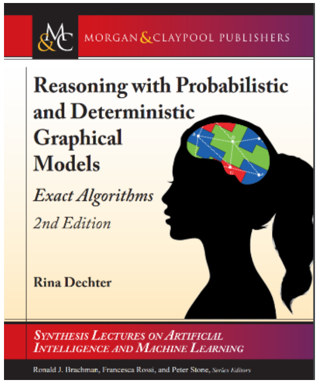
CS 295: Causal Reasoning

Rina Dechter

Exact Inference Algorithms
Bucket-elimination

[Information on the project]



Dechter chapter 4

Outline

- The do calculus (review)
- Bayesian networks, representation and inference
- Class project

Outline

- The do calculus (review)
- Bayesian networks, representation and inference
- Class project

Rules of Do-Calculus

Theorem 3.4.1. *The following transformations are valid for any do-distribution induced by a causal model M:*

Rule 1: Adding/removing Observations

$$P(y|do(x),z,w)=P(y|do(x),w) \quad \text{if} \quad (Z \perp\!\!\!\perp Y \mid W)_{G_{\overline{X}}}$$

Rule 2: Action/observation exchange

$$P(y|do(x),do(z),w)=P(y|do(x),z,w) \quad \text{if} \quad (Z \perp\!\!\!\perp Y \mid X, W)_{G_{\overline{X}\underline{Z}}}$$

Rule 3: Adding/removing Actions

$$P(y|do(x),do(z),w)=P(y|do(x),w) \quad \text{if} \quad (Z \perp\!\!\!\perp Y \mid X, W)_{G_{\overline{X}\overline{Z(W)}}}$$

where $Z(W)$ is the set of Z-nodes that are not ancestors of any W-node in $G_{\overline{X}}$.

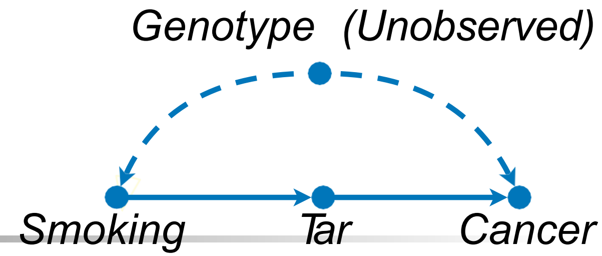
Properties of Do-Calculus

Theorem (soundness and completeness of do-calculus for causal identifiability from P(v)).

The causal quantity $Q = P(y|do(x))$ is identifiable from $P(v)$ and G if and only if there exists a sequence of application of the rules of do-calculus and the probability axioms that reduces Q into a do-free expression.

Syntactic goal: Re-express original Q without do()!

Derivation in Do-Calculus



$P(c \mid do(s)) = \sum_t P(c \mid do(s), t) P(t \mid do(s))$	Probability Axioms	
$= \sum_t P(c \mid do(s), do(t)) P(t \mid do(s))$	Rule 2 $(T \perp\!\!\!\perp C \mid S)_{G_{\underline{T}}}$	
$= \sum_t P(c \mid do(t)) P(t \mid do(s))$	Rule 3 $(S \perp\!\!\!\perp C \mid T)_{G_{\overline{C,T}}}$	
$= \sum_t P(c \mid do(t)) P(t \mid s)$	Rule 2 $(S \perp\!\!\!\perp T)_{G_{\underline{S}}}$	
$= \sum_t \sum_{s'} P(c \mid do(t), s') P(s' \mid do(t)) P(t \mid s)$	Probability Axioms	
$= \sum_t \sum_{s'} P(c \mid t, s') P(s' \mid do(t)) P(t \mid s)$	Rule 2 $(T \perp\!\!\!\perp C \mid S)_{G_{\underline{T}}}$	
$= \sum_t \sum_{s'} P(c \mid t, s') P(s') P(t \mid s)$	Rule 3 $(T \perp\!\!\!\perp S)_{G_{\overline{T}}}$	

Non-identifiability Machinery

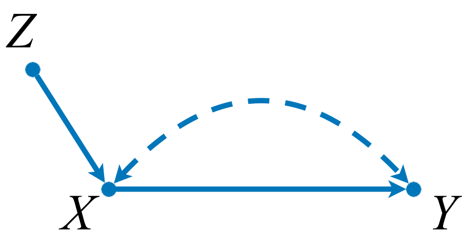
Lemma (Graph-subgraph ID (Tian and Pearl, 2002))

- *If $Q = P(y \mid do(x))$ is not identifiable in G, then Q is not identifiable in the graph resulting from adding a directed or bidirected edge to G.*
- Converse. *If $Q = P(y|do(x))$ is identifiable in G, Q is still identifiable in the graph resulting from removing a directed or bidirected edge from G.*

Non-identifiability Machinery

- *Proof idea. Suppose* M_1, M_2 *induce the same* $P(\mathbf{v})$ *but differ in* $P(y|do(x))$. *Construct two new models* M_1', M_2' *with any* $P(z)$ *and let*
  $P_i'(x|z,u_{xy})=P_i(x|u_{xy}),\ i=1,2.$

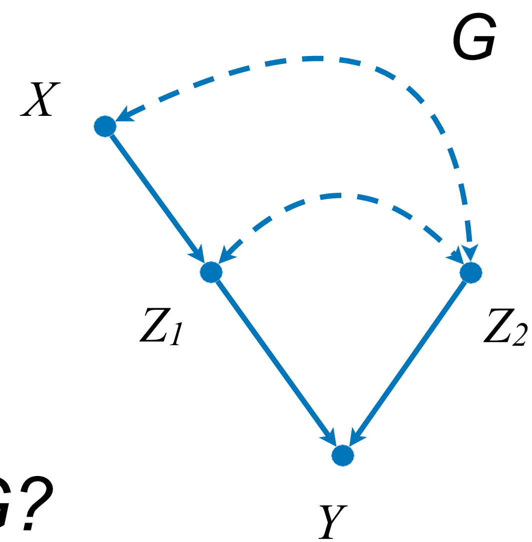
  This construction entails
  $P_1'(y|do(x))\neq P_2'(y|do(x)).$



Question: Do all non-ID models look like the bow graph?

Non-identifiability Puzzle

- *Is P(y | do(x)) identifiable from G?*
    - *Is G of bow-shape?*



- *Is P(y | do(x), z2) identifiable from G?*
- *Is P(y | do(x, z2)) identifiable from G?*

P(Y|do(x) is not identifiable
But when conditioning on Z_1, or Z_2 they are.
So, computing the effect of a joint intervention can be easier than
Their individual interventions.

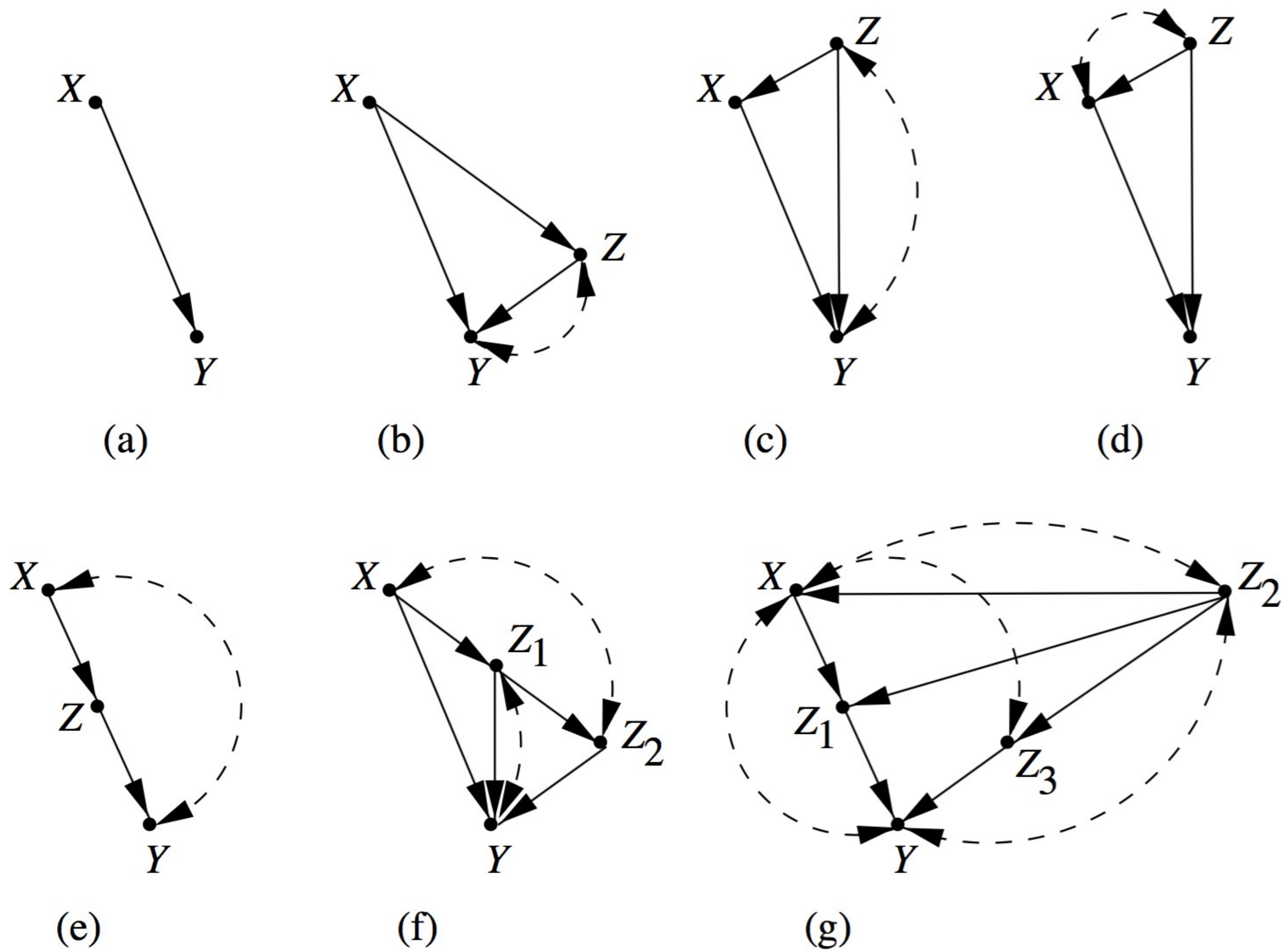
[C] sec 35.

