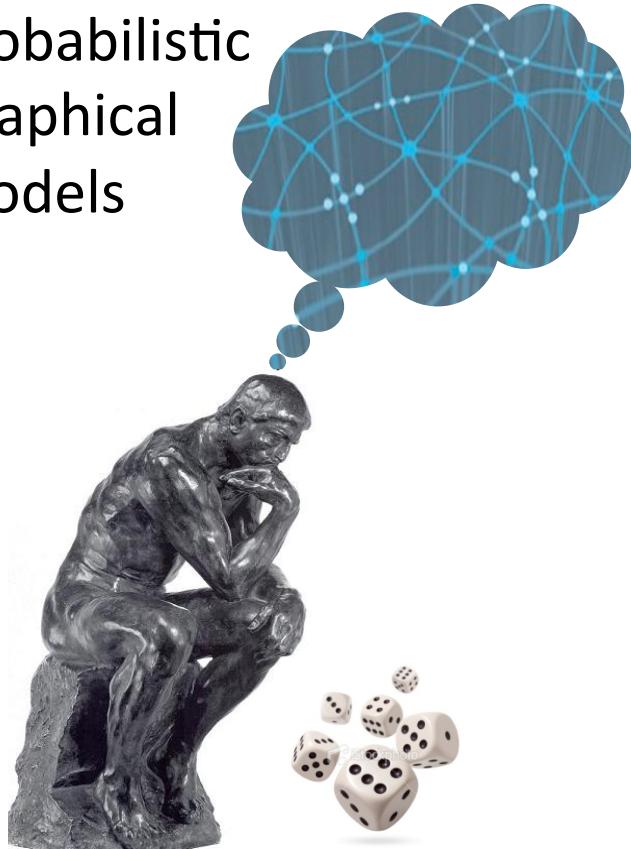


Probabilistic  
Graphical  
Models



Inference

---

Overview

---

Conditional  
Probability  
Queries

# Conditional Probability Queries

- Evidence:  $\underline{E = e}$
  - Query: a subset of variables  $\underline{y}$
  - Task: compute  $P(\underline{y} \mid \underline{E=e})$
- 
- Applications
    - Medical/fault diagnosis
    - Pedigree analysis ↙

# NP-Hardness

The following are all NP-hard

worst case  
general case

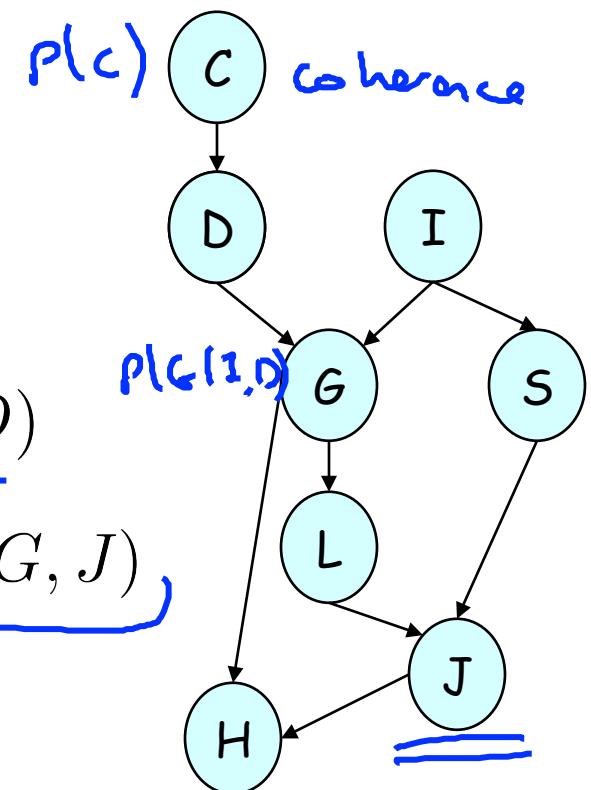
- Given a PGM  $P_\Phi$ , a variable  $X$  and a value  $x \in \text{Val}(X)$ , compute  $P_\Phi(X=x)$ 
  - Or even decide if  $P_\Phi(X=x) > 0$
- Let  $\varepsilon < 0.5$ . Given a PGM  $P_\Phi$ , a variable  $X$  and a value  $x \in \text{Val}(X)$ , and observation  $e \in \text{Val}(E)$ , find a number  $p$  that has  $|P_\Phi(X=x|E=e) - p| < \varepsilon$

# Sum-Product

$p(j)$

$$\sum_{C,D,I,G,S,L,H} \frac{\phi_C(C)\phi_D(C,D)\phi_I(I)\phi_G(G,I,D)}{\phi_S(S,I)\phi_L(L,G)\phi_J(J,L,S)\phi_H(H,G,J),}$$

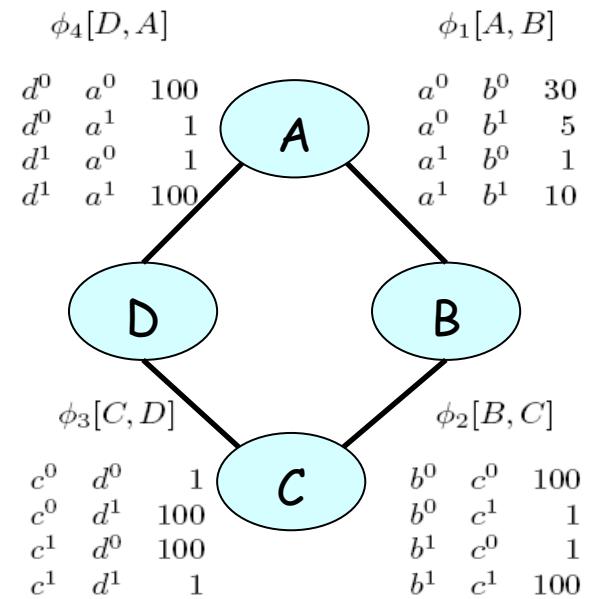
joint dist.  
chain rule



# Sum-Product

$$\tilde{p}(D) = \frac{\sum_{A,B,C} \phi_1(A, B) \phi_2(B, C) \phi_3(C, D) \phi_4(A, D)}{\tilde{p}(A, B, C, D)}$$

$\tilde{p}(D)$  =  $\frac{\tilde{p}(D)}{\text{renormalization}}$



Daphne Koller

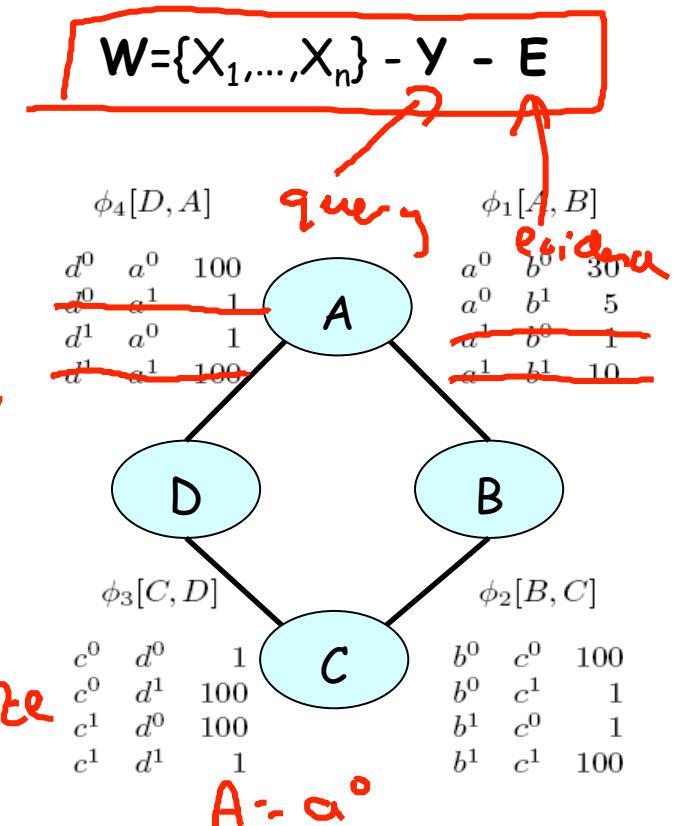
# Evidence: Reduced Factors

$$\underline{P(Y | E = e)} = \frac{\underline{P(Y, E = e)}}{\underline{P(E = e)}}$$

$$\underline{P(Y, E = e)} = \sum_W P(Y, \underline{W}, E = e)$$

$$= \sum_W \frac{1}{Z} \prod_k \phi_k(D_k, \underline{E = e})$$

$$\cancel{\propto} \sum_W \cancel{\frac{1}{Z}} \prod_k \phi'_k(D'_k) \quad \text{renormalize}$$



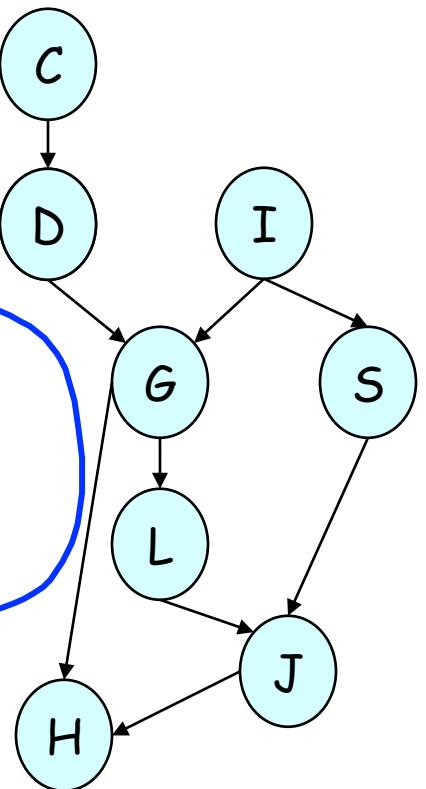
# Evidence: Reduced Factors

$$P(J|I=i, H=h)$$

$$P(J, I=i, H=h) =$$

$$\sum_{C,D,I,G,S,L,H} \phi_C(C)\phi_D(C,D)\phi_I(I)\phi_G(G,I,D) \\ \phi_S(S,J)\phi_L(L,G)\phi_J(J,L,S)\phi_H(H,G,J)$$

normalize



# Sum-Product

$$P(Y \mid E = e) = \frac{P(Y, E = e)}{P(E = e)}$$

$$P(Y, E = e) = \sum_{\underline{W}} \frac{1}{Z} \prod_k \phi'_k(D'_k)$$

*reduced numerator*

$$P(E = e) = \sum_Y \sum_{\underline{W}} \frac{1}{Z} \prod_k \phi'_k(D'_k)$$

*denominator*

Compute  $\sum_{\underline{W}} \prod_k \phi'_k(D'_k)$  and renormalizing

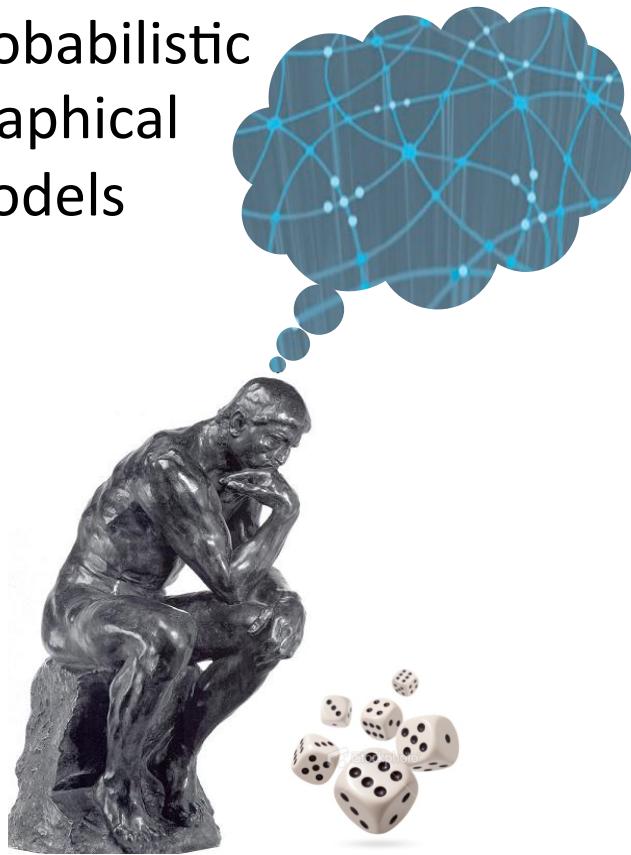
# Algorithms: Conditional Probability

- Push summations into factor product
    - Variable elimination (*dynamic programming*)
  - Message passing over a graph
    - Belief propagation
    - Variational approximations
  - Random sampling instantiations
    - Markov chain Monte Carlo (MCMC)
    - Importance sampling
- exact*  
*approx*  
*approx*

