Decision Programming

Fabricio Oliveira

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

I Workshop de Otimização sob Incerteza - UFSCar

March 23, 2022

Aalto University
School of Science

Γ-opt
Group of Applied Mathematical Modelling And Optimisation

The team

Juho Andelmin

Olli Heralla

Helmi Hankimaa

Topias Terho

Tommi Ekholm

Ahti Salo

Fabricio Oliveira

Outline of this talk

Outline of this talk

Introduction

Decision Programming

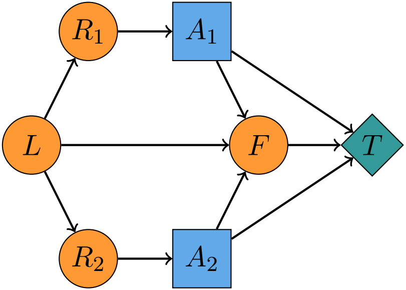
Computational experiments

Conclusions

Recent developments

Modelling decision problems

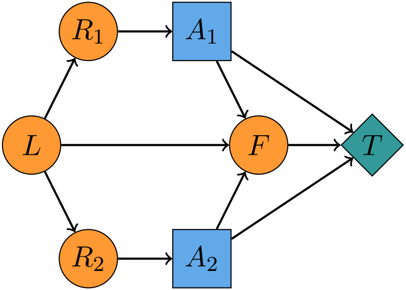
Influence diagrams are widely used to model decision problems under uncertainty.



- Circles denote chance events
- Squares denote decision events
- Diamonds denote value/utility calculation
- Arc represent influence (dependence).

Modelling decision problems

Influence diagrams are widely used to model decision problems under uncertainty.



- Circles denote chance events
- Squares denote decision events
- Diamonds denote value/utility calculation
- Arc represent influence (dependence).

A simple yet powerful tool that allows for representing a vast range of decision problems with endogenous uncertainties.

Influence diagrams

Despite its simplicity, obtaining solution strategies from influence diagrams is not trivial. Methods include:

- Form a decision tree and solve it (backward induction);
- Apply arc reversal/ node elimination methods;
- Apply Single Policy Update (SPU) (Lauritzen and Nilsson, 2001) or variant.

Influence diagrams

Despite its simplicity, obtaining solution strategies from influence diagrams is not trivial. Methods include:

- Form a decision tree and solve it (backward induction);
- Apply arc reversal/ node elimination methods;
- Apply Single Policy Update (SPU) (Lauritzen and Nilsson, 2001) or variant.

Influence diagrams represent Markov decision processes and are, likewise, generally hard to solve.

- Solving an influence diagram is NP-Hard (Mauá et al., 2013)
- Even obtaining approximate solutions is NP-Hard (Mauá et al., 2014)

Influence diagrams

Moreover, several limitations arise from relying on influence diagrams as a modelling framework:

- Solution methods require the no-forgetting assumption: assume single decision maker or perfect information sharing.

Influence diagrams

Moreover, several limitations arise from relying on influence diagrams as a modelling framework:

- Solution methods require the no-forgetting assumption: assume single decision maker or perfect information sharing.
- Imposing constraints among decisions is not possible.

Influence diagrams

Moreover, several limitations arise from relying on influence diagrams as a modelling framework:

- Solution methods require the no-forgetting assumption: assume single decision maker or perfect information sharing.
- Imposing constraints among decisions is not possible.
- Multiple value nodes (or objectives)

Influence diagrams

Moreover, several limitations arise from relying on influence diagrams as a modelling framework:

- Solution methods require the no-forgetting assumption: assume single decision maker or perfect information sharing.
- Imposing constraints among decisions is not possible.
- Multiple value nodes (or objectives)
- Considering measures on the outcome probabilistic distribution (e.g., chance constraints, risk measures) is not viable with traditional methods.

Influence diagrams

Moreover, several limitations arise from relying on influence diagrams as a modelling framework:

- Solution methods require the no-forgetting assumption: assume single decision maker or perfect information sharing.
- Imposing constraints among decisions is not possible.
- Multiple value nodes (or objectives)
- Considering measures on the outcome probabilistic distribution (e.g., chance constraints, risk measures) is not viable with traditional methods.

