Salo, Ahti; Andelmin, Juho; Oliveira, Fabricio

Decision programming for mixed-Integer multi-stage optimization under uncertainty

Published in:
European Journal of Operational Research

DOI:
10.1016/j.ejor.2021.12.013

Published: 01/06/2022

Document Version
Publisher's PDF, also known as Version of record

European Journal of Operational Research 299 (2022) 550–565

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Stochastics and Statistics

Decision programming for mixed-integer multi-stage optimization under uncertainty

Ahti Salo*, Juho Andelmin, Fabricio Oliveira

Department of Mathematics and Systems Analysis, Aalto University School of Science, P.O. Box 11100, Aalto 00076, Finland

ARTICLE INFO

Article history:
Received 24 August 2020
Accepted 7 December 2021
Available online 11 December 2021

Keywords:
Decision analysis
Influence diagrams
Decision trees
Contingent portfolio programming
Stochastic programming

ABSTRACT

Influence diagrams are widely employed to represent multi-stage decision problems in which each decision is a choice from a discrete set of alternative courses of action, uncertain chance events have discrete outcomes, and prior decisions may influence the probability distributions of uncertain chance events endogenously. In this paper, we develop the *Decision Programming* framework which extends the applicability of influence diagrams by developing mixed-integer linear programming formulations for such problems. In particular, Decision Programming makes it possible to (i) solve problems in which earlier decisions cannot necessarily be recalled later, for instance, when decisions are taken by agents who cannot communicate with each other; (ii) accommodate a broad range of deterministic and chance constraints, including those based on resource consumption, logical dependencies or risk measures such as Conditional Value-at-Risk; and (iii) determine all non-dominated decision strategies in problems which multiple value objectives. In project portfolio selection problems, Decision Programming allows scenario probabilities to depend endogenously on project decisions and can thus be viewed as a generalization of *Contingent Portfolio Programming* (Gustafsson & Salo, 2005). We present several illustrative examples, evidence on the computational performance of Decision Programming formulations, and directions for further development.

1. Introduction

Influence diagrams, in their many variants (see, e.g., Bielza, Gómez, & Shenoy, 2011; Diehl & Haimes, 2004; Díez, Luque, & Bermejo, 2018; Howard & Matheson, 1984; Howard & Matheson, 2005), are widely employed to represent decision problems whose consequences depend on uncertain chance events and decisions. Specifically, such decisions and chance events are represented by decision and chance nodes in an acyclic graph whose arcs indicate (i) what information is available to the decision maker (DM) and (ii) how realizations of chance events depend on earlier decisions and chance events. The value node represents consequences associated with the DM's decisions and the realization of chance events. Risk preferences are typically modeled with a utility function over the set of consequences.

The optimal solution to the influence diagram is the strategy that, at each decision node, assigns one of the decision alternatives to every possible state of information at the node so that the combination of these decisions maximizes the DM's expected utility. If the diagram fulfills the 'no-forgetting' assumption, meaning that earlier decisions can be recalled when making later ones (see, e.g., Jorgensen, Kristensen, & Nilsson, 2014; Lauritzen & Nilsson, 2001), this optimal strategy can be computed with well-established techniques, for example by carrying out local transformations such as arc reversals and node removals (Shachter, 1986; 1988), or by formulating the equivalent decision tree representation and solving it with dynamic programming (Tatman & Shachter, 1990). Mathematically, dynamic programming is based on the principle of optimality (see, e.g., Bertsekas, 2012) which allows the problem to be tackled by solving a sequence of nested subproblems whose solutions coincide with the corresponding optimal decisions in the original problem. This makes it possible to develop computationally efficient solution approaches to problems in which the principle of optimality holds.

Yet, while the 'no-forgetting' assumption often holds, there are important problems in which it does not. For example, in distributed decision making and adversarial risk analysis (Rios Insua, Rios, & Banks, 2009; Roponen, Ríos Insua, & Salo, 2020) there can be agents such as military patrols who cannot communicate with each other (for examples, see, e.g., Zhang, 1994; Zhang, Qi, & Poole,

* Corresponding author.:.
E-mail address: ahti.salo@aalto.fi (A. Salo).

https://doi.org/10.1016/j.ejor.2021.12.013

1994). Moreover, the adequate performance of safety-critical systems must be ensured even in situations where it may be impossible to synchronize information due to disruptions or communication delays, which makes it necessary to assess how system performance is affected by information sharing. In these systems, it is also crucial to pay attention to the full probability distribution over consequences (as opposed to expected consequences only), because high consequences with low probabilities are often of greatest concern (see, e.g., Mancuso, Compare, Salo, & Zio, 2019). Importantly, if the 'no-forgetting' assumption does not hold, then the principle of optimality breaks down and dynamic programming cannot be applied as usual, because the optimal strategy within a given branch of the decision tree depends on what decisions are taken in the other, non-overlapping branches of the decision tree. Furthermore, the consideration of risk measures such as Value-at-Risk, which reflect the full variability of consequences across the entire decision tree, also undermines this principle and consequently optimal solutions for the original problem cannot be obtained by combining solutions obtained for different branches of the decision tree. Project portfolio selection problems, too, involve comparable dependencies, because the consumption of shared resources, for example, implies that the optimal strategy for a given project cannot be determined without considering strategies for the other projects (Gustafsson & Salo, 2005).

In this paper, we develop the *Decision Programming* framework which uses the graphical representation of influence diagrams to capture the salient properties of multi-stage decision problems under uncertainty. The inputs of this framework consist of (i) the problem structure, represented by a connected, acyclic directed graph consisting of decision, chance and value *nodes* as well as informational and probabilistic dependencies between these, shown through *arcs*; (ii) discrete sets of states that represent the set of possible decisions at each decision node and the possible realisation of chance events at chance nodes; (iii) numerical parameters, such as probabilities at chance nodes and consequences (or their utilities) at value nodes. The inputs also include (iv) constraints, such as logical dependencies between decisions, bounds on resource consumption and requirements associated with risk preferences (eg, chance and VaR constraints). These constraints, too, typically involve numerical parameters.

Out of the five types of requirements that can be imposed on influence diagrams (see, e.g., Section 1.2 in Zhang et al., 1994), we do not require *regularity* (i.e., there exists a single path traversing all decision nodes), *no-forgetting* (i.e., all information which is known when making earlier decisions will be available when making later decisions) or *existence of a single value node*. Furthermore, the constraint of not allowing value nodes to have children (i.e., *no-children-to-value-node*) is not restrictive as it can be circumvented by restructuring the influence diagram (i.e., by converting any such value node into a chance node whose state is non-randomly dependent on the nodes from which there are arcs to this value node and by appending the equivalent new value node to the end of the diagram). Thus, we only retain the constraint of *acyclity* which, as an assumption, can be justified on the grounds that decisions and chance events are all typically associated with specific points in time.

Within this generic set-up, we allow for both deterministic (e.g., logical dependencies, costs arising at one or more nodes) and chance (i.e., probabilistic) constraints. The resulting joint problem representation (influence diagram plus constraints) is then converted into an equivalent mixed-integer linear programming (MILP) problem that is generic enough for solving also LImited Memory Influence Diagrams (LIMIDs) in which the 'no-forgetting' assumption does not hold. Furthermore, we provide an algorithm for computing all non-dominated strategies in problems that have multiple objectives associated with corresponding multiple value nodes. All these modeling features are cast into corresponding MILP problems that can be solved with available software tools (for a survey, see, e.g., Fourer, 2017).

As visual tools for problem representation, influence diagrams differ from decision trees since they do not communicate in what ways the problem structure may be asymmetric so that the sets of possible states at some decision and chance nodes may be constrained by the states at other nodes. Nevertheless, asymmetric problems can be modeled with influence diagrams by defining node states and their dependencies appropriately (see, e.g., Smith, Holtzman, & Matheson, 1993). Mathematically, the mapping of input parameters (i.e., probabilities and decisions) to outputs (i.e., expected utilities) in influence diagrams can be viewed as a piecewise multilinear function (Borgonovo & Tonoli, 2014), which underpins the developments in this paper. For an account of the evolution of influence diagrams, see Bielza et al. (2011).

Our contribution is relevant to stochastic programming (Birge & Louveaux, 2011) in outlining a general framework for problems in which decisions are made over several stages and realizations of uncertain events are observed between pairs of successive stages. In the first stage, an initial decision is selected, and subsequent recourse decisions are made after observing the realizations of uncertain earlier events. We distinguish between endogenous and exogenous uncertainties based on whether the probability distributions associated with chance events are impacted by decisions. In Decision Programming, both types of uncertainties are accommodated by converting influence diagrams and adjoining constraints into multi-stage stochastic integer programming (MSSIP) problems that can be solved using off-the-shelf MILP solvers. That is, the diagram is first converted into a sequence of decision and chance nodes. This sequence is then employed to build the deterministic equivalent MILP formulation of the MSSIP. More generally, the field of stochastic optimization spans a wide range problem types and solution approaches. Within the framework influence diagrams, we address three of the research challenges identified by Powell (2019) in the recent literature review, i.e., (i) consideration of risk measures, (ii) treatment of multiple objectives, and (iii) modelling of multiple agents.

This paper is structured as follows. Section 2 discusses earlier approaches. Section 3 develops the Decision Programming framework, and Section 4 presents illustrative examples. Section 5 develops approaches for dealing with risk preferences, chance constraints and multiple objectives. Section 6 presents results on computational performance and outlines directions for further research. Section 7 concludes.

2. Earlier approaches

Influence diagrams were initially developed in the1970ʳs (Howard & Matheson, 1984; Howard & Matheson, 2005; Howard & Matheson, 2006; Howard, Matheson, Merkhofer, Miller, & North, 2006; Olmsted, 1983) to represent informational and probabilistic dependencies between decisions and uncertain chance events whose realizations govern what the consequences for the DM will be. If the regularity, no-forgetting and single-value node assumptions hold and the aim is to maximize expected utility at the value node, these diagrams can be solved with well-established techniques, notably by forming the equivalent decision tree which can be solved through dynamic programming (Tatman & Shachter, 1990); or by removing decision and chance nodes from the diagram one-by-one, possibly after arc reversals (see, e.g., Howard & Matheson, 2005; Koller & Friedman, 2009; Shachter, 1986; Smith et al., 1993).

Problems in which the 'no-forgetting' assumption does not hold give rise to LIMIDs. It is well-known (see, e.g., Mauá & Cozman, 2016; Zhang et al., 1994) that LIMIDs are computationally chal-

lenging because optimal strategies cannot be determined through a straightforward series of local computations. They have been solved primarily in view of maximizing the expected utility (MEU) at a single value node, in the absence of constraints which would place restrictions on decision nodes in different parts of the influence diagram. For this problem context, Lauritzen & Nilsson (2001) develop an iterative Single Policy Updating (SPU) approach for LIMIDs by solving a series of expected utility maximization problems by message passing in a junction tree derived from the influence diagram. This approach is guaranteed to give the optimum in *soluble* problems which, in non-technical terms, means that there exists a sequence of decision nodes such that the optimum can be computed by solving a series of local optimization problems. Parmentier, Cohen, Leclère, Obozinski, & Salmon (2020) construct rooted junction trees and provide an MILP-based formulation which solves soluble MEU problems to optimality, but which is slower than the SPU algorithm. For problems which are not soluble, their formulation provides bounds which are better than those by applying the SPU algorithm to the soluble relaxation of the initial problem.

Yuan, Wu, & Hansen (2010) propose a branch-and-bound algorithm which presumes that the influence diagram satisfies the regularity condition. Koller & Milch (2003) consider multi-agent problems and provide algorithms for computing Nash equilibria based on MEU maximization under the assumption that agents have perfect recall. Mauá & Cozman (2016) study the computational performance of k-neighborhood local search algorithms and propose approximate algorithms.

MEU problems have also been tackled also in the context of credal networks which can be derived from influence diagrams by replacing each decision node by a corresponding chance node with incompletely specified probabilities such that these probabilities add up to one. Then, the computation of optimal MEU strategies can be viewed as a problem of probabilistic inference in which the aim is to determine for which combination of these incompletely specified probabilities the expected utility is highest (for an overview, see Mauá & Cozman, 2020). Linear constraints have been employed in this setting by de Campos & Ji (2008) who employ McCormick constraints (McCormick, 1976) which, however, are not computationally very efficient. Antonucci, de Campos, Huber, & Zaffalon (2013, 2015) present an approach that is based on conditioning the joint probability distribution on iteratively selected nodes and by solving ensuing linear optimization problems for these. However, they provide no formal proofs of convergence.

A limitation of the above approaches is that they are not capable of explicitly addressing multiple objectives which correspond to different value nodes (see, e.g., Diehl & Haimes, 2004). Furthermore, while approaches based on local computations and iterative message passing schemes are likely more efficient under a computational standpoint, they are not applicable to problems with constraints that pertain to several chance, decision and value nodes in different parts of the influence diagram (e.g., due to logical interdependencies, limited budgets, bounds on risk levels) and whose fulfilment cannot therefore be determined locally. For example, the DM may seek to maximize the expected net present value (NPV) subject to the requirement that the expectation in the lower tail of the NPV distribution is not too low (i.e., Conditional Value-at-Risk, which is a coherent risk measure; Artzner, Delbaen, Eber, & Heath, 1999).

In portfolio decision analysis (Liesiö, Salo, Keisler, & Morton, 2021; Salo, Keisler, & Morton, 2011), influence diagrams help portray the overall structure of probabilistic and informational dependencies, but they cannot handle constraints arising from limited budgets or logical dependencies between alternatives. For project selection problems, *Contingent Portfolio Programming* (Gustafsson & Salo, 2005) employs MILP to determine optimal project management strategies when the projects' cash flows are contingent on scenarios whose probabilities cannot depend endogenously on project decisions. Vilkkumaa, Liesiö, & Salo (2018) extend this approach to single-stage selection problems in which scenario probabilities can depend endogenously on project decisions. Liesiö & Salo (2012) derive decision recommendations for single-stage project selection problems with one objective and possibly incomplete utility and probability information. Yet, none of these approaches is equipped to handle problems in which there is a *combination* of endogenous uncertainties, several decision stages, and multiple objectives.

