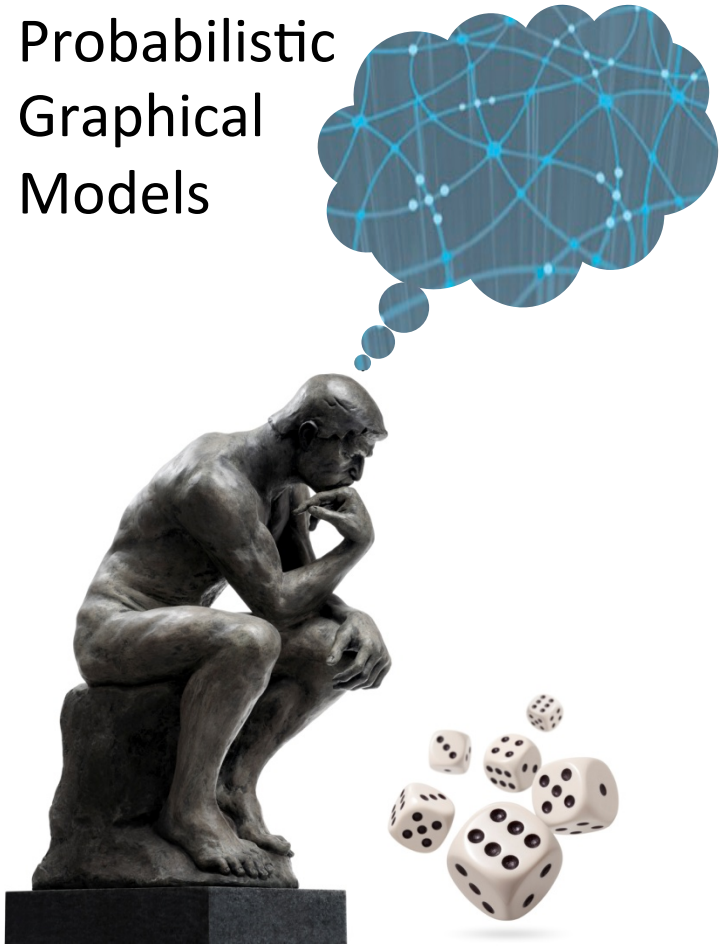
Probabilistic Graphical Models

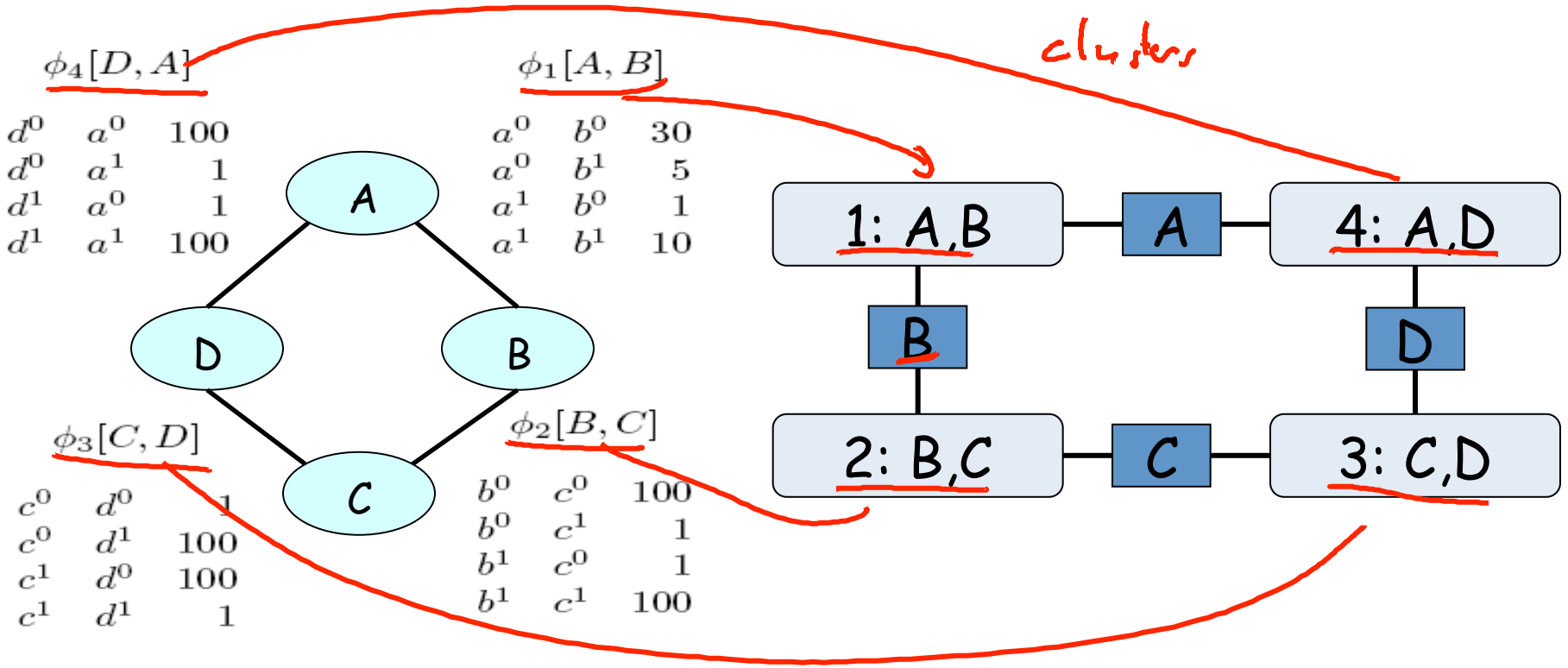


Inference

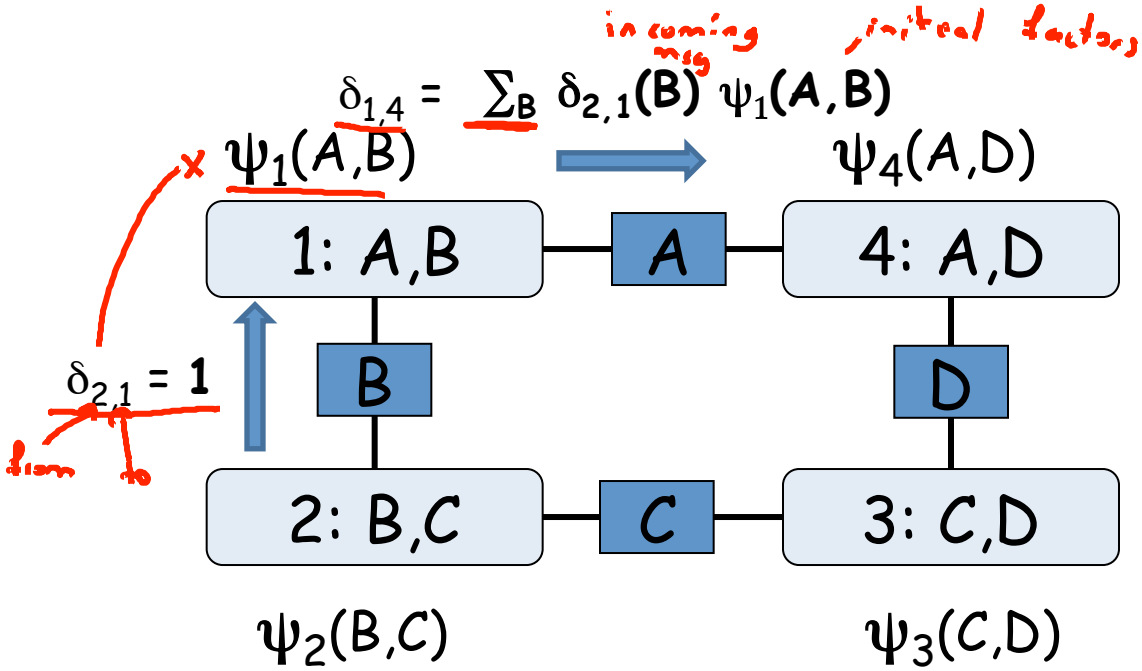
Message Passing

Belief Propagation Algorithm

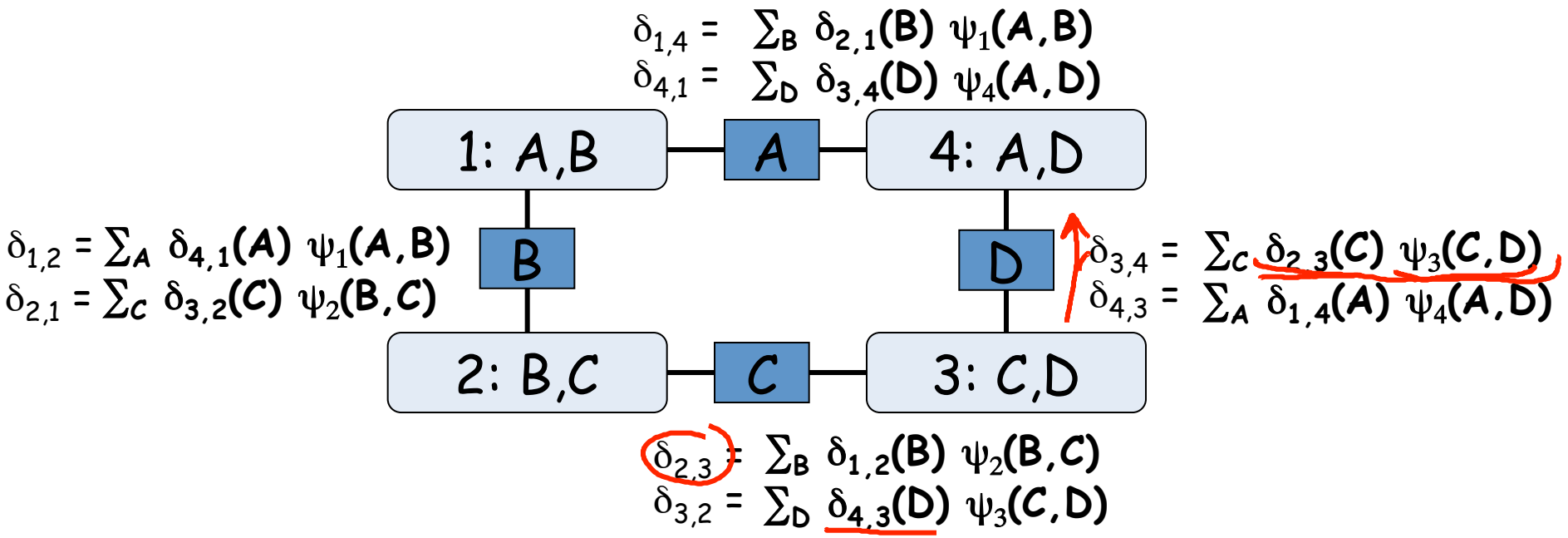
Cluster Graph



Passing Messages



Passing Messages



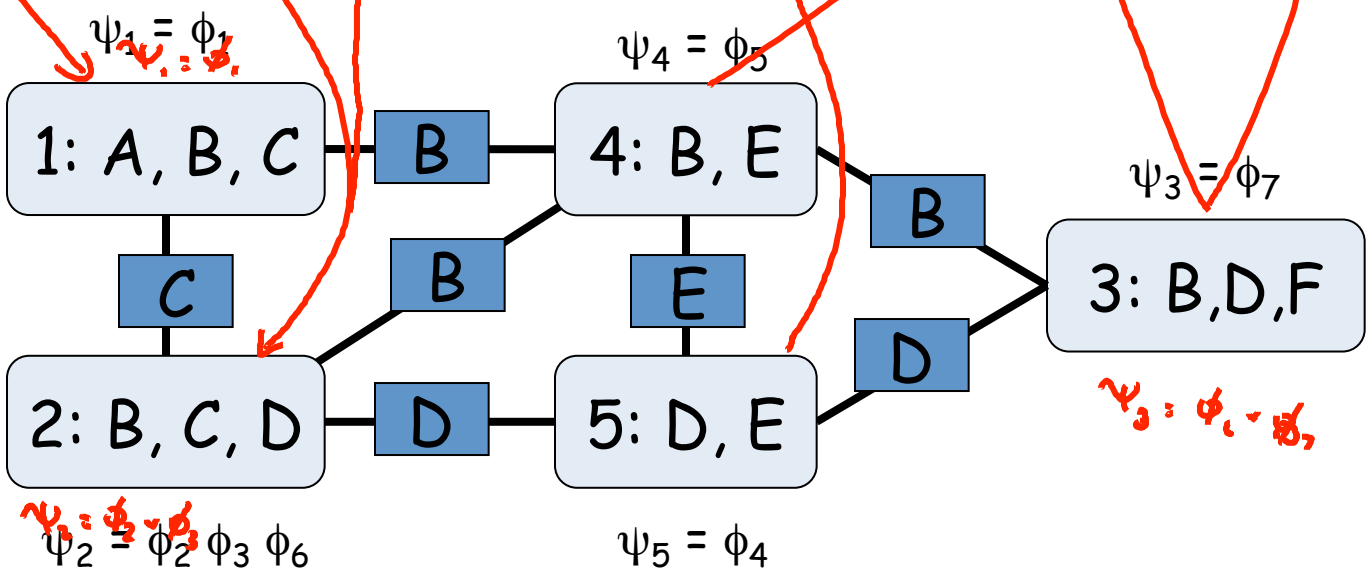
Daphne Koller

Cluster Graphs

- Undirected graph such that:
  - nodes are clusters $\boldsymbol{C}_i \subseteq \{X_1,\ldots,X_n\}$ — subsets of variables
  - edge between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ associated with sepset $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$ ← variables that they talk about
- Given set of factors Φ, we assign each ϕ_k to a cluster $\boldsymbol{C}_{\alpha(k)}$ s.t. Scope$[\phi_k] \subseteq \boldsymbol{C}_{\alpha(k)}$
- Define $$\psi_i(C_i) = \prod_{k:\alpha(k)=i} \phi_k$$ — subset; all factors assigned to it

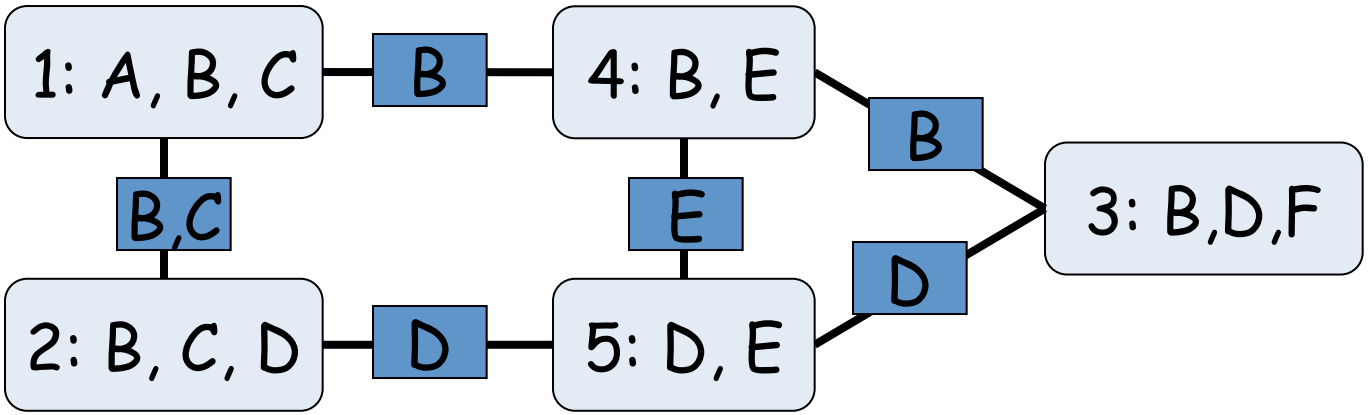
Example Cluster Graph

$\phi_1(A,B,C), \phi_2(B,C), \phi_3(B,D), \phi_4(D,E), \phi_5(B,E), \phi_6(B,D), \phi_7(B,D,F)$



Different Cluster Graph

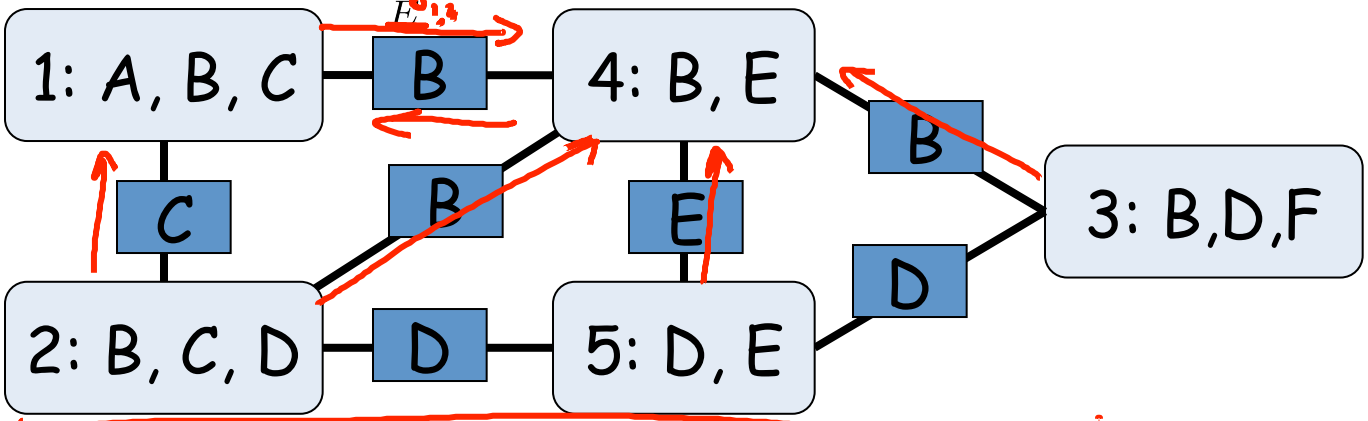
$$\phi_1(A,B,C), \phi_2(B,C), \phi_3(B,D), \phi_4(D,E), \phi_5(B,E), \phi_6(B,D), \phi_7(B,D,F)$$



Message Passing

$$\delta_{1\to 4}(B) = \sum_{A,C} \psi_1(A,B,C)\delta_{2\to 1}(C)$$

$$\delta_{4\to 1}(B) = \sum_{E} \psi_4(B,E) \times \delta_{2\to 4}(B) \times \delta_{5\to 4}(E) \times \delta_{3\to 4}(B)$$



$$\delta_{i\to j}(\boldsymbol{S}_{i,j}) = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \times \prod_{k\in(\mathcal{N}_i - \{j\})} \delta_{k\to i}$$

Belief Propagation Algorithm

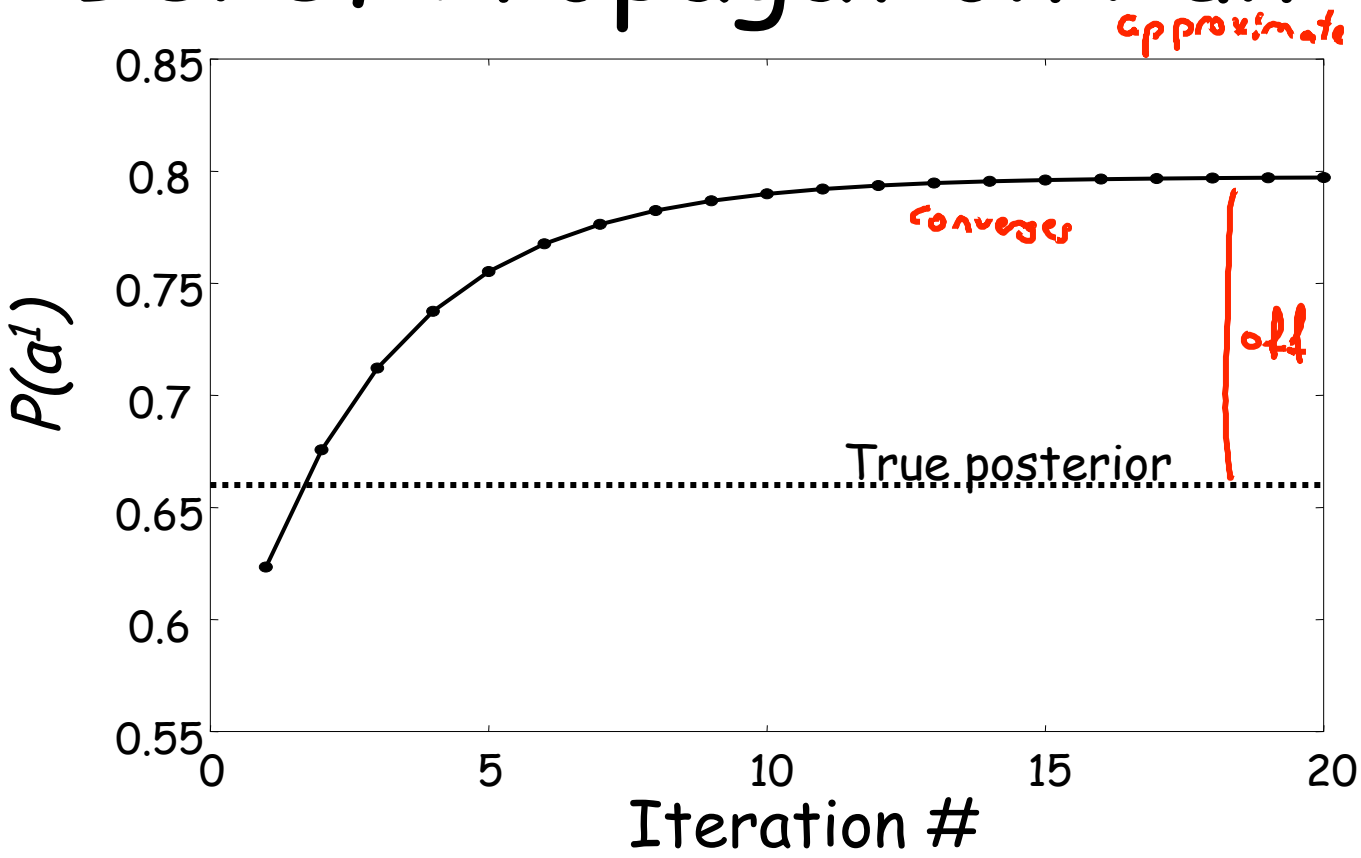
- Assign each factor $\phi_k \in \Phi$ to a cluster $C_{\alpha(k)}$
- Construct initial potentials $\psi_i(C_i) = \prod_{k:\alpha(k)=i} \phi_k$
- Initialize all messages to be **1**
- Repeat until when?
  - Select edge (i,j) and pass message round robin

$$\delta_{i \to j}(S_{i,j}) = \sum_{C_i - S_{i,j}} \psi_i \times \prod_{k \in (\mathcal{N}_i - \{j\})} \delta_{k \to i}$$

