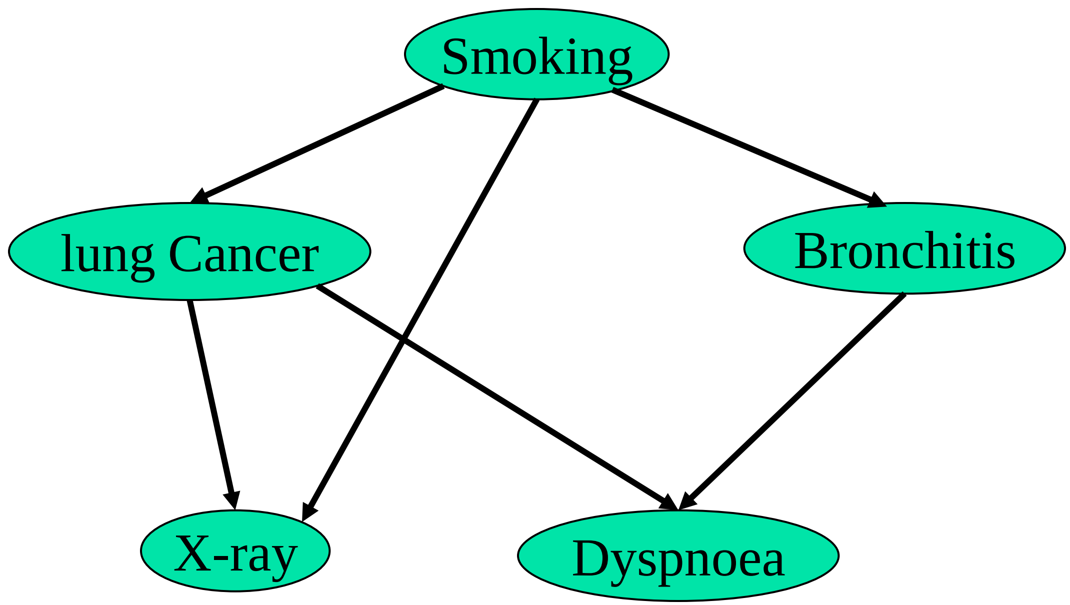
Exact Inference Algorithms Bucket-elimination

COMPSCI 276, Spring 2011

Class 5: Rina Dechter

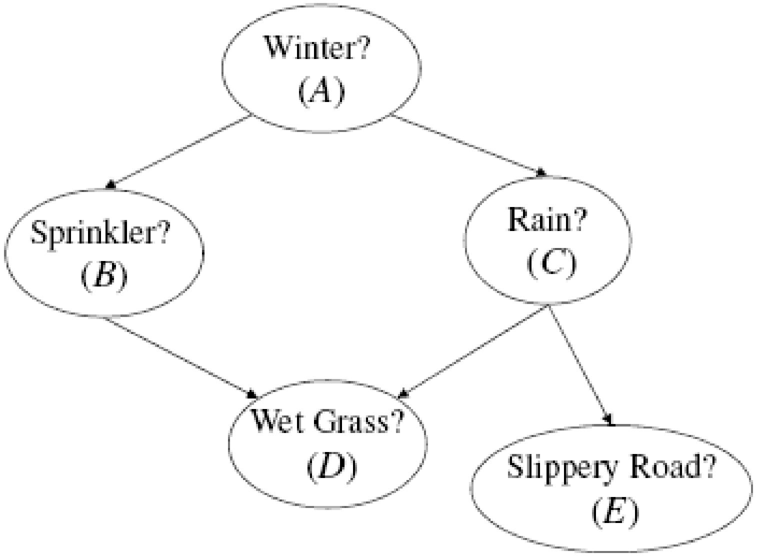
(Reading: class notes chapter 4 , Darwiche chapter 6)

Belief Updating



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

A Bayesian Network



A	Θ_A
true	.6
false	.4

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

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

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

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

Probabilistic Inference Tasks

- Belief updating: E is a subset {X1,…,Xn}, Y subset X-E, P(Y=y|E=e)
- P(e)?

$$\mathbf{BEL(X_i) = P(X_i = x_i \mid evidence)}$$

Finding most probable explanation (MPE)

$$\bar{x}^* = \arg\max_{\bar{x}} P(\bar{x}, e)$$

- Finding maximum a-posteriory hypothesis

$$(a_1^*,\dots,a_k^*) = \arg\max_{\bar{a}} \sum_{X/A} P(\bar{x}, e) \qquad A \subseteq X: \text{ hypothesis variables}$$

- Finding maximum-expected-utility (MEU) decision

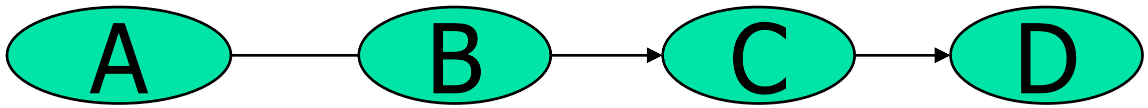
$$(d_1^*,\dots,d_k^*) = \arg\max_{\bar{d}} \sum_{X/D} P(\bar{x}, e) U(\bar{x}) \qquad D \subseteq X: \text{ decision variables}, \quad U(\bar{x}): \text{ utility function}$$

Belief updating is NP-hard

- Each sat formula can be mapped to a Bayesian network query.
- Example: (u,~v,w) and (~u,~w,y) sat?

