Reasoning with Graphical Models

Slides Set 4:  
Exact Inference Algorithms  
Bucket-elimination

Rina Dechter

(Dechter chapter 4, Darwiche chapter 6)
Inference for probabilistic networks

- Bucket elimination (Dechter chapter 4)
  - Belief-updating, P(e), partition function
  - Marginals, probability of evidence
  - The impact of evidence
  - for MPE (→ MAP)
  - for MAP (→ Marginal Map)
- Induced-Width (Dechter, Chapter 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)
- Induced-Width
Bayesian Networks: Example
(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

<table>
<thead>
<tr>
<th>$A$</th>
<th>$\Theta_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>.6</td>
</tr>
<tr>
<td>false</td>
<td>.4</td>
</tr>
</tbody>
</table>

| $A$ | $B$ | $\Theta_B|A$ |
|-----|-----|------------|
| true | true | .2 |
| true | false | .8 |
| false | true | .75 |
| false | false | .25 |

| $B$ | $C$ | $D$ | $\Theta_D|BC$ |
|-----|-----|-----|-------------|
| 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 |
### Types of queries

<table>
<thead>
<tr>
<th>Types</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max-Inference</td>
<td>$f(x^*) = \max_x \prod_{\alpha} f_\alpha(x_\alpha)$</td>
</tr>
<tr>
<td>Sum-Inference</td>
<td>$Z = \sum_x \prod_{\alpha} f_\alpha(x_\alpha)$</td>
</tr>
<tr>
<td>Mixed-Inference</td>
<td>$f(x^*<em>M) = \max</em>{x_M} \sum_{x_S} \prod_{\alpha} f_\alpha(x_\alpha)$</td>
</tr>
</tbody>
</table>

- **NP-hard**: exponentially many terms
- We will focus on exact and then 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

\[(\neg u \lor \neg w \lor y) \land (u \lor \neg v \lor w)\]
A Simple Network

How can we compute $P(D)$, $P(D|A=0)$, $P(A|D=0)$?

- Brute force $O(k^4)$
- Maybe $O(4k^2)$
Elimination as a Basis for Inference

$$A \rightarrow B \rightarrow C$$

To compute the prior marginal on variable $C$, $\Pr(C)$, we first eliminate variable $A$ and then variable $B$.

<table>
<thead>
<tr>
<th>$A$</th>
<th>$\Theta_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>.6</td>
</tr>
<tr>
<td>false</td>
<td>.4</td>
</tr>
</tbody>
</table>

| $A$ | $B$ | $\Theta_{B|A}$ |
|-----|-----|---------------|
| true| true| .9            |
| true| false| .1           |
| false| true| .2            |
| false| false| .8           |

| $B$ | $C$ | $\Theta_{C|B}$ |
|-----|-----|---------------|
| true| true| .3            |
| true| false| .7           |
| false| true| .5            |
| false| false| .5           |
Elimination as a Basis for Inference

- There are two factors that mention variable $A$, $\Theta_A$ and $\Theta_{B|A}$.
- We multiply these factors first and then sum out variable $A$ from the resulting factor.
- Multiplying $\Theta_A$ and $\Theta_{B|A}$:

| $A$  | $B$   | $\Theta_A \Theta_{B|A}$ |
|------|-------|-------------------------|
| true | true  | .54                     |
| true | false | .06                     |
| false| true  | .08                     |
| false| false | .32                     |

- Summing out variable $A$:

| $B$ | $\sum_A \Theta_A \Theta_{B|A}$ |
|-----|---------------------------------|
| true| $.62 = .54 + .08                |
| false| $.38 = .06 + .32               |
We now have two factors, \( \sum_A \Theta_A \Theta_{B|A} \) and \( \Theta_{C|B} \), and we want to eliminate variable \( B \).

Since \( B \) appears in both factors, we must multiply them first and then sum out \( B \) from the result.

