CMP717
Image Processing

Graphical Models

Erkut Erdem
Hacettepe University
Computer Vision Lab (HUCVL)

Energy Minimization

- Many vision tasks are naturally posed as energy minimization problems on a rectangular grid of pixels:

$$E(u) \ = \ E_{data}(u) \ + \ E_{smoothness}(u)$$

- The data term $E_{data}(u)$ expresses our goal that the optimal model u be consistent with the measurements.
- The smoothness energy $E_{smoothness}(u)$ is derived from our prior knowledge about plausible solutions.

- Recall Mumford-Shah functional

Sample Vision Tasks

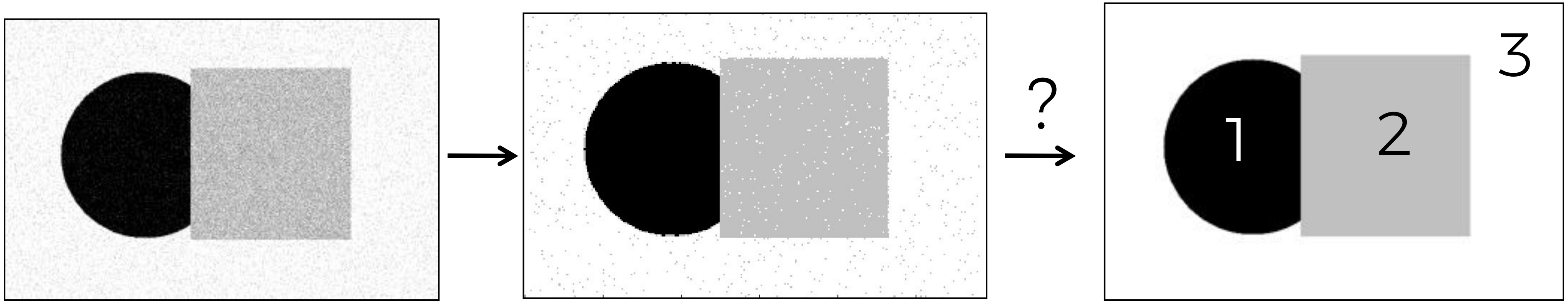
- <u>Image Denoising:</u> Given a noisy image $\hat{I}(x,y)$, where some measurements may be missing, recover the original image $I(x, y)$, which is typically assumed to be smooth.

- <u>Image Segmentation:</u> Assign labels to pixels in an image, e.g., to segment foreground from background.

- Stereo matching
- Surface Reconstruction
- …

Smoothing out cluster assignments

- Assigning a cluster label per pixel may yield outliers:



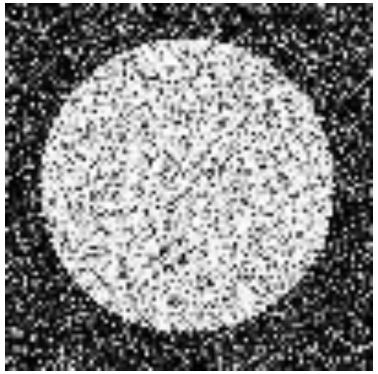
original

labeled by cluster center's intensity

- How to ensure they are spatially smooth?

K. Grauman

Solution



P(foreground | image)

Normalizing constant

$$P(\mathbf{y};\theta,data)=\frac{1}{Z}\prod_{i=1..N}f_1(y_i;\theta,data)\prod_{i,j\in edges}f_2(y_i,y_j;\theta,data)$$

Labels to be predicted

Individual predictions

Pairwise predictions

Encode dependencies between pixels

Writing Likelihood as an "Energy"

$$P(\mathbf{y};\theta,data)=\frac{1}{Z}\prod_{i=1..N}p_1(y_i;\theta,data)\prod_{i,j\in edges}p_2(y_i,y_j;\theta,data)$$

$$Energy(\mathbf{y};\theta,data)=\sum_i\psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

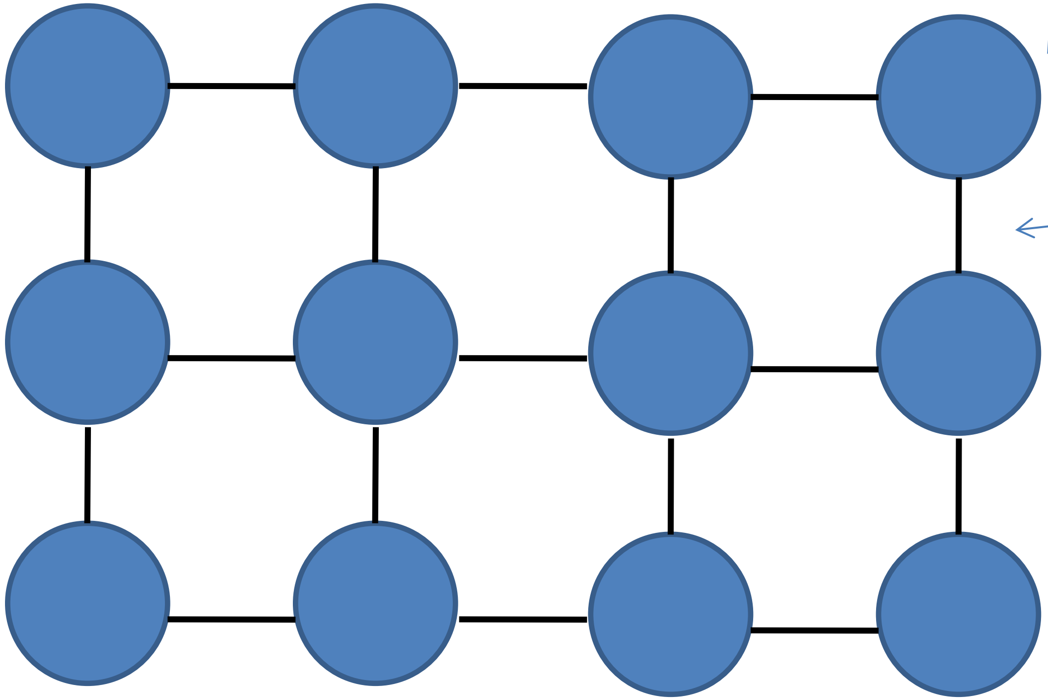
"Cost" of assignment y_i

"Cost" of pairwise assignment y_i, y_j

D. Hoiem

Markov Random Fields

Node y_i: pixel label



Edge:
constrained pairs

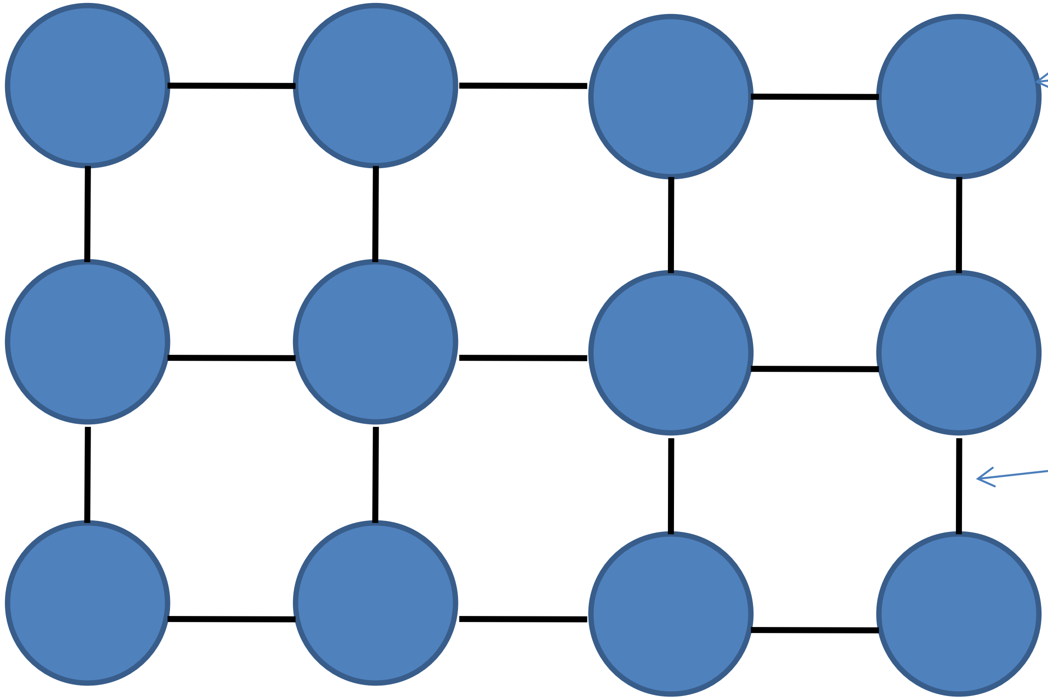
Cost to assign a label to each pixel

Cost to assign a pair of labels to connected pixels

$$Energy(\mathbf{y};\theta,data)=\sum_i \psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

D. Hoiem

Markov Random Fields



Unary potential

0: $-\log P(y_i = 0 \,;\, data)$

1: $-\log P(y_i = 1 \,;\, data)$

Pairwise Potential

	0	1
0	0	K
1	K	0

- Example: “label smoothing” grid

$$Energy(\mathbf{y};\theta,data)=\sum_{i}\psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

D. Hoiem

Binary MRF Example

- Consider the following energy function for two binary random variables, y_1 & y_2.

$$\begin{array}{c|c|} 0 & 5 \\ \hline 1 & 2 \end{array} \qquad \begin{array}{c|c|} 0 & 1 \\ \hline 1 & 3 \end{array} \qquad \begin{array}{c|c|c|} & 0 & 1 \\ 0 & 0 & 3 \\ \hline 1 & 4 & 0 \end{array}$$

$$E\left(y_1, y_2\right) = \psi_1(y_1) + \psi_2(y_2) + \psi_{12}(y_1, y_2)$$

Binary MRF Example

- Consider the following energy function for two binary random variables, y_1 & y_2.

0	5
1	2

0	1
1	3

	0	1
0	0	3
1	4	0

$$
\begin{aligned}
E(y_1, y_2) &= \psi_1(y_1) + \psi_2(y_2) + \psi_{12}(y_1, y_2) \\
&= \underbrace{5\bar{y}_1 + 2y_1}_{\psi_1} \\
&\qquad + \underbrace{\bar{y}_2 + 3y_2}_{\psi_2} \\
&\qquad\quad + \underbrace{3\bar{y}_1 y_2 + 4y_1\bar{y}_2}_{\psi_{12}}
\end{aligned}
$$

where $\bar{y}_1 = 1 - y_1$ and $\bar{y}_2 = 1 - y_2$.

Binary MRF Example

- Consider the following energy function for two binary random variables, y_1 & y_2.

$$
\begin{array}{c|c|}
0 & 5 \\
1 & 2
\end{array}
\qquad
\begin{array}{c|c|}
0 & 1 \\
1 & 3
\end{array}
\qquad
\begin{array}{c|c|c|}
 & 0 & 1 \\
0 & 0 & 3 \\
1 & 4 & 0
\end{array}
$$

$$
\begin{aligned}
E(y_1, y_2) &= \psi_1(y_1) + \psi_2(y_2) + \psi_{12}(y_1, y_2) \\
&= \underbrace{5\bar{y}_1 + 2y_1}_{\psi_1} \\
&\qquad + \underbrace{\bar{y}_2 + 3y_2}_{\psi_2} \\
&\qquad\quad + \underbrace{3\bar{y}_1 y_2 + 4y_1\bar{y}_2}_{\psi_{12}}
\end{aligned}
$$

where $\bar{y}_1 = 1 - y_1$ and $\bar{y}_2 = 1 - y_2$.

Graphical Model

Probability Table

y_1	y_2	E	P
0	0	6	0.244
0	1	11	0.002
1	0	7	0.090
1	1	5	0.664

Image Denoising

- Given a noisy image v, perhaps with missing pixels, recover an image u that is both smooth and close to v.
- Classical techniques:
  - Linear filtering (e.g. Gaussian filtering)
  - Median filtering
  - Wiener filtering
- Modern techniques
  - PDE-based techniques
  - Non-local methods
  - Wavelet techniques
  - MRF-based techniques

Denoising/smoothing techniques that preserve edges in images

Denoising as a Probabilistic Inference

- Perform maximum a posteriori (MAP) estimation by maximizing the a posteriori distribution:

$$p(\text{true image} \mid \text{noisy image}) = p(u \mid v)$$

likelihood of noisy image given true image

image prior

- By Bayes theorem:

$$p(u \mid v) = \frac{p(v \mid u)\, p(u)}{p(v)}$$

normalization term

- If we take logarithm:

$$\log p(u \mid v) = \log p(v \mid u) + \log p(u) - \cancel{\log p(v)}$$

- MAP estimation corresponds to minimizing the encoding cost

$$E(u) = -\log p(v \mid u) - \log p(u)$$

Modeling the Likelihood

- We assume that the noise at one pixel is independent of the others.

$$p(v \mid u) = \prod_{i,j} p(v_{ij} \mid u_{ij})$$

- We assume that the noise at each pixel is additive and Gaussian distributed:

$$p(v_{ij} \mid u_{ij}) = G_\sigma(v_{ij} - u_{ij})$$

- Thus, we can write the likelihood:

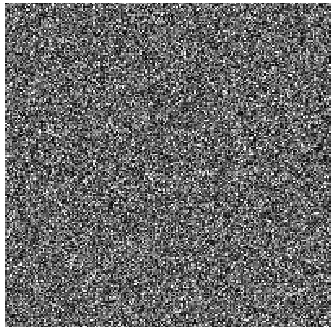
$$p(v \mid u) = \prod_{i,j} G_\sigma(v_{ij} - u_{ij})$$

Modeling the Prior

- How do we model the prior distribution of true images?
- What does that even mean?
  - We want the prior to describe how probable it is (a-priori) to have a particular true image among the set of all possible images.

probable

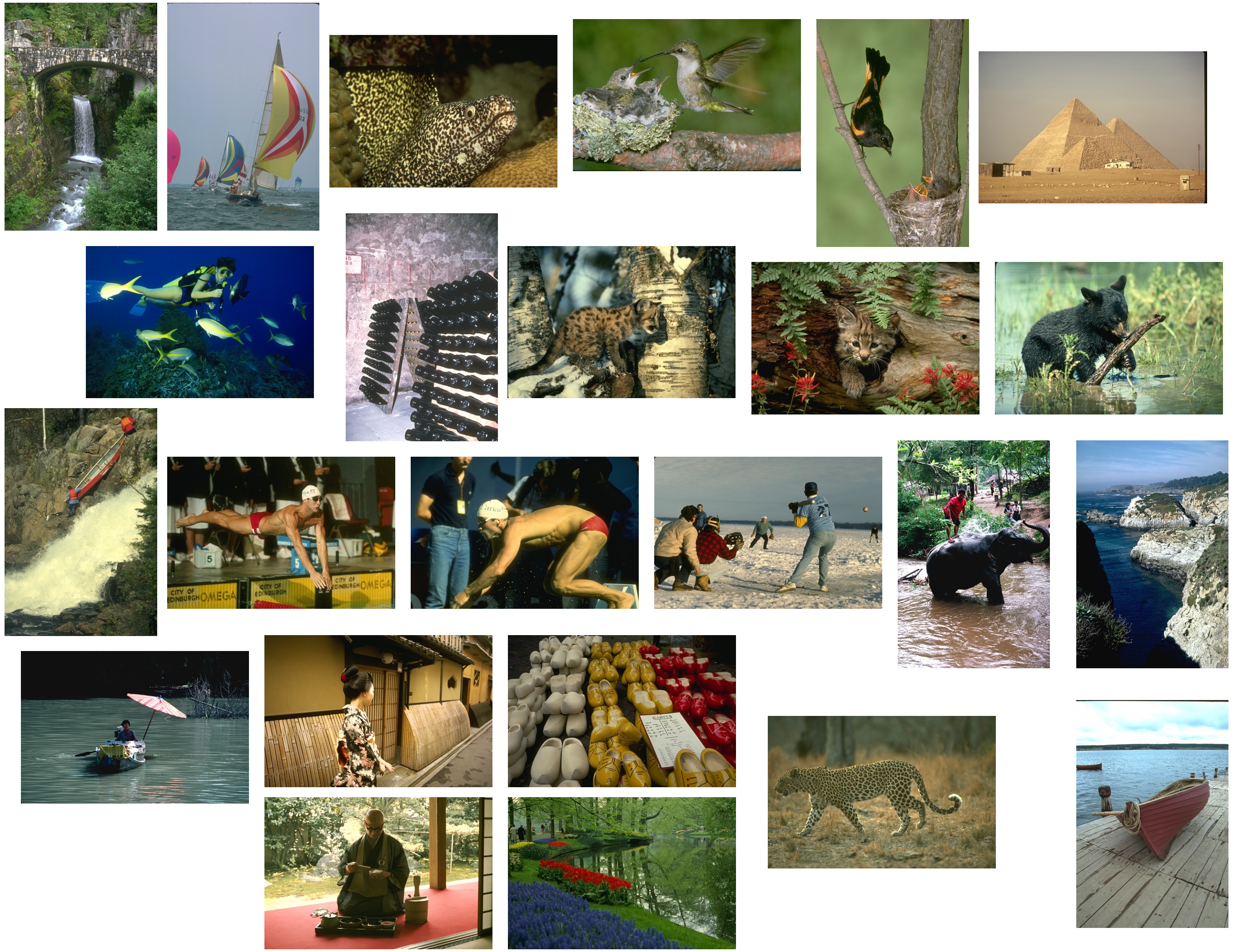
improbable



S. Roth

Natural Images

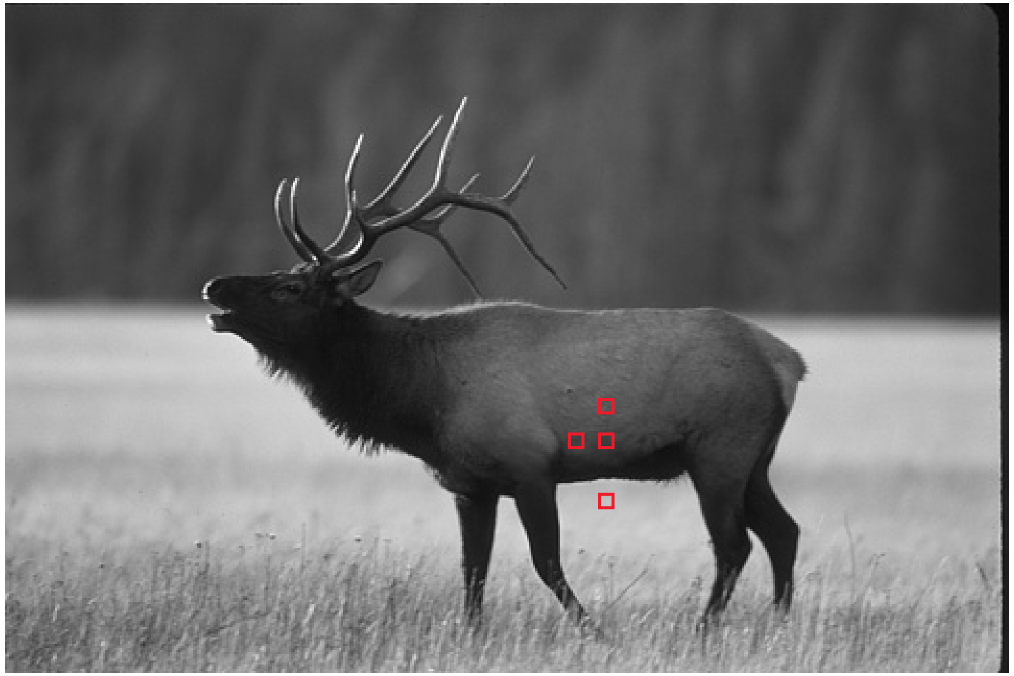
- What distinguishes “natural” images from “fake” ones?