Stochastic programming is widely employed as one of the underpinning frameworks for multi-stage decision problems under uncertainty. Nevertheless, the literature on endogenous uncertainties in stochastic programming is still sparse, because these uncertainties give rise to models that cannot be readily solved with existing solution techniques, most prominently convex programming in general, and linear programming in particular.

Specifically, most of the related stochastic programming literature focuses on problems in which decisions can influence the information structure, in particular the timing of unveiling uncertainties, but not the probability distributions associated with uncertain events. Goel & Grossmann (2006) develop a stochastic programming formulation for multi-stage problems for the timing of oil well exploitation, which decision is assumed not to influence the uncertain amount of recoverable oil. Building on Goel & Grossmann (2004), they propose a unified framework and solution methods for problems in which the decisions influence the time of observing uncertainties. Gupta & Grossmann (2011, 2014) present specialized solution methods for oil and gas field development. Colvin & Maravelias (2008) propose a stochastic programming model for novel product development in pharmaceutical research, further extended by Colvin & Maravelias (2009). In this context, the timing of the resolution of uncertainties is influenced endogenously by the decisions on how to perform clinical trials which, however, leads to computational challenges (Colvin & Maravelias, 2010). Solak, Clarke, Johnson, & Barnes (2010) optimize R&D project portfolios under endogenous uncertainty, acknowledging that the inclusion of decision dependent uncertainties significantly degrades tractability. To tackle this issue, they propose a sophisticated solution method, exploiting the formulation devised specifically for the problem. Apap & Grossmann (2017) provide a comprehensive recent literature overview and propose an approach for problems with a decision-dependent information structure.

Problems where decisions can (also) affect the probability distributions of uncertain events have been much less explored in stochastic programming. The predominant strategy has been to remove decision dependent probabilities using appropriate transformations in the probability measure, as described by Rubinstein & Shapiro (1993) (see also Pflug, 2012), or in the probability distribution itself (cf. Dupačová, 2006). In their overview of this scarce literature, Hellemo, Barton, & Tomasgard (2018) propose a taxonomy of distinct classes for stochastic programs with endogenous uncertainties and possible formulation approaches. They also report computational experiments to highlight how challenging these problems are for state-of-the-art optimization solvers.

In fact, multi-stage optimization problems under uncertainty can involve decision dependent probabilities, parameters, and/or information structures (Hellemo et al., 2018). The *Decision Programming* framework seeks to encompass all these variants, on condition that each chance event has a finite number of possible realizations and decisions correspond to choices from a finite set of discrete alternatives.

3. Methodological development

3.1. Influence diagram representation of the decision problem

Multi-stage decision problems under uncertainty can be modeled as connected acyclic networks $G = (N, A)$ whose nodes $N = C \cup D \cup V$ consist of chance nodes C, decision nodes D, and value nodes V. Chance nodes C represent uncertain events associated with random variables; decision nodes D correspond to decisions among discrete alternatives; and value nodes V represent consequences that result from the realizations of random variables at chance nodes and the decisions made at decision nodes.

Dependencies between nodes are represented by arcs $A = \{(i, j) \mid i, j \in N\}$. A directed *path* of length k is a sequence of nodes $(i_1, i_2, \ldots, i_k)$ such that $(i_l, i_{l+1}) \in A$ for all $l = 1, \ldots, k-1$. The *information set* of a node $j \in N$, defined as $I(j) = \{i \in N \mid (i, j) \in A\}$, consists of the direct predecessors (also referred to as parents) of j from which there is an arc to j. Because the network G is acyclic, the nodes N can be indexed consecutively with integers $1, 2, \ldots, |N|$ (where $|\cdot|$ denotes the number of elements in a set) so that for each node $j \in N$, the indices of the nodes in its information set $I(j)$ are smaller than j (i.e., $i < j$ for all $i \in I(j)$).

We denote the number of chance nodes by $n_C = |C|$ and the number of decision nodes by $n_D = |D|$. These $n = n_C + n_D$ chance and decision nodes are indexed as $C \cup D = \{1, 2, \ldots, n\}$, while the $n_V = |V| = |N| - n$ value nodes are indexed as $n+1, \ldots, n+n_V$. For now, we assume that there is a single value node in the influence diagram (the extension to multiple value nodes is covered in Section 5.4). Consequences at this value node are determined by the decisions and the realization of chance events. There are no arcs from the value node to chance and decision nodes, as these are not affected by the consequences. All chance events and decisions are relevant in the sense that there is a directed path from every chance and decision node to the value node. There is a path from every chance and decision node to the value node.

Each chance and decision node $j \in C \cup D$ has a finite set S_j of discrete states. The occurrence of states depend on their possible *information states* $s_{I(j)} \in S_{I(j)}$, defined as all possible combinations of states $S_{I(j)} \subseteq \prod_{i \in I(j)} S_i$ for the nodes in the information set $I(j)$. For each chance node $j \in C$, these states correspond to realizations of the random variable X_j, which depends probabilistically on the states s_i of the nodes $i \in I(j)$ in the information set of j. For a decision node $j \in D$, each state $s_j \in S_j$ corresponds to a decision that is made based on the information state $s_{I(j)}$. For brevity, we use X_j, $j = 1, \ldots, n$, to denote both random variables which are associated with chance nodes $j \in C$ and decision variables which are associated with decision nodes $j \in D$.

If $j \in C$ is a chance node whose information state is $s_{I(j)}$, then state $s_j \in S_j$ occurs with the conditional probability

$$\mathbb{P}(X_j = s_j \,|\, X_{I(j)} = s_{I(j)}), \qquad \forall\, j \in C,\ s_j \in S_j,\ s_{I(j)} \in S_{I(j)}, \tag{1}$$

where $X_{I(j)} = s_{I(j)}$ means that the variables X_i in the information set $i \in I(j)$ have same values that are assigned to them by the information state $s_{I(j)}$. For each decision node $j \in D$, a *local (decision) strategy* $Z_j : S_{I(j)} \mapsto S_j$ is a function that maps each information state in $S_{I(j)}$ to a decision in S_j. A *(global decision) strategy* Z is a collection of local decision strategies which specifies one local strategy Z_j for each decision node $j \in D$. The set of all decision strategies is denoted by $\mathbb{Z}$.

Fig. 1 illustrates the notation in the context of a simple oil wildcatter example in which a test may (but does not have to) be carried out to obtain information about the prospective oil well before the drilling decision (see, e.g., Yet, Neil, Fenton, Constantinou, & Dementiev, 2018). The diagram has two chance nodes $C = \{2, 3\}$, two decision nodes $D = \{1, 4\}$ and a single value node $V = \{5\}$. Thus, $n_C = n_D = 2$ and $n = n_C + n_D = 4$. The information sets $I(1) = I(2) = \emptyset, I(3) = \{1, 2\}, I(4) = \{1, 3\}, I(5) = \{1, 2, 4\}$. The decision at node D_1 node represents whether or not the test is carried out, modelled through the states $S_1 = \{test, \neg test\}$ (where $\neg$ stands for "do not"). At chance node 3, the test result $S_3 = \{wet, dry, none\}$ depends on the state of the oil well $s_2 \in S_2 = \{wet, dry\}$ and the testing decision made at node 1. If $s_1 = no$, then the result s_3 is *none*; otherwise the test result will be either *wet* or *dry*, depending on the state of the oil well and the properties of the test. For example, $\mathbb{P}(X_3 = dry \,|\, X_2 = wet)$ is the probability of getting a false negative result from the test when the oil well is wet. A local strategy Z_1 at node 1 specifies whether or not the test will be carried out. At node 4, the local strategy Z_4 maps all its information states $S_{I(4)}$, defined as combinations $(s_1, s_3) \in \{(yes, wet), (yes, dry), (no, none)\}$ of testing decisions s_1 and corresponding test results s_3, to a drilling decision $s_4 \in S_4 = \{drill, \neg drill\}$. Finally, the final net present value at the value node 5 depends on the costs arising from the testing and drilling decisions as well as the state of the oil well. Note that in Fig. 1, all arcs lead from a node with a lower index to a node with a higher one. This would be the situation also if the two first nodes were to be listed in the reverse order, starting with the state of the oil well, followed by the testing decision.

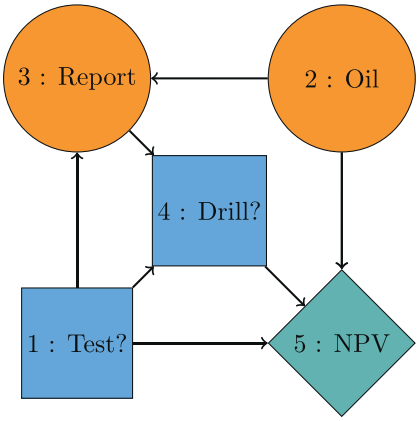

Fig. 1. An influence diagram for the oil wildcatter example.

3.2. Paths

A *path* $s = (s_1, s_2, \ldots, s_n)$ of length n is a sequence of states $s_i \in S_i$ of all chance and decision nodes, i.e., $i \in C \cup D$ for all $i = 1, \ldots, n$. The set S of all paths of length n is

$$S = S_{1:n} = \{(s_1, s_2, \ldots, s_n) \,|\, s_i \in S_i,\ \ i = 1, \ldots, n\}. \tag{2}$$

Paths of length $k < n$ are sequences $s_{1:k} = (s_1, s_2, \ldots, s_k)$ such that $s_i \in S_i,\ \ i \le k$. If $s_{1:k} \in S_{1:k}, k < n$, and $s_{k+1} \in S_{k+1}$, the state s_{k+1} can be appended to $s_{1:k}$ to form the path $s_{1:k+1} = (s_1, s_2, \ldots, s_k, s_{k+1}) \in S_{1:k+1}$. If $s_{1:k} \in S_{1:k},\ \ k \le n$, and $I \subsetneq \{1, \ldots, k\}$, then s_I is a subsequence of $s_{1:k}$ for the nodes $i \in I$. Thus, s_I is a sequence of length $|I|$ which contains the same states as $s_{1:k}$ for nodes $i \in I$.

Thus, at each decision node $j \in D$, $Z_j \in Z$ maps the information state $s_{I(j)}$ contained in s to the corresponding decision s_j in s. The strategy $Z \in \mathbb{Z}$ is *compatible* with the path $s \in S$ if and only if $Z_j(s_{I(j)}) = s_j, \forall\, Z_j \in Z,\ j \in D$. Conversely, the set of active paths for the strategy Z is $S_Z = \{s \in S | Z_j(s_{I(j)}) = s_j, \forall Z_j \in Z,\ j \in D\}$.

Referring to Fig. 1, the paths S correspond to sequences (s_1, s_2, s_3, s_4) where the states s_i, $i = 1, \ldots, 4$, indicate decisions and realizations of chance events. For example, $s = (s_1, s_2, s_3, s_4) = (test, dry, wet, drill)$ is the path in which the test is carried out $(s_1 = test)$, the well is dry $(s_2 = dry)$, the test result is positive $(s_3 = wet)$ and the well is drilled $(s_4 = drill)$. The strategy defined by first testing and then drilling if and only if the test result is

positive is compatible with this path. Strategies in which no test is carried out, regardless of the drilling decision, are not compatible with this path.

For any strategy $Z \in \mathbb{Z}$, the probability of a path $s \in S$ can be expressed recursively as a function of the conditional probabilities (1) and local decision strategies so that

$$\mathbb{P}(s_{1:k} \,|\, Z) = \Big(\prod_{\substack{i \in C \\ i \le k}} \mathbb{P}\big(X_i = s_i \,|\, X_{I(i)} = s_{I(i)}\big) \Big) \Big(\prod_{\substack{j \in D \\ j \le k}} \mathbb{I}\big(Z_j(s_{I(j)}) = s_j\big) \Big), \tag{3}$$

where the indicator function $\mathbb{I}(\cdot)$ is defined so that

$$\mathbb{I}(Z_j(s_{I(j)}) = s_j) = \begin{cases} 1, & \text{if } Z_j(s_{I(j)}) = s_j, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Note that if the strategy Z is compatible with s, then $\mathbb{I}(Z_j(s_{I(j)}) = s_j) = 1$ and thus $\mathbb{P}(s \,|\, Z) = \prod_{i \in C} \mathbb{P}(X_i = s_i \,|\, X_{I(i)} = s_{I(i)})$. Conversely, if Z is not compatible with s, it contains some local strategy Z_j, $j \in D$ such that the information state $s_{I(j)}$ contained in s is mapped to a decision that is not the same as the state s_j for node j along the given path s. Thus, choosing Z means that s cannot occur and therefore $\mathbb{P}(s \,|\, Z) = 0$. Moreover, $\mathbb{P}(s|Z) = 0$ for any $s \notin S_Z$.

3.3. Characterizing path probabilities using linear inequalities

A given strategy $Z \in \mathbb{Z}$ assigns probabilities to all paths $s_{1:k} \in S_{1:k}$, $k = 1, \ldots, n$, in accordance with (3). In principle, one could introduce binary variables for the indicator functions $\mathbb{I}\big(Z_j(s_{I(j)}) = s_j\big)$, for all $j \in D$, whose multiplication would lead to a mixed-integer nonlinear programming (MINLP) problem which could be converted into a equivalent MILP. An early version of the Decision Programming approach relied on this strategy, which, although feasible, led to an MILP formulation with a weak linear programming relaxation that was too inefficient for off-the-shelf solver performance.

Alternatively, the path probabilities $s_{1:k} \in S_{1:k}$, $k = 1, \ldots, n$, can be characterized through sets of linear inequalities. Towards this end, local decision strategies Z_j, $j \in D$, are modelled through corresponding binary variables $z(s_j \,|\, s_{I(j)}) \in \{0, 1\}$ such that $z(s_j \,|\, s_{I(j)}) = 1$ if and only if Z_j maps the information state $s_{I(j)}$ to the decision $s_j \in S_j$, i.e.,

$$Z_j(s_{I(j)}) = s_j \iff z(s_j \,|\, s_{I(j)}) = 1, \qquad \forall\, j \in D,\ s_j \in S_j,\ s_{I(j)} \in S_{I(j)}. \tag{5}$$

The mutual exclusivity of the decisions is ensured through the constraints

$$\sum_{s_j \in S_j} z(s_j \,|\, s_{I(j)}) = 1, \qquad \forall\, j \in D,\ s_{I(j)} \in S_{I(j)}, \tag{6}$$

which ensure that exactly one decision $s_j \in S_j$ is chosen for every information state $s_{I(j)} \in S_{I(j)}$.

