Universität Stuttgart
IPVS – Institute for Parallel and Distributed Systems

Analytic Computing

Advanced Topics in Machine Learning
6 Inference

Prof. Dr. Steffen Staab
Dr. Rafika Boutalbi
Zihao Wang
https://www.ipvs.uni-stuttgart.de/departments/ac/

Learning Objectives

Inference Methods:

- Conditional Probability Queries
  - sum product
- MAP Queries
- Variable Elimination
- Complexity Considerations
- Belief Propagation
  - Cluster Graphs
  - Cluster Graph Beliefs
  - Calibration

Disclaimer

Figures and examples not marked otherwise are taken from the book by Koller & Friedman

1 Conditional-Probability Queries

Conditional Probability Queries

- Evidence: $\boldsymbol{E} = \boldsymbol{e}$
- Query: a subset of random variables $\boldsymbol{Y}$
- Task: compute $P(\boldsymbol{Y} \mid \boldsymbol{E} = \boldsymbol{e}) = \dfrac{P(\boldsymbol{Y}, \boldsymbol{e})}{P(\boldsymbol{e})}$

- Typical applications
  - medical diagnoses
  - diagnosis for car maintenance
  - pedigree (blood type) analysis

Tractability

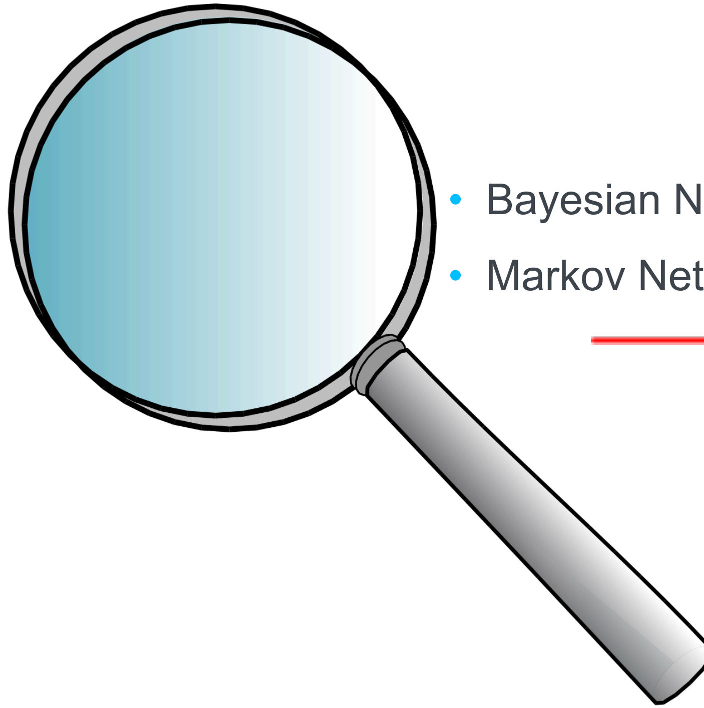
Bad News

- The following are all NP-hard:
- Given a PGM $\boldsymbol{P}_{\Phi}$, a variable X and a value x from X, compute:
  - $\boldsymbol{P}_{\Phi}(X = x)$
  - $\boldsymbol{P}_{\Phi}(X = x) > 0$
  - Find some p such that $\left| \boldsymbol{P}_{\Phi}\left(X = x \mid \boldsymbol{E} = \boldsymbol{e}\right) - p \right| < 0.5,$ for some given observation $\boldsymbol{E} = \boldsymbol{e}$

Good News

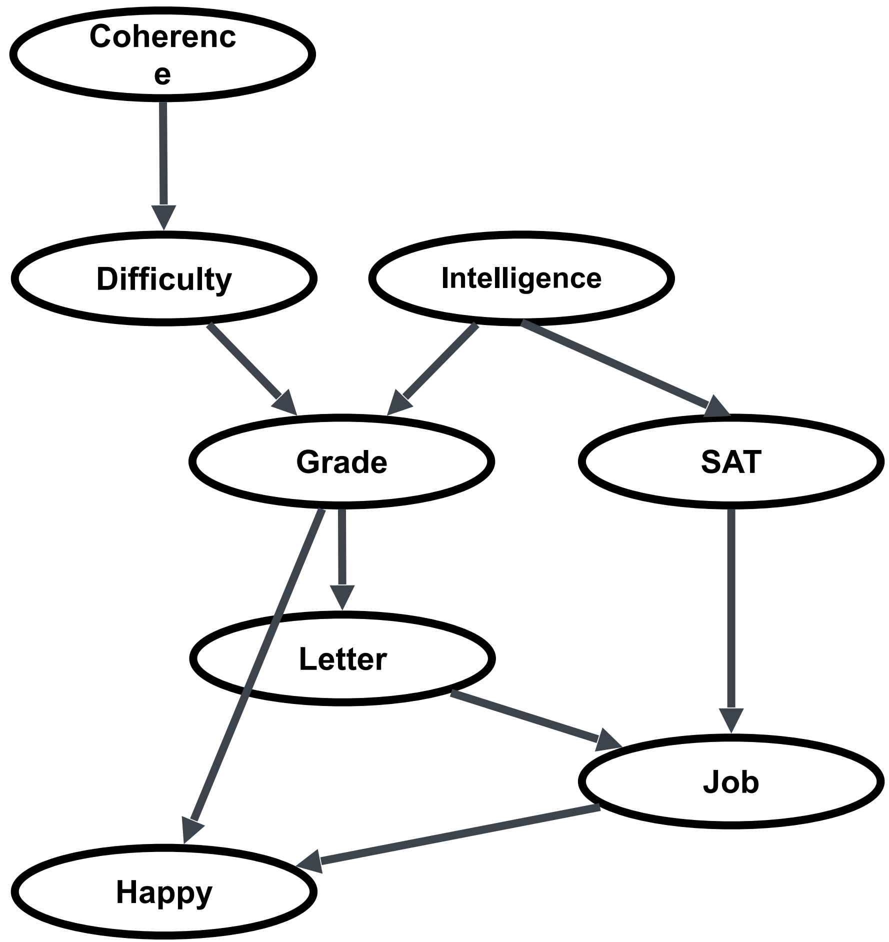
- Practical problems often do not exhibit this worst case difficulty

Inference using Graphs and Factors



- Bayesian Networks
- Markov Networks

By AbiClipart - http://openclipart.org/clipart/people/magnifying_glass_01.svg, CC0, https://commons.wikimedia.org/w/index.php?curid=1651079

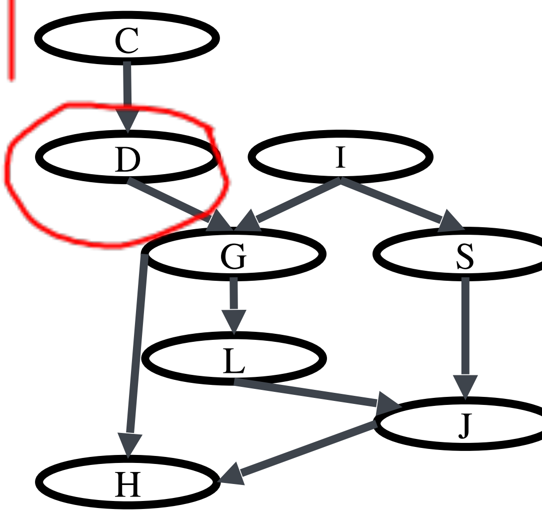

Coherenc
e
Difficulty
Intelligence
Grade
SAT
Letter
Job
Happy

Joint Distribution Chain Rule

$$\phi_C(C)\phi_D(C,D)\phi_I(\overset{2}{I})\phi_G(G,I,D) \bullet$$

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

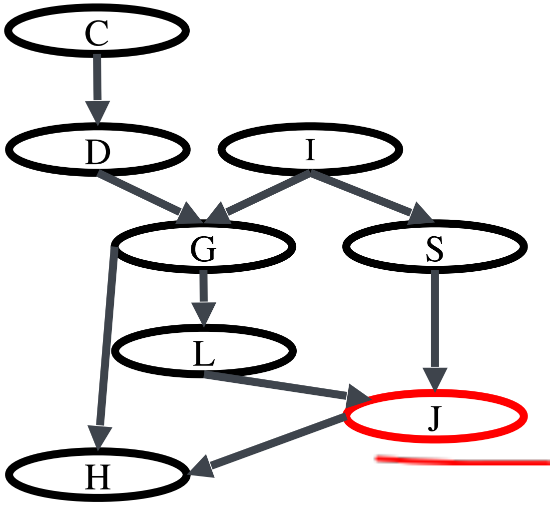
2^8



Note that in Bayesian Networks we have exactly one factor (representing the conditional probability table) per node X: ϕ_X

Sum Product: Bayes Network Example

$$P(J) = \tilde{P}(J) = = \sum_{C,D,I,G,S,L,H} \begin{pmatrix} \phi_C(C)\phi_D(C,D)\phi_I(I)\phi_G(G,I,D) \bullet \\ \phi_S(S,I)\phi_L(L,G)\phi_J(J,L,S)\phi_H(H,G,J) \end{pmatrix}$$



Note that **in Bayesian Networks** we have exactly one factor (representing the conditional probability table) per node X: ϕ_X

Sum Product: Markov Network Example

$$\tilde{P}(D) == \sum_{A,B,C} \begin{pmatrix} \phi_1(A,B)\phi_2(B,C) \bullet \\ \phi_3(C,D)\phi_4(D,A) \end{pmatrix}$$

$$P(D) = \frac{1}{Z}\tilde{P}(D)$$

renormalization

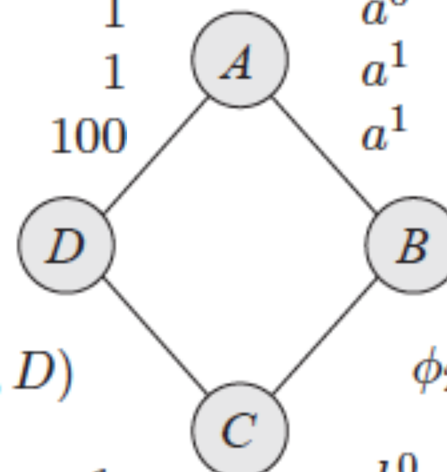
d^0	4
d^1	8

$\phi_4(D,A)$

d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100

$\phi_1(A,B)$

a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10



$\phi_3(C,D)$

c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

$\phi_2(B,C)$

b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

Given Evidence

$W = \{X_1, \ldots, X_n\} - Y - E$

Query for Y given observation $E = e$:

