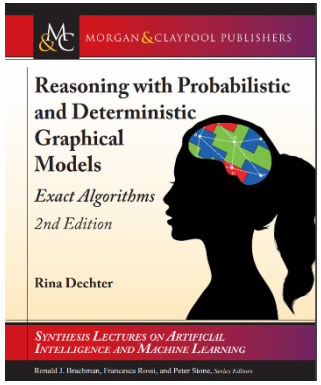
# Slides Set 4:

# Exact Inference Algorithms
# Bucket-elimination

*Rina Dechter*

(Dechter chapter 4, Darwiche chapter 6)



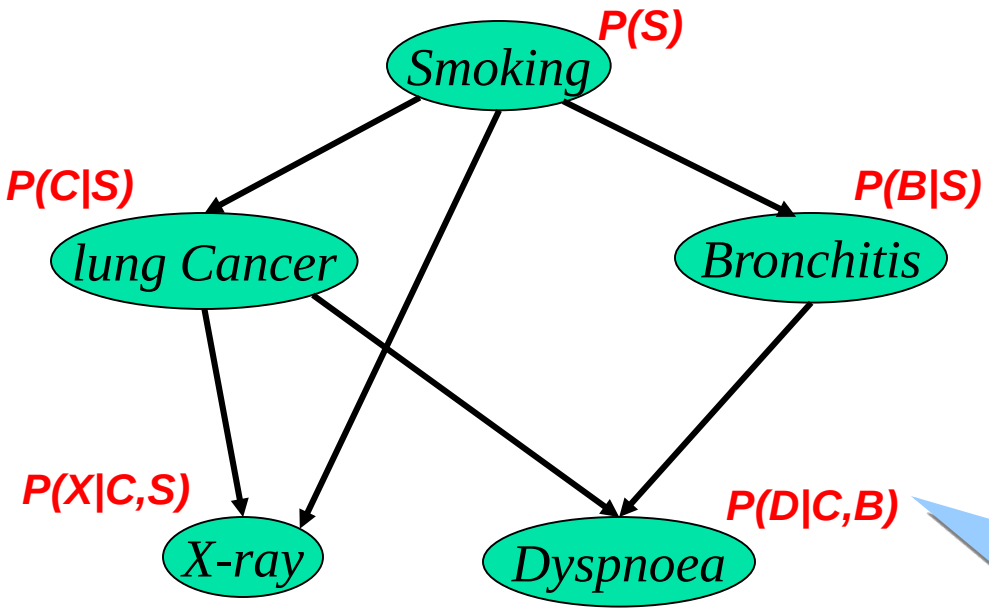
# Inference for probabilistic networks

- 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 (→MAP)
  - for MAP (→ Marginal Map)
- Induced-Width

# Bayesian Networks: Example
## (Pearl, 1988)




$$\mathbf{BN} = (\mathbf{G}, \Theta)$$

CPD:

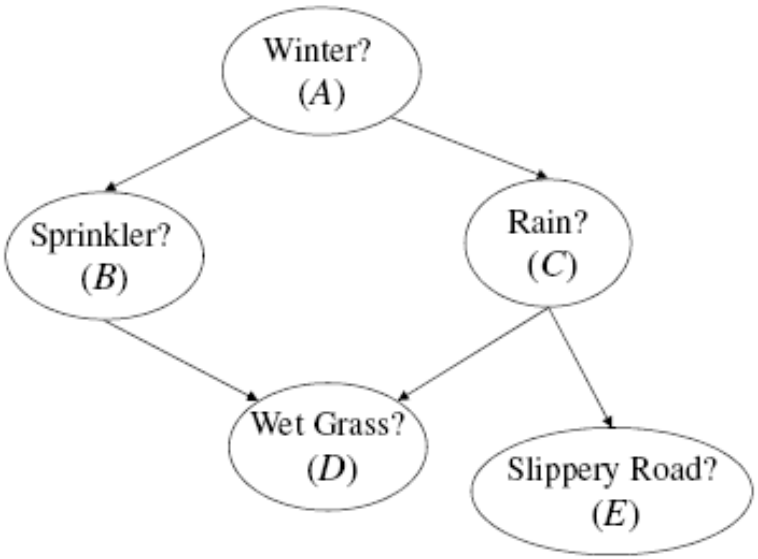
| C | B | P(D\|C,B) | |
|---|---|---|---|
| 0 | 0 | 0.1 | 0.9 |
| 0 | 1 | 0.7 | 0.3 |
| 1 | 0 | 0.8 | 0.2 |
| 1 | 1 | 0.9 | 0.1 |

$$P(S, C, B, X, D) = P(S)\, P(C|S)\, P(B|S)\, P(X|C,S)\, P(D|C,B)$$

***Belief Updating:***

*P (lung cancer=yes | smoking=no, dyspnoea=yes ) = ?*

# A Bayesian Network




| $A$ | $\Theta_A$ |
|---|---|
| true | .6 |
| false | .4 |

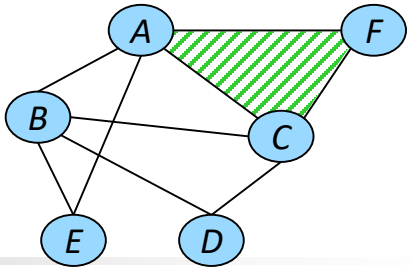
| $A$ | $B$ | $\Theta_{B\|A}$ |
|---|---|---|
| true | true | .2 |
| true | false | .8 |
| false | true | .75 |
| false | false | .25 |

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

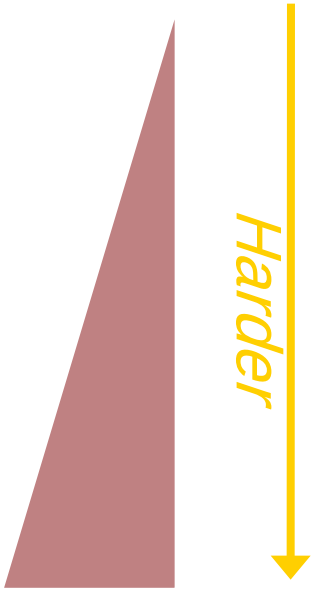
| $B$ | $C$ | $D$ | $\Theta_{D\|BC}$ |
|---|---|---|---|
| true | true | true | .95 |
| true | true | false | .05 |
| true | false | true | .9 |
| true | false | false | .1 |
| false | true | true | .8 |
| false | true | false | .2 |
| false | false | true | 0 |
| false | false | false | 1 |

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

# 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 exact and then on **approximation** algorithms
  - **Anytime**: very fast & very approximate ! Slower & more accurate

# Belief Updating is NP-hard

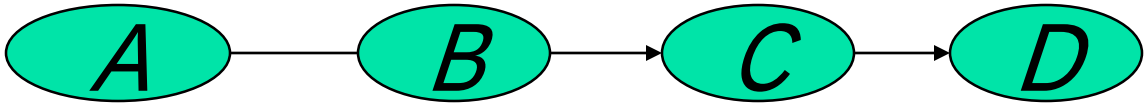
- Each SAT formula can be mapped into a belief updating query in a Bayesian network

- Example

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

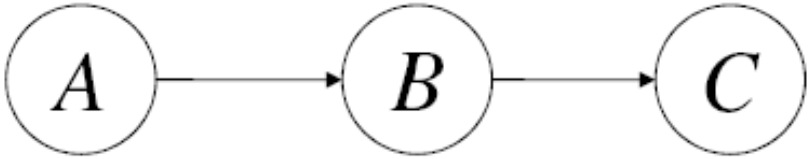
# A Simple Network

*Given:*




- 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




| $A$ | $\Theta_A$ |
|---|---|
| true | .6 |
| false | .4 |

| $A$ | $B$ | $\Theta_{B\|A}$ |
|---|---|---|
| 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 |

**To compute the prior marginal on variable $C$, $\Pr(C)$**

we first eliminate variable $A$ and then variable $B$

# 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 | $.38 = .06 + .32$ |

# Elimination as a Basis for Inference

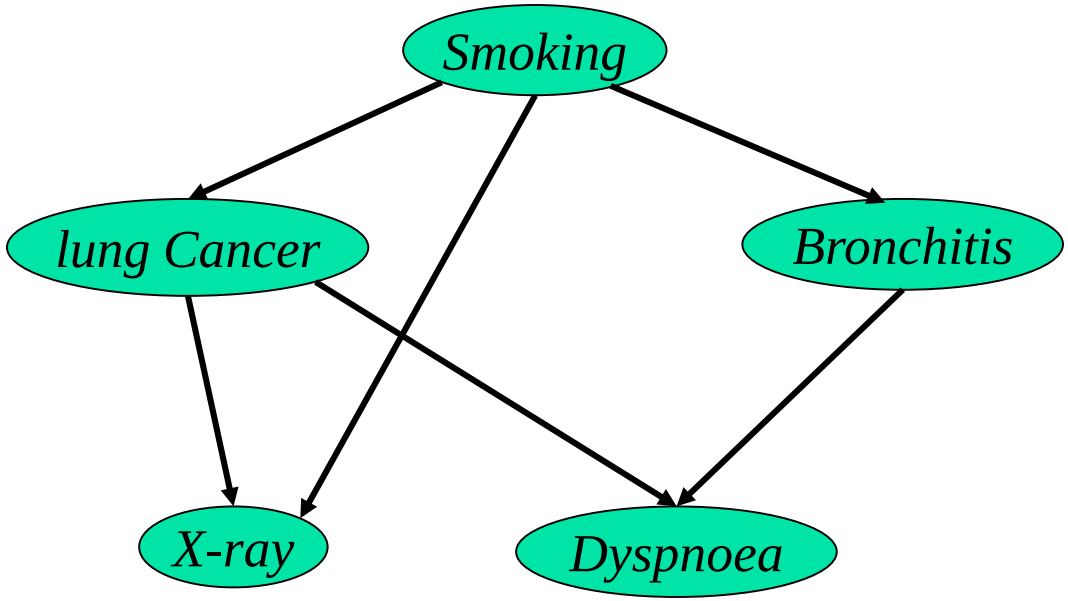
- We now have two factors, $\sum_A \Theta_A \Theta_{B|A}$ and $\Theta_{C|B}$, and we want to eliminate variable $B$
- Since $B$ appears in both factors, we must multiply them first and then sum out $B$ from the result.
- Multiplying:

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

- Summing out:

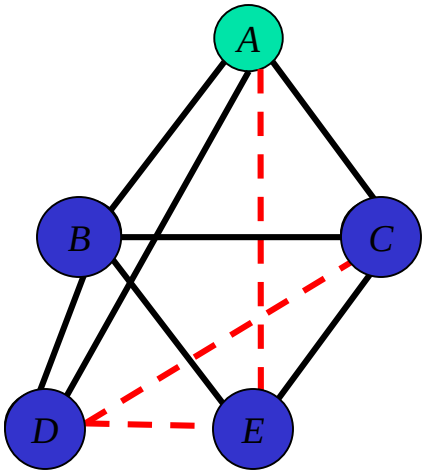
| $C$ | $\sum_B \Theta_{C\|B} \sum_A \Theta_A \Theta_{B\|A}$ |
|---|---|
| true | .376 |
| false | .624 |

# Belief Updating




*P (lung cancer=yes | smoking=no, dyspnoea=yes ) = ?*

# Belief updating: P(X|evidence)=?




*"Moral" graph*

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

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

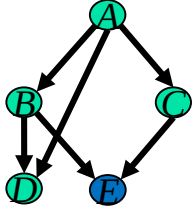
$$P(a)\sum_{e=0}\sum_{d}\sum_{c}P(c|a\sum_{b}P(b|a)P(d|b,a)P(e|b,c)$$

*Variable Elimination*

$$h^B(a,d,c,e)$$

# Bucket elimination

## Algorithm *BE-bel* (Dechter 1996)




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

$\sum_b \prod$ ← *Elimination operator*

*bucket B:* $P(b|a)$ $P(d|b,a)$ $P(e|b,c)$

*bucket C:* $P(c|a)$ $\lambda^B(a,d,c,e)$

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

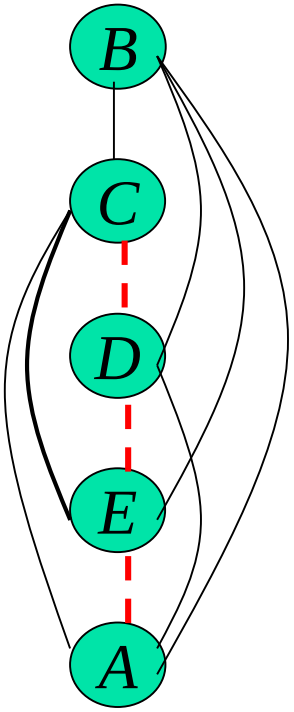
*bucket E:* $e=0$ $\lambda^D(a,e)$

*bucket A:* $P(a)$ $\lambda^E(a)$

$P(e=0)$

$P(a,e=0)$

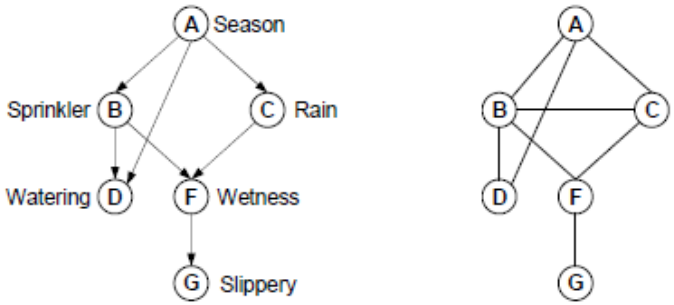
*W*=4*
*"induced width"*
*(max clique size)*




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

# A Bayesian Network

## Ordering: A,C,B,E,D,G




(a) Directed acyclic graph (b) Moral graph

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

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

$$P(a, g=1) = P(a)\sum_c P(c|a)\sum_b P(b|a)\sum_f P(f|b,c)\lambda_G(f)\sum_d P(d|b,a). \quad (4.2)$$

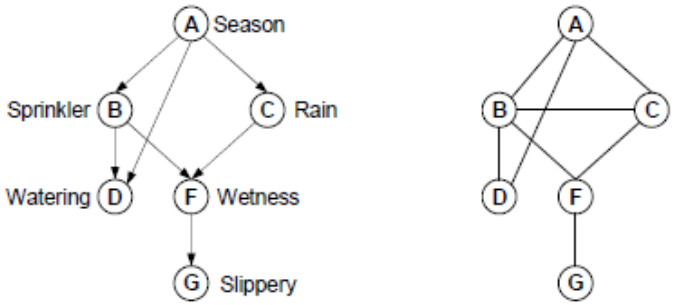
$$P(a, g=1) = P(a)\sum_c P(c|a)\sum_b P(b|a)\lambda_D(a,b)\sum_f P(f|b,c)\lambda_G(f) \quad (4.3)$$

$$P(a, g=1) = P(a)\sum_c P(c|a)\sum_b P(b|a)\lambda_D(a,b)\lambda_F(b,c) \quad (4.4)$$

$$P(a, g=1) = P(a)\sum_c P(c|a)\lambda_B(a,c) \quad (4.5)$$

# A Bayesian Network

## Ordering: A,C,B,E,D,G




(a) Directed acyclic graph (b) Moral graph

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

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

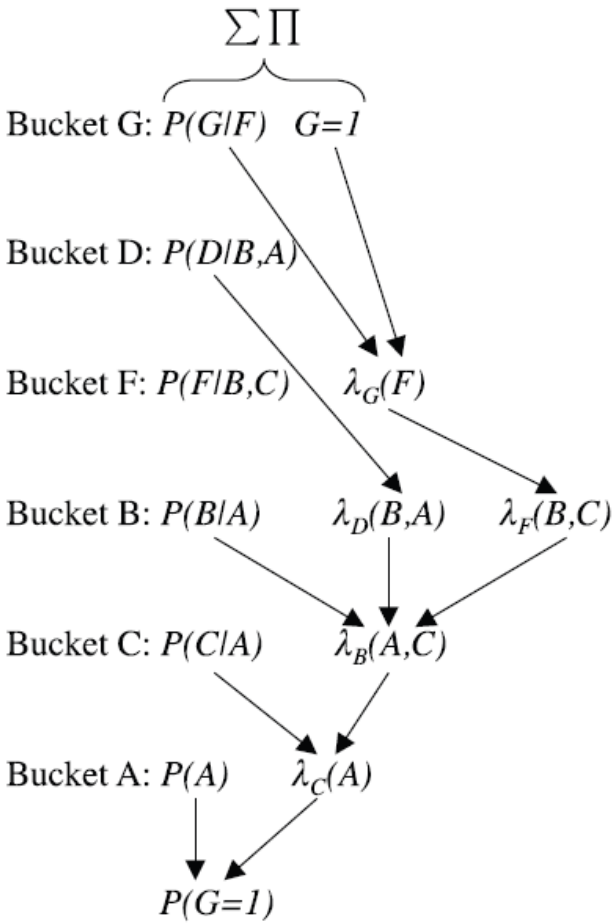
$$P(a, g=1) = P(a)\sum_c P(c|a)\sum_b P(b|a)\sum_f P(f|b,c)\lambda_G(f)\sum_d P(d|b,a). \quad (4.2)$$

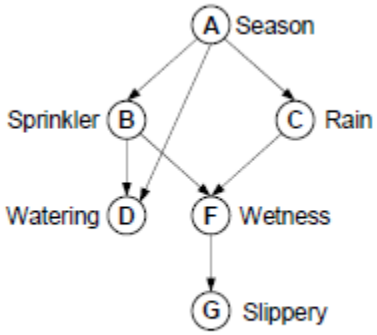
$$P(a, g=1) = P(a)\sum_c P(c|a)\sum_b P(b|a)\lambda_D(a,b)\sum_f P(f|b,c)\lambda_G(f) \quad (4.3)$$

$$P(a, g=1) = P(a)\sum_c P(c|a)\sum_b P(b|a)\lambda_D(a,b)\lambda_F(b,c) \quad (4.4)$$

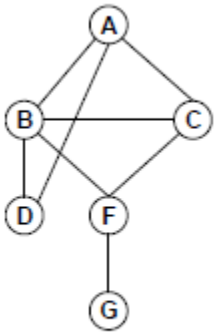
$$P(a, g=1) = P(a)\sum_c P(c|a)\lambda_B(a,c) \quad (4.5)$$

# A Bayesian Network
## Ordering: A,C,B,F,D,G



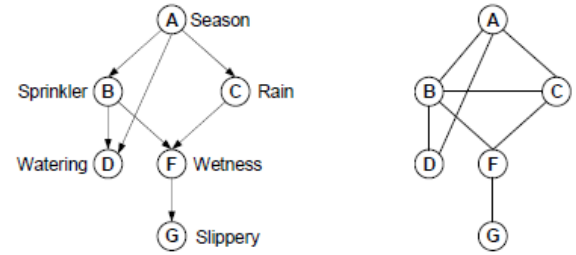




(a) Directed acyclic graph




(b) Moral graph

# A Different Ordering




(a) Directed acyclic graph (b) Moral graph

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

$$
\begin{aligned}
P(a, g=1) &= P(a) \textstyle\sum_f \sum_d \sum_c P(c|a) \sum_b P(b|a)\ P(d|a,b) P(f|b,c) \sum_{g=1} P(g|f) \\
&= P(a) \textstyle\sum_f \lambda_G(f) \sum_d \sum_c P(c|a) \sum_b P(b|a)\ P(d|a,b) P(f|b,c) \\
&= P(a) \textstyle\sum_f \lambda_G(f) \sum_d \sum_c P(c|a) \lambda_B(a,d,c,f) \\
&= P(a) \textstyle\sum_f \lambda_g(f) \sum_d \lambda_C(a,d,f) \\
&= P(a) \textstyle\sum_f \lambda_G(f) \lambda_D(a,f) \\
&= P(a) \lambda_F(a)
\end{aligned}
$$

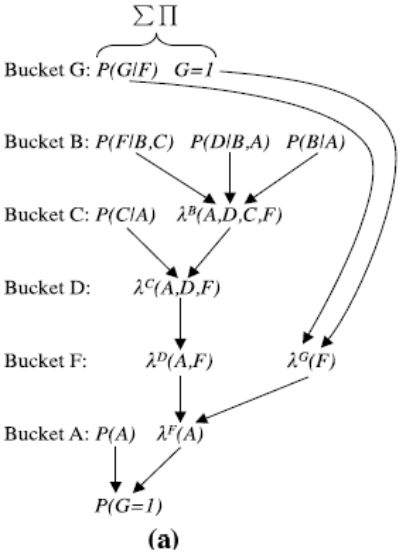
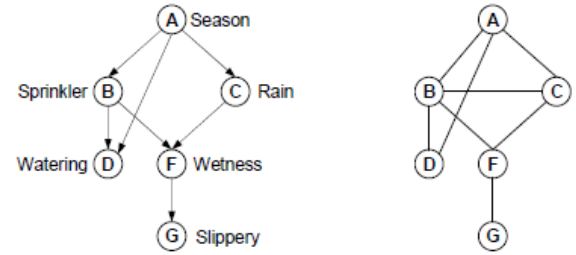



