Graph cuts for maximum a posteriori inference with Markov random field priors

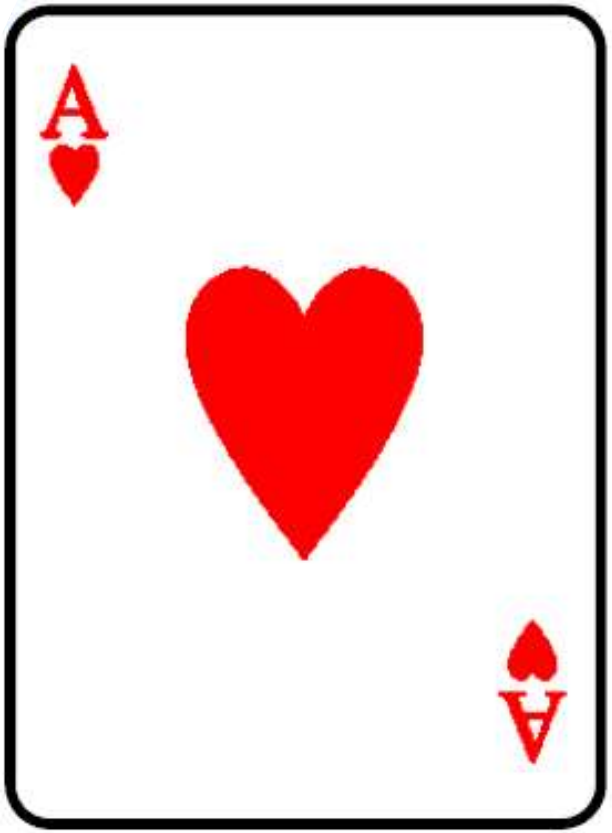
Simon Prince

s.prince@cs.ucl.ac.uk

Plan of Talk

- Denoising problem
- Markov random fields (MRFs)
- Max-flow / min-cut
- Binary MRFs (exact solution)
- Multi-label MRFs – submodular (exact solution)
- Multi-label MRFs - non-submodular (approximate)

Binary Denoising

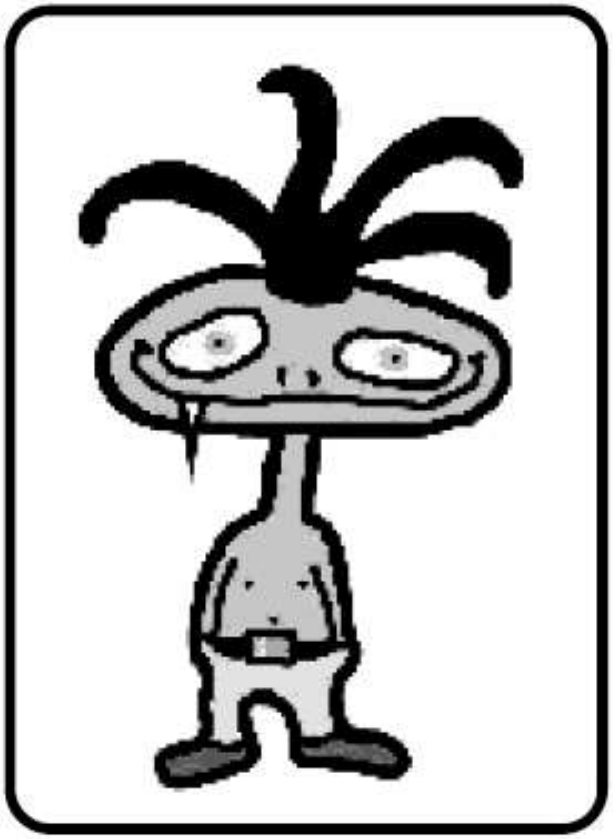


Before

After

Image represented as binary discrete variables. Some proportion of pixels randomly changed polarity.

Multi-label Denoising



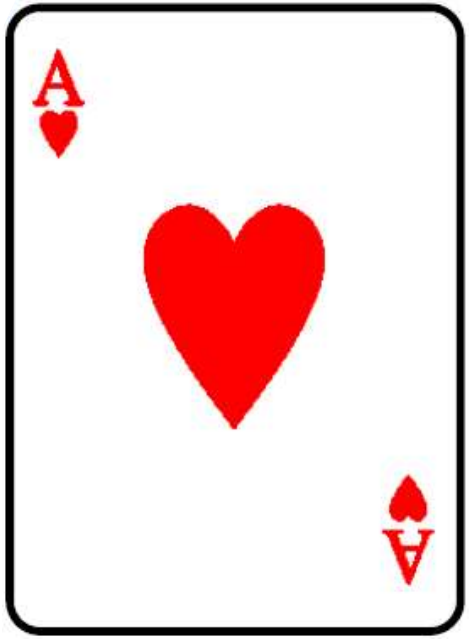
Before

After

Image represented as discrete variables representing intensity. Some proportion of pixels randomly changed according to a uniform distribution.

Denoising Task

Observed Data



Uncorrupted Image

$$\mathbf{x} = \{x_1, x_2 \ldots x_N\}$$

$$\mathbf{y} = \{y_1, y_2 \ldots y_N\}$$

Denoising

Bayes' rule:

$$Pr(y_{1\ldots N}|x_{1\ldots N}) = \frac{\prod_{n=1}^{N} Pr(x_n|y_n)Pr(y_{1\ldots N})}{Pr(x_{1\ldots N})}$$

Likelihoods:

$$\begin{aligned} Pr(x_n|y_n = 0) &= \text{Bern}_{x_n}[\rho] \\ Pr(x_n|y_n = 1) &= \text{Bern}_{x_n}[1-\rho] \end{aligned}$$

Prior: Markov random field (smoothness)

MAP Inference: Graph cuts

Many other vision tasks have this structure (stereo, segmentation etc.)

Plan of Talk

- Denoising problem
- Markov random fields (MRFs)
- Max-flow / min-cut
- Binary MRFs –submodular (exact solution)
- Multi-label MRFs – submodular (exact solution)
- Multi-label MRFs - non-submodular (approximate)

Undirected Models

- MRF is an example of an undirected model
- Product of positive potential functions $\phi_c[y_{1\ldots N}]$

$$Pr(y_{1\ldots N}) = \frac{1}{Z}\prod_{c=1}^{C}\phi_c[y_{1\ldots N}]$$

- Z is a normalizing constant

$$Z = \sum_{y_{1\ldots N}}\prod_{c=1}^{C}\phi_c[y_{1\ldots N}]$$

- Z referred to as “partition function”

Alternate Formulation

- Product of positive potential functions $\phi_c[y_{1\ldots N}]$

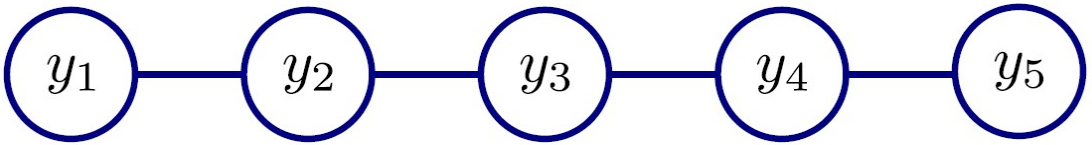
$$Pr(y_{1\ldots N}) = \frac{1}{Z}\prod_{c=1}^{C}\phi_c[y_{1\ldots N}]$$

- or product of exponential costs $\Psi_c[y_{1\ldots N}]$

$$Pr(y_{1\ldots N}) = \frac{1}{Z}\exp\left[-\sum_{c=1}^{C}\Psi_c[y_{1\ldots N}]\right]$$

where $\Phi_c[y_{1\ldots N}] = -\log[\Psi_c[y_{1\ldots N}]]$

MRF Example



- Consider a product of functions over neighbours

$$Pr(y_{1\ldots5}) = \frac{1}{Z}\phi_{12}(y_1, y_2)\phi_{23}(y_2, y_3)\phi_{34}(y_3, y_4)\phi_{45}(y_4, y_5)$$

