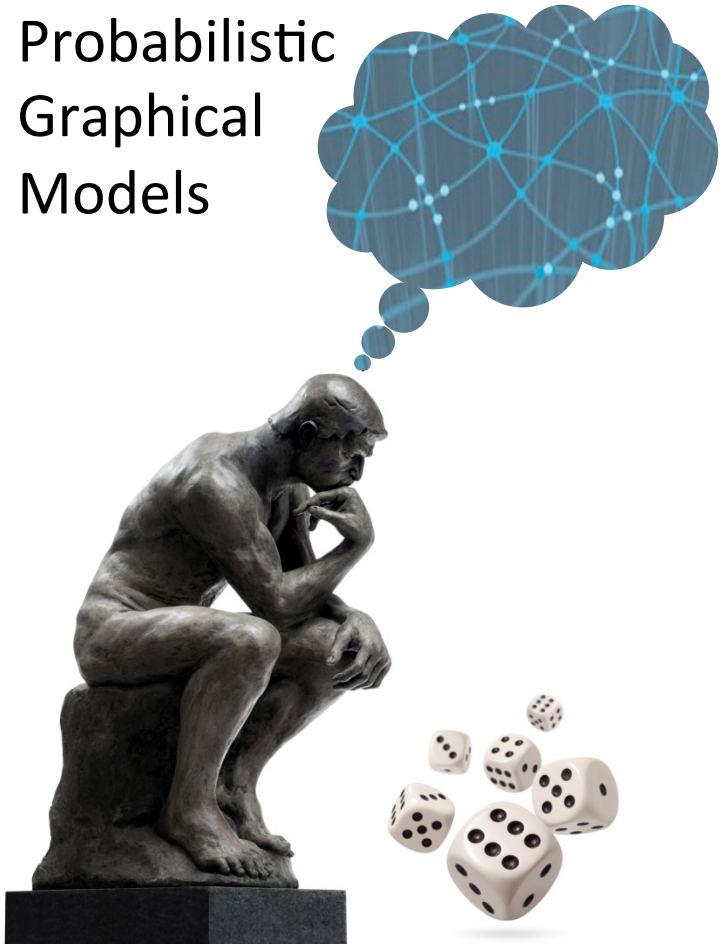
Probabilistic Graphical Models

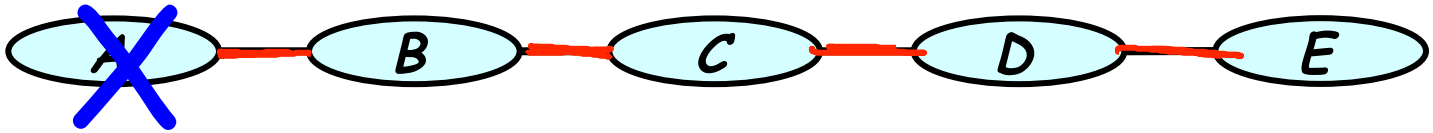


Inference

Variable Elimination

Variable Elimination Algorithm

Elimination in Chains



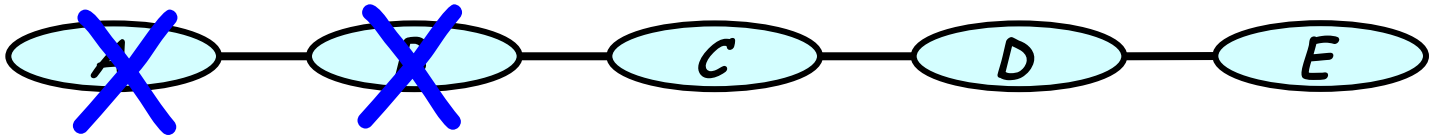
$$P(E) \propto \sum_D \sum_C \sum_B \sum_A \tilde{P}(A,B,C,D,E)$$

$$= \sum_D \sum_C \sum_B \sum_A \phi_1(A,B)\phi_2(B,C)\phi_3(C,D)\phi_4(D,E)$$

$$= \sum_D \sum_C \sum_B \phi_2(B,C)\phi_3(C,D)\phi_4(D,E) \sum_A \phi_1(A,B) \quad \tau_1(B)$$

$$= \sum_D \sum_C \sum_B \phi_2(B,C)\phi_3(C,D)\phi_4(D,E)\tau_1(B)$$

Elimination in Chains



$$P(E) \propto \sum_D \sum_C \sum_B \phi_2(B,C)\phi_3(C,D)\phi_4(D,E)\tau_1(B)$$

$$= \sum_D \sum_C \phi_3(C,D)\phi_4(D,E)\left(\sum_B \phi_2(B,C)\tau_1(B)\right)$$

