

Exact Inference Algorithms

Bucket-elimination



COMPSCI 276, Spring 2018

Class 4: Rina Dechter

(Reading: **Dechter chapters 4**, Darwiche chapter 6, **Dechter Section 3.4**)



Inference for probabilistic networks

- Bucket elimination
 - Belief-updating, $P(e)$, partition function
 - Marginals, probability of evidence
 - The impact of evidence
 - for MPE (\rightarrow MAP)
 - for MAP (\rightarrow Marginal Map)
 - Mixed networks
- Tree-decomposition schemes
 - Bucket tree elimination
 - Cluster tree elimination

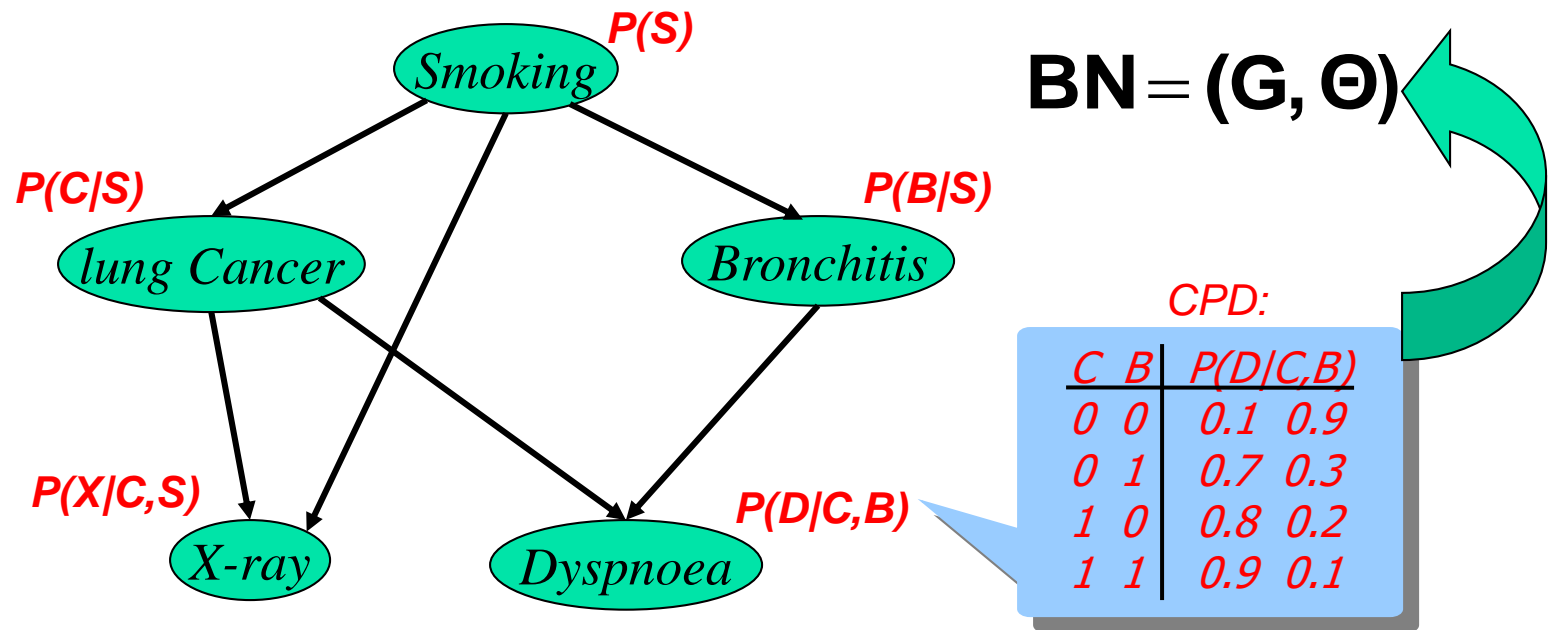


Inference for probabilistic networks

- Bucket elimination
 - Belief-updating, $P(e)$, partition function
 - Marginals, probability of evidence
 - The impact of evidence
 - for MPE (\rightarrow MAP)
 - for MAP (\rightarrow Marginal Map)
 - Mixed networks
- Tree-decomposition schemes
 - Bucket tree elimination
 - Cluster tree elimination

Bayesian Networks: Representation

(Pearl, 1988)

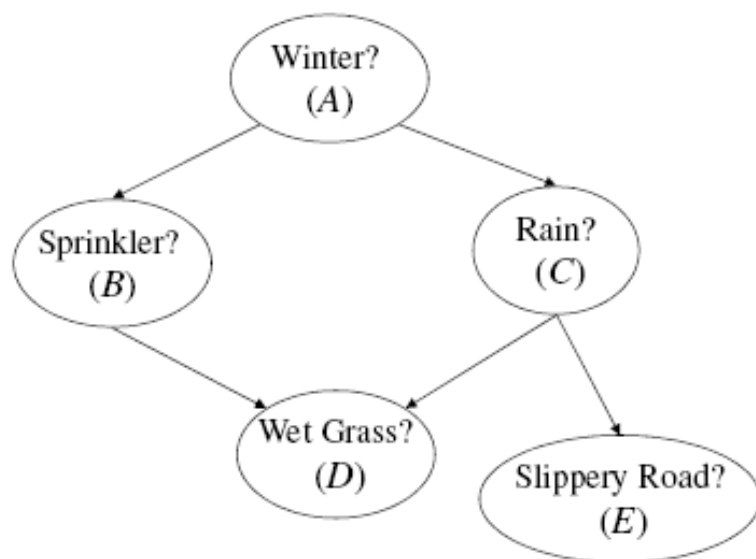


$$P(S, C, B, X, D) = P(S) P(C/S) P(B/S) P(X/C,S) P(D/C,B)$$

Belief Updating:

$P(\text{lung cancer}=\text{yes} \mid \text{smoking}=\text{no}, \text{dyspnoea}=\text{yes}) = ?$

A Bayesian Network



A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	C	$\Theta_{C A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	C	D	$\Theta_{D BC}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Graphical Models

A **graphical model** consists of:

$X = \{X_1, \dots, X_n\}$ -- variables

$D = \{D_1, \dots, D_n\}$ -- domains

$F = \{f_{\alpha_1}, \dots, f_{\alpha_m}\}$ -- functions or "factors"

Operators:

combination operator
(sum, product, join, ...)

elimination operator
(projection, sum, max, min, ...)

Types of queries:

Marginal:

$$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$$

MPE / MAP:

$$f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$$

Marginal MAP:

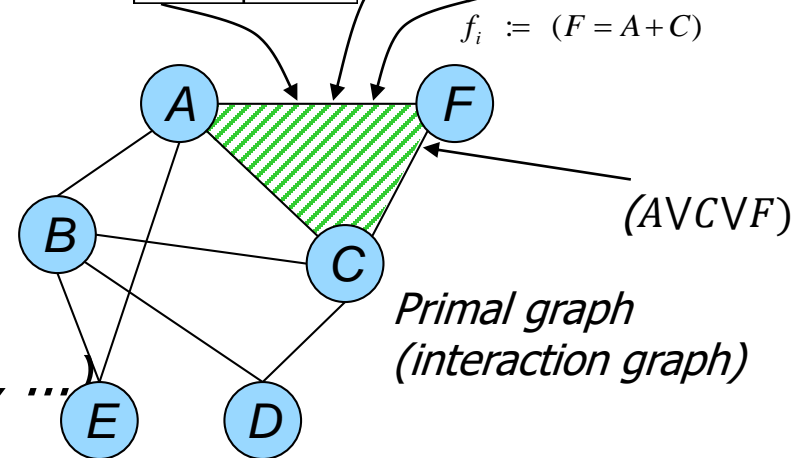
$$f(\mathbf{x}_M^*) = \max_{\mathbf{x}_M} \sum_{\mathbf{x}_S} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$$

Conditional Probability Table (CPT)

A	C	F	P(F A,C)
0	0	0	0.14
0	0	1	0.96
0	1	0	0.40
0	1	1	0.60
1	0	0	0.35
1	0	1	0.65
1	1	0	0.72
1	1	1	0.68

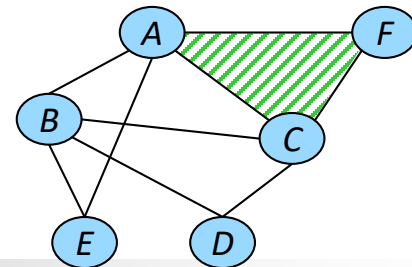
Relation

A	C	F
red	green	blue
blue	red	red
blue	blue	green
green	red	blue

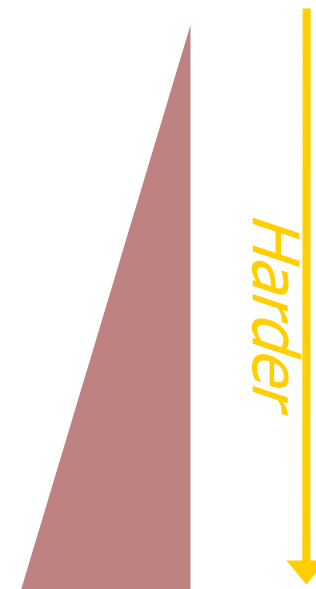


- All these tasks are NP-hard
 - exploit problem structure
 - identify special cases
 - approximate

Types of queries



▶ Max-Inference	$f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$
▶ Sum-Inference	$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$
▶ Mixed-Inference	$f(\mathbf{x}_M^*) = \max_{\mathbf{x}_M} \sum_{\mathbf{x}_S} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$



- **NP-hard**: exponentially many terms
- We will focus on **approximation** algorithms
 - **Anytime**: very fast & very approximate ! Slower & more accurate



Belief Updating is NP-hard

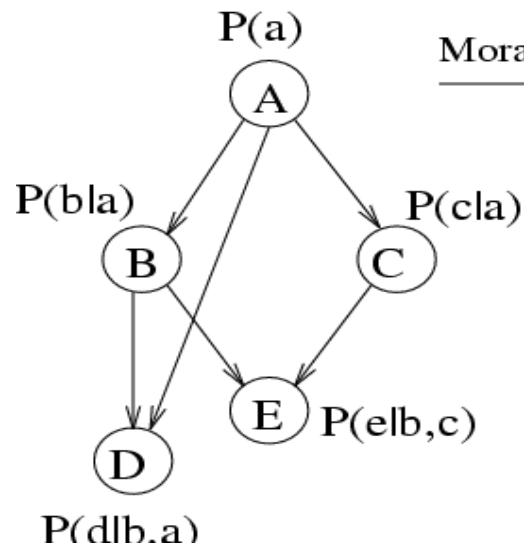
- Each SAT formula can be mapped into a belief updating query in a Bayesian network

- Example

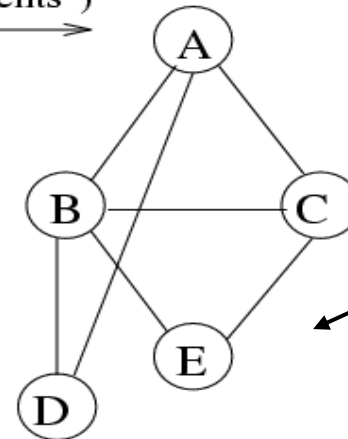
$$(\neg u \vee \neg w \vee y) \wedge (u \vee \neg v \vee w)$$

"Moral" Graph

$$P(X_1, \dots, X_n) = \prod_{i=1}^n \underbrace{P(X_i \mid \text{parents}(X_i))}_{\text{CPD}}$$



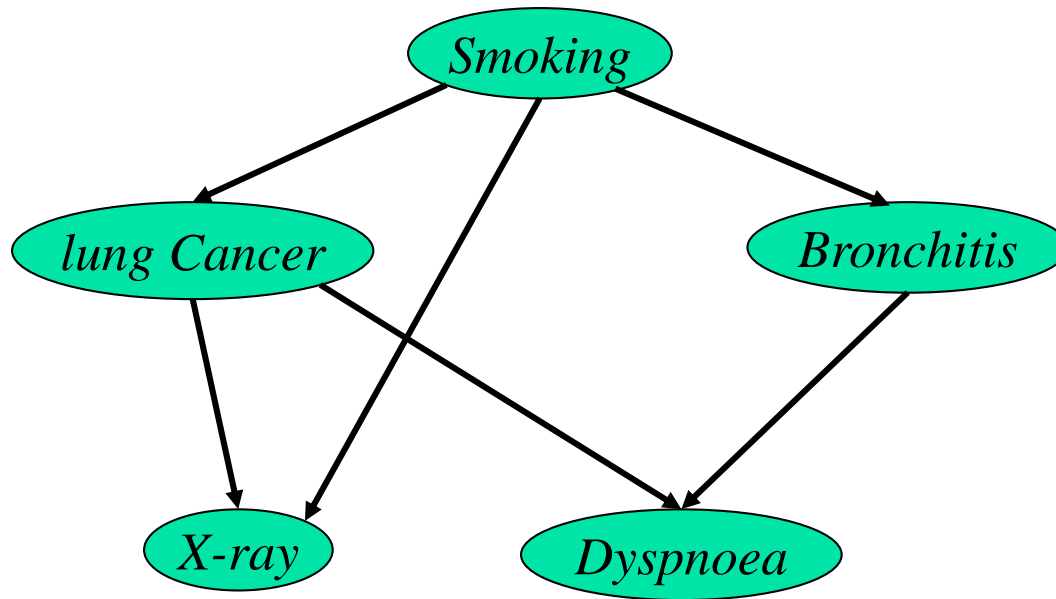
Moralize ("marry parents")



*Conditional
Probability
Distribution
(CPD)*

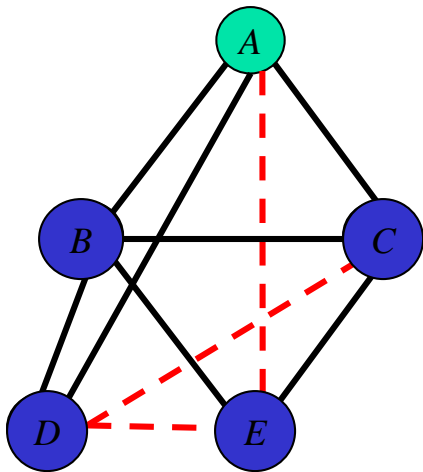
*Clique in
moral graph ("family")*

Belief Updating



$P(\text{lung cancer}=\text{yes} \mid \text{smoking}=\text{no}, \text{dyspnoea}=\text{yes}) = ?$

Belief updating: $P(X|\text{evidence})=?$



"Moral" graph

$$P(a/e=0) \propto P(a, e=0) =$$

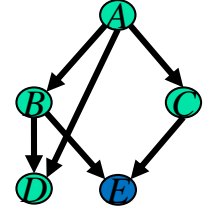
$$\sum_{e=0, d, c, b} P(a) \underbrace{P(b/a)} \underbrace{P(c/a) P(d/b, a) P(e/b, c)} =$$

$$P(a) \sum_{e=0} \sum_d \sum_c P(c/a) \underbrace{\sum_b P(b/a) P(d/b, a) P(e/b, c)}_{h^B(a, d, c, e)}$$

Variable Elimination

Bucket elimination

Algorithm *BE-bel* (Dechter 1996)



$$P(A | E = 0) = \alpha \sum_{E=0, D, C, B} P(A) \cdot P(B | A) \cdot P(C | A) \cdot P(D | A, B) \cdot P(E | B, C)$$

$\sum_b \prod$ ← Elimination operator

bucket B:

$$P(b/a) \quad P(d/b, a) \quad P(e/b, c)$$

bucket C:

$$P(c/a) \quad \lambda^B(a, d, c, e)$$

bucket D:

$$\lambda^C(a, d, e)$$

bucket E:

$$e=0 \quad \lambda^D(a, e)$$

bucket A:

$$P(a) \quad \lambda^E(a)$$

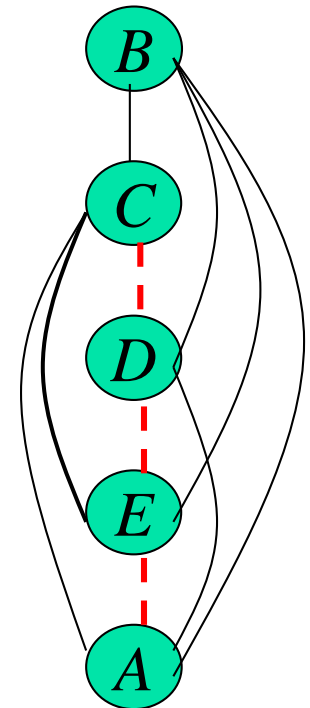
"induced width"
(max clique size)

$$W^*=4$$

$$P(e=0)$$

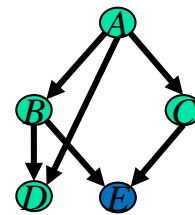
$$P(a, e=0)$$

$$P(a|e=0) = \frac{P(a, e=0)}{P(e=0)}$$



Bucket Elimination

Algorithm BE-bel [Dechter 1996]



$$p(A|E = 0) = \alpha \sum_{e,d,c,b} p(A) p(b|A) p(c|A) p(d|A, b) p(e|b, c) \mathbb{1}[e = 0]$$

$\sum_b \prod$ ← Elimination & combination operators

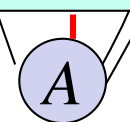
Time and space exponential in the induced-width / treewidth

bucket A:

$p(A)$

$\lambda_{E \rightarrow A}(A)$

induced width
(max clique size)



$p(E = 0)$

$$p(A|E = 0) = p(A, E = 0) / p(E = 0)$$



The Operation In a Bucket

- Multiplying functions
- Marginalizing (summing-out) functions

Combination of Cost Functions

A	B	f(A,B)
b	b	0.4
b	g	0.1
g	b	0
g	g	0.5

B	C	f(B,C)
b	b	0.2
b	g	0
g	b	0
g	g	0.8

A	B	C	f(A,B,C)
b	b	b	0.1
b	b	g	0
b	g	b	0
b	g	g	0.08
g	b	b	0
g	b	g	0
g	g	b	0
g	g	g	0.4

$$= 0.1 \times 0.8$$

Factors: Sum-Out Operation

The result of **summing out** variable X from factor $f(\mathbf{X})$ is another factor over variables $\mathbf{Y} = \mathbf{X} \setminus \{X\}$:

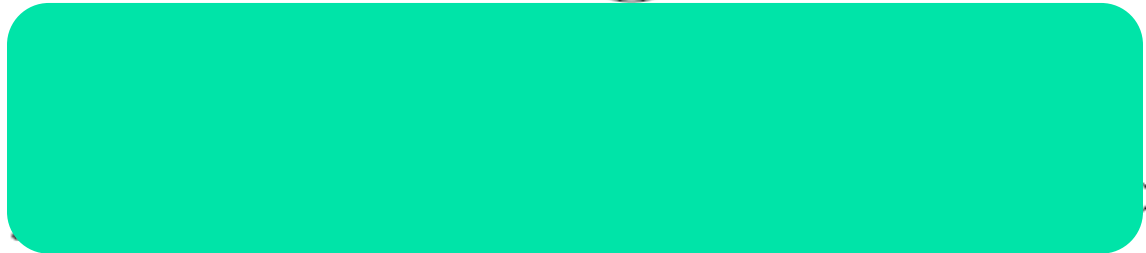
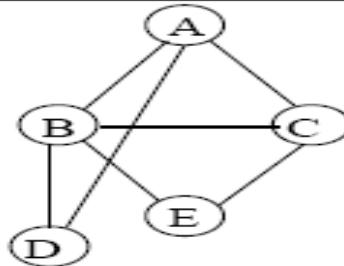
$$\left(\sum_X f \right) (\mathbf{y}) \stackrel{\text{def}}{=} \sum_x f(x, \mathbf{y})$$

B	C	D	f_1
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

B	C	$\sum_D f_1$
true	true	1
true	false	1
false	true	1
false	false	1

	$\sum_B \sum_C \sum_D f_1$
\top	4

Bucket Elimination and Induced Width



Ordering: a, e, d, c, b

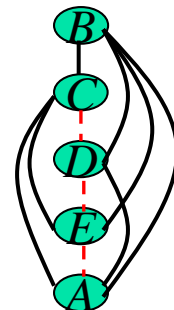
$$\text{bucket}(B) = P(e|b, c), P(d|a, b), P(b|a)$$

$$\text{bucket}(C) = P(c|a) \parallel \lambda_B(a, c, d, e)$$

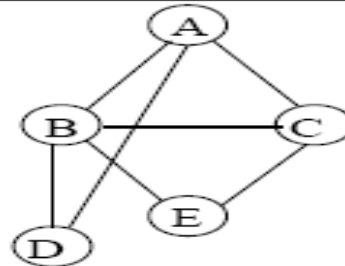
$$\text{bucket}(D) = \parallel \lambda_C(a, d, e)$$

$$\text{bucket}(E) = e = 0 \parallel \lambda_D(a, c)$$

$$\text{bucket}(A) = P(a) \parallel \lambda_E(a)$$

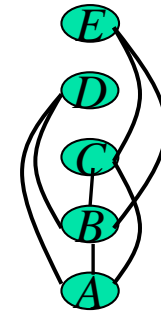


Bucket Elimination and Induced Width



Ordering: a, b, c, d, e

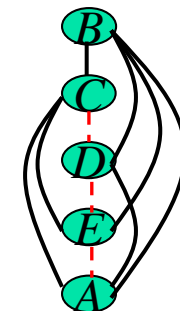
$bucket(E) = P(e|b, c), e = 0$
 $bucket(D) = P(d|a, b)$
 $bucket(C) = P(c|a) \parallel P(e = 0|b, c)$
 $bucket(B) = P(b|a) \parallel \lambda_D(a, b), \lambda_C(b, c)$
 $bucket(A) = P(a) \parallel \lambda_B(a)$



$W^*=2$

Ordering: a, e, d, c, b

$bucket(B) = P(e|b, c), P(d|a, b), P(b|a)$
 $bucket(C) = P(c|a) \parallel \lambda_B(a, c, d, e)$
 $bucket(D) = \parallel \lambda_C(a, d, e)$
 $bucket(E) = e = 0 \parallel \lambda_D(a, c)$
 $bucket(A) = P(a) \parallel \lambda_E(a)$



$W^*=4$

ALGORITHM BE-BEL

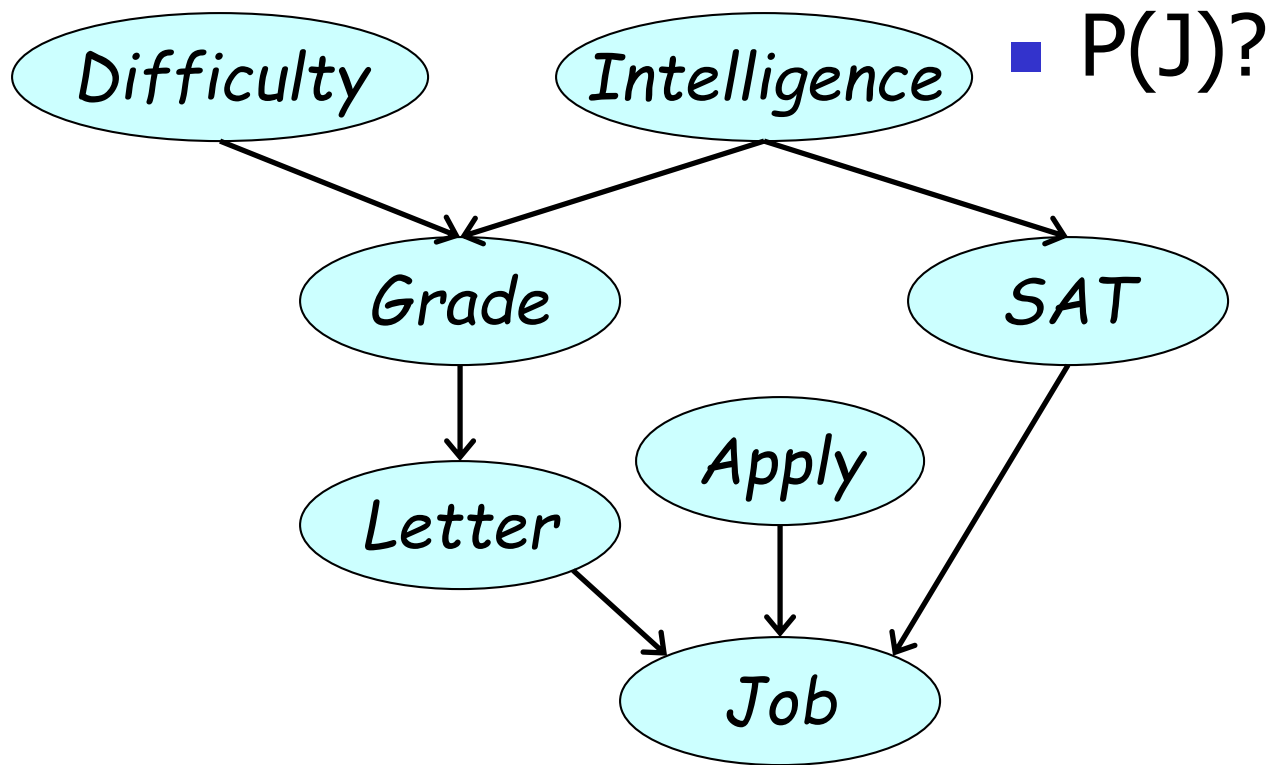
Input: A belief network $\mathcal{B} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}_G, \mathbf{I} \rangle$, an ordering $d = (X_1, \dots, X_n)$; evidence e

output: The belief $P(X_1|e)$ and probability of evidence $P(e)$

1. Partition the input functions (CPTs) into $bucket_1, \dots, bucket_n$ as follows:
 for $i \leftarrow n$ **downto** 1, put in $bucket_i$ all unplaced functions mentioning X_i .
 Put each observed variable in its bucket. Denote by ψ_i the product of input functions in $bucket_i$.
2. **backward:** **for** $p \leftarrow n$ **downto** 1 **do**
3. **for** all the functions $\psi_{S_0}, \lambda_{S_1}, \dots, \lambda_{S_j}$ in $bucket_p$ **do**
 If (observed variable) $X_p = x_p$ appears in $bucket_p$,
 assign $X_p = x_p$ to each function in $bucket_p$ and then
 put each resulting function in the bucket of the *closest* variable in its scope.
 else,
4. $\lambda_p \leftarrow \sum_{X_p} \psi_p \cdot \prod_{i=1}^j \lambda_{S_i}$
5. place λ_p in bucket of the latest variable in $scope(\lambda_p)$,
6. **return** (as a result of processing $bucket_1$):
 $P(e) = \alpha = \sum_{X_1} \psi_1 \cdot \prod_{\lambda \in bucket_1} \lambda$
 $P(X_1|e) = \frac{1}{\alpha} \psi_1 \cdot \prod_{\lambda \in bucket_1} \lambda$

Figure 4.5: BE-bel: a sum-product bucket-elimination algorithm.

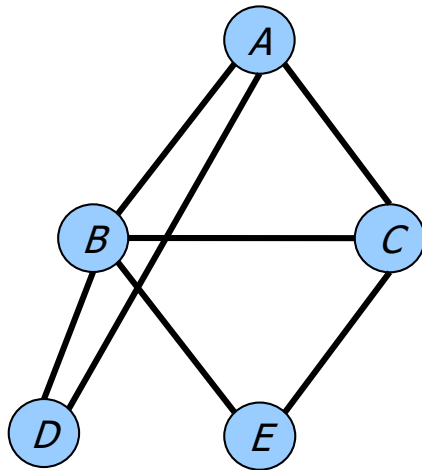
Student Network Example



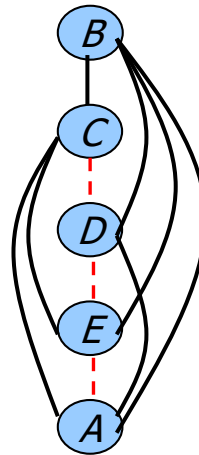
Induced Width (continued)

$w^*(d)$ – the induced width of the primal graph along ordering d

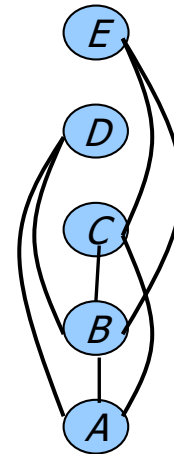
The effect of the ordering:



*Primal (moral)
graph*



$$w^*(d_1) = 4$$



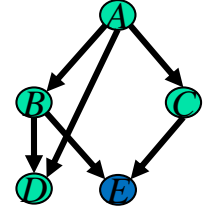
$$w^*(d_2) = 2$$



The impact of evidence

The impact of evidence?

Algorithm *BE-bel*



$$P(A | E = 0) = \alpha \sum_{E=0, D, C, B} P(A) \cdot P(B | A) \cdot P(C | A) \cdot P(D | A, B) \cdot P(E | B, C)$$

$\sum_b \prod$ ← Elimination operator

bucket B:

$$P(b/a) \quad P(d/b,a) \quad P(e/b,c)$$

$B=1$

bucket C:

$$P(c/a) \quad \lambda^B(a, d, c, e)$$

bucket D:

$$\lambda^C(a, d, e)$$

bucket E:

$$e=0 \quad \lambda^D(a, e)$$

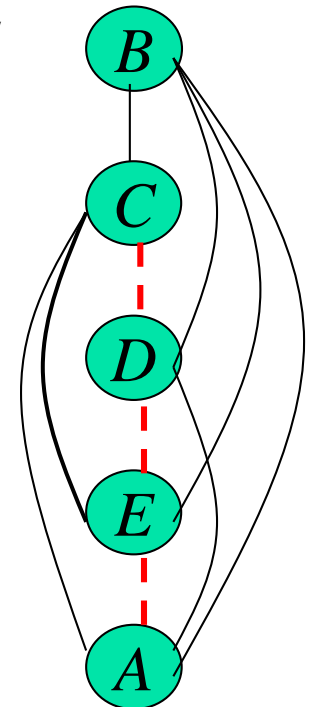
bucket A:

$$P(a) \quad \lambda^E(a)$$

"induced width"
(max clique size)

$$P(e=0)$$

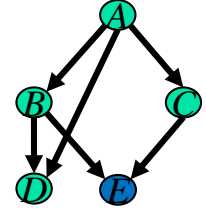
$$P(a/e=0)$$



$W^*=4$

The impact of evidence?

Algorithm *BE-bel*



$$P(A | E = 0) = \alpha \sum_{E=0, D, C, B} P(A) \cdot P(B | A) \cdot P(C | A) \cdot P(D | A, B) \cdot P(E | B, C)$$

$\sum_b \prod$ ← Elimination operator

bucket B:

$$P(b/a) \quad P(d/b, a) \quad P(e/b, c)$$

$B=1$

bucket C:

$$P(c/a)$$

$$P(e/b=1, c)$$

bucket D:

$$P(d/b=1, a)$$

bucket E:

$$e=0$$

bucket A:

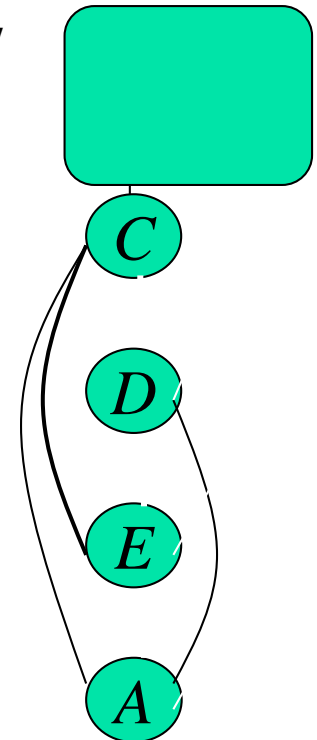
$$P(a)$$

$$P(b=1/a)$$

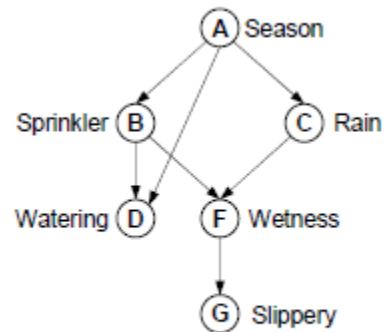
$$P(e=0)$$

$$P(a/e=0)$$

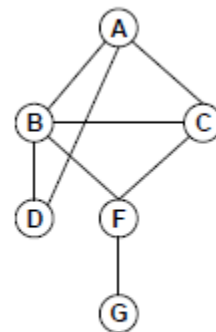
$$P(a|e=0) = \frac{P(a, e=0)}{P(e=0)}$$



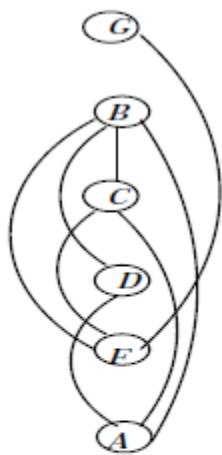
The Impact of Evidence



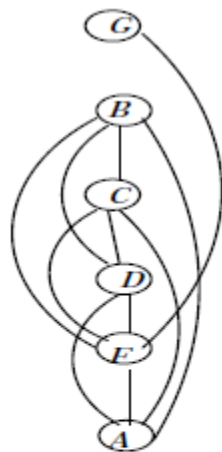
(a) Directed acyclic graph



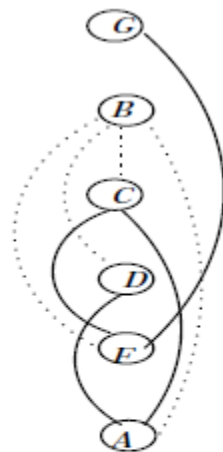
(b) Moral graph



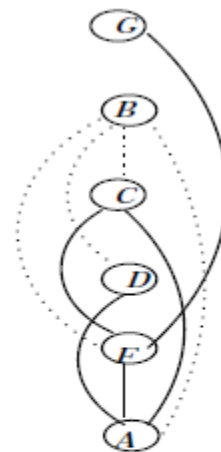
(a)



(b)

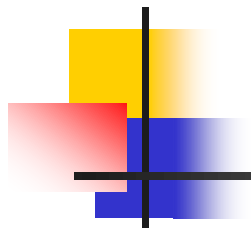


(c)

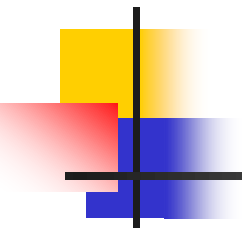


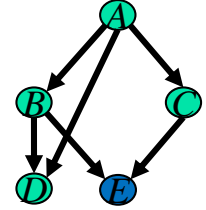
(d)

Figure 5.8: Adjusted induced graph relative to observing B



Bucket-elimination for MPE (MAP)

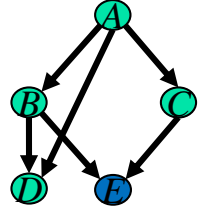

$$MPE = \max_{\bar{x}} P(\bar{x})$$



\sum is replaced by *max* :

