Discrete Inference and Learning
Lecture 1

MVA

2017 – 2018

http://thoth.inrialpes.fr/~alahari/disinflearn

Slides based on material from Stephen Gould, Pushmeet Kohli, Nikos Komodakis, M. Pawan Kumar, Carsten Rother

Optimization problems

- Can be written as

$$\min_x f(x)$$ (optimize an objective function)

$$\text{s.t. } x \in \mathcal{C}$$ (subject to some constraints)

feasible set, containing all x satisfying the constraints

discrete variables

Optimization problems

- Can be written as

$$\min_x f(x) \qquad \text{(optimize an objective function)}$$

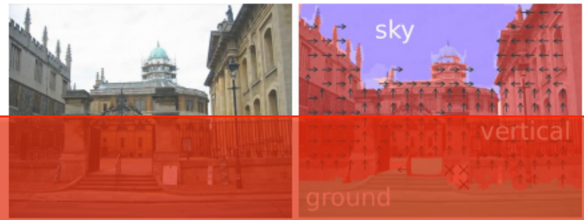
$$\text{s.t. } x \in \mathcal{C} \qquad \text{(subject to some constraints)}$$

- Two main problems in this context
  - Optimize the objective (**inference**)
  - Learn the parameters of f (**learning**)

Optimization problems

- Several applications, e.g., computer vision

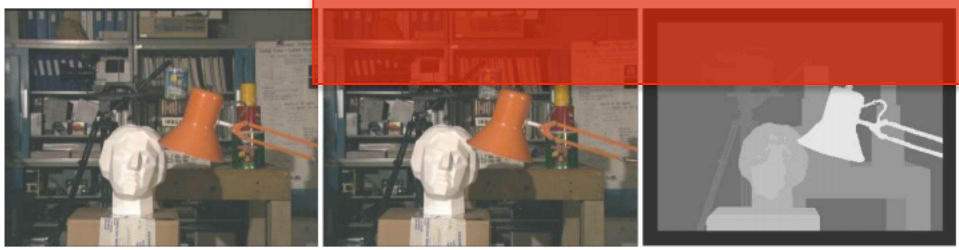
Interactive figure-ground segmentation [Boykov and Jolly, 2001; Boykov and Funka-Lea, 2006]



Surface context [Hoiem et al., 2005]

Semantic labeling [He et al., 2004; Shotton et al., 2006; Gould et al., 2009]

Low-level vision problems

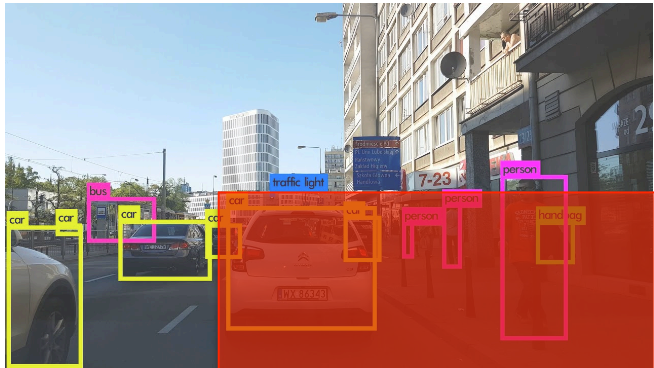


Stereo matching [Kolmogorov and Zabih, 2001; Scharstein and Szeliski, 2002]

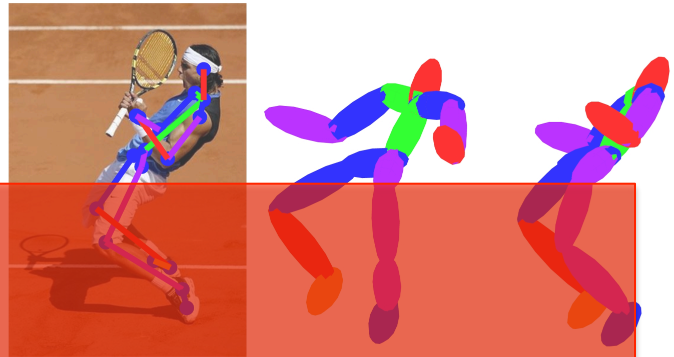
Image denoising [Felzenszwalb and Huttenlocher 2004]

Optimization problems

- Several applications, e.g., computer vision

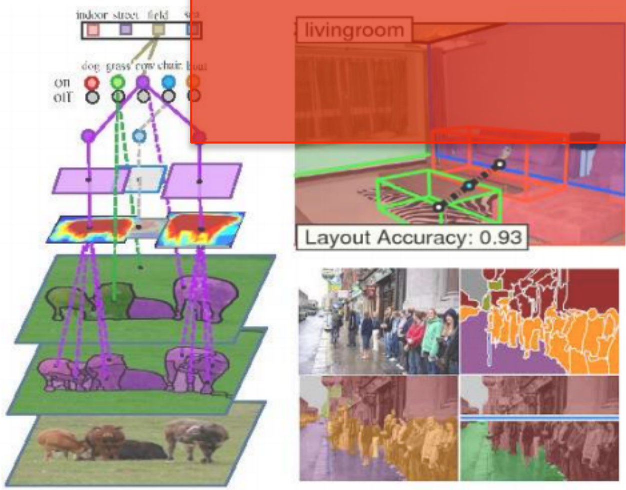


Object detection [Felzenszwalb et al., 2008]



Pose estimation [Akhter and Black, 2015; Ramakrishna et al., 2012]

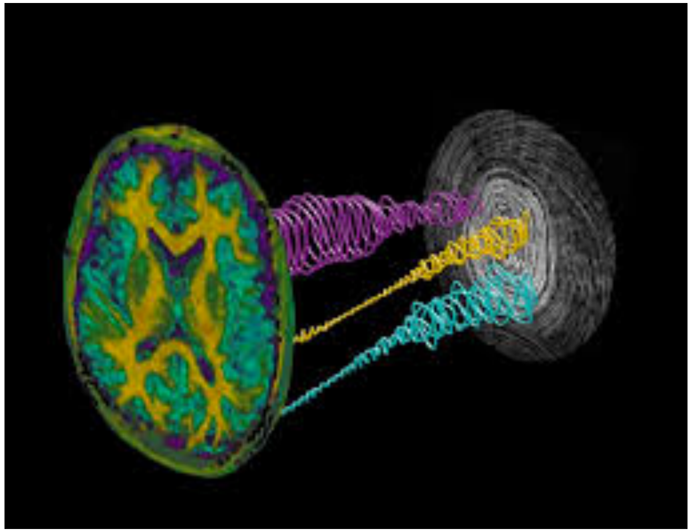
High-level vision problems

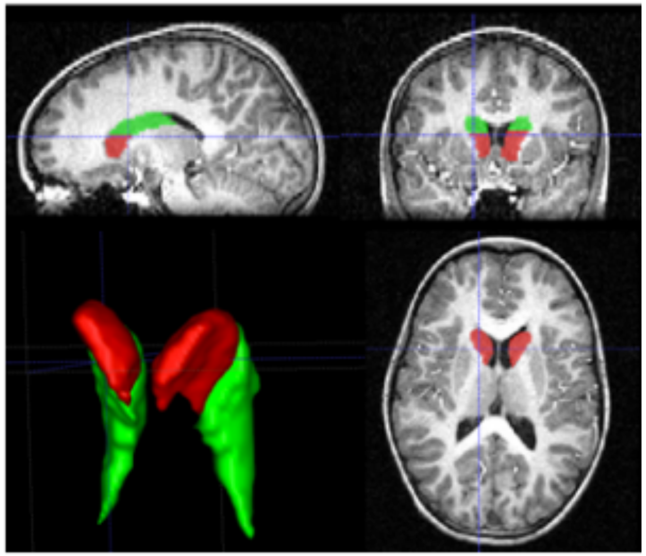


Scene understanding
[Fouhey et al., 2014; Ladicky et al., 2010;
Xiao et al., 2013; Yao et al., 2012]

Optimization problems

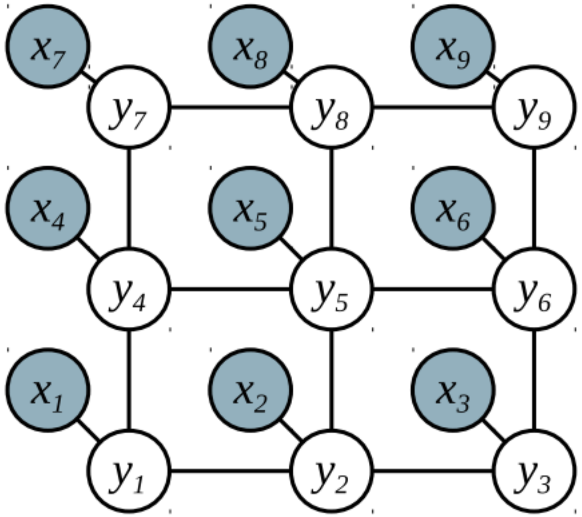
- Several applications, e.g., medical imaging



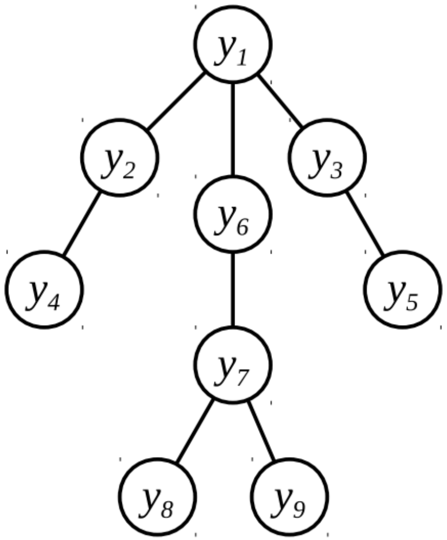


Optimization problems

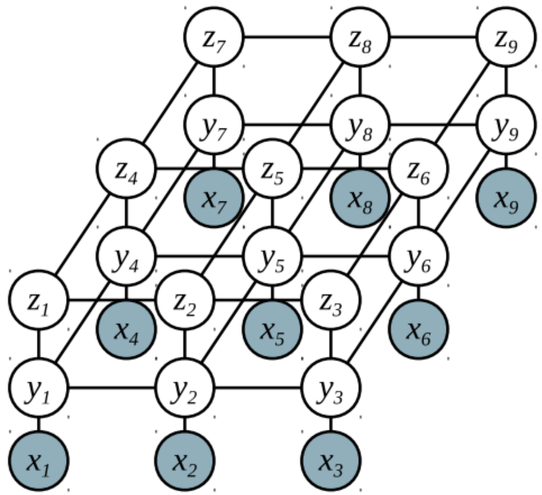
- Inherent in all these problems are graphical models



Pixel labeling



Object detection
Pose estimation



Scene understanding

Conditional Markov Random Fields

- Also known as: Markov networks, undirected graphical models, MRFs
- Note: Not making a distinction between CRFs and MRFs
- $\mathbf{X} \in \mathcal{X}$: observed random variables
- $\mathbf{Y} = (Y_1, \ldots, Y_n) \in \mathcal{Y}$: output random variables
- $\mathbf{Y}_c$ are subset of variables for clique $c \subseteq \{1, \ldots, n\}$
- Define a factored probability distribution

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$$

Partition function $= \sum_{\mathbf{Y} \in \mathcal{Y}} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$

Exponential number of configurations !

Maximum a posteriori (MAP) inference

$$
\begin{aligned}
\mathbf{y}^\star &= \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y} \mid \mathbf{x}) \\
&= \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \frac{1}{Z(\mathbf{X})} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X}) \\
&= \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \log \left(\frac{1}{Z(\mathbf{X})} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X}) \right) \\
&= \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \sum_c \log \Psi_c(\mathbf{Y}_c; \mathbf{X}) - \log Z(\mathbf{X}) \\
&= \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \sum_c \log \Psi_c(\mathbf{Y}_c; \mathbf{X})
\end{aligned}
$$

$\log \Psi_c(\mathbf{Y}_c; \mathbf{X}) \rightarrow -E(\mathbf{Y}; \mathbf{X})$

Maximum a posteriori (MAP) inference

$$\mathbf{y}^\star = \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}}\, P(\mathbf{y} \mid \mathbf{x}) = \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}} \sum_c \log \Psi_c(\mathbf{Y}_c; \mathbf{X})$$

$$= \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmin}}\, E(\mathbf{y}; \mathbf{x})$$

MAP inference ⇔ Energy minimization

The energy function is $E(\mathbf{Y}; \mathbf{X}) = \sum_c \psi_c(\mathbf{Y}_c; \mathbf{X})$

where $\psi_c(\cdot) = -\log \Psi_c(\cdot)$

Clique potential

Clique potentials

- Defines a mapping from an assignment of random variables to a real number

$$\psi_c : \mathcal{Y}_c \times \mathcal{X} \to \mathbb{R}$$

- Encodes a preference for assignments to the random variables (lower is better)

- Parameterized as $\psi_c(\mathbf{y}_c; \mathbf{x}) = \mathbf{w}_c^T \phi_c(\mathbf{y}_c; \mathbf{x})$

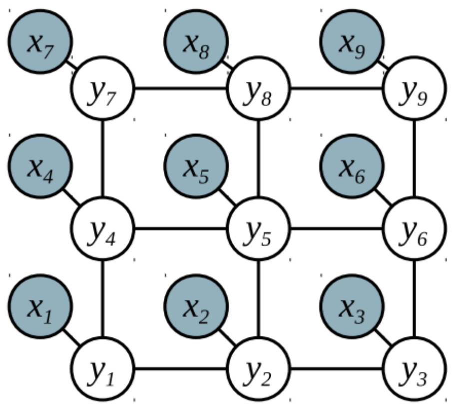
Parameters

Clique potentials

- Arity

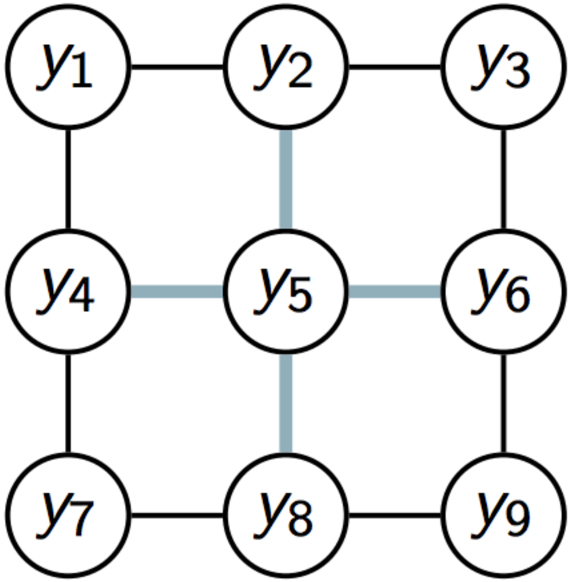
$$E(\mathbf{y};\mathbf{x}) = \sum_{c} \psi_c(\mathbf{y}_c;\mathbf{x})$$

$$= \underbrace{\sum_{i\in\mathcal{V}} \psi_i^U(y_i;\mathbf{x})}_{\text{unary}} + \underbrace{\sum_{ij\in\mathcal{E}} \psi_{ij}^P(y_i,y_j;\mathbf{x})}_{\text{pairwise}} + \underbrace{\sum_{c\in\mathcal{C}} \psi_c^H(\mathbf{y}_c;\mathbf{x})}_{\text{higher-order}}.$$

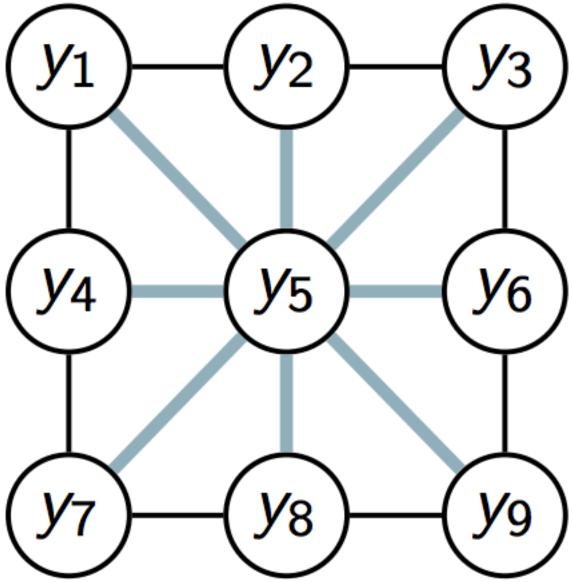


Clique potentials

- Arity

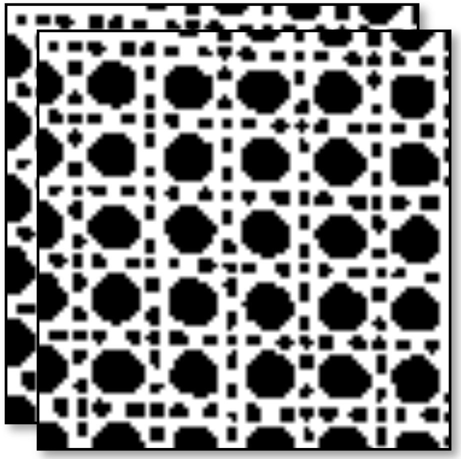


4-connected, $\mathcal{N}_4$

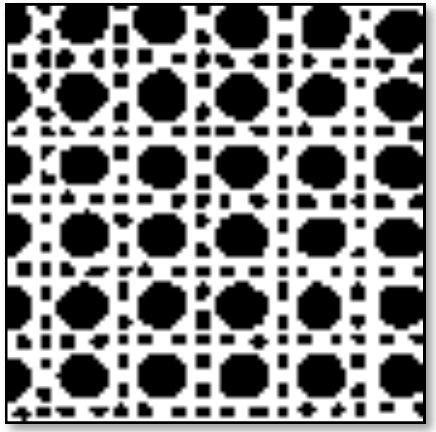


8-connected, $\mathcal{N}_8$

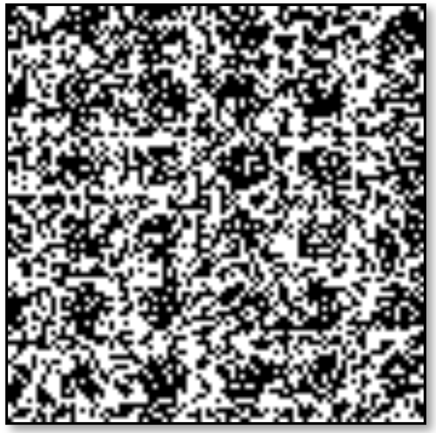
Reason 1: Texture modelling



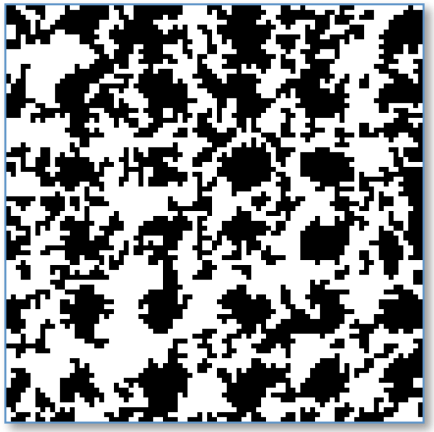
Training images



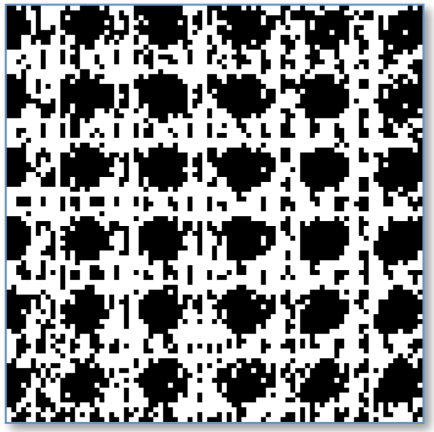
Test image



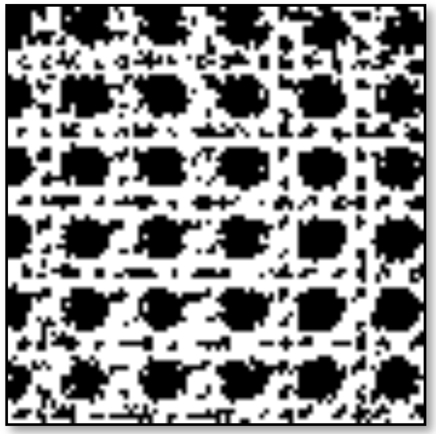
Test image (60% Noise)



Result MRF
4-connected
(neighbours)

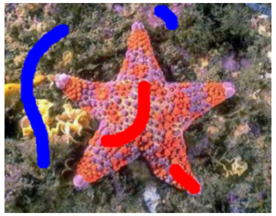


Result MRF
4-connected

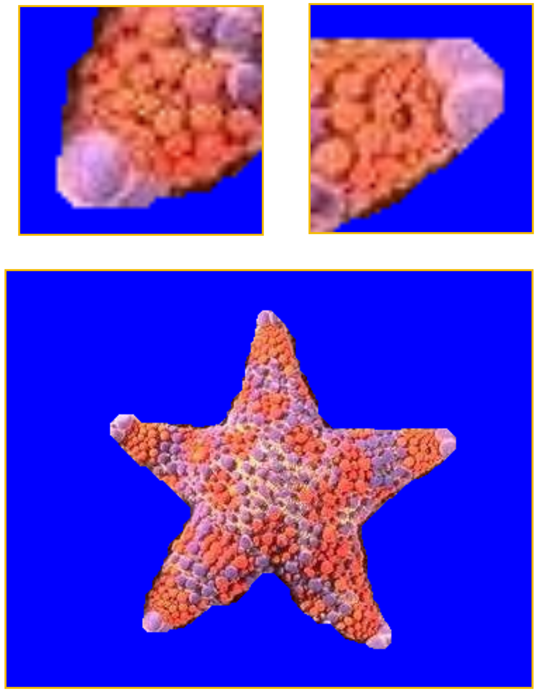


Result MRF
9-connected
(7 attractive; 2 repulsive)

Reason2: Discretization artefacts



4-connected
Euclidean



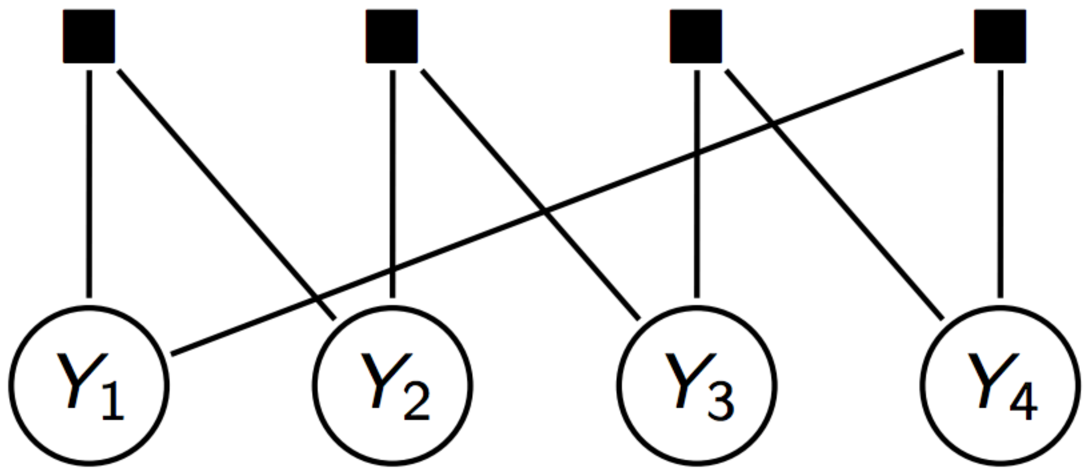
8-connected
Euclidean

[Boykov et al. ‘03; ‘05]

Graphical representation

- Example

$$E(\mathbf{y}) = \psi(y_1, y_2) + \psi(y_2, y_3) + \psi(y_3, y_4) + \psi(y_4, y_1)$$

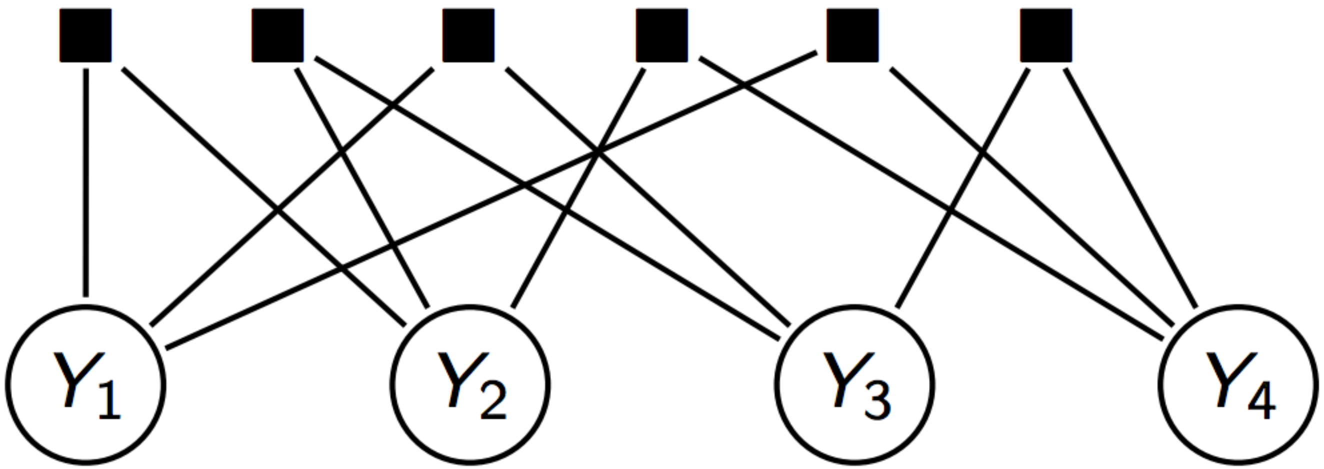


factor graph

Graphical representation

- Example

$$E(\mathbf{y}) = \sum_{i,j} \psi(y_i, y_j)$$

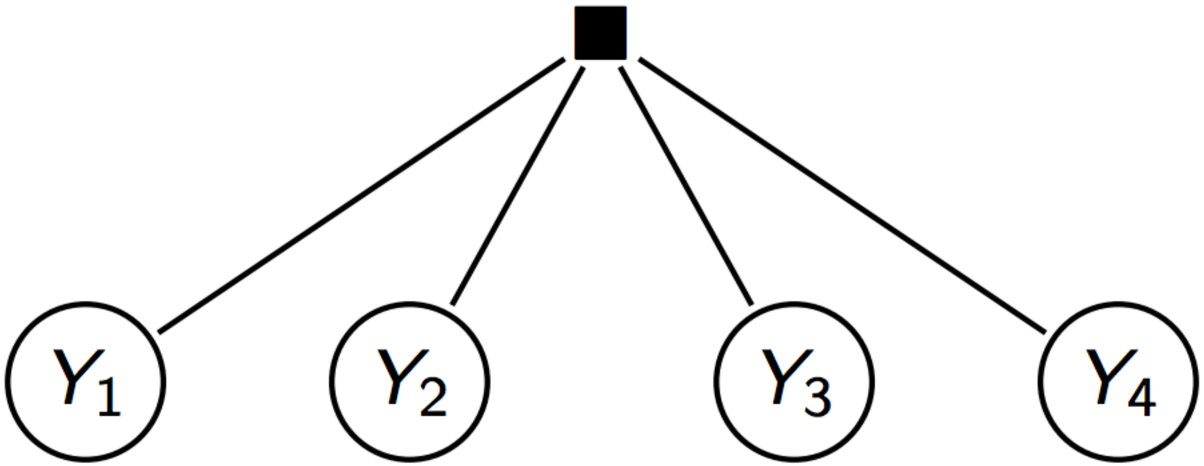


factor graph

Graphical representation

- Example

$$E(\mathbf{y}) = \psi(y_1, y_2, y_3, y_4)$$



factor graph

A Computer Vision Application

Binary Image Segmentation

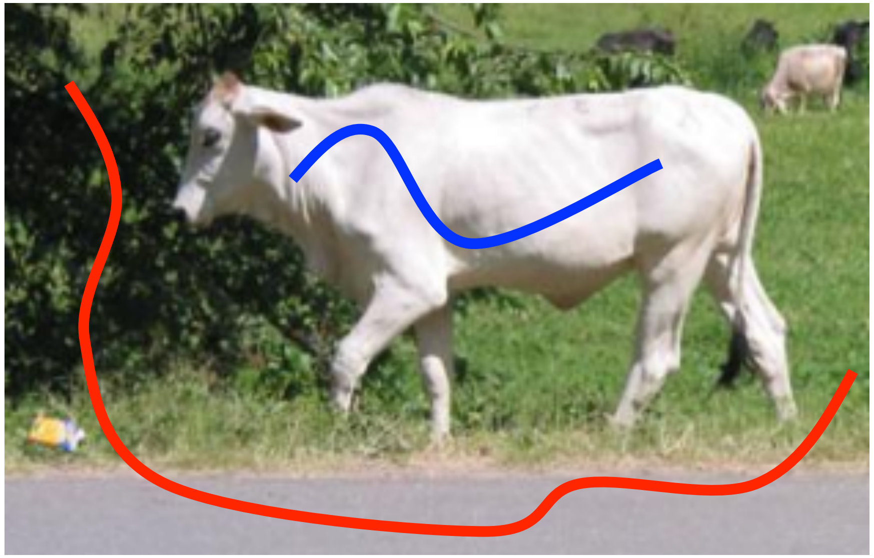
How ?

