The Surprising Performance of Random Partial Benders Decomposition

Jean Pauphilet
London Business School, jpauphilet@london.edu

Yupeng Wu
London Business School, yupengw@london.edu

Abstract. Benders decomposition is a technique to solve large-scale mixed-integer optimization problems by decomposing them into a pure-integer master problem and a continuous separation subproblem. To accelerate convergence, we propose Random Partial Benders Decomposition (RPBD), a decomposition method that randomly retains a subset of the continuous variables within the master problem. Unlike existing problem-specific approaches, RPBD is simple to implement and universally applicable. We present both computational and theoretical evidence to support its effectiveness. For example, in extensive numerical experiments on network design and facility location problems, we find that (i) RPBD accelerates the convergence of Benders decomposition for problems with and without relatively complete recourse; (ii) RPBD usually halves the optimality gap at termination compared with a standard Benders approach; (iii) a random retention strategy is just as effective as problem-specific approaches proposed in the literature.

Key words: Mixed-integer optimization; Benders decomposition; Acceleration

1. Introduction

Mixed-integer optimization (Nemhauser and Wolsey 1999) is a powerful mathematical framework to solve problems involving discrete decisions. Despite orders of magnitude speed-ups in both hardware and algorithms over the last decades (Bixby 2012), the combinatorial nature of integer variables is an added difficulty when solving mixed-integer optimization problems at scale.

For large Mixed-Integer Linear Optimization (MILO) problems, Benders (1962) propose a decomposition scheme, known as Benders Decomposition (BD), relying on two components: On the modeling side, BD reformulates the MILO problem as a pure-integer convex optimization problem, where the convex objective function is defined by the optimization over all the continuous variables. On the algorithmic side, BD solves this pure-integer formulation by constructing a piecewise linear outer approximation of the convex cost function. The optimization problem with the piecewise linear approximation of the convex cost, called *the master problem*, is then refined iteratively by solving continuous optimization problems called the *separation problem*. This approach has then been generalized to problems with nonlinear objectives and constraints (Geoffrion 1972, Duran and Grossmann 1986). Among others, BD has been successful in solving large-scale instances of production planning (Adulyasak et al. 2015), facility location (Fischetti et al. 2017), or network design problems (Bertsimas et al. 2024), and is a central technique for large-scale problems.

In this work, we propose a simple, generic, yet highly effective method to accelerate BD, which we call Random Partial Benders Decomposition (RPBD). Our method challenges the modeling step in BD, formulating the master problem as a mixed-integer optimization problem (instead of a pure-integer one) by randomly retaining a subset of the continuous variables.

1.1. Problem Formulation

We consider a generic MILO problem of the following form

$$\min_{\boldsymbol{y}\in\mathcal{P}} \min_{\boldsymbol{x}_r, r\in\mathcal{R}} \quad \boldsymbol{d}^\top \boldsymbol{y} + \sum_{r\in\mathcal{R}} \boldsymbol{c}_r^\top \boldsymbol{x}_r \tag{1}$$

$$\text{s.t.} \quad \boldsymbol{A}_r \boldsymbol{x}_r + \boldsymbol{B}_r \boldsymbol{y} \geq \boldsymbol{b}_r, \quad \forall r \in \mathcal{R}, \tag{1a}$$

$$\boldsymbol{x}_r \in \mathbb{R}_+^{N_r}, \tag{1b}$$

with $\mathcal{P} := \{\boldsymbol{y} \in \{0,1\}^m | \boldsymbol{G}\boldsymbol{y} \geq \boldsymbol{g}\}$. Problem (1) is best understood as a two-stage problem (Shapiro et al. 2021, section 2) where a decision maker first makes a binary "design" decision $\boldsymbol{y}$ (e.g., constructing edges in a network), followed by second-stage continuous decisions $\boldsymbol{x}_r$ (e.g., flow in the network). For a fixed $\boldsymbol{y}$, we assume that the problem over the continuous variables $\boldsymbol{x}_r$ can be separated into $|\mathcal{R}|$ independent subproblems. Note that each $\boldsymbol{x}_r$ satisfies linear constraints (1a) that could involve the first-stage decision $\boldsymbol{y}$.

The structure of Problem (1) is present in all two-stage stochastic optimization problems when the second-stage average cost is estimated via sample average approximation, such as the Multi-commodity Capacitated Network Design (MCND) problem studied in Crainic et al. (2021).

Example 1 (Multi-commodity Capacitated Network Design). *A network is described as a directed graph $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of all potential edges. Commodities need to be shipped on the network between demand and supply nodes. The goal is to construct (capacitated) edges in the network to minimize the construction cost plus the average flow transportation cost. Formally, $y_{i,j} \in \{0,1\}$ denotes the decision of constructing edge (i,j). The construction cost and capacity of edge (i,j) are given and denoted $f_{i,j}$ and $u_{i,j}$, respectively. Let $\mathcal{K}$ denote the set of commodities and $c_{i,j}^k$, $k \in \mathcal{K}$, $(i,j) \in \mathcal{E}$, the unit transportation cost. Finally, we are given a set of scenarios $\boldsymbol{d}^{k,r}, r \in \mathcal{R}$, $k \in \mathcal{K}$, each of them associated with a probability p_r. The MCND problem consists of finding the construction decision $\boldsymbol{y}$ that minimizes the sum of the construction cost and the average (across scenarios) of the total transportation cost, i.e.,*

$$\min_{\boldsymbol{y}} \quad \min_{\boldsymbol{x}^{k,r}} \quad \sum_{(i,j)\in\mathcal{E}} f_{i,j} y_{i,j} + \sum_{r\in\mathcal{R}} p_r \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{E}} c_{i,j}^k x_{i,j}^{k,r} \tag{2}$$

$$s.t. \quad \boldsymbol{A}\boldsymbol{x}^{k,r} = \boldsymbol{d}^{k,r}, \quad \forall k \in \mathcal{K}, r \in \mathcal{R}, \tag{2a}$$

$$\sum_{k\in\mathcal{K}} x_{i,j}^{k,r} \leq u_{i,j} y_{i,j}, \quad \forall (i,j) \in \mathcal{E}, r \in \mathcal{R}, \tag{2b}$$

$$x_{i,j}^{k,r} \geq 0, y_{i,j} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{E}, k \in \mathcal{K}, r \in \mathcal{R}. \tag{2c}$$

Constraint (2a) *ensures the difference between in and out-flow equals demand, with* $\boldsymbol{A}$ *the flow conservation matrix. Constraint* (2b) *guarantees that the flow on each edge does not exceed capacity and that flow only circulates on constructed edges.*

Moreover, the same separable structure as Problem (1) appears in other problems such as Uncapacitated Facility Location (UFL; Farahani and Hekmatfar 2009) and Maximum Coverage Facility Location (MCFL; Megiddo et al. 1983), which we formally describe in Appendix, Section A.

1.2. Benders Decomposition and Literature Review

We now describe the key modeling and algorithmic ideas behind BD, before describing the literature on accelerated BD to which we contribute.

From a modeling perspective, BD reformulates Problem (1) by partially minimizing with respect to (or projecting out) the continuous variables $\boldsymbol{x}$, leading to:

$$\min_{\boldsymbol{y}\in\mathcal{P}} \quad \boldsymbol{d}^\top \boldsymbol{y} + \sum_{r\in\mathcal{R}} \phi_r(\boldsymbol{y}), \tag{3}$$

with

$$\phi_r(\boldsymbol{y}) := \min_{\boldsymbol{x_r}\geq 0} \left\{ \boldsymbol{c}_r^\top \boldsymbol{x}_r \,\middle|\, \boldsymbol{A}_r \boldsymbol{x}_r + \boldsymbol{B}_r \boldsymbol{y} \geq \boldsymbol{b}_r \right\} = \max_{\boldsymbol{p}_r \geq 0} \left\{ (\boldsymbol{b}_r - \boldsymbol{B}_r \boldsymbol{y})^\top \boldsymbol{p}_r \,\middle|\, \boldsymbol{A}_r^\top \boldsymbol{p}_r \leq \boldsymbol{c}_r \right\}, \tag{4}$$

where the equality follows from strong linear duality. Here, each $\phi_r(\boldsymbol{y})$ is convex as a point-wise maximum of affine functions. From an algorithmic perspective, BD proceeds by solving a master problem, obtained by replacing the term $\sum_r \phi_r(\boldsymbol{y})$ in (3) by a piecewise linear lower approximation, and iteratively refining it by evaluating $\sum_r \phi_r$ at the incumbent solution and deriving a new linear constraint for the master problem (see pseudo-code Algorithm B.1 in Section B).

For a given vector $\tilde{\boldsymbol{y}}$ and $r \in \mathcal{R}$, one of two situations can occur when evaluating $\phi_r(\tilde{\boldsymbol{y}})$. Either $\phi_r(\tilde{\boldsymbol{y}}) < +\infty$ or $\phi_r(\tilde{\boldsymbol{y}}) = +\infty$. If $\phi_r(\tilde{\boldsymbol{y}}) < +\infty$, taking the optimal dual solution $\boldsymbol{p}_r^\star(\tilde{\boldsymbol{y}})$ in (4) provides a linear lower approximation on $\phi_r(\tilde{\boldsymbol{y}})$: $\phi_r(\boldsymbol{y}) \geq (\boldsymbol{b}_r - \boldsymbol{B}_r \boldsymbol{y})^\top \boldsymbol{p}_r^\star(\tilde{\boldsymbol{y}})$, which we refer to as an optimality cut. By construction, this approximation is tight for $\boldsymbol{y} = \tilde{\boldsymbol{y}}$. Otherwise, $\phi_r(\tilde{\boldsymbol{y}}) = +\infty$, i.e., the minimization problem in (4) is infeasible and its dual is unbounded. In this case, we can find an

extreme ray $\boldsymbol{q}_r^\star(\tilde{\boldsymbol{y}}) \geq 0 : \boldsymbol{A}_r^\top \boldsymbol{q}_r^\star(\tilde{\boldsymbol{y}}) \leq \boldsymbol{0}$ such that $(\boldsymbol{b}_r - \boldsymbol{B}_r\tilde{\boldsymbol{y}})^\top \boldsymbol{q}_r^\star(\tilde{\boldsymbol{y}}) > 0$. Consequently, any solution $\boldsymbol{y}$ to (3) should satisfy $(\boldsymbol{b}_r - \boldsymbol{B}_r\boldsymbol{y})^\top \boldsymbol{q}_r^\star(\tilde{\boldsymbol{y}}) \leq 0$, which we refer to as a feasibility cut.

In its naive implementation (Algorithm B.1), BD solves a mixed-integer linear optimization problem at each iteration. Modern solvers, however, can implement BD in a branch-and-cut or single-tree fashion (Fortz and Poss 2009) using lazy callbacks, which is typically more efficient.

Our work contributes to the rich literature on accelerating Benders decomposition, either by accelerating the algorithmic part of BD (the outer-approximation scheme) or by investigating modeling decisions in BD that impact tractability.

On the algorithmic front, acceleration has been achieved either by reducing the time per iteration or by reducing the total number of iterations. Time per iteration can be reduced by generating optimality cuts that are valid but not tight, for example, by omitting hard constraints on $\boldsymbol{x}_r$ (e.g., Fischetti et al. 2016, for capacitated facility location) or generating feasible dual solutions via averaging (Bertsimas et al. 2024). The number of iterations can be reduced by using cuts that are deemed more informative. For example, adding cuts generated at fractional points $\tilde{\boldsymbol{y}}$ (e.g., McDaniel and Devine 1977, Fischetti et al. 2017) or selecting cuts with specific properties when multiple optimal dual variables exist (Magnanti and Wong 1981, Hosseini and Turner 2024).

On the modeling side, some choices impact the performance of BD. First, we can introduce one epigraph variable η to model $\sum_r \phi_r(\boldsymbol{y})$ (as described in Algorithm B.1), deriving one new optimality cut at each iteration (single-cut variant). Alternatively, we could introduce one epigraph variable η_r for each $\phi_r(\boldsymbol{y})$ and impose $|\mathcal{R}|$ new constraints per iteration (multi-cut variant). Second, BD reformulates the original problem (1) as (3) by projecting out all the continuous variables $\boldsymbol{x}_r$. The separable structure of Problem (1), however, allows for a partial decomposition, in which a subset of the continuous variables remains in the master problem. This is the solution we investigate in this paper.

To the best of our knowledge, partial Benders decomposition has been investigated in two prior studies. In multi-commodity network design problems (Example 1), the second stage variables $\boldsymbol{x}_r$ imply constraints on the first-stage binary decision $\boldsymbol{y}$, which leads BD to call the separation oracle at incumbent solutions $\tilde{\boldsymbol{y}}$ that are infeasible—we say that the problem does not satisfy the relatively complete recourse assumption. To mitigate the issue, Crainic et al. (2021) suggest partial BD as a way to impose more constraints on $\boldsymbol{y}$ and they select which continuous variables to retain in the master problem based on the constraints they induce. For maximum coverage facility location (a problem with complete recourse), Legault and Frejinger (2024) select which variables to retain

based on the magnitude of their second-stage cost. However, both works investigate the benefit of partial decomposition for specific optimization problems and using problem-specific retention strategies. In this work, we propose to randomly pick which continuous variables $\boldsymbol{x}_r$ to retain in the master problem, hence being problem-agnostic. We show that a random strategy recovers the performance of the methods developed by Crainic et al. (2021), Legault and Frejinger (2024) on the problems for which they have been designed, while being more broadly applicable.

REMARK 1. Partial decomposition resembles scenario reduction (Dupačová et al. 2003) in two-stage stochastic optimization. Scenario reduction replaces the set $\mathcal{R}$ by a subset $\tilde{\mathcal{R}}$ such that the MILO problem (1) with $\tilde{\mathcal{R}}$ produces a near-optimal solution to the original problem. In contrast, partial BD retains a subset $\mathcal{R}^{\texttt{mp}}$ of continuous variables in the master problem, but does not discard $\mathcal{R} \setminus \mathcal{R}^{\texttt{mp}}$. Consequently, it leads to an exact reformulation (and solution) of (1).

1.3. Structure and Contributions

Our contributions and the structure of the paper are organized as follows.

We describe our algorithm, random partial Benders decomposition (RPBD), in Section 2. In short, RPBD retains a subset of continuous variables $\mathcal{R}^{\texttt{mp}} \subseteq \mathcal{R}$ in the master problem, before applying BD, and $\mathcal{R}^{\texttt{mp}}$ is obtained via sampling without replacement.

Section 3 evaluates the numerical performance of RPBD on several classes of optimization problems. Our experiments present a few surprises: (i) RPBD is beneficial both on problems with and without complete recourse; (ii) RPBD achieves optimality gaps twice smaller than those of the best-performing method on larger instances; (iii) RPBD matches the performance of the deterministic retention strategies of Crainic et al. (2021), Legault and Frejinger (2024), on the problems for which they have been designed (network design and covering facility location respectively), while being more generically applicable; (iv) RPBD has a non-trivial impact on the behavior of the branch-and-bound solver and its resulting convergence. Among others, it leads to better lower bounds and more effective branching decisions.