- Compute $\beta_i(C_i) = \psi_i \times \prod_{k \in \mathcal{N}_i} \delta_{k \to i}$

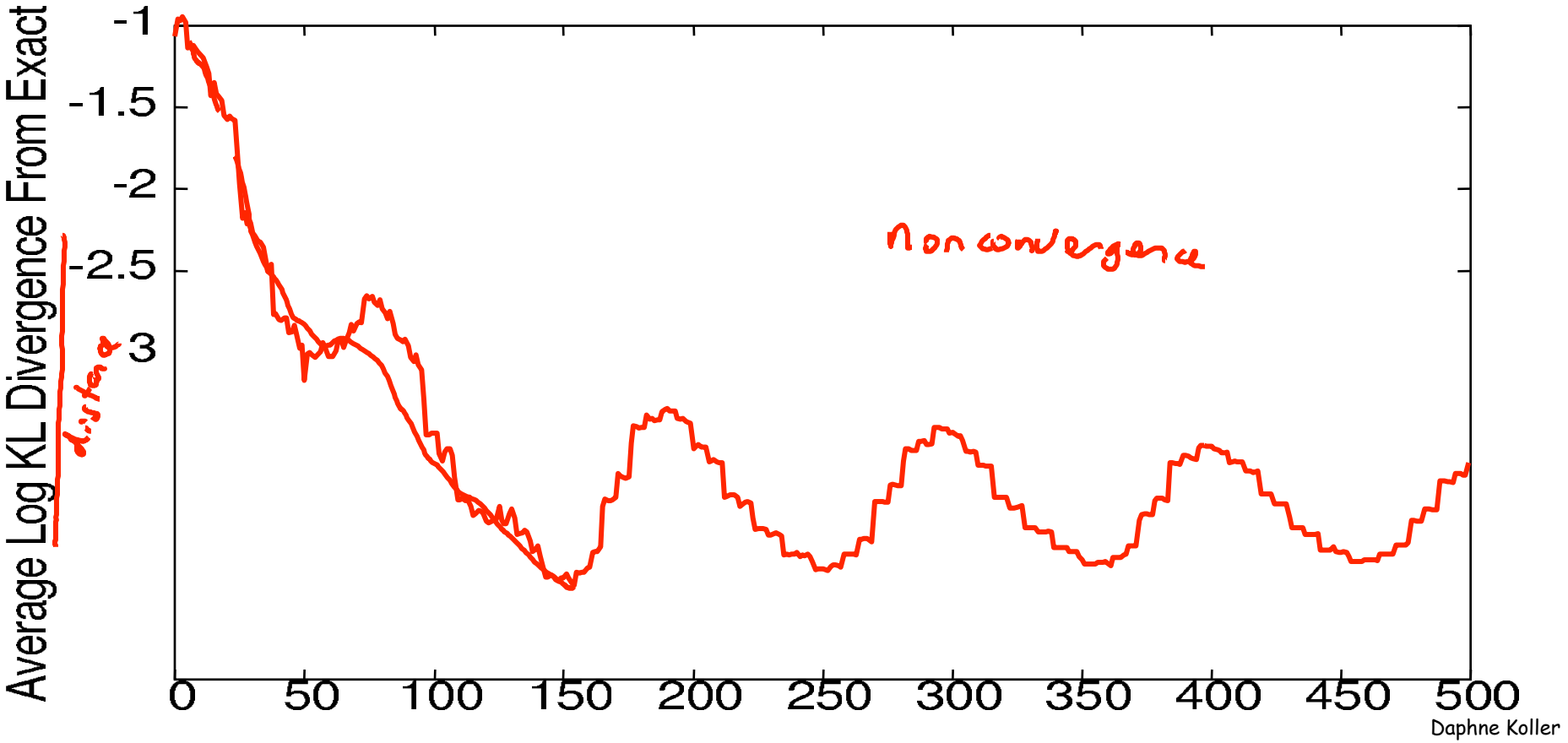
Beliefs — all neighbors

Belief Propagation Run



Daphne Koller

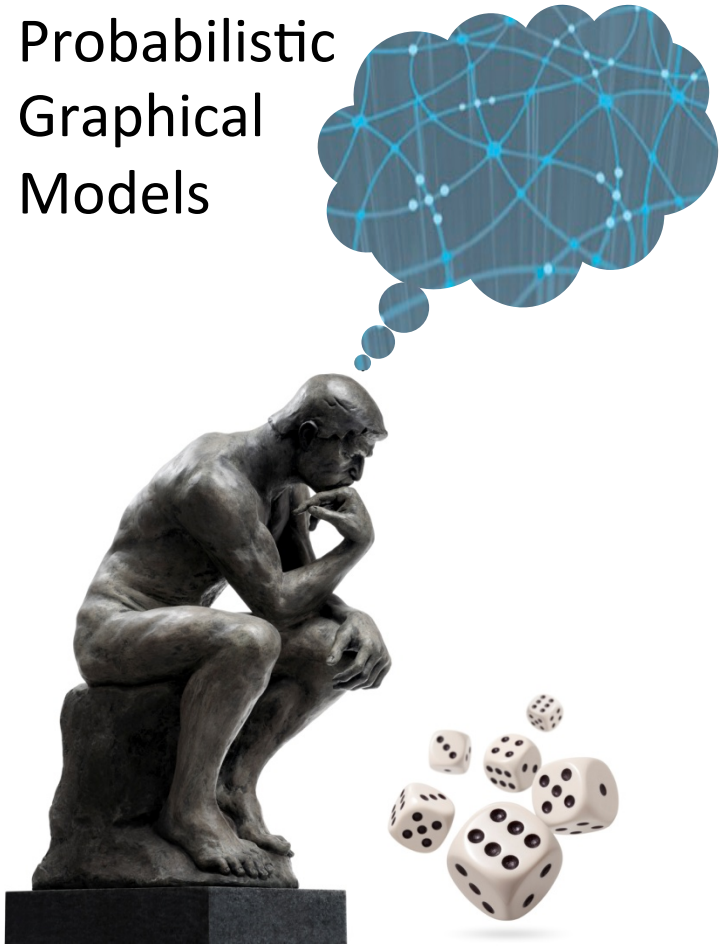
Different BP Run



Summary

- Graph of clusters connected by sepsets
- Adjacent clusters pass information to each other about variables in sepset
  - Message from i to j summarizes everything i knows, except information obtained from j
- Algorithm may not converge
- The resulting beliefs are pseudo-marginals
- Nevertheless, very useful in practice

Probabilistic Graphical Models



Inference

Message Passing

Cluster Graph Properties

Cluster Graphs

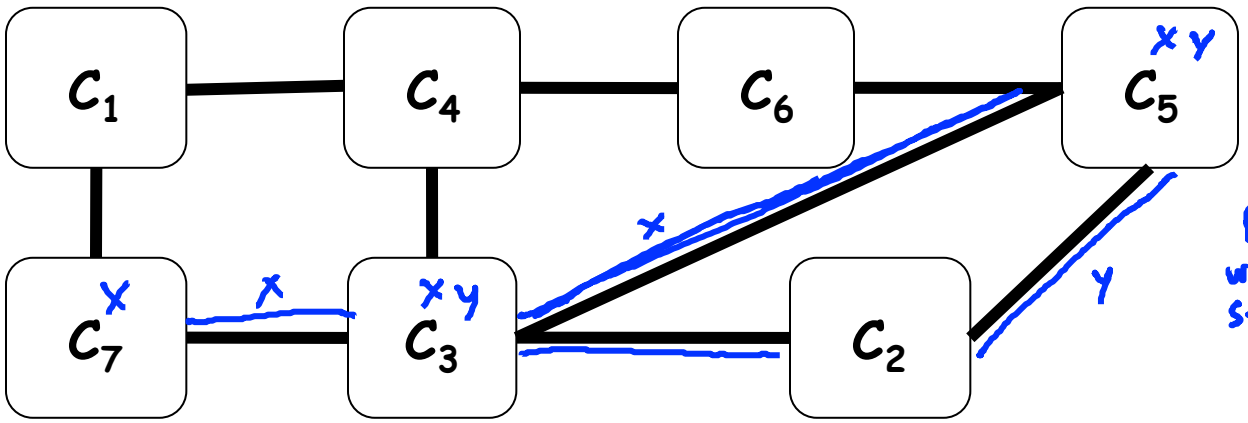
- Undirected graph such that:
  - nodes are clusters $\boldsymbol{C}_i \subseteq \{X_1,\ldots,X_n\}$
  - edge between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ associated with sepset $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$

Family Preservation

- Given set of factors Φ, we assign each ϕ_k to a cluster $\boldsymbol{C}_{\alpha(k)}$ s.t. $\text{Scope}[\phi_k] \subseteq \boldsymbol{C}_{\alpha(k)}$
  cluster can accomodate ϕ_k
- For each factor $\phi_k \in \Phi$, there exists a cluster $\boldsymbol{C}_i$ s.t. $\text{Scope}[\phi_k] \subseteq \boldsymbol{C}_i$ ← accomodates ϕ_k

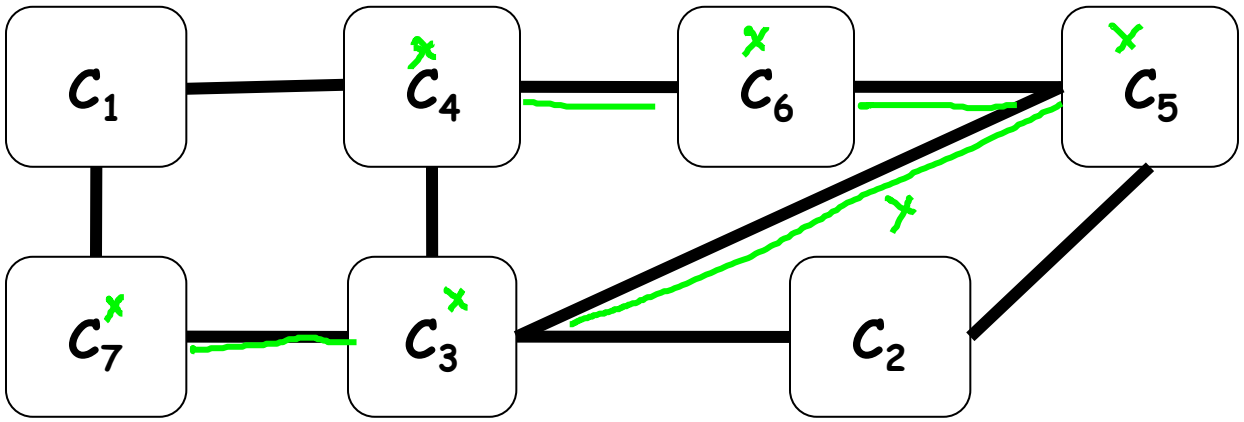
Running Intersection Property

- For each pair of clusters C_i, C_j and 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

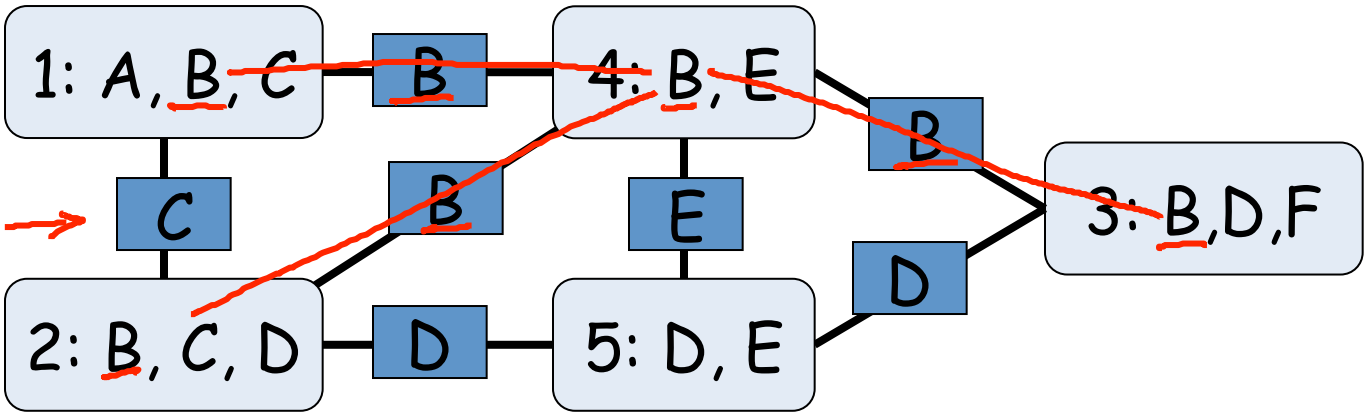


Running Intersection Property

- Equivalently: For any X, the set of clusters and sepsets containing X forms a tree

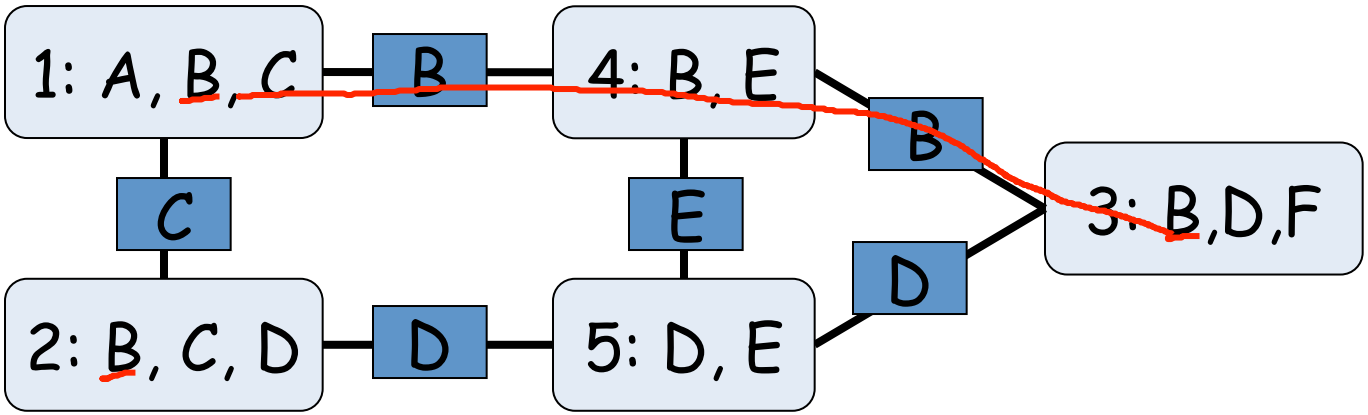


Example Cluster Graph



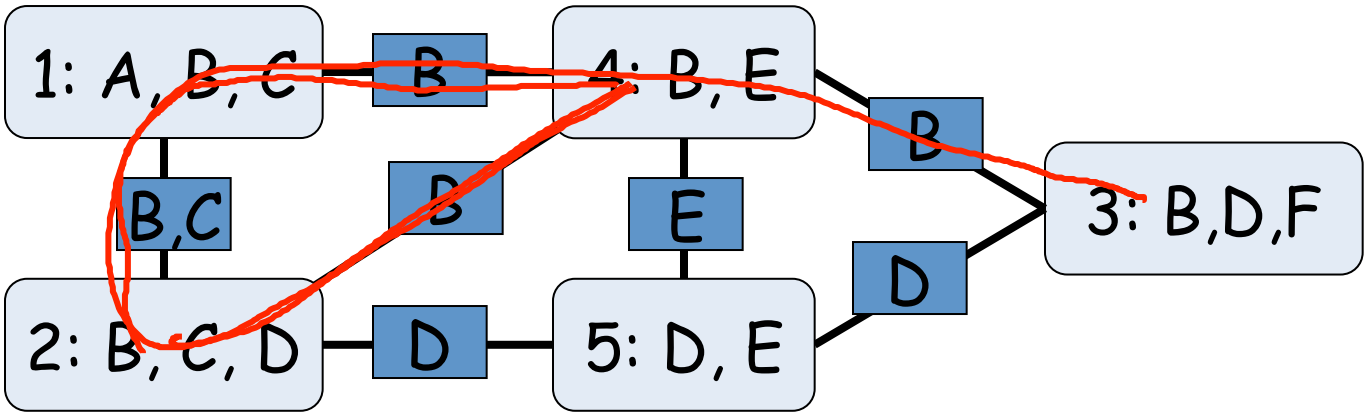
Illegal Cluster Graph I

violates existance

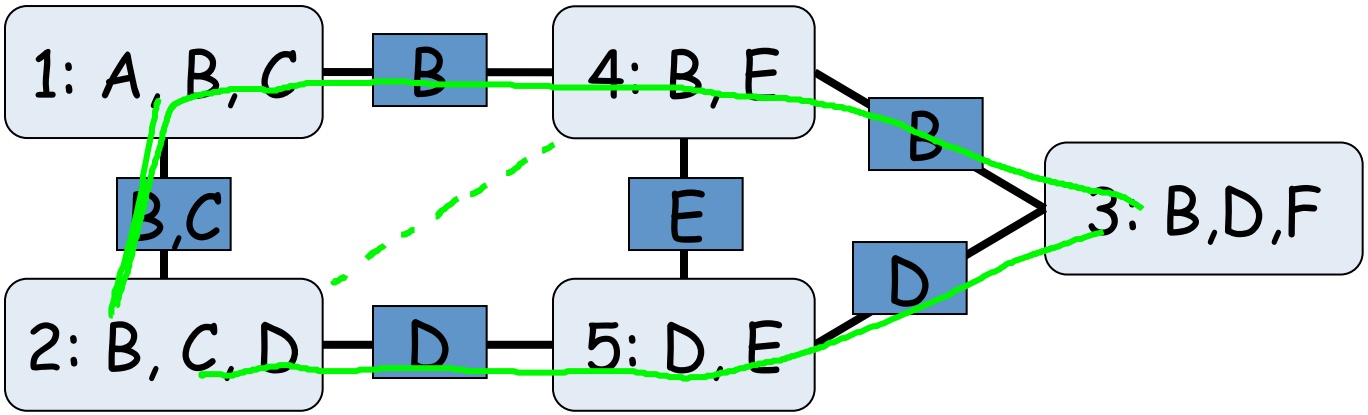


Illegal Cluster Graph II

violates uniqueness



Alternative Legal Cluster Graph

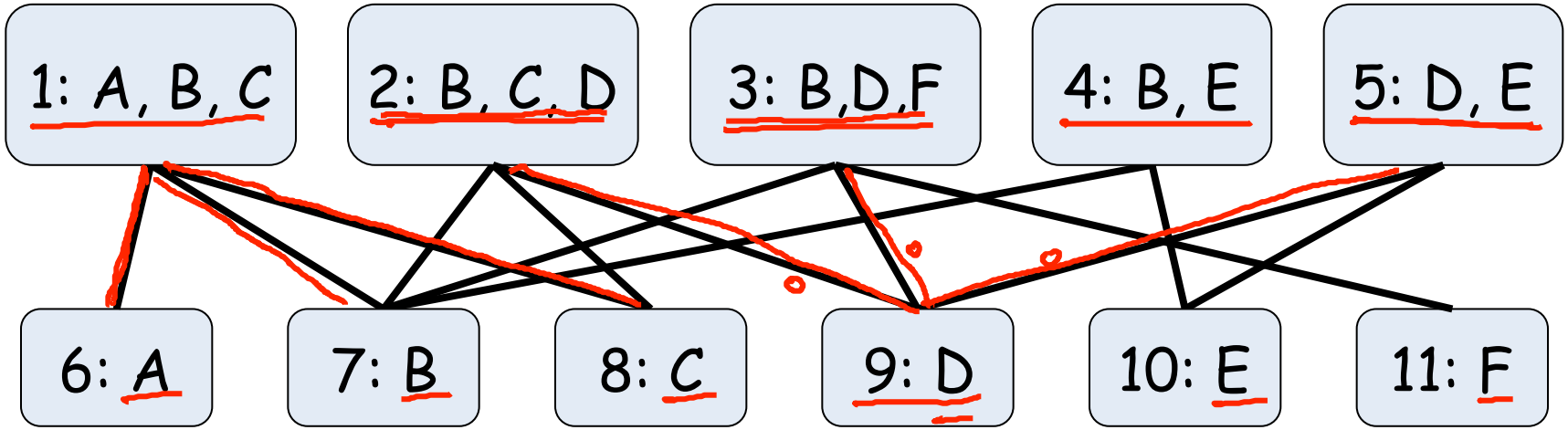


Daphne Koller

Bethe Cluster Graph

big clusters = factor in Φ

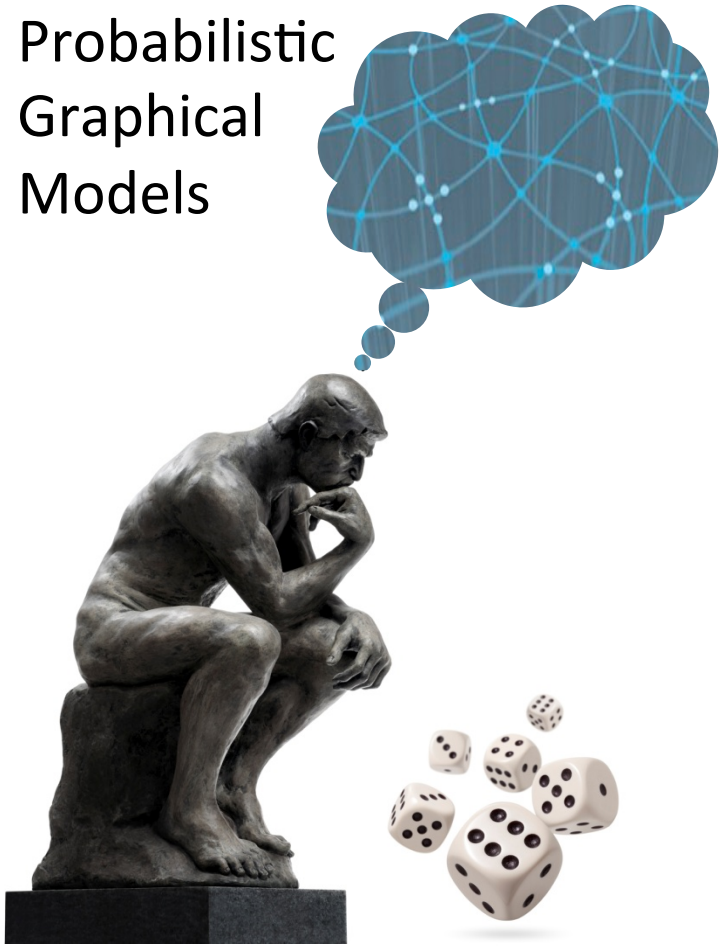
- For each $\phi_k \in \Phi$, a factor cluster C_k = Scope[ϕ_k]
- For each X_i a singleton cluster $\{X_i\}$
- Edge C_k — X_i if $X_i \in C_k$



