Local Structure and BE extensions

COMPSCI 276, Spring 2017
Set 5b: Rina Dechter

(Reading: Darwiche chapter 5, dechter chapter 4)
Outline

• Special representations of CPTs
• Bucket Elimination:
  • Finding induced-width
  • Bucket elimination over mixed networks
Outline

- Bayesian networks and queries
- Building Bayesian Networks
- Special representations of CPTs
  - Causal Independence (e.g., Noisy OR)
  - Context Specific Independence
  - Determinism
  - Mixed Networks
# Dealing with Large CPTs

## The size of a CPT for binary variable $E$ with binary parents $C_1, \ldots, C_n$

<table>
<thead>
<tr>
<th>Number of Parents: $n$</th>
<th>Parameter Count: $2^n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
</tr>
<tr>
<td>10</td>
<td>1024</td>
</tr>
<tr>
<td>20</td>
<td>1,048,576</td>
</tr>
<tr>
<td>30</td>
<td>1,073,741,824</td>
</tr>
</tbody>
</table>
Think about headache and 10 different conditions that may cause it.

A noisy-or circuit

A micro model

details the relationship between a variable \( E \) and its parents \( C_1, \ldots, C_n \).

We wish to specify cpt with less parameters
Binary OR

\[ P(X=0|A,B) \]

\[ P(X=1|A,B) \]

| A | B | P(X=0|A,B) | P(X=1|A,B) |
|---|---|------------|------------|
| 0 | 0 | 1          | 0          |
| 0 | 1 | 0          | 1          |
| 1 | 0 | 0          | 1          |
| 1 | 1 | 0          | 1          |
Noisy-or Model

- Cause $C_i$ is capable of establishing effect $E$, except under some unusual circumstances summarized by suppressor $Q_i$.

- When suppressor $Q_i$ is active, $C_i$ is no longer able to establish $E$.

- The leak variable $L$ represents all other causes of $E$ which were not modeled explicitly.

- When none of the causes $C_i$ are active, the effect $E$ may still be established by the leak variable $L$. 
Noisy-or Model

The noisy-or model requires $n + 1$ parameters.
Noisy-or Model

The noisy-or model requires $n + 1$ parameters.

To model the relationship between headache and ten different conditions

- $\theta_{q_i} = \Pr(Q_i = \text{active})$: probability that suppressor of $C_i$ is active.
- $\theta_l = \Pr(L = \text{active})$: probability that leak is active.
Let $l_\alpha$ be the indices of causes that are active in $\alpha$. 
Noisy-or Model

- Let $l_\alpha$ be the indices of causes that are active in $\alpha$.
- If

\[\alpha: C_1 = \text{active}, C_2 = \text{active}, C_3 = \text{passive}, C_4 = \text{passive}, C_5 = \text{active},\]

then $l_\alpha = \{1, 2, 5\}$. 
Let $l_\alpha$ be the indices of causes that are active in $\alpha$.

If

$\alpha: C_1 = \text{active}, C_2 = \text{active}, C_3 = \text{passive}, C_4 = \text{passive}, C_5 = \text{active},$

then $l_\alpha = \{1, 2, 5\}$.

We then have

$\Pr(E = \text{passive}|\alpha) = (1 - \theta_I) \prod_{i \in l_\alpha} \theta_{q_i}$

$\Pr(E = \text{active}|\alpha) = 1 - \Pr(E = \text{passive}|\alpha)$. 
Noisy-or Model

- Let \( l_\alpha \) be the indices of causes that are active in \( \alpha \).
- If

\[ \alpha: C_1 = \text{active}, \ C_2 = \text{active}, \ C_3 = \text{passive}, \ C_4 = \text{passive}, \ C_5 = \text{active}, \]

then \( l_\alpha = \{1, 2, 5\} \).
- We then have

\[
\Pr(E = \text{passive}|\alpha) = (1 - \theta_l) \prod_{i \in l_\alpha} \theta_{q_i} \\
Pr(E = \text{active}|\alpha) = 1 - \Pr(E = \text{passive}|\alpha).
\]

The full CPT for variable \( E \), with its \( 2^n \) parameters, can be induced from the \( n + 1 \) parameters of the noisy-or model.
Noisy-or Model

Example

Sore throat ($S$) has three causes: cold ($C$), flu ($F$), tonsillitis ($T$).
Noisy-or Model

Example
Sore throat \( (S) \) has three causes: cold \( (C) \), flu \( (F) \), tonsillitis \( (T) \).