For the given strategy $Z \in \mathbb{Z}$, the corresponding probability $\pi(s)$ of any path $s \in S$ can be derived recursively as follows. To initialize the recursion, let $\pi_0(s) = 1$. Suppose that the probabilities $\pi_i(s) = \mathbb{P}(X_{1:k-1} = s_{1:k-1} \,|\, Z)$ are known for nodes $i \le k-1$ and consider the next node $k \le n$. If $k \in C$ is a chance node, let

$$\pi_k(s) = \mathbb{P}\big(X_k = s_k \,|\, X_{I(k)} = s_{I(k)}\big) \pi_{k-1}(s), \tag{7}$$

where the first term on the right side of (7) is given by (1). If $k \in D$ is a decision node, let

$$\pi_k(s) = \begin{cases} \pi_{k-1}(s), & \text{if } z(s_k \,|\, s_{I(k)}) = 1 \\ 0, & \text{if } z(s_k \,|\, s_{I(k)}) = 0. \end{cases} \tag{8}$$

This assignment corresponds to the inequalities

$$\max\{0,\ \pi_{k-1}(s) + z(s_k \,|\, s_{I(k)}) - 1\} \le \pi_k(s) \le \min\{\pi_{k-1}(s),\ z(s_k \,|\, s_{I(k)})\},$$

which are equivalent to

$$\pi_k(s) \le \pi_{k-1}(s) \tag{9}$$

$$\pi_k(s) \le z(s_k \,|\, s_{I(k)}) \tag{10}$$

$$\pi_k(s) \ge 0 \tag{11}$$

$$\pi_k(s) \ge \pi_{k-1}(s) + z(s_k \,|\, s_{I(k)}) - 1. \tag{12}$$

Theorem 1 states that the path probabilities implied by strategy Z can be calculated through the assignment (5)–(8). Importantly, the equivalence between the assignments (5)–(8) and the inequalities (9)–(12) implies that the path probabilities implied by decision strategies can be determined by employing these inequalities as constraints on the variables $z(s_k|s_{I(k)})$, $k \in D$, $s_k \in S_k$, $s_{I(k)} \in S_{I(k)}$.

Theorem 1. *Let $Z \in \mathbb{Z}$ be a decision strategy and choose a path $s \in S$. If $\pi_k(s)$, $k = 1, \ldots, n$, and $z(s_j \,|\, s_{I(j)})$, $\forall j \in D$, satisfy the constraints (5)–(8), then*

$$\pi_k(s) = \mathbb{P}(X_{1:k} = s_{1:k} \,|\, Z), \qquad \forall\, k = 1, \ldots, n. \tag{13}$$

In particular, $\pi(s) \overset{def}{=} \pi_n(s)$ is the probability of the path s for the strategy Z.

Proof. See Appendix A.

3.4. Maximization of expected utility

We assume that at the value node $v \in V$, the function $Y_v : S_{I(v)} \mapsto \mathbb{C}$ maps combinations of states of the nodes in its information set $I(v)$ to the set of consequences $\mathbb{C}$ and that there exists a real-valued utility function $U : \mathbb{C} \mapsto \mathbb{R}$ that is defined over $\mathbb{C}$. Then, the utility associated with the path $s \in S$ can be precomputed as

$$\mathcal{U}(s) = U[Y_v(s_{I(v)})]. \tag{14}$$

Because the path probabilities $\pi(s)$, $s \in S$, for the selected strategy $Z \in \mathbb{Z}$ are given by Theorem 1, it follows that the strategy which maximizes the DM's expected utility is the solution to the optimization problem in Corollary 1.

Corollary 1. *The expected utility is maximized by the strategy $Z \in \mathbb{Z}$ which solves the optimization problem*

$$\underset{Z \in \mathbb{Z}}{\text{maximize}} \sum_{s \in S} \pi(s) \mathcal{U}(s) \tag{15}$$

subject to constraints (5)–(7) and (9)–(12) on decision variables $z(s_k|s_{I(k)}) \in \{0, 1\}$, $\forall\ k \in D$, $s_k \in S_k$, $s_{I(k)} \in S_{I(k)}$ and path probabilities $\pi_k(s) \in [0, 1]$, $\forall s \in S$.

In particular, the objective function and constraints in Corollary 1 are linear in the decision variables $z(s_j \,|\, s_{I(j)})$ and the corresponding path probabilities $\pi_k(s)$. This is an MILP problem for which the optimal strategy can be computed with off-the-shelf MILP solvers.

3.5. An improved MILP formulation

To enhance the formulation in Corollary 1, we note that the objective function (15) has path probabilities $\pi(s)$ only for full paths $s \in S = S_{1:n}$ of length n. Also, the probability $\pi(s)$ of each path $s \in S$ depends on two separable components. First, for each path $s \in S$, the conditional probabilities (1) of the states s_j for chance nodes

$j \in C$ can be multiplied to obtain the following upper bound for $\pi(s)$:

$$p(s) = \prod_{j \in C} \mathbb{P}(X_j = s_j \mid X_{I(j)} = s_{I(j)}). \tag{16}$$

Second, for a given strategy $Z \in \mathbb{Z}$, this upper bound $p(s)$ is the actual probability of s if and only if Z is compatible with s. That is, if $z(s_j \mid s_{I(j)}) = 1$, $\forall j \in D$, the inequalities (9)–(12) imply $\pi_j(s) = \pi_{j-1}(s)$ for each $j \in D$. This result can be used to solve the equations (7)–(8) recursively starting from $\pi_0(s) = 1$ to the last node n for which $\pi_n(s) = p(s)$ in (16). Conversely, if the strategy Z is not compatible with s, inequalities (9)–(10) imply that $\pi_n(s) \leq \pi_j(s) = 0$ for some $j \in D$. Thus, because $\pi(s) = \pi_n(s) = p(s)$ if and only if $z(s_j \mid s_{I(j)}) = 1 \forall j \in D$, the optimization problem in Corollary 1 can be reformulated as

$$\underset{Z \in \mathbb{Z}}{\text{maximize}} \quad \sum_{s \in S} \pi(s)\, \mathcal{U}(s) \tag{17}$$

$$\text{subject to} \quad \sum_{s_j \in S_j} z(s_j \mid s_{I(j)}) = 1, \qquad \forall j \in D,\ s_{I(j)} \in S_{I(j)} \tag{18}$$

$$0 \leq \pi(s) \leq p(s), \qquad \forall s \in S \tag{19}$$

$$\pi(s) \leq z(s_j \mid s_{I(j)}), \qquad \forall s \in S,\ j \in D \tag{20}$$

$$\pi(s) \geq p(s) + \sum_{j \in D} z(s_j \mid s_{I(j)}) - |D|, \quad \forall s \in S \tag{21}$$

$$z(s_j \mid s_{I(j)}) \in \{0, 1\}, \qquad \forall j \in D,\ s_j \in S_j,\ s_{I(j)} \in S_{I(j)}, \tag{22}$$

where the constraints (18) ensure that some decision $s_j \in S_j$ is made at each decision node $j \in D$ for every information state set $s_{I(j)} \in S_{I(j)}$ (as stated in (6)). Constraints (19) bound the probabilities of paths $s \in S$. Constraints (20) ensure that only those paths which are compatible with the strategy can have positive probabilities. Constraints (21) ensure that the probabilities of paths with negative utility $\mathcal{U}(s)$ cannot become smaller than their upper bounds $p(s)$ for paths s such that $z(s_j \mid s_{I(j)}) = 1$, $j \in D$. Constraints (22) enforce the domain of all binary variables $z(s_j \mid s_{I(j)})$. For clarity, we note that in constraints (20)–(21), the states s_j and $s_{I(j)}$ are taken from the selected path $s \in S$. The terms $\mathcal{U}(s)$ and $p(s)$ in (17) and (19), respectively, can be calculated from (16) and (14) before solving the model (17)–(22).

Because utility functions over consequences are unique to positive affine transformations and the value node has a finite number of $\prod_{i \in I(v)} |S_i|$ information states, one can normalize the utility function to an interval of non-negative utilities $\mathcal{U}^N(s) \in [0, 1]$ through the assignment $\mathcal{U}^N(s) = [\mathcal{U}(s) - \underline{\mathcal{U}}]/[\overline{\mathcal{U}} - \underline{\mathcal{U}}]$, where $\mathcal{U}(s)$ is the utility function to be normalized and $\overline{\mathcal{U}} = \max_{s \in S}\{\mathcal{U}(s)\} > \underline{\mathcal{U}} = \min_{s \in S}\{\mathcal{U}(s)\}$. In this case, the constraints (21) can be omitted, because the path probabilities will be naturally steered to their upper bounds.

In the oil wildcatter example of Fig. 1, the constraints (18) are given by $\sum_{s_1 \in \{test, \neg test\}} z(s_1) = 1$ and $\sum_{s_4 \in \{drill, \neg drill\}} z(s_4 \mid s_3) = 1$ where $s_3 \in \{wet, dry, none\}$ specifies on what information the drilling decision is made. For the path $s = (s_1, s_2, s_3, s_4) = (test, dry, wet, drill)$, constraint (20) bounds the path probability $\pi(s)$ from above $p(s) = \mathbb{P}(X_2 = dry)\mathbb{P}(X_3 = wet \mid X_2 = dry)$ where $\mathbb{P}(X_2 = dry)$ is the probability that the well is dry and $\mathbb{P}(X_3 = wet \mid X_2 = dry)$ is the probability of getting a report which says that the well is wet when it is actually dry. Finally, $\mathcal{U}(s)$ gives the utility in the situation where the dry well is drilled after the test.

3.6. Computational complexity

The computational complexity of the formulation (17)–(22) depends predominantly on the number of chance and decision nodes $i \in C \cup D$, as well as the number of their states S_i and information states $S_{I(i)}$. Here, we assume that the problem is symmetric, so that at each chance and decision node all states are possible for every one of their information states, noting that in asymmetric influence diagrams some states are impossible for some information states so that some paths can be eliminated. For example, if the decision at node 1 in Fig. 1 is not to test, then the test result s_3 will be *none* and the results *wet* and *dry* are impossible as $\mathbb{P}(s_3 = wet \mid s_1 = \neg test) = \mathbb{P}(s_3 = dry \mid s_1 = \neg test) = 0$. Thus, paths containing the states $s_1 = \neg test$ and $s_3 \in \{wet, dry\}$ can be eliminated from consideration.

In symmetric problems, the number of continuous path variables $\pi(s)$ is $\prod_{i \in C \cup D} |S_i|$. Thus, if there are at least two states at each node $i \in C \cup D$, the number of paths s grows exponentially with respect to the number of decision and chance nodes as the lower bound for the number of paths is $o(\prod_{i \in C \cup D} |S_i|) = o(2^{|C|+|D|})$. The number of binary decision variables $z(s_j \mid s_{I(j)})$, $j \in D, s_j \in S_j, s_{I(j)} \in S_{I(j)}$, depends on the number of decision nodes D as well as their states S_i and information states $S_{I(j)}$. Specifically, there are $\sum_{j \in D} |S_{\{j\} \cup I(j)}| = \sum_{j \in D} |S_j| \prod_{i \in I(j)} |S_i|$ binary decision variables. Thus, the number of these variables is at most $\mathcal{O}\big(\sum_{j \in D} |S_{\{j\} \cup I(j)}|\big)$. While the number of binary variables is not typically large, they make the problem becomes much harder to solve. The number of constraints in (18) is $\sum_{j \in D} |S_j|$, while that of constraints (19) is the same that of paths. In constraint (20), the path probability $\pi(s)$ is constrained by each decision along this path s.

3.7. Valid constraints

The formulation (17)–(22) can be solved more efficiently by introducing valid constraints derived from the problem structure and help compute the optimal decision strategies, as shown in Section 6. However, adding these constraints directly may slow down the overall solution process, especially in larger problems in which many of them can be derived from the problem structure.

Alternatively, one can include these valid constraints during the solution process as "lazy constraints" that can be used by the MILP solver to prune nodes of the branch-and-bound tree more efficiently. One can also add them during the solution process in a cutting plane fashion as "user cuts" for a subset of nodes in the tree based on some criterion (or multiple criteria), for example, if the upper bound has not improved enough within some time interval. Such lazy constraints and user cuts are standard features in off-the-shelf MILP solvers.

Specifically, the first set of equalities, referred to as *probability cuts*, exploit the fact that for any strategy $Z \in \mathbb{Z}$, the sum of the probabilities $\pi(s)$ must equal one so that $\sum_{s \in S} \pi(s) = 1$. These equalities are valid for any problem that can be formulated as (17)–(22). As an example of a probability cut that works as a lazy constraint, suppose that the optimal (fractional) solution of a node in the branch-and-bound tree does not satisfy the probability cut. Then, the problem at that node will be re-optimized after adding the probability cut, and if the new optimal cost is smaller than the current best primal bound, the node can be discarded. In our computational analyses, we have used probability cuts as lazy constraints that were not initially included in the MILP formulation, but checked against violation in the branch-and-cut procedure. This approach prevents the introduction of a large number of constraints that are unlikely to be violated. It is standard practice in the use of professional-grade solvers such as Gurobi and

CPLEX, and can be done in straightforward manner through their application protocol interfaces.

The second set of equalities can be used in problems whose structure makes it possible to determine in advance for a given strategy $Z \in \mathbb{Z}$ how many active paths $s \in S_Z$ have a positive probability $\pi(s) > 0$. For example, if the number of such active paths is n_s is for all feasible strategies, we can define a valid equality $\sum_{s \in S} \pi(s)/p(s) = n_s$ where $p(s)$ in (16) is the upper bound for $\pi(s)$. This approach can be generalized to asymmetric problems in which the number of active paths varies for different decision strategies. In such cases, several equalities can to be added to cover different possibilities in how the number of active paths depends on the states of decision or chance nodes. Such information, derived from an analysis of symmetries in the problem structure (see, e.g., Bielza et al., 2011), serve to improve computational efficiency.