# Summary

- Conditional probability queries of subset of variables given evidence on others
- Summing over factor product
- Evidence simply reduces the factors
- Many exact and approximate algorithms

Probabilistic  
Graphical  
Models



## Inference

---

### Overview

---

### MAP Inference

# Maximum a Posteriori (MAP)

- Evidence:  $E=e$
- Query: all other variables  $Y$  ( $Y=\{X_1, \dots, X_n\} - E$ ) *then*
- Task: compute  $\text{MAP}(Y|E=e) = \underset{y}{\operatorname{argmax}} P(Y=y | E=e)$ 
  - Note: there may be more than one possible solution
- Applications
  - Message decoding: most likely transmitted message
  - Image segmentation: most likely segmentation

# MAP ≠ Max over Marginals

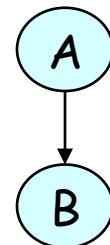
- Joint distribution

$$- P(a^0, b^0) = 0.04$$

$$\text{--- } P(\underline{a^0}, \underline{b^1}) = 0.36$$

$$- P(a^1, b^0) = 0.3$$

$$- P(a^1, b^1) = 0.3$$



$P(A)$

$P(B|A)$

<i>I</i>	
$a^0$	$a^1$
0.4	0.6

<i>B</i>		
<i>A</i>	$B^0$	$B^1$
$a^0$	0.1	0.9
$a^1$	0.5	0.5

Daphne Koller

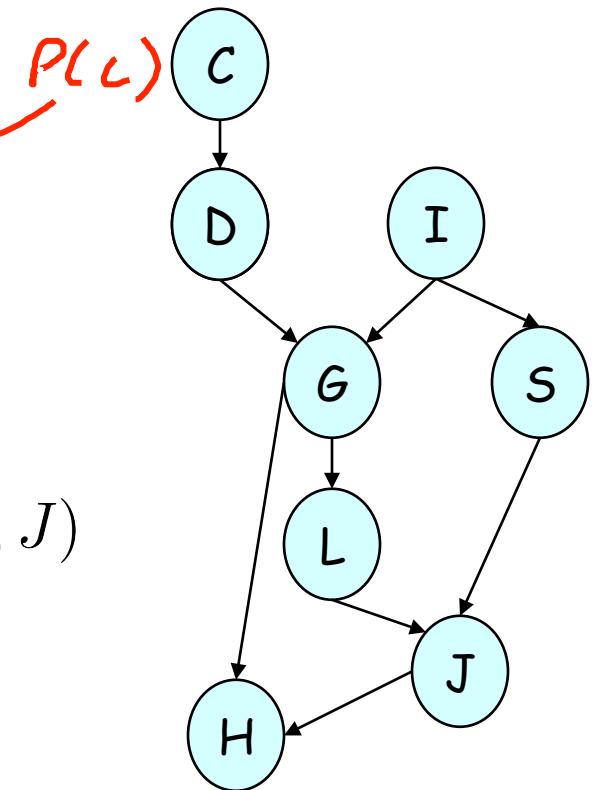
# NP-Hardness

The following are NP-hard

- Given a PGM  $P_\Phi$ , find a joint assignment  $x$  with highest probability  $P_\Phi(x)$
- Given a PGM  $\underline{P_\Phi}$  and a probability  $p$ , decide if there is an assignment  $\underline{x}$  such that  $P_\Phi(\underline{x}) > p$

# Max-Product

$$\text{argmax}_{\underline{C, D, J, G, S, G, L, J, H}} \phi_C(C) \phi_D(C, D) \phi_I(I) \phi_G(G, I, D) \\ \phi_S(S, I) \phi_L(L, G) \phi_J(J, L, S) \phi_H(H, G, J)$$



Daphne Koller

# Max-Product

$$P(\underline{Y} \mid \underline{E} = e) = \frac{P(Y, E = e)}{P(E = e)} \propto P(Y, E = e)$$

$\underline{Y} = \{X_1, \dots, X_n\} - E$

constant wrt  $\underline{Y}$

$$P(Y, E = e) = \frac{1}{Z} \prod_k \phi'_k(D'_k)$$

partition  $\rightarrow$  constant wrt  $\underline{Y}$

$$\propto \prod_k \phi'_k(D'_k)$$

reduced relative to  $E = e$

$$\text{argmax}_{\underline{Y}} P(\underline{Y} \mid \underline{E} = e) = \text{argmax}_{\underline{Y}} \prod_k \phi'_k(D'_k)$$

# Algorithms: MAP

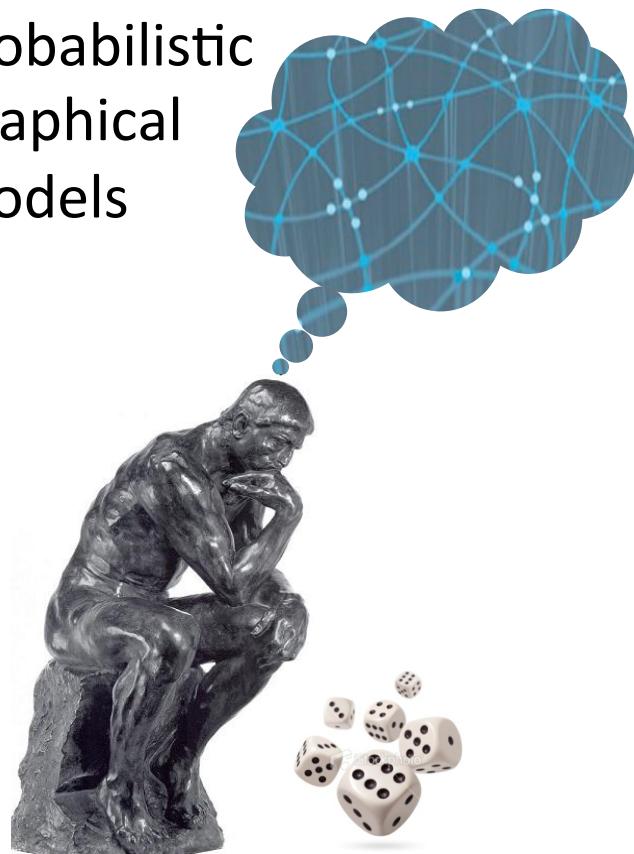
optimization

- Push maximization into factor product
  - Variable elimination      Max product
- Message passing over a graph
  - Max-product belief propagation
- Using methods for integer programming
- For some networks: graph-cut methods
- Combinatorial search

# Summary

- MAP: single coherent assignment of highest probability
  - Not the same as maximizing individual marginal probabilities
- Maxing over factor product
- Combinatorial optimization problem
- Many exact and approximate algorithms

Probabilistic  
Graphical  
Models



Inference

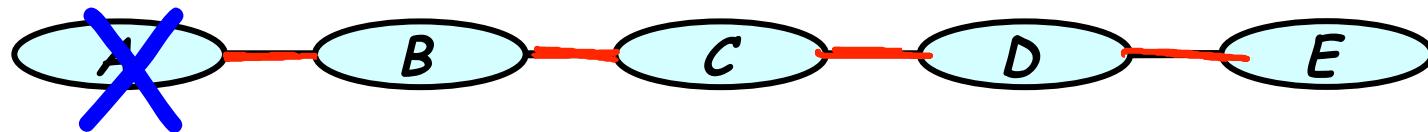
---

Variable Elimination

---

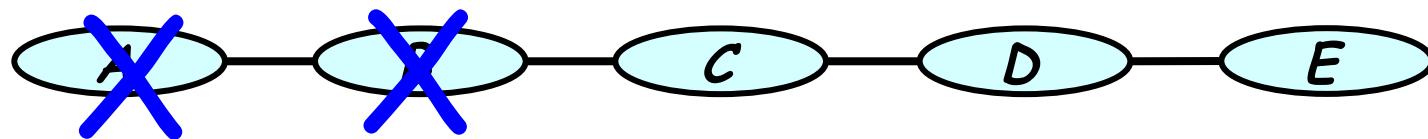
Variable  
Elimination  
Algorithm

# Elimination in Chains



$$\begin{aligned}
 \underline{P(E)} &\propto \sum_D \sum_C \sum_B \sum_A \widetilde{P}(A, B, C, D, E) \\
 &= \sum_D \sum_C \sum_B \sum_A \phi_1(A, B) \phi_2(B, C) \phi_3(C, D) \phi_4(D, E) \\
 &= \sum_D \sum_C \sum_B \phi_2(B, C) \phi_3(C, D) \phi_4(D, E) \sum_A \phi_1(A, B) \tau_1(B) \\
 &= \sum_D \sum_C \sum_B \phi_2(B, C) \phi_3(C, D) \phi_4(D, E) \tau_1(B)
 \end{aligned}$$

# Elimination in Chains



$$\begin{aligned} P(E) &\propto \sum_D \sum_C \sum_B \phi_2(B, C) \phi_3(C, D) \phi_4(D, E) \tau_1(B) \\ &= \sum_D \sum_C \phi_3(C, D) \phi_4(D, E) \left( \sum_B \phi_2(B, C) \tau_1(B) \right) \\ &= \sum_D \sum_C \phi_3(C, D) \phi_4(D, E) \tau_2(C) \end{aligned}$$

Annotations in red:

- A red curved arrow points from the term  $\sum_B \phi_2(B, C) \tau_1(B)$  to the label  $\tau_2(c)$ .
- The label  $\tau_2(c)$  is enclosed in a red bracket under the term  $\sum_B \phi_2(B, C) \tau_1(B)$ .