Cost function   Models *our* knowledge about natural images

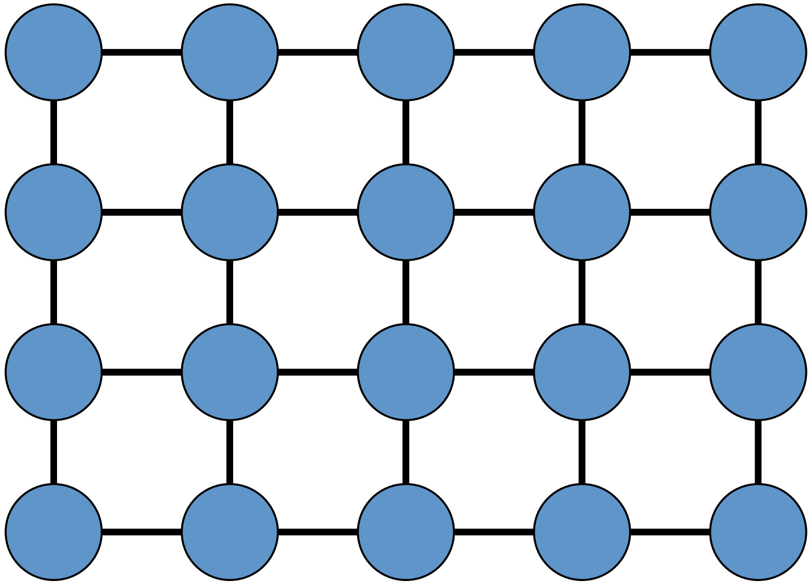
Optimize cost function to obtain the segmentation

A Computer Vision Application

Binary Image Segmentation



Object - white, Background - green/grey



Graph G = (V,E)

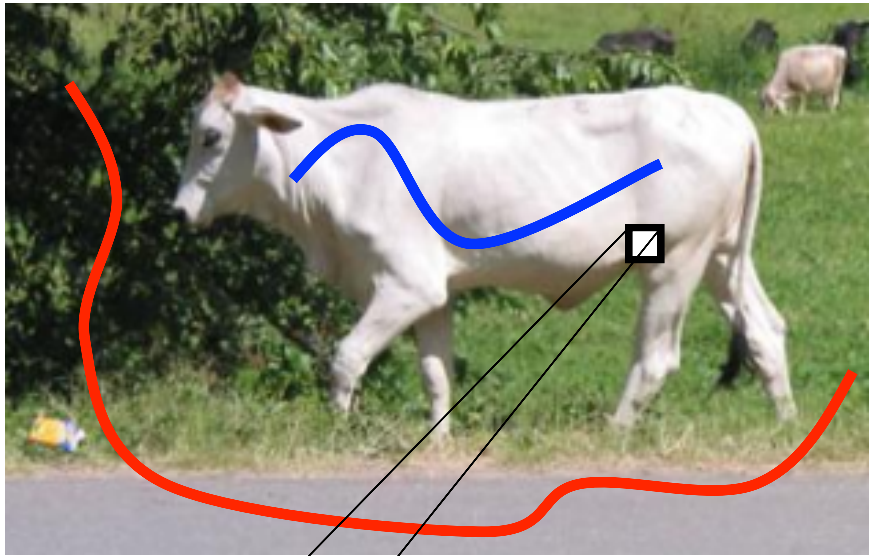
Each vertex corresponds to a pixel

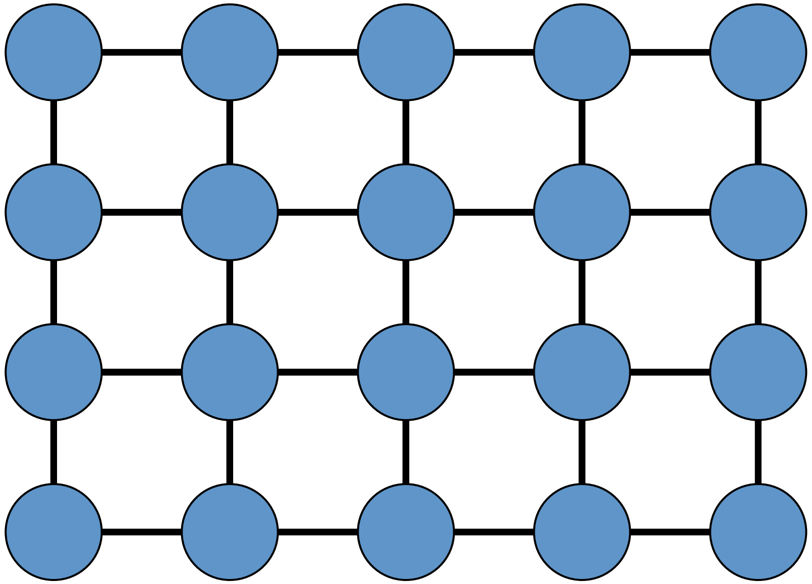
Edges define a 4-neighbourhood *grid* graph

Assign a label to each vertex from L = {obj,bkg}

A Computer Vision Application

Binary Image Segmentation





Object - white, Background - green/grey

Graph G = (V,E)

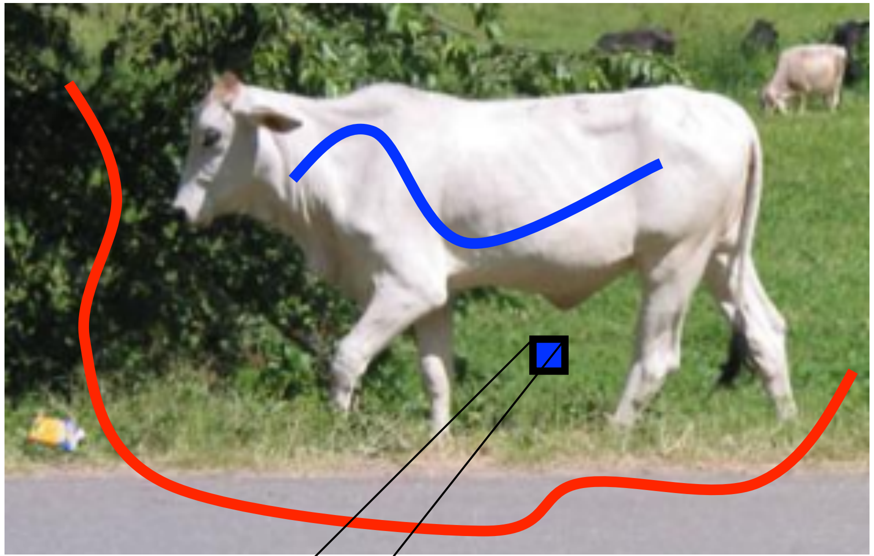
Cost of a labelling f : V ➔ L

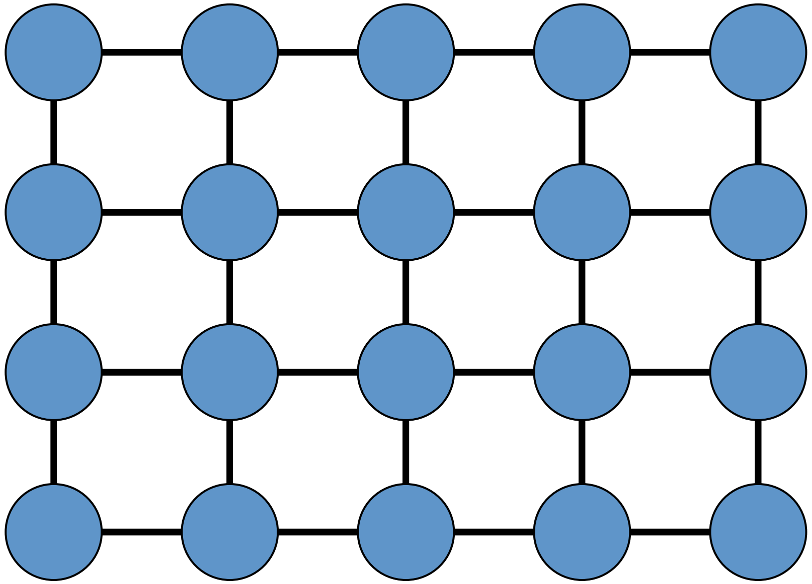
Per Vertex Cost

Cost of label 'obj' low Cost of label 'bkg' high

A Computer Vision Application

Binary Image Segmentation



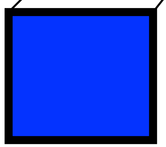


Object - white, Background - green/grey

Graph G = (V,E)

Cost of a labelling f : V ➔ L

Per Vertex Cost

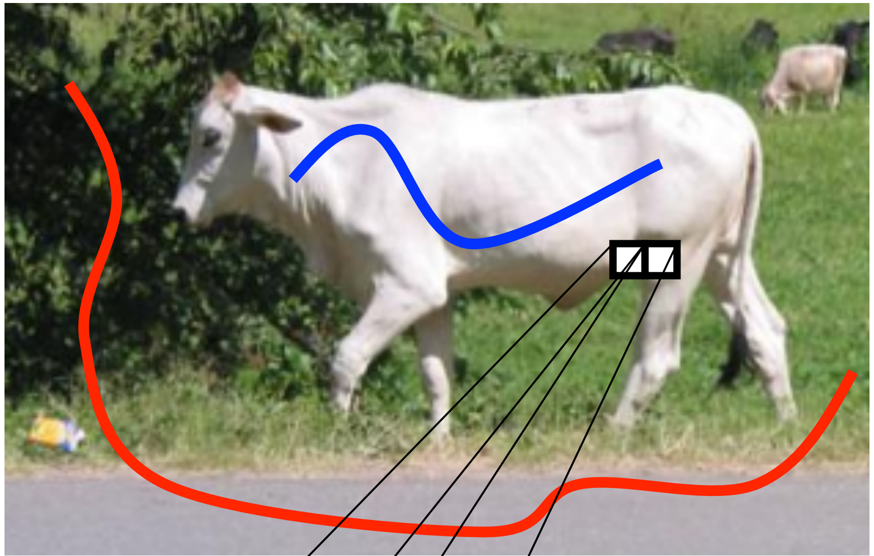


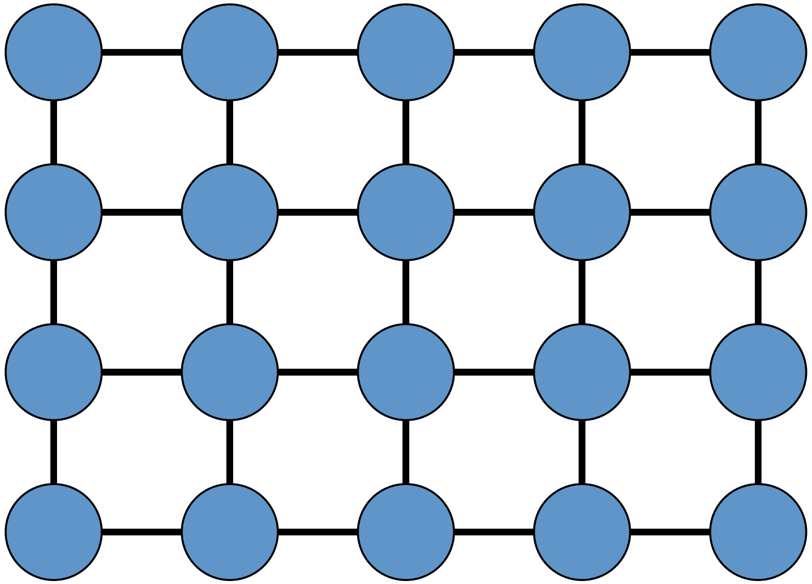
Cost of label 'obj' high Cost of label 'bkg' low

UNARY COST

A Computer Vision Application

Binary Image Segmentation



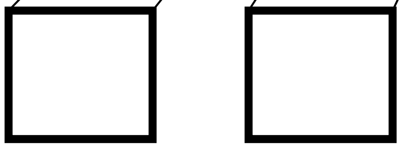


Object - white, Background - green/grey

Graph G = (V,E)

Cost of a labelling f : V ➔ L

Per Edge Cost

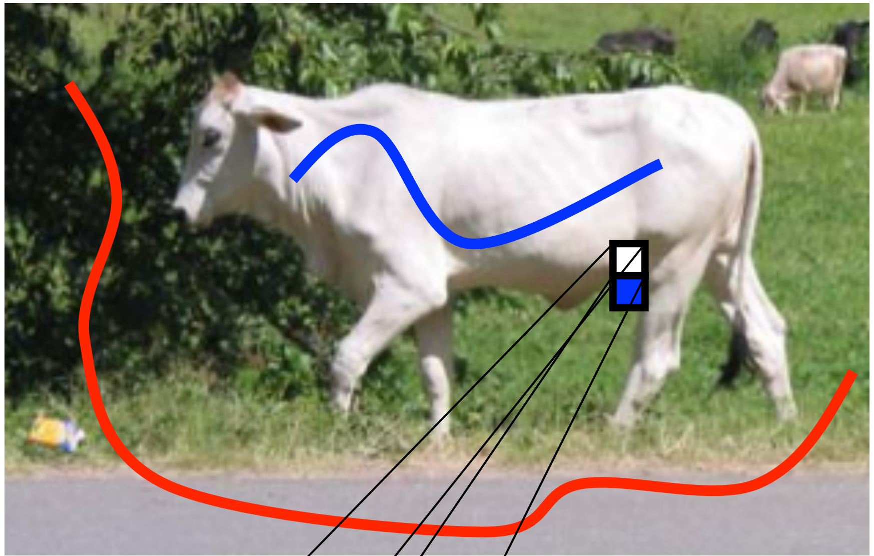


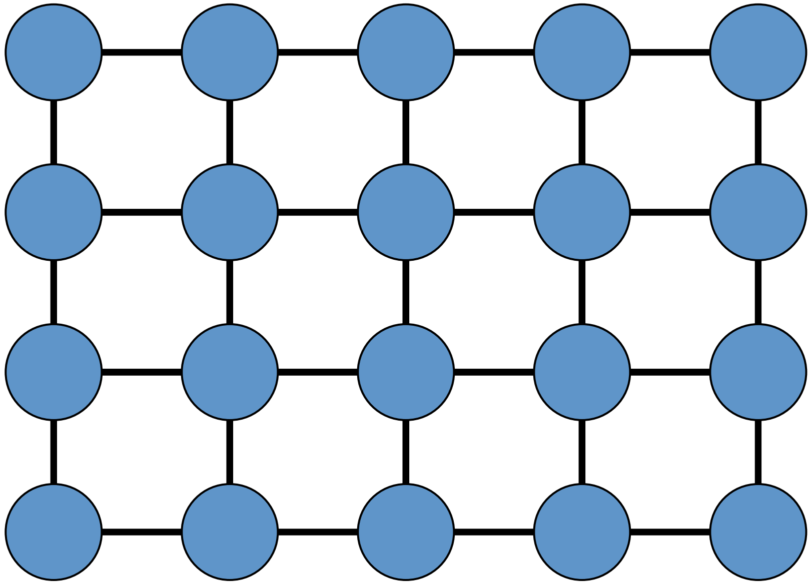
Cost of same label low

Cost of different labels high

A Computer Vision Application

Binary Image Segmentation



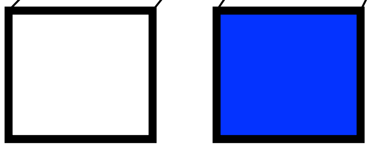


Object - white, Background - green/grey

Graph G = (V,E)

Cost of a labelling f : V ➔ L

Per Edge Cost



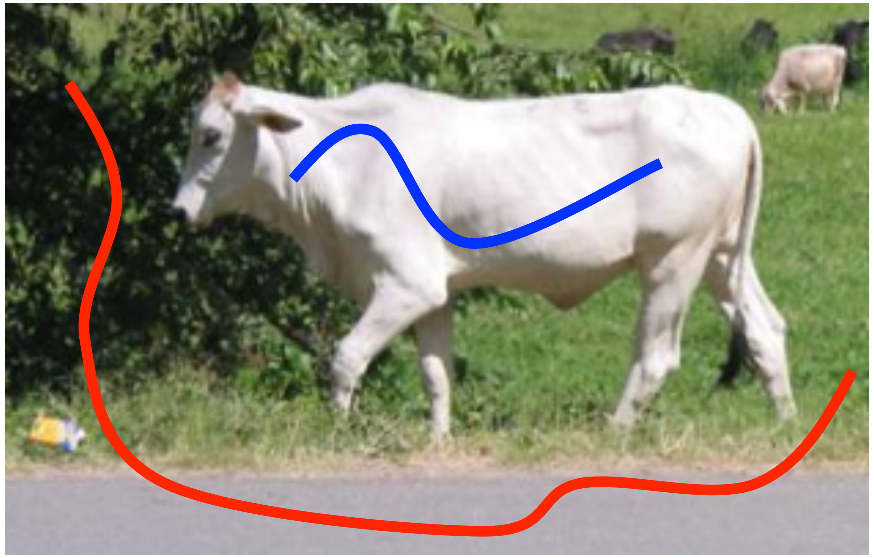
Cost of same label high

Cost of different labels low

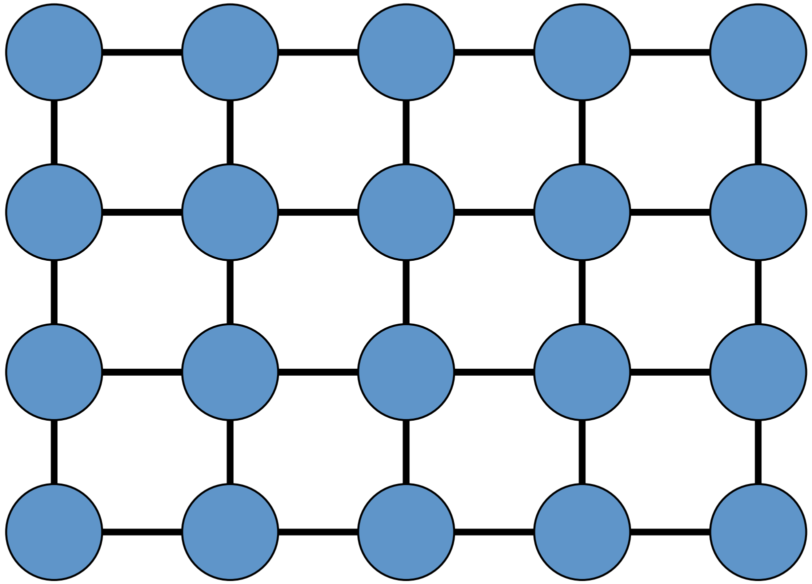
PAIRWISE COST

A Computer Vision Application

Binary Image Segmentation



Object - white, Background - green/grey

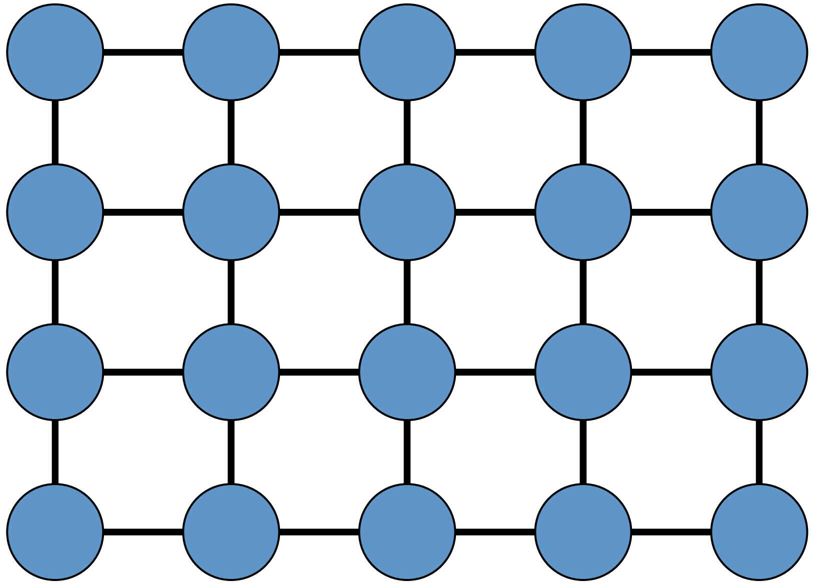


Graph G = (V,E)

Problem: Find the labelling with minimum cost f*

A Computer Vision Application

Binary Image Segmentation

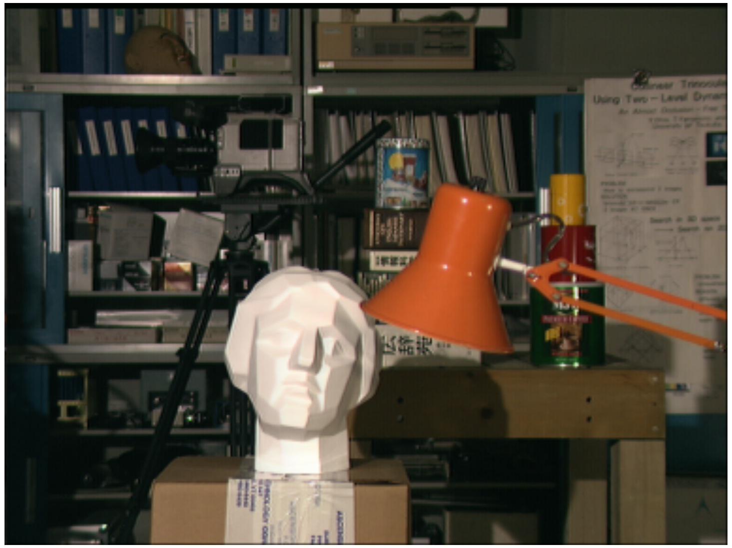


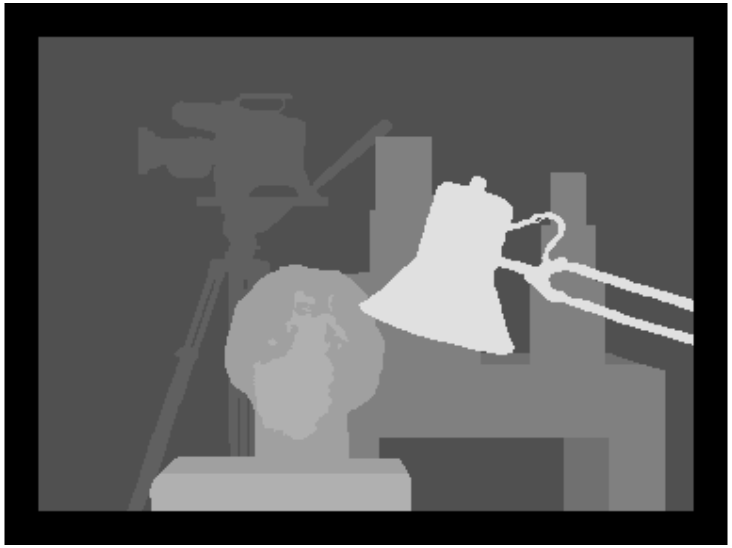
Graph $G = (V,E)$

Problem: Find the labelling with minimum cost f*

Another Computer Vision Application

Stereo Correspondence





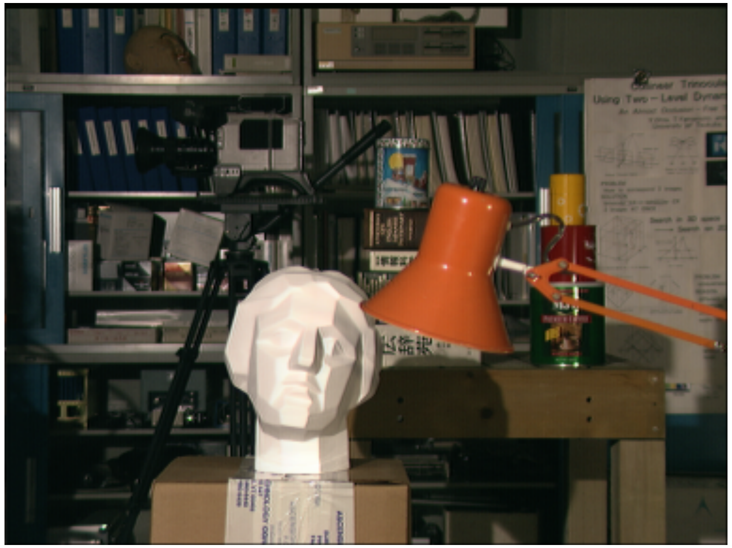
Disparity Map

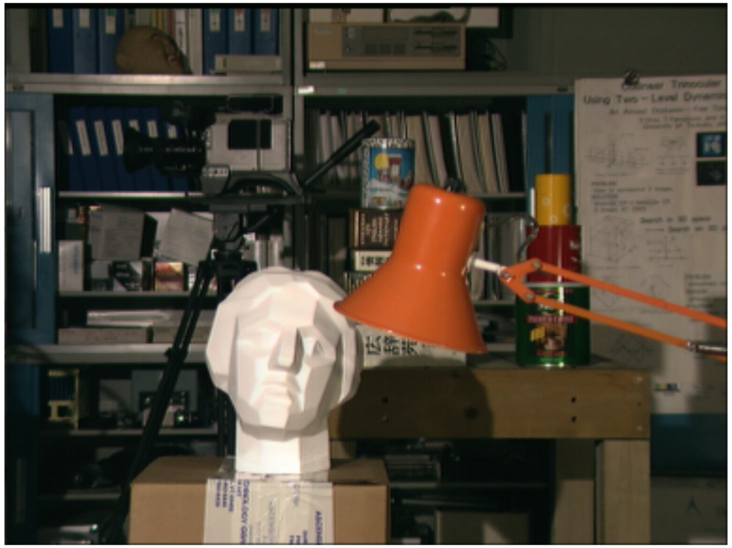
How ?

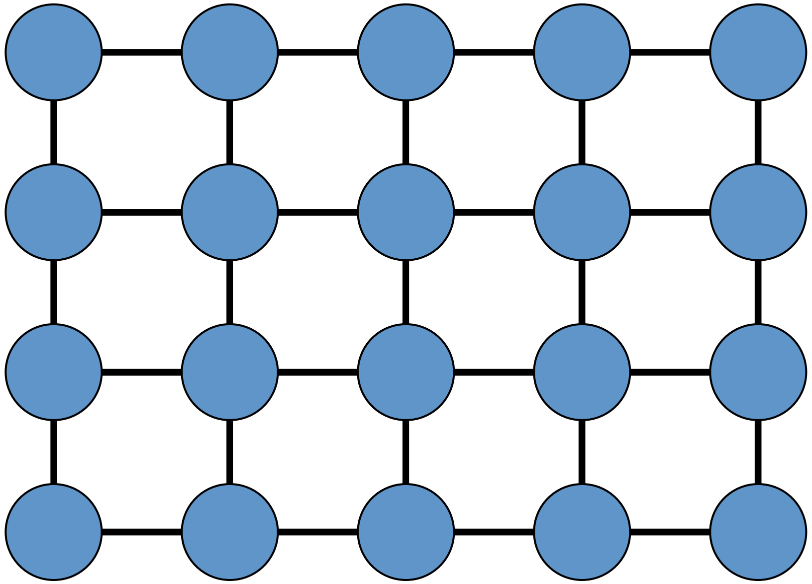
Minimizing a cost function

Another Computer Vision Application

Stereo Correspondence







Graph G = (V,E)

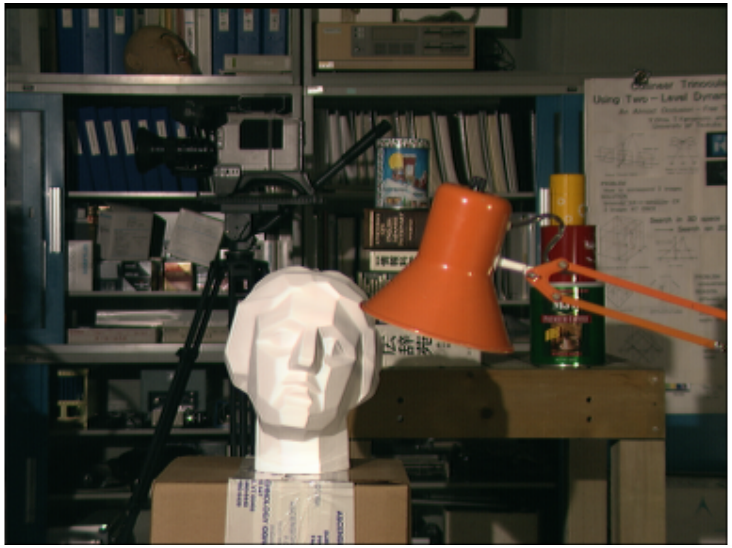
Vertex corresponds to a pixel

Edges define grid graph

L = {disparities}

Another Computer Vision Application

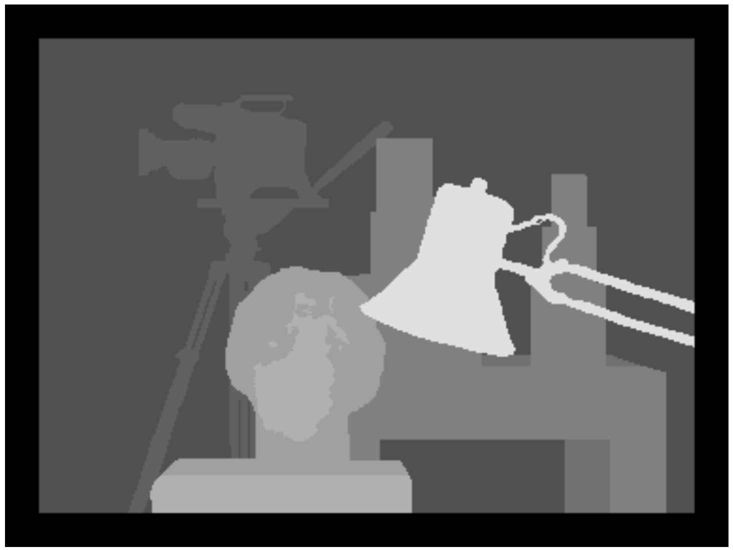
Stereo Correspondence



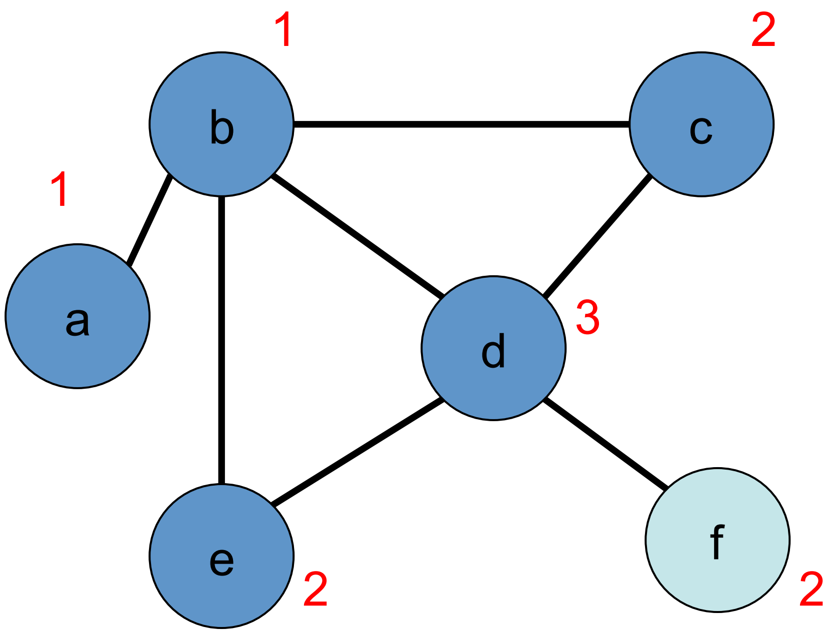
Cost of labelling f:

