Operations Research Letters 61 (2025) 107308

Contents lists available at ScienceDirect

Operations Research Letters

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

Risk-averse decision strategies for influence diagrams using rooted junction trees

Olli Herrala, Topias Terho, Fabricio Oliveira,*

Department of Mathematics and Systems Analysis, Aalto University, School of Science, FI-00076 Aalto, Finland

ARTICLE INFO

Keywords:
Influence diagram
Mixed-integer programming
Risk-aversion

ABSTRACT

This paper presents how a mixed-integer programming (MIP) formulation for influence diagrams that is based on their gradual rooted junction tree representation can be extended to incorporate more general modelling features, such as risk considerations and problem-specific constraints. We propose two algorithms that enable our reformulations by performing targeted modifications either to the underlying influence diagram or to the associated gradual rooted junction tree representation. We present computational experiments highlighting the superior computational performance of our reformulation against an alternative state-of-the-art MIP formulation for influence diagrams that, by default, can accommodate those modelling features.

1. Introduction

An influence diagram (ID) [11] is an intuitive structural representation of a decision problem with uncertainties and interdependencies between random events, decisions and consequences. Traditional solution methods for IDs [24] often require strong assumptions such as the no-forgetting assumption. Lauritzen and Nilsson [16] present the notion of a limited memory influence diagram (LIMID) that, albeit more general in terms of representation capabilities, does not satisfy the no-forgetting assumption and, therefore, is not amenable to these traditional methods.

The algorithms presented in the literature for solving decision problems represented as IDs are mostly suited only to problems where an expected utility function is maximized and risk is not explicitly constrained. Thus, often risk considerations are encoded in the utility function itself, by making it concave using, e.g., utility extraction techniques [4,8,19]. Utility functions often represent monetary values, such as costs or revenues. In that case, maximizing expected utility assumes that the decision-maker has a risk-neutral stance. However, decision-makers may still have different risk tolerance profiles, which must be represented in the decision process.

There are numerous ways to incorporate risk aversion into decision models without requiring utility extraction techniques. A typical method is to minimize a risk measure instead of expected utility [18]. A commonly used measure is the *conditional value-at-risk* (CVaR), which measures the expected value in the α-tail beyond the value-at-risk VaR_α, with α being a probability threshold parameter [21]. Another typical way of incorporating risk aversion is to use constraints such as those related to chance events or budget violations [2]. Both mentioned methods have been used widely in various applications (see, e.g., [6,13,26]). Directly optimizing a risk measure within an ID is challenging because, unlike expected utility, it prevents the use of methods that construct the optimal strategy by computing locally optimal strategies at individual decision nodes.

Recently, two different mixed-integer programming (MIP) reformulations for IDs have emerged, likely stemming from the considerable computational improvements in MIP solution methods. The reformulation considered in this paper is originally presented by Parmentier et al. [20], where the authors first show how to convert a LIMID representing an expected utility maximization problem into a gradual rooted junction tree. This junction tree consists of clusters of nodes from the LIMID and is reformulated as a MIP problem using marginal probability distributions of nodes within each cluster. However, Parmentier et al. [20] only consider expected utility maximization and do not show how risk can be accounted for in their formulation.

In contrast, Salo et al. [22] present decision programming, which reformulates a LIMID as a mixed-integer linear programming (MILP) formulation without the intermediate clustering step of forming a junction tree. The decision programming formulation used in this paper is the one presented in [10], which improves that originally proposed in [22] by means of valid inequalities and reformulations.

In the context of MIP formulations for influence diagrams, the main advantage of decision programming is that its formulation can be

* Corresponding author.
E-mail address: fabricio.oliveira@aalto.fi (F. Oliveira).

https://doi.org/10.1016/j.orl.2025.107308
Received 21 December 2023; Received in revised form 15 April 2025; Accepted 15 May 2025
Available online 19 May 2025

adapted to minimize risk measures, including CVaR. Similarly, chance, logical, and budget constraints can easily be incorporated into their MILP formulation, as discussed by Hankimaa et al. [10]. However, flexibility comes at a cost of computational efficiency compared to the rooted junction tree models in [20], which is demonstrated in our computational experiments.

Against this backdrop, this paper presents how risk measures such as CVaR and constraints such as the chance, budget, and logical constraints in [22] can be incorporated into the rooted junction tree model, which significantly reduces computational time compared to solving risk-averse decision strategies with decision programming. To enable our main contribution, the rooted junction tree from which the MIP is generated needs to have a specific structure. We present how rooted junction trees can be modified to achieve said structure. We also show that this can be achieved indirectly by modifying the underlying ID, which we see as more convenient from a user's standpoint.

In Section 2, we present background on (LIM)IDs and the MIP reformulations of such diagrams. Section 3 continues with extending the rooted junction tree-based reformulation to consider the aforementioned risk measures and constraints. Section 4 presents computational results. Finally, Section 5 concludes the paper with ideas on future research directions and the potential of reformulating IDs as MIP problems.

2. Background

2.1. Pig farm problem

The pig farm problem [16] is a classical example of an influence diagram and is used throughout this paper as a running example to illustrate the proposed developments. Readers should note that the pig farm problem can alternatively be cast as a partially observable Markov decision process (POMDP). Cohen and Parmentier [3] further discuss the modelling of POMDPs using the methodology from Parmentier et al. [20]. In contrast, our focus lies on the more general framework of influence diagrams.

In the pig farm problem [16], a farmer is raising pigs for a period of four months after which the pigs will be sold. During the breeding period, a pig may develop a disease, which negatively affects the retail price of the pig at the time they are sold. In the original formulation, a healthy pig commands a price of 1000 DKK and an ill pig commands a price of 300 DKK. During the first three months, a veterinarian visits the farm and tests the pigs for the disease. The specificity (or true negative rate) of the test is 80%, whereas the sensitivity (true positive rate) is 90%. Based on the test results, the farmer may decide to inject a medicine, which costs 100 DKK. The medicine cures an ill pig with a probability of 0.5, whereas an ill pig that is not treated is spontaneously cured with a probability of 0.1. If the medicine is given to a healthy pig, the probability of developing the disease in the subsequent month is 0.1, whereas the probability without the injection is 0.2. In the first month, a pig has the disease with a probability of 0.1.

