K-adaptability for two-stage stochastic optimization

Federica Donnini[*], Marianna De Santis[†], Jannis Kurtz[‡]

Abstract

Two-stage stochastic programs are used to model problems with uncertain data, where a decision maker first decides the values of first-stage decision variables, then observes the values of the uncertain data, and finally decides the values of the second-stage decisions. We focus on the case where uncertainty can be modeled by a finite number of scenarios. A popular approach to two-stage problems under uncertainty is K-adaptability, where a small set of K second-stage solutions is already calculated in the first stage. While this approach was studied in several works for two-stage robust problems, in terms of two-stage stochastic optimization it did not get the same attention. We study the K-adaptability approach for two-stage stochastic optimization for a general class of objective and constraint functions, where the uncertainty can appear in both. We present approximation bounds the K-adaptability approach provides for the exact two-stage stochastic problem and derive conditions under which it is optimal. We develop an exact partition-based branch-and-bound procedure which is able to solve the stochastic K-adaptability exactly and we provide experimental results for different problem structures, including linear and quadratic knapsack and facility location problems.

Key Words: Two-stage stochastic programming, K-adaptability, Branch-and-bound.

1 Introduction

Optimization under uncertainty is one of the leading paradigms frequently used in mathematical optimization problems. When modeling a real-world optimization problem usually several of the model parameters are uncertain. Frequently appearing uncertainties are future customer demands, transportation costs or traffic situations, but also measuring and rounding errors. Incorporating these uncertainties in an optimization model is crucial, since ignoring them can lead to infeasible or sub-optimal solutions in future situations. One of the popular frameworks in optimization under uncertainty is stochastic optimization, where it is assumed that the uncertain parameters follow a probability distribution. The goal is to find solutions which optimize a given risk measure as the expected value or the Value at Risk (Birge and Louveaux 1997, Shapiro et al. 2021). A large number of works consider two-stage stochastic optimization problems which are tailored for many societal applications involving a two-stage structure, i.e., some of the decisions have to be made *here-and-now* (also called first-stage decisions) and some of the decisions can be made after the uncertain parameters are known (also called *wait-and-see* or second-stage decisions). A classical example for a two-stage problem is the facility location problem, where in the first stage we have to decide in which locations we open our facilities before the demand of the customers is known. After the demand is revealed, we have to assign each customer to our facilities, which can be usually done on-the-fly. Two-stage stochastic optimization problems are inherently hard to solve under general probability distributions. However, in the case we have access to a finite number of scenarios drawn from the (unknown) distribution, the so-called *sample average approximation* (SAA) can be applied, where the empirical expectation over the scenario set is optimized, which provides a good approximation of the true expected value if the number of scenarios is large enough (Chen et al. 2019). A well-known reformulation of the problem is the so-called *extensive formulation*, where for each scenario a copy of the second-stage variables is introduced, which leads to a large-dimensional mixed-integer optimization problem. In addition to commonly discussed drawbacks, such as its high computational demand and its

[*]Information Engineering Department, University of Florence, Via di Santa Marta 3, 50139, Firenze, Italy, (`federica.donnini@unifi.it`)

[†]Information Engineering Department, University of Florence, Via di Santa Marta 3, 50139, Firenze, Italy, (`marianna.desantis@unifi.it`)

[‡]Faculty of Economics and Business, University of Amsterdam, Postbus 15953 1001 NL Amsterdam, Netherlands (`j.kurtz@uva.nl`)

potential for overfitting (Kuhn et al. 2019, Van Parys et al. 2021), the SAA (and two-stage stochastic optimization in general) has another property which is undesirable in many applications: the number of different second-stage policies can be extremely large. In fact, it can happen that for each scenario a different second-stage policy is optimal and may have to be implemented by the decision maker. While this can be less of an issue in certain applications, it can be a large burden for applications e.g. in radiotherapy or disaster management. In proton radiation therapy a small amount of pre-calculated treatment plans has to be prepared, since each of the plans has to be adjusted and accepted by a doctor, which involves time-consuming human labor (Qiu et al. 2025). Another example appears in disaster management, where a small number of emergency response plans is preferred to be known in advance to react quickly in case of a disaster (Weller and Oliveira 2025). A small number of emergency plans has the additional advantage that the emergency teams can be trained for each possible plan in advance. In these situations fully-adaptive models, as classical two-stage stochastic optimization, are not suitable. A popular approach which has the potential to tackle the latter issue is the K-adaptability approach, which was first studied for two-stage robust optimization problems (Bertsimas and Caramanis 2010, Subramanyam et al. 2020, Hanasusanto et al. 2015, Buchheim and Kurtz 2017, Kurtz 2024a) and later applied to stochastic optimization problems without first-stage variables (Buchheim and Pruente 2019, Malaguti et al. 2022, 2025). In case of a two-stage problem the idea of the approach is to calculate up to K second-stage policies already in the first stage and then proceed to choose the best of these solutions in the second stage, once the exact values of the uncertain parameters are known. The K-adaptability approach can be interpreted as an approximation of the fully adaptable two-stage stochastic problem, where the number K of second-stage policies can be used to control the approximation error. In practice a good tradeoff between a small number of policies and a good objective value is desired. As it was shown experimentally in Qiu et al. (2025) already a small number of solutions ($K \sim 5$) for the robust K-adaptable radiation therapy treatment planning is enough to achieve objective values close to the fully adaptable version. While the K-adaptability problem was studied extensively for robust two-stage problems, the amount of works for stochastic optimization problems is limited. The only works regarding the stochastic setup consider the case where no first-stage variables appear and the objective functions are linear (Buchheim and Pruente 2019, Malaguti et al. 2022, 2025). Furthermore, in the stochastic setting, nothing is known about the approximation guarantee the K-adaptability approach provides, depending on the number K. Such bounds are important to guide the decision maker in choosing the right K.
In this work we study the K-adaptability approach for two-stage stochastic optimization problems for a general class of (non-linear) objective and constraint functions where the decisions are allowed to be mixed-integer. The uncertainty appears both in the objective and the constraints.

Contributions

- We apply for the first time the K-adaptability approach to two-stage stochastic optimization problems with linear or non-linear objective and constraint functions.
- We derive approximation guarantees the K-adaptability approach achieves for a wide range of objective and constraint functions. Additionally, we find conditions on the number K for which the approach is equivalent to the exact two-stage stochastic problem.
- We derive a partition-based branch-and-bound procedure, which solves the K-adaptability problem exactly. The procedure can be applied to a wide range of (non-linear) problem structures.
- We test our algorithm on different types of problem classes, including two-stage stochastic versions of the linear and quadratic knapsack and facility location problem.

Outline The remainder of the paper is organized as follows. In the next section, we introduce a formulation of the K-adaptability that is based on partitions of the scenario set, as well as the notation used throughout the paper. In Section 3, we analyze conditions under which the K-adaptability problem is equivalent to the original two-stage problem and provide bounds on the approximation error depending on K. In Section 4 we present our branch-and-bound method. We describe how to compute valid lower and upper bounds for the K-adaptability problem through appropriately defined subproblems. The method relies on subpartitions of the scenario set, and we detail the corresponding branching and pruning rules that ensure both efficiency and correctness. An extensive computational study on linear and quadratic instances is presented in Section 5. Finally, Section 6 concludes the paper with a summary of findings and perspectives for future research.

1.1 Related Literature

The K-adaptability approach was first introduced in Bertsimas and Caramanis (2010) for two-stage robust optimization problems and can be considered as a decision rule approach for the case where the second-stage decisions are integer. This approach was later studied for binary and mixed-integer two-stage robust problems (Hanasusanto et al. 2015, Subramanyam et al. 2020). In Subramanyam et al. (2020) the authors derive a partition-based branch-and-bound method which follows a similar idea as the method we present in this work. A machine-learning supported version of the algorithm was studied in Julien et al. (2025). Another exact algorithm for the problem with only objective uncertainty was developed in Ghahtarani et al. (2023) by using a logic-based Benders' decomposition method. In Kurtz (2024a) for the case of objective uncertainty algorithms with provable approximation bounds were derived and later in Kurtz (2024b) bounds on the required number K of second-stage solutions were derived to recover the optimal two-stage robust solution from the K-adaptability approach for both, objective and constraint uncertainty. The K-adaptability approach was applied to robust container vessel sequencing problems (Zheng and Sui 2019), and disaster management (Weller and Oliveira 2025). A similar line of research studies the robust K-adaptability approach for the case where only a single stage exists. In this case the idea is to calculate K feasible solutions (instead of one) which are optimal in the worst-case where for each uncertain scenario the best of the calculated solutions can be chosen. This approach is also often called *min-max-min robust optimization*. It was first studied in Buchheim and Kurtz (2017) and was later studied in several other works for convex uncertainty sets (Chassein et al. 2019, Chassein and Goerigk 2021, Arslan et al. 2022), discrete uncertainty sets (Buchheim and Kurtz 2018, Goerigk et al. 2020) and for the capacitated vehicle routing problem (Eufinger et al. 2020). Recently the min-max-min robust problem was applied to treatment planning in proton radiotherapy (Qiu et al. 2025).
In the realm of stochastic optimization, where the uncertain parameters are assumed to follow a probability distribution, the K-adaptability approach was only studied in a limited number of works. In Buchheim and Pruente (2019), Malaguti et al. (2022, 2025) the idea of min-max-min robust optimization was adapted to the stochastic setting with linear objective functions and discrete probability distributions. The authors derive complexity results, showing that the problem is NP-hard and analyze its parametrized complexity. In Malaguti et al. (2022) an exact branch-and-prize method is presented. While all the latter works consider the one-stage setting, to the best of our knowledge there are no works which study the problem in the two-stage setting.
The K-adaptability approach was also applied to distributionally robust optimization in Hanasusanto et al. (2016), Han et al. (2023).

2 Preliminaries

2.1 Problem Definitions

In this paper we consider two-stage stochastic programs of the form

$$\begin{aligned} &\min\ f(x) + \mathbb{E}_\omega[\min\{g(y,\xi(\omega))\ :\ y \in Y(x,\xi(\omega))\}] \\ &\text{s.t. }\ x \in X, \end{aligned} \tag{1}$$

where $X \subseteq \mathbb{R}^{n_x}$, $Y(x,\xi(\omega)) \subseteq Y \subseteq \mathbb{R}^{n_y}$ are bounded sets which may include integrality constraints on the variables x and y. The function $f : \mathbb{R}^{n_x} \to \mathbb{R}$ is the first-stage and $g(\cdot,\xi(\omega)) : \mathbb{R}^{n_y} \to \mathbb{R}$ is the second-stage objective function. The vector $\omega \in \Omega$, is a random variable with domain Ω and $\xi : \Omega \to \mathbb{R}^{n_\xi}$ are the uncertain parameters of the problem, also called *scenarios*. Observe that we consider the most general case in which uncertainty can affect both the objective function and the feasible set, and functions f and g are not necessarily linear. We will study the case in which ω follows a finite discrete probability distribution, i.e., we can model the uncertainty through a finite number of ℓ scenarios. More precisely, ω attains only values in $\{1,\dots,\ell\}$ and $p_s \in [0,1]$ is the probability of outcome s, such that $\sum_{s=1}^{\ell} p_s = 1$.

For ease of notation we define $g_s(y) := g(y, \xi(s))$ and $Y_s(x) := Y(x, \xi(s))$ for all x, y and $s \in \{1, \dots, \ell\}$. Then, we can write the *extensive formulation* (Birge and Louveaux 1997) of problem (1) as

$$\begin{aligned} \min\ & f(x) + \sum_{s=1}^{\ell} p_s g_s(y_s) \\ \text{s.t. }\ & x \in X, \\ & y_s \in Y_s(x), \quad s = 1, \dots, \ell, \end{aligned} \tag{2}$$

where second-stage variables are explicitly described for each scenario $s \in \{1, \dots, \ell\}$ by the corresponding vector y_s.

The problem we study in this work is the K-adaptable version of the two-stage stochastic optimization problem described above. The idea of the approach is to calculate up to K second-stage policies already in the first stage and then proceed to choose the best of these solutions in the second stage, once the exact values of the uncertain parameters are known. Applied to problem (2) the K-adaptability problem is defined as

$$\begin{aligned} \min\ & f(x) + \sum_{s=1}^{\ell} p_s \min\{g_s(y) : y \in Y_s(x) \cap \{y^1, \dots, y^K\}\} \\ \text{s.t. }\ & x \in X, \\ & y^1, \dots, y^K \in Y. \end{aligned} \tag{k-2SSP}$$

Note, that the inner minimum takes the value ∞ if $Y_s(x) \cap \{y^1, \dots, y^K\}$ is the empty set. We call an instance of the previous problem feasible if $x \in X$ and $y^1, \dots, y^K \in Y$ exist such that, for each scenario s, one of the selected second-stage solutions $y^1, \dots, y^K$ is feasible, i.e., it is contained in $Y_s(x)$.

2.2 Partition-Based Reformulation

Problem (k-2SSP) can be reformulated as an optimization problem over the partitions of the set of scenarios $S = \{1, \dots, \ell\}$. The proposed method relies on this reformulation and operates by solving subproblems defined over *subpartitions* of S. Both concepts are formalized in the following definitions.

Definition 1. *$\mathcal{P} = (P_1, \dots, P_K)$ is a* partition *of S of order $M \le K$, if*

- *$\cup_{j=1}^{K} P_j = S$;*
- *$P_j \cap P_i = \emptyset, \forall\ j \neq i,\ i, j \in \{1, \dots, K\}$;*
- *$M = |\{j \in \{1, \dots, K\} | P_j \neq \emptyset\}|$.*

A *subpartition* is a collection of disjoint subsets that may not cover the whole scenario set S:

Definition 2. *$\mathcal{R} = (R_1, \dots, R_K)$ is a* subpartition *of S of order $M \le K$, if*

- *$\cup_{j=1}^{K} R_j \subseteq S$;*
- *$R_j \cap R_i = \emptyset,\ \forall\ j \neq i\ i, j \in \{1, \dots, K\}$;*
- *$M = |\{j \in \{1, \dots, K\} | R_j \neq \emptyset\}|$.*

In the remainder of the paper, we adopt the following notation. Given $\mathcal{P} = (P_1, \dots, P_K)$ and $\mathcal{R} = (R_1, \dots, R_K)$, a partition and subpartition of S, respectively, we say:

1. $\mathcal{R} \subseteq \mathcal{P}$ if $R_j \subseteq P_j,\ \forall j = 1, \dots, K$,
2. $s \in \mathcal{R}$ (or $s \in \mathcal{P}$) if $s \in S$ and there exists $j \in \{1, \dots, K\}$ such that $s \in R_j$ (or $s \in P_j$). Similarly, we write $s \notin \mathcal{R}$, if $s \in S$ and $\nexists\ j \in \{1, \dots, K\}$ s.t. $s \in R_j$.

Given $\mathcal{R}^1 = (R_1^1, \ldots, R_K^1)$ and $\mathcal{R}^2 = (R_1^2, \ldots, R_K^2)$ two subpartitions of S, we say that $\mathcal{R}^1 \neq \mathcal{R}^2$ if $\exists\, s \in S$ and $j \in \{1, \ldots, K\}$ such that $s \in R_j^1$ and $s \notin R_j^2$.
Given a partition $\mathcal{P}$, we consider the corresponding *partition induced problem*:

$$\begin{aligned} pip(\mathcal{P}) := \min\ & f(x) + \sum_{j=1}^{K} \sum_{s \in P_j} p_s g_s(y^j) \\ & x \in X, \\ & y^j \in Y_{P_j}(x) \quad j = 1, \ldots, K, \end{aligned} \tag{$pip(\mathcal{P})$}$$

where for each $j = 1, \ldots, K$ we define

$$Y_{P_j}(x) = \begin{cases} \mathbb{R}^{n_y} & \text{if } P_j = \emptyset, \\ \cap_{s \in P_j} Y_s(x) & \text{otherwise.} \end{cases}$$

Let us observe that ($pip(\mathcal{P})$) is feasible only if $x \in X$ exists such that $Y_{P_j}(x) \neq \emptyset$, $\forall\ j = 1, \ldots, K$. If ($pip(\mathcal{P})$) is feasible, then let $(x_P, y_P^1, \ldots, y_P^K)$ (or $(x_P, \{y_P^j\}_{j=1}^K)$) be its optimal solution, which we call *solution induced by partition* $\mathcal{P}$. Each y_P^j is feasible for all the scenarios s that belong to the set P_j, for every $j \in \{1, \ldots, K\}$, i.e. we find a set of second-stage policies that cover the entire scenario set. As stated in Buchheim and Pruente (2019), an exact approach to obtain the optimal K-adaptability solution consists in enumerating all possible partitions $\mathcal{P}$ of the scenario set S, solving the corresponding problem ($pip(\mathcal{P})$) whenever it is feasible, and selecting the one with the smallest objective value. Then, the K-adaptability problem (k-2SSP) can be formulated as

$$\min\{pip(\mathcal{P}) \mid \mathcal{P} \text{ is a partition of } S \text{ of order } K\}. \tag{3}$$

Note that the number of partitions to be considered corresponds to the Stirling number of the second kind, which counts the number of distinct ways to divide a set of ℓ elements into K non-empty subsets, and is denoted by $\mathcal{S}_{\ell,K}$. This number grows rapidly with both K and ℓ but decreases again as K approaches ℓ (Buchheim and Pruente 2019), making the exhaustive enumeration of all partitions computationally inefficient. In Section 4, we present our method, which uses a branch-and-bound procedure to compute the optimal partition for (3) while avoiding a complete enumeration of all possible partitions.