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

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

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

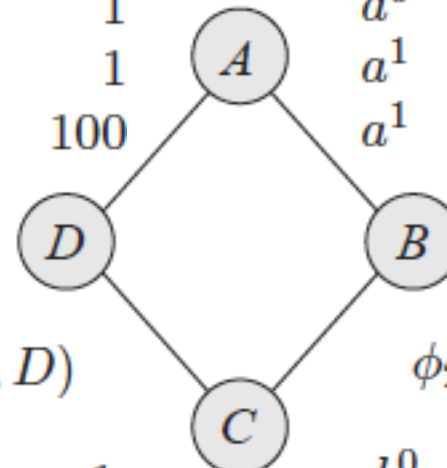
with $D_k = \text{scope}(\phi_k)$

$\phi_4(D, A)$

d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100

$\phi_1(A, B)$

a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10



$\phi_3(C, D)$

c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

$\phi_2(B, C)$

b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

Evidence: Reduced Factors

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

Query for Y given observation $A = a^0$:

$$P(Y \mid A = a^0) = \frac{P(Y, A = a^0)}{P(A = a^0)}$$

$$P(Y, a^0) = \sum_{W} P(Y, W, A = a^0) =$$

$$= \sum_{W} \frac{1}{Z} \prod_{k} \phi_k(D_k, A = a^0) =$$

$$\propto \sum_{W} \prod_{k} \phi'_k(D'_k)$$

with reduced factors ϕ'_k and their scopes D'_k

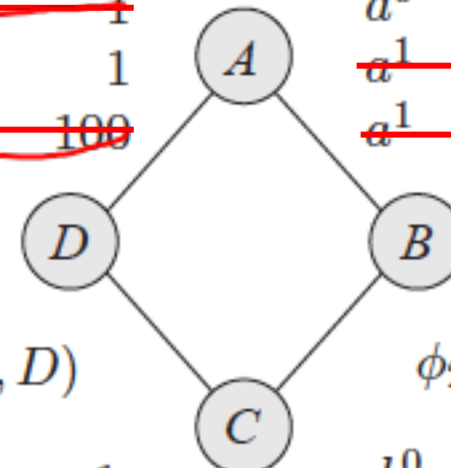
renormalize

$\phi_4(D, A)$

d^0	a^0	100
~~d^0~~	~~a^1~~	~~1~~
d^1	a^0	1
~~d^1~~	~~a^1~~	~~100~~

$\phi_1(A, B)$

a^0	b^0	30
a^0	b^1	5
~~a^1~~	~~b^0~~	~~1~~
~~a^1~~	~~b^1~~	~~10~~



$\phi_3(C, D)$

c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

$\phi_2(B, C)$

b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

05.12.21 13

Sum Product

$W = \{X_1, \ldots, X_n\} - Y - E$

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

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

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

Easier: only compute $\sum_W \frac{1}{Z} \prod_k \phi'_k(D'_k)$ and renorm

In general:
Computing a partition function is hard!
Must sum over all possible assignments!

Renormalization

Compute $\dfrac{P(Y,e)}{\cancel{P(e)}} = \dfrac{\phi^*}{\alpha}$

Algorithm 9.2 Using Sum-Product-VE **for computing conditional probabilities**

```
Procedure Cond-Prob-VE (
    𝒦,        // A network over 𝒳
    Y ,       // Set of query variables
    E = e     // Evidence
)
    Φ ← Factors parameterizing 𝒦
    Replace each φ ∈ Φ by φ[E = e]
    Select an elimination ordering ≺
    Z ← = 𝒳 − Y − E
    φ* ← Sum-Product-VE(Φ, ≺, Z)
    α ← Σ_{y∈Val(Y)} φ*(y)
    return α, φ*
```

Overview: Algorithms to Compute Conditional Probability

- Push summations into factor product
  - dynamic programming: Variable elimination — *exact*
- Message passing over a graph
  - belief propagation — *exact*
  - variational approximations — *approximation*
- Random sampling instantiations — *approximations*
  - Markov Chain Monte Carlo (MCMC)
  - Importance sampling

2 Maximum A-Posteriori (MAP) Queries

Maximum a Posteriori (MAP)

- Evidence: $E = e$
- Query: **<u>all</u>** other variables Y, $Y = \{X_1, \ldots, X_n\} - E$
- Task: compute

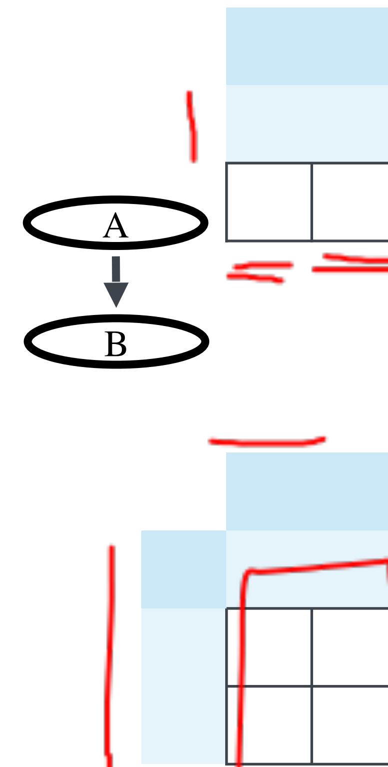
$$MAP(Y \mid E = e) = \text{argmax}_y P(Y = y \mid E = e)$$

  - Note: There may be more than one possible solution

- Typical Applications
  - Message decoding: most likely transmitted message
  - Image segmentation: most likely segmentation

MAP≠Max over Marginals

- Joint distribution
  - $P(a^0, b^0) = 0.04$
  - $P(a^0, b^1) = 0.36$
  - $P(a^1, b^0) = 0.3$
  - $P(a^1, b^1) = 0.3$
- $MAP(A) = a^1,\ \mathrm{P}(\mathrm{a}^1) = 0.6$
- $MAP(B) = b^1,\ \mathrm{P}(b^1) = 0.66$
- $MAP(A, B) = (a^0, b^1),\ P(a^0, b^1) = 0.36$
- $MAP(A \mid B = b^1) = a^0,\ P(A \mid B = b^1) = \frac{0.36}{0.36 + 0.3} = \frac{6}{11}$



Tractability

Bad News

- The following are all NP-hard:
- Given a PGM $\boldsymbol{P}_\Phi$,
  - find a joint assignment $\boldsymbol{x}$ with highest probability $P_\Phi(\boldsymbol{x})$
    - i.e. $E = \{\}$
  - decide if there is an assignment $\boldsymbol{x}$ such that $P_\Phi(\boldsymbol{x}) > \boldsymbol{p}$

Good News

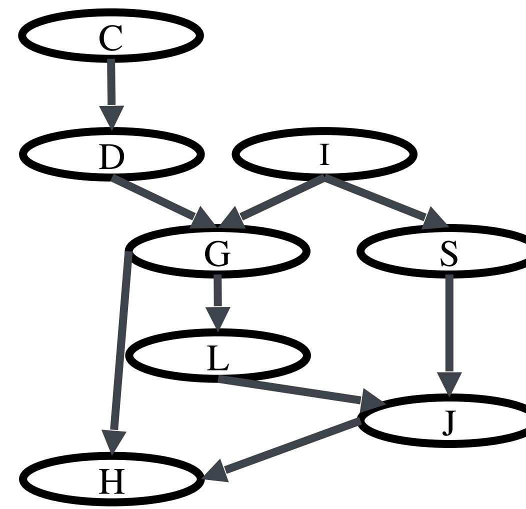
- Practical problems often do not exhibit this worst case difficulty

Max Product

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

$$\text{argmax}_{C,D,I,G,S,G,L,J,H}\ \phi_C(C)\phi_D(C,D)\phi_I(I)\phi_G(G,I,D) \bullet$$

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



Max Product

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

$$P(\boldsymbol{Y},\boldsymbol{E}=\boldsymbol{e}) = \frac{1}{Z}\prod_k \phi_k'(D_k')$$

$$\propto \prod_k \phi_k'(D_k')$$

$$\operatorname{argmax}_{\boldsymbol{y}} P\left(\boldsymbol{Y}=\boldsymbol{y} \,\middle|\, \boldsymbol{E}=\boldsymbol{e}\right) = \operatorname{argmax}_{\boldsymbol{y}\in\boldsymbol{Y}} \prod_k \phi_k'(D_k')$$

$P(\boldsymbol{E}=\boldsymbol{e})$ is constant in this task

$\phi_k'(D_k')$ reduced relative to $\boldsymbol{E}=\boldsymbol{e}$

Overview: Algorithms to Compute MAP

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

MAP is a discrete optimization task!