Non-Identifiability Criterion

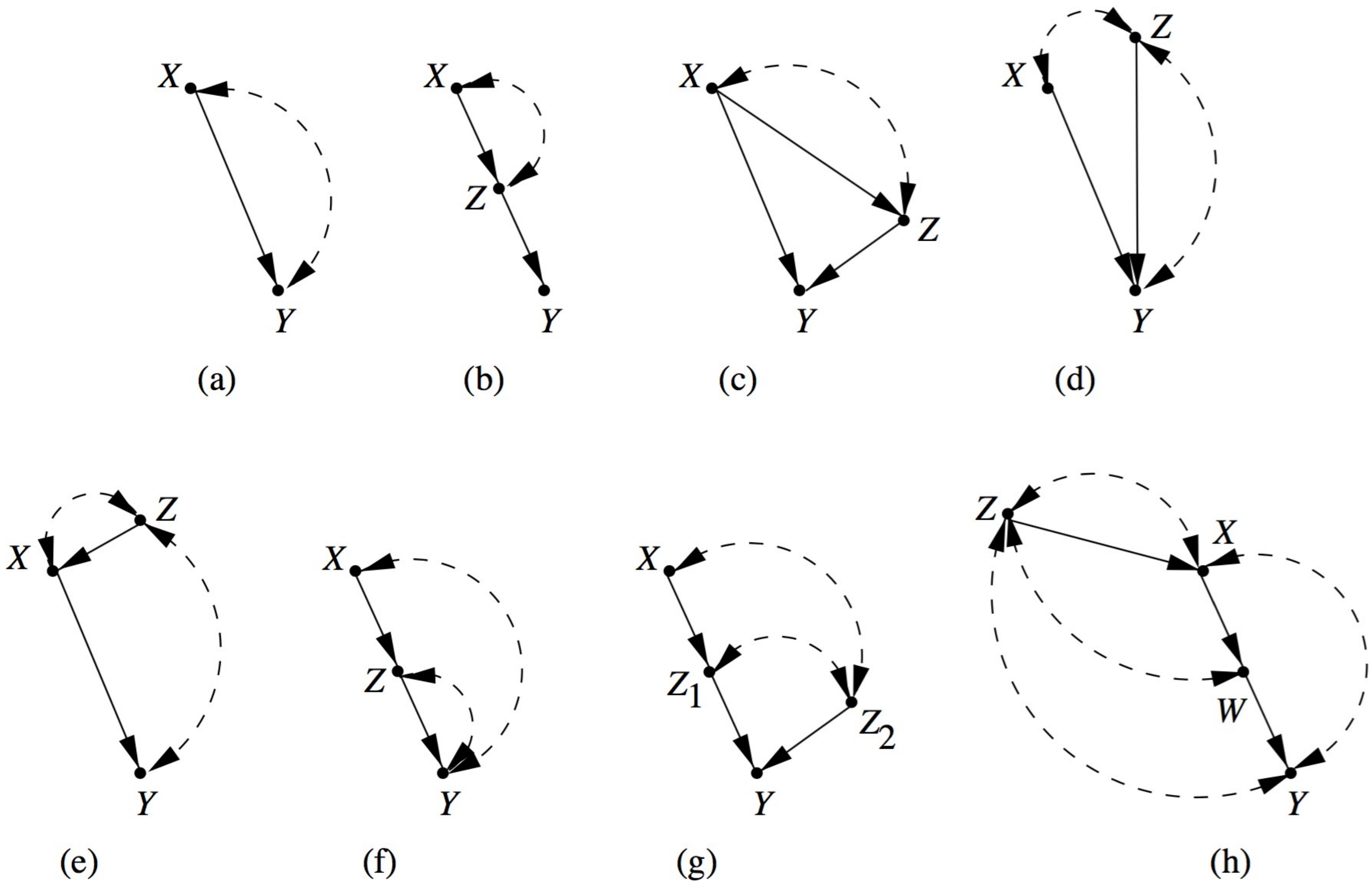
Theorem (Graphical criterion for non-identifiability of joint interventional distributions (Tian, 2002)).

- If there is a bidirected path connecting X to any of its children in G, then $P(\mathbf{v}|do(x))$ is not identifiable from $P(\mathbf{v})$ and G.

Some Identifiable Graphs



Some Non-Identifiable Graphs



Summary

- *The do-calculus provides a syntactical characterization to the problem of policy evaluation for atomic interventions.*
- *The problem of confounding and identification is essentially solved, non-parametrically.*
- *Simpson's Paradox is mathematized and dissolved.*
- *Applications are pervasive in the social and health sciences as well as in statistics, machine learning, and artificial intelligence.*

Outline

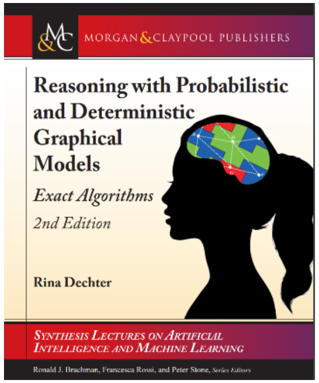
- The do calculus (review)
- Bayesian networks, representation and inference
- Class project

CS 295: Causal Reasoning

Rina Dechter

Exact Inference Algorithms
Bucket-elimination

Information on the project



Dechter chapter 4

Chain Rule and Factorization

Overcome the problem of exponential size by exploiting conditional independence

- The chain rule of probabilities:

$$\begin{aligned} P(X_1, X_2) &= P(X_1)P(X_2|X_1) \\ P(X_1, X_2, X_3) &= P(X_1)P(X_2|X_1)P(X_3|X_1, X_2) \\ &\dots \\ P(X_1, X_2, \dots, X_n) &= P(X_1)P(X_2|X_1)\dots P(X_n|X_1, \dots, X_{n-1}) \\ &= \prod_{i=1}^{n} P(X_i|X_1, \dots, X_{i-1}). \end{aligned}$$

- No gains yet. The number of parameters required by the factors is: $2^{n-1} + 2^{n-1} + \ldots + 1 = 2^n - 1$.

Conditional Independence

- About $P(X_i|X_1, \ldots, X_{i-1})$:
  - Domain knowledge usually allows one to identify a subset $pa(X_i) \subseteq \{X_1, \ldots, X_{i-1}\}$ such that
    - Given $pa(X_i)$, X_i is independent of all variables in $\{X_1, \ldots, X_{i-1}\} \setminus pa(X_i)$, i.e.

$$P(X_i|X_1, \ldots, X_{i-1}) = P(X_i|pa(X_i))$$

- Then

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|pa(X_i))$$

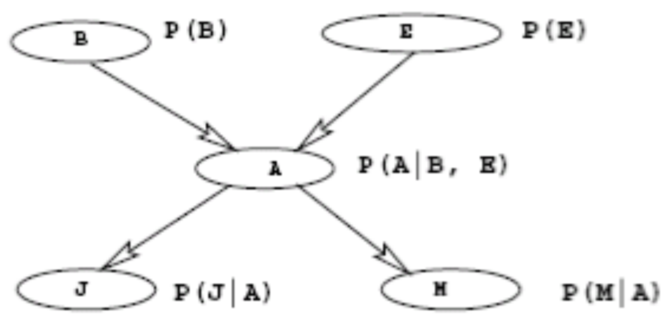
- Joint distribution factorized.
- The number of parameters might have been substantially reduced.

From Factorizations to Bayesian Networks

Graphically represent the conditional independency relationships:

- construct a directed graph by drawing an arc from X_j to X_i iff $X_j \in pa(X_i)$

$$pa(B) = \{\}, \; pa(E) = \{\}, \; pa(A) = \{B, E\}, \; pa(J) = \{A\}, \; pa(M) = \{A\}.$$



- Also attach the conditional probability (table) $P(X_i|pa(X_i))$ to node X_i.
- What results in is a Bayesian network. Also known as **belief network, probabilistic network**.

Graphs Convey Independence Statements

- Directed graphs by graph's d-separation
- Undirected graphs by graph separation
- Goal: capture probabilistic conditional independence by graphs.
- We focus on directed graphs here.

Capturing Independence Graphically

These examples of independence are all implied by a formal interpretation of each DAG as a set of conditional independence statements.

Given a variable V in a DAG G:

$\text{Parents}(V)$ are the parents of V in DAG G, that is, the set of variables N with an edge from N to V.

$\text{Descendants}(V)$ are the descendants of V in DAG G, that is, the set of variables N with a directed path from V to N (we also say that V is an ancestor of N in this case).

$\text{Non\_Descendants}(V)$ are all variables in DAG G other than V, $\text{Parents}(V)$ and $\text{Descendants}(V)$. We will call these variables the non-descendants of V in DAG G.

Capturing Independence Graphically

We will formally interpret each DAG G as a compact representation of the following independence statements (Markovian assumptions):

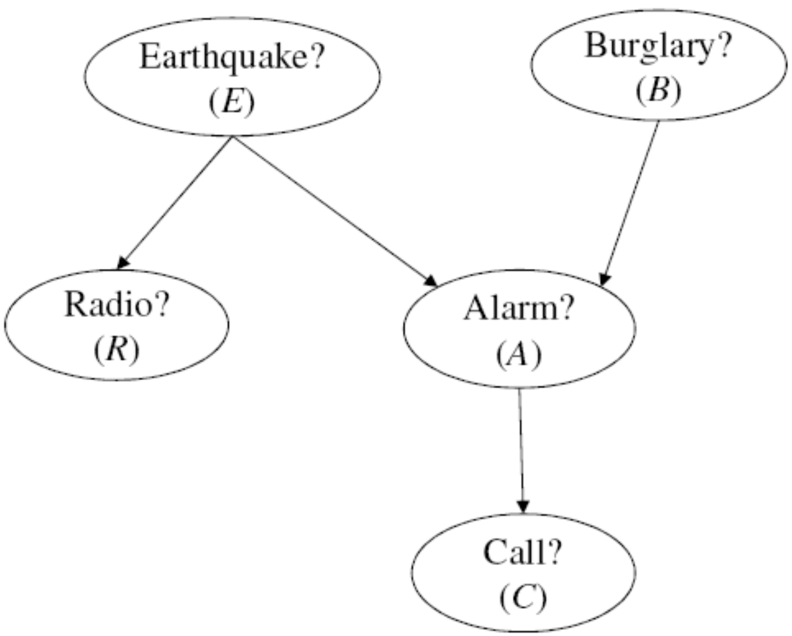
$$I(V, \mathrm{Parents}(V), \mathrm{Non\_Descendants}(V)),$$

for all variables V in DAG G.

- If we view the DAG as a causal structure, then $\mathrm{Parents}(V)$ denotes the direct causes of V and $\mathrm{Descendants}(V)$ denotes the effects of V.
- Given the direct causes of a variable, our beliefs in that variable will no longer be influenced by any other variable except possibly by its effects.

Capturing Independence Graphically

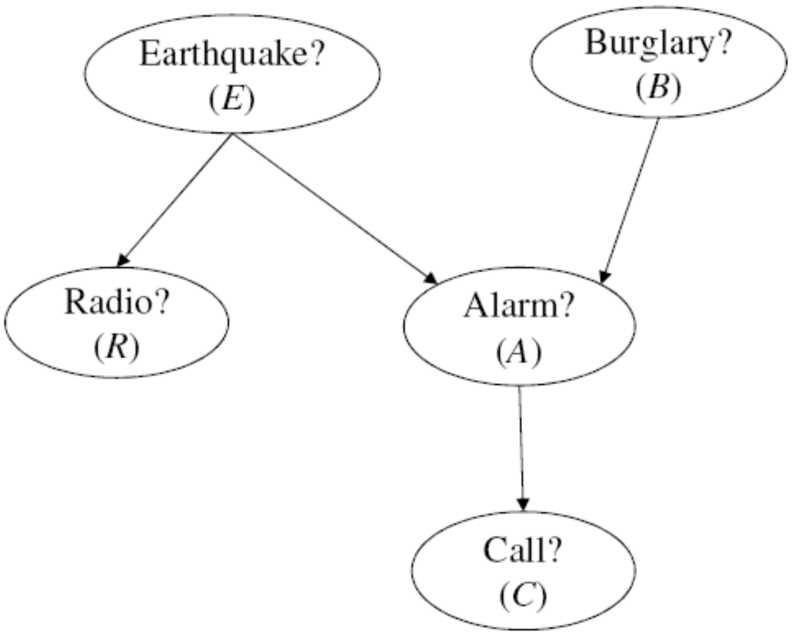
What are the Markov assumptions here?



Note that variables B and E have no parents, hence, they are marginally independent of their non-descendants.

Capturing Independence Graphically

What are the Markov assumptions here?



$I(C, A, \{B, E, R\})$
$I(R, E, \{A, B, C\})$
$I(A, \{B, E\}, R)$
$I(B, \emptyset, \{E, R\})$
$I(E, \emptyset, B)$

Note that variables B and E have no parents, hence, they are marginally independent of their non-descendants.

Capturing Independence Graphically

The formal interpretation of a DAG as a set of conditional independence statements makes no reference to the notion of causality, even though we have used causality to motivate this interpretation.

If one constructs the DAG based on causal perceptions, then one would tend to agree with the independencies declared by the DAG.

It is perfectly possible to have a DAG that does not match our causal perceptions, yet we agree with the independencies declared by the DAG.