Unary cost + Pairwise Cost

Find minimum cost f*



The General Problem



Graph G = (V, E)

Discrete label set L = {1,2,…,h}

Assign a label to each vertex
f: V ➜ L

Cost of a labelling Q(f)

Unary Cost   Pairwise Cost

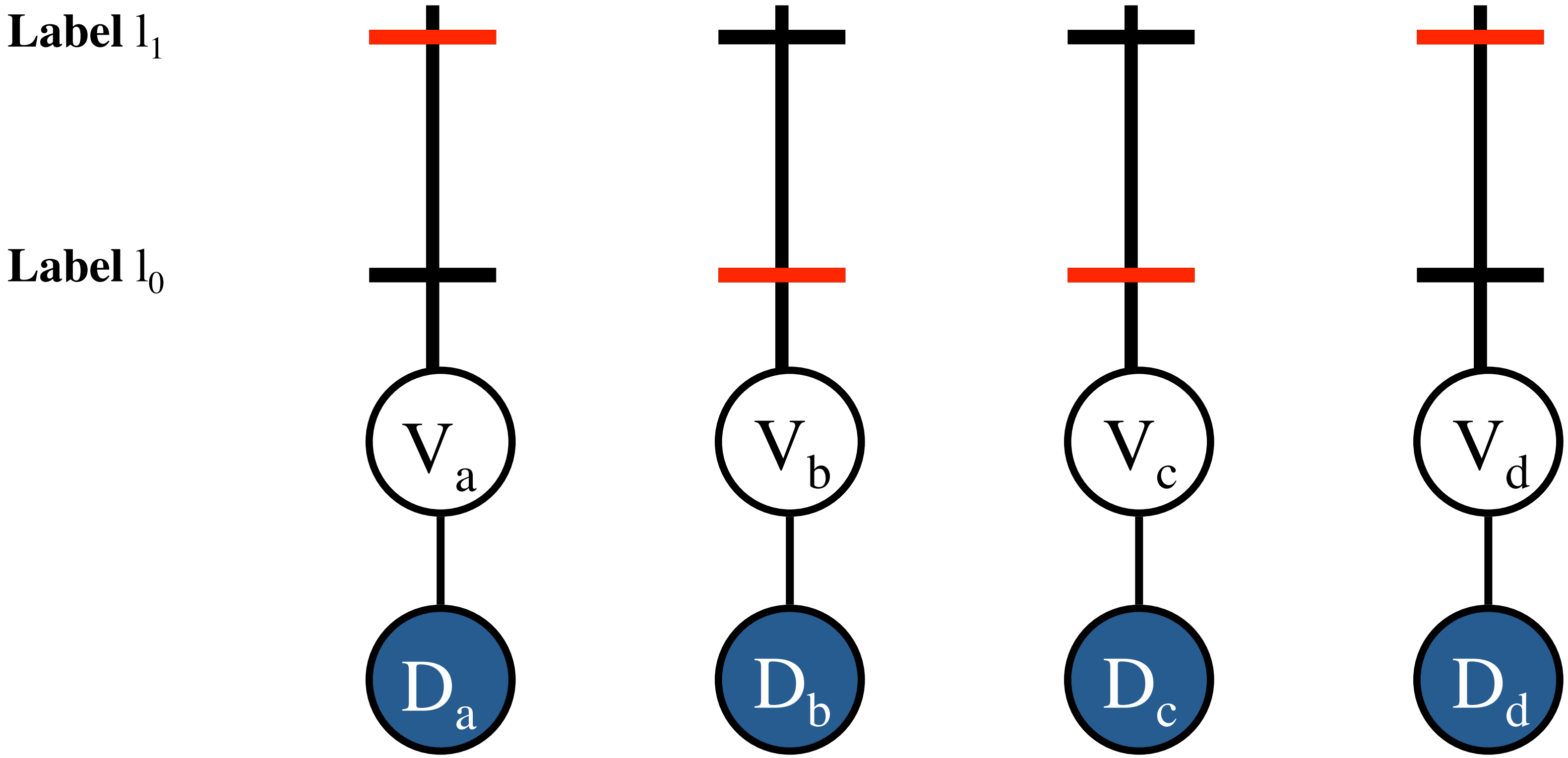
Find f* = arg min Q(f)

Overview

- Basics: problem formulation
  - Energy Function
  - MAP Estimation
  - Computing min-marginals
  - Reparameterization

- Solutions
  - Belief Propagation and related methods [Lecture 1]
  - Graph cuts [Lecture 2]

Energy Function

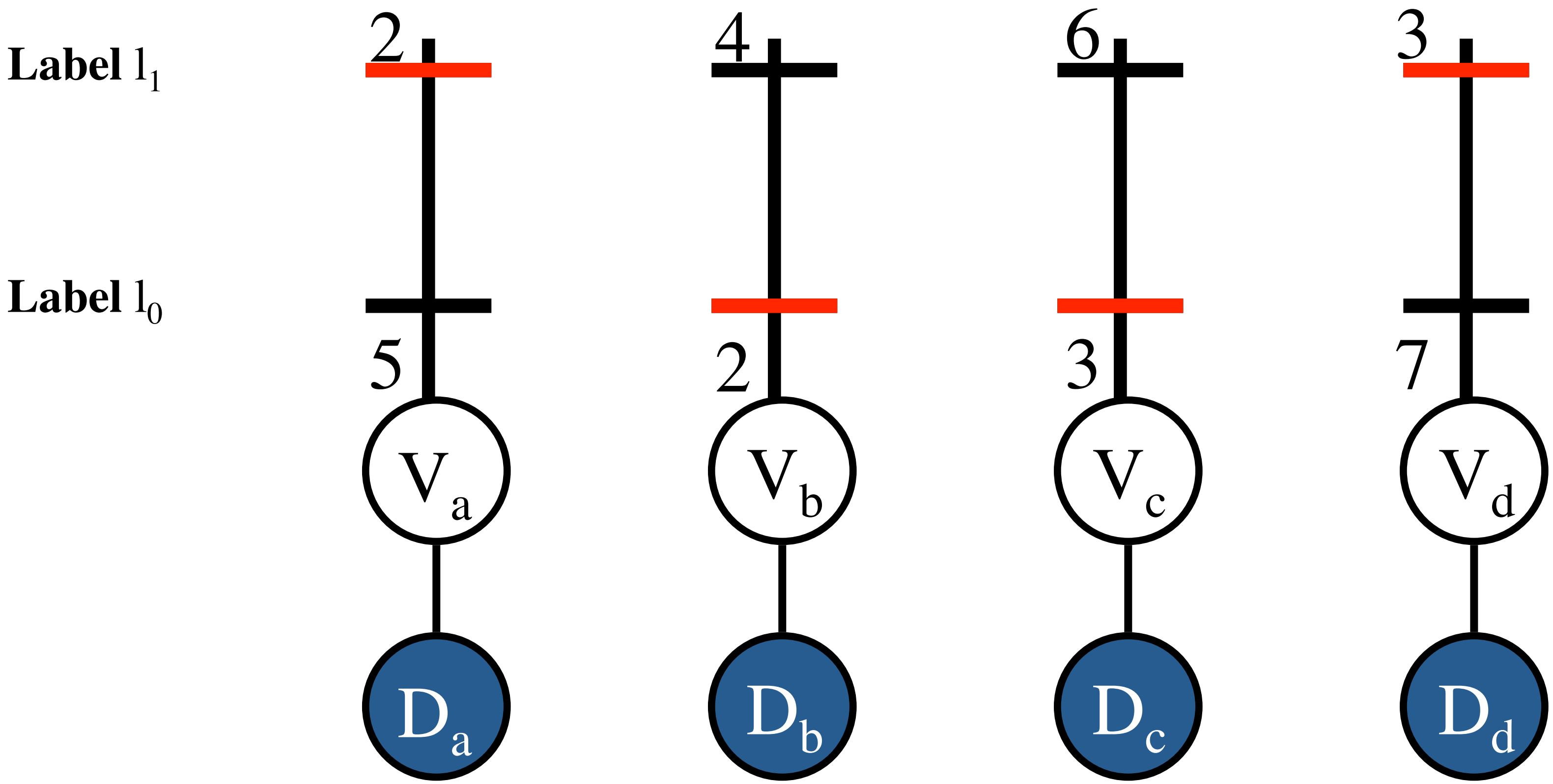


Random Variables V = $\{V_a, V_b, \ldots.\}$

Labels L = $\{l_0, l_1, \ldots.\}$ Data D

Labelling f: {a, b, …. } ➔ {0,1, …}

Energy Function



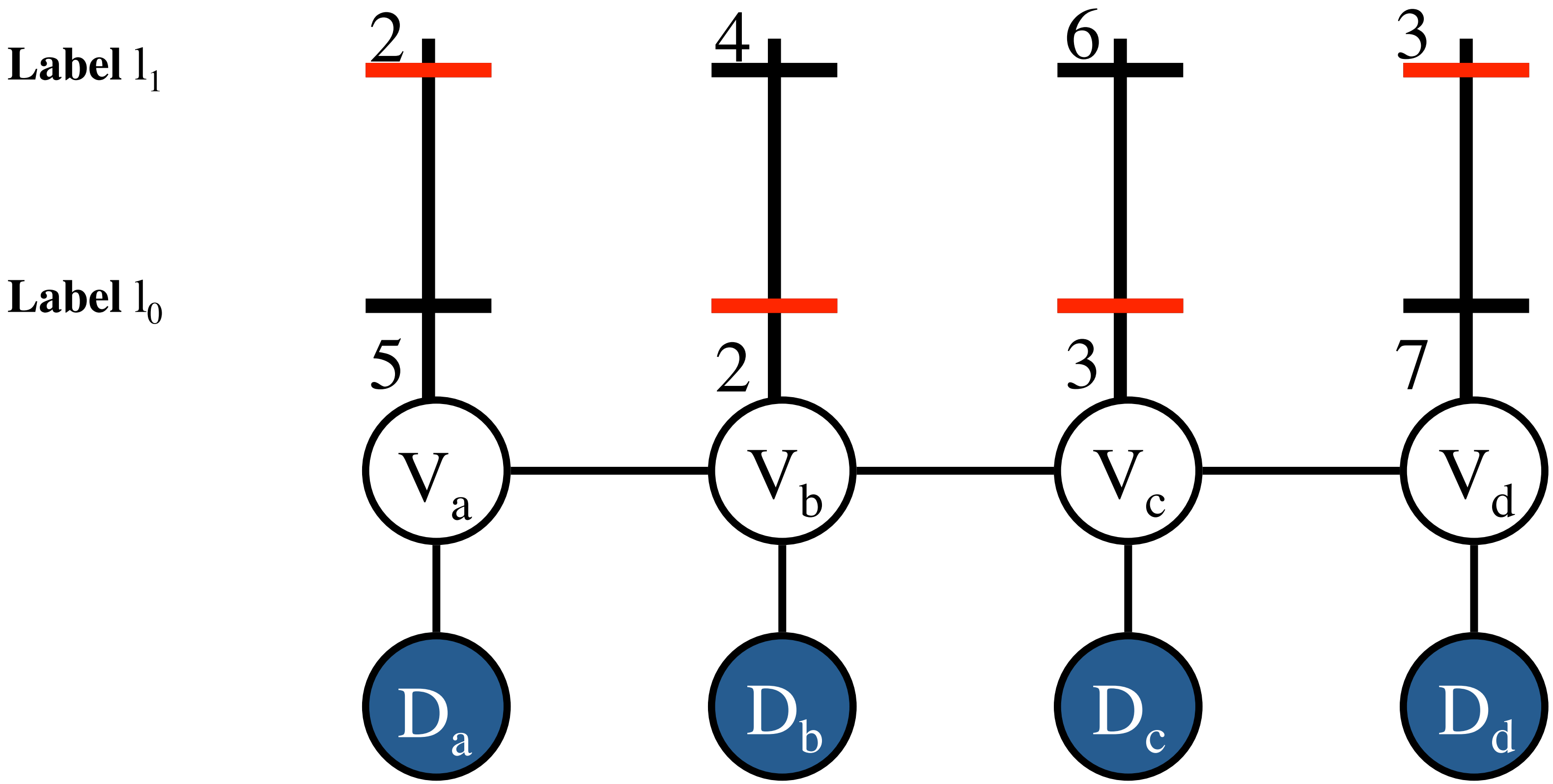
$$Q(f) = \sum_a \theta_{a;f(a)}$$

Unary Potential

Easy to minimize

Neighbourhood

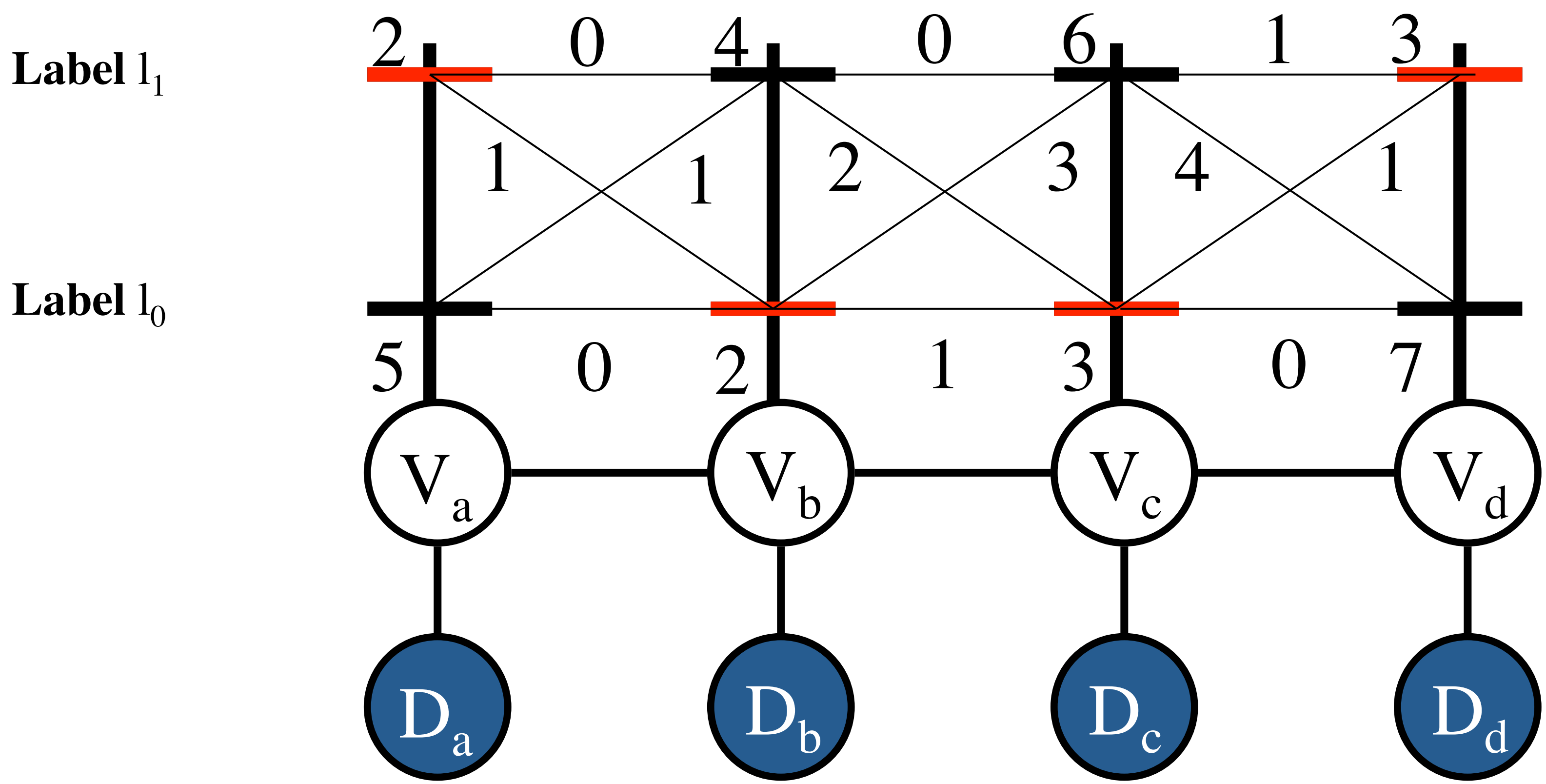
Energy Function



E : $(a,b) \in E$ iff V_a and V_b are neighbours

$E = \{ (a,b) , (b,c) , (c,d) \}$

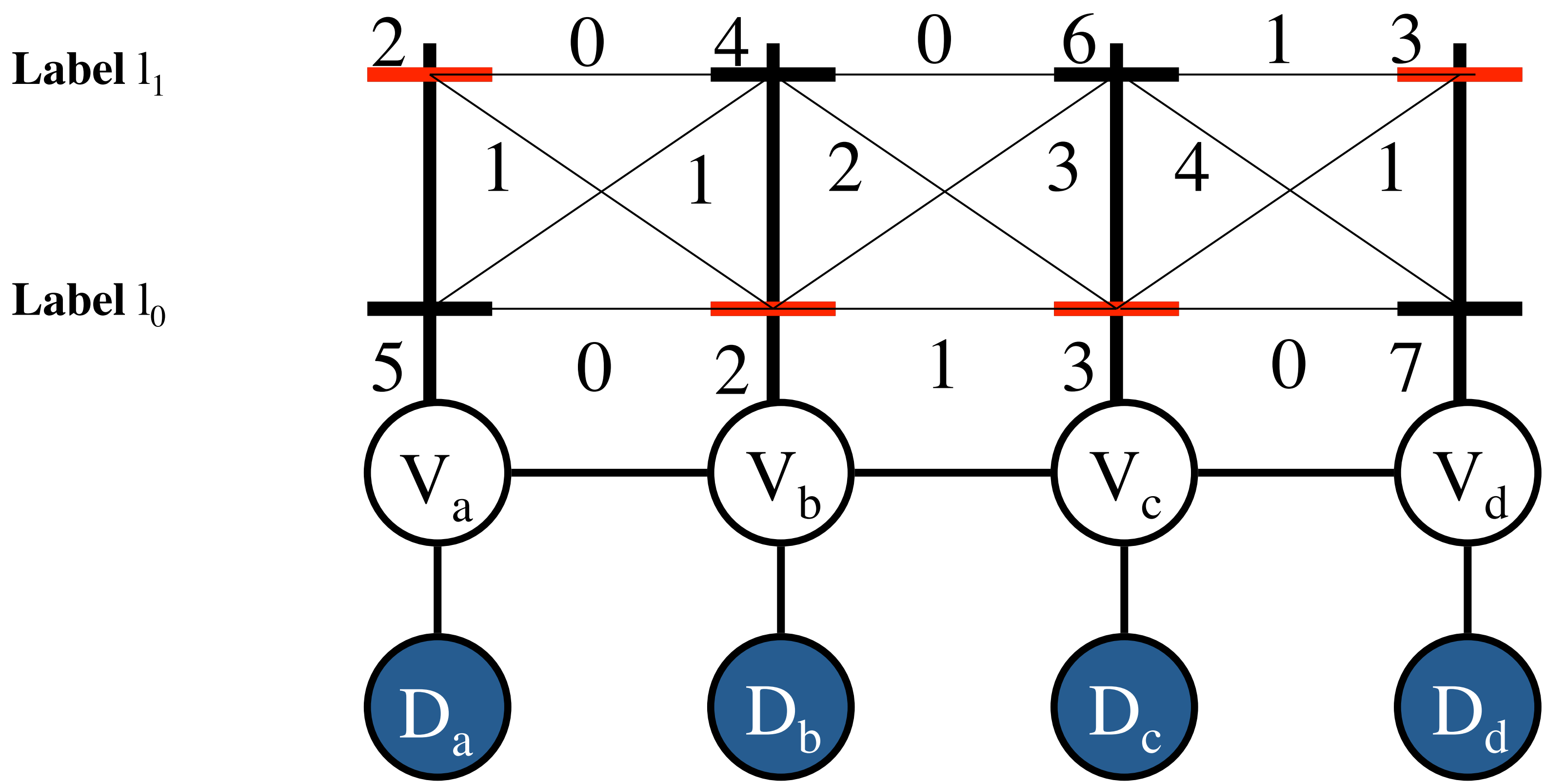
Energy Function



Pairwise Potential

$$Q(f) = \sum_a \theta_{a;f(a)} + \sum_{(a,b)} \theta_{ab;f(a)f(b)}$$

Energy Function



$$Q(f;\theta) = \sum_a \theta_{a;f(a)} + \sum_{(a,b)} \theta_{ab;f(a)f(b)}$$

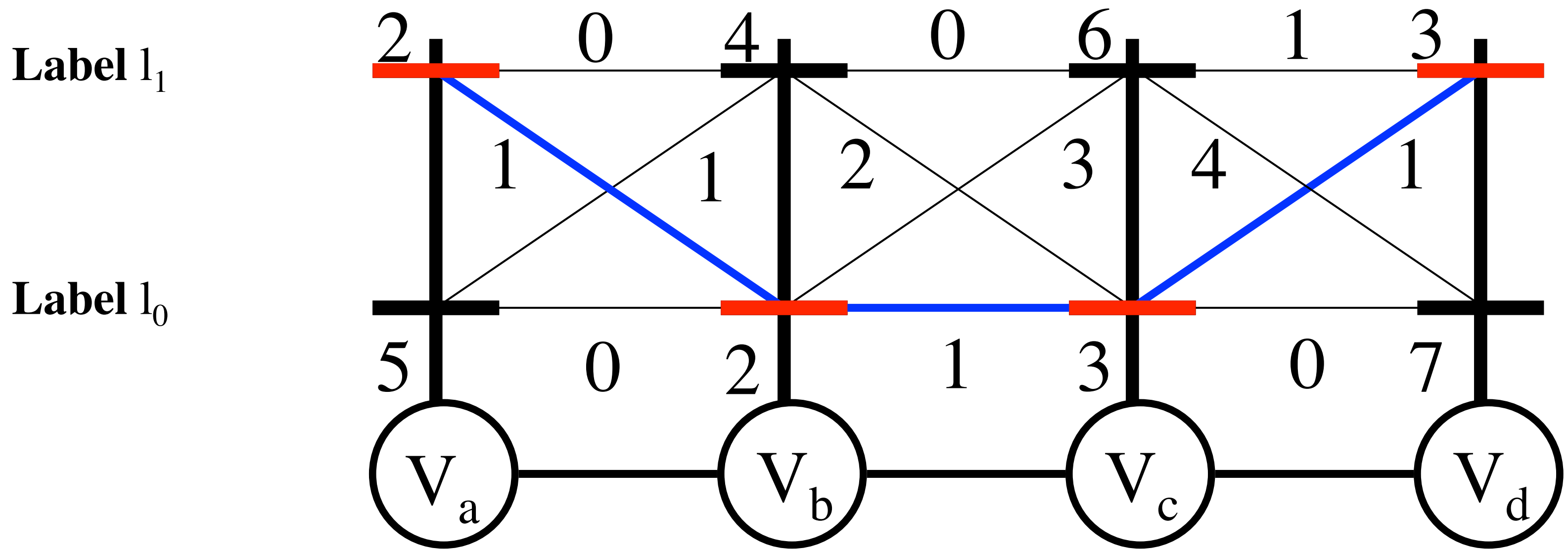
Parameter

Overview

- Basics: problem formulation
  – Energy Function
  – MAP Estimation
  – Computing min-marginals
  – Reparameterization

- Solutions
  – Belief Propagation and related methods [Lecture 1]
  – Graph cuts [Lecture 2]

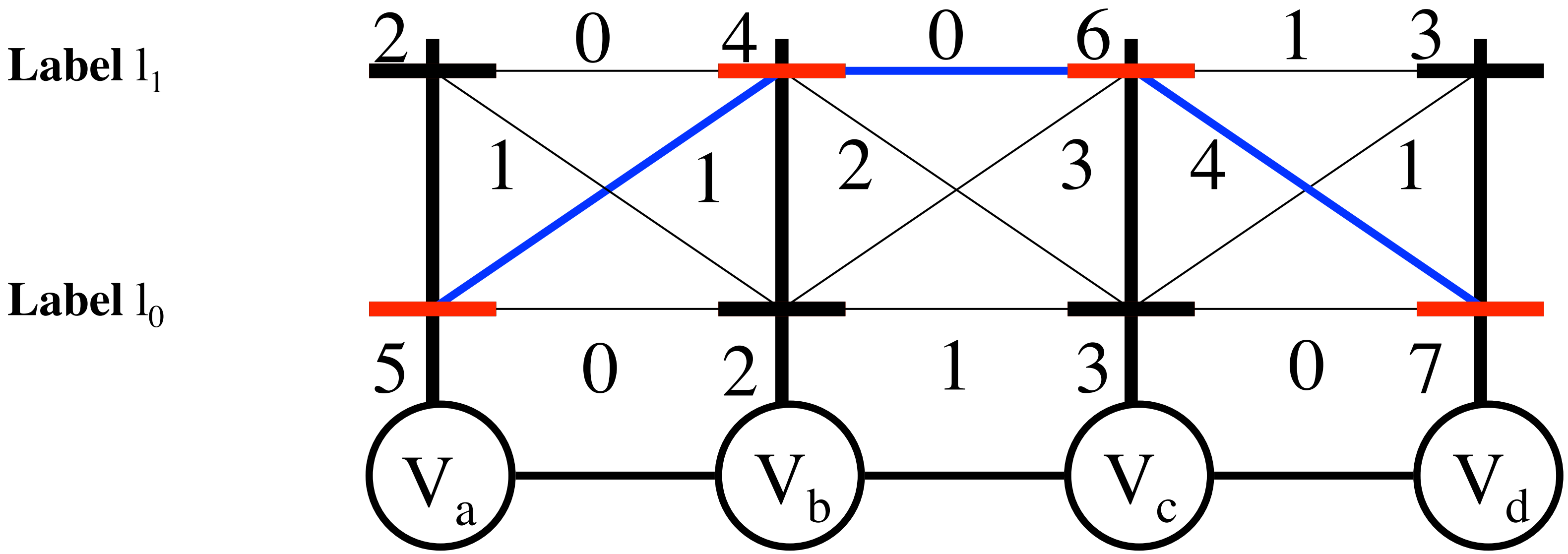
MAP Estimation



$$Q(f;\theta) = \sum_a \theta_{a;f(a)} + \sum_{(a,b)} \theta_{ab;f(a)f(b)}$$

2 + 1 + 2 + 1 + 3 + 1 + 3 = 13

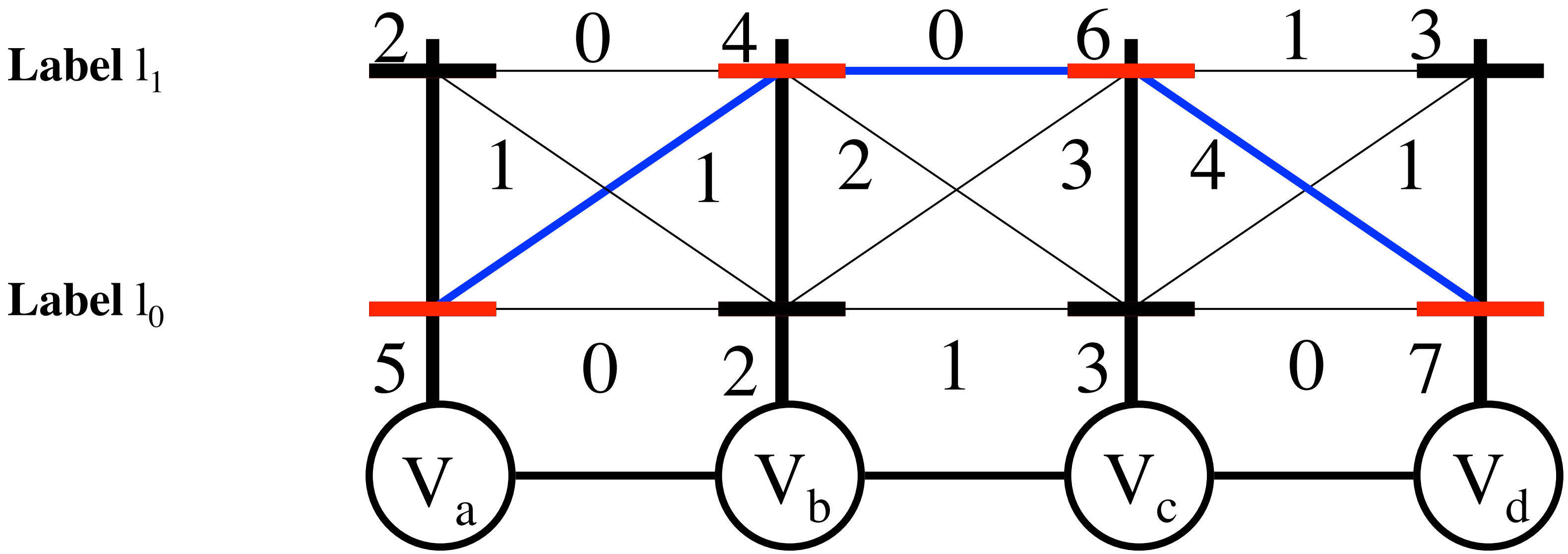
MAP Estimation



$$Q(f; \theta) = \sum_a \theta_{a;f(a)} + \sum_{(a,b)} \theta_{ab;f(a)f(b)}$$

5 + 1 + 4 + 0 + 6 + 4 + 7 = 27

MAP Estimation



$$q^* = \min Q(f; \theta) = Q(f^*; \theta)$$

$$Q(f; \theta) = \sum_a \theta_{a;f(a)} + \sum_{(a,b)} \theta_{ab;f(a)f(b)}$$

$$f^* = \arg\min Q(f; \theta)$$

Equivalent to maximizing the associated probability

MAP Estimation

16 possible labellings

f* = {1, 0, 0, 1}

q* = 13

f(a)	f(b)	f(c)	f(d)	Q(f; θ)
0	0	0	0	18
0	0	0	1	15
0	0	1	0	27
0	0	1	1	20
0	1	0	0	22
0	1	0	1	19
0	1	1	0	27
0	1	1	1	20

f(a)	f(b)	f(c)	f(d)	Q(f; θ)
1	0	0	0	16
1	0	0	1	13
1	0	1	0	25
1	0	1	1	18
1	1	0	0	18
1	1	0	1	15
1	1	1	0	23
1	1	1	1	16

Computational Complexity

Segmentation

$2^{|V|}$

|V| = number of pixels ≈ 153600

Can we do better than brute-force?

