Algorithms for Imprecise Probability
Part II

Cassio P. de Campos[1] and Fabio G. Cozman[2]

[1]Rensselaer Polytechnic Institute
Troy, NY, USA
[2]Engineering School, University of São Paulo
São Paulo, SP, Brazil

SIPTA School, July, 2008

Agenda

Introduction

Algorithms and approximation methods (for strong extensions)

Sequential decision making

Overview

- Part I: algorithms without independence (previous talk...).
- Part II: algorithms with independence (this talk).

Overview (some more)

- Part I: algorithms without independence (previous talk...).
- Part II: algorithms with independence (this talk).
  - Basics about strong/epistemic independence.
  - Credal networks under strong independence (exact/approximate inference).
  - Sequential decision making.

Reminder: stochastic independence

1. X and Y are independent when
$$P(\{Y \in B\}|\{X \in A\}) = P(\{Y \in B\})$$
whenever $P(\{X \in A\}) > 0$.
2. X and Y are independent when
$$P(\{Y \in B\} \cap \{X \in A\}) = P(\{Y \in B\})\, P(\{X \in A\})\,.$$

General stochastic independence

1. Variables $\{X_i\}_{i=1}^n$ are *independent* if

$$E[f_i(X_i) | \cap_{j \neq i} \{X_j \in A_j\}] = E[f_i(X_i)]\,,$$

   for

   - all functions $f_i(X_i)$
   - all events $\cap_{j \neq i} \{X_j \in A_j\}$ with positive probability.

2. That is, for all functions $f_i(X_i)$,

$$E\left[\prod_{i=1}^n f_i(X_i)\right] = \prod_{i=1}^n E[f_i(X_i)]\,.$$

   Or, for all sets of events $\{A_i\}_{i=1}^n$,

$$P(\cap_{i=1}^n \{X_i \in A_i\}) = \prod_{i=1}^n P(\{X_i \in A_i\})\,.$$

Strong independence

- X and Y are *strongly independent* when $K(X, Y)$ is the convex hull of a set of distributions satisfying strict independence.

- Equivalently (for closed credal sets):
  X and Y are strongly independent iff for any bounded function $f(X, Y)$,

$$\underline{E}[f(X, Y)] = \min\left(E_P[f(X, Y)] : P = P_X P_Y\right).$$

Epistemic independence

- Walley proposes a different concept: Y is epistemically irrelevant to X if for any bounded function $f(X)$,

$$\underline{E}[f(X)|Y \in B] = \underline{E}[f(X)] \qquad \text{for nonempty } \{Y \in B\}.$$

- Walley's clever idea: "symmetrize" irrelevance (this is actually a strategy by Keynes).

- X and Y are *epistemically independent* if Y is epistemically irrelevant to X and X is epistemically irrelevant to Y.

Strong $\neq$ Epistemic

- Two binary variables X and Y.
- $P(X = 0) \in [2/5, 1/2]$ and $P(Y = 0) \in [2/5, 1/2]$.
- Epistemic independence of X and Y: $K(X, Y)$ is convex hull of

$$[1/4, 1/4, 1/4, 1/4], [4/25, 6/25, 6/25, 9/25],$$

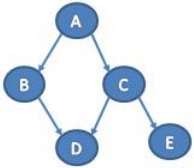
$$[1/5, 1/5, 3/10, 3/10], [1/5, 3/10, 1/5, 3/10],$$

$$[2/9, 2/9, 2/9, 1/3], [2/11, 3/11, 3/11, 3/11],$$

Exercise

Write down the linear constraints that must be satisfied by $K(X, Y)$ in the previous example.

Credal networks (epistemic)



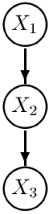
- Directed Acyclic Graph with “epistemic” Markov condition: *each variable is epistemically independent of non-descendants given its parents*.
- Local credal sets $K(X|pa(X))$ defined through convex constraints.
- Largest joint credal set satisfying all assessments: *epistemic extension* (not standard!).
- VERY difficult to handle.
- Does not even respect d-separation!

Small example

- Take 4 binary variables.
- Markov chain: $W \to X \to Y \to Z$.
- Impose the "epistemic" Markov condition.
- Joint credal set $K(W, X, Y, Z)$ has 6.000.000 vertices.
- There is an approach based on multilinear programming.

Multilinear program

- Multilinear program constructed in steps.
- For each new variable, constraints for all "previous" variables are built.
- Then each new variable introduces a set of constraints.



(a)

$$
\begin{aligned}
&p_k \geq 0 \text{ for all } k\\
&\textstyle\sum_k p_k = 1\\
&\underline{P}(1) \leq P(1) \leq \overline{P}(1)\\
&\underline{P}(2|1) \leq P(2|1) \leq \overline{P}(2|1)\\
&\underline{P}(2|\hat{1}) \leq P(2|\hat{1}) \leq \overline{P}(2|\hat{1})\\
&\underline{P}(3|2) \leq P(3|2,1) \leq \overline{P}(3|2)\\
&\underline{P}(3|\hat{2}) \leq P(3|\hat{2},1) \leq \overline{P}(3|\hat{2})\\
&\underline{P}(3|2) \leq P(3|2,\hat{1}) \leq \overline{P}(3|2)\\
&\underline{P}(3|\hat{2}) \leq P(3|\hat{2},\hat{1}) \leq \overline{P}(3|\hat{2})
\end{aligned}
$$

