

The Ancestral Benders' cutting plane algorithm with multi-term disjunctions for mixed-integer recourse decisions in stochastic programming

Yunwei Qi¹ · Suvrajeet Sen²

Received: 3 September 2013 / Accepted: 16 March 2016 / Published online: 23 April 2016
© Springer-Verlag Berlin Heidelberg and Mathematical Optimization Society 2016

Abstract This paper focuses on solving two-stage stochastic mixed integer programs (SMIPs) with general mixed integer decision variables in both stages. We develop a decomposition algorithm in which the first-stage approximation is solved by a branch-and-bound algorithm with its nodes inheriting Benders' cuts that are valid for their ancestor nodes. In addition, we develop two closely related convexification schemes which use multi-term disjunctive cuts to obtain approximations of the second-stage mixed-integer programs. We prove that the proposed methods are finitely convergent. One of the main advantages of our decomposition scheme is that we use a Benders-based branch-and-cut approach in which linear programming approximations are strengthened sequentially. Moreover as in many decomposition schemes, these subproblems can be solved in parallel. We also illustrate these algorithms using several variants of an SMIP example from the literature, as well as a new set of test problems, which we refer to as Stochastic Server Location and Sizing. Finally, we present our computational experience with previously known examples as well as the new collection of SMIP instances. Our experiments reveal that our algorithm is able to produce provably optimal solutions (within an hour of CPU time) even in instances for which a highly reliable commercial MIP solver is unable to provide an optimal solution within an hour of CPU time.

This work is supported in part by NSF-CMMI Grant 1100383 and AFOSR Grant FA9550-13-1-0015.

✉ Suvrajeet Sen
s.sen@usc.edu

Yunwei Qi
qi.47@osu.edu

¹ Integrated Systems Engineering, The Ohio State University, Columbus, OH 43210, USA

² Epstein Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA 90089-0193, USA

Keywords Two-stage stochastic mixed-integer programs · Cutting plane tree algorithm · Multi-term disjunctive cut · Benders' decomposition

Mathematics Subject Classification 90C15 · 90C10 · 90C06

1 Introduction

Stochastic mixed-integer programs (SMIPs) have long been recognized as an important class of models for many practical operational problems (see e.g. [26]). However, algorithmic advances to solve SMIP models have lagged behind other forms of stochastic programs. In addition to the standard difficulties associated with stochastic linear programming (e.g. designing scalable ways to approximate the expected recourse/value function), SMIP formulations with mixed-integer recourse decisions in the second stage encounter value functions that are possibly non-convex and discontinuous. Early work by Carøe and Tind [4] presented a decomposition algorithm based on mixed integer programming (MIP) duality, and while it is conceptually applicable to SMIPs with general mixed-integer recourse decisions, the algorithm is not easily realizable because it requires calculations involving exact MIP value functions. Such value functions are not only difficult to construct for the second stage, but also lead to discontinuous and non-convex first-stage approximations in general. Other decomposition algorithms, based on scenario decomposition, were proposed in [5, 14]. These algorithms essentially view the SMIP problem as a very large scale MIP using a deterministic equivalent formulation. The decomposition principles used by the above methods are dual to each other (price and resource directive decomposition respectively) and may be recommended for instances in which special structures associated with scenario subproblems can be exploited (as in unit-commitment models, lot-sizing models etc.). However, when the number of scenarios is very large, and special structures are either absent or difficult to exploit, such scenario decomposition methods are not particularly effective.

Subsequent to the work of Carøe and Tind [4], most authors addressed some sub-class of the two-stage SMIP problems. For instance, the global optimization algorithm of Ahmed et al. [1] assumed fixed tenders in the two-stage model. Others have addressed alternative sub-classes which either focus on mixed-binary recourse decisions, or pure integer recourse decisions. Decomposition-based cutting plane algorithms using disjunctive cuts (e.g. [15, 21, 24]), using Gomory cuts (e.g. [10, 28]), novel branch-and-bound methods [9], decomposition based branch-and-cut [23], as well as primal approaches using certain IP value function characterizations [12, 25] now offer a range of algorithms for alternative model characteristics. As a consequence of the sharper focus, there has been significant progress with SMIP algorithms for specialized cases of SMIP. In contrast to some of the earlier multi-stage SMIP algorithms, these methods are based on time-staged decomposition, very much in the spirit of Benders' decomposition. We refer to surveys by Louveaux and Schultz [13] and Sen [20] for discussions related to these and other advances.

In this paper, we are interested in designing time-staged decomposition algorithms for solving two-stage SMIPs in which mixed-integer decisions appear in both stages. In other words, we return to the class of models addressed in [4]. Fortunately, due

to significant algorithmic advances in the interim, we are able to draw upon new approximations which not only ensure finite convergence, but also avoid intractable operations in each iteration. The first thought which comes to mind for addressing problems with general integer variables is to replace the general integers by their binary expansion, thus, converting the general SMIP problem to one with mixed-binary variables. However, such transformations have already been discredited in the deterministic MIP literature where Owen and Mehrotra [18] have shown, mathematically and computationally, that branch-and-bound methods examine many more nodes for the transformed problem, than for the original instance with general integer variables. Consequently, the issue of devising decomposition algorithms using branch and cut methods for SMIP remains an open question which we address in this paper. Perhaps, the most critical result that helps us resolve this open question is the recent constructive characterization due to Chen et al. [7] that the feasible set of a mixed-integer linear program can be convexified using a hierarchy of multi-term disjunctions. This result helps us design the Ancestral Benders' Cutting Plane (ABC) algorithm presented in this paper.

In order to state the problem, let $A \in \mathbb{R}^{m_1 \times n_1}$, $b \in \mathbb{R}^{m_1}$, and $X_L = \{x \in \mathbb{R}^{n_1} : Ax \leq b\}$. Then consider the SMIP as stated below.

$$\min_{x \in X \cap Q_1} c^T x + \mathbf{E} [f(x, \tilde{\omega})] \tag{1}$$

where

$$X = \{x \in X_L \mid x_i \text{ is integer, } \forall i \in I_2 \subseteq I_1 = \{1, \dots, n_1\}\} \subseteq \mathbb{R}^{n_1},$$

and $Q_1 = \{x \mid l_1 \leq x \leq u_1\}$.

Also $\tilde{\omega}$ denotes a discrete random variable, and for each scenario (realization) ω of $\tilde{\omega}$, we define the recourse function by

$$\begin{aligned} f(x, \omega) = \min & \quad g(\omega)^T y \\ \text{s.t.} & \quad W(\omega)y \geq r(\omega) - T(\omega)x \\ & \quad y \in Y \cap Q_2, \end{aligned} \tag{2}$$

where,

$$Y = \{y \mid y_j \text{ is integer, } \forall j \in J_2 \subseteq J_1 = \{1, \dots, n_2\}\} \subseteq \mathbb{R}^{n_2},$$

and $Q_2 = \{y \mid l_2 \leq y \leq u_2\}$.

For subproblem (2), we let the decision variable $y \in \mathbb{R}^{n_2}$, objective coefficient $g(\omega) \in \mathbb{R}^{n_2}$, constraint matrix $W(\omega) \in \mathbb{R}^{m_2 \times n_2}$, $r(\omega) \in \mathbb{R}^{m_2}$ and $T(\omega) \in \mathbb{R}^{m_2 \times n_1}$. In our notation, when $I_2(J_2)$ is a strict subset of $I_1(J_1)$, then stage 1 (stage 2) has both continuous as well as integer variables. We assume the random variable $\tilde{\omega}$ is discrete with each scenario ω having a non-zero probability $p(\omega)$, for all $\omega \in \Omega$.

The algorithms that we propose in this paper impose the following assumptions on the model.

- A1** Both X and Y are assumed to be non-empty mixed-integer sets, and the integer variables in both stages are bounded.
- A2** The random variable $\tilde{\omega}$ in the problem is discrete, with a finite number of scenarios, $\omega \in \Omega$, each with an associated non-zero probability of occurrence $p(\omega)$, $\forall \omega \in \Omega$.
- A3** For any $x \in X_L \cap Q_1$, the set defined by $\{y \mid W(\omega)y \geq r(\omega) - T(\omega)x, y \in Y \cap Q_2\}$ is feasible for all $\omega \in \Omega$.

Due to assumption **A2**, (1) can be rewritten as:

$$\min_{x \in X \cap Q_1} c^T x + \sum_{\omega \in \Omega} p(\omega) f(x, \omega). \tag{3}$$

For the rest of the paper, we begin by first introducing the overall architecture in which a branch-and-bound (B&B) algorithm in the first-stage controls how approximations are created and passed from one generation of nodes to another. Subsequently, we present two closely related convexification schemes for the second stage, one based on the cutting plane tree method [7], and the other referred to as a B&B-based convexification method. Both of these methods can be used to *approximate* the second-stage recourse function $f(x, \omega)$. Finally, we illustrate our approach by using several variants of an SMIP example from the literature. In addition we present preliminary evidence that the decomposition framework promises to be more scalable than solving a deterministic equivalent problem (3) using a standard commercial MIP solver. We also present a new class of test instances which we refer to as Stochastic Server Location and Sizing. Computational experience with these instances demonstrates that under some restrictions on the growth of first-stage decisions, the algorithm provides a realistic approach to solving SMIP models with many scenarios. Overall, our framework provides the most comprehensive time-staged decomposition approach to date, allowing randomness in all data elements, while also allowing general mixed-integer variables as decision variables in both stages.

2 The Ancestral Benders' cutting plane algorithm

If we denote decision variables y under scenario ω as $y(\omega)$, the deterministic equivalent formulation (DEF) for (3) is

$$\min_{x, y(\omega)} c^T x + \sum_{\omega \in \Omega} p(\omega) g(\omega)^T y(\omega) \tag{4a}$$

$$\text{s.t. } T(\omega)x + W(\omega)y(\omega) \geq r(\omega), \quad \forall \omega \in \Omega \tag{4b}$$

$$x \in X \cap Q_1, y(\omega) \in Y \cap Q_2, \quad \forall \omega \in \Omega. \tag{4c}$$

Figure 1 shows the basic building blocks of the **ABC** algorithm.

It uses a branch-and-bound process to search for first-stage decisions while carrying out a convexification process in the second stage (see the box on the bottom right hand side), followed by a successive approximation scheme to update approximations of

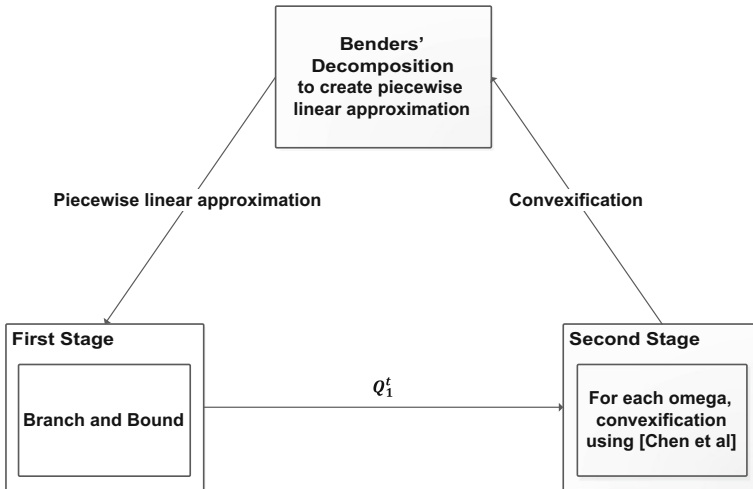


Fig. 1 Building blocks for the ABC algorithm

the expected recourse function $E[f(x, \tilde{\omega})]$ in the top box of Fig. 1. These ingredients provide a fully integrated algorithm which ties together all the pieces in a manner that is not only computationally realistic, but is also provably convergent. A brief description of the elements in Fig. 1 follows.

- (a) The B&B search in the first stage divides the range of first-stage integer variables into a partition consisting of disjoint subsets $\{Q_1^t\}$ covering the entire set Q_1 . The partition is refined using a breadth-first search strategy.
- (b) For any subset Q_1^t of the first-stage decisions (x), we create polyhedral approximations of

$$\mathcal{Z}(t, \omega) := \{(x, y(\omega)) \mid T(\omega)x + W(\omega)y(\omega) \geq r(\omega), x \in X \cap Q_1^t, y(\omega) \in Y \cap Q_2\}. \quad (5)$$

The existence and construction of the polyhedral set follows the work of Chen et al. [6] on using multi-term disjunctions.

- (c) Using Benders' decomposition for each box Q_1^t , we then create lower bounding approximations. Using these bounds, one can proceed to partition the most promising node, and continue with the branch-and-bound search.

Clearly, the above ideas are fairly straightforward. As with most stochastic programming algorithms however, the effectiveness of any scheme depends on the ease with which approximations are updated and optimized, sequentially. In addition, implementing these processes requires us to assemble a variety of *concepts*, including disjunctive programming, L-shaped decomposition, and appropriate data structures. In this section, we provide a summary of these aspects of the ABC algorithm. In the following section, we discuss multi-term disjunctions based on either the cutting plane tree (CPT) algorithm [6] or a branch-and-bound algorithm.

We use index k to denote an iteration. As in other Benders' type algorithms, we let x^k denote the first stage decisions in iteration k . Let $t(k)$ denote a node of the first-stage B&B tree such that $x^k \in Q_1^{t(k)}$. A B&B tree for the first stage provides a partition $\{Q_1^t\}$ of the box constraints Q_1 . In addition to node t 's bounds Q_1^t , $\mathcal{Z}(t, \omega)$ covers all constraints for variables x and $y(\omega)$. Next, we introduce a polyhedral approximation $\mathcal{Z}_L(t, \omega)$ such that

$$\mathcal{Z}_L(t, \omega) \supseteq \text{conv}\{\mathcal{Z}(t, \omega)\}. \tag{6}$$

This approximation $\mathcal{Z}_L(t, \omega)$, will be generated using a convexification procedure due to Chen et al. [6]. Since the approximations are in the space of variables $(x, y(\omega))$, the polyhedron $\mathcal{Z}_L(t, \omega)$ includes valid inequalities appended to the linear inequalities already in (5). That is,

$$\mathcal{Z}_L(t, \omega) := \left\{ (x, y(\omega)) \mid T(\omega)x + W(\omega)y(\omega) \geq r(\omega), x \in X_L \cap Q_1^t, y(\omega) \in Q_2, \right. \\ \left. \Pi_1(t, \omega)x + \Pi_2(t, \omega)y \geq \Pi_0(t, \omega) \right\}, \tag{7}$$

where the additional constraints in (7) represent the valid inequalities. Note that because of assumption **A3**, the projection of this set for any $x \in Q_1$ is non-empty. It is important to point out that the coefficients $\Pi_1(t, \omega)$, and the first stage decision x^k will be used to create right-hand side parameters of second stage subproblems. As a result, such inequalities are referred to as *parametric cuts*, and are useful for ‘‘warm-starting’’ approximations as Q_1 is successively refined in the B&B process.

