Probabilistic (Graphical) Models

and inference

Oliver Obst · Autumn 2024

WESTERN SYDNEY
UNIVERSITY

Probabilistic (Graphical) Models and Inference

(PGM: Probabilistic Graphical Models: Principles and Techniques by Daphne Koller and Nir Friedman. MIT Press)
(PMLI: Probabilistic Machine Learning: An introduction by Kevin Murphy. MIT Press)

Week	Lecture	Required reading	Assessment
1 Monday, 4 March 2024	Introduction, Probability Theory	PGM Chapter 2, PMLI Chapter 6.1	
2 Monday, 11 March 2024	Directed and undirected networks introduction	PGM Chapter 3 & 4	Quiz 1
3 Monday, 18 March 2024	Variable elimination	PGM Chapter 9	
4 Monday, 25 March 2024	Belief propagation	PGM Chapter 10/11	Quiz 2
5 Monday, 1 April 2024	public holiday		5 April 2024: census date
6 Monday, 8 April 2024	Message passing / Graph neural networks	https://distill.pub/2021/gnn-intro/	
7 Monday, 15 April 2024	Sampling		Quiz 3
8 Monday, 22 April 2024	Mid-term break		
9 Monday, 29 April 2024	Variational inference		Intra-session exam
10 Monday, 6 May 2024	Autoregressive models		Quiz 4
11 Monday, 13 May 2024	Variational Auto-Encoders		
12 Monday, 20 May 2024	GANs		Quiz 5
13 Monday, 27 May 2024	Energy-based models		
14 Monday, 3 June 2024	Evaluating generative models		Quiz 6
Monday, 17 June 2024			Project due

Conditional probability queries

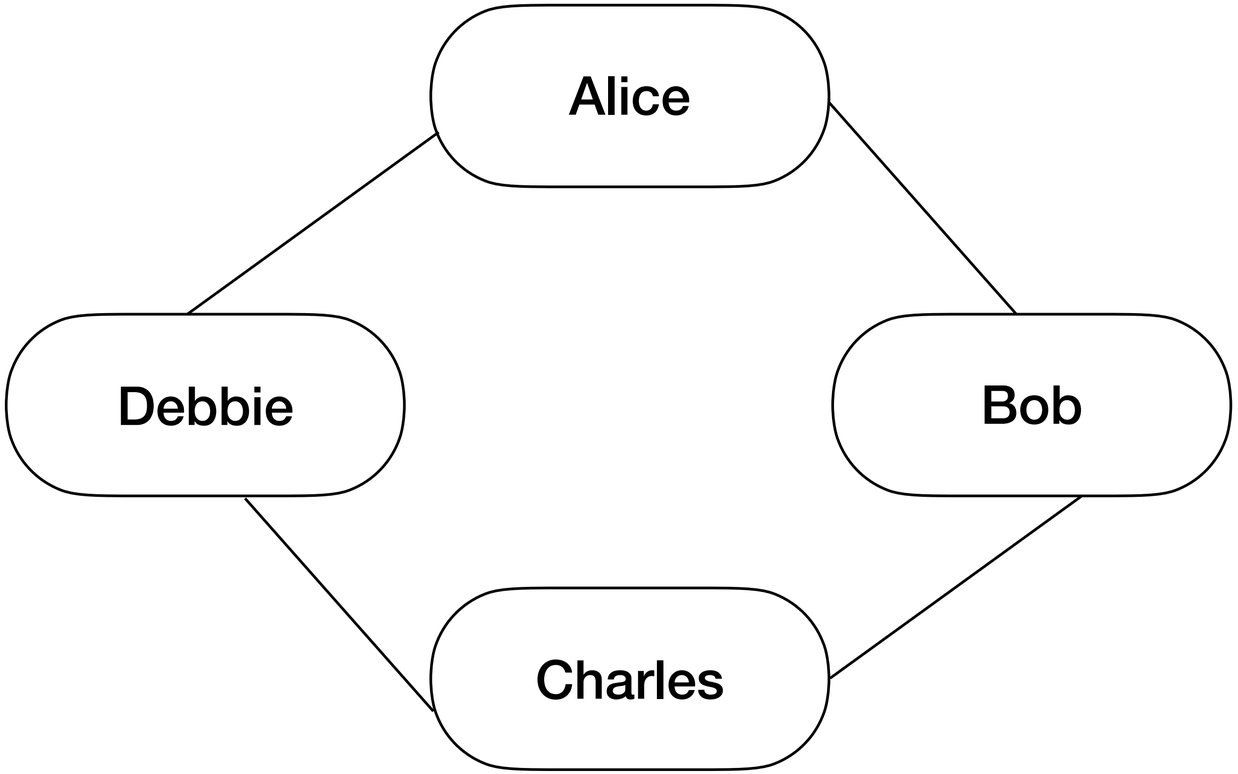
- Evidence: $\boldsymbol{E} = \boldsymbol{e}$
- Query: a subset of variables $\boldsymbol{Y}$
- Task: compute $P(\boldsymbol{Y} \mid \boldsymbol{E} = \boldsymbol{e})$

Message passing algorithms

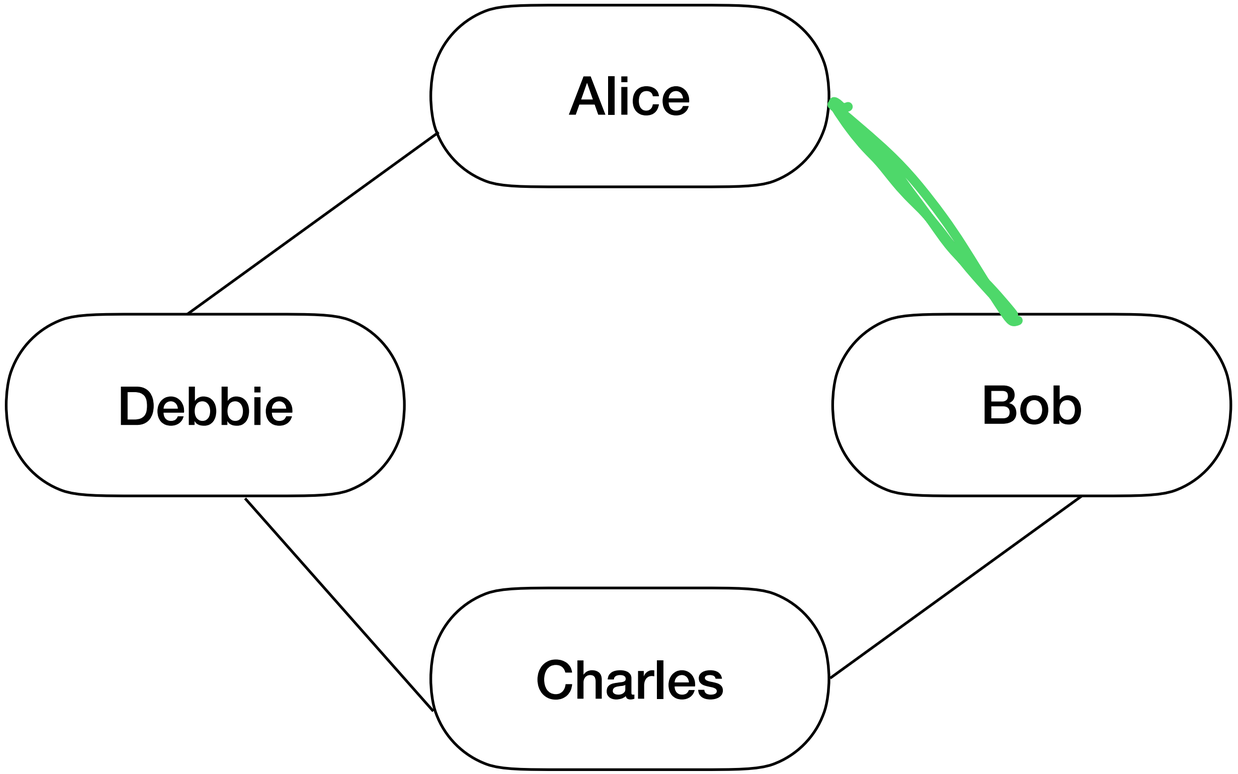
- We continue to do inference on graphical models
- But instead of Variable Elimination (last week), we use “Message Passing”

Markov Networks

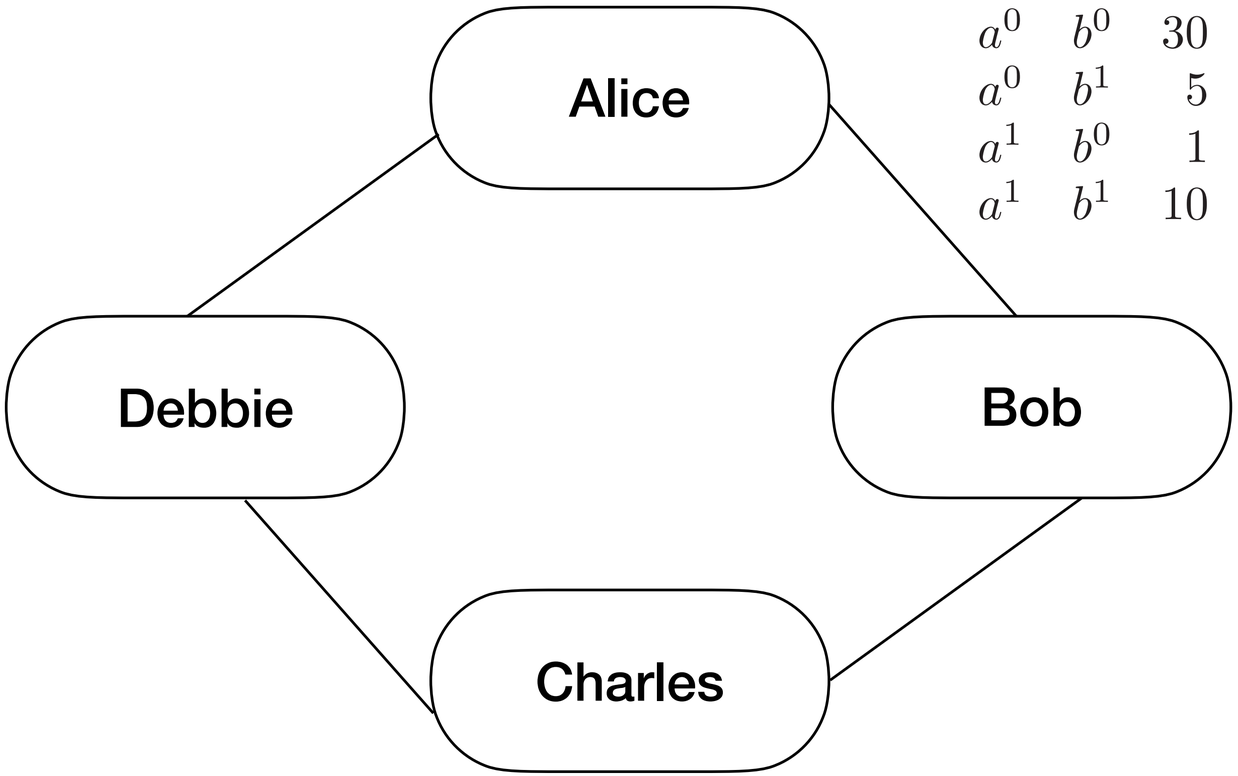
Pairwise Markov Networks



Pairwise Markov Networks



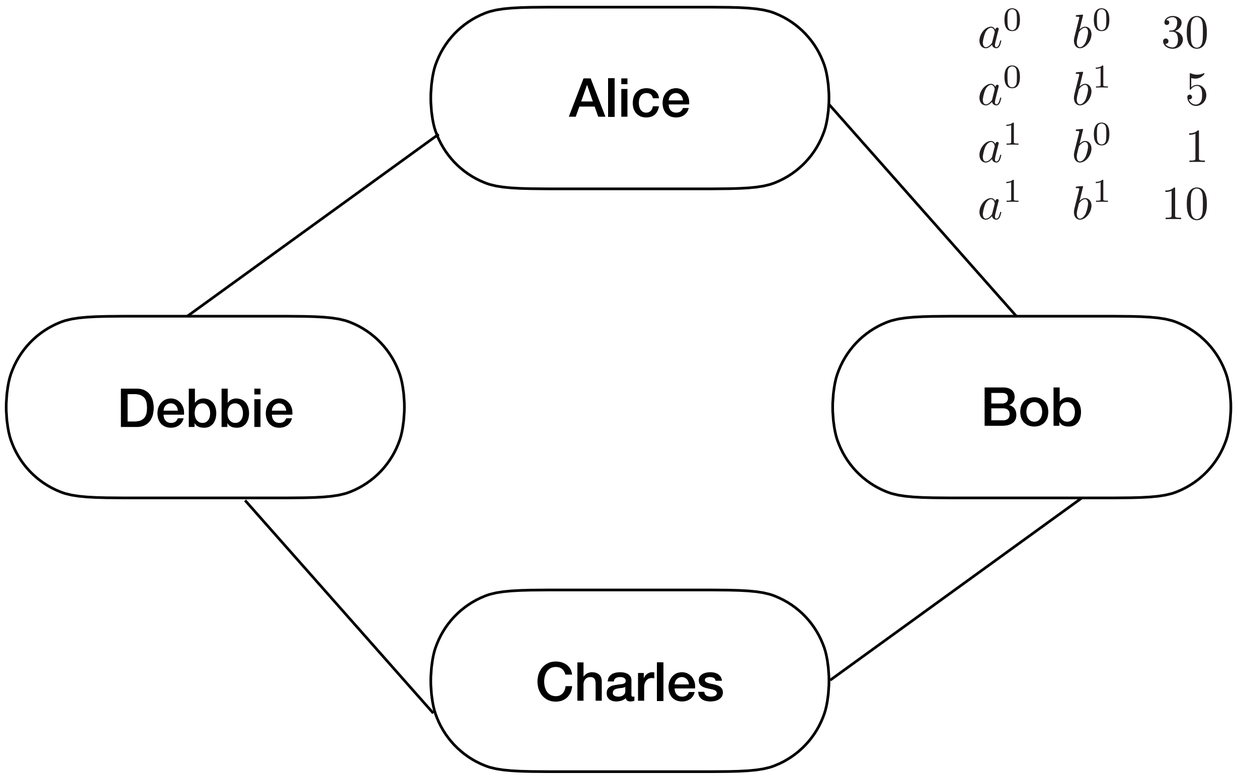
Pairwise Markov Networks



$\phi_1(A, B)$

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

Pairwise Markov Networks

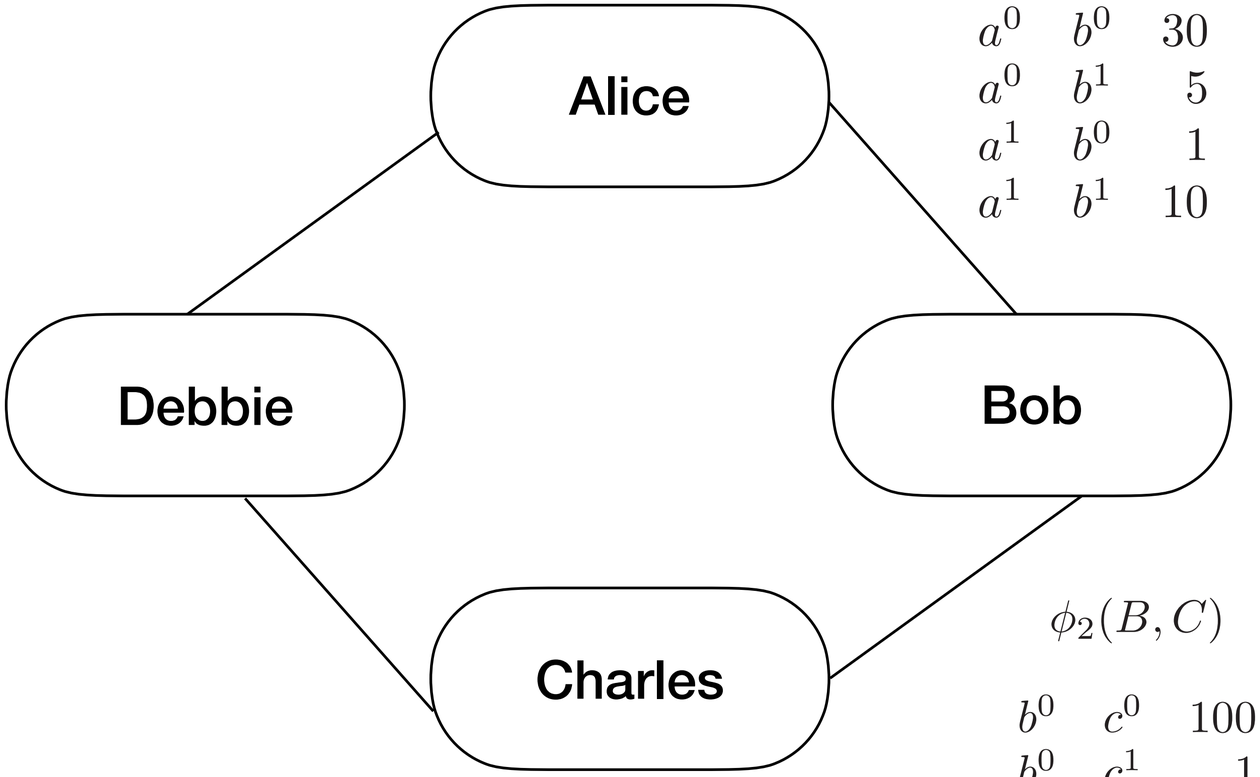


$\phi_1(A, B)$

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

- affinity
- compatibility
- soft constraints

Pairwise Markov Networks



$\phi_1(A, B)$

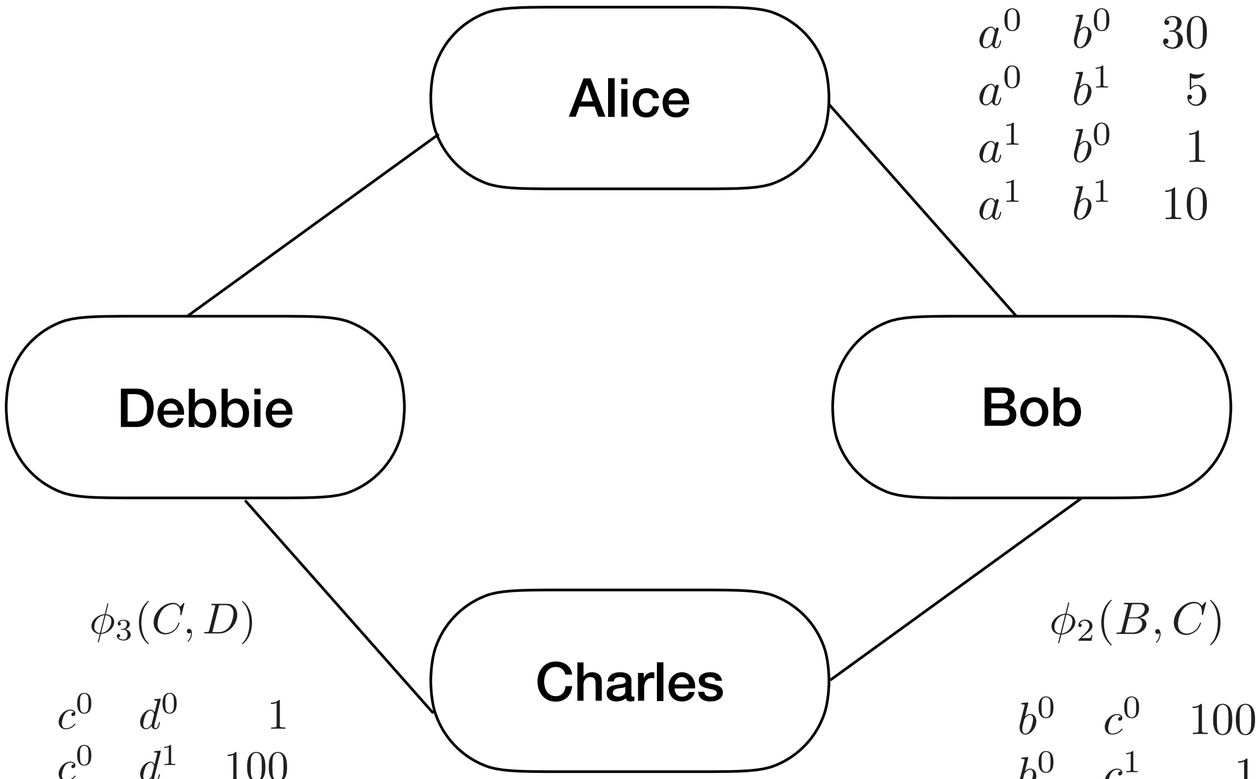
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

- affinity
- compatibility
- soft constraints

$\phi_2(B, C)$

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

Pairwise Markov Networks



$\phi_1(A, B)$

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

$\phi_3(C, D)$

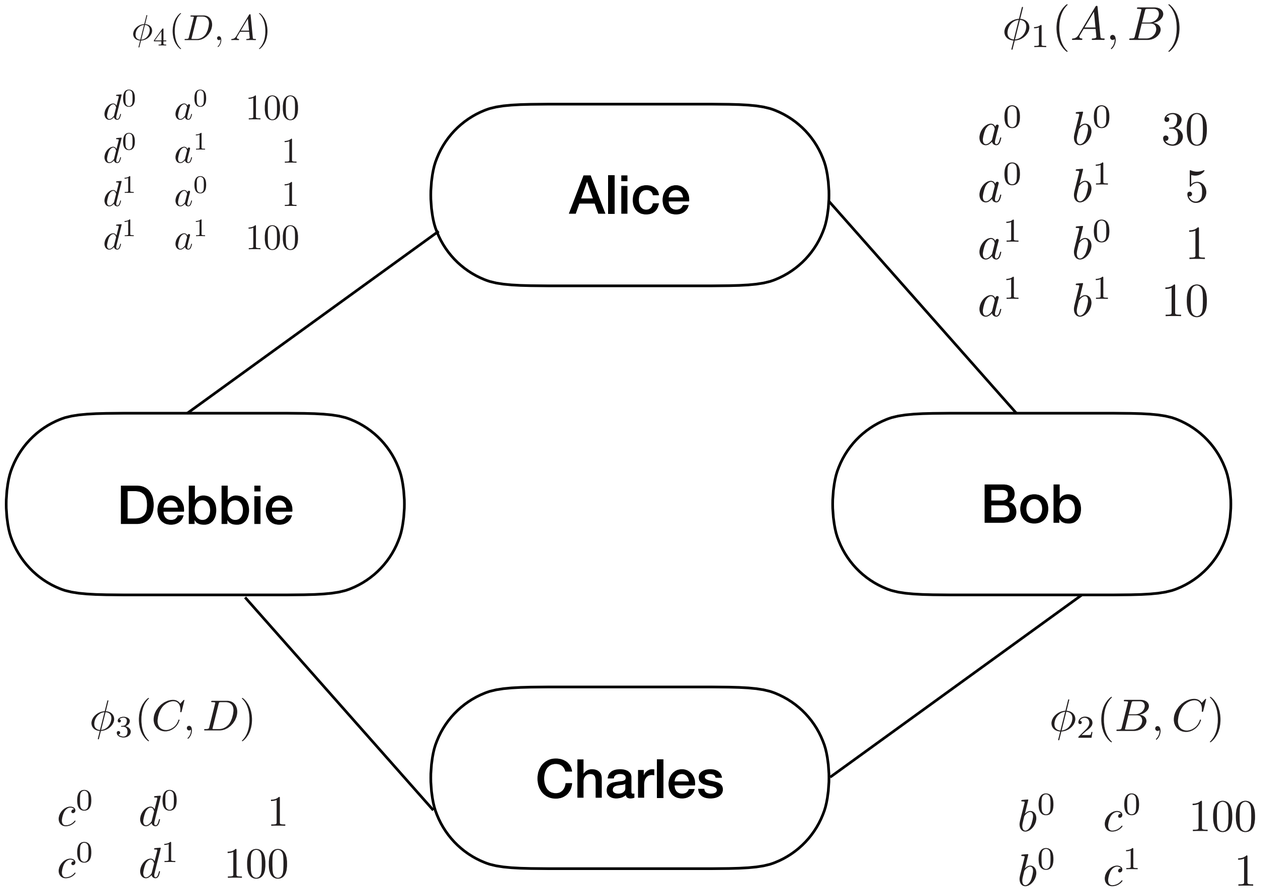
c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

$\phi_2(B, C)$

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

- affinity
- compatibility
- soft constraints

Pairwise Markov Networks



$\phi_4(D, A)$

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

- affinity
- compatibility
- soft constraints

$$\tilde{P}(A, B, C, D) = \phi_1(A, B) \times \phi_2(B, C) \times \phi_3(C, D) \times \phi_4(D, A)$$

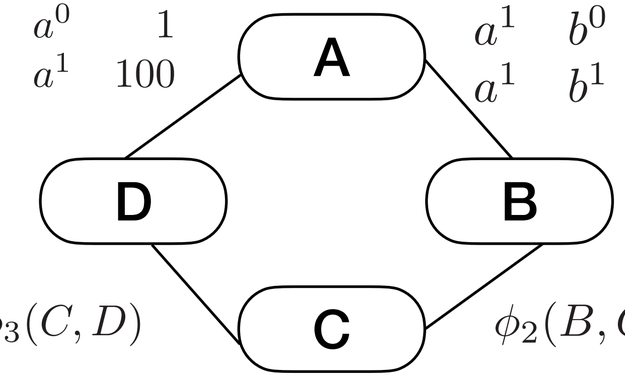
Unnormalised measure

$\phi_4(D, A)$

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

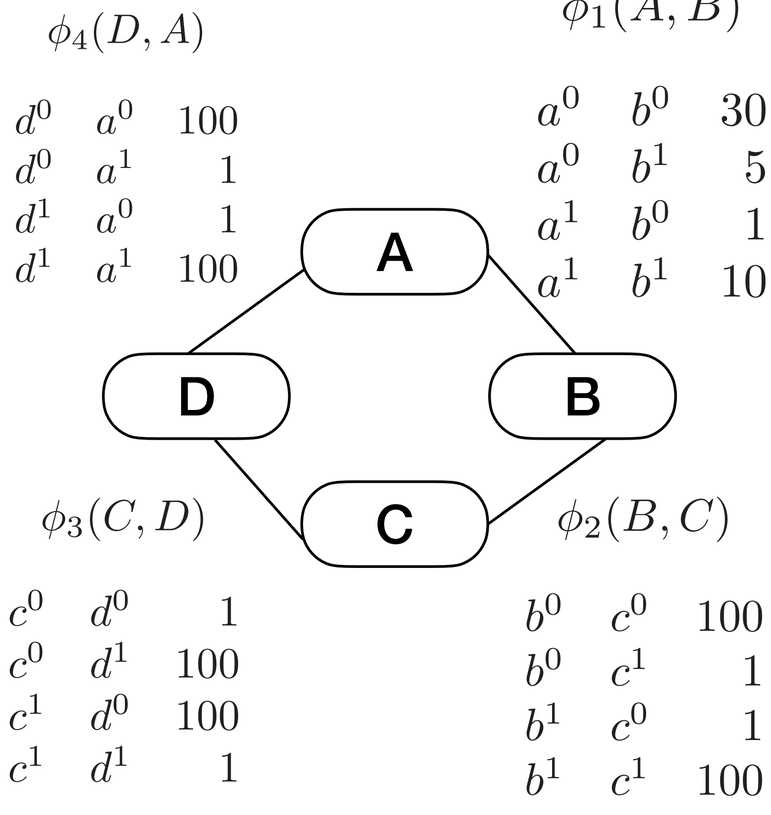
$$\tilde{P}(A,B,C,D) = \phi_1(A,B) \times \phi_2(B,C) \times \phi_3(C,D) \times \phi_4(D,A)$$

Unnormalised measure

$$P(A,B,C,D) = \frac{1}{Z}\tilde{P}(A,B,C,D)$$

Partition function

$Assignment$				$Unnormalized$
a^0	b^0	c^0	d^0	$300,000$
a^0	b^0	c^0	d^1	$300,000$
a^0	b^0	c^1	d^0	$300,000$
a^0	b^0	c^1	d^1	30
a^0	b^1	c^0	d^0	500
a^0	b^1	c^0	d^1	500
a^0	b^1	c^1	d^0	$5,000,000$
a^0	b^1	c^1	d^1	500
a^1	b^0	c^0	d^0	100
a^1	b^0	c^0	d^1	$1,000,000$
a^1	b^0	c^1	d^0	100
a^1	b^0	c^1	d^1	100
a^1	b^1	c^0	d^0	10
a^1	b^1	c^0	d^1	$100,000$
a^1	b^1	c^1	d^0	$100,000$
a^1	b^1	c^1	d^1	$100,000$



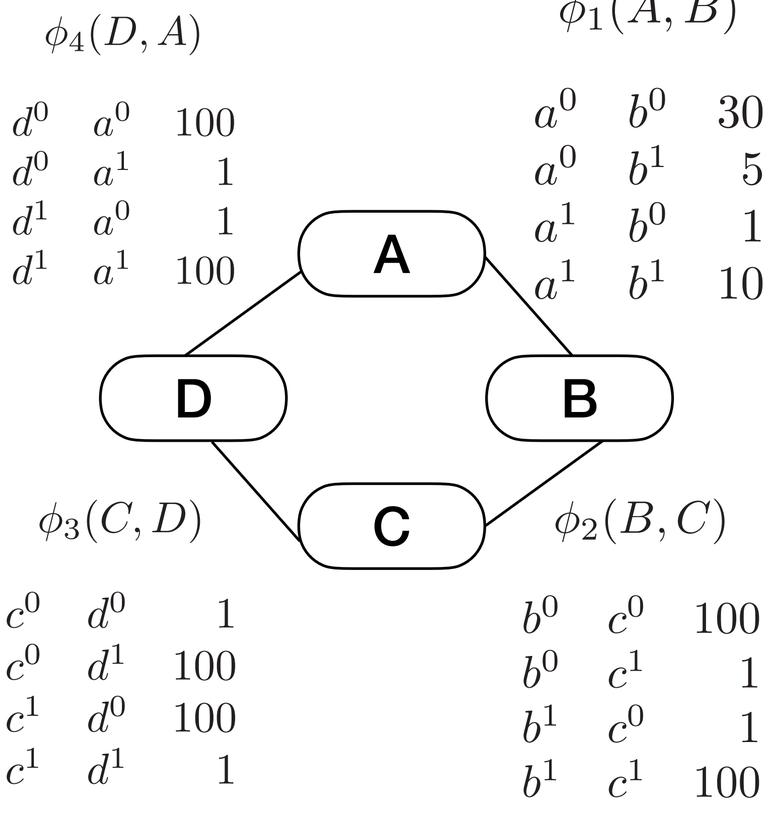
$$\tilde{P}(A,B,C,D) = \phi_1(A,B) \times \phi_2(B,C) \times \phi_3(C,D) \times \phi_4(D,A)$$

Unnormalised measure

$$P(A,B,C,D) = \frac{1}{Z}\tilde{P}(A,B,C,D)$$

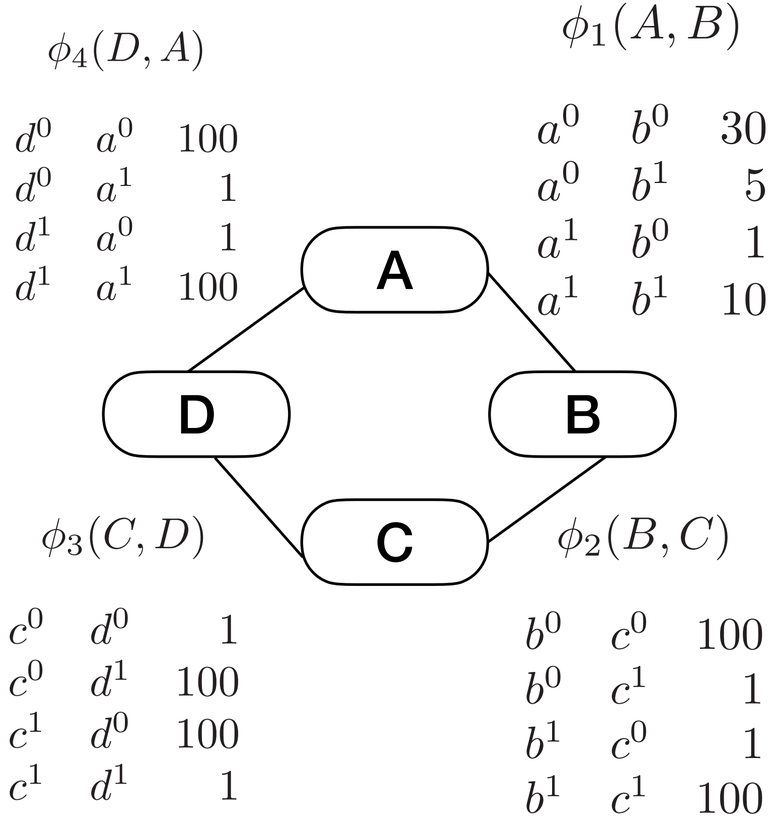
Partition function

$Assignment$				$Unnormalized$	$Normalized$
a^0	b^0	c^0	d^0	$300,000$	0.04
a^0	b^0	c^0	d^1	$300,000$	0.04
a^0	b^0	c^1	d^0	$300,000$	0.04
a^0	b^0	c^1	d^1	30	$4.1 \cdot 10^{-6}$
a^0	b^1	c^0	d^0	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^0	d^1	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^1	d^0	$5,000,000$	0.69
a^0	b^1	c^1	d^1	500	$6.9 \cdot 10^{-5}$
a^1	b^0	c^0	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^0	d^1	$1,000,000$	0.14
a^1	b^0	c^1	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^1	d^1	100	$1.4 \cdot 10^{-5}$
a^1	b^1	c^0	d^0	10	$1.4 \cdot 10^{-6}$
a^1	b^1	c^0	d^1	$100,000$	0.014
a^1	b^1	c^1	d^0	$100,000$	0.014
a^1	b^1	c^1	d^1	$100,000$	0.014
				Z	



What does the pairwise factor $\phi_1(A, B)$ mean?

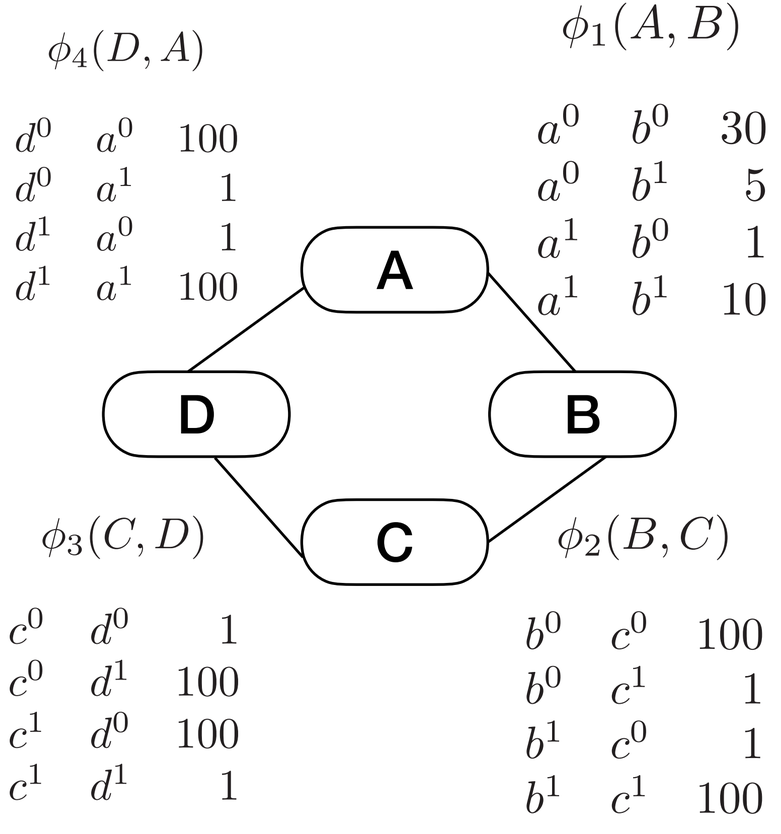
It is the local “happiness” between A and B. How does it relate to a probability distribution?



What does the pairwise factor $\phi_1(A, B)$ mean?

It is the local “happiness” between A and B. How does it relate to a probability distribution?

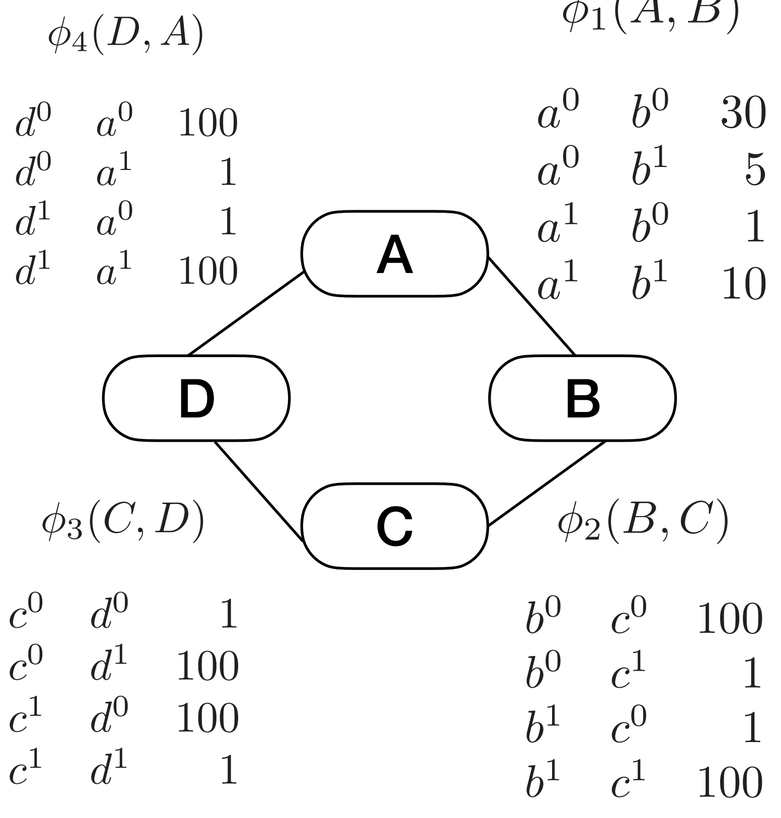
- $P(A, B)$
- $P(A \mid B)$
- $P(A, B \mid C, D)$
- ?