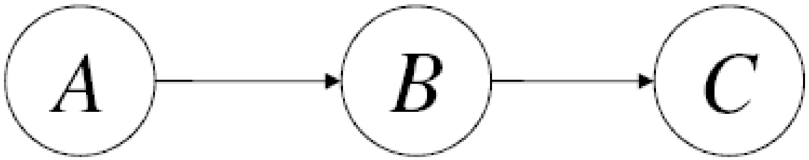
Motivation

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	Θ_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, Θ_A and $\Theta_{B|A}$
- We multiply these factors first and then sum out variable A from the resulting factor.
- Multiplying Θ_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

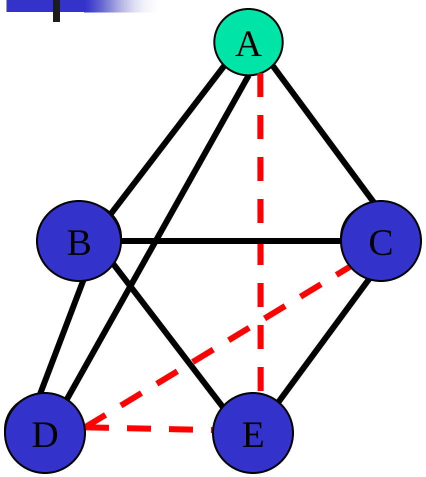
- We now have two factors, $\sum_A \Theta_A \Theta_{B|A}$ and $\Theta_{C|B}$, and we want to eliminate variable B
- Since B appears in both factors, we must multiply them first and then sum out B from the result.
- Multiplying:

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(X|evidence)=?



"Moral" graph

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

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

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

Variable Elimination

Backwards Computation,

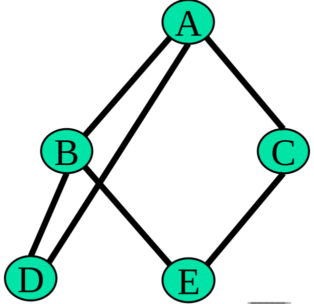
Ordering: a, e, d, c, b

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

$P(a) \sum_{e=0} \sum_d \sum_c P(c|a) \lambda_B(a,d,c,e)$
$P(a) \sum_{e=0} \sum_d \lambda_C(a,d,e)$
$P(a) \sum_{e=0} \lambda_D(a,e)$
$P(a) \lambda_D(a, e=0)$

The bucket elimination Process:

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



"Moral" graph

Backwards Computation = Elimination

Ordering: a, b, c, d, e

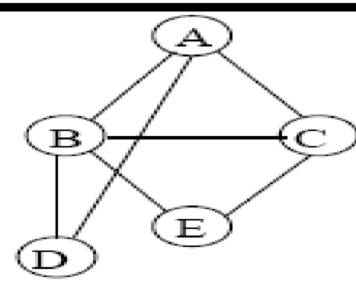
$$P(a)\sum_b P(b|a)\sum_c P(c|a)\sum_d P(d|b,a)\sum_{e=0} P(e|b,c)$$

$$
\begin{aligned}
&= P(a)\sum_b P(b|a)\sum_c P(c|a)P(e=0|b,c)\sum_d P(d|b,a)\\
&= P(a)\sum_b P(b|a)\lambda_D(a,b)\sum_c P(c|a)P(e=0|b,c)\\
&= P(a)\sum_b P(b|a)\lambda_D(a,b)\lambda_C(a,b)\\
&= P(a)\lambda_B(a)
\end{aligned}
$$

The Bucket elimination process:

$$
\begin{aligned}
bucket(E) &= \quad P(e|b,c),\ \ e=0\\
bucket(D) &= \quad P(d|a,b)\\
bucket(C) &= \quad P(c|a)\\
bucket(B) &= \quad P(b|a)\\
bucket(A) &= \quad P(a)
\end{aligned}
$$

Bucket Elimination and Induced Width



Ordering: a, b, c, d, e

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

Ordering: a, e, d, c, b

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

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

Factors: Sum-Out Operation

The sum-out operation is commutative

$$\sum_{Y}\sum_{X} f = \sum_{X}\sum_{Y} f$$

No need to specify the order in which variables are summed out.

If a factor f is defined over disjoint variables $\mathbf{X}$ and $\mathbf{Y}$

then $\sum_{\mathbf{X}} f$ is said to marginalize variables $\mathbf{X}$

If a factor f is defined over disjoint variables $\mathbf{X}$ and $\mathbf{Y}$

then $\sum_{\mathbf{X}} f$ is called the result of projecting f on variables $\mathbf{Y}$

Factors: Multiplication Operation

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

D	E	f_2
true	true	0.448
true	false	0.192
false	true	0.112
false	false	0.248

The result of multiplying the above factors:

B	C	D	E	$f_1(B,C,D)f_2(D,E)$
true	true	true	true	$0.4256 = (.95)(.448)$
true	true	true	false	$0.1824 = (.95)(.192)$
true	true	false	true	$0.0056 = (.05)(.112)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
false	false	false	false	$0.2480 = (1)(.248)$

Factors: Multiplication Operation

The result of multiplying factors $f_1(\mathbf{X})$ and $f_2(\mathbf{Y})$

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

$$(f_1 f_2)(\mathbf{z}) \stackrel{def}{=} f_1(\mathbf{x}) f_2(\mathbf{y}),$$

where $\mathbf{x}$ and $\mathbf{y}$ are compatible with $\mathbf{z}$; that is, $\mathbf{x} \sim \mathbf{z}$ and $\mathbf{y} \sim \mathbf{z}$

Factor multiplication is commutative and associative

It is meaningful to talk about multiplying a number of factors without specifying the order of this multiplication process.

