Strategy Selection in Influence Diagrams using Imprecise Probabilities

Cassio P. de Campos and Qiang Ji

{decamc,jiq}@rpi.edu

UAI 2008

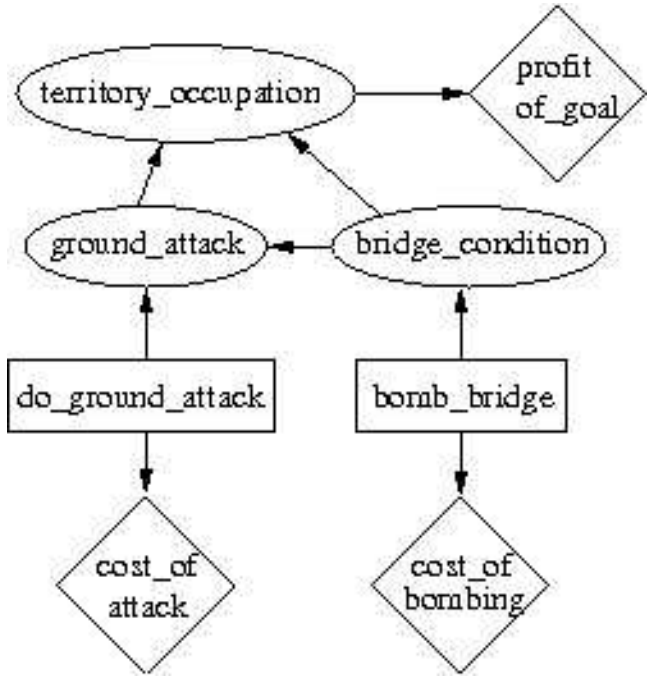
Agenda

- Limited Memory Influence Diagrams (LIMIDs)
- Strategy selection
- Reformulation as a credal network inference
- Experimental results
- Final remarks

Influence Diagrams

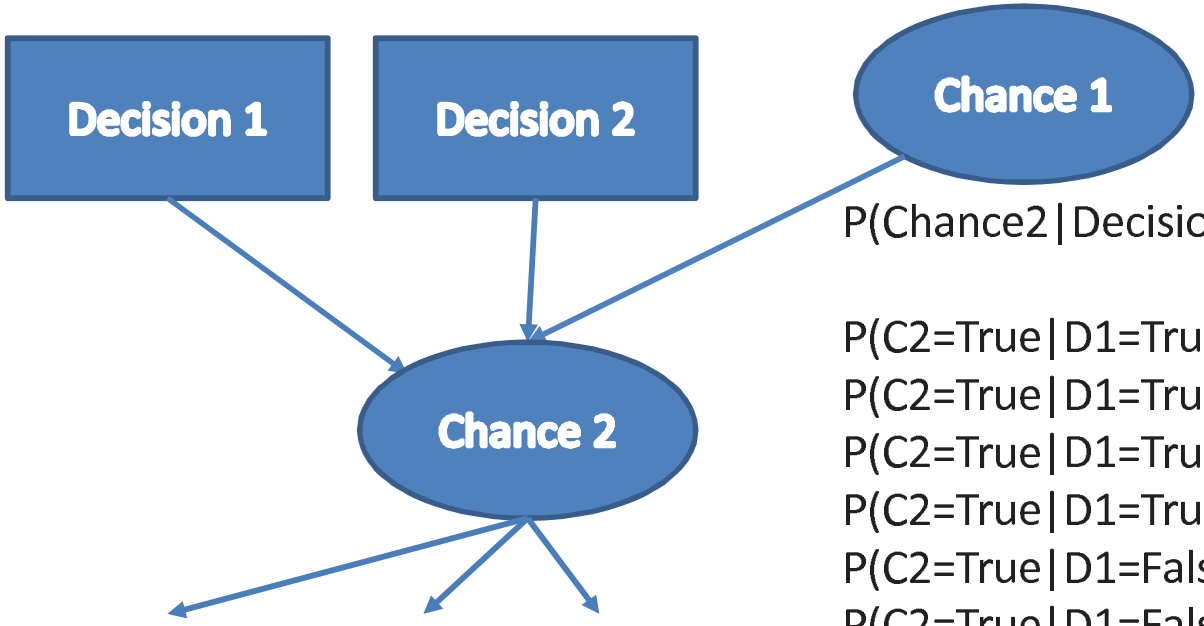
- Powerful graph based model for decision making.
- Extends the well known and widely used Bayesian networks with decision and utility nodes (besides chance nodes as in Bayesian nets).

1. Ellipses represent probabilistic dependencies.
2. Rectangles represent decisions.
3. Diamonds represent profits or costs.



Chance nodes (ellipses)

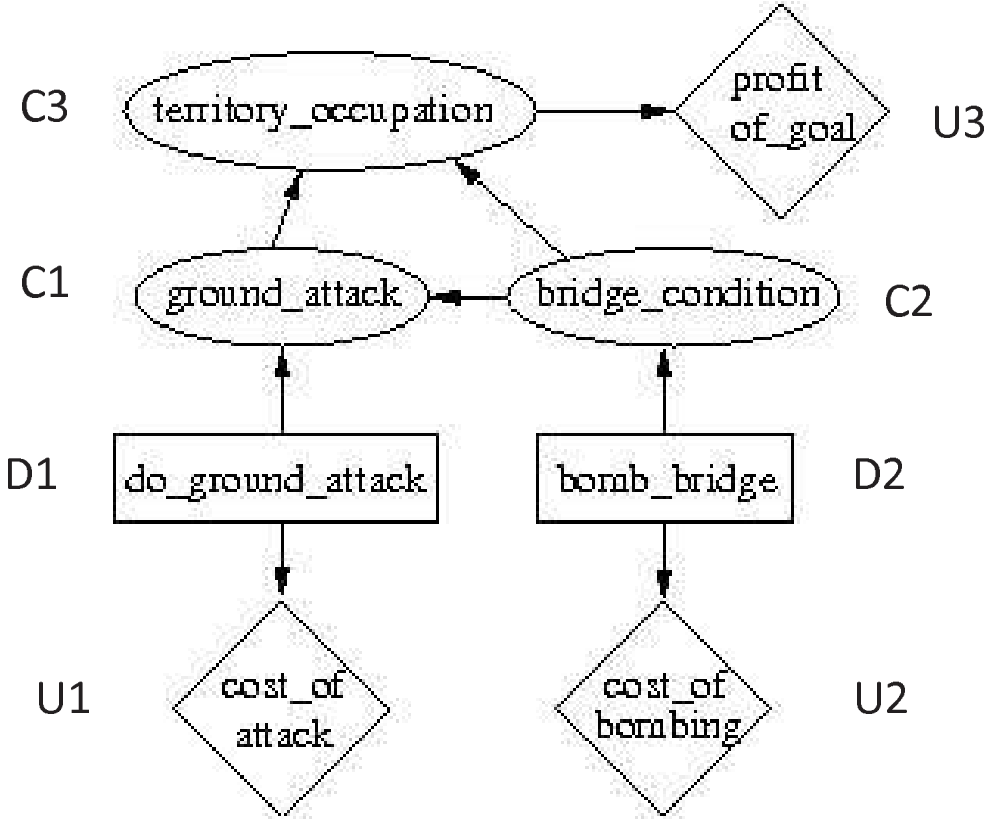
They define probabilistic dependences among the nodes. There are probabilistic distributions for each configuration of their parents.



