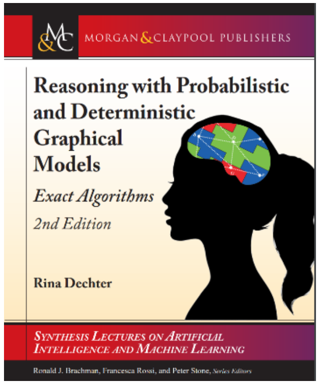
Slides Set 5:

Exact Inference Algorithms Bucket-elimination

Rina Dechter

(Dechter chapter 4, Darwiche chapter 6)



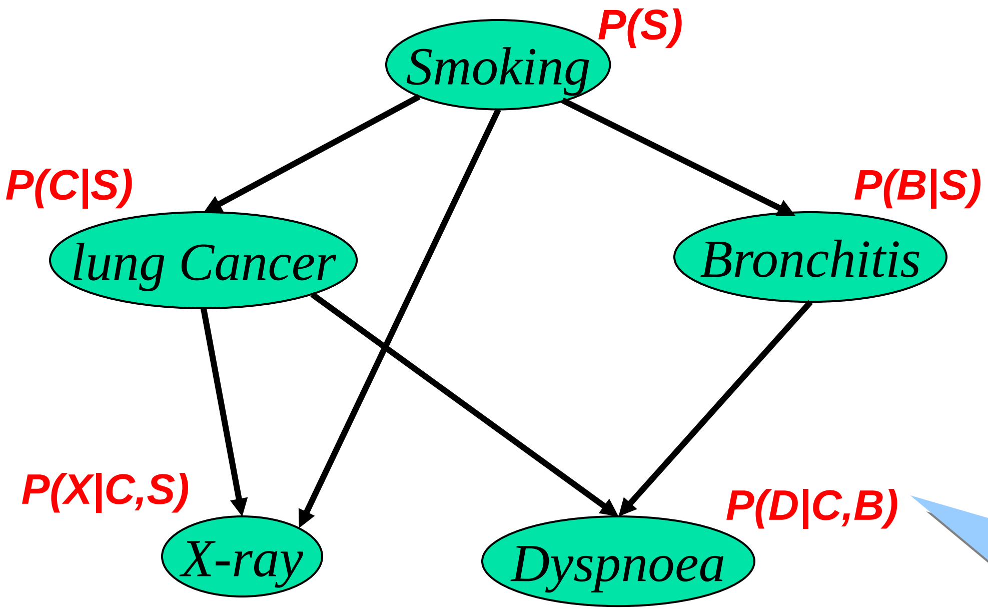
Inference for probabilistic networks

- 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)
  - Influence diagrams ?
- 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)



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

CPD:

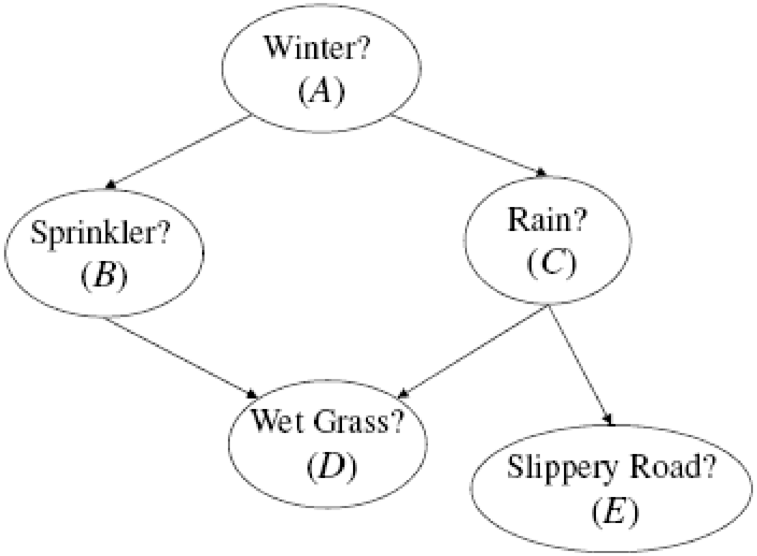
C	B	P(D|C,B)	
0	0	0.1	0.9
0	1	0.7	0.3
1	0	0.8	0.2
1	1	0.9	0.1

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

Belief Updating:

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

A Bayesian Network



A	Θ_A
true	.6
false	.4

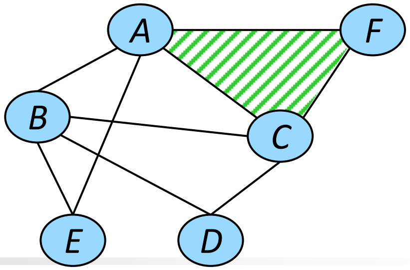
A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

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

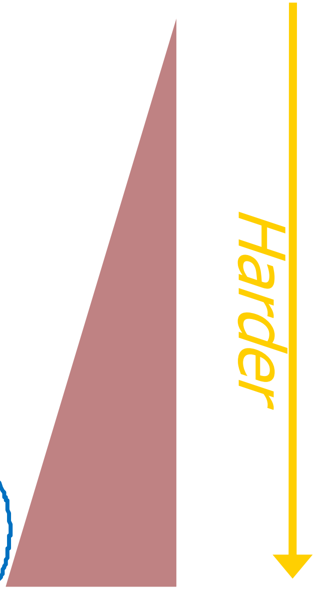
B	C	D	$\Theta_{D\|BC}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

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

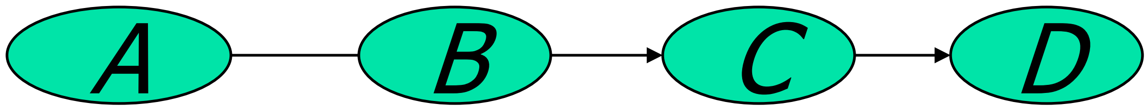
- Each SAT formula can be mapped into a belief updating query in a Bayesian network

- Example

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

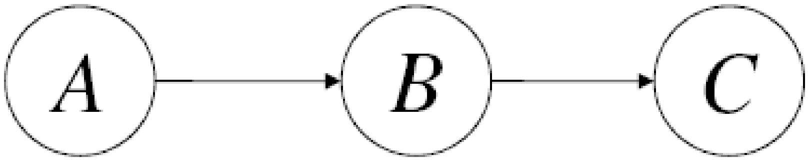
A Simple Network

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	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

Elimination as a Basis for Inference

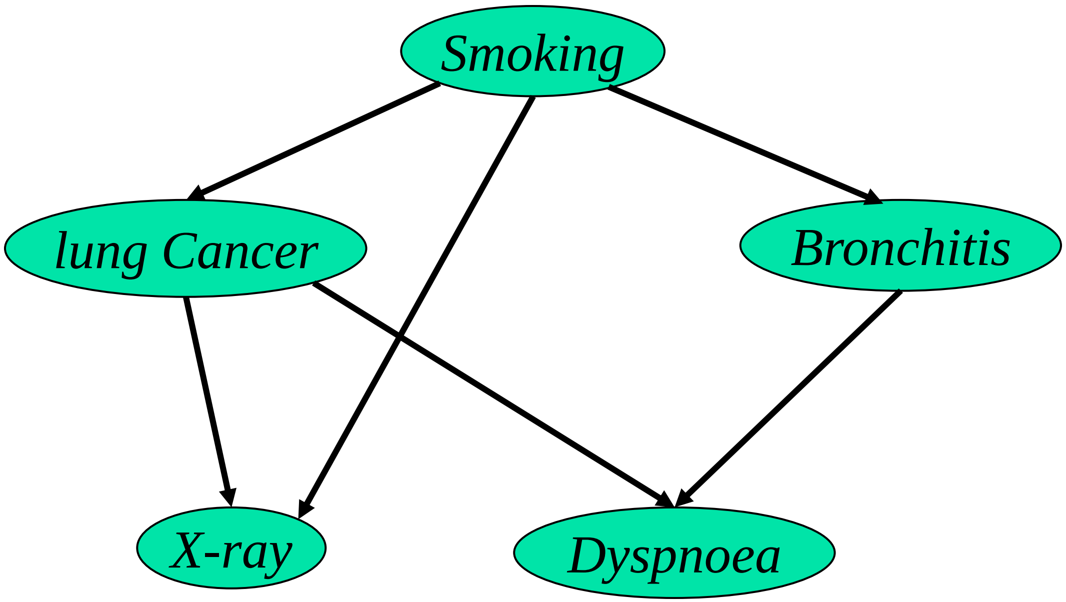
- We now have two factors, $\sum_A \Theta_A \Theta_{B|A}$ and $\Theta_{C|B}$, and we want to eliminate variable B
- Since B appears in both factors, we must multiply them first and then sum out B from the result.
- Multiplying:

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:

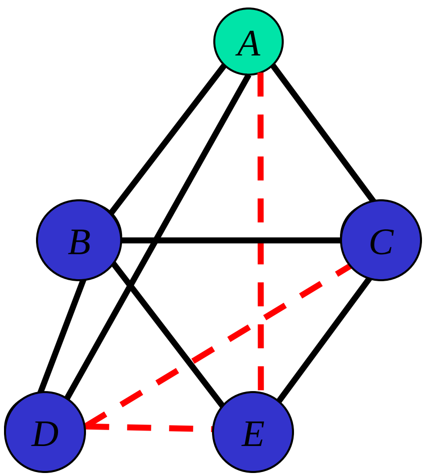
C	$\sum_B \Theta_{C|B} \sum_A \Theta_A \Theta_{B|A}$
true	.376
false	.624

Belief Updating



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

Belief updating: P(X|evidence)=?



"Moral" graph

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

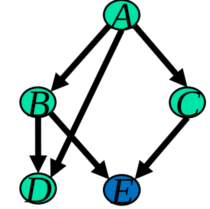
$$\sum_{e=0,d,c,b} P(a)\underbrace{P(b|a)}P(c|a)\underbrace{P(d|b,a)P(e|b,c)}=$$

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

Variable Elimination

Bucket elimination

Algorithm *BE-bel* (Dechter 1996)



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

$\sum_b \prod$ ← *Elimination operator*

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

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

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

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

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

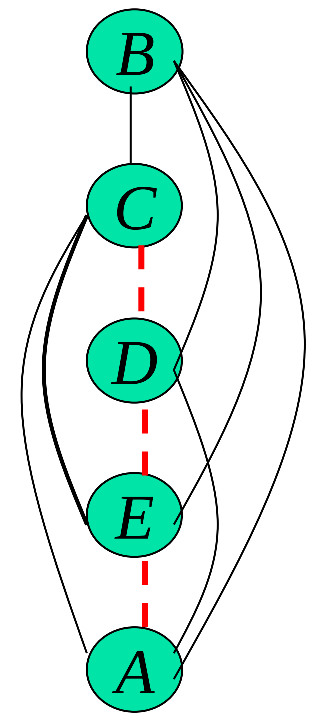
P(e=0)

P(a,e=0)

W=4*

"induced width"

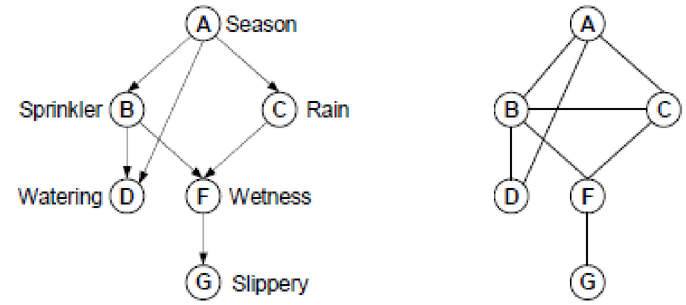
(max clique size)



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

A Bayesian Network

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



(a) Directed acyclic graph (b) Moral graph

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

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

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

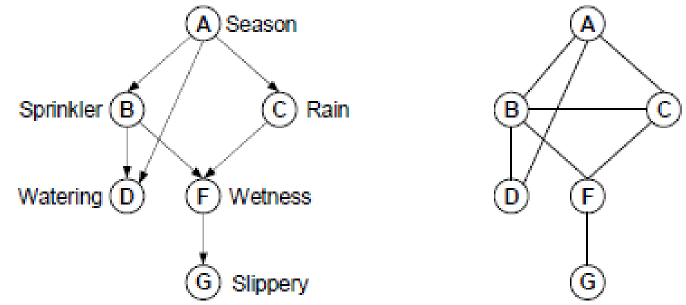
$$P(a, g = 1) = P(a)\sum_{c} P(c|a)\sum_{b} P(b|a)\lambda_D(a, b)\sum_{f} P(f|b, c)\lambda_G(f) \qquad (4.3)$$

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

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

A Bayesian Network

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



(a) Directed acyclic graph (b) Moral graph

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

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

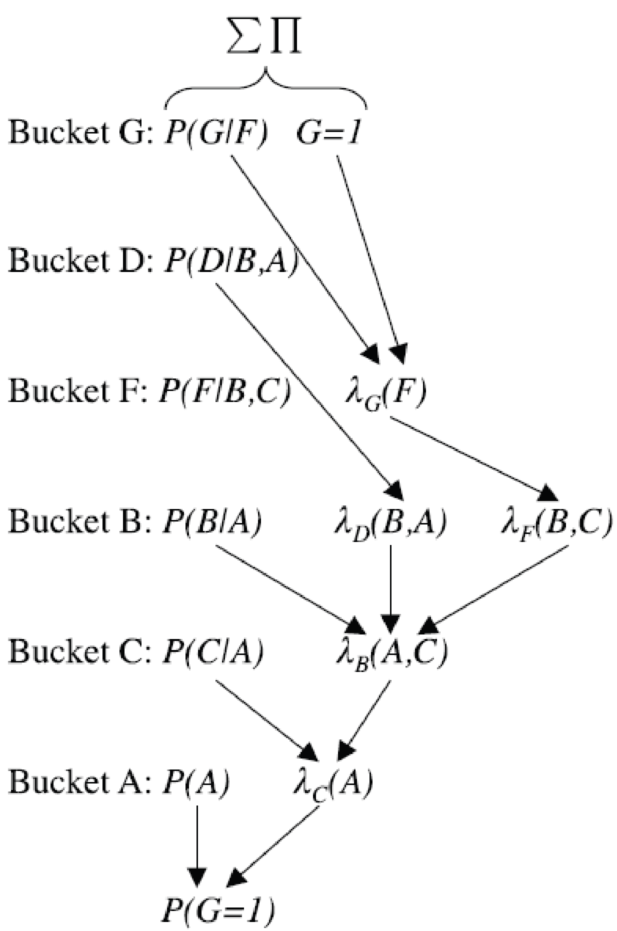
$$P(a, g=1) = P(a)\sum_{c} P(c|a)\sum_{b} P(b|a)\sum_{f} P(f|b,c)\lambda_G(f)\sum_{d} P(d|b,a). \quad (4.2)$$