# Variable Elimination

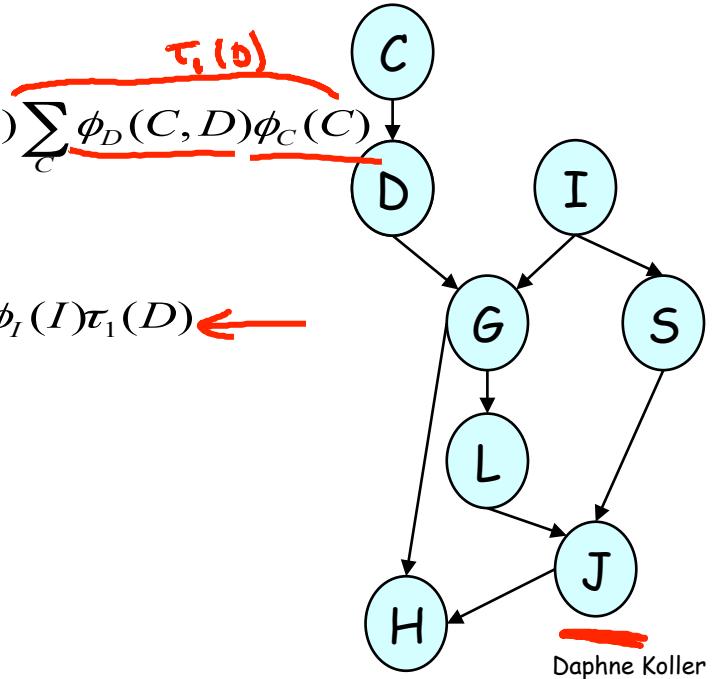
- Goal:  $P(J)$
- Eliminate: C,D,I,H,G,S,L

$$\sum_{L,S,G,H,I,D,C} \phi_J(J, L, S) \phi_L(L, G) \phi_S(S, I) \phi_G(G, I, D) \phi_H(H, G, J) \phi_I(I) \phi_D(C, D) \phi_C(C)$$

$$\sum_{L,S,G,H,I,D} \phi_J(J, L, S) \phi_L(L, G) \phi_S(S, I) \phi_G(G, I, D) \phi_H(H, G, J) \phi_I(I)$$

Compute  $\tau_1(D) = \sum_C \phi_C(C) \phi_D(C, D)$

$$= \sum_{L,S,G,H,I,D} \phi_J(J, L, S) \phi_L(L, G) \phi_S(S, I) \phi_G(G, I, D) \phi_H(H, G, J) \phi_I(I) \tau_1(D) \leftarrow$$



# Variable Elimination

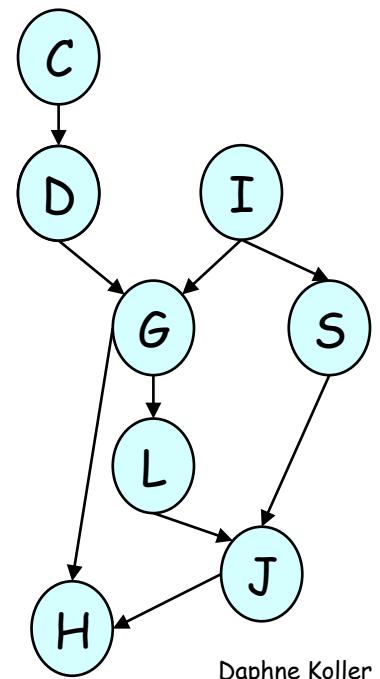
- Goal:  $P(J)$
- Eliminate: D,I,H,G,S,L

$$\sum_{L,S,G,H,I,D} \phi_J(J, L, S) \phi_L(L, G) \phi_S(S, I) \phi_G(G, I, D) \phi_H(H, G, J) \phi_I(I) \tau_1(D)$$

$$= \sum_{L,S,G,H,I} \phi_J(J, L, S) \phi_L(L, G) \phi_S(S, I) \phi_H(H, G, J) \phi_I(I) \sum_D \phi_G(G, I, D) \tau_1(D)$$

Compute  $\tau_2(G, I) = \sum_D \phi_G(G, I, D) \tau_1(D)$

$$= \sum_{L,S,G,H,I} \phi_J(J, L, S) \phi_L(L, G) \phi_S(S, I) \phi_H(H, G, J) \phi_I(I) \tau_2(G, I)$$



# Variable Elimination

- Goal:  $P(J)$
- Eliminate: I, H, G, S, L

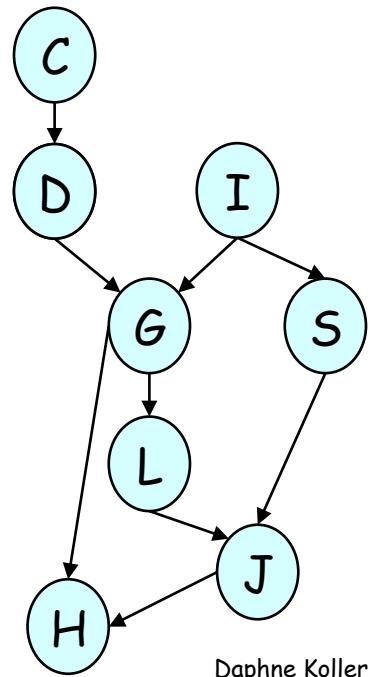
$$\sum_{L,S,G,H,I} \phi_J(J, L, S) \phi_L(L, G) \phi_S(S, I) \phi_H(H, G, J) \phi_I(I) \tau_2(G, I)$$

G, I, S

$$= \sum_{L,S,G,H} \phi_J(J, L, S) \phi_L(L, G) \phi_H(H, G, J) \sum_I \phi_S(S, I) \phi_I(I) \tau_2(G, I)$$

Compute  $\tau_3(S, G)$  =  $\sum_I \phi_S(S, I) \phi_I(I) \tau_2(G, I)$

$$= \sum_{L,S,G,H} \phi_J(J, L, S) \phi_L(L, G) \phi_H(H, G, J) \tau_3(S, G)$$



# Variable Elimination

- Goal:  $P(J)$
- Eliminate:  $H, G, S, L$

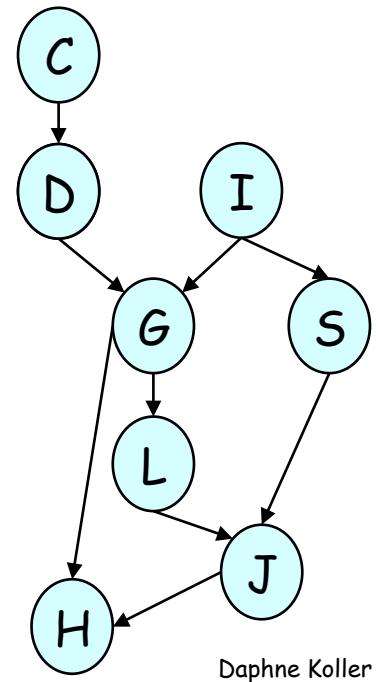
$$\sum_{L,S,G,H} \phi_J(J, L, S) \phi_L(L, G) \phi_H(H, G, J) \tau_3(S, G)$$

$$\sum_{L,S,G} \phi_J(J, L, S) \phi_L(L, G) \tau_3(S, G) \underbrace{\sum_H \phi_H(H, G, J)}_{\tau_4(G, J)}$$

$\cancel{\sum_H \phi_H(H, G, J) = 1}$

Compute  $\tau_4(G, J) = \sum_H \phi_H(H, G, J)$

$$\sum_{L,S,G} \phi_J(J, L, S) \phi_L(L, G) \tau_3(S, G) \tau_4(G, J)$$



# Variable Elimination

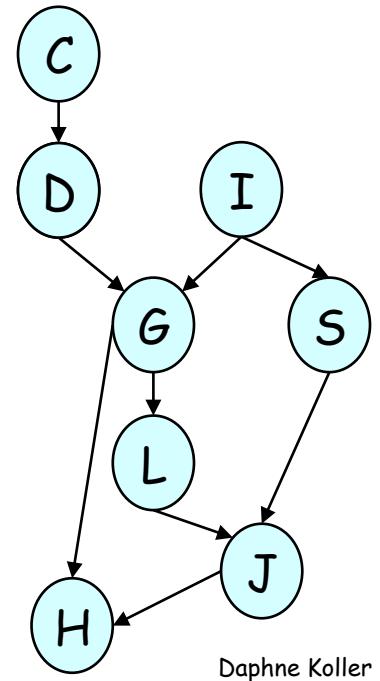
- Goal:  $P(J)$
- Eliminate:  $G, S, L$

$$\sum_{L,S,G} \phi_J(J, L, S) \phi_L(L, G) \tau_3(S, G) \tau_4(G, J)$$

$$\sum_{L,S} \phi_J(J, L, S) \sum_G \phi_L(L, G) \tau_4(G, J) \tau_3(S, G)$$

Compute  $\tau_5(L, J) = \sum_G \phi_L(L, G) \tau_3(S, G) \tau_4(G, J)$

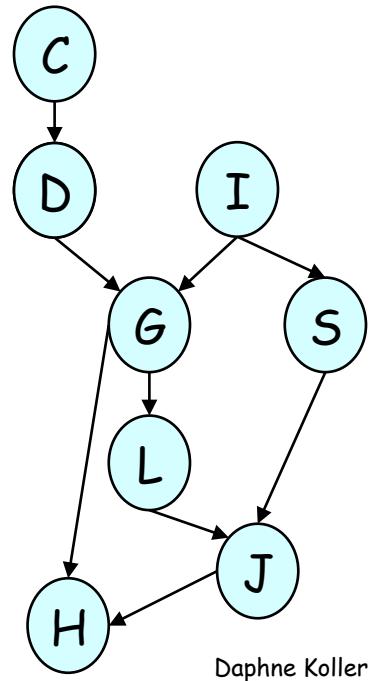
$$\sum_{L,S} \phi_J(J, L, S) \tau_5(L, J)$$



# Variable Elimination

- Goal:  $P(J)$
- Eliminate:  $S, L$

$$\sum_{L,S} \phi_J(J, L, S) \tau_S(L, J)$$



# Variable Elimination with evidence

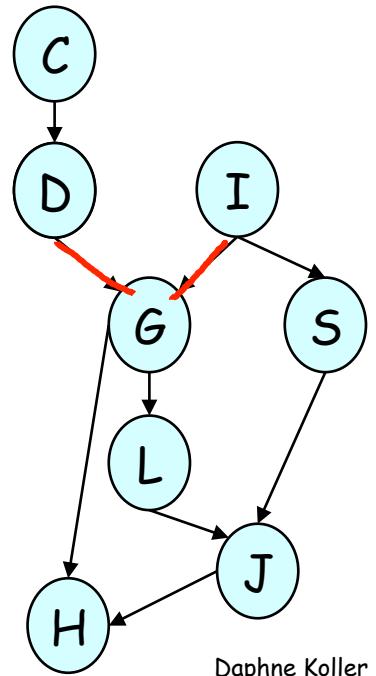
- Goal:  $P(J, I=i, H=h)$
- Eliminate:  $C, D, G, S, L \rightsquigarrow H, I$   
 $P(I=i, D) \Phi_I(\cdot)$

$$\sum_{L, S, G, D, C} \phi_J(J, L, S) \phi_L(L, G) \phi_S(S) \underbrace{\phi_G(G, D)}_{\Phi_G(\cdot)} \phi_H(H, J) \underbrace{\phi_D(D)}_{\Phi_D(\cdot)} \phi_C(C)$$

elimination as before

How do we get  $P(J | I=i, H=h)$ ?

normalize  
 $P(I=i, H=h)$  is the normalizing constant



# Variable Elimination in MNs

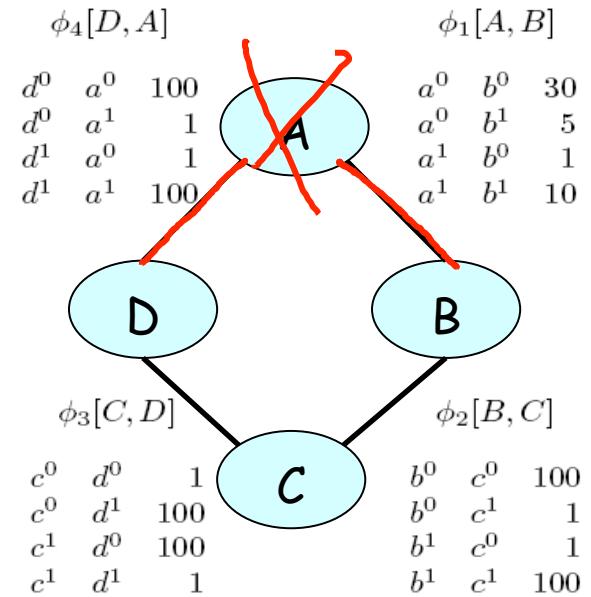
- Goal:  $P(D)$
- Eliminate:  $A, B, C$

$$\sum_{A,B,C} \phi_1(A, B) \phi_2(B, C) \phi_3(C, D) \phi_4(A, D)$$

$$\sum_{B,C} \phi_2(B, C) \phi_3(C, D) \sum_A \phi_1(A, B) \phi_4(A, D)$$

$$\sum_{B,C} \phi_2(B, C) \phi_3(C, D) \tau_1(B, D)$$

$= \tilde{P}(D)$   
 At the end of elimination get  $\underline{\tau_3(D)} \propto P(D)$   
 renormalize