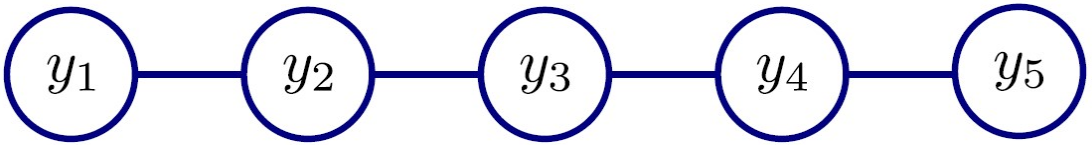
- In this model a variable is conditionally independent of all the others given its neighbours

- For example

$$Pr(y_3|y_1, y_2, y_4, y_5) = Pr(y_3|y_2, y_4)$$

- So connections in graphical model (top) tell us about independence relations

Proof of Markov Property

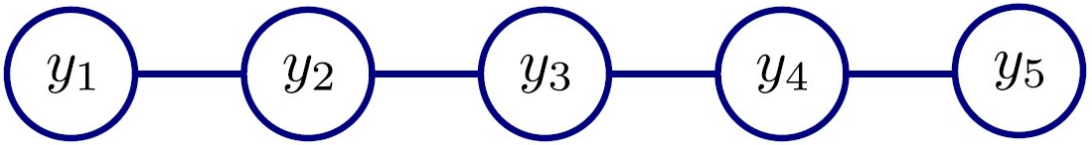


Using conditional probability relation

$$
\begin{aligned}
Pr(y_3|y_1,y_2,y_4,y_5) &= \frac{Pr(y_1,y_2,y_3,y_4,y_5)}{Pr(y_1,y_2,y_4,y_5)} \\
&= \frac{Pr(y_1,y_2,y_3,y_4,y_5)}{\sum_{y_3} Pr(y_1,y_2,y_3,y_4,y_5)} \\
&= \frac{\frac{1}{Z}\phi_{12}(y_1,y_2)\phi_{23}(y_2,y_3)\phi_{34}(y_3,y_4)\phi_{45}(y_4,y_5)}{\sum_{y_3} \frac{1}{Z}\phi_{12}(y_1,y_2)\phi_{23}(y_2,y_3)\phi_{34}(y_3,y_4)\phi_{45}(y_4,y_5)} \\
&= \frac{\phi_{23}(y_2,y_3)\phi_{34}(y_3,y_4)}{\sum_{y_3} \phi_{23}(y_2,y_3)\phi_{34}(y_3,y_4)}
\end{aligned}
$$

(only depends on neighbours)

MRF Example



$$Pr(y_{1\ldots5}) = \frac{1}{Z}\phi_{12}(y_1, y_2)\phi_{23}(y_2, y_3)\phi_{34}(y_3, y_4)\phi_{45}(y_4, y_5)$$

Consider the case where variables are binary, so functions return 4 different values depending on the combination of neighbours. Let's choose

$$\phi_{nm}(0,0) = 1.0 \qquad \phi_{nm}(0,1) = 0.1$$
$$\phi_{nm}(1,0) = 0.1 \qquad \phi_{nm}(1,1) = 1.0$$

$y_{1\ldots5}$	$Pr(y_{1\ldots5})$	$y_{1\ldots5}$	$Pr(y_{1\ldots5})$	$y_{1\ldots5}$	$Pr(y_{1\ldots5})$	$y_{1\ldots5}$	$Pr(y_{1\ldots5})$
00000	0.09877	01000	0.02469	10000	0.04938	11000	0.04938
00001	0.04938	01001	0.01235	10001	0.02469	11001	0.02469
00010	0.02469	01010	0.00617	10010	0.01235	11010	0.01235
00011	0.04938	01011	0.01235	10011	0.02469	11011	0.02469
00100	0.02469	01100	0.02469	10100	0.01235	11100	0.04938
00101	0.01235	01101	0.01235	10101	0.00617	11101	0.02469
00110	0.02469	01110	0.02469	10110	0.01235	11110	0.04938
00111	0.04938	01111	0.04938	10111	0.02469	11111	0.09877

MRF Definition

A Markov Random Field is determined by

- a set of sites $\mathcal{S} = \{1 \ldots N\}$. These will correspond to the N pixel locations,
- a set of random variables $y = \{y_1 \ldots y_N\}$ associated with each of the sites,
- a set of neighbors $\mathcal{N}_{1\ldots N}$ at each of the N sites. The set $\mathcal{N}_n$ contains the indices of the subset of random variables have an immediate probabilistic connection to variable y_n.

To be a Markov random field, the model must obey the Markov property,

$$Pr(y_n|y_{\mathcal{S}\setminus n}) = Pr(y_n|y_{\mathcal{N}_n}) \qquad \forall \quad n \in \mathcal{S},$$

Hammersley Clifford Theorem

Any distribution that obeys the Markov property

$$Pr(y_n|y_{\mathcal{S}\setminus n}) = Pr(y_n|y_{\mathcal{N}_n}) \qquad \forall \quad n \in \mathcal{S},$$

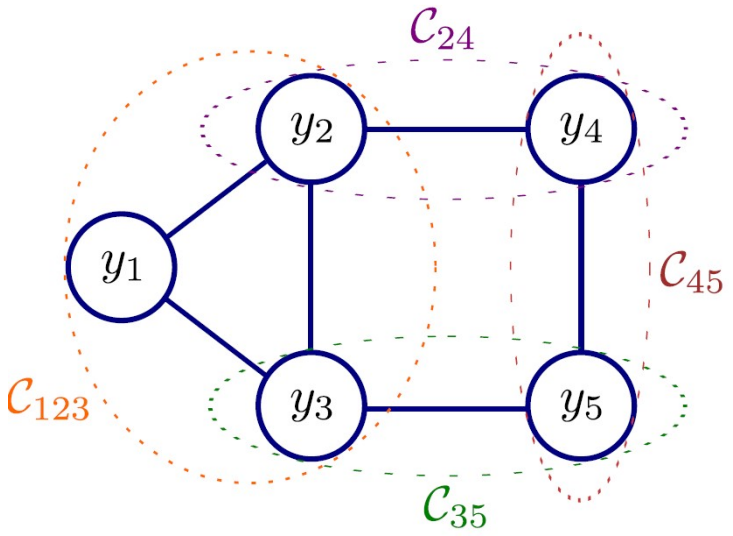
can be written in the form

$$Pr(\mathbf{y}) = \frac{1}{Z}\exp\left[-\sum_{c\in\mathcal{C}}\Psi_c(\mathbf{y})\right]$$

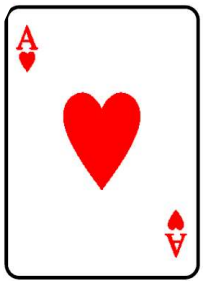
Where the *c* terms are maximal cliques

Cliques = subsets of variables that all connect to each other.