What does the pairwise factor $\phi_1(A, B)$ mean?

$$P_\phi(A, B) \qquad \phi = \{\phi_1, \phi_2, \phi_3, \phi_4\}$$

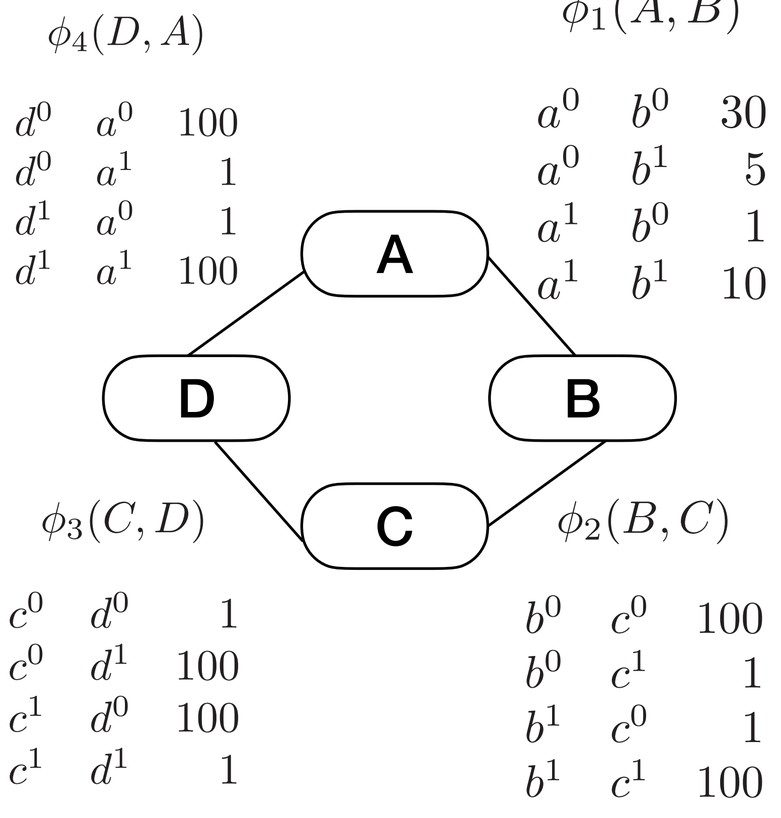
a^0	b^0	0.13
a^0	b^1	0.69
a^1	b^0	0.14
a^1	b^1	0.04



What does the pairwise factor $\phi_1(A, B)$ mean?

$$P_\phi(A, B) \qquad \phi = \{\phi_1, \phi_2, \phi_3, \phi_4\}$$

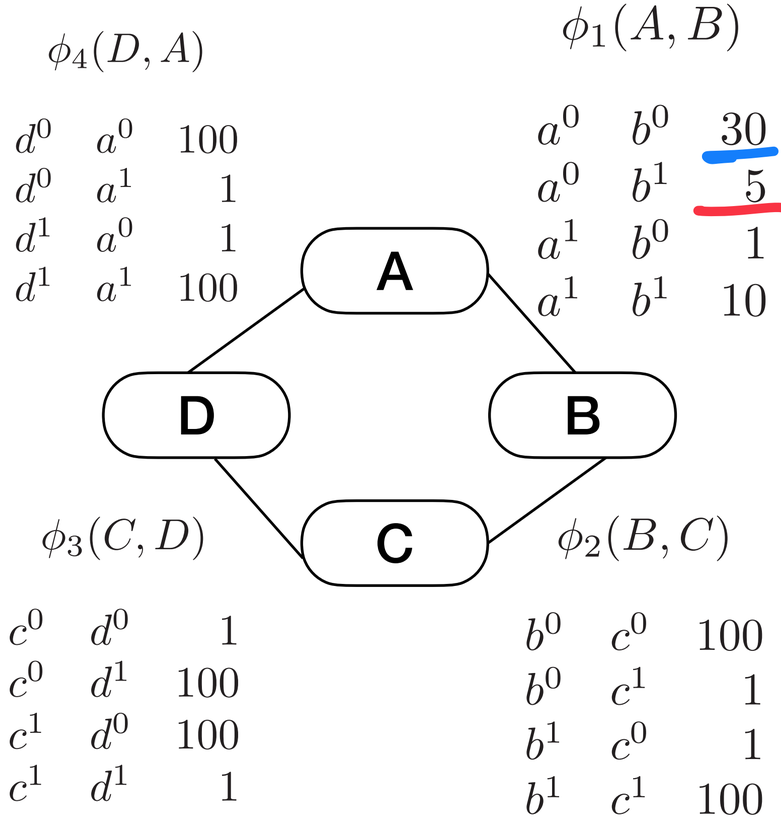
a^0	b^0	0.13
a^0	b^1	0.69
a^1	b^0	0.14
a^1	b^1	0.04



What does the pairwise factor $\phi_1(A, B)$ mean?

$$P_\phi(A, B) \qquad \phi = \{\phi_1, \phi_2, \phi_3, \phi_4\}$$

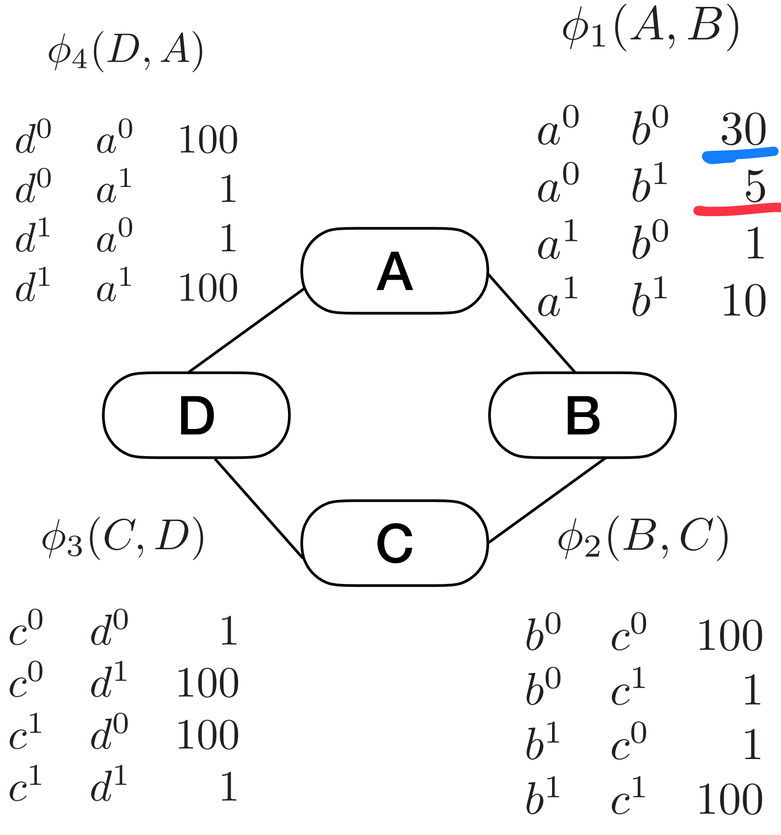
a^0	b^0	0.13
a^0	b^1	0.69
a^1	b^0	0.14
a^1	b^1	0.04



What does the pairwise factor $\phi_1(A, B)$ mean?

$$P_\phi(A, B) \qquad \phi = \{\phi_1, \phi_2, \phi_3, \phi_4\}$$

a^0	b^0	0.13
a^0	b^1	0.69
a^1	b^0	0.14
a^1	b^1	0.04

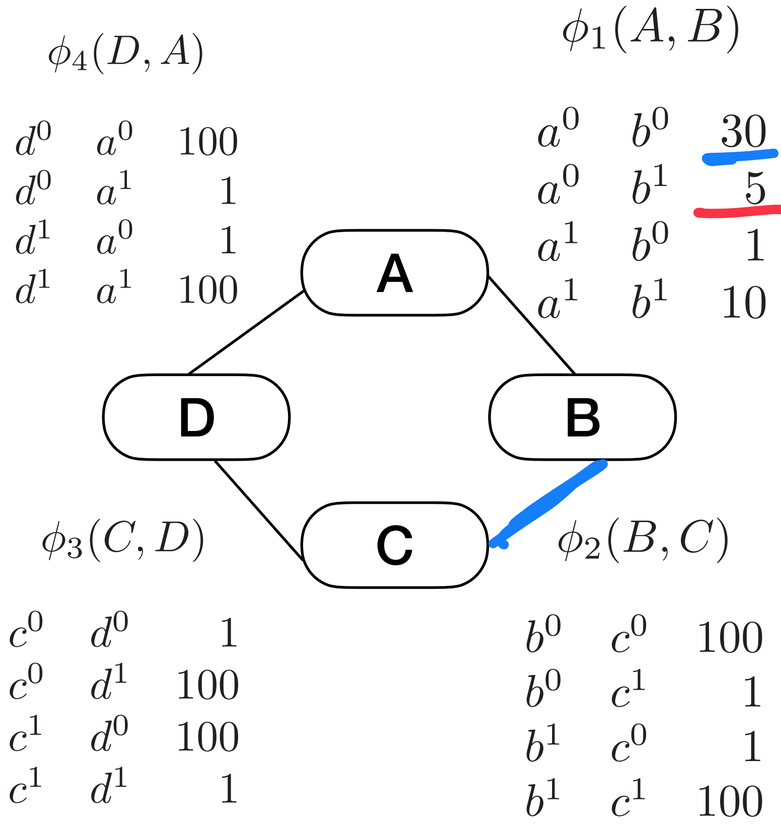


What does the pairwise factor $\phi_1(A, B)$ mean?

$P_\phi(A, B)$

a^0	b^0	0.13
a^0	b^1	0.69
a^1	b^0	0.14
a^1	b^1	0.04

$\phi = \{\phi_1, \phi_2, \phi_3, \phi_4\}$

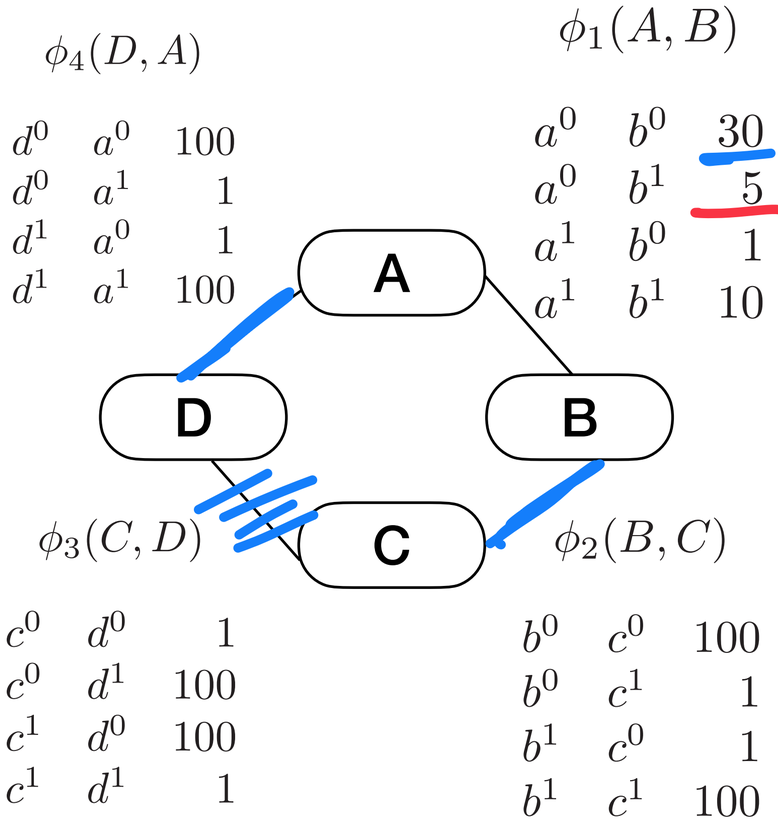


What does the pairwise factor $\phi_1(A, B)$ mean?

$P_\phi(A, B)$

a^0	b^0	0.13
a^0	b^1	0.69
a^1	b^0	0.14
a^1	b^1	0.04

$\phi = \{\phi_1, \phi_2, \phi_3, \phi_4\}$



Pairwise Markov Networks

- A pairwise Markov network is an undirected graph whose nodes are random variables $X_1, \ldots, X_n$

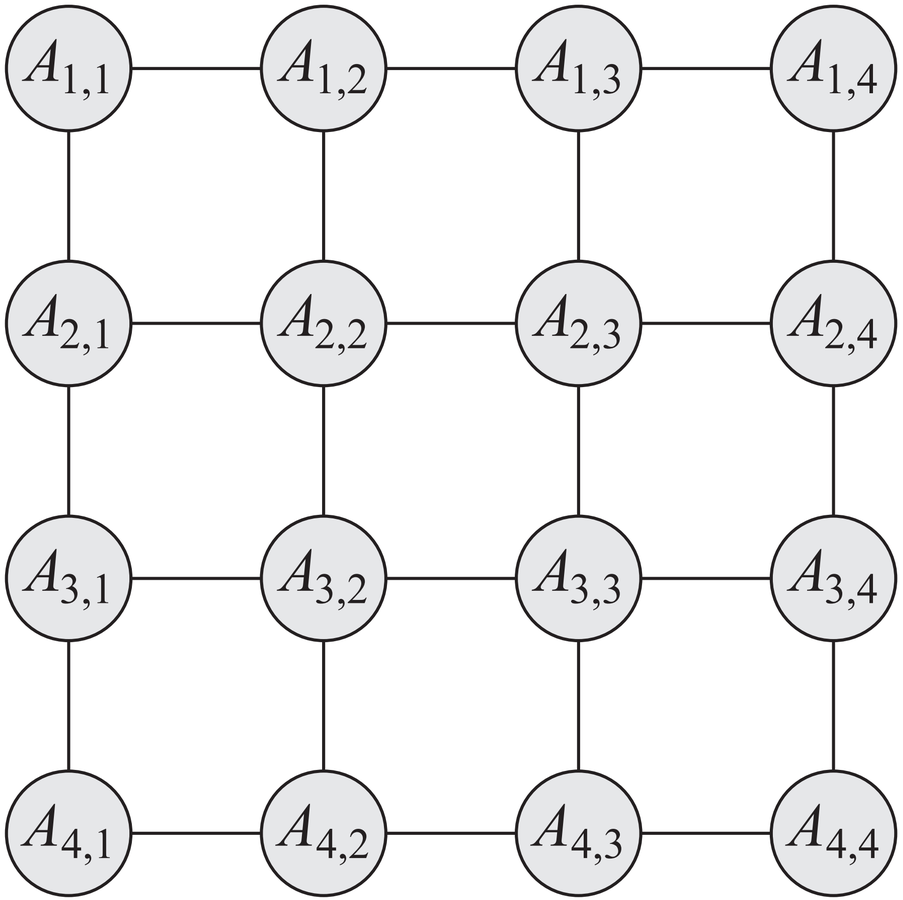
  and each edge $X_i - X_j$ is associated with a factor (aka potential) $\phi_{ij}(X_i, X_j)$

Pairwise Markov Networks

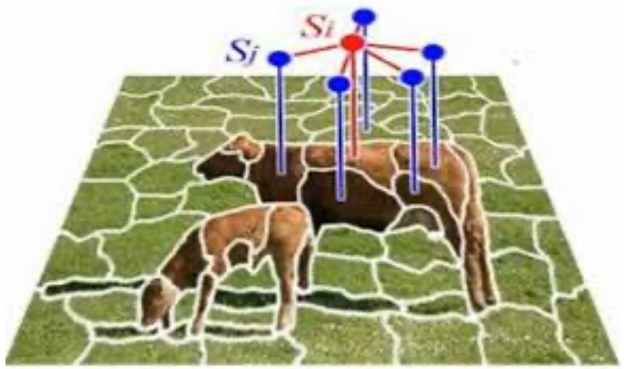
- A pairwise Markov network is an undirected graph whose nodes are random variables $X_1, \ldots, X_n$

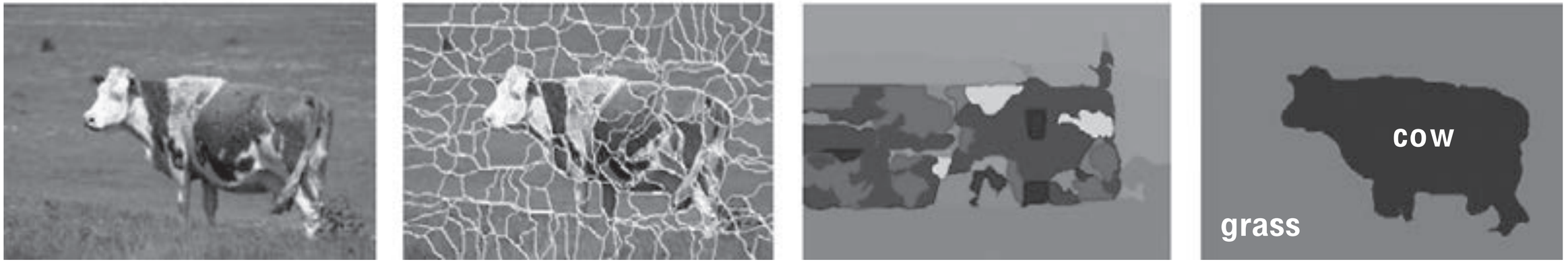
  and each edge $X_i - X_j$ is associated with a factor (aka potential) $\phi_{ij}(X_i, X_j)$

Pairwise Markov Networks



Pairwise Markov Networks





examples of image segmentation results (a) The original image. (b) An oversegmentation known as superpixels; each superpixel is associated with a random variable that designates its segment assignment. The use of superpixels reduces the size of the problems. (c) Result of segmentation using node potentials alone, so that each superpixel is classified independently. (d) Result of segmentation using a pairwise Markov network encoding interactions between adjacent superpixels.