3 Approximation Guarantees

In this section, we discuss the question of how well the K-adaptability problem approximates the original two-stage problem (2) and derive conditions under which both problems are equivalent. We define $\text{opt}(k)$ to be the optimal value of (k-2SSP) with $K = k$.
First, observe that by definition of the K-adaptability problem for $1 \leq k \leq \ell$ the following chain of inequalities holds:

$$\text{opt}(\ell) \leq \text{opt}(\ell - 1) \leq \ldots \leq \text{opt}(k) \leq \ldots \leq \text{opt}(1). \tag{4}$$

In particular, (k-2SSP) with $K = \ell$ is equivalent to (2).

Theorem 1. *Let g_s be Lipschitz continuous on Y with Lipschitz constant L_s and $L := \max_{s=1,\ldots,\ell} L_s$. Assume that an optimal solution $(x^*, \{y^*(s)\}_{s=1}^{\ell})$ of (k-2SSP) with $K = \ell$ exists such that $\bigcap_{s=1}^{\ell} Y_s(x^*) \neq \emptyset$. Then, for every integer k with $1 \leq k \leq \ell$ it holds*

$$\text{opt}(k) - \text{opt}(\ell) \leq \frac{\ell - k + 1}{\ell} L\text{diam}(Y)$$

where $\text{diam}(Y) := \max_{y,y' \in Y} \|y - y'\| < \infty$.

Proof. W.l.o.g. we assume $p_1 \geq p_2 \geq \ldots \geq p_\ell$. Let $(x^*, \{y^*(s)\}_{s=1}^{\ell})$ be an optimal solution of (k-2SSP) with $K = \ell$ such that $\cap_{s=1}^{\ell} Y_s(x^*) \neq \emptyset$, which exists by assumption. Then it holds

$$\text{opt}(\ell) = f(x^*) + \sum_{s=1}^{\ell} p_s g_s(y^*(s)). \tag{5}$$

We construct a feasible solution $(\bar{x}, \bar{y}^1, \dots, \bar{y}^k)$ of (k-2SSP) with $K = k$ as follows: we set $\bar{x} = x^*$, $\bar{y}^s = y^*(s)$ for $s = 1, \dots, k-1$ and

$$\bar{y}^k \in \bigcap_{s=k}^{\ell} Y_s(x^*).$$

Note that by assumption the latter solution $\bar{y}^k$ exists. Then it holds

$$\begin{aligned}
\text{opt}(k) - \text{opt}(\ell) &= \text{opt}(k) - \left(f(x^*) + \sum_{s=1}^{\ell} p_s g_s(y^*(s)) \right) \\
&\le f(x^*) + \sum_{s=1}^{k-1} p_s g_s(\bar{y}^s) + \sum_{s=k}^{\ell} p_s g_s(\bar{y}^k) - \left(f(x^*) + \sum_{s=1}^{\ell} p_s g_s(y^*(s)) \right) \\
&= \sum_{s=k}^{\ell} p_s \left(g_s(\bar{y}^k) - g_s(y^*(s)) \right) \\
&\le \sum_{s=k}^{\ell} p_s L_s \| \bar{y}^k - y^*(s) \| \\
&\le L \text{diam}(Y) \sum_{s=k}^{\ell} p_s
\end{aligned}$$

where the first equality follows from (5), the first inequality follows since $(\bar{x}, \bar{y}^1, \dots, \bar{y}^k)$ is a feasible solution for (k-2SSP), the second equality follows from $\bar{y}^s = y^*(s)$ for $s = 1, \dots, k-1$ and canceling out all possible terms, the second inequality follows from the Lipschitz continuity of g_s and the last inequality follows from the definition of L and the diameter. From $p_1 \ge p_2 \ge \dots \ge p_\ell$ it follows

$$\sum_{s=k}^{\ell} p_s = 1 - \sum_{s=1}^{k-1} p_s \le 1 - \sum_{s=1}^{k-1} \frac{1}{\ell} = \frac{\ell - k + 1}{\ell}$$

where the inequality follows since the minimum of $\sum_{s=1}^{k-1} p_s$ under the ordering $p_1 \ge p_2 \ge \dots \ge p_\ell$ is attained for $p_s = \frac{1}{\ell}$ for all $s = 1, \dots, k-1$. This proves the result. $\square$

The latter result shows that the additive approximation error we make when calculating the k-adaptable problem, instead of the fully adaptable problem (2), behaves as $\frac{\ell-k+1}{\ell}$ up to a constant factor. Observe that for $k = \ell$ the bound of the previous theorem is not tight, since it is trivially zero.

Remark 1. *Note that in case $Y = \{0,1\}^{n_y}$ and for linear objective functions $g_s(y) = q_s^\top y$ we have* $\text{diam}(Y) = \sqrt{n_y}$ *and $L_s = \|q_s\|$ and hence $L = \max_{s=1,\dots,\ell} \|q_s\|$.*

The following example shows that the approximation guarantee derived in the latter theorem is tight for $1 \le k < \ell$.

Example 1. *Consider the problem*

$$\begin{aligned}
\min \ & \sum_{s=1}^{\ell} p_s q_s^\top y_s \\
\text{s.t. } & y_s \in Y \quad \forall s = 1, \dots, \ell
\end{aligned} \tag{6}$$

where $Y = \{y \in \{0,1\}^{n_y} : \sum_{i=1}^{n_y} y_i = 1\}$, $\ell = n_y$, and $(q_s)_i = -M < 0$ if $i = s$ and $(q_s)_i = 0$ otherwise, for all $s = 1, \dots, \ell$. We have $\text{diam}(Y) = \sqrt{2}$ *and $g_s(y) := q_s^\top y$ is Lipschitz continuous with Lipschitz constant $L := \max_{s=1,\dots,\ell} \|q_s\| = M$. Assume that $p_1 = p_2 = \dots = p_\ell = \frac{1}{\ell}$. We denote the optimal solution in Y for scenario s as $y^*(s)$ which has values $y^*(s)_i = 1$ if $i = s$ and $y^*(s)_i = 0$ otherwise, i.e. $y^*(s) = e_s$, where e_s is the s-th unit vector. The corresponding optimal value is $q_s^\top y^*(s) = -M$, while for every other solution $y \in Y \setminus \{y^*(s)\}$ it holds $q_s^\top y = 0$. Then, we have* $\text{opt}(\ell) = \sum_{s=1}^{\ell} p_s(-M) = -M$.

An optimal solution for the k-adaptability version is given by any subset of k distinct solutions $y^*(s)$, in particular we consider $\bar{y}^s = e_s$, for $s = 1, \dots, k$. Then it holds for all $1 \le k < \ell$

$$\operatorname{opt}(k) - \operatorname{opt}(\ell) = \sum_{s=1}^{k} p_s(-M) + M = \frac{\ell - k}{\ell} M \ge \frac{1}{2\sqrt{2}} \frac{\ell - k + 1}{\ell} \operatorname{diam}(Y) L.$$

Together with Theorem 1 we have

$$\frac{1}{2\sqrt{2}} \frac{\ell - k + 1}{\ell} \operatorname{diam}(Y) L \le \operatorname{opt}(k) - \operatorname{opt}(\ell) \le \frac{\ell - k + 1}{\ell} \operatorname{diam}(Y) L.$$

In the following we show that under suitable conditions, the optimal value for (2) can be already achieved for $k < \ell$. If the uncertainty only appears in the constraints, we call a subset of scenarios $D \subseteq \{1, \dots, \ell\}$ a recourse stable region for $x \in X$ if for every $y \in Y(x)$, the solution y is either feasible for all scenarios in D or not feasible for all scenarios in D. This definition was first introduced in Kurtz (2024b) for robust k-adaptable problems.

Lemma 1. *Assume that uncertainty only appears in the constraints, i.e., $g := g_s = g_{s'}$ for all $s, s' \in \{1, \dots, \ell\}$. Furthermore, assume that an optimal solution $(x^*, \{y^*(s)\}_{s=1}^{\ell})$ of (k-2SSP) for $K = \ell$ exists such that for x^* there exists a partition of the scenario set in at most T, $T \le \ell$, recourse stable regions $D_1, \dots, D_T$, i.e., $D_1 \cup \ldots \cup D_T = \{1, \dots, \ell\}$ and $D_i \cap D_j = \emptyset$ for all $i \ne j$. Then it holds*

$$\operatorname{opt}(T) = \operatorname{opt}(\ell).$$

Proof. The inequality $\operatorname{opt}(\ell) \le \operatorname{opt}(T)$ follows from (4), since $T \le \ell$. We now prove that $\operatorname{opt}(T) \le \operatorname{opt}(\ell)$. We proceed by showing that there exists an optimal solution of (k-2SSP) with $K = \ell$ which is feasible for (k-2SSP) with $K = T$. Let $(x^*, y^*(1), \dots, y^*(\ell))$ be an optimal solution of (k-2SSP) with $K = \ell$ such that for x^* there exists a partition of S into T recourse stable regions $D_1, \dots, D_T$, which exists by assumption. Consider one recourse stable region D_i, then we have

$$\sum_{s \in D_i} p_s g(y^*(s)) \ge \sum_{s \in D_i} p_s g(y^*_{D_i})$$

where

$$y^*_{D_i} \in \operatorname*{arg\,min}_{y^* \in \{y^*(s) : s \in D_i\}} g(y^*).$$

Note that due to the recourse-stability $y^*_{D_i} \in Y_s(x^*)$ for all $s \in D_i$. Applying the latter inequality to every recourse stable region we obtain

$$\begin{aligned}
\operatorname{opt}(\ell) &= f(x^*) + \sum_{s=1}^{\ell} p_s g(y^*(s)) \\
&= f(x^*) + \sum_{i=1}^{T} \sum_{s \in D_i} p_s g(y^*(s)) \\
&\ge f(x^*) + \sum_{i=1}^{T} \sum_{s \in D_i} p_s g(y^*_{D_i})
\end{aligned}$$

and hence the solution $(x^*, y^*_{D_1}, \dots, y^*_{D_T})$ has objective value at most $\operatorname{opt}(\ell)$. Especially, the latter solution is feasible for (k-2SSP) with $K = T$, which proves the result. □

The following example shows that the latter reduction from $K = \ell$ to $K = T$ can be significant.

Example 2. *Consider the problem without objective uncertainty, i.e., $g := g_s$ for all $s \in \{1, \dots, \ell\}$ and the set of all binary scenarios*

$$S_\xi := \{0, 1\}^n.$$

Then S_ξ contains 2^n scenarios, i.e. $\ell = 2^n$. Consider the feasible regions

$$Y_\xi = \left\{ y \in \{0, 1\}^n : \sum_{i=1}^{n} y_i = \sum_{i=1}^{n} \xi_i \right\}.$$

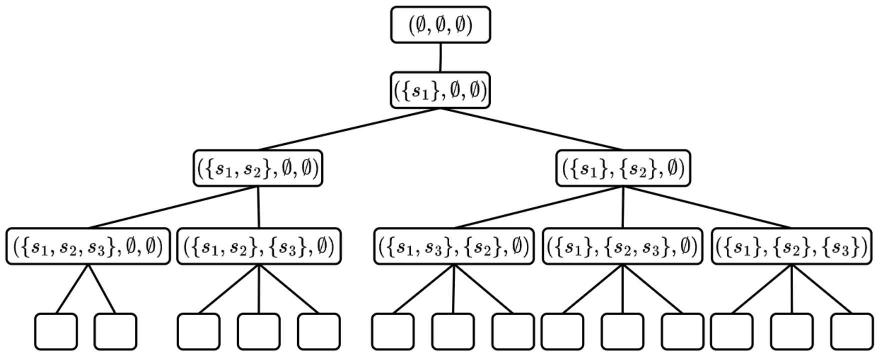
Then we can partition the set of scenarios into $n+1$ recourse stable regions given as

$$D_i := \{\xi \in \{0,1\}^n : \sum_{i=1}^{n} \xi_i = i\} \quad i = 0, \ldots, n.$$

Clearly, the union of all D_i covers the full set of scenarios S_ξ. Furthermore, by definition of Y_ξ, if a solution y is feasible for Y_ξ with $\xi \in D_i$, then it must be feasible for $Y_{\xi'}$ for all $\xi' \in D_i$. It follows from Lemma 1 that instead of $K = 2^n$ we can solve the problem with $K = n+1$ solutions to receive an optimal solution of (2).

4 An exact partition-based method

In this section, we outline an algorithmic framework able to solve the partition based formulation (3) of the K-adaptability two-stage stochastic problem. By iteratively adding scenarios to a current subpartition, our method produces a subpartition-based tree, where a node is identified by a *subpartition* $\mathcal{R} = (R_1, \ldots, R_K)$ of the scenario set (see Definition 2), by the number M of non-empty subsets in the subpartition and by its depth d in the tree. We denote this node by $N(\mathcal{R}, M, d)$. As it will be clarified in Section 4.3, if a node $N(\mathcal{R}, M, d)$ is not pruned, a scenario $s \notin \mathcal{R}$ is selected and $K' = \min\{M+1, K\}$ child nodes are produced, each one by adding s to one of the M non-empty elements of $\mathcal{R}$ or to the first empty one (if $M < K$). Note that a node at depth d in the tree corresponds to a subpartition that contains exactly d scenarios, that is $| \cup_{j=1}^{K} R_j| = d$. The leaf nodes of our subpartition-based tree are found at depth ℓ and they correspond to partitions of S according to Definition 1. In Figure 1, we report an example of a subpartition-based tree, for an instance with $\ell = 4$ and $K = 3$.



$(\emptyset,\emptyset,\emptyset)$
$(\{s_1\},\emptyset,\emptyset)$
$(\{s_1,s_2\},\emptyset,\emptyset)$
$(\{s_1\},\{s_2\},\emptyset)$
$(\{s_1,s_2,s_3\},\emptyset,\emptyset)$
$(\{s_1,s_2\},\{s_3\},\emptyset)$
$(\{s_1,s_3\},\{s_2\},\emptyset)$
$(\{s_1\},\{s_2,s_3\},\emptyset)$
$(\{s_1\},\{s_2\},\{s_3\})$

Figure 1: Example of a subpartition-based tree for $S = \{s_1, s_2, s_3, s_4\}$, $\ell = 4$ and $K = 3$.

To avoid the enumeration of all possible partitions, our solution scheme relies on solving a sequence of *subpartition induced problems*, which are primarily used to compute valid lower bounds on the global solution of (3). The subpartition induced problems are also key to guide scenario-selection, to warmstart the subsequent subproblems as well as to compute valid upper bounds, as it will be clarified in the following sections.
Given a subpartition $\mathcal{R}$, we define the *subpartition induced problem* as

$$\begin{aligned} spip(\mathcal{R}) := \min\ & f(x) + \sum_{j=1}^{K} \sum_{s \in R_j} p_s g_s(y^j) + \sum_{s \notin \mathcal{R}} p_s g_s(y_s) \\ \text{s.t.}\ & x \in X, \\ & y^j \in Y_{R_j}(x), \quad j = 1, \ldots, K, \\ & y_s \in Y_s(x), \quad s \notin \mathcal{R}, \end{aligned} \tag{$spip(\mathcal{R})$}$$

where for each $j = 1, \ldots, K$ we define

$$Y_{R_j}(x) = \begin{cases} \mathbb{R}^{n_y} & \text{if } R_j = \emptyset, \\ \cap_{s \in R_j} Y_s(x) & \text{otherwise.} \end{cases}$$

Let us observe that ($spip(\mathcal{R})$) is feasible only if $x \in X$ exists such that $Y_{R_j}(x) \neq \emptyset,\ \forall\ j = 1, \dots, K$ and $Y_s(x) \neq \emptyset,\ \forall\ s \notin \mathcal{R}$. When ($spip(\mathcal{R})$) is feasible, we will denote its optimal solution as $(x_R, \{y^j_R\}_{j=1}^K, \{y^R_s\}_{s \notin \mathcal{R}})$. Notice that in the subpartition induced problem we have a variable y^j for each element of the subpartition $\mathcal{R}$ and a variable y_s for each scenario that does not belong to the current subpartition. Therefore, if ($spip(\mathcal{R})$) is feasible, the set of K solutions $\{y^j_R\}_{j=1}^K$ is such that they cover all the scenarios belonging to the subpartition $\mathcal{R}$, while the scenarios s outside of $\mathcal{R}$ have individual policies y^R_s.
Finally, observe that if $\mathcal{R} = \mathcal{R}^0 = (\emptyset, \dots, \emptyset)$, then ($spip(\mathcal{R})$) is exactly equivalent to the extensive formulation (2) of the underlying two-stage stochastic program. In fact, since $R_j = \emptyset$ for all j, we have a variable y_s for each $s \in S$ and the term $\sum_{j=1}^K \sum_{s \in R_j} p_s g_s(y^j)$ is null. Using the notation from Section 3, we can write

$$\mathrm{opt}(\ell) = spip(\mathcal{R}^0).$$

On the other hand, if $\mathcal{R} = \mathcal{P}$, where $\mathcal{P}$ is a partition of S, then ($spip(\mathcal{R})$) is equivalent to ($pip(\mathcal{P})$), since there are no scenarios $s \notin \mathcal{R}$.

4.1 Computation of lower bounds through subpartitions

We can show that $spip(\mathcal{R})$ gives a valid lower bound for the subtree rooted in $\mathcal{R}$, i.e., it constitutes a lower bound on the optimal values of all problems induced by partitions that contain $\mathcal{R}$. First, we show that adding new scenarios to a subpartition produces a monotone increasing sequence of valid lower bounds.

