Decision Support Systems 2, Lecture #2

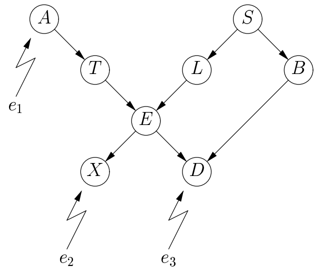
Inference in Discrete Bayesian Networks

by

Uffe Kjærulff <uk@cs.auc.dk>

Inference $\equiv$ Computing Posterior Probabilities

Summation is Commutative

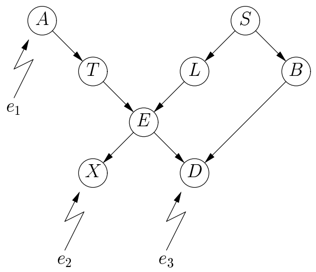


$$\begin{aligned} P(L \mid e) &\propto \sum_B \sum_E \sum_S \sum_T P(D=d \mid B,E) P(X=x \mid E) P(A=a) P(T \mid A=a) P(E \mid T,L) P(S) P(L \mid S) P(B \mid S) \\ &= \sum_E \left(\sum_B P(D=d \mid B,E) P(X=x \mid E) \left(\sum_T P(A=a) P(T \mid A=a) P(E \mid T,L) \right) \left(\sum_S P(S) P(L \mid S) P(B \mid S) \right) \right) \end{aligned}$$

What Happens ?

Tree-Structured Computation

$$P(L \mid e) \propto \sum_E \left(\sum_B P(D=d \mid B,E) P(X=x \mid E) \left(\sum_T P(A=a) P(T \mid A=a) P(E \mid T,L) \right) \left(\sum_S P(S) P(L \mid S) P(B \mid S) \right) \right)$$



$P(T, e_1) = P(A=a) P(T \mid A=a)$

$P(B, L) = \sum_S P(S) P(L \mid S) P(B \mid S)$

$P(E, e_1 \mid L) = \sum_T P(T, e_1) P(E \mid T, L)$

$P(B, L, E, e) = P(E, e_1 \mid L) P(e_2, e_3 \mid B, E) P(B, L)$

$P(e_2, e_3 \mid B, E) = P(D=d \mid B, E) P(e_2 \mid E)$

$P(e_2 \mid E) = P(X=x \mid E)$

$$P(L \mid e) = \frac{\sum_{B,E} P(B,L,E,e)}{\sum_{B,L,E} P(B,L,E,e)}$$

Interpretation of Inference Process

Message Passing in a Junction Tree

$P(T, e_1) = P(A=a) P(T \mid A=a)$

1

$P(B, L) = \sum_S P(S) P(L \mid S) P(B \mid S)$

3

$P(E, e_1 \mid L) = \sum_T P(T, e_1) P(E \mid T, L)$

2

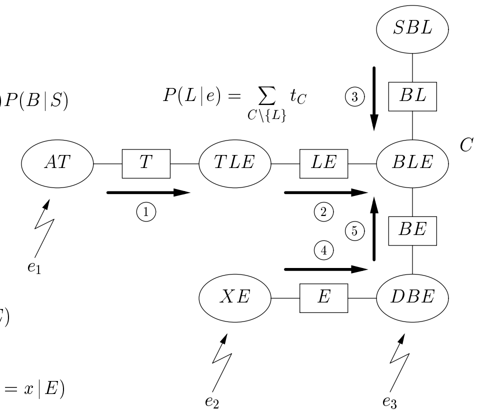
$P(B, L, E, e) = P(E, e_1 \mid L) P(e_2, e_3 \mid B, E) P(B, L)$

5

$P(e_2, e_3 \mid B, E) = P(D=d \mid B, E) P(e_2 \mid E)$

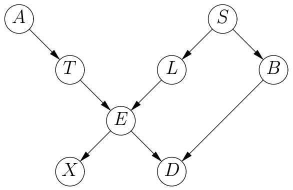
4

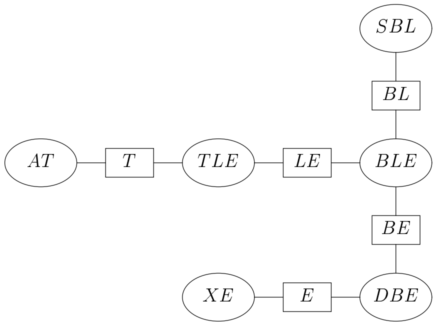
$P(e_2 \mid E) = P(X=x \mid E)$



Making Junction Trees and Propagating in Them

How ?