4. Decision modeling examples

4.1. Decision programming without the no-forgetting assumption

As an example of a problem in which the no-forgetting assumption does not hold, assume that there is an uncertain load L on a built structure which can be fortified through actions A_1 and A_2 to mitigate the risk of a structure failure F. These two decisions are informed by measurement reports R_1 and R_2 on the load L. The decision as to whether action A_1 should be implemented is informed by the report R_1 only and, similarly, decision A_2 is based on the report R_2 alone. In particular, the decision as to whether the fortification decision A_1 will be or has been installed is not known when making the decision A_2 (and conversely for A_2). The utility at the target node T depends on whether or not the structure fails and how much the fortification actions cost.

This problem structure also represents a situation where the reports are generated by sensors which inform safety controls (e.g., valves) that must activated instantaneously to prevent potential disruptions in a safety-critical system such as a nuclear plant (see, e.g., Mancuso et al., 2019). In particular, the safety must be ensured even if failures of communication equipment prevent the sensors from sharing information with a centralised server or other sensors.

Just as in the example in Figure 12 in Zhang et al. (1994), this problem structure is challenging in that the optimal strategies at the decision nodes are inderdependent and cannot be solved based on decomposition (see Chapter 8 in Zhang (1994) for a proof). In particular, the regularity and 'no-forgetting' assumptions do not hold, because the temporal order of the decision is not predetermined and there is no sequence of chance nodes $C = \{L, R_1, R_2, F\}$ and decision nodes $D = \{A_1, A_2\}$ such that for all decision nodes, the states of all preceding nodes would be known at the time of decision making. Fig. 2 presents an influence diagram representing this setting.

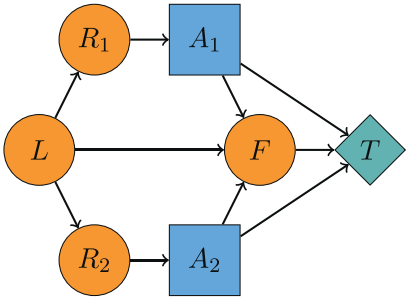

Fig. 2. Influence diagram of the double monitoring example.

Still, this problem can be solved using *Decision Programming*. The sequence $(L, R_1, R_2, A_1, A_2, F, T)$ captures the dependence structure: $I(R_i) = \{L\}$, $i = 1, 2$ (the reports depend on the load); $I(A_i) = \{R_i\}$, $i = 1, 2$ (decisions about the fortification actions are informed by respective reports); $I(F) = \{L, A_1, A_2\}$ (failure depends on the load and fortification decisions, but not on the reports); and $I(T) = \{A_1, A_2, F\}$ (the final outcome depends on the failure and the cost of implementing the fortification actions). By using node labels to indicate sets of states for corresponding nodes, the paths are sequences $s = (l, r_1, r_2, a_1, a_2, f) \in L \times R_1 \times R_2 \times A_1 \times A_2 \times F = S$. The probabilities $p(s)$ in (16) are $p(l, r_1, r_2, a_1, a_2, f) = \mathbb{P}(l)\mathbb{P}(r_1 \mid l)\mathbb{P}(r_2 \mid l)\mathbb{P}(f \mid l, a_1, a_2)$, and the decision strategies are defined by $Z = (Z_1, Z_2)$ such that $Z_i : R_i \mapsto A_i$.

Using this notation, the optimal fortification strategy can be obtained by solving the Eqs. (18)–(22), which in this example become

$$
\begin{aligned}
\underset{Z \in \mathbb{Z}}{\text{maximize}} \quad & \sum_{(l, r_1, r_2, a_1, a_2, f)} \pi(l, r_1, r_2, a_1, a_2, f) U\big[Y_T(a_1, a_2, f)\big] && \\
\text{subject to} \quad & \sum_{a_i \in A_i} z(a_i \mid r_i) = 1, && \forall\ r_i \in R_i, i = 1, 2 \\
& 0 \le \pi(l, r_1, r_2, a_1, a_2, f) \le p(l, r_1, r_2, a_1, a_2, f), && \forall\ (l, r_1, r_2, a_1, a_2, f) \in S \\
& \pi(l, r_1, r_2, a_1, a_2, f) \le z(a_i \mid r_i), && \forall\ (l, r_1, r_2, a_1, a_2, f) \in S, i = 1, 2 \\
& \pi(l, r_1, r_2, a_1, a_2, f) \ge p(l, r_1, r_2, a_1, a_2, f) + \sum_{i=1,2} z(a_i \mid r_i) - 2, && \forall\ (l, r_1, r_2, a_1, a_2, f) \in S \\
& z(a_i \mid r_i) \in \{0, 1\}, && \forall\ a_i \in A_i, r_i \in R_i, i = 1, 2,
\end{aligned}
$$

where $Y_T(a_1, a_2, f)$ gives the consequences associated with the failure state f and the actions a_1 and a_2. If all the decision and chance nodes have binary states, then there are 8 decision variables (4 per each fortification decision) and $2^6 = 64$ paths, resulting in 4 equality constraints and 192 inequality constraints (in the second inequality constraint, the states a_i, r_i are implied by the selected path and third inequality constraints can be omitted by normalizing the utility function so that it attains positive values only).

5. Extensions to modeling chance constraints and multiple value nodes

Apart from the use of nonlinear utility functions $U(\cdot)$ in (14), risk preferences can be accounted through risk measures ρ that map decision strategies to non-negative real numbers and can be introduced as additional terms into the objective function or employed as constraints. In the following, we assume that, at the value nodes $v \in V$, the aim is to maximize the consequences $\mathcal{C}(s) = Y_v(s_{I(v)}) \in \mathbb{C}$, which are assessed using real numbers.

5.1. Absolute and lower-semi absolute deviation

Let $t \in \mathbb{R}$ be a given target level for consequences and define the non-negative deviation variables

$$\Delta_t^+(s) = \max\{0, \mathcal{C}(s) - t\}, \qquad \Delta_t^-(s) = \max\{0, t - \mathcal{C}(s)\}. \tag{23}$$

By construction, $\Delta_t^+(s)$ (respectively $\Delta_t^-(s)$) measures how much the consequence $\mathcal{C}(s)$ is above (below) the target level t. The deviations (23) can be precomputed for the information states $S_{I(v)}$ at the value node v. The expected downside risk (EDR) of the strategy $Z \in \mathbb{Z}$ relative to the target level t is

$$\rho_{\text{EDR}}(Z; t) = \sum_{s \in S} \pi(s) \Delta_t^-(s). \tag{24}$$

If t is chosen to be the expected value of consequences $\mathbb{E}[\mathcal{C} \mid Z] = \sum_{s \in S} \pi(s) Y_v(s_{I(v)})$ for the strategy Z, the corresponding non-negative deviation (decision) variables $\Delta^+_{\mathbb{E}[\mathcal{C}\mid Z]}(s)$, $\Delta^-_{\mathbb{E}[\mathcal{C}\mid Z]}(s)$ can be employed with the constraint

$$\mathcal{C}(s) - \Delta^+_{\mathbb{E}[\mathcal{C}\mid Z]}(s) + \Delta^-_{\mathbb{E}[\mathcal{C}\mid Z]}(s) = \mathbb{E}[\mathcal{C} \mid Z]$$

to capture the deviations from $\mathbb{E}[\mathcal{C}]$. The absolute deviation (AD) and the lower semi-absolute deviation (LSAD) are then given by

$$\rho_{\text{AD}}(Z) = \sum_{s \in S} \pi(s) \left[\Delta^+_{\mathbb{E}[\mathcal{C}\mid Z]}(s) + \Delta^-_{\mathbb{E}[\mathcal{C}\mid Z]}(s) \right] \tag{25}$$

$$\rho_{\text{LSAD}}(Z) = \sum_{s \in S} \pi(s) \Delta^-_{\mathbb{E}[\mathcal{C}\mid Z]}(s). \tag{26}$$

These measures can be used to augment the objective function through an additional additive term which penalizes for risk. For example, if the aim is to maximize expected consequences while accounting for risks through (lower semi-)absolute deviation, one possibility is to formulate the objective function as $\max_{Z \in \mathbb{Z}} \left\{ (1 - \phi)\mathbb{E}[X_v|Z] - \phi \rho_{\text{LSAD}}(Z) \right\}$ where $\phi \in [0, 1]$ is a weighting coefficient that reflects the DM's risk aversion. Alternatively, as an example of using risk measures to constrain feasible decision strategies, assume that the consequences are defined as profits reported in monetary terms. Then, the constraint $\rho_{\text{AD}}(Z) \leq 10$ MEUR would rule out any strategy $Z \in \mathbb{Z}$ whose profits can be expected to differ more than 10 MEUR from the expected profits $\mathbb{E}[\mathcal{C} \mid Z]$.

5.2. Chance constraints and Value-at-Risk

Probabilistic chance constraints can be modeled as linear inequalities on the path probabilities $\pi(s)$ which depend linearly on the decision variables. For example, to assess whether the consequences $\mathcal{C}(s)$ meet or exceed the stated target level $t \in \mathbb{R}$, we define the parameters

$$\Lambda_t(s) = \begin{cases} 1, & \text{if } \mathcal{C}(s) \geq t \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

For example, if the consequence is required to reach the target level t with a probability that is higher than or equal to a stated threshold level p_t, we have the constraint

$$\mathbb{P}\Big(\{s \mid \mathcal{C}(s) \geq t\} \mid Z \Big) = \sum_{s \in S} \pi(s) \Lambda_t(s) \geq p_t, \tag{28}$$

which is linear in the path probabilities $\pi(s)$. As for utilities, the terms $\Lambda_t(s), \forall s \in S$ can be readily derived from the information states $s_{I(v)} \in S_{I(v)}$.

In the present context where the probability distributions over consequences are discrete, the Value-at-Risk (VaR) risk measure for the strategy Z can be defined as

$$\text{VaR}_\alpha(Z) = F_Z^{-1}(\alpha) = \sup\{t \mid \mathbb{P}(s \mid \mathcal{C}(s) \leq t) < \alpha\}, \tag{29}$$

where F_Z^{-1} is the inverse function of the cumulative probability distribution $F_Z : \mathbb{C} \mapsto [0, 1]$ which is defined as $F_Z(t) = \sum_{\{s \mid \mathcal{C}(s) \leq t\}} \pi(s)$.

Because the probability distribution over the set of paths is discrete, the definition (29) means that consequences which are less than or equal to $\text{VaR}_\alpha(Z)$ can occur with a probability greater than α (Rockafellar & Uryasev, 2002). This is the case if $\text{VaR}_\alpha(Z)$ coincides with a consequence where the cumulative probability distribution function jumps from a level below α to one that exceeds α so that $\mathbb{P}\Big(\{s \mid \mathcal{C}(s) < \text{VaR}_\alpha(Z)\}\Big) < \alpha < \mathbb{P}\Big(\{s \mid \mathcal{C}(s) \leq \text{VaR}_\alpha(Z)\}\Big)$.

Constraints such as (28) can be employed to introduce VaR requirements. That is, if the probability $\alpha > 0$ is associated with the corresponding VaR level t^α_{VaR}, then the path probabilities for any feasible strategy Z must satisfy the constraint

$$\sum_{s \in S} \pi(s)[1 - \Lambda_{t^\alpha_{\text{VaR}}}(s)] \leq \alpha, \tag{30}$$

where $\Lambda_{t^\alpha_{\text{VaR}}}(\cdot)$ is defined as in (27). This approach can be generalized to introduce chance constraints on the states of nodes $k \in C \cup D$ as well. For instance, assume that the state at node k needs to be in some set $\tilde{S}_k \subset S_k$ with a probability which is less than or equal to $\tilde{p}_k$. This requirement can be represented by the constraint $\sum_{s \in S} \pi(s) \Lambda_{\tilde{S}_k}(s) \leq \tilde{p}_k$ where $\Lambda_{\tilde{S}_k}(s) = 1$ if $s_k \in \tilde{S}_k$ and $\Lambda_{\tilde{S}_k}(s) = 0$ otherwise. Thus, for example, for a decision node $k \in D$, one could require that the probability of having to employ exceptional decisions, as represented by the states in $\tilde{S}_k$, does not exceed the given probability level $\tilde{p}_k$.

5.3. Conditional Value-at-Risk

For the strategy $Z \in \mathbb{Z}$,the Conditional Value-at-Risk (CVaR) at the given probability level $\alpha > 0$ is the expected level of consequences, conditioned on the event that the realized consequence is in the $\alpha \in (0, 1]$ lower tail of the probability distribution. Contributions to this expectation come from (i) paths $s \in S^<_{\text{VaR}_\alpha(Z)} = \{s \in S \mid \mathcal{C}(s) < \text{VaR}_\alpha(Z)\}$ which lead to consequences strictly less than $\text{VaR}_\alpha(Z)$; and (ii) paths $s \in S^=_{\text{VaR}_\alpha(Z)} = \{s \in S \mid \mathcal{C}(s) = \text{VaR}_\alpha(Z)\}$ which lead to the consequence $\text{VaR}_\alpha(Z)$. The share of the probability of these latter paths that needs to be accounted in the computation of the CVaR level is the difference $\alpha - \mathbb{P}(\{s \mid \mathcal{C}(s) < \text{VaR}_\alpha(Z)\}) = \alpha - \sum_{s \in S^<_{\text{VaR}_\alpha(Z)}} \pi(s)$. Thus, as in Liesiö & Salo (2012), we define the risk measure $\text{CVaR}_\alpha(Z)$ as

$$\text{CVaR}_\alpha(Z) = \frac{1}{\alpha} \left(\sum_{s \in S^<_{\text{VaR}_\alpha(Z)}} \pi(s) \mathcal{C}(s) + \sum_{s \in S^=_{\text{VaR}_\alpha(Z)}} \left(\alpha - \sum_{s \in S^<_{\text{VaR}_\alpha(Z)}} \pi(s) \right) \mathcal{C}(s) \right). \tag{31}$$

By Proposition 1, the VaR and CVaR levels for a given probability level $\alpha > 0$ and strategy $Z \in \mathbb{Z}$ can be determined by solving the optimization problem (33)–(43) with precomputed parameters $c^* = \max\{\mathcal{C}(s) \mid s \in S\}$, $c^\circ = \min\{\mathcal{C}(s) \mid s \in S\}$, $M = c^* - c^\circ$ and $\epsilon = \frac{1}{2} \min\{|\mathcal{C}(s) - \mathcal{C}(s')| \mid |\mathcal{C}(s) - \mathcal{C}(s')| > 0, s, s' \in S\}$.