Figure 4.3: The bucket's output when processing along $d_2 = A, F, D, C, B, G$

# A Different Ordering




(a) Directed acyclic graph (b) Moral graph

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

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

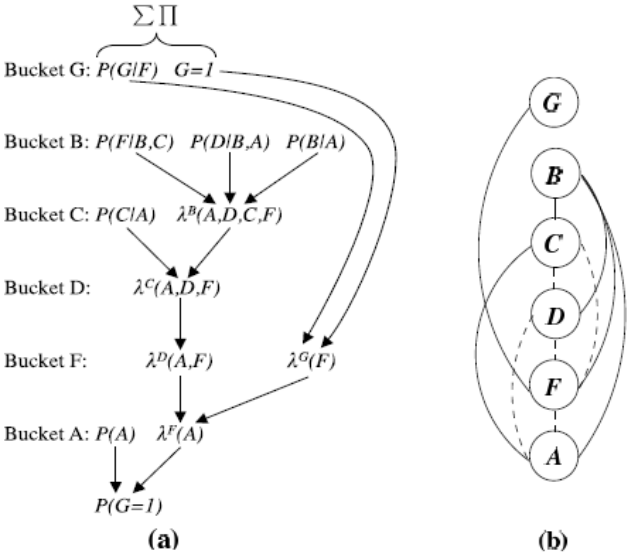
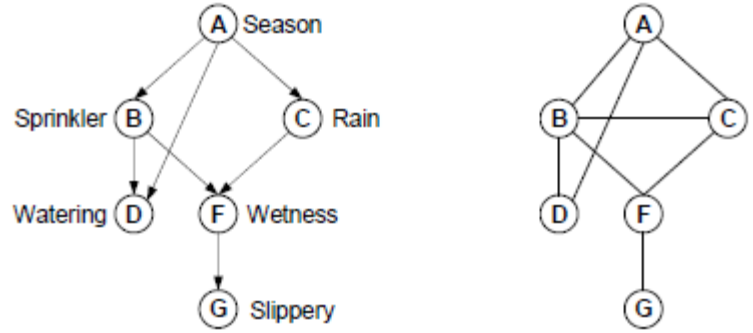
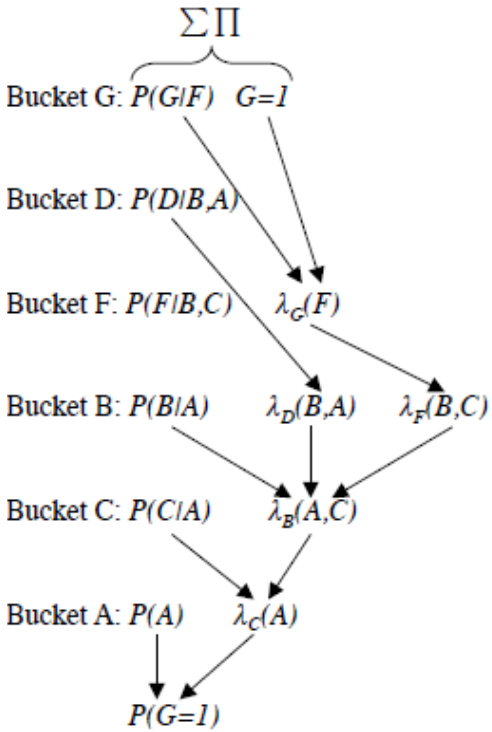


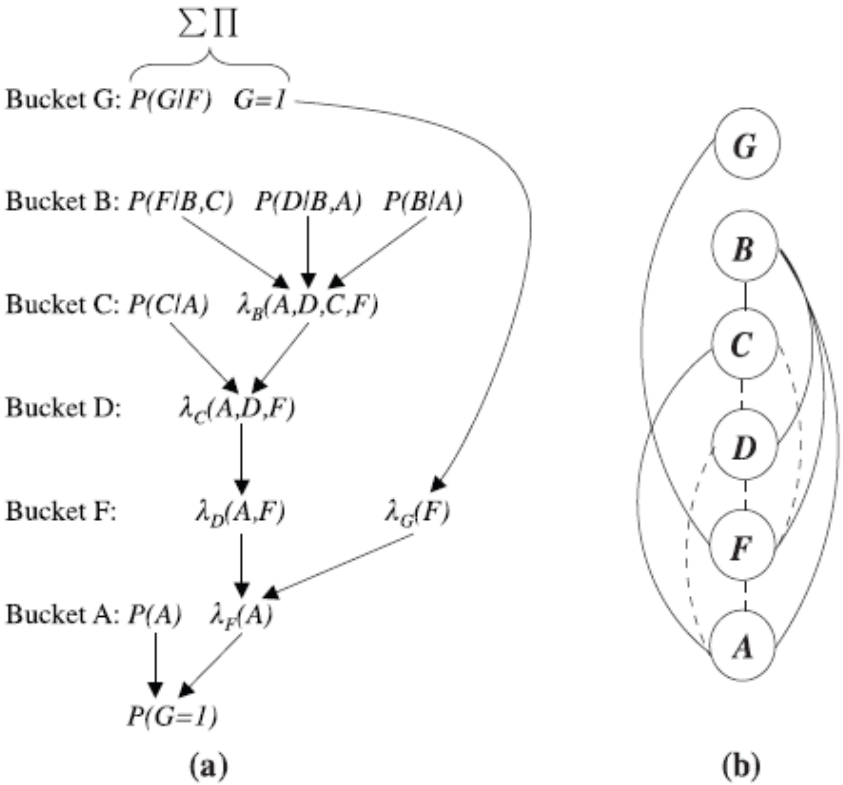

Figure 4.3: The bucket's output when processing along $d_2 = A, F, D, C, B, G$

# A Bayesian Network Processed Along 2 Orderings




(a) Directed acyclic graph (b) Moral graph




*d1=A,C,B,F,D,G*

**Figure 4.4:** The bucket's output when processing along $d_2 = A, F, D, C, B, G$.

# 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 x 0.8*

# Factors: Sum-Out Operation

**The result of summing out variable $X$ from factor $f(\mathbf{X})$**

is another factor over variables $\mathbf{Y} = \mathbf{X} \setminus \{X\}$:

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

| $B$ | $C$ | $D$ | $f_1$ |
|---|---|---|---|
| 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 |

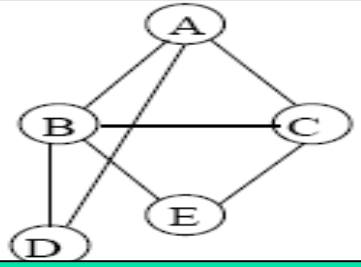
| $B$ | $C$ | $\sum_D f_1$ |
|---|---|---|
| true | true | 1 |
| true | false | 1 |
| false | true | 1 |
| false | false | 1 |

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



*Thanks to Darwiche*

# Bucket Elimination and Induced Width




**Ordering: a, e, d, c, b**

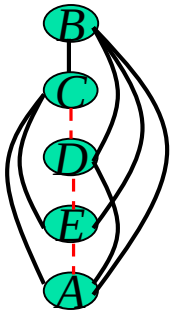
$bucket(B) = \quad P(e|b,c), P(d|a,b), P(b|a)$

$bucket(C) = \quad P(c|a) \quad || \quad \lambda_B(a,c,d,e)$

$bucket(D) = \quad || \ \lambda_C(a,d,e)$

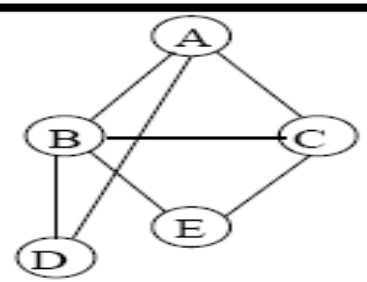
$bucket(E) = \quad e = 0 \quad || \ \lambda_D(a,c)$

$bucket(A) = \quad P(a) \quad || \quad \lambda_E(a)$




$W^*=4$

# Bucket Elimination and Induced Width




**Ordering: a, b, c, d, e**

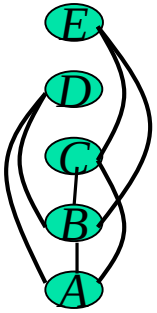
$bucket(E) = \quad P(e|b,c), \quad e = 0$

$bucket(D) = \quad P(d|a,b)$

$bucket(C) = \quad P(c|a) \quad || \quad P(e = 0|b,c)$

$bucket(B) = \quad P(b|a) \quad || \quad \lambda_D(a,b), \lambda_C(b,c)$

$bucket(A) = \quad P(a) \quad || \quad \lambda_B(a)$




W*=2

**Ordering: a, e, d, c, b**

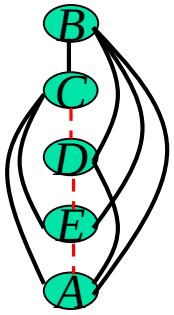
$bucket(B) = \quad P(e|b,c), P(d|a,b), P(b|a)$

$bucket(C) = \quad P(c|a) \quad || \quad \lambda_B(a,c,d,e)$

$bucket(D) = \quad || \ \lambda_C(a,d,e)$

$bucket(E) = \quad e = 0 \quad || \ \lambda_D(a,c)$

$bucket(A) = \quad P(a) \quad || \quad \lambda_E(a)$




W*=4

ALGORITHM BE-BEL

**Input:** A belief network $\mathcal{B} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}_G, \prod \rangle$, an ordering $d = (X_1, \ldots, X_n)$ ; evidence $e$
**output:** The belief $P(X_1|\mathbf{e})$ and probability of evidence $P(\mathbf{e})$

1. Partition the input functions (CPTs) into $bucket_1$, ..., $bucket_n$ 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 $\psi_i$ the product of input functions in $bucket_i$.
2. **backward: for** $p \leftarrow n$ **downto** 1 **do**
3. **for** all the functions $\psi_{S_0}, \lambda_{S_1}, \ldots, \lambda_{S_j}$ 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 $\lambda_p$ in bucket of the latest variable in scope($\lambda_p$),
6. **return** (as a result of processing $bucket_1$):

   $P(\mathbf{e}) = \alpha = \sum_{X_1} \psi_1 \cdot \prod_{\lambda \in bucket_1} \lambda$