Summary

- **Bayesian networks**
- Factors
- Markov networks

Belief Propagation Algorithm

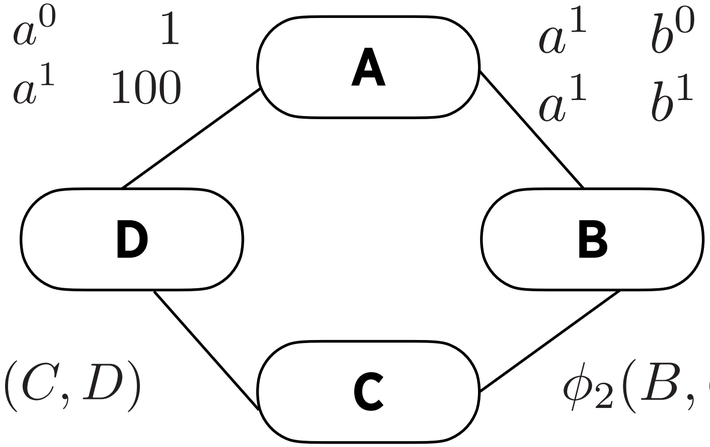
Cluster Graph

$\phi_4(D, A)$

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

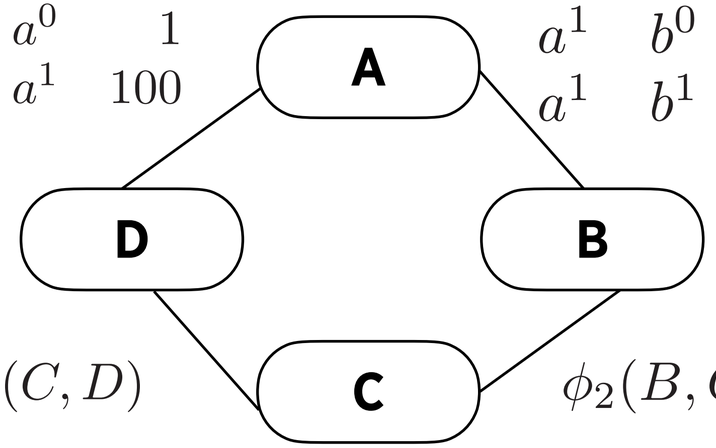
Cluster Graph

$\phi_4(D, A)$

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

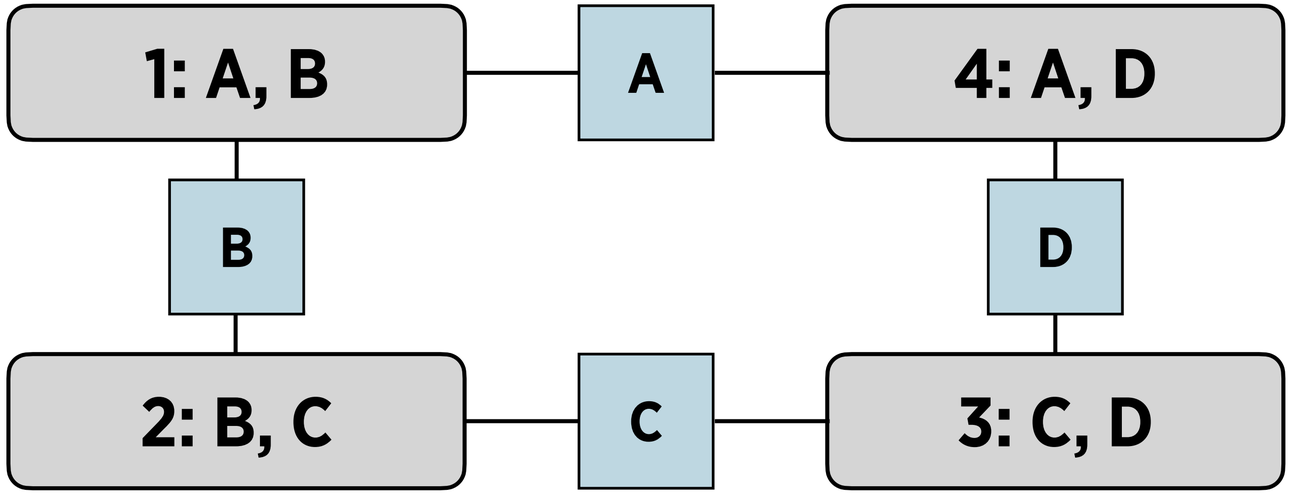


$\phi_3(C, D)$

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



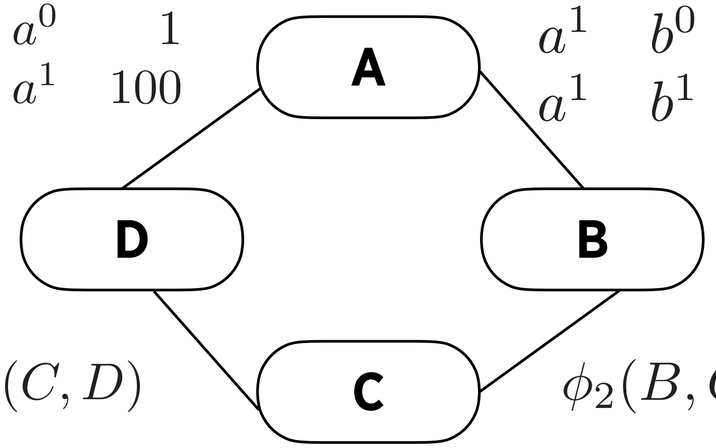
Cluster Graph

$\phi_4(D, A)$

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



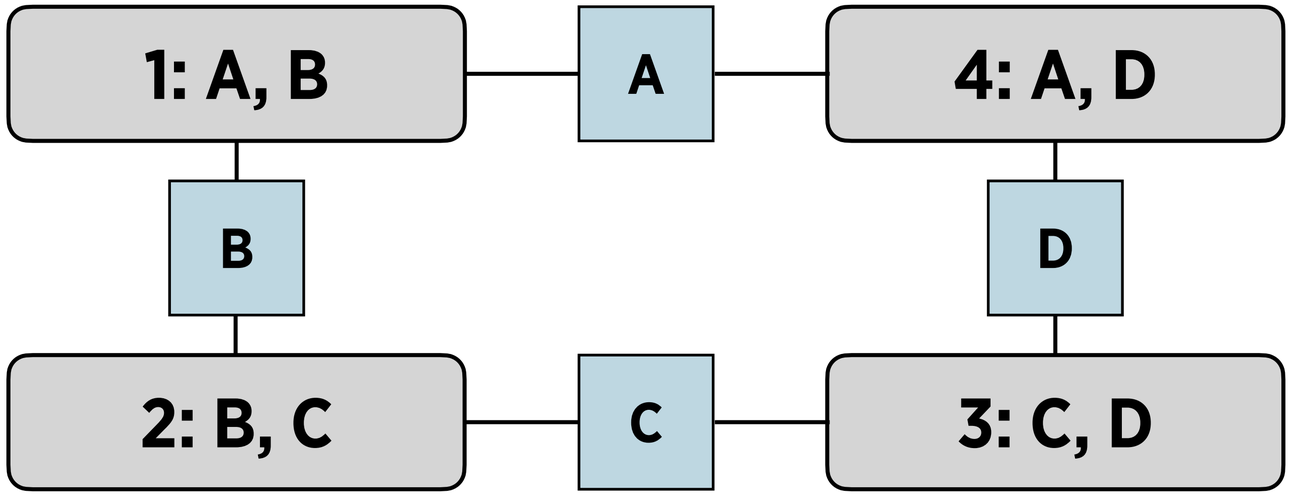
$\phi_3(C, D)$

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

clusters 1—4



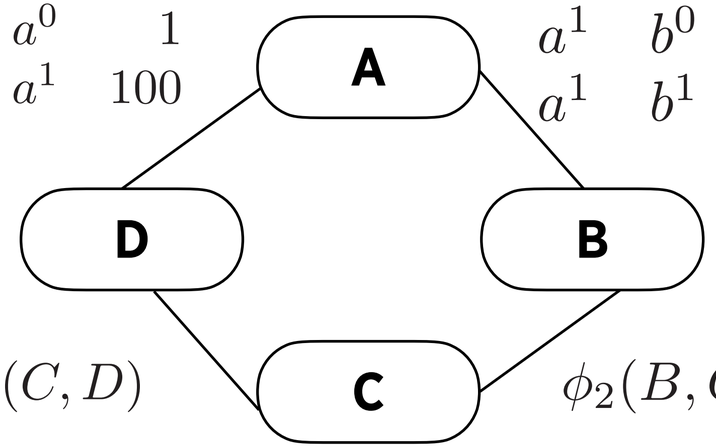
Cluster Graph

$\phi_4(D, A)$

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



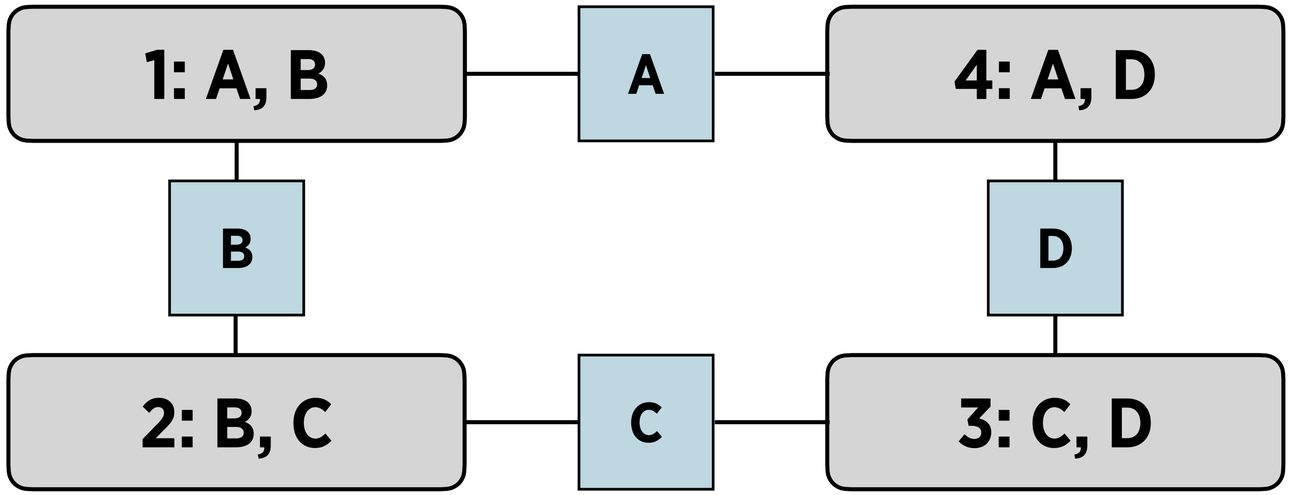
$\phi_3(C, D)$

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

clusters 1—4



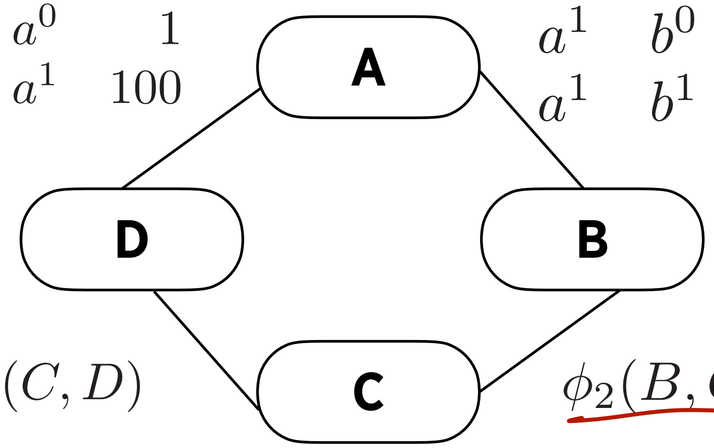
Cluster Graph

$\phi_4(D, A)$

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



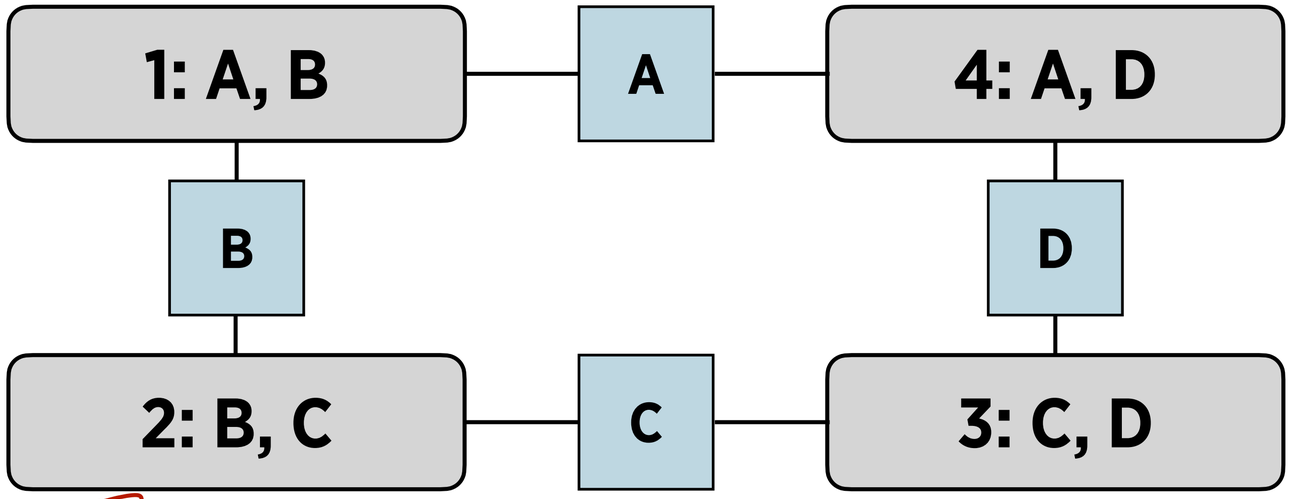
$\phi_3(C, D)$

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

clusters 1—4



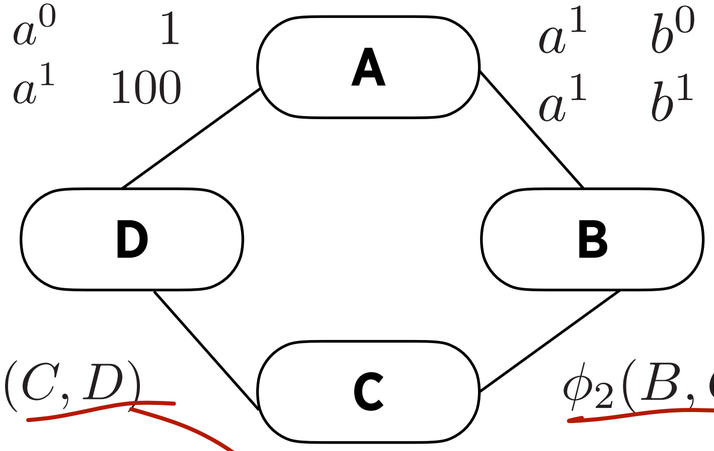
Cluster Graph

$\phi_4(D, A)$

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



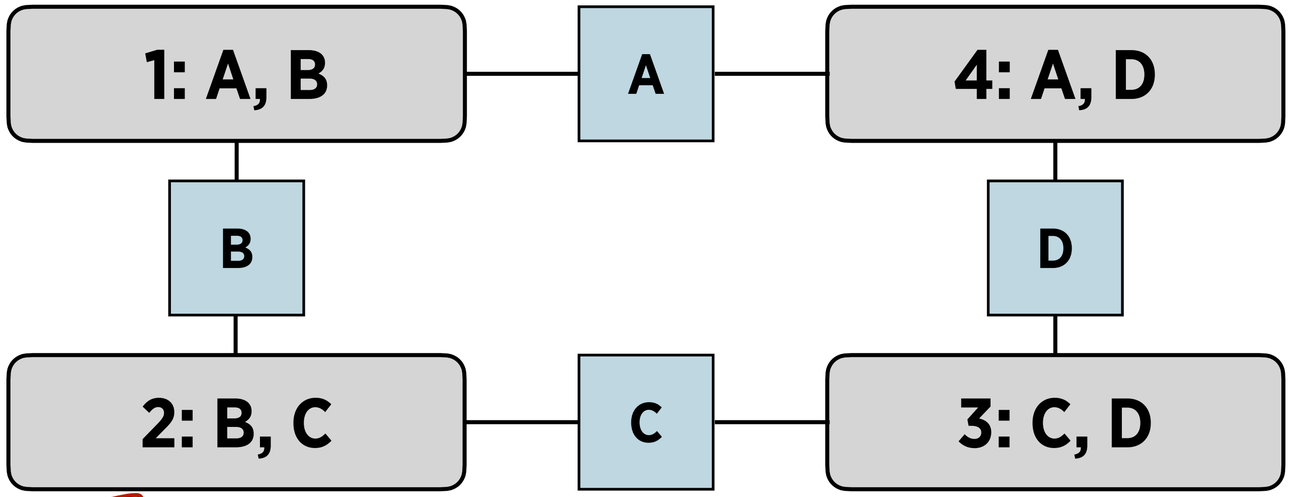
$\phi_3(C, D)$

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

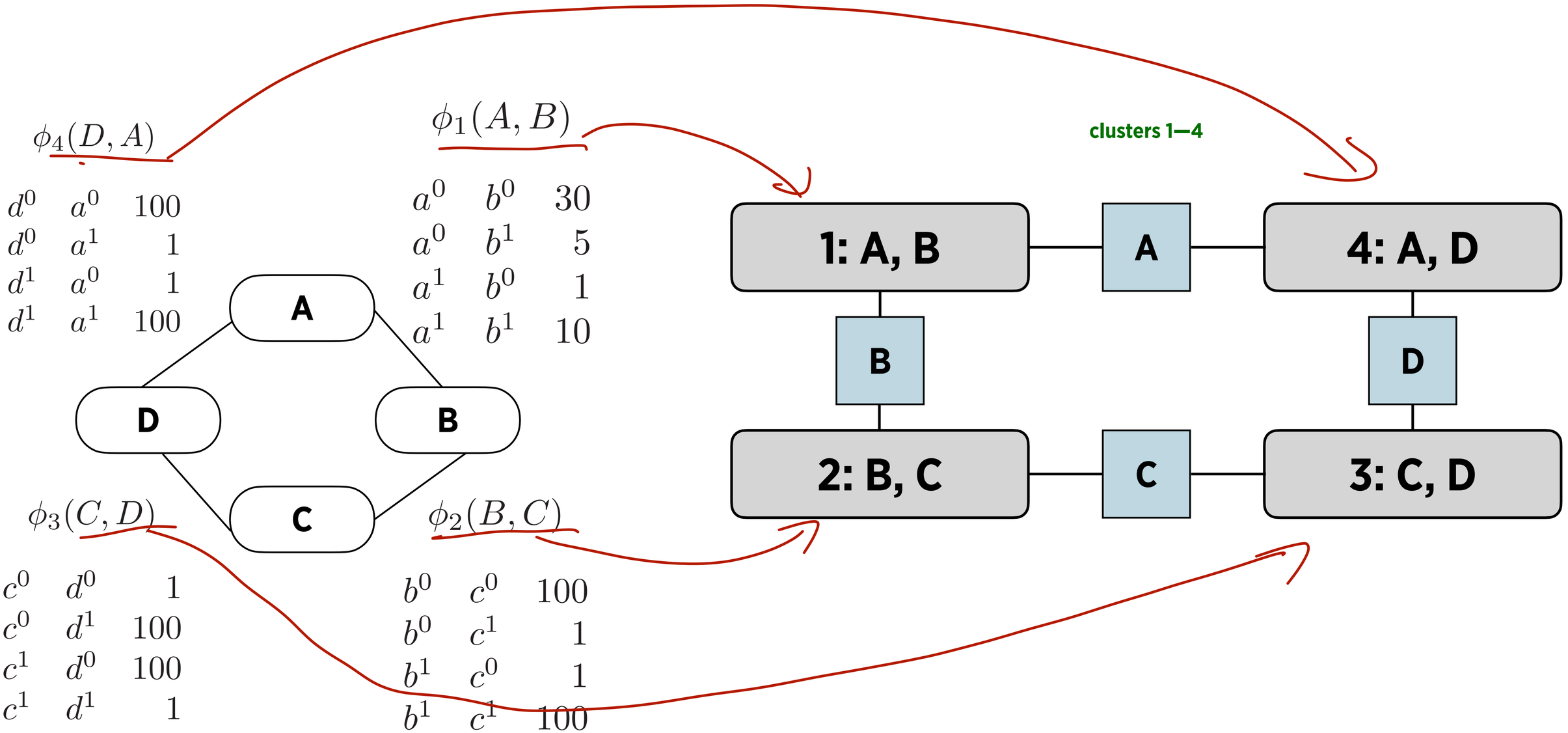
$\phi_2(B, C)$

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

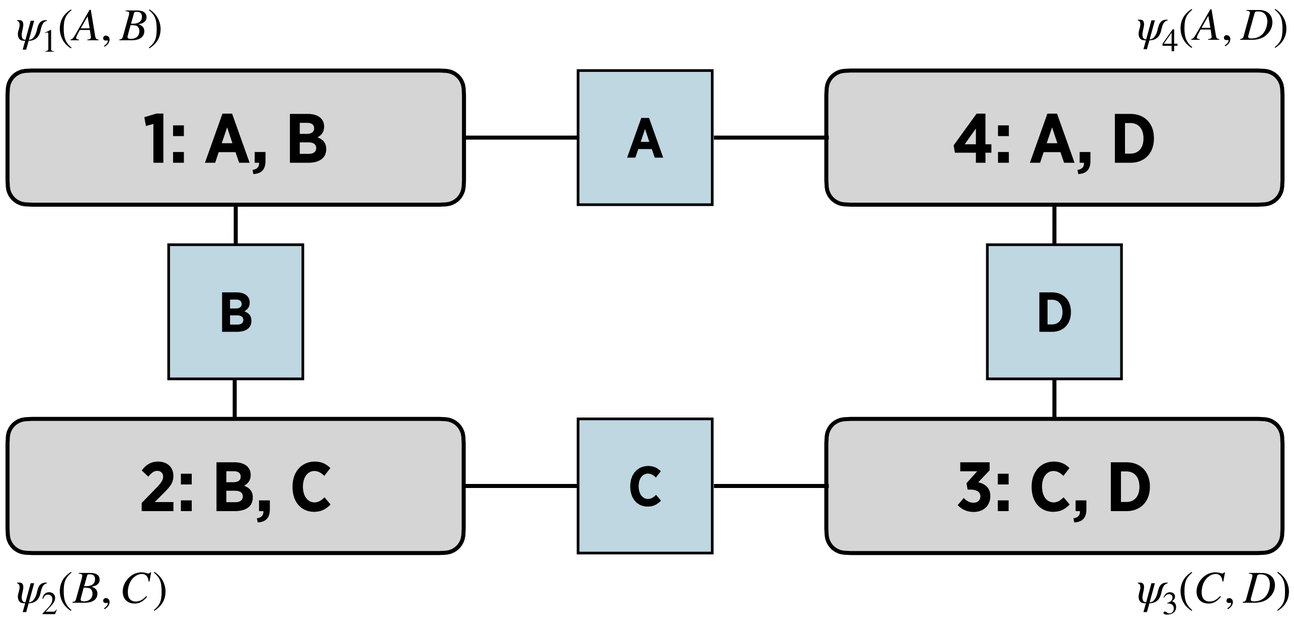
clusters 1—4



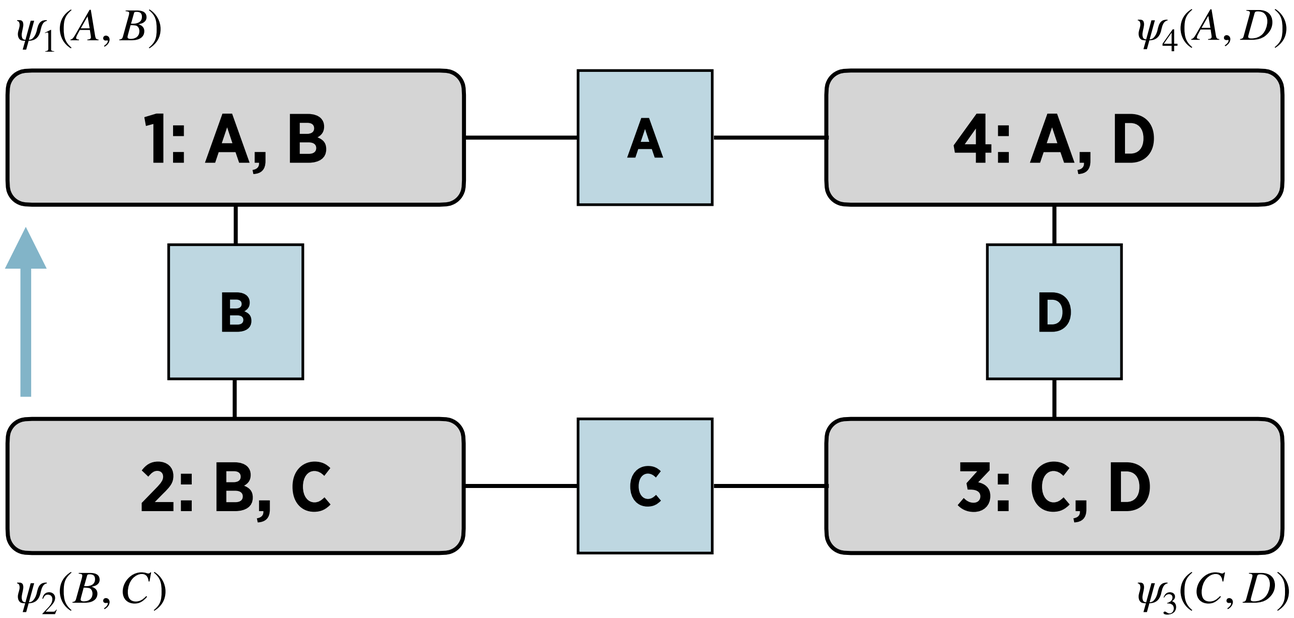
Cluster Graph



Passing messages

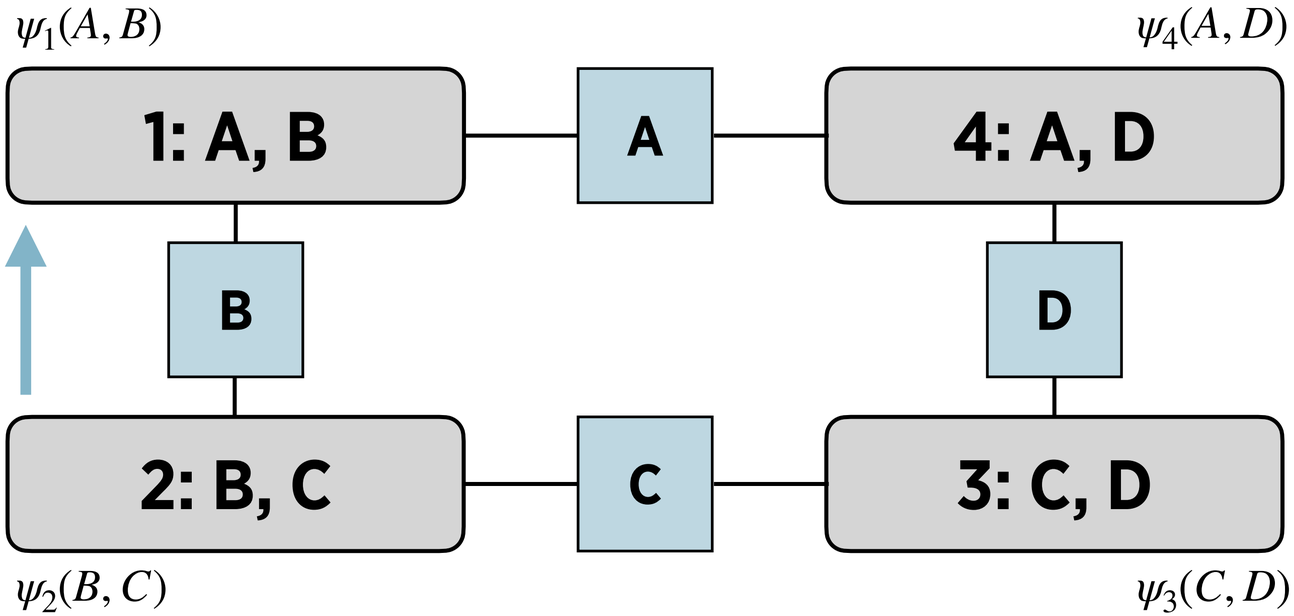


Passing messages

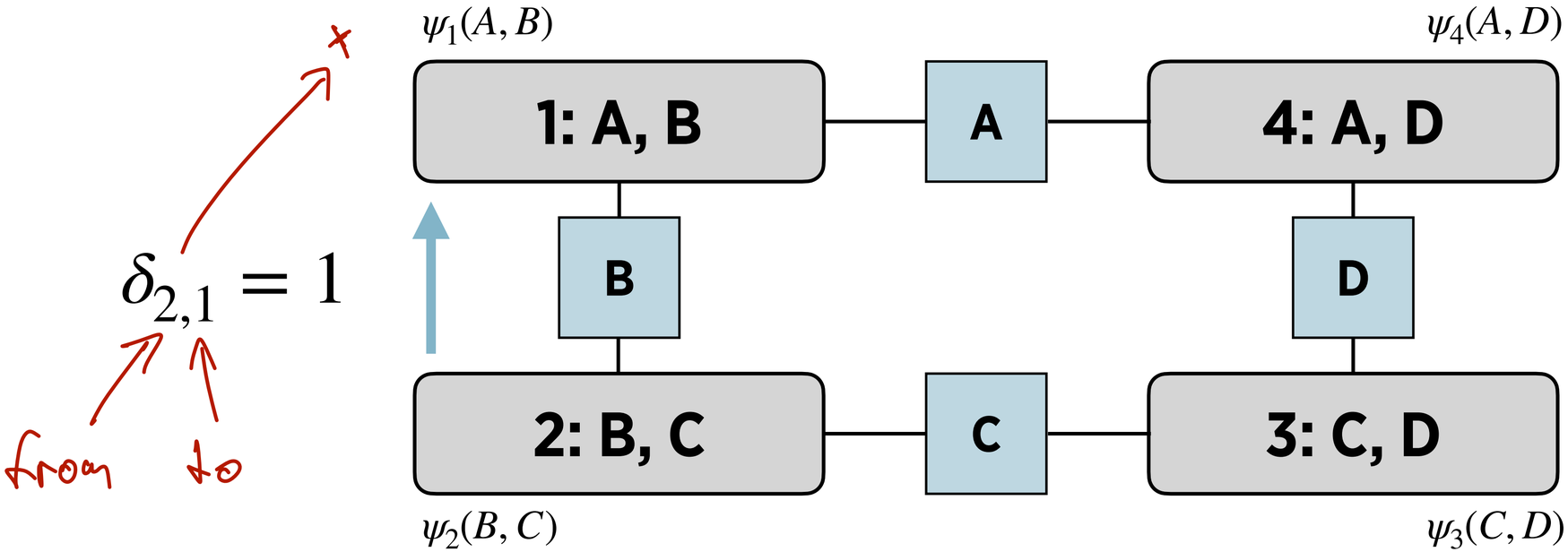


Passing messages

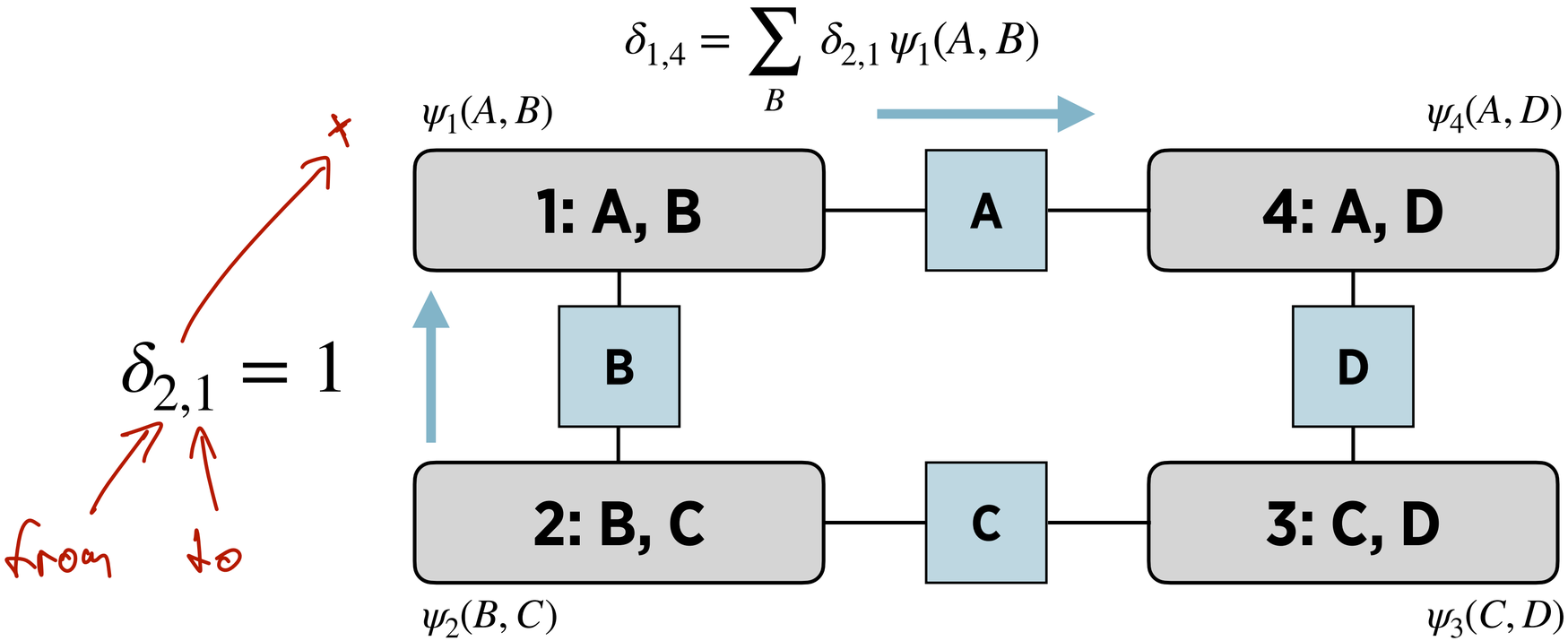
$$\delta_{2,1} = 1$$



Passing messages



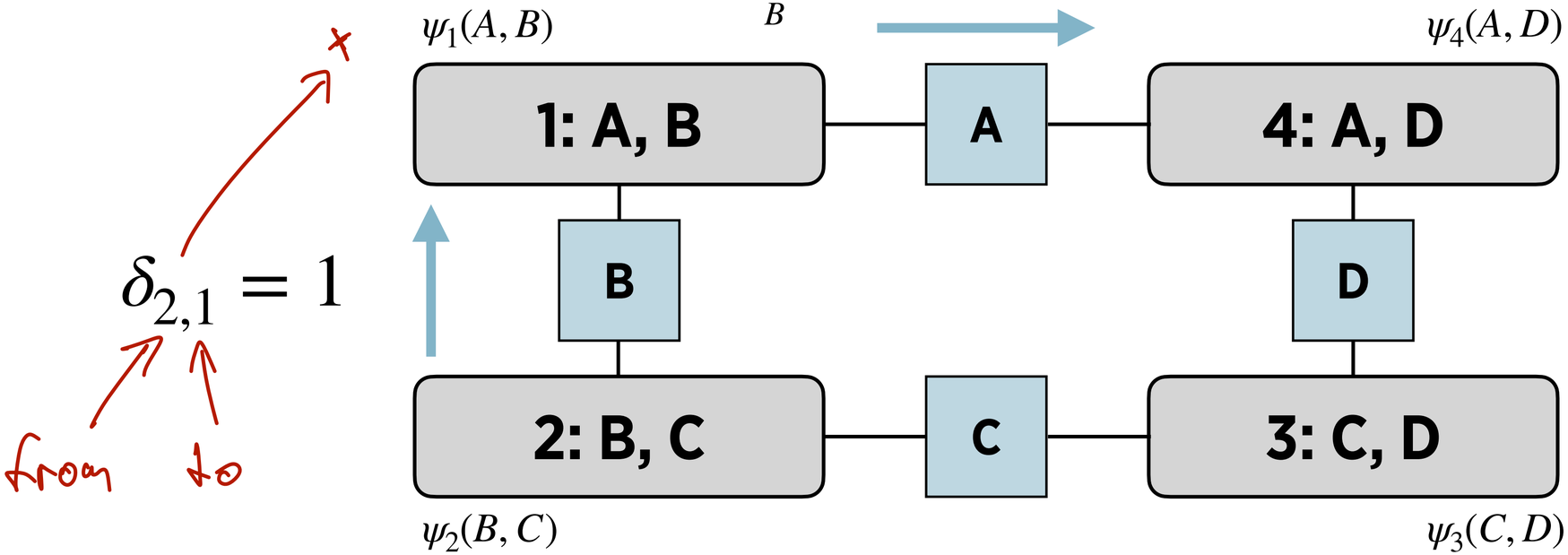
Passing messages



Passing messages

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

incoming message



$$\delta_{2,1} = 1$$

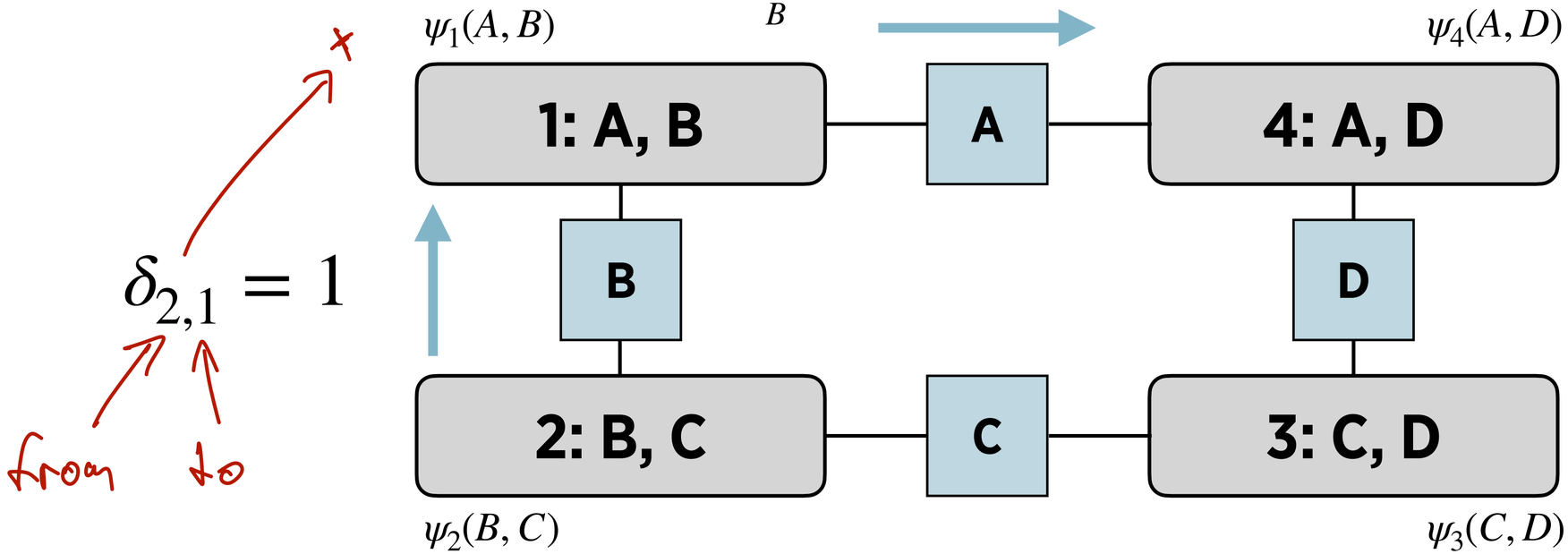
from

to

Passing messages

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

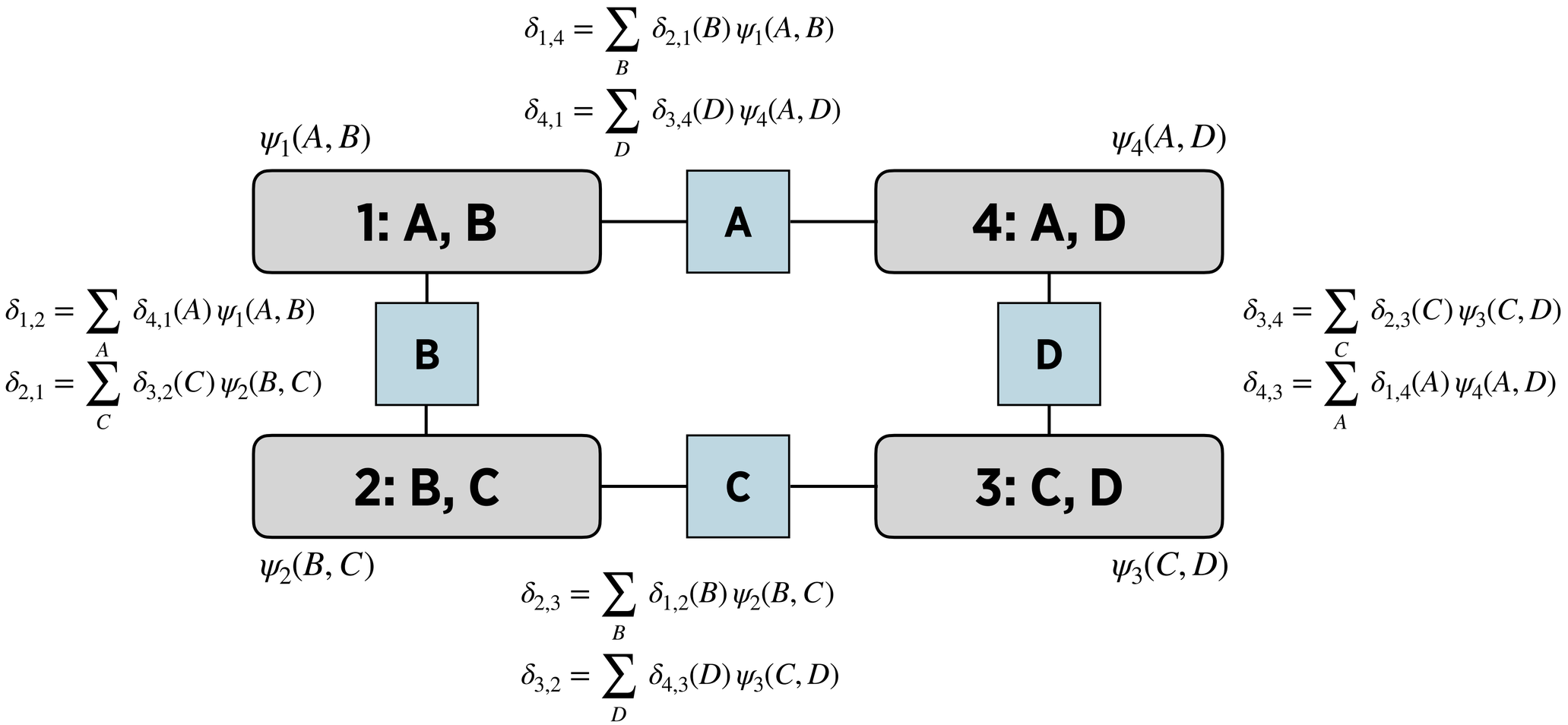
incoming message · initial factors



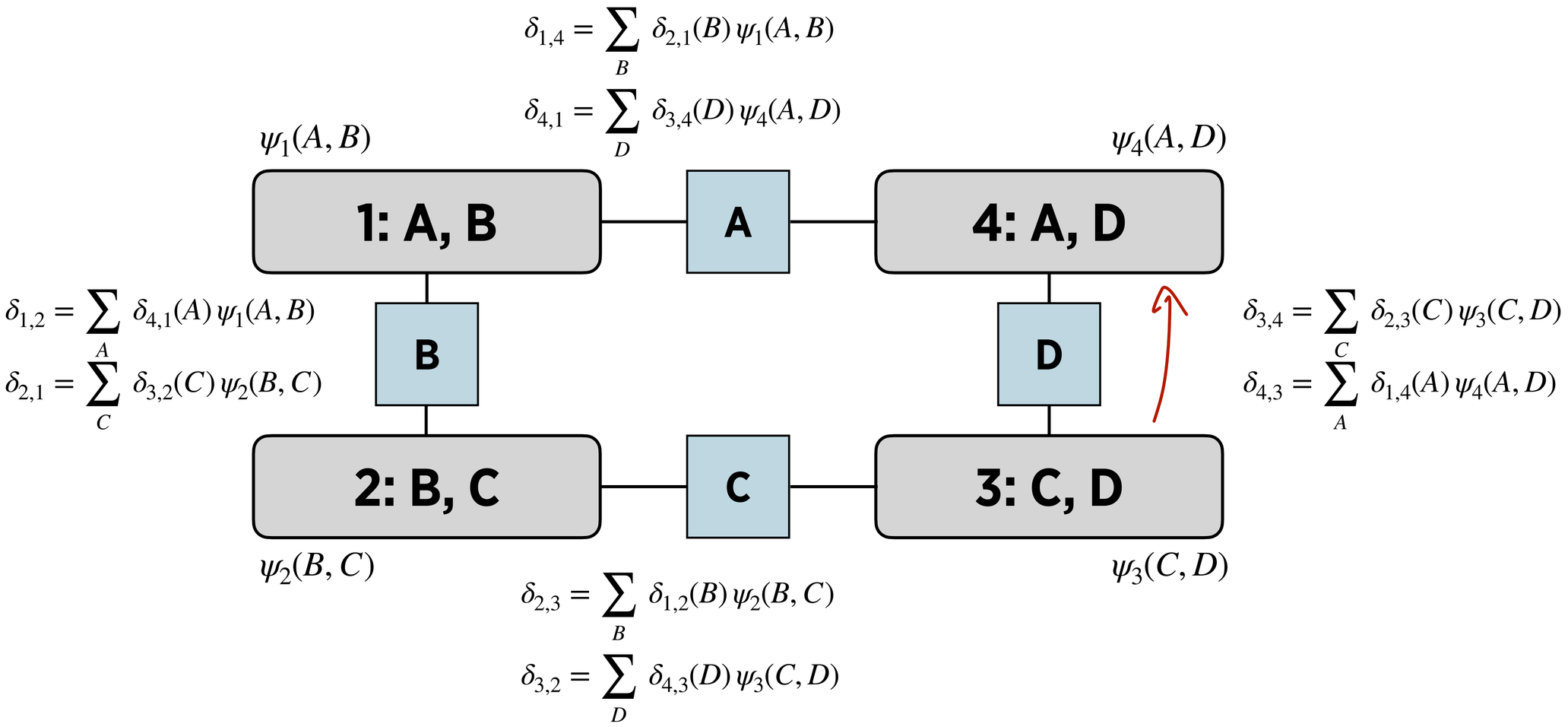
$$\delta_{2,1} = 1$$

from · to

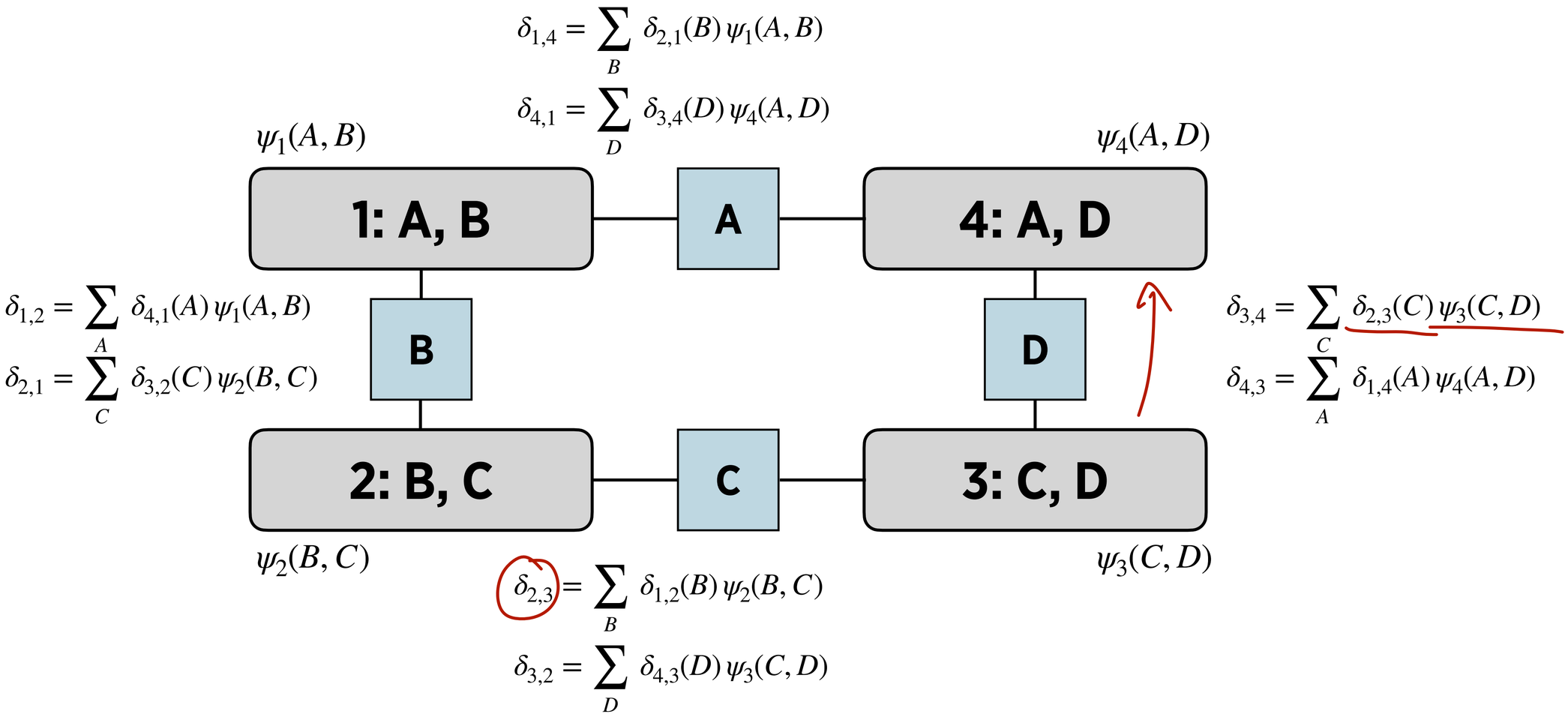
Passing messages



Passing messages



Passing messages



Cluster Graphs

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 $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$
    - Separation set: variables on both sides are separated by variables in $\boldsymbol{S}$

Cluster Graphs

- Undirected graph such that:
  - Nodes are clusters $\boldsymbol{C}_i \subseteq \{X_1, \ldots, X_n\}$ subset of variables
  - Edge between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ associated with sepset $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$
    - Separation set: variables on both sides are separated by variables in $\boldsymbol{S}$

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 $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$
    - Separation set: variables on both sides are separated by variables in $\boldsymbol{S}$

subset of variables

joint variables that they talk about

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 $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$
    - Separation set: variables on both sides are separated by variables in $\boldsymbol{S}$
- Given set of factors Φ, we assign each ϕ_k to a cluster $\boldsymbol{C}_{\alpha(k)}$ s.t. scope$[\phi_k] \subseteq \boldsymbol{C}_{\alpha(k)}$

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 $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$
    - Separation set: variables on both sides are separated by variables in $\boldsymbol{S}$
- 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 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 $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$
    - Separation set: variables on both sides are separated by variables in $\boldsymbol{S}$
- Given set of factors Φ, we assign each ϕ_k to a cluster $\boldsymbol{C}_{\alpha(k)}$ s.t. $\text{scope}[\phi_k] \subseteq \boldsymbol{C}_{\alpha(k)}$
- Define initial beliefs as $\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$

subset of variables

joint variables that they talk about

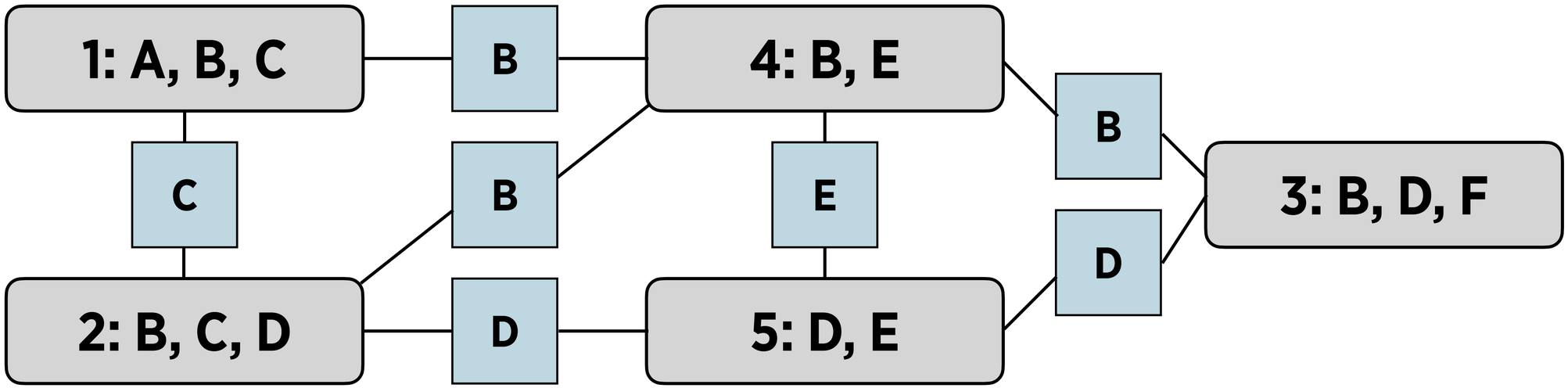
assign to only one cluster

Cluster Graphs

- Undirected graph such that:
  - Nodes are clusters $\boldsymbol{C}_i \subseteq \{X_1, \ldots, X_n\}$ — subset of variables
  - Edge between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ associated with sepset $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$ — joint variables that they talk about
    - Separation set: variables on both sides are separated by variables in $\boldsymbol{S}$
- Given set of factors Φ, we assign each ϕ_k to a cluster $\boldsymbol{C}_{\alpha(k)}$ s.t. $\text{scope}[\phi_k] \subseteq \boldsymbol{C}_{\alpha(k)}$ — assign to only one cluster
- Define initial beliefs as $\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$ — all factors assigned to that cluster

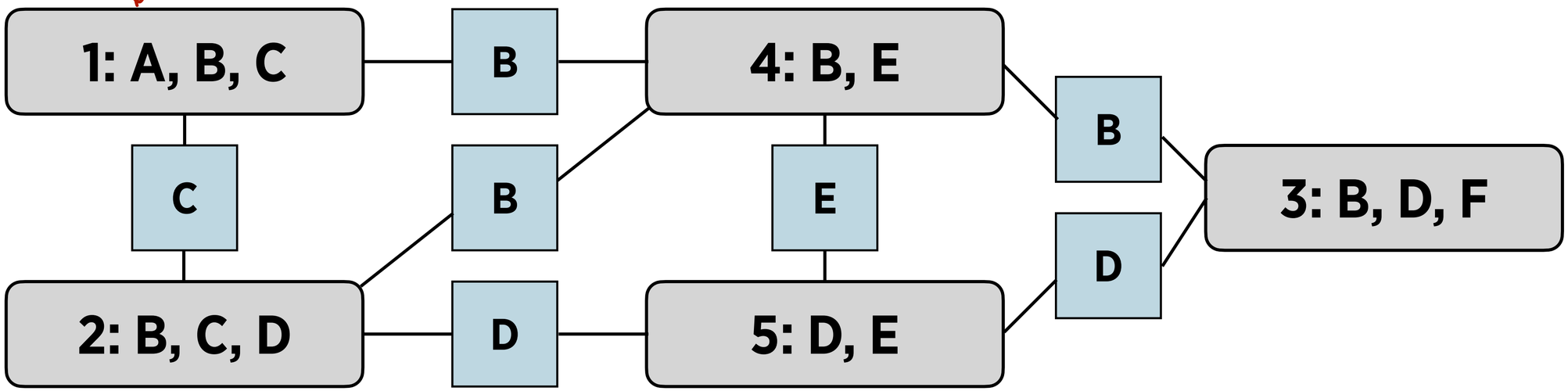
Example cluster graph

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



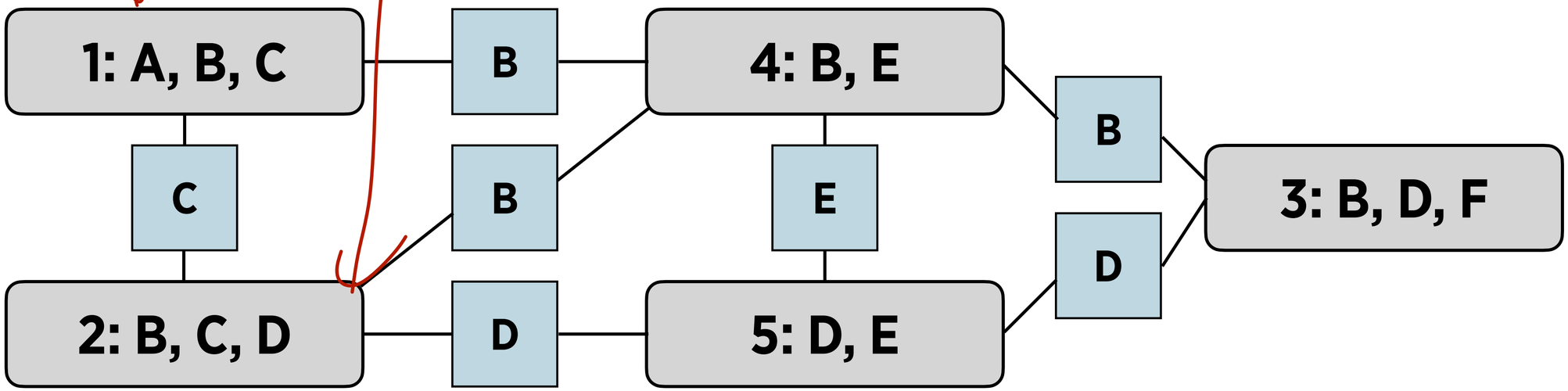
Example cluster graph

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



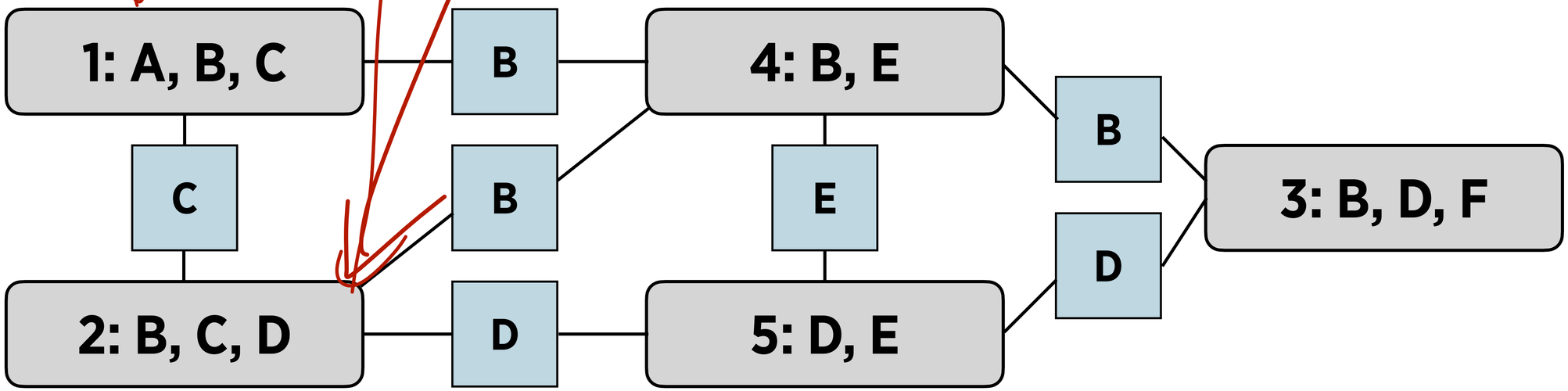
Example cluster graph

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



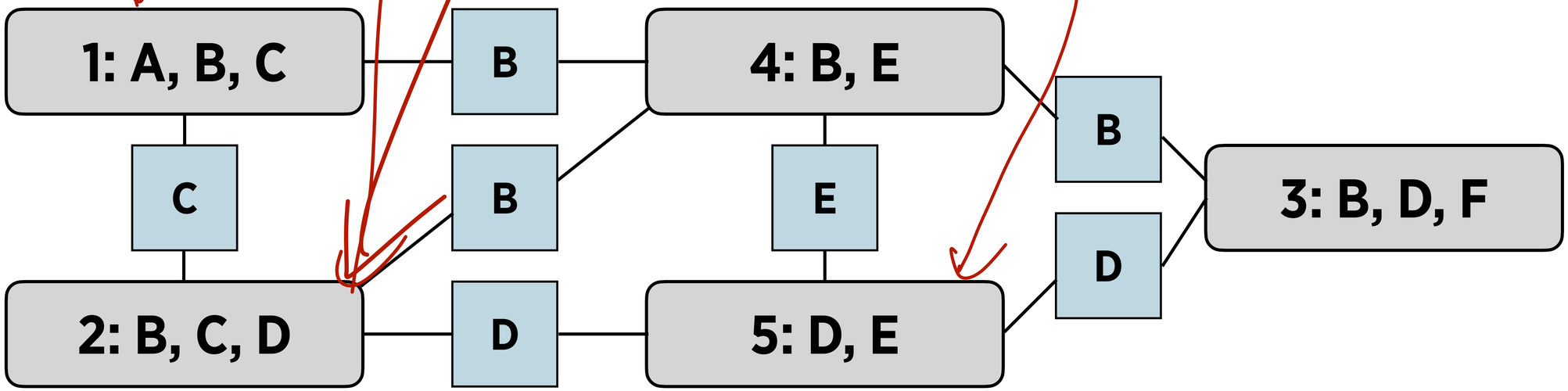
Example cluster graph

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



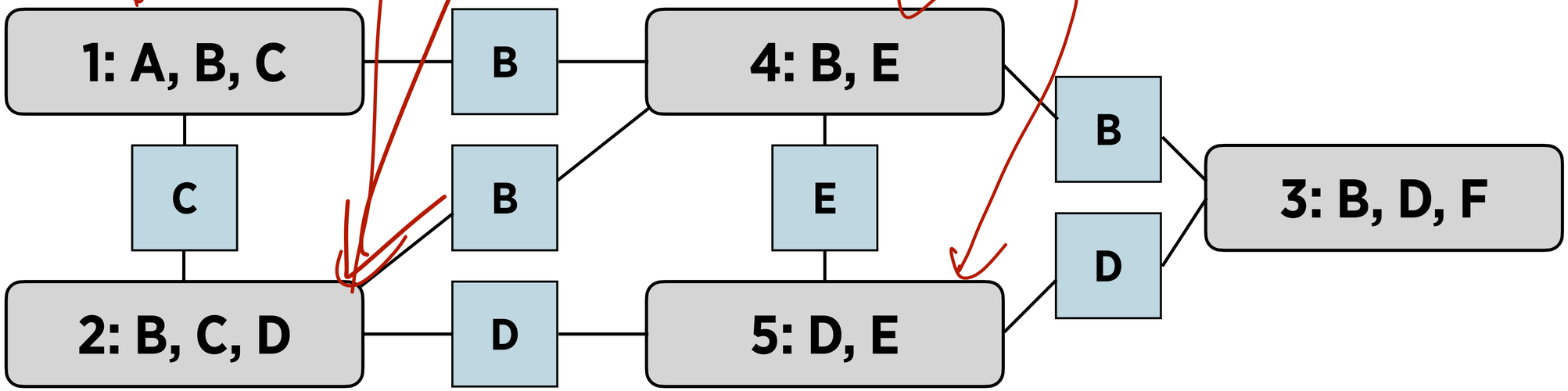
Example cluster graph

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



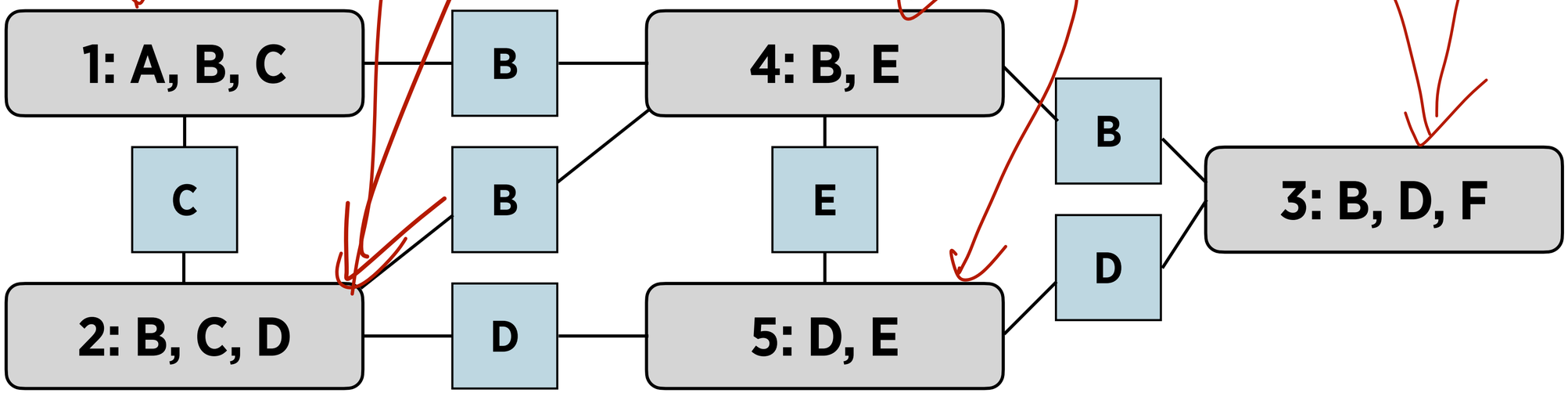
Example cluster graph

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



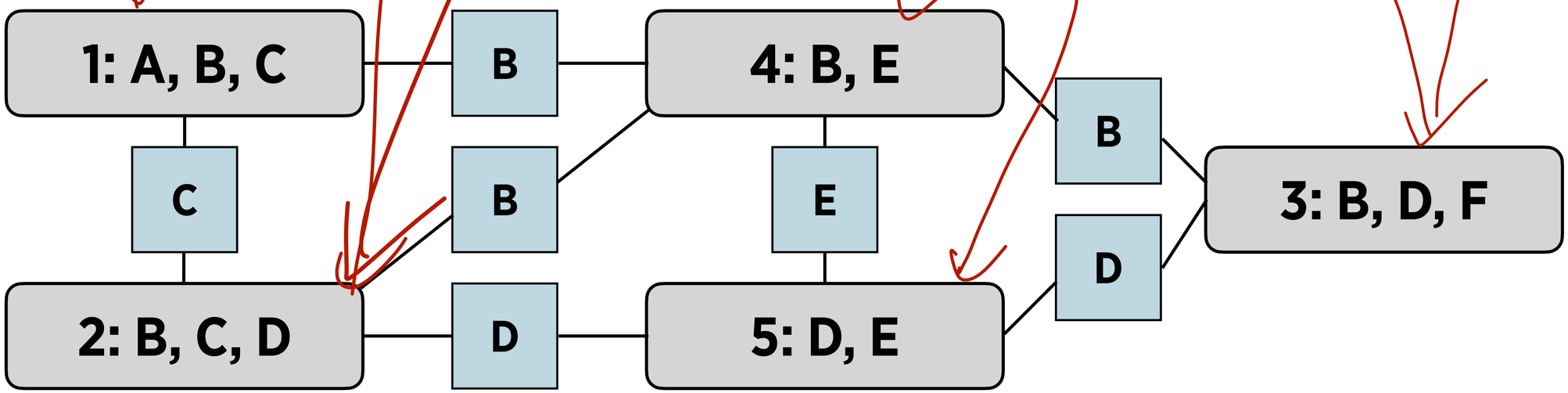
Example cluster graph

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



Example cluster graph

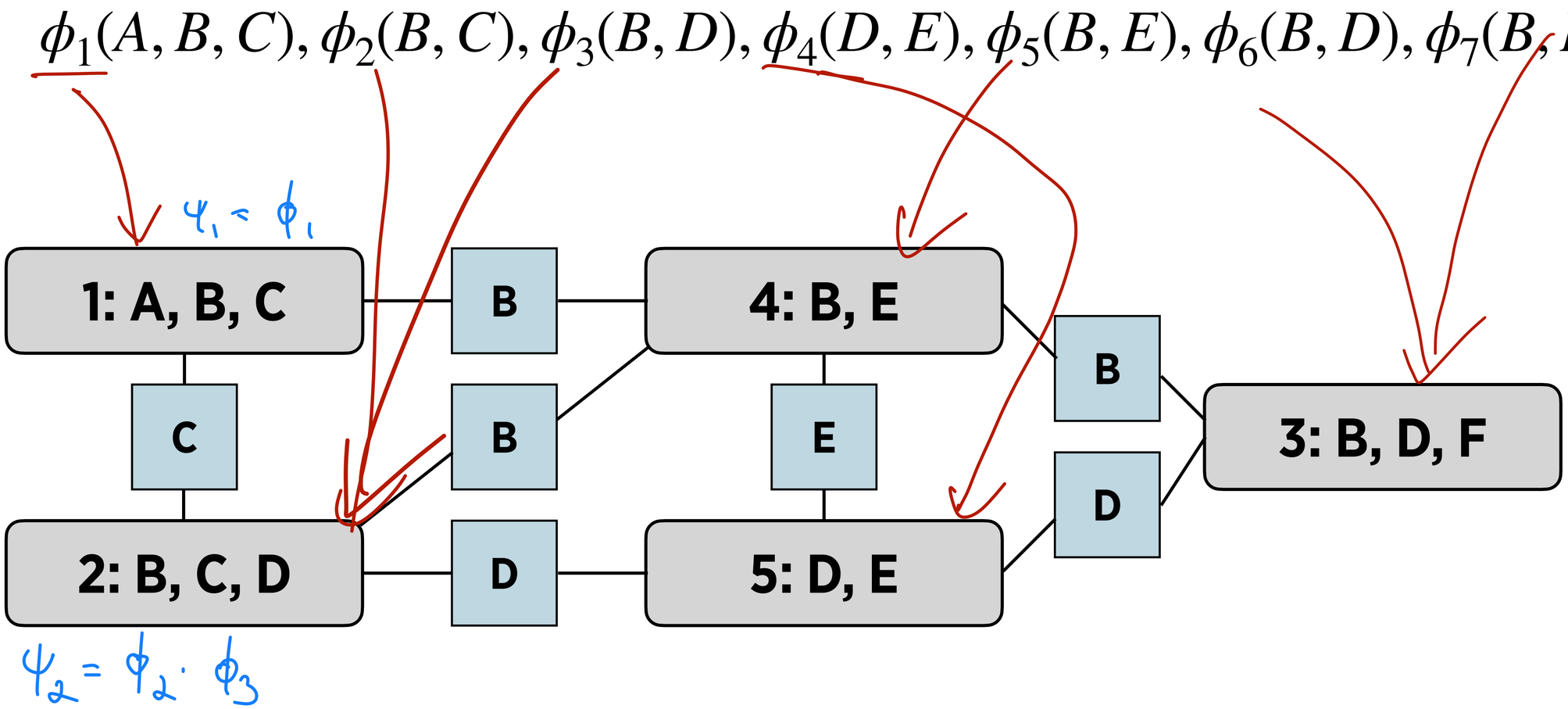
$$\phi_1(A,B,C), \phi_2(B,C), \phi_3(B,D), \phi_4(D,E), \phi_5(B,E), \phi_6(B,D), \phi_7(B,D,F)$$