Since the inequalities in (7) are valid for the entire set Q_1^t , they can be re-used for any of its subsets. At iteration k , we can define a lower bounding approximation of the second-stage objective by an approximation $f_L^{k,t}(x, \omega)$ as follows

$$f_L^{k,t}(x, \omega) := \min \left\{ g(\omega)^\top y(\omega) : (x, y(\omega)) \in \mathcal{Z}_L^k(t, \omega) \right\}. \tag{8}$$

where $\mathcal{Z}_L^k(t, \omega)$ includes only a subset of the inequalities of (7) revealed in the first k iterations.

$$\mathcal{Z}_L^k(t, \omega) = \left\{ (x, y(\omega)) \mid T(\omega)x + W(\omega)y(\omega) \geq r(\omega), x \in X_L \cap Q_1^t, \right. \tag{9}$$

$$\left. \Pi_1^k(t, \omega)x + \Pi_2^k(t, \omega)y(\omega) \geq \Pi_0^k(t, \omega), \right. \tag{10}$$

$$\left. l_2 \leq y(\omega) \leq u_2 \right\}. \tag{11}$$

The constraints in $\mathcal{Z}_L^k(t, \omega)$ are best treated as constraints that evolve through the iterations, that is, let $W^k(t, \omega) = \begin{bmatrix} W(\omega) \\ \Pi_2^k(t, \omega) \end{bmatrix}$, and similarly, $T^k(t, \omega) = \begin{bmatrix} T(\omega) \\ \Pi_1^k(t, \omega) \end{bmatrix}$, $r^k(t, \omega) = \begin{bmatrix} r(\omega) \\ \Pi_0^k(t, \omega) \end{bmatrix}$. Then a relaxation of the integer recourse (or value) function can be obtained as in Benders' decomposition by using the dual multipliers of the above cut-enhanced LP relaxation, for any choice of $x \in Q_1^t$.

Unlike standard Benders' decomposition (which is normally applied to the case where the second stage is a linear program), the subproblem (8) is a relaxation of

the MIP subproblem. Let $y^k(\omega)$ denote the optimal solution for (8) given $x = x^k$. Let $\Theta^k(t, \omega)$ denote the optimal dual multipliers associated with (9) and (10) and $\theta_l^k(t, \omega)$, $\theta_u^k(t, \omega)$ denote the multipliers associated with constraints (11). Then the Benders' approximation from (8) is given as

$$\eta(t, \omega) \geq \Theta^k(t, \omega)^\top r^k(t, \omega) + \theta_l^k(t, \omega)^\top l_2 - \theta_u^k(t, \omega)^\top u_2 - \Theta^k(t, \omega)^\top T^k(t, \omega)x, \quad \text{for } x \in Q_1^t \tag{12}$$

where $\eta(t, \omega)$ is the variable representing the recourse function for the pair (t, ω) . Finally, the above cuts are aggregated using all $\omega \in \Omega$ as in a Benders' cut for node t . In other words, let

$$\xi^{k,t} = \sum_{\omega \in \Omega} p(\omega) \left(\Theta^k(t, \omega)^\top r^k(t, \omega) + \theta_l^k(t, \omega)^\top l_2 - \theta_u^k(t, \omega)^\top u_2 \right),$$

$$\zeta^{k,t} = \left(\sum_{\omega \in \Omega} p(\omega) \Theta^k(t, \omega)^\top T^k(t, \omega) \right)^\top. \tag{13}$$

We define η_t as the value function cut variable for node t . Then the Benders' cut at iteration k for node t is as follows,

$$\eta_t \geq \xi^{k,t} - (\zeta^{k,t})^\top x, \quad \text{for } x \in Q_1^t. \tag{14}$$

Recall that Benders' cuts generated for Q_1^t can be used for all subsets $Q_1' \subseteq Q_1^t$. Similar inheritance also holds for the second-stage cuts in (10). In order to help record this inheritance we use index set $\mathcal{G}_x(t)$ to record Benders' cuts for node t and $\mathcal{G}_z(t, \omega)$ as the index set for the second-stage cuts. Both $\mathcal{G}_x(t)$ and $\mathcal{G}_z(t, \omega)$ start as empty sets, and when a new valid cut is derived for node t , either $\mathcal{G}_x(t)$ or $\mathcal{G}_z(t, \omega)$ are enlarged depending on whether it is a Benders' cut or a second stage (parametric) cut.

We now proceed to discuss the B&B method for the first stage. Suppose the set of active (unfathomed) nodes for the first stage is denoted as \mathcal{T}_1^k and let Q_1^t denote the bounding constraints for $t \in \mathcal{T}_1$. Then the lower bounding master problem for node t at iteration k is as follows,

$$\min \quad c^\top x + \eta_t \tag{15a}$$

$$\text{s.t.} \quad \eta_t \geq \xi_s - \zeta_s^\top x, \quad \text{for } \forall s \in \mathcal{G}_x(t) \tag{15b}$$

$$\eta_t \geq -M, \tag{15c}$$

$$x \in X_L \cap Q_1^t, \tag{15d}$$

where $-M$ is a valid lower bound on second-stage expected recourse function. In problem (15), Q_1^t will include ranges $[l_{1i}^t, u_{1i}^t]$ which denote the ranges for variable x_i at node t .

The entire algorithm starts from iteration $k = 1$ with $\mathcal{T}_1^k = \{o\}$ where the root node o has bounding constraints $Q_1^o \equiv Q_1$ and we initialize $\mathcal{G}_x(t)$ as an empty set. Let the

optimal objective value for (15) be v^t and let x^t denote an optimal first-stage solution. Thus, the global lower bound is $v \equiv \min_{t \in \mathcal{T}_1^k} \{v^t\}$. Also denote the upper bound of optimal objective value for (1) by V . At iteration k , problem (15) is solved for node $t(k-1)$ with updated (15b) from iteration $k-1$ and for our breadth-first approach, we choose the node with the least lower bound as the node to branch on. Suppose this node is \bar{t} and its optimal first-stage solution is $x^{\bar{t}}$. Having selected the node, one chooses the variable to be used for branching. We follow a common rule which uses the least relative fractional variable as follows. Let

$$\delta_i = \min \left\{ x_i^{\bar{t}} - l_{1i}^{\bar{t}}, u_{1i}^{\bar{t}} - x_i^{\bar{t}} \right\}, \tag{16}$$

$$p \in \operatorname{argmax}_{i \in I_2, u_{1i}^{\bar{t}} \neq l_{1i}^{\bar{t}}, \delta_i > 0} \left\{ \frac{\delta_i}{u_{1i}^{\bar{t}} - l_{1i}^{\bar{t}}} \right\}. \tag{17}$$

Then we select variable x_p and split its bounding constraint $[l_{1p}, u_{1p}]$ as shown in (18)

$$[l_{1p}, \lfloor x_p \rfloor] \text{ and } [\lceil x_p \rceil, u_{1p}]. \tag{18}$$

The two newly created nodes are denoted as \bar{t}^1, \bar{t}^2 . Both $\mathcal{G}_x(t)$ and $\mathcal{G}_z(t, \omega), \forall \omega \in \Omega$ are copied to initialize $\mathcal{G}_x(\bar{t}^1), \mathcal{G}_x(\bar{t}^2)$ and $\mathcal{G}_z(\bar{t}^1, \omega), \mathcal{G}_z(\bar{t}^2, \omega) \forall \omega \in \Omega$. Then the tree is updated as follows,

$$\mathcal{T}_1^k \leftarrow \left\{ \mathcal{T}_1^k \setminus \bar{t} \right\} \cup \left\{ \bar{t}^1, \bar{t}^2 \right\}. \tag{19}$$

We then solve the LP relaxation associated with the two new nodes, update the upper bound (if possible), and choose the node with the least lower bound to explore further. This process continues until a mixed-integer optimum is found.

The B&B process for the first stage is flexible, allowing us to either do a few iterations, or solving the first stage approximation to optimality. When a solution (denoted as x^k) to the master problem is found, we identify the first-stage node to which it belongs, and refer to it as $t(k)$ and its bounding constraints as $Q_1^{t(k)}$. Given x^k and $Q_1^{t(k)}$, we approximate the second-stage recourse function for all ω as described in (14). With Benders' cut indexed in $\mathcal{G}_x(t)$ updated, the master problem continues to find new integer solutions and updating nodal value functions until the node is fathomed, or the algorithm stops. A summary of the prescribed process is shown in Algorithm 1, and we refer to it as the **ABC** algorithm.

The sufficient conditions for convergence of the **ABC** algorithm are provided in the following proposition, which is consistent with sufficient conditions for finiteness of B&B methods. The precise cutting planes to be used are postponed to the following section where we will demonstrate how multi-term disjunctions ensure that these conditions are satisfied.

Proposition 1 *Let assumptions A1–A3 hold. In iteration k , let $t(k)$ denote the index identifying the subset $Q_1^{t(k)}$ for which a lower bound $v^{t(k)}$ is evaluated in iteration k . Let V^k denote an upper bound on the optimum at iteration k . Suppose that for any subset Q_1^t created during the B&B process, there exists a finite iteration $K(t)$ such*

Algorithm 1 Ancestral Benders' Cutting Plane Algorithm

Initialize: Iteration $k \leftarrow 1$, objective value upper bound $V \leftarrow \infty$, first-stage active nodes $\mathcal{T}_1^k \leftarrow \{o\}$ with $Q_1^o \equiv \{x \mid l_1 \leq x \leq u_1\}$. Initialize $\mathcal{G}_x(t) \leftarrow \emptyset$ and $\mathcal{G}_z(t, \omega) \leftarrow \emptyset$ for $\omega \in \Omega$. Let ϵ denote the stopping tolerance and $(x^*, y^*(\omega))$ for $\omega \in \Omega$ denote the incumbent solutions. Solve problem (15) at node o and get its objective value v^o .

while true do

while true do

Update the global lower bound:

$$v \leftarrow \min_{t \in \mathcal{T}_1^k} \{v^t\}.$$

Denote the node that has global lower bound as \bar{t}

if $x_{\bar{t}}^{\bar{t}}$ is integer for $\forall i \in I_2$ **then** $t(k) \leftarrow \bar{t}$, **break**.

end if

Choose variables to split and add two new child nodes \bar{t}^1, \bar{t}^2 of node \bar{t} as in (18) and (19).
 $\mathcal{G}_x(\bar{t}^1) \leftarrow \mathcal{G}_x(\bar{t}), \mathcal{G}_x(\bar{t}^2) \leftarrow \mathcal{G}_x(\bar{t})$.
 $\mathcal{G}_z(\bar{t}^1, \omega) \leftarrow \mathcal{G}_z(\bar{t}, \omega), \mathcal{G}_z(\bar{t}^2, \omega) \leftarrow \mathcal{G}_z(\bar{t}, \omega)$.

Solve problem (15) for both new nodes and obtain $v^{\bar{t}^1}$ and $v^{\bar{t}^2}$. If a new node is infeasible, it is removed from \mathcal{T}_1^k .

end while

for all $\omega \in \Omega$ **do**

Use $x^k, t(k)$ to construct $\mathcal{Z}_L^k(t(k), \omega)$ and update $\mathcal{G}_z(t(k), \omega)$.

Solve (8), get $y^k(\omega)$ and derive (12).

end for

Derive (14) and update $\mathcal{G}_x(t)$ to include the new cut.

if $V - v \leq \epsilon$ **then return** .

end if

Update V and incumbent $(x^*, y^*(\omega))$ if $y^k(\omega)$ satisfies mixed-integer restrictions for all $\omega \in \Omega$. Fathom nodes for which $v^t \geq V + \epsilon$; that is, update $\mathcal{T}_1^k: \mathcal{T}_1^k \leftarrow \mathcal{T}_1^k \setminus \{t \mid t \in \mathcal{T}_1^k, v^t \geq V + \epsilon\}$.

Update $v^{t(k)}$ by re-solving problem (15) for node $t(k)$ with updated $\mathcal{G}_x(t)$.

$k \leftarrow k + 1$

end while

that for $k \geq K(t)$ with $t(k) = t$, we either have $v^t \geq V^k$ or $f_L^{k,t}(x^k, \omega) = f(x^k, \omega)$ for all $\omega \in \Omega$. Then, the B&B procedure produces an optimal solution x^* , as well as its optimal value V^* in finitely many iterations.

Proof Let $K = K(t)$ be an iteration such that $t(K) = t$ and either the lower bound for node t matches or exceeds the global upper bound (i.e. $v^t := c^\top x^K + E[f_L^{K,t}(x^K, \omega)] \geq V^K \geq V^*$), or the lower bound for node t matches the true second-stage cost at node t (i.e. $f_L^{K,t}(x^K, \omega) = f(x^K, \omega)$) for all outcomes $\omega \in \Omega$. In case the former holds true, node t is deleted from further consideration, while if the latter holds true, there are two possibilities: either x^K satisfies the mixed-integer requirements or not. In case of the former, the B&B process deletes the node Q_1^t , and possibly updates the upper bound using the value V^K . On the other hand (i.e. x^K does not satisfy the MIP requirements) the node Q_1^t is replaced by two other descendant nodes in the first stage. Thus the only case for which the number of B&B nodes increases in the first stage is when the node Q_1^t is replaced by two descendant nodes. Due to the finiteness of bounds defining Q_1 , this subdivision can happen only finitely many times. Therefore, after finitely many iterations, the nodal solution x^K cannot violate the mixed-integer restrictions. Hence one of the nodes of the B&B tree

must reveal a mixed-integer solution x^K which becomes the incumbent and its optimal value V^K provides the value V^* in finitely many steps.

3 Approximations using multi-term disjunctions

This section presents two approaches for approximating the second-stage recourse function. Both methods convexify the set of feasible solutions of the second stage. However, one of these is based on a pure cutting plane approach, while the other performs a convexification prompted by a B&B tree.

