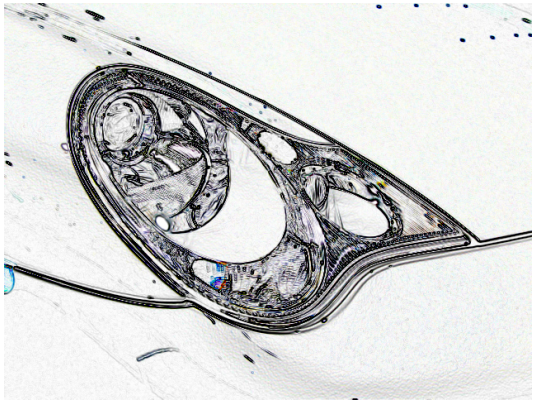
CSE578: Computer Vision

Spring 2017:

MRF for Segmentation


Anoop M. Namboodiri

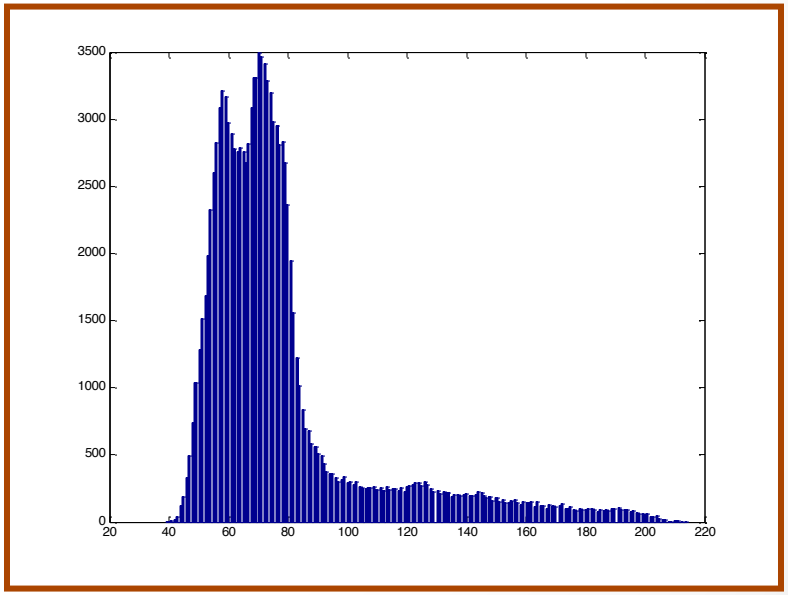
Center for Visual Information Technology

IIIT Hyderabad, INDIA

Optimal Thresholding

- The graylevel histogram is approximated using a mixture of two gaussian distributions and set the threshold to minimize the segmentation error

Grayscale Image



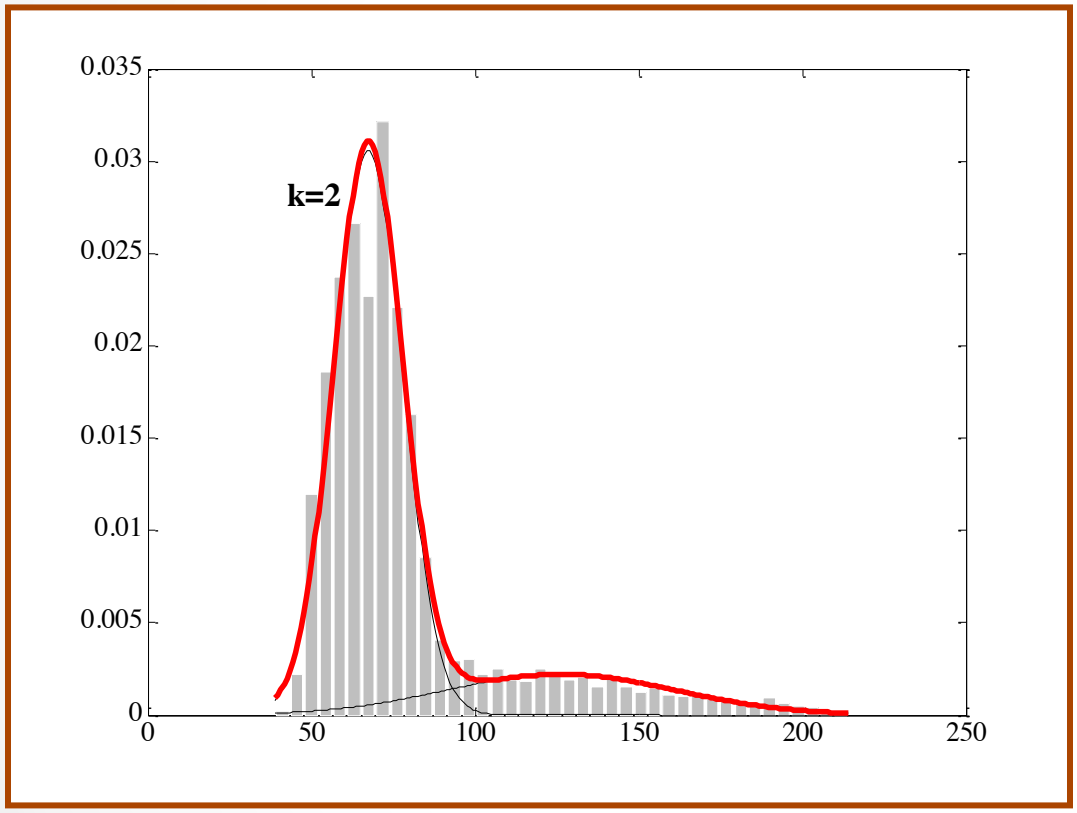
Histogram

Gaussian Mixture Estimation by EM

- Obj: = $$N(\mu_1,\sigma_1)=\frac{1}{\sigma_1\sqrt{2\pi}}e^{\frac{(x-\mu_1)^2}{2\sigma_1^2}}$$
- Bkg: = $$N(\mu_2,\sigma_2)=\frac{1}{\sigma_2\sqrt{2\pi}}e^{\frac{(x-\mu_2)^2}{2\sigma_2^2}}$$

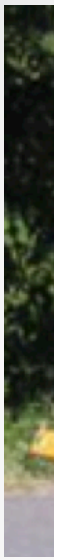
- Initialize μ_1, σ_1, μ_2, and σ_2.
- E-Step: Computed the expected pixel label assignments. This could be either hard or soft assignment.
- M-Step: Computed Maximum-(Log)Likelihood estimates of the parameters: $\mu_1,\sigma_1,\mu_2,\sigma_2$
- Repeat the E and M steps until convergence

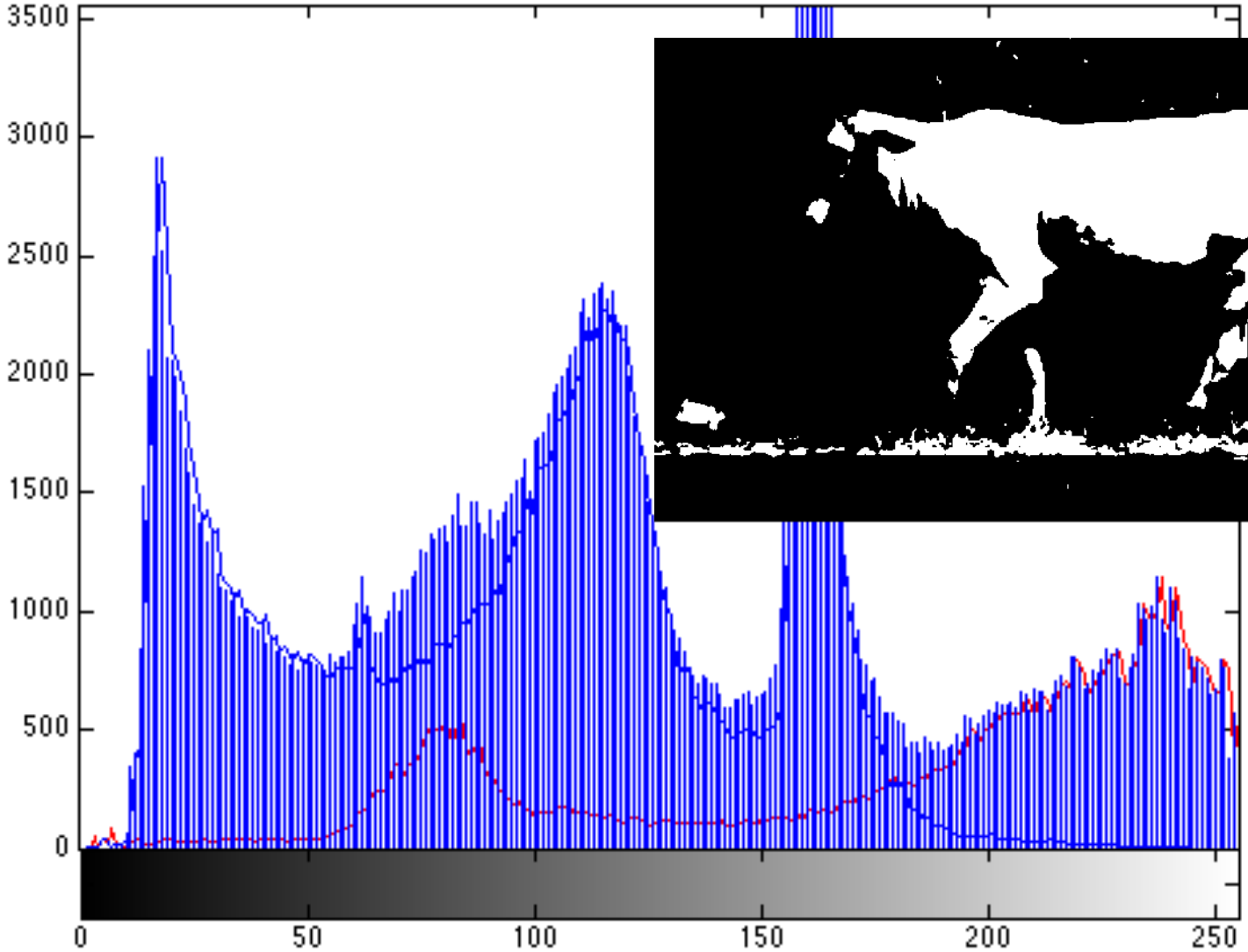
Optimal Thresholding



Histogram with bimodal fit

Thresholded (T=94)



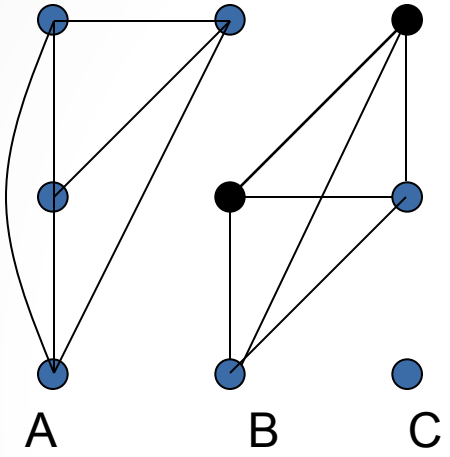

3500
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250


300

Segmentation as Optimal Labeling

- Model knowledge about the world
- Classify each pixel as belonging to a specific object
  - Independent classification does not work
  - Need to incorporate neighborhood information
- Consider a graph over the image
  - Each node in the graph need to be labeled
  - Edges in the graph represent neighborhood constraints
- Define a cost function, $Q(f)$, using the above
- Compute the optimal labeling wrt $Q(f)$.

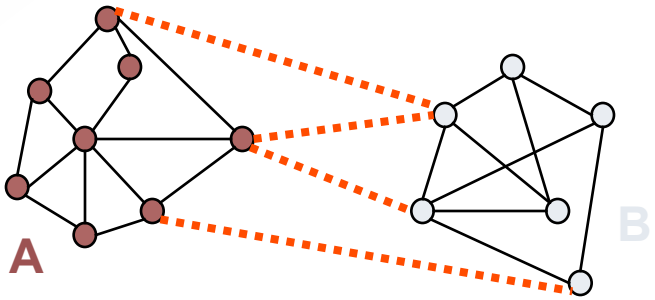
Segmentation by Graph Cuts



- Break Graph into Segments
  - Delete links that cross between segments
  - Easiest to break links that have low cost (low similarity)
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments

Source: Seitz

Cuts in a graph



- Link Cut
  - set of links whose removal makes a graph disconnected
  - cost of a cut:

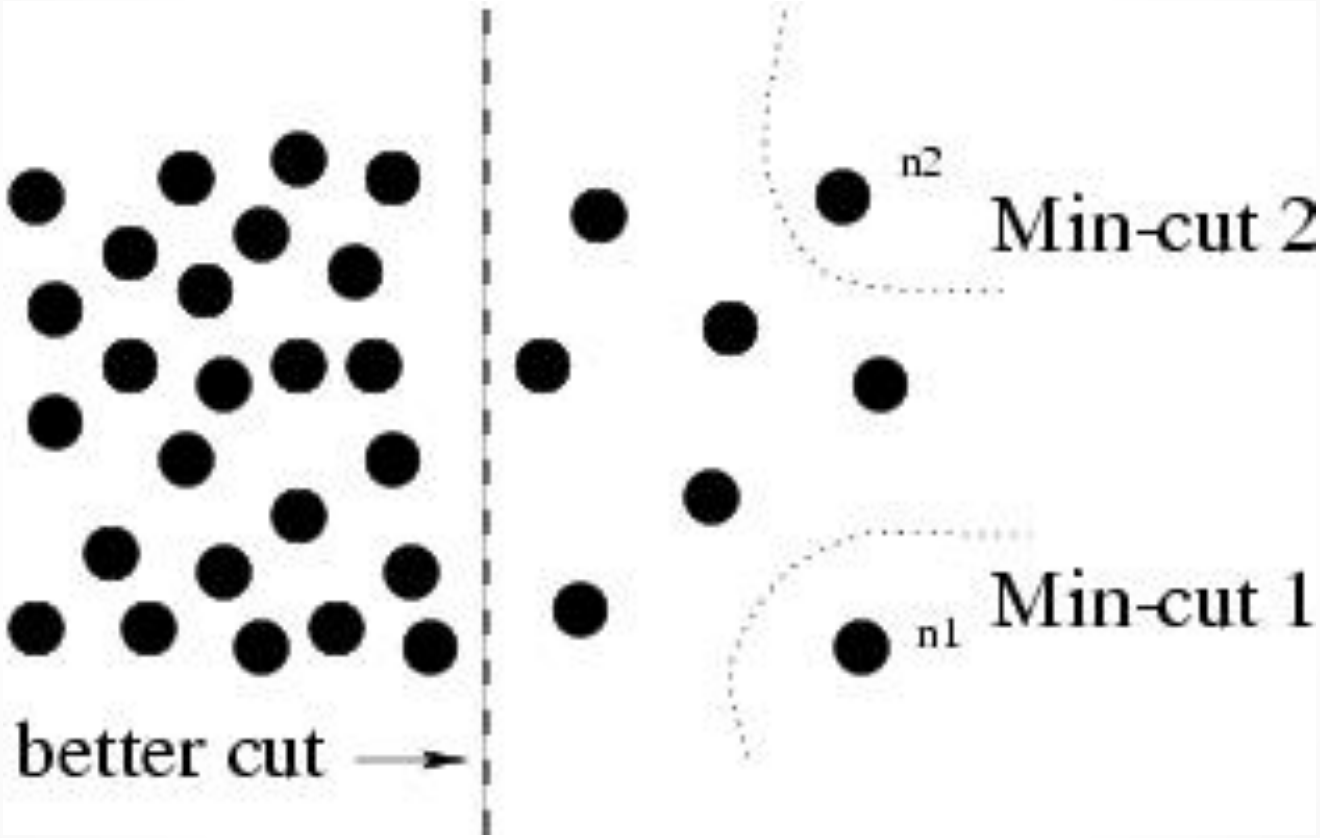
$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v).$$

One idea: Find minimum cut

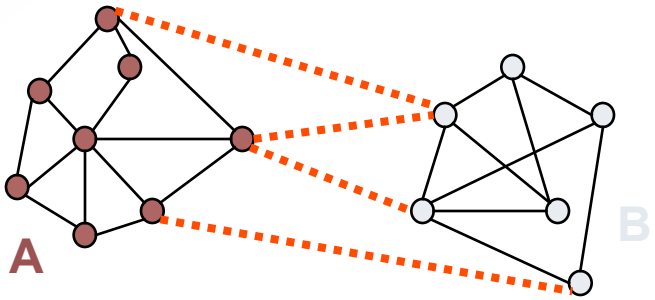
- gives you a segmentation
- fast algorithms exist for doing this

Source: Seitz

Min cut is not always the best



Cuts in a graph



Normalized Cut

- a cut penalizes large segments
- fix by normalizing for size of segments

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)},$$

where $assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$

Source: Seitz

Recursive normalized cuts

1. Given an image or image sequence, set up a weighted graph: G=(V, E)
   - Vertex for each pixel
   - Edge weight for nearby pairs of pixels

2. Solve for eigenvectors with the smallest eigenvalues: $(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda\mathbf{D}\mathbf{y}$, where
   - Use the eigenvector with the second smallest eigenvalue to bipartition the graph
   - Note: this is an approximation

3. Recursively repartition the segmented parts if necessary

Details: *http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf*

Normalized cuts results

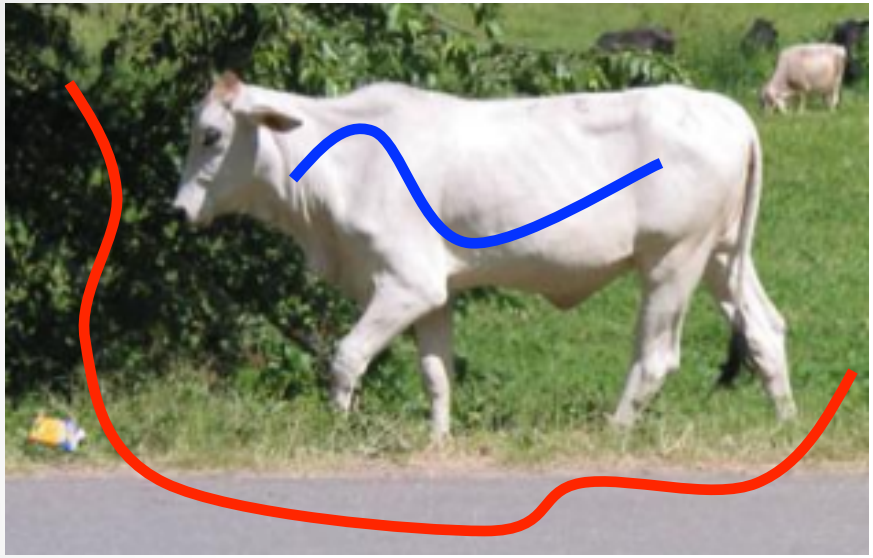
Graphcut for Image Segmentation

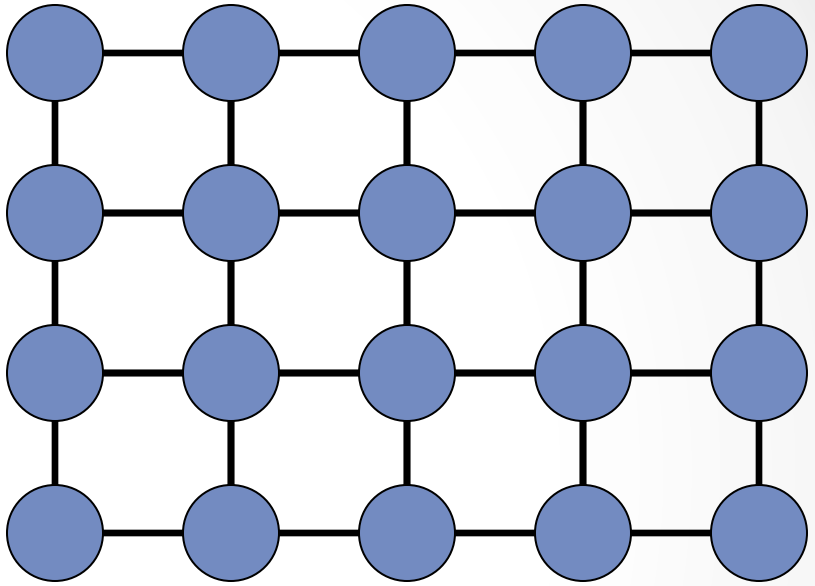
How ?

