K-Adaptability in stochastic optimization

E. Malaguti and M. Monaci
DEI "Guglielmo Marconi", Università di Bologna, Viale Risorgimento 2, I-40136 Bologna, Italy

J. Pruente
Fakultät für Mathematik, TU Dortmund, Vogelpothsweg 87, 44227 Dortmund, Germany

Abstract

We consider stochastic problems in which both the objective function and the feasible set are affected by uncertainty. We address these problems using a K-adaptability approach, in which K solutions for a given problem are computed before the uncertainty dissolves and afterwards the best of them can be chosen for the realized scenario. We analyze the complexity of the resulting problem from a theoretical viewpoint, showing that, even in case the deterministic problem can be solved in polynomial time, deciding if a feasible solution exists is $\mathcal{NP}$-complete for discrete probability distributions. Furthermore we prove that evaluating the objective function is # P-hard for continuous probability distributions. Besides that, we prove that an approximation factor for the underlying problem can be carried over to our problem. Finally, we present exact solution approaches including a branch-and-price algorithm. An extensive computational analysis compares the performances of the proposed algorithms on a large set of randomly generated instances.

I. INTRODUCTION

One of the most relevant assumptions in mathematical programming is that the exact value of the input data is fixed and known in advance. However, there is a large variety of real applications in which this is not the case, and some parameters used in the model are just estimates of real parameters and/or are subject to some uncertainty. In this case, an optimal solution according to the nominal values of the parameters can be suboptimal (or even infeasible) according to the actual parameters. From a practical viewpoint, it is thus extremely important to take these aspects into account when real-world applications are considered.

Two main approaches have been proposed in the literature to deal with uncertainty, namely Robust Optimization and Stochastic Optimization.

Robust Optimization ([14], [2], [5], [1]) assumes full knowledge about the domain of uncertainty parameters in all "reasonable" realizations of the data. Classic robust optimization defines the solution under the worst case scenario for uncertainty, imposing hard constraints to reduce the feasible set and to forbid solutions that become infeasible under certain realizations of the data. The computational tractability of the resulting model depends on the complexity of the underlying nominal problem and on the properties of the uncertain domain. Robust models typically produce solutions that are too conservative in terms of cost compared to non-robust solutions. For this reason, less conservative robust models have been proposed in the literature, either fixing in advance the maximum worsening of the solution value [8], [13], or relaxing somehow the level of protection against uncertainty and allowing for possible recovery of the solution [10]. As shown in [4], the solution produced by a robust optimization model can be arbitrarily bad in terms of cost compared to the optimal expected cost of the associated stochastic problem.

Stochastic Optimization ([6, 12]) is mainly used when some information about the probabilistic distribution of effective data is available. According to this paradigm, the solution is obtained solving an auxiliary optimization problem that includes additional variables modeling uncertainty, and penalizes feasible solutions that are most likely to become infeasible. Computing exactly the expected value of a stochastic program is computationally hard in general, and solution approaches are based on heuristic algorithms. In some applications, chance constraints are used to impose the solution to be feasible with a given probability [11]. Unfortunately, in real applications the number of scenarios to be considered is very large, producing very large models that are extremely hard to solve in practice. In addition, feasibility of the solution in the actual scenario cannot be guaranteed, and the approach may be combined with the definition of some recourse actions.

Indeed, in many applications there is a number of "wait-and-see" decisions that have to be taken after the actual realization of uncertainty. In these models, that are usually denoted as *two-stage*, there are some strategic decisions that have to be taken in advance, and some operational decisions that are used to recover the quality or the feasibility of the solution. The *K-adaptability* approach, firstly introduced in [3], is a variant of two-stage optimization in which the user pre-selects a finite set of (say) K solutions and chooses the best of these solutions at a second stage, when the exact nature of uncertainty materializes. This is extremely useful for those situations in which uncertainty is modeled by considering a finite number of scenarios, the actual scenario materializes a short time before decisions have to be taken, and the hardness of the problems prevents to compute a "reasonable" solution from scratch. This paradigm can be used on any optimization problem. In the remainder of the paper, we call the inner optimization problem the underlying problem.

As an example, consider a routing problem in which a fleet of vehicles has to be used to serve a given set of customers, whose demand (or time windows) is known only a few minutes before service starts. In this case, using an optimal solution computed according to nominal values of the parameters may be impractical, as this solution may be very bad in terms of cost or even infeasible. At the same time, computing a good-quality solution in a very short time may be extremely hard from a computational viewpoint. On the other hand, choosing a solution among K precomputed solutions can be done in a reasonably small time and improves the objective value compared to the classical stochastic solution, in which only one solution is precomputed. Another relevant application for a K-adaptability approach is disaster management, where the decision maker has to prepare escape plans for an evacuation that have to be trained in advance ([9]). It is clear that only training a small number of such plans is realistic. In other cases, the alternative solutions may have to be prepared physically, e.g., by establishing links in a network, so that computing and implementing a solution after the scenario materialized is again not possible, and only a small set of candidate solutions can be considered.

Compared with classical two-stage optimization, another advantage of K-adaptability is that it allows the user to take under control the flexibility that is used to face with different scenarios of the problem. This may be crucial in many applications, e.g., school bus routing and home care services for elderly persons. In these contexts, it is preferable to limit the number of different agents that provide service to each customer in different days. This is an immediate byproduct of the K-adaptability approach, as each day the actual solution is taken from the set of the K different precomputed solutions. Changing the value of parameter K allows the decision maker to implement different policies. Small values of K correspond to a conservative policy in which the actual solution has a small variability. On the other hand, the larger the value of K the larger the variability of the solution and, consequently, the better the solution fits with the actual scenario.

Finally, as recently observed in [15], K-adaptability appears to be coherent with human decision-making, which tends to forecast a limited number of alternative solutions and to apply one of them, instead of computing an optimal solution for each realization of the data. Indeed, changing decisions too frequently may be confusing to the final user, who can refuse the proposed solution.

In this paper we consider an underlying optimization problem of the form

$$\begin{array}{ll} \min & \xi^T x \\ \text{s.t.} & x \in X \subseteq \mathbb{R}^n, \end{array} \tag{1}$$

in which uncertainty may affect both the objective function and the feasible region. The objective is to determine at most K solutions $x^1, x^2, \dots, x^K$ to the underlying problem (1) such that the expected value of the best of these solutions is a minimum. In the following, a set of K solutions to the underlying problem will be denoted by $\mathcal{X}_K$ and will be called a *set of solutions*; conversely, a single solution $x \in X$ to the underlying problem will be called a *solution.* Note that, without loss of generality, we may assume that the problem asks to determine a set of *exactly* K solutions; indeed, if the optimal choice is to have less than K solutions, some of the solutions can be duplicated. Thus, the resulting problem is

$$\begin{array}{ll} \min & E\Big(\min\{\xi^T x : x \in \mathcal{X}_K \cap X\} \Big) \\ \text{s.t.} & |\mathcal{X}_K| \le K. \end{array} \tag{2}$$

In the following, we will denote this problem as the *Constrained Min-E-Min problem* (CmEm).

We assume that uncertainty is described by discrete variables, i.e., it can be modeled by defining a number (say) ℓ of scenarios. Each scenario j ($j = 1, \dots, \ell$) has associated a probability $p_j > 0$, an objective function $\xi^j \in \mathbb{R}^n$, and the associated feasible set $X^j \subseteq \mathbb{R}^n$ is a polyhedron, i.e., $X^j = \{x \in \mathbb{R}^n : A^j x \le b^j\}$. Thus, the subproblem associated with scenario j can be rewritten as $\min\{{\xi^j}^T x : x \in X^j\}$, and problem (2) can be reformulated as

$$\begin{array}{ll} \min & \displaystyle\sum_{j=1}^{\ell} p_j \min\{{\xi^j}^T x : x \in \mathcal{X}_K \cap X^j\} \\ \text{s.t.} & |\mathcal{X}_K| \le K. \end{array}$$

To simplify the notation, we incorporate the probabilities into the objective function vectors, i.e., we consider the following problem

$$\begin{aligned} \min\ & \sum_{j=1}^{\ell} \min\{{\xi^j}^T x : x \in \mathcal{X}_K \cap X^j\} \\ \text{s.t.}\ & |\mathcal{X}_K| \leq K. \end{aligned} \tag{3}$$

We denote the resulting problem by *Discrete Constrained Min-E-Min problem* (DCmEm).