2.2. Influence diagrams

An influence diagram is a directed acyclic graph $G = (N, A)$, where N is the set of nodes and A is the set of arcs. Let $N = N^C \cup N^D \cup N^V$ be the set of chance nodes N^C, decision nodes N^D, and value nodes N^V in the ID. Let $I(j)$, $\forall j \in N$, denote the information set (or parents) of j, i.e., nodes from which there is an arc to j. It is typical to assume that value nodes are not parents of other nodes. IDs can intuitively represent complex decision problems with multiple periods, each containing multiple (possibly interdependent) decisions and chance nodes. For a selection of examples, see [10,16,20].

Each node $j \in N$ has a discrete and finite state space S_j representing possible outcomes $s_j \in S_j$. Typically, state spaces can have any discrete number of alternatives, as evidenced in the examples in [10,11,20]. For a subset of nodes $C \subseteq N$, the state space and the realized outcome are defined as $S_C := \bigtimes_{c \in C} S_c$ and $s_C = (s_c)_{c \in C}$, respectively. The outcome (i.e., state) s_j of a stochastic node $j \in N^C \cup N^V$ is a random variable with a probability distribution $\mathbb{P}(s_j \mid s_{I(j)})$, which corresponds to the probability of node j being in state s_j given that the parents $I(j)$ are in state $s_{I(j)}$. We denote $\mathbb{P}_G(s_j \mid s_{I(j)})$ when we want to emphasize that the probability distribution is associated with diagram G. The outcome of a decision node $d \in N^D$ is determined by a *decision strategy* $\delta(s_d \mid s_{I(d)}) : S_{I(d)} \bigtimes S_d \to \{0, 1\}$, where $\delta(s_d \mid s_{I(d)}) = 1$ means that state s_d is selected if nodes $I(d)$ are in states $s_{I(d)}$. A *feasible decision strategy* is such that for each $s_{I(d)} \in S_{I(d)}$, exactly one element $s_d \in S_d$ attains $\delta(s_d \mid s_{I(d)}) = 1$ and all other $s'_d \in S_d \setminus \{s_d\}$ attain $\delta(s'_d \mid s_{I(d)}) = 0$. States $s_v \in S_v$ of a value node $v \in N^V$ represent different outcomes that have a utility value $u(s_v)$ associated with them. The total utility is calculated as a sum over the different value nodes $\sum_{v \in N^v} u(s_v)$.

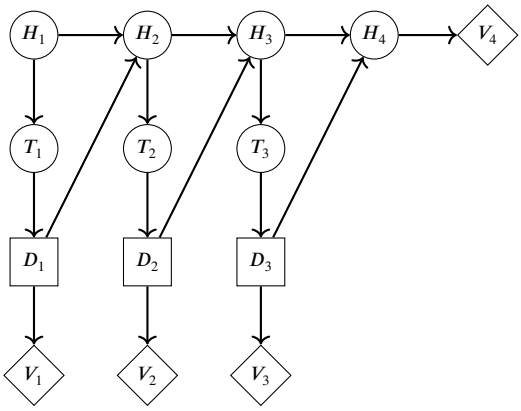

Fig. 1. ID of the pig farm problem [16].

The ID of the pig farm problem is presented in Fig. 1. In the diagram, nodes H_i are chance nodes representing the health status of the pig; chance nodes T_i represent the test result, which is conditional on the health status of the pig; decision nodes D_i represent treatment decisions; finally, value nodes V_i, for $i \leq 3$, represent treatment costs and the value node V_4 represents the pig's market price.

The solution of an ID is a decision strategy that optimizes the desired metric, typically expected utility, at the value nodes. A common additional assumption is perfect recall, meaning that previous decisions can be recalled in later stages. Under this assumption, the optimal decision strategy may be obtained by arc reversals and node removals [23] or dynamic programming [25], for example.

Perfect recall is a rather strict assumption and in many applications, it does not hold. This challenge is circumvented with LIMIDs [16]. Many algorithms for finding the decision strategy that maximizes the expected utility have been developed, such as the single policy update [16], multiple policy update [17], branch-and-bound search [12] and the aforementioned methods converting the ID to a MI(L)P [20,22].

2.3. Rooted junction trees

To achieve a MIP formulation for the decision problem, its ID, represented by the graph $G = (N, A)$, must first be transformed into a directed tree called a *gradual rooted junction tree* (RJT) $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ composed of *clusters* $C \in \mathcal{V}$, which are subsets of the nodes of the ID (i.e., $C \subseteq N$) and directed arcs $\mathcal{A}$ connecting the clusters so that each cluster only has one parent. In an ID, the set of nodes N consists of individual chance events, decisions and consequences, while the clusters in $\mathcal{V}$ comprise multiple nodes, hence the notational distinction between N and $\mathcal{V}$. The clusters are associated with a root node $j \in N$ and we refer to clusters based on the root node as the *root cluster* $C_j \in \mathcal{V}$ of node $j \in N$. Starting from an ID, we create an RJT guided by Definition 2.1, which states its necessary properties.

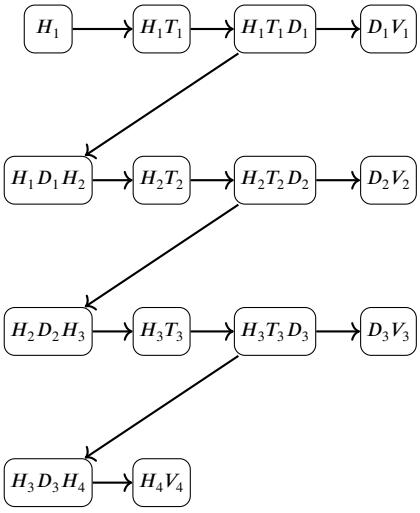

Fig. 2. Gradual RJT of the pig farm problem.

Definition 2.1. A directed rooted tree $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ consisting of clusters $C_j \in \mathcal{V}$ of nodes $j \in N$ is a gradual rooted junction tree corresponding to the influence diagram G if

(a) given two clusters C_i and C_j in the junction tree, any cluster C_k on the unique undirected path between C_i and C_j satisfies $C_i \cap C_j \subseteq C_k$;
(b) each cluster $C_j \in \mathcal{V}$ is the root cluster of exactly one node $j \in N$ (that is, the root of the subgraph induced by the clusters with node j) and all nodes $j \in N$ appear in at least one of the clusters;
(c) and, for each cluster, $I(j) \in C_j$.