Cost function Models *our* knowledge about natural images

Optimize cost function to obtain the segmentation

Binary Image Segmentation





Object - white, Background - green/grey

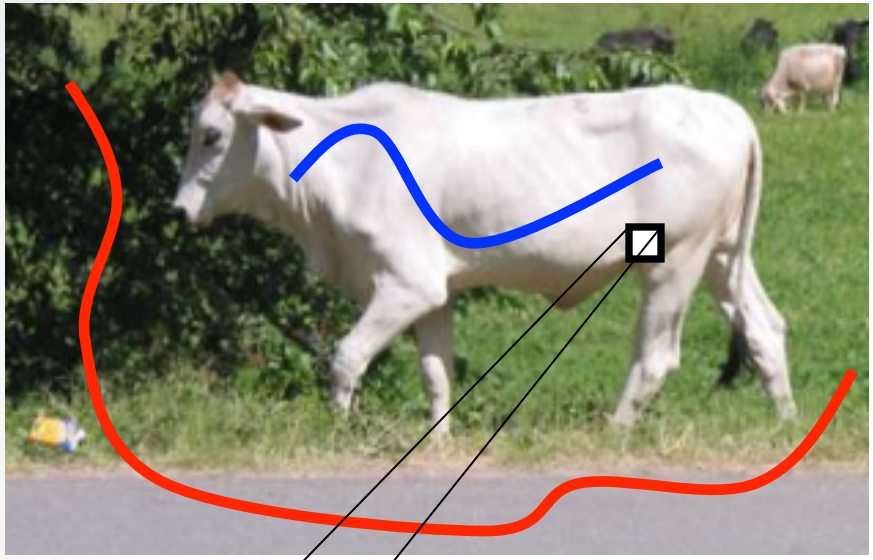
Graph G = (V,E)

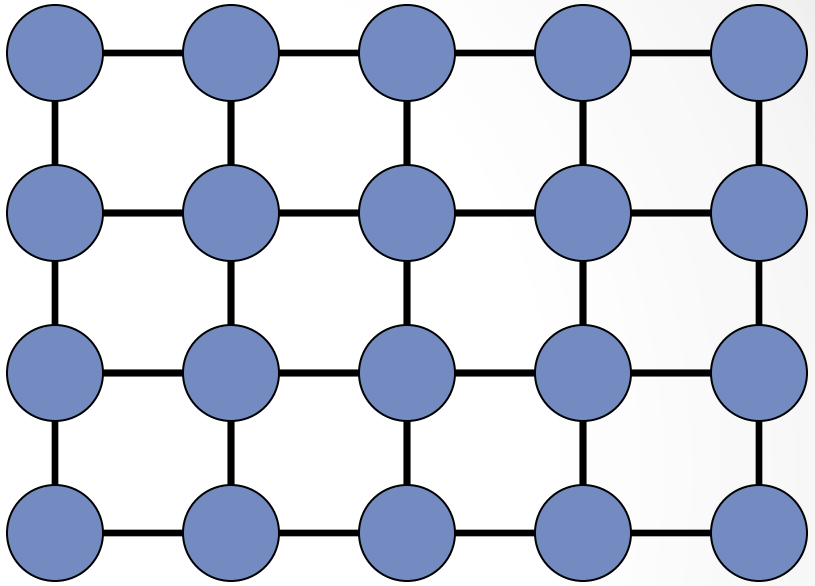
Each vertex corresponds to a pixel

Edges define a 4-neighbourhood *grid* graph

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

Binary Image Segmentation



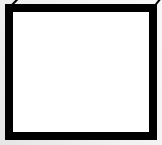


Object - white, Background - green/grey

Graph G = (V,E)

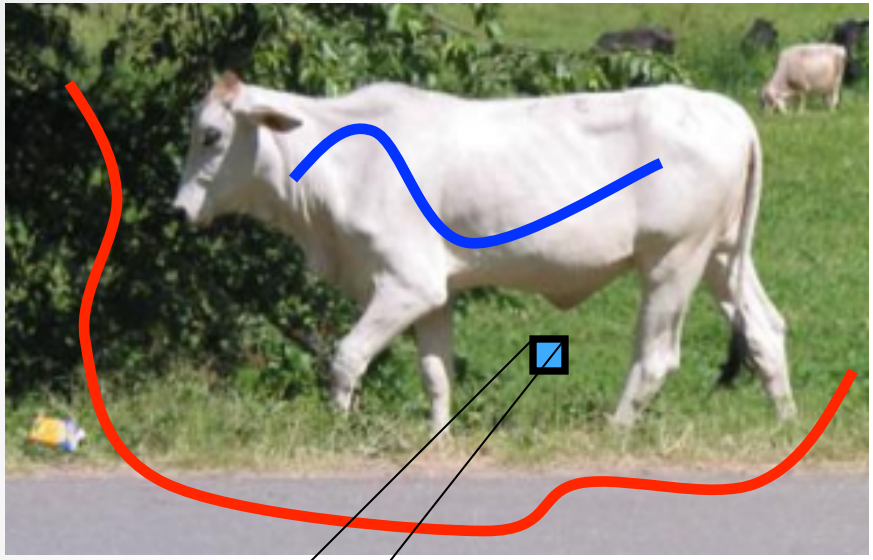
Cost of a labelling f : V ➔ L

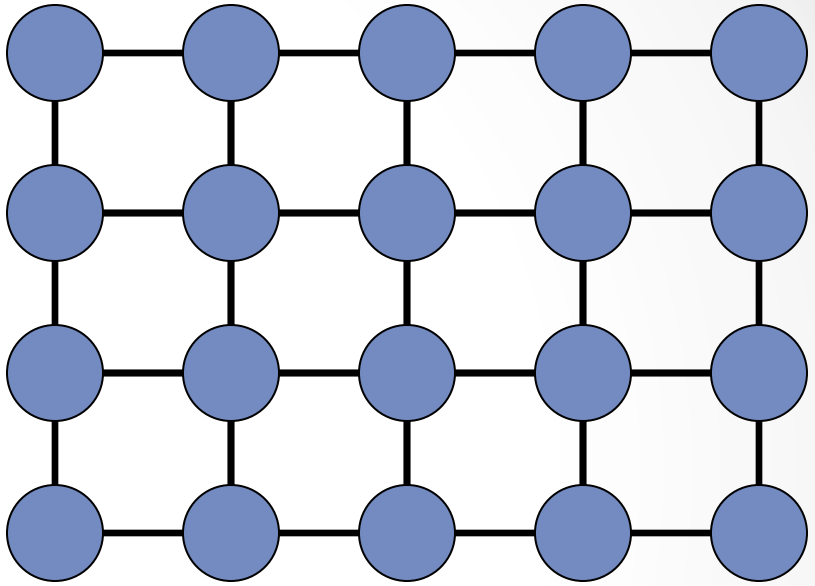
Per Vertex Cost



Cost of label ‘obj’ low Cost of label ‘bkg’ high

Binary Image Segmentation



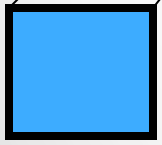


Object - white, Background - green/grey

Graph G = (V,E)

Cost of a labelling f : V ➔ L

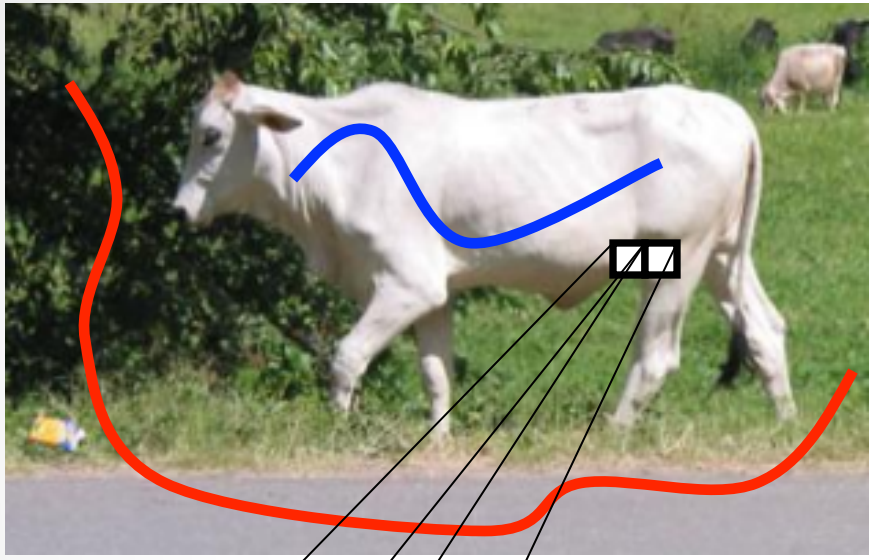
Per Vertex Cost

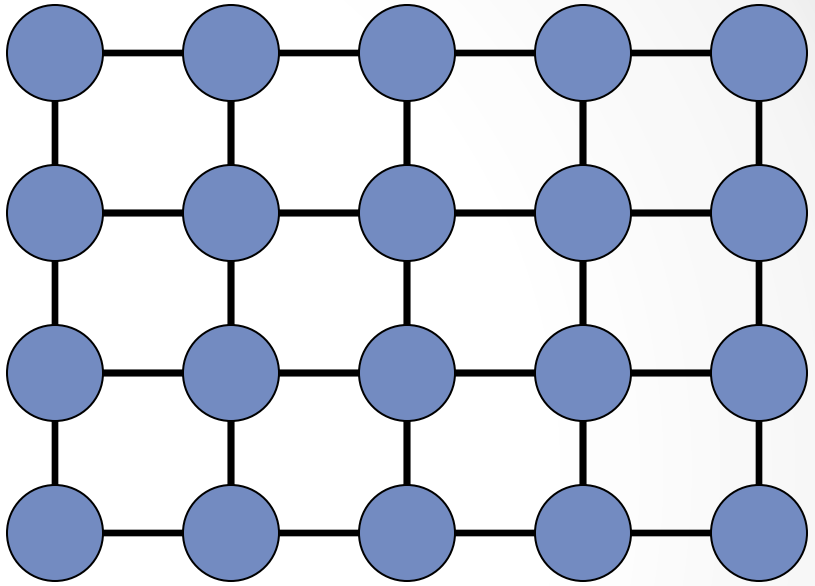


Cost of label ‘obj’ high Cost of label ‘bkg’ low

UNARY COST

Binary Image Segmentation



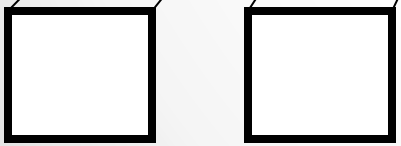


Object - white, Background - green/grey

Graph G = (V,E)

Cost of a labelling f : V ➔ L

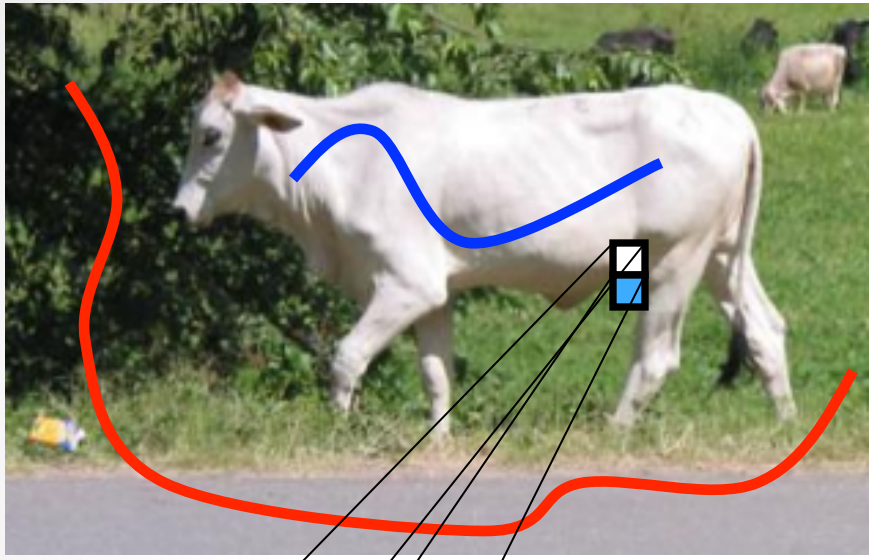
Per Edge Cost

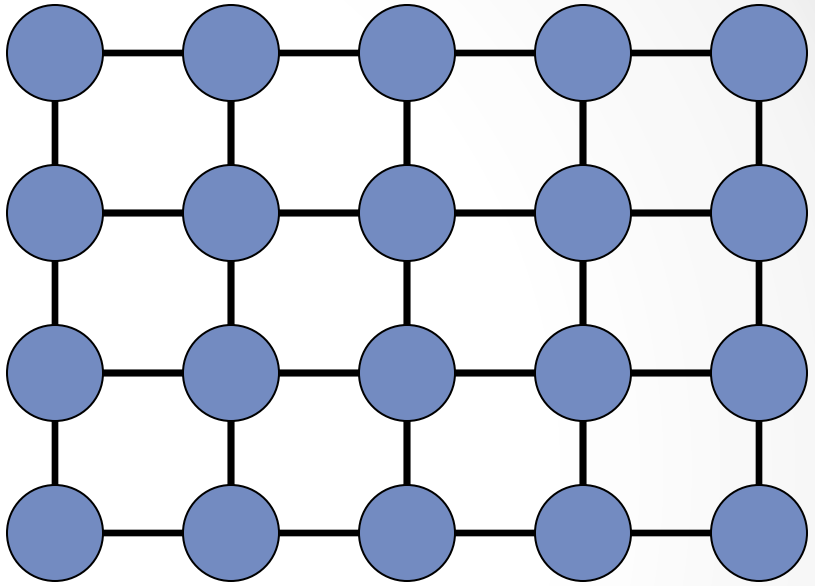


Cost of same label low

Cost of different labels high

Binary Image Segmentation



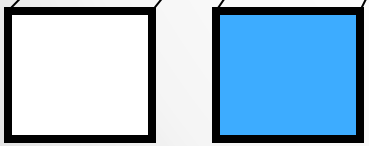


Object - white, Background - green/grey

Graph G = (V,E)

Cost of a labelling f : V ➔ L

Per Edge Cost

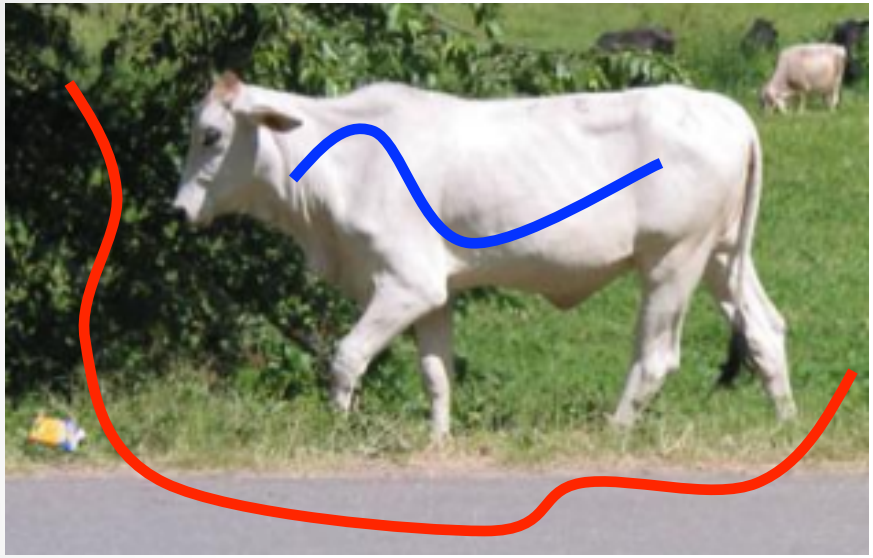


Cost of same label high

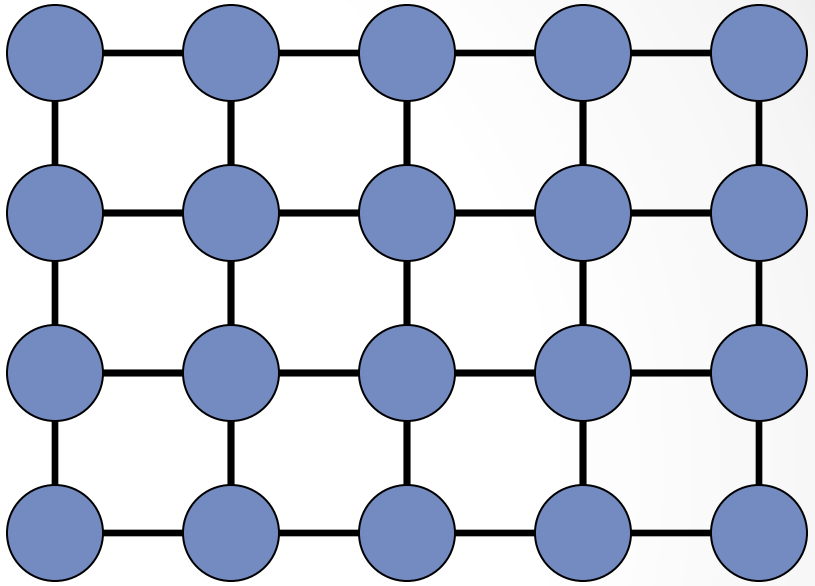
Cost of different labels low

PAIRWISE COST

Binary Image Segmentation



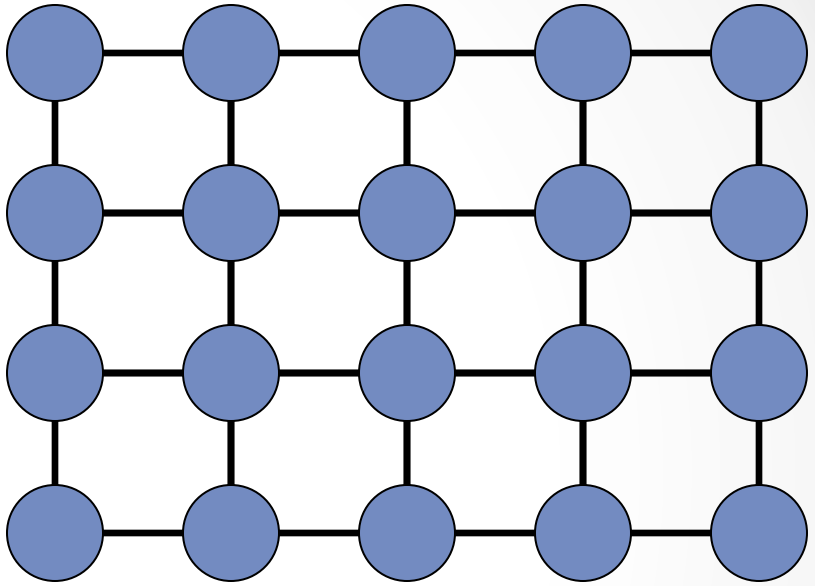
Object - white, Background - green/grey



Graph G = (V,E)

