j = 0, \tag{35b}$$

$$x_{ij} = \lfloor x_i^{\text{ref}} \rceil \quad \forall i \in \mathcal{I}_x, \tag{35c}$$

$$y_{ij} = \lfloor y_{ij}^{\text{ref}} \rceil \quad \forall i \in \mathcal{I}_y,\ j \in \mathcal{J} \tag{35d}$$

The dual decomposition finds the global optimum to (35) without a branch-and-bound method, because the problem (35) is a continuous linear program. Note that this simple rounding heuristic is equivalent to the fixing-first heuristic for problems with pure integer first-stage variables and continuous second-stage variables.

5 Numerical Experiments

In this section, we present numerical experiments to compare the new branching strategy with the existing ones, as described in Section 3.

5.1 Implementation

We have implemented the two existing branching methods (Algorithms 2 and 3) and the new proposed method (Algorithm 4) in the open-source parallel software package DSP [17]. We use an open-source software package Coin-ALPS [45] that implements a parallel tree search framework. Note, however, that we implemented the serial branch-and-bound methods. For Algorithm 1, we used the branch-and-price method implemented for solving power system production cost modeling in a long planning horizon [14]. The bundle master problem (6) and the Lagrangian subproblems (4) were solved by CPLEX 12.7 with default settings.

We observed that CPLEX experienced numerical stability issues for the master problem at a few iterations (particularly after adding many cuts). For such cases, we re-solved the master problem by temporarily setting the parameter value `CPX_PARAM_NUMERICALEMPHASIS` to 1. We have also implemented the warm-starting capability that reuses the bundle information generated from parent nodes. Because of the numerical instability, however, we report the numerical results without the warm-starting capability.

We have also implemented the primal heuristic algorithms described in Section 4. Here we report the numerical results without any heuristic, with the fixing-first heuristic, and with the simple rounding heuristic. The heuristic algorithm is called at the end of each subproblem solution. Rounding is performed to the nearest integer values. We emphasize that variations of the heuristic methods are possible but not reported here.

The experiments were run on the Bebop cluster at Argonne National Laboratory. The cluster has 664 Intel Broadwell nodes, each of which consists of 36 cores and 128 GB of RAM. Each run used 36 cores on single compute node to parallelize each node subproblem solution by scenario decomposition. The scenario subproblems were distributed to the 36 processes in a round-robin fashion. We set the parameters $u_{min} = 10^{-4}, m_L = 0.1, m_R = 0.5, \epsilon = 10^{-6}$, and $\epsilon_{\mathrm{BB}} = 10^{-6}$. Moreover, we use $\max_{i=1,\ldots,n_1} \sigma_i \leqslant \epsilon$ and $\max_{i=1,\ldots,n_1} \hat{\sigma}_i \leqslant \epsilon$ in place of $\max_{i=1,\ldots,n_1} \sigma_i = 0$ and $\max_{i=1,\ldots,n_1} \hat{\sigma}_i = 0$ in Algorithms 2 and 4, respectively, for practical purposes. At each node subproblem, the initial dual variable values were set to zero.

5.2 Problem instances

We use four test problem instances (`dcap`, `semi`, `sizes`, and `sslp`) from the SIPLIB available at http://www2.isye.gatech.edu/~sahmed/siplib/. Characteristics of the problem instances are given in Table 1.

All the problem instances have integer variables in the first and second stages except that `semi` instances have integer variables in the first stage only. The test problem instances allow us to test our method for various classes of SMIP. In particular, `dcap` has mixed-binary variables in the first stage and pure-binary variables in the second stage, whereas `sslp` has pure-binary variables in the first stage and mixed-binary variables in the second stage; and `sizes` has mixed-binary variables in both the first and second stages. For each problem instance, experiments are conducted for various numbers of scenarios. We note that the dual decomposition may not be the best algorithm for `semi` instances. For example, Benders decomposition is able to find a global optimal solution for `semi` instances, since no integer variable appears in the second stage.

In the Appendix, we also report the numerical results for `smkp` instances that have pure binary variables in both the first and second stages. As discussed in [40], these instances have a larger number of constraints and variables in the first stage than those in the second stage, which makes the instances particularly challenging for the dual decomposition. Therefore, the numerical results were obtained with a slightly different setting of parameters for our solver.

5.3 Benchmarking Branching Methods

We present the numerical results from different branching methods without any heuristic: CS of Algorithm 2, BNP of Algorithm 3, and CS+DW of Algorithm 4. The results were obtained without using any heuristic. Table 2 reports the best upper and lower bounds, numbers of nodes solved and left,

Table 1 Characteristics of the problem instances

		First Stage			Second Stage		
Name	Scen (N)	Cons	Vars	Ints	Cons	Vars	Ints
dcap233	200, 300, 500	6	12	6	15	27	27
dcap243	200, 300, 500	6	12	6	18	36	36
dcap332	200, 300, 500	6	12	6	12	24	24
dcap342	200, 300, 500	6	12	6	14	32	32
semi	2, 3, 4	2	612	612	5842	9802	0
sizes	3, 5, 10	31	75	10	31	75	10
sslp_5_25	50, 100	1	5	5	30	130	125
sslp_10_50	50, 100, 500, 1000, 2000	1	10	10	60	510	500
sslp_15_45	50, 100	1	15	15	60	690	675

and solution time in seconds for all `dcap`, `semi`, `sizes`, and `sslp` instances. The instances were run with a 2-hour wall-clock time limit. The instance that reached the time limit sre denoted by “TO” in the table. We set $\epsilon_{\text{BB}} = 10^{-6}$ for CS and $\epsilon_{\text{BB}} = 0$ and $\epsilon = 10^{-6}$ for CS+DW. The feasibility tolerance $\epsilon = 10^{-6}$ was also used for checking the sum of integrality violations in Algorithms 3 and 4. The optimality gap $< 0.01\%$ is considered optimum, denoted by “OPT.” The best known upper bound, denoted by “UB,” can be ∞ (so is “Gap”), if no upper bound is found.