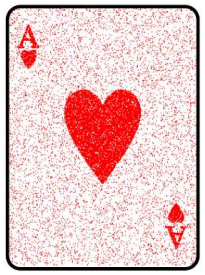
Maximal = cannot add any more variables and still be a clique



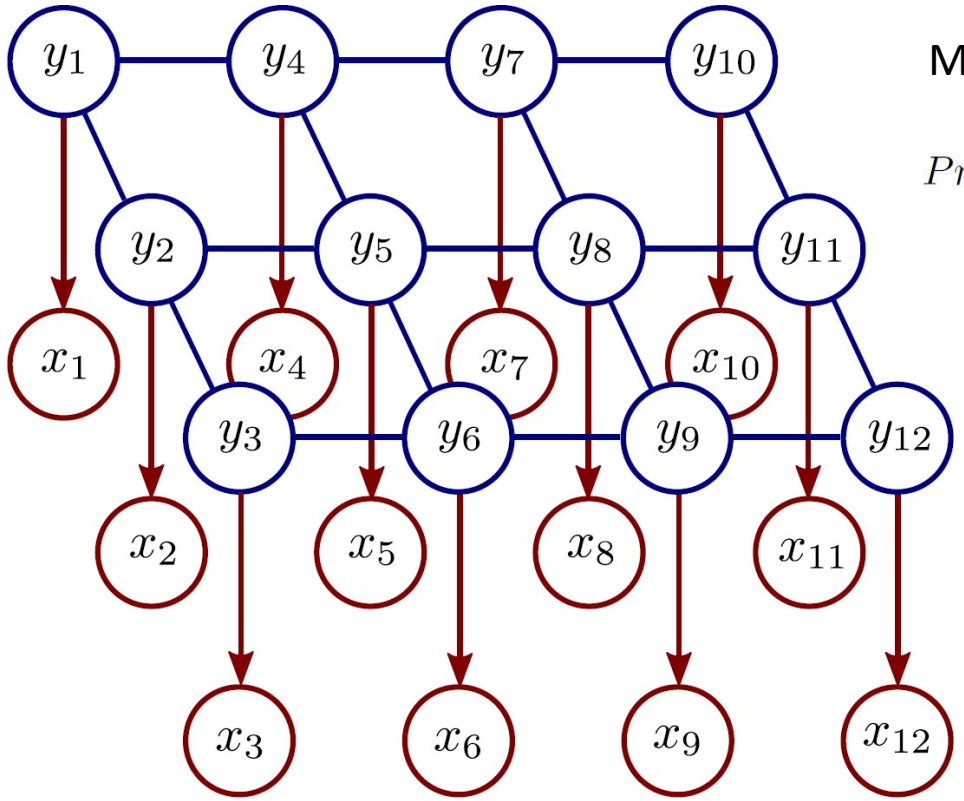
Denoising with MRFs



Original image, **y**



Observed image, **x**



MRF Prior (pairwise cliques)

$$Pr(\mathbf{y}) = \frac{1}{Z}\exp\left[-\sum_{c\in\mathcal{C}}\Psi_c(\mathbf{y})\right]$$

Likelihoods

$$\begin{aligned} Pr(x_n|y_n=0) &= \text{Bern}_{x_n}[\rho] \\ Pr(x_n|y_n=1) &= \text{Bern}_{x_n}[1-\rho] \end{aligned}$$

Inference via Bayes' rule:

$$Pr(y_{1\ldots N}|x_{1\ldots N}) = \frac{\prod_{n=1}^{N} Pr(x_n|y_n)Pr(y_{1\ldots N})}{Pr(x_{1\ldots N})}$$

MAP Inference

$$
\begin{aligned}
\hat{y}_{1\ldots N} &= \arg\max_{y_{1\ldots N}} Pr(y_{1\ldots N}|\mathbf{x}_{1\ldots N}) \\
&= \arg\max_{y_{1\ldots N}} \prod_{n=1}^{N} Pr(x_n|y_n) Pr(y_{1\ldots N}) \\
&= \arg\max_{y_{1\ldots N}} \sum_{n=1}^{N} \log[Pr(x_n|y_n)] + \log[Pr(y_{1\ldots N})] \\
&= \arg\min_{y_{1\ldots N}} \sum_{n=1}^{N} U_n(y_n) + \sum_{(m,n)\in\mathcal{C}} P_{m,n}(y_m, y_n)
\end{aligned}
$$

Unary terms
(compatability of data with label y)

Pairwise terms
(compatability of neighboring labels)

Graph Cuts Overview

Graph cuts used to optimise this cost function:

$$\arg\min_{y_{1\ldots N}} \sum_{n=1}^{N} U_n(y_n) + \sum_{(m,n)\in\mathcal{C}} P_{m,n}(y_m, y_n)$$

Unary terms
(compatability of data with label y)

Pairwise terms
(compatability of neighboring labels)

Three main cases:

- binary MRFs (i.e. $y_i \in \{0, 1\}$) where the costs for different combinations of adjacent labels are "submodular". Exact MAP inference is tractable here.
- multi-label MRFs (i.e. $y_i \in \{1, 2 \ldots, K\}$) where the costs are "submodular". Once more, exact MAP inference is possible.
- multi-label MRFs where the costs are more general. Exact MAP inference is intractable, but good approximate solutions can be found in some cases.

Graph Cuts Overview

Graph cuts used to optimise this cost function:

$$\arg\min_{y_{1\ldots N}} \sum_{n=1}^{N} U_n(y_n) + \sum_{(m,n)\in\mathcal{C}} P_{m,n}(y_m, y_n)$$

Unary terms
(compatability of data with label y)

Pairwise terms
(compatability of neighboring labels)

<u>Approach:</u>

Convert minimization into the form of a standard CS problem,

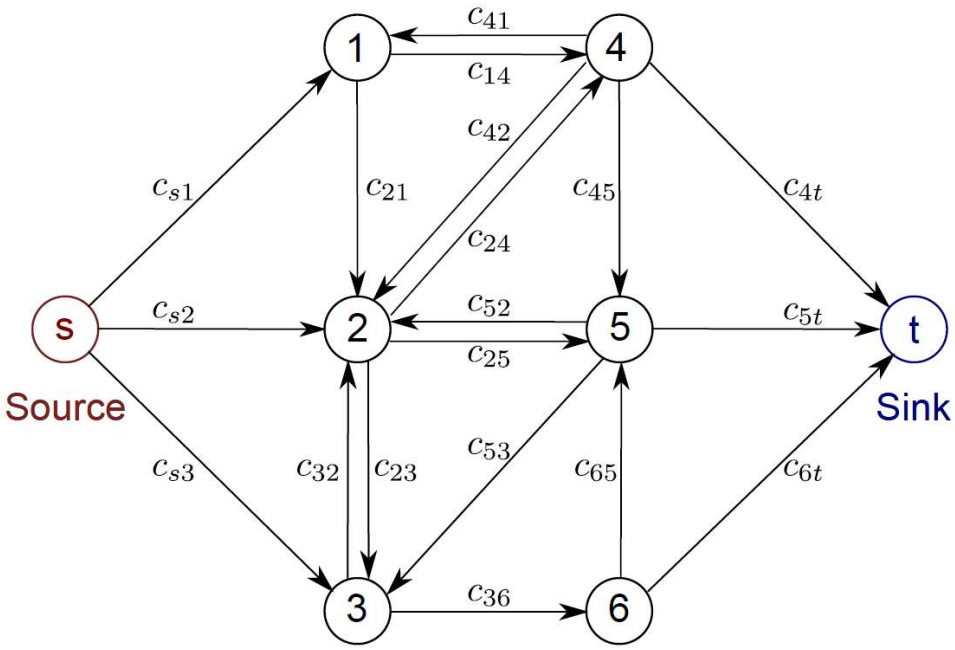
MAXIMUM FLOW or MINIMUM CUT ON A GRAPH

Low order polynomial methods for solving this problem are known

Plan of Talk

- Denoising problem
- Markov random fields (MRFs)
- Max-flow / min-cut
- Binary MRFs - submodular (exact solution)
- Multi-label MRFs – submodular (exact solution)
- Multi-label MRFs - non-submodular (approximate)