Proposition 1. *Choose $\alpha \in (0, 1]$ and let $\pi(s), \forall s \in S$, be the path probabilities for a strategy $Z \in \mathbb{Z}$. Then the optimization problem*

$$\min \quad \eta \tag{32}$$

$$\eta - \mathcal{C}(s) \leq M\lambda(s), \qquad \forall s \in S \tag{33}$$

$$\eta - \mathcal{C}(s) \geq (M + \epsilon)\lambda(s) - M, \qquad \forall s \in S \tag{34}$$

$$\eta - \mathcal{C}(s) \leq (M+\epsilon)\overline{\lambda}(s) - \epsilon, \qquad \forall s \in S \tag{35}$$

$$\eta - \mathcal{C}(s) \geq M(\overline{\lambda}(s) - 1), \qquad \forall s \in S \tag{36}$$

$$\overline{\rho}(s) \leq \overline{\lambda}(s), \qquad \forall s \in S \tag{37}$$

$$\pi(s) - (1 - \lambda(s)) \leq \rho(s) \leq \lambda(s), \qquad \forall s \in S \tag{38}$$

$$\rho(s) \leq \overline{\rho}(s) \leq \pi(s), \qquad \forall s \in S \tag{39}$$

$$\sum_{s \in S} \overline{\rho}(s) = \alpha, \tag{40}$$

$$\overline{\lambda}(s), \lambda(s) \in \{0, 1\}, \qquad \forall s \in S \tag{41}$$

$$\overline{\rho}(s), \rho(s) \in [0, 1], \qquad \forall s \in S \tag{42}$$

$$\eta \in [c^\circ, c^*], \tag{43}$$

has a solution such that the optimum value $\eta^* = \text{VaR}_\alpha(Z)$ *and* $\text{CVaR}_\alpha(Z) = \frac{1}{\alpha}\sum_{s \in S} \overline{\rho}(s)\mathcal{C}(s)$.

Proof. See Appendix A.

An inspection of the proof of Proposition 1 shows that for any feasible solution to the constraints (33)–(43), the expression $\sum_{s \in S} \overline{\rho}(s)\mathcal{C}(s)/\alpha$ gives the correct $\text{CVaR}_\alpha(Z)$ risk measure for Z. Thus, if the expectation of consequences in the lower α-tail of the probability distribution over consequences is required to be greater than or equal to the lower bound t^α_{CVaR}, this requirement can be enforced by adding the constraints (33)–(43) and $\sum_{s \in S} \overline{\rho}(s)\mathcal{C}(s) \geq \alpha t^\alpha_{\text{CVaR}}$ to (18)–(22).

One approach to address trade-offs between the maximization of conditional expectations for different levels of α is to treat these as different objectives with respective weighting coefficients. Thus, combining the unconditional expectation with the selected $\alpha \in (0, 1)$ for CVaR leads to the problem

$$\underset{Z \in \mathbb{Z}}{\text{maximize}} \quad w\left(\sum_{s \in S} \pi(s)\mathcal{C}(s)\right) + (1-w)\left(\frac{1}{\alpha}\sum_{s \in S} \overline{\rho}(s)\mathcal{C}(s)\right) \tag{44}$$

$$\text{subject to} \quad (18) - (22), (33) - (43) \tag{45}$$

whose solution depends on the parameter $w \in (0, 1)$ that represents trade-offs between (i) the overall expectation in the first term of (44) and (ii) the expectation in the lower α-tail as expressed by the second term. This parameter can be elicited by asking the DM to answer much of the overall expectation the DM is willing to give up in return for improving the CVaR level by one unit, which gives the ratio $\frac{1-w}{w}$. Note that the linear model assumes that the answer to this question does not depend on the overall expectation $\sum_{s \in S} \pi(s)\mathcal{C}(s)$.

5.4. Multiple value nodes and objectives

The consideration of CVaR levels together with the maximization of expected consequences is an example of the more general case in which there are multiple objectives $n_V > 1$ (see, e.g., Antunes, Alves, & Clímaco, 2016) associated with different value nodes. Then, if these objectives are compared based on their expected consequences, attention can be focused on non-dominated strategies $Z \in \mathbb{Z}_{ND}$. In this case, the dominated strategies can be defined so that Z is non-dominated if and only if there is no other feasible strategy $Z' \in \mathbb{Z}_F$ whose expectation is equal to or higher than that of Z at each value node and strictly higher for at least one value node, i.e.,

$$Z \in \mathbb{Z}_{ND} \iff Z \in \mathbb{Z}_F \wedge \nexists Z' \in \mathbb{Z}_F \text{ such that } \mathbb{E}[\mathcal{C}_v \,|\, Z'] \geq \mathbb{E}[\mathcal{C}_v \,|\, Z], \ \forall v \in V,$$

where $\mathbb{E}[\mathcal{C}_v \,|\, Z] = \sum_{s \in S} \pi(s)\mathcal{C}_v(s)$ denotes the expectation at value node $v \in V$ and the inequality is strict for at least one value node $v \in V$. Because the strategies are choices from a discrete set of alternatives, this is a discrete multi-objective optimization problem (MOO) in which the objectives correspond to the maximization of expectations for different value nodes. Thus, it can be solved with algorithms for this problem class. Holzmann & Smith (2018) provide an extensive review and propose an algorithm based on augmented Tchebychev norm, in which choices about the initial step size need to be made.

The weighting approach in (44) or, more generally, the maximization of the expression $\sum_{v \in V} w_v \mathbb{E}[\mathcal{C}_v \,|\, Z]$ can be employed to generate non-dominated strategies. However, a shortcoming of this approach is that it does not necessarily generate *all* non-dominated strategies even if all non-negative weighting coefficients $w_v \geq 0, \forall v \in V$, such that $\sum_{v \in V} w_v = 1$, are employed. This will be the case if a non-dominated strategy $Z' \in Z_{ND}$ is dominated by a weighted linear combination of other non-dominated strategies $Z_1, \ldots, Z_k \in Z_{ND}$ so that for some selection of positive weights $\omega_i > 0$ with $\sum_{i=1}^k \omega_i = 1$, it holds that $\mathbb{E}[\mathcal{C}_v \,|\, Z'] \leq \sum_{i=1}^k \omega_i \mathbb{E}[\mathcal{C}_v \,|\, Z_i]$ for all $v \in V$ (with a strict inequality for some $v \in V$).

This notwithstanding, the weighting approach can be adapted to generate all non-dominated strategies. First, if $Z' \in \mathbb{Z}_{ND}$ is a non-dominated strategy, then it can be eliminated from consideration in the computation of further candidates for non-dominated strategies through the linear constraint

$$\sum_{\{(s_i, s_{I(i)}) \,|\, z'(s_i \,|\, s_{I(i)}) = 0\}} z(s_i \,|\, s_{I(i)}) + \sum_{d \in D} \prod_{i \in I(d)} |S_i| - \sum_{\{(s_i, s_{I(i)}) \,|\, z'(s_i \,|\, s_{I(i)}) = 1\}} z(s_i \,|\, s_{I(i)}) \geq 1, \tag{46}$$

where $z'(s_i \,|\, s_{I(i)}), s_i \in S_i, s_{I(i)} \in S_{I(i)}$ are the decision variables for Z'. In (46), the left side for strategy Z will be greater than one if and only if Z differs from Z'.

Second, if $Z' \in Z_{ND}$, then further candidates for non-dominated strategies must not be dominated by Z'. A necessary condition for this can be stated by defining the binary variables $\lambda^+_{Z',v}(Z)$, $\lambda^-_{Z',v}(Z) \in \{0, 1\}, \forall v \in V$ so that $\lambda^+_{Z',v}(Z) + \lambda^-_{Z',v}(Z) = 1$ and

$$\mathbb{E}[\mathcal{C}_v \,|\, Z] \leq \mathbb{E}[\mathcal{C}_v \,|\, Z'] + M\lambda^+_{Z',v}(Z) \tag{47}$$

$$\mathbb{E}[\mathcal{C}_v \,|\, Z'] \leq \mathbb{E}[\mathcal{C}_v \,|\, Z] + M\lambda^-_{Z',v}(Z) \tag{48}$$

where M is a large constant (e.g., greater than $c^* = \max_{s \in S}\mathcal{C}(s)$). Now, consider any solution to (47)–(48) such that $\lambda^+_{Z',v}(Z) = 0, \forall v \in V$. Then $\mathbb{E}[\mathcal{C}_v \,|\, Z]$ is either strictly less than $\mathbb{E}[\mathcal{C}_v \,|\, Z']$ for all $v \in V$ so that Z is dominated by Z'; or if not, there exists some $v' \in V$ such that $\mathbb{E}[\mathcal{C}_{v'} \,|\, Z] = \mathbb{E}[\mathcal{C}_{v'} \,|\, Z']$ so that the values of the variables $\lambda^-_{Z',v'}(Z) = 1, \lambda^+_{Z',v'}(Z) = 0$ can be switched to $\lambda^-_{Z',v'}(Z) = 0, \lambda^+_{Z',v'}(Z) = 1$, in which case the constraints (47)–(48) are still satisfied. Thus, for any strategy Z which is not dominated by Z' there will exist a solution such that

$$\sum_{v \in V} \lambda^+_{Z',v}(Z) \geq 1, \ \ Z' \in Z_{ND}. \tag{49}$$

The above constraints (47)–(48) and (49) constitute a necessary but not a sufficient condition. That is, the candidate solution Z'

which maximizes $\sum_{v\in V} w_v \mathbb{E}[\mathcal{C}_v \mid Z]$ may be dominated by Z' if the value nodes can be partitioned into non-empty sets $V^{=} \cup V^{<} = V$ such that $\mathbb{E}[\mathcal{C}_v \mid Z''] = \mathbb{E}[\mathcal{C}_v \mid Z'], v \in V^{=}$ and $\mathbb{E}[\mathcal{C}_v \mid Z''] < \mathbb{E}[\mathcal{C}_v \mid Z'], v \in V^{<}$, i.e., Z'' is dominated by Z'. Consequently, explicit dominance checks are needed to evaluate whether the candidate solution Z'' is non-dominated. If it is, the set of non-dominated strategies can be updated by adding Z'' to this set and by introducing the constraint (46) to eliminate Z'' from further consideration. Adding this constraint to (47)–(48) for Z'' does not prevent the computation of alternative strategies whose expectations are the same for all value nodes, as such strategies do not dominate each other.

Next, the algorithm can be iterated by maximizing $\sum_{v\in V} w_v \mathbb{E}[\mathcal{C}_v \mid Z]$ to generate further candidate strategies and augmenting the sets of non-dominated strategies and constraints. If the algorithm is terminated prematurely, it provides a viable set of non-dominated strategies that can be examined to check in what region the expected values of any non-dominated strategies that have not yet been generated would lie. Because the number of non-dominated strategies is finite, the algorithm will generate them all. Moreover, if the DM's preferences are represented by a real-valued function which is strictly increasing in each of the expectations $\mathbb{E}[\mathcal{C}_v|Z], v \in V$, the computation of non-dominated strategies generates all the solutions that can be optimal for any such function.

The number of non-dominated solutions and the effort that is required to compute them depends on the problem characteristics. In general, this effort tends to grow with (i) the number of objectives and feasible decision strategies; and (ii) the presence of negative correlations between the objectives. To see why this the case, assume that there are two objectives v and v' such that the expected consequences $\mathbb{E}[\mathcal{C}_v \mid Z]$ and $\mathbb{E}[\mathcal{C}_{v'} \mid Z]$ are perfectly correlated across the set of feasible decision strategies $Z \in \mathbb{Z}_F$. Then, there is a positive linear relationship between $\mathbb{E}[\mathcal{C}_v \mid Z]$ and $\mathbb{E}[\mathcal{C}_{v'} \mid Z]$ and the strategies which maximize these two objectives are the same. Thus, there is only one non-dominated strategy assuming that these maximization problems do not have alternative optimal solutions. At the other end of the spectrum, if the objectives v and v' have a perfect negative correlation of minus one, there is a negative linear relationship between $\mathbb{E}[\mathcal{C}_v \mid Z]$ and $\mathbb{E}[\mathcal{C}_{v'} \mid Z]$ and consequently *all* feasible decision strategies are non-dominated. Between these extremes, the number of non-dominated strategies can be expected to be larger when there is a strong negative correlation between the objectives.

One reason for computing all non-dominated solutions is that in many problems the objectives are negatively correlated across the set of alternatives (e.g., low cost vs. high quality). In such problems, restricting the attention to the solutions generated by the weighted sum approach may suggest comparatively *extreme* strategies only (i.e., quite costly with very high quality; or unsatisfactory quality at a low cost) while neglecting more *balanced* strategies that are of interest to the DM in that they perform reasonably well on many objectives.

If there is a single value node v with real-valued consequences, the above multi-objective algorithm can be adapted to determine all the strategies which are non-dominated in the sense of first-order stochastic dominance. Specifically, the consequences associated with the information states $s_{I(v)}$ can be viewed as target levels so that the function $\Lambda_t(s)$ in (27) is defined for the different target levels t that correspond to the different consequences v. Then, for strategy Z, the probability of meeting or exceeding the level t is the expectation $\mathbb{E}[\Lambda_t \mid Z]$, which can be treated as the objective that corresponds to the target level t. Specifically, strategy Z will dominate Z' (in the sense of first-order stochastic dominance; see, e.g., Liesiö & Salo, 2012; Rockafellar & Uryasev, 2002) if and only if $\mathbb{E}[\Lambda_t \mid Z] \geq \mathbb{E}[\Lambda_t \mid Z']$ for all target levels t with a strict inequality for some target level. Often, it is reasonable to limit the attention to these stochastically non-dominated strategies, because only they can be optimal if the DM's utility function over consequences is known to be increasing. Thus, this approach can be used to prune the set of viable strategies based on weak assumptions about the DM's utility function.

The ability to screen non-dominated strategies can be particularly useful in group decision making problems as well, because there is no need to build a consensual representation of the group's utility function. Rather, once the non-dominated strategies have been identified, methods of multi-criteria decision analysis can be deployed to support the final selection (see Salo, Hämäläinen, & Lahtinen, 2021 for an overview).