**Multiplying:**

| \( B \) | \( C \) | \( \sum_A \Theta_A \Theta_{B|A} \Theta_{C|B} \) |
|--------|--------|------------------------------------------|
| true   | true   | .186                                    |
| true   | false  | .434                                    |
| false  | true   | .190                                    |
| false  | false  | .190                                    |

**Summing out:**

| \( C \) | \( \sum_B \sum_A \Theta_A \Theta_{B|A} \Theta_{C|B} \) |
|---------|-----------------------------------------------|
| true    | .376                                          |
| false   | .624                                          |
Belief Updating

\[ P(\text{lung cancer}=\text{yes} \mid \text{smoking}=\text{no}, \text{dyspnoea}=\text{yes}) = ? \]
Belief updating: $P(X|\text{evidence}) =$

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

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

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

Variable Elimination

$h^B(a,d,c,e)$
Bucket elimination
Algorithm \textit{BE-bel} (Dechter 1996)

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

\begin{align*}
\text{bucket } B: & \quad P(b | a) \quad P(d | b, a) \quad P(e | b, c) \\
\text{bucket } C: & \quad P(c | a) \lambda^B (a, d, c, e) \\
\text{bucket } D: & \quad \lambda^C (a, d, e) \\
\text{bucket } E: & \quad e=0 \lambda^D (a, e) \\
\text{bucket } A: & \quad P(a) \lambda^E (a) \\
\end{align*}

\[ \sum_b \prod \quad \text{Elimination operator} \]

\[ W^* = 4 \]

"induced width" (max clique size)

\[ P(a, e=0) = \frac{P(a, e=0)}{P(e=0)} \]

slides 4 COMPSCI 2020
A Bayesian Network
Ordering: A,C,B,E,D,G

\[ P(a, g = 1) = \sum_{c,b,e,d,g=1} P(a, b, c, d, e, g) = \sum_{c,b,f,d,g=1} P(g|f)P(f|b,c)P(d|a,b)P(c|a)P(b|a)P(a). \]

\[ P(a, g = 1) = P(a) \sum_c P(c|a) \sum_b P(b|a) \sum_f P(f|b,c) \sum_d P(d|b,a) \sum_{g=1} P(g|f). \] \tag{4.1} \]

\[ P(a, g = 1) = P(a) \sum_c P(c|a) \sum_b P(b|a) \sum_f P(f|b,c) \lambda_G(f) \sum_d P(d|b,a). \] \tag{4.2} \]

\[ P(a, g = 1) = P(a) \sum_c P(c|a) \sum_b P(b|a) \lambda_D(a,b) \sum_f P(f|b,c) \lambda_G(f) \] \tag{4.3} \]

\[ P(a, g = 1) = P(a) \sum_c P(c|a) \sum_b P(b|a) \lambda_D(a,b) \lambda_F(b,c) \] \tag{4.4} \]

\[ P(a, g = 1) = P(a) \sum_c P(c|a) \lambda_B(a,c) \] \tag{4.5} \]
A Bayesian Network
Ordering: A,C,B,E,D,G

\[
P(a, g = 1) = \sum_{c,b,e,d,g=1} P(c|a) \sum_{b} P(b|a) \sum_{f} P(f|b, c) \sum_{d} P(d|b, a) \sum_{g=1} P(g|f).
\]

\[
P(a, g = 1) = P(a) \sum_{c} P(c|a) \sum_{b} P(b|a) \sum_{f} P(f|b, c) \sum_{d} P(d|b, a) \sum_{g=1} P(g|f).
\]  \hspace{1cm} (4.1)

\[
P(a, g = 1) = P(a) \sum_{c} P(c|a) \sum_{b} P(b|a) \sum_{f} P(f|b, c) \lambda_G(f) \sum_{d} P(d|b, a).
\]  \hspace{1cm} (4.2)

\[
P(a, g = 1) = P(a) \sum_{c} P(c|a) \sum_{b} P(b|a) \lambda_D(a,b) \sum_{f} P(f|b, c) \lambda_G(f)
\]  \hspace{1cm} (4.3)

\[
P(a, g = 1) = P(a) \sum_{c} P(c|a) \sum_{b} P(b|a) \lambda_D(a,b) \lambda_F(b,c)
\]  \hspace{1cm} (4.4)

\[
P(a, g = 1) = P(a) \sum_{c} P(c|a) \lambda_B(a,c)
\]  \hspace{1cm} (4.5)
A Bayesian Network
Ordering: A, C, B, F, D, G

(a) Directed acyclic graph
(b) Moral graph
A Different Ordering

**Ordering:** A, F, D, C, B, G

\[
P(a, g = 1) = P(a) \sum_f \sum_d \sum_c P(c|a) \sum_b P(b|a) \; P(d|a, b) P(f|b, c) \sum_{g=1}^d P(g|f) \\
= P(a) \sum_f \lambda_G(f) \sum_d \sum_c P(c|a) \sum_b P(b|a) \; P(d|a, b) P(f|b, c) \\
= P(a) \sum_f \lambda_G(f) \sum_d \sum_c P(c|a) \lambda_B(a, d, c, f) \\
= P(a) \sum_f \lambda_G(f) \sum_d \lambda_C(a, d, f) \\
= P(a) \sum_f \lambda_G(f) \lambda_D(a, f) \\
= P(a) \lambda_F(a)
\]