Bucket elimination

Algorithm *BE-bel* (Dechter 1996)

$\sum_b \prod$ ← Elimination operator

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

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

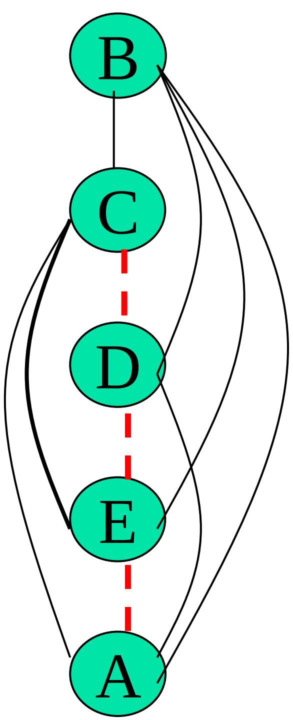
bucket D: $h^C(a,d,e)$

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

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

$P(a|e=0)$

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

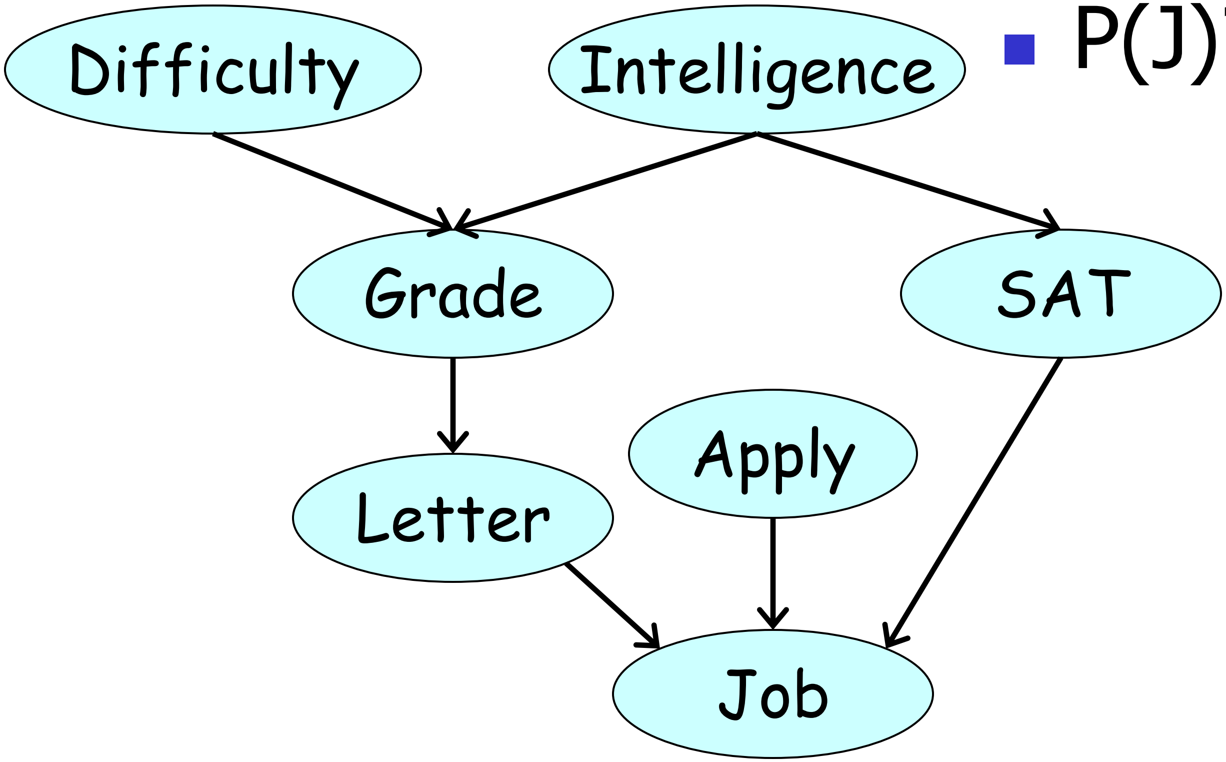


BE-BEL

Input: A belief network $\{P_1, ..., P_n\}$, d,e.
Output: belief of X_1 given e.

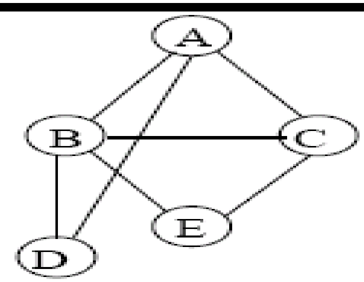
1. **Initialize:**
2. **Process buckets** from $p = n$ to 1
   for matrices $\lambda_1, \lambda_2, ..., \lambda_j$ in $bucket_p$ do
   - **If** (observed variable) $X_p = x_p$ assign $X_p = x_p$ to each λ_i.
   - **Else**, (multiply and sum) $\lambda_p = \sum_{X_p} \Pi_{i=1}^{j} \lambda_i$. Add λ_p to its bucket.
3. **Return** $Bel(x_1) = \alpha P(x_1) \cdot \Pi_i \lambda_i(x_1)$

Student Network example



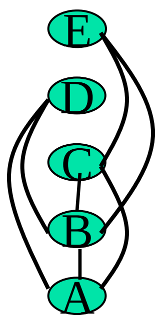
- P(J)?