If we assume that \( S \) is related to its causes by a noisy-or model
we can then specify the CPT for \( S \) by the following four probabilities:

- The suppressor probability for cold, say .15
- The suppressor probability for flu, say, .01
- The suppressor probability for tonsillitis, say .05
- The leak probability, say .02
Example

Sore throat ($S$) has three causes: cold ($C$), flu ($F$), tonsillitis ($T$).
## Noisy-or Model

### Example

Sore throat ($S$) has three causes: cold ($C$), flu ($F$), tonsillitis ($T$).

The CPT for sore throat is then determined completely as follows:

| $C$ | $F$ | $T$ | $S$   | $\theta_{s|c,f,t}$ | $1 - (1 - .02)(.15)(.01)(.05)$ |
|-----|-----|-----|-------|---------------------|---------------------------------|
| true| true| true| true  | 0.9999265           | $1 - (1 - .02)(.15)(.01)(.05)$ |
| true| true| false| true  | 0.99853             | $1 - (1 - .02)(.15)(.01)$      |
| true| false| true| true  | 0.99265             | $1 - (1 - .02)(.15)(.05)$      |
|     |     |     | true  | .02                 | $1 - (1 - .02)$                 |
Noisy/OR CPDs

Figure 11: The COPs network for diagnosis of internal diseases. The network contains 438 nodes, 906 links.
## Decision Trees

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>Pr(E=1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.9</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
</tr>
</tbody>
</table>

- **C1**
  - Pr(E=1) = 0.0
- **C2**
  - Pr(E=1) = 0.9
- **C3**
  - Pr(E=1) = 0.3
- **C4**
  - Pr(E=1) = 0.6, 0.8
A CPT for variable $E$ can be represented using a set of if-then rules of the form

If $\alpha_i$ then $P_x(e) = p_i$, for each value $e$ of variable $E$, where $\alpha_i$ is a propositional sentence constructed using the parents of variable $E$. 
A CPT for variable $E$ can be represented using a set of if-then rules of the form

If $\alpha_i$ then $\Pr(e) = p_i$, for each value $e$ of variable $E$, where $\alpha_i$ is a propositional sentence constructed using the parents of variable $E$.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Then</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>If $C_1 = 1$</td>
<td>then</td>
<td>$\Pr(E = 1) = 0.0$</td>
</tr>
<tr>
<td>If $C_1 = 0 \land C_2 = 1$</td>
<td>then</td>
<td>$\Pr(E = 1) = 0.9$</td>
</tr>
<tr>
<td>If $C_1 = 0 \land C_2 = 0 \land C_3 = 1$</td>
<td>then</td>
<td>$\Pr(E = 1) = 0.3$</td>
</tr>
<tr>
<td>If $C_1 = 0 \land C_2 = 0 \land C_3 = 0 \land C_4 = 1$</td>
<td>then</td>
<td>$\Pr(E = 1) = 0.6$</td>
</tr>
<tr>
<td>If $C_1 = 0 \land C_2 = 0 \land C_3 = 0 \land C_4 = 0$</td>
<td>then</td>
<td>$\Pr(E = 1) = 0.8$</td>
</tr>
</tbody>
</table>
If-Then Rules

A CPT for variable $E$ can be represented using a set of if-then rules of the form

If $\alpha_i$ then $\Pr(e) = p_i$, for each value $e$ of variable $E$, where $\alpha_i$ is a propositional sentence constructed using the parents of variable $E$.