3.1 Convexification using pure cutting planes

Until recently, the question of computing the convex hull of mixed-integer points which satisfy a system of rational linear inequalities was open. For example, Owen and Mehrotra [17] and Balas [2] have shown that for general MIP, traditional two-term disjunctive cuts [3] are inadequate to the task of satisfying the conditions of Proposition 1. The above question was resolved by Del Pia and Weismantel [8], although their characterization is not based on an algorithmic construction. For the case of bounded mixed-integer sets, Chen et al. [6] present two algorithms for constructing polyhedral approximations which provide the same solution as the original MIP in finitely many iterations. One of the algorithms, the convex hull tree algorithm, essentially provides a proof technique, and is not recommended as a solution methodology. The other algorithm, the cutting plane tree (CPT) algorithm, is a sequential cutting plane scheme which generates a finite sequence of cuts which, in the worst case, verifies that the polyhedral approximation provides an optimal solution which is the same as the MIP solution. This method may be summarized as one that generates cutting planes from a hierarchy of multi-term disjunctions which are recorded in the form of nodes of a search tree. So long as each visit to a node (i.e. a multi-term disjunction) generates one facet of the disjunction, and nodes can only be visited finitely many times, finiteness of the number of multi-term disjunctions implies that the CPT algorithm converges in finitely many iterations. We refer the reader to Chen et al. [6] for the details, and Chen et al. [7] for computational results for deterministic MIP problems. Another similar result has been proposed by Jörg [11], although its computability is not clear as of this writing. In any event, the method of Chen et al. [6] is able to deliver the property required by Proposition 1. Because these multi-term disjunctions are intimately tied to the CPT algorithm, we refer to the resulting cuts as ‘‘CPT cuts’’.

Suppose that at the k -th iteration of **ABC** algorithm, we have a fixed $x^k \in X_L \cap Q_1^{t(k)}$ and are given an initial approximation $f_L^{k,t(k)}(x, \omega)$ from previous iterations. We seek to approximate $f(x, \omega)$ further using x^k and $f_L^{k,t(k)}(x, \omega)$ by executing the CPT algorithm for a few iterations. Note that this approximation is only valid for $x \in X \cap Q_1^{t(k)}$. To initialize a sequence of subproblem approximations of the CPT algorithm, we start by solving (8). Let d denote the iteration counter of the CPT algorithm (for the second stage), and let $y^d(\omega)$ denote a solution with some fractional variable(s) at iteration d . Let $\mathcal{T}_2^d(\omega)$ denote an index set of the sets that constitute a

partition $\{Q_2^\tau(\omega), \tau \in \mathcal{T}_2^d(\omega)\}$ of Q_2 . For each $\tau \in \mathcal{T}_2^d(\omega)$, let $Q_2^\tau(\omega)$ denote bounding constraints for the vectors y as shown in (20),

$$Q_2^\tau(\omega) = \{y \mid l_2^\tau \leq y \leq u_2^\tau\}, \quad \forall \tau \in \mathcal{T}_2^d(\omega). \tag{20}$$

Given Y only imposes integer restrictions, the set $\mathcal{T}_2^d(\omega)$ can be used to index a disjunctive relaxation of $Y \cap Q_2$ because

$$\bigcup_{\tau \in \mathcal{T}_2^d(\omega)} Q_2^\tau(\omega) \supseteq Y \cap Q_2. \tag{21}$$

Since $y^d(\omega)$ is presumed to have some fractional values for integer variables, we separate the fractional point $(x^k, y^d(\omega))$ from the convex hull of those points $(x, y(\omega))$ which require $y(\omega)$ to be integral. Thus, we construct a disjunctive set which satisfies the following,

$$(x^k, y^d(\omega)) \notin \bigcup_{\tau \in \mathcal{T}_2^d} \{(x, y(\omega)) \mid x \in X_L \cap Q_1^{i(k)}, y(\omega) \in Q_2^\tau(\omega)\}. \tag{22}$$

We use the same rules as in the CPT algorithm [7] to construct the disjunctive set. Moreover, the hierarchy of disjunctions represented in $\mathcal{T}_2^d(\omega)$ is generated in the form of a tree structure (called cutting plane tree) and $\mathcal{T}_2^d(\omega)$ contains all nodes that do not have children nodes. The tree itself is initialized with one global node defining the constraints for Q_2 . If the solution $y^d(\omega)$ does not satisfy the mixed-integer restrictions, then the algorithm walks through the cutting plane tree to locate the deepest node (from the root node) that contains $y^d(\omega)$. Let us refer to this node as τ^d . If $\tau^d \in \mathcal{T}_2^d(\omega)$ (implying it does not have any children node), two nodes are created as its children nodes and τ^d is removed from $\mathcal{T}_2^d(\omega)$. On the other hand, if $\tau^d \notin \mathcal{T}_2^d(\omega)$, no new node is created. In this way, $\mathcal{T}_2^d(\omega)$ is updated such that conditions (21) and (22) are satisfied.

By intersecting every subset of the partition (20) with $Z_L^k(t, \omega)$, a cut generation linear program (CGLP) can then be formulated to derive multi-term disjunctive cuts as follows.

$$\max \quad \pi_0(\omega) - \pi_1(\omega)^\top x^k - \pi_2(\omega)^\top y^d(\omega) \tag{23a}$$

$$\text{s.t.} \quad \pi_1(\omega) = T^k(t, \omega)^\top \lambda_{2\tau} - A^\top \lambda_1 + \mu_1 - v_1 \quad \forall \tau \in \mathcal{T}_2^d(\omega) \tag{23b}$$

$$\pi_2(\omega) = W^k(t, \omega)^\top \lambda_{2\tau} + \mu_{2\tau} - v_{2\tau} \quad \forall \tau \in \mathcal{T}_2^d(\omega) \tag{23c}$$

$$\begin{aligned} r^k(t, \omega)^\top \lambda_{2\tau} - b^\top \lambda_1 + (l_2^\tau)^\top \mu_{2\tau} + (l_1^t)^\top \mu_1 - (u_2^\tau)^\top v_{2\tau} - (u_1^t)^\top v_1 \\ \geq \pi_0(\omega) \quad \tau \in \mathcal{T}_2^d(\omega) \end{aligned} \tag{23d}$$

$$-\mathbf{1} \leq \pi_1(\omega) \leq \mathbf{1}, \quad -\mathbf{1} \leq \pi_2(\omega) \leq \mathbf{1}, \tag{23e}$$

$$-1 \leq \pi_0(\omega) \leq 1 \tag{23f}$$

$$\lambda_1, \lambda_{2\tau} \geq 0, v_1, v_{2\tau} \geq 0, \mu_1, \mu_{2\tau} \geq 0, \quad \text{for } \forall \tau \in \mathcal{T}_2^d(\omega). \tag{23g}$$

There is significant latitude in formulating the CGLP (23) when it comes to the treatment of the first stage variables x , because the search for the optimal value of these variables is carried out via a B&B process in stage 1. As with the use of disjunctive programming to explain intersection cuts, one may ignore one or more first stage constraints by using elements of λ_1 to be 0. In formulating (23), we have opted to use $\lambda_{1t} = \lambda_1, \mu_{1t} = \mu_1$, and $v_{1t} = v_1$ for all t . This reduces the size of the CGLP, while recognizing that the first stage variables x must satisfy $x \in X_L \cap Q_1^t$. Note that such a restriction does not ensure access to facets of the convex hull of $X \cap Q_1^t$. However, such facets are not necessary for convergence because the B&B process of the first stage is used to ensure convergence. Note also that in generating the cut coefficients of the second stage, we do index the multipliers $\lambda_{2\tau}, \mu_{2,\tau}, v_{2,\tau}$ by the subsets in the disjunction. The critical aspect of these cuts is that they provide the correct *second stage value function* as required by Proposition 1. This result will be shown in Proposition 3. In any event, the solution to the above CGLP provides the following valid inequality.

$$\pi_1(\omega)^\top x + \pi_2(\omega)^\top y(\omega) \geq \pi_0(\omega). \tag{24}$$

Note that (24) is associated with scenario ω , and we derive separate cuts for each $\omega \in \Omega$. Since (22) includes inequalities that are valid for $X \cap Q_1^t$, this CGLP combines the convexification in the space $(x, y(\omega))$ where $x \in Q_1^t$. As a result, the CGLP suggested in this paper is larger than that used for the original D^2 algorithm [21]. However, the ABC algorithm does not require the convexification using the epi-reverse polar used in [21]. In this sense, the cuts used in the ABC algorithm are different from those in the D^2 algorithm.

Proposition 2 *Cutting plane (24) is valid for feasible set $\{(x, y(\omega)) \mid T^k(t, \omega)x + W^k(t, \omega)y(\omega) \geq r^k(t, \omega), x \in X \cap Q_1^t, y(\omega) \in Y \cap Q_2\}$.*

Proof For any $(x, y(\omega))$ satisfying $\{(x, y(\omega)) \mid T^k(t, \omega)x + W^k(t, \omega)y(\omega) \geq r^k(t, \omega), x \in X \cap Q_1^t, y(\omega) \in Y \cap Q_2\}$, (23b, c, d, g) imply that

$$\pi_1(\omega)^\top x + \pi_2(\omega)^\top y(\omega) \tag{25a}$$

$$= \left(\lambda_{2\tau}^\top T^k(t, \omega)x - \lambda_1^\top Ax + \mu_1^\top x - v_1^\top x \right) + \left(\lambda_{2\tau}^\top W^k(t, \omega)y(\omega) + \mu_{2\tau}^\top y(\omega) - v_{2\tau}^\top y(\omega) \right) \tag{25b}$$

$$\geq r^k(t, \omega)^\top \lambda_{2\tau} - b^\top \lambda_1 + l_2^\tau{}^\top \mu_{2\tau} + l_1^\tau{}^\top \mu_1 - u_2^\tau{}^\top v_{2\tau} - u_1^\tau{}^\top v_1 \tag{25c}$$

$$\geq \pi_0(\omega), \tag{25d}$$

as required.

The cut (24) is included in the second-stage formulation, leading to a stronger approximation of the second-stage polyhedron, and consequently, a stronger approximation of the recourse function, which is denoted as $f_L^{k,t,d}(x, \omega)$. In the following, a superscript $k-$ denotes the most recent update of any particular data element (e.g. W, T, r). In general, after d iterations of the CPT algorithm for the subproblem, we have

$$f_L^{k,t,d}(x, \omega) = \min \quad g(\omega)^\top y \tag{26a}$$

$$\text{s.t. } W^{k-}(t, \omega)y \geq r^{k-}(t, \omega) - T^{k-}(t, \omega)x \tag{26b}$$

$$\Pi_2^d(\omega)y \geq \Pi_0^d(\omega) - \Pi_1^d(\omega)x \tag{26c}$$

$$y \in Q_2 \tag{26d}$$

where constraint (26b) denotes the approximation of subproblem for $x \in Q_1^t$ before starting iteration k . In addition, (26c) includes all cuts generated during this round of iterations for solving/approximating the subproblem. There is another algorithmic point to be noted here; (26) is essentially the same as (8), but the left hand side of (26a) includes three superscripts (k, t, d) while the function in (8) has two superscripts. The reason for this distinction is to emphasize that there are “outer iterations” (first stage) indexed by k , whereas, the “inner iterations” (second stage) are indexed by d . Depending on the course of the CPT algorithm, (26c) needs to be included in the CGLP to ensure convergence of the algorithm [6]. After the approximation $f_L^{k,t,d}(x, \omega)$ is obtained, the cut-enhanced LP is re-solved and new cuts are generated as long as the inner iteration generates $y^d(\omega)$ that are fractional. At any outer iteration k , we allow at most D cuts to be added for the second-stage CPT (approximation) algorithm. Once the process of solving the subproblem stops, we form a Benders' cut as in (14) and return to the master problem. As a reminder, recall that the cuts for the first stage are recorded via the index set $\mathcal{G}_x(t)$, while cuts for the second stage are indexed by elements of $\mathcal{G}_z(t, \omega)$. Since x^k is integer feasible, then $x^k \in X_L \cap Q_1^{t(k)}$. An algorithm to approximately solve subproblems for each ω , with at most D cuts added, called cutting plane tree disjunction (CPT-D), is shown as Algorithm 2.

Algorithm 2 CPT-D

Initialize $d \leftarrow 1$, assign CPT tree leaves set $\mathcal{T}_2^d(\omega) \leftarrow \{o\}$ where o is the root node with bounding constraint $Q_2^o \leftarrow Q_2$. Populate $W^{k-}(t, \omega), r^{k-}(t, \omega), T^{k-}(t, \omega)$ with valid cuts based on $\mathcal{G}_z(t, \omega)$.
while $d \leq D$ **do**
 Evaluate $f_L^{k,t,d}(x, \omega)$ and get $y^d(\omega)$.
 if $y^d(\omega)$ satisfies mixed-integer restrictions **then**, STOP and $y^k(\omega) \leftarrow y^d(\omega)$
 else
 Find the deepest node σ that contains $y^d(\omega)$ in $\mathcal{T}_2^d(\omega)$.
 if σ is a leaf node **then**,
 Split node σ and use the updated $\mathcal{T}_2^d(\omega)$ and first-stage node bounds $Q_1^{t(k)}$ to formulate and solve (23) and obtain cut (24).
 else
 No splits are needed. Use σ and leaf nodes that in the subtree of σ and $Q_1^{t(k)}$ to formulate and solve (23) and obtain cut (24).
 end if
 Update $\Pi_1^d(\omega), \Pi_2^d(\omega), \Pi_0^d(\omega)$ with the new cut, enlarge $\mathcal{G}_z(t, \omega)$ with the index of the new cut.
 end if
 $d \leftarrow d + 1$
end while

Proposition 3 *Assume that the second-stage problems are all bounded, and moreover, suppose that the cuts coefficients (24) correspond to extreme point solutions of the*

*CGLP (23). Then for fixed $t(k)$ there exists a finite integer $D < \infty$, such that algorithm **CPT-D** finds mixed integer optimal solutions for all subproblems (2) indexed by ω ; that is, with $x = x^k$ we have $f_L^{k,t,D}(x^k, \omega) = f(x^k, \omega)$ for $\forall \omega$.*

Proof When D is large enough, algorithm **CPT-D** is the same as the CPT algorithm to solve each subproblem. Due to the finiteness of the CPT algorithm proved in [6], all we need to prove is that for fixed $T_2^d(\omega)$ and $Q_1^{t(k)}$, only finitely many constraints can be generated from the CGLP (23) as $(x^k, y^d(\omega))$ changes. To see this, note that the constraints in (23) depend on the disjunction that is violated, but not the specific vectors $(x^k, y^d(\omega))$. With fixed $T_2^d(\omega)$, there are only finitely many disjunctions that can be violated. Thus, the number of extreme points of (23) is finite. Since with the generation of each cut, there is one less extreme point of the CGLP that can be generated. So in the worst case, all extreme points are generated which implies the finiteness of the number of potential cuts. This completes the proof.