Bucket Elimination and Induced Width



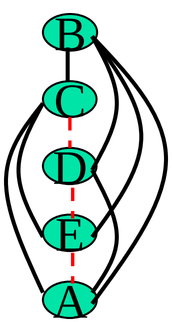
Ordering: a, b, c, d, e

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



Ordering: a, e, d, c, b

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

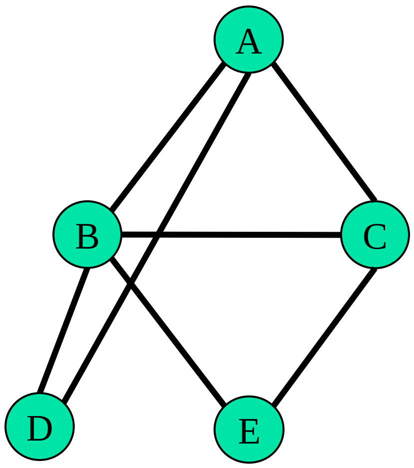


Complexity of elimination

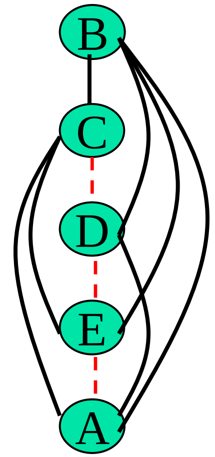
$$O(n \exp(w^*(d)))$$

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

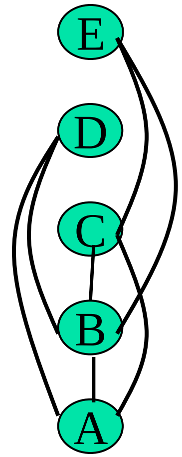
The effect of the ordering:



"Moral" graph



$w^*(d_1) = 4$



$w^*(d_2) = 2$

Complexity of bucket elimination

Theorem

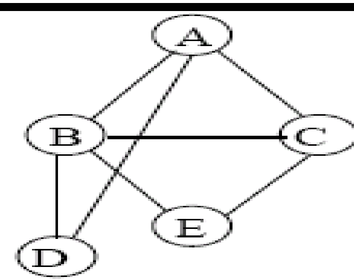
Given a belief network having n variables, observations e, the complexity of BE-BEL along d, is time and space

$$O(n \cdot exp(w*(d))$$

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?

Handling Observations



Observing $b = 1$

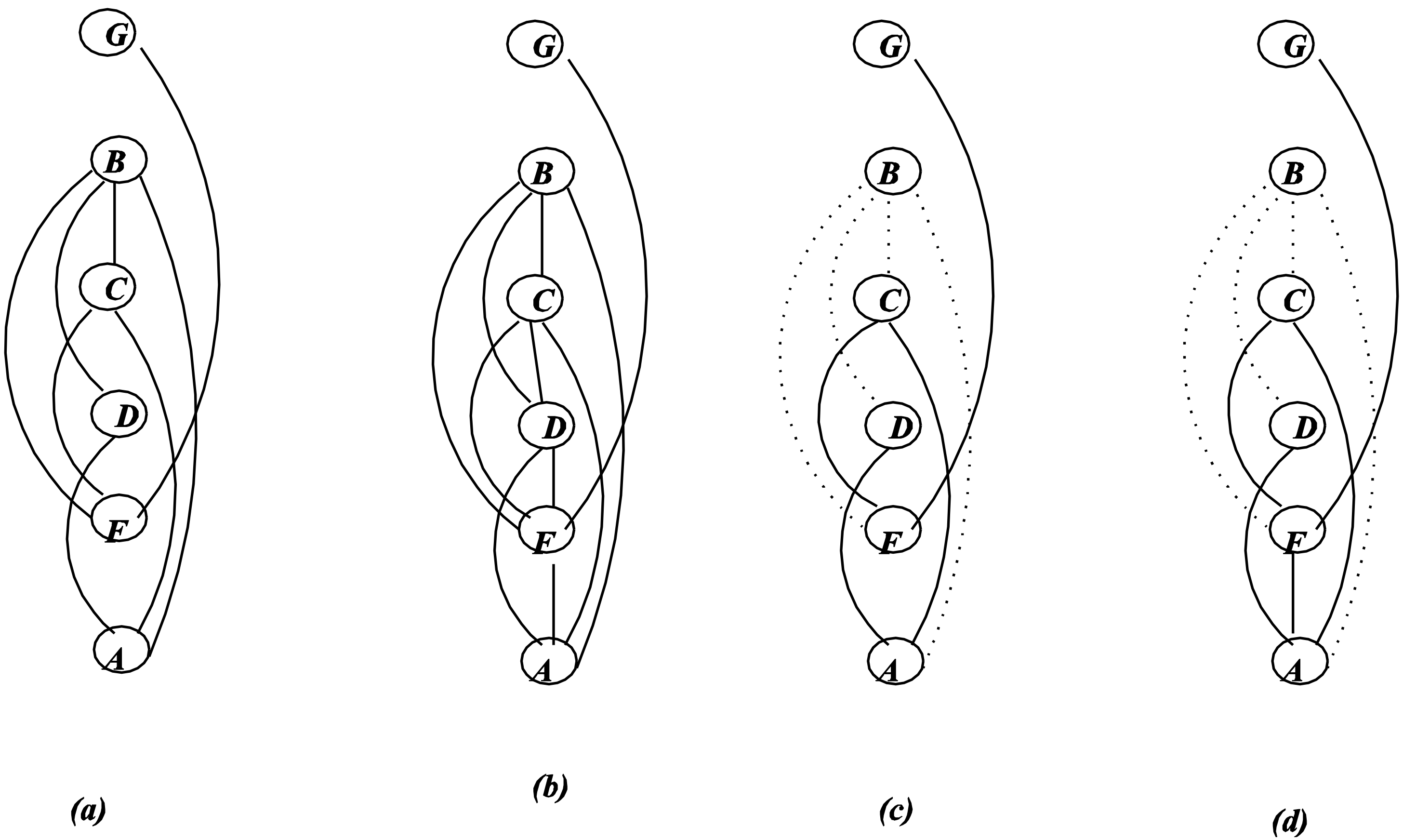
Ordering: a, e, d, c, b

$bucket(B) = \quad P(e|b,c), P(d|a,b), P(b|a), b = 1$
$bucket(C) = \quad P(c|a), \quad || \quad P(e|b=1,c)$
$bucket(D) = \quad || \quad P(d|a, b=1)$
$bucket(E) = \quad e = 0 \quad || \quad \lambda_C(e,a)$
$bucket(A) = \quad P(a), \quad || \quad P(b=1|a) \;\; \lambda_D(a), \lambda_E(e,a)$