Figure 4.3: The bucket's output when processing along \(d_2 = A, F, D, C, B, G\)
A Different Ordering

Ordering: \(A,F,D,C,B,G\)

\[
P(a, g = 1) = P(a) \sum_f \sum_d \sum_c P(c|a) \sum_b P(b|a) P(d|a, b) P(f|b, c) \sum_{g=1} P(g|f) \\
= P(a) \sum_f \lambda_G(f) \sum_d \sum_c P(c|a) \sum_b P(b|a) P(d|a, b) P(f|b, c) \\
= P(a) \sum_f \lambda_G(f) \sum_d \lambda_B(a, d, c, f) \\
= P(a) \sum_f \lambda_G(f) \sum_d \lambda_C(a, d, f) \\
= P(a) \sum_f \lambda_G(f) \lambda_D(a, f) \\
= P(a) \lambda_F(a)
\]
A Bayesian Network Processed Along 2 Orderings

\[ d1 = A, C, B, F, D, G \]

Figure 4.4: The bucket’s output when processing along \( d2 = A, F, D, C, B, G \).
The Operation In a Bucket

- Multiplying functions
- Marginalizing (summing-out) functions
Combination of Cost Functions

\[
\begin{align*}
\text{A} & \quad \text{B} & \quad \text{f(A,B)} \\
\text{b} & \quad \text{b} & \quad 0.4 \\
\text{b} & \quad \text{g} & \quad 0.1 \\
\text{g} & \quad \text{b} & \quad 0 \\
\text{g} & \quad \text{g} & \quad 0.5 \\
\end{align*}
\]

\[
\begin{align*}
\text{B} & \quad \text{C} & \quad \text{f(B,C)} \\
\text{b} & \quad \text{b} & \quad 0.2 \\
\text{b} & \quad \text{g} & \quad 0 \\
\text{g} & \quad \text{b} & \quad 0 \\
\text{g} & \quad \text{g} & \quad 0.8 \\
\end{align*}
\]

\[
\begin{align*}
\text{A} & \quad \text{B} & \quad \text{C} & \quad \text{f(A,B,C)} \\
\text{b} & \quad \text{b} & \quad \text{b} & \quad 0.1 \\
\text{b} & \quad \text{b} & \quad \text{g} & \quad 0 \\
\text{b} & \quad \text{g} & \quad \text{b} & \quad 0 \\
\text{b} & \quad \text{g} & \quad \text{g} & \quad 0.08 \\
\text{g} & \quad \text{b} & \quad \text{b} & \quad 0 \\
\text{g} & \quad \text{b} & \quad \text{g} & \quad 0 \\
\text{g} & \quad \text{g} & \quad \text{b} & \quad 0 \\
\text{g} & \quad \text{g} & \quad \text{g} & \quad 0.4 \\
\end{align*}
\]

\[= 0.1 \times 0.8\]
Factors: Sum-Out Operation

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

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

<table>
<thead>
<tr>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
<th>$f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>.95</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>.05</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>.9</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>.1</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>.8</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>.2</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$B$</th>
<th>$C$</th>
<th>$\sum_D f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>1</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>1</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>1</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>1</td>
</tr>
</tbody>
</table>

$$
\sum_B \sum_C \sum_D f_1
$$

Thanks to Darwiche
Bucket Elimination and Induced Width

Ordering: a, e, d, c, b

\[
\begin{align*}
\text{bucket}(B) &= P(e|b, c), P(d|a, b), P(b|a) \\
\text{bucket}(C) &= P(c|a) \parallel \lambda_B(a, c, d, e) \\
\text{bucket}(D) &= \parallel \lambda_C(a, d, e) \\
\text{bucket}(E) &= e = 0 \parallel \lambda_D(a, c) \\
\text{bucket}(A) &= P(a) \parallel \lambda_E(a)
\end{align*}
\]
Bucket Elimination and Induced Width