P(Chance2|Decision1,Decision2,Chance1) ∈ [0,1]

P(C2=True|D1=True,D2=True,C1=True) = ?
P(C2=True|D1=True,D2=True,C1=False) = ?
P(C2=True|D1=True,D2=False,C1=True) = ?
P(C2=True|D1=True,D2=False,C1=False) = ?
P(C2=True|D1=False,D2=True,C1=True) = ?
P(C2=True|D1=False,D2=True,C1=False) = ?
P(C2=True|D1=False,D2=False,C1=True) = ?
P(C2=True|D1=False,D2=False,C1=False) = ?

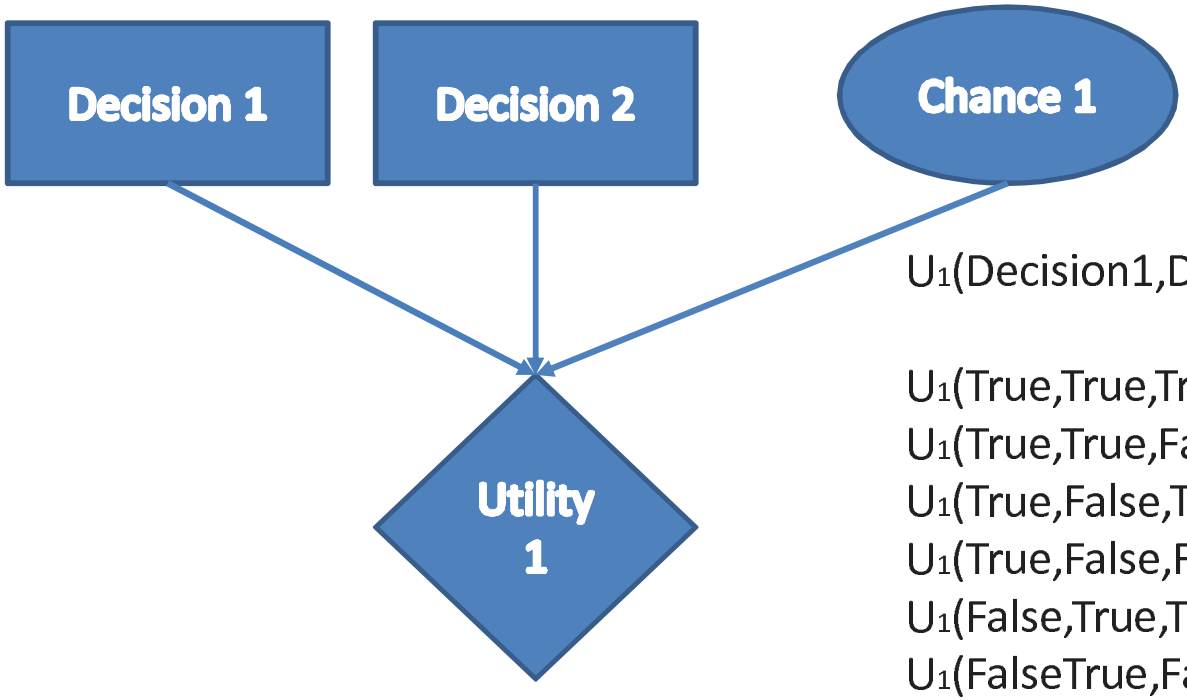
Parameterization of a Diagram



P(C3|C1,C2) → 8 parameters
P(C1|C2,D1) → 8 parameters
P(C2|D2) → 4 parameters

Utility nodes

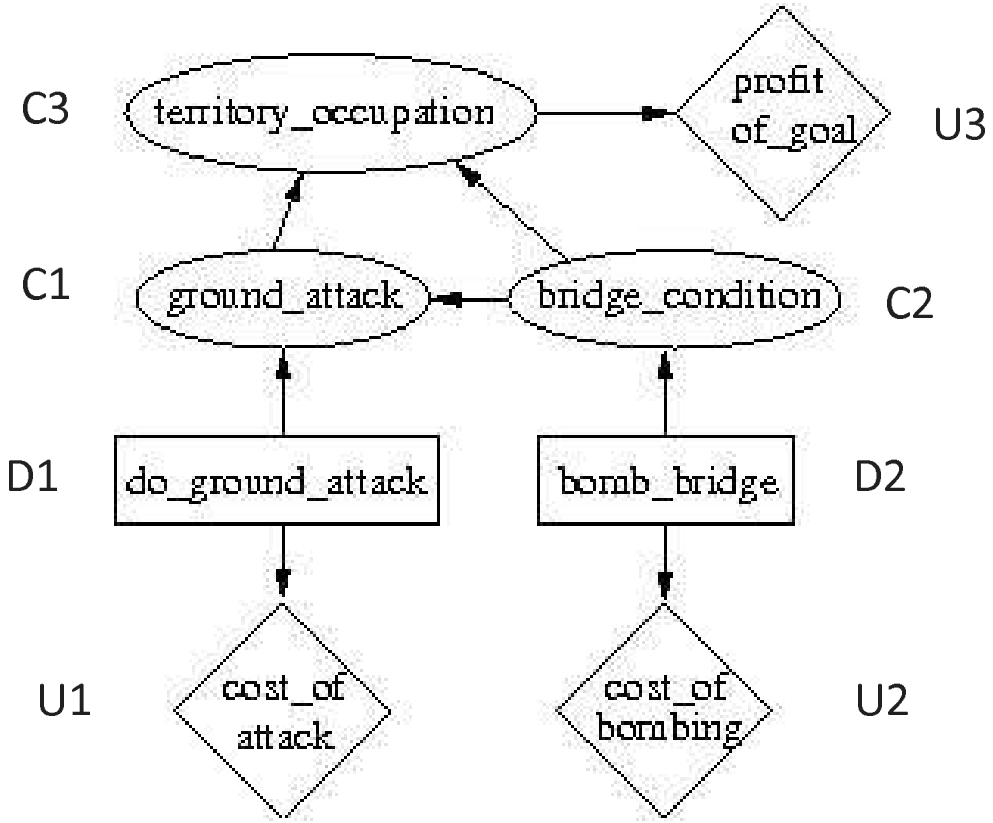
They define an utility value (profit or cost) for each configuration of their parents.



U_1(Decision1,Decision2,Chance1) $\in [-\infty, \infty]$

U_1(True,True,True) = ?
U_1(True,True,False) = ?
U_1(True,False,True) = ?
U_1(True,False,False) = ?
U_1(False,True,True) = ?
U_1(FalseTrue,False) = ?
U_1(False,False,True) = ?
U_1(False,False,False) = ?