Our contribution: a framework to address the above while being computationally reliable.

Outline of this talk

Decision Programming

In specific, we propose a framework that can:

1. exploit the expressiveness of influence diagrams
2. exploit linearity (i.e., solve Mixed-Integer Linear Programs - MIPs) as opposed to recursion.

Decision Programming

In specific, we propose a framework that can:

1. exploit the expressiveness of influence diagrams
2. exploit linearity (i.e., solve Mixed-Integer Linear Programs - MIPs) as opposed to recursion.

In a nutshell, Decision Programming combines:

- the structuring for decision problem under uncertainty from Decision Analysis with
- the structure of MIP formulation of deterministic equivalents for multistage Stochastic Programming problems.

Decision Programming

Information sets and paths

We represent an influence diagram as an acyclic graph $G(N, A)$.

- N consists of chance nodes $c \in C$, decision nodes $d \in D$, and value nodes $v \in V$. Let $n = |C| + |D|$.
- Arcs $A = \{(i, j) : i, j \in N\}$ represent dependencies between nodes.

Decision Programming

Information sets and paths

We represent an influence diagram as an acyclic graph $G(N, A)$.

- N consists of chance nodes $c \in C$, decision nodes $d \in D$, and value nodes $v \in V$. Let $n = |C| + |D|$.
- Arcs $A = \{(i, j) : i, j \in N\}$ represent dependencies between nodes.

With these in mind, we define two key concepts:

- **Information sets:** $I(j)$ consists of nodes from which there is an arc to j.
- **Information states:** $s_{I(j)} \in S_{I(j)} = \prod_{i \in I(j)} S_i$ is a combination of states s_i for nodes in the information set of $i \in I(j)$.

Decision Programming

Information sets and paths

Considering the nodes $i \in C \cup D$.

- Let X_i be the associated 'random' variable.

Decision Programming

Information sets and paths

Considering the nodes $i \in C \cup D$.

- Let X_i be the associated 'random' variable.
- **At chance nodes** $c \in C$**:** a state s_c is observed with (conditional) probability

$$\mathbb{P}(X_c = s_c \mid X_i = s_i, i \in I(c))$$

Decision Programming

Information sets and paths

Considering the nodes $i \in C \cup D$.

- Let X_i be the associated 'random' variable.
- **At chance nodes $c \in C$:** a state s_c is observed with (conditional) probability

$$\mathbb{P}(X_c = s_c \mid X_i = s_i, i \in I(c))$$

- **At decision nodes $d \in D$:** we define a local decision strategy as a function $Z_d : S_{I(d)} \mapsto S_d$.

$$\mathbb{P}(X_d = s_d \mid X_i = s_i, i \in I(d), Z_d) = 1 \iff Z_d(s_{I(d)}) = s_d$$

Decision Programming

Information sets and paths

Considering the nodes $i \in C \cup D$.

- Let X_i be the associated 'random' variable.
- **At chance nodes $c \in C$:** a state s_c is observed with (conditional) probability

$$\mathbb{P}(X_c = s_c \mid X_i = s_i, i \in I(c))$$

- **At decision nodes $d \in D$:** we define a local decision strategy as a function $Z_d : S_{I(d)} \mapsto S_d$.

$$\mathbb{P}(X_d = s_d \mid X_i = s_i, i \in I(d), Z_d) = 1 \iff Z_d(s_{I(d)}) = s_d$$

Remark: A (global) decision strategy $Z = \prod_{d \in D} Z_d$ is the combination of all local decision strategies.

Information sets and paths

Another key concept: the notion of a path.

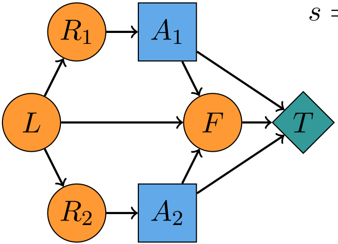
- Since G is acyclic, $i < j$ if $(i, j) \in A$ w.l.o.g.;
- A **path** of length k is a sequence $(s_1, s_2, \dots, s_k)$ such that $s_i \in S_i, i = 1, \dots, k$;
- Paths of length $n = |C| + |D|$ are denoted by

$$s = (s_1, \dots, s_n) \in S = \prod_{i \in C \cup D} S_i.$$

Information sets and paths

Another key concept: the notion of a path.