Ordering: \(a, b, c, d, e\)
- \(\text{bucket}(E) = P(e|b, c), \ e = 0\)
- \(\text{bucket}(D) = P(d|a, b)\)
- \(\text{bucket}(C) = P(c|a) \ || \ P(e = 0|b, c)\)
- \(\text{bucket}(B) = P(b|a) \ || \ \lambda_D(a, b), \lambda_C(b, c)\)
- \(\text{bucket}(A) = P(a) \ || \ \lambda_B(a)\)

Ordering: \(a, e, d, c, b\)
- \(\text{bucket}(B) = P(e|b, c), P(d|a, b), P(b|a)\)
- \(\text{bucket}(C) = P(c|a) \ || \ \lambda_B(a, c, d, e)\)
- \(\text{bucket}(D) = \ || \ \lambda_C(a, d, e)\)
- \(\text{bucket}(E) = e = 0 \ || \ \lambda_D(a, c)\)
- \(\text{bucket}(A) = P(a) \ || \ \lambda_E(a)\)

\(W^* = 2\)
\(W^* = 4\)
Algorithm BE-bel

Input: A belief network $B = \langle X, D, P_G, \prod \rangle$, an ordering $d = (X_1, \ldots, X_n)$; evidence $e$

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

1. Partition the input functions (CPTs) into $bucket_1, \ldots, bucket_n$ as follows:
   for $i \leftarrow n$ down to 1, put in $bucket_i$ all unplaced functions mentioning $X_i$.
   Put each observed variable in its bucket. Denote by $\psi_i$ the product of input functions in $bucket_i$.

2. backward: for $p \leftarrow n$ down to 1 do

3. for all the functions $\psi_{S_0}, \lambda_{S_1}, \ldots, \lambda_{S_f}$ 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. $\lambda_p \leftarrow \sum_{X_p} \psi_p \cdot \prod_{i=1}^{f} \lambda_{S_i}$

5. place $\lambda_p$ 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.
Last slide.
Belief Updating

Algorithm BE-bel [Dechter 1996]

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

bucket \( B \):

\[
p(b | A) p(d | b, A) p(e | b, c)
\]

bucket \( C \):

\[
p(c | A) \lambda_{B\rightarrow C}(A, d, c, e)
\]

bucket \( D \):

\[
\lambda_{C\rightarrow D}(A, d, e)
\]

bucket \( E \):

\[
\mathbb{1}[E = 0] \lambda_{D\rightarrow E}(A, e)
\]

bucket \( A \):

\[
p(A) \lambda_{E\rightarrow A}(A)
\]

\[
p(A | E = 0) = p(A, E = 0) / p(E = 0)
\]

W* = 4

“induced width” (max clique size)

slides4 COMPSCI 2020
Bucket Elimination

Algorithm BE-bel [Dechter 1996]

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

\[ \sum_{b} \prod_{A} \text{Elimination & combination operators} \]

Time and space exponential in the induced-width / treewidth

\[ p(A|E = 0) = \frac{p(A, E = 0)}{p(E = 0)} \]

slides4 COMPSCI 2020
Student Network Example

- $P(J)$?
Induced Width (continued)

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

**The effect of the ordering:**

\[ \begin{align*}
  w^*(d_1) &= 4 \\
  w^*(d_2) &= 2
\end{align*} \]
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)
- Induced-Width
The impact of evidence?

Algorithm BE-bel

\[ P(A \mid E = 0) = \alpha \sum_{E=0,D,C,B} P(A) \cdot P(B \mid A) \cdot P(C \mid A) \cdot P(D \mid A,B) \cdot P(E \mid B,C) \]

Elimination operator

bucket B: \( P(b \mid a) \cdot P(d \mid b,a) \cdot P(e \mid b,c) \)

bucket C: \( P(c \mid a) \cdot \lambda^B(a, d, c, e) \)

bucket D: \( \lambda^C(a, d, e) \)

bucket E: \( e=0 \cdot \lambda^D(a, e) \)

bucket A: \( P(a) \cdot \lambda^E(a) \)

\( P(a \mid e=0) \)

\( P(e=0) \)

\( W^* = 4 \)

"induced width" (max clique size)

slides4 COMPSCI 2020
The impact of evidence?