# Eliminate-Var Z from $\Phi$

$$\underline{\Phi'} = \{\phi_i \in \Phi : \underline{Z \in \text{Scope}[\phi_i]}\}$$

all factors that involve z

$$\psi = \prod_{\phi_i \in \Phi'} \phi_i$$

multiply them

$$\tau = \sum_Z \psi$$

sum out z

$$\Phi := \Phi - \underline{\Phi'} \cup \underline{\{\tau\}}$$

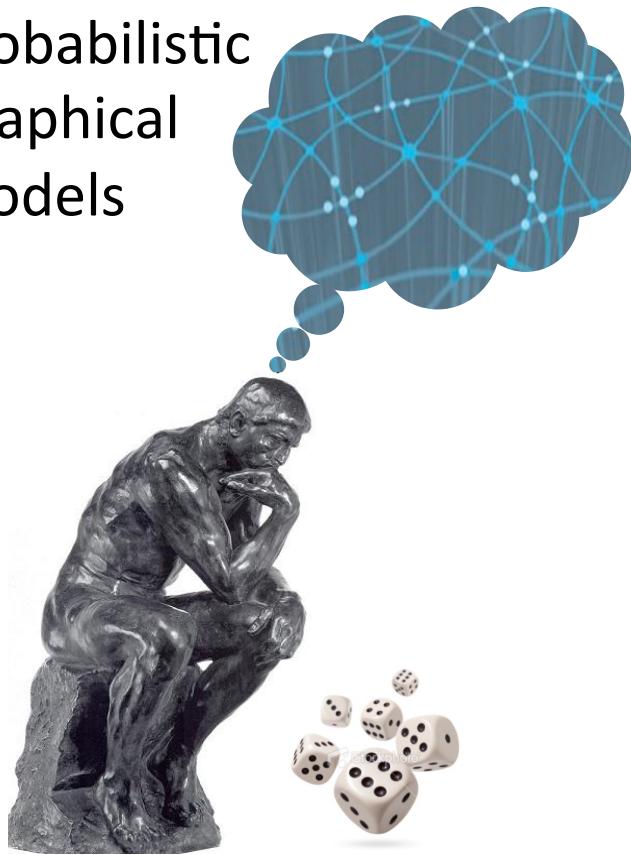
# VE Algorithm Summary

- Reduce all factors by evidence
  - Get a set of factors  $\underline{\Phi}$
- For each non-query variable  $Z$ 
  - Eliminate-Var  $Z$  from  $\underline{\Phi}$  adds removes  $\underline{\Phi}'$   
adds  $\underline{z}$
- Multiply all remaining factors
- Renormalize to get distribution

# Summary

- Simple algorithm
- Works for both BNs and MNs
- Factor product and summation steps can be done in any order, subject to:
  - when Z is eliminated, all factors involving Z have been multiplied in

Probabilistic  
Graphical  
Models



Inference

---

Variable Elimination

---

Complexity  
Analysis

# Eliminating $Z$

$$\psi_k(\mathbf{X}_k) = \prod_{i=1}^{m_k} \phi_i \quad \text{factor product}$$

$$\tau_k(\mathbf{X}_k - \{Z\}) = \sum_Z \psi_k(\mathbf{X}_k) \quad \text{marginalization}$$

# Reminder: Factor Product

$$\psi_k(\mathbf{X}_k) = \prod_{i=1}^{m_k} \phi_i$$

*each row  
 $m_{k-1}$  products*

$$N_k = |\text{Val}(\mathbf{X}_k)|$$

a <sup>1</sup>	b <sup>1</sup>	0.5
a <sup>1</sup>	b <sup>2</sup>	0.8
a <sup>2</sup>	b <sup>1</sup>	0.1
a <sup>2</sup>	b <sup>2</sup>	0
a <sup>3</sup>	b <sup>1</sup>	0.3
a <sup>3</sup>	b <sup>2</sup>	0.9

b <sup>1</sup>	c <sup>1</sup>	0.5
b <sup>1</sup>	c <sup>2</sup>	0.7
b <sup>2</sup>	c <sup>1</sup>	0.1
b <sup>2</sup>	c <sup>2</sup>	0.2

*B C*



a <sup>1</sup>	b <sup>1</sup>	c <sup>1</sup>	0.5 · 0.5 = 0.25
a <sup>1</sup>	b <sup>1</sup>	c <sup>2</sup>	0.5 · 0.7 = 0.35
a <sup>1</sup>	b <sup>2</sup>	c <sup>1</sup>	0.8 · 0.1 = 0.08
a <sup>1</sup>	b <sup>2</sup>	c <sup>2</sup>	0.8 · 0.2 = 0.16
a <sup>2</sup>	b <sup>1</sup>	c <sup>1</sup>	0.1 · 0.5 = 0.05
a <sup>2</sup>	b <sup>1</sup>	c <sup>2</sup>	0.1 · 0.7 = 0.07
a <sup>2</sup>	b <sup>2</sup>	c <sup>1</sup>	0 · 0.1 = 0
a <sup>2</sup>	b <sup>2</sup>	c <sup>2</sup>	0 · 0.2 = 0
a <sup>3</sup>	b <sup>1</sup>	c <sup>1</sup>	0.3 · 0.5 = 0.15
a <sup>3</sup>	b <sup>1</sup>	c <sup>2</sup>	0.3 · 0.7 = 0.21
a <sup>3</sup>	b <sup>2</sup>	c <sup>1</sup>	0.9 · 0.1 = 0.09
a <sup>3</sup>	b <sup>2</sup>	c <sup>2</sup>	0.9 · 0.2 = 0.18

Cost:  $(m_k - 1)N_k$  multiplications

# Reminder: Factor Marginalization

$$\tau_k(\underline{X}_k - \{Z\}) = \sum_Z \psi_k(\underline{X}_k)$$

$$N_k = |\text{Val}(\underline{X}_k)|$$

Cost:  $\sim N_k$  additions

each number used exactly once in marg B

The diagram illustrates the process of factor marginalization. It shows two tables. The left table has columns labeled  $a^1, b^1, c^1$  and a row of values  $0.25, 0.35, 0.08, 0.16, 0.05, 0.07, 0, 0, 0.15, 0.21, 0.09, 0.18$ . The right table has columns labeled  $a^1, c^1$  and a row of values  $0.33, 0.51, 0.05, 0.07, 0.24, 0.39$ . Red lines connect specific rows from the left table to specific columns in the right table, with a note above stating "each number used exactly once in marg B".

$a^1$	$b^1$	$c^1$	0.25
$a^1$	$b^1$	$c^2$	0.35
$a^1$	$b^2$	$c^1$	0.08
$a^1$	$b^2$	$c^2$	0.16
$a^2$	$b^1$	$c^1$	0.05
$a^2$	$b^1$	$c^2$	0.07
$a^2$	$b^2$	$c^1$	0
$a^2$	$b^2$	$c^2$	0
$a^3$	$b^1$	$c^1$	0.15
$a^3$	$b^1$	$c^2$	0.21
$a^3$	$b^2$	$c^1$	0.09
$a^3$	$b^2$	$c^2$	0.18

$a^1$	$c^1$	0.33
$a^1$	$c^2$	0.51
$a^2$	$c^1$	0.05
$a^2$	$c^2$	0.07
$a^3$	$c^1$	0.24
$a^3$	$c^2$	0.39

# Complexity of Variable Elimination

- Start with  $m$  factors
  - $m \leq n$  for Bayesian networks *(one for every variable)*
  - can be larger for Markov networks
- At each elimination step generate  $1$  factor,
- At most  $n$  elimination steps
- Total number of factors:  $m^* \leq m + n$

# Complexity of Variable Elimination

- $\underline{N} = \max(N_k) = \text{size of the largest factor}$
- Product operations:  $\sum_k (m_k - 1)N_k \leq N \sum_k (m_k - 1)$   
each factor multiply in at most one  
 $N_{\text{mt}} \leq m^*$
- Sum operations:  $\leq \sum_k N_k \leq N \cdot \# \text{elimination steps} \leq \underline{N \cdot n}$
- Total work is linear in  $\underline{N}$  and  $m^*$

# Complexity of Variable Elimination

- Total work is linear in N and m exponential blowup
- $N_k = |\text{Val}(X_k)| = O(d^{r_k})$  where # variables in k<sup>th</sup> factor
  - $d = \max(|\text{Val}(X_i)|)$  d values in their scope
  - $r_k = |X_k|$  = cardinality of the scope of the k<sup>th</sup> factor

# Complexity Example

$$\tau_1(D) = \sum_C \phi_C(C) \phi_D(C, D)$$
2

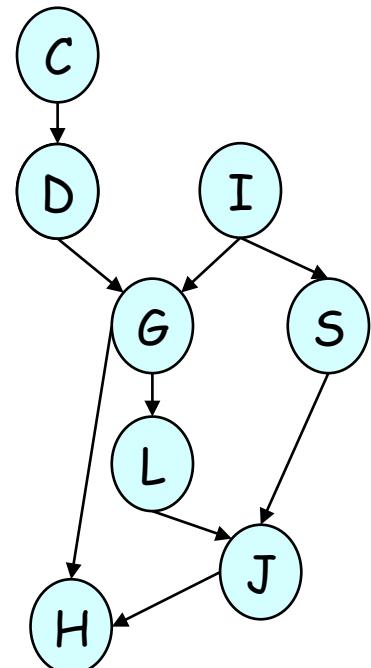
$$\tau_2(G, I) = \sum_D \phi_G(G, I, D) \tau_1(D)$$
3

$$\tau_3(S, G) = \sum_I \phi_S(S, I) \phi_I(I) \tau_2(G, I)$$
3

$$\tau_4(G, J) = \sum_H \phi_H(H, G, J)$$
3

$$\tau_5(J, L, S) = \sum_G \phi_L(L, G) \tau_3(S, G) \tau_4(G, J)$$
4 ←

$$\tau_6(J) = \sum_{L, S} \phi_J(J, L, S) \tau_5(J, L, S)$$
3



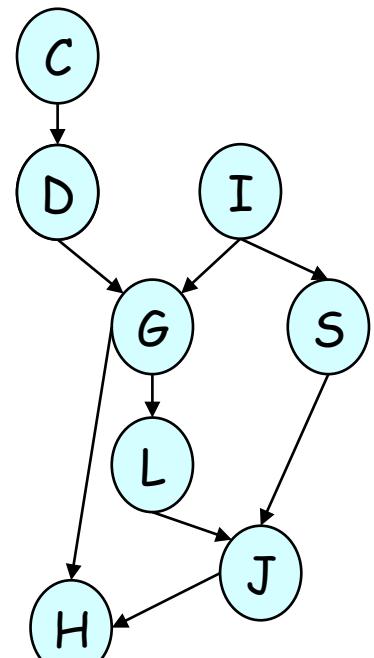
# Complexity and Elimination Order

$$\sum_{L,S,G,H,I,D,C} \phi_J(J, L, S) \underbrace{\phi_L(L, G)}_{6} \phi_S(S, I) \underbrace{\phi_G(G, I, D)}_{6} \underbrace{\phi_H(H, G, J)}_{6} \phi_I(I) \phi_D(C, D) \phi_C(C)$$