Chen et al. [7] shows that a formulation which minimizes the 1-norm provides better numerical stability than the CGLP in (23). Therefore, our computational experiments use the following as the CGLP to generate disjunctive cuts.

$$\min \sum_{i \in I} |\pi_{1i}(\omega)| + \sum_{j \in J} |\pi_{2j}(\omega)| \tag{27a}$$

$$\text{s.t. } \pi_1(\omega) = T^k(t, \omega)^\top \lambda_{2\tau} - A^\top \lambda_1 + \mu_1 - v_1 \quad \forall \tau \in T_2^d(\omega) \tag{27b}$$

$$\pi_2(\omega) = W^k(t, \omega)^\top \lambda_{2\tau} + \mu_{2\tau} - v_{2\tau} \quad \forall \tau \in T_2^d(\omega) \tag{27c}$$

$$\begin{aligned} r^k(t, \omega)^\top \lambda_{2\tau} - b^\top \lambda_1 + (I_2^t)^\top \mu_{2\tau} + (I_1^t)^\top \mu_1 - (u_2^t)^\top v_{2\tau} - (u_1^t)^\top v_1 \\ \geq \pi_1(\omega)^\top x^k + \pi_2(\omega)^\top y^d(\omega) + 1 \quad \tau \in T_2^d(\omega) \end{aligned} \tag{27d}$$

$$\lambda_1, \lambda_{2\tau} \geq 0, v_1, v_{2\tau} \geq 0, \mu_1, \mu_{2\tau} \geq 0, \quad \text{for } \forall \tau \in T_2^d(\omega). \tag{27e}$$

If one wishes to implement (27) and still ensure finiteness of the algorithm, one has to ensure that the cuts from (27) can be mapped to an extreme point of (23). This can be accomplished in several algorithmic ways. One should observe that any cut obtained by using (27) has an equivalent cut in (23). If this mapping reveals an extreme point of (23), then, finiteness is ensured. However, if the solution from (27) is not equivalent to an extreme point of (23), then, one should identify the lowest dimensional face of (23) whose interior contains the equivalent cut coefficients. Then one could optimize (23) by restricting the LP solution to belong to that face. Since this process would identify an extreme point of (23), finiteness of the cut generation scheme would be ensured.

In our implementation, we gradually increase the number of cuts we generate during any outer iteration k . To do so, we initialize an integer $D \leftarrow 2$ and at each outer iteration, we update $D \leftarrow D + 2$. This style of implementation is motivated by our prior experience (e.g. [27]) that seeking very accurate objective function estimates requires much more computational resources than what can be justified in early (outer) iterations of the algorithm; in later iterations however, seeking greater accuracy tends to pay off.

3.2 Convexification implied by branch-and-bound

Most deterministic MIP algorithms combine valid inequalities in the context of branch-and-bound methods, and as a result, branch-and-cut methods form the backbone for most state-of-the-art commercial solvers. The CPT algorithm used in the previous section is a pure cutting plane method. The fact that it also utilizes a tree structure to manage the disjunctive sets inspires a way that transforms the B&B tree obtained from an MILP solver to help create a polyhedral approximation which gives the same MIP optimal value as that obtained from the B&B method. In the remainder of this section, we describe such an algorithm and prove that this approximation can also be obtained in finitely many steps. It turns out that this combination (of B&B with valid inequalities) extends the branch-and-cut approaches of Sen and Sherali [23], and Yuan and Sen [27].

Suppose that for a fixed $x^k \in X_L \cap Q_1^{t(k)}$, the subproblem (2) is either solved to optimality by a B&B method, or an approximate solution is obtained via a truncated B&B process. The latter is typically stopped when a node limit or a time limit is reached. Let the optimal/incumbent solution be denoted $y^k(\omega)$. Due to the B&B (or truncated B&B) process, it is reasonable to assume that we have a set of leaf nodes of a B&B tree. Let $\mathcal{T}_2^{\text{remain}}$ denote the remaining leaf nodes in the B&B tree and $\mathcal{T}_2^{\text{fathom}}$ denote the leaf nodes that have been fathomed. Note that these nodes depend on ω , but we have dropped that dependence to ease the notational burden. However, we do define $\mathcal{T}_2(\omega) = \mathcal{T}_2^{\text{remain}} \cup \mathcal{T}_2^{\text{fathom}}$. Suppose the constraint set used in (7) is represented in the form

$$\begin{aligned} \mathcal{Z}_L^{k-}(t, \omega) = & \left\{ (x, y(\omega)) \mid T^{k-}(t, \omega)x + W^{k-}(t, \omega)y(\omega) \right. \\ & \left. \geq r^{k-}(t, \omega), x \in X_L \cap Q_1^t, y(\omega) \in Q_2 \right\}, \end{aligned} \tag{28}$$

where similar to (26), a superscript $k-$ denotes the most recent update of any data element (e.g. W, r) before iteration k . We also define

$$\mathcal{Z}_D^k(t, \omega) = \bigcup_{\tau \in \mathcal{T}_2(\omega)} \left(\mathcal{Z}_L^{k-}(t, \omega) \cap \{(x, y(\omega)) \mid x \in \mathbb{R}^{n_1}, y \in Q_2^\tau\} \right). \tag{29}$$

Since $Q_2^\tau, \forall \tau \in \mathcal{T}_2(\omega)$ are disjoint from each other, $\mathcal{T}_2(\omega)$ provides a disjunctive relaxation in the space of $(x, y(\omega))$. Thus, the same form of CGLP as in (23) can be used to derive multi-term disjunctive cuts using the partition $\mathcal{T}_2(\omega)$. The rest of the algorithm is similar to **CPT-D**. There are two phases: the first phase is to use a B&B method to either solve the second-stage MIP, or obtain an approximate solution using a truncated B&B process. In either case, we have $\mathcal{T}_2(\omega)$ which is used to obtain a disjunctive approximation. The second phase starts by seeking the value $f_L^{k,t,d}(x^k, \omega)$ with $d = 1$. At iteration d , if the optimal solution of (26) is fractional (denoted $y^d(\omega)$), we formulate (23) based on $\mathcal{T}_2(\omega)$ and Q_1^t to cut off $(x^k, y^d(\omega))$. The index of the new cut is added into $\mathcal{G}_z(t, \omega)$. Then, we re-solve (26), and this process continues until $(x^k, y^d(\omega))$ belongs to $\mathcal{Z}_D^k(t, \omega)$. The method is called Branch and Bound

Algorithm 3 BB-D

Initialize iteration $d \leftarrow 1$. Initialize $\mathcal{G}_z(t, \omega) \leftarrow \emptyset$ if it does not exist,

Phase 1:

Solve subproblem: Evaluate $f^{k,t}(x, \omega)$ by using a B&B method with $x = x^k$ for D iterations and get leaf nodes set $\mathcal{T}_2(\omega)$ and solution $y^*(\omega)$.

Phase 2:

while true do

Evaluate $f_L^{k,t,d}(x, \omega)$ and get $y^d(\omega)$.

if $(x^k, y^d(\omega)) \in \mathcal{Z}_D^k(t, \omega)$ **then**, $y^k(\omega) \leftarrow y^d(\omega)$ and **return** .

else

Use $\mathcal{T}_2(\omega)$ and Q_1^t to formulate and solve (23) and obtain cut (24).

Update $\Pi_1^d(\omega)$, $\Pi_2^d(\omega)$, $\Pi_0^d(\omega)$ with the new cut and enlarge $\mathcal{G}_z(t, \omega)$ with the index of the new cut.

end if

$d \leftarrow d + 1$

end while

with Disjunction (**BB-D**) and is shown in Algorithm 3. While the form of the **BB-D** process is similar to the **CPT-D** process presented in the previous section, the inclusion of B&B, especially its truncated version, makes this decomposition approach much more realistic for practical instances of MIP in the second stage. Nevertheless, the proof of convergence derives from the same concept that one can obtain a polyhedral approximation of a disjunctive set embodied by a B&B tree. This is summarized in the following proposition.

Proposition 4 *Under the same assumptions as in Proposition 3, Algorithm 3 terminates in finitely many steps, provided D is sufficiently large so that the B&B process provides an optimal second-stage solution for any node indexed by t .*

Proof A B&B tree embodies a multi-term disjunction. Using such a disjunction to formulate the CGLP in (23) results in a polyhedral approximation which creates the convex hull of the partition provided by the B&B tree. Hence there exists a counter $K < \infty$ such that for any subset Q_1^t , we obtain the same solution as from B&B. Hence for a B&B algorithm such that there is $D < \infty$ such that $f(x^K, \omega) = f_L^{K,t,D}(x^K, \omega)$, for all $\omega \in \Omega$. Since this satisfies the requirements of Proposition 1, the result follows.

Both Algorithms 2 and 3 use multi-term disjunctive cuts which are added to the second stage MIP, and the resulting polyhedral approximation then allows us to update a piecewise linear approximation of the expected recourse function as in (14–13). The difference between the two algorithms is simply the manner in which the disjunctive sets are constructed: Algorithm 3 uses the B&B nodes to construct one disjunctive set, whereas, Algorithm 2 iteratively builds up a collection of cuts based on a sequence of disjunctions, which may vary depending on the second stage fractional points encountered during the CPT process.

4 Illustrative examples

This section illustrates the workings of the algorithm using four examples, all of which are variants of Example 1.0. This example, which originally appeared in [19], has been

used to illustrate several algorithms whose structures are specialized to binary second stage problems in [22], and to problems with fixed tenders [1]. In our presentation, we will start with the binary instance, and then, make the example progressively more difficult, allowing general integers in the second stage (Example 1.1), general integers in both stages (Example 1.2), and finally allowing randomness in the T matrix (Example 1.3). Figures illustrating the progress of the first-stage search are provided in the "Appendix".

Example 1.0

$$\min -1.5x_1 - 4x_2 + \sum_{\omega \in \Omega} p(\omega) f(x, \omega) \tag{30a}$$

$$\text{s.t. } x_1, x_2 \text{ binary} \tag{30b}$$

where

$$f(x, \omega) = \min -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \tag{31a}$$

$$\text{s.t. } \begin{bmatrix} 2y_1 + 3y_2 + 4y_3 + 5y_4 - R \\ 6y_1 + 1y_2 + 3y_3 + 2y_4 - R \end{bmatrix} \leq r(\omega) - T(\omega)x \tag{31b}$$

$$y_i \text{ binary } i = 1, \dots, 4, R \geq 0, \tag{31c}$$

$\Omega = \{\omega_1, \omega_2\}$, $p(\omega_1) = p(\omega_2) = 0.5$, $r(\omega_1) = \begin{bmatrix} 10 \\ 3 \end{bmatrix}$, $T(\omega_1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $r(\omega_2) = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$, $T(\omega_2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. We first apply the **ABC** algorithm with **CPT-D** to solve this example.

At iteration $k = 1$, the algorithm starts by solving a relaxed LP master problem. We put a very low bound on η .

$$\min -1.5x_1 - 4x_2 + \eta \tag{32a}$$

$$\text{s.t. } 0 \leq x_1, x_2 \leq 1 \tag{32b}$$

$$\eta \geq -M \tag{32c}$$

We get a solution $(x_1, x_2, \eta) = (1, 1, -M)$ with objective $v = -M - 5.5$. Here only the root node is in the *B&B* tree. With $x = (1, 1)$ and $Q_1^o = \{0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$ we solve the subproblems. We now invoke **CPT-D** (Algorithm 2) as explained next. For scenario ω_1 , (8) is solved and we get $y(\omega_1) = (0, 1, 0, 0.5, 0)$. Note that y_4 is fractional and partitions are formed for that integer variable: $\{y_4 \leq 0\} \cap Q_2$ or $\{y_4 \geq 1\} \cap Q_2$. The cut derived from CGLP (27) for $x \in Q_1^o$ is

$$-2y_2 - 2y_4 + 2R \geq -4 + 2x_2. \tag{33}$$

After including the cut, (8) is re-optimized and the solution is $y(\omega_1) = (0, 0, 0, 1, 0)$. This satisfies the integer constraints. For scenario ω_2 , (8) is solved and we get

$y(\omega_2) = (0, 1, 0, 0, 0)$. Again, this solution satisfies the integer constraints, and hence, no additional cuts are needed. Since all scenarios have integer solution, we update V : $V = -29$ and, the Benders' cut for $x \in Q_1^o$ is

$$- 16.5x_2 + \eta \geq -40. \tag{34}$$

At iteration $k = 2$, the master problem continues to be solved by a B&B method. We get a solution $(x_1, x_2, \eta) = (1, 0, -40)$ with objective $v = -41.5$. The first stage B&B tree still contains only the root node. With $x = (1, 0)$ and $Q_1^o = \{0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$ we solve the subproblems. Again we invoke **CPT-D**. For scenario ω_1 , (8) is initialized as follows:

$$f_L(x, \omega_1) = \min \quad - 16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \tag{35a}$$

$$\text{s.t.} \quad 2y_1 + 3y_2 + 4y_3 + 5y_4 - R \leq 10 - x_1 \tag{35b}$$

$$6y_1 + 1y_2 + 3y_3 + 2y_4 - R \leq 3 - x_2 \tag{35c}$$

$$- 2y_2 - 2y_4 + 2R \geq -4 + 2x_2 \tag{35d}$$

$$0 \leq y_i \leq 1 \quad i = 1, \dots, 4, R \geq 0 \tag{35e}$$

where constraint (35d) is from cut (33) generated in iteration 1. (35) is solved and we get $y(\omega_1) = (0, 1, 0, 1, 0)$. The solution satisfies integer constraints. For scenario ω_2 , (8) is solved and we get $y(\omega_2) = (0.1154, 1, 0, 0.1538, 0)$. We now choose y_1 as the variable to split. The partitions we form are: $\{y_1 \leq 0\} \cap Q_2$ or $\{y_1 \geq 1\} \cap Q_2$. The cut derived from CGLP (27) for $x \in Q_1^o$ is

$$- 4.875y_2 - 6.5y_4 + 1.625R \geq -6.5 + 1.625x_1. \tag{36}$$

After including the cut, (8) is re-optimized and the solution is

$$y(\omega_2) = (0.056, 1, 0.222, 0, 0). \tag{37}$$

Following the CPT algorithm, we locate $y(\omega_2)$ on the root node of CPT tree. There are no splits needed. We continue with the same partition: $\{y_1 \leq 0\} \cap Q_2$ or $\{y_1 \geq 1\} \cap Q_2$. The cut derived from (27) for $x \in Q_1^o$ is

$$- 2.25y_2 - 4.5y_3 + 2.25R \geq -2.25. \tag{38}$$

After including the cut, (8) is re-optimized and the new solution is $y(\omega_2) = (0.06, 0.68, 0.16, 0.24, 0)$. Again, $y(\omega_2)$ is located on the root node, and no more splits are needed. The same partition: $\{y_1 \leq 0\} \cap Q_2$ or $\{y_1 \geq 1\} \cap Q_2$ is used to formulate (27). With only $y(\omega_2)$ changed, another cut derived from (27) for $x \in Q_1^o$ and that is

$$- 2.5y_3 - 2.5y_4 + 2.5R \geq -2.5 + 2.5x_1. \tag{39}$$

After including the cut, (8) is re-optimized and the solution is $y(\omega_2) = (0.1667, 1, 0, 0, 0)$. Since $y(\omega_2)$ is located at the root node no more splits are necessary. We use the same partition: $\{y_1 \leq 0\} \cap Q_2$ or $\{y_1 \geq 1\} \cap Q_2$ to formulate (27). The cut derived from (27) for $x \in Q_1^o$ is

$$-6y_1 + 1.5R \geq 0. \tag{40}$$

After including the cut, (8) is re-optimized and the solution is $y(\omega_2) = (0, 1, 0, 0, 0)$. This solution satisfies the integer constraints. Since all scenarios have integer solutions, we update V : $V = -34.5$ and, Benders' optimality cut for $x \in Q_1^o$ is

$$-7.55x_1 - 3.8333x_2 + \eta \geq -40.55. \tag{41}$$

At iteration $k = 3$, the master problem continues to be solved by the B&B method. We get $(x_1, x_2, \eta) = (0, 0, -40)$ with objective $v = -40$. The solution is on node 1 with $Q_1^1 = \{0 \leq x_1 \leq 0, 0 \leq x_2 \leq 1\}$. Hence, $x = (0, 0)$ and Q_1^1 are treated as input parameters for **CPT-D** for each $\omega \in \Omega$. For scenario ω_1 , no cuts are needed. The solution is $y(\omega_1) = (0, 1, 0, 1, 0)$. For scenario ω_2 , one cut is needed and is shown below.