Table 2 Computational results for `dcap`, `semi`, `sizes`, and `sslp` instances by using different branching methods in the dual decomposition

Instance	Method	UB	LB	Gap (%)	Nodes Solved	Nodes Left	Time
dcap233_200	BNP	∞	1833.48	∞	1740	1741	TO
	CS	∞	1833.48	∞	633	631	TO
	CS+DW	1834.57	1834.55	OPT	55	0	230
dcap233_300	BNP	∞	1642.75	∞	402	401	TO
	CS	∞	1642.75	∞	35	34	TO
	CS+DW	1644.20	1644.19	OPT	56	0	1119
dcap233_500	BNP	∞	1736.69	∞	109	108	TO
	CS	1739.07	1736.69	0.13	236	133	TO
	CS+DW	1737.54	1737.52	OPT	46	0	535
dcap243_200	BNP	∞	2321.28	∞	742	739	TO
	CS	∞	2321.28	∞	365	347	TO
	CS+DW	2322.49	2322.49	OPT	159	0	1811
dcap243_300	BNP	∞	2556.60	∞	538	537	TO
	CS	∞	2556.74	∞	1401	184	TO
	CS+DW	2559.19	2559.19	OPT	28	0	319
dcap243_500	BNP	∞	2165.45	∞	248	247	TO
	CS	∞	2165.50	∞	100	99	TO
	CS+DW	2167.36	2167.35	OPT	219	0	6936
dcap332_200	BNP	∞	1059.08	∞	643	642	TO
	CS	∞	1059.10	∞	278	276	TO
	CS+DW	1060.70	1060.65	OPT	119	0	498

continued on next page

continued from previous page

Instance	Method	UB	LB	Gap (%)	Nodes Solved	Nodes Left	Time
dcap332_300	BNP	∞	1250.84	∞	282	279	TO
	CS	∞	1250.94	∞	272	271	TO
	CS+DW	1252.75	1252.75	OPT	191	0	4161
dcap332_500	BNP	∞	1586.95	∞	136	137	TO
	CS	∞	1587.09	∞	227	226	TO
	CS+DW	∞	1587.09	∞	176	175	TO
dcap342_200	BNP	∞	1618.11	∞	529	528	TO
	CS	∞	1618.09	∞	430	429	TO
	CS+DW	∞	1618.03	∞	613	612	TO
dcap342_300	BNP	∞	2065.51	∞	308	307	TO
	CS	∞	2065.44	∞	114	112	TO
	CS+DW	2067.54	2067.49	OPT	225	0	2869
dcap342_500	BNP	∞	1902.90	∞	149	148	TO
	CS	∞	1903.02	∞	138	137	TO
	CS+DW	∞	1904.65	∞	77	76	TO
semi2	BNP	∞	147	∞	3	3	TO
	CS	∞	147	∞	1	1	TO
	CS+DW	∞	147	∞	3	3	TO
semi3	BNP	∞	147	∞	1	1	TO
	CS	∞	147	∞	1	1	TO
	CS+DW	∞	147	∞	1	1	TO
semi4	BNP	∞	147	∞	1	1	TO
	CS	∞	147	∞	1	1	TO
	CS+DW	∞	147	∞	1	1	TO
sizes3	BNP	226191	226190	OPT	176	0	59
	CS	226191	226191	OPT	637	0	252
	CS+DW	226191	226179	OPT	70	0	32
sizes5	BNP	225553	225530	OPT	1661	0	1053
	CS	∞	225356	∞	286	282	TO
	CS+DW	225535	225514	OPT	163	0	142
sizes10	BNP	∞	224319	∞	981	980	TO
	CS	∞	224319	∞	735	732	TO
	CS+DW	224565	224564	OPT	774	0	2676
sslp_5_25_50	BNP	-121.6	-121.6	OPT	1	0	1
	CS	-121.6	-121.6	OPT	4	0	1
	CS+DW	-121.6	-121.6	OPT	1	0	1
sslp_5_25_100	BNP	-127.37	-127.37	OPT	1	0	1
	CS	-127.37	-127.37	OPT	3	0	1
	CS+DW	-127.37	-127.37	OPT	1	0	1
sslp_10_50_50	BNP	-364.64	-364.64	OPT	1	0	31
	CS	-364.64	-364.64	OPT	21	0	112
	CS+DW	-364.64	-364.64	OPT	1	0	31
sslp_10_50_100	BNP	-354.19	-354.19	OPT	1	0	31
	CS	-354.19	-354.19	OPT	20	0	122
	CS+DW	-354.19	-354.19	OPT	1	0	31
sslp_10_50_500	BNP	-349.136	-349.136	OPT	1	0	651
	CS	-349.136	-349.137	OPT	20	0	953
	CS+DW	-349.136	-349.136	OPT	1	0	652
sslp_10_50_1000	BNP	-351.711	-351.711	OPT	1	0	1256
	CS	-351.711	-351.711	OPT	20	0	2683

continued on next page

continued from previous page

Instance	Method	UB	LB	Gap (%)	Nodes Solved	Nodes Left	Time
	CS+DW	-351.711	-351.711	OPT	1	0	1258
sslp_10_50_2000	BNP	-347.263	-347.263	OPT	1	0	2603
	CS	-347.265	-347.262	OPT	19	0	6984
	CS+DW	-347.263	-347.263	OPT	1	0	2729
sslp_15_45_5	BNP	-262.4	-262.4	OPT	1	0	4
	CS	-262.4	-262.4	OPT	1	0	4
	CS+DW	-262.4	-262.4	OPT	1	0	4
sslp_15_45_10	BNP	-260.5	-260.5	OPT	1	0	101
	CS	-260.5	-260.5	OPT	1	0	101
	CS+DW	-260.5	-260.5	OPT	1	0	101
sslp_15_45_15	BNP	-253.601	-253.601	OPT	1	0	267
	CS	-253.601	-253.626	OPT	1	0	264
	CS+DW	-253.601	-253.601	OPT	1	0	267

The dual decomposition with CS+DW (i.e., Algorithm 4) found global optimal solutions for 9 out of 12 `dcap` instances (i.e., all but `dcap`332_500 and `dcap`342_500) and all the `sslp` and `sizes` instances. In contrast, within the 2-hour time limit, BNP and CS did not find an optimum for any `dcap` instances. The use of CS+DW also saw improvement in branching that led to a reduced number of node subproblem solutions required, compared with the existing branching methods CS and BNP. For the `dcap`_243_300 instance, CS could not find an upper bound after solving 1,401 node subproblems, whereas CS+DW found an optimum after solving 28 node subproblems. BNP solved more node subproblems, while resulting in slightly worse lower bounds, than did CS for most `dcap` instances. The reason is that it branches on both first- and second-stage integer variables, the number of which increases with the number of scenarios.

