Graphical Models
Discrete Inference and Learning

MVA

2023 – 2024

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

Lecturers

Karteek Alahari

Demian Wassermann

Email: <firstname>.lastname@inria.fr

Organization

- 7 lectures of 3 hours each
  - Today + 30/1, 6/2, 13/2, 5/3, 12/3, 19/3

- 13:30 – 16:45 with a short break or two
- Last lecture: 19th March

- Subscribe to the mailing list: https://sympa.inria.fr/sympa/subscribe/grmdil

Requirements

- Solid understanding of mathematical models
  - Linear algebra
  - Integral transforms
  - Differential equations

- Ideally, a basic course in discrete optimization

Topics covered

- Basic concepts, Bayesian networks, Markov random fields
- Inference algorithms: belief propagation, tree-reweighted message passing, graph cuts, move-making algorithms, Parameter learning
- Deep learning in graphical models, graph neural networks, other recent advances
- Causality

Evaluation

- Projects
- In groups of at most 3 people
- Report and presentation – Dates TBD (last week of March)
- Topics: your own or see list next week

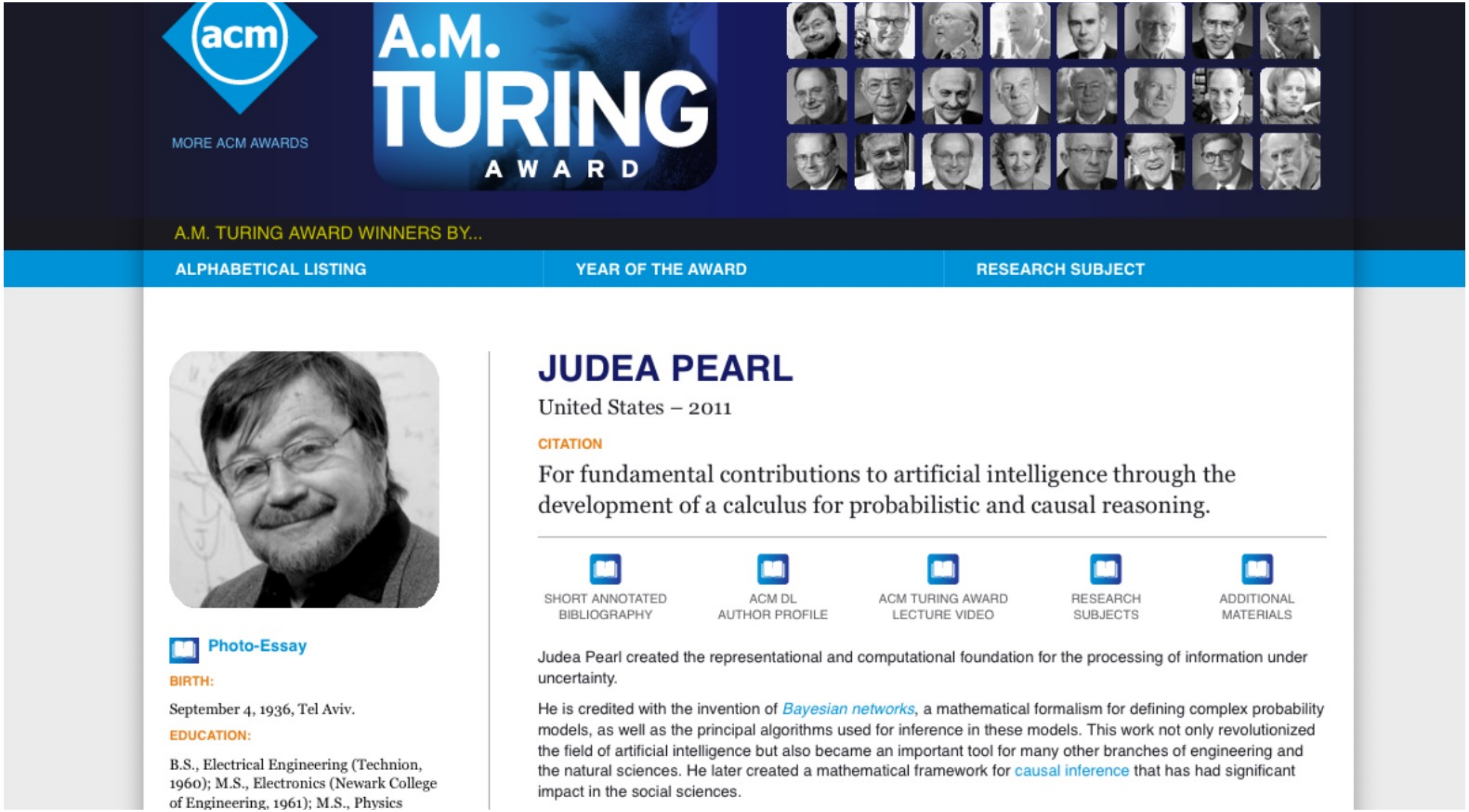
What you will learn?

- Fundamental methods
- Real-world applications
- Also, pointers to using these methods in your work

Your tasks

- Following the lectures and participating actively
- Reading the literature
- Doing well in the project

Graphical Models ?



Slide courtesy: Dhruv Batra

What this class is about?

- Making **global** predictions from **local** observations

Inference

- Learning such models from large quantities of data

Learning

Motivation

- Consider the example of medical diagnosis

Predisposing factors
Symptoms
Test results

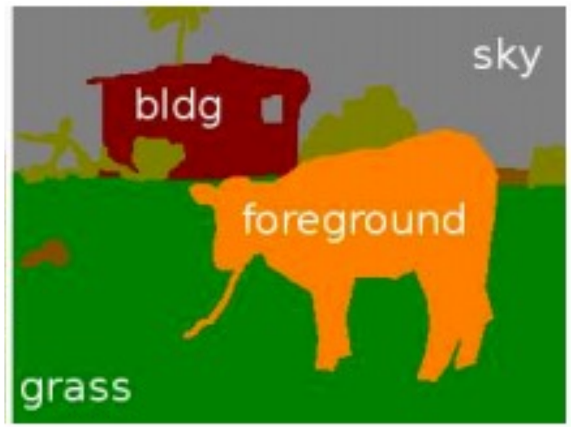
↓

Diseases
Treatment outcomes

Slide inspired by PGM course, Daphne Koller

Motivation

- A very different example: image segmentation



e.g., [He et al., 2004; Shotton et al., 2006; Gould et al., 2009]

Millions of pixels
Colours / features

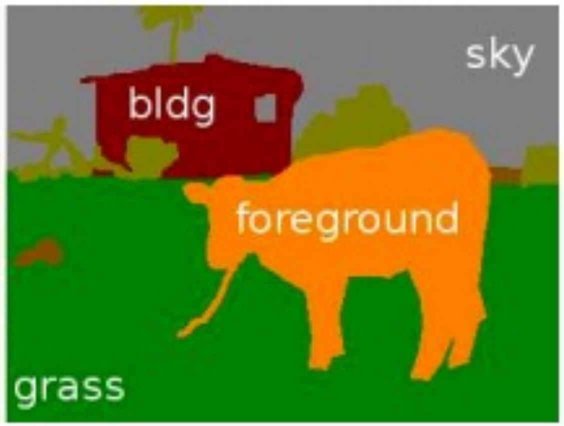
↓

Pixel labels
{building, grass, cow, sky}

Slide inspired by PGM course, Daphne Koller

Motivation

- What do these two problems have in common?



Slide inspired by PGM course, Daphne Koller

Motivation

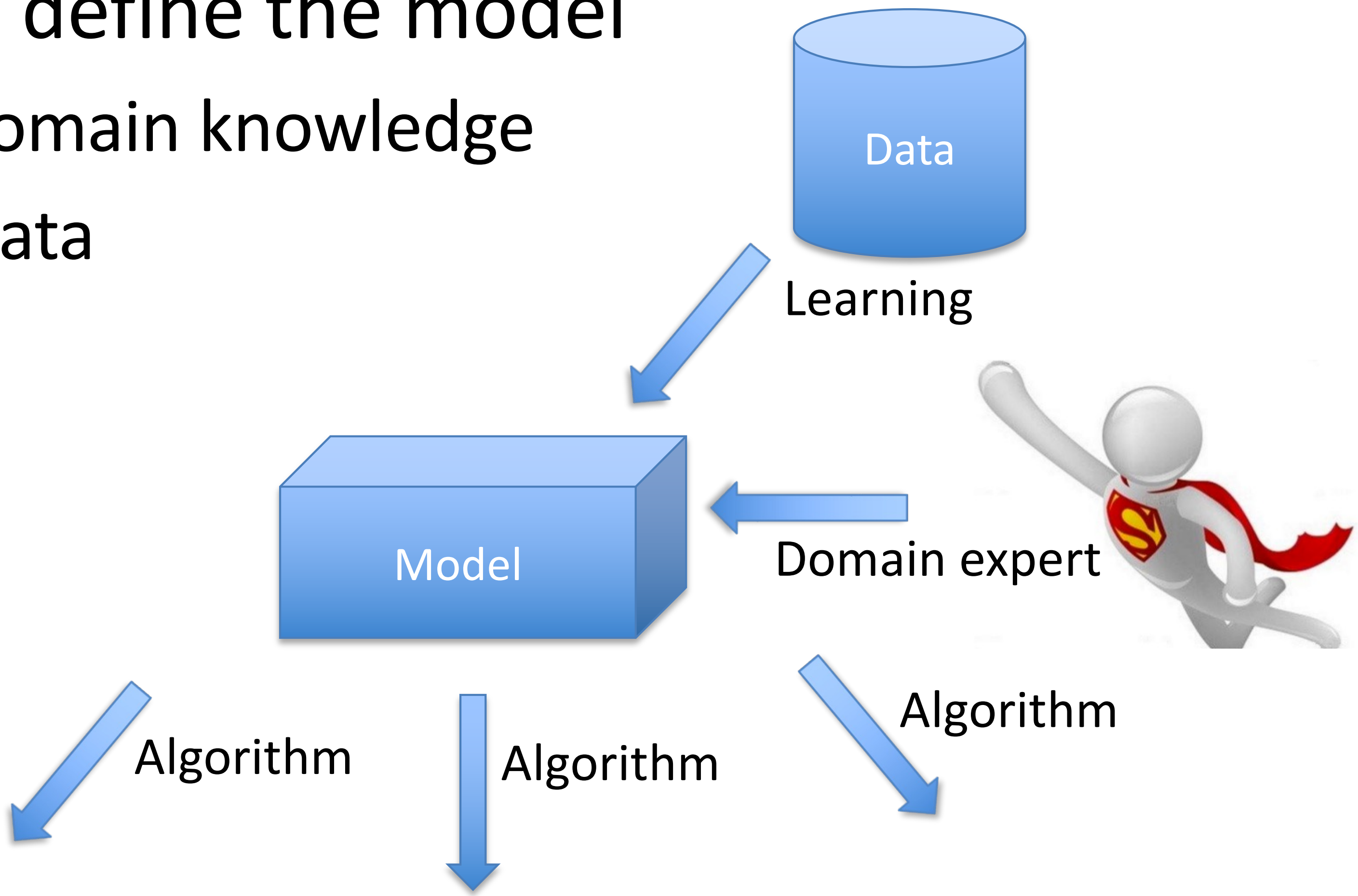
- What do these two problems have in common?
  - Many variables
  - Uncertainty about the correct answer

Graphical Models (or Probabilistic Graphical Models) provide a framework to address these problems

Slide inspired by PGM course, Daphne Koller

(Probabilistic) Graphical Models

- First, it is a model: a declarative representation
- Can also define the model
  - with domain knowledge
  - from data



Slide inspired by PGM course, Daphne Koller

(Probabilistic) Graphical Models

- Why probabilistic ?
- To model uncertainty
- Uncertainty due to:
  - Partial knowledge of state of the world
  - Noisy observations
  - Phenomena not observed by the model
  - Inherent stochasticity

Slide inspired by PGM course, Daphne Koller

(Probabilistic) Graphical Models

- Probability theory provides
  - Standalone representation with clear semantics
  - Reasoning patterns (conditioning, decision making)
  - Learning methods

Slide inspired by PGM course, Daphne Koller

(Probabilistic) Graphical Models

- Why graphical ?
- Intersection of ideas from probability theory and computer science
  - To represent large number of variables

Predisposing factors
Symptoms
Test results

Millions of pixels
Colours / features

Random variables $\mathbf{Y_1, Y_2, \ldots, Y_n}$

Goal: capture uncertainty through joint distribution $P(Y_1,\ldots,Y_n)$

Slide inspired by PGM course, Daphne Koller

(Probabilistic) Graphical Models

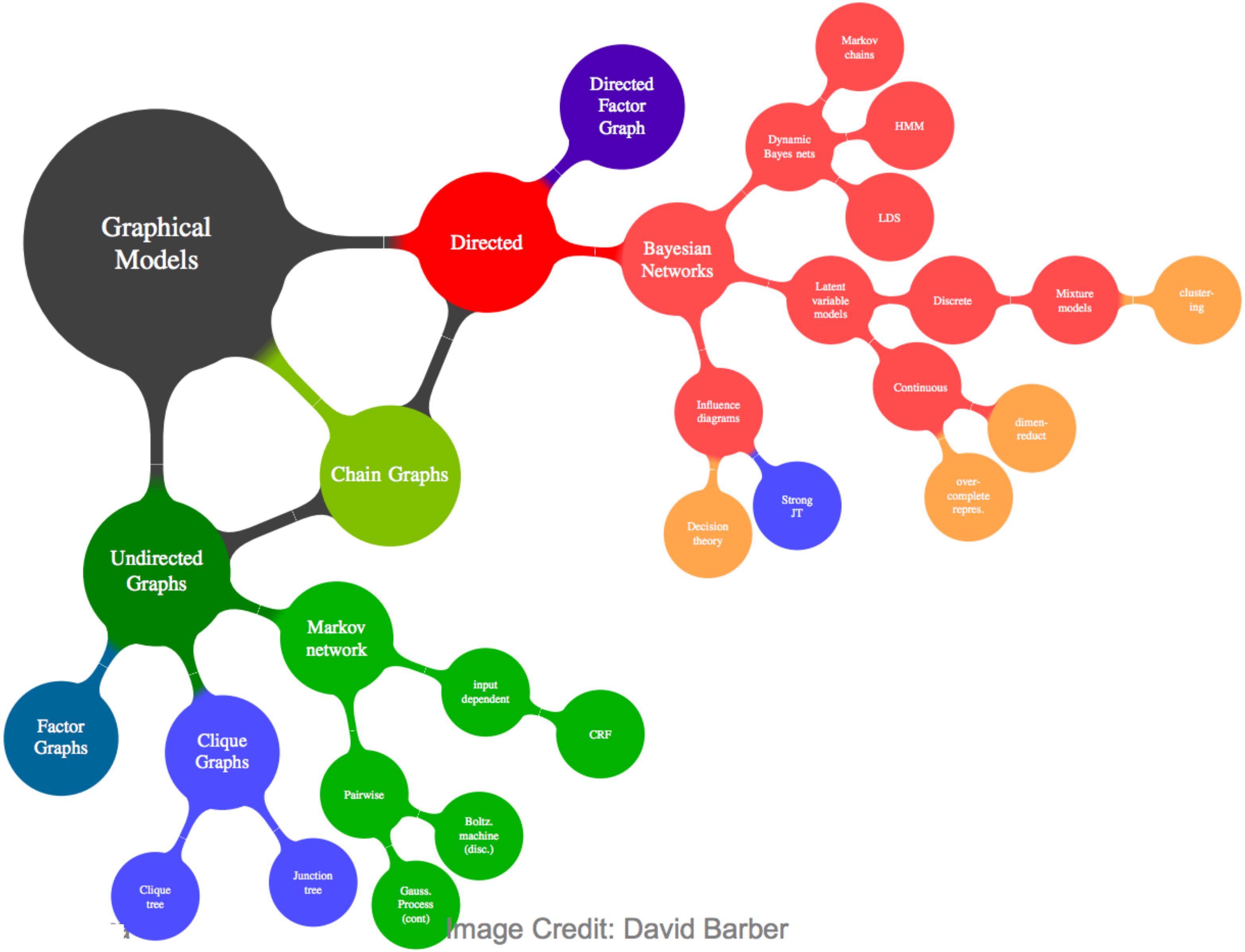
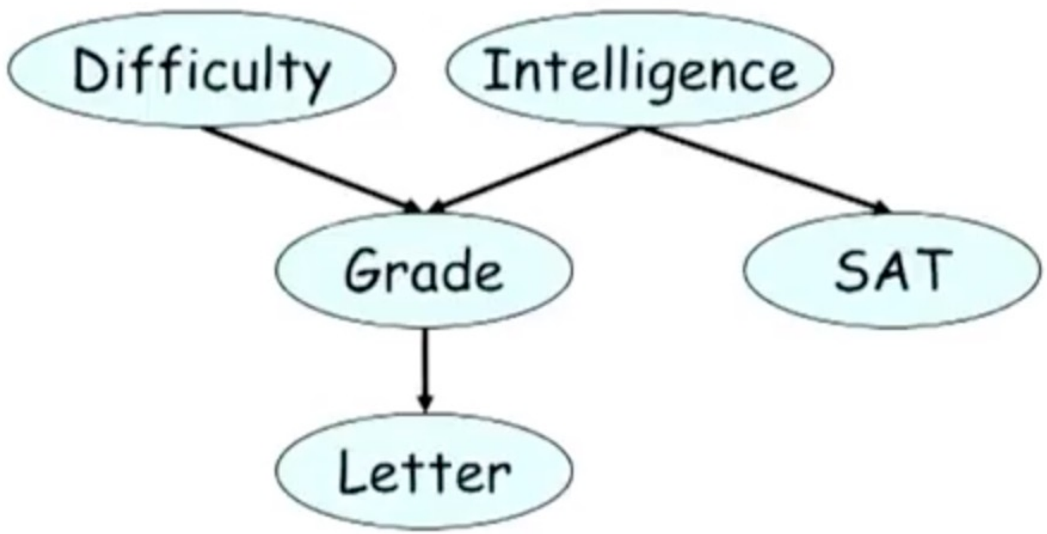


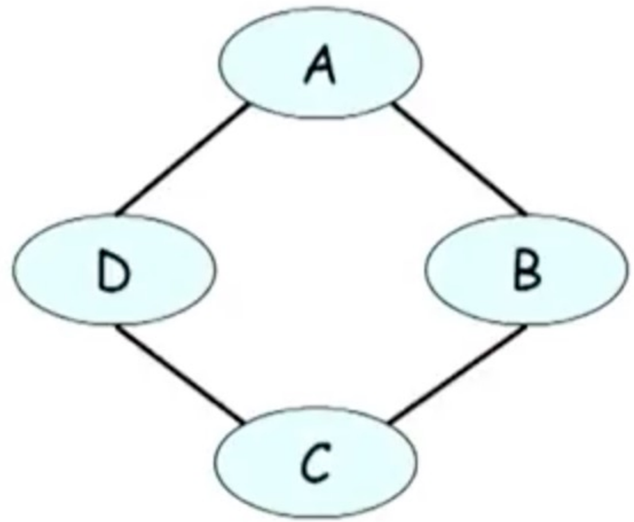
Image Credit: David Barber

(Probabilistic) Graphical Model

- Examples



Bayesian network
(directed graph)

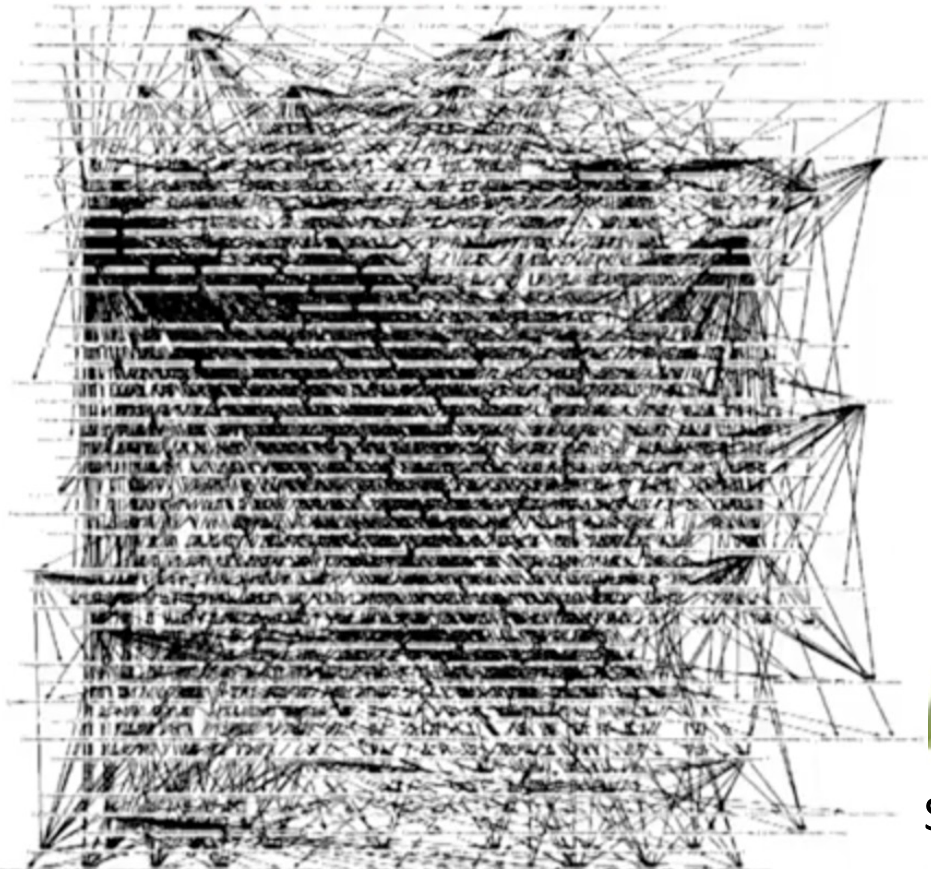


Markov network
(undirected graph)

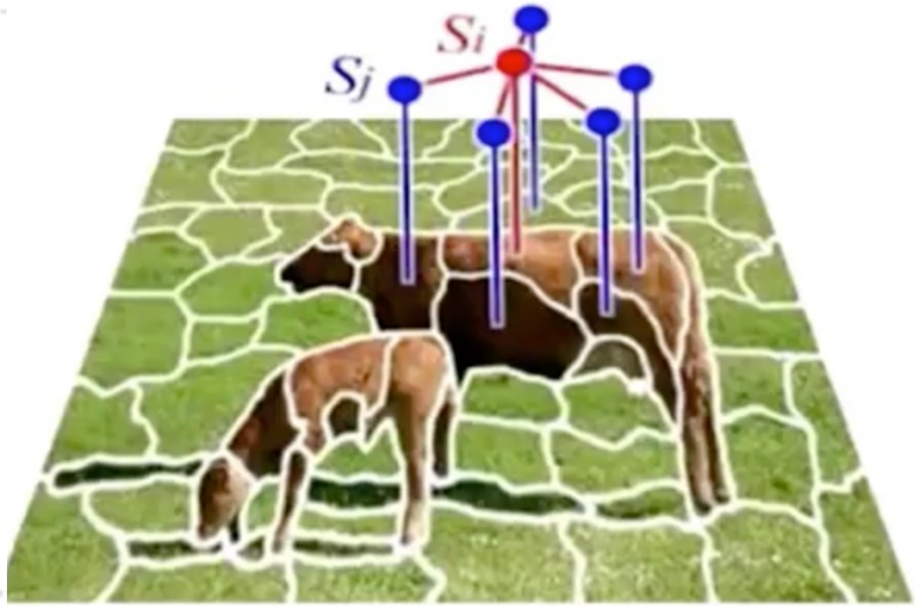
Figure courtesy: D. Koller

(Probabilistic) Graphical Model

- Examples



Diagnosis network: Pradhan et al., UAI'94



Segmentation network (Courtesy D. Koller)

(Probabilistic) Graphical Model

- Intuitive & compact data structure
- Efficient reasoning through general-purpose algorithms
- Sparse parameterization
  - Through expert knowledge, or
  - Learning from data

Slide inspired by PGM course, Daphne Koller

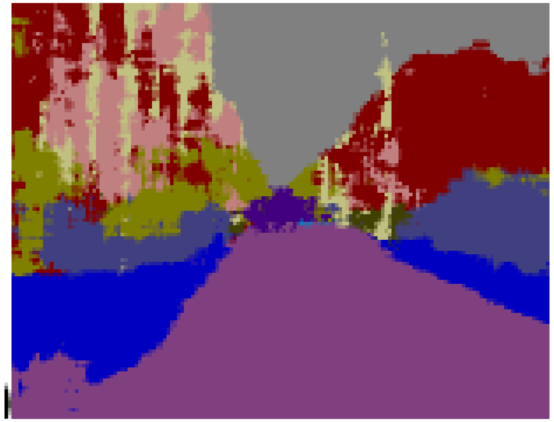
(Probabilistic) Graphical Model

- Many many applications
  - Medical diagnosis
  - Fault diagnosis
  - Natural language processing
  - Traffic analysis
  - Social network models
  - Message decoding
  - Computer vision: segmentation, 3D, pose estimation
  - Speech recognition
  - Robot localization & mapping

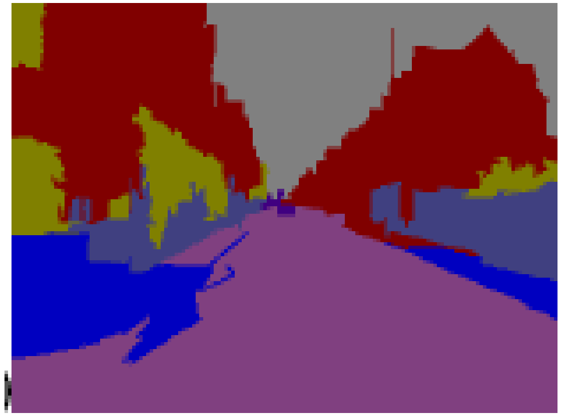
Slide courtesy: PGM course, Daphne Koller

Image segmentation

Image



No graphical model

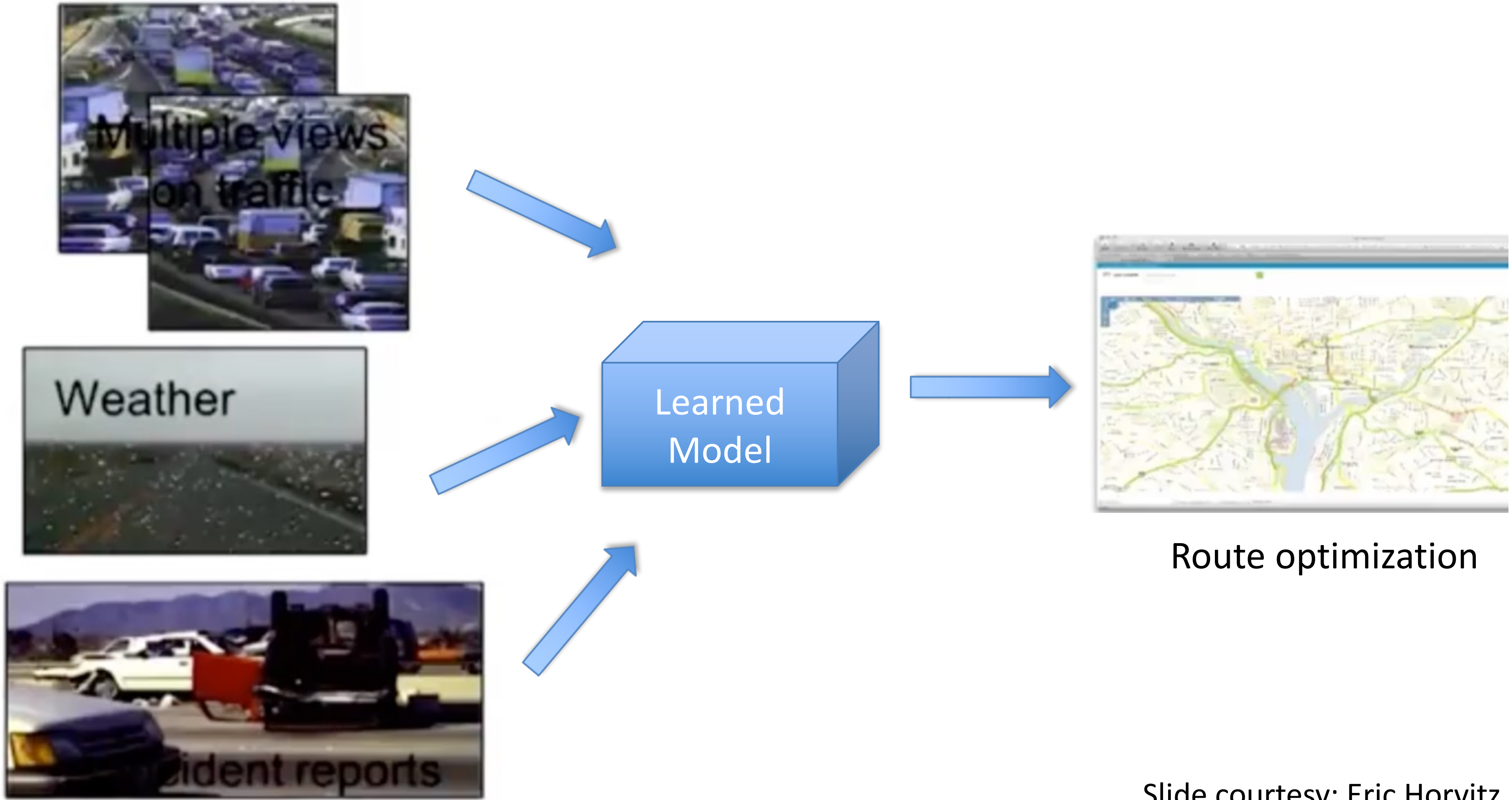


With graphical model

Sturgess et al., 2009

Multi-sensor integration: Traffic

- Learn from historical data to make predictions



Slide courtesy: Eric Horvitz

Going global: Local ambiguity

- Text recognition

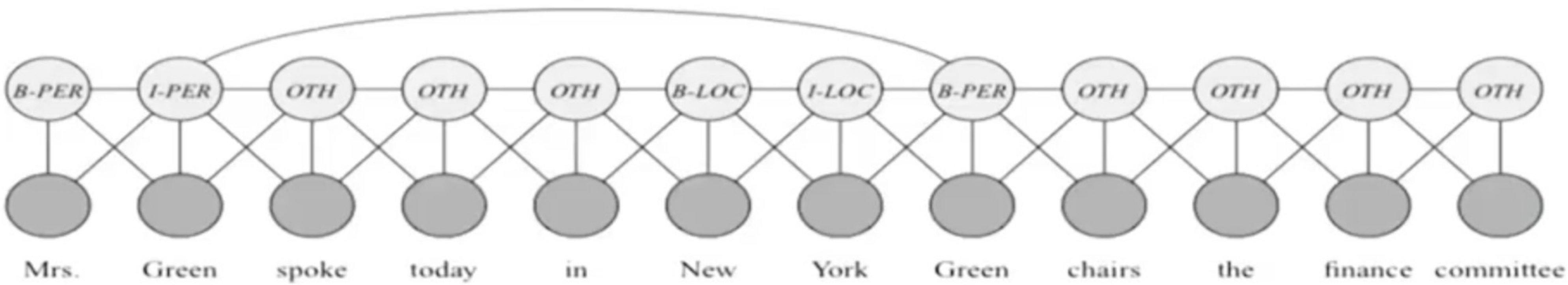
Smyth et al., 1994

Slide courtesy: Dhruv Batra

Going global: Local ambiguity

- Textual information extraction

e.g., Mrs. Green spoke today in New York. Green chairs the financial committee.



Slide courtesy: PGM course, Daphne Koller

Overview

- Representation
  - How do we store $P(Y_1,\ldots Y_n)$
  - Directed and undirected (model implications/assumptions)
- Inference
  - Answer questions with the model
  - Exact and approximate (marginal/most probable estimate)
- Learning
  - What model is right for data
  - Parameters and structure

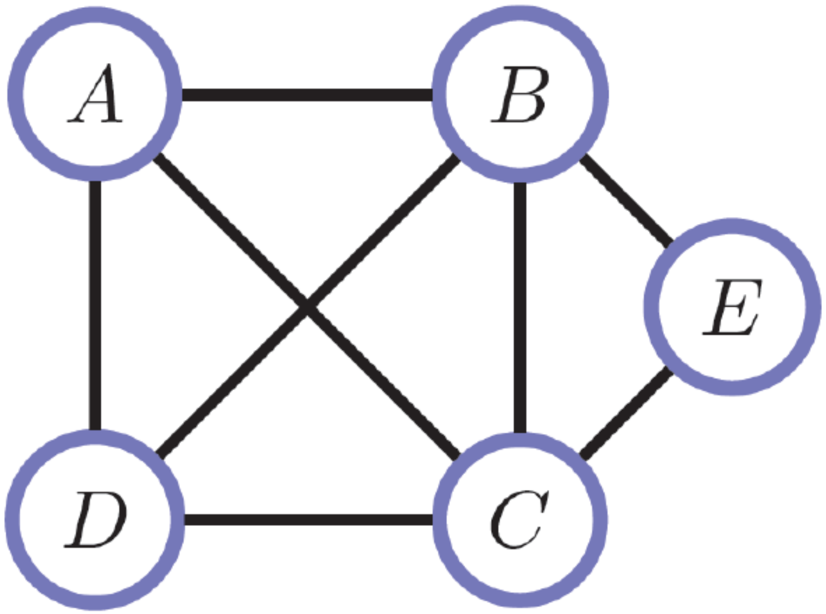
Slide inspired by D. Batra, D. Koller ’s courses

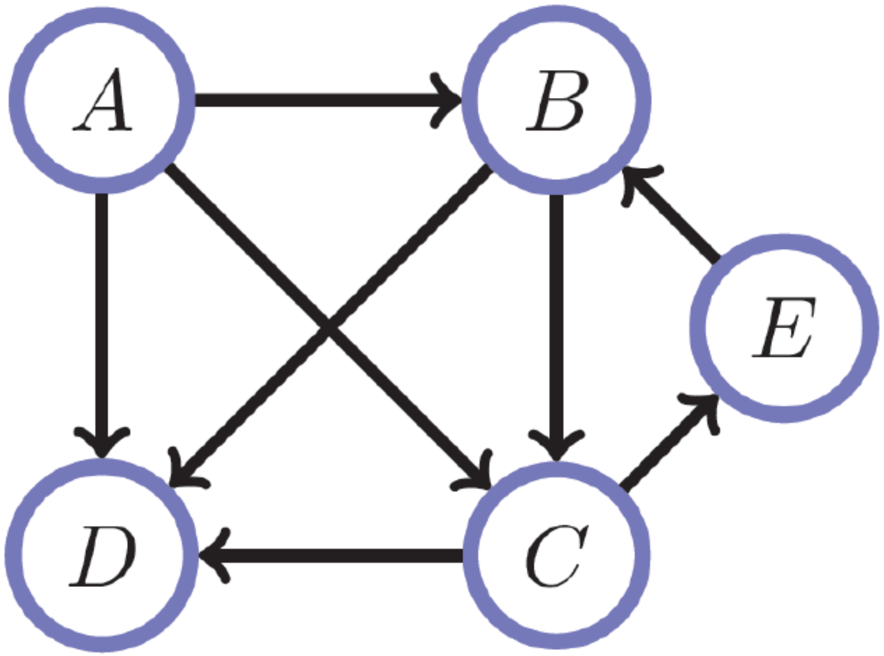
First, a recap of basics

Graphs

- Concepts
  - Definition of G
  - Vertices/Nodes
  - Edges
  - Directed vs Undirected
  - Neighbours vs Parent/Child
  - Degree vs In/Out degree
  - Walk vs Path vs Cycle

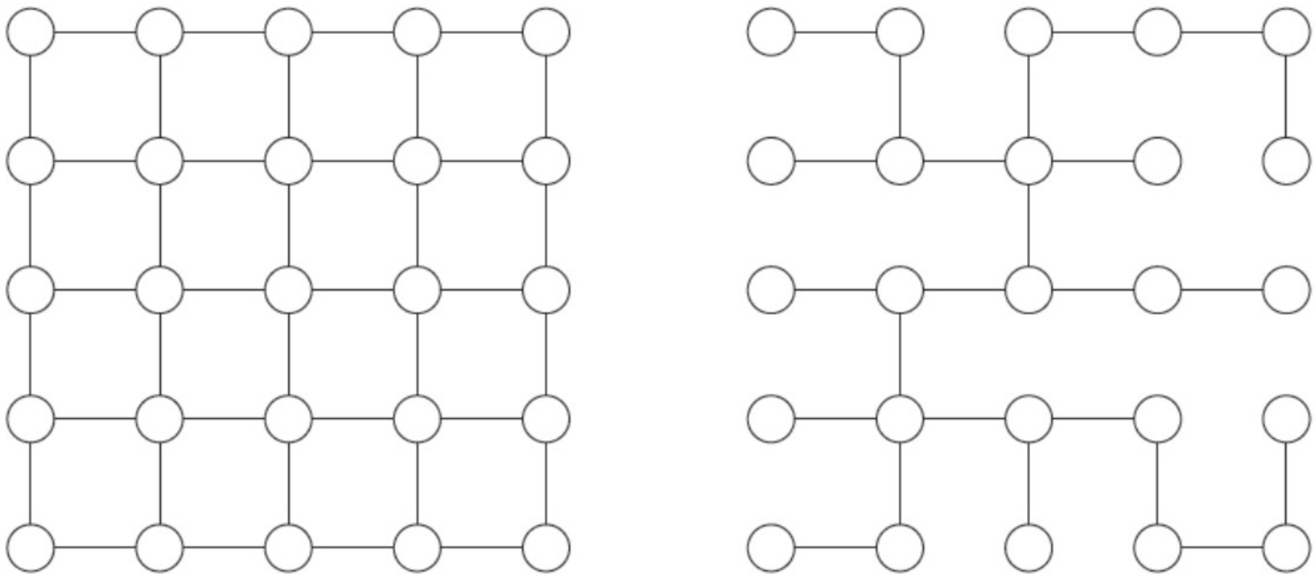
Graphs





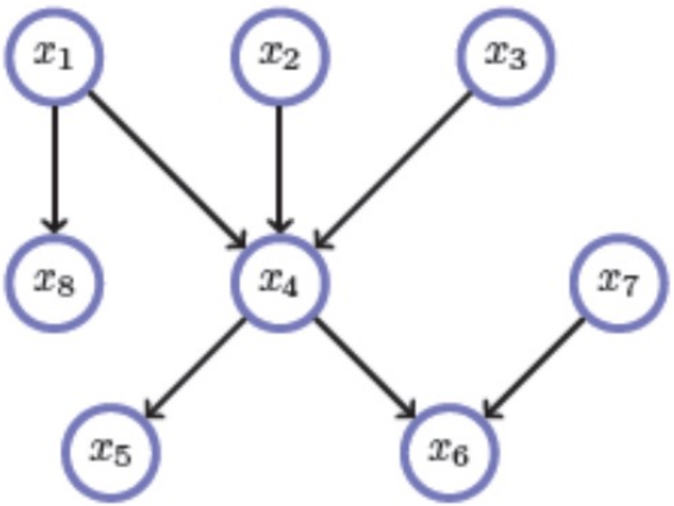
Special graphs

- Trees: undirected graph, no cycles
- Spanning tree: Same set of vertices, but subset of edges, connected and no cycles



Slide courtesy: D. Batra

Directed acyclic graphs (DAGs)



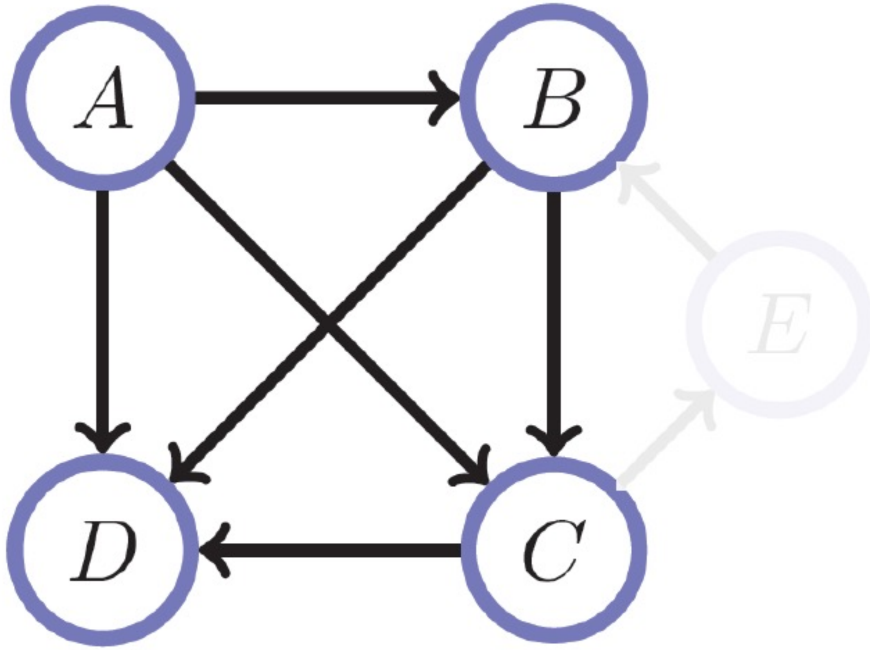


Figure courtesy: D. Batra

Joint distribution

- 3 variables
  - Intelligence (I)
  - Difficulty (D)
  - Grade (G)

I	D	G	Prob.
i^0	d^0	g^1	0.126
i^0	d^0	g^2	0.168
i^0	d^0	g^3	0.126
i^0	d^1	g^1	0.009
i^0	d^1	g^2	0.045
i^0	d^1	g^3	0.126
i^1	d^0	g^1	0.252
i^1	d^0	g^2	0.0224
i^1	d^0	g^3	0.0056
i^1	d^1	g^1	0.06
i^1	d^1	g^2	0.036
i^1	d^1	g^3	0.024

Example courtesy: PGM course, Daphne Koller

Conditioning

- Condition on g^1

I	D	G	Prob.
i^0	d^0	g^1	0.126
i^0	d^0	g^2	0.168
i^0	d^0	g^3	0.126
i^0	d^1	g^1	0.009
i^0	d^1	g^2	0.045
i^0	d^1	g^3	0.126
i^1	d^0	g^1	0.252
i^1	d^0	g^2	0.0224
i^1	d^0	g^3	0.0056
i^1	d^1	g^1	0.06
i^1	d^1	g^2	0.036
i^1	d^1	g^3	0.024

Example courtesy: PGM course, Daphne Koller

Conditioning

- $P(Y = y \mid X = x)$
- Informally,
  – What do you believe about Y=y when I tell you X=x ?

- P(France wins Euro 2024) ?
- What if I tell you:
  – France almost won the world cup 2022
  – Hasn't had catastrophic results since ☺

Slide based on D. Batra's course

Conditioning: Reduction

- Condition on g^1

I	D	G	Prob.
i^0	d^0	g^1	0.126
i^0	d^1	g^1	0.009
i^1	d^0	g^1	0.252
i^1	d^1	g^1	0.06

Example courtesy: PGM course, Daphne Koller

Conditioning: Renormalization

I	D	G	Prob.
i^0	d^0	g^1	0.126
i^0	d^1	g^1	0.009
i^1	d^0	g^1	0.252
i^1	d^1	g^1	0.06

$P(I, D, g^1)$

Unnormalized measure

I	D	Prob.
i^0	d^0	0.282
i^0	d^1	0.02
i^1	d^0	0.564
i^1	d^1	0.134

$P(I, D \mid g^1)$

Example courtesy: PGM course, Daphne Koller

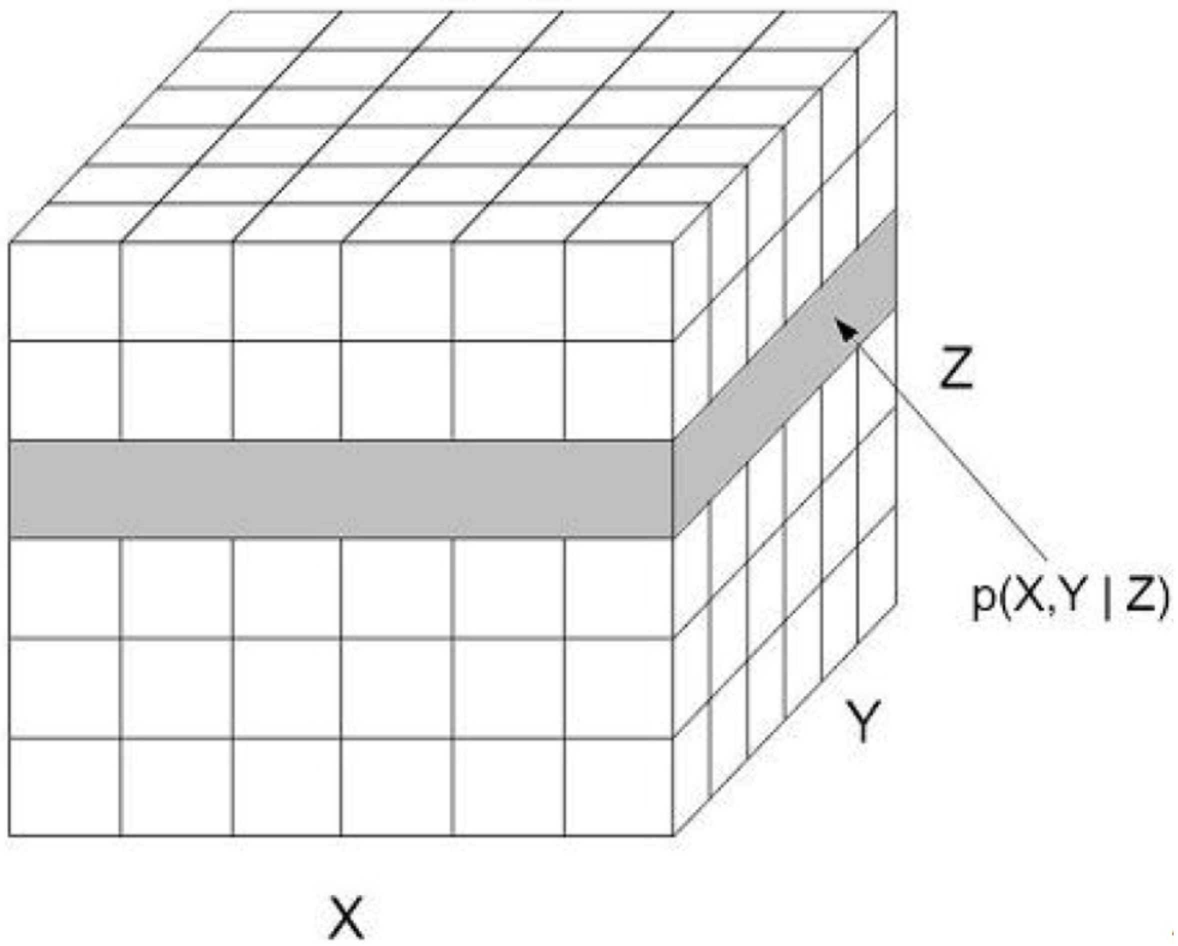
Conditional probability distribution

- Example $P(G \mid I, D)$

	g^1	g^2	g^3
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^1,d^0	0.9	0.08	0.02
i^1,d^1	0.5	0.3	0.2

Example courtesy: PGM course, Daphne Koller

Conditional probability distribution



$$p(x, y \mid Z = z) = \frac{p(x, y, z)}{p(z)}$$

Slide courtesy: Erik Sudderth

Marginalization

P(I,D)

I	D	Prob.
i^0	d^0	0.282
i^0	d^1	0.02
i^1	d^0	0.564
i^1	d^1	0.134

Marginalize I

D	Prob.
d^0	0.846
d^1	0.154

Example courtesy: PGM course, Daphne Koller

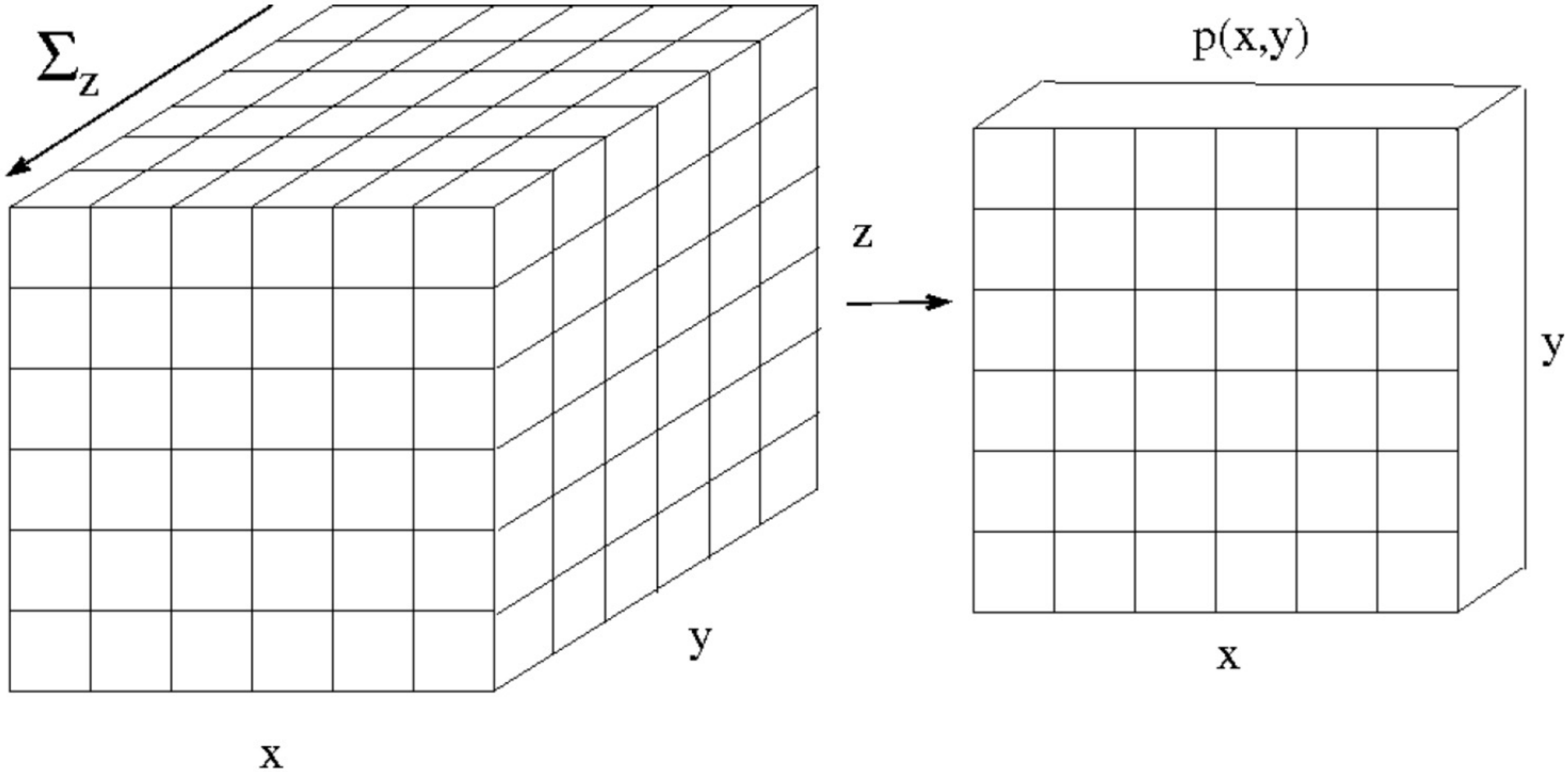
Marginalization

- Events
  - P(A) = P(A and B) + P(A and not B)

- Random variables
  - $P(X = x) = \sum_y P(X = x, Y = y)$

Slide courtesy: Dhruv Batra

Marginalization



$$p(x, y) = \sum_{z \in \mathcal{Z}} p(x, y, z) \qquad p(x) = \sum_{y \in \mathcal{Y}} p(x, y)$$

Slide courtesy: Erik Sudderth

Factors

- A factor $\Phi(Y_1,\ldots,Y_k)$

  Φ: Val$(Y_1,\ldots,Y_k) \rightarrow \mathrm{R}$

- Scope = $\{Y_1,\ldots,Y_k\}$

General factors

- Not necessarily for probabilities

A	B	ϕ
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

Example courtesy: PGM course, Daphne Koller

Factor product

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

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

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

Example courtesy: PGM course, Daphne Koller

Factor marginalization

a^1	b^1	c^1	0.25
a^1	b^1	c^2	0.35
a^1	b^2	c^1	0.08
a^1	b^2	c^2	0.16
a^2	b^1	c^1	0.05
a^2	b^1	c^2	0.07
a^2	b^2	c^1	0
a^2	b^2	c^2	0
a^3	b^1	c^1	0.15
a^3	b^1	c^2	0.21
a^3	b^2	c^1	0.09
a^3	b^2	c^2	0.18

a^1	c^1	0.33
a^1	c^2	0.51
a^2	c^1	0.05
a^2	c^2	0.07
a^3	c^1	0.24
a^3	c^2	0.39

Example courtesy: PGM course, Daphne Koller

Factor reduction

a^1	b^1	c^1	0.25
a^1	b^1	c^2	0.35
a^1	b^2	c^1	0.08
a^1	b^2	c^2	0.16
a^2	b^1	c^1	0.05
a^2	b^1	c^2	0.07
a^2	b^2	c^1	0
a^2	b^2	c^2	0
a^3	b^1	c^1	0.15
a^3	b^1	c^2	0.21
a^3	b^2	c^1	0.09
a^3	b^2	c^2	0.18

a^1	b^1	c^1	0.25
a^1	b^2	c^1	0.08
a^2	b^1	c^1	0.05
a^2	b^2	c^1	0
a^3	b^1	c^1	0.15
a^3	b^2	c^1	0.09

Why factors ?