S. Roth

Simple Observation

- Nearby pixels often have a similar intensity/color:



- But sometimes there are large intensity/color changes.

MRF-based Image Denoising

- Let each pixel be a node in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with 4-connected neighborhoods.

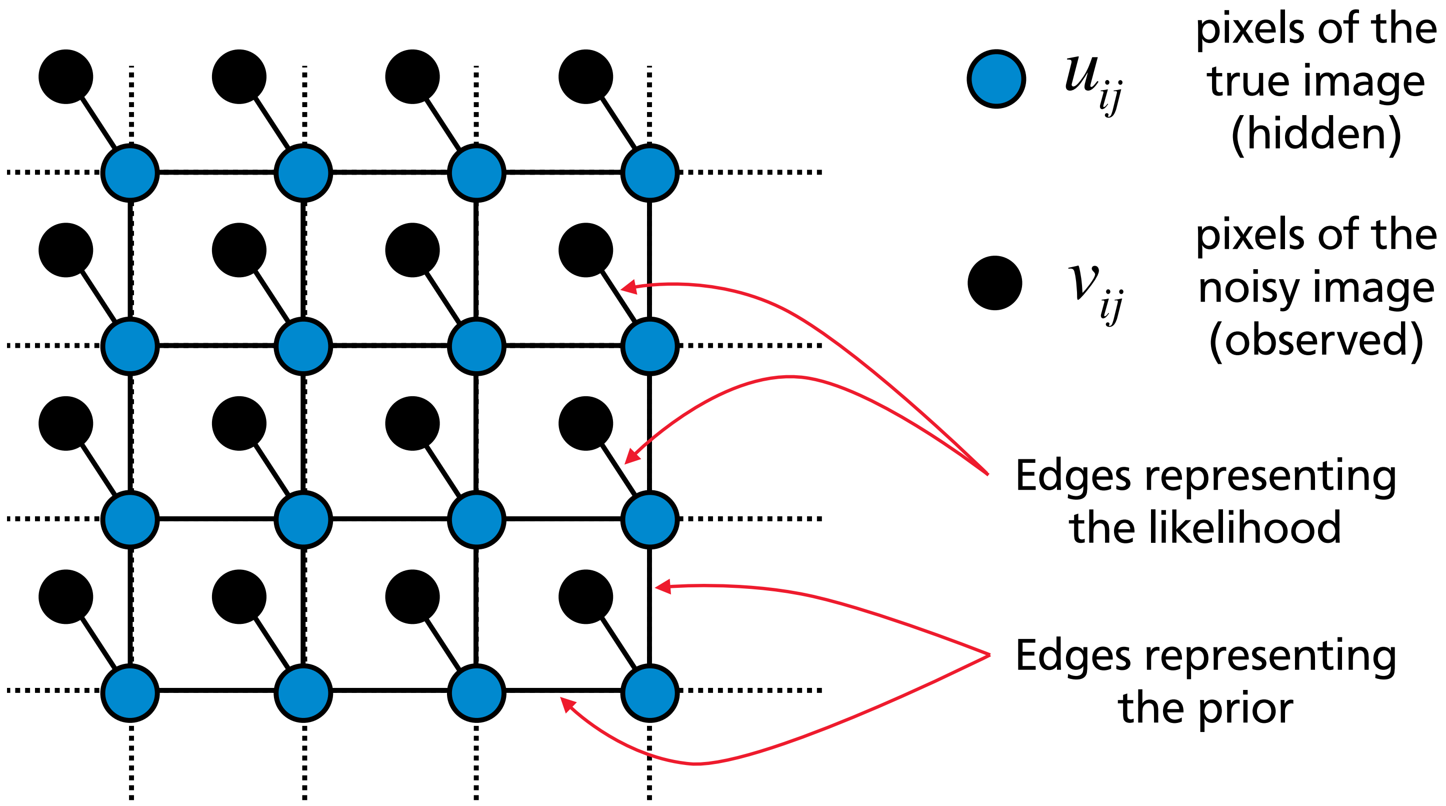


Image Denoising

- The energy function is given by

$$E(u) \;=\; \sum_{i \in \mathcal{V}} D(u_i) \;+\; \sum_{(i,j) \in \mathcal{E}} V(u_i, u_j)$$

- Unary (clique) potentials D stem from the measurement model, penalizing the discrepancy between the data v and the solution u .

- Interaction (clique) potentials V provide a definition of smoothness, penalizing changes in u, between pixels and their neighbors.

Denoising as Inference

- <u>Goal:</u> Find the image u that minimizes $E(u)$
- Several options for MAP estimation process:
  - Gradient techniques
  - Gibbs sampling
  - Simulated annealing
  - Belief propagation
  - Graph cut
  - …

Quadratic Potentials in 1D

- Let v be the sum of a smooth 1D signal u and IID Gaussian noise e : where $u = (u_1, ..., u_N)$, $v = (v_1, ..., v_N)$, and $e = (e_1, ..., e_N)$.

- With Gaussian IID noise, the negative log likelihood provides a quadratic *data term.* If we let the *smoothness term* be quadratic as well, then up to a constant, the log posterior is

$$E(u) \;=\; \sum_{n=1}^{N} (u_n - v_n)^2 + \lambda \sum_{n=1}^{N-1} (u_{n+1} - u_n)^2$$

Quadratic Potentials in 1D

- To find the optimal u^*, we take derivatives of $E(u)$ with respect to u_n :

$$\frac{\partial E(u)}{\partial u_n} = 2(u_n - v_n) + 2\lambda(-u_{n-1} + 2u_n - u_{n+1})$$

and therefore the necessary condition for the critical point is

$$u_n + \lambda(-u_{n-1} + 2u_n - u_{n+1}) = v_n$$

- For endpoints we obtain different equations:

$$u_1 + \lambda(u_1 - u_2) = v_1$$

$$u_N + \lambda(u_N - u_{N-1}) = v_N$$

N linear equations in the N unknowns

D. J. Fleet

Missing Measurements

- Suppose our measurements exist at a subset of positions, denoted P. Then we can write the energy function as

$$E(u) \; = \; \sum_{n \in P} (u_n - v_n)^2 + \lambda \sum_{\text{all}\, n} (u_{n+1} - u_n)^2$$

- At locations n where no measurement exists, we have:

$$-u_{n-1} + 2\, u_n - u_{n+1} \; = \; 0$$

- The Jacobi update equation in this case becomes:

$$u_n^{(t+1)} = \begin{cases} \frac{1}{1+2\lambda} \left(v_n + \lambda u_{n-1}^{(t)} + \lambda u_{n+1}^{(t)} \right) & \text{for}\, n \in P \,, \\ \frac{1}{2} \left(u_{n-1}^{(t)} + u_{n+1}^{(t)} \right) & \text{otherwise} \end{cases}$$

2D Image Smoothing

- For 2D images, the analogous energy we want to minimize becomes:

$$
\begin{aligned}
E(u) \;=\; & \sum_{n,m \in P} (u[n,m] - v[n,m])^2 \\
& + \lambda \sum_{\text{all}\, n,m} (u[n{+}1,m] - u[n,m])^2 + (u[n,m{+}1] - u[n,m])^2
\end{aligned}
$$

where P is a subset of pixels where the measurements v are available.

Looks familiar??

Robust Potentials

- Quadratic potentials are not robust to *outliers* and hence they over-smooth edges. These effects will propagate throughout the graph.
- Instead of quadratic potentials, we could use a robust error function ρ:

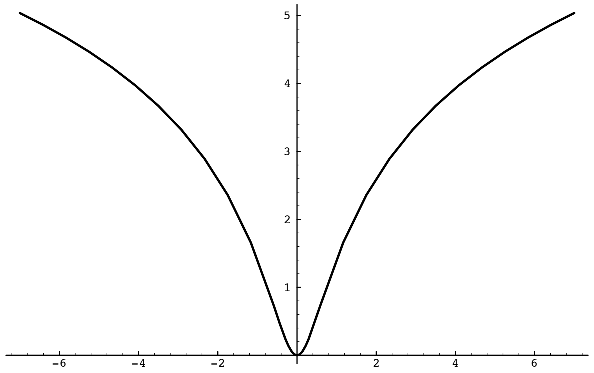
$$E(u) \;=\; \sum_{n=1}^{N} \rho(u_n - v_n,\, \sigma_d) + \lambda \sum_{n=1}^{N-1} \rho(u_{n+1} - u_n,\, \sigma_s)\,,$$

where σ_d and σ_s are scale parameters.

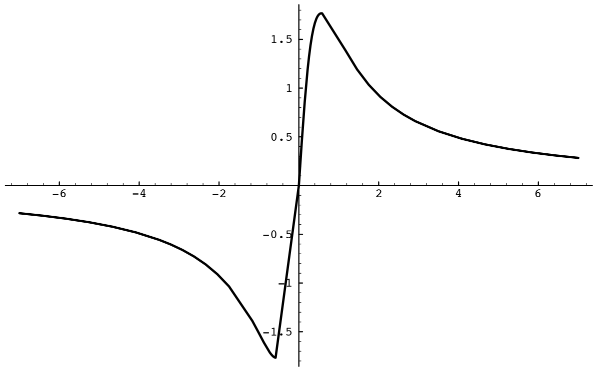
Robust Potentials

- Example: the *Lorentzian* error function

$$\rho(z,\sigma) = \log\left(1 + \frac{1}{2}\left(\frac{z}{\sigma}\right)^2\right), \quad \rho'(z,\sigma) = \frac{2z}{2\sigma^2 + z^2}.$$



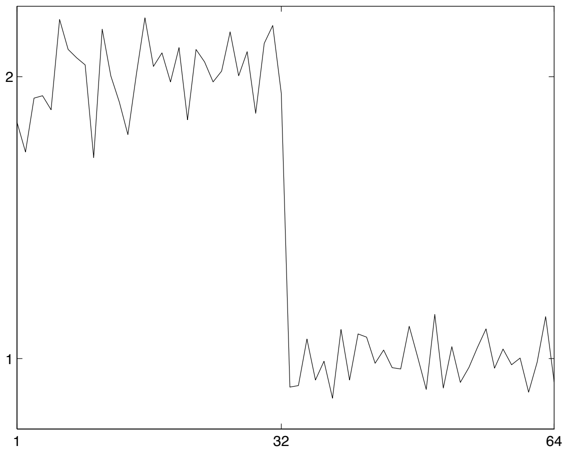
Error function



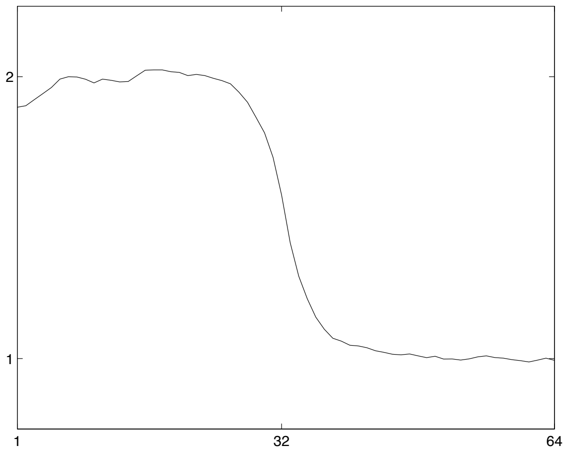
Influence function

Robust Potentials

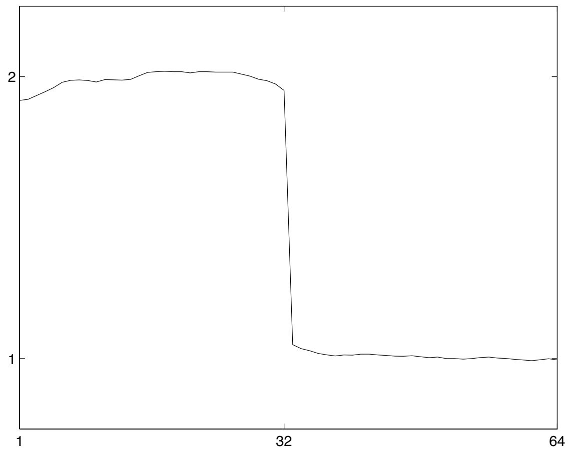
- <u>Example:</u> the *Lorentzian* error function
- Smoothing a noisy step edge



Noisy step



LS smoother



Lorentzian smoother

Robust Image Smoothing

- A Lorentzian smoothness potential encourages an approximately piecewise constant result:

Original image

Output of robust smoothing

Edges

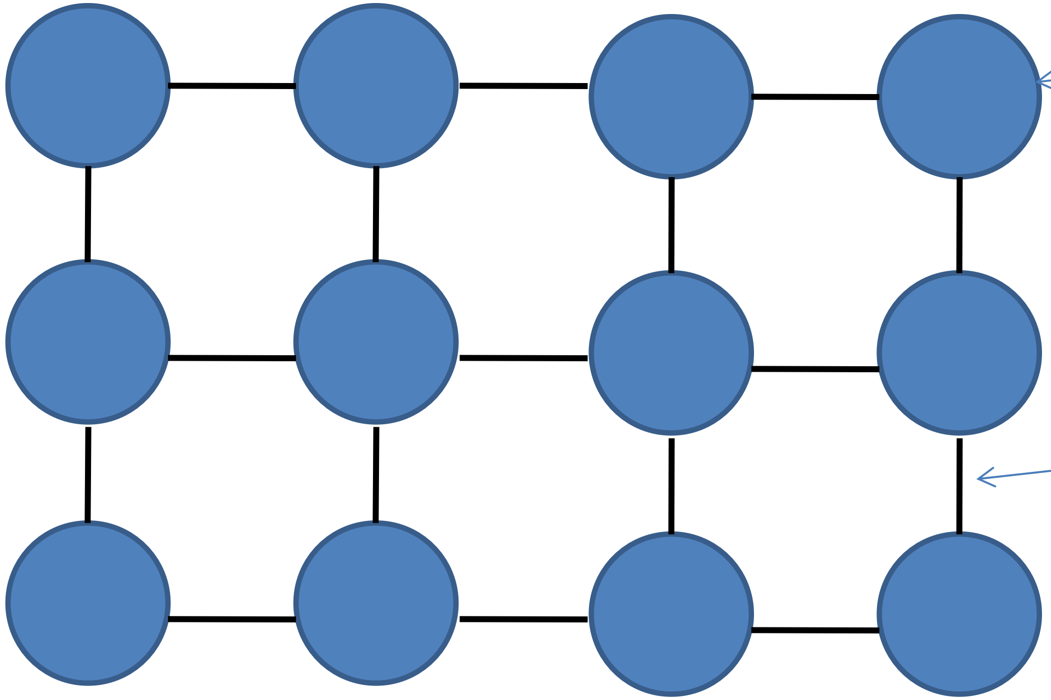
D. J. Fleet

Image Segmentation

- Given an image, partition it into meaningful regions or segments.
- Approaches
  - Variational segmentation models
  - Clustering-based approaches (K-means, Mean Shift)
  - Graph-theoretic formulations
- MRF-based techniques

MRFs and Graph-cut

Markov Random Fields



Unary potential

0: -logP(y_i = 0 ; data)
1: -logP(y_i = 1 ; data)

Pairwise Potential

	0	1
0	0	K
1	K	0

- Example: “label smoothing” grid

$$Energy(\mathbf{y};\theta,data)=\sum_i \psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