$$-3.6923y_2 - 2.4615y_3 - 3.6923y_4 + 1.2308R \geq -3.6923. \tag{42}$$

The solution is $y(\omega_2) = (0, 0, 0, 1, 0)$. We update V : $V = -37.5$ and, the Benders' optimality cut for Q_1^1 is

$$-8.3333x_2 + \eta \geq -37.5. \tag{43}$$

At iteration $k = 4$, with updated Benders' cut for node 1, the master problem continues to be solved by a B&B method. We get solution $(x_1, x_2, \eta) = (0, 0, -37.5)$ with objective $v = -37.5$. Since $V - v \leq \epsilon$, the algorithm stops. A short summary of using **ABC** algorithm with **CPT-D** to solve this problem is shown in Table 1. Each row shows the information for one iteration. Column "Num Nodes" denotes the number of active nodes in the B&B tree. "Num Cuts" means the number of multi-term disjunctive cuts generated for that scenario.

We also apply the same **ABC** algorithm but with **BB-D** to solve this example. The algorithm starts by solving a relaxed LP master problem. We put a very low bound on η . At iteration $k = 1$, we get a solution $(x_1, x_2, \eta) = (1, 1, -M)$ with objective $v = -M - 5.5$ from the B&B method. With $x = (1, 1)$ and $Q_1^o = \{0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$, we invoke **BB-D** to solve the subproblems for each $\omega \in \Omega$. For scenario ω_1 , (8) is solved by the B&B method and we get 2 nodes in T_2 with bounds $\{y_4 \leq 0\} \cap Q_2$ and $\{y_4 \geq 1\} \cap Q_2$. With one cut derived from (27) for $x \in Q_1^o$, we get a new constraint

Table 1 ABC algorithm with CPT-D for Example 1.0

Iter	v	V	x	Num Nodes	$f(x, \omega_1)$	Num Cuts	$f(x, \omega_2)$	Num Cuts	Benders' cut for Node
1	$-M - 5.5$	inf	(1, 1)	1	-28	1	-19	0	$\eta \geq -40 + 16.5x_2$
2	-41.5	-29	(1, 0)	1	-47	0	-19	4	$\eta \geq -40.55 + 7.55x_1 + 3.8333x_2$
3	-40	-34.5	(0, 0)	2	-47	0	-28	1	$\eta \geq -37.5 + 8.3333x_2$
4	-37.5	-37.5	(0, 0)	3					

$$- 2y_2 - 2y_4 + 2R \geq -4 + 2x_2, \tag{44}$$

and upon solving the updated formulation, we get $y(\omega_1) = (0, 0, 0, 1, 0)$, and no more cuts are necessary. For scenario ω_2 , (8) is solved by a B&B method and we get only one node in \mathcal{T}_2 with Q_2 untouched. No cuts are needed, and $y(\omega_2) = (0, 1, 0, 0, 0)$. Since all scenarios have integer solutions, we update V : $V = -29$ and, the Benders' cut for $x \in Q_1^o$ is

$$- 16.5x_2 + \eta \geq -40. \tag{45}$$

At iteration $k = 2$, the master problem continues to be solved by a B&B method. We get a solution $(x_1, x_2, \eta) = (1, 0, -40)$ with objective $v = -41.5$. Using $x = (1, 0)$ with $Q_1^o = \{0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$ we now derive the subproblems. The **BB-D** is called for each $\omega \in \Omega$. For scenario ω_1 , (8) is initialized as follows:

$$f(x, \omega_1) = \min \quad - 16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \tag{46a}$$

$$\text{s.t.} \quad 2y_1 + 3y_2 + 4y_3 + 5y_4 - R \leq 10 - x_1 \tag{46b}$$

$$6y_1 + 1y_2 + 3y_3 + 2y_4 - R \leq 3 - x_2 \tag{46c}$$

$$- 2y_2 - 2y_4 + 2R \geq -4 + 2x_2 \tag{46d}$$

$$y_i \text{ binary } i = 1, \dots, 4, R \geq 0 \tag{46e}$$

where constraint (46d) is inherited from iteration 1 [see (44)]. We solve (46) by a B&B method and we get \mathcal{T}_2 with one node. Its bound is Q_2 . We have $y(\omega_1) = (0, 1, 0, 1, 0)$. For scenario ω_2 , (8) is solved by a B&B method and we get \mathcal{T}_2 with 4 nodes. Their bounds are $\{y_1 = 1\} \cap Q_2$, $\{y_1 = 1, y_3 = 1\} \cap Q_2$, $\{y_1 = 0, y_3 = 0, y_4 = 0\} \cap Q_2$ and $\{y_1 = 0, y_3 = 0, y_4 = 1\} \cap Q_2$. Here 4 cuts are derived from (27) for $x \in Q_1^o$:

$$\begin{aligned} - 8.6667y_1 + 2.1667R &\geq 0 \\ -4y_3 + 4R &\geq 0 \\ -3.75y_2 - 5y_4 + 1.25R &\geq -5 \\ -x_1 - y_4 + R &\geq -1. \end{aligned} \tag{47}$$

With these 4 cuts added, the solution $y(\omega_2) = (0, 1, 0, 1, 0)$. Since all scenarios have integer solutions, we update V : $V = -34.5$ and, the Benders' cut for $x \in Q_1^o$ is

$$- 4.5x_1 - 3.8333x_2 + \eta \geq -37.5. \tag{48}$$

At iteration $k = 3$, the master problem continues to be solved by the B&B method. We get a solution $(x_1, x_2, \eta) = (0, 0, -37.5)$ with objective $v = -37.5$. The solution belongs to node 2 with $Q_1^2 = \{0 \leq x_1 \leq 1, 0 \leq x_2 \leq 0\}$. Here the first stage solution $x = (0, 0)$ and the box Q_1^2 are treated as input parameters for **BB-D** for each scenario $\omega \in \Omega$. For scenario ω_1 , the solution is $y(\omega_1) = (0, 1, 0, 1, 0)$. For scenario ω_2 , the solution is $y(\omega_2) = (0, 0, 0, 1, 0)$. $V = -37.5$ and, the Benders' cut for Q_1^o is

$$- 2.8333x_1 - 5.1667x_2 + \eta \geq -37.5. \tag{49}$$

At iteration $k = 4$, with updated Benders' cut for node 2, the master problem continues to be solved by a B&B method. We obtain $(x_1, x_2, \eta) = (0, 0, -37.5)$ with objective $v = -37.5$. $V - v \leq \epsilon$ and, the algorithm stops. A short summary of the **ABC** algorithm with **BB-D** is shown in Table 2. As one might notice, there is only a small difference between **BB-D** and **CPT-D** for Example 1.0. At iteration 2, because **BB-D** uses partitions with 4 terms to generate cuts, the quality of the Benders' cut is better than **CPT-D**.

Example 1.1 is an extension of Example 1.0, and is intended to illustrate the workings of the algorithm when we include general integer variables in the second stage.

$$\min \quad -1.5x_1 - 4x_2 + \sum_{\omega \in \Omega} p(\omega)f(x, \omega) \tag{50a}$$

$$\text{s.t.} \quad x_1, x_2 \text{ binary} \tag{50b}$$

where

$$f(x, \omega) = \min -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \tag{51a}$$

$$\text{s.t.} \quad \begin{bmatrix} 2y_1 + 3y_2 + 4y_3 + 5y_4 - R \\ 6y_1 + 1y_2 + 3y_3 + 2y_4 - R \end{bmatrix} \leq r(\omega) - T(\omega)x \tag{51b}$$

$$y_i \in \{0, 1 \dots 5\}, i = 1, \dots, 4; R \geq 0, \tag{51c}$$

$$\Omega = \{\omega_1, \omega_2\}, p(\omega_1) = p(\omega_2) = 0.5, r(\omega_1) = \begin{bmatrix} 10 \\ 3 \end{bmatrix}, T(\omega_1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, r(\omega_2) = \begin{bmatrix} 5 \\ 2 \end{bmatrix}, T(\omega_2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The summaries of applying **ABC** algorithm with **CPT-D** and **BB-D** on Example 1.1 are shown in Table 3 and Table 4.

In Tables 3 and 4, the rows show the information generated in each successive iteration of the algorithm. The column header “Num Nodes” indicates the number of active nodes in the B&B tree and “Num Cuts” indicates the number of multi-term disjunctive cuts generated for that scenario. In the first iteration, $v = -M - 5.5$ where $-M$ denotes the lower bound for η_t [see (15)] and in our experiment we set $M = 1000000$. From the Table 4, we can observe that both algorithms require one more iteration and generate more cuts in the subproblem than in Example 1.0 (Tables 1, 2), but the differences between the execution of **BB-D** and **CPT-D** for this example are minimal.

To better illustrate the algorithm, we also include two sets of figures (Figs. 2, 3 in “Appendix 2”) which show the master problem B&B tree at each iteration of the algorithm. The node with bold circle contains the solution x^k and gets the objective function approximation updated in the iteration. Similar to the results shown in the Tables 1 and 2, there are only minor differences for the master problem B&B tree between the two algorithms.

In connection with this illustration, we also present two figures (Figs. 4, 5 in “Appendix 2”) which show how one tracks the index sets of cuts associated with various nodes

Table 2 ABC algorithm with **BB-D** for Example 1.0

Iter	v	V	x	Num Nodes	$f(x, \omega_1)$	Num Cuts	$f(x, \omega_2)$	Num Cuts	Benders' cut for Node
1	$-M - 5.5$	inf	(1,1)	1	-28	1	-19	0	$\eta \geq -40 + 16.5x_2$
2	-41.5	-29	(1,0)	1	-47	0	-19	4	$\eta \geq -37.5 + 4.5x_1 + 3.8333x_2$
3	-37.5	-34.5	(0,0)	2	-47	0	-28	0	$\eta \geq -37.5 + 2.8333x_1 + 5.1667x_2$
4	-37.5	-37.5	(0,0)	2					

Table 3 ABC algorithm with CPT-D for Example 1.1

Iter	v	V	x	Num Nodes	$f(x, \omega_1)$	Num Cuts	$f(x, \omega_2)$	Num Cuts	Benders' cut for Node
1	-M - 5.5	Inf	(1, 1)	1	-57	0	-76	1	$\eta \geq -73.7560 + 6.2292x_1 + 1.0268x_2$
2	-76.7292	-72	(0, 1)	1	-57	0	-80.625	4	$\eta \geq -80.0714 + 0.8929x_1 + 11.2589x_2$
3	-73.7560	-72.8125	(0, 0)	2	-60.2979	6	-80	5	$\eta \geq -70.1489 + 2.4734x_1 + 0.9468x_2$
4	-72.8125	-72	(0, 1)	3	-57	0	-80	2	$\eta \geq -79 + 1.4583x_1 + 10.5x_2$
5	-72.5	-72.5	(0, 1)	3					

Table 4 ABC algorithm with **BB-D** for Example 1.1

Iter	v	V	x	Num Nodes	$f(x, \omega_1)$	Num Cuts	$f(x, \omega_2)$	Num Cuts	Benders' cut for Node
1	-M-5.5	Inf	(1, 1)	1	-57	0	-76	1	$\eta \geq -73.7560 + 6.2292x_1 + 1.0268x_2$
2	-76.7292	-72	(0, 1)	1	-57	0	-80.1250	6	$\eta \geq -79.7232 + 1.5089x_1 + 11.1607x_2$
3	-73.7560	-72	(0, 0)	2	-57	6	-80	4	$\eta \geq -68.5 + 2x_1$
4	-72.5625	-72	(0, 1)	2	-57	0	-80	2	$\eta \geq -78.8571 + 1.4643x_1 + 10.3571x_2$
5	-72.5	-72.5	(0, 1)	3					

of the B&B trees. With all generated Benders' cuts as one pool and all generated sub-problem' cuts as another pool, the sequence encodes the validity of the cuts from each pool for any given node of the master problem B&B tree. Again, there are only minor differences in the execution of the two algorithms for this example.

Example 1.2 This example extends Example 1.1 by requiring general integer variables in the first-stage as well. So instead of having x_1, x_2 to be binary, they are now allowed to be integers and bounded by $0 \leq x_1, x_2 \leq 5$. The summaries of applying two algorithms are shown in Tables 5 and 6. Compared to the previous example, Tables 5 and 6 demonstrate that due to a larger number of choices resulting from general integer variables in the first stage, the algorithms require more iterations to solve the problem. Note also that using **BB-D** requires more iterations than **CPT-D** for this instance, although the average number of cuts for **BB-D** is fewer than that used by **CPT-D**.

Example 1.3 As with Example 1.2, we maintain the same range of integers in the range $[0, 5]$, although we allow randomness in the T matrix, stated as follows.

$$T(\omega_1) = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.5 \end{bmatrix}, T(\omega_2) = \begin{bmatrix} 0.3 & 0.2 \\ 0 & 0.2 \end{bmatrix}.$$

The summary of applying the algorithms are shown in Tables 7 and 8. From Tables 7 and 8 we can conclude that **BB-D** requires fewer disjunctive cuts on average for solving subproblems than **CPT-D**. In addition, the total number of iterations and Benders' cuts in the first stage are also fewer. Comparing Example 1.3 with Example 1.2, we also observe it takes more iterations to solve Example 1.3 (with random T) than to solve Example 1.2 (with fixed T). This is due the fact that T in Example 1.3 has higher density than in Example 1.2.