Max-Flow Problem

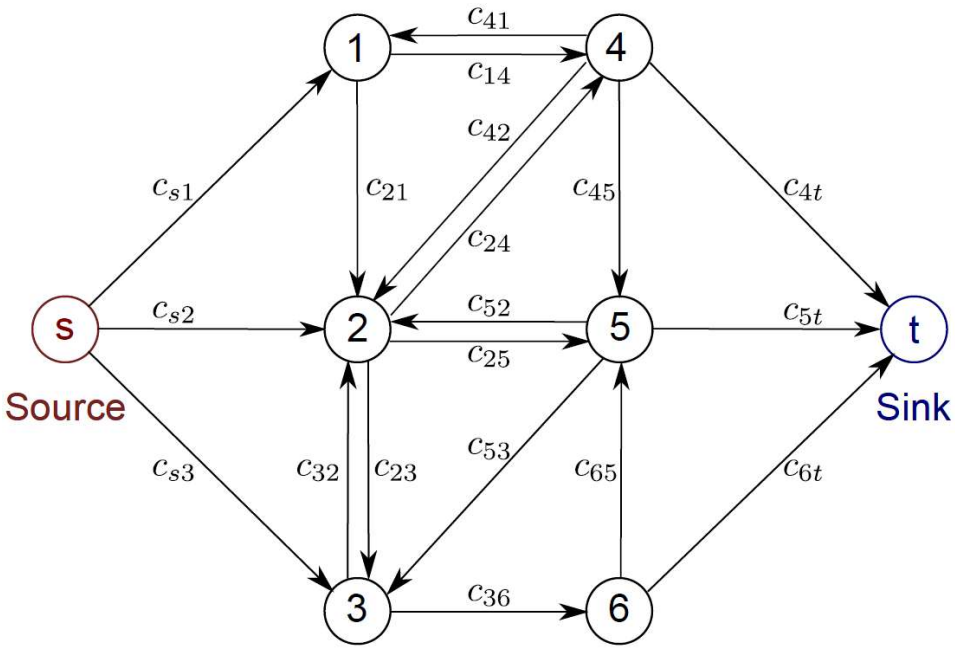


Goal:

To push as much ‘flow’ as possible through the directed graph from the source to the sink.

Cannot exceed the (non-negative) capacities c_{ij} associated with each edge.

Saturated Edges

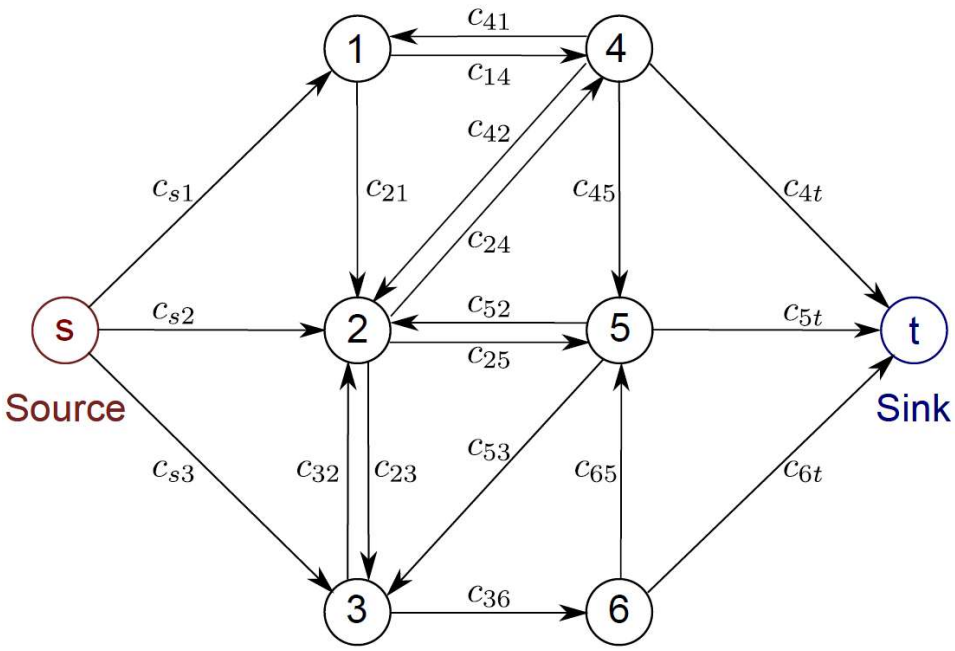


When we are pushing the maximum amount of flow:

- There must be at least one saturated edge on any path from source to sink
  (otherwise we could push more flow)

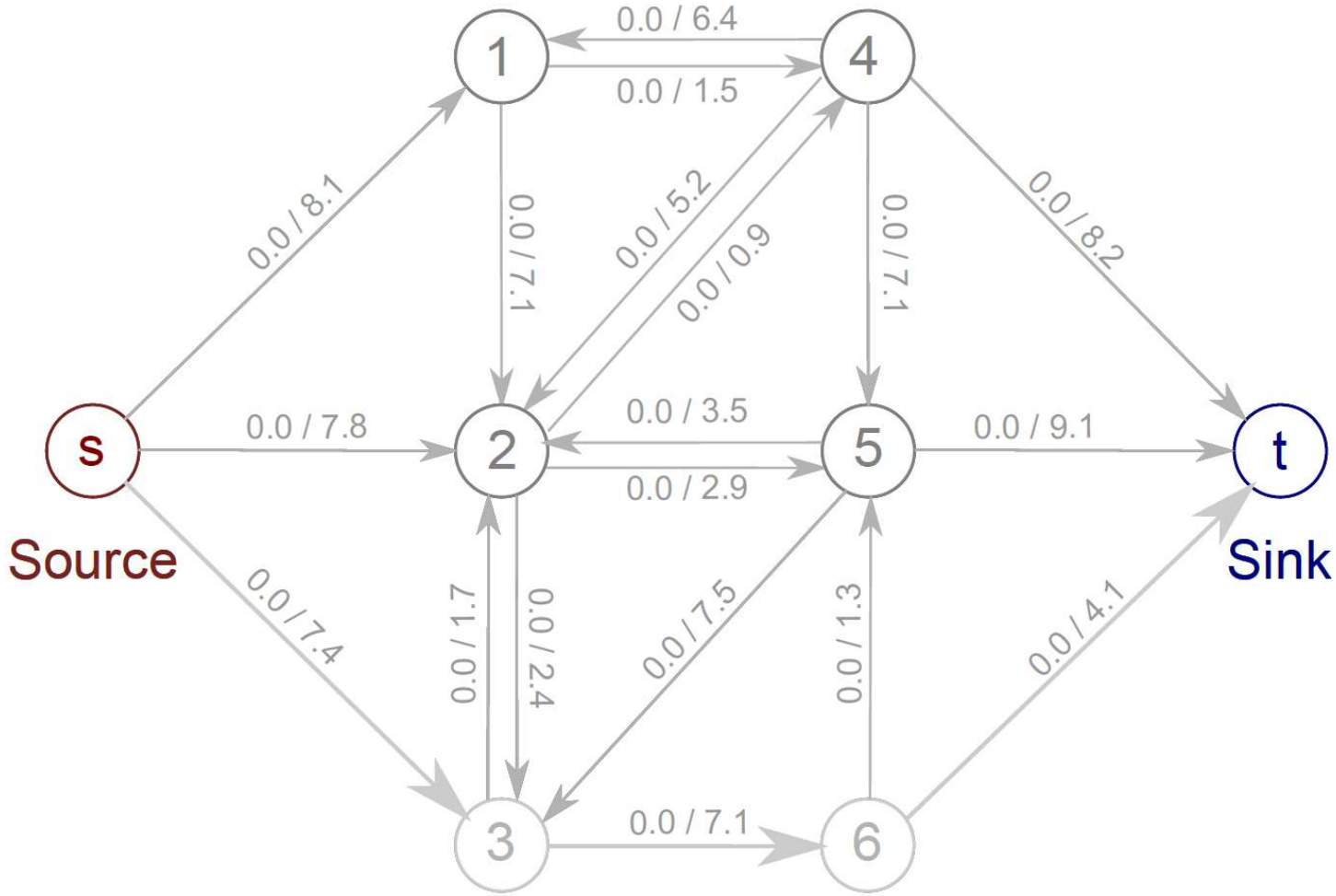
- The set of saturated edges hence separate the source and sink