Recently, a special case of problem (3) in which uncertainty affects the objective function only has been addressed in [7]. For this special case, that will be denoted as (mEm) in the following, the authors studied the computational complexity, gave approximation results, and introduced mathematical formulations and solution algorithms. A similar context was considered in [15], where the relation between classical two-stage optimization and the min-max-min version of K-adaptability was considered. In addition, a branch-and-bound approach for the optimal solution of the problem was introduced, establishing conditions for its asymptotic and finite time convergence.

a. Contributions The first main contribution of this paper is an investigation of the complexity of Problem (DCmEm). We show that deciding whether a feasible solution exists is $\mathcal{NP}$-complete, even if $K \geq 2$ is fixed and the underlying problem can be solved in polynomial time. In case K is part of the input, we prove that the same result holds even if all the coefficients of the constraints defining the feasible region are binary and $|X_j|$ is a polynomial in the input size for every scenario j. The second major conbtribution is the development of mathematical models and sound exact approaches (including a branch-and-price algorithm) for the solution of (DCmEm). The third contribution is a computational evaluation of the performances of the proposed algorithms on a large benchmark of randomly generated instances, obtained by considering underlying problems with different levels of complexity.

b. Outline The rest of this paper is organized as follows. In the next section, we investigate the complexity (DCmEm) and (CmEm) in terms of $\mathcal{NP}$-hardness, fixed-parameter tractability and $\#P$-hardness. In Section III, we describe a compact formulation and two different Set Partitioning formulations for solving (DCmEm). All other exact solution approaches including the branch-and-price algorithm are presented in Section IV. In Section V we show how using approximation algorithms for solving the underlying problem affects the solution quality of (DCmEm). The different exact algorithms for solving (DCmEm) are then analyzed in an extensive computational study, that is presented in Section VI. Section VII concludes the paper.

II. COMPLEXITY RESULTS

In [7], the authors showed that the (mEm) is $\mathcal{NP}$-hard. In addition they proved that this problem is not in APX if K is part of the input, and the same holds for $K \geq 3$ if K is not part of the input. Besides, they showed that the problem is W[2]-hard for parameter K and W[1]-hard for parameter $n - K$. Trivially, all these results apply to (DCmEm) as well.

In the next section, we discuss the complexity of the problem of finding a feasible solution in case uncertainty affects the problem constraints as well. Section II B introduces a subproblem that will be used in our solution algorithms and analyzes its complexity. Finally, Section II C gives some results on the $\#P$-hardness of the continuous version of problem (3).

A. Uncertain coefficients in the constraints

Theorem 1. Deciding whether (DCmEm) has a feasible solution is W[2]-complete parameterized by K if K is part of the input, even if all the coefficients are binary and $|X^j|$ is a polynomial in the input size for every scenario j.

Proof. We prove the statement by reduction of Set Cover: given a positive integer $\overline{K}$, a set $U = \{u_1, u_2, \ldots\}$ of items, and a collection $S = \{s_1, s_2, \ldots\}$ of subsets of U, the Set Cover problem asks if there exists a sub-collection of S with cardinality at most $\overline{K}$, so that every item is in least one of the subsets. The Set Cover is known to be W[2]-complete parameterized by $\overline{K}$.

We will show that, given an instance of Set Cover, we can define an instance of (DCmEm) that is feasible if and only if the Set Cover instance has a positive answer. For the reduction, we define a (DCmEm) instance with $n := |S|$ variables and $\ell := |U|$ scenarios. For each scenario j, the feasible set X^j is the set of base vectors of $\mathbb{R}^n$ that satisfy a single constraint of the form ${a^j}^T x \geq b$, where $b := 1$ and the coefficient of each variable h $(h = 1, \ldots, n)$ is defined as:

$$a^j_h := \begin{cases} 1 & \text{if } u_j \in s_h, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we set $K := \overline{K}$, and therefore we have a parameterized reduction.
The set of solutions of (DCmEm) $\mathcal{X}_K$ induces the Set Cover as follows: if the h-th base vector belongs to $\mathcal{X}_K$, then item set s_h belongs to the Set Cover solution. With this definition, it holds that the j-th item is included in at least one subset if and only if there exists a selected subset s_h such that $u_j \in s_h$; this means that the associated coefficient a_h^j is 1, i.e., the h-th solution is feasible for scenario j. Hence we have a Set Cover if and only if, for every scenario j, at least one solution in $\mathcal{X}_K$ is feasible. $\square$ $\square$

We now discuss a similar result for the case of fixed K.

Theorem 2. Deciding whether (DCmEm) has a feasible solution is $\mathcal{NP}$-complete if $K = 2$ is not part of the input.

Proof. We show $\mathcal{NP}$-completeness by reduction from the decision variant of the Maximum Cut Problem. Let $G = (V, E)$ be an undirected graph, and let $W \subseteq V$ be a subset the vertex set. The induced cut is the set of edges that have exactly one endpoint in W, i.e., $\delta(W) = \{(v_i, v_j) \in E : v_i \in W, v_j \notin W\}$. Given a positive integer q, the Maximum Cut Problem asks whether there exists a subset W such that $|\delta(W)| \geq q$. This problem is known to be $\mathcal{NP}$-complete even for $q = 2$.
Given a graph $G = (V, E)$, we define a (DCmEm) instance with $n := 2|V|$ variables and $\ell := 2|V|$ scenarios. In all scenarios, we impose that

$$x_h \in \{0, 1\} \quad h = 1, \dots, |V|,$$

i.e., the first $|V|$ variables are constrained to be binary and

$$x_h \geq \epsilon, \quad h = |V| + 1, \dots, 2|V|,$$

where $\epsilon = \frac{1}{|V|+1}$. In addition, scenarios are partitioned into two classes having the same cardinality $|V|$. Each scenario j in the first class ($j = 1, \dots, |V|$) is characterized by two additional constraints

$$\sum_{h=1}^{|V|} a_h^j x_h = x_{|V|+j}, \qquad \text{and} \qquad \sum_{h=1}^{|V|} x_{|V|+h} \geq q,$$

where the coefficients a^j are defined as follows

$$a_h^j := \begin{cases} \delta(v_j) & \text{if } h = j, \\ -1 & \text{if } (v_h, v_j) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

and $\delta(v_j) = |\{(v_i, v_j) \in E\}|$ denotes the degree of a given vertex $v_j \in V$.
Finally, each scenario j in the second class is characterized by a single constraint

$$x_{|V|+j} = \epsilon.$$

Assume that (DCmEm) has a feasible solution $\mathcal{X}_2 = \{x^1, x^2\}$, i.e., for each scenario either x^1 or x^2 is feasible (or both).
We first observe that, for each $j = 1, \dots, |V|$, the j-th scenario in the first class and the j-th scenario in the second class cannot be covered by the same solution. Indeed, while the first constraint of the former scenario sets variable $x_{|V|+j}$ to an integer value, in the former scenario the same variable must take a fractional value. Thus, no scenario may have that both x^1 and x^2 are feasible, and each scenario is covered by exactly one solution. Let $S \subseteq \{1, \dots, |V|\}$ be the set of indices of the scenarios in the first class for which x^1 is feasible, and let $j \in S$ be one of these scenarios. Therefore, we must have $x_j^1 = 1$ since otherwise $\sum_{h=1}^{|V|} a_h^j x_h^1 \leq 0$ which is not compatible with the domain of variable $x_{|V|+j}$.