D. Hoiem

Solving MRFs with graph cuts

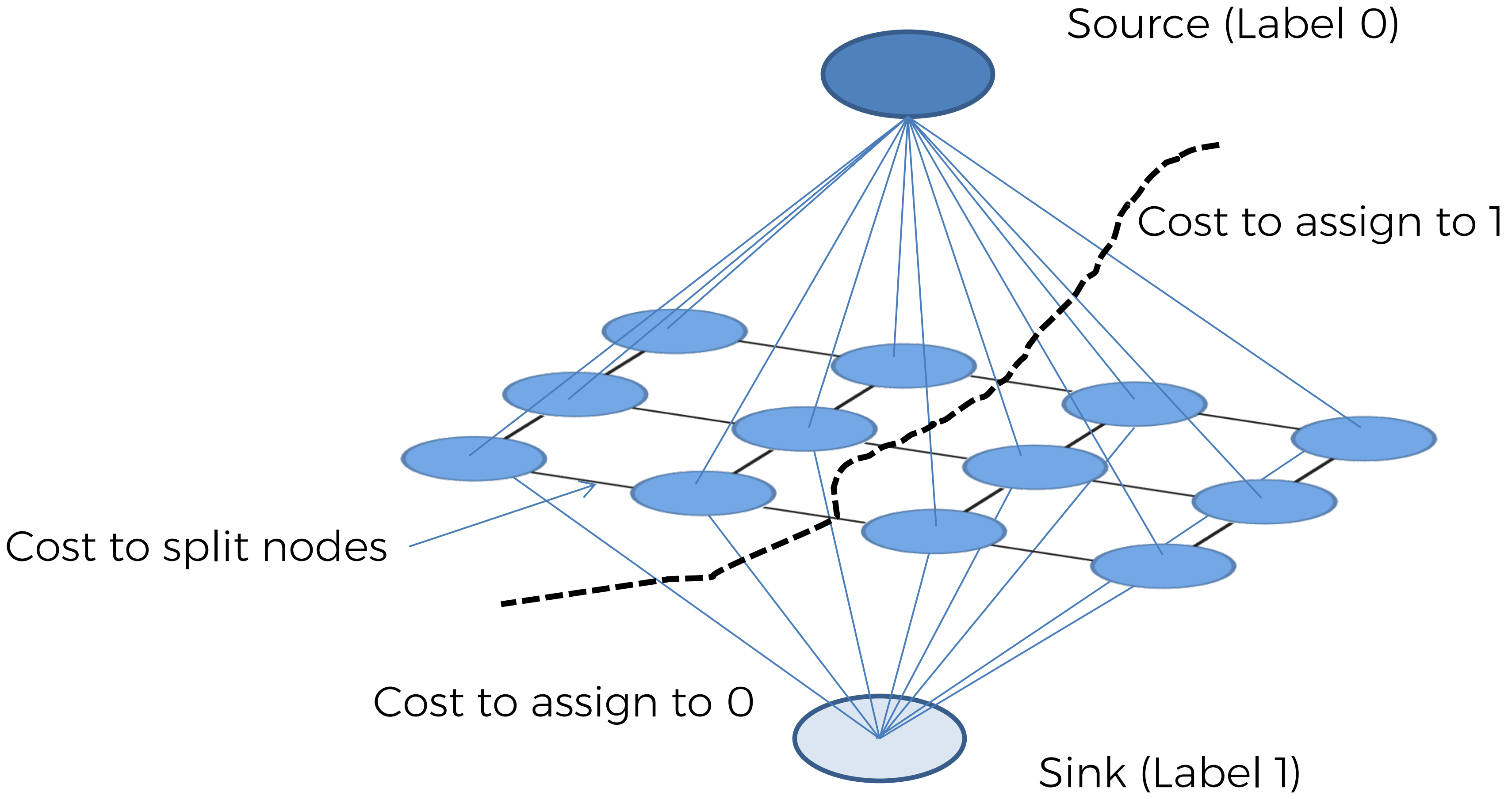
Main idea:

- Construct a graph such that every *st*-cut corresponds to a joint assignment to the variables $\mathbf{y}$
- The cost of the cut should be equal to the energy of the assignment, $E(\mathbf{y}; \text{data})^*$.
- The minimum-cut then corresponds to the minimum energy assignment, $\mathbf{y}^\star = \text{argmin}_\mathbf{y} E(\mathbf{y}; \text{data})$.

* Requires non-negative energies

S. Gould

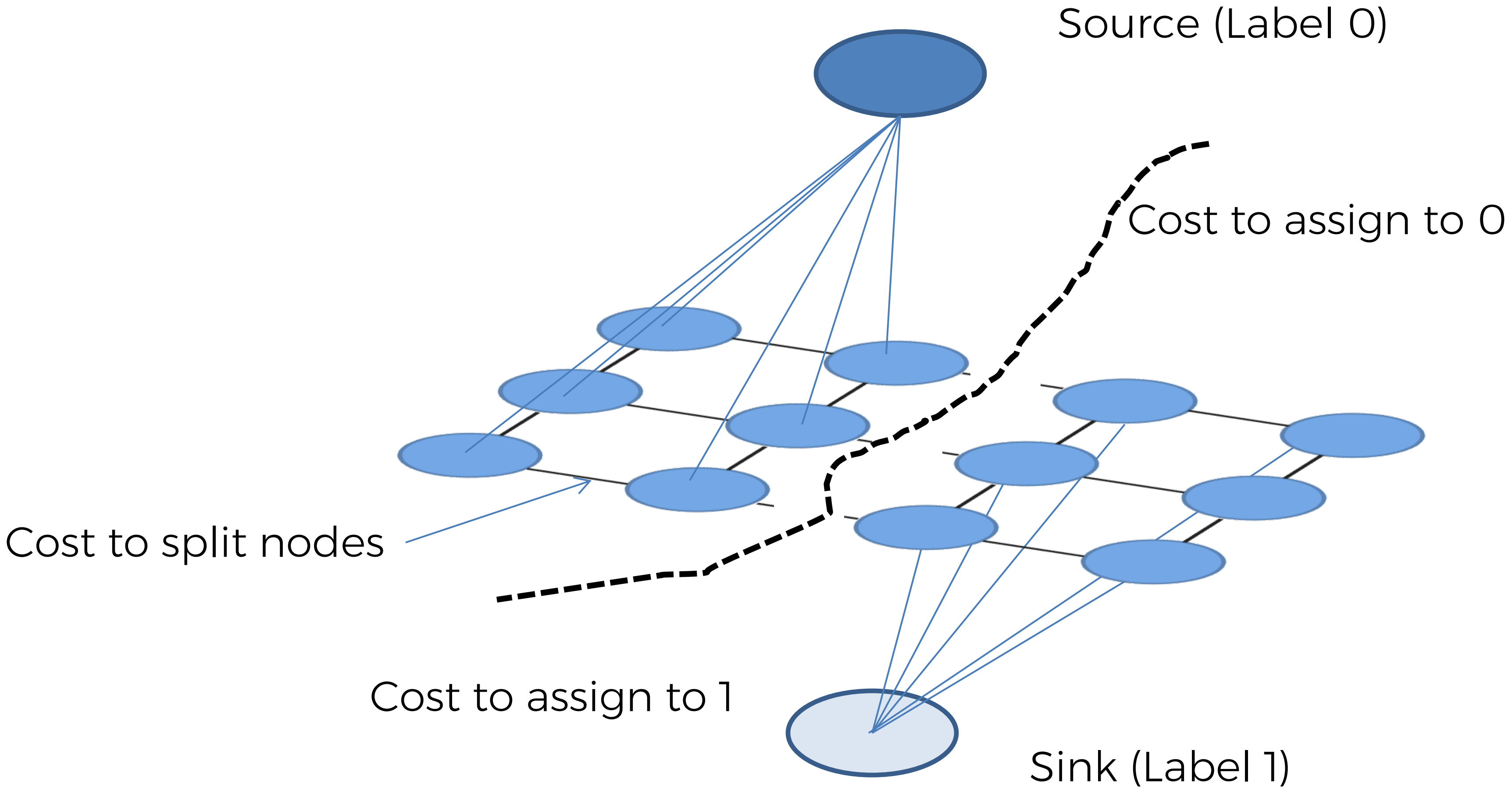
Solving MRFs with graph cuts



$$Energy(\mathbf{y};\theta,data)=\sum_i \psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

D. Hoiem

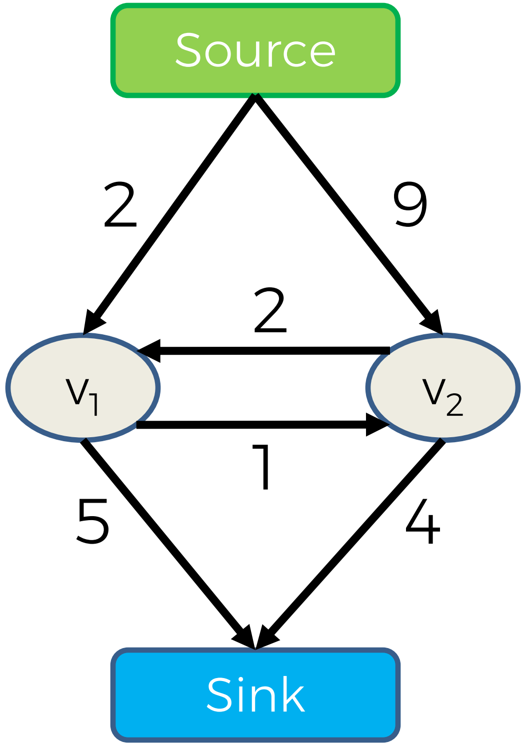
Solving MRFs with graph cuts



$$Energy(\mathbf{y};\theta,data)=\sum_i \psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

D. Hoiem

The *st*-Mincut Problem



Graph (V, E, C)

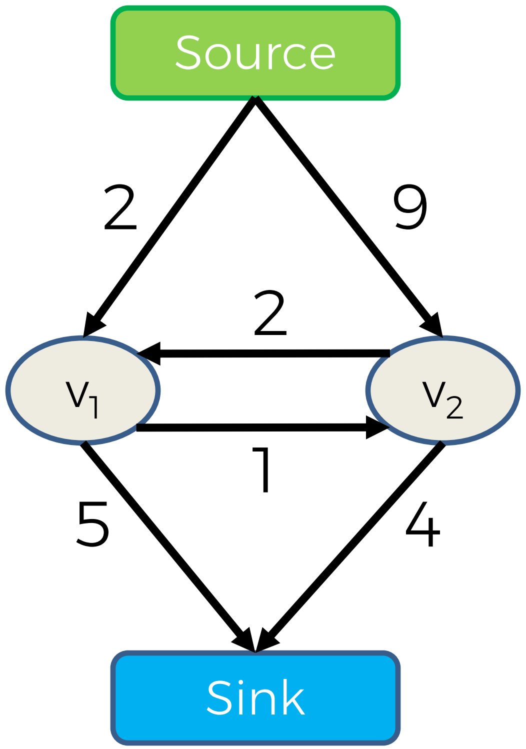
Vertices $V = \{v_1, v_2 \dots v_n\}$

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

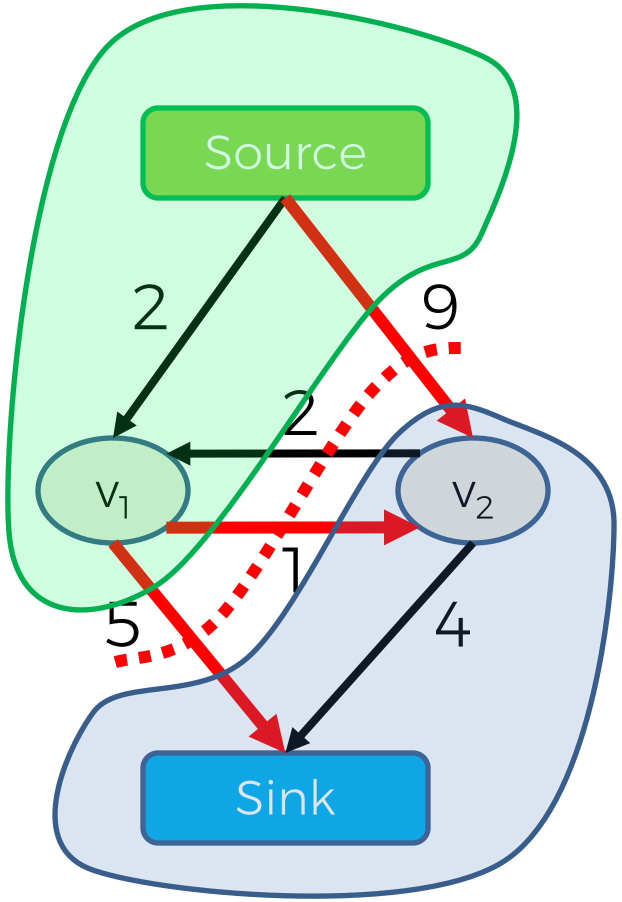
Costs $C = \{c_{(1, 2)} \dots\}$

The *st*-Mincut Problem

What is a st-cut?



The *st*-Mincut Problem



5 + 1 + 9 = 15

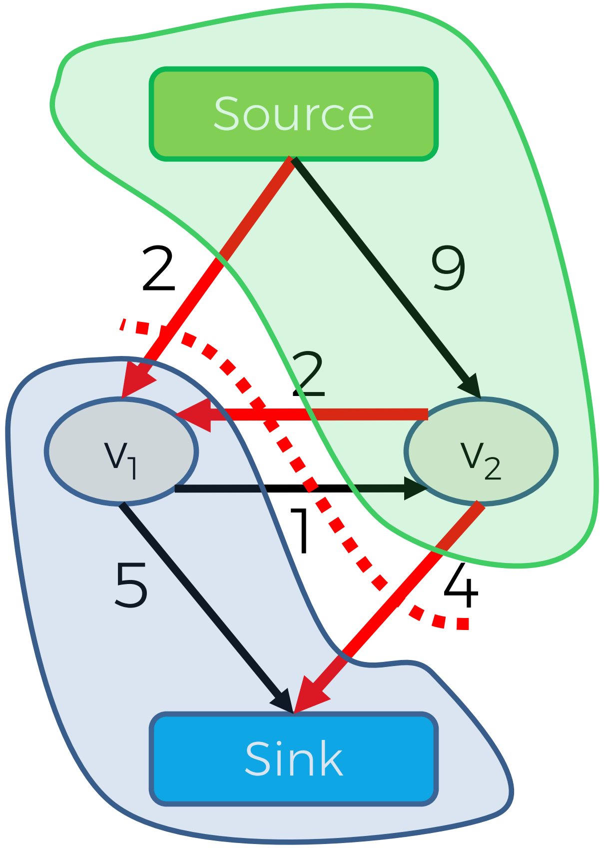
What is a st-cut?

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

What is the cost of a st-cut?

Sum of cost of all edges going from S to T

The st-Mincut Problem



2 + 2 + 4 = 8

What is a st-cut?

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

What is the cost of a st-cut?

Sum of cost of all edges going from S to T

What is the st-mincut?

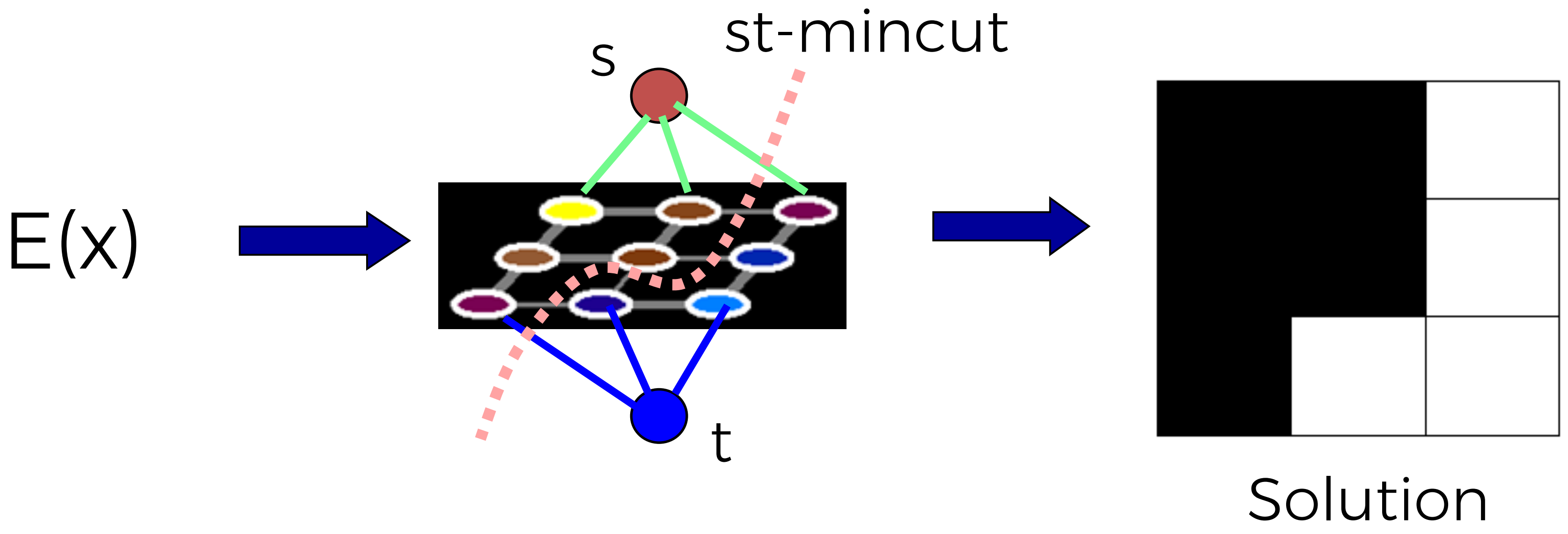
st-cut with the minimum cost

P. Kohli

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)



[Hammer, 1965] [Kolmogorov and Zabih, 2002]