- Eliminate:  $G$

$\overbrace{L, G, I, D, H, J}^6$

$$\sum_G \phi_L(L, G) \phi_G(G, I, D) \phi_H(H, G, J)$$



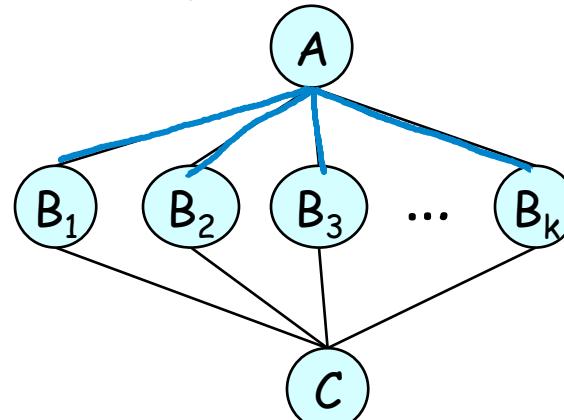
Daphne Koller

# Complexity and Elimination Order

Eliminate A first:

$$\{A, B_1, \dots, B_k\}$$

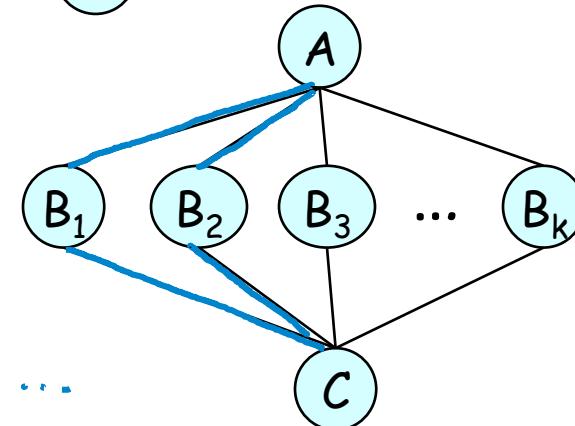
size of factor is exp in k



$$\prod_i \tau_i(A, c)$$

Eliminate Bi's first:

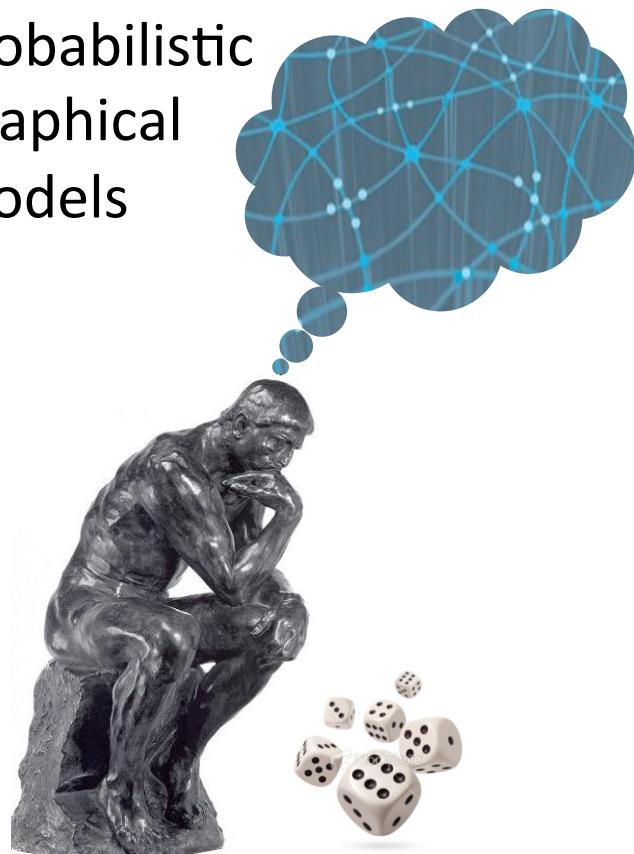
$$\underbrace{\phi_{i+1}(A, B_i) \cdot \phi_{i+1}(C, B_i)}_{\text{Scope } A, B_i, C} \Rightarrow \tau_i(A, c) \quad \tau_2(A, c) \dots$$



# Summary

- Complexity of variable elimination linear in
  - size of the model (# factors, # variables)
  - size of the largest factor generated
- Size of factor is exponential in its scope
- Complexity of algorithm depends heavily on elimination ordering

Probabilistic  
Graphical  
Models



Inference

---

Variable Elimination

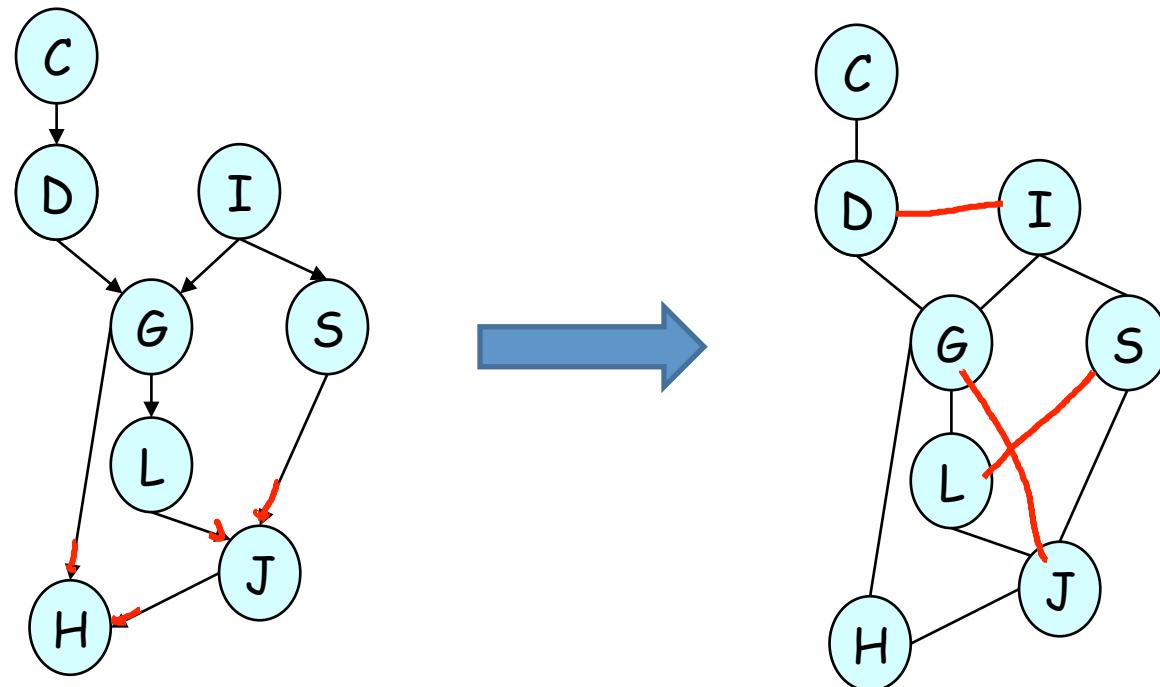
---

Graph-Based  
Perspective

# Initial Graph

$\phi_J(J, L, S) \phi_L(L, G) \phi_S(S, I) \phi_G(G, I, D) \phi_H(H, G, J) \phi_I(I) \phi_D(C, D) \phi_C(C)$

*moralization*



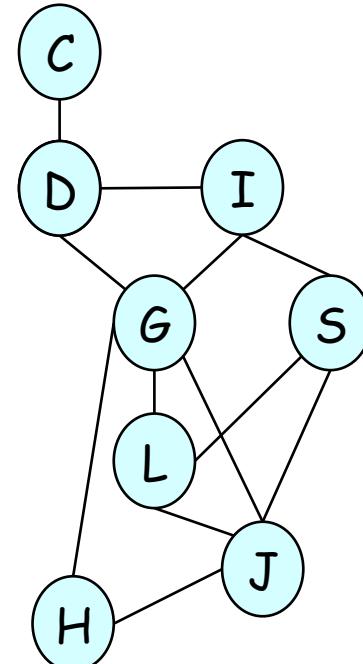
Daphne Koller

# Elimination as Graph Operation

$$\phi_J(J, L, S) \phi_L(L, G) \phi_S(S, I) \phi_G(G, I, D) \phi_H(H, G, J) \phi_I(I) \cancel{\phi_D(C, D)} \cancel{\phi_C(C)}$$

- Eliminate: C

$$\underline{\tau_1(D)} = \sum_C \phi_C(C) \phi_D(C, D)$$



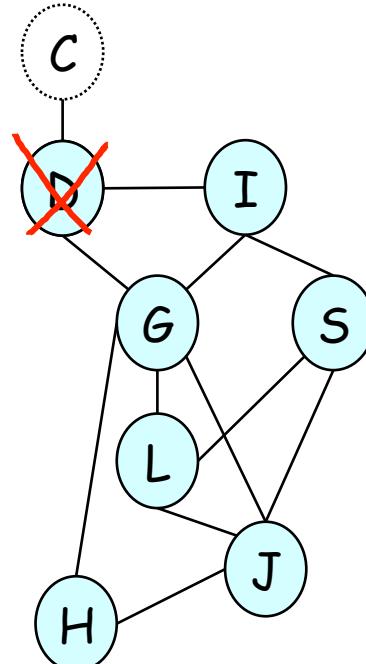
Induced Markov network for the current set of factors

# Elimination as Graph Operation

$$\phi_J(J, L, S) \phi_L(L, G) \phi_S(S, I) \underline{\phi_G(G, I, D)} \phi_H(H, G, J) \phi_I(I) \underline{\tau_1(D)}$$

- Eliminate: D

$$\tau_2(G, I) = \sum_D \phi_G(G, I, D) \tau_1(D)$$



Induced Markov network for the current set of factors

Daphne Koller

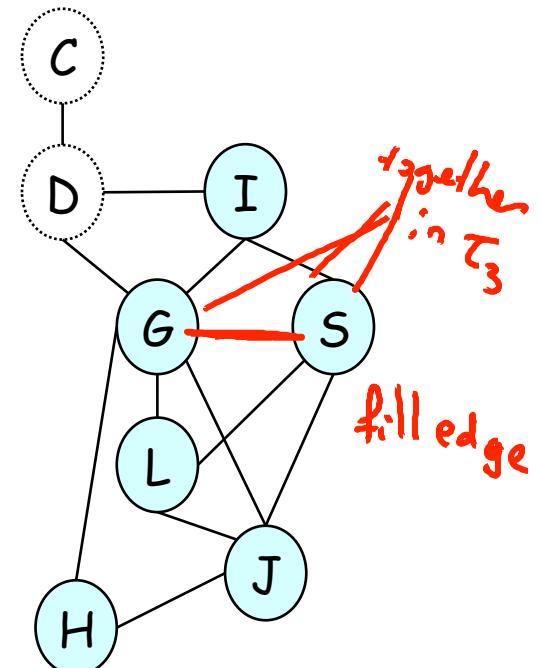
# Elimination as Graph Operation

$$\phi_J(J, L, S) \phi_L(L, G) \underline{\phi_S(S, I)} \underline{\phi_I(I)} \phi_H(H, G, J) \underline{\tau_2(G, I)}$$

- Eliminate: I

$$\tau_3(S, G) = \sum_I \phi_S(S, I) \phi_I(I) \tau_2(G, I)$$

all variables connected to I become  
connected directly



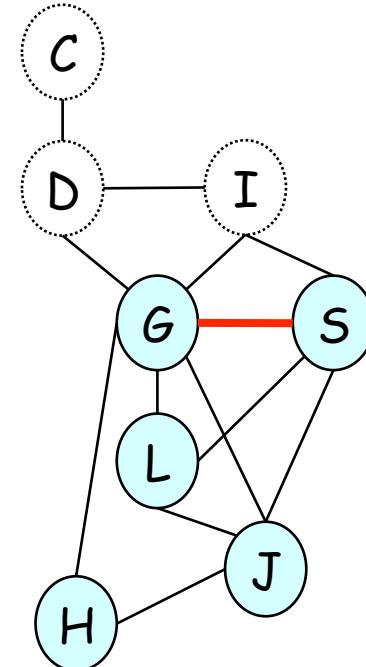
Induced Markov network for the current set of factors