Using the previous observation, x^2 is not feasible for scenario j. Thus, in case $x_j^2 = 1$, we can define a new solution having $x_j^2 = 0$ since this operation does not affect the feasibility of x^2 for scenarios in which this solution is feasible. Hence, in the first solution, variable $x_{|V|+j}$ takes a value equal to the number of neighbors of vertex v_j whose associated scenario is not in S. Finally, observe that x^1 is feasible in all scenarios in the second class that are not in S, i.e., $x_{|V|+h}^1 = \epsilon$ for all $h \notin S$. Given the second constraint of each scenario in S, solution x^1 must satisfy

$$\sum_{\substack{h=1 \\ v_h \in S}}^{|V|} x_{|V|+h}^1 \geq q \tag{4}$$

as the total contribution of the remaining $x^1_{|V|+h}$ variables for $v_h \notin S$ is smaller than 1.
Condition (4) implies that the cut induced by vertex set $W = \{v_j : j \in S\}$ has size at least equal to q.
Conversely, assume now that we have a subset W of vertices whose associated cut has size q (or more), and define the following set of solutions $\mathcal{X}_2 = \{x^1, x^2\}$: for each vertex $v_j \in W$ we set

$$x^1_j := 1, \quad x^1_{|V|+j} := \sum_{h=1}^{|V|} a^j_h x_h, \quad x^2_j := 0, \quad x^2_{|V|+j} := \epsilon$$

while for each vertex $v_j \notin W$ we have

$$x^1_j := 0, \quad x^1_{|V|+j} := \epsilon, \quad x^2_j := 1, \quad x^2_{|V|+j} := \sum_{h=1}^{|V|} a^j_h x_h,$$

where the a^j_h coefficients are the same as in the first part of the proof. First of all observe that each variable $x_{|V|+j}$ associated with a vertex $v_j \in W$ has, in the first solution, a value equal to the number of edges that connect v_j to vertices outside W.
We now prove that every scenario admits a feasible solution in $\mathcal{X}_2$. Let j (say) be a scenario in the first class such that $v_j \in W$. By construction, solution x^1 satisfies the first constraint of the scenario. In addition, for each vertex $v_h \in W$, variable $x^1_{|V|+h}$ gives the number of edges that connect vertex v_h to vertices in the other side of the partition; finally, for vertices $v_h \notin W$, the associated variable is set to a positive value. Thus, we have

$$\sum_{h=|V|+1}^{2|V|} x^1_h = \sum_{h=1}^{|V|} x^1_{|V|+h} \geq \sum_{\substack{h=1 \\ v_h \in W}}^{|V|} x^1_{|V|+h} \geq q$$

where the last inequality derives from the value of the cut induced by W. The case with $v_j \notin W$ is analogous, using solution x^2.
Finally, it is easy to see that solution x^2 (resp. x^1) is feasible for each scenario j of the second set such that $v_j \notin W$ (resp. $v_j \in W$). □ □

Theorem 3. Deciding whether (DCmEm) has a feasible solution is $\mathcal{NP}$-complete if $K \geq 3$ is not part of the input.

Proof. For $K \geq 3$, we show $\mathcal{NP}$-completeness by reduction from the Vertex Coloring Problem. A $\overline{K}$-vertex coloring of a graph $G = (V, E)$ is an assignment of $\overline{K}$ different colors to all vertices in V such that no edge in E connects two vertices with the same color. Vertices that receive the same color are called a color class. It is easy to see that each color class is a stable set and that the problem remains $\mathcal{NP}$-complete even when a vertex is allowed to belong to more than one color class.
Given a graph $G = (V, E)$ and an integer $\overline{K}$, we define a (DCmEm) instance with $n := |V|$ binary variables and $\ell := |V|$ scenarios. For each scenario j, there is a single constraint of the form ${a^j}^T x \leq b$, and the coefficient of each variable h is given by

$$a^j_h := \begin{cases} -1 & \text{if } h = j, \\ 1 & \text{if } (v_j, v_h) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we set $b := -1$ and define $K := \overline{K}$.
A set of solutions $\mathcal{X}_K$ induces the color classes for Vertex Coloring as follows: for each solution $x^i \in \mathcal{X}_K$, all vertices that correspond to variables that are taken in the solution belong to the i-th color class. This solution satisfies the coloring constraint as scenario constraints forbid any two neighbors to be taken in the same solution. Finally, feasibility of each scenario in at least one solution implies that every vertex belongs to a color class.
Conversely, given a Vertex Coloring solution, we can define a set of solutions $\mathcal{X}_K$ as follows: each solution x^i includes all variables that are associated to vertices belonging to the i-th color class. Remember that each color class corresponds to a stable set. Thus, a feasible solution for the j-th scenario is induced by the color class including vertex j, as by definition this color class cannot include any neighbor of j. Since all vertices receive a color, then all scenarios are satisfied by at least one solution.

□

B. The Completion problem

The *Completion problem* is a variant of the (DCmEm) arising when $\overline{K}$ (say) solutions have been fixed and are part of the input. The objective is then to determine the remaining $K_f := K - \overline{K}$ *free* solutions. We denote by $\mathcal{X}_{\overline{K}}$ the set of fixed solutions and by $\mathcal{X}_{K_f}$ the set of free solutions. The completion problem can be formulated as follows

$$\begin{aligned} \min \quad & \sum_{j=1}^{\ell} \min\{\xi^T x : x \in (\mathcal{X}_{\overline{K}} \cup \mathcal{X}_{K_f}) \cap X^j\} \\ \text{s.t.} \quad & |\mathcal{X}_{K_f}| \leq K_f. \end{aligned}$$

The completion problem appears as a subproblem in a solution approach based on column generation (see Section IV B 1). We now discuss its complexity.

Theorem 4. The completion problem is $\mathcal{NP}$-hard and W[2]-hard if
$2 \leq K_f < \ell$.

Proof. It is easy to see that an instance of (DCmEm) with parameter K can be modeled as an instance of the completion problem with parameter $K_f = K$ and $\overline{K} = 1$, using a dummy solution that is infeasible (or high-costly) for all scenarios. The result follows from the complexity of (DCmEm). □ □

More interesting is what happens for $K_f = 1$: in this case the completion problem turns out to be $\mathcal{NP}$-hard if at least one solution is fixed, though (DCmEm) is not.

Theorem 5. The completion problem is $\mathcal{NP}$-hard even if $K_f = \overline{K} = 1$, the uncertainty affects the objective function only and $X = \{0,1\}^n$.

Proof. We prove the statement by reduction from Maximum Cut Problem.

Given a graph $G = (V, E)$, we define an instance of the completion problem with $\overline{K} = K_f = 1$ as follows. There are $n := |V|$ binary variables and $\ell := 2|E|$ scenarios, all with no explicit constraints, i.e., $X_j := \{0,1\}^n$ $(j = 1, \dots, \ell)$. For each edge $e = (u, v) \in E$, there are two scenarios numbered as e and $|E| + e$. For the former, the objective function coefficients for the variables are

$$\xi_h^e := \begin{cases} -1 & \text{if } h = u, \\ 2 & \text{if } h = v, \\ 0 & \text{otherwise,} \end{cases}$$

while for the latter the objective function is defined as follows

$$\xi_h^{|E|+e} := \begin{cases} 2 & \text{if } h = u, \\ -1 & \text{if } h = v, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, the fixed solution is a vector of zeros, hence, it has a zero cost for every scenario.

We now show that there exists a subset $W \subseteq V$ of vertices whose induced cut has size q or more if and only if there exists a solution for the completion problem with value of $-q$ or less.

Assume that G has a cut with value q, i.e., there exists a subset W of vertices such that $|\{e = (u, v) \in E : u \in W, v \notin W\}| \geq q$. Define a solution for the completion problem by setting $x_h^1 = 1$ if $h \in W$ and 0 otherwise. Since the number of edges with exactly one endpoint in W is at least q, there are at least q scenarios for which this solution has a value -1. For all the remaining scenarios there exists a solution with zero value, hence the value of the completion problem is at least $-q$.

Conversely, assume now that a solution of the completion problem exists with value $-q$. Define set W that includes all the vertices associated with variables that take value 1 in the free solution. As the completion problem has value $-q$, there are q scenarios whose value is -1. Every such scenario is associated with an edge that must have exactly one endpoint in W (otherwise, it would have a positive cost), hence the thesis. □ □

C. #P-hardness

We now discuss the #P-hardness of the continuous version of (CmEm). To this aim, we use the following result, that was proved in [9] (see the proof to Lemma 2 combined with theorem 1):

Corollary 1. Computing $\mathcal{Q}(a,b)\colon = E_{\xi}(\max\{a^T\tilde{\xi}-b,0\})$ is $\#P$-hard if $\tilde{\xi}\sim\mathcal{U}[0,1]^n$.

Theorem 6. Evaluating the objective function of (CmEm) for a fixed solution is $\#P$-hard for general continuous distributions even if $K=2$.

Proof. We reduce the problem of calculating $\mathcal{Q}(a,b)$ to the evaluation of (DCmEm). Let $a\in\mathbb{Z}^n$ and $b\in\mathbb{Z}$ be our input. For our reduction we choose $\tilde{\xi}\sim\mathcal{U}[0,1]^n$ and $\xi=\begin{pmatrix}\tilde{\xi}\\1\end{pmatrix}$. We set the 2 fixed solutions for (CmEm) to $x_1=\begin{pmatrix}-a\\0\end{pmatrix}$ and $x_2=\begin{pmatrix}0\\-b\end{pmatrix}$ with $x_1,x_2\in\mathbb{Z}^{n+1}$. Now we reformulate the objective function of (CmEm):

$$E_{\xi}(\min\{\xi^Tx_1,\xi^Tx_2\})$$

$$=-E_{\xi}(\max\{a^T\tilde{\xi},b\})$$

$$=-b-E_{\xi}(\max\{a^T\tilde{\xi}-b,0\})$$

$$=-b-\mathcal{Q}(a,b)$$

Therefore $\mathcal{Q}(a,b)$ can be computed by computing $E_{\xi}(\min\{\xi^Tx_1,\xi^Tx_2\})$ and it follows the statement. □ □

Theorem 6 implies that it is not possible for every solution to evaluate the objective function for (CmEm) with a continuous distribution, unless $\mathcal{P}=\mathcal{NP}$.