Algorithm *BE-bel*

\[
P(A \mid E = 0) = \alpha \sum_{E=0,D,C,B} P(A) \cdot P(B \mid A) \cdot P(C \mid A) \cdot P(D \mid A,B) \cdot P(E \mid B,C)
\]

\[P(A\mid E=0,B=1)?\]

**bucket B:** \[P(b\mid a) \quad P(d\mid b,a) \quad P(e\mid b,c)\]

**bucket C:** \[P(c\mid a)\]

**bucket D:** \[P(d\mid b=1,a)\]

**bucket E:** \[e=0\]

**bucket A:** \[P(a) \quad P(b=1\mid a)\]

\[P(e=0)\]

\[P(a\mid e=0) = \frac{P(a,e=0)}{P(e=0)}\]
The impact of observations

Ordered graph  Induced graph  Ordered conditioned graph
Pruning Nodes: Example

Example of pruning irrelevant subnetworks

network structure joint on $B$, $E$ joint on $B$
Pruning Nodes

Given a Bayesian network $\mathcal{N}$ and query $(Q, e)$, one can remove any leaf node (with its CPT) from the network as long as it does not belong to variables $Q \cup E$, yet not affect the ability of the network to answer the query correctly.

If $\mathcal{N}' = \text{pruneNodes}(\mathcal{N}, Q \cup E)$

then $Pr(Q, e) = Pr'(Q, e)$, where $Pr$ and $Pr'$ are the probability distributions induced by networks $\mathcal{N}$ and $\mathcal{N}'$, respectively.
Pruning Nodes: Example

Example of pruning irrelevant subnetworks

network structure  joint on $B, E$  joint on $B$
Pruning Edges: Example

Example of pruning edges due to evidence or conditioning

\[ A \quad B \quad \Theta_{B|A} \]
\[
\begin{array}{ccc}
true & true & .2 \\
true & false & .8 \\
false & true & .75 \\
false & false & .25 \\
\end{array}
\]

\[ A \quad C \quad \Theta_{C|A} \]
\[
\begin{array}{ccc}
true & true & .8 \\
true & false & .2 \\
false & true & .1 \\
false & false & .9 \\
\end{array}
\]

\[ A \quad Θ_A \]
\[
\begin{array}{ccc}
true & .6 \\
false & .4 \\
\end{array}
\]

\[ B \quad D \quad \sum_C Θ_{C=false}^{D=B|BC} \]
\[
\begin{array}{ccc}
true & true & .9 \\
true & false & .1 \\
false & true & 0 \\
false & false & 1 \\
\end{array}
\]

\[ E \quad \sum_C Θ_{C=false}^{E|C} \]
\[
\begin{array}{ccc}
true & 0 \\
false & 1 \\
\end{array}
\]

Evidence e : C = false
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)
- Induced-Width
MPE = \max_{\bar{x}} P(\bar{x})

\sum \text{ is replaced by } \max:

MPE = \max_{a,e,d,c,b} P(a)P(c \mid a)P(b \mid a)P(d \mid a,b)P(e \mid b,c)
MPE = \max \; P(\bar{x})

\[ \sum \text{ is replaced by } \max : \]

\[ MPE = \max_{a,e,d,c,b} P(a)P(c \mid a)P(b \mid a)P(d \mid a,b)P(e \mid b,c) \]

**Bucket B:** \( P(b \mid a) \) \( P(d \mid b,a) \) \( P(e \mid b,c) \)

**Bucket C:** \( P(c \mid a) \) \( h^B(a, d, c, e) \)

**Bucket D:** \( h^C(a, d, e) \)

**Bucket E:** \( e=0 \) \( h^D(a, e) \)

**Bucket A:** \( P(a) \) \( h^E(a) \) MPE

"induced width" (max clique size) W*=4

slides4 COMPSCI 2020
Generating the MPE-tuple

1. \( a' = \text{arg max}_a P(a) \cdot h^E(a) \)

2. \( e' = 0 \)

3. \( d' = \text{arg max}_d h^C(a', d, e') \)

4. \( c' = \text{arg max}_c P(c | a') \times h^B(a', d', c, e') \)

5. \( b' = \text{arg max}_b P(b | a') \times P(d' | b, a') \times P(e' | b, c') \)

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 $d$.

Induced Width

slide4 COMPSCI 2020
Complexity of Bucket Elimination

**Bucket-Elimination is time and space**

$$O\left(r \exp(w^*_d)\right)$$

$w^*_d$: the induced width of the primal graph along ordering $d$

$r = \text{number of functions}$

The effect of the ordering:

$w^*(d_1) = 4$

$w^*(d_2) = 2$

Finding smallest induced-width is hard!
Example with mpe?