5 Computational experiments

This section presents three computational experiments. In Sect. 5.1, we present computations using larger versions of the examples of Sect. 4. These examples are solved using a combination of MATLAB and CPLEX 12.3. The MATLAB code implements the first-stage B&B, whereas, the rest of the algorithm was coded in C++. The purpose of this combination was to study whether a deterministic equivalent formulation (DEF), solved using a commercial state-of-the-art solver provides any advantage over our decomposition methodology. One would expect that the DEF approach would win this race because of the advances in commercial solvers for deterministic MIPs. The results of Sect. 5.1 reveal that despite the strides made by commercial MIP solvers for DEF, they are not competitive with the decomposition approach for SMIP models.

There are two further sub-sections of computations in this section. In Sect. 5.2 we upgrade the first-stage branch-and-bound implementation from MATLAB to C++, and report whether the algorithm scales well with first-stage decisions. And finally in Sect. 5.3, we present computations with a new class of test instances which are generalizations of the SSLP instances [16]. The new test instances, which we refer to as Stochastic Server Location and Sizing (SSLS), allow general integer variables in both stages.

Table 5 ABC algorithm with CPT-D for Example 1.2

Iter	v	V	x	Num Nodes	$f(x, \omega_1)$	Num Cuts	$f(x, \omega_2)$	Num Cuts	Benders' cut for Node
1	$-M - 27.5$	Inf	(5, 5)	1	100	0	-47	2	$\eta \geq -261 + 5x_1 + 52.5x_2$
2	-261	-1	(0, 0)	1	-60.9545	4	-81.9048	4	$\eta \geq -71.4297 + 3.7564x_1 + 1.0352x_2$
3	-80.3241	-1	(0, 3)	2	-19	0	-77.9165	6	$\eta \geq -83.2386 + 11.5934x_2$
4	-74.3945	-1	(0, 1)	3	-57	0	-80	4	$\eta \geq -72.0974 + 1.1915x_1 + 3.5974x_2$
5	-72.5	-72.5	(0, 1)	4					

Table 6 ABC algorithm with **BB-D** for Example 1.2

Iter	v	V	x	Num Nodes	$f(x, \omega_1)$	Num Cuts	$f(x, \omega_2)$	Num Cuts	Benders' cut for Node
1	$-M - 27.5$	Inf	(5, 5)	1	100	0	-47	2	$\eta \geq -261 + 5x_1 + 52.5x_2$
2	-261	-1	(0, 0)	1	-59.6667	5	-80	5	$\eta \geq -69.8333 + 2x_1 + 1.3333x_2$
3	-77.8333	-1	(0, 3)	2	-19	0	-76	3	$\eta \geq -90.25 + 14.25x_2$
4	-73.6667	-59.5	(3, 2)	5	-38	0	-61	5	$\eta \geq -74.1 + 1.8667x_1 + 9.5x_2$
5	-72.75	-62	(2, 2)	5	-38	0	-66	6	$\eta \geq -81 + 5x_1 + 9.5x_2$
6	-72.5	-63	(0, 1)	5	-57	0	-80	0	$\eta \geq -70.5 + 2x_1 + 2x_2$
7	-72.5	-72.5	(0, 1)	5					

Table 7 ABC algorithm with CPT-D for Example 1.3

Iter	v	V	x	Num Nodes	$f(x, \omega_1)$	Num Cuts	$f(x, \omega_2)$	Num Cuts	Benders' cut for Node
1	$-M - 27.5$	Inf	(5, 5)	1	-22.8333	2	-66.5	2	$\eta \geq -91.1061 + 0.95x_1 + 8.3379x_2$
2	-93.8561	Inf	(5, 0)	1	-59.3929	4	-71.3750	4	$\eta \geq -74.6060 + 1.8444x_1 + 0.4550x_2$
3	-81.6960	Inf	(0, 2)	2	-57	0	-79.36	6	$\eta \geq -79.2 + 0.84x_1 + 5.51x_2$
4	-80.8424	Inf	(5, 3)	2	-38	4	-66.925	8	$\eta \geq -65.1930 + 2.3312x_1 + 0.3581x_2$
5	-79.48	Inf	(5, 2)	3	-57	0	-67.7086	10	$\eta \geq -84.1132 + 2.3273x_1 + 5.0612x_2$
6	-78.6816	Inf	(4, 2)	4	-57	0	-69.2180	12	$\eta \geq -75.4437 + 0.5315x_1 + 5.1043x_2$
7	-78.16	Inf	(3, 2)	4	-57	0	-70.5438	14	$\eta \geq -82.267 + 2.7281x_1 + 5.1554x_2$
8	-77.8543	Inf	(5, 2)	6	-57	0	-66.4930	16	$\eta \geq -71.2465 + 4.75x_2$
9	-77.8065	Inf	(1, 1)	7	-57	3	-76	13	$\eta \geq -72.3252 + 5.8252x_2$
10	-77.69	-72	(0, 1)	7	-57	1	-77.3333	20	$\eta \geq -67.1667$
11	-77.5	-72	(2, 2)	7	-57	0	-76	0	$\eta \geq -79.2524 + 1.1355x_1 + 5.2407x_2$
12	-77.5	-77.5	(2, 2)	7					

Table 8 ABC algorithm with **BB-D** for Example 1.3

Iter	v	V	x	Num Nodes	$f(x, \omega_1)$	Num Cuts	$f(x, \omega_2)$	Num Cuts	Benders' cut for Node
1	-M-27.5	Inf	(5, 5)	1	-22.8333	2	-65.6250	3	$\eta \geq -91.2652 + 1.0375x_1 + 8.3697x_2$
2	-93.5777	Inf	(5, 0)	1	-57	4	-70.5	4	$\eta \geq -73.6806 + 1.9861x_1$
3	-81.6806	Inf	(0, 2)	2	-57	0	-76	4	$\eta \geq -77.2950 + 5.3975x_2$
4	-80.4686	-74.5	(5, 3)	2	-38	6	-64.6375	5	$\eta \geq -65.8580 + 2.7049x_1 + 0.3382x_2$
5	-79.7361	-74.5	(4, 2)	3	-57	0	-67.5333	7	$\eta \geq -73.1165 + 0.2569x_1 + 4.9111x_2$
6	-79	-74.5	(3, 2)	4	-57	0	-68.4810	10	$\eta \geq -81.0190 + 3.7595x_1 + 3.5x_2$
7	-77.5098	-74.5	(5, 2)	5	-57	0	-66	4	$\eta \geq -69.8442 + 1.0174x_1 + 1.6287x_2$
8	-77.5	-77	(2, 2)	5	-57	0	-76	0	$\eta \geq -76.0454 + 1.1008x_1 + 3.6719x_2$
9	-77.5	-77.5	(2, 2)	5					

5.1 Comparisons between ABC and DEF

In this subsection we compare the decomposition approach of ABC with solving a deterministic equivalent formulation (DEF) using state-of-the-art CPLEX MIP solvers with default setting through CPLEX 12.3 MATLAB API. We designed a MATLAB implementation of the ABC algorithm which calls the CPLEX LP solver whenever an LP solution is required. For all other purposes (e.g. managing the B&B tree) the MATLAB script operates in the MATLAB environment.

Three sets of instances are generated based on Examples 1.1–ex1.3. For each example, we create 4, 9, 36, 121, 441, 1681, 10201 scenarios by generating the right hand sides $r(\omega)$ from equidistant lattice points in $[5, 15] \times [5, 15]$ with equal probability assigned to each point. This methodology was borrowed from Schultz et al. [19] (see also [1]). For the seven instances based on Example 1.3, we use the same random right hand side $r(\omega) = \begin{bmatrix} r_1(\omega) \\ r_2(\omega) \end{bmatrix}$ as in Example 1.2, but in addition $T(\omega)$ is also random. The entries for these matrices were 0 or 1 with equal probability.

Table 9 compares the performance of three algorithms: two based on using the ABC algorithm with CPT-D and BB-D whereas, the third algorithm used CPLEX 12.3 (with default setting) for the MILP formulation of the deterministic equivalent formulation (DEF). All approaches were run on a Windows 7 PC operating with Intel i7-3770K 3.5 GHz processors and 8 GB memory. Instances 1-3 in the table correspond to the variations based on Examples 1.1–ex1.3. The column heading Obj denotes the optimal objective value of the SMIP, Var and Constr denote the number of variables and constraints in the DEF. The entries in column ABC (CPT – D) and ABC (BB – D) denote Iterations (Master Nodes, Second-stage Leaf Nodes, Running Time) which correspond to the total number of iterations, the total number of nodes in the B&B tree in the master problem, the maximum leaf nodes encountered during solving the subproblem and the total running time. The DEF column shows the CPU time (in s) required to solve DEF using the default version CPLEX 12.3 MIP solver. The maximum CPU time allowed was 60 min.

The results reported in Table 9 clearly demonstrate that the approach of solving a DEF with a commercial solver does not scale well, failing to solve 9 instances for which the number of scenarios were somewhat large. In comparing the performance of ABC (CPT – D) and ABC (BB – D), we observe that the former also runs into numerical difficulties for 4 of the larger instances. In contrast, ABC (BB – D) produces optimal solutions for all the instances within very reasonable computational times. From the log-log plot in Fig. 6 (see “Appendix”), as number of scenarios increases, the running time also increases polynomially for ABC (BB – D). The slopes of each of the three graphs are slightly larger than one (with values 1.0048, 1.1245, 1.1677) which suggests that as the number of scenarios increases, the running time increases at a rate that is only slightly worse than “linear”. To sum up, based on the instances tested, algorithm ABC shows very stable results and scales quite well with the number of scenarios. When CPLEX is able to solve an instance, it does so faster than our approach; however, when the number of scenarios increases, it does not find an optimal solution within an hour of CPU time, while the ABC approach does.

Table 9 Comparison of ABC with DEF

Scenarios	Obj	Var	Constr	ABC(CPT-D) $K(\mathcal{T}_1, \mathcal{T}_2, s)$	ABC(BB-D) $K(\mathcal{T}_1, \mathcal{T}_2, s)$	DEF $s(Gap)$
<i>Instance 1</i>						
4	-63.50	26	17	7 (4, 5, 0.234)	5 (4,5,0.14)	0.23
9	-66.56	56	37	6 (2, 6, 0.29)	18 (15, 10, 0.98)	0.02
36	-66.83	218	145	7 (2, 7, 1.36)	6 (2, 7, 1.01)	0.02
121	-67.17	728	485	7 (1, 8, 4.43)	6 (1, 8, 2.96)	0.16
441	-65.58	2648	1765	16 (3, 15, 46.63)	10 (2, 7, 15.27)	1.58 ^a
1681	-64.72	10088	6725	16 (3, 17, 262.44)	12 (3, 23, 85.29)	Timed out (0.2%)
10201	-64.19	61208	40805	Numerical issue	13 (3, 23, 583.08)	Timed out (0.18%)
<i>Instance 2</i>						
4	-63.50	26	17	12 (10, 10, 0.37)	12 (14, 10, 0.3)	0.02
9	-66.56	56	37	20 (15, 10, 1.92)	18 (15, 10, 0.94)	0.02
36	-69.86	218	145	19 (16, 12, 7.64)	18 (16, 10, 4.54)	0.03
121	-71.12	728	485	18 (16, 11, 25.18)	17 (16, 11, 13.51)	4.09
441	-69.64	2648	1765	20 (16, 22, 168.43)	18 (15, 21, 58.97)	Timed out (0.53%)
1681	-68.85	10088	6725	Numerical issue	20 (16, 27, 312.36)	Timed out (1.59%)
10201	-68.45	61208	40805	Numerical issue	21 (18, 31, 2333.307)	Timed out (2.76%)
<i>Instance 3</i>						
4	-63.50	26	17	10 (6, 7, 0.55)	19 (18,8,0.41)	0.05
9	-64.22	56	37	19 (15, 10, 2.09)	19 (15, 10, 0.83)	0.06
36	-66.42	218	145	25 (21, 12, 7.85)	25 (19, 13, 3.62)	1.25
121	-64.78	728	485	25 (20, 12, 26.94)	25 (19, 13, 12.04)	12.12
441	-63.33	2648	1765	28 (21, 24, 222.82)	27 (20, 24, 82.60)	Timed out (1.94%)
1681	-62.19	10088	6725	31 (22, 28, 1210.94)	28 (23, 33, 419.42)	Timed out (2.28%)
10201	-61.83	61208	40805	Numerical issue	31 (24, 34, 3243.82)	Timed out (3.04%)

K is the total number of iterations; \mathcal{T}_1 is the total number of nodes in the B&B tree in the master problem; \mathcal{T}_2 is the maximum leaf nodes encountered during solving the subproblem; s is the total running time (s) and time limit is 3600s; Gap is the MIP gap returned by CPLEX when it timed out

^a Obj = 65.576 and integer tolerance, optimal

5.2 Computations with larger first-stage instances

In this subsection we study how the ABC algorithm performs when the number of first-stage (search) variables increase. The larger instances are generated by including two more variables at a time to Examples 1.2 and 1.3. For the i -th time, the newly added pair of variables have objective coefficients given by $-1.5 * 2^i$ and $-4 * 2^i$. For the larger instance of Example 1.2, matrix $T(\omega)$ is fixed. The fixed matrix is extended by a 2×2 identity matrix with the newly pair of variables added. In the case of Example 1.3, $T(\omega)$'s elements are uniformly generated from set $\{0.1, 0.2, 0.3, \dots, 0.9\}$. The rest of the parameters are generated the same way as in instance 2 and instance 3.

We remind the reader that the objective function contours of this instance have been depicted in the case of 2 variables by Schultz et al. [19] and Ahmed et al. [1]. These papers also captured the challenges posed by the two variable instance. In comparison, the total number of first-stage feasible solutions in our extension increases the number of feasible first-stage solutions exponentially according to the sequence 36^i , $i = 1, 2$, etc. The largest instances we attempt have a total number of feasible first-stage solutions given by 36^6 , implying a total of more than 2.1 billion feasible solutions. Of course, using MATLAB to search for optimal solutions of such instances would be untenable. So, the entire ABC algorithm is implemented in C++ including the first-stage B&B and second-stage decomposition. All linear programs were solved using CPLEX 12.6. Finally, we also note that our experiments were conducted for the most promising scheme among those compared in the previous section. Thus, the following reports our results using the **BB-D** version of the ABC algorithm.