6. Computational experiments

We next report results from computational experiments to demonstrate the viability of Decision Programming. All implementation were coded in Julia 1.1.0, using the package JuMP to implement models which were solved with Gurobi 8.1.0. using 2 out of 8 available threads. All problem instances were solved with an Intel Xeon E3-1230 v5 desktop clocked at 3.40 GHz with 32 GB RAM running Windows 10 x64 Education Edition. The open-source code for these examples and supporting documentation are available at http://github.com/gamma-opt/DecisionProgramming.jl.

6.1. N-monitoring problem

The N-monitoring problem has the same structure as the double monitoring problem in Section 4.1 except that there are N binary reinforcement decisions of which each is informed by its own load report with two states, *low* and *high*. For every problem size, we solve 100 instances with randomly generated data, both with and without the probability cuts in Section 3.7.

Data sets with plausible characteristics were generated as follows. The utility of the structure not failing was set to 100 and that of failing to 0. For the load node L, the probability p_H of the high load state was generated from the uniform distribution $U(0, 1)$ over the unit interval and the remaining probability $p_L = 1 - p_H$ was assigned to the low load state. All reports were conditionally independent of each other given the load. For both loads, the probability of receiving a correct report was $\max\{x, 1 - x\}$ where x was generated from the uniform distribution $U(0, 1)$. Further realizations of x and y from $U(0, 1)$ were used to set the prior probability of failure in the case of high load to $\max\{x, 1 - x\}$ and that in the case of low load to $\min\{y, 1 - y\}$. The costs of fortification $c_i, i = 1, \ldots, N$ actions were also generated from $U(0, 1)$. The posterior probability of failure after implementing the actions $A \subseteq \{1, \ldots, N\}$ was taken to be that of the prior divided by $e^{\sum_{i\in A} c_i}$. Thus, the actions served to mitigate the possibility of failure and the more costly actions are effective in doing so. This is an example of a portfolio problem with endogenous uncertainties in which the failure probability depends on all fortification decisions.

Table 1 shows the solution times in seconds for randomly generated instances, comparing the computational performance with and without the probability cuts discussed in Section 3.7. At each decision node, there are 4 decision variables (i.e., $z(s_i \mid s_{I(i)})$, two possible decisions for each information state). Thus, for the given number of reports N, there are 4^N different decision strategies so that in the case of 9 decisions, there $4^9 = 262\ 122$ strategies. Because all nodes are binary and there are $2 + 2N$ nodes, the number of real variables (i.e., $\pi(s)$ for path probabilities) is 2^{2N+2} The results show the average (Avg) and standard deviation (Std) for 100 problem instances. A time limit of 25 200 seconds (7 hours) was used. The entry "-" denotes cases for which no solution could be found within thee 7h time limit. As can be observed, the probability cuts greatly improve the performance of the solver.

Table 1
Results for samples of 100 randomly generated N-monitoring instances.

	Number of variables		Without probability cuts		With probability cuts	
# Decisions N	Binary	Real	Avg	Std	Avg	Std
2	8	64	0.01	0.01	0.01	0.00
3	12	256	0.12	0.08	0.02	0.01
4	16	1 024	0.79	0.53	0.07	0.02
5	20	4 096	5.94	2.80	0.35	0.19
6	24	16 384	77.35	46.31	2.44	1.63
7	28	65 536	676.35	468.09	20.58	17.48
8	32	262 144	8 474.00	7 377.28	268.93	330.89
9	36	1 048 576	-	-	1 727.19	2 880.20

Table 2
Results for the pig farm problem for different numbers of decision periods.

# Months	Optimal value (DKK)	Solution time (s)
3	764	0.01
4	727	0.04
5	703	0.62
6	686	19.52
7	674	617.21

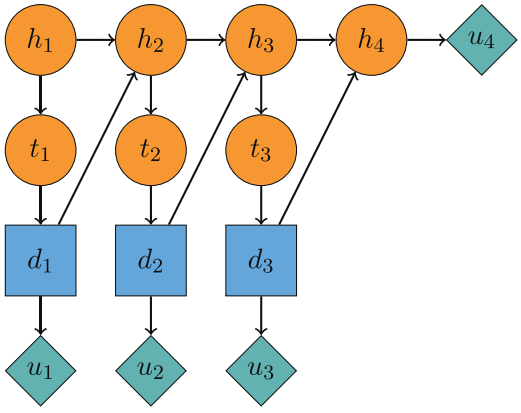

Fig. 3. The pig farm problem with three decision periods (Lauritzen & Nilsson, 2001).

6.2. The pig farm problem

In the pig farm problem (see Lauritzen & Nilsson, 2001), a veterinary doctor visits a pig farm each month to test each pig for a disease and decides, based on the uncertain test result, whether or not to inject the pig with a drug that helps prevent and cure the disease at a cost of 100 DKK. After four months, healthy pigs are sold for 1 000 DKK and diseased ones for 300 DKK. Because the doctor has no access to individual records for each pig, she has to make the treatment decision based on the most recent test result without knowing earlier injection decisions. This problem is represented by a limited memory influence diagram (LIMID) in Fig. 3.

Despite its practical relevance and conceptual simplicity, this problem is not *soluble*. As a result, the Single Policy Update algorithm (which is based on solving a series of local optimization problems) is not guaranteed to converge to the global optimum (for details, see Lauritzen & Nilsson, 2001). With Decision Programming, this problem can nevertheless be solved to global optimality relatively efficiently. Table 2 presents the optimal solutions and their computation times both for the original four-month version of the problem with three decision periods (in which there are 64 different strategies, corresponding to $4 \times 4 \times 4$ combinations of the four local decision strategies in each of these three months), as well as extensions for the same problem up to seven monthly decision periods with the same numerical parameters. Here, the number of strategies grows rapidly with the number of periods, meaning that solving the problem through explicit enumeration becomes increasingly impractical. In the case of seven periods, for example, there are $4^7 = 16\,384$ decision strategies.

The formulations in Section 5.4 help determine the non-dominated strategies based on the consideration of the two objectives of maximizing (i) the overall expected utility and (ii) the conditional expectation in the lower $\alpha = 20\%$ tail. Fig. 4 shows the overall expected utility (assuming risk-neutral preferences over monetary consequences) and the conditional CVaR expectation in the lower tail of for each of the 64 decision strategies for this 4-month pig problem.

In Fig. 4, the four non-dominated strategies are connected and marked with orange circles, while the remaining 60 dominated strategies are marked with blue circles. Going from left to right, the first non-dominated strategy has the highest expected utility, while the fourth one has the highest conditional expectation in the 20% lower tail. The vaccination policies in these non-dominated strategies are, respectively, as follows:

1. Never treat at 1st month. Treat at 2nd and 3rd month if and only if test results are positive.
2. Never treat at 1st and 2nd month. Treat at 3rd month if and only if test results are positive.
3. Never treat at 1st and 3rd month. Treat at 2nd month if and only if test results are positive.
4. Never treat at any of the 3 months.

Thus, the local strategy of never treating in the first month is a *robust* decision recommendation as it is contained in all non-dominated strategies and thus in the set of 'core' selections in the sense of Robust Portfolio Modelling (RPM) (Liesiö, Mild, & Salo, 2007; 2008). Furthermore, all local strategies which suggest treatments based on negative test results can be ruled out from consideration, because they are not included in any non-dominated strategies and thus belong to the set of 'exterior' RPM selections.

The chosen level $\alpha = 20\%$ could be have been set at other levels, too (say, at 5% or 10%), leading to the introduction of other objective functions that could complement or replace the objective associated with $\alpha = 20\%$. Furthermore, because the fourth year cash flow is either 1 000 or 300, preceded by either 0, 1, 2 or 3 injections at a cost of 100 during the first three months, the final cash position will be 0, 100, 200, 300, 700, 800, 900 or 1 000 (all in DKK) whereby the probability of each of these positions depends on the selected decision. For example, in the pig farm example there are eleven first-order stochastically non-dominated strategies. Among these, there are seven that have an expected final cash position over 670 DKK; but none of these seven strategies involve any first period injection. Thus, in the light of this information, one could consider omitting the first period test entirely. This notwithstanding, the eleven stochastically non-dominated

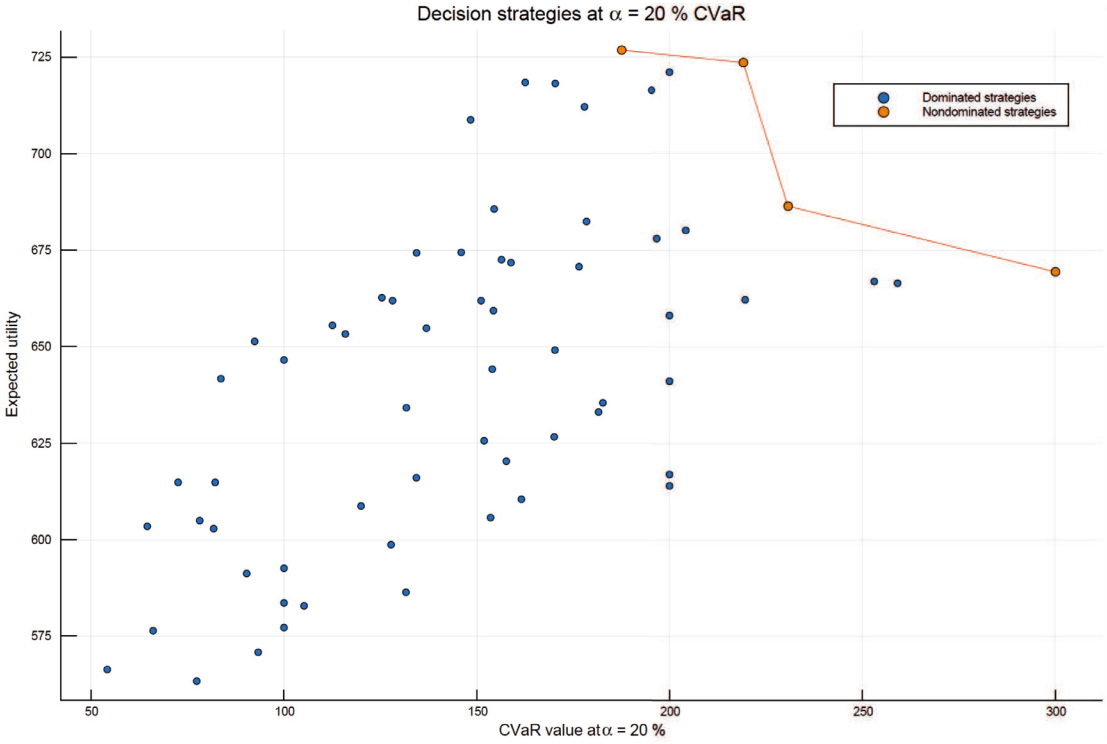

Fig. 4. Expected utilities and conditional expectations in the lower $\alpha = 0.20$ tail. The four non-dominated strategies are connected and marked with orange circles.

strategies are instructive in that they also reflect the preferences of a highly risk averse DM who is intent on maximizing the probability of having a strictly positive final cash position. In this case, it is optimal to provide first period injections regardless of the test result. This strategy is stochastically non-dominated, albeit with a very low expected cash position at 583 DKK.

6.3. Computational performance, comparative advantages and future extensions

The computational performance of Decision Programming can be enhanced in several ways. For example, if the influence diagram contain chance nodes that do not belong to the information set of any decision node, then this structure can be exploited as follows. Specifically, let $\overline{D} = D \cup \{i \mid \exists\ j \in D \text{ s.t. } i \in I(j)\}$ be the union of decision nodes and their information sets and similarly $\overline{C} = C \cup \{i \mid \exists\ j \in C \text{ s.t. } i \in I(j)\}$. Then, for a given strategy Z, the path probability s is $p(s) = p(s_{\overline{C}}) = \prod_{i \in C} p(s_i \mid s_{I(i)})$ for any active path $s \in S_Z = \{s \in S \mid Z(s_{I(j)}) = s_j, j \in D\}$ and 0 otherwise. Now, define binary decision variables $x(s') \in \{0,1\}, s' \in S_{\overline{D}} = \{s_{\overline{D}} \mid s \in S\}$ such that the constraints $1 - [|D| - \sum_{j \in D} z(s'_j \mid s'_{I(j)})] \leq x(s') \leq \frac{1}{|D|} \sum_{j \in D} z(s'_j \mid s'_{I(j)})$ hold for $s' \in S_{\overline{D}}$. Then, $x(s_{\overline{D}}) = 1$ if and only if the $s_{\overline{D}}$ is contained in an active path $s \in S_Z$ and the objective function can be written as $\mathbb{E}[\mathcal{U}(s) \mid Z] = \sum_{s \in S_Z} x(s_{\overline{D}}) p(s_{\overline{C}}) \mathcal{U}(s_V)$. Thus, the number of decision variables $x(s_{\overline{D}})$ is at most $i \in \prod_{i \in \overline{D}} |S_i|$, which can be smaller than the total number of paths $|S| = i \in \prod_{i \in C \cup D} |S_i|$. In (20), it is also possible to replace the decision constraints for each decision by the single constraint $\pi(s) \leq x(s_{\overline{D}})$.

Further improvements in computational performance can be sought by noting that because the path probabilities satisfy the constraint $\pi(s) \leq p(s)$ in (19) and because the utilities can be normalized into the [0,1] interval so that $\mathcal{U}(s) \leq 1, s \in S$, we have $\sum_{s \in S'} \pi(s) \mathcal{U}(s) \leq \sum_{s \in S'} p(s)$. Thus, the contribution of any given subset of paths $S' \subsetneq S$ to the total expected utility is bounded from above by $\sum_{s \in S'} p(s')$. This can be exploited to obtain an approximate solution by omitting paths with very low probabilities and by computing the optimum based on the more probable paths. The computed optimum will then provide a lower bound for the true optimum, while the sum of probabilities for the omitted paths gives an upper bound for how much higher than the computed approximate solution the true optimum can be. For example, in the N monitoring example in Section 6.1. with an initial binary load L and $N = 8$ binary reports on this load, there are $2^9 = 512$ subpaths of length nine, defined by the initial load L followed by the eight reports $R_i, i = 1 \ldots, 8$. Based on the probability distributions in Section 6.1, the 62% most probable subpaths account for about 99% of the total probability. Hence, a solution which is within 1% of the optimum can be obtained by using paths which are extensions of these most probable subpaths.