Problem: Find the labeling with minimum cost f*

Binary Image Segmentation



Graph G = (V,E)

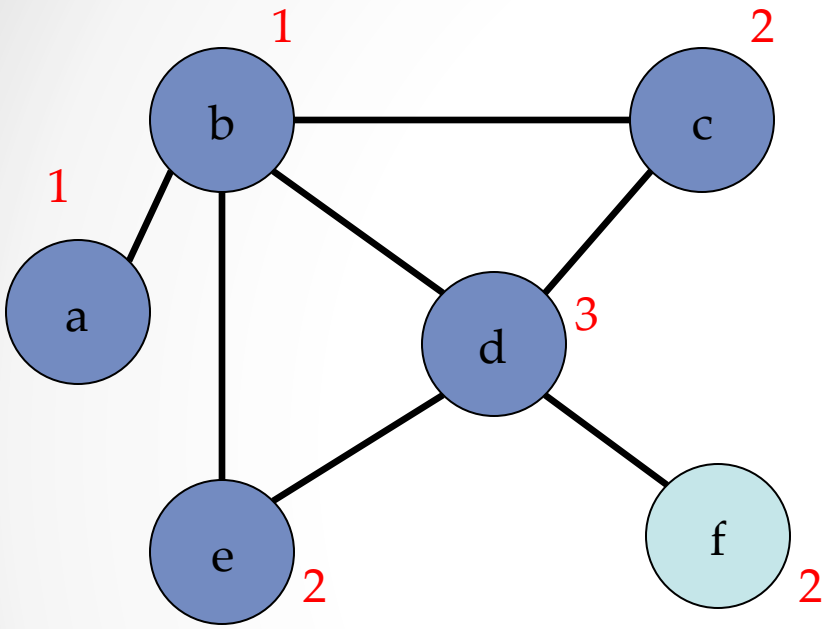
$L = \{fg, bg\}$

Vertex corresponds to a pixel

Edges define grid graph

Problem: Find the labeling with minimum cost f*

The General Problem



Graph G = (V, E)

Discrete label set L = {1,2,…,h}

Assign a label to each vertex

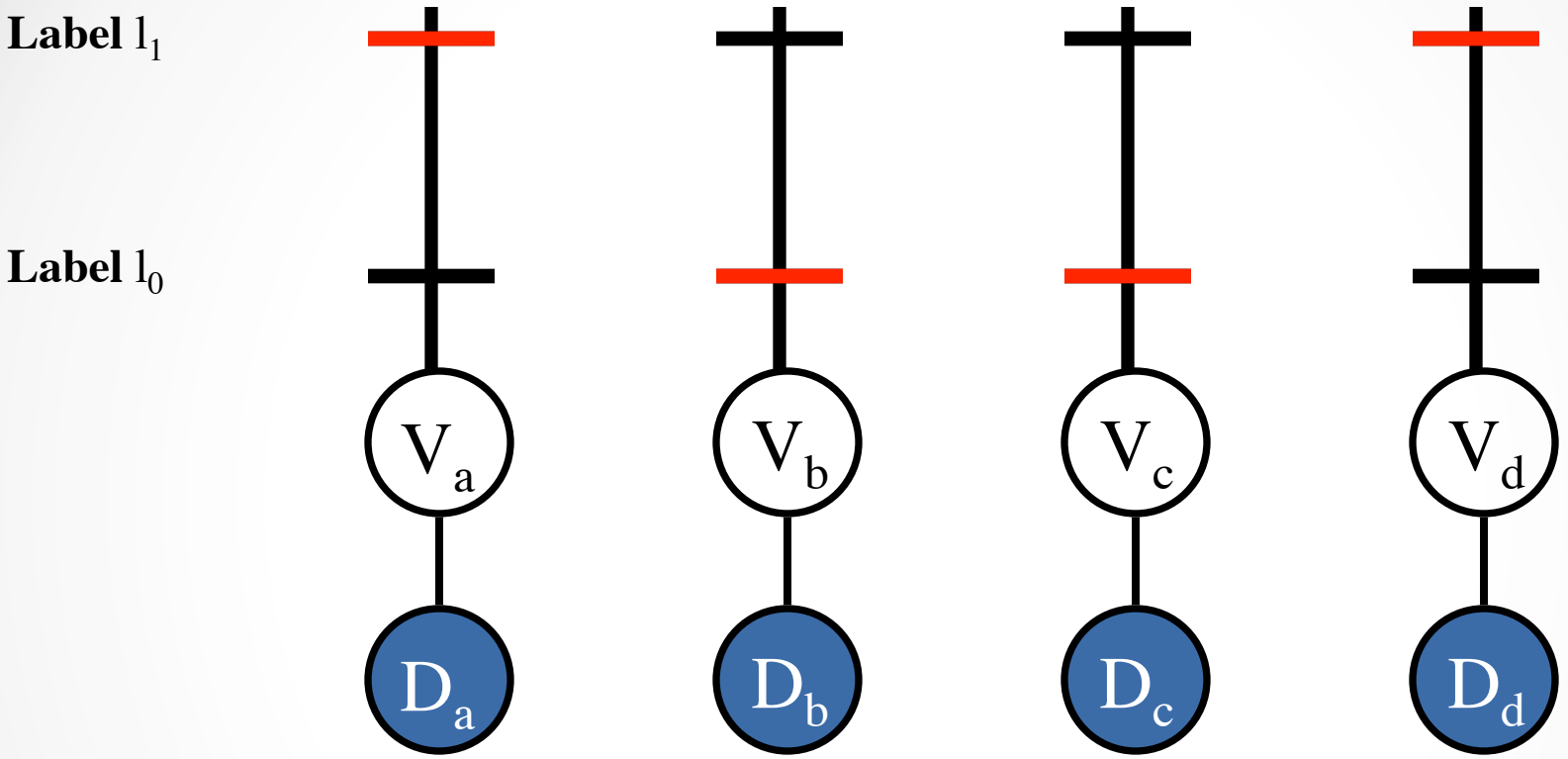
f: V ➜ L

Cost of a labelling Q(f)

Unary Cost Pairwise Cost

Find f* = arg min Q(f)