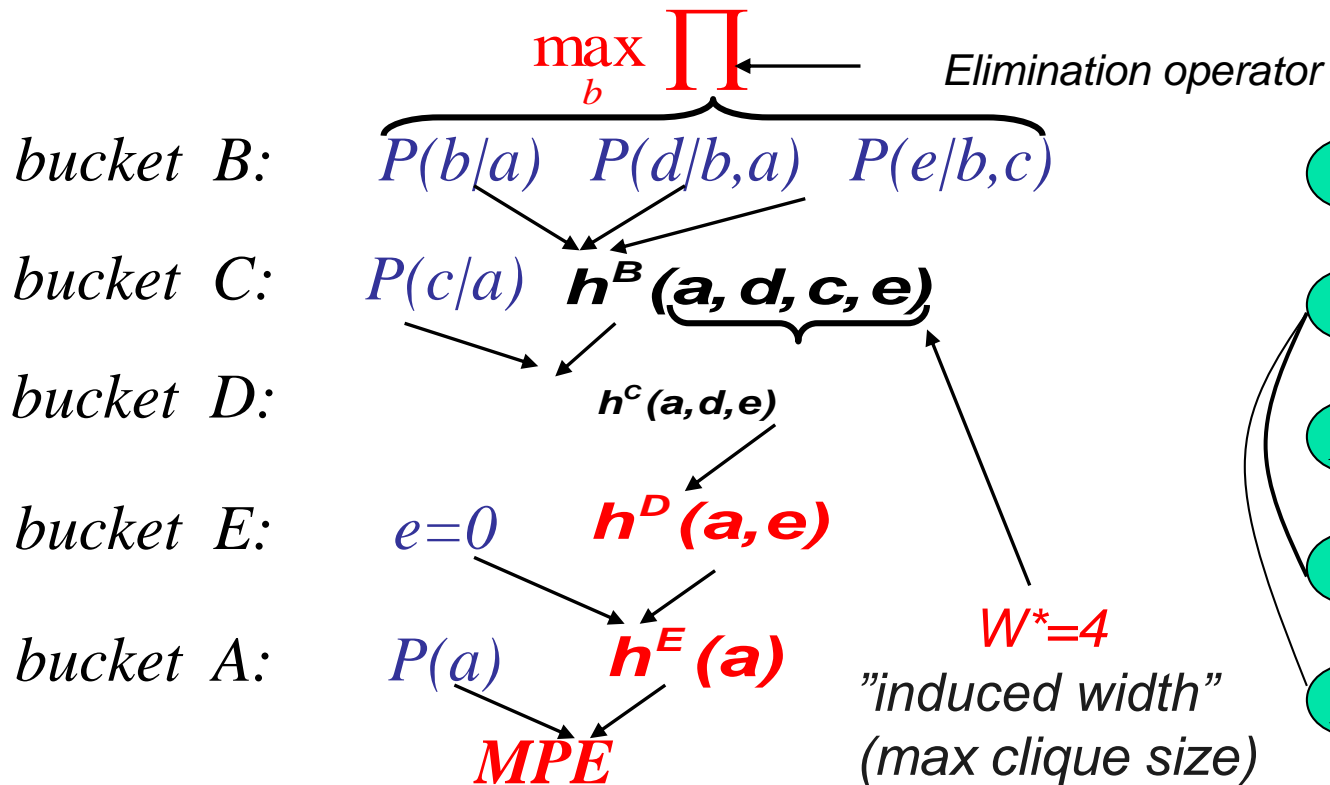
$$MPE = \max_{a,e,d,c,b} P(a)P(c | a)P(b | a)P(d | a,b)P(e | b,c)$$

$$MPE = \max_{\bar{x}} P(\bar{x})$$



\sum is replaced by *max* :

$$MPE = \max_{a,e,d,c,b} P(a)P(c|a)P(b|a)P(d|a,b)P(e|b,c)$$



Generating the MPE-tuple

5. $b' = \arg \max_b P(b | a') \times P(d' | b, a') \times P(e' | b, c')$

4. $c' = \arg \max_c P(c | a') \times h^B(a', d', c, e')$

3. $d' = \arg \max_d h^C(a', d, e')$

2. $e' = 0$

1. $a' = \arg \max_a P(a) \cdot h^E(a)$

B: $P(b/a) \quad P(d/b,a) \quad P(e/b,c)$

C: $P(c/a) \quad h^B(a,d,c,e)$

D: $h^C(a,d,e)$

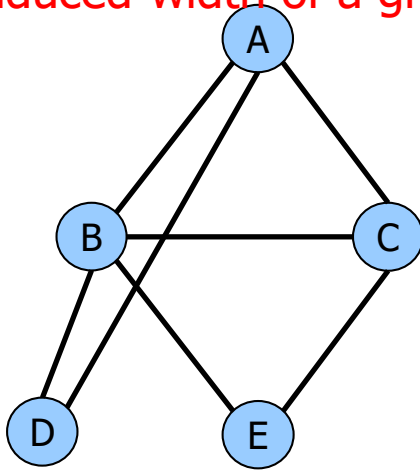
E: $e=0 \quad h^D(a,e)$

A: $P(a) \quad h^E(a)$

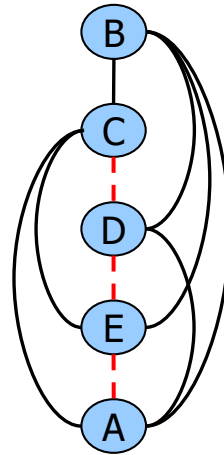
Return (a', b', c', d', e')

Induced Width

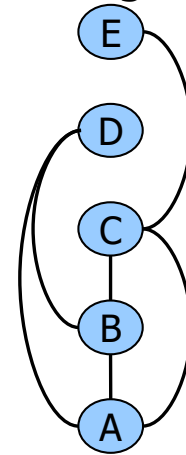
- **Width** is the max number of parents in the ordered graph
- **Induced-width** is the width of the induced ordered graph: recursively connecting parents going from last node to first.
- **Induced-width $w^*(d)$** is the max induced-width over all nodes in ordering d
- **Induced-width of a graph, w^*** is the min $w^*(d)$ over all orderings d



primal
graph



$$w^*(d_1) = 4$$



$$w^*(d_2) = 2$$

Complexity of Bucket Elimination

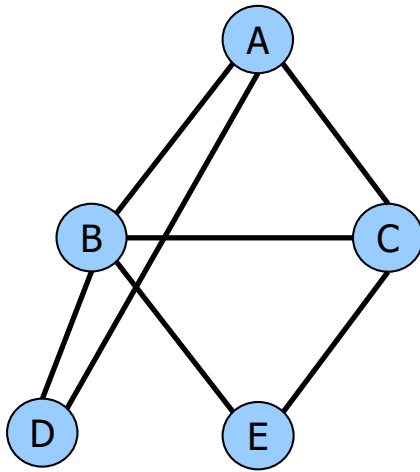
*Bucket-Elimination is **time** and **space***

$$O(r \exp(w_d^*))$$

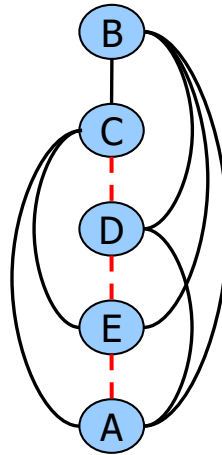
w_d^* : the induced width of the primal graph along ordering d

r = number of functions

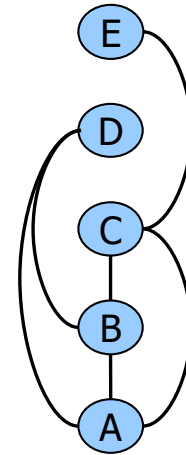
The effect of the ordering:



primal
graph



$$w^*(d_1) = 4$$

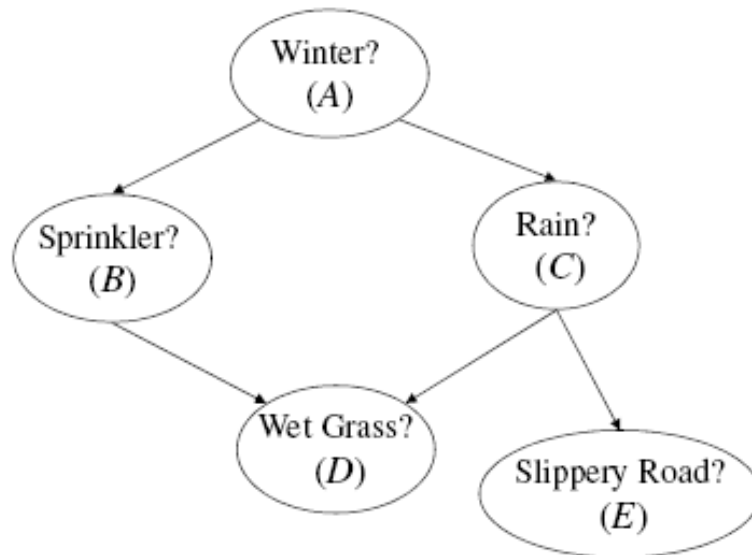


$$w^*(d_2) = 2$$

Finding smallest induced-width is hard!

A Bayesian Network

Example with mpe?



A	Θ_A
true	.6
false	.4

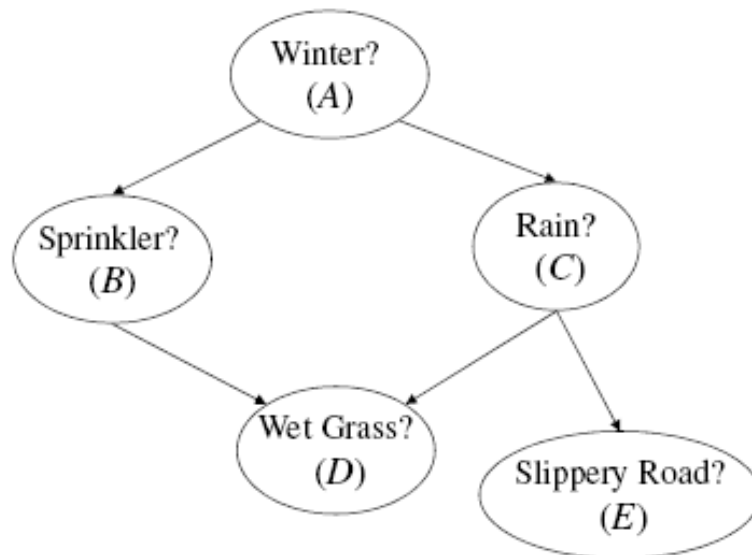
A	B	$\Theta_{B A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	C	$\Theta_{C A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	C	D	$\Theta_{D BC}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Try to compute MPE when $E=0$



A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	C	$\Theta_{C A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	C	D	$\Theta_{D BC}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Cost networks

$$P(a, b, c, d, f, g) = P(a)P(b|a)P(c|a)P(f|b, c)P(d|a, b)P(g|f)$$

becomes

$$C(a, b, c, d, e) = -\log P = C(a) + C(b, a) + C(c, a) + C(f, b, c) + C(d, a, b) + C(g, f)$$

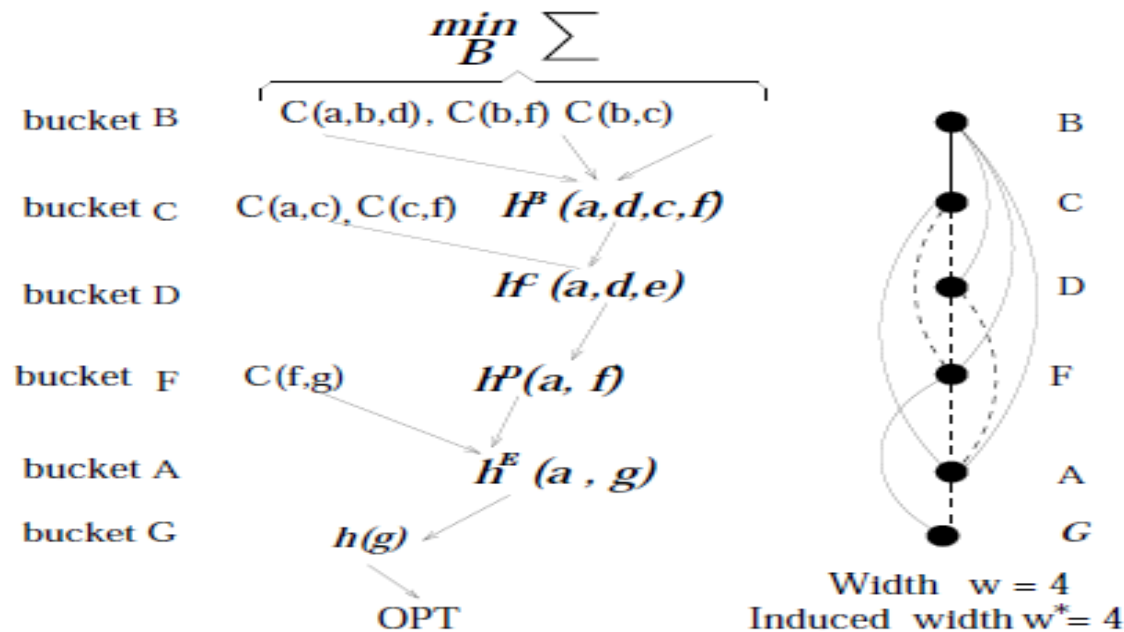
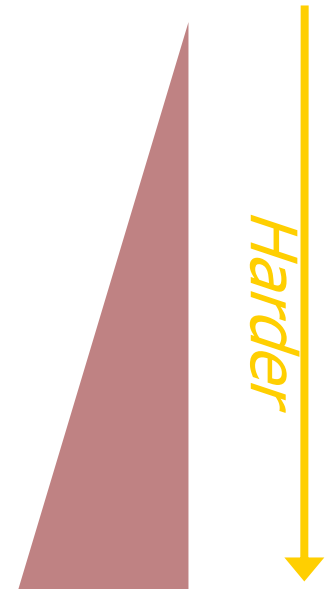


Figure 5.12: Schematic execution of BE-Opt

Marginal Map

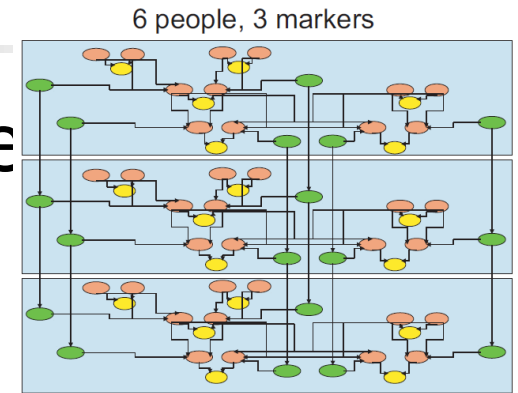
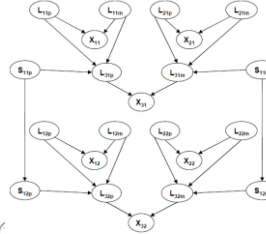
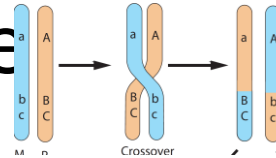
▶ Max-Inference	$f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$
▶ Sum-Inference	$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$
▶ Mixed-Inference	$f(\mathbf{x}_M^*) = \max_{\mathbf{x}_M} \sum_{\mathbf{x}_S} \prod_{\alpha} f_{\alpha}(\mathbf{x}_{\alpha})$



- **NP-hard**: exponentially many terms

Example for MMAP Applications

- Haplotype family



- Coding net

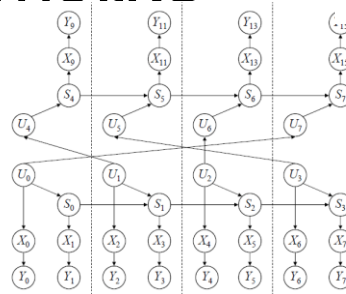
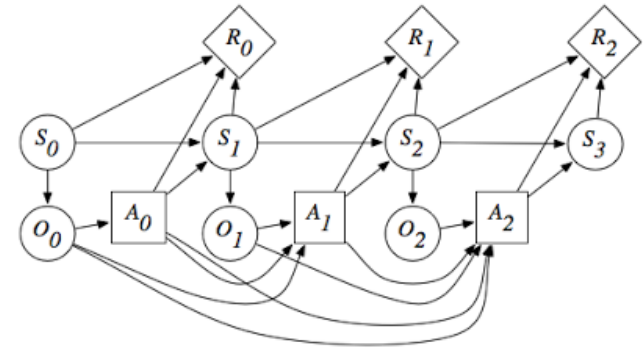
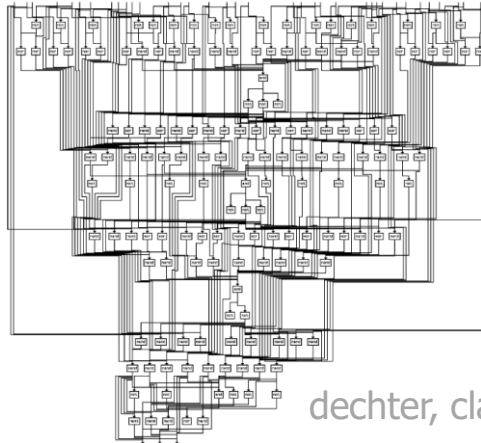


Figure 5.24: A Bayesian network for a turbo code.