A rooted tree satisfying part (a) in Definition 2.1 is said to satisfy the *running intersection property*. As a consequence of property (a), a subgraph induced by the clusters containing node j is connected. Moreover, as a consequence of property (b), we see that an RJT has as many clusters as the original influence diagram has nodes, and all nodes in the influence diagram are root nodes of exactly one cluster. Another consequence is that a cluster can contain only one node that is not contained in its parent cluster. Property (c) ensures that the root cluster contains all relevant nodes to evaluate $\delta(s_j \mid s_{I(j)})$ if j is a decision node, or $\mathbb{P}(s_j \mid s_{I(j)})$ if j is a chance node or a value node.

The RJT is created by a function $f : (N, A) \to (\mathcal{V}, \mathcal{A})$. Any function that creates an RJT satisfying the properties in Definition 2.1 can be used to derive the MIP formulation. In [20], the authors present two alternatives for f. The first function uses a given topological ordering of the nodes and builds the RJT starting from the root cluster of the last node in the topological ordering and proceeding in the reverse direction of this topological ordering. This function returns an RJT with minimum treewidth given the ordering of nodes. The second function has an additional step of finding a "good" topological ordering that results in an RJT with minimum treewidth. For simplicity, we chose to use the function requiring a topological ordering. Using $H_1, T_1, D_1, V_1, H_2, ..., H_4, V_4$ as a topological ordering, the pig farm ID in Fig. 1 is transformed to the gradual RJT in Fig. 2.

Formulating an optimization model based on the RJT representation starts by introducing a vector of moments μ_{C_j} for each root cluster C_j, $\forall j \in N$. Parmentier et al. [20] show that for RJTs, we can impose constraints so that these become moments of a distribution μ_N that factorizes according to $G(N, A)$. The joint distribution $\mathbb{P}$ is said to factorize [14] according to G if

$$\mathbb{P}(s_N) = \prod_{j \in N} \mathbb{P}(s_j \mid s_{I(j)}). \tag{1}$$

In the formulation, $\mu_{C_j}(s_{C_j})$ represents the probability of the nodes within the cluster C_j being in states s_{C_j} and part (c) of Definition 2.1 ensures that $\mathbb{P}(s_j \mid s_{I(j)})$ can thus be obtained from $\mu_{C_j}(s_{C_j})$ for each $j \in N$. The resulting MIP model is

$$\max \sum_{v \in N^V} \sum_{s_{C_v} \in S_{C_v}} \mu_{C_v}(s_{C_v}) u(s_v) \tag{2}$$

$$\text{s.t.} \sum_{s_{C_j} \in S_{C_j}} \mu_{C_j}(s_{C_j}) = 1, \ \forall j \in N \tag{3}$$

$$\sum_{\substack{\{s_{C_i} \in S_{C_i} \mid \\ s_{C_i \cap C_j} = s'_{C_i \cap C_j}\}}} \mu_{C_i}(s_{C_i}) = \sum_{\substack{\{s_{C_j} \in S_{C_j} \mid \\ s_{C_i \cap C_j} = s'_{C_i \cap C_j}\}}} \mu_{C_j}(s_{C_j}),$$

$$\forall (C_i, C_j) \in \mathcal{A}, s'_{C_i \cap C_j} \in S_{C_i \cap C_j} \tag{4}$$

$$\mu_{C_j}(s_{C_j}) = \mu_{\overline{C}_j}(s_{\overline{C}_j}) \mathbb{P}(s_j \mid s_{I(j)}), \ \forall j \in N^C \cup N^V, s_{C_j} \in S_{C_j} \tag{5}$$

$$\mu_{C_j}(s_{C_j}) = \mu_{\overline{C}_j}(s_{\overline{C}_j}) \delta(s_j \mid s_{I(j)}), \ \forall j \in N^D, s_{C_j} \in S_{C_j} \tag{6}$$

$$\mu_{C_j}(s_{C_j}) \geq 0, \ \forall j \in N, s_{C_j} \in S_{C_j} \tag{7}$$

$$\delta(s_j \mid s_{I(j)}) \in \{0, 1\}, \ \forall j \in N^D, s_j \in S_j, s_{I(j)} \in S_{I(j)}. \tag{8}$$

The formulation (2)-(8) is an expected utility maximization problem where the decision variables in the model are δ and μ and parameters are u and $\mathbb{P}$. In the objective function (2), s_v is extracted from s_{C_v} to evaluate the utility of each state combination of nodes in C_v. For notational brevity, we use $\overline{C}_j = C_j \setminus \{j\}$ to represent cluster C_j without the root node j in constraints (5) and (6) and $\mu_{\overline{C}_j}(s_{\overline{C}_j}) = \sum_{s_j \in S_j} \mu_{C_j}(s_{C_j})$ to represent the marginal distribution for cluster C_j with the node j marginalized out in constraints (5) and (6).

Combined, constraints (3) and (7) state that the variables μ_{C_j} must represent valid probability distributions, with nonnegative probabilities summing to one. Constraint (4) enforces local consistency between adjacent clusters, meaning that for a pair C_i, C_j of clusters connected by an edge, the marginal distribution for the nodes in both C_i and C_j (that is, $C_i \cap C_j$) must be the same when obtained from either C_i or C_j. For example, for the RJT in Fig. 2, constraint (4) enforces that the joint probability distribution for H_3 and D_3 is the same when evaluated from clusters $H_3T_3D_3$ and $H_3D_3H_4$.

Constraint (6) enforces that $\mu_{C_j}(s_{C_j})$ either take value 0 or $\mu_{\overline{C}_j}(s_{\overline{C}_j})$, depending on the decided strategy. Constraint (5) enforces that $\mu_{C_j}(s_{C_j})$ follows the defined conditional probability distribution for s_j. For a more extensive explanation of the RJT model formulation, see [20].