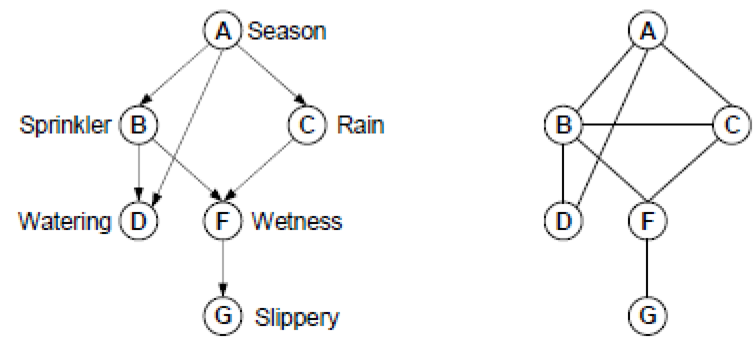
$$P(a, g=1) = P(a)\sum_{c} P(c|a)\sum_{b} P(b|a)\lambda_D(a,b)\sum_{f} P(f|b,c)\lambda_G(f) \quad (4.3)$$

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

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

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

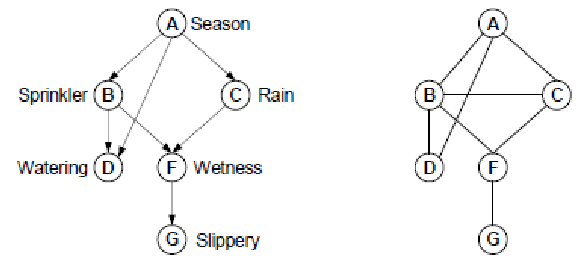




(a) Directed acyclic graph

(b) Moral graph

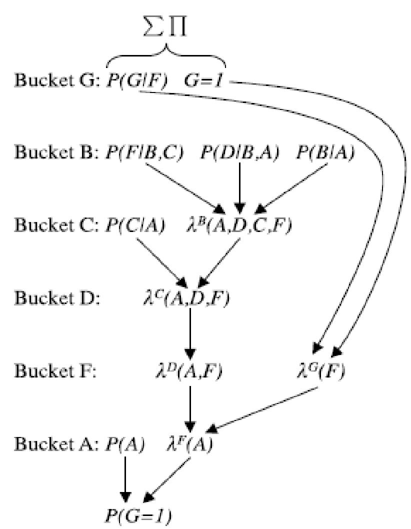
A Different Ordering



(a) Directed acyclic graph (b) Moral graph

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

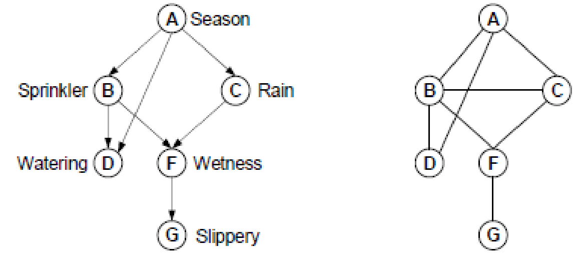
$$P(a,g=1) = P(a)\sum_f \sum_d \sum_c P(c|a)\sum_b P(b|a)\ P(d|a,b)P(f|b,c)\sum_{g=1} P(g|f)$$
$$= P(a)\sum_f \lambda_G(f)\sum_d \sum_c P(c|a)\sum_b P(b|a)\ P(d|a,b)P(f|b,c)$$
$$= P(a)\sum_f \lambda_G(f)\sum_d \sum_c P(c|a)\lambda_B(a,d,c,f)$$
$$= P(a)\sum_f \lambda_g(f)\sum_d \lambda_C(a,d,f)$$
$$=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



(a)

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

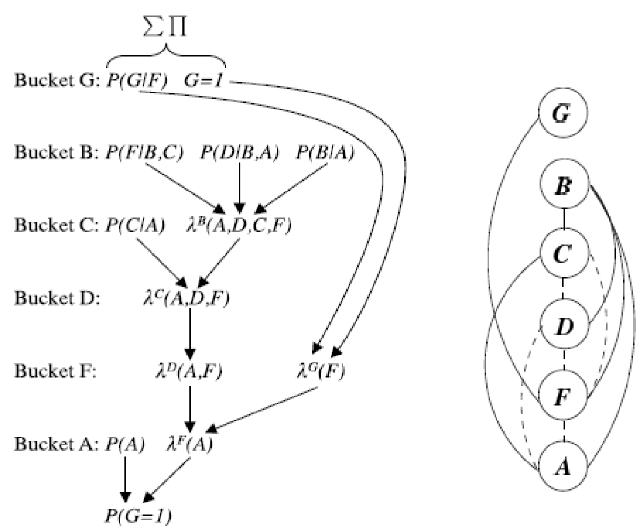
A Different Ordering



(a) Directed acyclic graph (b) Moral graph

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

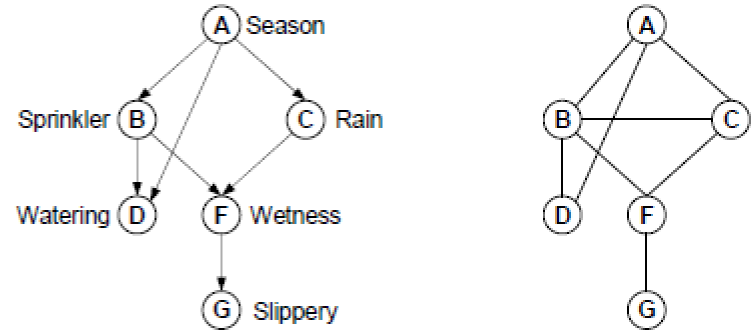
$$P(a,g=1) = P(a)\sum_f \sum_d \sum_c P(c|a)\sum_b P(b|a)\ P(d|a,b)P(f|b,c)\sum_{g=1} P(g|f)$$
$$= P(a)\sum_f \lambda_G(f)\sum_d \sum_c P(c|a)\sum_b P(b|a)\ P(d|a,b)P(f|b,c)$$
$$= P(a)\sum_f \lambda_G(f)\sum_d \sum_c P(c|a)\lambda_B(a,d,c,f)$$
$$= P(a)\sum_f \lambda_g(f)\sum_d \lambda_C(a,d,f)$$
$$=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



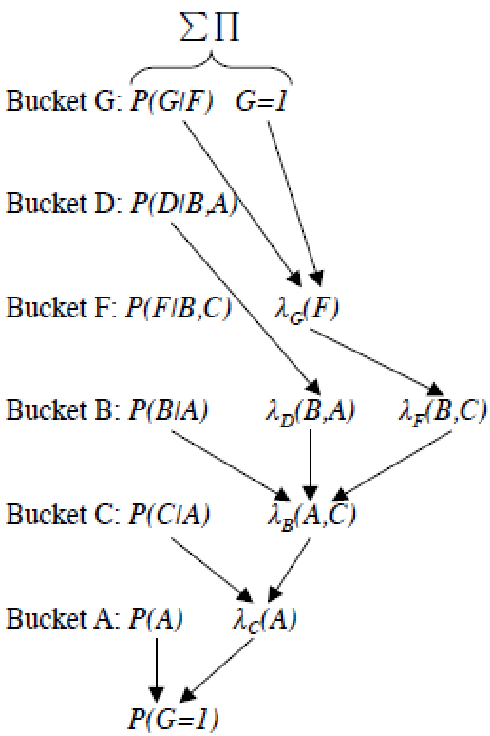
(a) **(b)**

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

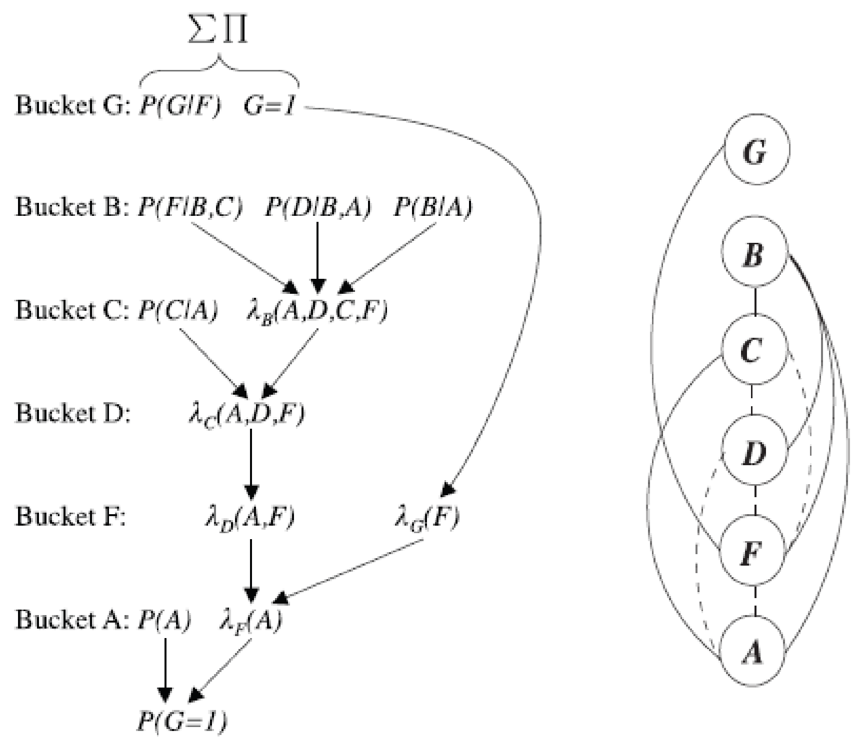
A Bayesian Network Processed Along 2 Orderings



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

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

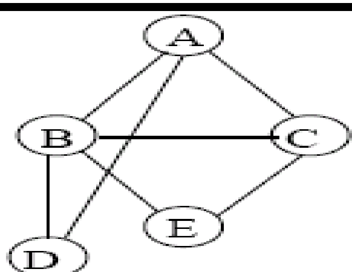
B	C	D	f_1
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

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

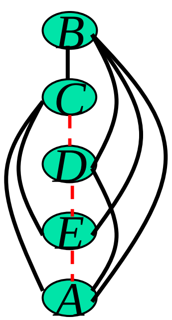
Thanks to Darwiche

Bucket Elimination and Induced Width



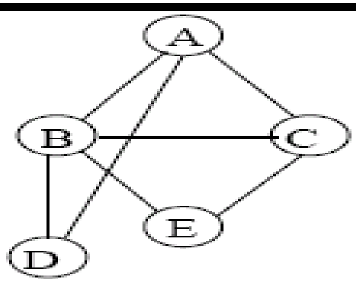
Ordering: a, e, d, c, b

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



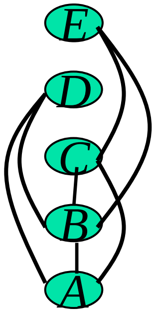
W*=4

Bucket Elimination and Induced Width



Ordering: a, b, c, d, e

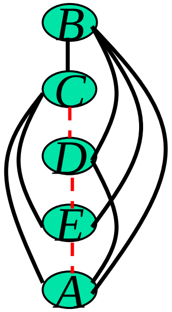
$bucket(E) = \quad P(e|b,c), \; e = 0$
$bucket(D) = \quad P(d|a,b)$
$bucket(C) = \quad P(c|a) \;\; || \;\; P(e=0|b,c)$
$bucket(B) = \quad P(b|a) \;\; || \;\; \lambda_D(a,b), \lambda_C(b,c)$
$bucket(A) = \quad P(a) \;\; || \;\; \lambda_B(a)$



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

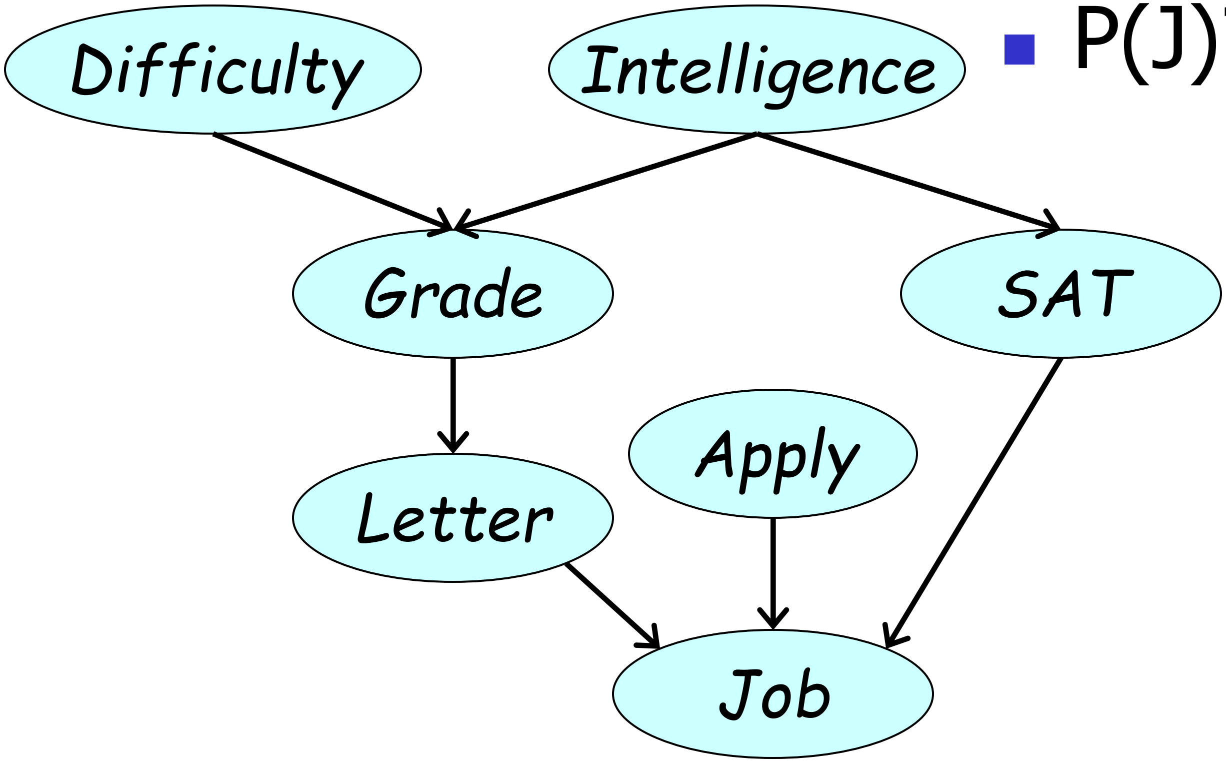
ALGORITHM BE-BEL

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 $P(X_1|e)$ and probability of evidence $P(e)$

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

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

Student Network Example

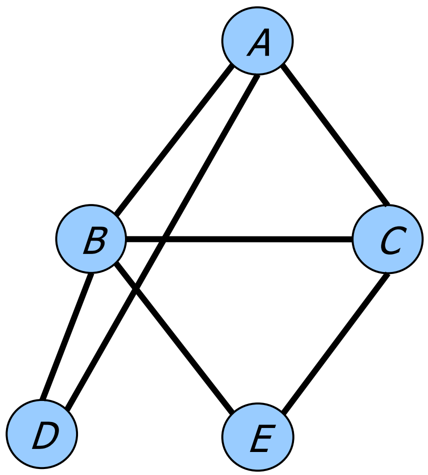


- P(J)?