- Building blocks for defining distributions in high-dimensional spaces

- Set of basic operations for manipulating these distributions

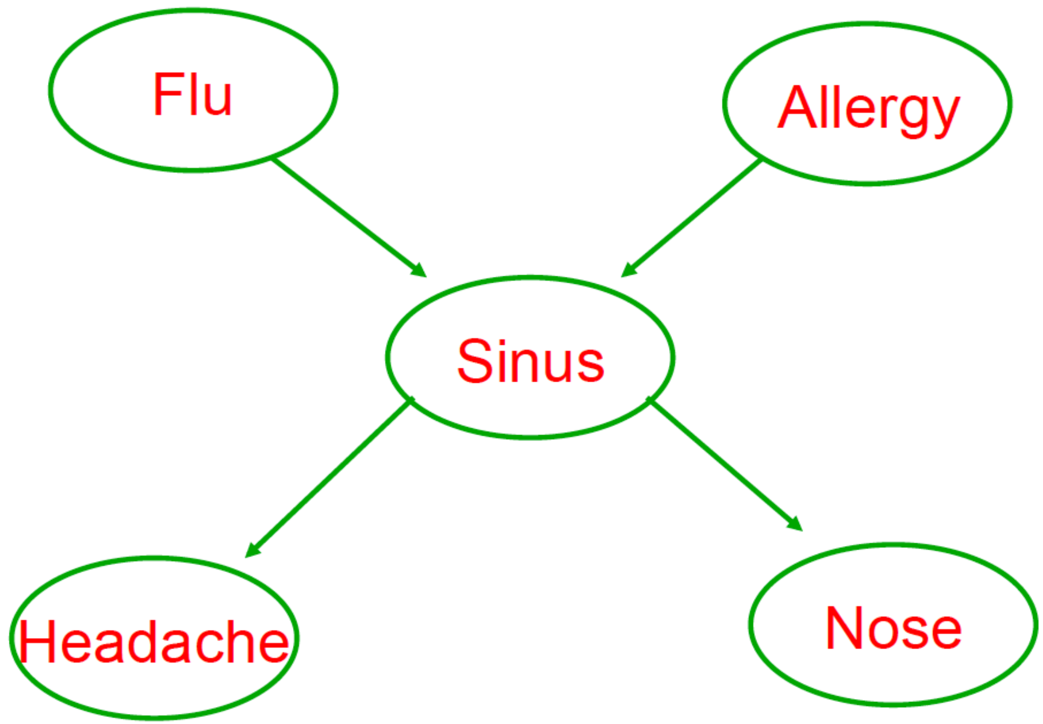
Bayesian Networks

- DAGs
  - nodes represent variables in the Bayesian sense
  - edges represent conditional dependencies
- Example
  - Suppose that we know the following:
    - The flu causes sinus inflammation
    - Allergies cause sinus inflammation
    - Sinus inflammation causes a runny nose
    - Sinus inflammation causes headaches
  - How are these connected ?

Slide courtesy: Dhruv Batra

Bayesian Networks

- Example



Slide courtesy: Dhruv Batra

Bayesian Networks

- A general Bayes net
  - Set of random variables
  - DAG: encodes independence assumptions
  - Conditional probability trees
  - Joint distribution

$$P(Y_1,\ldots,Y_n) = \prod_{i=1}^{n} P(Y_i \mid \mathrm{Pa}_{Y_i})$$

Slide courtesy: Dhruv Batra

Bayesian Networks

- A general Bayes net
  - How many parameters ?
    - Discrete variables $Y_1,\ldots,Y_n$
    - Graph: Defines parents of Y_i, i.e., (Pa_{Y_i})
    - CPTs: $P(Y_i|Pa_{Y_i})$

Slide courtesy: Dhruv Batra

Markov nets

- Set of random variables
- Undirected graph
  - Encodes independence assumptions
- Factors

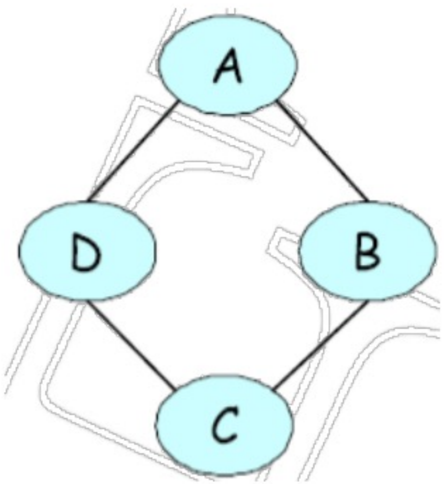
Comparison to Bayesian Nets ?

Slide courtesy: Dhruv Batra

Pairwise MRFs

- Composed of pairwise factors
  - A function of two variables
  - Can also have unary terms

- Example



$\phi_1[A,B]$			$\phi_2[B,C]$			$\phi_3[C,D]$			$\phi_4[D,A]$		
a^0	b^0	30	b^0	c^0	100	c^0	d^0	1	d^0	a^0	100
a^0	b^1	5	b^0	c^1	1	c^0	d^1	100	d^0	a^1	1
a^1	b^0	1	b^1	c^0	1	c^1	d^0	100	d^1	a^0	1
a^1	b^1	10	b^1	c^1	100	c^1	d^1	1	d^1	a^1	100

Slide courtesy: Dhruv Batra

Markov Nets: Computing probabilities

- Can only compute ratio of probabilities directly

$\phi_1[A,B]$			$\phi_2[B,C]$			$\phi_3[C,D]$			$\phi_4[D,A]$		
a^0	b^0	30	b^0	c^0	100	c^0	d^0	1	d^0	a^0	100
a^0	b^1	5	b^0	c^1	1	c^0	d^1	100	d^0	a^1	1
a^1	b^0	1	b^1	c^0	1	c^1	d^0	100	d^1	a^0	1
a^1	b^1	10	b^1	c^1	100	c^1	d^1	1	d^1	a^1	100

- Need to normalize with a **partition function**
  - Hard ! (sum over all possible assignments)
- In Bayesian Nets, can do by multiplying CPTs

Slide courtesy: Dhruv Batra

Markov nets ←→ Factorization

- Given an undirected graph H over variables $Y=\{Y_1,\ldots,Y_n\}$
- A distribution P factorizes over H if there exist
  - Subsets of variables $S^i \subseteq Y$ s.t. S^i are fully-connected in H
  - Non-negative potentials (factors) $\Phi_1(S^1),\ldots, \Phi_m(S^m)$: clique potentials
  - Such that

$$P(Y_1,\ldots,Y_n) = \frac{1}{Z}\prod_{i=1}^{m}\Phi_i(S^i)$$

Slide courtesy: Dhruv Batra

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 !

MRFs / CRFs

- 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

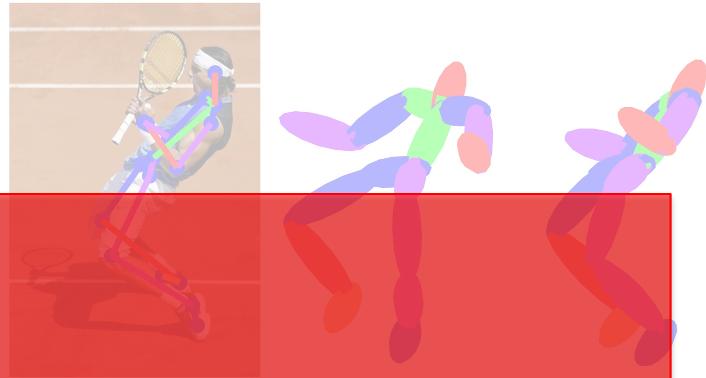
Stereo matching [Kolmogorov and Zabih, 2001; Scharstein and Szeliski, 2002]

Image denoising [Felzenszwalb and Huttenlocher 2004]

MRFs / CRFs

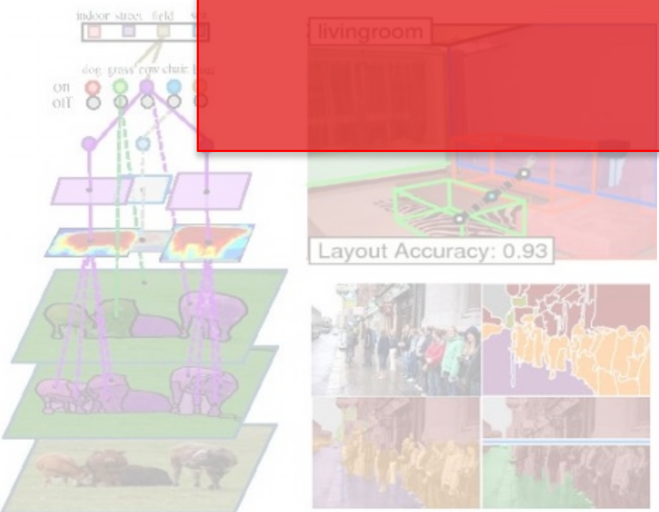
- Several applications, e.g., computer vision

Object detection [Felzenszwalb et al., 2008]



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

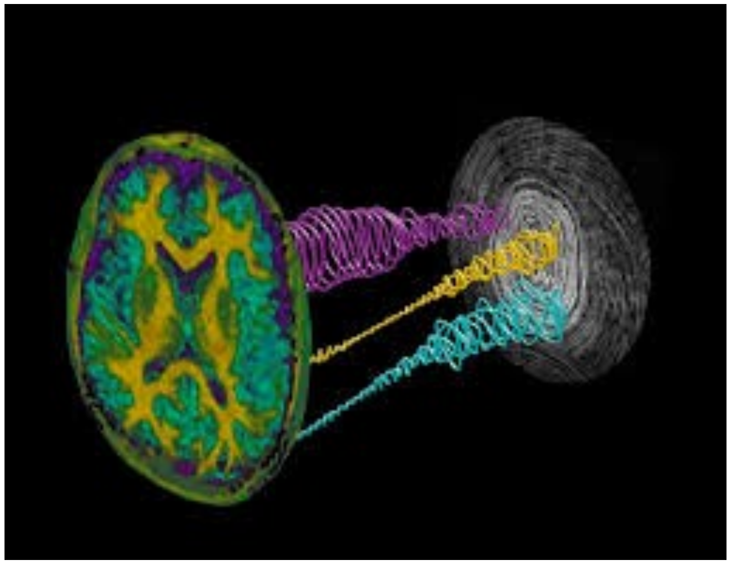
High-level vision problems

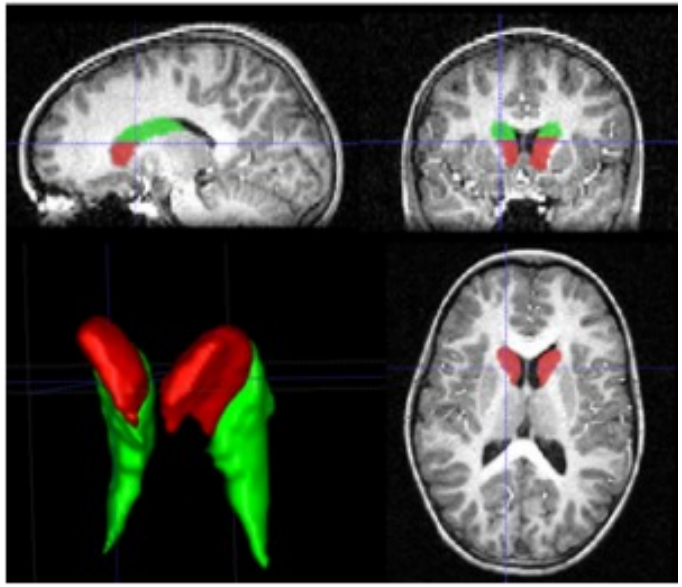


Scene understanding
[Fouhey et al., 2014; Ladicky et al., 2010;
Xiao et al., 2013; Yao et al., 2012]

MRFs / CRFs

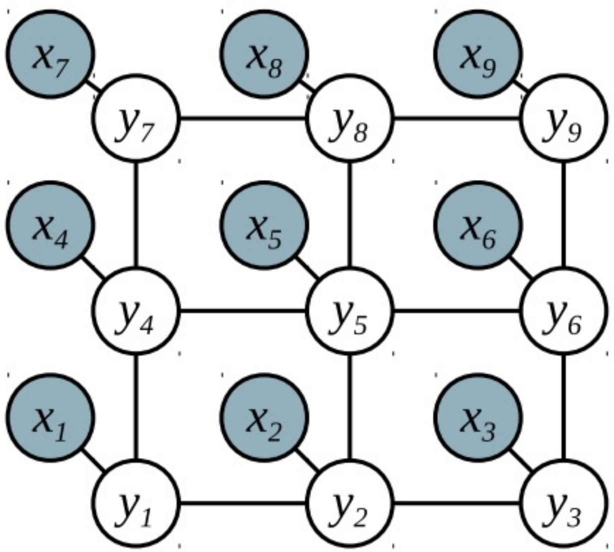
- Several applications, e.g., medical imaging



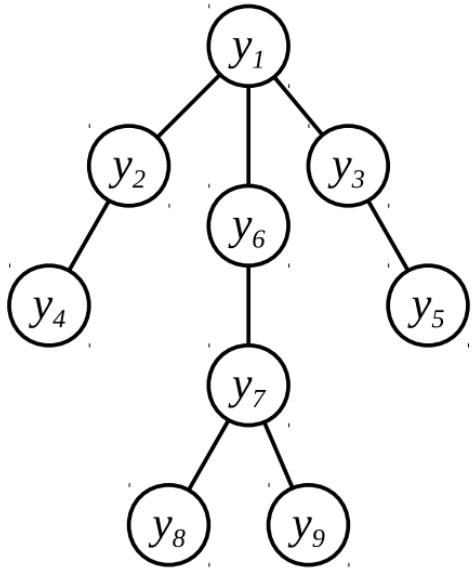


MRFs / CRFs

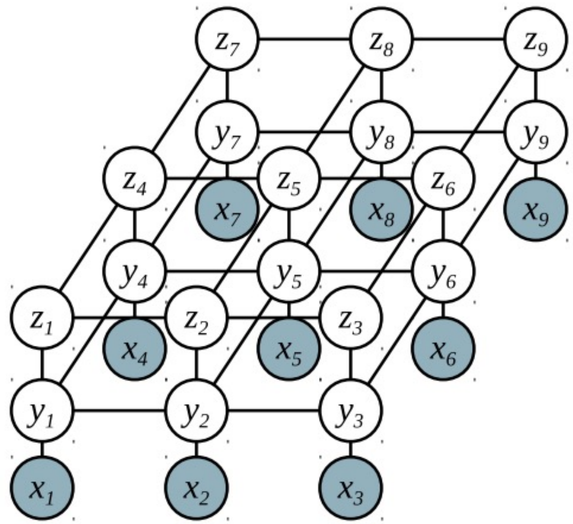
- Inherent in all these problems are graphical models



Pixel labeling



Object detection
Pose estimation



Scene understanding

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

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

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

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

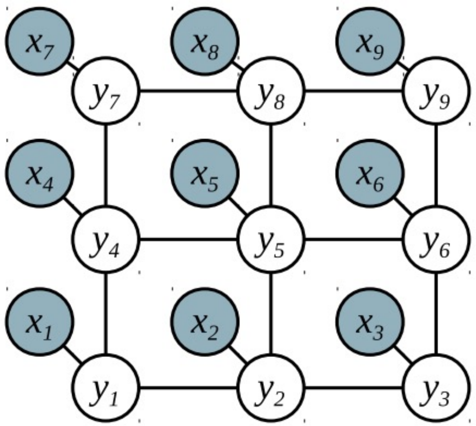
Parameters

Clique potentials

- Arity

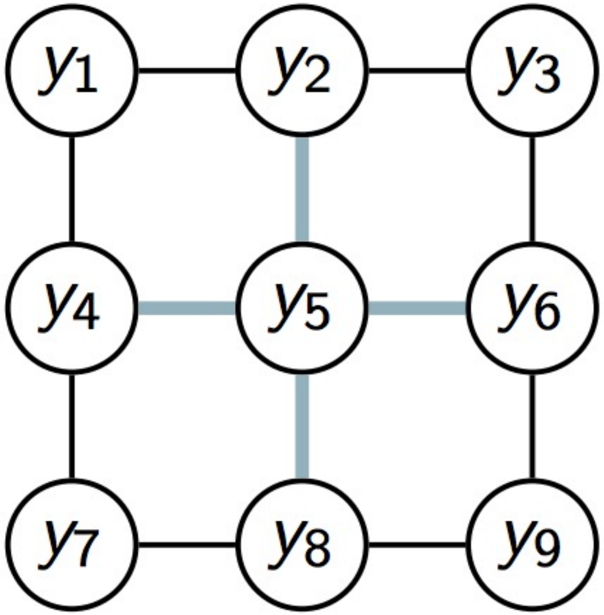
$$E(\mathbf{y};\mathbf{x}) = \sum_{c} \psi_c(\mathbf{y}_c;\mathbf{x})$$

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

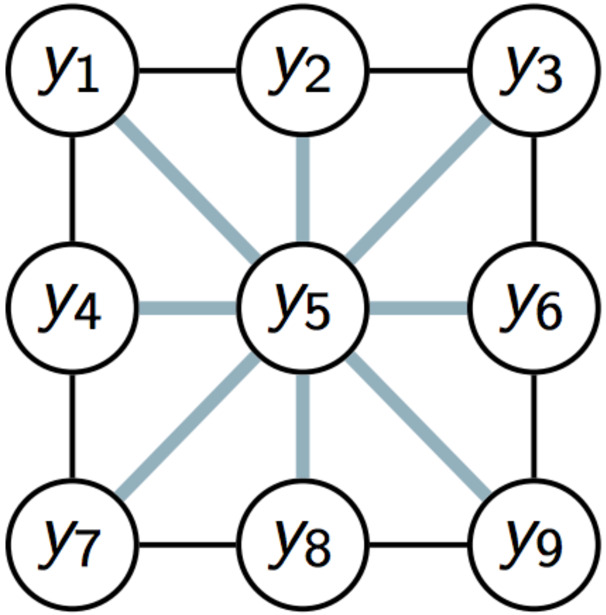


Clique potentials

- Arity

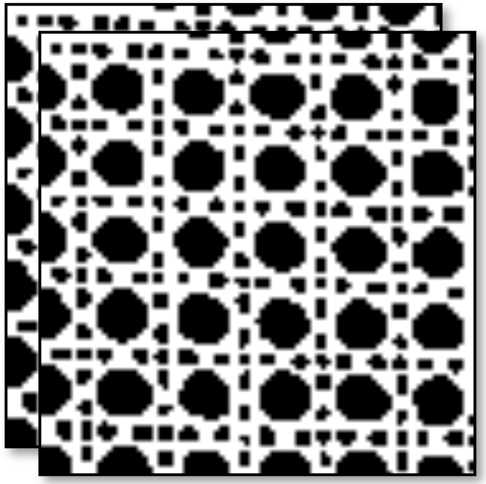


4-connected, $\mathcal{N}_4$

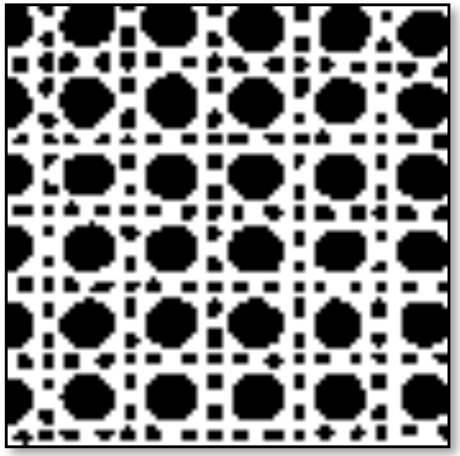


8-connected, $\mathcal{N}_8$

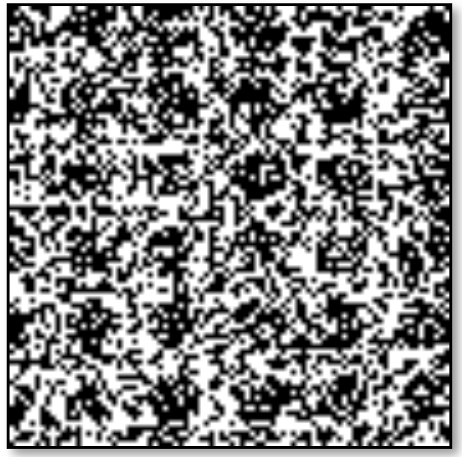
Reason 1: Texture modelling



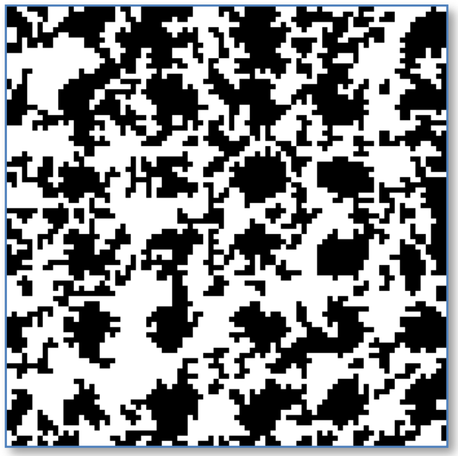
Training images



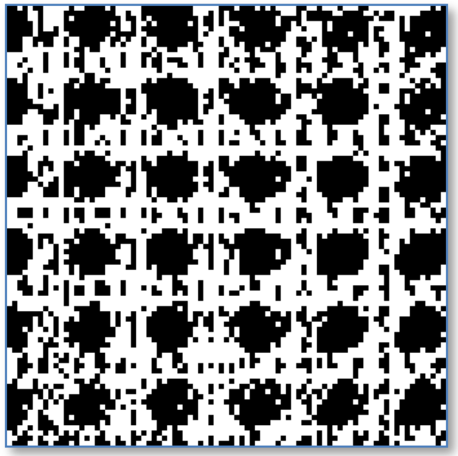
Test image



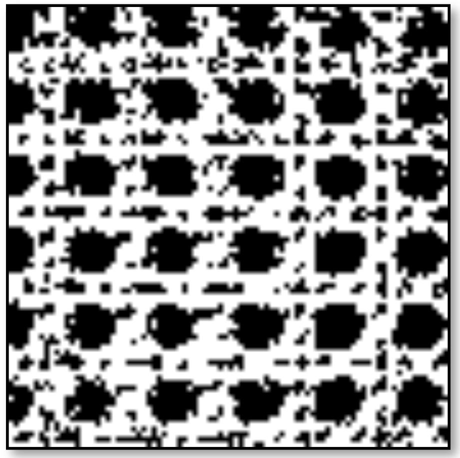
Test image (60% Noise)



Result MRF
4-connected
(neighbours)

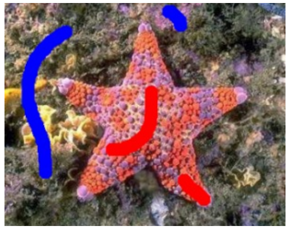


Result MRF
4-connected



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

Reason2: Discretization artefacts



4-connected
Euclidean

8-connected
Euclidean

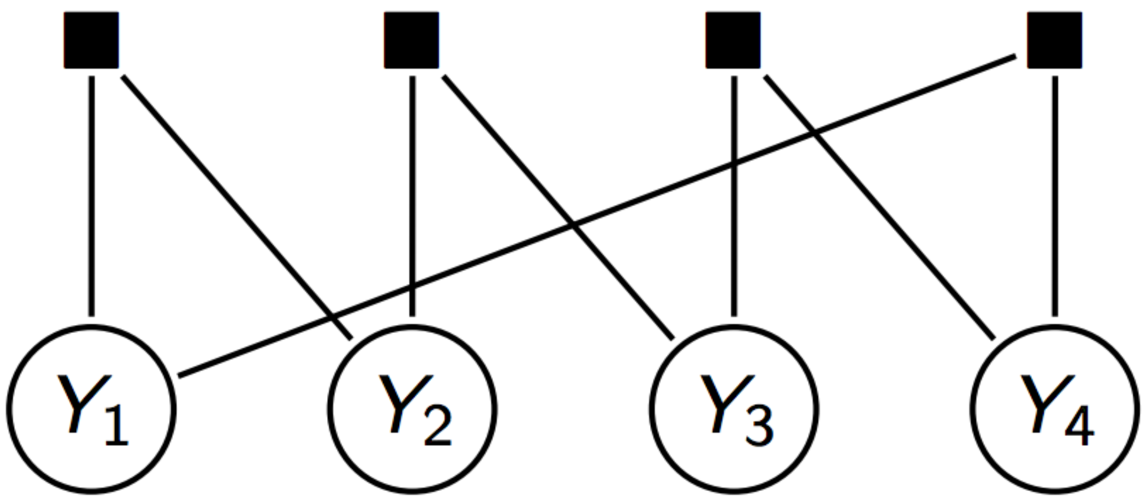
higher-connectivity can model
true Euclidean length

[Boykov et al. ’03; ’05]

Graphical representation

- Example

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

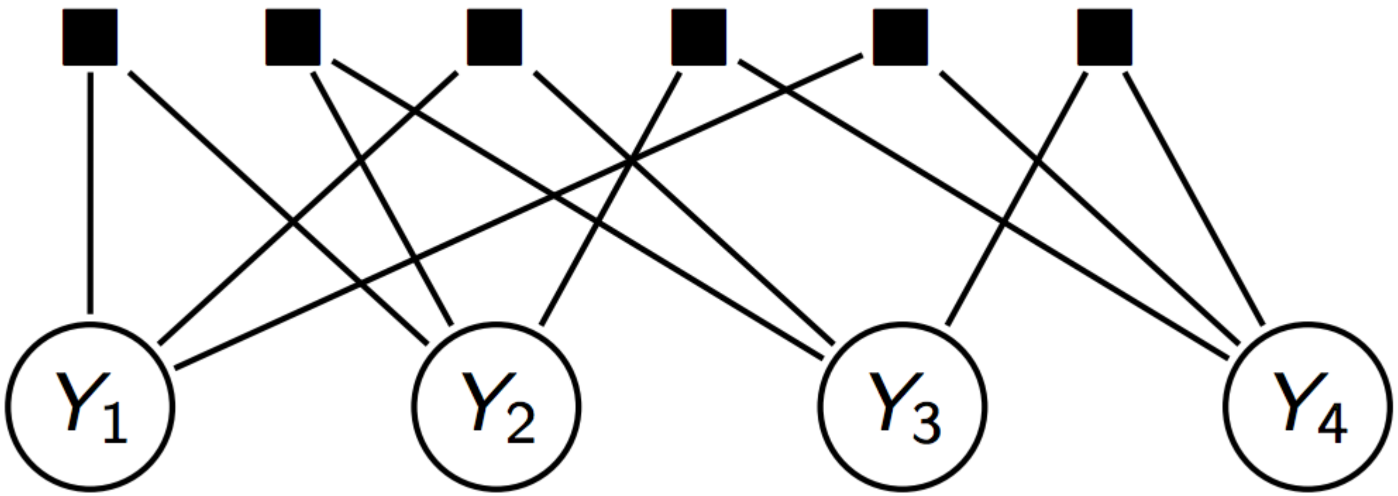


factor graph

Graphical representation

- Example

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

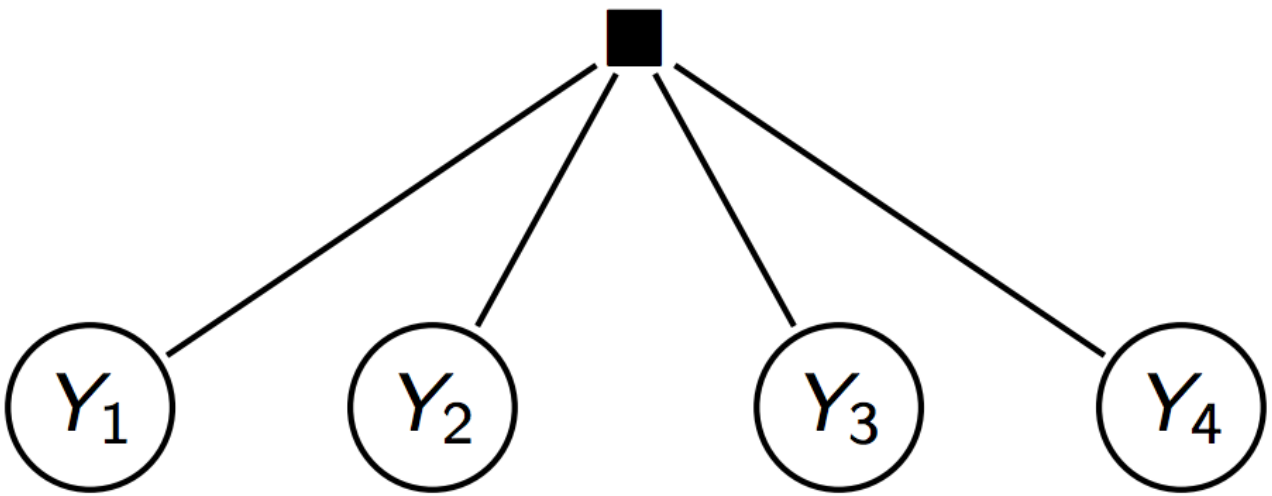


factor graph

Graphical representation

- Example

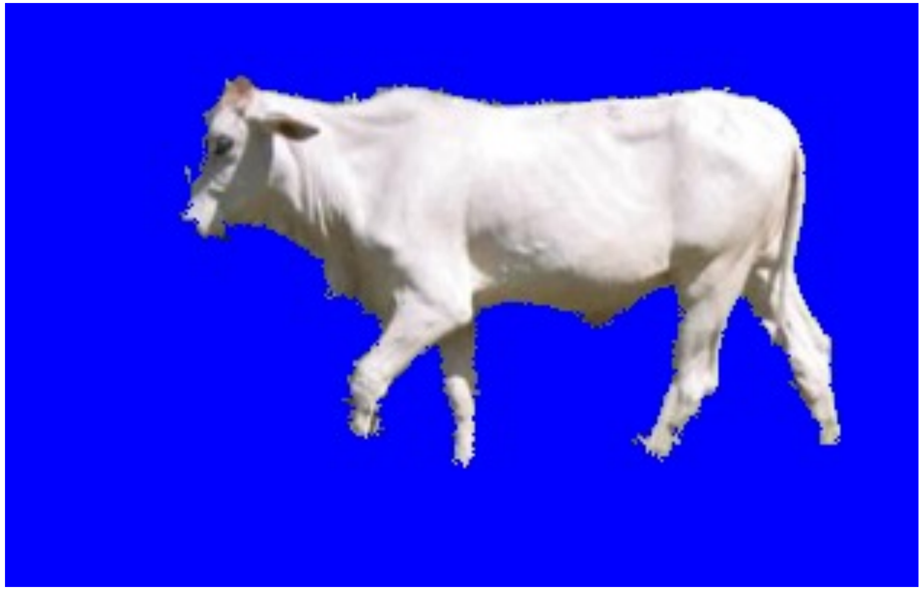
$$E(\mathbf{y}) = \psi(y_1, y_2, y_3, y_4)$$



factor graph

A Computer Vision Application

Binary Image Segmentation



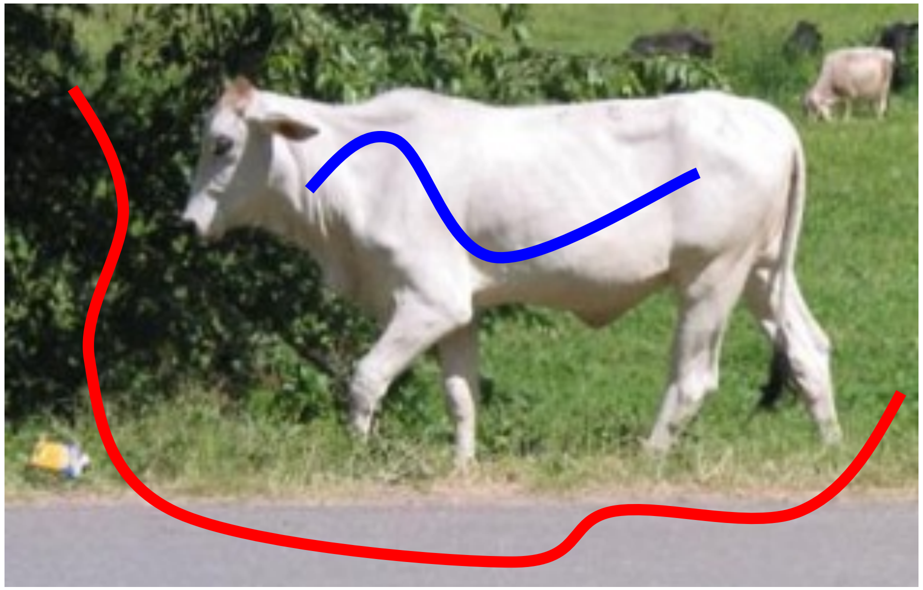
How ?

Cost function   Models *our* knowledge about natural images

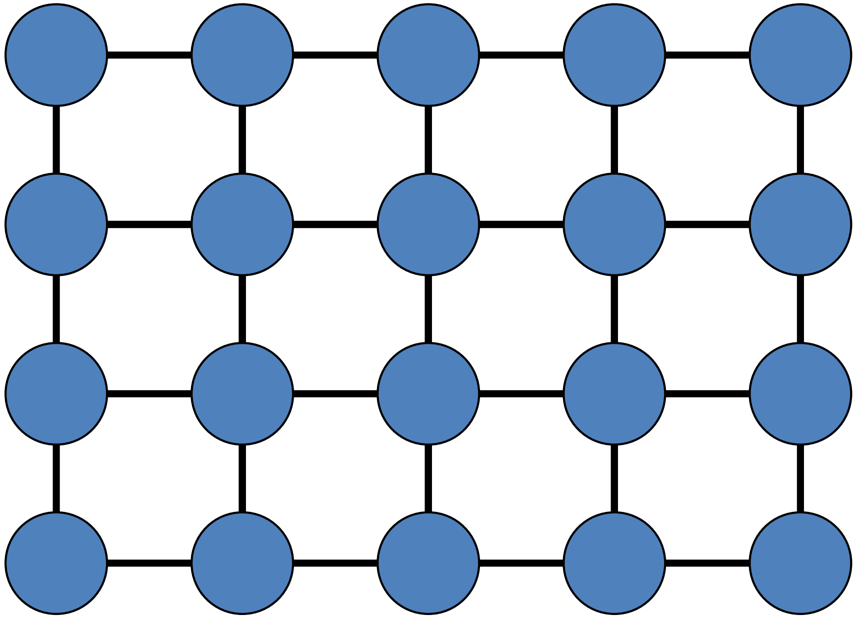
Optimize cost function to obtain the segmentation

A Computer Vision Application

Binary Image Segmentation



Object - white, Background - green/grey



Graph G = (V,E)

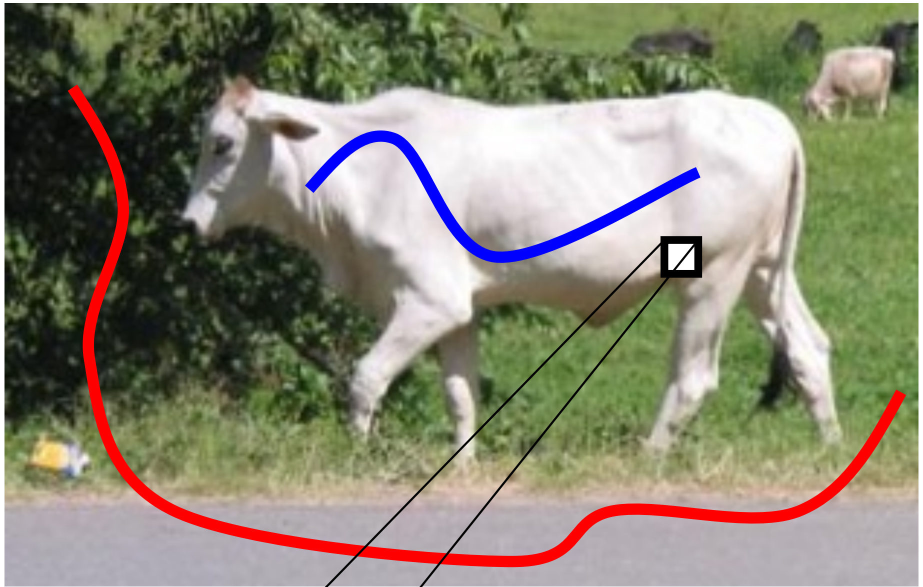
Each vertex corresponds to a pixel

Edges define a 4-neighbourhood *grid* graph

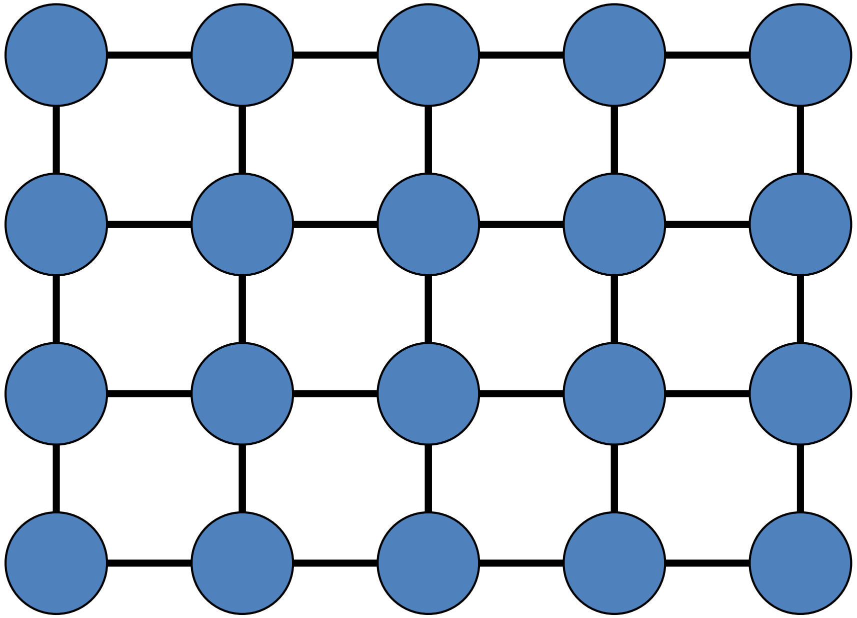
Assign a label to each vertex from L = {obj,bkg}

A Computer Vision Application

Binary Image Segmentation



Object - white, Background - green/grey



Graph G = (V,E)

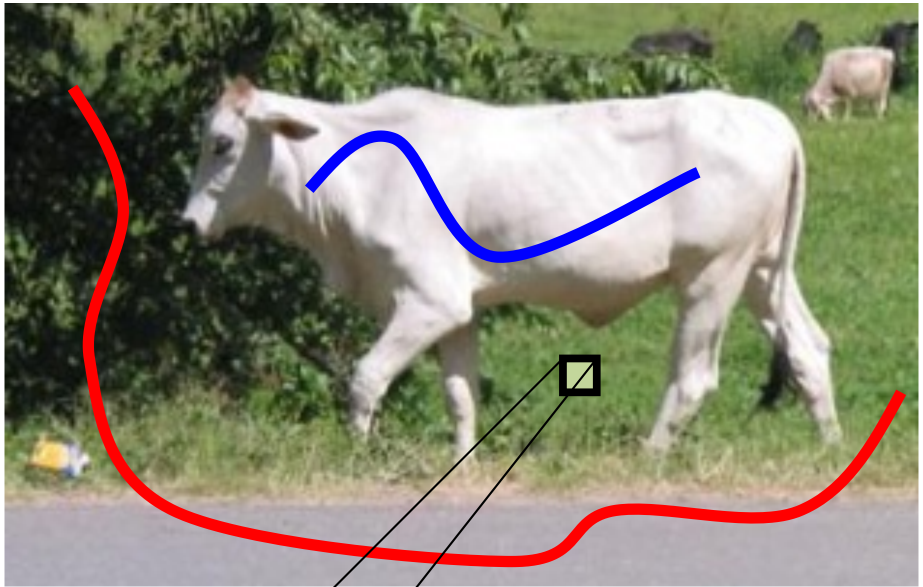
Cost of a labelling f : V ➔ L

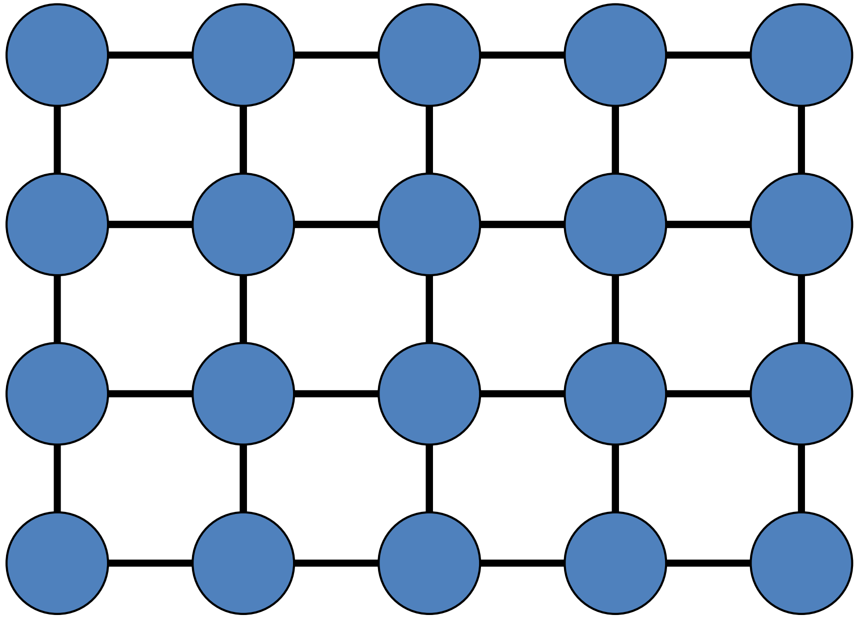
Per Vertex Cost

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

A Computer Vision Application

Binary Image Segmentation



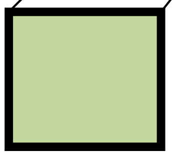


Object - white, Background - green/grey

Graph G = (V,E)

Cost of a labelling f : V ➔ L

Per Vertex Cost

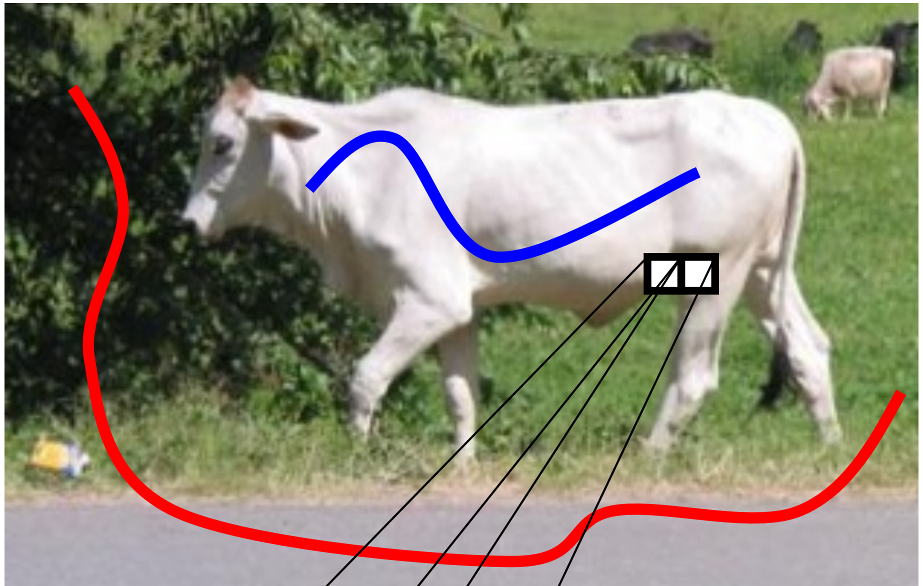


Cost of label 'obj' high Cost of label 'bkg' low

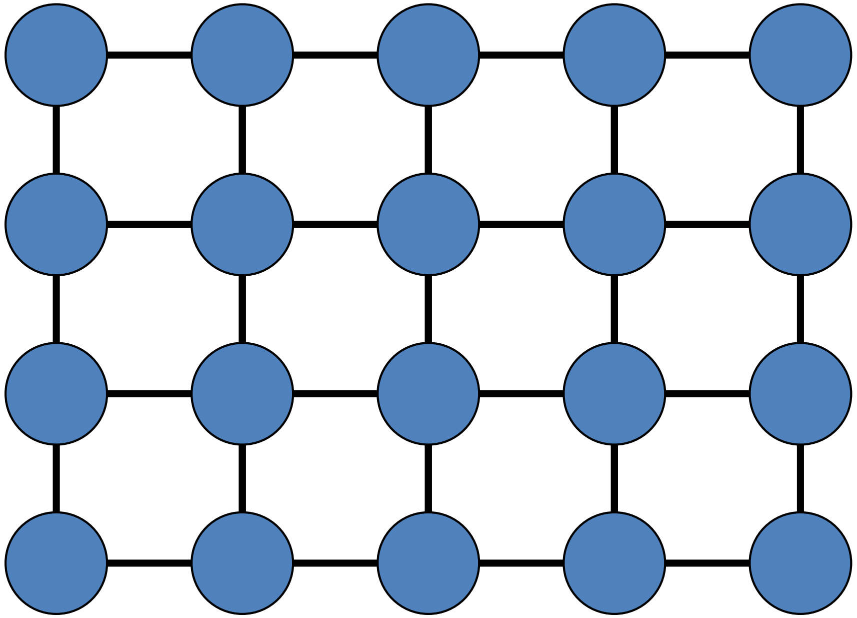
UNARY COST

A Computer Vision Application

Binary Image Segmentation



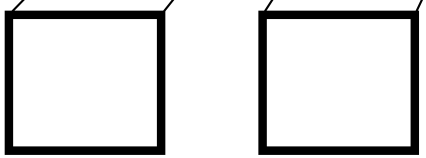
Object - white, Background - green/grey



Graph G = (V,E)

Cost of a labelling f : V ➔ L

Per Edge Cost

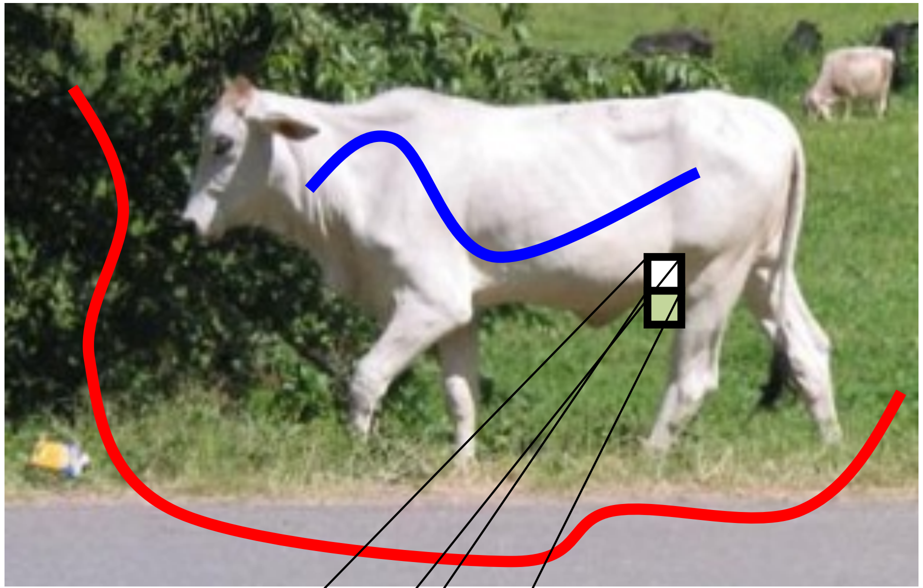


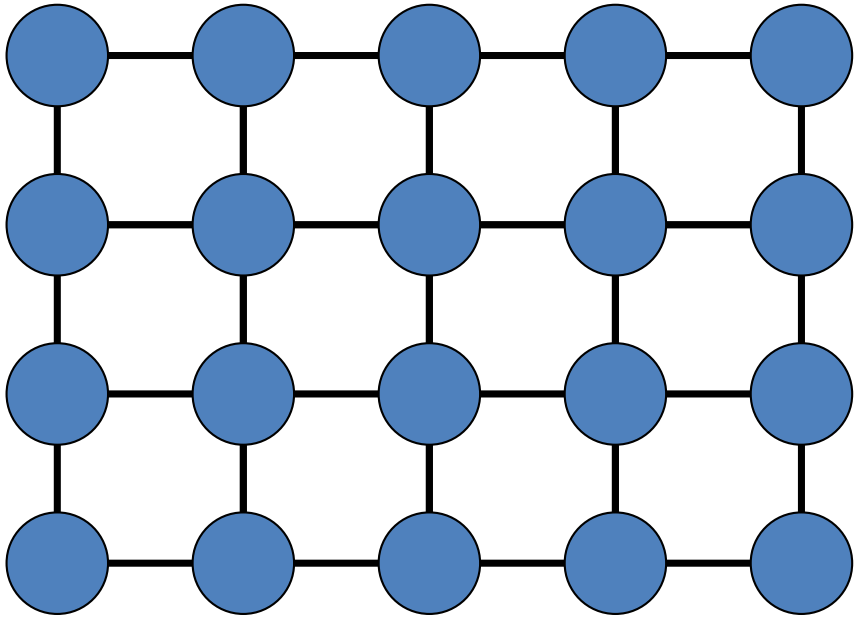
Cost of same label low

Cost of different labels high

A Computer Vision Application

Binary Image Segmentation



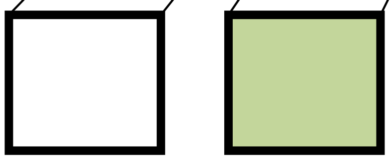


Object - white, Background - green/grey

Graph G = (V,E)