Ordering: a, b, c, d, e

$bucket(E) = \quad P(e|b,c), \;\; e = 0$
$bucket(D) = \quad P(d|a,b)$
$bucket(C) = \quad P(c|a) \quad || \quad \lambda_E(b,c)$
$bucket(B) = \quad P(b|a), b = 1 \quad || \quad \lambda_D(a,b), \lambda_C(a,b)$
$bucket(A) = \quad P(a) \quad || \quad \lambda_B(a)$

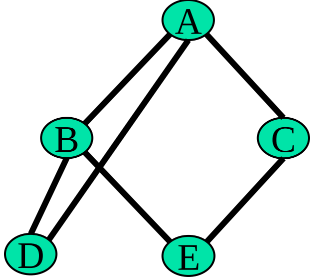
The impact of observations



Ordered graph

Induced graph

Ordered conditioned graph



"Moral" graph

Irrelevant buckets for BE-BEL

Buckets that sum to 1 are **irrelevant**.
Identification: no evidence, no new functions.

Recursive recognition : ($bel(a|e)$)

$bucket(E) =$ $P(e|b,c),\ e = 0$
$bucket(D) =$ $P(d|a,b)$,...skipable bucket
$bucket(C) =$ $P(c|a)$
$bucket(B) =$ $P(b|a)$
$bucket(A) =$ $P(a)$

Complexity: Use induced width in moral graph without irrelevant nodes, then update for evidence arcs.

Use the ancestral graph only

Pruning Nodes

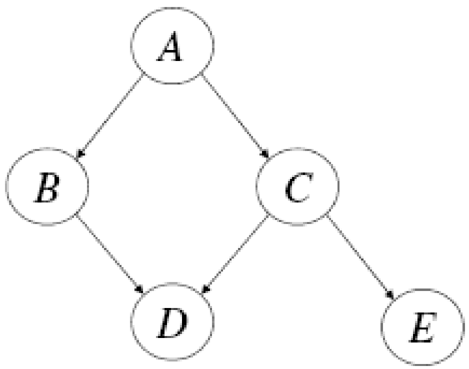
Given a Bayesian network $\mathcal{N}$ and query $(\mathbf{Q}, \mathbf{e})$

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.

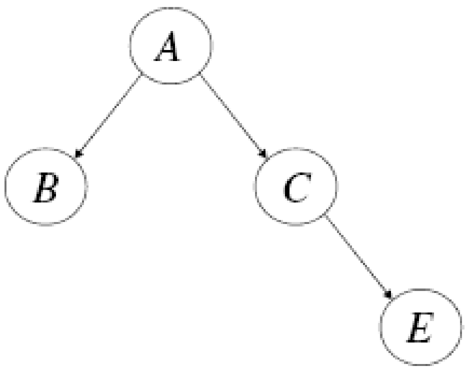
If $\mathcal{N}' = \text{pruneNodes}(\mathcal{N}, \mathbf{Q} \cup \mathbf{E})$

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.

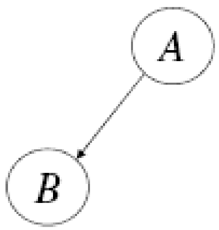
Pruning Nodes: Example



network structure



joint on B, E



joint on B

Pruning Edges

For each edge $U \rightarrow X$ which originates from a node U in $\mathbf{E}$

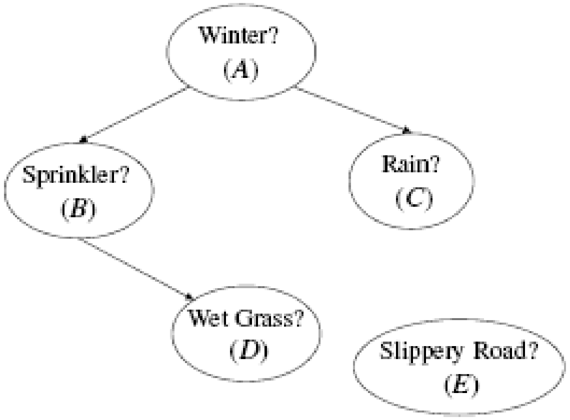
1. Remove the edge $U \rightarrow X$ from the network.
2. Replace the CPT $\Theta_{X|\mathbf{U}}$ for node X by a smaller CPT, which is obtained from $\Theta_{X|\mathbf{U}}$ by assuming the value u of parent U given in evidence $\mathbf{e}$. This new CPT corresponds to $\sum_U \Theta^u_{X|\mathbf{U}}$

If $\mathcal{N}' = \text{pruneEdges}(\mathcal{N}, \mathbf{e})$

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 Edges: Example

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	Θ_A
true	.6
false	.4

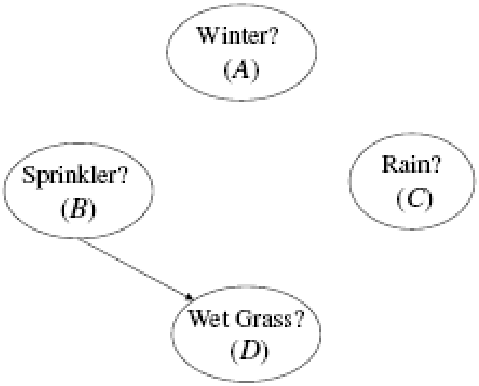
B	D	$\sum_C \Theta_{D\|BC}^{C=\text{false}}$
true	true	.9
true	false	.1
false	true	0
false	false	1