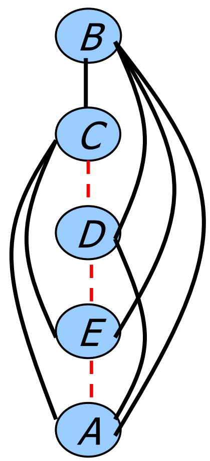
Induced Width (continued)

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

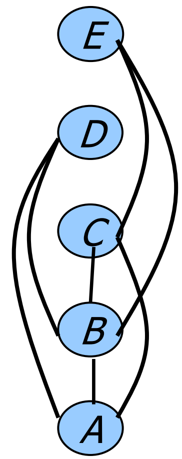
The effect of the ordering:



Primal (moraal) graph



$w^*(d_1) = 4$



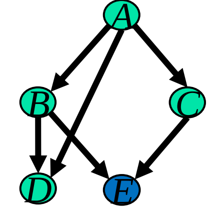
$w^*(d_2) = 2$

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*



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

$\sum_b \prod \longleftarrow$ *Elimination operator*

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

B=1

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

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

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

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

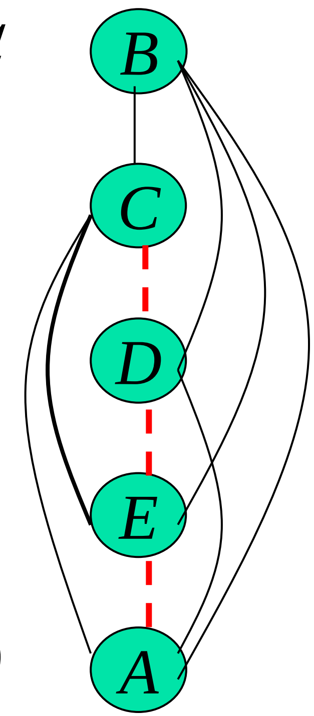
$P(e=0)$

$P(a|e=0)$

W=4*

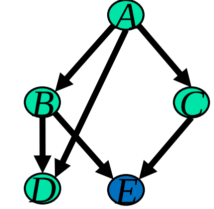
"induced width"

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

B=1

$P(e|b=1,c)$

bucket C: $P(c|a)$

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

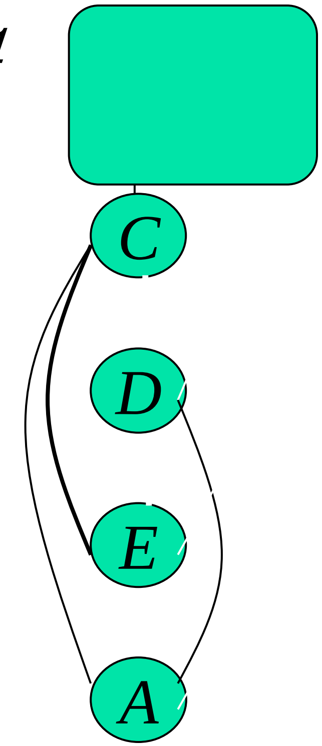
bucket E: $e=0$

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

$$\mathbf{P(e=0)}$$

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

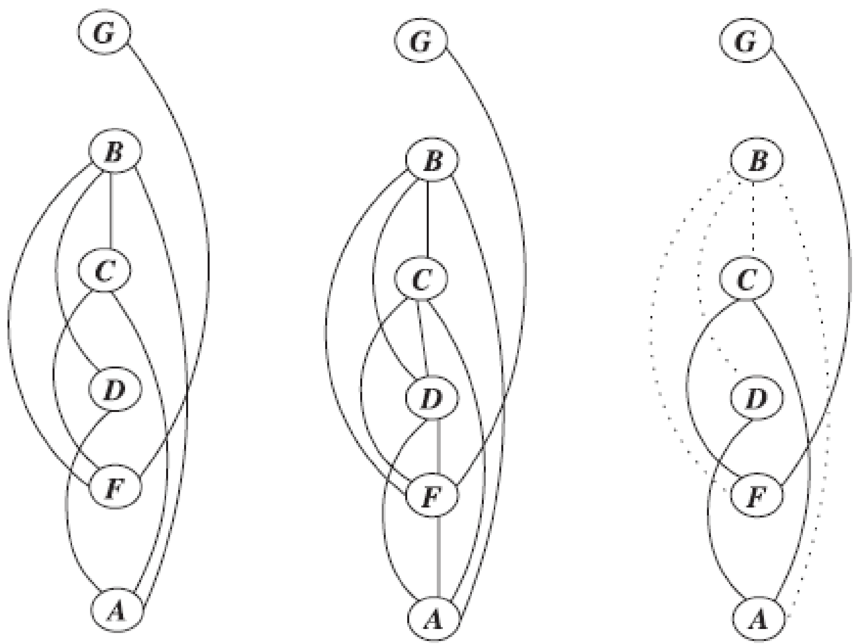
$$\mathbf{P(a|e=0)} = \frac{\mathbf{P(a,e=0)}}{\mathbf{P(e=0)}}$$



The Impact of Observations

(a) Directed acyclic graph

(b) Moral graph



(a)

(b)

(c)

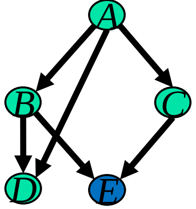
Figure 4.9: Adjusted induced graph relative to observing B.

Ordered graph *Induced graph* *Ordered conditioned graph*

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

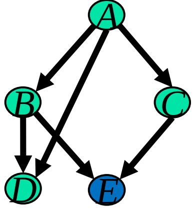
$$\text{MPE} = \max_{\bar{x}} P(\bar{x})$$



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

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

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



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

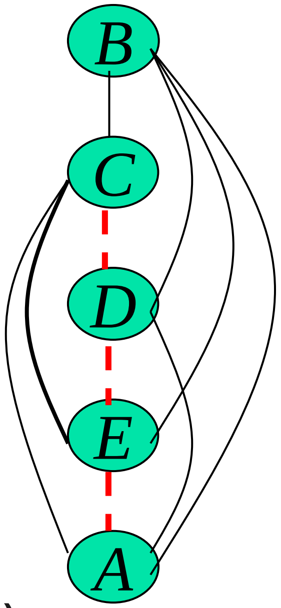
bucket A: $P(a)$ $h^E(a)$

MPE

W=4*

"induced width"

(max clique size)



Generating the MPE-tuple

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

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

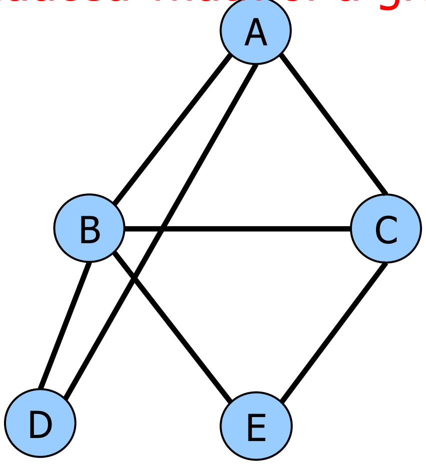
E: $e=0$ $h^D(a,e)$

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

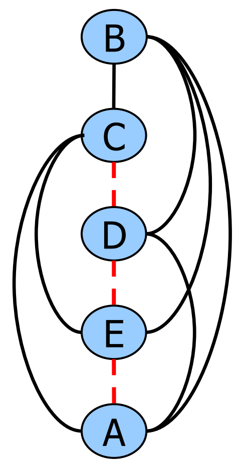
Return (a', b', c', d', e')

Induced Width

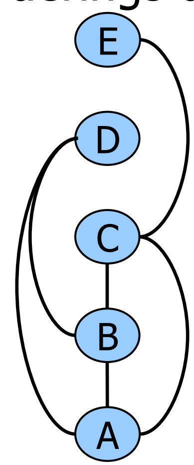
- Width is the max number of parents in the ordered graph
- Induced-width is the width of the induced ordered graph: recursively connecting parents going from last node to first.
- Induced-width w*(d) is the max induced-width over all nodes in ordering d
- Induced-width of a graph, w* is the min w*(d)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



primal graph



$w^*(d_1) = 4$



$w^*(d_2) = 2$

Complexity of Bucket Elimination

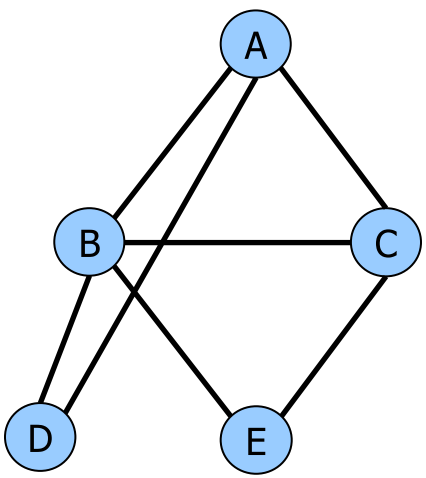
*Bucket-Elimination is **time** and **space***

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

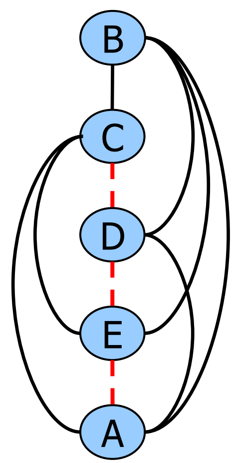
w_d^* *: the induced width of the primal graph along ordering d*

r = number of functions

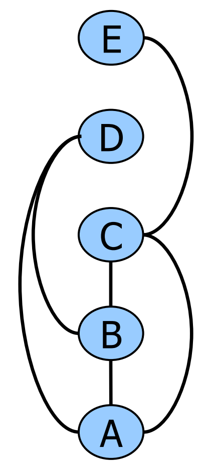
The effect of the ordering:



primal graph



$w^*(d_1) = 4$

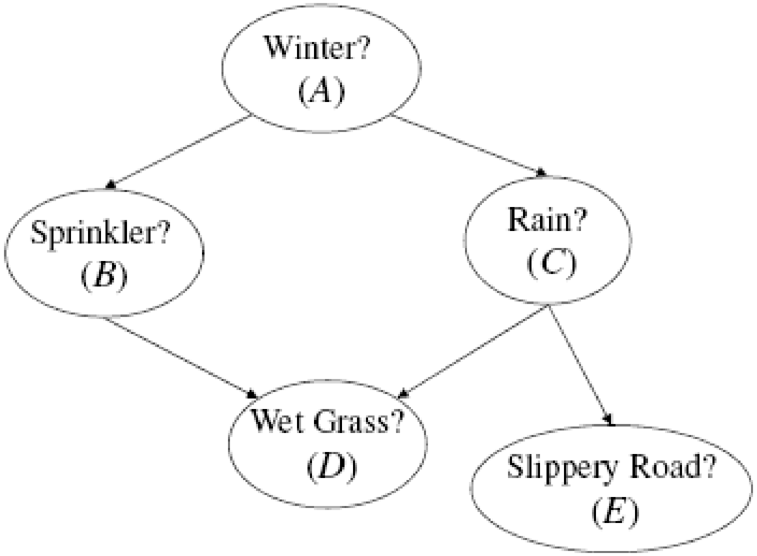


$w^*(d_2) = 2$

Finding smallest induced-width is hard!

A Bayesian Network

Example with mpe?