(1) How do we perform inference in a junction tree?

(2) How do we build a junction tree?

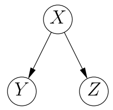
Notation

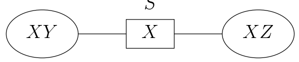
$\mathcal{C}$: set of all clusters of a junction tree

$\mathcal{S}$: set of all separators of a junction tree

t_A : table over $\mathrm{Sp}(A)$

Example:





$\mathcal{C} = \{V, W\}$, $\mathcal{S} = \{S\}$

t_V : table over $\mathrm{Sp}(V) = \mathrm{Sp}(\{X, Y\}) = \mathrm{Sp}(X) \times \mathrm{Sp}(Y) = \{(x_1, y_1), (x_1, y_2), \ldots, (x_n, y_m)\}$

t_S : table over $\mathrm{Sp}(S) = \mathrm{Sp}(X) = \{x_1, \ldots, x_n\}$

t_W : table over $\mathrm{Sp}(W) = \mathrm{Sp}(\{X, Z\}) = \mathrm{Sp}(X) \times \mathrm{Sp}(Z) = \{(x_1, z_1), (x_1, z_2), \ldots, (x_n, z_r)\}$

To each cluster, C, (and each separator, S) of a junction tree is associated a table over the product space of the spaces of the variables of C (S).

Initialisation of a Junction Tree

The cluster tables and separator tables of a junction tree are initialised as follows.

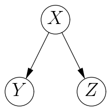
(1) For each $C \in \mathcal{C}$ let $t_C = \mathbf{1}$

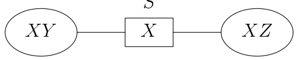
(2) For each $S \in \mathcal{S}$ let $t_S = \mathbf{1}$

(3) For each variable $A \in U$ choose exactly one cluster, C, containing $\mathrm{pa}(A) \cup \{A\}$ and let

$$t_C \leftarrow t_C * P(A \mid \mathrm{pa}(A))$$

Example:



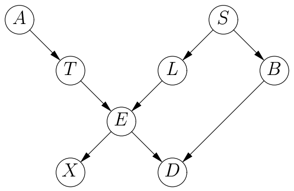


$t_V = P(X)P(Y \mid X)$ $\quad t_S = \mathbf{1}$ $\quad t_W = P(Z \mid X)$

$$P(U) = P(X)P(Y \mid X)P(Z \mid X) \qquad t_U = \frac{t_V \cdot t_W}{t_S} = P(X)P(Y \mid X)P(Z \mid X) = P(U)$$

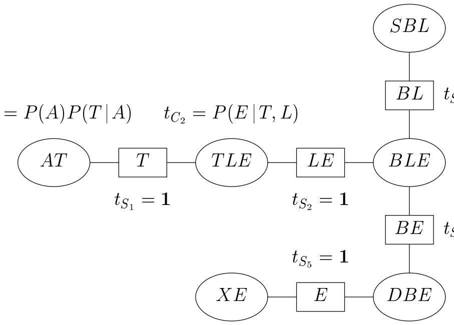
Initialisation of a Junction Tree

Example: Asia



$t_{C_3} = P(S)P(L \mid S)P(B \mid S)$

$t_{C_1} = P(A)P(T \mid A)$ $\quad t_{C_2} = P(E \mid T, L)$



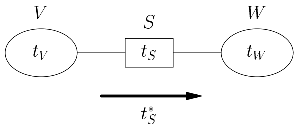
$t_{S_3} = \mathbf{1}$ $\quad t_{C_4} = \mathbf{1}$ $\quad t_{S_1} = \mathbf{1}$ $\quad t_{S_2} = \mathbf{1}$ $\quad t_{S_4} = \mathbf{1}$ $\quad t_{S_5} = \mathbf{1}$

$t_{C_6} = P(X \mid E)$ $\quad t_{C_5} = P(D \mid B, E)$

$$\begin{aligned} P(U) &= P(A)P(T \mid A)P(S)P(L \mid S)P(B \mid S) \\ &\quad P(E \mid T, L)P(D \mid B, E)P(X \mid E) \end{aligned}$$

$$t_U = \frac{\prod_{i=1}^{6} t_{C_i}}{\prod_{i=1}^{5} t_{S_i}} = \prod_{i=1}^{6} t_{C_i} = P(U)$$

