Decision Programming

Fabricio Oliveira

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

ICSP 2019 - Trondheim, Norway

August 8, 2019

Aalto University
School of Science

The team

Fabricio Oliveira

Juho Andelmin

Ahti Salo

Outline of this talk

Introduction

Decision Programming

Computational experiments

Conclusions

Outline of this talk

Introduction

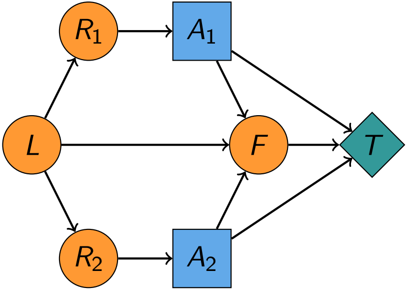
Decision Programming

Computational experiments

Conclusions

Modelling decision problems under uncertainty

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

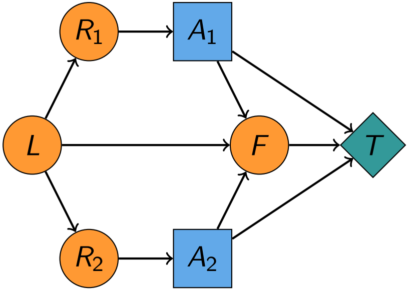


- Circles denote chance events
- Squares denote decision events
- Diamonds denote value/ utility calculation
- Arc represent influence (dependence).

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

Influence diagrams

Example: influence diagram for the 2-monitoring problem.



- L is an uncertain load;
- R_1 and R_2 are uncertain load readings;
- F is a possible failure.
- A_1 and A_2 are fortification decisions;
- T accumulates utilities (fortification/ failure);

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:

- Most methods require the perfect recall (or no-forgetting) assumption: assume single decision maker of perfect information sharing.

Influence diagrams

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

- Most methods require the perfect recall (or no-forgetting) assumption: assume single decision maker of 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:

- Most methods require the perfect recall (or no-forgetting) assumption: assume single decision maker of perfect information sharing.
- Imposing constraints among decisions is not possible.
- Considering measures on the outcome probabilistic distribution (as chance constraints, CVaR and such) is not viable with traditional methods.

Outline of this talk

Decision Programming = Decision Analysis + Stochastic Programming

With those challenges in mind, we build a framework that could:

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

Decision Programming = Decision Analysis + Stochastic Programming

With those challenges in mind, we build a framework that could:

1. exploit the expressiveness of (limited memory) 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 formulation of deterministic equivalents for multistage Stochastic Programming problems.

Information sets and paths

We start from an influence diagram, which is a acyclic graph $G(V, A)$.

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

Information sets and paths

We start from an influence diagram, which is a acyclic graph $G(V, A)$.

- V consists of chance nodes $c \in C$, decision nodes $d \in D$, and value nodes $u \in U$. Let $n = |C| + |D|$.
- Arcs $(i, j) \in A$ 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 $i \in I(j)$.

Information sets and paths

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

- Let X_i be the associated 'random' variable.

.

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

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

.

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

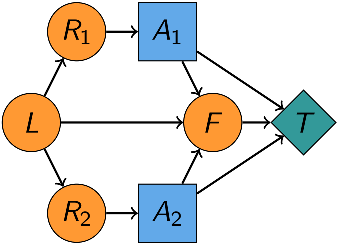
$$s = (s_1, \ldots, s_n) \in S = \prod_{i \in C \cup D} S_i.$$

Information sets and paths

Another key concept: the notion of a path.

- Since A 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 $s = (l, r_1, r_2, a_1, a_2, f) = (+, +, +, +, +, +)$ forms 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 $z(s_j \mid s_{I(j)}) \in \{0, 1\}$.

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

Towards an MIP formulation

Our objective is to encode this logic into decision variables.

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

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

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

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

Towards an MIP formulation

Our objective is to encode this logic into decision variables.

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

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

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

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

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

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

Towards a MIP formulation

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

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

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

[1] i.e., if Z maps to path $s \in S$.

Towards a MIP formulation

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

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

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[1] with $s \in S$, then $\pi(s) = p(s)$
- otherwise, $\pi(s) = 0$.

[1] i.e., if Z maps to path $s \in S$.

Towards a MIP formulation

The complete formulation is given by

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

MIP formulation: key features

Some points worth highlighting:

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

MIP formulation: key features

Some points worth highlighting:

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

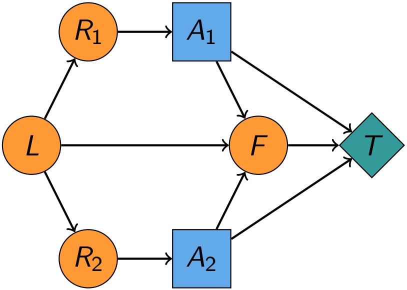
$$\pi_k(s) = \mathbb{P}\left(X_k = s_k \mid X_{I(k)} = s_{I(k)}\right) \pi_{k-1}(s),$$

   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

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

MIP formulation: example

For the 2-monitoring example we obtain:



The nodes are

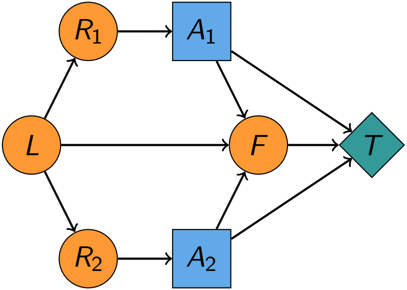
- $C = \{L, R^1, R^2, F\}$,
- $D = \{A^1, A^2\}$

The information structure:

- $I(R^i) = \{L\}, i = 1, 2,$
- $I(A^i) = \{R^i\}, i = 1, 2$
- $I(F) = \{L, A^1, A^2\}$
- $I(T) = \{A^1, A^2, F\}$

MIP formulation: example

For the 2-monitoring example we obtain:



The nodes are

- $C = \{L, R^1, R^2, F\}$,
- $D = \{A^1, A^2\}$

The information structure:

- $I(R^i) = \{L\}, i = 1, 2$,
- $I(A^i) = \{R^i\}, i = 1, 2$
- $I(F) = \{L, A^1, A^2\}$
- $I(T) = \{A^1, A^2, F\}$

An order satisfying the information structure:

$$s = (l, r^1, r^2, a^1, a^2, f).$$

MIP formulation: example

For the 2-monitoring example we obtain:

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

where $Y_T(a_1, a_2, f)$ gives the consequences associated with the failure state $F = f$ and the actions $A^1 = a_1$ and $A^2 = a_2$.

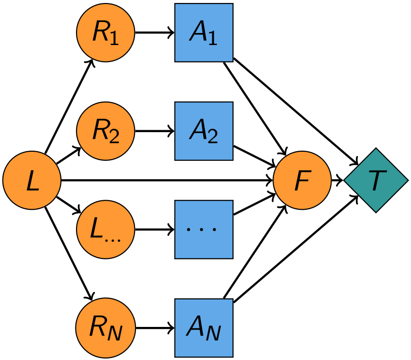
MIP formulation: example

EXCEL Example

N-monitoring problem

A stylised problem where the no-forgetting assumption doesn't hold: no decision tree formulation is possible.

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

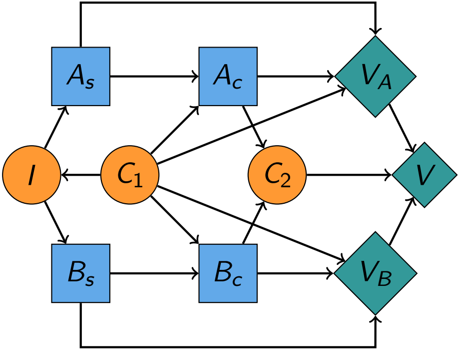


Remark: this problem can be shown to not be soluble (Lauritzen and Nilsson, 2001), a sufficient condition for SPU to converge to optimal strategies.

ECP-selection problem

An endogenously uncertain portfolio selection problem.

- Select portfolio of actions to maximise expected benefit.
- The probability distribution of the outcomes is affected by decisions
- Likewise, outcomes could be affected.



Can be seen as a generalisation of Gustafsson and Salo (2005) for to consider the endogenous case.

Outline of this talk

Computational experiments[2]

	With probability cuts		Without probability cuts		
Instance	Solution	Time (s)	Solution	Time (s)	%opt
7M1	73.21	77	73.21	21795	0.00
7M2	79.58	60	79.58	24951	0.00
7M3	67.84	105	67.84	13318	0.00
7M4	70.25	216	70.25	23464	0.00
7M5	62.48	101	62.48	18865	0.00
7M6	64.61	136	64.61	20787	0.00
7M7	60.57	110	60.57	11597	0.00
7M8	81.75	61	81.75	20531	0.00
7M9	79.50	78	79.50	17329	0.00
7M10	69.41	186	69.41	22044	0.00
Average	70.92	113	70.92	19468	0.00

Table: 10 randomly generated 7-monitoring instances. 65,536 paths.

[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[3]

	With probability cuts		Without probability cuts		
Instance	Solution	Time (s)	Solution	Time (s)	%opt
8M1	65.10	1776	27.86	25200	11260.89
8M2	75.34	2099	50.28	25200	7863.56
8M3	78.75	1090	46.27	25200	7033.49
8M4	73.59	1122	73.59	25200	4962.71
8M5	55.67	2458	34.36	25200	8092.27
8M6	78.68	1689	78.53	25200	5515.04
8M7	70.59	1714	7.73	25200	49966.90
8M8	82.67	643	13.27	25200	24215.28
8M9	75.17	1356	60.01	25200	5543.61
8M10	73.33	879	71.03	25200	4767.88
Average	72.89	1483	46.29	25200	12922.16

Table: 10 randomly generated 8-monitoring instances. 262,144 paths.

[3]**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.

Outline of this talk

Key points and takeaways

Main take aways

Decision Programming =

Decision Analysis + Stochastic Programming

Key points and takeaways

Main take aways

Decision Programming =

Decision Analysis + Stochastic 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).
- Math. programming as underpinning framework allows for flexibility in terms of imposing constraints.

Decision Programming

Fabricio Oliveira

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

ICSP 2019 - Trondheim, Norway

August 8, 2019

Aalto University
School of Science

References I

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

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.