$\psi_2 = \phi_2 \cdot \phi_3$

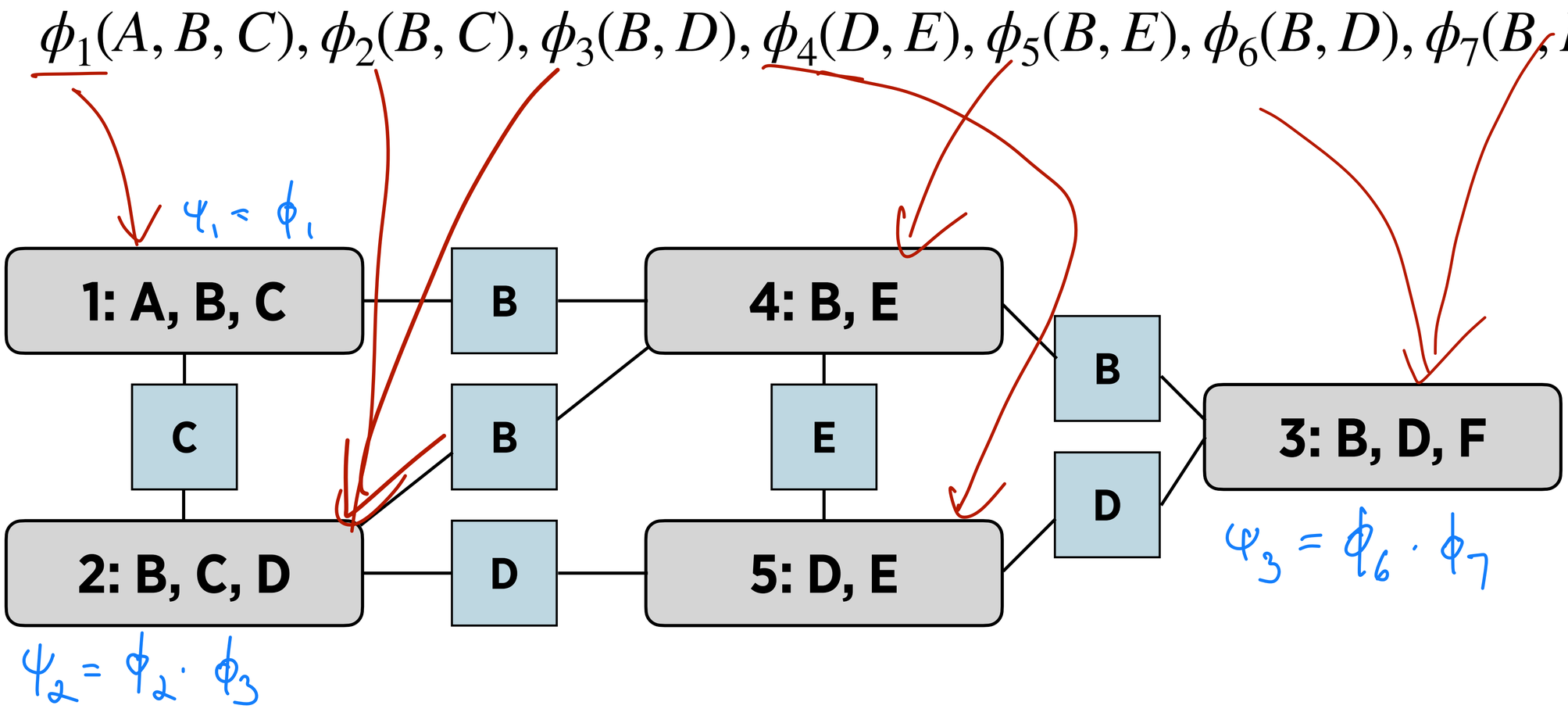
Example cluster graph

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



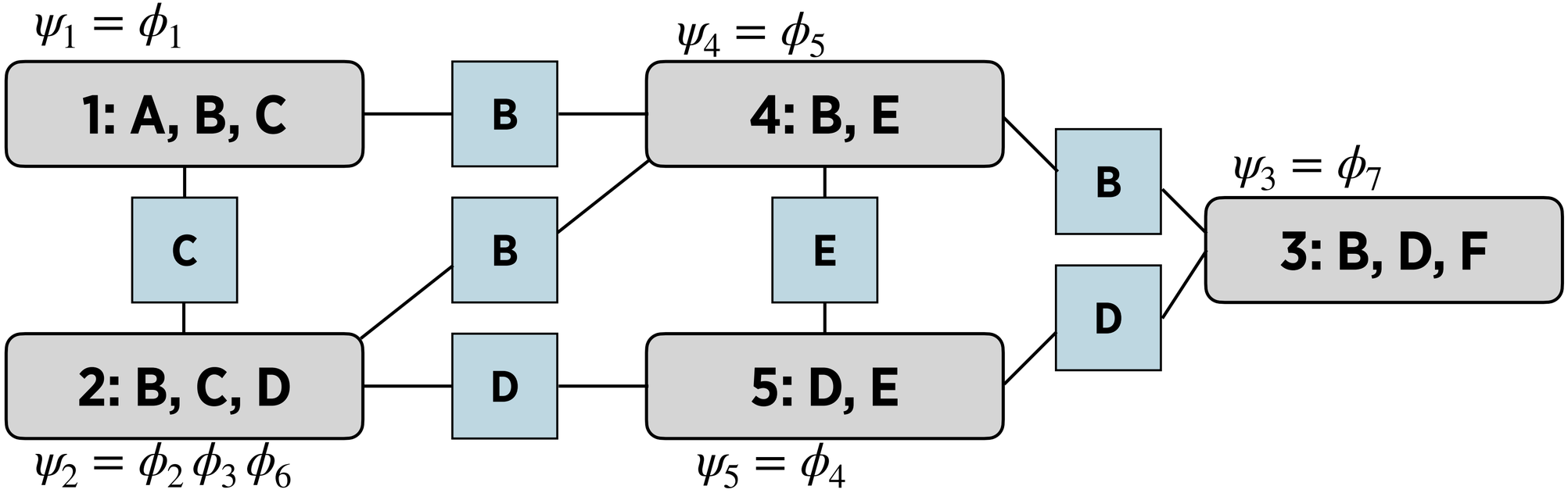
Example cluster graph

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



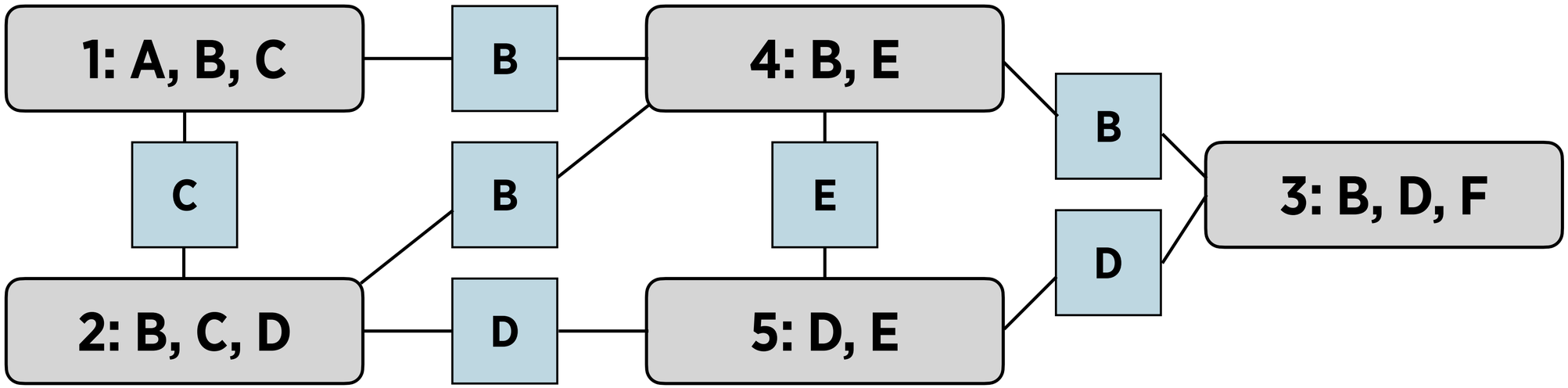
Example cluster graph

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



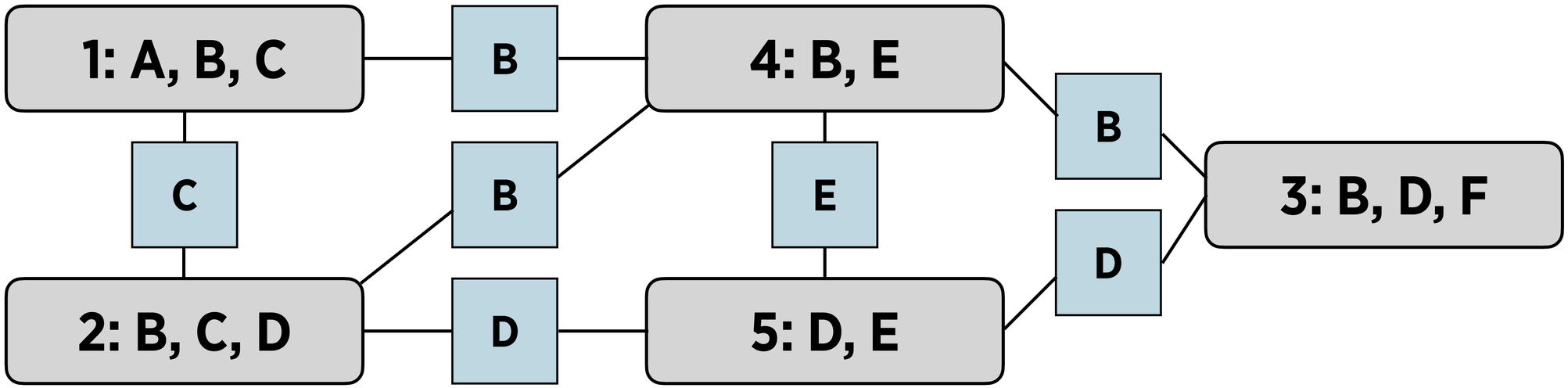
Example cluster graph

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



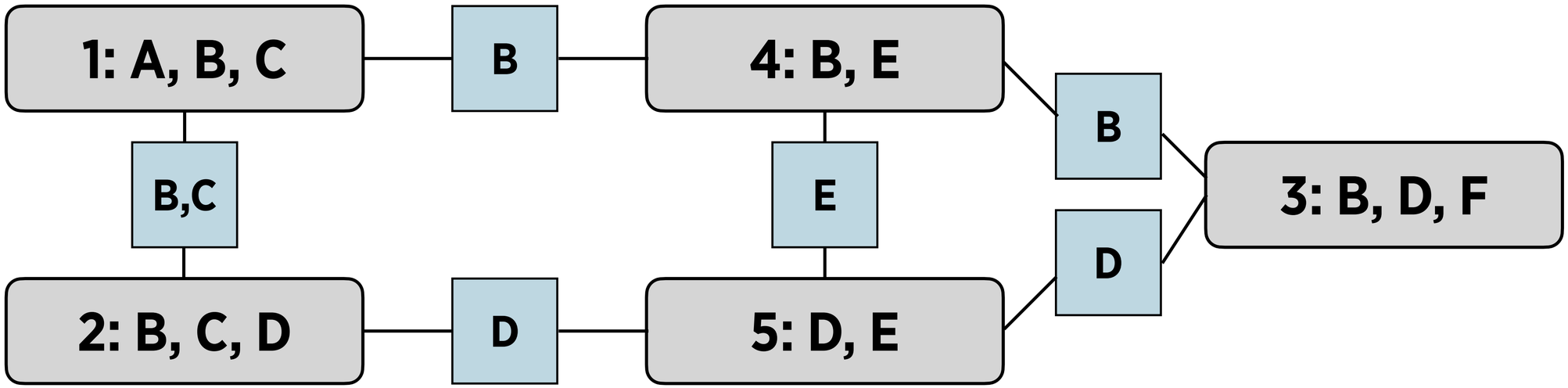
Example cluster graph

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

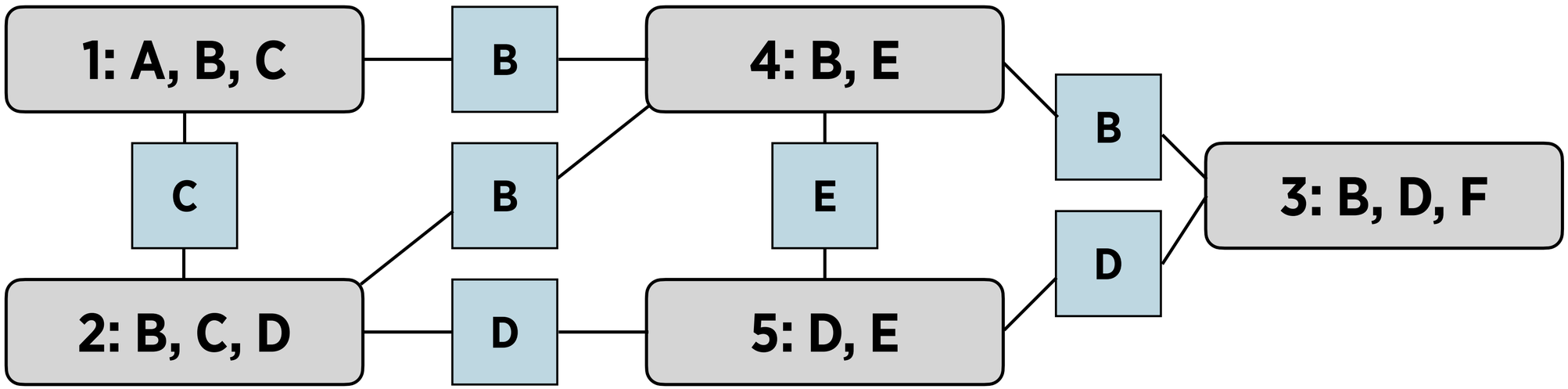


Different cluster graph

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

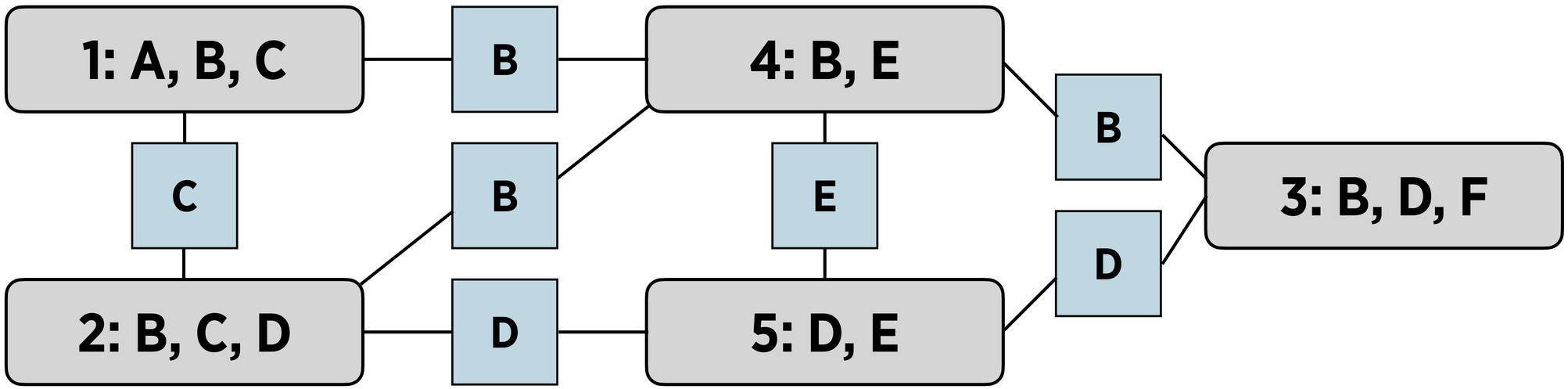


Message passing



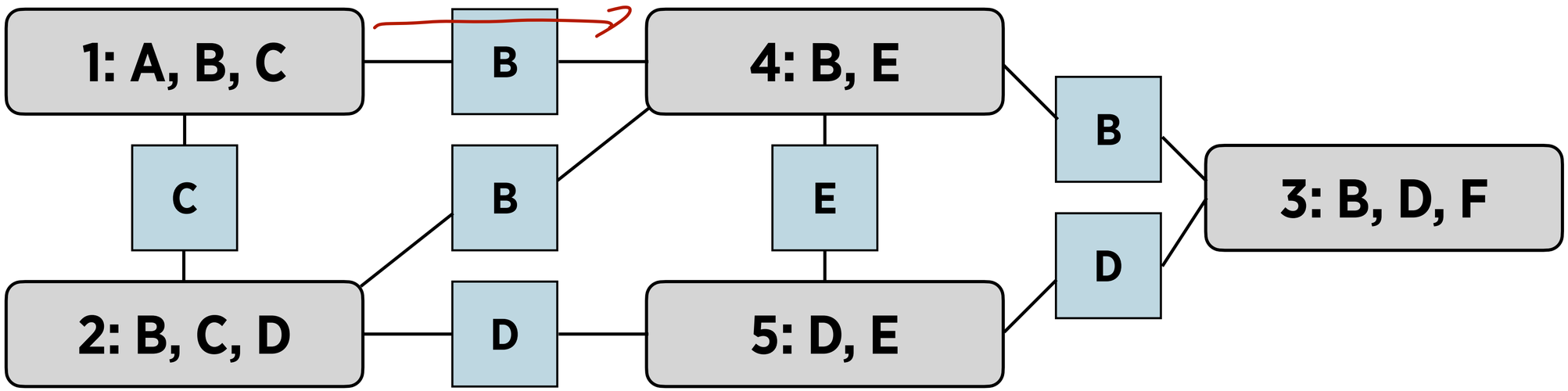
Message passing

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



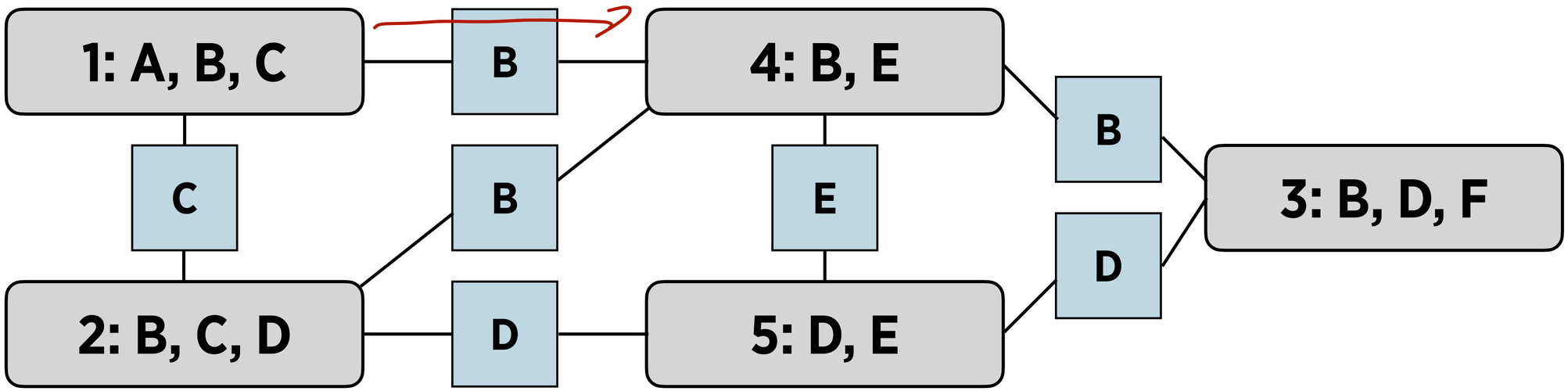
Message passing

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



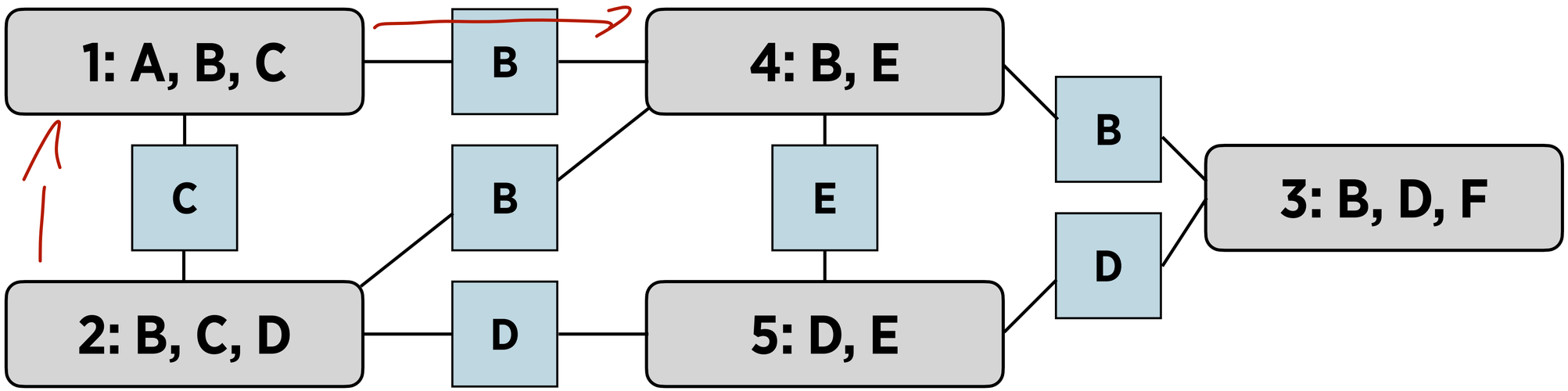
Message passing

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



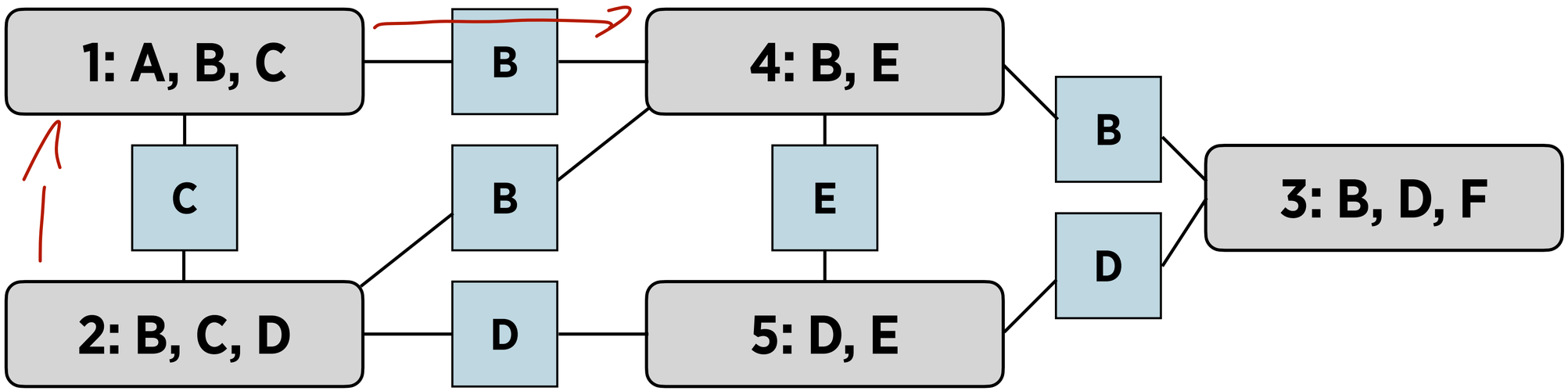
Message passing

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



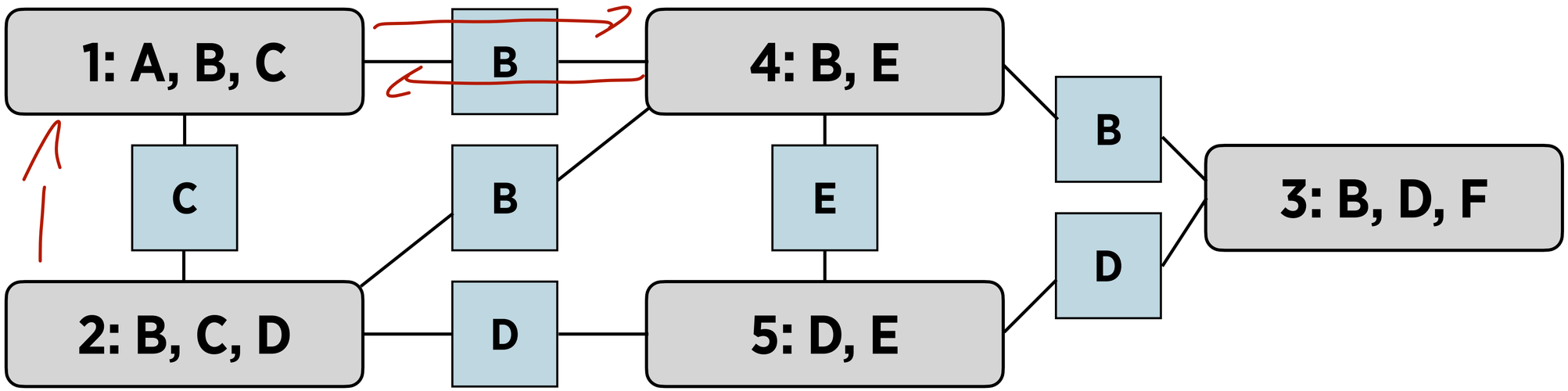
Message passing

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



Message passing

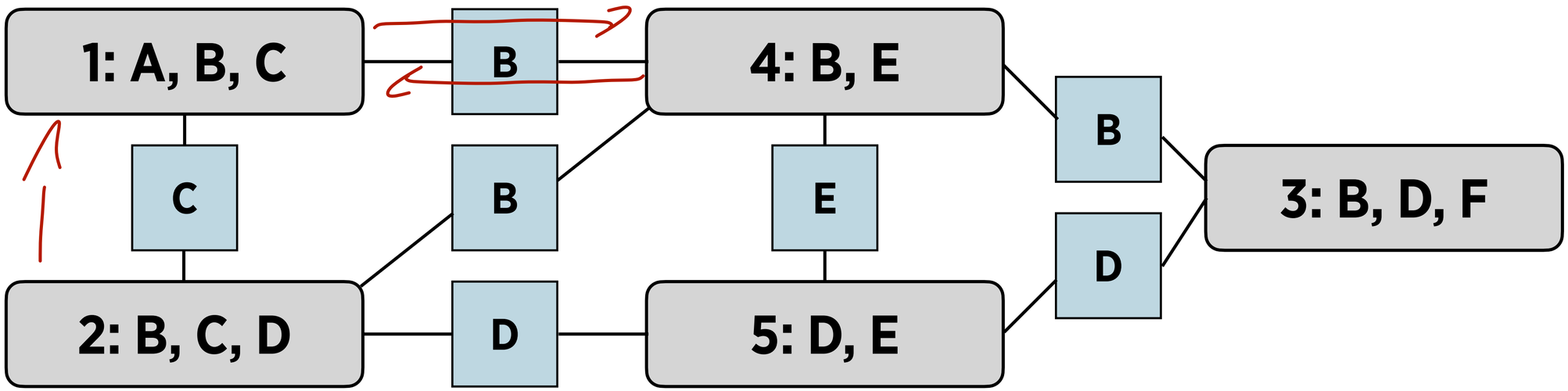
$$\delta_{1\to 4}(B) = \sum_{A,C} \psi_1(A, B, C)\, \delta_{2\to 1}(C)$$



Message passing

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

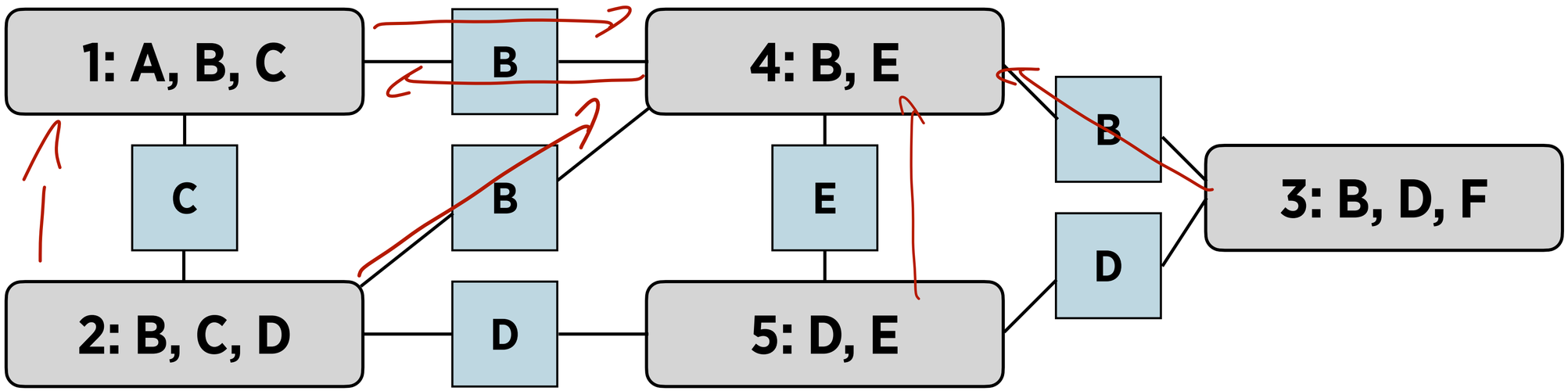
$$\delta_{4\to 1}(B) = \sum_{E} \psi_4(B,E)\,\delta_{2\to 4}(B)\,\delta_{5\to 4}(E)\,\delta_{3\to 4}(B)$$



Message passing

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

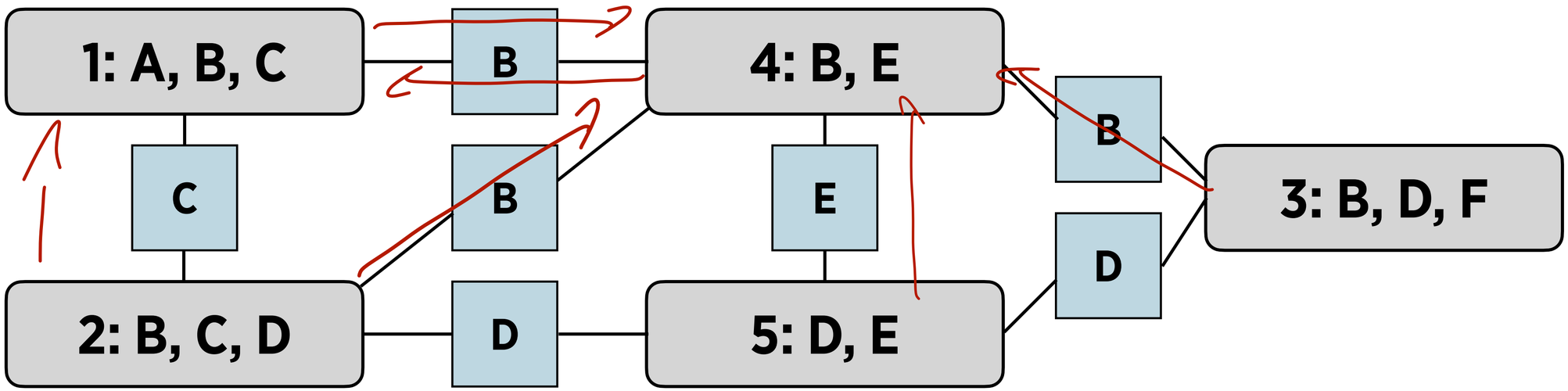
$$\delta_{4\to 1}(B) = \sum_{E} \psi_4(B,E)\,\delta_{2\to 4}(B)\,\delta_{5\to 4}(E)\,\delta_{3\to 4}(B)$$



Message passing

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

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

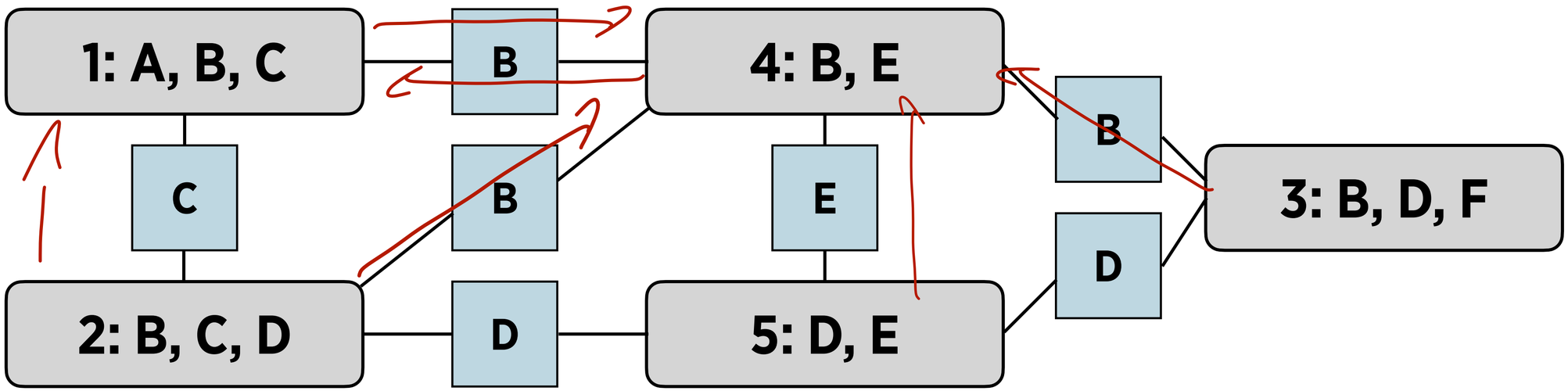


$$\delta_{i\to j}(\boldsymbol{S}_{i,j}) = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \prod_{k\in(\mathcal{N}_i - \{j\})} \delta_{k\to i}$$

Message passing

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

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



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

Belief Propagation Algorithm

Belief Propagation Algorithm

- Assign each factor $\phi_k \in \Phi$ to a cluster $\boldsymbol{C}_{\alpha(k)}$

Belief Propagation Algorithm

- Assign each factor $\phi_k \in \Phi$ to a cluster $\boldsymbol{C}_{\alpha(k)}$
- Construct initial potentials $\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$

Belief Propagation Algorithm

- Assign each factor $\phi_k \in \Phi$ to a cluster $\boldsymbol{C}_{\alpha(k)}$
- Construct initial potentials $\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$
- Initialise all messages to be 1

Belief Propagation Algorithm

- Assign each factor $\phi_k \in \Phi$ to a cluster $\boldsymbol{C}_{\alpha(k)}$
- Construct initial potentials $\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$
- Initialise all messages to be 1
- Repeat

Belief Propagation Algorithm

- Assign each factor $\phi_k \in \Phi$ to a cluster $\boldsymbol{C}_{\alpha(k)}$
- Construct initial potentials $\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$
- Initialise all messages to be 1
- Repeat
  - Select edge (i, j) and pass message

Belief Propagation Algorithm

- Assign each factor $\phi_k \in \Phi$ to a cluster $\boldsymbol{C}_{\alpha(k)}$
- Construct initial potentials $\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$
- Initialise all messages to be 1
- Repeat
  - Select edge (i, j) and pass message

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

Belief Propagation Algorithm

- Assign each factor $\phi_k \in \Phi$ to a cluster $\boldsymbol{C}_{\alpha(k)}$
- Construct initial potentials $\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$
- Initialise all messages to be 1
- Repeat
  - Select edge (i, j) and pass message

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

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

Belief Propagation Algorithm

- Assign each factor $\phi_k \in \Phi$ to a cluster $\boldsymbol{C}_{\alpha(k)}$
- Construct initial potentials $\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$
- Initialise all messages to be 1
- Repeat
  - Select edge (i, j) and pass message

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

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

beliefs — all neighbours

Belief Propagation Algorithm

- Assign each factor $\phi_k \in \Phi$ to a cluster $\boldsymbol{C}_{\alpha(k)}$
- Construct initial potentials $\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$
- Initialise all messages to be 1
- Repeat until when ?
  - Select edge (i, j) and pass message

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

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

beliefs — all neighbours

Belief Propagation Algorithm

- Assign each factor $\phi_k \in \Phi$ to a cluster $\boldsymbol{C}_{\alpha(k)}$
- Construct initial potentials $\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$
- Initialise all messages to be 1
- Repeat until when ?
  - Select edge (i, j) and pass message

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

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

beliefs — all neighbours

Belief Propagation Algorithm

- Assign each factor $\phi_k \in \Phi$ to a cluster $\boldsymbol{C}_{\alpha(k)}$
- Construct initial potentials $\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$
- Initialise all messages to be 1
- Repeat until when ?
  - Select edge (i, j) and pass message