Cost of a labelling f : V → L

Per Edge Cost



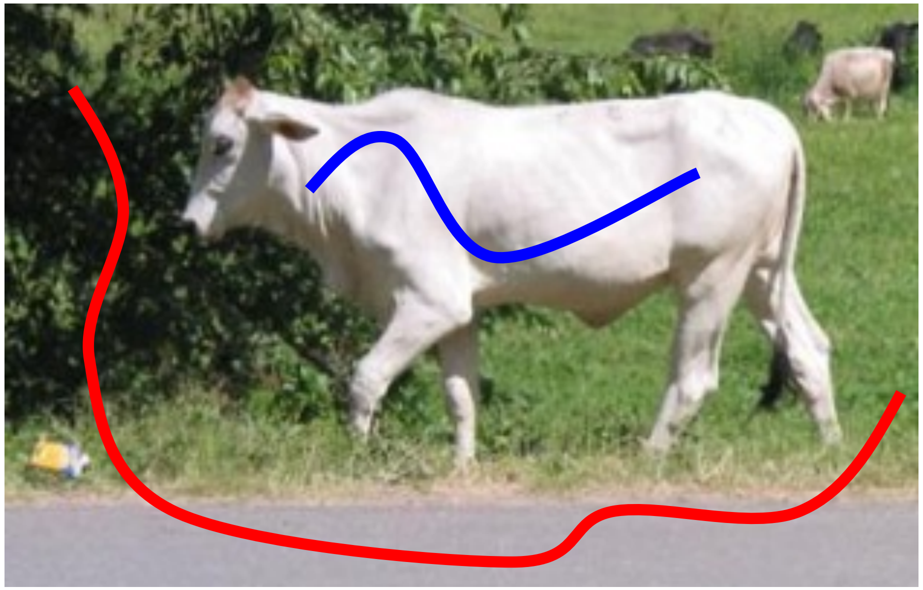
Cost of same label high

Cost of different labels low

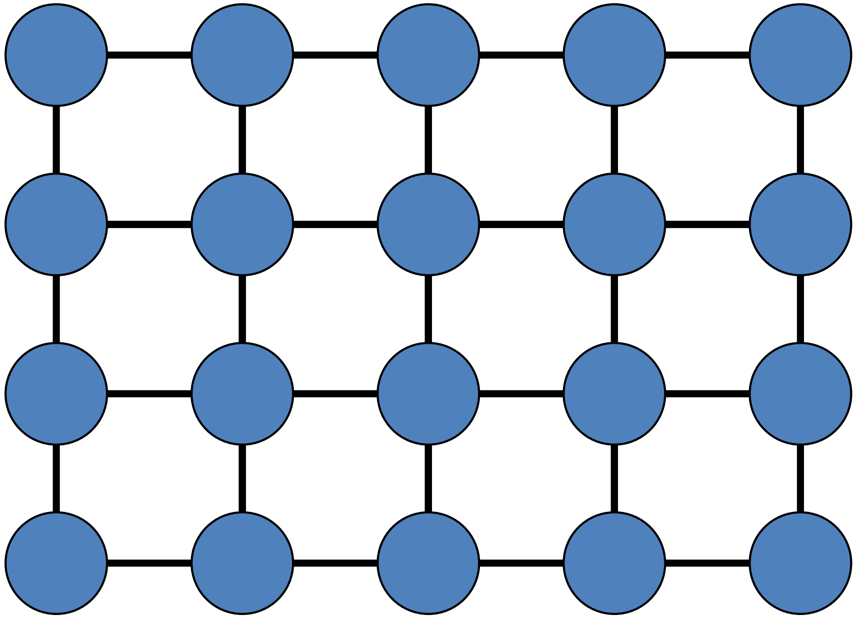
PAIRWISE COST

A Computer Vision Application

Binary Image Segmentation



Object - white, Background - green/grey

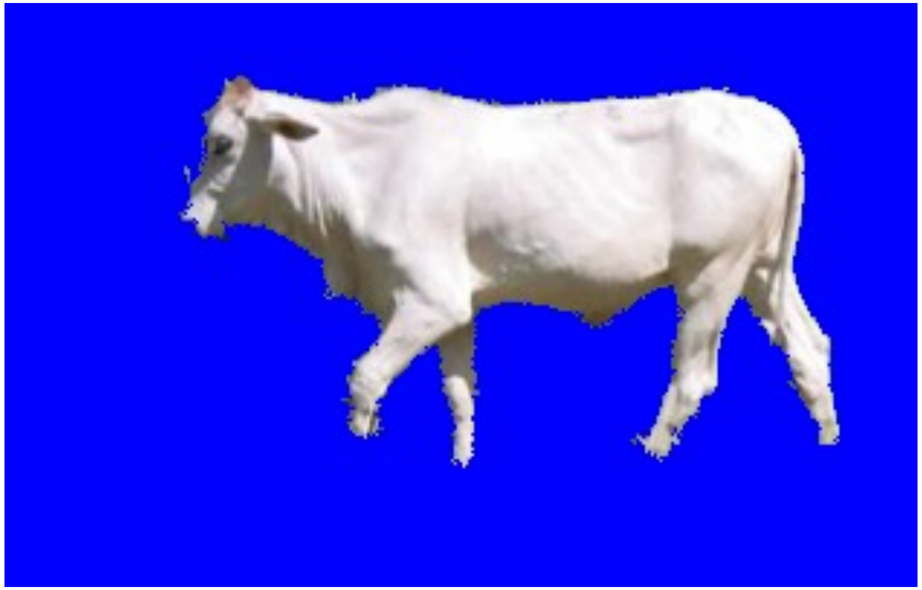


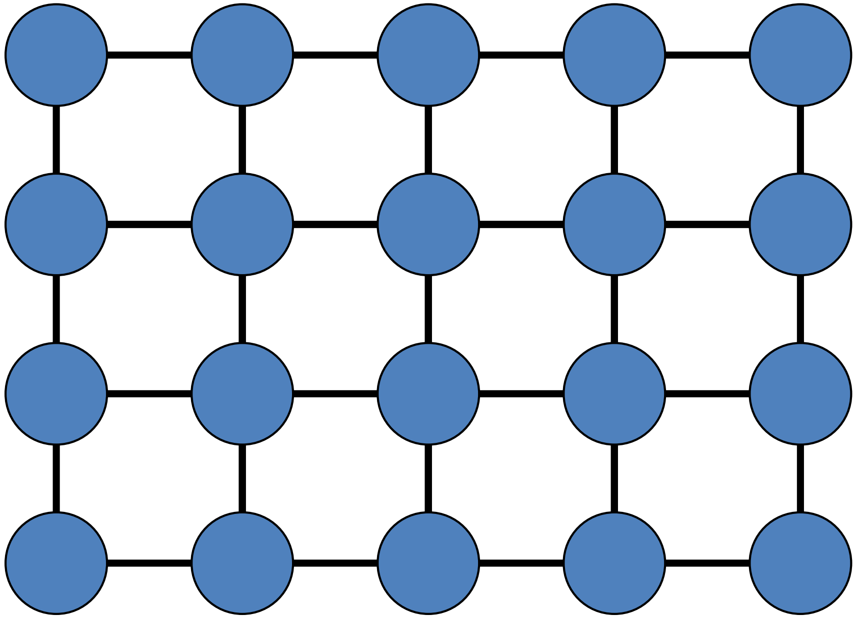
Graph G = (V,E)

Problem: Find the labelling with minimum cost f*

A Computer Vision Application

Binary Image Segmentation



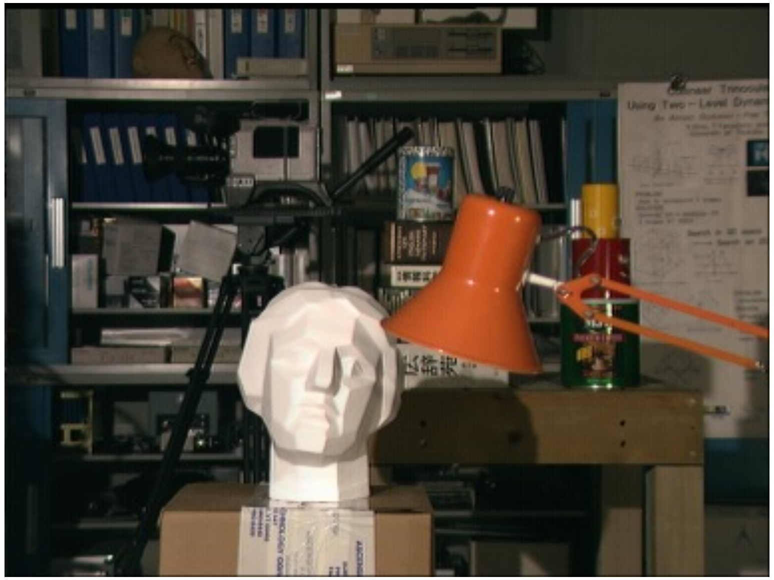


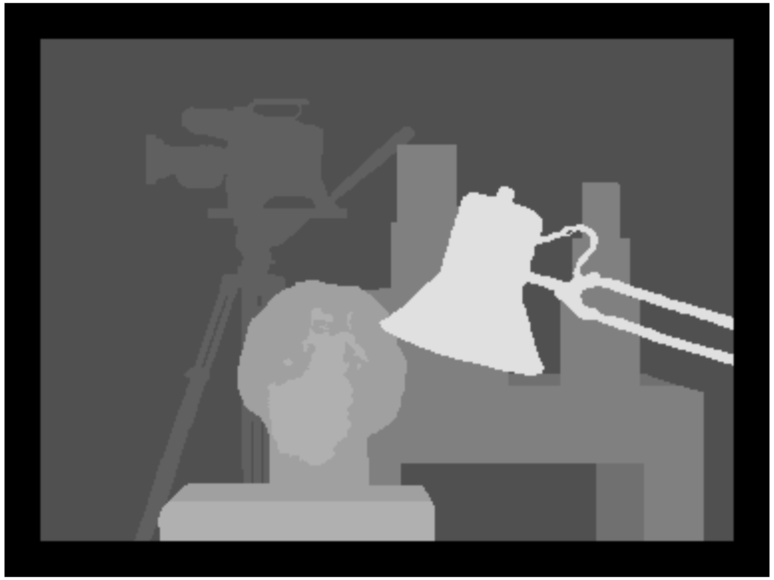
Graph G = (V,E)

Problem: Find the labelling with minimum cost f*

Another Computer Vision Application

Stereo Correspondence





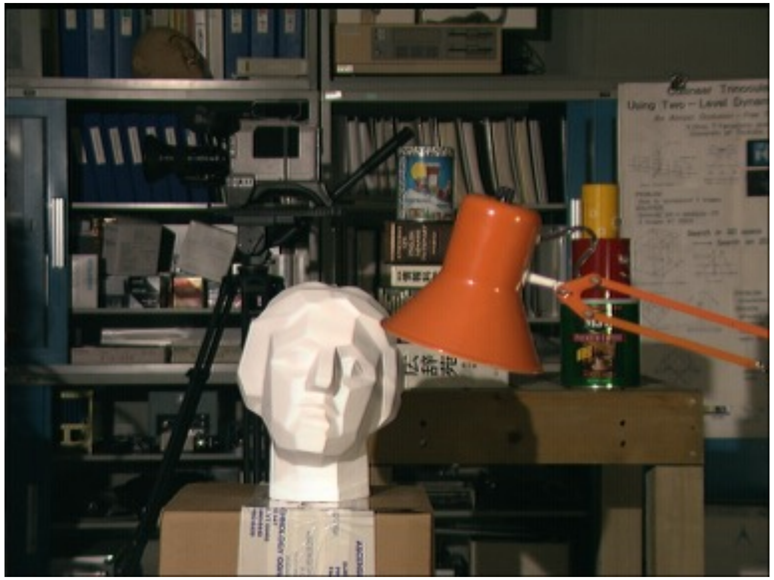
Disparity Map

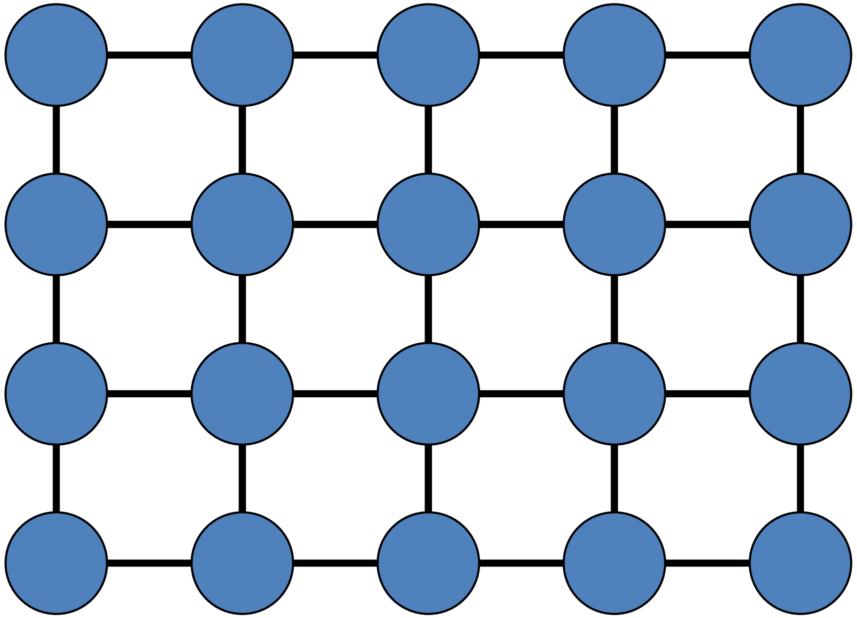
How ?

Minimizing a cost function

Another Computer Vision Application

Stereo Correspondence





Graph G = (V,E)

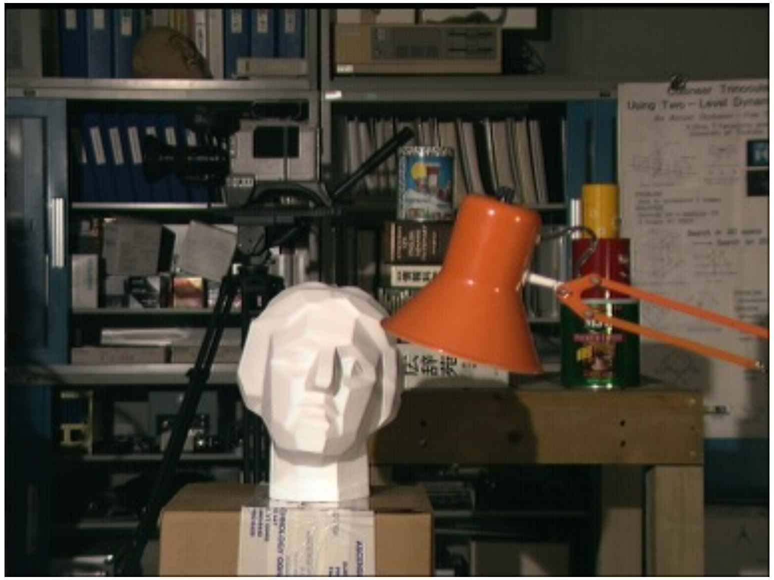
Vertex corresponds to a pixel

Edges define grid graph

L = {disparities}

Another Computer Vision Application

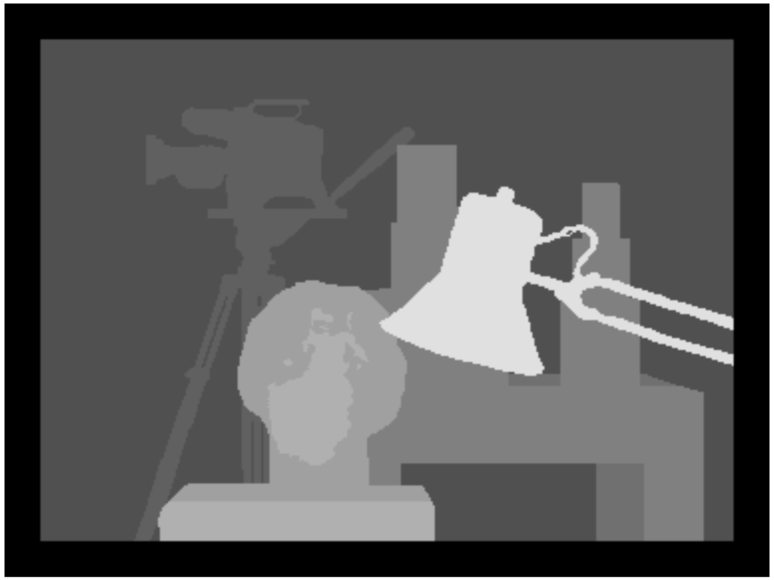
Stereo Correspondence



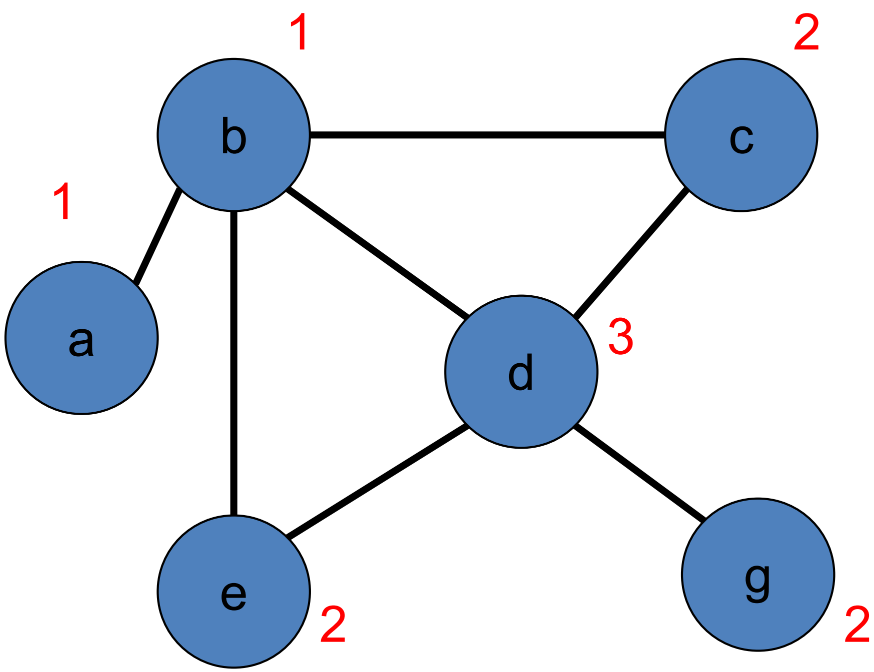
Cost of labelling f :

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

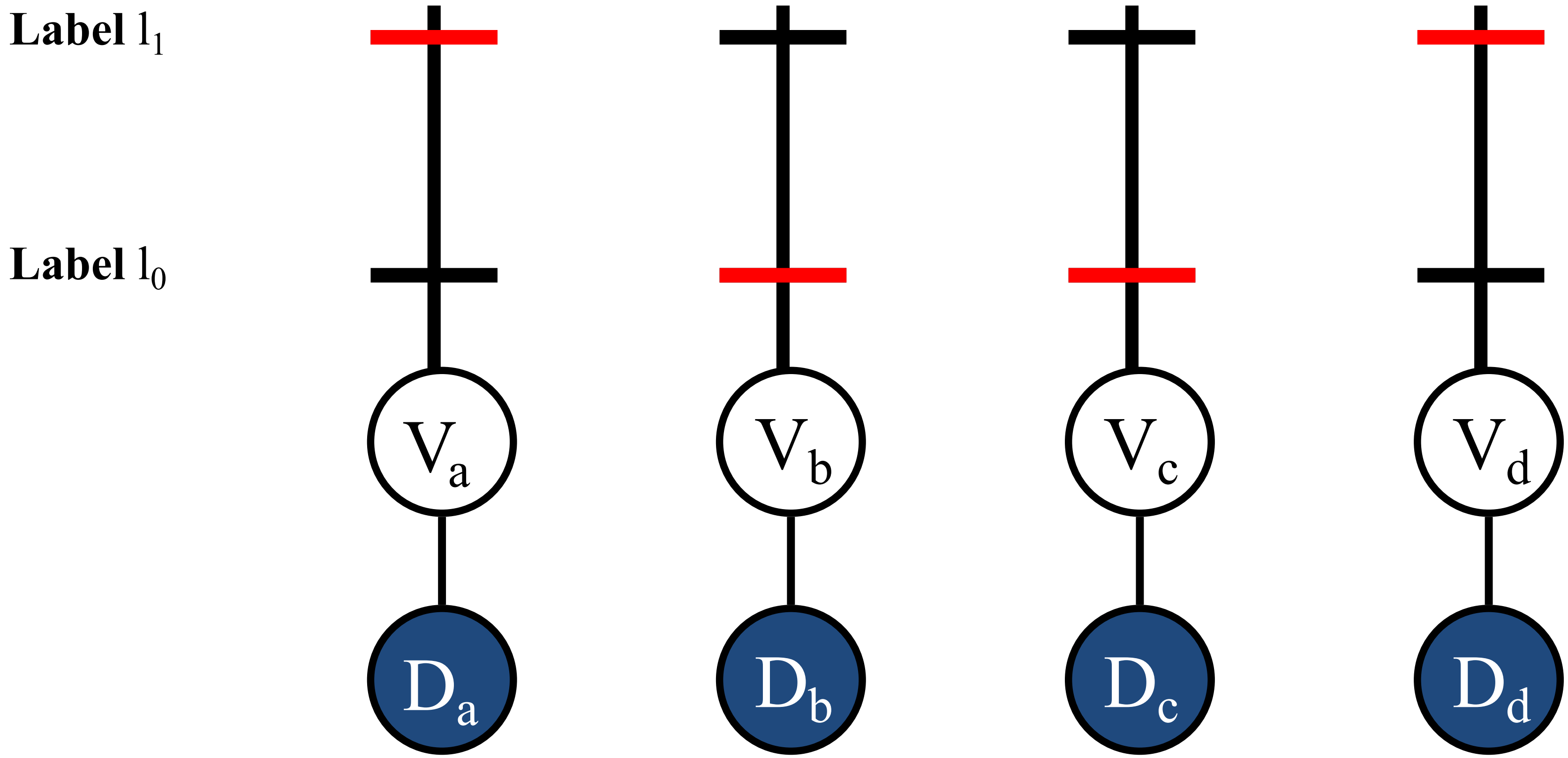
Find f* = arg min Q(f)

Overview

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

- Solutions
  - Belief Propagation and related methods
  - Graph cuts

Energy Function

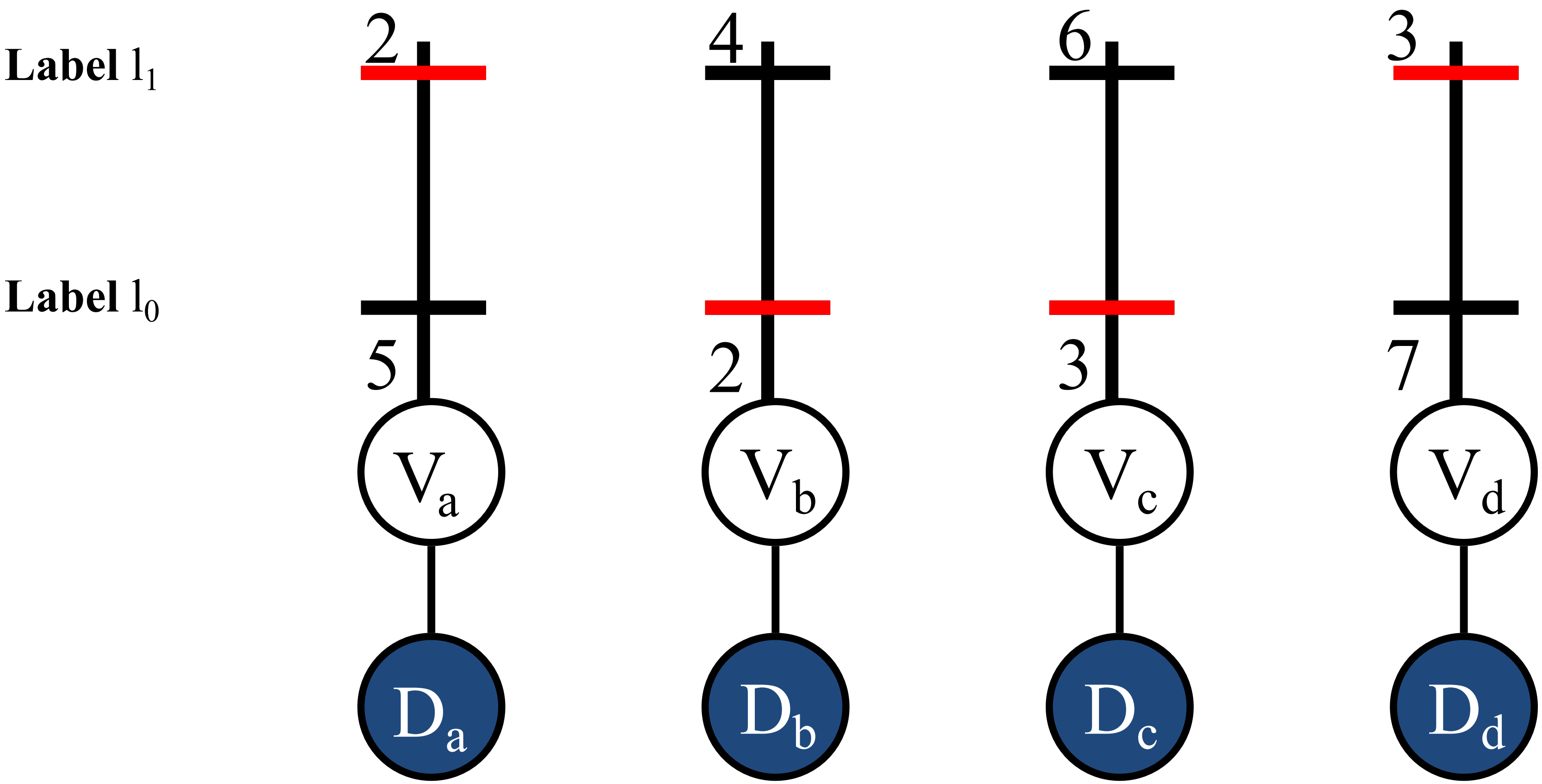


Random Variables $V = \{V_a, V_b, \ldots.\}$

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

Labelling f: $\{a, b, \ldots. \} \rightarrow \{0,1, \ldots\}$

Energy Function



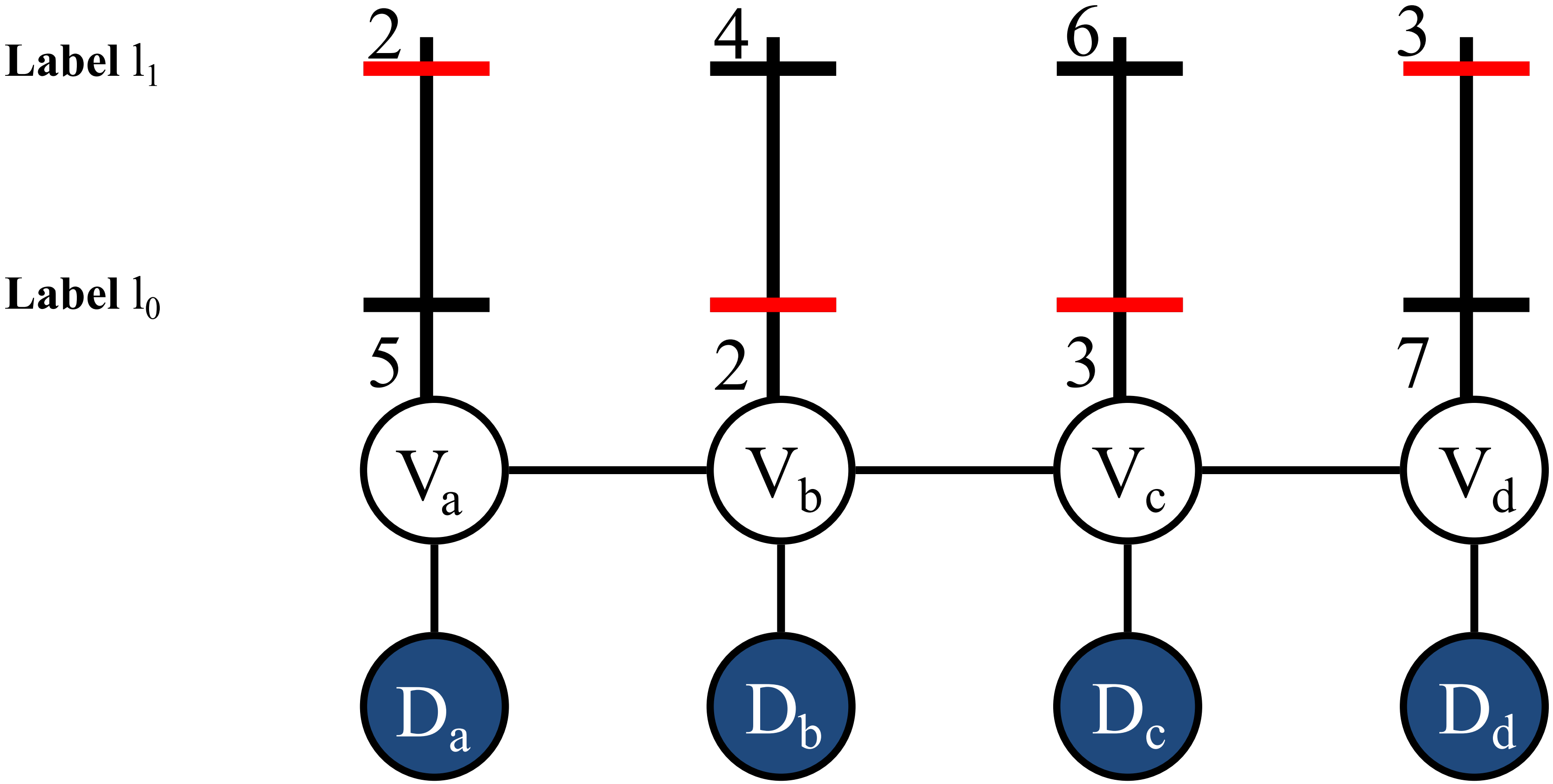
$$Q(f) = \sum_a \theta_{a;f(a)}$$

Unary Potential

Easy to minimize

Neighbourhood

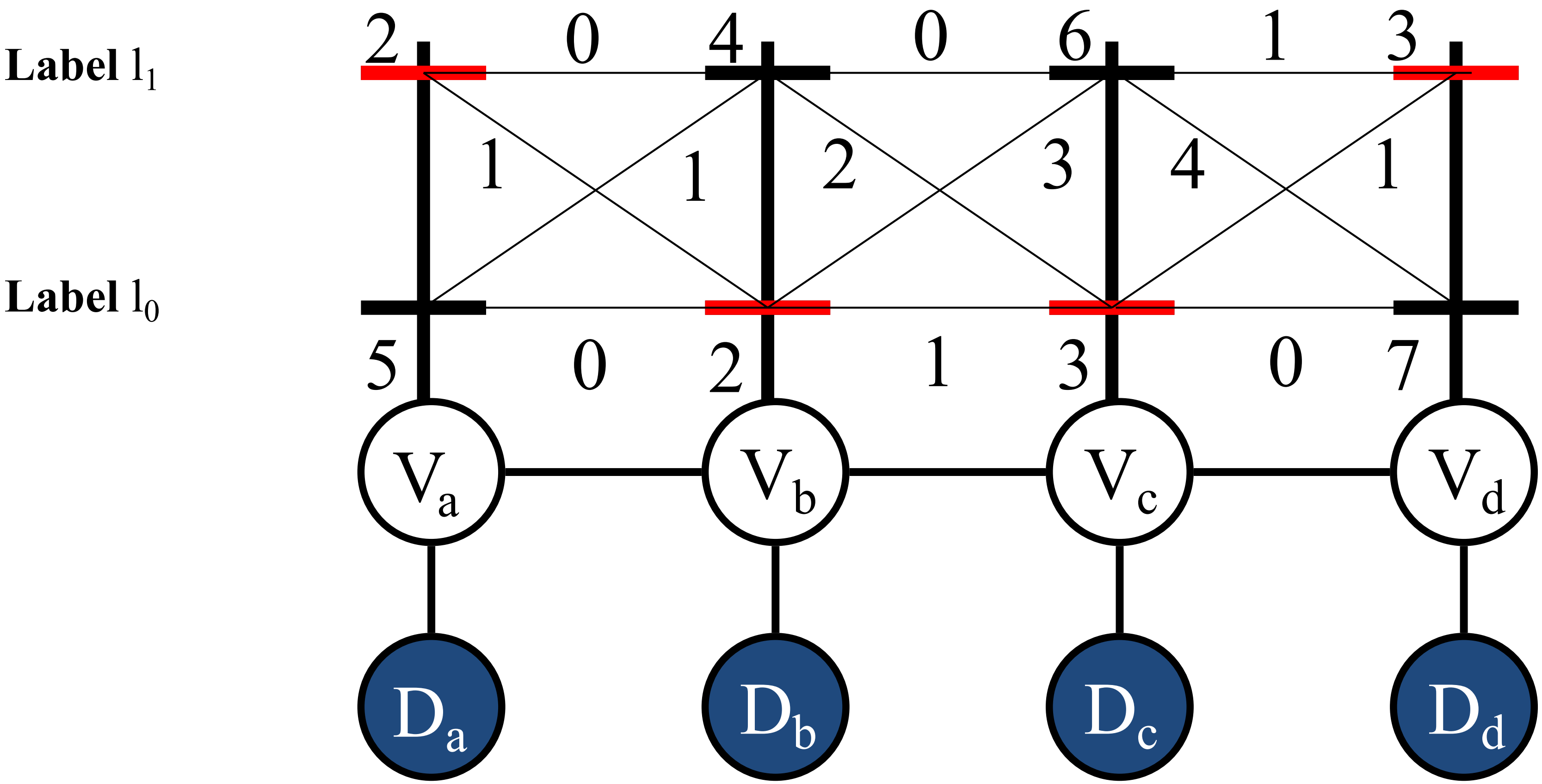
Energy Function



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

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

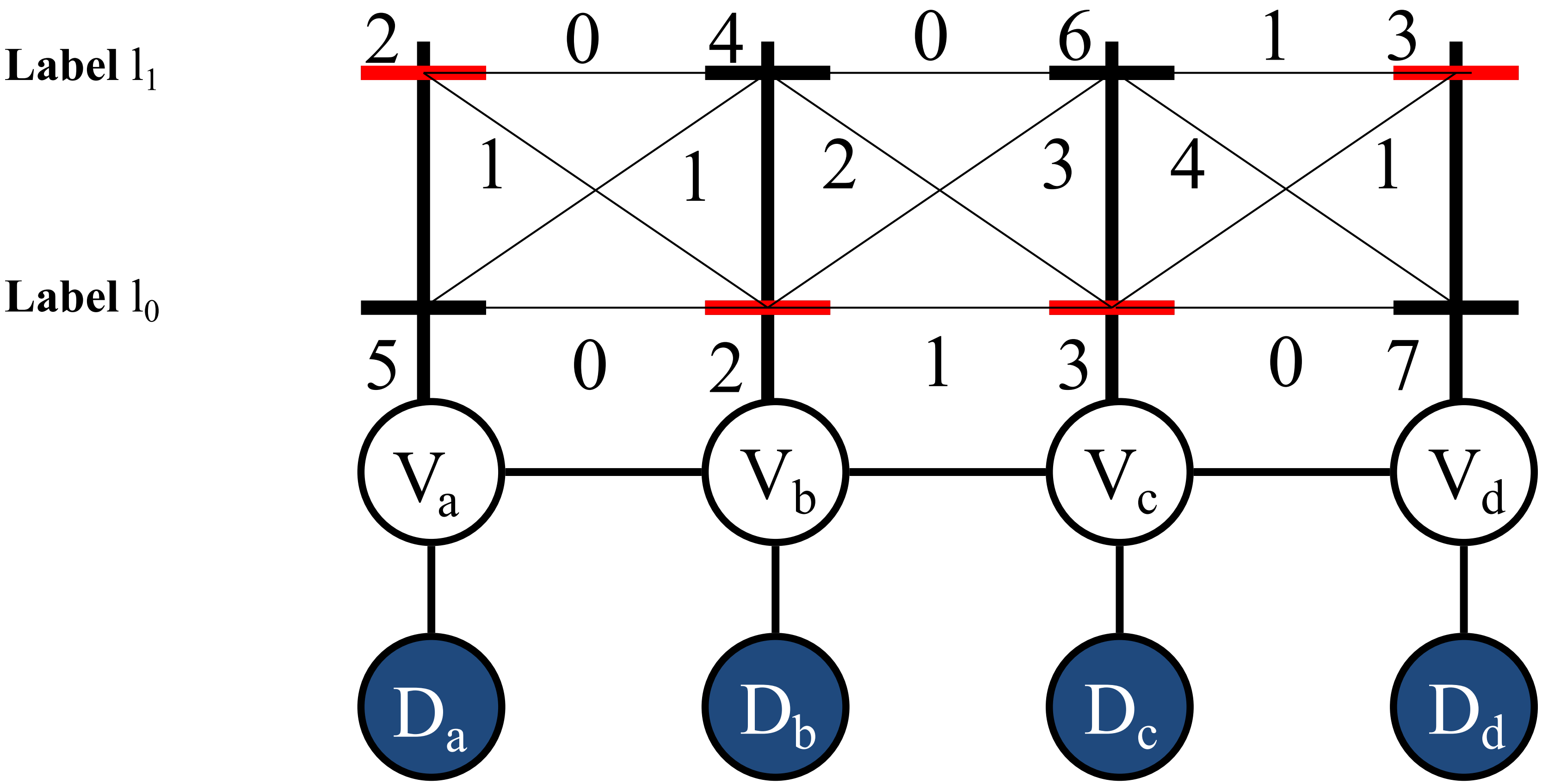
Energy Function



Pairwise Potential

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

Energy Function



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

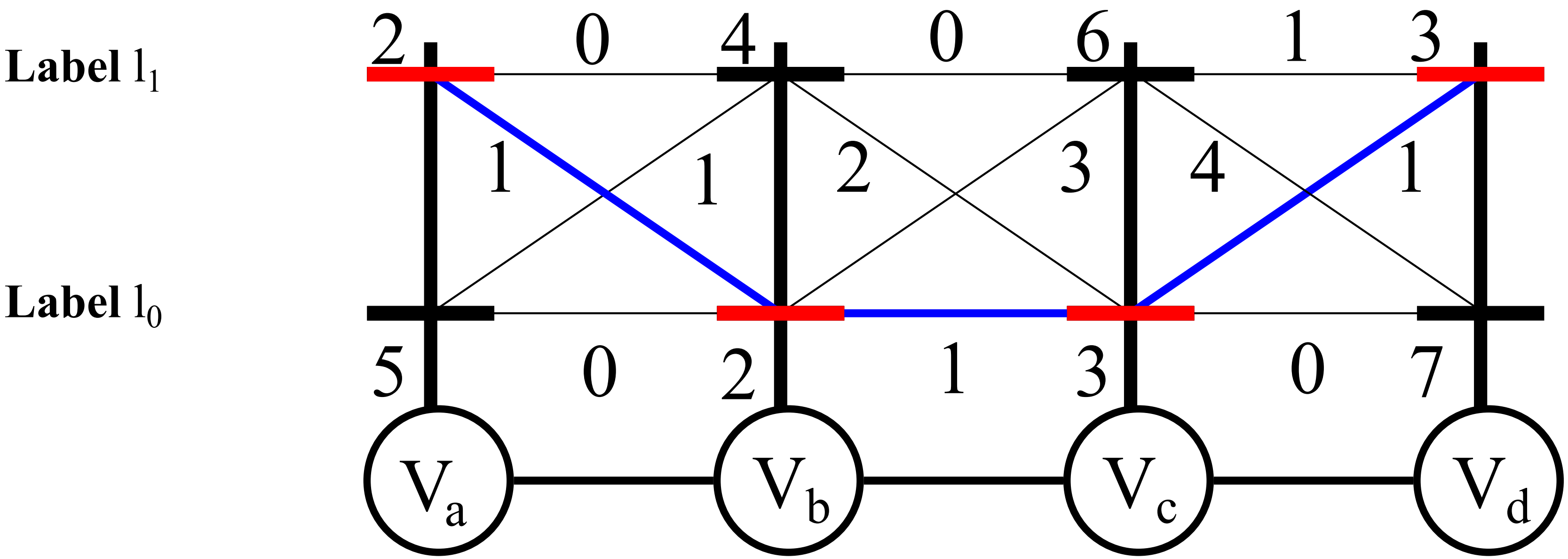
Parameter

Overview

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

- Solutions
  - Belief Propagation and related methods
  - Graph cuts

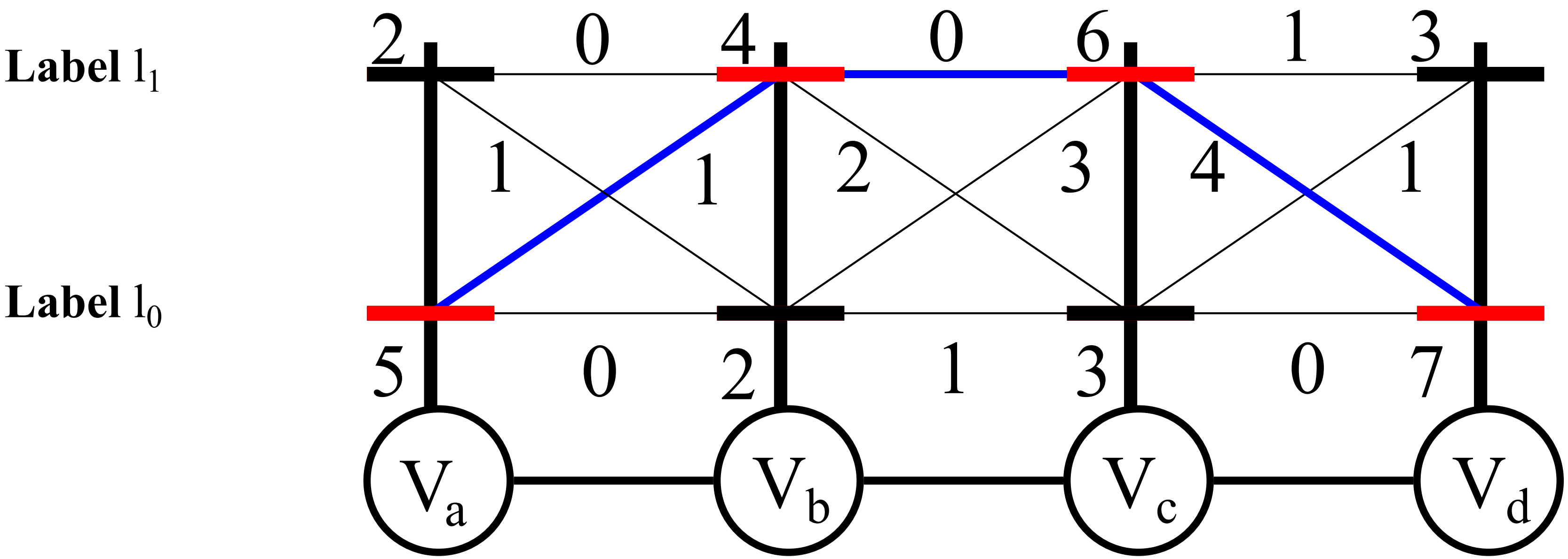
MAP Estimation



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

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

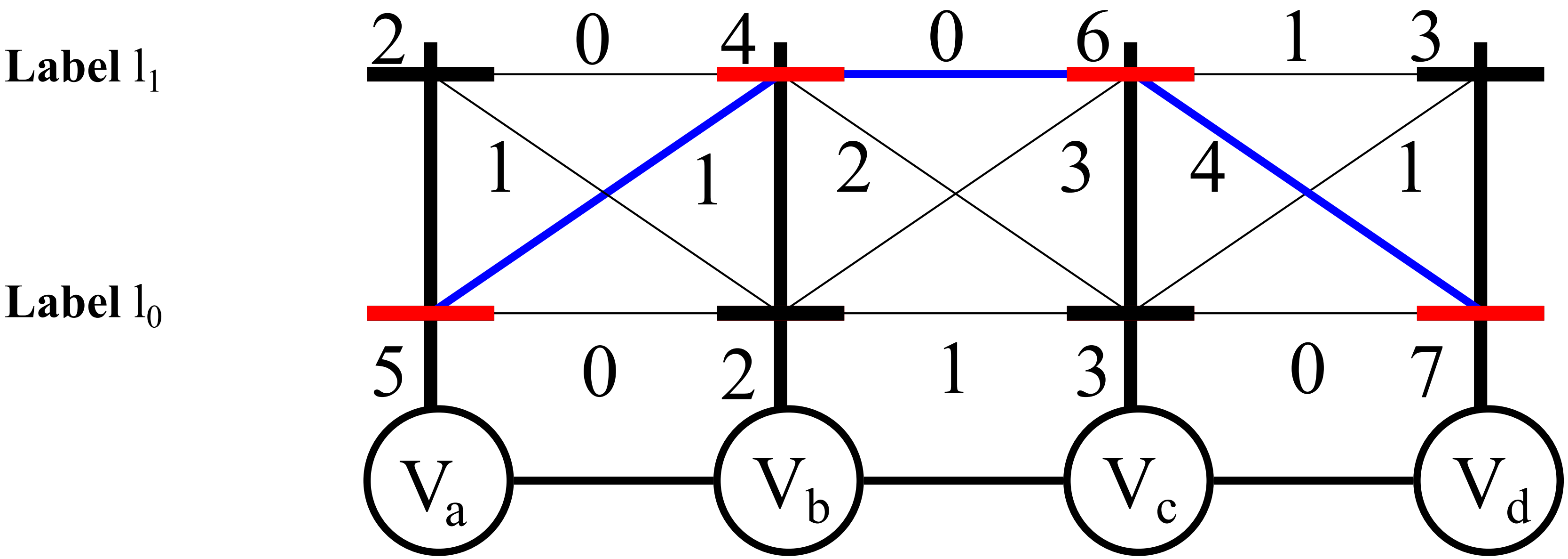
MAP Estimation



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

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

$f^* = \{1, 0, 0, 1\}$

$q^* = 13$

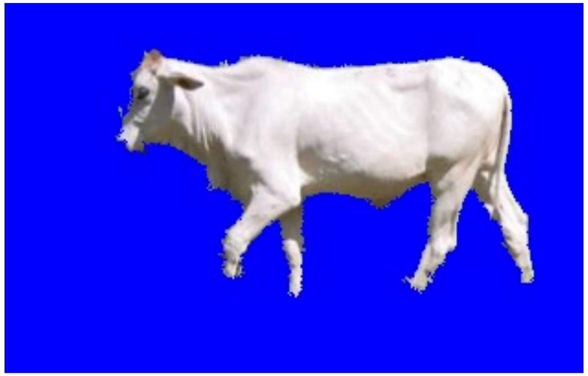
f(a)	f(b)	f(c)	f(d)	Q(f; θ)
0	0	0	0	18
0	0	0	1	15
0	0	1	0	27
0	0	1	1	20
0	1	0	0	22
0	1	0	1	19
0	1	1	0	27
0	1	1	1	20

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

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

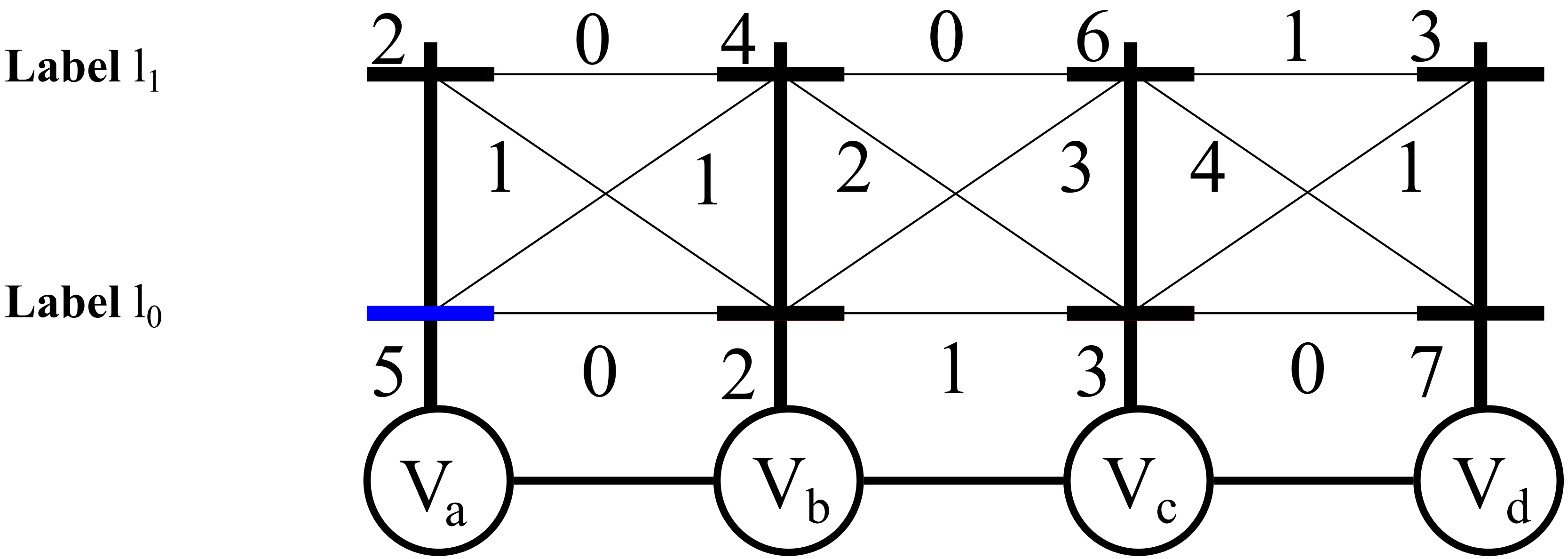
- Exact inference is possible in some cases, e.g.,
  – Low treewidth graphs → message-passing
  – Submodular potentials → graph cuts
- Efficient approximate inference algorithms exist
  – Message passing on general graphs
  – Move-making algorithms
  – Relaxation algorithms

Overview

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

- Solutions
  - Belief Propagation and related methods
  - Graph cuts

Min-Marginals



Not a marginal (no summation)

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

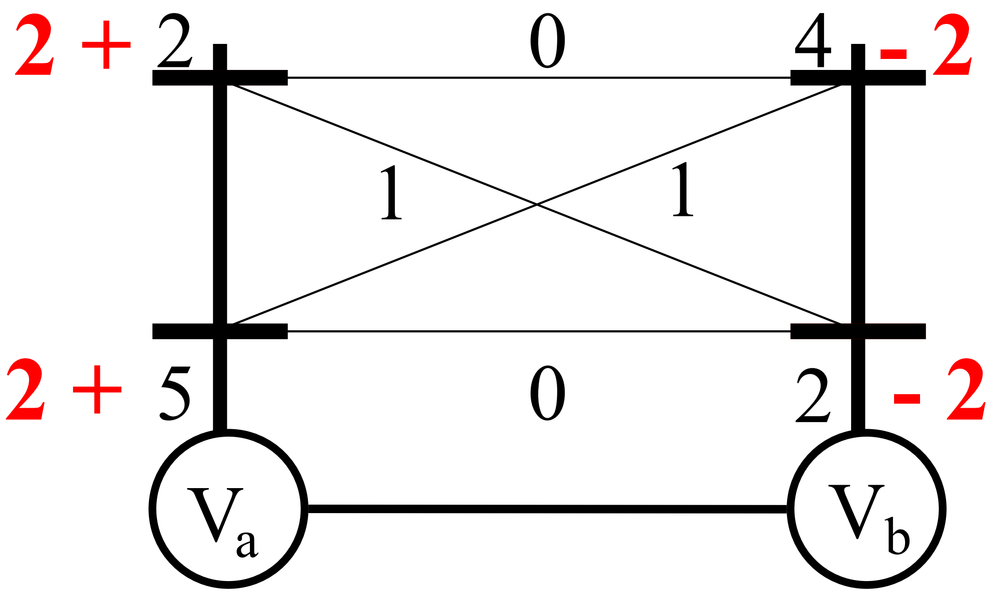
$$q_{a;i} = \min Q(f; \theta) \quad \text{s.t. } f(a) = i$$