- Since G is acyclic, $i < j$ if $(i,j) \in A$ w.l.o.g.;
- A **path** of length k is a sequence $(s_1, s_2, \dots, s_k)$ such that $s_i \in S_i, i = 1, \dots, k$;
- Paths of length $n = |C| + |D|$ are denoted by



$$s = (s_1, \dots, s_n) \in S = \prod_{i \in C \cup D} S_i.$$

Example: assume that $L = R_1 = R_2 = ... = F = \{+, -\}$.

Then $s = (l, r_1, r_2, a_1, a_2, f) = (+,+,+,+,+,+)$ is a path.

Information sets and paths

We can now formally state (recursively) the probability of a path given a decision strategy Z

$$\mathbb{P}(s_{1:k} \mid Z) = \left(\prod_{i \in C : i \leq k} \mathbb{P}\big(X_i = s_i \mid X_{I(i)} = s_{I(i)}\big) \right) \left(\prod_{j \in D : j \leq k} \mathbb{I}\big(Z_j(s_{I(j)}) = s_j\big) \right),$$

where $\mathbb{I}(\,\cdot\,)$ is defined so that

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

Towards an MIP formulation

Our objective is to encode this logic into decision variables.

- We represent decisions with variables $z(s_j \mid s_{I(j)}) \in \{0, 1\}$.

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

Towards an MIP formulation

Our objective is to encode this logic into decision variables.

- We represent decisions with variables $z(s_j \mid s_{I(j)}) \in \{0, 1\}$.

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

- Mutual exclusivity implies

$$\sum_{s_j \in S_j} z(s_j \mid s_{I(j)}) = 1, \forall j \in D, s_{I(j)} \in S_{I(j)} \tag{2}$$

Towards an MIP formulation

Our objective is to encode this logic into decision variables.

- We represent decisions with variables $z(s_j \mid s_{I(j)}) \in \{0, 1\}$.

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

- Mutual exclusivity implies

$$\sum_{s_j \in S_j} z(s_j \mid s_{I(j)}) = 1, \forall j \in D, s_{I(j)} \in S_{I(j)} \tag{2}$$

- And we define $\pi_k(s) \in [0, 1]$ to represent the path probability.

$$\pi_k(s) = \mathbb{P}\left(X_k = s_k \mid X_{I(k)} = s_{I(k)}\right) \pi_{k-1}(s), \tag{3}$$

For $k \in D$ being a decision node, we have that

$$\pi_k(s) = \begin{cases} \pi_{k-1}(s), & \text{if } z(s_k \mid s_{I(k)}) = 1 \\ 0, & \text{if } z(s_k \mid s_{I(k)}) = 0. \end{cases} \tag{4}$$

Information sets and paths

Theorem 1

Let $Z \in \mathbb{Z}$ be a decision strategy and choose a path $s \in S$. If $\pi_k(s)$, $k = 1, \dots, n$, and $z(s_j \mid s_{I(j)})$, $\forall j \in D$, satisfy the constraints (1) – (4), then

$$\pi_k(s) = \mathbb{P}(X_{1:k} = s_{1:k} \mid Z), \forall\, k = 1, \dots, n$$

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

Towards an MIP formulation

Variables $\pi_k(s)$ can be defined by the inequalities

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

which are equivalent to the linear inequalities

$$\begin{aligned} \pi_k(s) &\leq \pi_{k-1}(s) && (5) \\ \pi_k(s) &\leq z(s_k \mid s_{I(k)}) && (6) \\ \pi_k(s) &\geq 0 && (7) \\ \pi_k(s) &\geq \pi_{k-1}(s) + z(s_k \mid s_{I(k)}) - 1. && (8) \end{aligned}$$

Towards a MIP formulation

We want to maximise expected utilities using $\mathcal{U} : S_{I(v)} \to \mathbb{R}$.

$$\text{max.}_{Z \in \mathbb{Z}} \sum_{s \in S} \pi_n(s) \mathcal{U}(s)$$

which only involve $\pi_n(s) = \pi(s)$.