$$= \sum_D \sum_C \phi_3(C,D)\phi_4(D,E)\tau_2(C)$$

Variable Elimination

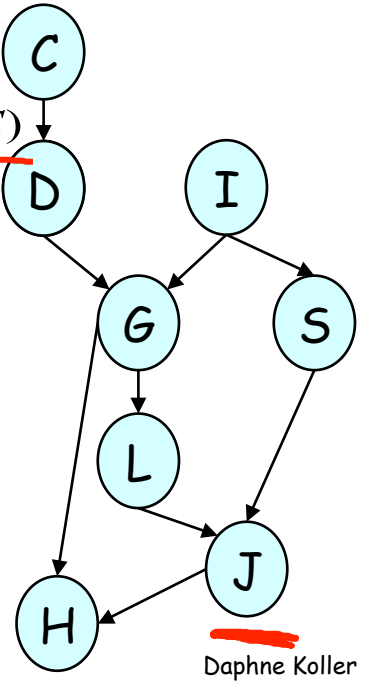
- Goal: P(J)
- Eliminate: C,D,I,H,G,S,L

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

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

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

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



Variable Elimination

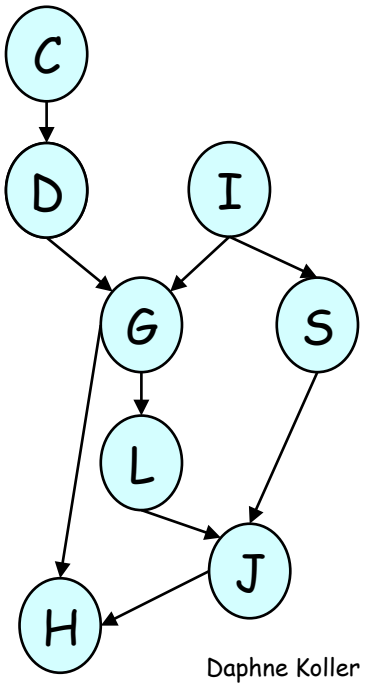
- Goal: P(J)
- Eliminate: D,I,H,G,S,L

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

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

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

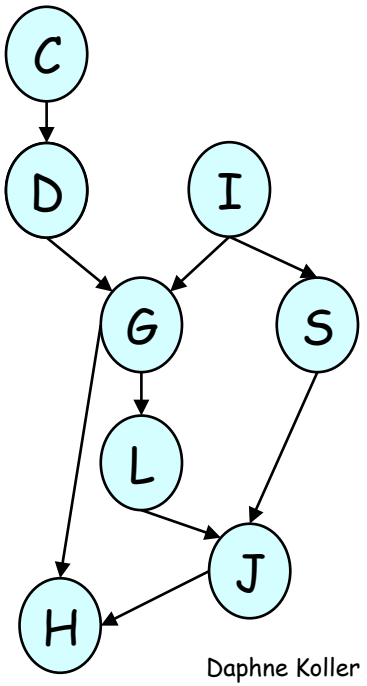
- Goal: P(J)
- Eliminate: I,H,G,S,L

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

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

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

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



Daphne Koller

Variable Elimination

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

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

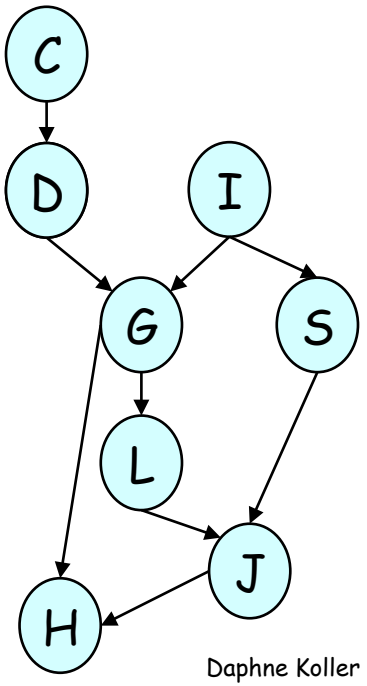
$$\sum_{L,S,G} \phi_J(J,L,S)\phi_L(L,G)\tau_3(S,G)\sum_H \phi_H(H,G,J)$$

$\sum_H P(H|G,J) = 1$

$\tau_4(G,J)$

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

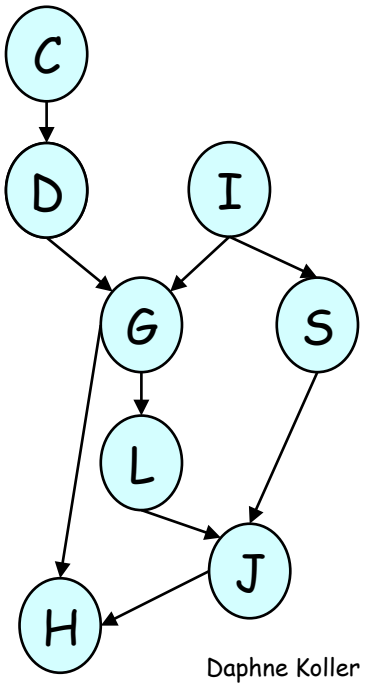
$$\sum_{L,S,G} \phi_J(J,L,S)\phi_L(L,G)\tau_3(S,G)\tau_4(G,J)$$

L, G, S, J

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

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

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

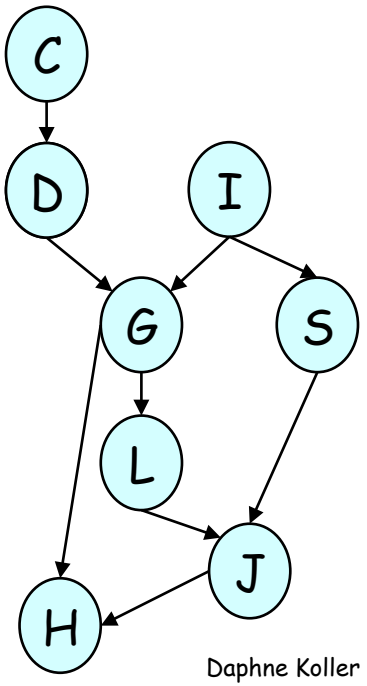


Daphne Koller

Variable Elimination

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

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



Daphne Koller

Variable Elimination with evidence

- Goal: P(J,I=i,H=h)
- Eliminate: C,D,G,S,L (no H, I)

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

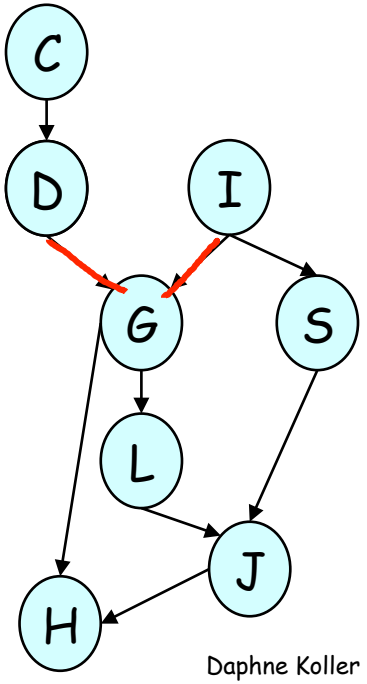
$\phi_G'(G,D)$: P(G|i,D) $\quad$ $\phi_I'()$: $\phi_I(i)$