P. Kohli

st-mincut and Energy Minimization

$$E(x) = \sum_i \theta_i (x_i) + \sum_{i,j} \theta_{ij} (x_i,x_j)$$

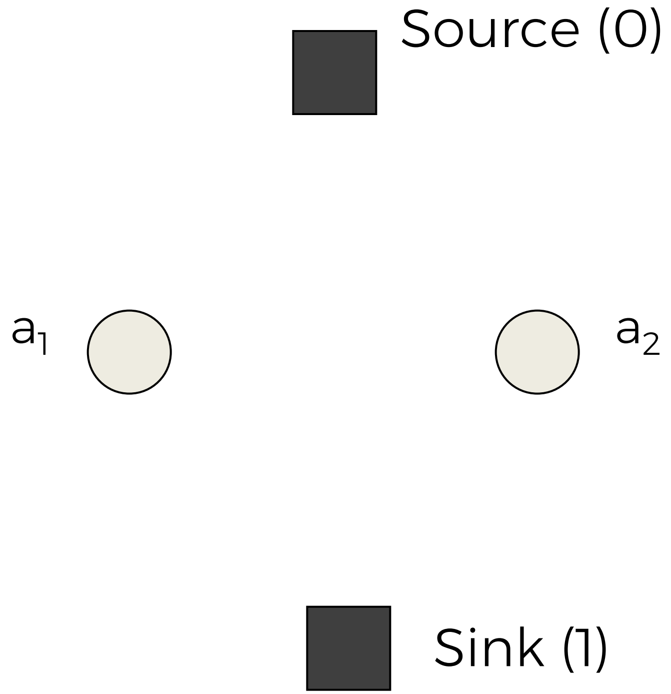
For all ij $\theta_{ij}(0,1) + \theta_{ij}(1,0) \geq \theta_{ij}(0,0) + \theta_{ij}(1,1)$

Equivalent (transformable)

$$E(x) = \sum_i c_i x_i + \sum_{i,j} c_{ij} x_i(1-x_j)$$ $c_{ij} \geq 0$

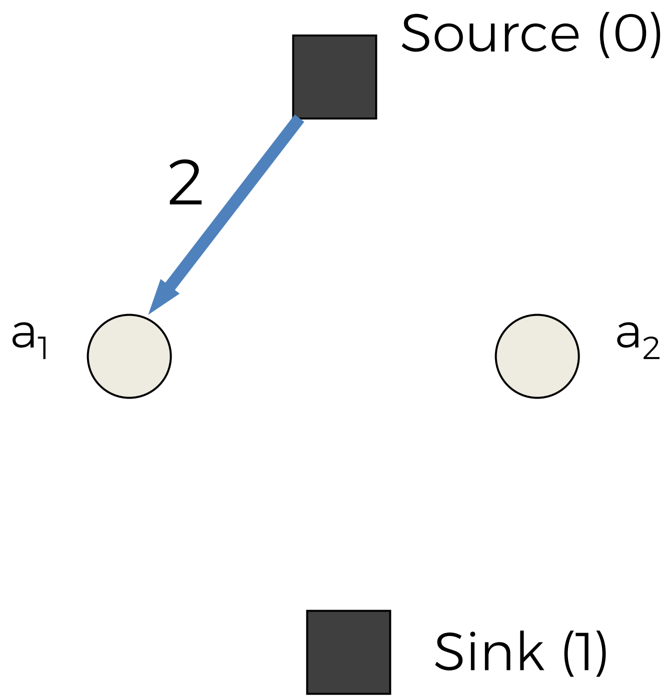
Graph Construction

$E(a_1,a_2)$



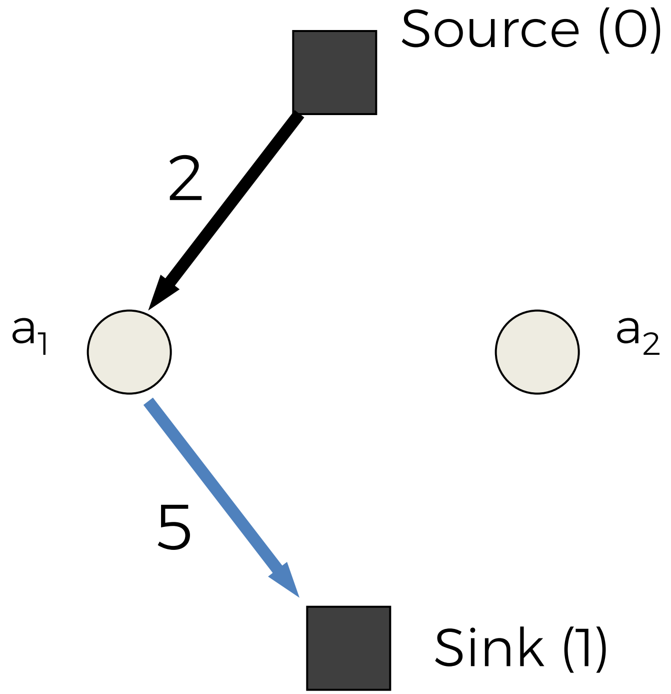
Graph Construction

$E(a_1,a_2) = 2a_1$



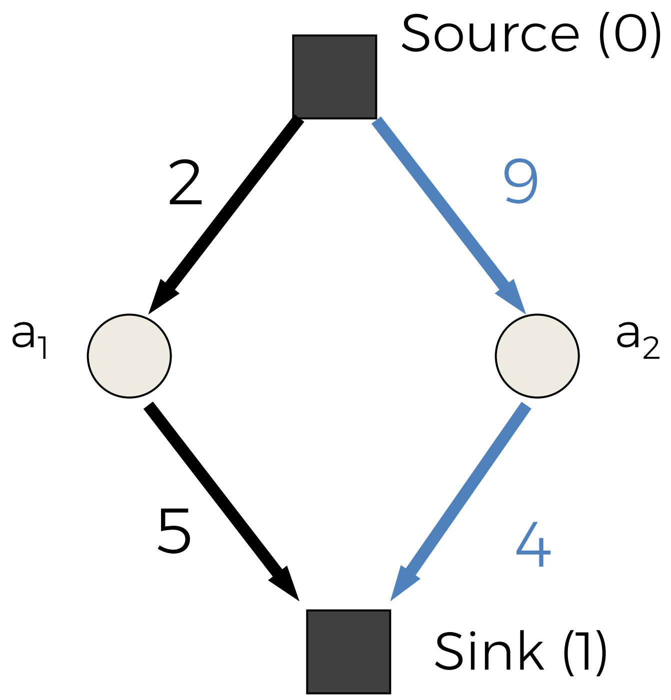
Graph Construction

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



Graph Construction

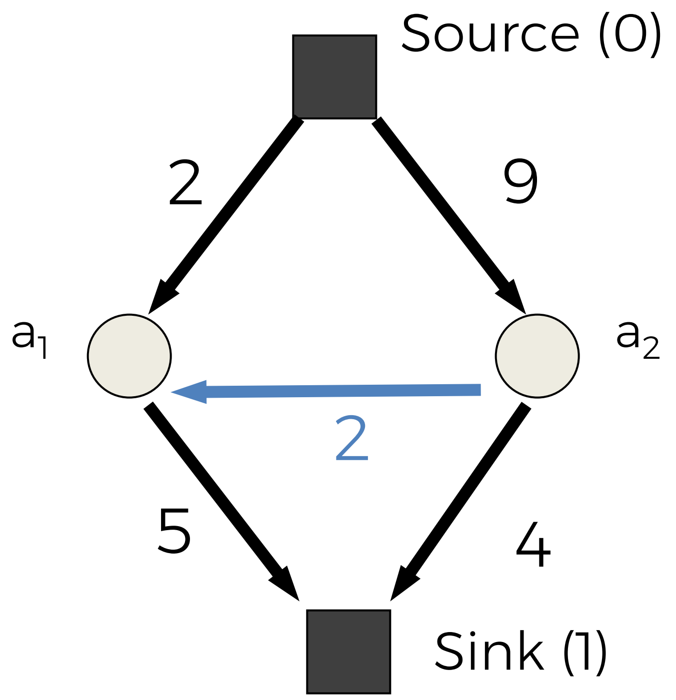
$$E(a_1,a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2$$



P. Kohli

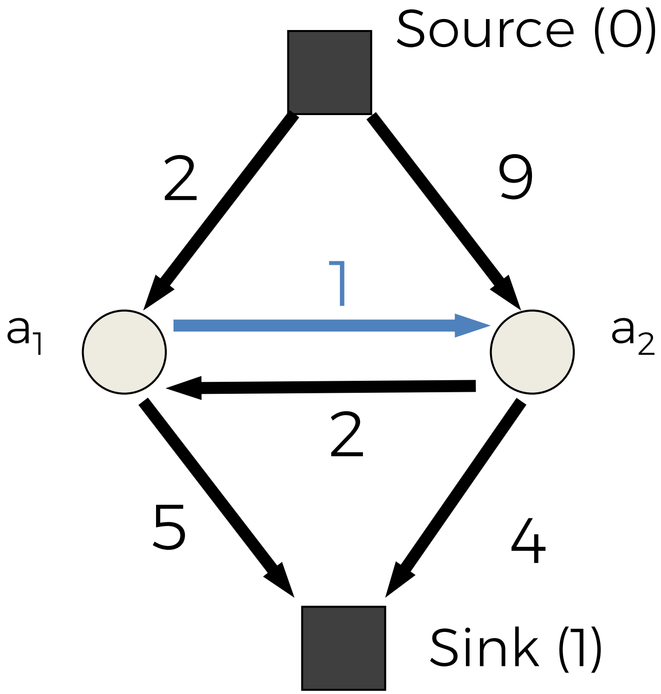
Graph Construction

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



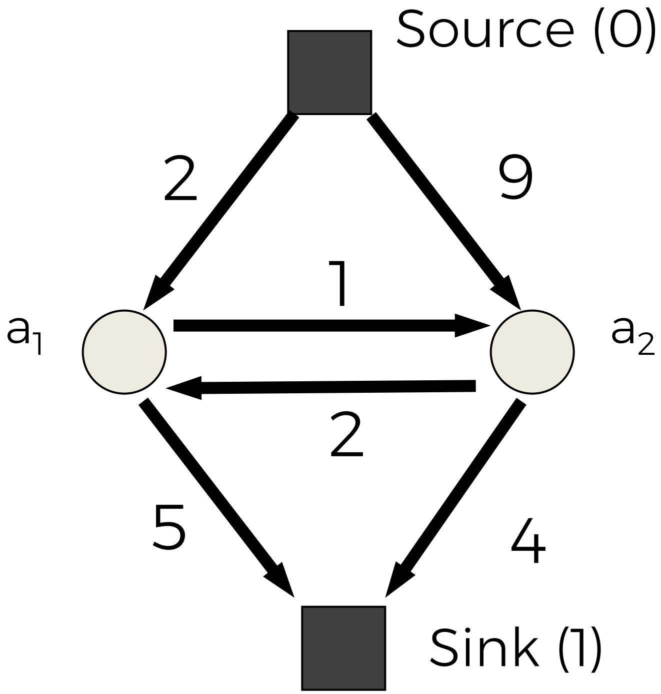
Graph Construction

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



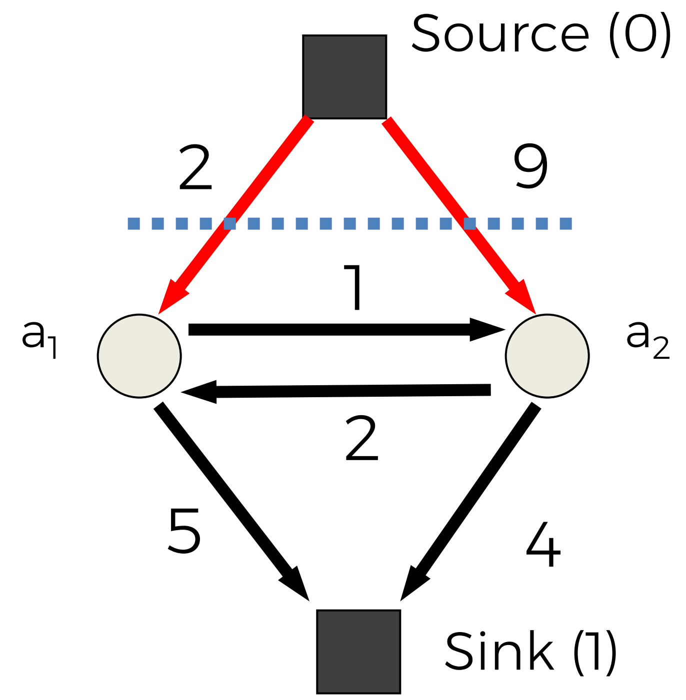
Graph Construction

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



Graph Construction

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



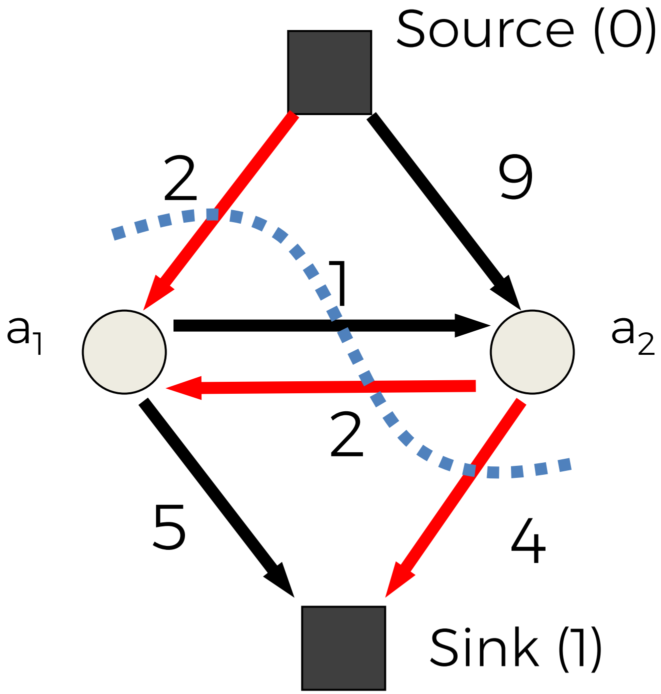
Cost of cut = 11

$a_1 = 1 \quad a_2 = 1$

E (1,1) = 11

Graph Construction

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

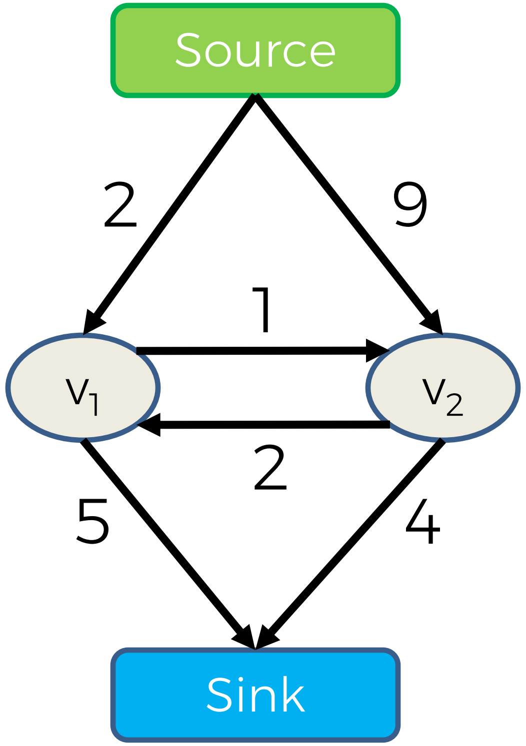


st-mincut cost = 8

$a_1 = 1 \quad a_2 = 0$

$E(1,0) = 8$

How to compute the st-mincut?



Solve the dual maximum flow problem

Compute the maximum flow between Source and Sink s.t.