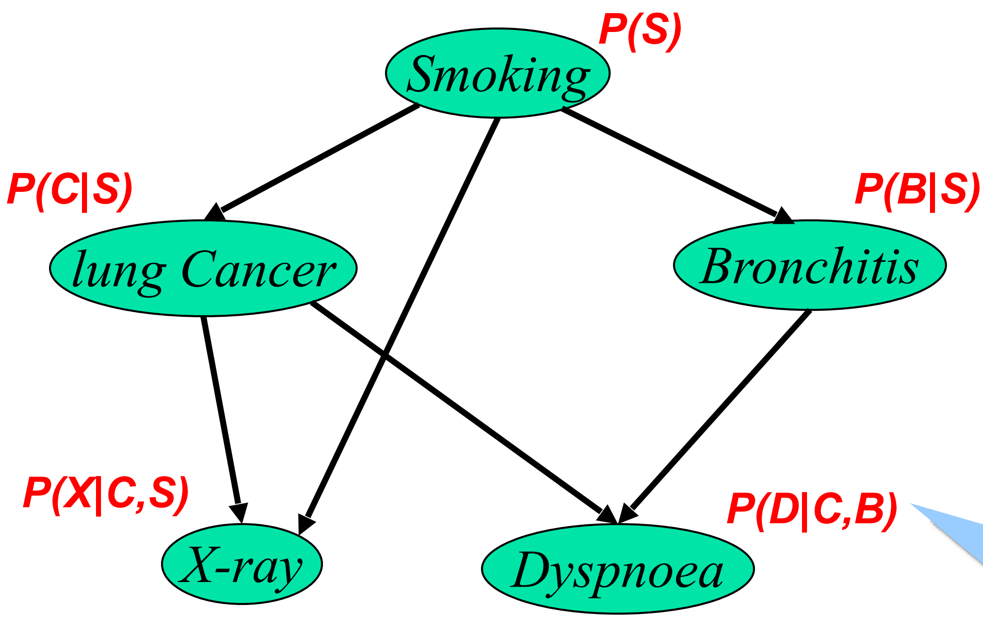
Inference for probabilistic networks

- 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)



CPD:

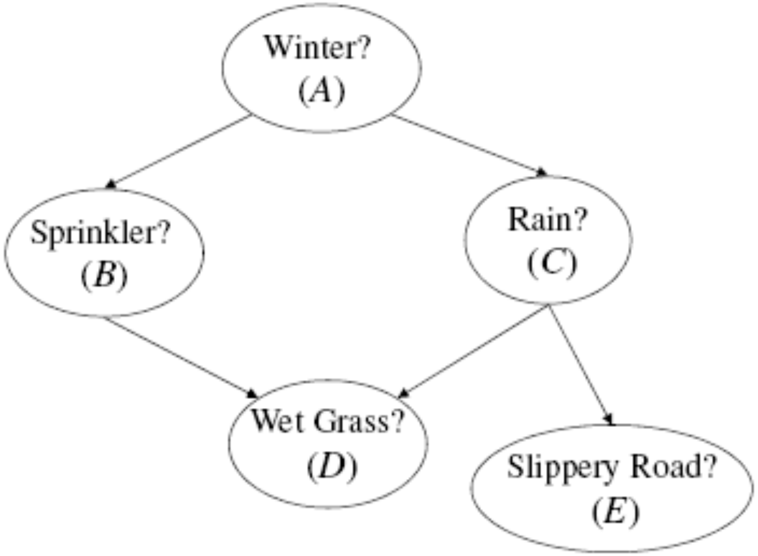
C	B	P(D|C,B)	
0	0	0.1	0.9
0	1	0.7	0.3
1	0	0.8	0.2
1	1	0.9	0.1

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

Belief Updating:

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

A Bayesian Network



A	Θ_A
true	.6
false	.4

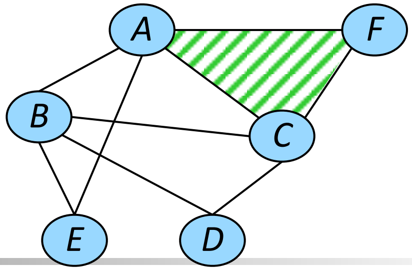
A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

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

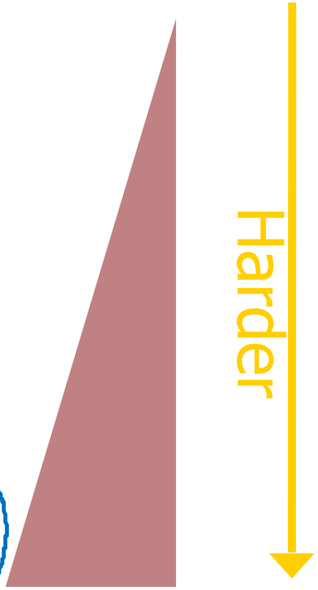
B	C	D	$\Theta_{D\|BC}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

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

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



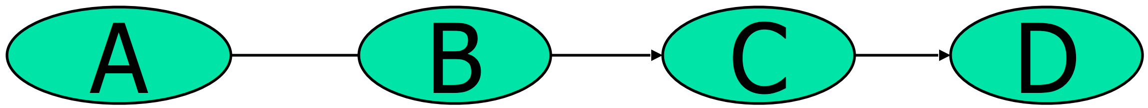
- **NP-hard**: exponentially many terms
- We will focus on exact and then on **approximation** algorithms
  - **Anytime**: very fast & very approximate ! Slower & more accurate

Belief Updating is NP-hard

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

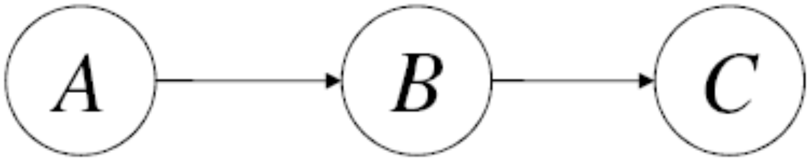
A Simple Network

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	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

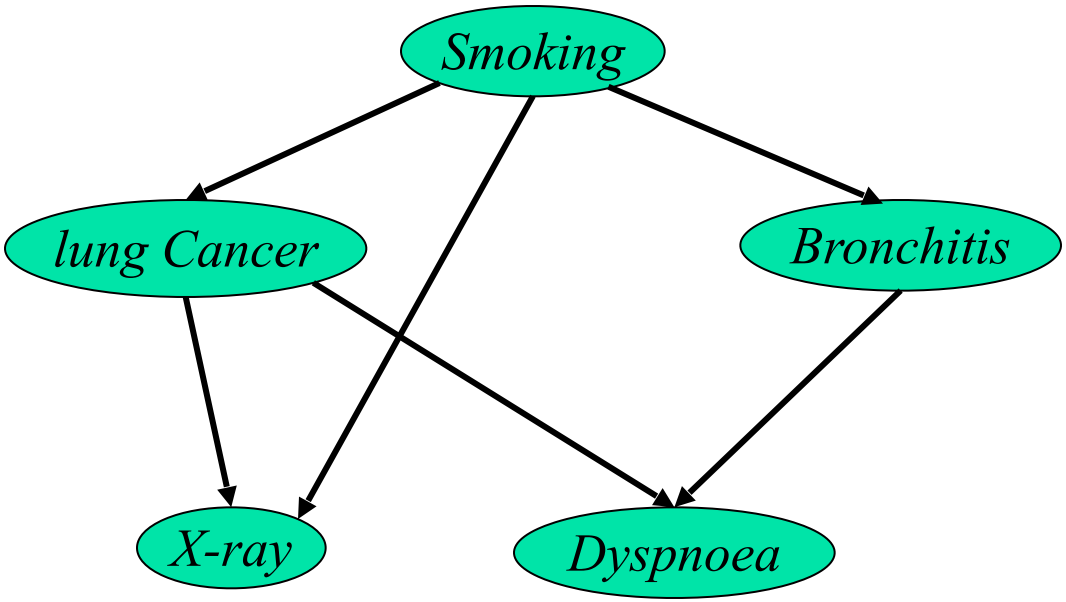
- We now have two factors, $\sum_A \Theta_A \Theta_{B|A}$ and $\Theta_{C|B}$, and we want to eliminate variable B
- Since B appears in both factors, we must multiply them first and then sum out B from the result.
- Multiplying:

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:

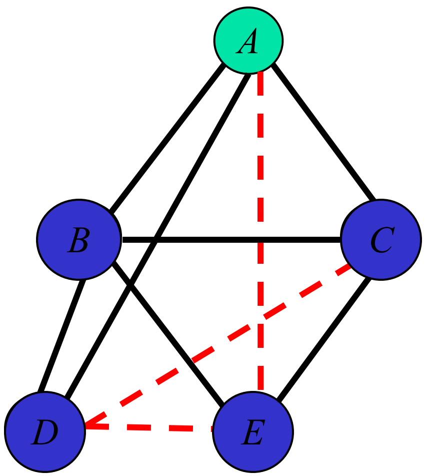
C	$\sum_B \Theta_{C\|B} \sum_A \Theta_A \Theta_{B\|A}$
true	.376
false	.624

Belief Updating



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

Belief updating: P(X|evidence)=?



“Moral” graph

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

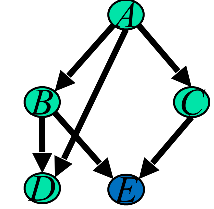
$$P(a)P(b|a)P(c|a)P(d|b,a)P(e|b,c)=$$

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

Variable Elimination

Bucket elimination

Algorithm BE-bel (Dechter 1996)



Elimination operator

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

bucket C: $P(c|a)$

bucket D:

bucket E: $e=0$

bucket A: $P(a)$

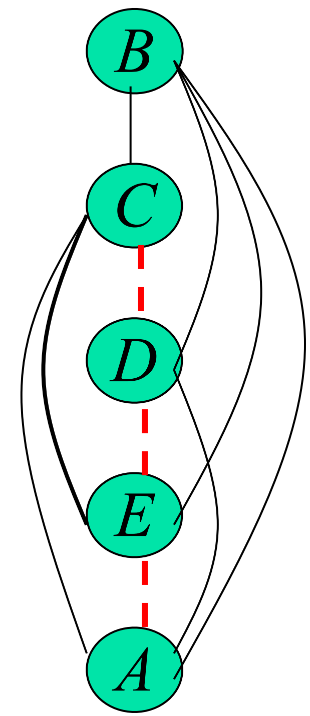
$P(e=0)$

$P(a,e=0)$

$W^*=4$

"induced width"

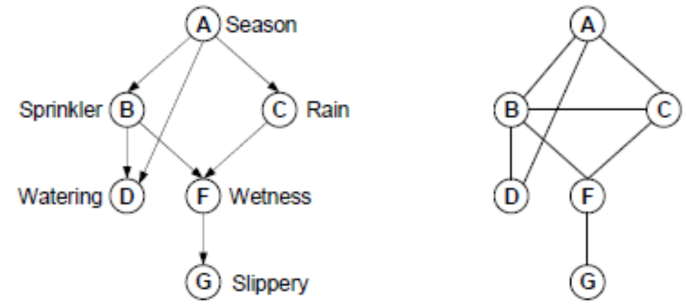
(max clique size)



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

A Bayesian Network

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



(a) Directed acyclic graph (b) Moral graph

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

$$P(a, g=1) = P(a)\sum_{c} P(c|a)\sum_{b} P(b|a)\sum_{f} P(f|b,c)\sum_{d} P(d|b,a)\sum_{g=1} P(g|f). \qquad (4.1)$$

$$P(a, g=1) = P(a)\sum_{c} P(c|a)\sum_{b} P(b|a)\sum_{f} P(f|b,c)\lambda_G(f)\sum_{d} P(d|b,a). \qquad (4.2)$$

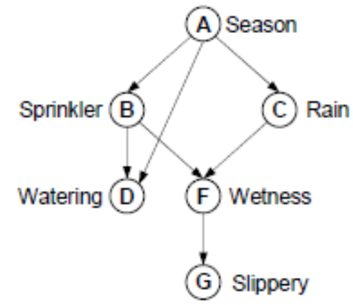
$$P(a, g=1) = P(a)\sum_{c} P(c|a)\sum_{b} P(b|a)\lambda_D(a,b)\sum_{f} P(f|b,c)\lambda_G(f) \qquad (4.3)$$

$$P(a, g=1) = P(a)\sum_{c} P(c|a)\sum_{b} P(b|a)\lambda_D(a,b)\lambda_F(b,c) \qquad (4.4)$$