Finally, to support our empirical observations, we present a theoretical analysis of the lower bounds obtained by RPBD in Section 4. We show that, even for small sets $\mathcal{R}^{\texttt{mp}}$, RPBD obtains lower bounds much tighter than a simple interpolation between the bounds of (1) and that of BD.

2. Method

In this section, we present our algorithm, Random Partial Benders Decomposition (RPBD), discuss hyperparameter calibration, and describe the existing partial BD methods from the literature.

2.1. Random Partial Benders Decomposition

The separability in Problem (1) allows us to keep a subset of the second-stage continuous variables in the master problem and partially minimize with respect to the other ones. Specifically, we partition the set of second-stage variables $\mathcal{R}$ into two, $\mathcal{R}^{\text{mp}}$ and $\mathcal{R} \setminus \mathcal{R}^{\text{mp}}$, and reformulate (1) as

$$\min_{\boldsymbol{y}\in\mathcal{P}} \min_{\boldsymbol{x}_r, r\in\mathcal{R}^{\text{mp}}} \left\{ \boldsymbol{d}^\top \boldsymbol{y} + \sum_{r\in\mathcal{R}^{\text{mp}}} \boldsymbol{c}_r^\top \boldsymbol{x}_r + \sum_{r\in\mathcal{R}\setminus\mathcal{R}^{\text{mp}}} \phi_r(\boldsymbol{y}) \;\middle|\; \boldsymbol{A}_r\boldsymbol{x}_r + \boldsymbol{B}_r\boldsymbol{y} \geq \boldsymbol{b}_r, \forall r \in \mathcal{R}^{\text{mp}} \right\}. \tag{5}$$

Our algorithm (see pseudo-code in Algorithm B.2) then follows the same outer-approximation procedure as BD but solving (5) instead of (3). Clearly, Problem (5) reduces to the standard BD formulation (3) when $\mathcal{R}^{\text{mp}} = \emptyset$ and reduces to the original MILO formulation (1) when $\mathcal{R}^{\text{mp}} = \mathcal{R}$.

The intuition for partial decomposition is as follows: The classical reformulation in BD (3) is very concise. It involves the $\boldsymbol{y}$ variable only and thus can be orders of magnitude smaller than the original MILO formulation. On the other hand, in the objective, it replaces a term that can be expressed linearly in $(\boldsymbol{y}, \boldsymbol{x}_r)$ by a convex function in $\boldsymbol{y}$, which is iteratively approximated by a piecewise linear function. Such an outer-approximation scheme can be inaccurate and inefficient, requiring many iterations to obtain a satisfying approximation. This is most notably the case for problems without relatively complete recourse like the network design problems studied in Crainic et al. (2021), i.e., problems where the continuous variables $\boldsymbol{x}_r$ induce constraints on $\boldsymbol{y}$ that are not already implied by $\boldsymbol{y} \in \mathcal{P}$. In this case, BD can waste a significant number of iterations exploring infeasible solutions $\boldsymbol{y} \in \mathcal{P}$. In general, partial decomposition helps by introducing a small number of second-stage variables to keep the size of the master problem under control, while improving the accuracy of the objective function in the master problem.

In addition, the performance of MILO solvers relies heavily on heuristics to identify feasible solutions or make branching decisions, which involve solving auxiliary problems. For example, local search solves an auxiliary problem with some variables fixed to certain values; strong branching solves multiple node Boolean relaxations to evaluate different branches. By eliminating the continuous variables, BD conceals some of the problem structure and can make these auxiliary problems uninformative, limiting the relevance of these heuristics. For this reason, retaining even a few continuous variables in the master problem can be beneficial. The subset $\mathcal{R}^{\text{mp}}$ provides a compressed representation of the problem and helps these heuristics recover their acceleration abilities.

Compared with previous attempts from the literature (Crainic et al. 2021, Legault and Frejinger 2024), we use a random strategy where $\mathcal{R}^{\text{mp}}$ is constructed by sampling a fraction α of the indices

in $\mathcal{R}$. Randomization makes our strategy agnostic to the problem considered. It also prevents us from introducing any bias (the variables introduced are uniformly sampled from $\mathcal{R}$), which gives us confidence that we provide the solver with a representative subset of the continuous variables.

2.2. Instance-Level Calibration of the Hyperparameter α

The only hyper-parameter in our algorithm is the fraction of continuous variables to introduce in the master problem, i.e., $\alpha = |\mathcal{R}^{\texttt{mp}}|/|\mathcal{R}|$. We propose a systematic procedure to calibrate its value for a given instance.

The first iteration of BD for (5) solves the reduced problem

$$\min_{\boldsymbol{y}\in\mathcal{P}} \min_{\boldsymbol{x}_r\geq 0, r\in\mathcal{R}^{\texttt{mp}}} \left\{ \boldsymbol{d}^\top \boldsymbol{y} + \sum_{r\in\mathcal{R}^{\texttt{mp}}} \boldsymbol{c}_r^\top \boldsymbol{x}_r \,\middle|\, \boldsymbol{A}_r\boldsymbol{x}_r + \boldsymbol{B}_r\boldsymbol{y} \geq \boldsymbol{b}_r, \forall r \in \mathcal{R}^{\texttt{mp}} \right\}. \tag{6}$$

Accordingly, we use the time needed to solve (6) as a criterion to calibrate α. Formally, we set a time limit for solving (6), typically as a fraction of the total time limit given for solving (1) (e.g., 5% of the total time limit in our implementation). We gradually increase α along a predefined grid of values (e.g., $\{0.01, 0.05, 0.1, 0.2, 0.5\}$ in our implementation) until the time needed to solve (6) exceeds the time limit.

This procedure is simple and provides a clear control over the time needed to calibrate α, as the total time is upper-bounded by the time threshold multiplied by the number of α values considered.

REMARK 2. The solutions $\boldsymbol{y}$ obtained by solving the reduced problems (6) could serve as warm starts when solving the full problem (1). However, to ensure that we evaluate the benefit of RPBD only, and not that of warm-starting, we do not implement this idea in our numerical experiments.

2.3. Selection Rules from the Literature

As points of comparison, we now describe the two selection rules proposed in the literature for multi-commodity network design and maximum coverage facility location, respectively.

Crainic et al. (2021) consider a multi-commodity network design problem without relatively complete recourse. In this setting, every set of continuous variables $\mathcal{R}^{\texttt{mp}} \subseteq \mathcal{R}$ induces a feasible set on $\boldsymbol{y}$: $\mathcal{P}^{\texttt{mp}}(\mathcal{R}^{\texttt{mp}}) := \{\boldsymbol{y} \in \mathcal{P} \mid \forall r \in \mathcal{R}^{\texttt{mp}}, \exists \boldsymbol{x}_r \in \mathbb{R}^{N_r} \text{ s.t. } \boldsymbol{A}_r\boldsymbol{x}_r + \boldsymbol{B}_r\boldsymbol{y} \geq \boldsymbol{b}_r\}$. In their terminology, they say that a scenario $r \in \mathcal{R}$ is covered by $\mathcal{R}^{\texttt{mp}}$ if it does not impose more constraints on $\boldsymbol{y}$ than those already induced by $\mathcal{R}^{\texttt{mp}}$, i.e., $\mathcal{P}^{\texttt{mp}}(\mathcal{R}^{\texttt{mp}}) \subseteq \mathcal{P}^{\texttt{mp}}(\{r\})$. Their *row covering* rule constructs $\mathcal{R}^{\texttt{mp}}$ as the set with the largest coverage, i.e.,

$$\mathcal{R}^{\texttt{mp}} \in \operatorname*{arg\,max}_{\substack{\mathcal{R}'\subseteq\mathcal{R} \\ |\mathcal{R}'|=\alpha|\mathcal{R}|}} \sum_{r\in\mathcal{R}\setminus\mathcal{R}'} \mathbb{I}\left[\mathcal{P}^{\texttt{mp}}(\mathcal{R}') \subseteq \mathcal{P}^{\texttt{mp}}(\{r\})\right].$$

We refer to Crainic et al. (2021, section 4.5) for details on how to solve the above maximization problem as a MILO. This approach has two drawbacks: First, solving the discrete optimization above can take a non-negligible amount of time. Second, it focuses on the induced feasible set and does not apply to problems with relatively complete recourse.

For the maximum coverage facility location problem, Legault and Frejinger (2024) propose a weight-based selection rule that retains in the master problem the continuous variables that contribute the most to the overall cost. For this problem, the variables x_r (and c_r) are scalars. Hence, they can rank $r \in \mathcal{R}$ by decreasing c_r values and include the top-$\alpha|\mathcal{R}|$ ones in $\mathcal{R}^{\texttt{mp}}$. Extensions of their approach to any problem of the form (1), where the costs $\boldsymbol{c}_r$ are vectors, however, is ambiguous.

3. Numerical Results

In this section, we evaluate the computational benefit of our RPBD approach on three problem classes: multi-commodity network design, uncapacitated facility location, and maximum coverage facility location. After describing the problem instances and our implementation (Section 3.1), we organize our findings in three key take-aways: RPBD significantly accelerates the classical BD algorithm, across a range of problem classes, including some with relatively complete recourse (Section 3.2); Our random sampling strategy for constructing $\mathcal{R}^{\texttt{mp}}$ is just as efficient as problem-specific methods (Crainic et al. 2021, Legault and Frejinger 2024) on the problems for which they have been designed (Section 3.3); RPBD has a deep impact on solver performance and behavior, including lower bounds, upper bounds, and branching decisions (Section 3.4).

3.1. Methodology

We evaluate our approach on three problem classes: multi-commodity network design (MCND), uncapacitated facility locations (UFL), and maximum coverage facility location (MCFL). We briefly describe the instances we use for each problem class here and provide a detailed description in Section C.1. For MCND, we consider the largest and hardest problems (R6.9, R8.9, R10.9) from a widely used library (the $\mathbf{R}^\star$ instances of Crainic et al. 2011). The largest size (R10.9) corresponds to a network with 20 nodes, 120 edges, and 40 commodities. For UFL, we use the instances from the $\mathbf{M}^\star$ dataset (Kratica et al. 2001) with the three largest sizes: $n = m = 300$, $n = m = 500$, and $n = m = 1000$, where n and m represent the number of customers and facility locations, respectively.[1] For MCFL, we generate instances of size $n = m = 4000$, $n = m = 6000$ and $n = m = 8000$ following the procedure of ReVelle et al. (2008), Cordeau et al. (2019).

[1] In general, 'square' facility location problems (with $n = m$) are considered harder because they contain the largest number of decision variables for a fixed total number of locations $n + m$.

In terms of algorithm, we compare our RPBD algorithm with two benchmarks: directly solving Problem (1) using CPLEX, which we refer to as MILO, and applying Benders decomposition to (3) directly (BD). We describe our implementation of Benders (single-cut version, fractional cuts at the beginning of the algorithm, one feasibility cut per iteration) in detail in Section C.2 and use the same configuration for BD and RPBD, to allow for a fair comparison. All algorithms are provided with the same initial feasible solution. Unless specified otherwise, the value of α for RPBD is calibrated using the approach described in Section 2.2.

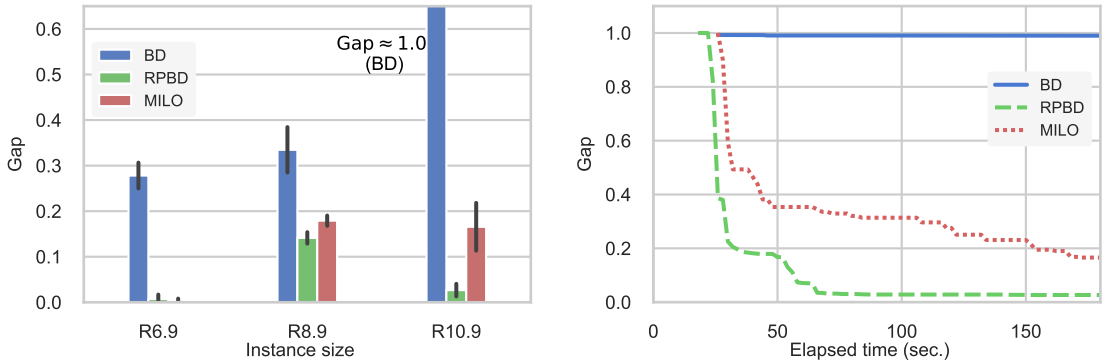
We conduct all experiments on a Windows laptop equipped with a 13th-Gen i7-1365U processor (1.80 GHz) and 16GB RAM, using `CPLEX-22.1.1` in a single-thread mode with all hyperparameters set to default. We use the Julia package `JuMP` (Lubin et al. 2023). All algorithms terminate when reaching an optimality gap of 10^{-6} or a `TIMELIMIT` set to 180 seconds (we report results with a one-hour time limit in Section C.6).

3.2. RPBD Significantly Accelerates BD

In this section, we demonstrate the benefit of RPBD on our three problem classes. We first present results on MCND, which is the problem that motivated Crainic et al. (2021). In particular, MCND does not satisfy relatively complete recourse, meaning that first-stage decisions $\boldsymbol{y} \in \mathcal{P}$ can be infeasible for the original problem (1) and that the separation oracle in BD can generate both optimality and feasibility cuts. It is widely believed that feasibility considerations are a prime bottleneck for BD schemes, which partial BD can alleviate. However, as we will show on UFL and MCFL instances, which satisfy relatively complete recourse, our RPBD approach provides substantial benefits even in the absence of any feasibility considerations.

Figure 1a reports the average optimality gap at termination of each method on MCND instances (grouped by size). We observe that BD performs worse than MILO on all the selected instances, with an average gap difference ranging from 0.14 to 0.84. The poor performance of BD relative to MILO is largely explained by feasibility issues. Regarding RPBD, its performance on the small and medium instances (< 0.01 on R6.9 and 0.14 on R8.9) is significantly better than BD (0.28 and 0.33 respectively) and comparable with MILO (< 0.01 and 0.18 respectively). On the largest instances (R10.9), RPBD is the clear winner, achieving 0.03 optimality gaps on average vs. 0.16 for MILO—an 80% reduction in optimality gap. In short, RPBD provides a significant acceleration over a classical BD implementation and achieves the best performance on the largest instances.

As a performance metric, the optimality gap at termination is censored by the chosen time limit. To evaluate the robustness of our findings with respect to the time limit, we report the complete gap



(a) Gap at termination (180-second time limit)

(b) Time vs. gap trade-off on R10.9.

