Optimized Scenario Reduction: Solving Large-scale Stochastic Programs with Quality Guarantees

Wei Zhang
Faculty of Business, The Hong Kong Polytechnic University, Hong Kong

Alexandre Jacquillat
Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

Kai Wang
School of Vehicle and Mobility, Tsinghua University, Beijing, China

Shuaian Wang
Faculty of Business, The Hong Kong Polytechnic University, Hong Kong

Stochastic programming involves large-scale optimization with exponentially many scenarios. This paper proposes an optimization-based scenario reduction approach to generate high-quality solutions and tight lower bounds by only solving small-scale instances, with a limited number of scenarios. First, we formulate a scenario subset selection model that optimizes the recourse approximation over a pool of solutions. We provide a theoretical justification of our formulation, and a tailored heuristic to solve it. Second, we propose a scenario assortment optimization approach to compute a lower bound—hence, an optimality gap—by relaxing nonanticipativity constraints across scenario "bundles". To solve it, we design a new column-evaluation-and-generation algorithm, which provides a generalizable method for optimization problems featuring many decision variables and hard-to-estimate objective parameters. We test our approach on stochastic programs with continuous and mixed-integer recourse. Results show that (i) our scenario reduction method dominates scenario reduction benchmarks, (ii) our scenario assortment optimization, combined with column-evaluation-and-generation, yields tight lower bounds, and (iii) our overall approach results in stronger solutions, tighter lower bounds, and faster computational times than state-of-the-art stochastic programming algorithms.

Key words: Stochastic programming; Scenario reduction; Column evaluation and generation.

1. Introduction

Since its introduction by **?**, stochastic programming has had a tremendous impact on decision making under uncertainty, with applications in supply chain management, energy systems, healthcare, urban operations, cloud computing, portfolio management, etc. Despite being considerably larger and more complex than their deterministic counterparts, stochastic programs can now be solved in large-scale instances (see **??**). Yet, there remain limits on the size and complexity that existing algorithms can handle. Moreover, stochastic programs with integer recourse remain notoriously challenging, with a lack of general-purpose algorithms that can routinely scale to realistic problems (**?**).

Accordingly, *scenario reduction* seeks a smaller problem instance that can yield high-quality solutions. Consider a "full" two-stage stochastic program $\min_{\boldsymbol{x}\in\mathcal{X}} \left\{\boldsymbol{c}^T\boldsymbol{x} + \sum_{s\in\mathcal{S}} h_s Q(\boldsymbol{x},\xi_s)\right\}$, where

$\boldsymbol{x} \in \mathcal{X}$ is the first-stage decision, $\mathcal{S}$ is a finite set of scenarios with probabilities $(h_s)_{s\in\mathcal{S}}$, and $Q(\cdot,\cdot)$ is the recourse function. The scenario set $\mathcal{S}$ can be exponentially large (for instance, n uncertain components each with m possible values lead to m^n scenarios). Scenario reduction aims to approximate $\mathcal{S}$ with a small subset of K scenarios (say, $K = 10$, 100, or $1,000$), such that the resulting stochastic program can be handled with available solution algorithms. The most common approach, sample average approximation (SAA), randomly samples equiprobable scenarios (**?**). This approach is asymptotically optimal and has shown considerable success in practice, but can also lead to suboptimality in small samples. Another approach, termed "distribution-based scenario reduction" ("DSR" for short), generates scenarios and their probabilities to minimize their distance from the full scenario set (see, e.g., **?**). However, the scenarios that best approximate the input distribution may not lead to the best stochastic programming solution. Moreover, DSR does not yield lower bounds of the full problem and SAA yields stochastic bounds.

This paper proposes new scenario reduction methods to address these limitations. Throughout, we impose that the stochastic program can be solved with up to K scenarios, where K is dictated by the complexity of the problem, the solution algorithm, and practical requirements. Our methods aim to: (i) generate a high-quality solution, in view of the full scenario set $\mathcal{S}$, and (ii) derive a lower bound of the full stochastic program, hence an optimality gap. By design, we seek general-purpose methods that encompass a broad class of stochastic programming models, including problems with mixed-integer recourse (as opposed to relying on strong inner duality).

In response to the first objective, we propose an optimization approach to scenario reduction, via a scenario subset selection (SSS) model that optimizes the approximation of the recourse function over a pool of first-stage solutions—as opposed to approximating the input probability distribution. The SSS model is formulated as a mixed-integer program, using connections with sparse regression in statistical learning (**?**), and we solve it via a tailored heuristic. This approach is motivated by a theoretical bound on the quality of the solution based on (i) the representativeness of the solution pool, and (ii) the quality of the recourse function approximation.

In response to the second objective, we propose a scenario assortment optimization (SAO) approach that relaxes nonanticipativity constraints across scenario "bundles", solves a smaller-scale stochastic program within each bundle, and sums costs across bundles. This approach yields a valid lower bound to the full stochastic program, and extends previous methods by allowing each scenario to be "split" into multiple bundles. We seek the largest bound within this family.

The SAO formulation features two sources of complexity: (i) a large, or even infinite, number of bundle-based variables; and (ii) hard-to-evaluate cost parameters (a stochastic program, in our case). To circumvent these difficulties, we propose a column-evaluation-and-generation approach that leverages a fast relaxation-approximation of the cost parameters (a stochastic program over

a restricted solution space, in our case). The algorithm iterates between a column generation step to derive a solution based on the relaxation (addressing difficulty (i)), and a column evaluation step to update the cost parameters for selected solutions (addressing difficulty (ii)). We prove that the column-evaluation-and-generation procedure exhibits the same convergence properties as the underlying column-generation algorithm—in particular, it converges finitely to the optimum if the number of variables is finite. Ultimately, the SAO approach yields a valid lower bound at each iteration, and exhibits strong empirical convergence. The column-evaluation-and-generation algorithm provides a stand-alone contribution of this paper to solve optimization problems where objective parameters are hard to estimate but a relaxation-approximation thereof is available.

We test our models and algorithms on four problems: (i) the production routing problem with continuous recourse (PRP–CR) from **?**; (ii) a new one with mixed-integer recourse (PRP–IR); (iii) a facility location problem with continuous recourse (FLP–CR); and (iv) one with mixed-integer recourse (FLP–IR). We first apply standard stochastic programming algorithms (i.e., CPLEX implementation, Benders decomposition for problems with continuous recourse, the integer L-shaped method for problems with mixed-integer recourse) and more advanced ones using additional cuts ("enhanced" Benders decomposition and "enhanced" integer L-shaped). Results show that all four problems are highly complex, both from a stochastic standpoint—scenario-based solutions exhibit poor out-of-sample performance—and from a computational standpoint—all baseline algorithms fail to solve the full 500-scenario problems within a five-hour time limit.

Our computational results fall into three categories. First, our SSS solution outperforms the scenario reduction benchmarks (SAA and DSR) for all problems, by up to 12–14%. This suggests that our optimization-based problem-driven approach to scenario reduction can provide benefits as compared to randomized or distribution-driven scenario reduction. Second, our scenario assortment optimization, combined with our column-evaluation-and-generation algorithm, generates a strong lower bound to the full stochastic program. These lower-bounding results guarantee that our solution (obtained with only 10–30 out of 500 scenarios) is optimal for three out of four problem settings and lies within 3.6% of the optimum for the fourth one. Third, our overall approach yields a Pareto improvement over state-of-the-art stochastic programming algorithms for all four problems: higher-quality solutions, tighter lower bounds, and faster computational times.

In summary, this paper makes three main contributions. First, we develop an optimization-based scenario reduction approach to approximate the recourse function in two-stage stochastic programming, and a dedicated heuristic algorithm to solve the resulting scenario subset selection model. Second, we propose a scenario assortment optimization approach to generate a lower bound by only solving small-scale instances. In particular, we develop a new column-evaluation-and-generation algorithm for large-scale optimization formulations with a prohibitive number of hard-to-evaluate

objective parameters. Third, we demonstrate, through extensive computational experiments, that our solution outperforms scenario reduction benchmarks and is competitive with state-of-the-art stochastic programming algorithms. These results do not imply that scenario reduction should replace advanced stochastic programming algorithms. Rather, this paper suggests a complementary way to achieve high-quality solutions and strong performance guarantees through dimensionality reduction, and proposes general-purpose models and algorithms to do so.

2. Literature review

Scenario reduction. Stochastic programming typically models uncertainty with discrete scenarios (**???**). Scenario reduction seeks the "best" set of K scenarios. Scenario reduction methods can be classified into distribution-based approaches, which approximate the distribution of the input parameters, and problem-driven approaches, which approximate the downstream stochastic programming formulation.

Within distribution-based methods, the most common approach is sample average approximation (SAA) that generates equiprobable scenarios through Monte Carlo sampling. SAA is asymptotically optimal, that is, its solution converges to the stochastic programming optimum as the number of scenarios grows infinitely, and provides statistical guarantees on convergence rates (see, e.g. **??????**). SAA convergence can be accelerated via variance reduction such as quasi-Monte Carlo, antithetic, Latin hypercube and importance sampling (**??????**). SAA has shown considerable success in transportation (**??**), supply chain management (**?**), health care (**??**), energy (**?**), etc. Yet, the number of scenarios to guarantee near-optimality can grow prohibitively large for large-scale problems. Thus, SAA can produce optimistically-biased estimates and potentially suboptimal solutions when evaluated out of sample (**???**).

In response, a branch of the distribution-based scenario reduction literature seeks "representative" scenarios by minimizing the distance to the input distribution (**?**). **?** use non-linear optimization to generate scenarios that satisfy user-specified statistical properties. **?** extend this approach to match distribution moments. **?** minimize the Wasserstein metric to approximate continuous stochastic processes. **?** and **?** formulate the scenario reduction problem with the Fortet-Mourier metric, which they solve via heuristic algorithms. **?** formulate the problem as a facility location problem, and solve it with forward and backward heuristics. **?** instead solve the scenario reduction problem, based on a transportation distance, via integer optimization.

As opposed to distribution-based methods, problem-driven scenario reduction considers the downstream optimization problem to capture the impact on expected costs. In unit commitment, **?** cluster scenarios based on a solution sensitivity index that encompasses generation costs and load imbalances. In power systems expansion, **?** aggregate scenarios around representative ones, using

such features as active power flows and investment costs. In portfolio selection, **??** combine many scenarios to focus on relevant scenarios in a sub-area of the distribution. In a similar risk-averse setting, **?** seek "effective" scenarios (that impact the optimal objective value) while minimizing the distance to the original distribution. Turning to general-purpose problem-driven scenario reduction methods, **?** develop an iterative scenario expansion procedure guided by the model's solutions for problems with continuous recourse and right-hand side uncertainty. With binary uncertainty, **?** identify, and eliminate, scenarios that do not impact the decisions. **?** minimize the difference between the optimal expected objective values of the problems with the reduced and full scenario sets. **?** minimize a loss function characterizing the error in the recourse function approximation, both in and out of sample, over a pool of solutions. This leads to a non-convex loss minimization problem, solved by heuristics. **?** proposed a *prescription divergence* metric to partition a large number of scenarios into a small number of clusters. This also leads to a non-convex problem, solved by iterating between partitioning scenarios (given the cluster centroids) and updating the centroids (given the scenario partition). In concurrent work, **?** extended the p-median model from **?** with a new distance metric that incorporates recourse-based deviations based on solutions to small-scale instances of the problem, as opposed to merely minimizing the transportation distance between input distributions.

Our paper extends this literature in three ways. First, we formalize a scenario subset selection problem for scenario reduction, drawing connections with the sparse regression problem in machine learning. This problem is formulated via mixed-integer linear optimization and solved via tailored algorithms. This differs from previous formulations based on semi-infinite programming (**?**), clustering (**?????**), and non-convex optimization (**??**). Second, our approach is applicable to a broad class of two-stage stochastic programs, including problems with mixed-integer recourse—a notoriously challenging class of problems. This differs from settings with continuous recourse (e.g., **?**). Third, unlike all aforementioned papers, we derive a lower bound of the full stochastic program to provide solution quality guarantees. This relates to the literature on scenario decomposition, which we review next.

Scenario decomposition. Our lower-bounding scenario assortment optimization falls into the scenario decomposition literature. The progressive hedging algorithm from **?** dualizes nonanticipativity constraints, iterating between solving scenario-based relaxations and aggregating solutions to restore nonanticipativity. It has been applied to stochastic integer programming (**?**), stochastic binary programming (**?**), stochastic mixed-integer programming (**?**), and chance-constrained optimization (**?**). In this paper, we leverage scenario decomposition to derive a lower bound of the stochastic program. This builds upon recent "scenario grouping" (SG) methods using, for instance,

k-means clustering (**?**) or adaptive partitioning (**?**). Closely related, **?** propose an optimization-driven scenario grouping method that maximizes the lower bound obtained by relaxing nonanticipativity across disjoint groups.

Our scenario assortment optimization (SAO) approach enhances the SG approach in two ways. From a modeling standpoint, we allow each scenario to be "broken down" into several bundles, so the SAO lower bound is at least as tight as the SG one. From a computational standpoint, we develop a new, and generalizable, column-evaluation-and-generation algorithm to solve a SAO set partitioning formulation, with hard-to-estimate objective parameters. Our results show that this approach results in a tighter bound than an SG benchmark inspired from **?**.

3. Upper bound: An optimization-based scenario reduction approach

We consider a two-stage stochastic program, where $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}, \xi)$ denote first-stage and second-stage decisions, uncertainty is characterized by a finite set of scenarios $\mathcal{S}$ each associated with a realization ξ_s and a probability h_s, and $Q(\cdot, \cdot)$ denotes the recourse function:

$$\mathcal{P}(\mathcal{S}, \boldsymbol{h}) = \min_{\boldsymbol{x} \in \mathcal{X}} \left\{ \boldsymbol{c}^T \boldsymbol{x} + \sum_{s \in \mathcal{S}} h_s Q(\boldsymbol{x}, \xi_s) \right\}, \qquad \text{where:} \quad Q(\boldsymbol{x}, \xi_s) = \min_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}, \xi_s)} \left\{ \boldsymbol{q}^T \boldsymbol{y} \right\}. \tag{1}$$

We assume that $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$ has relatively complete recourse, i.e., $\mathcal{Y}(\boldsymbol{x}, \xi_s) \neq \emptyset$ for all first-stage decisions $\boldsymbol{x}$ and all scenarios s. However, we make no assumption on the structure of the first-stage and second-stage problems, thus capturing two-stage stochastic continuous and integer programming.

We assume that $\mathcal{S}$ is finite, yet too large for Equation (1) to be solved directly. Throughout this paper, we only solve instances with up to K scenarios, determined by the problem's complexity, the solution algorithm, and practical requirements. This section focuses on scenario reduction to derive a solution to Equation (1); the next one turns to scenario assortment to derive a lower bound.

3.1. Scenario subset selection (SSS) model

We seek a subset $\mathcal{S}_0 \subseteq \mathcal{S}$ such that $|\mathcal{S}_0| \leq K$, along with weights $(w_s)_{s \in \mathcal{S}_0}$, so that the recourse function estimated over $\mathcal{S}_0$ with the weight vector $\boldsymbol{w}$ approximates the recourse function estimated over the full scenario set $\mathcal{S}$ with the probability vector $\boldsymbol{h}$. Our scenario subset selection (SSS) model minimizes the error in this approximation over a pool of feasible first-stage solutions $\mathcal{X}^P \subseteq \mathcal{X}$:

$$SSS(\mathcal{X}^P, K) \quad \Delta = \min_{\boldsymbol{w} \geq \boldsymbol{0}} \left\{ \sum_{\boldsymbol{x} \in \mathcal{X}^P} \left| \sum_{s \in \mathcal{S}} h_s Q(\boldsymbol{x}, \xi_s) - \sum_{s \in \mathcal{S}} w_s Q(\boldsymbol{x}, \xi_s) \right| : \sum_{s \in \mathcal{S}} \mathbb{1}(w_s \neq 0) \leq K \right\}. \tag{2}$$

Note that we restrict the weights w_s to be nonnegative for interpretability purposes. However, we do not restrict them to sum up to 1 in order to avoid unnecessarily constraining the SSS model.

Our SSS formulation relates to **?**, who also approximate a recourse function to preserve the ranking among the first-stage solutions in $\mathcal{X}^P$. Their formulation applies differentiated weights to

different solutions (to focus on "promising" regions) and penalizes solutions that perform better in sample than out of sample (to avoid overconfident outliers). Our formulation is somewhat simpler. From a modeling standpoint, it avoids tuning weights for different solutions and another one for penalizing outliers. From a statistical standpoint, it does not rely on finding "good" solutions in the pool $\mathcal{X}^P$, but instead reconstructs a global approximation of the recourse function. From a computational standpoint, it can tackle general stochastic programs, as opposed to problems with simple recourse or binary uncertainty as in **?**.

Technically, our SSS model minimizes the absolute error from the "true" recourse function, under scenario sparsity constraints. This connects to the sparse regression problem, which minimizes the prediction error under feature sparsity constraints (**?**). With this interpretation, each "observation" is a first-stage solution $\boldsymbol{x} \in \mathcal{X}^P$, each "feature" is the recourse function in a scenario $s \in \mathcal{S}$, and the "outcome variable" is the expected recourse $\sum_{s\in\mathcal{S}} h_s Q(\boldsymbol{x}, \xi_s)$. Our setting results in highly correlated features, leading to a challenging optimization problem.

We can reformulate SSS as the following mixed-integer optimization model, where M denotes a large number and z_s denotes a binary variable equal to 1 if $s \in \mathcal{S}_0$ and 0 otherwise.

$$SSS(\mathcal{X}^P, K) \quad \Delta = \min \quad \sum_{\boldsymbol{x}\in\mathcal{X}^P} \left| \sum_{s\in\mathcal{S}} h_s Q(\boldsymbol{x}, \xi_s) - \sum_{s\in\mathcal{S}} w_s Q(\boldsymbol{x}, \xi_s) \right| \tag{3}$$

$$\text{s.t.} \quad w_s \leq M z_s \quad \forall s \in \mathcal{S} \tag{4}$$

$$\sum_{s\in\mathcal{S}} z_s \leq K \tag{5}$$

$$w_s \geq 0, z_s \in \{0,1\} \quad \forall s \in \mathcal{S}. \tag{6}$$

Benchmarks

Sample average approximation (SAA): build $\mathcal{S}_0$ by uniformly sampling K scenarios from the set $\mathcal{S}$ and assigning the same probability to each, that is, $w_s = \frac{1}{K}$ for all $s \in \mathcal{S}_0$ (**?**).

Distribution-based scenario reduction (DSR): select a scenario subset $\mathcal{S}_0$ by minimizing the distance to the full distribution $(\xi_s)_{s\in\mathcal{S}}$. We define a distance from scenario s to scenario s' as $D^{DSR}_{ss'} = h_s \|\xi_s - \xi_{s'}\|_2$. We define binary variables $\zeta_{s'}$ equal to 1 if $s' \in \mathcal{S}_0$, and $\psi_{ss'}$ equal to 1 if scenario $s \in \mathcal{S}$ is mapped to $s' \in \mathcal{S}_0$. The DSR is formulated as the following p-median problem (**?**), and we recover $\mathcal{S}_0 = \{s' \in \mathcal{S} : \zeta^*_{s'} = 1\}$ and $w_{s'} = \sum_{s\in\mathcal{S}} h_s \psi^*_{ss'}$ for $s' \in \mathcal{S}_0$.

$$\min_{\substack{\boldsymbol{\zeta}\in\{0,1\}^{\mathcal{S}} \\ \boldsymbol{\psi}\in\{0,1\}^{\mathcal{S}\times\mathcal{S}}}} \left\{ \sum_{s\in\mathcal{S}} \sum_{s'\in\mathcal{S}} D^{DSR}_{ss'} \psi_{ss'} : \sum_{s'\in\mathcal{S}} \zeta_{s'} \leq K,\ \psi_{ss'} \leq \zeta_{s'}\ \forall s, s' \in \mathcal{S},\ \sum_{s'\in\mathcal{S}} \psi_{ss'} = 1\ \forall s \in \mathcal{S} \right\}. \tag{7}$$

The main difference between DSR and SSS lies in the objective function: DSR approximates the input distribution whereas SSS approximates the recourse function of the optimization problem.

All methods yield a scenario subset $\mathcal{S}_0$ of cardinality K and corresponding weights $(w_s)_{s\in\mathcal{S}_0}$. We then solve the stochastic programming problem $\mathcal{P}(\mathcal{S}_0, \boldsymbol{w})$. We evaluate the solution (denoted by $\widetilde{\boldsymbol{x}}$) out of sample in view of the original stochastic program, $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$, by computing the total expected cost across the full scenario set, i.e.: $\boldsymbol{c}^T\widetilde{\boldsymbol{x}} + \sum_{s\in\mathcal{S}} h_s Q(\widetilde{\boldsymbol{x}}, \xi_s)$.

An illustrative example

To motivate our approach, assume that $\mathcal{S}$ comprises three equiprobable scenarios such that $\xi_1 = 0, \xi_2 = 40, \xi_3 = 100$. Suppose that $Q(\boldsymbol{x}, \xi) = \max_{y\in\{0,100\}} 3\,|y-\xi|\,\|\boldsymbol{x}\|_2$. The full problem is thus:

$$\mathcal{P}(\mathcal{S}, \boldsymbol{h}) = \min_{\boldsymbol{x}\in\mathcal{X}} \left\{\boldsymbol{c}^T\boldsymbol{x} + 260\|\boldsymbol{x}\|_2\right\}.$$

Suppose that we need to select two scenarios ($K = 2$). Since $Q(\cdot, \xi_1) = Q(\cdot, \xi_3)$, SSS selects $\mathcal{S}_0 = \{1, 2\}$ with $w_1 = \frac{2}{3}$ and $w_2 = \frac{1}{3}$, or $\mathcal{S}_0 = \{2, 3\}$ with $w_2 = \frac{1}{3}$ and $w_3 = \frac{2}{3}$. Either way, we obtain:

$$\mathcal{P}(\mathcal{S}_0, \boldsymbol{w})_{SSS} = \min_{\boldsymbol{x}\in\mathcal{X}} \left\{\boldsymbol{c}^T\boldsymbol{x} + 260\|\boldsymbol{x}\|_2\right\}.$$

With SAA, each scenario is sampled (with replacement) with probability 0.5, so that:

$$\mathcal{P}(\mathcal{S}_0, \boldsymbol{w})_{SAA} = \begin{cases} \min_{\boldsymbol{x}\in\mathcal{X}} \{\boldsymbol{c}^T\boldsymbol{x} + 300\|\boldsymbol{x}\|_2\} & \text{if } \mathcal{S}_0 = \{1,1\},\ \mathcal{S}_0 = \{1,3\}, \text{ or } \mathcal{S}_0 = \{3,3\}, \\ \min_{\boldsymbol{x}\in\mathcal{X}} \{\boldsymbol{c}^T\boldsymbol{x} + 240\|\boldsymbol{x}\|_2\} & \text{if } \mathcal{S}_0 = \{1,2\} \text{ or } \mathcal{S}_0 = \{2,3\}, \\ \min_{\boldsymbol{x}\in\mathcal{X}} \{\boldsymbol{c}^T\boldsymbol{x} + 180\|\boldsymbol{x}\|_2\} & \text{if } \mathcal{S}_0 = \{2,2\}. \end{cases}$$

With DSR, we can select $\mathcal{S}_0 = \{1, 3\}$ with $w_1 = \frac{2}{3}$ and $w_3 = \frac{1}{3}$ (scenario 2 is mapped to scenario 1), or $\mathcal{S}_0 = \{2, 3\}$ with $w_2 = \frac{2}{3}$ and $w_3 = \frac{1}{3}$ (scenario 1 is mapped to scenario 2). We obtain:

$$\mathcal{P}(\mathcal{S}_0, \boldsymbol{w})_{DSR} = \begin{cases} \min_{\boldsymbol{x}\in\mathcal{X}} \{\boldsymbol{c}^T\boldsymbol{x} + 300\|\boldsymbol{x}\|_2\} & \text{if } \mathcal{S}_0 = \{1,3\}, \\ \min_{\boldsymbol{x}\in\mathcal{X}} \{\boldsymbol{c}^T\boldsymbol{x} + 220\|\boldsymbol{x}\|_2\} & \text{if } \mathcal{S}_0 = \{2,3\}. \end{cases}$$

In this example, $\mathcal{P}(\mathcal{S}_0, \boldsymbol{w})_{SSS} = \mathcal{P}(\mathcal{S}, \boldsymbol{h})$, so SSS returns the optimal solution to $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$. In contrast, $\mathcal{P}(\mathcal{S}_0, \boldsymbol{w})_{SAA}$ and $\mathcal{P}(\mathcal{S}_0, \boldsymbol{w})_{DSR}$ may provide suboptimal solutions to $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$.