Parameterization of a Diagram



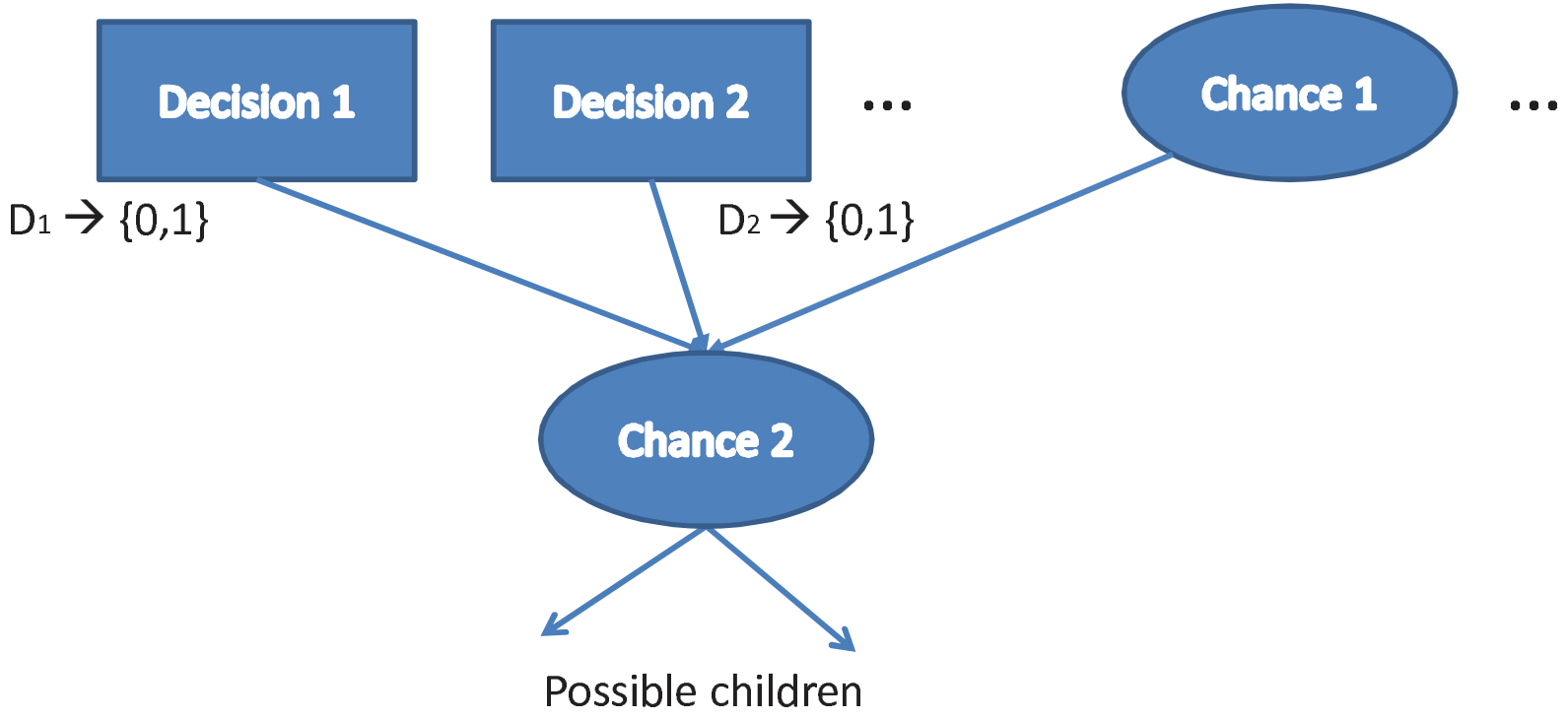
$U_1(D1)$ → 2 parameters

$U_2(D2)$ → 2 parameters

$U_3(C3)$ → 2 parameters

Decision nodes

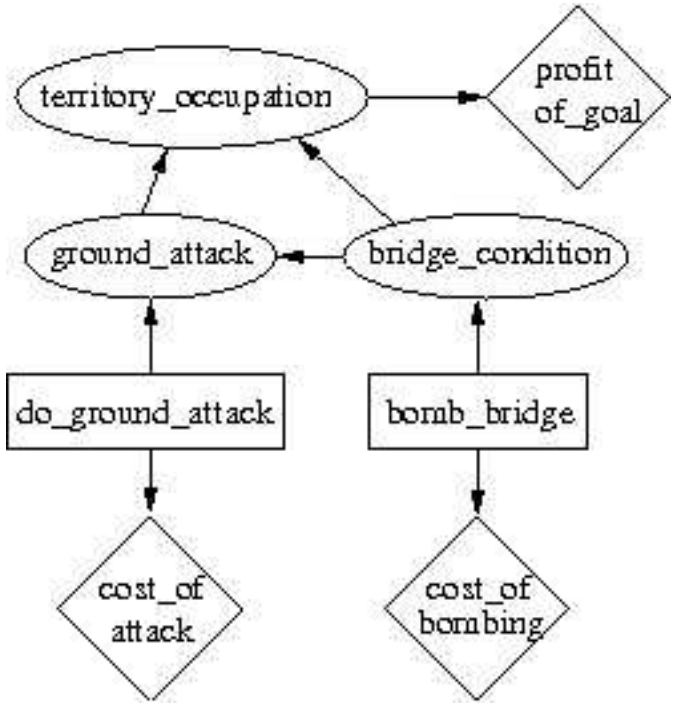
They define a decision to be taken. These values (which are unknown) are the aim of strategy selection. We do not specify them but infer them using the strategy selection procedure.



Strategy selection

- Strategy selection is the problem of taking decisions (at each decision node).
  - Shall we do a ground attack?
  - Bomb the bridge? Both?

  None?

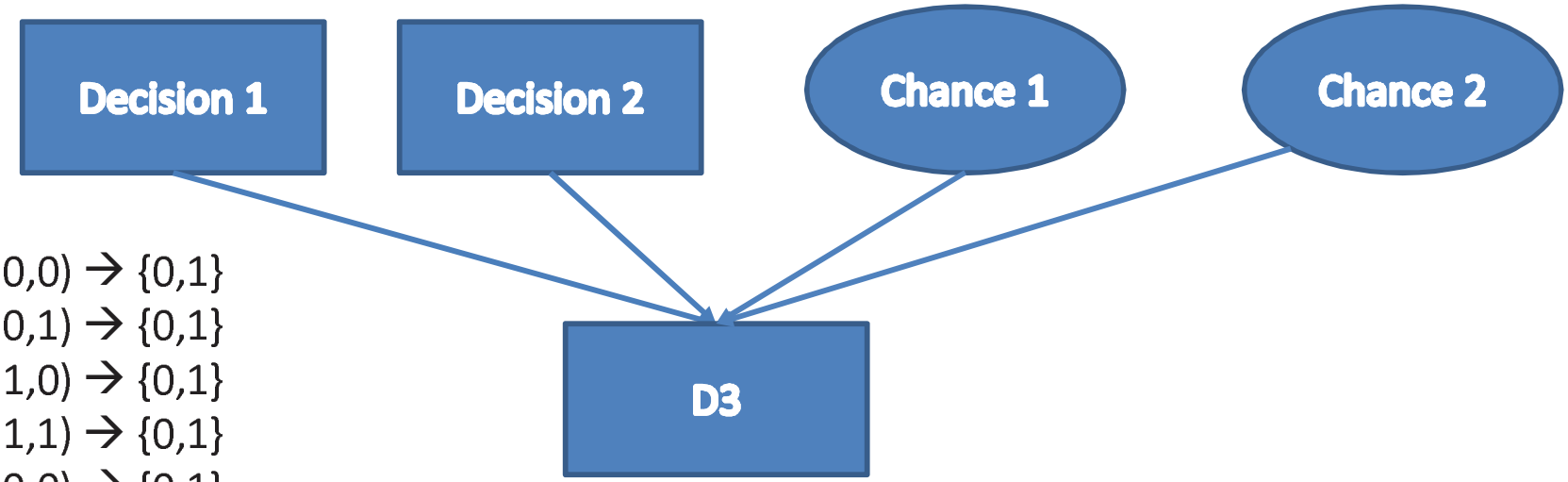


$D_{\text{do-ground-attack}} \rightarrow \{0,1\}$

$D_{\text{bomb-bridge}} \rightarrow \{0,1\}$

Number of possible strategies is huge

- If D3 is binary and has 4 binary parents, there are 2^4=16 possible configurations for the parents, which results in 2^16=65536 distinct strategies that can be taken just related to D3.