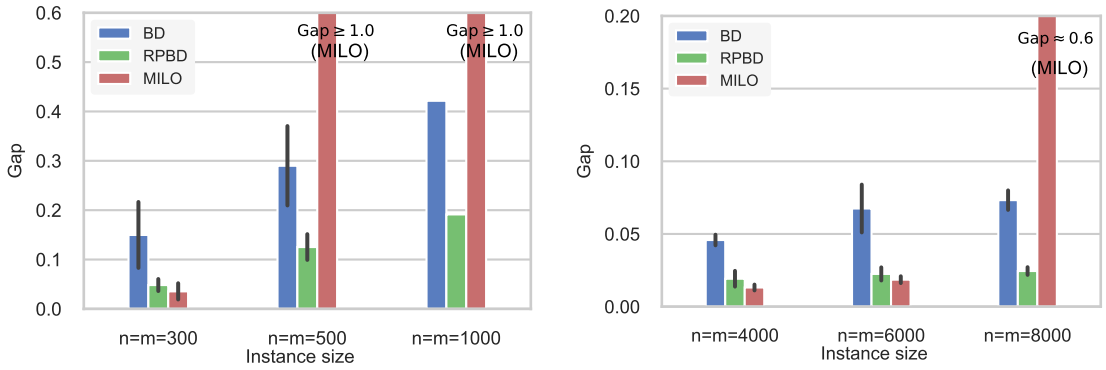
Figure 1 **Comparative performance of BD, RPBD, and MILO on MCND instances. (a) Optimality gap at termination (180-second time limit) for 15 MCND instances, grouped by size. Error bars represent standard deviations. (b) Time vs. gap Pareto curve on the R10.9 instances.**

versus time trade-off curve on the R10.9 instances in Figure 1b. Our key observation is that RPBD demonstrates significant and robust improvement for all time limits considered. In particular, MILO achieves an 18% gap after 180 seconds, while RPBD achieves the same gap within 50 seconds—a threefold reduction in time.

Next, we extend our evaluation to UFL and MCFL problems in Figure 2 (Pareto curves are reported in Section C.3). On the small-to-medium size problems ($n = m = 300$ for UFL, $n = m = 4000$ and $n = m = 6000$ for MCFL), we observe that RPBD matches the performance of the best benchmark (MILO in both cases). For larger problems, it provides a significant reduction in optimality gap at termination: For UFL instances, it achieves an optimality gap of 0.13 and 0.19 for $n = m = 500$ and $n = m = 1000$ respectively, vs. 0.29 and 0.42 for the second-best performing method (BD). For MCFL instances with $n = m = 8000$, it reaches an optimality gap three times lower than the best benchmark (0.02 vs. 0.07). Hence, our main observation from the MCND experiments is confirmed: RPBD significantly improves the performance of BD and is the best-performing method at a larger scale.

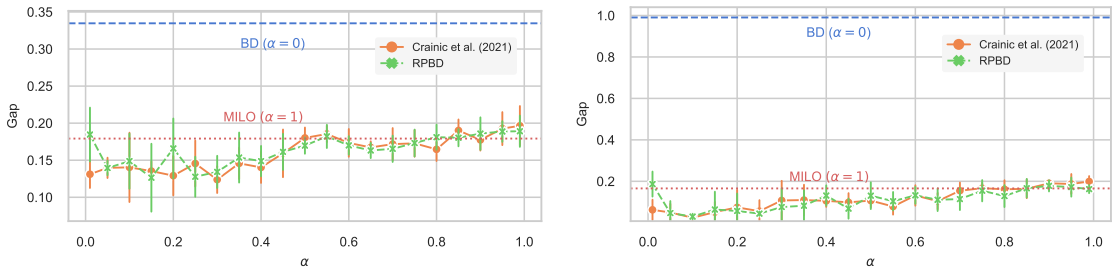
Remark 3. Observe that, unlike for MCND problems, BD performs better than MILO on the larger instances of UFL and MCFL. Indeed, these problems satisfy relatively complete recourse and constitute a friendly setting for Benders-like approaches. In particular, in MCFL, the functions $\phi_r(\boldsymbol{y})$ are piecewise linear with only two pieces. Hence, we can expect BD to be particularly efficient and outperform solving (1) directly (MILO; as shown in Cordeau et al. 2019).

Remark 4. Here, we report the performance of BD and RPBD with a single epigraph variable η (single-cut). Our conclusions hold for the multi-cut implementation as well; see Section C.4.



(a) UFL instances. (b) MCFL instances.

Figure 2 **Average optimality gaps at termination for BD, RPBD, and MILO, on UFL and MCFL instances. Error bars represent standard deviations. There is no error bar for UFL n=m=1000 because only one instance of this class is available in the benchmarking dataset.**



(a) R8.9 instances. (b) R10.9 instances.

Figure 3 **Gaps at termination of different partial BD strategies, namely RPBD and Crainic et al. (2021), for different values of α ($\alpha = \frac{|\mathcal{R}^{\mathtt{mp}}|}{|\mathcal{R}|}$) on (a) R8.9 instances and (b) R10.9 instances. Error bars represent standard deviations. Gaps achieved by BD and MILO are represented by horizontal lines to facilitate comparison.**

3.3. Random Selection Replicates the Benefit of Problem-Specific Selection Rules

In this section, we compare the random selection strategy in RPBD against the rules proposed by Crainic et al. (2021) and Legault and Frejinger (2024) for MCND and MCFL, respectively.

Figure 3 compares the gaps at termination of RPBD and Crainic et al. (2021), under different values of α. We observe that RPBD performs almost identically to Crainic et al. (2021) across all values of α, with comparable average gap and overlapping error bars (see Appendix C.5 for results of formal t-tests supporting this observation). Note that we did not include in the 3-minute time limit the precomputation time required to construct the set $\mathcal{R}^{\mathtt{mp}}$, which can be non-negligible for the method of Crainic et al. (2021). As a result, we view our estimate of the performance of Crainic et al. (2021) as conservative.

Table 1 Comparison of gap at termination (in %) between RPBD (random) and Legault and Frejinger (2024) (weight) for different values of α.

Problem	Size	$\alpha = 0.01$		$\alpha = 0.05$		$\alpha = 0.1$		$\alpha = 0.2$		$\alpha = 0.5$	
		random	weight	random	weight	random	weight	random	weight	random	weight
MCFL	n=m=4000	3.8(0.8)	4.1(0.4)	2.9(0.3)	2.5(0.5)	2.1(0.5)	2.1(0.2)	1.9(0.5)	1.7(0.3)	**1.4**(0.4)	**1.2**(0.3)
	n=m=6000	4.8(0.8)	5.1(0.6)	3.3(0.6)	2.9(0.5)	2.6(0.4)	2.2(0.4)	1.8(0.3)	1.5(0.3)	**1.4**(0.4)	**1.3**(0.4)
	n=m=8000	5.9(1.0)	5.4(1.0)	3.5(0.4)	2.6(0.1)	2.5(0.3)	1.9(0.3)	**1.8**(0.2)	**1.6**(0.3)	3.2(1.1)	1.6(0.4)

Standard deviations are provided in parentheses. The best gap for each method is highlighted in bold.

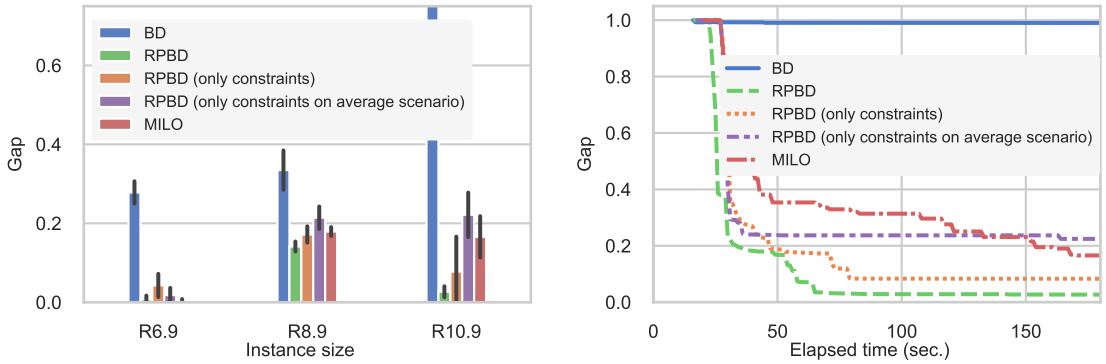
Furthermore, a surprising feature of Figure 3 is that RPBD provides a significant improvement over BD (more than 50% reduction in optimality gap) even with the smallest α ($\alpha = 0.01$), suggesting that RPBD is a reasonable strategy even for massive scale problems where we want $\alpha = o(n)$ to maintain tractability. We also observe that the performance of RPBD (and that of Crainic et al. 2021) is fairly robust with respect to the choice of α: The curve exhibits a mild U-shape and, except at $\alpha = 0$, a small variation in α does not result in drastically different performance.

Finally, Table 1 compares the performance of RPBD with the method of Legault and Frejinger (2024) on MCFL instances. Again, we observe that the performance of RPBD is similar to that of Legault and Frejinger (2024); see Appendix C.5 for t-tests supporting this observation.

3.4. Unpacking the Benefit: What Happens in the Solver

In this section, we focus on MCND instances to illustrate how RPBD impacts the behavior of the solver in terms of enforcing second-stage feasibility constraints, lower bound, quality of the incumbent $\tilde{\boldsymbol{y}}$, upper bound, and branching decisions.

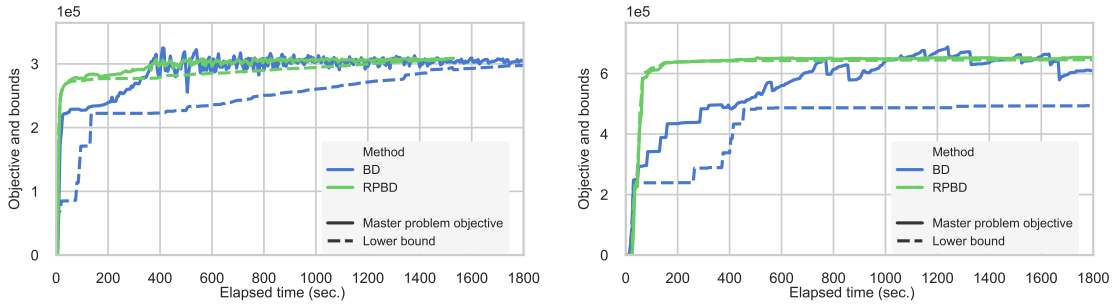
For MCND (a problem without relative complete recourse), the original motivation for partial BD is to impose second-stage feasibility constraints. To isolate the contribution of enforcing second-stage constraints, we now compare the performance of BD/RPBD to two benchmarks: One where we introduce some second-stage variables $\boldsymbol{x}_r, r \in \mathcal{R}^{\texttt{mp}}$ and second-stage constraints, but we keep the epigraph variable η_r and still use Benders cut to approximate the second-stage cost (instead of using $\boldsymbol{c}_r^\top \boldsymbol{x}_r$). A second where we introduce a second-stage variable $\bar{\boldsymbol{x}}$ associated with the average demand scenario. Results are reported in Figure 4. While their performance is comparable to RPBD on the small instances (and thus much better than BD), we observe that RPBD is more accurate on the largest instances, throughout the entire runtime, with a gap at termination of 2.5% compared to 15% on average for the two benchmarks, thereby highlighting that the value of RPBD is not limited to enforcing feasibility.



(a) Gap at termination (180-second time limit)

(b) Time vs. gap trade-off on R10.9.

Figure 4 **Comparative performance of BD, RPBD, two variants of RPBD, and MILO on MCND instances. (a) Optimality gap at termination (180-second time limit) for 15 MCND instances, grouped by size. Error bars represent standard deviations. (b) Time vs. gap Pareto curve on the R10.9 instances.**

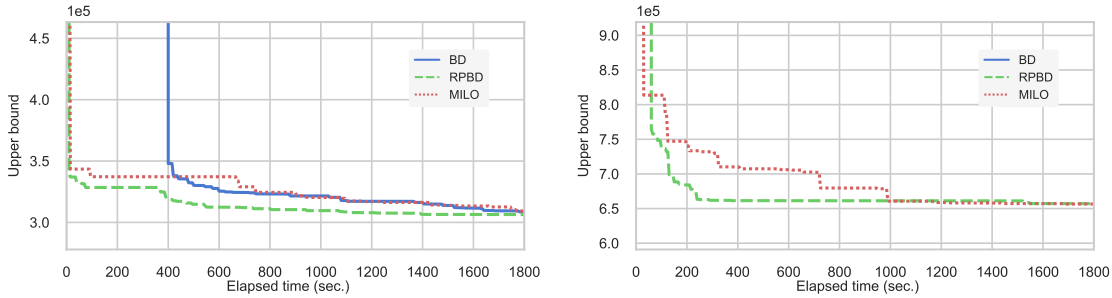


(a) R8.9 instances.

(b) R10.9 instances.

Figure 5 **Comparison of the master problem objective between RPBD and BD on MCND instances with `TIMELIMIT` set to 1800 seconds. The lower bound is plotted in dashed lines.**

RPBD also improves the approximation of the objective function. The lower bound primarily reflects the quality of the piecewise linear approximation of the objective function—although, after the root node, the global lower bound also depends on the branching decisions made. We compare the lower bounds obtained by BD and RPBD in Figure 5 (dashed lines). We observe that the lower bound obtained by RPBD is significantly tighter, and increases in a more rapid and smooth manner. In particular, BD achieves bounds after 30 minutes that RPBD meets or even largely exceeds in less than a minute. Another relevant metric is the quality of the incumbent $\tilde{\boldsymbol{y}}$ where the separation oracle is called. For BD and RPBD, we report (solid lines in Figure 5) the master problem objective $\boldsymbol{d}^\top \tilde{\boldsymbol{y}} + \sum_{r \in \mathcal{R}^{\texttt{mp}}} \boldsymbol{c}_r^\top \tilde{\boldsymbol{x}}_r + \tilde{\eta}$, which corresponds to the solver's current estimate of the objective value at $\tilde{\boldsymbol{y}}$ before calling the oracle. We observe that the master problem objective in RPBD (i) converges substantially faster than that of BD and (ii) is much more aligned with the global lower bound.



(a) R8.9 instances.

(b) R10.9 instances.

Figure 6 **Comparison of the upper bound by time among BD, RPBD and MILO on MCND instances. Note that BD fails to get a comparable upper bound in R10.9, so it does not appear in the plot (at the 1e7 magnitude).**

In particular, it does not exhibit the oscillating behavior of BD, suggesting that RPBD provides more accurate and reliable information to inform the various algorithmic decisions of the solver, leading to a less erratic search procedure. In Section C.7, we also observe that RPBD accelerates the convergence of Kelley's algorithm, i.e., outer approximation for solving the continuous relaxation of these problems.

On the other side, Figure 6 presents the upper bound (i.e., quality of the best solution found) over time. Recall that all algorithms are provided with the same warm start (the vector of all ones). Although BD obtains a valid lower bound quickly, we observe that it needs minutes to find a solution that materially improves upon the warm start (namely, that appears in the plot). This behavior could be due to two limitations: First, because of feasibility issues, BD can explore many infeasible incumbents $\tilde{\boldsymbol{y}}$ and waste time generating feasibility cuts. Second, by hiding second-stage variables, hence the problem structure, BD may hinder the performance of the solver's feasibility heuristics. In any case, we observe that RPBD overcomes these limitations and is able to recover solutions as good and even better than those found by MILO in the same period of time.