It should be noted that constraint (6) involves a product of two variables, and is thus not linear. Since we are limiting ourselves to settings with deterministic strategies (i.e., $\delta(s_d \mid s_{I(d)}) : S_{I(d)} \bigtimes S_d \to \{0, 1\}$), these constraints become indicator constraints and can be efficiently handled by solvers such as Gurobi [9]. We note that this would not be the case for more general strategies of the form $\delta(s_d \mid s_{I(d)}) : S_{I(d)} \bigtimes S_d \to [0, 1]$.

The number of constraints (4)-(6) grows exponentially with respect to the number of nodes within a single cluster. This highlights the need to find a gradual RJT representation where the clusters are as small as possible (i.e., with minimal treewidth).

3. Our contributions

3.1. Extracting the utility distribution

For problems with multiple value nodes, e.g., multi-stage decision problems, the expected utility has the property that the total expected utility is the sum of expected utilities in each value node. This property can be exploited in the solution process, and for this reason, many solution methods for IDs, including the RJT approach in [20], only tackle maximum expected utility (MEU) problems.

In contrast, risk measures (such as CVaR) require that the full probability distribution of the consequences is explicitly represented in the model. However, such representations are lost when the value nodes are placed in separate clusters, as in Fig. 2, since probability distributions are only defined for each cluster separately. For example, in the pig farm problem described in Section 2.1, the joint distribution of V_1 and V_2 cannot be inferred from the probability distributions of clusters C_{V_1} and C_{V_2}, as we cannot assume the probabilities of consequences in V_1 and V_2 to be independent.

The issue can be circumvented by generating the RJT based on an alternative equivalent ID $\overline{G} = (\overline{N}, \overline{A})$ that collects all consequences under a single value node. We present Algorithm 1, which transforms an ID into a single-value-node diagram that is equivalent to the original ID in terms of how joint probabilities and utilities are calculated. First, we formalize the notion of equivalence between IDs in Definition 3.1.

Definition 3.1. We say that two IDs $G_1 = (N_1, A_1)$ and $G_2 = (N_2, A_2)$ are equivalent if

(a) G_1 and G_2 share the same chance and decision nodes, and arcs to these nodes.
(b) There exists a bijection $g : S_{N_1} \to S_{N_2}$ such that for any $\delta(s_d \mid s_{I(d)}), \forall d \in N_1^D$, the following holds: $\prod_{n\in N_1} \mathbb{P}(s_n \mid s_{I(n)}) = \prod_{n\in N_2} \mathbb{P}(\bar{s}_n \mid \bar{s}_{I(n)})$ and $\sum_{v\in N_1^v} u(s_v) = \sum_{v\in N_2^v} u(\bar{s}_v)$, where $\bar{s} = g(s)$ for each $s \in S_{N_1}$.

Part (b) in Definition 3.1 implies that for each possible state combination of value nodes from one diagram (say, G_1), an equivalent combination in the other diagram (G_2), that evaluates to the same probability and utility, must exist. For instance, if a diagram G_1 has two value nodes with two states each, an equivalent ID G_2 could have one value node with four states that correspond to all possible combinations of the two value nodes in G_1.

Algorithm 1 Single-value-node conversion.

1: **Require** $G = (N, A)$
2: **Initialize** $\overline{G} = (\overline{N} = \emptyset, \overline{A} = \emptyset)$
3: Add $N^C \cup N^D$ to $\overline{N}$
4: Set $\mathbb{P}_{\overline{G}}(s_j \mid s_{I(j)}) = \mathbb{P}_G(s_j \mid s_{I(j)}), \forall j \in N^C$
5: Add $\{(a, b) \in A \mid b \in N^C \cup N^D\}$ to $\overline{A}$
6: Create node $\overline{v}$ such that $S_{\overline{v}} = \bigtimes_{v\in N^v} S_v$, add $\overline{v} \in \overline{N}$
7: Add $(a, \overline{v})$ to A for each $a \in I(v)$ such that $v \in N^V$
8: Set $u(s_{\overline{v}}) = \sum_{v\in N^V} u(s_v)$
9: Set $\mathbb{P}(s_{\overline{v}} \mid s_{I(\overline{v})}) = \prod_{v\in N^v} \mathbb{P}(s_v \mid s_{I(v)})$
10: **Return** $\overline{G}$

Proposition 3.2. *Let $G = (N, A)$ be an ID with $|N^V| > 1$. An ID $\overline{G} = (\overline{N}, \overline{A})$ constructed using Algorithm 1 is equivalent to G (cf. Definition 3.1).*

Proof. See Appendix A. □

When applying Algorithm 1 and transforming the ID into an RJT, according to Definition 2.1, part (c), we have that $\bigcup_{v\in V} I(v) \subseteq C_{\bar{v}}$, where $\bar{v}$ represents the unique value node in the new diagram. Consequently, the marginal probability distribution $\mu_{C_{\bar{v}}}$ contains information on the joint probability distribution of the consequences $\mathbb{P}(s_{\overline{v}} \mid s_{I(\overline{v})})$ and this can be used to expose the probability distribution of the utility values. Following this approach, the modified ID of the pig farm problem is presented in Fig. 3 and the corresponding gradual RJT in Fig. 4.

However, this incurs in computationally more demanding versions of model (2)-(8). In the single-value-node version of the pig farm problem, all decision nodes D_k, $k = 1, 2, 3$, are in the information set of $\bar{V}$. It follows from the running intersection property that D_k must be contained in every cluster that is in the undirected path between C_{D_k} and $C_{\bar{v}}$. Therefore, the clusters become larger as the parents of value nodes are "carried over", instead of evaluating separable components of the utility function at different value nodes. As discussed in [20], this increases the computational complexity of the resulting model.

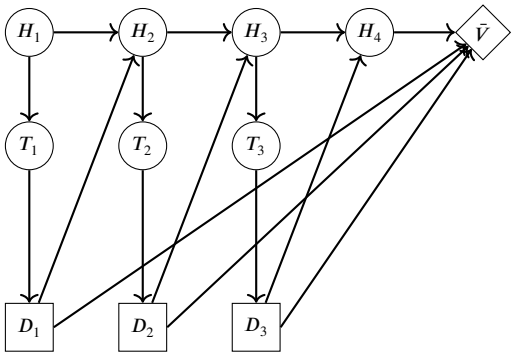

Fig. 3. The pig farm problem reformulated ID.