III. MATHEMATICAL MODELS

Before discussing mathematical formulations that can be used to model (DCmEm), observe that there are relevant situations in which uncertainty affects the feasible set at an interdiction level. This special case arises, e.g., when the underlying problem is shortest path or spanning tree and uncertainty affects the *availability* of some edges in some scenarios. In this case, one can change the objective function of each scenario to strongly penalize the use of forbidden edges, thus reducing the problem to an (mEm), and allowing its solution through the algorithms for the unconstrained case.

A. Compact formulation

We now present a quadratic programming formulation, derived from a similar model in [7], that involves a polynomial number of variables and constraints. From now on, we will say that a scenario j is *covered* by the i-th solution if solution i is feasible for scenario j and it is selected, among those that are feasible, for that scenario. For each solution i and scenario j, let y_{ij} be a binary variable taking value 1 if and only if scenario j is covered by solution i. The model reads

$$\min \sum_{j=1}^{\ell}\xi^{jT}\left(\sum_{i=1}^{K}y_{ij}\,x^i\right) \tag{5}$$

$$\text{s.t.}\quad \sum_{i=1}^{K}y_{ij}=1 \qquad j=1,\dots,\ell \tag{6}$$

$$y_{ij}=1\Rightarrow x^i\in X^j \qquad i=1,\dots,K;j=1,\dots,\ell \tag{7}$$

$$y_{ij}\in\{0,1\} \qquad i=1,\dots,K;j=1,\dots,\ell. \tag{8}$$

Constraints (6) impose that each scenario is covered by exactly one solution, whose cost is taken into account in the objective function (5). Domain for the y-variables are imposed by (8). Constraints (7) guarantee that each scenario is covered by a solution that is feasible for that scenario. We now discuss possible methods for handling this kind of constraints when using a general-purpose MILP solver. Denoting by n the number of variables in the underlying problem, and assuming that the feasible set of each scenario j is a polyhedron $X^j = \{x \in \mathbb{R}^n : a^j_r x \geq b^j_r; r = 1, \dots, m\}$ characterized by m linear constraints, the first option is to apply linearization, and to replace (7) by

$$a^{jT}_r\, x^i + BIGM(1 - y_{ij}) \geq b^j_r \qquad i = 1, \dots, K; j = 1, \dots, \ell; r = 1, \dots, m$$

where BIGM is a large enough coefficient.

A second option is to exploit the availability of state-of-the-art MILP solvers to use the so-called indicator constraints, which are a modelling tool to express disjunctive conditions. We can then replace constraints (7) by

$$y_{ij} = 1 \Rightarrow a^{jT}_r\, x^i \geq b^j_r \qquad i = 1, \dots, K; j = 1, \dots, \ell; r = 1, \dots, m.$$

Preliminary computational experiments showed that, on our benchmark, the two approaches were comparable, though the former was more robust from a numerical viewpoint, and was adopted in subsequent experiments. Finally, observe that the objective function contains bilinear terms too. Each such term can be handled by introducing an additional variable and a bilinear constraint, to be linearized in a similar way. In this case, however, we experienced better performances by letting the solver to determine the best way to handle non-linearities.

B. Set Partitioning formulations

We now introduce two novel formulations for (DCmEm). Both formulations are pure 0-1 linear programs and involve an exponential number of variables.

Formulation 1. Let $\mathcal{F}$ be the family of all subsets of the scenarios. Each subset $S \in \mathcal{F}$ can be described by an ℓ-dimensional binary vector z whose j-th component is equal to 1 if scenario $j \in S$, and 0 otherwise. Given a subset $S \in \mathcal{F}$, the associated cost $C(S)$ is defined by

$$\begin{aligned} c(S) :=\ & \min \sum_{j \in S} \xi^{jT} x \\ \text{s.t.}\ & x \in X^j \qquad\qquad j \in S. \end{aligned} \tag{9}$$

In the remainder, we will assume that an oracle is available for solving subproblem (9). In addition, the solution $x(S)$ returned by the oracle will be denoted as the *solution induced* by S.

By introducing a binary variable ϑ_t for each feasible subset of scenarios $S_t \in \mathcal{F}$, we obtain the following model

$$\begin{aligned} \min\ & \sum_{t=1}^{|\mathcal{F}|} c_t\, \vartheta_t \\ \text{s.t.}\ & \sum_{t=1}^{|\mathcal{F}|} z_{jt}\, \vartheta_t = 1 \qquad j = 1, \dots, \ell \\ & \sum_{t=1}^{|\mathcal{F}|} \vartheta_t \leq K \\ & \vartheta_t \in \{0, 1\} \qquad t = 1, \dots, |\mathcal{F}|, \end{aligned} \tag{SPP1}$$

where $\vartheta_t = 1$ if and only if scenarios in subset S_t are covered by the same solution.

The model requires to select at most K subsets, so that each scenario belongs to a selected subset. By associating the i-th selected subset to solution x^i, we obtain a set of solutions for (DCmEm), whose cost is given by the sum of the costs of the selected subsets.

Formulation 2. We derive another formulation for (DCmEm) by working in the space of solutions. For each scenario j, let Q^j denote the set of all feasible solutions, and let $Q = \cup_j Q^j$. In addition, let r_{jp} be the cost of solution p in scenario j; possibly $r_{jp} = +\infty$ if $p \notin Q^j$. For each solution $p \in Q$, introduce a binary variable σ_p, taking value 1 if solution p is among the K chosen solutions, and 0 otherwise. In addition, for each solution p and scenario j, let $\rho_{jp} = 1$ be a binary variable taking value 1 if and only if scenario j is covered by solution p. Thus, (DCmEm) can be formulated as

$$
\begin{array}{lll}
\min & \displaystyle\sum_{j=1}^{\ell}\sum_{p=1}^{|Q|} r_{jp}\,\rho_{jp} & \\
\text{s.t.} & \displaystyle\sum_{p=1}^{|Q|} \rho_{jp} = 1 & j = 1,\dots,\ell \\
& \sigma_p \geq \rho_{jp} & j = 1,\dots,\ell; p = 1,\dots,|Q| \\
& \displaystyle\sum_{p=1}^{|Q|} \sigma_p \leq K & \\
& \sigma_p \in \{0,1\} & p = 1,\dots,|Q| \\
& \rho_{jp} \in \{0,1\} & j = 1,\dots,\ell; p = 1,\dots,|Q|.
\end{array}
\tag{SPP2}
$$

The first set of constraints ensures that every scenario is covered by exactly one solution, while the second and third sets of constraints ensure that at most K solutions are used.

Models SPP1 and SPP2 require enumeration of all the feasible subsets of scenarios and of all feasible solutions for every scenario, respectively. Enumeration can be performed implicitly by means of column generation techniques, as it will be described in Section IV B 1). We conclude this section by showing that the two set partitioning models have the same tightness in terms of continuous relaxation.

Theorem 7. Models SPP1 and SPP2 are equivalent in terms of continuous relaxation.

Proof. In the following we will denote by SPP1$_c$ and SPP2$_c$ the continuous relaxations of models SPP1 and SPP2, respectively. To prove the statement we show that, given an optimal solution of SPP1$_c$, there exists a feasible solution of SPP2$_c$ with the same value, and vice versa.