We observe that the method that solved more node subproblems differed depending on the time. For example, CS and CS+DW methods solved 114 and 225 node subproblems for `dcap`_342_300, respectively. Another example is that CS and CS+DW methods solved 137 and 77 node subproblems for `dcap`_342_500, respectively. We found that the different solution times in the dual decomposition of node subproblems are due to different paths taken from the root node to leaf nodes, the different numbers of dual decomposition iterations, and the numerical conditions of the master problems.

For `semi` instances, none of the methods was able to find an upper bound of the problem instances within the 2-hour time limit. The reason is that each scenario subproblem is difficult to solve and not enough time was allowed for exploring multiple node subproblems. We note that the global optimal values were found by using the heuristic methods described in Section 4 and reported in the following sections.

The CS+DW method found the global optimal solutions for all `sizes` instances. Consistent with the computational results for `dcap` instances, the CS+DW method solved fewer node subproblems than the other methods did, however. For the given time limit, CS could only solve fewer node subproblems for the `sizes`5 instances than the CS+DW method did, because one particular node subproblem was difficult to solve and took a significant amount of time.

The BNP and CS+DW methods found the global optimum at the root node for all the `sslp` instances, because the optimal restricted master solutions (i.e., $(\hat{x}_j, \hat{y}_j)$) at the root node were global optima. On the other hand, CS explored more nodes to prove the optimality (e.g., `sslp`_5_25_N and `sslp`_10_50_N), because it does not check the restricted master solution for branching. Consequently, CS took a much longer solution time than did the other methods (e.g., `sslp`_10_50_100 by a factor 4).

We note that all node subproblems by CS+DW method were fathomed by the integer feasibility criterion, even before the criterion $\max_{i=1,\dots,n_1} \hat{\sigma}_i < \epsilon$ was checked.

5.4 Benchmarking Branching Methods with Fixing-First Heuristic

In this section, we consider the fixing-first heuristic, as described in Section 4.1, to find upper bounds after each node subproblem is solved. The heuristic was run at the first-stage primal solutions x^{ref} (as defined in Section 4.1) obtained after solving the node subproblem at each algorithm.

Table 3 Computational results for `dcap`, `semi`, `sizes`, and `sslp` instances by using different branching methods in the dual decomposition with fixing-first heuristic

Instance	Method	UB	LB	Gap (%)	Nodes Solved	Nodes Left	Heur. Time	Total Time
dcap233_200	BNP	1846.64	1833.45	0.71	1554	1553	198	TO
	CS	1834.63	1834.50	OPT	35	0	2	94
	CS+DW	1834.58	1834.54	OPT	36	0	2	130
dcap233_300	BNP	1671.87	1642.75	1.74	392	391	120	TO
	CS	1644.25	1644.18	OPT	64	0	11	4137
	CS+DW	1644.25	1644.18	OPT	32	0	5	281
dcap233_500	BNP	1779.95	1736.70	2.42	111	110	64	TO
	CS	1737.61	1737.50	OPT	41	0	9	505
	CS+DW	1737.52	1737.52	OPT	24	0	6	370
dcap243_200	BNP	2336.56	2321.21	0.65	790	787	133	TO
	CS	2322.65	2322.46	OPT	40	0	3	2322
	CS+DW	2322.58	2322.48	OPT	34	0	2	128
dcap243_300	BNP	2590.27	2556.72	1.29	545	544	174	TO
	CS	2559.34	2559.12	OPT	57	0	9	381
	CS+DW	2559.19	2559.18	OPT	21	0	3	123
dcap243_500	BNP	2197.13	2165.50	1.43	251	250	140	TO
	CS	2167.48	2167.30	OPT	55	0	15	1157
	CS+DW	2167.39	2167.32	OPT	44	0	13	614
dcap332_200	BNP	1082.83	1059.10	2.19	637	636	85	TO
	CS	1060.78	1059.10	0.15	392	192	43	TO
	CS+DW	1060.75	1060.64	OPT	89	0	5	369
dcap332_300	BNP	1309.43	1250.94	4.46	180	179	59	TO
	CS	1252.85	1252.72	OPT	407	0	61	4793