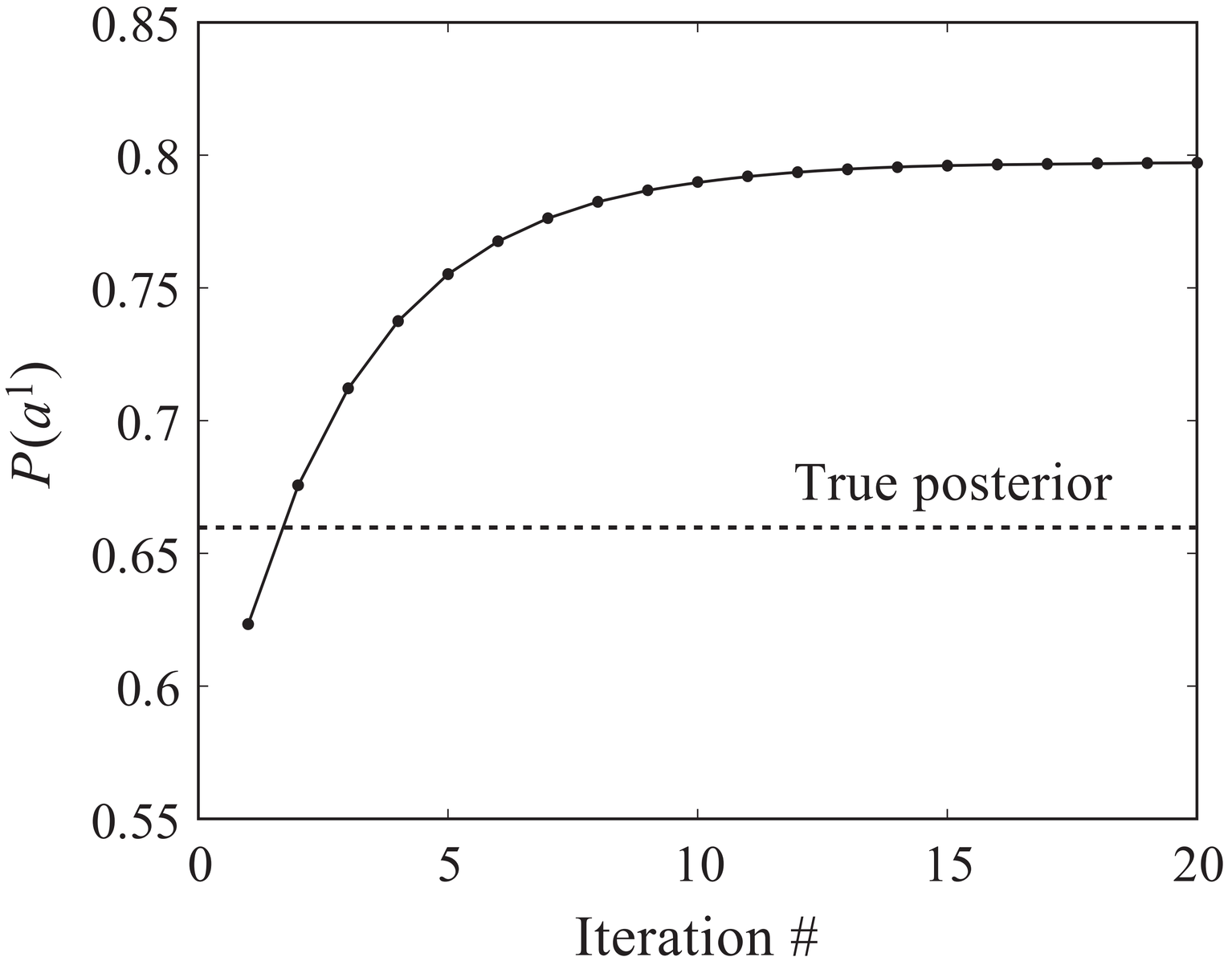
$$\delta_{i\to j}(\boldsymbol{S}_{i,j}) = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \prod_{k\in(\mathcal{N}_i - \{j\})} \delta_{k\to i}$$

e.g. round robin

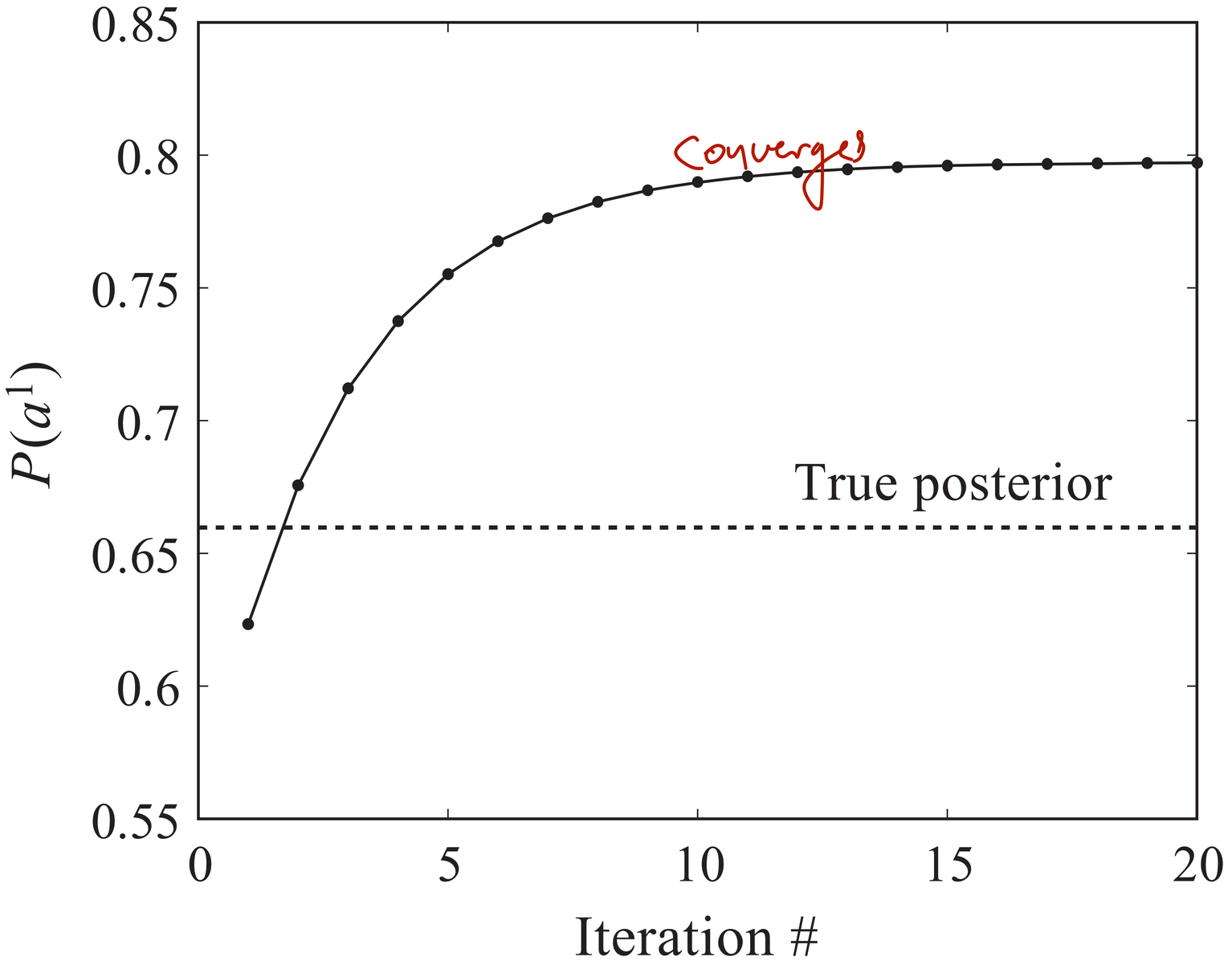
- Compute $\beta_i(\boldsymbol{C}_i) = \psi_i \prod_{k\in\mathcal{N}_i} \delta_{k\to i}$

beliefs — all neighbours

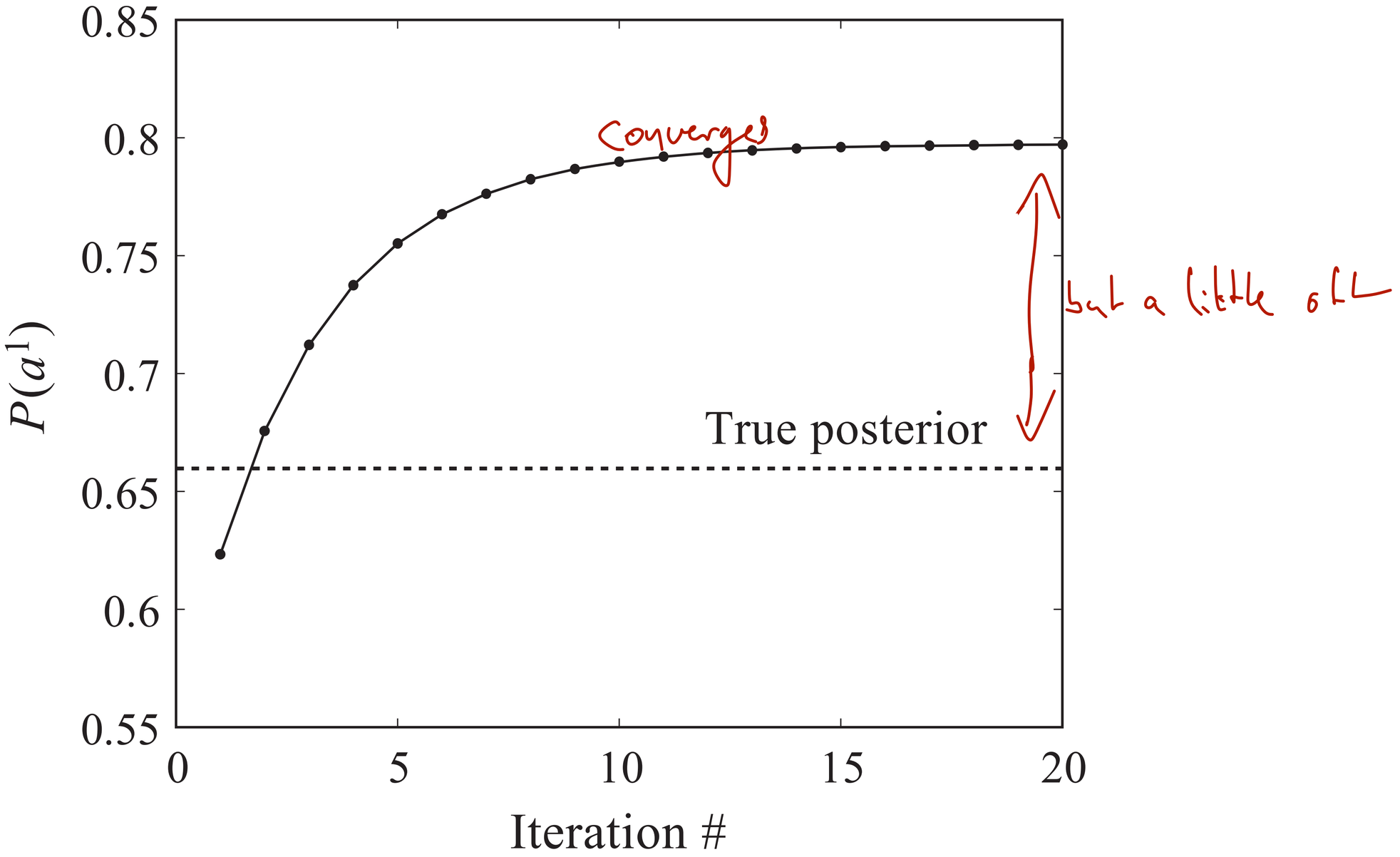
Belief propagation run



Belief propagation run



Belief propagation run



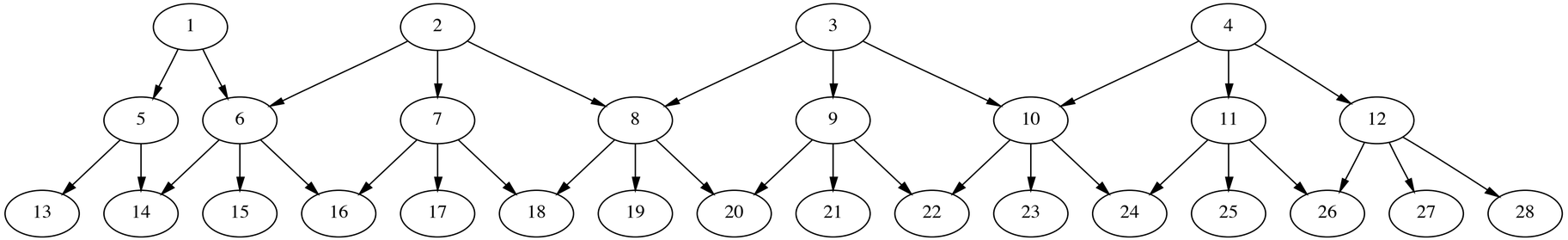
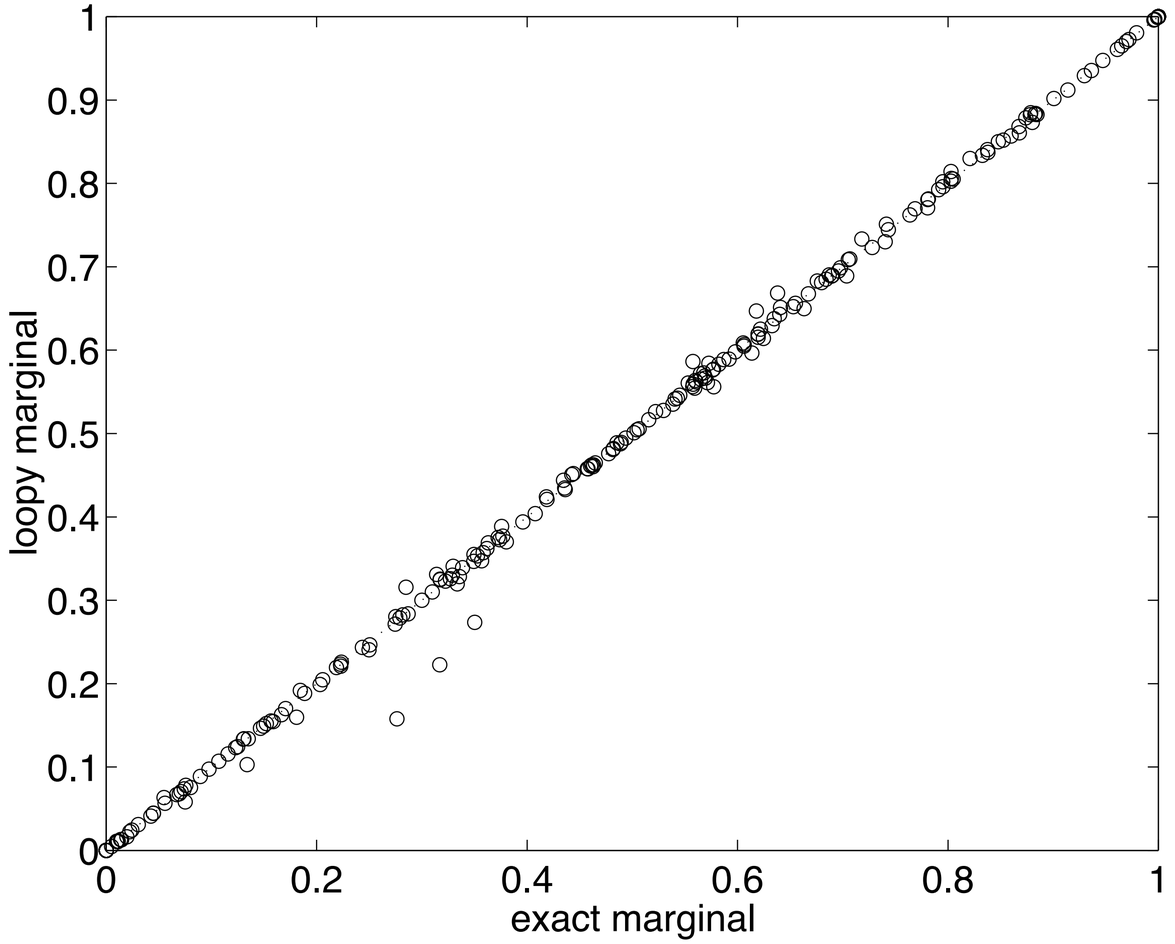


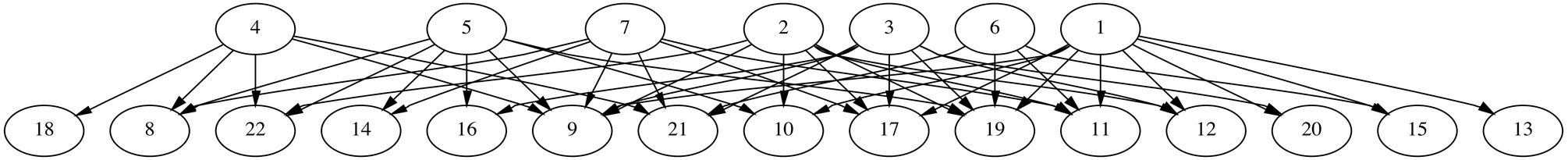
Figure 1: The structure of the PYRAMID network. All nodes are binary and observations appear only on the bottom layer. Such networks occur often in image analysis where the bottom layer would correspond to pixels.

Murphy, Weiss, Jordan, UAI'99

Loopy Belief Propagation for Approximate Inference: An Empirical Study

Kevin P. Murphy, Yair Weiss and Michael I. Jordan*
Computer Science Division
University of California
Berkeley, CA 94705
{murphyk,yweiss,jordan}@cs.berkeley.edu

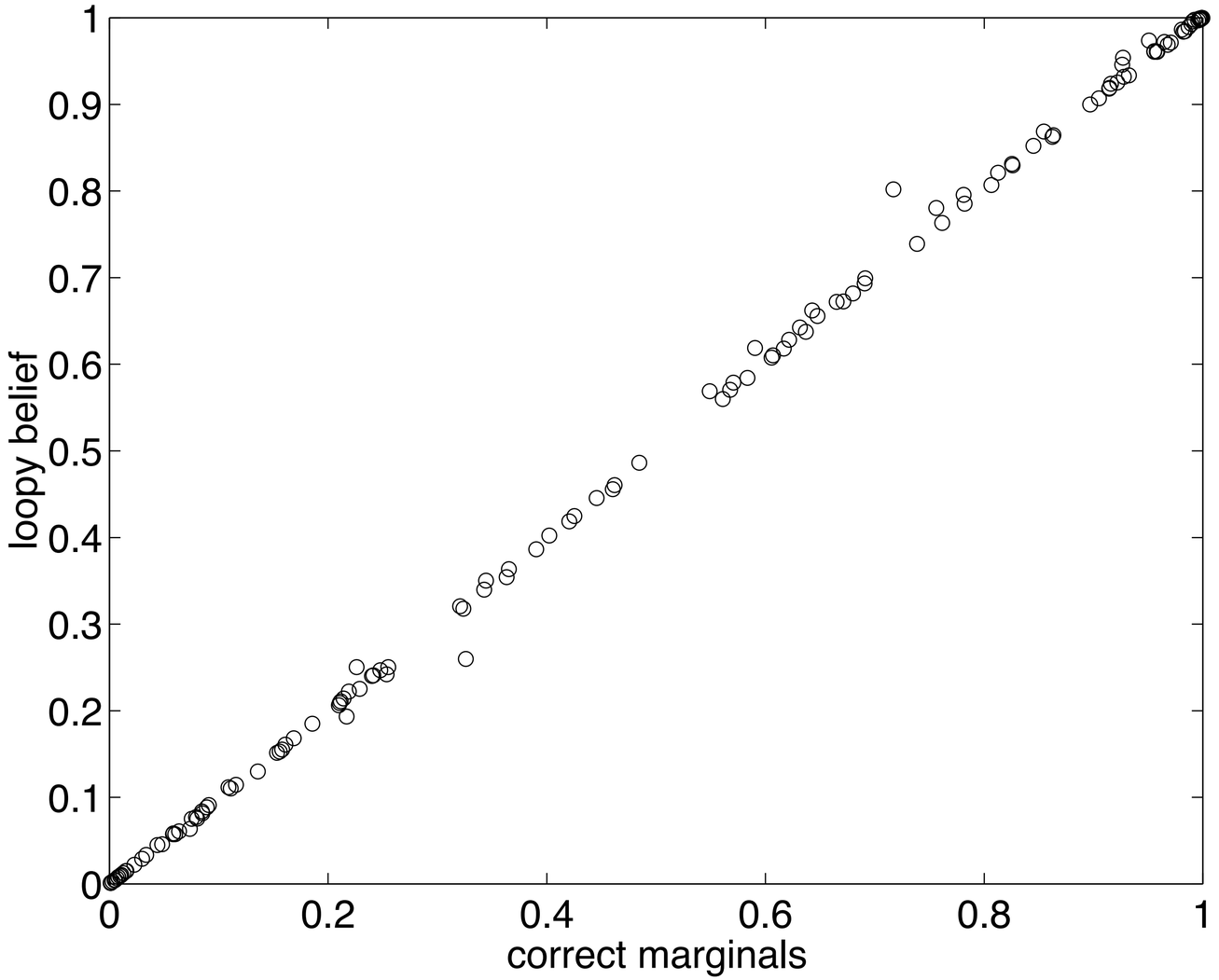




Murphy, Weiss, Jordan, UAI'99

Loopy Belief Propagation for Approximate Inference: An Empirical Study

Kevin P. Murphy, Yair Weiss and Michael I. Jordan*
Computer Science Division
University of California
Berkeley, CA 94705
{murphyk,yweiss,jordan}@cs.berkeley.edu



Summary Belief Propagation

- Graph of clusters connected by sepsets

Summary Belief Propagation

- Graph of clusters connected by sepsets
- Adjacent clusters pass information to each other about variables in sepset

Summary Belief Propagation

- Graph of clusters connected by sepsets
- Adjacent clusters pass information to each other about variables in sepset
  - Message from i to j summarises everything i knows, except information obtained from j (no double counting)

Summary Belief Propagation

- Graph of clusters connected by sepsets
- Adjacent clusters pass information to each other about variables in sepset
  - Message from i to j summarises everything i knows, except information obtained from j (no double counting)
- Algorithm may not converge

Summary Belief Propagation

- Graph of clusters connected by sepsets
- Adjacent clusters pass information to each other about variables in sepset
  - Message from i to j summarises everything i knows, except information obtained from j (no double counting)
- Algorithm may not converge
- The resulting beliefs are pseudo-marginals (not necessarily exact)

Summary Belief Propagation

- Graph of clusters connected by sepsets
- Adjacent clusters pass information to each other about variables in sepset
  - Message from i to j summarises everything i knows, except information obtained from j (no double counting)
- Algorithm may not converge
- The resulting beliefs are pseudo-marginals (not necessarily exact)
- Nevertheless, very useful in practice

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 $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$
    - Separation set: variables on both sides are separated by variables in $\boldsymbol{S}$

Family Preservation

- Given set of factors Φ, we assign each ϕ_k to a cluster $\boldsymbol{C}_{\alpha(k)}$ s.t. scope$[\phi_k] \subseteq \boldsymbol{C}_{\alpha(k)}$

Family Preservation

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

Family Preservation

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

Family Preservation

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

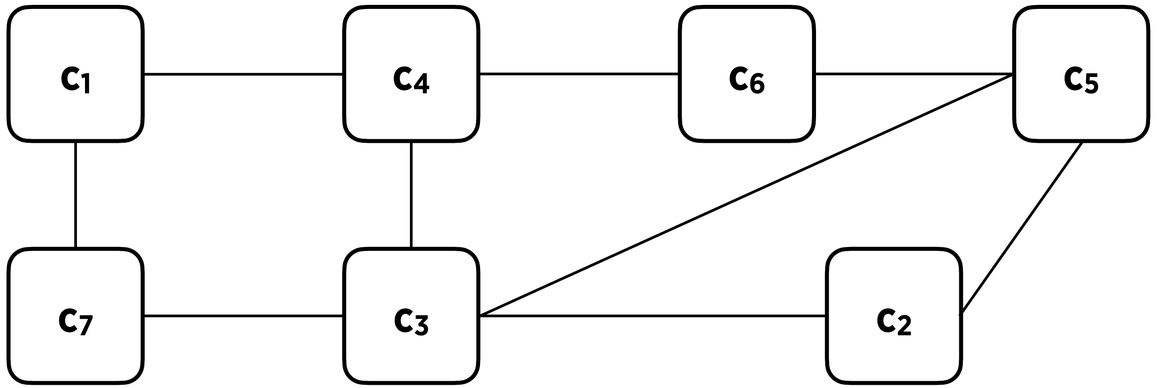
Family Preservation

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

can accommodate

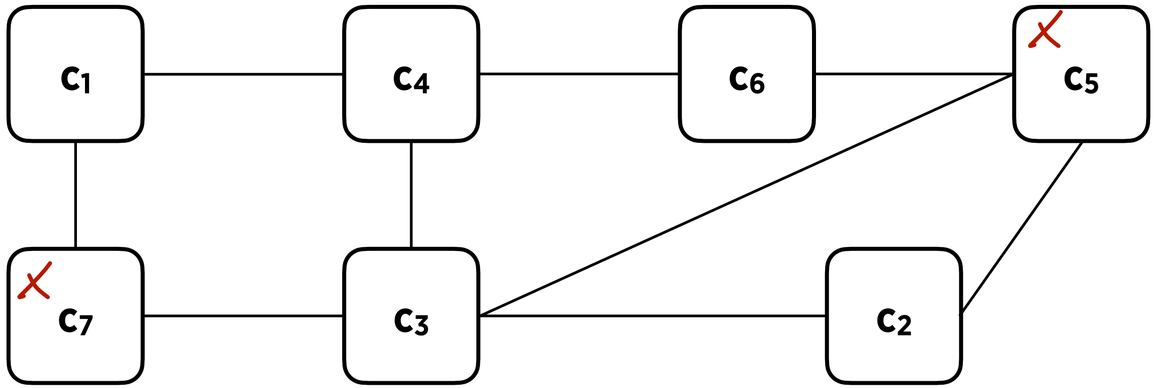
Running Intersection Property

- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$, and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$
  there exists a unique path
  between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain X



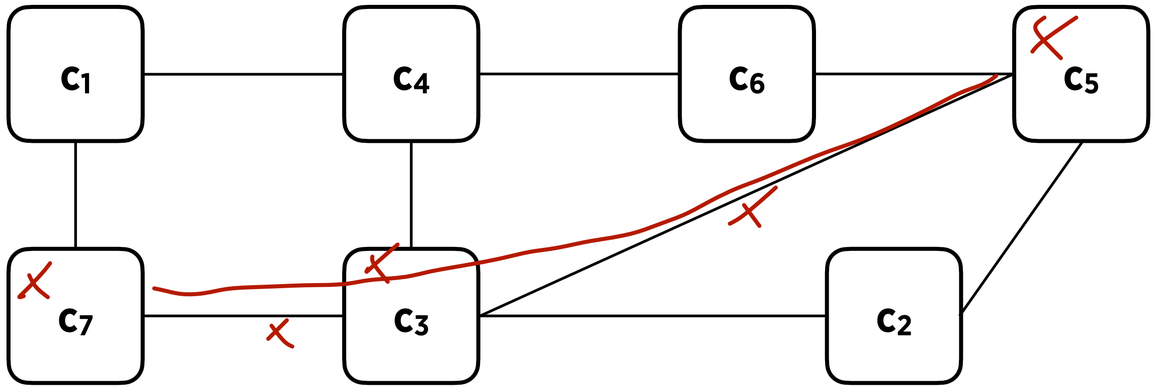
Running Intersection Property

- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$, and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$
  there exists a unique path
  between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain X



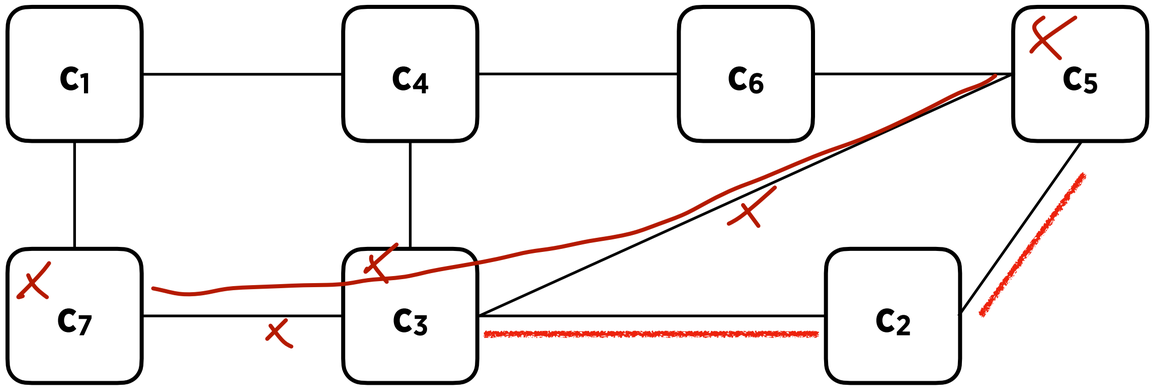
Running Intersection Property

- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$, and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$
  there exists a unique path
  between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain X



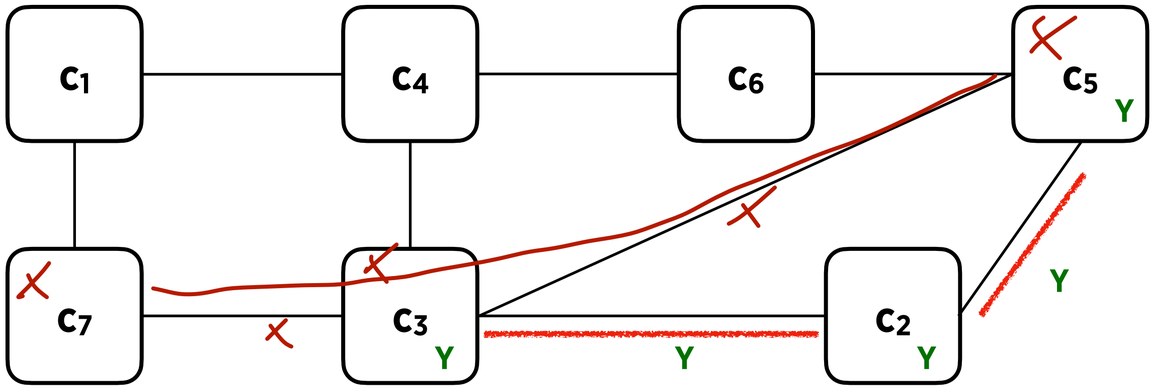
Running Intersection Property

- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$, and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$
  there exists a unique path
  between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain X



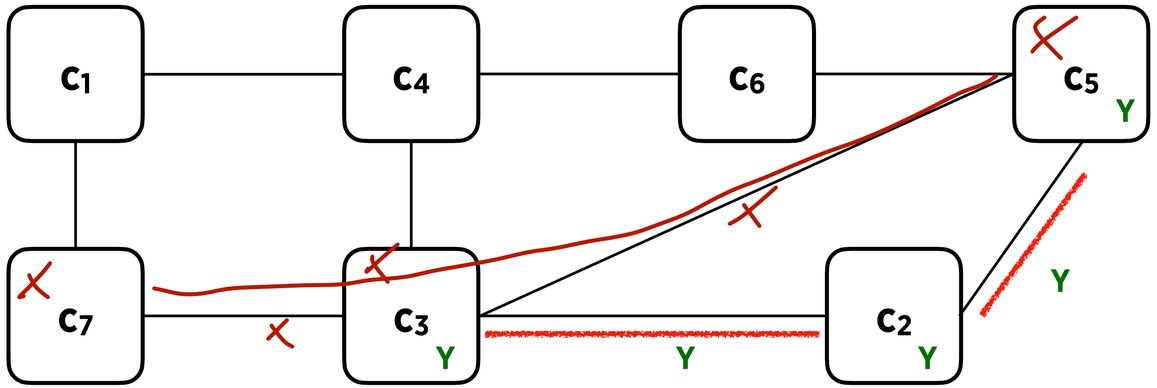
Running Intersection Property

- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$, and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$
  there exists a unique path
  between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain X



Running Intersection Property

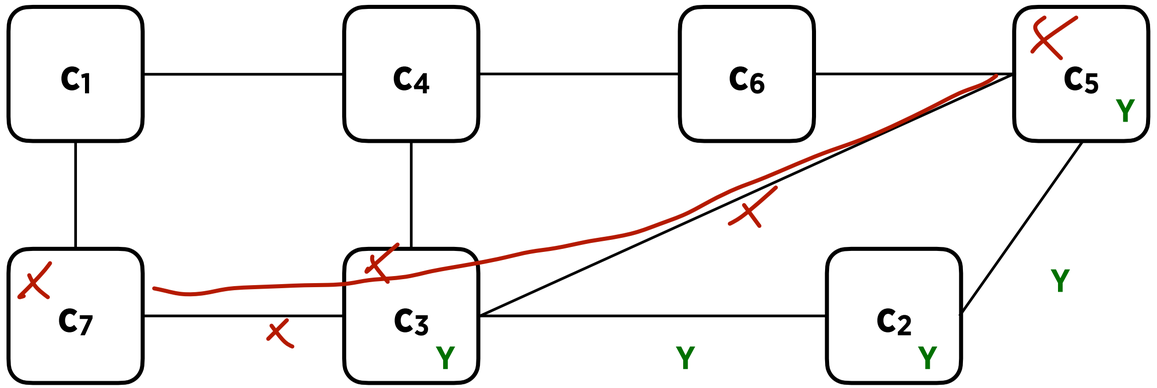
- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$, and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$
  there exists a unique path
  between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain X



BP does poorly when we have strong correlation

Running Intersection Property

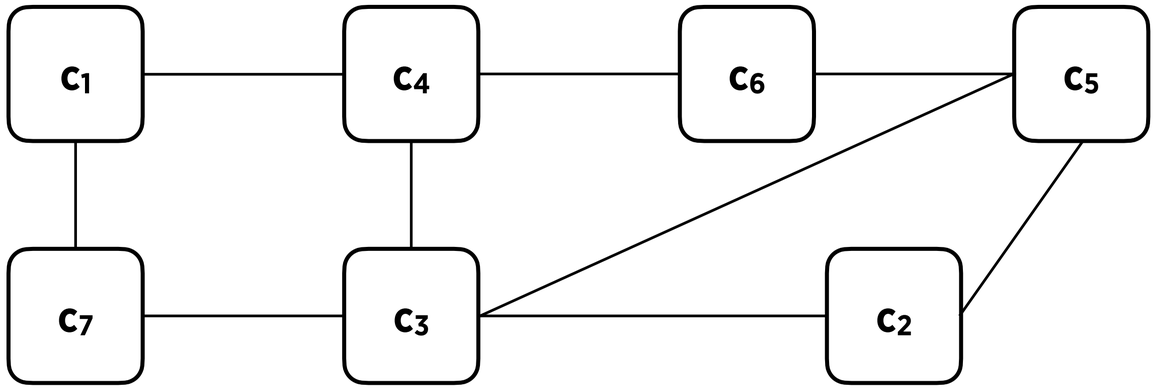
- For each pair of clusters C_i, C_j, and variable $X \in C_i \cap C_j$
  there exists a unique path
  between C_i and C_j for which all clusters and sepsets contain X