Min Cut



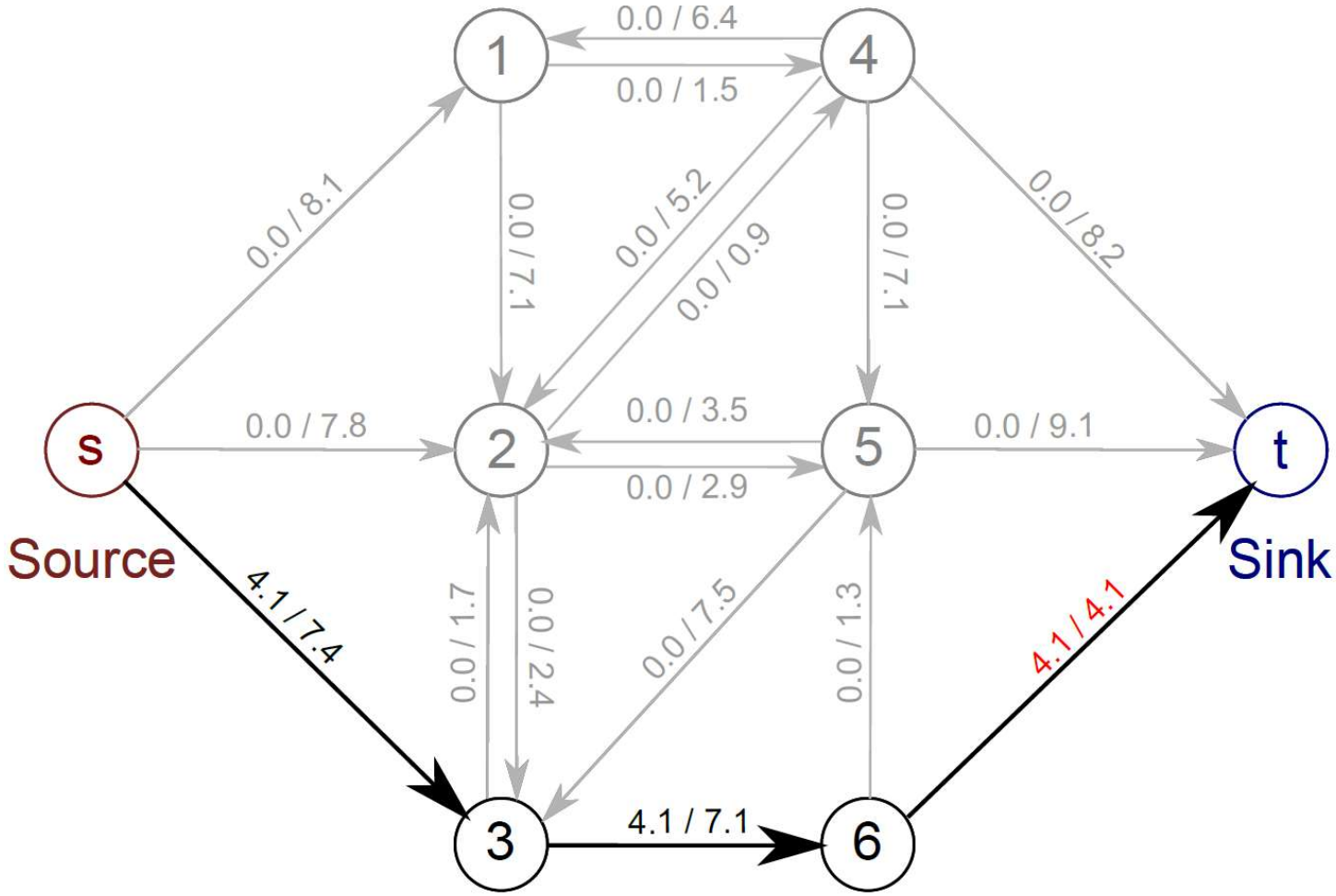
- Define a cut on the graph as a set of edges that separate the source and sink.
- Cost of cut = total capacity of these edges
- Edges that saturate in the maximum flow solution form the minimum cost cut
- Can talk interchangeably about max-flow or min-cut

Augmenting Paths



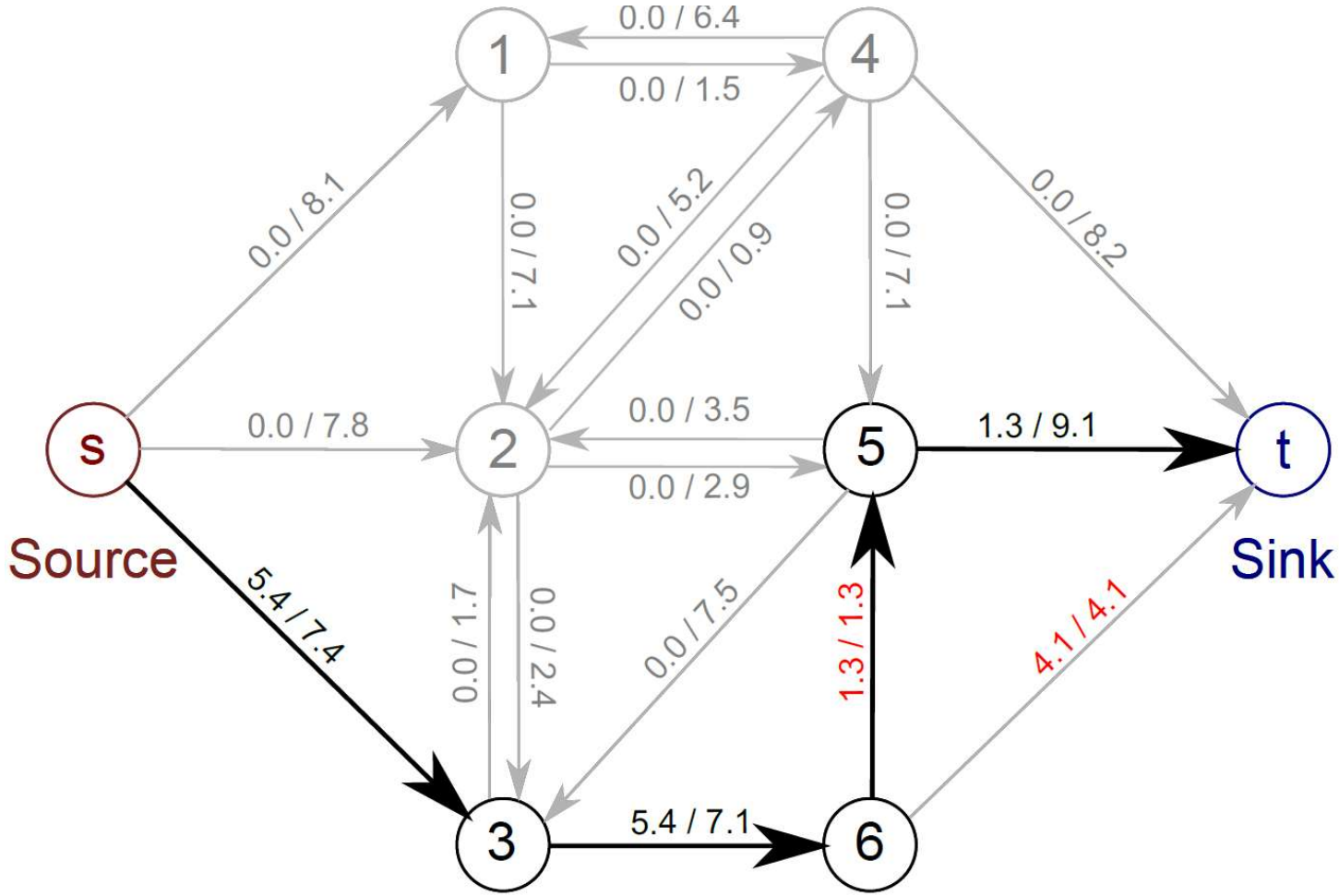
Two numbers represent: current flow / total capacity