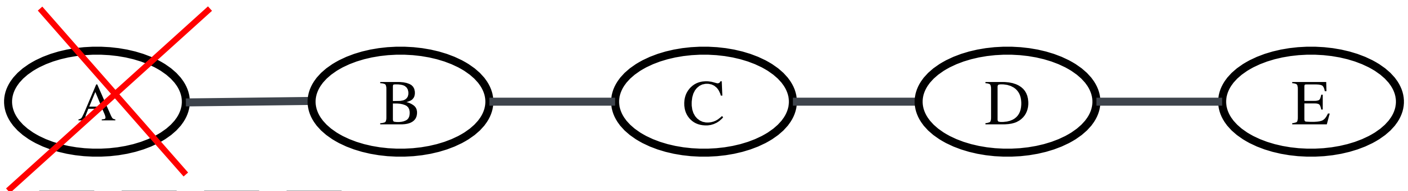
Summary of MAP

- MAP: single coherent assignment of highest probability
- Maximizing over factor product
- Combinatorial optimization problem
- Many exact and approximate algorithms

Question 2:
MAP

3 Variable Elimination

Variable elimination in chains



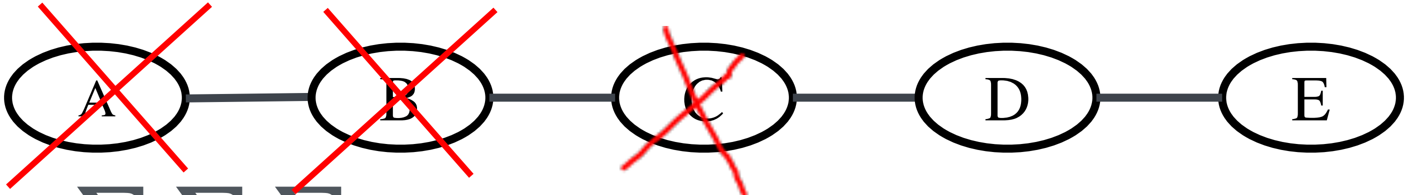
$$P(E) \propto \sum_D \sum_C \sum_B \sum_A \tilde{P}(A, B, C, D, E) =$$

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

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

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

Variable elimination in chains



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

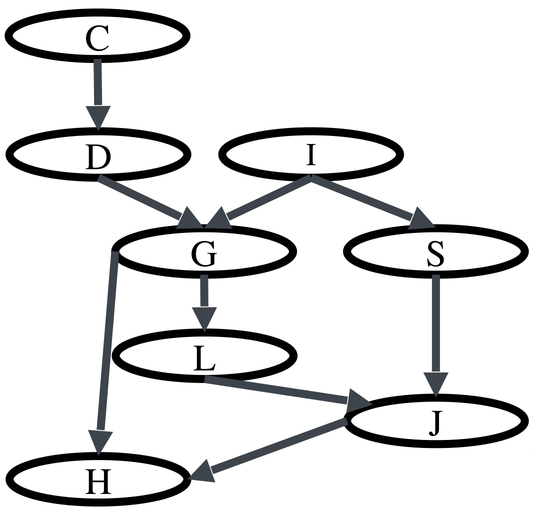
$$= \sum_D \sum_C \phi_3(C,D)\phi_4(D,E) \sum_B \phi_2(B,C)\tau_1(B) =$$

$\tau_2(C)$

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

Variable elimination in graphs

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



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

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

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

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

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

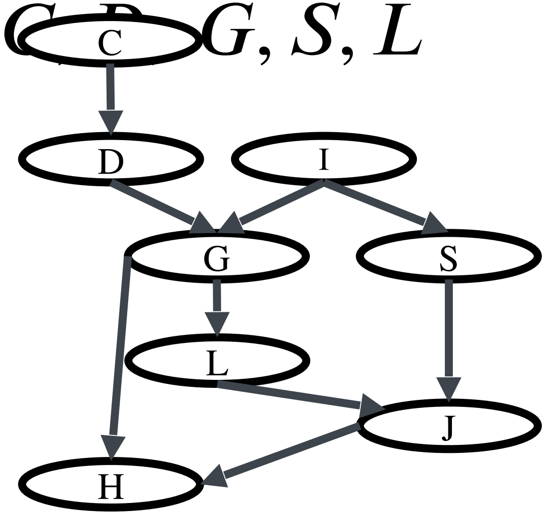
Variable elimination with evidence

- Goal:

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

- Eliminate:

$$C, D, G, S, L$$



$$P(J \mid I = i, H = h) \propto \propto \sum_{L,S,G,D,C} \left(\begin{array}{l} \phi_C(C)\phi_D(C,D)\phi_I(I)\phi_G(G,I,D)\phi_S(S,I) \bullet \\ \bullet\, \phi_L(L,G)\phi_J(J,L,S)\phi_H(H,G,J) \end{array} \right)$$

Reduce factors

$$= \sum_{L,S,G,D,C} \left(\begin{array}{l} \phi_C(C)\phi_D(C,D)\phi'_I\,\phi'_G(G,D)\phi'_S(S) \bullet \\ \bullet\, \phi_L(L,G)\phi_J(J,L,S)\phi'_H(G,J) \end{array} \right) = \ldots$$

Eventually: renormalize
using $P(I = i, H = h)$ in the denominator

General procedure for eliminating one variable in PGM

- Objective: eliminate random variable Z from factors Φ
- Do:
  - $\Phi' = \{\phi_i \in \Phi : Z \in \text{scope}(\phi_i)\}$
  - $\psi = \prod_{\phi_i \in \Phi'} \phi_i$
  - $\tau = \sum_{Z} \psi$
  - $\Phi = \Phi - \Phi' \cup \{\tau\}$

General procedure for variable elimination (summary)

1. Reduce all factors by evidence
   - get a set of factors Φ
2. For each non-query variable Z
   - Eliminate random variable Z from factors Φ
3. Multiply all remaining factors
4. Renormalize to get distribution

So far: Naive algorithm
Tricks must be applied!

Question 3: Variable elemination

4 Computational Complexity of Variable Elimination

Sources of computational complexity

- Z to be eliminated
- Multiplications: factor product

$$\psi_k(X_k) = \prod_{i=1}^{m_k} \phi_i$$

- Additions: marginalization

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

- Z in scope of m_k factors in round k
- scope of ψ_k

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

Each new row comes from $m_k - 1$ multiplications

Scope:

A, B

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

Scope:

B, C

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

Scope:

A, B, C

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

$N_k(m_k - 1)$ multiplications to create $\psi_k(\boldsymbol{X}_k)$

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

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

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

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

Each of the N_k rows is added once

Complexity of variable elimination

1. Start with m factors, n random variables
   - $m \leq n$ for Bayesian networks
   - $1 \leq m \leq 2^n - 2$ for Markov networks
2. At each elimination step generate 1 new factor τ_k
   - At most n elimination steps
   - Total number of factors: $m^* \leq m + n$
3. **Number of Operations**
   - Let $N = \max N_k$ be the size of the largest factor
   - Multiplications: $\sum_k N_k(m_k - 1) \leq N \sum_k (m_k - 1) \leq Nm^*$
   - Additions: $\sum_k N_k \leq Nn$

But: size of N ?

$\mathcal{O}(Nm^*)$

Size of Largest Factor

$$N_k = \left| \mathrm{Val}(X_k) \right| = \mathcal{O}(d^{r_k})$$

- $d = \max_{\mathrm{k}} |\mathrm{Val}(X_k)|$
- $r_k = |\boldsymbol{X}_{\mathrm{k}}|$

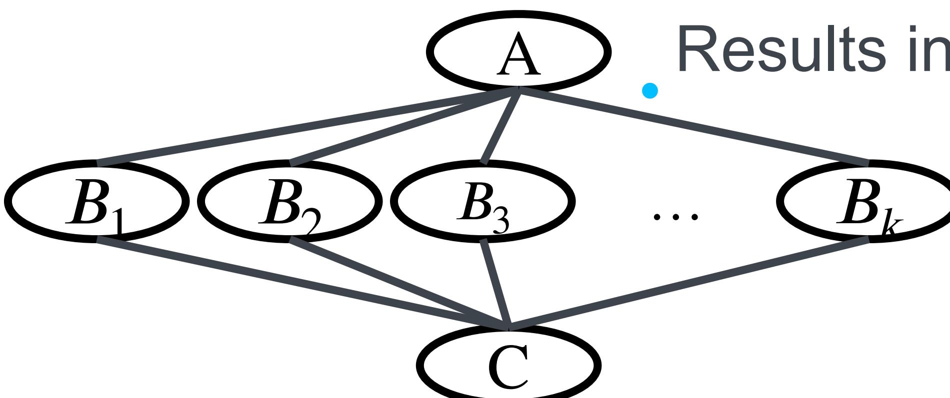
Exponential in the size of largest factor!

Complexity and Elimination Order

- Eliminate A first
  - $\psi_k(\{A, B_1, \ldots B_k\}) = \prod_{i=1}^{k} \phi_{\{A,B_i\}}$
  - size of first new factor is exponential in k
- Eliminate the B_i first
  - $\psi_i\Big(\{A, B_i, C\}\Big) = \phi_{\{A,B_i\}}\phi_{\{C,B_i\}}$
  - $\tau_i\big(\{A, C\}\big) = \sum_{B_i} \psi_i(\{A, B_i, C\})$

k

- Results in:



Effort may depend a lot on ordering of elimination!

Finding Elimination Orderings

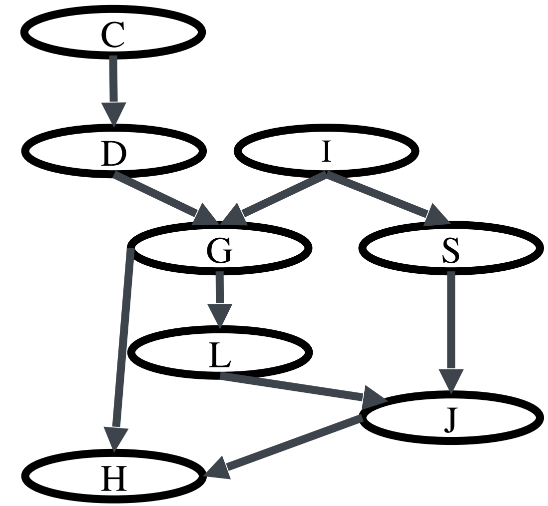
- Finding the best elimination ordering is NP hard
- However: greedy search using heuristic cost function
  - at each point, eliminate node with smallest cost
  - for this purpose:
    analyse (dynamic) graph structure
  - Possible cost functions:
    - min-neighbors: #neighbors in current graph (smallest factor)
    - min-weight: weight (#values) of factor formed
    - min-fill: minimize number of newly created dependencies (creating τs)
    - aggregations of these

Analysing the graph structure is key to understanding the costs of variable elimination. Many more details are in the book about this.