MAP Estimation is NP-hard !!

MAP Inference / Energy Minimization

- Computing the assignment minimizing the energy in NP-hard in general

$$\underset{\mathbf{y}\in\mathcal{Y}}{\text{argmin}}\, E(\mathbf{y};\mathbf{x}) = \underset{\mathbf{y}\in\mathcal{Y}}{\text{argmax}}\, P(\mathbf{y} \mid \mathbf{x})$$

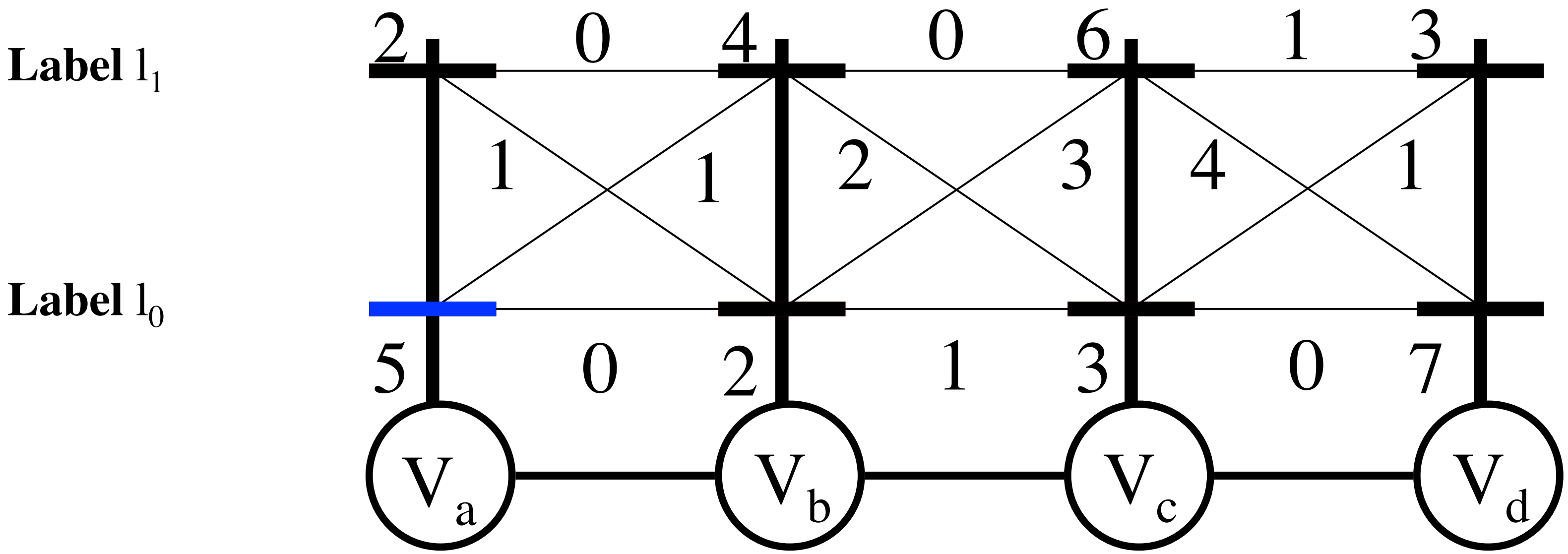
- Exact inference is possible in some cases, e.g.,
  - Low treewidth graphs → message-passing
  - Submodular potentials → graph cuts
- Efficient approximate inference algorithms exist
  - Message passing on general graphs
  - Move-making algorithms
  - Relaxation algorithms

Overview

- Basics: problem formulation
  - Energy Function
  - MAP Estimation
  - Computing min-marginals
  - Reparameterization

- Solutions
  - Belief Propagation and related methods [Lecture 1]
  - Graph cuts [Lecture 2]

Min-Marginals



Not a marginal (no summation)

f* = arg min Q(f; θ) **such that f(a) = i**

Min-marginal $q_{a;i}$

Min-Marginals

16 possible labellings

$q_{a;0} = 15$

f(a)	f(b)	f(c)	f(d)	Q(f; θ)
0	0	0	0	18
0	0	0	1	15
0	0	1	0	27
0	0	1	1	20
0	1	0	0	22
0	1	0	1	19
0	1	1	0	27
0	1	1	1	20

f(a)	f(b)	f(c)	f(d)	Q(f; θ)
1	0	0	0	16
1	0	0	1	13
1	0	1	0	25
1	0	1	1	18
1	1	0	0	18
1	1	0	1	15
1	1	1	0	23
1	1	1	1	16

Min-Marginals

16 possible labellings

$q_{a;1} = 13$

f(a)	f(b)	f(c)	f(d)	Q(f; θ)
0	0	0	0	18
0	0	0	1	15
0	0	1	0	27
0	0	1	1	20
0	1	0	0	22
0	1	0	1	19
0	1	1	0	27
0	1	1	1	20

f(a)	f(b)	f(c)	f(d)	Q(f; θ)
1	0	0	0	16
1	0	0	1	13
1	0	1	0	25
1	0	1	1	18
1	1	0	0	18
1	1	0	1	15
1	1	1	0	23
1	1	1	1	16

Min-Marginals and MAP

- Minimum min-marginal of any variable = energy of MAP labelling

$$\min_i \; q_{a;i}$$

$$\min_i \left(\min_f Q(f; \theta) \quad \text{such that } f(a) = i \right)$$

V_a has to take one label

$$\min_f Q(f; \theta)$$

Summary

Energy Function

$$Q(f; \theta) = \sum_a \theta_{a;f(a)} + \sum_{(a,b)} \theta_{ab;f(a)f(b)}$$

MAP Estimation

$$f^* = \arg\min Q(f; \theta)$$

Min-marginals

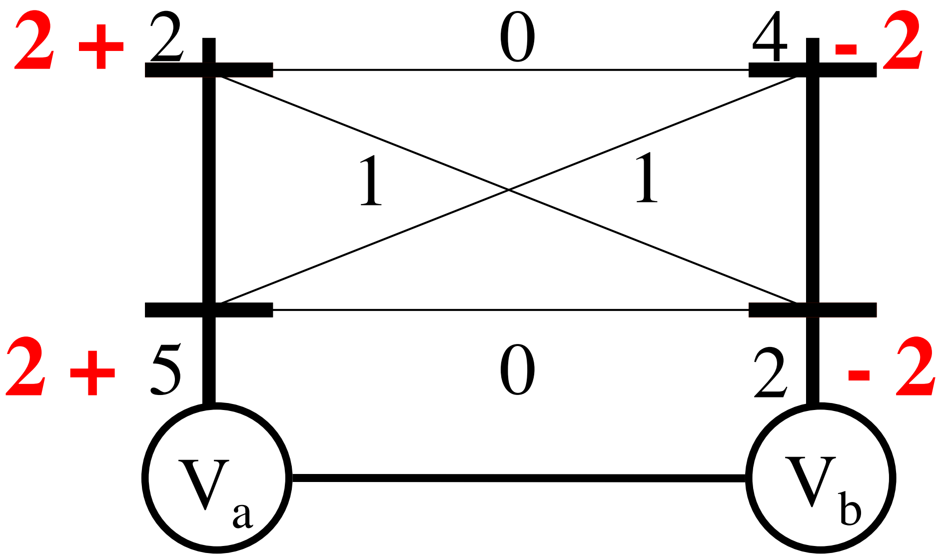
$$q_{a;i} = \min Q(f; \theta) \quad \text{s.t. } f(a) = i$$

Overview

- Basics: problem formulation
  - Energy Function
  - MAP Estimation
  - Computing min-marginals
  - Reparameterization

- Solutions
  - Belief Propagation and related methods [Lecture 1]
  - Graph cuts [Lecture 2]

Reparameterization



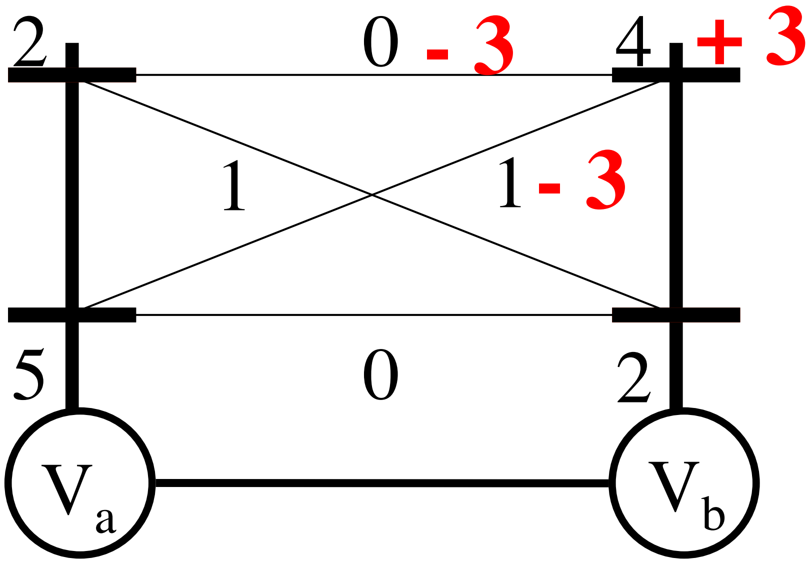
f(a)	f(b)	Q(f; θ)
0	0	7 + 2 - 2
0	1	10 + 2 - 2
1	0	5 + 2 - 2
1	1	6 + 2 - 2

Add a constant to all $\theta_{a;i}$

Subtract that constant from all $\theta_{b;k}$

$Q(f; \theta') = Q(f; \theta)$

Reparameterization



f(a)	f(b)	Q(f; θ)
0	0	7
0	1	10 - 3 + 3
1	0	5
1	1	6 - 3 + 3

Add a constant to one $\theta_{b;k}$

Subtract that constant from $\theta_{ab;ik}$ for all 'i'

$Q(f; \theta') = Q(f; \theta)$

Reparameterization

θ' is a reparameterization of θ, iff

$$Q(f;\theta') = Q(f;\theta), \text{ for all } f \qquad \theta' \equiv \theta$$

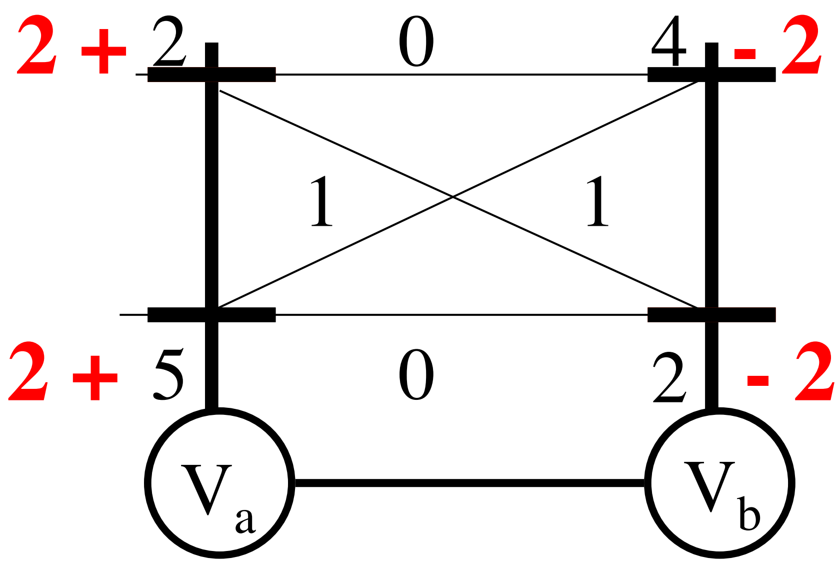
Equivalently

Kolmogorov, PAMI, 2006

$$\theta'_{a;i} = \theta_{a;i} + M_{ba;i}$$

$$\theta'_{b;k} = \theta_{b;k} + M_{ab;k}$$

$$\theta'_{ab;ik} = \theta_{ab;ik} - M_{ab;k} - M_{ba;i}$$



Recap

MAP Estimation

$$f^* = \arg\min Q(f; \theta)$$

$$Q(f; \theta) = \sum_a \theta_{a;f(a)} + \sum_{(a,b)} \theta_{ab;f(a)f(b)}$$

Min-marginals

$$q_{a;i} = \min Q(f; \theta) \quad \text{s.t. } f(a) = i$$

Reparameterization

$$Q(f; \theta') = Q(f; \theta), \text{ for all } f \qquad \theta' \equiv \theta$$

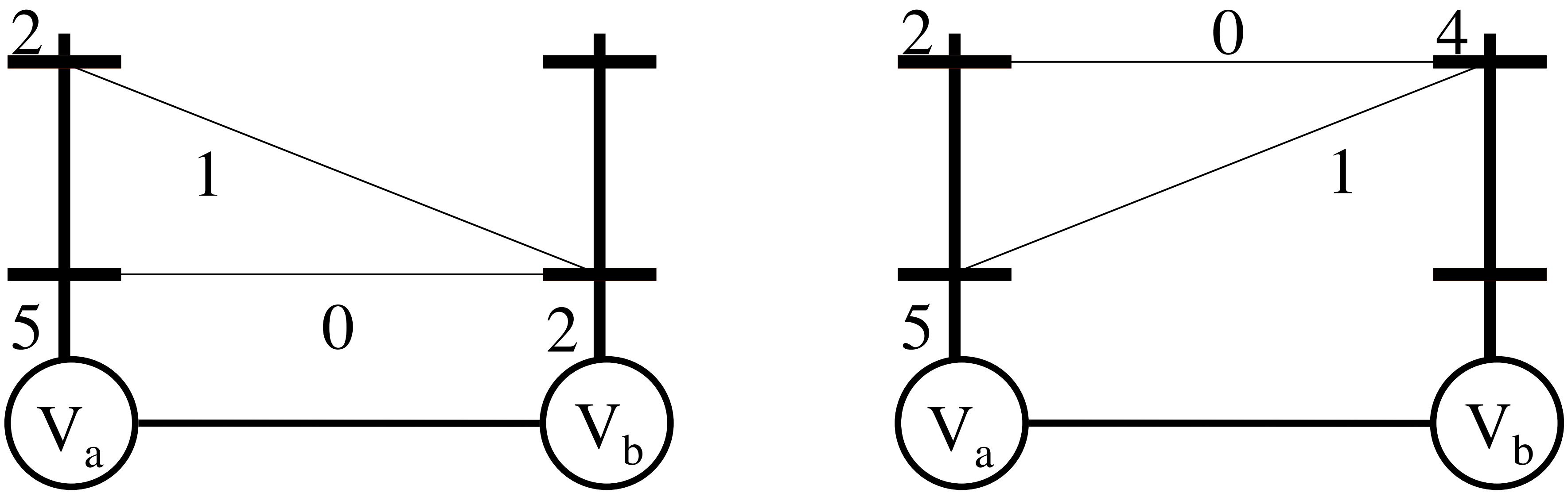
Overview

- Basics: problem formulation
  - Energy Function
  - MAP Estimation
  - Computing min-marginals
  - Reparameterization
- Solutions
  - Belief Propagation and related methods [Lecture 1]
  - Graph cuts [Lecture 2]

Belief Propagation

- Remember, some MAP problems are easy
- Belief Propagation gives exact MAP for chains
- Exact MAP for trees
- Clever Reparameterization

Two Variables

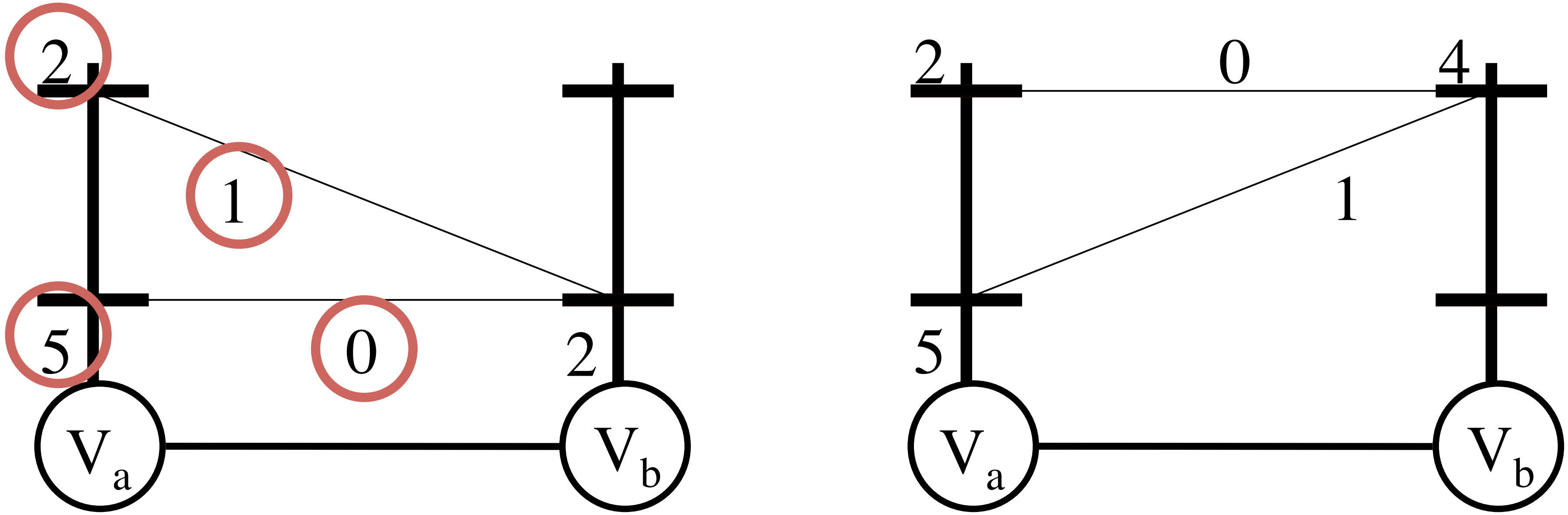


Add a constant to one $\theta_{b;k}$

Subtract that constant from $\theta_{ab;ik}$ for all 'i'

Choose the ***right*** constant $\theta'_{b;k} = q_{b;k}$

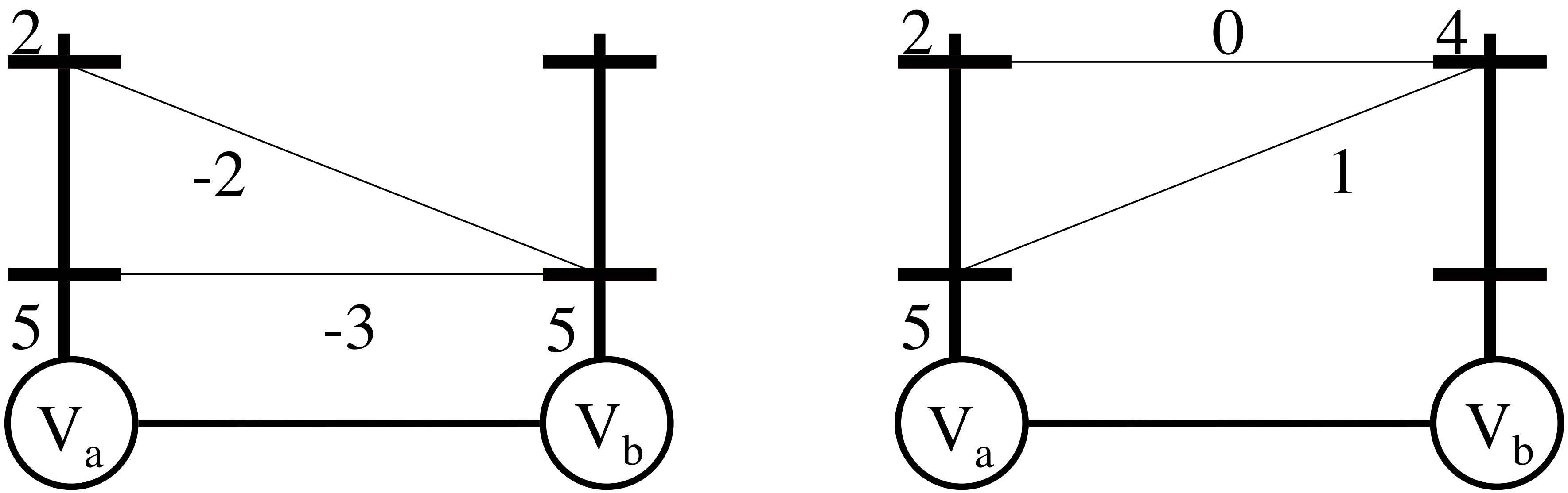
Two Variables



$$M_{ab;0} = \min \begin{cases} \theta_{a;0} + \theta_{ab;00} = 5 + 0 \\ \theta_{a;1} + \theta_{ab;10} = 2 + 1 \end{cases}$$

Choose the ***right*** constant $\theta'_{b;k} = q_{b;k}$

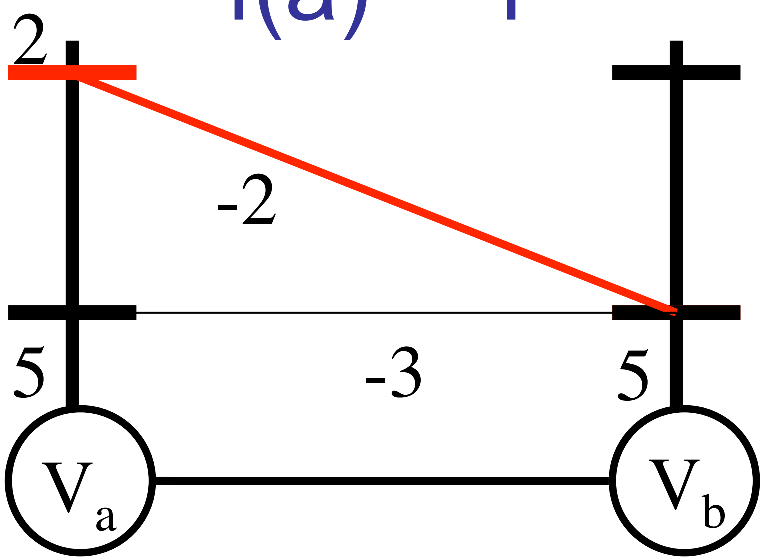
Two Variables

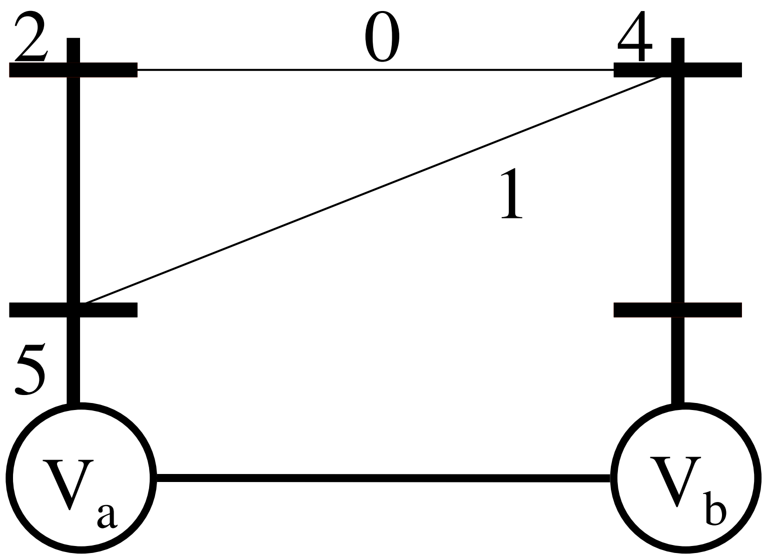


Choose the ***right*** constant $\theta'_{b;k} = q_{b;k}$

Two Variables

f(a) = 1



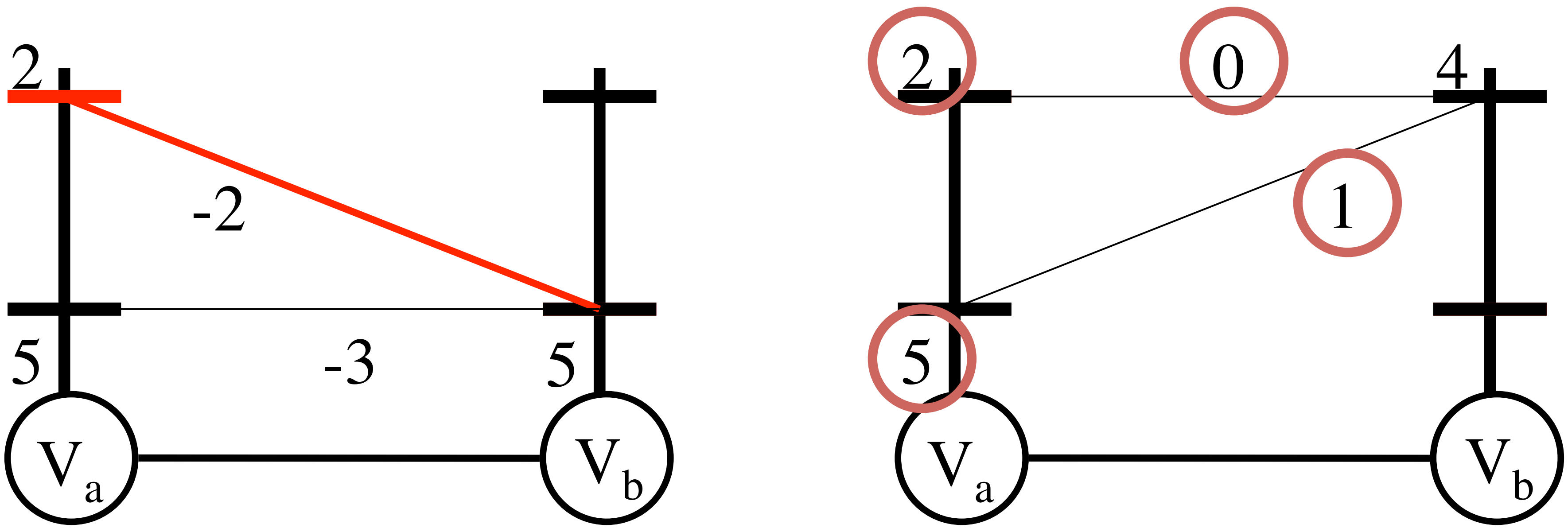


$\theta'_{b;0} = q_{b;0}$

Potentials along the red path add up to 0

Choose the ***right*** constant $\theta'_{b;k} = q_{b;k}$

Two Variables



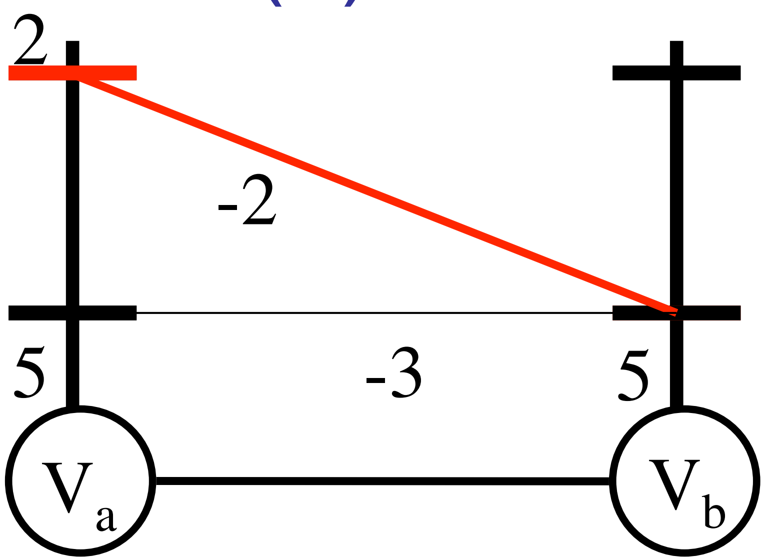
$M_{ab;1}$ = min $\theta_{a;0} + \theta_{ab;01}$ = 5 + 1