In comparison with solution approaches based on the explicit enumeration of strategies, Decision Programming has comparative advantages when the total number of strategies is large relative to the number of paths. For example, in the N monitoring example, it could be of interest to assess what benefits could be gained by sharing some or even all measurement reports to inform the actions. In this case, the number of paths would remain the same, but the number of decision strategies would grow extremely rapidly. To see this, assume that all information is shared to the actions. Then each of the 2^N possible combinations of reports corresponds to an information state that is available to the N actions. For each of these combinations, one could, in principle, select any one of the 2^N possible combinations of binary actions at the N action nodes. Because these selections can be made separately for each information state, the number total number of strategies becomes $(2^N)^{(2^N)}$.

Concretely, for four $N = 4$ actions there are $2^{10} = 1\ 024$ paths. Now, if all reports are shared among the actions, the number of strategies grows from $4^4 = 256$ to $(2^4)^{(2^4)} = 1.84 \times 10^{19}$. Still, the size of the optimization model (17)–(22) grows rather moderately. The number of binary decision variables $z(s_i \mid s_{I(i)})$ grows from 16 to 128, the number of equality constraints (18) grows from 8 to 48, and the number inequality constraints (19)-(20) stays unchanged

at 1 024 and 4 096, respectively. This problem is small enough so that it can be promptly solved to optimality.

Within this setup, it is possible to assess if the benefits of sharing information about the actions outweigh the possible costs of such information sharing. Such an assessment can be carried out only on condition that the optimal solution can be determined both (i) when the information is shared and (ii) when it is not. In the latter case, the no-forgetting assumption does not hold and dynamic programming approaches cannot be deployed effectively. From this perspective, we believe that Decision Programming has advantages in the class of problems that support the *design of systems* in which information is to be shared and exploited optimally.

We also note that the formulation (17)–(22) has been applied to compute optimal strategies for a realistic Prognostics and Health Management problem (Mancuso, Compare, Salo, & Zio, 2021) with such a large number of strategies that the solution could not have been derived through explicit enumeration. This influence diagram in this application has seven five-state chance nodes and two three-state decision nodes so that the number of paths is $5^7 \times 3^2 = 703\ 125$ paths. Each of the two decision nodes are informed by two five-state chance nodes and thus have twenty-five information states. The total number of strategies is therefore $3^{25} \times 3^{25} \approx 7.18 \times 10^{22}$.

Decision Programming can be extended to problems represented by *hybrid* influence diagrams (cf. Li & Shenoy (2012)), in which some of the decision and random variables associated with decision and chance nodes, respectively, are continuous. There are several cases, depending on whether random variables, decision variables or both are continuous. Here, we offer ideas for approaching such problems, under the assumption that the domains of all continuous variables are compact and that the utility and probability density functions are continuous.

First, problems involving discrete decisions based on some continuous random variables can be tackled with techniques such as dynamic discretization (Neil, Tailor, & Marquez, 2007). That is, if the random variable X_i, $i \in C$, is represented by the real-valued random variable $r \in \mathbb{R}$ with the probability density function $f_i(r \mid s_{I(i)})$ for the discrete information states $s_{I(i)} \in S_{I(i)}$, then this variable can be discretized into n_i states by defining the intervals $[r_k, r_{k+1}), k = 1, \ldots, n_i$ with probabilities $\int_{r_k}^{r_{k+1}} f_i(r \mid s_{I(i)})dr$. If the information set of this chance node i contains other chance nodes with continuous random variables (say, X_j, $j \in I(i)$), the discretization needs to proceed in the order of increasing indices so that the discrete information sets $S_{I(i)}$ are defined first. In deriving the conditional probabilities (1), information about the distribution of X_j as well as the conditional distribution of X_i given X_j, $j \in I(i)$ will be needed. Multivariate uncertainties described by m-dimensional random variables $r^m \in \mathbb{R}^m$ could be accommodated by defining intervals over each of the m dimensions and by deriving probabilities by integrating over the resulting m-boxes.

The formulation based on binary variables $x_{\overline{D}}(s)$, as outlined at the beginning of this section, can be very useful in discretizing continuous random variables that are not in the information set of any decision node. In this case, increasing the granularity of the approximation does not much affect computational performance, because the number of variables or constraints stays unchanged although the number of paths in the objective function grows. This would be would be the case, for instance, in the N monitoring example where the initial load L and the magnitude of the failure F do not belong to the information sets of the fortification actions A_i. Thus, the distributions of L and F could be approximated with many intervals without increasing the computation time markedly. Such approximations can be guided by minimizing the Kullback-Leibler distance between the discretized distribution and the underlying continuous distribution (for an illustration, see Yet et al., 2018). Even other approximations techniques can also be employed (see, e.g., Hammond & Bickel, 2013).

If there is some continuous random variable which belongs to the information set of a discrete decision variable, then improving the accuracy of the approximation increases the number of paths and constraints significantly. However, because the number of discrete strategies is finite, there then exists a partition of the domains of the continuous variable such that one of the discrete strategies is optimal over each interval in this partition. Thus, one possibility is to explore such partitions iteratively and to solve the resulting optimization models, keeping track of which strategies perform best over the different intervals. Comparable simulation-optimization approaches have been tackled in the context of diagnostic testing problems, for example (see, Hynninen, Vilkkumaa, & Salo, 2020; Müller, Berry, Grieve, Smith, & Krams, 2007).

If there are continuous decision variables with discrete information states, then the task is to choose which real-valued decision should be chosen for each information state. Here, one possibility is to generate manageable numbers of candidates (for instance through sampling) that represent possible local decision strategies and to solve the resulting problems repeatedly to identify 'good' strategies which are optimal within their own set of candidates and from which the best performing ones are taken forward to generate recommendations. In principle, the generation of additional candidates for local decision strategies could be guided by applying ideas from augmented nested sampling which has been applied successfully in MEU problems in two-stage decisions with a single endogenous uncertainty (see, e.g., Bielza, Müller, & Rios Insua, 1999; Ekin, Polson, & Soyer, 2017; Ekin, Walker, & Damien, 2020). Yet, in our context the characterization of the joint probability distribution over all decisions and chance events would in most cases prove challenging. Moreover, it could be hard to provide convergence guarantees, because some of the conditional probabilities at chance nodes may be zero for some information states.

Importantly, the generality of the MILP-formulation makes it possible to exploit remarkable improvements in professional-grade solver implementations such as Gurobi and CPLEX in a relatively straightforward manner instead of relying on ad-hoc implementations which are not readily available, tend to be problem specific and may provide few guarantees for having followed sound software engineering practices in terms of versioning, updating and continuous improvement. In this regard, our contribution represents an important step forward in addressing an increasingly large class of problems based techniques such as decomposition, parallelization and heuristic methods that have made considerable inroads in solving other challenging MILPs. Here, there are two particularly promising avenues for investigation. The first is to investigate the formulation (17)–(22) through the lenses of combinatorial analysis and convex analysis on polyhedral spaces to derive stronger formulations from the standpoint of LP relaxation. This would suggest additional valid inequalities and/or relaxations of the integrality constraints on $z(s_j \mid s_{I(j)})$. The second direction is to investigate decomposition approaches that exploit the problem structure to identify possibilities for parallelization. In particular, because for any strategy $Z \in \mathbb{Z}$ the set of active paths $s \in S_Z$ is a small subset of all paths, it is possible develop column generation-based approaches (using a professional-grade framework such as Solving Constraint Integer Programs, see https://www.scipopt.org/). Both avenues, which have proven very efficient in other challenging MILPs, such as vehicle routing and scheduling problems, represent exciting research directions for future work.

7. Summary and conclusions

In this paper, we have developed Decision Programming as an MILP optimization approach for solving mixed-integer multi-stage

decision problems with discrete decisions and chance events. Such problems can be represented as influence diagrams, including LIMIDs in which the usual 'no-forgetting' assumption may not hold. In our approach, risk preferences can be captured through non-linear utility functions over consequences or, alternatively, by extending the objective function with terms for risk measures or by introducing risk constraints. Multiple objectives can be handled, for instance, by using a weighted additive linear function to aggregate consequences (or their utilities) across different value nodes. The set of all non-dominated strategies, too, can be computed with MILP by employing a weighted linear objective function together with the sequential introduction of constraints to eliminate dominated strategies and already discovered non-dominated strategies from consideration.

In the context of stochastic optimization, Decision Programming is particularly useful in mixed-integer decision problems where the probabilities in the scenario tree depend endogenously on earlier integer-valued decisions. This ability to handle endogenous uncertainties can be helpful, for instance, when appraising R&D and marketing investments, because the size of the market as well as the products' market performance are often contingent on these earlier decisions. From this perspective, the proposed approach can be viewed as a generalization of Contingent Portfolio Programming (see Gustafsson & Salo (2005) and the on-line companion) to problems where the probabilities of chance events associated with branches of the scenario tree depend on project selection decisions.

Importantly, the Decision Programming framework can be employed to address problems that are not amenable to dynamic programming techniques, such as problems in which earlier decisions cannot be recalled, there are multiple agents, or deterministic and chance constraints make it impractical or impossible to conventional techniques. Therefore, although Decision Programming has parallels to developments in stochastic mixed-integer dynamic programming (such as employing mathematical programming formulation to find optimal policies, as in the seminal work of Manne (1960) and ensuing literature; see Bertsekas (2012) for a thorough exposition), Decision Programming makes it possible to tackle a broader class of problems by exploiting the expressiveness of influence diagrams for problem structuring, whereafter the equivalent MILP formulations that can be solved using off-the-shelf MILP solvers.

Based on our numerical experiments, the Decision Programming approach can be used to solve problems of realistic size to optimality, even if it does suffer from the curse of dimensionality just like other exact approaches for solving dynamic problems with a larger number of decision periods and uncertainties. We have therefore outlined approaches for enhancing its computational performance, for example by exploiting the properties of the problem structure or by assessing which paths could be eliminated from consideration based on their low probabilities in order to compute good approximate solutions. We also believe that future research is warranted for investigating how techniques which have been proposed for convex optimization problems with continuous variables (such as those proposed by Dupačová, Gröwe-Kuska, & Römisch, 2003) can be adapted to deal with continuous decision and random variables.

In summary, Decision Programming holds promise in extending the expressiveness of influence diagrams to problems in which it may be necessary to account for the probability distribution or decision consequences and their uncertainties, the presence of multiple objectives or the interdependencies between decisions that taken by multiple agents. At the same time, it provides a structured and systematic approach so that practitioners need not be overly concerned with specific problem characteristics (such as whether or not the regularity or no-forgetting assumptions are fulfilled) in deciding how the problem should be formulated as an optimization model. This helps circumvent the need to implement different solution methods, allowing practitioners to focus on modelling, while relying on the maturity and ever improving performance of mathematical programming solvers.

Acknowledgements

This research has been partly funded bythe project *Platform Value Now* of the Strategic Council of the Academy of Finland (funding decision number 314207) and by the Academy of Finland project *Decision Programming: A Stochastic Optimization Framework for Multi-Stage Decision Problems* (funding decision number 332180).

Appendix A

Proof of Theorem 1. Let $Z \in \mathbb{Z}$ and take any path $s \in S$. The information set of the first node $k = 1$ is empty. If this node is a chance node, the random variable X_1 does not depend on Z and thus $\pi_1(s) = \mathbb{P}(X_1 = s_1) = \mathbb{P}(X_1 = s_1 \,|\, Z)$. If it is a decision node, there are two cases. First, if $Z_1 = Z_1(\emptyset) = s_1$, it follows that $\mathbb{P}(X_1 = s_1 \,|\, Z) = 1$ while (5) gives $z(s_1) = 1$. Thus, by (8) we have $\pi_1(s) = z(s_1) = 1 = \mathbb{P}(X_1 = s_1 \,|\, Z)$. Second, if $Z_1 \neq s_1$, then $\mathbb{P}(X_1 = s_1 \,|\, Z) = 0$ while $z(s_1) = 0$ gives $\pi_1(s) = 0$, and hence $\pi_1(s) = 0 = \mathbb{P}(X_1 = s_1 \,|\, Z)$ in this case, too. Thus, Theorem 1 holds for $k = 1$.