(b)

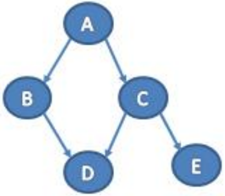
$$
\begin{aligned}
P(2,3) \times q_{3,3}(1,2) &= P(1,2,3) \times (q_{3,3}(1,2) + q_{3,3}(\hat{1},2))\\
P(\hat{2},3) \times q_{3,3}(1,\hat{2}) &= P(1,\hat{2},3) \times (q_{3,3}(1,\hat{2}) + q_{3,3}(\hat{1},\hat{2}))\\
P(2,3) \times q_{3,3}(\hat{1},2) &= P(\hat{1},2,3) \times (q_{3,3}(1,2) + q_{3,3}(\hat{1},2))\\
P(\hat{2},3) \times q_{3,3}(\hat{1},\hat{2}) &= P(\hat{1},\hat{2},3) \times (q_{3,3}(1,\hat{2}) + q_{3,3}(\hat{1},\hat{2}))
\end{aligned}
$$

(c)

$$
\begin{aligned}
&q_{3,3}(X_1,X_2) \geq 0, \text{ all } X_1, X_2\\
&\textstyle\sum_{X_1,X_2} q_{3,3}(X_1,X_2) = 1, \quad \underline{P}(1) \leq q_{3,3}(1) \leq \overline{P}(1)\\
&\underline{P}(2|1) \leq q_{3,3}(2|1) \leq \overline{P}(2|1), \quad \underline{P}(2|\hat{1}) \leq q_{3,3}(2|\hat{1}) \leq \overline{P}(2|\hat{1})
\end{aligned}
$$

(d)

Credal networks (strong extensions)



- Directed Acyclic Graph with strong Markov condition: *each variable is strongly independent of non-descendants given its parents*.
- Local credal sets $K(X|pa(X))$ defined through convex constraints.
- Largest joint credal set satisfying all assessments: *strong extension*.

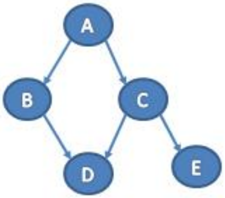
Agenda

Introduction

Algorithms and approximation methods (for strong extensions)

Sequential decision making

Reminder: Parameterization

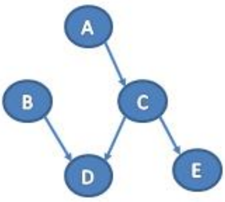


We have convex constraints on parameters:

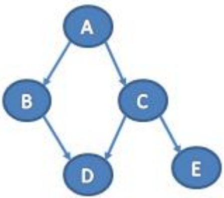
- $p(A)$,
- $p(B|a), p(B|\neg a)$,
- $p(C|a), p(C|\neg a)$,
- $p(D|b, c), p(D|\neg b, c), p(D|b, \neg c), p(D|\neg b, \neg c)$,
- $p(E|c), p(E|\neg c)$,

and we want to optimize an objective function.

Reminder: Network topology



(a) Polytree



(b) Multi-connected

Inferences

We will mainly deal with the following problem:

$$\underline{p}(\mathbf{q}|\mathbf{e}) = \min_{p \in K(\mathcal{X})} p(\mathbf{q}|\mathbf{e}).$$

or

$$\overline{p}(\mathbf{q}|\mathbf{e}) = \max_{p \in K(\mathcal{X})} p(\mathbf{q}|\mathbf{e}).$$

where $\mathbf{q}$ is an instantiation for $\mathbf{Q}$ (query variables) and $\mathbf{e}$ is an instantiation for $\mathbf{E}$ (observation variables) such that $\mathbf{Q}, \mathbf{E} \subseteq \mathcal{X}$ and $\mathbf{Q} \cap \mathbf{E} = \emptyset$.
This is known as the *belief updating problem* in credal networks.

Basic result

- Every lower/upper expectation is attained at a vertex of the strong extension.
- Every lower/upper conditional expectation is attained at a vertex of the strong extension (discarding zero probabilities).

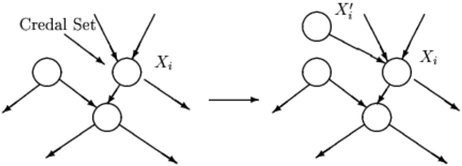
Cano–Cano–Moral transformation

A first approach is to use CCM transformation and to apply a MAP inference. Advantages:

- We can straightforward employ existent MAP techniques to solve the problem.

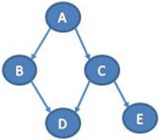
Disadvantages:

- Extreme points of credal sets must be available.
  - Large number of extreme points generate hard MAP instances.
- Instances of MAP problems are "close to" worst-case scenarios w.r.t. number of MAP variables.
  - The worst-case for MAP inferences happens when half of variables are MAP variables, which is exactly our situation.



Exercises

- Evaluate $\overline{p}(a|e)$ using the following credal network.



$p(a) \in [0.1, 0.3]$, $p(c|a) = 0.5$, $p(c|\neg a) = 0.8$, $p(e|c) \in [0.6, 0.9]$, $p(e|\neg c) = 0.5$, $p(b|\neg a) \in [0.1, 0.5]$, $p(d|b, c) \in [0.1, 0.5]$, $p(d|\neg b, c) = 0.2$ and other parameters are vacuous.