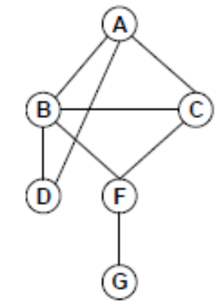
$$P(a, g=1) = P(a)\sum_{c} P(c|a)\lambda_B(a,c) \qquad (4.5)$$

A Bayesian Network

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



(a) Directed acyclic graph



(b) Moral graph

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

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

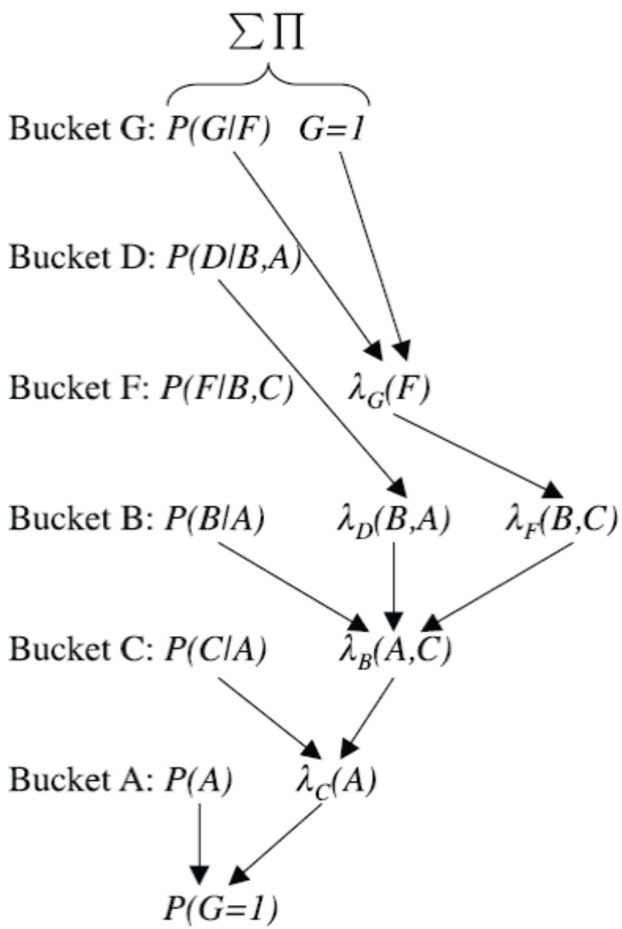
$$P(a, g=1) = P(a)\sum_c P(c|a)\sum_b P(b|a)\sum_f P(f|b,c)\lambda_G(f)\sum_d P(d|b,a). \quad (4.2)$$

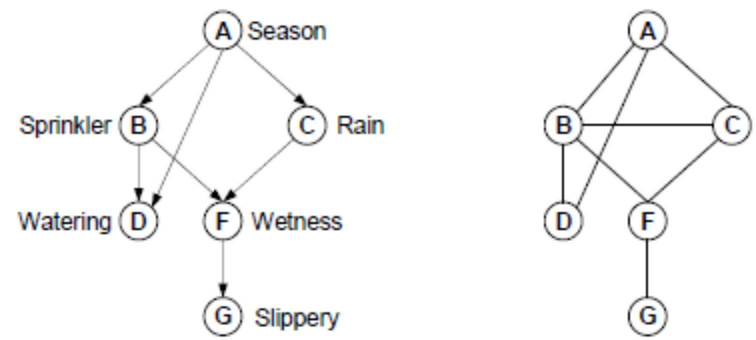
$$P(a, g=1) = P(a)\sum_c P(c|a)\sum_b P(b|a)\lambda_D(a,b)\sum_f P(f|b,c)\lambda_G(f) \quad (4.3)$$

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

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

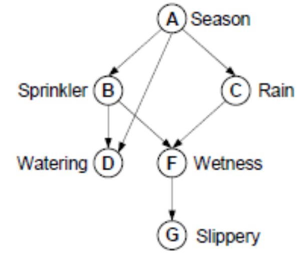
A Bayesian Network
Ordering: A,C,B,F,D,G



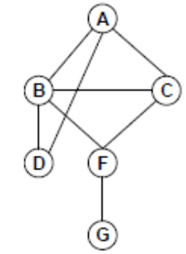


(a) Directed acyclic graph (b) Moral graph

A Different Ordering



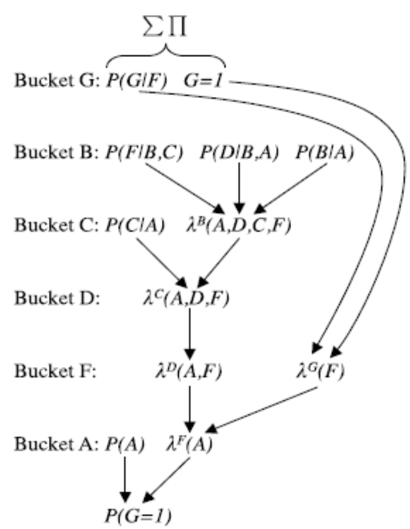
(a) Directed acyclic graph



(b) Moral graph

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

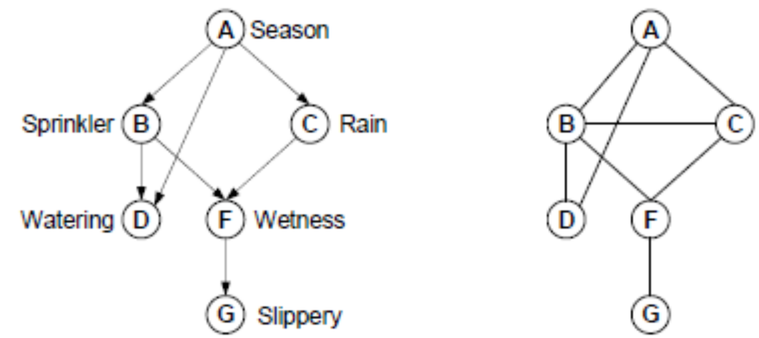
$$P(a, g = 1) = P(a) \sum_f \sum_d \sum_c P(c|a) \sum_b P(b|a)\ P(d|a,b) P(f|b,c) \sum_{g=1} P(g|f)$$
$$= P(a) \sum_f \lambda_G(f) \sum_d \sum_c P(c|a) \sum_b P(b|a)\ P(d|a,b) P(f|b,c)$$
$$= P(a) \sum_f \lambda_G(f) \sum_d \sum_c P(c|a) \lambda_B(a,d,c,f)$$
$$= P(a) \sum_f \lambda_g(f) \sum_d \lambda_C(a,d,f)$$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



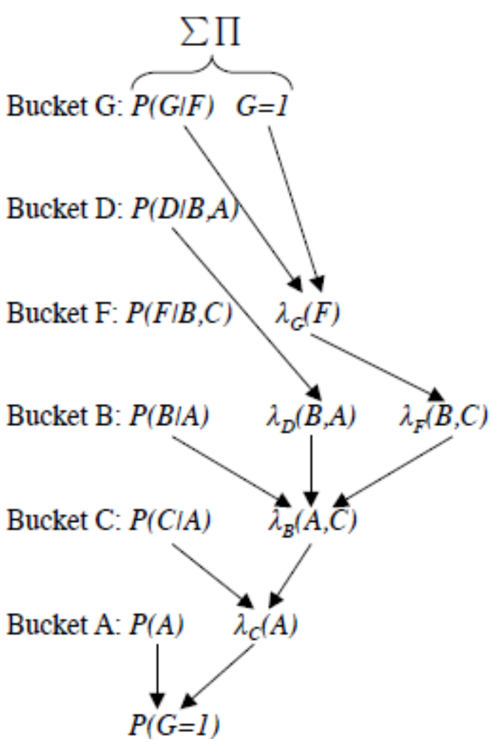
(a)

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

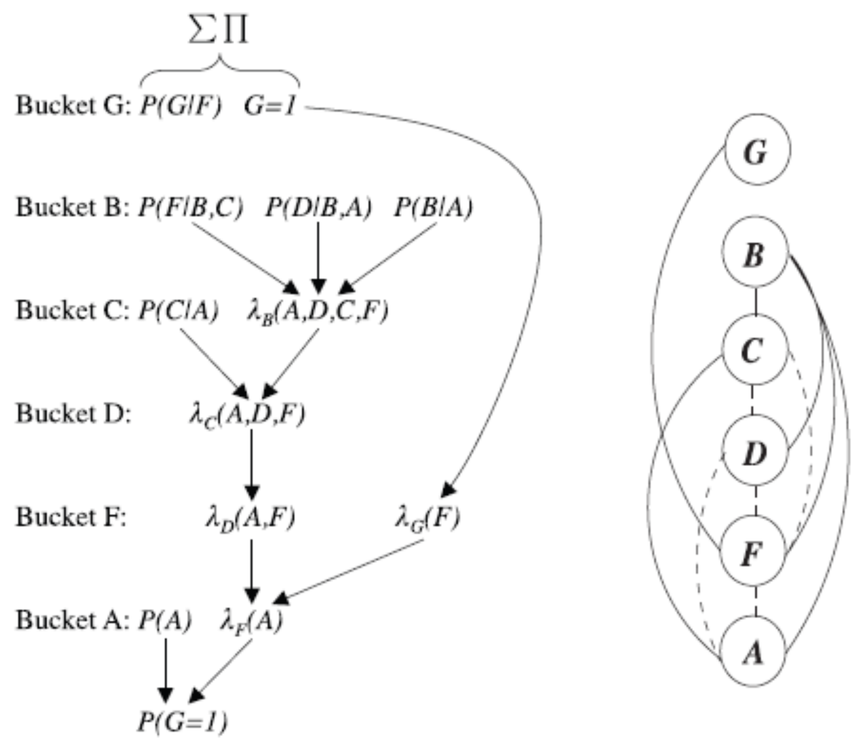
A Bayesian Network Processed Along 2 Orderings



(a) Directed acyclic graph (b) Moral graph



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



(a) (b)

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 $\mathbf{Y} = \mathbf{X} \setminus \{X\}$:

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

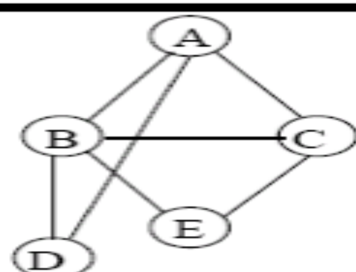
B	C	D	f_1
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

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

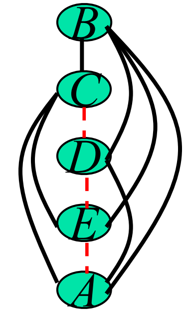
$bucket(B) = \quad P(e|b,c), P(d|a,b), P(b|a)$

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

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

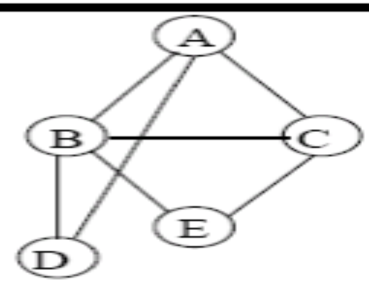
$bucket(E) = \quad e = 0 \quad || \ \lambda_D(a,c)$

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



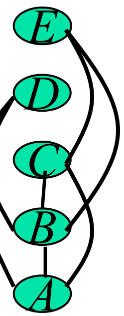
W*=4

Bucket Elimination and Induced Width



Ordering: a, b, c, d, e

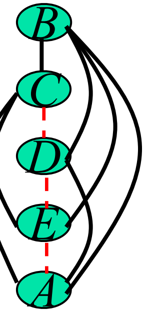
$bucket(E) = \quad P(e|b,c),\ e = 0$
$bucket(D) = \quad P(d|a,b)$
$bucket(C) = \quad P(c|a) \ \ || \ P(e=0|b,c)$
$bucket(B) = \quad P(b|a) \ \ || \ \ \lambda_D(a,b), \lambda_C(b,c)$
$bucket(A) = \quad P(a) \ \ || \ \ \lambda_B(a)$