Augmenting Paths



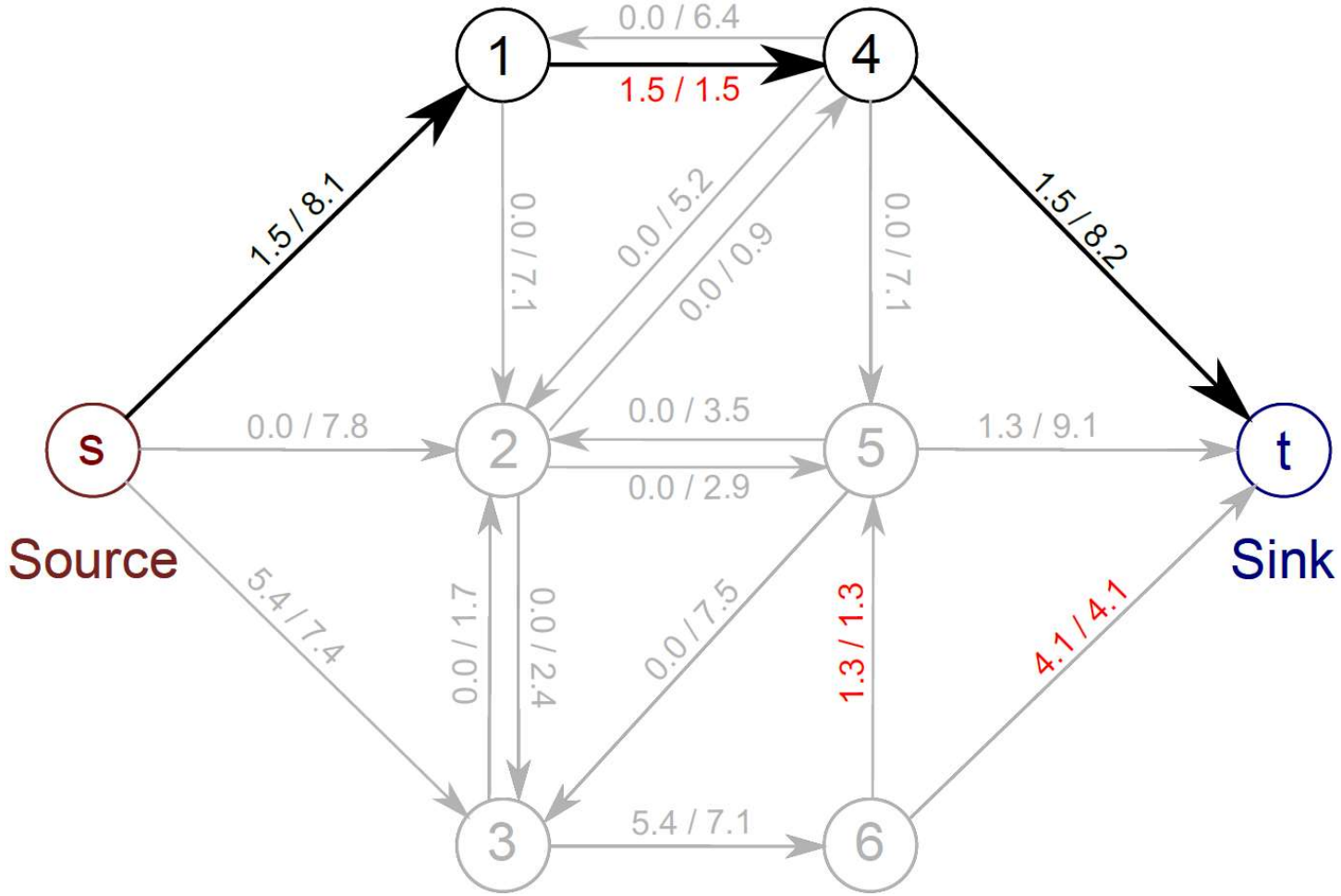
Choose any route from source to sink with spare capacity and push as much flow as you can. One edge (here 6-t) will saturate.

Augmenting Paths



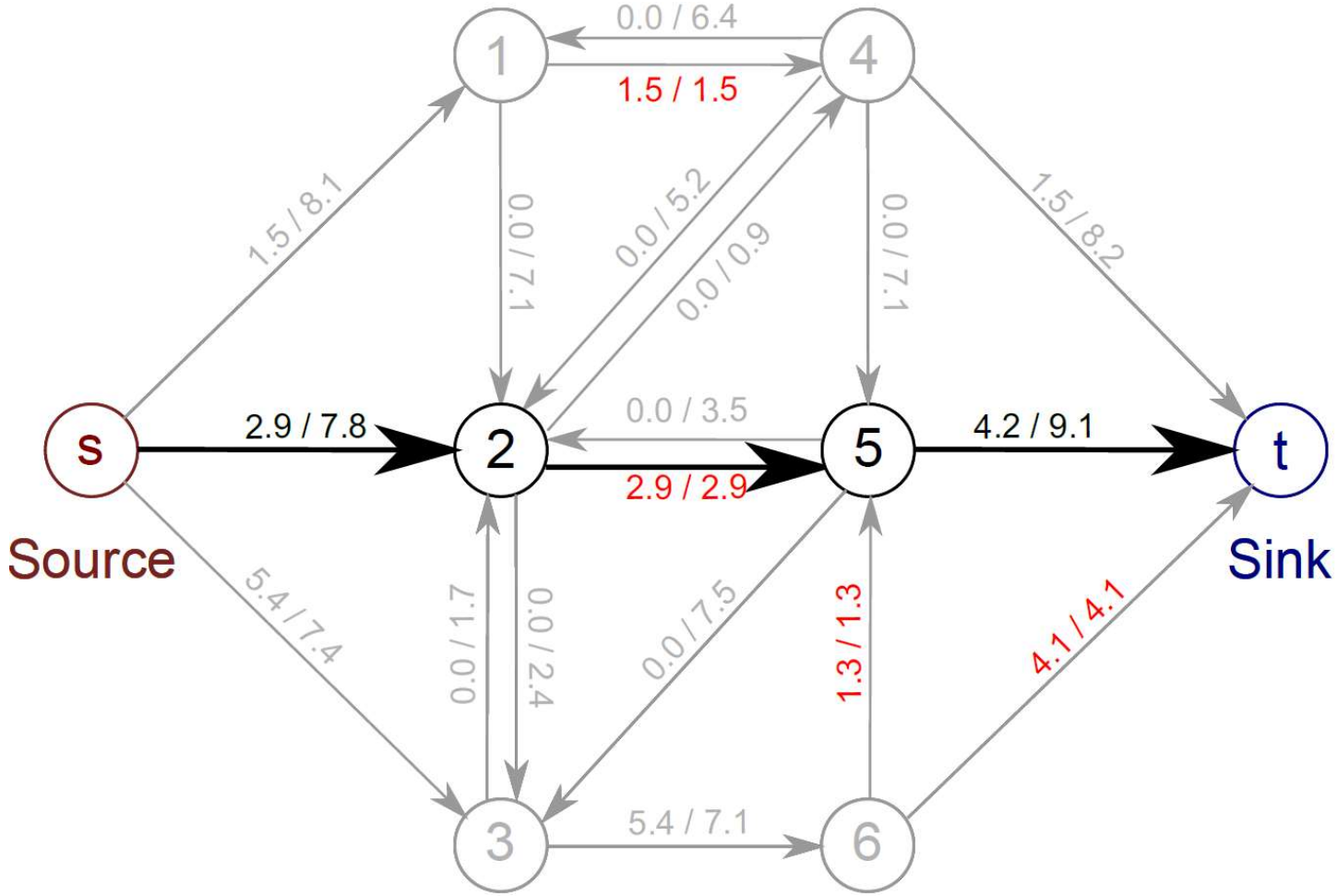
Choose another route, respecting remaining capacity. This time edge 5-6 saturates.