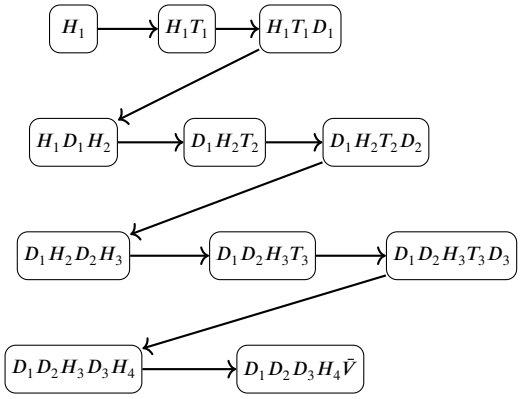

Fig. 4. Gradual RJT of the reformulated pig farm problem.

3.2. Modifying the RJT

A single-value-node ID guarantees that the full probability distribution of the consequences can be exposed in the MIP model to optimize or constrain different risk metrics. However, decision-makers may not only be interested in imposing risk constraints on consequences. For instance, in the pig farm problem, the farmer may want to impose chance constraints to ensure that pigs stay healthy throughout the breeding period with a certain likelihood. This requires that the joint probability distribution of the health states of a pig in all periods is available. That is, one needs an RJT that has a cluster containing all health status nodes $H_1, ..., H_4$, which is not found in the RJTs created with functions from [20] or with Algorithm 1 (Figs. 2 and 4).

A way to generate a cluster that exposes the desired probability distribution is to directly modify an RJT obtained following [20] while ensuring that the modified RJT fulfils Definition 2.1. Assume that there exists a topological ordering among the nodes of the RJT $\mathcal{G} = (\mathcal{V}, \mathcal{A})$. Suppose one wants to create a cluster that exposes the joint probability distribution of nodes $M \subseteq N$. Then, one needs to generate an RJT $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \bar{\mathcal{A}})$ such that $\exists C'_v \in \bar{\mathcal{V}}$ such that $M \subseteq C'_v$. Let $\preccurlyeq$ represent a topological order of the nodes N and $\max_{\preccurlyeq} M$ return the node with the highest topological order in set M. Let

$$P(C_1, C_k) := \{C_j \in \mathcal{V} \mid \exists (C_1, ..., C_j, ..., C_k) \text{ with } (C_l, C_{l+1}) \in \mathcal{A}, \forall l \in \{1, \dots, k\}\}$$

be the set of clusters that are contained on any directed path between clusters C_1 and C_k (including themselves). Let $F(j) := \{k \in N \mid P(C_j, C_k) \neq \emptyset\}$ be the set of nodes, whose root cluster can be reached

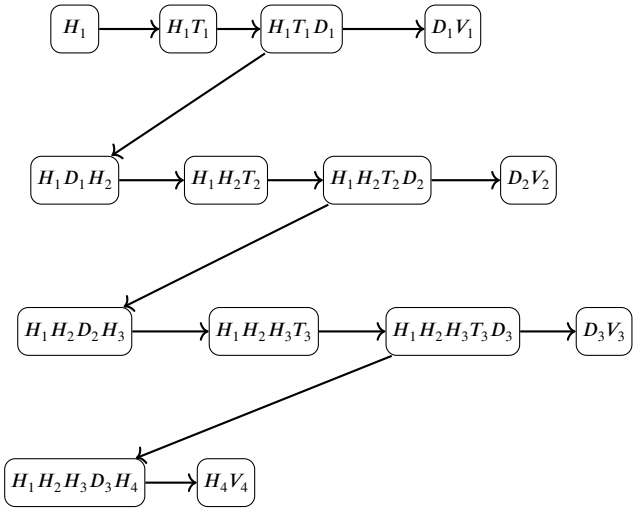

Fig. 5. Gradual RJT for pig farm problem with a cluster that contains nodes $H_1, ..., H_4$.

via a directed path starting from cluster C_j. Then, Algorithm 2 leads to an RJT with the desired structure, as stated in Proposition 3.3.

Algorithm 2 RJT modification.

Require $M \subseteq N$, topological order $\preccurlyeq$
Initialize $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \bar{\mathcal{A}})$ equal to $\mathcal{G}$ and denote C'_n as the root cluster of $n \in N$ in $\bar{\mathcal{G}}$
Find $m = \max_{\preccurlyeq} M$
For $n \in M \setminus \{m\}$ such that $n \notin C_m$ **do:**
  If $m \notin F(n)$
    Find $e = max_{\preccurlyeq}\{j \in N \mid n, m \in F(j)\}$
    Find $g \in N$ such that $(C_e, C_g) \in \mathcal{A}, m \in F(g), n \notin F(g)$
    Set $C'_e \cap C'_g \in C'_c, \forall C'_c \in P(C_e, C_n)$
    Set $(C'_e, C'_g) \notin \bar{\mathcal{A}}$, $(C'_n, C'_g) \in \bar{\mathcal{A}}$
  Set $n \in C'_a, \forall C'_a \in P(C'_n, C'_m)$
Return $\bar{\mathcal{G}}$

Proposition 3.3. *Assume that an RJT $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ satisfies Definition 2.1. The RJT $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \bar{\mathcal{A}})$ generated from $\mathcal{G}$ using Algorithm 2 satisfies Definition 2.1.*

Proof. See Appendix A. □

As an example, we can apply Algorithm 2 to the RJT in Fig. 2, which is created by a function given in [20], to include a cluster that contains nodes $H_1, ..., H_4$. The output of Algorithm 2 is the diagram presented in Fig. 5. The cluster that exposes the desired joint probability distribution for nodes $H_1, ..., H_4$ is the root cluster of H_4. A more detailed example of applying Algorithm 2 can be found in Appendix A.

3.3. Imposing chance, logical, and budget constraints

Our proposed developments allow one to expose the joint probability distribution of any combination of nodes in the ID, which in turn, enables the formulation of a broad range of risk-aversion-related constraints.