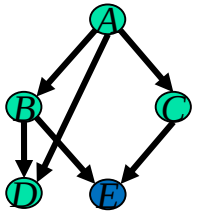
   $P(X_1|\mathbf{e}) = \frac{1}{\alpha}\psi_1 \cdot \prod_{\lambda \in bucket_1} \lambda$

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

*Last slide.*

# Belief Updating

*Algorithm BE-bel* *[Dechter 1996]*




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

$\sum_b \prod$ ⟵ *Elimination & combination operators*

*bucket B:* $p(b|A)\ p(d|b,A)\ p(e|b,c)$

*bucket C:* $p(c|A)$ $\lambda_{B\to C}(A,d,c,e)$

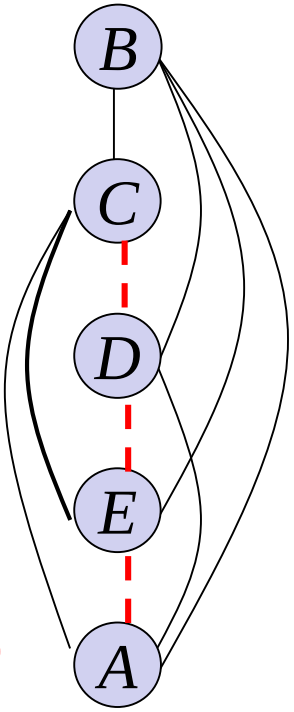
*bucket D:* $\lambda_{C\to D}(A,d,e)$

*bucket E:* $\mathbb{1}[E=0]\ \lambda_{D\to E}(A,e)$

*bucket A:* $p(A)$ $\lambda_{E\to A}(A)$

$p(E=0)$

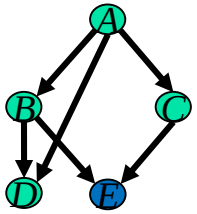
*W*=4*
*"induced width"*
*(max clique size)*




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

# Bucket Elimination

*Algorithm BE-bel* [Dechter 1996]



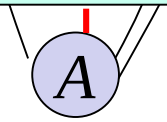

$$p(A|E=0) = \alpha \sum_{e,d,c,b} p(A)\, p(b|A)\, p(c|A)\, p(d|A,b)\, p(e|b,c)\, \mathbb{1}[e=0]$$

$\sum_b \prod$ ⟵ *Elimination & combination operators*

***Time and space exponential in the induced-width / treewidth***

*bucket A:* $p(A)$ $\lambda_{E\to A}(A)$

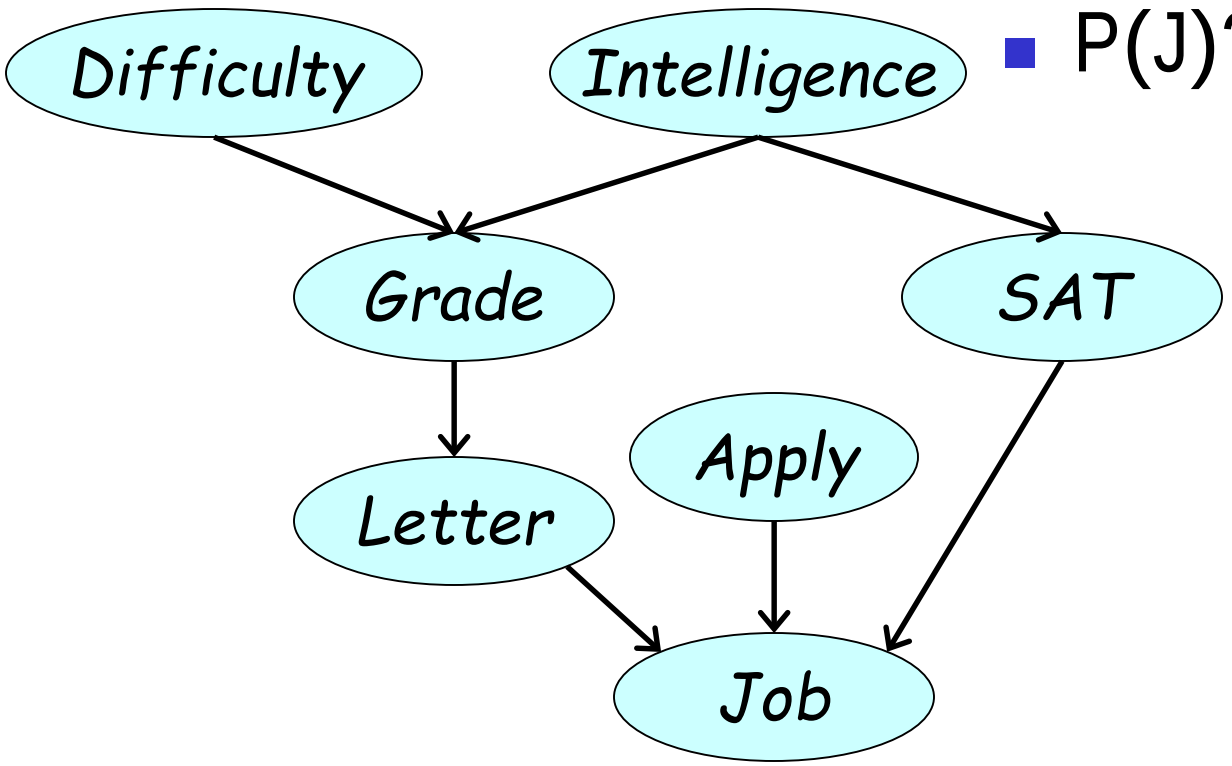
*induced width (max clique size)*




$p(E=0)$

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

# Student Network Example

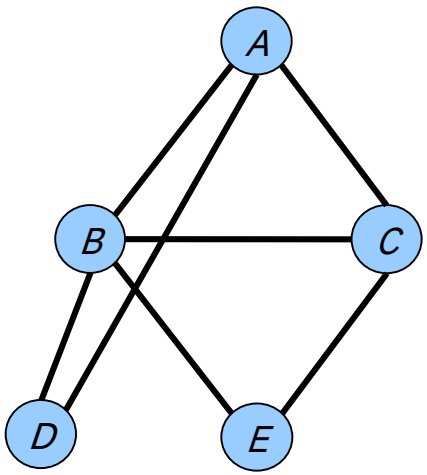



- P(J)?

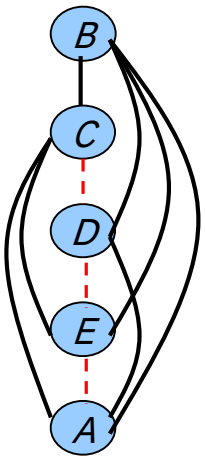
# Induced Width (continued)