Finally, we show that RPBD has a deep impact on the behavior of the branch-and-bound algorithm by comparing the branching behavior in Figure 7. To do so, we use the `CPLEX-PYTHON` API instead of `JuMP`. However, this API is slower in code execution, so we consider smaller instances (R4.9) for this experiment. For each instance, we record the index $j \in \{1, \dots, m\}$ of the first branched variable y_j and report the histogram over multiple instances (5) and multiple runs of each algorithm (20 times per instance). We observe that BD makes very predictable branching decisions, branching first on y_1 in most cases. On the contrary, we observe more variability for MILO, so we can assume that MILO makes more instance-specific branching decisions because it has more information about the problem. In this regard, RPBD largely recovers the behavior of MILO. We posit that this

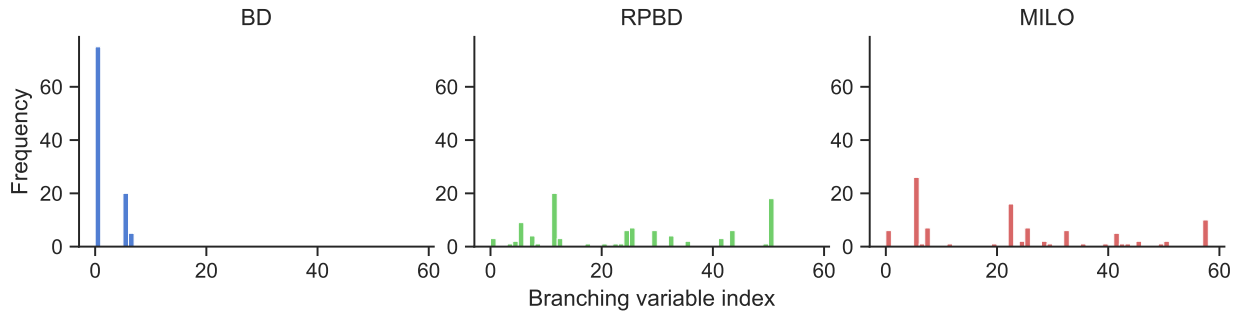


Figure 7 **Frequency of the first branching variable on R4.9 instances. We run each algorithm 20 times on each of the five instances and record the first branching variable indices.**

difference is due to the fact that the retained continuous variables $\boldsymbol{x}_r, r \in \mathcal{R}^{\text{mp}}$ provides sufficient information about the problem structure to make branching heuristics more effective.

4. Theoretical Analysis

In this section, we provide some theoretical results to explain the empirical performance of RPBD. In particular, to support the improvement in the lower bound observed in Section 3.4, we analyze the value of the relaxations obtained by RPBD compared with MILO and BD. However, we should acknowledge that our analysis will not be able to explain the deeper impact that RPBD has on the many heuristics and resolution decisions made by the solver.

4.1. Notations and Setting

We consider one iteration of the Benders decomposition scheme (Algorithm B.1). At this iteration, each convex term ϕ_r is approximated by a piecewise linear lower approximation $\hat{\phi}_r$. With these notations, the node relaxations obtained by the MILO and the BD approaches are

$$v^\star := \min_{\boldsymbol{y} \in \mathcal{P}_c} \sum_{r \in \mathcal{R}} \phi_r(\boldsymbol{y}) \quad \text{and } \hat{v}^\star := \min_{\boldsymbol{y} \in \mathcal{P}_c} \sum_{r \in \mathcal{R}} \hat{\phi}_r(\boldsymbol{y})$$

respectively, where $\mathcal{P}_c \subseteq \{\boldsymbol{y} \in [0,1]^m \ : \ \boldsymbol{G}\boldsymbol{y} \geq \boldsymbol{g}\}$ corresponds to the domain of the Boolean relaxation (we allow for some coordinates of $\boldsymbol{y}$ to be fixed). For concision, we omit the first stage-cost $\boldsymbol{d}^\top \boldsymbol{y}$, which can be subsumed in the definition of the functions ϕ_r and $\hat{\phi}_r$.

With these notations, we can express the lower bound obtained by RPBD by the value of the minimization problem

$$\hat{v}^\star(\boldsymbol{u}^\alpha) := \min_{\boldsymbol{y} \in \mathcal{P}_c} \sum_{r \in \mathcal{R}} (1 - u_r^\alpha)\hat{\phi}_r(\boldsymbol{y}) + u_r^\alpha \phi_r(\boldsymbol{y}),$$

where $\boldsymbol{u}^\alpha \in \{0,1\}^{\mathcal{R}}$ is a random variable encoding which indices $r \in \mathcal{R}$ are retained in the master problem, i.e., $u_r^\alpha = 1$ iff $r \in \mathcal{R}^{\text{mp}}$. Obviously, we have $\hat{v}^\star \le \hat{v}^\star(\boldsymbol{u}^\alpha) \le v^\star$. In other words, we know that the bound obtained by RPBD is at least as good as that of BD. The objective of this section is to quantify this improvement.

REMARK 5. In our analysis, if $r \notin \mathcal{R}^{\text{mp}}$ (i.e., $u_r^\alpha = 0$), RPBD approximates the function ϕ_r by $\hat{\phi}_r$, which is the same approximating function as the one used by BD. We do this simplification to allow for a theoretical comparison between RPBD and BD. In practice, however, RPBD and BD construct these piecewise linear approximations based on the sequence of incumbent solutions $\tilde{\boldsymbol{y}}$ where they call the separation oracle, which is algorithm-specific.

The main complication to analyze $\hat{v}^\star(\boldsymbol{u}^\alpha)$ is the fact that it is a random quantity, defined as the value of an optimization problem with a random objective function. To address this difficulty, we introduce a deterministic optimization problem, which we refer to as the expected counterpart, defined as the minimization of the average objective in RPBD, namely

$$\bar{v}(\alpha) := \min_{\boldsymbol{y} \in \mathcal{P}_c} \sum_{r \in \mathcal{R}} (1-\alpha)\hat{\phi}_r(\boldsymbol{y}) + \alpha \phi_r(\boldsymbol{y}).$$

The expected counterpart is not viable in practice: In terms of scalability, it is as large as the original MILO problem (1) because it involves all the terms $\phi_r(\boldsymbol{y}), r \in \mathcal{R}$. However, its lower bound obtained can only be weaker, $\bar{v}(\alpha) \le v^\star$. Nonetheless, its deterministic nature makes it more amenable to analysis. Our strategy is as follows: First, we show that $\mathbb{E}[\hat{v}^\star(\boldsymbol{u}^\alpha)]$ can be controlled by $\bar{v}(\alpha)$, and then we analyze the difference between $v^\star$ and $\bar{v}(\alpha)$.

4.2. Example

We illustrate our theoretical analysis with a simple network design example illustrated in Figure 8: We consider a network consisting of two nodes A and B, with two edges connecting them, with capacity 3 and 5 units, respectively. The decision $y \in \{0,1\}$ is whether to construct a third edge with a capacity of 10 units. We assume that all the demand scenarios are the same and equal to 10 units. Hence, all the functions $\phi_r, r \in \mathcal{R}$ are identical (we will simply denote them ϕ in the rest of this section). In this case, they are piecewise linear with three pieces as represented by the black line in Figure 8. The integer relaxation of (1) provides a lower bound $v^\star = 35$ achieved at $y^\star = 0.5$.

On the other hand, we consider the piecewise linear lower approximation $\hat{\phi}$ consisting of two pieces (blue line in Figure 8), obtained by calling the separation oracle at $\tilde{y} = 0$ and $\tilde{y} = 1$ respectively. With this approximation, BD provides a lower bound $\hat{v}^\star = 32$ achieved at $\hat{y}^\star = 0.2$.

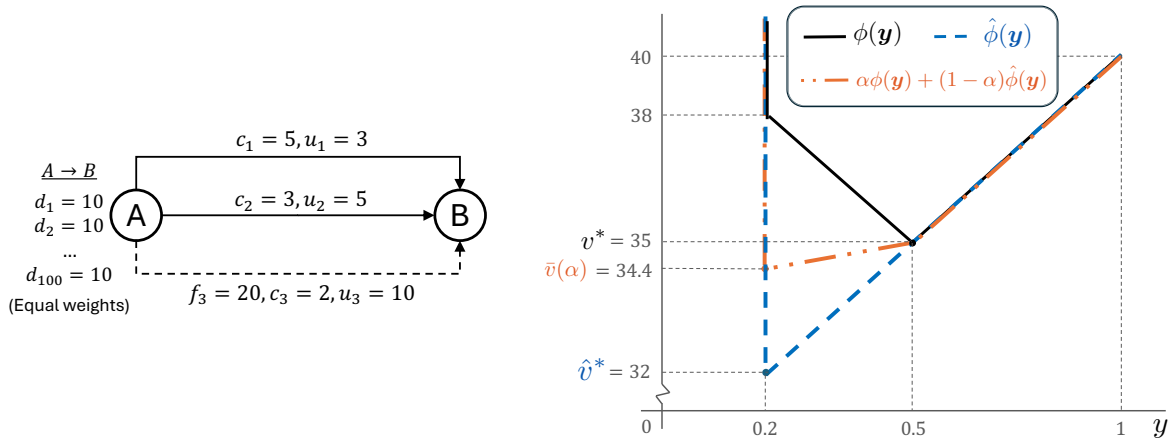


Figure 8 **Capacitated Network Design example (see Example 1 for notations) with a single commodity and 100 identical scenarios (demand to be shipped from node A to node B). The second-stage cost ϕ is estimated via two Benders cuts generated at $\tilde{y} = 0$ and $\tilde{y} = 1$. Here, $\alpha = 0.4$ is plotted for RPBD.**

Because all the functions ϕ_r are identical in this example, we have

$$\sum_{r\in\mathcal{R}}(1-u_r^\alpha)\hat{\phi}_r(\boldsymbol{y}) + u_r^\alpha\phi_r(\boldsymbol{y}) = \hat{\phi}(\boldsymbol{y})\sum_{r\in\mathcal{R}}(1-u_r^\alpha) + \phi(\boldsymbol{y})\sum_{r\in\mathcal{R}}u_r^\alpha = \hat{\phi}(\boldsymbol{y})|\mathcal{R}|(1-\alpha) + \phi(\boldsymbol{y})|\mathcal{R}|\alpha.$$

In other words, $\hat{v}^\star(\boldsymbol{u}^\alpha) = \bar{v}(\alpha)$ almost surely and the performance of RPBD is exactly the same as that of its expected counterpart. In general, these two optimization problems are different. However, we will prove that $\bar{v}(\alpha)$ generally provides a good approximation of $\hat{v}^\star(\boldsymbol{u}^\alpha)$, i.e., $\hat{v}^\star(\boldsymbol{u}^\alpha) \geq \bar{v}(\alpha) - O(\alpha(1-\alpha))$; see Proposition 1.

It is easy to prove that the expected counterpart provides a bound that is at least as good as the linear interpolation between MILO and BD, i.e., $\bar{v}(\alpha) \geq (1-\alpha)\hat{v}^\star + \alpha v^\star$. Our current example shows that the bound can be much tighter. Here, $\bar{v}(\alpha)$ is obtained by minimizing a piecewise linear function with three pieces. Varying α modifies the slope of the second piece and moves the minimum from $y = 0.2$ when $\alpha < 0.5$ to $y = 0.5$ when $\alpha > 0.5$. In particular, we have $\bar{v}(\alpha) = \min\{(1-2\alpha)\hat{v}^\star + 2\alpha v^\star, v^\star\}$. For $\alpha \leq 0.5$ for example, we have $\bar{v}(\alpha) = \hat{v}^\star + 2\alpha(v^\star - \hat{v}^\star)$, which can be much tighter than $\hat{v}^\star$. We will show (Proposition 2) that a similar relationship holds more generally.

4.3. Main Results

First, we control the difference between $\hat{v}^\star(\boldsymbol{u}^\alpha)$ and $\bar{v}(\alpha)$.

PROPOSITION 1. *Denote $\Delta_{\max}$ an upper bound on the approximation error at the RPBD solutions, i.e., $\Delta_{\max} = \max_r \max_{\boldsymbol{u}\in\{0,1\}^{\mathcal{R}}}(\phi_r - \hat{\phi}_r)(\boldsymbol{y}^\star(\boldsymbol{u}))$ where $\boldsymbol{y}^\star(\boldsymbol{u})$ denotes an optimal solution of $\hat{v}^\star(\boldsymbol{u})$. For any $\alpha \in [0,1]$, we have $\mathbb{E}[\hat{v}^\star(\boldsymbol{u}^\alpha)] \leq \bar{v}(\alpha)$ and $\bar{v}(\alpha) - \alpha(1-\alpha)\Delta_{\max}|\mathcal{R}| \leq \hat{v}^\star(\boldsymbol{u}^\alpha)$.*

Proof of Proposition 1 For the upper bound, evaluating the objective function in $\hat{v}^\star(\boldsymbol{u}^\alpha)$ at $\boldsymbol{y} = \bar{\boldsymbol{y}}(\alpha)$ an optimal solution of $\bar{v}(\alpha)$ yields $\hat{v}^\star(\boldsymbol{u}^\alpha) \le \sum_{r\in\mathcal{R}} (1-u_r^\alpha)\hat{\phi}_r(\bar{\boldsymbol{y}}(\alpha)) + u_r^\alpha \phi_r(\bar{\boldsymbol{y}}(\alpha))$. Taking expectations on both sides leads to $\mathbb{E}[\hat{v}^\star(\boldsymbol{u}^\alpha)] \le \bar{v}(\alpha)$.

Analogously, to obtain the lower bound, we evaluate the objective of $\bar{v}(\alpha)$ at an optimal solution of $\hat{v}^\star(\boldsymbol{u}^\alpha)$, $\boldsymbol{y}^\star(\boldsymbol{u}^\alpha)$:

$$\begin{aligned}
\bar{v}(\alpha) - \hat{v}^\star(\boldsymbol{u}^\alpha) &\le \left(\sum_{r\in\mathcal{R}} (1-\alpha)\hat{\phi}_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha)) + \alpha\phi_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha))\right) - \left(\sum_{r\in\mathcal{R}} (1-u_r^\alpha)\hat{\phi}_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha)) + u_r^\alpha\phi_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha))\right) \\
&= \sum_{r\in\mathcal{R}} (\alpha - u_r^\alpha)\left(\phi_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha)) - \hat{\phi}_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha))\right) \\
&= \sum_{r:u_r^\alpha=0} \alpha\left(\phi_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha)) - \hat{\phi}_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha))\right) + \sum_{r:u_r^\alpha=1} (\alpha-1)\left(\phi_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha)) - \hat{\phi}_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha))\right).
\end{aligned}$$

Recall that, for any $\boldsymbol{y}^\star(\boldsymbol{u})$ and any $r \in \mathcal{R}$, $0 \le \phi_r(\boldsymbol{y}^\star(\boldsymbol{u})) - \hat{\phi}_r(\boldsymbol{y}^\star(\boldsymbol{u}))$ because $\hat{\phi}_r$ is a lower approximation of ϕ_r, and $\phi_r(\boldsymbol{y}^\star(\boldsymbol{u})) - \hat{\phi}_r(\boldsymbol{y}^\star(\boldsymbol{u})) \le \Delta_{\max}$ by assumption. Hence,

$$\begin{aligned}
\sum_{r:u_r^\alpha=0} \alpha\left(\phi_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha)) - \hat{\phi}_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha))\right) &\le \alpha\Delta_{\max}|\{r \,:\, u_r^\alpha = 0\}| = \alpha\Delta_{\max}(1-\alpha)|\mathcal{R}|, \\
\sum_{r:u_r^\alpha=1} (\alpha-1)\left(\phi_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha)) - \hat{\phi}_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha))\right) &\le 0,
\end{aligned}$$

which concludes the proof. □

Proposition 1 bounds the difference between the average RPBD bound, $\mathbb{E}[\hat{v}^\star(\boldsymbol{u}^\alpha)]$, and $\bar{v}(\alpha)$ by an error term $\alpha(1-\alpha)\Delta_{\max}|\mathcal{R}|$, which captures the inherent randomness in constructing $\mathcal{R}^{\texttt{mp}}$. In particular, the error term vanishes when $\alpha = 0$ or 1, in which case $\bar{v}(\alpha) = \hat{v}^\star$ or $v^\star$ respectively. In practice, we are interested in small values of α so Proposition 1 supports the use of $\bar{v}(\alpha)$ as a good (conservative) proxy for $\hat{v}^\star(\boldsymbol{u}^\alpha)$ in this regime. For the interested reader, Section D derives concentration inequalities on $\hat{v}^\star(\boldsymbol{u}^\alpha) - \bar{v}(\alpha)$.