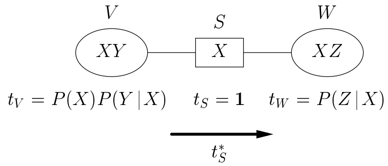
Message Absorption



1. $t_S^* = \sum_{V \setminus S} t_V$ (compute message)
2. $t_W \leftarrow t_W \dfrac{t_S^*}{t_S}$ (W absorbs message)
3. $t_S \leftarrow t_S^*$ (store message in S)

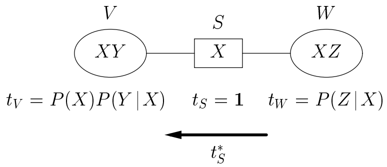
Message Absorption

Example



1. $t_S^* = \sum_Y t_V = \sum_Y P(X,Y) = P(X)$
2. $t_W \leftarrow t_W \dfrac{t_S^*}{t_S} = P(Z \mid X)\dfrac{P(X)}{\mathbf{1}} = P(X,Z)$
3. $t_S \leftarrow t_S^*$

Now t_V, t_S, t_W contain true marginals!



1. $t_S^* = \sum_Z t_W = \sum_Z P(Z \mid X) = \mathbf{1}$
2. $t_V \leftarrow t_V \dfrac{t_S^*}{t_S} = t_V \dfrac{\mathbf{1}}{\mathbf{1}} = t_V$
3. $t_S \leftarrow t_S^*$ (NB: $t_S^* = t_S$)

Nothing changed!

Message Absorption

Consistence

The link between V and W is *consistent* if

$$\sum_{V \setminus S} t_V = t_S = \sum_{W \setminus S} t_W.$$

Then absorption doesn't change anything. If all links of a junction tree, T, are consistent, then T is consistent.

Assume that the link is consistent, but now some evidence changes t_V to t_V^*. Then after W has absorbed from V, the three tables all contain V's new information on S:

$$\sum_{W \setminus S} t_W^* = \sum_{W \setminus S} t_W \frac{t_S^*}{t_S} = \frac{t_S^*}{t_S} \sum_{W \setminus S} t_W = \frac{t_S^*}{t_S} t_S = t_S^* = \sum_{V \setminus S} t_V^*.$$

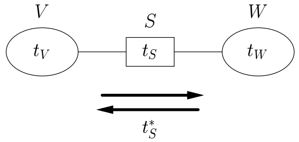
Or, shorter

$$\sum_{W \setminus S} t_W^* = t_S^* = \sum_{V \setminus S} t_V^*,$$

i.e., absorption makes the link consistent.

Message Absorption

Supportiveness



W can only absorb from V (and vice versa) if

$$t_S^*(x) > 0 \Rightarrow t_S(x) > 0 \quad \text{for all } x \in \mathrm{Sp}(S).$$

If this requirement is fulfilled, the link is *supportive*. Supportiveness is preserved under absorption, since

$$t_W^* = t_W \frac{t_S^*}{t_S}, \quad \text{where} \quad t_S^* = \sum_{V \setminus S} t_V.$$

That is, any zero entry in t_S^* is also a zero entry in t_W^*, or, more precisely,

$$t_S^*(x_S) = 0 \Rightarrow t_W^*(x_W) = 0$$

for each $x_W \in \mathrm{Sp}(W)$ for which $x_S \in \mathrm{Sp}(S)$ is a marginal entry of x_W with respect to S (i.e., $x_S = (x_W)_S$).

Message Absorption

Invariance of t_U

$t_U = \frac{\prod_{C\in\mathcal{C}} t_C}{\prod_{S\in\mathcal{S}} t_S}$ is invariant under absorption, since

$$t_U^* = \frac{\prod_{C\in\mathcal{C}} t_C^*}{\prod_{S\in\mathcal{S}} t_S^*} = \frac{\prod_{C\in\mathcal{C}\setminus\{W\}} t_C}{\prod_{S\in\mathcal{S}\setminus\{S'\}} t_S} \cdot \frac{t_W^*}{t_{S'}^*} = \frac{\prod_{C\in\mathcal{C}\setminus\{W\}} t_C}{\prod_{S\in\mathcal{S}\setminus\{S'\}} t_S} \cdot \frac{t_W \frac{t_{S'}^*}{t_{S'}}}{t_{S'}^*} = \frac{\prod_{C\in\mathcal{C}\setminus\{W\}} t_C}{\prod_{S\in\mathcal{S}\setminus\{S'\}} t_S} \cdot \frac{t_W}{t_{S'}} = \frac{\prod_{C\in\mathcal{C}} t_C}{\prod_{S\in\mathcal{S}} t_S} = t_U,$$

when W absorbs from V via S'.