Summary

- Cluster graph 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 first default
- Richer cluster graph structures can offer different tradeoffs wrt computational cost and preservation of dependencies

Probabilistic Graphical Models



Inference

Message Passing

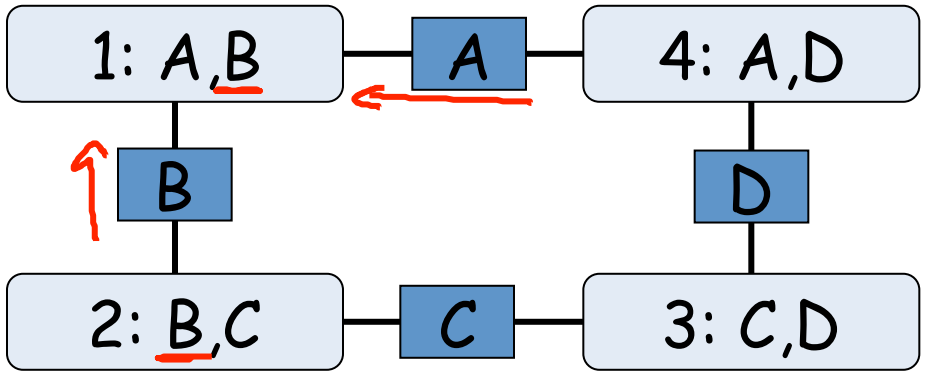
Properties of BP Algorithm

Calibration

$$\beta_1(A,B) = \psi_1(A,B) \times \delta_{4\to 1}(A) \times \delta_{2\to 1}(B)$$

- Cluster beliefs:

$$\beta_i(\boldsymbol{C}_i) = \psi_i \times \prod_{k \in \mathcal{N}_i} \delta_{k\to i}$$



- A cluster graph is calibrated if every pair of adjacent clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ agree on their sepset $\boldsymbol{S}_{i,j}$

$$\sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i(\boldsymbol{C}_i) = \sum_{\boldsymbol{C}_j - \boldsymbol{S}_{i,j}} \beta_j(\boldsymbol{C}_j)$$

sepset $S_{i,j}$

Convergence ⇨ Calibration

- Convergence: $\delta_{i\to j}(\boldsymbol{S}_{i,j}) = \delta'_{i\to j}(\boldsymbol{S}_{i,j})$

$$\beta_i(\boldsymbol{C}_i) = \psi_i \times \prod_{k\in\mathcal{N}_i} \delta_{k\to i}$$

all msgs

$$\delta'_{i\to j}(\boldsymbol{S}_{i,j}) = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \left(\psi_i \times \prod_{k\in(\mathcal{N}_i - \{j\})} \delta_{k\to i} \right) = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \frac{\beta_i(\boldsymbol{C}_i)}{\delta_{j\to i}(\boldsymbol{S}_{i,j})} =$$

msgs except from j

$$\delta_{j\to i}(\boldsymbol{S}_{i,j})\delta_{i\to j}(\boldsymbol{S}_{i,j}) = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i(\boldsymbol{C}_i)$$

calibration

$$\sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i(\boldsymbol{C}_i) = \sum_{\boldsymbol{C}_j - \boldsymbol{S}_{i,j}} \beta_j(\boldsymbol{C}_j)$$

$$\delta_{j\to i}(\boldsymbol{S}_{i,j})\delta_{i\to j}(\boldsymbol{S}_{i,j}) = \sum_{\boldsymbol{C}_j - \boldsymbol{S}_{i,j}} \beta_j(\boldsymbol{C}_j)$$

$$\mu_{i,j}(\boldsymbol{S}_{i,j}) = \delta_{j\to i}\delta_{i\to j} = \sum_{\boldsymbol{C}_j - \boldsymbol{S}_{i,j}} \beta_j(\boldsymbol{C}_j)$$

sepset beliefs

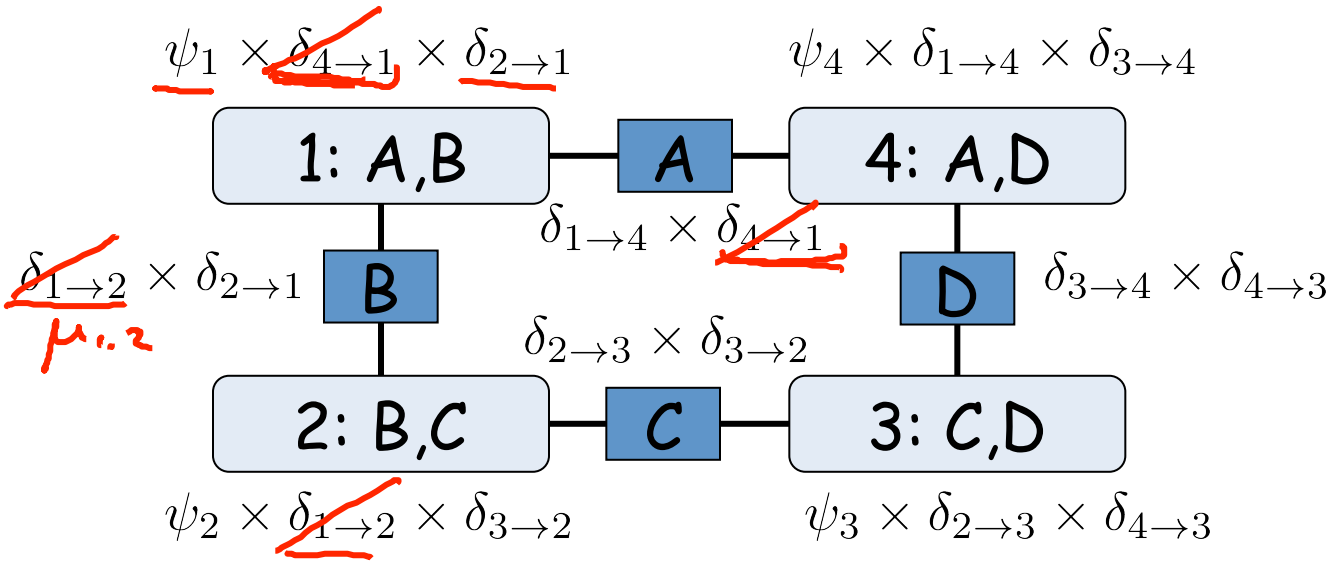
Reparameterization

$$\beta_i(\boldsymbol{C}_i) = \psi_i \times \prod_{k \in \mathcal{N}_i} \delta_{k \to i}$$

$$\mu_{i,j}(\boldsymbol{S}_{i,j}) = \delta_{j \to i} \delta_{i \to j}$$

sepset belief

$$\frac{\prod_i \beta_i}{\prod_{i,j} \mu_{i,j}}$$



Reparameterization

no information loss

$$\beta_i(\boldsymbol{C}_i) = \psi_i \times \prod_{k \in \mathcal{N}_i} \delta_{k \to i} \qquad \mu_{i,j}(\boldsymbol{S}_{i,j}) = \delta_{j \to i} \delta_{i \to j}$$

$$\frac{\prod_i \beta_i}{\prod_{i,j} \mu_{i,j}} = \frac{\prod_i \left(\psi_i \prod_{j \in \mathcal{N}_i} \delta_{j \to i} \right)}{\prod_{i,j} \delta_{i \to j}}$$

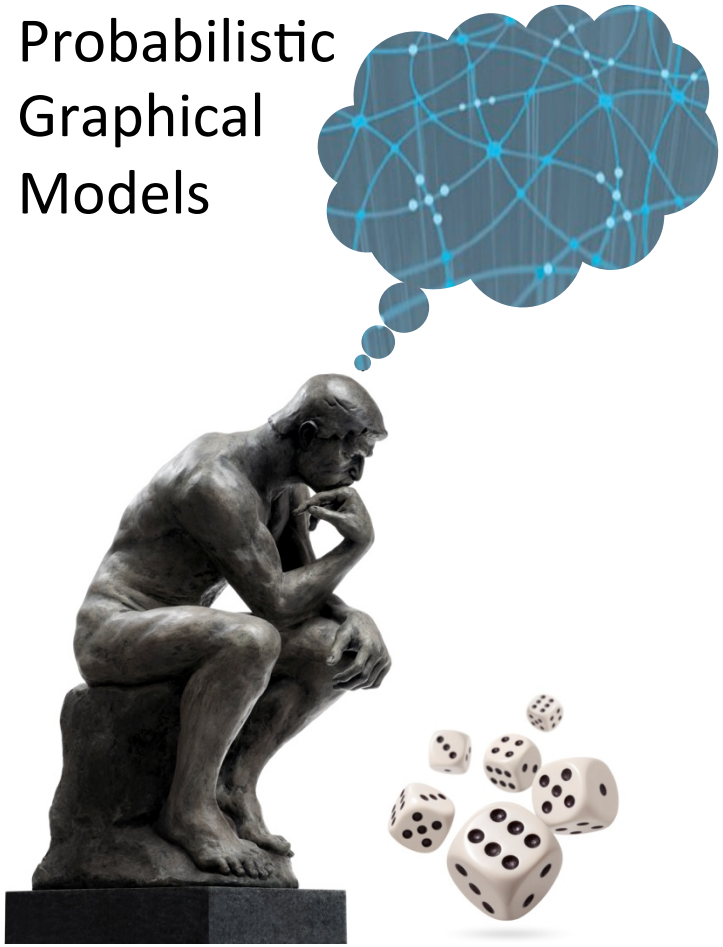
$$= \prod_i \psi_i = \tilde{P}_\Phi(X_1, \ldots, X_n)$$

unnormalized measure

Summary

- At convergence of BP, cluster graph beliefs are calibrated:
  - beliefs at adjacent clusters agree on sepsets
- Cluster graph beliefs are an alternative, calibrated parameterization of the original unnormalized density
  - No information is lost by message passing

Probabilistic Graphical Models

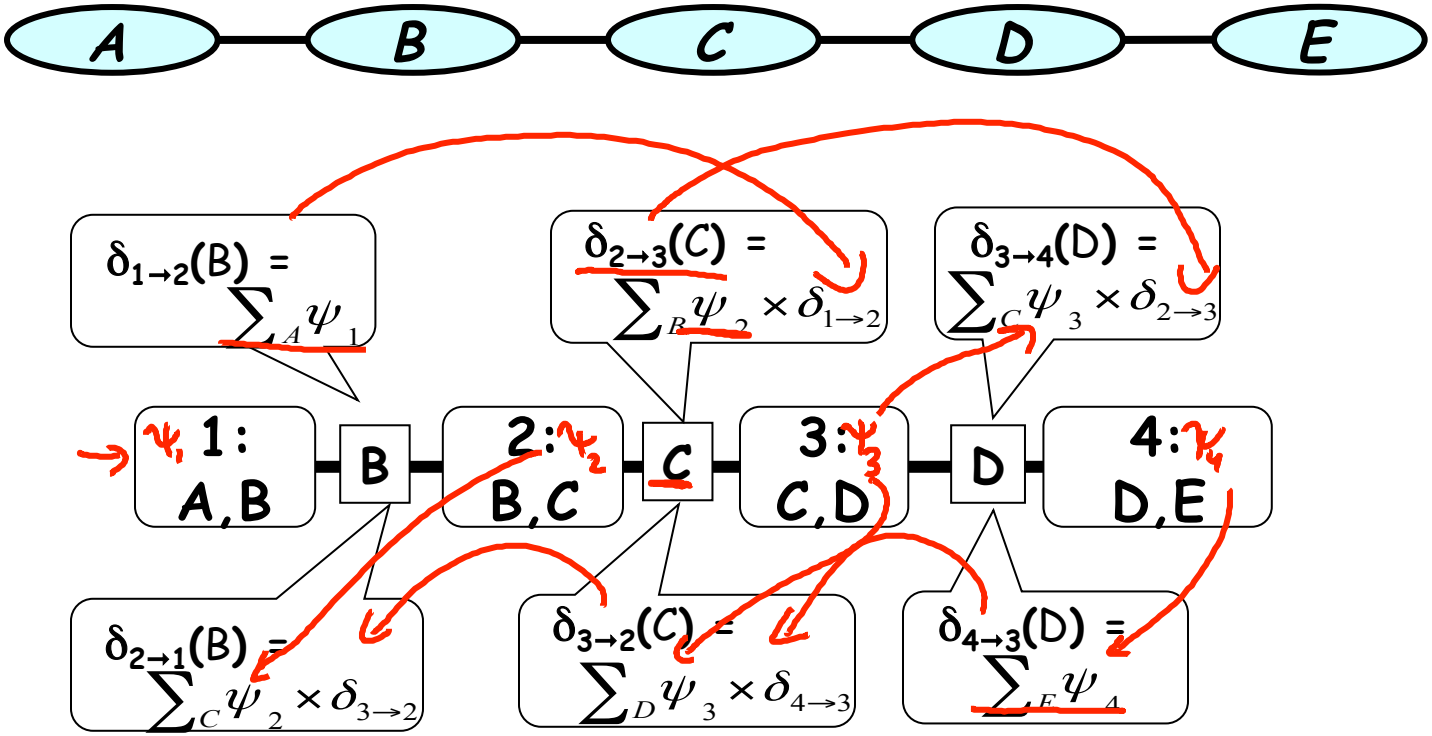


Inference

Message Passing

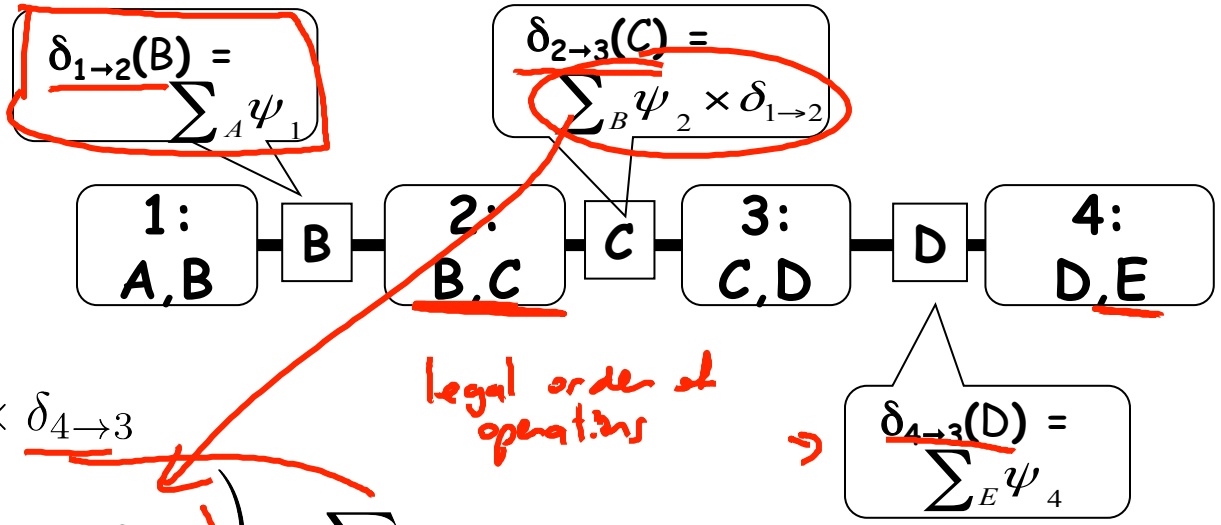
Clique Tree Algorithm & Correctness

Message Passing in Trees



Daphne Koller

Correctness



legal order of operations

$$\beta_3(C,D) = \psi_3 \times \delta_{2\to 3} \times \delta_{4\to 3}$$
$$= \psi_3 \times \left(\sum_B \psi_2 \times \delta_{1\to 2}\right) \times \sum_E \psi_4$$
$$= \psi_3 \times \left(\sum_B \psi_2 \times \sum_A \psi_1\right) \times \sum_E \psi_4$$

product of factors marginalized out unnecessary variables

Daphne Koller

Clique Tree

- Undirected tree such that:
  - nodes are clusters $\boldsymbol{C}_i \subseteq \{X_1,\ldots,X_n\}$
  - edge between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ associated with sepset $\boldsymbol{S}_{i,j} = \boldsymbol{C}_i \cap \boldsymbol{C}_j$

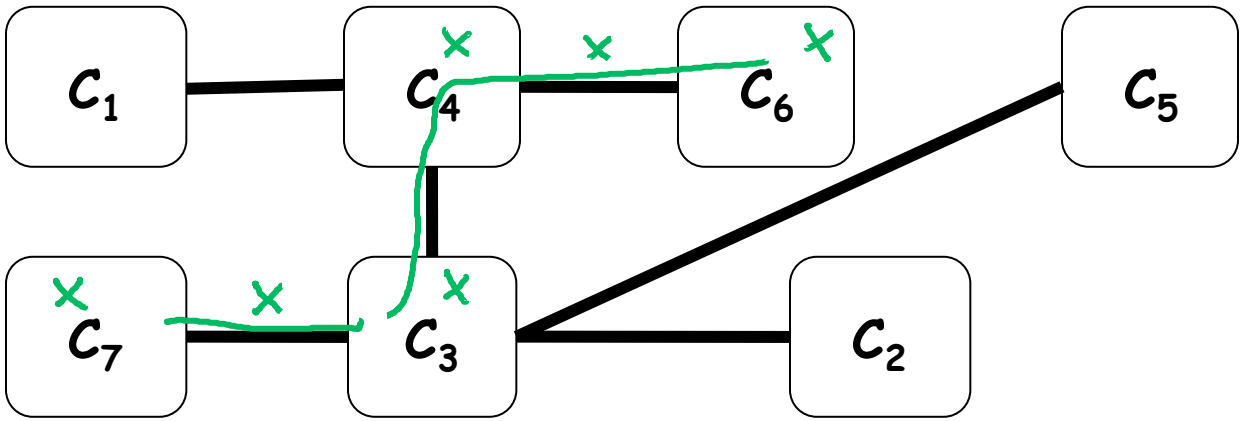
Family Preservation

- Given set of factors Φ, we assign each $\phi_k \in \Phi$ to a cluster $\boldsymbol{C}_{\alpha(k)}$ s.t. $\text{Scope}[\phi_k] \subseteq \boldsymbol{C}_{\alpha(k)}$
- For each factor $\phi_k \in \Phi$, there exists a cluster $\boldsymbol{C}_i$ s.t. $\text{Scope}[\phi_k] \subseteq \boldsymbol{C}_i$