Towards a MIP formulation

We want to maximise expected utilities using $\mathcal{U} : S_{I(v)} \to \mathbb{R}$.

$$\text{max.}_{Z \in \mathbb{Z}} \sum_{s \in S} \pi_n(s) \mathcal{U}(s)$$

which only involve $\pi_n(s) = \pi(s)$. Notice that these can be pre-calculated for any given strategy $Z \in \mathbb{Z}$.

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

And then

- if Z is compatible with $s \in S$ (i.e., if Z maps to path $s \in S$), then $\pi(s) = p(s)$
- otherwise, $\pi(s) = 0$.

Towards an MIP formulation

Corollary 2

The expected utility is maximised by the strategy $Z \in \mathbb{Z}$ which solves the optimisation problem

$$\max_{Z \in \mathbb{Z}} \sum_{s \in S} \pi(s)\mathcal{U}(s)$$

subject to constraints (1) – (3) *and* (5) – (8) *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* $\pi_k(s) \in [0,1], \forall s \in S$.

Towards an MIP formulation

Corollary 2

The expected utility is maximised by the strategy $Z \in \mathbb{Z}$ which solves the optimisation problem

$$\max_{Z \in \mathbb{Z}} \sum_{s \in S} \pi(s)\mathcal{U}(s)$$

subject to constraints (1) – (3) *and* (5) – (8) *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* $\pi_k(s) \in [0,1], \forall s \in S$*.*

The formulation recursively simplified to only consider $k = n$, since

- (5) – (8) imply that $\pi_j(s) = \pi_{j-1}(s)$ for each $j \in D$ if $z(s_j \mid s_{I(j)}) = 1$
- Analogously, if the strategy Z is not compatible with s, $\pi_n(s) \leq \pi_j(s) = 0$ if $z(s_j \mid s_{I(j)}) = 0$ for some $j \in D$.

Towards a MIP formulation

The complete formulation is given by

$$\text{max.}_{Z \in \mathbb{Z}} \quad \sum_{s \in S} \pi(s)\mathcal{U}(s)$$

s.t.:

$$\begin{aligned}
&\sum_{s_j \in S_j} z(s_j \mid s_{I(j)}) = 1, && \forall\, j \in D, s_{I(j)} \in S_{I(j)} \\
&0 \leq \pi(s) \leq p(s), && \forall\, s \in S \\
&\pi(s) \leq z(s_j \mid s_{I(j)}), && \forall\, j \in D, s \in S \\
&\pi(s) \geq p(s) + \sum_{j \in D} z(s_j \mid s_{I(j)}) - |D|, && \forall\, s \in S \\
&z(s_j \mid s_{I(j)}) \in \{0, 1\}, && \forall\, j \in D,\ s_j \in S_j,\ s_{I(j)} \in S_{I(j)}.
\end{aligned}$$

MIP formulation: key features

Some points worth highlighting:

1. Notice that utilities $\mathcal{U}(s)$ and probabilities $p(s)$ can be (efficiently) computed beforehand.

MIP formulation: key features

Some points worth highlighting:

1. Notice that utilities $\mathcal{U}(s)$ and probabilities $p(s)$ can be (efficiently) computed beforehand.
2. We tried to linearise the product of variables in

$$\pi_k(s) = \mathbb{P}\left(X_k = s_k \mid X_{I(k)} = s_{I(k)}\right) \pi_{k-1}(s),$$

   but the formulation obtained was weaker (in terms of LP relaxation).