Formulation: Energy Function

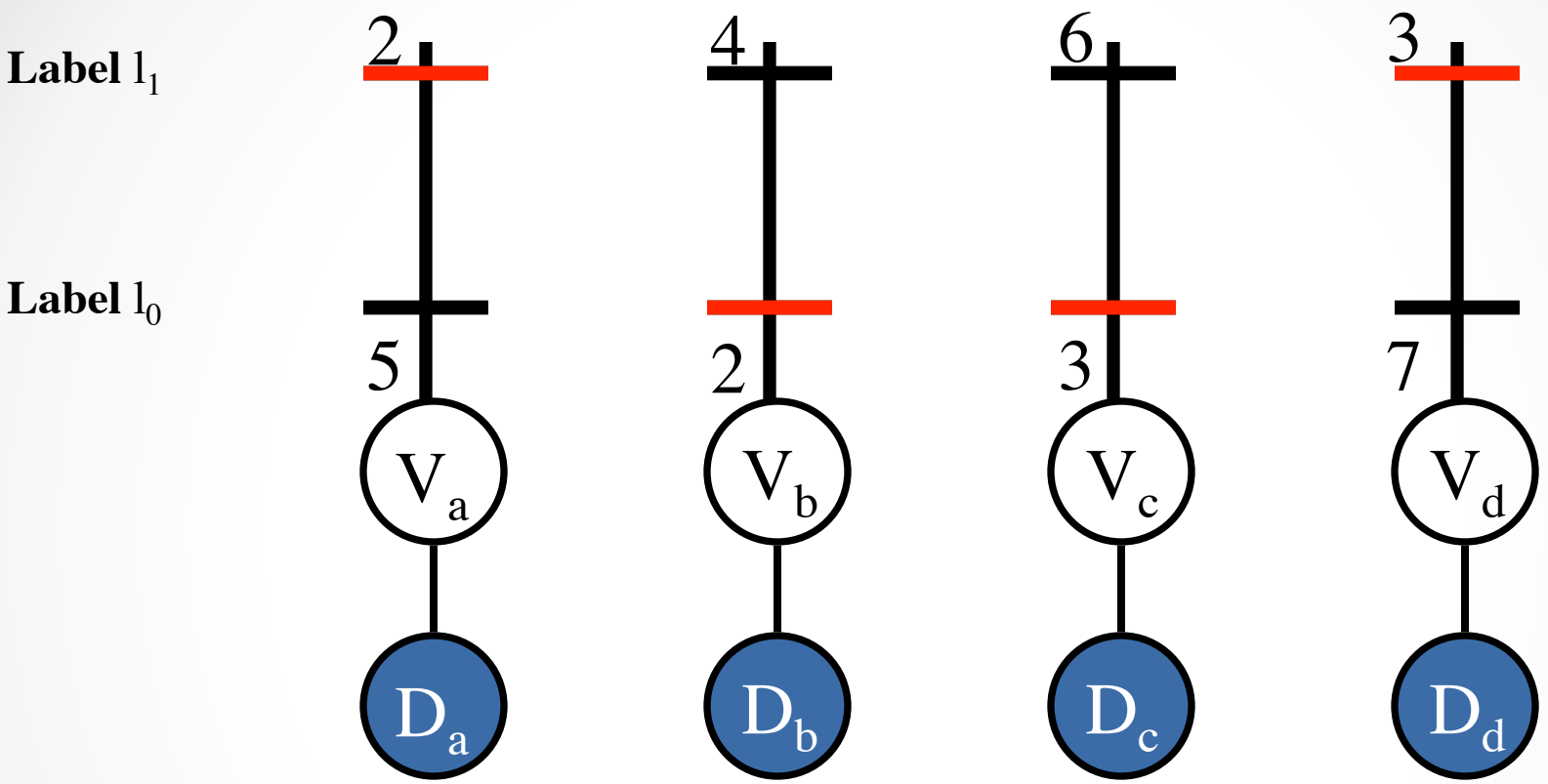


Random Variables V = {V_a, V_b, ….}

Labels L = {l_0, l_1, ….} Data D

Labelling f: {a, b, …. } ➔ {0,1, …}

Energy Function



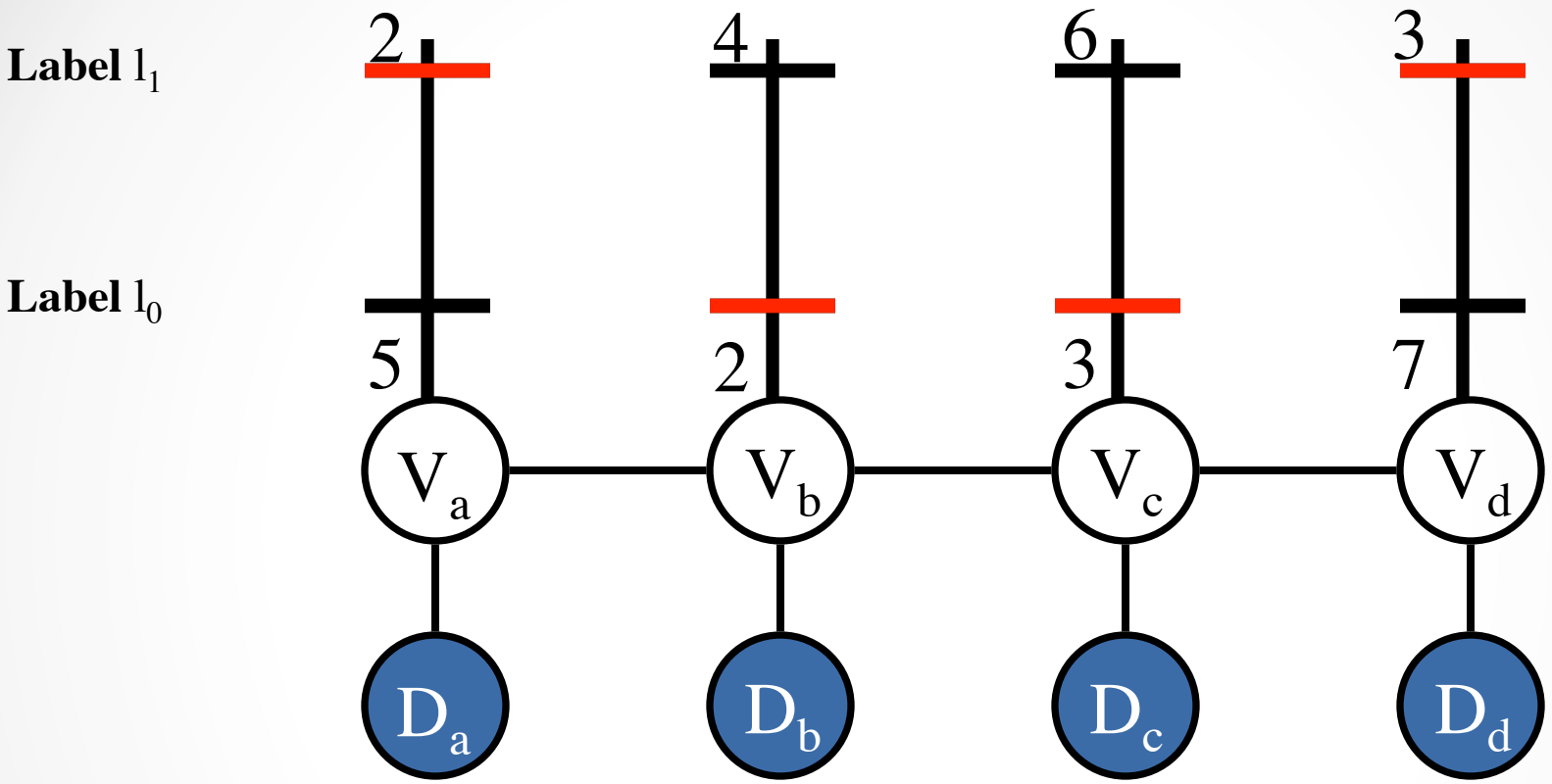
$$Q(f) = \sum_a \theta_{a;f(a)}$$

Unary Potential

Easy to minimize

Neighbourhood

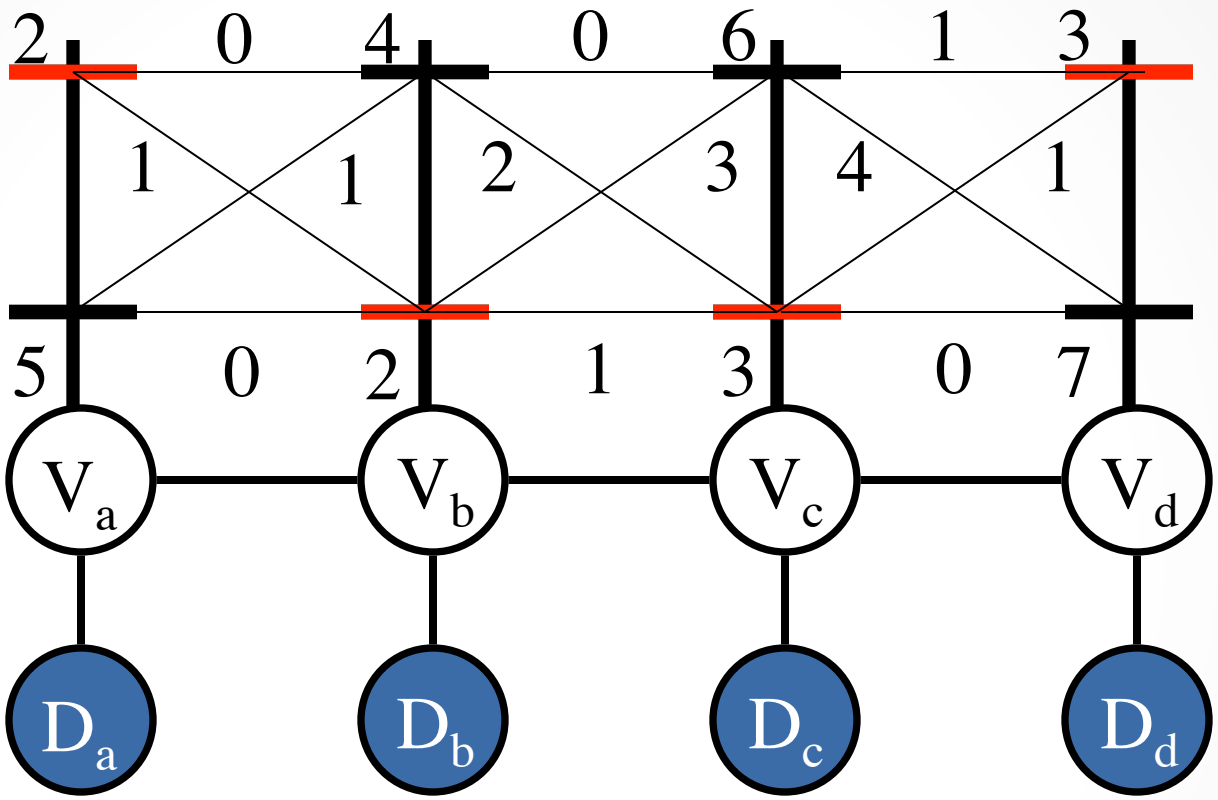
Energy Function



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

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

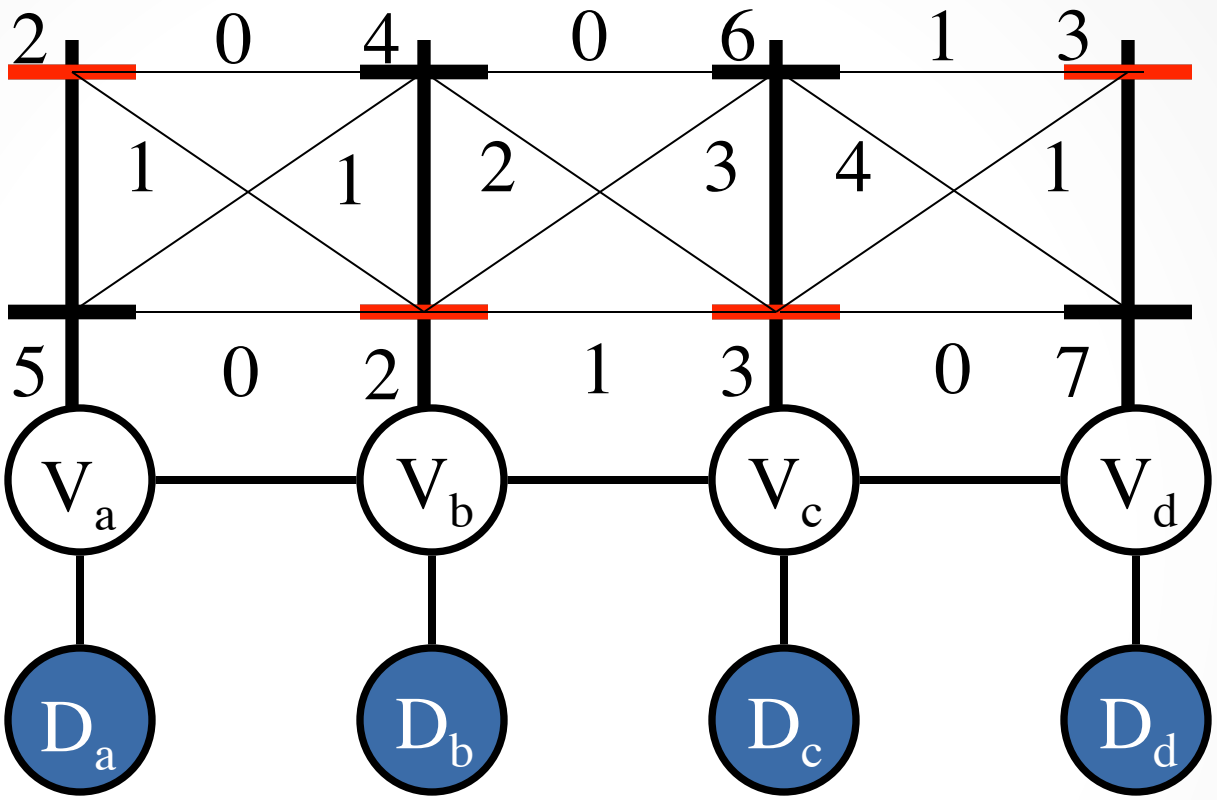
Energy Function



Pairwise Potential

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

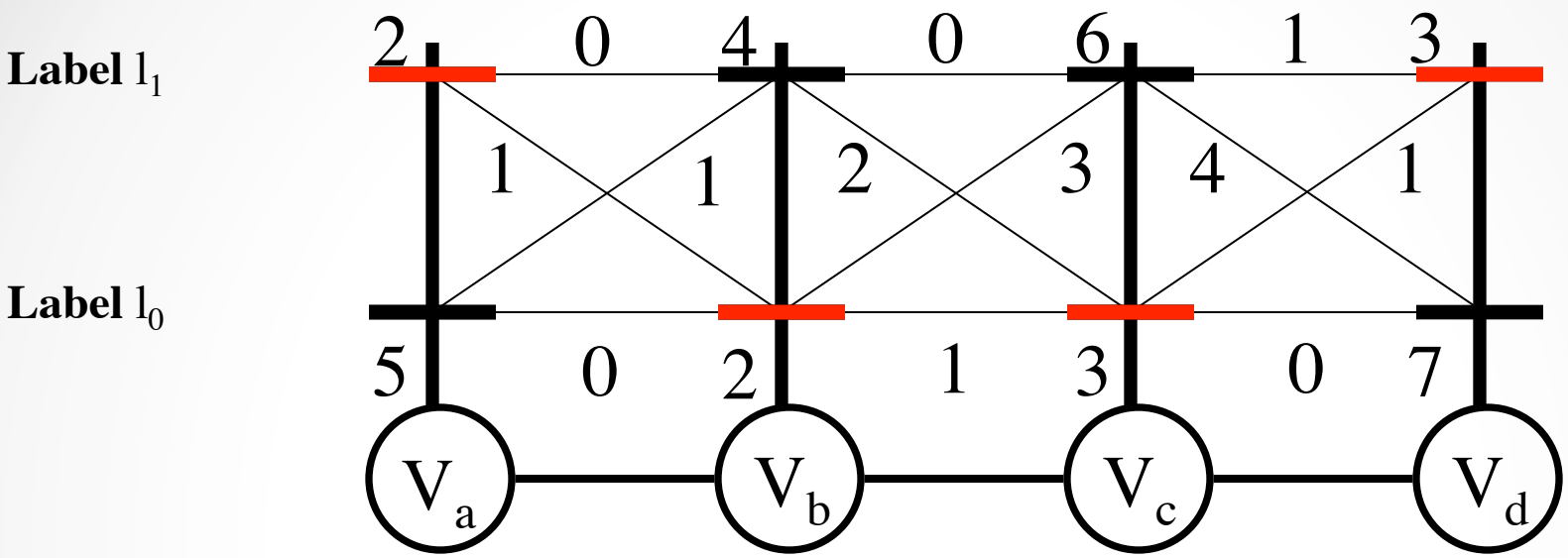
Energy Function



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

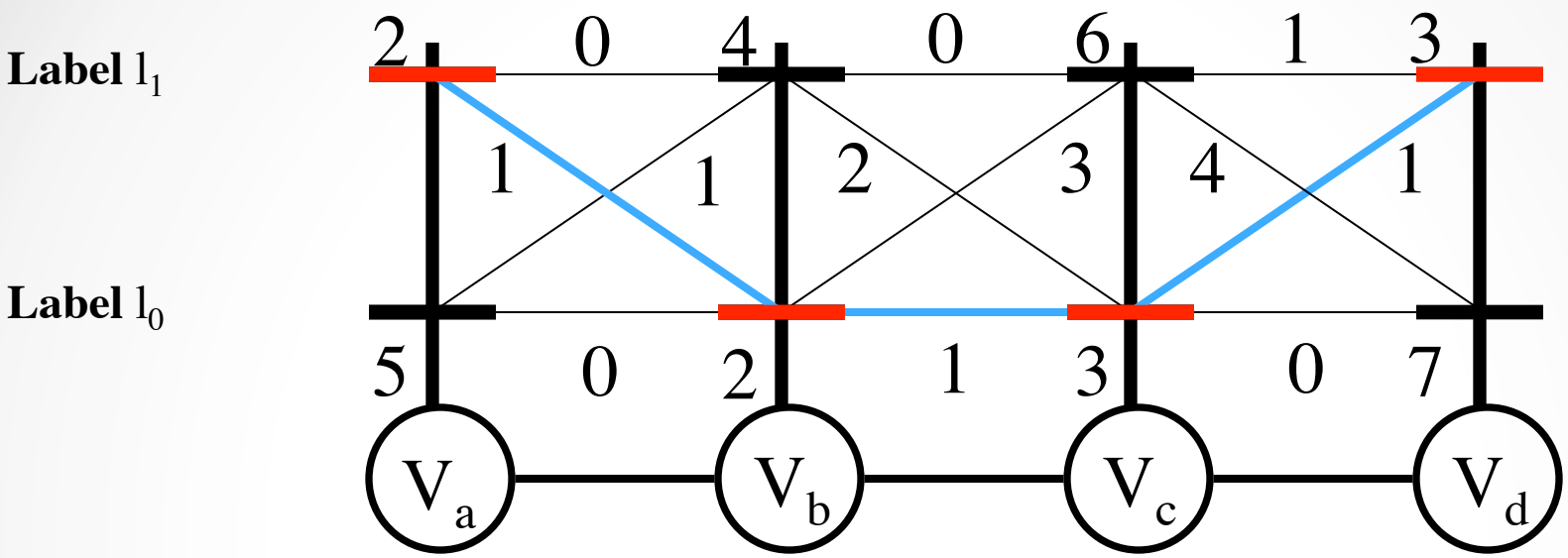
Parameter

MAP Estimation



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

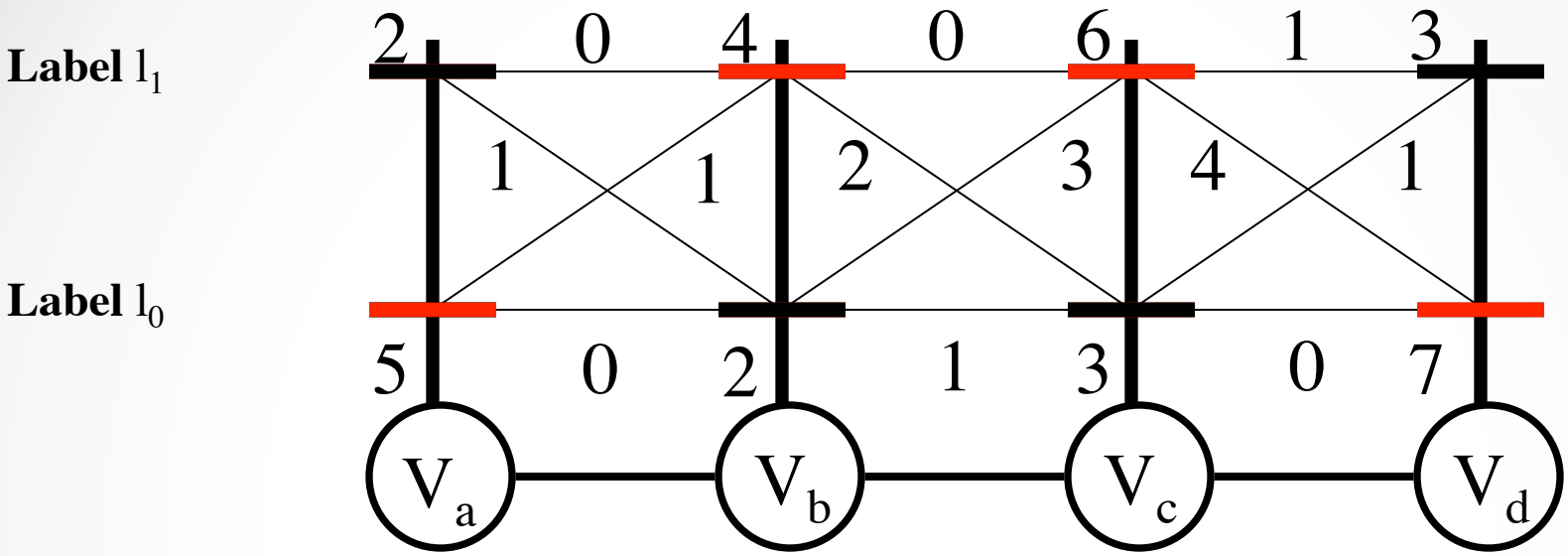
MAP Estimation



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

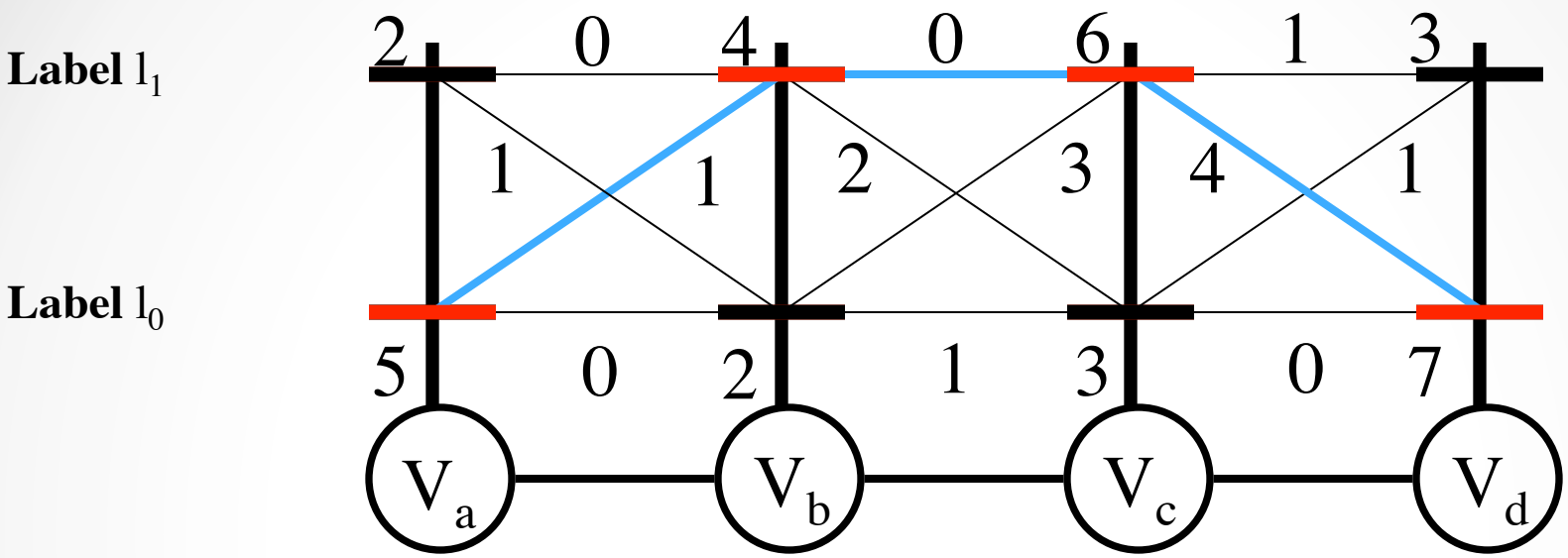
2 + 1 + 2 + 1 + 3 + 1 + 3 = 13

MAP Estimation



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

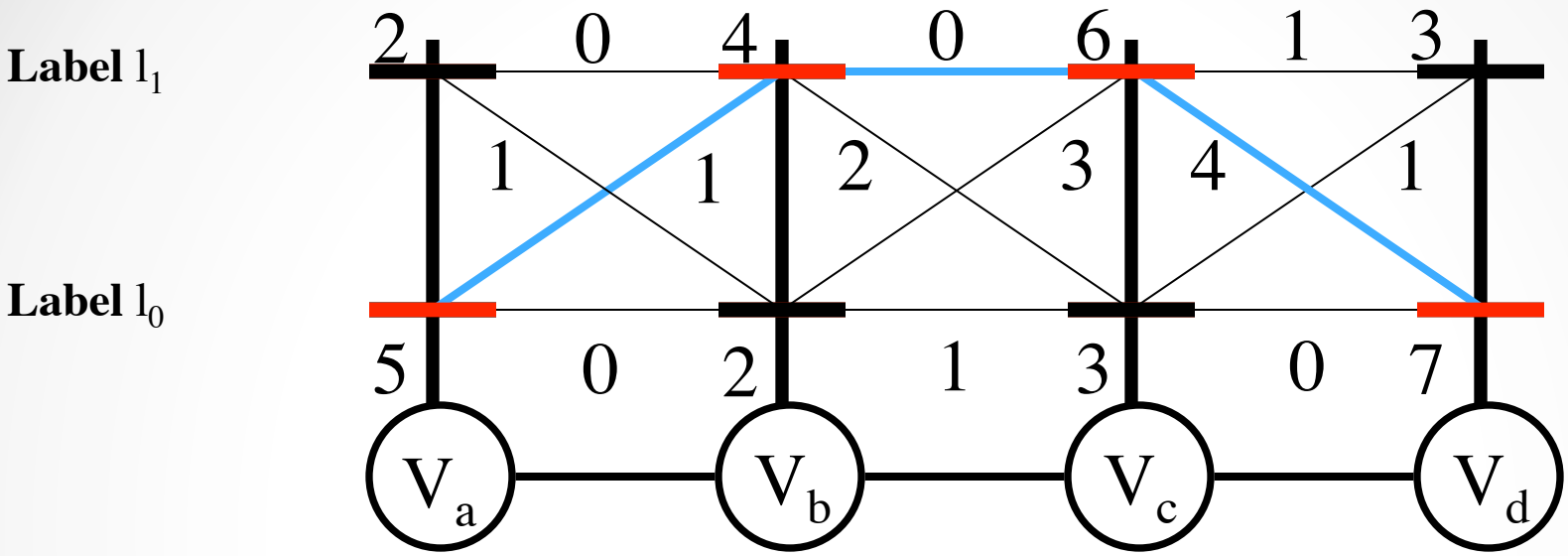
MAP Estimation



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

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

MAP Estimation



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

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

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

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 ≈ 320 * 480 = 153600

Can we do better than brute-force?

MAP Estimation is NP-hard !!