In essence,

$$\frac{t_W^*}{t_{S'}^*} = \frac{t_W \frac{t_{S'}^*}{t_{S'}}}{t_{S'}^*} = \frac{t_W}{t_{S'}}.$$

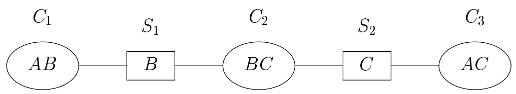
Junction Tree Property

Motivation

If T is a consistent cluster tree, then if $A \in V$ and $A \in W$,

$$\sum_{V\setminus\{A\}} t_V = \sum_{W\setminus\{A\}} t_W \qquad (*)$$

if V and W are neighbours in T. If T is given by



there is, however, no guarantee that $(*)$ is fulfilled for C_1, C_3! For example, information about A in C_1 cannot be communicated to C_3!

Therefore we need *global consistence*, i.e.,

$$\sum_{V\setminus I} t_V = \sum_{W\setminus I} t_W, \qquad I = V \cap W$$

for *any* pair of clusters V, W.

Junction Tree Property

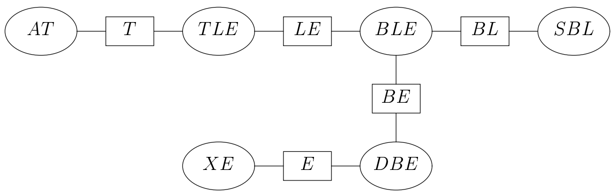
Definition

To guarantee global consistence, we must require the *junction tree property* to hold:

> A cluster tree is a *junction tree* if for each pair of clusters V, W, all clusters on the path between V and W contain $V \cap W$.

A consistent junction tree is globally consistent

Example

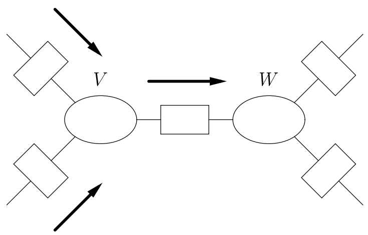


Message Passing

When ?

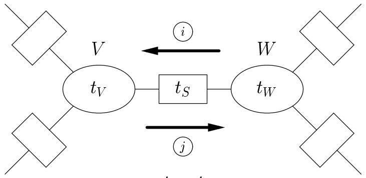
"W absorbs a message from V" $\equiv$ "V sends (or passes) a message to W"

V can send exactly one message to W, but only when it has received a message from each of its other neighbours



Message Passing

Consistence



$i < j$

Before message i : t_V t_S t_W^* (W can have received messages from other neighbours)

After message i : t_V^* t_S^* t_W^* $\left(t_S^* = \sum_{W \setminus S} t_W^*, \quad t_V^* = t_V \dfrac{t_S^*}{t_S}\right)$

Before message j : t_V^{**} t_S^* t_W^* (W hasn't reveived further messages)

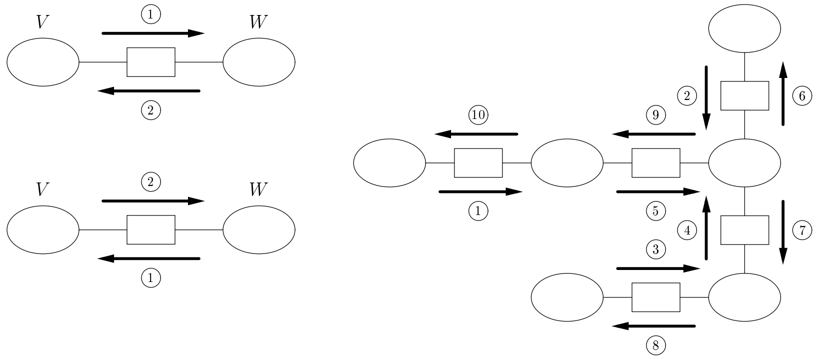
After message j : t_V^{**} t_S^{**} t_W^{**} $\left(t_S^{**} = \sum_{V \setminus S} t_V^{**}, \quad t_W^{**} = t_W \dfrac{t_S^{**}}{t_S^*}\right)$