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

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

Pruning Nodes and Edges: Example

B	$\Theta'_B = \sum_A \Theta^{A=\text{true}}_{B\|A}$
true	.2
false	.8



C	$\Theta'_C = \sum_A \Theta^{A=\text{true}}_{C\|A}$
true	.8
false	.2

A	Θ_A
true	.6
false	.4

B	D	$\Theta'_{D\|B} = \sum_C \Theta^{C=\text{false}}_{D\|BC}$
true	true	.9
true	false	.1
false	true	0
false	false	1

Query $\mathbf{Q} = \{D\}$ and $\mathbf{e} : A = \text{true}, C = \text{false}$

Probabilistic Inference Tasks

- Belief updating:

$$\mathrm{BEL}(X_i) = P(X_i = x_i \mid \mathrm{evidence})$$

- Finding most probable explanation (MPE)

$$\bar{x}^* = \arg\max_{\bar{x}} P(\bar{x}, e)$$

- Finding maximum a-posteriory hypothesis

$$(a_1^*, \ldots, a_k^*) = \arg\max_{\bar{a}} \sum_{X/A} P(\bar{x}, e)$$

$A \subseteq X$: **hypothesis variables**

- Finding maximum-expected-utility (MEU) decision

$$(d_1^*, \ldots, d_k^*) = \arg\max_{\bar{d}} \sum_{X/D} P(\bar{x}, e) U(\bar{x})$$

$D \subseteq X$: **decision variables**

$U(\bar{x})$: **utility function**

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

Algorithm *elim-mpe* (Dechter 1996)

$\sum$ 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: $\underbrace{P(b|a) \quad P(d|b,a) \quad P(e|b,c)}$

bucket C: P(c|a) $h^B(\underbrace{a,d,c,e})$

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

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

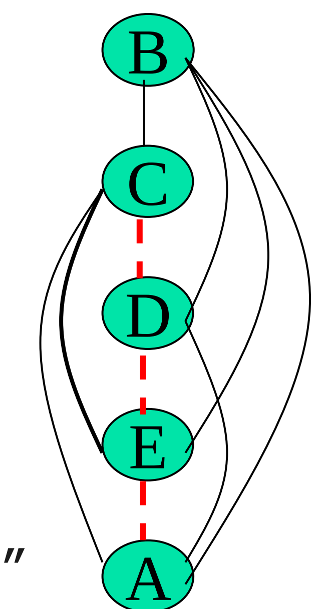
bucket A: P(a) $h^E(a)$

MPE

W*=4

"induced width"

(max clique size)



Generating the MPE-tuple

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

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

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

2. $e' = 0$

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

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

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

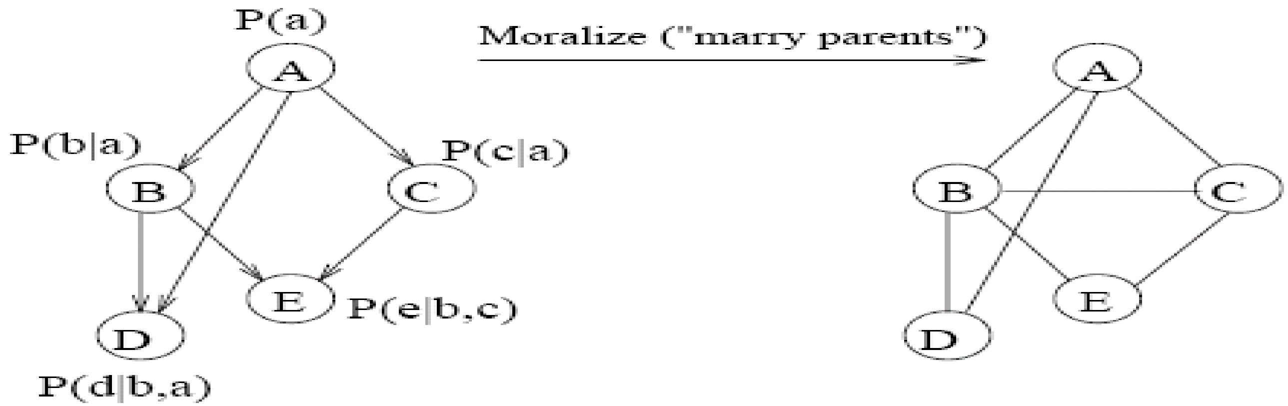
D: $h^C(a, d, e)$

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

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

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

Finding the MPE
(An optimization task)



Ordering: $a,\ b,\ c,\ d,\ e$

$m = \max_{a,b,c,d,e=0} P(a,b,c,d,e) =$

$= \max_a P(a) \max_b P(b|a) \max_c P(c|a) \max_d P(d|b,a)$

$\max_{e=0} P(e|b,c)$

Ordering: $a,\ e,\ d,\ c,\ b$

$m = \max_{a,e=0,d,c,b} P(a,b,c,d,e)$

$m = \max_a P(a) \max_e \max_d \cdot$

$\max_c P(c|a) \max_b P(b|a) P(d|a,b) P(e|b,c)$

Algorithm BE-MPE

Algorithm BE-mpe

Input: A belief network $\mathcal{B} =< X, D, G, \mathcal{P} >$, where $\mathcal{P} = \{P_1, ..., P_n\}$; an ordering of the variables, $d = X_1, ..., X_n$; observations e.

Output: The most probable assignment.