$\theta_{a;1} + \theta_{ab;11}$ = 2 + 0

Choose the ***right*** constant $\theta'_{b;k} = q_{b;k}$

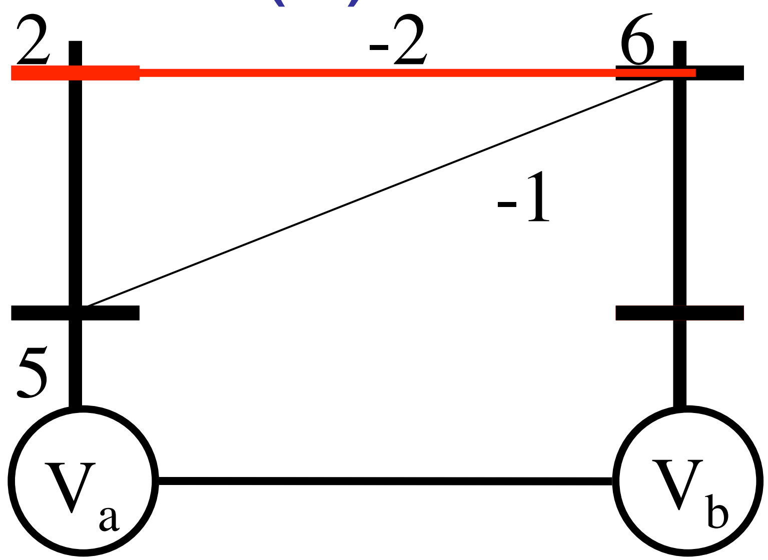
Two Variables

f(a) = 1



$\theta'_{b;0} = q_{b;0}$

f(a) = 1

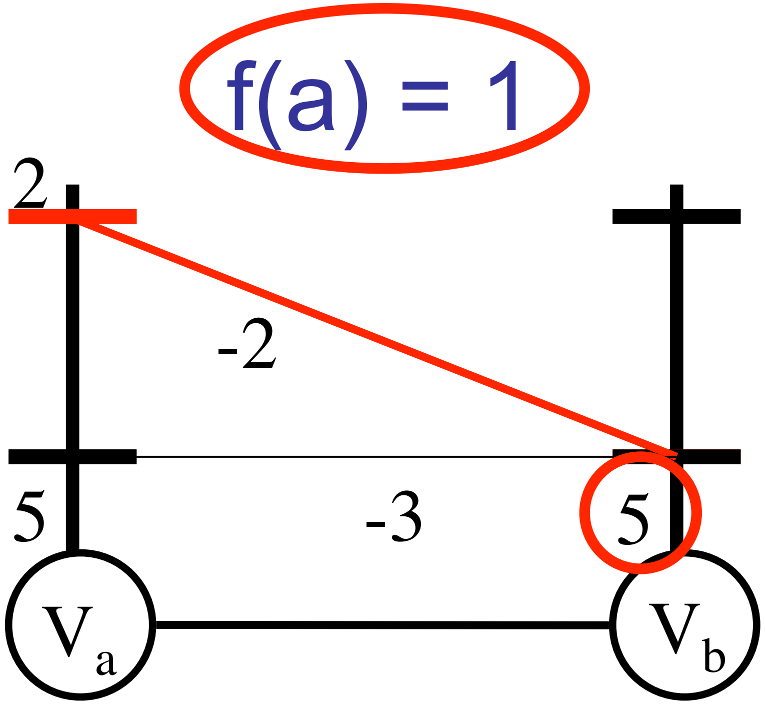


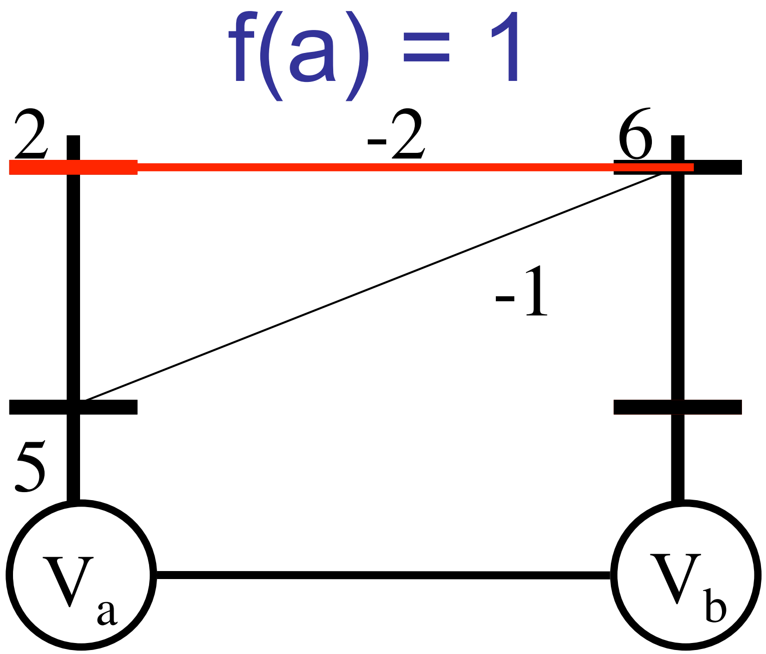
$\theta'_{b;1} = q_{b;1}$

Minimum of min-marginals = MAP estimate

Choose the ***right*** constant $\theta'_{b;k} = q_{b;k}$

Two Variables



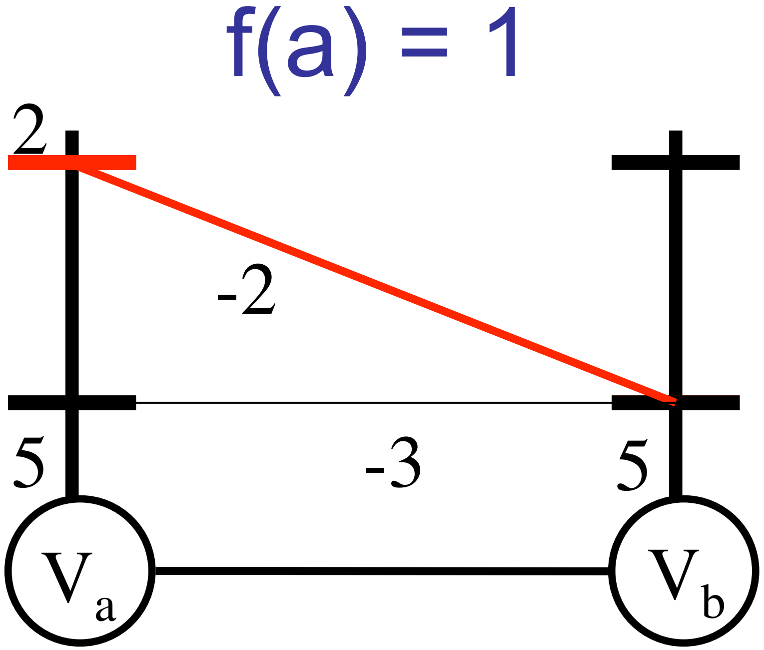


$\theta'_{b;0} = q_{b;0}$ $\theta'_{b;1} = q_{b;1}$

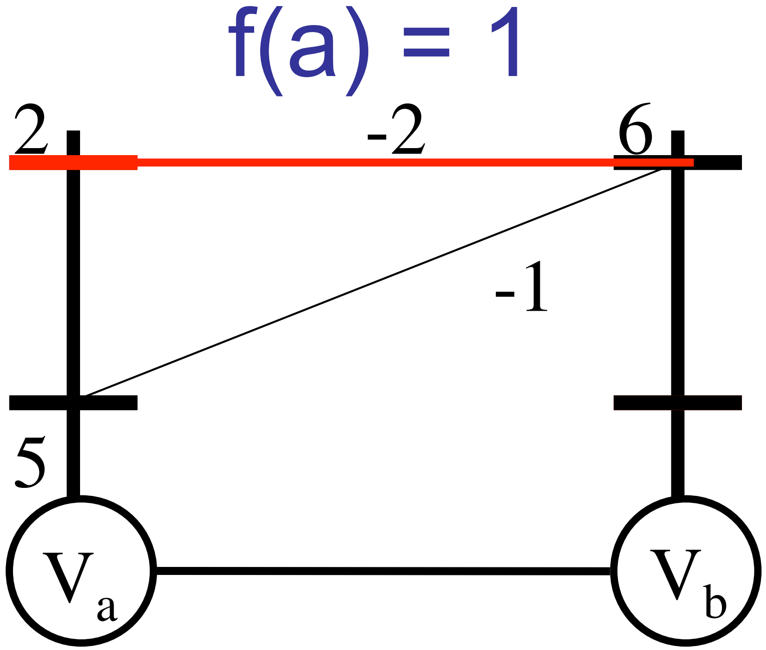
f*(b) = 0 f*(a) = 1

Choose the ***right*** constant $\theta'_{b;k} = q_{b;k}$

Two Variables



$\theta'_{b;0} = q_{b;0}$



$\theta'_{b;1} = q_{b;1}$

We get all the min-marginals of V_b

Choose the ***right*** constant $\theta'_{b;k} = q_{b;k}$

Recap

We only need to know two sets of equations

General form of Reparameterization

$$\theta'_{a;i} = \theta_{a;i} + M_{ba;i} \qquad \theta'_{b;k} = \theta_{b;k} + M_{ab;k}$$

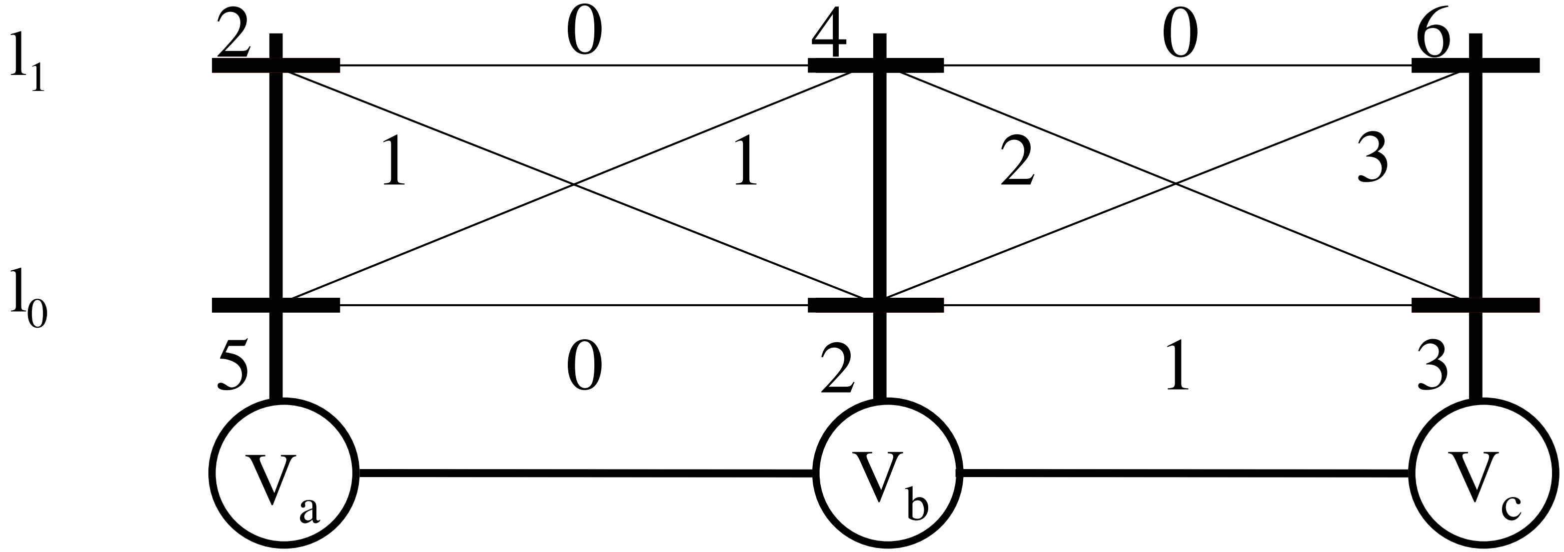
$$\theta'_{ab;ik} = \theta_{ab;ik} - M_{ab;k} - M_{ba;i}$$

Reparameterization of (a,b) in Belief Propagation

$$M_{ab;k} = \min_i \{ \theta_{a;i} + \theta_{ab;ik} \}$$

$$M_{ba;i} = 0$$

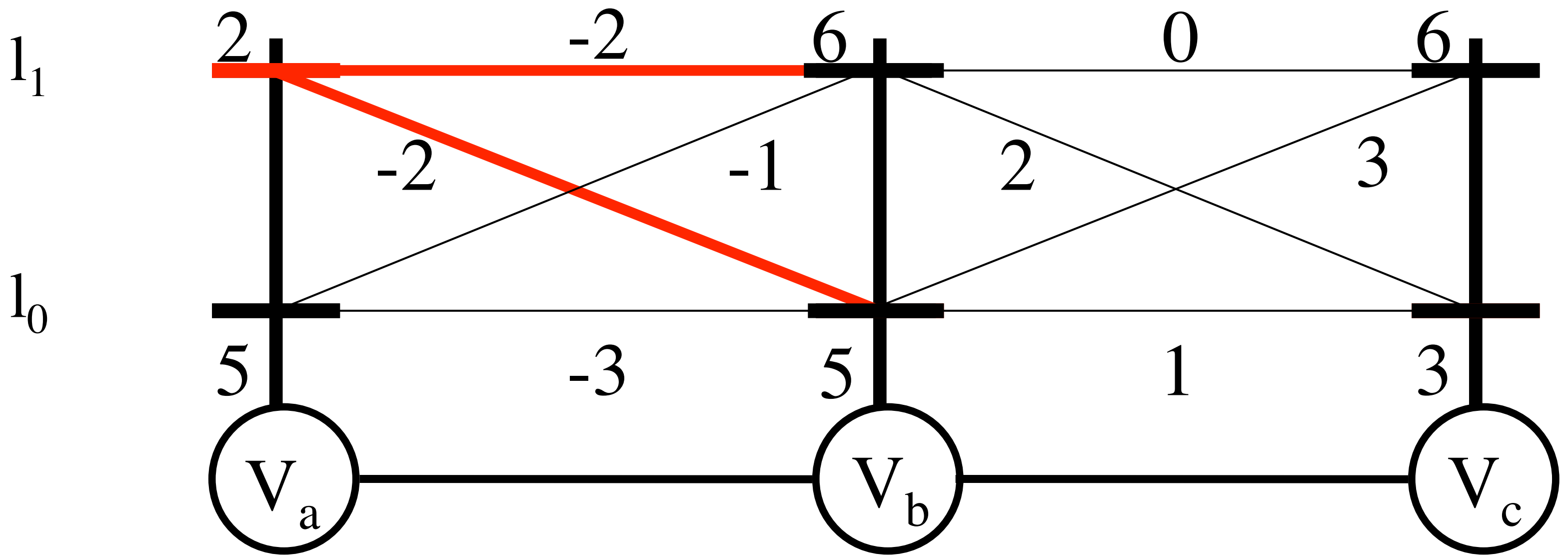
Three Variables



Reparameterize the edge (a,b) as before

Three Variables

f(a) = 1

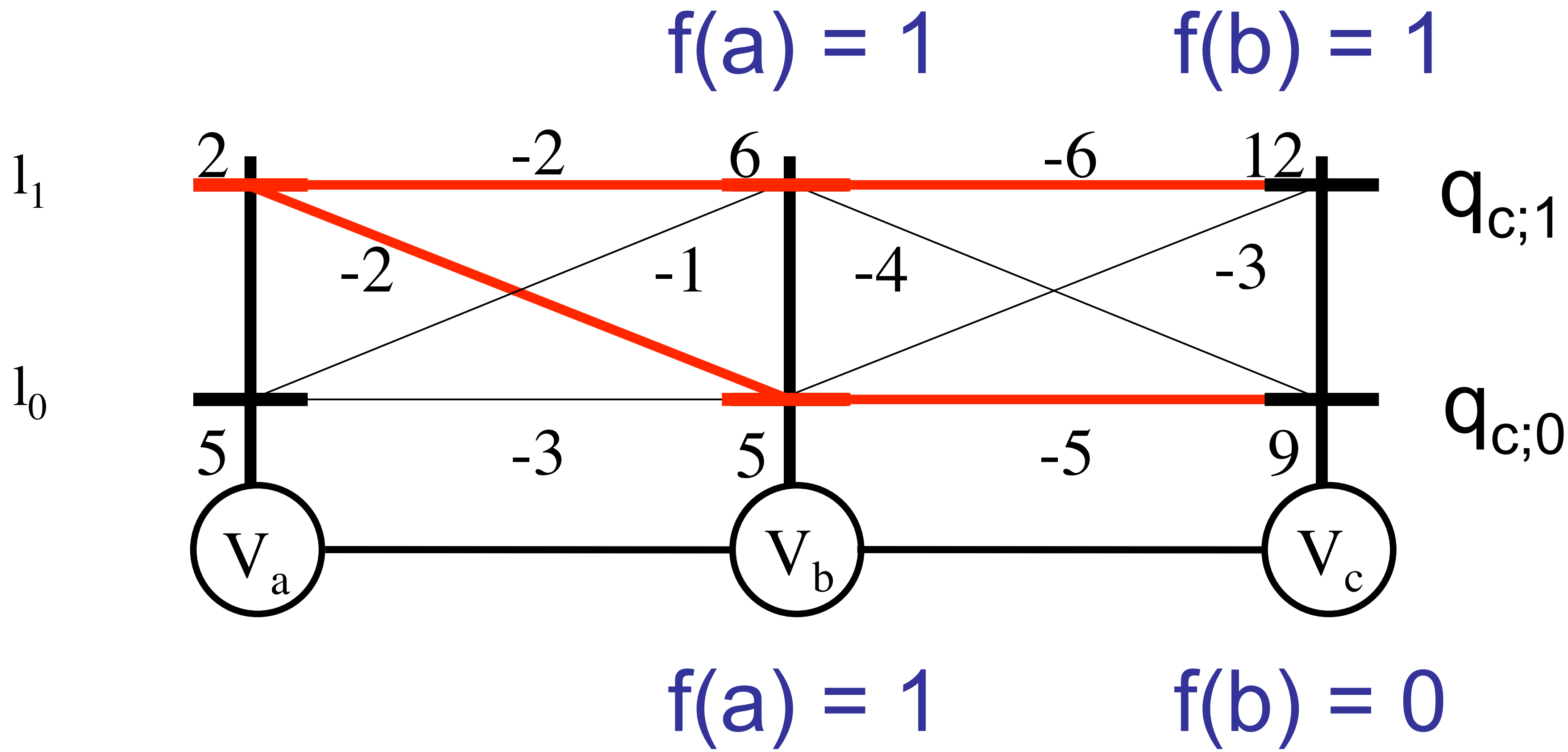


f(a) = 1

Reparameterize the edge (a,b) as before

Potentials along the red path add up to 0

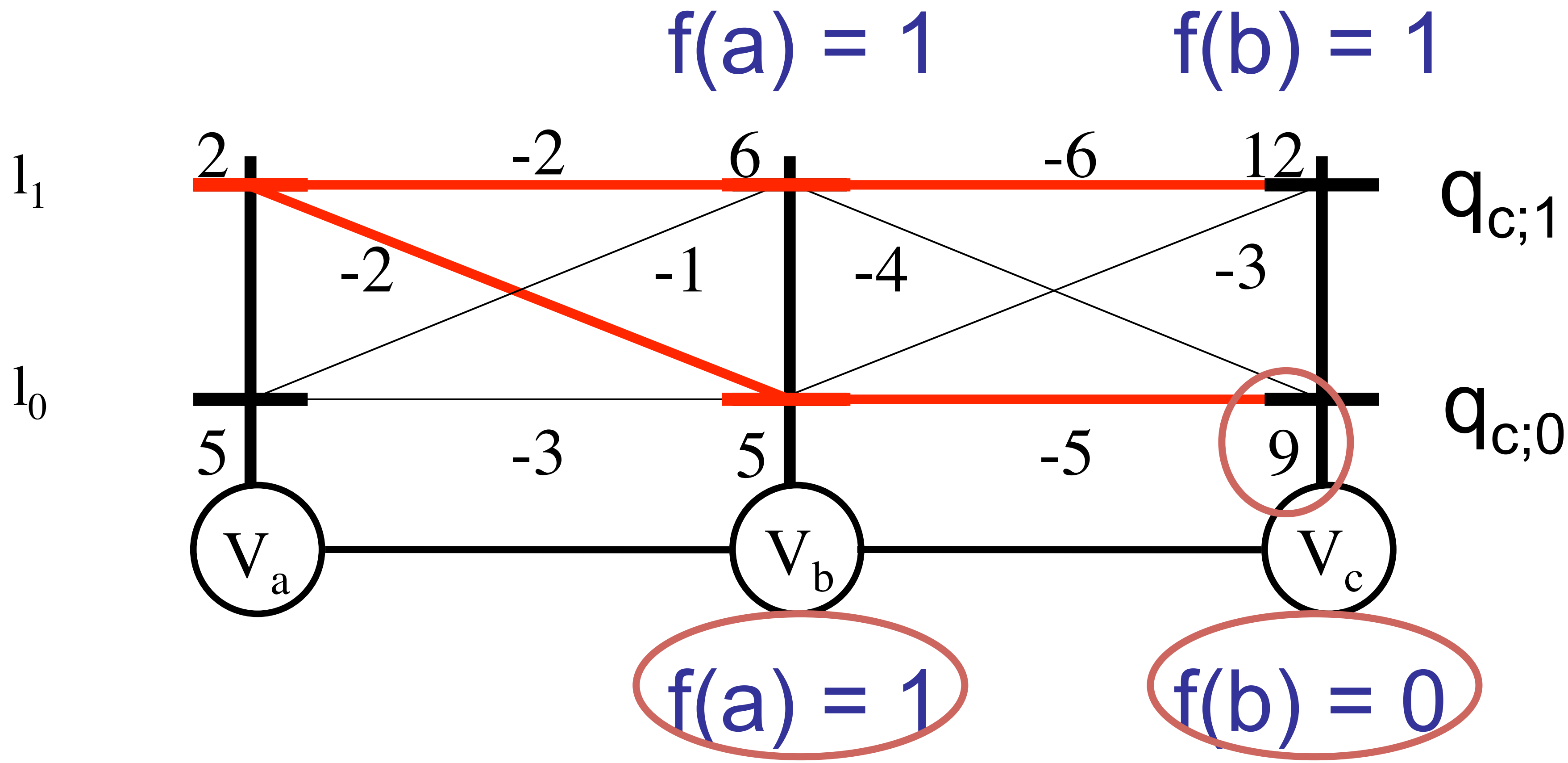
Three Variables



Reparameterize the edge (b,c) as before

Potentials along the red path add up to 0

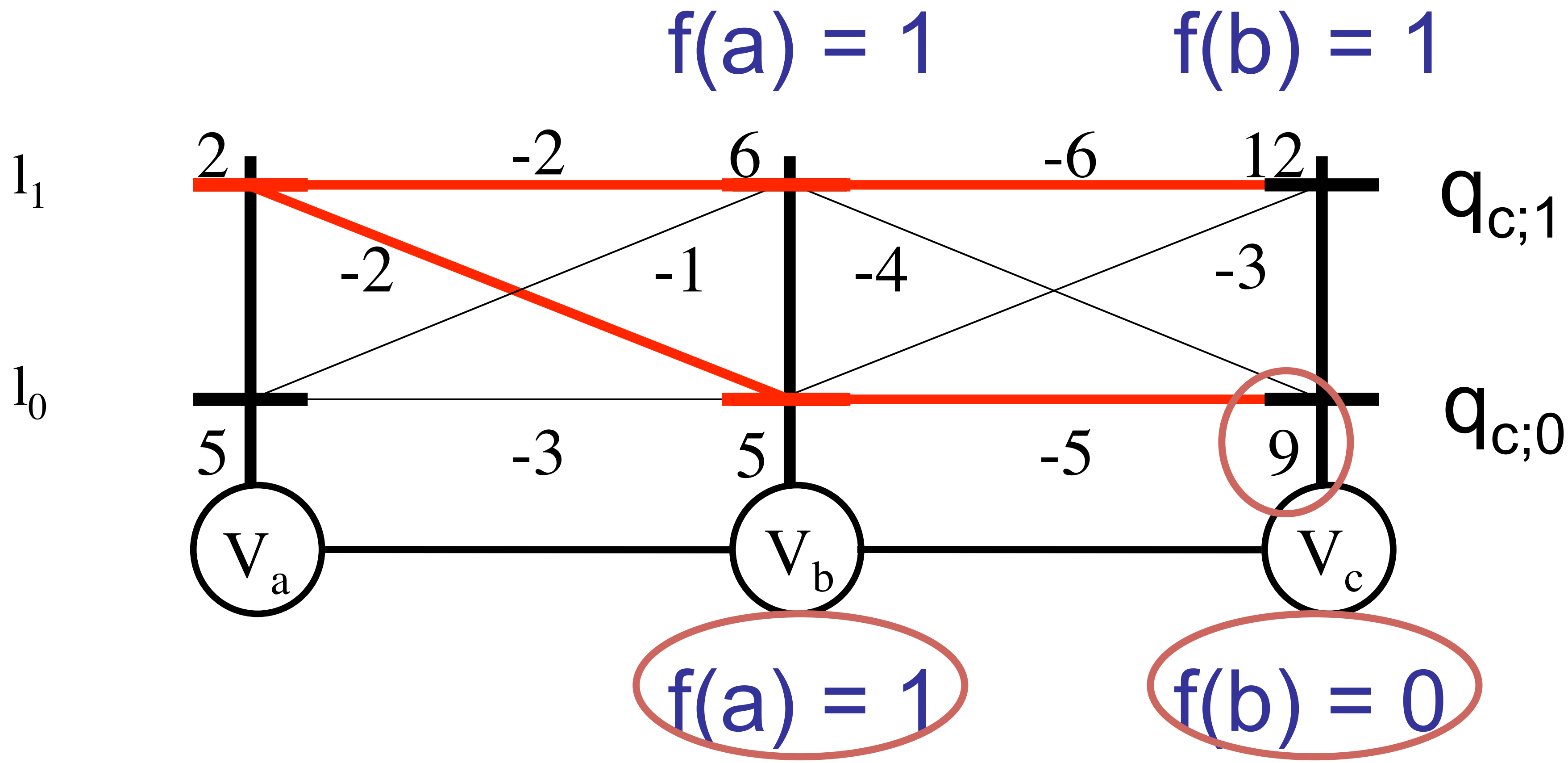
Three Variables



f*(c) = 0 f*(b) = 0 f*(a) = 1

Generalizes to any length chain

Three Variables



f*(c) = 0 f*(b) = 0 f*(a) = 1

Only Dynamic Programming

Why Dynamic Programming?

3 variables ≡ 2 variables + book-keeping

n variables ≡ (n-1) variables + book-keeping

Start from left, go to right

Reparameterize current edge (a,b)

$$M_{ab;k} = \min_i \{ \theta_{a;i} + \theta_{ab;ik} \}$$

$$\theta'_{b;k} = \theta_{b;k} + M_{ab;k} \quad \theta'_{ab;ik} = \theta_{ab;ik} - M_{ab;k}$$

Repeat

Why Dynamic Programming?

Messages   Message Passing

Why stop at dynamic programming?