Given an optimal solution ϑ of SPP1$_c$, the corresponding solution for SPP2$_c$ is defined as follows:

```
1. for $p := 1$ to $|Q|$ do
       $\sigma_p := 0$;
       for $j := 1$ to $\ell$ do $\rho_{jp} := 0$;
   endfor
2. for each $S_t \in \mathcal{F}$ such that $\vartheta_t > 0$ do
       let $p$ be the solution induced by subset $S_t$;
       $\sigma_p := \sigma_p + \vartheta_t$;
       for $j := 1$ to $\ell$ do $\rho_{jp} := \rho_{jp} + z_{jt}\,\vartheta_t$
   endfor
```

Step **1** initializes an empty solution, while the second step defines the correct value for the σ and ρ variables. In this step, only subsets S_t that are actually selected in solution ϑ are taken into account. For each such subset, the associated induced solution p is considered, i.e., an optimal solution of subproblem (9). Every positive value ϑ_t is used to increase the value of a single σ variable, hence $K = \sum_{t=1}^{|\mathcal{F}|} \vartheta_t = \sum_{p=1}^{|Q|} \sigma_p$. Similarly, for each scenario j, each variable ρ_{jp} is increased by ϑ_t when considering a subset S_t that (i) is selected; (ii) includes scenario $j \in S_t$ (i.e., $z_{jt} = 1$); and (iii) induces solution p. It follows that $1 = \sum_{t=1}^{|\mathcal{F}|} z_{jt}\,\vartheta_t = \sum_{p=1}^{|Q|} \rho_{jp}$, hence solution (σ,ρ) is feasible to SPP2$_c$. Finally, consider a selected subset S_t, which contributes with a cost $c_t\vartheta_t$ to the objective function of SPP1$_c$. Let p be the associated induced solution and note that, by definition, we have $c_t = \sum_{j\in S_t} r_{jp}$. Increasing by ϑ_t each variable ρ_{jp} $(j \in S_t)$ produces a cost increase equal to $\sum_{j\in S_t} r_{jt}\,\vartheta_t = \vartheta_t \sum_{j\in S_t} r_{jt} = c_t\,\vartheta_t$ in the objective function of SPP2$_c$. Hence, the two solutions have the same cost.

Assume now that (σ,ρ) is an optimal solution for SPP2$_c$. The following procedure defines a solution ϑ for SPP1$_c$:

```
1. for each $S_t \in \mathcal{F}$ do $S_t := 0$;
2. for $p := 1$ to $|Q|$ do
       for $j := 1$ to $\ell$ do $\bar{\rho}_{jp} := \rho_{jp}$;
```

```
        while there exists a $j \in \{1, \dots, \ell\}$ such that $\bar{\rho}_{jp} > 0$ do
            let $S_t = \{j : \bar{\rho}_{jp} > 0\}$, and $\rho_{\min} := \min_{j \in S_t}\{\bar{\rho}_{jp}\}$;
            $\vartheta_t := \vartheta_t + \rho_{\min}$;
            for each $j \in S_t$ do $\bar{\rho}_{jp} := \bar{\rho}_{jp} - \rho_{\min}$
        endwhile
    endfor
```

By optimality of (σ, ρ), each set S_t of scenarios defined by the procedure is assigned to its optimal induced solution p. As c_t is the sum of the costs of solution p in the included scenarios, SPP2$_c$ and SPP1$_c$ have the same solution value. By construction, for each $p \in Q$, we have

$$\sum_{t:p \text{ optimal for } S_t} \vartheta_t = \sigma_p = \max_{j \in \{1,\dots,\ell\}} \rho_{jp},$$

which makes the sum of all the ϑ_t equal to K. In addition, for each scenario j, we have $\sum_{t:j \in St} \vartheta_t = \sum_{p=1}^{|Q|} \rho_{jp} = 1$. Hence solution ϑ is feasible for SPP1$_c$. $\square$

IV. EXACT ALGORITHMS

Solving the set partitioning formulations of Section III B requires to enumerate all possible subsets of scenarios, which is possible only for instances with small size. In this section we describe two alternative exact methods for solving instances that cannot be attacked by the direct application of an MILP solver.

A. Enumeration of partitions

The first exact method is an extension of an approach introduced in [7] for the (mEm). This algorithm is proven to be an oracle-polynomial time algorithm if $\ell - K$ is fixed, i.e., it provides a positive complexity result in this special case of (DCmEm). The proposed enumerative approach is based on the following consideration: let P be a partition of the scenarios in K subsets $S_1, S_2, \dots, S_K$. Then, an optimal solution for (DCmEm) can be determined by computing K solutions, the i-th induced by subset S_i according to (9). The K solutions induced by the subsets constitute a set of solutions for (DCmEm); with an abuse of notation, we denote this set as the *solution induced* by P.

Using this observation, one can design an exact algorithm that defines all possible partitions of scenarios into K subsets and, for each partition P, computes the induced solution. It is easy to see that, to avoid symmetric partitions and empty subsets (which are unnecessary to define an optimal solution), it is enough to consider "only" $S_{\ell,K}$ partitions, where $S_{\ell,K}$ is the Sterling number of the second kind.

This algorithm has two main pitfalls. First, the number of partitions to be considered can be very large, in particular when $\ell - K$ is large. In addition, the determination of the solution induced by a subset of scenarios requires to solve a problem that does not have the same structure as the deterministic underlying problem, and may be much more challenging than the latter from a computational viewpoint. For example, if the underlying problem is a knapsack problem, then the problem to be solved is a multidimensional knapsack problem, in which different knapsack constraints (one for every scenario) have to be considered.

B. Branch-and-price algorithm

In this section we present an exact algorithm based on the first set partitioning formulation introduced in Section III B, which is more convenient for generating columns than the second one. Section IV B 2 describes a branch-and-price scheme that uses, at each node, the algorithm described in the next section for computing a lower bound by solving the LP relaxation of the model.

1. Column generation

By dropping the binary requirements, the domain of the variables of model SPP1 can be replaced by $\vartheta_t \geq 0$ ($t = 1, \ldots, |\mathcal{F}|$), and hence the dual of the resulting model is:

$$\begin{array}{lll} \max & \sum_{j=1}^{\ell} \lambda_j + K & \mu \\ \text{s.t.} & \sum_{j=1}^{\ell} z_{jt}\lambda_j + & \mu \leq c_t \quad t = 1, \ldots, |\mathcal{F}| \\ & & \mu \geq 0, \end{array}$$

where λ_j ($j = 1, \ldots, \ell$) and μ are the dual variables associated with the constraints of SPP1.

Column generation defines a *restricted master* problem, in which a subset of the ϑ_t variables is used, and solves this continuous problem to optimality. Given the associated dual variables, column generation asks for a variable (*column*) that has a negative reduced cost, i.e., whose associated dual constraint is violated by the current dual solution. For a given subset S of scenarios, the associated dual constraint is violated if

$$\sum_{j \in S} \lambda_j + \mu > c(S)$$

where $c(S) = \sum_{j \in S} {\xi^j}^T x(S)$ is the cost of the induced solution. This solution must satisfy all constraints of the scenarios in the subset. The problem of determining the subset S (if any) whose associated dual constraint is maximally violated can be formulated by introducing, for each scenario j, a binary variable π_j taking value 1 if and only if scenario j belongs to subset S. The reduced cost $\bar{c}(S)$ of this subset is given by the optimal solution of the following problem

$$\begin{array}{lll} \bar{c}(S) = \min & \sum_{j=1}^{\ell} ({\xi^j}^T x - \lambda_j)\pi_j - \mu & \\ \text{s.t.} & \pi_j = 1 \quad \Rightarrow \quad x \in X^j & j = 1, \ldots, \ell \\ & \pi_j \in \{0, 1\} & j = 1, \ldots, \ell. \end{array} \tag{10}$$

If the optimal solution of the model has a negative value, then the subset of scenarios $S = \{j : \pi_j = 1\}$ corresponds to a variable with negative reduced cost and should be added on-the-fly to the current restricted master.

The model above includes a set of non-linear constraints that can be either linearized using a BIGM coefficient or formulated as indicator constraints. In our experiments with a general-purpose solver, we experienced better performances using the first strategy. Note that the pricing problem is $\mathcal{NP}$-hard, as shown the following reduction from the Completion problem (see Section II B).

Theorem 8. The pricing problem (10) is equivalent to Completion problem with $K_f = \bar{K} = 1$.

Proof. We can reformulate the objective function of the pricing problem as

$$\min \sum_{j=1}^{\ell} {\xi^j}^T x \pi_j + \sum_{j=1}^{\ell} \lambda_j (1 - \pi_j) - \sum_{j=1}^{\ell} \lambda_j - \mu$$

where the last two terms are constant and do not depend on the variables.