Question 4: Complexity of Variable Elimination

5 Belief Propagation in Trees

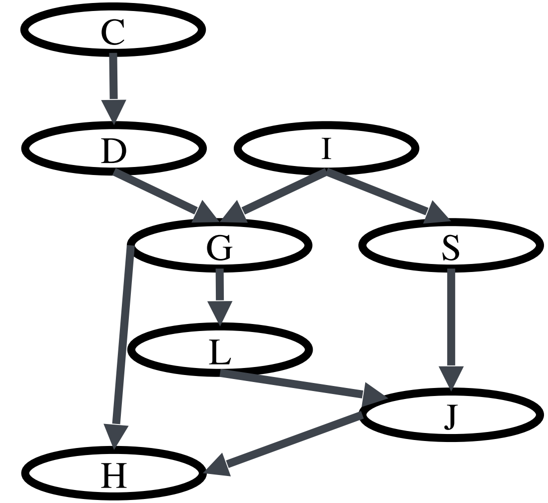
Message Passing

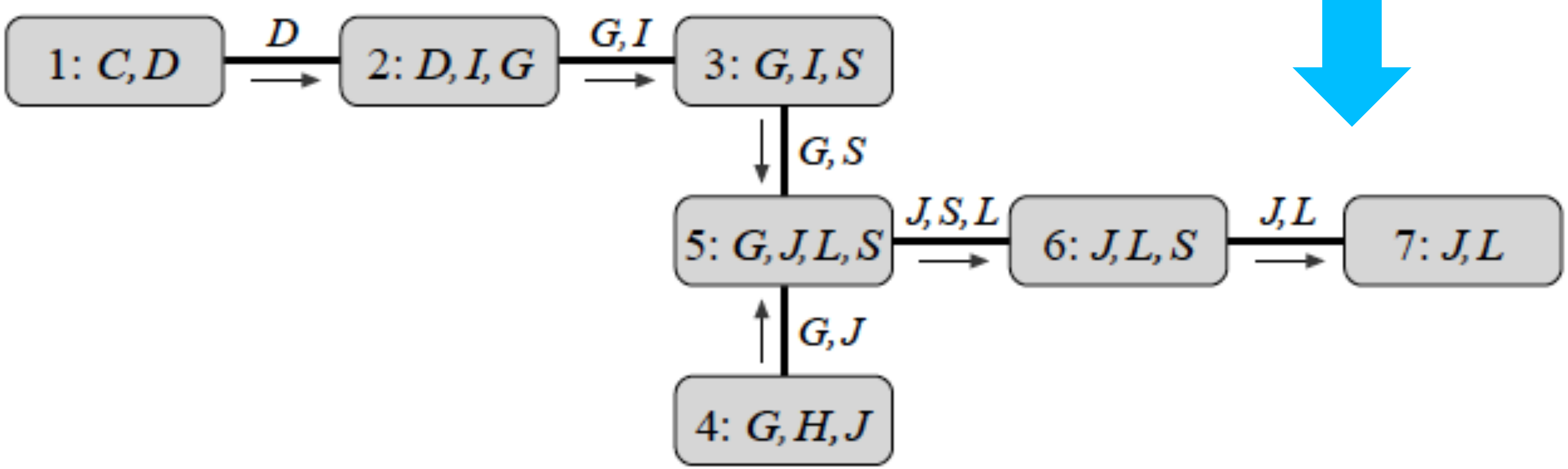


- A local view of sum-product computations
- Messages inform about local computations

Cluster Graphs

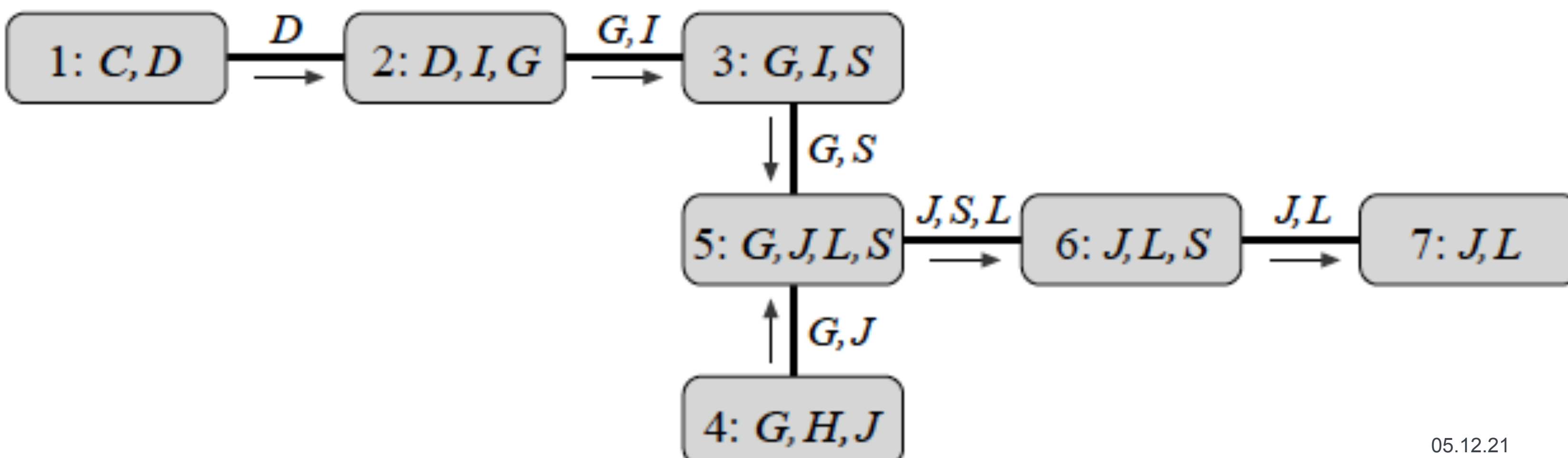
- A graphical view of factor-manipulations
- Result:
  - exact for tree-shaped cluster graphs
  - approximation for arbitrary cluster graphs



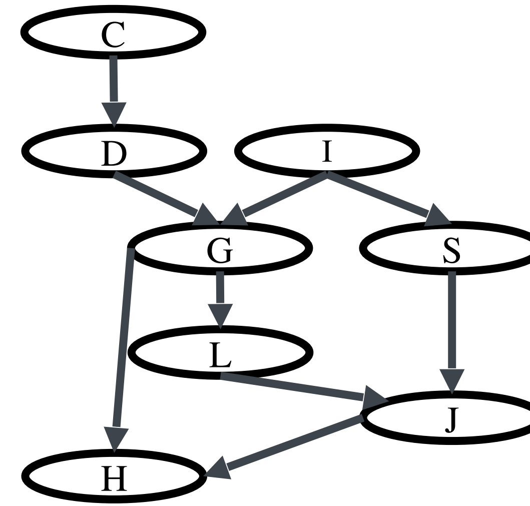


What is a cluster graph?

- Undirected graph such that:
  - nodes are clusters $C_i \subseteq \{X_1 \ldots X_n\}$
  - edge between C_i and C_j implies that **sepset** $S_{i,j} \subseteq C_i \cap C_j$, $S_{i,j} \neq \emptyset$
- Given set of factors Φ, we assign each ϕ_k to exactly one cluster $C_{\alpha(k)}$ such that $\mathrm{scope}(\phi_k) \subseteq C_{\alpha(k)}$
- Define $\psi_i(C_i) = \prod_{k:\alpha(k)=i} \phi_k$



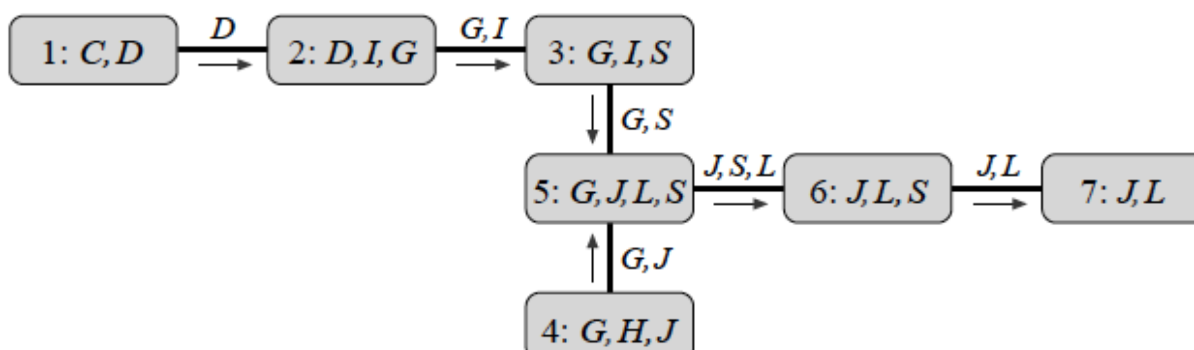
Example Cluster Graph



- Clusters $C_1 \ldots C_7$
  - $C_1 = \{C, D\}, C_2 \subseteq \{D, I, G\}, \ldots$
  - Edge $(1,2)$: $\{D\} \subseteq C_1 \cap C_2$,
  - Edge $(2, 3)$: $\{G, I\} \subseteq C_2 \cap C_3$
- Factors:

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

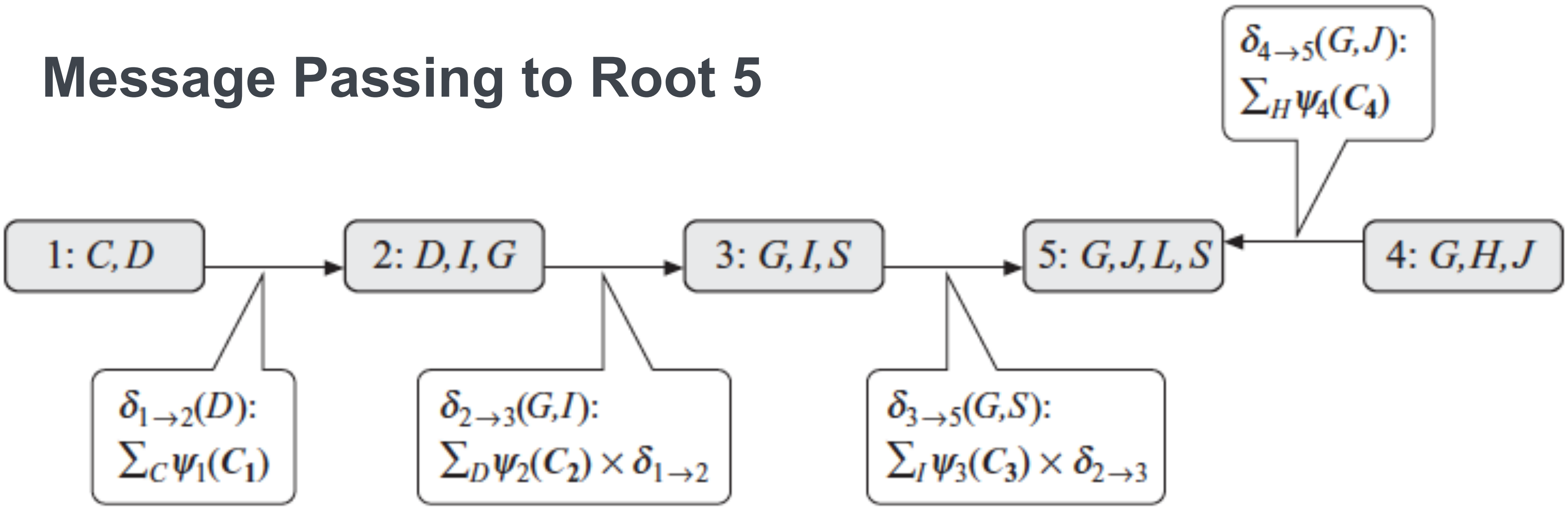
Factor assignmer



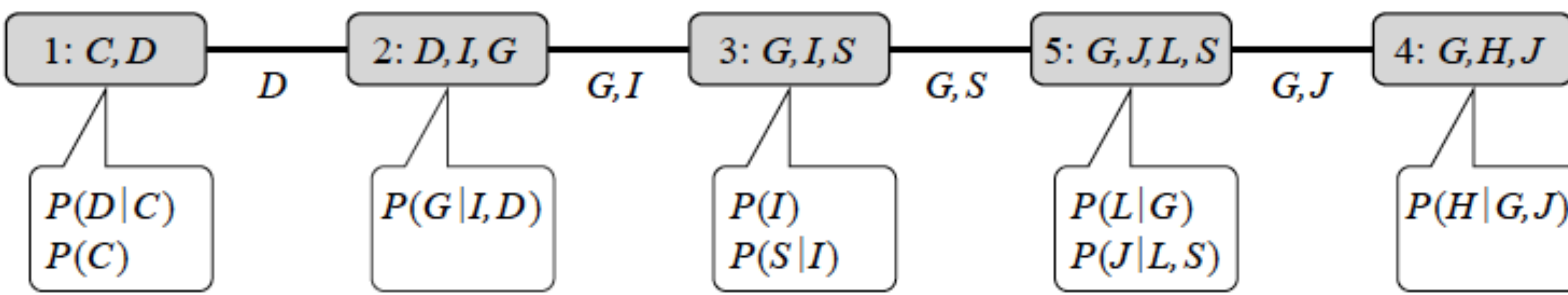
- Cluster factors:

$$\psi_1 = \phi_C\phi_D, \ldots$$

Message Passing to Root 5

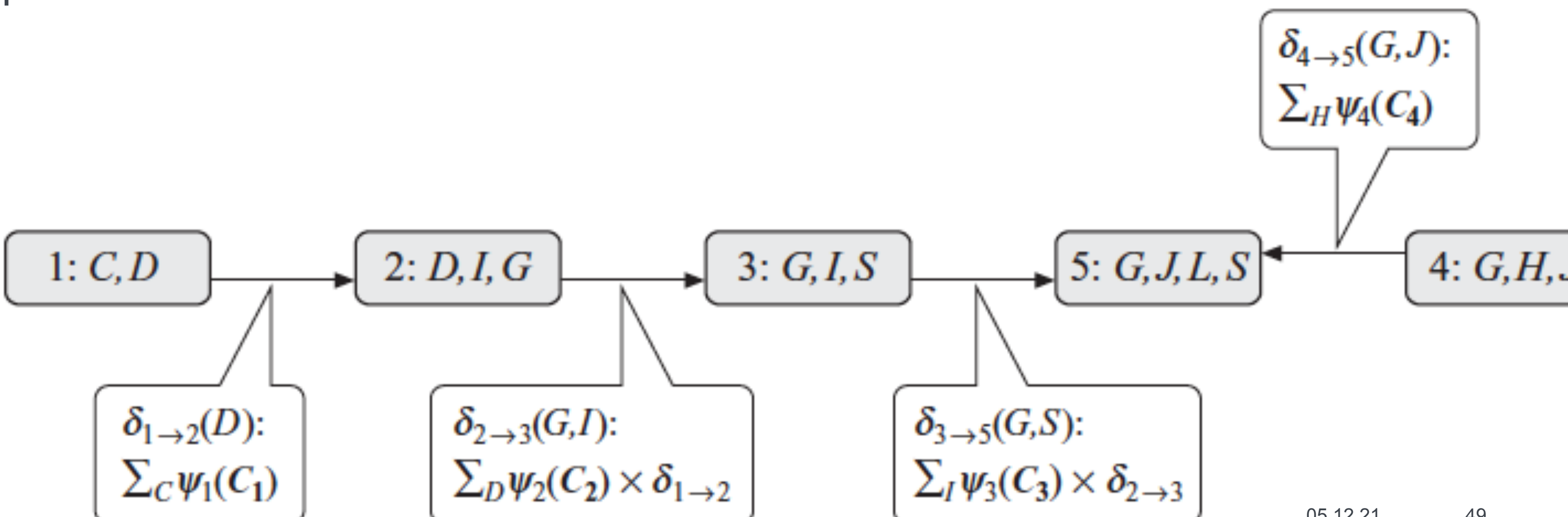


Alternative cluster graph:



What does it mean to pass a message?

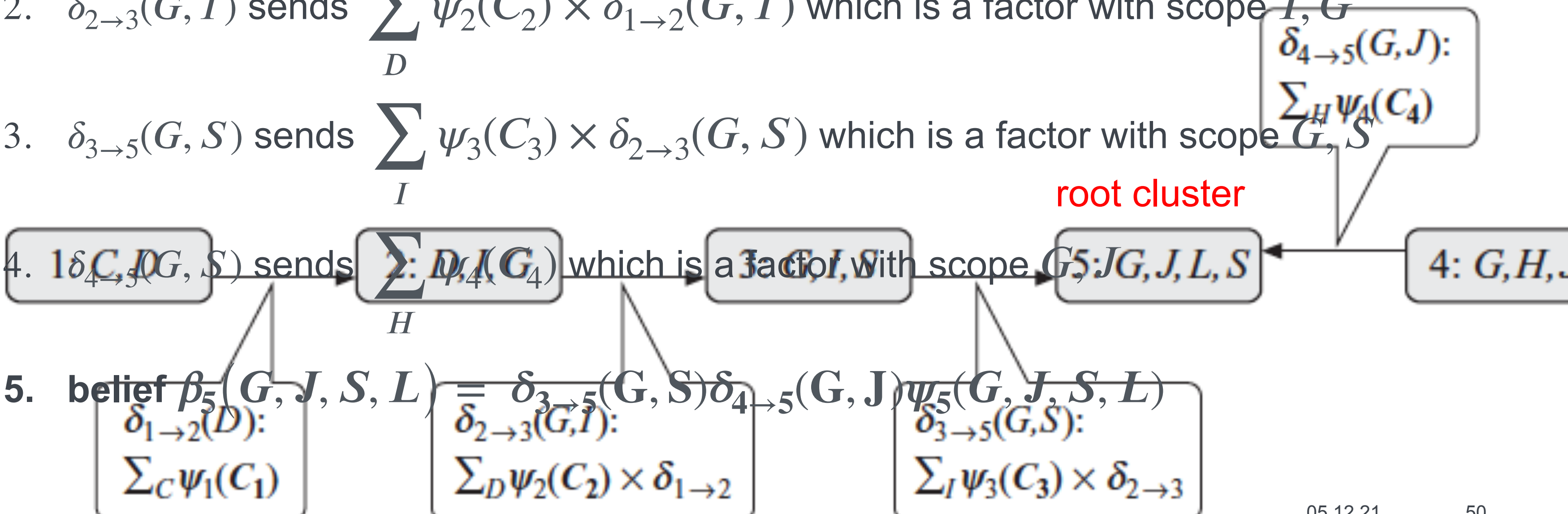
- Idea: compute local sum-product under outside influence
  - aggregate messages from other nodes
  - weave in local information
  - sum product
  - pass on to nodes where this information does not come from



What does it mean to pass a message?

- Message: $\delta_{\text{from}\rightarrow\text{to}}(\text{scope})$ is a factor with origin & target that share „ scope“
  - Example:

1. $\delta_{1\rightarrow 2}(D)$ sends $\sum_{C} \psi_1(C_1) = \sum_{C} \phi_C(C)\phi_D(C, D)$ which is a factor with scope D
2. $\delta_{2\rightarrow 3}(G, I)$ sends $\sum_{D} \psi_2(C_2) \times \delta_{1\rightarrow 2}(G, I)$ which is a factor with scope I, G
3. $\delta_{3\rightarrow 5}(G, S)$ sends $\sum_{I} \psi_3(C_3) \times \delta_{2\rightarrow 3}(G, S)$ which is a factor with scope G, S
4. $\delta_{4\rightarrow 5}(G, S)$ sends $\sum_{H} \psi_4(C_4)$ which is a factor with scope G, J
5. **belief** $\beta_5(G, J, S, L) = \delta_{3\rightarrow 5}(G, S)\delta_{4\rightarrow 5}(G, J)\psi_5(G, J, S, L)$



What happened?