(0,0,0,0) → {0,1}
(0,0,0,1) → {0,1}
(0,0,1,0) → {0,1}
(0,0,1,1) → {0,1}
(0,1,0,0) → {0,1}
…

D_3(Decision1,Decision2,Chance1,Chance2) → {0,1}

And how to compare distinct strategies?

- Each strategy is composed by a set of local decisions (inside decision nodes).
- Expected utility measures the expected profit/cost of a given strategy.
- The goal is to find the strategy that has the maximum expected utility.

Expected Utility of a Strategy

- Given a strategy (all decisions are already taken), then the expected utility (*EU)* is

$$EU(\{D_i\}) = \sum_x \left(p(x) \sum_U f_U(parents_x(U)) \right),$$

$$p(x) = \prod_{x \downarrow C} p(C \mid parents_x(C)) \prod_{x \downarrow D} d_D(parents_x(D)),$$

- This equation is the sum of utilities of each joint configuration *x* of the diagram, weighted by the probability *p(x)* of each configuration.

How to find the best strategy?

- Reformulation that creates an equivalent credal network inference problem
- The main advantages:
  - There are already “good” solvers for the credal inference problem.
  - Create a bridge between the problems so as algorithms for one problem may be employed to solve the other.

Reformulation

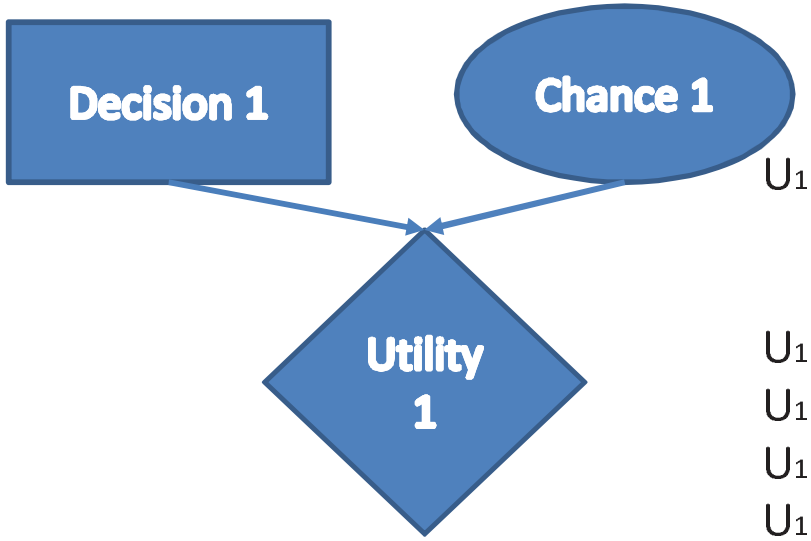
- Influence Diagram → credal network
  - Utility nodes → chance nodes
  - Decision nodes → imprecise probability nodes
  - Chances nodes do not change
- Important properties:
  - Solution to the new problem is also a solution to the original problem.
  - Strategy selection is as hard as credal network inferences.

Credal network

- A credal network is a Bayesian network where some probability values (parameter of some nodes) are unknown (while other are fixed).
- Credal inference: aims to find the parameters that maximize some objective function (in our case, the expected utility).
  - These optimal parameters will precisely define the best strategy in the influence diagram.

Reformulation – Utility nodes

- Cooper's transformation makes utility nodes become chance nodes such that $f_U(parents(U))$ is replaced by $p(u \mid parents(U))$ (a simple "careful" normalization suffices to deal with numbers that do not belong to [0,1]).



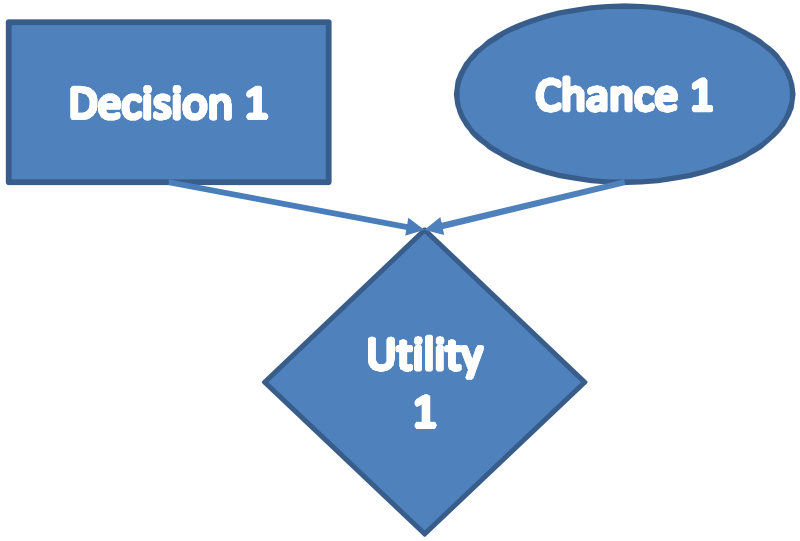
U_1(Decision1,Chance1) $\in [-\infty, \infty]$

[-100,100] → [0,1]
U_1(True,True) = 100 → $p(u_1 \mid$True,True) = 1
U_1(True,False) = -50 → $p(u_1 \mid$True,False) = 1/4
U_1(False,True) = -100 → $p(u_1 \mid$False,True) = 0
U_1(False,False) = 0 → $p(u_1 \mid$False,False) = 1/2

Reformulation – Decision nodes

- We propose to translate decision nodes to chance nodes, but with imprecise probability distributions (that is, their probability values are unknown in the model).
- It is the aim of strategy selection to find the distribution of such nodes that maximizes expected utility.

$D_1 \in \{0,1\} \rightarrow p(D_1) \in [0,1]$, that is, it is imprecise/unknown



Reformulation – objective function

$$EU = \sum_x \left(p(x) \sum_U f_U(parents_x(U)) \right)$$

- Expected utility in the credal network is

$$\sum_x \left(p(x) \sum_U p(u \mid parents_x(U)) \right) =$$

$$\sum_x \sum_U p(x)\, p(u \mid parents_x(U)) =$$

$$\sum_x \sum_U p(x,u) = \sum_U \sum_x p(x,u) = \sum_U p(u)$$

where some of the probability parameters are unknown.

Credal inference (1)

- We need to find the local probability distributions that maximizes the expected utility