continued on next page

continued from previous page

Instance	Method	UB	LB	Gap (%)	Nodes Solved	Nodes Left	Heur. Time	Total Time
	CS+DW	1252.76	1252.67	OPT	61	0	9	677
dcap332_500	BNP	1694.22	1587.09	6.32	140	141	70	TO
	CS	1589.28	1587.09	0.13	252	217	119	TO
	CS+DW	1588.85	1588.69	OPT	379	0	93	6974
dcap342_200	BNP	1642.72	1618.09	1.49	515	514	79	TO
	CS	1619.63	1619.47	OPT	543	0	38	2792
	CS+DW	1619.56	1619.47	OPT	196	0	14	842
dcap342_300	BNP	2129.66	2065.44	3.01	297	296	97	TO
	CS	2067.63	2065.44	0.10	182	62	40	TO
	CS+DW	2067.50	2067.41	OPT	119	0	18	742
dcap342_500	BNP	2006.01	1903.00	5.13	148	149	79	TO
	CS	1904.81	1903.02	0.09	169	124	89	TO
	CS+DW	1904.72	1904.57	OPT	79	0	19	1537
semi2	BNP	147	147	OPT	1	0	1	1674
	CS	∞	147	∞	1	1	1	TO
	CS+DW	147	147	OPT	1	0	1	1664
semi3	BNP	147	147	OPT	1	0	1	2917
	CS	∞	147	∞	1	1	1	TO
	CS+DW	147	147	OPT	1	0	1	2889
semi4	BNP	147	147	OPT	1	0	1	4165
	CS	1889.47	147	92.22	1	1	1	TO
	CS+DW	147	147	OPT	1	0	1	4223
sizes3	BNP	226191	226168	OPT	97	0	1	33
	CS	226192	226191	OPT	637	0	1	254
	CS+DW	226196	226179	OPT	70	0	1	33
sizes5	BNP	225532	225524	OPT	1313	0	11	865
	CS	225555	225532	OPT	241	0	1	215
	CS+DW	225532	225512	OPT	147	0	1	130
sizes10	BNP	224600	224319	0.12	986	971	30	TO
	CS	224584	224564	OPT	1272	0	13	2443
	CS+DW	224579	224562	OPT	725	0	9	2545
sslp_5_25_50	BNP	-121.6	-121.6	OPT	1	0	0	1
	CS	-121.6	-121.6	OPT	1	0	1	1
	CS+DW	-121.6	-121.6	OPT	1	0	0	1
sslp_5_25_100	BNP	-127.37	-127.37	OPT	1	0	0	1
	CS	-127.37	-127.37	OPT	1	0	1	1
	CS+DW	-127.37	-127.37	OPT	1	0	0	1
sslp_10_50_50	BNP	-364.64	-364.64	OPT	1	0	0	31
	CS	-364.64	-364.64	OPT	1	0	1	32
	CS+DW	-364.64	-364.64	OPT	1	0	0	32
sslp_10_50_100	BNP	-354.19	-354.19	OPT	1	0	0	35
	CS	-354.19	-354.19	OPT	1	0	1	32
	CS+DW	-354.19	-354.19	OPT	1	0	0	35
sslp_10_50_500	BNP	-349.136	-349.136	OPT	1	0	0	649
	CS	-349.136	-349.136	OPT	1	0	1	653
	CS+DW	-349.136	-349.136	OPT	1	0	0	650
sslp_10_50_1000	BNP	-351.711	-351.711	OPT	1	0	0	1256
	CS	-351.711	-351.711	OPT	1	0	1	1257
	CS+DW	-351.711	-351.711	OPT	1	0	0	1257
sslp_10_50_2000	BNP	-347.263	-347.263	OPT	1	0	0	3072

continued on next page

continued from previous page

Instance	Method	UB	LB	Gap (%)	Nodes Solved	Nodes Left	Heur. Time	Total Time
	CS	-347.263	-347.263	OPT	1	0	3	3425
	CS+DW	-347.263	-347.263	OPT	1	0	0	3961
sslp_15_45_5	BNP	-262.4	-262.4	OPT	1	0	0	5
	CS	-262.4	-262.4	OPT	1	0	1	4
	CS+DW	-262.4	-262.4	OPT	1	0	0	4
sslp_15_45_10	BNP	-260.5	-260.5	OPT	1	0	0	101
	CS	-260.5	-260.5	OPT	1	0	1	100
	CS+DW	-260.5	-260.5	OPT	1	0	0	101
sslp_15_45_15	BNP	-253.601	-253.601	OPT	1	0	0	268
	CS	-253.601	-253.601	OPT	1	0	1	265
	CS+DW	-253.601	-253.601	OPT	1	0	0	268

Table 3 shows the numerical results for solving the `dcap`, `semi`, `sizes`, and `sslp` instances within the 2-hour time limit. We report the heuristic running time in the "Heur. Time" column of the table. The solution times of CS and CS+DW were significantly reduced for all the instances. CS+DW found global optimal solutions for *all the test instances.* We also emphasize that for almost all instances, CS+DW outperformed CS by exploring consistently fewer node subproblems and thus reducing the solution times. CS found optimal solutions for 8 out of 12 `dcap` instances and all `sizes` instances; in contrast, none of the `dcap` instances and only one `sizes` instance found the global optimum without the heuristic. For a few instances (i.e., `dcap`233_200 and `dcap`233_500), the computational performances (i.e., number of nodes and solution times) resulting from CS are comparable to those from CS+DW. This is due to both methods finding a good upper bound allowing for a reduction of the search tree size early in the search. However, CS found no upper bound or only a poor one for `semi` instances. On the other hand, BNP could not find a global optimal solution for any `dcap` instances within the time limit. Such differences come from the different choice of deriving the first-stage variable values (i.e., $\check{x}$ vs. $\hat{x}$). Note that the heuristic still allows CS to prove the global optimum at the root node for all the `sslp` instances.

5.5 Benchmarking Branching Methods with Simple Rounding Heuristic

In this section, we report the numerical results from the branching methods with the simple rounding heuristic described in Section 4.2.

Table 4 Computational results for `dcap`, `semi`, `sizes`, and `sslp` instances by using different branching methods in the dual decomposition with the simple rounding heuristic

Instance	Method	UB	LB	Gap (%)	Nodes Solved	Nodes Left	Heur. Time	Total Time
dcap233_200	BNP	1838.10	1833.48	0.25	976	975	3163	TO
	CS	1834.70	1834.53	OPT	33	0	123	205