Computational Complexity

Segmentation

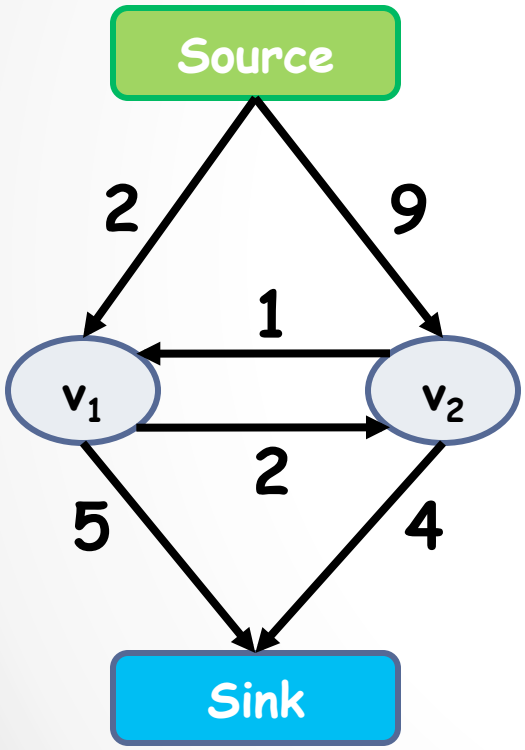
$2^{|V|}$

$|V|$ = number of pixels ≈ 320 * 480 = 153600

Exact algorithms do exist for special cases

Good approximate algorithms for general case

The st-Mincut Problem



Graph (V, E, C)

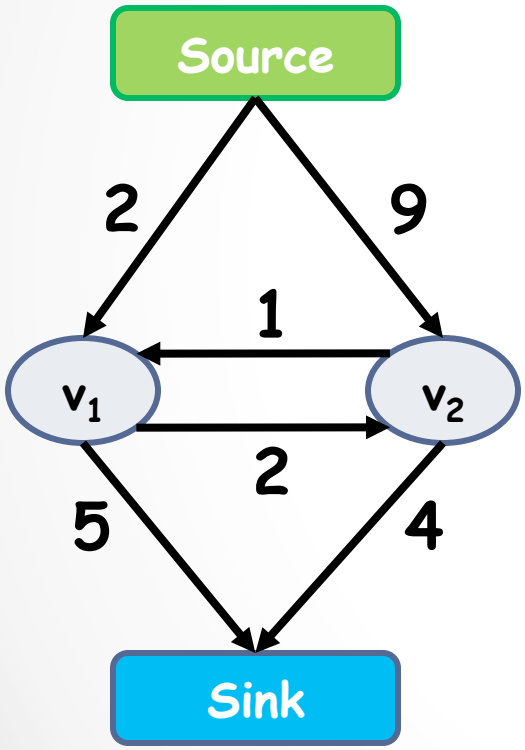
Vertices $V = \{v_1, v_2 \ldots v_n\}$

Edges $E = \{(v_1, v_2) \ldots.\}$

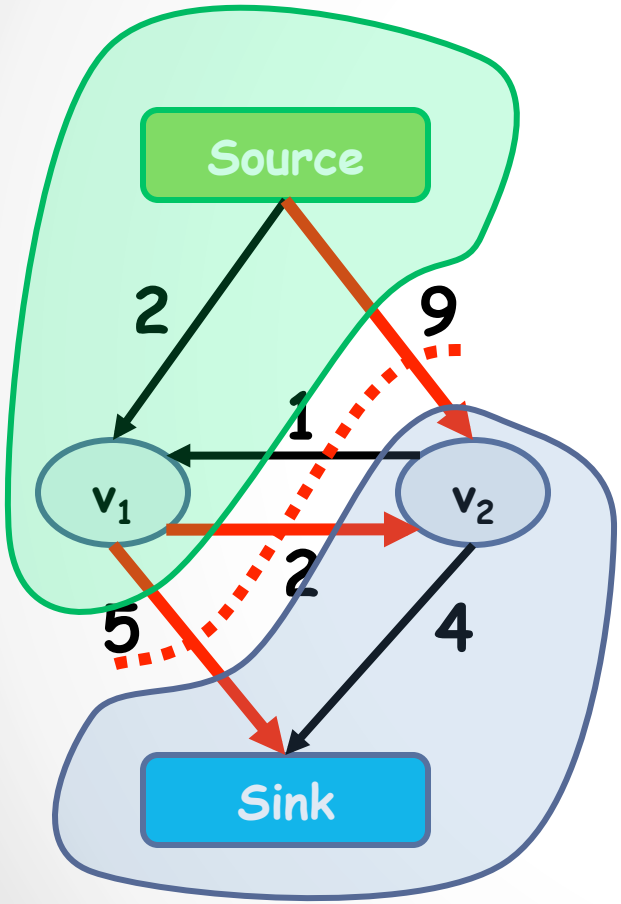
Costs $C = \{c_{(1, 2)} \ldots.\}$

The Solution: st-Mincut

What is an st-cut?



The st-Mincut Problem



What is an st-cut?

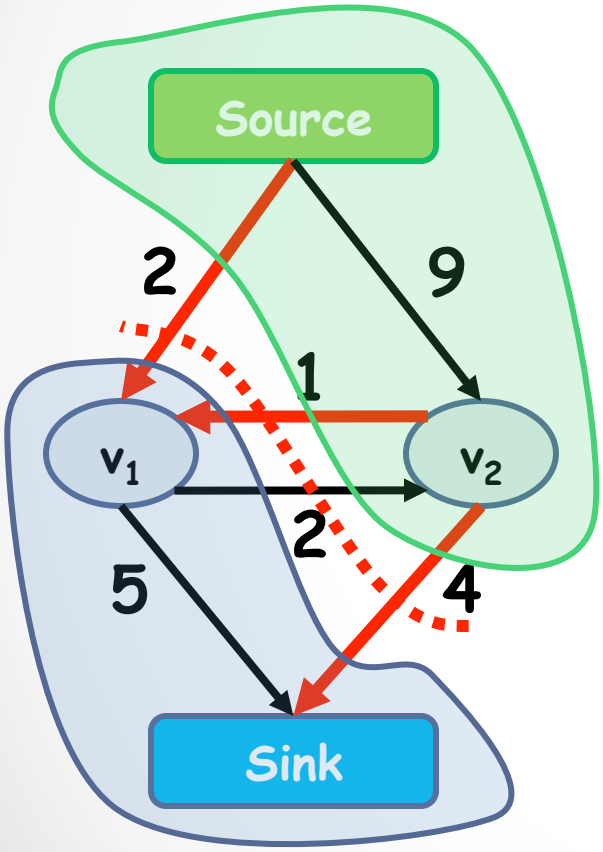
An st-cut (S,T) divides the nodes between source and sink.

What is the cost of an st-cut?

Sum of cost of all edges going from S to T

5 + 2 + 9 = 16

The st-Mincut Problem



2 + 1 + 4 = 7

What is an st-cut?

An st-cut (S,T) divides the nodes between source and sink.

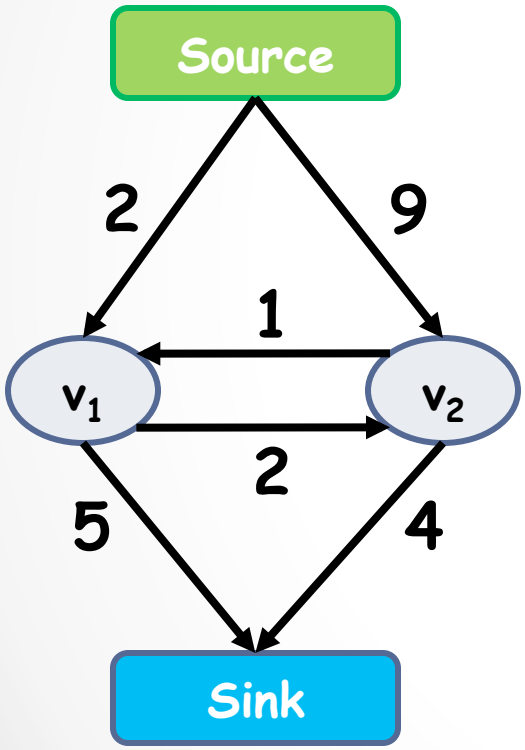
What is the cost of an st-cut?

Sum of cost of all edges going from S to T

What is the st-mincut?

st-cut with the minimum cost

How to Compute the st-Mincut?



Solve the dual **maximum flow** problem

Compute the maximum flow between Source and Sink

Constraints

Edges: Flow < Capacity

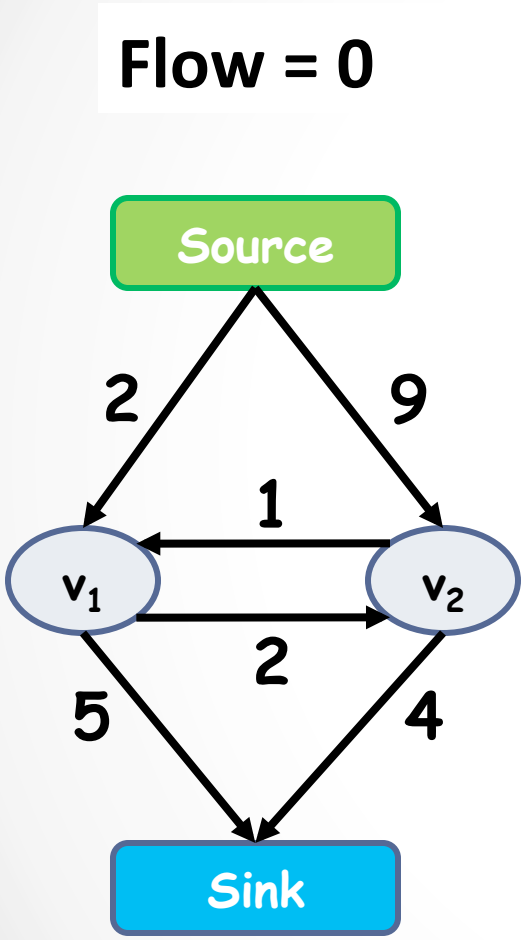
Nodes: Flow in = Flow out

Min-cut/Maxflow Theorem

In every network, the maximum flow equals the cost of the st-mincut

Dantzig, Dinitz, Edmonds & Karp, Ford & Fulkerson, Goldberg & Tarjan, Karzanov, …

Maxflow Algorithms

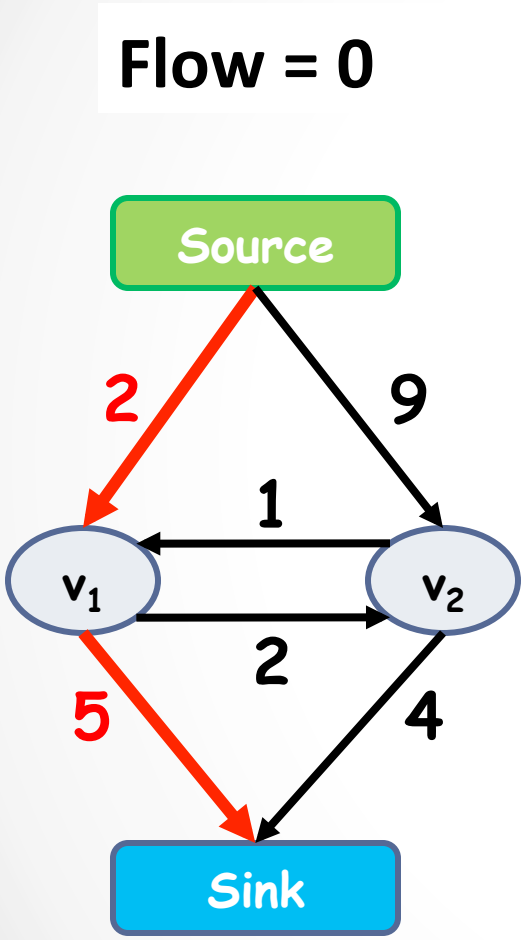


Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Maxflow Algorithms

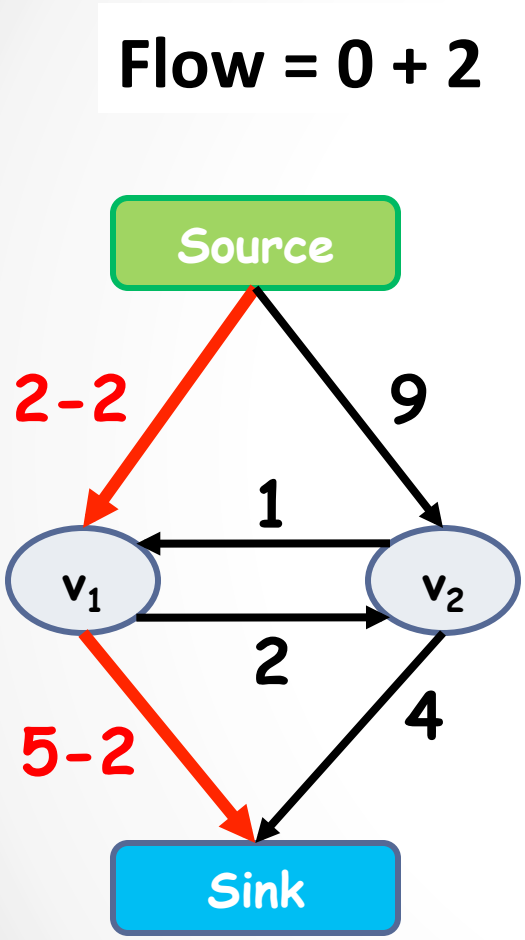


Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Maxflow Algorithms

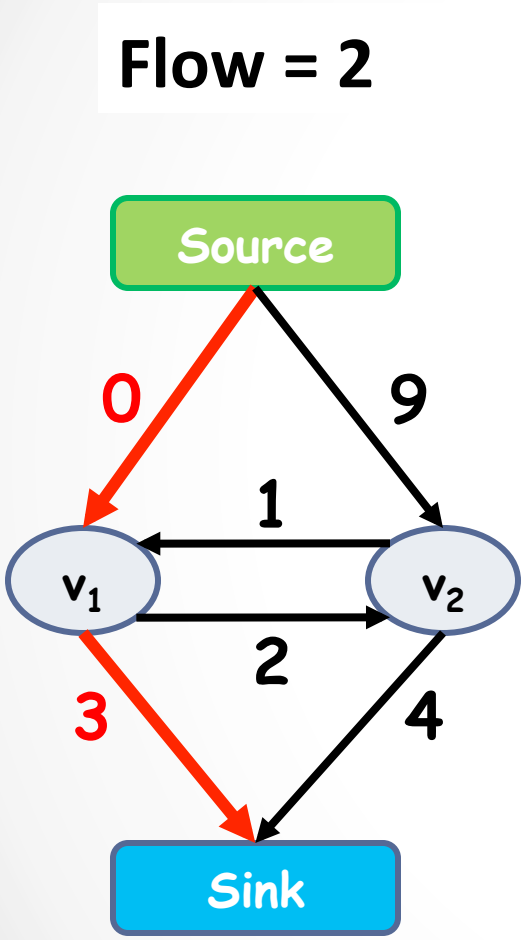


Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Maxflow Algorithms

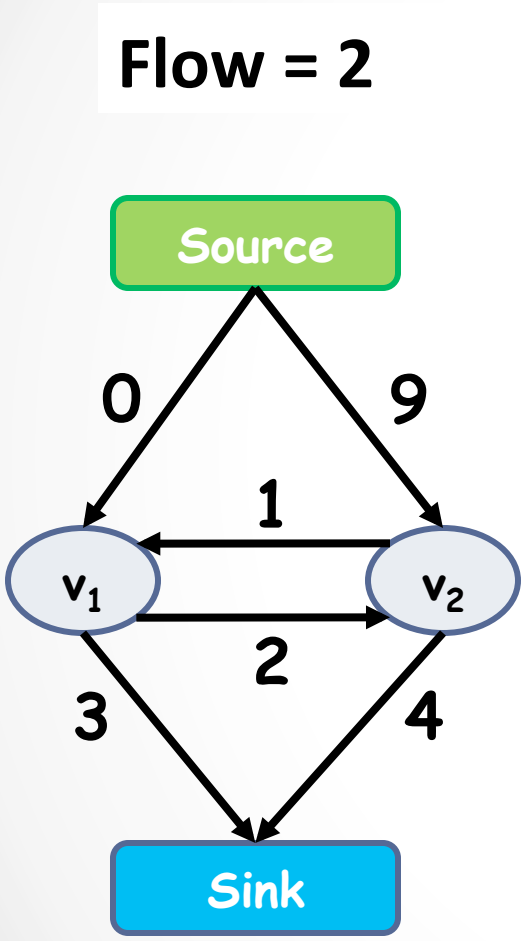


Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Maxflow Algorithms

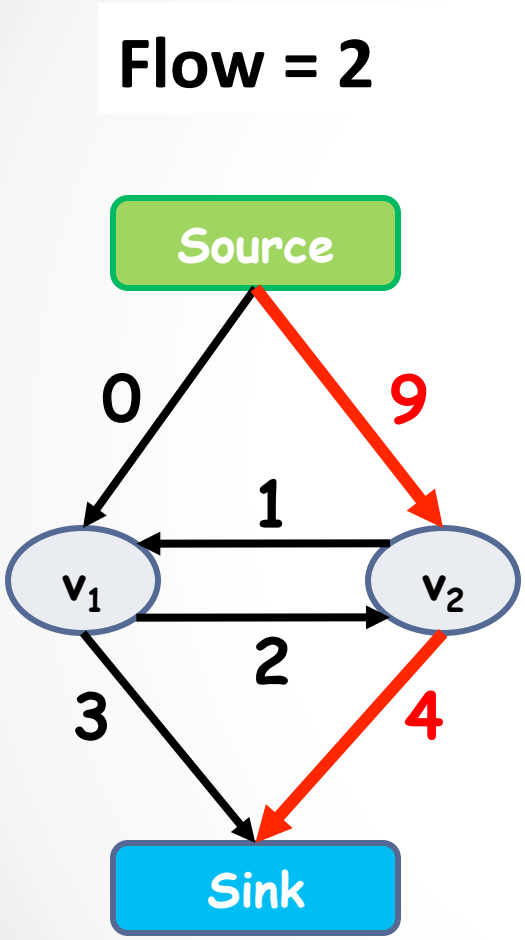


Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Maxflow Algorithms



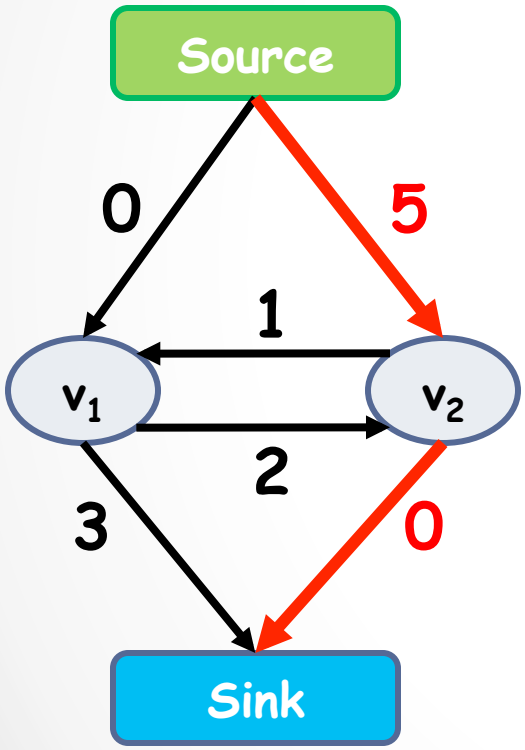
Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Maxflow Algorithms

