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

Wei Zhang, Alexandre Jacquillat, Kai Wang, Shuaian Wang

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 $Q^+(\boldsymbol{x}, \xi_s) = Q(\boldsymbol{x}, \xi_s) + R^C$:

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

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 $\boldsymbol{x} \in \mathcal{X}^P \setminus \mathcal{X}_0^P$:

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

It comes:

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

$$
\begin{aligned}
&= \sum_{\boldsymbol{x}\in\mathcal{X}^P\setminus\mathcal{X}_0^P} \left| \sum_{s\in\mathcal{S}} h_s Q^+(\boldsymbol{x},\xi_s) - \sum_{s\neq s'} w_s Q^+(\boldsymbol{x},\xi_s) - U_{s'} Q^+(\boldsymbol{x},\xi_{s'}) \right| \\
&\quad + \sum_{\boldsymbol{x}\in\mathcal{X}_0^P} \left| \sum_{s\in\mathcal{S}} h_s Q^+(\boldsymbol{x},\xi_s) - \sum_{s\neq s'} w_s Q^+(\boldsymbol{x},\xi_s) - U_{s'} Q^+(\boldsymbol{x},\xi_{s'}) \right| \\
&= \sum_{\boldsymbol{x}\in\mathcal{X}^P} \left| \sum_{s\in\mathcal{S}} h_s Q^+(\boldsymbol{x},\xi_s) - \sum_{s\in\mathcal{S}} w'_s Q^+(\boldsymbol{x},\xi_s) \right|.
\end{aligned}
$$

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:

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

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:

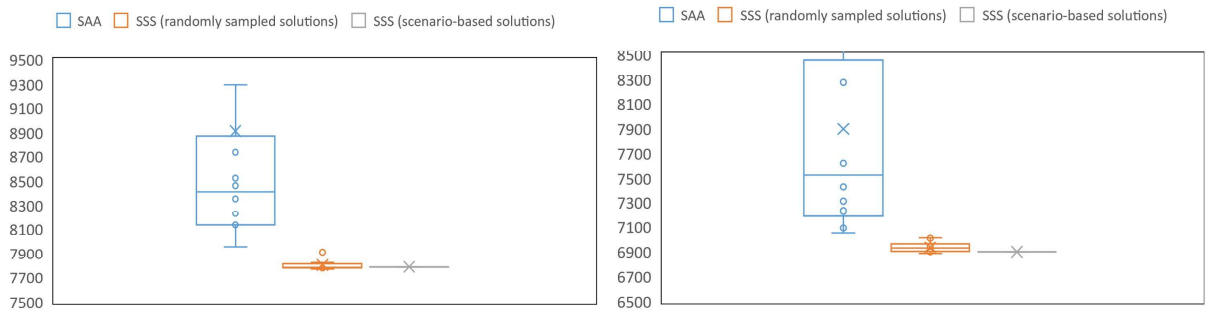
$$\Lambda \leq 2\left[\Delta + \left(1 + K\overline{U}\right)\left(\Delta^P + L\Delta^A\right)\right]. \tag{43}$$

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 (Wallace 2000). 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 Narum (2020), 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.



(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.

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: 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.

1: **procedure** SSS HEURISTIC($\mathcal{X}^P, K, \mathcal{K}_0, \Omega$) ▷ One thread with an initialization
2: $\tau \leftarrow 0$ ▷ Iteration index
3: **while** $\mathcal{K}_\tau \neq \mathcal{K}_{\tau-1}$ **do** ▷ Check if the solution is improved
4: **while** $\mathcal{K}_\tau \neq \mathcal{K}_{\tau-1}$ **do** ▷ LSO I: Lines 4–9
5: $(\widehat{s}_1, \cdots, \widehat{s}_K) \leftarrow \mathcal{K}_\tau$ ▷ Update the neighborhood centers
6: $(\mathcal{S}_1, \cdots, \mathcal{S}_K) \leftarrow SNM(\mathcal{X}^P, \widehat{s}_1, \cdots, \widehat{s}_K, \Omega)$ ▷ Re-define scenario neighborhoods
7: $\mathcal{K}_{\tau+1} \leftarrow NSSS(\mathcal{X}^P, \mathcal{S}_1, \cdots, \mathcal{S}_K)$ ▷ Conduct local search
8: $\tau \leftarrow \tau + 1$
9: **end while**
10: **for** $k = 1, \cdots, K$ **do** ▷ LSO II: Lines 10–16
11: $(\widehat{s}_1, \cdots, \widehat{s}_K) \leftarrow \mathcal{K}_\tau$ ▷ Update the incumbent solution
12: $\mathcal{K}_{\tau+1} \leftarrow PSSS(\mathcal{X}^P, k, \widehat{s}_1, \cdots, \widehat{s}_K)$ ▷ Conduct local search
13: **if** $\mathcal{K}_{\tau+1} \neq \mathcal{K}_\tau$ **then**
14: **break** ▷ Stops LSO II if the solution is improved
15: **end if**
16: **end for**
17: $\tau \leftarrow \tau + 1$ and $\widetilde{\mathcal{K}} \leftarrow \mathcal{K}_\tau$ ▷ Record the incumbent solution
18: **end while**
19: **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.