A	Θ_A
true	.6
false	.4

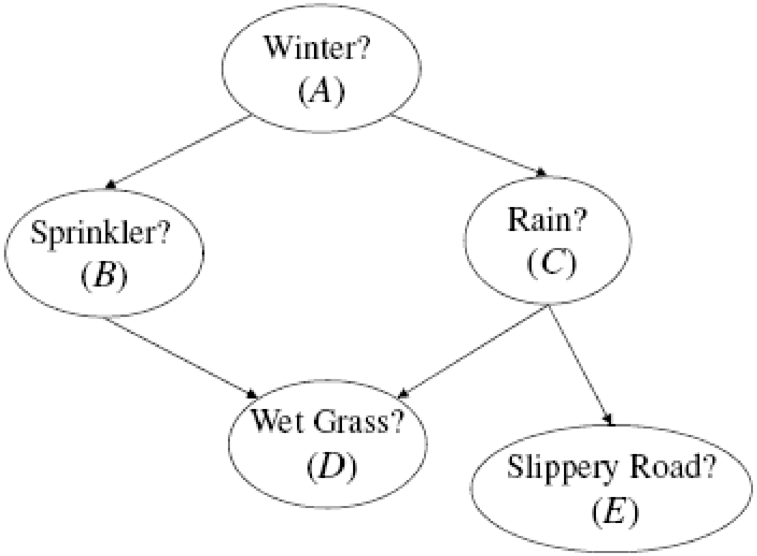
A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

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

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 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]

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]

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)

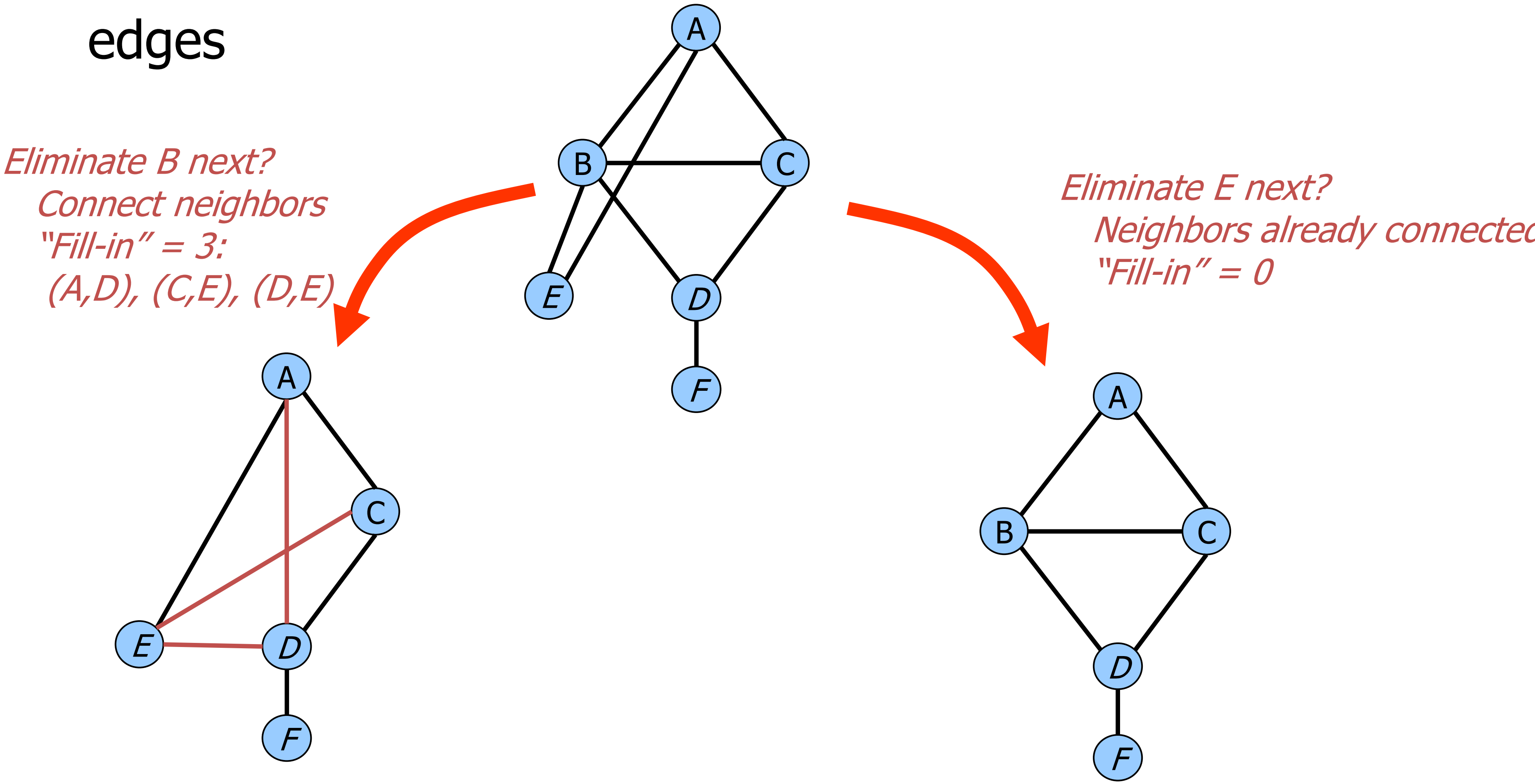
Greedy Orderings Heuristics

- 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

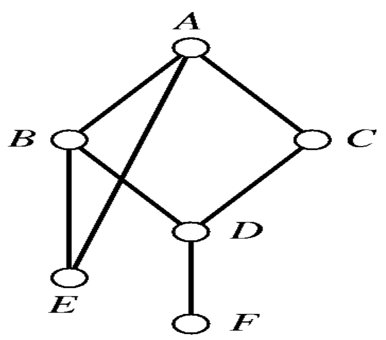
Complexity? $O(n^3)$

Min-Fill Heuristic

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

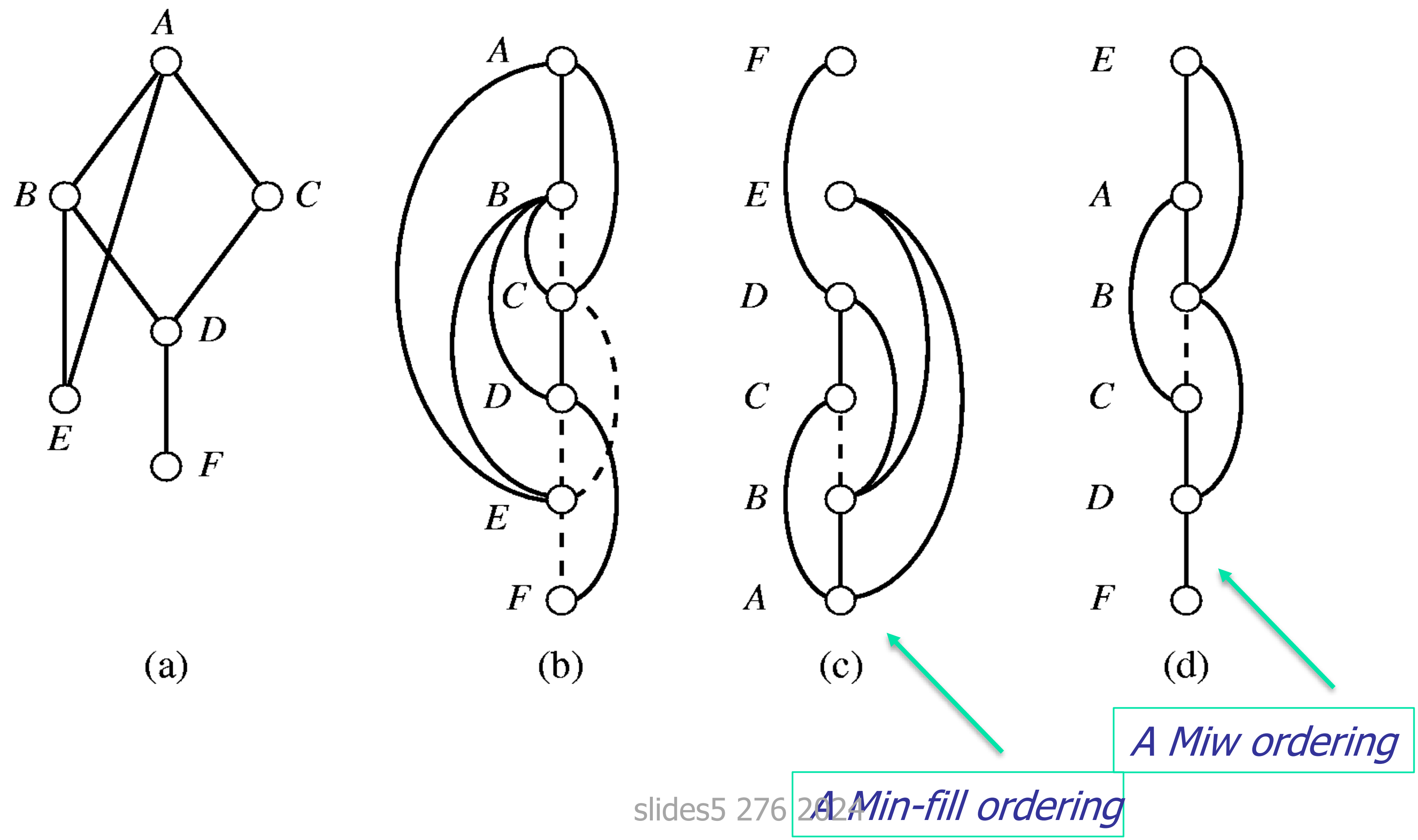


Example



(a)

Different Induced-Graphs



Which Greedy Algorithm is Best?

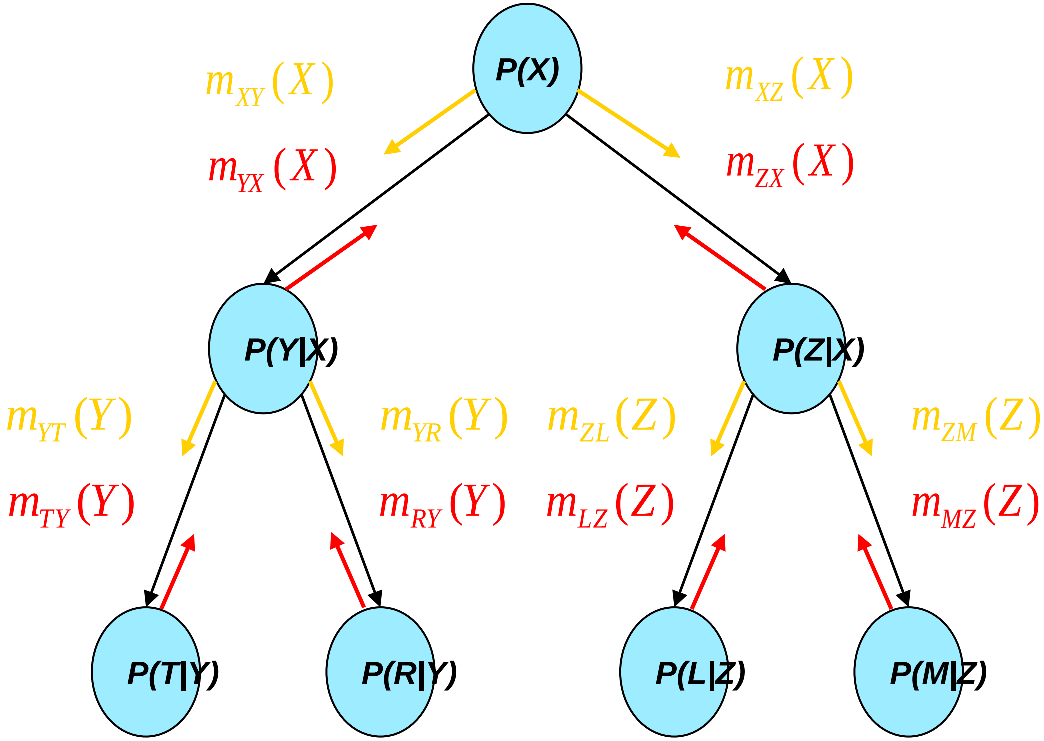
- Min-Fill, prefers a node who add the least number of fill-in arcs.

- 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)

Propagation in Both Directions

- Messages can propagate both ways and we get beliefs for each variable

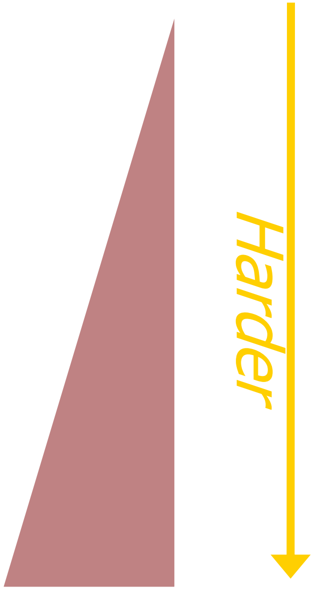


Inference for probabilistic networks

- 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)
  - Influence diagrams ?
- Induced-Width (Dechter, Chapter 3.4)

Marginal Map

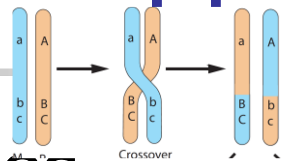
Max-Inference	$f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Sum-Inference	$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Mixed-Inference	$f(\mathbf{x}_M^*) = \max_{\mathbf{x}_{\mathrm{M}}} \sum_{\mathbf{x}_S} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$

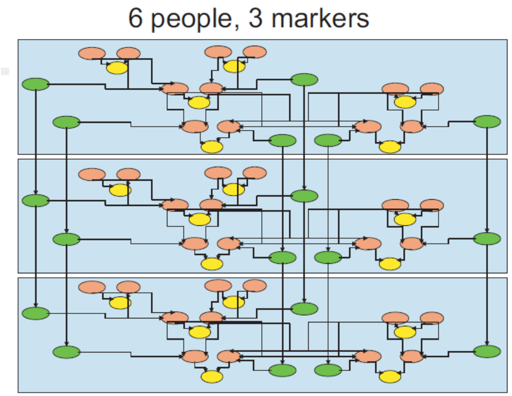


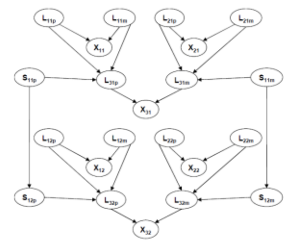
- **NP-hard**: exponentially many terms

Example for MMAP Applications

- Haplotype in Family pedigrees







- Coding networks

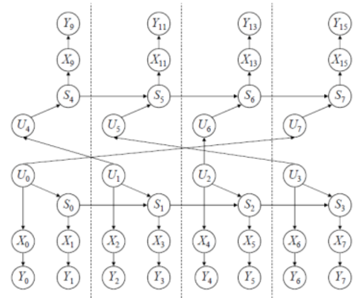
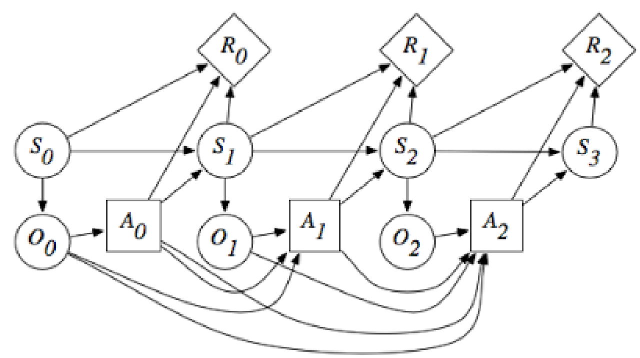
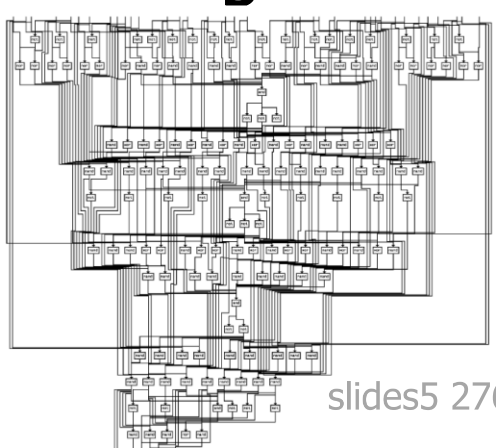


Figure 5.24: A Bayesian network for a turbo code.