Flow = 2 + 4

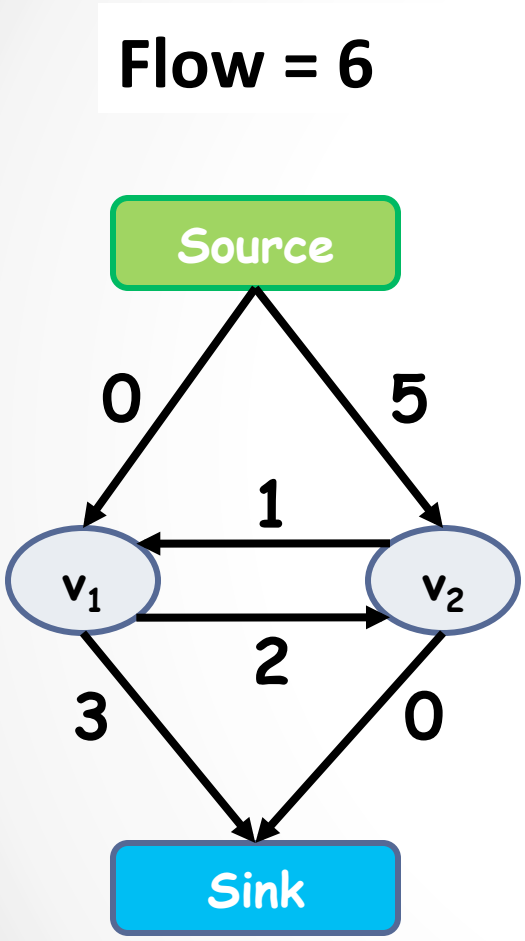


Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Maxflow Algorithms

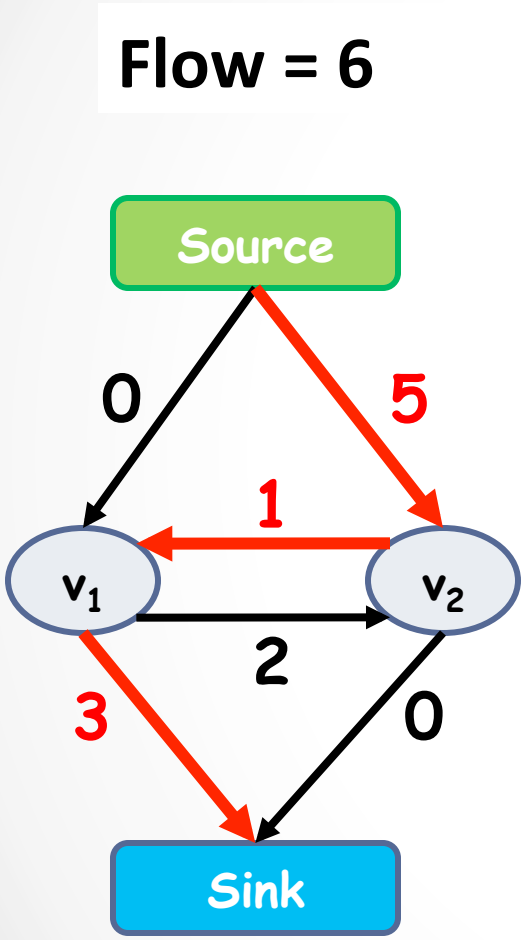


Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Maxflow Algorithms



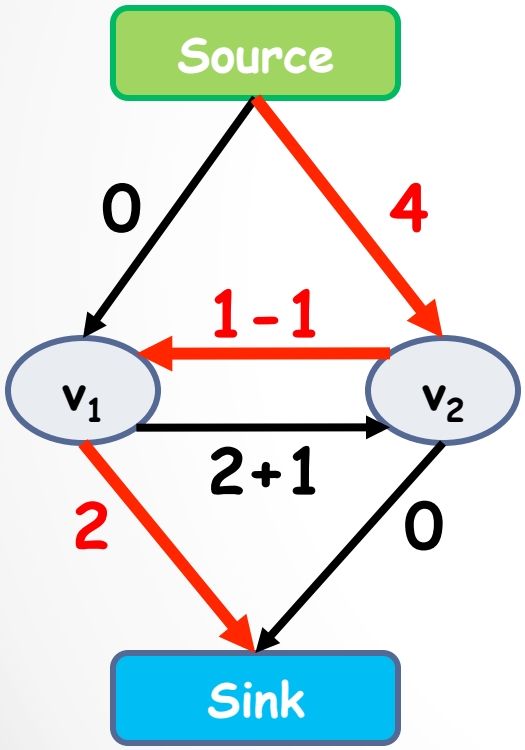
Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Maxflow Algorithms

Flow = 6 + 1

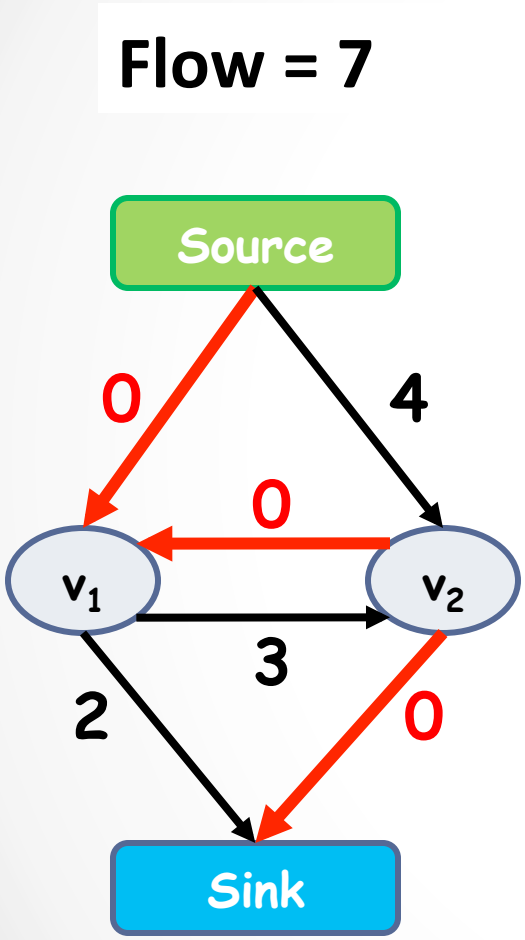


Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Maxflow Algorithms

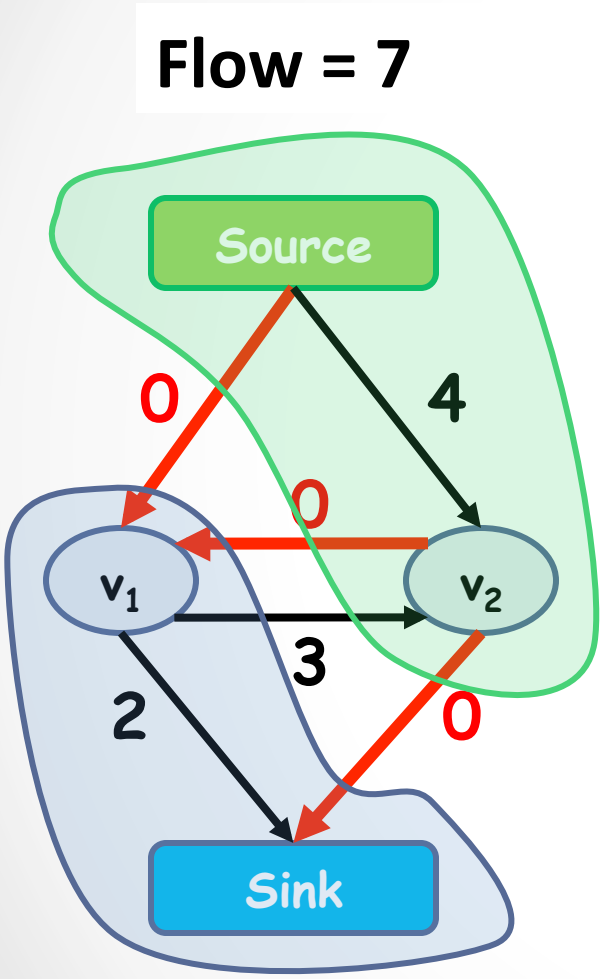


Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Maxflow Algorithms



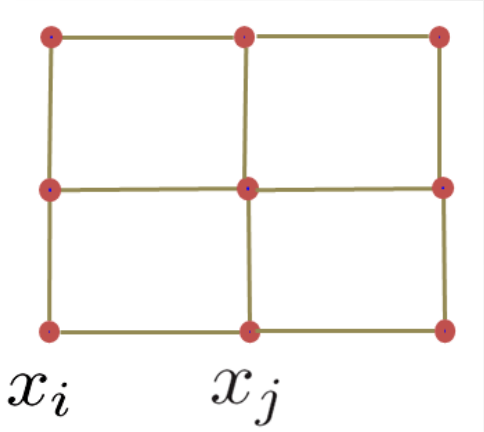
Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

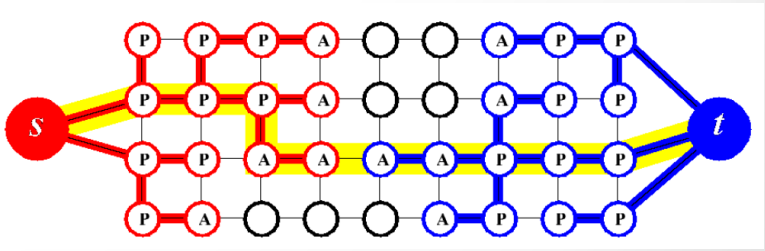
Maxflow in Computer Vision

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



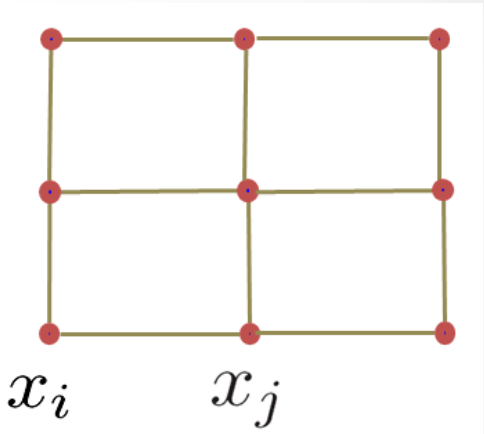
- Dual search tree augmenting path algorithm

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



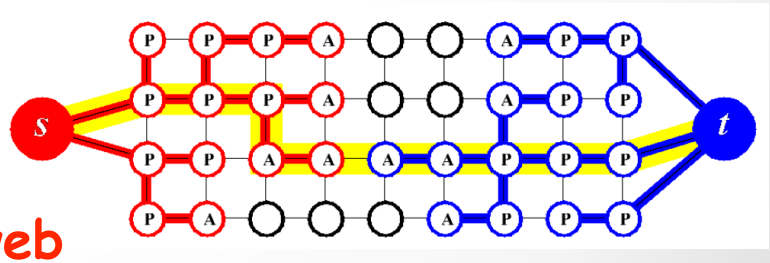
Maxflow in Computer Vision

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



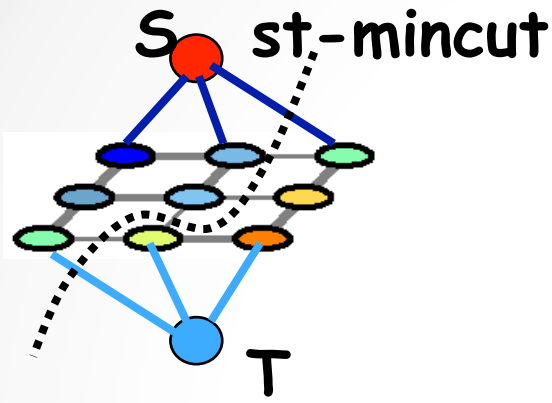
- Dual search tree augmenting path algorithm

  [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



- http://pub.ist.ac.at/~vnk/software.html

St-mincut and Energy Minimization



Minimizing a Qudratic Pseudoboolean function E(x)

Functions of boolean variables

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

$$E(x) = \sum_i c_i x_i + \sum_{i,j} c_{ij} x_i(1-x_j)$$

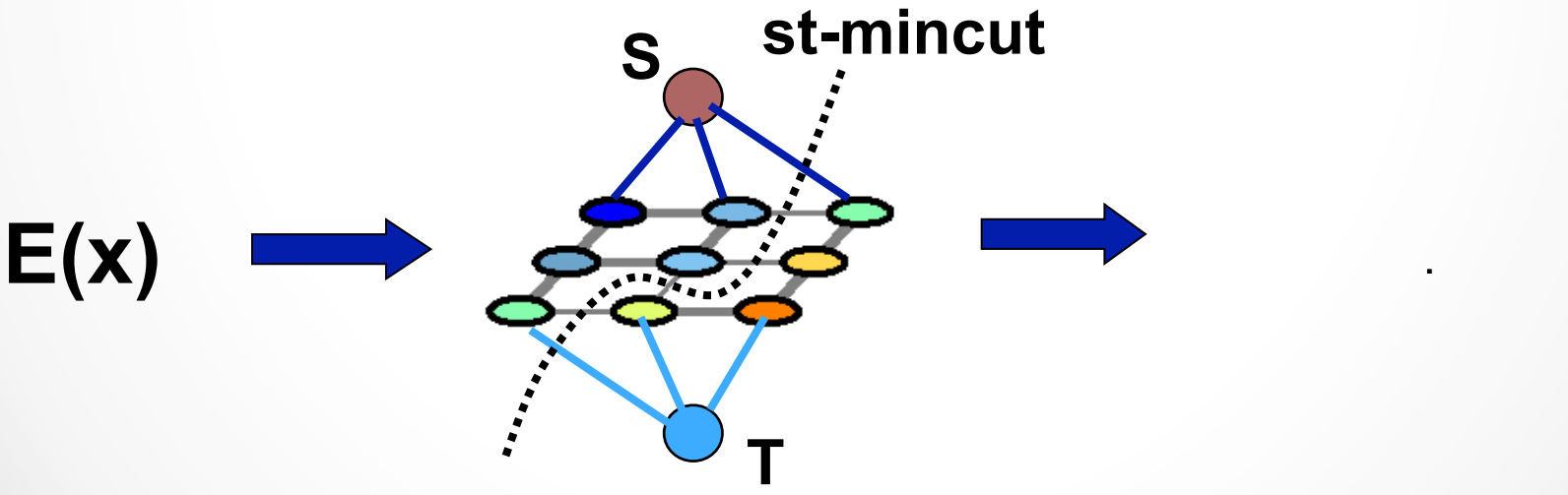
$c_{ij} \geq 0$

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

How does this Work?

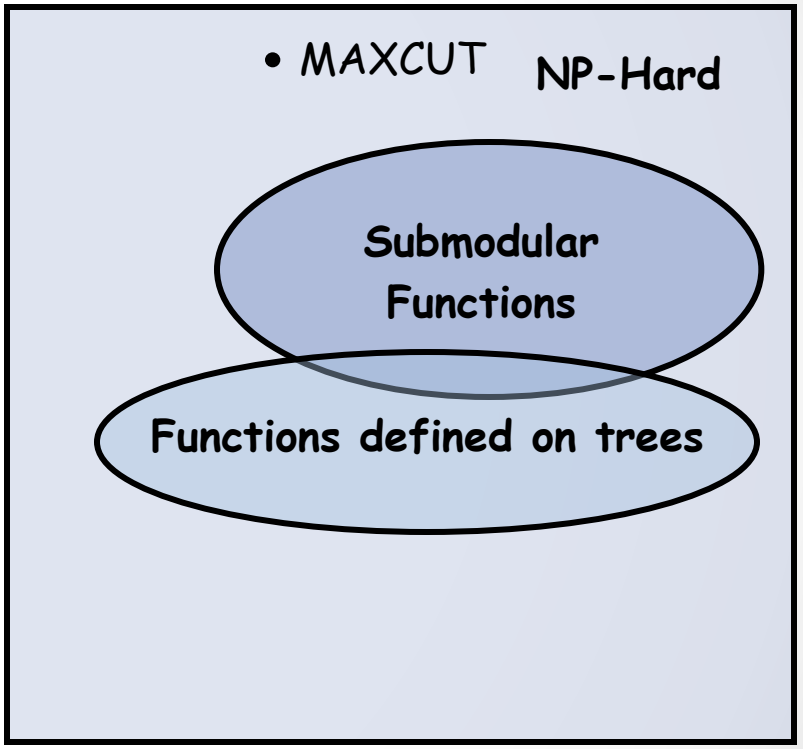
Construct a graph such that:

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)



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

GrabCut

Interactive Foreground Extraction using Iterated Graph Cuts

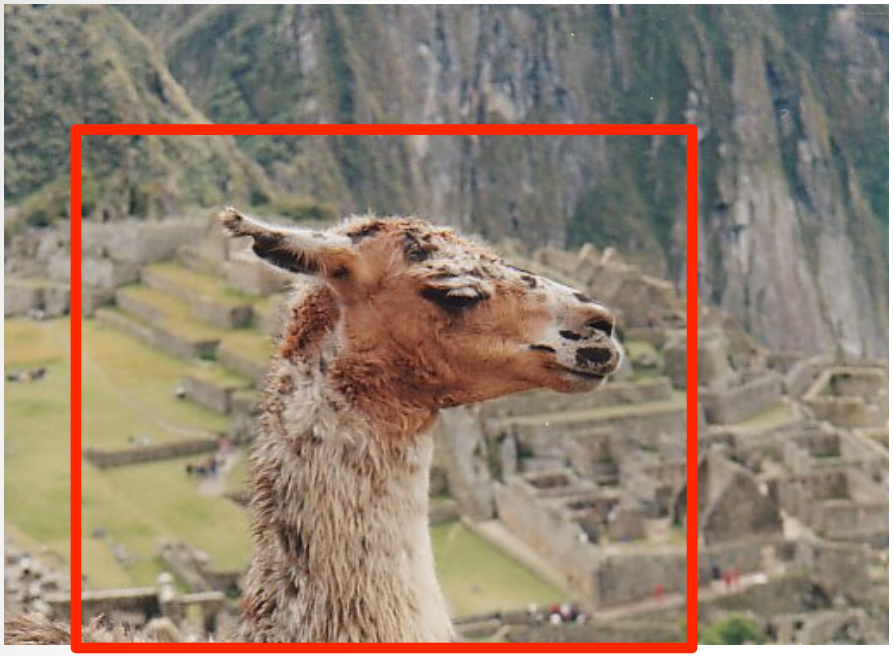
Carsten Rother

Vladimir Kolmogorov

Andrew Blake

Microsoft Research Cambridge-UK

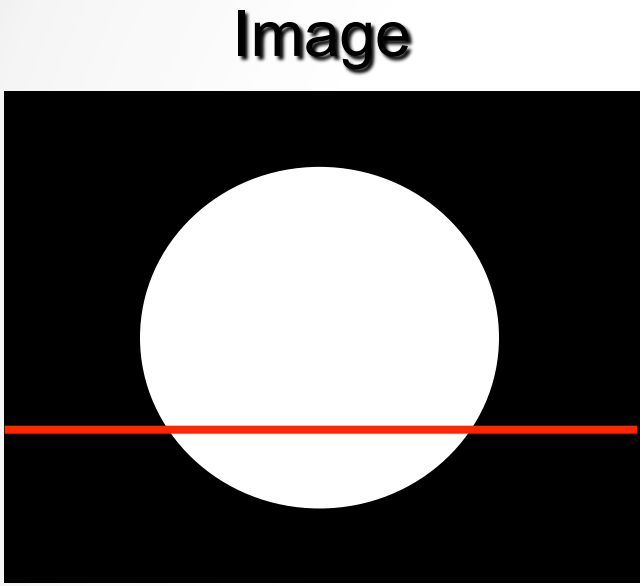
The Problem

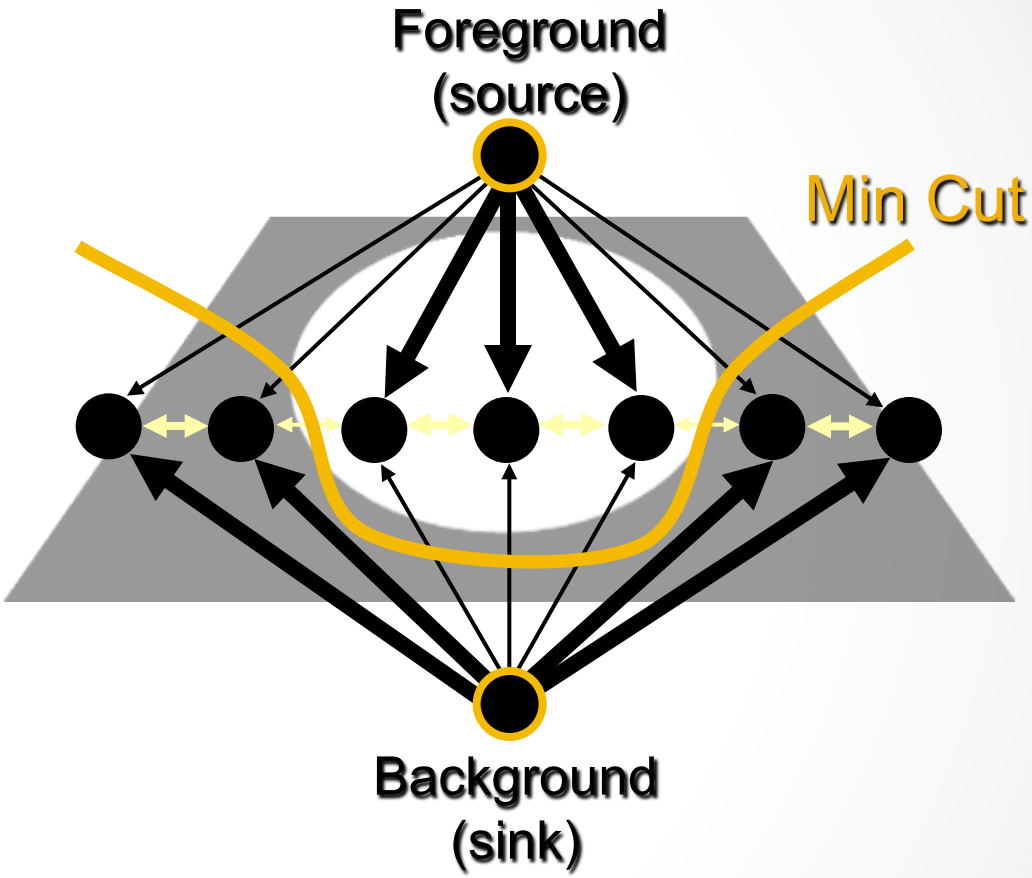


Fast &
Accurate ?