# Elimination as Graph Operation

$$\phi_J(J, L, S) \phi_L(L, G) \phi_H(H, G, J) \tau_3(S, G)$$

- Eliminate: H

$$\tau_4(G, J) = \sum_H \phi_H(H, G, J)$$



Induced Markov network for the current set of factors

Daphne Koller

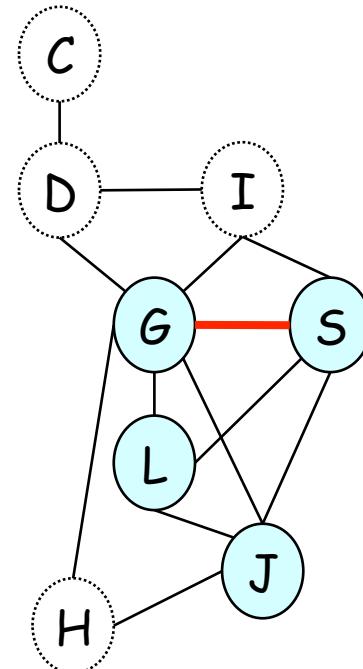
# Elimination as Graph Operation

$$\phi_J(J, L, S) \phi_L(L, G) \tau_3(S, G) \tau_4(G, J)$$

- Eliminate:  $G$

$$\tau_5(L, J) = \sum_G \phi_L(L, G) \tau_3(S, G) \tau_4(G, J)$$

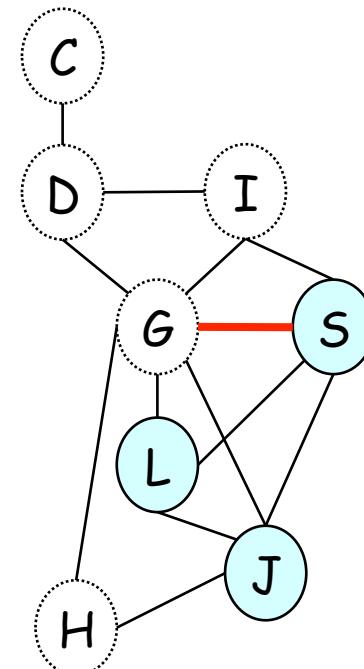
Induced Markov network for the current set of factors



# Elimination as Graph Operation

$$\phi_J(J, L, S) \tau_5(L, J)$$

- Eliminate: L, S



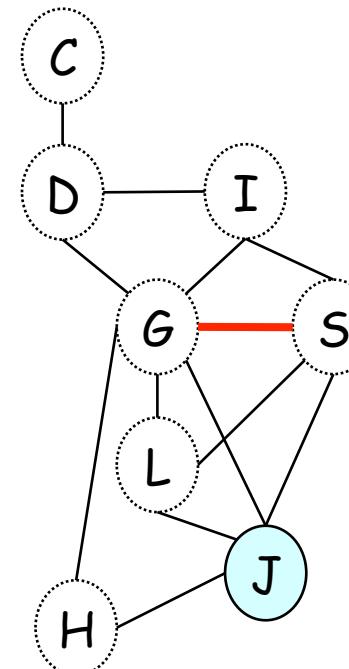
Induced Markov network for the current set of factors

Daphne Koller

# Elimination as Graph Operation

$$\phi_J(J, L, S) \tau_5(L, J)$$

- Eliminate: L, S

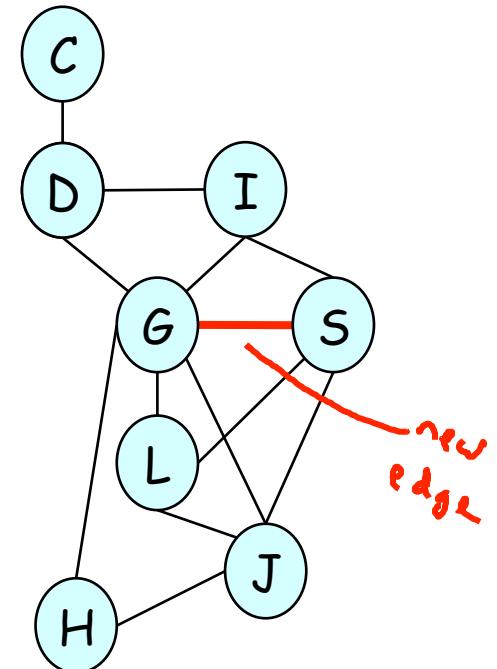


Induced Markov network for the current set of factors

Daphne Koller

# Induced Graph

- The induced graph  $I_{\Phi, \alpha}$  over factors  $\Phi$  and ordering  $\alpha$ :
  - Undirected graph
  - $X_i$  and  $X_j$  are connected if they appeared in the same factor in a run of the VE algorithm using  $\alpha$  as the ordering



Daphne Koller

# Cliques in the Induced Graph

*maximal fully connected subgraph*

- ~~Theorem~~: Every factor produced during VE is a clique in the induced graph

$$\tau_1(D) = \sum_C \phi_C(C) \phi_D(C, D)$$

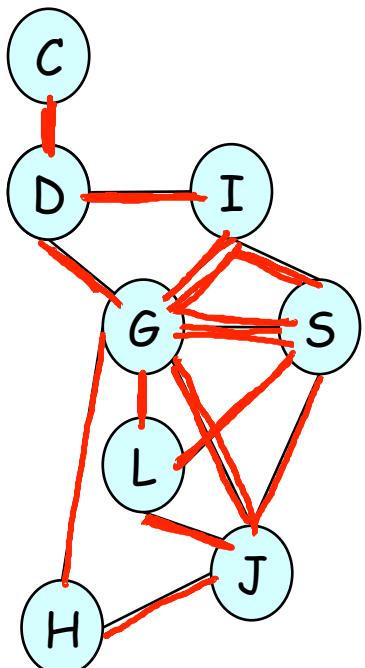
$$\tau_2(G, I) = \sum_D \phi_G(G, I, D) \tau_1(D)$$

$$\tau_3(S, G) = \sum_I \phi_S(S, I) \phi_I(I) \tau_2(G, I)$$

$$\tau_4(G, J) = \sum_H \phi_H(H, G, J)$$

$$\tau_5(L, J) = \sum_G \phi_L(L, G) \tau_3(S, G) \tau_4(G, J)$$

$$\tau_6 = \sum_{L, S} \phi_J(J, L, S) \tau_5(L, J)$$



Daphne Koller

# Cliques in the Induced Graph

- **Theorem:** Every (maximal) clique in the induced graph is a factor produced during VE

$$\tau_1(D) = \sum_C \phi_C(C) \phi_D(C, D)$$

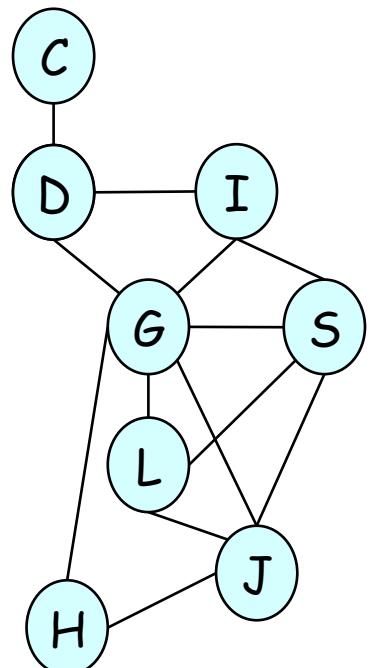
$$\tau_2(G, I) = \sum_D \phi_G(G, I, D) \tau_1(D)$$

$$\tau_3(S, G) = \sum_I \phi_S(S, I) \phi_I(I) \tau_2(G, I)$$

$$\tau_4(G, J) = \sum_H \phi_H(H, G, J)$$

$$\tau_5(L, J) = \sum_G \phi_L(L, G) \tau_3(S, G) \tau_4(G, J)$$

$$\tau_6 = \sum_{L, S} \phi_J(J, L, S) \tau_5(L, J)$$



Daphne Koller

# Cliques in the Induced Graph

- **Theorem:** Every (maximal) clique in the induced graph is a factor produced during VE

Consider a max clique -

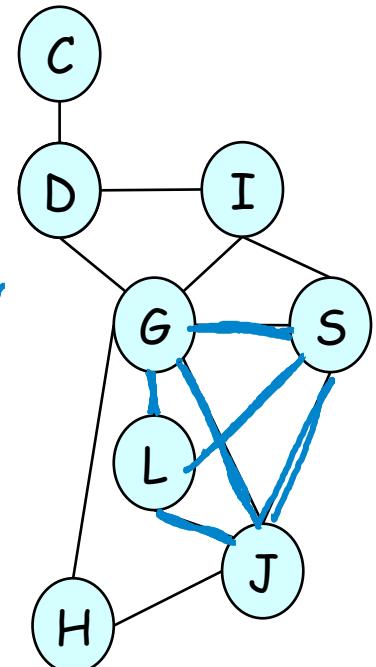
some variable is first to be eliminated

once a variable is eliminated - no new neighbor

⇒ when eliminated it already had all the  
clique members as neighbors

⇒ participated in factors with all these other variables

⇒ when multiplied together, we have a factor  
over all of them



Daphne Koller

# Induced Width

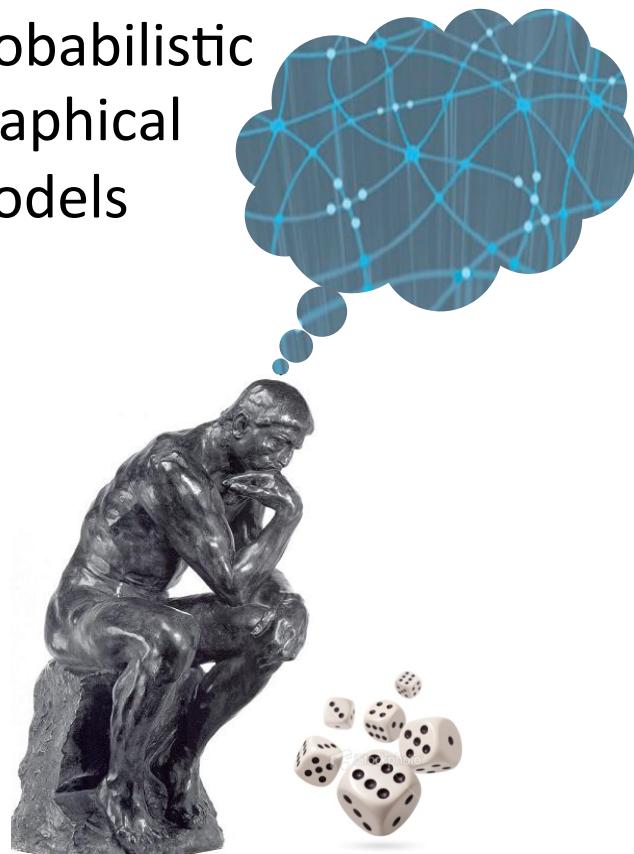
- The width of an induced graph is the number of nodes in the largest clique in the graph minus 1
- Minimal induced width of a graph  $K$  is  $\min_{\alpha} (\underline{\text{width}}(I_{K,\alpha}))$
- Provides a lower bound on best performance of VE to a model factorizing over  $K$

# Summary

- Variable elimination can be viewed as transformations on undirected graph
  - Elimination connects all node's current neighbors
- Cliques in resulting induced graph directly correspond to algorithm's complexity



Probabilistic  
Graphical  
Models



Inference

---

Variable Elimination

---

Finding  
Elimination  
Orderings

# Finding Elimination Orderings

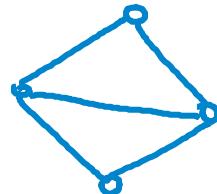
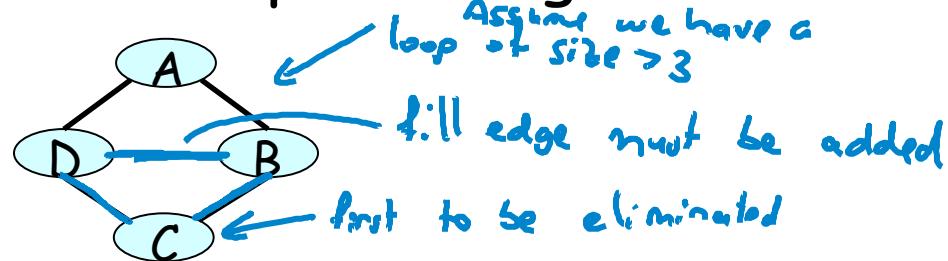
- **Theorem:** For a graph  $H$ , determining whether there exists an elimination ordering for  $H$  with induced width  $\leq K$  is NP-complete
- **Note:** This NP-hardness result is distinct from the NP-hardness result of inference
  - Even given the optimal ordering, inference may still be exponential