In this simplified setting, the SSS problem is solved to optimality (with a zero objective) and the weights sum up to 1; as mentioned above, this needs not be the case. Moreover, this example does not depend on the subset of solutions $\mathcal{X}^P$; however, this is also not the case in general.

3.2. The solution pooling model

Our SSS problem (Equation (2)) approximates the recourse function over a pool of first-stage solutions $\mathcal{X}^P \subseteq \mathcal{X}$. Ideally, we would set $\mathcal{X}^P = \mathcal{X}$ but this would lead to intractability in the SSS model, as $\mathcal{X}$ is either infinite or exponentially large. We thus define a restricted pool $\mathcal{X}^P$ of first-stage solutions, with limited cardinality (denoted by P), in order for the resulting recourse function approximation across $\mathcal{X}^P$ to capture the overall recourse function over the entire set $\mathcal{X}$.

We propose a simple pooling approach. We first start from a large solution subset $\mathcal{X}^A \subseteq \mathcal{X}$. Our pooling model then reduces $\mathcal{X}^A$ into a "representative" subset $\mathcal{X}^P \subseteq \mathcal{X}^A$ of cardinality P

to minimize the largest deviation from $\mathcal{X}^A$ to $\mathcal{X}^P$, using a deviation function that captures the absolute difference in the recourse functions: $d_{\boldsymbol{x},\boldsymbol{x}',s} = |Q(\boldsymbol{x},\xi_s) - Q(\boldsymbol{x}',\xi_s)|$. Let $\zeta_{\boldsymbol{x}'} = 1$ if $\boldsymbol{x}' \in \mathcal{X}^P$, and $\psi_{\boldsymbol{x},\boldsymbol{x}'} = 1$ if $\boldsymbol{x} \in \mathcal{X}^A$ is mapped to $\boldsymbol{x}' \in \mathcal{X}^P$. The pooling problem is formulated as:

$$\min_{\boldsymbol{\zeta}\in\{0,1\}^{\mathcal{X}^A},\ \boldsymbol{\psi}\in\{0,1\}^{\mathcal{X}^A\times\mathcal{X}^A}} \left\{ \Delta^P: \quad \begin{aligned} &\Delta^P \geq d_{\boldsymbol{x},\boldsymbol{x}',s}\psi_{\boldsymbol{x},\boldsymbol{x}'} \quad \forall \boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}^A, \forall s \in \mathcal{S}, \\ &\psi_{\boldsymbol{x},\boldsymbol{x}'} \leq \zeta_{\boldsymbol{x}'} \quad \forall \boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}^A, \\ &\sum_{\boldsymbol{x}'\in\mathcal{X}^A} \psi_{\boldsymbol{x},\boldsymbol{x}'} = 1 \quad \forall \boldsymbol{x} \in \mathcal{X}^A, \\ &\sum_{\boldsymbol{x}'\in\mathcal{X}^A} \zeta_{\boldsymbol{x}'} = P \end{aligned} \right\}. \tag{8}$$

The solution pooling model reduces to a facility location problem, and is thus NP-hard. We design a two-step heuristic to solve it efficiently. We present and evaluate this heuristic in Appendix A.1.

3.3. Theoretical justification of the scenario reduction approach

Altogether, our scenario reduction approach involves (i) generating a large solution set $\mathcal{X}^A \subseteq \mathcal{X}$, (ii) reducing it into a smaller pool $\mathcal{X}^P \subseteq \mathcal{X}^A$ (Equation (8)), and (iii) selecting a scenario subset $\mathcal{S}_0$ via SSS (Equations (3)–(6)). We then solve $\mathcal{P}(\mathcal{S}_0, \boldsymbol{w})$ as an approximation of $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$.

Figure 1 illustrates the approximation (in blue) of the full recourse function (in green). The SSS formulation minimizes the approximation error Δ, estimated in discrete points $\boldsymbol{x} \in \mathcal{X}^P$ (in red). Our pooling procedure aims to map each point of $\mathcal{X}^A$ (as a proxy for $\mathcal{X}$) to a nearby neighbor in $\mathcal{X}^P$, so that the recourse function approximation generalizes over $\mathcal{X}^A \setminus \mathcal{X}^P$ (as a proxy for $\mathcal{X} \setminus \mathcal{X}^P$). Theorem 1 (proved in Appendix A.2) justifies this approach by bounding the difference between the out-of-sample cost achieved by the solution of $\mathcal{P}(\mathcal{S}_0, \boldsymbol{w})$ and the optimum of $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$.

LEMMA 1. *Let $\boldsymbol{w}$ denote the optimal solution of SSS. There exists $\overline{U}$ that depends on the characteristics of $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$, such that $w_s \leq \overline{U}$ for all $s \in \mathcal{S}$.*

THEOREM 1. *Let $\widetilde{\boldsymbol{x}}$ be an optimal solution of $\mathcal{P}(\mathcal{S}_0, \boldsymbol{w})$ and $\boldsymbol{x}^*$ be an optimal solution of $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$. Let $\Delta^A = \max_{\boldsymbol{x}\in\mathcal{X}} \min_{\boldsymbol{x}^A\in\mathcal{X}^A} \|\boldsymbol{x} - \boldsymbol{x}^A\|$, and assume that the recourse function $Q(\cdot, \xi_s)$ is L-Lipschitz continuous for all $s \in \mathcal{S}$. Using the notations from Lemma 1, we have:*

$$\left(\boldsymbol{c}^T\widetilde{\boldsymbol{x}} + \sum_{s\in\mathcal{S}} h_s Q(\widetilde{\boldsymbol{x}}, \xi_s)\right) - \left(\boldsymbol{c}^T\boldsymbol{x}^* + \sum_{s\in\mathcal{S}} h_s Q(\boldsymbol{x}^*, \xi_s)\right) \leq 2\left[\Delta + \left(1 + K\overline{U}\right)\left(\Delta^P + L\Delta^A\right)\right]. \tag{9}$$

In words, the error of the solution is bounded by (i) the error in the SSS recourse function approximation over $\mathcal{X}^P$, (ii) the maximum pooling distance from $\mathcal{X}^A$ to $\mathcal{X}^P$, and (iii) the covering distance between $\mathcal{X}^A$ and $\mathcal{X}$. In an ideal case, (i) if the SSS is "perfect", with a zero objective, (ii) if the solution pooling process is "perfect", with a zero objective, and (iii) if the pool $\mathcal{X}^A$ is

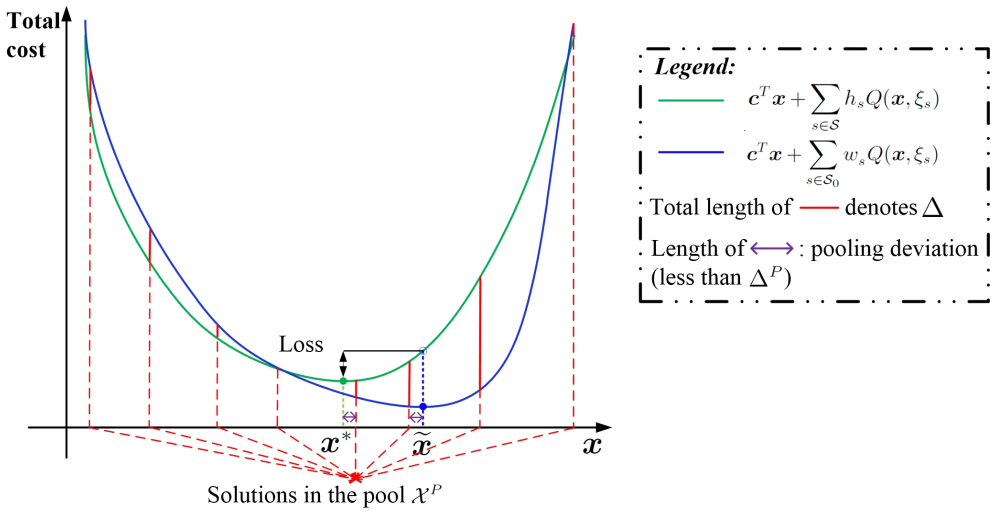


Figure 1 **One-dimensional visualization of the optimization-based scenario reduction procedure.**

vast enough to "cover" the full solution space $\mathcal{X}$, then the solution of $\mathcal{P}(\mathcal{S}_0, \boldsymbol{w})$ will be optimal in view of $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$. Obviously, none of these conditions is exactly satisfied in practice. Yet, this result motivates our approach to generate a vast subset $\mathcal{X}^A$ initially, to build a representative subset thereof Δ^P (via pooling) and to optimize the recourse function approximation (via SSS).

Note that this result relies on an assumption of Lipschitz continuity of the recourse function in each scenario. This condition is satisfied by two-stage stochastic linear programs with relatively complete, continuous, bounded and fixed recourse, i.e., with constraints of the form $\boldsymbol{W}\boldsymbol{y} = \boldsymbol{r}_s - \boldsymbol{U}_s\boldsymbol{x}$ where $\boldsymbol{y} \geq \boldsymbol{0}$ is a continuous variable and where the second-stage problem is feasible and bounded (**?**). However, this condition can fail in general, especially in the presence of discrete second-stage variables. Nonetheless, Theorem 1 provides a justification for the design of our scenario reduction approach and, as our computational results show, our approach generates high-quality solutions in the presence of continuous or discrete second-stage variables.

Let us conclude with a few remarks. First, the pooling process does not necessarily aim to build high-quality solutions in $\mathcal{X}^P$. Whereas, in theory, it could be sufficient to approximate the recourse function around the optimal region, insufficient variability may create instability in the recourse function approximation, hence high generalization error. Instead, the pooling procedure starts with a large set $\mathcal{X}^A$, and then builds a pool $\mathcal{X}^P$ to be representative of the solution space $\mathcal{X}$. To build the initial set $\mathcal{X}^A$, one possibility would be to proceed by random sampling; however, it can be challenging to generate random feasible solutions in constrained optimization. In our experiments, we will build $\mathcal{X}^A$ with $|\mathcal{S}|$ feasible solutions from deterministic scenario-based problems.

3.4. A heuristic for solving the SSS model

Our SSS problem (Equation (2)) reduces to the sparse regression problem in statistical learning, known to be NP-hard (**?**). Moreover, as described earlier, the "features" are highly collinear,

resulting in significant computational challenges. In our experiments, off-the-shelf implementations do not scale to even moderately large solution pools and scenario sets.

We thus design a bi-level heuristic that iterates between two local search operators (LSO) (Figure 2): an inner procedure (LSO I) that iteratively improves the solution within neighborhoods, and a perturbation scheme (LSO II) to escape from local optima. We iterate between LSO I and LSO II, until convergence. We outline the heuristic below, and detail the algorithm in Appendix A.4.

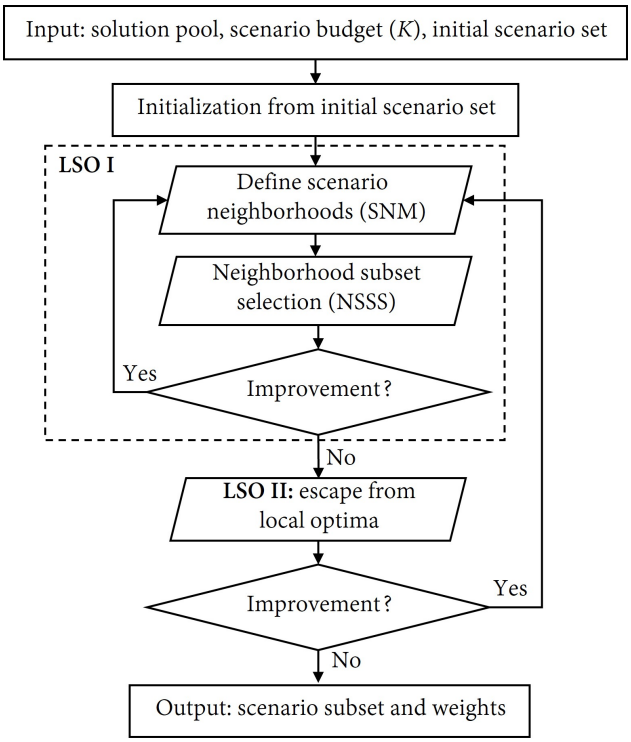


Figure 2 Overview of the bi-level heuristic, iterating between LSO I and LSO II.

LSO I. The inner local search procedure, itself, iterates until convergence between two modules:

1. Scenario-neighborhood model (SNM): expand each of the K scenarios into Ω "neighboring" ones. Let us index the incumbent solution by $\{\widehat{s}_1, \cdots, \widehat{s}_K\}$. This model builds K disjoint neighborhoods $\mathcal{S}_k \subseteq \mathcal{S}$, such that $\widehat{s}_k \in \mathcal{S}_k$ and $|\mathcal{S}_k| = \Omega$ for each $k = 1, ..., K$. The "neighbors" are defined based on the recourse values, namely: $D_{s\widehat{s}_k} = h_s \sum_{\boldsymbol{x} \in \mathcal{X}^P} \left| Q(\boldsymbol{x}, \xi_s) - Q(\boldsymbol{x}, \xi_{\widehat{s}_k}) \right|$. We define a variable α_{sk} equal to 1 if scenario s is included in neighborhood k, and 0 otherwise. We formulate SNM as follows, and retrieve each neighborhood as $\mathcal{S}_k = \{s \in \mathcal{S} : \alpha_{s,\widehat{s}_k} = 1\}$.

$$SNM(\mathcal{X}^P, \widehat{s}_1, \cdots, \widehat{s}_K, \Omega) = \min_{\boldsymbol{\alpha} \in \{0,1\}^{\mathcal{S} \times K}} \left\{ \sum_{s \in \mathcal{S}} \sum_{k=1}^{K} D_{s\widehat{s}_k} \alpha_{sk} : \quad \sum_{s \in \mathcal{S}} \alpha_{sk} = \Omega, \quad \forall k \in \{1, \cdots, K\}, \right.$$
$$\left. \sum_{k=1}^{K} \alpha_{sk} \leq 1, \quad \forall s \in \mathcal{S} \right\}. \tag{10}$$

2. Neighborhood-based scenario subset selection (NSSS): pick one scenario per neighborhood:

$$NSSS(\mathcal{X}^P, \mathcal{S}_1, \cdots, \mathcal{S}_K) = \min_{\boldsymbol{w} \geq \boldsymbol{0}} \left\{ \sum_{\boldsymbol{x} \in \mathcal{X}^P} \left| \sum_{s \in \mathcal{S}} h_s Q(\boldsymbol{x}, \xi_s) - \sum_{s \in \mathcal{S}} w_s Q(\boldsymbol{x}, \xi_s) \right|, \right. \tag{11}$$
$$\left. \text{s.t.} \sum_{s \in \mathcal{S}_k} \mathbb{1}(w_s \neq 0) = 1, \forall k \in \{1, \cdots, K\} \right\}.$$

We update the scenario subset as $\{s \in \mathcal{S} : w_s \neq 0\}$. By design, the objective is non-increasing across iterations. We continue until convergence, that is, until $w_{\widehat{s}_k} \neq 0$ for all $k = 1, \cdots, K$.

The parameter Ω defines the size of the neighborhood at each inner iteration. If Ω is small, each iteration will be fast, but the overall local search procedure can converge to a poor local optimum. If Ω is large, each iteration will be more impactful but also slower. We calibrate it in Appendix A.4.

LSO II. We perturb the solution from LSO I by fixing $K-1$ scenarios, and re-optimizing the remaining scenario and all the weights w_s. We still index the incumbent solution by $\{\widehat{s}_1, \cdots, \widehat{s}_K\}$, and define a partial scenario subset selection (PSSS) model as follows:

$$PSSS(\mathcal{X}^P, k, \widehat{s}_1, \cdots, \widehat{s}_K) = \min_{\boldsymbol{w} \geq \boldsymbol{0}} \left\{ \sum_{\boldsymbol{x} \in \mathcal{X}^P} \left| \sum_{s \in \mathcal{S}} h_s Q(\boldsymbol{x}, \xi_s) - \sum_{s \in \mathcal{S}} w_s Q(\boldsymbol{x}, \xi_s) \right|, \right. \tag{12}$$
$$\left. \text{s.t.} \sum_{s \in \mathcal{S} \setminus \{\widehat{s}_1, \cdots, \widehat{s}_{k-1}, \widehat{s}_{k+1}, \cdots, \widehat{s}_K\}} \mathbb{1}(w_s \neq 0) = 1 \right\}.$$

We repeat the process by re-optimizing each scenario from the incumbent solution. Whenever we obtain a (strict) solution improvement, we move to LSO I. The algorithm terminates when all LSO II re-optimizations do not find any improved solution over the sequence of K scenarios. As such, either the LSO II procedure improves the objective function or the algorithm terminates.

4. Lower bound: A scenario assortment optimization approach

We complement our scenario reduction approach with a scenario assortment optimization (SAO) approach to generate a lower bound of the full stochastic program $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$. We, again, impose that no instance of the stochastic program can be solved with more than K scenarios.

4.1. Scenario assortment optimization (SAO) formulation

Our SAO approach seeks bundles $(\mathcal{S}_b)_{b \in \mathcal{B}}$, each comprising up to K scenarios from $\mathcal{S}$. We define a parameter $\eta_b \geq 0$ for each bundle and introduce a mapping from scenarios to bundles, where $\beta_{bs} \geq 0$ denotes the weight of scenario $s \in \mathcal{S}$ into bundle $b \in \mathcal{B}$. We ensure that $\sum_{b \in \mathcal{B}} \eta_b = 1$ and $\sum_{b \in \mathcal{B}} \beta_{bs} = h_s$, that is, the total weight of each scenario across bundles remains unchanged. Note that each scenario $s \in \mathcal{S}$ can be "split" into multiple bundles, whenever $\beta_{bs} \in (0, 1)$.

We then solve the stochastic program within each bundle, and sum the cost across all bundles. This approach can be viewed as relaxing the nonanticipativity constraints across bundles. As Proposition 1 shows, it yields a lower bound of $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$.

PROPOSITION 1. *Let us construct bundles $(\mathcal{S}_b)_{b\in\mathcal{B}}$ such that $\mathcal{S}_b \subseteq \mathcal{S}$ for all $b \in \mathcal{B}$. We define non-negative vectors $(\eta_b)_{b\in\mathcal{B}}$ and $(\beta_{bs})_{b\in\mathcal{B},s\in\mathcal{S}}$ such that $\sum_{b\in\mathcal{B}} \eta_b = 1$ and $\sum_{b\in\mathcal{B}} \beta_{bs} = h_s$ for all $s \in \mathcal{S}$, and $\beta_{bs} = 0$ for all $s \notin \mathcal{S}_b, b \in \mathcal{B}$. Then, $\sum_{b\in\mathcal{B}} \mathcal{P}(\mathcal{S}_b, \eta_b, \boldsymbol{\beta}_b)$ is a lower bound to problem $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$, where:*

$$\mathcal{P}(\mathcal{S}_b, \eta_b, \boldsymbol{\beta}_b) = \min_{\boldsymbol{x}_b\in\mathcal{X}} \left\{ \eta_b \boldsymbol{c}^T \boldsymbol{x}_b + \sum_{s\in\mathcal{S}_b} \beta_{bs} Q(\boldsymbol{x}_b, \xi_s) \right\}, \quad \forall b \in \mathcal{B}. \tag{13}$$

The quality of this lower bound depends on the bundles $\mathcal{S}_b$ and the parameters η_b and β_{bs}. Our scenario assortment optimization (SAO) problem seeks the largest bound within this family:

$$\begin{aligned} \max_{\substack{(\mathcal{S}_b)_{b\in\mathcal{B}} \\ (\eta_b)_{b\in\mathcal{B}} \\ (\beta_{bs})_{b\in\mathcal{B},s\in\mathcal{S}}}} \Bigg\{ \sum_{b\in\mathcal{B}} \mathcal{P}(\mathcal{S}_b, \eta_b, \boldsymbol{\beta}_b) : \quad & \sum_{b\in\mathcal{B}} \eta_b = 1, \\ & \beta_{bs} = 0, \ \forall s \notin \mathcal{S}_b, b \in \mathcal{B}, \\ & |\mathcal{S}_b| \leq K, \ \forall b \in \mathcal{B}, \\ & \sum_{b\in\mathcal{B}} \beta_{bs} = h_s, \ \forall s \in \mathcal{S}, \\ & \mathcal{S}_b \subseteq \mathcal{S}, \ \forall b \in \mathcal{B}, \quad \eta_b \geq 0, \ \forall b \in \mathcal{B}, \quad \beta_{bs} \geq 0, \ \forall b \in \mathcal{B}, s \in \mathcal{S} \Bigg\}. \end{aligned} \tag{14}$$

Baseline: scenario grouping (SG). The SG approach from **?** also relaxes nonanticipativity constraints. However, SG can only partition the full scenario set $\mathcal{S}$ into subsets $\mathcal{S}_b$, whereas our SAO approach allows to "break down" each scenario into several bundles (through the continuous parameters η_b and β_{bs}). Mathematically, the SG baseline can be obtained by fixing the weights to the original probabilities h_s, that is, $\eta_b = \sum_{s\in\mathcal{S}_b} h_s$, and $\beta_{bs} = h_s \cdot \mathbf{1}(s \in \mathcal{S}_b)$, i.e.:

$$\begin{aligned} (SG) \quad & \max_{(\mathcal{S}_b)_{b\in\mathcal{B}}} \left\{ \sum_{b\in\mathcal{B}} \widetilde{P}(\mathcal{S}_b) : |\mathcal{S}_b| \leq K, \ \mathcal{S}_b \subseteq \mathcal{S}, \ \forall b \in \mathcal{B}, \ \cup_{b\in\mathcal{B}} \mathcal{S}_b = \mathcal{S}, \ \mathcal{S}_b \cap \mathcal{S}_{b'} = \emptyset, \ \forall b \neq b' \in \mathcal{B} \right\}, \\ & \text{where} \qquad \widetilde{P}(\mathcal{S}_b) = \min_{\boldsymbol{x}_b\in\mathcal{X}} \left\{ \sum_{s\in\mathcal{S}_b} h_s \boldsymbol{c}^T \boldsymbol{x}_b + \sum_{s\in\mathcal{S}_b} h_s Q(\boldsymbol{x}_b, \xi_s) \right\}, \ \forall b \in \mathcal{B}. \end{aligned} \tag{15}$$

Solution approaches. ? proved that SG is NP-hard as soon as $K \geq 3$ (hence, so is SAO). Computationally, SAO and SG are highly challenging since they involve stochastic programs in the objective function. Therefore, we reformulate them as structured optimization problems and propose two decomposition algorithms to solve them:

1. *Row generation benchmark (**?**).* Our first reformulation enumerates all first-stage solutions in $\mathcal{X}$, with an exponential number of constraints for a first-stage linear programming or discrete optimization structure. Accordingly, we design a row generation algorithm that iterates between a master problem (selecting bundles with a restricted set of first-stage solutions) and a subproblem (expanding the set of first-stage solutions by solving a stochastic program for each bundle). This benchmark is detailed in Appendix B.1.

2. *Column evaluation and generation.* We propose a set partitioning reformulation that enumerates all possible bundles. One challenge is that the cost parameters are themselves the solutions of stochastic programs, hence hard to estimate. We propose a novel column evaluation scheme, which iterates between an optimization step (to select bundles based on cost approximations) and a column evaluation step (to update the cost parameters for selected bundles). We prove that this algorithm converges to an optimum of any (solvable) optimization model. Another challenge is that our set partitioning formulation has an infinite number of variables (due to the continuous variables η_b and β_{bs}). We thus propose a column-evaluation-and-generation approach that embeds our column evaluation scheme into a column generation scheme.

4.2. Column-evaluation-and-generation algorithm

Set partitioning. Proposition 2 reformulates the SAO problem as a set partitioning problem.