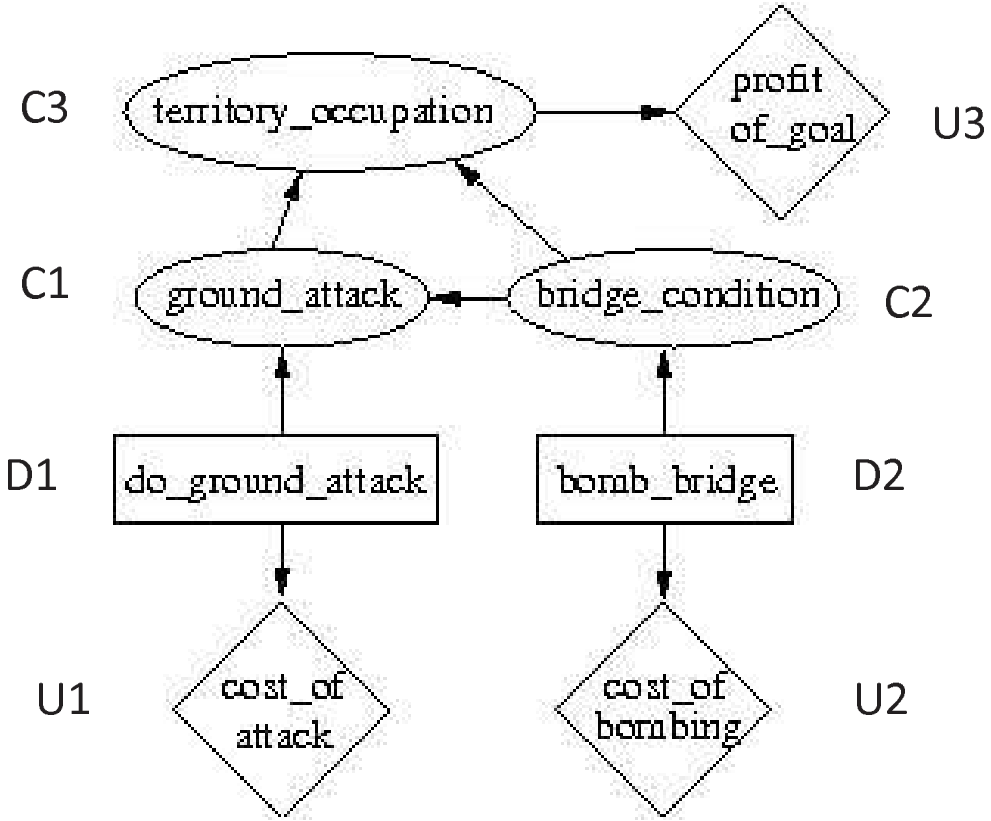
$$\sum_{U} p(u)$$

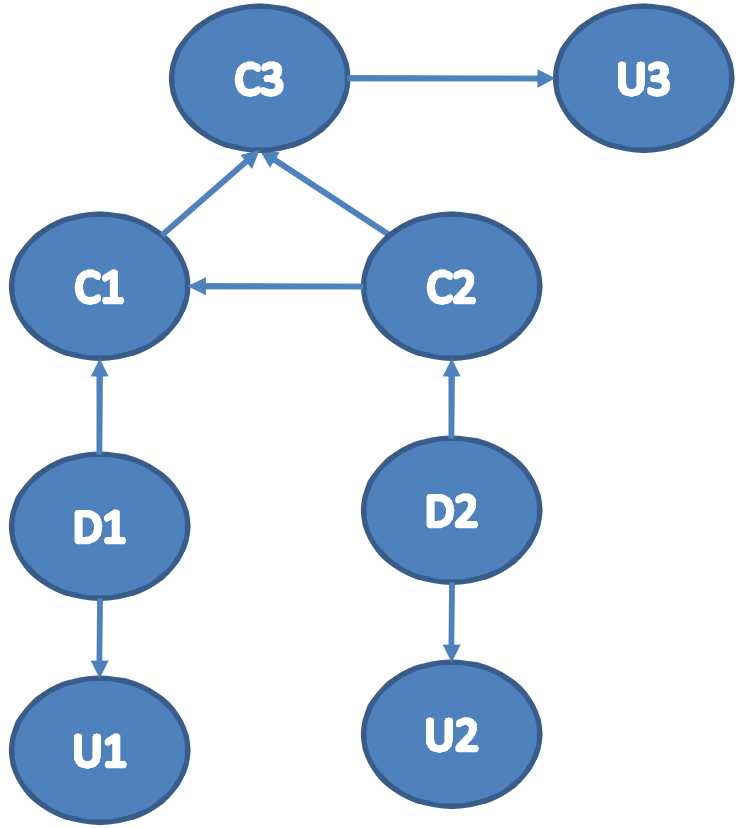
- This is a known inference in a credal network.

Credal Inference (2)

- There are many proposals for inference in credal networks.
  - Multi-linear programming
  - Linear integer programming
- Linear integer programming is a good choice because of the great number of ideas and "efficient" algorithms to solve it.
  - We have employed the CPLEX solver, one of the best linear integer implementations available.
  - Possible improvement: with small changes, we can perform parallel computations to improve performance.

Reformulation example





$U_3(C3)$ → p(U3|C3)
$U_1(D1)$ → p(U1|D1)
$U_2(D2)$ → p(U2|D2)
D_1 → p(D1) (unknown)
D_2 → p(D2) (unknown)

Now we need to find parameters that maximize the expected utility (Parameters of all nodes are known expect for D1 and D2)

Bilinear programming

Objective is $\max_p \sum_U p(u)$

s.t.

$$p(u_1) = \sum_{d \in \{d_1, \neg d_1\}} p(u_1 \mid d)\, p(d)$$

$$p(u_2) = \sum_{d \in \{d_2, \neg d_2\}} p(u_2 \mid d)\, p(d)$$

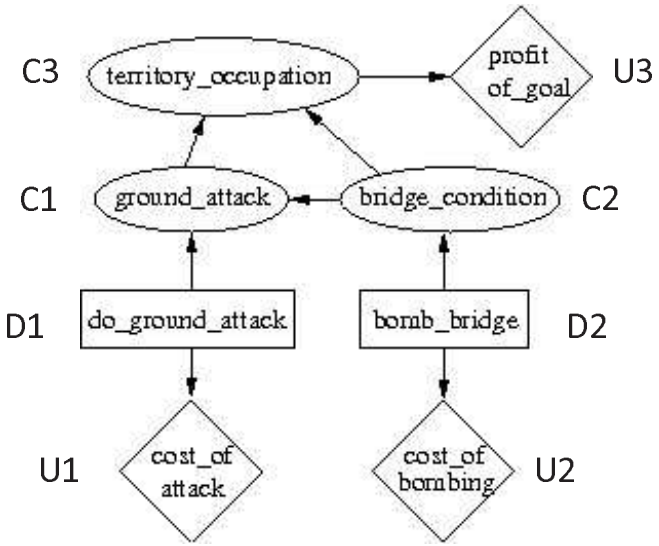
$$p(u_3) = \sum_{d \in \{d_2, \neg d_2\}} p(d)\, p(u_3 \mid d)$$

$$\forall_{d \in \{d_2, \neg d_2\}}\, p(u_3 \mid d) = \sum_{c \in \{c_2, \neg c_2\}} p(c \mid d)\, p(u_3 \mid c)$$

$$\forall_{c \in \{c_2, \neg c_2\}}\, p(u_3 \mid c) = \sum_{d \in \{d_1, \neg d_1\}} p(d)\, p(u_3 \mid c, d)$$

$$\forall_{d \in \{d_1, \neg d_1\}, c \in \{c_2, \neg c_2\}}\, p(u_3 \mid c, d) = \sum_{c' \in \{c_1, \neg c_1\}} p(c' \mid c, d)\, p(u_3 \mid c, c')$$

$$\forall_{c' \in \{c_1, \neg c_1\}, c \in \{c_2, \neg c_2\}}\, p(u_3 \mid c, c') = \sum_{c'' \in \{c_3, \neg c_3\}} p(c'' \mid c, c')\, p(u_3 \mid c'')$$