- Translate the credal network into a Bayesian network using the CCM transformation.
- Find a parameterization that respects the credal network and maximizes the entropy in each local conditional distribution.

The key insight

- To obtain exact inferences, it is necessary to “translate” a credal network into an optimization problem.
    - No “easy” alternative in general (no “propagation” scheme that works in general is known).
    - This translation may exploit the structure of the network.
- Approximate inferences can mimic the “propagation” schemes that are used in Bayesian networks.

Translating credal networks...

1. Andersen & Hooker's method.
2. Symbolic variable elimination.
3. Bilinear formulation (for some cases).

Andersen and Hooker's algorithm

The idea is to solve a non-linear program where:

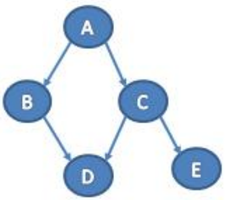
- Optimization variables are the *atoms* of the problem (all possible worlds)
  - There are an exponential number of them w.r.t the number of random variables.
- Use constraints that define the local credal sets.
  - Summations of atoms define them.
- Include also all independence relations implied by the Markov condition.
  - The amount of such constraints can also be exponential.

Hard to deal even with small networks

This idea spends a huge computational effort.

Different idea: Symbolic Variable Elimination

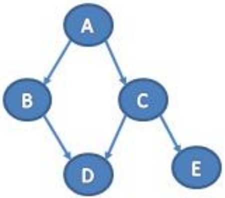
Query example: $\overline{p}(d)$



- Multi-linear problem: $\max p(d)$ subject to
  - Bucket A: $\sum_A p(A)p(B|A)p(C|A) = p(B, C)$ for all B, C.
  - Bucket B: $\sum_B p(B, C)p(d|B, C) = p(C, d)$, for all C.
  - Bucket C: $\sum_C p(C, d) = p(d)$.
- $p(\cdot)$ are the optimization variables. They are subject to these constraints plus local constraints of the credal network.

Symbolic Variable Elimination I

Query example: $\overline{p}(a|d)$



- Multi-linear problem: $\max t$ subject to
  - Constraint of t: $tp(d) = p(a, d)$ and $t \in [0, 1]$.
  - To relate $p(d)$: use constraints of previous slide.
  - Bucket A: $p(B|a)p(C|a)p(a) = p(a, B, C)$ for all B, C.
  - Bucket B: $\sum_B p(a, B, C)p(d|B, C) = p(a, C, d)$, for all C.
  - Bucket C: $\sum_C p(a, C, d) = p(a, d)$.

Symbolic Variable Elimination II

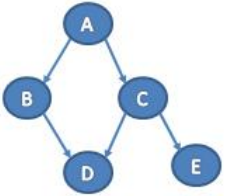
- Exactly same procedure as in Bayesian networks, but it is run symbolically to generate multi-linear constraints.
- Complexity is the same as in Bayesian networks, that is, exponential in the induced width (or tree-width) of the network: maximum clique size in the moralized graph.
  - Graph moralization: marry parents (include edges connecting parents of common children) and remove arc directions.
- Multi-linear programming techniques may be employed to find exact and approximate solutions.

Bilinear formulation

- Same general idea of the variable elimination: to produce multi-linear constraints that define the query to later on apply optimization techniques.
- Differently from variable elimination, variables are processed in a top-down ordering using conditional auxiliary probabilities.
- Complexity is proportional to the path-width instead of tree-width.

Bilinear procedure

Query example: $\overline{p}(d)$



- Bilinear problem: $\max p(d)$ subject to
  - A: $p(d) = \sum_A p(A)p(d|A)$.
  - B: $p(d|A) = \sum_B p(B|A)p(d|A,B)$ for all A.
  - C: $p(d|A,B) = \sum_C p(C|A)p(d|B,C)$ for all A, B.
  - Note that $p(d|A)$ and $p(d|A,B)$ are auxiliary optimization variables.

Bilinear formulation

Disadvantage:

- Path-width is usually greater than tree-width.
- Let w_p be the path-width and w_t the tree-width of a graph. It is known that $w_p \leq w_t \log w_t$, but that is still much greater, as the algorithms are exponential in these numbers...

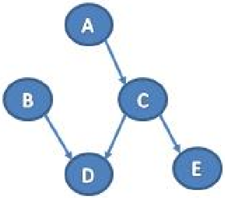
$$\exp(w_t) << w_t^{w_t} = \exp(w_t \log w_t)$$

Advantages:

- All non-linear terms have only two factors.
- One of them is a network parameter! Linear integer programming can be used when extreme points of credal sets are known.
- Path-width and tree-width are equivalent in polytrees.

Inference is not always hard: 2U

Marginal queries in binary polytree credal networks are treated by the 2U algorithm.



2U is a propagation algorithm with ideas similar to belief propagation in Bayesian networks.