Overview

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

- Solutions
  - Belief Propagation and related methods
  - Graph cuts

Reparameterization



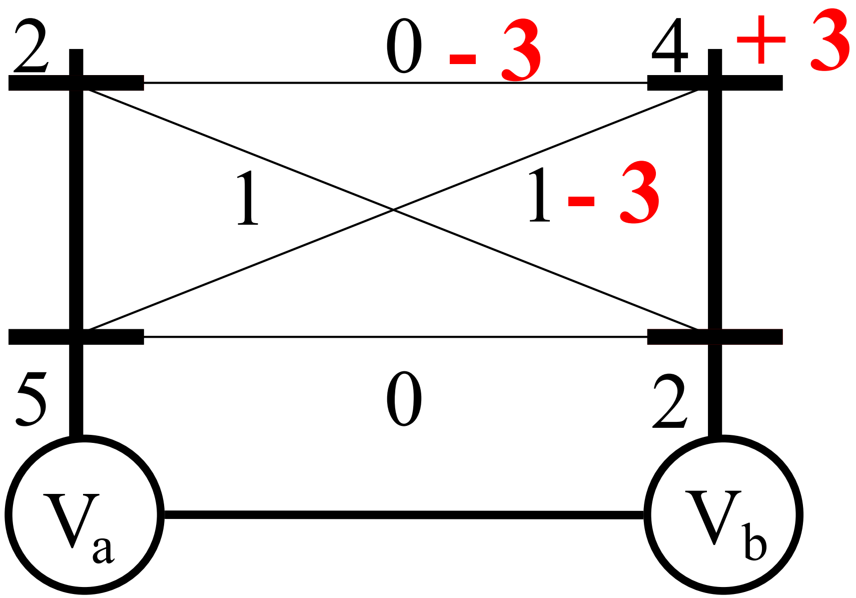
f(a)	f(b)	Q(f; θ)
0	0	7 + 2 - 2
0	1	10 + 2 - 2
1	0	5 + 2 - 2
1	1	6 + 2 - 2

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

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

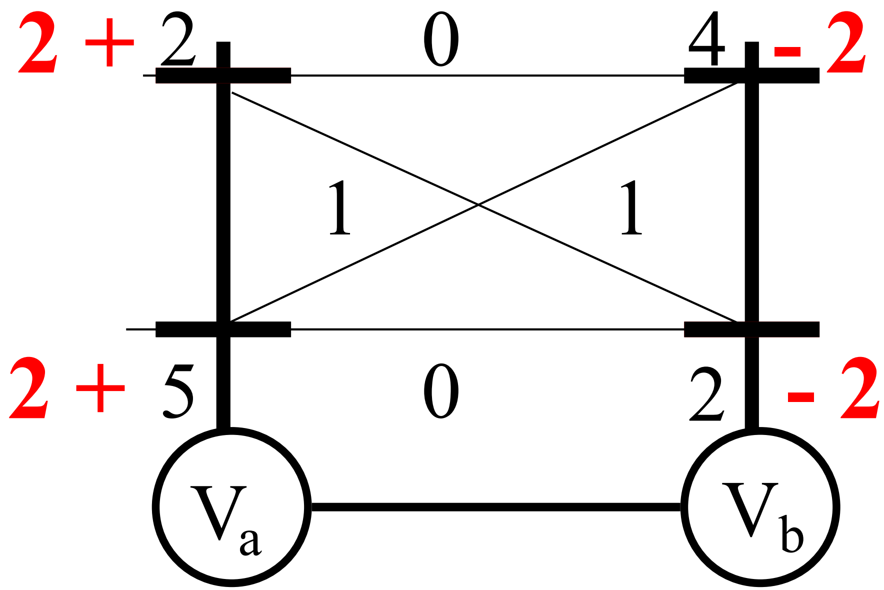
Equivalently

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

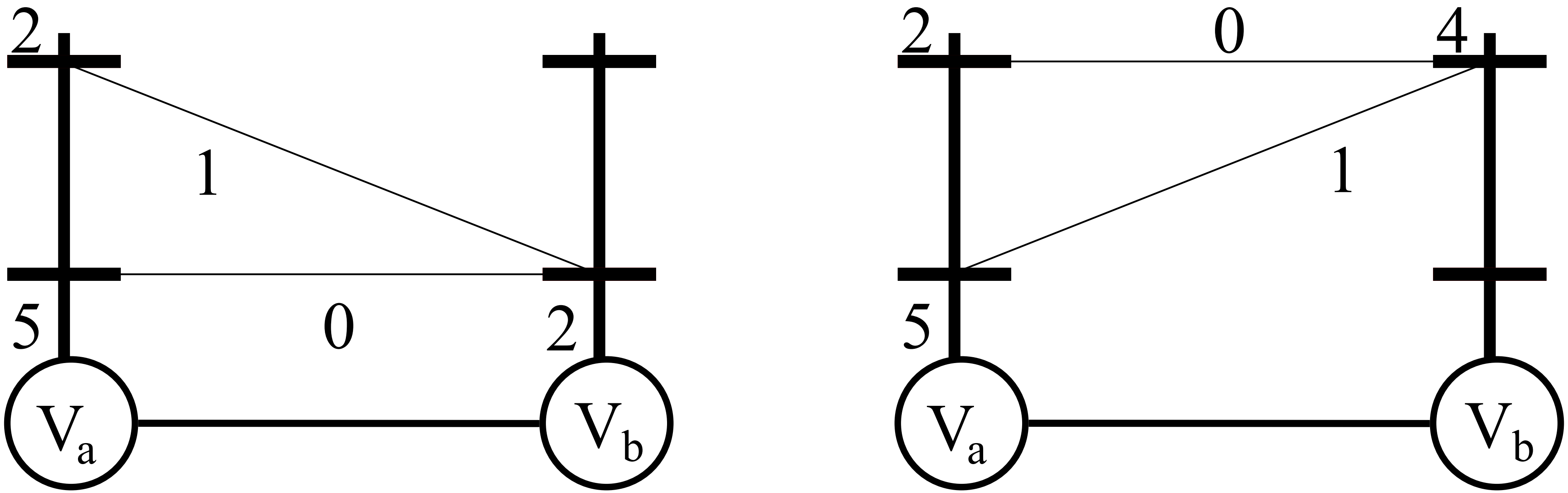
- Basics: problem formulation
  - Energy Function
  - MAP Estimation
  - Computing min-marginals
  - Reparameterization

- Solutions
  - Belief Propagation and related methods
  - Graph cuts

Belief Propagation

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

Two Variables

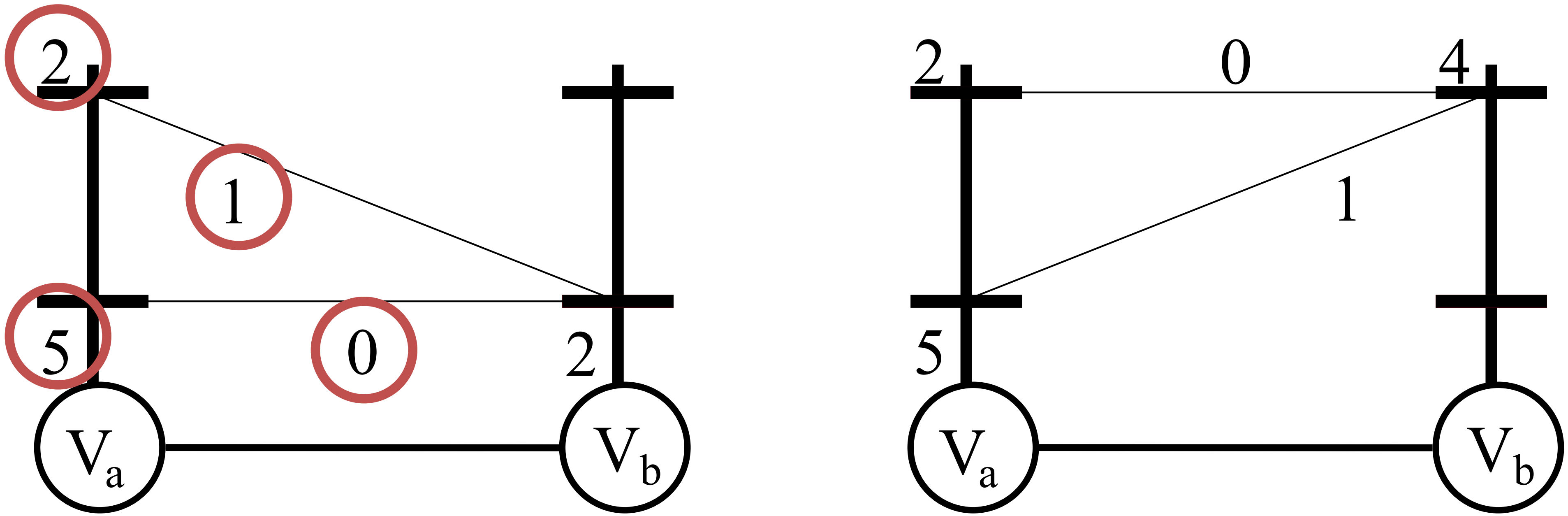


Add a constant to one $\theta_{b;k}$

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

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

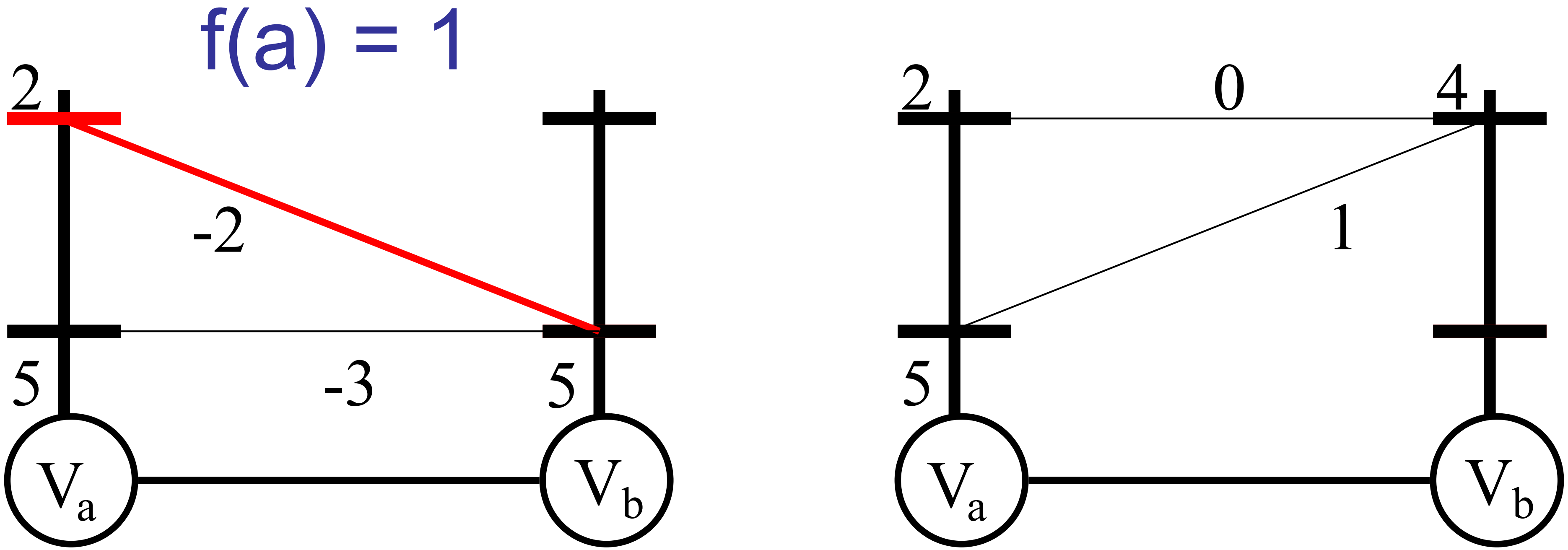
Two Variables



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

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

Two Variables

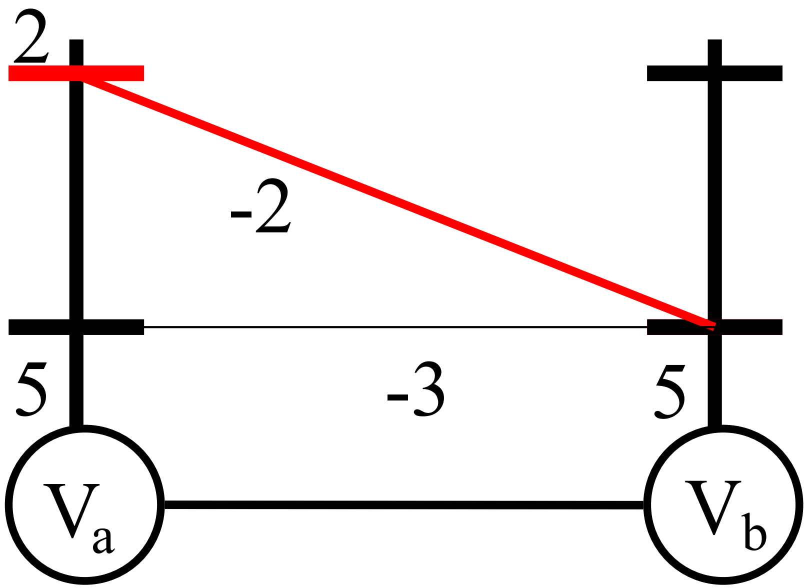


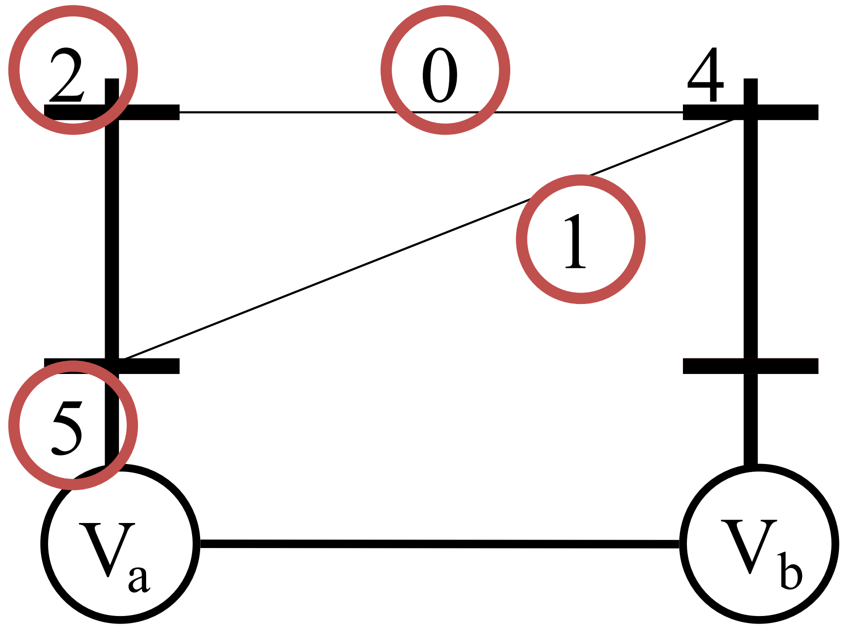
$$\theta'_{b;0} = q_{b;0}$$

Potentials along the red path add up to 0

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

Two Variables



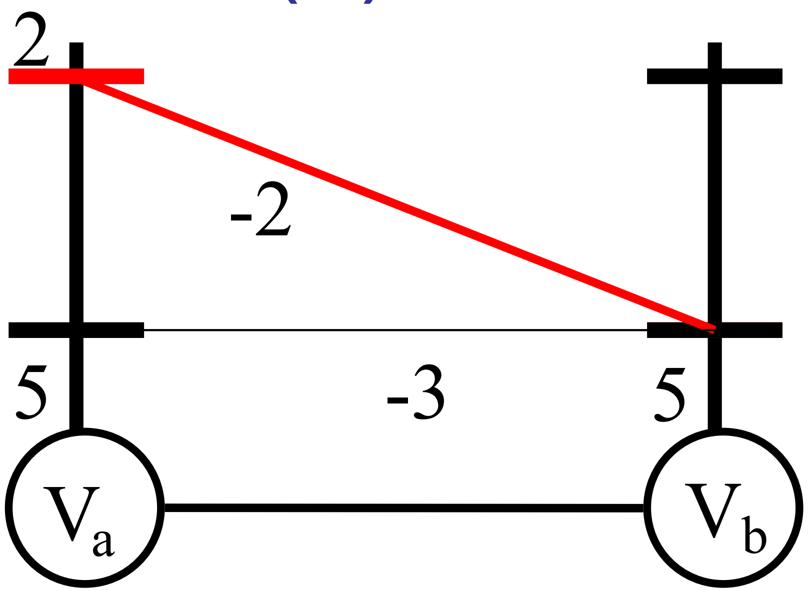


$$M_{ab;1} = \min \begin{array}{l} \theta_{a;0} + \theta_{ab;01} = 5 + 1 \\ \theta_{a;1} + \theta_{ab;11} = 2 + 0 \end{array}$$

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

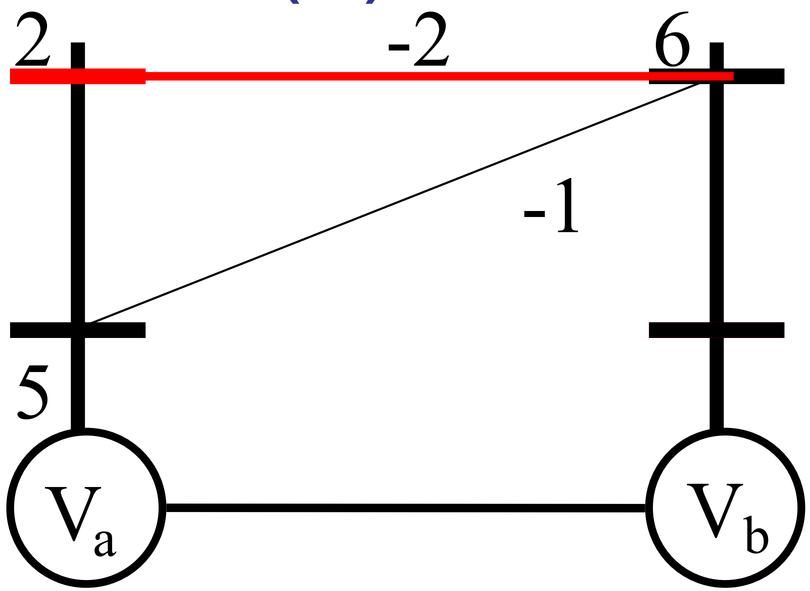
Two Variables

f(a) = 1



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

f(a) = 1

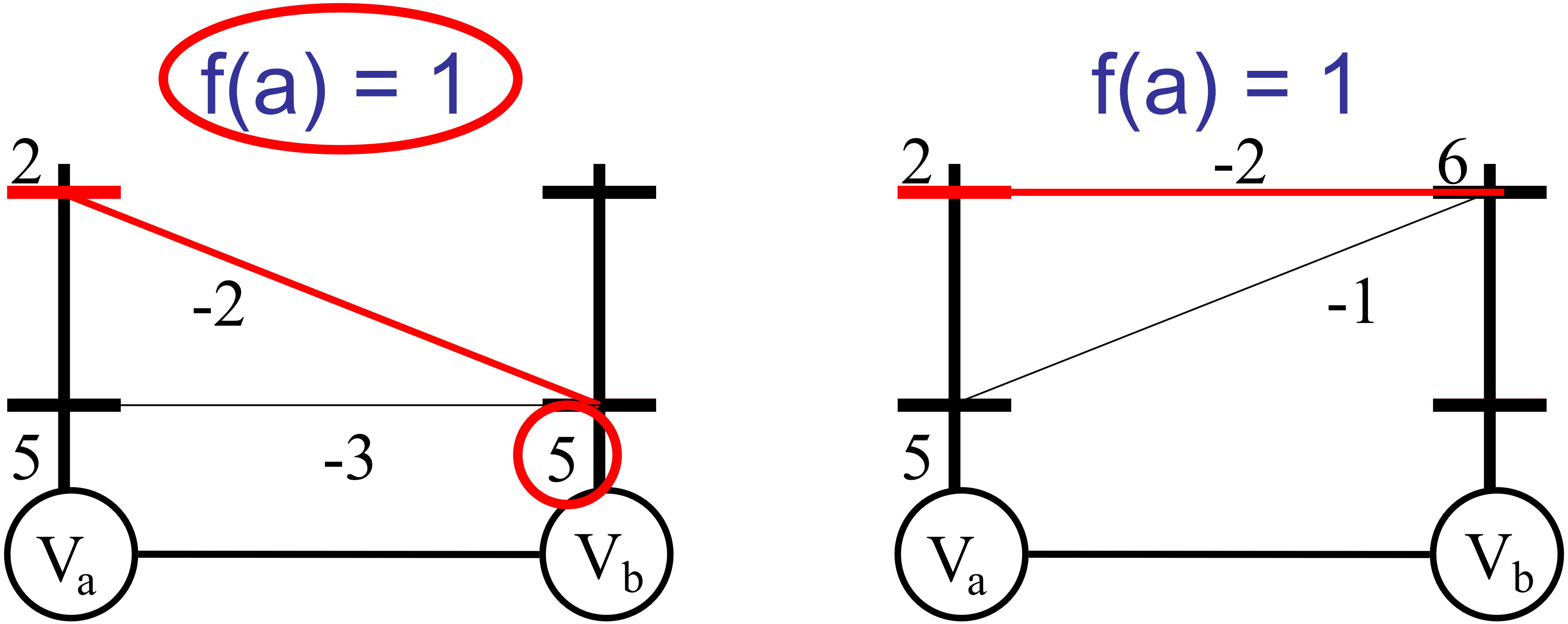


$$\theta'_{b;1} = q_{b;1}$$

Minimum of min-marginals = MAP estimate

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

Two Variables

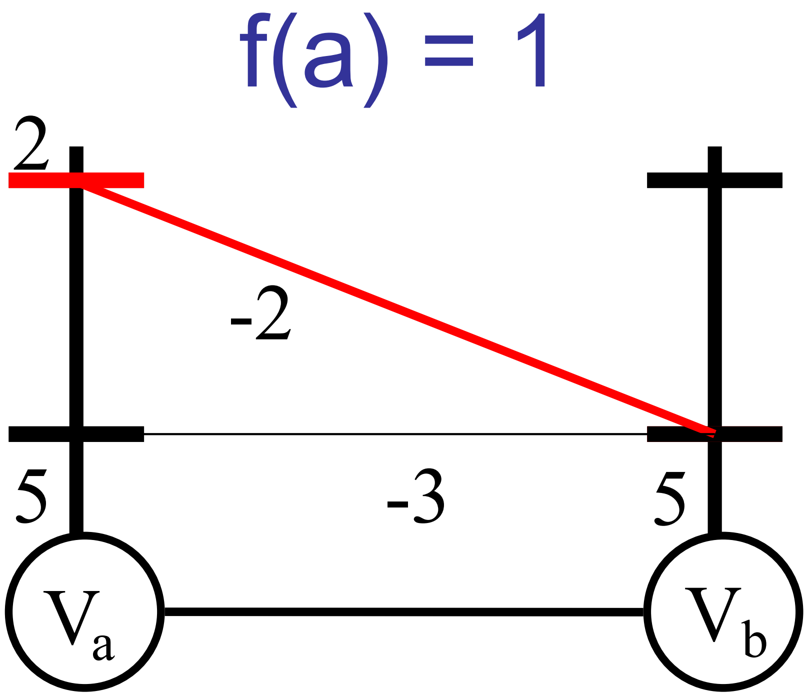


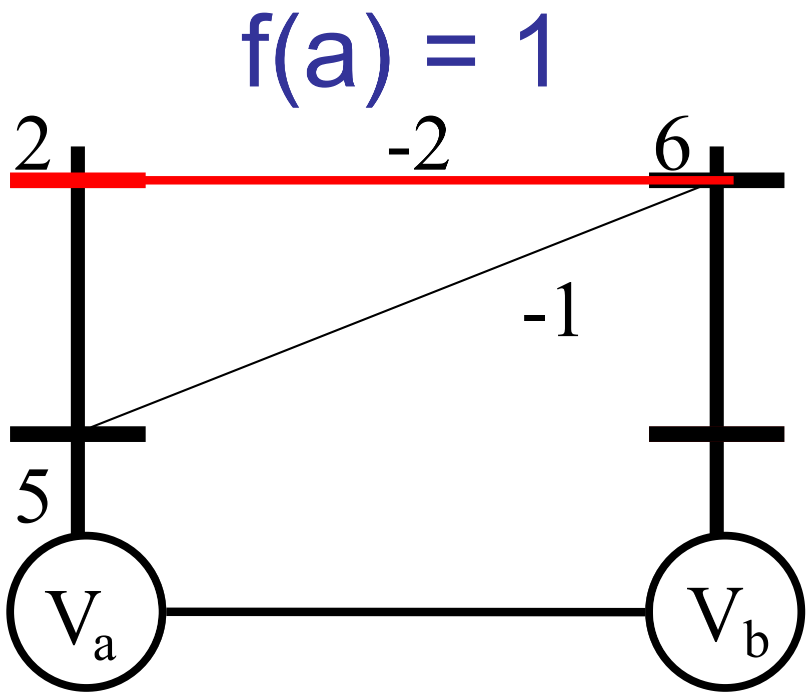
$\theta'_{b;0} = q_{b;0}$ $\theta'_{b;1} = q_{b;1}$

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

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

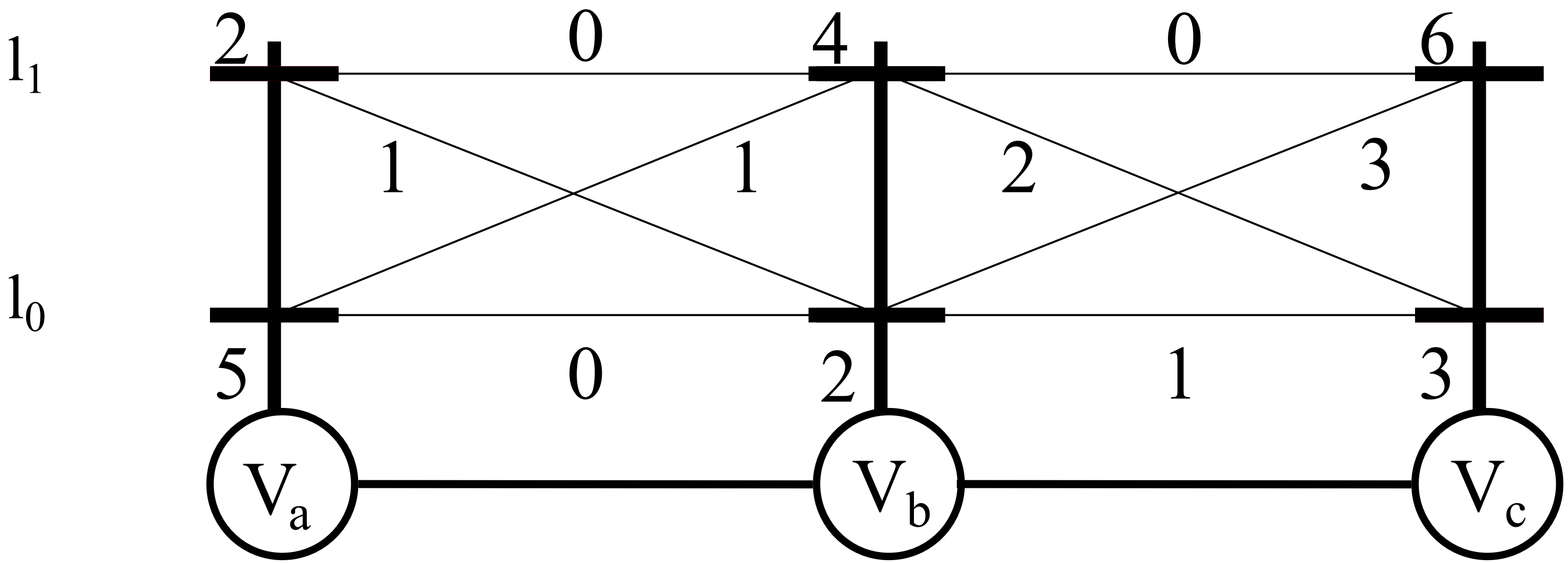
$$\theta'_{ab;ik} = \theta_{ab;ik} - M_{ab;k} - M_{ba;i}$$

Reparameterization of (a,b) in Belief Propagation

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

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

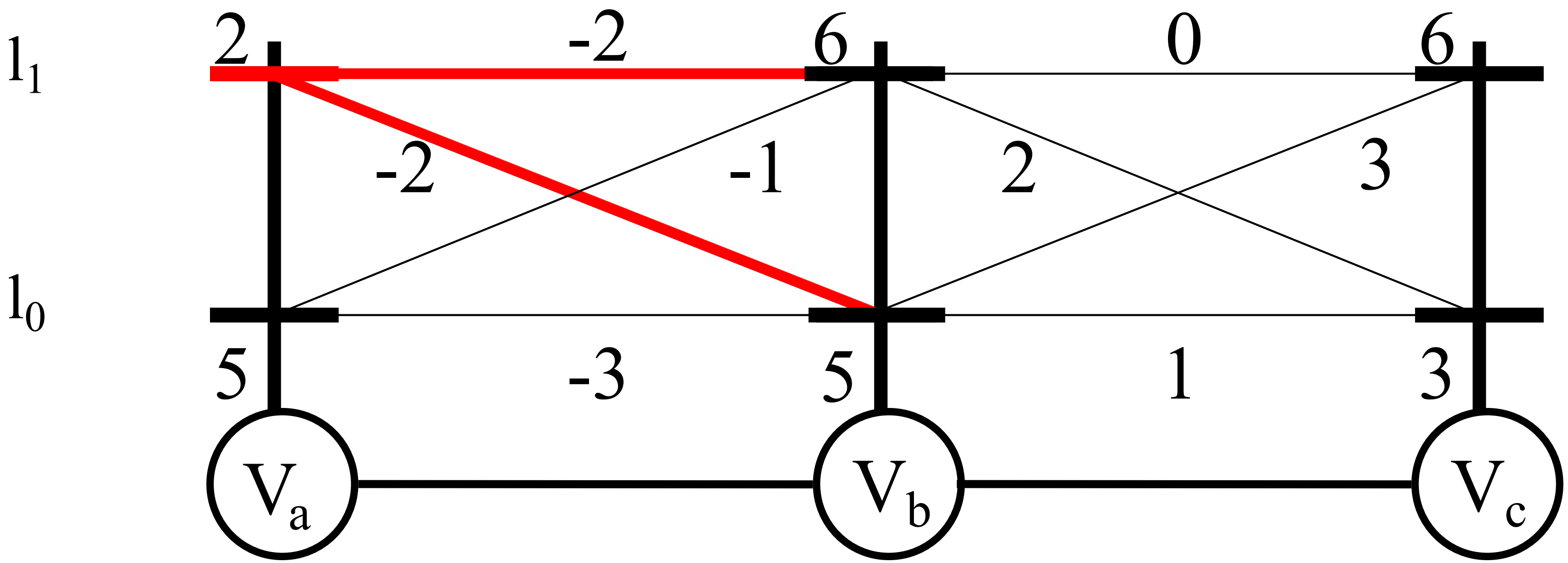
Three Variables



Reparameterize the edge (a,b) as before

Three Variables

f(a) = 1

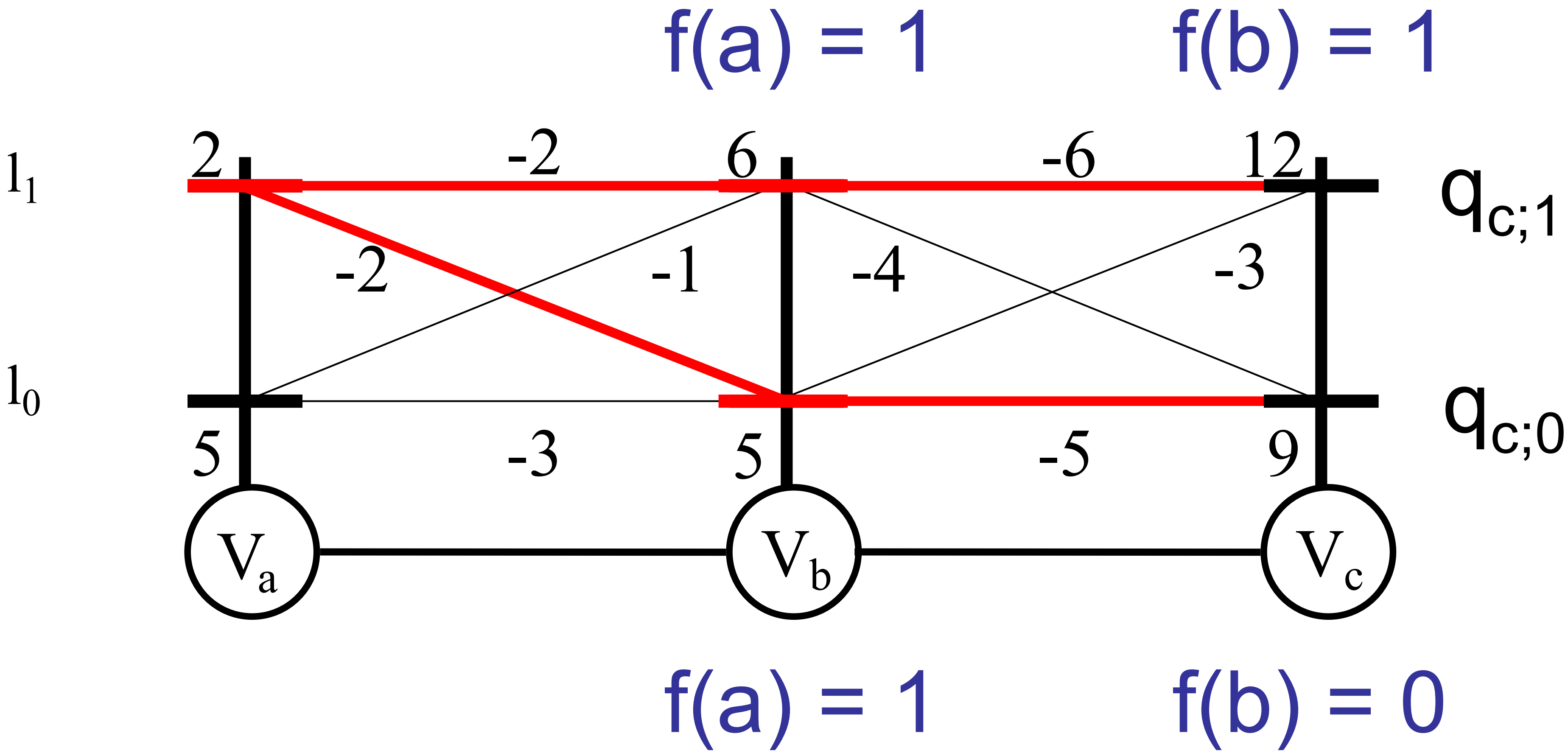


f(a) = 1

Reparameterize the edge (a,b) as before

Potentials along the red path add up to 0

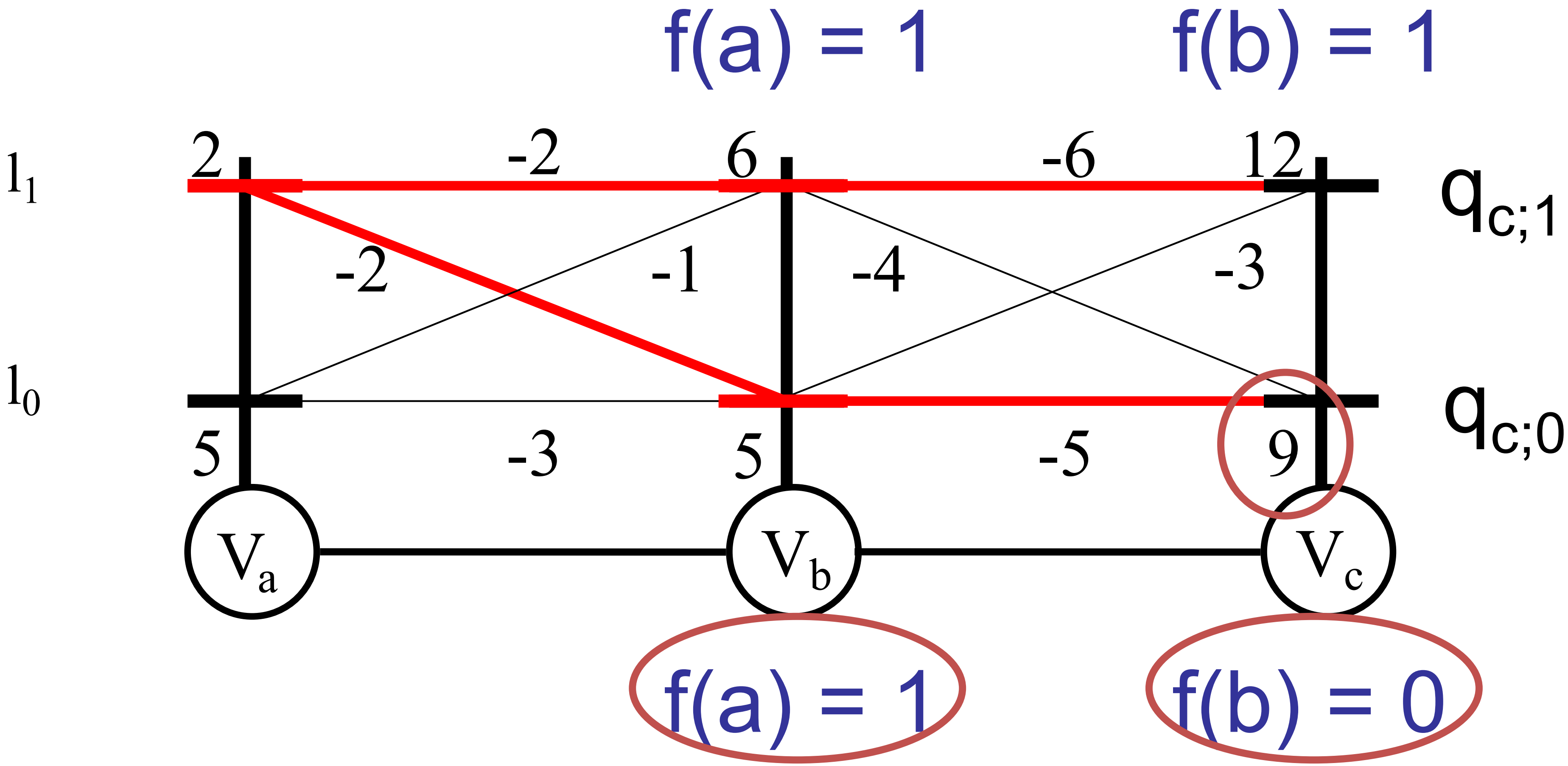
Three Variables



Reparameterize the edge (b,c) as before

Potentials along the red path add up to 0

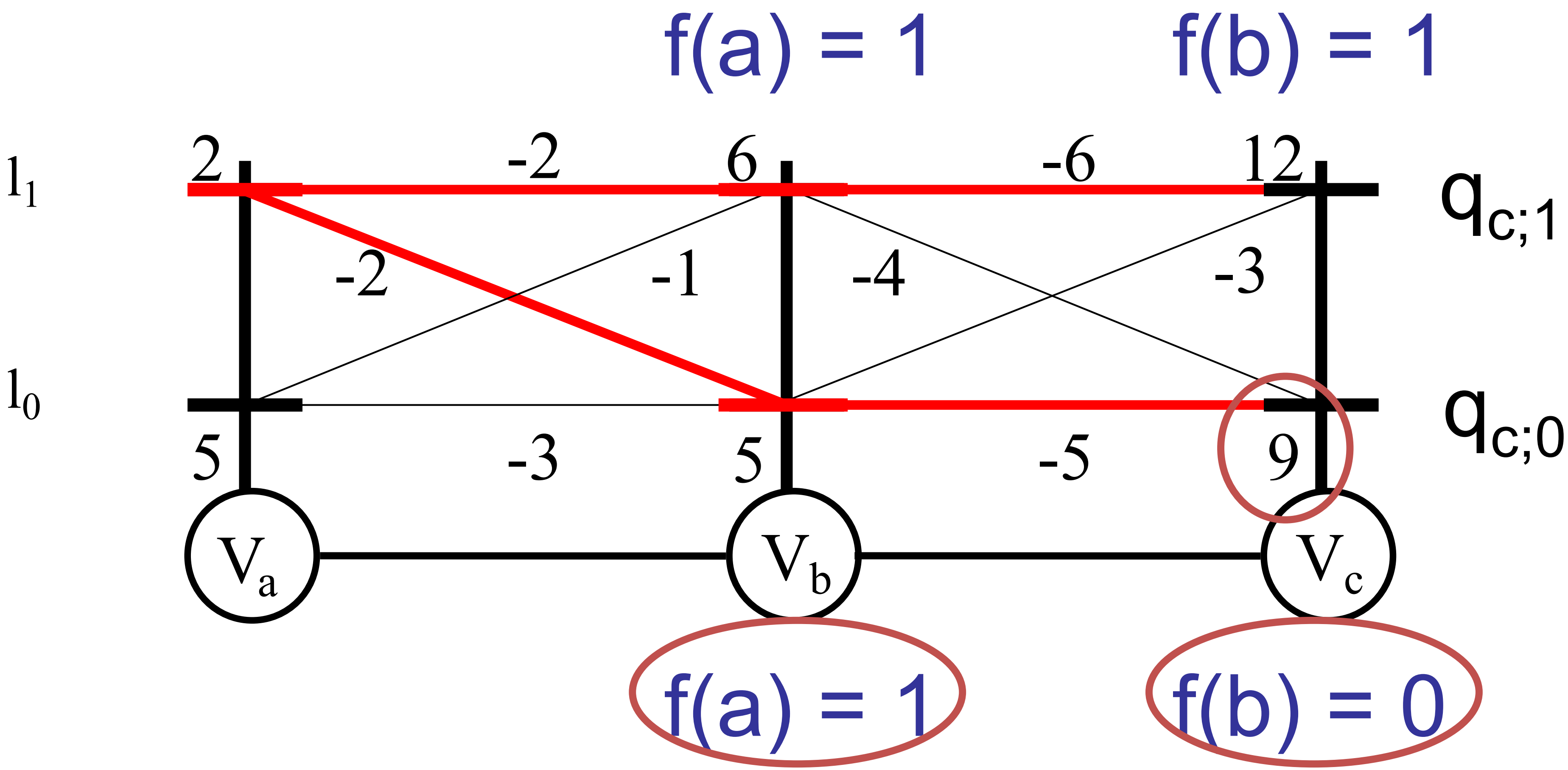
Three Variables



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 $\equiv$ 2 variables + book-keeping

n variables $\equiv$ (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

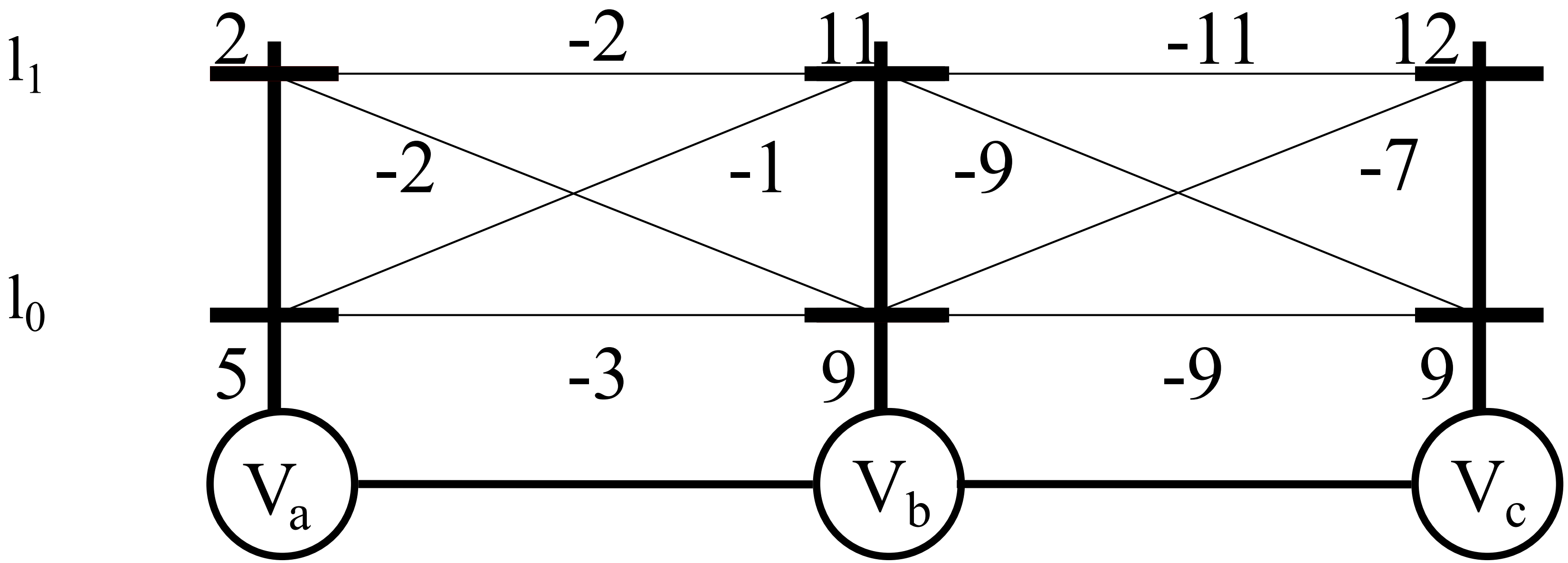
Reparameterize current edge (a,b)

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

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

Repeat

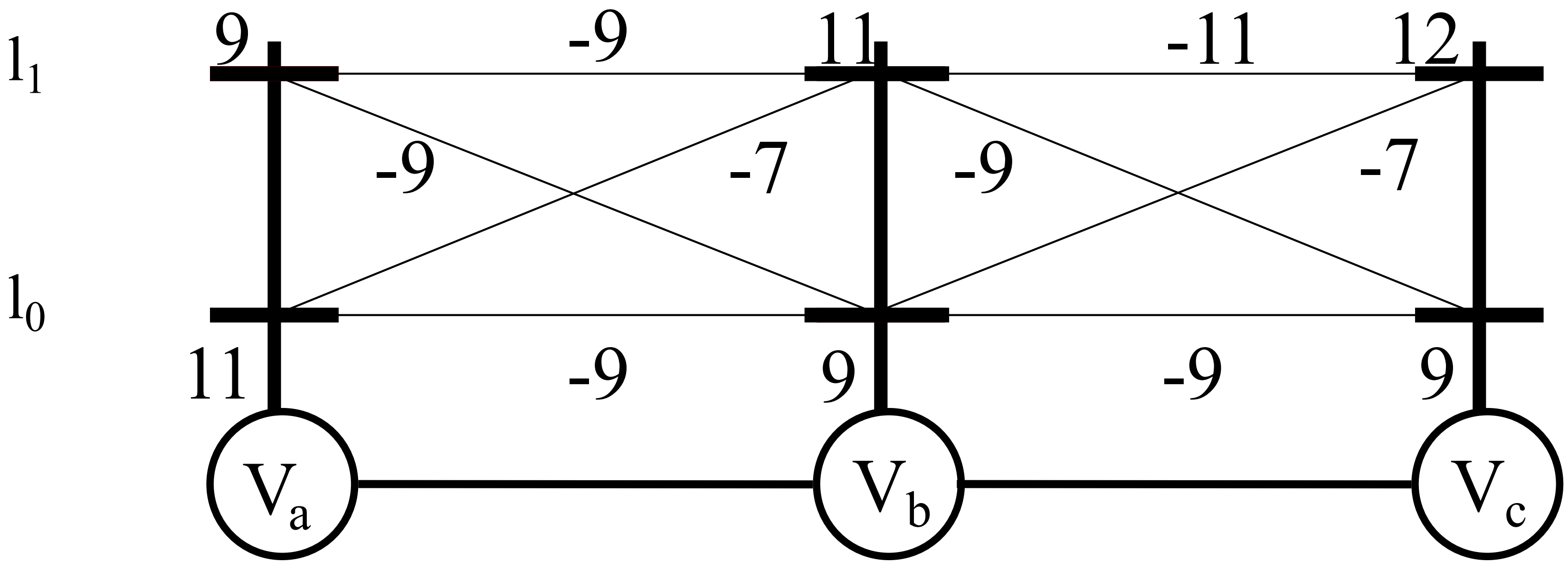
Three Variables



Reparameterize the edge (c,b) as before

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

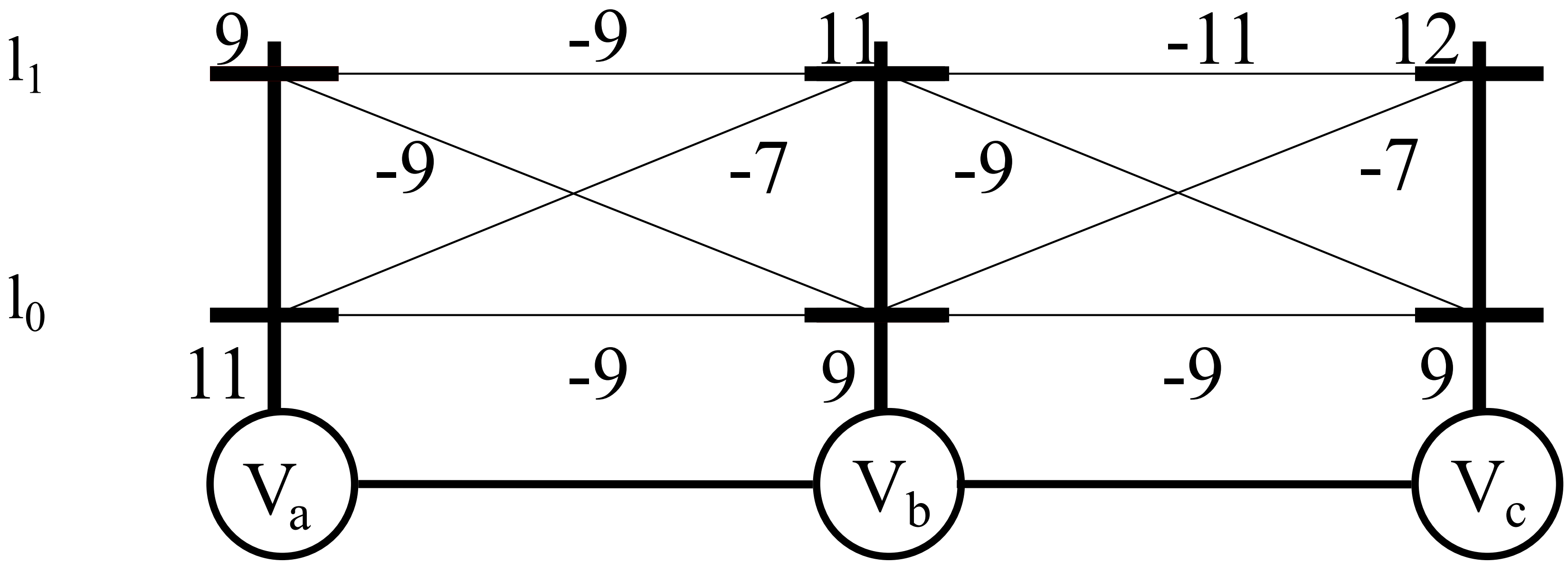
Three Variables



Reparameterize the edge (b,a) as before

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

Three Variables



Forward Pass ➔ ← Backward Pass

All min-marginals are computed

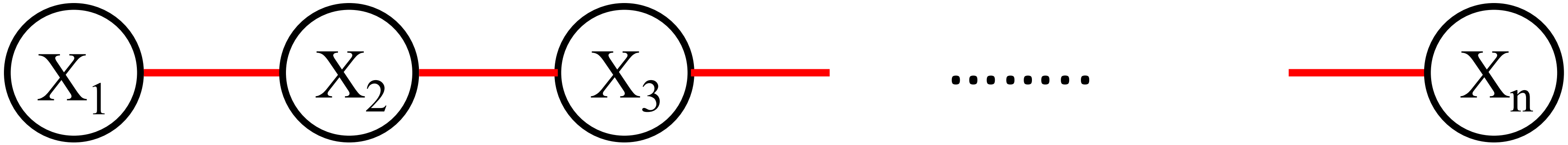
Chains

Reparameterize the edge (1,2)