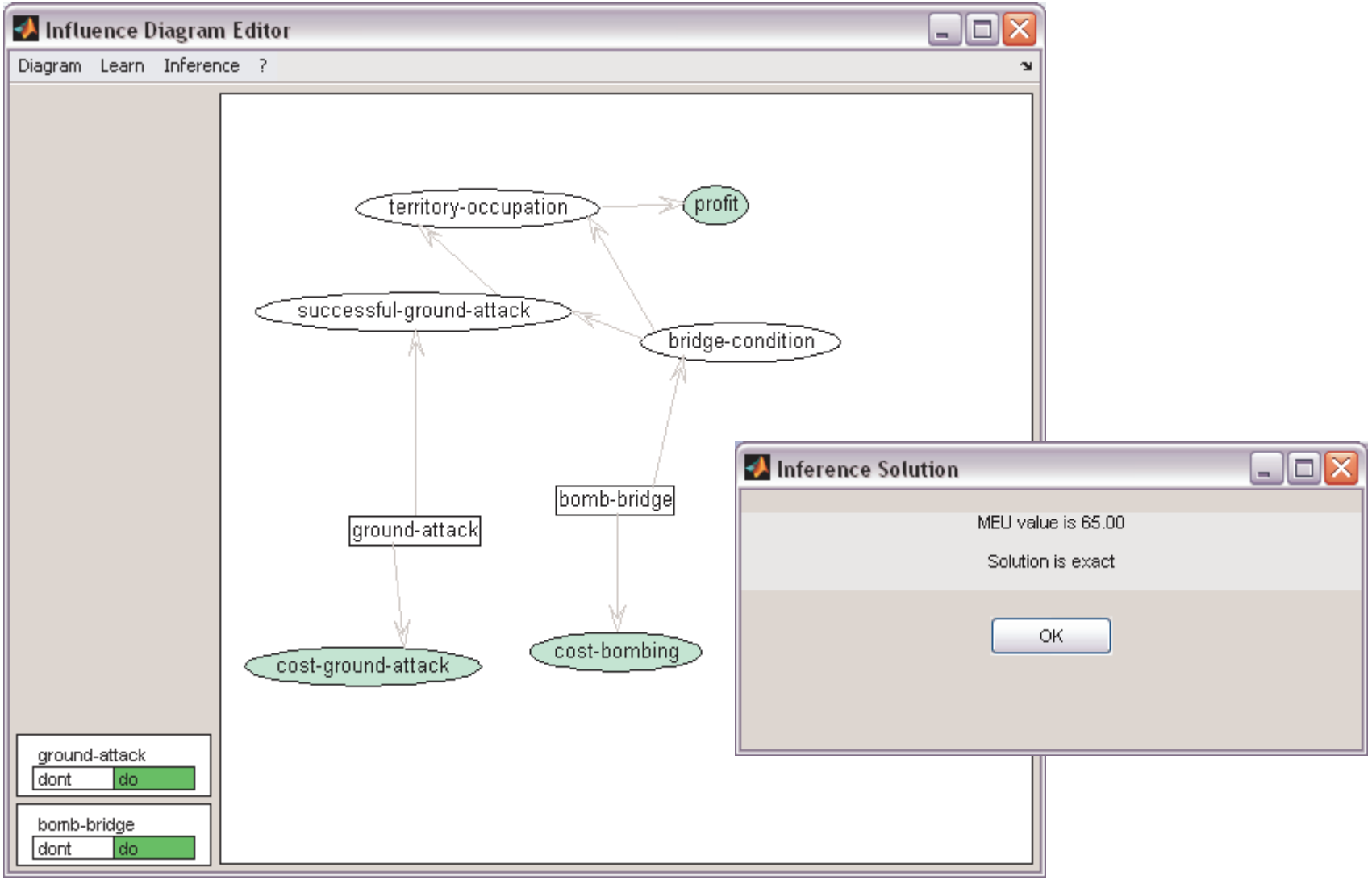
Parameter in the net

p(U3|C3)
p(C3|C1,C2)
p(C1|C2,D1)
p(C2|D2)
p(U1|D1)
p(U2|D2)
unknown p(D1)
unknown p(D2)

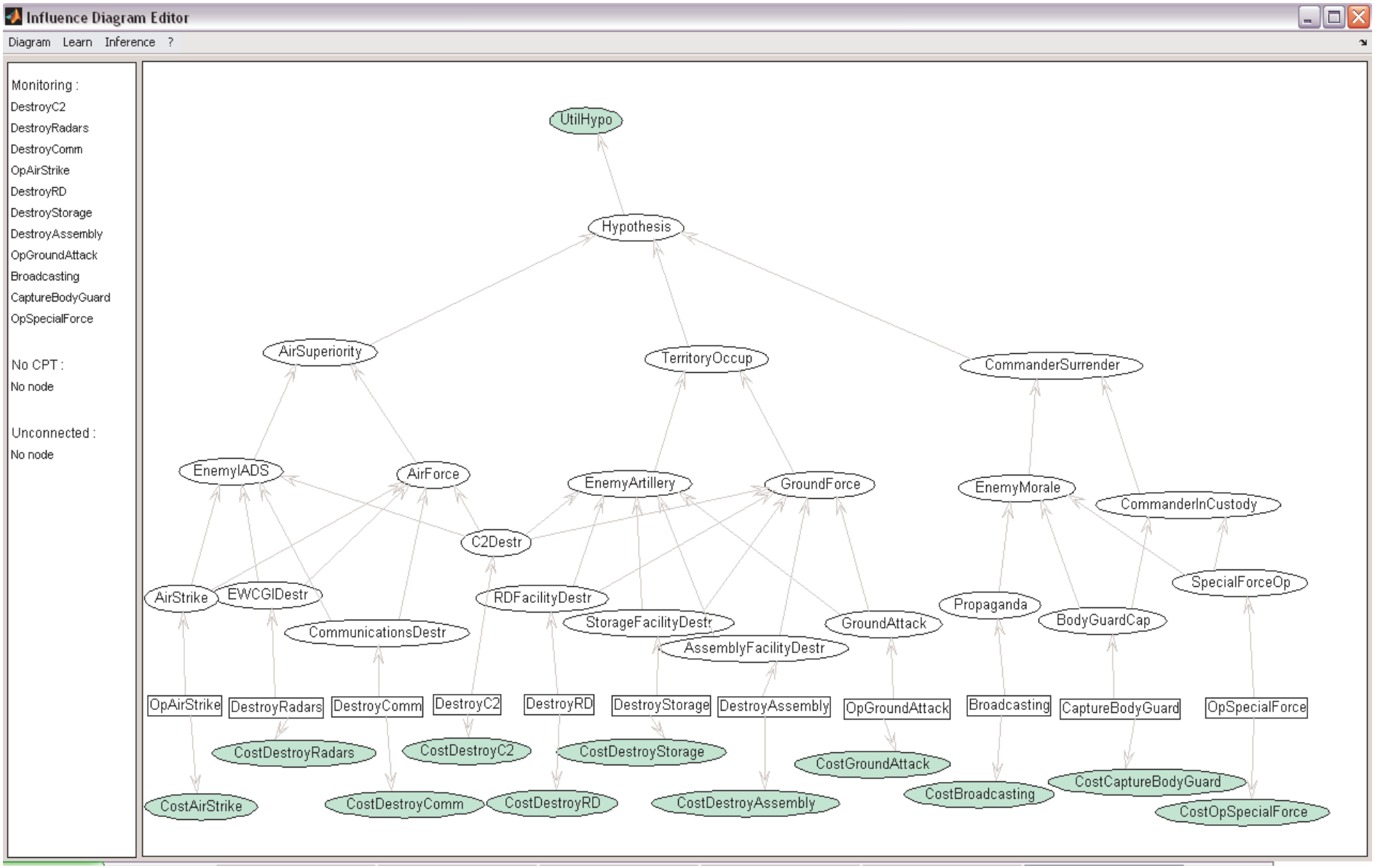
Comparison with SPU

- The usual approach to strategy selection in the literature is SPU (Single Policy Updating).
  - It is fast, but…
  - Provides approximate result.
  - There is no guarantee about optimality.
- Our approach is
  - Slower than SPU, but much faster than brute force.
  - Guarantees global optimality.
  - It is an anytime procedure (if stopped, it provides current best solution and how far in worst case it is from the global optimum).