PROPOSITION 2. *Let $\mathcal{B}^{all}$ be the set of all feasible bundles, with parameters η_b and $\boldsymbol{\beta}_b$. For each $b \in \mathcal{B}^{all}$, let O_b be the optimal objective of $\mathcal{P}(\mathcal{S}_b, \eta_b, \boldsymbol{\beta}_b)$ (Equation (13)). Let $y_b \geq 0$ be a variable denoting the weight assigned to each bundle $b \in \mathcal{B}^{all}$. Equation (14) is equivalent to:*

$$\max \quad \sum_{b \in \mathcal{B}^{all}} O_b y_b \tag{16}$$

$$s.t. \quad \sum_{b \in \mathcal{B}^{all}} \eta_b y_b = 1 \tag{17}$$

$$\sum_{b \in \mathcal{B}^{all}} \beta_{bs} y_b = h_s \quad \forall s \in \mathcal{S} \tag{18}$$

$$y_b \geq 0 \quad \forall b \in \mathcal{B}^{all}. \tag{19}$$

Let $(y_b^*)_{b \in \mathcal{B}^{all}}$ denote the optimal solution of Equations (16)–(19). We can transform it into the SAO solution, by letting $\mathcal{B} = \{b \in \mathcal{B}^{all} : y_b^* > 0\}$ and $\mathcal{P}(\mathcal{S}_b, y_b^* \eta_b, y_b^* \boldsymbol{\beta}_b) = y_b^* \mathcal{P}(\mathcal{S}_b, \eta_b, \boldsymbol{\beta}_b),\ \forall b \in \mathcal{B}$.

This formulation comes with two challenges (i) estimating the objective parameters O_b is computationally intensive, requiring to solve a stochastic programming model with K scenarios (Equation (13)); and (ii) the set partitioning model is formulated as a semi-infinite optimization model due to the infinite number of decision variables $(y_b)_{b \in \mathcal{B}^{all}}$, itself due to the continuous variables η_b and β_{bs}. These two challenges motivate our solution algorithm, which combines column evaluation (in response to challenge (i)) and column generation (in response to challenge (ii)).

Column evaluation. Note that we can easily derive an upper bound of O_b by solving a simplified stochastic program over a subset $\mathcal{X}^{\mathcal{R}} \subseteq \mathcal{X}$, of limited cardinality—so optimizing over $\mathcal{X}^R$ is computationally efficient. We then define a relaxation-approximation $\widehat{O}_b$ as follows:

$$\widehat{O}_b = \min_{\boldsymbol{x}_b \in \mathcal{X}^{\mathcal{R}}} \left\{ \eta_b \boldsymbol{c}^T \boldsymbol{x}_b + \sum_{s \in \mathcal{S}_b} \beta_{bs} Q(\boldsymbol{x}_b, \xi_s) \right\} \geq O_b, \quad \forall b \in \mathcal{B}^{all}. \tag{20}$$

Accordingly, our column evaluation scheme iteratively *solves* the set partitioning formulation with a relaxation-approximation of the objective function, and *evaluates* the true objective parameters O_b for all selected bundles. It keeps track of an approximated parameter $\widetilde{O}_b \in \left\{O_b, \widehat{O}_b\right\}$ for each bundle $b \in \mathcal{B}^{all}$, and updates $\widetilde{O}_b$ by O_b for each selected bundle. As such, this approach calls the "oracle" (to estimate O_b) with parsimony—for selected bundles as opposed to all bundles.

Specifically, the proposed column evaluation procedure alternates between two modules, updating at each iteration a relaxation bound and a lower bound.[1]

1. *optimization:* solve set partitioning model with the current estimates of objective parameters:

$$\max \quad \sum_{b \in \mathcal{B}^{all}} \widetilde{O}_b y_b, \qquad \text{s.t.} \quad (17)\text{–}(19). \tag{21}$$

   Let $\widetilde{\boldsymbol{y}}$ be the optimal solution and $\widetilde{\mathcal{B}} \subset \mathcal{B}^{all}$ be the set of bundles such that $\widetilde{y}_b > 0$. For any feasible solution $\boldsymbol{y}$ of Equations (16)–(19), we have $\sum_{b \in \mathcal{B}^{all}} \widetilde{O}_b \widetilde{y}_b \geq \sum_{b \in \mathcal{B}^{all}} \widetilde{O}_b y_b \geq \sum_{b \in \mathcal{B}^{all}} O_b y_b$, due to the optimality of $\widetilde{\boldsymbol{y}}$ and the fact that $\widetilde{O}_b \geq O_b$. We obtain the following relaxation bound (RB), which yields an upper bound of the SAO problem:

$$RB = \sum_{b \in \mathcal{B}^{all}} \widetilde{O}_b \widetilde{y}_b.$$

2. a column evaluation step, to replace the parameters $\widetilde{O}_b$ with the true parameters O_b for all selected bundles $b \in \widetilde{\mathcal{B}}$. To this end, we solve the stochastic program:

$$\widetilde{O}_b \leftarrow O_b = \min_{\boldsymbol{x}_b \in \mathcal{X}} \left\{ \eta_b \boldsymbol{c}^T \boldsymbol{x}_b + \sum_{s \in \mathcal{S}_b} \beta_{bs} Q(\boldsymbol{x}_b, \xi_s) \right\} \text{ for all } b \in \mathcal{B}^{all} \text{ such that } \widetilde{y}_b > 0. \tag{22}$$

   From Proposition 1, the following expression is a valid lower bound (LB) of the SAO problem, hence a lower bound of the full stochastic program $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$:

$$LB = \sum_{b \in \widetilde{\mathcal{B}}} O_b \widetilde{y}_b.$$

As such, the algorithm provides a feasible solution and an optimality gap for the SAO problem. The critical observation is that, if $\widetilde{O}_b = O_b$ for all selected bundles $b \in \widetilde{\mathcal{B}}$, then $\widetilde{\boldsymbol{y}}$ is an optimal solution of Equations (16)–(19). Theorem 2 establishes that the column evaluation procedure converges finitely to the optimal solution of any (solvable) optimization model.

THEOREM 2. *Define* $OPT(\boldsymbol{c}) = \max\{\boldsymbol{c}^\top \boldsymbol{x} : \ s.t. \ \boldsymbol{x}, \boldsymbol{x} \in \mathcal{X}\}$. *Let* $\widehat{\boldsymbol{c}}$ *be a vector satisfying* $\widehat{c}_i \geq c_i, \forall i \in \{1, ..., n\}$. *The following algorithm is finitely convergent to the optimum of* $OPT(\boldsymbol{c})$.

[1] The relaxation bound can be viewed as "an upper bound of the SAO lower bound". To avoid confusion with the upper bound of the full stochastic program derived in Section 3, we refer to it as "relaxation bound".

1. *Initialize* $\widetilde{c} = \widehat{c}$.
2. *Solve* $OPT(\widetilde{\boldsymbol{c}})$ *and retrieve its optimal solution* $\widetilde{\boldsymbol{x}}$.
3. *Update* $\widetilde{c}_i \leftarrow c_i$ *for all* $i \in \{1, ..., n\}$ *such that* $\widetilde{x}_i > 0$ *and* $\widetilde{c}_i > c_i$.
4. *If* $\widetilde{c}_i = c_i$ *for all* $i \in \{1, ..., n\}$ *such that* $\widetilde{x}_i > 0$, *stop. Otherwise, go to Step 2.*

Although valid for any optimization model, this result is particularly useful for problems with a very large number of decision variables, so estimating all objective parameters is prohibitively expensive and the fraction of "active" solutions (with $\widetilde{x}_i > 0$) is small. In particular, this can be the case for set partitioning formulations with an exponential number of variables. This is especially relevant in our context, where the set partitioning formulation involves an infinite number of variables and each objective parameter is the solution of a stochastic program.

The next question is how to solve the semi-infinite set partitioning model (Equation (21)) at each iteration. This is the purpose of our column generation algorithm.

Column generation. Our column generation algorithm iterates between (i) a restricted master problem (RMP) with a limited set of bundles $\widetilde{\mathcal{B}}^{all} \subseteq \mathcal{B}^{all}$, and (ii) a pricing problem (PP) that adds new bundles to the set $\widetilde{\mathcal{B}}^{all}$. The restricted master problem is formulated as follows:

$$(RMP) \quad \max \quad \sum_{b \in \widetilde{\mathcal{B}}^{all}} \widetilde{O}_b y_b \tag{23}$$

$$\text{s.t.} \quad \sum_{b \in \widetilde{\mathcal{B}}^{all}} \eta_b y_b = 1 \tag{24}$$

$$\sum_{b \in \widetilde{\mathcal{B}}^{all}} \beta_{bs} y_b = h_s \quad \forall s \in \mathcal{S} \tag{25}$$

$$y_b \geq 0 \quad \forall b \in \widetilde{\mathcal{B}}^{all}. \tag{26}$$

Let $\pi \in \mathbb{R}$ and $\lambda_s \in \mathbb{R}$ denote the dual variables of the master problem corresponding to constraints (24) and (25), respectively. The pricing problem is formulated as follows:

$$PP(\mathcal{X}^R, \pi, \boldsymbol{\lambda}) = \quad \max \quad \widetilde{O} - \pi\eta - \sum_{s \in \mathcal{S}} \lambda_s \beta_s \tag{27}$$

$$\text{s.t.} \quad \widetilde{O} \leq \eta \boldsymbol{c}^T \boldsymbol{x} + \sum_{s \in \mathcal{S}} \beta_s Q(\boldsymbol{x}, \xi_s) \quad \forall \boldsymbol{x} \in \mathcal{X}^R \tag{28}$$

$$\sum_{s \in \mathcal{S}} \theta_s \leq K \tag{29}$$

$$\beta_s \leq h_s \theta_s \quad \forall s \in \mathcal{S} \tag{30}$$

$$\theta_s \in \{0, 1\}, \beta_s \geq 0, \eta \geq 0, O \in \mathbb{R} \quad \forall s \in \mathcal{S}. \tag{31}$$

We iterate between the restricted master problem and the pricing problem, until the pricing problem returns a non-positive reduced cost. At that point, the solution of the restricted master

problem solves the full set partitioning problem (Equation (21)) to optimality. However, the set partitioning formulation involves an infinite number of variables. This differs from traditional column generation settings, which involve an exponential, but finite number of variables (**??**). Therefore, the column generation algorithm does not guarantee finite convergence. Yet, we consistently obtain strong convergence empirically (Section 6).

Column-evaluation-and-generation algorithm (Algorithm 1). The algorithm involves an outer loop and an inner loop. Outer iterations involve column evaluation, alternating between solving the set partitioning model and updating the objective parameters of all selected bundles. Inner iterations relate to column generation, alternating between the restricted master problem and the pricing problem. The inner loop terminates when the pricing problem does not return any new bundle with a positive reduced cost.[2] The outer loop terminates when all selected bundles are evaluated with their true objective parameters. Meanwhile, the algorithm maintains a lower bound LB, which provides, at any point in time, a solution guarantee for the full stochastic program.

Algorithm 1 Column-evaluation-and-generation for scenario assortment optimization.

Input: Initial set $\mathcal{X}^R$, initial set $\widetilde{\mathcal{B}}^{all}$, $\widetilde{O}_b = \widehat{O}_b,\ \forall b \in \widetilde{\mathcal{B}}^{all}$, $RB = +\infty$, $LB = -\infty$, tolerance ε.

while $\frac{RB-LB}{LB} > \varepsilon$ or $LB = -\infty$ **do** ▷ Outer loop: column evaluation
  while NOT(`terminate`) **do** ▷ Inner loop: column generation
    Solve RMP (Equations (23)–(26)) $\longrightarrow$ solution $\widetilde{\boldsymbol{y}}$, active bundles $\widetilde{\mathcal{B}}$, dual variables $\pi, \boldsymbol{\lambda}$
    Solve PP (Equations (27)–(31)) $\longrightarrow$ $b_0 = \arg\max_{b \in \mathcal{B}^{all}} \left(\widetilde{O}_b - \pi\eta_b - \sum_{s\in\mathcal{S}} \lambda_s \beta_{bs} \right)$
    If $\widetilde{O}_{b_0} - \pi\eta_{b_0} - \sum_{s\in\mathcal{S}} \lambda_s \beta_{b_0,s} > 0$, **then** update $\widetilde{\mathcal{B}}^{all} \leftarrow \widetilde{\mathcal{B}}^{all} \cup \{b_0\}$
    Else, update $RB = \sum_{b\in\widetilde{\mathcal{B}}^{all}} \widetilde{O}_b y_b^*$ and update `terminate` = TRUE
  end while
  Update $\widetilde{O}_b \leftarrow O_b = \mathcal{P}(\mathcal{S}_b, \eta_b, \boldsymbol{\beta}_b)$, for all $b \in \widetilde{\mathcal{B}}$, $\longrightarrow$ solution $\boldsymbol{x_b^*}$, for all $b \in \widetilde{\mathcal{B}}$
  Update $LB \leftarrow \max\left\{\sum_{b\in\widetilde{\mathcal{B}}} O_b \widetilde{y}_b, LB\right\}$
  Expand restricted solution space $\mathcal{X}^R \leftarrow \mathcal{X}^R \cup \{\boldsymbol{x}_b^*,\ b \in \widetilde{\mathcal{B}}\}$
  Update $\widetilde{O}_b \leftarrow \min\left\{\eta_b \boldsymbol{c}^T\boldsymbol{x} + \sum_{s\in\mathcal{S}} \beta_{bs} Q(\boldsymbol{x}, \xi_s), \boldsymbol{x} \in \mathcal{X}^R\right\}$, for all $b \in \widetilde{\mathcal{B}}^{all} \setminus \widetilde{\mathcal{B}}$
end while

Let us conclude with some remarks on the subset $\mathcal{X}^{\mathcal{R}} \subset \mathcal{X}$. First, benchmark algorithms (such as the row generation from **?**) expand iteratively the subset $\mathcal{X}^R$ to guarantee convergence. Our column-evaluation scheme reduces the reliance on $\mathcal{X}^R$, by guaranteeing convergence for any subset $\mathcal{X}^R$. The next question is how to design this subset $\mathcal{X}^R$ in our algorithm. A large subset $\mathcal{X}^{\mathcal{R}}$ results in more computationally intensive iterations; vice versa, a small subset $\mathcal{X}^{\mathcal{R}}$ makes the relaxation

[2] Since the column generation algorithm is not guaranteed to terminate finitely for our semi-infinite set partitioning formulation, an alternative termination criterion could be applied. We have not faced this situation in our experiments.

approximation looser, leading to more iterations. We propose an acceleration strategy based on *adaptive relaxation*, starting with a small subset $\mathcal{X}^{\mathcal{R}}$ and expanding it at each outer iteration with all stochastic programming solutions (one for each active bundle). This approach (Line 10 of Algorithm 1) tightens the relaxation approximation around "promising" solutions.

In summary, our scenario assortment optimization approach (SAO) relaxes the nonanticipativity constraints to derive a lower bound of the full stochastic program, and the column-evaluation-and-generation algorithm computes the tightest possible bound within this family. Ultimately, this paper thus provides a solution and quality guarantee for large-scale stochastic programs, without solving any instance with more than K scenarios. Moreover, the column-evaluation-and-generation algorithm provides a stand-alone contribution of this paper to solve optimization problems where the objective parameters are hard to estimate but a relaxation-approximation thereof (i.e., an upper bound in maximization problems, a lower bound in minimization problems) is easily available.

5. Experimental setup

We now evaluate our approach to scenario reduction and scenario assortment optimization in extensive computational case studies. All optimization models are solved with CPLEX 12.5 on a workstation with a 32-core Intel Xeon CPU (3.0 GHz) and 128 GB RAM.

5.1. Problem settings

We evaluate our methods in two-stage production routing and facility location settings. For each one, we define formulations with continuous and mixed-integer recourse. We thus consider four problems: production routing problem with continuous recourse (PRP–CR), production routing problem with mixed-integer recourse (PRP–IR), facility location problem with continuous recourse (FLP–CR) and facility location problem with mixed-integer recourse (FLP–IR).

These four problems complement each other. First, stochastic programs with mixed-integer recourse are much more challenging than with continuous recourse. Second, the two case study settings exhibit different sources of complexity: the PRP–CR and PRP–IR are medium-scale problems (with hundreds of thousands of variables and constraints) but highly challenging due to the combination of plant setup, routing, and inventory control decisions; in comparison, the FLP–CR and FLP–IR are hard due to their very large scale (millions of variables and constraints).

Production routing problem. We consider a two-stage production routing problem (PRP) under demand uncertainty. The PRP jointly optimizes production decisions (e.g., plant setup, production quantities, and product inventories) and distribution decisions (e.g., routing). The two variants are defined as follows (see Appendix C.1 for their formulation):

– PRP–CR: The decision-maker optimizes plant setup and vehicle routing decisions in the first stage (before demand is observed), as well as production quantities, inventory management, and distribution decisions in the second stage (**?**). We formulate the routing problem using two-commodity network flows, inspired by **?**.
– PRP–IR: The problem is equivalent to PRP–CR, except that routing decisions are made in the second stage. That is, the decision-maker can adjust vehicle routes, along with production, inventory and distribution plans, upon observing customer demand. This PRP variant is highly relevant in practice, since firms often commit to customer appointments early on but not necessarily to vehicle routes. However, it has not been tackled in the literature.

Facility location problem. We consider a two-stage facility location problem (FLP) under demand uncertainty. The variants are defined as follows (see Appendix C.2 for their formulation):
– FLP–CR: The decision-maker optimizes facility setup in the first stage (before customer demand is observed), and commodity distribution in the second stage.
– FLP–IR: The decision-maker optimizes facility setup in the first stage. Upon observing customer demand, the decision-maker activates facilities along with optimizing distribution.

5.2. Problem complexity

For PRP–CR and PRP–IR, we generate problem instances following **?**, with 10 customers, 10 time periods, 3 vehicles and 500 equiprobable scenarios. For FLP–CR and FLP–IR, we generate problem instances with 300 customers, 200 candidate facilities and 500 equiprobable scenarios (see Appendix C.2). The PRP–CR and PRP–IR comprise around 650,000 decision variables and 500,000 constraints; the FLP–CR and FLP–IR comprise around 30 million decision variables and constraints. We make all instances available in the online supplement.

From a stochastic standpoint, this setup leads to challenging problems that cannot be solved with simple deterministic variants. Indeed, Table 1 shows that, as compared to the best-known solution, scenario-based solutions lead to expected out-of-sample gaps of 36.2–139.6% for the PRP–CR, of 21.6–84.3% for the PRP–IR, of 5.5–478.8% for the FLP–CR, and of 3.6–552.4% for the FLP–IR.

Table 1 Performance of optimized vs. scenario-based solutions.

		Scenario-based solutions					
		Best		Median		Worst	
Problem	Best known	Solution	Gap	Solution	Gap	Solution	Gap
PRP–CR	502,080	683,699	36.17%	862,776	71.84%	1,202,730	139.55%
PRP–IR	752,394	914,564	21.55%	1,093,900	45.39%	1,386,622	84.29%
FLP–CR	7,763	8,188	5.48%	20,382	162.57%	44,913	478.81%
FLP–IR	6,887	7,132	3.56%	20,240	193.88%	44,931	552.40%

Before proceeding, we start by attempting to solve each problem with existing stochastic programming algorithms. For problems with continuous recourse, we consider (i) direct CPLEX implementation, (ii) Benders decomposition, and (iii) enhanced Benders decomposition (E-Benders), using the lower bound lifting inequalities from **?** for PRP–CR and Pareto-optimality cuts from **?** and **?** for FLP–CR. For problems with mixed-integer recourse, we consider (i) direct CPLEX implementation, (ii) the integer L-shaped method from **?**, and (iii) an enhanced integer L-shaped method (E-L-shaped) that leverages standard Benders cuts, Pareto-optimality cuts, as well as, for FLP–IR, tailored L-shaped cuts from **?**. Collectively, these methods span traditional stochastic programming algorithms (direct CPLEX implementation, Benders decomposition, integer L-shaped method) and recent advanced ones (E-Benders, E-L-shaped).

Table 2 shows that the four problems under consideration are highly challenging from a computational standpoint as well. Direct CPLEX implementation scales to 30 scenarios for PRP–CR and FLP–CR. For FLP–IR and PRP–IR, CPLEX does not consistently find the optimal solution within the 5-hour limit with 15 scenarios. Next, Benders decomposition and the integer L-shaped method cannot derive optimal solutions with as few as 10 scenarios for PRP–IR, FLP–CR and FLP–IR. With the full scenario set, CPLEX, Benders decomposition and the integer L-shaped methods all fail to provide any reasonable solution and leave very large optimality gaps. The E-Benders and E-L-shaped methods scale to 30 scenarios for PRP–CR, FLP–CR and FLP–IR, but still leave an optimality gap of 1.3%–40.2% with the full scenario set. Overall, "simple" stochastic programming methods (Benders, L-shaped) do not scale beyond 15–30 scenarios and "advanced" ones (E-Benders, E-L-shaped) cannot tackle the full problem with 500 scenarios.

Table 2 Computational performance of stochastic programming algorithms.

		10 scenarios		15 scenarios		20 scenarios		25 scenarios		30 scenarios		500 scenarios			
Problem	Benchmark	CPU	Opt	CPU	Opt	CPU	Opt	CPU	Opt	CPU	Opt	CPU	Solution	LB	Gap
PRP–CR	CPLEX	2	10	5	10	8	10	12	10	14	10	>300	2,230,621	449,730	79.8%
	Benders	7	10	9	10	14	10	15	10	19	10	>300	546,032	393,345	28.0%
	E-Benders	4	10	6	10	11	10	11	10	12	10	>300	505,636	490,748	2.9%
PRP–IR	CPLEX	21	10	85	9	195	6	183	6	254	3	>300	2,214,462	683,183	69.1%
	L-shaped	>300	0	>300	0	>300	0	>300	0	>300	0	>300	2,230,621	6,000	99.7%
	E-L-shaped	>300	0	>300	0	>300	0	>300	0	>300	0	>300	862,682	515,559	40.2%
FLP–CR	CPLEX	6	10	17	10	32	10	57	10	76	10	>300	44,931	0	100.0%
	Benders	>300	0	>300	0	>300	0	>300	0	>300	0	>300	37,298	557	98.5%
	E-Benders	10	10	16	10	21	10	24	10	35	10	>300	7,770	7,668	1.3%
FLP–IR	CPLEX	38	10	78	7	151	4	260	2	>300	0	>300	44,931	0	100.0%
	L-shaped	>300	0	>300	0	>300	0	>300	0	>300	0	>300	37,565	304	99.2%
	E-L-shaped	23	10	59	10	92	10	110	10	141	10	>300	6,957	6,786	2.5%

Results are averaged over 10 random samples of scenarios, out of a full set of 500 scenarios.
We report the average computational time (CPU, in minutes) and the number of instances solved to 1% optimality gap within 5 hours (Opt). With the full scenario set, we also report the solution, lower bound (LB), and optimality gap after 5 hours.

6. Computational results

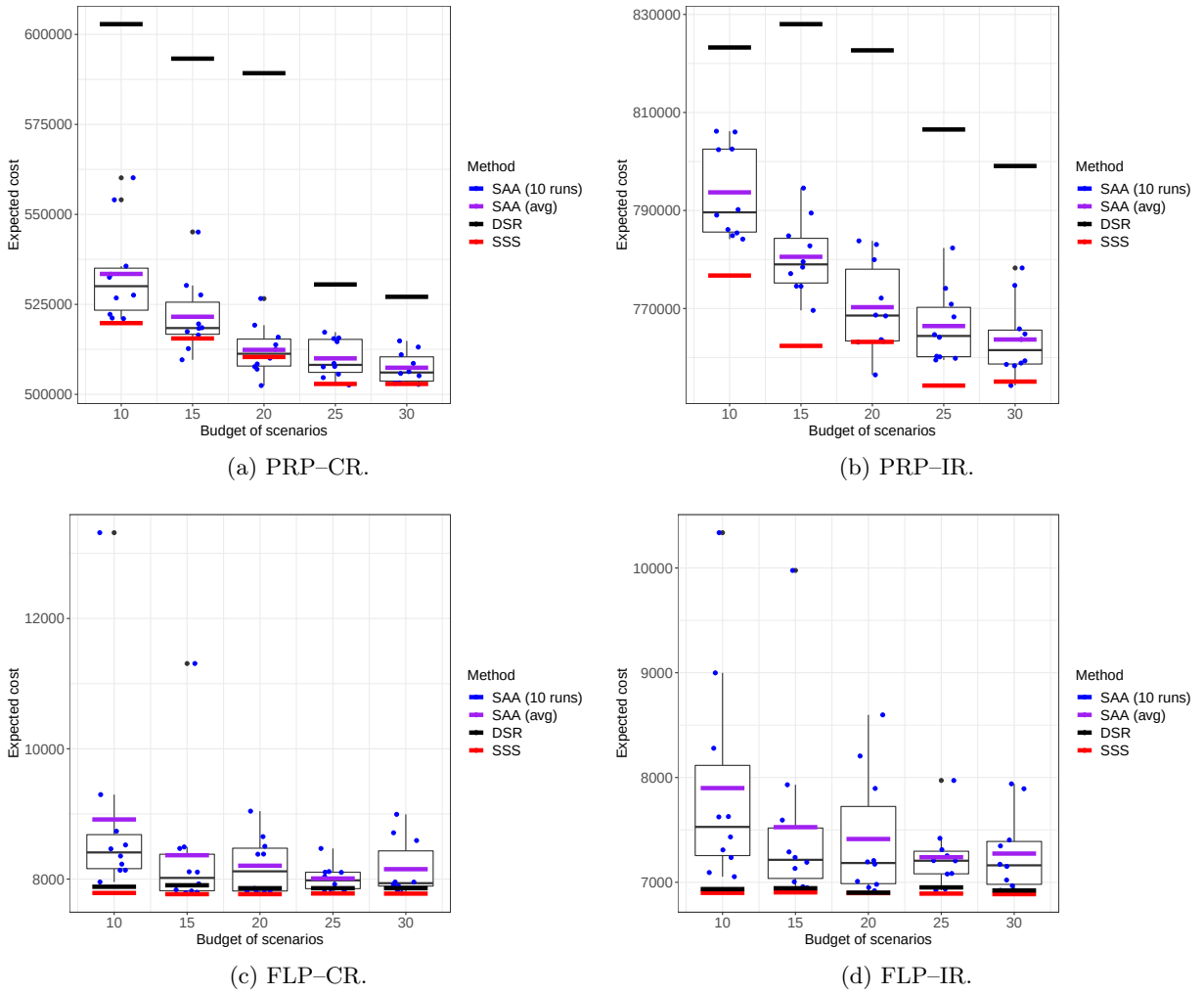
This section first compares the solutions from our optimization-based scenario reduction approach to SAA and DSR. Second, we evaluate our scenario assortment optimization approach, along with our column-evaluation-and-generation algorithm. Third, we assess our results against stochastic programming benchmarks in terms of solutions, lower bounds, and computational times.