Edges: Flow < Capacity

Nodes: Flow in = Flow out

Min-cut\Max-flow Theorem

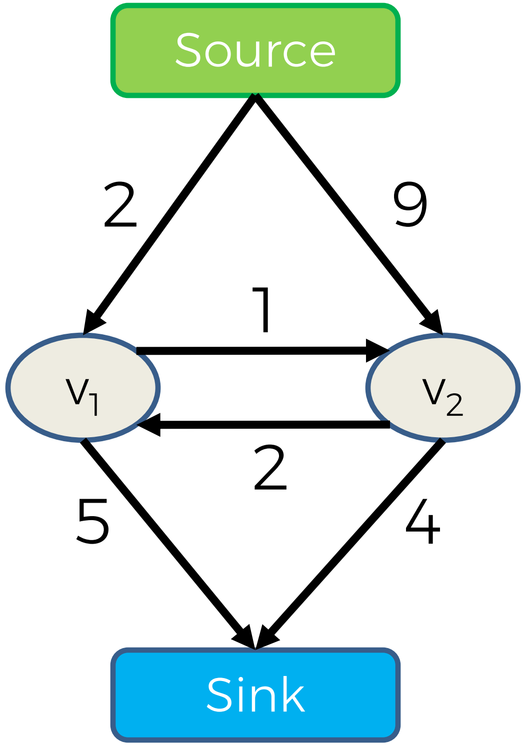
In every network, the maximum flow equals the cost of the st-mincut

Assuming non-negative capacity

P. Kohli

Maxflow Algorithms

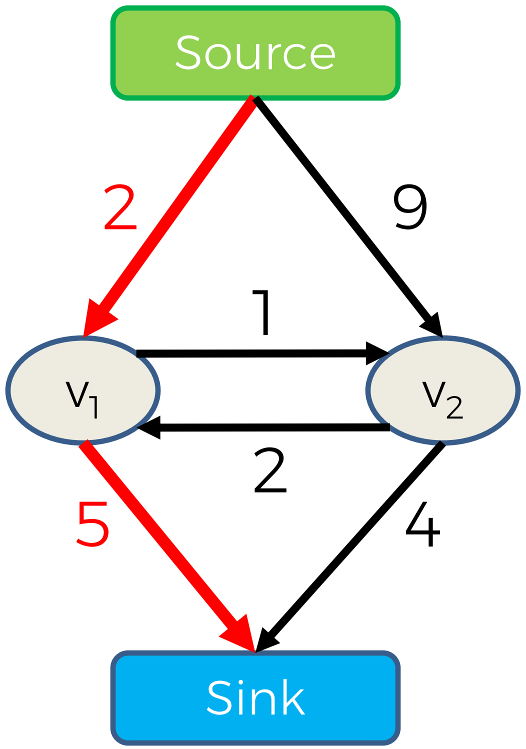
Flow = 0



Augmenting Path Based Algorithms

Maxflow Algorithms

Flow = 0

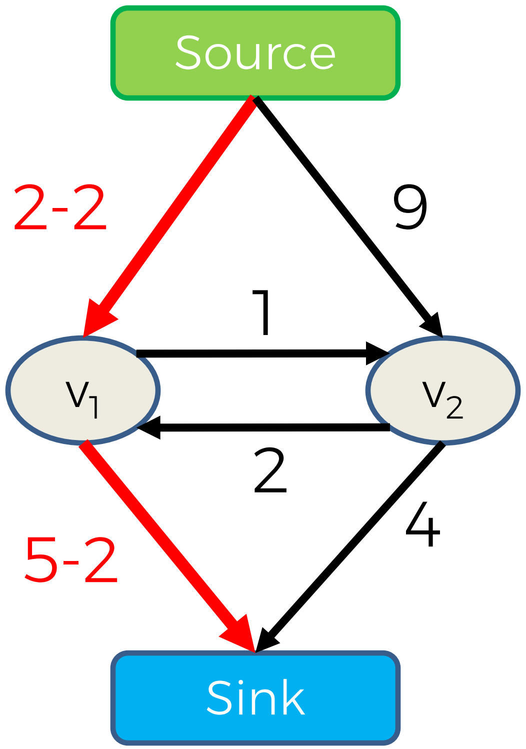


Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity

Maxflow Algorithms

Flow = 0 + 2

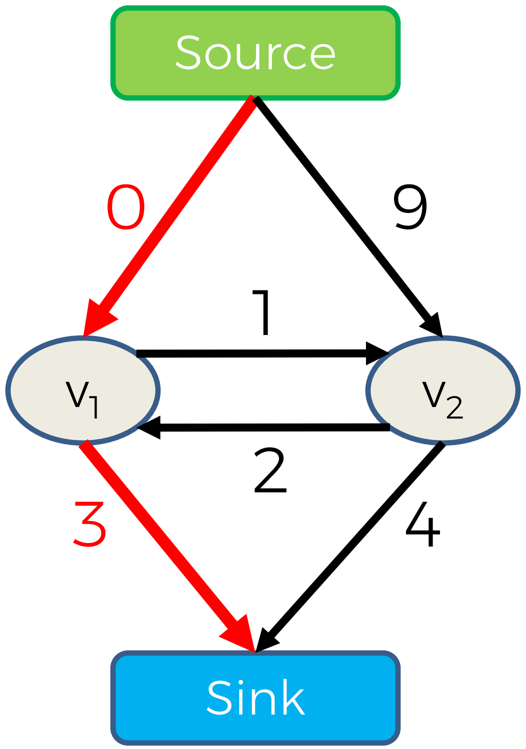


Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path

Maxflow Algorithms

Flow = 2



Augmenting Path Based Algorithms

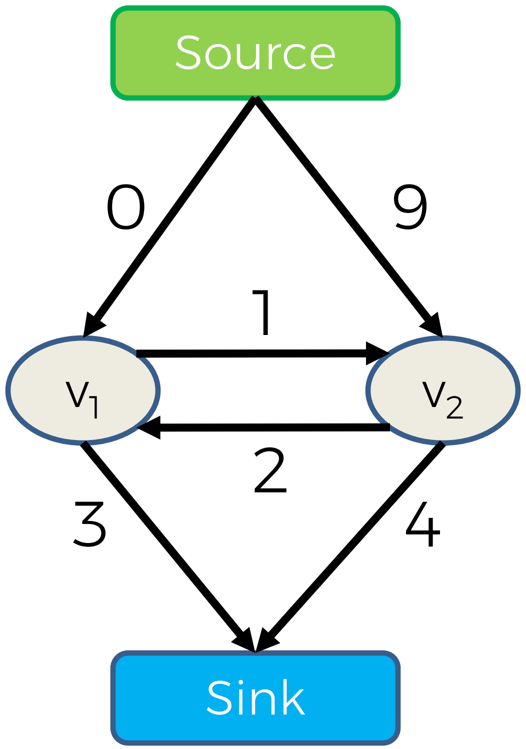
1. Find path from source to sink with positive capacity

2. Push maximum possible flow through this path

P. Kohli

Maxflow Algorithms

Flow = 2



Augmenting Path Based Algorithms

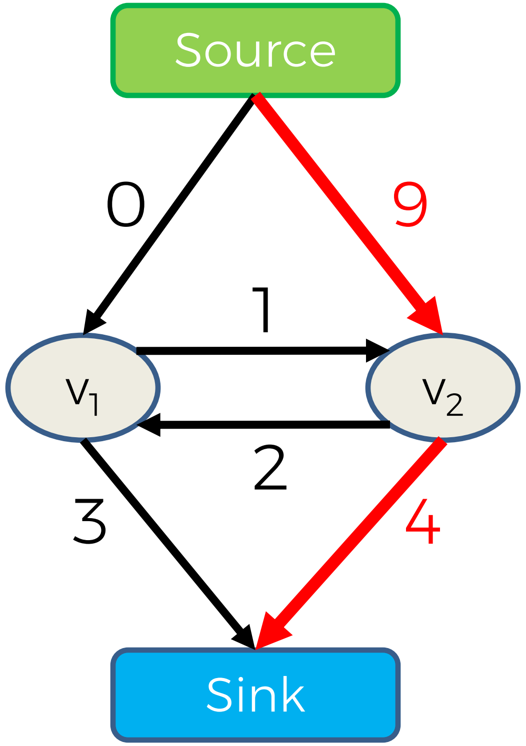
1. Find path from source to sink with positive capacity

2. Push maximum possible flow through this path

3. Repeat until no path can be found

Maxflow Algorithms

Flow = 2

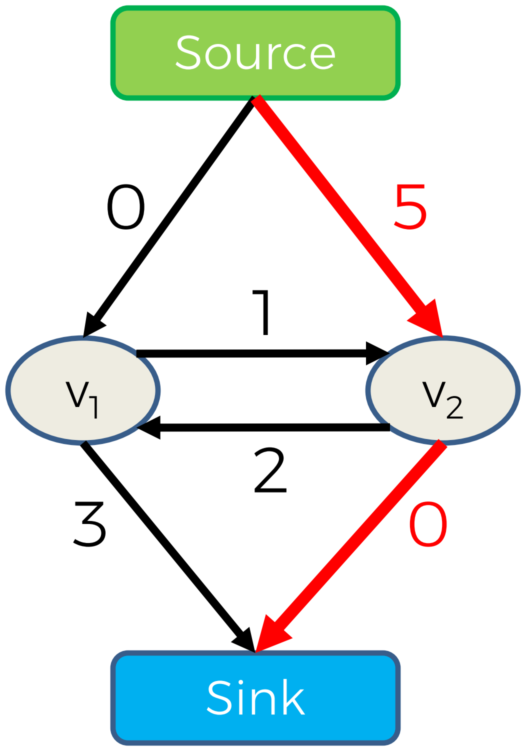


Augmenting Path Based Algorithms

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

Maxflow Algorithms

Flow = 2 + 4

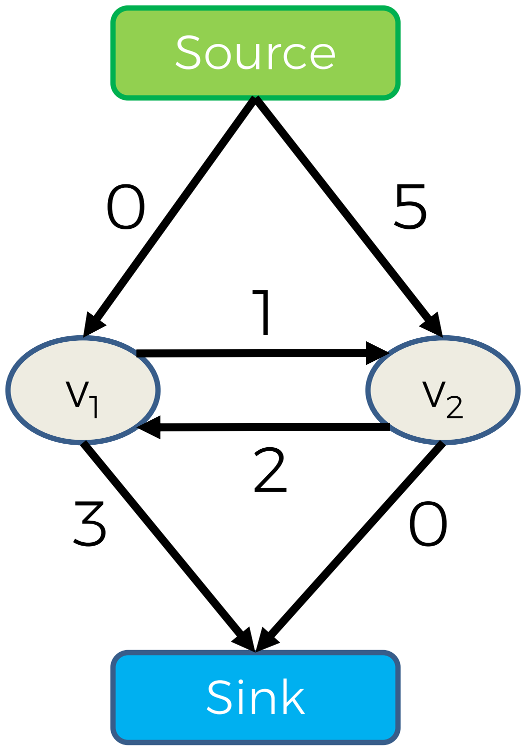


Augmenting Path Based Algorithms

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

Maxflow Algorithms

Flow = 6



Augmenting Path Based Algorithms

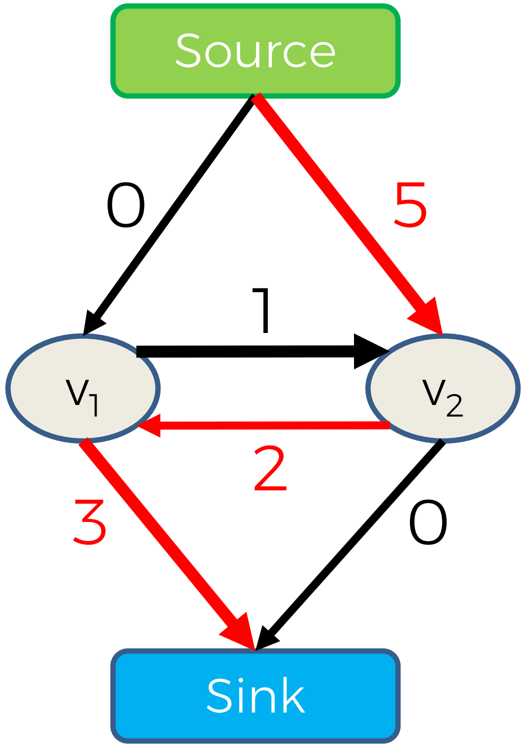
1. Find path from source to sink with positive capacity

2. Push maximum possible flow through this path

3. Repeat until no path can be found

Maxflow Algorithms

Flow = 6



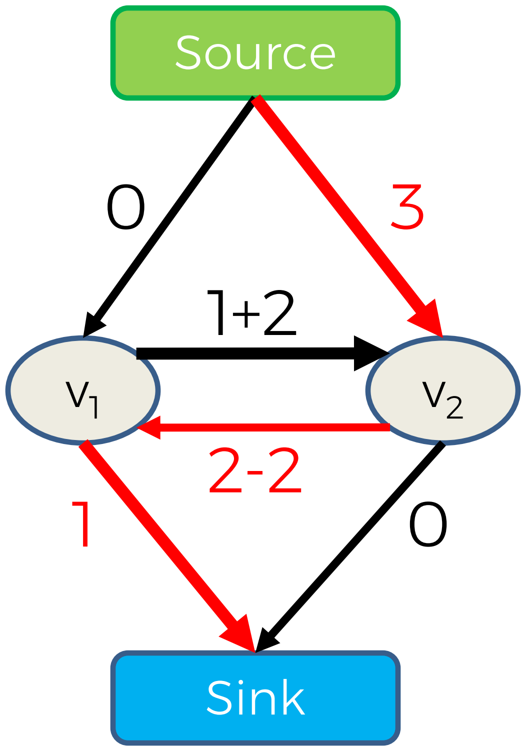
Augmenting Path Based Algorithms

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

P. Kohli

Maxflow Algorithms

Flow = 6 + 2

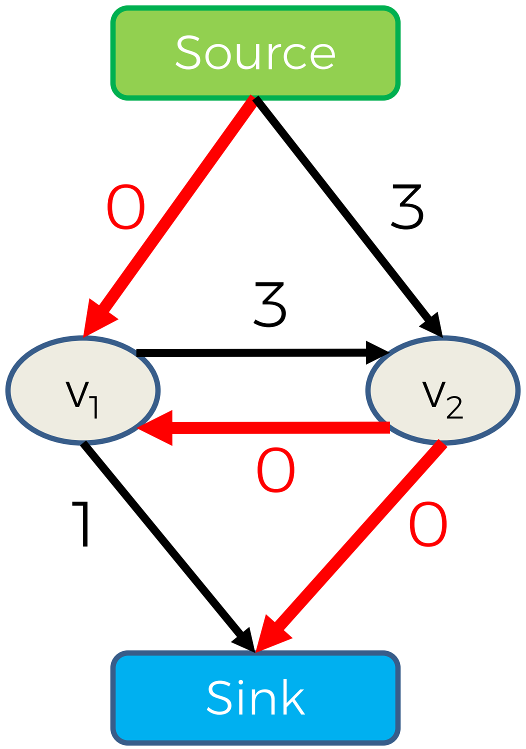


Augmenting Path Based Algorithms

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

Maxflow Algorithms

Flow = 8



Augmenting Path Based Algorithms

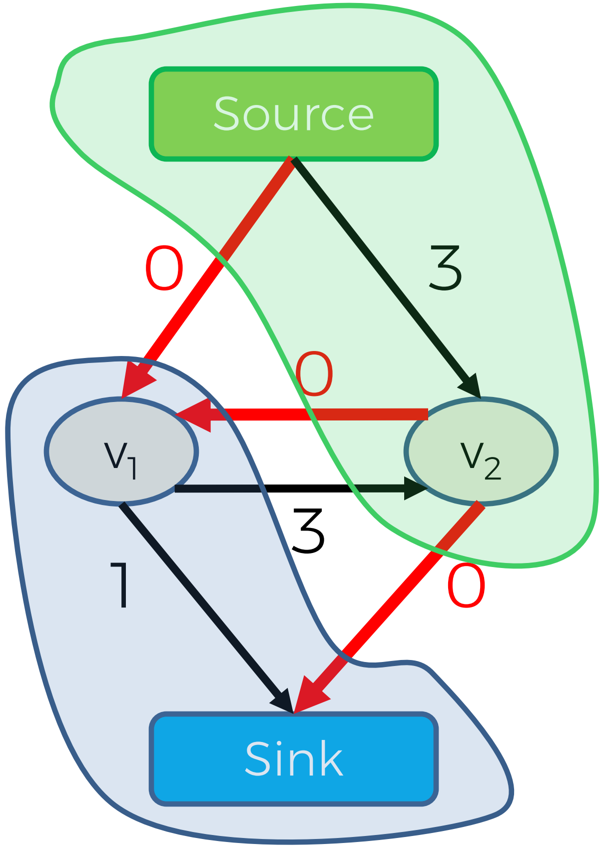
1. Find path from source to sink with positive capacity