When both messages have been sent, the link is consistent:

$$\sum_{W \setminus S} t_W^{**} = \sum_{W \setminus S} t_W^* \frac{t_S^{**}}{t_S^*} = \frac{t_S^{**}}{t_S^*} \sum_{W \setminus S} t_W^* = \frac{t_S^{**}}{t_S^*} t_S^* = t_S^{**} = \sum_{V \setminus S} t_V^{**}$$

Message Scheduling

Examples

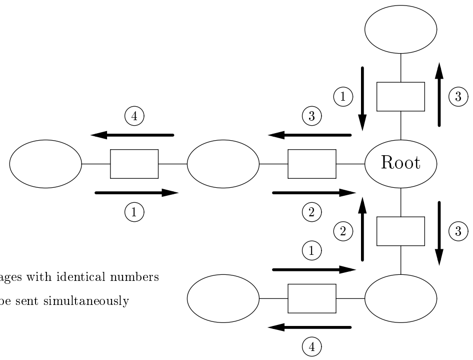


NB: When a full schedule has been executed, all links are consistent (i.e., the tree is consistent).

Inward/outward message passing

The Hugin Approach

Performing an inward pass (from the leaves towards a root cluster) followed by an outward pass (from the root cluster towards the leaves) provides an easy-to-implement message scheduling algorithm.



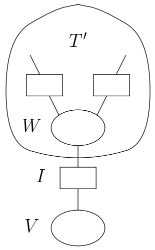
Messages with identical numbers may be sent simultaneously

$\mathbf{t_V = \sum_{U \setminus V} t_U}$ in a consistent junction tree

Let T be a consistent junction tree over variables U, and let $t_U = \dfrac{\prod_{C \in \mathcal{C}} t_C}{\prod_{S \in \mathcal{S}} t_S}$. Then $t_V = \sum_{U \setminus V} t_U$, $V \in \mathcal{C}$.

Proof: Induction on the number of clusters, n. Holds for $n = 1$. Assume it holds for $n \le k$, and let $n = k + 1$ and $U' = U \setminus (V \setminus I)$ (see figure). Then

$$t_U = \frac{\prod_{C \in \mathcal{C} \setminus \{V\}} t_C}{\prod_{S \in \mathcal{S} \setminus \{I\}} t_S} \cdot \frac{t_V}{t_I} = t_{U'} \cdot \frac{t_V}{t_I}.$$



Consistence of T gives

$$\sum_{V \setminus I} t_V = t_I = \sum_{W \setminus I} t_W,$$

hence

$$\sum_{V \setminus I} t_U = \sum_{V \setminus I} t_{U'} \cdot \frac{t_V}{t_I} = t_{U'} \cdot \frac{\sum_{V \setminus I} t_V}{t_I} = t_{U'}.$$

Therefore, by the induction hypothesis, for all C in T'

$$\sum_{U \setminus C} t_U = \sum_{U' \setminus C} \sum_{V \setminus I} t_U = \sum_{U' \setminus C} t_{U'} = t_C \quad \text{and} \quad \sum_{U \setminus V} t_U = \sum_{U' \setminus I} t_{U'} \cdot \frac{t_V}{t_I} = \frac{t_V}{t_I} \sum_{U' \setminus I} t_{U'} = \frac{t_V}{t_I} t_I = t_V. \quad \square$$

A consistent junction tree contains true marginals

Let T be a consistent junction tree over variables U, and let $P(U)$ be the joint probability distribution for U given by the corresponding Bayesian network. Then for each $C \in \mathcal{C}$

$$t_C = \sum_{U \setminus C} P(U) = P(C).$$

Proof: Before propagation we have

$$t_U = \frac{\prod_{C \in \mathcal{C}} t_C}{\prod_{S \in \mathcal{S}} t_S} = \prod_{C \in \mathcal{C}} t_C = P(U).$$

Since t_U is invariant under message passing, i.e.,

$$t_U = t_U^* = \frac{\prod_{C \in \mathcal{C}} t_C^*}{\prod_{S \in \mathcal{S}} t_S^*} = P(U)$$

after propagation (execution of a full message schedule), and $t_C = \sum_{U \setminus C} t_U$, the result follows. □