For the rule-based representation to be complete and consistent

- The premises $\alpha_i$ must be mutually exclusive. That is, $\alpha_i \land \alpha_j$ is inconsistent for $i \neq j$. This ensures that the rules will not conflict with each other.
- The premises $\alpha_i$ must be exhaustive. That is, $\bigvee_i \alpha_i$ must be valid. This ensures that every CPT parameter $\theta_{e | \ldots}$ is implied by the rules.
A student’s example

Difficulty

Intelligence

Grade

SAT

Letter

Apply

Job

Apply

Job

Letter

Grade

SAT

Intelligence

Difficulty
Tree CPD

If the student does not **Apply**, **SAT** and **L** are irrelevant

Tree-CPD for job  

Causal Indepedence
Captures irrelevant variables
Multiplexer CPD

A CPD $P(Y|A,Z_1,Z_2,\ldots,Z_k)$ is a multiplexer iff
Val($A$)=1,2,…,$k$, and

$P(Y|A,Z_1,\ldots,Z_k)=Z_a$
Figure 1: A mixture of trees over a domain consisting of random variables $V = \{a, b, c, d, e\}$, where $z$ is a hidden choice variable. Conditional on the value of $z$, the dependency structure is a tree. A detailed presentation of the mixture-of-trees model is provided in Section 3.

Meila and Jordan, 2000
Mixture model with shared structure

Figure 4: A mixture of trees with shared structure (MTSS) represented as a Bayes net (a) and as a Markov random field (b).

Meila and Jordan, 2000
### Deterministic CPTs

Can we use hidden variables?

| $A$  | $X$   | $C$    | $\theta_{c|a,x}$ |
|------|-------|--------|-------------------|
| high | ok    | high   | 0                 |
| low  | ok    | high   | 1                 |
| high | stuckat0 | high | 0                 |
| low  | stuckat0 | high | 0                 |
| high | stuckat1 | high | 1                 |
| low  | stuckat1 | high | 1                 |

We can represent this CPT as follows:

\[
(X = \text{ok} \land A = \text{high}) \lor X = \text{stuckat0} \iff C = \text{low}
\]
\[
(X = \text{ok} \land A = \text{low}) \lor X = \text{stuckat1} \iff C = \text{high}
\]
Mixed Networks
(Dechter 2013)

Augmenting Probabilistic networks with constraints because:

- Some information in the world is deterministic and undirected ($X \neq Y$)
- Some queries are complex or evidence are complex (cnfs)

Queries are probabilistic queries
Probabilistic Reasoning

Party example: the weather effect

Alex is **likely** to go in bad weather
Chris **rarely** goes in bad weather
Becky is indifferent but **unpredictable**

Questions:

*Given bad weather, which group of individuals is most likely to show up at the party?*

*What is the probability that Chris goes to the party but Becky does not?*

\[ P(W,A,C,B) = P(B|W) \cdot P(C|W) \cdot P(A|W) \cdot P(W) \]
\[ P(A,C,B|W=bad) = 0.9 \cdot 0.1 \cdot 0.5 \]
Query: 
*Is it likely that Chris goes to the party if Becky does not but the weather is bad?*

\[ P(C, \neg B \mid w = bad, A \rightarrow B, C \rightarrow A) \]
Outline

- Special representations of CPTs
- **Bucket Elimination:**
  - Finding induced-width
  - Bucket elimination over mixed networks
Complexity of bucket elimination

Theorem
Given a belief network having $n$ variables, observations $e$, the complexity of elim-mpe, elim-bel, elim-map along $d$, is time and space $O(n \exp(w^*(+1) + 1))$ and $O(n \exp(w^*))$, respectively

where $w^*(d)$ is the induced width of the moral graph whose edges connecting evidence to earlier nodes, were deleted.