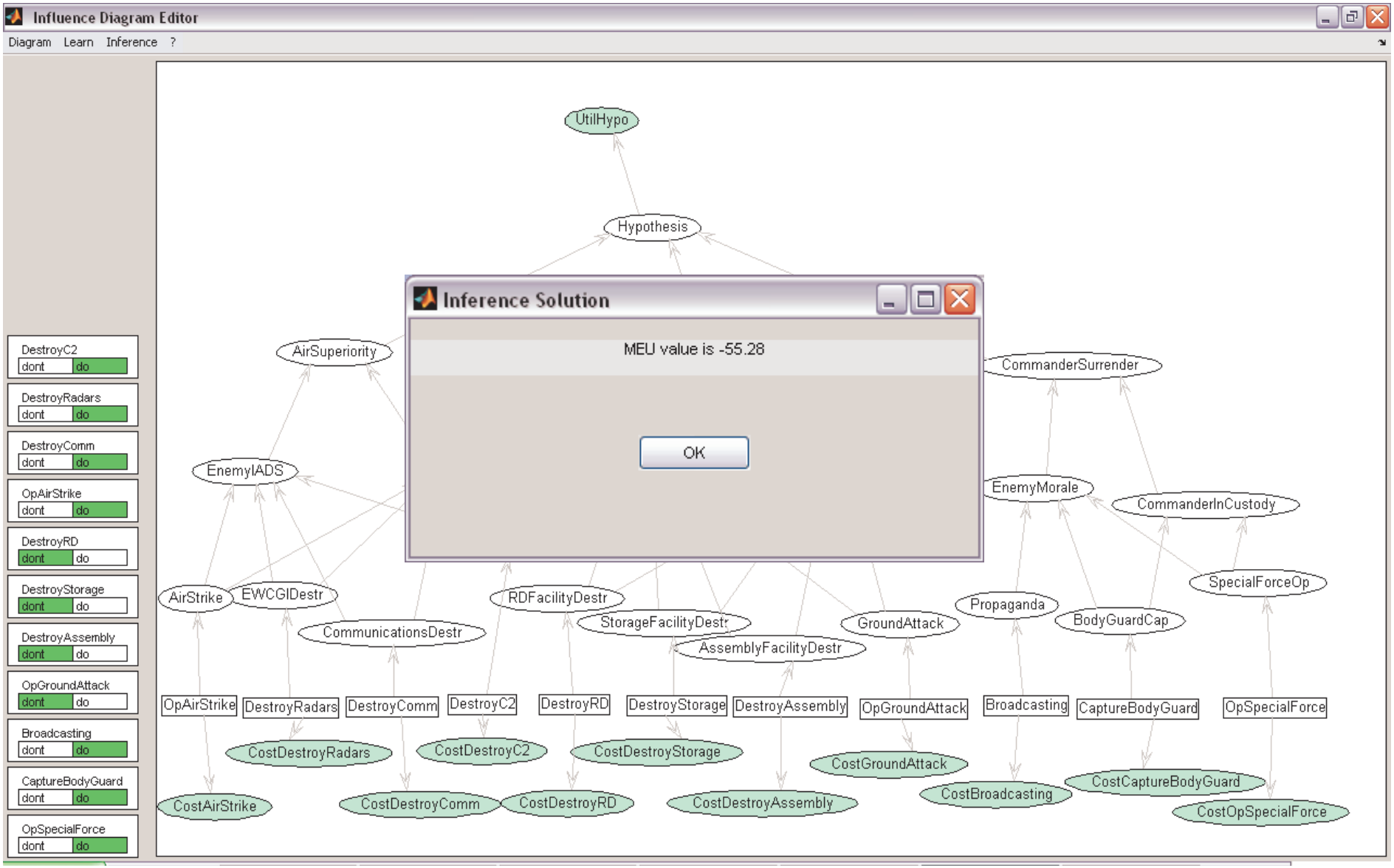
Simple EBO example



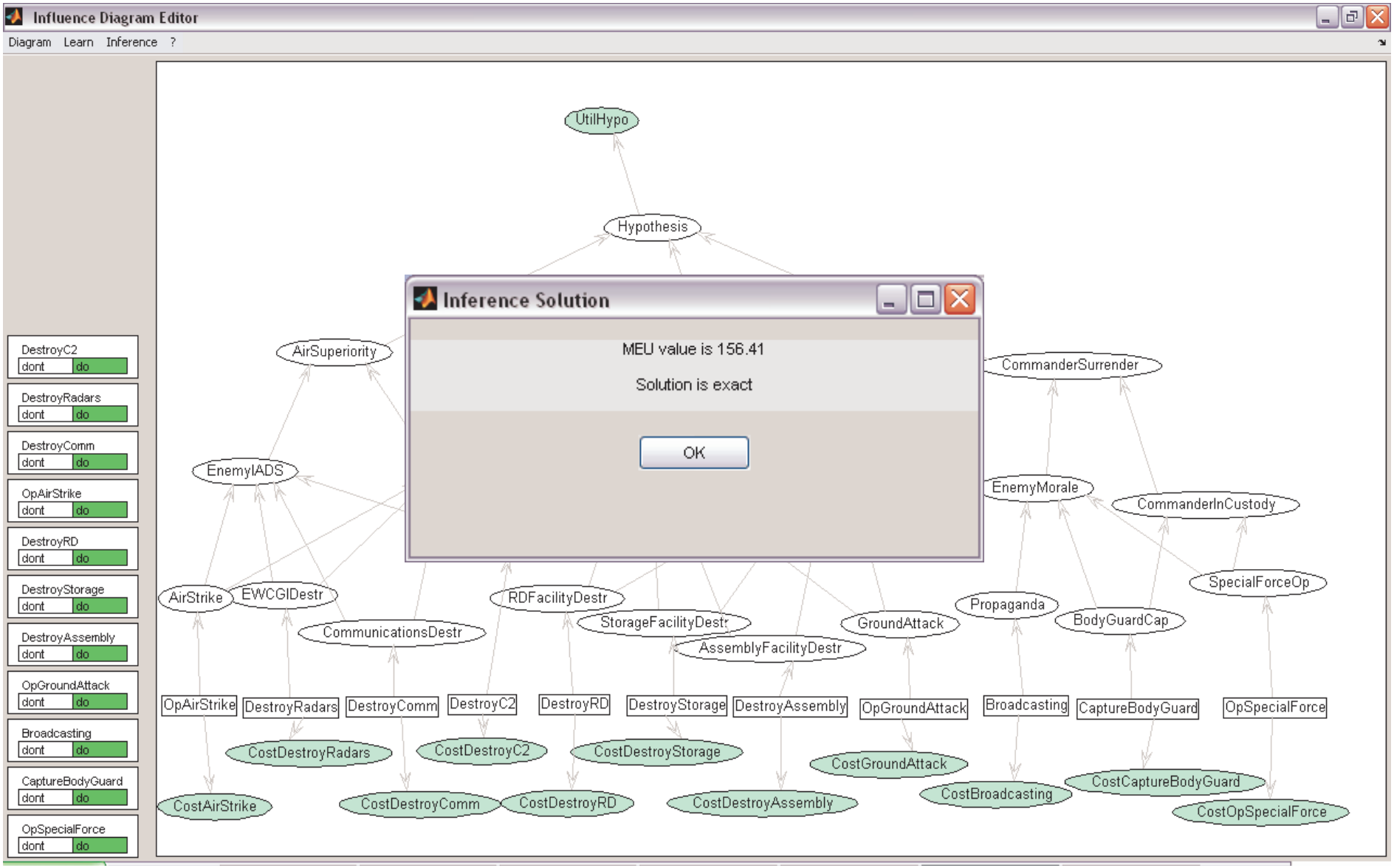
Complete EBO example



EBO example – SPU inference



EBO example – CR: new inference idea



Inference results

Random Influence Diagrams (30 cases for each configuration)