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

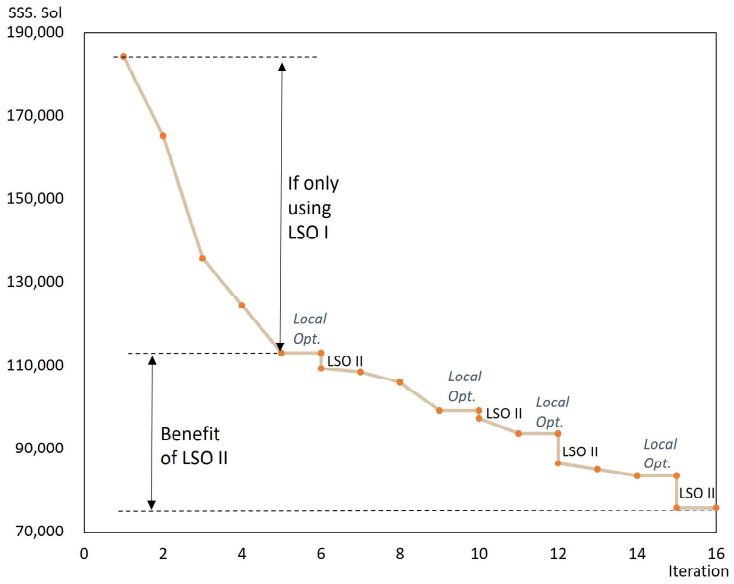


Figure 8 **Benefit of LSO II (PRP–CR problem, $K = 20$, $|\mathcal{X}^P| = 60$).**

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.

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

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 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 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 Ryan et al. (2020). 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 Ryan et al. (2020), 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:*

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

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

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

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

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

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

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

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:

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

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

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

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:

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

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, $Z_1\le Z_2$.

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 $|\mathcal{B}|\ge|\mathcal{B}^*|$) 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, $Z_1\ge Z_2$. □

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:

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

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 $\widetilde{\boldsymbol{c}}\ge\boldsymbol{c}$. 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 Adulyasak et al. (2015), 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 $\overline{v}_{ijt}$ equal to the vehicle load on edge $\{i,j\} \in \overline{\mathcal{E}}$, and $\overline{v}_{jit}$ equal to the empty space on the vehicle on edge $\{i,j\} \in \overline{\mathcal{E}}$ i.e., $\overline{v}_{jit} = \overline{Q} - \overline{v}_{ijt}$ (see Baldacci et al. (2004) 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.

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

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

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

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

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

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

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

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

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

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

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.

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

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

$$y_j = \begin{cases} 1 & \text{if facility } j \in \mathcal{V} \text{ is built} \\ 0 & \text{otherwise.} \end{cases}$$

$x_{ij}^s =$ amount shipped from facility $j \in \mathcal{V}$ to customer $i \in \mathcal{N}$ in scenario $s \in \mathcal{S}$.

$e_i^s =$ amount of unmet demand from customer $i \in \mathcal{N}$ 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 $s \in \mathcal{S}$,

$$Q_s(\boldsymbol{y}) = \min \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{V}} c_{ij} x_{ij}^s + \sum_{i \in \mathcal{N}} \sigma_i e_i^s \tag{84c}$$

$$\text{s.t. } \sum_{j \in \mathcal{V}} x_{ij}^s + e_i^s = d_i^s \quad \forall i \in \mathcal{N} \tag{84d}$$

$$\sum_{i \in \mathcal{N}} x_{ij}^s \leq u_j y_j \quad \forall j \in \mathcal{V} \tag{84e}$$

$$x_{ij}^s \leq d_i^s y_j \quad \forall i \in \mathcal{N}, j \in \mathcal{V} \tag{84f}$$

$$x_{ij}^s \geq 0 \quad \forall i \in \mathcal{N}, j \in \mathcal{V} \tag{84g}$$

$$e_i^s \geq 0 \quad \forall i \in \mathcal{N}. \tag{84h}$$

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 (Smith and Penuel 2008, Penuel et al. 2010). 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:

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

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 $\mathcal{R} = \{(\alpha,\beta)\in\mathbb{R}^2 : 0\le\alpha\le X, 0\le\beta\le Y\}$. We set $X = Y = 10$. In different scenarios, we sample a "horizontal" region $\mathcal{H} = \{(\alpha,\beta)\in\mathbb{R}^2 : x_1\le\alpha\le x_2, 0\le\beta\le Y\}$ $(x_1\le x_2)$ and a "vertical" region $\mathcal{V} = \{(\alpha,\beta)\in\mathbb{R}^2 : 0\le\alpha\le X, y_1\le\beta\le y_2\}$ $(y_1\le y_2)$. 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 $\gamma_2\sim\mathcal{U}([0.4,0.8])$.

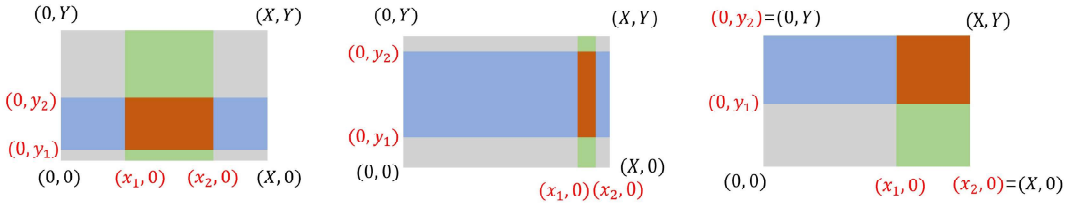


Figure 9 **Examples of customer demand patterns for different scenarios in the facility location problems.**