6.1. Results from scenario reduction: evaluation of solution quality

To apply our SSS approach, we generate 500 solutions in $\mathcal{X}^A \subseteq \mathcal{X}$ by solving one deterministic problem in each scenario, i.e., $\mathcal{X}^A = \{\boldsymbol{x}_s^* : s \in \mathcal{S}\}$, where $\boldsymbol{x}_s^* \in \arg\min_{\boldsymbol{x}\in\mathcal{X}} \{\boldsymbol{c}^T\boldsymbol{x} + Q(\boldsymbol{x}, \xi_s)\}, \forall s \in \mathcal{S}$ (see Appendix A.3 for additional results on this point). By default, we consider 90 solutions in $\mathcal{X}^P$ (see Appendix A.3), we set the parameter Ω in our heuristic so that $\Omega \times K \sim 100$ (see Appendix A.4).

Main results. The SSS, DSR, and SAA methods yield a scenario subset $\mathcal{S}_0$ of cardinality K and corresponding weights $\boldsymbol{w}$. We then solve the stochastic programming problem $\mathcal{P}(\mathcal{S}_0, \boldsymbol{w})$, using the most efficient algorithm for the problem and the scenario budget under consideration, shown in bold in Table 2. Specifically, we use CPLEX with 10–20 scenarios and E-Benders with 25 or more scenarios for the PRP–CR; we use CPLEX with 10 scenarios and E-Benders with 15 or more scenarios for the FLP–CR; we always use CPLEX for the PRP–IR; and we always use E-L-shaped for the FLP–IR. We evaluate the out-of-sample performance of SSS, DSR, and SAA. Figure 3 reports the out-of-sample expected costs, as a function of the number K of scenarios in $\mathcal{S}_0$. Given the stochasticity of SAA, we run it 10 times for each value of K, and plot the average cost as well its variability (via a box plot). To complement these observations, Table 3 reports the out-of-sample cost for each method, as well as the variability over 10 runs for SAA (median, minimum, maximum). In addition, we use the 10 runs of SAA to compute a statistical lower bound at the 95% confidence level (via a one-sided t-test) as well as a corresponding statistical optimality gap.

These results show that our SSS method provides significant improvements, as compared to both benchmarks, and that these benefits are robust to the problem instance and the number of scenarios. First, the SSS method reduces expected out-of-sample costs by up to 12.7% as compared to SAA. In 14 out of 20 instances, SSS results in even stronger solutions than the best SAA solution (out of 10 runs), by up to 2.1%. Second, it reduces expected costs by up to 13.8% as compared to DSR. That is, approximating the input distributions does not necessarily yield the best solutions to the downstream optimization. It is interesting to note, also, that DSR outperforms SAA for FLP but not for PRP. This observation can be interpreted as a trade-off between generating "central" scenarios (via DSR) for relatively stable problems (like FLP) versus generating "diverse" scenarios (via randomization) for less stable problems (like PRP). Regardless, SSS provides strong and robust benefits, with performance improvements across all problem instances.



(a) PRP–CR.

(b) PRP–IR.

(c) FLP–CR.

(d) FLP–IR.

Figure 3 Performance of all scenario reduction methods.

Finally, turning to the (statistical) optimality gap from SAA, we observe that, as expected, the SAA upper bound decreases and the statistical lower bound increases as the number of scenarios increases, resulting in a shrinking optimality gap. Yet, the statistical optimality gap remains significant: 8.5–12.5% for PRP and over 29% for FLP with 10 scenarios; 2.2–3.5% for PRP and 13.5–15.3% for FLP with 30 scenarios. These results motivate our scenario assortment optimization approach to generate tighter (and exact) solution guarantees (Section 6.2).

Computational budget. We now compare the three scenario reduction methods with fixed computational budgets. For SSS, the computational time includes both the scenario reduction time and the stochastic programming time. For DSR, we only account for the stochastic programming time—ignoring the scenario reduction time. For SAA, we define four benchmarks. In the first three, termed "SAA(10)", "SAA(20)" and "SAA(30)", we repeatedly sample 10, 20, and 30 scenarios, respectively, and solve the resulting stochastic program, allowing for multiple replications until the

Table 3 Summary statistics comparing DSR, SAA and SSS.

				SAA						SSS assessment	
Problem	K	SSS	DSR	Average	Median	Min.	Max.	Bound	Gap	vs. DSR	vs. SAA
PRP–CR	10	519,799	602,861	533,457	530,053	521,020	560,161	480,200	8.5%	−13.8%	−2.6%
	15	515,534	593,268	521,562	518,420	509,593	545,126	487,910	4.4%	−13.1%	−1.2%
	20	510,392	589,226	512,365	511,277	502,444	526,616	487,407	3.1%	−13.4%	−0.4%
	25	502,904	530,533	509,998	508,192	502,612	517,288	489,791	2.6%	−5.2%	−1.4%
	30	502,903	527,098	507,402	506,074	502,767	514,859	491,762	2.2%	−4.6%	−0.9%
PRP–IR	10	776,723	823,264	793,696	789,624	784,150	806,175	697,001	12.5%	−5.7%	−2.1%
	15	762,369	828,029	780,548	778,998	769,625	794,570	712,844	8.0%	−7.9%	−2.3%
	20	763,182	822,674	770,262	768,564	756,458	783,795	718,282	5.3%	−7.2%	−0.9%
	25	754,271	806,528	766,417	764,394	759,503	782,348	722,186	5.2%	−6.5%	−1.6%
	30	755,081	799,079	763,670	761,501	754,294	778,268	728,629	3.5%	−5.5%	−1.1%
FLP–CR	10	7,787	7,882	8,915	8,409	7,956	13,317	6,102	30.4%	−1.2%	−12.7%
	15	7,769	7,905	8,366	8,018	7,791	11,309	6,507	19.7%	−1.7%	−7.1%
	20	7,769	7,860	8,205	8,119	7,793	9,043	6,455	20.7%	−1.2%	−5.3%
	25	7,779	7,860	8,009	7,981	7,802	8,470	6,917	12.8%	−1.0%	−2.9%
	30	7,777	7,860	8,152	7,937	7,788	8,994	6,754	15.3%	−1.1%	−4.6%
FLP–IR	10	6,899	6,935	7,899	7,529	7,054	10,337	5,463	29.1%	−0.5%	−12.7%
	15	6,904	6,943	7,526	7,215	6,948	9,977	5,926	17.2%	−0.6%	−8.3%
	20	6,897	6,901	7,413	7,184	6,919	8,598	5,729	20.8%	−0.1%	−7.0%
	25	6,892	6,951	7,239	7,206	6,924	7,972	6,249	10.8%	−0.8%	−4.8%
	30	6,887	6,922	7,274	7,162	6,923	7,940	6,099	13.5%	−0.5%	−5.3%

computational budget is exhausted. The fourth one, "SAA(hybrid)", applies 10-scenario SAA, 20-scenario SAA, and 30-scenario SAA, looping until the computational budget is exhausted. Table 4 reports the SSS solution, the DSR solution and the SAA solutions, all evaluated out of sample.

Table 4 Comparison of DSR, SAA and SSS under the same computational time budget.

	Time	SSS	SAA(10)	SAA(20)	SAA(30)	SAA(hybrid)	SAA(best)	vs. SSS	DSR	vs. SSS
PRP–CR	0.5 h	519,799	522,017	512,023	507,523	506,014	SAA(hybrid)	−2.7%	527,098	+1.4%
	1 h	502,903	521,102	507,456	505,148	504,134	SAA(hybrid)	+0.2%	527,098	+4.8%
	1.5 h	502,903	521,062	505,200	503,600	503,578	SAA(hybrid)	+0.1%	527,098	+4.8%
PRP–IR	1 h	776,723	793,222	—	—	793,046	SAA(10)	+2.1%	823,264	+6.0%
	3 h	762,369	785,078	770,023	—	770,832	SAA(20)	+1.0%	822,674	+7.9%
	5 h	754,271	784,423	765,217	763,829	770,832	SAA(30)	+1.3%	799,079	+5.9%
FLP–CR	0.5 h	7,787	8,091	7,976	8,153	8,116	SAA(20)	+2.4%	7,860	+0.9%
	1 h	7,769	8,051	7,883	7,965	7,943	SAA(20)	+1.5%	7,860	+1.2%
	1.5 h	7,777	8,031	7,849	7,892	7,846	SAA(hybrid)	+0.9%	7,860	+1.1%
FLP–IR	1 h	6,899	7,407	—	—	7,887	SAA(10)	+7.4%	6,935	+0.5%
	3 h	6,897	7,146	7,448	—	7,235	SAA(10)	+3.6%	6,901	+0.1%
	5 h	6,887	7,097	7,125	7,284	7,040	SAA(hybrid)	+2.2%	6,922	+0.5%

First, our SSS method reduces expected costs over even the best SAA strategy, in 11 out of 12 cases. That is, even if one picked the "best" SAA strategy (which is challenging by itself given the variability of the "best" SAA strategy) and let the SAA algorithm run for the entire time budget, the best solution would remain inferior to the one obtained with our SSS method—with a cost differential of up to 7.4%. Similarly, the DSR method consistently results in higher costs than our SSS method. The cost increases remain moderate for FLP but can be significant for PRP (up to

7.9%). Ultimately, these results show that our SSS model provides significant and robust benefits over the SAA and DSR benchmarks, even after accounting for the scenario reduction time.

Effectiveness of SSS heuristic. Table 5 compares our heuristic given in Algorithm 2 ("SSS heuristic") to direct CPLEX ("SSS CPLEX") toward solving the scenario subset selection (SSS) model. For both methods, the table reports the solution of the SSS model (SSS.Sol), the computational time (CPU, in minutes), the expected out-of-sample cost (Solution), and the difference to the average SAA solution (%SAA). We also report the difference of the SSS solutions (%SSS.Sol) and PRP solutions (%Solution) between SSS CPLEX and SSS heuristic. For SSS CPLEX, we impose a maximum runtime of 80 minutes (the longest time required by the heuristic).

Table 5 Comparison of SAA, SSS CPLEX and SSS heuristic for PRP.

		SAA	SSS CPLEX				SSS heuristic				CPLEX vs. heuristic	
Problem	K	SAA.Avg	SSS.Sol	CPU	Solution	%SAA	SSS.Sol	CPU	Solution	%SAA	%SSS.Sol	%Solution
PRP–CR	10	533,457	709,667	>80	522,568	2.0%	491,396	21	519,799	2.6%	31%	0.5%
	15	521,562	438,992	>80	519,734	0.4%	354,770	31	515,534	1.2%	19%	0.8%
	20	512,365	241,772	>80	507,565	0.9%	228,149	39	510,392	0.4%	6%	−0.6%
	25	509,998	209,625	>80	507,143	0.6%	141,187	58	502,904	1.4%	33%	0.8%
	30	507,402	126,789	>80	507,394	0.0%	109,817	51	502,903	0.9%	13%	0.9%
PRP–IR	10	793,696	672,022	>80	789,273	0.6%	576,000	41	776,723	2.1%	14%	1.6%
	15	780,548	517,660	>80	765,555	1.9%	373,847	33	762,369	2.3%	28%	0.4%
	20	770,262	280,836	>80	774,385	−0.5%	218,181	69	763,182	0.9%	22%	1.4%
	25	766,417	223,755	>80	765,010	0.2%	161,799	75	754,271	1.6%	28%	1.4%
	30	763,670	145,041	>80	761,789	0.2%	122,972	51	755,081	1.1%	15%	0.9%

">80" means that CPLEX does not converge to the optimum in 80 minutes.
%SAA = (SAA.Avg − Solution) / SAA.Avg, where "Solution" comes from SSS CPLEX or SSS heuristic.
%SSS.Sol = (SSS.Sol of SSS CPLEX − SSS.Sol of SSS heuristic) / SSS.Sol of SSS CPLEX.
%Solution = (Solution of SSS CPLEX − Solution of SSS heuristic) / Solution of SSS CPLEX.

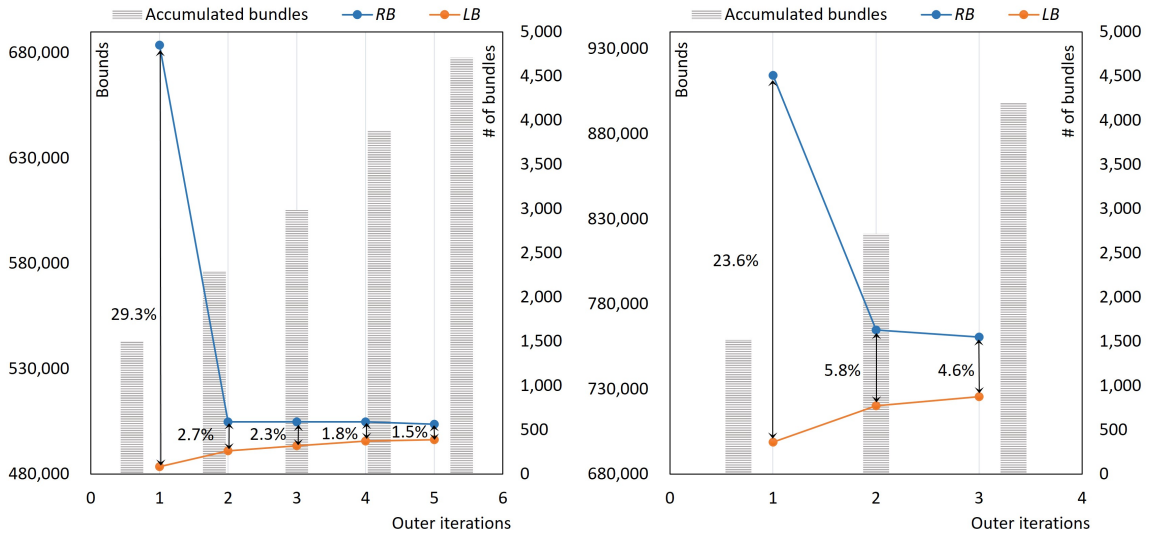
Note, first and foremost, the robust benefits of our SSS approach toward deriving high-quality PRP solutions, especially with small scenario budgets. Indeed, the SSS heuristic leads to lower out-of-sample costs in all 10 problem instances, and SSS CPLEX improves over the SAA benchmark in all but one cases—with improvements of up to 2–3%. In addition, the SSS heuristic outperforms SSS CPLEX in terms of computational times (less than 75 minutes) and SSS objective (6%–33% lower). These improvements in the SSS solutions ultimately result in stronger PRP solutions: in 9 out of the 10 instances, SSS heuristic leads to a lower expected PRP cost, by up to 1.6%. These results can be viewed as an empirical validation of our SSS formulation and our SSS heuristic.

We report in the appendix additional results to guide the design of $\mathcal{X}^A$ and $\mathcal{X}^P$ (Appendix A.3), to assess the benefits of our two local search operators (LSO I and LSO II) in our SSS heuristic, and to guide the setting of the parameter Ω (Appendix A.4).

6.2. Results from scenario aggregation: evaluation of lower bound

We now investigate the performance of our lower-bounding scenario assortment optimization and our column-evaluation-and-generation algorithm. Figure 4 reports the relaxation bound RB and

the lower bound LB at each outer iteration, as well as the number of bundles generated. Note that the column-evaluation-and-generation algorithm terminates in a few outer loop iterations (five for PRP–CR and three for PRP–IR). At each outer iteration, the column generation procedure terminates in a finite number of inner iterations, generating a total of 4,000–5,000 bundles. Ultimately, the column-evaluation-and-generation algorithm returns valid lower bounds, with small optimality gaps (measured as the difference between RB and LB) within the two-hour limit.



(a) PRP–CR (10 scenarios) (b) PRP–IR (10 scenarios)

Figure 4 Convergence of the column-evaluation-and-generation algorithm for SAO, with a two-hour time limit.

Next, Table 6 evaluates the quality of the lower bounds obtained with our SAO model and our column-evaluation-and-generation (CEG) algorithm, against the scenario grouping model and our SAO model solved with row generation. The table reports the lower bound (LB), the relaxation bound (RB), and the convergence gap of each algorithm.

Note, first, that SAO does not necessarily outperform SG when solved with row generation. Recall that the SAO model relaxes the SG model, leading to a theoretically better lower bound but also to a more computationally challenging optimization. The net effects can be negative: for PRP, in two cases out of four, SAO yields a looser bound than SG with the row generation algorithm. However, our SAO model provides significant benefits when combined with our column-evaluation-and-generation algorithm. Notably, for PRP–CR and FLP–CR, the row generation benchmarks need 30 scenarios to converge to optimality, but our column-evaluation-and-generation algorithm reaches optimality with only 20 scenarios. Altogether, these results show the strength of our column-evaluation-and-generation algorithm toward computing strong lower bounds, which tighten those from the row generation benchmarks, by up to 2% (e.g., PRP–CR and FLP–CR with 10 scenarios).

Table 6 Lower bounds obtained with the scenario assortment optimization (SAO) and scenario grouping (SG) approaches, with the row generation (RG) and column-evaluation-and-generation (CEG) algorithms.

	$K=10$			$K=20$			$K=30$			$K=10$		
	SG (RG)	SAO (RG)	SAO (CEG)	SG (RG)	SAO (RG)	SAO (CEG)	SG (RG)	SAO (RG)	SAO (CEG)	SG (RG)	SAO (RG)	SAO (CEG)
	PRP–CR									PRP–IR		
LB	488,737	486,441	496,391	497,547	498,193	501,993	498,648	500,794	502,053	712,571	711,935	725,610
RB	502,558	512,688	503,716	501,316	502,566	502,122	498,648	500,794	502,053	759,346	767,523	760,679
$\frac{RB-LB}{RB}$	2.8%	5.1%	1.5%	0.8%	0.9%	0.0%	0.0%	0.0%	0.0%	6.2%	7.2%	4.6%
	FLP–CR									FLP–IR		
LB	7,178	7,594	7,750	7,670	7,719	7,762	7,728	7,762	7,762	6,261	6,768	6,863
RB	7,862	7,777	7,764	7,712	7,763	7,762	7,728	7,762	7,762	7,015	6,910	6,887
$\frac{RB-LB}{RB}$	9.5%	2.4%	0.2%	0.6%	0.6%	0.0%	0.0%	0.0%	0.0%	12.0%	2.1%	0.4%

In Algorithm 4, we set $|\mathcal{B}|=55$ for $K=10$, $|\mathcal{B}|=25$ for $K=20$, and $|\mathcal{B}|=17$ for $K=30$. In Algorithm 1, $|\mathcal{B}|$ is endogenous.
The time limit is two hours for PRP–CR and PRP–IR; one hour for FLP–CR and FLP–IR.

Table 7 Lower bounds obtained with the scenario assortment optimization approach, against all benchmarks.

	PRP–CR			PRP–IR	FLP–CR			FLP–IR
	$K=10$	$K=20$	$K=30$	$K=10$	$K=10$	$K=20$	$K=30$	$K=10$
U	502,080	502,080	502,080	752,394	7,762	7,762	7,762	6,887
$\frac{U-LB}{U}$	1.1%	0.0%	0.0%	3.6%	0.2%	0.0%	0.0%	0.3%
$U(K)$	519,799	510,392	502,903	776,723	7,787	7,769	7,777	6,931
$\frac{U(K)-LB}{U(K)}$	4.5%	1.6%	0.2%	6.6%	0.5%	0.1%	0.2%	1.0%
$\frac{LB-L^C}{LB}$	9.4%	10.4%	10.4%	5.9%	100.0%	100.0%	100.0%	100.0%
$\frac{LB-L_1}{LB}$	49.4%	50.0%	50.0%	99.2%	94.6%	94.6%	94.6%	96.0%
$\frac{LB-L_2}{LB}$	1.6%	2.7%	2.7%	28.9%	7.0%	7.1%	7.1%	5.1%

U denotes the best (known) overall solution; $U(K)$ denotes the best solution obtained with K scenarios.
L^C denotes the lower bound obtained with CPLEX (449,730 for PRP–CR, 683,138 for PRP–IR, 0 for FLP–CR, 0 for FLP–IR).
L_1 denotes the lower bound obtained with Benders decomposition (251,102 for PRP–CR, 419 for FLP–CR) or the integer L-shaped method (6,000 for PRP–IR, 274 for FLP–IR).
L_2 denotes the lower bound obtained with enhanced Benders decomposition (488,660 for PRP–CR, 7,207 for FLP–CR) or the enhanced integer L-shaped method (515,559 for PRP–IR, 6,515 for FLP–IR).
The time limit is two hours for PRP–CR and PRP–IR; one hour for FLP–CR and FLP–IR.

To evaluate the quality of the lower bound, Table 7 reports the gap from the lower bound LB to the best known solution (U) and to the one obtained with the same number of scenarios ($U(K)$). Comparisons between LB and U provide our best estimates of the optimality gap, whereas comparisons between LB and $U(K)$ reflect the optimality gaps that would be obtained with a fixed scenario budget. Our lower bounds guarantee optimality for PRP–CR and FLP–CR, and guarantee that the best known PRP–IR and FLP–IR solutions lie within 3.6% and 0.3% of the optimum, respectively. When comparing with $U(K)$, we obviously obtain larger optimality gaps, but these remain reasonable—at most 6.6% for PRP and at most 1.0% for FLP. Ultimately, our SAO model, combined with our column-evaluation-and-generation algorithm, provides novel lower bounds in stochastic programming that can prove the optimality, or near-optimality of the solutions derived from our optimization-based scenario reduction approach without ever solving the full problem.

Finally, Table 7 compares the lower bound LB to the ones obtained with the stochastic programming benchmarks: CPLEX (L^C), Benders decomposition or the integer L-shaped method (L_1), and E-Benders or E-L-shaped (L_2). For PRP–CR, Benders decomposition returns very loose bounds. CPLEX provides a tighter lower bound, mainly due to the two-commodity flow formulation (**?**). By leveraging problem-specific lower bound lifting inequalities, E-Benders generates a much stronger bound. Yet, our SAO approach improves the bounds from all three methods, by 50% as compared to Benders, by 9–10% as compared to CPLEX, and by up to 3% as compared to E-Benders. For PRP–IR, the strongest bound is obtained with CPLEX, but our SAO approach tightens it by up to 6%. For the facility location problems, CPLEX, Benders decomposition and the integer L-shaped method yield very loose bounds and, again, our SAO approach tightens the strongest bounds (vs. E-Benders and E-L-shaped) by 5–7%.