Proposition 1. *Let $\mathcal{R}^1, \mathcal{R}^2$ be subpartitions of S such that $\mathcal{R}^1 \subseteq \mathcal{R}^2$ and such that ($spip(\mathcal{R})$) is feasible for both $\mathcal{R} = \mathcal{R}^1$ and $\mathcal{R} = \mathcal{R}^2$. Then*

$$spip(\mathcal{R}^1) \leq spip(\mathcal{R}^2).$$

Proof. Let $(\hat{x}, \{\hat{y}^j\}_{j=1}^K, \{\hat{y}_s\}_{s \notin \mathcal{R}^2})$ be a feasible solution for ($spip(\mathcal{R})$) with $\mathcal{R} = \mathcal{R}^2$. Then, we can rewrite the corresponding value of the objective function as follows,

$$f(\hat{x}) + \sum_{j=1}^K \sum_{s \in R^2_j} p_s g_s(\hat{y}^j) + \sum_{s \notin \mathcal{R}^2} p_s g_s(\hat{y}_s) = f(\hat{x}) + \sum_{j=1}^K \sum_{s \in R^1_j} p_s g_s(\hat{y}^j) + \sum_{j=1}^K \sum_{s \in R^2_j \setminus R^1_j} p_s g_s(\hat{y}^j) + \sum_{s \notin \mathcal{R}^2} p_s g_s(\hat{y}_s) = \hat{z}.$$

For every $s \in \mathcal{R}^2 \setminus \mathcal{R}^1$, $\exists\ j_s \in \{1, \dots, K\}$ such that $s \in R^2_{j_s} \setminus R^1_{j_s}$, and for each $s \notin \mathcal{R}^1$ we define

$$\tilde{y}_s = \begin{cases} \hat{y}_s & \text{if } s \notin \mathcal{R}^2, \\ \hat{y}^{j_s} & \text{if } s \in \mathcal{R}^2 \setminus \mathcal{R}^1. \end{cases}$$

The solution $(\hat{x}, \{\hat{y}^j\}_{j=1}^K, \{\tilde{y}_s\}_{s \notin \mathcal{R}^1})$ is feasible for ($spip(\mathcal{R})$) with $\mathcal{R} = \mathcal{R}^1$, and by continuing the previous chain of equalities we obtain

$$\begin{aligned} \hat{z} &= f(\hat{x}) + \sum_{j=1}^K \sum_{s \in R^1_j} p_s g_s(\hat{y}^j_p) + \sum_{s \notin \mathcal{R}^1} p_s g_s(\tilde{y}_s) \\ &\geq spip(\mathcal{R}^1). \end{aligned}$$

Since the previous inequality holds for any feasible solution of ($spip(\mathcal{R})$) with $\mathcal{R} = \mathcal{R}^2$, in particular it holds for the optimal solution and we get that $spip(\mathcal{R}^2) \geq spip(\mathcal{R}^1)$. □

From Proposition 1, we get that any partition $\mathcal{P}$ containing $\mathcal{R}$ will lead to an optimal value that is greater or equal to $spip(\mathcal{R})$. This also means that $spip(\mathcal{R})$ constitutes a lower bound for (3). We formalize this result in the following proposition.

Proposition 2. *Let $\mathcal{R}$ be a subpartition of S and let $\mathcal{P}$ be a partition of S such that $\mathcal{R} \subseteq \mathcal{P}$. Then*

$$spip(\mathcal{R}) \leq pip(\mathcal{P}).$$

Proof. From $\mathcal{R} \subseteq \mathcal{P}$ and Proposition 1 we have $spip(\mathcal{R}) \leq spip(\mathcal{P})$. Since $\mathcal{P}$ is a partition of S, then it holds $spip(\mathcal{P}) = pip(\mathcal{P})$ which proves the result. □

Remark 2. *For any node $N(\mathcal{R}, M, d)$ such that ($spip(\mathcal{R})$) is feasible, we have that*

$$\mathrm{opt}(M + (\ell - d)) \leq spip(\mathcal{R}).$$

Indeed, we have already observed that by solving ($spip(\mathcal{R})$), when it is feasible, we obtain M policies y^j, $j = 1, \ldots, M$, that cover all the scenarios in $\mathcal{R}$ and $\ell - d$ solutions y_s, $s \notin \mathcal{R}$, that cover the remaining scenarios, for a total of $M + \ell - d$ second stage policies. Therefore, we have a feasible solution for the K-adaptability problem corresponding to $K = M + (\ell - d)$.

From Proposition 1 we can also show that, in certain cases, we can avoid solving ($spip(\mathcal{R})$), since its optimal value can be inherited from previously solved problems.

Corollary 1. *Let $\mathcal{R} = (R_1, \ldots, R_M, \emptyset, \ldots, \emptyset)$ be a subpartition of order $M < K$ such that $\hat{s} \notin \mathcal{R}$ exists. Let $\mathcal{R}' = (R_1, \ldots, R_M, \{\hat{s}\}, \emptyset, \ldots, \emptyset)$, then it holds*

$$spip(\mathcal{R}) = spip(\mathcal{R}').$$

In particular, given $s_1, \ldots, s_K \in S$, $\mathcal{R}^0 = (\emptyset, \ldots, \emptyset)$ and $\mathcal{R}^K = (\{s_1\}, \ldots, \{s_K\})$, it holds

$$spip(\mathcal{R}^0) = spip(\mathcal{R}^K).$$

Proof. Since $\mathcal{R} \subseteq \mathcal{R}'$, from Proposition 1 it holds $spip(\mathcal{R}) \leq spip(\mathcal{R}')$. Let $(x_R, \{y_R^j\}_{j=1}^K, \{y_s^R\}_{s \notin \mathcal{R}})$ be the optimal solution of ($spip(\mathcal{R})$) induced by $\mathcal{R}$. Then,

$$\begin{aligned}
spip(\mathcal{R}) &= f(x_R) + \sum_{j=1}^{K} \sum_{s \in R_j} p_s g_s(y_R^j) + \sum_{s \notin \mathcal{R}} p_s g_s(y_s^R) \\
&= f(x_R) + \sum_{j=1}^{M} \sum_{s \in R_j} p_s g_s(y_R^j) + p_{\hat{s}} g_{\hat{s}}(y_{\hat{s}}^R) + \sum_{s \notin \mathcal{R}, s \neq \hat{s}} p_s g_s(y_s^R) \\
&= f(x_R) + \sum_{j=1}^{K} \sum_{s \in R'_j} p_s g_s(\tilde{y}_{R'}^j) + \sum_{s \notin \mathcal{R}'} p_s g_s(y_s^R) \geq spip(\mathcal{R}')
\end{aligned}$$

where for every $j \in \{1, \ldots, K\}$ we have defined

$$\tilde{y}_{R'}^j = \begin{cases} y_{\hat{s}}^R & \text{if } j = M + 1, \\ y_R^j & \text{otherwise,} \end{cases}$$

and the last inequality follows from $(x_R, \{\tilde{y}_{R'}^j\}_{j=1}^K, \{y_s^R\}_{s \notin \mathcal{R}'})$ being a feasible solution for ($spip(\mathcal{R})$) with $\mathcal{R} = \mathcal{R}'$.
Furthermore, given $s_1, \ldots, s_K \in S$, by iteratively applying the above reasoning to the sequence of subpartitions $\mathcal{R}^0 \subseteq \mathcal{R}^1 \subseteq \ldots \subseteq \mathcal{R}^K$, where $\mathcal{R}^i = (\{s_1\}, \ldots, \{s_i\}, \emptyset, \ldots, \emptyset)$ for $i \in \{1, \ldots, K\}$, we obtain $spip(\mathcal{R}^0) = spip(\mathcal{R}^K)$. □

Proposition 3. *Let $\mathcal{R}^1, \mathcal{R}^2$ be subpartitions of S such that $\mathcal{R}^1 \subseteq \mathcal{R}^2$ and such that ($spip(\mathcal{R})$) is infeasible for $\mathcal{R} = \mathcal{R}^1$, then it holds that ($spip(\mathcal{R})$) is infeasible for $\mathcal{R} = \mathcal{R}^2$.*

Proof. By assumption, ($spip(\mathcal{R})$) with $\mathcal{R} = \mathcal{R}^1$ is infeasible, then for every $x \in X$ there exists $j \in \{1, \ldots, K\}$ such that $Y_{R_j^1}(x) = \emptyset$ or there exists $s \notin \mathcal{R}^1$ such that $Y_s(x) = \emptyset$.
Let us assume by contradiction that ($spip(\mathcal{R})$) with $\mathcal{R} = \mathcal{R}^2$ is feasible and let $(\hat{x}, \{\hat{y}^j\}_{j=1}^K, \{\hat{y}_s\}_{s \in \mathcal{R}^2})$ be a feasible solution. Then, for $\hat{x}$ there exists $j \in \{1, \ldots, K\}$ such that $Y_{R_j^1}(x) = \emptyset$ or there exists $s \notin \mathcal{R}^1$ such that $Y_s(x) = \emptyset$. From $\mathcal{R}^1 \subseteq \mathcal{R}^2$, it holds $R_j^1 \subseteq R_j^2$ for every $j \in \{1, \ldots, K\}$ and

$$\hat{y}^j \in Y_{R_j^2}(x) = \cap_{s \in R_j^2} Y_s(x) \subseteq \cap_{s \in R_j^1} Y_s(x) = Y_{R_j^1}(x) \neq \emptyset.$$

With a similar reasoning we can show that $Y_s(\hat{x}) \neq \emptyset$ for each $s \notin \mathcal{R}_1$, which is a contradiction. □

4.2 Computation of upper bounds

Valid upper bounds on the optimal solution of (3) can be obtained by building a partition $\mathcal{P}$ of the scenario set and evaluating the corresponding $pip(\mathcal{P})$ or by evaluating the objective function of (k-2SSP) on any of its feasible solutions.
Whenever we reach a leaf of our subpartition-based tree, we get a node $N(\mathcal{R}, M, d)$ with $d = \ell$ and $\mathcal{R}$ being a partition of the scenario set, as $\{s | s \notin \mathcal{R}\} = \emptyset$. Therefore, on leaf nodes, the subpartition-induced problem is equivalent to the partition induced problem ($pip(\mathcal{P})$) with $\mathcal{P} = \mathcal{R}$, so that $spip(\mathcal{R})$ gives a valid upper bound.
In non-leaf nodes $N(\mathcal{R}, M, d)$, i.e. when $d < \ell$, we have subpartitions that only *cover* a subset of the scenario set. However, if problem ($spip(\mathcal{R})$) is feasible, a partition of the scenario set can be derived following the intuition behind the heuristic approach introduced in Buchheim and Pruente (2019).
Given $s \in S$ and the optimal solution $(x_R, \{y_R^j\}_{j=1}^K, \{y_s^R\}_{s \notin \mathcal{R}})$ of ($spip(\mathcal{R})$), we define

$$sip(s; \mathcal{R}) = \{j \in \{1, \dots, M\} \mid y_R^j \in Y_s(x_R)\}$$

as the set of indices corresponding to second stage solutions y_R^j that are feasible for scenario s.
If the set $sip(s; \mathcal{R})$ is non-empty for each scenario $s \in S$ a partition $\mathcal{P}^R = (P_1^R, \dots, P_K^R)$ induced by $(x_R, y_R^1, \dots, y_R^M)$ can be defined setting

$$P_j^R := \left\{ s \in S : j \in \operatorname*{arg\,min}_{k \in sip(s;\mathcal{R})} \{p_s g_s(y_R^k)\} \right\}, \tag{7}$$

for $j = 1, \dots, M$ and $P_j^R = \emptyset$, $j = M+1, \dots, K$. Hence, to build a feasible partition we assign every scenario s to the subset P_j^R corresponding to the solution y_R^j, $j \in sip(s; \mathcal{R})$, that has the smallest value of the objective function.

Algorithm 1 `PartitionInduced(`$\mathcal{R}$`)`

```
INPUT: $(x_R, \{y_R^j\}_{j=1}^K, \{y_s^R\}_{s \notin \mathcal{R}})$, optimal solution of ($spip(\mathcal{R})$), $M$
1: Set PartitionInduced($\mathcal{R}$) $= 1$
2: for $s \in S$ do
3:     Set $sip(s; \mathcal{R}) := \{j \in \{1, \dots, M\} \mid y_R^j \in Y_s(x_R)\}$
4:     if $sip(s; \mathcal{R}) = \emptyset$ then
5:         Set PartitionInduced($\mathcal{R}$) $= 0$ and Return
6:     end if
7: end for
```

If `PartitionInduced(`$\mathcal{R}$`)` $= 1$, that is, we have successfully built a partition starting from the optimal solution of the subpartition induced problem, then we have two possibilities to compute valid upper bounds, which differ in terms of tightness and computational effort.
A first possibility is to get a valid upper bound for (3) by evaluating ($pip(\mathcal{P})$) with $\mathcal{P} = \mathcal{P}^R$. This approach requires the solution of an additional optimization problem, but it provides the best possible upper bound induced by partition $\mathcal{P}^R$. A second option is to obtain a possibly worse valid upper bound which, however, is free of additional computation costs, since it relies solely on the solution of problem ($spip(\mathcal{R})$) and does not re-optimize for the optimal x for the given partition. Indeed, in case a partition $\mathcal{P}^R$ can be induced by the optimal solution of ($spip(\mathcal{R})$), we have that $(x_R, \{y_R^j\}_{j=1}^K)$ is a feasible solution for ($pip(\mathcal{P})$) with $\mathcal{P} = \mathcal{P}^R$ and then the value

$$f(x_R) + \sum_{j=1}^K \sum_{s \in P_j^R} p_s g_s(y_R^j),$$

which is greater or equal than the optimal value $pip(\mathcal{P}^R)$, constitutes a valid upper bound for (3).

4.3 Pruning rules

As detailed later in Algorithm 2, a node $N(\mathcal{R}, M, d)$ belonging to the partition-induced tree is pruned whenever one of the following conditions is satisfied:

(i) Problem ($spip(\mathcal{R})$) is infeasible,

(ii) $spip(\mathcal{R}) > UB$, with UB being the current best known upper bound,

(iii) $M + \ell - d < K$.

The first two conditions correspond to the usual *pruning by infeasibility* and *pruning by bound*. The third condition states that we can prune any node such that the number of scenarios that do not belong to the current subpartition, that is $(\ell - d)$, is smaller than the number of empty elements of $\mathcal{R}$, that is $(K - M)$. This follows from the fact that in (3) we are optimizing over all the partitions of order exactly K of the scenario set. Observe that additionally, we can prove that if an optimal partition of order $M < K$ exists, then there also exists an optimal partition of order $M = K$. This result is established in the following proposition, whose proof is given in the Appendix together with an auxiliary result used therein.

Proposition 4. *Let $\mathcal{P} = (P_1, \ldots, P_K)$ be the optimal solution of (3), then one of the following two conditions hold:*

1. *$\mathcal{P}$ is of order K, i.e. $P_j \neq \emptyset$, for all $j = 1, \ldots, K$;*

2. *$\mathcal{P}$ is of order $M < K$ and there exists $\widetilde{\mathcal{P}} = (\tilde{P}_1, \ldots, \tilde{P}_K)$ of order K such that $pip(\mathcal{P}) = pip(\widetilde{\mathcal{P}})$.*

Proposition 4 implies that any node $N(\mathcal{R}, M, d)$ such that $M + \ell - d < K$ can be pruned. Indeed, since the number of scenarios not contained in $\mathcal{R}$ is $\ell - d$, if $M + \ell - d < K$, we have that the partitions obtained from the subtree rooted in $\mathcal{R}$ will contain empty sets, or, in other words, they all will have order $M < K$. We report an example in the following.

Example 3. *Let $S = \{s_1, \ldots, s_4\}$ and $K = 3$. The node $N(\mathcal{R}, M, d)$ with $\mathcal{R} = (\{s_1, s_2, s_3\}, \emptyset, \emptyset)$, $M = 1$ and $d = 3$ is such that $2 = M + \ell - d < K = 3$. Then, the child nodes of $\mathcal{R}$ will be the partitions $\mathcal{P}_1 = (\{s_1, s_2, s_3, s_4\}, \emptyset, \emptyset)$ and $\mathcal{P}_2 = (\{s_1, s_2, s_3\}, \{s_4\}, \emptyset)$. Note that these partitions of S contain empty sets and the order of $\mathcal{P}_1$ is $M = 1$, while the order of $\mathcal{P}_2$ is $M = 2$. Therefore, even if one of the partitions $\mathcal{P}_1$ or $\mathcal{P}_2$ is optimal for (3), we have that partition $\bar{\mathcal{P}} = (\{s_1, s_2\}, \{s_3\}, \{s_4\})$ is optimal as well.*

We are now able to show that if one optimal partition of problem (3) of order $M = K$ exists, this is not eliminated by our pruning rule.