We now show that the pricing problem can be reduced to the Completion problem. Given an instance of the former, we define an instance of the latter with $n+1$ variables and $\ell + 1$ scenarios. Every scenario $j \leq \ell$ gets the feasible set $(X^j \cup \{0\}^n) \times \{0, 1\}$. As to the objective function, it has a coefficient ξ_h^j for each variable x_h ($j = 1, \ldots, \ell; h = 1, \ldots, n$), and a coefficient λ_j for variable x_{n+1}. The last scenario has a feasible set $X^{\ell+1} = \{0, 1\}^{n+1}$, zero objective function for the first n variables, and a coefficient equal to BIGM for the last variable. Finally, we set the fixed solution as follows: $\bar{x}_h = 0$ for $h = 1, \ldots, n$ and $\bar{x}_{n+1} = 1$.
Consider a scenario j covered by the fixed solution: the contribution to the objective value is λ_j if $j \leq \ell$, and BIGM otherwise. Let x be the free solution determined solving the Completion problem with $K_f = \bar{K} = 1$. Assuming a sufficiently large value BIGM, the free solution must have $x_{n+1} = 0$, i.e., the solution covers the last scenario with zero cost. For the remaining scenarios $j \in \{1, \ldots, \ell\}$, the free solution has a cost ${\xi^j}^T x$, i.e., each scenario j will be covered by the free solution x if $x \in X^j$ and ${\xi^j}^T x < \lambda_j$ holds and by the fixed solution otherwise. In other words, the scenarios that are covered by the free solution determine the set S that corresponds to the optimal solution of the pricing problem.

Now we want to reduce the Completion problem to the pricing problem. To this aim, given a fixed solution $\bar{x}$, we set $\lambda_j = {\xi^j}^T \bar{x}$ for each scenario j, and use the same set of scenarios in both problems. It is easy to see that the objective functions of the two problems are equivalent to each other and solution x_f is equivalent to x. □ □

```
Algorithm Heur_Pricing
Sample at random $p_1$ subsets of scenarios $S_1, \ldots, S_{p_1}$
for $q = 1$ to $p_1$ do
    compute an approximate reduced cost $\bar{l}(S_q)$;
endfor
sort the subsets by non-decreasing $\bar{l}(S)$ values;
for $q = 1$ to $p_2$ do
    if $\bar{l}(S_q) < 0$ then
        compute the reduced cost $\bar{c}(S_q)$;
        if $\bar{c}(S_q) < 0$ then add a new variable to the restricted master
    endif
return
```

FIG. 1: Heuristic algorithm for pricing.

As the pricing problem is in general $\mathcal{NP}$-hard, it makes sense to solve it using a heuristic algorithm, resorting to an exact method only in case the former failed in producing a variable with negative reduced cost. Our heuristic algorithm for pricing, described in Figure 1, takes in input two integer parameters p_1 and p_2. The algorithm randomly selects p_1 subsets of scenarios (chosen with equal probability) and checks whether one of them corresponds to a column with negative reduced cost. As already observed, evaluating the reduced cost of a subset S requires to compute the solution induced by S, which can be time consuming in practice. For this reason, the following heuristic rule is used to reduce the number of subsets that are evaluated: for each candidate subset, we determine an approximated reduced cost, and compute the real reduced cost for the most promising p_2 subsets only. The reduced cost of a subset S is given by $-\mu - \sum_{j \in S} \lambda_j + \sum_{j \in S} {\xi^j}^T x(S)$, where $x(S)$ is the solution induced by S. Hence, an approximate algorithm for determining this solution produces an approximate value for the reduced cost as well. The design of the specific algorithm used to compute an approximate induced solution depends on the underlying problem.

As a further improvement to the pricing step, note that the reduced cost of a column (variable) consists of the the costs of the induced solutions and of a linear term depending on the dual variables. Computing the first part can be very time consuming depending on the underlying problem, whereas the second part can be computed with reduced computational effort. Observe that only the second part of the reduced costs changes through the column generation process. Therefore, by creating a hash table with sets of scenarios as keys and the corresponding induced solutions and costs as values, one can compute "for free" the reduced cost of a column involving a subset of the scenarios that occurred before. Every time an induced solution is computed, we store it and its cost in this table. Before using the heuristic, we compute the reduced cost for each variable corresponding to a subset of scenarios stored in the hash table; if a variable with negative reduced cost is found, we add it to the master problem, which is then re-optimized. Our computational experiments (see Section VI) show that this may have a dramatic impact in the performances of the overall procedure, mainly when the underlying problem is hard to solve. Using hash tables makes the column generation procedure faster and faster the more iterations are executed. As a consequence, the method shows a speed-up during the exploration of the enumeration tree in the branch-and-price algorithm because large parts of these hash tables can be passed on from the parent node to a child node.

2. Branching scheme

The algorithm for computing the LP relaxation of model SPP1 has been embedded into a branch-and-price algorithm. At the root node, a feasible solution is computed using a fast heuristic algorithm. At each node, the continuous relaxation of the current subproblem is solved, producing a lower bound on the optimal solution value of the current subproblem. If the solution of the relaxation is integer, the incumbent may be improved and the node is fathomed. Otherwise, if the lower bound is smaller than the incumbent value, the optimal solution of the continuous relaxation is used to branch, producing two subproblems that are explored according to a depth-first strategy. In particular, let a be a scenario that is included in more than one subset that is selected in the current fractional solution (note that this scenario always exists in a non-integer solution). Let S_1 and S_2 be two of these subsets, and let b be a scenario that belongs to the symmetric difference of S_1 and S_2. In the first node we impose the scenarios a and b belong to the same subset, while in the other node we forbid it. Observe that these branching rules affect the pricing subproblems at the descendant nodes. However, a nice property of this branching scheme is that handling these modification is very easy both in the heuristic generation, and in the exact model. Indeed, in the latter case, it is enough to enforce in (10) the additional constraints $\pi_a = \pi_b$ for the first node, and $\pi_a + \pi_b \leq 1$ for the second one.

In general, we have a number of candidate scenarios for a and b, thus we use the following tie breaking rules. For each scenario j, we define a score

$$\texttt{score}^1_j = \sum_{t: j \in S_t, \vartheta_t > 0} \min\{1 - s_t, s_t\},$$

that gives a measure of the distance to the closest integer of all the variables associated with subsets that contain j, and select the scenario a having the maximum score.

For a given a, we assign to each scenario a second score

$$\texttt{score}^2_j = |\{t : \vartheta_t > 0, a \in S_t, j \notin S_t\}| - |\{t : S_t > 0, a \in S_t, j \in S_t\}|,$$

and take the scenario b that maximizes this figure. In this way, we define a branching which balances (among the descendant nodes) the number of subsets that were in the solution and that violate branching conditions.

V. APPROXIMATION

We now present some results concerning the approximation of (DCmEm). Given the negative results on approximability in [7], we show that, replacing the exact oracle for problem (9) with an approximation algorithm yields an approximation algorithm for (DCmEm) as well.

Theorem 9. Assuming that all the optimal values of the oracle are either always non-negative or always non-positive, using an α-approximation as oracle gives an α-approximation algorithm with respect to the algorithm embedding an exact oracle.

Proof. Without loss of generality, we assume that all the optimal values of the oracle are non-negative and that we have a minimization problem. (A similar proof can be given in case of non-positive optimal values).

Consider an arbitrary partition $\widetilde{P} = \{S_1, S_2, \dots, S_K\}$ of the scenarios, and let $val_\alpha(\widetilde{P})$ be the solution value obtained solving each problem (9) through an α-approximation algorithm. In addition, let $val_1(\widetilde{P})$ denote the solution value when the exact oracle is used.
By definition of α-approximation, for each subset S of scenarios in the partition, we have $val_\alpha(S) \leq \alpha \cdot val_1(S)$, and hence

$$val_\alpha(\widetilde{P}) = \sum_{S \in \widetilde{P}} val_\alpha(S) \leq \sum_{S \in \widetilde{P}} \alpha \cdot val_1(S) = \alpha \cdot val_1(\widetilde{P})$$

Denote now by $val^*_\alpha = \min_P val_\alpha(P)$ the best solution value, over all possible partitions, using the α-approximation algorithm. We have

$$val^*_\alpha \leq val_\alpha(\widetilde{P}) \leq \alpha \cdot val_1(\widetilde{P}) \tag{11}$$

Finally, let val^*_1 denote the optimal solution value for the problem, i.e., the solution obtained solving with an exact oracle the subproblems associated with an optimal partitioning.
Observe that (11) is valid for any partition $\widetilde{P}$. Using an optimal partition, we get $val^*_\alpha \leq \alpha \cdot val^*_1$, which concludes the proof. □ □

Note that Theorem 9 requires that the optimal values of the subproblems are all non-negative, a typically mild assumption in practice. However, the theorem does not imply the existence of a polynomial time α-approximation algorithm for (DCmEm), neither in case a polynomial time α-approximation for subproblem (9) is available, as (DCmEm) can not be solved in oracle polynomial time in general (and is $\mathcal{NP}$-hard even for underlying problems that are solvable in polynomial time).