MIP formulation: key features

Some points worth highlighting:

1. Notice that utilities $\mathcal{U}(s)$ and probabilities $p(s)$ can be (efficiently) computed beforehand.
2. We tried to linearise the product of variables in

$$\pi_k(s) = \mathbb{P}\left(X_k = s_k \mid X_{I(k)} = s_{I(k)}\right)\pi_{k-1}(s),$$

   but the formulation obtained was weaker (in terms of LP relaxation).
3. The model has exploitable structure. For example, we use (as lazy constraints) probability cuts of the form

$$\sum_{s \in S} \pi(s) = 1.$$

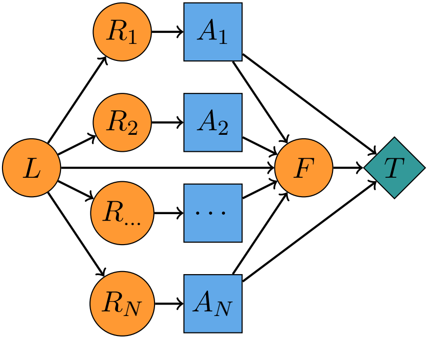
Outline of this talk

Examples

N-monitoring problem

N agents independently intervening without sharing information.

- Independent parallel measures;
- Decisions that can't be communicated;
- No no-forgetting: each action can be seen as taken by independent decision makers.

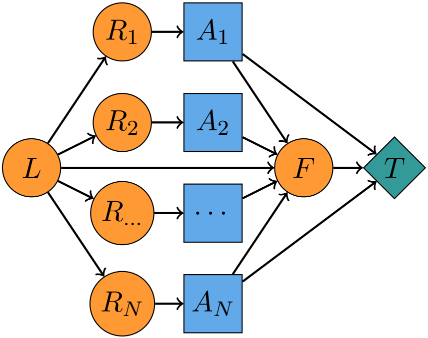


Examples

N-monitoring problem

N agents independently intervening without sharing information.

- Independent parallel measures;
- Decisions that can't be communicated;
- No no-forgetting: each action can be seen as taken by independent decision makers.



Remark: can be shown to not be soluble (Lauritzen and Nilsson, 2001), a sufficient condition for SPU to converge to optimal strategies.

Computational experiments[1]

N-monitoring problem

	Number of variables		No probability cuts		With probability cuts	
# Nodes	Binary	Real	A	SD	A	SD
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: Solution times (s) for 10 randomly generated instances.

[1] **Computational setting:** Intel Xeon E3-1230 @ 3.40 GHz with 32 GB RAM; coded in Julia 1.1.0 (JuMP 0.18.6); solved with Gurobi 8.1.0.

Examples

Pig farm problem

The original problem introducing LIMID as not soluble.

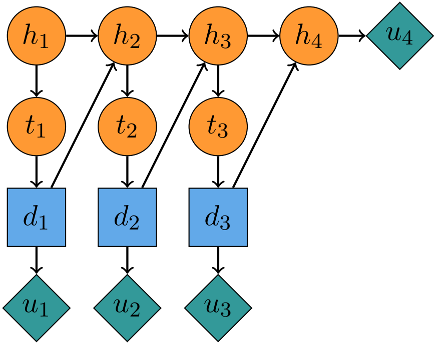


Figure: The pig farm problem with 4 periods (Lauritzen and Nilsson, 2001).

- Each month pigs are tested for a disease
- Decide whether to inject curative/preventive drug.
- Sick pigs worth less at the end.
- No record is kept for individual pigs.

Computational experiments[2]

Pig farm problem

Obtaining optimal solutions is fairly easy.

# 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

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

[2] **Computational setting:** Intel Xeon E3-1230 @ 3.40 GHz with 32 GB RAM; coded in Julia 1.1.0 (JuMP 0.18.6); solved with Gurobi 8.1.0.

Computational experiments[2]

Pig farm problem

Obtaining optimal solutions is fairly easy.

# 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

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

We extend the example incorporating risk aversion (CVaR) and calculating all non-dominated strategies, originally not possible.

[2] **Computational setting:** Intel Xeon E3-1230 @ 3.40 GHz with 32 GB RAM; coded in Julia 1.1.0 (JuMP 0.18.6); solved with Gurobi 8.1.0.