Chance constraints for the joint probability of nodes $M \subseteq N$ can be imposed on the marginal probability distribution of a cluster C_n such that $M \subseteq C_n$. If no such cluster exists in the generated RJT, a suitable cluster can be created with Algorithm 2, or regenerating the RJT based on an ID created with Algorithm 1 if M only contains value nodes and their parents. Then, chance constraints can be imposed as follows:

$$\sum_{s_{C_n} \in S_{C_n} | s_n \in S_n^o} \mu(s_{C_n}) \leq p, \tag{9}$$

where S_n^o is the set of outcomes that the decision maker wishes to constrain and $p \in [0, 1]$ represents a threshold. For instance, assume that a decision-maker wishes to enforce that the probability of the payout of the process being less than some fixed limit b is at most p. Then, a suitable RJT can be generated based on an ID created with Algorithm 1. Constraints can then be enforced for the root cluster of the single value node $C_{\bar{v}}$ and $S_{\bar{v}}^o$ would contain all states $s_{\bar{v}}$ such that $u(s_{\bar{v}}) < b$. Note that this formulation can be enforced for any cluster C_k such that $M \subseteq C_k$.

Logical constraints can be seen as a special case of chance constraints. For example, in the pig farm problem (in Section 2), the farmer may wish to attain an optimal decision strategy while ensuring that the number of injections is at most two per pig due to, e.g., potential side effects. Then, $S_{\bar{v}}^o$ would contain all realizations of the nodes in $C_{\bar{v}}$ that would lead to a violation of the constraint, i.e., the state combinations in which three injections would be given to a pig. In that case, constraint (10) that makes these scenarios infeasible could be imposed.

$$\sum_{s_{C_{\bar{v}}} \in S_{C_{\bar{v}}} | s_{\bar{v}} \in S_{\bar{v}}^o} \mu(s_{C_{\bar{v}}}) \leq 0. \tag{10}$$

Budget constraints are analogous to logical constraints, as the farmer could instead have an injection budget, say 200 DKK per pig. Then, $S_{\bar{v}}^o$ should contain all states $s_{\bar{v}}$, where more than 200 DKK is used for treating a pig, with the constraint enforced similarly as in (10).

3.4. Conditional value-at-risk (CVaR)

In addition to a number of risk constraints, the proposed reformulations also enable the consideration of alternative risk measures. Next, we focus our presentation on how to maximize CVaR, given its widespread adoption in the context of decision making under uncertainty. However, we highlight that other risk metrics, such as absolute or lower semi-absolute deviation [22], or the entropic risk measure [7] can, in principle, be used.

The proposed formulation for CVaR maximization is analogous to the method developed for decision programming in [22]. It assumes that the joint probability distribution of utility values is available, and hence, an RJT generated based on the single-value-node representation of the ID is sufficient for generating a suitable cluster. Let us assume that the decision problem has a single value node $\bar{v}$ with possible utility values $u \in U$. Let $p(u)$ be the probability of attaining utility value u. In the presence of a single value node, we would define $p(u) = \sum_{s_{C_{\bar{v}}} \in S_{C_{\bar{v}}} | U(s_{C_{\bar{v}}}) = u} \mu(s_{C_{\bar{v}}})$ and pose the constraints

$$\eta - u \leq M\lambda(u), \qquad \forall u \in U \tag{11}$$

$$\eta - u \geq (M + \epsilon)\lambda(u) - M, \qquad \forall u \in U \tag{12}$$

$$\eta - u \leq (M + \epsilon)\overline{\lambda}(u) - \epsilon, \qquad \forall u \in U \tag{13}$$

$$\eta - u \geq M(\overline{\lambda}(u) - 1), \qquad \forall u \in U \tag{14}$$

$$\overline{\rho}(u) \leq \overline{\lambda}(u), \qquad \forall u \in U \tag{15}$$

$$p(u) - (1 - \lambda(u)) \leq \rho(u) \leq \lambda(u), \qquad \forall u \in U \tag{16}$$

$$\rho(u) \leq \overline{\rho}(u) \leq p(u), \qquad \forall u \in U \tag{17}$$

$$\sum_{u \in U} \overline{\rho}(u) = \alpha \tag{18}$$

$$\lambda(u), \overline{\lambda}(u) \in \{0, 1\}, \qquad \forall u \in U \tag{19}$$

$$\rho(u), \overline{\rho}(u) \in [0, 1], \qquad \forall u \in U \tag{20}$$

$$\eta \in \mathbb{R}, \tag{21}$$

where α is the probability threshold in VaR_α. Table 1 describes which values the decision variables take due to the constraints (11)-(21).

Table 1
Variables and the corresponding values that satisfy (11)-(20).

Variable	Value
η	VaR_α
$\lambda(u)$	1 if $u < \eta$
$\bar{\lambda}(u)$	0 if $u > \eta$
$\rho(u)$	0 if $\lambda(u) = 0$, $p(u)$ otherwise
$\bar{\rho}(u)$	$\begin{cases} p(u) & \text{if } u < \eta, \\ \alpha - \sum_{u \in U} p(u) & \text{if } u = \eta, \\ 0 & \text{if } u > \eta \ (\bar{\lambda}(u) = 0) \end{cases}$

In constraints (11)-(20), M is a large positive number and ϵ is a small positive number. The parameter ϵ is used to model strict inequalities, which cannot be directly used in mathematical optimization solvers. For example, $x \geq \epsilon$ is assumed to be equivalent to $x > 0$. In practice, it is enough to set ϵ strictly smaller than the minimum difference of distinct utility values. In [22], the authors use $\epsilon = \frac{1}{2} \min\{|U(s_{\overline{v}}) - U(s'_{\overline{v}})| : |U(s_{\overline{v}}) - U(s'_{\overline{v}})| > 0, s_{\overline{v}}, s'_{\overline{v}} \in S_{\overline{v}}\}$. When $\lambda(u) = 0$, constraints (11) and (12) become $-M \leq \eta - u \leq 0$, or $\eta \leq u$. When $\lambda(u) = 1$, they instead become $\epsilon \leq \eta - u \leq M$, or $\eta > u$. Constraints (13) and (14) can be examined similarly to obtain the results in Table 1.