Proposition 5. *Let $\mathcal{P}^*$ be an optimal partition of problem (3) of order $M = K$. Then no subpartition $\mathcal{R}$ that contains $\mathcal{P}^*$ is pruned.*

Proof. Assume by contradiction that a subpartition $\mathcal{R}$ with $\mathcal{R} \subseteq \mathcal{P}^*$ exists such that it is pruned. Case (i): If ($spip(\mathcal{R})$) is infeasible then, from Proposition 3, since $\mathcal{R} \subseteq \mathcal{P}^*$, it follows that ($pip(\mathcal{P})$) with $\mathcal{P} = \mathcal{P}^*$ is infeasible, which is a contradiction. Case (ii): From Proposition 2, we know that $spip(\mathcal{R})$ provides a lower bound on any optimal value $pip(\mathcal{P})$ corresponding to partitions satisfying $\mathcal{R} \subseteq \mathcal{P}$. Therefore, if $spip(\mathcal{R}) > UB$, we can conclude that any upper bound obtainable from the partitions induced by $\mathcal{R}$ would be worse than UB, contradicting the optimality of $\mathcal{P}^*$. Case (iii): If $M + \ell - d < K$, it cannot be that $\mathcal{R} \subseteq \mathcal{P}^*$ as the partitions obtained from the subtree rooted in $\mathcal{R}$ will contain empty sets, or, in other words, they all will have order $M < K$. □

Algorithm 2 `Prune`$(N(\mathcal{R}, M, d))$

```
INPUT: $N(\mathcal{R}, M, d)$, $UB$
1: Set Prune$(N(\mathcal{R}, M, d)) = 0$
2: if (($spip(\mathcal{R})$) is infeasible) or ($spip(\mathcal{R}) > UB$) or ($M + \ell - d < K$) ) then
3:     Set Prune$(N(\mathcal{R}, M, d)) = 1$
4: end if
```

4.4 Branching rule

Given a subpartition $\mathcal{R}$ of the scenario set S, the $K!$ subpartitions obtained by permuting $\mathcal{R}$ lead to equivalent subpartition-induced problems. Therefore, the generation of such equivalent or *symmetric* subpartitions should be avoided. In Subramanyam et al. (2020), this issue was addressed by defining a suitable branching strategy, which we will call *child generation rule*. We will prove (see Proposition 9) that in the context of our branch-and-bound method, this branching strategy allows us to successfully avoid symmetries.

Definition 3 (Child generation rule)**.** *Given a node $N(\mathcal{R}, M, d)$ with $\mathcal{R} = (R_1, \dots, R_K)$ and a new scenario $s \notin \mathcal{R}$, we define*

$$K' = \min\{M + 1, K\}. \tag{8}$$

Then, K' child nodes are generated: $N(\mathcal{R}^j, M, d+1)$ for $j = 1, \dots, K'-1$ and $N(\mathcal{R}^{K'}, K', d+1)$, with $\mathcal{R}^j = (R_1, \dots, R_j \cup \{s\}, \dots, \mathcal{R}_K)$, for $j = 1, \dots, K'$.

Note that if $M < K$, by implementing this branching rule we obtain exactly M child subpartitions of order M and one child subpartition of order $M + 1$. Since our branch-and-bound method starts from the subpartition containing only empty sets, $\mathcal{R} = (\emptyset, \dots, \emptyset)$ of order $M = 0$, by iteratively applying this child generation rule, we obtain subpartitions of order $M = 1, \dots, K$ such that the M non empty elements are always the first M subsets of the corresponding node, i.e. if $\mathcal{R}$ is of order M, then $\mathcal{R} = (R_1, \dots, R_M, \emptyset, \dots, \emptyset)$, where $R_j \neq \emptyset$ for $j = 1, \dots, M$.

4.5 Algorithm and correctness of PiM

We report in Algorithm 3 the complete scheme of the partition-induced branch-and-bound method, whose main components have been described in the previous subsections.

Algorithm 3 `PiM`: a partition induced method

INPUT: (k-2SSP)

OUTPUT: optimal solution of (k-2SSP) or detect infeasibility (by $UB = +\infty$)

```
 1: Set $\mathcal{R} = (R_1, \dots, R_K) = (\emptyset, \dots, \emptyset)$, $M = 0$, $d = 0$
 2: Set $UB = +\infty$
 3: $\mathcal{L} := \{N(\mathcal{R}, M, d)\}$
 4: while $\mathcal{L} \neq \emptyset$ do
 5:     Choose $N(\mathcal{R}, M, d) \in \mathcal{L}$
 6:     $\mathcal{L} \leftarrow \mathcal{L} \setminus \{N(\mathcal{R}, M, d)\}$
 7:     if Prune($N(\mathcal{R}, M, d)$) then
 8:         Continue
 9:     else if $d = \ell$ then
10:         Update $UB := \min\{UB, spip(\mathcal{R})\}$
11:     else
12:         if PartitionInduced($\mathcal{R}$) then
13:             Define a partition $\mathcal{P}^R$ as in (7)
14:             Update $UB := \min\{UB, pip(\mathcal{P}^R)\}$
15:         end if
16:         Choose $s \notin \mathcal{R}$ and produce $K' = \min\{M + 1, K\}$ child nodes
17:         $\mathcal{L} \leftarrow \mathcal{L} \cup \{N(\mathcal{R}_1, M, d+1), \dots, N(\mathcal{R}_{K'-1}, M, d+1), N(\mathcal{R}_{K'}, K', d+1)\}$
18:     end if
19: end while
```

Denoting by $\mathcal{L}$ the list of nodes that need to be explored, we start by considering the node $N(\mathcal{R}, M, d) = N((\emptyset, \dots, \emptyset), 0, 0)$ as the root node, i.e. we start with the naive subpartition made of all empty subsets. The starting upper bound UB is set to $+\infty$. We go on evaluating nodes until the list $\mathcal{L}$ is empty. While $\mathcal{L} \neq \emptyset$, let $N(\mathcal{R}, M, d)$ be a node and let ($spip(\mathcal{R})$) be the corresponding subpartition induced problem. If ($spip(\mathcal{R})$) is feasible, we obtain a solution $(x_R, \{y_R^j\}_{j=1}^K, \{y_s^R\}_{s \notin \mathcal{R}})$. As shown in Section 4.1, the optimal value of ($spip(\mathcal{R})$) is a valid lower bound for (3). Node $N(\mathcal{R}, M, d)$ can be pruned - and we set `Prune`$(N(\mathcal{R}, M, d)) = 1$ (see Algorithm 2) - in case either problem ($spip(\mathcal{R})$) is infeasible or in case the optimal solution of ($spip(\mathcal{R})$) gives a lower bound greater than the current upper bound UB or in case $M + \ell - d < K$ (see Section 4.4).

If the optimal solution of ($spip(\mathcal{R})$) induces a partition $\mathcal{P}^R$, a valid upper bound for (k-2SSP) can be obtained as detailed in Section 4.2 and the current upper bound is updated accordingly (see Step 14 in Algorithm 3). The algorithm proceeds by selecting a new scenario that was not included in the previously analyzed subpartition. At Step 16 in Algorithm 3, a scenario $s \notin \mathcal{R}$ is selected and $K' = \min\{M+1, K\}$ child nodes are defined, according to the child generation rule defined in Definition 3. Obviously, the child nodes of $N(\mathcal{R}, M, d)$ will be at depth $d+1$ in the subpartition-based tree. Using the results shown in the previous sections, we are able to prove that our partition-induced method correctly detects an optimal solution of (k-2SSP) in case it exists.

Theorem 2. *Algorithm 1 stops after a finite number of iterations either returning that Problem (k-2SSP) is infeasible, or returning its optimal solution.*

Proof. Every level d of the subpartition-based tree built by Algorithm 1 is made of $\sum_{k=1}^{K} \mathcal{S}_{d,k}$ nodes (where $\mathcal{S}_{d,k} = 0$ in case $k > d$). Therefore, the total number of nodes of the tree is finite and equal to

$$1 + \sum_{d=1}^{\ell} \sum_{k=1}^{K} \mathcal{S}_{d,k}.$$

If problem (k-2SSP) is infeasible, then problem (3) is also infeasible, meaning that no partition $\mathcal{P}$ of the scenario set S yields a feasible problem ($pip(\mathcal{P})$). This means that the upper bound UB will never be updated along the iterations of Algorithm 1 and the Algorithm stops with $UB = +\infty$, detecting infeasibility of (k-2SSP). On the other hand, if at least one partition $\mathcal{P}$ exists that yields a feasible problem ($pip(\mathcal{P})$) the optimal solution of (k-2SSP) can be obtained by identifying an optimal partition for problem (3). From Proposition 4, we can restrict the search among partitions of order $M = K$. Thanks to Proposition 5, we have that the pruning rule adopted and detailed in Algorithm 2 is correct, meaning that if one optimal partition $\mathcal{P}^*$ of problem (3) of order $M = K$ exists, this is not eliminated. Then, when problem ($pip(\mathcal{P})$) with $\mathcal{P} = \mathcal{P}^*$ is solved, the upper bound UB will be updated accordingly and Algorithm 1 will stop after having enumerated a finite number of nodes with $UB = pip(\mathcal{P}^*)$ being the optimal value for (k-2SSP). The corresponding solution $(x_{P^*}, y^1_{P^*}, \dots, y^K_{P^*})$ is the optimal solution of (k-2SSP). □

4.6 Scenario and node selection strategies

In this section, we discuss possible strategies that can be adopted within Algorithm 3 to define new subpartitions, i.e. how to select a new scenario (Step 16), and to determine the order in which the subpartitions should be explored during the tree search (Step 5).

4.6.1 Scenario selection

Let $N(\mathcal{R}, M, d)$ be a node in our subpartition-based tree. Clearly, there are several ways for selecting a new scenario $s^* \notin \mathcal{R}$ to build the subpartitions of its child nodes. Choosing s^* as a random scenario among those that do not belong to any element of the subpartition $\mathcal{R}$, is of course a feasible strategy, but - from our computational experience - not very effective in most cases. Another possibility, is exploiting the information we obtain by solving the subpartition induced problem ($spip(\mathcal{R})$) for the subpartition $\mathcal{R}$. If problem ($spip(\mathcal{R})$) is infeasible, node $N(\mathcal{R}, M, d)$ is pruned by infeasibility. On the other hand, if problem ($spip(\mathcal{R})$) is feasible, we can use its optimal solution $(x_R, \{y^j_R\}_{j=1}^K, \{y^R_s\}_{s \notin \mathcal{R}})$ to guide the scenario selection as follows. For every $s \notin \mathcal{R}$ we have a corresponding vector y^R_s. The new scenario s^* can be chosen as the one that corresponds to the largest objective function value, that is

$$s^* \in \arg\max_{s \notin \mathcal{R}} p_s g_s(y^R_s).$$

This strategy is motivated by the fact that a node can be pruned if the lower bound that we obtain by solving the corresponding subpartition induced problem exceeds the incumbent upper bound. Therefore, by prioritizing scenarios that majorly contribute to the objective function, we aim to obtain stronger lower bounds earlier in the exploration of the tree.
We change the previous rule when the solution induced by $\mathcal{R}$ does not itself induce a partition. Indeed, if `PartitionInduced(`$\mathcal{R}$`)` $= 0$, we have that at least one $s^* \notin \mathcal{R}$ exists such that $sip(s^*; x_R, y^1_R, \dots, y^K_R) = \emptyset$.

Then, we can select scenario s^* to be used in the child generation rule at the next step. The rationale is that by introducing earlier the scenarios for which no current second-stage strategy is feasible, we can more rapidly identify infeasible combinations of scenarios that render problem ($spip(\mathcal{R})$) infeasible, thus allowing to prune the new generated node by infeasibility.

4.6.2 Node selection

Let $\mathcal{L}$ be, as before, the set of generated nodes that have not been explored yet. Selecting a random node from $\mathcal{L}$ is a feasible node selection strategy, however, after some preliminary computational tests, we implemented the following policy.
Given a node $N(\mathcal{R}, M, d)$, we implement different node selection strategies, according to the value of its depth d:

- if $0 \leq d < \ell - 1$, given the new scenario s^*, we select a random index $k_{new} \in \{1, \ldots, K'\}$ (where K' is defined according to (8)), and we first visit the node $\mathcal{R}^{new} = (R_1^{new}, \ldots, R_K^{new})$ where $R_j^{new} = R_j$ for each $j \neq k_{new}$ and $R_{k_{new}}^{new} = R_{k_{new}} \cup \{s^*\}$;
- if $d = \ell - 1$, we sequentially visit all the child nodes $\{\mathcal{F}^1, \ldots, \mathcal{F}^{K'}\}$ which are complete partitions of the scenario set corresponding to $i = \ell$. After these nodes have been exhausted, we select a random element from $\mathcal{L}$.

This node selection strategy is convenient for multiple reasons. First, it allows to obtain feasible solutions early on, since by starting with a depth-first strategy we quickly reach leaf nodes, i.e. partitions of the scenario set. Furthermore, in the first case, the problem induced by $\mathcal{R}^{new}$ can be obtained from the one induced by $\mathcal{R}$ by removing variable y_{s^*} and its associated constraints and including the term $p_{s^*} g_{s^*}(y^{k_{new}})$ in the objective function. In the second case, corresponding to nodes that are subpartitions of depth $\ell - 1$, it is computationally efficient to consider immediately all the child nodes of $\mathcal{R}$, which are partitions of S, since the corresponding induced problems can be derived similarly from the model induced by $\mathcal{R}$, with only minor modifications.

4.7 Breaking *symmetry* and avoiding *emptiness*

In this section, we show that our partition based tree enjoys two properties. On the one hand, given a subpartition $\mathcal{R}$, the generation of any of its permutations is excluded, or, in other words, *symmetries* are avoided. A formal analysis is given in the Appendix. On the other hand, by modifying the child generation rule previously adopted for branching, only partitions of order K, not containing empty elements, are generated, or in other words, *emptiness* is avoided.
We show that if the current node $N(\mathcal{R}, M, d)$ is such that the number of its empty elements is exactly the same as the number of scenarios that do not belong to $\mathcal{R}$, then, the optimal solution of ($spip(\mathcal{R})$) gives an upper bound to (3). Observe that this is what we stated in Remark 2. In this case $M + (\ell - d) = K$, therefore the inequality in Remark 2 becomes $\text{opt}(M + (\ell - d)) = \text{opt}(K) \leq spip(\mathcal{R})$. We formally prove the result in the following proposition.

Proposition 6. *Let $N(\mathcal{R}, M, d)$ be a node such that $u = \ell - d = K - M$. W.l.o.g. let $\{s_1, \ldots, s_u\} = S \setminus \mathcal{R}$. Let $\mathcal{P} = (P_1, \ldots, P_K)$, where*

$$P_j = \begin{cases} R_j & \text{if } j \in \{1, \ldots, M\}, \\ \{s_i\} & \text{if } j = M + i, \text{ for } i \in \{1, \ldots, u\}. \end{cases}$$

Then the following statements hold:

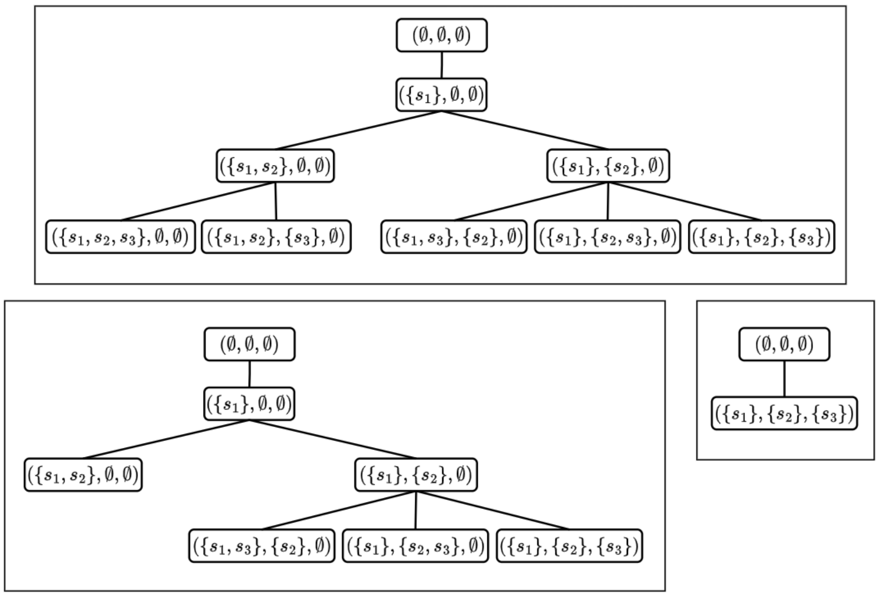
1. *$\mathcal{P}$ is the only partition of order K such that $\mathcal{R} \subseteq \mathcal{P}$;*
2. *$spip(\mathcal{R}) = pip(\mathcal{P})$;*
3. *$pip(\mathcal{P}) \leq pip(\widetilde{\mathcal{P}})$ for any other partition $\widetilde{\mathcal{P}} \neq \mathcal{P}$ such that $\mathcal{R} \subseteq \widetilde{\mathcal{P}}$.*

Proof. The first statement follows directly by the definition of $\mathcal{P}$ and the fact that $u = \ell - d = K - M$. The second statement follows by iteratively applying to $\mathcal{R}$ the first result in Corollary 1 considering scenarios s_i, $i = 1, \dots, u$, until we obtain $\mathcal{P}$.
To prove the third statement, let $\widetilde{\mathcal{P}}$ be a partition of S of order M such that $\mathcal{R} \subseteq \widetilde{\mathcal{P}}$ and $\mathcal{P}^* \neq \mathcal{P}$. From the first statement it must hold $M < K$. Let $(\tilde{x}, \{\tilde{y}^j\}_{j=1}^K)$ be the optimal solution of ($pip(\mathcal{P})$) corresponding to partition $\widetilde{\mathcal{P}}$. For every $i \in \{1, \dots, u\}$ let $k(i) \in \{1, \dots, M\}$ be such that $s_i \in \widetilde{P}_{k(i)}$. Then $(x, \{y_j\}_{j=1}^K)$, where $x = \tilde{x}$, $y^j = \tilde{y}^j$ for $j = 1, \dots, M$ and $y^{M+i} = \tilde{y}^{k(i)}$ for $i = 1, \dots, u$, is feasible for the problem ($pip(\mathcal{P})$) induced by partition $\mathcal{P}$, hence $pip(\mathcal{P}) \leq pip(\widetilde{\mathcal{P}})$. $\square$

Following the previous proposition, whenever a node $N(\mathcal{R}, M, d)$ is such that $\ell - d = K - M$, we can use $spip(\mathcal{R})$ to check if we can update the incumbent upper bound, since $spip(\mathcal{R}) = pip(\mathcal{P})$, where $\mathcal{P}$ is the only partition of order K that contains $\mathcal{R}$, and then, either way, we do not generate the child-nodes of $\mathcal{R}$. This result can be directly included into a modified child generation rule:

Definition 4 (Enhanced child generation rule)**.** *Let $N(\mathcal{R}, M, d)$ be a node with $\mathcal{R} = (R_1, \dots, R_K)$ and let $\{s_1, \dots, s_u\} = S \setminus \mathcal{R}$, with $u = \ell - d$. If $u = K - M$, a single child node is built, $N(\mathcal{R}^{new}, K, d+1)$ where $\mathcal{R}^{new} = (R_1, \dots, R_M, \{s_1\}, \dots, \{s_u\})$. Otherwise $K' = \min\{M+1, K\}$ nodes are built according to Definition 3.*

Figure 2 shows an example of how introducing this enhanced child generation rule effectively reduces the size of the partition-based tree.