2. Push maximum possible flow through this path

3. Repeat until no path can be found

Maxflow Algorithms

Flow = 8

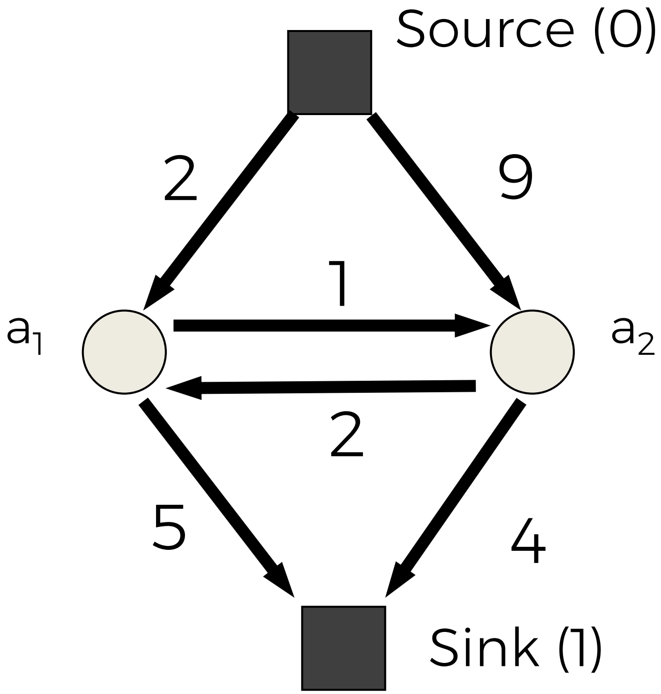


Augmenting Path Based Algorithms

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

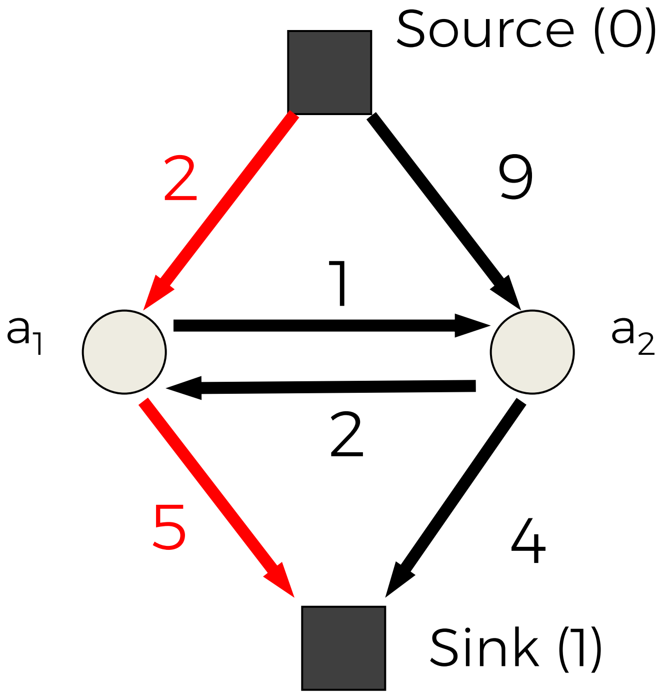
Flow and Reparametrization

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



Flow and Reparametrization

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



$2a_1 + 5\bar{a}_1$

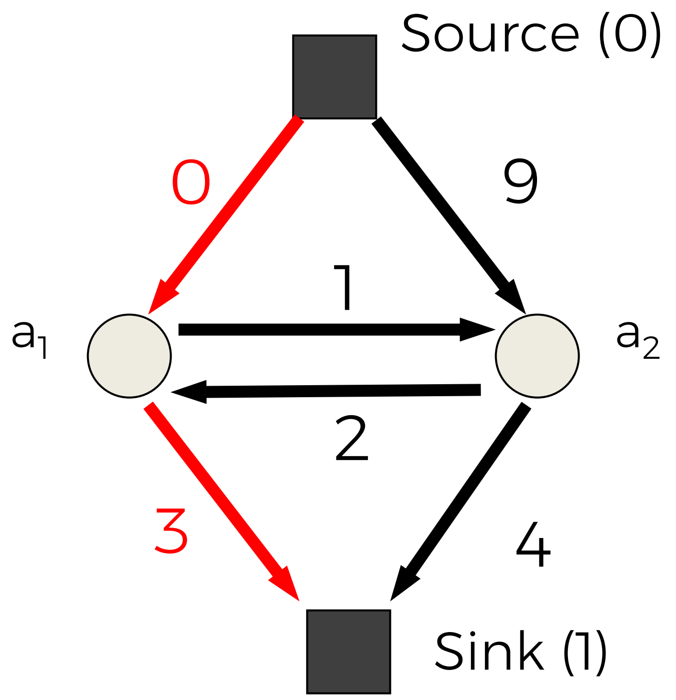
$= 2(a_1+\bar{a}_1) + 3\bar{a}_1$

$= 2 + 3\bar{a}_1$

P. Kohli

Flow and Reparametrization

$$E(a_1,a_2) = 2 + 3\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2$$



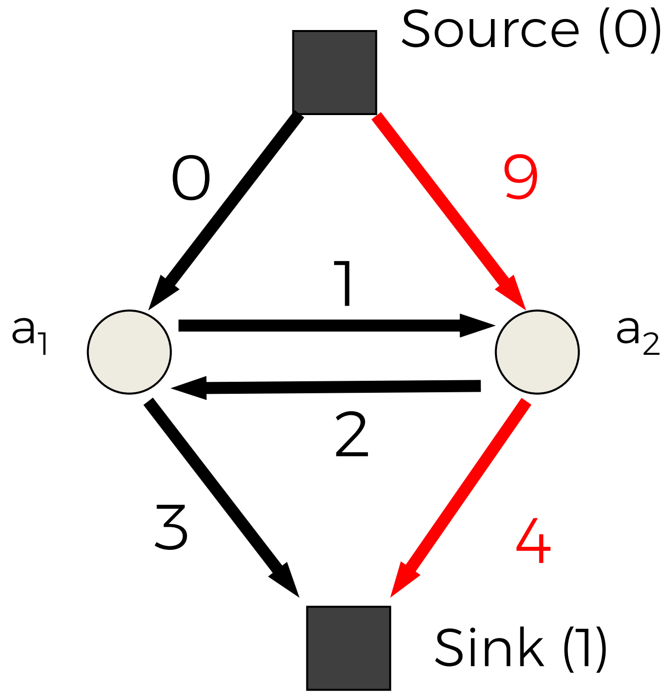
$2a_1 + 5\bar{a}_1$

$= 2(a_1+\bar{a}_1) + 3\bar{a}_1$

$= 2 + 3\bar{a}_1$

Flow and Reparametrization

$$E(a_1,a_2) = 2 + 3\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1 a_2$$



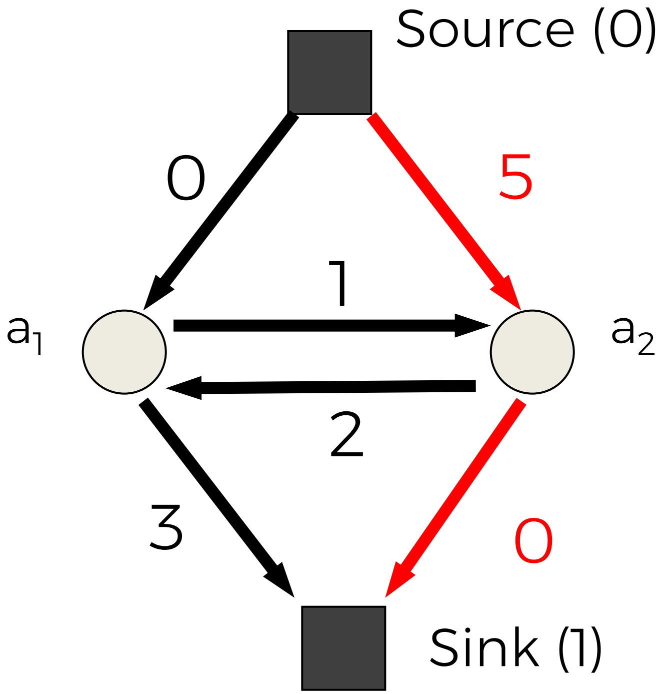
$9a_2 + 4\bar{a}_2$

$= 4(a_2+\bar{a}_2) + 5\bar{a}_2$

$= 4 + 5\bar{a}_2$

Flow and Reparametrization

$$E(a_1,a_2) = 2 + 3\bar{a}_1 + 5a_2 + 4 + 2a_1\bar{a}_2 + \bar{a}_1a_2$$



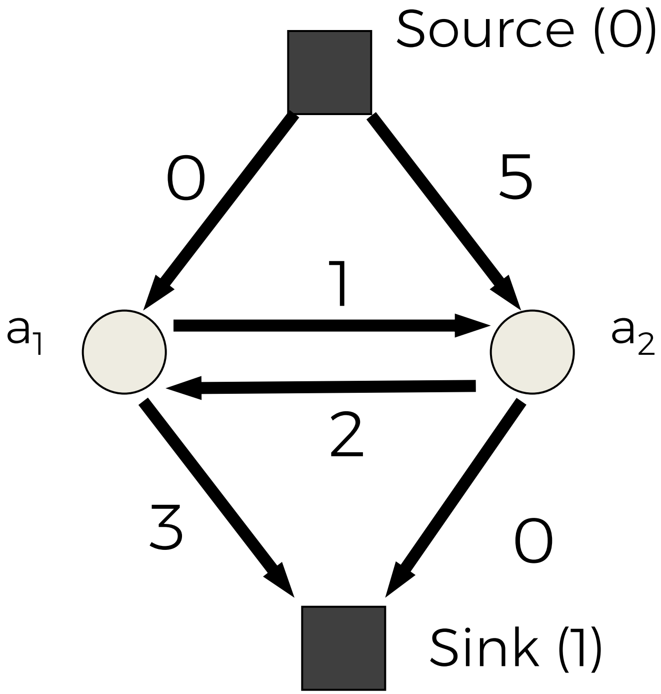
$9a_2 + 4\bar{a}_2$

$= 4(a_2+\bar{a}_2) + 5\bar{a}_2$

$= 4 + 5\bar{a}_2$

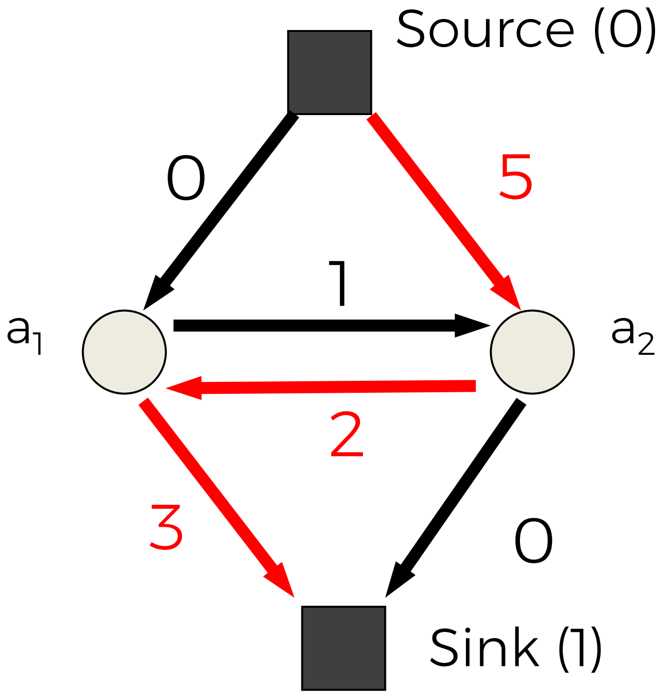
Flow and Reparametrization

$$E(a_1,a_2) = 6 + 3\bar{a}_1 + 5a_2 + 2a_1\bar{a}_2 + \bar{a}_1 a_2$$



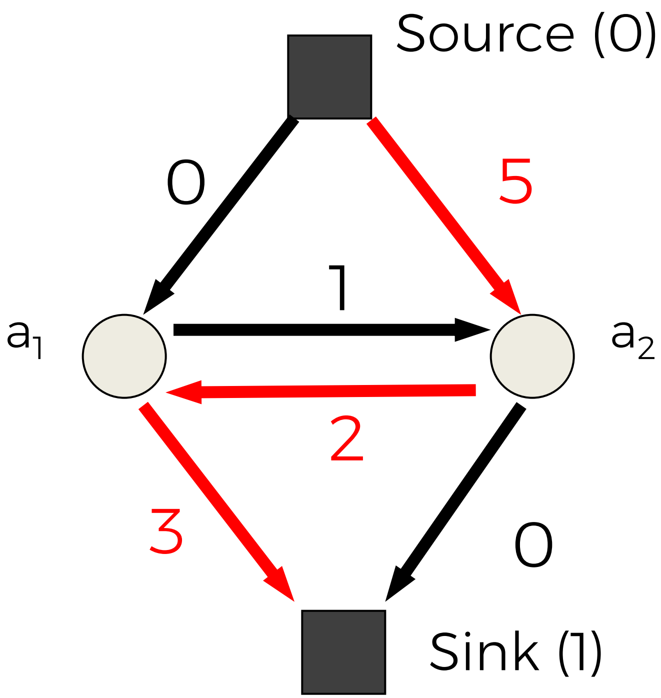
Flow and Reparametrization

$$E(a_1,a_2) = 6 + 3\bar{a}_1 + 5a_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2$$



Flow and Reparametrization

$$E(a_1,a_2) = 6 + 3\bar{a}_1 + 5a_2 + 2a_1\bar{a}_2 + \bar{a}_1 a_2$$



$3\bar{a}_1 + 5a_2 + 2a_1\bar{a}_2$

$= 2(\bar{a}_1+a_2+a_1\bar{a}_2) + \bar{a}_1 + 3a_2$

$= 2(1+\bar{a}_1 a_2) + \bar{a}_1 + 3a_2$

$F_1 = \bar{a}_1 + a_2 + a_1\bar{a}_2$

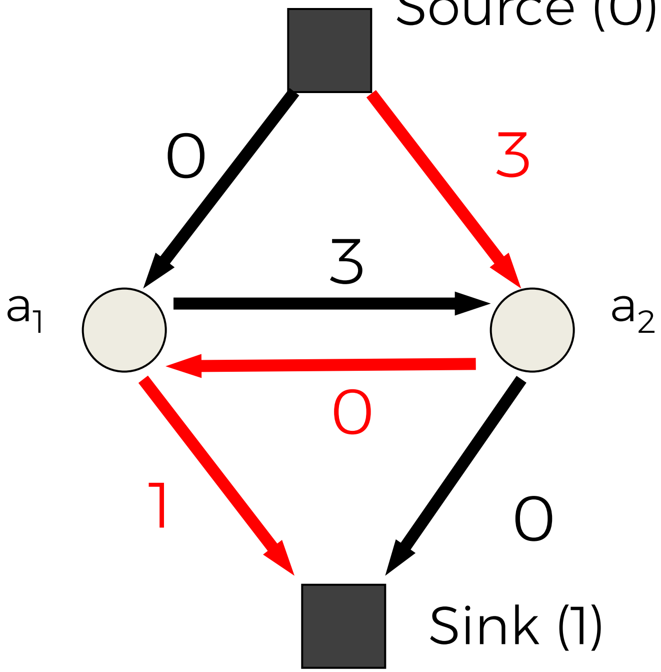
$F_2 = 1 + \bar{a}_1 a_2$

a_1	a_2	F_1	F_2
0	0	1	1
0	1	2	2
1	0	1	1
1	1	1	1

P. Kohli

Flow and Reparametrization

$$E(a_1,a_2) = 8 + \bar{a}_1 + 3a_2 + 3\bar{a}_1 a_2$$



$3\bar{a}_1 + 5a_2 + 2a_1\bar{a}_2$

$= 2(\bar{a}_1 + a_2 + a_1\bar{a}_2) + \bar{a}_1 + 3a_2$

$= 2(1 + \bar{a}_1 a_2) + \bar{a}_1 + 3a_2$

$F_1 = \bar{a}_1 + a_2 + a_1\bar{a}_2$

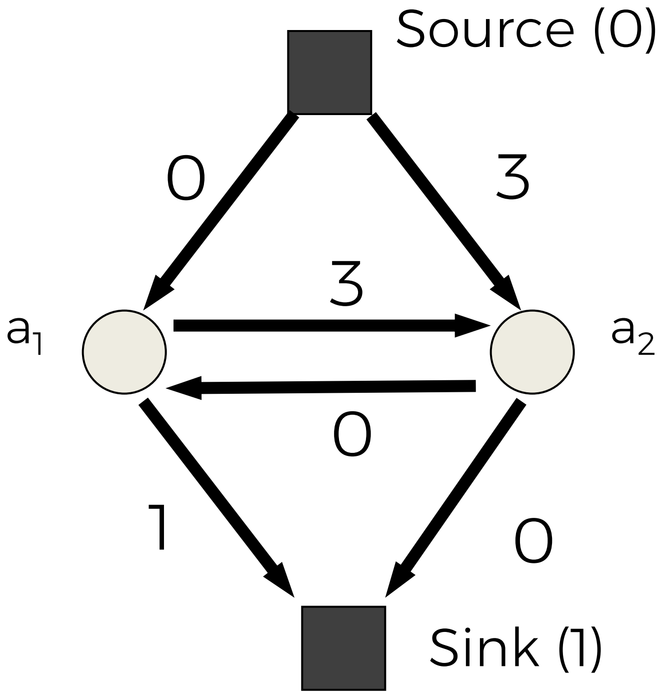
$F_2 = 1 + \bar{a}_1 a_2$

a_1	a_2	F_1	F_2
0	0	1	1
0	1	2	2
1	0	1	1
1	1	1	1

P. Kohli

Flow and Reparametrization

$$E(a_1,a_2) = 8 + \bar{a}_1 + 3a_2 + 3\bar{a}_1 a_2$$



No more augmenting paths possible

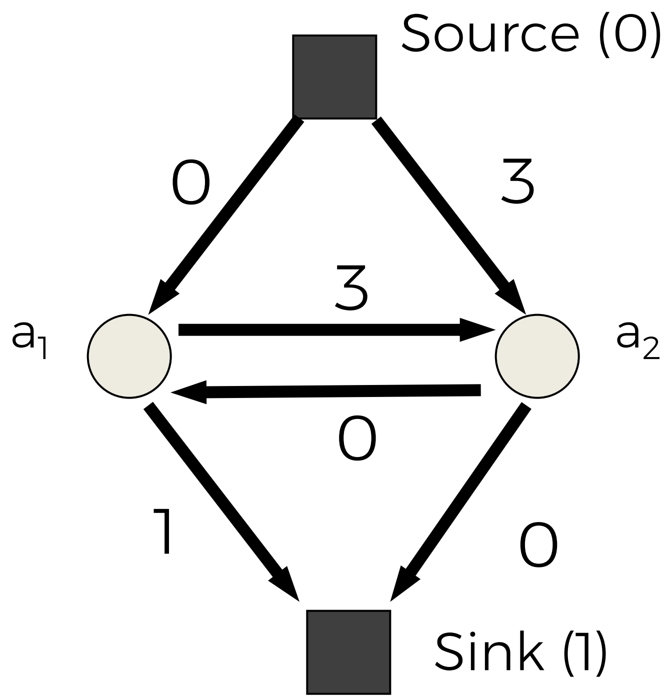
Flow and Reparametrization

$E(a_1,a_2) = \boxed{8} + \boxed{\bar{a}_1 + 3a_2 + 3\bar{a}_1 a_2}$

Residual Graph
(positive coefficients)

Total Flow

bound on the
optimal solution



Tight Bound >> Inference of the optimal solution becomes trivial

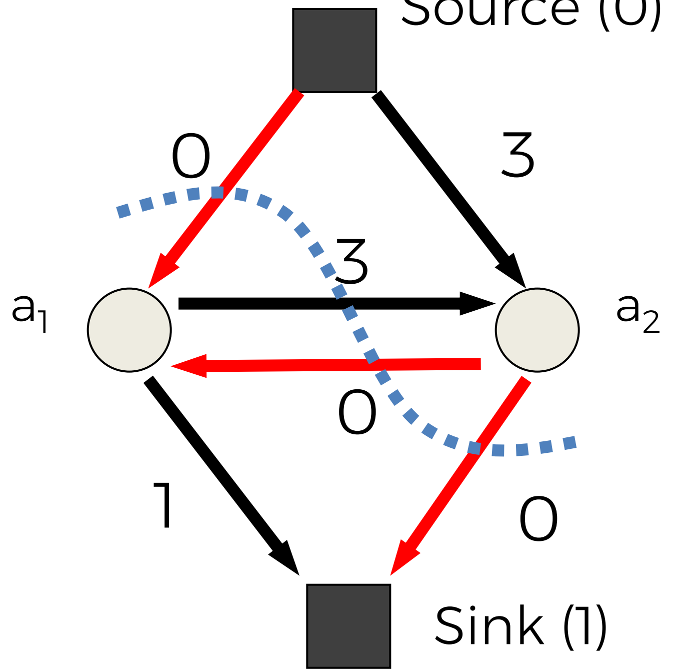
Flow and Reparametrization

$E(a_1,a_2) = \boxed{8} + \boxed{\bar{a}_1 + 3a_2 + 3\bar{a}_1 a_2}$

Residual Graph
(positive coefficients)

Total Flow

bound on the energy
of the optimal
solution



st-mincut cost = 8

$a_1 = 1 \;\; a_2 = 0$

E (1,0) = 8

P. Kohli

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

Code for Image Segmentation

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

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

$0 \rightarrow fg$

$1 \rightarrow bg$

n = number of pixels

Global Minimum (x^*)

$$x^* = \arg\min_x E(x)$$

How to minimize E(x)?