Running Intersection Property

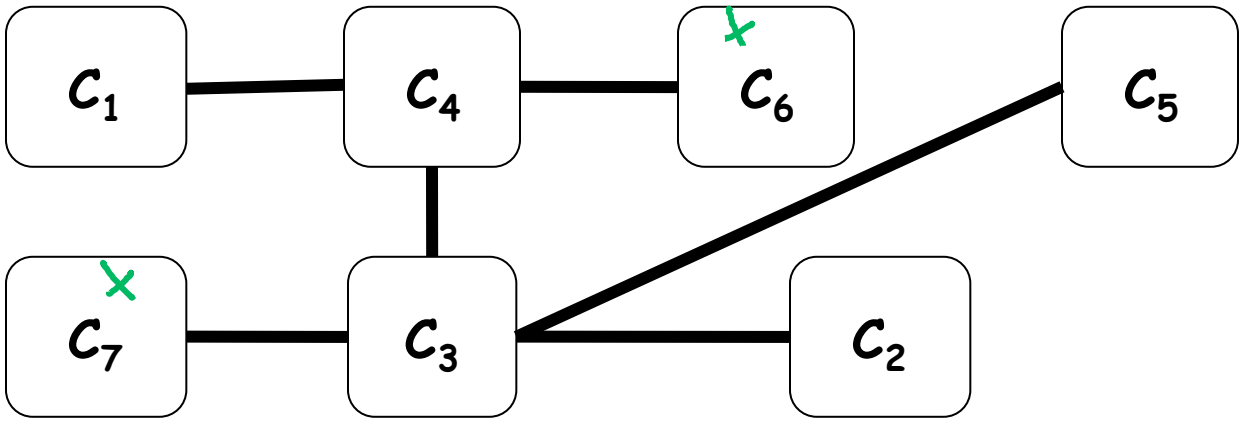
Cluster graph variant

- For each pair of clusters C_i, C_j and 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

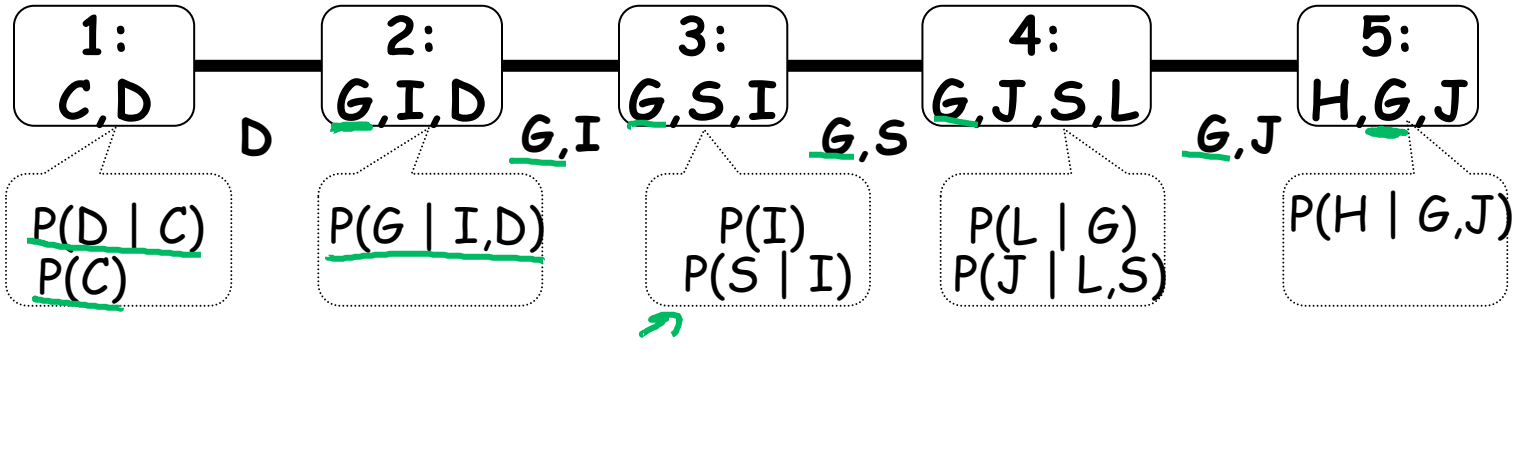


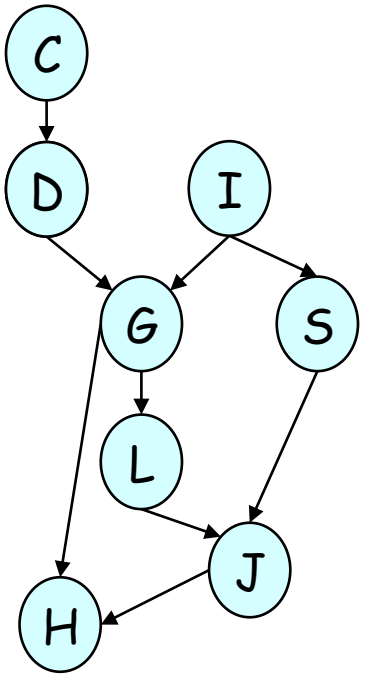
Running Intersection Property

- For each pair of clusters C_i, C_j and variable $X \in C_i \cap C_j$, in the unique path between C_i and C_j, all clusters and sepsets contain X

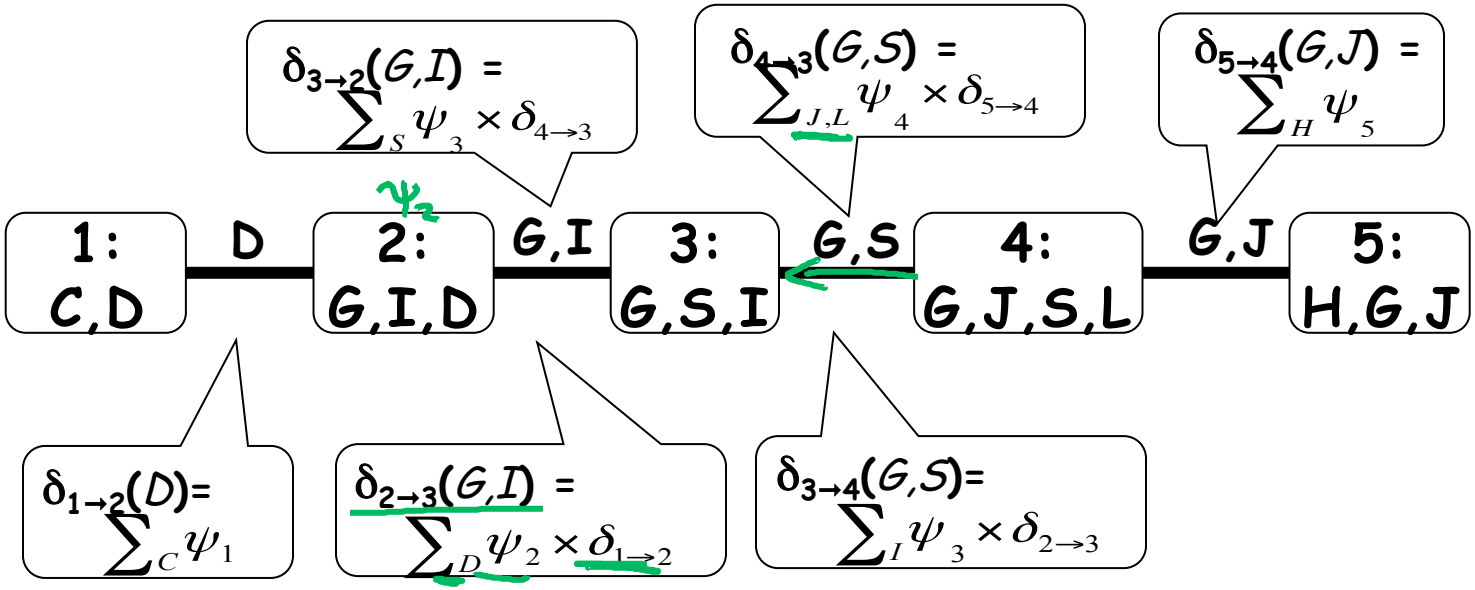


More Complex Clique Tree



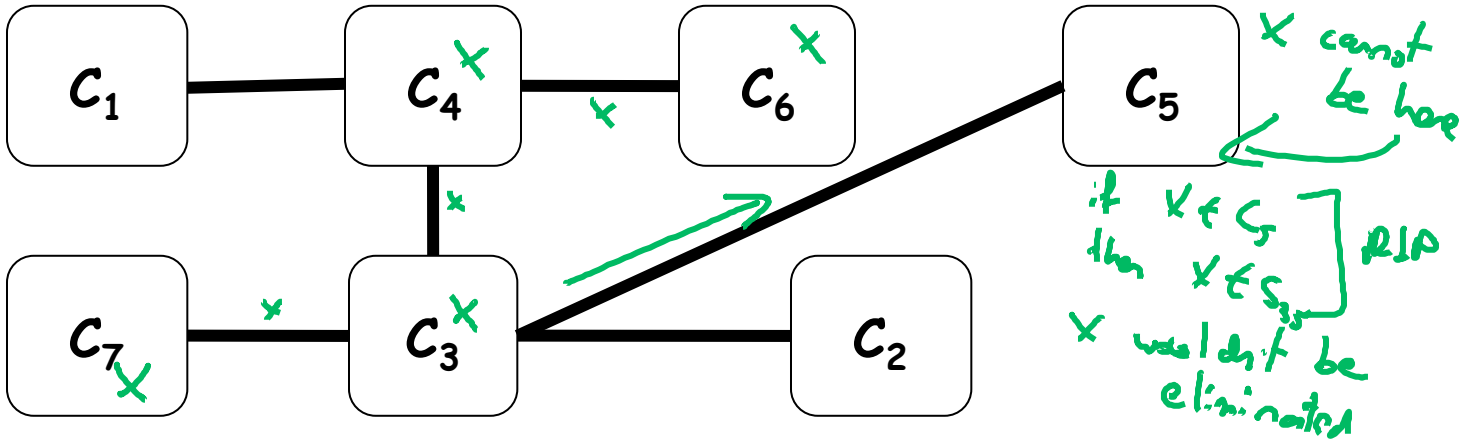


Clique Tree Message Passing

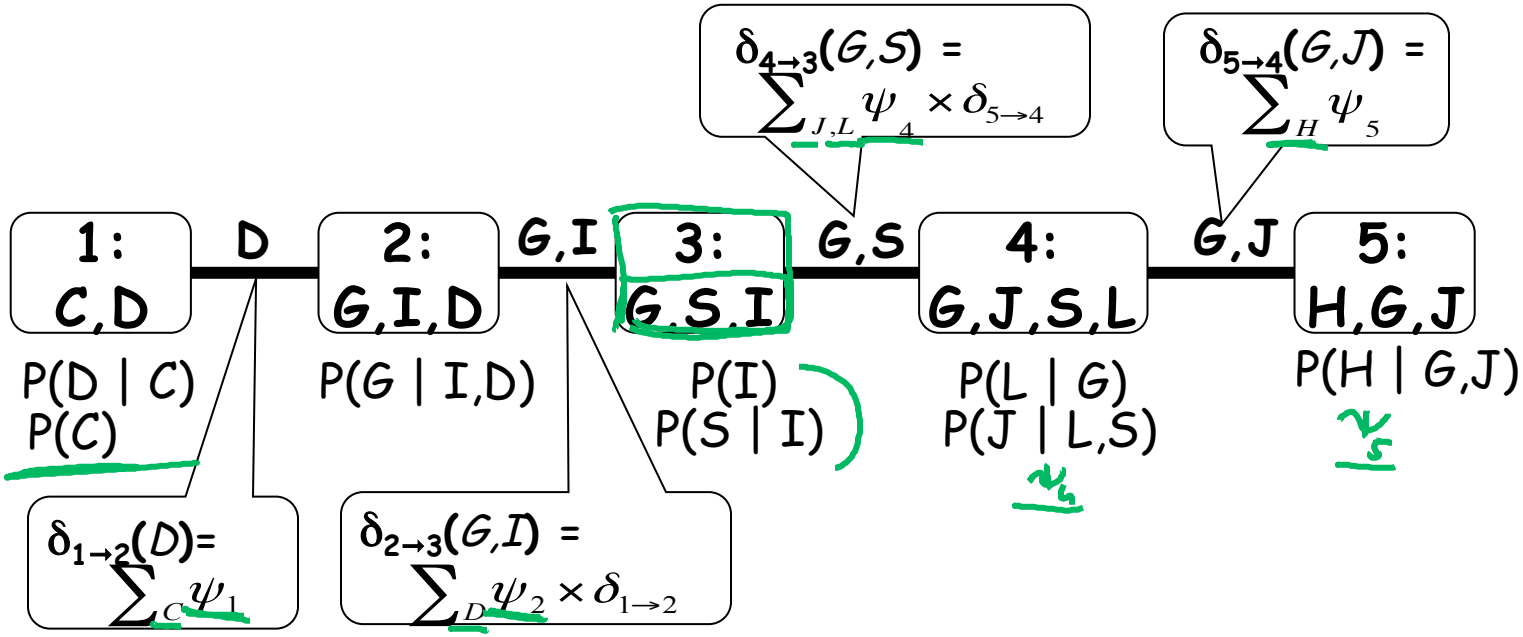


RIP ⇨ Clique Tree Correctness

- If X is eliminated when we pass the message $C_i \rightarrow C_j$
- Then X does not appear in the C_j side of the tree



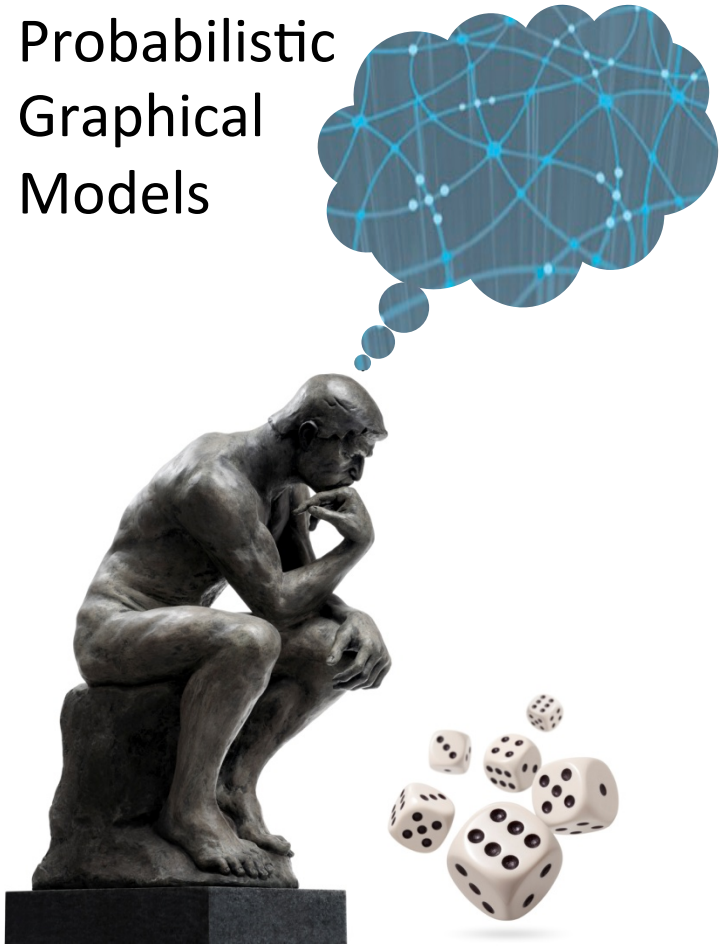
Clique Tree Correctness



Summary

- Belief propagation can be run over a tree-structured cluster graph
- In this case, computation is a variant of variable elimination
- Resulting beliefs are guaranteed to be correct marginals

Probabilistic Graphical Models

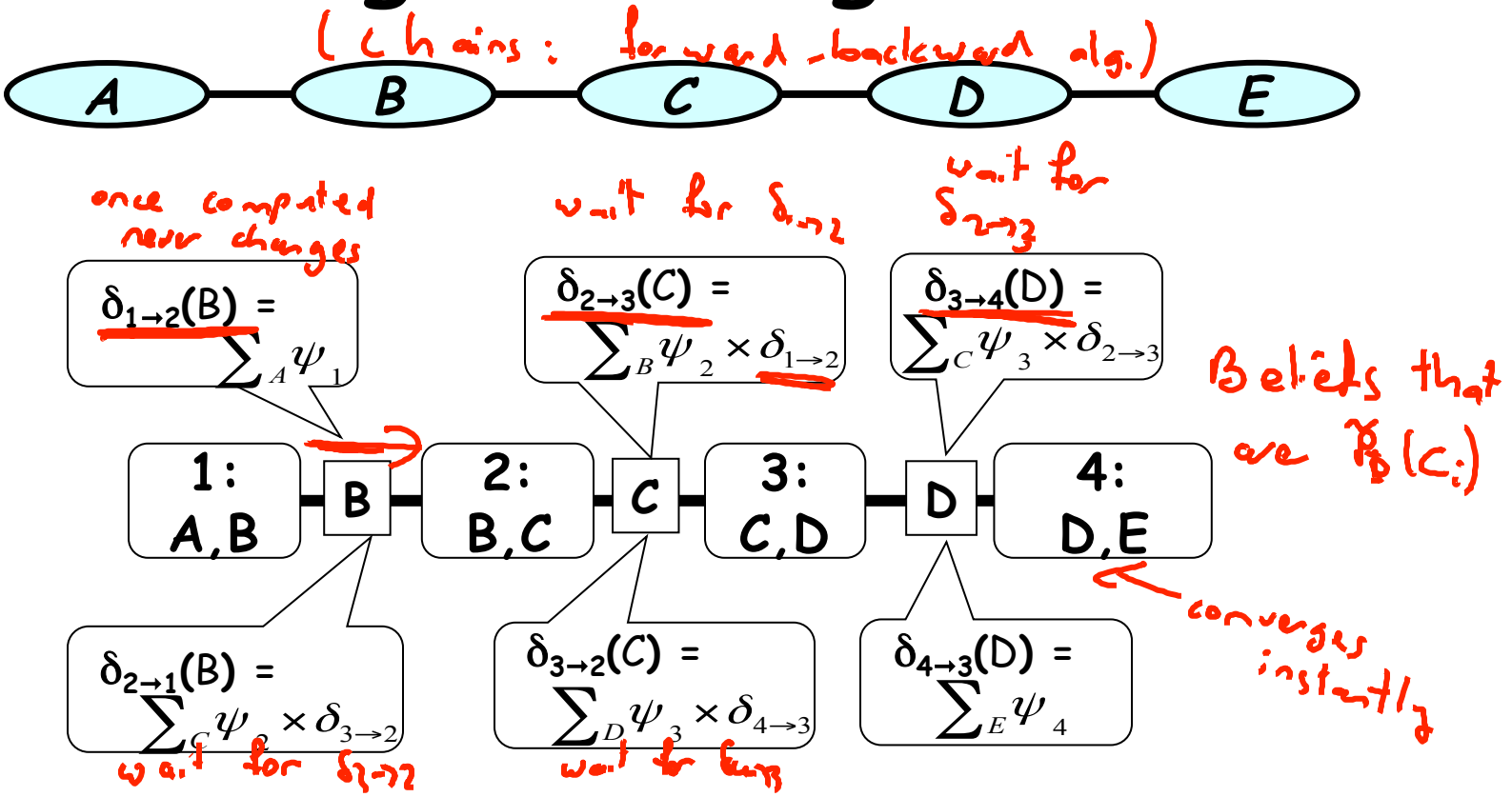


Inference

Message Passing

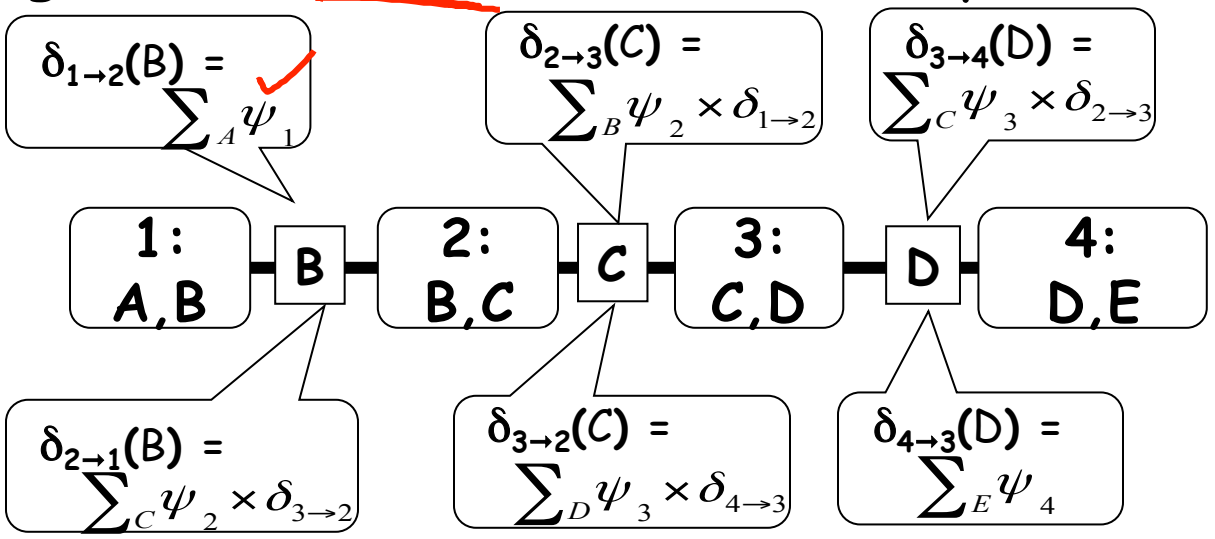
Clique Tree Algorithm: Computation

Message Passing in Trees



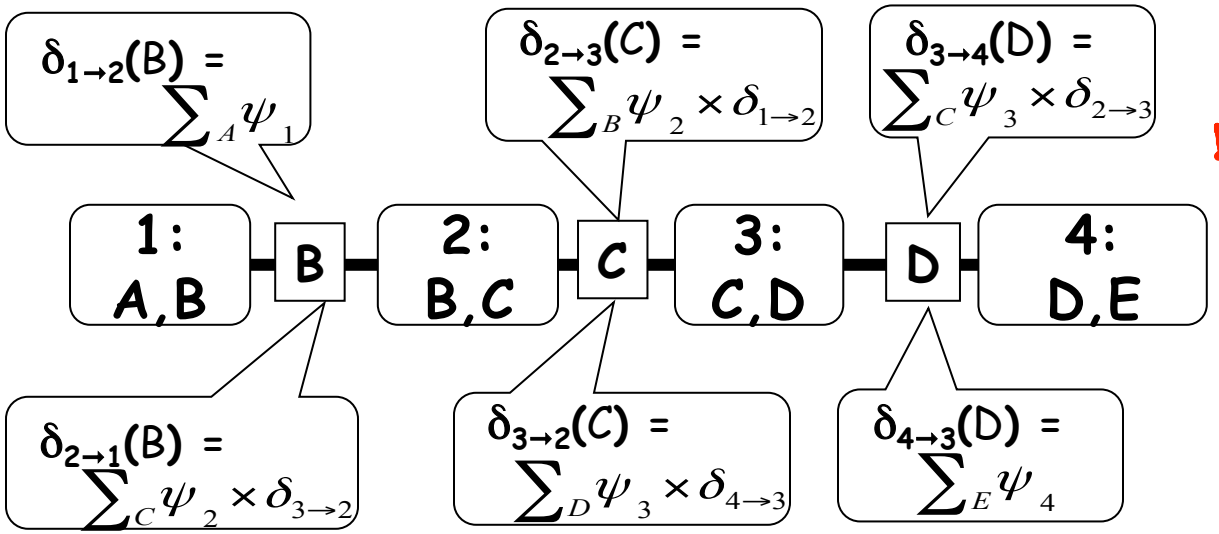
Convergence of Message Passing

- Once C_i receives a final message from all neighbors except C_j, then $\delta_{i \to j}$ is also final (will never change)
- Messages from leaves are immediately final