$(\emptyset, \emptyset, \emptyset)$
$(\{s_1\}, \emptyset, \emptyset)$
$(\{s_1, s_2\}, \emptyset, \emptyset)$
$(\{s_1\}, \{s_2\}, \emptyset)$
$(\{s_1, s_2, s_3\}, \emptyset, \emptyset)$
$(\{s_1, s_2\}, \{s_3\}, \emptyset)$
$(\{s_1, s_3\}, \{s_2\}, \emptyset)$
$(\{s_1\}, \{s_2, s_3\}, \emptyset)$
$(\{s_1\}, \{s_2\}, \{s_3\})$
$(\emptyset, \emptyset, \emptyset)$
$(\{s_1\}, \emptyset, \emptyset)$
$(\{s_1, s_2\}, \emptyset, \emptyset)$
$(\{s_1\}, \{s_2\}, \emptyset)$
$(\{s_1, s_3\}, \{s_2\}, \emptyset)$
$(\{s_1\}, \{s_2, s_3\}, \emptyset)$
$(\{s_1\}, \{s_2\}, \{s_3\})$
$(\emptyset, \emptyset, \emptyset)$
$(\{s_1\}, \{s_2\}, \{s_3\})$

Figure 2: Example of the partition-based tree for $\ell = 3$ and $K = 3$. On the top, the classical tree obtained by applying Definition 3. On the bottom left, the tree we obtain by applying the pruning rule (iii) to the classical tree. On the bottom right the tree we obtain by Definition 4.

4.8 Improvements to the basic scheme of `PiM`

Along the iterations of `PiM`, additional information can be stored to accelerate the enumeration of the nodes. We describe here several computational enhancements that can be incorporated into the implementation of our partition-induced method.
Given a node $N(\mathcal{R}, M, d)$, where $\mathcal{R} \neq (\emptyset, \dots, \emptyset)$, we denote by $N(\mathcal{R}^{old}, M^{old}, d-1)$ its parent. By $(x_{old}, \{y^j_{old}\}_{j=1}^K, \{y^{old}_s\}_{s \notin \mathcal{R}^{old}})$ we denote the optimal solution of ($spip(\mathcal{R})$) for $\mathcal{R} = \mathcal{R}^{old}$ and by lb_{old} the optimal value $spip(\mathcal{R}^{old})$. Note that the subpartition $\mathcal{R}$ is obtained from $\mathcal{R}^{old}$ adding a scenario $s^* \notin \mathcal{R}^{old}$. In particular, an index k_{new} exists such that $R_j = R^{old}_j$ for all $j \in \{1, \dots, K\}$, $j \neq k_{new}$ and $R_{k_{new}} = R^{old}_{k_{new}} \cup \{s^*\}$.

Prune by bound using lb_{old}

Thanks to Proposition 1, we have that $spip(\mathcal{R}^{old}) \le spip(\mathcal{R})$. This means that lb_{old} is a valid lower bound and in case $lb_{old} > UB$ we can prune node $N(\mathcal{R}, M, d)$ without the need of solving ($spip(\mathcal{R})$) (i.e. we should check this condition between lines 6 and 7 of Algorithm 3). Clearly this may work only in case the upper bound UB improved before addressing node $N(\mathcal{R}, M, d)$.

Skip a node

If $\mathcal{R}^{old}$ is a subpartition of order $M^{old} < K$, and $k_{new} = M^{old}+1$, then $\mathcal{R}^{old} = (R_1^{old}, \dots, R_{M^{old}}^{old}, \emptyset, \dots, \emptyset)$ and $\mathcal{R} = (R_1^{old}, \dots, R_{M^{old}}^{old}, \{s^*\}, \emptyset, \dots, \emptyset)$. Following Lemma 1, there is no need to solve ($spip(\mathcal{R})$) induced by partition $\mathcal{R}$, since it holds $lb_{old} = spip(\mathcal{R}^{old}) = spip(\mathcal{R})$. Furthermore, the optimal solution of ($spip(\mathcal{R})$) corresponding to $\mathcal{R}$ can be easily built as $x_R = x_{old}$, $y_R^j = y_{old}^j$ for all $j \neq k_{new}$ and $y_R^{k_{new}} = y_{s^*}^{old}$, and $y_s^R = y_s^{old}$ for all $s \notin \mathcal{R}$.

Time limit at the node

In solving ($spip(\mathcal{R})$) at node $N(\mathcal{R}, M, d)$, we can introduce a safeguard parameter $t_{\max}$ which is the maximum time we allow for the solution of the subpartition induced problem at a node $\mathcal{R}$. If we cannot prove its infeasibility or its optimality within the time limit, then we proceed to generate the child nodes according to the rule in Definition 4, simply selecting the new scenario s^* as a random scenario among those that are not in $\mathcal{R}$. Indeed, we cannot rely on the optimal solution of ($spip(\mathcal{R})$) to guide the the scenario selection as detailed in section 4.6.1 and we have no values to record for $x_R, \{y_R^j\}_{j=1}^K, \{y_s^R\}_{s\notin\mathcal{R}}, lb_R$.

Fixing variables within ($spip(\mathcal{R})$)

In order to speed up nodes enumeration, we decided to implement the following *fixing strategy*. Since $\mathcal{R}$ and $\mathcal{R}^{old}$ differ only for the set $R_{k_{new}}$, starting from the optimal solution of ($spip(\mathcal{R})$) with $\mathcal{R} = \mathcal{R}_{old}$, i.e. $(x_{old}, \{y_{old}^j\}_{j=1}^K, \{y_s^{old}\}_{s\notin\mathcal{R}_{old}})$, it is usually sufficient to change at most the value of $y_{old}^{k_{new}}$ to obtain a feasible solution for ($spip(\mathcal{R})$).
Therefore, instead of ($spip(\mathcal{R})$), we address

$$\begin{aligned} \min_{y^{k_{new}}} \; & f(x_{old}) + \sum_{j=1,\; j\neq k_{new}}^{K} \sum_{s\in R_j} p_s g_s(y_{old}^j) + \sum_{s\in R_{k_{new}}} p_s g_s(y^{k_{new}}) + \sum_{s\notin\mathcal{R}} p_s g_s(y_s^{old}) \\ \text{s.t.} \quad & y^{k_{new}} \in Y_{R_{k_{new}}}(x_{old}). \end{aligned} \tag{9}$$

Note that the only variable in (9) is $y^{k_{new}}$, as everything else is fixed. This clearly makes this problem much easier to solve than ($spip(\mathcal{R})$). In case (9) is feasible, its optimal value, that we denote by $V_{\mathcal{R}}^f$, is an upper bound to the optimal value $spip(\mathcal{R})$. This means that additional checks should be considered before pruning, to maintain the correctness of the algorithm. If $V_{\mathcal{R}}^f < UB$, we can proceed to generate the child nodes of the current node. On the other hand, if $V_{\mathcal{R}}^f \ge UB$ or (9) is infeasible, we further solve the original problem ($spip(\mathcal{R})$) to ensure that we can indeed prune node $N(\mathcal{R}, M, d)$ by infeasibility or by bound.

No-good cuts

We have seen in Section 4.2 that, starting from the solution of ($spip(\mathcal{R})$), we can try to build a partition and then an upper bound of (3). Let $(\hat{x}, \{\hat{y}^j\}_{j=1}^K, \{\hat{y}_s\}_{s\notin\mathcal{R}})$ be an optimal solution of problem ($spip(\mathcal{R})$) at node $N(\mathcal{R}, M, d)$.
If $s \in S$ exists such that $sip(s; \mathcal{R}) = \emptyset$ we can show that $(\hat{x}, \{\hat{y}^j\}_{j=1}^K)$ cannot be an optimal solution of (3). In particular, such solution is not feasible for any problem ($pip(\mathcal{P})$).

Proposition 7. *Let $(\hat{x}, \{\hat{y}^j\}_{j=1}^K, \{\hat{y}_s\}_{s\notin\mathcal{R}})$ be an optimal solution for problem ($spip(\mathcal{R})$) induced by a subpartition $\mathcal{R}$. If there exists $\hat{s} \in S$ such that $sip(\hat{s}; \mathcal{R}) = \emptyset$, then $(\hat{x}, \{\hat{y}^j\}_{j=1}^K)$ is not feasible for any problem ($pip(\mathcal{P})$) induced by a partition $\mathcal{P}$ of the scenario set.*

Proof. Let us assume by contradiction that there exists a partition $\mathcal{P} = (P_1, \dots, P_K)$ of the scenario set such that $(\hat{x}, \{\hat{y}^j\}_{j=1}^K)$ is feasible for the problem ($pip(\mathcal{P})$) induced by $\mathcal{P}$. Then, let $\hat{\jmath} \in \{1, \dots, K\}$ be the index such that $\hat{s} \in P_{\hat{\jmath}}$, which exists, since $\mathcal{P}$ is a partition of S. Then, it must be that $y^{\hat{\jmath}} \in Y_{P_{\hat{\jmath}}}(\hat{x})$, since $(\hat{x}, \{\hat{y}^j\}_{j=1}^K)$ is feasible for ($pip(\mathcal{P})$) induced by $\mathcal{P}$. From $Y_{P_{\hat{\jmath}}}(\hat{x}) = \cap_{s \in P_{\hat{\jmath}}} Y_s(\hat{x})$, it holds that $\hat{y}^{\hat{\jmath}} \in Y_{\hat{s}}(\hat{x})$, since $\hat{s} \in P_{\hat{\jmath}}$. Hence, $\hat{\jmath} \in sip(\hat{s}; \hat{x}, \hat{y}^1, \dots, \hat{y}^K)$, which contradicts the assumption of $sip(\hat{s}; \mathcal{R}) = \emptyset$. $\square$

Following Proposition 7, if all first and second stage variables are integer, as soon as we find $s \in S$ such that $sip(s; \mathcal{R}) = \emptyset$, we are allowed to add a no-good-cut to exclude $(x_R, \{y_R^j\}_{j=1}^K)$ from the feasible set of ($spip(\mathcal{R})$). Note that Proposition 7 can be easily adapted considering any feasible solution of problem ($spip(\mathcal{R})$).

5 Computational experiments

In this section, we present the numerical tests we performed to evaluate the effectiveness and efficiency of our approach. As a first set of experiments we considered linear stochastic programs. In fact, while we have presented our method for the general case of non-linear two-stage stochastic programs, we show that for stochastic linear problems the absence of first stage variables allows us to exploit decomposition techniques and to solve smaller problems at each node. Next, we focus on two-stage stochastic linear problems. For these sets of linear instances, besides a complete enumeration approach, where we evaluate all possible partitions of the scenario set, we can compare our partition based method with the performance of the MIQP solver of Gurobi on a compact formulation of (2) that will be presented in the next section.
As a third set of experiments, to assess the ability of the partition based branch-and-bound to deal with nonlinear two-stage stochastic problems, we address instances that involve quadratic objectives. In this case, we only compare to the complete enumeration approach since no other methods for this case exist. For both the linear and quadratic problems, we considered randomly generated instances, that turned out to be challenging to solve even for a small number of scenarios ℓ.
Note that, in the presence of constraint uncertainty, the K-adaptability problem may not be feasible even if an optimal solution of the underlying two-stage stochastic problem exists. In Malaguti et al. (2025), to deal with the possible infeasibility of the K-adaptability problem it is assumed that a dummy policy with infinitely large cost always exists. Here, instead, for the two-stage stochastic linear instances, we specifically study the case in which such dummy policy does not exist and therefore we cannot guarantee the feasibility of the K-adaptability problem for values of $K > 1$. On this set of instances we tested the ability of our partition-based method to detect infeasibility, with respect to the MIQP solver of Gurobi.
In Table 1, we report a summary of the tests performed, grouped by instance characteristics. Finally, in Figure 3, we also compare the performance of our partition-based method and the MIQP solver of Gurobi for two-stage stochastic linear instances corresponding to larger scenario sets.
We included in `PiM` all the improvements described in the previous section. In particular, based on preliminary tuning experiments (not reported here for brevity), we set the time limit at each node to $t_{max} = 1s$. All instances considered next have binary first and second stage variables, therefore we were able to generate no-good-cuts, which were added as lazy-constraints to the `Gurobi` model. While, on one hand they help in finding better valid lower bounds at each node, on the other hand they may significantly increase the size of the problem. In order to reach a trade off, we decided to only consider a subset of no-good-cuts at each node.
In all cases, the scenarios are assumed to be equiprobable, in particular $p_s = \frac{1}{\ell}$, for each $s \in S$, $|S| = \ell$.
The code for our approach is written in `Python3` and is available at https://github.com/dfede3/KPiM-2SS. All experiments were conducted on a computer with the following characteristics: Ubuntu 22.04 OS, Intel Xeon Processor E5–2430 v2 (6 cores, 2.50 GHz), 32 GB of RAM. For all the linear and quadratic integer programming problems we used Gurobi version 12.0.1.

Table 1: Tests Performed According to Instance Characteristics

Table	**1st stage variables**	**Obj. fun. uncertainty**	**Constr. uncertainty**	**Linear obj. fun.**	**Quadratic obj. fun.**	(k-2SSP) **feasible for** $K = 1$
2	✗	✓	✓	✓	✗	✓
3	✓	✓	✓	✓	✗	✓
4	✓	✓	✓	✓	✗	✗
5	✓	✓	✓	✗	✓	✗

5.1 Linear stochastic and two-stage stochastic problems

In this section, we consider instances with $f(x) = c^\top x$ and $g_s(y) = q_s^\top y$ for some vectors $c \in \mathbb{R}^{n_x}$, $q_s \in \mathbb{R}^{n_y}$ for $s \in \{1, \dots, \ell\}$. The mixed-integer quadratic (MIQP) formulation of the K-adaptability problem for linear stochastic programs proposed in Malaguti et al. (2022) can be extended to the case of two-stage stochastic linear programs. Let us recall that a scenario s is *covered* by the j-th solution $y^j \in \{y^1, \dots, y^K\}$, if y^j is feasible for scenario s and it is selected among those that are feasible for that scenario (Buchheim and Pruente 2019, Malaguti et al. 2022).
Then, the following additional binary variables can be defined:

$$u_{sj} = \begin{cases} 1 & \text{if scenario } s \text{ is } \textit{covered} \text{ by solution } y^j, \\ 0 & \text{otherwise,} \end{cases} \qquad s = 1, \dots, \ell,\ j = 1, \dots, K.$$

The quadratic formulation of the K-adaptability two-stage stochastic linear problem can be written as follows:

$$\min\ c^\top x + \sum_{s=1}^{\ell} p_s q_s^\top \Big(\sum_{j=1}^{K} y^j u_{sj} \Big) \tag{10}$$

$$\text{s.t. } x \in X, \tag{11}$$

$$\sum_{j=1}^{K} u_{sj} = 1, \quad s = 1, \dots, \ell, \tag{12}$$

$$u_{sj} = 1 \quad \Rightarrow \quad y^j \in Y_s(x), \quad s = 1, \dots, \ell,\ j = 1, \dots, K, \tag{13}$$

$$u_{sj} \in \{0, 1\}, \quad s = 1, \dots, \ell,\ j = 1, \dots, K. \tag{14}$$

Constraints (12) impose that each scenario is covered by exactly one of the second-stage solutions, whose cost is taken into account in the objective function (10). Constraints (13) guarantee that each scenario is covered by a solution that is feasible for that scenario. In Malaguti et al. (2022), different ways for handling this kind of constraints are presented. In our computational experiments we relied on the so-called indicator constraints of `Gurobi`.
The only differences with the compact formulation in Malaguti et al. (2022) are that in the objective function we add the cost of first stage decisions, and in constraint (13) the feasible set $Y_s(x)$ also depends on the value of first stage variables x.