BP does poorly when we have strong correlation

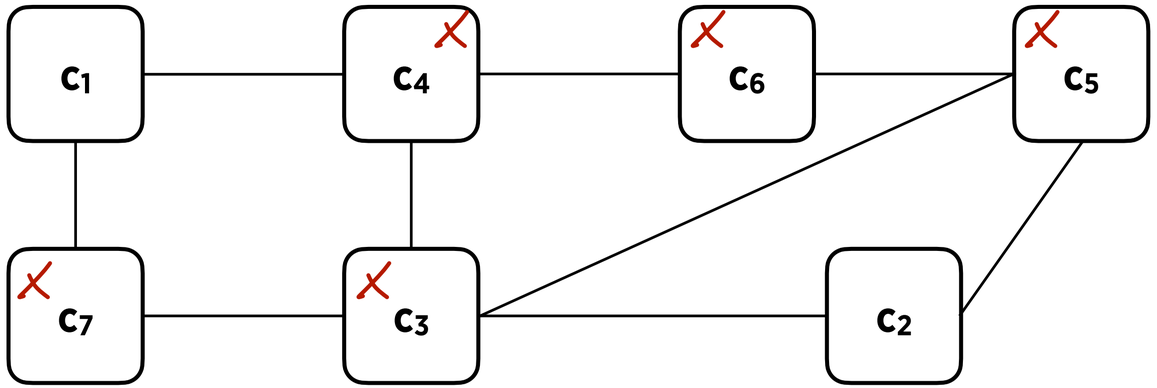
Running Intersection Property

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



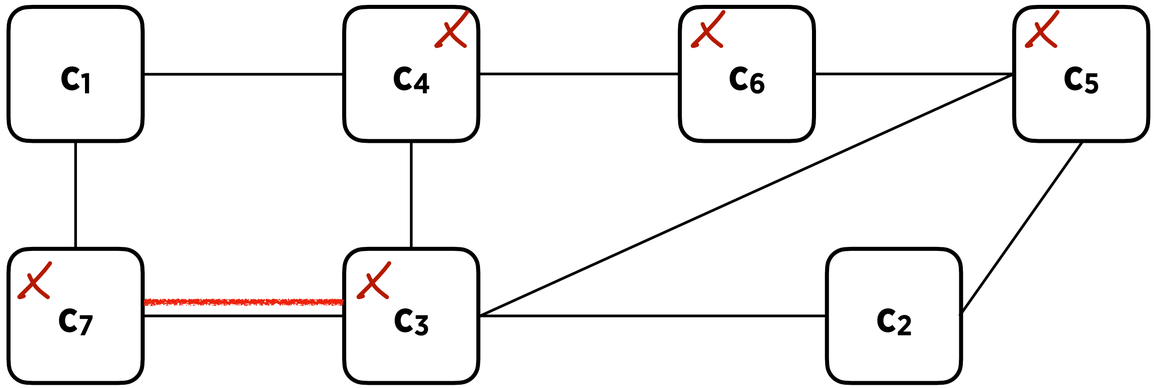
Running Intersection Property

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



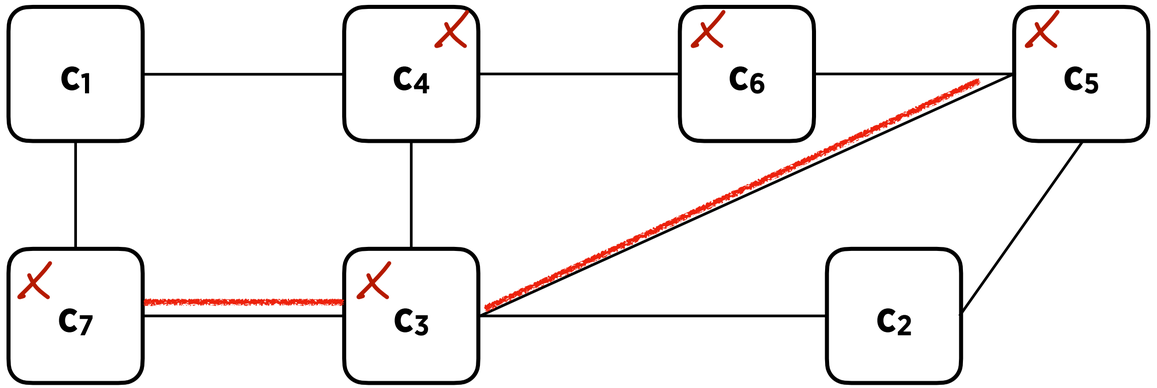
Running Intersection Property

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



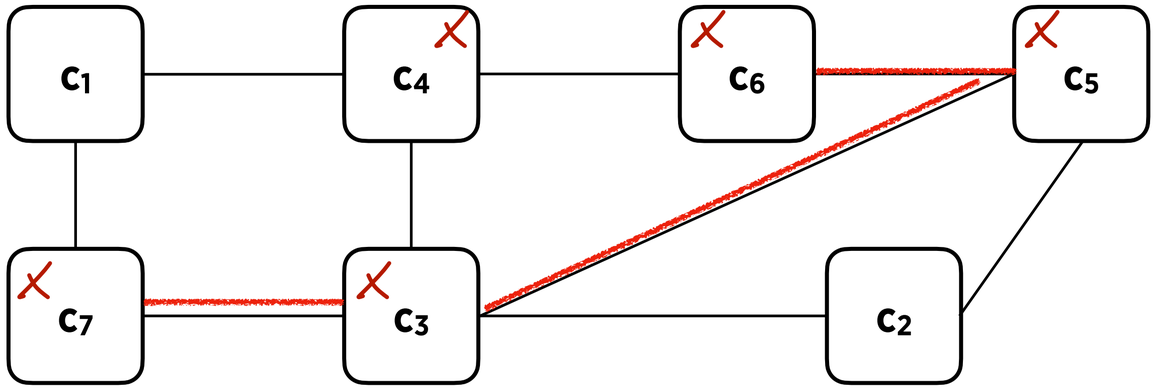
Running Intersection Property

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



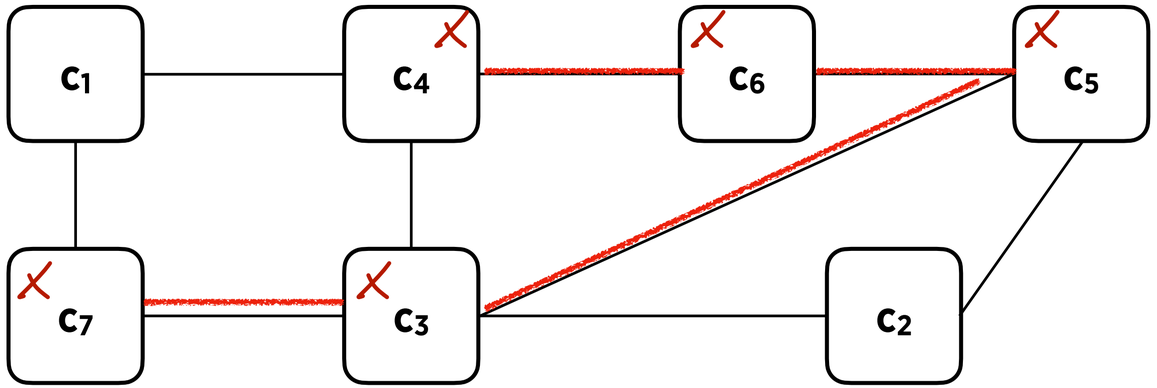
Running Intersection Property

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



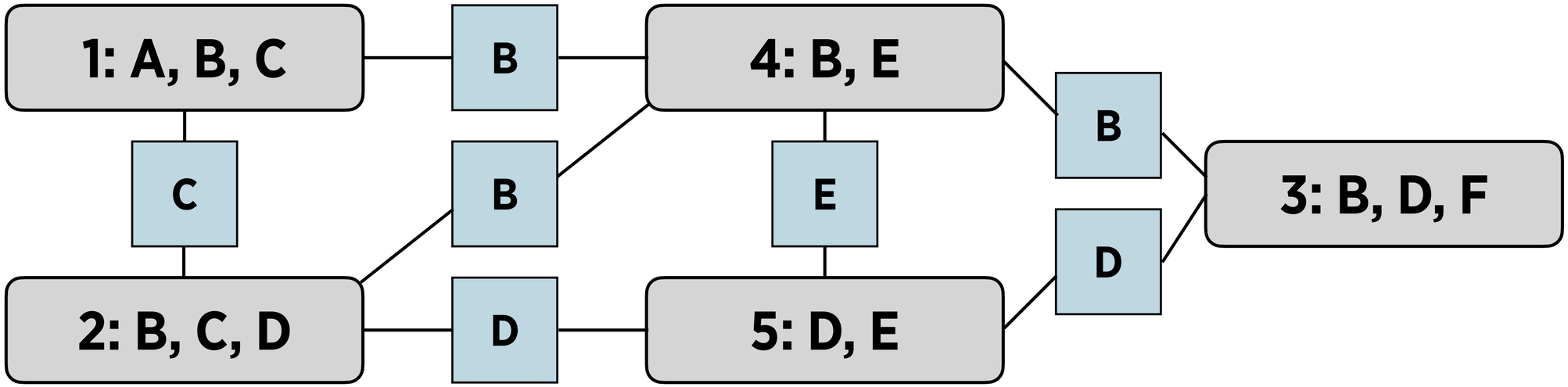
Running Intersection Property

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



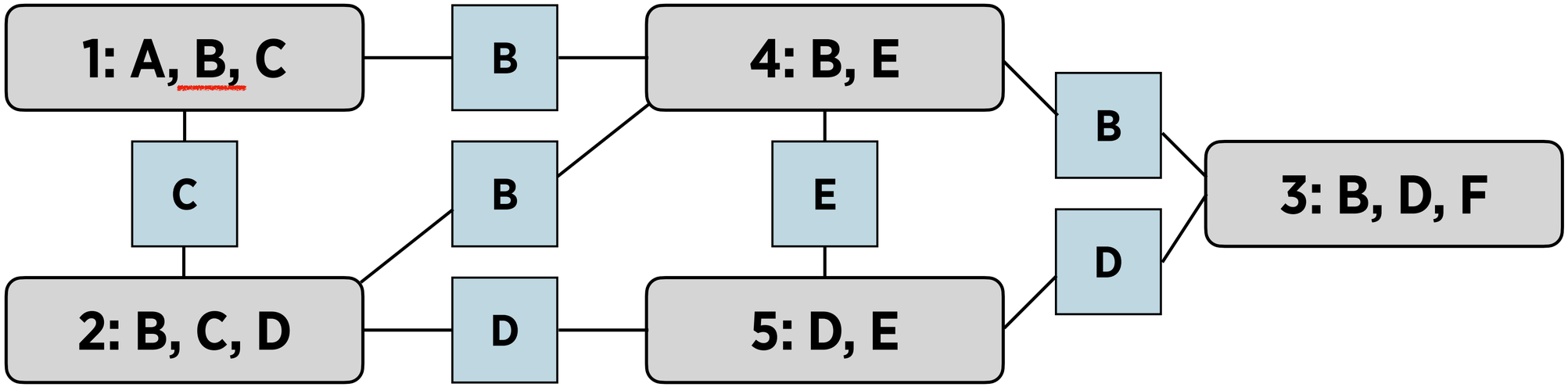
Example cluster graph

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



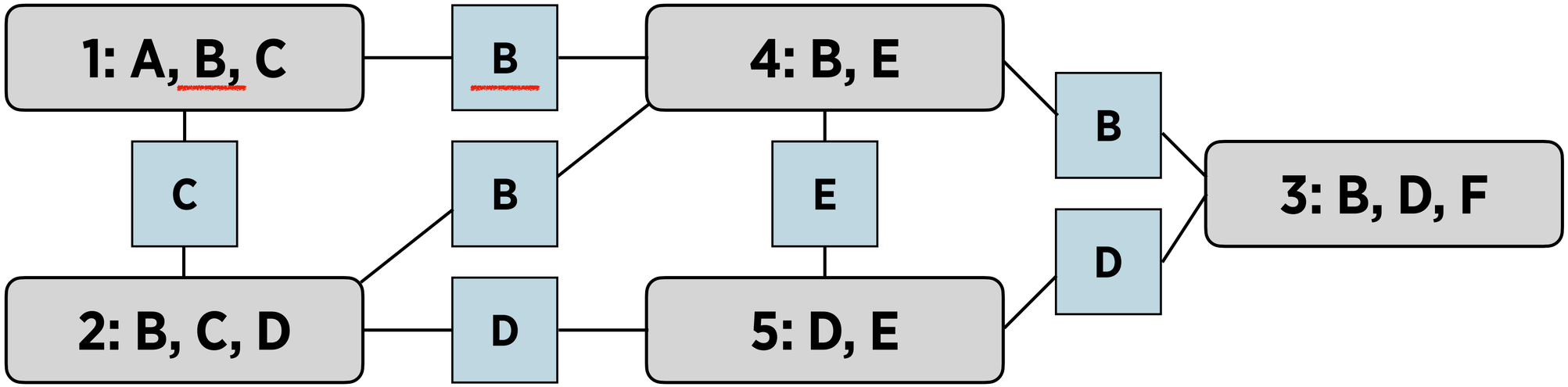
Example cluster graph

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



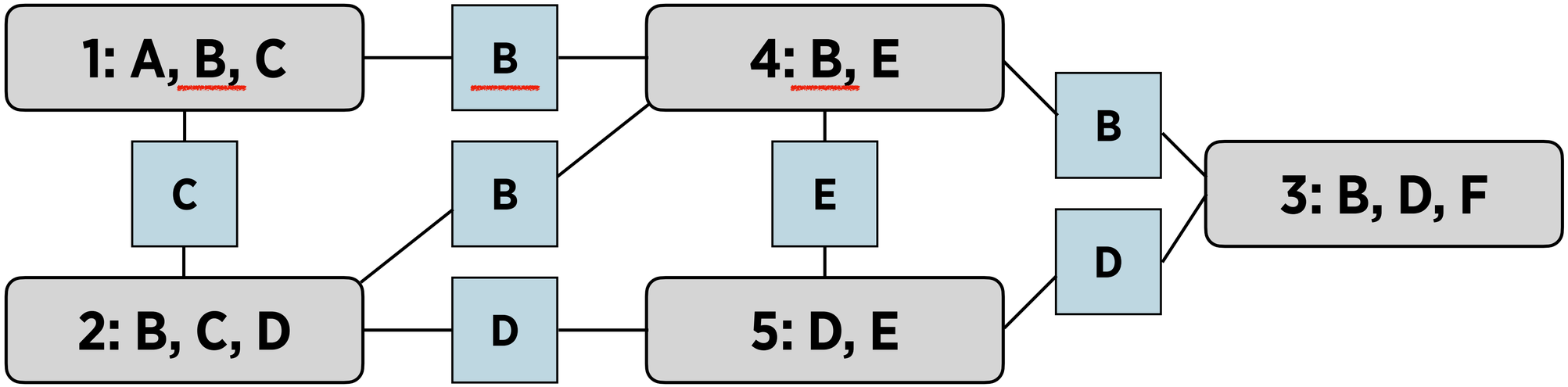
Example cluster graph

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



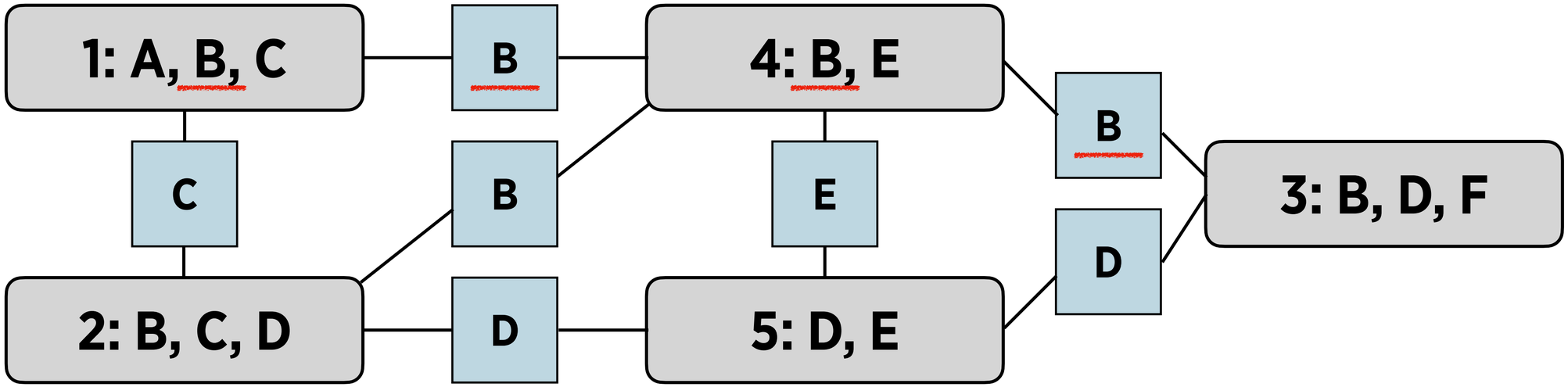
Example cluster graph

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



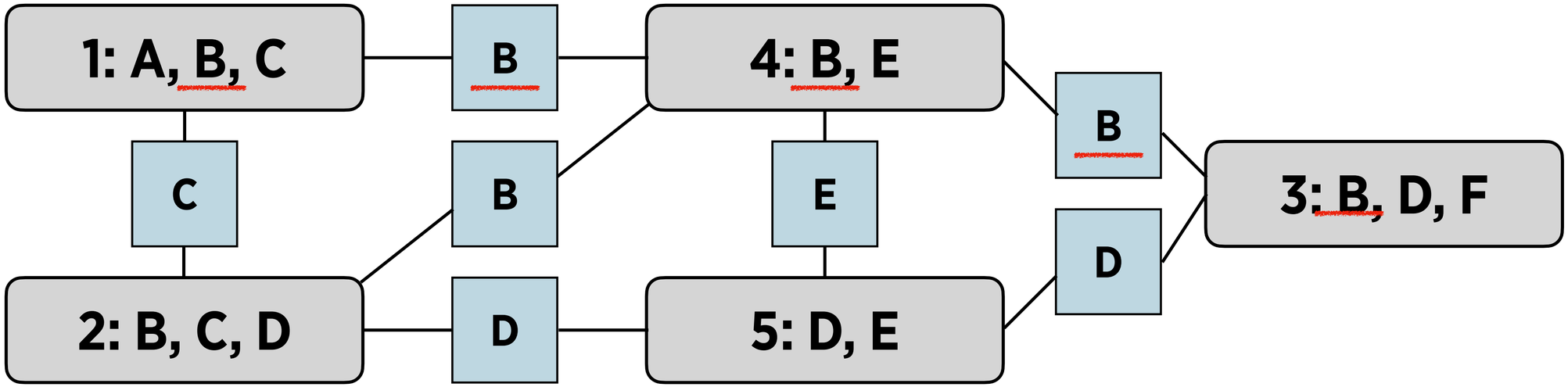
Example cluster graph

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



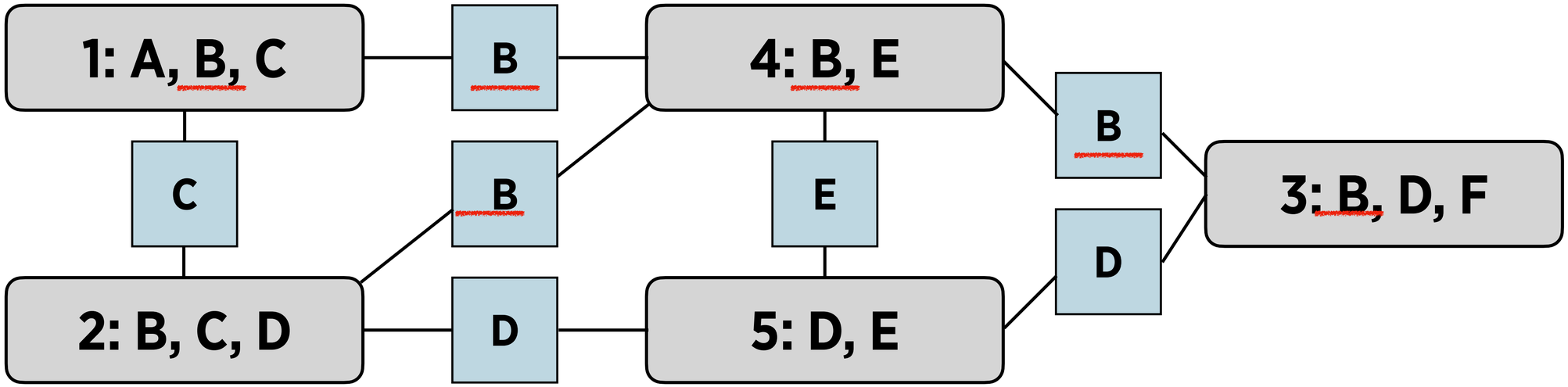
Example cluster graph

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



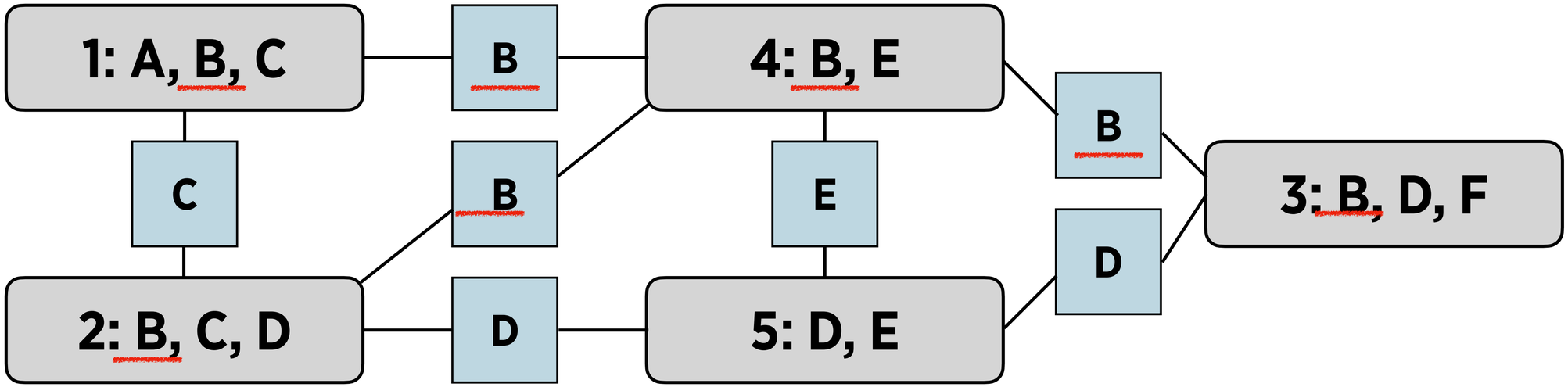
Example cluster graph

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



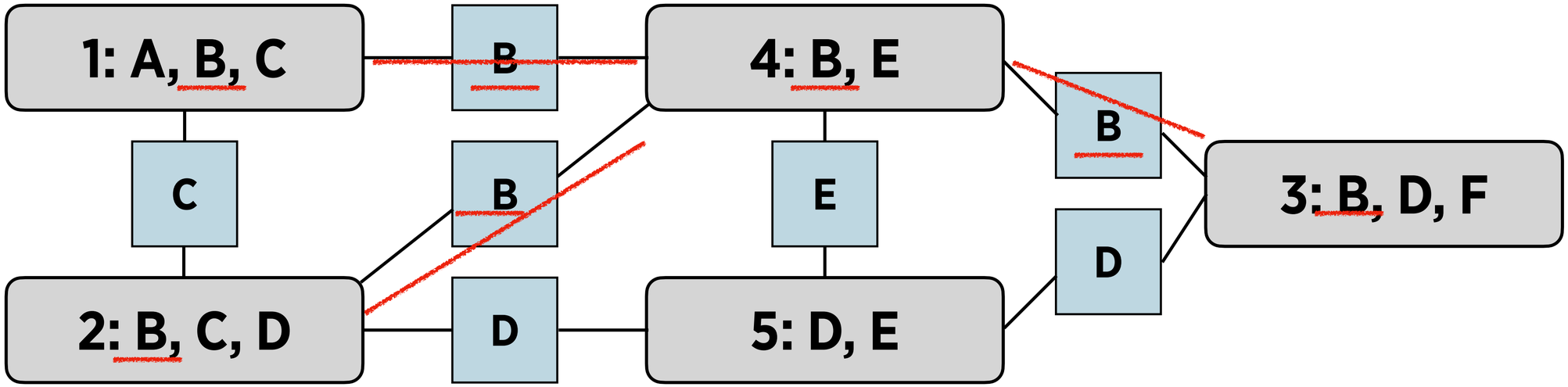
Example cluster graph

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

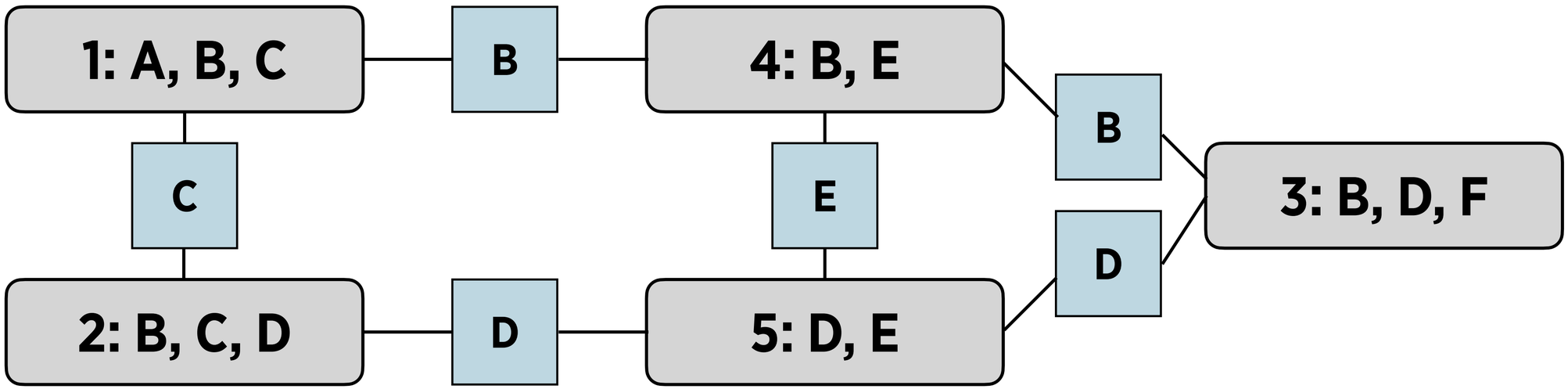


Example cluster graph

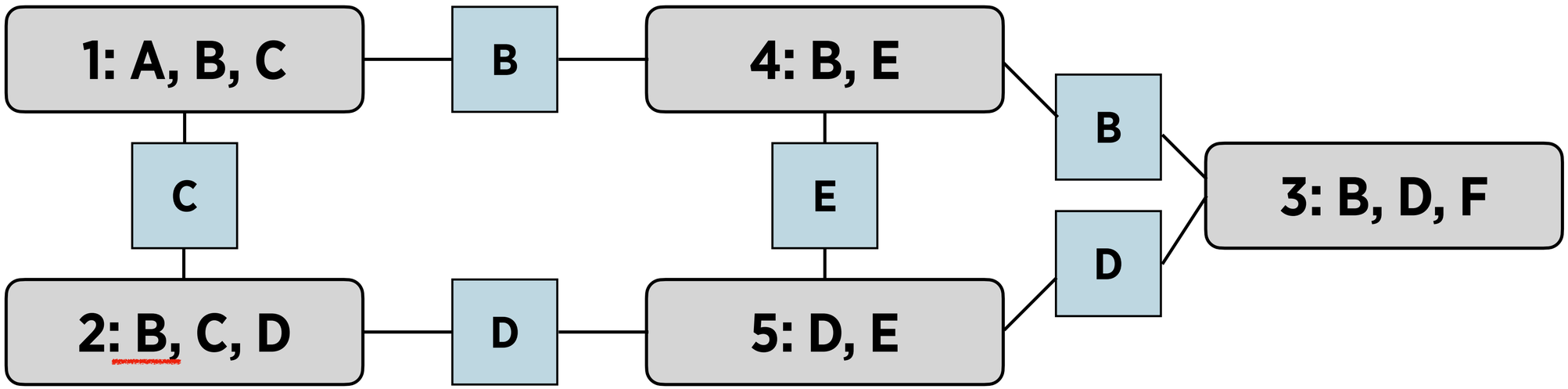
- Equivalently: For any X, the set of clusters and sepsets containing X form a tree



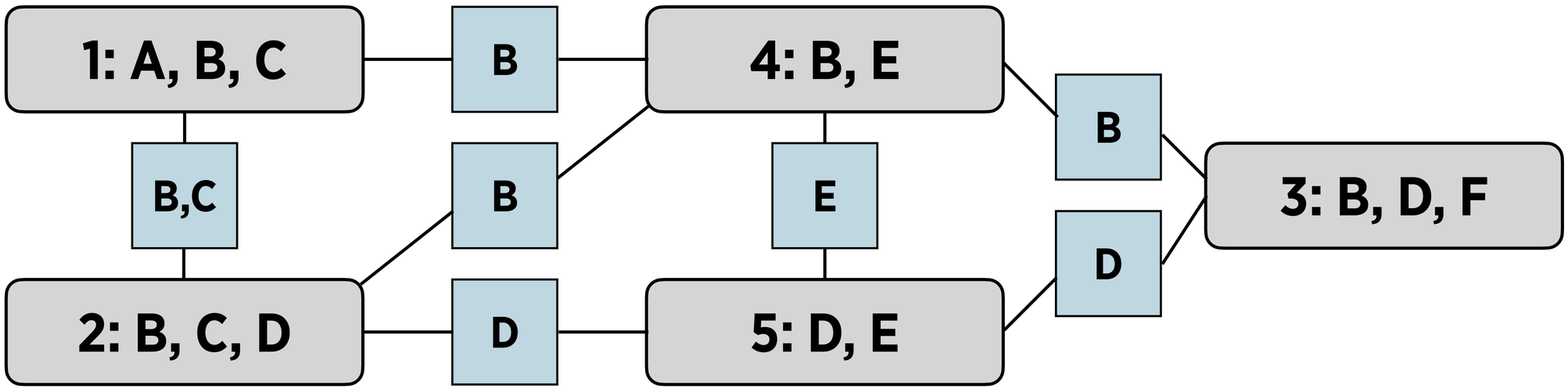
Illegal cluster graph (I)



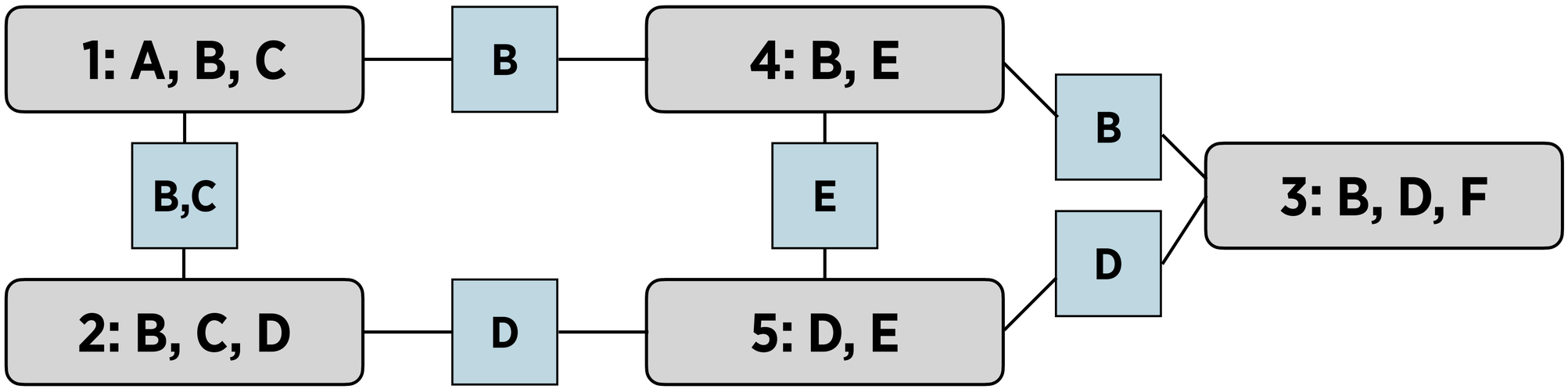
Illegal cluster graph (I)



Illegal cluster graph (II)



Alternative legal cluster graph



Bethe Cluster Graph

Bethe Cluster Graph

- For each $\phi_k \in \Phi$, a factor cluster $\boldsymbol{C}_k = \text{scope}[\phi_k]$

Bethe Cluster Graph

"big clusters" = factor in Φ

- For each $\phi_k \in \Phi$, a factor cluster $C_k = \text{scope}[\phi_k]$

Bethe Cluster Graph

"big clusters" = factor in Φ

- For each $\phi_k \in \Phi$, a factor cluster $\boldsymbol{C}_k = \text{scope}[\phi_k]$
- For each X_i a singleton cluster $\{X_i\}$

Bethe Cluster Graph

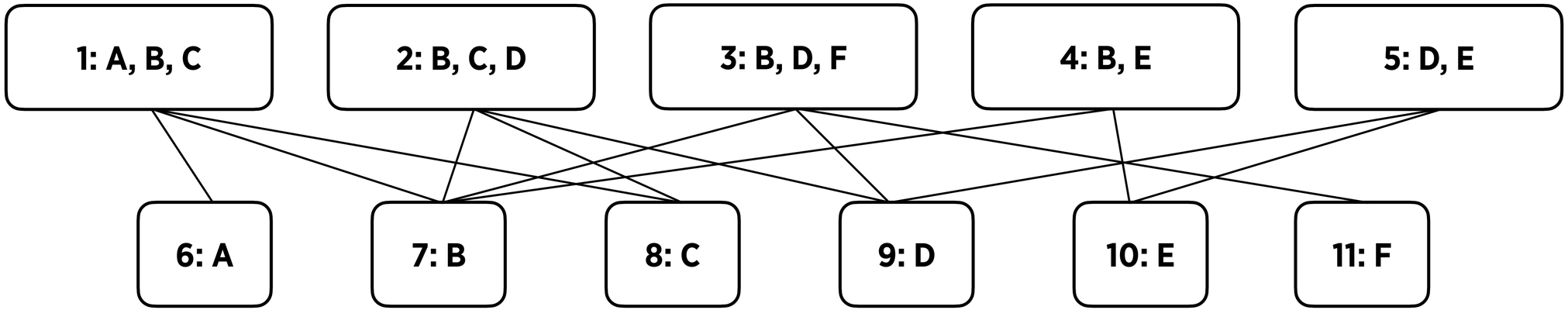
"big clusters" = factor in Φ

- For each $\phi_k \in \Phi$, a factor cluster $\boldsymbol{C}_k = \text{scope}[\phi_k]$
- For each X_i a singleton cluster $\{X_i\}$
- Edge $\boldsymbol{C}_k - X_i$ if $X_i \in \boldsymbol{C}_k$

Bethe Cluster Graph

"big clusters" = factor in Φ

- For each $\phi_k \in \Phi$, a factor cluster $\boldsymbol{C}_k = \text{scope}[\phi_k]$
- For each X_i a singleton cluster $\{X_i\}$
- Edge $\boldsymbol{C}_k - X_i$ if $X_i \in \boldsymbol{C}_k$



Summary cluster graphs

Summary cluster graphs

- Cluster graphs must satisfy two properties

Summary cluster graphs

- Cluster graphs must satisfy two properties
  - Family preservation: allows Φ to be encoded

Summary cluster graphs

- 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

Summary cluster graphs

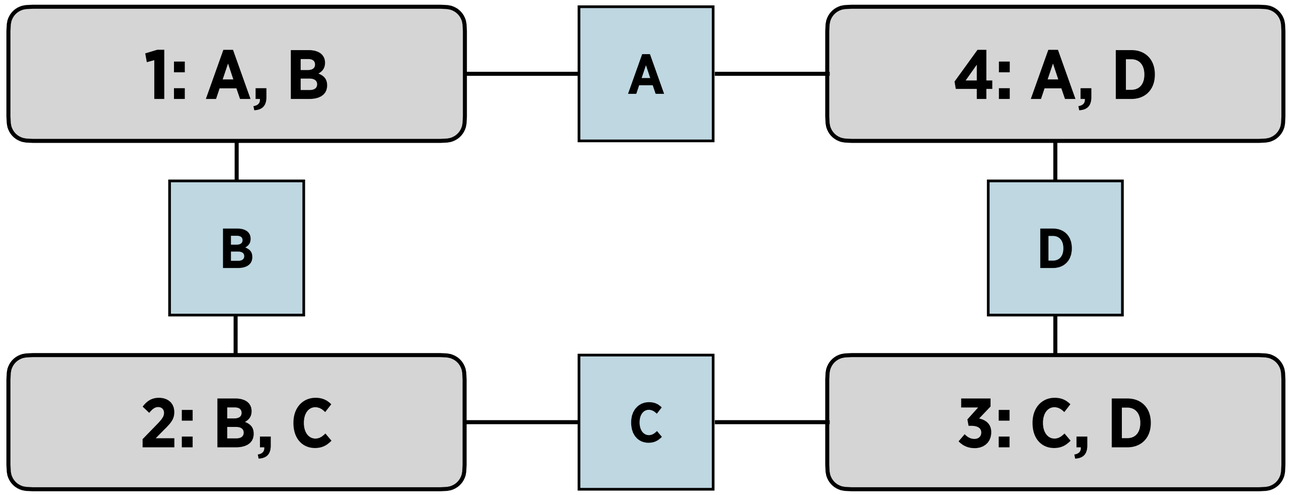
- Cluster graphs must satisfy two properties
  - Family preservation: allows Φ to be encoded
  - Running intersection: connects all information about any variable, but without feedback loops
- Bethe cluster graph is often first default

Summary cluster graphs

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

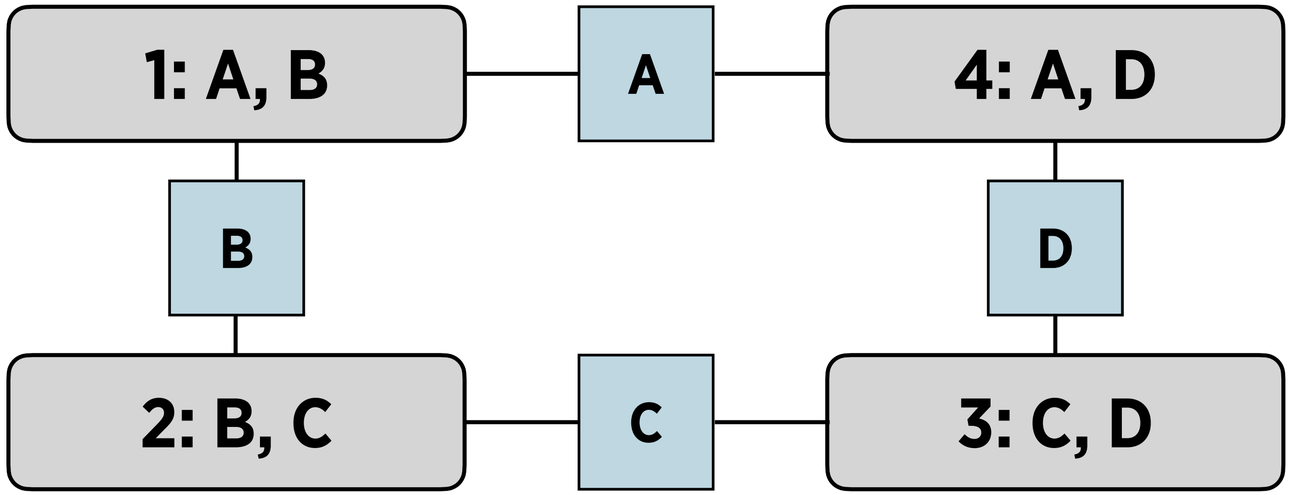
Properties of BP Algorithm

Calibration



Calibration

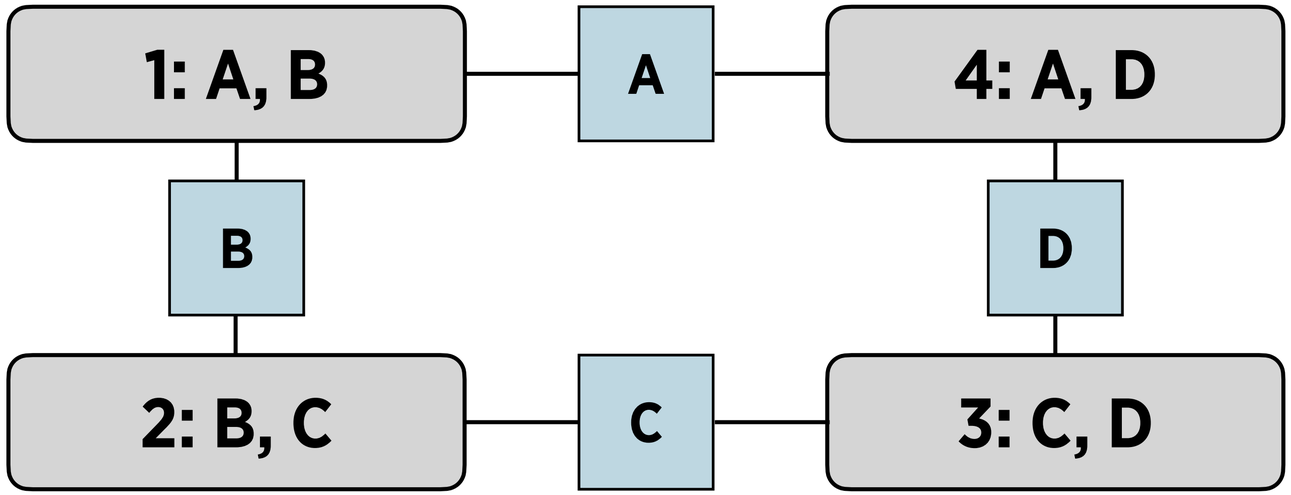
- Cluster beliefs:



Calibration

- Cluster beliefs:

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

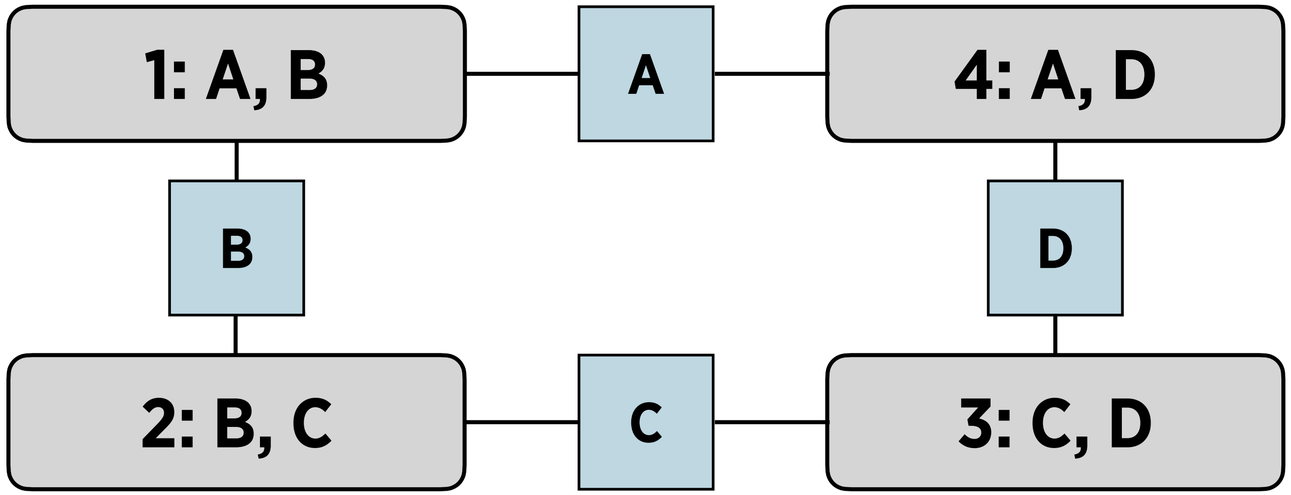


Calibration

- Cluster beliefs:

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

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



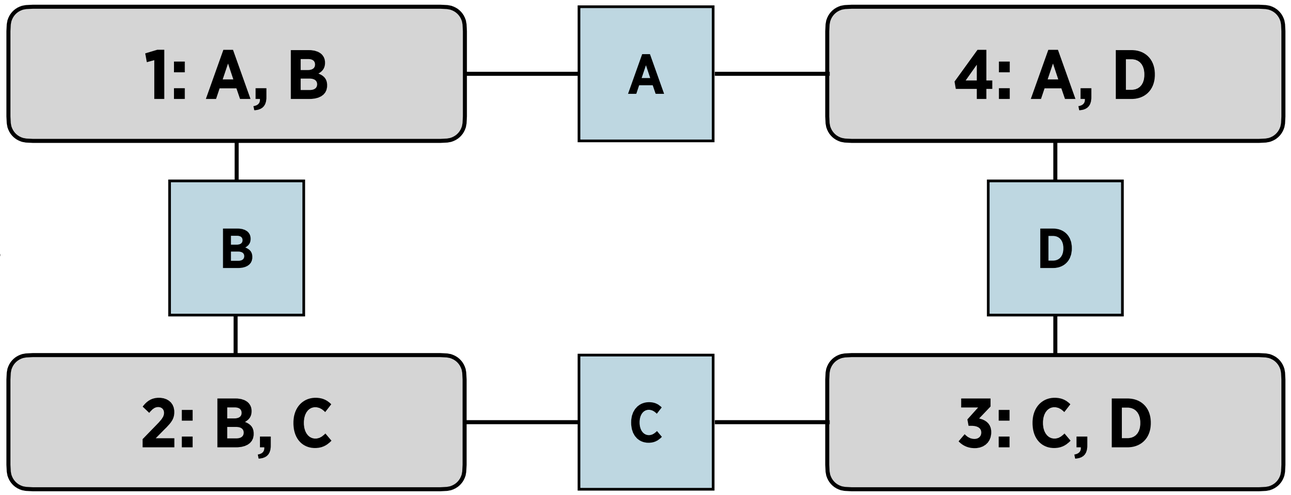
Calibration

- Cluster beliefs:

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

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

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



Calibration

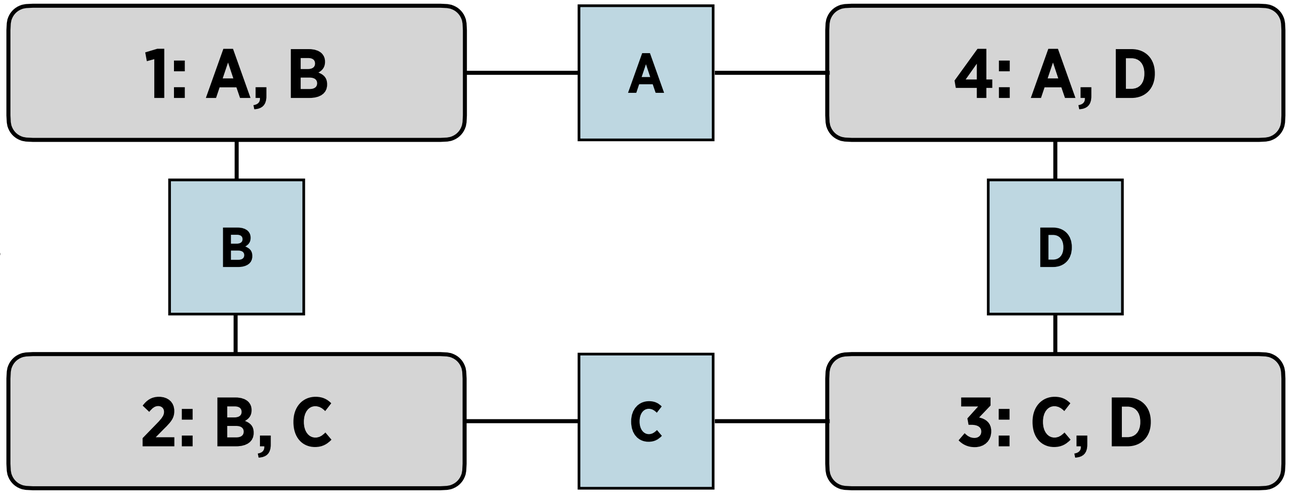
- Cluster beliefs:

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

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

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

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



Convergence ⇒ Calibration

Convergence ⇒ Calibration

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

Convergence ⇒ Calibration

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

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

Convergence ⇒ Calibration

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

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

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

Convergence ⇒ Calibration

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

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

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

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

Convergence ⇒ Calibration

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

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

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

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

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

Convergence ⇒ Calibration

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

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

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

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

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

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

Convergence ⇒ Calibration

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

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

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

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

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

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

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

Convergence ⇒ Calibration

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

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

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

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

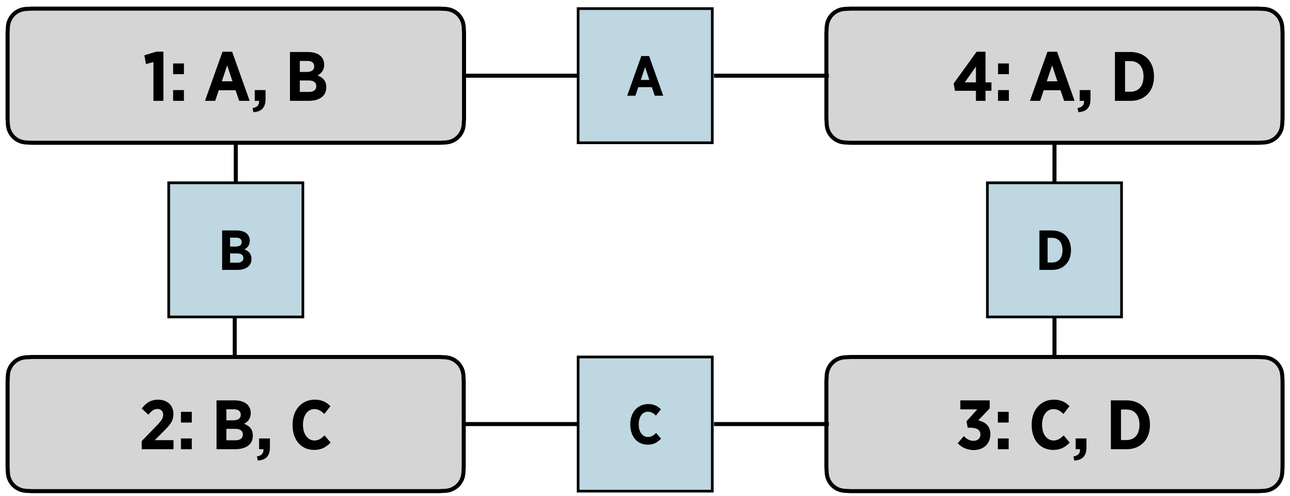
$$\delta_{j \to i}(\boldsymbol{S}_{i,j})\, \delta_{i \to j}(\boldsymbol{S}_{i,j}) = \sum_{\boldsymbol{C}_j - \boldsymbol{S}_{i,j}} \beta_j(\boldsymbol{C}_j)$$

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

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

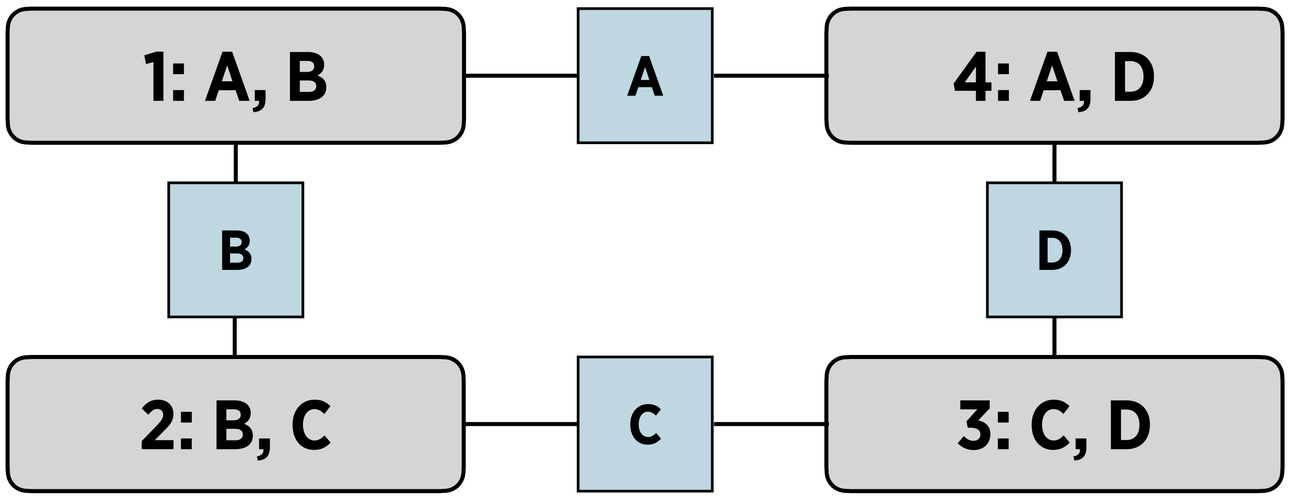
sepset belief

Reparameterisation



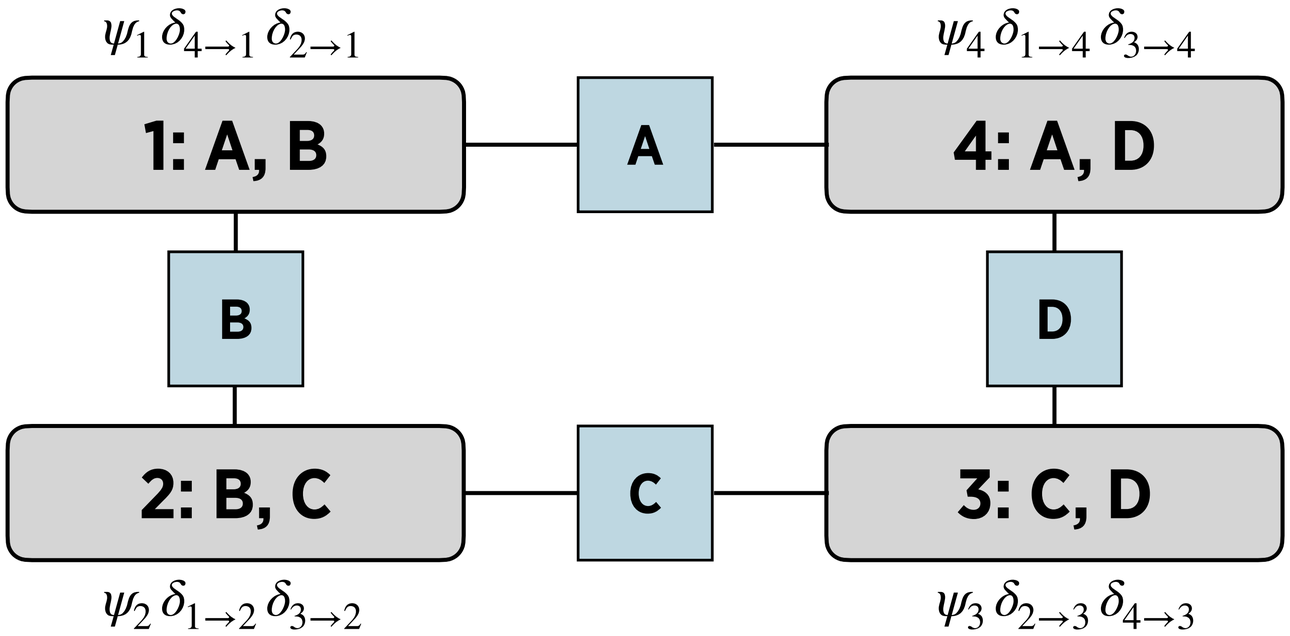
Reparameterisation

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



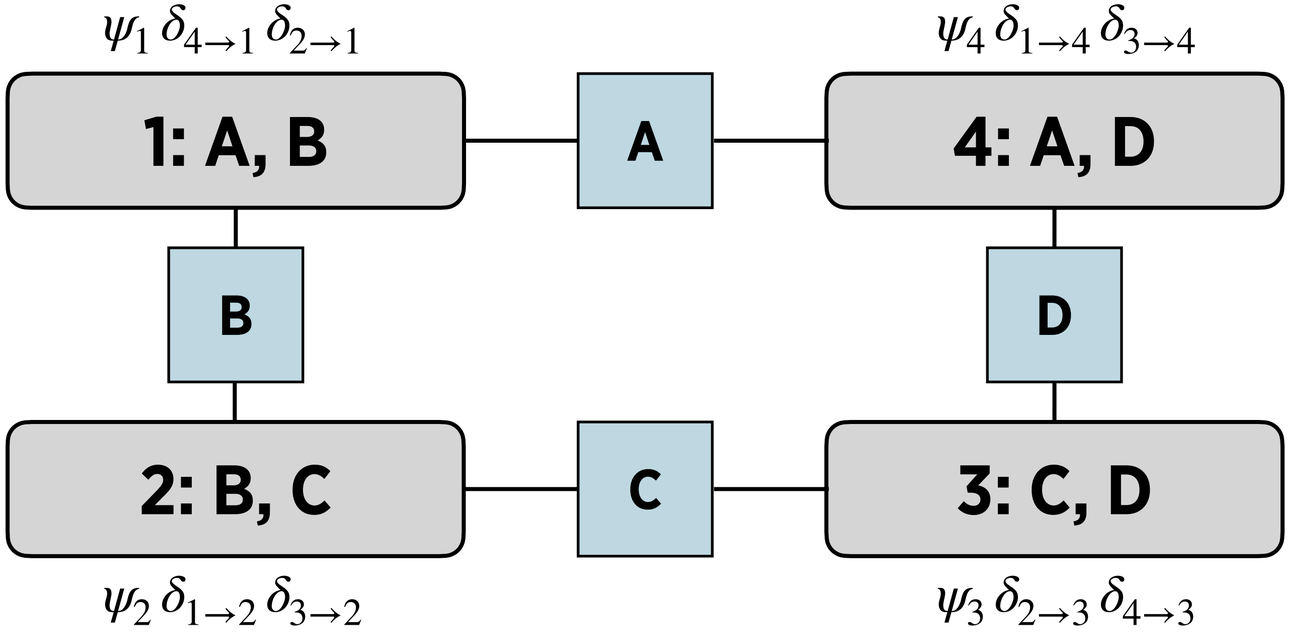
Reparameterisation

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



Reparameterisation

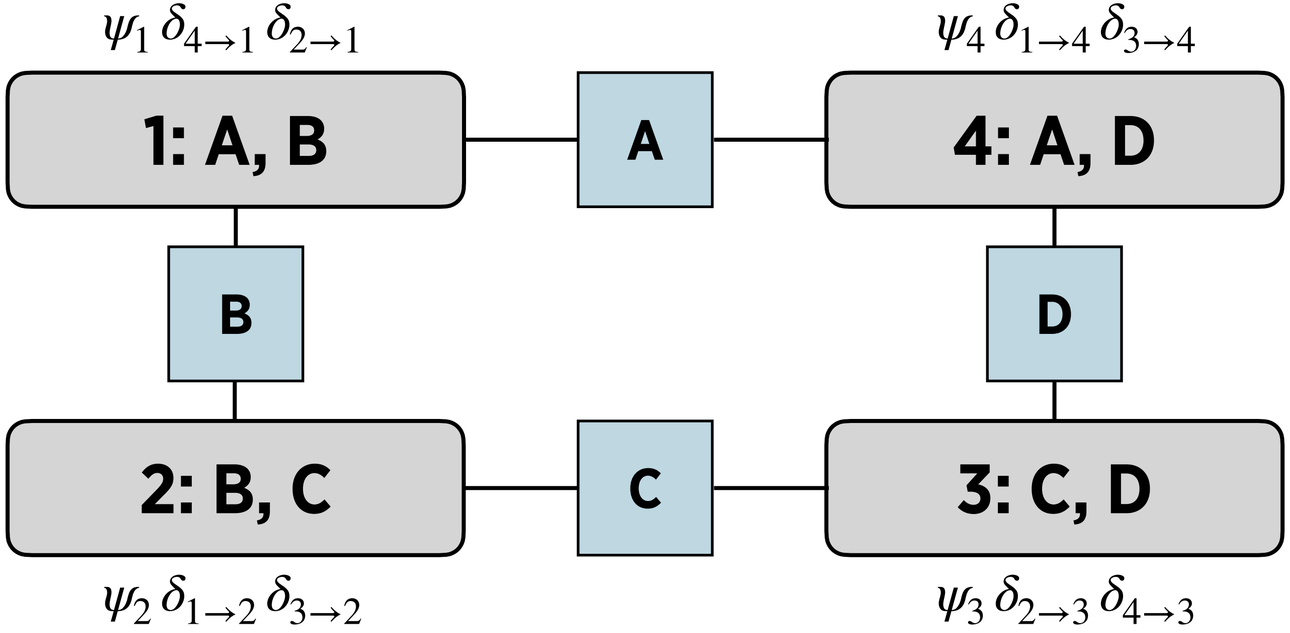
$$\beta_i(\boldsymbol{C}_i) = \psi_i \prod_{k \in \mathcal{N}_i} \delta_{k \to i} \qquad \mu_{i,j}(\boldsymbol{S}_{i,j}) = \delta_{j \to i} \delta_{i \to j}$$