elimination as before

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

renormalize

P(I=i,H=h) is the renormalizing constant



Variable Elimination in MNs

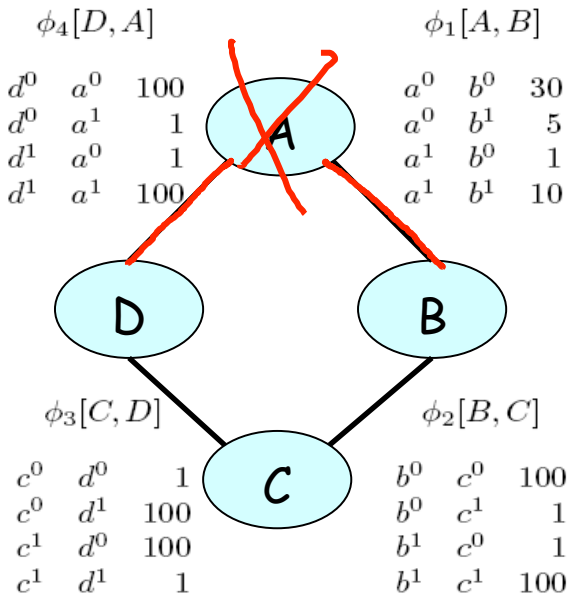
- Goal: P(D)
- Eliminate: A,B,C

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

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

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

At the end of elimination get $\tau_3(D) \propto P(D)$



Eliminate-Var Z from Φ

all factors that involve z

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

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

multiply them

$$\tau = \sum_{Z} \psi$$

sum out z

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

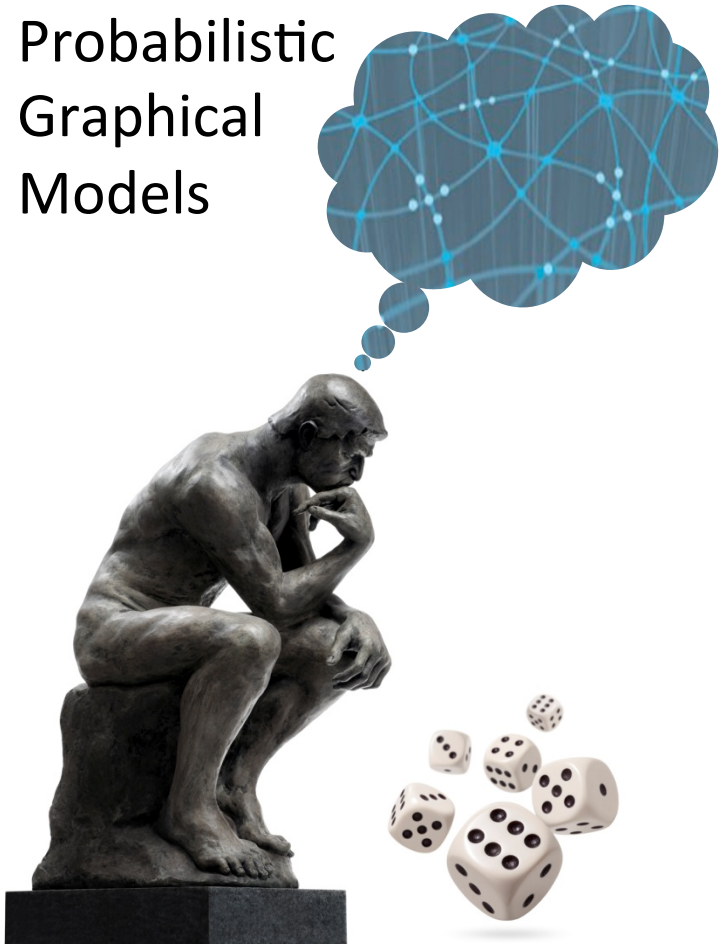
VE Algorithm Summary

- Reduce all factors by evidence
  - Get a set of factors Φ
- For each non-query variable Z
  - Eliminate-Var Z from Φ adds removes Φ' adds τ
- 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

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

factor product

$$\tau_k(\boldsymbol{X}_k - \{Z\}) = \sum_{Z} \psi_k(\boldsymbol{X}_k)$$

marginalization

Reminder: Factor Product

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

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

a^1	b^1	0.5
a^1	b^2	0.8
a^2	b^1	0.1
a^2	b^2	0
a^3	b^1	0.3
a^3	b^2	0.9

b^1	c^1	0.5
b^1	c^2	0.7
b^2	c^1	0.1
b^2	c^2	0.2

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

Cost: $(m_k-1)N_k$ multiplications

Reminder: Factor Marginalization

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

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

Cost: ~N_k additions

each number used exactly once 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 ≤ 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* ≤ m + n

Complexity of Variable Elimination

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

Complexity of Variable Elimination

- Total work is linear in N and m — exponential blowup
- $N_k = |Val(\mathbf{X}_k)| = O(d^{r_k})$ where — # variables in k^{th} factor
  - $d = \max(|Val(X_i)|)$ — d values in their scope
  - $r_k = |\mathbf{X}_k|$ = cardinality of the scope of the k^{th} factor

Complexity Example

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

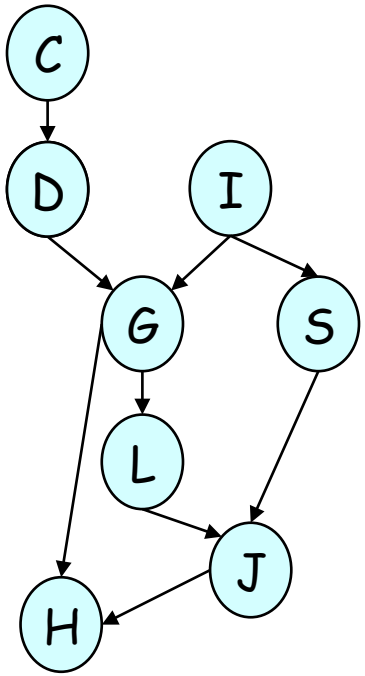
$$\tau_2(G,I) = \sum_D \phi_G(G,I,D)\tau_1(D)$$ 3

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

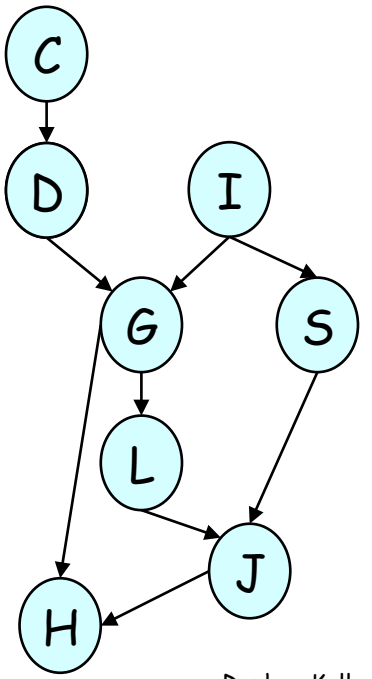


Complexity and Elimination Order

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

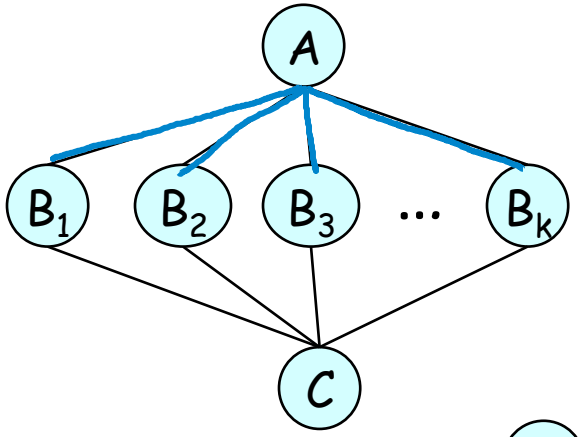
- Eliminate: G

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



Complexity and Elimination Order

Eliminate A first:

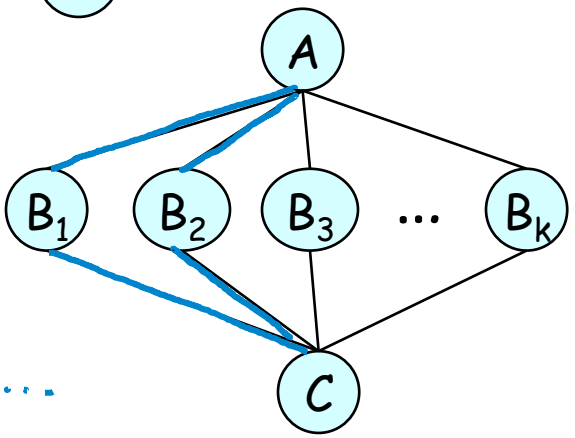


$\{A, B_1, \ldots, B_k\}$

size of factor is exp in k

$\prod_i \tau_i(A,C)$

Eliminate B_i's first:



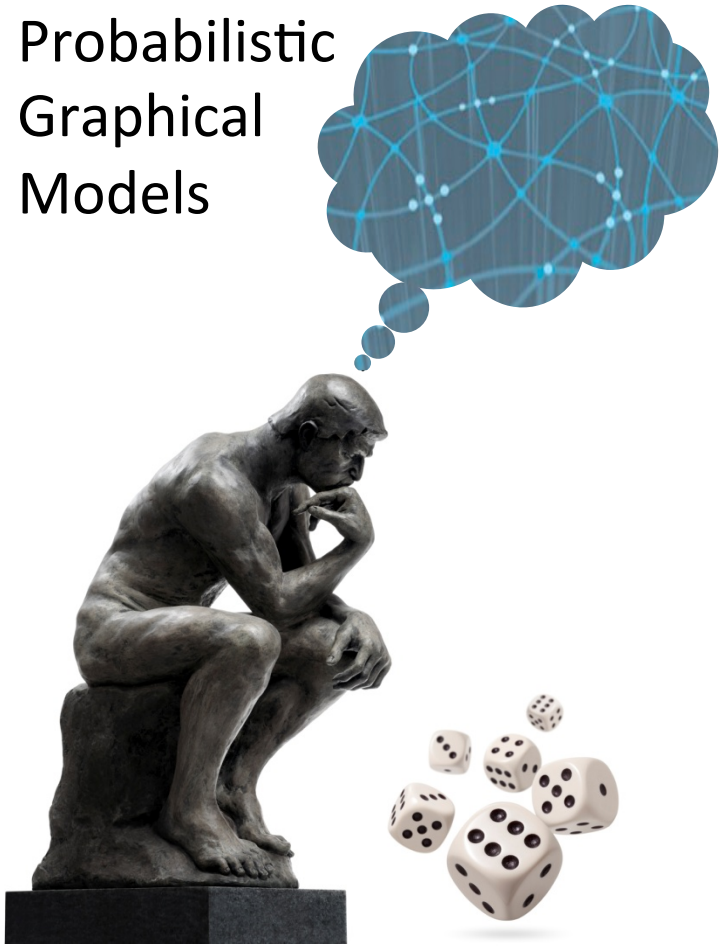
$\underbrace{\phi_1(A,B_1) \cdot \phi_2(C,B_2)}_{\text{scope } A,B_1,C} \Rightarrow \tau_1(A,C)$

$\tau_2(A,C) \ldots$

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

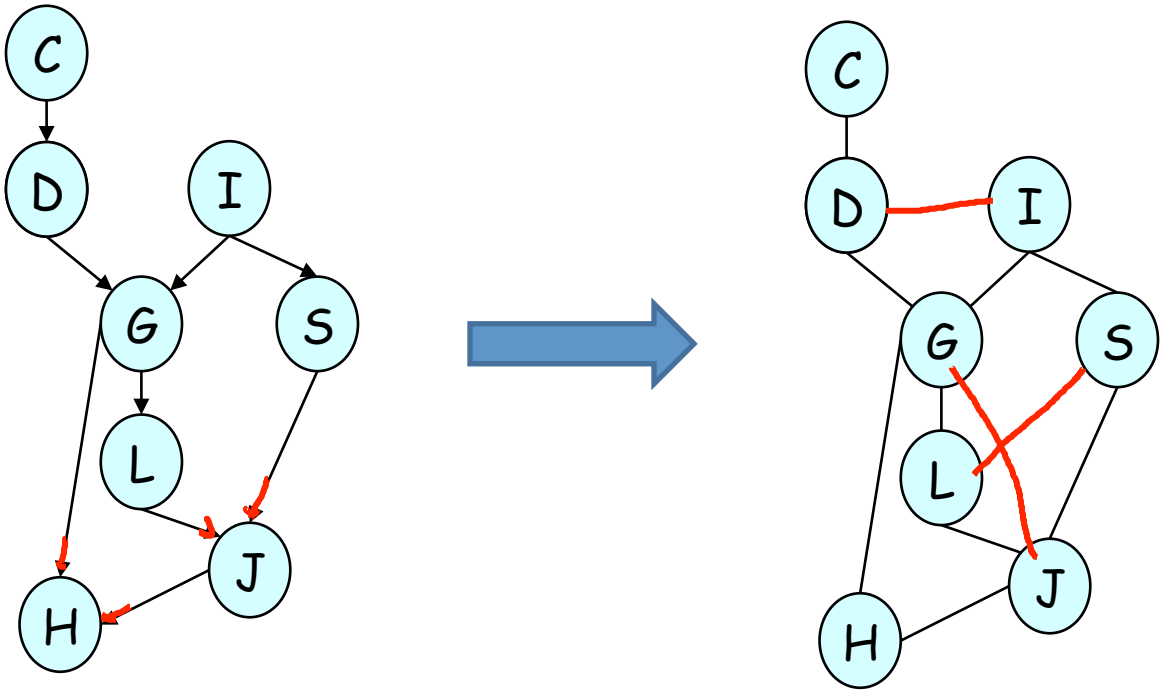
Variable Elimination

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

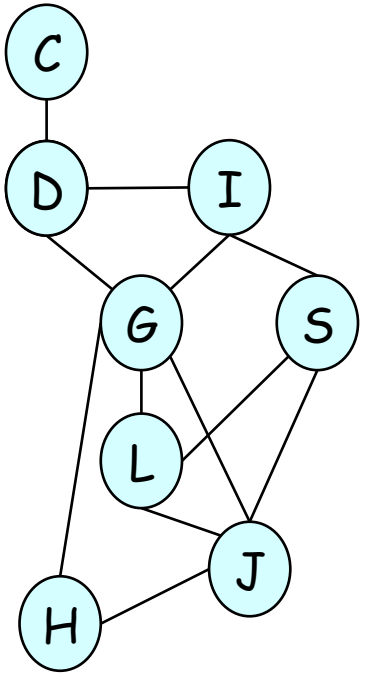


Elimination as Graph Operation

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

- Eliminate: C

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



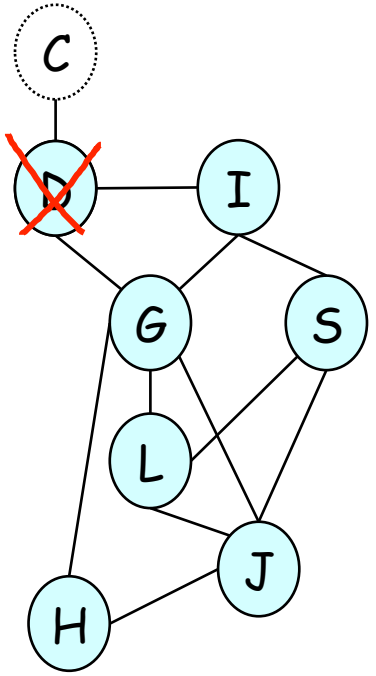
Induced Markov network for the current set of factors