Very simple example

$p(a) \in [0.1, 0.5]$, $p(b|a) \in [0.6, 0.9]$, $p(b|\neg a) \in [0.5, 0.6]$,
$p(c|b) \in [0.3, 0.5]$, $p(c|\neg b) \in [0.7, 0.8]$,

$$\overline{p}(a) = 0.5, \underline{p}(a) = 0.1$$

$$\overline{p}(b) = \max_{p'(a) \in \{\underline{p}(a), \overline{p}(a)\}} \sum_A \overline{p}(b|A) \cdot p'(A)$$

$$\overline{p}(b) = \max_{p'(a) \in \{0.1, 0.5\}} (0.9 \cdot p'(a) + 0.6 \cdot (1 - p'(a)))$$

$$\overline{p}(b) = \max\{(0.9 \cdot 0.1 + 0.6 \cdot 0.9) = 0.63; (0.9 \cdot 0.5 + 0.6 \cdot 0.5) = 0.75\} = 0.75$$

Very simple example

$p(a) \in [0.1, 0.5]$, $p(b|a) \in [0.6, 0.9]$, $p(b|\neg a) \in [0.5, 0.6]$,
$p(c|b) \in [0.3, 0.5]$, $p(c|\neg b) \in [0.7, 0.8]$,

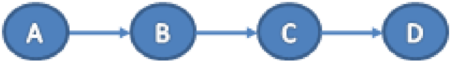
$$\overline{p}(a) = 0.5, \underline{p}(a) = 0.1$$

$$\underline{p}(b) = \min_{p'(a) \in \{\underline{p}(a), \overline{p}(a)\}} \sum_A \underline{p}(b|A) \cdot p'(A)$$

$$\underline{p}(b) = \min_{p'(a) \in \{0.1, 0.5\}} (0.6 \cdot p'(a) + 0.5 \cdot (1 - p'(a)))$$

$$\underline{p}(b) = \min\{(0.6 \cdot 0.1 + 0.5 \cdot 0.9) = 0.51; (0.6 \cdot 0.5 + 0.5 \cdot 0.5) = 0.55\} = 0.51$$

Very simple example



$p(a) \in [0.1, 0.5]$, $p(b|a) \in [0.6, 0.9]$, $p(b|\neg a) \in [0.5, 0.6]$,
$p(c|b) \in [0.3, 0.5]$, $p(c|\neg b) \in [0.7, 0.8]$,

$$\overline{p}(b) = 0.75, \underline{p}(b) = 0.51$$

$$\underline{p}(c) = \min_{p'(b) \in \{\underline{p}(b), \overline{p}(b)\}} \sum_{B} \underline{p}(c|B) \cdot p'(B)$$

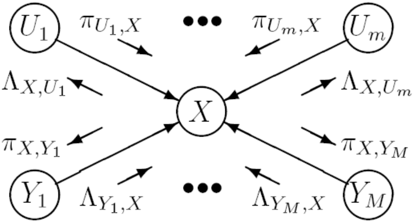
$$\underline{p}(c) = \min_{p'(b) \in \{0.51, 0.75\}} (0.3 \cdot p'(b) + 0.7 \cdot (1 - p'(b)))$$

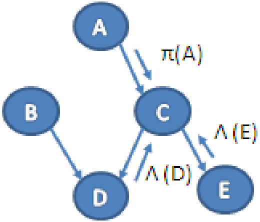
$$\underline{p}(c) = \min\{(0.3 \cdot 0.51 + 0.7 \cdot 0.49); (0.3 \cdot 0.75 + 0.7 \cdot 0.25)\}$$

And so on for $\overline{p}(c)$, $\overline{p}(d)$, $\underline{p}(d)$.

2U

With evidence, back-propagation is necessary.





X receives messages from parents and children, and propagate to nodes that have not received messages yet.

2U updating equations

$$\underline{P}[x|e] = \left(1 + \left(\frac{1}{\underline{\pi}(x)} - 1\right)\frac{1}{\underline{\Lambda}^X}\right)^{-1}$$

$$\overline{P}[x|e] = \left(1 + \left(\frac{1}{\overline{\pi}(x)} - 1\right)\frac{1}{\overline{\Lambda}^X}\right)^{-1}$$

$$\underline{\pi}(x) = \min_{\substack{j\in\{1,\ldots,n\} \\ \pi_X(u_j)\in\left\{\underline{\pi}_X(u_j),\overline{\pi}_X(u_j)\right\}}} \sum_U \underline{P}[x|U] \prod_i \pi_X(U_i)$$

$$\overline{\pi}(x) = \max_{\substack{j\in\{1,\ldots,n\} \\ \pi_X(u_j)\in\left\{\underline{\pi}_X(u_j),\overline{\pi}_X(u_j)\right\}}} \sum_U \overline{P}[x|U] \prod_i \pi_X(U_i)$$

$$\underline{\Lambda}^X = \prod_j \underline{\Lambda}^X_{Y_j}$$

$$\overline{\Lambda}^X = \prod_j \overline{\Lambda}^X_{Y_j}$$

$$\underline{\pi}_{Y_j}(x) = \left(1 + \left(\frac{1}{\underline{\pi}(x)} - 1\right)\frac{1}{\prod_{k\neq j} \underline{\Lambda}^X_{Y_k}}\right)^{-1}$$

$$\overline{\pi}_{Y_j}(x) = \left(1 + \left(\frac{1}{\overline{\pi}(x)} - 1\right)\frac{1}{\prod_{k\neq j} \overline{\Lambda}^X_{Y_k}}\right)^{-1}$$