continued on next page

continued from previous page

Instance	Method	UB	LB	Gap (%)	Nodes Solved	Nodes Left	Heur. Time	Total Time
	CS+DW	1834.60	1834.48	OPT	12	0	44	82
dcap233_300	BNP	1647.46	1642.75	0.28	320	317	1309	TO
	CS	1644.24	1644.18	OPT	33	0	233	1606
	CS+DW	1644.25	1644.13	OPT	22	0	155	268
dcap233_500	BNP	1741.20	1736.69	0.25	80	79	1993	TO
	CS	1737.53	1737.50	OPT	40	0	531	938
	CS+DW	1737.59	1737.46	OPT	8	0	270	430
dcap243_200	BNP	2323.93	2321.28	0.11	515	512	2274	TO
	CS	2322.68	2322.46	OPT	31	0	125	285
	CS+DW	2322.50	2322.40	OPT	14	0	58	103
dcap243_300	BNP	2560.02	2556.74	0.12	267	268	3823	TO
	CS	2559.44	2559.18	OPT	56	0	237	585
	CS+DW	2559.23	2559.01	OPT	11	0	86	154
dcap243_500	BNP	2167.92	2165.50	0.11	124	125	3672	TO
	CS	2167.37	2167.15	OPT	43	0	890	1691
	CS+DW	2167.36	2167.24	OPT	23	0	518	733
dcap332_200	BNP	1064.34	1059.10	0.49	638	637	80	TO
	CS	1060.75	1060.64	OPT	389	0	981	3059
	CS+DW	1060.77	1060.67	OPT	109	0	193	602
dcap332_300	BNP	∞	1250.94	∞	197	196	32	TO
	CS	1252.80	1252.69	OPT	275	0	1042	3391
	CS+DW	1252.79	1252.67	OPT	66	0	178	883
dcap332_500	BNP	∞	1587.09	∞	131	132	33	TO
	CS	1589.25	1587.09	0.13	187	152	2068	TO
	CS+DW	1588.85	1587.09	0.11	233	106	1229	TO
dcap342_200	BNP	1624.57	1618.09	0.39	502	501	233	TO
	CS	1619.60	1619.49	OPT	625	0	1064	4984
	CS+DW	1619.59	1619.43	OPT	97	0	190	565
dcap342_300	BNP	∞	2065.44	∞	316	315	50	TO
	CS	2067.61	2067.40	OPT	193	0	737	5404
	CS+DW	2067.55	2067.35	OPT	77	0	129	524
dcap342_500	BNP	∞	1903.02	∞	152	153	41	TO
	CS	1904.88	1903.02	0.09	131	100	1756	TO
	CS+DW	1904.76	1904.57	OPT	91	0	296	2537
semi2	BNP	147	147	OPT	1	0	1	1668
	CS	2569.97	147	94.28	1	1	1	TO
	CS+DW	147	147	OPT	1	0	1	1657
semi3	BNP	147	147	OPT	1	0	1	2875
	CS	3145.38	147	95.32	1	1	1	TO
	CS+DW	147	147	OPT	1	0	1	2907
semi4	BNP	147	147	OPT	1	0	1	4224
	CS	2682.29	147	94.51	1	1	1	TO
	CS+DW	147	147	OPT	1	0	1	4261
sizes3	BNP	226191	226168	OPT	97	0	9	42
	CS	226191	226169	OPT	109	0	8	50
	CS+DW	226191	226168	OPT	47	0	6	28
sizes5	BNP	225532	225509	OPT	777	0	152	652
	CS	225532	225509	OPT	131	0	23	142
	CS+DW	225532	225512	OPT	107	0	41	143
sizes10	BNP	224564	224319	0.10	782	767	1480	TO

continued on next page

continued from previous page

					Nodes	Nodes	Heur.	Total
Instance	Method	UB	LB	Gap (%)	Solved	Left	Time	Time
	CS	224565	224544	OPT	547	0	447	1701
	CS+DW	224564	224541	OPT	411	0	385	1885
sslp_5_25_50	BNP	-121.6	-121.6	OPT	1	0	0	1
	CS	-121.6	-121.6	OPT	4	0	1	1
	CS+DW	-121.6	-121.6	OPT	1	0	0	1
sslp_5_25_100	BNP	-127.37	-127.37	OPT	1	0	0	1
	CS	-127.37	-127.37	OPT	3	0	1	1
	CS+DW	-127.37	-127.37	OPT	1	0	0	1
sslp_10_50_50	BNP	-364.64	-364.64	OPT	1	0	0	31
	CS	-364.64	-364.64	OPT	21	0	1	113
	CS+DW	-364.64	-364.64	OPT	1	0	0	31
sslp_10_50_100	BNP	-354.19	-354.19	OPT	1	0	0	31
	CS	-354.19	-354.19	OPT	20	0	1	123
	CS+DW	-354.19	-354.19	OPT	1	0	0	32
sslp_10_50_500	BNP	-349.136	-349.136	OPT	1	0	0	653
	CS	-349.136	-349.136	OPT	20	0	7	2478
	CS+DW	-349.136	-349.136	OPT	1	0	0	651
sslp_10_50_1000	BNP	-351.711	-351.711	OPT	1	0	0	1258
	CS	-351.711	-351.711	OPT	20	0	16	2715
	CS+DW	-351.711	-351.711	OPT	1	0	0	1260
sslp_10_50_2000	BNP	-347.263	-347.263	OPT	1	0	0	3241
	CS	-347.263	-347.263	OPT	19	0	41	6712
	CS+DW	-347.263	-347.263	OPT	1	0	0	3966
sslp_15_45_5	BNP	-262.4	-262.4	OPT	1	0	0	4
	CS	-262.4	-262.4	OPT	1	0	1	5
	CS+DW	-262.4	-262.4	OPT	1	0	0	5
sslp_15_45_10	BNP	-260.5	-260.5	OPT	1	0	0	100
	CS	-260.5	-260.5	OPT	1	0	1	101
	CS+DW	-260.5	-260.5	OPT	1	0	0	102
sslp_15_45_15	BNP	-253.601	-253.601	OPT	1	0	0	268
	CS	-253.601	-253.601	OPT	1	0	1	269
	CS+DW	-253.601	-253.601	OPT	1	0	0	264

For `dcap` and `sizes` instances, the times spent on the rounding heuristics are significantly larger than those for the fixing-first heuristics. These instances have mixed-binary first-stage variables, for which each run of the simple rounding heuristic requires solving a dual decomposition of the fixed problem with a number of iterations. With the extra computational effort on running the heuristic for the `dcap` and `sizes` instances, the branch-and-bound methods solve fewer node subproblems than those with the fixing-first heuristic, which reduces the total solution times. Because of the increased time, however, the CS+DW method was not able to find the global optimum for `dcap`332_500. On the other hand, this heuristic was not effective at all with the CS method for solving `sslp` instances.

6 Summary and Directions of Future Work

We have developed a new branching method for the dual decomposition of stochastic mixed-integer programs. The method is based on the branching method proposed in [9], which we modify to improve the process of choosing the branching variables. Specifically, instead of using the average solution for branching variables, we use the optimal solutions from the restricted master problem in the dual of the dual decomposition (i.e., Dantzig-Wolfe reformulation). We show how our branching method can improve the branching decisions compared with the branching method in [9]. Another branching method is based on a branch-and-price method that takes advantage of the Dantzig-Wolfe reformulation and corresponding primal solutions. However, this method suffers from the increasing number of integer variables with the number of scenarios. In constrast, our branching method is scalable because the size of the search tree is independent of the number of scenarios. We have implemented our new method, as well as the two existing branching methods of Carøe and Schultz and of Dantzig-Wolfe, in the open source software package DSP [17]. Numerical results from our experiments show that our new method outperforms the existing methods with respect to the number of nodes solved and solution times.