P. Kohli

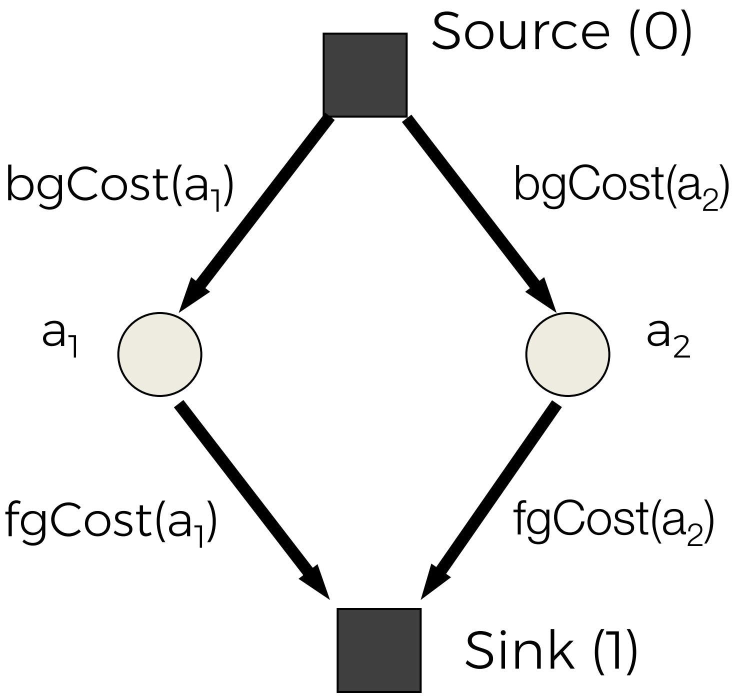
How does the code look like?

```
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)
```

P. Kohli

How does the code look like?

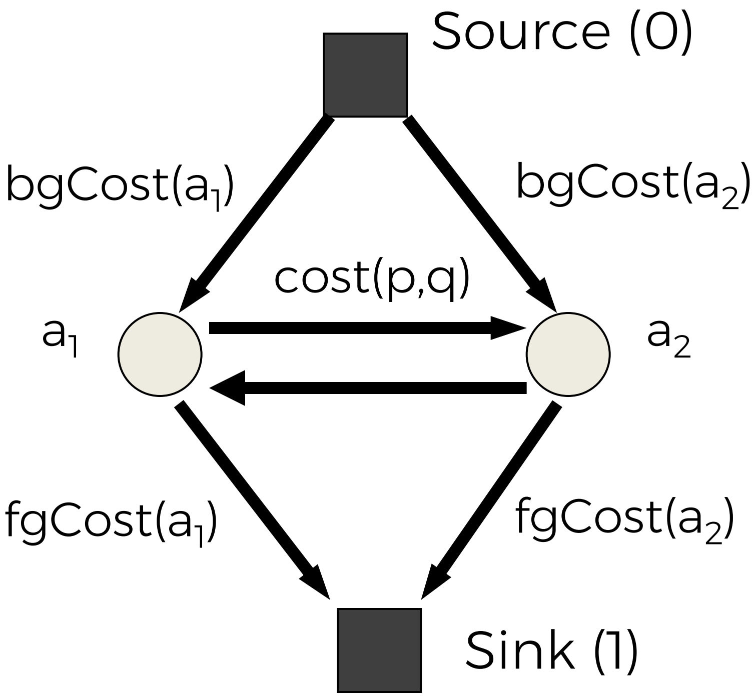
```
Graph *g;

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)
```



P. Kohli

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)
```

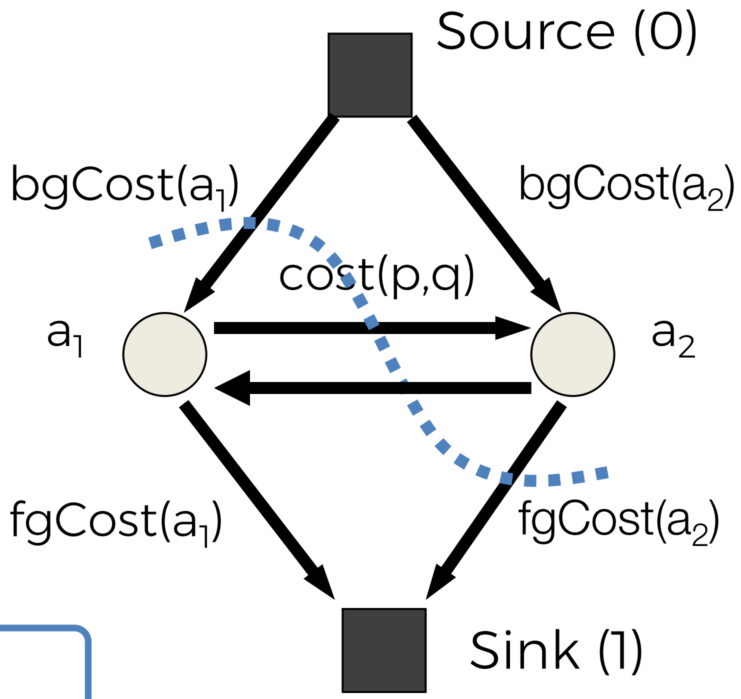


P. Kohli

How does the code look like?

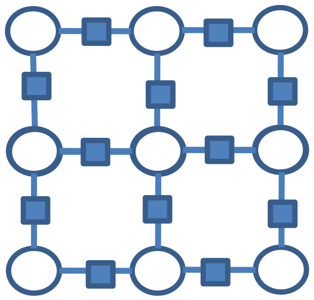
```
Graph *g;

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

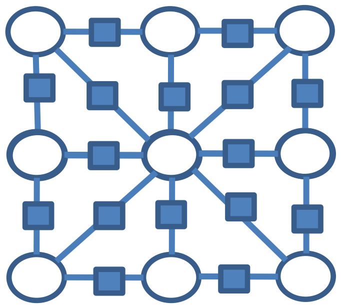
Random Fields in Vision



4-connected;
pairwise MRF

$$E(x) = \sum_{i,j \in N_4} \theta_{ij}(x_i, x_j)$$

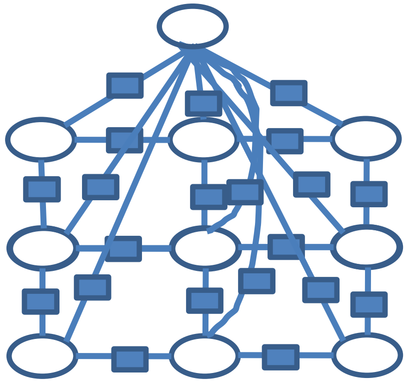
Order 2



higher(8)-connected;
pairwise MRF

$$E(x) = \sum_{i,j \in N_8} \theta_{ij}(x_i, x_j)$$

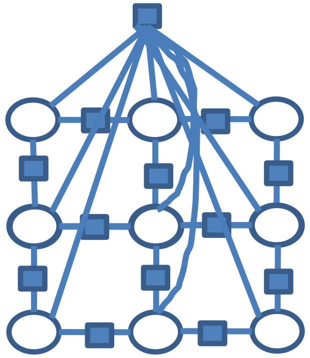
Order 2



MRF with
global variables

$$E(x) = \sum_{i,j \in N_8} \theta_{ij}(x_i, x_j)$$

Order 2



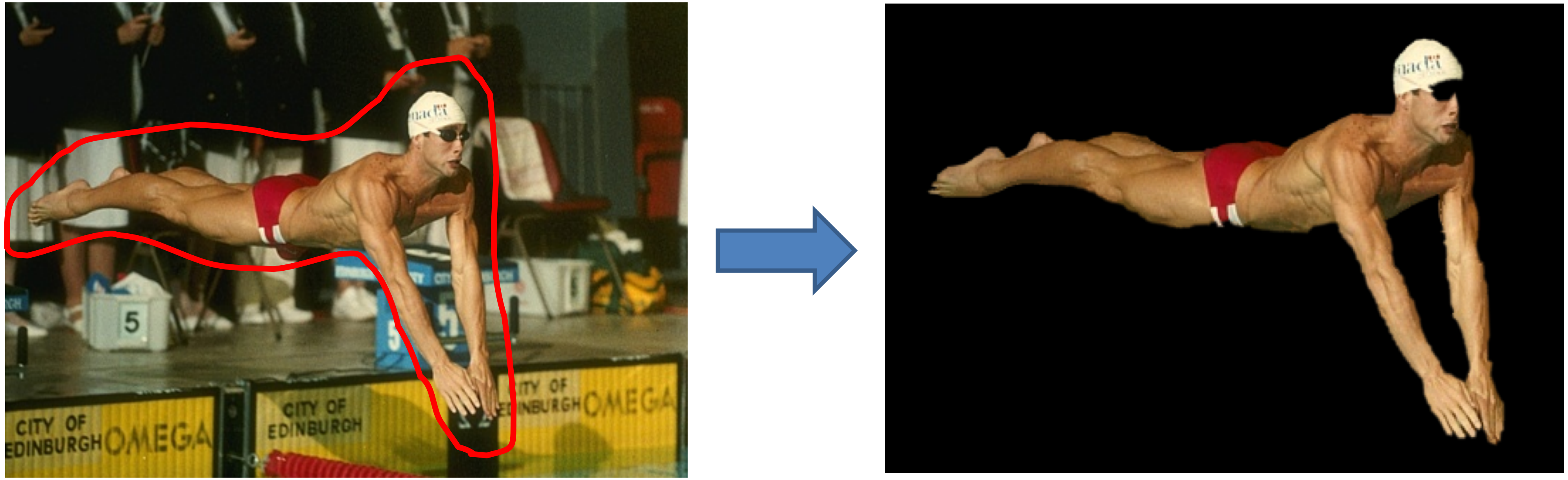
Higher-order MRF

$$E(x) = \sum_{i,j \in N_4} \theta_{ij}(x_i, x_j) + \theta(x_1, \ldots, x_n)$$

Order n

C. Rother

GrabCut segmentation



User provides rough indication of foreground region.

Goal: Automatically provide a pixel-level segmentation.

D. Hoiem

MRF with global potential - GrabCut model

[Rother et. al. '04]

$$E(x,\theta^F,\theta^B) = \sum_i F_i(\theta^F)x_i + B_i(\theta^B)(1-x_i) \quad + \sum_{i,j \in N} |x_i - x_j|$$

$$F_i = -\log \Pr(z_i|\theta^F) \qquad B_i = -\log \Pr(z_i|\theta^B)$$

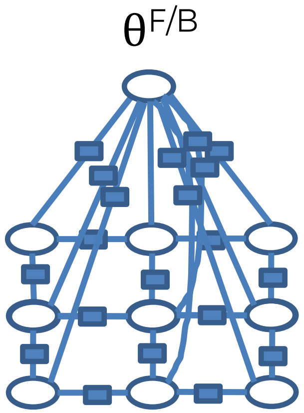
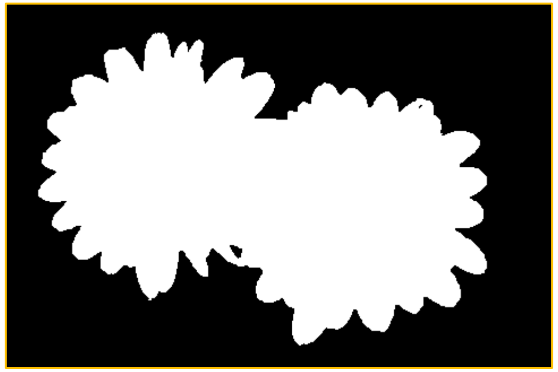
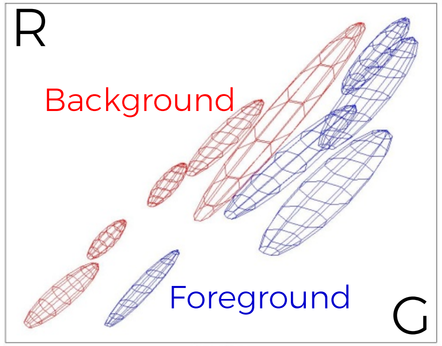


Image z



Output x



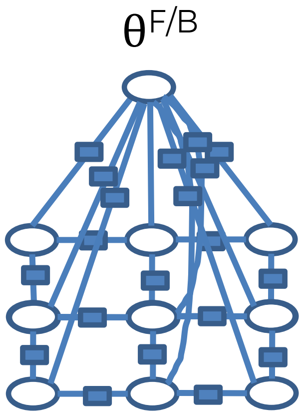
$\theta^{F/B}$ Gaussian Mixture models

Problem: for unknown x,θ^F,θ^B the optimization is NP-hard!
[Vicente et al. '09]

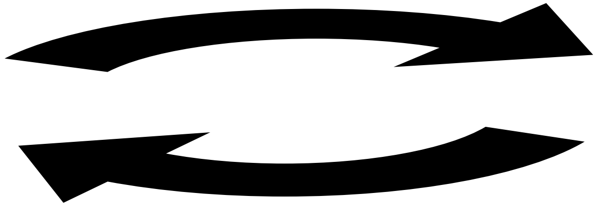
C. Rother

GrabCut: Iterated Graph Cuts

[Rother et al. Siggraph ‘04]



$$\min_{\theta^F, \theta^B} E(x, \theta^F, \theta^B)$$



$$\min_{x} E(x, \theta^F, \theta^B)$$

Learning of the colour distributions

Graph cut to infer segmentation

Most systems with global variables work like that
e.g. [ObjCut Kumar et. al. ‘05, PoseCut Bray et al. ’06, LayoutCRF Winn et al. ’06]

GrabCut: Iterated Graph Cuts

1. Define graph
   – usually 4-connected or 8-connected
2. Define unary potentials
   – Color histogram or mixture of Gaussians for background and foreground

$$unary\_potential(x) = -\log\left(\frac{P(c(x);\theta_{foreground})}{P(c(x);\theta_{background})}\right)$$

3. Define pairwise potentials

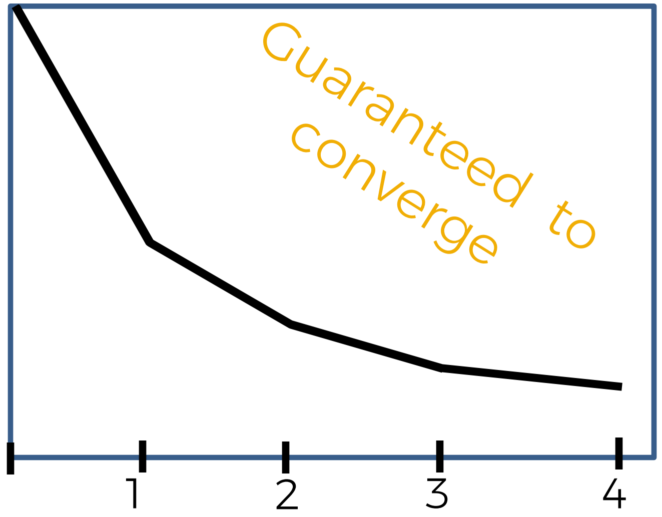
$$edge\_potential(x,y) = k_1 + k_2 \exp\left\{\frac{-\left\|c(x)-c(y)\right\|^2}{2\sigma^2}\right\}$$

4. Apply graph cuts
5. Return to 2, using current labels to compute foreground, background models

D. Hoiem

GrabCut: Iterated Graph Cuts

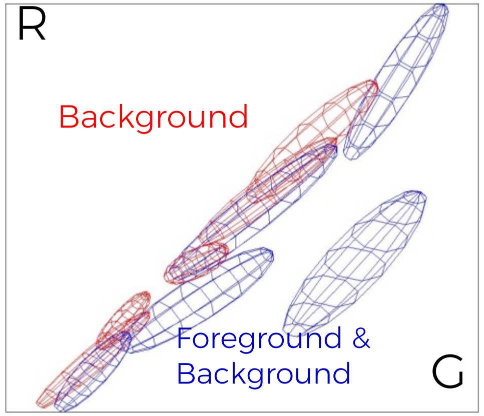
Result



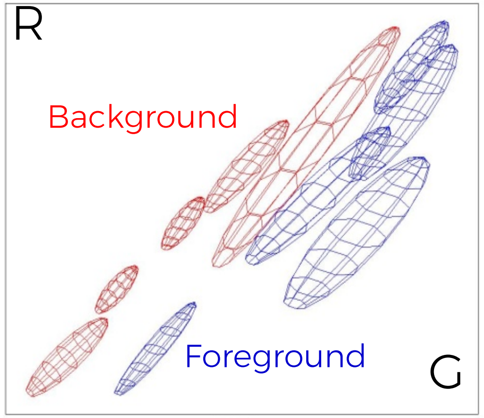
Energy after each Iteration

C. Rother

Colour Model



Iterated
graph cut



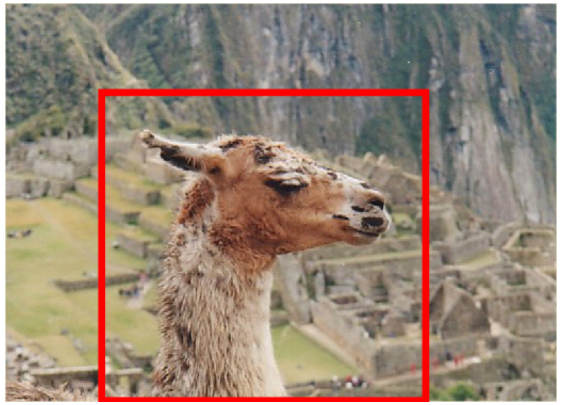
C. Rother

Optimizing over θ's help

Input

no iteration
[Boykov&Jolly '01]

after convergence
[GrabCut '04]

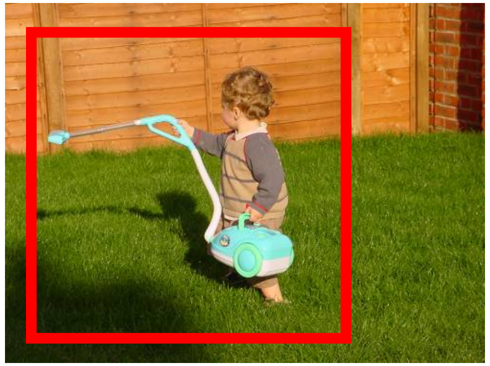


Input

after convergence
[GrabCut '04]

C. Rother

What is easy or hard about these cases for graphcut-based segmentation?