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

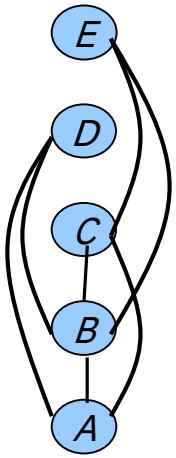
*The effect of the ordering:*




*Primal (moraal) graph*




$w^*(d_1) = 4$



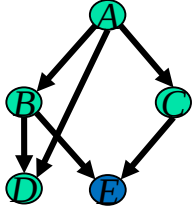

$w^*(d_2) = 2$

# 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)
- Induced-Width

# The impact of evidence?

## Algorithm *BE-bel*




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

$\sum_b \prod$ ← *Elimination operator*

*bucket B:* $P(b|a)$ $P(d|b,a)$ $P(e|b,c)$

*bucket C:* $P(c|a)$ $\lambda^B(a,d,c,e)$

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

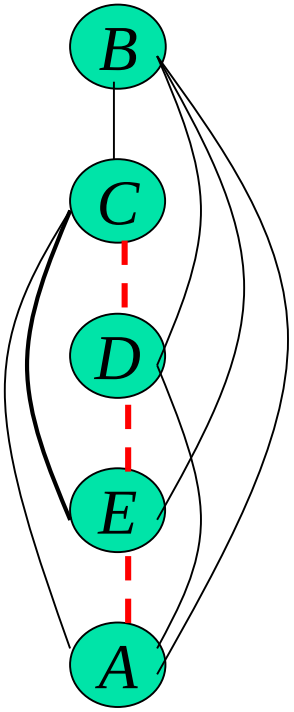
*bucket E:* $e=0$ $\lambda^D(a,e)$

*bucket A:* $P(a)$ $\lambda^E(a)$

$P(e=0)$

$P(a|e=0)$

*B=1*

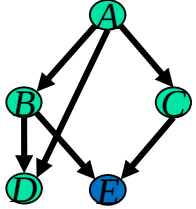



*W*=4*

*"induced width"*
*(max clique size)*

# 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|E=0,B=1)?*

$\sum_b \prod$ ← *Elimination operator*

*bucket B:* $P(b|a)$ $P(d|b,a)$ $P(e|b,c)$

*bucket C:* $P(c|a)$ *P(e|b=1,c)*

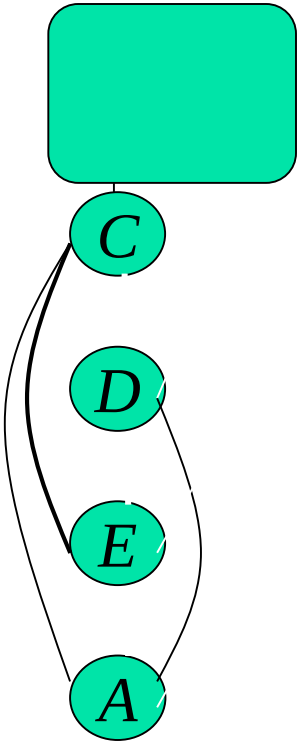
*bucket D:* *P(d|b=1,a)*

*bucket E:* $e=0$

*bucket A:* $P(a)$ *P(b=1|a)*

***P(e=0)***

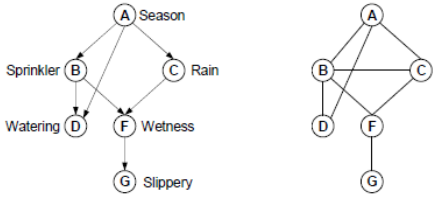
*B=1*




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

***P(a|e=0)***

# The impact of observations




(a) Directed acyclic graph (b) Moral graph

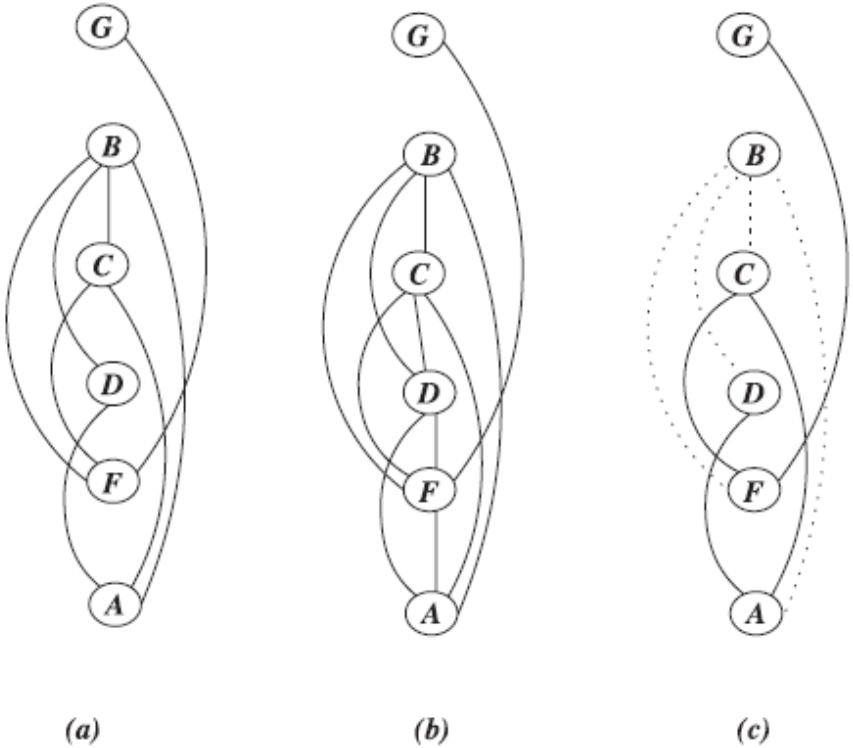
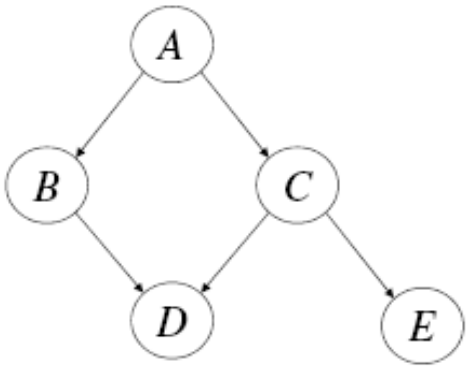



**Figure 4.9:** Adjusted induced graph relative to observing $B$.

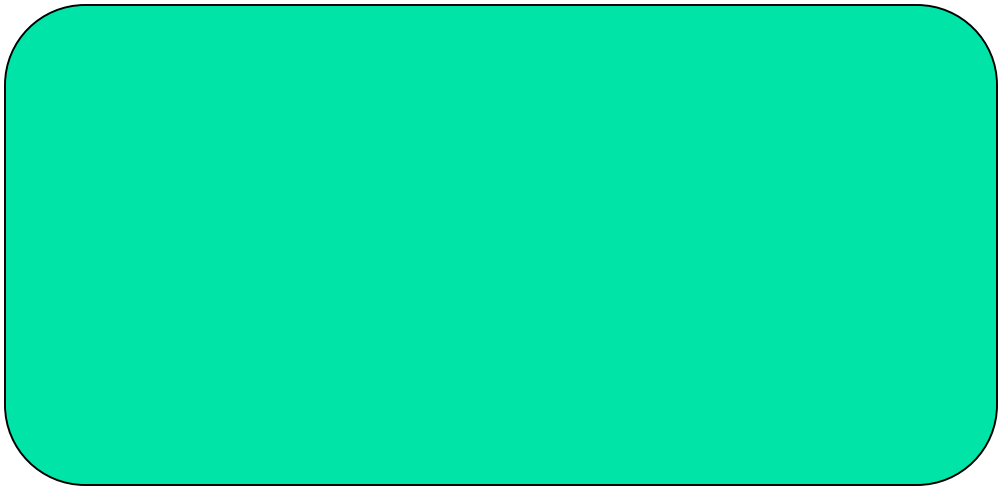
*Ordered graph* *Induced graph* *Ordered conditioned graph*

# 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 $(\mathbf{Q}, \mathbf{e})$**

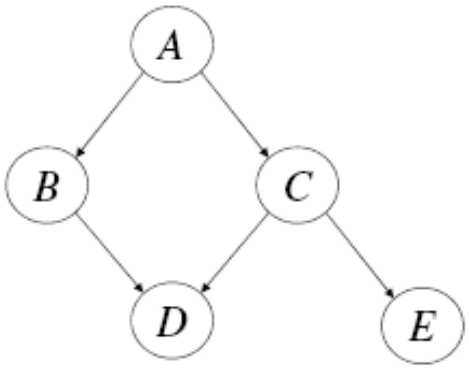
one can remove any leaf node (with its CPT) from the network as long as it does not belong to variables $\mathbf{Q} \cup \mathbf{E}$, yet not affect the ability of the network to answer the query correctly.

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

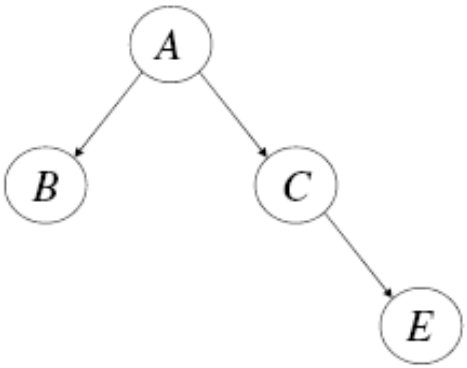
then $\Pr(\mathbf{Q}, \mathbf{e}) = \Pr'(\mathbf{Q}, \mathbf{e})$, where $\Pr$ and $\Pr'$ are the probability distributions induced by networks $\mathcal{N}$ and $\mathcal{N}'$, respectively.

# Pruning Nodes: Example

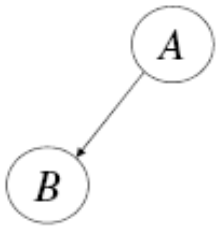
*Example of pruning irrelevant subnetworks*




network structure




joint on $B, E$



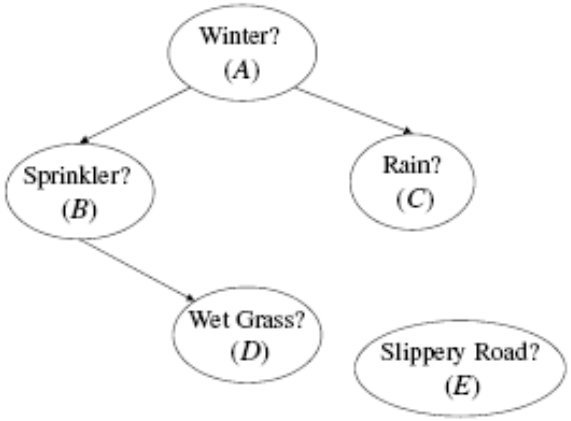

joint on $B$

# Pruning Edges: Example

*Example of pruning edges due to evidence or conditioning*

| $A$ | $B$ | $\Theta_{B\|A}$ |
|---|---|---|
| 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 |




| $A$ | $\Theta_A$ |
|---|---|
| true | .6 |
| false | .4 |

| $B$ | $D$ | $\sum_C \Theta_{D\|BC}^{C=\text{false}}$ |
|---|---|---|
| true | true | .9 |
| true | false | .1 |
| false | true | 0 |
| false | false | 1 |

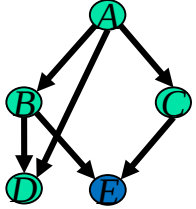
| $E$ | $\sum_C \Theta_{E\|C}^{C=\text{false}}$ |
|---|---|
| true | 0 |
| false | 1 |

Evidence $\mathbf{e}$ : $C = \text{false}$

# 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)
- Induced-Width

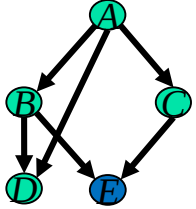
# $\text{MPE} = \max_{\overline{x}} P(\overline{x})$




$\sum$ is replaced by ***max*** *:*

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

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




$\sum$ is replaced by ***max*** :

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

$\max_b \prod$ ← *Elimination operator*

*bucket B:* $P(b|a)$ $P(d|b,a)$ $P(e|b,c)$

*bucket C:* $P(c|a)$ $h^B(a,d,c,e)$

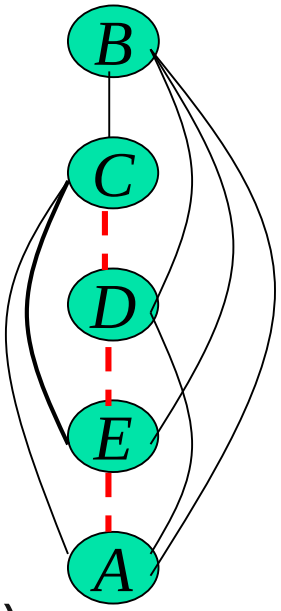
*bucket D:* $h^C(a,d,e)$

*bucket E:* $e=0$ $h^D(a,e)$

*bucket A:* $P(a)$ $h^E(a)$

***MPE***

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



# Generating the MPE-tuple

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

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

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

2. $e' = 0$

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

$B$: $P(b|a)$ $P(d|b,a)$ $P(e|b,c)$

$C$: $P(c|a)$ $h^B(a,d,c,e)$

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

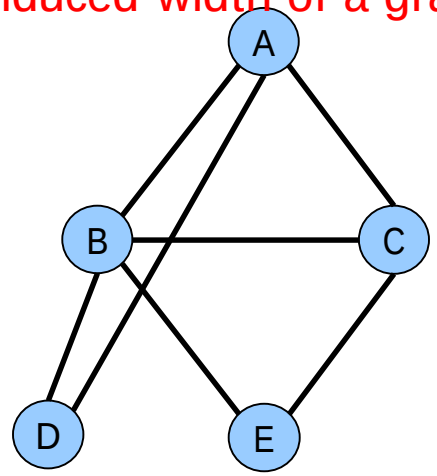
$E$: $e=0$ $h^D(a,e)$

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

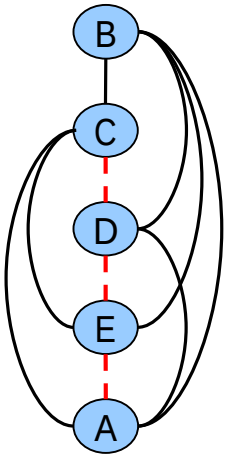
**Return** $(a', b', c', d', e')$

# Induced Width

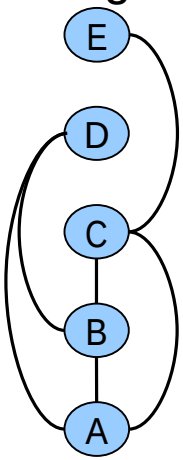
- Width is the max number of parents in the ordered graph
- Induced-width is the width of the induced ordered graph: recursively connecting parents going from last node to first.
- Induced-width w*(d) is the max induced-width over all nodes in ordering d
- Induced-width of a graph, w* is the min w*(d)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