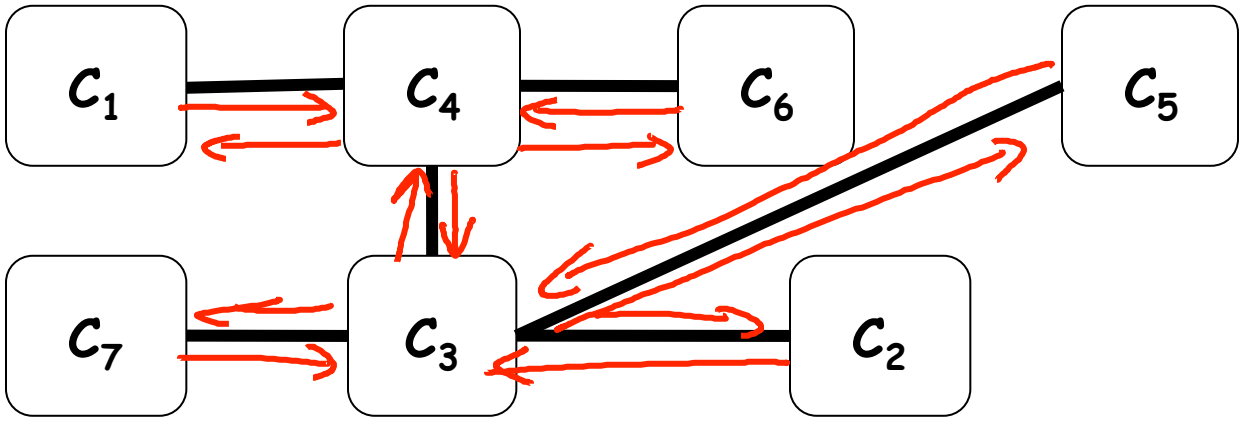
Convergence of Message Passing

- Can pass messages from leaves inward
- If messages are passed in the right order, only need to pass 2(K-1) messages

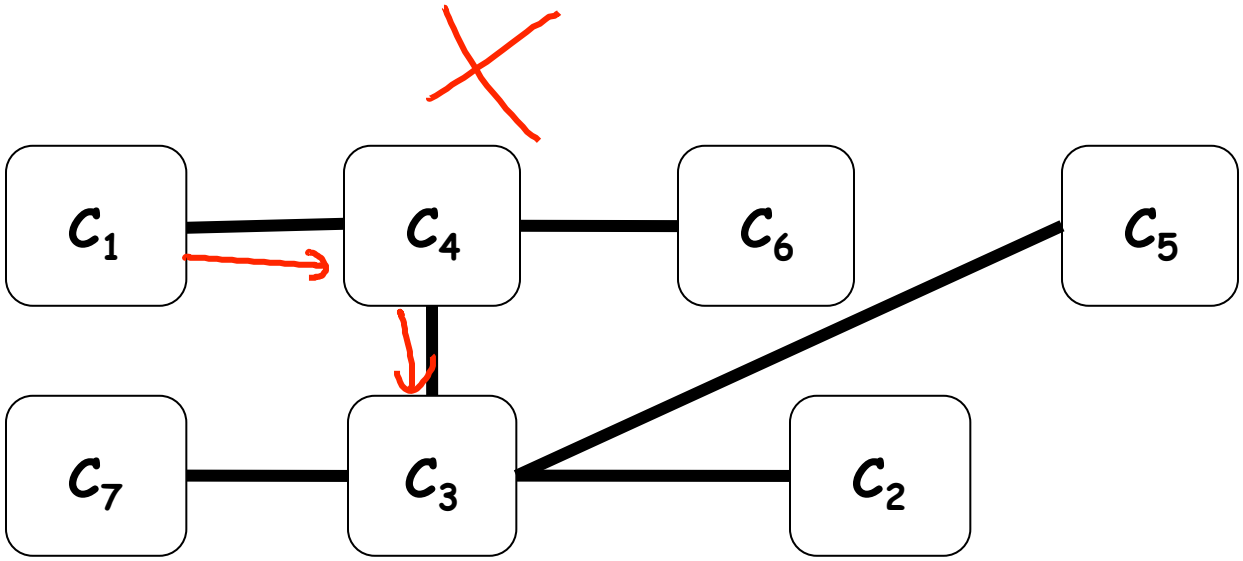


K total number of clusters.
⇓
K-1 edges
⇓
2(K-1) msgs

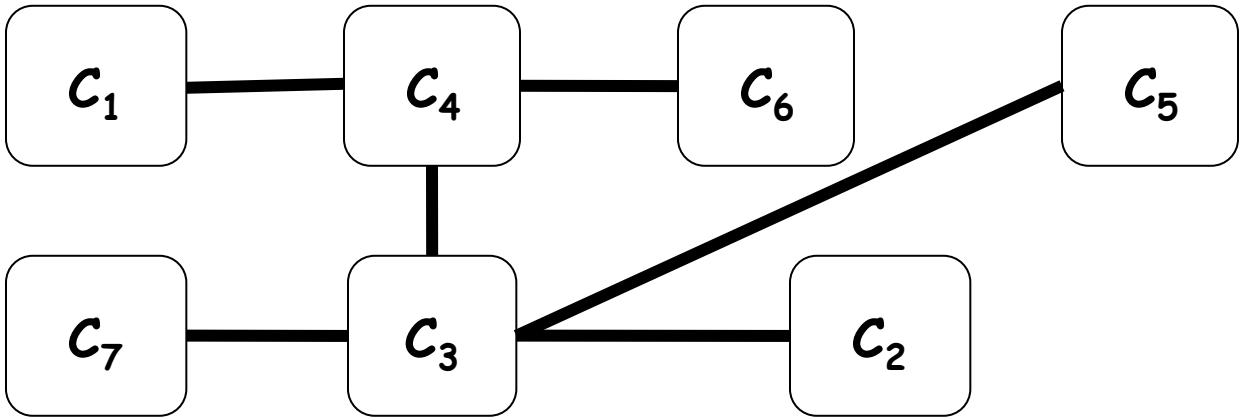
Message Passing Order I



Message Passing Order II



Message Passing Order III

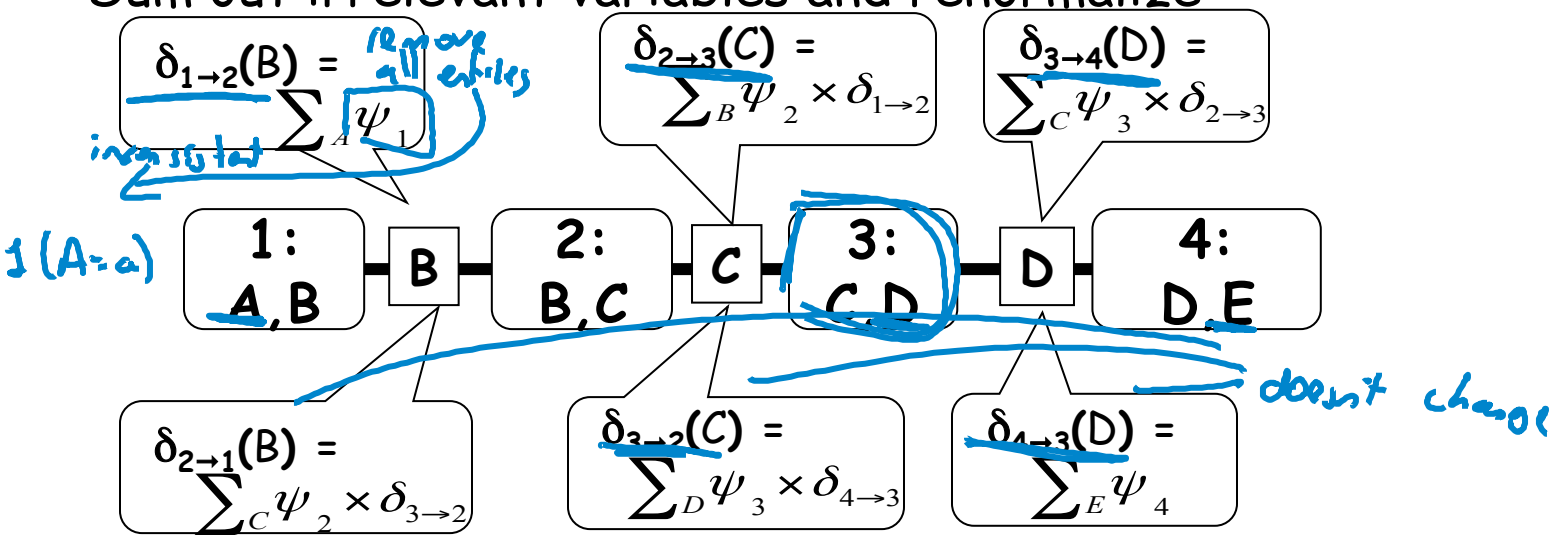


Answering Queries

- Posterior distribution queries on variables that appear together in clique
  - Sum out irrelevant variables from any clique containing those variables
- Introducing new evidence Z=z and querying X
  - If X appears in clique with Z
    - Multiply clique that contains X and Z with indicator function 1(Z=z)
    - Sum out irrelevant variables and renormalize

And More Queries

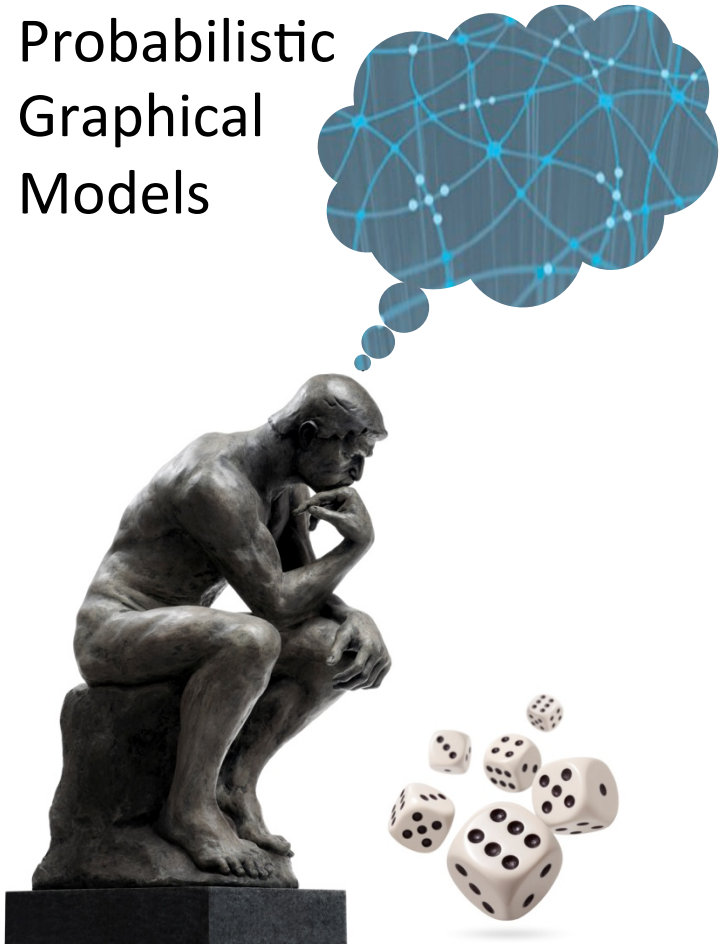
- Introducing new evidence Z=z and querying X if X does not share a clique with Z
  - Multiply **1**(Z=z) into some clique containing Z
  - Propagate messages along path to clique containing X
  - Sum out irrelevant variables and renormalize



Summary

- In clique tree with K cliques, if messages are passed starting at leaves, 2(K-1) messages suffice to compute all beliefs
- Can compute marginals over all variables at only twice the cost of variable elimination
- By storing messages, inference can be reused in incremental queries

Probabilistic Graphical Models

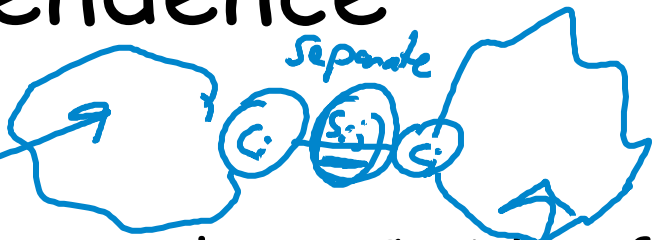


Inference

Message Passing

Clique Tree & Independence

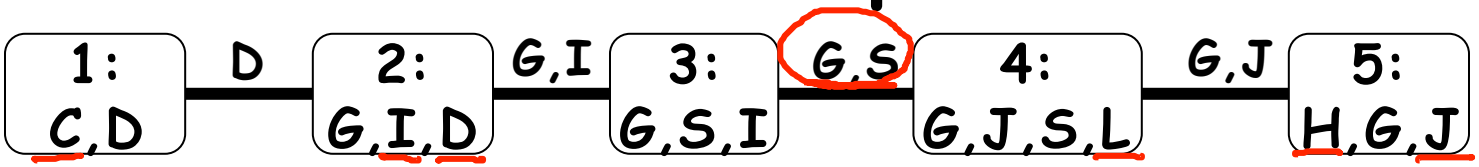
RIP and Independence



- For an edge (i,j) in T, let:
  - $\mathbf{W}_{<(i,j)}$ = all variables that appear only on C_i side of T
  - $\mathbf{W}_{<(j,i)}$ = all variables that appear only on C_j side
  - Variables on both sides are in the sepset $S_{i,j}$

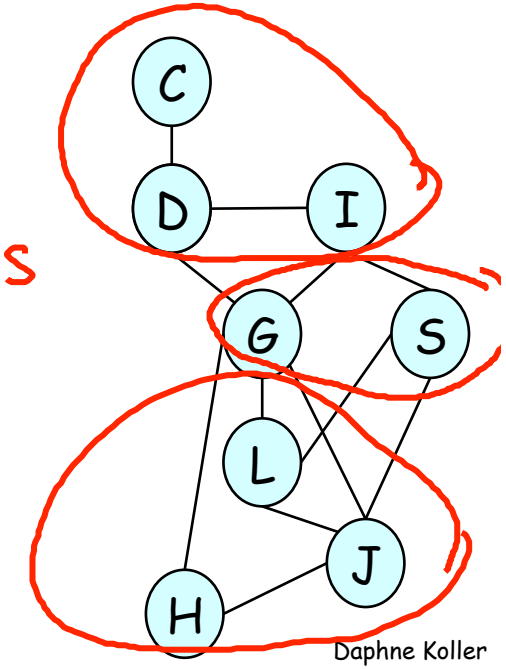
- Theorem: T satisfies RIP if and only if, for every (i,j) $P_\Phi \models (\boldsymbol{W}_{<(i,j)} \perp \boldsymbol{W}_{<(j,i)} \mid \boldsymbol{S}_{i,j})$

RIP and Independence



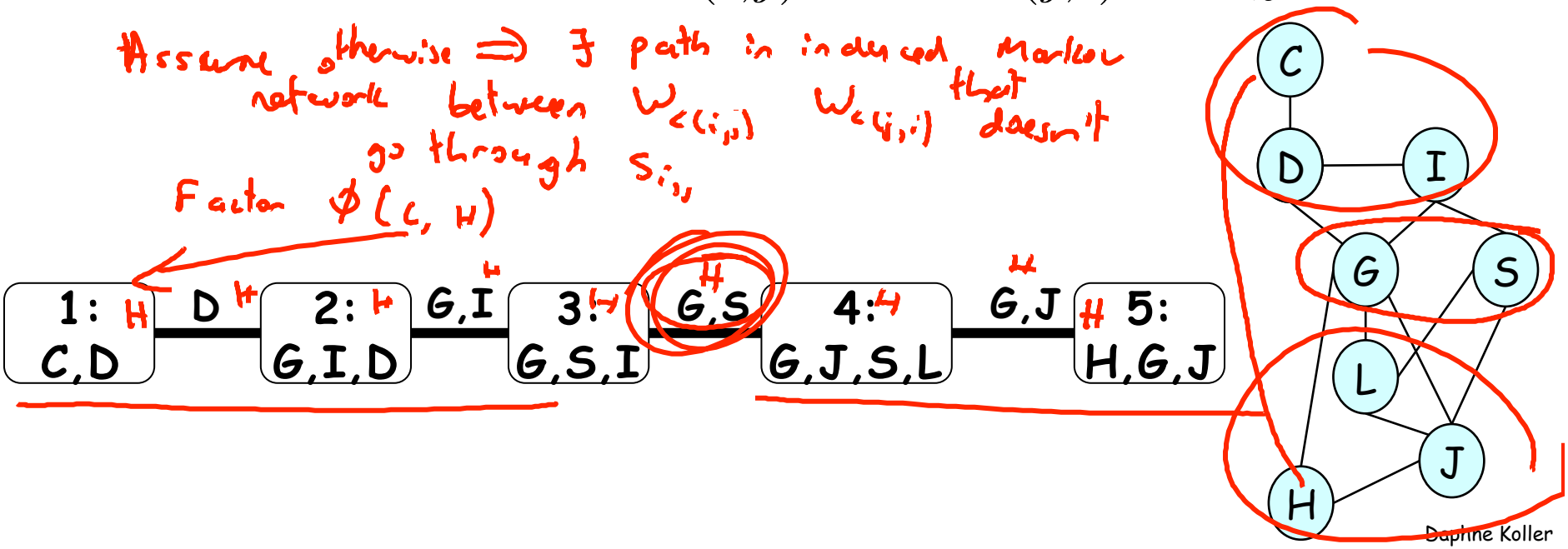
$$P_\Phi \models (\{C, I, D\} \perp \{J, L, H\} \mid \{G, S\})$$