Elimination as Graph Operation

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

- 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

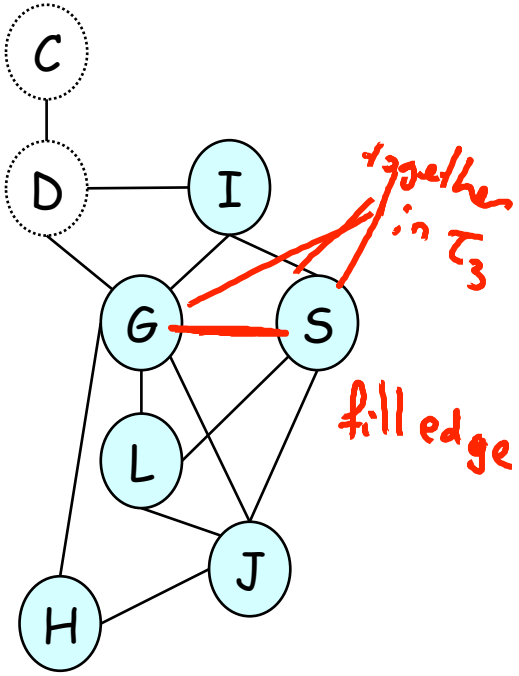
Elimination as Graph Operation

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

- 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



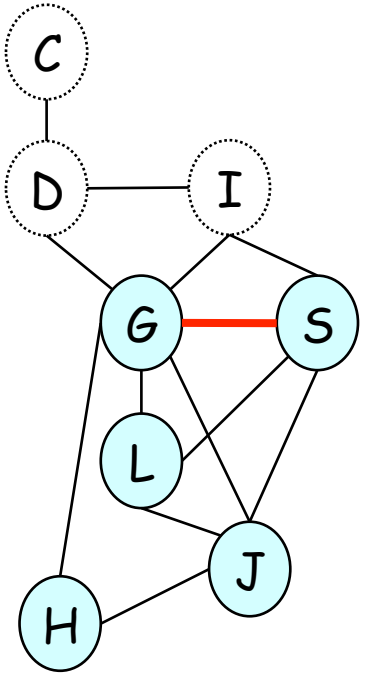
Induced Markov network for the current set of factors

Elimination as Graph Operation

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

- Eliminate: H

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

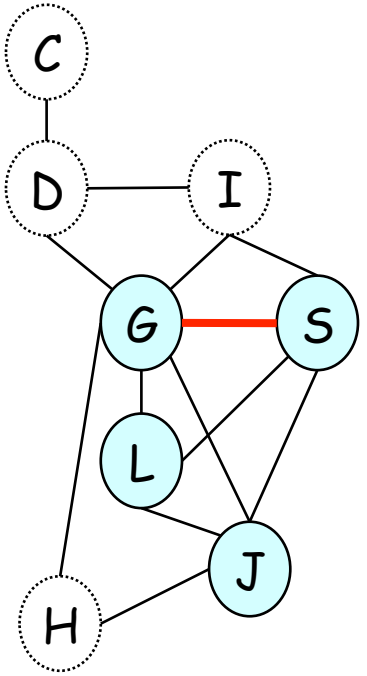


Induced Markov network for the current set of factors

Elimination as Graph Operation

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

- Eliminate: *G*



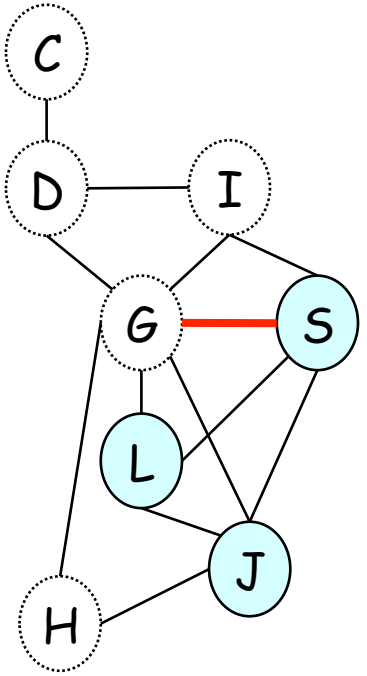
$$\tau_5(L,J) = \sum_G \phi_L(L,G)\tau_3(S,G)\tau_4(G,J)$$

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

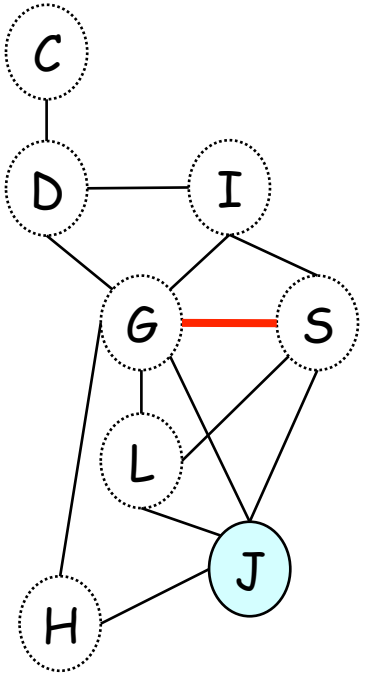


Induced Markov network for the current set of factors

Elimination as Graph Operation

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

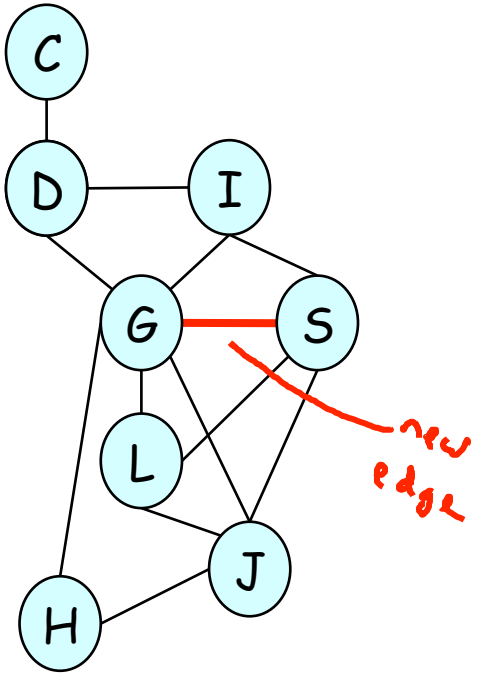
- Eliminate: L,S



Induced Markov network for the current set of factors

Induced Graph

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



Cliques in the Induced Graph

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

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

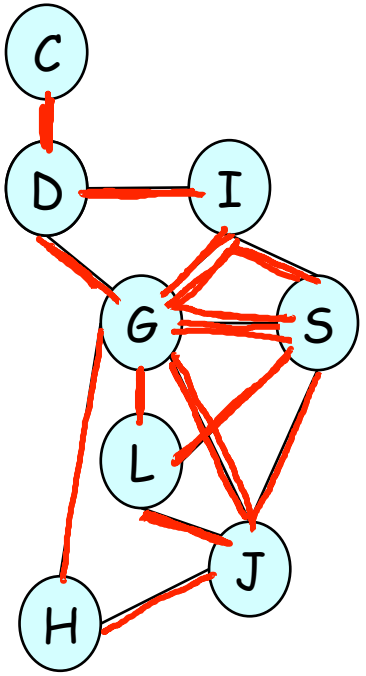
$$\tau_2(G,I) = \sum_D \phi_G(G,I,D)\tau_1(D)$$

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

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

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

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



Cliques in the Induced Graph

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

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

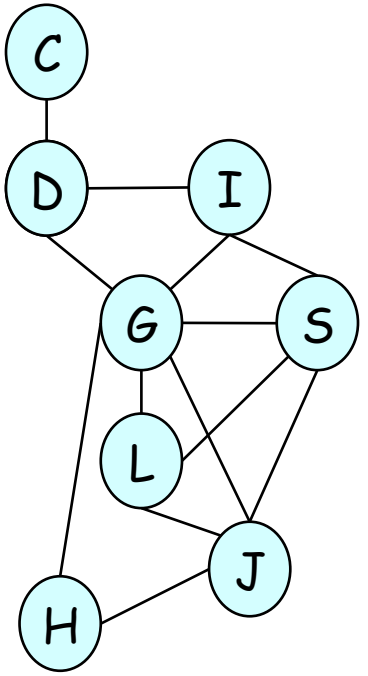
$$\tau_2(G,I) = \sum_D \phi_G(G,I,D)\tau_1(D)$$