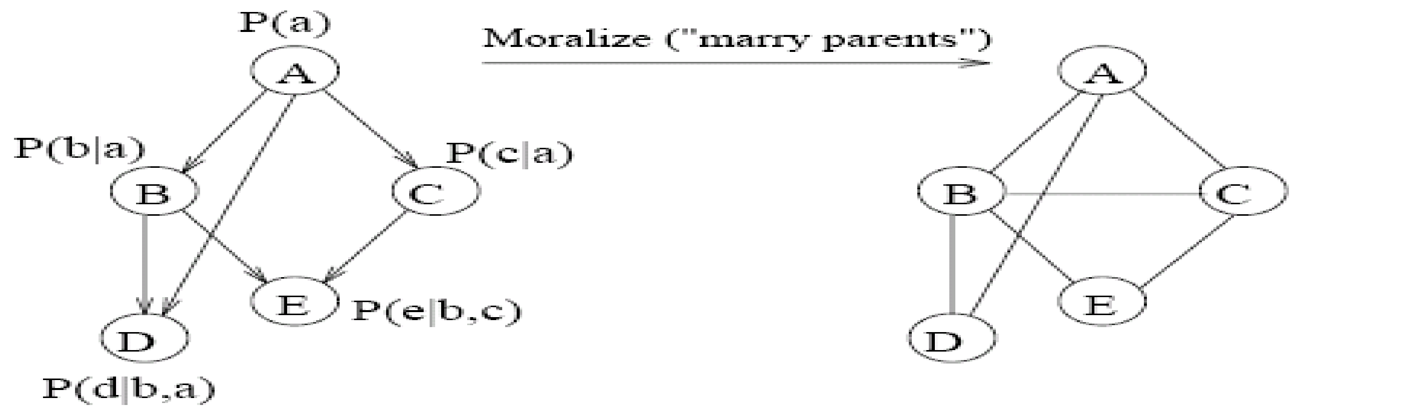
1. **Initialize:** Generate an ordered partition of the conditional probability matrices, $bucket_1, \ldots, bucket_n$, where $bucket_i$ contains all matrices whose highest variable is X_i. Put each observed variable in its bucket. Let $S_1, ..., S_j$ be the subset of variables in the processed bucket on which matrices (new or old) are defined.

2. **Backward:** For $p \leftarrow n$ downto 1, do
for all the matrices $h_1, h_2, ..., h_j$ in $bucket_p$, do

- **If** (observed variable) $bucket_p$ contains $X_p = x_p$, assign $X_p = x_p$ to each h_i and put each in appropriate bucket.
- **else**, $U_p \leftarrow \bigcup_{i=1}^{j} S_i - \{X_p\}$. Generate functions $h_p = \max_{X_p} \Pi_{i=1}^{j} h_i$ and $x_p^o = argmax_{X_p} h_p$. Add h_p to bucket of largest-index variable in U_p.

3. **Forward:** The mpe value is obtained by maximizing over $X_1$the product in $bucket_1$.
An mpe tuple is obtained by assigning values in the ordering d consulting recorded functions in each bucket as follows.
Given the assignment $x = (x_1, ..., x_{i-1})$ choose $x_i = x_i^o(x)$ (x_i^o is in $bucket_i$), or Choose $x_i = argmax_{X_i} \Pi_{\{h_j \in \ bucket_i | \ x=(x_1,...,x_{i-1})\}} h_j$

Finding the MAP
(An optimization task)



Variables A and B are the hypothesis variables.
Ordering: $a,\ b,\ c,\ d,\ e$
$max_{a,b}P(a,b,e=0) = \max_{a,b}\sum_{c,d,e=0} P(a,b,c,d,e)$
$= \max_a P(a) \max_b P(b|a) \sum_c P(c|a) \sum_d P(d|b,a)$
$\sum_{e=0} P(e|b,c)$

Ordering: $a,\ e,\ d,\ c,\ b$ .... illegal ordering
$\max_{a,b} P(a,e,e=0) = \max_{a,b} \sum_P (a,b,c,d,e)$
$\max_{a,b} P(a,b,e=0) = \max_a P(a) \max_b P(b|a) \sum_d \cdot$
$\max_c P(c|a) P(d|a,b) P(e=0|b,c)$

Algorithm BE-MAP

Variable ordering:
Restricted: Max buckets should
Be processed after sum buckets

Algorithm BE-map

Input: A Bayesian network $\mathcal{P} =< X, D, P_G, \Pi > P = \{P_1, ..., P_n\}$; a subset of hypothesis variables $A = \{A_1, ..., A_k\}$; an ordering of the variables, d, in which the A's are first in the ordering; observations e.

Output: A most probable assignment $A = a$.

1. **Initialize:** Generate an ordered partition of the conditional probability matrices, $bucket_1, \ldots, bucket_n$, where $bucket_i$ contains all matrices whose highest variable is X_i.

2. **Backwards** For $p \leftarrow n$ downto 1, do
for all the matrices $\beta_1, \beta_2, ..., \beta_j$ in $bucket_p$, do

- **If** (observed variable) $bucket_p$ contains the observation $X_p = x_p$, assign $X_p = x_p$ to each β_i and put each in appropriate bucket.
- **else**, $U_p \leftarrow \bigcup_{i=1}^{j} S_i - \{X_p\}$. If X_p is not in A, then $\beta_p = \sum_{X_p} \Pi_{i=1}^{j} \beta_i$; else, $X_p \in A$, and $\beta_p = \max_{X_p} \Pi_{i=1}^{j} \beta_i$ and $a^0 = argmax_{X_p} \beta_p$. Add β_p to the bucket of the largest-index variable in U_p.

3. **Forward:** Assign values, in the ordering $d = A_1, ..., A_k$, using the information recorded in each bucket.