The correct behaviour of variables $\rho(u)$ is enforced by (16) and (17). If $\lambda(u) = 0$, constraint (16) forces $\rho(u)$ to zero. If $\lambda(u) = 1$, then $\rho(u) = p(u)$. Finally, assuming η is equal to VaR_α and $\rho(u)$ equal to $p(u)$ for all $u < \eta$, the value of $\bar{\rho}(u)$ must be $\alpha - \sum_{u \in U} \rho(u)$ for $u = \eta$. It is easy to see that η must be equal to VaR_α for there to be a feasible solution for the other variables. For an equivalence proof, see Salo et al. [22, Appendix A].

By introducing constraints (11)-(20), the CVaR for a probability threshold α ($CVaR_\alpha$) can then be obtained as

$$CVaR_\alpha = \frac{1}{\alpha} \sum_{u \in U} \bar{\rho}(u) u.$$

This can be either used as in the objective function or as a part of the constraints of the problem. We also note that the described approach is very versatile in that u can be selected to be, e.g., a stage-specific utility function, thus allowing us to limit risk in specific stages of a multi-stage problem. Krokhmal et al. [15] discusses the implications of stage-wise CVaR constraints in detail.

4. Computational experiments

To assess the computational performance of the model (2)-(8), we use the pig farm problem described earlier. We compare two different versions of the pig farm problem: one with the RJT formulation and the other with the decision programming formulation from [10]. An additional computational example and an analysis of the resulting model sizes from each formulation can be found in Appendix A. All problems were solved using a single thread on an Intel E5-2680 CPU at 2.5 GHz and 16 GB of RAM, provided by the Aalto University School of Science "Science-IT" project. The models were implemented using Julia v1.10.3 [1] and JuMP v1.23.0 [5] and solved with the Gurobi solver v11.0.2 [9]. The code and data used in this section are available at www.github.com/gamma-opt/risk-averse-RJT.

4.1. Risk-averse pig farm problem

A risk-averse version of the pig farm problem, which maximizes CVaR for an 85%-confidence level (i.e., $\alpha = 0.15$ as we wish to maximize CVaR) is solved for different numbers of breeding periods. We use the RJT based on the single-value-node ID from Fig. 4 to create the optimization model and add the constraints described in Section 3.4 to represent CVaR. The solution times using our RJT-based formulation are compared to the solution times derived using the decision programming formulation from [10]. For practical reasons, we solve the same single-value-node version of the pig farm problem to compare the solutions. In practice, decision programming can maximize CVaR in IDs with any number of value nodes. However, decision programming creates the exact same MILP model regardless of the number of value nodes.

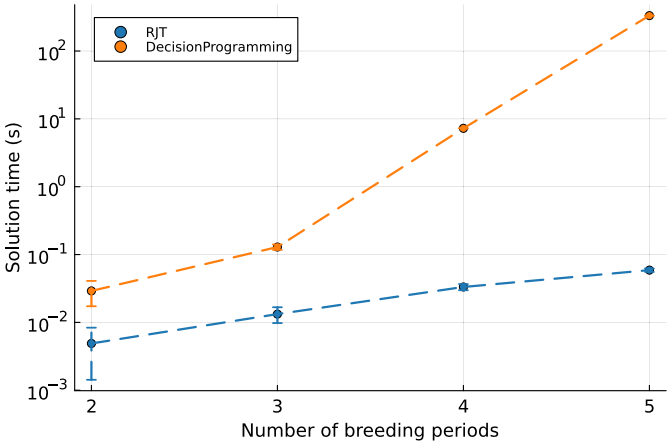

Fig. 6. Mean solution times and standard deviations (bars) for 50 random instances in the risk-averse pig farm problem with 2-5 breeding periods on a logarithmic scale.

The solution times of 50 randomly generated instances of the risk-averse pig farm problem with different sizes are presented in Fig. 6. The RJT-based formulation consistently offers better computational performance than decision programming for the pig farm problem. In larger pig farm instances, the RJT-based formulation is three orders of magnitude faster than the decision programming formulation. However, the RJT-based formulation still grows exponentially with respect to the number of breeding periods, which could result in computational challenges for larger instances. Still, this exponential growth of the RJT model can be seen as a worst-case scenario, while many problems, including the original pig farm problem (Fig. 1), exhibit treewidth independence of the number of stages. In contrast, for decision programming, as discussed in [10], the number of constraints is exponential in the number of nodes.

4.2. Chance-constrained pig farm problem

In addition, we analyze the computational performance of our formulations on a chance-constrained version of the pig farm problem with different numbers of breeding periods. We use the RJT in Fig. 5 and assign chance constraints to the root cluster of H_4 enforcing that the probability of a pig being ill at any time during the breeding period must be less than 40%. Chance constraints are enforced as described in Section 3.3. In Fig. 7, we compare the results by solving the same problem with decision programming [10].

The optimization model created based on RJT solves the chance-constrained problem faster than the corresponding decision programming model. In accordance with the results of the risk-averse pig farm problem, RJT is an order of magnitude faster than the corresponding decision programming model.

5. Conclusions

In this paper, we have described a MIP reformulation of decision problems presented as IDs, originally proposed in [20]. Our main contribution is to extend the modelling framework proposed by Parmentier et al. [20] to embed it with more general modelling capabilities. We illustrate how chance constraints and CVaR can be incorporated into the formulation. We demonstrate how suitable RJTs can be generated, either by modifying the underlying ID (Algorithm 1) or directly modifying the RJT (Algorithm 2).

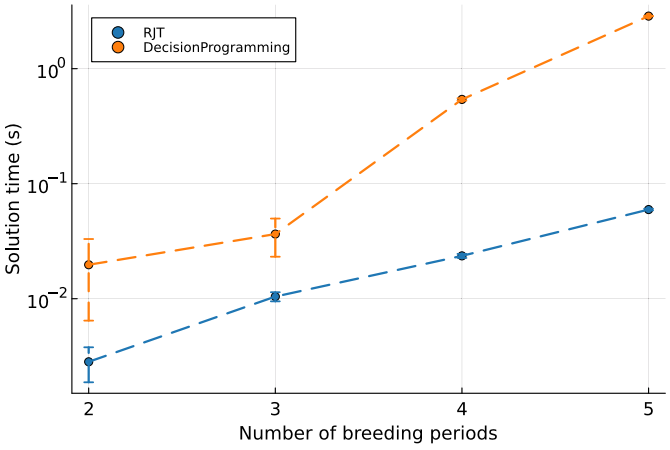

Fig. 7. Mean solution times and standard deviations (bars) for 50 random instances in the chance-constrained pig farm problem with 2-5 breeding periods on a logarithmic scale.