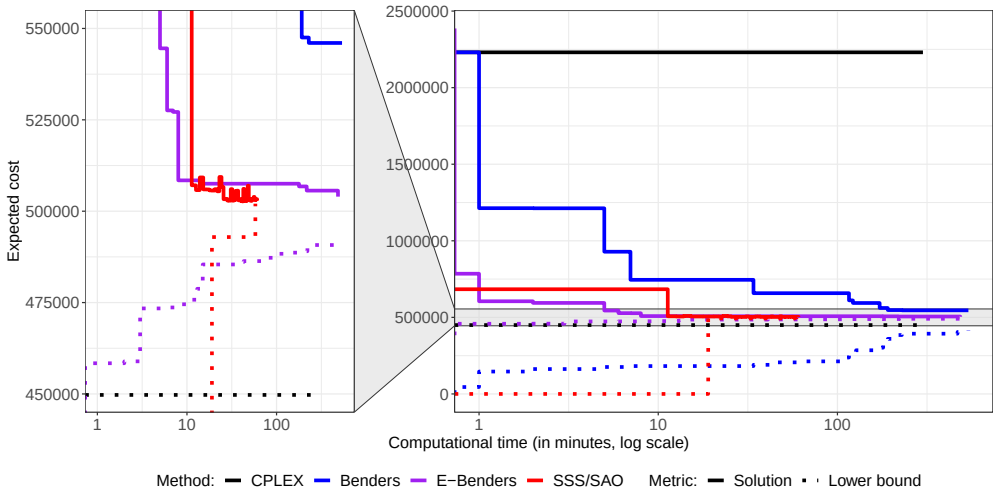
Ultimately, these results demonstrate that our SAO approach generates strong lower bounds that tighten the solution guarantees resulting from stochastic programming algorithms. In other words, our methods not only improve scenario reduction, but also provide new optimality guarantees.

6.3. Comparison with stochastic programming algorithms

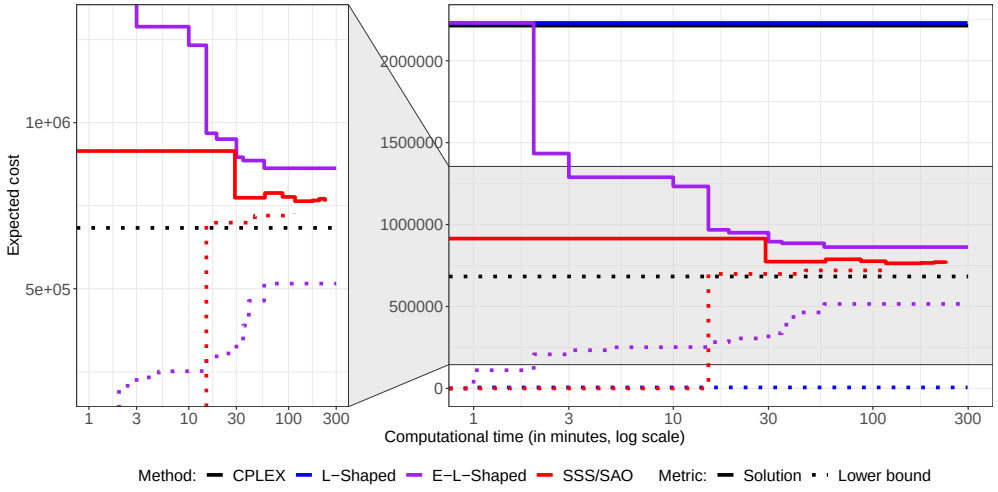
Figures 5 and 6 summarize our results for the production routing and facility location problems, respectively. The figures show the incumbent solution (solid lines) and lower bound (dashed lines) as a function of CPU time, for our method (SSS for the solution and SAO for the lower bound) and the three stochastic programming benchmarks (i.e., CPLEX, Benders and E-Benders for PRP–CR and FLP–CR, and CPLEX, L-shaped and E-L-shaped for PRP–IR and FLP–IR).

Note, first, that standard stochastic programming algorithms (CPLEX, Benders decomposition and the integer L-shaped method) all converge slowly, returning poor solutions, loose bounds, or both, echoing the results from Table 2. Next, the E-Benders and E-L-shaped methods provide significant improvements by leveraging stronger cutting planes and other acceleration strategies, both in terms of solution and lower bound. These results can be viewed as a validation of the algorithms from **?** for PRP–CR and from **?** for FLP–IR.

In comparison, our scenario reduction methods consistently outperform all benchmarks. For all problems, our SSS/SAO method yields the strongest solution as well as the tightest lower bound, even when the benchmarks are allowed to run much longer. To summarize these results, Table 8 reports the incumbent solution, lower bound and optimality gap from each method at the time of SSS/SAO termination. For PRP–CR, our SSS/SAO method yields a near-optimal solution (0.2% gap) within one hour of computation, when E-Benders returns an inferior solution (by 0.9%) and a looser lower bound (by 3.1%). For PRP–IR, our SSS/SAO approach leaves a 5% gap in two hours, but improves the solution (by over 10%) and tightens the lower bound (by over 40%) as compared



(a) PRP–CR (30 scenarios)

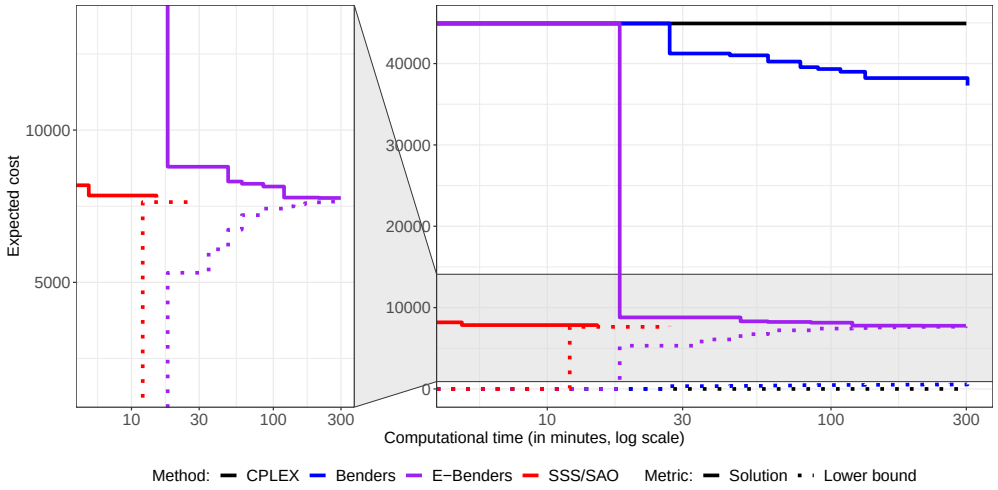


(b) PRP–IR (10 scenarios)

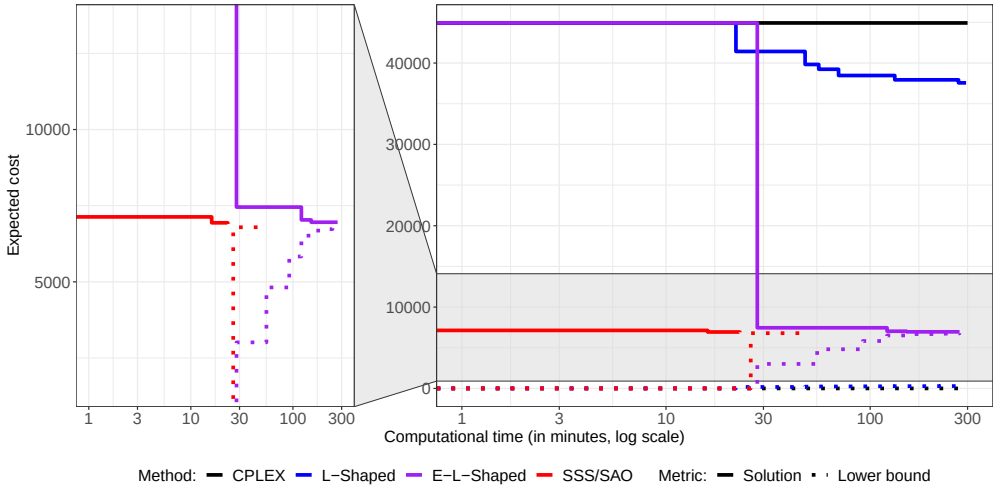
Figure 5 Comparison of the solutions and lower bounds obtained via scenario reduction and scenario assortment (SSS/SAO) to benchmark stochastic programming algorithms, for the production routing problems.

to the best benchmark. For facility location problems, our SSS/SAO method reaches a provably optimal solution (with a zero optimality gap) in 30–60 minutes. At that time, the E-Benders or E-L-shaped methods yield inferior solutions (by 8–12%) and looser bounds (by 20–30%); when allowed to run for five hours, the benchmarks leave 1% and 3% optimality gaps, respectively.

In conclusion, our scenario reduction and scenario assortment methods consistently provide a Pareto improvement, as compared to state-of-the-art stochastic programming algorithms: better solutions, tighter lower bounds, and faster computational times. Moreover, the proposed approach does not rely on any particular structure, and is thus generally applicable to a broad class of stochastic programming problems—including problems with mixed-integer recourse. Obviously, this does not mean that the proposed scenario reduction method should replace stochastic pro-



(a) FLP–CR (10 scenarios)



(b) FLP–IR (10 scenarios)

Figure 6 **Comparison of the solutions and lower bounds obtained via scenario reduction and scenario assortment (SSS/SAO) to benchmark stochastic programming algorithms, for the facility location problems.**

gramming algorithms. Instead, this paper provides evidence of an extra degree of freedom to tackle very large-scale stochastic programming problems via optimization-based scenario reduction.

7. Conclusion

A core challenge in stochastic programming involves characterizing uncertainty while avoiding the curse of dimensionality. This paper tackles this question by developing a new optimization approach to scenario reduction that generates a high-quality solution and a tight lower bound by only solving small-scale instances within a scenario budget—dictated by the problem's complexity, solution algorithms and practical requirements. The proposed approach comprises two elements:

1. A scenario subset selection (SSS) model that minimizes the recourse estimation over a pool of solutions. The SSS is cast as a sparse regression problem, and solved using a tailored heuristic.

Table 8 Summary of main results at the time of SSS/SAO convergence.

Problem	Metric	CPLEX	Benders/L-shaped	E-Benders/E-L-shaped	SSS/SAO
PRP–CR	Solution	2,230,621 (+343.5%)	657,892 (+30.8%)	507,525 (+0.9%)	502,903 (base)
	Lower bound	449,730 (−10.4%)	212,623 (−57.6%)	486,352 (−3.1%)	502,053 (base)
	Gap	79.8%	67.7%	4.2%	0.2%
	CPU (mins.)	60	60	60	60
PRP–IR	Solution	2,214,462 (+190.0%)	2,230,621 (+192.2%)	862,681 (+13.0%)	763,486 (base)
	Lower bound	683,183 (−5.8%)	6,000 (−99.2%)	515,559 (−28.9%)	725,610 (base)
	Gap	69.1%	99.7%	40.2%	5.0%
	CPU (mins.)	120	120	120	120
FLP–CR	Solution	44,931 (+477.0%)	41,236 (+429.5%)	8,796 (+13.0%)	7,787 (base)
	Lower bound	0 (−100.0%)	345 (−95.5%)	6,085 (−21.5%)	7,750 (base)
	Gap	100.0%	99.2%	30.8%	0.5%
	CPU (mins.)	30	30	30	30
FLP–IR	Solution	44,931 (+551.3%)	39,239 (+468.8%)	7,454 (+8.0%)	6,899 (base)
	Lower bound	0 (−100.0%)	241 (−96.5%)	4,816 (−29.8%)	6,863 (base)
	Gap	100.0%	99.4%	35.4%	0.5%
	CPU (mins.)	60	60	60	60

2. A scenario assortment optimization approach that generates a lower bound of the full stochastic program by relaxing nonanticipativity constraints. To maximize the lower bound, we develop a new column-evaluation-and-generation algorithm that iterates between a column generation step (to select scenario bundles) and a column evaluation step (to update the costs of the selected bundles). This approach provides a generalizable algorithm for optimization formulations featuring many decision variables and hard-to-evaluate objective parameters.

Computational results highlight the effectiveness of our methodological approach toward generating high-quality solutions and tight lower bounds for otherwise-challenging stochastic programs. First, our SSS model outperforms benchmarks based on sample average approximation and distribution-based scenario reduction. Moreover, our scenario assortment optimization approach, combined with our column-evaluation-and-generation algorithm, yields a tight lower bound in a small number of iterations. Finally, our results are competitive with state-of-the-art stochastic programming algorithms. Our methods even achieve Pareto improvements: stronger solutions, tighter lower bounds, and faster computational times. These results are robust across problem settings (production routing and facility location) and problem structure (continuous recourse and mixed-integer recourse). Ultimately, these results demonstrate the value of our optimization-based scenario

reduction and scenario assortment optimization approaches, as a complement of advanced solution algorithms, for solving large-scale stochastic programs and providing tight quality guarantees.

References

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

Ahmed S (2013) A scenario decomposition algorithm for 0–1 stochastic programs. *Operations Research Letters* 41(6):565–569.

Ahmed S, Luedtke J, Song Y, Xie W (2017) Nonanticipative duality, relaxations, and formulations for chance-constrained stochastic programs. *Mathematical Programming* 162(1-2):51–81.

Arpón S, Homem-de Mello T, Pagnoncelli B (2018) Scenario reduction for stochastic programs with conditional value-at-risk. *Mathematical Programming* 170(1):327–356.

Baldacci R, Hadjiconstantinou E, Mingozzi A (2004) An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research* 52(5):723–738.

Bertsimas D, Kallus N (2020) From predictive to prescriptive analytics. *Management Science* 66(3):1025–1044.

Bertsimas D, King A, Mazumder R (2016) Best subset selection via a modern optimization lens. *The Annals of Statistics* 44(2):813–852.

Bertsimas D, Mundru N (2022) Optimization-based scenario reduction for data-driven two-stage stochastic optimization. *Operations Research* in press.

Birge J, Louveaux F (2011) *Introduction to Stochastic Programming* (Springer Science & Business Media).

Carøe CC, Schultz R (1999) Dual decomposition in stochastic integer programming. *Operations Research Letters* 24(1-2):37–45.

Casey MS, Sen S (2005) The scenario generation algorithm for multistage stochastic linear programming. *Mathematics of Operations Research* 30(3):615–631.

Crainic T, Hewitt M, Rei W (2014) Scenario grouping in a progressive hedging-based meta-heuristic for stochastic network design. *Computers & Operations Research* 43:90–99.

Dantzig GB, Madansky A (1961) On the solution of two-stage linear programs under uncertainty. *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, 165–176 (University of California Press Berkeley).

Dempster MAH (2006) Sequential importance sampling algorithms for dynamic stochastic programming. *Journal of Mathematical Sciences* 133(4):1422–1444.

Desaulniers G, Desrosiers J, Solomon MM (2006) *Column generation*, volume 5 (Springer Science & Business Media).

Drew SS, Homem-de-Mello T (2006) Quasi-Monte Carlo strategies for stochastic optimization. *Proceedings of the 2006 Winter Simulation Conference*, 774–782 (IEEE).

Dupačová J, Gröwe-Kuska N, Römisch W (2003) Scenario reduction in stochastic programming. *Mathematical Programming* 95(3):493–511.

Fairbrother J, Turner A, Wallace SW (2018) Scenario generation for single-period portfolio selection problems with tail risk measures: coping with high dimensions and integer variables. *INFORMS Journal on Computing* 30(3):472–491.

Fairbrother J, Turner A, Wallace SW (2019) Problem-driven scenario generation: an analytical approach for stochastic programs with tail risk measure. *Mathematical Programming* 1–42, URL http://dx.doi.org/10.1007/s10107-019-01451-7.

Feng Y, Ryan SM (2016) Solution sensitivity-based scenario reduction for stochastic unit commitment. *Computational Management Science* 13(1):29–62.

Gade D, Hackebeil G, Ryan S, Watson JP, Wets R, Woodruff D (2016) Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs. *Mathematical Programming* 157(1):47–67.

Guo C, Bodur M, Aleman DM, Urbach DR (2021) Logic-based benders decomposition and binary decision diagram based approaches for stochastic distributed operating room scheduling. *INFORMS Journal on Computing* 33(4):1551–1569.

Heitsch H, Römisch W (2007) A note on scenario reduction for two-stage stochastic programs. *Operations Research Letters* 35(6):731–738.

Henrion R, Römisch W (2018) Problem-based optimal scenario generation and reduction in stochastic programming. *Mathematical Programming* 1–23, URL http://dx.doi.org/10.1007/s10107-018-1337-6.

Higle JL (1998) Variance reduction and objective function evaluation in stochastic linear programs. *INFORMS Journal on Computing* 10(2):236–247.

Hochreiter R, Pflug GC (2007) Financial scenario generation for stochastic multi-stage decision processes as facility location problems. *Annals of Operations Research* 152(1):257–272.

Homem-de-Mello T (2008) On rates of convergence for stochastic optimization problems under non–independent and identically distributed sampling. *SIAM Journal on Optimization* 19(2):524–551.

Høyland K, Kaut M, Wallace SW (2003) A heuristic for moment-matching scenario generation. *Computational Optimization and Applications* 24(2-3):169–185.

Høyland K, Wallace SW (2001) Generating scenario trees for multistage decision problems. *Management Science* 47(2):295–307.

Khassiba A, Bastin F, Cafieri S, Gendron B, Mongeau M (2020) Two-stage stochastic mixed-integer programming with chance constraints for extended aircraft arrival management. *Transportation Science* 54(4):897–919.

King A, Wallace SW (2012) *Modeling with Stochastic Programming* (Springer Science & Business Media).

Kleywegt AJ, Shapiro A, Homem-de-Mello T (2002) The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization* 12(2):479–502.

Koivu M (2005) Variance reduction in sample approximations of stochastic programs. *Mathematical Programming* 103(3):463–485.

Laporte G, Louveaux FV (1993) The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters* 13(3):133–142.

Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. *Operations Research* 53(6):1007–1023.

Luedtke J (2016) Tutorial: Stochastic Integer Programming. *XIV International Conference on Stochastic Programming*.

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

Mak WK, Morton DP, Wood RK (1999) Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters* 24(1-2):47–56.

Narum BS (2020) *Problem-based scenario generation in stochastic programming with binary distributions-Case study in Air Traffic Flow Management*. Master's thesis, Norwegian University of Science and Technology.

Nemirovski A, Shapiro A (2006) Scenario approximations of chance constraints. *Probabilistic and Randomized Methods for Design under Uncertainty*, 3–47 (Springer).

Park J, Stockbridge R, Bayraksan G (2021) Variance reduction for sequential sampling in stochastic programming. *Annals of Operations Research* 300(1):171–204.

Parpas P, Ustun B, Webster M, Tran QK (2015) Importance sampling in stochastic programming: A Markov chain Monte Carlo approach. *INFORMS Journal on Computing* 27(2):358–377.

Penuel J, Smith JC, Yuan Y (2010) An integer decomposition algorithm for solving a two-stage facility location problem with second-stage activation costs. *Naval Research Logistics* 57(5):391–402.

Pflug GC (2001) Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical Programming* 89(2):251–271.

Prochazka V, Wallace SW (2018) Stochastic programs with binary distributions: structural properties of scenario trees and algorithms. *Computational Management Science* 15(3-4):397–410.

Prochazka V, Wallace SW (2020) Scenario tree construction driven by heuristic solutions of the optimization problem. *Computational Management Science* 17:277–307.

Ralph D, Xu H (2011) Convergence of stationary points of sample average two-stage stochastic programs: A generalized equation approach. *Mathematics of Operations Research* 36(3):568–592.

Rockafellar T, Wets R (1991) Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research* 16(1):119–147.

Römisch W (2009) Scenario reduction techniques in stochastic programming. *International Symposium on Stochastic Algorithms*, 1–14 (Springer).

Ryan K, Ahmed S, Dey SS, Rajan D, Musselman A, Watson JP (2020) Optimization-driven scenario grouping. *INFORMS Journal on Computing* 32(3):805–821.

Schütz P, Tomasgard A, Ahmed S (2009) Supply chain design under uncertainty using sample average approximation and dual decomposition. *European Journal of Operational Research* 199(2):409–419.

Seljom P, Tomasgard A (2021) Sample average approximation and stability tests applied to energy system design. *Energy Systems* 12(1):107–131.

Shapiro A, Dentcheva D, Ruszczyński A (2014) *Lectures on Stochastic Programming: Modeling and Theory* (SIAM).

Shapiro A, Homem-de-Mello T (2000) On the rate of convergence of optimal solutions of monte carlo approximations of stochastic programs. *SIAM Journal on Optimization* 11(1):70–86.

Shapiro A, Xu H (2008) Stochastic mathematical programs with equilibrium constraints, modelling and sample average approximation. *Optimization* 57(3):395–418.

Sherali HD, Lunday BJ (2013) On generating maximal nondominated Benders cuts. *Annals of Operations Research* 210(1):57–72.

Smith JC, Penuel J (2008) Solving a two-stage facility location problem with second-stage activation costs. *IIE Annual Conference. Proceedings*, 1939 (Institute of Industrial and Systems Engineers (IISE)).

Song Y, Luedtke J (2015) An adaptive partition-based approach for solving two-stage stochastic programs with fixed recourse. *SIAM Journal on Optimization* 25(3):1344–1367.

Sun M, Teng F, Konstantelos I, Strbac G (2018) An objective-based scenario selection method for transmission network expansion planning with multivariate stochasticity in load and renewable energy sources. *Energy* 145:871–885.

Van Parys BP, Esfahani PM, Kuhn D (2021) From data to decisions: Distributionally robust optimization is optimal. *Management Science* 67(6):3387–3402.

Wallace SW (2000) Decision making under uncertainty: Is sensitivity analysis of any use? *Operations Research* 48(1):20–25.

Wang K, Jacquillat A (2020) A stochastic integer programming approach to air traffic scheduling and operations. *Operations Research* 68(5):1375–1402.

Yoon S, Albert LA, White VM (2021) A stochastic programming approach for locating and dispatching two types of ambulances. *Transportation Science* 55(2):275–296.

Zhou L, Geng N, Jiang Z, Jiang S (2022) Integrated multiresource capacity planning and multitype patient scheduling. *INFORMS Journal on Computing* 34(1):129–149.

Appendix A: Details on the optimization-based scenario reduction approach

A.1. Benefits of the heuristic algorithm for scenario pooling

Recall that our solution pooling model (Equation (8)) reduces to a facility location problem, and is thus NP-hard. We propose a tailored two-step heuristic to solve it. In the first step, we optimize the "location" decisions $\boldsymbol{\zeta}$ (which determine $\mathcal{X}^P$). In the second step, we optimize the "mapping" decisions $\boldsymbol{\psi}$.

Step 1. The first step relies on a greedy procedure to maximize the diversity of the solutions $\boldsymbol{x} \in \mathcal{X}^P$. We define $\bar{d}_{\boldsymbol{x},\boldsymbol{x}'} = \sum_{s\in\mathcal{S}} d_{\boldsymbol{x},\boldsymbol{x}',s}$, as the total distance between solutions $\boldsymbol{x}$ and $\boldsymbol{x}'$. We initialize $\mathcal{X}^P$ with the solution that is farthest from all others in $\mathcal{X}^A$, i.e.: $\mathcal{X}^P \leftarrow \{\boldsymbol{x}''\},\ \boldsymbol{x}'' \in \arg\max_{\boldsymbol{x}\in\mathcal{X}^A}\{\sum_{\boldsymbol{x}'\in\mathcal{X}^A} d_{\boldsymbol{x},\boldsymbol{x}'}\}$. We then update $\mathcal{X}^P$ iteratively to include solutions that are farthest from the current ones, i.e., $\mathcal{X}^P \leftarrow \mathcal{X}^P \cup \{\boldsymbol{x}''\},\ \boldsymbol{x}'' \in \arg\max_{\boldsymbol{x}\in\mathcal{X}^A\setminus\mathcal{X}^P}\{\sum_{\boldsymbol{x}'\in\mathcal{X}^P} d_{\boldsymbol{x},\boldsymbol{x}'}\}$. We repeat the process until $|\mathcal{X}^P| = L$.

Step 2. Given $\mathcal{X}^P$ in Step 1, we fix variables $\zeta_{\boldsymbol{x}'} = 1$ for all $\boldsymbol{x}' \in \mathcal{X}^P$, and $\zeta_{\boldsymbol{x}'} = 0$ for all $\boldsymbol{x}' \notin \mathcal{X}^P$. Then, we solve the corresponding solution pooling model (Equation (8)), which determines $\boldsymbol{\psi}$.

We evaluate this two-step heuristic for the production routing problem (see Section 5). Table 9 shows that it generates much stronger solutions much faster, as compared to direct CPLEX implementation of Equation (8). In all our test instances, CPLEX does not converge within a maximum runtime of one hour. In contrast, our heuristic algorithm consistently terminates in less than one second. Moreover, the heuristic algorithm generates much stronger solutions, reducing the objective value by 30%–60%.

Table 9 Performance of the heuristic algorithm for solution pooling, with $|\mathcal{X}^S| = 500$.