Theorem 10. Replacing the exact oracle with an algorithm having absolute approximation $OPT + c$ yieds an algorithm for (DCmEm) having absolute approximation $(OPT + K \cdot c)$.

Proof. The proof is similar to that of Theorem 9. The only difference is that, for each subset S of scenarios, the algorithm has an absolute approximation, i.e., the associated solution value is

$$val_\alpha(S) \leq c + val_1(S)$$

where c is a positive constant. Using the approximation algorithm for all subsets of scenarios we get

$$val_\alpha(\widetilde{P}) = \sum_{S \in \widetilde{P}} val_\alpha(S) \leq \sum_{S \in \widetilde{P}} (c + val_1(S)) = K \cdot c + val_1(\widetilde{P})$$

□ □

Note that in the case of an absolute approximation, we do not need to make the assumption that all the optimal values of the oracle are non-negative or non-positive.

VI. COMPUTATIONAL EXPERIMENTS

The described algorithms were implemented in Java version 1.8.0_212 and executed on machines using Intel Xeon processors with 2.6 GHz. For all the linear integer programming problems we used IBM-ILOG Cplex version 12.9.

In our experiments we considered the following three different underlying problems with increasing level of complexity:

MFP: Maximum Flow Problem. For these problems, we always used a quadratic grid graph as network. The capacity of the arcs were generated in each scenario as equally independent distributed rational numbers between 0.0 and 1.0, while the entries of each cost vector ξ were set to 1.0 if the corresponding arc leaves the source and 0.0 otherwise. All numbers were rounded to the second digit;

1KP: Knapsack Problems. In each scenario, the profits and weights of the items were randomly generated as uncorrelated uniformly distributed values between 0.0 and 1.0. The capacity value was always set to $0.75W$, where W denotes the current sum of the weights of the items. In this case, all values were rounded to the third digit;

MKP: Multidimensional Knapsack Problems. We considered a variant of 1KP in which there are 5 knapsack constraints. The profit, weight and capacity values for each scenario were generated as in the knapsack case;

In this section we compare exact algorithms for (DCmEm) on instances defined with $K \in \{2, 3, 4, 5\}$ and $\ell \in \{15, 20, 25\}$. For the Maximum Flow Problems, variables are associated with arcs, and we used $n \in \{24, 40\}$. For 1KP and MKP instances, variables are associated with items, and we used $n \in \{10, 15\}$. In all cases, for each combination of parameters K, ℓ, and n, we randomly generated 10 instances.

We compare the following four exact algorithms, each executed with a time limit of 600 seconds per instance:

- Compact Formulation: direct application of the solver to the compact formulation (5)–(8). In this case, we used BIGM constraints instead of indicator constraints, and to let the solver handle the the linearization of the objective function;
- Set Partitioning: direct application of the solver to the set partitioning formulation SPP1. In our implementation, we spent at most 90% of the time for subsets enumeration; in case this limit is reached, the solver is applied to a restricted formulation, thus producing a heuristic approach;
- Enum. of Partitions: enumeration of partitions according to the procedure described in Section IV A;
- Branch-and-price: enumerative algorithm based on column generation, as described in Section IV B.

Tables I - III - report the results for the different classes of instances. Each table reports, for each algorithm:

- value: average ratio (over 10 instances) between the solution produced by the algorithm and the best known solution for each instance;
- # TL: number of instances (out of 10) for which the algorithm hit the time limit;
- time: average computing time (with respect to the instances solved to optimality only).

The results in Table I show that, on the Maximum Flow Problem instances, the compact formulation is the best approach for $K \leq 3$. Increasing the value of K, this method could solve to proven optimality only a few instances with small values of K and/or ℓ, though it produced high-quality solutions in the other cases. The branch-and-price algorithm is the best approach for $K \geq 4$. It is the only algorithm that is able to solve instances of all settings. Finally, note that the enumeration of partitions is able to solve a large fraction of the instances. However, the computing time required by this algorithm is typically larger than that for set partitioning and branch-and-price. Also, it strongly depends on the number of scenarios, and the approximation given for unsolved instances is unsatisfactory in some cases.

For the single Knapsack Problem (see Table II) the situation is similar: the compact formulation has good performances for most settings with $K = 2$ and for $K = 3$, when $\ell = 15$, while this algorithm becomes unpractical for larger values of K. For most of the other settings branch-and-price outperforms the other algorithms. Observe that there are some instances for which Set Partitioning does not reach the global time limit but hits the time limit imposed on the subsets enumeration phase, which prevents the possibility to compute an optimal solution.

The results show that an increase of ℓ has a higher impact on the running time of all algorithms apart the branch-and-price algorithm. Therefore, we expect that branch-and-price algorithm could outperform the compact formulation even more in case of larger instances and could also be able to be better in some instances with $K = 2$ in case a larger time limit is used. However, when both algorithms hit the time limit, frequently the compact formulation has a slightly better solution value than the branch-and-price algorithm. This is due to the lack of heuristics in the latter approach, while the former takes advantage of the heuristic algorithms that the MILP solver may use on a compact formulation. The integration of heuristic algorithms at the various nodes of the branch-and-price algorithm could improve the performances of this method, but this is outside the scope of the current paper.

Finally, Table III illustrates the results of the tests for the Multidimensional Knapsack Problem. In this case too, the compact formulation is the best exact algorithm for $K \leq 3$ and $\ell = 15$ only. The enumeration of partitions solves to optimality only the cases with $\ell = 15$. In all the other settings the branch-and-price is the best algorithm.

VII. CONCLUSIONS

We considered stochastic problems in which recourse actions can be taken after the exact nature of uncertainty materializes, and one is allowed to implement a specific solution chosen among K that have been computed in advance. The resulting K-adaptability paradigm is extremely challenging both from a theoretical and from a computational viewpoint: we proved that even determining if the problem has a solution is $\mathcal{NP}$-complete, also in case the underlying problem can be solved in polynomial time. We introduced mathematical formulations of the problem and exact

TABLE I: Results for the exact methods for the Maximum Flow Problem

				Compact Formulation			Set Partitioning			Enum. of Partitions			Branch-and-price		
K	ℓ	n	$\lvert V\rvert$	value	# TL	time	value	# TL	time	value	# TL	time	value	# TL	time
2	15	24	16	1.000	0	0.0	1.000	0	17.8	1.000	0	15.6	1.000	0	13.9
2	15	40	25	1.000	0	0.0	1.000	0	18.3	1.000	0	15.9	1.000	0	16.8
2	20	24	16	1.000	0	0.0	1.000	0	344.0	1.000	0	306.2	1.000	0	43.7
2	20	40	25	1.000	0	0.7	1.000	0	398.7	1.000	0	363.6	1.000	0	42.6
2	25	24	16	1.000	0	0.7	–	10	–	0.807	10	–	1.000	0	107.6
2	25	40	25	1.000	0	1.9	–	10	–	0.705	10	–	1.000	0	78.6
3	15	24	16	1.000	0	2.8	1.000	0	17.6	1.000	0	17.7	1.000	0	13.8
3	15	40	25	1.000	0	5.5	1.000	0	15.7	1.000	0	18.4	1.000	0	15.0
3	20	24	16	1.000	0	32.1	1.000	0	355.4	0.919	10	–	1.000	0	59.7
3	20	40	25	1.000	0	37.7	1.000	0	411.9	0.987	10	–	1.000	0	80.1
3	25	24	16	1.000	0	137.5	–	10	–	0.741	10	–	1.000	2	319.0
3	25	40	25	1.000	3	272.7	–	10	–	0.743	10	–	1.000	1	280.3
4	15	24	16	1.000	0	21.1	1.000	0	17.4	1.000	0	78.9	1.000	0	13.4
4	15	40	25	1.000	0	92.4	1.000	0	15.7	1.000	0	83.7	1.000	0	23.9
4	20	24	16	0.990	7	285.3	0.937	1	379.7	0.754	10	–	1.000	0	153.3
4	20	40	25	0.998	9	93.0	1.000	0	412.7	0.816	10	–	1.000	0	83.7
4	25	24	16	0.995	10	–	0.117	10	–	0.668	10	–	1.000	5	375.8
4	25	40	25	0.995	10	–	0.234	10	–	0.678	10	–	1.000	2	400.1
5	15	24	16	1.000	4	45.3	1.000	0	17.6	1.000	0	384.5	1.000	0	13.4
5	15	40	25	1.000	7	60.3	1.000	0	15.4	1.000	0	394.9	1.000	0	15.1
5	20	24	16	0.997	10	–	1.000	0	353.9	0.721	10	–	1.000	0	124.1
5	20	40	25	0.986	10	–	1.000	0	407.7	0.769	10	–	1.000	0	86.8
5	25	24	16	1.000	10	–	0.181	10	–	0.663	10	–	0.915	4	363.5
5	25	40	25	0.991	10	–	0.428	10	–	0.661	10	–	0.972	3	335.1