primal graph




$w^*(d_1) = 4$




$w^*(d_2) = 2$

# Complexity of Bucket Elimination

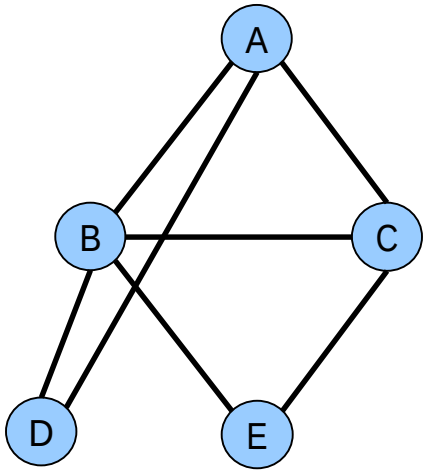
*Bucket-Elimination is **time** and **space***

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

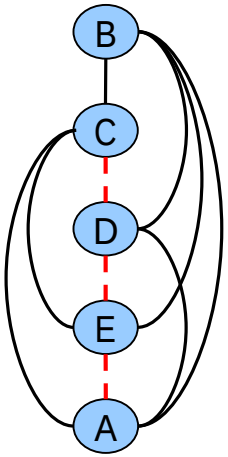
$w_d^*$ : *the induced width of the primal graph along ordering d*

*r = number of functions*

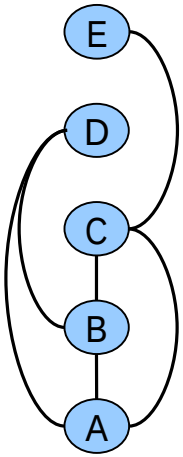
*The effect of the ordering:*




primal graph




$w^*(d_1) = 4$

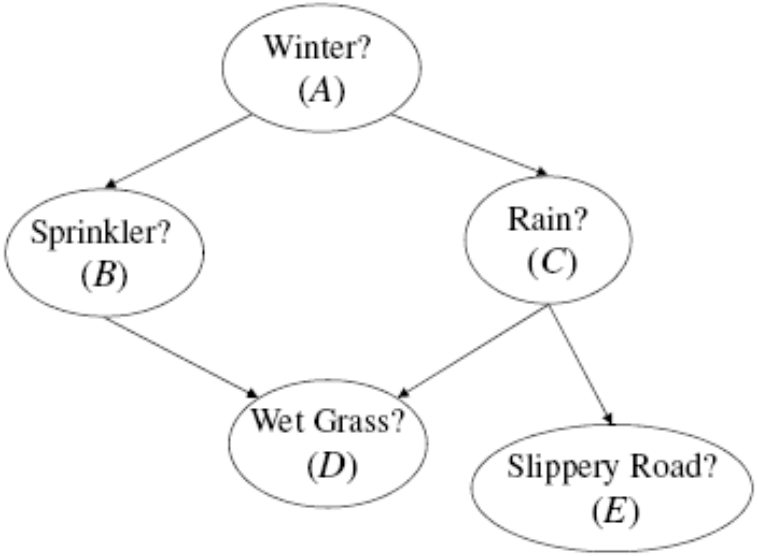



$w^*(d_2) = 2$

***Finding smallest induced-width is hard!***

# A Bayesian Network

*Example with mpe?*




| $A$ | $\Theta_A$ |
|---|---|
| true | .6 |
| false | .4 |

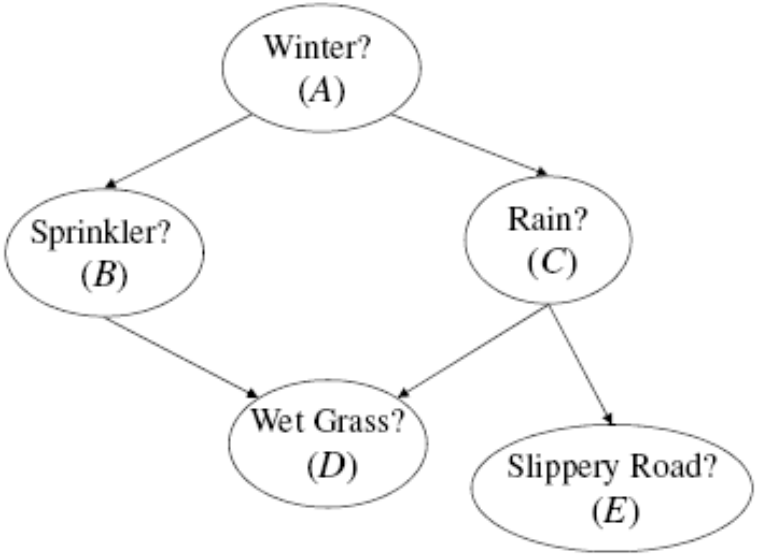
| $A$ | $B$ | $\Theta_{B\|A}$ |
|---|---|---|
| true | true | .2 |
| true | false | .8 |
| false | true | .75 |
| false | false | .25 |

| $A$ | $C$ | $\Theta_{C\|A}$ |
|---|---|---|
| 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




| $A$ | $\Theta_A$ |
|---|---|
| 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

$$C(a,b,c,d,e) = -logP = C(a) + C(b,a) + C(c,a) + C(f,b,c) + C(d,a,b) + C(g,f)$$

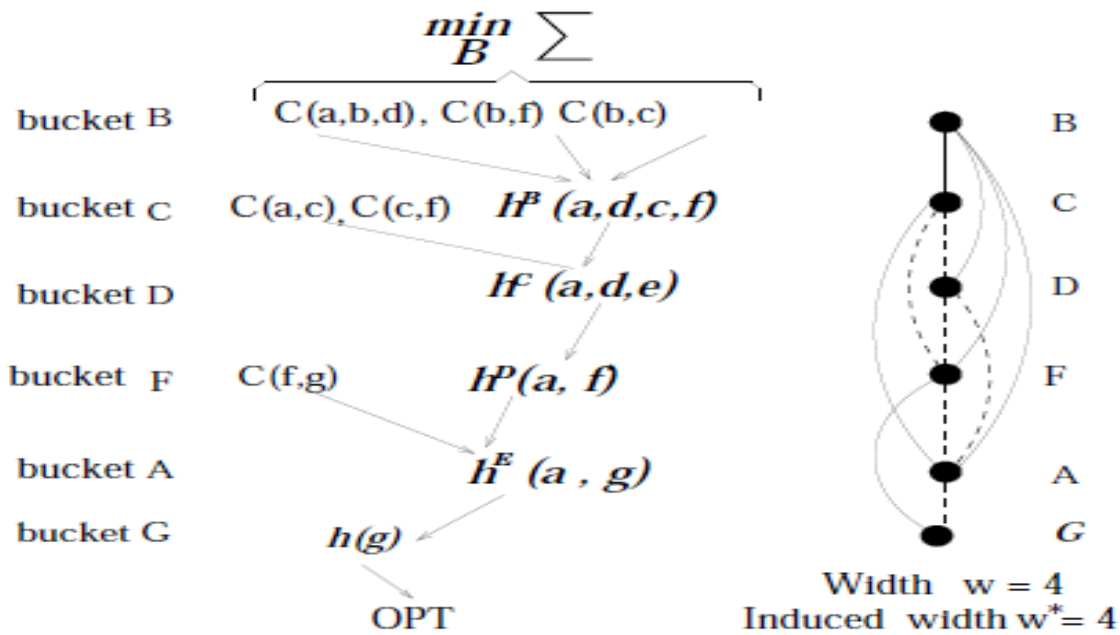



Figure 5.12: Schematic execution of BE-Opt

# 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)
- Induced-Width

# Marginal Map

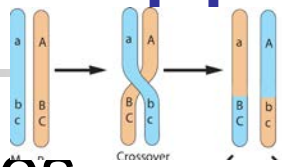
| | |
|---|---|
| Max-Inference | $f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$ |
| Sum-Inference | $Z = \sum_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$ |
| Mixed-Inference | $f(\mathbf{x}_M^*) = \max_{\mathbf{x}_\mathbf{M}} \sum_{\mathbf{x}_S} \prod_{\alpha}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

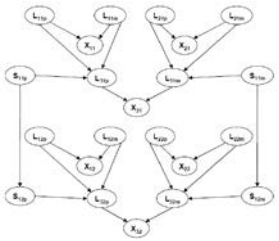
Harder

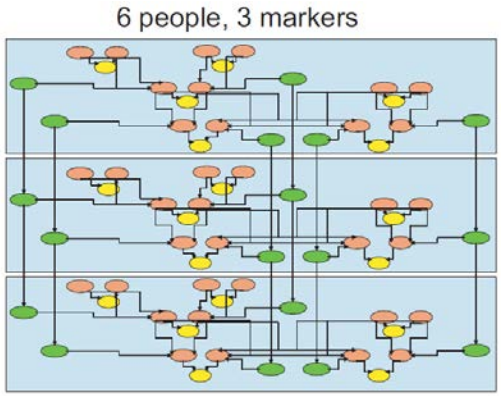
- **NP-hard**: exponentially many terms

# Example for MMAP Applications

- Haplotype in Family pedigrees






- Coding networks

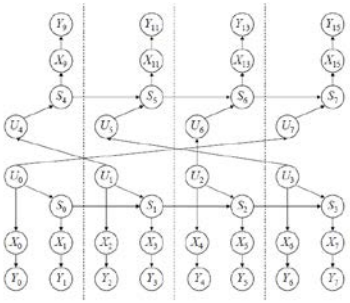
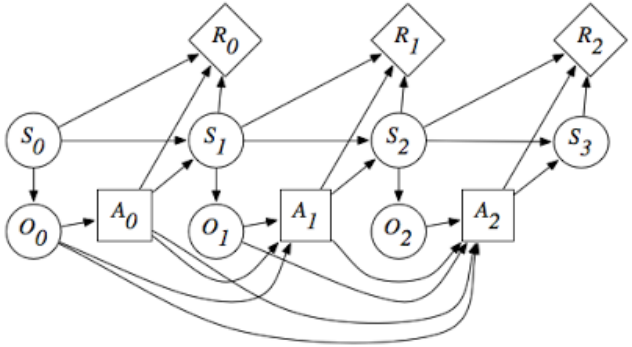
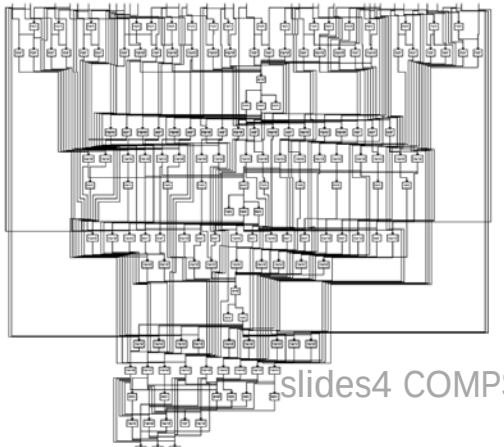


Figure 5.24: A Bayesian network for a turbo code.