Start from left, go to right

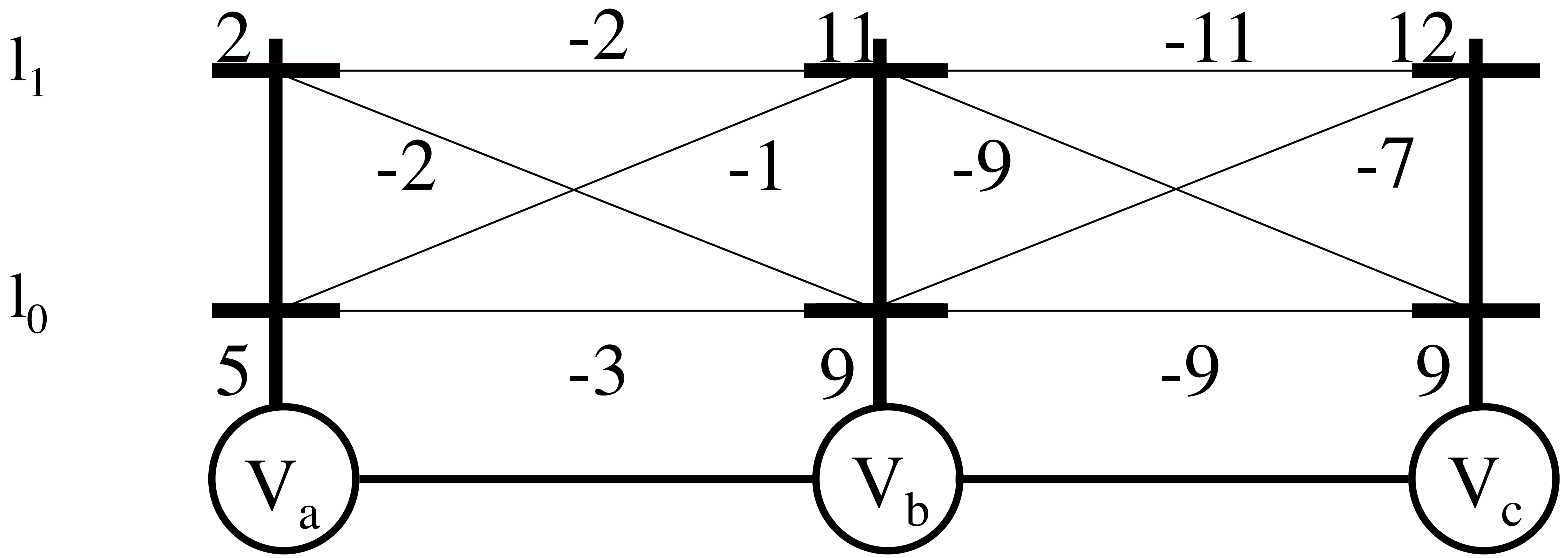
Reparameterize current edge (a,b)

$$M_{ab;k} = \min_i \{ \theta_{a;i} + \theta_{ab;ik} \}$$

$$\theta'_{b;k} = \theta_{b;k} + M_{ab;k} \quad \theta'_{ab;ik} = \theta_{ab;ik} - M_{ab;k}$$

Repeat

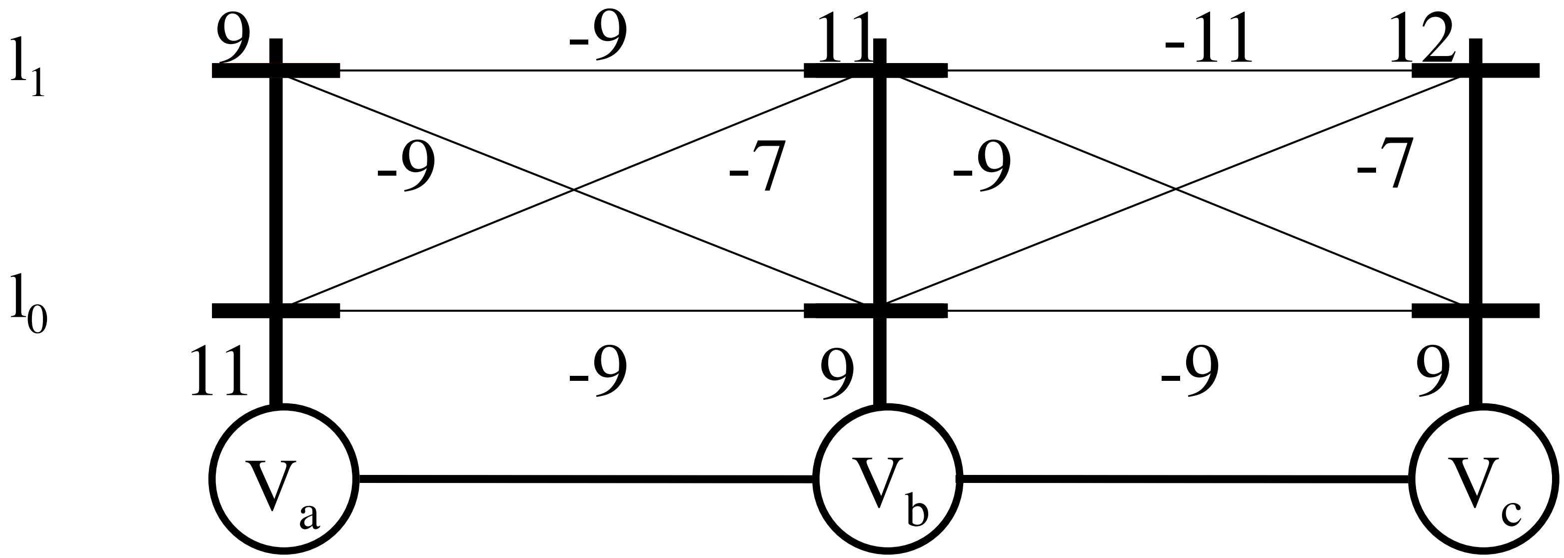
Three Variables



Reparameterize the edge (c,b) as before

$$\theta'_{b;i} = q_{b;i}$$

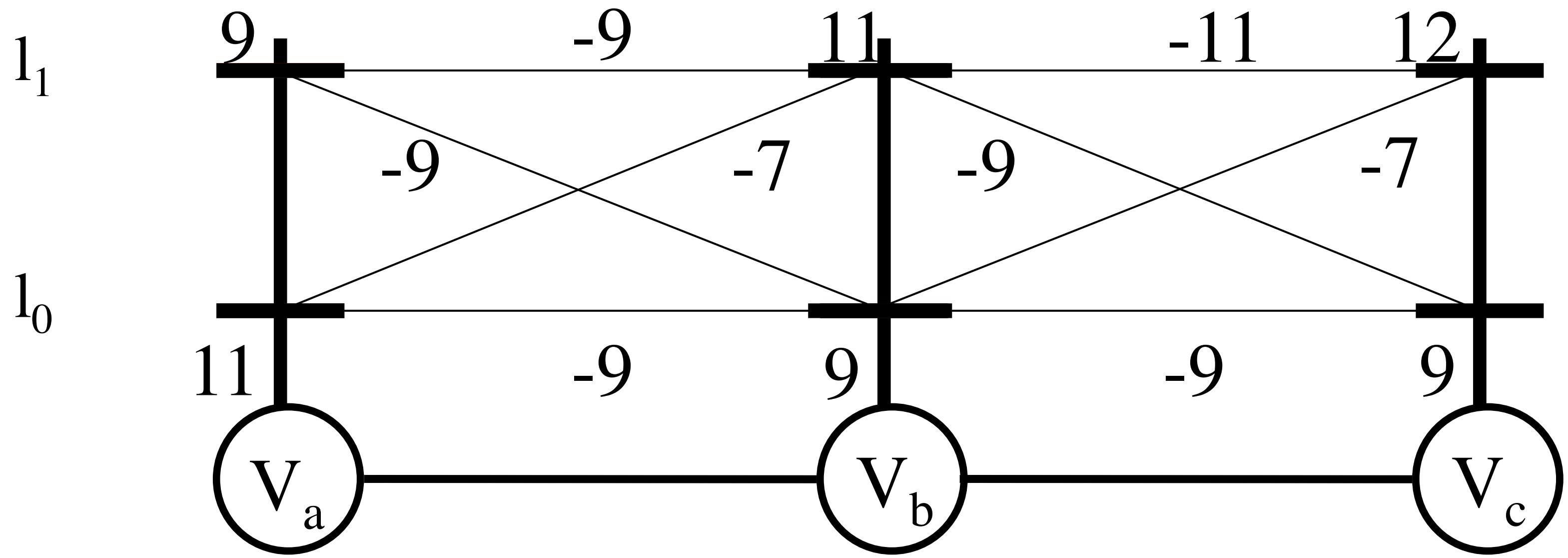
Three Variables



Reparameterize the edge (b,a) as before

$$\theta'_{a;i} = q_{a;i}$$

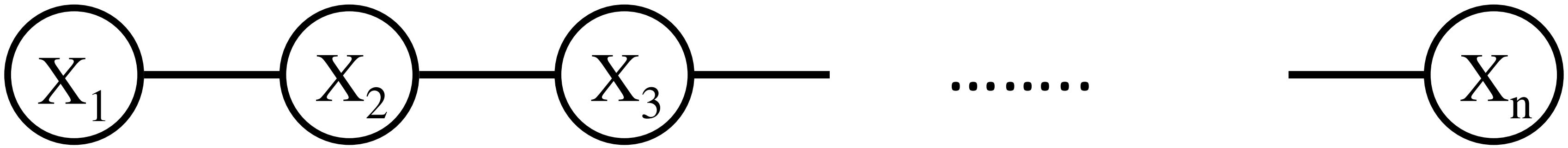
Three Variables



Forward Pass → ← Backward Pass

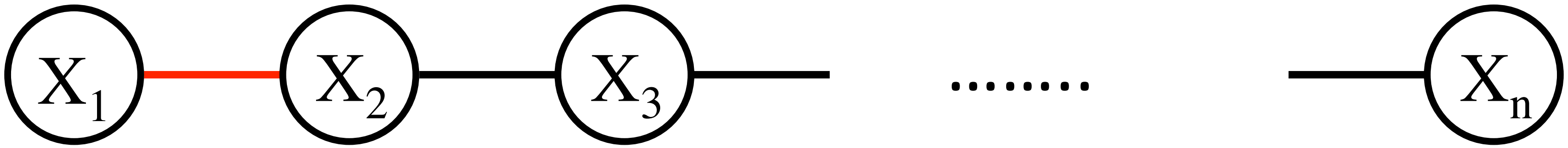
All min-marginals are computed

Chains



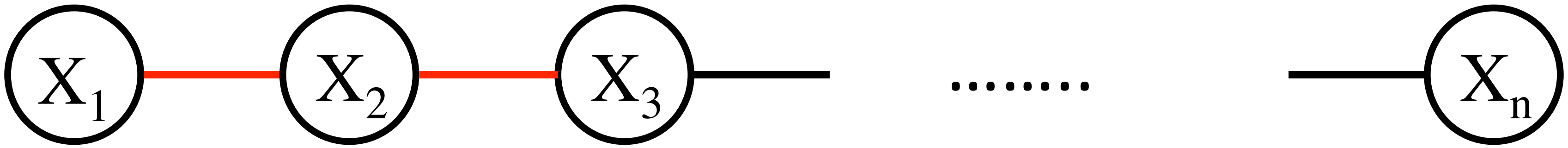
Reparameterize the edge (1,2)

Chains



Reparameterize the edge (1,2)

Chains



Reparameterize the edge (2,3)

Chains

Reparameterize the edge (n-1,n)

Min-marginals $e_n(i)$ for all labels

Belief Propagation on Chains

Start from left, go to right

Reparameterize current edge (a,b)

$$M_{ab;k} = \min_i \{ \theta_{a;i} + \theta_{ab;ik} \}$$

$$\theta'_{b;k} = \theta_{b;k} + M_{ab;k} \quad \theta'_{ab;ik} = \theta_{ab;ik} - M_{ab;k}$$

Repeat till the end of the chain

Start from right, go to left

Repeat till the end of the chain

Belief Propagation on Chains

- Generalizes to chains of any length
- A way of computing reparam constants
- Forward Pass - Start to End
  - MAP estimate
  - Min-marginals of final variable
- Backward Pass - End to start
  - All other min-marginals

Computational Complexity

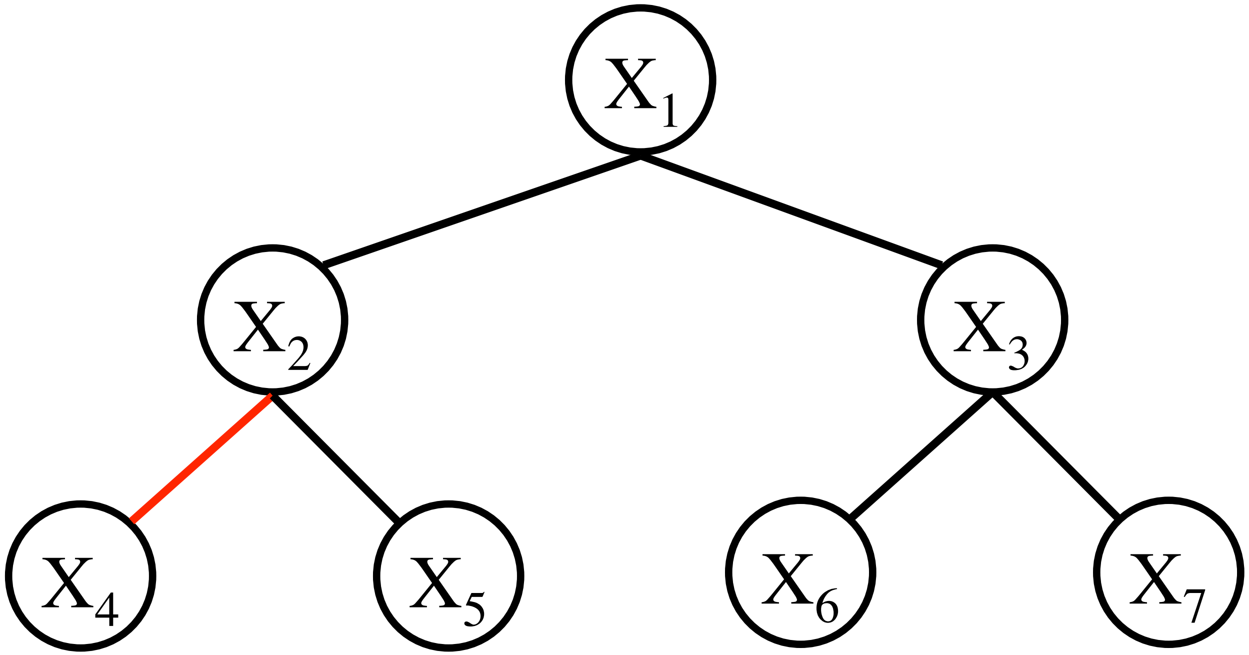
Number of reparameterization constants = (n-1)h

Complexity for each constant = O(h)

Total complexity = $O(nh^2)$

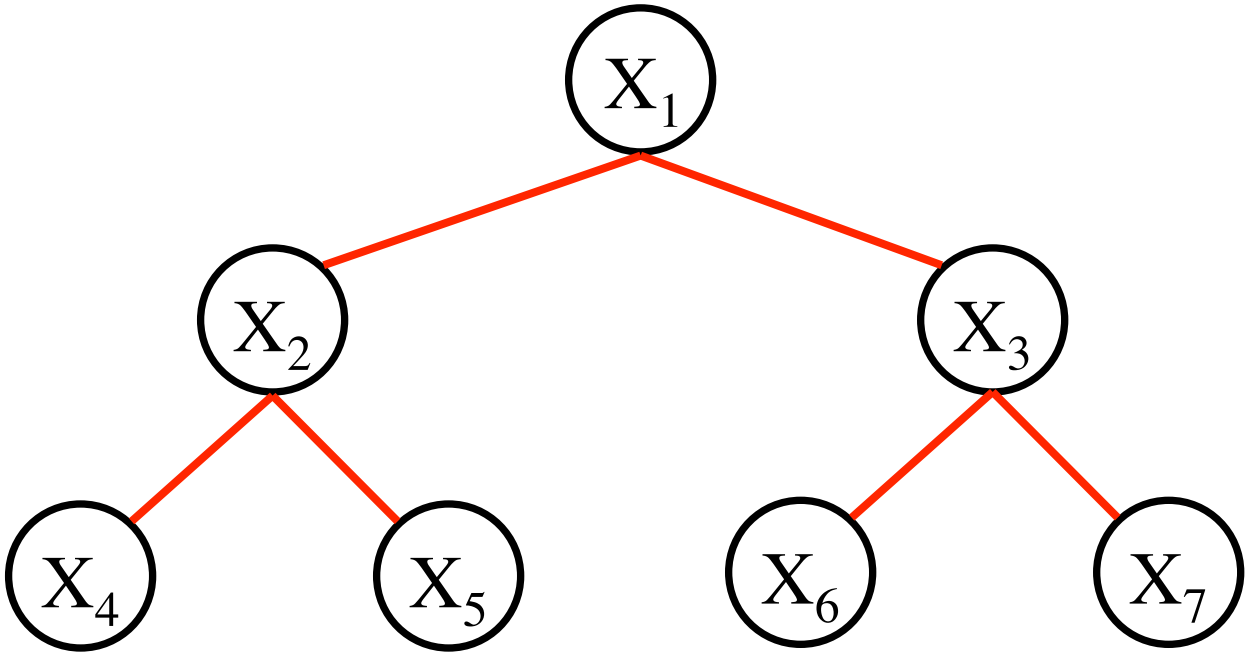
Better than brute-force $O(h^n)$

Trees



Reparameterize the edge (4,2)

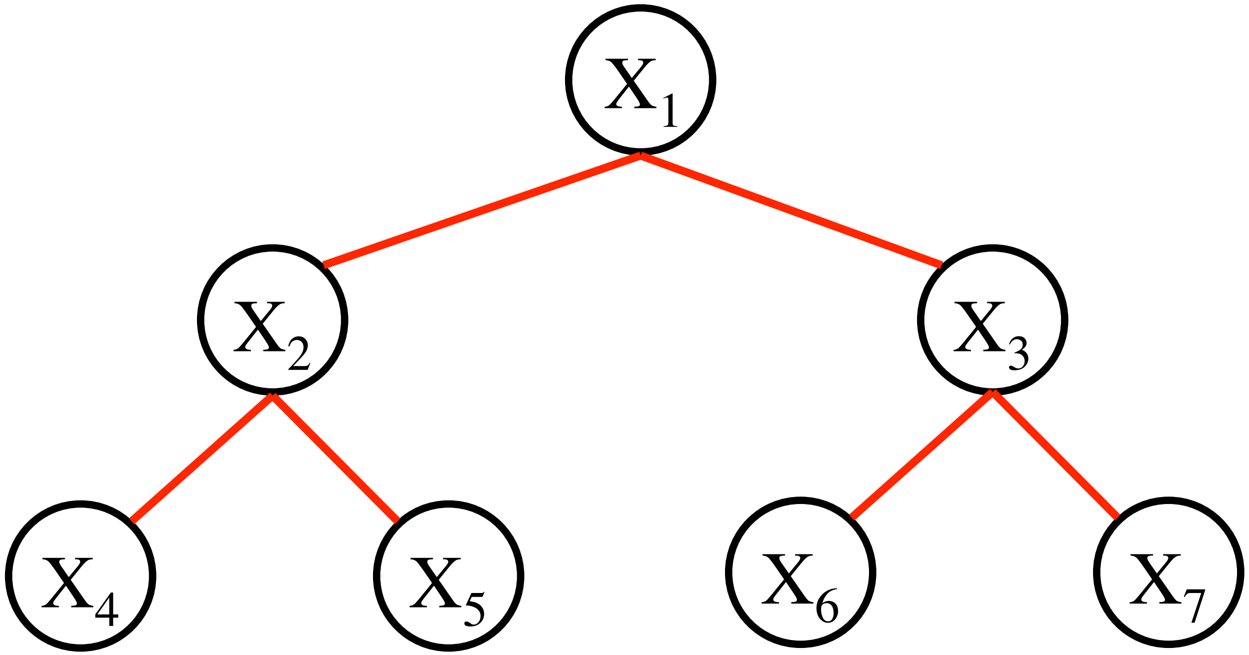
Trees



Reparameterize the edge (3,1)

Min-marginals $e_1(i)$ for all labels

Trees



Start from leaves and move towards root

Pick the minimum of min-marginals

Backtrack to find the best labeling **x**

Computational Complexity

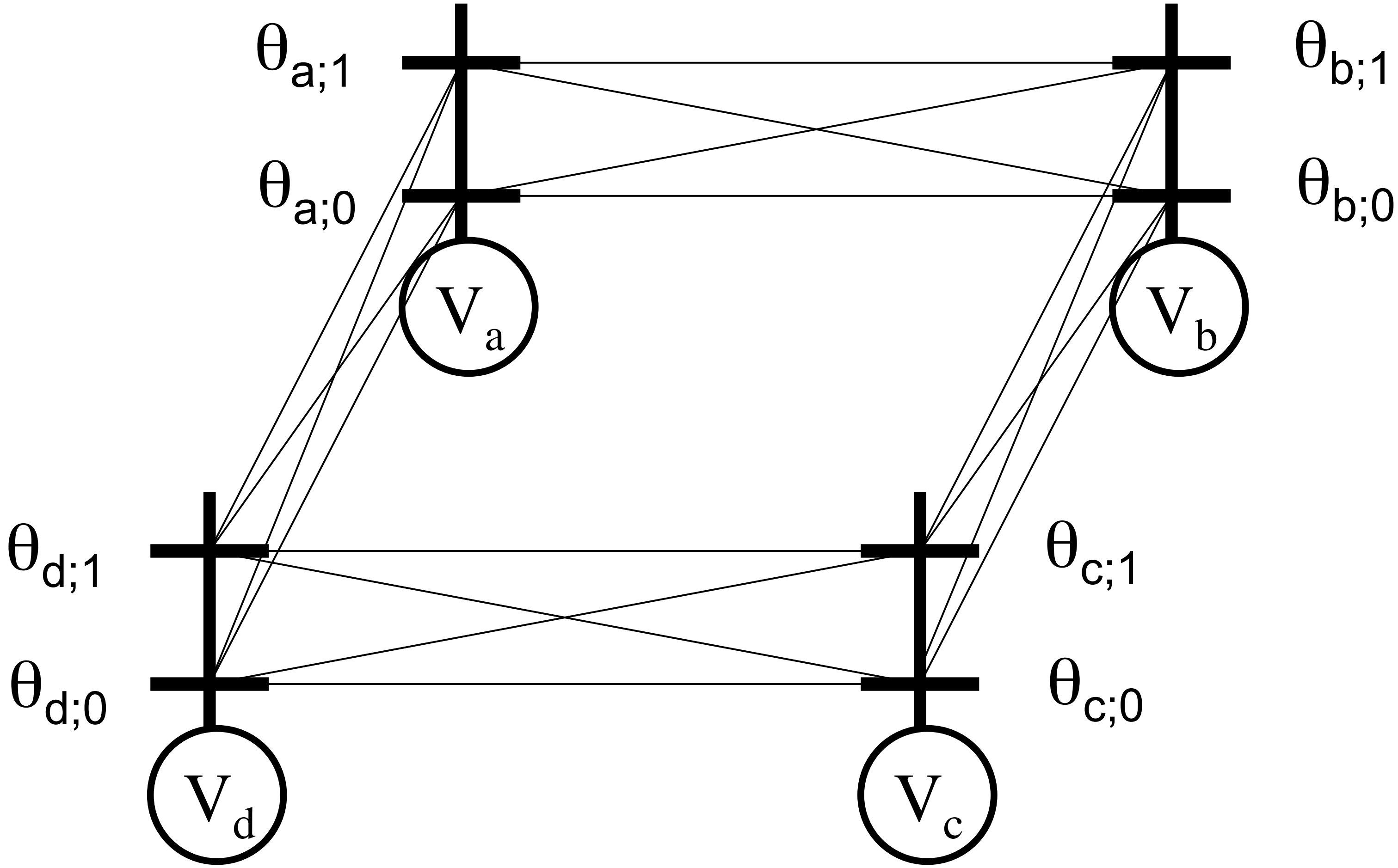
Number of reparameterization constants = (n-1)h

Complexity for each constant = O(h)

Total complexity = $O(nh^2)$

Better than brute-force $O(h^n)$

Belief Propagation on Cycles

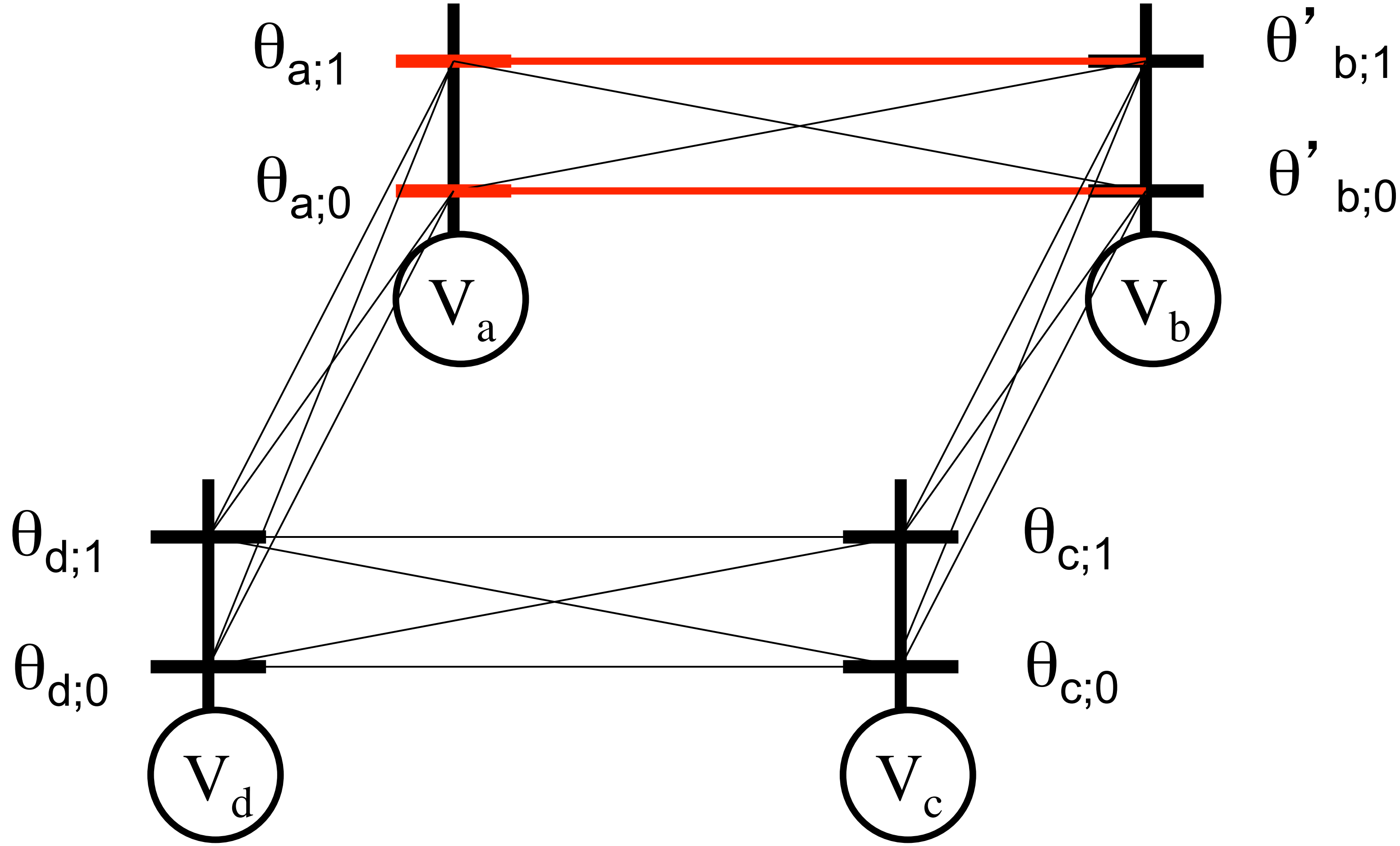


Where do we start?

Arbitrarily

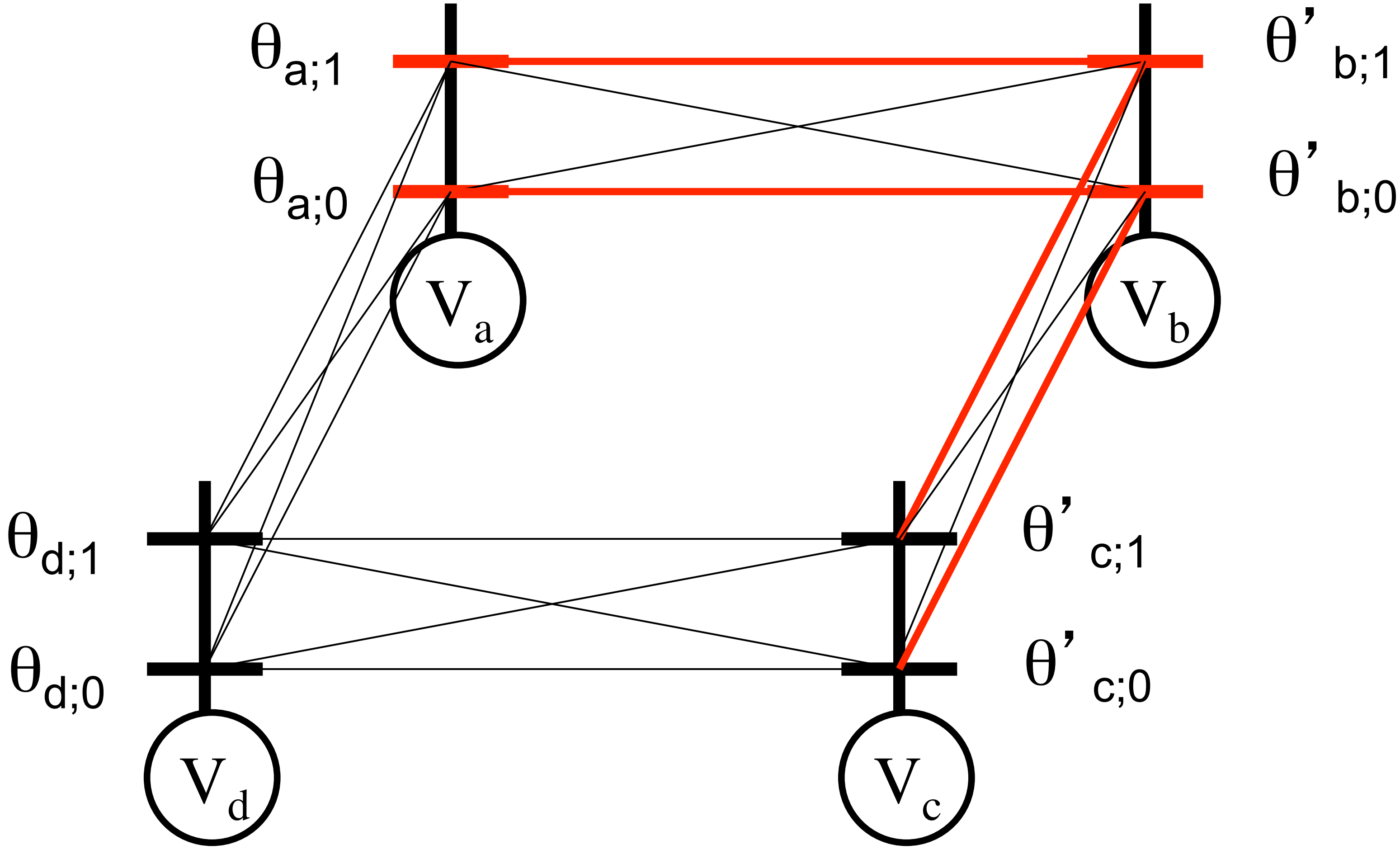
Reparameterize (a,b)

Belief Propagation on Cycles



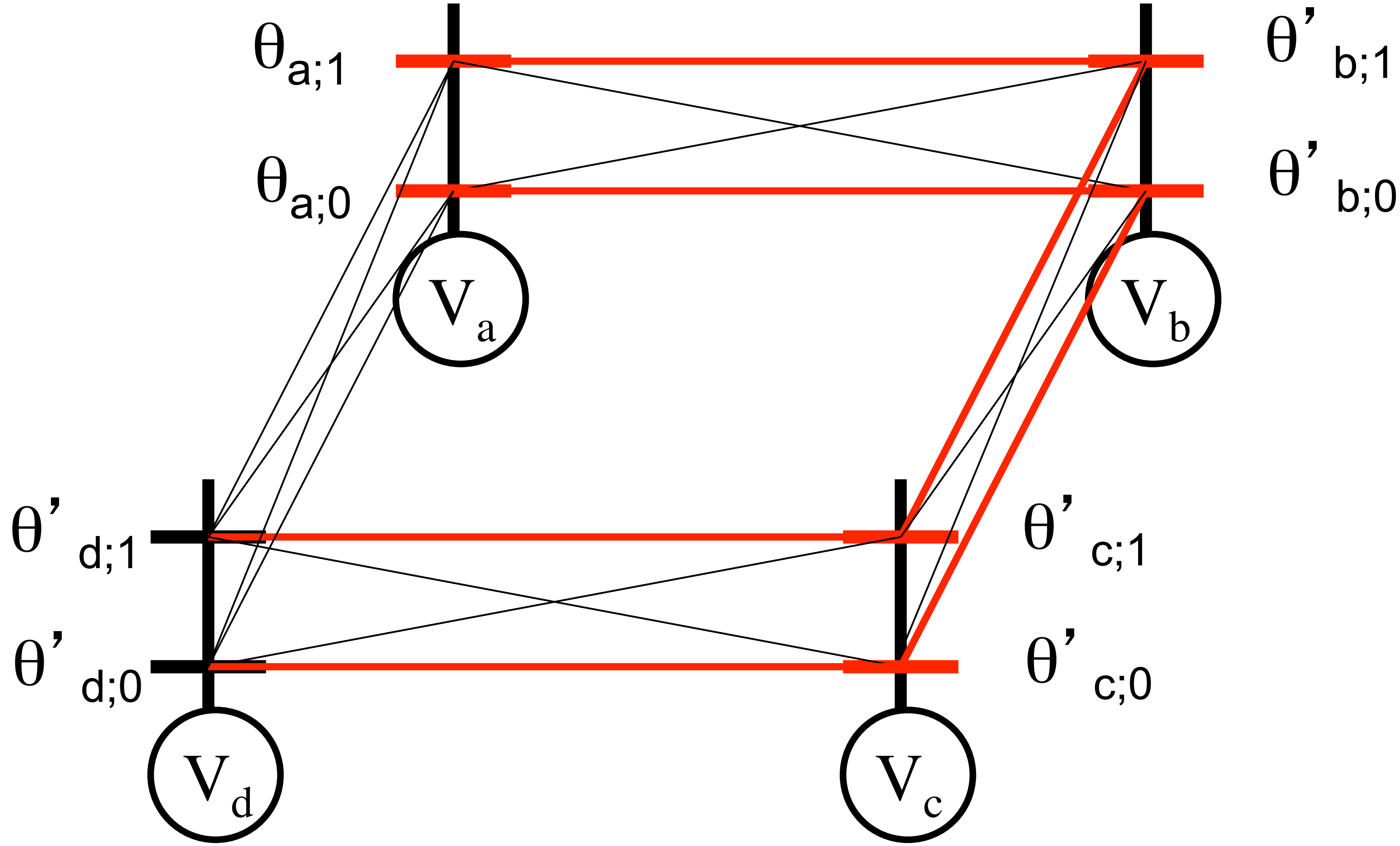
Potentials along the red path add up to 0