5.1.1 Linear stochastic problems

Our partition based method can be directly applied to solve linear stochastic problems. However, in the absence of first stage variables, the subpartition induced problem at a node can be decomposed into independent smaller problems, one for each element of the subpartition and one for each scenario outside of the subpartition.
Formally, the problem induced by a subpartition $\mathcal{R}$ becomes the following:

$$\begin{aligned} \min\ & \sum_{j=1}^{K} \sum_{s \in R_j} p_s g_s(y^j) + \sum_{s \notin \mathcal{R}} p_s g_s(y_s) \\ \text{s.t.}\quad & y^j \in Y_{R_j}, \quad j = 1, \dots, K, \\ & y_s \in Y_s, \quad s \notin \mathcal{R}, \end{aligned}$$

and it can be decomposed into independent smaller problems,

$$\begin{aligned} \min_{y^j} \ & \sum_{s \in R_j} p_s g_s(y^j) \\ \text{s.t. } & y^j \in Y_{R_j}, \end{aligned} \tag{15}$$

one for each $j \in \{1, \dots, K\}$ and

$$\begin{aligned} \min_{y_s} \ & p_s g_s(y_s) \\ \text{s.t. } & y_s \in Y_s, \end{aligned} \tag{16}$$

one for each $s \notin \mathcal{R}$.
This decomposition is particularly suitable to be combined with our partition based method. Indeed, problems (16) can be solved once at the root node, to collect their optimal value and optimal solution that can then be reused at every other node. Regarding problems (15), at each node we only have to solve the one corresponding to the value of j such that the current node $\mathcal{R}$ was obtained from its parent node by adding a new scenario in R_j. In fact, since all the other elements of $\mathcal{R}$ are unchanged with respect to its parent node, there is no need to reoptimize the corresponding problems (15).

As a test case, we consider stochastic knapsack instances generated as in Malaguti et al. (2022):

- *knapsack problem*: in each scenario, the profits and weights of the n_y items were randomly generated as independent uniformly distributed values between 0.0 and 1.0. The capacity was set to $0.75W$, where W is the sum of the weights of the items for that scenario. All values were rounded to three digits.

We considered $n_y = 10$ items in the knapsack and scenarios sets of size $\ell \in \{10, 15, 20\}$, while $K \in \{2, 3, 4, 5\}$. For every combination of ℓ and K we generated 5 random instances that were executed with a time limit of 30 minutes each. Table 2 shows the comparison among `PiM`, the complete enumeration approach (`CE`), the MIQP solver of Gurobi applied to the compact formulation (`MIQP`) and the partition-based method that we obtain by exploiting the decomposable structure of the stochastic problem, which we call `PiMd`. For each combination of ℓ and K, we report t, the average time (in seconds) with respect to the instances solved to optimality within the time limit of half an hour; TL, the number of instances (out of 5) for which the algorithm hit the time limit. For `PiM` and `PiMd` we also report # nodes, the total number of nodes explored and # leaves, the total number of leaves visited. Between parenthesis we show the corresponding percentage with respect to $\mathcal{S}_{\ell,K}$.

Table 2: Stochastic knapsack problem, $n_y = 10$.

K	ℓ	`CE`		`PiM`				`PiMd`				`MIQP`	
		t	TL	t	# nodes	# leaves	TL	t	# nodes	# leaves	TL	t	TL
2	10	7.8	0	3.3	298	24 (4.8%)	0	**1.6**	353	28 (5.5%)	0	**1.6**	0
3	10	143.0	0	6.7	683	17 (0.2%)	0	3.2	833	33 (0.4%)	0	**1.9**	0
4	10	433.1	0	6.5	735	28 (<0.1%)	0	5.0	1305	51 (0.1%)	0	**1.6**	0
5	10	466.0	0	6.6	741	41 (<0.1%)	0	4.1	1066	96 (0.2%)	0	**1.2**	0
2	15	565.9	0	83.2	3626	30 (0.2%)	0	50.0	5554	95 (0.6%)	0	**29.0**	0
3	15	-	5	424.1	23349	39 (<0.1%)	0	252.1	41130	241 (<0.1%)	0	**198.7**	0
4	15	-	5	729.4	47921	82 (<0.1%)	0	**520.3**	89506	735 (<0.1%)	0	631.9	2
5	15	-	5	**655.4**	46635	25 (<0.1%)	0	750.6	141722	722 (<0.1%)	0	1281.4	3
2	20	-	5	1191.4	32180	8 (<0.1%)	3	1258.1	77020	121 (<0.1%)	0	**282.9**	0
3	20	-	5	-	-	-	5	-	-	-	5	**543.3**	0
4	20	-	5	-	-	-	5	-	-	-	5	-	5
5	20	-	5	-	-	-	5	-	-	-	5	-	5

Results in Table 2 show that our partition based methods always outperform `CE` both with respect to average running time and number of instances solved before hitting the time limit. Overall, the computational performance of `MIQP` and `PiM` methods is comparable, yet `MIQP` is generally faster across most of the configurations considered. Comparing `PiM` with `PiMd`, it is evident that the time required to solve a subpartition induced problem at a node is significantly reduced in the latter: even if many more nodes are visited, the total running time is generally smaller than what we obtain with `PiM`. The increase in number of nodes can be explained by the fact that in `PiMd` we solve a smaller independent problem at each node, therefore, we cannot include the no-good-cuts that help `PiM` obtain tighter lower bounds at each node.

5.1.2 Linear two-stage stochastic problems

For the first set of computational experiments on linear two-stage stochastic programs we considered the knapsack problem with setup and the capacitated facility location problem. The respective formulations are reported in the Appendix.
The instances are generated as follows:

- *knapsack problem with setup:* in each scenario, the profits and weights of the n_y items were randomly generated as independent uniformly distributed values between 0.0 and 100.0 for profits and between 10.0 and 100.0 for weights. The knapsack capacity for a scenario was set to 0.5 W, where W denotes the sum of the item weights in that scenario. Items were partitioned into n_x classes, and for each class the setup parameters (profit and weight) were computed as 20% of the average profit and weight over all scenarios of the items in that class. All values were rounded to three digits;

- *capacitated facility location:* the fixed opening costs of the n_x facilities were randomly generated as uniformly distributed values between 100.0 and 1000.0, and the capacities between 50.0 and 500.0. In each scenario, the service costs and demands of each of the n_y customers were independently drawn from uniform distributions between 0.0 and 100.0, and 1.0 and 100.0, respectively. All values were rounded to the nearest integer.

For both problems we considered $n_x = 5$ first-stage variables and $n_y = 10$ second-stage variables. Concerning the number of scenarios we used $\ell \in \{10, 15, 20\}$ while $K \in \{2, 3, 4, 5\}$. For each combination of parameters K and ℓ we randomly generated 5 instances. Each instance was executed with a time limit of 30 minutes. The values reported in Table 3 are the same as the ones described for Table 2.

Table 3: Results for the two-stage stochastic knapsack problem with setup and the two-stage stochastic capacitated facility location for $n_x = 5$ and $n_y = 10$.

		Knapsack with setup								Capacitated facility location							
K	**ℓ**	`CE`		`PiM`				`MIQP`		`CE`		`PiM`				`MIQP`	
		t	TL	t	# nodes	# leaves	TL	t	TL	t	TL	t	# nodes	# leaves	TL	t	TL
2	10	18.5	0	8.2	180	8 (1.6%)	0	**5.8**	0	47.5	0	19.9	343	49 (9.7%)	0	**18.8**	0
3	10	441.1	0	36.7	618	31 (0.3%)	0	**26.9**	0	699.2	1	144.4	1917	94 (1.0%)	0	**62.4**	0
4	10	1355.1	4	**55.2**	898	50 (0.1%)	0	64.6	0	-	5	323.6	3171	222 (0.7%)	0	**162.5**	0
5	10	-	5	**42.2**	661	52 (0.1%)	0	104.3	0	-	5	**209.3**	2770	280 (0.7%)	0	247.8	0
2	15	783.1	0	43.9	626	2 (<0.1%)	0	**25.5**	0	1617.7	3	400.8	4795	82 (0.5%)	0	**18.0**	0
3	15	-	5	527.8	7038	25 (<0.1%)	1	**438.1**	0	-	5	1263.9	15381	57 (<0.1%)	4	**285.2**	1
4	15	-	5	**847.3**	9263	74 (<0.1%)	3	-	5	-	5	-	-	-	5	**1171.6**	3
5	15	-	5	**1016.9**	11223	112 (<0.1%)	3	-	5	-	5	**278.2**	1521	125 (<0.1%)	4	581.0	3
2	20	-	5	273.5	2900	4 (<0.1%)	0	**60.4**	0	-	5	1560.1	10230	6 (<0.1%)	4	**345.8**	0
3	20	-	5	-	-	-	5	**756.8**	0	-	5	-	-	-	5	**268.1**	4
4	20	-	5	-	-	-	5	-	5	-	5	-	-	-	5	-	5
5	20	-	5	-	-	-	5	-	5	-	5	-	-	-	5	-	5

Results in Table 3 show that for the two-stage knapsack problem `PiM` has good performance in most settings. For a fixed ℓ, the partition based method outperforms `MIQP` for larger values of K, for which the size of the tree grows, showing that the bounds obtained at each node are good and a lot of nodes can be pruned. For the facility location instances the performance of `PiM` is slightly worse, however we can still observe a similar trend, with `PiM` having a better average running time for $K = 5$.

Observe that for all the previous instances of the stochastic knapsack problem with setup and the stochastic capacitated facility problem there always exists a solution that is feasible for all scenarios, i.e. the K-adaptability problem with $K = 1$ is always feasible. Now, we test `PiM` on two-stage stochastic linear

Table 4: Two-stage stochastic knapsack problem with setup and capacitated facility location problem.

K	ℓ	n_y	**Knapsack with setup**								**Capacitated facility location**							
			PiM					**MIQP**			**PiM**					**MIQP**		
			t_o	t_i	# nodes	# leaves	TL	t_o	t_i	TL	t_o	t_i	# nodes	# leaves	TL	t_o	t_i	TL
2	10	10	-	**3.6** (5)	42	0	0	-	4.9 (5)	0	-	**4.6** (5)	23	0	0	-	9.5 (5)	0
3	10	10	38.2 (5)	-	468	9 (<0.1%)	0	**25.7** (5)	-	0	**91.7** (1)	**16.4** (4)	159	1 (<0.1%)	0	96.5 (1)	86.7 (4)	0
4	10	10	**94.1** (5)	-	946	18 (<0.1%)	0	**94.1** (5)	-	0	174.0 (4)	**49.3** (1)	771	28 (<0.1%)	0	**110.5** (4)	703.6 (1)	0
5	10	10	**157.5** (5)	-	1433	71 (0.2%)	0	219.1 (5)	-	0	239.7 (5)	-	1310	195 (0.5%)	0	**238.5** (4)	-	1
2	15	10	-	**5.7** (5)	42	0	0	-	7.2 (5)	0	-	**9.2** (5)	25	0	0	-	17.2 (5)	0
3	15	10	-	85.7 (5)	648	0	0	-	**59.3** (5)	0	-	**81.1** (5)	233	0	0	-	227.5 (5)	0
4	15	10	**616.8** (3)	-	4804	10 (<0.1%)	2	1257.1 (3)	-	2	-	**245.6** (4)	916	0	1	**1255.3** (1)	649.2 (3)	1
5	15	10	**995.1** (1)	-	6841	5 (<0.1%)	4	-	-	5	**134.3** (1)	**651.5** (1)	3105	2 (<0.1%)	3	913.9 (1)	-	4
2	20	10	-	9.9 (5)	34	0	0	-	**5.1** (5)	0	-	**8.9** (5)	18	0	0	-	24.3 (5)	0
3	20	10	-	106.7 (5)	472	0	0	-	**92.7** (5)	0	-	**49.6** (5)	105	0	0	-	101.2 (5)	0
4	20	10	-	944.0 (3)	4915	0	2	-	**825.1** (3)	2	-	**328.7** (5)	773	0	0	-	608.1 (4)	1
5	20	10	-	-	-	-	5	-	-	5	-	**667.3** (2)	4272	0	3	-	970.7 (1)	4
2	10	15	**35.6** (5)	-	251	13 (2.6%)	0	191.0 (5)	-	0	**236.4** (1)	**40.8** (4)	140	13 (2.7%)	0	-	559.1 (4)	1
3	10	15	208.4 (5)	-	1024	28 (0.3%)	0	**155.1** (5)	-	0	**1218.2** (3)	-	2458	707 (7.6%)	2	-	-	5
4	10	15	**282.4** (5)	-	1432	60 (0.2%)	0	287.9 (5)	-	0	1626.5 (1)	-	3017	607 (1.8%)	4	**1201.8** (1)	-	4
5	10	15	**260.7** (5)	-	1370	81 (0.2%)	0	402.4 (5)	-	0	-	-	-	-	5	**1354.7** (1)	- (0)	4
2	15	15	-	**76.3 (5)**	322	0	0	-	361.6 (5)	0	-	**117.3** (5)	147	0	0	-	493.9 (3)	2
3	15	15	-	-	-	-	5	-	-	5	-	**1045.7** (1)	3353	0	4	-	-	5
4	15	15	-	-	-	-	5	-	-	5	-	-	-	-	5	-	-	5
5	15	15	-	-	-	-	5	-	-	5	-	-	-	-	5	-	-	5
2	20	15	-	**152.6** (5)	357	0	0	-	941.7 (5)	0	-	**180.5** (5)	206	0	0	-	188.8 (4)	1
3	20	15	-	-	-	-	5	-	-	5	-	**579.5** (1)	1202	0	4	-	-	5
4	20	15	-	-	-	-	5	-	-	5	-	-	-	-	5	-	-	5
5	20	15	-	-	-	-	5	-	-	5	-	-	-	-	5	-	-	5

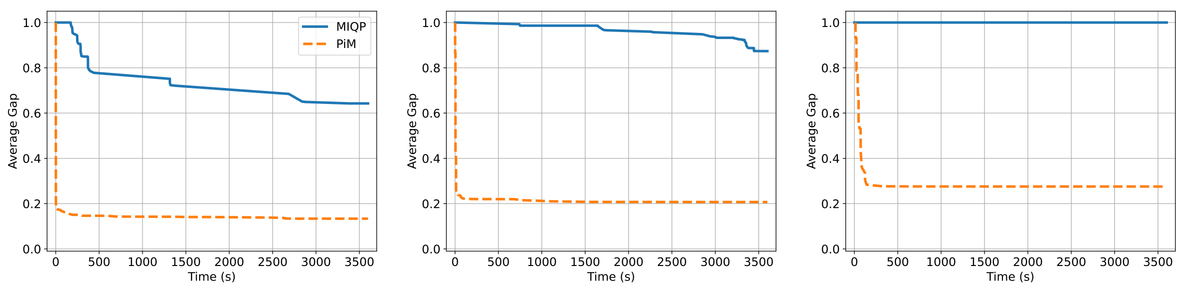
instances such that the K-adaptability approach is not necessarily feasible for $K = 1$. In particular, we consider the previous instances to which we now add a constraint requiring that a minimum capacity must also be filled:

- *knapsack problem with setup:* same as above. A minimum capacity equal to 95% of the total capacity is also required to be filled for each scenario;
- *capacitated facility location:* same as above. A minimum capacity equal to the 80% of the total capacity for that scenario is also considered for each facility.

For this set of instances we do not show the results obtained with `CE`, since they do not differ much from the ones obtained in Table 3. In particular, the values in Table 4 are the same as the ones from the previous tables, except instead of t we have the following two values:

- t_o, the average time (in seconds) with respect to the instances solved to optimality within the time limit,
- t_i, the average time (in seconds) with respect to the instances that proven to be infeasible within the time limit.

In the columns corresponding to t_o and t_i we also report between parenthesis the number of instances on which this average is computed, whenever this number is different from 0.
In Table 4 we observe that for the knapsack instances, `PiM` is the best approach for $K = 2$, and often also for $K = 4$ and $K = 5$. When $\ell = 20$ and $n_y = 10$, the compact formulation is faster for all values of K. Observe that for $n_y = 15$ whenever the time limit is not reached, `PiM` is usually the faster method for either detecting infeasibility or proving optimality. For $n_y = 10$ this in not always the case.
For the facility location instances the compact formulation has good performances, but `PiM` always clearly outperforms it, either by having a smaller computation time or by being able to solve more instances before hitting the time limit (the only exception being $n = 15$, $\ell = 10$ and $K = 4, 5$). For this problem we can see that `PiM` is by far the best approach to prove infeasibility in terms of average running time.
Finally, let us compare Table 4 with the results in Table 3. Average running times for instances solved to optimality are comparable for both `MIQP` and `PiM`. In the second table, however, we can notice a significant reduction in the total number of nodes and leaves explored in the subpartition-induced tree by `PiM`. A major contributing factor is that, for this second set of instances, the subpartition induced problem at a node is not always feasible, thus allowing us to prune earlier unpromising parts of the tree.



(a) $\ell = 25,\ K = 5$. (b) $\ell = 50,\ K = 5$. (c) $\ell = 100,\ K = 5$.

Figure 3: Comparison of the average gap over time obtained by `MIQP` and `PiM` on 5 instances of the original two-stage stochastic knapsack problem with setup for different sizes of the scenario set.