The computational results of extending Example 1.2 are shown in Table 10. The column heading **First-Stage Var Num** denotes the number of first-stage variables, and the remaining information follows the same style as reported in Table 9. Comparing **First-Stage Var Num** = 2 with instance 2 from Table 9, we can see the speed up of C++ implementation compared to using MATLAB. And as **First-Stage Var Num** grows each time by two, Iterations and Master Nodes required to solve the problem grow exponentially. We encounter some numerical difficulties for solving instances of 8 variables with 121 and 441 scenarios and 10 variables with 441 scenarios. Nevertheless, the algorithm did solve instances of 12 variables in reasonable time. Thus, as the number of first-stage variables grows, instances may not become harder, and such anomalies are also known to exist for deterministic MIPs as well.

The computational result of extending Example 1.3 (random T) is shown in Table 11. We can see the same speed up of C++ implementation compared to MATLAB as well as the exponential growth of Iterations and Master Nodes as **First-Stage Var Num** are added by two each time. There are numerical difficulties for solving instances of 8 and 10 variables with 121 and 441 scenarios and 12 variables with 441 scenarios. Because T is random and relatively dense, these instances become harder to solve.

5.3 Computational results with stochastic server location and sizing

One application of SMIP is in the network design and planning where servers need to be located in a least cost manner before demand from customers is known. This problem arises in a variety of domains such as telecommunication, electricity power, water distribution, internet service and anti-terrorism. One of the instances available in the literature, attributed to Ntaimo and Sen [16] is a stochastic server location problem (SSLP) in which demand uncertainty is limited to whether or not demand will occur at nodes of a graph. The goal of SSLP is to place servers at discrete locations in such a way that the expected cost of serving demand is minimized. That collection of instances have 0–1 variables in both stages. In order to test the ABC algorithm, we extend the above instances to a class of problems where in addition to location, we are interested in sizing the servers, where the size is measured in discrete units

Table 10 Computational results with **ABC/BB-D** as first-stage variables increase for Example 1.2 (fixed T)

First-Stage Var Num	Scenarios	Obj	Var	Constr	ABC(BB-D)	DEF
					$K(\mathcal{T}_1, \mathcal{T}_2, s)$	$s(Gap)$
2	4	-63.5	26	17	11 (11, 8, 0.04)	0.05
	9	-66.56	56	37	10 (7, 10, 0.07)	0.25
	36	-69.86	218	145	12 (9, 10, 0.35)	0.09
	121	-71.12	728	485	12 (11, 10, 1.24)	0.76
	441	-70.23	2648	1765	14 (17, 21, 9.97)	Timed out (0.36%)
4	4	-74	28	17	21 (47, 9, 0.09)	0.13
	9	-81.89	58	37	26 (51, 10, 0.17)	0.02
	36	-87.08	220	145	21 (35, 10, 0.63)	0.05
	121	-89.07	730	485	31 (61, 10, 2.71)	0.09
	441	-87.39	2650	1765	30 (69, 19, 20.90)	1.37
6	4	-125.75	30	17	181 (833, 8, 0.49)	0.02
	9	-130.78	60	37	135 (635, 10, 0.76)	0.03
	36	-134.17	222	145	145 (603, 10, 3.73)	0.34
	121	-135.3	732	485	152 (591, 10, 17.99)	1.15
	441	-134.07	2652	1765	177 (677, 17, 158.55)	Timed out (0.09%)
8	4	-235.75	32	17	56 (185, 10, 0.18)	0.03
	9	-240.78	62	37	1810 (6681, 9, 18.82)	0.03
	36	-244.17	224	145	2255 (8031, 10, 193.42)	0.03
10	4	-455.95	34	17	24 (107, 8, 0.07)	0.02
	9	-460.87	64	37	19 (63, 8, 0.11)	0.02
	36	-489.45	226	145	78 (229, 9, 1.70)	0.03
	121	-500.53	736	485	442 (1325, 9, 73.96)	0.45
12	4	-965.25	36	17	14611 (41993, 9, 184.77)	0.02
	9	-1045.33	66	37	217 (905, 9, 1.14)	0.01
	36	-1089.56	228	145	52 (231, 8, 0.92)	0.02
	121	-1108.23	738	485	65 (245, 8, 4.53)	0.05
	441	-1115.72	2658	1765	58 (215, 11, 20.97)	0.95

K is the total number of iterations; \mathcal{T}_1 is the total number of nodes in the B&B tree in the master problem; \mathcal{T}_2 is the maximum number of leaf nodes encountered during solving the subproblem; s is the total running time (s) and time limit is 3600 s; Gap is the MIP gap returned by CPLEX when it timed out

[0, 1, 2, . . . , 5]. We refer to these test instances as the Stochastic Server Location and Sizing (SSLS) instances. The precise model formulation is given in ‘‘Appendix 1’’.

The size of the model is captured by the name ‘‘SSLS- $(m \times u)-(n \times v)$ - S ’’, where m is the number of potential server locations, u is the maximum number of servers allowed for each location, n is the number of potential client locations, v is the maximum number of possible clients for each location and S is the number of scenarios. We report results on problem instances with $m \in \{2, 3, 4\}$, $n \in \{5, 10, 15\}$, $S \in \{50, 100, 500\}$, $u = 5$ and $v = 5$. A summary of the notations is shown in Table 12. We should note

Table 11 Computational results with ABC/BB-D as first-stage variables increase Example 1.3 (random T)

First-Stage Var Num	Scenarios	Obj	Var	Constr	ABC(BB-D)	DEF
					$K(\mathcal{T}_1, \mathcal{T}_2, s)$	$s(Gap)$
2	4	-63.5	26	17	13 (31, 16, 0.10)	0.05
	9	-64.22	56	37	24 (63, 23, 0.44)	0.06
	36	-66.46	218	145	22 (47, 21, 1.40)	1.25
	121	-64.78	728	485	27 (55, 32, 7.07)	12.12
	441	-63.33	2648	1765	27 (55, 28, 28.23)	Timed out (0.36%)
4	4	-79.7	28	17	13 (31, 16, 0.10)	0.14
	9	-82.78	58	37	24 (63, 23, 0.44)	0.09
	36	-83.02	220	145	22 (47, 21, 1.40)	1.86
	121	-83.14	730	485	27 (55, 32, 7.07)	26.5
	441	-82.41	2650	1765	27 (55, 28, 28.23)	Timed out (0.29%)
6	4	-127.75	30	17	135 (371, 20, 0.54)	0.03
	9	-124.49	60	37	139 (407, 21, 1.49)	0.14
	36	-126.56	222	145	163 (435, 22, 7.28)	2.04
	121	-128.61	732	485	183 (475, 24, 39.77)	Timed out (1.19%)
	441	-128.87	2652	1765	186 (463, 23, 246.74)	Timed out (2.48%)
8	4	-232.5	32	17	1127 (4105, 20, 8.16)	0.17
	9	-242.78	62	37	1657 (5627, 22, 26.94)	0.05
	36	-229.61	224	145	2324 (7243, 23, 293.37)	0.77
10	4	-444.75	34	17	27505 (87493, 20, 111.41)	0.17
	9	-488.11	64	37	144 (325, 15, 1.03)	0.19
	36	-517.42	226	145	2098 (6297, 21, 205.03)	0.14
12	4	-886.4	36	17	5 (1, 17, 0.02)	0.16
	9	-1016.78	66	37	8 (3, 17, 0.04)	0.03
	36	-1070.31	228	145	158 (367, 17, 3.27)	0.08
	121	-1094.80	738	485	146 (355, 16, 13.23)	0.17

K is the total number of iterations; \mathcal{T}_1 is the total number of nodes in the B&B tree in the master problem; \mathcal{T}_2 is the maximum number of leaf nodes encountered during solving the subproblem; s is the total running time (s) and time limit is 3600s; Gap is the MIP gap returned by CPLEX when it timed out

Table 12 SSLS- $(m \times u)$ - $(n \times v)$ -S

Notation	Interpretation	Value
m	Number of potential server locations	2, 3, 4
u	Maximum number of servers allowed for each location	5
n	Number of potential client locations	5, 10, 15
v	Maximum number of clients for each location	5
S	Number of scenarios	50, 100, 500

Table 13 Computational results with **ABC/BB-D** on **SSLs** instances

Instance	Scenarios	Obj	Var	Constr	ABC(BB-D) $K(\bar{T}_1, \bar{T}_2, s)$	DEF $s(Gap)$
SSLs_(2 × 5)_(5 × 5)_50	50	-128.05	602	601	11 (13, 16, 0.30)	0.09
SSLs_(2 × 5)_(5 × 5)_100	100	-40.45	1202	1201	9 (13, 2, 0.38)	0.2
SSLs_(2 × 5)_(5 × 5)_500	500	-151.2	6002	6001	12 (13, 22, 3.58)	0.73
SSLs_(2 × 5)_(10 × 5)_50	50	-232.63	1102	1101	14 (15, 50, 0.52)	0.96
SSLs_(2 × 5)_(10 × 5)_100	100	-237.67	2202	2201	13 (13, 515, 4.84)	Timed out (0.15%)
SSLs_(2 × 5)_(10 × 5)_500	500	-256.88	11002	11001	13 (17, 258, 4.43)	0.76
SSLs_(2 × 5)_(15 × 5)_50	50	-457.24	1602	1601	20 (23, 344, 4.26)	Timed out (0.24%)
SSLs_(2 × 5)_(15 × 5)_100	100	-420.35	3202	3201	16 (21, 100, 1.64)	0.78
SSLs_(2 × 5)_(15 × 5)_500	500	-399.47	16002	16001	15 (17, 411, 51.28)	Timed out (0.06%)
SSLs_(3 × 5)_(5 × 5)_50	50	-110.32	903	651	23 (33, 2, 0.59)	0.27
SSLs_(3 × 5)_(5 × 5)_100	100	-107.63	1803	1301	23 (33, 2, 1.23)	0.14
SSLs_(3 × 5)_(5 × 5)_500	500	-86.67	9003	6501	23 (35, 5, 7.09)	163.77
SSLs_(3 × 5)_(10 × 5)_50	50	-362.43	1653	1151	36 (59, 343, 4.68)	Timed out (0.19%)
SSLs_(3 × 5)_(10 × 5)_100	100	-339.12	3303	2301	32 (45, 8632, 158.61)	2715.04
SSLs_(3 × 5)_(10 × 5)_500	500	-364.58	16503	11501	28 (41, 566, 27.15)	Timed out (0.02%)
SSLs_(3 × 5)_(15 × 5)_50	50	-436.42	2403	1651	37 (55, 9435, 191.92)	9.73
SSLs_(3 × 5)_(15 × 5)_100	100	-371.24	4803	3301	29 (33, 1290, 19.56)	Timed out (0.14%)
SSLs_(3 × 5)_(15 × 5)_500	500	-426.54	24003	16501	40 (57, 8984, 1069.92)	Timed out (0.16%)
SSLs_(4 × 5)_(5 × 5)_50	50	-131.05	1204	701	58 (87, 8, 1.59)	1.59
SSLs_(4 × 5)_(5 × 5)_100	100	-119.18	2404	1401	59 (91, 9, 3.36)	5.07
SSLs_(4 × 5)_(5 × 5)_500	500	-73.55	12004	7001	57 (87, 2, 20.99)	4.46
SSLs_(4 × 5)_(10 × 5)_50	50	-308.69	2204	1201	63 (91, 130, 3.60)	0.98

Table 13 continued

Instance	Scenarios	Obj	Var	Constr	ABC(BB-D) $K(\mathcal{T}_1, \mathcal{T}_2, s)$	DEF $s(Gap)$
SSL _S _(4 × 5)_(10 × 5)_100	100	-350.08	4404	2401	68 (111, 6419, 261.89)	Timed out (0.15%)
SSL _S _(4 × 5)_(10 × 5)_500	500	-311.3	22004	12001	64 (97, 8683, 745.61)	Timed out (0.05%)
SSL _S _(4 × 5)_(15 × 5)_50	50	-521.07	3204	1701	104 (161, 5849, 1653.67)	Timed out (0.09%)
SSL _S _(4 × 5)_(15 × 5)_100	100	N/A	6404	3401	Timed out (>3600)	Timed out (0.01%)
SSL _S _(4 × 5)_(15 × 5)_500	500	N/A	32004	17001	Timed out (>3600)	Timed out (1.29%)

K is the total number of iterations; \mathcal{T}_1 is the total number of nodes in the B&B tree in the master problem; \mathcal{T}_2 is the maximum number of leaf nodes encountered during solving the subproblem; s is the total running time (s) and time limit is 3600s; Gap is the MIP gap returned by CPLEX when it timed out

that while the number of potential server locations appear to be small, the fact that there are alternative sizes possible makes the combinatorial content significant, and more-or-less on par with the SSLP instances.

The computational environment is the same as in Section 5.2 and we used the C++ implementation for this experiment. Table 13 shows the results which are reported in a manner similar to Table 9, 10 etc. Again the number of variables (**Var**) and constraints (**Constr**) refer to numbers in the DEF formulation. The metrics reported in the column labeled **ABC(BB-D)** refer to the number of iterations and in parentheses, the numerical quantities refer to the number of master nodes, the number of second-stage leaf nodes, and the running time. The algorithm can handle most of the instances except the last two where the task of generating cuts for all outcomes slowed down the process. One should expect that problems with fixed recourse would be easier to handle. Among the runs completed, we can see that as m gets larger, Iterations and Master Nodes required to solve the problem grow dramatically as expected. On the other hand, as n gets larger while m, u, v and S stay the same, we find the Iterations and Master Nodes do not change in general, while the Second-stage Leaf Nodes and Running Time grow exponentially. We conclude this section by evaluating the potential of the ABC algorithm by comparing the run time and number of successfully solved problems. As for total run time in Table 13, the ABC algorithm requires about 3.18 h and successfully solved 25 instances. In comparison, CPLEX 12.6 requires 12.81 h and solved 15 instances successfully. Based on these comparisons, the ABC approach shows far greater potential for SMIP models.

6 Conclusion

As stated at the outset, our paper returns to the general class of two stage SMIP problems that was the focus of the paper by Carøe and Tind [4]. This class of problems involves mixed-integer variables in both stages, and randomness is also allowed in all data elements of the second-stage MIP. Despite the elegance of the work of Carøe and Tind [4], the chasm between first and second stage strategies has persisted over the ensuing decades. Using several building blocks that have been effective in the interim, we have developed a time-staged decomposition algorithm for very general SMIP models. Other effective ideas, such as allowing the second-stage problem to be solved inexactly are also permitted within the overall strategy. The key feature of this algorithm is a first-stage B&B process (i.e. the ABC algorithm) which simultaneously guides both the construction of approximations as well as the search for optimal first-stage decisions. Furthermore, the recourse function approximations remain piecewise linear and convex for each first-stage B&B node, and similarly, the second-stage relaxations (built using multi-term disjunctions) are also polyhedral. While these elements maintain convex building blocks, the overall search is facilitated by a B&B scheme which selectively chooses valid approximations of functions and sets in a manner that allows the algorithm to discover an optimal solution to the overall problem.