Chains

Reparameterize the edge (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

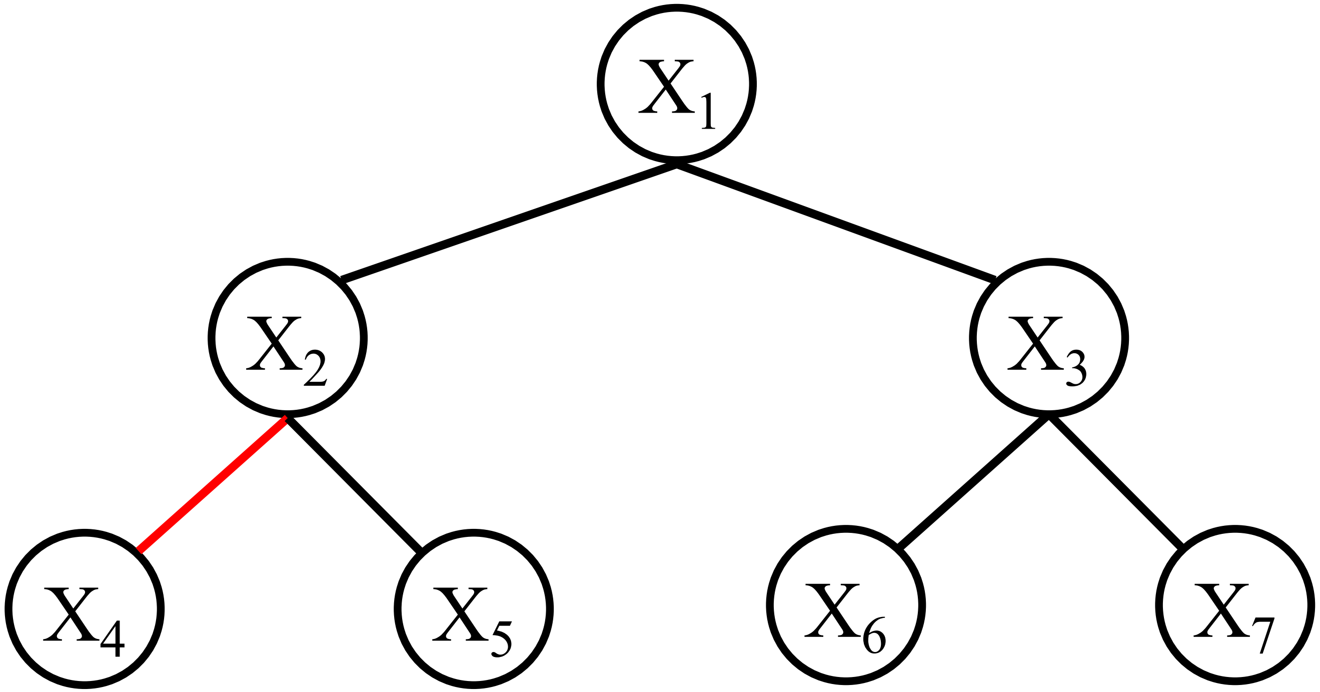
Number of reparameterization constants = (n-1)h

Complexity for each constant = O(h)

Total complexity = $O(nh^2)$

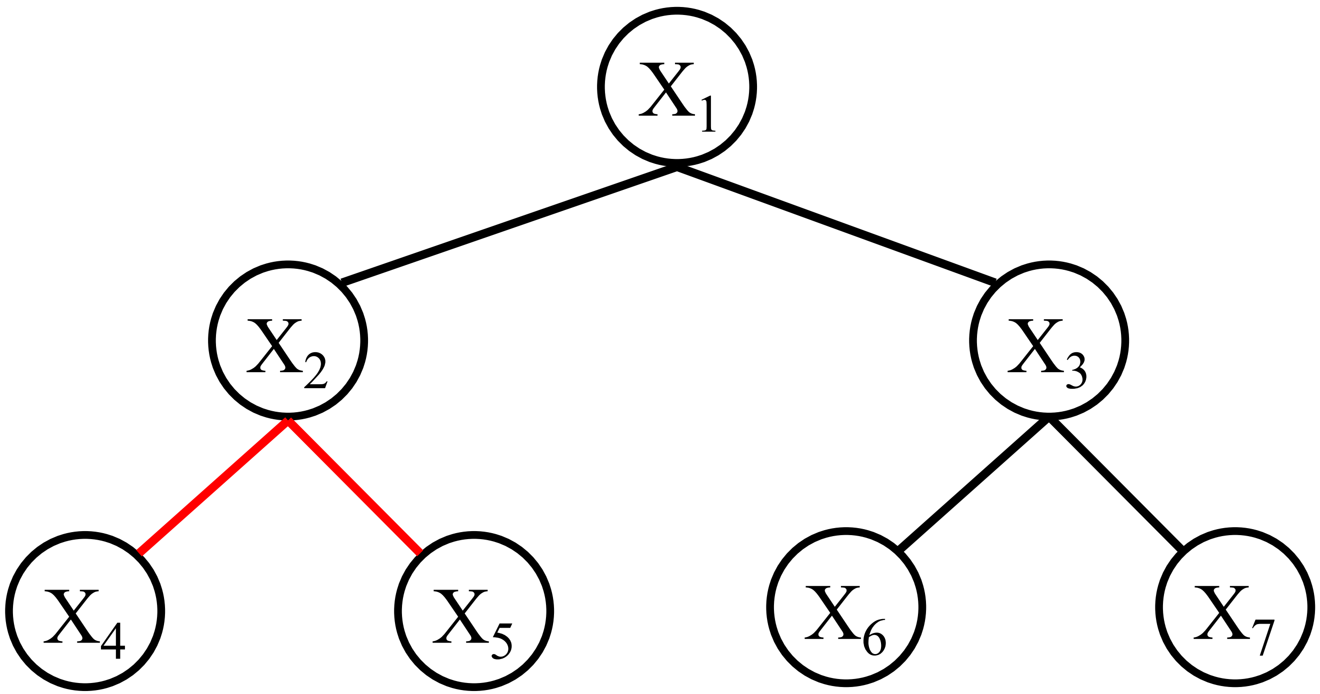
Better than brute-force $O(h^n)$

Trees



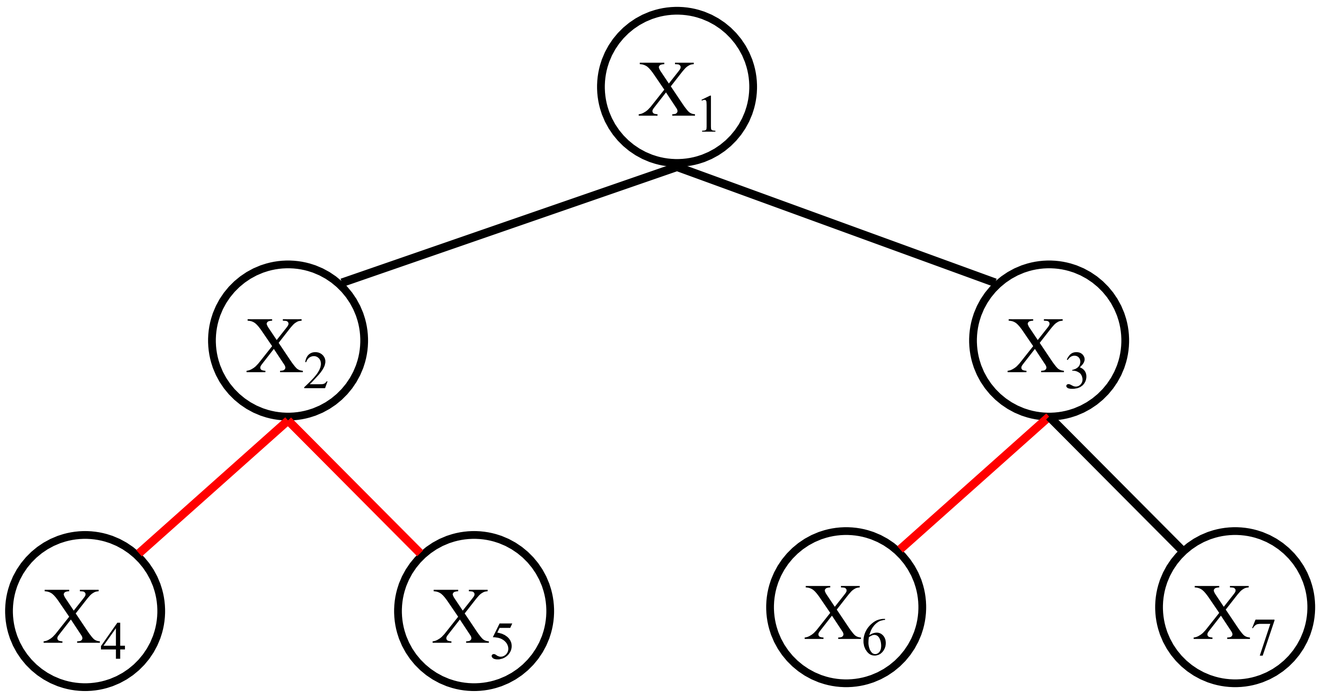
Reparameterize the edge (4,2)

Trees



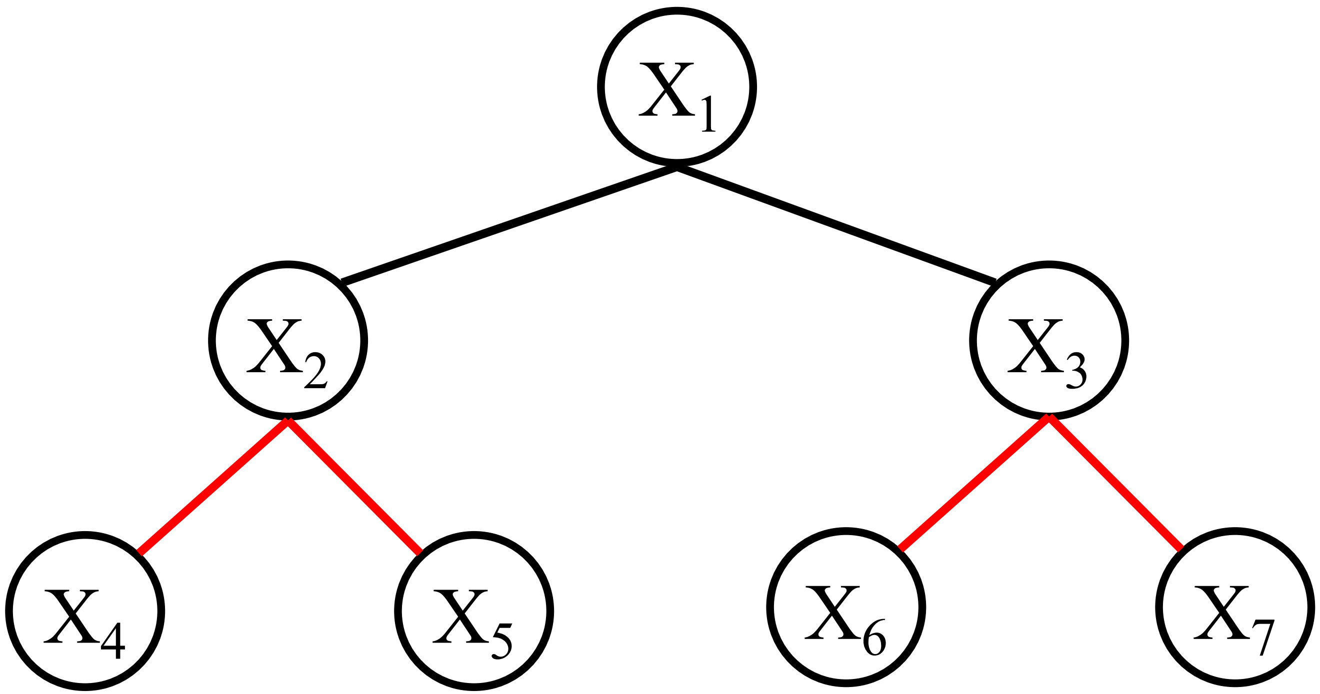
Reparameterize the edge (5,2)

Trees



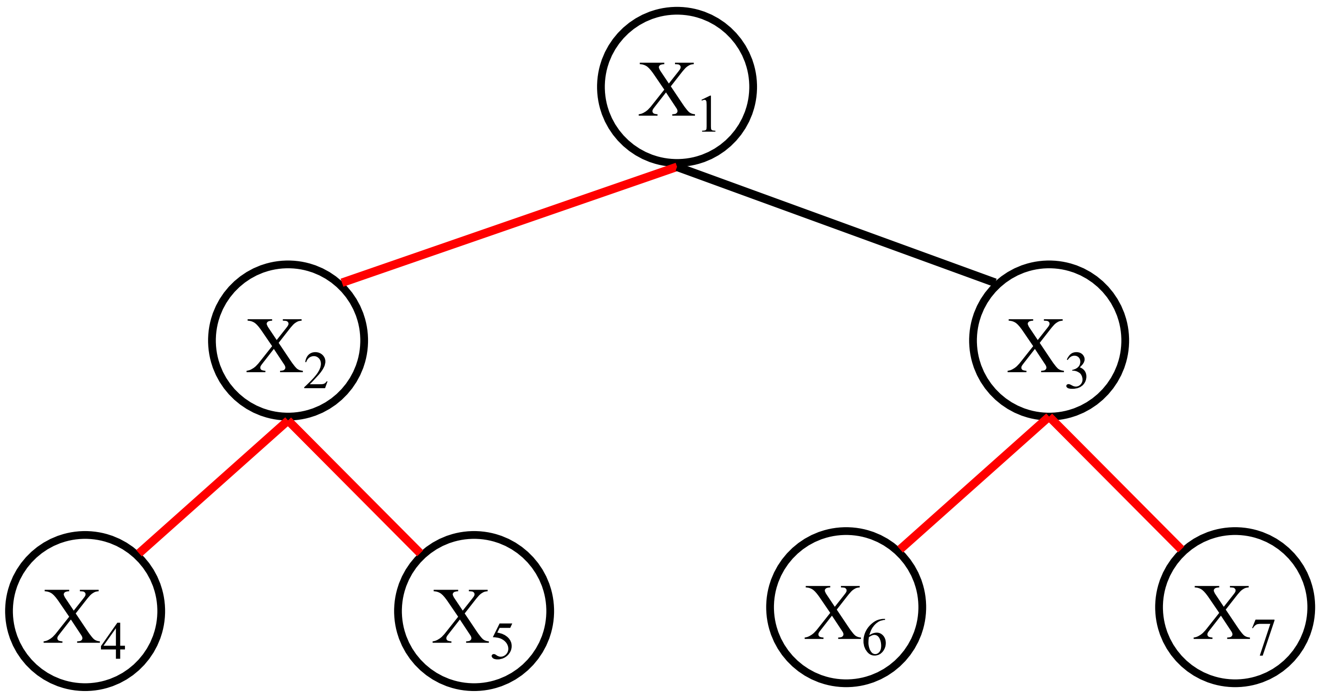
Reparameterize the edge (6,3)

Trees



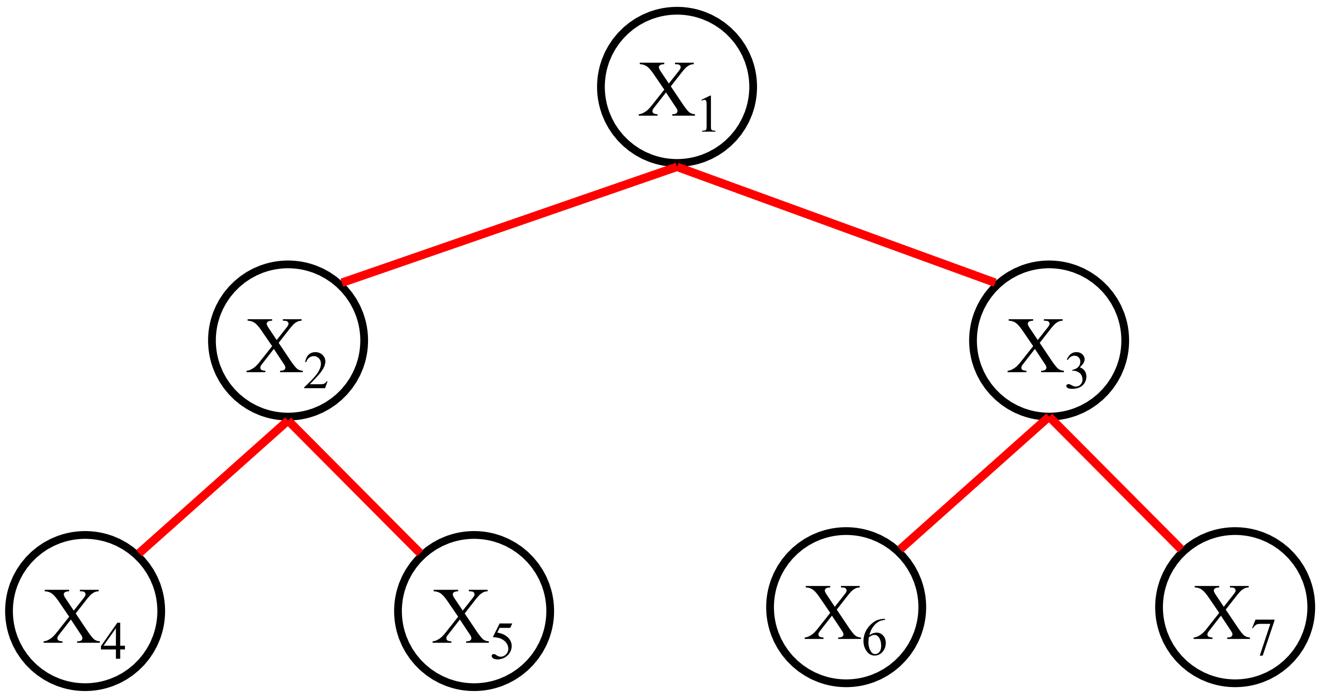
Reparameterize the edge (7,3)

Trees



Reparameterize the edge (2,1)

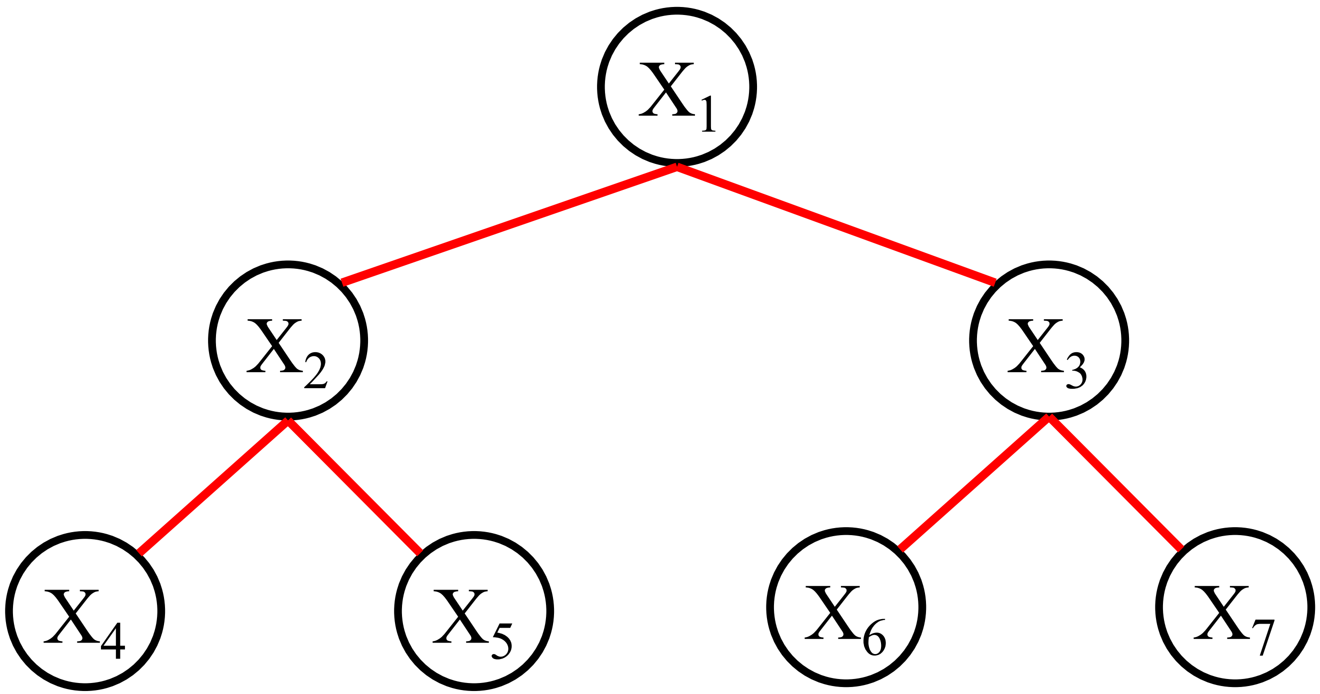
Trees



Reparameterize the edge (3,1)

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

Trees



Start from leaves and move towards root

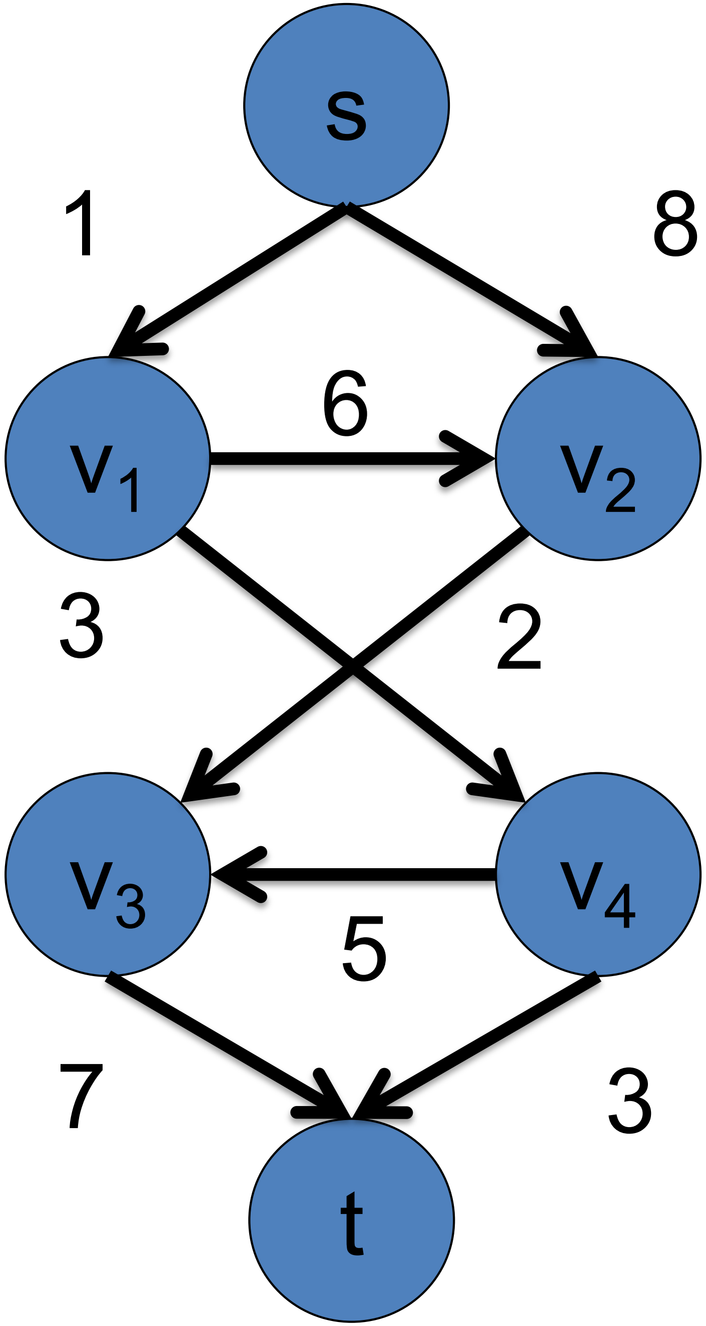
Pick the minimum of min-marginals

Backtrack to find the best labeling **x**

Outline

- Preliminaries
  - **s-t Flow**
  - s-t Cut
  - Flows vs. Cuts
- Maximum Flow
- Algorithms
- Energy minimization with max flow/min cut

s-t Flow



Function flow: A → R

Flow of arc ≤ arc capacity

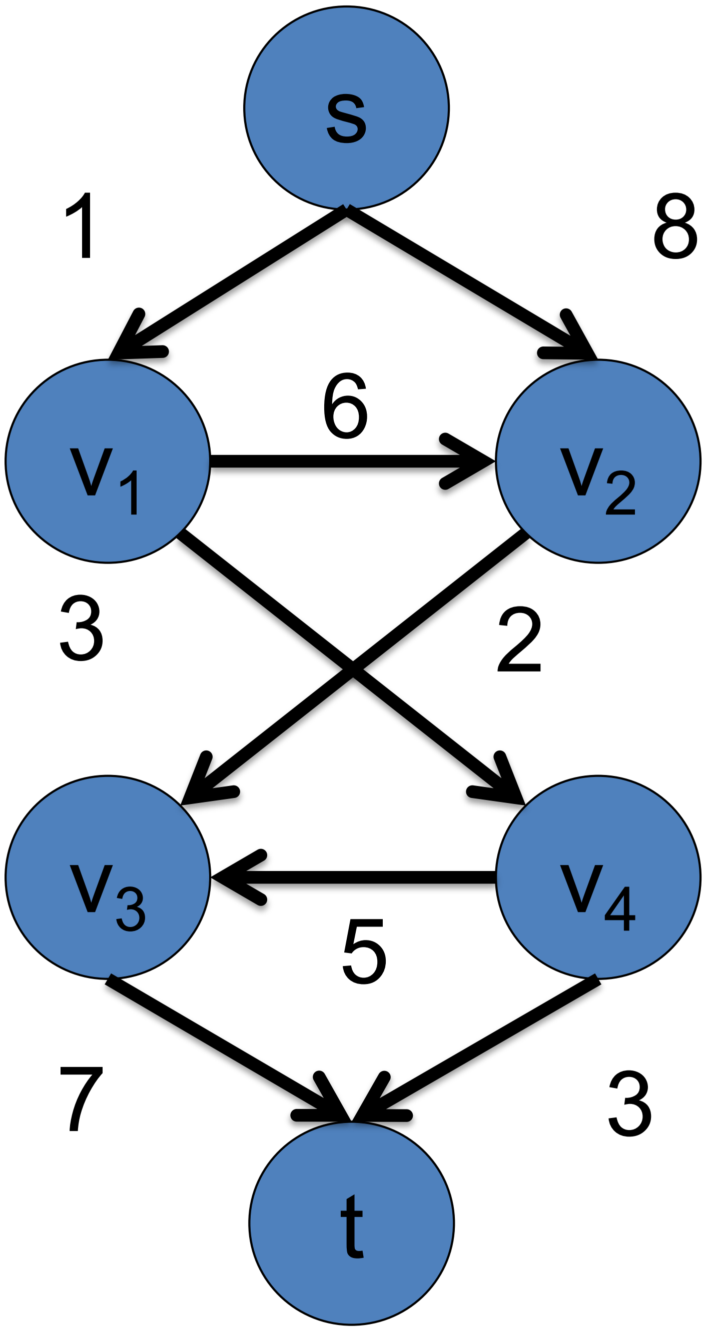
Flow is non-negative

For all vertex except s,t

Incoming flow

= Outgoing flow

s-t Flow



Function flow: A ➔ R

flow(a) ≤ c(a)

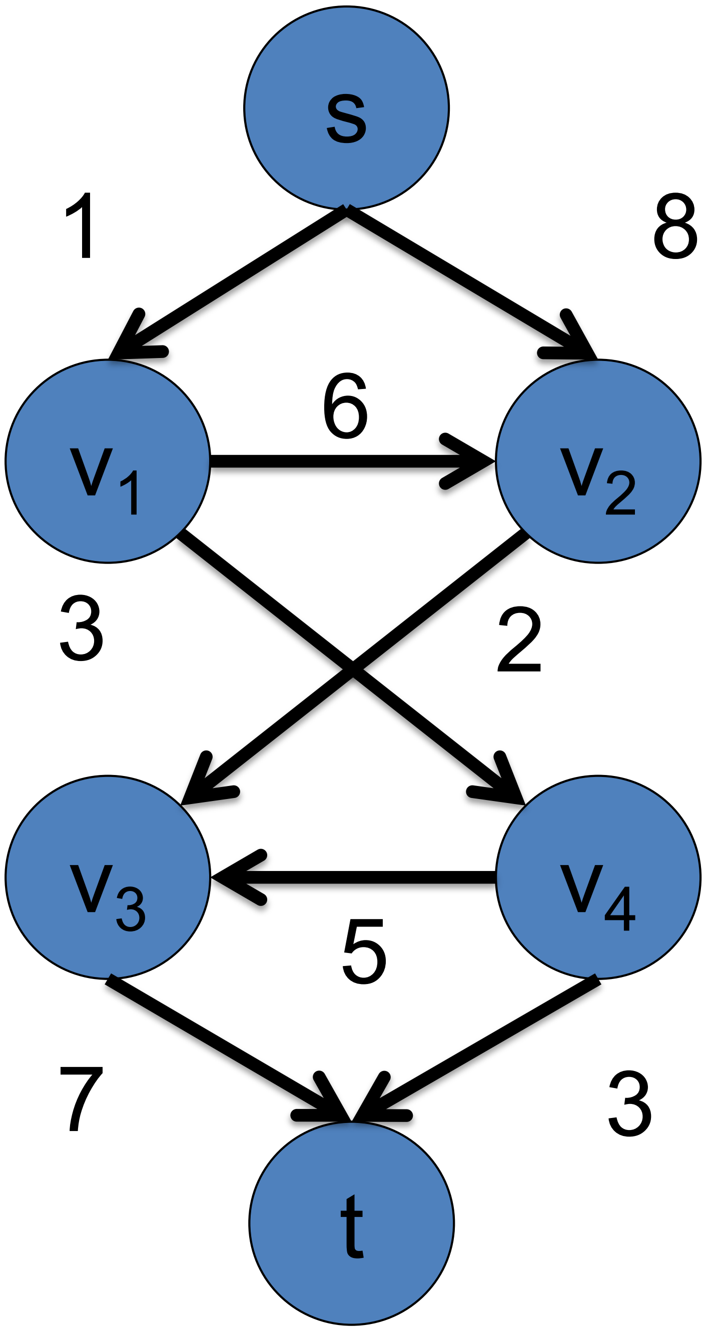
Flow is non-negative

For all vertex except s,t

Incoming flow

= Outgoing flow

s-t Flow



Function flow: A → R

flow(a) ≤ c(a)

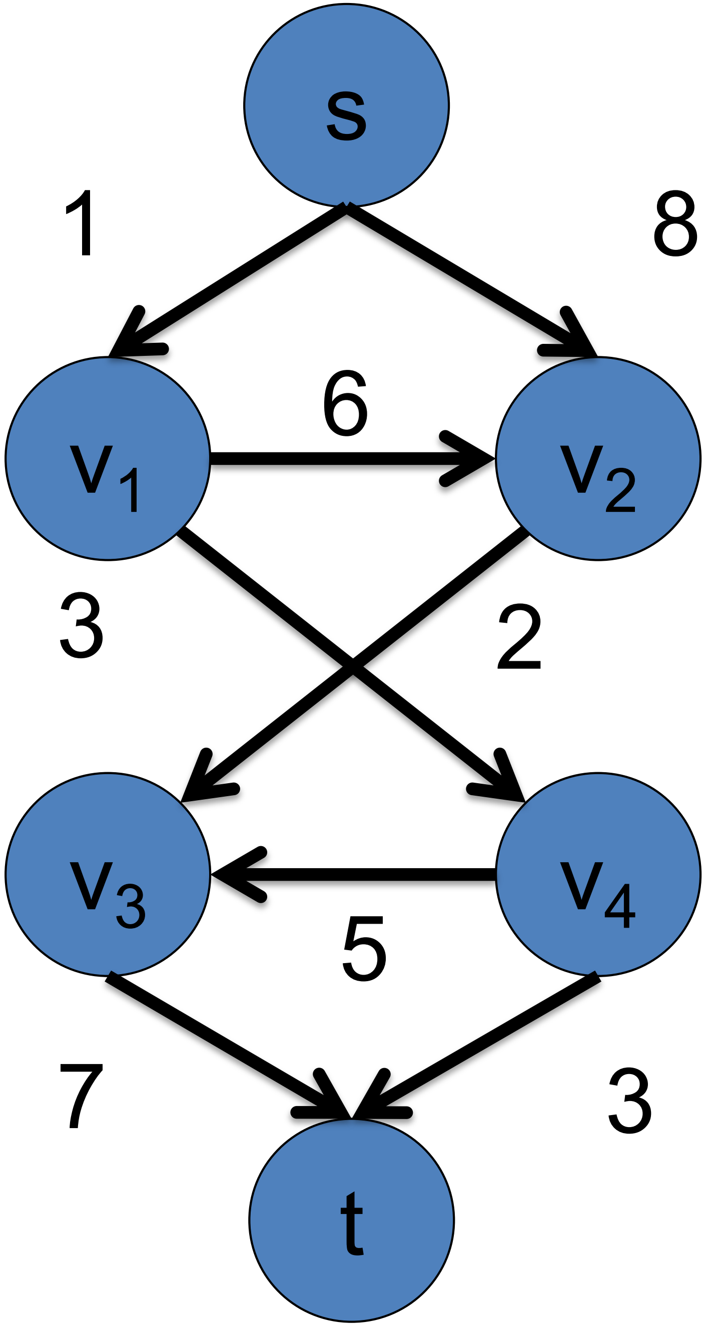
flow(a) ≥ 0

For all vertex except s,t

Incoming flow

= Outgoing flow

s-t Flow



Function flow: A → R

flow(a) ≤ c(a)

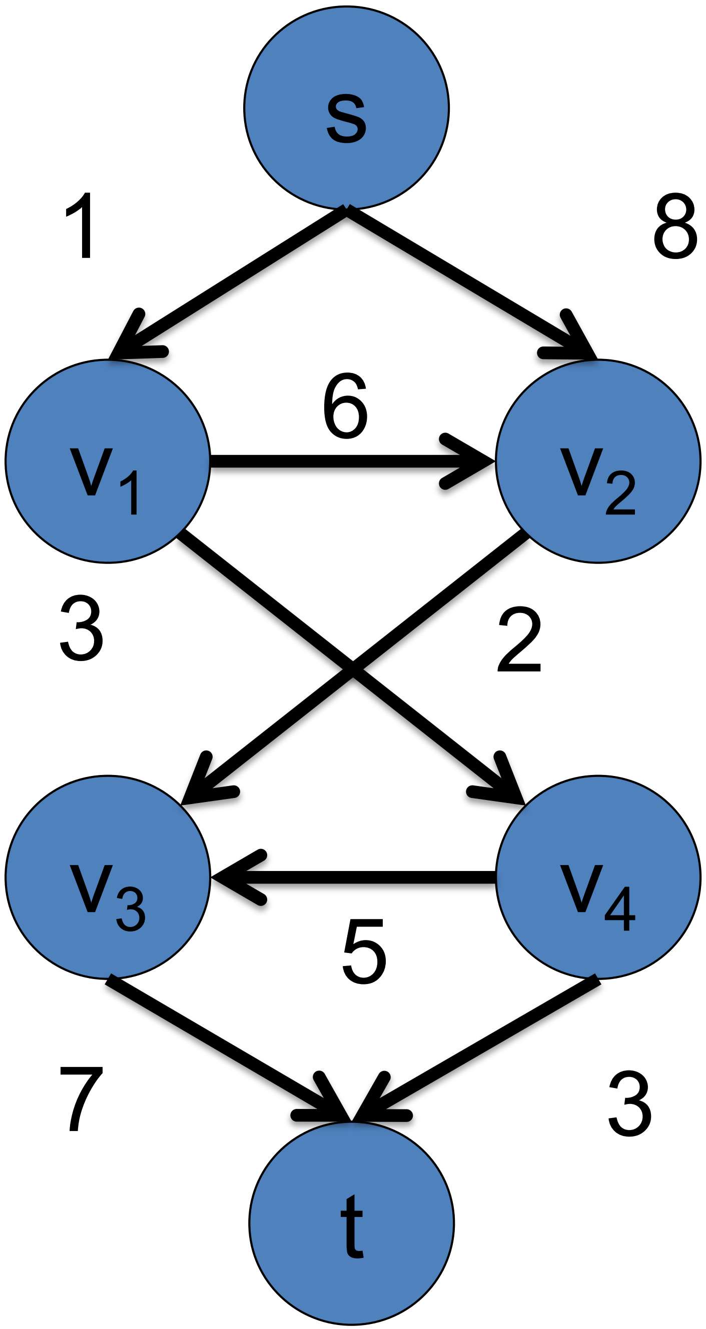
flow(a) ≥ 0

For all v ∈ V \ {s,t}

Incoming flow

= Outgoing flow

s-t Flow



Function flow: A ➔ R

flow(a) ≤ c(a)

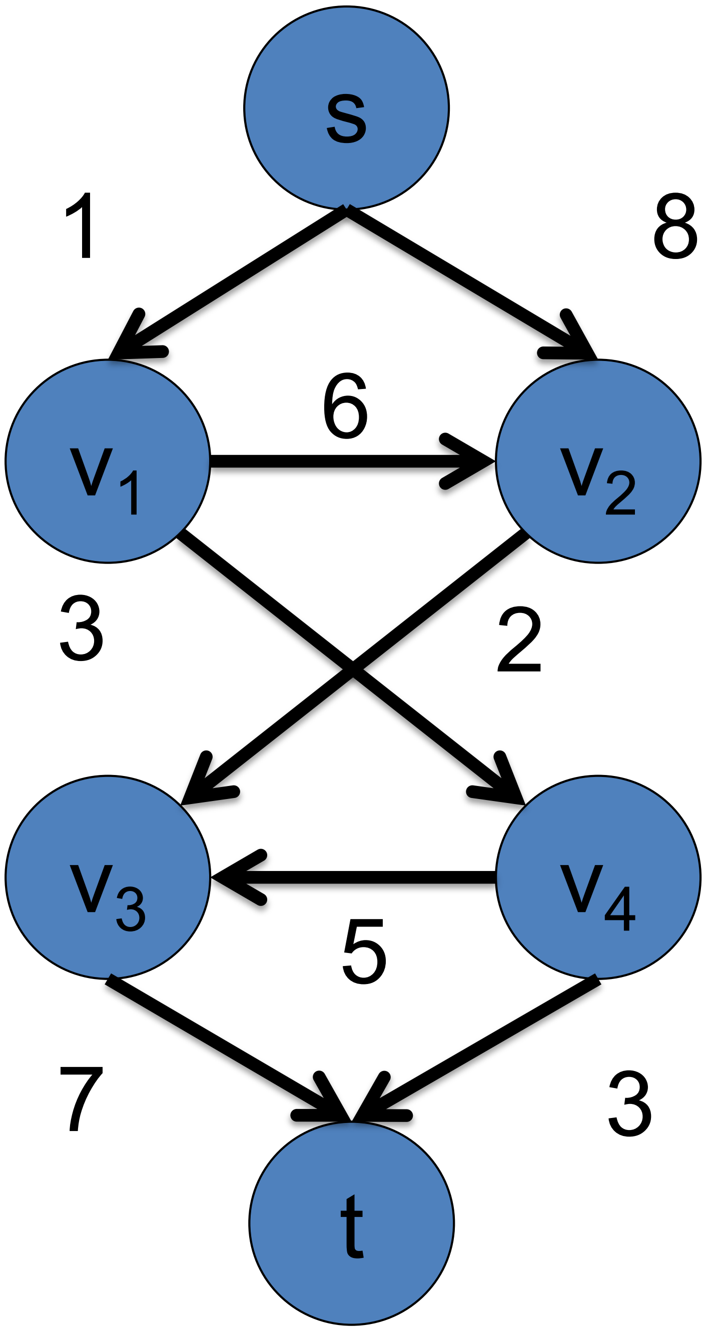
flow(a) ≥ 0

For all $v \in V \setminus \{s,t\}$

$\Sigma_{(u,v)\in A}$ flow((u,v))

= Outgoing flow

s-t Flow



Function flow: A ➔ R

flow(a) $\leq$ c(a)

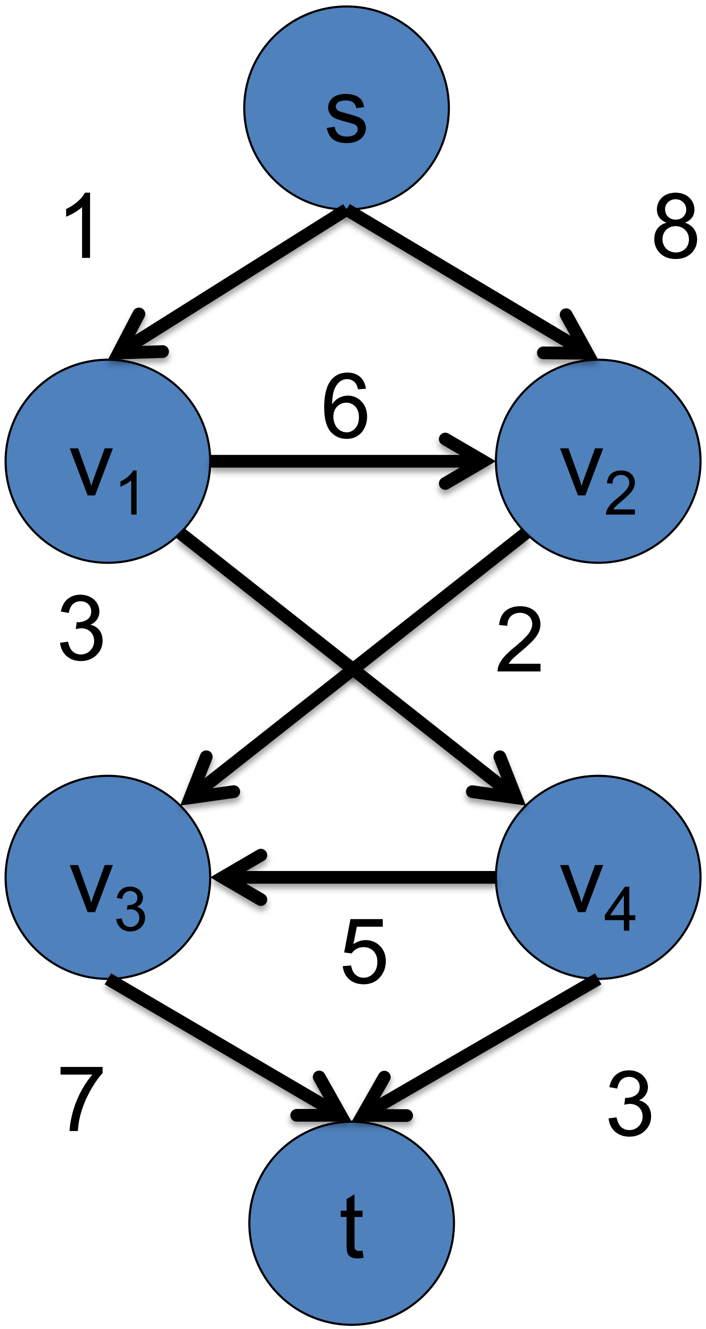
flow(a) $\geq$ 0

For all $v \in V \setminus \{s,t\}$

$$\Sigma_{(u,v)\in A} \text{flow}((u,v))$$

$$= \Sigma_{(v,u)\in A} \text{flow}((v,u))$$

s-t Flow



Function flow: A → R

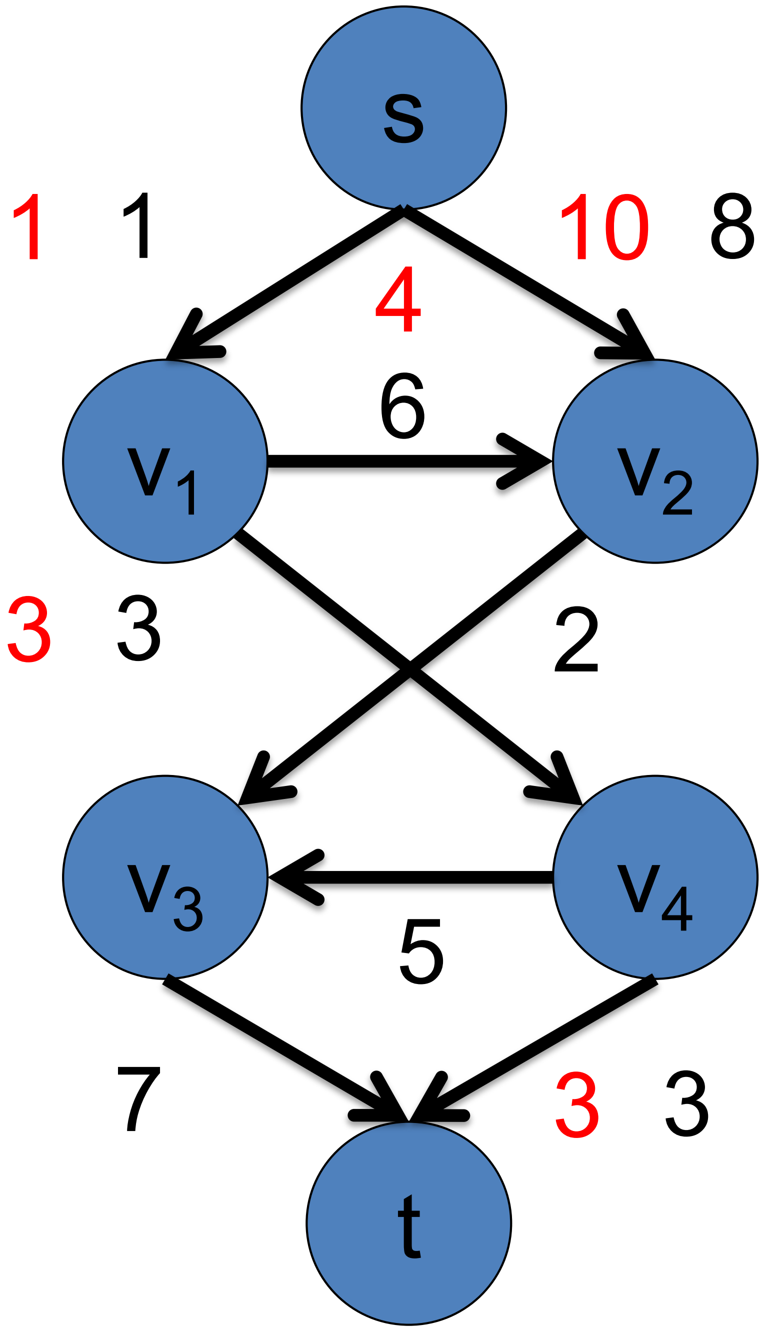
flow(a) ≤ c(a)

flow(a) ≥ 0

For all $v \in V \setminus \{s,t\}$

$E_{flow}(v) = 0$

s-t Flow



Function flow: A ➔ R

flow(a) ≤ c(a)

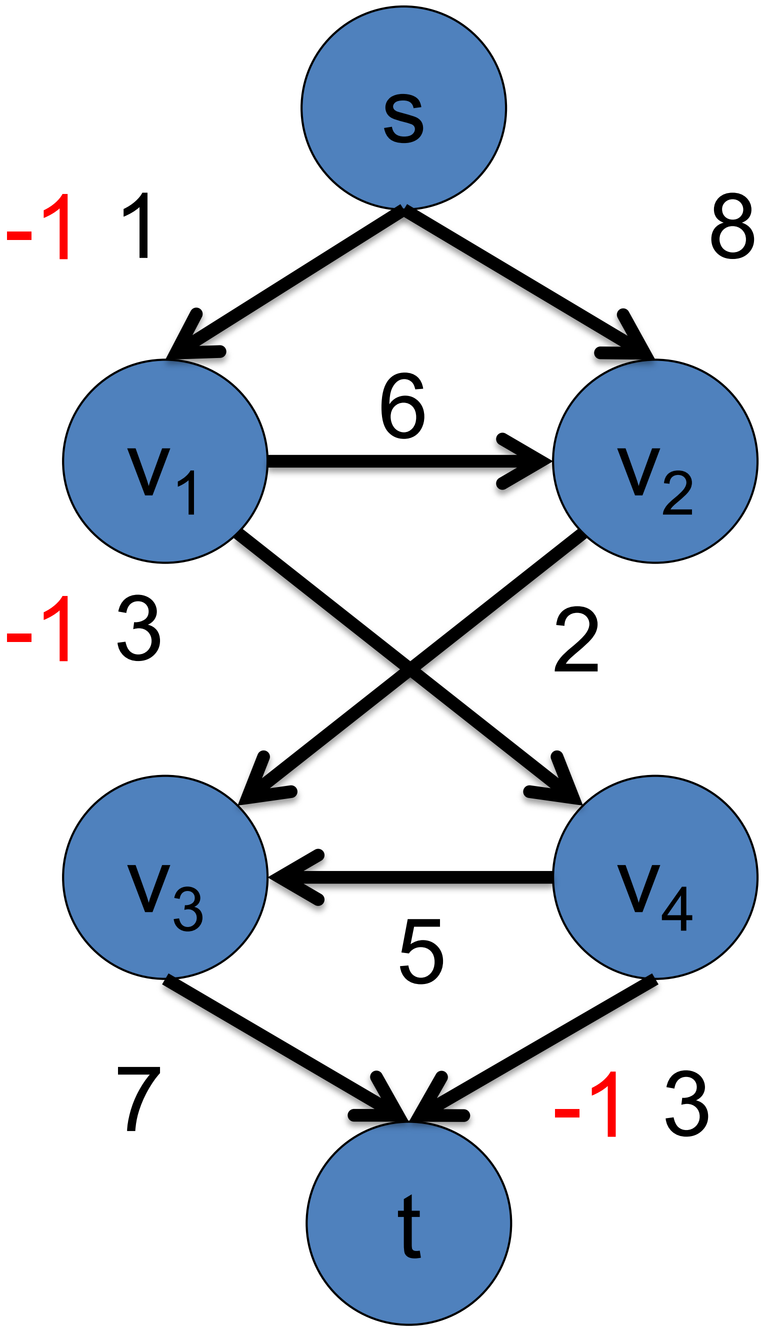
flow(a) ≥ 0

For all v ∈ V \ {s,t}

$E_{flow}(v) = 0$

X

s-t Flow



Function flow: A → R

flow(a) ≤ c(a)

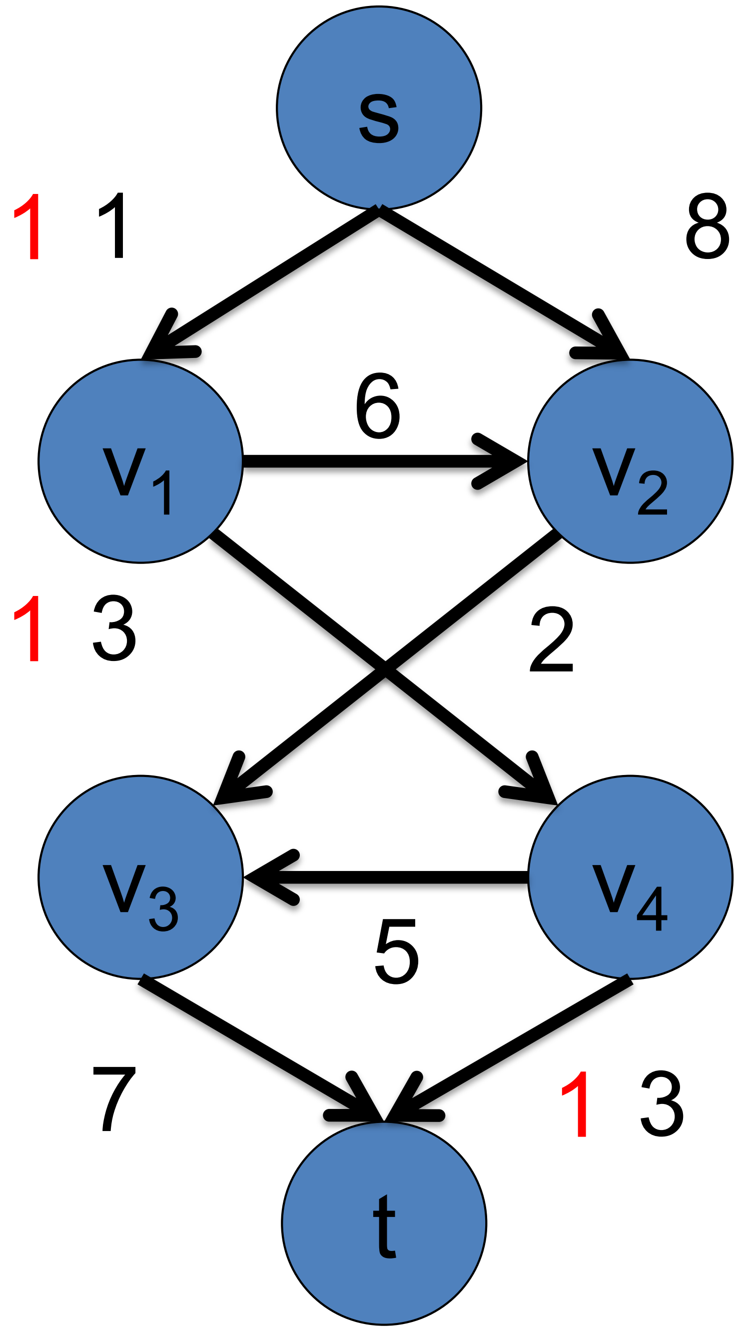
flow(a) ≥ 0

For all v ∈ V \ {s,t}

$E_{flow}(v) = 0$

X

s-t Flow



Function flow: A → R

flow(a) ≤ c(a)