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

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

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

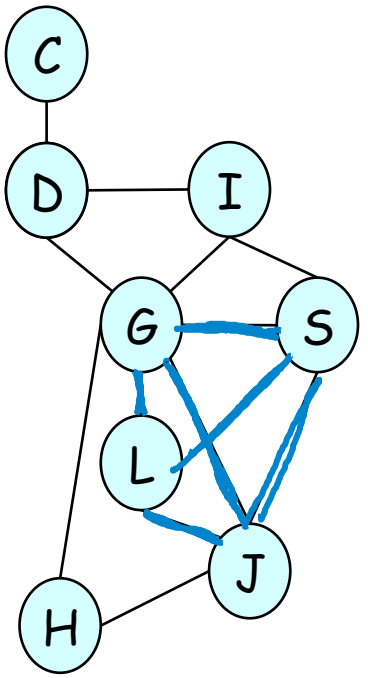
$$\tau_6 = \sum_{L,S} \phi_J(J,L,S)\tau_5(L,J)$$



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 neighbors are added to it
⇒ when eliminated it already had all the clique members as neighbors
⇒ participated in factor with all these other variables
⇒ when multiplied together, we have a factor over all of them



Induced Width

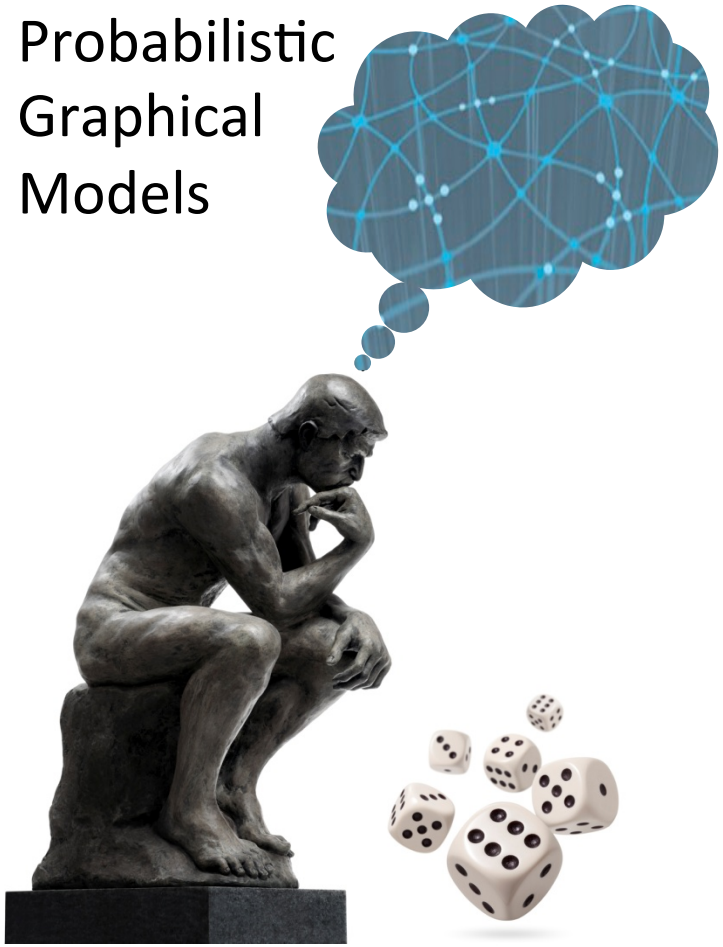
- The width of an induced graph is the number of nodes in the largest clique in the graph minus 1
- Minimal induced width of a graph K is $\min_{\alpha}(\text{width}(I_{K,\alpha}))$
- Provides a lower bound on best performance of VE to a model factorizing over K

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 ≤ K is NP-complete

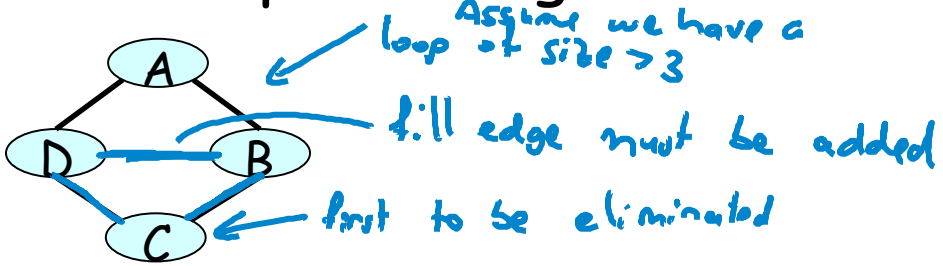
- Note: This NP-hardness result is distinct from the NP-hardness result of inference
  - Even given the optimal ordering, inference may still be exponential

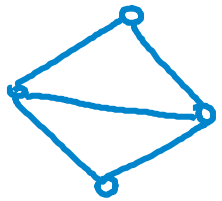
Finding Elimination Orderings

- Greedy search using heuristic cost function
  - At each point, eliminate node with smallest cost
- Possible cost functions:
  - min-neighbors: # neighbors in current graph (smallest factor)
  - 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)