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

$$O(n \cdot exp(w*(d))$$

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

- NP-complete
- A tree has induced-width of ?
- Greedy algorithms:
  - Min width
  - Min induced-width
  - Max-cardinality
  - Fill-in (thought as the best)
  - See anytime min-width (Gogate and Dechter)

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

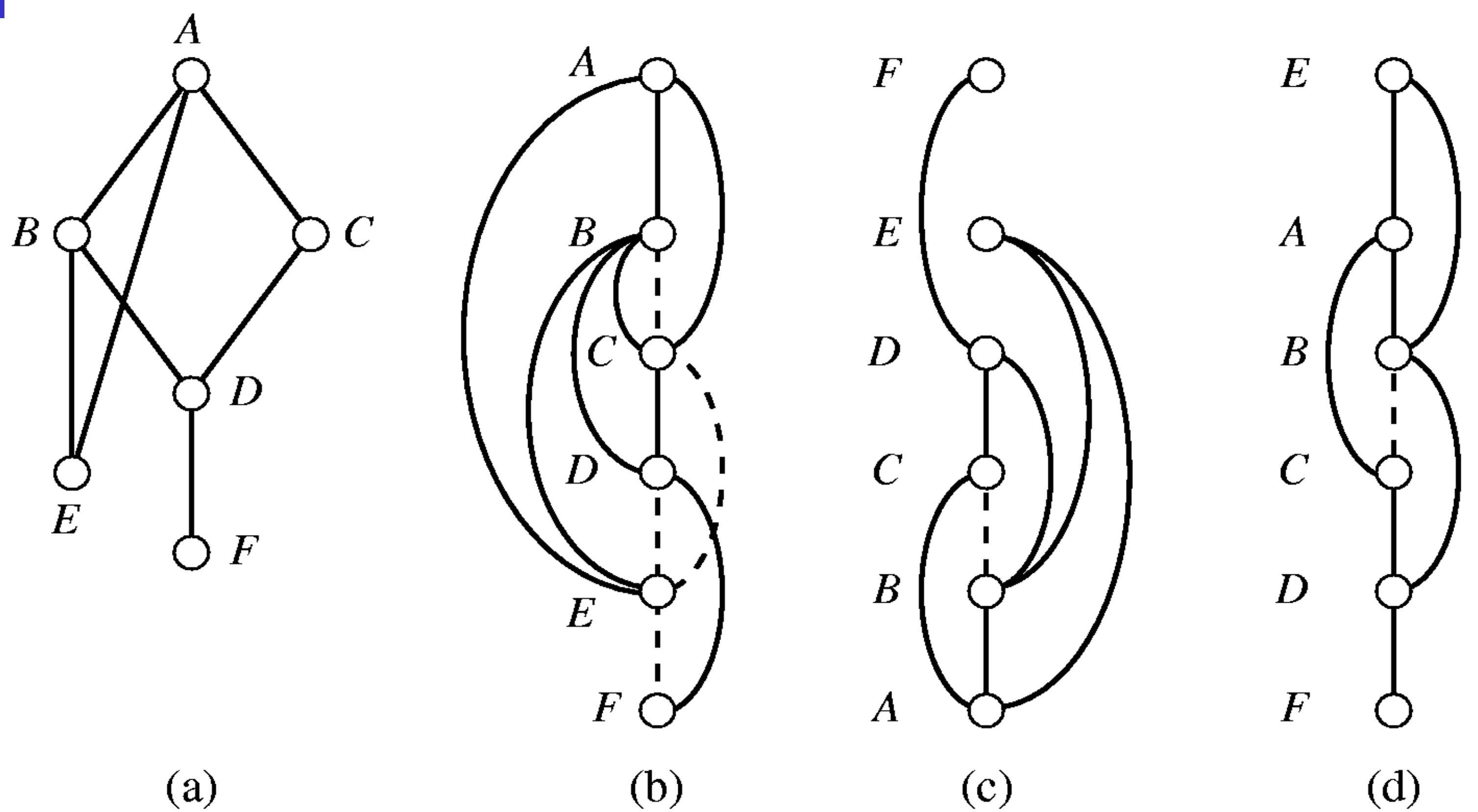
Greedy orderings heuristics

min-induced-width (miw)
input: a graph *G = (V;E), V = {1; :::; vn}*
output: An ordering of the nodes *d = (v1; :::; vn).*
1. for *j = n to 1 by -1 do*
2. *r ← a node in V with smallest degree.*
3. put *r in position j.*
4. connect *r's neighbors: E ← E union {(vi; vj)| (vi; r) in E; (vj ; r) 2 in E},*
5. remove *r from the resulting graph: V ←V - {r}.*

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

min-fill (min-fill)
input: a graph *G = (V;E), V = {v1; :::; vn}*
output: An ordering of the nodes *d = (v1; :::; vn).*
1. for *j = n to 1 by -1 do*
2. *r ←a node in V with smallest fill edges for his parents.*
3. put *r in position j.*
4. connect *r's neighbors: E ←E union {(vi; vj)| (vi; r) 2 E; (vj ; r) in E},*
5. remove *r from the resulting graph: V ←V –{r}.*

Different Induced-graphs



Min-induced-width

MIN-INDUCED-WIDTH (MIW)

input: a graph $G = (V, E)$, $V = \{v_1, ..., v_n\}$
output: An ordering of the nodes $d = (v_1, ..., v_n)$.
1. **for** $j = n$ to 1 by -1 do
2. $r \leftarrow$ a node in V with smallest degree.
3. put r in position j.
4. connect r's neighbors: $E \leftarrow E \cup \{(v_i, v_j) | (v_i, r) \in E, (v_j, r) \in E\}$,
5. remove r from the resulting graph: $V \leftarrow V - \{r\}$.

Figure 4.3: The min-induced-width (MIW) procedure

Min-fill algorithm

- 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)

Chordal graphs and Max-cardinality ordering

- 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
- If G* is an induced graph it is chordal chord
- K-trees are special chordal graphs (A graph is a k-tree if all its max-clique are of size k+1, created recursively by connection a new node to k earlier nodes in a cliques
- Finding the max-clique in chordal graphs is easy (just enumerate all cliques in a max-cardinality ordering

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**

Figure 4.5 The max-cardinality (MC) ordering procedure.