- Probabilistic planning



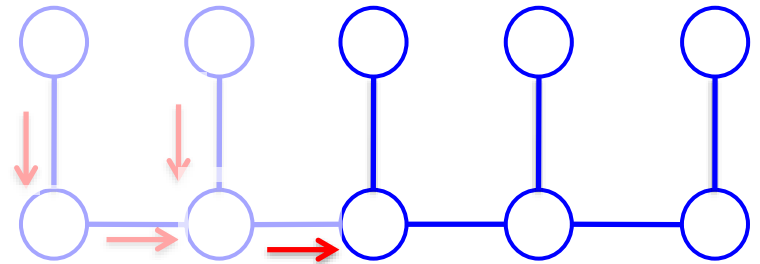
- Diagnostic



Marginal MAP is not Easy on Trees

- Pure MAP or summation tasks

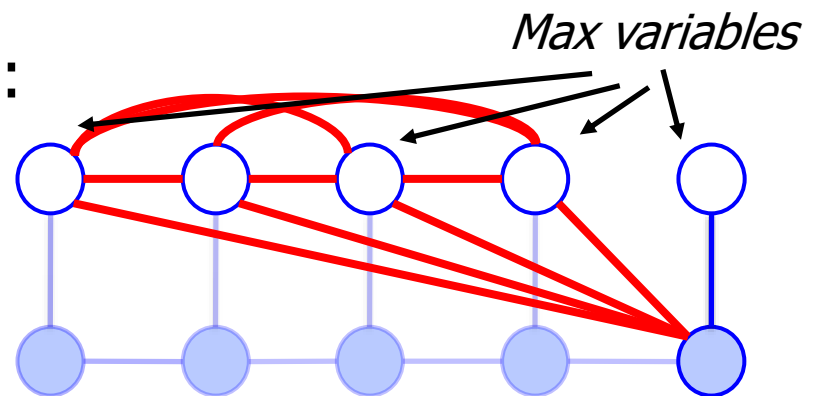
- Dynamic programming
- Ex: efficient on trees



- Marginal MAP

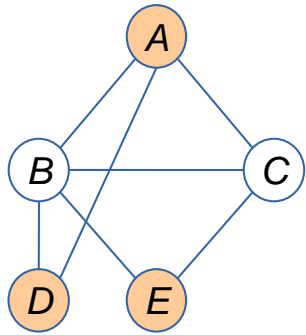
- Operations do not commute:
- Sum must be done first!

$$\sum \max \neq \max \sum$$



Bucket Elimination for MMAP

Bucket Elimination



$$\mathbf{X}_M = \{A, D, E\}$$

$$\mathbf{X}_S = \{B, C\}$$

$$\max_{\mathbf{X}_M} \sum_{\mathbf{X}_S} P(\mathbf{X})$$

constrained elimination order

SUM

MAX

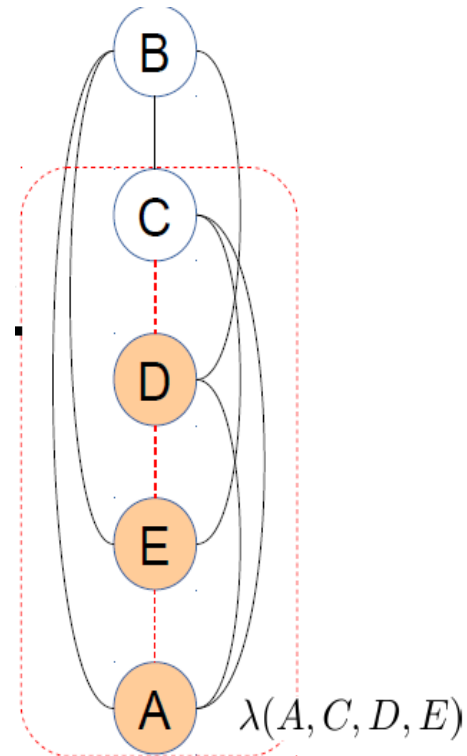
$$B: \underbrace{f(A, B) f(B, C) f(B, D) f(B, E)}_{\Sigma_B}$$

$$C: \underbrace{\lambda^B(A, C, D, E) f(A, C) f(C, E)}_{\Sigma_C}$$

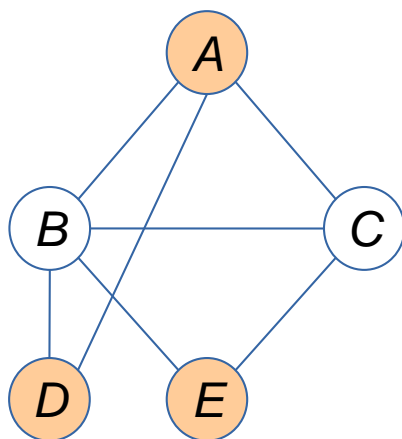
$$D: \underbrace{\lambda^C(A, D, E) f(A, D)}_{\max_D}$$

$$E: \underbrace{\lambda^D(A, E)}_{\max_E}$$

$$A: \underbrace{\lambda^E(A)}_{\text{MAP}^* \text{ is the marginal MAP value}}$$



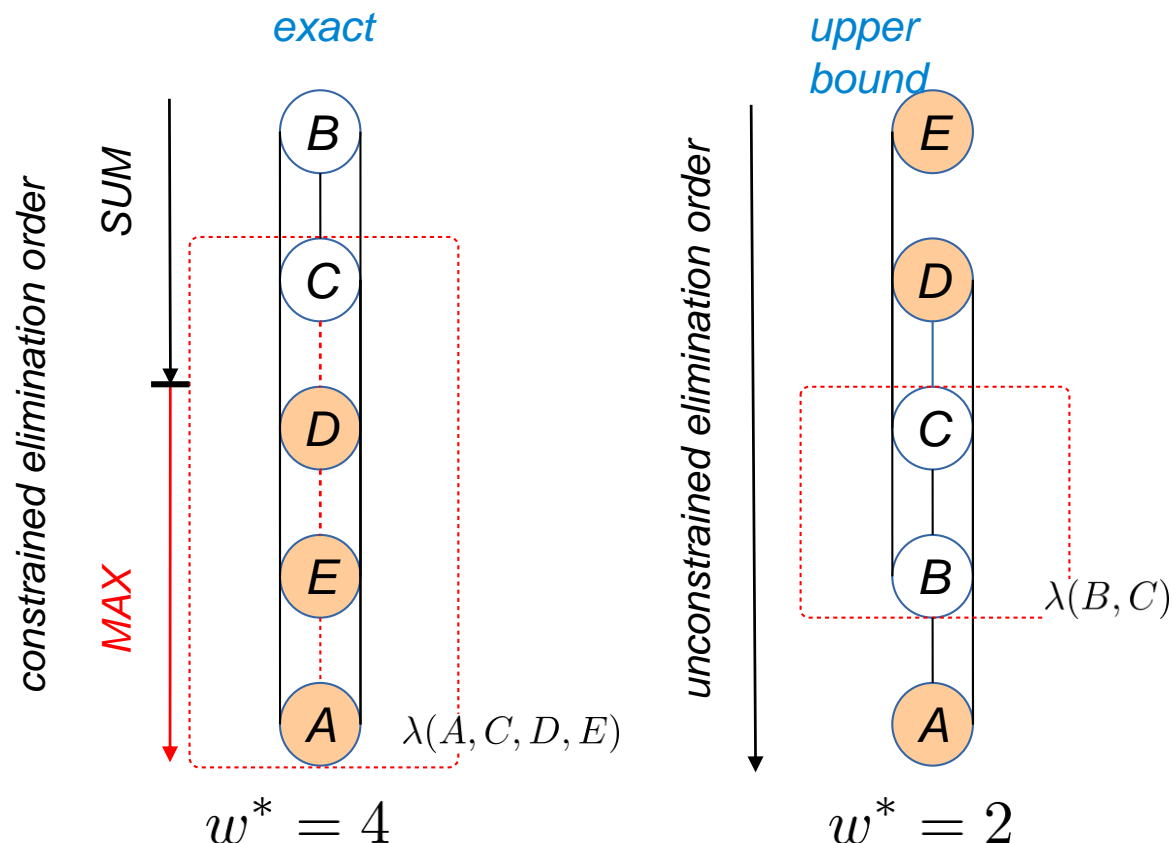
Why is MMAP harder?



$$\mathbf{X}_M = \{A, D, E\}$$

$$\mathbf{X}_S = \{B, C\}$$

(Park & Darwiche, 2003)
(Yuan & Hansen, 2009)



In practice, constrained induced is much larger!

$$\max_{\mathbf{X}} \sum \phi \leq \sum_Y \max_{\mathbf{X}} \phi$$

dechter, class4 236-18



Complexity of Bucket-Elimination

■ **Theorem:**

BE is $O(n \exp(w^*+1))$ time and $O(n \exp(w^*))$ space, when w^* is the induced-width of the moral graph along d when evidence nodes are processed (edges from evidence nodes to earlier variables are removed.)

More accurately: $O(r \exp(w^(d)))$ where r is the number of CPTs.
For Bayesian networks $r=n$. For Markov networks?*



Inference with Markov Networks

- Undirected graphs with potentials on cliques
- Query: find *partition function*. Same as probability of the evidence in a Bayesian network.
- The joint probability distribution of a Markov network is defined by:

$$P(x) = \frac{1}{Z} \sum_{x \in \mathcal{D}} \prod_{C \in \mathcal{C}} \Psi_C(x_C) \quad \text{BE is equally applicable}$$

$$Z = \sum_x \prod_{C \in \mathcal{C}} \Psi_C(x_C) \quad (2.2)$$

For example. A markov network over the moral graph in Figure 2.4(b) is defined by:

$$P(a, b, c, d, f, g) = \frac{\Psi(a, b, c) \cdot \Psi(b, c, f) \cdot \Psi(a, b, d) \cdot \Psi(f, g)}{Z} \quad (2.3)$$

where,

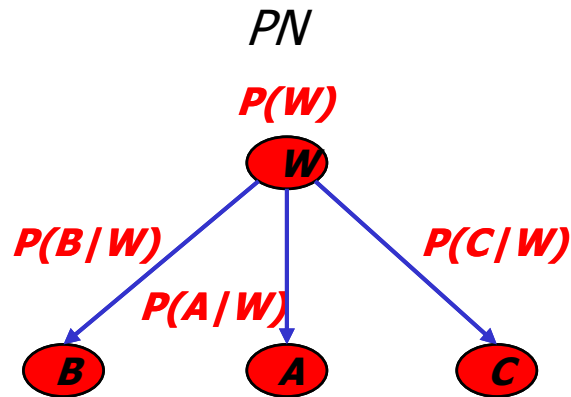
$$Z = \sum_{a, b, c, d, e, f, g} \Psi(a, b, c) \cdot \Psi(b, c, f) \cdot \Psi(a, b, d) \cdot \Psi(f, g) \quad (2.4)$$



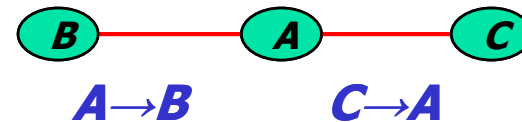
Inference for probabilistic networks

- **Bucket elimination**
 - Belief-updating, $P(e)$, partition function
 - Marginals, probability of evidence
 - The impact of evidence
 - for MPE (\rightarrow MAP)
 - for MAP (\rightarrow Marginal Map)
 - Mixed networks
- **Tree-decomposition schemes**
 - Bucket tree elimination
 - Cluster tree elimination

Party Example



CN



Semantics?

Algorithms?

Query:

Is it likely that Chris goes to the party if Becky does not but the weather is bad?

$$P(C, \neg B \mid w = \text{bad}, A \rightarrow B, C \rightarrow A)$$



Bucket Elimination for Mixed Networks

The CPE query

$$P_{\mathcal{B}}(\varphi) = \sum_{\mathbf{x}_{\varphi} \in \text{Mod}(\varphi)} P(\mathbf{x}_{\varphi})$$

Using the belief network product form we get:

$$P_{\mathcal{B}}(\varphi) = \sum_{\{\mathbf{x} | \mathbf{x}_{\varphi} \in \text{Mod}(\varphi)\}} \prod_{i=1}^n P(x_i | \mathbf{x}_{pa_i}).$$

$P((C \rightarrow B) \text{ and } P(A \rightarrow C))$

Processing Mixed Buckets

Example 4.18 Consider the belief network in Figure 4.14, which is similar to the one in Figure 2.5, and the query $\varphi = (B \vee C) \wedge (G \vee D) \wedge (\neg D \vee \neg B)$. The initial partitioning into buckets along the ordering $d = A, C, B, D, F, G$, as well as the output buckets are given in Figure 4.15. We compute:

In Bucket G : $\lambda_G(f, d) = \sum_{\{g|g \vee d = \text{true}\}} P(g|f)$

In Bucket_F : $\lambda_F(b, c, d) = \sum_f P(f|b, c) \lambda_G(f, d)$

In Bucket_D : $\lambda_D(a, b, c) = \sum_{\{d|\neg d \vee \neg b = \text{true}\}} P(d|a, b) \lambda_F(b, c, d)$

In Bucket_B : $\lambda_B(a, c) = \sum_{\{b|b \vee c = \text{true}\}} P(b|a) \lambda_D(a, b, c) \lambda_F(b, c)$

In Bucket_C : $\lambda_C(a) = \sum_c P(c|a) \lambda_B(a, c)$

In Bucket_A : $\lambda_A = \sum_a P(a) \lambda_C(a)$

$P(\varphi) = \lambda_A.$

Bucket-Elimination example for a Mixed Network

$$\varphi = (B \vee C), (G \vee D), (\sim D \vee \sim B)$$

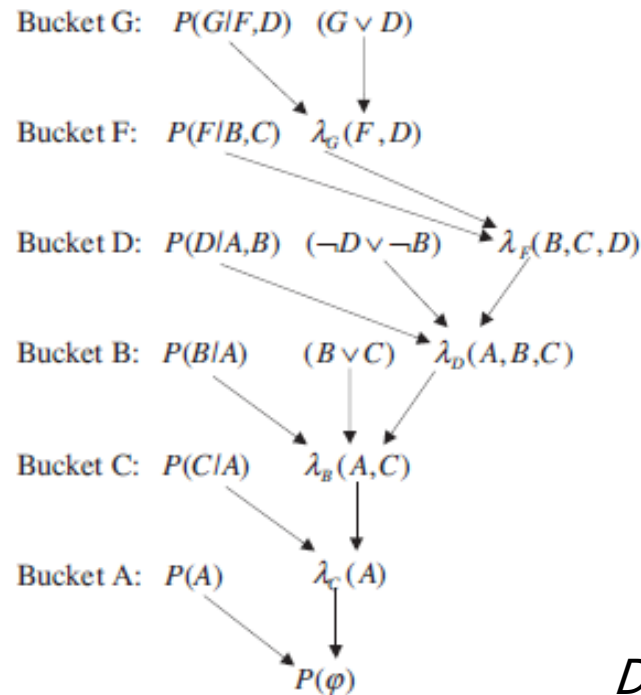
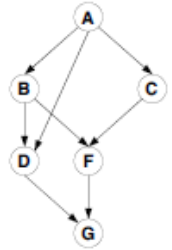
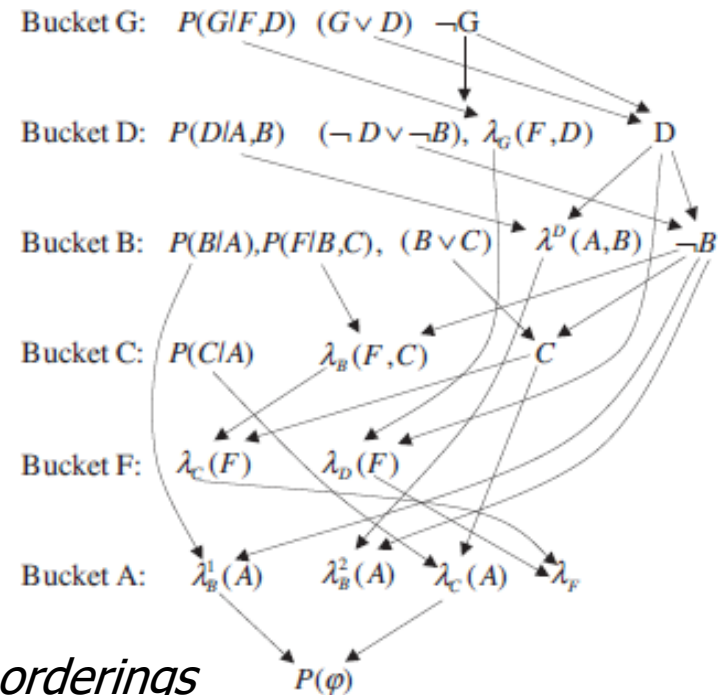


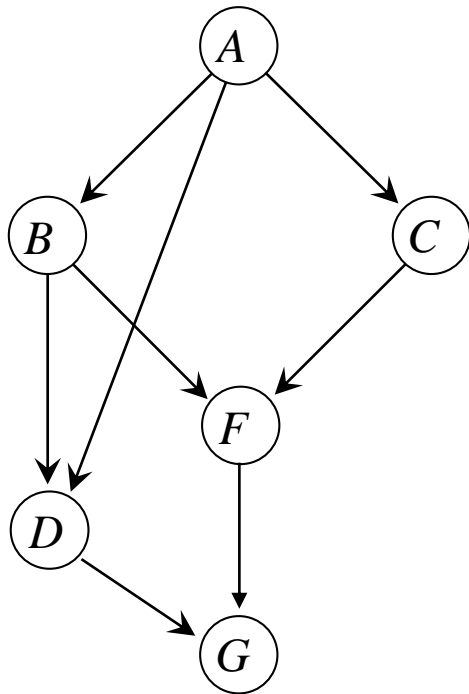
Figure 4.15: Execution of BE-CPE.



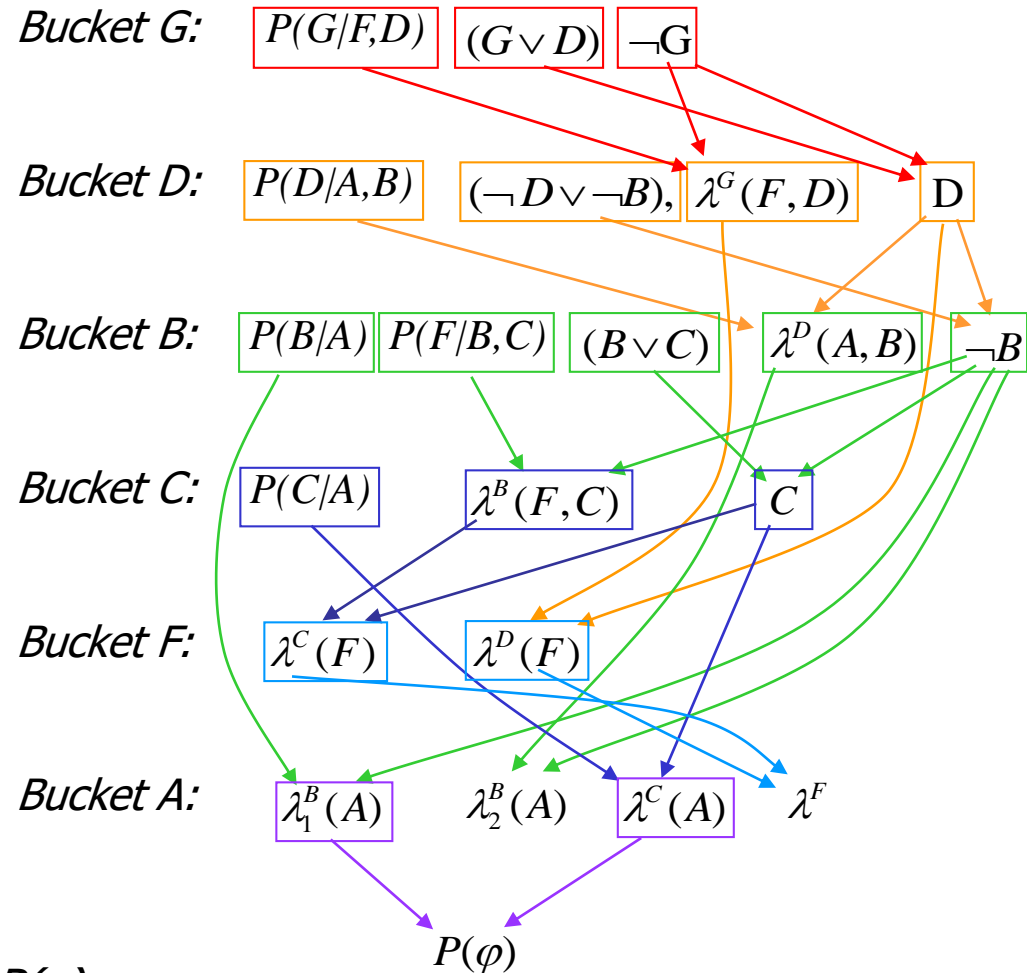
Different orderings

Figure 4.16: Execution of BE-CPE (evidence $\neg G$).