A Bayesian Network

<table>
<thead>
<tr>
<th>A</th>
<th>(\Theta_A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>.6</td>
</tr>
<tr>
<td>false</td>
<td>.4</td>
</tr>
</tbody>
</table>

| A | B | \(\Theta_B|A\) |
|---|---|---|
| true | true | .2 |
| true | false | .8 |
| false | true | .75 |
| false | false | .25 |

| 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 |
Try to compute MPE when $E=0$
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

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

\[
\min_B \sum_{C(a, b, d), \ C(b, f), \ C(b, c)}
\]

Figure 5.12: Schematic execution of BE-Opt
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)

- **Induced-Width**
### Marginal Map

<table>
<thead>
<tr>
<th>Inference Type</th>
<th>Mathematical Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Max-Inference</strong></td>
<td>$f(x^*) = \max_x \prod_{\alpha} f_\alpha(x_\alpha)$</td>
</tr>
<tr>
<td><strong>Sum-Inference</strong></td>
<td>$Z = \sum_x \prod_{\alpha} f_\alpha(x_\alpha)$</td>
</tr>
<tr>
<td><strong>Mixed-Inference</strong></td>
<td>$f(x_M^*) = \max_{x_M} \sum_{x_S} \prod_{\alpha} f_\alpha(x_\alpha)$</td>
</tr>
</tbody>
</table>

- **NP-hard**: exponentially many terms
Example for MMAP Applications

- Haplotype in Family pedigrees
- Coding networks
- Probabilistic planning
- Diagnosis
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!

\[ \sum \text{max} \neq \text{max} \sum \]
Bucket Elimination for MMAP

Bucket Elimination

\[ \mathbf{X}_M = \{A, D, E\} \]
\[ \mathbf{X}_S = \{B, C\} \]

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

\[ \lambda^E(A) \]
\[ \max_E \]

\[ \lambda(D, E) f(A, D) \]
\[ \sum_C \]

\[ \lambda^C(A, D, E) f(A, C) f(C, E) \]
\[ \sum_B \]

\[ f(A, B) f(B, C) f(B, D) f(B, E) \]

MAP* is the marginal MAP value
Why is MMAP harder?

\( X_M = \{A, D, E\} \)
\( X_S = \{B, C\} \)

\( w^* = 4 \)

\( w^* = 2 \)

In practice, constrained induced is much larger!

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

\[
P(x) = \frac{1}{Z} \sum_{x \in \mathcal{D}} \prod_{C \in \mathcal{C}} \Psi_C(x_C)
\]

\[
Z = \sum_{x} \prod_{C \in \mathcal{C}} \Psi_C(x_C)
\]  

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

\[
P(a, b, c, d, f, g) = \frac{\Psi(a, b, c) \cdot \Psi(b, c, f) \cdot \Psi(a, b, d) \cdot \Psi(f, g)}{Z}
\]

where,

\[
Z = \sum_{a,b,c,d,e,f,g} \Psi(a, b, c) \cdot \Psi(b, c, f) \cdot \Psi(a, b, d) \cdot \Psi(f, g)
\]
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)

- **Induced-Width** (Dechter 3.4,3.5)
Finding a Small Induced-Width

- NP-complete
- A tree has induced-width of ?
- Greedy algorithms:
  - Min width
  - Min induced-width
  - Max-cardinality and chordal graphs
  - Fill-in (thought as the best)

- Anytime algorithms
  - Search-based [Gogate & Dechter 2003]
  - Stochastic (CVO) [Kask, Gelfand & Dechter 2010]
Min-width Ordering

**MIN-WIDTH (MW)**

**input:** a graph $G = (V, E)$, $V = \{v_1, ..., v_n\}$

**output:** A min-width ordering of the nodes $d = (v_1, ..., v_n)$.

1. for $j = n$ to 1 by -1 do
2.     $r \leftarrow$ a node in $G$ with smallest degree.
3.     put $r$ in position $j$ and $G \leftarrow G - r$.
       (Delete from $V$ node $r$ and from $E$ all its adjacent edges)
4. endfor

**Proposition:** algorithm min-width finds a min-width ordering of a graph

**What is the Complexity of MW?**

$O(e)$

class2 828X 2019
Greedy Orderings Heuristics

- **Min-induced-width**
  - From last to first, pick a node with smallest width, then connect parent and remove

- **Min-Fill**
  - From last to first, pick a node with smallest fill-edges

*Complexity? $O(n^3)$*
Min-Fill Heuristic

- Select the variable that creates the fewest “fill-in” edges

Eliminate B next?
Connect neighbors
“Fill-in” = 3: (A,D), (C,E), (D,E)

Eliminate E next?
Neighbors already connected
“Fill-in” = 0
Example
Different Induced-Graphs

(a) A Min-fill ordering

(b) A Miw ordering

(c) A Miw ordering

(d) A Miw ordering

E F A B C D
A graph is chordal if every cycle of length at least 4 has a chord.