	PRP–CR					PRP–IR				
$\lvert\mathcal{X}^P\rvert$	30	60	90	120	150	30	60	90	120	150
Δ^P (CPLEX)	973,333	954,613	967,601	1,126,479	1,126,479	1,179,871	1,191,264	945,642	1,191,264	1,191,264
Δ^P (Heuristic)	599,746	563,584	544,199	490,342	482,633	689,450	636,225	618,504	547,142	534,933
Gap	38%	41%	44%	56%	57%	42%	47%	35%	54%	55%

CPLEX terminates in one hour and the heuristic algorithm in less than one second for all the tests.
Gap = (Δ^P (CPLEX) − Δ^P (Heuristic))/ Δ^P (CPLEX).

A.2. Proof of statements

Proof of Lemma 1 We start with the following definitions:

$$R_s^U = \max_{\boldsymbol{x}\in\mathcal{X}} Q(\boldsymbol{x},\xi_s), \quad \forall s \in \mathcal{S} \tag{32}$$

$$R_s^L = \min_{\boldsymbol{x}\in\mathcal{X}} Q(\boldsymbol{x},\xi_s), \quad \forall s \in \mathcal{S} \tag{33}$$

$$R^C = \max\left\{0, -\min_{s\in\mathcal{S}} R_s^L\right\} \tag{34}$$

$$R^M = \max_{s\in\mathcal{S}} R_s^U. \tag{35}$$

Note that the range $[R_s^L, R_s^U]$ for the recourse function for each scenario s is well defined because of our assumptions that $\mathcal{X}$ is compact (if continuous) or finite (if discrete) and that the problem has relatively complete recourse. The constants R^C and R^M are also well defined given the finiteness of the scenario set $\mathcal{S}$.

Let us define problem $\mathcal{P}'(\mathcal{S},\boldsymbol{h})$, where the recourse function is replaced by $Q^+(\boldsymbol{x},\xi_s)=Q(\boldsymbol{x},\xi_s)+R^C$:

$$\mathcal{P}'(\mathcal{S},\boldsymbol{h})=\min_{\boldsymbol{x}\in\mathcal{X}}\left\{\boldsymbol{c}^T\boldsymbol{x}+\sum_{s\in\mathcal{S}}h_sQ^+(\boldsymbol{x},\xi_s)\right\}=\min_{\boldsymbol{x}\in\mathcal{X}}\left\{\boldsymbol{c}^T\boldsymbol{x}+\sum_{s\in\mathcal{S}}h_sQ(\boldsymbol{x},\xi_s)+R^C\right\}.$$

Clearly, an optimal solution of $\mathcal{P}(\mathcal{S},\boldsymbol{h})$ is optimal for $\mathcal{P}'(\mathcal{S},\boldsymbol{h})$, and vice versa. The main difference is that the recourse function in $\mathcal{P}'(\mathcal{S},\boldsymbol{h})$ is positive. We leverage this fact in the following claim.

Claim 1. *We have $w_s\leq U_s$ for all $s\in\mathcal{S}$, where:*

$$U_s=\max_{\boldsymbol{x}\in\mathcal{X}^A:Q^+(\boldsymbol{x},\xi_s)\neq 0}\frac{\sum_{\sigma\in\mathcal{S}}h_\sigma Q^+(\boldsymbol{x},\xi_\sigma)}{Q^+(\boldsymbol{x},\xi_s)}=\max_{\boldsymbol{x}\in\mathcal{X}^A:Q(\boldsymbol{x},\xi_s)+R^C\neq 0}\frac{\sum_{\sigma\in\mathcal{S}}h_\sigma Q(\boldsymbol{x},\xi_\sigma)+R^C}{Q(\boldsymbol{x},\xi_s)+R^C}. \tag{36}$$

Assume, by contradiction, that there exists $s'\in\mathcal{S}$ such that $w_{s'}>U_{s'}$. Then, we define a solution $\boldsymbol{w}'$ as:

$$w'_s=\begin{cases}w_s & \text{if } s\neq s',\\ U_{s'} & \text{if } s=s'.\end{cases}$$

Clearly, we have $\sum_{s\in\mathcal{S}}\mathbf{1}(w'_s\neq 0)\leq\sum_{s\in\mathcal{S}}\mathbf{1}(w_s\neq 0)\leq K$, so $\boldsymbol{w}'$ is a feasible solution to the SSS model. Moreover, we show that it yields a smaller value of the objective function.

Let us denote by $\mathcal{X}_0^P=\{\boldsymbol{x}\in\mathcal{X}^P:Q^+(\boldsymbol{x},\xi_{s'})=0\}$. By definition of U_s, we have, for all $\boldsymbol{x}\in\mathcal{X}^P\setminus\mathcal{X}_0^P$:

$$U_{s'}Q^+(\boldsymbol{x},\xi_{s'})\geq\sum_{s\in\mathcal{S}}h_sQ^+(\boldsymbol{x},\xi_s)\geq\sum_{s\in\mathcal{S}}h_sQ^+(\boldsymbol{x},\xi_s)-\sum_{s\neq s'}w_sQ^+(\boldsymbol{x},\xi_s). \tag{37}$$

It comes:

$$\begin{aligned}
&\sum_{\boldsymbol{x}\in\mathcal{X}^P}\left|\sum_{s\in\mathcal{S}}h_sQ^+(\boldsymbol{x},\xi_s)-\sum_{s\in\mathcal{S}}w_sQ^+(\boldsymbol{x},\xi_s)\right|\\
&=\sum_{\boldsymbol{x}\in\mathcal{X}^P\setminus\mathcal{X}_0^P}\left|\sum_{s\in\mathcal{S}}h_sQ^+(\boldsymbol{x},\xi_s)-\sum_{s\neq s'}w_sQ^+(\boldsymbol{x},\xi_s)-w_{s'}Q^+(\boldsymbol{x},\xi_{s'})\right|\\
&\quad+\sum_{\boldsymbol{x}\in\mathcal{X}_0^P}\left|\sum_{s\in\mathcal{S}}h_sQ^+(\boldsymbol{x},\xi_s)-\sum_{s\neq s'}w_sQ^+(\boldsymbol{x},\xi_s)-w_{s'}Q^+(\boldsymbol{x},\xi_{s'})\right|\\
&=-\sum_{\boldsymbol{x}\in\mathcal{X}^P\setminus\mathcal{X}_0^P}\left[\sum_{s\in\mathcal{S}}h_sQ^+(\boldsymbol{x},\xi_s)-\sum_{s\neq s'}w_sQ^+(\boldsymbol{x},\xi_s)-w_{s'}Q^+(\boldsymbol{x},\xi_{s'})\right]\\
&\quad+\sum_{\boldsymbol{x}\in\mathcal{X}_0^P}\left|\sum_{s\in\mathcal{S}}h_sQ^+(\boldsymbol{x},\xi_s)-\sum_{s\neq s'}w_sQ^+(\boldsymbol{x},\xi_s)-w_{s'}Q^+(\boldsymbol{x},\xi_{s'})\right|\\
&>-\sum_{\boldsymbol{x}\in\mathcal{X}^P\setminus\mathcal{X}_0^P}\left[\sum_{s\in\mathcal{S}}h_sQ^+(\boldsymbol{x},\xi_s)-\sum_{s\neq s'}w_sQ^+(\boldsymbol{x},\xi_s)-U_{s'}Q^+(\boldsymbol{x},\xi_{s'})\right]\\
&\quad+\sum_{\boldsymbol{x}\in\mathcal{X}_0^P}\left|\sum_{s\in\mathcal{S}}h_sQ^+(\boldsymbol{x},\xi_s)-\sum_{s\neq s'}w_sQ^+(\boldsymbol{x},\xi_s)-w_{s'}Q^+(\boldsymbol{x},\xi_{s'})\right|\\
&=\sum_{\boldsymbol{x}\in\mathcal{X}^P\setminus\mathcal{X}_0^P}\left|\sum_{s\in\mathcal{S}}h_sQ^+(\boldsymbol{x},\xi_s)-\sum_{s\neq s'}w_sQ^+(\boldsymbol{x},\xi_s)-U_{s'}Q^+(\boldsymbol{x},\xi_{s'})\right|\\
&\quad+\sum_{\boldsymbol{x}\in\mathcal{X}_0^P}\left|\sum_{s\in\mathcal{S}}h_sQ^+(\boldsymbol{x},\xi_s)-\sum_{s\neq s'}w_sQ^+(\boldsymbol{x},\xi_s)-U_{s'}Q^+(\boldsymbol{x},\xi_{s'})\right|\\
&=\sum_{\boldsymbol{x}\in\mathcal{X}^P}\left|\sum_{s\in\mathcal{S}}h_sQ^+(\boldsymbol{x},\xi_s)-\sum_{s\in\mathcal{S}}w'_sQ^+(\boldsymbol{x},\xi_s)\right|.
\end{aligned}$$

The first equality is straightforward. The second equality stems from Equation (37) and the assumption $w_{s'} > U_{s'}$. The third inequality comes from $w_{s'} > U_{s'}$. The last equality comes from the definition of $U_{s'}$ in (36). Ultimately, this result contradicts the optimality of $\boldsymbol{w}$. This completes the proof of Claim 1.

Next, note that U_s is upper bounded by $\overline{U}$ defined as follows:

$$U_s \leq \overline{U} = \frac{R^M + R^C}{m}, \quad \forall s \in \mathcal{S}, \quad \text{where } m = \min_{s\in\mathcal{S}} \min_{\boldsymbol{x}\in\mathcal{X}: Q(\boldsymbol{x},\xi_s)+R^C\neq 0} \left(Q(\boldsymbol{x},\xi_s) + R^C\right).$$

Note that $m > 0$ and, hence $\overline{U}$ is a finite constant that only depends on the stochastic program. We complete the proof of Lemma 1 by combining the facts that $w_s \leq U_s$ (Claim 1) and $U_s \leq \overline{U}$. □

Proof of Theorem 1 Let us denote by Λ the optimization loss, as follows:

$$\Lambda = \left(\boldsymbol{c}^T\widetilde{\boldsymbol{x}} + \sum_{s\in\mathcal{S}} h_s Q(\widetilde{\boldsymbol{x}},\xi_s)\right) - \left(\boldsymbol{c}^T\boldsymbol{x}^* + \sum_{s\in\mathcal{S}} h_s Q(\boldsymbol{x}^*,\xi_s)\right). \tag{38}$$

Since $\widetilde{\boldsymbol{x}}$ is the optimal solution of $\mathcal{P}(\mathcal{S}_0, \boldsymbol{w})$, we have

$$\boldsymbol{c}^T\widetilde{\boldsymbol{x}} + \sum_{s\in\mathcal{S}} w_s Q(\widetilde{\boldsymbol{x}},\xi_s) = \boldsymbol{c}^T\widetilde{\boldsymbol{x}} + \sum_{s\in\mathcal{S}_0} w_s Q(\widetilde{\boldsymbol{x}},\xi_s) \leq \boldsymbol{c}^T\boldsymbol{x}^* + \sum_{s\in\mathcal{S}_0} w_s Q(\boldsymbol{x}^*,\xi_s) = \boldsymbol{c}^T\boldsymbol{x}^* + \sum_{s\in\mathcal{S}} w_s Q(\boldsymbol{x}^*,\xi_s). \tag{39}$$

We obtain, from Equations (38) and (39):

$$\begin{aligned}\Lambda &\leq \left(\sum_{s\in\mathcal{S}} h_s Q(\widetilde{\boldsymbol{x}},\xi_s) - \sum_{s\in\mathcal{S}} w_s Q(\widetilde{\boldsymbol{x}},\xi_s)\right) - \left(\sum_{s\in\mathcal{S}} h_s Q(\boldsymbol{x}^*,\xi_s) - \sum_{s\in\mathcal{S}} w_s Q(\boldsymbol{x}^*,\xi_s)\right) \\ &\leq \left|\sum_{s\in\mathcal{S}} h_s Q(\widetilde{\boldsymbol{x}},\xi_s) - \sum_{s\in\mathcal{S}} w_s Q(\widetilde{\boldsymbol{x}},\xi_s)\right| + \left|\sum_{s\in\mathcal{S}} h_s Q(\boldsymbol{x}^*,\xi_s) - \sum_{s\in\mathcal{S}} w_s Q(\boldsymbol{x}^*,\xi_s)\right|. \end{aligned} \tag{40}$$

We process the two terms similarly. Let us denote by $\boldsymbol{x}^A \in \mathcal{X}^A$ the closest neighbor of $\boldsymbol{x}^*$ in $\mathcal{X}^A$, i.e., $\arg\min_{\boldsymbol{x}^A\in\mathcal{X}^A} \|\boldsymbol{x}^* - \boldsymbol{x}^A\|$. Let $\boldsymbol{\zeta}^*$ be the optimal solution of the solution pooling model (Equation (8)). There exists a unique $\boldsymbol{x}^P \in \mathcal{X}^P$ such that $\psi^*_{\boldsymbol{x}^A,\boldsymbol{x}^P} = 1$. It comes, from the triangular inequality:

$$\begin{aligned}\left|\sum_{s\in\mathcal{S}} h_s Q(\boldsymbol{x}^*,\xi_s) - \sum_{s\in\mathcal{S}} w_s Q(\boldsymbol{x}^*,\xi_s)\right| \leq{}& \sum_{s\in\mathcal{S}} h_s \left|Q(\boldsymbol{x}^*,\xi_s) - Q(\boldsymbol{x}^A,\xi_s)\right| \\ &+ \sum_{s\in\mathcal{S}} h_s \left|Q(\boldsymbol{x}^A,\xi_s) - Q(\boldsymbol{x}^P,\xi_s)\right| \\ &+ \left|\sum_{s\in\mathcal{S}} h_s Q(\boldsymbol{x}^P,\xi_s) - \sum_{s\in\mathcal{S}} w_s Q(\boldsymbol{x}^P,\xi_s)\right| \\ &+ \sum_{s\in\mathcal{S}} w_s \left|Q(\boldsymbol{x}^P,\xi_s) - Q(\boldsymbol{x}^A,\xi_s)\right| \\ &+ \sum_{s\in\mathcal{S}} w_s \left|Q(\boldsymbol{x}^A,\xi_s) - Q(\boldsymbol{x}^*,\xi_s)\right|. \end{aligned} \tag{41}$$

The first and last terms can be bounded above by the distance between $\boldsymbol{x}^*$ and its closest neighbor in $\mathcal{X}^A$, leveraging the Lipschitz continuity of the recourse function. Formally, we have:

$$\sum_{s\in\mathcal{S}} h_s \left|Q(\boldsymbol{x}^*,\xi_s) - Q(\boldsymbol{x}^A,\xi_s)\right| \leq \sum_{s\in\mathcal{S}} h_s \cdot L\|\boldsymbol{x}^* - \boldsymbol{x}^A\| \leq \sum_{s\in\mathcal{S}} h_s \cdot L\Delta^A = L\Delta^A$$

where the first inequality follows Lipschitz continuity, the second one comes from the definition of Δ^A, and the third one holds because $\sum_{s\in\mathcal{S}} h_s = 1$. Similarly, we have:

$$\sum_{s\in\mathcal{S}} w_s \left|Q(\boldsymbol{x}^*,\xi_s) - Q(\boldsymbol{x}^A,\xi_s)\right| \leq \sum_{s\in\mathcal{S}} w_s \cdot L\Delta^A \leq K\overline{U}L\Delta^A,$$

where the second inequality follows from Lemma 1 and from the fact that $\sum_{s\in\mathcal{S}} \mathbb{1}(w_s \neq 0) \leq K$ (Equation (2)).

The second and fourth terms can be bounded above due to the small distance between $\boldsymbol{x}^A$ and its closest neighbor in $\mathcal{X}^P$, resulting from the solution pooling model. Formally, we have:

$$\sum_{s\in\mathcal{S}} h_s \left|Q(\boldsymbol{x}^A,\xi_s) - Q(\boldsymbol{x}^P,\xi_s)\right| = \sum_{s\in\mathcal{S}} h_s d_{\boldsymbol{x}^A,\boldsymbol{x}^P,s} \leq \sum_{s\in\mathcal{S}} h_s \Delta^P = \Delta^P$$

where the first inequality follows the definition of $d_{\boldsymbol{x},\boldsymbol{x}',s}$, the second one comes from Equation (8), and the third one holds because $\sum_{s\in\mathcal{S}} h_s = 1$. Similarly, we have:

$$\sum_{s\in\mathcal{S}} w_s \left|Q(\boldsymbol{x}^A,\xi_s) - Q(\boldsymbol{x}^P,\xi_s)\right| \leq \sum_{s\in\mathcal{S}} w_s \Delta^P \leq K\overline{U}\Delta^P.$$

Finally, the third term of Equation (41) is bounded from the SSS model (Equation (2)):

$$\left|\sum_{s\in\mathcal{S}} h_s Q(\boldsymbol{x}^P,\xi_s) - \sum_{s\in\mathcal{S}} w_s Q(\boldsymbol{x}^P,\xi_s)\right| \leq \sum_{\boldsymbol{x}\in\mathcal{X}^P} \left|\sum_{s\in\mathcal{S}} h_s Q(\boldsymbol{x},\xi_s) - \sum_{s\in\mathcal{S}} w_s Q(\boldsymbol{x},\xi_s)\right| \leq \Delta. \tag{42}$$

In summary, we can bound Equation (41) by $\Delta + \left(1 + K\overline{U}\right)\Delta^P + \left(1 + K\overline{U}\right) L\Delta^A$. By proceeding similarly for the second term of Equation (40), we conclude that:

$$\Lambda \leq 2\left[\Delta + \left(1 + K\overline{U}\right)\left(\Delta^P + L\Delta^A\right)\right]. \tag{43}$$

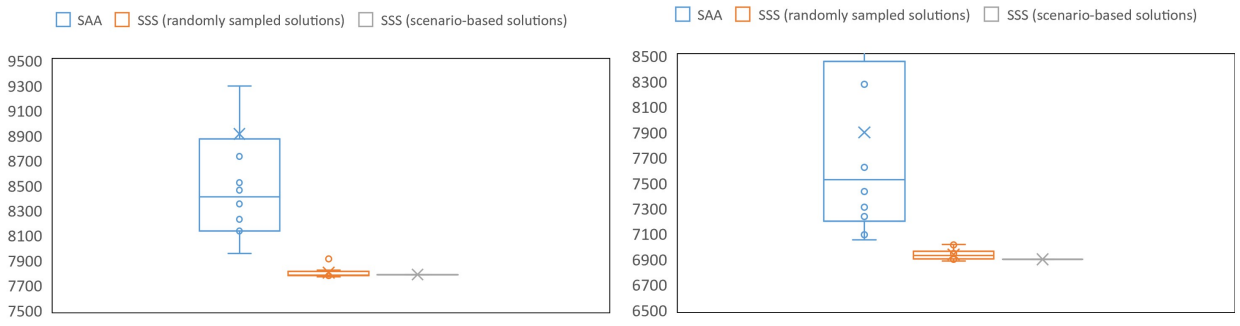
This completes the proof. □

A.3. Design of solution sample $\mathcal{X}^A$ and of the solution pool $\mathcal{X}^P$

Design of $\mathcal{X}^A$. We generate solutions in $\mathcal{X}^A \subseteq \mathcal{X}$ by solving one deterministic problem in each scenario, that is $\mathcal{X}^A = \{\boldsymbol{x}_s^* : s \in \mathcal{S}\}$ where $\boldsymbol{x}_s^* \in \arg\min_{\boldsymbol{x}\in\mathcal{X}} \{\boldsymbol{c}^T\boldsymbol{x} + Q(\boldsymbol{x},\xi_s)\}, \forall s \in \mathcal{S}$. Obviously, these solutions can be of poor quality (**?**). Yet, given the variability in the realizations of uncertainty, these solutions do carry some variability. From a computational standpoint, these solutions can be obtained efficiently via parallelization and can be obtained deterministically (which avoids the need for multiple replications). Still, these solutions are only used as a starting point into our SSS algorithm (akin to **?**, for example).

For FLP, we can also use a random set of solutions in $\mathcal{X}^A$, by sampling each decision variable $x_i \in \{0,1\}$ with a pre-defined probability (equal to the percentage of facilities built in the optimal solution). Results, reported in Figure 7, validate our scenario-based approach for generating the initial pool of solutions: our of the SSS solutions, the one obtained by constructing $\mathcal{X}^A$ through the scenario-specific solutions outperforms the one following random sampling. Most importantly, these results show the robustness of the overall scenario reduction methodology to the initial set $\mathcal{X}^A$: SSS improves upon SAA in terms of average performance and variability regardless of the set $\mathcal{X}^A$. That is, even if the starting point is of poor quality by itself, the SSS method manages to build a strong scenario subset that leads to high-quality solutions.

Design of $\mathcal{X}^P$. Next, we reduce $\mathcal{X}^A$ into a smaller "representative" pool $\mathcal{X}^P$ (Equation (8)). Table 10 reports the sensitivity of our results with the size of the solution pool $\mathcal{X}^P$ for the production routing problem (Section 5). It reports the expected out-of-sample cost (Solution) from the SSS heuristic, the computational time (CPU, in minutes), and the difference to the average SAA solution (%SAA). Note that larger solution pools $\mathcal{X}^P$ generally lead to better PRP solutions. This is consistent with the underlying principle of SSS:



(a) FLP–CR (b) FLP–IR

Figure 7 Comparison of the FLP–CR and FLP–IR solutions with SAA, SSS with random sampling of $\mathcal{X}^A$, and SSS with scenario-based solutions in $\mathcal{X}^A$, using a budget of 10 scenarios.

the larger the solution set, the more closely SSS can approximate the recourse function over the full solution space, leading to a better scenario subsets. However, these improvements come at a cost—longer computational times. In some cases, the increase in computational complexity can outweigh the benefits of a larger solution pool, resulting in worse PRP solutions with a larger solution pool. Still, these results underscore the robustness of the proposed method, as compared to the SAA benchmark: the proposed method reduces the expected PRP costs in 36 out of the 40 problem instances. Based on these results, we choose a budget of $|\mathcal{X}^P| = 90$ to balance strong PRP solutions and reasonable computational times.

Table 10 Sensitivity of SSS heuristic with the size of the solution pool $\mathcal{X}^P$.

		SSS: $\lvert\mathcal{X}^P\rvert = 60$			SSS: $\lvert\mathcal{X}^P\rvert = 90$			SSS: $\lvert\mathcal{X}^P\rvert = 120$			SSS: $\lvert\mathcal{X}^P\rvert = 150$		
Problem	K	Solution	CPU	%SAA	Solution	CPU	%SAA	Solution	CPU	%SAA	Solution	CPU	%SAA
PRP–CR	10	525,420	7	1.5%	519,799	21	2.6%	515,943	42	3.3%	511,677	32	4.1%
	15	520,234	13	0.3%	515,534	31	1.2%	512,166	65	1.8%	508,873	86	2.4%
	20	518,455	23	−1.2%	510,392	39	0.4%	508,165	>120	0.8%	505,970	>120	1.2%
	25	508,360	15	0.3%	502,904	58	1.4%	508,768	>120	0.2%	503,654	>120	1.2%
	30	505,825	22	0.3%	502,903	51	0.9%	508,592	>120	−0.2%	502,080	>120	1.0%
PRP–IR	10	787,825	11	0.7%	776,723	41	2.1%	762,213	65	4.0%	774,728	93	2.4%
	15	760,757	12	2.5%	762,369	33	2.3%	775,616	72	0.6%	775,636	>120	0.6%
	20	771,302	26	−0.1%	763,182	69	0.9%	760,892	>120	1.2%	763,234	>120	0.9%
	25	758,598	23	1.0%	754,271	75	1.6%	755,257	>120	1.5%	754,859	>120	1.5%
	30	766,466	19	−0.4%	755,081	51	1.1%	758,613	>120	0.7%	752,394	>120	1.5%

">120" means that the SSS heuristic does not terminate in 120 minutes.
%SAA = (Average SAA solution − Solution of SSS heuristic)/ Average SAA solution.

A.4. Details on the SSS heuristic

Algorithmic implementation. Our SSS heuristic, outlined in Section 3.4 and in Figure 2, relies on an initial subset of scenarios. We can implement the algorithm in multiple parallel threads, each starting from a different initialization. Ultimately, we select the solution that leads to the lowest SSS objective value, across all threads. Algorithm 2 summarizes the overall algorithm, and Algorithm 3 presents a rule to generate initialized solutions—with the goal of including diverse scenarios in each initial set.

Algorithm 2 A heuristic for solving SSS.