5.1.3 Large instances of linear stochastic problems

The previous results show that the K-adaptability problem is harder to solve for all algorithms when the number ℓ of scenarios we are considering increases. Here, we consider additional randomly generated instances of larger size than those addressed in the previous experiments. In particular, we solve instances of the original (i.e. without the additional constraints) two-stage stochastic knapsack problem with setup with parameters $n_x = 5,\ n_y = 10,\ K = 5$ and $\ell \in \{25, 50, 100\}$. For these experiments we tested `MIQP` and `PiM` over 5 instances, with a time limit of one hour per instance, while the time limit at a node for `PiM` was increased to 5 seconds. We do not report the results in a table since both algorithms always reach the time limit for all the instances of the different combinations of parameters considered. Instead, we show in Figure 3 how the average gaps progress over time. In particular, for each instance we compute the gap at time t as $\min\left\{\frac{ub_t - lb_t}{ub_t}, 1\right\}$ where ub_t is the best incumbent upper bound at time t and lb_t is the best known lower bound at time t. We compute the gap at each time t when the best incumbent upper bound is updated.

Observe that for `PiM` the lower bound lb_t is given by the optimal objective function value found by solving the root node, i.e. $lb_t = spip(\mathcal{R}^0) = \text{opt}(\ell)$ for every t. In order to obtain the exact solution of the root node, we do not impose the time limit of 5 seconds in that node. However, any lower bound on the solution of the root node is also a valid lower bound for the K-adaptability problem and could, in principle, be considered to compute this gap. On the other hand, the value of lb_t that we consider for `MIQP` is the value of the best bound given by `Gurobi` at any time t when the best incumbent upper bound is updated.

From the plots in Figure 3 it is evident that `PiM` always finds good upper-bounds quickly and is able to reach small gaps in short time. Note that, for our choice of lower bounds for `PiM`, whenever K is such that $\text{opt}(\ell) < \text{opt}(K)$, a gap equal to 0 can never be obtained. Furthermore, we can see that the overall gap grows as we increase the size of the scenario set, as the error we make by approximating the underlying stochastic program with ℓ scenarios using K-adaptability increases as we augment ℓ while maintaining the same value of $K = 5$. `MIQP` can, in theory, obtain a gap equal to zero, however, for any time t, it finds worse gaps than `PiM`. The overall quality of these gaps decreases as ℓ grows, while `PiM` is not as affected by the value of ℓ. In particular, for $\ell = 100$, `MIQP` is never able to find upper and lower bounds such that a meaningful gap lower than 1 can be reached before the time limit of 1 hour.

In conclusion, even if for large instances `PiM` is not able to prove the optimality of a solution before the time limit, it still provides us with useful information, like provably good upper bounds, in a reasonable time, unlike `MIQP`.

5.2 Quadratic two-stage stochastic problems

For the final computational experiments on quadratic two-stage stochastic programs we considered the quadratic counterparts of the problems considered in the previous section. As before, the complete formulations are reported in the Appendix.

The instances are generated as follows:

- *quadratic knapsack problem with setup:* quadratic cost coefficients were independently drawn from a uniform distribution between 0.0 and 100.0. The rest of the values for each scenario were generated as in the (second) linear knapsack case;

- *quadratic capacitated facility location:* for each pair of customers served by the same facility an extra cost equal to 25.0 was added to the objective function in each scenario. The rest of the values for each scenario were generated as in the (second) linear facility location case.

The values considered for n_x, ℓ and K are chosen as for the linear instances, while n_y is fixed to 10. Note that for these instances we no longer make the distinction between t_o and t_i, instead we indicate with t the average time with respect to the instances addressed within the time limit of half an hour, detecting either an optimal solution or infeasibility.
As already observed, for the quadratic instances we can only compare the performance of `PiM` to the complete enumeration approach `CE`. Table 5 clearly shows that the partition-based branch-and-bound is the best approach since it only visits a small percentage of the total number of the nodes and the leaves, even for bigger values of ℓ and K, allowing this method to solve more instances in a smaller amount of time.

Table 5: Two-stage stochastic quadratic knapsack problem with setup and quadratic capacitated facility location problem, $n_x = 5$ and $n_y = 10$.

		Quadratic knapsack						**Quadratic facility location**					
		CE		**PiM**				**CE**		**PiM**			
K	ℓ	t	TL	t	# nodes	# leaves	TL	t	TL	t	# nodes	# leaves	TL
2	10	66.2	0	**23.6**	41	0	0	76.8	0	**40.7**	48	0	0
3	10	1656.3	0	**192.0**	465	9 (<0.1%)	0	1109.1	3	**210.0**	294	0	1
4	10	-	5	**846.5**	1788	68 (0.2%)	0	-	5	**209.0**	721	42 (<0.1%)	3
5	10	-	5	**1314.8**	2528	247 (0.3%)	3	-	5	**856.0**	2312	630 (0.7%)	3
2	15	-	5	**36.3**	44	0	0	-	5	**192.0**	203	0	0
3	15	-	5	**602.8**	828	0	0	-	5	**73.2**	141	0	3
4	15	-	5	-	-	-	5	-	5	**517.7**	1136	0	3
5	15	-	5	-	-	-	5	-	5	**1654.2**	6378	40 (<0.1%)	4
2	20	-	5	**75.5**	81	0	0	-	5	**222.9**	240	0	0
3	20	-	5	**1265.4**	1475	0	2	-	5	**143.4**	226	0	3
4	20	-	5	-	-	-	5	-	5	**553.6**	1516	0	4
5	20	-	5	-	-	-	5	-	5	-	-	-	5

6 Conclusions

We study two-stage stochastic programs in which uncertainty is represented by a finite set of scenarios, and we introduce and analyze the K-adaptability approach for this type of problems. We derive bounds on the approximation guarantee the K-adaptability approach attains regarding the two-stage stochastic problem. Furthermore, we derive and discuss conditions under which the original two-stage problem is equivalent to the K-adaptability. A branch-and-bound method is devised to exactly solve the partition-based reformulation of the K-adaptability problem. The flexibility offered by addressing the subpartition-induced problem enables the method to handle a very general class of two-stage stochastic programs, not necessarily linear and allowing uncertainty to appear both in the objective function and in the constraints. To the best of our knowledge, this is the first time such a general framework has been presented. The subpartition-induced problem plays a central role in defining pruning rules and guiding scenario selection during node generation. This generation process is designed to avoid exploring symmetric partitions as well as partitions containing empty sets. Numerical experiments demonstrate the strong performance of the proposed method, which is able to enumerate only a negligible fraction of all possible partitions in order to identify the optimal one. Furthermore, in the special case of linear instances - such as stochastic knapsack and facility location - where a mixed-integer quadratic reformulation for the K-adaptability problem is available, the proposed method proves competitive with the MIQP solver of GUROBI.
As our experiments show solving the K-adaptability problem for small values of K is very challenging. Hence, future research should be conducted to improve the performance of the method. Fast heuristic

methods could be designed which provide better primal bounds in the branch and bound procedure. Furthermore, taking advantage of problem-specific structures would help in speeding up the method in applications to real life problems with many scenarios. The introduction of decomposition techniques, like Benders decomposition, in the partition-based method should also be investigated, since in the single-stage case they greatly improve the overall running time.

References

Ayşe N Arslan, Michael Poss, and Marco Silva. Min-sup-min robust combinatorial optimization with few recourse solutions. *INFORMS Journal on Computing*, 34(4):2212–2228, 2022.

Dimitris Bertsimas and Constantine Caramanis. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766, 2010.

John R Birge and Francois Louveaux. *Introduction to stochastic programming.* Springer, 1997.

Christoph Buchheim and Jannis Kurtz. Min–max–min robust combinatorial optimization. *Mathematical Programming*, 163(1):1–23, 2017.

Christoph Buchheim and Jannis Kurtz. Complexity of min–max–min robustness for combinatorial optimization under discrete uncertainty. *Discrete Optimization*, 28:1–15, 2018.

Christoph Buchheim and Jonas Pruente. K-adaptability in stochastic combinatorial optimization under objective uncertainty. *European Journal of Operational Research*, 277(3):953–963, 2019.

André Chassein and Marc Goerigk. On the complexity of min–max–min robustness with two alternatives and budgeted uncertainty. *Discrete Applied Mathematics*, 296:141–163, 2021.

André Chassein, Marc Goerigk, Jannis Kurtz, and Michael Poss. Faster algorithms for min-max-min robustness for combinatorial problems with budgeted uncertainty. *European Journal of Operational Research*, 279(2): 308–319, 2019.

Xiaojun Chen, Alexander Shapiro, and Hailin Sun. Convergence analysis of sample average approximation of two-stage stochastic generalized equations. *SIAM Journal on Optimization*, 29(1):135–161, 2019.

Lars Eufinger, Jannis Kurtz, Christoph Buchheim, and Uwe Clausen. A robust approach to the capacitated vehicle routing problem with uncertain costs. *INFORMS Journal on Optimization*, 2(2):79–95, 2020.

Laura Galli, Silvano Martello, Carlos Rey, and Paolo Toth. The quadratic knapsack problem with setup. *Computers & Operations Research*, 173:106873, 2025.

Alireza Ghahtarani, Ahmed Saif, Alireza Ghasemi, and Erick Delage. A double-oracle, logic-based benders decomposition approach to solve the k-adaptability problem. *Computers & Operations Research*, 155: 106243, 2023.

Marc Goerigk, Jannis Kurtz, and Michael Poss. Min–max–min robustness for combinatorial problems with discrete budgeted uncertainty. *Discrete Applied Mathematics*, 285:707–725, 2020.

Eojin Han, Chaithanya Bandi, and Omid Nohadani. On finite adaptability in two-stage distributionally robust optimization. *Operations Research*, 71(6):2307–2327, 2023.

Grani A Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. K-adaptability in two-stage robust binary programming. *Operations Research*, 63(4):877–891, 2015.

Grani A Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. K-adaptability in two-stage distributionally robust binary programming. *Operations Research Letters*, 44(1):6–11, 2016.

Esther Julien, Krzysztof Postek, and Ş İlker Birbil. Machine learning for k-adaptability in two-stage robust optimization. *INFORMS Journal on Computing*, 37(3):644–665, 2025.

Daniel Kuhn, Peyman Mohajerin Esfahani, Viet Anh Nguyen, and Soroosh Shafieezadeh-Abadeh. Wasserstein distributionally robust optimization: Theory and applications in machine learning. In *Operations research & management science in the age of analytics*, pages 130–166. Informs, 2019.

Jannis Kurtz. Approximation guarantees for min-max-min robust optimization and-adaptability under objective uncertainty. *SIAM Journal on Optimization*, 34(2):2121–2149, 2024a.

Jannis Kurtz. Bounding the optimal number of policies for robust k-adaptability. *arXiv preprint arXiv:2409.12630*, 2024b.

Enrico Malaguti, Michele Monaci, and Jonas Pruente. K-adaptability in stochastic optimization. *Mathematical Programming*, 196(1):567–595, 2022.

Enrico Malaguti, Michele Monaci, and Jonas Pruente. Heuristic algorithms for k-adaptability. *Available at SSRN 5266670*, 2025.

Sophie Michel, Nancy Perrot, and François Vanderbeck. Knapsack problems with setups. *European Journal of Operational Research*, 196(3):909–918, 2009.

Zihang Qiu, Ali Ajdari, Mislav Bobić, Thomas Bortfeld, Dick den Hertog, Jannis Kurtz, and Hoyeon Lee. A k-adaptability approach to proton radiation therapy robust treatment planning. *arXiv preprint arXiv:2508.07368*, 2025.

Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. *Lectures on stochastic programming: modeling and theory.* SIAM, 2021.

Anirudh Subramanyam, Chrysanthos E Gounaris, and Wolfram Wiesemann. K-adaptability in two-stage mixed-integer robust optimization. *Mathematical Programming Computation*, 12:193–224, 2020.

Bart PG Van Parys, Peyman Mohajerin Esfahani, and Daniel Kuhn. From data to decisions: Distributionally robust optimization is optimal. *Management Science*, 67(6):3387–3402, 2021.

Paula Weller and Fabricio Oliveira. Streamlining emergency response: A k-adaptable model and a column-and-constraint-generation algorithm. *European Journal of Operational Research*, 324(3):925–940, 2025.

Feifeng Zheng and Yang Sui. K-adaptability in robust container vessel sequencing problem in a route with fluctuating demand. In *2019 International Conference on Industrial Engineering and Systems Management (IESM)*, pages 1–6. IEEE, 2019.

A Proof of Proposition 4

To prove Proposition 4 we need to first state the following additional result.

Proposition 8. *Let $\mathcal{P} = (P_1, \dots, P_K)$ be a partition of S of order $M \le K$, $K \le |S|$. Then a partition $\widetilde{\mathcal{P}} = (\tilde{P}_1, \dots, \tilde{P}_K)$ of S of order K exists such that $\forall\ j = 1, \dots, K$ a unique $i \in \{1, \dots, M\}$ exists such that $\tilde{P}_j \subseteq P_i$.*

Proof. Proof W.l.o.g. we can assume that the elements of $\mathcal{P}$ are ordered so that $|P_1| \ge |P_2| \ge \ldots \ge |P_M|$. Let $N = \max\{k \in \{1, \dots, M\} | |P_k| > 1\}$. Then, the following equality holds,

$$\begin{aligned}
\sum_{j=1}^{N}(|P_j|-1) \quad &= \sum_{j=1}^{N}|P_j| - N \\
&= \sum_{j=1}^{M}|P_j| - \sum_{j=N+1}^{M}|P_j| - N \\
&= \sum_{j=1}^{M}|P_j| - \sum_{N+1}^{M} 1 - N \\
&= |S| - (M-N) - N \\
&= |S| - M \ge K - M,
\end{aligned}$$

where the last inequality derives from the assumption $|S| > K$. Next, we define

$$u = \min_{i \in \{1,\dots,N\}} \left\{ \sum_{j=1}^{i}(|P_j|-1) \ge K - M \right\}.$$

Let $p = \sum_{j=1}^{u-1}(|P_j|-1)$ and let $r = K - M - p$, $(r \le |P_u| - 1)$. Finally, let
$T = \{s | s \in P_j, j = \{1, \dots, u-1\}\} \cup \{s^{P_u}(1), \dots, s^{P_u}(r)\}$, where with $s^{P_u}(j)$ we indicate the jth element of the set P_u.
Observe that $|T| = \sum_{j=1}^{u-1}|P_j| + |\{s^{P_u}(1), \dots, s^{P_u}(r)\}| = p + (u-1) + r = u - 1 + K - M$.
Now, we define

$$\begin{aligned}
\tilde{P}_j &= P_{u+j} \quad j = 1, \dots, M-u, \\
\tilde{P}_{M-u+1} &= P_u \setminus \{s^{P_u}(1), \dots, s^{P_u}(r)\}, \\
\tilde{P}_{M-u+1+j} &= s^T(j), \quad j = 1, \dots, u-1+K-M.
\end{aligned}$$

Note that the index $M-u+1+j$ corresponding to $j=u-1+K-M$ is $M-u+1+u-1+K-M=K$, and $\tilde{P}_{M-u+1}\neq\emptyset$ since $|P_u|>r$, i.e. $\widetilde{\mathcal{P}}=(\tilde{P}_1,\ldots,\tilde{P}_K)$ has K non empty elements and by construction it is a partition of S, i.e. it is a partition of order K and by construction it satisfies the assumption that for every $j\in\{1,\ldots,K\}$ a unique $i\in\{1,\ldots,M\}$ exists such that $\tilde{P}_j\subseteq P_i$. □

We are now able to report the proof of Proposition 4:

Proof. Proof Let $\mathcal{P}$ be an optimal solution of (3). Let $(x_P,y_P^1,\ldots,y_P^K)$ be the optimal solution of $pip(\mathcal{P})$. Let M be the order of $\mathcal{P}$, i.e. $M=|\{j\in\{1,\ldots,K\}|P_j\neq\emptyset\}|$. Otherwise, if $M=K$, then condition 1 holds. If $M<K$, then let $\widetilde{\mathcal{P}}$ be the partition of order K such that for all $j\in\{1,\ldots,K\}$, $i(j)\in\{1,\ldots,M\}$ exists such that

$$\tilde{P}_j\subseteq P_{i(j)}, \tag{17}$$

which always exists according to Proposition 8. Let us prove now that $pip(\mathcal{P})=pip(\widetilde{\mathcal{P}})$.
Let us set $x=x_P$ and $y^j=y_P^{i(j)}$ where for every $j\in\{1,\ldots,K\}$, $i(j)$ is the index that satisfies (17). Observe that $(x,y^1,\ldots,j^K)$ is feasible for $pip(\widetilde{\mathcal{P}})$, and

$$\begin{aligned} pip(\widetilde{\mathcal{P}}) &\leq f(x)+\textstyle\sum_{j=1}^K\sum_{s\in\tilde{P}_j}p_sg_s(y_P^j)\\ &=f(x_P)+\textstyle\sum_{j=1}^K\sum_{s\in\tilde{P}_j}p_sg_s(y_P^{i(j)})\\ &=f(x_P)+\textstyle\sum_{i=1}^M\sum_{s\in P_i}p_sg_s(y_P^i)\\ &=pip(\mathcal{P}),\end{aligned}$$

where the second to last equality holds since $P_i=\{\cup\tilde{P}_j|i(j)=i,\ j\in\{1,\ldots,K\}\}$ and if $i(j_1)=i(j_2),\ j_1,j_2\in\{1,\ldots,K\}$ then $y_P^{i(j_1)}=y_P^{i(j_2)}$.
Then in must hold that $pip(\mathcal{P})=pip(\widetilde{\mathcal{P}})$. □

B Breaking symmetries: a formal analysis

In the following, we prove that the child generation rule defined in Definition 3 is sufficient to ensure that all the partitions in the tree are *substantially* distinct. In particular, for any two distinct nodes corresponding to subpartitions $\mathcal{R}^1$ and $\mathcal{R}^2$, it is impossible to obtain $\mathcal{R}^1$ by simply applying a permutation to the elements of $\mathcal{R}^2$. To prove this result, we will first define an equivalence relation between subpartitions: we say that two subpartitions $\mathcal{R}^1$ and $\mathcal{R}^2$ are in relation if a permutation exists such that, when applied to the elements of $\mathcal{R}^2$ yields exactly $\mathcal{R}^1$. Then, we will show that, the partition based tree contains exactly one representative subpartition for each equivalence class induced by this relation.
In order to properly state our results, we need some further notation. Let $\mathcal{R}^1=(R_1^1,\ldots,R_K^1)$ and $\mathcal{R}^2=(R_1^2,\ldots,R_K^2)$ be two subpartitions of S. We say that $\mathcal{R}^1$ is related to $\mathcal{R}^2$ and we write $\mathcal{R}^1\sim\mathcal{R}^2$, if there exists a permutation $\sigma:\{1,\ldots,K\}\to\{1,\ldots,K\}$ such that $\mathcal{R}^1=\sigma(\mathcal{R}^2)=(R^2_{\sigma^{-1}(1)},\ldots,R^2_{\sigma^{-1}(K)})$, that is, we have that for all $j\in\{1,\ldots,K\}$ $R_j^1=R^2_{\sigma^{-1}(j)}$, where σ^{-1} is the inverse of σ.