TABLE II: Results for the exact methods for the Knapsack Problem

			Compact Formulation			Set Partitioning			Enum. of Partitions			Branch-and-price		
K	ℓ	n	value	# TL	time	value	# TL	time	value	# TL	time	value	# TL	time
2	15	10	1.000	0	0.5	1.000	0	7.8	1.000	0	7.7	1.000	0	10.2
2	15	15	1.000	0	2.5	1.000	0	9.7	1.000	0	10.7	1.000	0	25.3
2	20	10	1.000	0	21.0	1.000	0	239.6	1.000	0	236.9	1.000	0	102.8
2	20	15	1.000	0	71.2	1.000	0	407.9	1.000	0	403.7	0.999	3	300.7
2	25	10	1.000	2	356.4	–	10	–	0.980	10	–	0.996	4	319.0
2	25	15	1.000	10	–	–	10	–	0.990	10	–	0.994	10	–
3	15	10	1.000	0	2.4	1.000	0	7.6	1.000	0	7.8	1.000	0	5.4
3	15	15	1.000	0	7.5	1.000	0	9.4	1.000	0	10.7	1.000	0	12.7
3	20	10	1.000	0	209.1	1.000	0	240.5	1.000	0	238.3	1.000	0	95.4
3	20	15	0.997	7	378.7	1.000	0	408.6	1.000	0	407.3	1.000	0	267.7
3	25	10	0.995	10	–	0.991	1	545.0	0.987	10	–	1.000	5	305.8
3	25	15	0.998	10	–	–	10	–	0.992	10	–	0.995	9	230.0
4	15	10	1.000	0	4.3	1.000	0	7.1	1.000	0	7.2	1.000	0	4.0
4	15	15	1.000	0	18.2	1.000	0	8.8	1.000	0	8.5	1.000	0	4.4
4	20	10	0.998	7	311.7	1.000	0	237.8	1.000	0	298.9	1.000	0	48.5
4	20	15	0.996	10	–	1.000	0	406.6	1.000	0	467.5	1.000	0	132.2
4	25	10	0.996	10	–	0.999	0	546.6	0.978	10	–	1.000	0	183.9
4	25	15	0.994	10	–	0.999	0	543.3	0.978	10	–	0.995	8	313.0
5	15	10	1.000	0	6.1	1.000	0	5.5	1.000	0	4.9	1.000	0	2.2
5	15	15	1.000	0	26.9	1.000	0	7.5	1.000	0	8.4	1.000	0	2.8
5	20	10	0.996	8	430.0	1.000	0	232.6	1.000	6	576.2	1.000	0	9.2
5	20	15	0.994	10	–	1.000	0	396.7	1.000	10	–	1.000	0	49.3
5	25	10	0.988	10	–	1.000	0	543.4	0.969	10	–	1.000	0	80.3
5	25	15	0.987	10	–	1.000	0	542.1	0.978	10	–	0.999	4	352.7

solution approaches. Finally, an extensive computational analysis has been carried out on a large set of randomly generated instances.

Acknowledgements

This work has been supported by the German Research Foundation within the Research Training Group 1855. The authors are grateful to Christoph Buchheim for stimulating discussion on the topic.

[1] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization.* Princeton series in applied mathematics. Princeton university press, 2009.

[2] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23:769–805, 1998.

[3] D. Bertsimas and C. Caramanis. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55:2751–2766, 2010.

[4] D. Bertsimas and V. Goyal. On the power of robust solutions in two-stage stochastic and adaptive optimization problems. *Mathematics of Operations Research*, 35:284–305, 2010.

[5] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52:35–53, 2004.

[6] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming.* Springer Series in Operations Research and Financial Engineering. Springer-Verlag, 1997.

[7] C. Buchheim and J. Pruente. k-adaptability in stochastic combinatorial optimization under objective uncertainty. *European Journal of Operational Research*, 277:953–963, 2019.

[8] M. Fischetti and M. Monaci. Light robustness. In R.K. Ahuja, R. Möhring, and C. Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, Lecture Notes in Computer Science 5868, pages 61–84. Springer-Verlag, Berlin Heidelberg, 2009.

[9] G.A. Hanasusanto, D. Kuhn, and W. Wiesemann. k-adaptability in two-stage robust binary programming. *Operations Research*, 63:877–891, 2015.

TABLE III: Results for the exact methods for the Multidimensional Knapsack Problem with 5 constraints

			Compact Formulation			Set Partitioning			Enum. of Partitions			Branch-and-price		
K	ℓ	n	value	# TL	time	value	# TL	time	value	# TL	time	value	# TL	time
2	15	10	1.000	0	6.7	1.000	0	20.9	1.000	0	21.3	1.000	0	32.8
2	15	15	1.000	0	22.0	1.000	0	23.4	1.000	0	22.1	1.000	0	45.9
2	20	10	1.000	1	374.9	0.999	0	540.3	0.998	10	–	0.999	4	315.2
2	20	15	1.000	4	443.0	0.998	0	540.4	0.998	10	–	0.999	3	379.3
2	25	10	1.000	7	412.0	–	10	–	0.985	10	–	0.998	6	453.0
2	25	15	0.999	10	–	–	10	–	0.979	10	–	0.989	10	–
3	15	10	1.000	0	21.2	1.000	0	20.7	1.000	0	20.6	1.000	0	32.4
3	15	15	1.000	0	51.1	1.000	0	23.1	1.000	0	21.7	1.000	0	53.5
3	20	10	0.995	10	–	1.000	0	541.3	0.998	10	–	1.000	2	256.5
3	20	15	0.996	10	–	0.999	0	540.8	0.998	10	–	0.999	2	264.4
3	25	10	0.997	10	–	–	10	–	0.982	10	–	1.000	8	410.0
3	25	15	1.000	10	–	–	10	–	0.990	10	–	0.993	10	–
4	15	10	1.000	0	38.2	1.000	0	19.6	1.000	0	18.7	1.000	0	22.3
4	15	15	1.000	0	108.4	1.000	0	21.8	1.000	0	20.7	1.000	0	22.9
4	20	10	0.987	10	–	1.000	0	540.8	0.999	10	–	1.000	1	183.9
4	20	15	0.988	10	–	1.000	0	540.4	0.999	10	–	0.999	1	267.6
4	25	10	0.984	10	–	0.997	0	542.1	0.969	10	–	0.998	8	254.5
4	25	15	0.993	10	–	0.999	0	541.9	0.983	10	–	0.993	9	476.0
5	15	10	1.000	0	42.1	1.000	0	16.5	1.000	0	16.1	1.000	0	11.0
5	15	15	1.000	0	127.1	1.000	0	18.7	1.000	0	17.4	1.000	0	11.9
5	20	10	0.993	10	–	1.000	0	540.5	0.997	10	–	1.000	0	101.2
5	20	15	0.989	10	–	1.000	0	540.9	0.999	10	–	1.000	0	145.3
5	25	10	0.985	10	–	0.999	0	541.1	0.967	10	–	0.998	2	300.2
5	25	15	0.986	10	–	1.000	0	541.0	0.978	10	–	0.993	8	361.0

[10] C. Liebchen, M. Lübbecke, R.H. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In R.K. Ahuja, R. Möhring, and C. Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, Lecture Notes in Computer Science 5868, pages 1–27. Springer-Verlag, Berlin Heidelberg, 2009.

[11] A. Prékopa. Programming under probabilistic constraint and maximizing probabilities under constraints. In *Stochastic Programming*, Mathematics and Its Applications 324, pages 319–371. Springer, Dordrecht, 1995.

[12] A. Ruszczynski and A. Shapiro. *Stochastic Programming.* Hanbooks in Operations Research and Management Science. Elsevier, 2003.

[13] A. Schöbel. Generalized light robustness and the trade-off between robustness and nominal quality. *Mathematical Methods of Operations Research*, 80:161–191, 2014.

[14] A.L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21:1154–1157, 1973.

[15] A. Subramanyam, C.E. Gounaris, and W. Wiesemann. k-adaptability in two-stage mixed-integer robust optimization. *Mathematical Programming Computation*, 21:1154–1157, 2019.