- Less user input: only rectangle
- Handle color
- Extract matte as post-process

Use Basic Graph Cut for Segmentation





Cut: separating source and sink; Energy: collection of edges

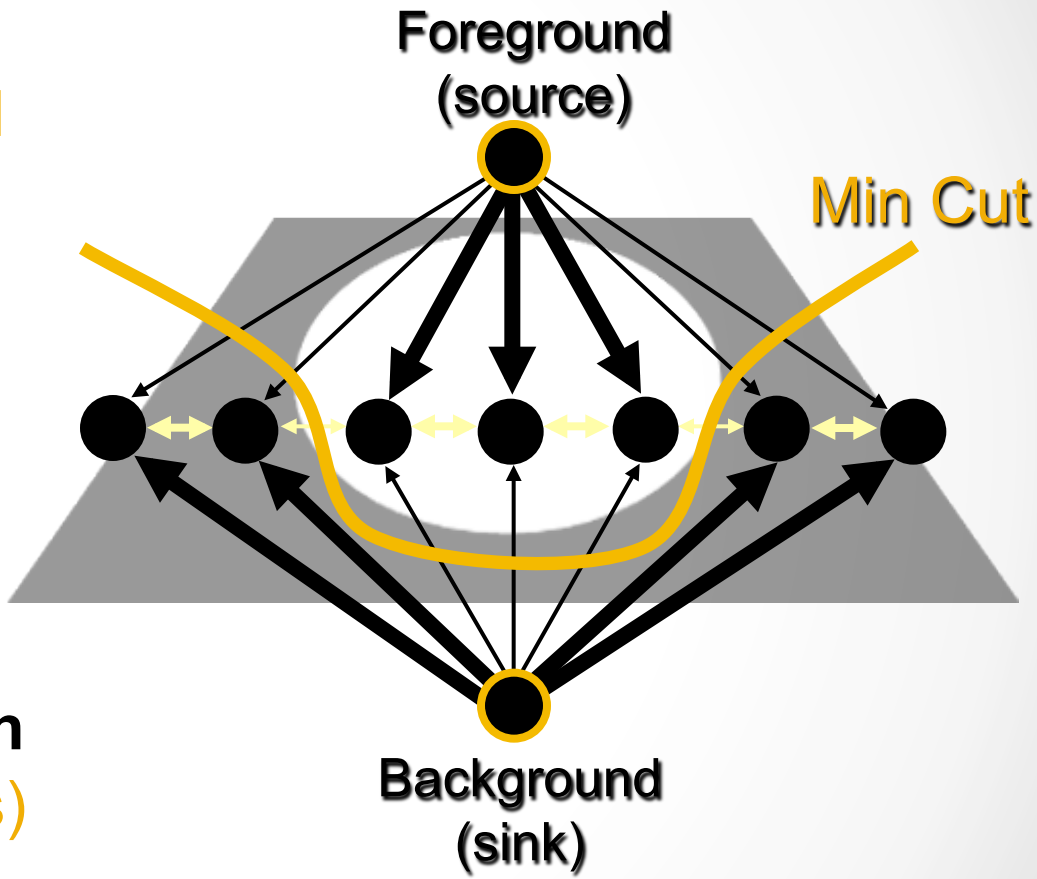
Min Cut: Global minimal enegry in polynomial time

Graph Cuts for Foreground Extraction

Assume we know foreground is **white** and background is **black**

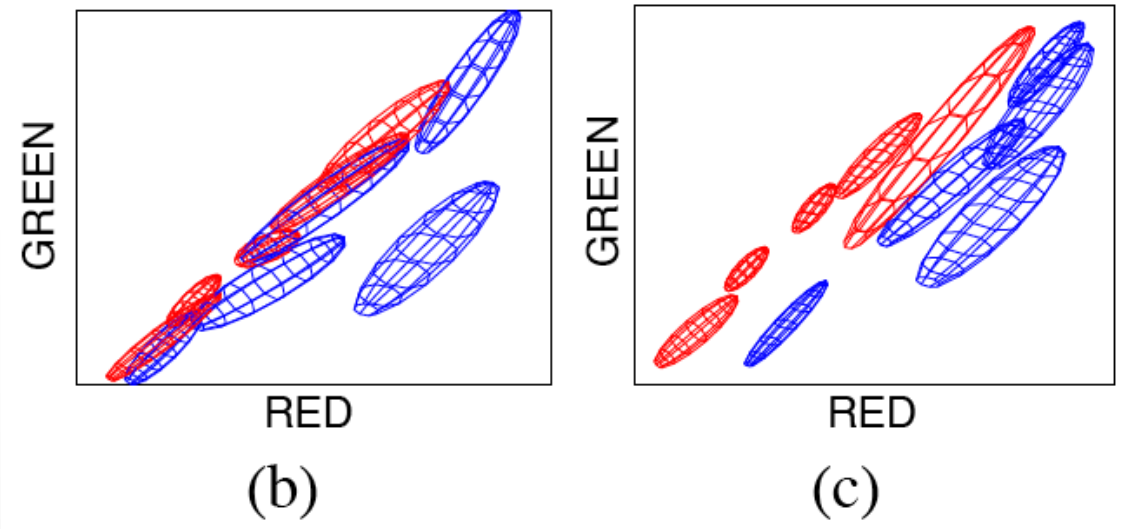
Data term = **whiteness**
(cost of assigning label)

Regularization = **color match**
(cost of separating neighbors)



Color Data Term

- Model 3D color histogram with Gaussians
  - Because brute force histogram would be sparse
  - Gaussian Mixture Model (GMM)
  - Just means histogram = sum of Gaussians
    - They advise 5 Gaussians



Iterated Graph Cuts

User Initialisation

Learn foreground color model

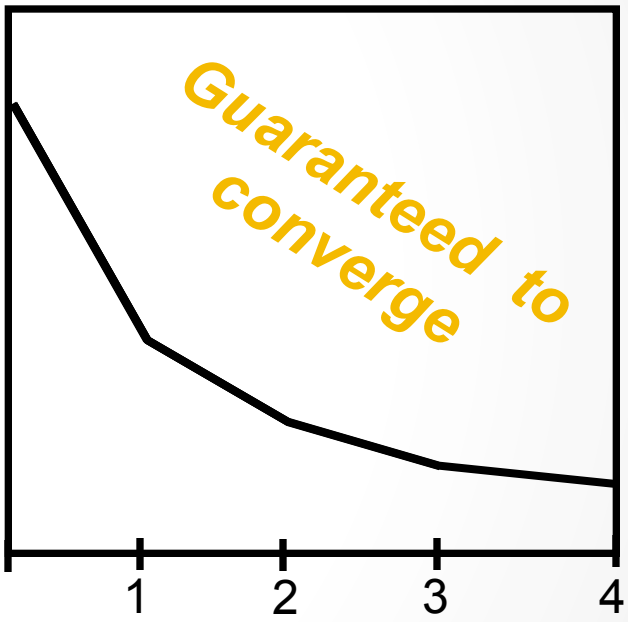
Graph cuts to infer the foreground

Grabcut: Iterative approach

- Initialize
  - Background with rectangle boundary pixels
  - Foreground with the interior of rectangle
- Iterate until convergence
  - Compute color probabilities (GMM) of each region
  - Perform graphcut segmentation
- Apply matting at boundary
- Potentially, user edits to correct mistakes

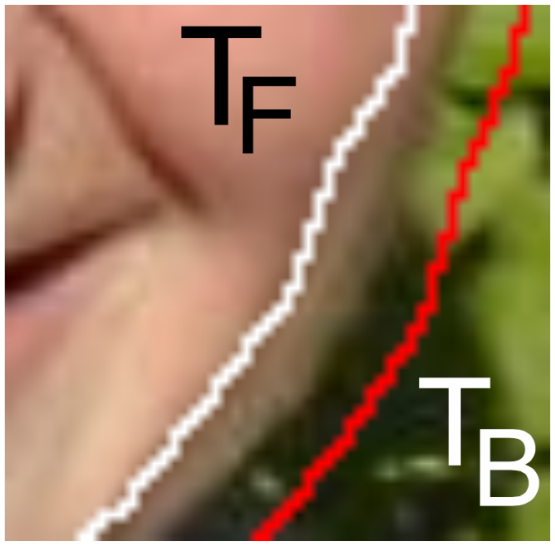
Iterated Graph Cuts

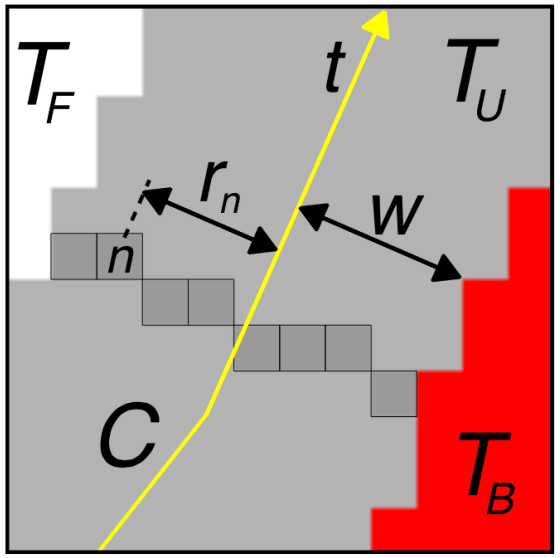
Result

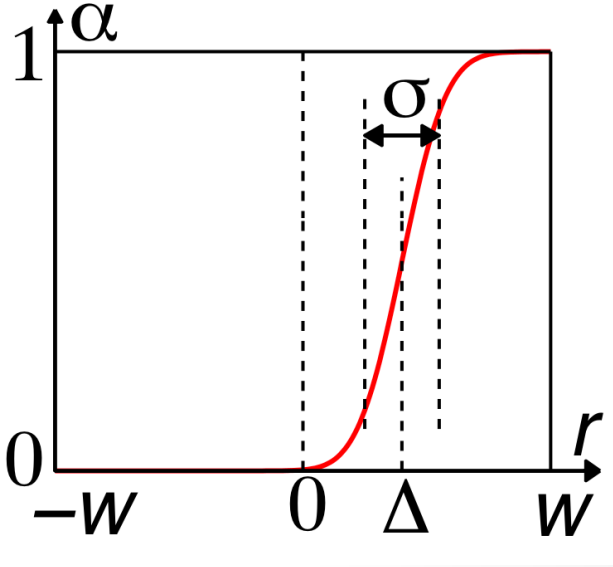


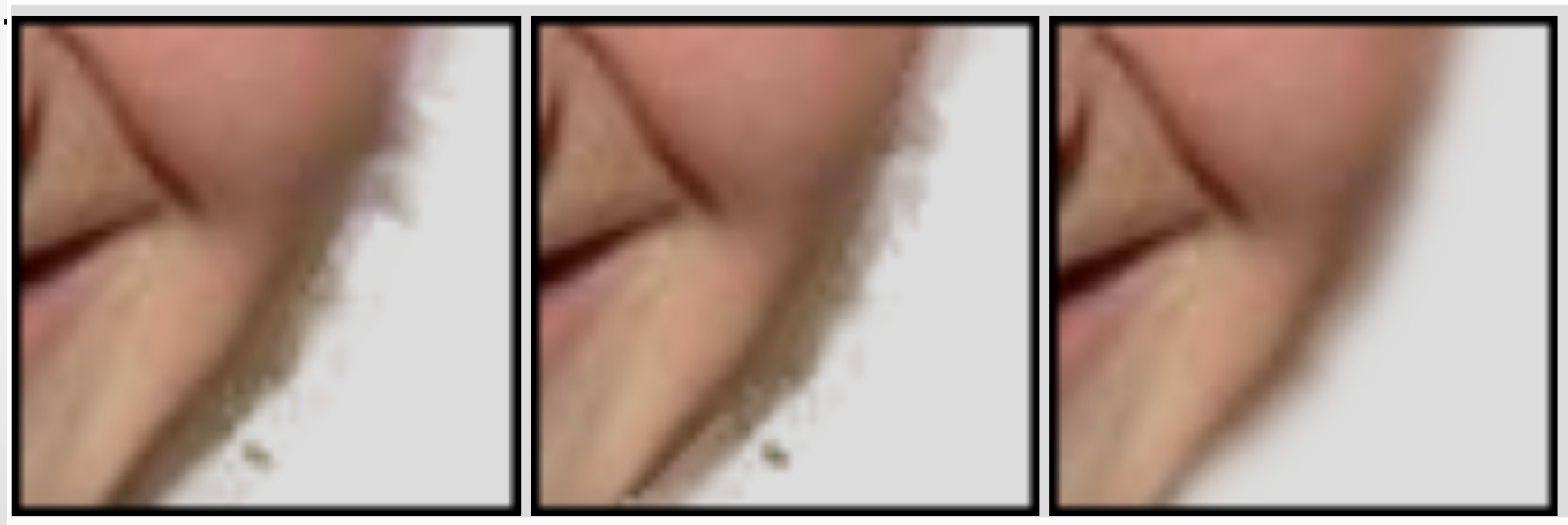
Energy after each Iteration

Border Matting





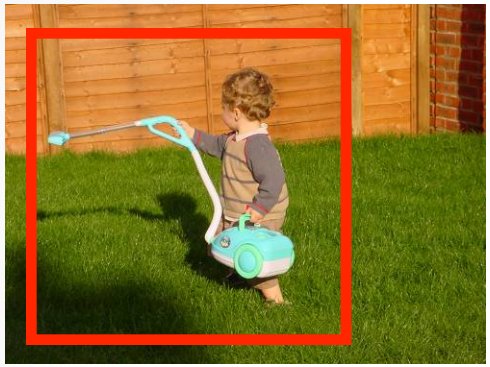




Knockout 2 Bayes Matte GrabCut

Border Matting Results

Examples: Moderate Complexity






… GrabCut completes automatically

User Interaction

Automatic

Segmentation

User

Interaction

Automatic

Segmentation

Examples: Difficult Ones

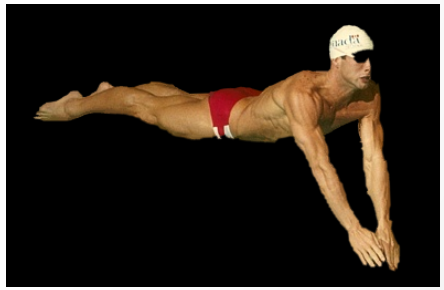
Camouflage & Low Contrast

Fine structure

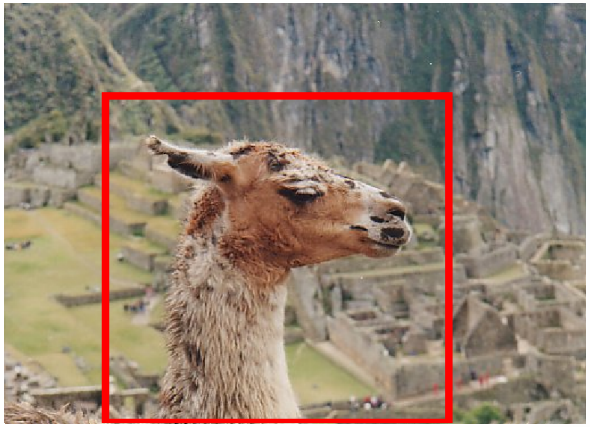
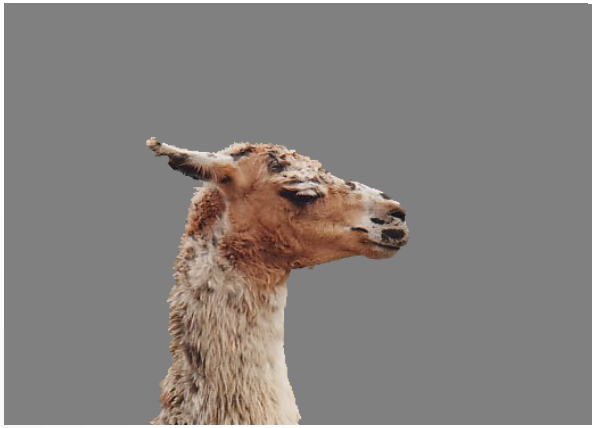
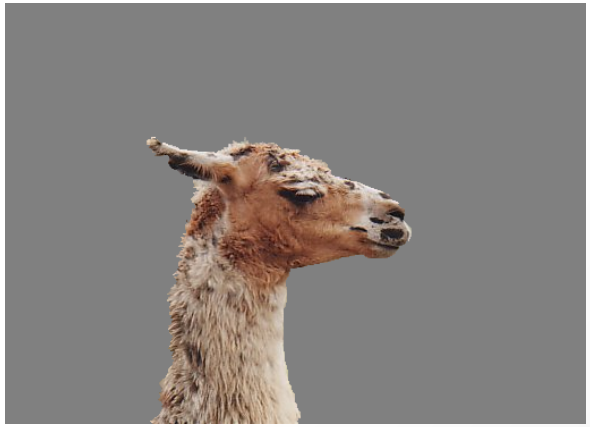
No telepathy

Initial Rectangle

Initial Result



GraphCut vs. GrabCut

	Boykov and Jolly (2001)	GrabCut
User Input		
Result		
	Error Rate: 0.72%	**Error Rate: 0.72%**

Reading Material

- Required
  - "Normalized Cuts and Image Segmentation", by Jianbo Shi and Jitendra Malik, TPAMI, Aug 2000
  - "Grabcut: Interactive Foreground Extraction using Iterated Graph Cuts", by Rother, Kolmogorov and Blake, Siggraph 2004.
- Additional
  - "OBJ CUT", by M. Pawan Kumar P.H.S. Torr and A. Zisserman, CVPR 2005.