Finding Elimination Orderings

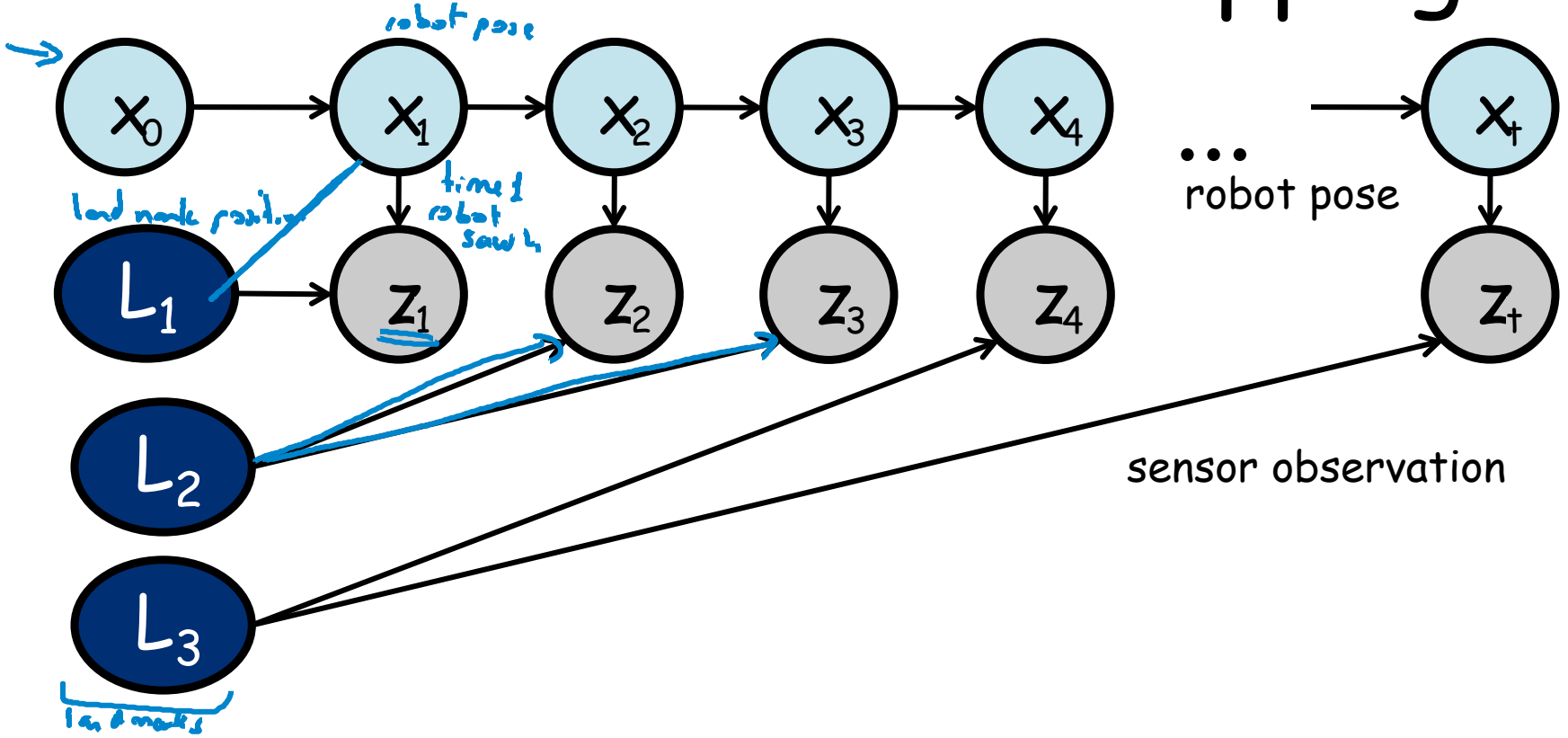
- Theorem: The induced graph is triangulated
  - No loops of length > 3 without a "bridge"





- Can find elimination ordering by finding a low-width triangulation of original graph H_Φ