Remark 6. In theory, we can have $\Delta_{\max} = +\infty$ if a given $\boldsymbol{y}^\star(\boldsymbol{u})$ is infeasible for ϕ_r but feasible for $\hat{\phi}_r$. In practice, however, it is fair to assume that the cuts generated early in the algorithm ensure that $\phi_r(\boldsymbol{y}^\star(u))$ for the node relaxation solutions $\boldsymbol{y}^\star(u)$.

We now turn our attention to analyzing $\bar{v}(\alpha)$. Obviously, we have $(1-\alpha)\hat{v}^\star + \alpha v^\star \le \bar{v}(\alpha) \le v^\star$. However, we now show that the lower bound can be much tighter in general.

Proposition 2. *For any $\alpha \in [0,1]$, the following holds:*

$$(1-\alpha)\hat{v}^\star + \alpha v^\star + \rho(\alpha) \;\le\; \bar{v}(\alpha) \;\le\; v^\star,$$

where the function $\rho : [0,1] \to \mathbb{R}$ satisfies the following:

1. $\rho(\alpha)$ *is a piecewise linear, concave function;*
2. $\rho(\alpha) = 0$*, if* $\alpha = 0$ *or* 1*; and*
3. $\rho'(0) = \sum_{r\in\mathcal{R}} \phi_r(\hat{\boldsymbol{y}}) - v^\star \geq 0$ *is the sub-optimality of the relaxed solution of* $\hat{v}^\star$.

Proof of Proposition 2 Let $\bar{\boldsymbol{y}}(\alpha)$ denote an optimal solution to $\bar{v}(\alpha)$, and $\boldsymbol{y}^\star$ an optimal solution to $v^\star$. From $\hat{v}^\star \leq \sum_{r\in\mathcal{R}} \hat{\phi}_r(\bar{\boldsymbol{y}}(\alpha))$, we have

$$\bar{v}^\star(\alpha) - \left[(1-\alpha)\hat{v}^\star + \alpha v^\star\right] \geq \alpha \sum_{r\in\mathcal{R}} \left[\phi_r(\bar{\boldsymbol{y}}^\star(\alpha)) - \phi_r(\boldsymbol{y}^\star)\right] \geq 0.$$

Notice that $\bar{v}^\star(\alpha) = \min_{\boldsymbol{y}\in\mathcal{P}_c} (1-\alpha) \sum_{r\in\mathcal{R}} \hat{\phi}_r(\boldsymbol{y}) + \alpha \sum_{r\in\mathcal{R}} \phi_r(\boldsymbol{y})$ is a point-wise minimum of functions that are concave (indeed, linear) in α, hence is concave in α. As a result, we can say that $\bar{v}^\star(\alpha) - \left[(1-\alpha)\hat{v}^\star + \alpha v^\star\right] \geq \rho(\alpha)$ for $\rho : [0,1] \to \mathbb{R}$ the concave envelope of $\alpha \mapsto \alpha \sum_{r\in\mathcal{R}} \left[\phi_r(\bar{\boldsymbol{y}}^\star(\alpha)) - \phi_r(\boldsymbol{y}^\star)\right]$.

We now verify that the function ρ satisfies the three properties claimed: 1. The vector $\boldsymbol{y}^\star(\alpha)$ is an optimal solution of a linear optimization problem $(\bar{v}^\star(\alpha))$ whose objective is parametrized by α. Hence, $\boldsymbol{y}^\star(\alpha)$ is piecewise constant almost everywhere. Consequently, $\alpha \mapsto \alpha \sum_{r\in\mathcal{R}} \left[\phi_r(\bar{\boldsymbol{y}}^\star(\alpha)) - \phi_r(\boldsymbol{y}^\star)\right]$ is piecewise linear and ρ is piecewise linear as well. 2. For any $\alpha \in [0,1]$, $\bar{v}^\star(\alpha) - \left[(1-\alpha)\hat{v}^\star + \alpha v^\star\right] \geq \rho(\alpha) \geq 0$. For $\alpha \in \{0,1\}$, both the left- and the right-hand sides are equal to 0, so we must have $\rho(0) = \rho(1) = 0$. 3. For α in a neighborhood of 0^+, ρ is linear with $\rho(\alpha) = \alpha \sum_{r\in\mathcal{R}} \left[\phi_r(\bar{\boldsymbol{y}}^\star(0)) - \phi_r(\boldsymbol{y}^\star)\right]$ leading to $\rho'(0) = \sum_{r\in\mathcal{R}} \left[\phi_r(\bar{\boldsymbol{y}}^\star(0)) - \phi_r(\boldsymbol{y}^\star)\right] = \sum_{r\in\mathcal{R}} \phi_r(\bar{\boldsymbol{y}}^\star(0)) - v^\star$. □

Figure 9 provides a visualization of Proposition 2 (and some of its proof ingredients). Proposition 2 states that $\bar{v}(\alpha)$ is at least equal to the simple interpolation between $v^\star$ and $\hat{v}^\star$, $(1-\alpha)\hat{v}^\star + \alpha v^\star$, plus a concave term $\rho(\alpha)$. In particular, we can say that $\bar{v}(\alpha)$ provides a bound strictly tighter than $(1-\alpha)\hat{v}^\star + \alpha v^\star$ if $\rho'(0) > 0 \iff \sum_r \phi_r(\hat{\boldsymbol{y}}^\star) > v^\star$, i.e., if the relaxed solution found by Benders is sub-optimal for the MILO relaxation, which is often the case in practice. Notably, for small values of α, we have that $\bar{v}(\alpha) \geq \hat{v}^\star + \alpha(\sum_r \phi_r(\hat{\boldsymbol{y}}^\star) - \hat{v}^\star)$, highlighting that even small values of α can significantly improve the bounds obtained compared with BD, as in our numerical experiments.

5. Conclusion

In this paper, we introduce RPBD, a simple and generic approach that accelerates BD by randomly retaining a subset of continuous variables within the master problem. The benefit of RPBD is balancing the efficiency in solving the master problem and the accuracy of the outer approximation.

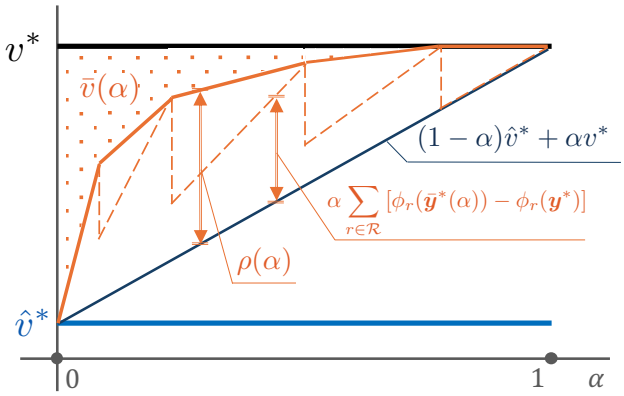


Figure 9 **Illustration of results in Proposition 2.**

In numerical experiments, we observe that RPBD outperforms BD on large-scale instances, achieving more than a 50% reduction in the optimality gap at termination. The effectiveness of RPBD demonstrates its potential for scaling MILO problems. Moreover, our theoretical analysis provides insights into how RPBD improves BD concerning the relaxation value, offering a framework for understanding the impact of partial decomposition.

From an application point of view, future work could investigate the applicability and efficiency of RPBD to problems without separability, although we suspect non-random retention strategies would be more powerful in these contexts. We also believe that RPBD offers an inexpensive way to strengthen BD for two-stage optimization problems with mixed-integer recourse. Indeed, for these problems, BD requires to relax the integrality constraints on the second-stage variables, hence providing a lower approximation of second-stage costs. Instead, we could retain some second-stage mixed-integer variables in the master problem and apply BD to a subset of second-stage variables. On preliminary experiments (Section E), we observe that RPBD can reduce the approximation error by a factor of five and find solutions two to three times better than BD. As such, we believe RPBD could be an easy-to-implement way to extend the scope of instances we can solve using Benders approximation, before resorting to more sophisticated approaches like Lagrangian cuts (Chen and Luedtke 2022).

On the theory side, a more comprehensive theoretical understanding of the impact of RPBD (in particular, its impact on branching or presolving) constitutes an interesting future direction. More broadly, we believe that the surprising performance of RPBD underscores the inefficiency in the outer-approximation process inherent to Benders decomposition. Designing algorithms that leverage the compact optimization formulation of Benders but solve it more efficiently, e.g., using

other convex optimization methods, could further enhance our ability to solve discrete optimization problems at scale.

References

Adulyasak Y, Cordeau JF, Jans R (2015) Benders decomposition for production routing under demand uncertainty. *Operations Research* 63(4):851–867.

Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4(1):238–252.

Bertsimas D, Cory-Wright R, Pauphilet J, Petridis P (2024) A stochastic Benders decomposition scheme for large-scale stochastic network design. *INFORMS Journal on Computing* .

Bixby RE (2012) A brief history of linear and mixed-integer programming computation. *Documenta Mathematica* 2012:107–121.

Chen R, Luedtke J (2022) On generating lagrangian cuts for two-stage stochastic integer programs. *INFORMS Journal on Computing* 34(4):2332–2349.

Cordeau JF, Furini F, Ljubić I (2019) Benders decomposition for very large scale partial set covering and maximal covering location problems. *European Journal of Operational Research* 275(3):882–896.

Crainic TG, Fu X, Gendreau M, Rei W, Wallace SW (2011) Progressive hedging-based metaheuristics for stochastic network design. *Networks* 58(2):114–124.

Crainic TG, Hewitt M, Maggioni F, Rei W (2021) Partial Benders decomposition: general methodology and application to stochastic network design. *Transportation Science* 55(2):414–435.

Dupačová J, Gröwe-Kuska N, Römisch W (2003) Scenario reduction in stochastic programming. *Mathematical Programming* 95:493–511.

Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming* 36:307–339.

Farahani RZ, Hekmatfar M (2009) *Facility Location: Concepts, Models, Algorithms and Case Studies* (Springer Science & Business Media).

Fischetti M, Ljubić I, Sinnl M (2016) Benders decomposition without separability: A computational study for capacitated facility location problems. *European Journal of Operational Research* 253(3):557–569.

Fischetti M, Ljubić I, Sinnl M (2017) Redesigning Benders decomposition for large-scale facility location. *Management Science* 63(7):2146–2162.

Fortz B, Poss M (2009) An improved Benders decomposition applied to a multi-layer network design problem. *Operations Research Letters* 37(5):359–364.

Geoffrion AM (1972) Generalized Benders decomposition. *Journal of Optimization Theory and Applications* 10:237–260.

Hosseini M, Turner J (2024) Deepest cuts for Benders decomposition. *Operations Research* .

Kratica J, Tošic D, Filipović V, Ljubić I (2001) Solving the simple plant location problem by genetic algorithm. *RAIRO-Operations Research* 35(1):127–142.

Legault R, Frejinger E (2024) A model-free approach for solving choice-based competitive facility location problems using simulation and submodularity. *INFORMS Journal on Computing* .

Lubin M, Dowson O, Garcia JD, Huchette J, Legat B, Vielma JP (2023) Jump 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation* 15(3):581–589.

Magnanti TL, Wong RT (1981) Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research* 29(3):464–484.

McDaniel D, Devine M (1977) A modified Benders' partitioning algorithm for mixed integer programming. *Management Science* 24(3):312–319.

Megiddo N, Zemel E, Hakimi SL (1983) The maximum coverage location problem. *SIAM Journal on Algebraic Discrete Methods* 4(2):253–261.

Nemhauser GL, Wolsey LA (1999) *Integer and Combinatorial Optimization* (John Wiley & Sons).

Rahmaniani R, Crainic TG, Gendreau M, Rei W (2017) The Benders decomposition algorithm: A literature review. *European Journal of Operational Research* 259(3):801–817.

Rahmaniani R, Crainic TG, Gendreau M, Rei W (2018) Accelerating the Benders decomposition method: Application to stochastic network design problems. *SIAM Journal on Optimization* 28(1):875–903.

ReVelle C, Scholssberg M, Williams J (2008) Solving the maximal covering location problem with heuristic concentration. *Computers & Operations Research* 35(2):427–435.

Serfling RJ (1974) Probability inequalities for the sum in sampling without replacement. *The Annals of Statistics* 39–48.

Shapiro A, Dentcheva D, Ruszczynski A (2021) *Lectures on Stochastic Programming: Modeling and Theory* (SIAM).

Appendix A: Omitted Problem Formulations

We provide the MILO formulation for the uncapacitated and maximum coverage facility location problems.

A.1. Uncapacitated Facility Location (UFL)

We consider the problem of locating facilities ($y_j = 1$ if a facility at location j is opened, 0 otherwise) in order to serve customer demand ($x_{ij} \in [0,1]$ corresponds to the fraction of demand of customer i satisfied by facility j. The objective is to minimize construction cost and transportation cost by solving

$$\min_{\boldsymbol{y}\in\{0,1\}^m} \min_{\boldsymbol{x}\in[0,1]^{n\times m}} \quad \sum_{j=1}^{m} f_j y_j + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \quad \text{s.t. } \sum_{j=1}^{m} x_{ij} = 1, \quad \forall i \in \{1,\dots,n\},$$
$$x_{ij} \le y_j, \quad \forall i \in \{1,\dots,n\} I, \forall j \in \{1,\dots,m\}.$$

This problem is of the form (1) with $\mathcal{R} = \{1,\dots,n\}$ and $\boldsymbol{x}_r = (x_{ij})_{j=1,\dots,m}$. In other words, for a fixed set of open facilities $\boldsymbol{y}$, the assignment problem is separable across customers.