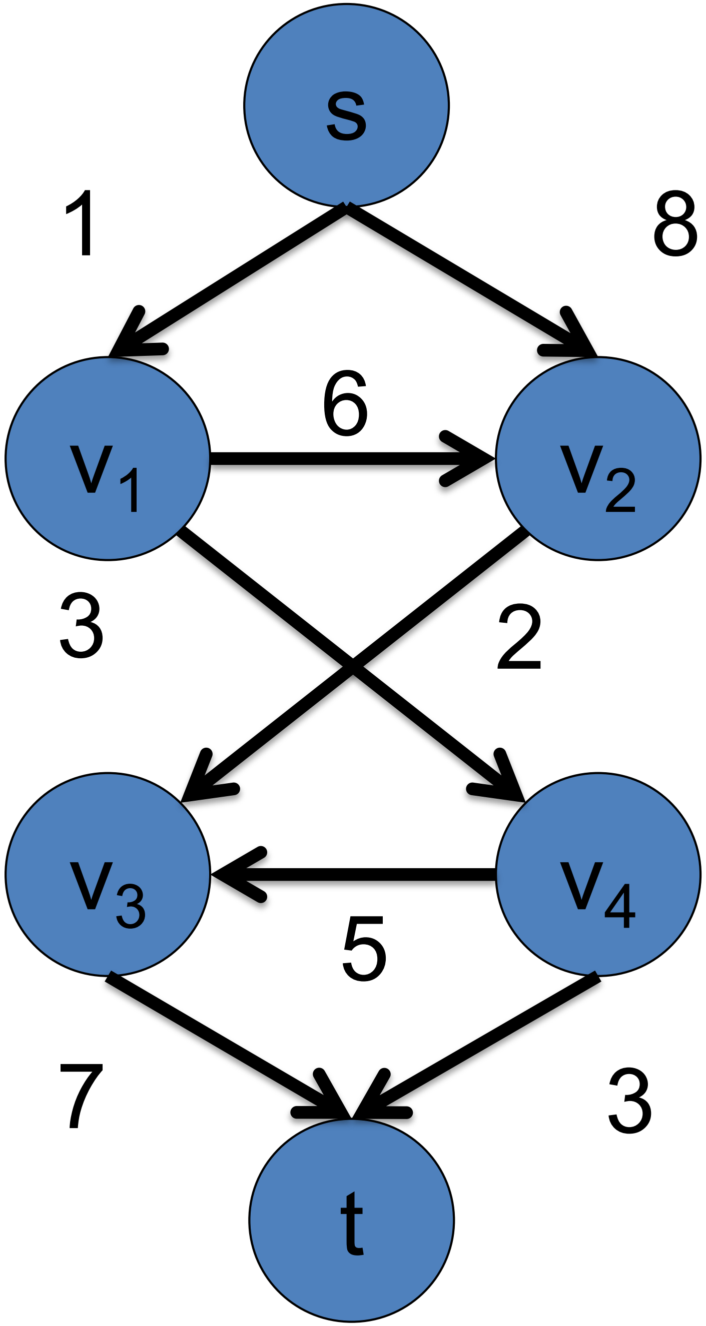
flow(a) ≥ 0

For all $v \in V \setminus \{s,t\}$

$E_{flow}(v) = 0$

✓

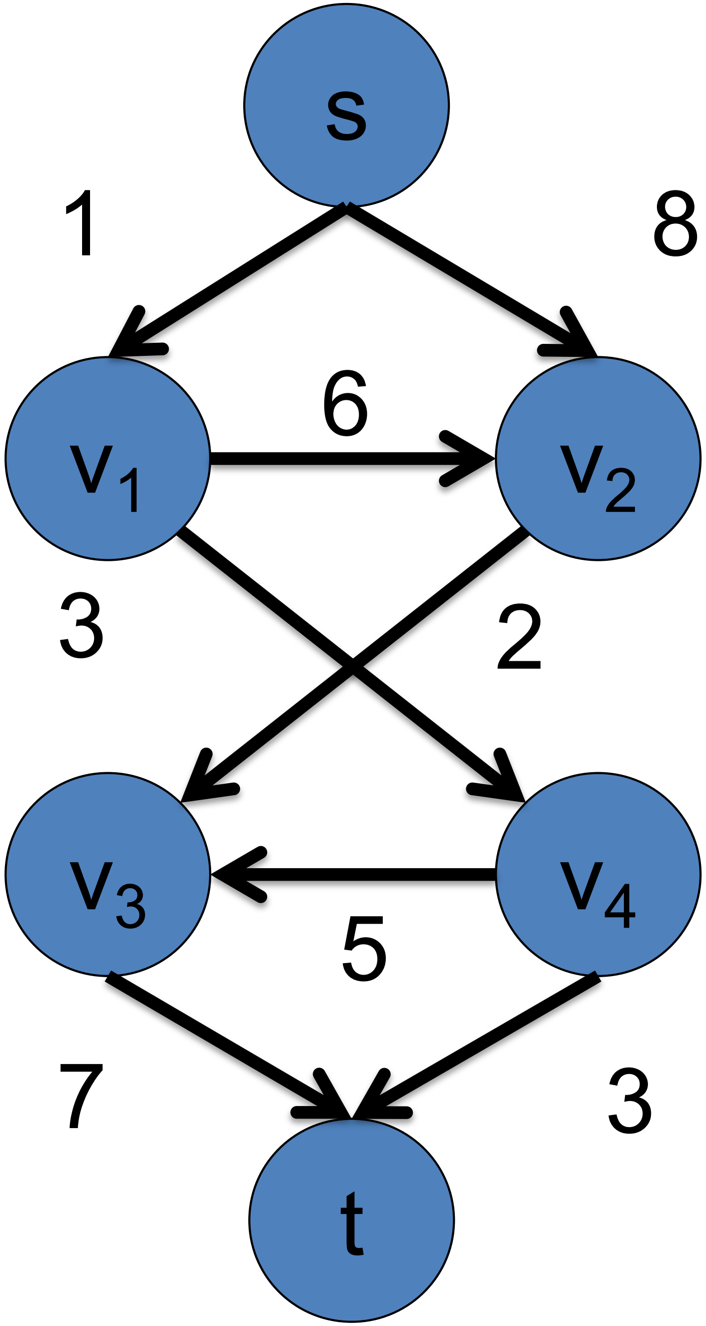
Value of s-t Flow



Outgoing flow of s

- Incoming flow of s

Value of s-t Flow

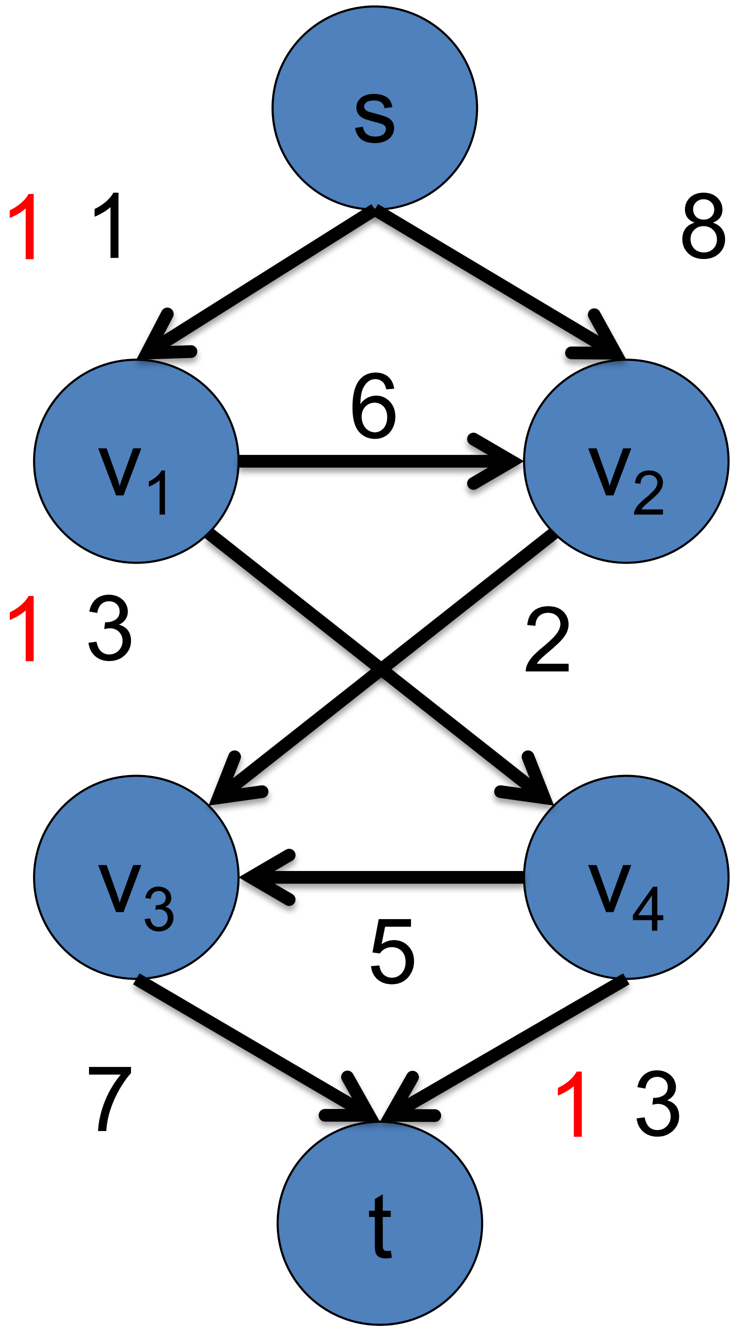


$-E_{flow}(s)$ $E_{flow}(t)$

$\Sigma_{(s,v)\in A}$ flow((s,v))

$- \Sigma_{(u,s)\in A}$ flow((u,s))

Value of s-t Flow



$-E_{flow}(s)$ $\quad$ $E_{flow}(t)$

$$\Sigma_{(s,v)\in A}\, flow((s,v))$$

$$-\ \Sigma_{(u,s)\in A}\, flow((u,s))$$

Value = 1

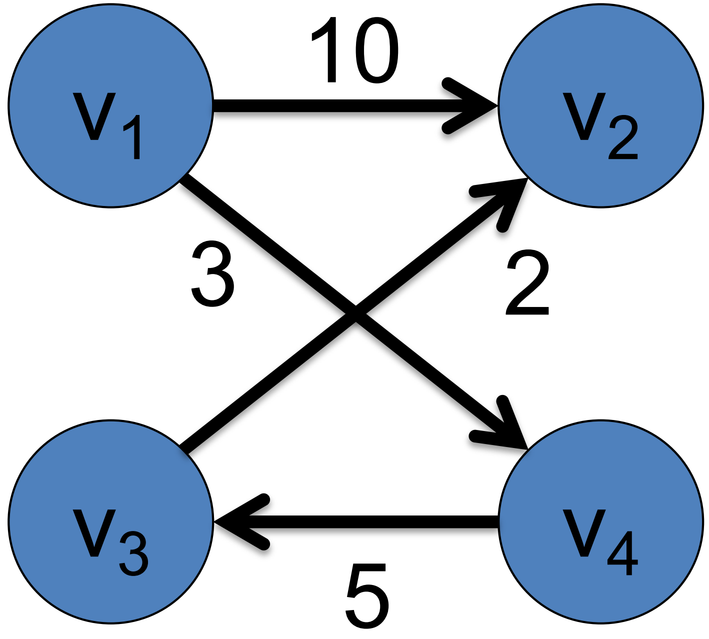
Outline

- Preliminaries
  - Functions and Excess Functions
  - s-t Flow
  - **s-t Cut**
  - Flows vs. Cuts
- Maximum Flow
- Algorithms
- Energy minimization with max flow/min cut

Cut

D = (V, A)

Let U be a subset of V

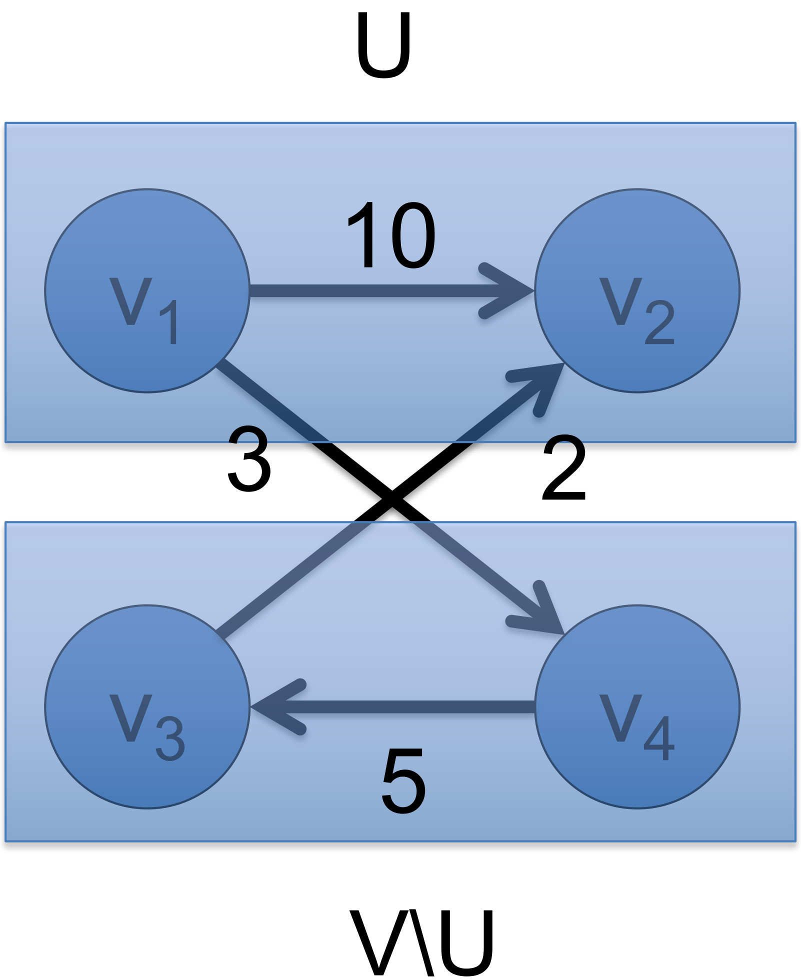


C is a set of arcs such that

- $(u,v) \in A$
- $u \in U$
- $v \in V \backslash U$

C is a cut in the digraph D

Cut



D = (V, A)

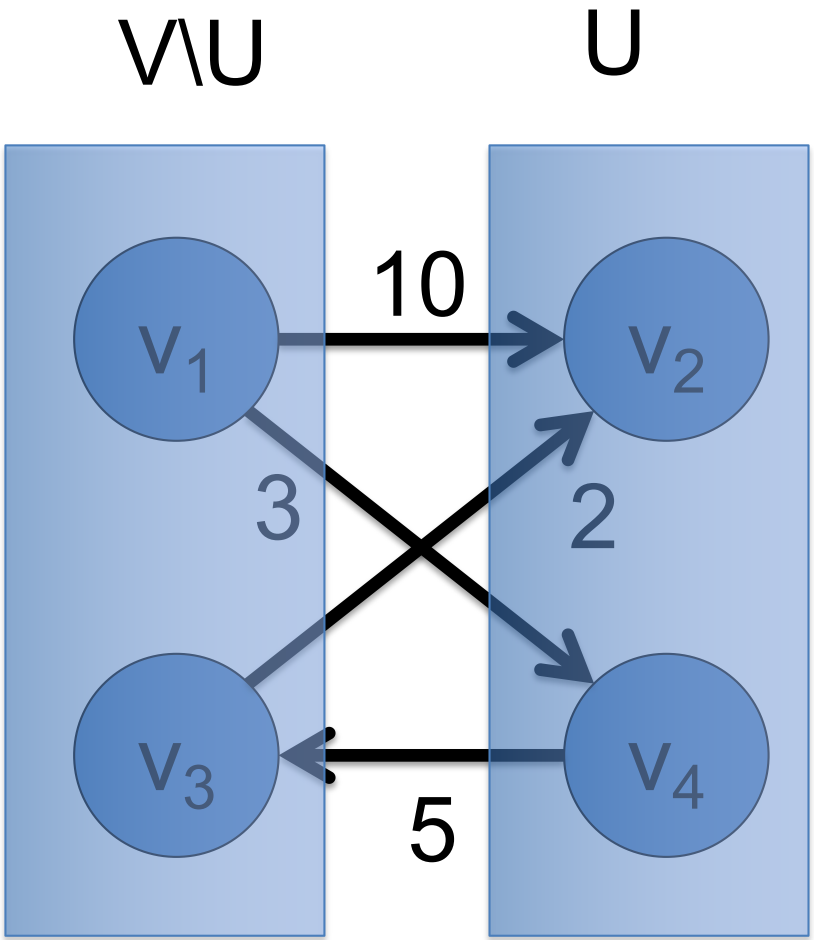
What is C?

$\{(v_1,v_2),(v_1,v_4)\}$?

$\{(v_1,v_4),(v_3,v_2)\}$?

✓ $\{(v_1,v_4)\}$?

Cut



D = (V, A)

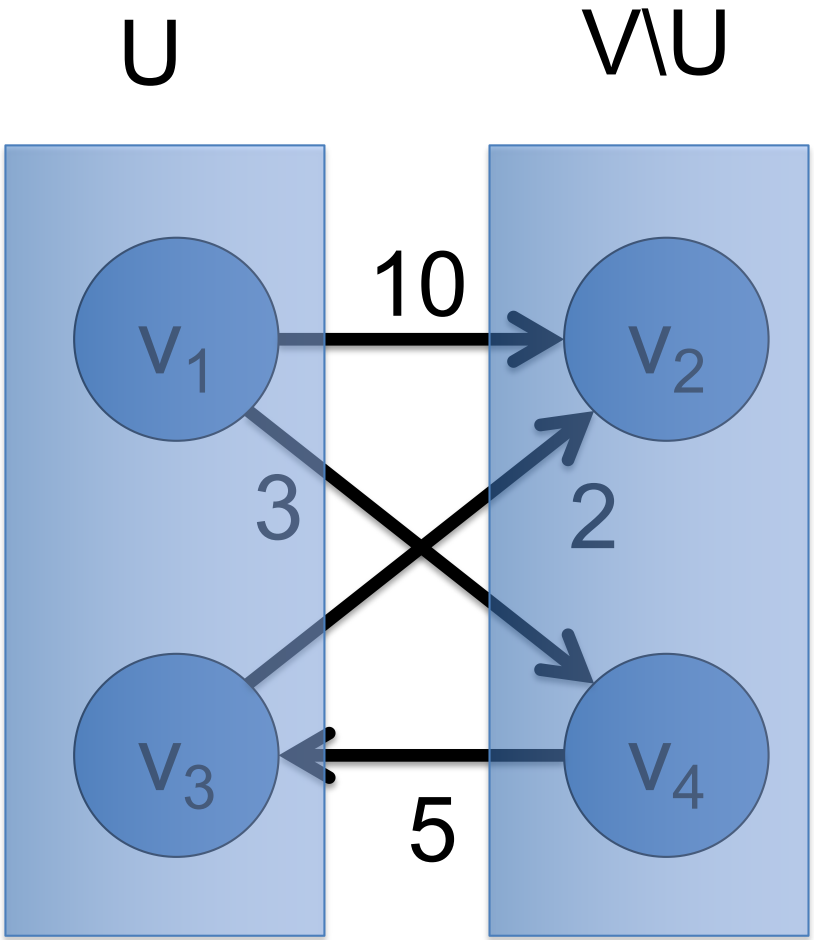
What is C?

$\{(v_1,v_2),(v_1,v_4),(v_3,v_2)\}$?

✓ $\{(v_4,v_3)\}$?

$\{(v_1,v_4),(v_3,v_2)\}$?

Cut



$D = (V, A)$

What is C?

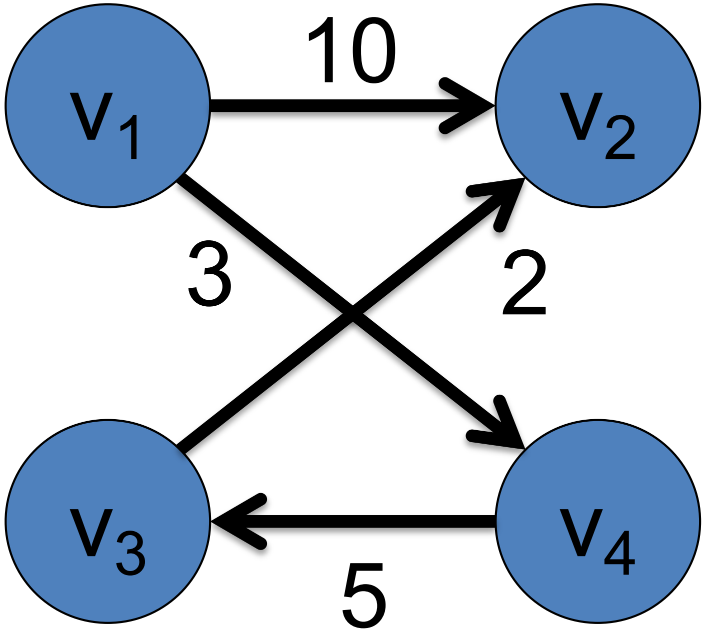
✓ $\{(v_1,v_2),(v_1,v_4),(v_3,v_2)\}$?

$\{(v_3,v_2)\}$?

$\{(v_1,v_4),(v_3,v_2)\}$?

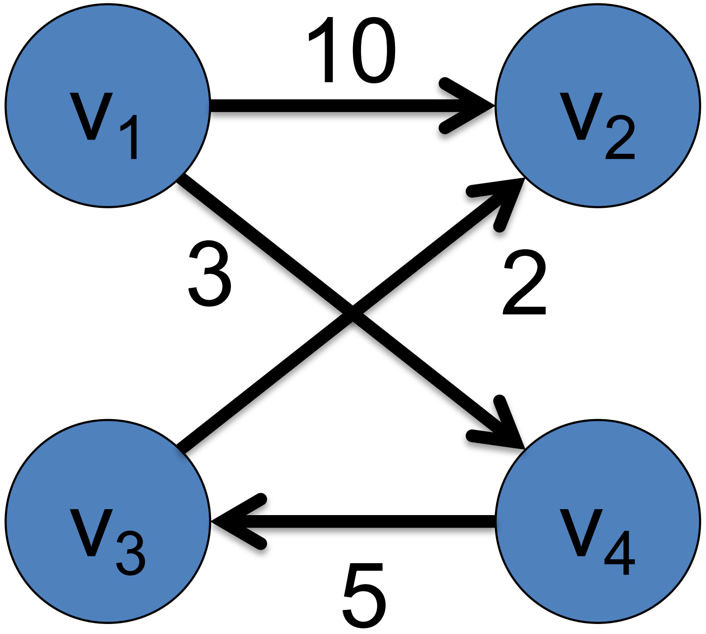
Cut

D = (V, A)



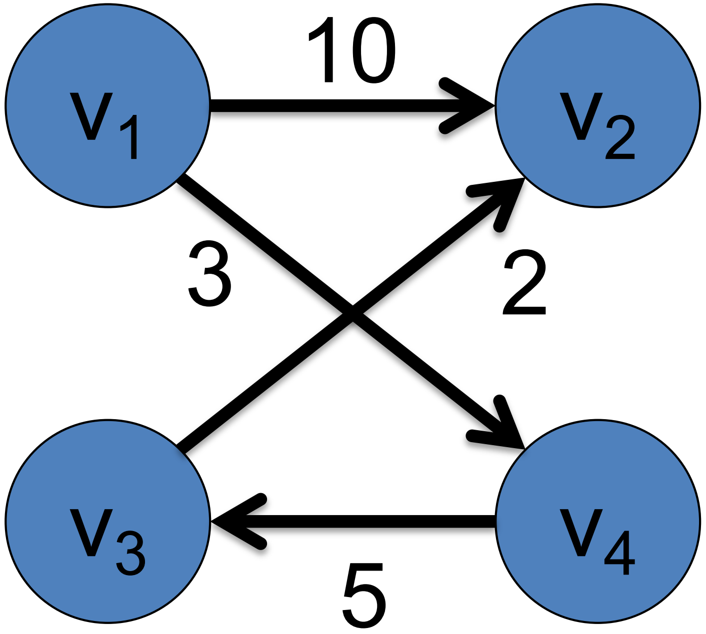
C = out-arcs(U)

Capacity of Cut



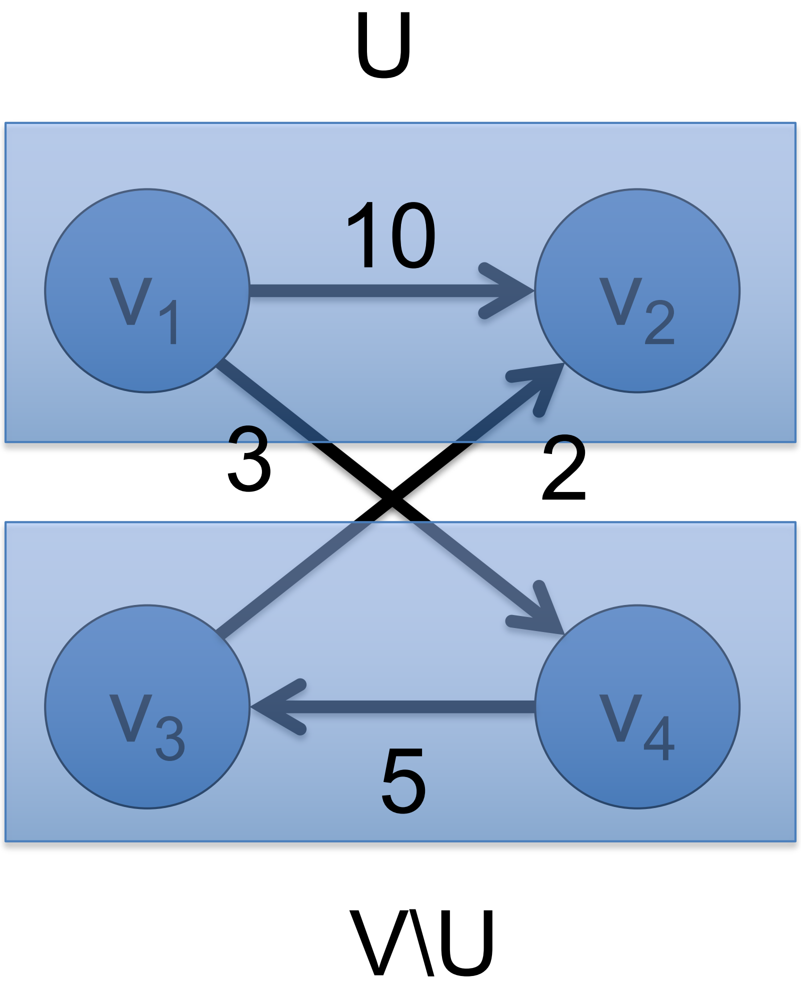
Sum of capacity of all arcs in C

Capacity of Cut



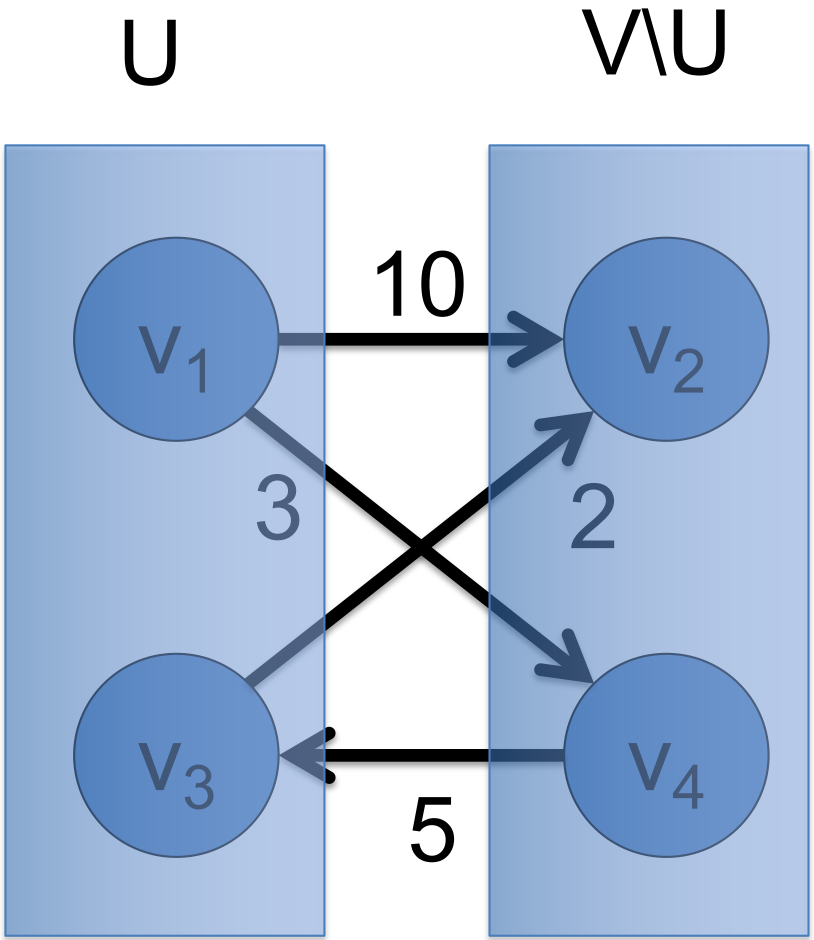
$\Sigma_{a \in C} c(a)$

Capacity of Cut



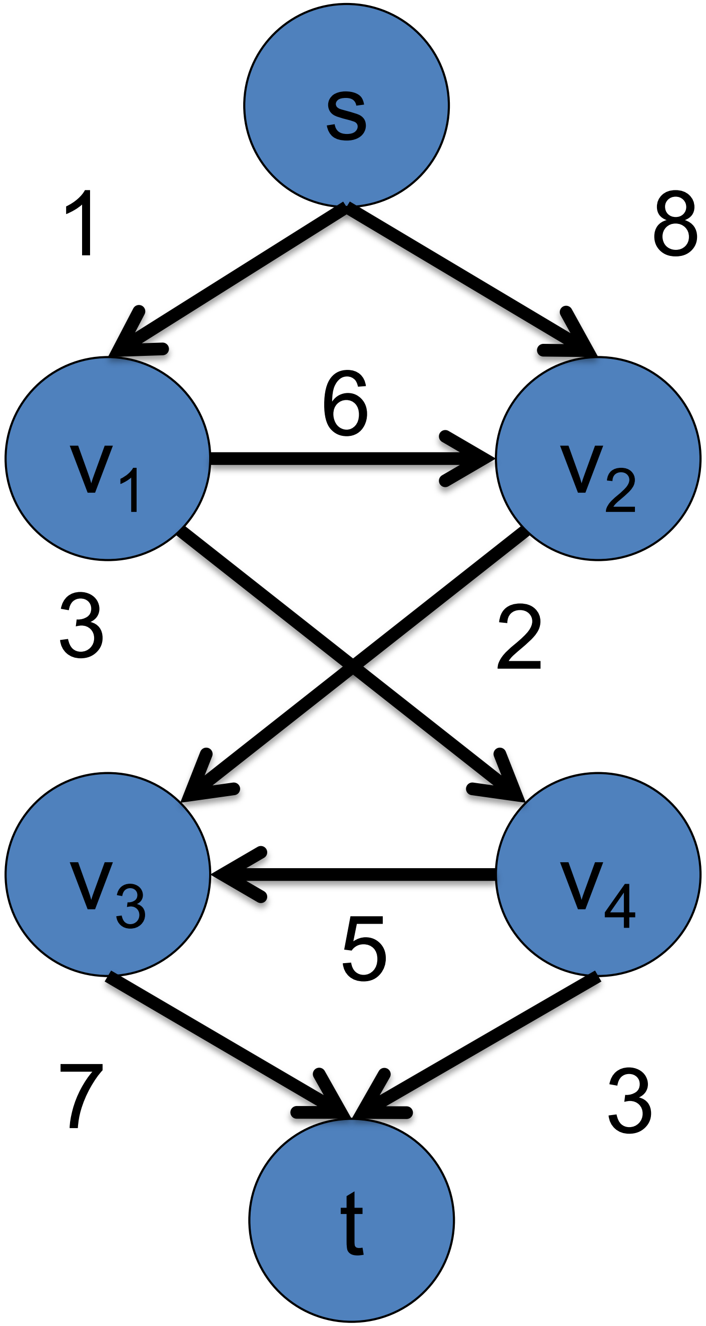
3

Capacity of Cut



15

s-t Cut



D = (V, A)

A source vertex “s”

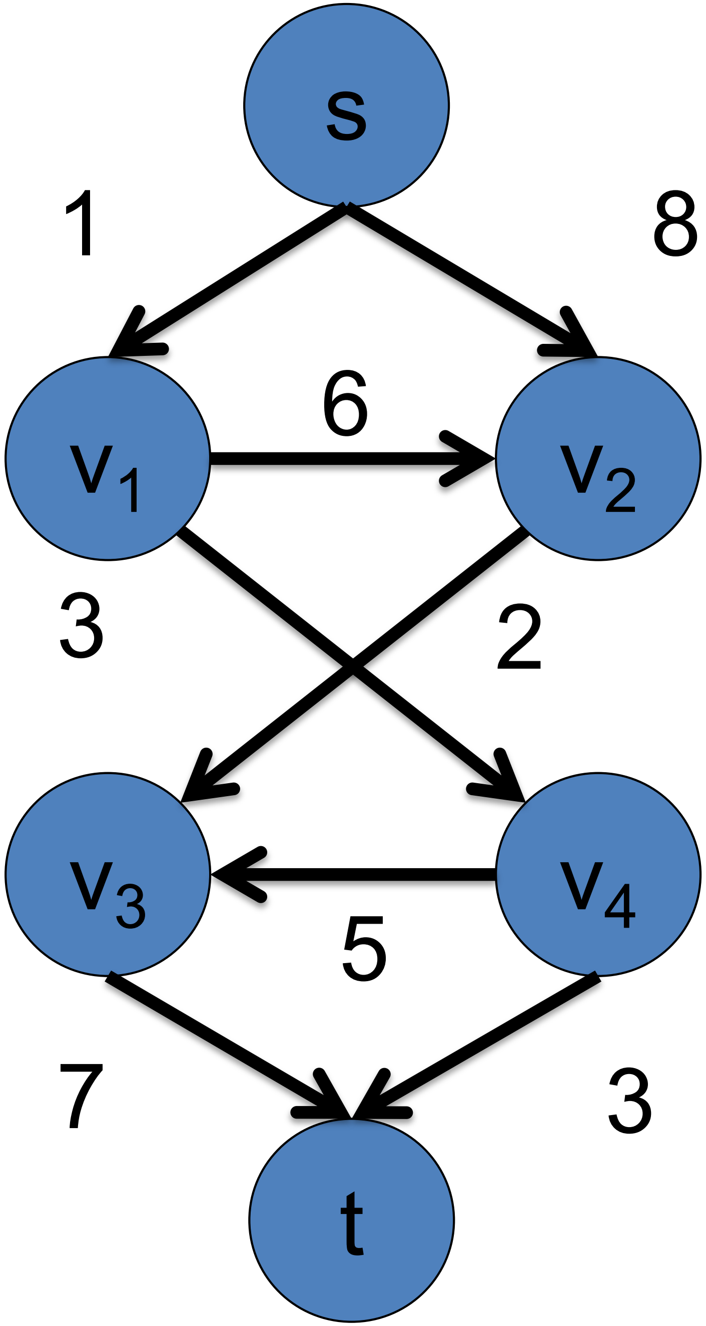
A sink vertex “t”

C is a cut such that

- $s \in U$
- $t \in V \backslash U$

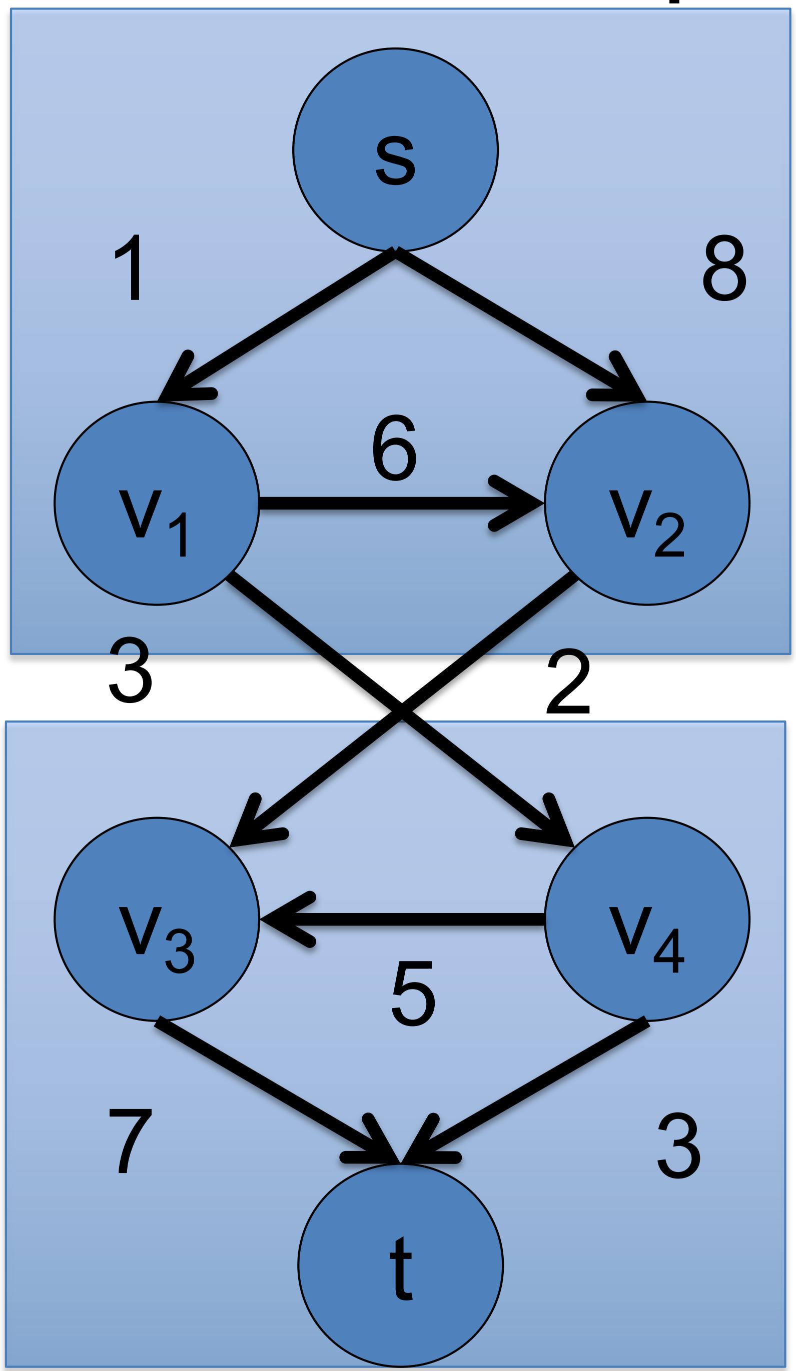
C is an s-t cut

Capacity of s-t Cut



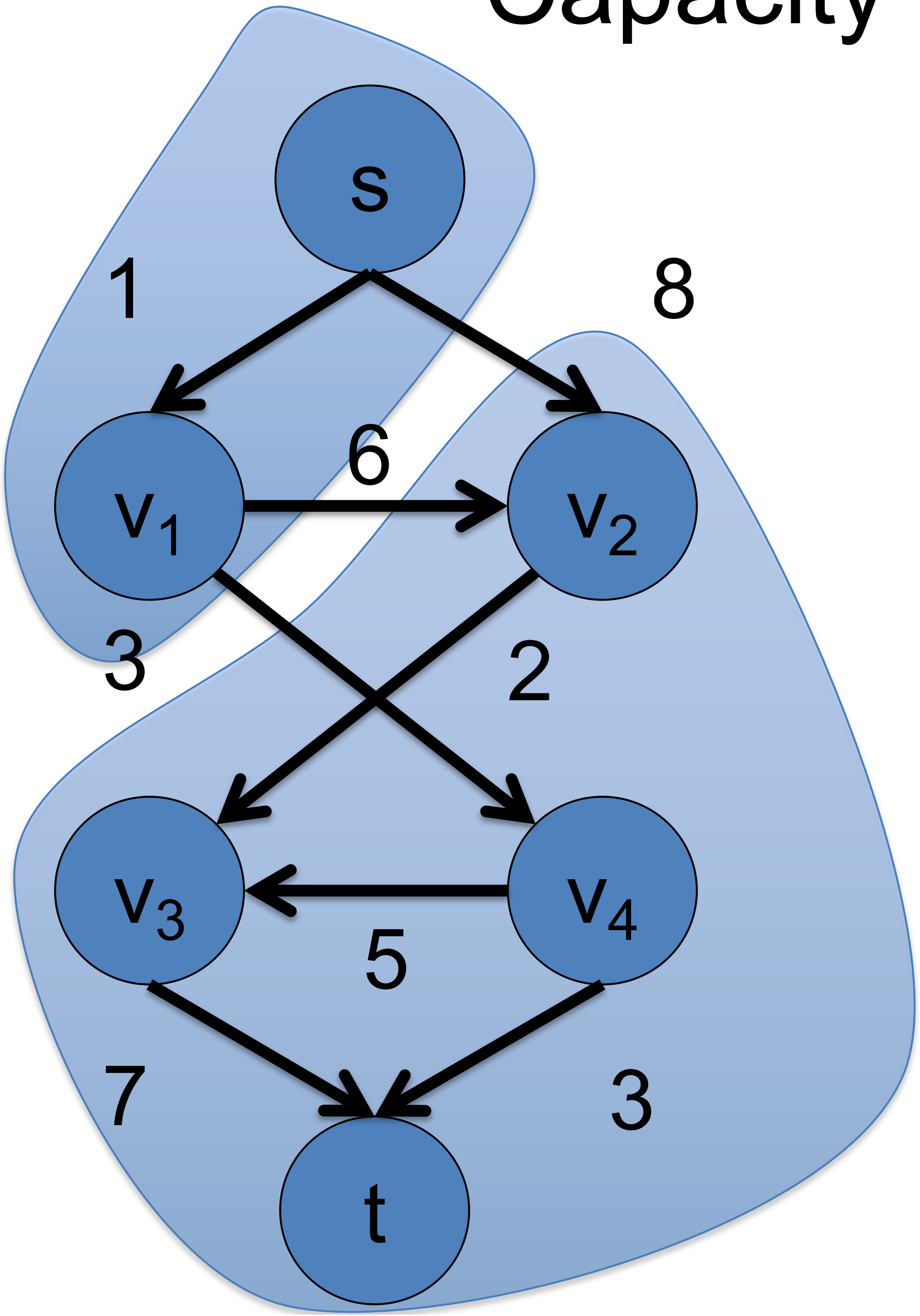
$\Sigma_{a \in C} c(a)$

Capacity of s-t Cut



5

Capacity of s-t Cut

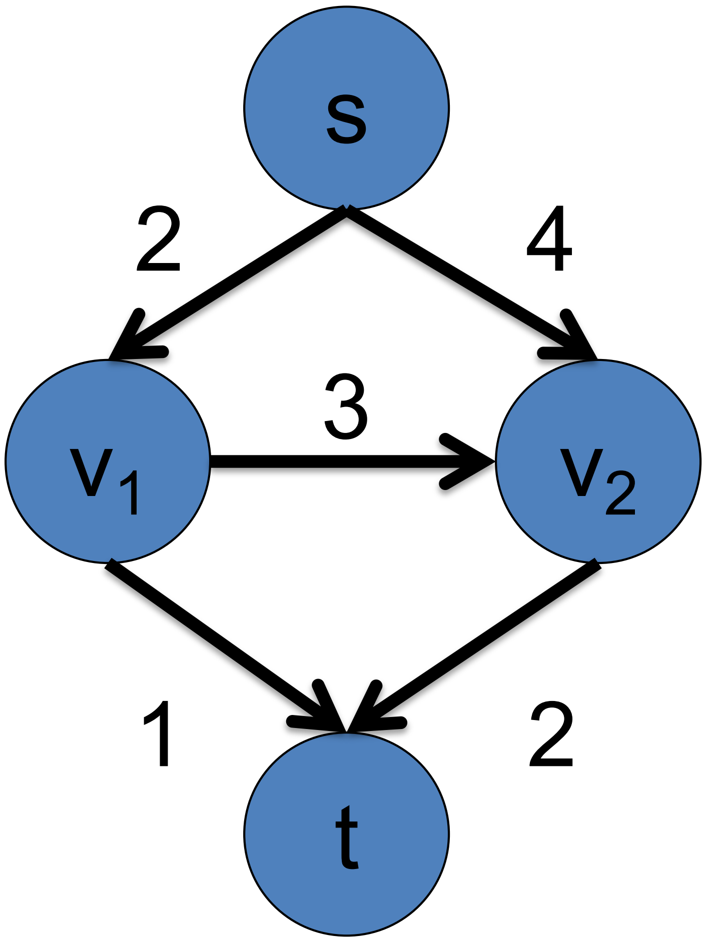


17

Outline

- Preliminaries
- **Maximum Flow**
  - Residual Graph
  - Max-Flow Min-Cut Theorem
- Algorithms
- Energy minimization with max flow/min cut

Maximum Flow Problem

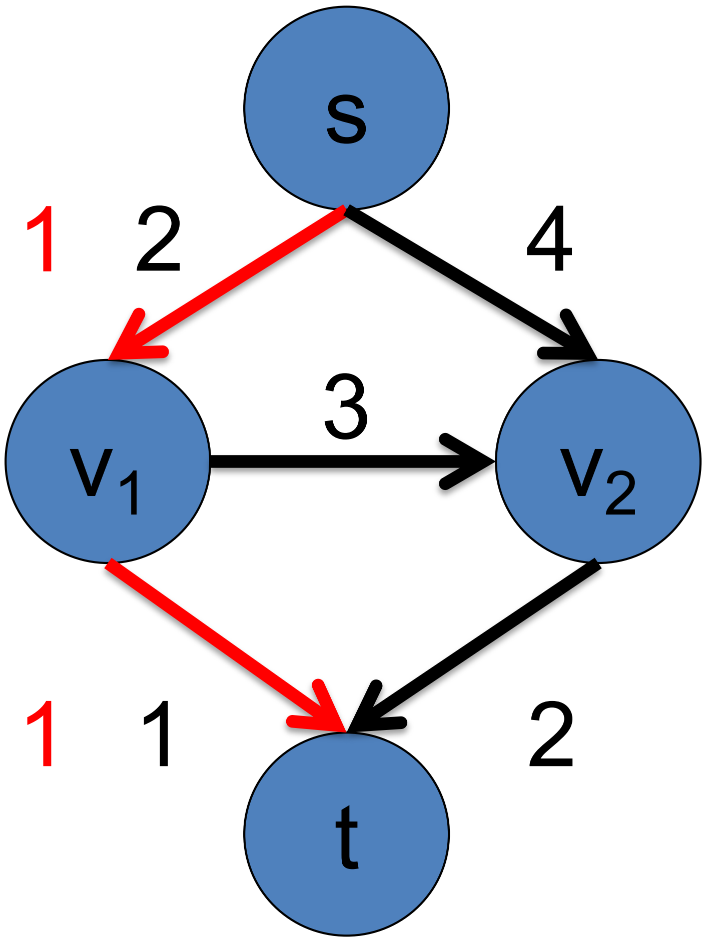


Find the flow with the maximum value !!

$$\Sigma_{(s,v)\in A} \text{flow}((s,v)) - \Sigma_{(u,s)\in A} \text{flow}((u,s))$$

First suggestion to solve this problem !!

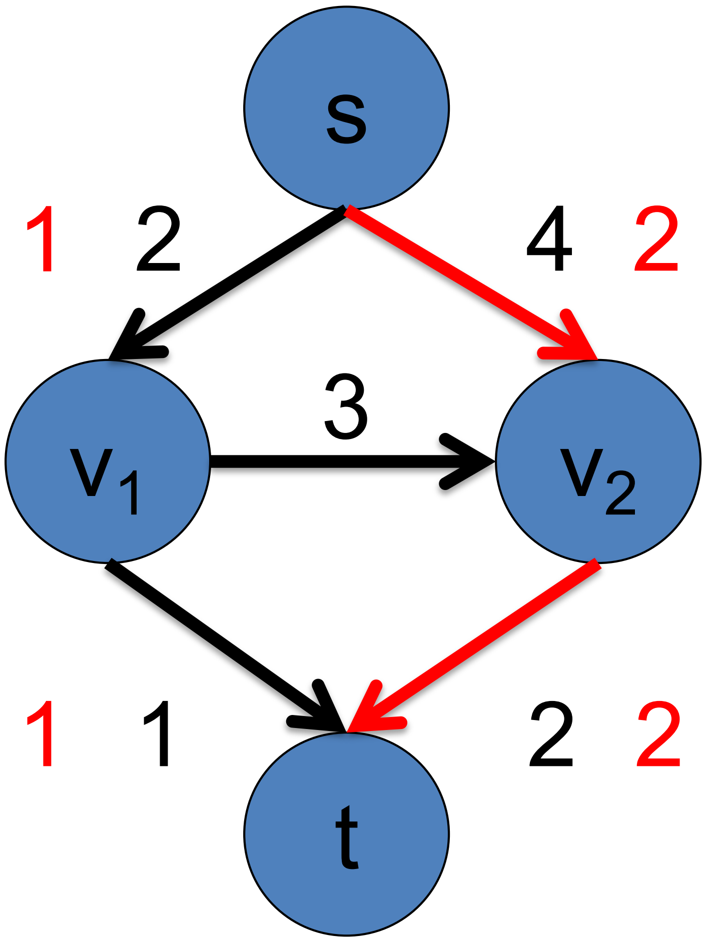
Passing Flow through s-t Paths



Find an s-t path where flow(a) < c(a) for all arcs

Pass maximum allowable flow through the arcs

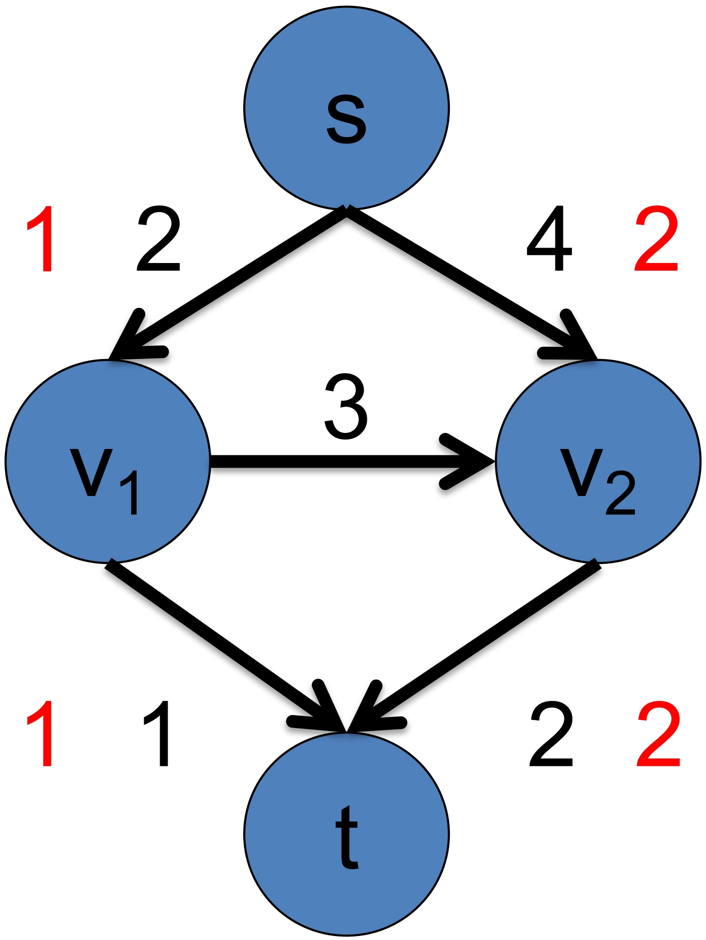
Passing Flow through s-t Paths



Find an s-t path where flow(a) < c(a) for all arcs

Pass maximum allowable flow through the arcs

Passing Flow through s-t Paths

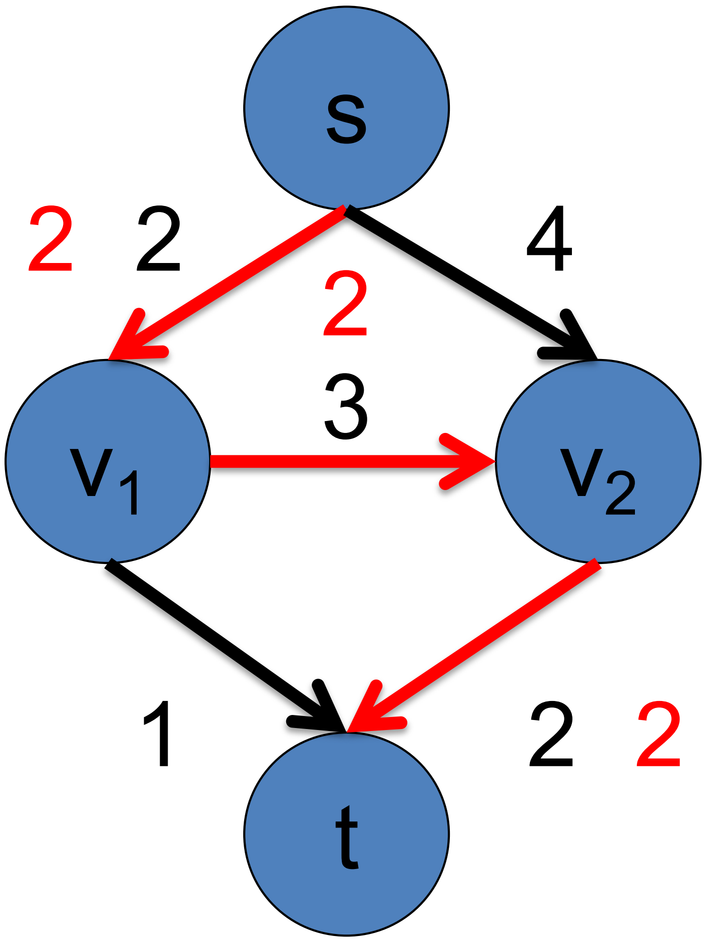


Find an s-t path where flow(a) < c(a) for all arcs

No more paths. Stop.