... also when there's evidence

$$P(U, e) = \prod_{A \in U} P(A \mid \mathrm{pa}(A)) * \mathcal{E}_A = \prod_{A \in U} P^*(A \mid \mathrm{pa}(A)),$$

where

$$\mathcal{E}_A(a) = \begin{cases} 1 & \text{if } A = a \text{ or if there is no evidence on } A \\ 0 & \text{otherwise.} \end{cases}$$

Before propagation,

$$t_U = \frac{\prod_{C \in \mathcal{C}} t_C}{\prod_{S \in \mathcal{S}} t_S} = \prod_{C \in \mathcal{C}} t_C = \prod_{A \in U} P^*(A \mid \mathrm{pa}(A)) = P(U, e).$$

After propagation,

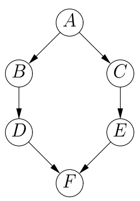
$$t_C = \sum_{U \setminus C} P(U, e) = P(C, e),$$

using the preceding result.

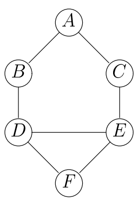
Construction of Junction Trees

Chordless Cycles Must be Eliminated

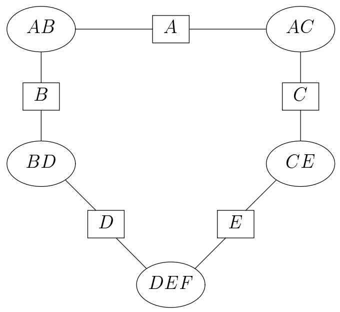
Any method for exact inference in Bayesian networks implicitly or explicitly performs a triangulation of the moral graph: Chordless cycles of length 4 or greater in the moral graph yield indissoluble cycles in the junction graph.



DAG



Moral graph

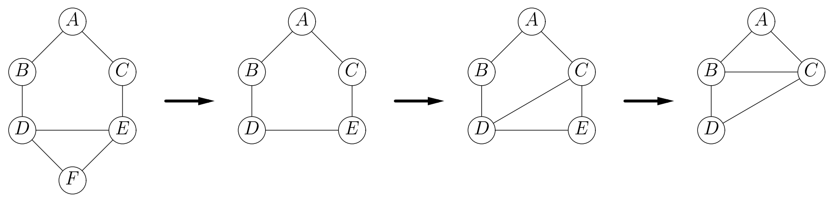


Junction graph

Elimination of chordless cycles

An Example

All cycles of length 4 or greater including F has chord D—E $\Longrightarrow$ Dissolution of chordless cycles of length 4 or greater can be performed in the subgraph obtained by removing F



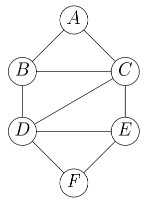
Eliminate F Add C—D Elim. E and add B—C No chordless cycles

An undirected graph without chordless cycles of length 4 or greater is called a *triangulated* graph

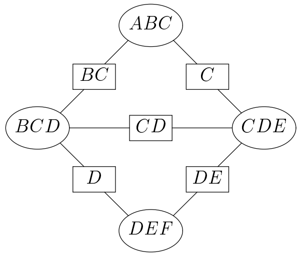
Chordality ⇔ Junction Graph has a Junction Tree

An undirected graph is triangulated if and only if its junction graph has a junction tree.

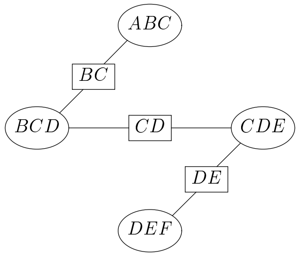
Example:



Triangulated graph
Elim. seq.: F, E, D, C, B, A

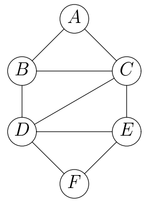


Junction graph

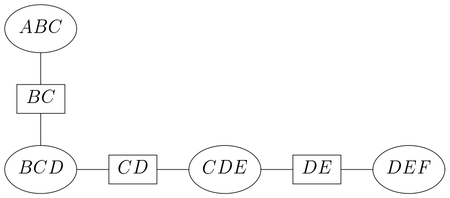


Junction tree

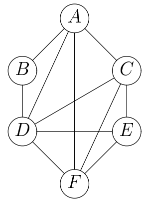
(Near-)Optimal Elimination Sequences are Important



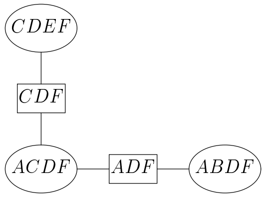
Elim. seq.: F, E, D, C, B, A



Cost: $4 \times 5^3 = 500$ with $|\mathrm{Sp}(A)| = \cdots = |\mathrm{Sp}(F)| = 5$



Elim. seq.: E, C, A, B, D, F



Cost: $3 \times 5^4 = 1875$ with $|\mathrm{Sp}(A)| = \cdots = |\mathrm{Sp}(F)| = 5$

From Chain Graph to Junction Tree

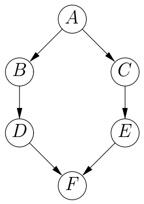
(1) Moralise chain graph.