Deciding chordality by max-cardinality ordering:
- from 1 to n, always assigning a next node connected to a largest set of previously selected nodes.

A graph along max-cardinality order has no fill-in edges iff it is chordal.

The maximal cliques of chordal graphs form a tree.

[Tarjan & Yanakakis 1980]
Greedy Orderings Heuristics

- **Min-Induced-width**
  - From last to first, pick a node with smallest width

- **Min-Fill**
  - From last to first, pick a node with smallest fill-edges
    - *Complexity?  $O(n^3)$*

- **Max-Cardinality search**  [Tarjan & Yanakakis 1980]
  - From **first to last**, pick a node with largest neighbors already ordered.
    - *Complexity?  $O(n + m)$*
Max-cardinality ordering

MAX-CARDINALITY (MC)

**input:** a graph $G = (V, E)$, $V = \{v_1, ..., v_n\}$

**output:** An ordering of the nodes $d = (v_1, ..., v_n)$.

1. Place an arbitrary node in position 0.
2. for $j = 1$ to $n$ do
3.  $r \leftarrow$ a node in $G$ that is connected to a largest subset of nodes in positions 1 to $j - 1$, breaking ties arbitrarily.
4. endfor

Proposition 5.3.3 [56] Given a graph $G = (V, E)$ the complexity of max-cardinality search is $O(n + m)$ when $|V| = n$ and $|E| = m$. 
Example

We see again that $G$ in the Figure (a) is not chordal since the parents of $A$ are not connected in the max-cardinality ordering in Figure (d). If we connect $B$ and $C$, the resulting induced graph is chordal.
Which Greedy Algorithm is Best?

- Min-Fill, prefers a node who add the least number of fill-in arcs.

- Empirically, fill-in is the best among the greedy algorithms (MW, MIW, MF, MC)

- Complexity of greedy orderings?
  - MW is $O(e)$, MIW: $O(n^3)$ MF $O(n^3)$ MC is $O(e+n)$
Definition 5.3.4 (k-trees) A subclass of chordal graphs are k-trees. A k-tree is a chordal graph whose maximal cliques are of size $k+1$, and it can be defined recursively as follows: (1) A complete graph with $k$ vertices is a k-tree. (2) A k-tree with $r$ vertices can be extended to $r+1$ vertices by connecting the new vertex to all the vertices in any clique of size $k$. A partial k-tree is a k-tree having some of its arcs removed. Namely it will clique of size smaller than $k$. 
Finding a Small Induced-Width

- NP-complete
- A tree has induced-width of ?
- Greedy algorithms:
  - Min width (MW)
  - Min induced-width (MIW)
  - Max-cardinality and chordal graphs (MC)
  - Min-Fill (thought as the best) (MIN-FILL)

- Anytime algorithms
  - Search-based  [Gogate & Dechter 2003]
  - Stochastic (CVO)  [Kask, Gelfand & Dechter 2010]
Algorithms for Reasoning with graphical models

Slides Set 5:
Exact Inference Algorithms
Bucket-elimination

Rina Dechter
(Dechter chapter 5, Darwiche chapter 6)
Inference for probabilistic networks

- **Bucket elimination**
  - Belief-updating, $P(e)$, partition function
  - Marginals, probability of evidence
  - The impact of evidence
  - for MPE (→ MAP)
  - for MAP (→ Marginal Map)
  - Mixed networks

- **Tree-decomposition schemes**
  - Bucket tree elimination
  - Cluster tree elimination
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 that 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 buckets + messages

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

\[
P(D) = \sum_{a,b} P(D|a,b)\pi_{B\rightarrow D}(a,b)
\]
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 $\pi$. 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 $\pi$ messages from its parent bucket and all the $\lambda$ 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.
**Theorem:** When BTE terminates, the product of functions in each bucket is the beliefs of the variables joint with the evidence.