Reparameterisation

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

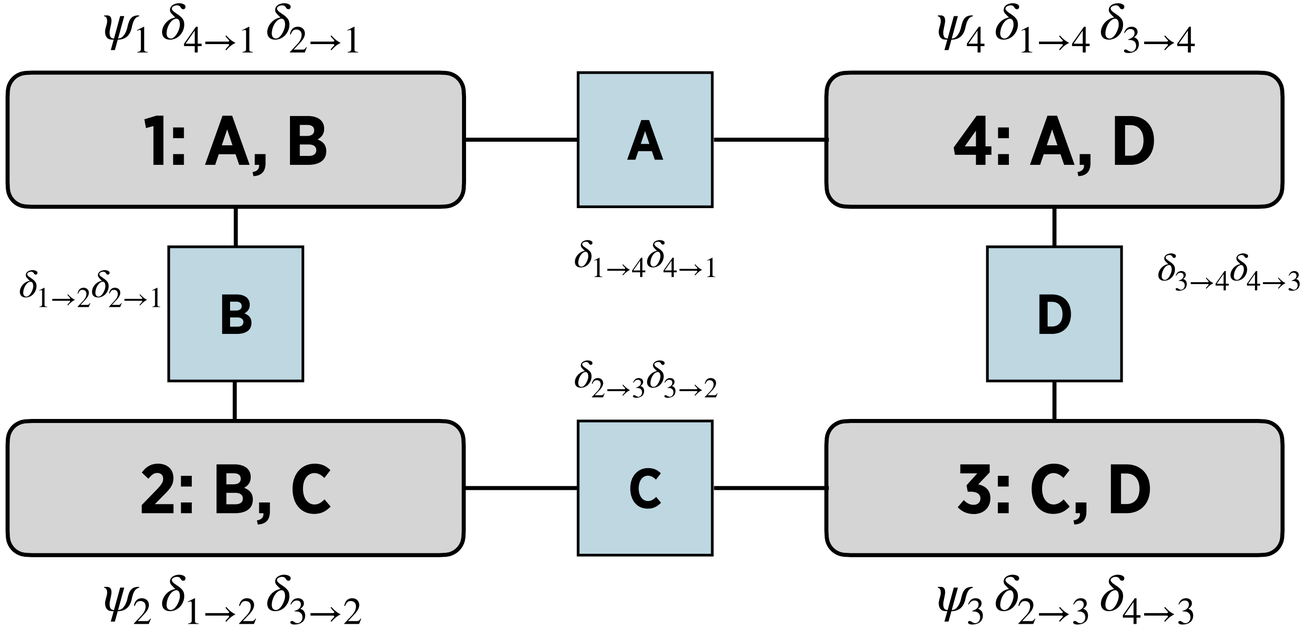
sepset belief



Reparameterisation

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

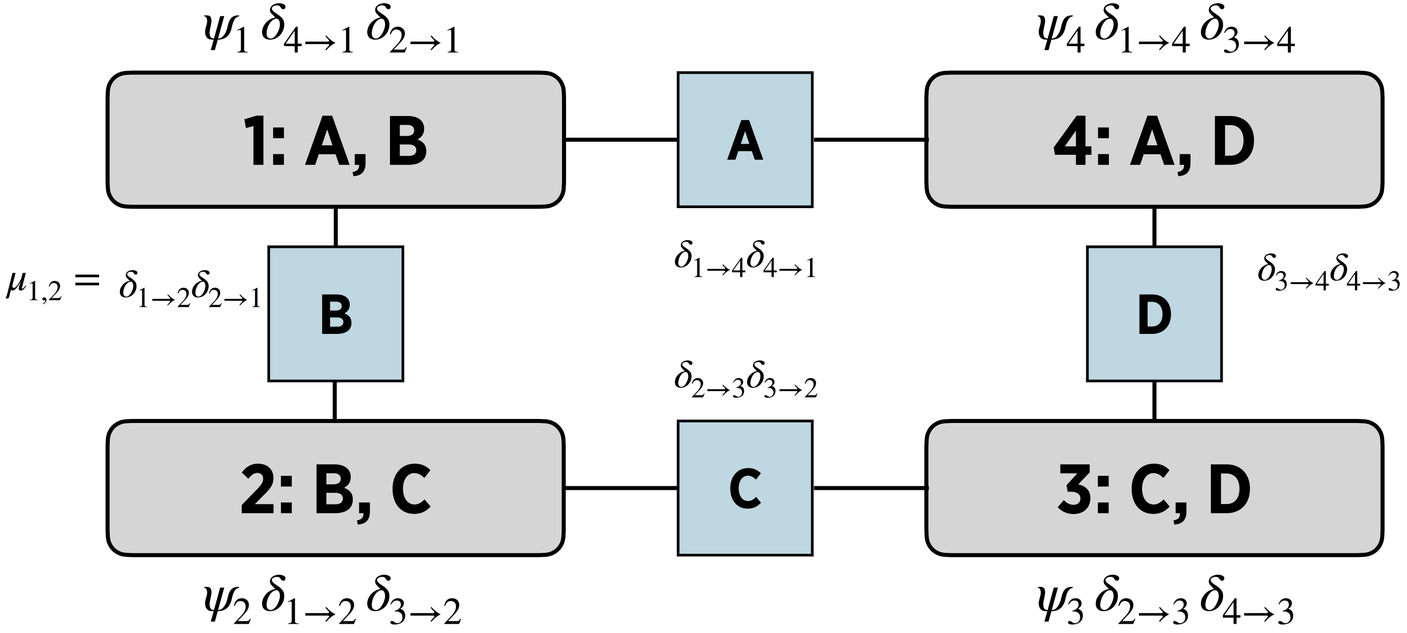
sepset belief



Reparameterisation

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

sepset belief

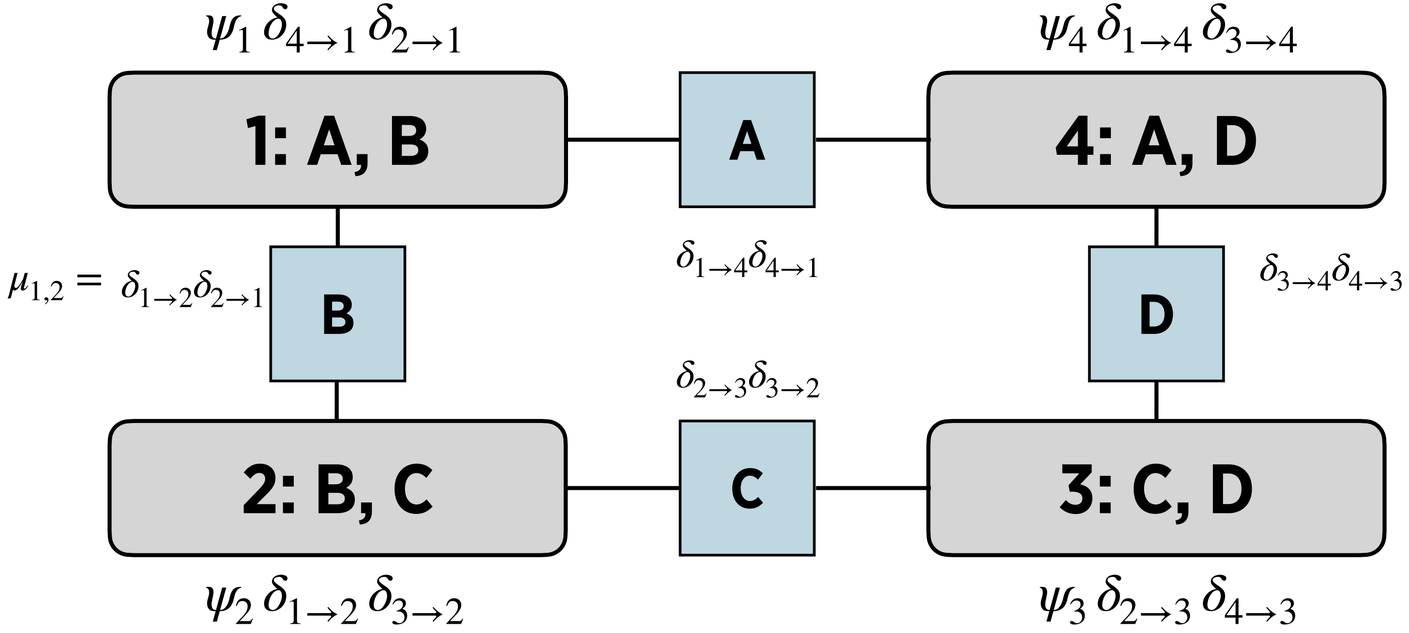


Reparameterisation

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

sepset belief

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



Reparameterisation

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

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

Reparameterisation

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

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

Reparameterisation

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

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

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

Summary Properties of Belief Propagation

- At convergence of BP, cluster graph beliefs are calibrated:

Summary Properties of Belief Propagation

- At convergence of BP, cluster graph beliefs are calibrated:
- Beliefs at adjacent clusters agree on sepsets

Summary Properties of Belief Propagation

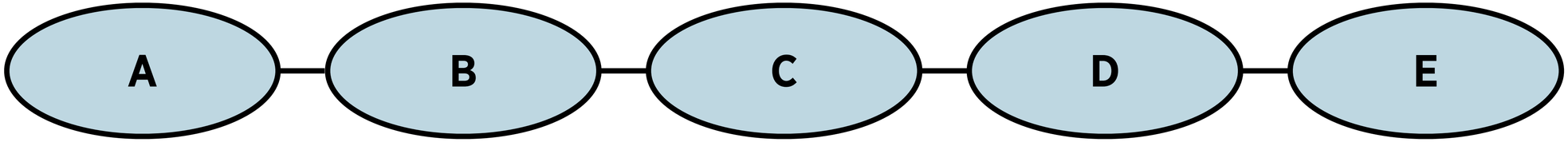
- At convergence of BP, cluster graph beliefs are calibrated:
- Beliefs at adjacent clusters agree on sepsets
- Cluster graph beliefs are an alternative calibrated parameterisation of the original unnormalised density

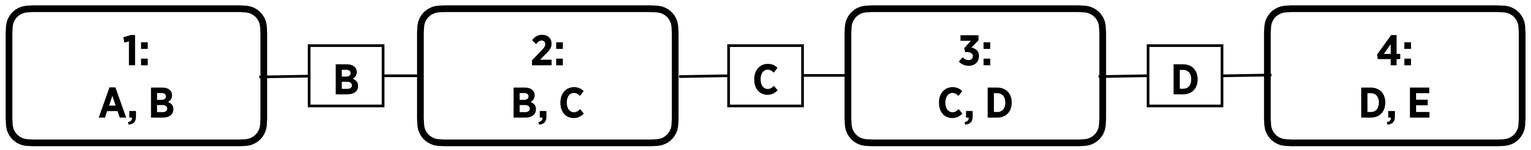
Summary Properties of Belief Propagation