A.2. Maximum Coverage Facility Location (MCFL)

Like in the UFL problem, we consider facility and customer locations. Each customer location $i \in \{1,\dots,n\}$ can be covered by a fixed subset of facilities $\mathcal{J}(i) \subseteq \{1,\dots,m\}$ (e.g., the facilities that are within an x-km radius) and is assigned a weight v_i. The goal is to open at most B facilities in order to maximize the total customer value covered:

$$\max_{\boldsymbol{y}\in\{0,1\}^m} \max_{\boldsymbol{x}\in[0,1]^n} \quad \sum_{i=1}^{n} v_i x_i \quad \text{s.t. } x_i \le \sum_{j\in\mathcal{J}(i)} y_j, \forall i \in \{1,\dots,n\},$$
$$\sum_{j=1}^{m} y_j \le B.$$

Appendix B: Pseudo-code

B.1. Benders Decomposition

Algorithm B.1: Benders Decomposition

set: $C^{\texttt{opt}} \leftarrow \emptyset$, $C^{\texttt{feas}} \leftarrow \emptyset$, $LB \leftarrow -\infty$, $UB \leftarrow +\infty$

define:

$$\min_{\boldsymbol{y}\in\mathcal{P},\,\eta\geq 0} \boldsymbol{d}^\top \boldsymbol{y} + \eta \quad \text{s.t.} \quad \eta \geq \sum_{r\in\mathcal{R}} (\boldsymbol{b}_r - \boldsymbol{B}_r \boldsymbol{y})^\top \boldsymbol{p}_r^t, \quad \forall \boldsymbol{p}^t \in C^{\texttt{opt}}, \qquad 0 \geq (\boldsymbol{b}_r - \boldsymbol{B}_r \boldsymbol{y})^\top \boldsymbol{q}_r^t, \quad \forall (r, \boldsymbol{q}_r^t) \in C^{\texttt{feas}}. \tag{MP}$$

repeat
- find $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{\eta}})$ optimal solution of (MP)
- update $LB \leftarrow \boldsymbol{d}^\top \tilde{\boldsymbol{y}} + \tilde{\eta}$
- evaluate $\sum_r \phi_r(\tilde{\boldsymbol{y}})$
- **if** $\sum_r \phi_r(\tilde{\boldsymbol{y}}) < +\infty$ **then**
  - $C^{\texttt{opt}} \leftarrow C^{\texttt{opt}} \cup \{ (\boldsymbol{p}_r^\star(\tilde{\boldsymbol{y}}))_{r\in\mathcal{R}} \}$
  - update $UB \leftarrow \min(UB, \boldsymbol{d}^\top \tilde{\boldsymbol{y}} + \sum_r \phi_r(\tilde{\boldsymbol{y}}))$
- **else**
  - find r s.t. $\phi_r(\tilde{\boldsymbol{y}}) = +\infty$
  - $C^{\texttt{feas}} \leftarrow C^{\texttt{feas}} \cup \{ (r, \boldsymbol{q}_r^\star(\tilde{\boldsymbol{y}})) \}$
- **end**

until $|UB - LB| \leq \epsilon\, UB$

B.2. Random Partial Benders Decomposition

Algorithm B.2: Random Partial Benders Decomposition (RPBD)

Input: α

Sample without replacement $\mathcal{R}^{\texttt{mp}} \subseteq \mathcal{R}$ such that $|\mathcal{R}^{\texttt{mp}}| = \alpha|\mathcal{R}|$

Run BD (Algorithm B.1) to solve (5) instead of (3)

Appendix C: Supplement to the Numerical Results

C.1. Methodology: Instances

Multi-Commodity Network Design (MCND) We use the popular $\mathbf{R}^\star$ instances (Crainic et al. 2011) for the MCND problem, considering three sizes of instances: R6.9, R8.9 and R10.9. Each instance size (e.g., R10.9) is defined by a major (e.g., 10) followed by a minor (e.g., 9) number. The major number characterizes the network structure (number of nodes, edges, commodities), and the minor number corresponds to different values for demand, edge capacities, and costs. Although not in an absolute manner, a greater major number represents a larger network and a greater minor a problem harder. Since we are mostly interested in solving hard large-scale instances, we focus on the three larger major classes (6, 8, and 10) and the hardest minor class (9). The largest size, R10.9, corresponds to a network with up to 20 nodes, 120 edges, and 40 commodities. See Crainic et al. (2011, table 1) for detailed attributes of these instances.

The $\mathbf{R}^\star$ instances were originally proposed for MCND without demand uncertainty. For the stochastic version of MCND, Rahmaniani et al. (2018) provides a data set of 1000 demand scenarios for each problem size. For each of our problem size (R6.9, R8.9, R10.9), we generate five instances by randomly sampling 256 different scenarios from those

1000. Lastly, we generate i.i.d. scenario weights p_r uniformly from $\{1, 2, \ldots, 100\}$ followed by a normalization. We vary the scenario weights to make the benchmarking method Legault and Frejinger (2024) directly applicable.

In our experiments, we warm-start all algorithms by setting $\boldsymbol{y}$ to the vector of all ones, i.e., constructing all the edges. We also use basic valid inequalities regarding network connectivity (Rahmaniani et al. 2018, section 4.1.3): $\forall j \in \mathcal{V}, \sum_{(i,j)\in\mathcal{E}} y_{i,j} \geq \max_{r\in\mathcal{R}} \max_{k\in\mathcal{K}} \mathbb{I}\{d_j^{k,r} > 0 \text{ OR} d_j^{k,r} < 0\}$, namely, at least one edge to each destination node (one edge from each origin node) should be activated.

Uncapacitated Facility Location (UFL) We use the instances from the classic $\mathbf{M}^\star$ dataset (Kratica et al. 2001) on the three largest scales: $n = m = 300$ (5 instances), $n = m = 500$ (5 instances), and $n = m = 1000$ (1 instance), where n and m represent the number of customers and facility locations, respectively. We apply a valid inequality requiring at least two facilities to be opened (Fischetti et al. 2017), $\sum_{j=1}^{m} y_j \geq 2$, which ensures relatively complete recourse, and use a warm start made of randomly opening two facilities for all methods.

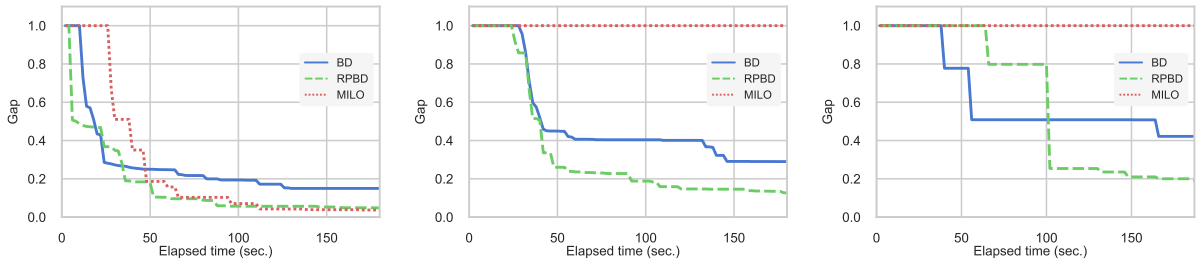
Maximum Coverage Facility Location (MCFL) We generate synthetic instances of size $n = m = 4000$, $n = m = 6000$ and $n = m = 8000$ using the same procedures as ReVelle et al. (2008), Cordeau et al. (2019). The 2D-coordinates of facility and customer locations are independently sampled from $\mathcal{U}[0, 30]$, For each customer i its weight or value is sampled as $v_i \sim \mathcal{U}[0, 100]$ and rounded to the nearest integer. The budget B is fixed to 15 facilities, and customers are considered as "covered" if their ℓ_2-distance to an opened facility is lower than 4.5. For each instance size, we generate 5 instances. We warm-start all algorithms by randomly opening $B = 15$ facilities. Note that MCFL satisfies the relatively complete recourse condition.

C.2. Methodology: Algorithms

We implement a Benders decomposition scheme in a single-tree fashion (using callbacks in CPLEX) in the following way: The separation problem is called at integer solutions (`CPX_CALLBACKCONTEXT_CANDIDATE` in CPLEX) via lazy callbacks or fractional solutions to the node relaxation (`CPX_CALLBACKCONTEXT_RELAXATION`) via user callbacks. In accordance with recommendations that fractional cuts are more beneficial early during the optimization process (Rahmaniani et al. 2017, section 5), we reduce the frequency of calling the user callback by half when 20% of the time limit is elapsed. We use a single epigraph variable η (single-cut) to model $\sum_{r\in\mathcal{R}} \phi_r(\boldsymbol{y})$ or $\sum_{r\in\mathcal{R}^{\text{mp}}} \phi_r(\boldsymbol{y})$ for BD and RPBD respectively. If the incumbent $\tilde{\boldsymbol{y}}$ is infeasible, we add one feasibility cut corresponding to the first (with the smaller index) infinite $\phi_r(\tilde{\boldsymbol{y}}$, motivated by the observation that feasibility cuts from different scenarios r at the same $\tilde{\boldsymbol{y}}$ are often similar (Rahmaniani et al. 2018).

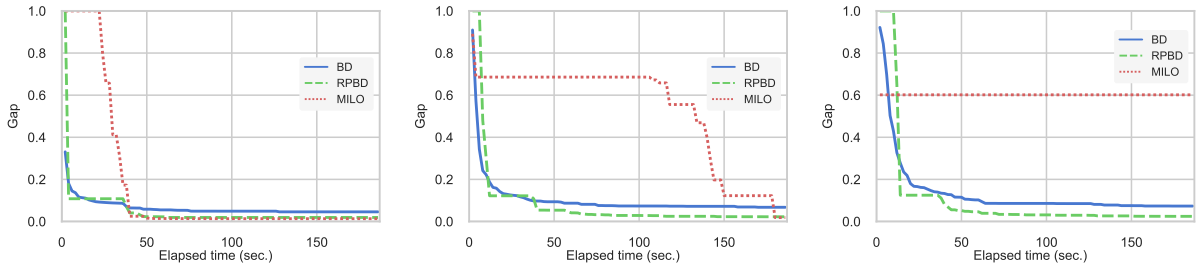
C.3. Time Versus Gap Comparison on UFL and MCFL Instances

Figures 2a and 2b compare the optimality gaps at termination (3-minute time limit) on the UFL and MCFL instances, respectively. To provide a more complete description of the relative performance of our method (RPBD) compared with solving the MILO formulation directly or applying Benders decomposition (BD), Figures C.1 and C.2 report the evolution of the gap over time for each method on the same instances.



(a) $n = m = 300$. (b) $n = m = 500$. (c) $n = m = 1000$

Figure C.1 Time vs. optimality gap trade-off curve for BD, RPBD, and MILO, on the UFL instances.



(a) $n = m = 4000$. (b) $n = m = 6000$. (c) $n = m = 8000$.

Figure C.2 Time vs. optimality gap trade-off curve for BD, RPBD, and MILO, on the MCFL instances.

C.4. Comparison Between Single-cut and Multi-cut

In Table C.1, we provide a comparison between Multi-cut and single-cut (RP)BD on three classes of instances. In the multi-cut version, there is one epigraph variable η_r for each scenario $r \in \mathcal{R} \setminus \mathcal{R}^{\text{mp}}$. Instead of adding only one cut at each iteration, we follow the common implementation of multi-cut BD, adding $|\mathcal{R}|$ (one for each scenario) cuts at each iteration. We keep all other components the same as those in single-cut (RP)BD.

We observe in Table C.1 that both single-cut and multi-cut (RP)BD do not dominate the other, aligned with the practical experience that the choice between them is highly problem-specific. Nonetheless, the gaps are improved when switching from BD to RPBD for all classes of instances. This demonstrates that the benefit of RPBD does not rely on the single-cut version we adopt in the main experiments.

Table C.1 Comparison of final gaps (%) between single-cut and multi-cut versions of BD and RPBD.

Problem	Instance	Single-cut		Multi-cut	
		BD	RPBD	BD	RPBD
MCND	R6.9	27.82(2.84)	**0.71**(0.98)	15.64(13.11)	1.51(3.37)
	R8.9	33.47(4.99)	14.13(1.27)	44.21(21.71)	**13.25**(3.29)
	R10.9	99.05(0.42)	**2.67**(1.38)	80.05(1.02)	12.28(9.98)
UFL	n=m=300	14.98(6.68)	4.83(1.22)	1.08(2.41)	**0.98**(1.83)
	n=m=500	29.00(8.04)	12.53(2.61)	11.46(8.53)	**7.06**(2.25)
	n=m=1000	42.17	19.14	50.98	**8.22**
MCFL	n=m=4000	4.58(0.37)	1.92(0.54)	1.26(0.26)	**1.20**(0.25)
	n=m=6000	6.75(1.65)	2.25(0.45)	64.36(13.97)	**1.48**(0.40)
	n=m=8000	7.33(0.67)	**2.45**(0.27)	64.43(20.47)	7.74(5.72)

Standard deviations are provided in parentheses. The best gap for every class of instances is highlighted in bold

C.5. Paired t-Tests for Comparing Different Selection Rules

To better support our argument that random selection is already as good as the problem-specific selection rules on our tested instances, we conduct paired t-tests, comparing the gap at termination for RPBD and Crainic et al. (2021) (in Table C.2) and Legault and Frejinger (2024) (in Table C.3), where the null hypothesis is RPBD performs the same as or better than the other. We observe that the majority of the results are not statistically significant, so we cannot reject the null hypothesis.

Table C.2 p-values of paired t-tests for comparing RPBD and Crainic et al. (2021) on MCND instances.

Instance size $\setminus \alpha$	0.01	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
R6.9	0.649	0.430	0.891	0.799	0.629	0.226	0.806	0.331	0.384	0.815	0.157
R8.9	0.019*	0.507	0.227	0.677	0.085	0.933	0.265	0.306	0.325	0.462	0.980
R10.9	0.011*	0.529	0.237	0.407	0.604	0.711	0.808	0.877	0.122	0.858	0.286
Instance size $\setminus \alpha$	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	0.99	
R6.9	0.385	0.784	0.131	0.783	0.484	0.287	0.446	0.191	0.151	0.321	
R8.9	0.704	0.714	0.797	0.700	0.488	0.004*	0.804	0.096	0.652	0.738	
R10.9	0.213	0.492	0.422	0.933	0.647	0.897	0.022*	0.767	0.667	0.948	

We test (H_0): RPBD performs the same as or better than Crainic et al. (2021) in terms of the gap at termination and indicate by * instances with p-value < 0.05.

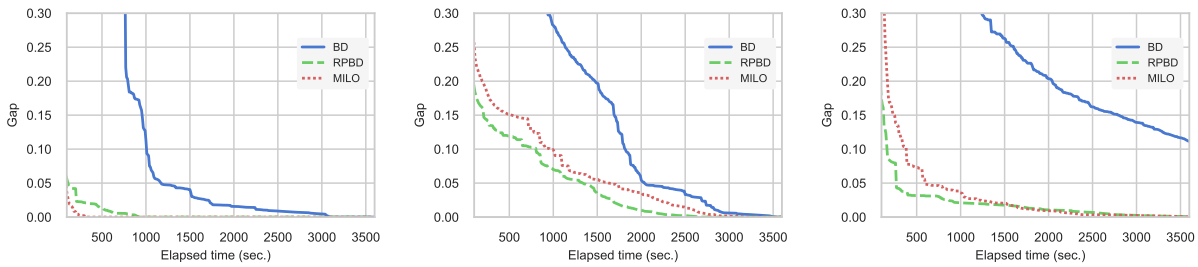
Table C.3 p**-values of paired** t**-tests for comparing RPBD and Legault and Frejinger (2024) on MCFL instances.**

Problem	Instance size	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.5$
MCFL	n=m=4000	0.365	0.891	0.083	0.003*	0.752
	n=m=6000	0.702	0.475	0.414	0.039	0.187
	n=m=8000	0.529	0.915	0.883	0.012	0.251

We test (H_0): RPBD performs the same as or better than Legault and Frejinger (2024) in terms of the gap at termination and indicate by * instances with p-value < 0.05.