We have also presented the most comprehensive computational experiment to date for problems in which mixed-integer variables appear in both stages. Our computations reveal that as the number of scenarios grow, our MATLAB-guided implementation was

faster and more stable than the commercial solver for an extensive form SMIP. We also reported computational results for larger instances which required a more sophisticated implementation than the MATLAB prototype. Both of those experiments (one with an extended test instance, and another with a new test problem) reveal that the ABC framework, especially in the BB-D setting, leads to a viable approach for realistic instances. We recommend at least two avenues in which the ABC/BB-D approach may be promising: (a) implementing these ideas on parallel machines should be fruitful, and (b) these concepts should be extended to multi-stage models. Our computational experiment shows that when the SMIP models have large number of first stage variables and large number of scenarios, our decomposition approach is preferable to using a state-of-the-art commercial solver for the DEF.

Acknowledgements The authors are grateful to the referees for their detailed reading of the paper. Their suggestions improved the presentation, and made the paper more compelling.

7 Appendix 1: SSSL model formulation

To describe the model, let I be the index set for client locations and J be the index sets for possible server locations. For $i \in I$ and $j \in J$, we define the following.

Parameters

- c_j Cost of locating a server at location j .
- q_i Revenue from a client at location i being served by servers at location j .
- q_{j0} Loss of revenue because of overflow at server j .
- d_{ij} Resource requirement of client i for server at location j .
- r Upper bound on the total number of servers that can be located.
- u Upper bound of number of server located at one location.
- w Server capacity.
- v Upper bound of possible number of clients.
- $h^i(\omega)$ The number of clients at location i in scenario ω
- $h(\omega)$ The vector made of elements $h^i(\omega)$.
- $p(\omega)$ Probability of occurrence for scenario $\omega \in \Omega$.

Decisions variables

- x_j number of servers located at site j .
- y_{ij} number of clients at location i served by servers at location j .
- y_{j0} overflow amount at server location j .

The SSSL can be formulated as follows.

$$\min \sum_{j \in J} c_j x_j - E[f(x, \tilde{\omega})] \tag{52a}$$

$$\text{s.t.} \sum_{j \in J} x_j \leq r, \tag{52b}$$

$$x_j \in \{0, 1, \dots, u\}, \quad \forall j \in J, \tag{52c}$$

where

$$E[f(x, \tilde{\omega})] = \sum_{\omega \in \Omega} p(\omega) f(x, \omega) \tag{53}$$

For any x satisfying (52b, 52c) and $\omega \in \Omega$,

$$f(x, \omega) = \min - \sum_{i \in I} \sum_{j \in J} q_{ij} y_{ij} + \sum_{j \in J} q_{j0} y_{j0} \tag{54a}$$

$$\text{s.t. } \sum_{i \in I} d_{ij} y_{ij} - y_{j0} \leq w x_j, \quad \forall j \in J, \tag{54b}$$

$$\sum_{j \in J} y_{ij} = h^i(\omega), \quad \forall i \in I, \tag{54c}$$

$$y_{ij} \in \{0, 1, 2, \dots, v\}, \quad \forall i \in I, j \in J, \tag{54d}$$

$$y_{j0} \geq 0, \quad \forall j \in J. \tag{54e}$$

We generated a set of SSLS instances following the method in [16]. For problem data, the server location costs c_j were generated randomly from the uniform distribution in the interval [40, 80] and the client demands d_{ij} were generated in the interval [0, 25]. The client-server revenue were set at one unit per unit of client demand. The overflow costs q_{j0} were fixed at 1000. For scenario data, the number of clients available in each scenario $h^i(\omega)$ were generated from a binomial distribution with $p = 0.5$ and v trials.

8 Appendix 2: Figures for examples

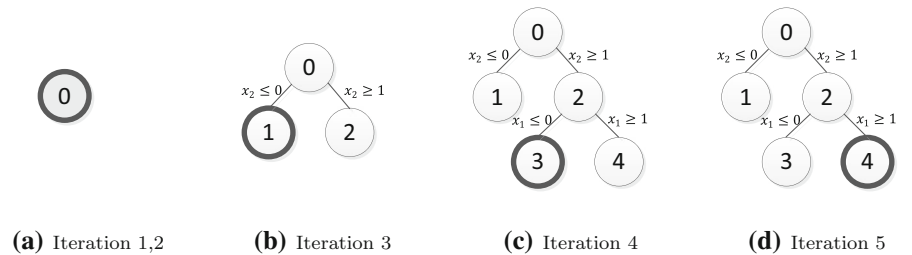


Fig. 2 Master problem B&B tree for ABC algorithm with CPT-D on Example 1.1

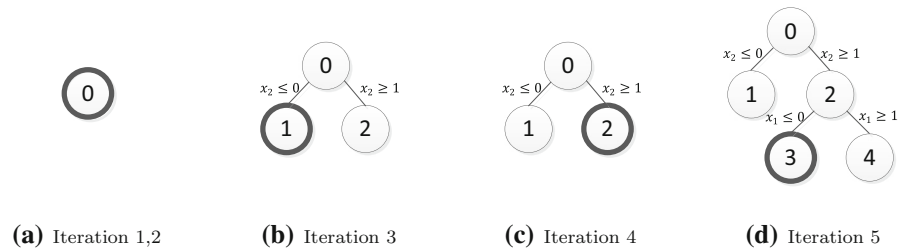


Fig. 3 Master problem B&B tree for ABC algorithm with BB-D on Example 1.1

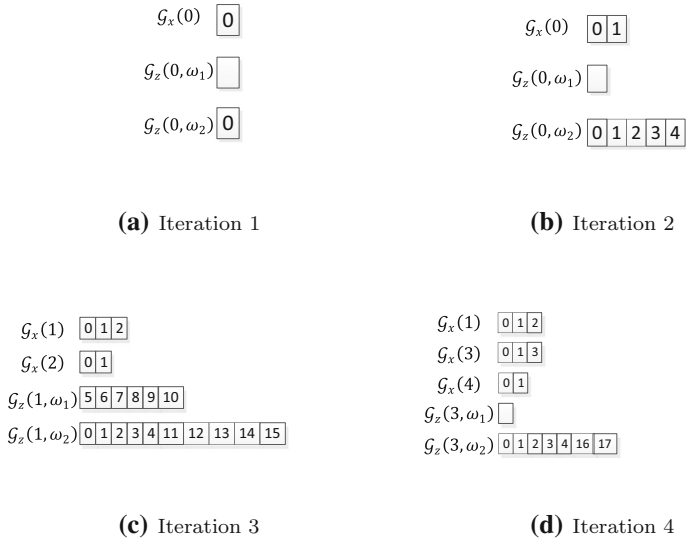


Fig. 4 $\mathcal{G}_x(t), \mathcal{G}_z(t, \omega)$ for ABC algorithm with CPT-D on Example 1.1

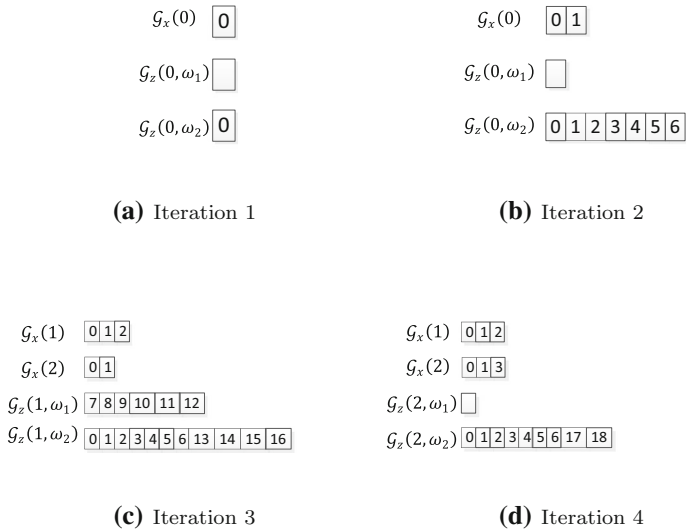


Fig. 5 $\mathcal{G}_x(t), \mathcal{G}_z(t, \omega)$ for ABC algorithm with BB-D on Example 1.1

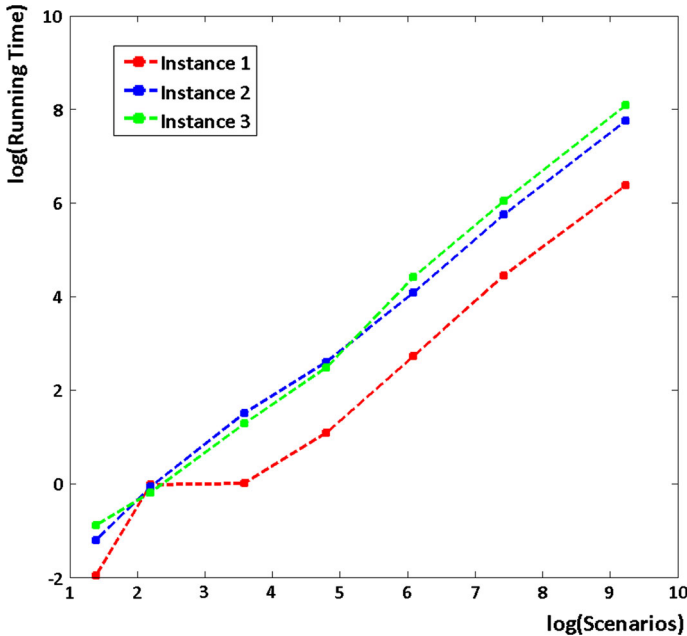


Fig. 6 Log-log plot of running time and scenarios for ABC/BB-D for instances in Table 9

References

- Ahmed, S., Tawarmalani, M., Sahinidis, N.V.: A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Math. Program.* **100**(2), 355–377 (2004)
- Balas, E.: Disjunctive programming. *Ann. Discret. Math.* **5**, 3–51 (1979)
- Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Math. Program.* **58**, 295–324 (1993)
- Carøe, C., Tind, J.: L-shaped decomposition of two-stage stochastic programs with integer recourse. *Math. Program.* **83**, 451–464 (1998)
- Carøe, C.C., Schultz, R.: Dual decomposition in stochastic integer programming. *Oper. Res. Lett.* **24**, 37–45 (1997)
- Chen, B., Küçükyavuz, S., Sen, S.: Finite disjunctive programming characterizations for general mixed integer linear programs. *Oper. Res.* **59**(1), 202–210 (2011)
- Chen, B., Küçükyavuz, S., Sen, S.: A computational study of the cutting plane tree algorithm for general mixed-integer linear programs. *Oper. Res. Lett.* **40**(1), 15–19 (2012)
- Del Pia, A., Weismantel, R.: On convergence in mixed integer programming. *Math. Program.* **135**(1–2), 397–412 (2012)
- Escudero, L., Garn, A., Merino, M., Pérez, G.: A two-stage stochastic integer programming approach as a mixture of branch-and-fix coordination and benders decomposition schemes. *Ann. Oper. Res.* **152**, 395–420 (2007). doi:[10.1007/s10479-006-0138-0](https://doi.org/10.1007/s10479-006-0138-0)
- Gade, D., Küçükyavuz, S., Sen, S.: Decomposition algorithms with parametric gomory cuts for two-stage stochastic integer programs. *Math. Program.* **144**(1–2), 1–26 (2012)
- Jörg, M.: k-Disjunctive cuts and a finite cutting plane algorithm for general mixed integer linear programs. Ph.D. thesis, Technische Universität München (2008)
- Kong, N., Schaefer, A.J., Hunsaker, B.: Two-stage integer programs with stochastic right-hand sides. *Math. Program.* **108**(24), 275–296 (2006)
- Louveaux, F., Schultz, R.: Stochastic integer programming. *Handb. Oper. Res. Manag. Sci.* **10**, 213–266 (2003)

14. Lulli, G., Sen, S.: A branch-and-price algorithm for multistage stochastic integer programming with application to stochastic batch-sizing problems. *Manag. Sci.* **50**(6), 786–796, arXiv preprint [arXiv:0707.3945](https://arxiv.org/abs/0707.3945) (2004)
15. Ntaimo, L.: Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. *Oper. Res.* **58**(1), 229–243 (2010)
16. Ntaimo, L., Sen, S.: The million-variable “march” for stochastic combinatorial optimization. *J. Global Optim.* **32**(3), 385–400 (2005)
17. Owen, J.H., Mehrotra, S.: A disjunctive cutting plane procedure for general mixed-integer linear programs. *Math. Program.* **89**(3), 437–448 (2001)
18. Owen, J.H., Mehrotra, S.: On the value of binary expansions for general mixed-integer linear programs. *Oper. Res.* **50**(5), 810–819 (2002)
19. Schultz, R., Stougie, L., van der Vlerk, M.H.: Solving stochastic programs with integer recourse by enumeration: a framework using Gröbner basis. *Math. Program.* **83**(1–3), 229–252 (1998)
20. Sen, S.: *Stochastic Mixed-Integer Programming Algorithms: Beyond Benders' Decomposition*. Wiley, London (2010)
21. Sen, S., Hige, J.L.: The C3 theorem and a D2 algorithm for large scale stochastic mixed-integer programming: set convexification. *Math. Program.* **104**(1), 1–20 (2005)
22. Sen, S., Hige, J.L., Ntaimo, L.: A summary and illustration of disjunctive decomposition with set convexification. In: Woodruff, D. (ed.) *Network Interdiction and Stochastic Integer Programming*, pp. 105–125. Springer, New York (2003)
23. Sen, S., Sherali, H.D.: Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Math. Program.* **106**(2), 203–223 (2006)
24. Sherali, H.D., Zhu, X.: On solving discrete two-stage stochastic programs having mixed-integer first- and second-stage variables. *Math. Program.* **108**(2–3), 597–616 (2006)
25. Trapp, A.C., Prokopyev, O.A., Schaefer, A.J.: On a level-set characterization of the integer programming value function and its application to stochastic programming. *Oper. Res.* **61**(2), 498–511 (2013)
26. Wollmer, R.D.: Two-stage linear programming under uncertainty with 0–1 integer first stage variables. *Math. Program.* **19**(3), 279–288 (1980)
27. Yuan, Y., Sen, S.: Enhanced cut generation methods for decomposition-based branch and cut for two-stage stochastic mixed-integer programs. *INFORMS J. Comput.* **21**(3), 480–487 (2009)
28. Zhang, M., Küçükyavuz, S.: Finitely convergent decomposition algorithms for two-stage stochastic pure integer programs. *SIAM J. Optim.* **24**(4), 1933–1951 (2014)