# Finding Elimination Orderings

- Greedy search using heuristic cost function
  - At each point, eliminate node with smallest cost
- Possible cost functions:
  - min-neighbors: # neighbors in current graph
  - min-weight: weight (# values) of factor formed
  - min-fill: number of new fill edges
  - weighted min-fill: total weight of new fill edges  
(edge weight = product of weights of the 2 nodes)  
smallest factor

# Finding Elimination Orderings

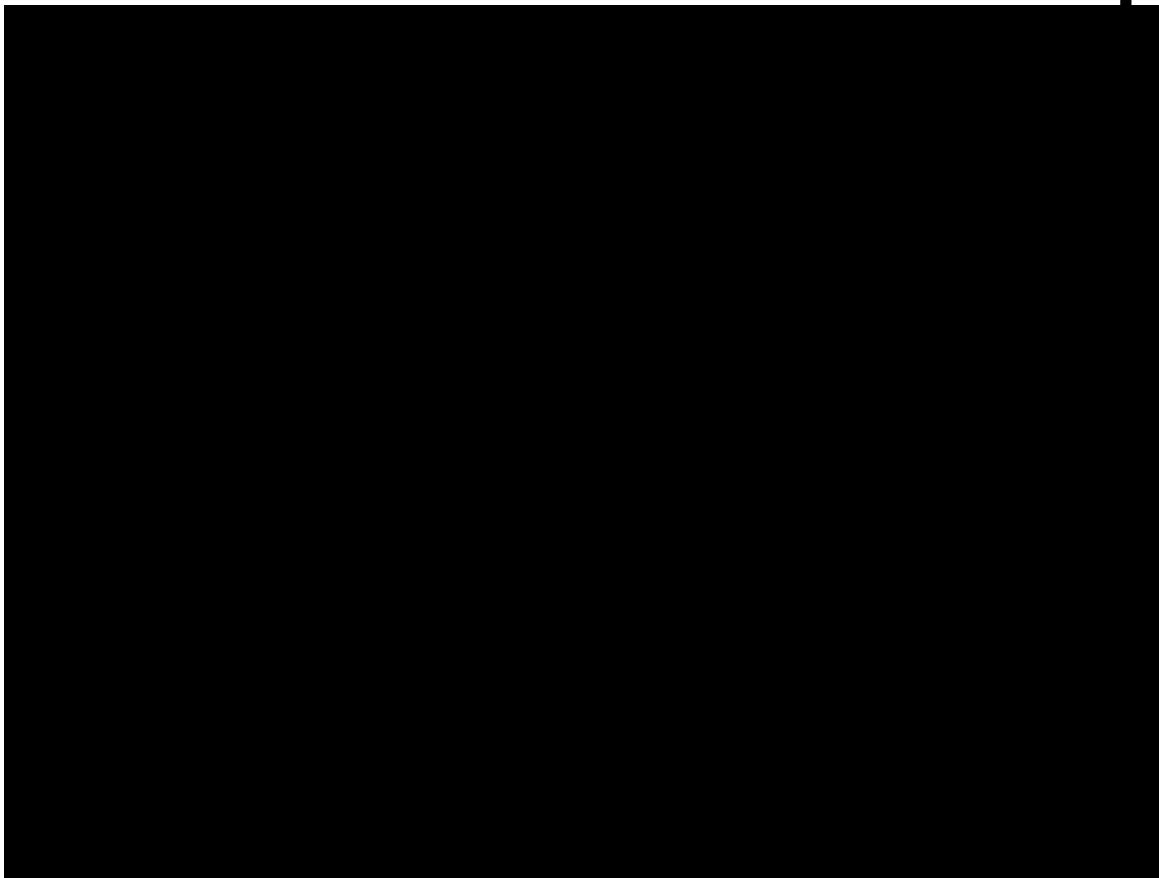
- **Theorem:** The induced graph is triangulated
  - No loops of length  $> 3$  without a "bridge"



- Can find elimination ordering by finding a low-width triangulation of original graph  $H_\Phi$

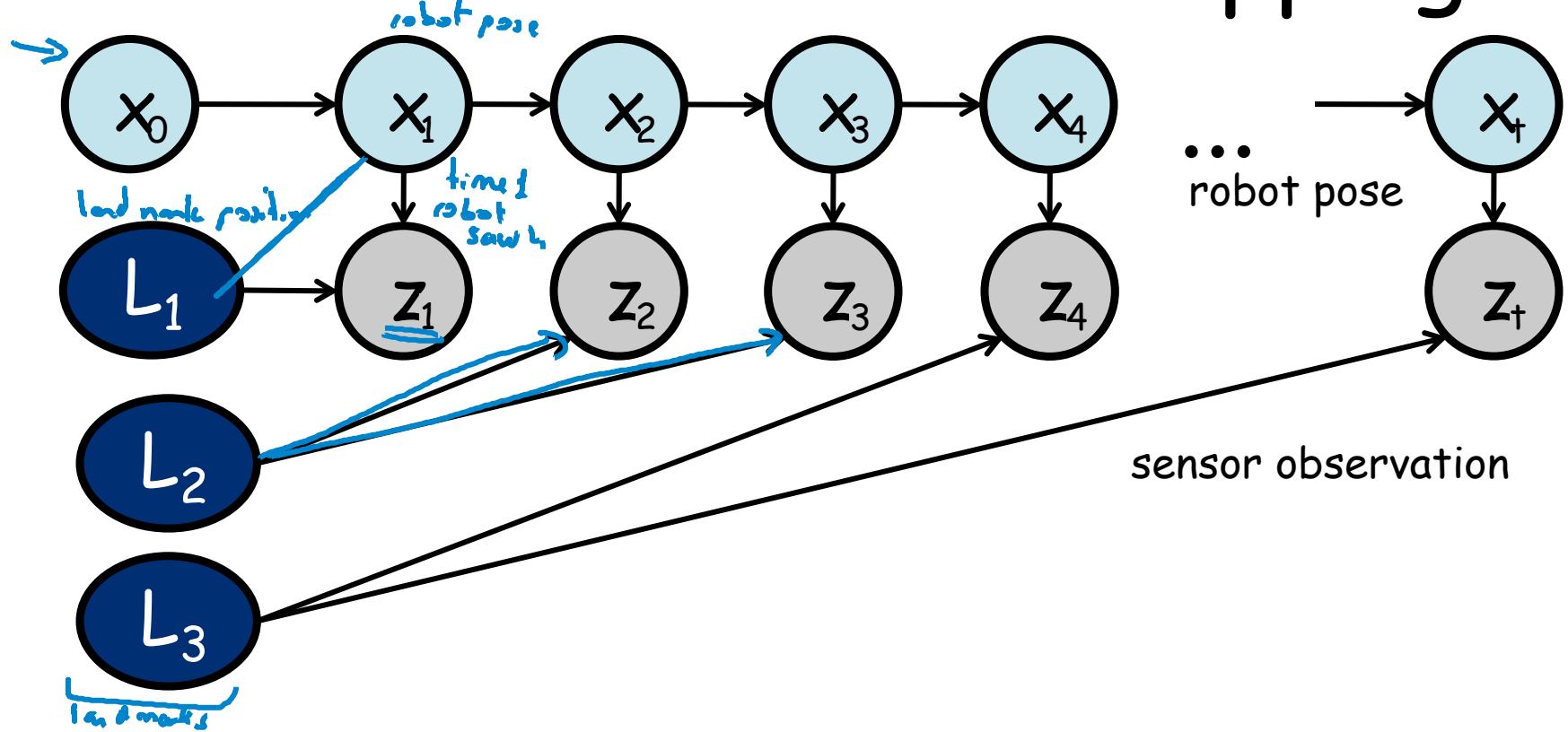
Square Root SAM, F. Dellaert and M. Kaess, IJRR, 2006