W*=2

Ordering: a, e, d, c, b

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



W*=4

Algorithm BE-bel

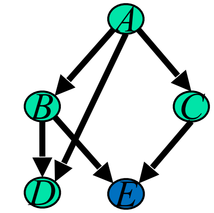
Input: A belief network $\mathcal{B} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}_G, \prod \rangle$, an ordering $d = (X_1, \ldots, X_n)$; evidence e
output: The belief $P(X_1|\mathrm{e})$ and probability of evidence $P(\mathrm{e})$

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 ψ_i the product of input functions in $bucket_i$.
2. **backward: for** $p \leftarrow n$ **downto** 1 **do**
3. **for** all the functions $\psi_{S_0}, \lambda_{S_1}, \ldots, \lambda_{S_j}$ 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 λ_p in bucket of the latest variable in scope(λ_p),
6. **return** (as a result of processing $bucket_1$):
   $P(\mathrm{e}) = \alpha = \sum_{X_1} \psi_1 \cdot \prod_{\lambda \in bucket_1} \lambda$
   $P(X_1|\mathrm{e}) = \frac{1}{\alpha} \psi_1 \cdot \prod_{\lambda \in bucket_1} \lambda$

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

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

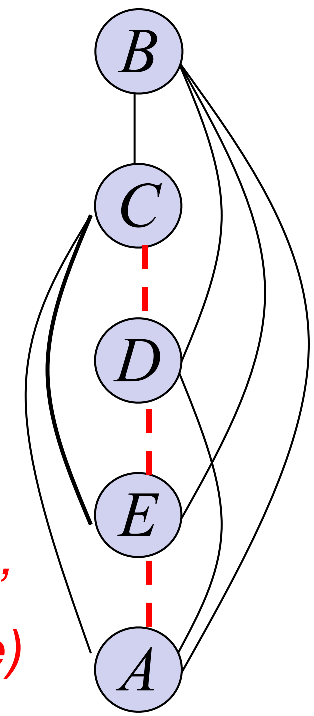
bucket D: $\lambda_{C\to D}(A,d,e)$

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

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

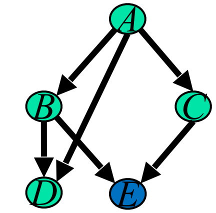
$p(E=0)$

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]

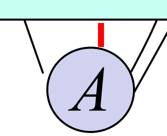
$$p(A|E=0) = \alpha \sum_{e,d,c,b} p(A)\, p(b|A)\, p(c|A)\, p(d|A,b)\, p(e|b,c)\, \mathbb{1}[e=0]$$

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

Time and space exponential in the induced-width / treewidth

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

(max clique size)



$p(E=0)$

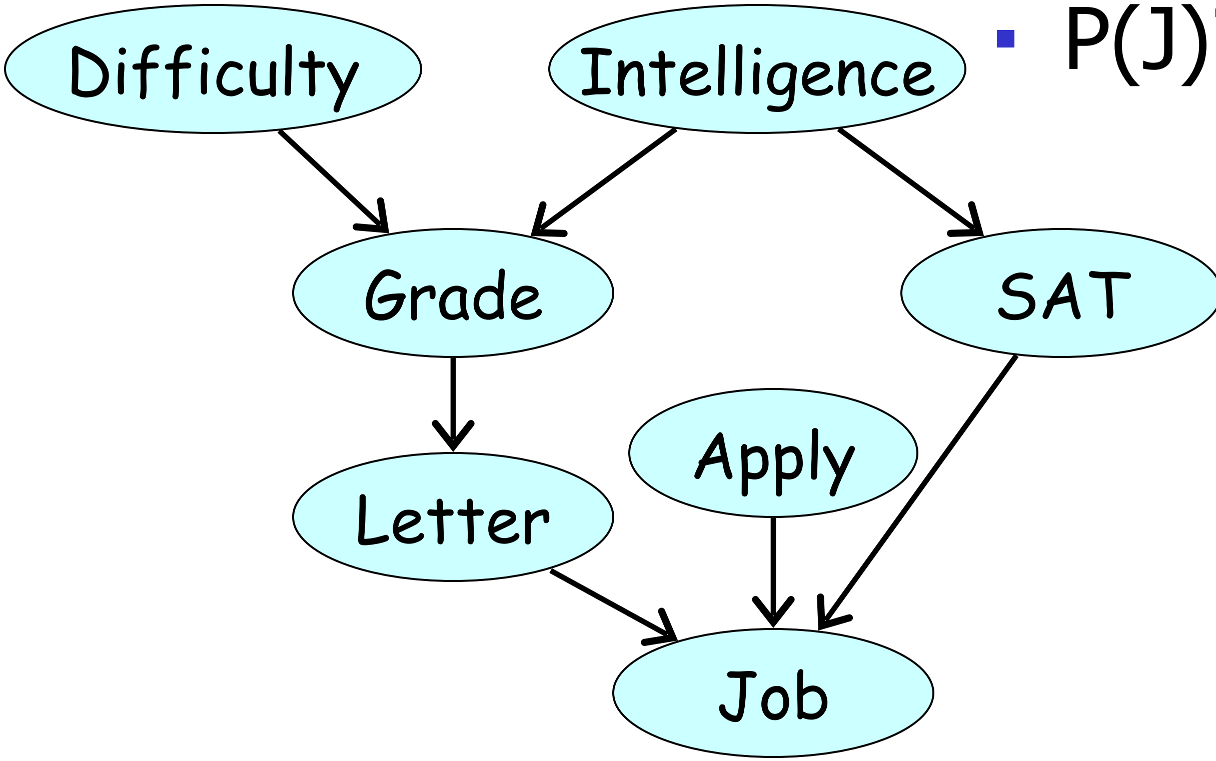
$$p(A|E=0) = p(A, E=0) \,/\, p(E=0)$$

Outline

- The do calculus (review)
- Bayesian networks, representation and inference
- Class project

Information on the project

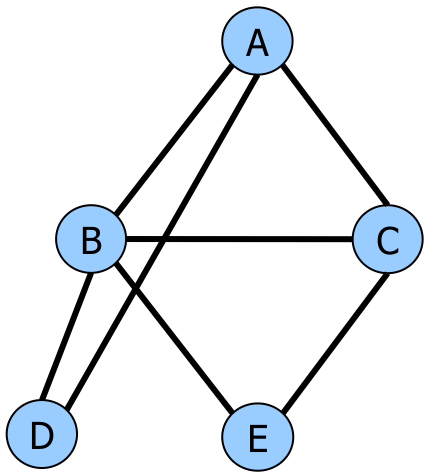
Student Network Example



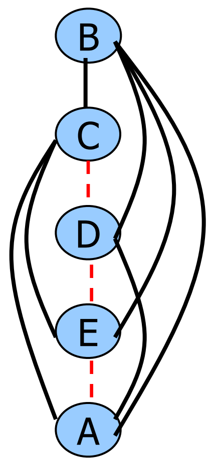
- P(J)?

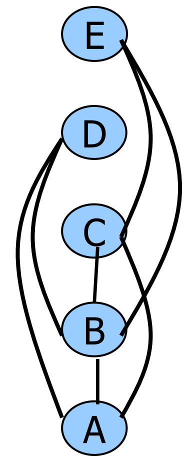
Induced Width (continued)

The effect of the ordering:



Primal (moraal) graph



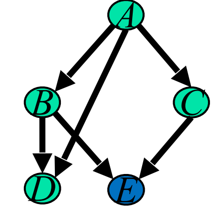


Inference for probabilistic networks

- 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



← *Elimination operator*

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

bucket C: $P(c|a)$

bucket D:

bucket E: $e=0$

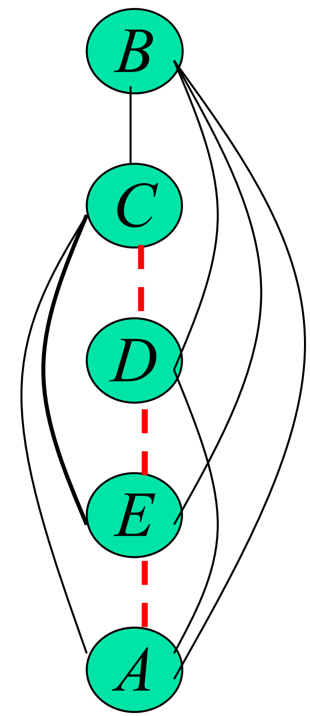
bucket A: $P(a)$

$P(e=0)$

$P(a|e=0)$

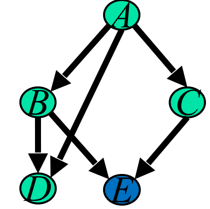
W=4*
"induced width"
(max clique size)

B=1



The impact of evidence?

Algorithm BE-bel



P(A|E=0,B=1)?

Elimination operator

bucket B: $P(b|a)$ $P(d|b,a)$ $P(e|b,c)$ B=1

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

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

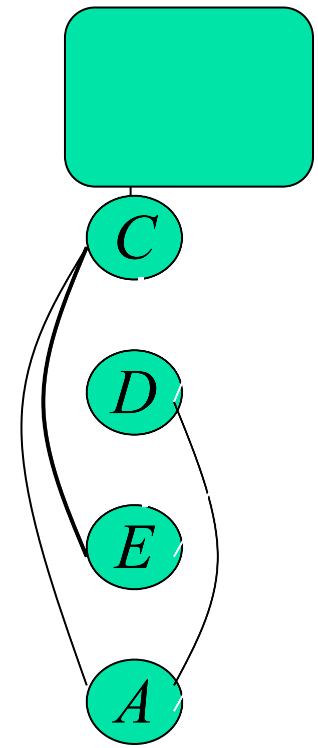
bucket E: $e=0$

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

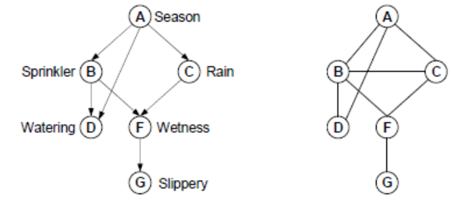
$P(e=0)$

$P(a|e=0)$

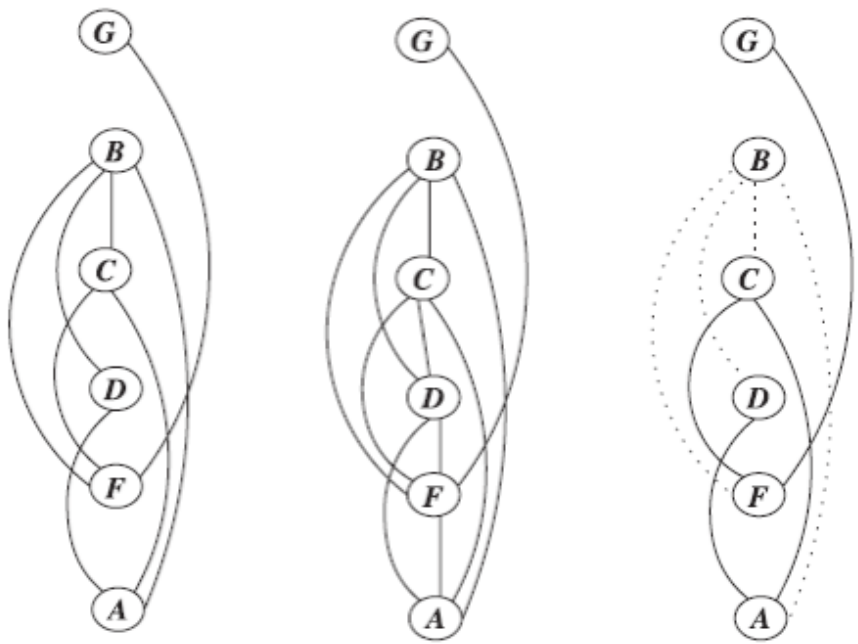
$$P(a|e=0)=\frac{P(a,e=0)}{P(e=0)}$$



The impact of observations



(a) Directed acyclic graph (b) Moral graph



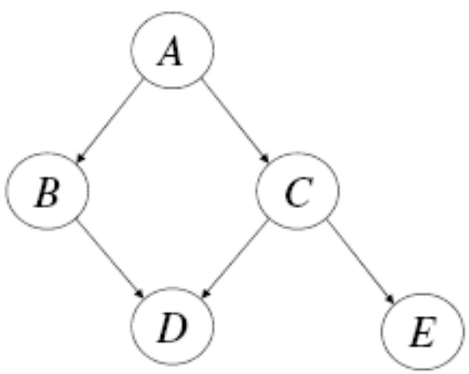
(a) (b) (c)