Input: Limited solution pool $\mathcal{X}^P$, scenario budget K, initial scenario set $\mathcal{K}_0$, parameter Ω.

procedure SSS HEURISTIC($\mathcal{X}^P, K, \mathcal{K}_0, \Omega$) ▷ One thread with an initialization
  $\tau \leftarrow 0$ ▷ Iteration index
  while $\mathcal{K}_\tau \neq \mathcal{K}_{\tau-1}$ **do** ▷ Check if the solution is improved
    while $\mathcal{K}_\tau \neq \mathcal{K}_{\tau-1}$ **do** ▷ LSO I: Lines 4–9
      $(\widehat{s}_1, \cdots, \widehat{s}_K) \leftarrow \mathcal{K}_\tau$ ▷ Update the neighborhood centers
      $(\mathcal{S}_1, \cdots, \mathcal{S}_K) \leftarrow SNM(\mathcal{X}^P, \widehat{s}_1, \cdots, \widehat{s}_K, \Omega)$ ▷ Re-define scenario neighborhoods
      $\mathcal{K}_{\tau+1} \leftarrow NSSS(\mathcal{X}^P, \mathcal{S}_1, \cdots, \mathcal{S}_K)$ ▷ Conduct local search
      $\tau \leftarrow \tau + 1$
    end while
    for $k = 1, \cdots, K$ **do** ▷ LSO II: Lines 10–16
      $(\widehat{s}_1, \cdots, \widehat{s}_K) \leftarrow \mathcal{K}_\tau$ ▷ Update the incumbent solution
      $\mathcal{K}_{\tau+1} \leftarrow PSSS(\mathcal{X}^P, k, \widehat{s}_1, \cdots, \widehat{s}_K)$ ▷ Conduct local search
      if $\mathcal{K}_{\tau+1} \neq \mathcal{K}_\tau$ **then**
        break ▷ Stops LSO II if the solution is improved
      end if
    end for
    $\tau \leftarrow \tau + 1$ and $\widetilde{\mathcal{K}} \leftarrow \mathcal{K}_\tau$ ▷ Record the incumbent solution
  end while
end procedure

Output: $\widetilde{\boldsymbol{w}} \leftarrow \arg\min_{\boldsymbol{w} \geq \mathbf{0}} \left\{ \sum_{\boldsymbol{x} \in \mathcal{X}^P} \left| \sum_{s \in \mathcal{S}} h_s Q(\boldsymbol{x}, \xi_s) - \sum_{s \in \mathcal{S}} w_s Q(\boldsymbol{x}, \xi_s) \right|, \text{ s.t. } w_s = 0, \ \forall s \notin \widetilde{\mathcal{K}} \right\}$

Impact of LSO I and LSO II. Overall, Algorithm 2 solves the SSS model via two local search operators, until convergence: an inner procedure (LSO I) that iteratively improves the solution within neighborhoods, and a perturbation scheme (LSO II) to escape from local optima. Figure 8 shows a typical evolution of the SSS objective function value over time by LSO I and II, for the production routing problem (see Section 5). If we were only using LSO I, the algorithm would terminate when reaching the first local optimum, with an SSS solution of 113,010. However, with LSO II, the algorithm terminates after 4 outer iterations and 16 overall iterations, with an SSS of 75,794. This demonstrates the benefits of LSO II for escaping the local optimum—in this case, it improves the SSS solution by 32.9%. In all our experiments, the algorithm terminates within just a few outer iterations, for all problems and all scenario budgets.

Selecting the value of Ω. Within the inner procedure (LSO I), the parameter Ω defines the size of the neighborhood. If Ω is small, each iteration will be fast, but the local search can converge to a poor local optimum. If Ω is large, each iteration will be more impactful but slower. We seek for a balance between solution quality and computational time by adjusting the product of $K \times \Omega$. Results are reported in Table 11.

Initially, the SSS solution improves as Ω increases: a larger Ω defines a larger neighborhood, hence aids the algorithm escape from local optima. However, as Ω increases further, the NSSS model becomes too

Algorithm 3 Scenario set initializations and multi-thread implementation.

Input: Limited solution pool $\mathcal{X}^P$, scenario budget K, parameter Ω, number of threads Θ_{max}.

procedure SSS INITIALIZATION
  $\Theta \leftarrow 1$ ▷ Generate the initial scenario set for thread 1
  $\mathcal{K}_0^1 \leftarrow \{s_0\}$, where s_0 is randomly sampled in the full scenario set $\mathcal{S}$
  for $k \leftarrow 2$ to K **do**
    $\mathcal{K}_0^1 \leftarrow \mathcal{K}_0^1 \cup \{s'\}, s' \in \arg\max_{s \in \mathcal{S} \setminus \mathcal{K}_0^1} \left\{ \sum_{s' \in \mathcal{K}_0^1} D_{s,s'} \right\}$ ▷ Select scenarios to maximize coverage
  end for
  for $\Theta \leftarrow 2$ to Θ_{max} **do** ▷ Generate the initial scenario set for threads $\Theta \in \{2, \cdots, \Theta_{max}\}$
    $(\widehat{s}_1, \cdots, \widehat{s}_K) \leftarrow \mathcal{K}_0^{\Theta-1}$
    $(\mathcal{S}_1, \cdots, \mathcal{S}_K) \leftarrow SNM(\mathcal{X}^P, \widehat{s}_1, \cdots, \widehat{s}_K, \lfloor |\mathcal{S}|/K \rfloor)$ ▷ Partition scenario set $\mathcal{S}$ into K pools
    $\mathcal{K}_0^\Theta \leftarrow \{s'_k, k = 1, \cdots, K\}, s'_k \in \arg\max_{s \in \mathcal{S}_k} \{D_{s\widehat{s}_k}\}, k = 1, \cdots, K$ ▷ Maximize difference from previous solution
  end for
end procedure
for $\Theta \leftarrow 1$ to Θ_{max} **do** ▷ Solve SSS for each initialization, using multi-thread parallelization
  $\widetilde{\boldsymbol{w}}^\Theta \leftarrow$ SSS HEURISTIC$(\mathcal{X}^P, K, \mathcal{K}_0^\Theta, \Omega)$ (Algorithm 2)
end for

Output: $\widetilde{\boldsymbol{w}} \in \arg\min_{\Theta=1,\cdots,\Theta_{max}} \left\{ \sum_{\boldsymbol{x} \in \mathcal{X}^P} \left| \sum_{s \in \mathcal{S}} h_s Q(\boldsymbol{x}, \xi_s) - \sum_{s \in \mathcal{S}} w_s^\Theta Q(\boldsymbol{x}, \xi_s) \right| \right\}$

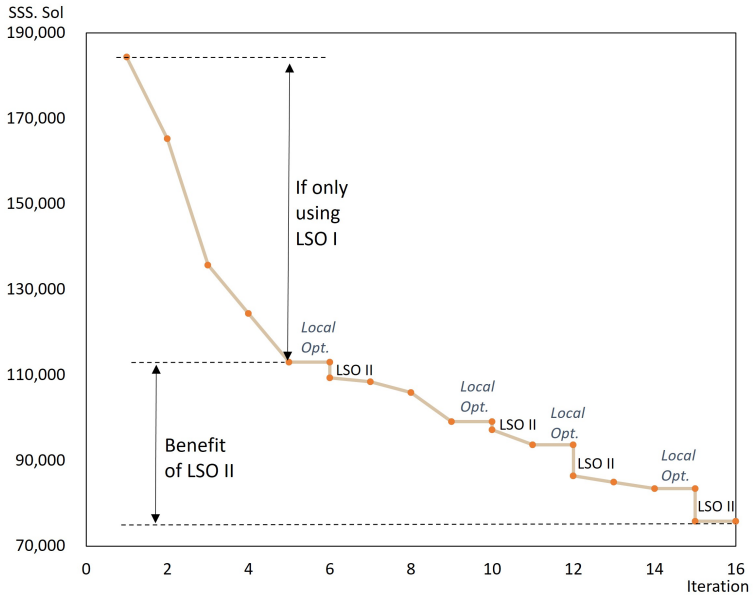


Figure 8 **Benefit of LSO II (PRP–CR problem, $K = 20$, $|\mathcal{X}^P| = 60$).**

computationally intensive at each iteration, and fails to terminate within the time limit. In turn, the solution actually deteriorates while the algorithm becomes slower. These results suggests a "sweet spot" in the value of

Table 11 Computational times and SSS solution as a function of the parameter Ω, for different values of K.

	$K=10$			$K=20$			$K=30$		
	Ω	CPU(min)	SSS.Sol	Ω	CPU(min)	SSS.Sol	Ω	CPU(min)	SSS.Sol
$K\times\Omega=50$–60	5	15	507,731	3	31	256,553	2	47	123,427
$K\times\Omega=90$–100	10	21	491,396	5	39	228,149	3	51	109,817
$K\times\Omega=140$–150	15	>120	647,570	7	>120	326,315	5	>120	109,817

Ω such that $K\times\Omega$ lies around 90–100, which leads to the strongest SSS solution in reasonable computational times. Note, importantly, that this "sweet spot" is robust across all values of K.

Appendix B: Details on the scenario assortment optimization approach

B.1. A row-generation algorithm for SAO and SG

Recall that our scenario assortment optimization (SAO) problem extends the scenario grouping (SG) approach from **?**. Both SAO and SG cannot be solved directly, because their objective functions call the stochastic programming formulation (Equation (13)). In Section 4.2, we developed a new column-evaluation-and-generation algorithm to solve the SAO model. In this appendix, we describe a benchmark decomposition algorithm based on row generation, inspired by **?**, which can be applied to SAO and SG (we focus the exposition on SAO). This algorithm will be used as a benchmark to assess the results of the column-evaluation-and-generation algorithm.

The first step involves reformulating SAO as a mixed-integer linear program (MILP), using an epigraph representation. This is stated in Proposition 3. Equation (44) maximizes the lower bound; Equation (45) retrieves the optimal solution for each bundle; Equation (46) enforces the budget of K scenarios per bundle; Equations (47) and (48) state the assumptions of Proposition 1; Equation (49) ensures that $\beta_{bs}=0$ if $s\notin\mathcal{S}_b$; Equation (50) defines the domain of all variables.

Proposition 3. *Scenario assortment optimization (Equation* (14)*) is equivalent to the MILP:*

$$MP(\mathcal{X}) = \max \sum_{b\in\mathcal{B}} O_b \tag{44}$$

$$s.t.\ \ O_b \le \eta_b \boldsymbol{c}^T\boldsymbol{x} + \sum_{s\in\mathcal{S}} \beta_{bs} Q(\boldsymbol{x},\xi_s) \quad \forall b\in\mathcal{B}, \boldsymbol{x}\in\mathcal{X} \tag{45}$$

$$\sum_{s\in\mathcal{S}} \theta_{bs} \le K \quad \forall b\in\mathcal{B} \tag{46}$$

$$\sum_{b\in\mathcal{B}} \eta_b = 1 \tag{47}$$

$$\sum_{b\in\mathcal{B}} \beta_{bs} = h_s \quad \forall s\in\mathcal{S} \tag{48}$$

$$\beta_{bs} \le h_s\theta_{bs} \quad \forall s\in\mathcal{S}, b\in\mathcal{B} \tag{49}$$

$$\theta_{bs}\in\{0,1\}, \beta_{bs}\ge 0, \eta_b\ge 0, O_b\in\mathbb{R} \quad \forall s\in\mathcal{S}, b\in\mathcal{B}. \tag{50}$$

Yet, the problem remains intractable, because Equation (45) enumerates all feasible first-stage solutions in $\mathcal{X}$. Accordingly, the row-generation algorithm iterates between a master problem and a subproblem:

– Master problem $MP(\mathcal{X}^R)$: we solve Equations (44)–(50) with a subset of feasible first-stage solutions $\mathcal{X}^R\subseteq\mathcal{X}$ (as opposed to the full set as in Equation (45)). This yields a "relaxation bound" (RB), i.e.,

an upper bound of $MP(\mathcal{X})$. Let $(\boldsymbol{\theta}^*, \boldsymbol{\beta}^*, \boldsymbol{\eta}^*, \boldsymbol{O}^*)$ be the optimal solution of $MP(\mathcal{X}^R)$. We retrieve the incumbent bundles $\mathcal{S}_b^* = \{s \in \mathcal{S} : \theta_{bs}^* = 1\}$ for all $b \in \mathcal{B}$.

– Subproblem $\mathcal{P}(\mathcal{S}_b^*, \eta_b^*, \boldsymbol{\beta}_b^*)$: we solve the stochastic program for each bundle $b \in \mathcal{B}$ (Equation (13)). By design, this problem involves at most K scenarios. Let $\boldsymbol{x}_b^*$ be its optimal solution. We expand the solution pool $\mathcal{X}^R$ with all solutions $\boldsymbol{x}_b^*,\ b \in \mathcal{B}$. We then compute $\sum_{b\in\mathcal{B}} \mathcal{P}(\mathcal{S}_b^*, \eta_b^*, \boldsymbol{\beta}_b^*)$, which yields a lower bound to $MP(\mathcal{X})$, hence, from Proposition 1, to the full stochastic program $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$ (LB).

We iterate between the master problem and the subproblems, until $RB = LB$. In that case, the solution provides the tightest possible bound from the scenario assortment approach (Equation (14)). Algorithm 4 summarizes this approach and Proposition 4 establishes its convergence.

Algorithm 4 Iterative row-generation algorithm for solving SAO (Equation (14)).

Input: Initial set $\mathcal{X}^R$, $RB = +\infty$ (relaxation bound), $LB = -\infty$ (lower bound), tolerance ε.

1: **while** $\frac{RB-LB}{LB} > \varepsilon$ or $LB = -\infty$ **do**
2: Solve $MP(\mathcal{X}^R)$; store the optimum $(\boldsymbol{\theta}^*, \boldsymbol{\beta}^*, \boldsymbol{\eta}^*, \boldsymbol{O}^*)$; update $RB \leftarrow MP(\mathcal{X}^R)$
3: Solve $\mathcal{P}(\mathcal{S}_b^*, \eta_b^*, \boldsymbol{\beta}_b^*)$ for each $b \in \mathcal{B}$; update $LB \leftarrow \max\left\{\sum_{b\in\mathcal{B}} \mathcal{P}(\mathcal{S}_b^*, \eta_b^*, \boldsymbol{\beta}_b^*), LB\right\}$
4: $\mathcal{X}^R \leftarrow \mathcal{X}^R \cup \{\boldsymbol{x}_b^*, b \in \mathcal{B}\}$ ▷ Row generation: update solution set
5: **end while**

Output: LB

PROPOSITION 4. *Algorithm 4 converges to the optimal solution of Equation* (14)*: if $LB = RB$, then the optimal solution of $MP(\mathcal{X}^R)$ is also optimal for $MP(\mathcal{X})$. Moreover, the algorithm improves the relaxation bound and the lower bound at each iteration, and terminates in a finite number of iterations if $|\mathcal{X}|$ is finite.*

B.2. Proof of statements

Proof of Proposition 1 Let $\boldsymbol{x}^*$ and $\boldsymbol{x}_b^*$ be optimal solutions of $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$ and $\mathcal{P}(\mathcal{S}_b, \eta_b, \boldsymbol{\beta}_b)$, respectively. We have:

$$\eta_b \boldsymbol{c}^T \boldsymbol{x}_b^* + \sum_{s\in\mathcal{S}_b} \beta_{bs} Q(\boldsymbol{x}_b^*, \xi_s) \leq \eta_b \boldsymbol{c}^T \boldsymbol{x}^* + \sum_{s\in\mathcal{S}_b} \beta_{bs} Q(\boldsymbol{x}^*, \xi_s), \quad \forall b \in \mathcal{B}. \tag{51}$$

By taking the summation over $b \in \mathcal{B}$, it holds that

$$\sum_{b\in\mathcal{B}} \left(\eta_b \boldsymbol{c}^T \boldsymbol{x}_b^* + \sum_{s\in\mathcal{S}_b} \beta_{bs} Q(\boldsymbol{x}_b^*, \xi_s) \right) \leq \sum_{b\in\mathcal{B}} \left(\eta_b \boldsymbol{c}^T \boldsymbol{x}^* + \sum_{s\in\mathcal{S}_b} \beta_{bs} Q(\boldsymbol{x}^*, \xi_s) \right). \tag{52}$$

Since, by definition, $\sum_{b\in\mathcal{B}} \eta_b = 1$ and $\sum_{b\in\mathcal{B}} \beta_{bs} = h_s$ for all $s \in \mathcal{S}$, we obtain:

$$\sum_{b\in\mathcal{B}} \left(\eta_b \boldsymbol{c}^T \boldsymbol{x}_b^* + \sum_{s\in\mathcal{S}_b} \beta_{bs} Q(\boldsymbol{x}_b^*, \xi_s) \right) \leq \boldsymbol{c}^T \boldsymbol{x}^* + \sum_{s\in\mathcal{S}} h_s Q(\boldsymbol{x}^*, \xi_s), \tag{53}$$

indicating $\sum_{b\in\mathcal{B}} \mathcal{P}(\mathcal{S}_b, \eta_b, \boldsymbol{\beta}_b)$ is a valid lower bound to $\mathcal{P}(\mathcal{S}, \boldsymbol{h})$. □

Proof of Proposition 2 Let Z_1 and Z_2 denote the optimal values of Equation (14) and of Equations (16)–(19), respectively. Note that since $\mathcal{B}^{all}$ is the inclusive set of all possible bundles, we have $\mathcal{B} \subseteq \mathcal{B}^{all}$.

Let us first consider an optimal solution to Equation (14), denoted as $\mathcal{S}_b^*, \eta_b^*, \beta_{bs}^*$ for all $b \in \mathcal{B}$. We define a solution to Equations (16)–(19) as follows:

$$y_b = 1, \ \forall b \in \mathcal{B}$$
$$y_b = 0, \ \forall b \in \mathcal{B}^{all} \setminus \mathcal{B}.$$

By construction, this solution defines a feasible solution to the set partition formulation. Moreover, it achieves the same objective function value, since $\sum_{b\in\mathcal{B}^{all}} O_b y_b = \sum_{b\in\mathcal{B}} O_b = \sum_{b\in\mathcal{B}} \mathcal{P}(\mathcal{S}_b^*, \eta_b^*, \boldsymbol{\beta}_b^*)$. Therefore, $Z_1 \leq Z_2$.

Conversely, let us consider an optimal solution of Equations (16)–(19). We extract a subset $\mathcal{B}^* = \{b \in \mathcal{B}^{all} : y_b^* > 0\}$. For each $b \in \mathcal{B}^*$, we map it to only one $b' \in \mathcal{B}$ (which is feasible as long as $|\mathcal{B}| \geq |\mathcal{B}^*|$) by setting $\mathcal{S}_{b'} = \mathcal{S}_b$, $\eta_{b'} = y_b^* \eta_b$, $\beta_{b's} = y_b^* \beta_{bs}$ for all $s \in \mathcal{S}$. For any $b' \in \mathcal{B}$ such that no bundle from $\mathcal{B}^*$ is mapped into b', we set $\mathcal{S}_{b'} = \emptyset$, $\eta_{b'} = 0$, and $\beta_{b's} = 0$ for all $s \in \mathcal{S}$. Then, we have the same objective function value $\sum_{b'\in\mathcal{B}} \mathcal{P}(\mathcal{S}'_b, \eta_{b'}, \boldsymbol{\beta}_{b'}) = \sum_{b\in\mathcal{B}^*} \mathcal{P}(\mathcal{S}_b, \eta_b, \boldsymbol{\beta}_b) y_b^* = \sum_{b\in\mathcal{B}^*} O_b y_b^* = \sum_{b\in\mathcal{B}^{all}} O_b y_b^*$. Therefore, $Z_1 \geq Z_2$. □

Proof of Theorem 2 Let $\widetilde{\boldsymbol{x}}$ denote an optimal solution to $OPT(\widetilde{\boldsymbol{c}})$. Upon convergence, we have $\widetilde{c}_i = c_i$ for all $i = 1, \cdots, n$ such that $\widetilde{x}_i > 0$. Then, for all $\boldsymbol{x} \in \mathcal{X}$, the following holds:

$$\boldsymbol{c}^\top \widetilde{\boldsymbol{x}} = \widetilde{\boldsymbol{c}}^\top \widetilde{\boldsymbol{x}} \geq \widetilde{\boldsymbol{c}}^\top \boldsymbol{x} \geq \boldsymbol{c}^\top \boldsymbol{x}, \tag{54}$$

where the first equality stems from the stopping criterion, the second inequality comes from the fact that $\widetilde{\boldsymbol{x}}$ solves $OPT(\widetilde{\boldsymbol{c}})$, and the third inequality is due to the fact that $\widetilde{\boldsymbol{c}} \geq \boldsymbol{c}$. This proves that the algorithm returns an optimal solution of $OPT(\boldsymbol{c})$ upon termination.

Next, denote by $\boldsymbol{y}^{(k)}$ the solution of the algorithm at the k^{th} iteration, and assume that there exist $k < l$ such that $\boldsymbol{y}^{(k)} = \boldsymbol{y}^{(l)}$. Denote by $\widetilde{\mathcal{B}}$ the set of bundles $b \in \mathcal{B}^{all}$ such that $y_b^{(k)} = y_b^{(l)} > 0$. Then, we know that at the l^{th} iteration, $\widetilde{O}_b = O_b$ for all $b \in \widetilde{\mathcal{B}}$. Therefore, the subsequent column evaluation step will no longer update the objective parameters, and the algorithm terminates. Therefore, the column evaluation procedure never cycles back to previously visited solutions. Since the solution space is finite and there is at most one parameter update per variable, column evaluation terminates in a finite number of iterations. □

Proof of Proposition 3 Let Z_1 and Z_2 denote the optimal values of Equation (14) and of Equations (44)–(50), respectively. Let us first consider an optimal solution to Equation (14), denoted as $\mathcal{S}_b^*, \eta_b^*, \beta_{bs}^*$. We define a solution to Equations (44)–(50) as follows:

$$O_b = \mathcal{P}(\mathcal{S}_b, \eta_b, \boldsymbol{\beta}_b), \ \forall b \in \mathcal{B}$$
$$\theta_{bs} = \begin{cases} 1 & \text{if } s \in \mathcal{S}_b^* \\ 0 & \text{otherwise} \end{cases}$$
$$\eta_b = \eta_b^*, \ \forall b \in \mathcal{B}$$
$$\beta_{bs} = \beta_{bs}^*, \ \forall b \in \mathcal{B}, s \in \mathcal{S}.$$

By construction, this solution defines a feasible solution to $MP(\mathcal{X})$. Moreover, it achieves the same objective function value, since $\sum_{b\in\mathcal{B}} O_b = \sum_{b\in\mathcal{B}} \mathcal{P}(\mathcal{S}_b^*, \eta_b^*, \boldsymbol{\beta}_b^*)$. Therefore, $Z_1 \leq Z_2$.

Conversely, let us consider an optimal solution of Equations (44)–(50). We define a solution to Equation (14) by letting $\mathcal{S}_b = \{s \in \mathcal{S} | \theta^*_{bs} = 1\}$, $\eta_b = \eta^*_b$, $\beta_{bs} = \beta^*_{bs}$ for all $b \in \mathcal{B}, s \in \mathcal{S}$. By construction, this solution is feasible to Equation (14). Meanwhile, we know from Equations (44)–(45) that $O^*_b = \min_{\boldsymbol{x} \in \mathcal{X}} \{\eta^*_b \boldsymbol{c}^T \boldsymbol{x} + \sum_{s \in \mathcal{S}} \beta^*_{bs} Q(\boldsymbol{x}, \xi_s)\}$ for all $b \in \mathcal{B}$. This implies that $O^*_b = \mathcal{P}(\mathcal{S}_b, \eta_b, \boldsymbol{\beta}_b)$ for all $b \in \mathcal{B}$, hence $\sum_{b \in \mathcal{B}} O^*_b = \sum_{b \in \mathcal{B}} \mathcal{P}(\mathcal{S}_b, \eta_b, \boldsymbol{\beta}_b)$. Therefore, $Z_1 \geq Z_2$. □

Proof of Proposition 4 First, by design, the restricted set $\mathcal{X}^R$ of feasible first-stage solutions gets expanded from one iteration to the next. It is easy to see that, for any $\mathcal{X}^R \subseteq \widehat{\mathcal{X}}^C$, the master problem $MP(\mathcal{X}^R)$ is a relaxation of $MP(\widehat{\mathcal{X}}^C)$. Therefore, the relaxation bound RB is non-increasing over the iterations.

Second, the solution $\sum_{b \in \mathcal{B}} \mathcal{P}(\mathcal{S}^*_b, \eta^*_b, \boldsymbol{\beta}^*_b)$ is not necessarily non-decreasing over the iterations. However, by design, we keep the best lower bound at each iteration (line 4 of Algorithm 4). Therefore, the lower bound LB is non-decreasing over the iterations.