Assume that (13) holds for $j \in 1, \ldots, k-1$ with $k - 1 < n$. We show that it holds for k, too. If $k \in C$ is a chance node, $\{j \,|\, j \in D, j \leq k\} = \{j \,|\, j \in D, j \leq k-1\}$ and

$$\begin{aligned}
\mathbb{P}(s_{1:k} \,|\, Z) &= \Big(\prod_{\substack{i \in C \\ i \leq k}} \mathbb{P}\big(X_i = s_i \,|\, X_{I(i)} = s_{I(i)}\big) \Big) \Big(\prod_{\substack{j \in D \\ j \leq k}} \mathbb{I}\big(Z_j(s_{I(j)}) = s_j\big) \Big) \\
&= \mathbb{P}\big(X_k = s_k \,|\, X_{I(k)} = s_{I(k)}\big) \Big(\prod_{\substack{i \in C \\ i \leq k-1}} \mathbb{P}\big(X_i = s_i \,|\, X_{I(i)} = s_{I(i)}\big) \Big) \\
&\quad \Big(\prod_{\substack{j \in D \\ j \leq k-1}} \mathbb{I}\big(Z_j(s_{I(j)}) = s_j\big) \Big) \\
&= \mathbb{P}\big(X_k = s_k \,|\, X_{I(k)} = s_{I(k)}\big) \pi_{k-1}(s) = \pi_k(s),
\end{aligned}$$

where the last equality follows the induction hypothesis and (7). Analogously, if $k \in D$ is a decision node, then

$$\begin{aligned}
\mathbb{P}(s_{1:k} \,|\, Z) &= \mathbb{I}\big(Z_k(s_{I(k)}) = s_k\big) \Big(\prod_{\substack{i \in C \\ i \leq k-1}} \mathbb{P}\big(X_i = s_i \,|\, X_{I(i)} = s_{I(i)}\big) \Big) \\
&\quad \Big(\prod_{\substack{j \in D \\ j \leq k-1}} \mathbb{I}\big(Z_j(s_{I(j)}) = s_j\big) \Big) \\
&= z(s_k \,|\, s_{I(k)}) \pi_{k-1}(s) = \pi_k(s),
\end{aligned}$$

where the last equality follows the induction hypothesis and Eqs. (5) and (8). □

Proof of Proposition 1. Choose $\alpha \in (0, 1]$ and consider $\eta^* = \mathrm{VaR}_\alpha(Z)$ which, by (29), is well defined. Then constraints (33) – (36) are satisfied by $\rho(s), \overline{\rho}(s), \lambda(s)$ and $\overline{\lambda}(s)$, defined so that $\lambda(s) = \overline{\lambda}(s) = 1$ for paths such that $\mathcal{C}(s) < \eta^*$; $\overline{\lambda}(s) = 1$ and $\lambda(s) = 0$ for $\mathcal{C}(s) = \eta^*$; and $\lambda(s) = \overline{\lambda}(s) = 0$ for $\mathcal{C}(s) > \eta^*$. From (37)-(39) it follows that $\rho(s) = \overline{\rho}(s) = \pi(s)$ when $\mathcal{C}(s) < \eta^*$; $0 = \rho(s) = 0 \leq \overline{\rho}(s) \leq \pi(s)$ for $\mathcal{C}(s) = \eta^*$; and $\rho(s) = \overline{\rho}(s) = 0$ when $\mathcal{C}(s) > \eta^*$. By the choice of η^*, is it possible to choose variables $\overline{\rho}(s) \geq 0$ for $\mathcal{C}(s) = \eta^*$ so that (40) gives the correct tail expectation $\sum_{s \in S} \overline{\rho}(s) \mathcal{C}(s) / \alpha$ in (31). Finally, assume that there exists another

solution for some $\eta' < \eta^*$. But then (40) implies that the probability α is attained as the sum of those paths whose consequence is lower than or equal to η', violating the assumption that $\eta^* = \mathrm{VaR}_\alpha(Z)$. □

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.ejor.2021.12.013

References

Antonucci, A., de Campos, C. P., Huber, D., & Zaffalon, M. (2013). Approximating credal network inferences by linear programming. In L. C. van der Gaag (Ed.), *Symbolic and quantitative approaches to reasoning with uncertainty* (pp. 13–24). Berlin, Heidelberg: Springer.

Antonucci, A., de Campos, C. P., Huber, D., & Zaffalon, M. (2015). Approximate credal network updating by linear programming with applications to decision making. *International Journal of Approximate Reasoning, 58*, 25–38.

Antunes, C. H., Alves, M. J. a., & Clímaco, J. a. (2016). Multiobjective linear and integer programming. *EURO Advanced Tutorials on Operational Research*. Springer.

Apap, R. M., & Grossmann, I. E. (2017). Models and computational strategies for multistage stochastic programming under endogenous and exogenous uncertainties. *Computers & Chemical Engineering, 103*, 233–274.

Artzner, P., Delbaen, F., Eber, J.-M., & Heath, D. (1999). Coherent measures of risk. *Mathematical Finance, 9*, 203–228.

Bertsekas, D. (2012). *Dynamic programming and optimal control*: vol. 2 (4th). Athena Scientific.

Bielza, C., Gómez, M., & Shenoy, P. P. (2011). A review of representation issues and modelling challenges with influence diagrams. *Omega, 39*, 227–241.

Bielza, C., Müller, P., & Rios Insua, D. (1999). Decision analysis by augmented probability simulation. *Management Science, 45*(7), 995–1007.

Birge, J. R., & Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

Borgonovo, E., & Tonoli, F. (2014). Decision-network polynomials and the sensitivity of decision-support models. *European Journal of Operational Research, 239*(2), 490–503.

de Campos, C. P., & Ji, Q. (2008). Strategy selection in influence diagrams using imprecise probabilities. In *Proceedings of the twenty-fourth conference on uncertainty in artificial intelligence*. In *UAI 2008*. https://www.auai.org/uai2008/UAI_camera_ready/decampos.pdf

Colvin, M., & Maravelias, C. T. (2008). A stochastic programming approach for clinical trial planning in new drug development. *Computers & Chemical Engineering, 32*(11), 2626–2642.

Colvin, M., & Maravelias, C. T. (2009). Scheduling of testing tasks and resource planning in new product development using stochastic programming. *Computers & Chemical Engineering, 33*(5), 964–976.

Colvin, M., & Maravelias, C. T. (2010). Modeling methods and a branch and cut algorithm for pharmaceutical clinical trial planning using stochastic programming. *European Journal of Operational Research, 203*(1), 205–215.

Diehl, M., & Haimes, Y. (2004). Influence diagrams with multiple objectives and tradeoff analysis. *IEEE Transactions on Systems, Man, and Cybernetics–Part A, 34*(3), 293–304.

Díez, F. J., Luque, M., & Bermejo, I. n. (2018). Decision analysis networks. *International Journal of Approximate Reasoning, 96*, 1–17.

Dupačová, J. (2006). Optimization under exogenous and endogenous uncertainty. *University of West Bohemia in Pilsen*.

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

Ekin, T., Polson, N. G., & Soyer, R. (2017). Augmented nested sampling for stochastic programs with recourse and endogenous uncertainties. *Naval Research Logistics, 64*, 613–627.

Ekin, T., Walker, S., & Damien, P. (2020). Augmented simulation methods for discrete stochastic optimization with recourse. *Annals of Operations Research*. https://doi.org/10.1007/s10479-020-03836-w.

Fourer, R. (2017). Linear programming. *ORMS Today, 44*(3).

Goel, V., & Grossmann, I. E. (2004). A stochastic programming approach to planning of offshore gas field developments under uncertainty in reserves. *Computers & Chemical Engineering, 28*(8), 1409–1429.

Goel, V., & Grossmann, I. E. (2006). A class of stochastic programs with decision dependent uncertainty. *Mathematical Programming, 108*(2-3), 355–394.

Gupta, V., & Grossmann, I. E. (2011). Solution strategies for multistage stochastic programming with endogenous uncertainties. *Computers & Chemical Engineering, 35*(11), 2235–2247.

Gupta, V., & Grossmann, I. E. (2014). A new decomposition algorithm for multistage stochastic programs with endogenous uncertainties. *Computers & Chemical Engineering, 62*, 62–79.

Gustafsson, J., & Salo, A. (2005). Contingent portfolio programming for the management of risky projects. *Operations Research, 53*(6), 946–956.

Hammond, R. K., & Bickel, J. E. (2013). Reexamining discrete approximations to continuous distributions. *Decision Analysis, 10*(1), 6–25.

Hellemo, L., Barton, P. I., & Tomasgard, A. (2018). Decision-dependent probabilities in stochastic programs with recourse. *Computational Management Science, 15*(3–4), 369–395.

Holzmann, T., & Smith, J. C. (2018). Solving discrete multi-objective optimization problems using modified augmented weighted tchebychev scalarizations. *European Journal of Operational Research, 271*(2), 436–449.

Howard, R. A., & Matheson, J. E. (1984). Influence diagrams. In R. A. Howard, & J. E. Matheson (Eds.), *The principles and applications of decision analysis, vol. II* (pp. 719–763). Menlo Park, California: Strategic Decisions Group.

Howard, R. A., & Matheson, J. E. (2005). Influence diagrams. *Decision Analysis, 2*(3), 127–143.

Howard, R. A., & Matheson, J. E. (2006). Influence diagrams retrospective. *Decision Analysis, 2*(3), 144–147.

Howard, R. A., Matheson, J. E., Merkhofer, M. W. L., Miller, A. C., & North, D. W. (2006). Comment on influence diagram retrospective. *Decision Analysis, 3*(2), 117–119.

Hynninen, Y., Vilkkumaa, E., & Salo, A. (2020). Operationalization of utilitarian and egalitarian objectives for optimal allocation of healthcare resources. *Decision Sciences*. https://doi.org/10.1111/deci.12448.

Jorgensen, E., Kristensen, A. R., & Nilsson, D. (2014). Markov LIMID processes for representing and solving renewal problems. *Annals of Operations Research, 219*, 63–84.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. Cambridge, MA: The MIT Press.

Koller, D., & Milch, B. (2003). Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior, 45*(1), 181–221.

Lauritzen, S. L., & Nilsson, D. (2001). Representing and solving decision problems with limited information. *Management Science, 47*(9), 1235–1251.

Li, Y., & Shenoy, P. P. (2012). A framework for solving hybrid influence diagrams containing deterministic conditional distributions. *Decision Analysis, 9*(1), 55–75.

Liesiö, J., Mild, P., & Salo, A. (2007). Preference programming for robust portfolio modeling and project selection. *European Journal of Operational Research, 181*(3), 1488–1505.

Liesiö, J., Mild, P., & Salo, A. (2008). Robust portfolio modeling with incomplete cost information and project interdependencies. *European Journal of Operational Research, 190*(3), 679–695.

Liesiö, J., & Salo, A. (2012). Scenario-based portfolio selection of investment projects with incomplete probability and utility information. *European Journal of Operational Research, 217*(1), 162–172.

Liesiö, J., Salo, A., Keisler, J. M., & Morton, A. (2021). Portfolio decision analysis: Recent developments and future prospects. *European Journal of Operational Research, 293*(3), 811–825.

Mancuso, A., Compare, M., Salo, A., & Zio, E. (2019). Portfolio optimization of safety measures for the prevention of time-dependent accident scenarios. *Reliability Engineering & System Safety, 190*, 106500.

Mancuso, A., Compare, M., Salo, A., & Zio, E. (2021). Optimal prognostics and health management-driven inspection and maintenance strategies for industrial systems. *Reliability Engineering & System Safety, 210*, 107536.

Manne, A. S. (1960). Linear programming and sequential decisions. *Management Science, 6*(3), 259–267.

Mauá, D. D., & Cozman, F. G. (2016). Fast local search methods for solving limited memory influence diagrams. *International Journal of Approximate Reasoning, 68*, 1235–1251.

Mauá, D. D., & Cozman, F. G. (2020). Thirty years of credal networks: Specification, algorithms and complexity. *International Journal of Approximate Reasoning, 126*, 133–157.

McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part I convex underestimating problems. *Mathematical Programming, 10*(1), 147–175.

Müller, P., Berry, D. A., Grieve, A. P., Smith, M., & Krams, M. (2007). Simulation-based sequential Bayesian design. *Journal of Statistical Planning and Inference, 137*(10), 3140–3150.

Neil, M., Tailor, M., & Marquez, D. (2007). Inference in hybrid Bayesian networks using dynamic discretization. *Statistics and Computing, 17*, 219–233.

Olmsted, M. (1983). *On representing and solving decision problems (PhD dissertation)*. Stanford University, Stanford. CA.

Parmentier, A., Cohen, V., Leclère, V., Obozinski, G., & Salmon, J. (2020). Integer programming on the junction tree polytope for influence diagrams. *INFORMS Journal on Optimization, 2*(3), 209–228.

Pflug, G. C. (2012). *Optimization of stochastic models: The interface between simulation and optimization*: vol. 373. Springer Science & Business Media.

Powell, W. B. (2019). A unified framework for stochastic optimization. *European Journal of Operational Research, 275*(3), 795–821.

Rios Insua, D., Rios, J., & Banks, D. (2009). Adversarial risk analysis. *Journal of the American Statistical Association, 104*(486), 841–854.

Rockafellar, R. T., & Uryasev, S. (2002). Conditional value-at-Risk for general loss distributions. *Journal of Banking & Finance, 26*, 1443–1471.

Roponen, J., Ríos Insua, D., & Salo, A. (2020). Adversarial risk analysis under partial information. *European Journal of Operational Research, 287*(1), 306–316.

Rubinstein, R. Y., & Shapiro, A. (1993). *Discrete event systems: Sensitivity analysis and stochastic optimization by the score function method*. John Wiley & Sons Inc.

Salo, A., Hämäläinen, R. P., & Lahtinen, T. J. (2021). Multicriteria methods for group decision processes: an overview. In D. M. Kilgour, & C. Eden (Eds.), *Handbook of Group Decision and Negotiation* (pp. 863–891)). Cham: Springer International Publishing.

Salo, A., Keisler, J., & Morton, A. (2011). *Portfolio decision analysis: Methods for improved resource allocation*: vol. 162. Springer International Series in Operations Research & Management Science.
Shachter, R. D. (1986). Evaluating influence diagrams. *Operations Research, 34*(6), 871–882.
Shachter, R. D. (1988). Probabilistic inference and influence diagrams. *Operations Research, 36*(4), 589–604.
Smith, J. E., Holtzman, S., & Matheson, J. E. (1993). Structuring conditional relationships in influence diagrams. *Operations Research, 41*(2), 280–297.
Solak, S., Clarke, J.-P. B., Johnson, E. L., & Barnes, E. R. (2010). Optimization of R&D project portfolios under endogenous uncertainty. *European Journal of Operational Research, 207*(1), 420–433.
Tatman, J. A., & Shachter, R. D. (1990). Dynamic programming and influence diagrams. *IEEE Transactions on Systems, Man, and Cybernetics, 30*(2), 365–379.
Vilkkumaa, E., Liesiö, J., & Salo, A. (2018). Scenario-based portfolio model for building robust and proactive strategies. *European Journal of Operational Research, 266*(1), 205–220.
Yet, B., Neil, M., Fenton, N., Constantinou, A., & Dementiev, E. (2018). An improved method for solving hybrid influence diagrams. *International Journal of Approximate Reasoning, 95*, 93–112.
Yuan, C., Wu, X., & Hansen, E. A. (2010). Solving multistage influence diagrams using branch-and-bound search. In *Proceedings of the twenty-sixth conference on uncertainty in artificial intelligence*. In *UAI-10* (pp. 670–691). Arlington, VA: AUAI Press.
Zhang, N. L. (1994). *A computational theory of decision networks*. PhD Thesis, Department of Computer Science, University of British Columbia, Vancouver, Canada.
Zhang, N. L., Qi, R., & Poole, D. (1994). A computational theory of decision networks. *International Journal of Approximate Reasoning, 11*, 83–158.