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

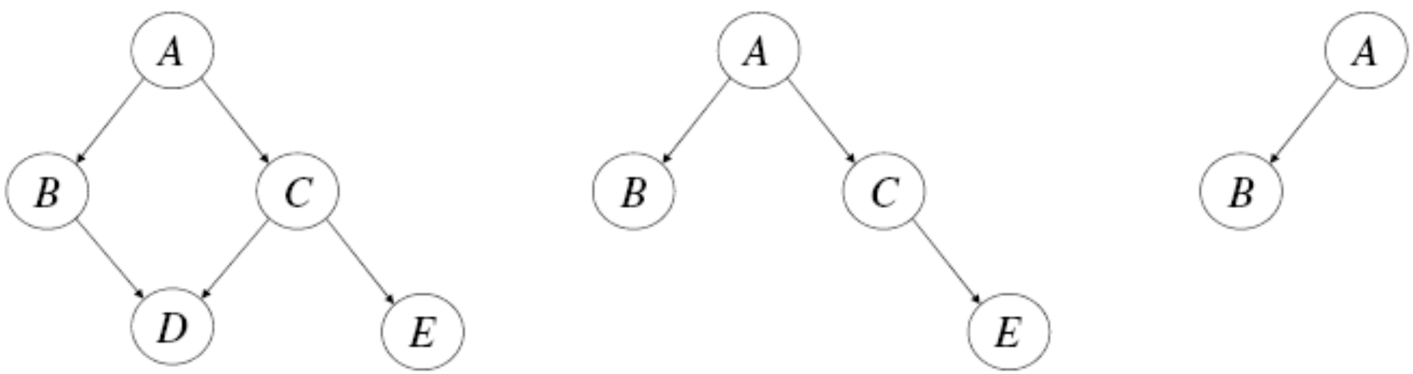
one can remove any leaf node (with its CPT) from the network as long as it does not belong to variables $\mathbf{Q} \cup \mathbf{E}$, yet not affect the ability of the network to answer the query correctly.

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

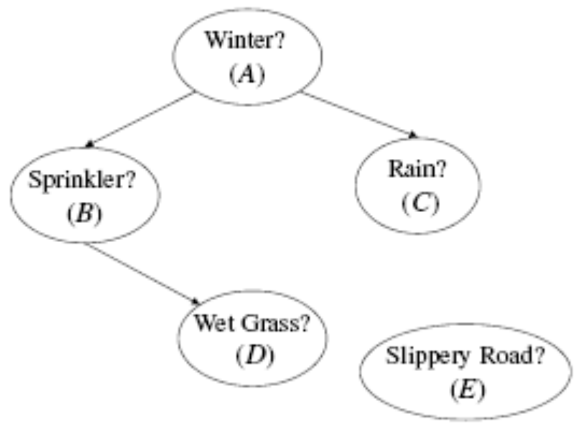


network structure — joint on B, E — joint on B

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	Θ_A
true	.6
false	.4

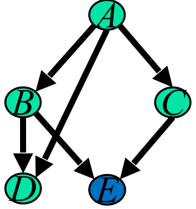
B	D	$\sum_C \Theta_{D\|BC}^{C=\text{false}}$
true	true	.9
true	false	.1
false	true	0
false	false	1

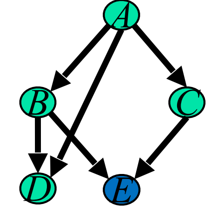
E	$\sum_C \Theta_{E\|C}^{C=\text{false}}$
true	0
false	1

Evidence **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


A
B
C
D
E



Elimination operator

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

bucket C: $P(c|a)$

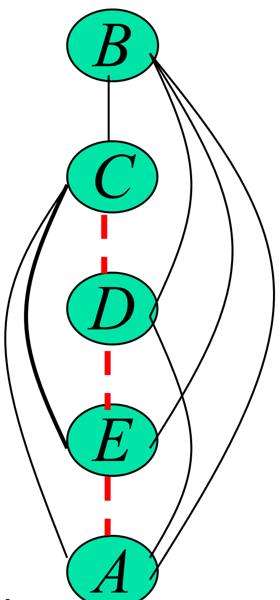
bucket D:

bucket E: $e=0$

bucket A: $P(a)$

MPE

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



Generating the MPE-tuple

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

C: $P(c|a)$

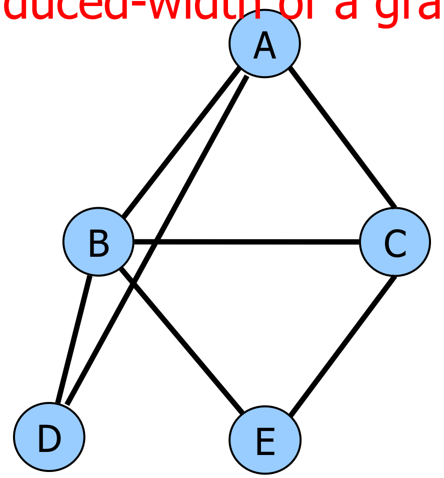
D:

E: $e=0$

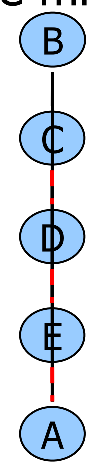
A: $P(a)$

Induced Width

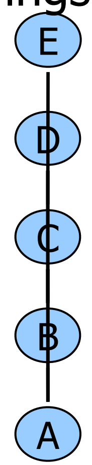
- Width is the max number of parents in the ordered graph
- Induced-width is the width of the induced ordered graph: recursively connecting parents going from last node to first.
- Induced-width w*(d) is the max induced-width over all nodes in ordering d
- Induced-width of a graph, w* is the min w*(d)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



primal graph



$w^*(d_1) = 4$



$w^*(d_2) = 2$

Complexity of Bucket Elimination

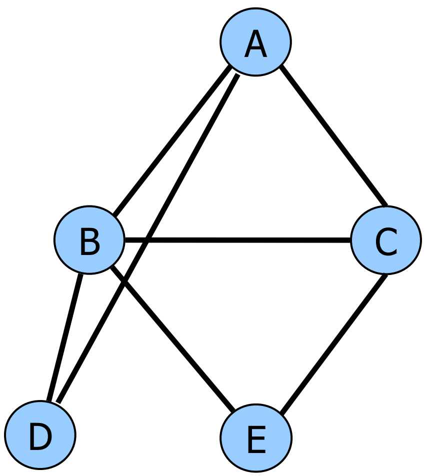
Bucket-Elimination is **time** and **space**

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

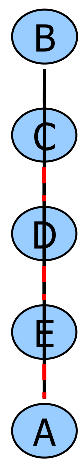
w_d^*: the induced width of the primal graph along ordering d

r = number of functions

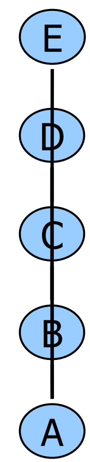
The effect of the ordering:



primal graph



$w^*(d_1) = 4$

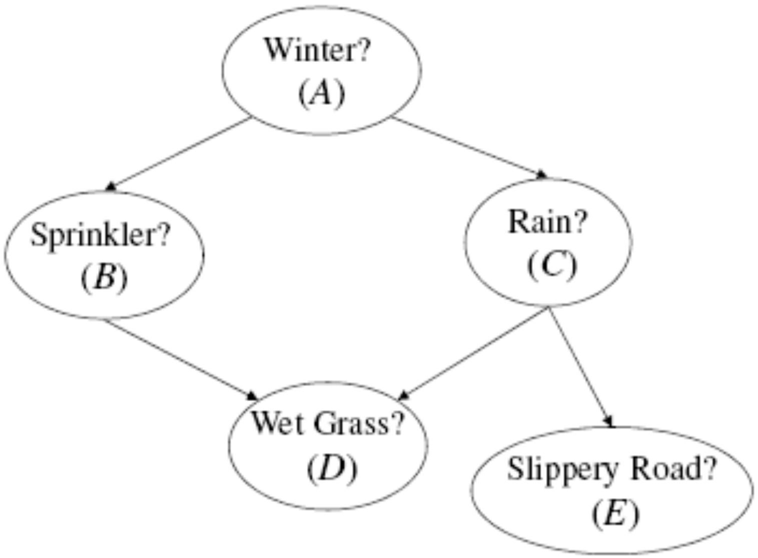


$w^*(d_2) = 2$

Finding smallest induced-width is hard!

A Bayesian Network

Example with mpe?



A	Θ_A
true	.6
false	.4

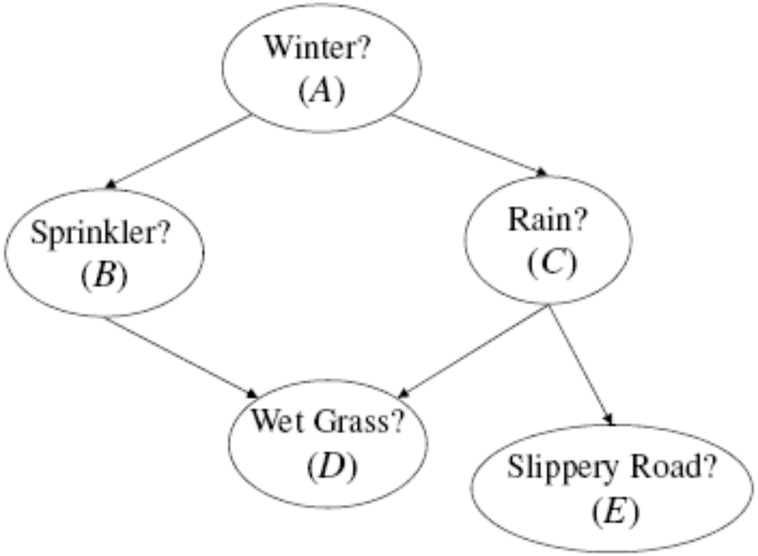
A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