- Probabilistic planning

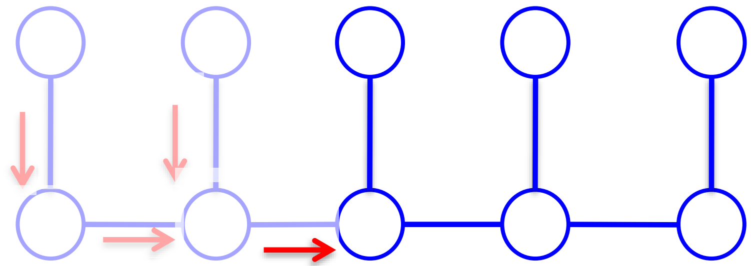


- Diagnosis

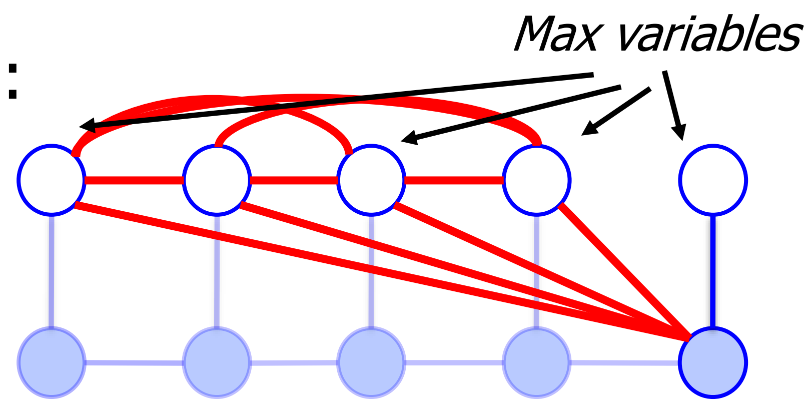


Marginal MAP is Not Easy on Trees

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



- Marginal MAP
  - Operations do not commute:

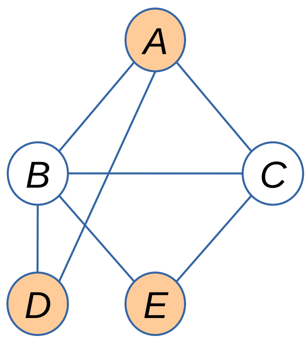


  - Sum must be done first!

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

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

MAX

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

Σ_B

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

Σ_C

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

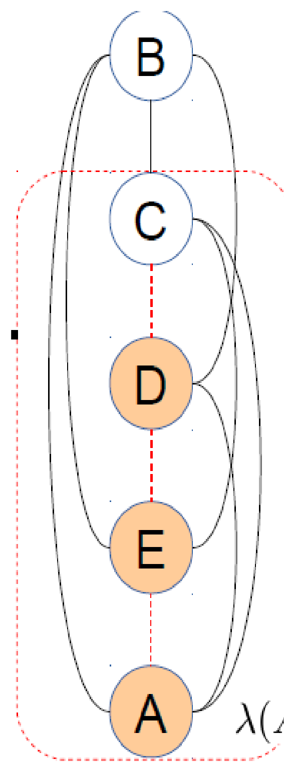
max_D

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

max_E

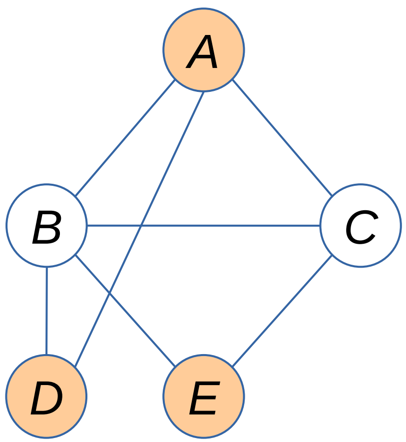
A: $\underbrace{\lambda^E(A)}$

MAP is the marginal MAP value*

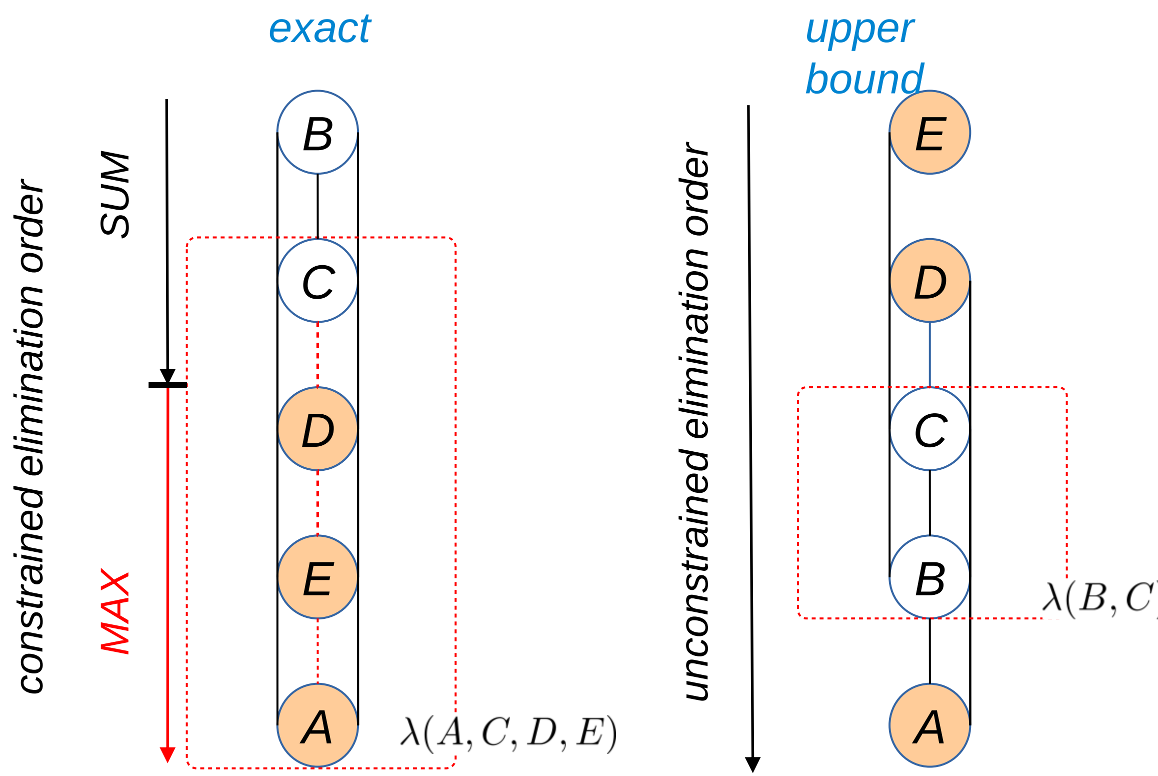


$\lambda(A,C,D,E)$

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)

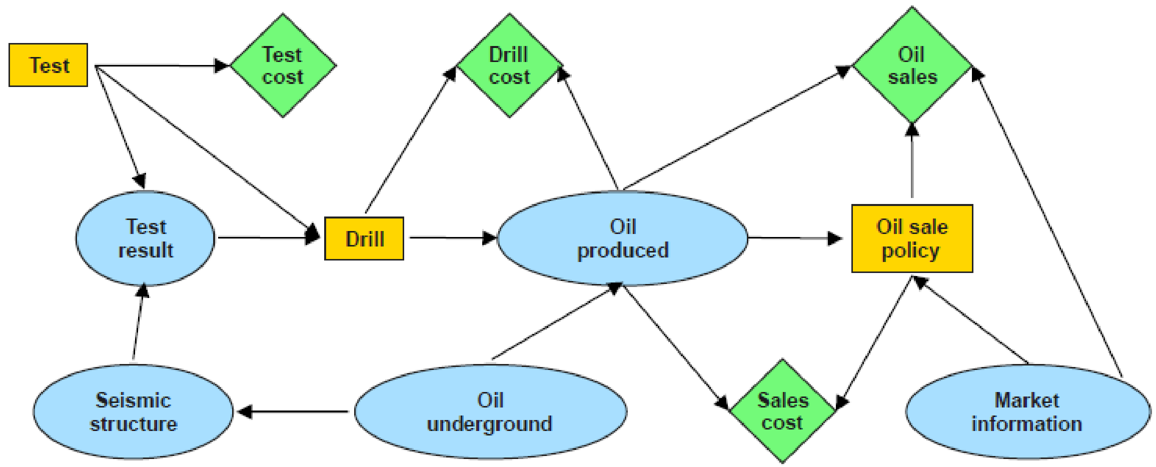
Inference for probabilistic networks

- 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)
- Mixed networks
- Influence diagrams ?

Ex: "oil wildcatter"

e.g., [Raiffa 1968; Shachter 1986]

- Influence diagram:



- Three actions: test, drill, sales policy
- Chance variables:

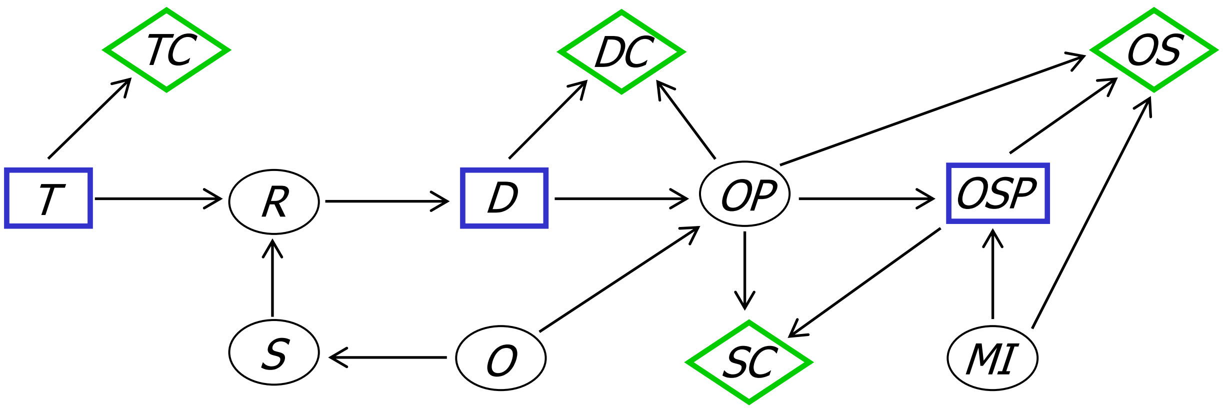
  P(oil) P(seismic|oil) P(result | seismic, test) P(produced | oil, drill) P(market)

- Utilities capture costs of actions, rewards of sale

  Oil sales - Test cost - Drill cost - Sales cost

Influence Diagrams

Influence diagram *ID = (X,D,P,R).*



Chance variables $X = X_1,\ldots,X_n$ ***over domains.***

Decision variables $D = D_1,\ldots,D_m$

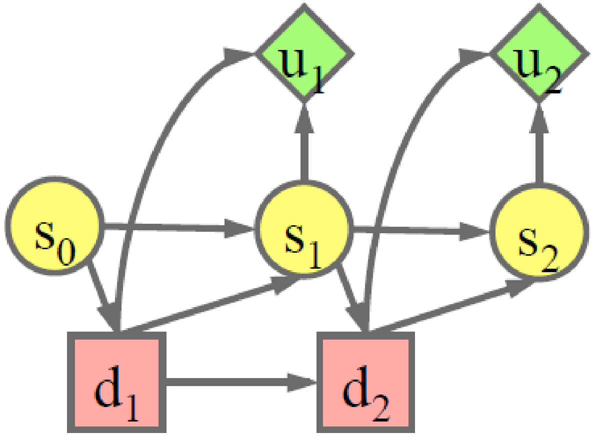
CPT's for chance variables $P_i = P(X_i \mid pa_i), i = 1..n$

Reward components $R = \{r_1,\ldots,r_j\}$

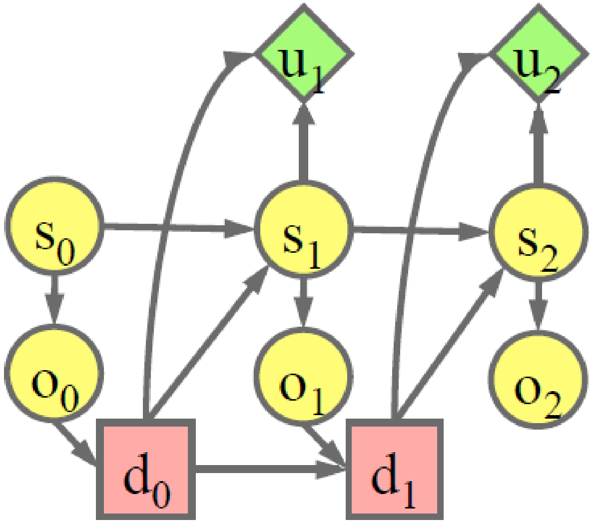
Utility function $u = \sum_i r_i$

Common examples

- Markov decision process
  - Markov chain state sequence
  - Actions “di” influence state transition
  - Rewards based on action, new state
  - Temporally homogeneous



- Partially observable MDP
  - Hidden Markov chain state sequence
  - Generate observations
  - Actions based on observations



Influence Diagrams (continue)

A decision rule *for* D_i *is a mapping:* $\delta_i : \Omega_{pa_{D_i}} \rightarrow \Omega_{D_i}$

where Ω_S *is the cross product of domains in* S*.*

A policy *is a list of decision rules* $\Delta = (\delta_1, \ldots, \delta_m)$

Task: *Find an optimal policy that maximizes the expected utility.*

$$E = \max_{\Delta=(\delta_1,\ldots,\delta_m)} \sum_{x=(x_1,\ldots,x_n)} \prod_i P_i(x) u(x)$$

The Car Example (Howard 1976)

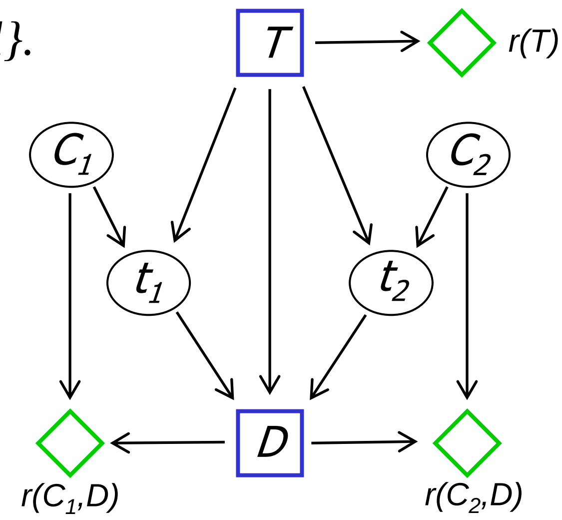
A car buyer needs to buy one of two used cars. The buyer can carry out tests with various costs, and then, decide which car to buy.