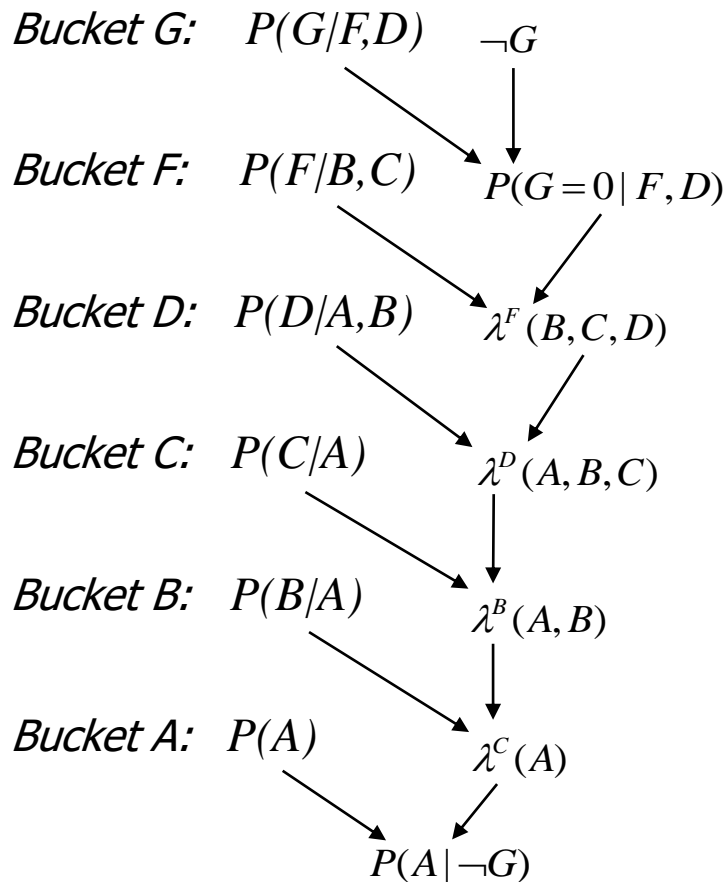
Trace of Elim-CPE



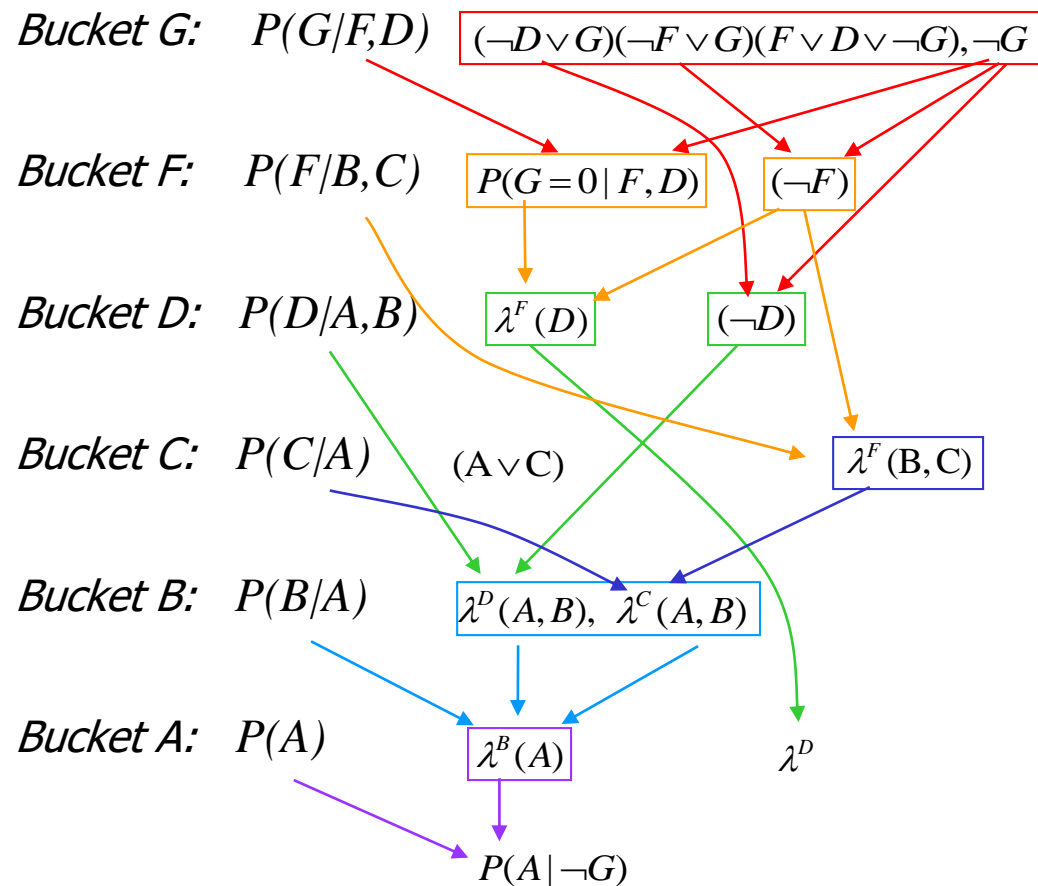
Belief network $P(g,f,d,c,b,a)$
 $=P(g/f,d)P(f/c,b)P(d/b,a)P(b/a)P(c/a)P(a)$



Bucket-Elimination for CPE



(a) regular Elim-CPE



(b) Elim-CPE-D with clause extraction

Belief Updating Example

SUM-PROD operators
POLY-TREE structure

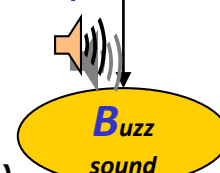
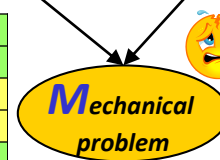
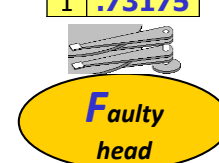
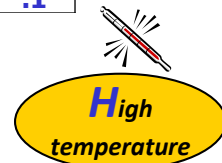
H	P(H)
0	.9
1	.1

F	P(F)
0	.99
1	.01

F	$h_3(F)$
0	.1245
1	.73175

F	$h_4(F)$
0	1
1	1

F	P(F,B=1)
0	.123255
1	.073175



H	F	M	$B(M H,F)$
0	0	0	.0405
0	0	1	.072
0	1	0	.0045
0	1	1	.649
1	0	0	.008
1	0	1	.008
1	1	0	.00005
1	1	1	.0792

F	R	P(R F)
0	0	.8
0	1	.2
1	0	.3
0	1	.7

M	B	P(B M)
0	0	.95
0	1	.05
1	0	.2
1	1	.8

$$P(h,f,r,m,b) = P(h) P(f) P(m|h,f) P(r|f) P(b|m)$$

$$P(F | B=1) = ?$$

$$P(B=1) = .19643$$

$$P(F=1|B=1) = .3725$$

Probability of evidence

Updated belief

B: $P(B|M)$
H: $P(M|H,F), P(H)$
M:
R: $P(R|F)$
F: $P(F)$



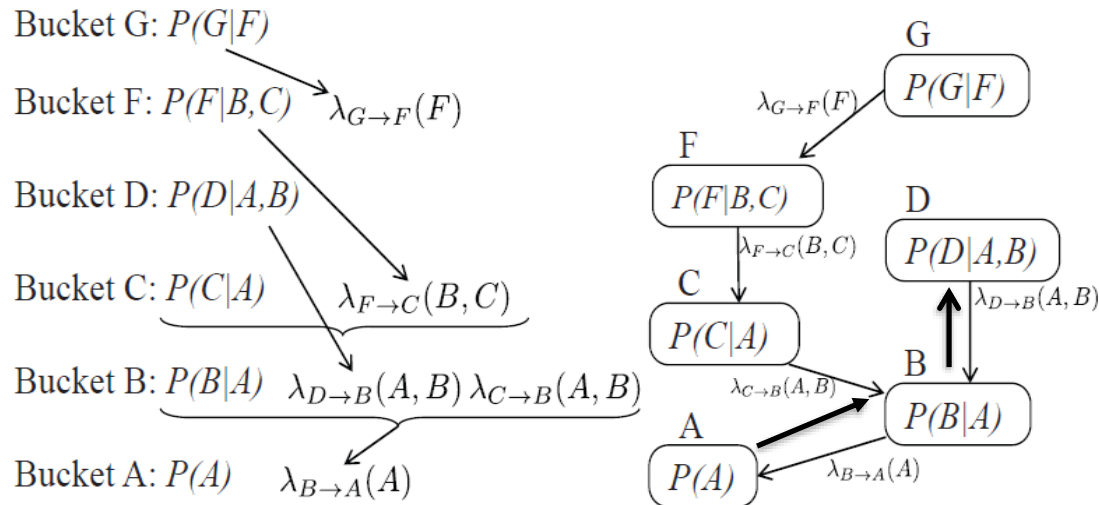
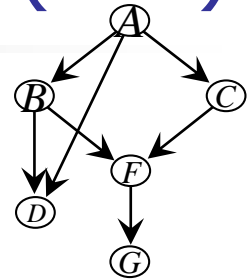
Outline

- From bucket-elimination (BE) to bucket-tree elimination (BTE)
- From BTE to CTE, Acyclic networks, the join-tree algorithm
- Generating join-trees, the treewidth
- Examples of CTE for Bayesian network
- Belief-propagation on acyclic probabilistic networks (poly-trees) and Loopy networks

From BE to Bucket-Tree Elimination(BTE)

First, observe the BE operates on a tree.

Second, What if we want the marginal on D?



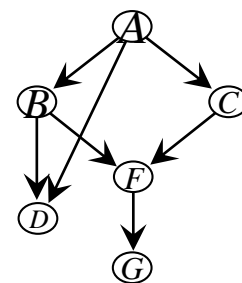
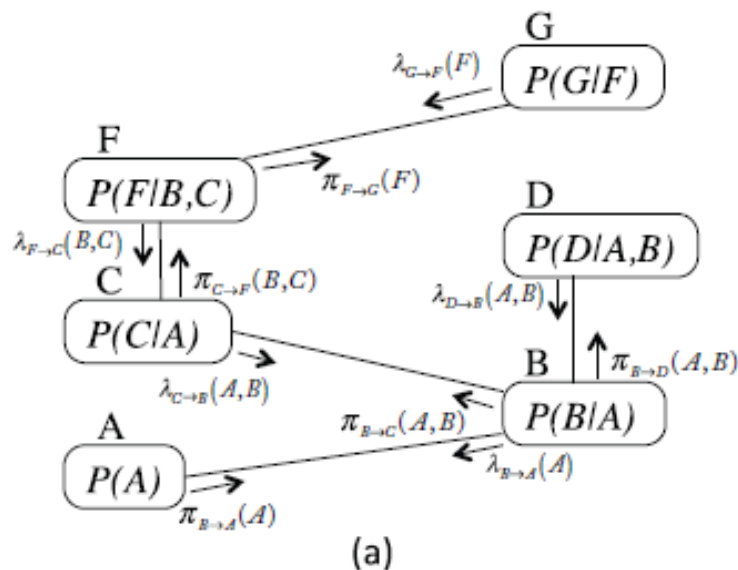
$$\pi_{A \rightarrow B}(a) = P(A),$$

$$\pi_{B \rightarrow D}(a, b) = p(b|a) \cdot \pi_{A \rightarrow B}(a) \cdot \lambda_{C \rightarrow B}(b)$$

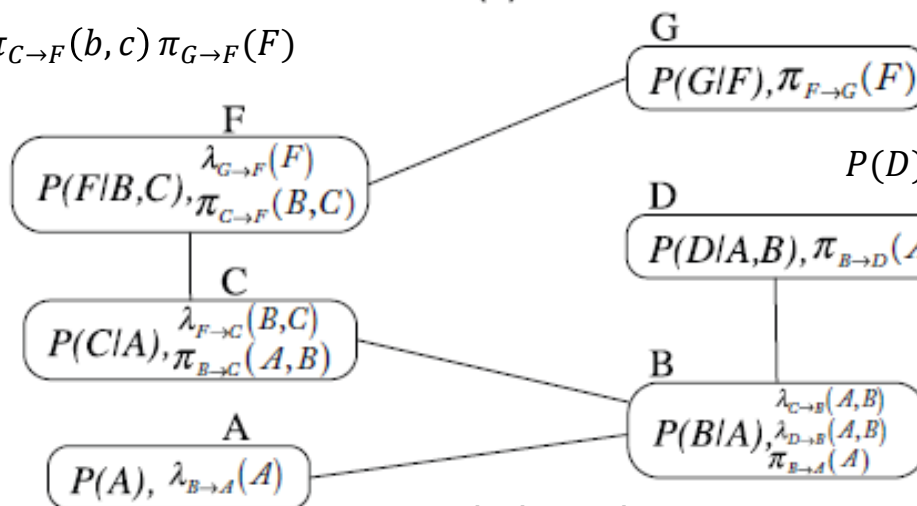
$$bel(d) = \alpha \sum_{a,b} P(d|a, b) \cdot \pi_{B \rightarrow D}(a, b).$$

BTE: Allows Messages Both Ways

Initial bucket
+ messages

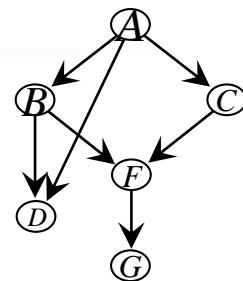


$$P(F) = \sum_{b,c} P(F|b,c) \pi_{C \rightarrow F}(b,c) \pi_{G \rightarrow F}(F)$$



$$P(D) = \sum_{a,b} P(D|a,b) \pi_{B \rightarrow D}(a,b)$$

BTE: Allows Messages Both Ways



Bucket G: $P(G/F)$

Bucket F: $P(F/B, C) \rightarrow \lambda_G^F(F)$

Bucket D: $P(D/A, B)$

Bucket C: $P(C/A)$

Bucket B: $P(B/A)$

Bucket A: $P(A)$

$\pi_F^G(F)$

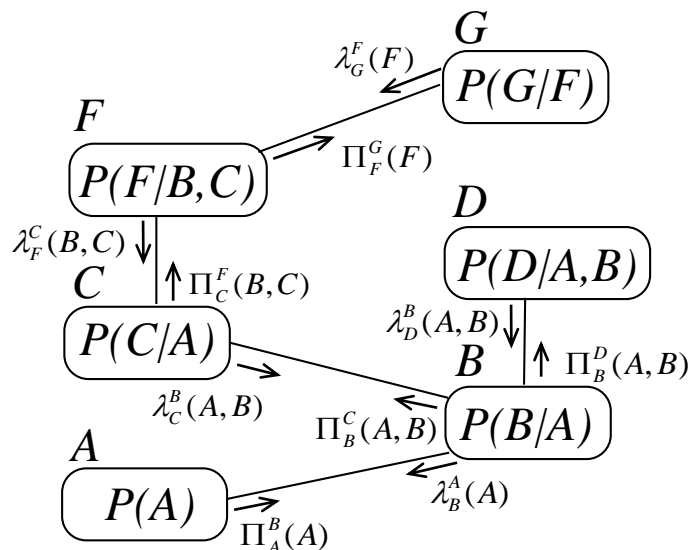
$\pi_C^F(B, C)$

$\pi_B^D(A, B)$

$\pi_B^C(A, B)$

$\pi_A^B(A)$

Each bucket can
Compute its
marginal probability



$$\pi_A^B(a) = P(a)$$

$$\pi_B^C(c, a) = P(b|a) \lambda_D^B(a, b) \pi_A^B(a)$$

$$\pi_B^D(a, b) = P(b|a) \lambda_C^B(a, b) \pi_A^B(a, b)$$

$$\pi_C^F(c, b) = \sum_a P(c|a) \pi_B^C(a, b)$$

$$\pi_F^G(f) = \sum_{b,c} P(f|b, c) \pi_C^F(c, b)$$



Idea of BTE

This example can be generalized. We can compute the belief for every variable by a second message passing from the root to the leaves along the original bucket-tree, such that at termination the belief for each variable can be computed locally consulting only the functions in its own bucket. In the following we will describe the idea of message

passing in Bayesian networks. Given an ordering of the variables d the first step generates the bucket-tree by partitioning the functions into buckets and connecting the buckets into a tree. The subsequent *top-down* phase is identical to general bucket-elimination. The *bottom-up* messages are defined as follows. The messages sent from the root up to the leaves will be denoted by π . The message from B_j to a child B_i is generated by combining (e.g., multiplying) all the functions currently in B_j including the π messages from its parent bucket and all the λ messages from its *other* child buckets and marginalizing (e.g., summing) over the eliminator from B_j to B_i . By construction, downward messages are generated by eliminating a single variable. Upward messages, on the other hand, may be generated by eliminating zero, one or more variables.