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

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

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

Try to compute MPE when E=0



A	Θ_A
true	.6
false	.4

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

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

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

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

Cost Networks

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

becomes

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

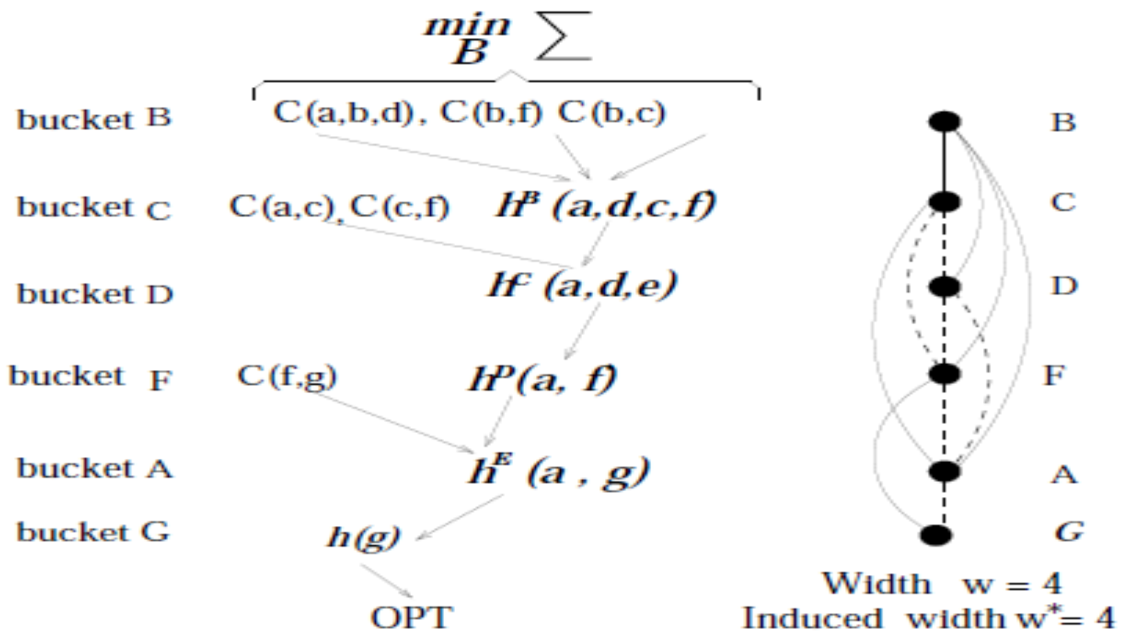


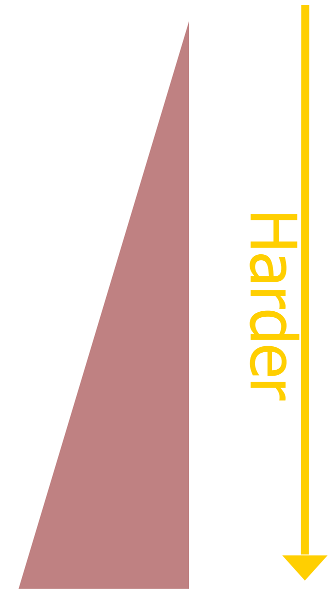
Figure 5.12: Schematic execution of BE-Opt

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

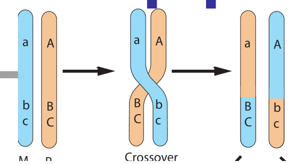
Max-Inference	$f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Sum-Inference	$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Mixed-Inference	$f(\mathbf{x}_M^*) = \max_{\mathbf{x}_\mathrm{M}} \sum_{\mathbf{x}_S} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$

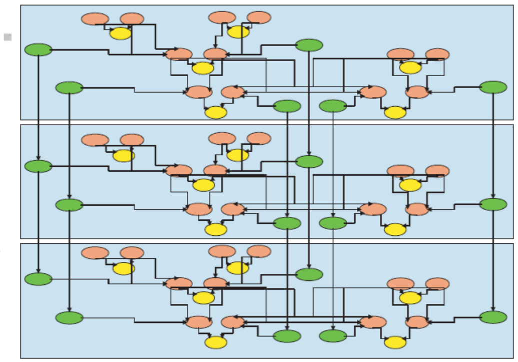


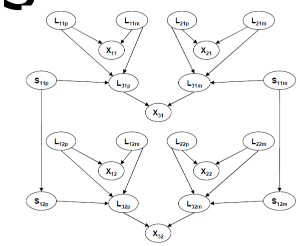
- **NP-hard**: exponentially many terms

Example for MMAP Applications

- Haplotype in Family pedigrees







- Coding networks

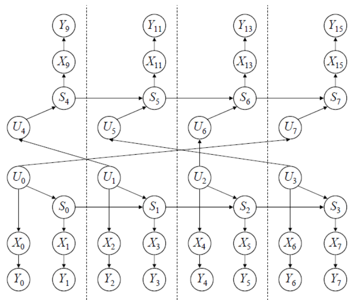
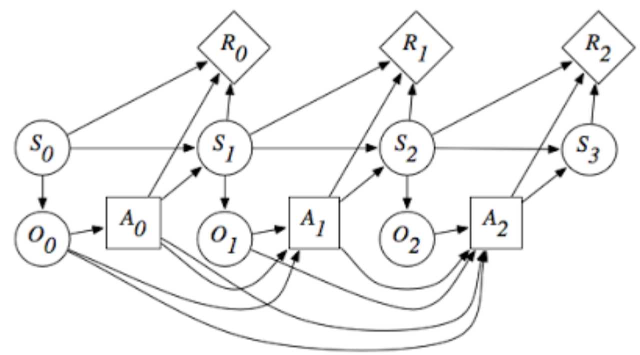
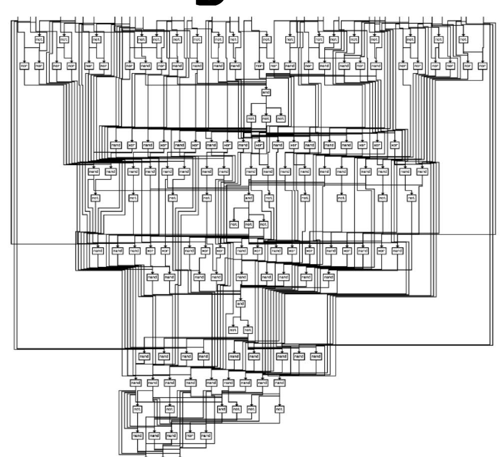


Figure 5.24: A Bayesian network for a turbo code.

- Probabilistic planning

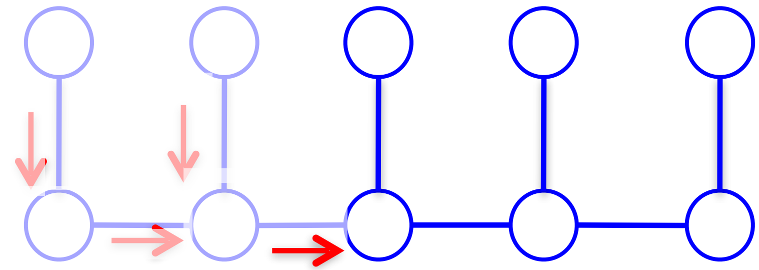


- Diagnosis



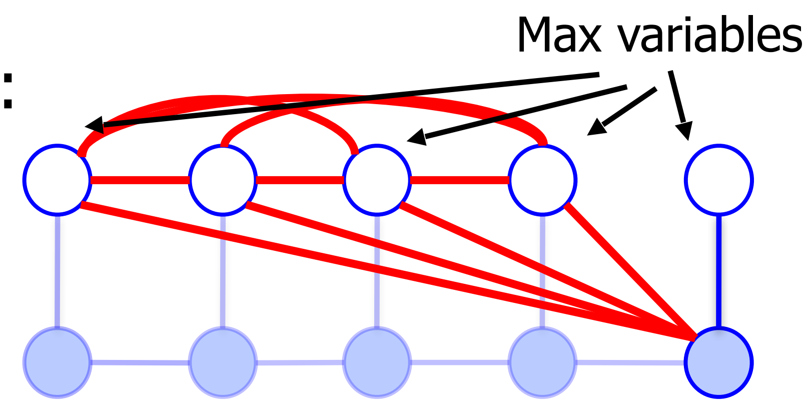
Marginal MAP is Not Easy on Trees

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



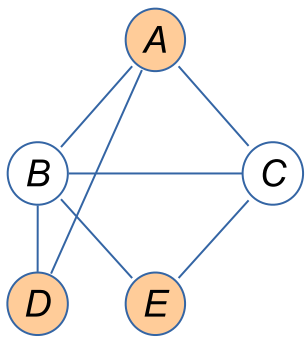
- Marginal MAP
  - Operations do not commute:
  - Sum must be done first!

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



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

B: $f(A,B)\, f(B,C)\, f(B,D) f(B,E)$

Σ_B

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

Σ_C

MAX

D: $\lambda^C(A,D,E)\, f(A,D)$

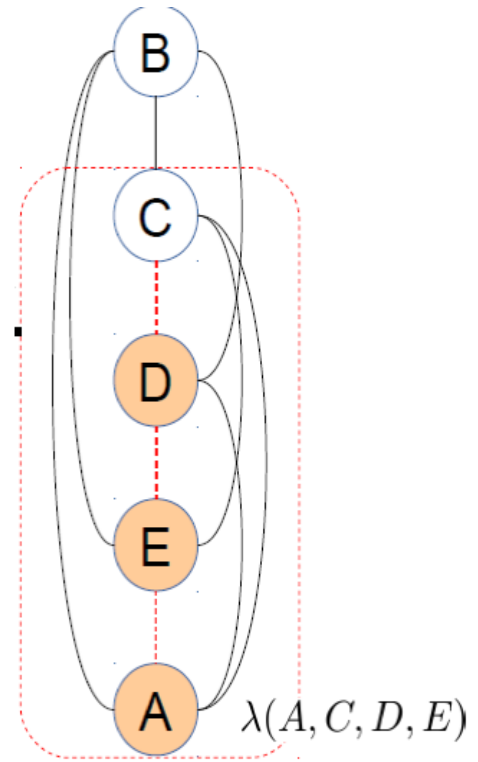
max_D

E: $\lambda^D(A,E)$

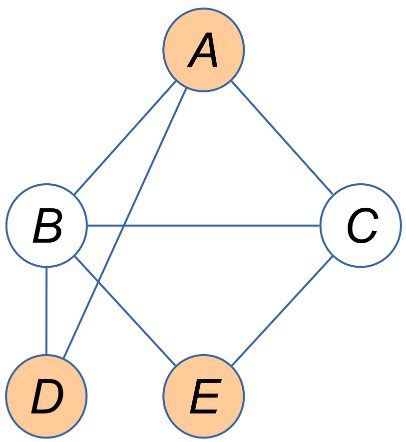
max_E

A: $\lambda^E(A)$

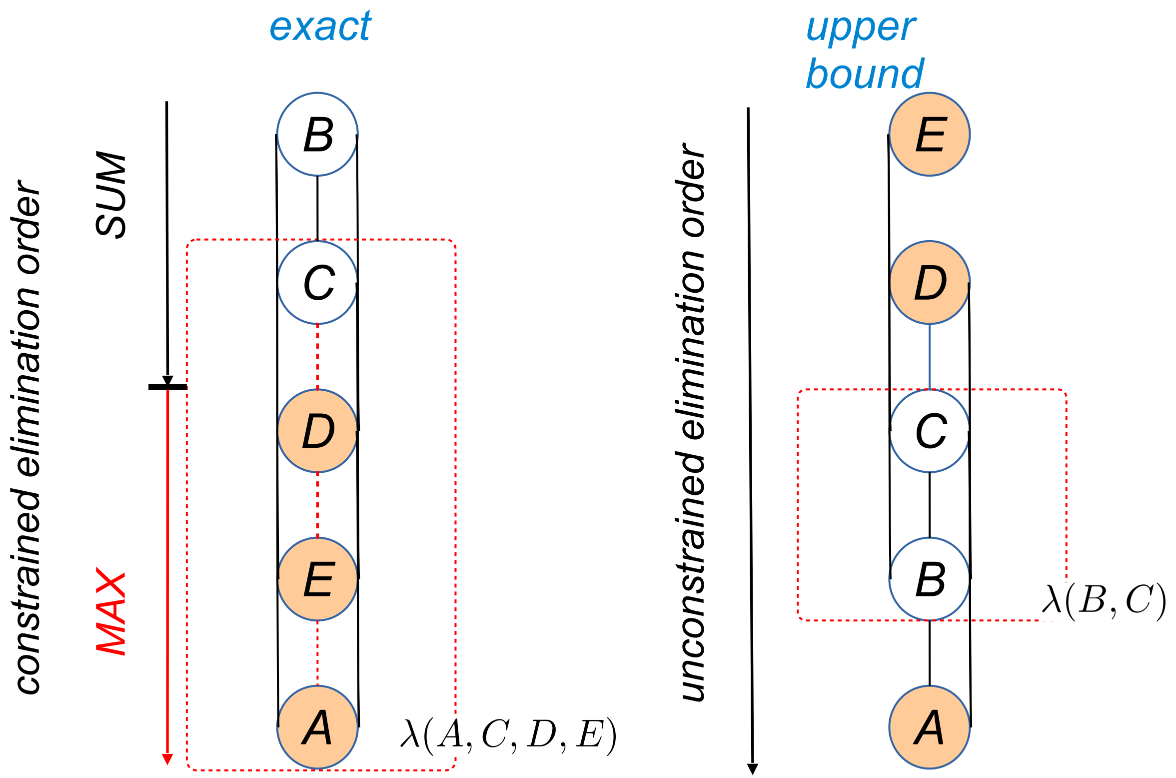
MAP is the marginal MAP value*



Why is MMAP harder?