- T: *Test variable* (t_0, t_1, t_2) (t_1 *test car 1*, t_2 *test car 2)*
- D: *the decision of which car to buy,* $D \in \{buy1, buy2\}$
- C_i: *the quality of car* i, $C_i \in \{q_1, q_2\}$
- t_i: *the outcome of the test on car* i, $t_i \in \{pass, fail, null\}$.
- $r(T)$: *The cost of testing,*
- $r(C_1,D)$, $r(C_2,D)$: *the reward in buying cars 1 and 2.*
- *The utility is:* $r(T) + r(C_1,D) + r(C_2,D)$.

Task: determine decision rules T and D such that:

$$E = \max_{T,D} \sum_{t_2,t_1,C_2,C_1} P(t_2,|C_2,T)P(C_2)P(t_1|C_1,T) \cdot$$

$$P(C_1)[r(T) + r(C_2,D) + r(C_1,D)]$$



Bucket Elimination for meu (Algorithm Elim-meu-id)

Input: *An Influence diagram* $ID = \{P_1,\ldots,P_n,r_1,\ldots,r_j\}$
Output: *Meu and optimizing policies.*

1. ***Order the variables and partition into buckets.***
2. ***Process buckets from last to first:***

$o = T,t_2,t_2,D,C_2,C_1$

$bucket(C_1)$: $P(C_1), P(t_1|C_1,T), r(C_1,D)$

$bucket(C_2)$: $P(C_2), P(t_2|C_2,T), r(C_2,D)$

$bucket(D)$: $\theta_{C_1}(t_1,T,D), \theta_{C_2}(t_2,T,D)$

$bucket(t_1)$: $\lambda_{C_1}(t_1,T)$ $\theta_D(t_1,t_2,T),$ $\delta(t_1,t_2,T)$

$bucket(t_2)$: $\lambda_{C_2}(t_2,T)$ $\theta_{t_1}(t_2,T)$

$bucket(T)$: $r(T)$ $\lambda_{t1}(T)$ $\lambda_{t2}(T)$ $\theta_{t1}(T)$

θ_T, δ_T

3. ***Forward:*** *Assign values in ordering d*

The Bucket Description

Final buckets: *(λs or Ps) utility components (θ's or r's).*

bucket(C_1): $P(C_1), P(t_1|C_1,T), r(C_1,D)$

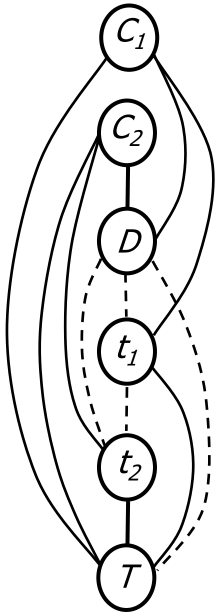
bucket(C_2): $P(C_2), P(t_2|C_2,T), r(C_2,D)$

bucket(D): $\theta_{C_1}(t_1,T,D), \theta_{C_2}(t2,T,D)$

bucket(t_1): $\lambda_{C_1}(t_1,T), \quad \theta_D(t_1,t_2,T)$

bucket(t_2): $\lambda_{C_2}(t_2,T), \quad \theta_{t_1}(t_2,T)$

bucket(T): r(T)



Optimizing policies: δ_T is argmax of θ_T computed in bucket(T), and $\theta_D(t_1,t_2,T)$ in bucket(t_1).

General Graphical Models

Definition 2.2 Graphical model. A *graphical model* $\mathcal{M}$ is a 4-tuple, $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \bigotimes \rangle$, where:

1. $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a finite set of variables;
2. $\mathbf{D} = \{D_1, \ldots, D_n\}$ is the set of their respective finite domains of values;
3. $\mathbf{F} = \{f_1, \ldots, f_r\}$ is a set of positive real-valued discrete functions, defined over scopes of variables $\mathcal{S} = \{S_1, ..., S_r\}$, where $\mathbf{S}_i \subseteq \mathbf{X}$. They are called *local* functions.
4. $\bigotimes$ is a *combination* operator (e.g., $\bigotimes \in \{\prod, \sum, \bowtie\}$ (product, sum, join)). The combination operator can also be defined axiomatically as in [Shenoy, 1992], but for the sake of our discussion we can define it explicitly, by enumeration.

The graphical model represents a *global function* whose scope is $\mathbf{X}$ which is the combination of all its functions: $\bigotimes_{i=1}^{r} f_i$.

General Bucket Elimination

Algorithm General bucket elimination (GBE)

Input: $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \otimes \rangle$. $F = \{f_1, ..., f_n\}$ an ordering of the variables, $d = X_1, ..., X_n$; $\mathbf{Y} \subseteq \mathbf{X}$.

Output: A new compiled set of functions from which the query $\Downarrow_Y \otimes_{i=1}^n f_i$ can be derived in linear time.

1. **Initialize:** Generate an ordered partition of the functions into $bucket_1, ..., bucket_n$, where $bucket_i$ contains all the functions whose highest variable in their scope is X_i. An input function in each bucket ψ_i, $\psi_i = \otimes_{i=1}^n f_i$.
2. **Backward:** For $p \leftarrow n$ downto 1, do
for all the functions $\psi_p, \lambda_1, \lambda_2, ..., \lambda_j$ in $bucket_p$, do

- **If** (observed variable) $X_p = x_p$ appears in $bucket_p$, assign $X_p = x_p$ in ψ_p and to each λ_i and put each resulting function in appropriate bucket.
- **else,** (combine and marginalize)
$\lambda_p \leftarrow \Downarrow_{S_p} \psi_p \otimes (\otimes_{i=1}^j \lambda_i)$ and add λ_p to the largest-index variable in $scope(\lambda_p)$.

3. **Return:** all the functions in each bucket.

Theorem 4.23 Correctness and complexity. *Algorithm GBE is sound and complete for its task. Its time and space complexities is exponential in the $w^*(d) + 1$ and $w^*(d)$, respectively, along the order of processing d.*

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]

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]

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. $\quad r \leftarrow$ a node in G with smallest degree.
3. $\quad$ 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

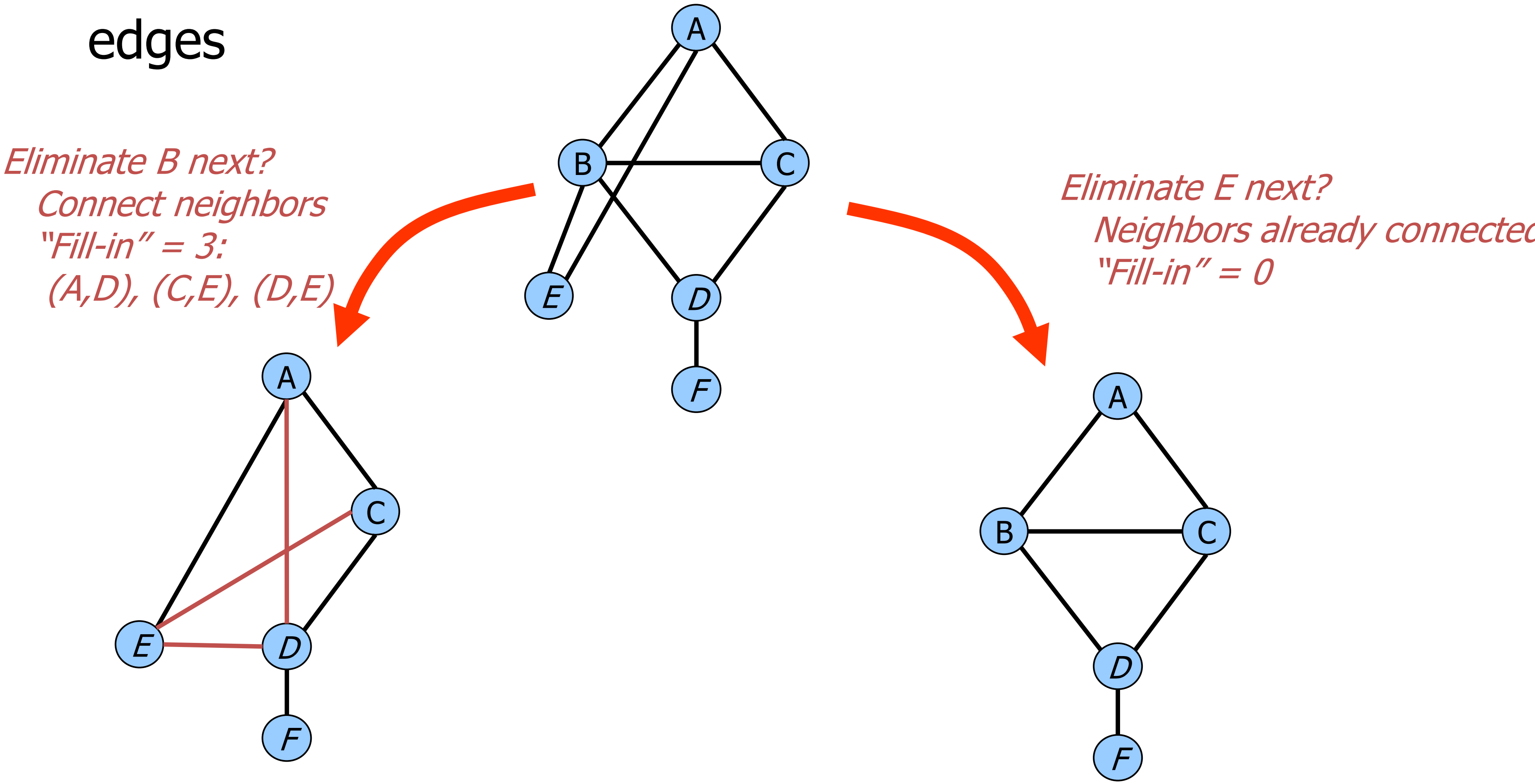
- Min-induced-width
  - From last to first, pick a node with smallest width, then connect parent and remove

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

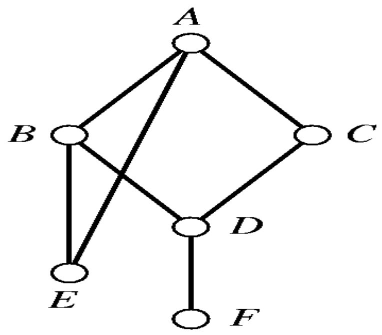
Complexity? $O(n^3)$

Min-Fill Heuristic

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

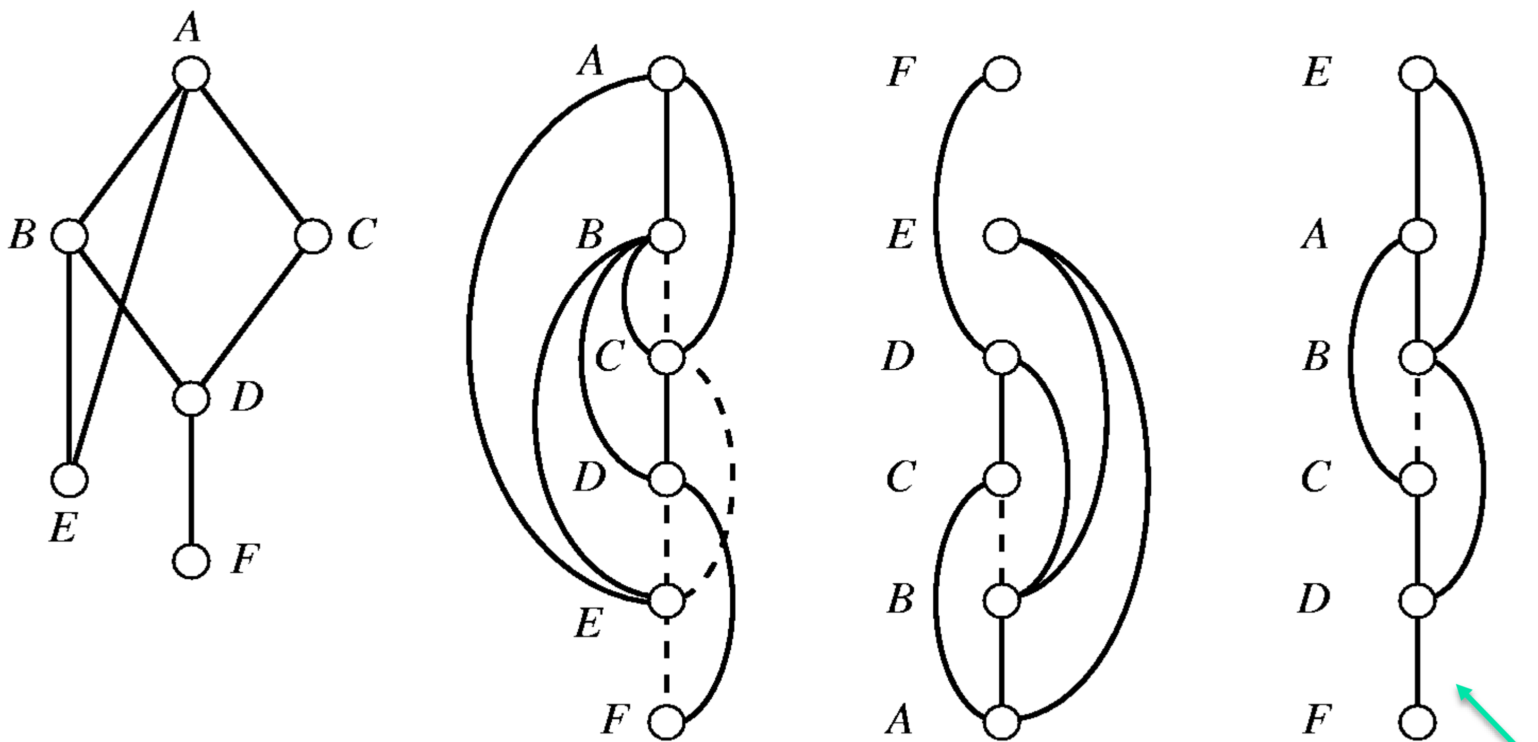


Example



(a)

Different Induced-Graphs



(a) (b) (c) (d)

A Miw ordering

A Min-fill ordering

Which Greedy Algorithm is Best?