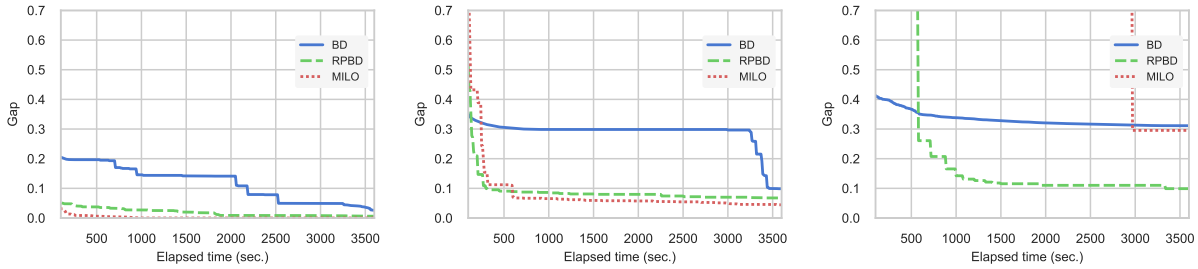
C.6. Experiments with One-Hour Time Limit

In the main paper, we compare algorithmic performance under a 3-minute time limit. In this section, we replicate our main experiments with a 1-hour time limit instead. In particular, for RPBD, the fine-tuning of α (described in Section 2.2) is performed with this new time limit in mind. Figure C.3, C.4, and C.5 report the optimality gap over time given for the three algorithms, on network design (MCND), uncapacitated facility location (UFL), and maximum coverage facility location (MCFL) instances, respectively. For MCFL, since we are using synthetic instances, we generated even larger instances for these instances.



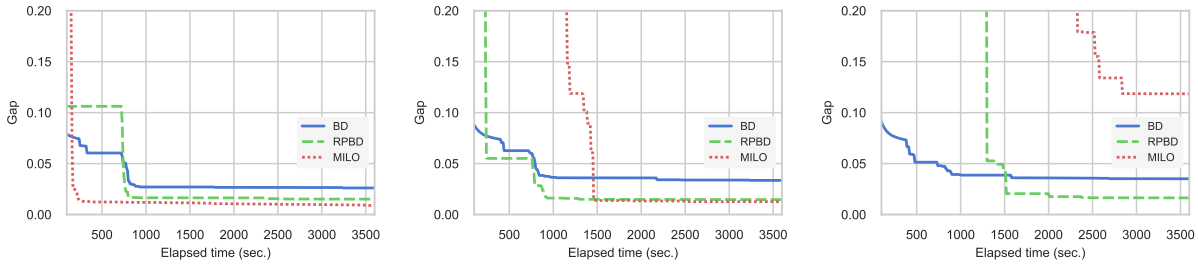
(a) R6.9. (b) R8.9 (c) R10.9.

Figure C.3 **Time vs. optimality gap trade-off curve for BD, RPBD, and MILO, with a one-hour time limit on the MCND instances.**



(a) $n = m = 300$. (b) $n = m = 500$. (c) $n = m = 1000$.

Figure C.4 **Time vs. optimality gap trade-off curve for BD, RPBD, and MILO, with a one-hour time limit on the UFL instances.**



(a) $n = m = 6,000$. (b) $n = m = 10,000$. (c) $n = m = 14,000$.

Figure C.5 Time vs. optimality gap trade-off curve for BD, RPBD, and MILO, with a one-hour time limit on the MCFL instances.

For MCND instances, we also summarize the cumulative distribution of termination time (i.e., the cumulative percentage of solved instances as a function of elapsed time) in Figure C.6.

Overall, these results confirm our main findings, namely that RPBD is an effective acceleration strategy that is particularly relevant in large-scale settings. Although the precise dimension values defining 'large-scale' are obviously relative to the amount of resources (and time limit) available.

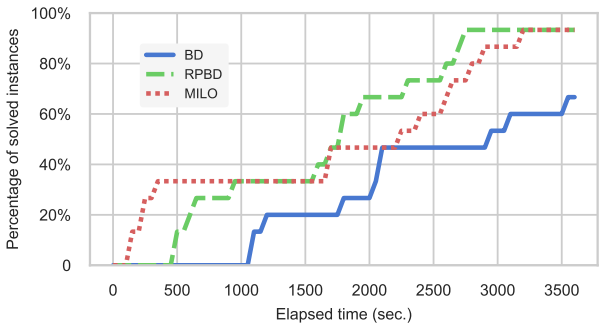
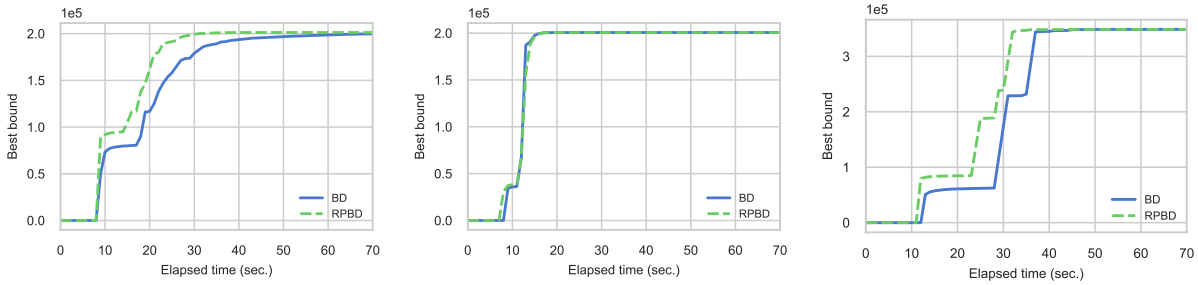


Figure C.6 Percentage of instances solved to near optimum (optimality gap $< 10^{-4}$) over elapsed time on MCND instances (R6.9, R8.9, and R10.9).

C.7. Impact on Solving the Continuous Relaxation

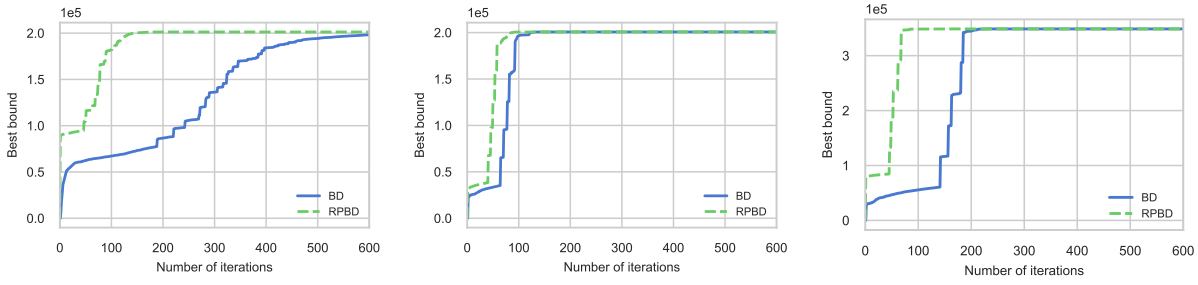
To isolate the benefit of RPBD on the lower bound, irrespective of discrete optimization-specific aspects such as branching, we solve the continuous relaxation of MCND problems via outer approximation and compare the behavior of RPBD and BD. We use the same configuration as in Section 3.

Figure C.7 and C.8 report the best bound by time/iterations for (RP)BD. We observe that RPBD achieves near optimality after 35 seconds on all instances, vs. 20-70 seconds for BD. The acceleration is even more pronounced when considering the per-iteration performance (Figure C.8) with RPBD requiring in general in half as many iterations as BD. These results confirm that even in the absence of integer constraints, RPBD improves the outer-approximation quality of the master problem and accelerates Benders schemes.



(a) R6.9. (b) R8.9. (c) R10.9

Figure C.7 Best (lower) bound over time, for BD and RPBD, when solving the continuous relaxation of MCND instances.



(a) R6.9. (b) R8.9. (c) R10.9

Figure C.8 Best (lower) bound over iterations, for BD and RPBD, when solving the continuous relaxation of MCND instances.

Appendix D: Complementary Theoretical Analysis

Proposition 1 compares the value of the relaxation with RPBD, $\hat{v}^\star(\boldsymbol{u}^\alpha)$ with that of the robust counterpart, $\bar{v}(\alpha)$. In this section, we derive a more comprehensive comparison between these optimization problems.

First, for a fixed vector $\boldsymbol{y}$, we compare the objective values of each problem at that particular point $\boldsymbol{y}$:

PROPOSITION D.1. *Fix $\boldsymbol{y}$ and denote $\Delta_{\max}(\boldsymbol{y}) := \max_{r\in\mathcal{R}}[\phi_r(\boldsymbol{y}) - \hat{\phi}_r(\boldsymbol{y})] < +\infty$. For any $\alpha \in [0,1]$ and any $t > 0$,*

$$\mathbb{P}\left(\left[\sum_r u_r^\alpha \phi_r(\boldsymbol{y}) + (1-u_r^\alpha)\hat{\phi}_r(\boldsymbol{y})\right] - \left[\sum_r \alpha\phi_r(\boldsymbol{y}) + (1-\alpha)\hat{\phi}_r(\boldsymbol{y})\right] \geq t\right) \leq \exp\left\{-\frac{2t^2}{\Delta_{\max}^2(\boldsymbol{y})}\cdot\frac{\alpha|\mathcal{R}|^2}{(1-\alpha)|\mathcal{R}|+1}\right\}.$$

REMARK D.1. If $\Delta_{\max}(\boldsymbol{y}) = +\infty$, it means that $\boldsymbol{y}$ is infeasible for the original problem ($\phi_r(\boldsymbol{y}) = +\infty$) but deemed feasible by the outer approximation ($\hat{\phi}_r(\boldsymbol{y}) < +\infty$). We can extend the result from Proposition D.1 to this case, by conditioning on the event where none of the indices $r \in \mathcal{R}$ such that $\phi_r(\boldsymbol{y}) - \hat{\phi}_r(\boldsymbol{y}) = +\infty$ are sampled.

Proof of Proposition D.1 To simplify the derivation, we introduce $k = \alpha|\mathcal{R}|$ random variables $\{h_1, \ldots, h_k\}$ sampled without replacement from the set $\{\phi_r(\boldsymbol{y}) - \hat{\phi}_r(\boldsymbol{y}),\ r \in \mathcal{R}\}$. With this notation,

$$\sum_r u_r^\alpha \phi_r(\boldsymbol{y}) + (1-u_r^\alpha)\hat{\phi}_r(\boldsymbol{y}) = \sum_r \hat{\phi}_r(\boldsymbol{y}) + \sum_k h_k,$$
$$\sum_r \alpha\phi_r(\boldsymbol{y}) + (1-\alpha)\hat{\phi}_r(\boldsymbol{y}) = \sum_r \hat{\phi}_r(\boldsymbol{y}) + \alpha\sum_r(\phi_r - \hat{\phi}_r)(\boldsymbol{y}).$$

As a result, we can apply concentration inequalities on the sum of random variables obtained when sampling without replacement (Serfling 1974, corollary 1.1): For any $t > 0$,

$$\begin{aligned}\mathbb{P}\left(\left[\sum_r u_r^\alpha \phi_r(\boldsymbol{y}) + (1-u_r^\alpha)\hat{\phi}_r(\boldsymbol{y})\right] - \left[\sum_r \alpha\phi_r(\boldsymbol{y}) + (1-\alpha)\hat{\phi}_r(\boldsymbol{y})\right] \geq t\right) &= \mathbb{P}\left(\left[\sum_k h_k\right] - \left[\alpha\sum_r(\phi_r - \hat{\phi}_r)\right] \geq t\right)\\ &\leq \exp\left\{-\frac{2kt^2}{(1-f)\Delta_{\max}^2(\boldsymbol{y})}\right\},\end{aligned}$$

with $k = \alpha|\mathcal{R}|$ and $f = (k-1)/|\mathcal{R}|$. □

In particular, Proposition D.1 allows us to compare the level sets of the different optimization problems. For a given $t \in \mathbb{R}$, the level set at level t of the robust counterpart is defined as

$$\bar{Q} := \left\{\boldsymbol{y} \,:\, \alpha\sum_r \phi_r(\boldsymbol{y}) + (1-\alpha)\sum_r \hat{\phi}_r(\boldsymbol{y}) \leq t\right\}.$$

It corresponds to the set of all the feasible solutions whose objective value is lower than t. When solving a mixed-integer optimization problem, for example, the solver updates a global upper bound (equal to the objective value of the best solution found) and prunes, i.e., ignores, parts of the feasible space where the objective value cannot be lower than the upper bound. In this context, the size of the lower set characterizes the size of the remaining search space. Denoting Q and $\hat{Q}$ the t-level set for the MILO and the BD objective, respectively, we have that $\hat{Q} \subseteq Q$: Because it approximates the objective from below, the level sets from BD are systematically larger than for the original MILO. In other words, for the same global solution (best solution found), the remaining part of the feasible space to consider is larger with

BD than it is for MILO. Because it linearly interpolates between the two objectives, the expected counterpart satisfies $\hat{Q} \subseteq \bar{Q} \subseteq Q$. The following proposition shows that the t-level set for RPBD is not significantly bigger than the t-level set of the expected counterpart, in the sense that for any $\boldsymbol{y} \notin \bar{Q}$, it does not belong to the t-level set of RPBD with high probability:

COROLLARY D.1. *Using the same notations and assumptions as Proposition D.1, consider a vector $\boldsymbol{y}$ satisfying*

$$\alpha \sum_r \phi_r(\boldsymbol{y}) + (1-\alpha) \sum_r \hat{\phi}_r(\boldsymbol{y}) > t.$$

for some $t \in \mathbb{R}$, then

$$\mathbb{P}\left(\sum_r u_r^\alpha \phi_r(\boldsymbol{y}) + (1-u_r^\alpha)\hat{\phi}_r(\boldsymbol{y}) > t \right) \geq 1 - \exp\left\{ -\frac{2\epsilon^2}{\Delta_{\max}(\boldsymbol{y})^2} \cdot \frac{\alpha|\mathcal{R}|^2}{(1-\alpha)|\mathcal{R}|+1} \right\},$$

with $\epsilon = \alpha \sum_r \phi_r(\boldsymbol{y}) + (1-\alpha)\sum_r \hat{\phi}_r(\boldsymbol{y}) - t > 0$.

Proof of Corollary D.1

$$\begin{aligned} &\sum_r u_r^\alpha \phi_r(\boldsymbol{y}) + (1-u_r^\alpha)\hat{\phi}_r(\boldsymbol{y}) \leq t \\ &\implies \sum_r u_r^\alpha \phi_r(\boldsymbol{y}) + (1-u_r^\alpha)\hat{\phi}_r(\boldsymbol{y}) - \left[\alpha \sum_r \phi_r(\boldsymbol{y}) + (1-\alpha)\sum_r \hat{\phi}_r(\boldsymbol{y}) \right] \leq -\epsilon \end{aligned}$$

and the result follows directly from Proposition D.1. □

The statement of Corollary D.1, however, applies to a fixed $\boldsymbol{y}$. We now derive concentration bounds on $\hat{v}^\star(\boldsymbol{u}^\alpha)$ by applying a uniform bound on the approximation error $\Delta_{\max}(\boldsymbol{y})$.