C, I, D separated from H, L, J given G, S



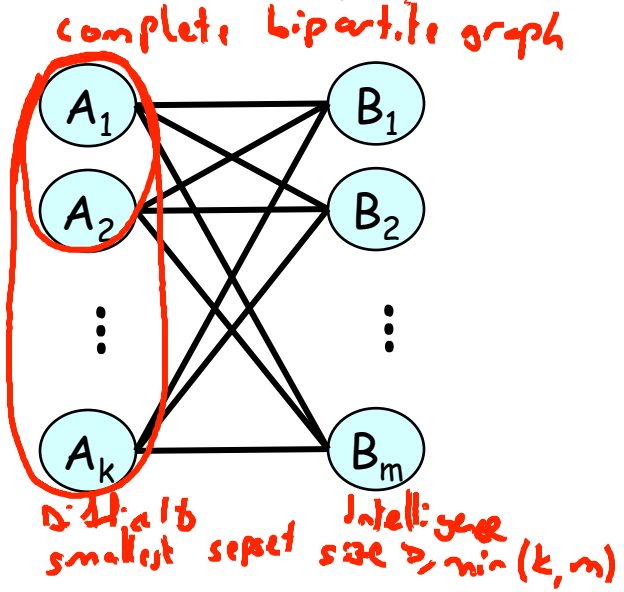
RIP and Independence

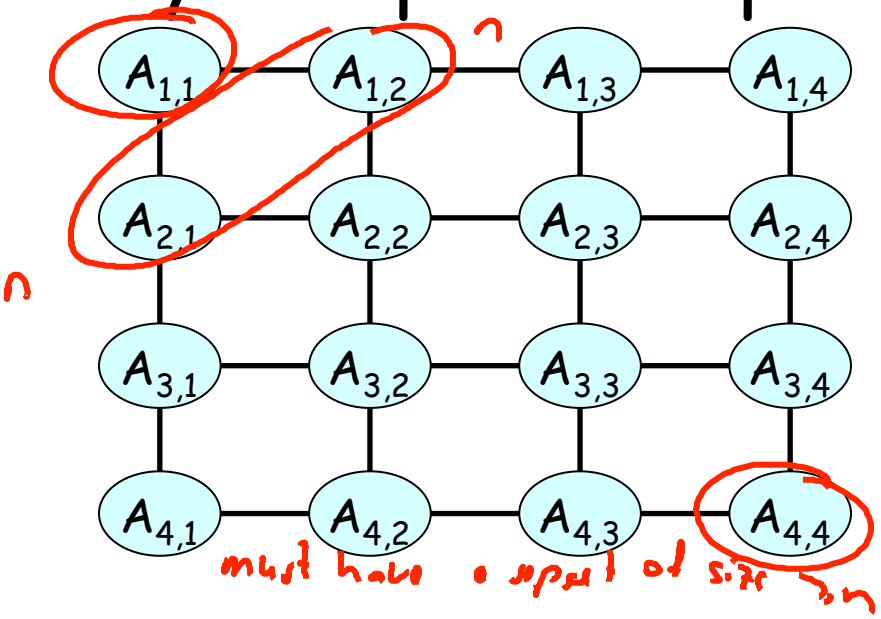
- Theorem: T satisfies RIP if and only if, for every edge (i,j) $P_\Phi \models (\boldsymbol{W}_{<(i,j)} \perp \boldsymbol{W}_{<(j,i)} \mid \boldsymbol{S}_{i,j})$



Implications

- Each sepset needs to separate graph into two conditionally independent parts

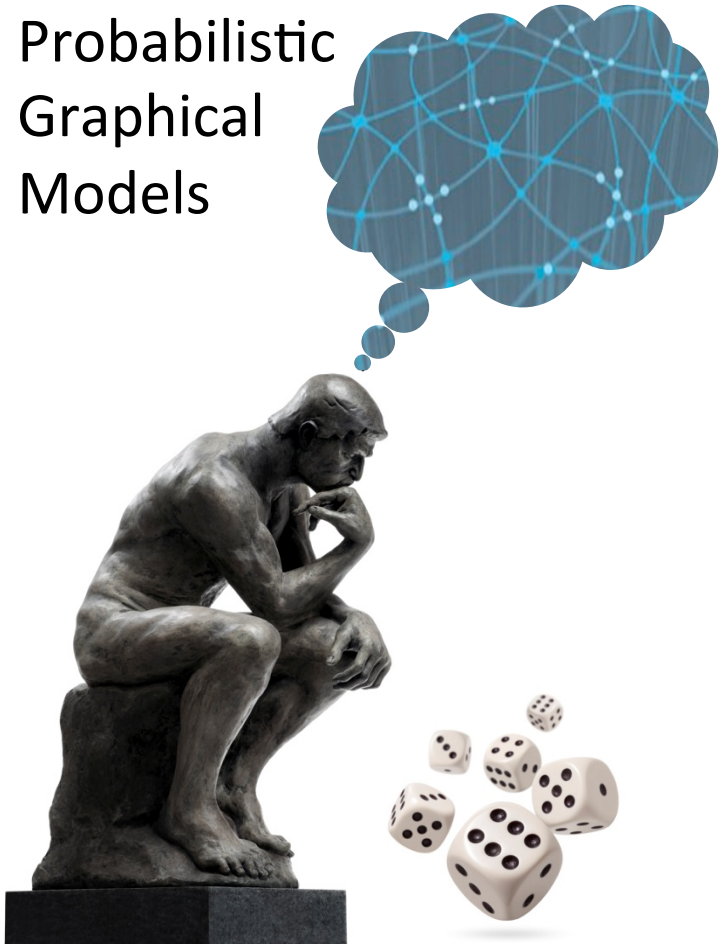




Summary

- Correctness of clique tree inference relies on running intersection property
- Running intersection property implies separation in original distribution
- Implies minimal complexity incurred by any clique tree:
  - Related to minimal induced width of graph

Probabilistic Graphical Models



Inference

Message Passing

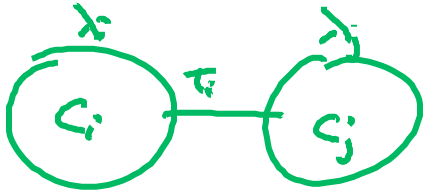
Clique Tree and VE

Variable Elimination & Clique Trees

- Variable elimination
  - Each step creates a factor λ_i through factor product
  - A variable is eliminated in λ_i to generate new factor τ_i
  - τ_i is used in computing other factors λ_j
- Clique tree view
  - Intermediate factors λ_i are cliques
  - τ_i are "messages" generated by clique λ_i and transmitted to another clique λ_j

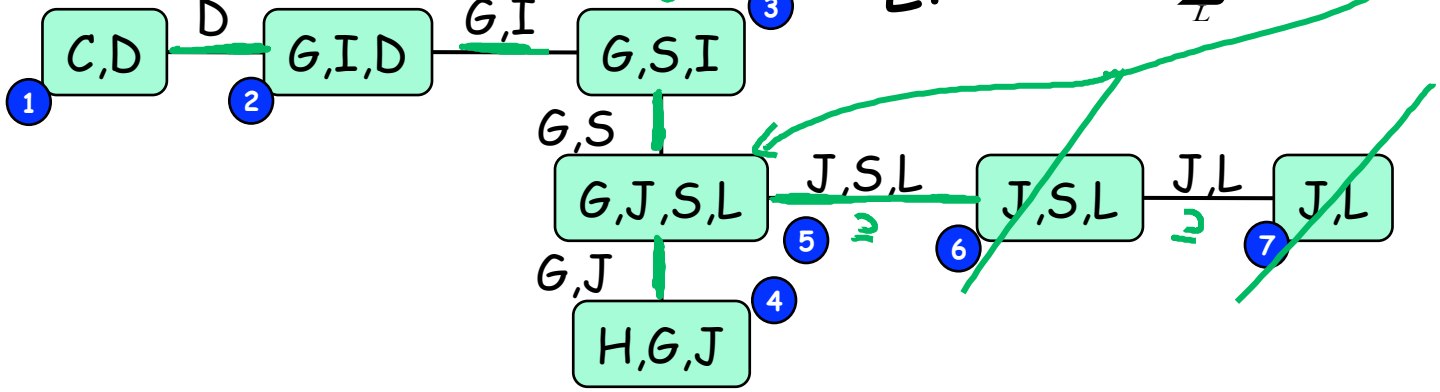
Clique Tree from VE

- VE defines a graph
  - Cluster C_i for each factor λ_i used in the computation
  - Draw edge C_i–C_j if the factor generated from λ_i is used in the computation of λ_j



Example

- C: $\tau_1(D) = \sum_C \phi_C(C)\phi_D(C,D)$
- D: $\tau_2(G,I) = \sum_D \phi_G(G,I,D)\tau_1(D)$
- I: $\tau_3(G,S) = \sum_I \phi_I(I)\phi_S(S,I)\tau_2(G,I)$
- H: $\tau_4(G,J) = \sum_H \phi_H(H,G,J)$
- G: $\tau_5(J,L,S) = \sum_G \phi_L(L,G)\tau_3(G,S)\tau_4(G,J)$
- S: $\tau_6(J,L) = \sum_S \phi(J,L,S)\tau_5(J,L,S)$
- L: $\tau_7(J) = \sum_L \tau_6(J,L)$



Remove redundant cliques:

those whose scope is a subset of adjacent clique's scope

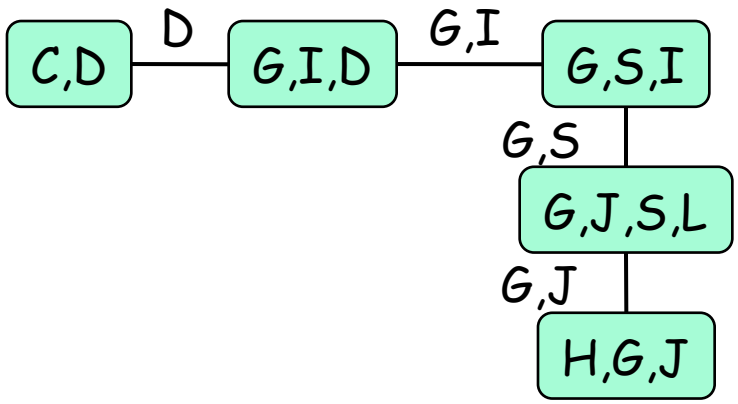
Properties of Tree

- VE process induces a tree
  - In VE, each intermediate factor is used only once
  - Hence, each cluster "passes" a factor (message) to exactly one other cluster

- Tree is family preserving:
  - Each of the original factors must be used in some elimination step
  - And therefore contained in scope of associated Ψ_i

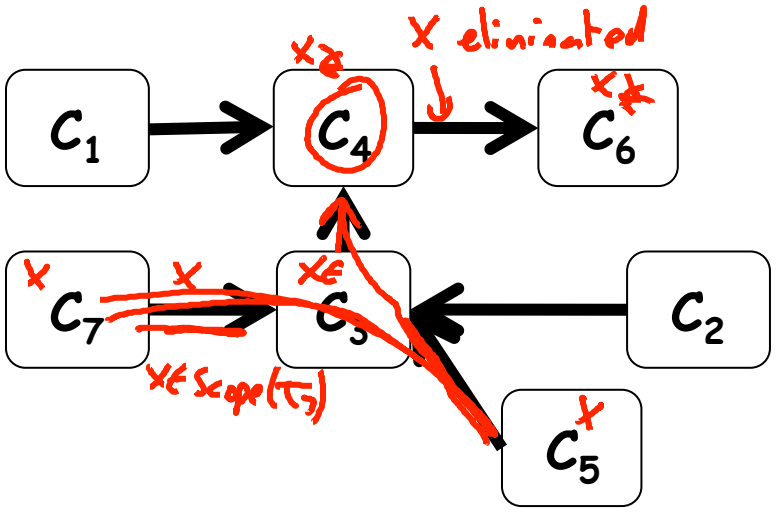
Properties of Tree

- Tree obeys running intersection property
  - If $X \in C_i$ and $X \in C_j$ then X is in each cluster in the (unique) path between C_i and C_j



Running Intersection Property

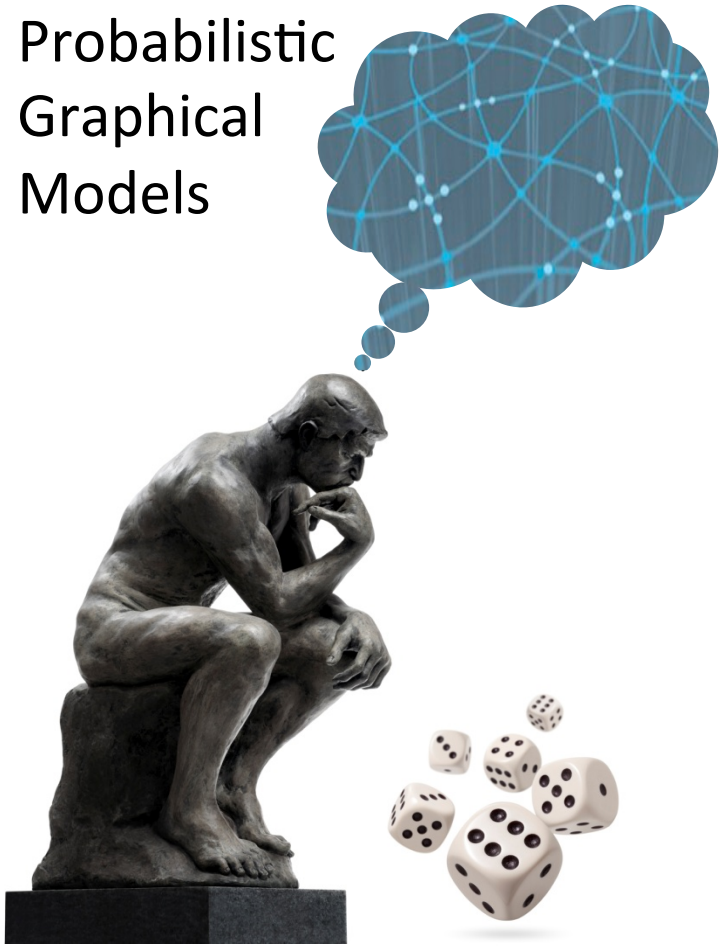
- Theorem: If T is a tree of clusters induced by VE, then T obeys RIP



Summary

- A run of variable elimination implicitly defines a correct clique tree
  - We can "simulate" a run of VE to define cliques and connections between them
- Cost of variable elimination is ~ the same as passing messages in one direction in tree
- Clique trees use dynamic programming (storing messages) to compute marginals over all variables at only twice the cost of VE