- General:

$$\delta_{i\to j} = \underbrace{\sum_{C_i - S_{i,j}} \underbrace{\psi_i \underbrace{\prod_{k\in(\mathrm{neighbor}(i)-\{j\})} \delta_{k\to j}}_{\text{factor product}}}_{\text{marginalization (sum)}}}_{\text{variable elimination}}$$

Message passing in cluster trees $\triangleq$ variable elimination of $C_i - S_{i,j}$

Belief $\beta_r(C_r)$

- Let r index the root cluster
  - with C_r being the random variables in that cluster
  - $\beta_r(C_r)$ being computed as indicated on the previous slide
- Then:

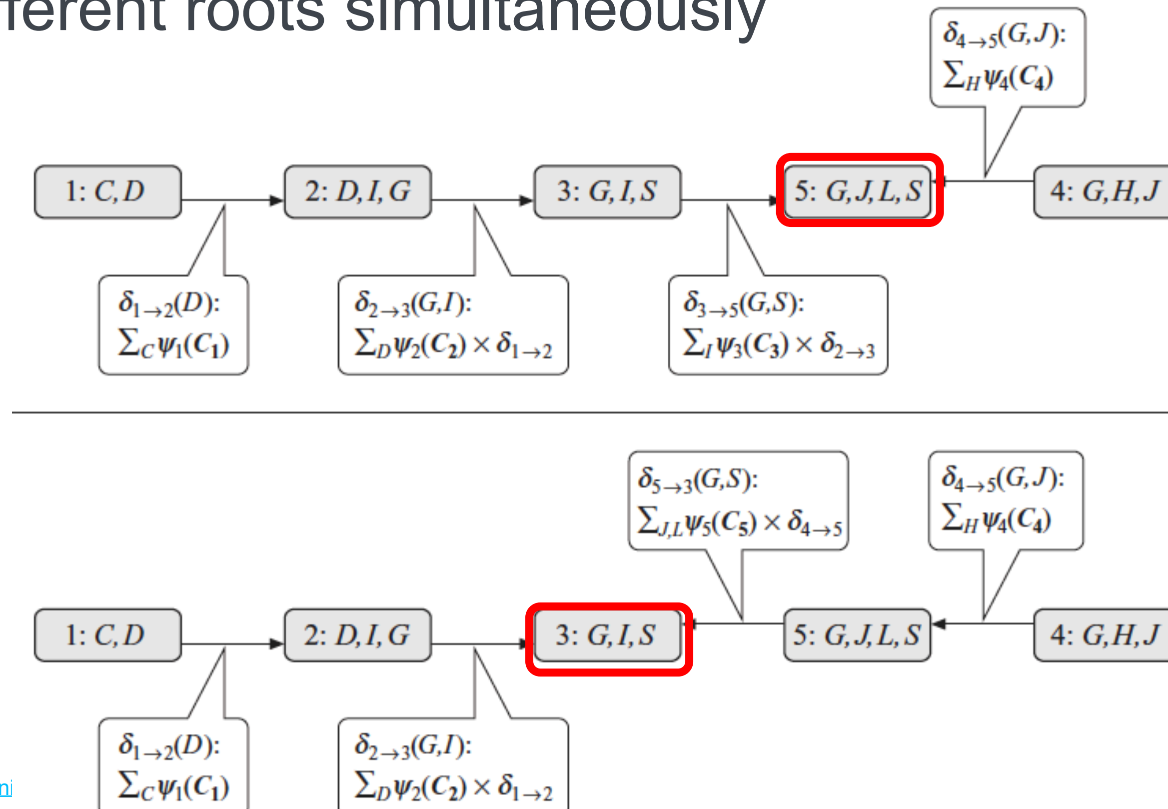
$$\beta_r(C_r) = \sum_{\mathscr{X}-C_r} \tilde{P}_{\Phi}(\mathscr{X})$$

evidence may be provided by working with reduced factors

- Bayesian network: $\beta_r(C_r) = P_{\mathscr{B}}(C_r)$
- Markov network: $\beta_r(C_r) = \tilde{P}_{\Phi}(C_r)$ with partition function defined by sum of all entries in $\beta_r(C_r)$

Compute posterior over every random variable

- Naive: separately for each random variable / for each cluster
- Better: compute for different roots simultaneously

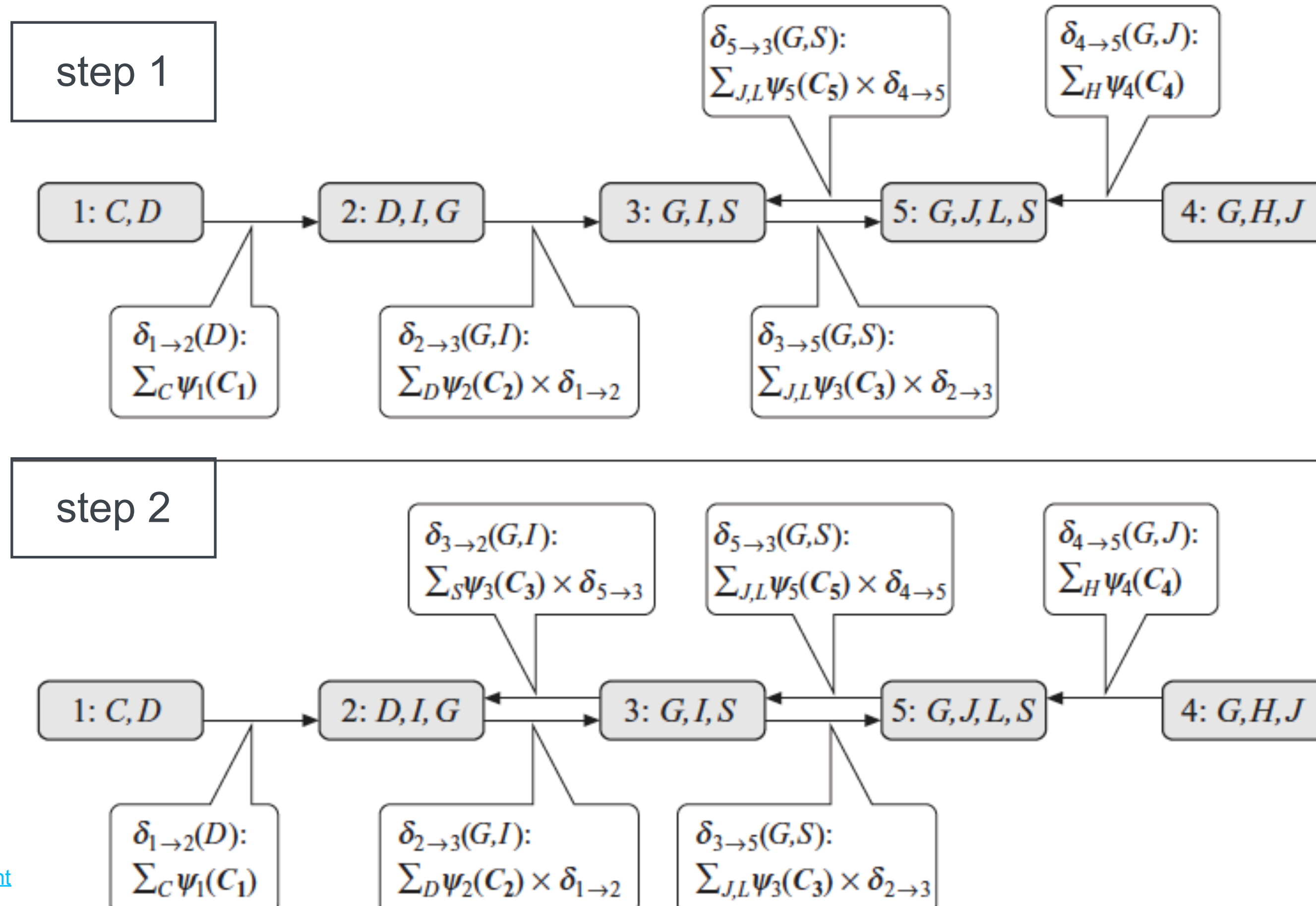


Upward and Downward Pass

- Better: compute for different roots simultaneously

Upward Pass:
same as before!

Downward Pass:
Make sure that content of message is not passed back to where it came from!