# Robot Localization & Mapping



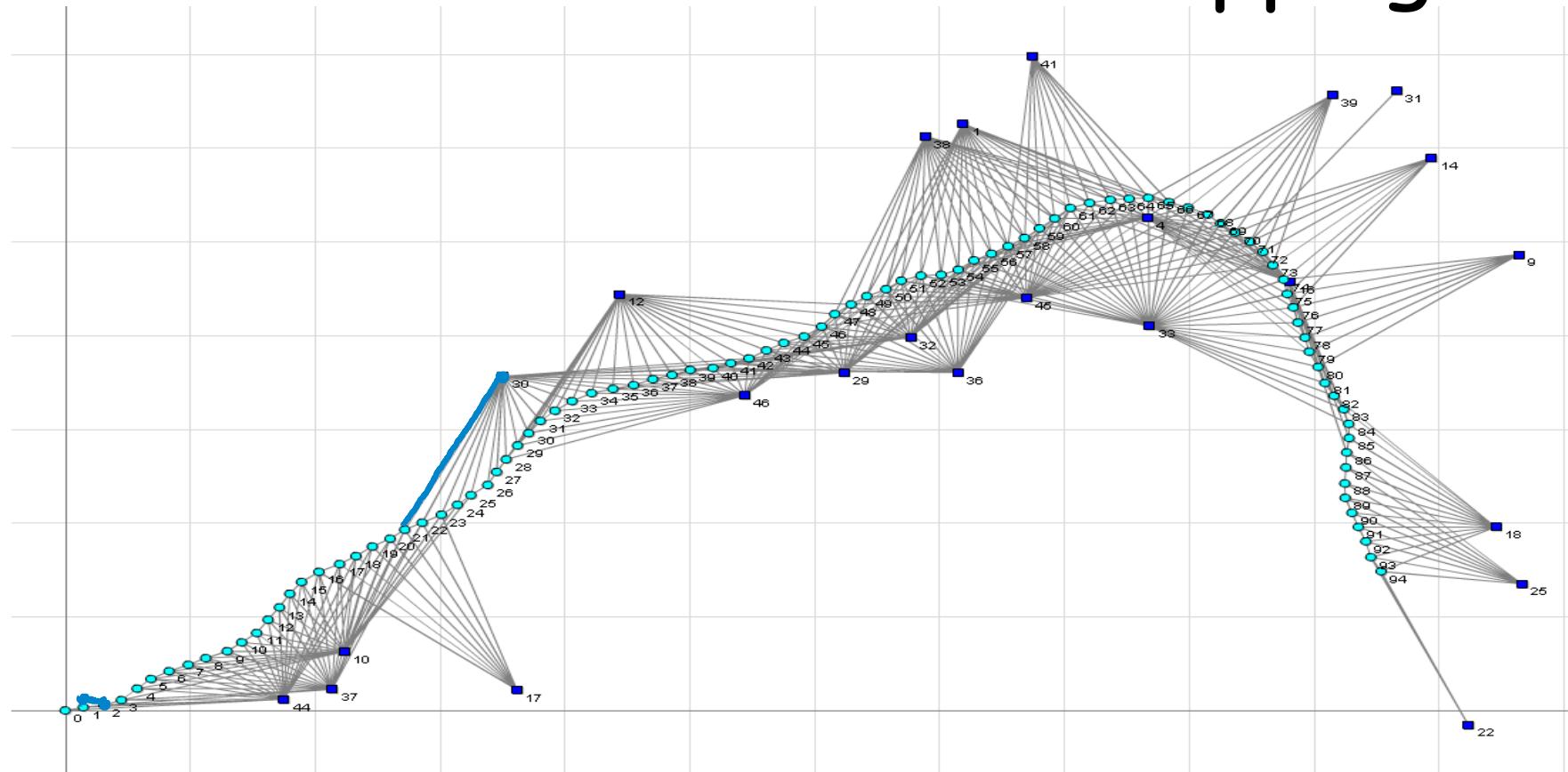
Daphne Koller

# Robot Localization & Mapping



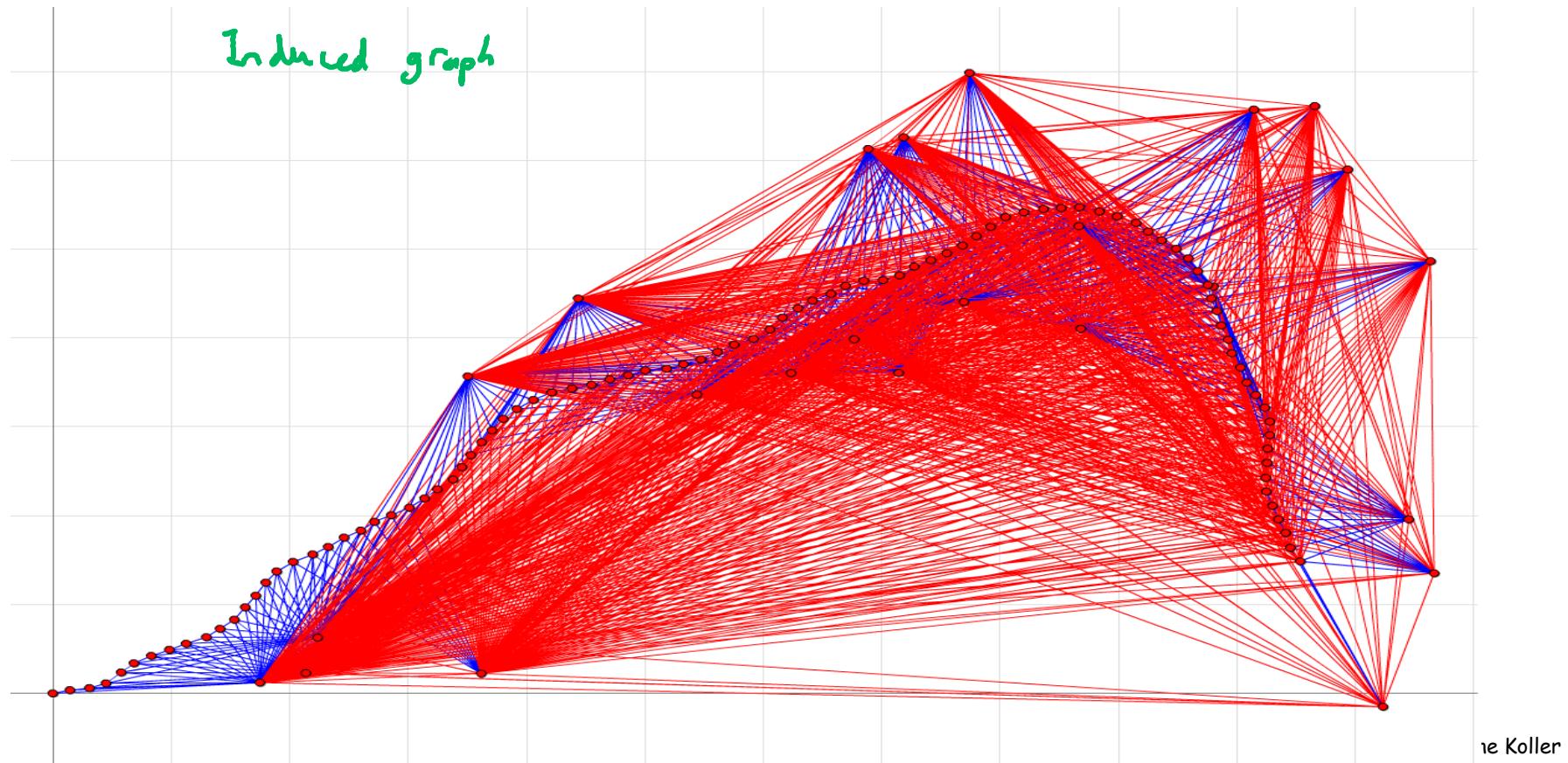
Square Root SAM, F. Dellaert and M. Kaess, IJRR, 2006

# Robot Localization & Mapping



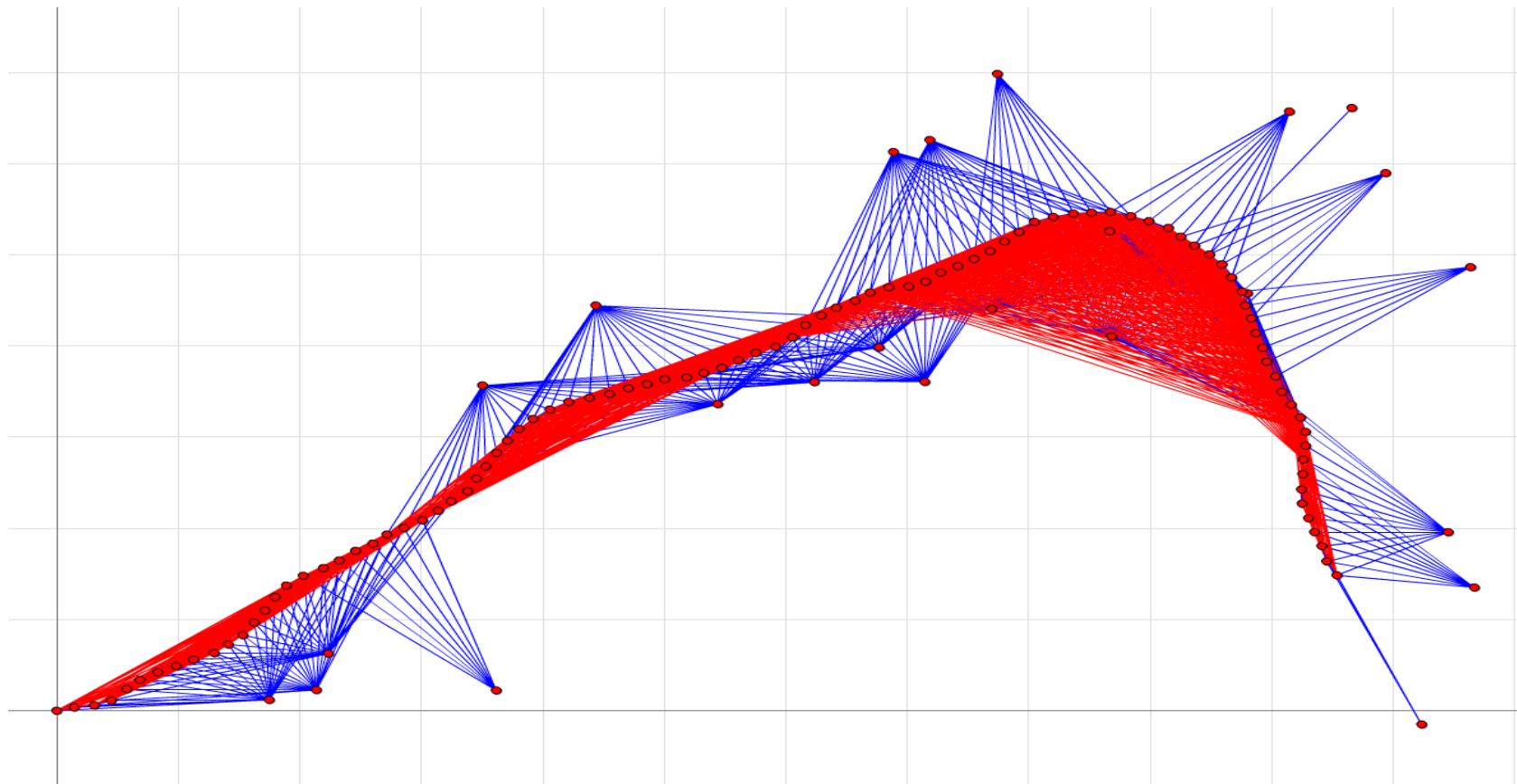
Square Root SAM, F. Dellaert and M. Kaess, IJRR, 2006

# Eliminate Poses then Landmarks



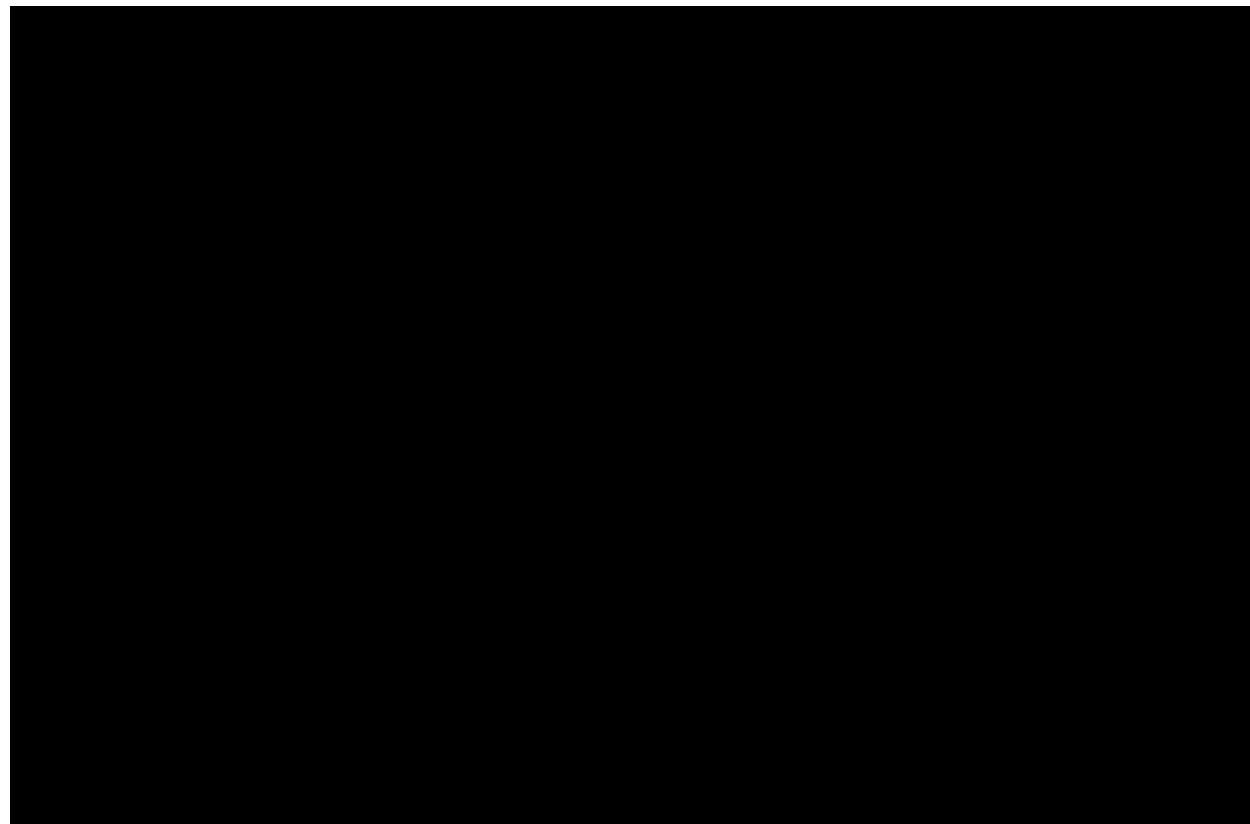
Square Root SAM, F. Dellaert and M. Kaess, IJRR, 2006

# Eliminate Landmarks then Poses



Square Root SAM, F. Dellaert and M. Kaess, IJRR, 2006

# Min-Fill Elimination



Daphne Koller

# Summary

- Finding the optimal elimination ordering is NP-hard
- Simple heuristics that try to keep induced graph small often provide reasonable performance