Extra: Pig farm problem with risk considerations

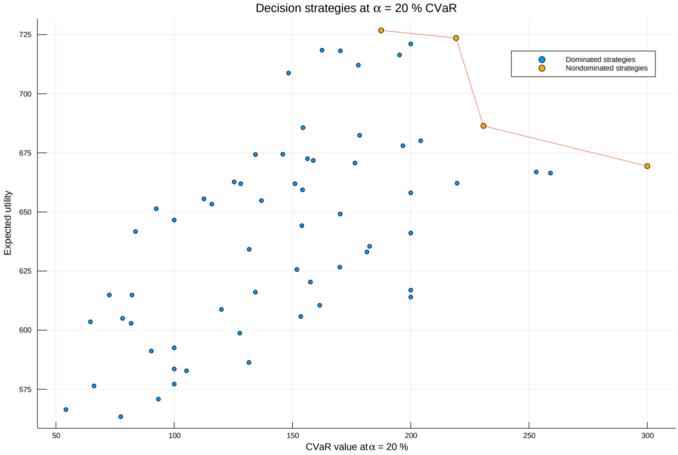


Figure: Expected utilities and conditional expectations in the lower $\alpha = 0.20$ tail for all 64 strategies of the 4-month pig problem.

Outline of this talk

Key points and takeaways

Decision Programming =

Decision Analysis + Mathematical Programming

Key points and takeaways

Decision Programming =

Decision Analysis + Mathematical Programming

- Decision Programming exploits linearity instead of recursion to solve decision diagrams.
- Pre-calculating the path probabilities $p(s)$ and utilities $\mathcal{U}$ can be done efficiently (in parallel).
- Mathematical programming as underpinning framework allows for flexibility in terms of imposing constraints.
- "Future" work: modelling endogenously uncertain problems and solution methods (preprocessing and heuristics).

To learn more:

Main reference: Salo et al. (2022), Decision programming for multi-stage optimization under uncertainty, EJOR, 299 (2), 550-565. DOI: 10.1016/j.ejor.2021.12.013

Julia package with many other examples:
github.com/gamma-opt/DecisionProgramming.jl

Some newer WiP:

Andelmin, Juho, et al. "DecisionProgramming.jl - A framework for modelling decision problems using mathematical programming." arXiv preprint arXiv:2307.13299 (2023).

Herrala, Olli, Tommi Ekholm, and Fabricio Oliveira. "A decomposition strategy for decision problems with endogenous uncertainty using mixed-integer programming." arXiv preprint arXiv:2304.02338 (2023).

Outline of this talk

More recent developments

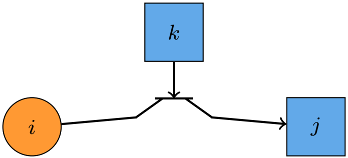
1. Modelling long-term endogenous climate uncertainty

- Consider decision-dependent (endogenous) uncertainties
- Take into account continuous decision spaces

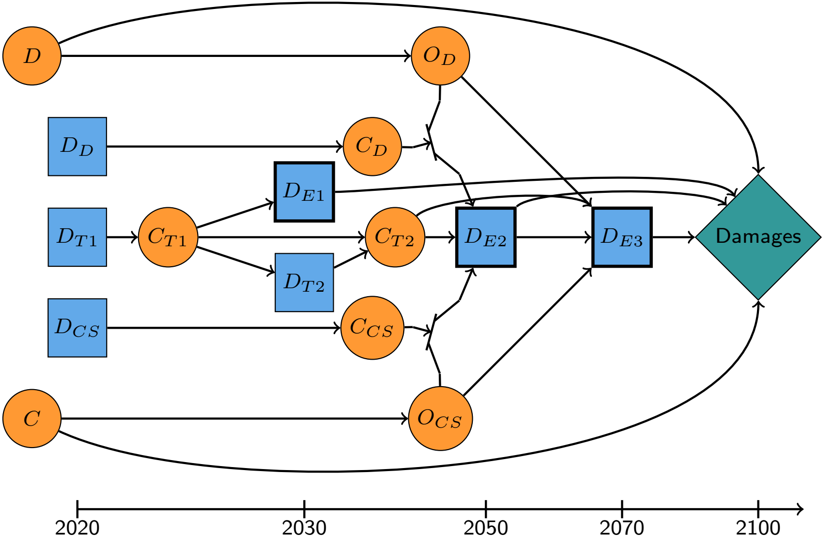
We develop the notions of extended value node