Our new branching method is a first step to advance branch-and-bound methods in the dual decomposition framework, through its use of the dispersion measure of the nonanticipativity variable values against column generation solutions. The resulting branching rule provides an interesting analog to the most fractional rule in the traditional branch-and-bound method. Other branching rules such as strong branching and pseudo-cost branching may be developed for the dual decomposition setting. For example, pseudocosts for branching on $\hat{x}_i$ may be computed as

$$P_i^L = \frac{z^{\mathcal{N}_i^L} - z^{\mathcal{N}}}{\sum_{j\in\mathcal{J}} \sum_{l=0}^{k} p_j \hat{\alpha}_j^l (x_{ij}^l - \hat{x}_i)_+} \quad \text{and} \quad P_i^R = \frac{z^{\mathcal{N}_i^R} - z^{\mathcal{N}}}{\sum_{j\in\mathcal{J}} \sum_{l=0}^{k} p_j \hat{\alpha}_j^l (\hat{x}_i - x_{ij}^l)_+},$$

where the numerator computes the objective value change and the denominator represents the *directional* dispersion that the changes of the nonanticivativity variable values affected by the branching. We will develop and implement such pseudo-cost branching rules as future work. We close this section by presenting more directions of future work.

– An interesting extension of this work is to find general branching disjunctions by exploiting the structures embedded in SMIP and its solutions. For example, the thin direction branching ideas that have shown promising results in mixed-integer programs (e.g., [27,10,15]) can be investigated in the SMIP setting.
– Recent developments in cut generation approaches [37], improvements in primal heuristics [44,18], or the combination of both [34] can further accelerate the search for a global optimal solution and appear to be worth studying.
– Warm-starting the dual decomposition for each node subproblem can be achieved by reusing the cuts generated from the parent node. This will require efficient bundle management in order to avoid numerical instability, as discussed in Section 5.1.

– Extensions to nonlinear problems may be informed from the recent developments [22,23,24].
– Also warranted is the ongoing research to better understand the relationships between our work and other varied yet related theoretical and algorithmic frameworks [41,42,43,39].

References

1. Ahmed, S.: A scenario decomposition algorithm for 0–1 stochastic programs. Operations Research Letters **41**(6), 565–569 (2013)
2. Ahmed, S., Tawarmalani, M., Sahinidis, N.: A finite branch-and-bound algorithm for two-stage stochastic integer programs. Mathematical Programming **100**(2), 355–377 (2004). DOI 10.1007/s10107-003-0475-6. URL https://doi.org/10.1007/s10107-003-0475-6
3. Amor, H.B., Desrosiers, J., Frangioni, A.: On the choice of explicit stabilizing terms in column generation. Discrete Applied Mathematics **157**(6), 1167–1184 (2009)
4. Atakan, S., Sen, S.: A progressive hedging based branch-and-bound algorithm for mixed-integer stochastic programs. Computational Management Science pp. 1–40 (2018)
5. Berthold, T.: Primal heuristics for mixed integer programs. Ph.D. thesis, Technischen Universität Berlin (2006)
6. Birge, J.R., Louveaux, F.: Introduction to Stochastic Programming. Springer Science & Business Media (2011)
7. Boland, N., Christiansen, J., Dandurand, B., Eberhard, A., Linderoth, J., Luedtke, J., Oliveira, F.: Progressive hedging with a Frank-Wolfe based method for computing stochastic mixed-integer programming Lagrangian dual bounds. SIAM Journal on Optimization **28**(2), 1312–1336 (2018)
8. Boland, N., Christiansen, J., Dandurand, B., Eberhard, A., Oliveira, F.: A parallelizable augmented Lagrangian method applied to large-scale non-convex-constrained optimization problems. Mathematical Programming (2018). DOI 10.1007/s10107-018-1253-9. URL https://doi.org/10.1007/s10107-018-1253-9
9. Carøe, C.C., Schultz, R.: Dual decomposition in stochastic integer programming. Operations Research Letters **24**(1), 37–45 (1999)
10. Elhedhli, S., Naoum-Sawaya, J.: Improved branching disjunctions for branch-and-bound: An analytic center approach. European Journal of Operational Research **247**(1), 37–45 (2015)
11. Feltenmark, S., Kiwiel, K.: Dual applications of proximal bundle methods, including Lagrangian relaxation of nonconvex problems. SIAM Journal on Optimization **10**(3), 697–721 (2000)
12. Gamrath, G., Lübbecke, M.E.: Experiments with a generic Dantzig-Wolfe decomposition for integer programs. In: International Symposium on Experimental Algorithms, pp. 239–252. Springer (2010)
13. Kim, K., Anitescu, M., Zavala, V.M.: An asynchronous decomposition algorithm for security constrained unit commitment under contingency events. In: 2018 Power Systems Computation Conference (PSCC), pp. 1–8. IEEE (2018)
14. Kim, K., Botterud, A., Qiu, F.: Temporal decomposition for improved unit commitment in power system production cost modeling. IEEE Transactions on Power Systems (2018)
15. Kim, K., Mehrotra, S.: A two-stage stochastic integer programming approach to integrated staffing and scheduling with application to nurse management. Operations Research **63**(6), 1431–1451 (2015)
16. Kim, K., Petra, C.G., Zavala, V.M.: An asynchronous bundle-trust-region method for dual decomposition of stochastic mixed-integer programming. SIAM Journal on Optimization **29**(1), 318–342 (2019)
17. Kim, K., Zavala, V.M.: Algorithmic innovations and software for the dual decomposition method applied to stochastic mixed-integer programs. Mathematical Programming Computation pp. 1–42 (2017)