Belief Propagation on Cycles



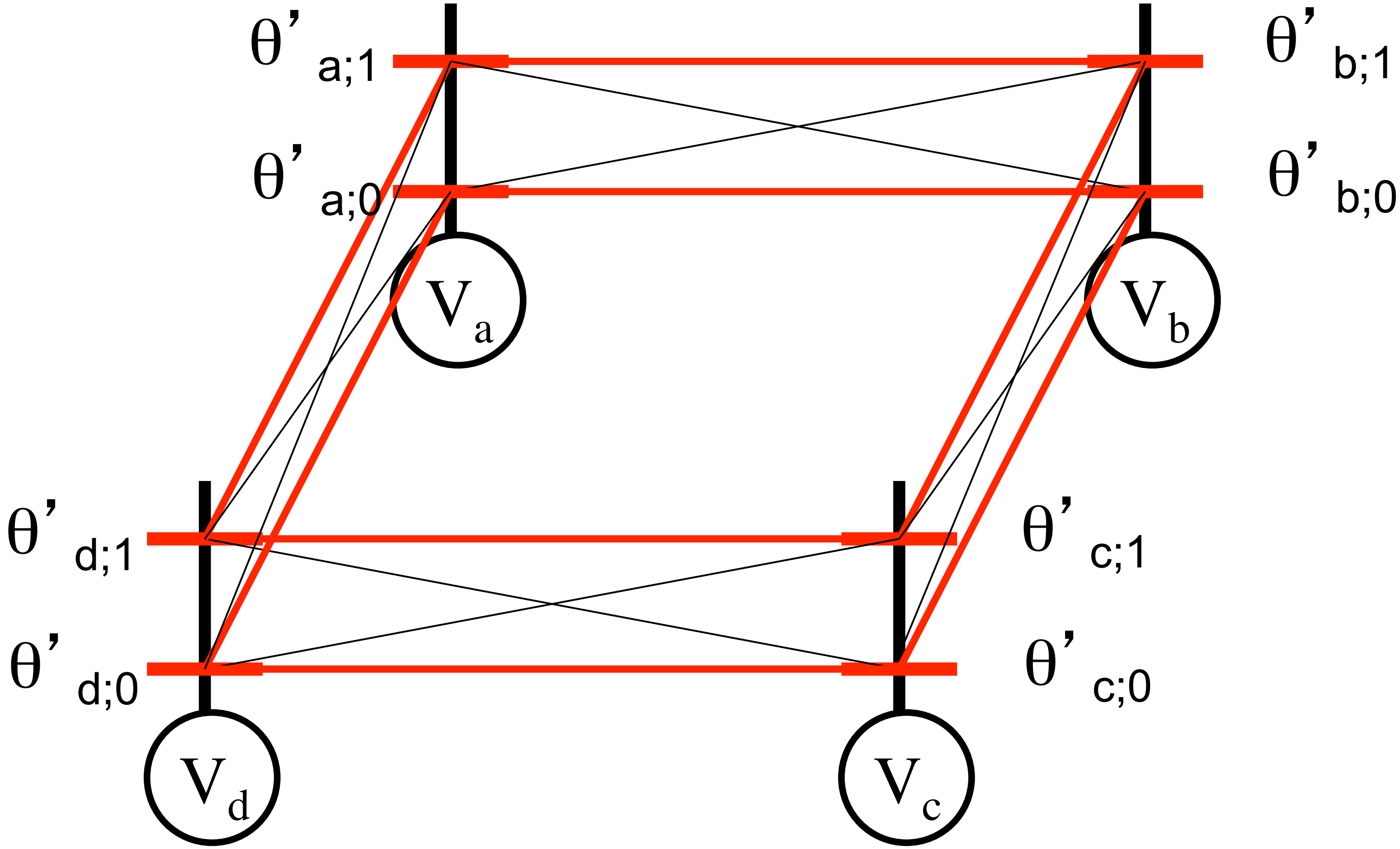
Potentials along the red path add up to 0

Belief Propagation on Cycles



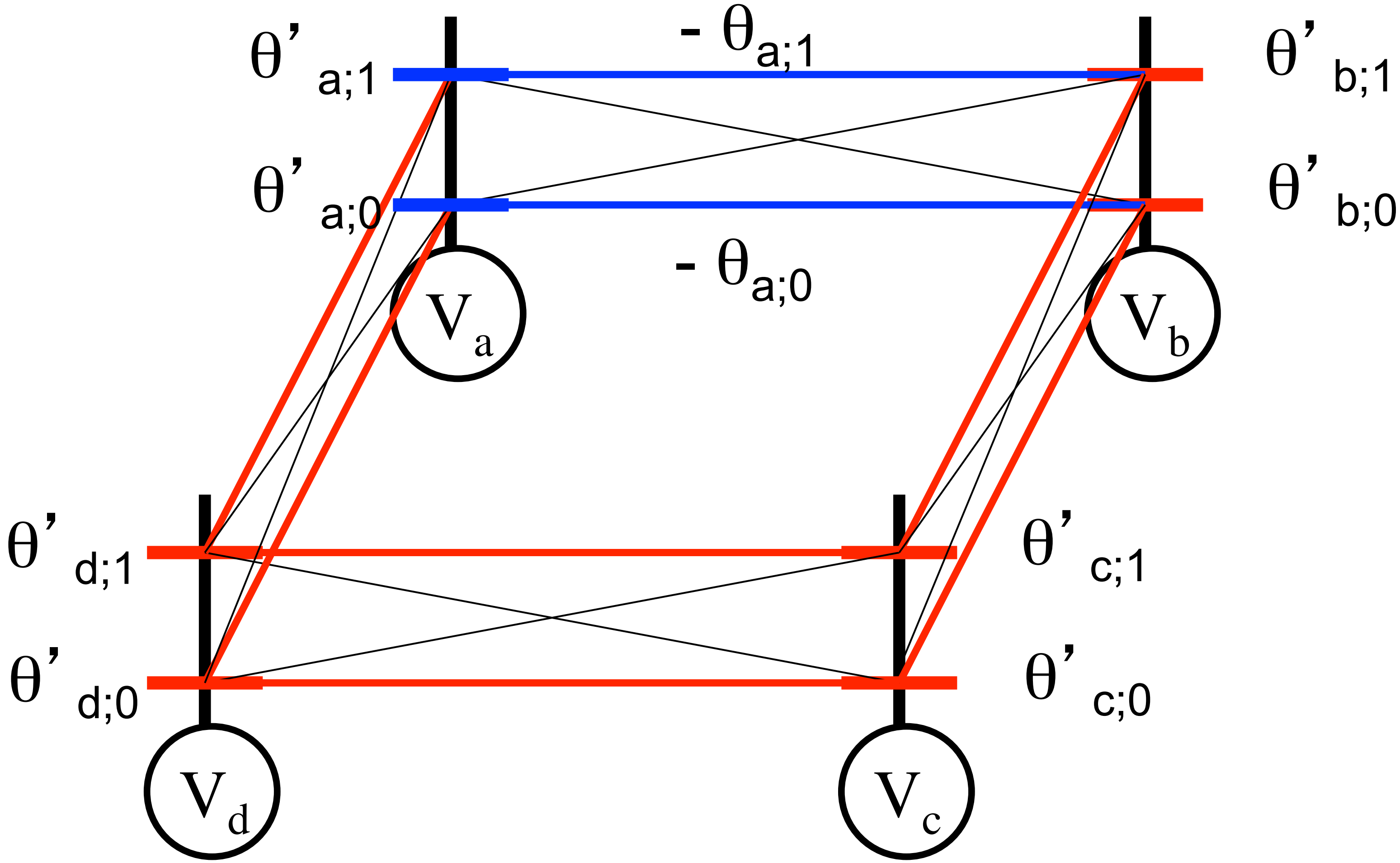
Potentials along the red path add up to 0

Belief Propagation on Cycles



Potentials along the red path add up to 0

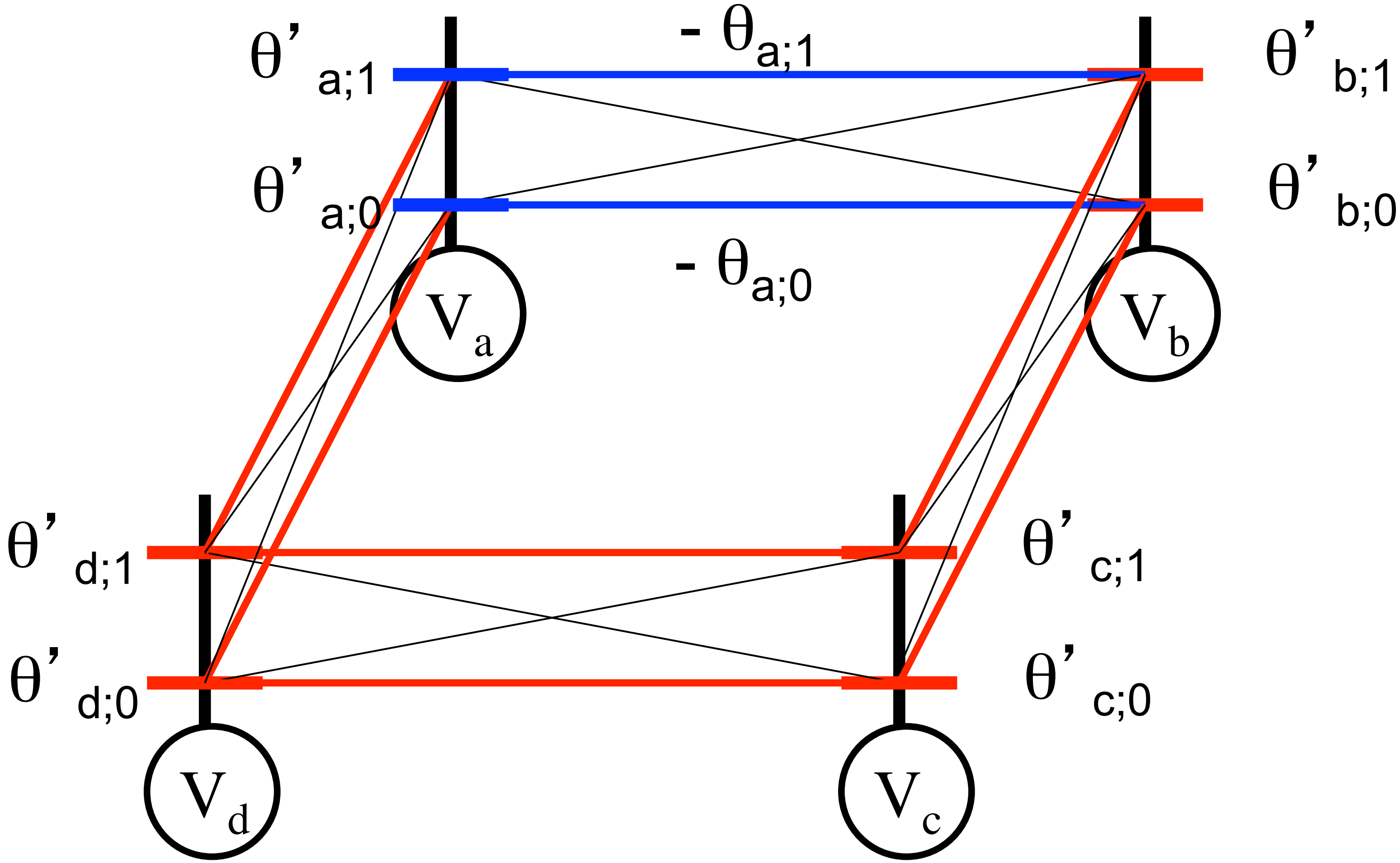
Belief Propagation on Cycles



$\theta'_{a;0} - \theta_{a;0} = q_{a;0}$ $\theta'_{a;1} - \theta_{a;1} = q_{a;1}$

Potentials along the red path add up to 0

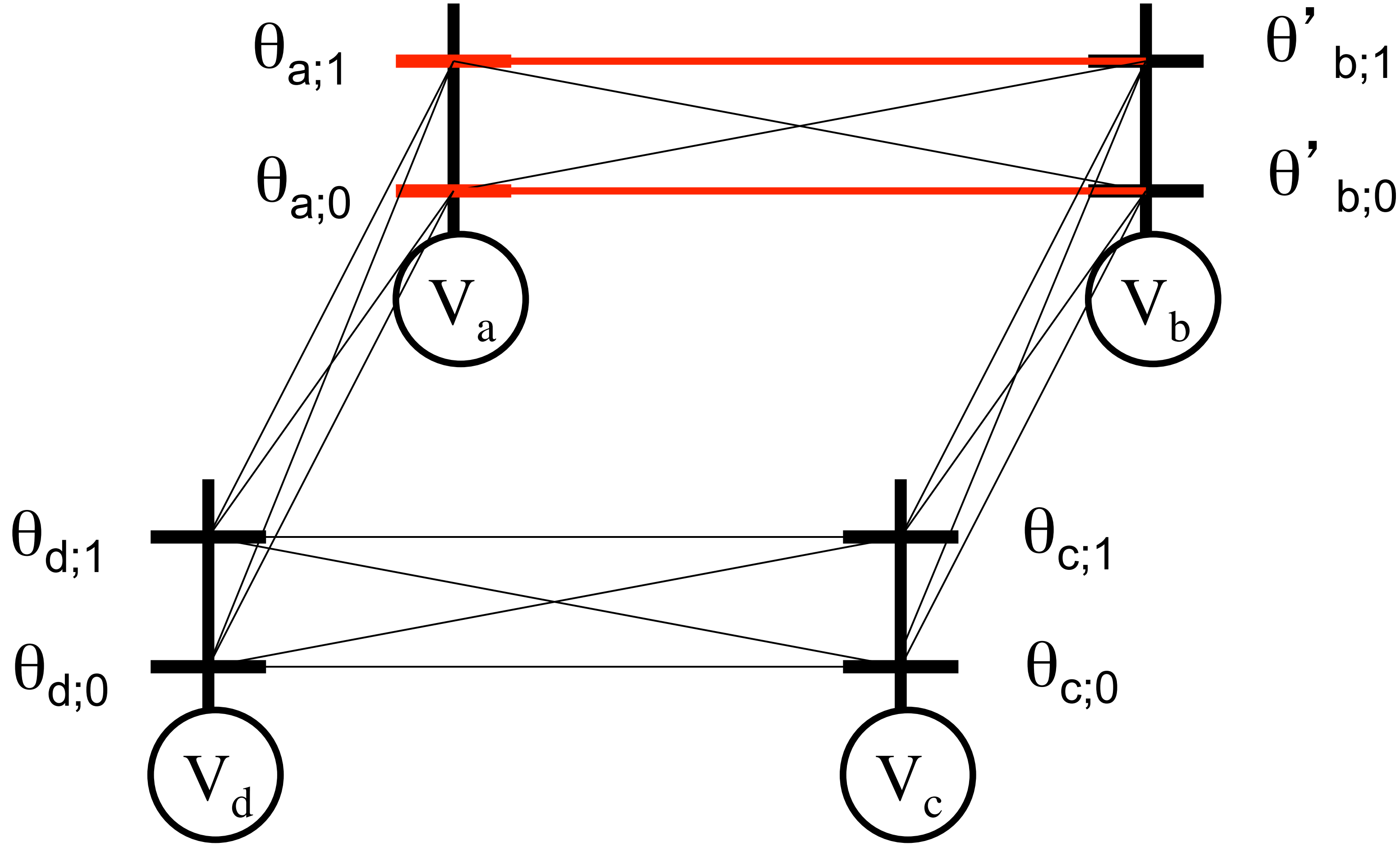
Belief Propagation on Cycles



$\theta'_{a;0} - \theta_{a;0} = q_{a;0}$ $\theta'_{a;1} - \theta_{a;1} = q_{a;1}$

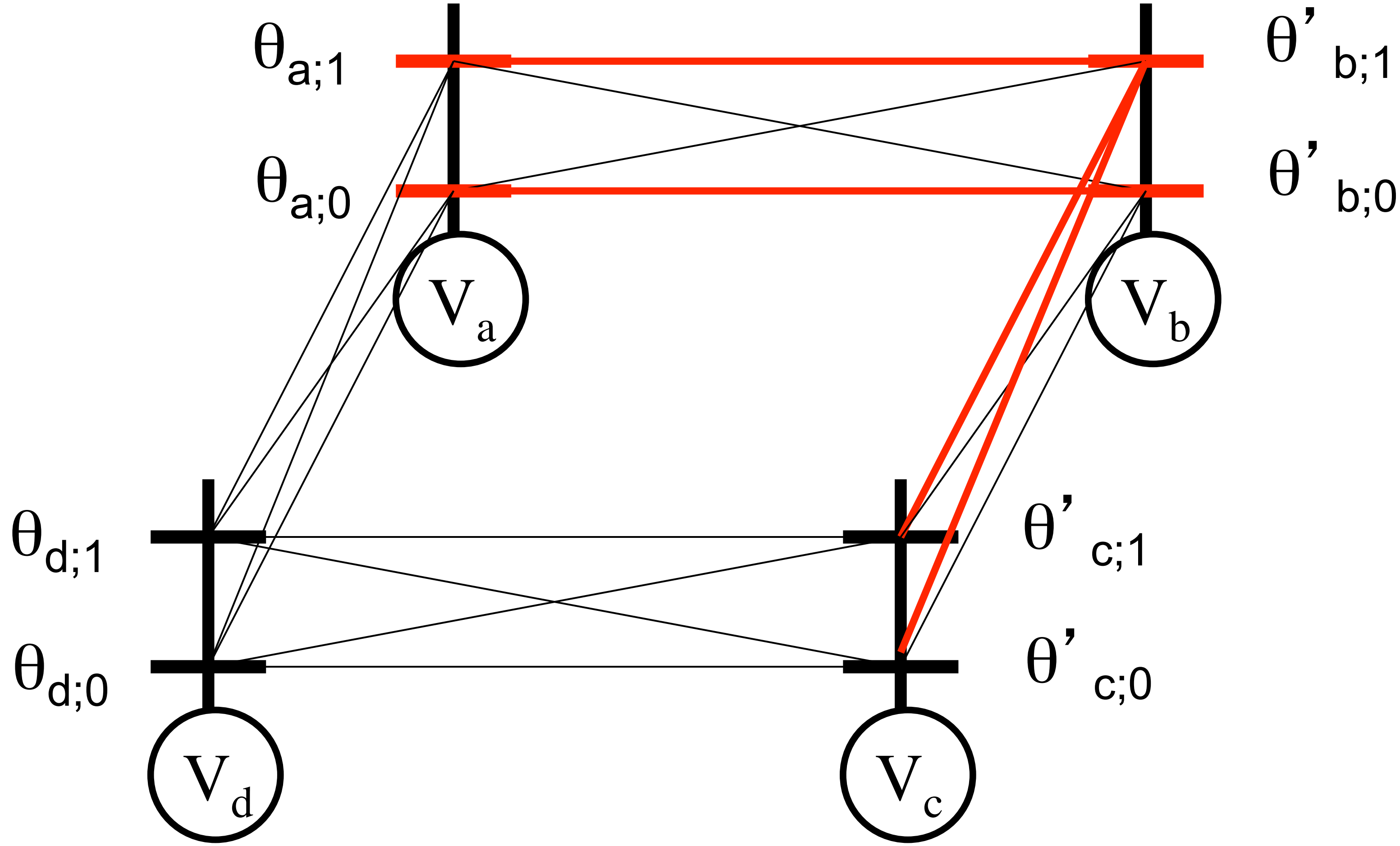
Pick minimum min-marginal. Follow red path.

Belief Propagation on Cycles



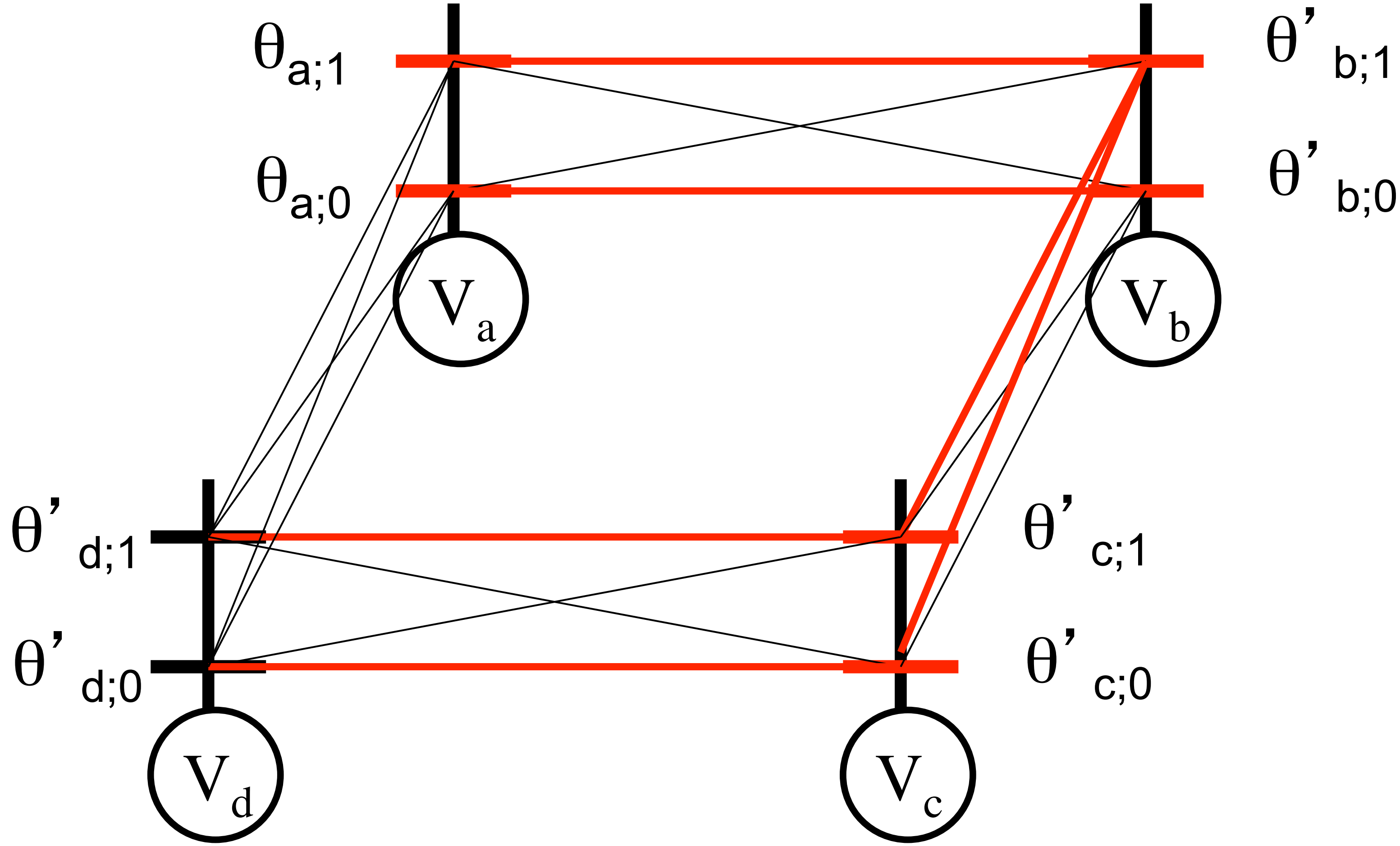
Potentials along the red path add up to 0

Belief Propagation on Cycles



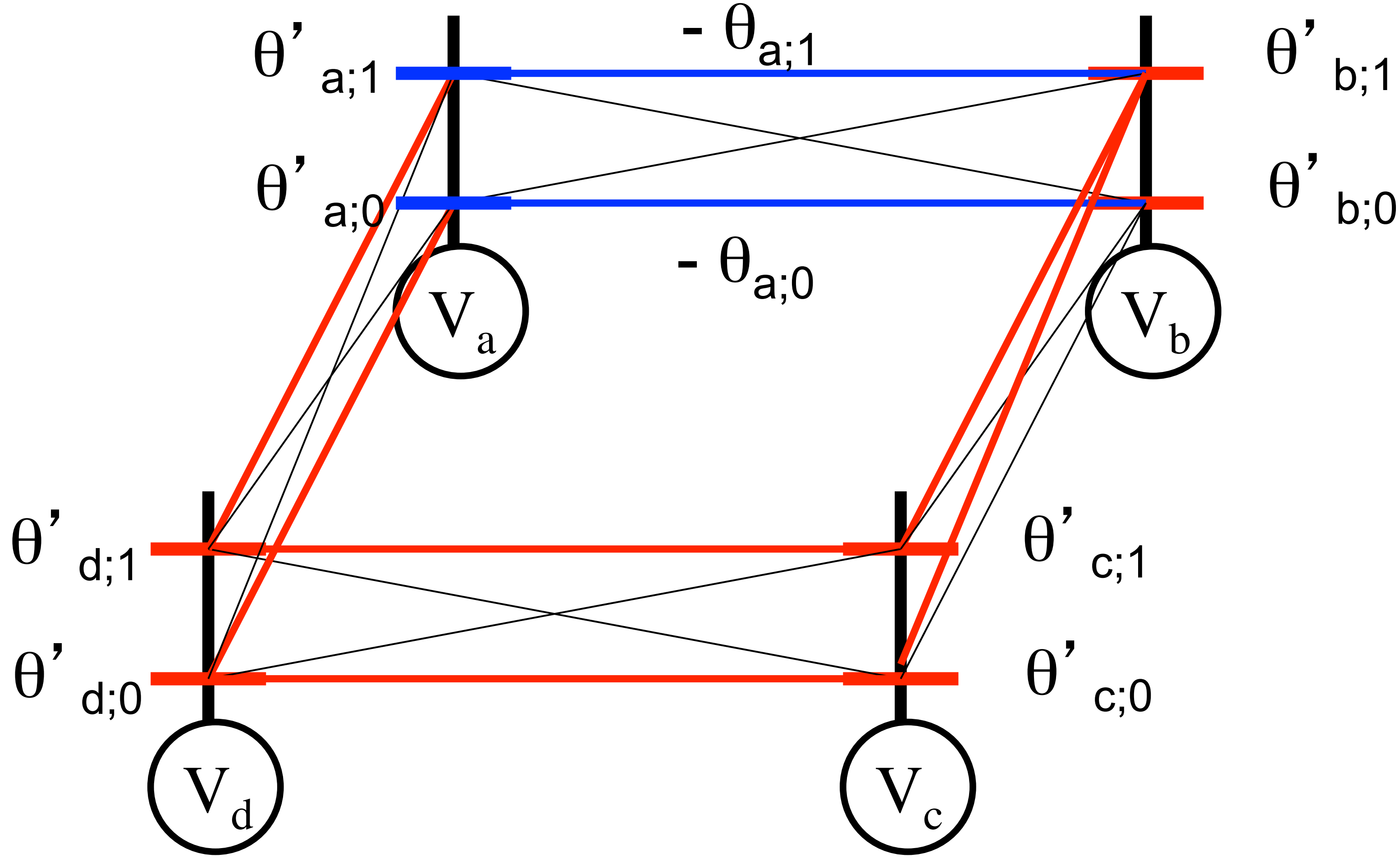
Potentials along the red path add up to 0