BTE

Theorem: When BTE terminates The product of functions in each bucket is the beliefs of the variables joint with the evidence.

$$Elim(i,j) = B_i - B_j$$

ALGORITHM BUCKET-TREE ELIMINATION (BTE)

Input: A problem $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \Pi, \Sigma \rangle$, ordering d .

$X = \{X_1, \dots, X_n\}$ and $F = \{f_1, \dots, f_r\}$

Evidence $E = e$.

Output: Augmented buckets $\{B'_i\}$, containing the original functions and all the π and λ functions received from neighbors in the bucket tree.

1. **Pre-processing:** Partition functions to the ordered buckets as usual and generate the bucket tree.
2. **Top-down phase:** λ messages (BE) **do**
 - for** $i = n$ to 1, in reverse order of d process bucket B_i :
 - The message $\lambda_{i \rightarrow j}$ from B_i to its parent B_j , is:

$$\lambda_{i \rightarrow j} \Leftarrow \sum_{elim(i,j)} \psi_i \cdot \prod_{k \in child(i)} \lambda_{k \rightarrow i}$$
 - endfor**
3. **bottom-up phase:** π messages
 - for** $j = 1$ to n , process bucket B_j **do**:
 - B_j takes $\pi_{k \rightarrow j}$ received from its parent B_k , and computes a message $\pi_{j \rightarrow i}$ for each child bucket B_i by

$$\pi_{j \rightarrow i} \Leftarrow \sum_{elim(j,i)} \pi_{k \rightarrow j} \cdot \psi_j \cdot \prod_{r \neq i} \lambda_{r \rightarrow j}$$
 - endfor**
4. **Output:** augmented buckets B'_1, \dots, B'_n , where each B'_i contains the original bucket functions and the λ and π messages it received.

Figure 5.3: Algorithm bucket-tree elimination.



Bucket-Tree Construction From the Graph

1. Pick a (good) variable ordering, d .
2. Generate the induced ordered graph
3. From top to bottom, each bucket of X is mapped to (variables, functions) pairs
4. The variables are the clique of X , the functions are those placed in the bucket
5. Connect the bucket of X to earlier bucket of Y if Y is the closest node connected to X



Asynchronous BTE: Bucket-tree Propagation (BTP)

BUCKET-TREE PROPAGATION (BTP)

Input: A problem $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \prod, \sum \rangle$, ordering d . $X = \{X_1, \dots, X_n\}$ and $F = \{f_1, \dots, f_r\}$, $\mathbf{E} = \mathbf{e}$. An ordering d and a corresponding bucket-tree structure, in which for each node X_i , its bucket B_i and its neighboring buckets are well defined.

Output: Explicit buckets. Assume functions assigned with the evidence.

1. **for** bucket B_i **do**:
2. **for** each neighbor bucket B_j **do**,
 once all messages from all other neighbors were received, **do**
 compute and send to B_j the message

$$\lambda_{i \rightarrow j} \Leftarrow \sum_{elim(i,j)} \psi_i \cdot (\prod_{k \neq j} \lambda_{k \rightarrow i})$$

3. **Output:** augmented buckets B'_1, \dots, B'_n , where each B'_i contains the original bucket functions and the λ messages it received.



Query Answering

COMPUTING MARGINAL BELIEFS

Input: a bucket tree processed by BTE with augmented buckets: Bt_1, \dots, Bt_n

output: beliefs of each variable, bucket, and probability of evidence.

$$bel(B_i) \Leftarrow \alpha \cdot \prod_{f \in Bt_i} f$$

$$bel(X_i) \Leftarrow \alpha \cdot \sum_{B_i - \{X_i\}} \prod_{f \in Bt_i} f$$

$$P(evidence) \Leftarrow \sum_{B_i} \prod_{f \in Bt_i} f$$

Figure 5.4: Query answering.



Explicit functions

Definition 5.4 Explicit function and explicit sub-model. Given a graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \Pi \rangle$, and reasoning tasks defined by marginalization \sum and given a subset of variables $Y, Y \subseteq \mathbf{X}$, we define \mathcal{M}_Y , the explicit function of \mathcal{M} over Y :

$$\mathcal{M}_Y = \sum_{\mathbf{X}-Y} \prod_{f \in F} f, \quad (5.4)$$

We denote by F_Y any set of functions whose scopes are subsumed in Y over the same domains and ranges as the functions in \mathbf{F} . We say that (Y, F_Y) is an explicit submodel of \mathcal{M} iff

$$\prod_{f \in F_Y} f = \mathcal{M}_Y \quad (5.5)$$



Complexity of BTE/BTP on Trees

Theorem 5.6 Complexity of BTE. *Let $w^*(d)$ be the induced width of (G^*, d) where G is the primal graph of $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \prod, \sum \rangle$, r be the number of functions in \mathbf{F} and k be the maximum domain size. The time complexity of BTE is $O(r \cdot \deg \cdot k^{w^*(d)+1})$, where \deg is the maximum degree of a node in the bucket tree. The space complexity of BTE is $O(n \cdot k^{w^*(d)})$.*

Proposition 5.8 BTE on trees *For tree graphical models, algorithms BTE and BTP are time and space $O(nk^2)$ and $O(nk)$, respectively, when k bound the domain size and n bounds the number of variables.*

This will be extended to acyclic graphical models shortly

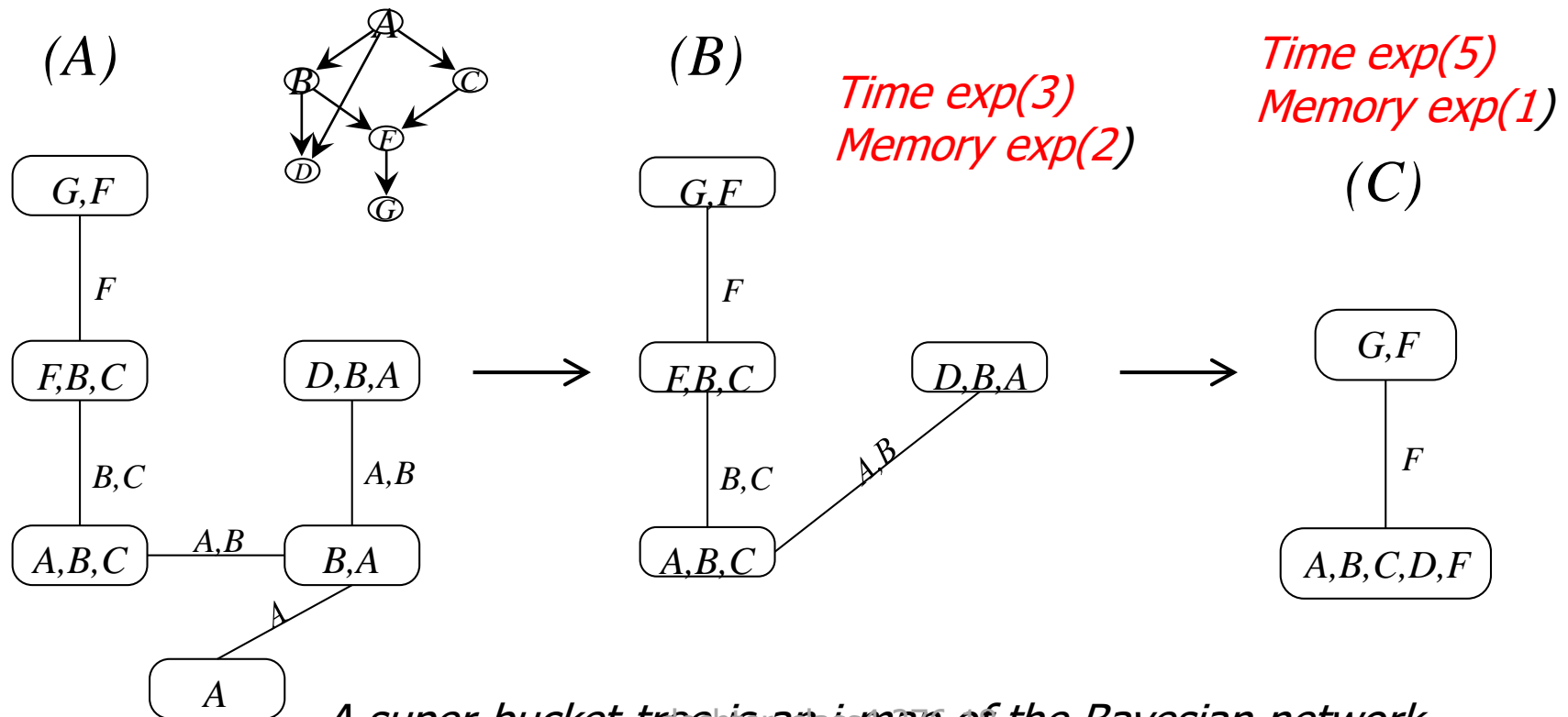


Outline

- From bucket-elimination (BE) to bucket-tree elimination (BTE)
- From BTE to CTE, Acyclic networks, the join-tree algorithm
- Examples of CTE for Bayesian network
- Belief-propagation on acyclic probabilistic networks (poly-trees) and Loopy networks

From Buckets to Clusters

- Merge non-maximal buckets into maximal clusters.
- Connect clusters into a tree: each cluster to one with which it shares a largest subset of variables.
- Separators are variable- intersection on adjacent clusters.



A super-bucket-tree is an i-map of the Bayesian network

Sometime the dual graph seems to not be a tree, but it is in fact, a tree. This is because some of its arcs are redundant and can be removed while not violating the original independency relationships that is captured by the graph.

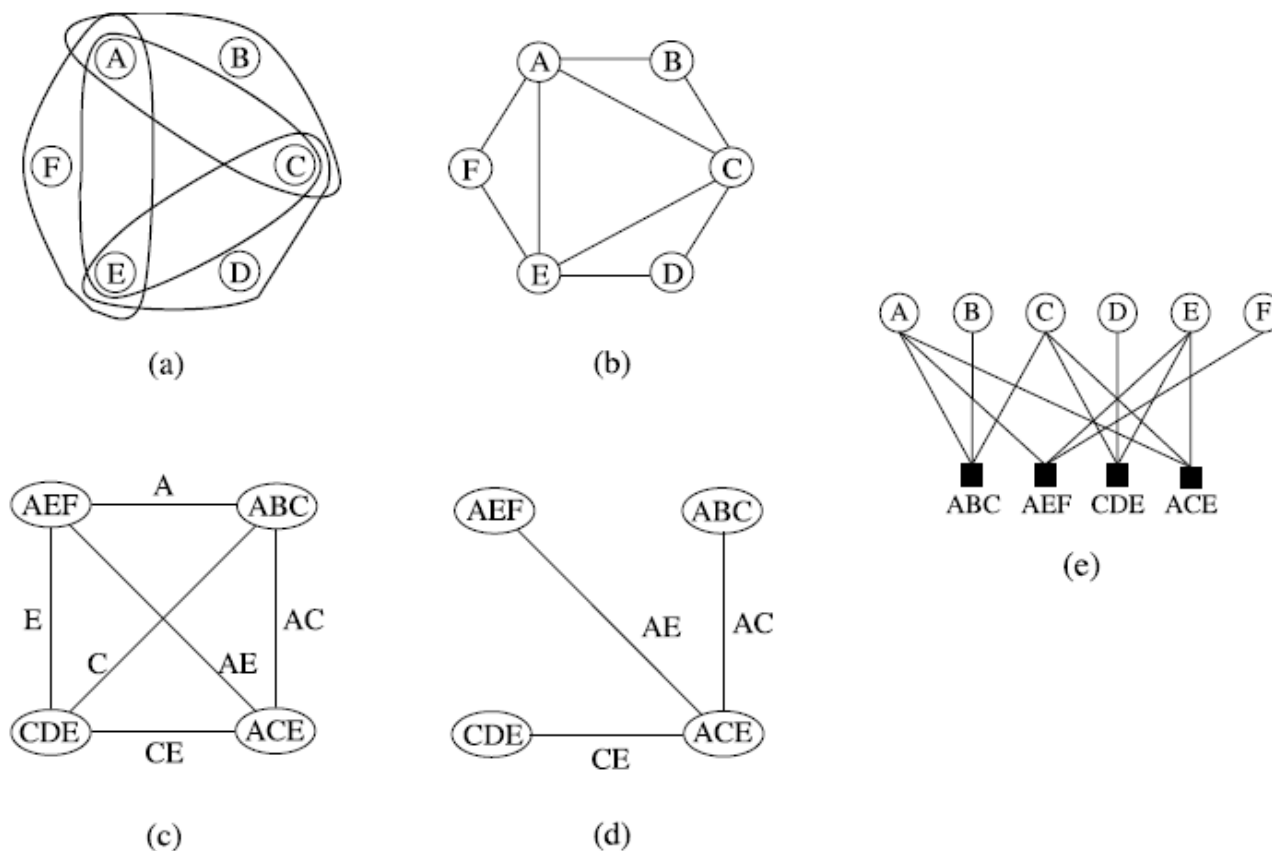


Figure 5.1: (a)Hyper, (b)Primal, (c)Dual and (d)Join-tree of a graphical model having scopes ABC , AEF , CDE and ACE . (e) the factor graph



Connectedness and Acyclic dual Graphs

(The Running Intersection Property)

Definition 5.11 Connectedness, join-trees. Given a dual graph of a graphical model \mathcal{M} , an arc subgraph of the dual graph satisfies the *connectedness* property iff for each two nodes that share a variable, there is at least one path of labeled arcs of the dual graph such that each contains the shared variables. An arc subgraph of the dual graph that satisfies the connectedness property is called a *join-graph* and if it is a tree, it is called a *join-tree*.

Definition: A graphical model whose dual graph has a join-tree is *acyclic*

Theorem: BTE is time and space linear on acyclic graphical models

Tree-decomposition: If we transform a general model into an acyclic one it can then be solved by a BTE/BTP scheme. Also known as tree-clustering



Tree-Decompositions

A *tree decomposition* for a belief network $BN = \langle X, D, G, P \rangle$ is a triple $\langle T, \chi, \psi \rangle$, where $T = (V, E)$ is a tree and χ and ψ are labeling functions, associating with each vertex $v \in V$ two sets, $\chi(v) \subseteq X$ and $\psi(v) \subseteq P$ satisfying :

1. For each function $p_i \in P$ there is exactly one vertex such that $p_i \in \psi(v)$ and $\text{scope}(p_i) \subseteq \chi(v)$
2. For each variable $X_i \in X$ the set $\{v \in V / X_i \in \chi(v)\}$ forms a connected subtree (running intersection property)

Tree-width: maximum number of nodes in a node of Tree-decomposition – 1

Seperator-width: maximum intersection between adjacent nodes

Eliminator: $\text{elim}(u, v) = \chi(u) - \chi(v)$



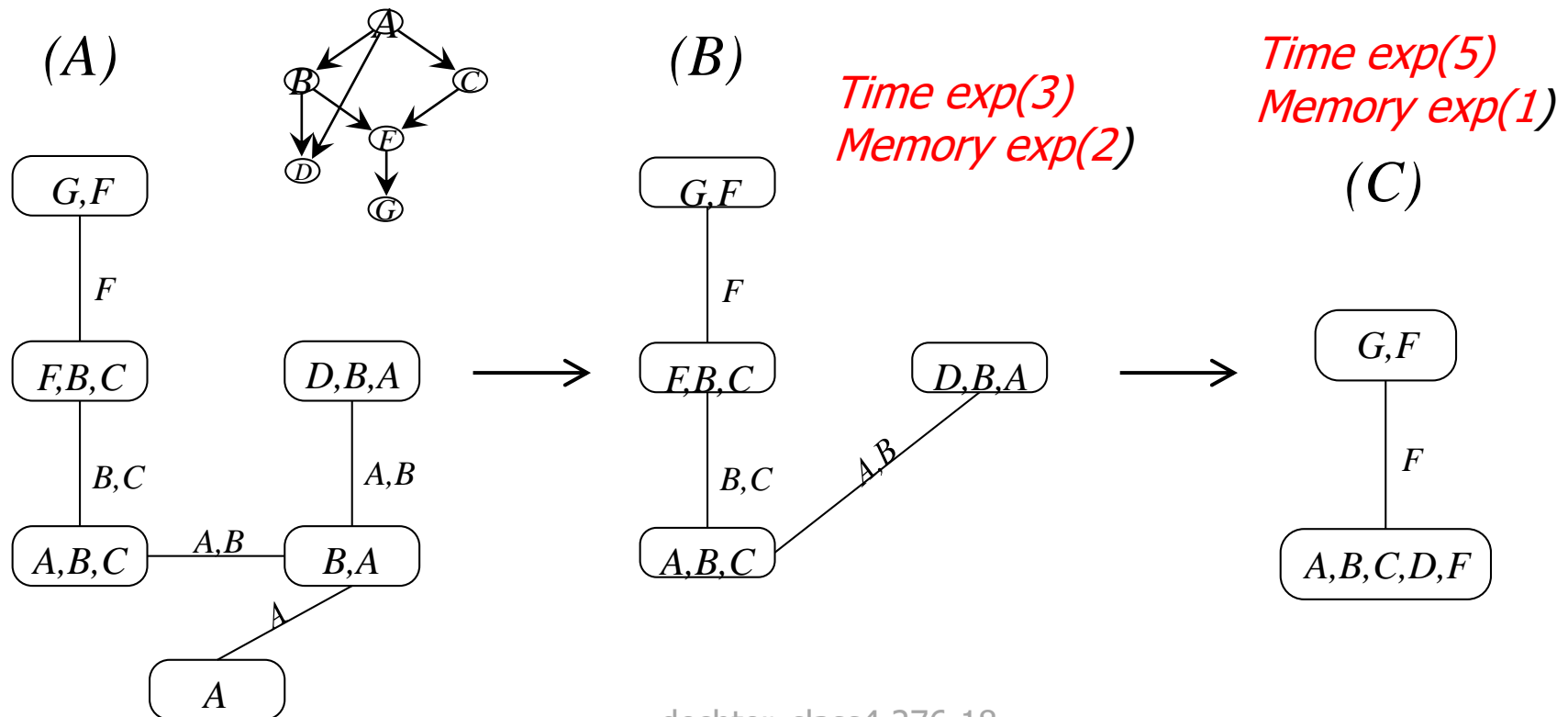
Generating Tree-Decompositions

Proposition 6.2.12 *If T is a tree-decomposition, then any tree obtained by merging adjacent clusters is also a tree-decomposition.*

A bucket-tree of a graphical model is a tree-decomposition of the model

From Buckets to Clusters

- Merge non-maximal buckets into maximal clusters.
- Connect clusters into a tree: each cluster to one with which it shares a largest subset of variables.
- Separators are variable- intersection on adjacent clusters.