More accurately: $O(r \exp(w^*(d)))$ where $r$ is the number of cpts. For Bayesian networks $r=n$. For Markov networks?
Finding Small Induced-Width
(Dechter 3.4-3.5)

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)
See anytime min-width (Gogate and Dechter)
Type of graphs

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
The induced width

Definition 5.2.1 (width) Given an undirected graph $G = (V, E)$, an ordered graph is a pair $(G, d)$, where $V = \{v_1, ..., v_n\}$ is the set of nodes, $E$ is a set of arcs over $V$, and $d = (v_1, ..., v_n)$ is an ordering of the nodes. The nodes adjacent to $v$ that precede it in the ordering are called its parents. The width of a node in an ordered graph is its number of parents. The width of an ordering $d$ of $G$, denoted $w_d(G)$ (or $w_d$ for short) is the maximum width over all nodes. The width of a graph is the minimum width over all the orderings of the graph.

Definition 5.2.3 (induced width) The induced width of an ordered graph $(G, d)$, denoted $w^*_d$, is the width of the induced ordered graph along $d$ obtained as follows: nodes are processed from last to first; when node $v$ is processed, all its parents are connected. The induced width of a graph, denoted by $w^*$, is the minimal induced width over all its orderings. Formally

$$w^*(G) = \min_{d \in \text{orderings}} w^*_d(G)$$
Different Induced-graphs

(a)  

(b)  

(c)  

(d)
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: (Freuder 1982) algorithm min-width finds a min-width ordering of a graph. Complexity $O(|E|)$
Greedy Orderings Heuristics

**Theorem:** A graph is a tree iff it has both width and induced-width of 1.

**MIN-INDUCED-WIDTH (MIW)**

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

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

1. for $j = n$ to 1 by -1 do
2. \hspace{1em} $r \leftarrow$ a node in $V$ with smallest degree.
3. \hspace{1em} put $r$ in position $j$.
4. \hspace{1em} connect $r$'s neighbors: $E \leftarrow E \cup \{(v_i, v_j) | (v_i, r) \in E, (v_j, r) \in E\}$,
5. \hspace{1em} remove $r$ from the resulting graph: $V \leftarrow V - \{r\}$.

**MIN-FILL (MIN-FILL)**

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

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

1. for $j = n$ to 1 by -1 do
2. \hspace{1em} $r \leftarrow$ a node in $V$ with smallest fill edges for his parents.
3. \hspace{1em} put $r$ in position $j$.
4. \hspace{1em} connect $r$'s neighbors: $E \leftarrow E \cup \{(v_i, v_j) | (v_i, r) \in E, (v_j, r) \in E\}$,
5. \hspace{1em} remove $r$ from the resulting graph: $V \leftarrow V - \{r\}$.
Different Induced-Graphs

(a)  
(b)  
(c)  
(d)
Induced-width for chordal graphs

Definition: A graph is chordal if every cycle of length at least 4 has a chord

Finding $w^*$ over chordal graph is easy using the max-cardinality ordering: order vertices from 1 to $n$, always assigning the next number to the node connected to a largest set of previously numbered nodes. Let's $d$ be such an ordering

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

On chordal graphs width=induced-width.
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. \hspace{1em} $r \leftarrow$ 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 $|E| = m$.

What is the complexity of min-fill? Min-induced-width? $O(n^3)$
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 $k$. 
Which greedy algorithm is best?

- MinFill, 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 $O(n^2)$...maybe $O(n \log n + m)$?
  - MIW: $O(n^3)$,
  - MF ($O(n^3)$,
  - MC is $O(m+n)$, m edges.
Recent work in my group


Kask, Gelfand and Dechter, BEEM: Bucket Elimination with External memory, AAAI 2011 or UAI 2011

Potential project
Mixed Networks

Augmenting Probabilistic networks with constraints because:

Some information in the world is deterministic and undirected ($X \neq Y$).