Easier examples

D. Hoiem

More difficult Examples

Camouflage & Low Contrast

Fine structure

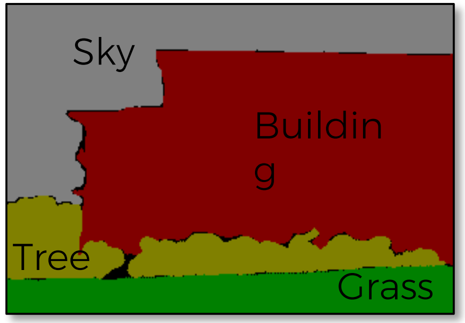
Harder Case

Initial Rectangle

Initial Result

Semantic Segmentation

Joint Object recognition & segmentation

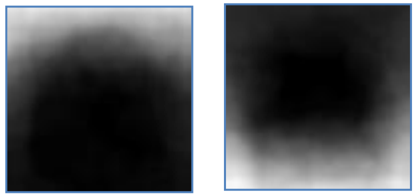


$$E(x,\omega) = \sum_i \theta_i(\omega, x_i) + \sum_i \theta_i(x_i) + \sum_i \theta_i(x_i) + \sum_{i,j} \theta_{ij}(x_i,x_j)$$

(color) (location) (class) (edge aware Ising prior)

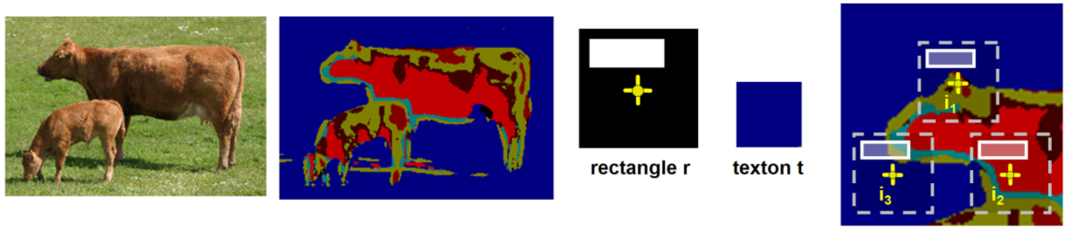
$x_i \in \{1,\dots,K\}$ for K object classes

Location



sky grass

Class (boosted textons)



(a) Input image (b) Texton map (c) Feature pair = (r,t) (d) Superimposed rectangles

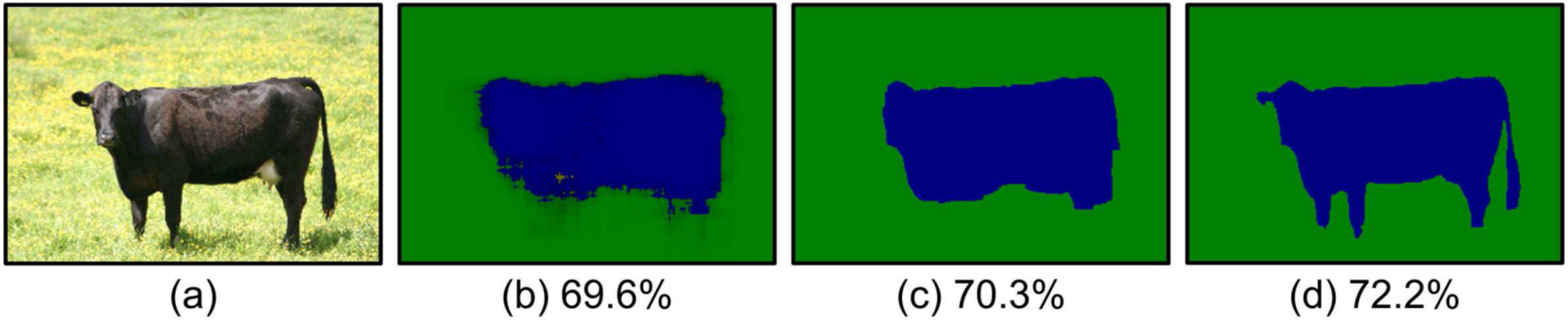
[TextonBoost; Shotton et al, '06]

C. Rother

Semantic Segmentation

Joint Object recognition & segmentation

[TextonBoost; Shotton et al, '06]



Class+
location

+ edges

+ color

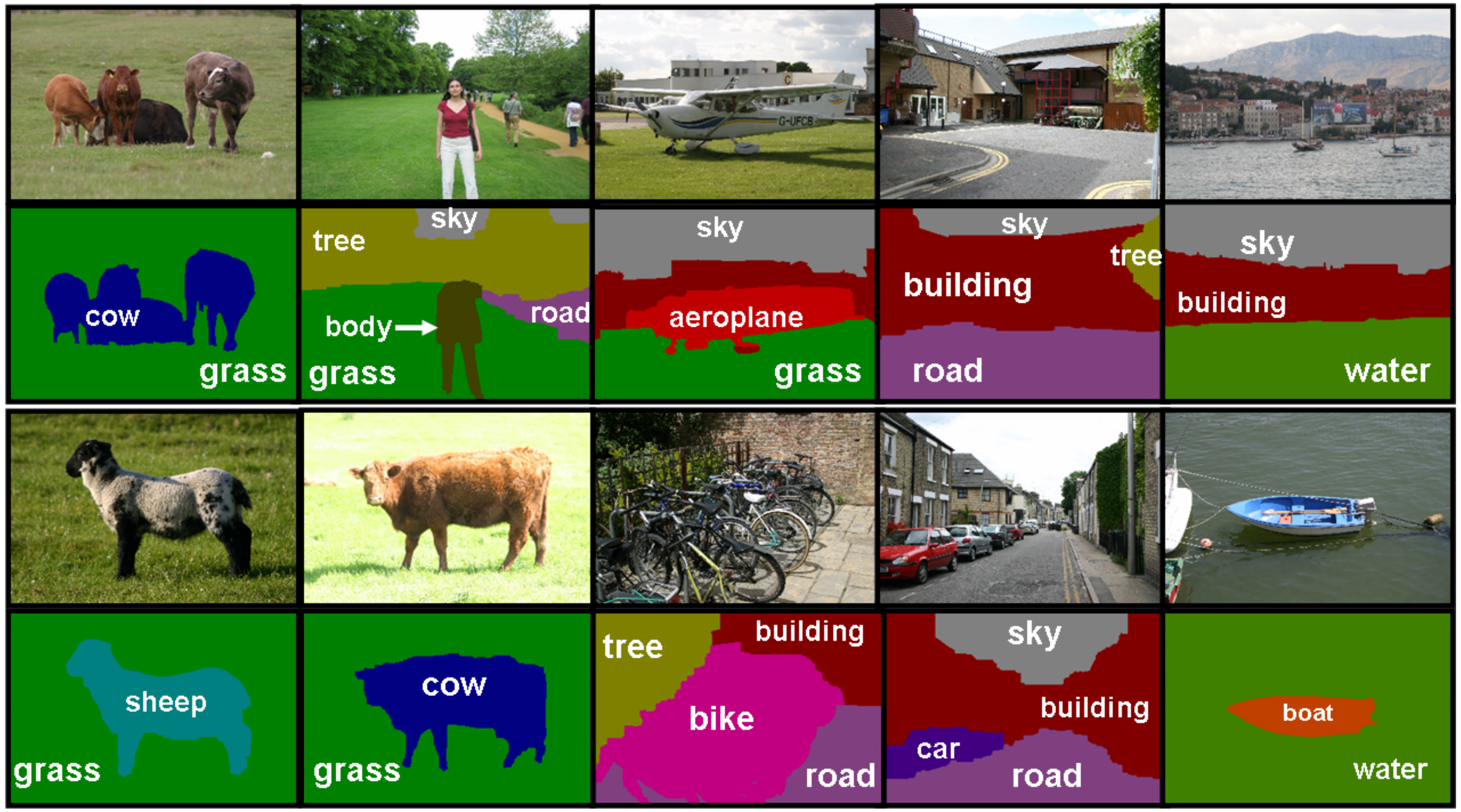
C. Rother

Semantic Segmentation

Joint Object recognition & segmentation

Good results …

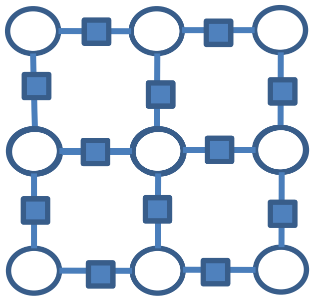
[TextonBoost;
Shotton et al, ‘06]



Object classes	Building	Grass	Tree	Cow	Sheep	Sky	Aeroplane	Water	Face	Car
Bike	Flower	Sign	Bird	Book	Chair	Road	Cat	Dog	Body	Boat

C. Rother

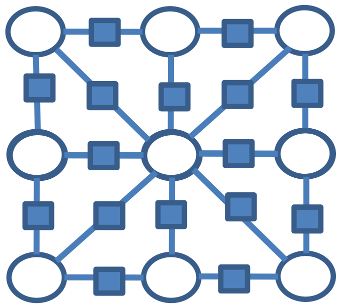
Random Fields in Vision



4-connected; pairwise MRF

$$E(x) = \sum_{i,j \in N_4} \theta_{ij}(x_i,x_j)$$

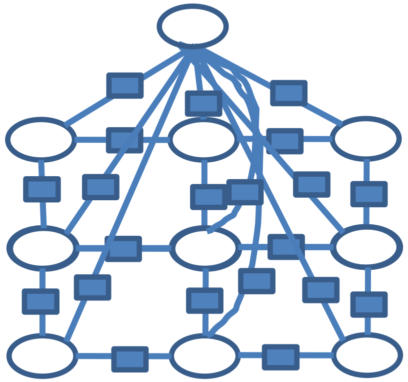
Order 2



higher(8)-connected; pairwise MRF

$$E(x) = \sum_{i,j \in N_8} \theta_{ij}(x_i,x_j)$$

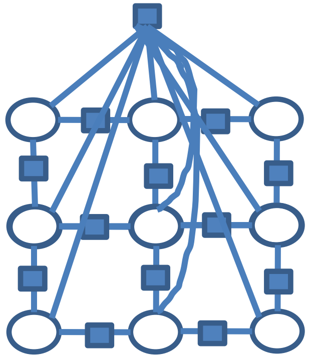
Order 2



MRF with global variables

$$E(x) = \sum_{i,j \in N_8} \theta_{ij}(x_i,x_j)$$

Order 2



Higher-order MRF

$$E(x) = \sum_{i,j \in N_4} \theta_{ij}(x_i,x_j) + \theta(x_1,\ldots,x_n)$$

Order n

C. Rother

Why Higher-order Functions?

- In general $\theta(x_1,x_2,x_3) \neq \theta(x_1,x_2) + \theta(x_1,x_3) + \theta(x_2,x_3)$

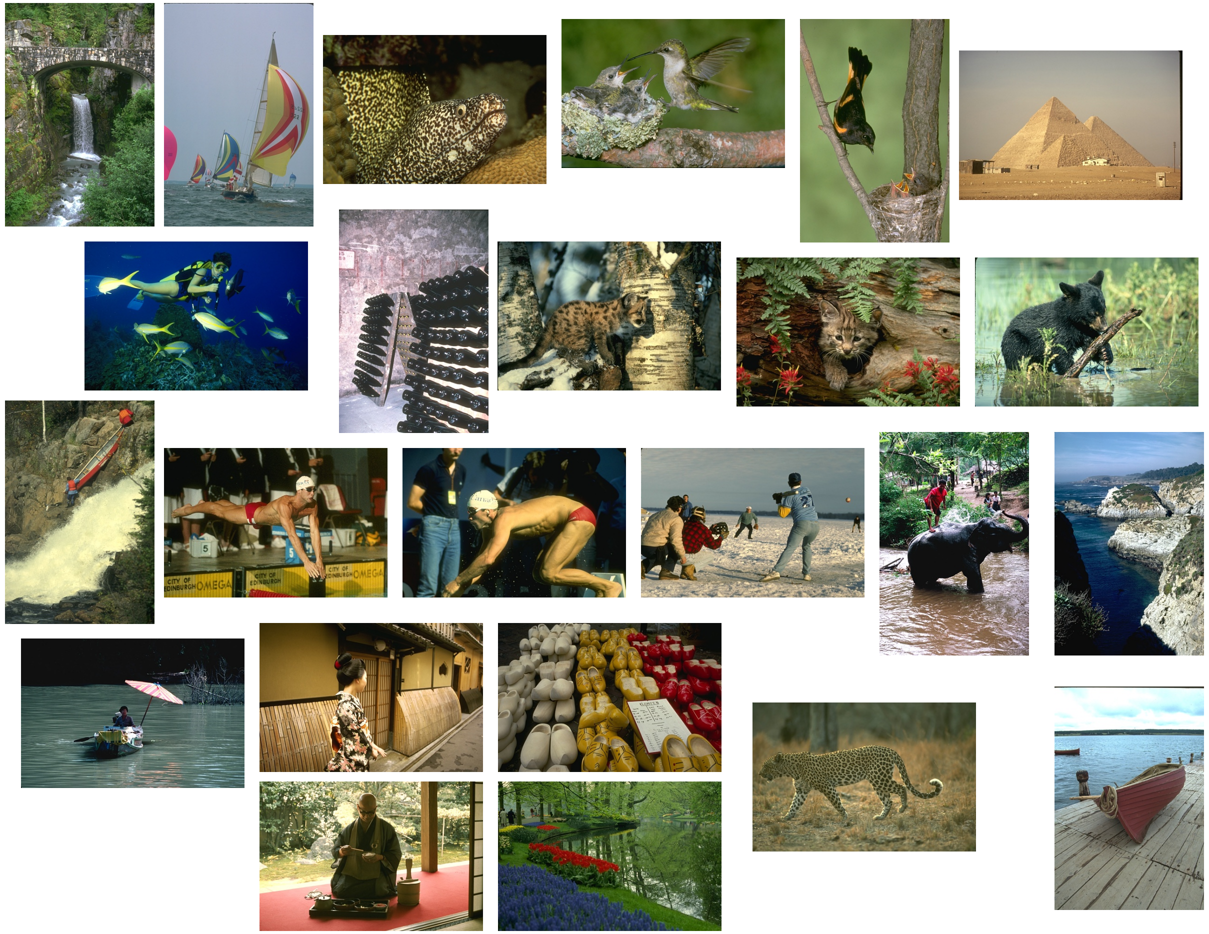
- Reasons for higher-order RFs:

1. Even better image(texture) models:
   - Field-of Expert [FoE, Roth et al. '05]
   - Curvature [Woodford et al. '08]

2. Use global Priors:
   - Connectivity [Vicente et al. '08, Nowozin et al. '09]
   - Better encoding label statistics [Woodford et al. '09]
   - Convert global variables to global factors [Vicente et al. '09]

Modeling the Potentials

- Could the potentials (image priors) be learned from natural images?



Field of Experts (FoE), S. Roth & M. J. Black, CVPR 2005

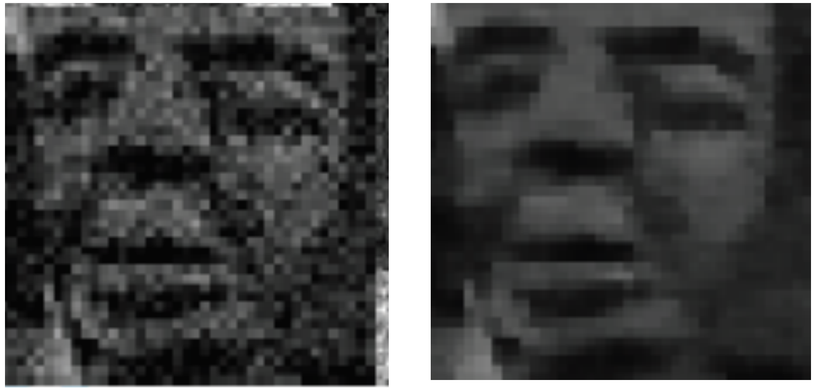
De-noising with Field-of-Experts

[Roth and Black ’05, Ishikawa ‘09]

$$E(X) = \sum_i (z_i\text{-}x_i)^2 / 2\sigma^2 \quad + \sum_c \sum_k \alpha_k \, (1 + 0.5(J_k \, x_c)^2)$$

Unary likelihood   FoE prior

x_c set of nxn patches (here 2x2)
J_k set of filters:



Z   X

From [Ishikawa PAMI ’09, Roth et al ‘05]

non-convex optimization problem

How to handle continuous labels in discrete MRF?

De-noising with Field-of-Experts

[Roth and Black ’05, Ishikawa ‘09]

original image

noisy image,
σ=20

PSNR 22.49dB
SSIM 0.528

denoised using
gradient ascent

PSNR 27.60dB
SSIM 0.810

- Very sharp discontinuities.
- No blurring across boundaries.
- Noise is removed quite well nonetheless.

S. Roth