Nodes		Approx.# of	CR			SPU	
Total	Decision	Strategies	Time(sec)	Evals (B&B)	Max.Error(%)	Time(sec)	Max.Error(%)
10	3	2^{17}	0.66	5	0.000	0.10	0.740
20	6	2^{34}	1.73	125	0.000	0.39	2.788
50	10	2^{51}	30.42	4048	0.000	1.62	2.837
60	15	2^{52}	29.77	2937	0.000	2.99	1.964
70	20	2^{54}	125.06	7132	0.000	5.52	3.448
120	25	2^{102}	254.80	15626	0.544	11.58	2.193
120	30	2^{116}	403.13	5617	4.639	13.79	7.281
120	35	2^{120}	578.99	9307	5.983	16.87	11.584

- In average, CR obtains better solution than SPU's solution
- In many cases, CR found the best one within 10 minutes.

It provides an upper bound in the other cases.

Final remarks

- New idea to perform strategy selection based on credal networks.
  - Imprecise probabilities can be treated.
- Better results than previous methods.
  - Finding the global optimum is guaranteed…
  - If stopped, it provides a current solution and an upper bound about the best possible strategy.
- Other inferences in credal networks can be applied to influence diagrams.

Bilinear programming

Objective is $\max_p \sum_U p(u)$

s.t.

$$p(u_1) = \sum_{d \in \{d_1, \neg d_1\}} p(u_1 \mid d)\, p(d)$$

$$p(u_2) = \sum_{d \in \{d_2, \neg d_2\}} p(u_2 \mid d)\, p(d)$$

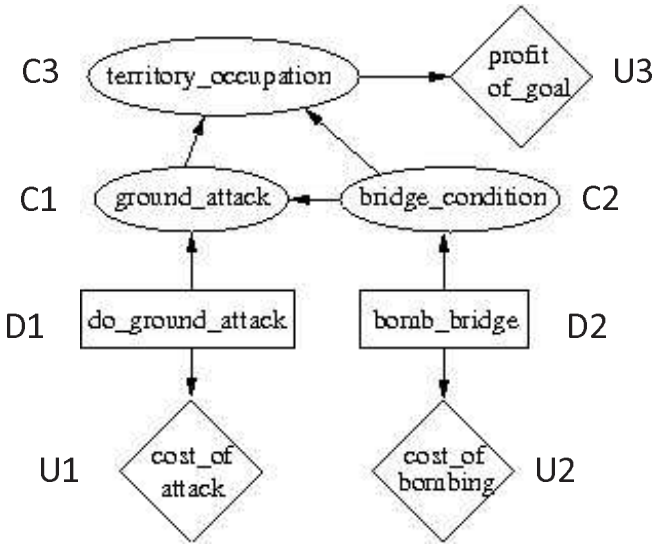
$$p(u_3) = \sum_{d \in \{d_2, \neg d_2\}} p(d)\, p(u_3 \mid d)$$

$$\forall_{d \in \{d_2, \neg d_2\}}\, p(u_3 \mid d) = \sum_{c \in \{c_2, \neg c_2\}} p(c \mid d)\, p(u_3 \mid c)$$

$$\forall_{c \in \{c_2, \neg c_2\}}\, p(u_3 \mid c) = \sum_{d \in \{d_1, \neg d_1\}} p(d)\, p(u_3 \mid c, d)$$

$$\forall_{d \in \{d_1, \neg d_1\}, c \in \{c_2, \neg c_2\}}\, p(u_3 \mid c, d) = \sum_{c' \in \{c_1, \neg c_1\}} p(c' \mid c, d)\, p(u_3 \mid c, c')$$

$$\forall_{c' \in \{c_1, \neg c_1\}, c \in \{c_2, \neg c_2\}}\, p(u_3 \mid c, c') = \sum_{c'' \in \{c_3, \neg c_3\}} p(c'' \mid c, c')\, p(u_3 \mid c'')$$



Parameter in the net

p(U3|C3)
p(C3|C1,C2)
p(C1|C2,D1)
p(C2|D2)
p(U1|D1)
p(U2|D2)
unknown p(D1)
unknown p(D2)

Linear integer programming

Fixed Parameter in the net

p(U3|C3)
p(C3|C1,C2)
p(C1|C2,D1)
p(C2|D2)
p(U1|D1)
p(U2|D2)

Objective is $\max_p \sum_U p(u)$ ← linear

$$p(u_1) = \sum_{d \in \{d_1, \neg d_1\}} p(u_1 \mid d)\, p(d)$$ ← Linear integer (p(u_1|d) is fixed)

$$p(u_2) = \sum_{d \in \{d_2, \neg d_2\}} p(u_2 \mid d)\, p(d)$$ ← Linear integer (p(u_2|d) is fixed)

$$p(u_3) = \sum_{d \in \{d_2, \neg d_2\}} p(d)\, p(u_3 \mid d)$$ ← non-linear, but p(d) is zero-one

$$\forall_{d \in \{d_2, \neg d_2\}}\, p(u_3 \mid d) = \sum_{c \in \{c_2, \neg c_2\}} p(c \mid d)\, p(u_3 \mid c)$$ ← Linear (p(c|d) is fixed)

$$\forall_{c \in \{c_2, \neg c_2\}}\, p(u_3 \mid c) = \sum_{d \in \{d_1, \neg d_1\}} p(d)\, p(u_3 \mid c, d)$$ ← non-linear, but p(d) is zero-one

$$\forall_{d \in \{d_1, \neg d_1\}, c \in \{c_2, \neg c_2\}}\, p(u_3 \mid c, d) = \sum_{c' \in \{c_1, \neg c_1\}} p(c' \mid c, d)\, p(u_3 \mid c, c')$$ ← Linear (p(c'|c,d) is fixed)

$$\forall_{c' \in \{c_1, \neg c_1\}, c \in \{c_2, \neg c_2\}}\, p(u_3 \mid c, c') = \sum_{c'' \in \{c_3, \neg c_3\}} p(c'' \mid c, c')\, p(u_3 \mid c'')$$ ← Linear (p(c''|c,c') is fixed)

Artificial variable to remove nonlinearity

- All terms in expression of the programming problem are already linear or linear integer, expect for $p(d)\,p(u_3 \mid c,d)$
- As *p(d)* is Boolean (in {0,1}), we can replace this term with an artificial variable *y* such that these two additional constraints are enforced:

$$0 \le y \le p(u_3 \mid c,d)$$

$$p(u_3 \mid c,d) + p(d) - 1 \le y \le p(d)$$

- Note that *p(d)=0* implies y=0, while *p(d)=1* implies *y=* $p(u_3 \mid c,d)$ exactly as the non-linear term would