$$\underline{\Lambda}^{U_i}_X = \min_{\substack{j\in\{1,\ldots,n\},j\neq i \\ \pi_X(u_j)\in\left\{\underline{\pi}_X(u_j),\overline{\pi}_X(u_j)\right\}}} \left(\min_{\Lambda^X\in\left\{\underline{\Lambda}^X,\overline{\Lambda}^X\right\}} \hat{\Lambda}^{U_i}_X\left(\Lambda^X\right)\right)$$

$$\overline{\Lambda}^{U_i}_X = \max_{\substack{j\in\{1,\ldots,n\},j\neq i \\ \pi_X(u_j)\in\left\{\underline{\pi}_X(u_j),\overline{\pi}_X(u_j)\right\}}} \left(\max_{\Lambda^X\in\left\{\underline{\Lambda}^X,\overline{\Lambda}^X\right\}} \hat{\overline{\Lambda}}^{U_i}_X\left(\Lambda^X\right)\right)$$

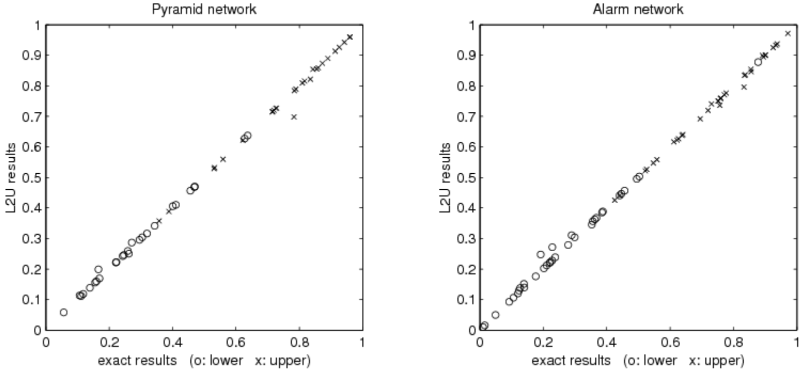
L2U

Loopy 2U is an extension of 2U to work with multi-connected networks. The idea is to employ

- Loopy belief propagation.
- 2U algorithm for dealing with intervals.

In practice, convergence is fast and error rate is small, although there is no theoretical guarantee.

L2U results


Pyramid network
L2U results
exact results (o: lower x: upper)
Alarm network
L2U results
exact results (o: lower x: upper)

Other methods

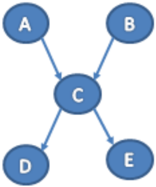
- A/R+ and A/R++.
- Iterative Local Search.
- Probability tree-based inference.
- Branch-and-bound on vertices.
- ... and others!

Remember the key insight:

...necessary to "translate" a credal network into an optimization problem.

A/R+

Approximate algorithm for polytrees (not necessarily binary).



- C receives intervals $[\underline{p}(A), \overline{p}(A)]$ for each value of A and $[\underline{p}(B), \overline{p}(B)]$ for each value of B.
- It computes $\overline{p}(C) = \max_p p(c) = \max_p \sum_{A,B} p(c|A,B)p(A)p(B)$ and $\underline{p}(C) = \min_p p(c)$ using those intervals and the credal sets $K(C|A,B)$.
- And propagates $[\underline{p}(C), \overline{p}(C)]$ (for all C) to children.

Just intervals are propagated in each node. They approximate the exact credal sets. Solution is an outer approximation.

A/R++

A/R+: $[\underline{p}(C), \overline{p}(C)]$ can be written as

$$\alpha_C \leq p(C) \leq \beta_C,$$

Possible extension: instead of propagating just those intervals, choose additional linear constraints (linearly independent).

- Only useful for non-binary variables.

Just as A/R+, the result is an outer approximation.

Exercises

- Show that A/R+ provides outer approximations for the credal belief updating problem.
- In polytrees, which reformulation usually produces a simpler optimization program: variable elimination or the bilinear translation idea? Explain your answer.
- Show that no additional constraints is useful while treating binary networks. Provide a useful constraint that could be propagated in a ternary credal network.

Iterative Local Search

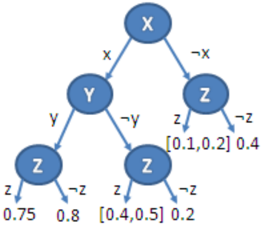
Simple idea:

- Choose a Bayesian network that comply with the credal network constraints.
- Allow parameters of a single node to vary and take the best solution.
    - Evaluations can be performed using any Bayesian network inference for each extreme point of local credal sets.
    - Extreme points need to be known, although it is possible to overcome this limitation.
- Iterate on nodes until the solution does not improve.

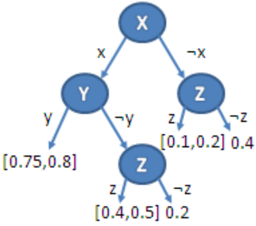
Good approximate results. It provides an inner approximation.

Outer approximation using probability trees

- Create transparent variables to describe the extreme points of the separately specified credal sets.
- Represent probability tables using probability trees (extended to deal with intervals).
- Use a Bayesian network inference but performing calculations over probability trees, while keeping trees small by pruning probability values that are close to each other.
  - The idea is to represent those similar values by small intervals and perform computations with them.