Augmenting Paths



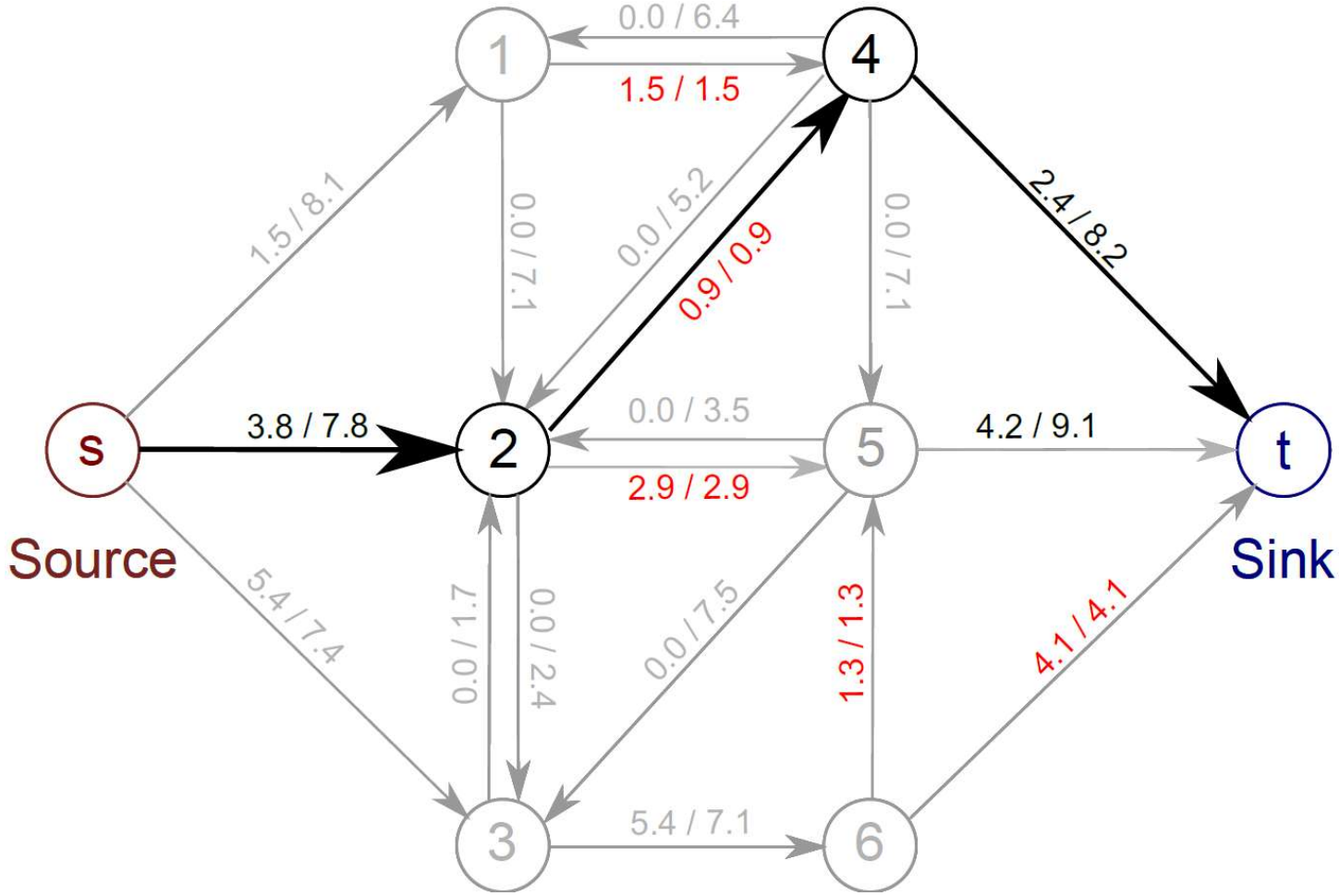
A third route. Edge 1-4 saturates

Augmenting Paths



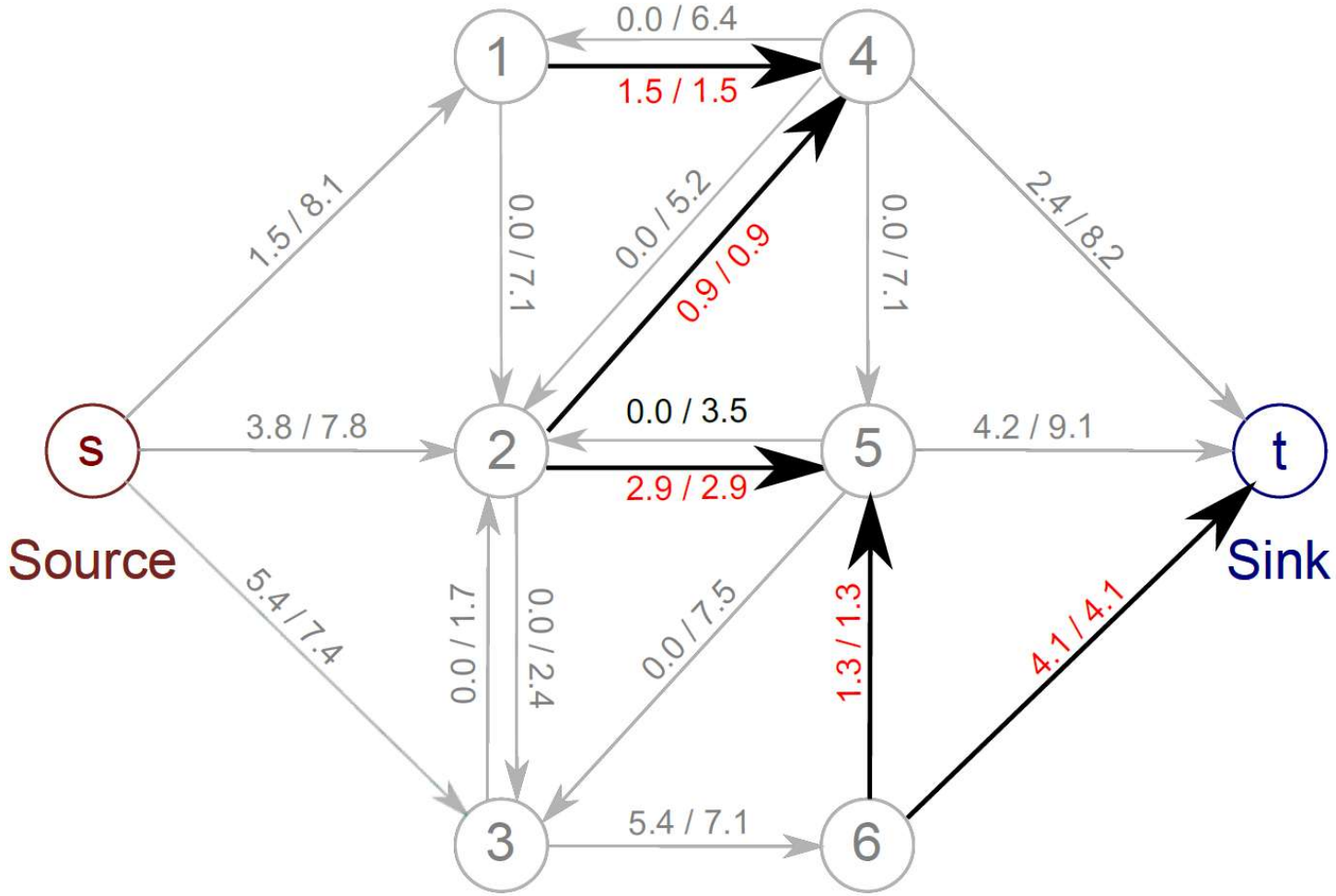
A fourth route. Edge 2-5 saturates

Augmenting Paths



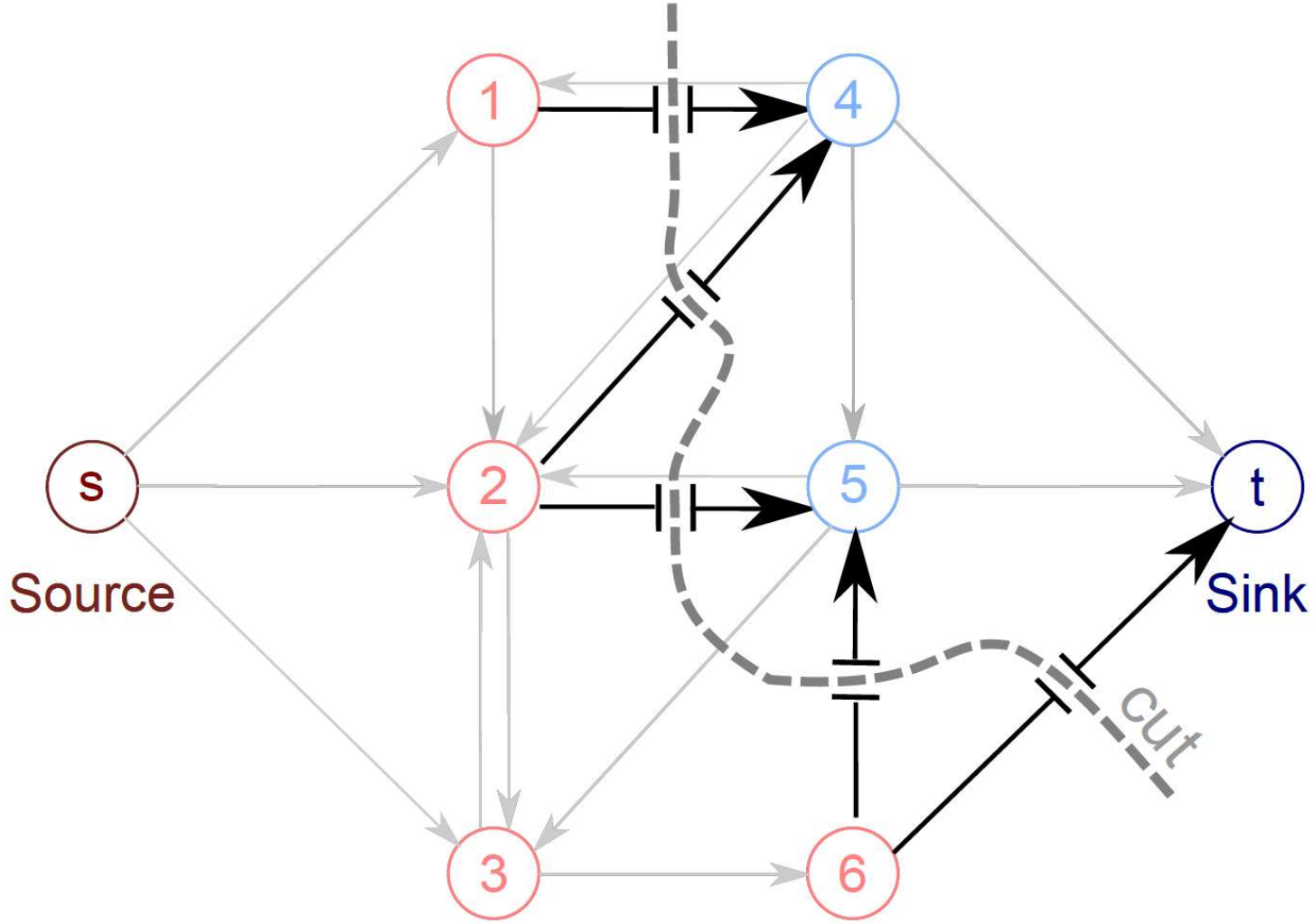
A fifth route. Edge 2-4 saturates

Augmenting Paths



There is now no further route from source to sink – there is a saturated edge along every possible route (highlighted arrows)

Augmenting Paths



The saturated edges separate the source from the sink and form the min-cut solution. Nodes either connect to the source or connect to the sink.

Plan of Talk

- Denoising problem
- Markov random fields (MRFs)
- Max-flow / min-cut
- Binary MRFs – submodular (exact solution)
- Multi-label MRFs – submodular (exact solution)