Next, let us consider an iteration, with a restricted solution pool $\mathcal{X}^R$, such that $LB = RB$. We show that $MP(\mathcal{X}^R) = MP(\mathcal{X})$.

- Since $\mathcal{X}^P \subseteq \mathcal{X}$, we clearly have $RB = MP(\mathcal{X}^R) \geq MP(\mathcal{X})$.
- By design, there exists a master problem solution derived in the iterations to date, denoted by $(\boldsymbol{\theta}^*, \boldsymbol{\beta}^*, \boldsymbol{\eta}^*, \boldsymbol{O}^*)$, such that $LB = \sum_{b \in \mathcal{B}} \mathcal{P}(\mathcal{S}^*_b, \eta^*_b, \boldsymbol{\beta}^*_b)$, with $\mathcal{S}^*_b = \{s \in \mathcal{S} | \theta^*_{bs} = 1\}$. Let us introduce $O'_b = \mathcal{P}(\mathcal{S}^*_b, \eta^*_b, \boldsymbol{\beta}^*_b) = \min_{\boldsymbol{x} \in \mathcal{X}} \{\eta^*_b \boldsymbol{c}^T \boldsymbol{x} + \sum_{s \in \mathcal{S}} \beta^*_{bs} Q(\boldsymbol{x}, \xi_s)\}$ for all $b \in \mathcal{B}$. Then, $(\boldsymbol{\theta}^*, \boldsymbol{\beta}^*, \boldsymbol{\eta}^*, \boldsymbol{O}')$ is a feasible solution to $MP(\mathcal{X})$, achieving an objective of $\sum_{b \in \mathcal{B}} O'_b = LB$. Therefore, $MP(\mathcal{X}) \geq LB$.

This shows that $MP(\mathcal{X}^R) = MP(\mathcal{X})$.

Finally, we show that the restricted set $\mathcal{X}^R$ is being updated until $LB = RB$. Suppose by contradiction that, at a given iteration, the set $\mathcal{X}^R$ can no longer be updated, that is, $\boldsymbol{x}^*_b \in \mathcal{X}^R$ for all $b \in \mathcal{B}$ (where $\boldsymbol{x}^*_b$ denotes the optimal solution of $\mathcal{P}(\mathcal{S}^*_b, \eta^*_b, \boldsymbol{\beta}^*_b)$). By definition, the optimal solution satisfies $O^*_b = \eta^*_b \boldsymbol{c} \boldsymbol{x}^*_b + \sum_{s \in \mathcal{S}} \beta^*_{bs} Q(\boldsymbol{x}^*_b, \xi_s), \forall b \in \mathcal{B}$. Since $\boldsymbol{x}^*_b \in \mathcal{X}^R$, we have:

$$O^*_b \leq \eta^*_b \boldsymbol{c}^T \boldsymbol{x} + \sum_{s \in \mathcal{S}} \beta^*_{bs} Q(\boldsymbol{x}, \xi_s) \quad \forall b \in \mathcal{B}, \boldsymbol{x} \in \mathcal{X}^R.$$

Hence, Equation (45) of $MP(\mathcal{X}^R)$ is satisfied, and the other constraints are also clearly satisfied. Therefore, we have $MP(\mathcal{X}^P) = \sum_{b \in \mathcal{B}} \mathcal{P}(\mathcal{S}^*_b, \eta^*_b, \boldsymbol{\beta}^*_b)$, hence $RB = LB$. As we showed earlier, this implies that the algorithm terminates at that point.

In summary, the algorithm keeps updating the restricted set $\mathcal{X}^R$ until termination. Over the iterations, the relaxation bound RB is non-increasing and the lower bound LB is non-decreasing. When they match, the algorithm terminates at the optimal solution of Equation (14). If the set of solutions in $\mathcal{X}$ is finite, the algorithm terminates in a finite number of iterations. □

Appendix C: Formulation of the optimization problems

C.1. Production routing problems

Graph structure. Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ denote a complete undirected graph, where $\mathcal{N} = \{0, \cdots, n\}$ is the set of nodes and $\mathcal{E} = \{(i,j) : i, j \in \mathcal{N}, i < j\}$ is the set of edges. By convention, node 0 represents the plant and $\mathcal{N}_c = \mathcal{N} \setminus \{0\}$ represents customers. A transportation cost c_{ij} is incurred between nodes i and j. We define the extended graph $\overline{\mathcal{G}} = (\overline{\mathcal{N}}, \overline{\mathcal{E}})$ by adding node $n+1$, a copy of node 0, so that $\overline{\mathcal{N}} = \mathcal{N} \cup \{n+1\}$, $\overline{\mathcal{E}} = \mathcal{E} \cup \{\{i, n+1\}, i \in \mathcal{N}_c\}$, and $c_{i,n+1} = c_{0i}, \forall\ i \in \mathcal{N}_c$.

Time horizon. We consider a discrete and finite time horizon, denoted by $\mathcal{T}=\{1,\cdots,T\}$.

Demand. Let $\mathcal{S}$ denote a set of demand scenarios, and let h_s be the probability of scenario s. Let ϕ_{its} denote the demand from customer i in time period t under scenario s. Following **?**, we assume that all random variables get realized simultaneously at the beginning of the second stage. A unit cost σ_i is incurred if some demand of customer i is unmet at the end of a period (backlogging is not allowed).

Inventory and replenishment. At the beginning of the planning horizon, an initial inventory I_{i0} is available at node i. Let also $I_{i0s}=I_{i0}$ for $i\in\mathcal{N}$, $s\in\mathcal{S}$. In each period, a production capacity of $\overline{C}$ is available. A fixed plant setup cost f is incurred if production takes place, and it associates with a unit production cost u. A set of F identical vehicles of capacity $\overline{Q}$ can be dispatched from the plant to deliver inventory to customers. Inventory can be held both at the plant and at the customer nodes. The inventory held at node i cannot exceed L_i. A unit inventory holding cost ρ_i is incurred in each period.

The following decision variables are common to PRP–CR and PRP–IR. The first-stage decisions include plant setup decisions (defined by a variable y_t equal to 1 if production takes place in period t, and 0 otherwise) and customer appointments (defined by a variable z_{it} equal to 1 if node i will be visited in period t, and 0 otherwise). The second-stage decisions include the production quantities p_{ts}, the delivery quantities q_{its}, the amount of unmet demand e_{its}, and the inventory level I_{its} (by the end of the period), for period $t\in\mathcal{T}$, customer $i\in\mathcal{N}_c$ and scenario $s\in\mathcal{S}$. As stated earlier, the main difference lies in whether the routing decisions are made in the first stage or the second stage; accordingly, they will be modeled by means of scenario-agnostic variables χ_{ijt} in PRP–CR and scenario-dependent variables χ_{ijts} in PRP–IR.

PRP–CR formulation. In PRP–CR, the firm pre-commits to the vehicles' routes in the first stage. We thus add a first-stage variable χ_{ijt} equal to 1 if a vehicle travels from node i to node j in period t. We add auxiliary first-stage flow variables $\bar{v}_{ijt}$ equal to the vehicle load on edge $\{i,j\}\in\overline{\mathcal{E}}$, and $\bar{v}_{jit}$ equal to the empty space on the vehicle on edge $\{i,j\}\in\overline{\mathcal{E}}$ i.e., $\bar{v}_{jit}=\overline{Q}-\bar{v}_{ijt}$ (see **?** for more descriptions on these variable definitions). We define their second-stage counterparts as v_{ijts} and v_{jits}.

Since the routing decisions are made in the first stage but customer demand is materialized in the second stage, the routing decisions cannot capture the delivery quantity at each node. Yet, in the two-commodity flow formulation, we need to capture the flows of the vehicle loads. To address this issue, we assume a very small delivery quantity $\varepsilon\ll\overline{Q}$ for each customer. This is equivalent to making routing decisions without load consideration in the first stage, and then adjusting load quantities upon observing demand realizations.

The PRP–CR is formulated as follows, where $M_{ts}=\min\{\overline{C},\sum_{i\in\mathcal{N}_c}\sum_{t'=t}^{T}\phi_{it's}\}$ and $M'_{its}=\min\{L_i,\overline{Q},\sum_{t'=t}^{T}\phi_{it's}\}$ denote Big-M parameters.

$$PRP\text{–}CR(\mathcal{S},\boldsymbol{h})=\min\sum_{t\in\mathcal{T}}\left(fy_t+\sum_{(i,j)\in\overline{\mathcal{E}}}c_{ij}\chi_{ijt}\right)+\sum_{s\in\mathcal{S}}h_sQ_s(\boldsymbol{\chi},\boldsymbol{y},\boldsymbol{z}),\tag{55}$$

$$\text{s.t.}\sum_{j\in\overline{\mathcal{N}},i<j}\chi_{ijt}+\sum_{j\in\overline{\mathcal{N}},i>j}\chi_{jit}=2z_{it}\quad\forall i\in\mathcal{N}_c,\ \forall t\in\mathcal{T}\tag{56}$$

$$\sum_{j\in\overline{\mathcal{N}}}(\bar{v}_{jit}-\bar{v}_{ijt})=2\varepsilon z_{it}\quad\forall i\in\mathcal{N}_c,\ \forall t\in\mathcal{T}\tag{57}$$

$$\sum_{j\in\mathcal{N}_c}\bar{v}_{0jt}=\sum_{j\in\mathcal{N}_c}\varepsilon z_{jt}\quad\forall t\in\mathcal{T} \tag{58}$$

$$\sum_{j\in\mathcal{N}_c\cup\{n+1\}}\bar{v}_{j0t}=F\overline{Q}-\sum_{j\in\mathcal{N}_c}\varepsilon z_{jt}\quad\forall t\in\mathcal{T} \tag{59}$$

$$\sum_{j\in\mathcal{N}_c\cup\{0\}}\bar{v}_{n+1,j,t}=F\overline{Q}\quad\forall t\in\mathcal{T} \tag{60}$$

$$\bar{v}_{ijt}+\bar{v}_{jit}=\overline{Q}\chi_{ijt}\quad\forall i,j\in\overline{\mathcal{E}},\ \forall t\in\mathcal{T} \tag{61}$$

$$\bar{v}_{ijt}\geq 0,\bar{v}_{jit}\geq 0\quad\forall\{i,j\}\in\overline{\mathcal{E}},\ \forall t\in\mathcal{T} \tag{62}$$

$$y_t,z_{it}\in\{0,1\}\quad\forall i\in\mathcal{N},\ \forall t\in\mathcal{T} \tag{63}$$

$$\chi_{ijt}\in\{0,1\}\quad\forall\{i,j\}\in\overline{\mathcal{E}},\ \forall t\in\mathcal{T}, \tag{64}$$

where for each $s\in\mathcal{S}$,

$$Q_s(\boldsymbol{\chi},\boldsymbol{y},\boldsymbol{z})=\min\sum_{t\in\mathcal{T}}\left(up_{ts}+\sum_{i\in\mathcal{N}}\rho_i I_{its}+\sum_{i\in\mathcal{N}_c}\sigma_i e_{its}\right) \tag{65}$$

$$\text{s.t. } I_{0,t-1,s}+p_{ts}=\sum_{i\in\mathcal{N}_c}q_{its}+I_{0ts}\quad\forall t\in\mathcal{T} \tag{66}$$

$$I_{i,t-1,s}+q_{its}+e_{its}=\phi_{its}+I_{its}\quad\forall i\in\mathcal{N}_c,\ \forall t\in\mathcal{T} \tag{67}$$

$$I_{0ts}\leq L_0\quad\forall t\in\mathcal{T} \tag{68}$$

$$I_{its}+\phi_{its}\leq L_i\quad\forall i\in\mathcal{N}_c,\ \forall t\in\mathcal{T} \tag{69}$$

$$p_{ts}\leq M_{ts}y_t\quad\forall t\in\mathcal{T} \tag{70}$$

$$q_{its}\leq M'_{its}z_{it}\quad\forall i\in\mathcal{N}_c,\ \forall\ t\in\mathcal{T} \tag{71}$$

$$\sum_{j\in\overline{\mathcal{N}}}(v_{jits}-v_{ijts})=2q_{its}\quad\forall i\in\mathcal{N}_c,\ \forall t\in\mathcal{T} \tag{72}$$

$$\sum_{j\in\mathcal{N}_c}v_{0jts}=\sum_{j\in\mathcal{N}_c}q_{jts}\quad\forall t\in\mathcal{T} \tag{73}$$

$$\sum_{j\in\mathcal{N}_c\cup\{n+1\}}v_{j0ts}=F\overline{Q}-\sum_{j\in\mathcal{N}_c}q_{jts}\quad\forall t\in\mathcal{T} \tag{74}$$

$$\sum_{j\in\mathcal{N}_c\cup\{0\}}v_{n+1,j,t,s}=F\overline{Q}\quad\forall t\in\mathcal{T} \tag{75}$$

$$v_{ijts}+v_{jits}=\overline{Q}\chi_{ijt}\quad\forall i,j\in\overline{\mathcal{E}},\ \forall t\in\mathcal{T} \tag{76}$$

$$v_{ijts}\geq 0,v_{jits}\geq 0\quad\forall\{i,j\}\in\overline{\mathcal{E}},\ \forall t\in\mathcal{T} \tag{77}$$

$$e_{its},p_{ts},I_{its},q_{its}\geq 0\quad\forall i\in\mathcal{N},\ \forall t\in\mathcal{T}. \tag{78}$$

Equation (55) minimizes the cost of the first-stage decisions and the expected cost of the second-stage decisions. Equation (56) requires the number of incident edges to be 2 if customer i is visited in period t. Equations (57)–(61) are typical flow constraints in the two-commodity flow formulation. Equations (62)–(64) define the domain of the first-stage variables.

For each scenario, Equation (65) minimizes the total cost of production, inventory, and unmet demand. Equations (66) and (67) enforce the inventory flow balance at the plant and customer nodes, respectively. Equations (68) and (69) impose the maximum inventory level. Equation (70) allows a positive production quantity only if the plant is set up. Equation (71) allows a positive delivery quantity only if the corresponding

customer is visited. Equations (72)–(76) define feasible flows from the plant to the customers with the capacitated vehicles. Equations (77) and (78) define the domain of the second-stage variables.

PRP–IR formulation. In PRP–IR, the firm has flexibility to route the vehicles after observing customer demand. We thus add a second-stage variable χ_{ijts} equal to 1 if a vehicle travels from node i to node j in period t in scenario s. We add a customer reservation cost g for each customer appointment in the first stage (otherwise, the problem becomes over-simplified as the decision-maker can simply make appointments with all customers in the first stage). We only define the second-stage auxiliary flow variables v_{ijts} and v_{jits}. The recourse function becomes $Q_s(\boldsymbol{y},\boldsymbol{z})$, and the problem is formulated as follows.

$$PRP\text{–}IR(\mathcal{S},\boldsymbol{h}) = \min \sum_{t\in\mathcal{T}}\left(fy_t + g\sum_{i\in\mathcal{N}_c} z_{it}\right) + \sum_{s\in\mathcal{S}} h_s Q_s(\boldsymbol{y},\boldsymbol{z}), \tag{79}$$
$$\text{s.t. Equation (63),}$$

where for each $s\in\mathcal{S}$,

$$Q_s(\boldsymbol{y},\boldsymbol{z}) = \min \sum_{t\in\mathcal{T}}\left(\sum_{(i,j)\in\overline{\mathcal{E}}} c_{ij}\chi_{ijts} + up_{ts} + \sum_{i\in\mathcal{N}}\rho_i I_{its} + \sum_{i\in\mathcal{N}_c}\sigma_i e_{its}\right) \tag{80}$$
$$\text{s.t. Equations (66)–(75), (77)–(78)}$$
$$v_{ijts} + v_{jits} = \overline{Q}\chi_{ijts} \quad \forall i,j\in\overline{\mathcal{E}},\ \forall t\in\mathcal{T} \tag{81}$$
$$\sum_{j\in\overline{\mathcal{N}},i<j}\chi_{ijts} + \sum_{j\in\overline{\mathcal{N}},i>j}\chi_{jits} = 2z_{it} \quad \forall i\in\mathcal{N}_c,\ \forall t\in\mathcal{T} \tag{82}$$
$$\chi_{ijts},\chi_{jits}\in\{0,1\} \quad \forall\{i,j\}\in\overline{\mathcal{E}},\ \forall t\in\mathcal{T}. \tag{83}$$

C.2. Facility location problems

Notations. We consider a facility location problem under demand uncertainty. Let $\mathcal{N}$ be the set of customers, $\mathcal{V}$ be the set of potential facility locations, and $\mathcal{S}$ be set of demand scenarios. Each scenario $s\in\mathcal{S}$ is associated with a probability h_s such that $\sum_{s\in\mathcal{S}} h_s = 1$. We denote the demand from customer $i\in\mathcal{N}$ in scenario $s\in\mathcal{S}$ by $d_i^s \geq 0$.

The first-stage problem involves determining whether to build each facility $j\in\mathcal{V}$. Let f_j denote the construction cost of facility $j\in\mathcal{V}$ and let u_j denote its capacity. The second-stage involves optimizing operations with the constructed facilities. We denote the unit transportation cost from facility $j\in\mathcal{V}$ to customer $i\in\mathcal{N}$ by c_{ij}. Each unit of unmet demand incurs a penalty cost σ_i.

We define the following decision variables

$$y_j = \begin{cases} 1 & \text{if facility } j\in\mathcal{V} \text{ is built} \\ 0 & \text{otherwise.}\end{cases}$$
$$x_{ij}^s = \text{amount shipped from facility } j\in\mathcal{V} \text{ to customer } i\in\mathcal{N} \text{ in scenario } s\in\mathcal{S}.$$
$$e_i^s = \text{amount of unmet demand from customer } i\in\mathcal{N} \text{ in scenario } s\in\mathcal{S}.$$

FLP–CR formulation. Equation (84a) minimizes the first-stage cost and the expected second-stage cost. Equation (84b) defines the domain of the first-stage variables. For each scenario, Equation (84c) minimizes the total cost of transportation and unmet demand. Equation (84d) defines the unmet demand as the total demand d_i^s minus the shipment amount received from the facilities. Equation (84e) limits the capacity u_j at

each constructed facility. Equation (84f) are valid inequalities to tighten the formulation. Equations (84g) and (84h) define the domain of the second-stage variables.

$$FLP\text{–}CR(\mathcal{S},\boldsymbol{h}) = \min \sum_{j\in\mathcal{V}} f_j y_j + \sum_{s\in\mathcal{S}} h_s Q_s(\boldsymbol{y}) \tag{84a}$$
$$\text{s.t. } y_j \in \{0,1\} \quad \forall j \in \mathcal{V}, \tag{84b}$$

where for each $s \in \mathcal{S}$,

$$Q_s(\boldsymbol{y}) = \min \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{V}} c_{ij}x^s_{ij} + \sum_{i\in\mathcal{N}} \sigma_i e^s_i \tag{84c}$$
$$\text{s.t. } \sum_{j\in\mathcal{V}} x^s_{ij} + e^s_i = d^s_i \quad \forall i \in \mathcal{N} \tag{84d}$$
$$\sum_{i\in\mathcal{N}} x^s_{ij} \le u_j y_j \quad \forall j \in \mathcal{V} \tag{84e}$$
$$x^s_{ij} \le d^s_i y_j \quad \forall i \in \mathcal{N}, j \in \mathcal{V} \tag{84f}$$
$$x^s_{ij} \ge 0 \quad \forall i \in \mathcal{N}, j \in \mathcal{V} \tag{84g}$$
$$e^s_i \ge 0 \quad \forall i \in \mathcal{N}. \tag{84h}$$

FLP–IR formulation. In some cases, constructing facilities is not the "final word". Instead, some facility location problems have binary second-stage variables in order to activate facilities (**??**). For instance, in the context of relief shelter location, facilities are first deployed at the strategic level. Following a natural disaster, each shelter must then be staffed and equipped prior to being ready for operations. In this instance, scenarios represent natural disasters, and shelter activation involves a cost g_j. Accordingly, we define the following additional second-stage binary variables:

$$z^s_j = \begin{cases} 1 & \text{if facility } j \in \mathcal{V} \text{ is active in scenario } s \in \mathcal{S} \\ 0 & \text{otherwise.} \end{cases}$$

The problem is then formulated as follows, where Equation (85g) imposes that a facility cannot be active unless a facility has been built.

$$FLP\text{–}IR(\mathcal{S},\boldsymbol{h}) = \min \sum_{j\in\mathcal{V}} f_j y_j + \sum_{s\in\mathcal{S}} h_s Q_s(\boldsymbol{y}) \tag{85a}$$
$$\text{s.t. } y_j \in \{0,1\} \quad \forall j \in \mathcal{V}, \tag{85b}$$

where for each $s \in \mathcal{S}$,

$$Q_s(\boldsymbol{y}) = \min \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{V}} c_{ij}x^s_{ij} + \sum_{i\in\mathcal{N}} \sigma_i e^s_i + \sum_{j\in\mathcal{V}} g_j z^s_j \tag{85c}$$
$$\text{s.t. } \sum_{j\in\mathcal{V}} x^s_{ij} + e^s_i = d^s_i \quad \forall i \in \mathcal{N} \tag{85d}$$
$$\sum_{i\in\mathcal{N}} x^s_{ij} \le u_j z^s_j \quad \forall j \in \mathcal{V} \tag{85e}$$
$$x^s_{ij} \le d^s_i z^s_j \quad \forall i \in \mathcal{N}, j \in \mathcal{V} \tag{85f}$$
$$z^s_j \le y_j \quad \forall j \in \mathcal{V} \tag{85g}$$
$$x^s_{ij} \ge 0 \quad \forall i \in \mathcal{N}, j \in \mathcal{V} \tag{85h}$$
$$e^s_i \ge 0 \quad \forall i \in \mathcal{N} \tag{85i}$$
$$z^s_j \in \{0,1\} \quad \forall j \in \mathcal{V}. \tag{85j}$$

Experimental setup. We sample customer and facility locations uniformly within a rectangle area with width X and height Y, defined as $\mathcal{R} = \{(\alpha, \beta) \in \mathbb{R}^2 : 0 \leq \alpha \leq X, 0 \leq \beta \leq Y\}$. We set $X = Y = 10$. In different scenarios, we sample a "horizontal" region $\mathcal{H} = \{(\alpha, \beta) \in \mathbb{R}^2 : x_1 \leq \alpha \leq x_2, 0 \leq \beta \leq Y\}$ $(x_1 \leq x_2)$ and a "vertical" region $\mathcal{V} = \{(\alpha, \beta) \in \mathbb{R}^2 : 0 \leq \alpha \leq X, y_1 \leq \beta \leq y_2\}$ $(y_1 \leq y_2)$. We then assume that demand is "high" (sampled uniformly between 30 and 40) in $\mathcal{H} \cap \mathcal{V}$ (orange in Figure 9); "medium" (between 20 and 30) in $\mathcal{H} \setminus (\mathcal{H} \cap \mathcal{V})$ (green); "low" (between 10 and 20) in $\mathcal{V} \setminus (\mathcal{H} \cap \mathcal{V})$ (blue); and zero in $\mathcal{R} \setminus (\mathcal{H} \cup \mathcal{V})$ (grey). We set the unit transportation cost c_{ij} as the Euclidean distance between facility $j \in \mathcal{V}$ to customer $i \in \mathcal{N}$ times a factor uniformly sampled between 0.8 and 1.2. We uniformly sample facility capacity u_j between 50 and 500. We uniformly sample the unmet penalty cost σ_i between 20 and 30. In FLP–CR, we set the construction cost $f_j = 100 + u_j\gamma_1$, where $\gamma_1 \sim \mathcal{U}([0.8, 1.2])$. In FLP–IR, we set the construction cost $f_j = 50 + u_j\gamma_2$ and the activation cost $g_j = 50 + u_j\gamma_2$, where $\gamma_2 \sim \mathcal{U}([0.4, 0.8])$.

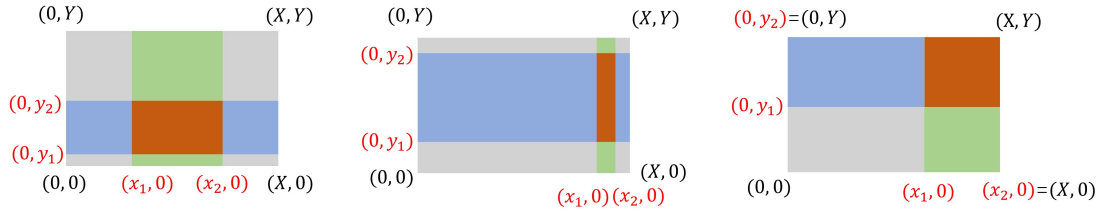


Figure 9 Examples of customer demand patterns for different scenarios in the facility location problems.