Cluster-Tree Elimination

CLUSTER-TREE ELIMINATION (CTE)

Input: A tree decomposition $\langle T, \chi, \psi \rangle$ for a problem $M = \langle X, D, F, \Pi, \Sigma \rangle$,
 $X = \{X_1, \dots, X_n\}$, $F = \{f_1, \dots, f_r\}$. Evidence $E = e$, $\psi_u = \prod_{f \in \psi(u)} f$

Output: An augmented tree decomposition whose clusters are all model explicit.

Namely, a decomposition $\langle T, \chi, \bar{\psi} \rangle$ where $u \in T$, $\bar{\psi}(u)$ is model explicit relative to $\chi(u)$.

1. **Initialize.** (denote by $m_{u \rightarrow v}$ the message sent from vertex u to vertex v .)
2. **Compute messages:**
 - For every node u in T , once u received messages from all neighbors but v ,
 - Process observed variables:**
 - For each node $u \in T$ assign relevant evidence to $\psi(u)$
 - Compute the message:**
 - $$m_{u \rightarrow v} \leftarrow \sum_{\chi(u) - \text{sep}(u, v)} \psi_u \cdot \prod_{r \in \text{neighbor}(u), r \neq v} m_{r \rightarrow u}$$
 - endfor**

Note: functions whose scopes do not contain any separator variable do not need to be combined and can be directly passed on to the receiving vertex.

3. **Return:** The explicit tree $\langle T, \chi, \bar{\psi} \rangle$, where

$\bar{\psi}(v) \leftarrow \psi(v) \cup_{u \in \text{neighbor}(v)} \{m_{u \rightarrow v}\}$

return the explicit function: for each v , $M_{\chi(v)} = \prod_{f \in \bar{\psi}(v)} f$

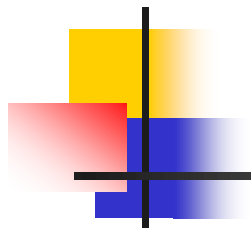


Properties of CTE

- Correctness and completeness: Algorithm CTE is correct, i.e. it computes the exact joint probability of a single variable and the evidence.
- Time complexity:
 - $O(deg \times (n+N) \times d^{w^*+1})$
- Space complexity: $O(N \times d^{sep})$

where

 - deg = the maximum degree of a node
 - n = number of variables (= number of CPTs)
 - N = number of nodes in the tree decomposition
 - d = the maximum domain size of a variable
 - w^* = the induced width, treewidth
 - sep = the separator size

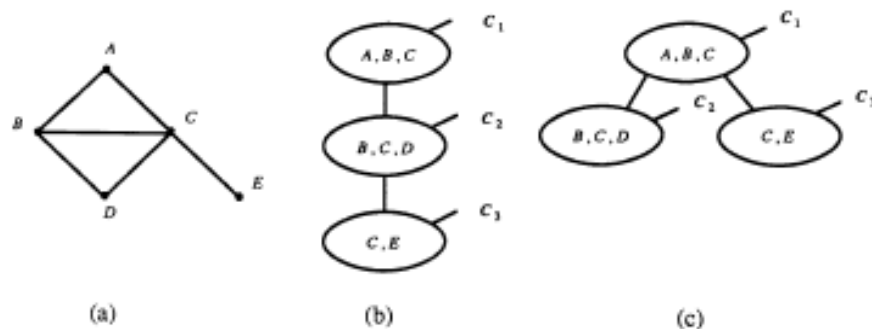


Generating Join-trees (Junction-trees); a special type of tree-decompositions



ASSEMBLING A JOIN TREE

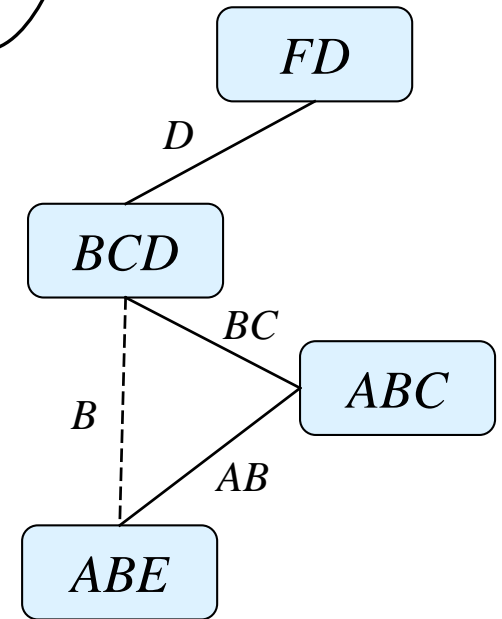
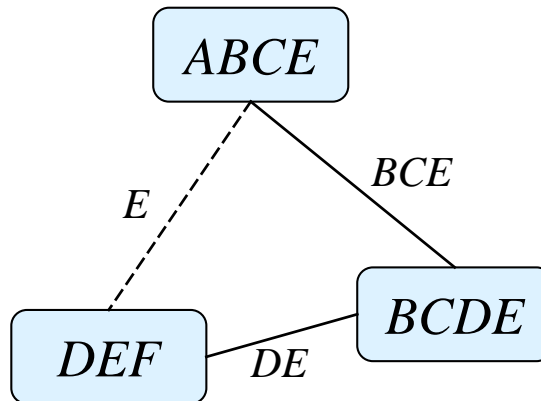
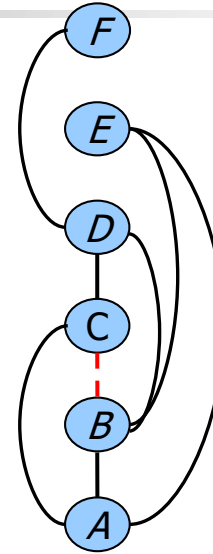
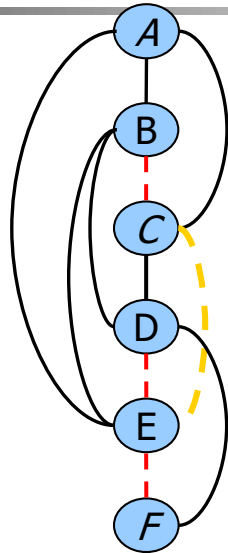
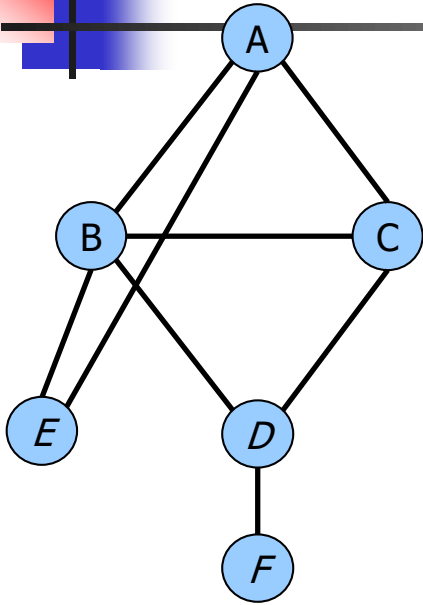
1. Use the fill-in algorithm to generate a chordal graph G' (if G is chordal, $G = G'$).
2. Identify all cliques in G' . Since any vertex and its parent set (lower ranked nodes connected to it) form a clique in G' , the maximum number of cliques is $|V|$.
3. Order the cliques C_1, C_2, \dots, C_t by rank of the highest vertex in each clique.
4. Form the join tree by connecting each C_i to a predecessor C_j ($j < i$) sharing the highest number of vertices with C_i .



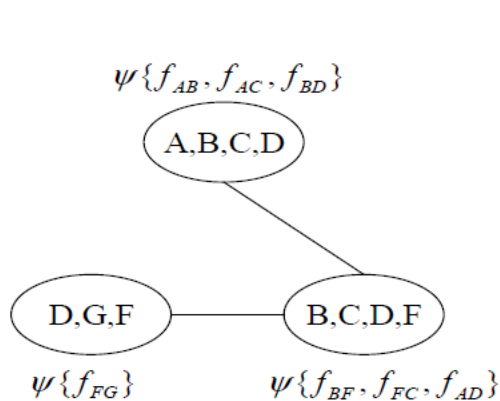
EXAMPLE: Consider the graph in Figure 3.9a. One maximum cardinality ordering is (A, B, C, D, E) .

- Every vertex in this ordering has its preceding neighbors already connected, hence the graph is chordal and no edges need be added.
- The cliques are ranked C_1 , C_2 , and C_3 as shown in Figure 3.9b.
- $C_3 = \{C, E\}$ shares only vertex C with its predecessors C_2 and C_1 , so either one can be chosen as the parent of C_3 .
- These two choices yield the join trees of Figures 3.9b and 3.9c.
- Now suppose we wish to assemble a join tree for the same graph with the edge (B, C) missing.
- The ordering (A, B, C, D, E) is still a maximum cardinality ordering, but now when we discover that the preceding neighbors of node D (i.e., B and C) are nonadjacent, we should fill in edge (B, C) .
- This renders the graph chordal, and the rest of the procedure yields the same join trees as in Figures 3.9b and 3.9c.

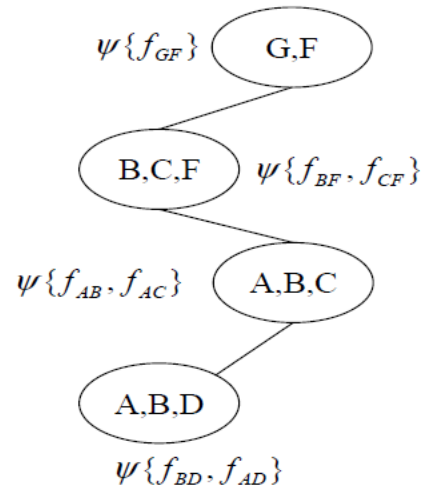
Examples of (Join)-Trees Construction



Tree-clustering and message-passing

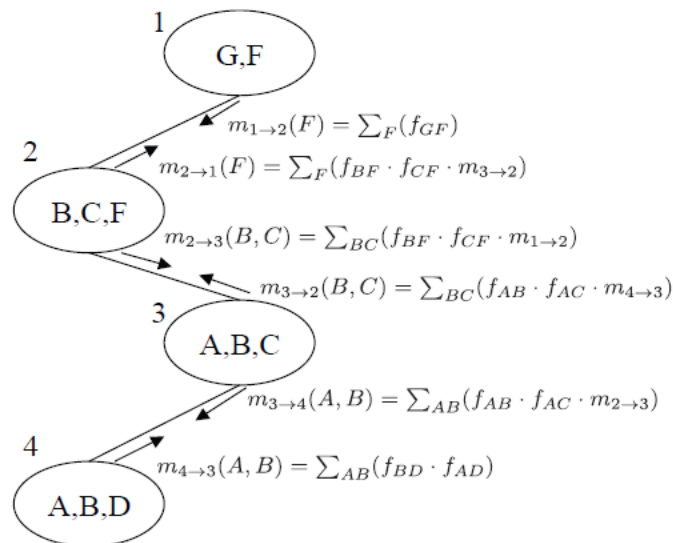
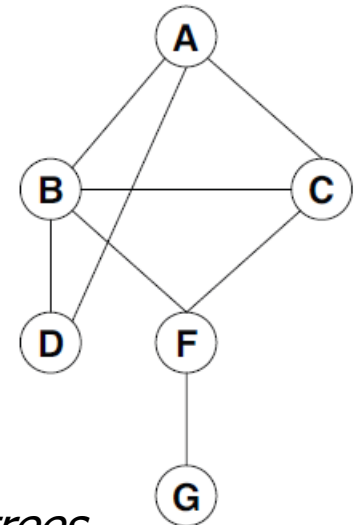


(a)



(b)

Two join-trees



*Message-passing by CTE on
The tree in (b)*



Outline

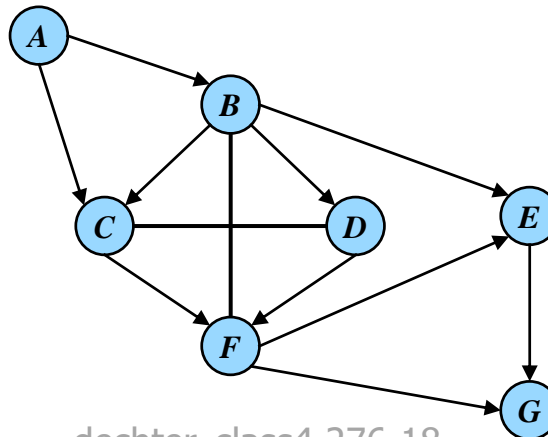
- From bucket-elimination (BE) to bucket-tree elimination (BTE)
- From BTE to CTE, Acyclic networks, the join-tree algorithm
- Examples of CTE for Bayesian network
- Belief-propagation on acyclic probabilistic networks (poly-trees) and Loopy networks

Tree Decompositions

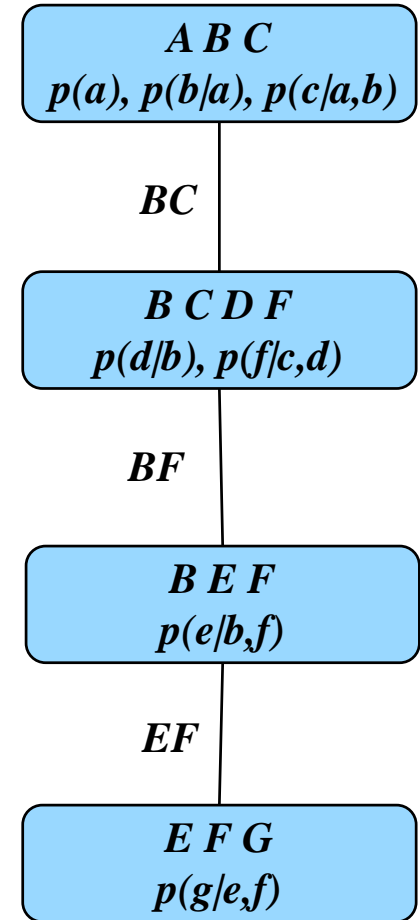
A *tree decomposition* for a graphical model $\langle X, D, P \rangle$ is a triple $\langle T, \chi, \psi \rangle$, where $T = (V, E)$ is a tree and χ and ψ are labeling functions, associating with each vertex $v \in V$ two sets, $\chi(v) \subseteq X$ and $\psi(v) \subseteq P$ satisfying :

1. For each function $p_i \in P$ there is exactly one vertex such that $p_i \in \psi(v)$ and $\text{scope}(p_i) \subseteq \chi(v)$
2. For each variable $X_i \in X$ the set $\{v \in V \mid X_i \in \chi(v)\}$ forms a connected subtree (running intersection property)

*Connectedness, or
Running intersection property*

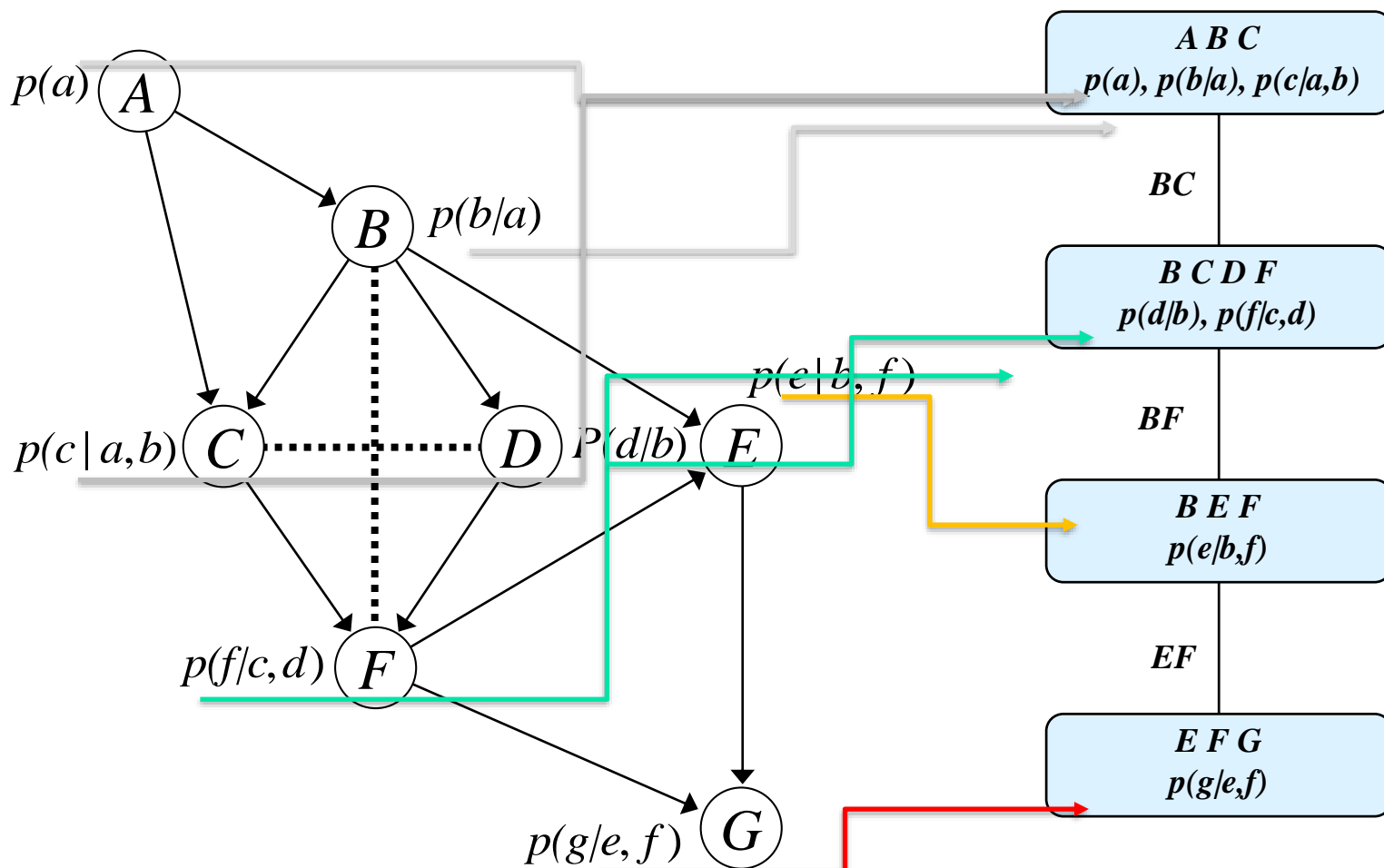


dechter, class4 276-18

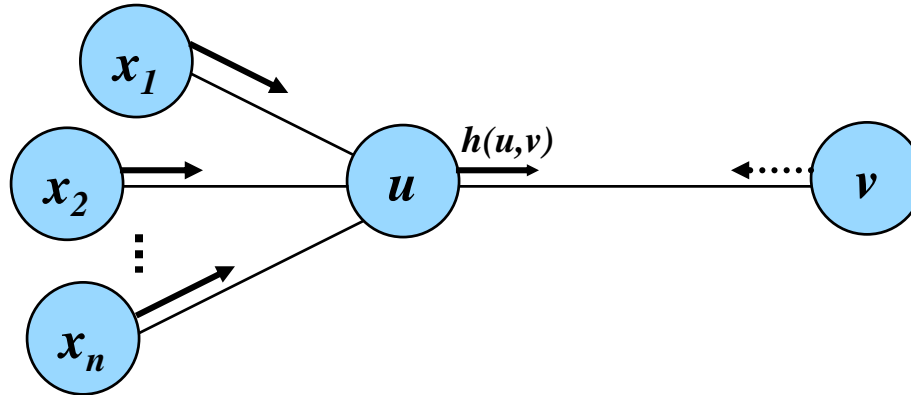


Tree decomposition

Example of a Tree Decomposition



Message passing on a tree decomposition



$$cluster(u) = \psi(u) \cup \{h(x_1, u), h(x_2, u), \dots, h(x_n, u), h(v, u)\}$$