Will this give us maximum flow? **NO !!!**

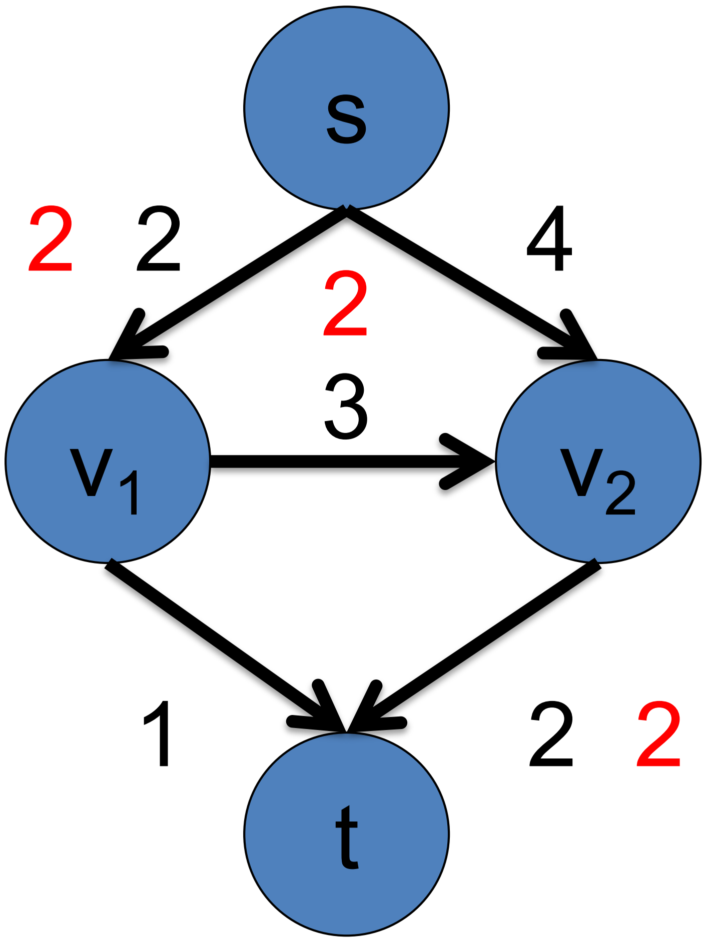
Passing Flow through s-t Paths



Find an s-t path where flow(a) < c(a) for all arcs

Pass maximum allowable flow through the arcs

Passing Flow through s-t Paths



Find an s-t path where flow(a) < c(a) for all arcs

No more paths. Stop.

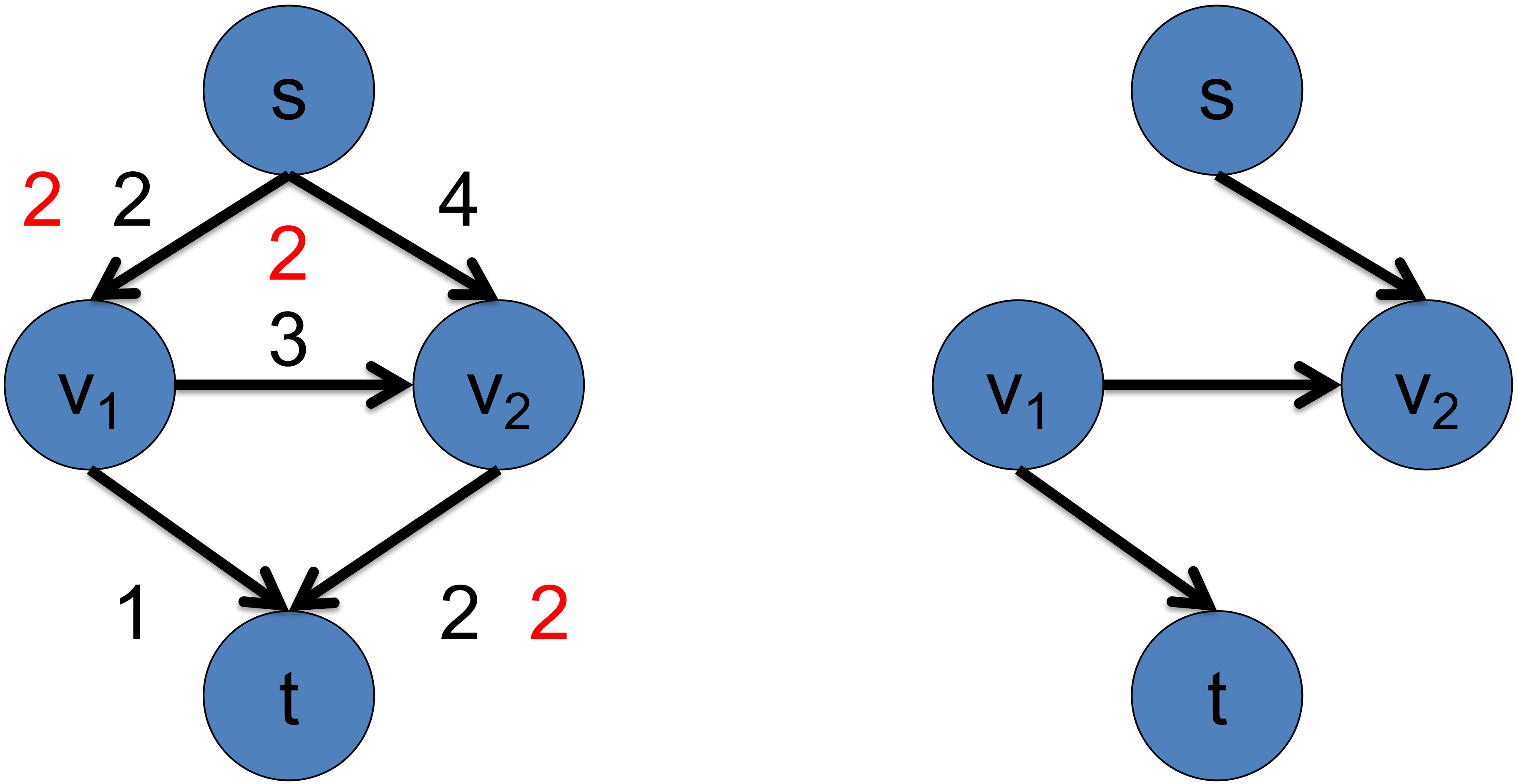
Another method?

Incorrect Answer !!

Outline

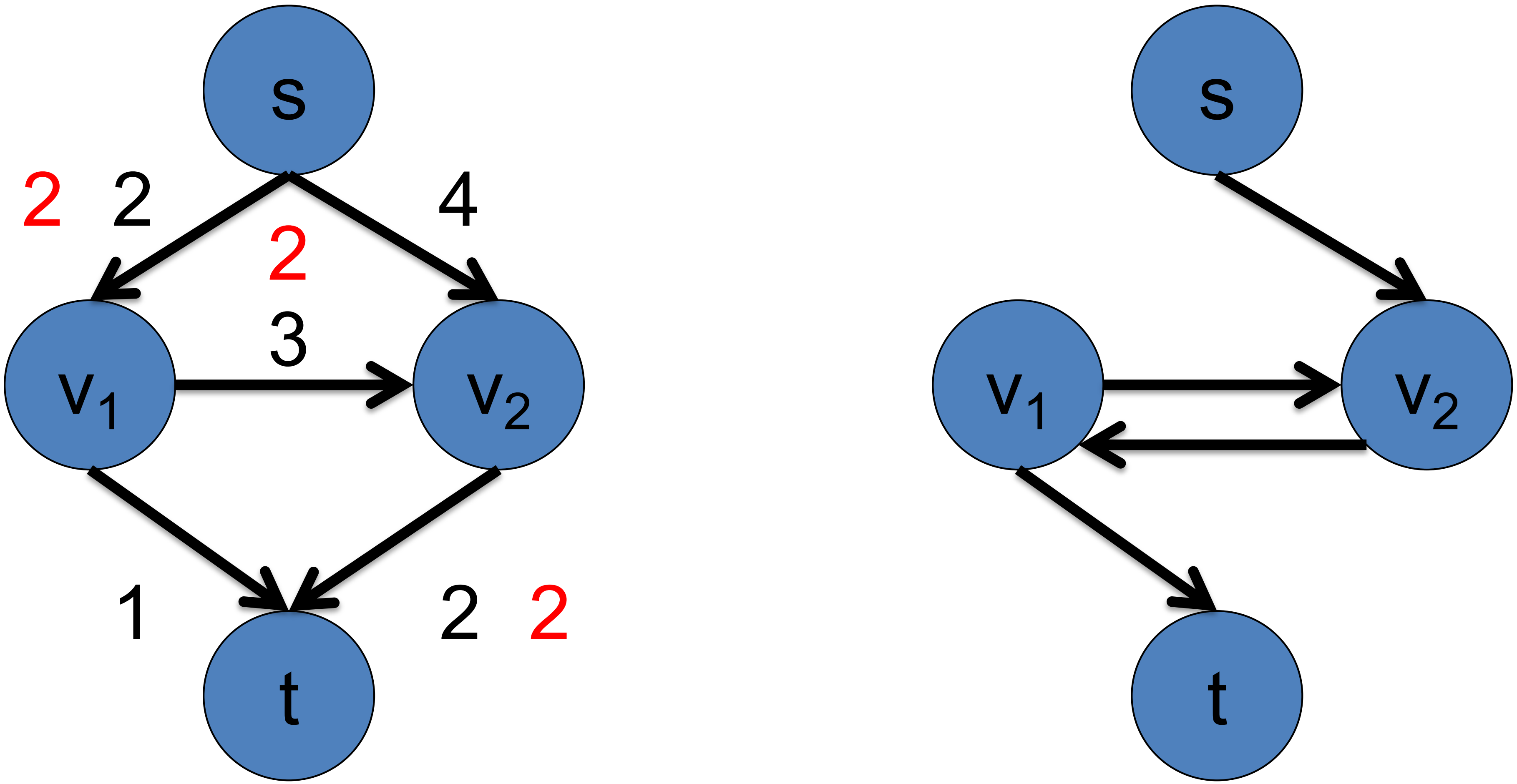
- Preliminaries
- Maximum Flow
  - **Residual Graph**
  - Max-Flow Min-Cut Theorem
- Algorithms
- Energy minimization with max flow/min cut

Residual Graph



Arcs where flow(a) < c(a)

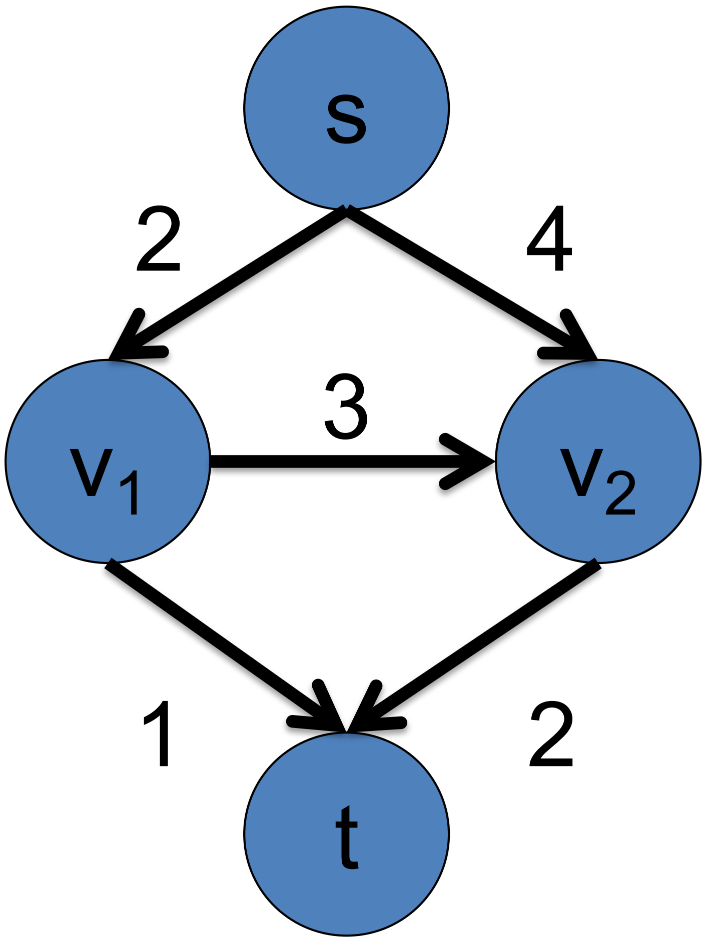
Residual Graph

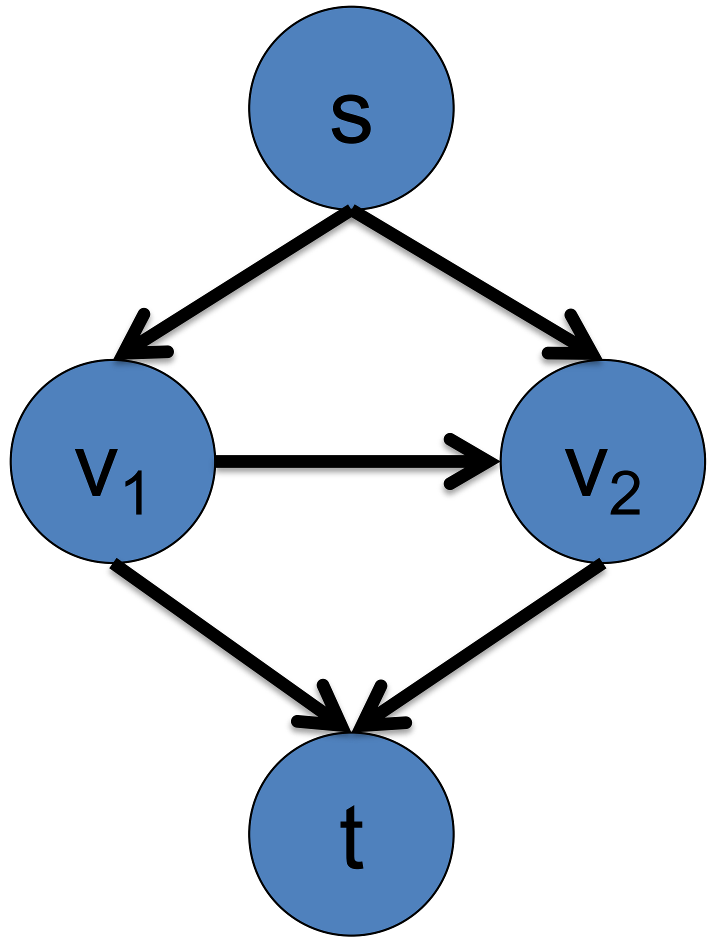


Including arcs to s and from t is not necessary

Inverse of arcs where flow(a) > 0

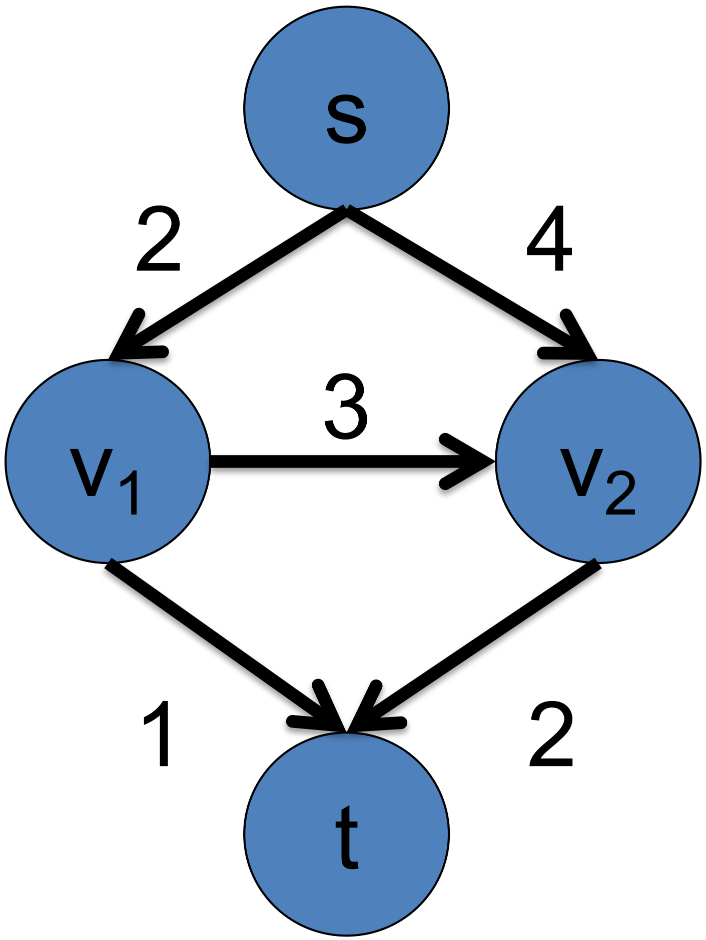
Maximum Flow using Residual Graphs

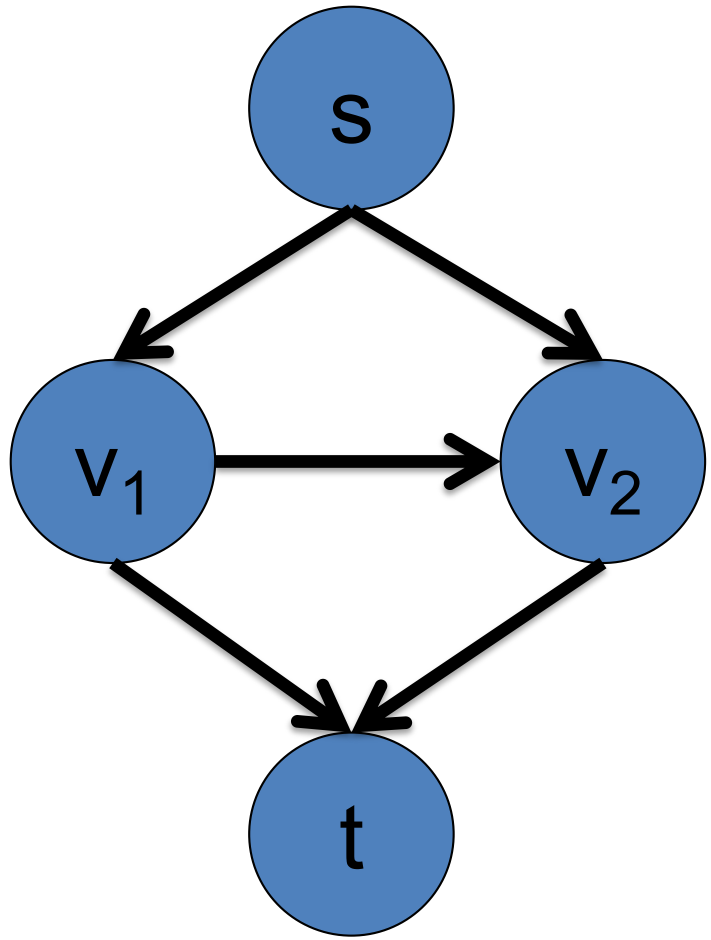




Start with zero flow.

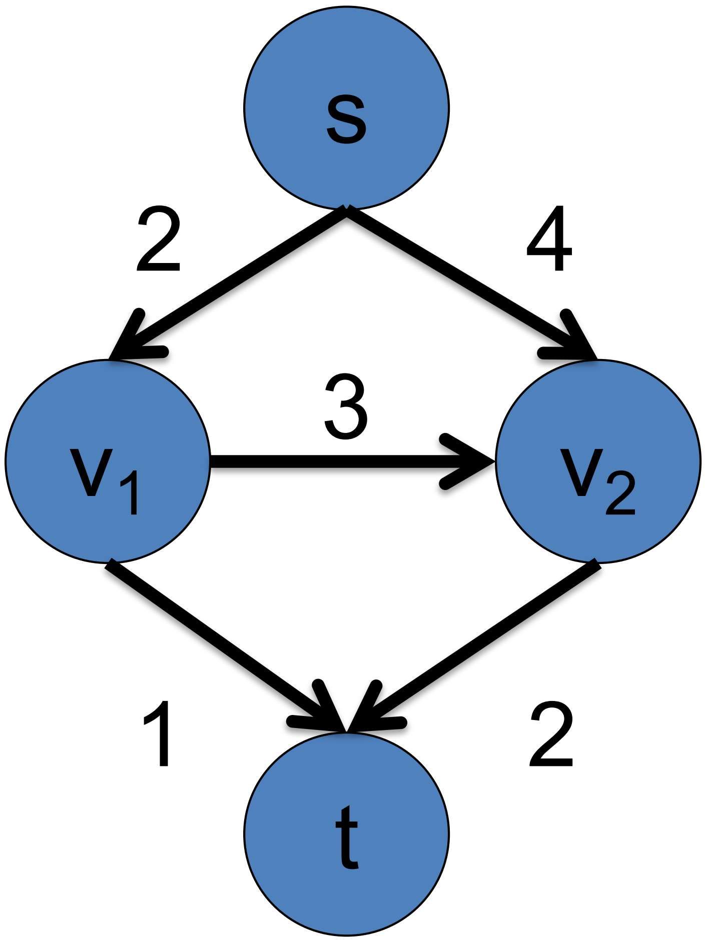
Maximum Flow using Residual Graphs

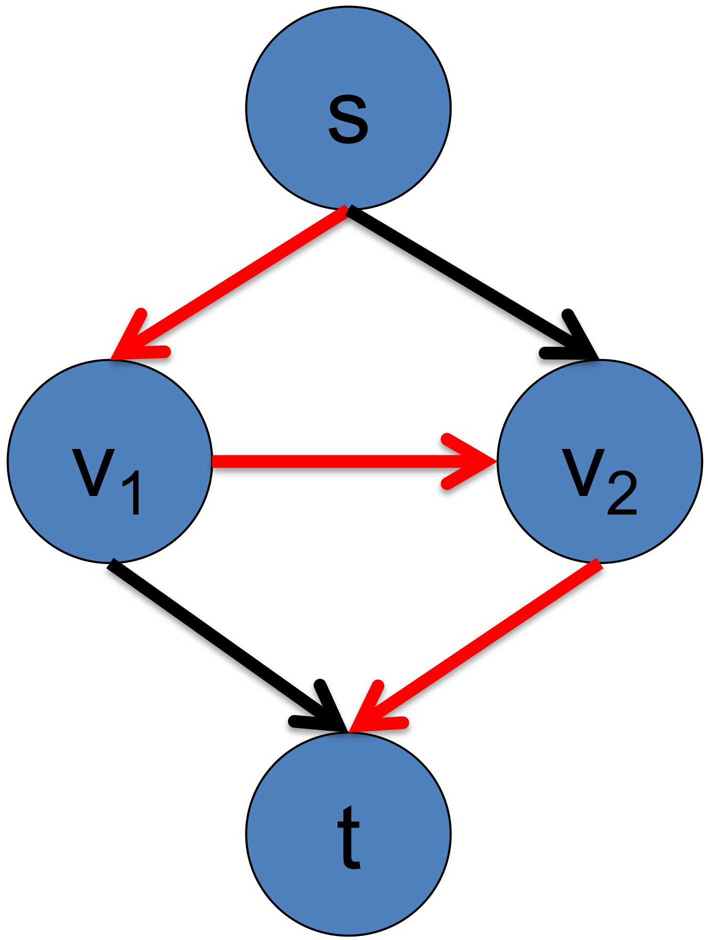




Find an s-t path in the residual graph.

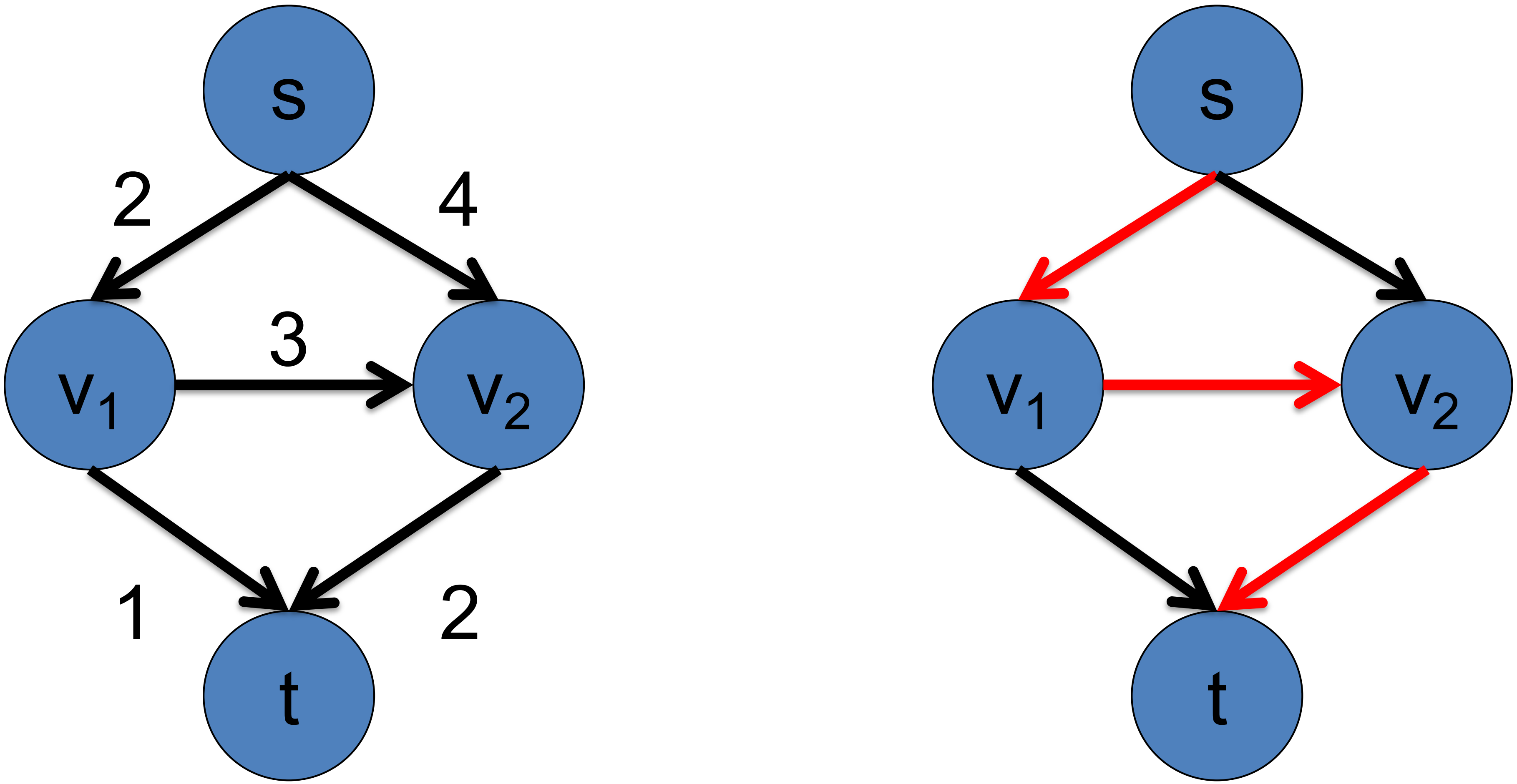
Maximum Flow using Residual Graphs





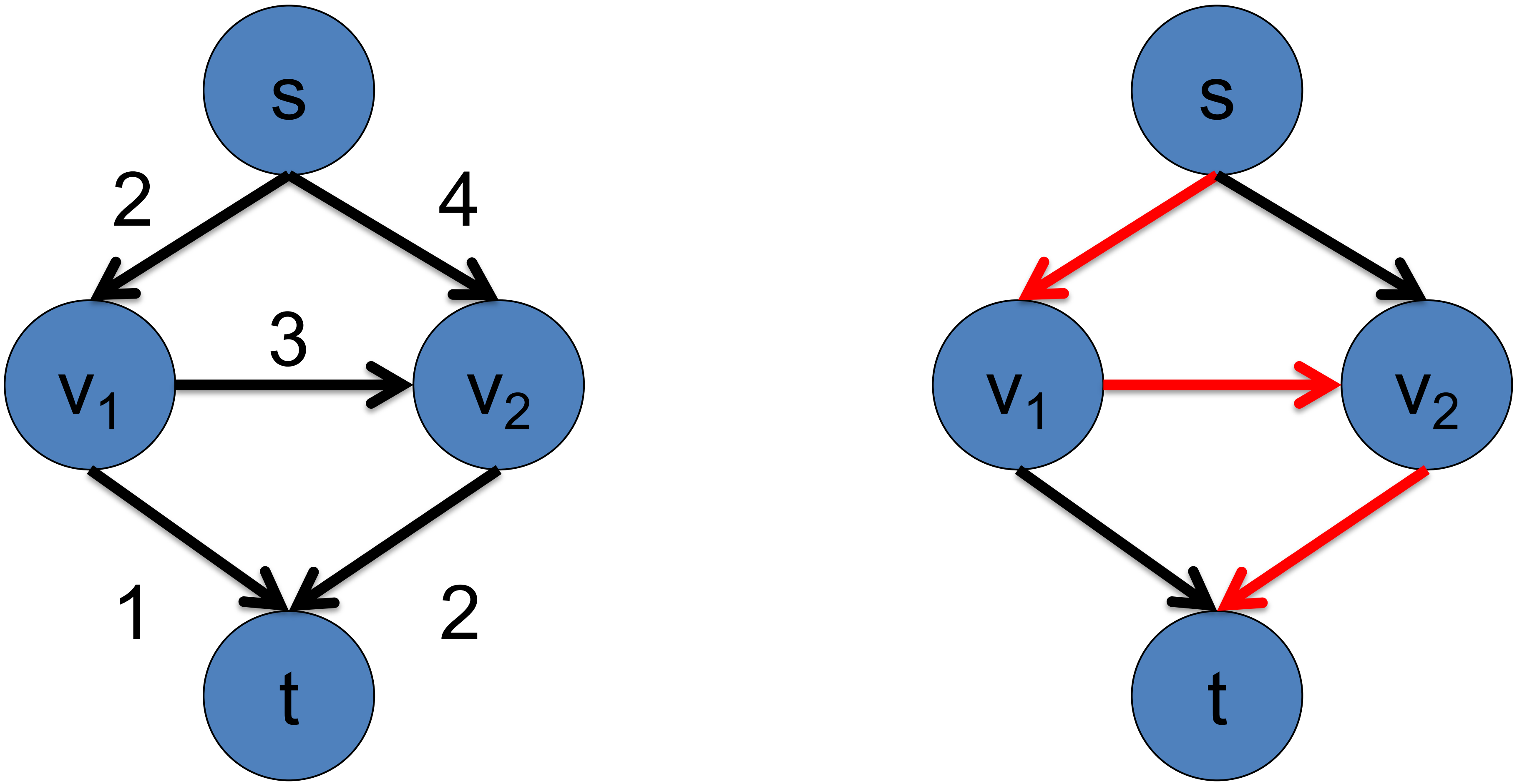
Find an s-t path in the residual graph.

Maximum Flow using Residual Graphs



For inverse arcs in path, subtract flow K.

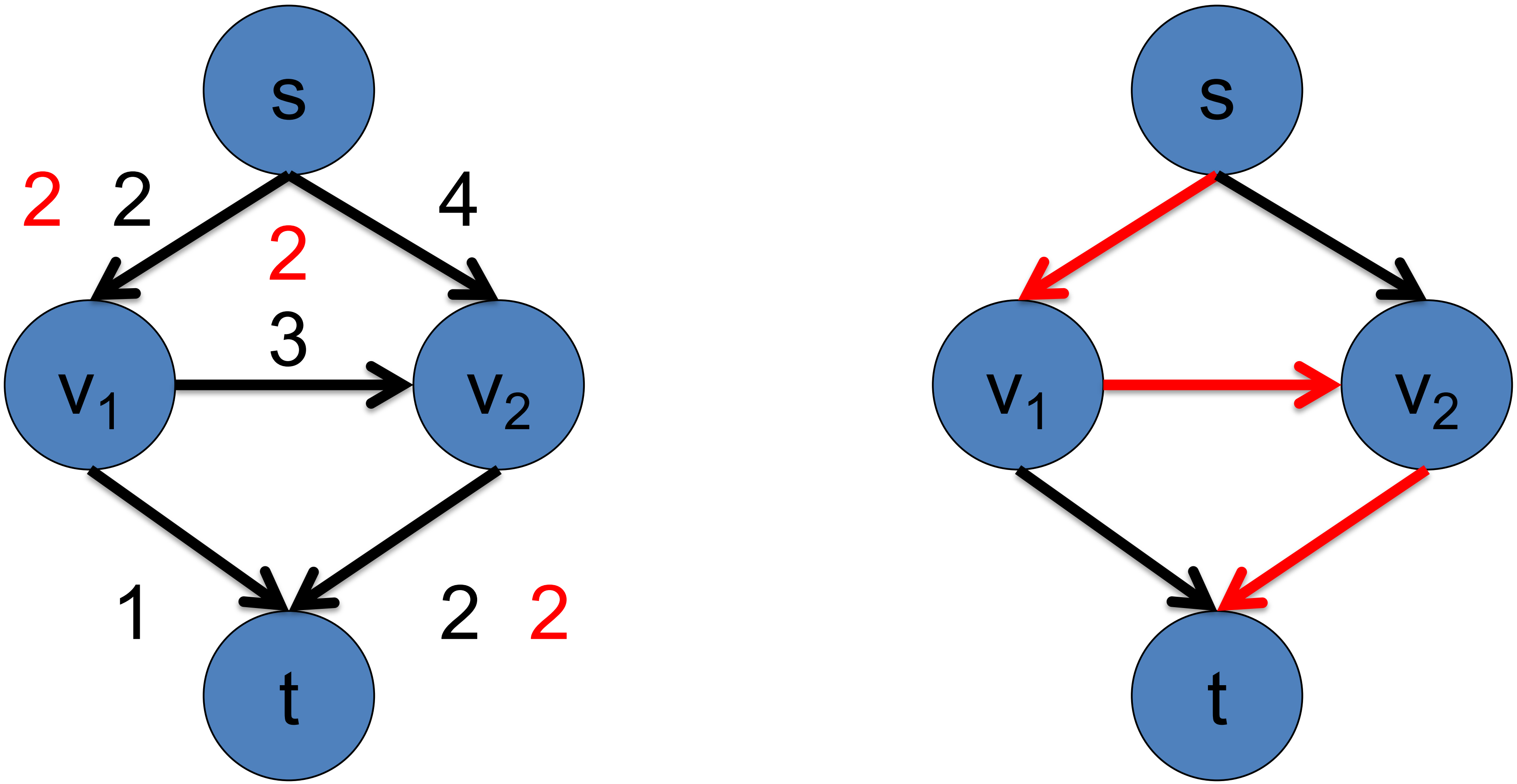
Maximum Flow using Residual Graphs



Choose maximum allowable value of K.

For forward arcs in path, add flow K.

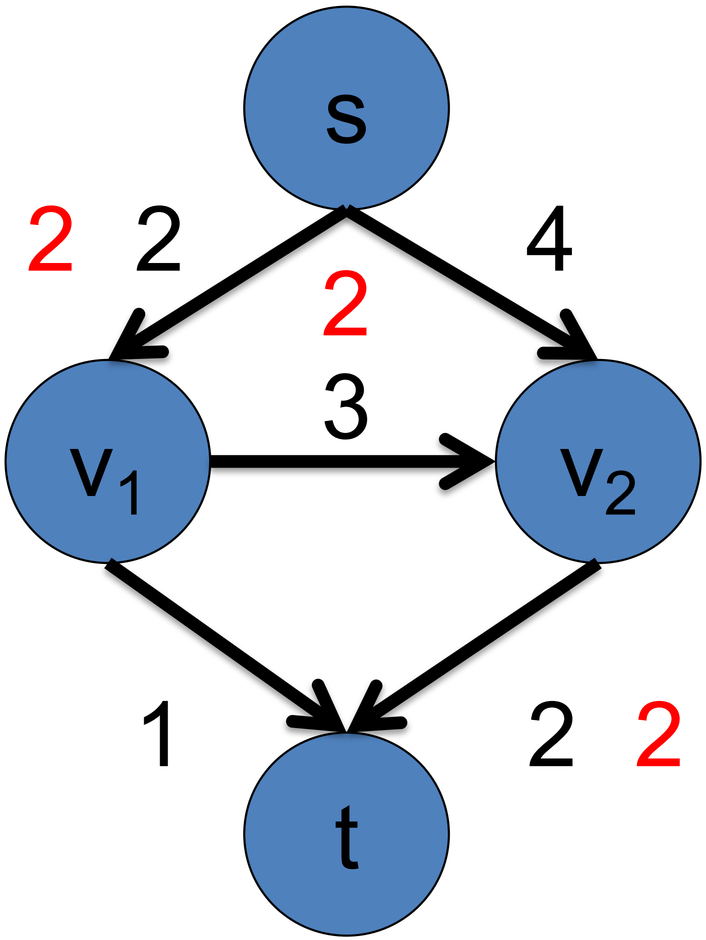
Maximum Flow using Residual Graphs

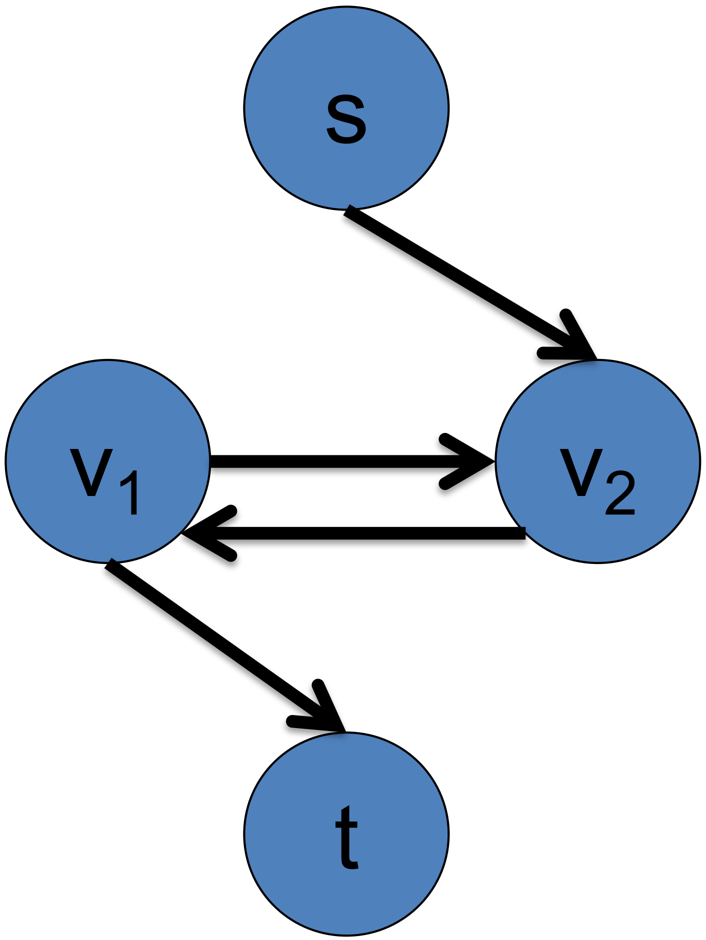


Choose maximum allowable value of K.

For forward arcs in path, add flow K.

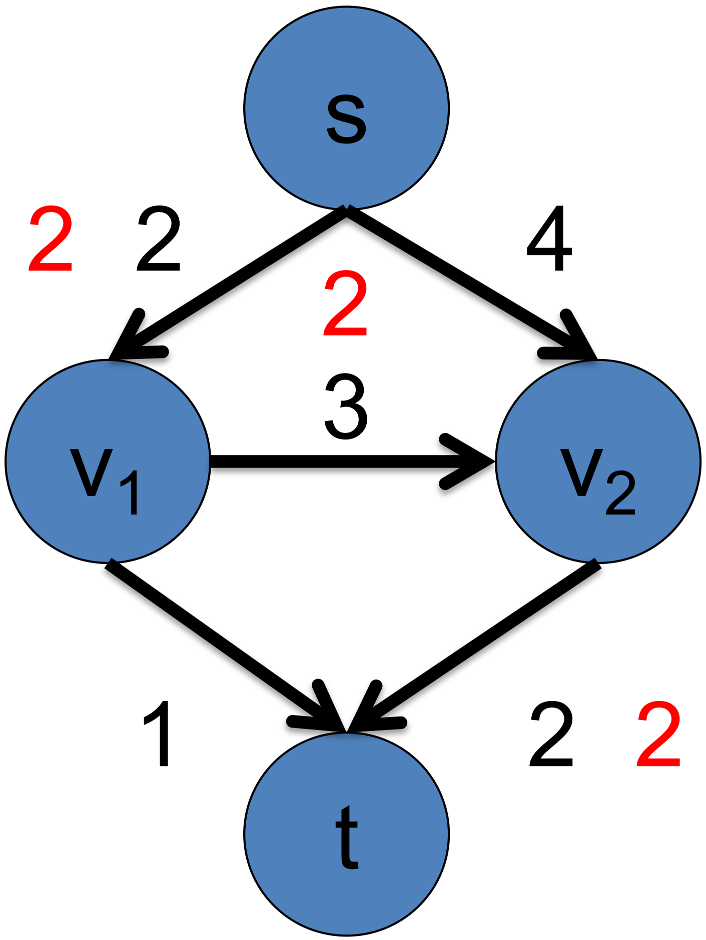
Maximum Flow using Residual Graphs

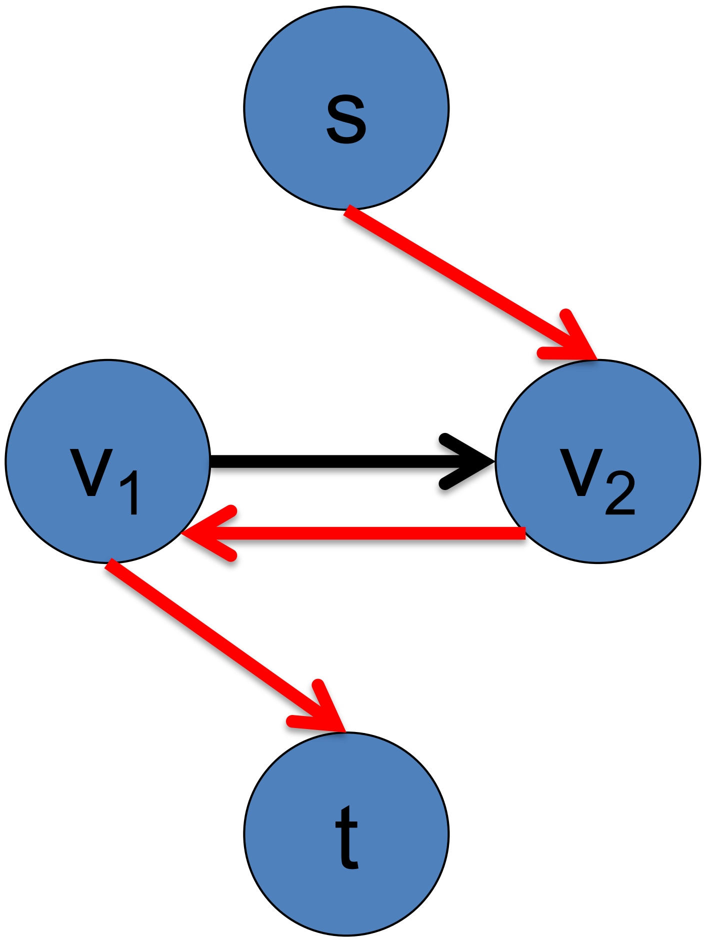




Update the residual graph.

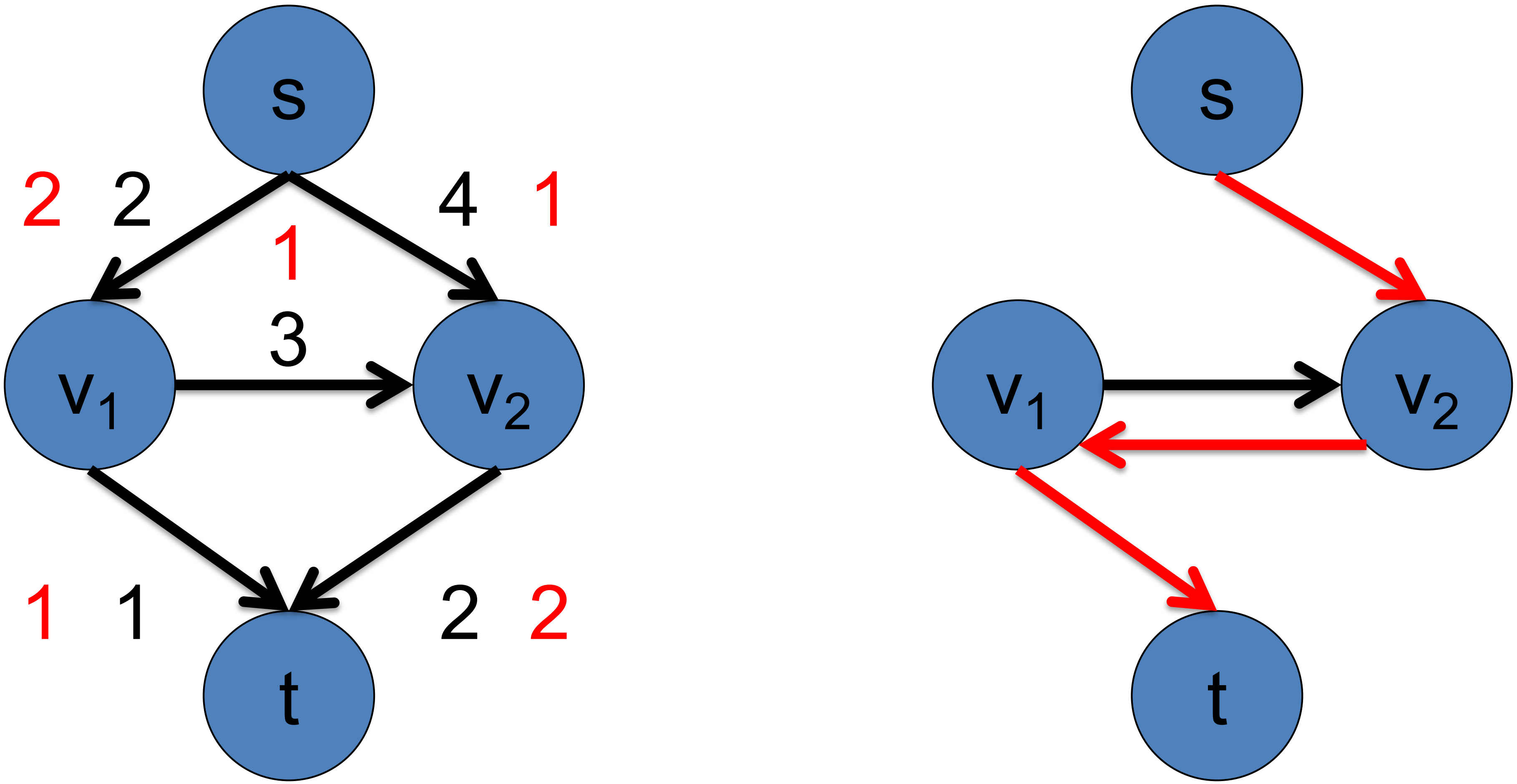
Maximum Flow using Residual Graphs





Find an s-t path in the residual graph.

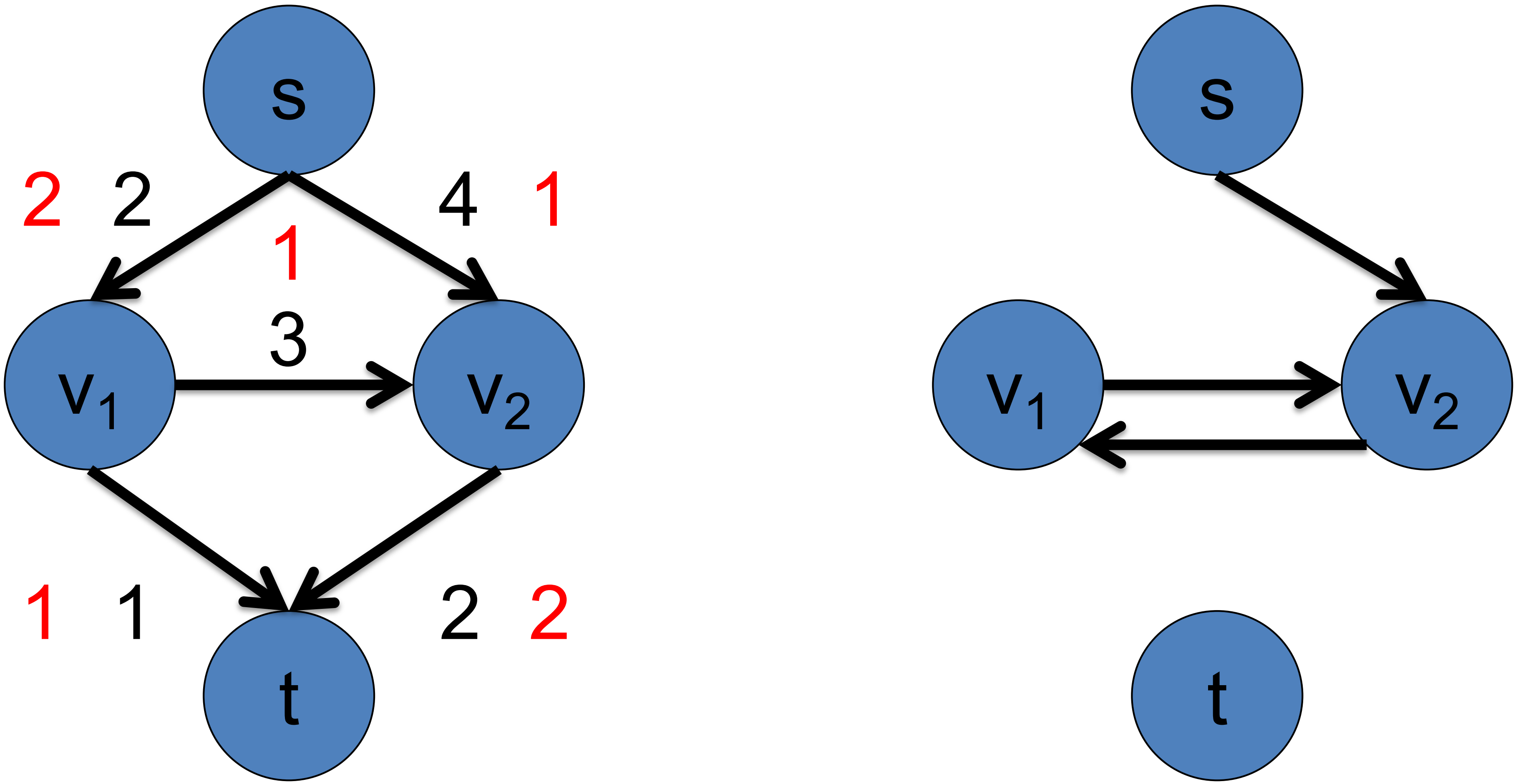
Maximum Flow using Residual Graphs



Choose maximum allowable value of K.