Probabilistic Graphical Models

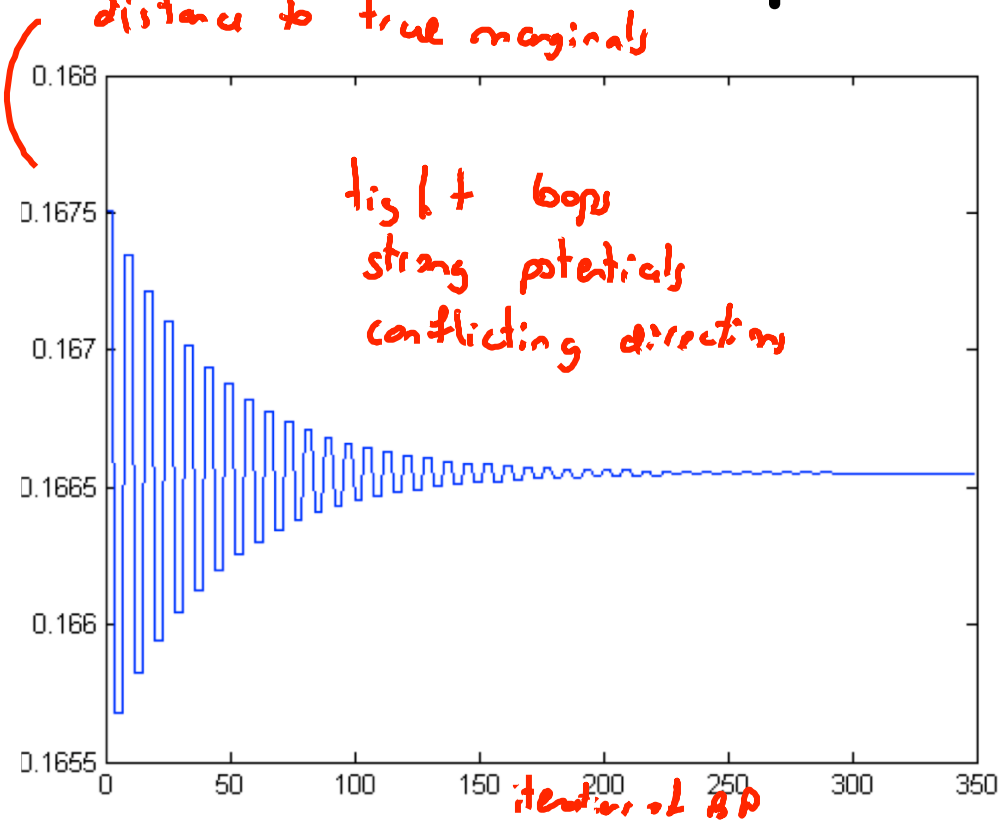


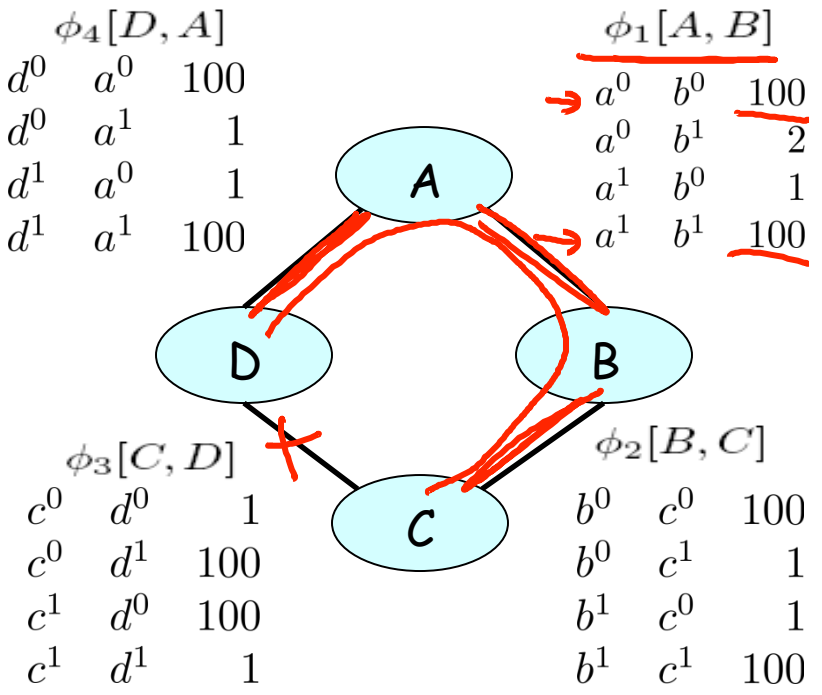
Inference

Message Passing

BP In Practice

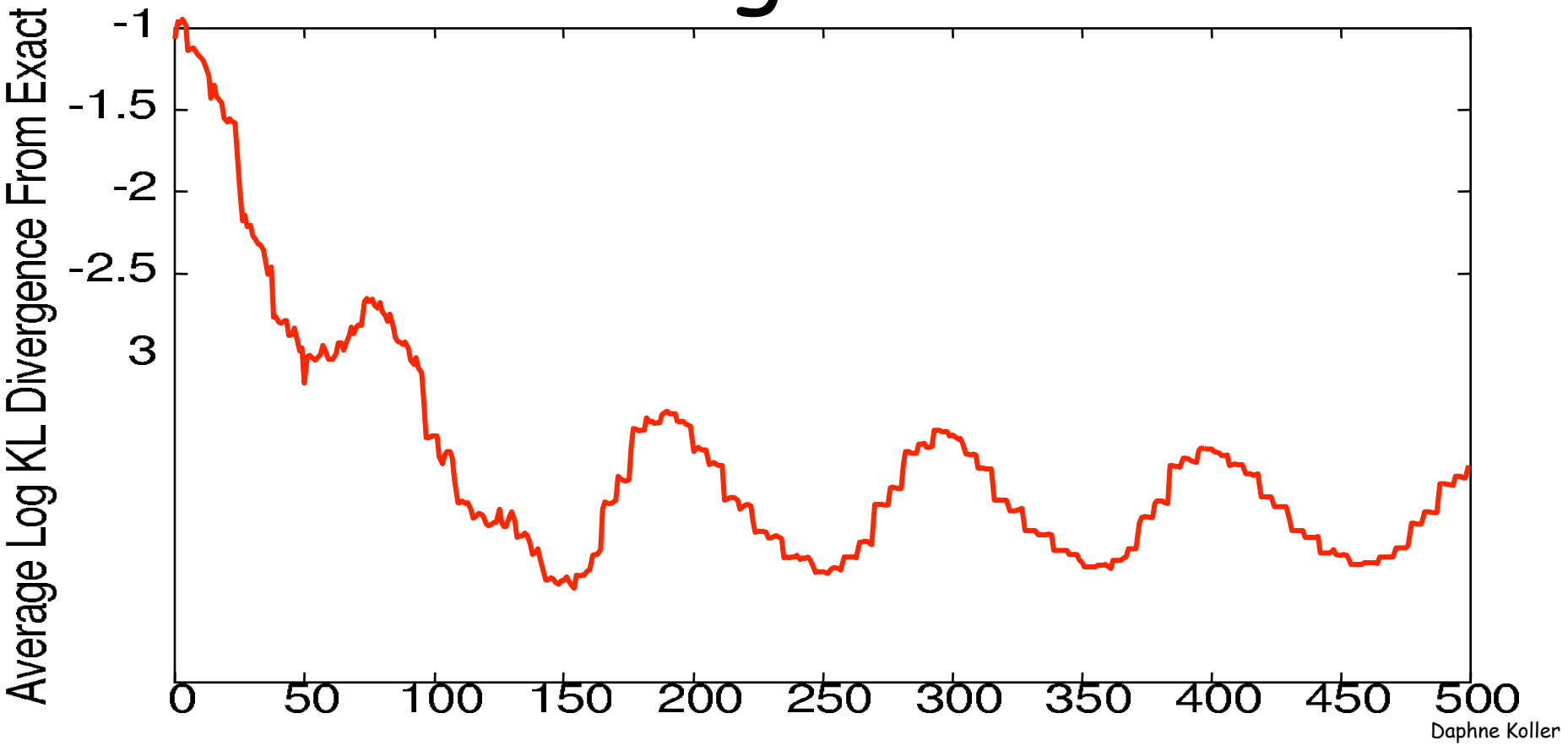
Misconception Revisited





Daphne Koller

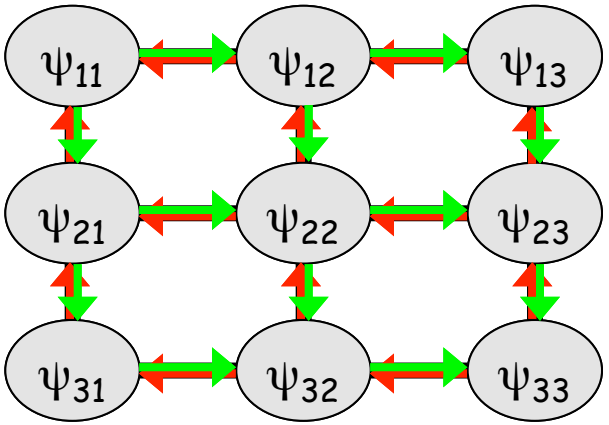
Nonconvergent BP Run

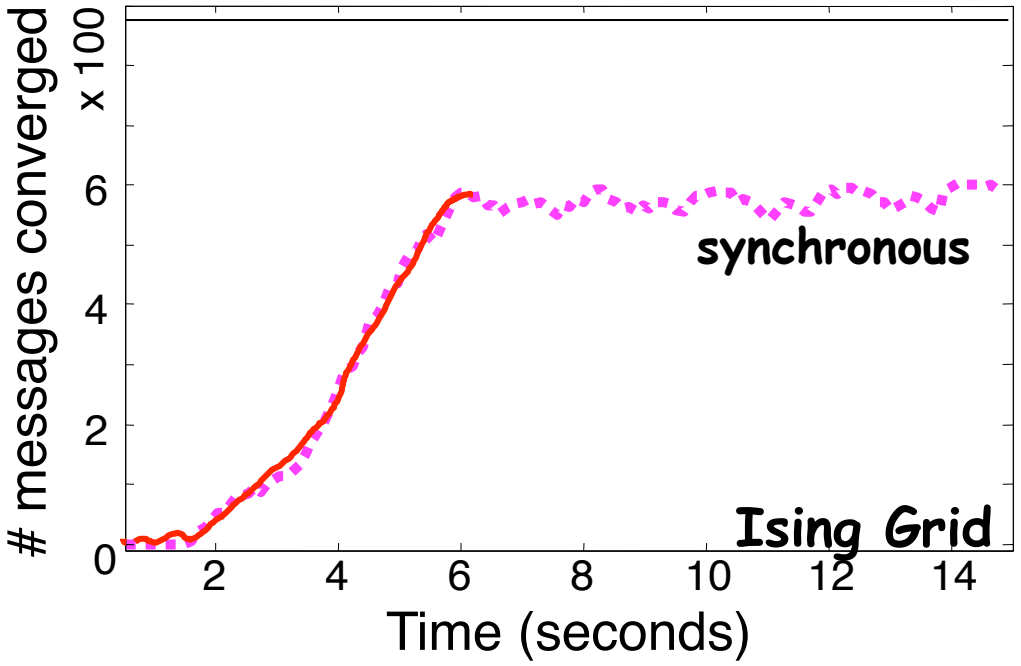


Different Variants of BP

Synchronous BP:

all messages are updated in parallel



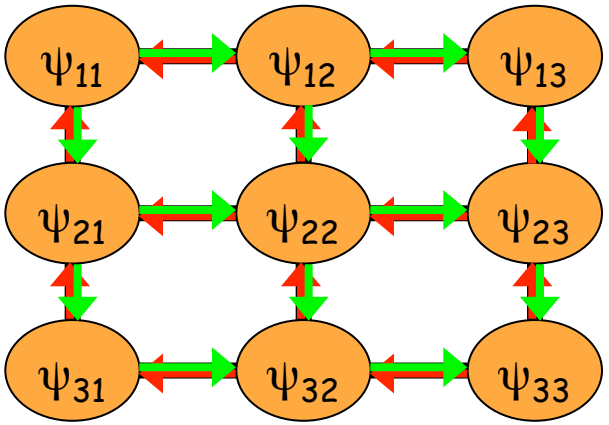


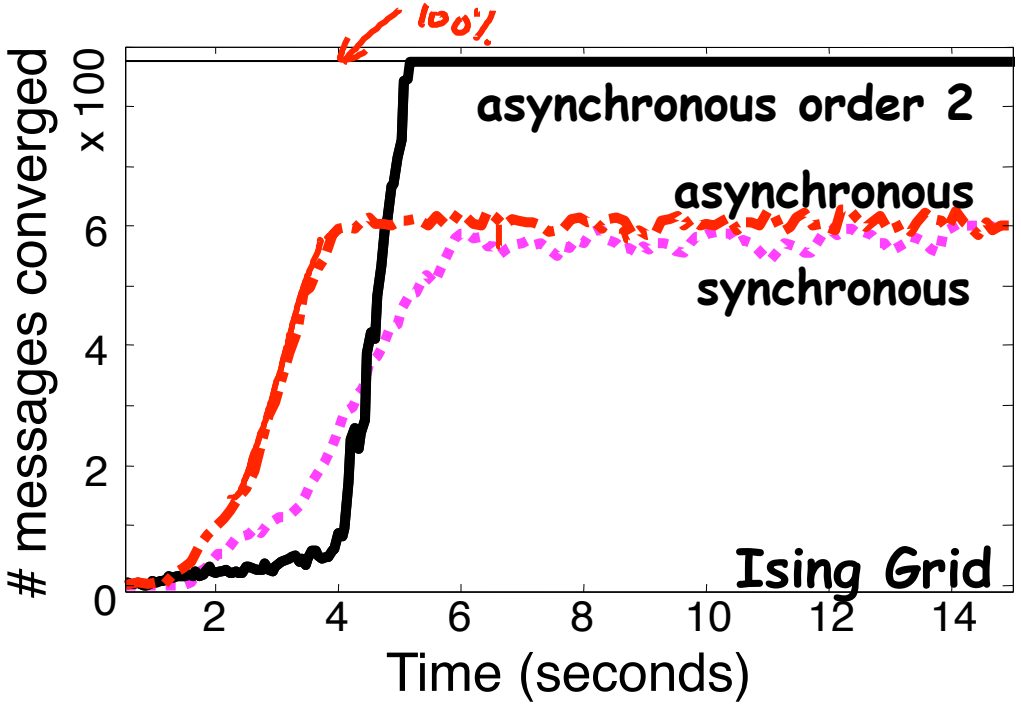
Daphne Koller

Different Variants of BP

Asynchronous BP:

Messages are updated one at a time





Daphne Koller

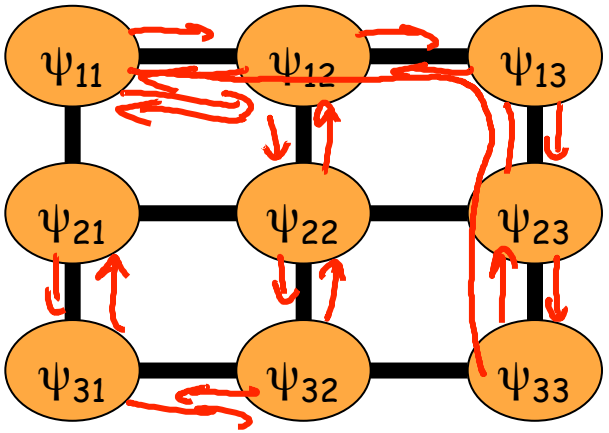
Observations

- Convergence is a local property:
  - some messages converge soon
  - others may never converge
- Synchronous BP converges considerably worse than asynchronous
- Message passing order makes a difference to extent and rate of convergence

Informed Message Scheduling

- Tree reparameterization (TRP)
  - Pick a tree and pass messages to calibrate

trees must cover all edges
improves performance if trees are larger (spanning trees)



Informed Message Scheduling

- Tree reparameterization (TRP)
  - Pick a tree and pass messages to calibrate

- Residual belief propagation (RBP)
  - Pass messages between two clusters whose beliefs over the sepset disagree the most

priority queue of edges

Smoothing (Damping) Messages

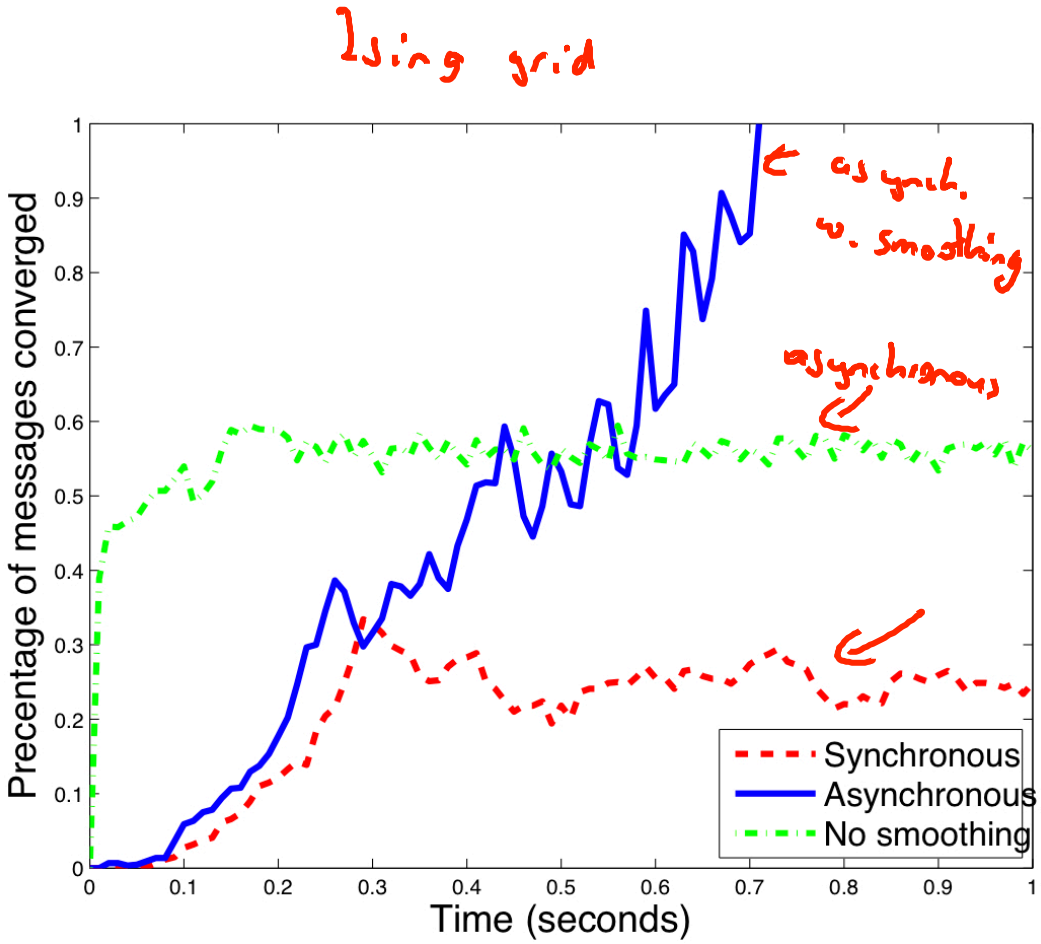
$$\delta_{i\to j} \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \prod_{k\neq j} \delta_{k\to i}$$

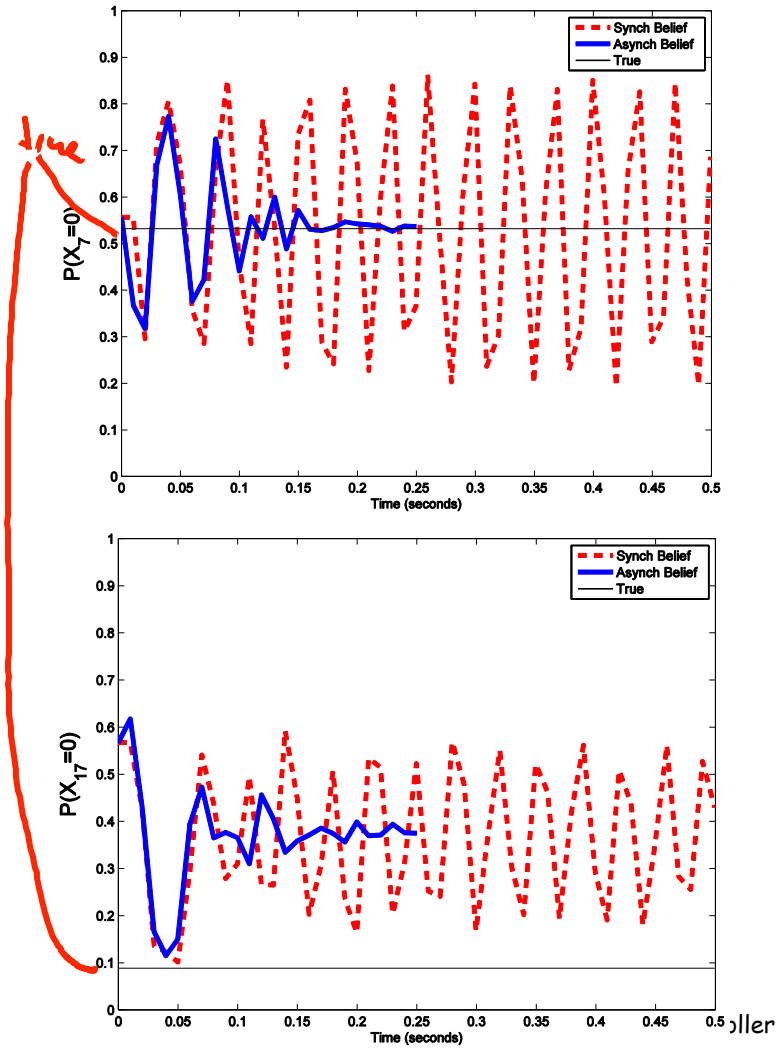
$$\delta_{i\to j} \leftarrow \lambda \left(\sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \prod_{k\neq j} \delta_{k\to i} \right) + (1-\lambda)\delta^{\text{old}}_{i\to j}$$

new msg

old msg

- Dampens oscillations in messages


Ising grid
async. w. smoothing
asynchronous
Precentage of messages converged
Time (seconds)
Synchronous
Asynchronous
No smoothing

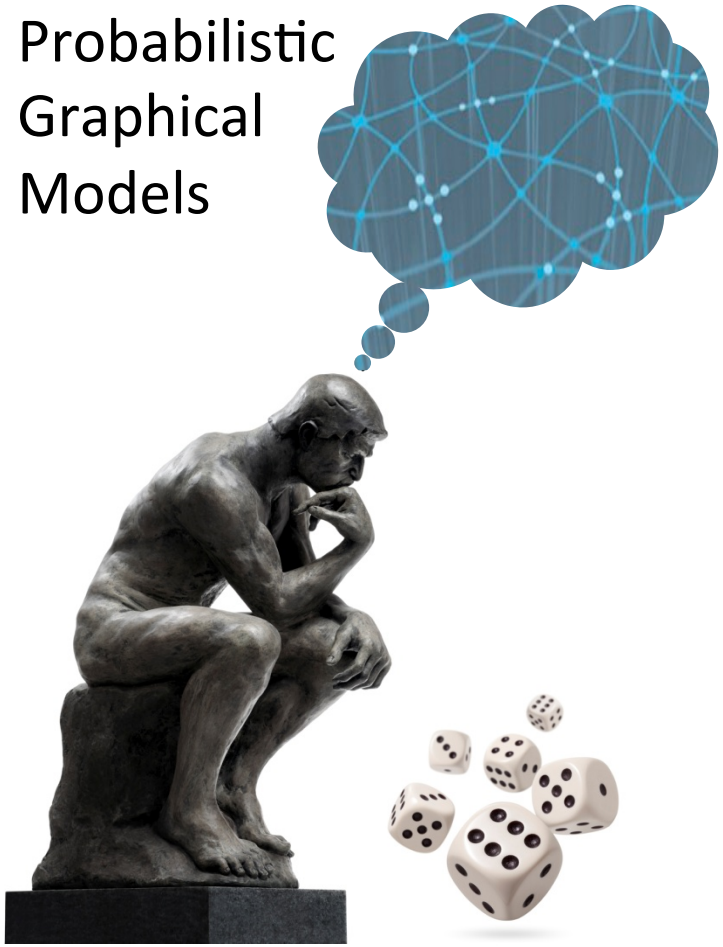

true
Synch Belief
Asynch Belief
True
P(X7=0)
P(X17=0)
Time (seconds)

...ller

Summary

- To achieve BP convergence, two main tricks
  - Damping
  - Intelligent message ordering
- Convergence doesn't guarantee correctness
- Bad cases for BP – both convergence & accuracy:
  - Strong potentials pulling in different directions
  - Tight loops
- Some new algorithms have better convergence:
  - Optimization-based view to inference