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



$w^* = 4$ $w^* = 2$

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

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

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

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

where,

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

Inference for probabilistic networks

- Bucket elimination
  - Belief-updating, P(e), partition function
  - Marginals, probability of evidence
  - The impact of evidence
  - for MPE (→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)

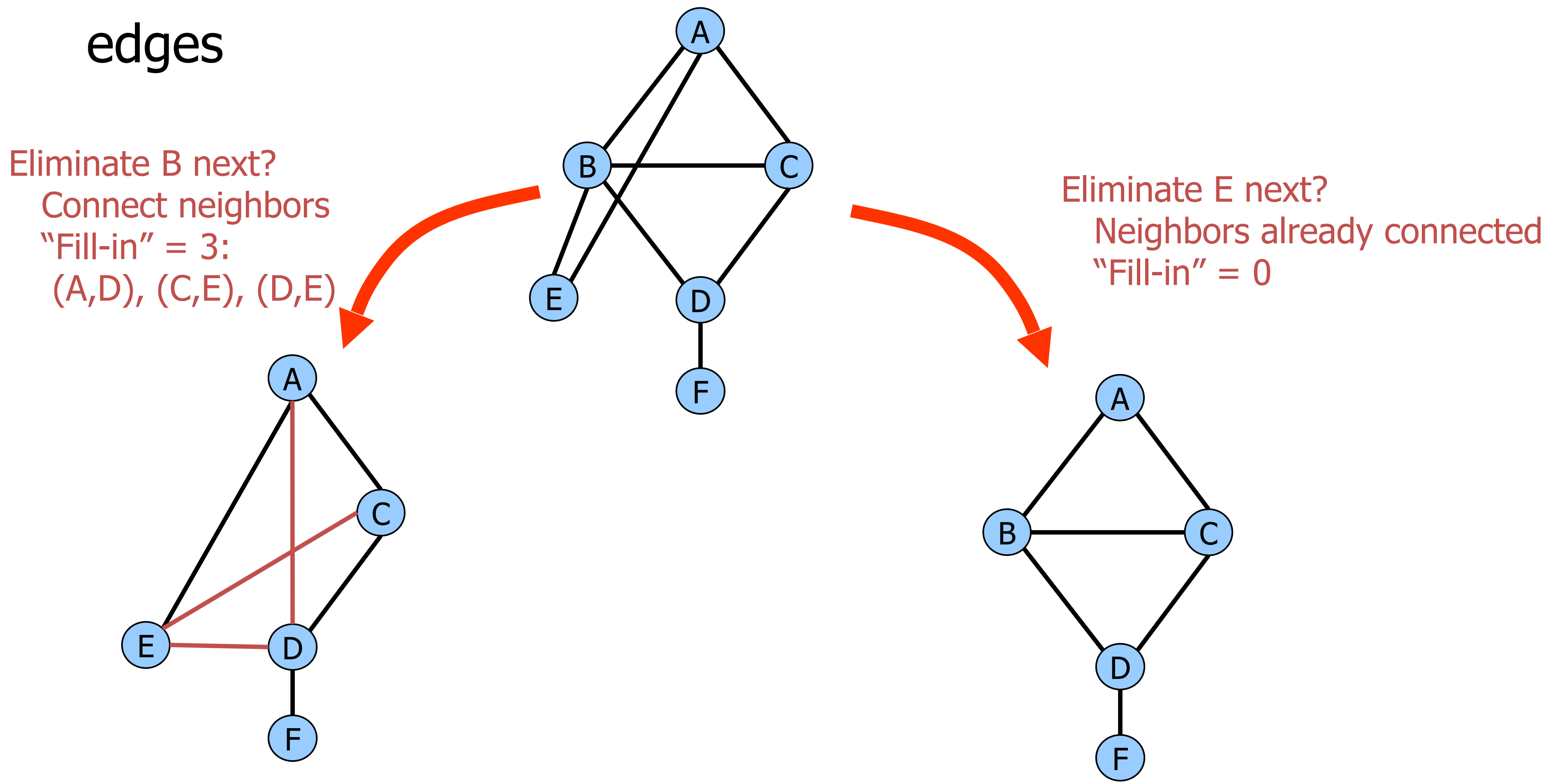
Greedy Orderings Heuristics

- 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

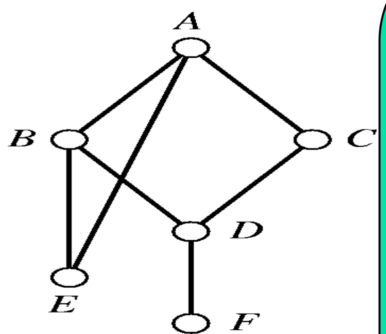
Complexity? $O(n^3)$

Min-Fill Heuristic

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

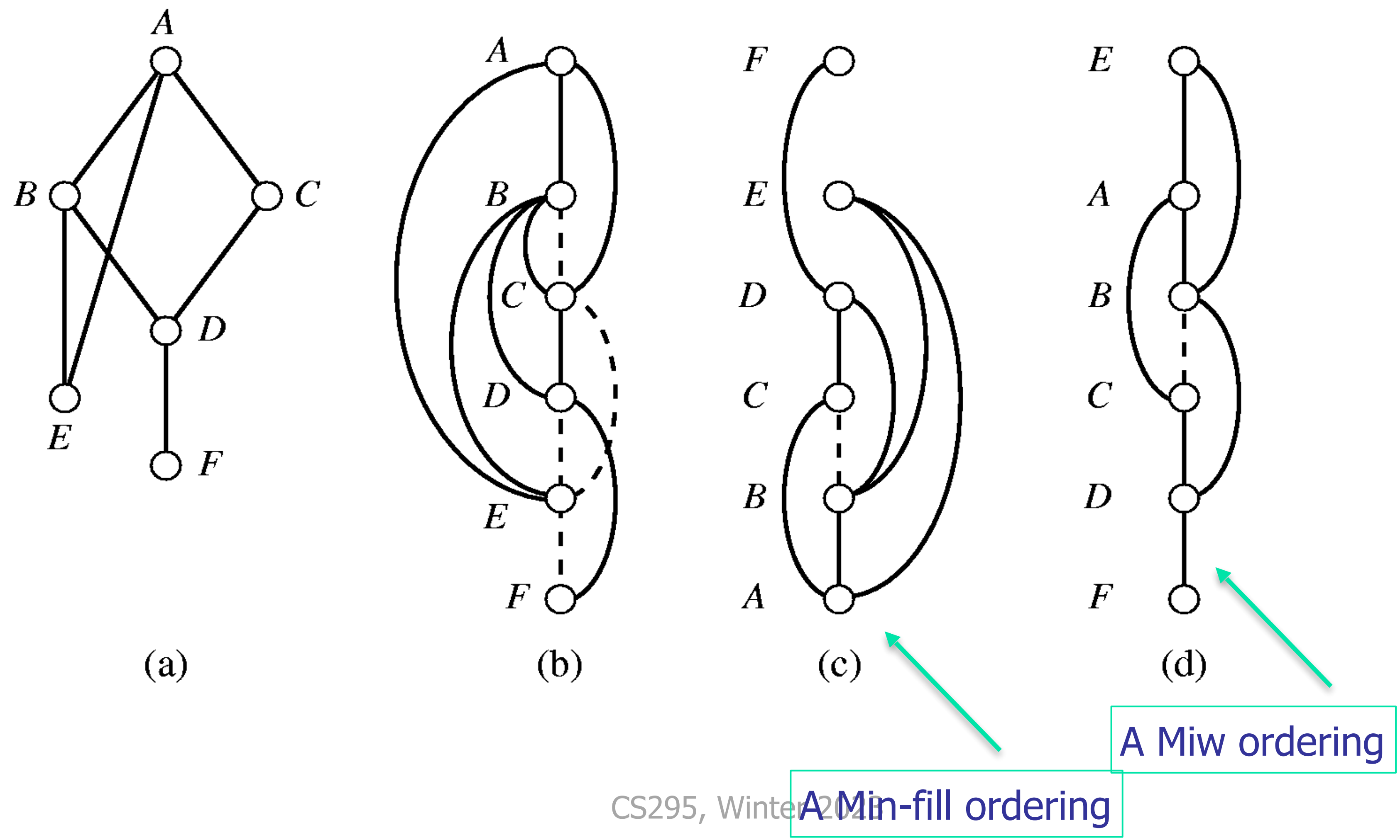


Example



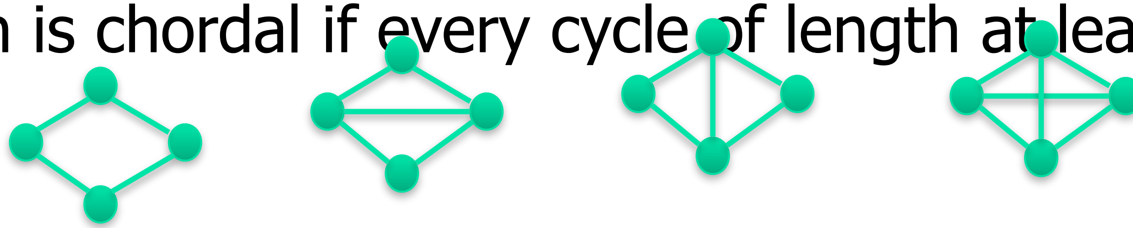
(a)

Different Induced-Graphs



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

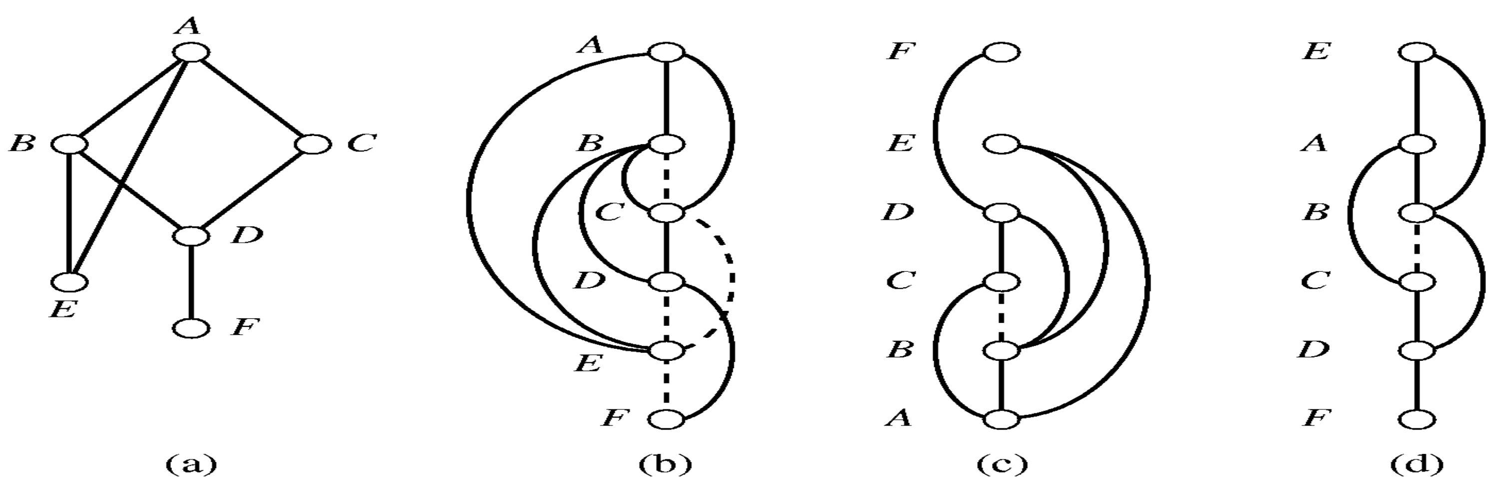
MAX-CARDINALITY (MC)

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. $r \leftarrow$ 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]