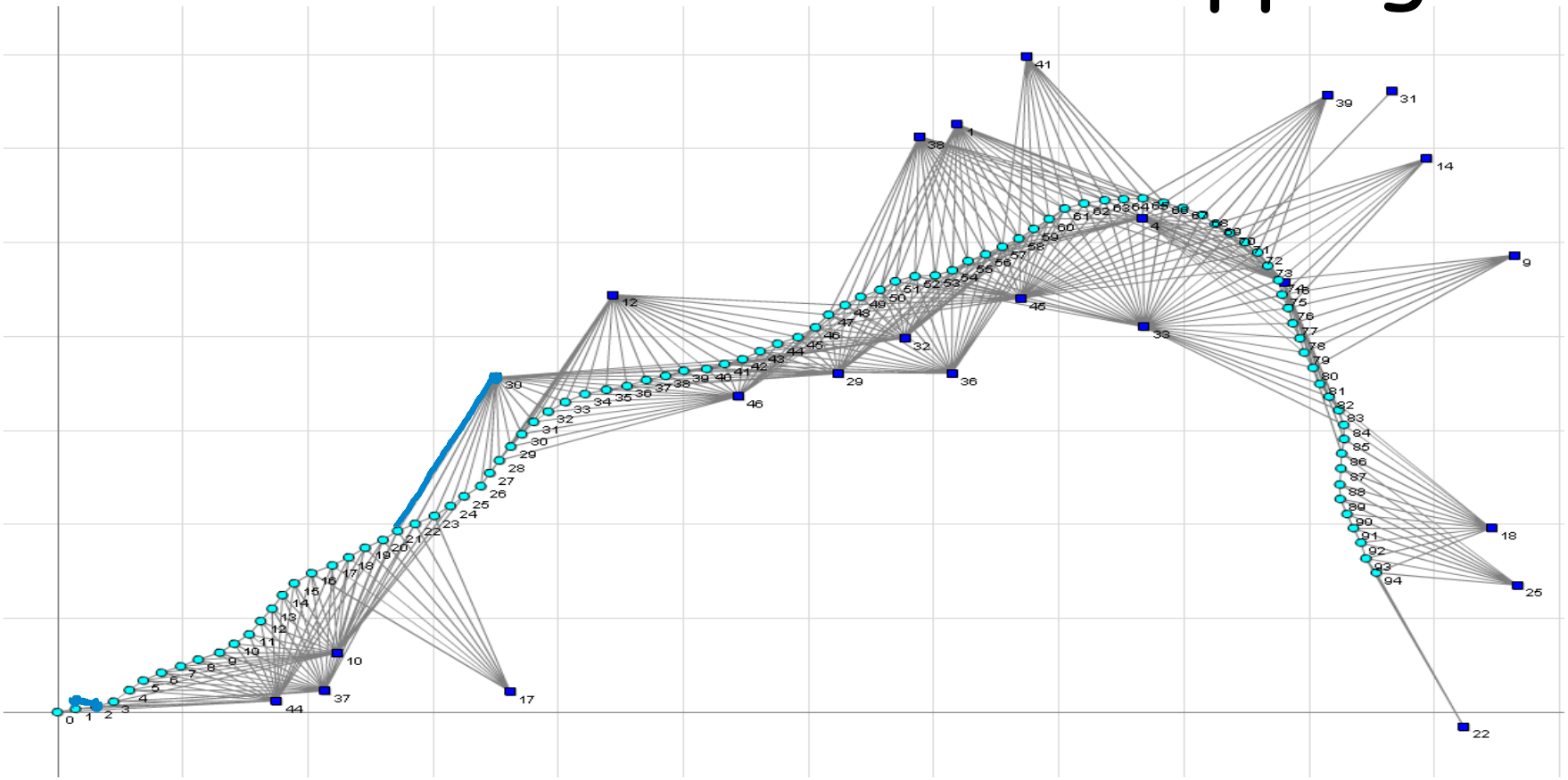
Robot Localization & Mapping



Daphne Koller

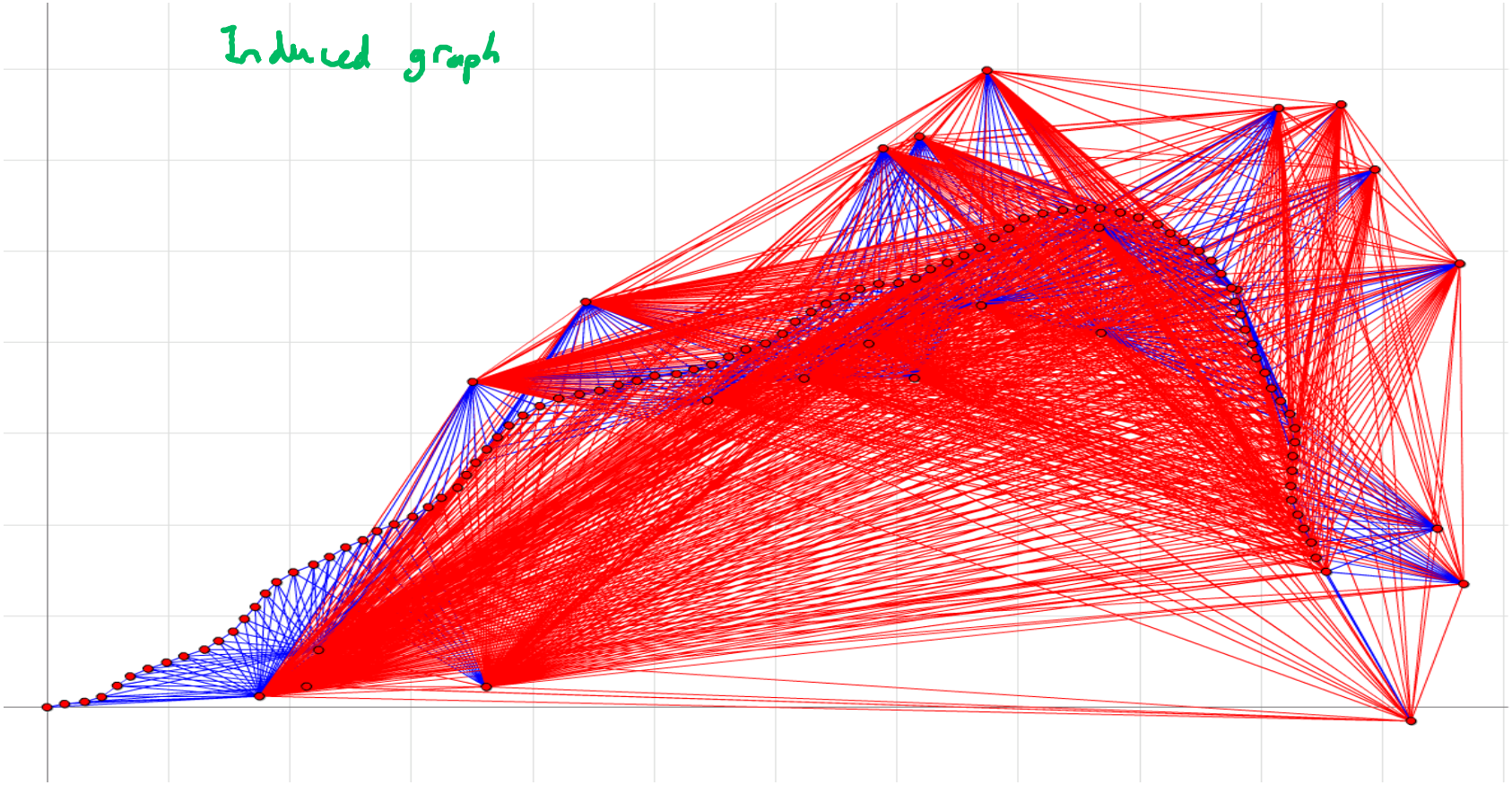
Square Root SAM, F. Dellaert and M. Kaess, IJRR, 2006

Robot Localization & Mapping



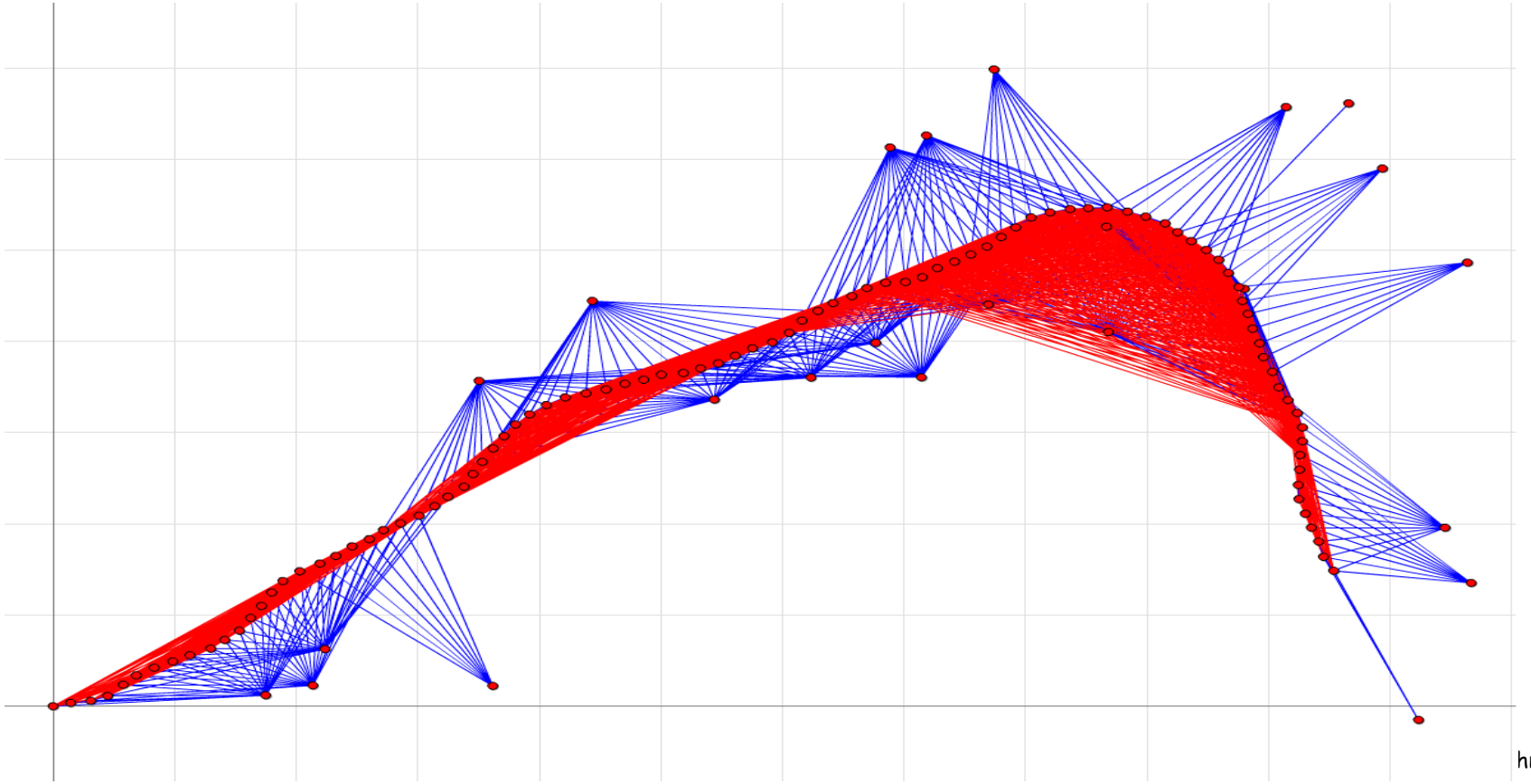
Square Root SAM, F. Dellaert and M. Kaess, IJRR, 2006

Eliminate Poses then Landmarks



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

Eliminate Landmarks then Poses



hne Koller

Summary

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