- Probabilistic planning

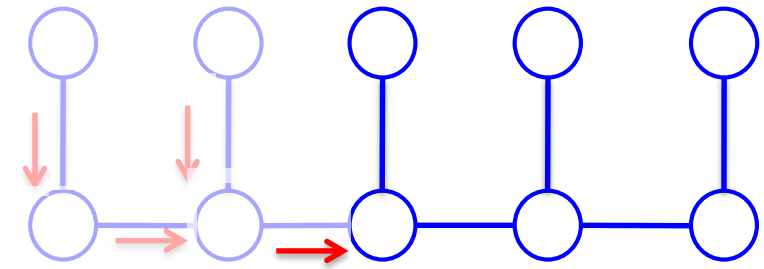



- Diagnosis



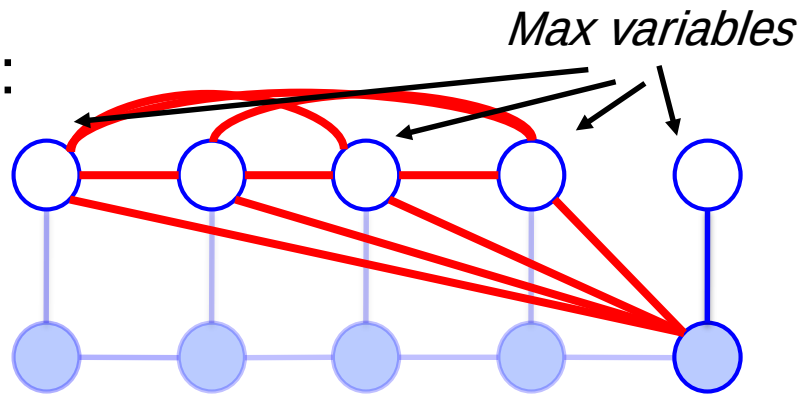
# Marginal MAP is Not Easy on Trees

- Pure MAP or summation tasks
  - Dynamic programming
  - Ex: efficient on trees



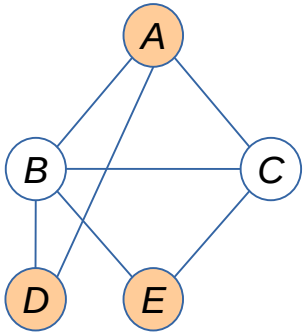
- Marginal MAP
  - Operations do not commute:
  - Sum must be done first!

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



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

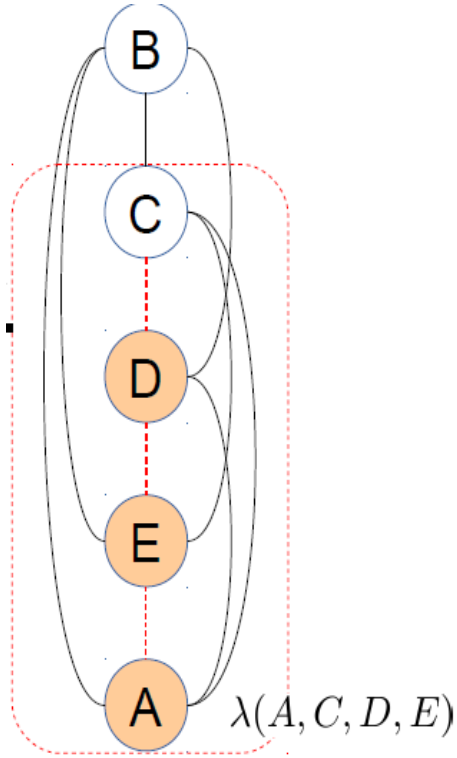
$max_D$

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

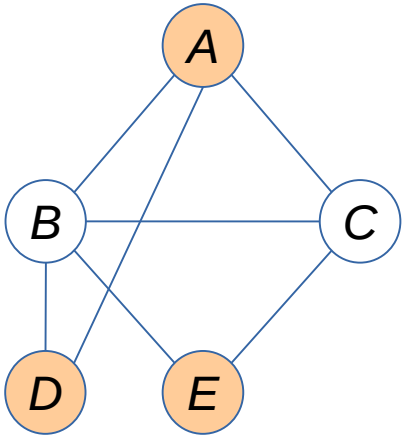
$max_E$

*A:* $\underbrace{\lambda^E(A)}$

*MAP* is the marginal MAP value*

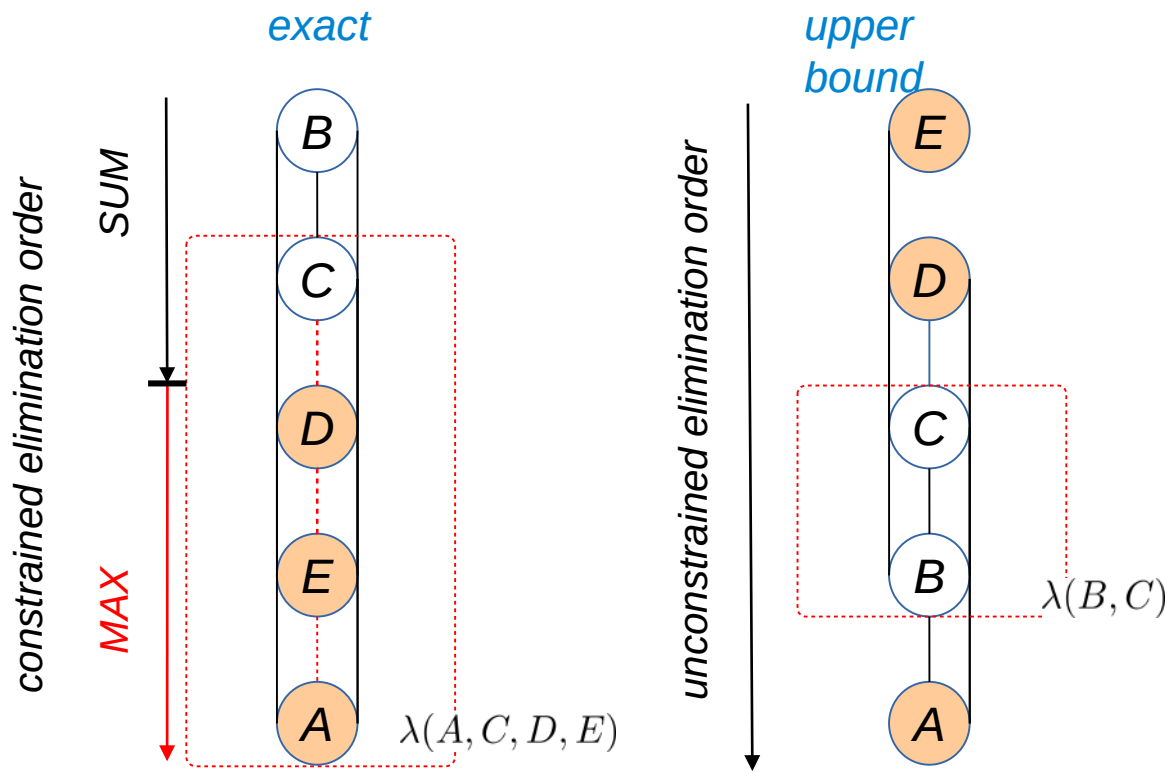


# Why is MMAP harder?




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

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




$w^* = 4$ $\qquad$ $w^* = 2$

***In practice, constrained induced is much larger!***

$$\max_X \sum_Y \phi \leq \sum_Y \max_X \phi$$

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

# 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:

$$P(x) = \frac{1}{Z} \sum_{x \in \mathcal{D}} \Pi_{C \in \mathcal{C}} \Psi_C(x_C)$$