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

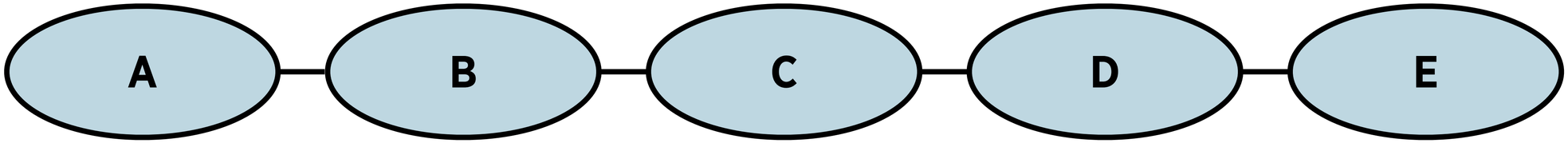
Clique Tree Algorithm & Correctness

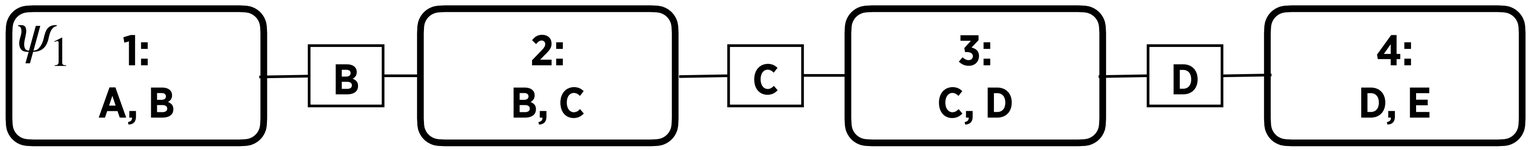
Message Passing in Trees



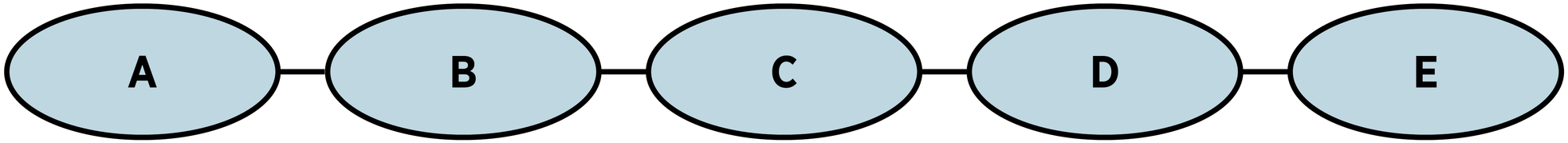


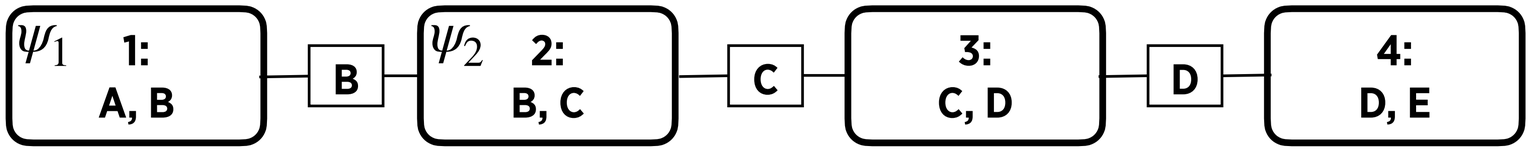
Message Passing in Trees



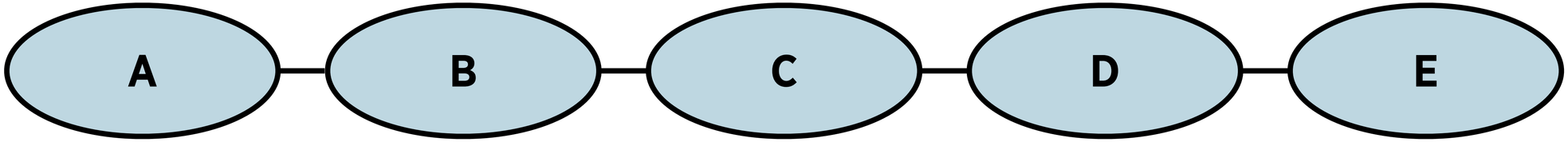


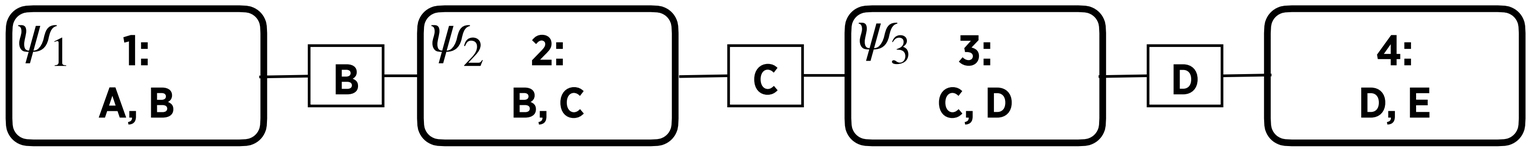
Message Passing in Trees



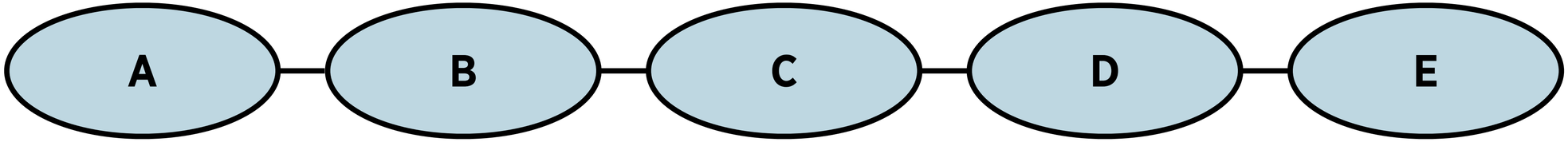


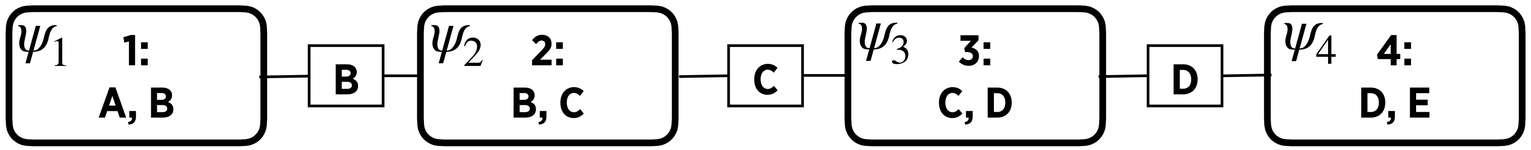
Message Passing in Trees



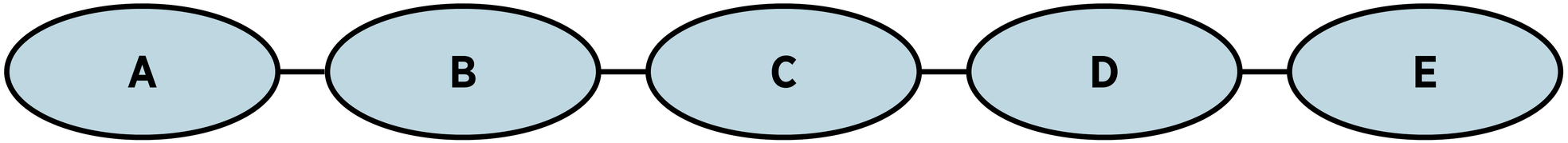


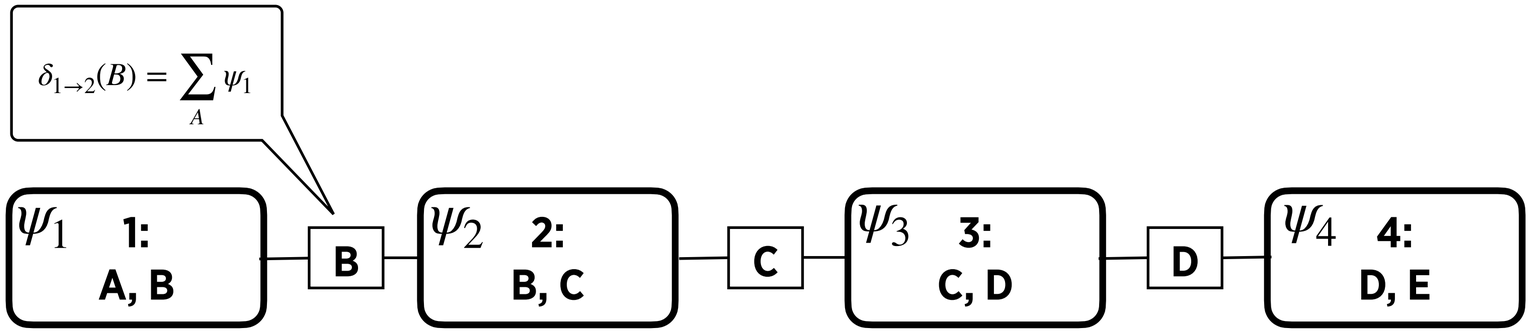
Message Passing in Trees



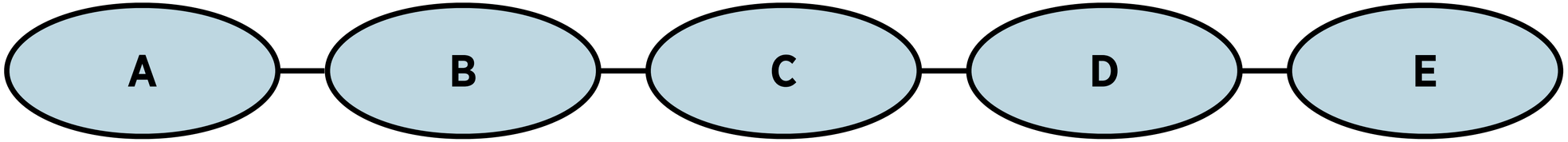


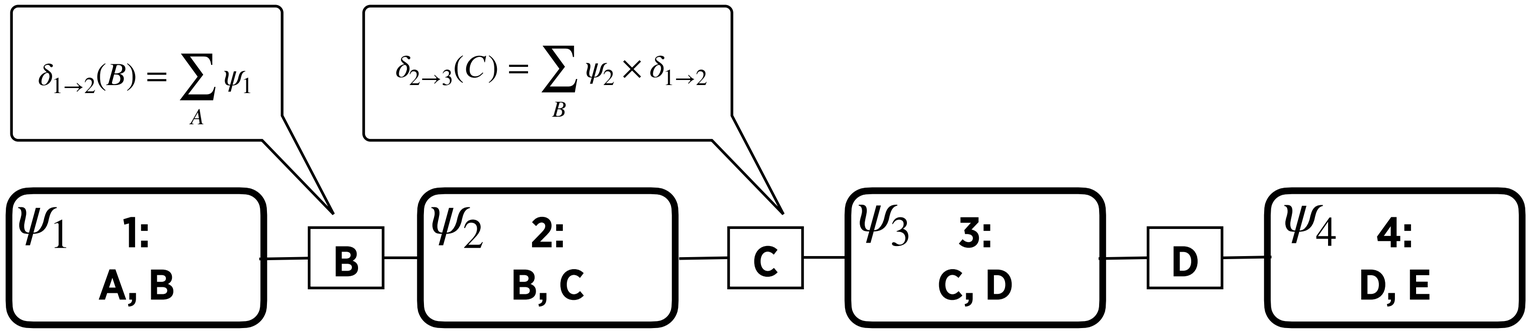
Message Passing in Trees



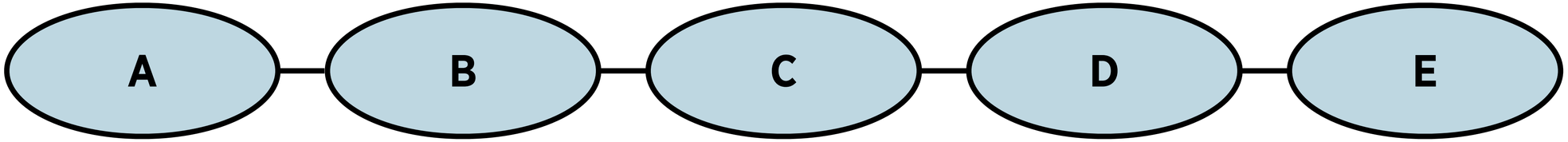


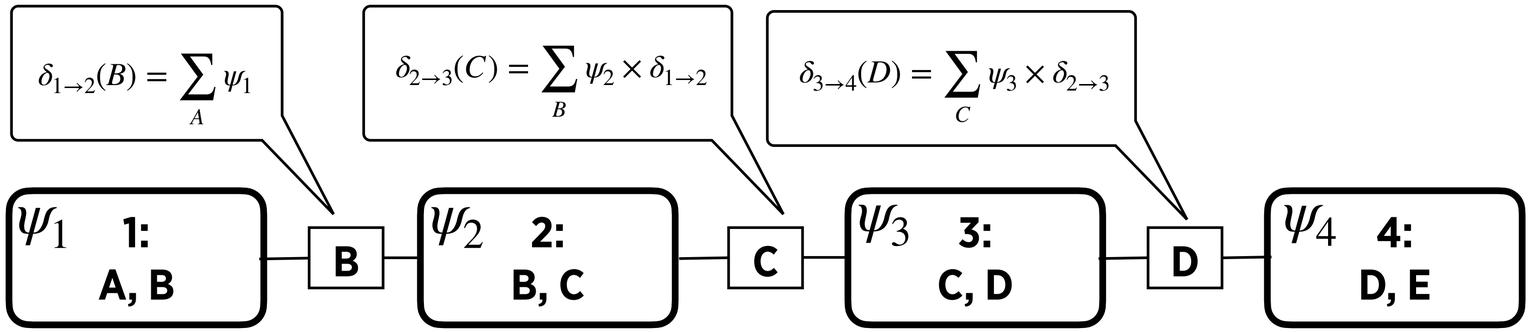
Message Passing in Trees



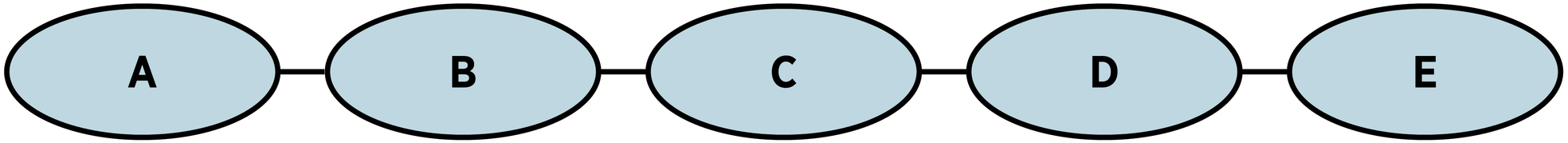


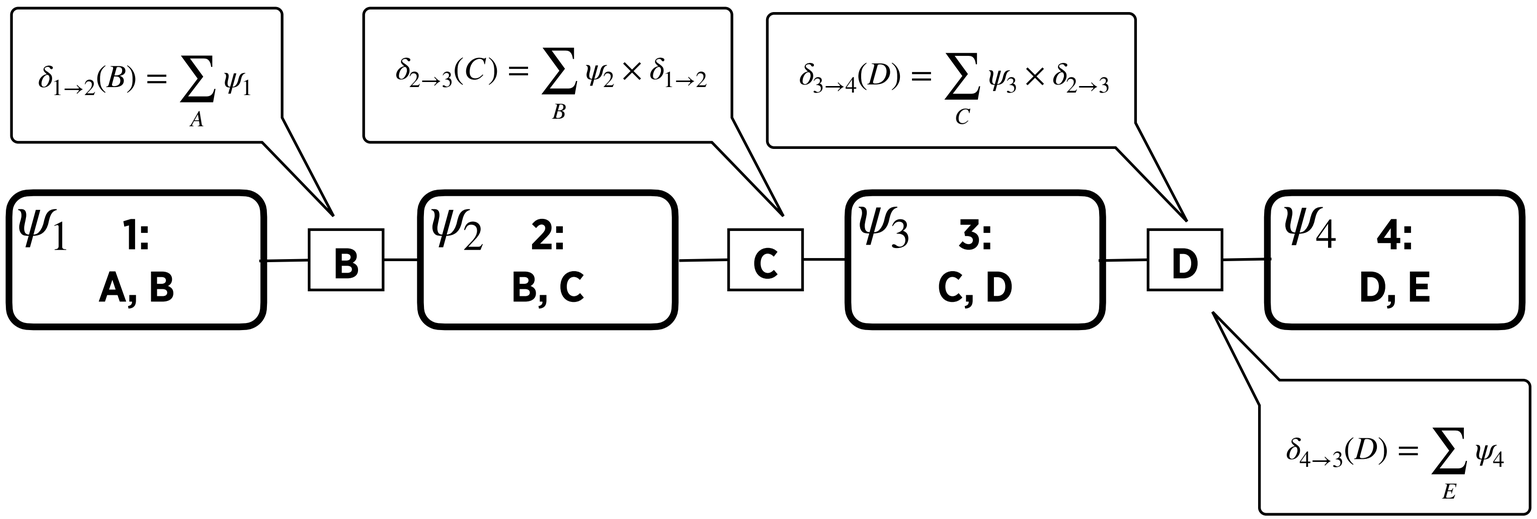
Message Passing in Trees



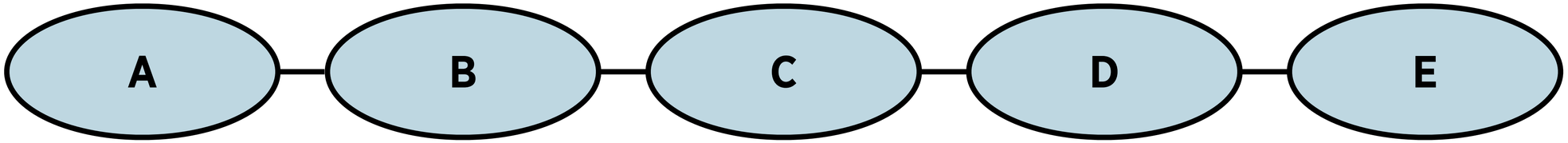


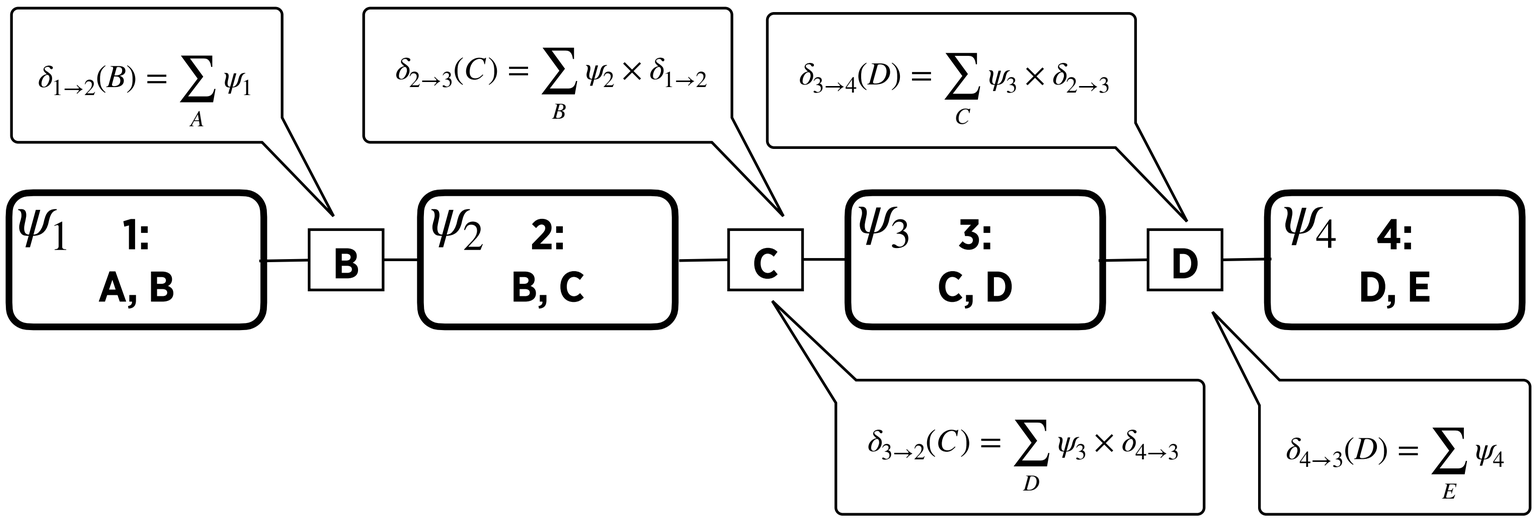
Message Passing in Trees



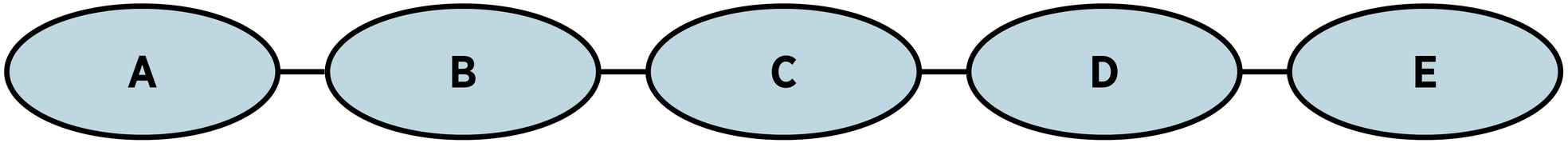


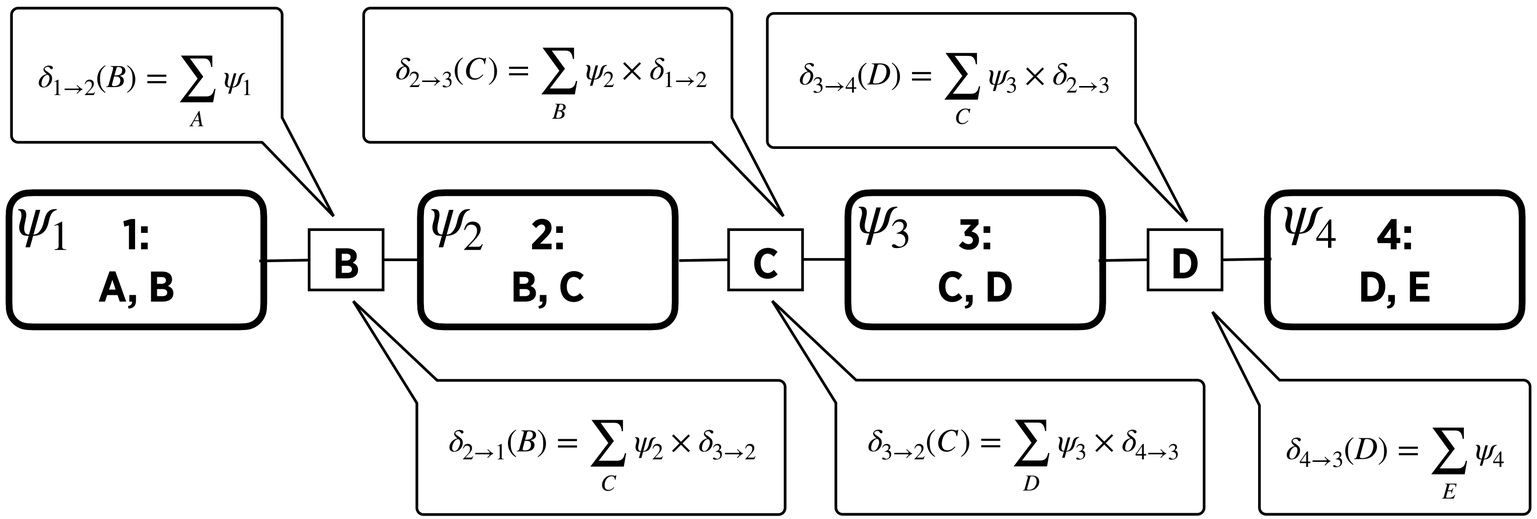
Message Passing in Trees



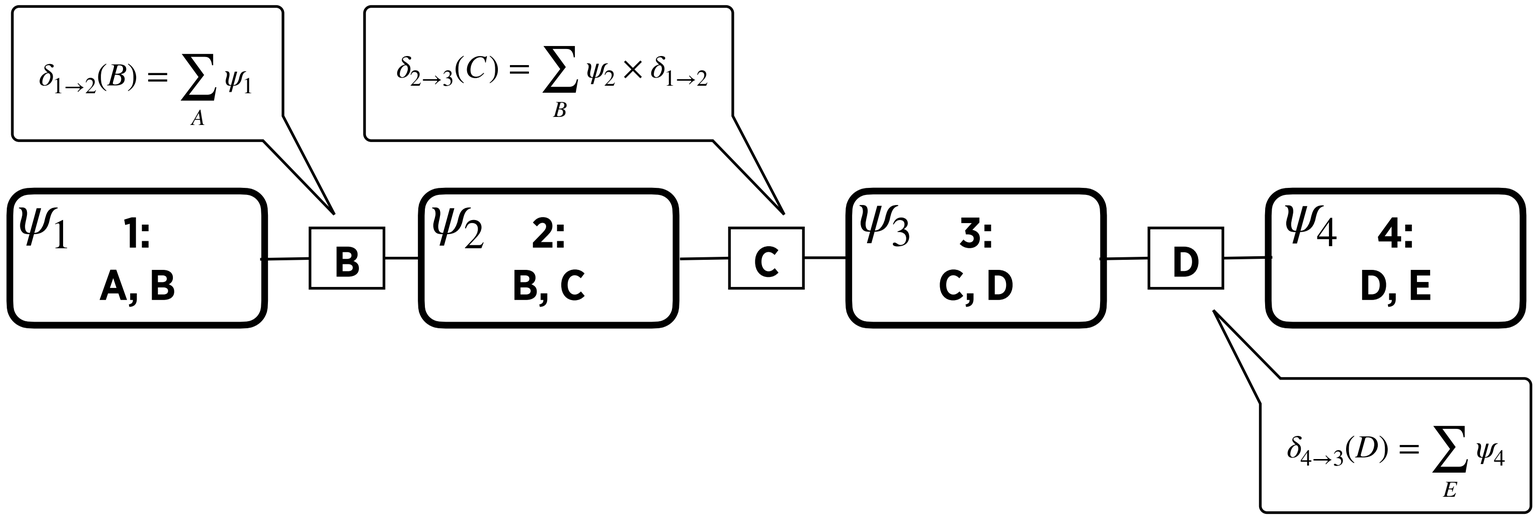


Message Passing in Trees

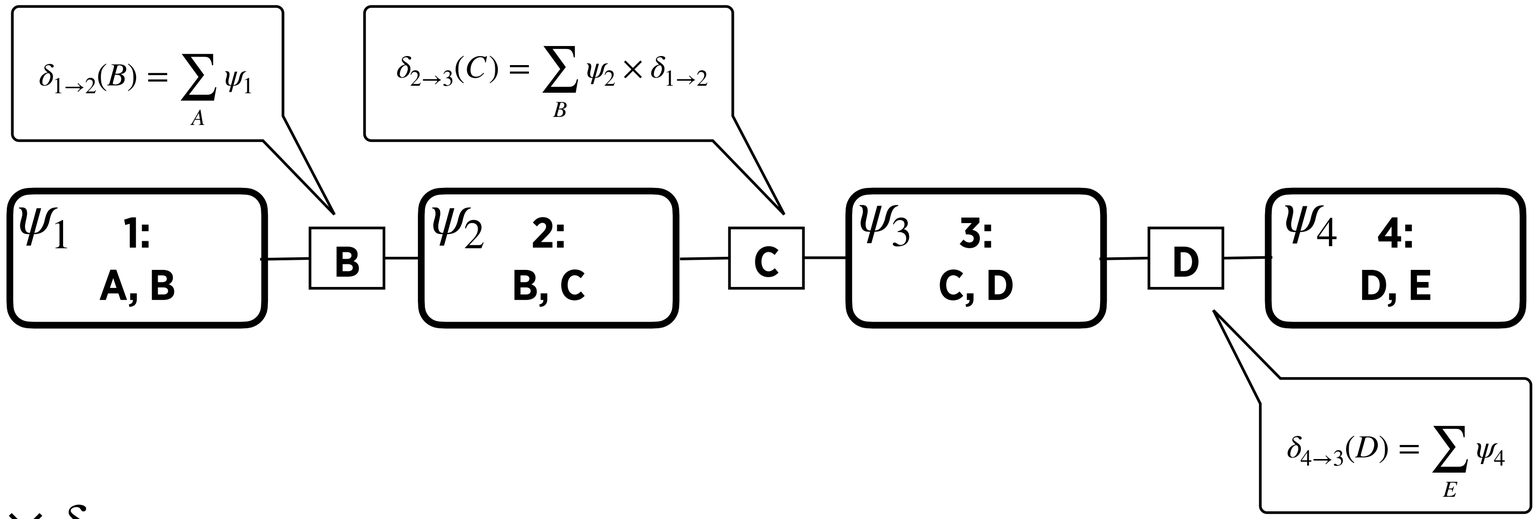




Correctness

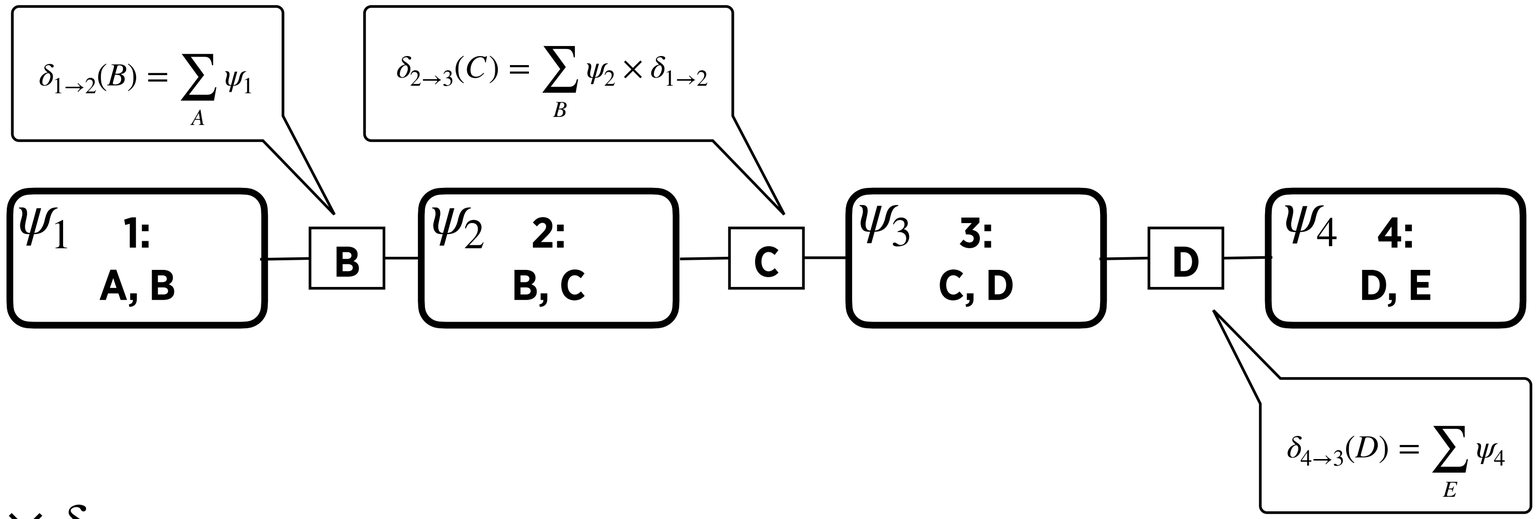


Correctness



$$\beta_3(C, D) = \psi_3 \times \delta_{2\to 3} \times \delta_{4\to 3}$$

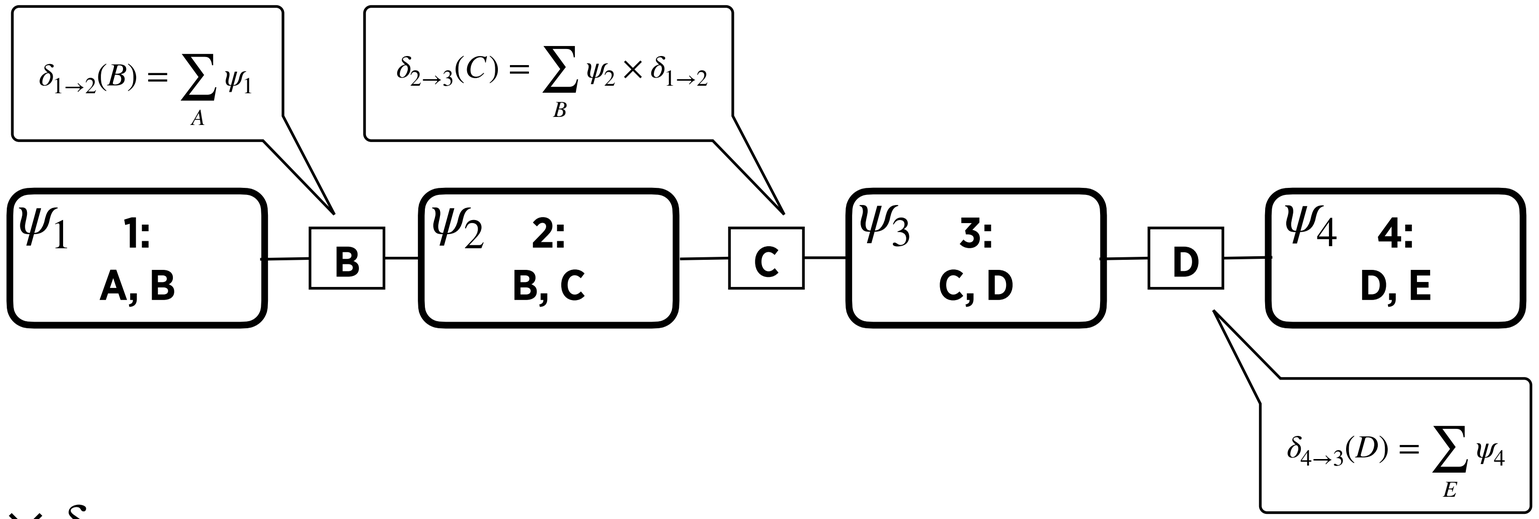
Correctness



$$\beta_3(C,D) = \psi_3 \times \delta_{2\to3} \times \delta_{4\to3}$$

$$= \psi_3 \times \left(\sum_B \psi_2 \times \delta_{1\to2} \right) \times \sum_E \psi_4$$

Correctness



$$\beta_3(C,D) = \psi_3 \times \delta_{2\to3} \times \delta_{4\to3}$$

$$= \psi_3 \times \left(\sum_B \psi_2 \times \delta_{1\to2} \right) \times \sum_E \psi_4$$

$$= \psi_3 \times \left(\sum_B \psi_2 \times \sum_A \psi_1 \right) \times \sum_E \psi_4$$

Clique Tree

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 $\boldsymbol{S}_{i,j} = \boldsymbol{C}_i \cap \boldsymbol{C}_j$

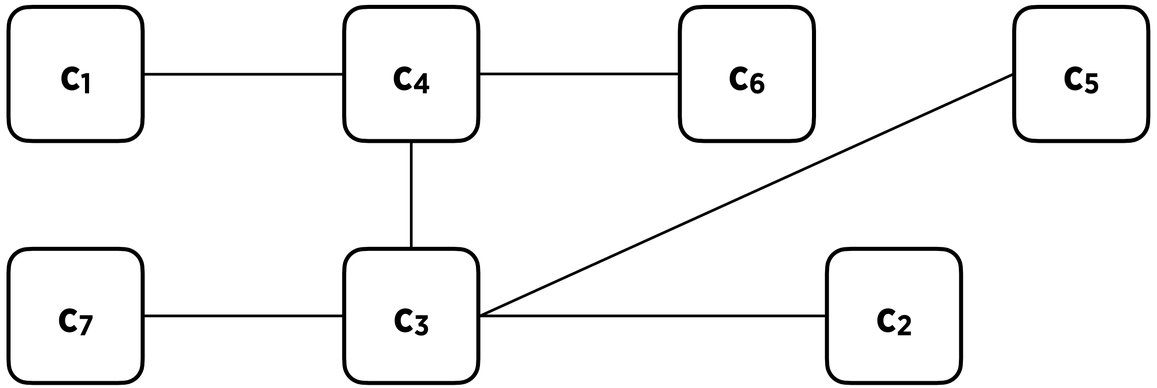
Family Preservation

- Given set of factors Φ, we assign each $\phi_k \in \Phi$ to a cluster $\boldsymbol{C}_{\alpha(k)}$ s.t. scope$[\alpha_k] \subseteq \boldsymbol{C}_{\alpha(k)}$

- For each factor $\phi_k \in \Phi$, there exists a cluster $\boldsymbol{C}_i$ s.t. scope$[\phi_k] \subseteq \boldsymbol{C}_i$

Running Intersection Property

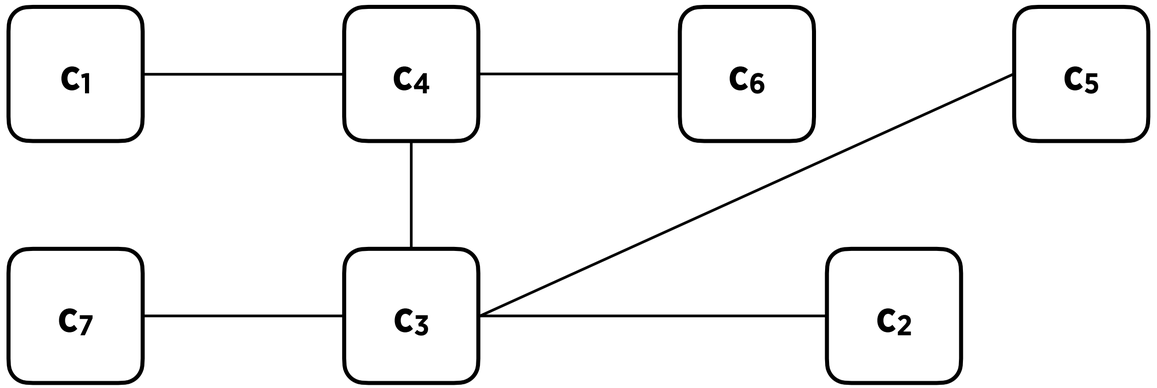
- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$ there exists a unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain X



Running Intersection Property

Cluster graph variant of the property

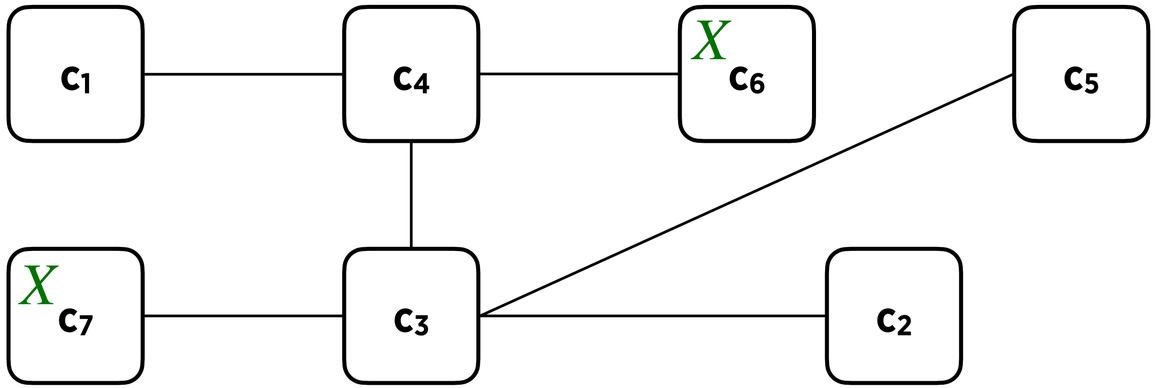
- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$ there exists a unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain X



Running Intersection Property

Cluster graph variant of the property

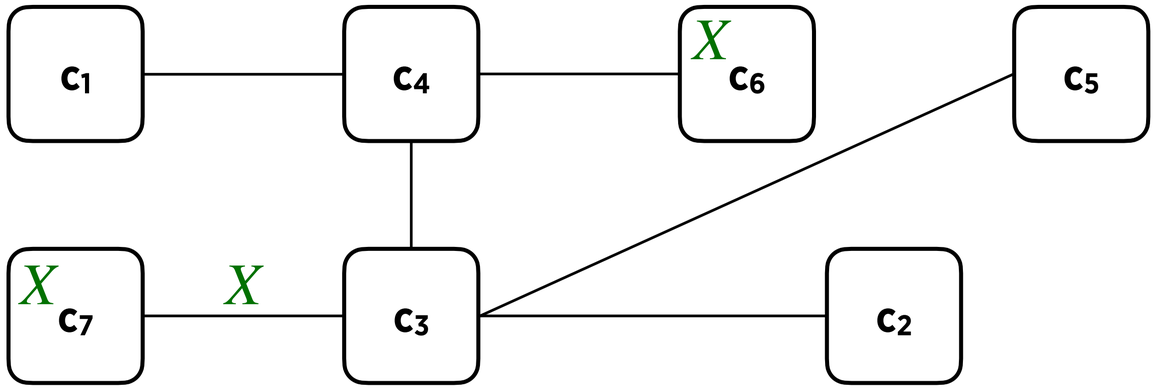
- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$ there exists a unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain X



Running Intersection Property

Cluster graph variant of the property

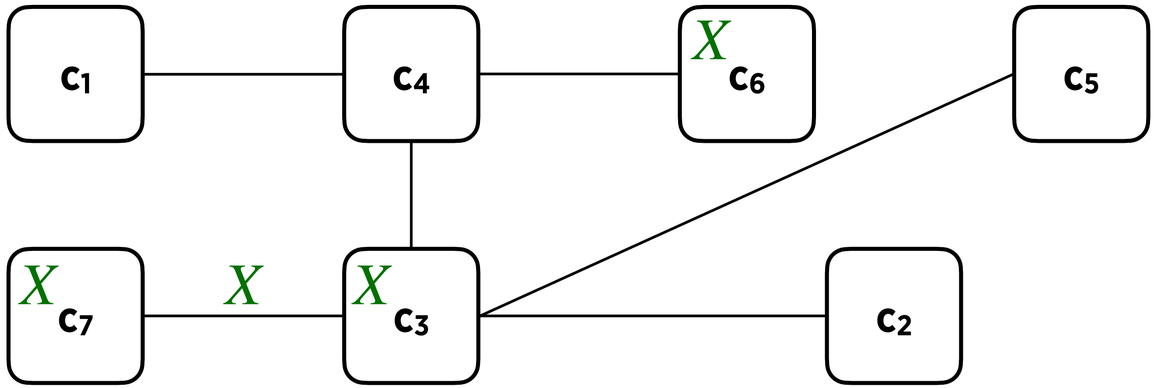
- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$ there exists a unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain X



Running Intersection Property

Cluster graph variant of the property

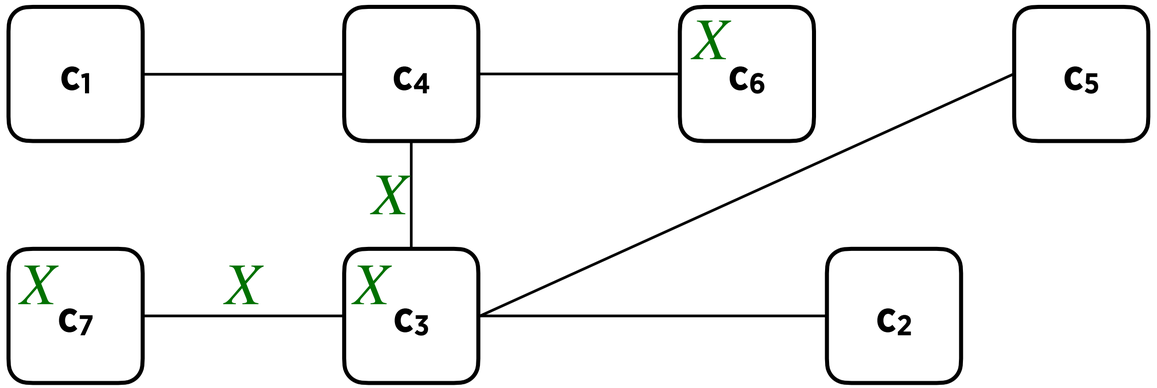
- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$ there exists a unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain X



Running Intersection Property

Cluster graph variant of the property

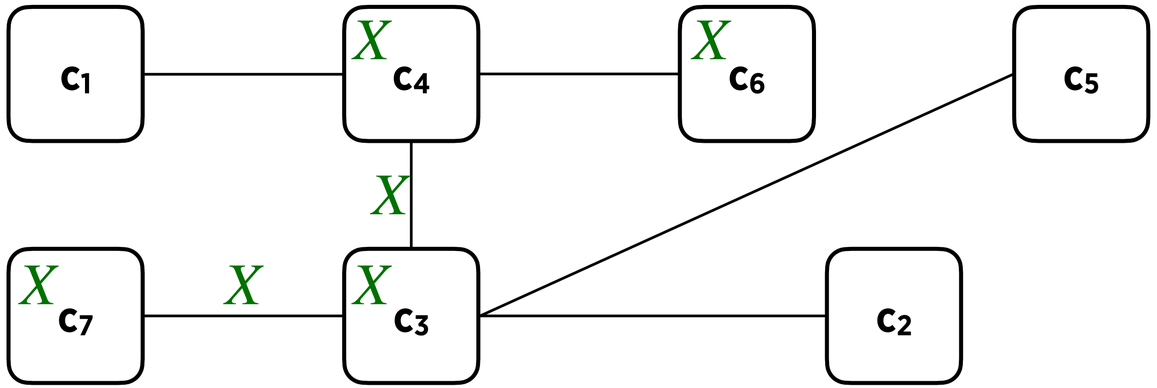
- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$ there exists a unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain X



Running Intersection Property

Cluster graph variant of the property

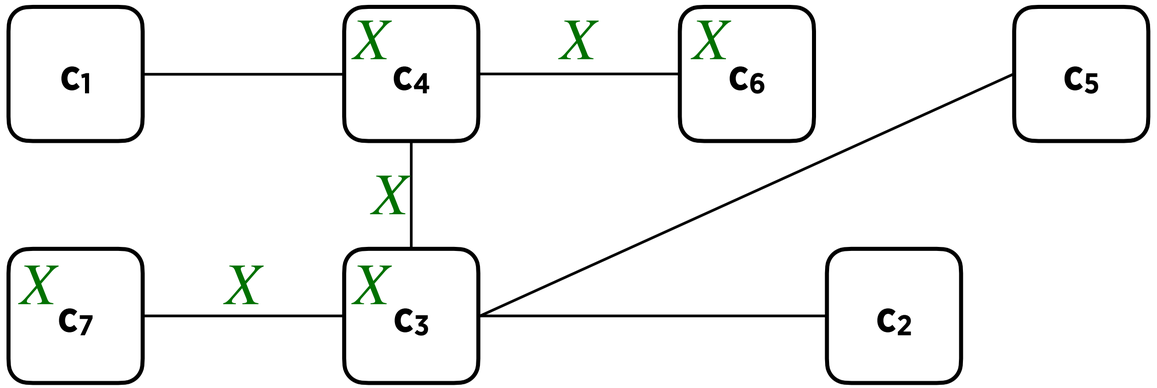
- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$ there exists a unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain X



Running Intersection Property

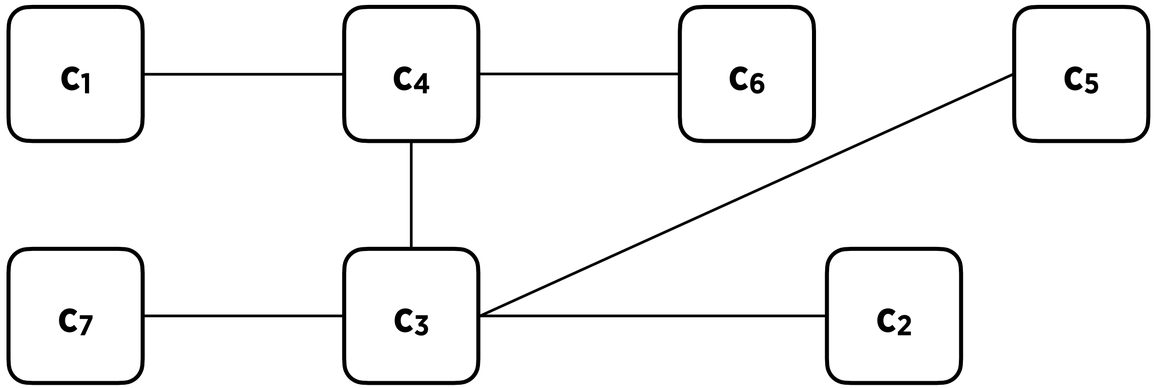
Cluster graph variant of the property

- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$ there exists a unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which all clusters and sepsets contain X



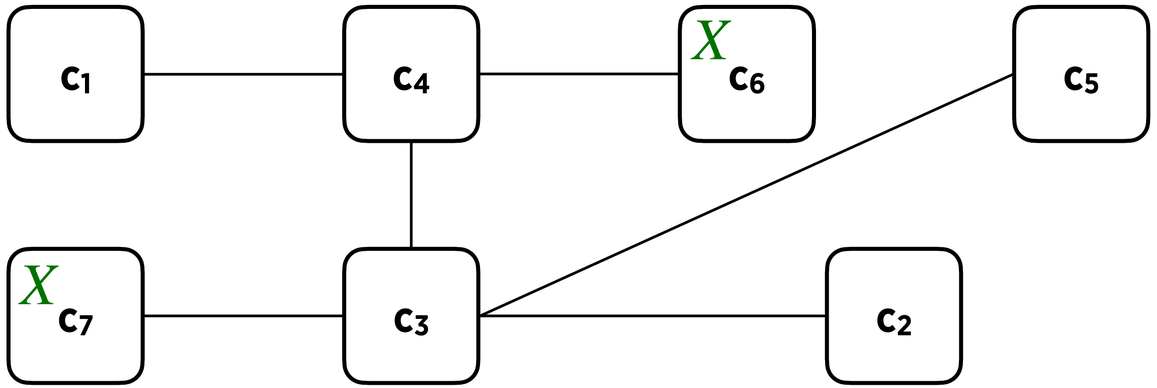
Running Intersection Property

- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$
  in the unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$
  all clusters and sepsets contain X



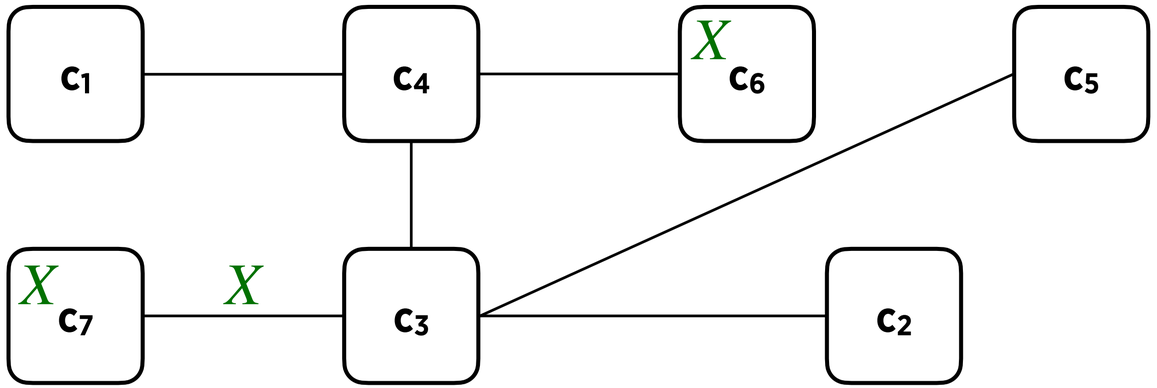
Running Intersection Property

- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$
  in the unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$
  all clusters and sepsets contain X



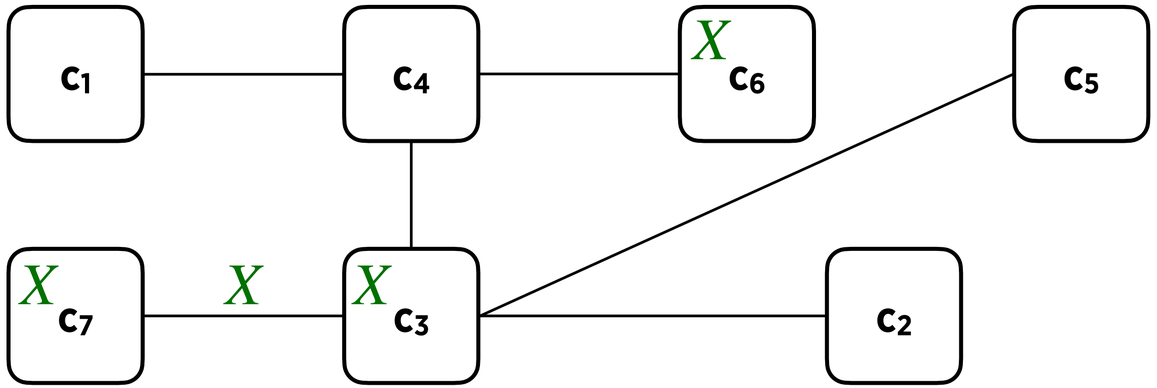
Running Intersection Property

- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$
  in the unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$
  all clusters and sepsets contain X



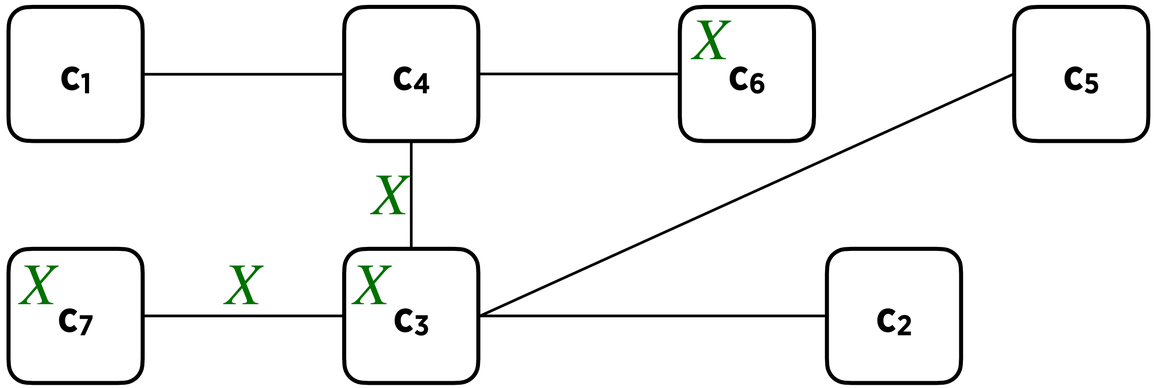
Running Intersection Property

- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$
  in the unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$
  all clusters and sepsets contain X



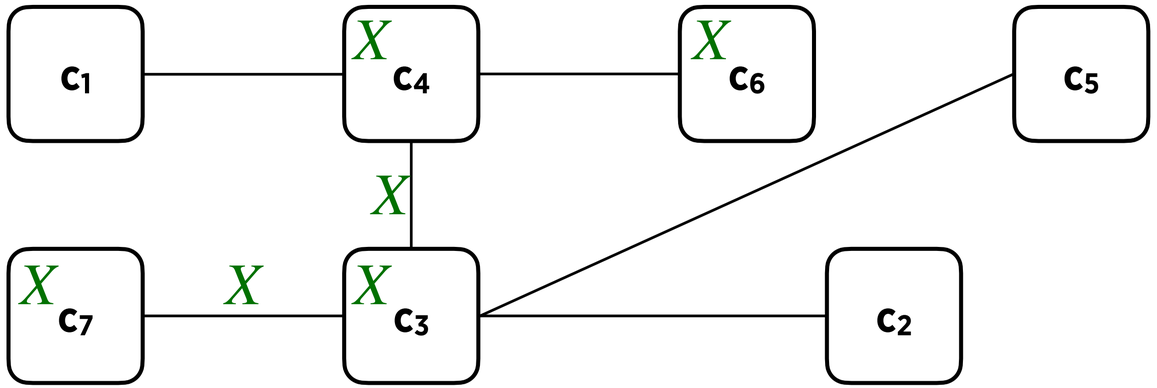
Running Intersection Property

- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$
  in the unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$
  all clusters and sepsets contain X



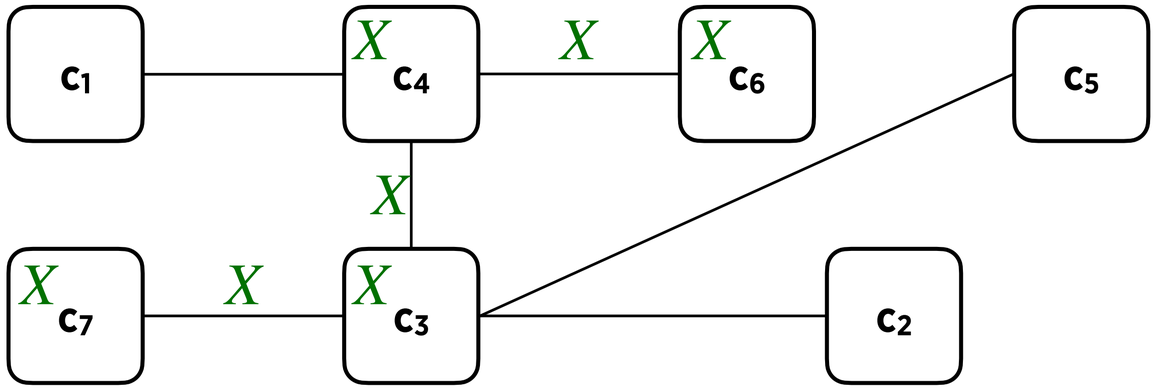
Running Intersection Property

- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$
  in the unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$
  all clusters and sepsets contain X

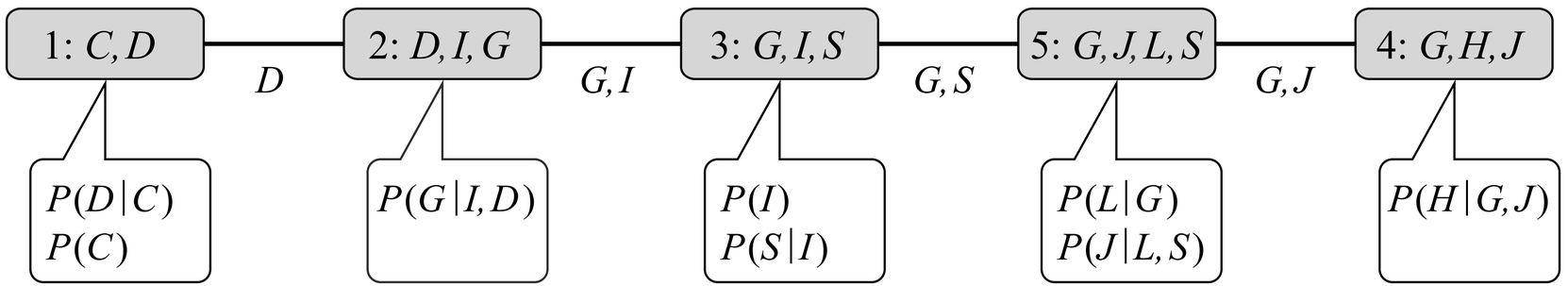


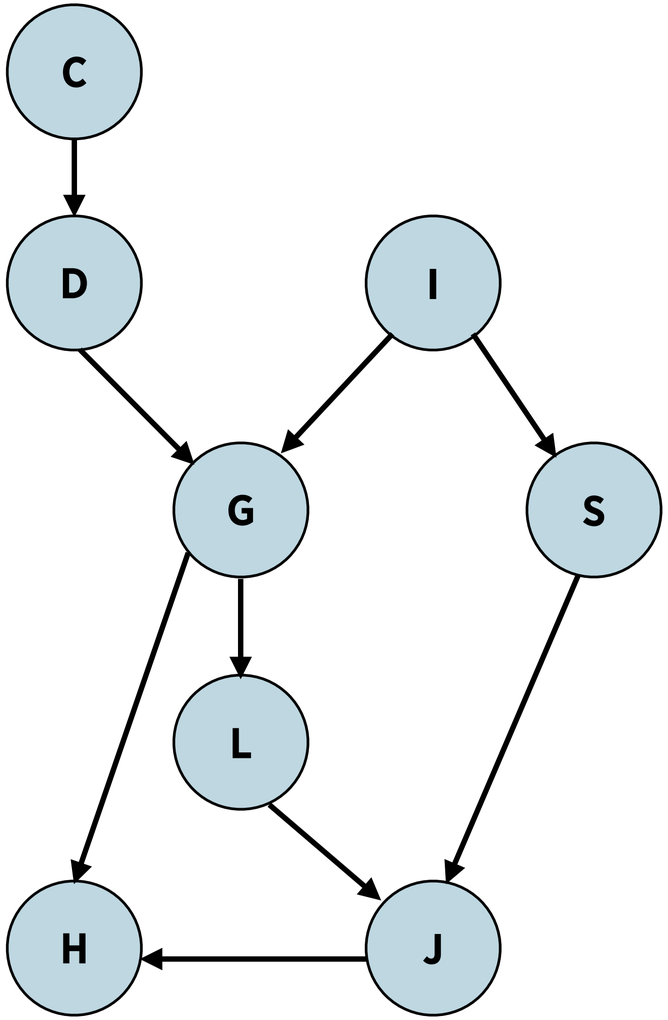
Running Intersection Property

- For each pair of clusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ and variable $X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$
  in the unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$
  all clusters and sepsets contain X



More complex clique tree





Summary properties of BP

Summary properties of BP

- Belief propagation can be run over a tree-structured cluster graph

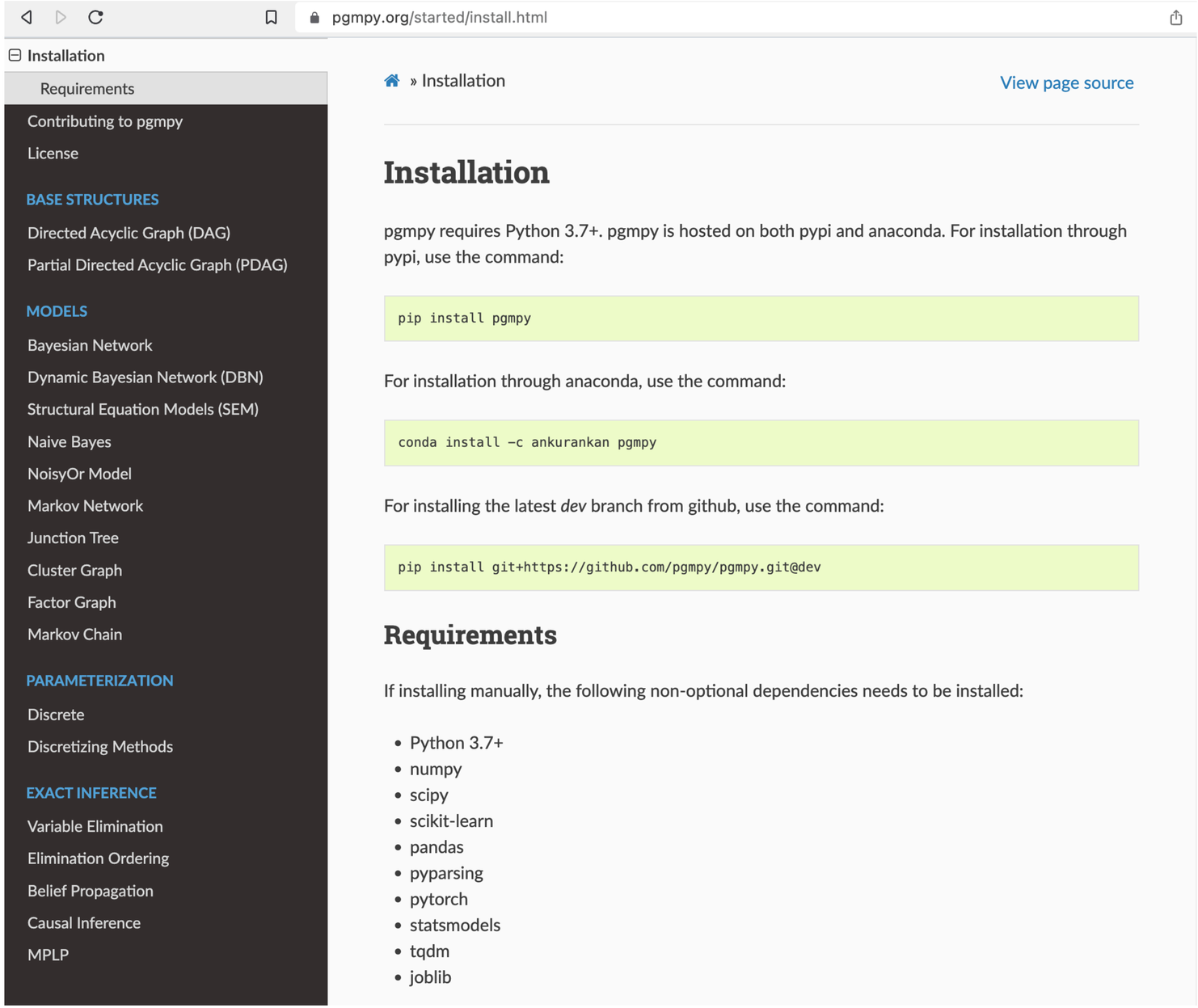
Summary properties of BP

- Belief propagation can be run over a tree-structured cluster graph
- In this case, computation is a variant of variable elimination

Summary properties of BP

- 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

Pgmpy


pgmpy.org/started/install.html
Installation
Requirements
Contributing to pgmpy
License
BASE STRUCTURES
Directed Acyclic Graph (DAG)
Partial Directed Acyclic Graph (PDAG)
MODELS
Bayesian Network
Dynamic Bayesian Network (DBN)
Structural Equation Models (SEM)
Naive Bayes
NoisyOr Model
Markov Network
Junction Tree
Cluster Graph
Factor Graph
Markov Chain
PARAMETERIZATION
Discrete
Discretizing Methods
EXACT INFERENCE
Variable Elimination
Elimination Ordering
Belief Propagation
Causal Inference
MPLP
» Installation
View page source
Installation
pgmpy requires Python 3.7+. pgmpy is hosted on both pypi and anaconda. For installation through pypi, use the command:
pip install pgmpy
For installation through anaconda, use the command:
conda install -c ankurankan pgmpy
For installing the latest dev branch from github, use the command:
pip install git+https://github.com/pgmpy/pgmpy.git@dev
Requirements
If installing manually, the following non-optional dependencies needs to be installed:
Python 3.7+
numpy
scipy
scikit-learn
pandas
pyparsing
pytorch
statsmodels
tqdm
joblib

Bayesian Network

class **pgmpy.models.BayesianNetwork.BayesianNetwork**(*ebunch=None, latents={}*) [source]

Base class for Bayesian Models.

add_cpds(**cpds*) [source]

Add CPD (Conditional Probability Distribution) to the Bayesian Model.

Parameters: **cpds** (*list*, *set*, *tuple* *(array-like)*) – List of CPDs which will be associated with the model

Examples

```
>>> from pgmpy.models import BayesianNetwork
>>> from pgmpy.factors.discrete.CPD import TabularCPD
>>> student = BayesianNetwork([('diff', 'grades'), ('intel', 'grades')])
>>> grades_cpd = TabularCPD('grades', 3, [[0.1,0.1,0.1,0.1,0.1,0.1],
...                                       [0.1,0.1,0.1,0.1,0.1,0.1],
...                                       [0.8,0.8,0.8,0.8,0.8,0.8]],
...                         evidence=['diff', 'intel'], evidence_card=[2, 3])
>>> student.add_cpds(grades_cpd)
```

diff:	easy			hard		
intel:	dumb	avg	smart	dumb	avg	smart
gradeA	0.1	0.1	0.1	0.1	0.1	0.1
gradeB	0.1	0.1	0.1	0.1	0.1	0.1
gradeC	0.8	0.8	0.8	0.8	0.8	0.8

add_edge(*u*, *v*, ***kwargs*) [source]

Add an edge between u and v.

Variable Elimination

class **pgmpy.inference.ExactInference.VariableElimination**(*model*) [source]

induced_graph(*elimination_order*) [source]

Returns the induced graph formed by running Variable Elimination on the network.

Parameters: **elimination_order** (*list*, *array like*) – List of variables in the order in which they are to be eliminated.

Examples

```
>>> import numpy as np
>>> import pandas as pd
>>> from pgmpy.models import BayesianNetwork
>>> from pgmpy.inference import VariableElimination
>>> values = pd.DataFrame(np.random.randint(low=0, high=2, size=(1000, 5)),
...                       columns=['A', 'B', 'C', 'D', 'E'])
>>> model = BayesianNetwork([('A', 'B'), ('C', 'B'), ('C', 'D'), ('B', 'E')])
>>> model.fit(values)
>>> inference = VariableElimination(model)
>>> inference.induced_graph(['C', 'D', 'A', 'B', 'E'])
```

induced_width(*elimination_order*) [source]

Returns the width (integer) of the induced graph formed by running Variable Elimination on the network. The width is the defined as the number of nodes in the largest clique in the graph minus 1.

Parameters: **elimination_order** (*list*, *array like*) – List of variables in the order in which they are to be eliminated.

Examples

```
>>> import numpy as np
>>> import pandas as pd
```