(g) Probability tree



(h) Collapsed node

Branch-and-bound on vertices

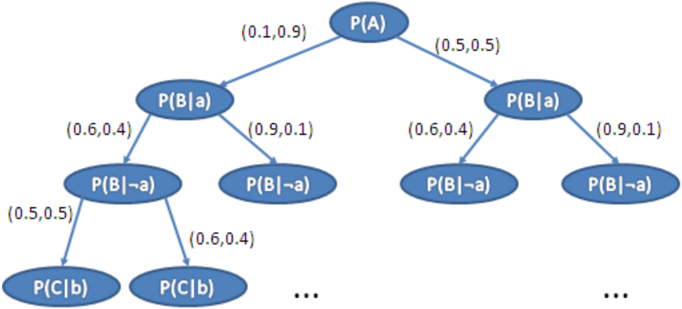
- Suppose local credal sets are separately specified and consider a decision tree where each level corresponds to a local credal set of the network.
- Each node of this tree has as many children as vertices in the local credal set. Choosing a path means fixing a local credal set in one of its vertices.
- A leaf of this tree has precise values for all local credal sets and so it is easy to compute the objetive function.

B&B procedure: search this tree for the best solution. At each node,

- Use an inner approximation to look for a better solution.
- Use an outer approximation to bound the best possible solution.
  - If the outer value of a given subtree is worse than the current value, this subtree does not need to be evaluated.

B&B

$p(a) \in [0.1, 0.5]$, $p(b|a) \in [0.6, 0.9]$, $p(b|\neg a) \in [0.5, 0.6]$,
$p(c|b) \in [0.3, 0.5]$, $p(c|\neg b) \in [0.7, 0.8]$,

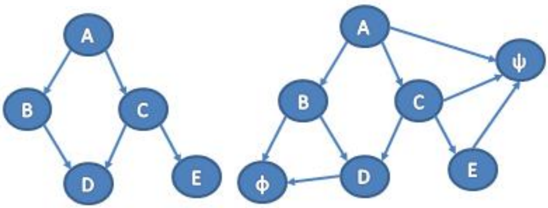


Other approximate methods

- B&B on vertices.
  - Simulated Annealing.
  - Genetic Algorithms.
- Approximate optimization methods that are able to handle multi-linear constraints.
  - Multi-linear Local search.
  - MINOS: projected Lagrangian method and reduced-gradient method.
  - SNOPT: sparse Sequential Quadradic Programming algorithm.
  - IPOPT: Primal-Dual Interior Point Filter Line Search Algorithm.
  - and much more...
- Approximate methods that handle linear integer programming.

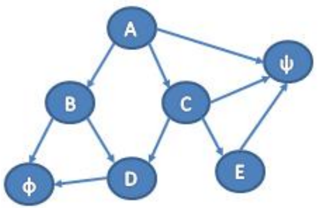
Case-study: Probabilistic Propositional Logic Networks

- Appeared first as *Bayesian Logic* (Andersen and Hooker 1994).
- Nodes are associated to propositions.
- The graph encodes (in)dependence relation among propositions.
- Probabilistic propositional sentences are not restricted to parameters of the network.
  - E.g. $p(\phi) + p(\psi) \leq 0.7$, where $\phi = b \vee d$ and $\psi = (a \vee \neg e) \wedge c$.



(i) Network structure (j) Augmented network

PPL networks: extra nodes

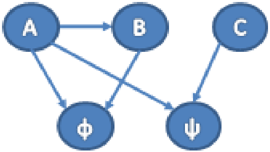


- $p(\phi) + p(\psi) \leq 0.7$, where $\phi = b \vee d$ and $\psi = (a \vee \neg e) \wedge c$.
- The conditional probability distributions at nodes ϕ and ψ are the true-tables of the corresponding logical sentences.
- Now we perform a credal network inference using a symbolic algorithm to relate the marginal probabilities $p(\phi)$ and $p(\psi)$ to other network parameters. This will generate a collection of multi-linear constraints that we include in the optimization problem, together with the constraints $p(\phi) + p(\psi) \leq 0.7$.
- Conditionals are treated similarly. E.g. for $p(\phi|\psi) \geq 0.2$, we perform a symbolic query to relate $p(\phi|\psi)$ and other network parameters.

Exercise: simple PPL manipulation

- Three boolean variables A, B, C.
- Logical sentence: $\psi = a \vee c$.
- Probabilistic logic sentence: $p(\phi) \leq 0.3$, where $\phi = \neg a \vee b$.
- Local credal sets $K(A), K(B|a), K(B|\neg a), K(C)$.

Simple PPL manipulation: obtaining an optimization program



- Nodes ϕ and ψ have true-tables as conditional distributions: $p(\psi|a,c) = p(\psi|\neg a, c) = p(\psi|a, \neg c) = 1$, $p(\psi|\neg a, \neg c) = 0$. $p(\phi|a,b) = p(\phi|\neg a, b) = p(\phi|\neg a, \neg b) = 1$, $p(\phi|a, \neg b) = 0$.
- $p(\psi) = 1$ and $p(\psi)$ is related to A and C using constraints.
  - Bucket A: $\sum_A p(A)p(\psi|A,C) = p(\psi|C)$ for all C.
  - Bucket C: $\sum_C p(C)p(\psi|C) = p(\psi)$.
- $p(\phi) \leq 0.3$ and $p(\phi)$ is related to A and B:
  - Bucket A: $\sum_A p(A)p(B|A)p(\phi|A,B) = p(\phi, B)$ for all B.
  - Bucket B: $\sum_B p(\phi, B) = p(\phi)$.