Probabilistic Graphical Models

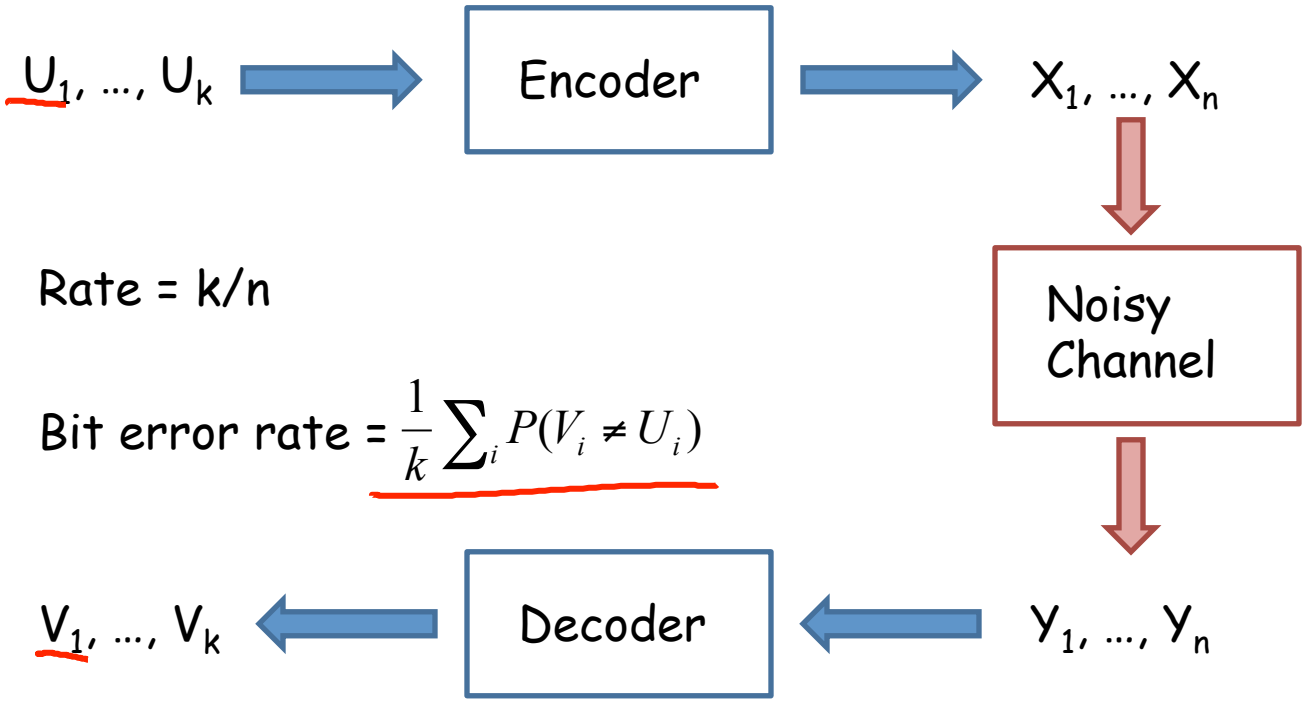


Inference

Message Passing

Loopy BP and Message Decoding

Message Coding & Decoding



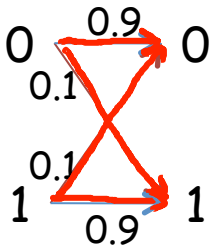
Rate = k/n

Bit error rate = $\frac{1}{k}\sum_i P(V_i \neq U_i)$

Noisy Channel

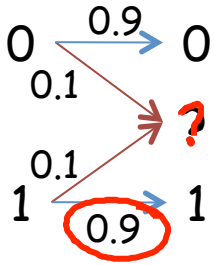
Channel Capacity

Binary symmetric channel

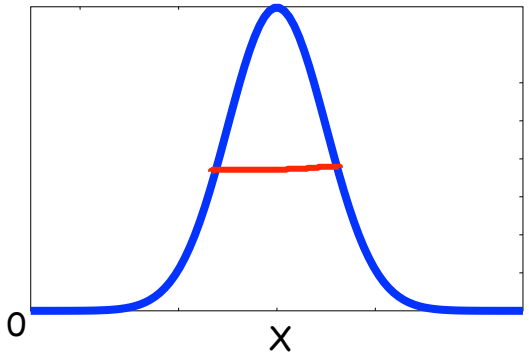


capacity = 0.531

Binary erasure channel



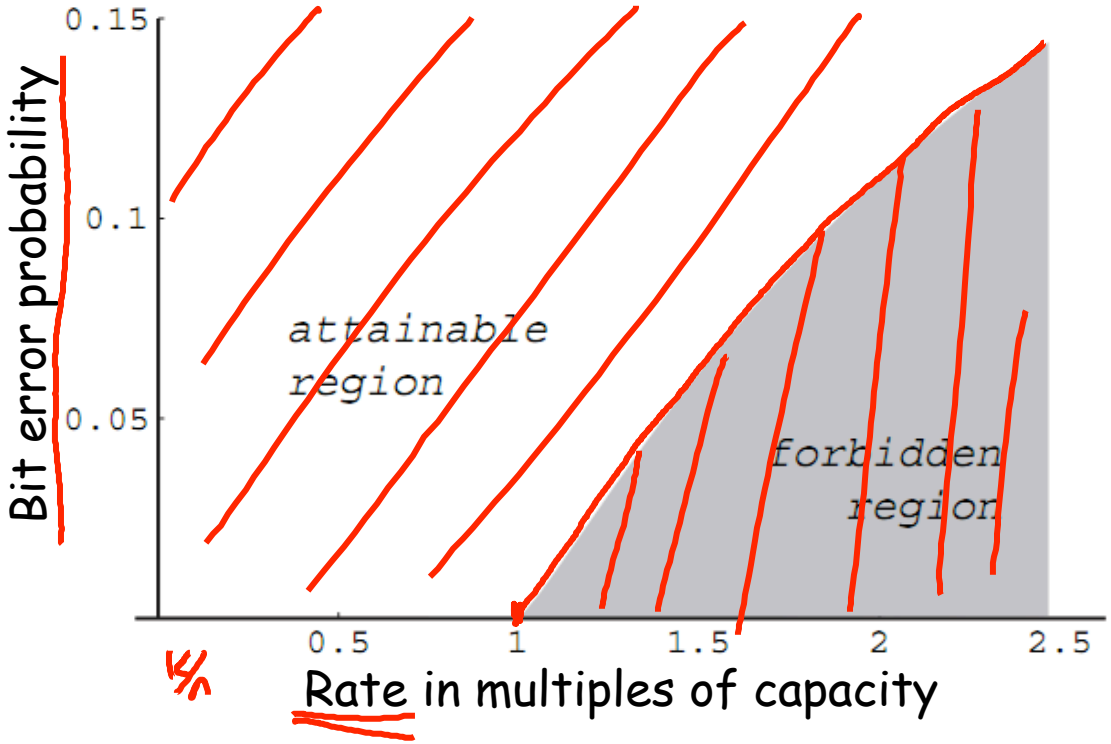
capacity = 0.9



capacity = $\frac{1}{2}\log\left(1+\frac{E(X^2)}{\sigma^2}\right)$

McEliece

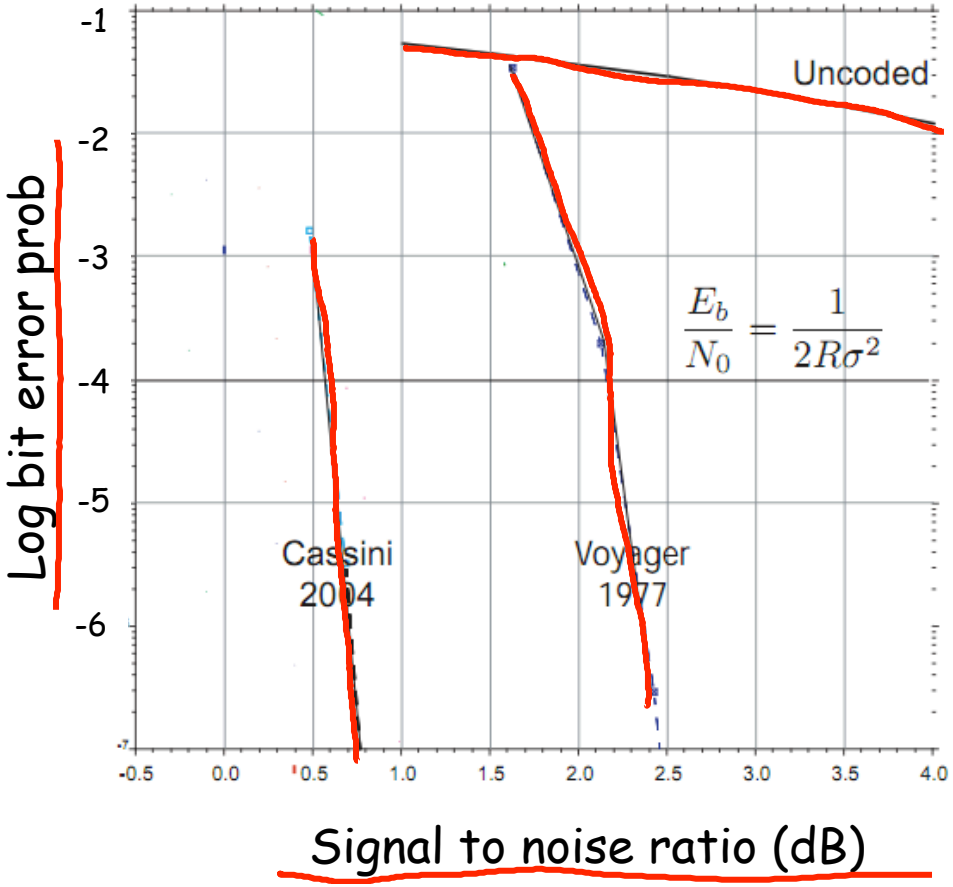
Shannon's Theorem



Daphne Koller

McEliece

How close to C can we get?



Daphne Koller

Turbocodes (May 1993)

NEAR SHANNON LIMIT ERROR - CORRECTING CODING AND DECODING : TURBO-CODES (1)

Claude Berrou, Alain Glavieux and Punya Thitimajshima

Claude Berrou, Integrated Circuits for Telecommunication Laboratory

Alain Glavieux and Punya Thitimajshima, Digital Communication Laboratory

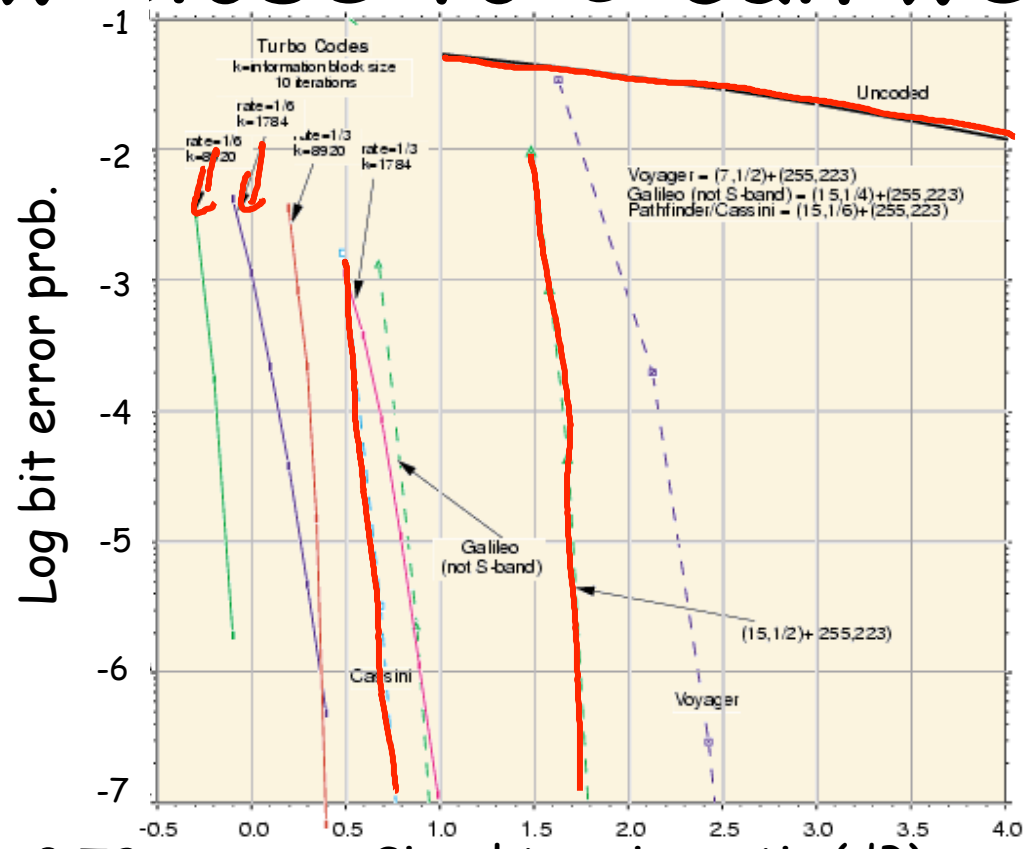
Ecole Nationale Supérieure des Télécommunications de Bretagne, France

(1) Patents N° 9105279 (France), N° 92460011.7 (Europe), N° 07/870,483 (USA)

Daphne Koller

McEliece

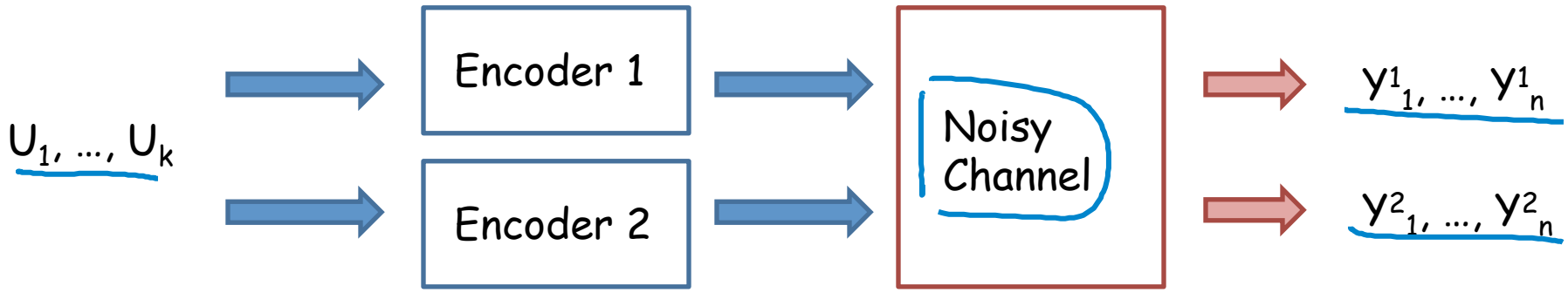
How close to C can we get?



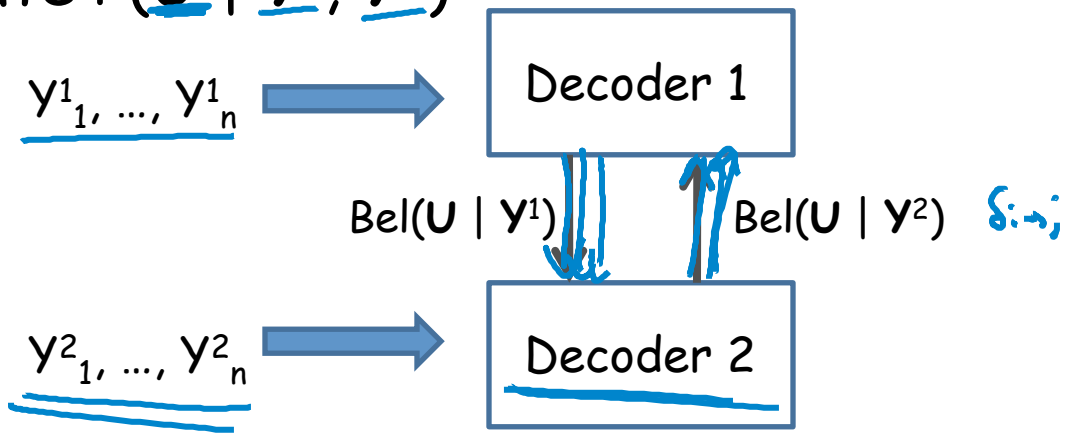
Shannon limit = -0.79

Daphne Koller

Turbocodes: The Idea



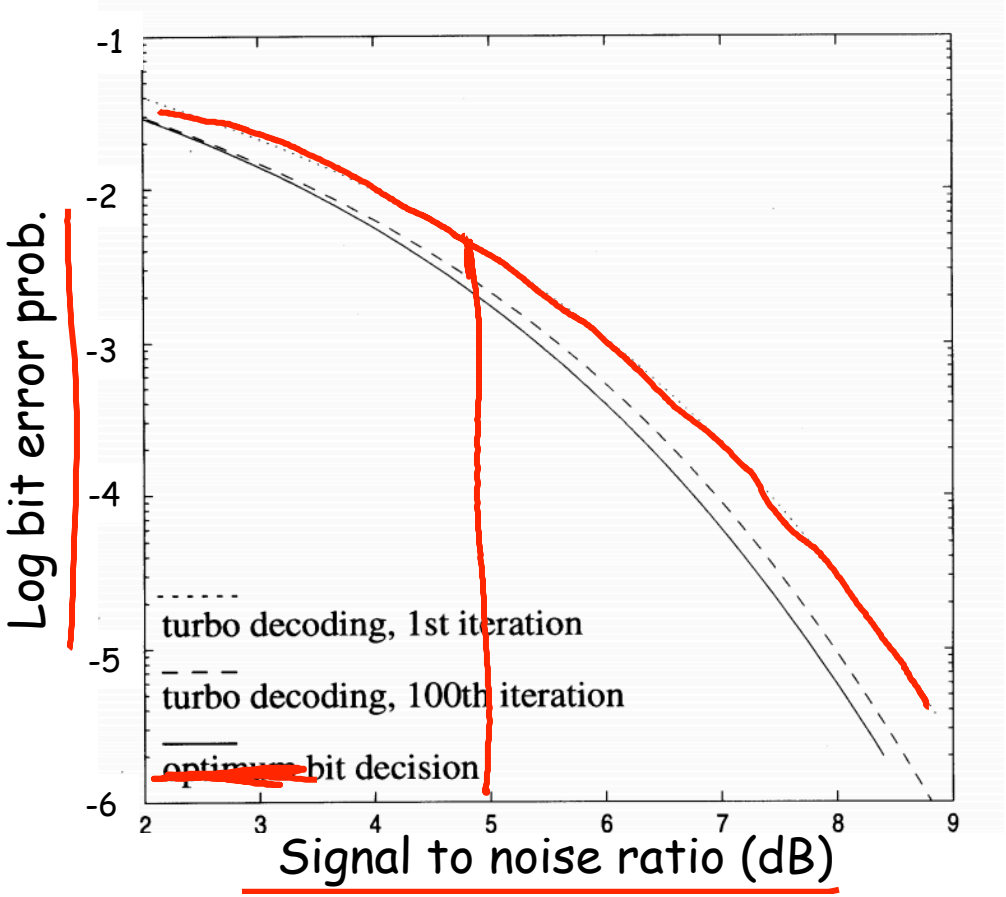
Compute $P(U \mid y^1, y^2)$



Daphne Koller

McEliece

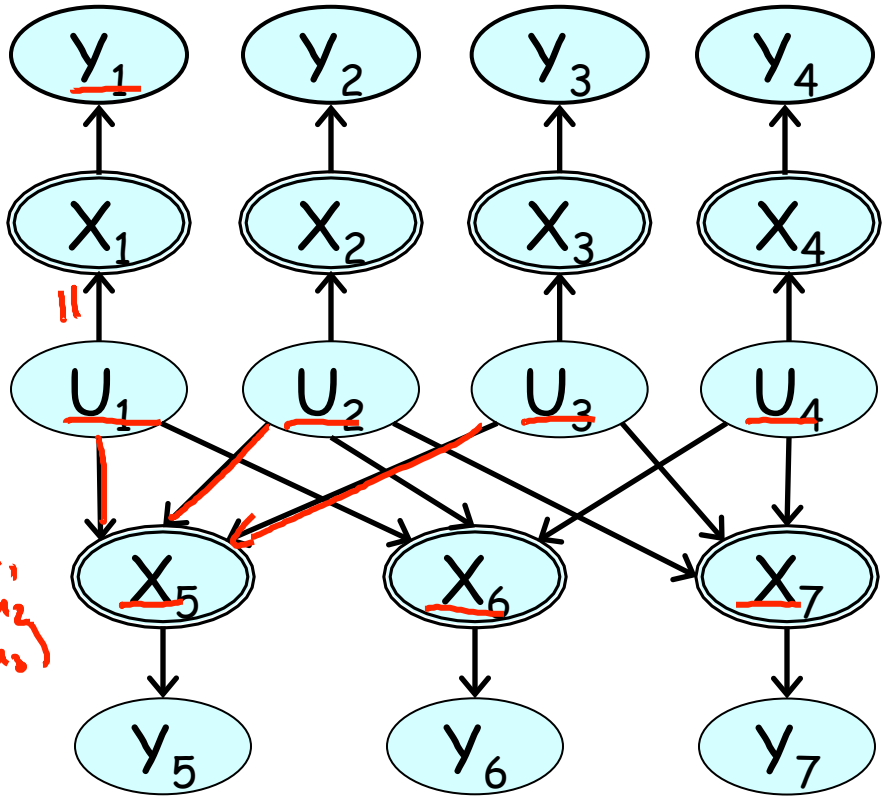
Iterations of Turbo Decoding



Daphne Koller

Low-Density Parity Checking Codes

rate 4/7



noisy bits received

original message bits

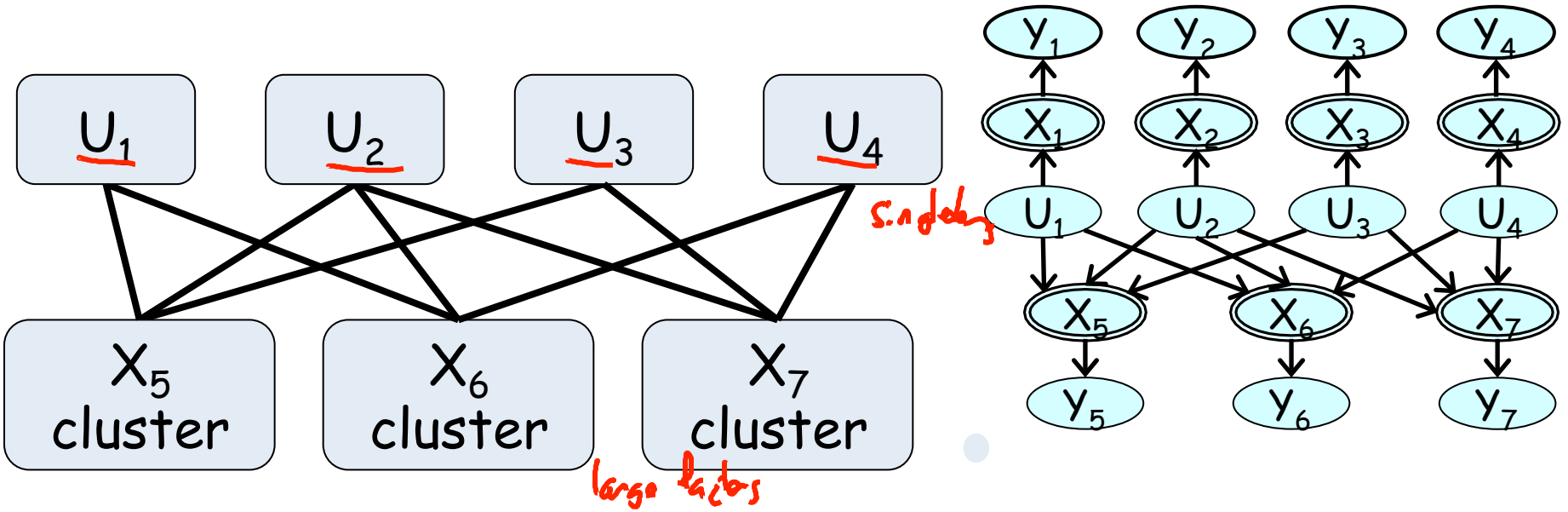
Invented by Robert Gallager in 1962!

parity(u_1, u_2, u_3)

parity bits

Daphne Koller

Decoding as Loopy BP



Turbo-Codes & LDPCs

- 3G and 4G mobile telephony standards
- Mobile television system from Qualcomm
- Digital video broadcasting
- Satellite communication systems
- New NASA missions (e.g., Mars Orbiter)
- Wireless metropolitan network standard

Summary

- Loopy BP rediscovered by coding practitioners
- Understanding turbocodes as loopy BP led to development of many new and better codes
  - Current codes coming closer and closer to Shannon limit
- Resurgence of interest in BP led to much deeper understanding of approximate inference in graphical models
  - Many new algorithms