*BE is equally applicable*

$$Z = \Sigma_x \Pi_{C \in \mathcal{C}} \Psi_C(x_C) \tag{2.2}$$

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

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

where,

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

# 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)
- 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)*

# Greedy Orderings Heuristics

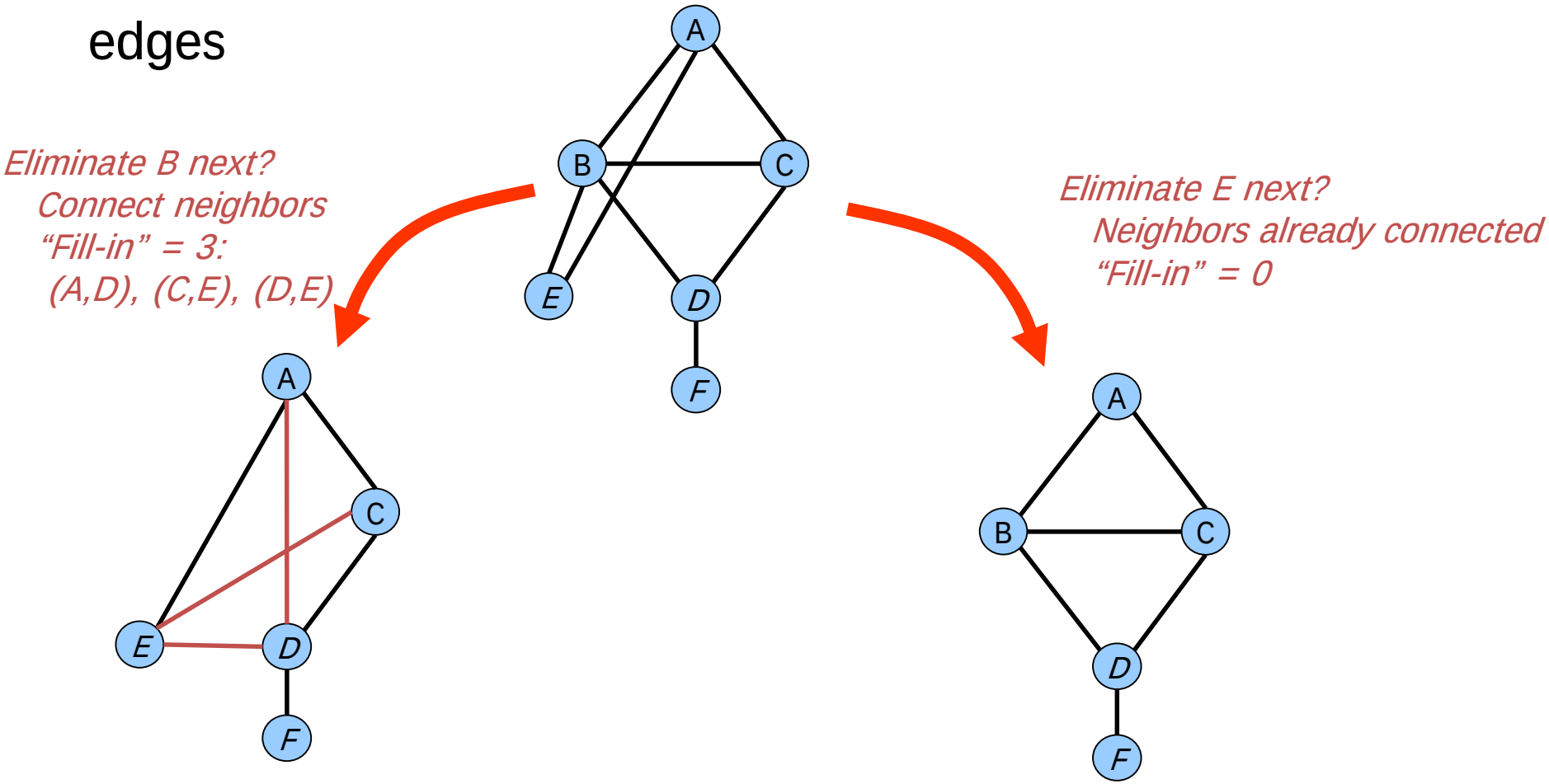
- Min-induced-width
  - From last to first, pick a node with smallest width, then connect parent and remove

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

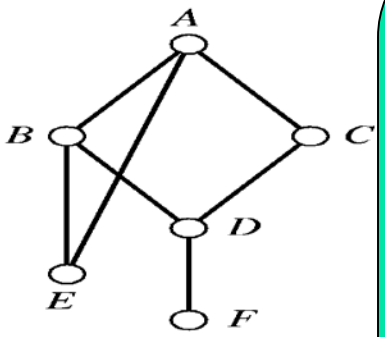
*Complexity?* $O(n^3)$

# Min-Fill Heuristic

- Select the variable that creates the fewest "fill-in" edges

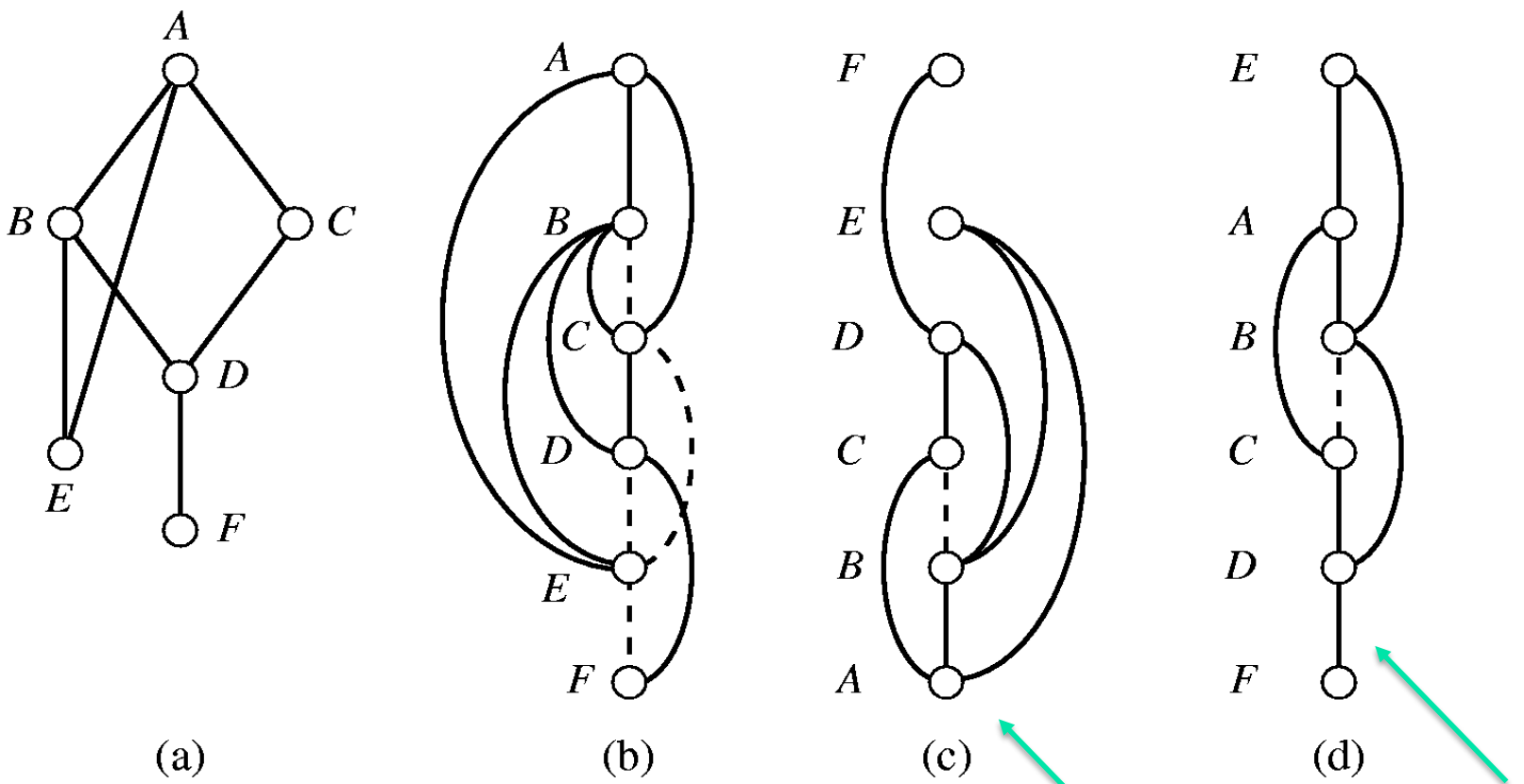


# Example




(a)

# Different Induced-Graphs

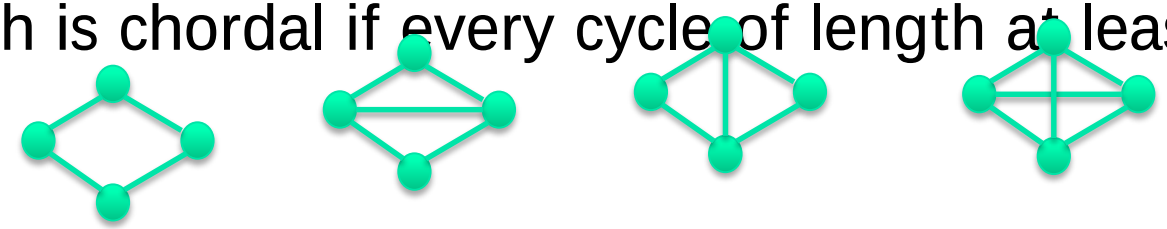



A Miw ordering

A Min-fill ordering

# Chordal Graphs

- 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 $G = (V, E)$, $V = \{v_1, ..., v_n\}$

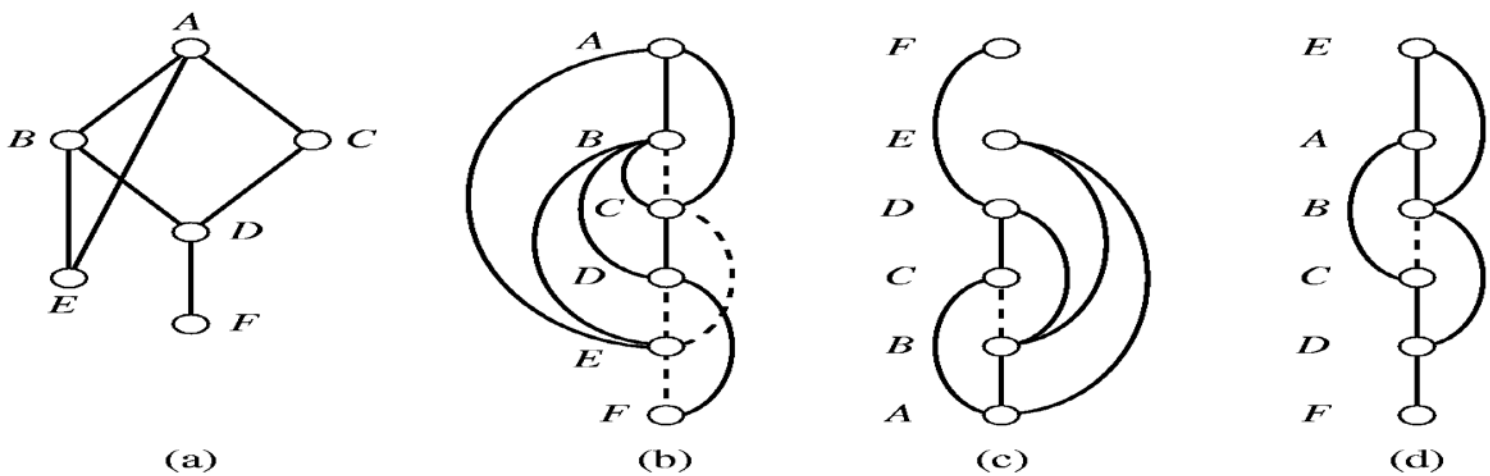
**output:** An ordering of the nodes $d = (v_1, ..., v_n)$.

1. Place an arbitrary node in position 0.
2. **for** $j = 1$ to $n$ do
3. $\quad r \leftarrow$ 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)

# K-trees

**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]