Usage of MRF: An Example

Binarizing Natural Scene Text

Anand Mishra, Karteek Alahari and C. V. Jawahar

"An MRF Model for Binarization of Natural Scene Text",

Proc. ICDAR, Sept. 2011, Beijing, China

Natural Scene Text

- Challenging ICDAR 2003 and Street View Text (SVT) datasets

Natural Scene Text Recognition

Text Detection → Binarization → Character segmentation → Recognition

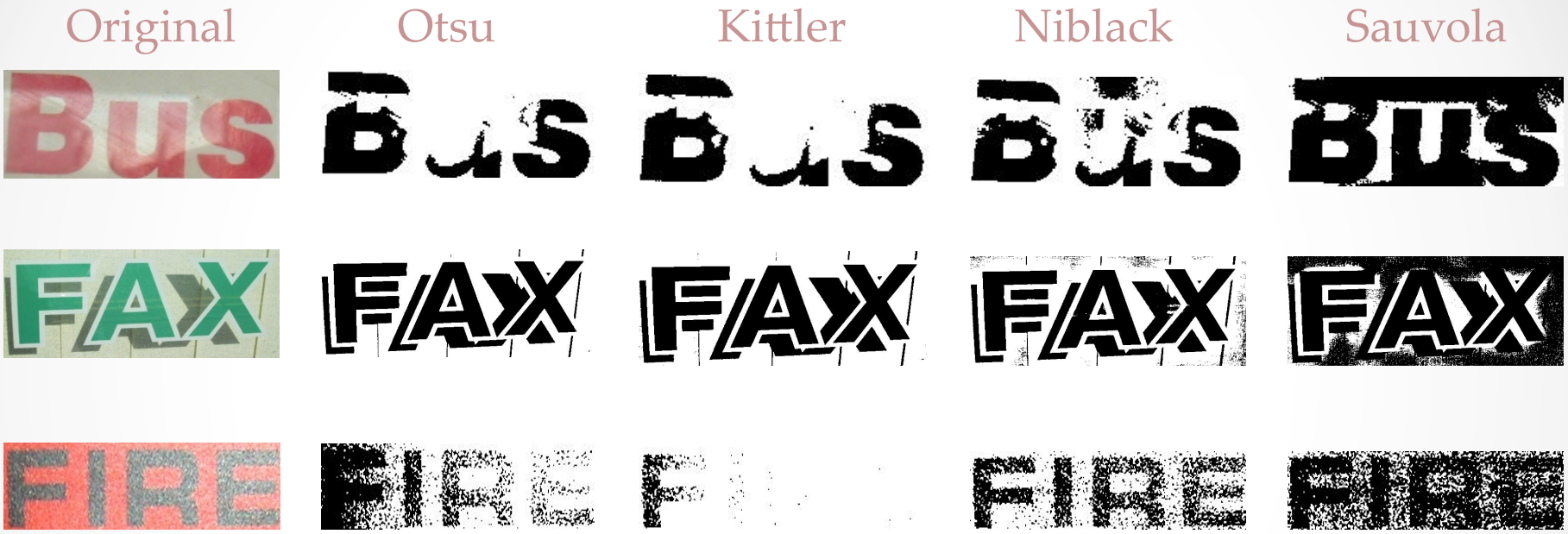
Scene Text Binarization: Challenges

- Reflections/ Transparency
- Illumination/Shadows

- Specularity
- Low Contrast
- Complex BG

- Noise
- Blur

Failure of Existing Methods



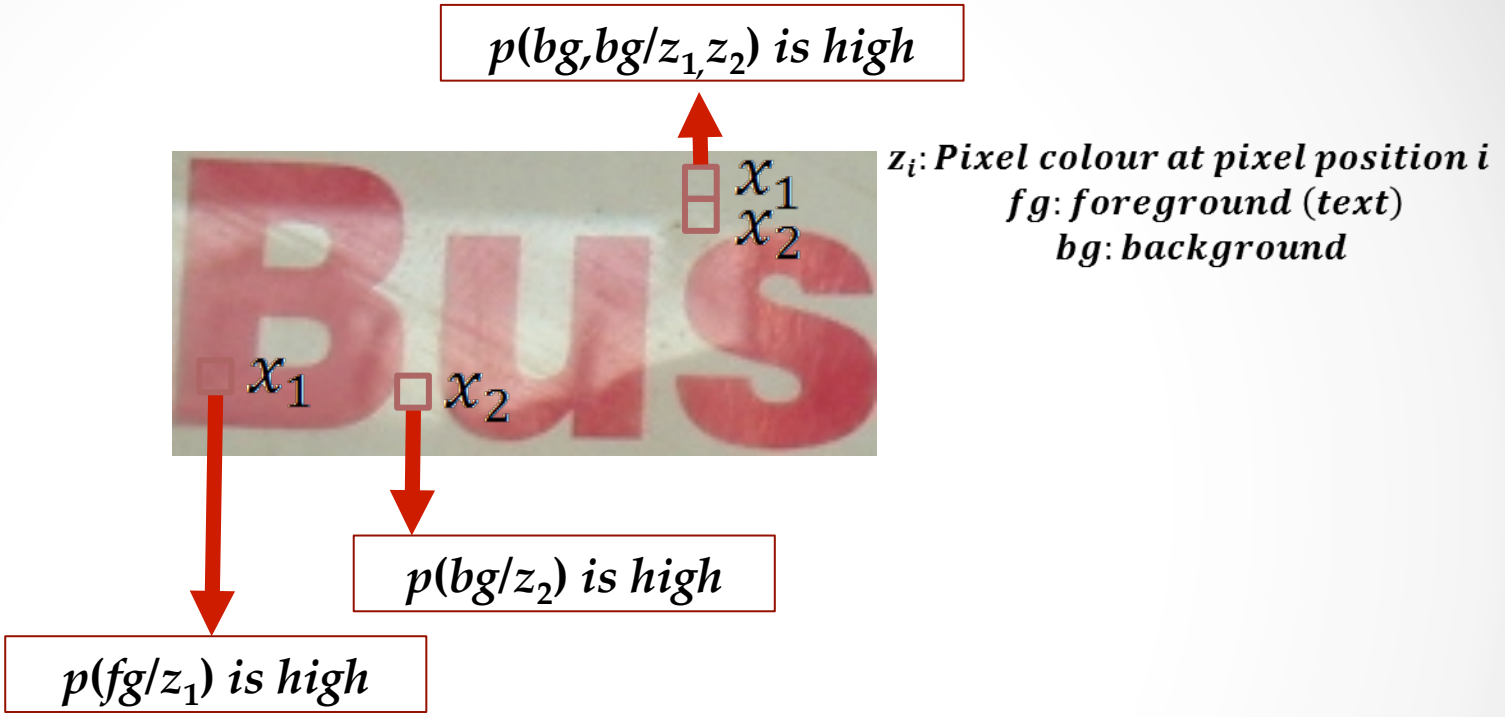
- ICDAR 2003 Robust Word Recognition Competition
  - A.Mishra et al. (ICDAR 2011): State-of-the-art
  - Y. Zhou et al. (ICDAR 2013): +2%, spotting using a small lexicon

An MRF based Binarization

Assign a label to each pixel from L = {Text (0), Background(1)}

Many labelings possible, we are interested in
“the optimal” one

MRF Formulation: Potentials



Minimize

$$E(x) = -\sum_i \log p(x_i|z_i) + \lambda_1 \sum_{i,j \in N} \exp\left(-\beta \|z_i - z_j\|^2\right)$$

Unary (data) Term ***Pair wise (smoothness) Term***

MRF Formulation: Gradient Potential

Gradient magnitude at pixel position i

Pair wise term = $$\lambda_1 \sum_{i,j \in N} \exp\left(-\beta \, ||z_i - z_j||^2\right) + \lambda_2 \sum_{i,j \in N} \exp\left(-\beta \, ||w_i - w_j||^2\right)$$

Edginess Term

An MRF based Binarization

The problem is to minimize following energy (MRF energy):

$$E(x) = Unary\ term + Pairwise\ term$$

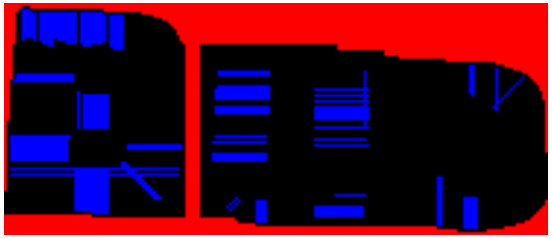
Two questions:

1)How to learn the probabilities $p(x_i|z_i)$ used to compute the unary term?
2)How to find the minima of above energy?

Learning Probabilities

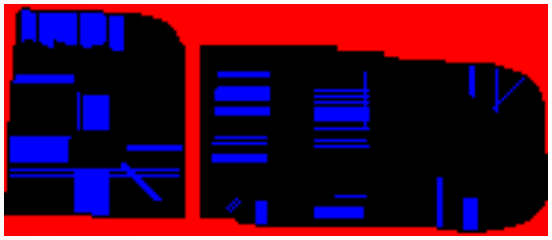
Canny Edge operator

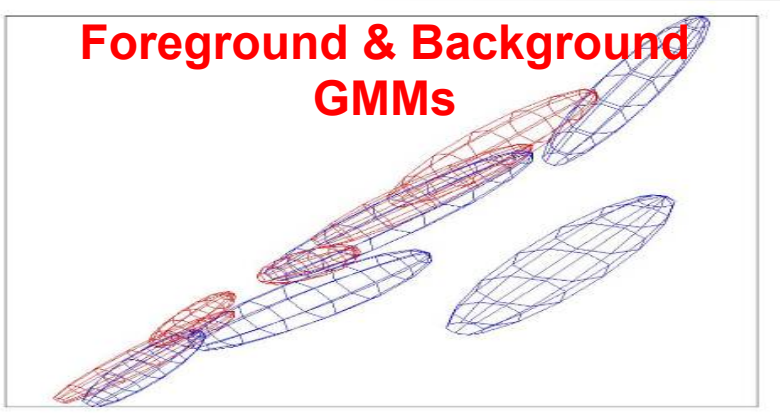
Find foreground - background seeds



Blue colour: Foreground
Red colour: Background

Color Modeling through GMMs





Unary term is calculated based on the probability of a pixel colour belonging to one of the GMM components

---

GrabCut (Carsten Rother et al., SIGGRAPH 2004) uses similar modelling for object segmentation problems

An MRF based Binarization

The problem is to minimize following energy (MRF energy):

$$E(x) = Unary\ term + Pair\ wise\ term$$

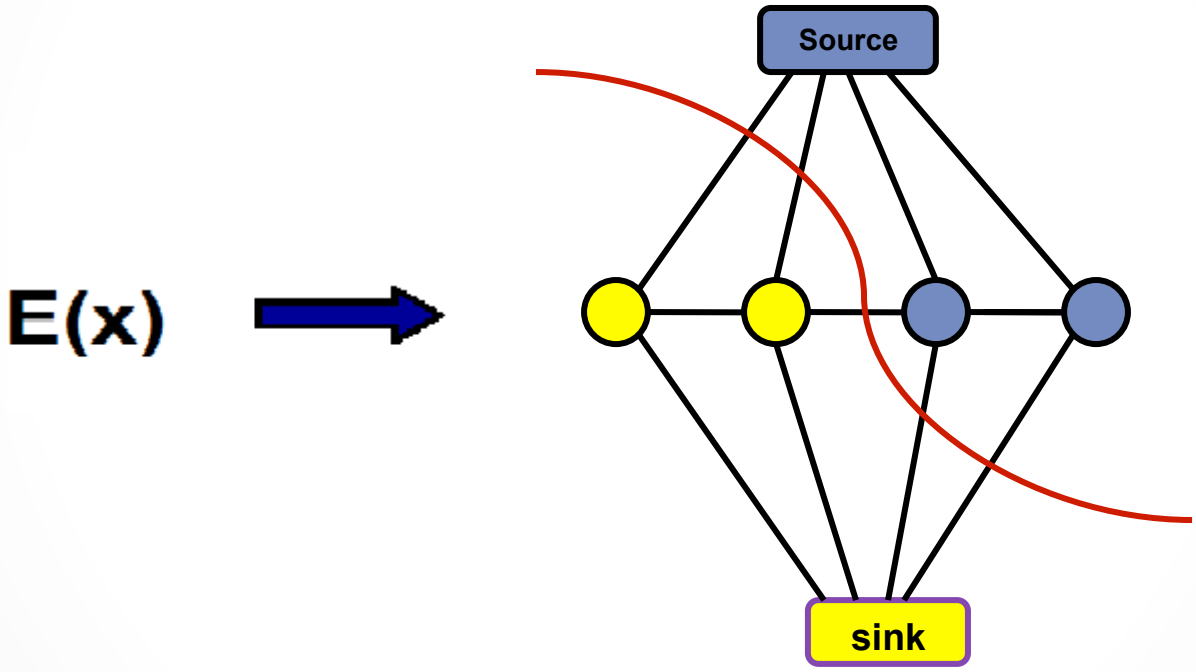
Two questions:

1) How to learn the probabilities $p(x_i|z_i)$ used to compute the unary term?

2) **How to find the minima of above energy?**

Graph Cut

Minimum of MRF energy = min cut of graph



Efficient codes available to compute min cut of such graph

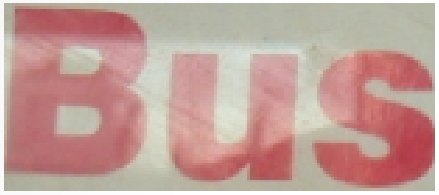
Vladimir Kolmogorov and Ramin Zabih, "What Energy Functions can be minimized via Graph Cut", PAMI 2004

An Iterative Graph Cut based Approach

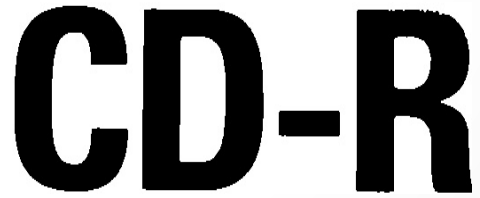
Learn GMMs to model foreground and background colours

Graph cuts to refine binarization

Qualitative Results

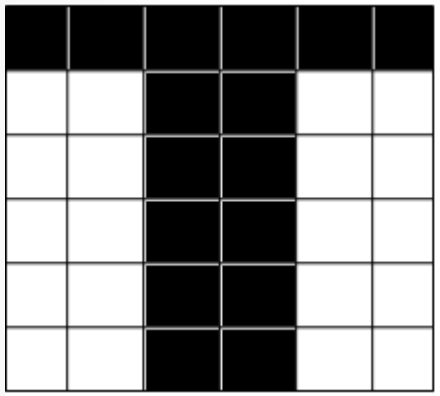


Qualitative Results

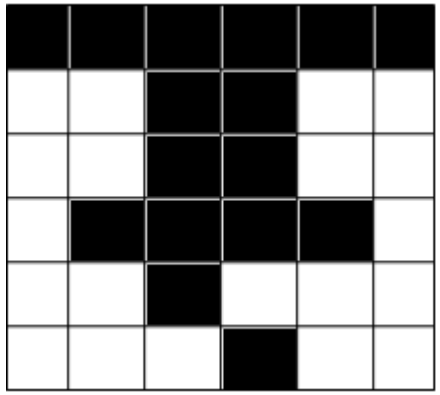


Quantitative Results

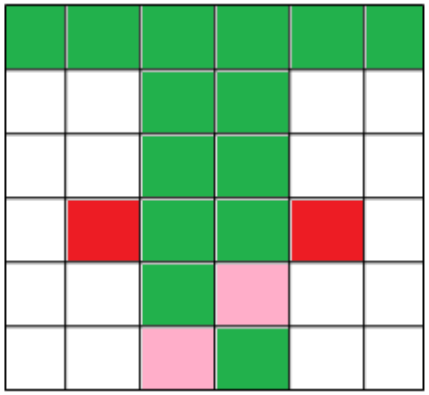
- OCR accuracy
- Pixel level accuracy



Ground truth



Binarization result



Evaluation result

True positive

False positive

False Negative

$$precision = \frac{Total\ number\ of\ green\ boxes}{Total\ number\ of\ green\ boxes + Total\ number\ of\ red\ boxes}$$

$$recall = \frac{Total\ number\ of\ green\ boxes}{Total\ number\ of\ green\ boxes + Total\ number\ of\ pink\ boxes}$$

$$f - score = \frac{2 \times precison \times recall}{precision + recall} \times 100$$

Results (ABBYY OCR Accuracy)

Method	Word Accuracy (%)	Character Accuracy (%)
Otsu	41.52	51.74
Sauvola	39.77	51.63
Niblack	39.18	42.31
Kittler	41.12	49.88
Otsu + CT	45.03	51.98
MRF (without edginess diff.)	49.12	55.94
MRF (with edginess diff.)	**52.04**	**60.14**

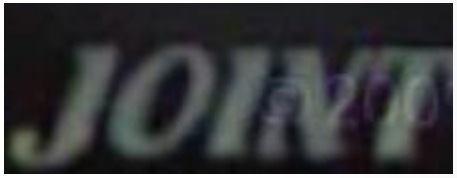
Results (Pixel Level Accuracy)

Method	f-score (%)
Otsu	79.32
Sauvola	73.87
Niblack	76.86
Kittler	72.89
Otsu + CT	78.12
MRF (without edginess diff.)	87.84
MRF (with edginess diff.)	**88.64**

More Results

Results based on Street View Text Dataset

Method	Word Recognition accuracy (%)
ABBYY	32.61%
MRF Binarization + ABBYY	**42.81%**



Kai Wang and Serge Belongie (ECCV 2010) have introduced a challenging Street View Text (SVT) dataset

When can it Fail?

- Colour may not characterize the chatacter.

- Failing to learn text-BG probabilities

Reading Material

- Required
  - "Normalized Cuts and Image Segmentation", by Jianbo Shi and Jitendra Malik, TPAMI, Aug 2000
  - "Grabcut: Interactive Foreground Extraction using Iterated Graph Cuts", by Rother, Kolmogorov and Blake, Siggraph 2004.
  - "An MRF Model for Binarization of Natural Scene Text", by Anand Mishra, Karteek Alahari and C.V. Jawahar, ICDAR 2011.
- Additional
  - "OBJ CUT", by M. Pawan Kumar P.H.S. Torr and A. Zisserman, CVPR 2005.