Agenda

Introduction

Algorithms and approximation methods (for strong extensions)

Sequential decision making

Changing gears: Decision making

- Set of acts $\mathcal{A}$, need to choose one.
  - There are several criteria!
- Γ-*minimax*:

$$\arg \max_{X \in \mathcal{A}} \underline{E}[X] .$$

- *Maximality*: maximal elements of the partial order $\succ$. That is, X is *maximal* if

  there is no $Y \in \mathcal{A}$ such that $E_P[Y - X] > 0$ for all $P \in K$.

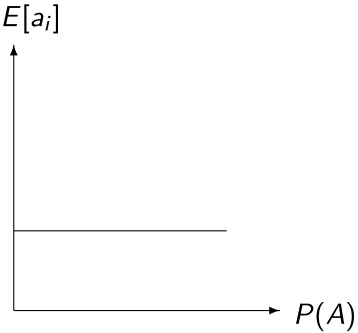
- *E-admissibility*: maximality for at least a distribution. That is, X is *E-admissible* if

  there is $P \in K$ such that $E_P[X - Y] \geq 0$ for all $Y \in \mathcal{A}$.

- Maximax, interval dominance, etc.

Comparing criteria

Three acts: $a_1 = 0.4$; $a_2 = 0/1$ if A/A^c; $a_3 = 1/0$ if A/A^c.



$P(A) \in [0.3, 0.7]$.
Γ-minimax: a_1; Maximal: all of them; E-admissible: $\{a_2, a_3\}$.

Exercise

Credal set $\{P_1, P_2\}$:

$$P_1(s_1) = 1/8, \quad P_1(s_2) = 3/4, \quad P_1(s_3) = 1/8,$$

$$P_2(s_1) = 3/4, \quad P_2(s_2) = 1/8, \quad P_2(s_3) = 1/8,$$

Acts $\{a_1, a_2, a_3\}$:

	s_1	s_2	s_3
a_1	3	3	4
a_2	2.5	3.5	5
a_3	1	5	4.

Which one to select?

Solution

$$P_1(s_1) = 1/8,\ P_1(s_2) = 3/4, ; P_1(s_3) = 1/8, \quad P_2(s_1) = 3/4,\ P_2(s_2) = 1/8,\ P_2(s_3) = 1/8.$$

Acts $\{a_1, a_2, a_3\}$:

	s_1	s_2	s_3
a_1	3	3	4
a_2	2.5	3.5	5
a_3	1	5	4.

Then:

$$\begin{aligned}
E_1[a_1] &= 3/8 + 18/8 + 4/8 = 25/8; \\
E_1[a_2] &= 2.5/8 + 21/8 + 5/8 = 28.5/8; \\
E_1[a_3] &= 1/8 + 15/8 + 4/8 = 35/8. \\
E_2[a_1] &= 18/8 + 3/8 + 4/8 = 25/8; \\
E_2[a_2] &= 15/8 + 3.5/8 + 5/8 = 23.5/8 \\
E_2[a_3] &= 2/8 + 5/8 + 4/8 = 11/8.
\end{aligned}$$

A quick discussion

- Limited to finite set of acts.

- Consider Γ-minimax:
  - Compute $\underline{E}[a_i]$ for each act.
  - Select act with highest $\underline{E}[a_i]$.
- (Considerable minimax theory in Berger's book (1985).)

Maximality

- Find Γ-minimax solution a_0.
- For each other act $a_i \neq a_0$, verify whether

$$E_P[a_0 - a_i] \geq 0;$$

  for all P; if so, discard a_i.
- That is, verify whether

$$\underline{E}[a_0 - a_i] \geq 0.$$

E-admissibility

- For each act a_i:
  - Collect all constraints that must be satisfied by P.
  - Add constraints
    $$E_P[a_i - a_j] \geq 0$$
    for every $a_j \neq a_i$.
  - If all these constraints can be satisfied for some P, then a_i is E-admissible.

- This scheme can be extended to problems with mixed acts (Utkin and Augustin 2005).

Exercise

Credal set $\{P_1, P_2\}$:

$$P_1(s_1) = 1/8, \quad P_1(s_2) = 3/4, \quad P_1(s_3) = 1/8,$$

$$P_2(s_1) = 3/4, \quad P_2(s_2) = 1/8, \quad P_2(s_3) = 1/8,$$

Acts $\{a_1, a_2, a_3\}$:

	s_1	s_2	s_3
a_1	3	3	4
a_2	2.5	3.5	5
a_3	1	5	4.

Which one to select?
And if we take convex hull of credal set?

Solution

$$P_1(s_1) = 1/8,\ P_1(s_2) = 3/4, ; P_1(s_3) = 1/8, \quad P_2(s_1) = 3/4,\ P_2(s_2) = 1/8,\ P_2(s_3) = 1/8.$$

Acts $\{a_1, a_2, a_3\}$:

	s_1	s_2	s_3
a_1	3	3	4
a_2	2.5	3.5	5
a_3	1	5	4.