Some queries are complex or evidence are complex (cnf formulas)

Queries are probabilistic queries
Mixed Beliefs and Constraints

If the constraint is a cnf formula

Queries over hybrid network:

Complex evidence structure

All reduce to cnf queries over a Belief network:

CPE (CNF probability evaluation): Given a belief network, and a cnf formula, find its probability.
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

\[ P_B(\varphi) = \sum_{x_\varphi \in Mod(\varphi)} P(x_\varphi) \]

Using the belief network product form we get:

\[ P_B(\varphi) = \sum_{\{x | x_\varphi \in Mod(\varphi)\}} \prod_{i=1}^{n} P(x_i | x_{pa_i}). \]

\[ P((C \rightarrow B) \text{ and } P(A \rightarrow C)) \]
Algorithm 1: BE-CPE

Input: A belief network $\mathcal{M} = (\mathcal{B}, \simeq)$, $\mathcal{B} = (\mathbf{X}, \mathbf{D}, \mathbf{P}_G, \prod)$, where $\mathcal{B} = \{P_1, \ldots, P_n\}$; a CNF formula on $k$ propositions $\varphi = \{\alpha_1, \ldots, \alpha_m\}$ defined over $k$ propositions; an ordering of the variables, $d = \{X_1, \ldots, X_n\}$.

Output: The belief $P(\varphi)$.

1. Place buckets with unit clauses last in the ordering (to be processed first).
   // Initialize
   Partition $\mathcal{B}$ and $\varphi$ into $\text{bucket}_1, \ldots, \text{bucket}_n$, where $\text{bucket}_i$ contains all the CPTs and clauses whose highest variable is $X_i$.
   Put each observed variable into its appropriate bucket. (We denote probabilistic functions by $\lambda$s and clauses by $\alpha$s).

2. for $p \leftarrow n$ downto 1 do  // Backward
   Let $\lambda_1, \ldots, \lambda_j$ be the functions and $\alpha_1, \ldots, \alpha_r$ be the clauses in $\text{bucket}_p$
   Process-\text{bucket}_p(\sum, (\lambda_1, \ldots, \lambda_j), (\alpha_1, \ldots, \alpha_r))

3. return $P(\varphi)$ as the result of processing $\text{bucket}_1$. 

Procedure `Process-bucket_p(\sum_i, (\lambda_1, \ldots, \lambda_j), (\alpha_1, \ldots, \alpha_r))`.

if `bucket_p` contains evidence `X_p = x_p` then

1. Assign `X_p = x_p` to each `\lambda_i` and put each resulting function in the bucket of its latest variable.
2. Resolve each `\alpha_i` with the unit clause, put non-tautology resolvents in the buckets of their latest variable and move any bucket with unit clause to top of processing.

else

\[ \lambda_p \leftarrow \sum_{x_p | x_{U_p} \in Mod(\alpha_1, \ldots, \alpha_r)} \prod_{i=1}^j \lambda_i \]

Add `\lambda_p` to the bucket of the latest variable in `S_p`, where `S_p = scope(\lambda_1, \ldots, \lambda_j, \alpha_1, \ldots, \alpha_r)`, `U_p = scope(\alpha_1, \ldots, \alpha_r)`.

---

(a) Directed acyclic graph

(b) Moral graph
Processing Mixed Buckets

We compute:

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

In Bucket $F$:
\[ \lambda_F(b, c, d) = \sum_f P(f \mid b, c) \lambda_G(f, d) \]

In Bucket $D$:
\[ \lambda_D(a, b, c) = \sum_{\{d \mid \neg d \vee \neg b = \text{true}\}} P(d \mid a, b) \lambda_F(b, c, d) \]

In Bucket $B$:
\[ \lambda_B(a, c) = \sum_{\{b \mid b \vee c = \text{true}\}} P(b \mid a) \lambda_D(a, b, c) \lambda_F(b, c) \]

In Bucket $C$:
\[ \lambda_C(a) = \sum_c P(c \mid a) \lambda_B(a, c) \]

In Bucket $A$:
\[ \lambda_A = \sum_a P(a) \lambda_C(a) \]

For example in bucket $G$, \( \lambda_G(f, d = 0) = P(g = 1 \mid f) \), because if $D = 0$ $g$ must get the value “1”, while \( \lambda_G(f, d = 1) = P(g = 0 \mid f) + P(g = 1 \mid f) \). In summary, we have the following.
A Hybrid Belief Network