$$U_v(s_{I(v)}) := \max_y\{f_{s_{I(v)}}(y) \mid y \in Y_{s_{I(v)}}\}, \text{ for } v \in V,$$

and conditional arcs



Climate change mitigation



More recent developments

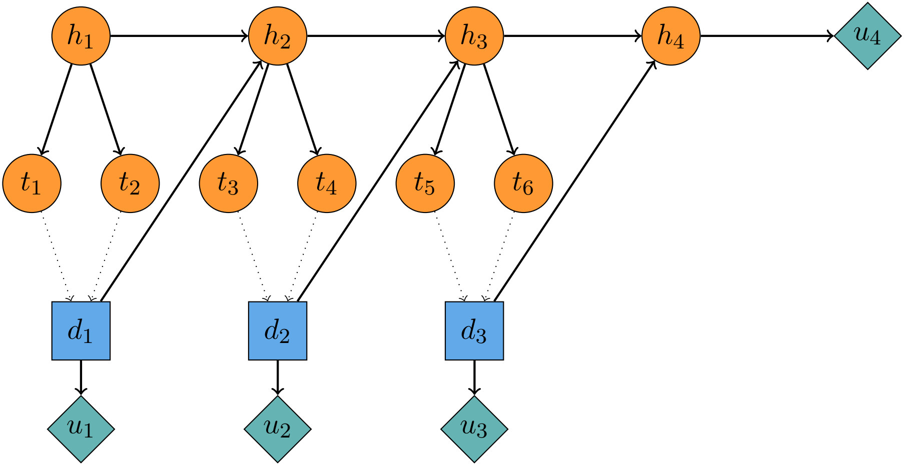
2. Optimal information structures

- We are interested in knowing what information to acquire and when
- Find optimal information structure and decision strategy

We propose three alternative formulations:

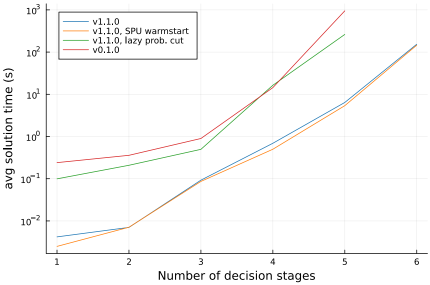
1. Constraints on path probabilities
2. Constraints on local decisions
3. Extended state space

The extended pig farm problem

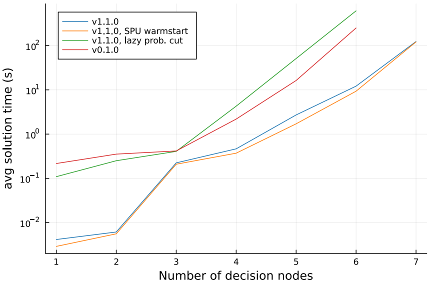


Better formulations

New formulations: stronger formulations in which we can replace indicator variables $\pi(s)$ with continuous variables.



(a) The pig farm problem



(b) The N-monitoring problem

Figure: The solution times of the two example problems with different number of decision nodes using different formulations. Notice the logarithmic y-axis.

Decision Programming

Fabricio Oliveira

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

I Workshop de Otimização sob Incerteza - UFSCar

March 23, 2022

Aalto University
School of Science

Group of Applied Mathematical Modelling And Optimisation

References I

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

Mauá, D. D., De Campos, C. P., Benavoli, A., and Antonucci, A. (2014). Probabilistic inference in credal networks: new complexity results. *Journal of Artificial Intelligence Research*, 50:603–637.

Mauá, D. D., De Campos, C. P., and Zaffalon, M. (2013). On the complexity of solving polytree-shaped limited memory influence diagrams with binary variables. *Artificial Intelligence*, 205:30–38.