\[ Elim(i,j) = B_i - B_j \]
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 $\mathcal{M} = \langle X, D, F, \prod, \sum \rangle$, ordering $d$. $X = \{X_1, ..., X_n\}$ and $F = \{f_1, ..., f_r\}$, $E = e$. 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 $B'_1, ..., B'_n$, where each $B'_i$ contains the original bucket functions and the $\lambda$ messages it received.
Computing Marginal Beliefs

Input: a bucket tree processed by BTE with augmented buckets: $B_{i_1}, \ldots, B_{i_n}$

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

\[ bel(B_i) \leftarrow \alpha \cdot \prod_{f \in B_{i_t}} f \]
\[ bel(X_i) \leftarrow \alpha \cdot \sum_{B_i - \{X_i\}} \prod_{f \in B_{i_t}} f \]
\[ P(\text{evidence}) \leftarrow \sum_{B_i} \prod_{f \in B_{i_t}} f \]

Figure 5.4: Query answering.
Explicit functions

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

$$\mathcal{M}_Y = \sum \prod_{X-Y \ f \in F} f,$$

(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 $F$. We say that $(Y, F_Y)$ is an explicit submodel of $\mathcal{M}$ iff

$$\prod_{f \in F_Y} f = \mathcal{M}_Y$$

(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 $\mathcal{M} = (X, D, F, \prod, \sum)$, $r$ be the number of functions in $F$ and $k$ be the maximum domain size. The time complexity of BTE is $O(r \cdot \text{deg} \cdot k^{w^*(d)+1})$, where $\text{deg}$ 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
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
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 $\chi$ and $\psi$ 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)

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

**Separator-width:** maximum intersection between adjacent nodes

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

\[ (A) \]

\[ G,F \]
\[ F,B,C \]
\[ F,B,C \]
\[ A,B,C \]
\[ A,B,C \]
\[ A,B \]
\[ B,C \]

\[ (B) \]

\[ G,F \]
\[ F,B,C \]
\[ D,B,A \]
\[ A,B \]
\[ B,A \]

\[ (C) \]

\[ G,F \]
\[ F \]
\[ D,B,A \]
\[ A,B \]
\[ B,C \]

Time \( \exp(3) \)
Memory \( \exp(2) \)

Time \( \exp(5) \)
Memory \( \exp(1) \)
CLUSTER-TREE ELIMINATION (CTE)

**Input:** A tree decomposition $< T, \chi, \psi >$ for a problem $M = < X, D, F, \Pi, \Sigma >$, $X = \{X_1, ..., X_n\}$, $F = \{f_1, ..., f_r\}$. 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 $< T, \chi, \bar{\psi} >$ 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 $< T, \chi, \bar{\psi} >$, 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:
  - $O ( \text{deg} \times (n+N) \times d^{w*+1} )$

- Space complexity: $O ( N \times d^{\text{sep}} )$
  
  where
  - $\text{deg} = \text{the maximum degree of a node}$
  - $n = \text{number of variables (} = \text{number of CPTs)}$
  - $N = \text{number of nodes in the tree decomposition}$
  - $d = \text{the maximum domain size of a variable}$
  - $w^* = \text{the induced width, treewidth}$
  - $\text{sep} = \text{the separator size}$
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 $C_1, C_2, ..., C_t$ 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 preceeding 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

```
A
 / \
B   C
 |   |
D   E
 |   |
F   F

ABCE

DEF

BCDE

BCD

ABE

ABC

FD

D

BC

AB
```

slides4 COMPSCI 2020
Tree-clustering and message-passing

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
A tree decomposition for a graphical model $<X,D,P>$ is a triple $<T, \chi, \psi>$, where $T = (V,E)$ is a tree and $\chi$ and $\psi$ 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)

### Example

**Tree decomposition**

$A B C$

$p(a), p(b|a), p(c|a,b)$

$B C D F$

$p(d|b), p(f|c,d)$

$B E F$

$p(e|b,f)$

$E F G$

$p(g|e,f)$

*Connectedness, or Running intersection property*
Example of a Tree Decomposition
Message passing on a tree decomposition

For max-product
Just replace $\Sigma$
With max.

\[
\text{cluster}(u) = \psi(u) \cup \{h(x_1, u), h(x_2, u), \ldots, h(x_n, u), h(v, u)\}
\]

Compute the message:

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

\[
\text{Elim}(u, v) = \text{cluster}(u) - \text{sep}(u, v)
\]
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
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