Belief network \( P(g,f,d,c,b,a) \)
\[ = P(g|f,d)P(f|c,b)P(d|b,a)P(b|a)P(c|a)P(a) \]
Variable elimination for a mixed network:

Bucket G: $P(G|F,D) \rightarrow \neg G$

Bucket F: $P(F|B,C) \rightarrow P(G = 0| F, D)$

Bucket D: $P(D|A,B) \rightarrow \lambda^F(B,C,D)$

Bucket C: $P(C|A) \rightarrow \lambda^D(A,B,C)$

Bucket B: $P(B|A) \rightarrow \lambda^B(A,B)$

Bucket A: $P(A) \rightarrow \lambda^C(A)$

$P(A|\neg G)$

(a) regular Elim-CPE

Bucket G: $P(G|F,D)$

Bucket G: $(\neg D \lor G)(\neg F \lor G)(F \lor D \lor \neg G), \neg G$

Bucket F: $P(F|B,C)$

Bucket F: $P(G = 0| F, D)$

Bucket F: $\neg F$

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

Bucket D: $\lambda^F(D)$

Bucket D: $\neg D$

Bucket D: $(A \lor C)$

Bucket C: $P(C|A)$

Bucket C: $\lambda^F(B,C)$

Bucket B: $P(B|A)$

Bucket B: $\lambda^D(A,B), \lambda^C(A,B)$

Bucket A: $P(A)$

Bucket A: $\lambda^B(A)$

Bucket A: $\lambda^D$

$P(A|\neg G)$

(b) Elim-CPE-D with clause extraction
Trace of Elim-CPE

Belief network \( P(g,f,d,c,b,a) = P(g\mid f,d)P(f\mid c,b)P(d\mid b,a)P(b\mid a)P(c\mid a)P(a) \)
Bucket-elimination example for a mixed network

Figure 4.15: Execution of BE-CPE.

Figure 4.16: Execution of BE-CPE (evidence ¬G).
Definition 2.23 Markov networks. A Markov network is a graphical model $\mathcal{M} = \{X, D, H, \Pi\}$ where $H = \{\psi_1, \ldots, \psi_m\}$ is a set of potential functions where each potential $\psi_i$ is a non-negative real-valued function defined over a scope of variables $S = \{S_1, \ldots, S_m\}$. The Markov network represents a global joint distribution over the variables $X$ given by:

$$P_\mathcal{M} = \frac{1}{Z} \prod_{i=1}^{m} \psi_i, \quad Z = \sum_{X} \prod_{i=1}^{m} \psi_i$$

where the normalizing constant $Z$ is called the partition function.
Theorem 4.21  Complexity of BE-cpe.  Given a mixed network $M_{B,\phi}$ having mixed graph is $G$, with $w^*(d)$ its induced width along ordering $d$, $k$ the maximum domain size and $r$ be the number of input functions. The time complexity of BE-cpe is $O(r \cdot k^{w^*(d)+1})$ and its space complexity is $O(n \cdot k^{w^*(d)})$. (Prove as an exercise.)
**DEFINITION:** An undirected graph $G = (V, E)$ is said to be *chordal* if every cycle of length four or more has a chord, i.e., an edge joining two nonconsecutive vertices.

**THEOREM 7:** Let $G$ be an undirected graph $G = (V, E)$. The following four conditions are equivalent:

1. $G$ is chordal.

2. The edges of $G$ can be directed acyclically so that every pair of converging arrows emanates from two adjacent vertices.

3. All vertices of $G$ can be deleted by arranging them in separate piles, one for each clique, and then repeatedly applying the following two operations:
   - Delete a vertex that occurs in only one pile.
   - Delete a pile if all its vertices appear in another pile.

4. There is a tree $T$ (called a *join tree*) with the cliques of $G$ as vertices, such that for every vertex $v$ of $G$, if we remove from $T$ all cliques not containing $v$, the remaining subtree stays connected. In other words, any two cliques containing $v$ are either adjacent in $T$ or connected by a path made entirely of cliques that contain $v$. 

The running intersection property