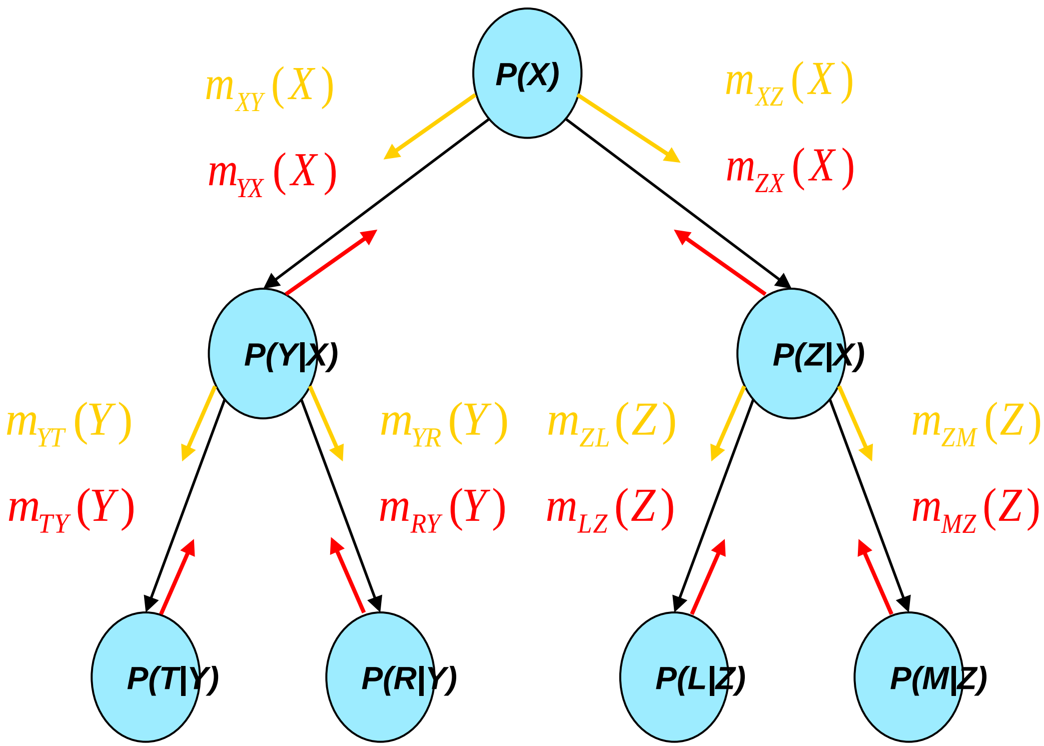
- Min-Fill, prefers a node who add the least number of fill-in arcs.

- 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)

Propagation in Both Directions

- Messages can propagate both ways and we get beliefs for each variable

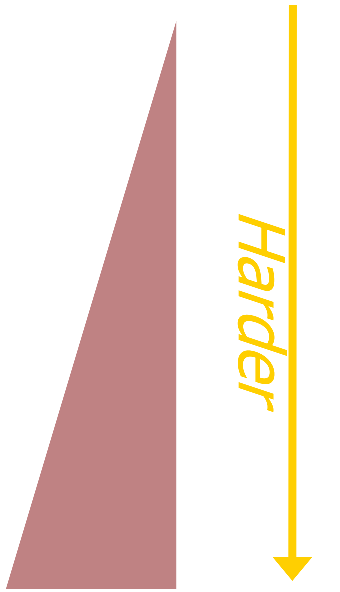


Inference for probabilistic networks

- 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)
  - Influence diagrams ?
- Induced-Width (Dechter, Chapter 3.4)

Marginal Map

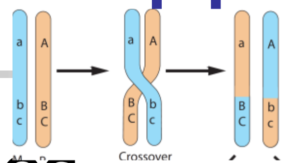
Max-Inference	$f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Sum-Inference	$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Mixed-Inference	$f(\mathbf{x}_M^*) = \max_{\mathbf{x}_\mathrm{M}} \sum_{\mathbf{x}_S} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$

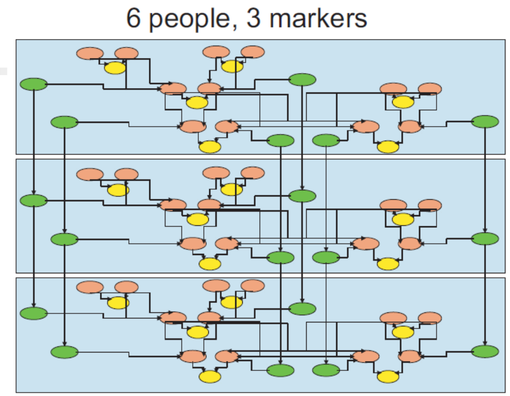


- **NP-hard**: exponentially many terms

Example for MMAP Applications

- Haplotype in Family pedigrees





- Coding networks

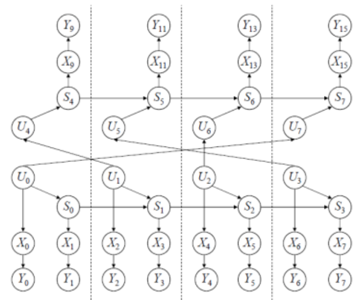
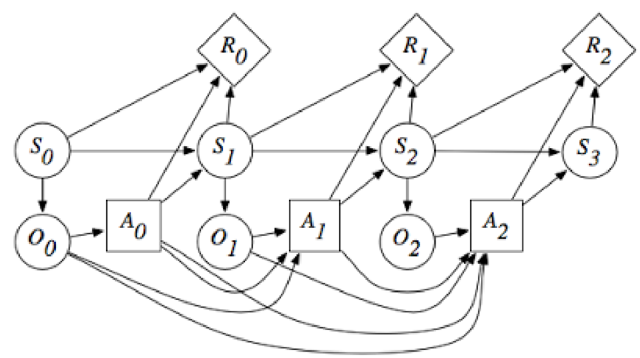
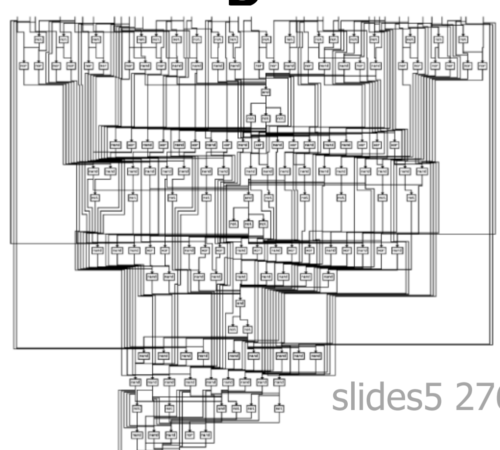


Figure 5.24: A Bayesian network for a turbo code.

- Probabilistic planning

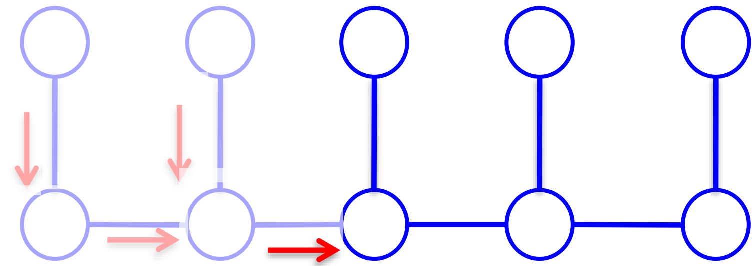


- Diagnosis



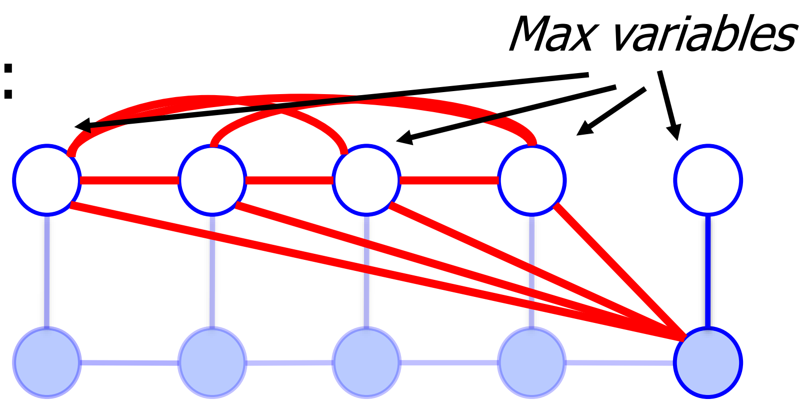
Marginal MAP is Not Easy on Trees

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



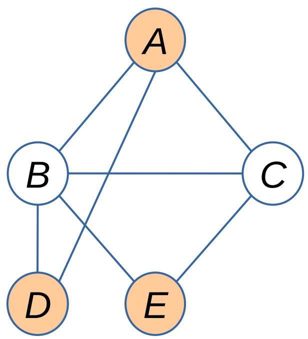
- Marginal MAP
  - Operations do not commute:
  - Sum must be done first!

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



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

Σ_B

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

Σ_C

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

max_D

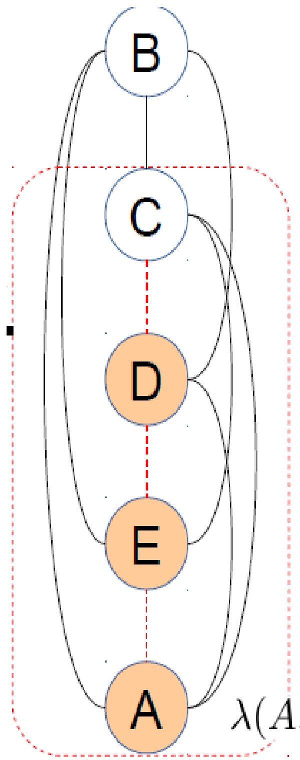
E: $\underbrace{\lambda^D(A,E)}$

max_E

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

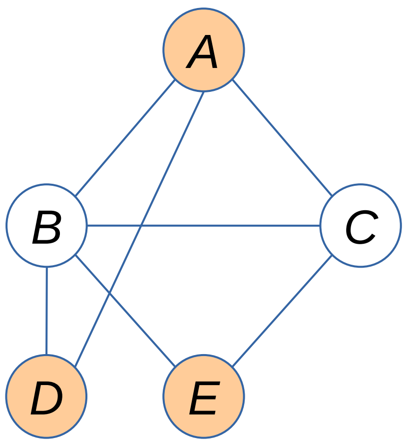
MAX

MAP is the marginal MAP value*

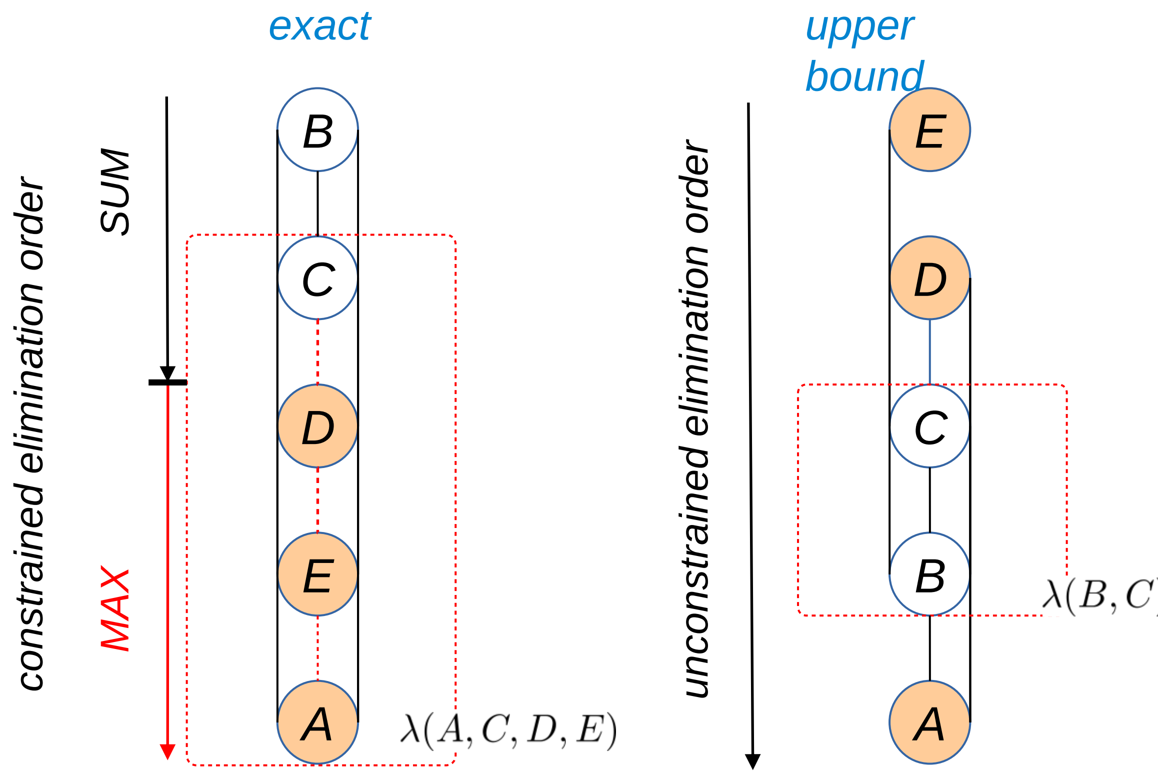


$\lambda(A,C,D,E)$

Why is MMAP harder?