(2) Triangulate moral graph (find good elimination sequence)

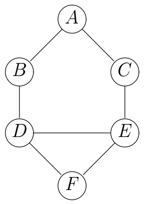
(3) Identify the cliques (maximal complete subgraphs) of the triangulated graph.

(4a) Create junction tree: successively add a link between two cliques whose intersection is maximal, and such that no cycles are created (Kruskal's algorithm), or

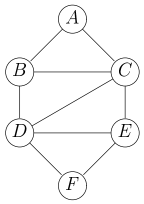
(4b) start with a tree containing a single clique and successively add a link between a clique of the tree and a clique outside the tree whose intersection is maximal (Prim's algorithm).



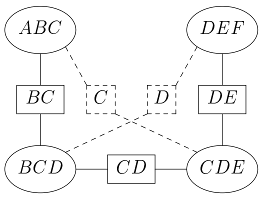
Chain graph (DAG)



Moral graph



Triangulated graph



Junction tree

Max-Marginalization

By substituting for the usual (sum-)marginalization operation the max-marginalization operation

$$t_U^{\max}(y) = \mathop{\mathsf{M}}_{V \setminus U} t_V(z) = \max_{x \in \mathrm{Sp}(V \setminus U)} t_V(x, y),$$

where $x \in \mathrm{Sp}(V \setminus U)$, $y \in \mathrm{Sp}(U)$, and $z = (x, y) \in \mathrm{Sp}(V)$, we compute the most probable configuration of the variables in the marginal set.

Example: Let $V = \{A, B, C\}$ be a set of binary variables, and let $U = \{A, B\}$.

	b_1	b_1	b_2	b_2
	c_1	c_2	c_1	c_2
a_1	0.10	0.05	0.17	0.09
a_2	0.07	0.12	0.27	0.13

t_V

	b_1	b_2
a_1	0.10	0.17
a_2	0.12	0.27

$t_U^{\max}$

That is, configuration (a_2, b_2) is the most probable configuration in subspace $\mathrm{Sp}(U)$. Obviously, $t_V \equiv t_V^{\max}$.

Max-Potentials Provide a Valid Potential Representation

Having performed max-propagation (i.e., propagation using the max-marginalization operation), we have

$$t_V = t_V^{\max} = \frac{\prod_{C \in \mathcal{C}} t_C^{\max}}{\prod_{S \in \mathcal{S}} t_S^{\max}}.$$

Consequently, the sum-margins will be reestablished by a subsequent sum-propagation.

Finding the Most Probable Configuration

When max-propagation has been performed the most probable configuration of all the variables may be found by

FIND-MAX-CONFIG(C, P, t_S^*)

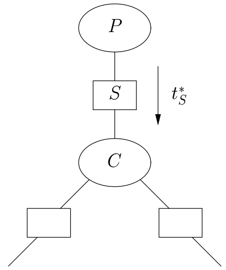
$t_C \leftarrow t_C \dfrac{t_S^*}{t_S}$

$x_C^{\max} = \arg \max_{x \in \mathrm{Sp}(C)} t_C(x)$

Let $t_C(x) \leftarrow 0$ for all $x \neq x_C^{\max}$

For $N \in \mathrm{Neighbours}(C) \setminus \{P\}$ do

FIND-MAX-CONFIG(N, C, $\mathop{\mathsf{M}}\limits_{C \setminus N} t_C$)



After execution of FIND-MAX-CONFIG(R), where R is a root in the junction tree, all variables have been instantiated, and the joint instantiation equals $x_V^{\max} = \mathop{\mathsf{M}}\limits_{V} t_V$, the most probable configuration of all the variables, V.