Belief Propagation on Cycles



Potentials along the red path add up to 0

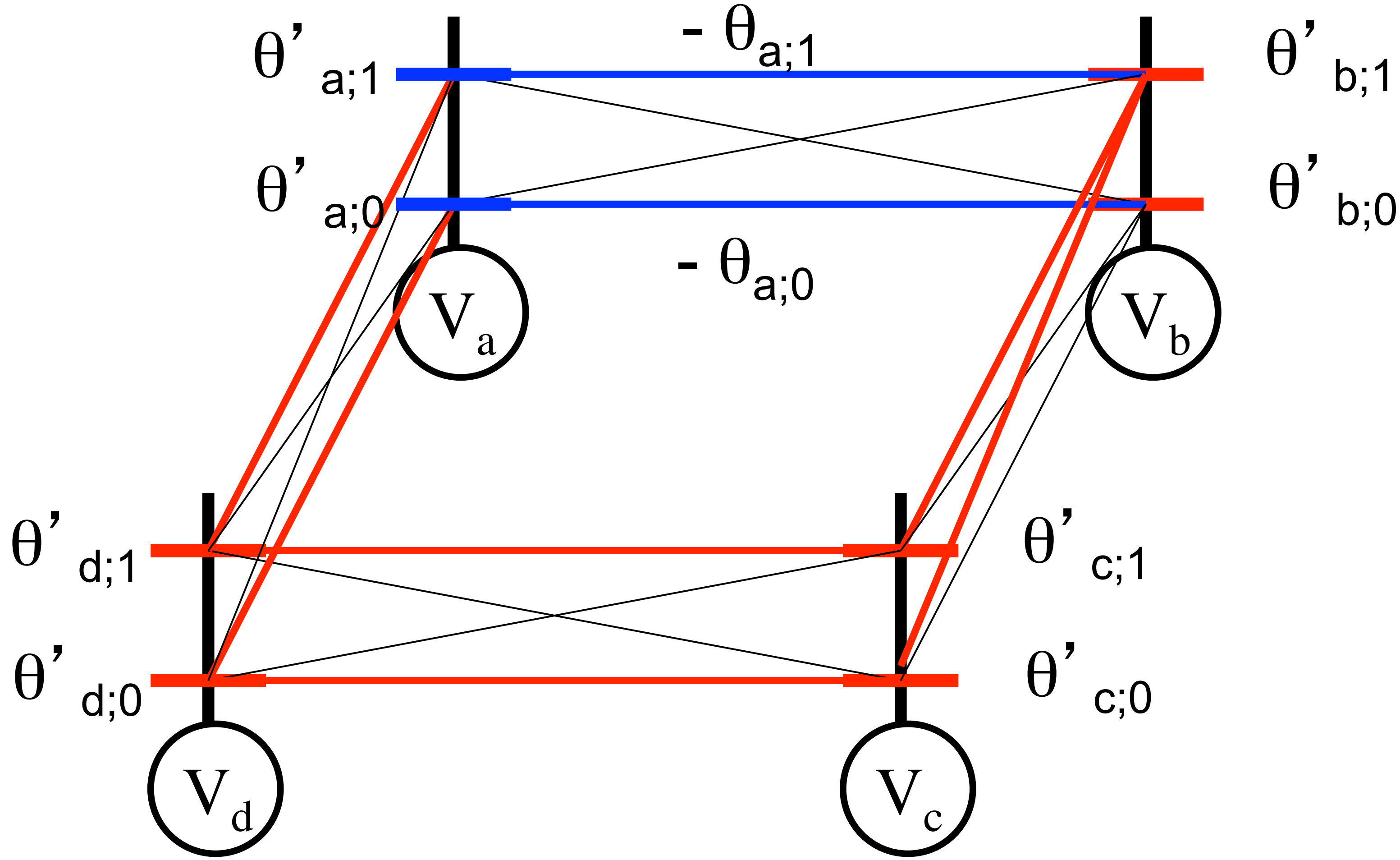
Belief Propagation on Cycles



$$\theta'_{a;1} - \theta_{a;1} = q_{a;1} \leq \theta'_{a;0} - \theta_{a;0} = q_{a;0}$$

Potentials along the red path add up to 0

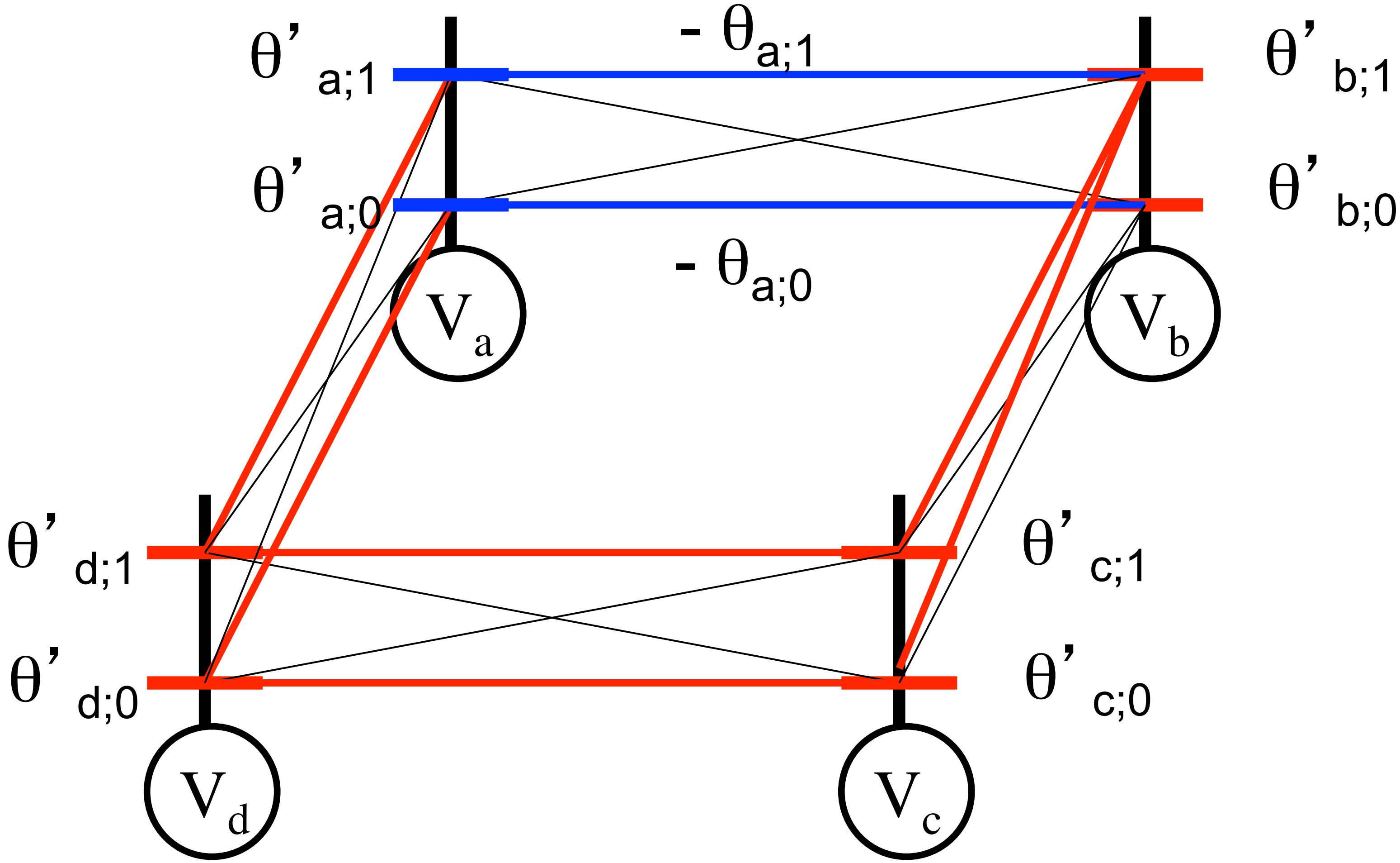
Belief Propagation on Cycles



$\theta'_{a;1} - \theta_{a;1} = q_{a;1} \leq \theta'_{a;0} - \theta_{a;0} = q_{a;0}$

Problem Solved

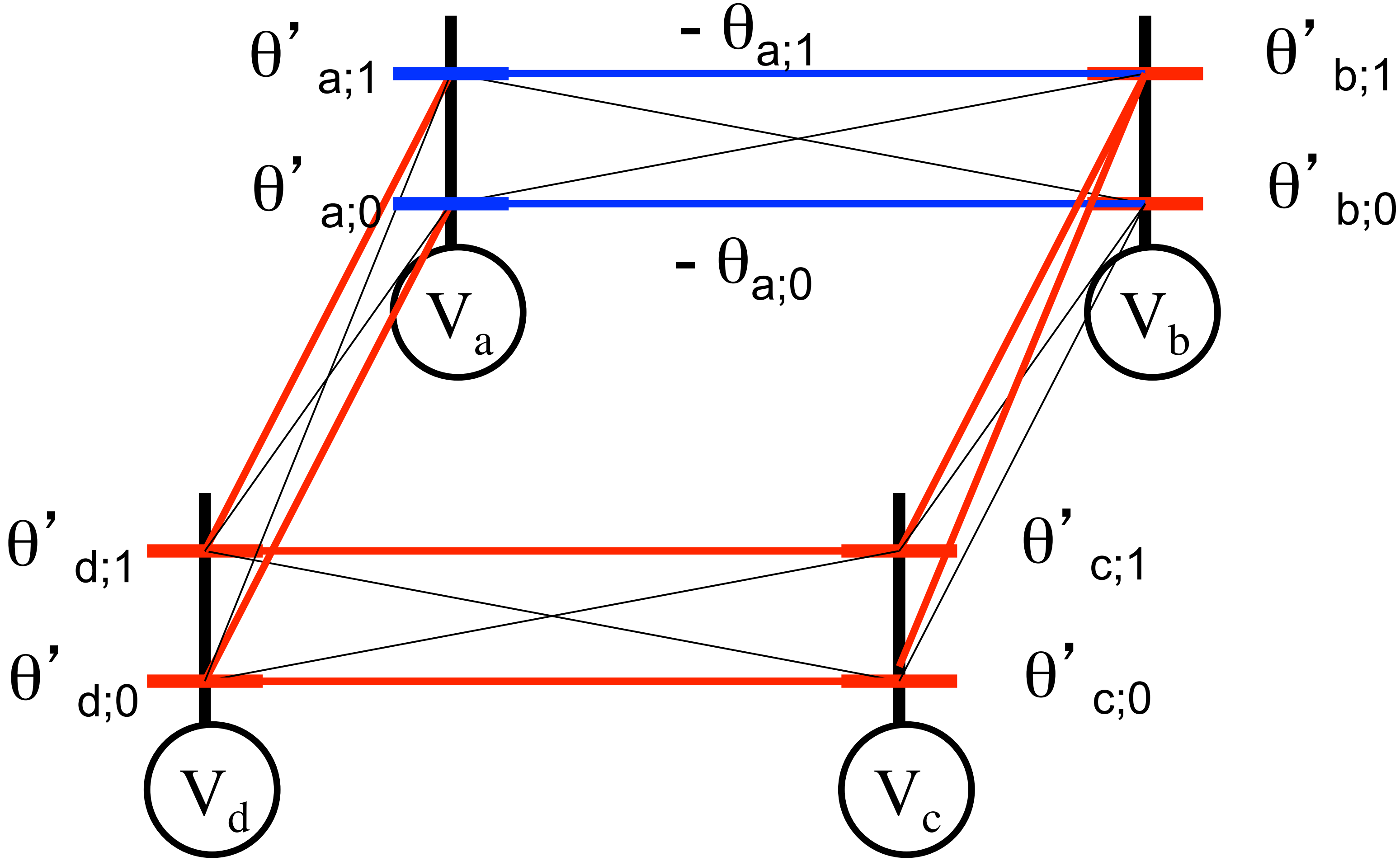
Belief Propagation on Cycles



$\theta'_{a;1} - \theta_{a;1} = q_{a;1} \geq \theta'_{a;0} - \theta_{a;0} = q_{a;0}$

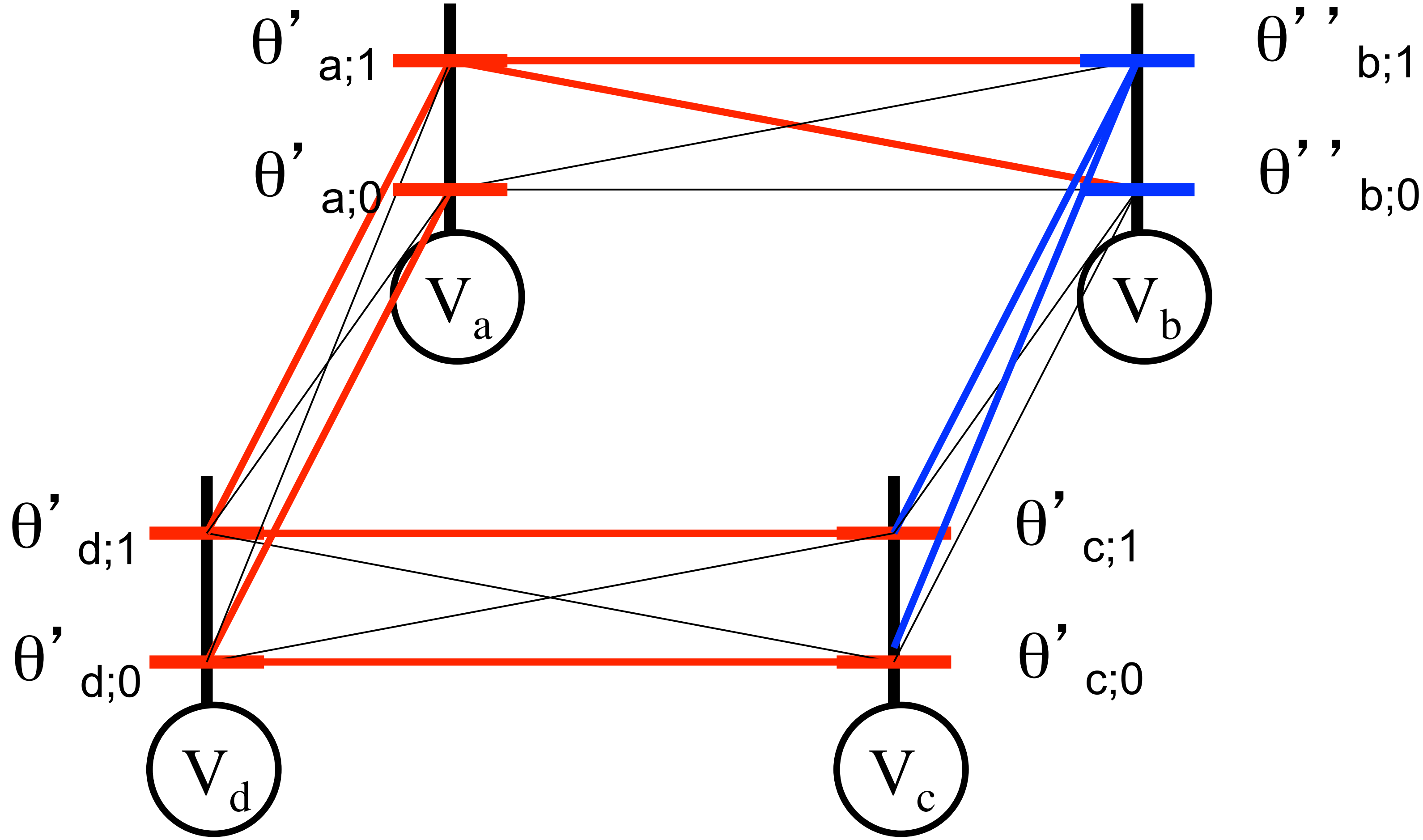
Problem Not Solved

Belief Propagation on Cycles



Reparameterize (a,b) again

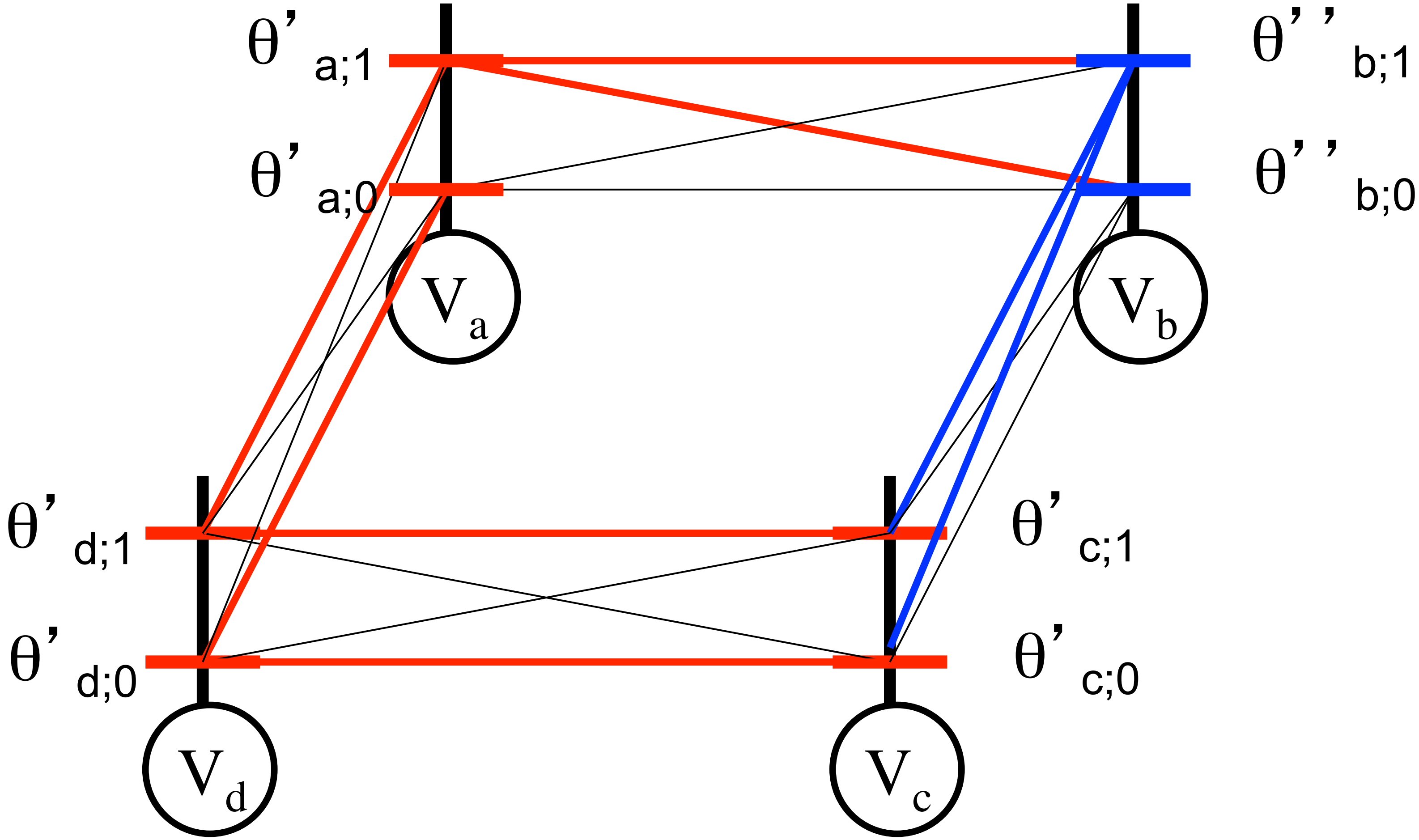
Belief Propagation on Cycles



Reparameterize (a,b) again

But doesn't this overcount some potentials?

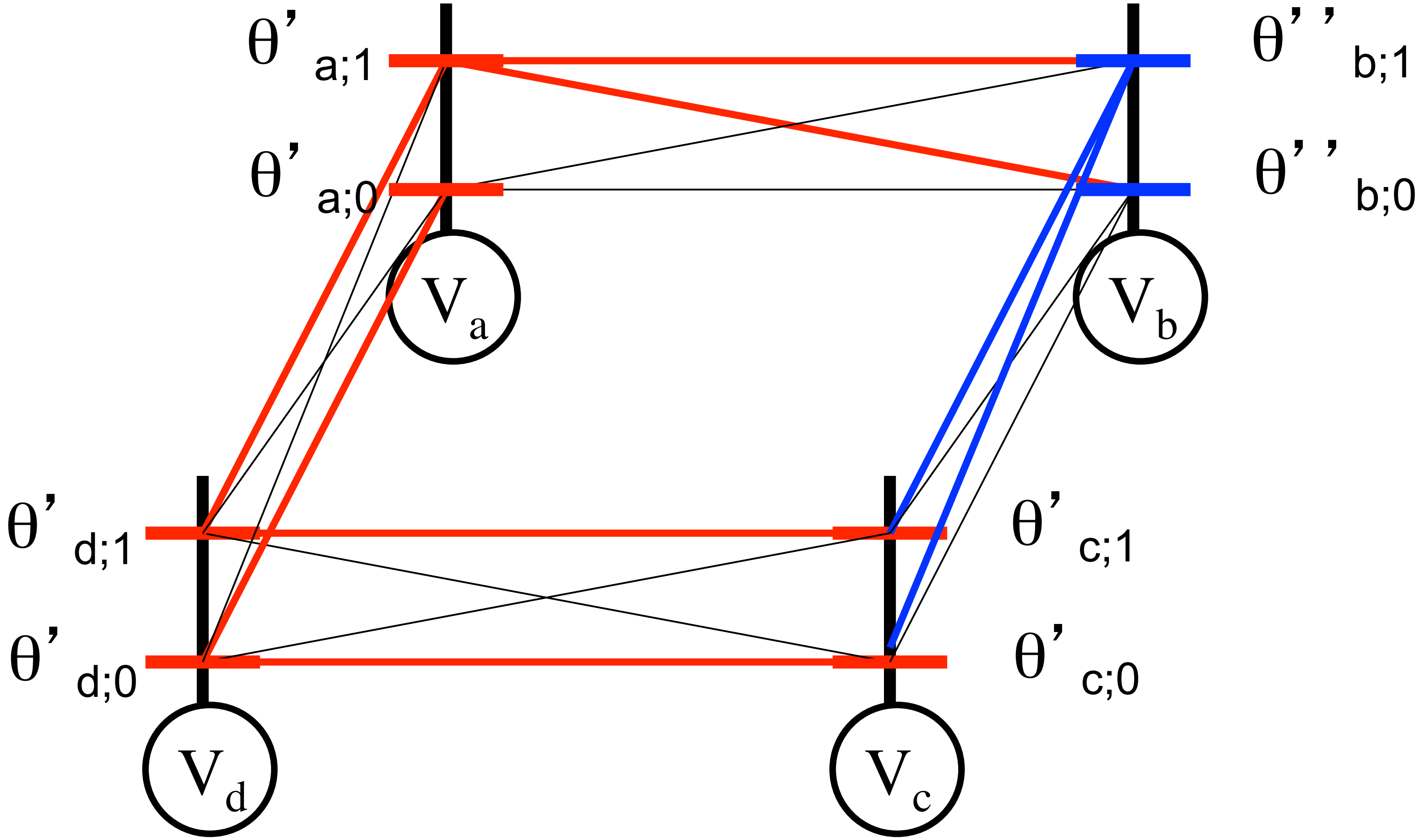
Belief Propagation on Cycles



Reparameterize (a,b) again

Yes. But we will do it anyway

Belief Propagation on Cycles



Keep reparameterizing edges in some order

Hope for convergence and a good solution

Belief Propagation

- Generalizes to any arbitrary random field

- Complexity per iteration ?

$$O(|E||L|^2)$$

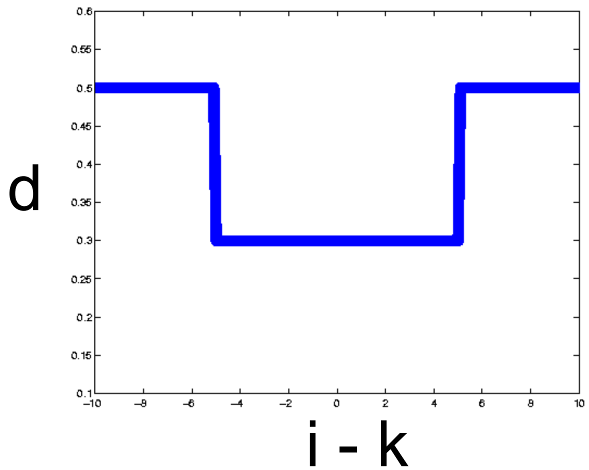
- Memory required ?

$$O(|E||L|)$$

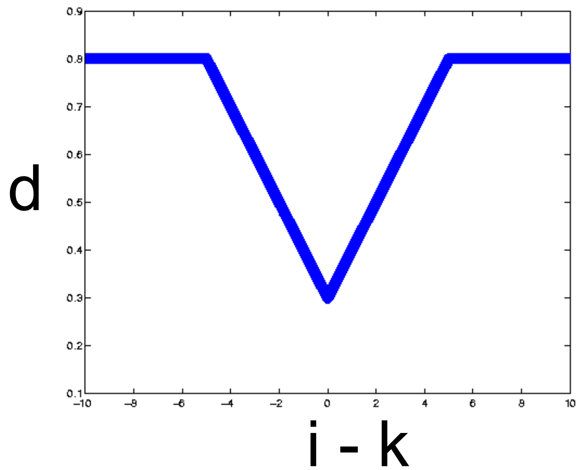
Computational Issues of BP

Complexity per iteration $O(|E||L|^2)$

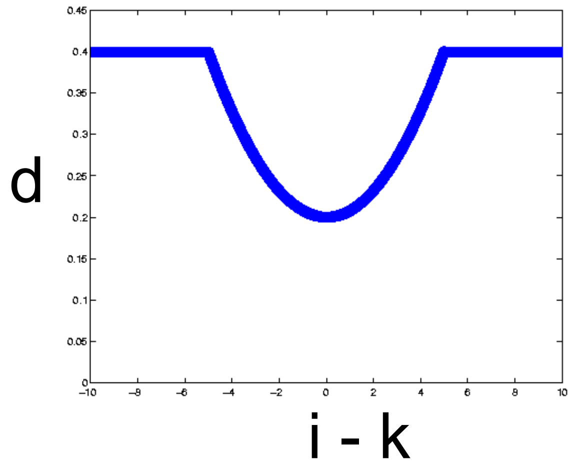
Special Pairwise Potentials $\theta_{ab;ik} = w_{ab}d(|i-k|)$



Potts



Truncated Linear



Truncated Quadratic

$O(|E||L|)$ Felzenszwalb & Huttenlocher, 2004

Computational Issues of BP

Memory requirements **$O(|E||L|)$**

Half of original BP Kolmogorov, 2006

Some approximations exist

Yu, Lin, Super and Tan, 2007

Lasserre, Kannan and Winn, 2007

But memory still remains an issue

Computational Issues of BP

Order of reparameterization

Randomly

In some fixed order

The one that results in maximum change

Residual Belief Propagation

Elidan et al., 2006

Summary of BP

Exact for chains

Exact for trees

Approximate MAP for general cases

Not even convergence guaranteed

So can we do something better?

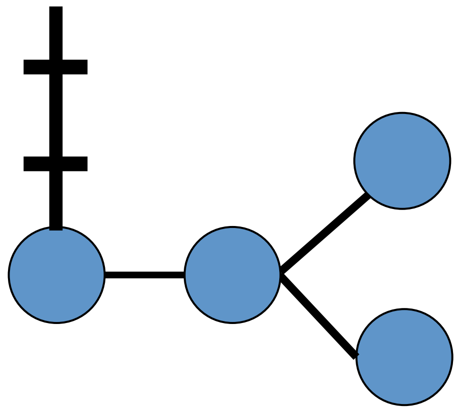
Other alternatives

- Integer linear programming and relaxation

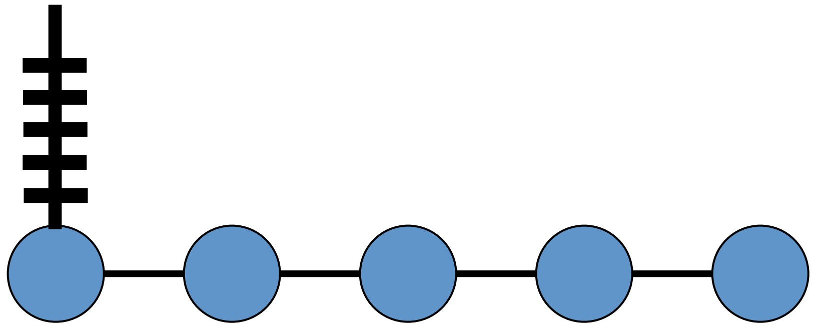
- TRW, Dual decomposition methods

- Extensively studied
  - Schlesinger, 1976
  - Koster et al., 1998, Chekuri et al., ’01, Archer et al., ’04
  - Wainwright et al., 2001, Kolmogorov, 2006
  - Globerson and Jaakkola, 2007, Komodakis et al., 2007
  - Kumar et al., 2007, Sontag et al., 2008, Werner, 2008
  - Batra et al., 2011, Werner, 2011, Zivny et al., 2014

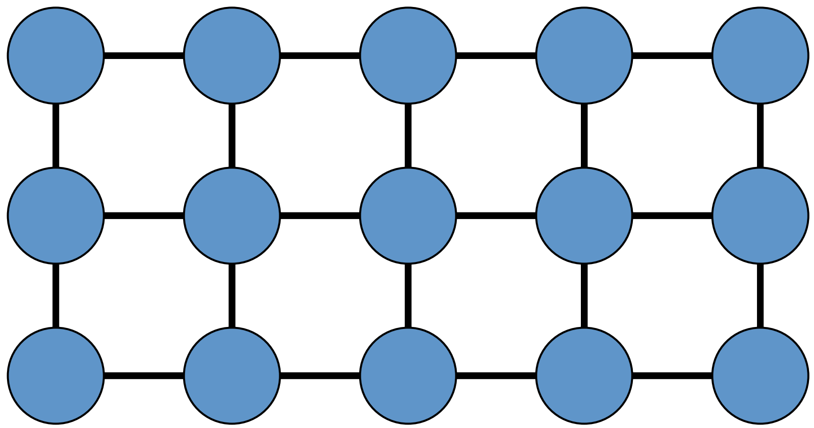
Where do we stand ?



Chain/Tree, 2-label: Use BP



Chain/Tree, multi-label: Use BP



Grid graph: Use TRW, dual decomposition, relaxation

Note on Dynamic Programming

Dynamic Programming (DP)

- DP ≈ "careful brute force"
- DP ≈ recursion + memoization + guessing
- Divide the problem into subproblems that are connected to the original problem
- Graph of subproblems has to be acyclic (DAG)
- Time = #subproblems · time/subproblem

5 easy steps of DP

Analysis:

1. Define subproblems — #subproblems
2. Guess part of solution — #choices
3. Relate subproblems (recursion) — time/subproblem
4. Recurse + memoize — time

   OR build DP table bottom-up

   - check subprobs be acyclic / topological order
5. Solve original problem — extra time

5 easy steps of DP

	Fibonacci	Shortest paths
1. Subproblems	F_k, $1 \le k \le n$	$\delta_k(s, v)$, $v \in V$, $0 \le k \le V$
#subproblems	n	V^2
2. Guessing	F_{n-1}, F_{n-2}	edges coming into v
#choices	1	indegree(v)
3. Recurrence	$F_n = F_{n-1} + F_{n-2}$	$\delta_k(s, v) = \min\{ \delta_{k-1}(s, u) + w(u, v) \mid (u, v) \in E \}$
time/subproblem	O(1)	O(indegree(v))
4. Topological order	for k = 1, …, n	for k = 0, 1, …, V - 1 for v ∈ V
total time	O(n)	O(V E)
5. Original problem	F_n	$\delta_{V-1}(s, v)$
extra time	O(1)	O(1)