Question 5: Belief Propagation in Trees

6 Belief Propagation in Graphs

Generalization to Arbitrary Cluster Graphs

- No longer guarantees soundness
- The following properties have to hold (as they had to hold before):
  1. Family Preservation
  2. Running Intersection Property

Family Preservation

- Given set of factors Φ, we assign each ϕ_k to exactly one cluster $C_{\alpha(k)}$ such that $\mathrm{scope}(\phi_k) \subseteq C_{\alpha(k)}$

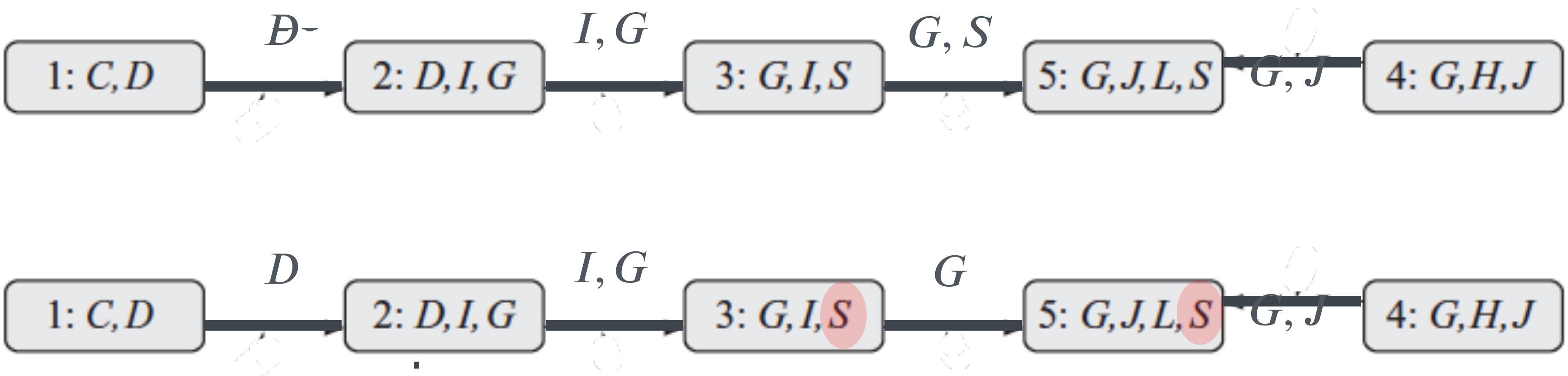
Mentioned before!

Running Intersection Property

- For each pair of clusters C_i and C_j and a variable $X \in C_i \cap C_j$, there exists a unique path between C_i and C_j for which all clusters and sepsets contain X

Running Intersection Property: Existence

- For each pair of clusters C_i and C_j and a variable $X \in C_i \cap C_j$, there exists a unique path between C_i and C_j for which all clusters and sepsets



Not mentioned before!

Running Intersection Property: Uniqueness

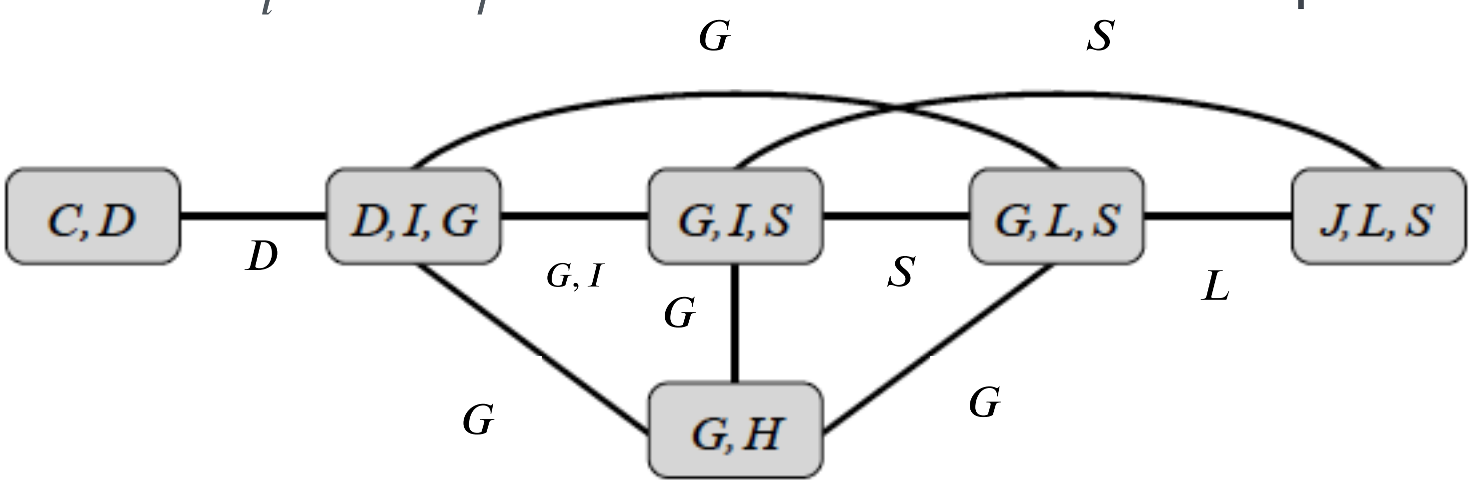
- For each pair of clusters C_i and C_j and a variable $X \in C_i \cap C_j$, there exists a unique path between C_i and C_j for which all clusters and sepsets

- Trivial for a tree!
- Non-trivial for a general graph!

Always holds in a tree!

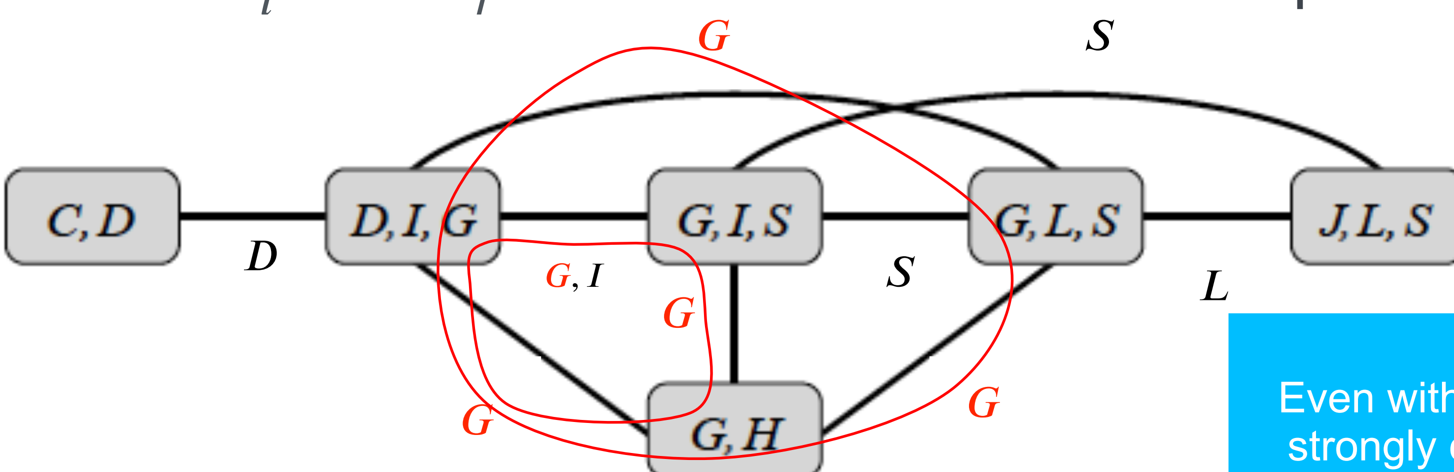
Running Intersection Property: Uniqueness

- For each pair of clusters C_i and C_j and a variable $X \in \ C_i \cap C_j$, there exists a unique path between C_i and C_i for which all clusters and sepsets



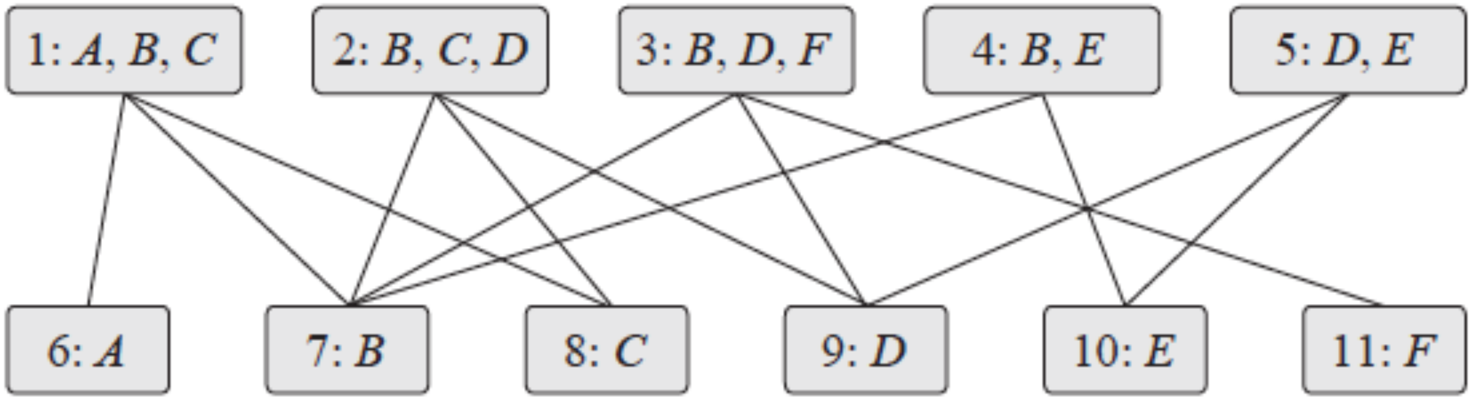
Running Intersection Property: Uniqueness

- For each pair of clusters C_i and C_j and a variable $X \in C_i \cap C_j$, there exists a unique path between C_i and C_j for which all clusters and sepsets



Even without loops: strongly correlated variables may create *pseudo-loops*

There is always a Cluster Graph: Bethe cluster graph



Information between different clusters is passed through univariate marginal distributions.
Interactions between variables are lost during propagations.