18. Kirst, P., Stein, O., Steuermann, P.: Deterministic upper bounds for spatial branch-and-bound methods in global minimization with nonconvex constraints. TOP **23**(2), 591–616 (2015)
19. Kiwiel, K.C.: Proximity control in bundle methods for convex nondifferentiable minimization. Mathematical programming **46**(1-3), 105–122 (1990)
20. Kiwiel, K.C.: Approximations in proximal bundle methods and decomposition of convex programs. Journal of Optimization Theory and Applications **84**(3), 529–548 (1995)
21. Kiwiel, K.C., Lemaréchal, C.: An inexact bundle variant suited to column generation. Mathematical Programming **118**(1), 177–206 (2009)
22. Li, C., Grossmann, I.: An improved L-shaped method for two-stage convex 0-1 mixed integer nonlinear stochastic programs. Computers & Chemical Engineering **112**, 165–179 (2018). DOI https://doi.org/10.1016/j.compchemeng.2018.01.017. URL http://www.sciencedirect.com/science/article/pii/S0098135418300413
23. Li, C., Grossmann, I.: A finite ϵ-convergence algorithm for two-stage stochastic convex nonlinear programs with mixed-binary first and second-stage variables. Journal of Global Optimization **75**(4), 921–947 (2019)
24. Li, C., Grossmann, I.: A generalized Benders decomposition-based branch and cut algorithm for two-stage stochastic programs with nonconvex constraints and mixed-binary first and second stage variables. Journal of Global Optimization **75**(2), 247–272 (2019)
25. Lubin, M., Martin, K., Petra, C., Sandıkçı, B.: On parallelizing dual decomposition in stochastic integer programming. Operations Research Letters **41**(3), 252–258 (2013)
26. Lulli, G., Sen, S.: A branch-and-price algorithm for multistage stochastic integer programming with application to stochastic batch-sizing problems. Management Science **50**(6), 786–796 (2004)
27. Mahajan, A., Ralphs, T.K.: Experiments with branching using general disjunctions. In: Operations Research and Cyber-Infrastructure, pp. 101–118. Springer (2009)
28. McMullen, P.: The maximum number of faces of a convex polytope. Mathematika **XVII**, 179–184 (1970)
29. McMullen, P., Shephard, G.: Convex polytopes and the upper bound conjecture. Cambridge University Press (1971)
30. Munguía, L.M., Oxberry, G., Rajan, D.: PIPS-SBB: A parallel distributed-memory branch-and-bound algorithm for stochastic mixed-integer programs. In: Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International, pp. 730–739. IEEE (2016)
31. Nemhauser, G., Wolsey, L.: Integer and Combinatorial Optimization. Wiley-Interscience, New York (1988)
32. Ntaimo, L.: Decomposition algorithms for stochastic combinatorial optimization: Computational experiments and extensions. Ph.D. thesis, The University of Arizona (2004)
33. Ntaimo, L., Sen, S.: The million-variable "march" for stochastic combinatorial optimization. Journal of Global Optimization **32**(3), 385–400 (2005). DOI 10.1007/s10898-004-5910-6. URL https://doi.org/10.1007/s10898-004-5910-6
34. Ogbe, E., Li, X.: A new cross decomposition method for stochastic mixed-integer linear programming. European Journal of Operational Research **256**(2), 487–499 (2017)
35. de Oliveira, W., Sagastizábal, C.: Bundle methods in the XXIst century: A bird's-eye view. Pesquisa Operacional **34**, 647–670 (2014)
36. Oliveira, W., Sagastizábal, C., Scheimberg, S.: Inexact bundle methods for two-stage stochastic programming. SIAM Journal on Optimization **21**(2), 517–544 (2011)
37. Qi, Y., Sen, S.: The ancestral Benders' cutting plane algorithm with multi-term disjunctions for mixed-integer recourse decisions in stochastic programming. Mathematical Programming **161**(1), 193–235 (2017)
38. Ralphs, T.K., Galati, M.V.: Decomposition in integer linear programming. In: Integer Programming, pp. 73–126. CRC Press (2005)
39. Romeijnders, W., Schultz, R., van der Vlerk, M., Klein Haneveld, W.: A convex approximation for two-stage mixed-integer recourse models with a uniform error bound. SIAM Journal on Optimization **26**(1), 426–447 (2016)
40. Ryan, K., Rajan, D., Ahmed, S.: Scenario decomposition for 0-1 stochastic programs: Improvements and asynchronous implementation. In: Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International, pp. 722–729. IEEE (2016)

41. Sherali, H., Zhu, X.: On solving discrete two-stage stochastic programs having mixed-integer first- and second-stage variables. Mathematical Programming **108**(2), 597–616 (2006)
42. Still, C., Westerlund, T.: Solving convex MINLP optimization problems using a sequential cutting plane algorithm. Computational Optimization and Applications **34**(1), 63–83 (2006)
43. Tavaslioglu, O., Prokopyev, O., Schaefer, A.: Solving stochastic and bilevel mixed-integer programs via a generalized value function. Operations Research **67**(6), 1659–1677 (2019)
44. Trespalacios, F., Grossmann, I.E.: Lagrangean relaxation of the hull-reformulation of linear generalized disjunctive programs and its use in disjunctive branch and bound. European Journal of Operational Research **253**(2), 314–327 (2016)
45. Xu, Y., Ralphs, T.K., Ladányi, L., Saltzman, M.J.: Alps: A framework for implementing parallel tree search algorithms. In: The next wave in computing, optimization, and decision technologies, pp. 319–334. Springer (2005)

Appendix

We report the numerical results from the different branching methods for solving `smkp` instances. This set of instances is challenging for the dual decomposition because the instances have a larger number of constraints and variables in the first stage than those in the second stage. This makes each iteration of the dual decomposition take a significant amount of time.

To circumvent the issue, we set a 300-second time limit for each scenario subproblem solution and use a (possibly suboptimal) feasible solution to generate the inequalities (5b) for the Lagrangian dual problem. We emphasize that the cuts generated at a suboptimal feasible solution are valid. Moreover, we use 4-hour time limit for each instance.

We report the numerical results without any heuristic in Table 6. The BNP and CS+DW methods found the global optimal solutions at the root node for all the instances, because the primal solutions $(\hat{x}_j, \hat{y}_j)$ obtained from the root node subproblem were integer feasible. Note that the differences in total solution time were due mainly to the wall-clock time limit of 300 seconds for subproblem solutions. For the given time limit, CPLEX may return slightly different solutions even for the same problem. On the other hand, the CS method was not able to find any feasible solution (and thus an upper bound) for all the instances. However, we found that the lower bounds obtained by CS method are the same as the optimal objective values for all the instances. The reason is that the average point $\bar{x}$ computed by the CS method did not represent valid primal solutions for the instances, which required solving more node subproblems for finding a primal feasible solution. This observation is consistent with results with the CS method for `sslp` instances, as shown in Table 2.