COROLLARY D.2. 1. *Denote $\bar{\Delta}_{\max} := \max_r (\phi_r - \hat{\phi}_r)(\bar{\boldsymbol{y}}(\alpha))$ where $\bar{\boldsymbol{y}}(\alpha)$ denotes an optimal solution of $\bar{v}(\alpha)$. For any $t > 0$,*

$$\mathbb{P}\left(\hat{v}^\star(\boldsymbol{u}^\alpha) - \bar{v}(\alpha) \geq t \right) \leq \exp\left\{ -\frac{2t^2}{\bar{\Delta}_{\max}^2} \cdot \frac{\alpha|\mathcal{R}|^2}{(1-\alpha)|\mathcal{R}|+1} \right\}.$$

2. *Denote $\Delta_{\max} := \max_{\boldsymbol{u}} \max_r (\phi_r - \hat{\phi}_r)(\boldsymbol{y}^\star(\boldsymbol{u}))$ where $\boldsymbol{y}^\star(\boldsymbol{u})$ denotes an optimal solution of $\hat{v}^\star(\boldsymbol{u})$. For any $t > 0$,*

$$\mathbb{P}\left(\hat{v}^\star(\boldsymbol{u}^\alpha) - \bar{v}(\alpha) \leq -t \right) \leq \binom{|\mathcal{R}|}{\alpha|\mathcal{R}|} \exp\left\{ -\frac{2t^2}{\Delta_{\max}^2} \cdot \frac{\alpha|\mathcal{R}|^2}{(1-\alpha)|\mathcal{R}|+1} \right\}.$$

Proof of Corollary D.2 1. Denoting $\bar{\boldsymbol{y}}(\alpha)$ an optimal solution of $\bar{v}(\alpha)$, we have

$$\begin{aligned} &\hat{v}^\star(\boldsymbol{u}^\alpha) - \bar{v}(\alpha) \\ &\qquad \leq \left[\sum_r u_r^\alpha \phi_r(\bar{\boldsymbol{y}}(\alpha)) + (1-u_r^\alpha)\hat{\phi}_r(\bar{\boldsymbol{y}}(\alpha)) \right] - \left[\sum_r \alpha\phi_r(\bar{\boldsymbol{y}}(\alpha)) + (1-\alpha)\hat{\phi}_r(\bar{\boldsymbol{y}}(\alpha)) \right] =: Z \end{aligned}$$

so,

$$\mathbb{P}\left(\hat{v}^\star(\boldsymbol{u}^\alpha) - \bar{v}(\alpha) \geq t \right) \leq \mathbb{P}\left(Z \geq t \right),$$

and the result follows from Proposition D.1.

2. Similarly, we have

$$\begin{aligned}
&\hat{v}^\star(\boldsymbol{u}^\alpha) - \bar{v}(\alpha) \\
&\quad \geq \left[\sum_r u_r^\alpha \phi_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha)) + (1-u_r^\alpha)\hat{\phi}_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha))\right] - \left[\sum_r \alpha\phi_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha)) + (1-\alpha)\hat{\phi}_r(\boldsymbol{y}^\star(\boldsymbol{u}^\alpha))\right] \\
&\quad \geq \min_{\boldsymbol{y}:\exists \boldsymbol{u}:\boldsymbol{y}=\boldsymbol{y}^\star(\boldsymbol{u})} \left[\sum_r u_r^\alpha \phi_r(\boldsymbol{y}) + (1-u_r^\alpha)\hat{\phi}_r(\boldsymbol{y})\right] - \left[\sum_r \alpha\phi_r(\boldsymbol{y}) + (1-\alpha)\hat{\phi}_r(\boldsymbol{y})\right] =: \min_{\boldsymbol{y}:\exists \boldsymbol{u}:\boldsymbol{y}=\boldsymbol{y}^\star(\boldsymbol{u})} Z(\boldsymbol{y}).
\end{aligned}$$

By a union bound,

$$\begin{aligned}
\mathbb{P}\left(\hat{v}^\star(\boldsymbol{u}^\alpha) - \bar{v}(\alpha) \leq -t\right) &\leq \mathbb{P}\left(\min_{\boldsymbol{y}:\exists \boldsymbol{u}:\boldsymbol{y}=\boldsymbol{y}^\star(\boldsymbol{u})} Z(\boldsymbol{y}) \leq -t\right) \\
&= \mathbb{P}\left(\exists \boldsymbol{u} \,:\, Z(\boldsymbol{y}^\star(\boldsymbol{u})) \leq -t\right) \\
&\leq \binom{|\mathcal{R}|}{\alpha|\mathcal{R}|} \exp\left\{-\frac{2t^2}{\Delta_{\max}^2} \cdot \frac{\alpha|\mathcal{R}|^2}{(1-\alpha)|\mathcal{R}|+1}\right\}.
\end{aligned}$$

□

Appendix E: Computational Results on Two-Stage Problems with Mixed-Integer Recourse

A growing line of research applies BD to two-stage stochastic problems with mixed-integer second-stage variables. In this setting, applying BD requires to relax the integrality constraints on the second-stage variables, hence leading to relaxation (instead of a reformulation) of the original problem.

In this section, we are interested in problems of the form $\min_{\boldsymbol{y}\in\mathcal{P}} \boldsymbol{d}^\top \boldsymbol{y} + \sum_r \phi_r^{\texttt{int}}(\boldsymbol{y})$, with

$$\begin{aligned}
\phi_r^{\texttt{int}}(\boldsymbol{y}) := \min_{\boldsymbol{x}_r} \quad & \boldsymbol{c}_r^\top \boldsymbol{x}_r \quad \text{s.t.} \quad \boldsymbol{A}_r\boldsymbol{x}_r + \boldsymbol{B}_r\boldsymbol{y} \geq \boldsymbol{b}_r, \\
& \boldsymbol{x}_r \in \mathbb{R}_+^{C_r} \times \mathbb{Z}_+^{I_r},
\end{aligned}$$

where the second-stage variables $\boldsymbol{x}_r$ contain I_r integer coordinates.

To apply Benders decomposition, we construct a lower bound on $\phi_r^{\texttt{int}}(\boldsymbol{y})$, $\phi_r(\boldsymbol{y})$, obtained by relaxing the integrality constraints on $\boldsymbol{x}_r$. By solving $\min_{\boldsymbol{y}\in\mathcal{P}} \boldsymbol{d}^\top \boldsymbol{y} + \sum_r \phi_r(\boldsymbol{y})$, we obtain a lower bound on the original problem. We also obtain a feasible solution, $\hat{\boldsymbol{y}}$, with a relaxed cost $z^{\texttt{rel}} = \boldsymbol{d}^\top \hat{\boldsymbol{y}} + \sum_r \phi_r(\hat{\boldsymbol{y}})$. We can also estimate the true cost the true cost of $\hat{\boldsymbol{y}}$, $z^{\texttt{IP}} = \boldsymbol{d}^\top \hat{\boldsymbol{y}} + \sum_r \phi_r^{\texttt{int}}(\hat{\boldsymbol{y}})$, thus providing an upper bound on the original problem. The difference $z^{\texttt{IP}} - z^{\texttt{rel}}$ captures the approximation error, at $\hat{\boldsymbol{y}}$, of relaxing the integrality constraints. Note that when solving $\min_{\boldsymbol{y}\in\mathcal{P}} \boldsymbol{d}^\top \boldsymbol{y} + \sum_r \phi_r(\boldsymbol{y})$ via (RP)BD, we may not have solved the problem at optimality, i.e., $z^{\texttt{rel}}$ may not be equal to the lower bound. Since $z^{\texttt{IP}} \geq z^{\texttt{rel}}$, the optimality gap of (RP)BD at termination is a lower bound on the optimality gap for the original problem.

By applying RPBD, we can keep the integrality constraints on the retained second-stage variables $\boldsymbol{x}_r, r \in \mathcal{R}^{\texttt{mp}}$, thus providing a tighter lower bound and potentially a higher quality solution.

We illustrate our approach on instances of the stochastic server location problem (SSLP). We adopt the formulation and dataset of Chen and Luedtke (2022). We consider two classes of SSLP instances (we generate 3 instances per class) with 40 servers, 50 clients, and 50 (resp. 200) scenarios. We scale the capacity parameter by dividing it by the number of servers to consider more challenging instances. Note that the number of second-stage binary variables equals $40 \times 50 \times 50$ (resp. $40 \times 50 \times 200$).

For each instance, we apply BD, RPBD ($\alpha = 0.1$), and MILO with a 300-second time limit. We also try to solve the problem to optimality by solving the MILO formulation, warm-started with the best solution found by the three

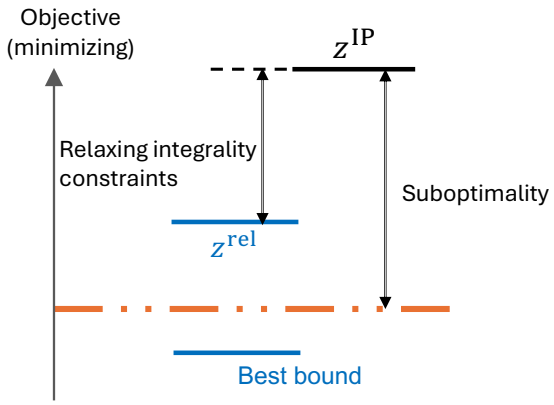


Figure E.1 **Illustration of solving problems with (RP)BD when the second-stage problems involve binary variables. Explicitly, $z^{\mathtt{rel}}$ is the optimal objective of relaxation: $d^\top \hat{y} + \sum_{r\in\mathcal{R}^{\mathtt{mp}}} c_r^\top \hat{x}_r + \sum_{r\in\mathcal{R}\setminus\mathcal{R}^{\mathtt{mp}}} \phi_r(\hat{y})$ where $(\hat{y}, \hat{x}_{r\in\mathcal{R}^{\mathtt{mp}}})$ is the master problem solution; $z^{\mathtt{IP}}$ is the true cost at $(\hat{y}, \hat{x}_{r\in\mathcal{R}^{\mathtt{mp}}})$: $d^\top \hat{y} + \sum_{r\in\mathcal{R}^{\mathtt{mp}}} c_r^\top \hat{x}_r + \sum_{r\in\mathcal{R}\setminus\mathcal{R}^{\mathtt{mp}}} \phi_r^{\mathtt{int}}(\hat{y})$; $z^\star$ is the optimal objective.**

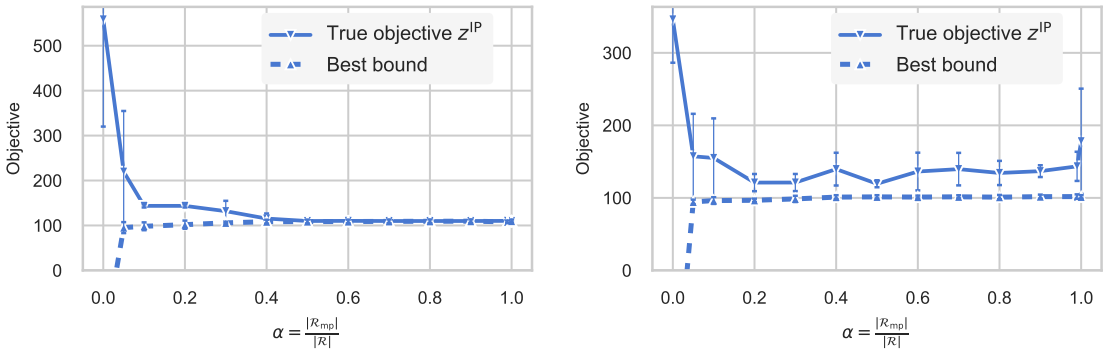
Table E.1 **Computational results on SSLP instances**

$\lvert\mathcal{R}\rvert$	Opt. obj. $z^\star$	Method	Solver information			True cost z^{IP}	Relaxation gap $z^{\mathrm{IP}} - z^{\mathrm{rel}}$	Suboptimality $z^{\mathrm{IP}} - z^\star$
			Feasible obj. z^{rel}	Best bound	Gap			
50	109.920	BD	113.383	−169.534	2.491	558.495	445.112	448.568
		RPBD	98.209	98.160	0.001	143.602	**45.393**	33.675
		MILO	109.920	109.809	0.001	109.920	-	**0.000**
200	[106.561, 116.208]	BD	120.711	−216.823	2.803	346.092	225.381	[229.884,239.531]
		RPBD	99.010	96.302	0.025	155.018	**56.008**	**[38.809, 48.456]**
		MILO	178.070	101.371	0.354	178.070	-	[61.862,71.509]

The "solver information" column shows the results (upper bound, lower bound, and relative gap) returned by the solver at termination. The "true cost" column is the final objective obtained by evaluating the master problem solution (from solving the relaxation) in the true formulation. For MILO, there is no relaxation, so z^{rel} is the same as z^{IP}.

aforementioned approaches, and ran for one hour. Results are presented in Table E.1. First, we observe that BD fails to converge within the time limit, with an average optimality gap at termination greater than 200% (which implies that it solved the original problem at best to that level of accuracy). In contrast, RPBD terminates with 0.1–2.5% optimality gap. Second, the lower bounds obtained by BD are negative, while the bounds from RPBD are much tighter, although less tight than MILO. Third, the solutions found by RPBD achieve a true cost (z^{IP}) two to three times lower than the solutions found by BD and even outperform the solutions found by MILO for the larger instance class. In particular, the approximation error ($z^{\mathrm{IP}} - z^{\mathrm{rel}}$) made by RPBD is five times smaller than that of BD (50 vs. 250), despite retaining only $\alpha = 10\%$ of the second-stage variables.

Lastly, we investigate the impact of α on the performance of RPBD in Figure E.2. We observe that, on the smaller instances ($|\mathcal{R}| = 50$), the upper and lower bounds converge to near optimum with (almost) no relaxation gap for $\alpha \geq 0.5$, meaning that only half of the second-stage variables are needed in the master problem to ensure integrality of the remaining ones. For the larger instances ($|\mathcal{R}| = 200$), RPBD outperforms both BD ($\alpha = 0$) and MILO ($\alpha = 1$) for most α values, reducing the suboptimality by more than half relative to either benchmark for $\alpha = 0.3$–0.5.



(a) $|\mathcal{R}| = 50$. (b) $|\mathcal{R}| = 200$.

Figure E.2 Comparison of performance on SSLP instances among different values of α. Error bars represent standard deviations. $\alpha = 0$ (1) corresponds to BD (MILO).

In conclusion, RPBD appears like an efficient solution to improve the accuracy of BD approximations for two-stage stochastic problems with mixed-integer recourse decisions, with the parameter α controlling the tightness–tractability trade-off. However, since it relies on relaxing integrality constraints, RPBD is not guaranteed to solve the original problem to optimality for $\alpha < 1$ (although we observe that it does in some cases), so it does not prevent the need for more sophisticated approaches like Lagrangian cuts (Chen and Luedtke 2022). However, it might increase the scale of practical problems we can solve simply using Benders approximation.