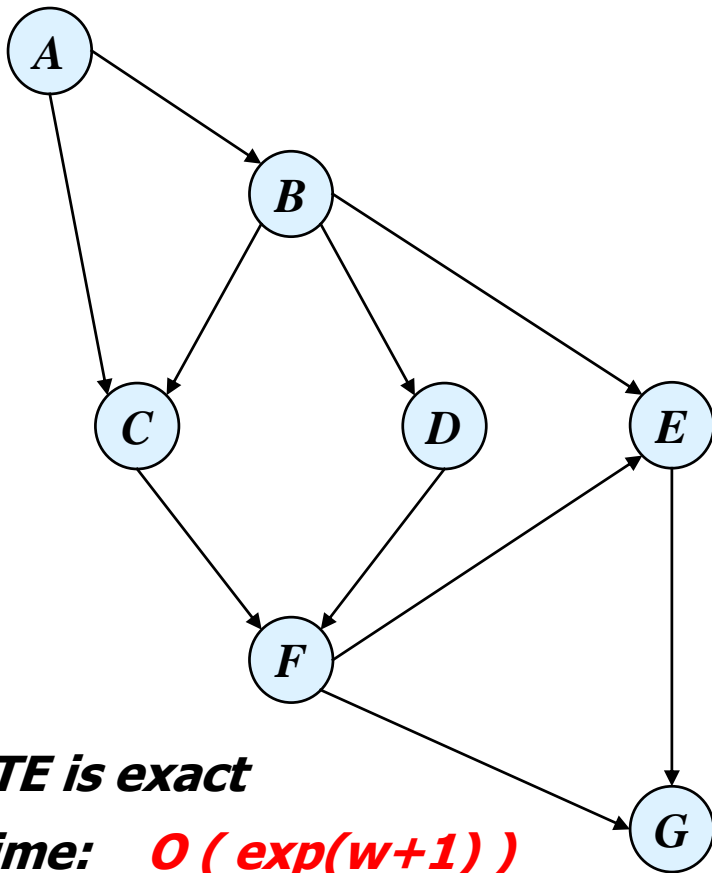
*For max-product
Just replace Σ
With max.*

Compute the message :

$$h(u, v) = \sum_{elim(u, v)} \prod_{f \in cluster(u) - \{h(v, u)\}} f$$

$$Elim(u, v) = cluster(u) - sep(u, v)$$

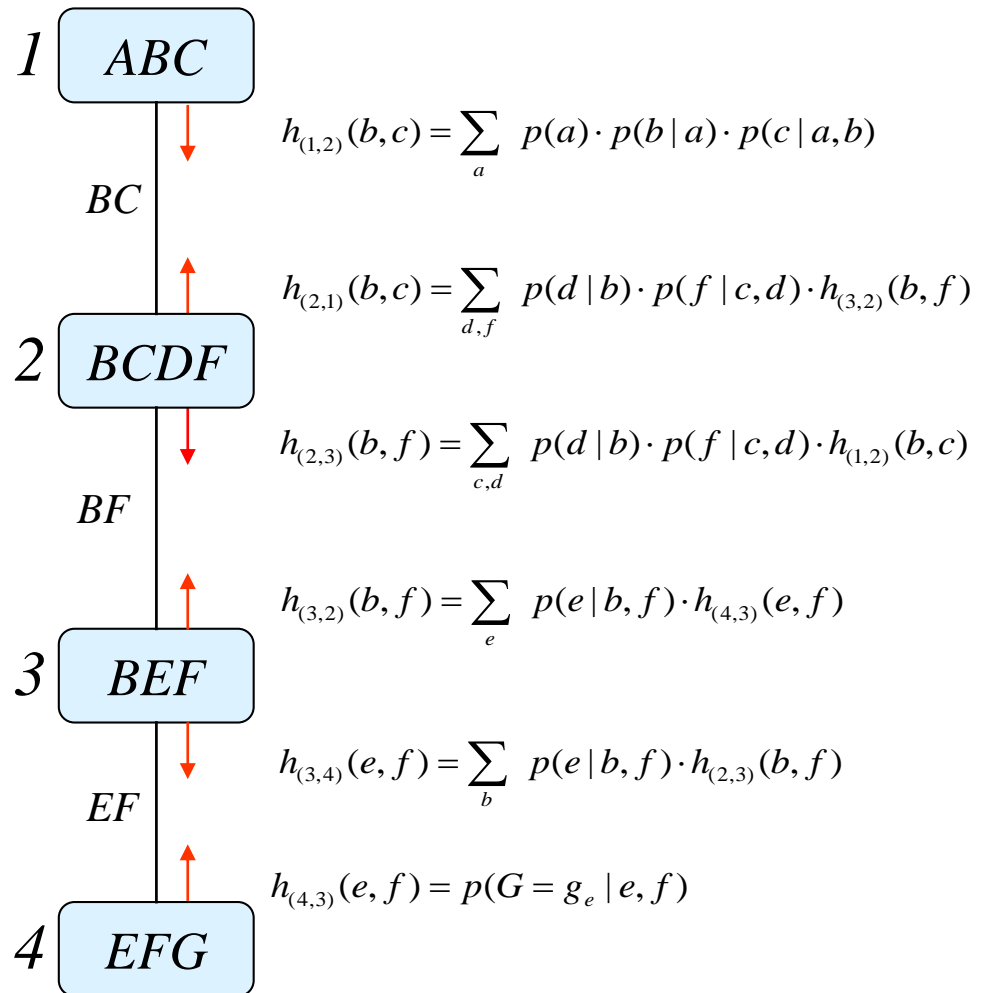
Cluster-Tree Elimination (CTE), or Join-Tree Message-passing



CTE is exact

Time: $O(\exp(w+1))$

Space: $O(\exp(sep))$



For each cluster $P(X|e)$ is computed, also $P(e)$



Outline

- From bucket-elimination (BE) to bucket-tree elimination (BTE)
- From BTE to CTE, Acyclic networks, the join-tree algorithm
- Examples of CTE for Bayesian network
- Belief-propagation on acyclic probabilistic networks (poly-trees) and Loopy networks

Polytrees and Acyclic Networks

- **Polytree:** a BN whose undirected skeleton is a tree
- **Acyclic network:** A network is acyclic if it has a tree-decomposition where each node has a single original CPT.
- A polytree is an acyclic model.

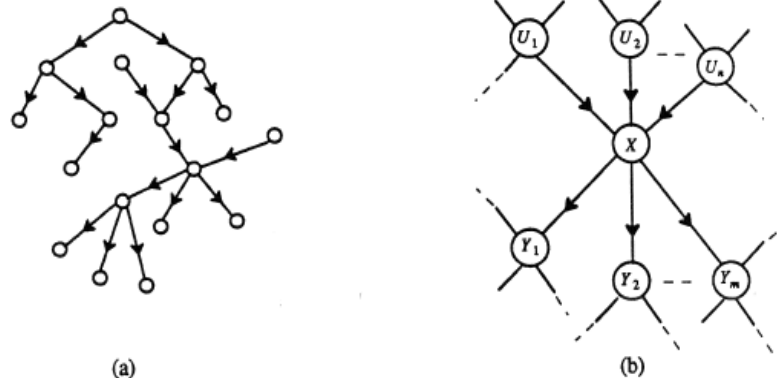
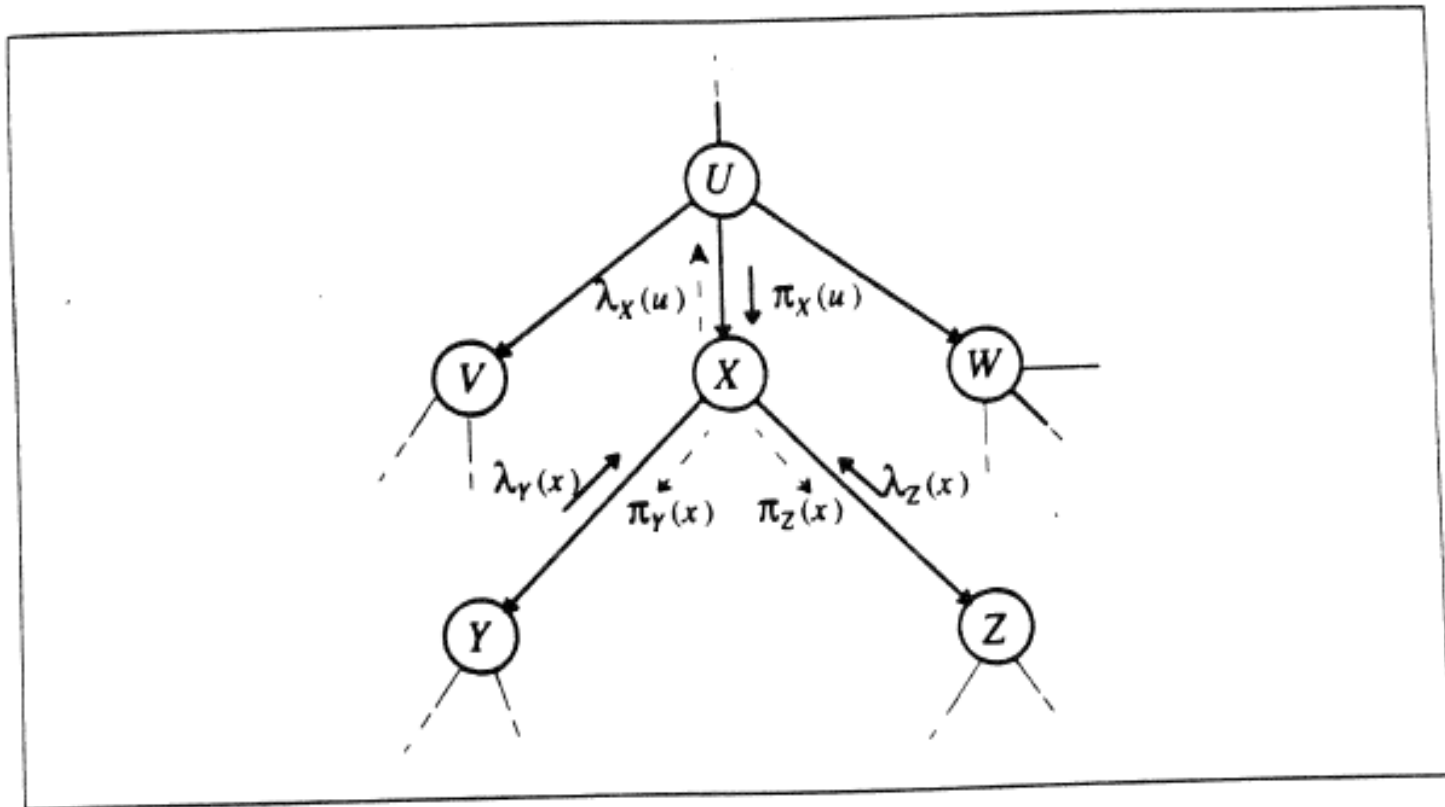


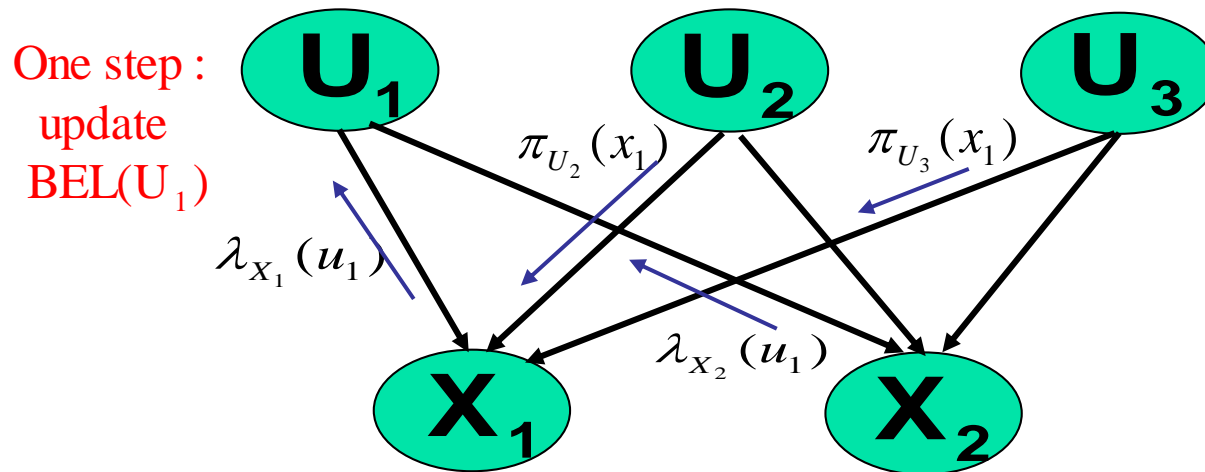
Figure 4.18. (a) A fragment of a polytree and (b) the parents and children of a typical node X .

Pearl's Belief Propagation



From Exact to Approximate: Iterative Belief Propagation

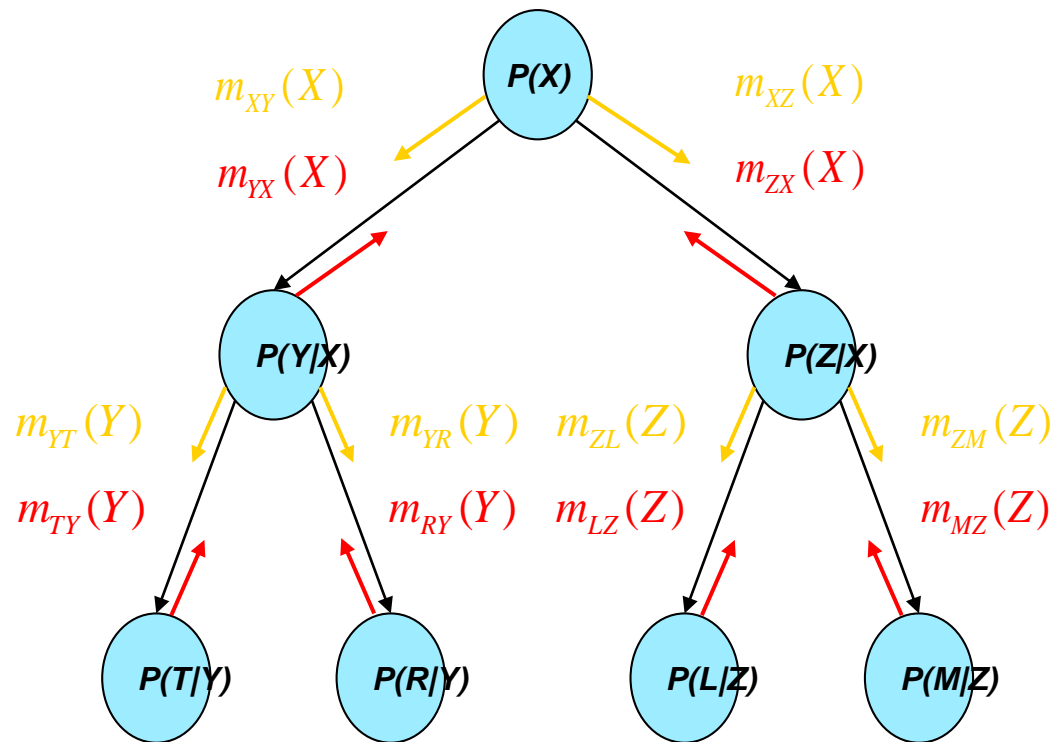
- Belief propagation is exact for poly-trees
- IBP - applying BP iteratively to cyclic networks

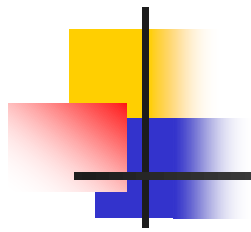


- No guarantees for convergence
- Works well for many coding networks

Propagation in both directions

- Messages can propagate both ways and we get beliefs for each variable





The end