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



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

In practice, constrained induced is much larger!

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

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

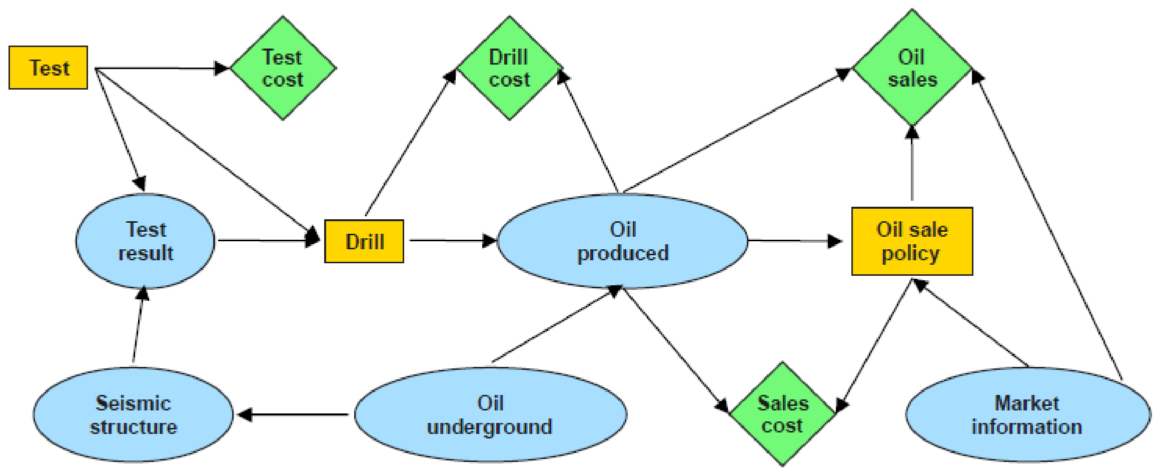
Inference for probabilistic networks

- 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)
- Mixed networks
- Influence diagrams ?

Ex: “oil wildcatter”

e.g., [Raiffa 1968; Shachter 1986]

- Influence diagram:



- Three actions: test, drill, sales policy
- Chance variables:

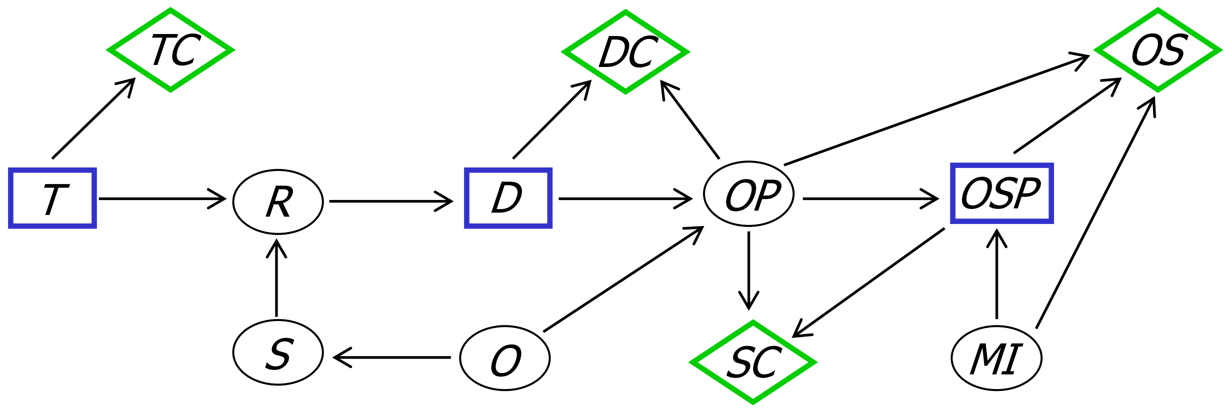
  P(oil) P(seismic|oil) P(result | seismic, test) P(produced | oil, drill) P(market)

- Utilities capture costs of actions, rewards of sale

  Oil sales - Test cost - Drill cost - Sales cost

Influence Diagrams

Influence diagram *ID = (X,D,P,R).*



Chance variables $X = X_1,\ldots,X_n$ ***over domains.***

Decision variables $D = D_1,\ldots,D_m$

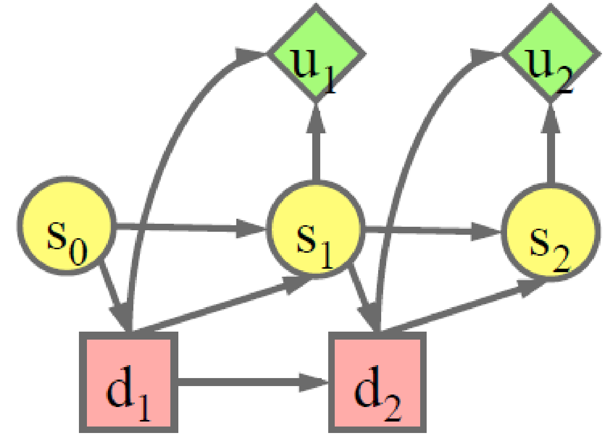
CPT's for chance variables $P_i = P(X_i \mid pa_i), i = 1..n$

Reward components $R = \{r_1,\ldots,r_j\}$

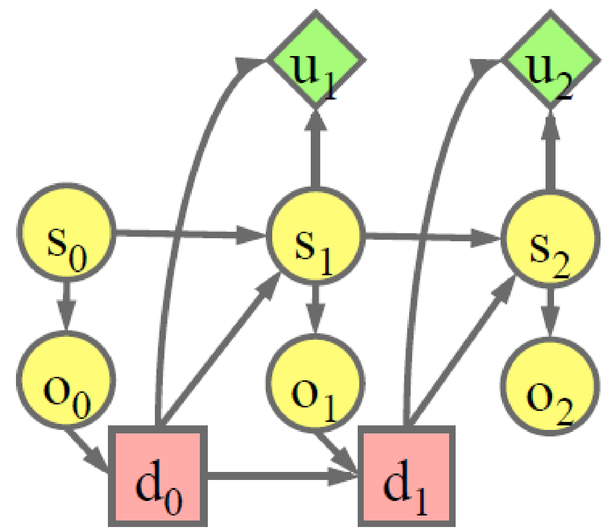
Utility function $u = \sum_i r_i$

Common examples

- Markov decision process
  - Markov chain state sequence
  - Actions "di" influence state transition
  - Rewards based on action, new state
  - Temporally homogeneous



- Partially observable MDP
  - Hidden Markov chain state sequence
  - Generate observations
  - Actions based on observations



Influence Diagrams (continue)

A decision rule *for* D_i *is a mapping:* $\delta_i : \Omega_{pa_{D_i}} \rightarrow \Omega_{D_i}$

where Ω_S *is the cross product of domains in* S.

A policy *is a list of decision rules* $\Delta = (\delta_1, \ldots, \delta_m)$

Task: *Find an optimal policy that maximizes the expected utility.*

$$E = \max_{\Delta=(\delta_1,\ldots,\delta_m)} \sum_{x=(x_1,\ldots,x_n)} \prod_i P_i(x) u(x)$$

The Car Example (Howard 1976)

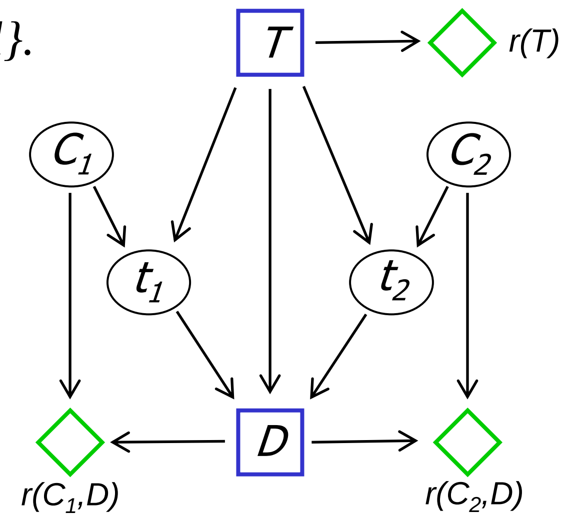
A car buyer needs to buy one of two used cars. The buyer can carry out tests with various costs, and then, decide which car to buy.

- T: *Test variable* (t_0, t_1, t_2) *(*t_1 *test car 1,* t_2 *test car 2)*
- D: *the decision of which car to buy,* $D \in \{buy1, buy2\}$
- C_i: *the quality of car* i, $C_i \in \{q_1, q_2\}$
- t_i: *the outcome of the test on car* i, $t_i \in \{pass, fail, null\}$.
- $r(T)$: *The cost of testing,*
- $r(C_1,D)$, $r(C_2,D)$: *the reward in buying cars 1 and 2.*
- *The utility is:* $r(T) + r(C_1,D) + r(C_2,D)$.

Task: determine decision rules T and D such that:

$$E = \max_{T,D} \sum_{t_2,t_1,C_2,C_1} P(t_2,|C_2,T)P(C_2)P(t_1 | C_1,T) \cdot$$

$$P(C_1)[r(T) + r(C_2,D) + r(C_1,D)]$$



Bucket Elimination for meu (Algorithm Elim-meu-id)

Input: *An Influence diagram* $ID = \{P_1,\ldots,P_n,r_1,\ldots,r_j\}$
Output: *Meu and optimizing policies.*

1. ***Order the variables and partition into buckets.***
2. ***Process buckets from last to first:***

$o = T,t_2,t_2,D,C_2,C_1$

$bucket(C_1)$: $P(C_1), P(t_1|C_1,T), r(C_1,D)$

$bucket(C_2)$: $P(C_2), P(t_2|C_2,T), r(C_2,D)$

$bucket(D)$: $\theta_{C_1}(t_1,T,D), \theta_{C_2}(t_2,T,D)$

$bucket(t_1)$: $\lambda_{C_1}(t_1,T)$ $\theta_D(t_1,t_2,T)$, $\delta(t_1,t_2,T)$

$bucket(t_2)$: $\lambda_{C_2}(t_2,T)$ $\theta_{t_1}(t_2,T)$

$bucket(T)$: $r(T)$ $\lambda_{t_1}(T)$ $\lambda_{t_2}(T)$ $\theta_{t_1}(T)$

θ_T, δ_T

3. ***Forward:*** *Assign values in ordering d*

The Bucket Description

***Final buckets:** (λs or Ps) utility components (θ's or r's).*

bucket(C_1): $P(C_1), P(t_1|C_1,T), r(C_1,D)$

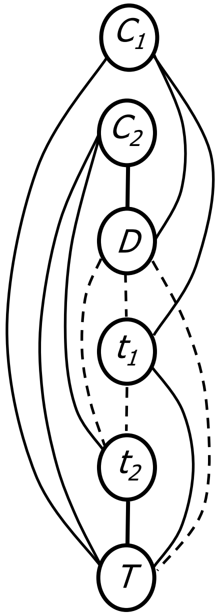
bucket(C_2): $P(C_2), P(t_2|C_2,T), r(C_2,D)$

bucket(D): $\theta_{C_1}(t_1,T,D), \theta_{C_2}(t2,T,D)$

bucket(t_1): $\lambda_{C_1}(t_1,T), \quad \theta_D(t_1,t_2,T)$

bucket(t_2): $\lambda_{C_2}(t_2,T), \quad \theta_{t_1}(t_2,T)$

bucket(T): r(T)



Optimizing policies: δ_T is argmax of θ_T computed in bucket(T), and $\theta_D(t_1,t_2,T)$ in bucket(t_1).

General Graphical Models

Definition 2.2 Graphical model. A *graphical model* $\mathcal{M}$ is a 4-tuple, $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \bigotimes \rangle$, where:

1. $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a finite set of variables;
2. $\mathbf{D} = \{D_1, \ldots, D_n\}$ is the set of their respective finite domains of values;
3. $\mathbf{F} = \{f_1, \ldots, f_r\}$ is a set of positive real-valued discrete functions, defined over scopes of variables $\mathcal{S} = \{S_1, ..., S_r\}$, where $\mathbf{S}_i \subseteq \mathbf{X}$. They are called *local* functions.
4. $\bigotimes$ is a *combination* operator (e.g., $\bigotimes \in \{\prod, \sum, \bowtie\}$ (product, sum, join)). The combination operator can also be defined axiomatically as in [Shenoy, 1992], but for the sake of our discussion we can define it explicitly, by enumeration.

The graphical model represents a *global function* whose scope is $\mathbf{X}$ which is the combination of all its functions: $\bigotimes_{i=1}^{r} f_i$.

General Bucket Elimination

Algorithm General bucket elimination (GBE)

Input: $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \otimes \rangle$. $F = \{f_1, ..., f_n\}$ an ordering of the variables, $d = X_1, ..., X_n$; $\mathbf{Y} \subseteq \mathbf{X}$.

Output: A new compiled set of functions from which the query $\Downarrow_Y \otimes_{i=1}^n f_i$ can be derived in linear time.

1. **Initialize:** Generate an ordered partition of the functions into $bucket_1, ..., bucket_n$, where $bucket_i$ contains all the functions whose highest variable in their scope is X_i. An input function in each bucket ψ_i, $\psi_i = \otimes_{i=1}^n f_i$.
2. **Backward:** For $p \leftarrow n$ downto 1, do
for all the functions $\psi_p, \lambda_1, \lambda_2, ..., \lambda_j$ in $bucket_p$, do

- **If** (observed variable) $X_p = x_p$ appears in $bucket_p$, assign $X_p = x_p$ in ψ_p and to each λ_i and put each resulting function in appropriate bucket.
- **else,** (combine and marginalize)
$\lambda_p \leftarrow \Downarrow_{S_p} \psi_p \otimes (\otimes_{i=1}^j \lambda_i)$ and add λ_p to the largest-index variable in $scope(\lambda_p)$.

3. **Return:** all the functions in each bucket.

Theorem 4.23 Correctness and complexity. *Algorithm GBE is sound and complete for its task. Its time and space complexities is exponential in the* $w^*(d) + 1$ *and* $w^*(d)$*, respectively, along the order of processing* d.