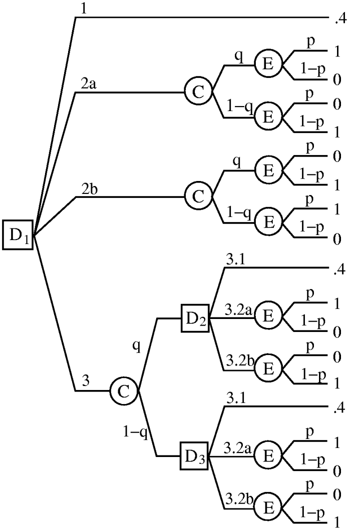
- Consider $P = \alpha P_1 + (1-\alpha)P_2$.
- Then:
$$E_P[a_2 - a_1] = 10\alpha - 3 \geq 0; \quad \alpha \geq 3/10.$$
- And:
$$E_P[a_2 - a_3] = -30\alpha + 17 \geq 0; \quad \alpha \leq 17/30.$$

Challenge: many criteria!

- Γ-minimax, maximality, E-admissibility, etc.
- With independence, we must face the multilinear problems that appear in inference.
- A few models are important: decision trees, influence diagrams, Markov decision processes.

Digression: the challenges of sequential decision making

Teddy Seidenfeld 2004: $p \in [0.25, 0.75]$; $q = 0.5$.



Markov decision processes (MDPs)

- MDPs are quite popular in economics, management and operations research.
- An MDP consists of
  1. A state space S.
  2. An action space A.
  3. Transition probabilities $p_a(r|s) = P_a(s_{t+1} = r|s_t = s)$.
  4. Costs $c_a(s)$.
- Often represented as graphs where nodes are states.
- Another representation: a transition matrix P_a for each action a.

Policies and their costs

- A *policy* specifies an action for each state (possibly indexed by t).
- A *stationary policy* is a policy that does not depend on t.
- A policy π_1 dominates policy π_2 if π_1 has total cost smaller than π_2.
- But how to measure "cost" of a policy?

Costs

- **Additive cost:** just add costs for all transitions.
- **Discounted cost:** add costs, but with discount γ:

$$c(s_0) + \gamma c(s_1) + \gamma^2 c(s_2) + \dots$$

- **Average cost:** add costs, divide by number of transitions.
- **Goal state:** all costs are ignored, what matters is to reach some state.

Discounted cost

- The most popular, and easiest to handle, is discounted cost.
- We must find the optimal policy π^*:

$$\pi^* = \arg\min_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t c_{\pi(s_t)}(s_t)\right].$$

- For discounted cost, the optimal policy always exists (not necessarily true for other costs!).

Basic relation about discounted cost

- Denote by $E[\pi|s]$ the expected cost when the state is s at $t = 0$.
- Then:

$$E[\pi|s] = c_{\pi(s)}(s) + \gamma \sum_{r \in S} p_{\pi(s)}(r|s) E[\pi|r] \, .$$

- How about the optimal policy and the optimal expected cost?

Bellman equation

- Denote by $E^*[s]$
  - the optimal expected cost when the state is s at $t = 0$;
  - called the *value function* (it depends only on s!).
- By dynamic programming we obtain:

$$E^*[s] = \min_{a \in A} \left(c_a(s) + \gamma \sum_{r \in S} p_a(r|s) E^*[r] \right).$$

- From the optimal cost, we obtain:

$$\pi^*(s) = \arg\min_{a \in A} \left(c_a(s) + \gamma \sum_{r \in S} p_a(r|s) E^*[r] \right).$$

Algorithms

1. Linear programming solution: polynomial algorithm, but rarely used.
2. Value iteration.
3. Policy iteration.

...and many variants of these.

Factored representations

- Usually MDPs represent states explicitly.
- However, representations in terms of variables are more compact.
- Factored representations use Bayesian networks to represent $P_a(r|s)$ (a dynamic Bayesian network indexed by actions).
- There are graphical representations for costs and policies as well.

MDPIPs

- A *Markov decision process with imprecise probabilities* consists of:
  1. A state space S.
  2. An action space A.
  3. Transition credal sets $K_a(r|s) = K_a(s_{t+1} = r|s_t = s)$.
  4. Costs $c_a(s)$.
- Proposed in the seventies, analysis restricted to Γ-minimax.
  - Proofs of convergence and stationarity are available.
  - Bellman equation:

$$E^*[s] = \min_{a \in A} \max_{p \in K} \left(c_a(s) + \gamma \sum_{r \in S} p_a(r|s) E^*[r] \right).$$

  - Algorithms: versions of value iteration and policy iteration.
  - Special cases have been used in planning.
- *Factored* MDPIPs use credal networks to represent transitions.

Conclusion

- Independence relations introduce nonlinear constraints.
- Inference is usually solved through optimization (many standard optimization tricks can be applied to these problems).
- For credal networks:
  - 2U is the only pocket of tractability (and Loopy-2U is the most promising approximation scheme).
  - Symbolic variable elimination/bilinear transformation produce exact answers for medium-sized problems.
- Propositional Probabilistic Logic networks can be dealt with the same techniques.
- Sequential decision making usually relies on independence assumptions.
  - There are even controversies about criteria.
  - In simple cases, reduces to inference.
  - Other than that, MDPIPs are ok in some cases.