Drawing a Random Configuration From $P(V)$

A random configuration can be drawn from the joint distribution, $P(V)$ (given by the joint table t_V), through local operations on the junction tree. Assuming that the junction tree is in *sum-equilibrium*, the following procedure will generate a random sample (possibly given some evidence):

FIND-RANDOM-CONFIG(C, P, t_S^*)

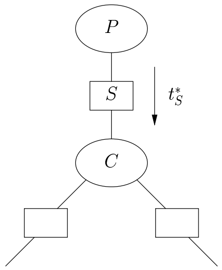
$t_C \leftarrow t_C \dfrac{t_S^*}{t_S}$

$x_C^* = \arg \operatorname*{random}_{x \in \mathrm{Sp}(C)} t_C(x)$

Let $t_C(x) \leftarrow 0$ for all $x \neq x_C^*$

For $N \in \mathrm{Neighbours}(C) \setminus \{P\}$ do

FIND-RANDOM-CONFIG(N, C, $\sum_{C \setminus N} t_C$)



After execution of FIND-RANDOM-CONFIG(R), where R is a root in the junction tree, all variables have been instantiated, and the joint instantiation is drawn from $P(V)$.

Summary

- Basic feature: Summation is commutative
- Initialization of a junction tree
- Message absorption
- Invariance of t_V
- Junction tree property
- Message scheduling (including the inward/outward approach)
- Construction of junction trees
  - Moralization
  - Triangulation
  - Construction (Kruskal's or Prim's algorithm)
- Max-propagation
- Simulation

Any inference exact method for inference in a Bayesian network can be formulated in terms of message passing in a junction tree.

Further Issues

- Triangulation algorithms
- Computing arbitrary marginals
- Conflict analysis
- Sensitivity analysis
- Value of information
- Learning probabilities from data base of cases
- Learning structure from data base of cases
- Solving troubleshooting problems using Bayesian networks
- Influence diagrams (decision networks)
  - Large range of issues
- Etc.

Other Inference Methods

- Other architectures for exact inference based (explicitly) on message passing in a junction tree:
  - Shafer-Shenoy architecture (Shafer & Shenoy 1989)
  - Lauritzen-Spiegelhalter architecture (Lauritzen & Spiegelhalter 1988)
- Other exact methods for inference include (among many others):
  - Bucket elimination (Dechter 1998)
  - Arc-reversal (Shachter 1989)
  - Lazy propagation (Madsen & Jensen 1999)
- Simulation-based methods for inference include (among many others):
  - Gibbs sampling (Geman & Geman 1984)
  - Blocking Gibbs sampling (Jensen, Kong & Kjærulff 1995)
  - Logic sampling (forward sampling with rejection) (Henrion 1988)
  - Likelihood-weighted sampling (Shachter & Peot 1990)

References

Dechter, R. (1998). Bucket elimination: A unifying framework for probabilistic inference, *in* M. I. Jordan (ed.), *Learning in Graphical Models*, NATO ASI SERIES, Kluwer Academic Publishers.

Geman, S. & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**(6): 721–741.

Henrion, M. (1988). Propagating uncertainty in Bayesian networks by probabilistic logic sampling, *in* J. F. Lemmer & L. M. Kanal (eds), *Uncertainty in Artificial Intelligence 2*, Elsevier Science Publishers B. V. (North-Holland), Amsterdam, pp. 149–163.

Jensen, C. S., Kong, A. & Kjærulff, U. (1995). Blocking-Gibbs sampling in very large probabilistic expert systems, *International Journal of Human-Computer Studies* **42**: 647–666.

Lauritzen, S. L. & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems, *Journal of the Royal Statistical Society, Series B* **50**(2): 157–224.

Madsen, A. L. & Jensen, F. V. (1999). Lazy propagation: A junction tree algorithm based on lazy evaluation, *Artificial Intelligence* . To appear.

Shachter, R. D. (1989). Evidence absorption and propagation through evidence reversals, *Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence*, Association for Uncertainty in Artificial Intelligence, pp. 303–310.

Shachter, R. D. & Peot, M. A. (1990). Simulation approaches to general probabilistic inference on belief networks, *in* M. Henrion, R. D. Shachter, L. M. Kanal & J. F. Lemmer (eds), *Uncertainty in Artificial Intelligence 5*, Elsevier Science Publishers B. V. (North-Holland), Amsterdam, pp. 221–231.

Shafer, G. & Shenoy, P. P. (1989). Probability propagation, *Proceedings of the Second International Workshop on Artificial Intelligence*.