Remark 3. *The relation $\sim$ is an equivalence relation. Indeed:*

- *reflexivity holds: $\mathcal{R}^1\sim\mathcal{R}^1$, since $\mathcal{R}^1=\sigma(\mathcal{R}^1)$, being σ the identity permutation;*
- *symmetry holds: if $\mathcal{R}^1\sim\mathcal{R}^2$, a permutation σ exists such that $\mathcal{R}^1=\sigma(\mathcal{R}^2)$. Then, $\mathcal{R}^2\sim\mathcal{R}^1$ as $\mathcal{R}^2=\sigma^{-1}(\mathcal{R}^1)$, being σ^{-1} the inverse permutation of σ;*
- *transitivity holds: if $\mathcal{R}^1\sim\mathcal{R}^2$ and $\mathcal{R}^2\sim\mathcal{R}^3$, permutations σ,τ exist such that $\mathcal{R}^1=\sigma(\mathcal{R}^2)$ and $\mathcal{R}^2=\tau(\mathcal{R}^3)$. Then, $\mathcal{R}^1=\mu(\mathcal{R}^3)$, being $\mu=\sigma\circ\tau$, so that $\mathcal{R}^1\sim\mathcal{R}^3$.*

Observe that subpartitions belonging to the same equivalence class yield the same objective function value in ($spip(\mathcal{R})$). Therefore, in the subpartition-based tree, only one representative per equivalence class should be retained, and the generation of all $K!$ permutations of a given subpartition $\mathcal{R}$ should be avoided.
In our setting, where the scenario set S is finite, it can be shown that the child-generation rule in Definition 3 ensures that exactly one representative is generated for each equivalence class.

Proposition 9. *Let $\mathcal{R}^1$ and $\mathcal{R}^2$ be two nodes of the branch-and-bound tree. If $\mathcal{R}^1\neq\mathcal{R}^2$, then $\mathcal{R}^1\not\sim\mathcal{R}^2$.*

Proof. Proof From the definition of the subpartition-based tree, the root node is the trivial subpartition of K empty subsets, while a node at depth d in the tree contains exactly d scenarios, that is, if $\mathcal{R} = (R_1, \ldots, R_K)$ is the subpartition of a node of depth d, a subset $T \subseteq S$ exists such that $|T| = d$ and $T = \cup_{j=1}^{K} R_j$.
Note that, if $s \in S$ exists such that $s \in \mathcal{R}^1$ and $s \notin \mathcal{R}^2$, then $\mathcal{R}^1 \nsim \mathcal{R}^2$. Hence, nodes at different depths can never be equivalent.
Let us consider the case where $\mathcal{R}^1$ and $\mathcal{R}^2$ are two nodes of depth d and such that $s \in \mathcal{R}^1$ if and only if $s \in \mathcal{R}^2$. This means, in particular, that $T \subseteq S$, with $|T| = d$ exists such that $T = \cup_{j=1}^{K} R_j^1 = \cup_{j=1}^{K} R_j^2$. We want to prove that, under the assumption that $\mathcal{R}^1 \neq \mathcal{R}^2$, we have $\mathcal{R}^1 \nsim \mathcal{R}^2$.

Claim 1. *Let $\mathcal{A}$ be the unique node generated at depth $d = 1$ following Definition 3, i.e. $\exists\ s_1 \in S$, such that $\mathcal{A} = (\{s_1\}, \emptyset, \ldots, \emptyset)$. Then, for any given node $\mathcal{R}^1$ of depth $d \geq 2$, it holds $\mathcal{A} \subseteq \mathcal{R}^1$.*

Proof. Proof We prove the claim by induction over the depth d.
If $d = 2$, from the child generation rule in Definition 3, we have that $s_2 \in S$, $s_2 \notin \mathcal{A}$ exists such that $\mathcal{R}^1 = (\{s_1, s_2\}, \emptyset, \ldots, \emptyset)$ or (if $K > 1$) $\mathcal{R}^1 = (\{s_1\}, \{s_2\}, \emptyset, \ldots, \emptyset)$. In both cases $\mathcal{A} \subseteq \mathcal{R}^1$.
Let us assume now that for any $\widetilde{\mathcal{R}}$ of depth $d \geq 2$, we have that $\mathcal{A} \subseteq \widetilde{\mathcal{R}}$.
Given any node $\mathcal{R}^1$ of depth $d+1$, let $\mathcal{R}^0$ be the subpartition of its parent node, then, by construction $\mathcal{R}^0$ is of depth d and such that $\mathcal{R}^0 \subseteq \mathcal{R}^1$. From the induction hypothesis, $\mathcal{A} \subseteq \mathcal{R}^0$, then it also holds $\mathcal{A} \subseteq \mathcal{R}^1$. □

Claim 1 states that node $\mathcal{A} = (\{s_1\}, \emptyset, \ldots, \emptyset)$ is a common ancestor for any $\mathcal{R}^1$ and $\mathcal{R}^2$, $\mathcal{R}^1 \neq \mathcal{R}^2$, nodes of depth $d > 1$. Clearly, $\mathcal{R}^1$ and $\mathcal{R}^2$ can also have other common ancestors of different depths i, with $1 \leq i < d$. Let $\mathcal{R} = (R_1, \ldots, R_K)$ be the common ancestor of $\mathcal{R}^1$ and $\mathcal{R}^2$ of maximum depth, i.e. a node such that $\mathcal{R} \subseteq \mathcal{R}^1$ and $\mathcal{R} \subseteq \mathcal{R}^2$ and of depth i, $1 \leq i < d$ as big as possible. Since $i < d \leq l$, according to Definition 3, $s \notin \mathcal{R}$ is chosen such that every subpartition of a child node of $\mathcal{R}$ is defined as $\mathcal{R}^k = (R_1, \ldots, R_k \cup \{s\}, \ldots, R_K)$ for $k = 1, \ldots, K'$, where K' is defined as in (8). Observe that $s \notin \mathcal{R}$, but $s \in \mathcal{R}^1$ and $s \in \mathcal{R}^2$.

Claim 2. *Let $j_1, j_2 \in \{1, \ldots, K'\}$ be such that $s \in R_{j_1}^1$ and $s \in R_{j_2}^2$. Then $j_1 \neq j_2$.*

Proof. Proof By construction, $R_{j_1} \subseteq R_{j_1}^1$ and $R_{j_1} \subseteq R_{j_1}^2$. Let us assume by contradiction that $j_1 = j_2$ so that $R_{j_1} \subseteq R_{j_1}^2 = R_{j_2}^2$. Hence, node $\mathcal{F}^{j_1} = (R_1, \ldots, R_{j_1} \cup \{s\}, \ldots, R_K)$ has depth $i+1$ and is such that $\mathcal{F}^{j_1} \subseteq \mathcal{R}^1$ and $\mathcal{F}^{j_1} \subseteq \mathcal{R}^2$, which contradicts the fact that $\mathcal{R}$ is the node of maximum depth i such that $\mathcal{R} \subseteq \mathcal{R}^1$ and $\mathcal{R} \subseteq \mathcal{R}^2$. □

From Claim 2, scenario s is such that $s \in R_{j_1}^1$ and $s \in R_{j_2}^2$ with $j_1 \neq j_2$. We shall now show that this implies that $\mathcal{R}^1 \nsim \mathcal{R}^2$.
Let us assume by contradiction that $\mathcal{R}^1 \sim \mathcal{R}^2$. Then, there exists a permutation σ such that $\mathcal{R}^1 = \sigma(\mathcal{R}^2)$. We distinguish two cases.

- If $\sigma^{-1}(j_1) = j_2$, then it must hold $R_{j_1}^1 = R_{\sigma^{-1}(j_1)}^2 = R_{j_2}^2$. Recall that $\mathcal{R} = \cup_{j=1}^{K} R_j$ is the common ancestor of maximum depth of $\mathcal{R}^1$ and $\mathcal{R}^2$. Without loss of generality we can assume that $1 \leq j_1 < j_2 \leq K'$, so that $R_{j_1} \neq \emptyset$ by definition of K' in (8). Since $\mathcal{R}$ is a subpartition we have $R_{j_1} \neq R_{j_2}$, since $R_{j_1} \cap R_{j_2} = \emptyset$. Then, there exists $s^* \in \mathcal{R}$ such that $s^* \in R_{j_1}, s^* \notin R_{j_2}$. Hence $s^* \in R_{j_1} \subseteq R_{j_1}^1 = R_{j_2}^2$, as well as $s^* \in R_{j_1} \subseteq R_{j_1}^2$ by construction. In particular, we have that $s^* \in R_{j_1}^2 \cap R_{j_2}^2 = \emptyset$, which is a contradiction since $\mathcal{R}^2$ is a subpartition and $j_1 \neq j_2$.
- If $\sigma^{-1}(j_1) = j$, $j \neq j_2$, then it must hold $R_{j_1}^1 = R_j^2$. However, $s \in R_{j_1}^1 = R_j^2$ and $s \in R_{j_2}^2$, which implies that $s \in R_j^2 \cap R_{j_2}^2 = \emptyset$, since $\mathcal{R}^2$ is a subpartition and $j \neq j_2$, which is a contradiction.

In both cases we reached a contradiction, hence we can conclude that $\mathcal{R}^1 \nsim \mathcal{R}^2$. □

Remark 4. *Claim 1 would not hold in case the child generation rule in Definition 3 is not adopted. In fact, if we generate K child nodes of $\mathcal{R}$ by adding a new scenario $s \notin \mathcal{R}$ to every element of $\mathcal{R}$ then, the only common ancestor to every couple of nodes in the tree is the empty partition. Proposition 9 does not hold for the tree in Figure 4.*

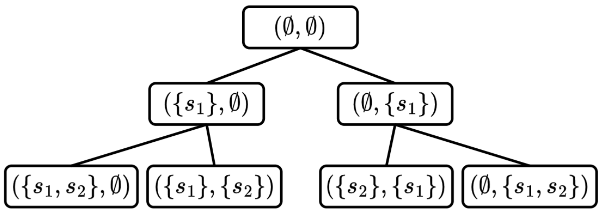


Figure 4: Example of the tree obtained by always generating K child nodes, instead of relying on Definition 3 ($\ell = 2$, $K = 2$).

C Formulation of the problems used in the computational tests

C.1 Two-stage stochastic knapsack problem with setup

The knapsack problem with setup Michel et al. (2009) is a two-stage variant of the knapsack problem that has pure binary first and second stage variables. In the knapsack problem with setup the set of items $N = \{1, \dots, n_y\}$ is partitioned into n_x classes $R_h \subseteq N$, $h \in R = \{1, \dots, n_x\}$ and the items in a class can only be inserted into the knapsack only if the corresponding class was activated in the first stage. Activating a class involves a non-negative cost f_h and a non-negative reduction d_h of the knapsack capacity. Here we consider the stochastic version of this problem, in particular the cost q_{si} of each item $i \in N$ is scenario dependent, as well as its weight w_{si} and the maximum (c_s) and minimum (g_s) capacity of the knapsack. The objective is to choose a set of classes to be activated and a subset of items to be selected so that the difference between the average profit over all scenarios and the activation costs is maximized, while the total weight of the items lies in the prescribed interval between maximum and minimum capacity, computed by taking into account the class setup reductions. In particular, first stage binary variables x_h, $h \in R$, take value 1 if class h is activated, and second stage binary variables y_{si} are set equal to one if item i is selected in scenario s. The extensive formulation of the problem is as follows:

$$
\begin{aligned}
\max \quad & -\sum_{h\in R} f_h x_h + \sum_{s\in S} p_s \Big(\sum_{i\in N} q_{si} y_{si} \Big) \\
\text{s.t.} \quad & \sum_{h\in R} d_h x_h + \sum_{i\in N} w_{si} y_{si} \le c_s, \quad s \in S, \\
& \sum_{h\in R} d_h x_h + \sum_{i\in N} w_{si} y_{si} \ge g_s, \quad s \in S, \\
& y_{si} \le x_h, \quad h \in R,\ i \in R_h,\ s \in S, \\
& x_h \in \{0,1\}, \quad h \in R, \\
& y_{si} \in \{0,1\}, \quad i \in N,\ s \in S.
\end{aligned}
$$

C.2 Two-stage stochastic quadratic knapsack problem with setup

The quadratic knapsack problem with setup Galli et al. (2025), is the generalization of the previous problem that includes a quadratic objective function. It inherits all parameters and constraints from the knapsack problem with setup, with the addition of pairwise profits h_{sij} which contribute to the objective function if both items i and j are inserted into the knapsack in scenario s. Once again, we considered the stochastic version of the problem and the extensive formulation can be written as follows:

$$
\begin{aligned}
\max \ & -\sum_{h\in R} f_h x_h + \sum_{s\in S} p_s \Big(\sum_{i\in N} q_{si} y_{si} + \sum_{i\in N} \sum_{j\in N: i<j} h_{sij} y_{si} y_{sj} \Big) \\
\text{s.t.} \ & \sum_{h\in R} d_h x_h + \sum_{i\in N} w_{si} y_{si} \le c_s, \quad s\in S, \\
& \sum_{h\in R} d_h x_h + \sum_{i\in N} w_{si} y_{si} \ge g_s, \quad s\in S, \\
& y_{si} \le x_h, \quad h\in R,\ i\in R_h,\ s\in S, \\
& x_h \in \{0,1\}, \quad h\in R, \\
& y_{si} \in \{0,1\}, \quad i\in N,\ s\in S.
\end{aligned}
$$

C.3 Two-stage stochastic capacitated facility location problem

For the capacitated facility location problem we considered the stochastic version of the classical formulation from the OR library http://old.math.nsc.ru/AP/benchmarks/CFLP/cflp-eng.html, with the addition of a new set of constraints. In particular, for each facility location $i \in I = \{1, \dots, n_x\}$, we have an opening cost c_i, and a maximum and minimum value of the production that must be carried out if that facility is opened, respectively V_i and v_i. For every pair of facility i and client $j \in J = \{1, \dots, n_y\}$, there is a non-negative cost g_{sij} if client j is served by facility i, and a non-negative demand d_{sij} that client i requires from facility j, in scenario $s \in S$. Binary variables x_i take value 1 if facility i is opened, while second stage binary variable y_{sij} is set to 1 if client j is served by facility i in scenario s. The extensive formulation of the problem is the following:

$$
\begin{aligned}
\min \ & \sum_{i\in I} c_i x_i + \sum_{s\in S} p_s \left(\sum_{i\in I} \sum_{j\in J} g_{sij} y_{sij} \right) \\
\text{s.t.} \ & \sum_{i\in I} y_{sij} = 1, \quad j\in J,\ s\in S, \\
& \sum_{j\in J} d_{sij} y_{sij} \le V_i x_i, \quad i\in I,\ s\in S, \\
& \sum_{j\in J} d_{sij} y_{sij} \ge v_i x_i, \quad i\in I,\ s\in S, \\
& x_i \in \{0,1\}, y_{sij} \in \{0,1\}, \quad i\in I, j\in J, s\in S.
\end{aligned}
$$

C.4 Two-stage stochastic quadratic capacitated facility location problem

The quadratic capacitated facility location problem is a direct generalization of the previous problem. In fact, it inherits all of its parameters and constraints, and the only addition is represented by pairwise costs h_{si} which are included in the objective function if more than one client is served by facility i in scenario s. The extensive formulation of the problem is as follows:

$$
\begin{aligned}
\min \ & \sum_{i\in I} c_i x_i + \sum_{s\in S} p_s \left(\sum_{i\in I} \sum_{j\in J} g_{sij} y_{sij} + \sum_{i\in I} h_{si} \left(\sum_{j\in J} \sum_{j'\in J: j<j'} y_{sij} y_{sij'} \right) \right) \\
\text{s.t.} \ & \sum_{i\in I} y_{sij} = 1, \quad j\in J,\ s\in S, \\
& \sum_{j\in J} d_{sij} y_{sij} \le V_i x_i, \quad i\in I,\ s\in S, \\
& \sum_{j\in J} d_{sij} y_{sij} \ge v_i x_i, \quad i\in I,\ s\in S, \\
& x_i \in \{0,1\}, y_{sij} \in \{0,1\}, \quad i\in I, j\in J, s\in S.
\end{aligned}
$$