Generalized Belief Propagation Algorithm

Given factors Φ, cluster graph with clusters $\{C_1, \ldots, C_l\}$:

1. Assign each factor $\phi_k \in \Phi$ to exactly one cluster $C_{\alpha(k)}$
2. Construct potentials $\psi_i(C_i) = \prod_{k:\alpha(k)=i} \phi_k$
3. Initialize all messages to be unit factor $\mathbf{1}$
4. Repeat
   - Select edge (i, j) and pass message

$$\delta_{\mathrm{i}\to\mathrm{j}}\left(S_{i,j}\right) = \sum_{C_i - S_{i,j}} \psi_i \prod_{k \in (\mathrm{neighbors}(i) - \{j\})} \delta_{\mathrm{k}\to i}$$

5. Compute belief $\beta_i(C_i) = \psi_i \prod_{k \in \mathrm{neighbors}(i)} \delta_{\mathrm{k}\to i}$

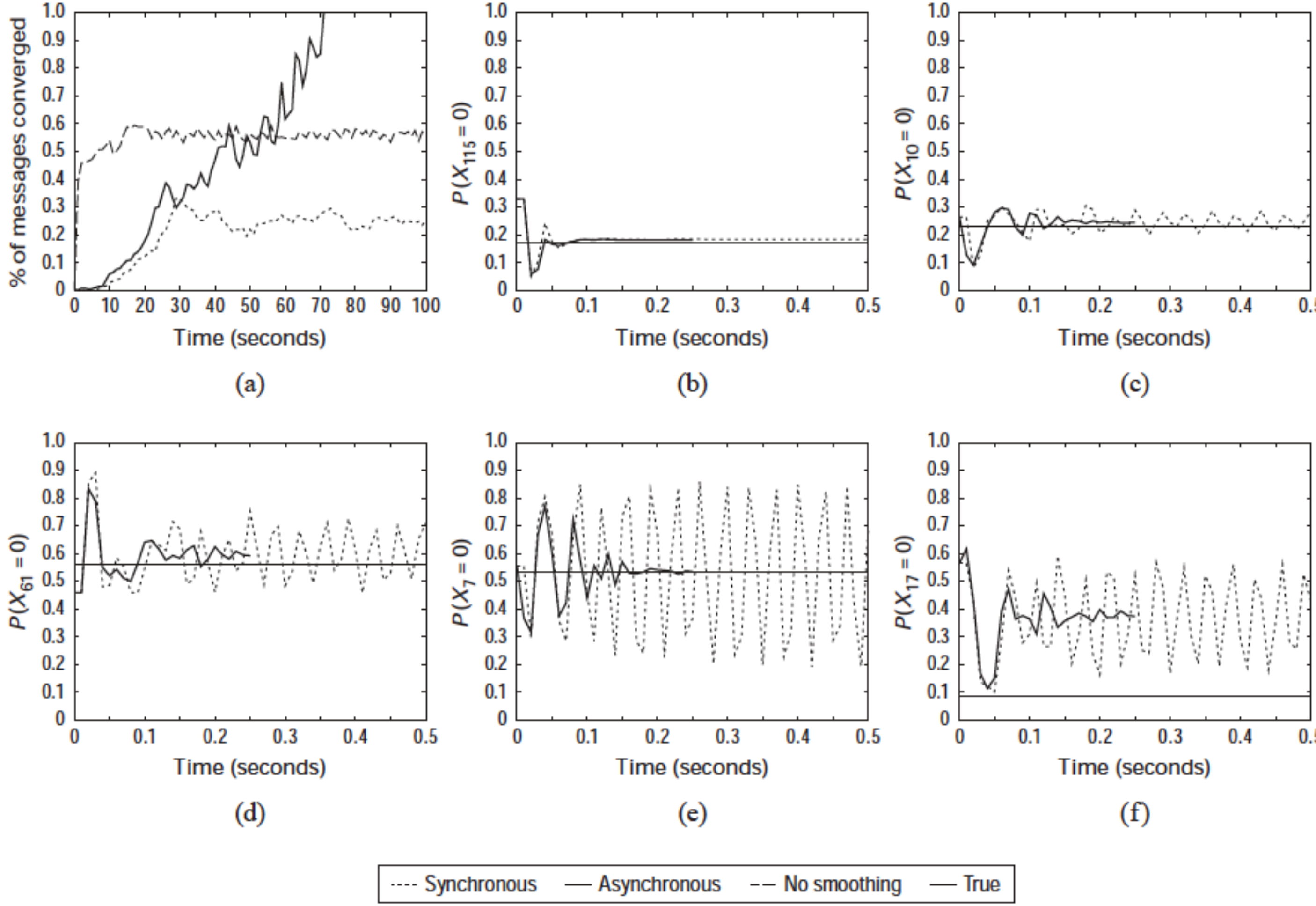


Figure 11.C.1 — Example of behavior of BP in practice on an 11×11 Ising grid. (a) Percentage of messages converged as a function of time for three different BP variants. (b) A marginal where both variants converge rapidly. (c–e) Marginals where the synchronous BP marginals oscillate around the asynchronous BP marginals. (f) A marginal where both variants are inaccurate.

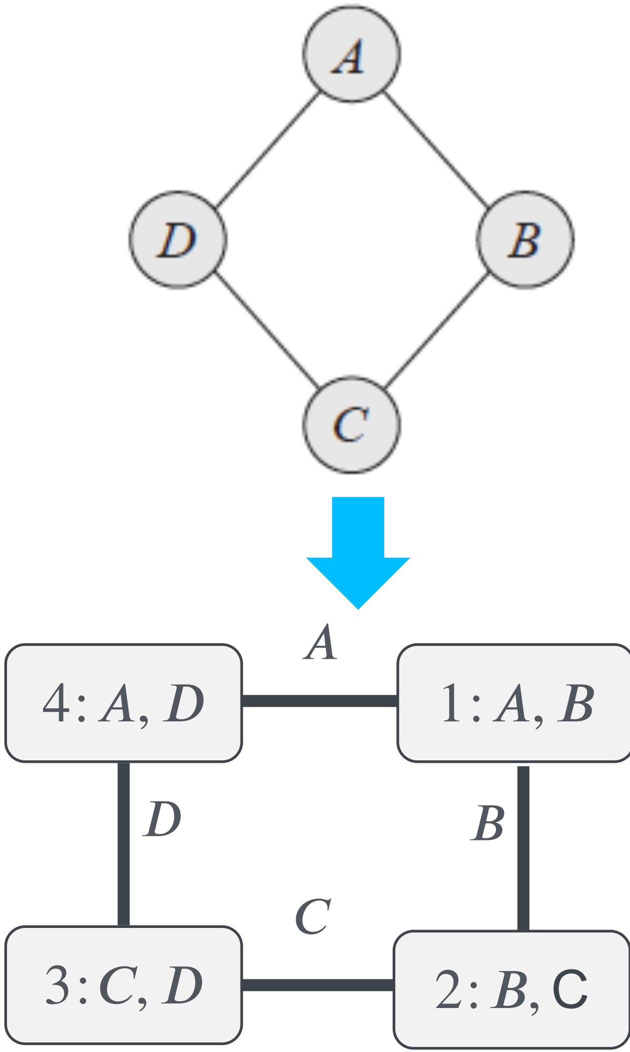
Calibration

- Cluster beliefs:

$$\beta_i(C_i) = \psi_i \prod_{k \in \text{neighbors}(i)} \delta_{\mathrm{k} \to i}$$

- A cluster graph is calibrated if every pair of adjacent clusters C_i, C_j agree on their sepset $S_{i,j}$:

$$\sum_{C_i - S_{i,j}} \beta_i(C_i) = \sum_{C_j - S_{i,j}} \beta_j(C_j)$$



05.12.21 70

Does Convergence imply Calibration?

- Convergence: $\delta_{i\to j}^{(t)}\left(S_{i,j}\right) = \delta_{i\to j}^{(t+1)}\left(S_{i,j}\right)$
- $\delta_{i\to j}^{(t+1)}\left(S_{i,j}\right) = \sum_{C_i - S_{i,j}} \left(\psi_i \prod_{k\in(\text{neighbors}(i)-\{j\})} \delta_{k\to i}^{(t)}\right) = \sum_{C_i - S_{i,j}} \frac{\beta_i\left(C_i\right)}{\delta_{j\to i}^{(t)}}$

With $\delta_{i\to j}^{(t)} = \delta_{i\to j}^{(t+1)} \Longrightarrow$ $\delta_{i\to j}^{(t)}\left(S_{i,j}\right)\delta_{j\to i}^{(t)}\left(S_{i,j}\right) = \sum_{C_i - S_{i,j}} \beta_i\left(C_i\right)$

Likewise: $\delta_{j\to i}^{(t)}\left(S_{i,j}\right)\delta_{i\to j}^{(t)}\left(S_{i,j}\right) = \sum_{C_j - S_{i,j}} \beta_j\left(C_j\right)$

$$\Longrightarrow \sum_{C_i - S_{i,j}} \beta_i\left(C_i\right) = \sum_{C_j - S_{i,j}} \beta_j\left(C_j\right)$$

Reparameterization

$$\beta_i(C_i) = \psi_i \prod_{k\in \text{neighbors}(i)} \delta_{k\to i} \qquad \mu_{i,j}\left(S_{i,j}\right) = \delta_{i\to j}\left(S_{i,j}\right)\delta_{j\to i}\left(S_{i,j}\right)$$

$$\frac{\prod_i \beta_i(C_i)}{\prod_{i,j} \mu_{i,j}(S_{i,j})} = \frac{\prod_i \beta_i(C_i)}{\prod_{i,j} \delta_{i\to j}(S_{i,j})\delta_{j\to i}(S_{i,j})} =$$

$$= \frac{\prod_i \psi_i \prod_{j\in \text{neighbors}(i)} \delta_{j\to i}}{\prod_{i,j} \delta_{i\to j}} = \prod_i \psi_i = \tilde{P}(\Phi)$$

Message passing does not delete information.
Message passing reparameterizes the factors.

Summary

- **Cluster graphs** must satisfy two properties
  - family preservation: allows Φ to be encoded
  - running intersection: connects all information about any variable, but without feedback loops
- **Bethe cluster graph** is often the first default
  - Richer cluster graph structures can offer different tradeoffs wrt computational cost and preservation of dependencies
- **Cluster graph beliefs** represent an alternative, calibrated parameterization of the original unnormalized density

Question 6: Belief Propagation in Graphs

Universität Stuttgart
IPVS

Thank you!

Steffen Staab

E-Mail Steffen.staab@ipvs.uni-stuttgart.de
Telefon +49 (0) 711 685- To be defined
www. ipvs.uni-stuttgart.de/departments/ac/

Universität Stuttgart
Analytic Computing, IPVS
Universitätsstraße 32, 50569 Stuttgart