We show that the model in [20] can be extended beyond expected utility maximization problems to incorporate most of the constraints and objective functions present in decision programming, the alternative MILP reformulation based on LIMIDs described by Salo et al. [22] and Hankimaa et al. [10]. The advantage of using the models described in this paper is that, in terms of model size, decision programming models grow exponentially with respect to the number of nodes, whereas the RJT model grows exponentially with respect to treewidth, which is only indirectly influenced by the number of nodes.

We also present computational results comparing the computational performance of decision programming and our extension of the RJT model when applied to risk-averse and chance-constrained variants of the pig farm problem. The computational results indicate that risk-averse decision strategies for IDs can be solved considerably faster by using the RJT formulation.

Although this paper furthers the state-of-the-art for MIPs solving risk-averse IDs, typically the resulting MIP is a large-scale model, which in turn limits the size of the problems that can be solved. Hence, future research should concentrate on improving the computational tractability of the MIP model by developing specialized decomposition methods and more efficient formulations.

CRediT authorship contribution statement

Olli Herrala: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Topias Terho:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Fabricio Oliveira:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Investigation, Funding acquisition.

Acknowledgements

This work was supported by the Research Council of Finland (decision 332180). We are also thankful for the contributions from Prof. Ahti Salo and the computer resources from the Aalto University School of Science “Science-IT” project.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.orl.2025.107308.

Data availability

All data and implementations are available in a repository referenced in the paper

References

[1] J. Bezanson, A. Edelman, S. Karpinski, V.B. Shah, Julia: a fresh approach to numerical computing, SIAM Rev. 59 (2017) 65–98.
[2] A. Charnes, W. Cooper, Chance constrained programming, Manag. Sci. 6 (1959) 73–79.
[3] V. Cohen, A. Parmentier, Future memories are not needed for large classes of POMDPs, Oper. Res. Lett. 51 (2023) 270–277.
[4] D. Davidson, P. Suppes, S. Siegel, Decision Making; an Experimental Approach, Stanford University Press, Pages, 1957, p. 121.
[5] I. Dunning, J. Huchette, M. Lubin, JuMP: a modeling language for mathematical optimization, SIAM Rev. 59 (2017) 295–320.
[6] C. Filippi, G. Guastaroba, M. Speranza, Conditional value-at-risk beyond finance: a survey, Int. Trans. Oper. Res. 27 (2020) 1277–1319.
[7] H. Föllmer, T. Knispel, Entropic risk measures: coherence vs. convexity, model ambiguity and robust large deviations, Stoch. Dyn. 11 (2011) 333–351.
[8] S. Geissel, J. Sass, F.T. Seifried, Optimal expected utility risk measures, Stat. Risk. Model. 35 (2018) 73–87.
[9] Gurobi Optimization, LLC, Gurobi optimizer reference manual, https://www.gurobi.com, 2022.
[10] H. Hankimaa, O. Herrala, F. Oliveira, J. Tollander de Balsch, Solving influence diagrams via efficient mixed-integer programming formulations and heuristics, arXiv:2307.13299, 2023.
[11] R.A. Howard, J.E. Matheson, Influence diagrams, Decis. Anal. 2 (2005) 127–143.
[12] A. Khaled, E.A. Hansen, C. Yuan, Solving limited-memory influence diagrams using branch-and-bound search, in: Uncertainty in Artificial Intelligence (UAI-13), AUAI Press, Arlington, Virginia, USA, 2013, pp. 331–341.
[13] A. Khassiba, F. Bastin, S. Cafieri, B. Gendron, M. Mongeaua, Two-stage stochastic mixed-integer programming with chance constraints for extended aircraft arrival management, Transp. Sci. 54 (2020) 897–919.
[14] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009.
[15] P. Krokhmal, J. Palmquist, S. Uryasev, Portfolio optimization with conditional value-at-risk objective and constraints, J. Risk 4 (2002) 43–68.
[16] S.L. Lauritzen, D. Nilsson, Representing and solving decision problems with limited information, Manag. Sci. 47 (2001) 1235–1251.
[17] D.D. Mauà, C.P. de Campos, M. Zaffalon, Solving limited memory influence diagrams, J. Artif. Intell. Res. 44 (2012) 97–140.
[18] T. Homem-de Mello, B.K. Pagnoncelli, Risk aversion in multistage stochastic programming: a modeling and algorithmic perspective, Eur. J. Oper. Res. 249 (2016) 188–199.
[19] T.D. Nielsen, F.V. Jensen, Learning a decision maker’s utility function from (possibly) inconsistent behavior, Artif. Intell. 160 (2004) 53–78.
[20] A. Parmentier, V. Cohen, V. Leclère, G. Obozinski, J. Salmon, Integer programming on the junction tree polytope for influence diagrams, INFORMS J. Optim. 2 (2020) 209–228.
[21] R. Rockafellar, S. Uryasev, Optimization of conditional value-at-risk, J. Risk 2 (2000) 21–42.
[22] A. Salo, J. Andelmin, F. Oliveira, Decision programming for mixed-integer multistage optimization under uncertainty, Eur. J. Oper. Res. 299 (2022) 550–565.
[23] R.D. Shachter, Evaluating influence diagrams, Oper. Res. 34 (1986) 871–882.
[24] R.D. Shachter, D. Bhattacharjya, Solving influence diagrams: Exact algorithms, 2010.
[25] J.A. Tatman, R.D. Shachter, Dynamic programming and influence diagrams, IEEE Trans. Syst. Man Cybern. 30 (1990) 365–379.
[26] B. Xu, S.E. Boyce, Y. Zhang, Q. Liu, L. Guo, P.A. Zhong, Stochastic programming with a joint chance constraint model for reservoir refill operation considering flood risk, J. Water Resour. Plan. Manag. 143 (2017).