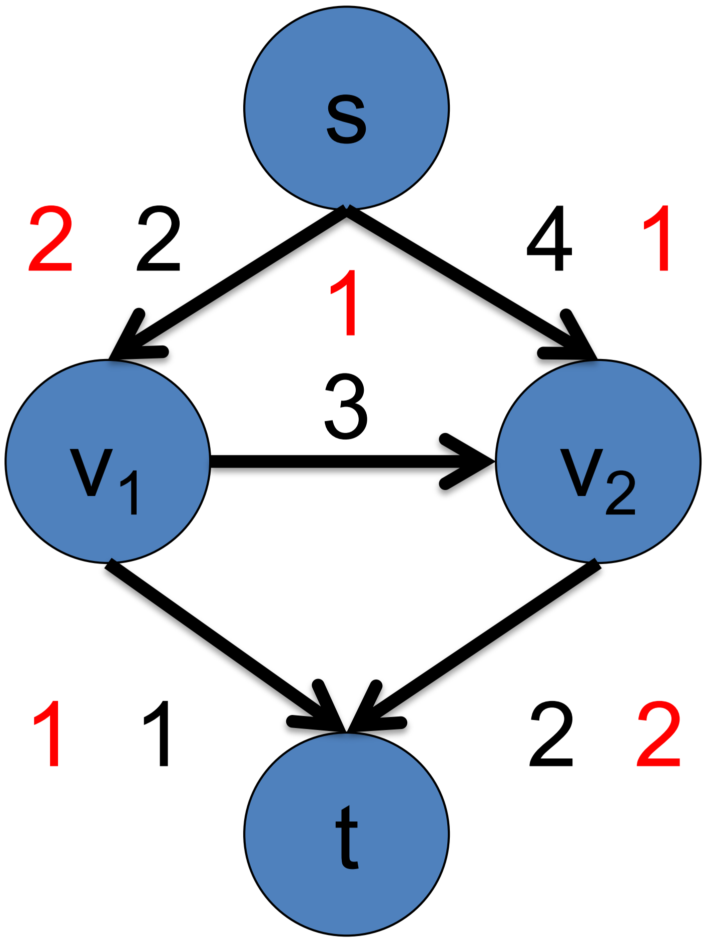
Add K to (s,v_2) and (v_1,t). Subtract K from (v_1,v_2).

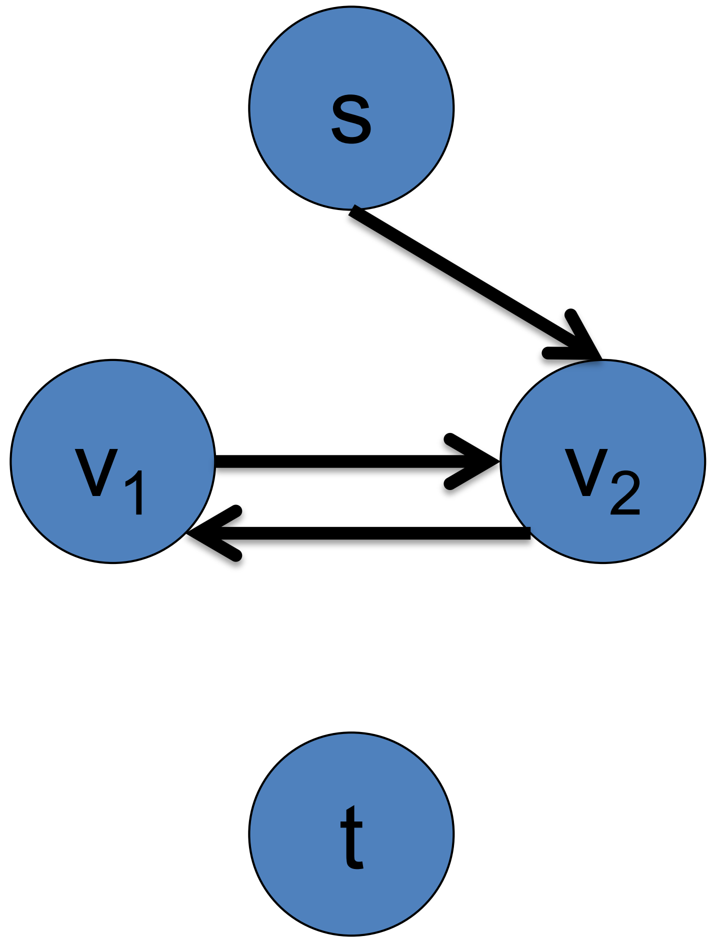
Maximum Flow using Residual Graphs



Update the residual graph.

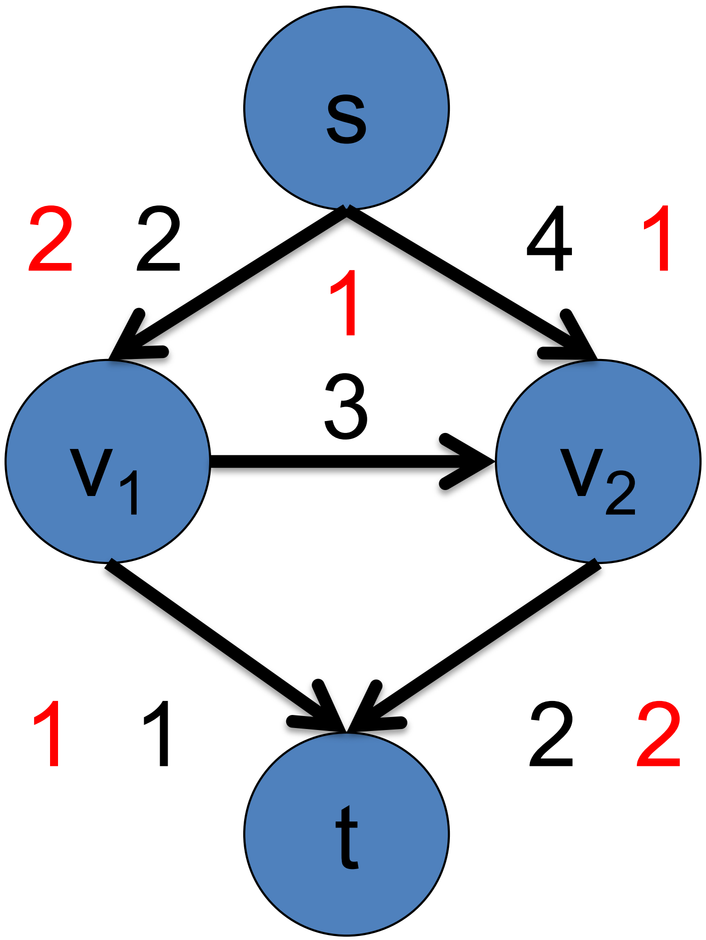
Maximum Flow using Residual Graphs

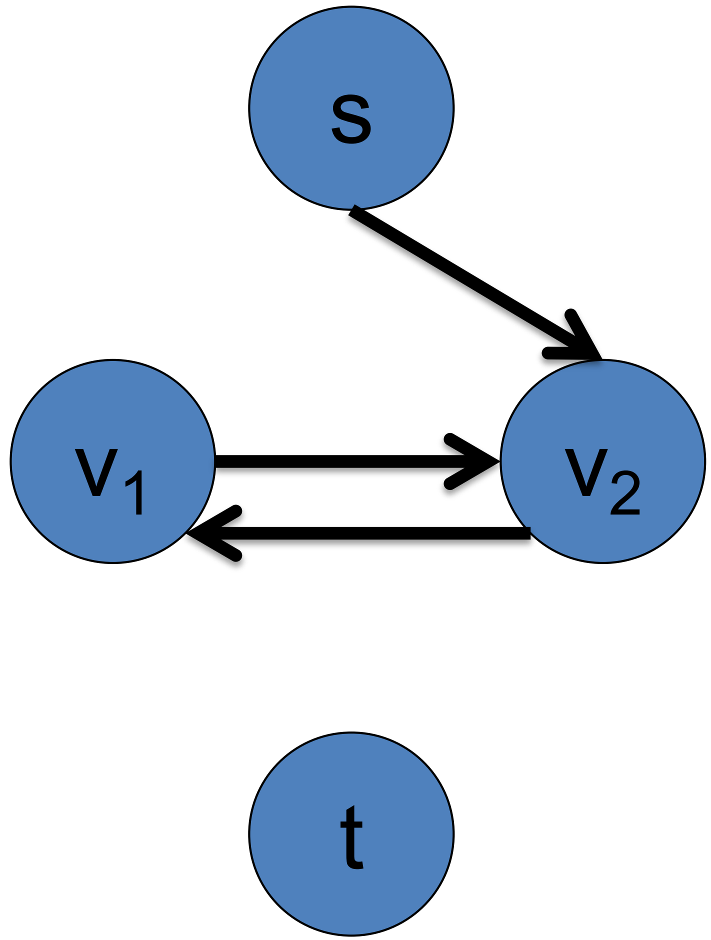




Find an s-t path in the residual graph.

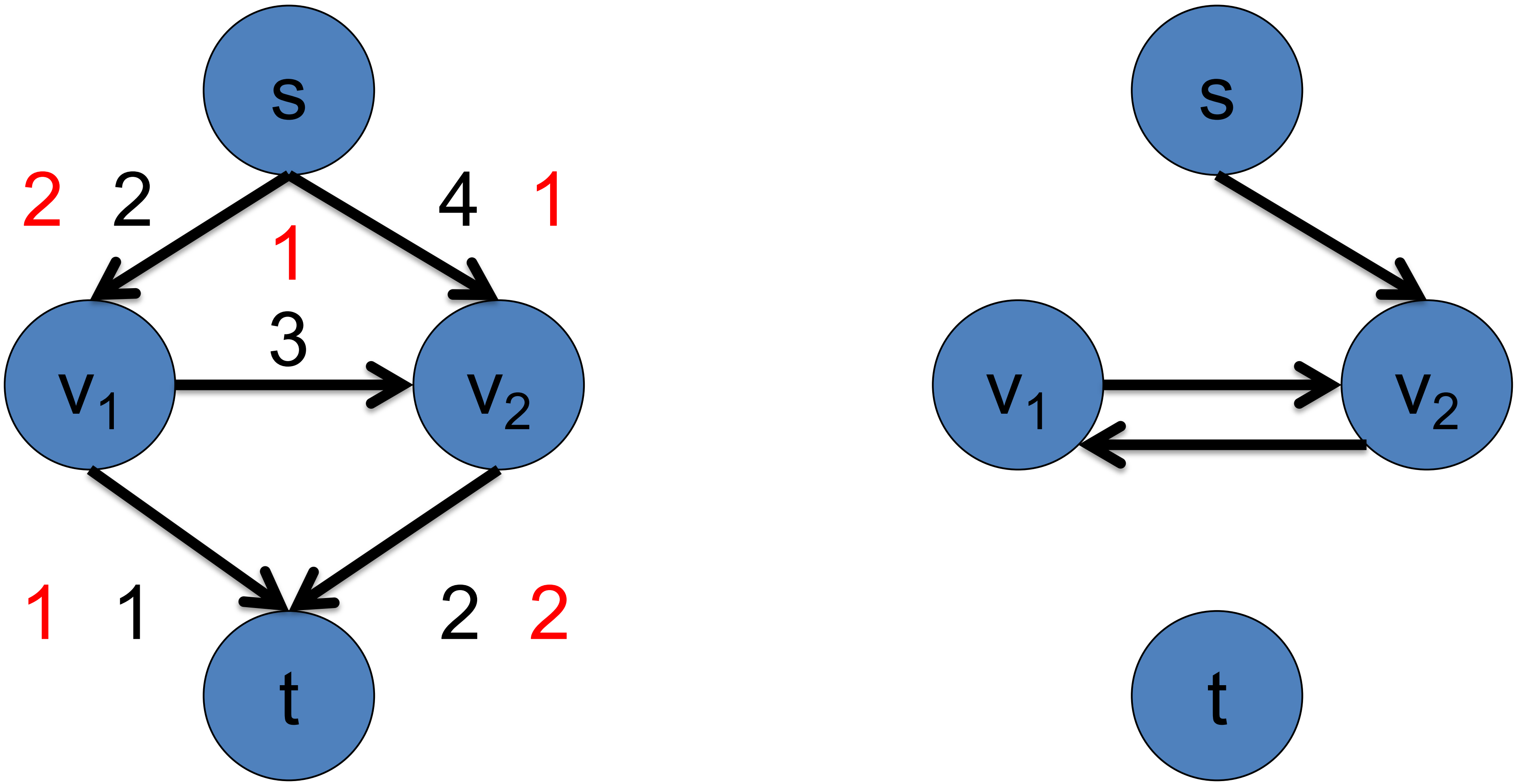
Maximum Flow using Residual Graphs





No more s-t paths. Stop.

Maximum Flow using Residual Graphs



Correct Answer.

Outline

- Preliminaries
- Maximum Flow
  - Residual Graph
  - Max-Flow Min-Cut Theorem
- Algorithms
- Energy minimization with max flow/min cut

History of Maxflow Algorithms

Augmenting Path and Push-Relabel

year	discoverer(s)	bound
1951	Dantzig	$O(n^2 mU)$
1955	Ford & Fulkerson	$O(m^2 U)$
1970	Dinitz	$O(n^2 m)$
1972	Edmonds & Karp	$O(m^2 \log U)$
1973	Dinitz	$O(nm \log U)$
1974	Karzanov	$O(n^3)$
1977	Cherkassky	$O(n^2 m^{1/2})$
1980	Galil & Naamad	$O(nm \log^2 n)$
1983	Sleator & Tarjan	$O(nm \log n)$
1986	Goldberg & Tarjan	$O(nm \log(n^2/m))$
1987	Ahuja & Orlin	$O(nm + n^2 \log U)$
1987	Ahuja et al.	$O(nm \log(n\sqrt{\log U}/m))$
1989	Cheriyan & Hagerup	$E(nm + n^2 \log^2 n)$
1990	Cheriyan et al.	$O(n^3/\log n)$
1990	Alon	$O(nm + n^{8/3} \log n)$
1992	King et al.	$O(nm + n^{2+\epsilon})$
1993	Phillips & Westbrook	$O(nm(\log_{m/n} n + \log^{2+\epsilon} n))$
1994	King et al.	$O(nm \log_{m/(n \log n)} n)$
1997	Goldberg & Rao	$O(m^{3/2} \log(n^2/m) \log U)$ $O(n^{2/3} m \log(n^2/m) \log U)$

n: #nodes

m: #edges

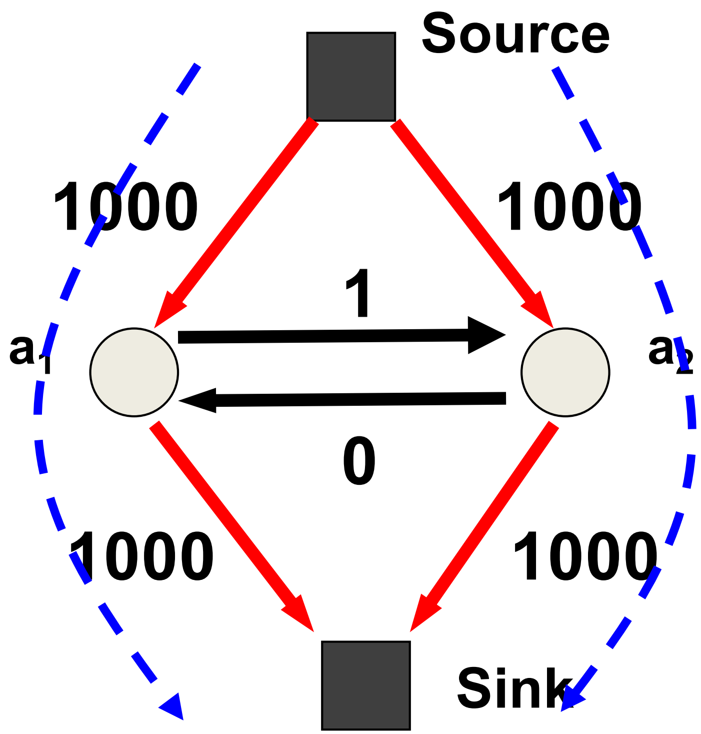
U: maximum edge weight

Algorithms assume non-negative edge weights

[Slide credit: Andrew Goldberg]

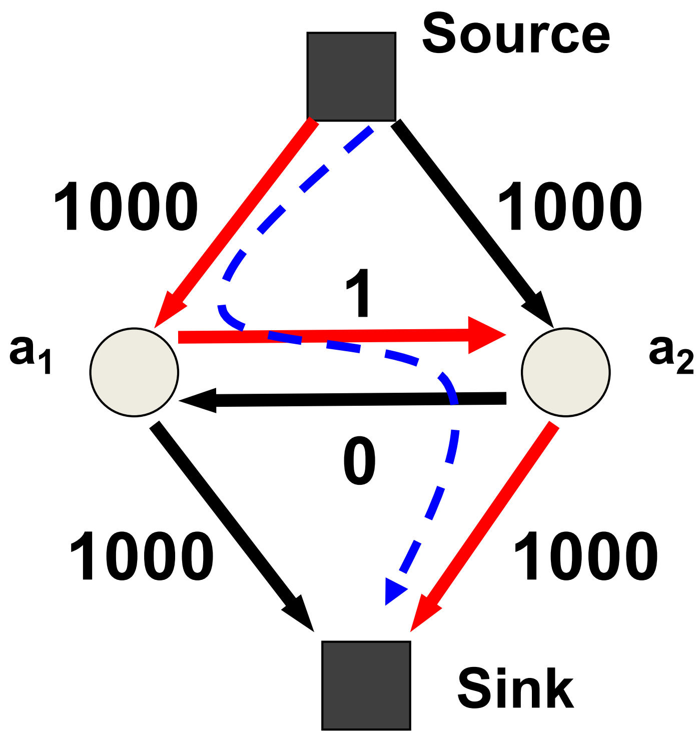
Augmenting Path based Algorithms

Ford Fulkerson: Choose **any** augmenting path



Augmenting Path based Algorithms

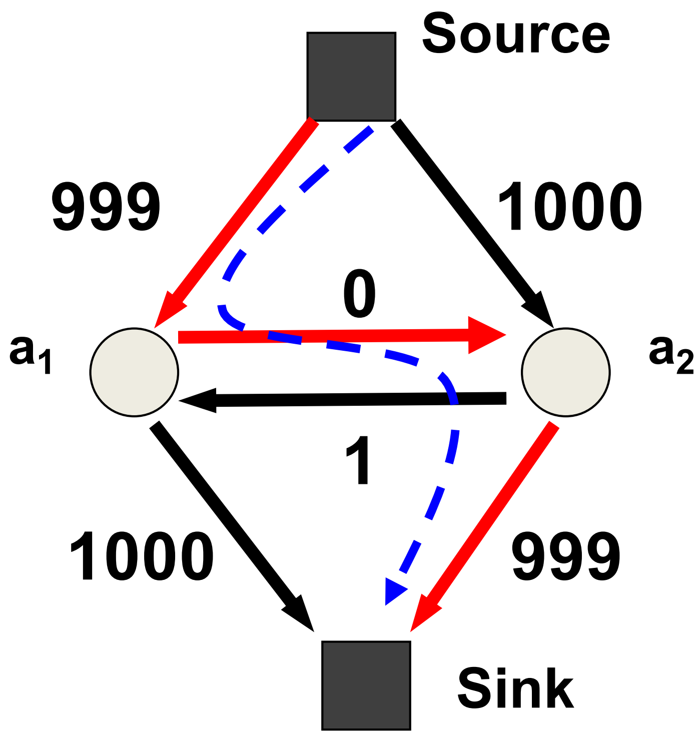
Ford Fulkerson: Choose **any** augmenting path



Bad Augmenting Path

Augmenting Path based Algorithms

Ford Fulkerson: Choose **any** augmenting path

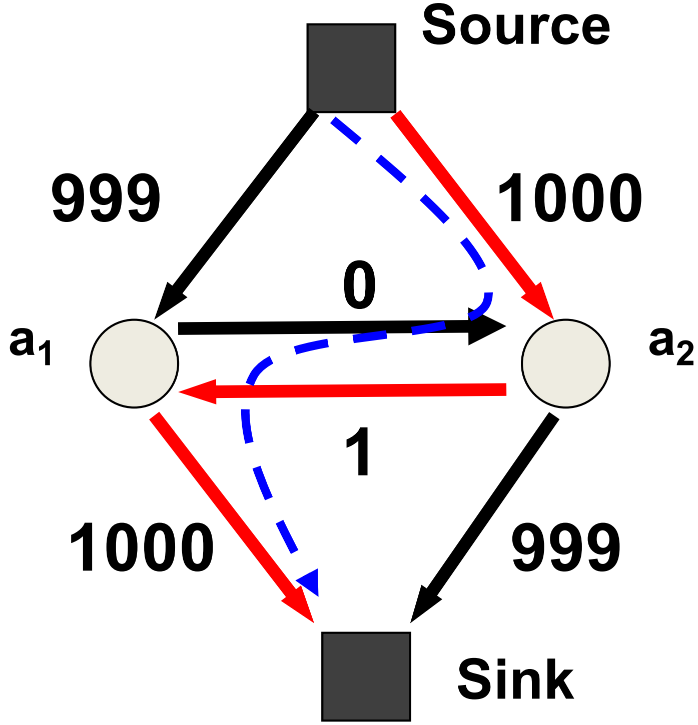


Augmenting Path based Algorithms

n: #nodes

m: #edges

Ford Fulkerson: Choose **any** augmenting path



We will have to perform 2000 augmentations!

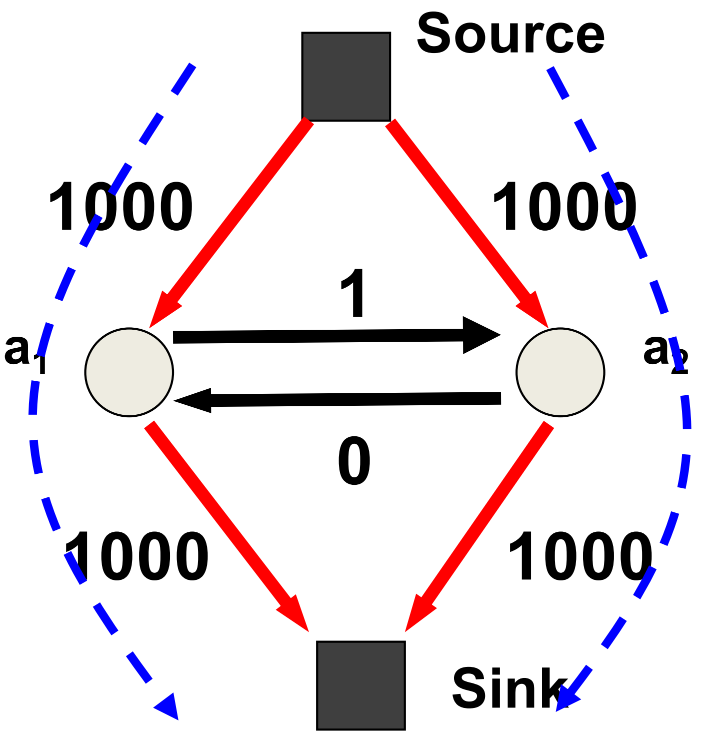
Worst case complexity: O (m x Total_Flow)
(Pseudo-polynomial bound: depends on flow)

Augmenting Path based Algorithms

n: #nodes

m: #edges

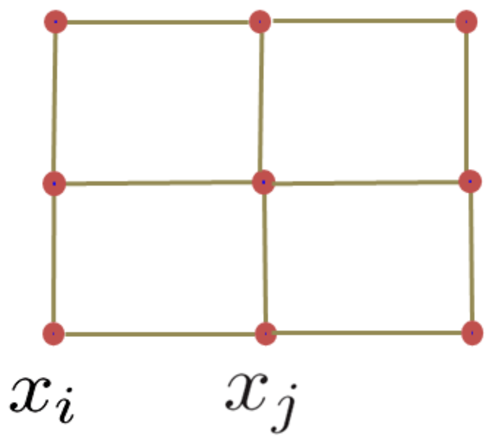
Dinitz: Choose **shortest** augmenting path



Worst case complexity: O (m n^2)

Maxflow in Computer Vision

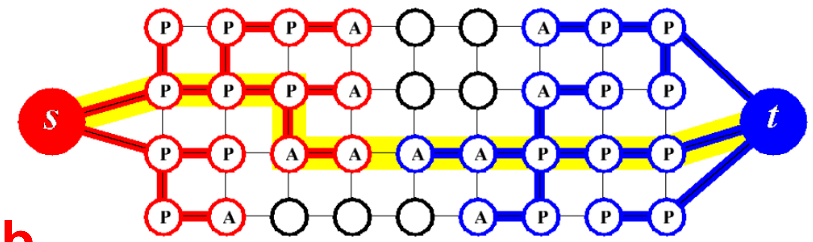
- **Specialized algorithms for vision problems**
  - Grid graphs
  - Low connectivity (m ~ O(n))



- **Dual search tree augmenting path algorithm**

  [Boykov and Kolmogorov PAMI 2004]
  - Finds approximate shortest augmenting paths efficiently
  - High worst-case time complexity
  - Empirically outperforms other algorithms on vision problems
  - Efficient code available on the web

    e.g., http://pub.ist.ac.at/~vnk/software.html



Outline

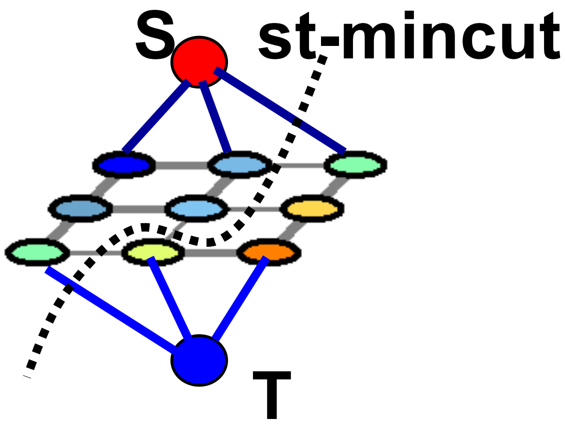
The st-mincut problem

Connection between st-mincut and energy minimization?

What problems can we solve using st-mincut?

st-mincut based Move algorithms

St-mincut and Energy Minimization



↔

Minimizing a Qudratic Pseudoboolean function E(x)

Functions of boolean variables

Pseudoboolean?

$$E: \{0,1\}^n \rightarrow R$$

$$E(y) = \sum_i c_i y_i + \sum_{i,j} c_{ij} y_i(1-y_j)$$

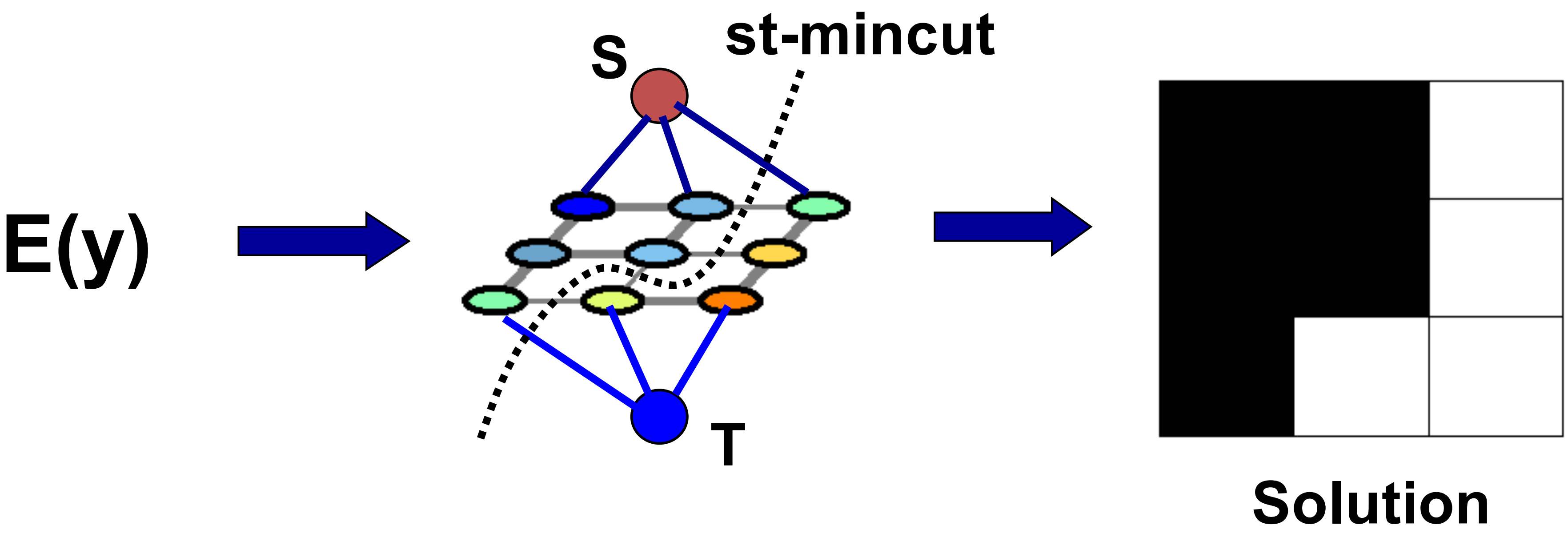
$c_{ij} \geq 0$

Polynomial time st-mincut algorithms require non-negative edge weights

So how does this work?

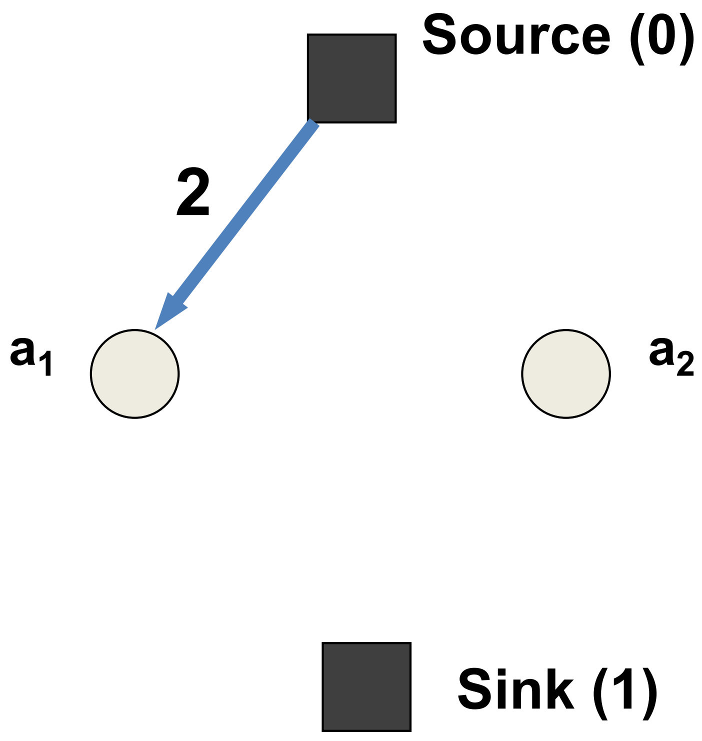
Construct a graph such that:

1. Any st-cut corresponds to an assignment of x
2. The cost of the cut is equal to the energy of x : E(x)



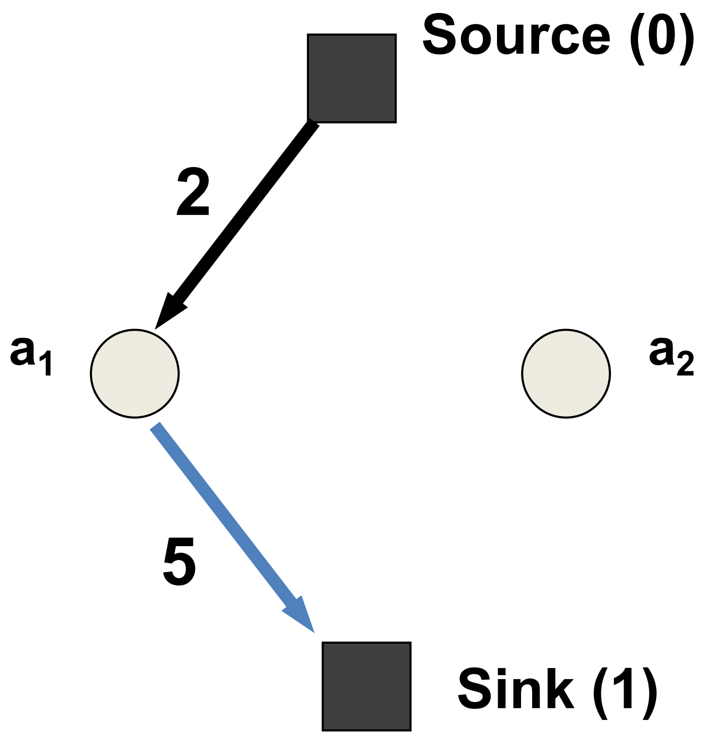
Graph Construction

$E(a_1,a_2) = 2a_1$



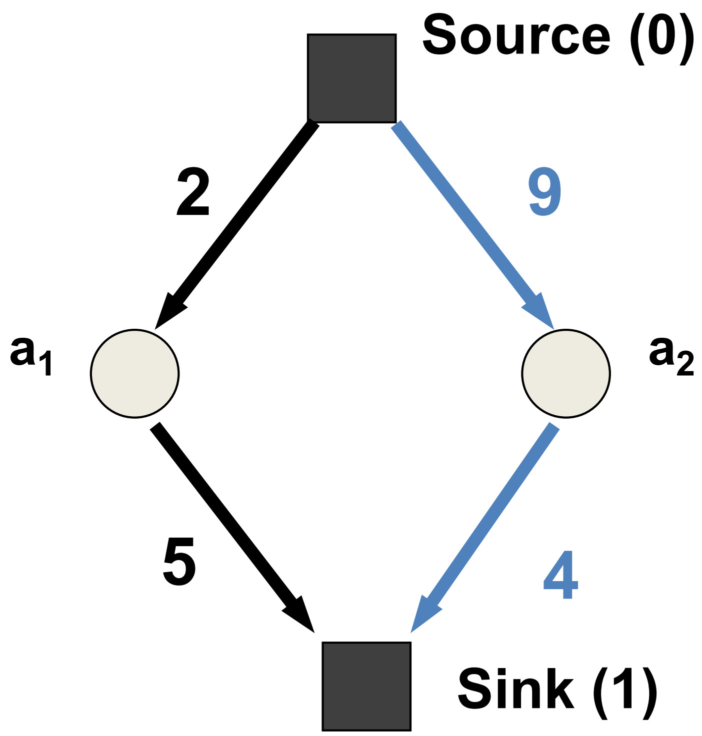
Graph Construction

$E(a_1,a_2) = 2a_1 + 5\bar{a}_1$



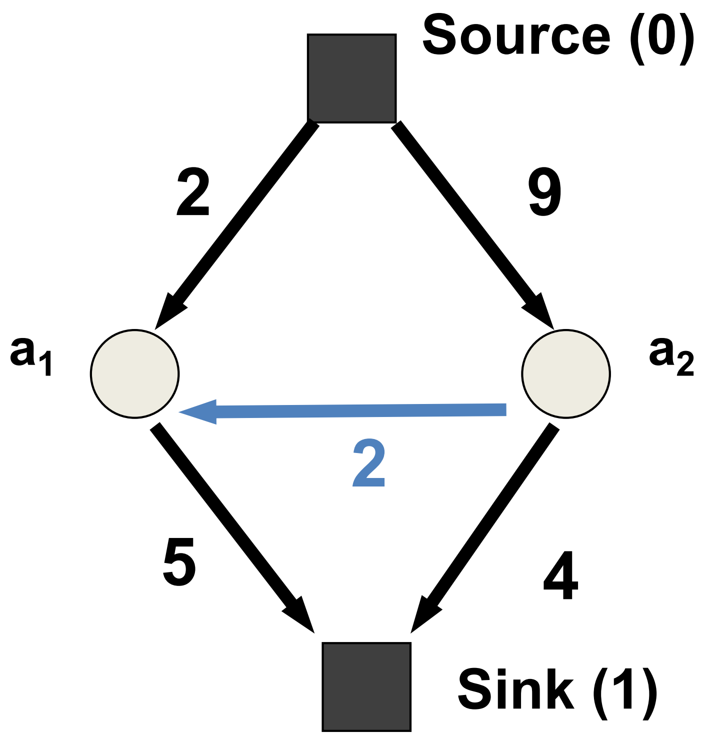
Graph Construction

$E(a_1,a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2$



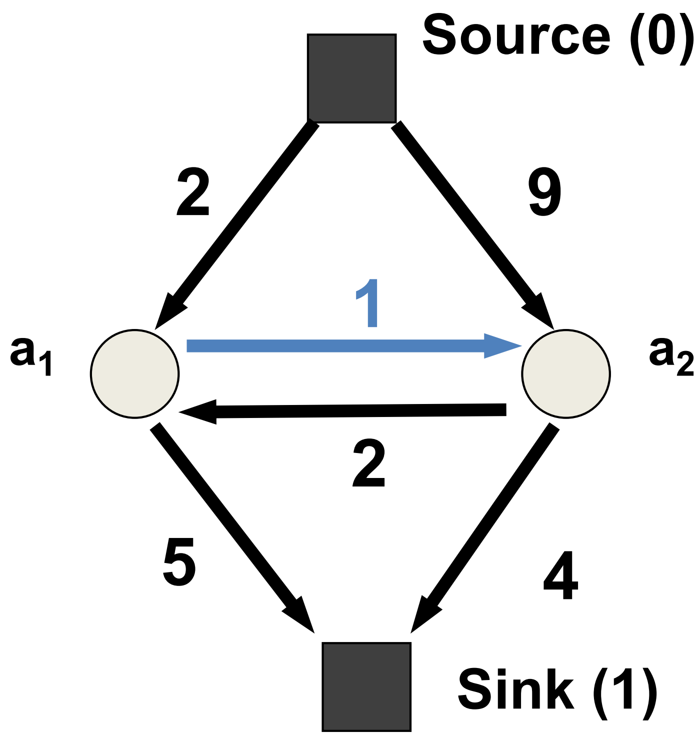
Graph Construction

$E(a_1,a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2$



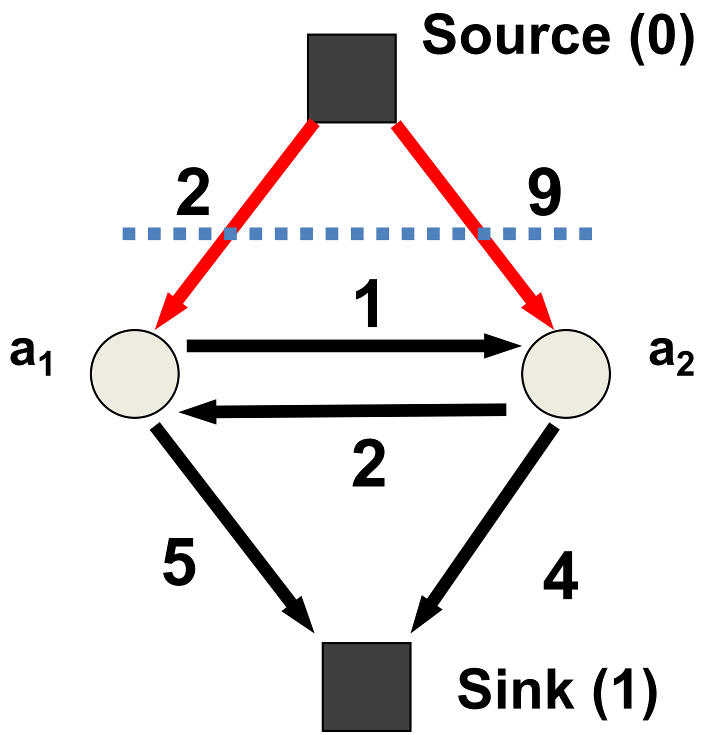
Graph Construction

$E(a_1,a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1 a_2$



Graph Construction

$E(a_1,a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2$



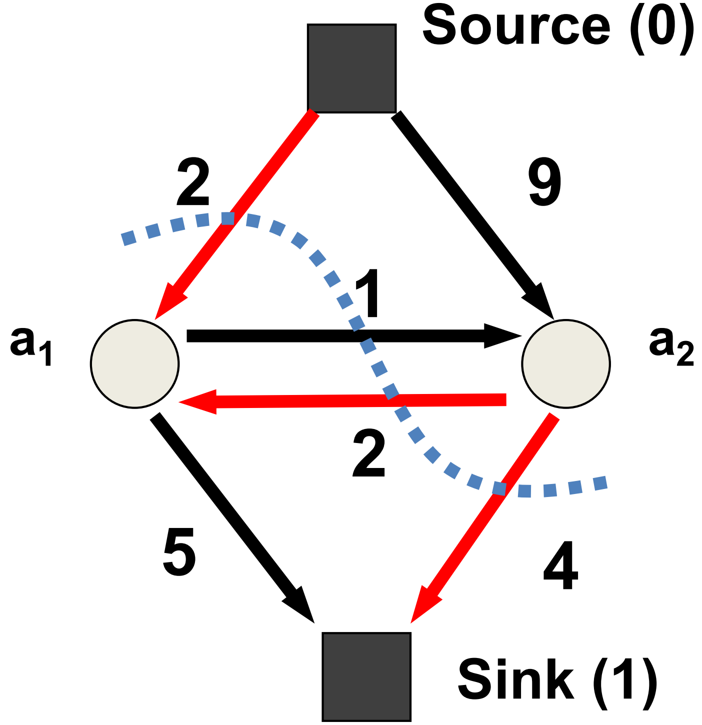
Cost of cut = 11

$a_1 = 1 \quad a_2 = 1$

E (1,1) = 11

Graph Construction

$E(a_1,a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2$



st-mincut cost = 8

$a_1 = 1 \quad a_2 = 0$

E (1,0) = 8

Example: Image Segmentation

$$E(y) = \sum_{i} c_i y_i + \sum_{i,j} c_{ij} y_i(1-y_j)$$

$E: \{0,1\}^n \rightarrow R$
$0 \rightarrow fg$
$1 \rightarrow bg$

Global Minimum (y^*)

$$y^* = \arg\min_{y} E(y)$$

How to minimize E(x)?

How does the code look like?

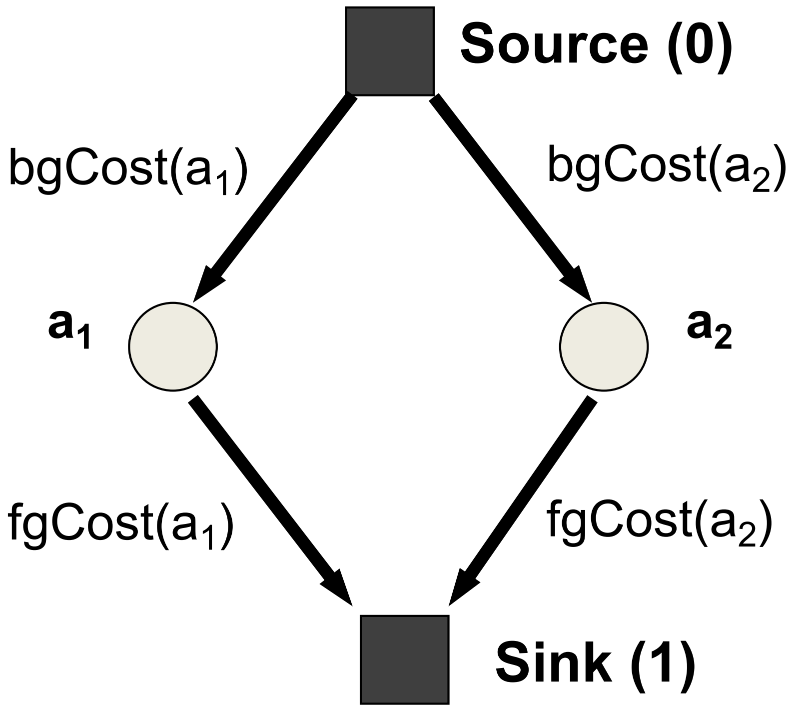
```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q),  cost);
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```

Source (0)

Sink (1)

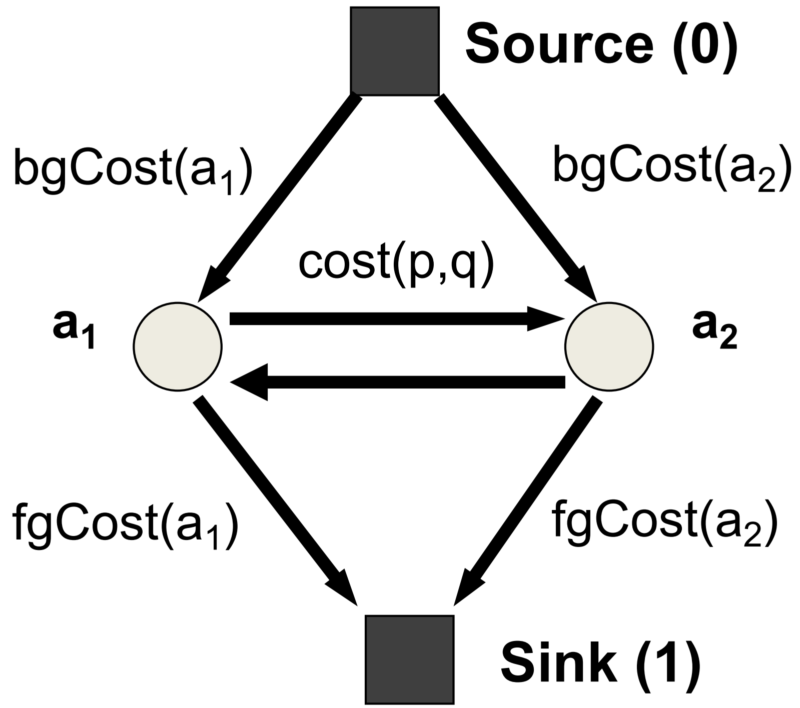
How does the code look like?

```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q),  cost);
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



How does the code look like?

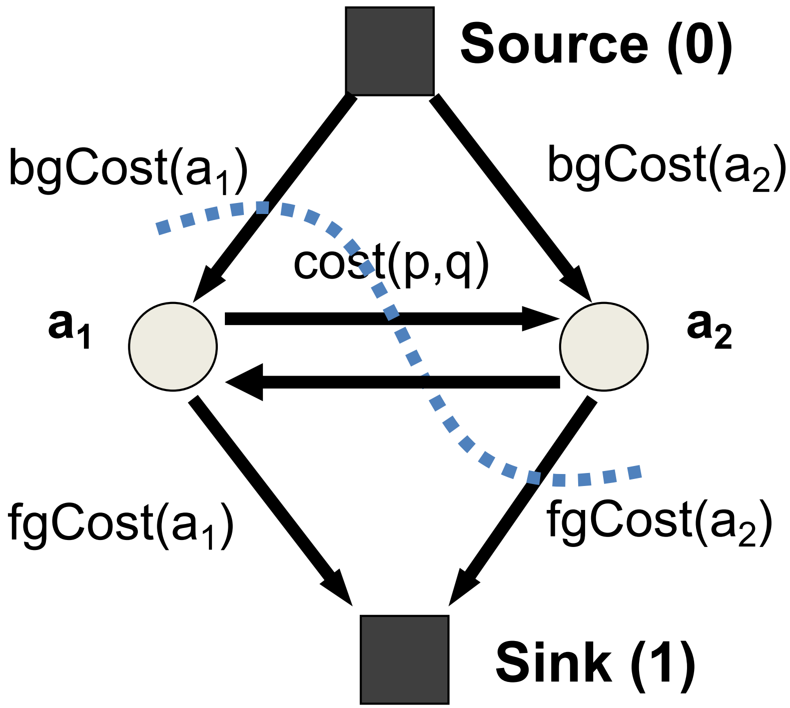
```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q),  cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



How does the code look like?

```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q),  cost(p,q));
end
```

```
g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



a_1 = bg a_2 = fg

Outline

The st-mincut problem

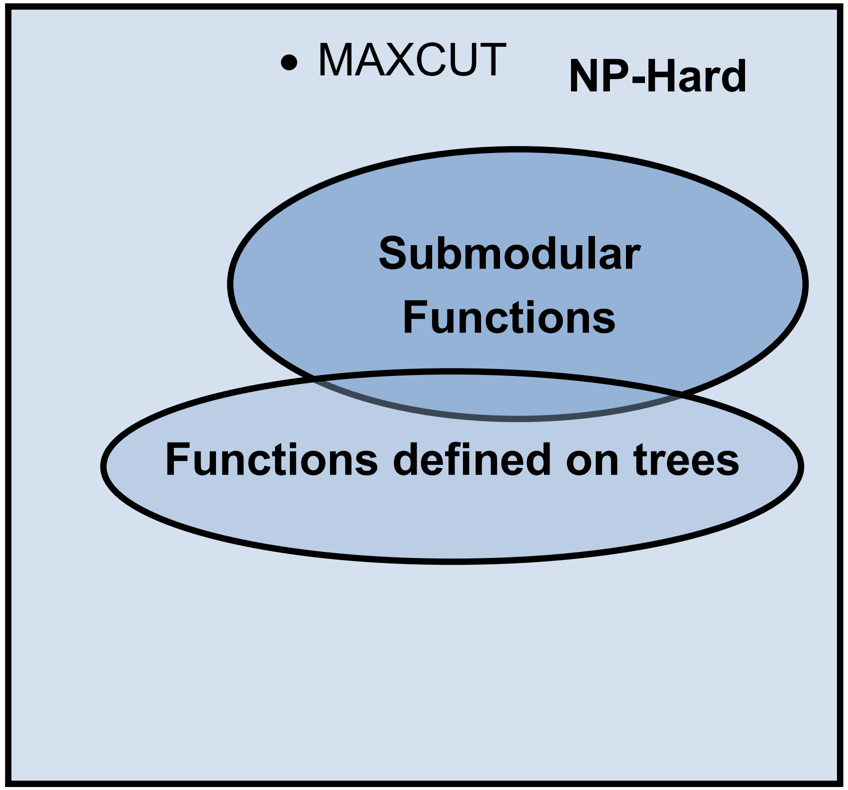
Connection between st-mincut and energy minimization?

What problems can we solve using st-mincut?

st-mincut based Move algorithms

Minimizing Energy Functions

- General Energy Functions
  - NP-hard to minimize
  - Only approximate minimization possible
- Easy energy functions
  - Solvable in polynomial time
  - Submodular ~ $O(n^6)$



Space of Function Minimization Problems