Table 6 reports the numerical results from using the CS method with the fixing-first heuristic. Note that we do not report the results for the other methods because the global optimal solutions were already obtained at the root node without any heuristic. With the heuristic, the CS method found feasible solutions for 12 instances, of which 8 instances were optimal. However, the CS method still failed to find a feasible solution for the other 8 instances within the 4-hour time limit.

Since `smkp` instances have binary variables only, the simple rounding heuristic does nothing but fixing the average point for checking the feasibility. As a result, the numerical results are same as those without any heuristic and thus not reported.

Table 5 Computational results for `smkp` instances by using different branching methods

Instance	Method	UB	LB	Gap (%)	Nodes Solved	Nodes Left	Total Time
smkp1	BNP	9339.15	9339.15	OPT	1	0	2484
	CS	∞	9339.15	∞	2	2	TO
	CS+DW	9339.15	9339.15	OPT	1	0	2485
smkp2	BNP	9001.3	9001.3	OPT	1	0	2340
	CS	∞	9001.3	∞	2	2	TO
	CS+DW	9001.3	9001.3	OPT	1	0	2339
smkp3	BNP	8560.7	8560.7	OPT	1	0	3356
	CS	∞	8560.7	∞	2	2	TO
	CS+DW	8560.7	8560.7	OPT	1	0	3557
smkp4	BNP	8916.9	8916.9	OPT	1	0	5266
	CS	∞	8916.9	∞	2	2	TO
	CS+DW	8916.9	8916.9	OPT	1	0	4664
smkp5	BNP	9423	9423	OPT	1	0	2514
	CS	∞	9423	∞	2	2	TO
	CS+DW	9423	9423	OPT	1	0	2514
smkp6	BNP	9143.2	9143.2	OPT	1	0	2669
	CS	∞	9143.15	∞	3	3	TO
	CS+DW	9143.2	9143.2	OPT	1	0	2673
smkp7	BNP	9635.5	9635.5	OPT	1	0	2371
	CS	∞	9635.5	∞	2	2	TO
	CS+DW	9635.5	9635.5	OPT	1	0	2894
smkp8	BNP	9116.8	9116.7	OPT	1	0	5703
	CS	∞	9116.8	∞	1	1	TO
	CS+DW	9116.8	9116.7	OPT	1	0	6304
smkp9	BNP	9763.75	9763.75	OPT	1	0	4502
	CS	∞	9763.75	∞	3	3	TO
	CS+DW	9763.75	9763.75	OPT	1	0	4490
smkp10	BNP	8793.1	8793.1	OPT	1	0	5317
	CS	∞	8793.1	∞	2	2	TO
	CS+DW	8793.1	8793.1	OPT	1	0	5315
smkp11	BNP	9431.3	9431.25	OPT	1	0	6063
	CS	∞	9431.25	OPT	1	1	TO
	CS+DW	9431.25	9431.25	OPT	1	0	6651
smkp12	BNP	9499.95	9499.95	OPT	1	0	7114
	CS	∞	9499.95	∞	1	1	TO
	CS+DW	9499.95	9499.95	OPT	1	0	8612
smkp13	BNP	9189.55	9189.55	OPT	1	0	6004
	CS	∞	9189.55	∞	1	1	TO
	CS+DW	9189.55	9189.55	OPT	1	0	7413
smkp14	BNP	9447.1	9447.1	OPT	1	0	9759
	CS	∞	9447.1	∞	1	1	TO
	CS+DW	9447.1	9447.1	OPT	1	0	8759
smkp15	BNP	9614.8	9614.75	OPT	1	0	9008
	CS	∞	9614.8	∞	1	1	TO
	CS+DW	9614.8	9614.8	OPT	1	0	8107
smkp16	BNP	9072.85	9072.85	OPT	1	0	10687
	CS	∞	9072.85	∞	1	1	TO
	CS+DW	9072.85	9072.85	OPT	1	0	9783
smkp17	BNP	9443.7	9443.7	OPT	1	0	8445
	CS	∞	9443.6	∞	1	1	TO
	CS+DW	9443.6	9443.6	OPT	1	0	7975
smkp18	BNP	8830.7	8830.7	OPT	1	0	10088
	CS	∞	8830.7	∞	1	1	TO
	CS+DW	8830.7	8830.7	OPT	1	0	13119
smkp19	BNP	9358.3	9358.3	OPT	1	0	9181
	CS	∞	9358.3	∞	1	1	TO
	CS+DW	9358.3	9358.3	OPT	1	0	8611
smkp20	BNP	9617.85	9617.85	OPT	1	0	10000
	CS	∞	9617.8	∞	1	1	TO
	CS+DW	9617.85	9617.85	OPT	1	0	9789

Table 6 Computational results for `smkp` instances by using CS branching methods with the fixing-first heuristic

Instance	UB	LB	Gap (%)	Nodes Solved	Nodes Left	Heur. Time	Total Time
smkp1	9339.15	9339.15	OPT	1	0	1	2487
smkp2	9001.3	9001.3	OPT	1	0	1	2347
smkp3	8560.7	8560.7	OPT	1	0	1	3057
smkp4	8916.9	8916.9	OPT	1	0	1	4971
smkp5	9423	9423	OPT	1	0	1	2811
smkp6	9143.2	9143.2	OPT	5	0	1	13675
smkp7	9635.5	9635.5	OPT	2	0	1	4912
smkp8	9162.7	9116.75	0.50	1	1	1	TO
smkp9	9763.75	9763.75	OPT	2	0	1	7202
smkp10	∞	8793.1	∞	2	2	1	TO
smkp11	9484.15	9431.3	0.50	1	0	1	TO
smkp12	9502.40	9499.95	0.02	2	1	1	TO
smkp13	∞	9189.55	∞	1	1	1	TO
smkp14	∞	9446.95	∞	1	1	1	TO
smkp15	∞	9614.65	∞	1	1	1	TO
smkp16	9271.85	9072.85	2.14	1	1	1	TO
smkp17	∞	9443.6	∞	1	1	1	TO
smkp18	∞	8830.7	∞	1	1	1	TO
smkp19	∞	9358.3	∞	1	1	1	TO
smkp20	∞	9617.35	∞	1	1	1	TO