- Multi-label MRFs - non-submodular (approximate)

Graph Cuts: Binary MRF

Graph cuts used to optimise this cost function:

$$\arg\min_{y_{1\ldots N}} \sum_{n=1}^{N} U_n(y_n) + \sum_{(m,n)\in\mathcal{C}} P_{m,n}(y_m, y_n)$$

Unary terms
(compatability of data with label y)

Pairwise terms
(compatability of neighboring labels)

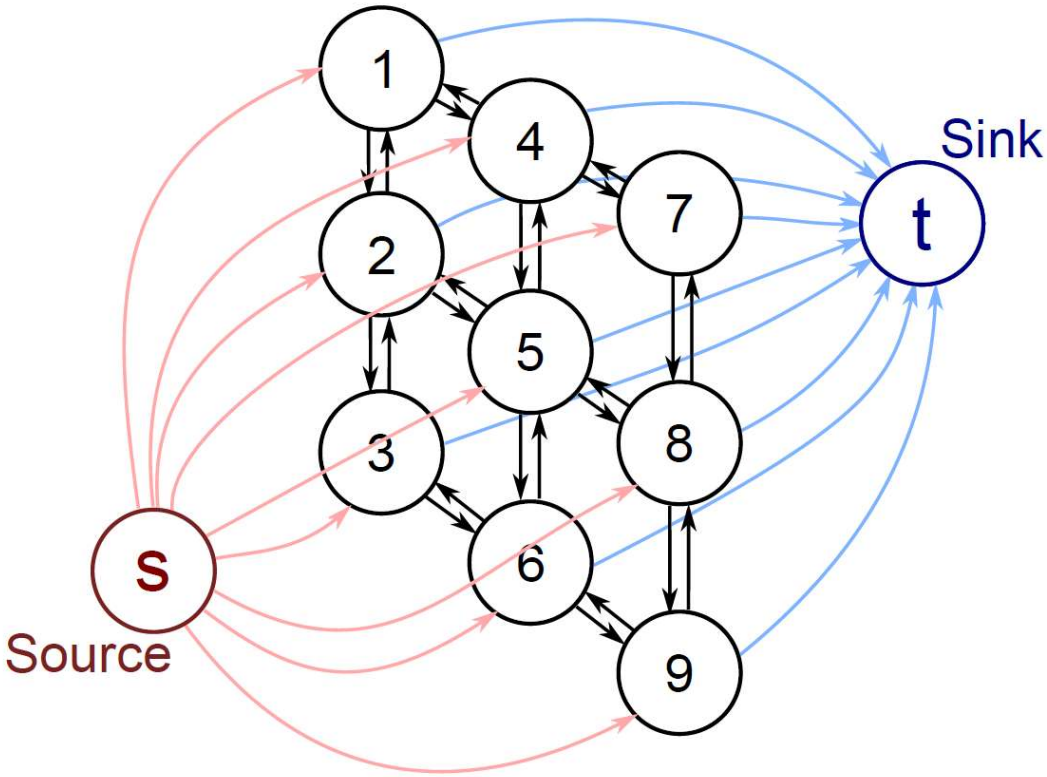
First work with binary case (i.e. True label y is 0 or 1)

Constrain pairwise costs so that they are "zero-diagonal"

$$P_{m,n}(0,0) = 0 \qquad P_{m,n}(1,0) = \theta_{10}$$
$$P_{m,n}(0,1) = \theta_{01} \qquad P_{m,n}(1,1) = 0,$$

Graph Construction

- One node per pixel (here a 3x3 image)
- Edge from source to every pixel node
- Edge from every pixel node to sink
- Reciprocal edges between neighbours

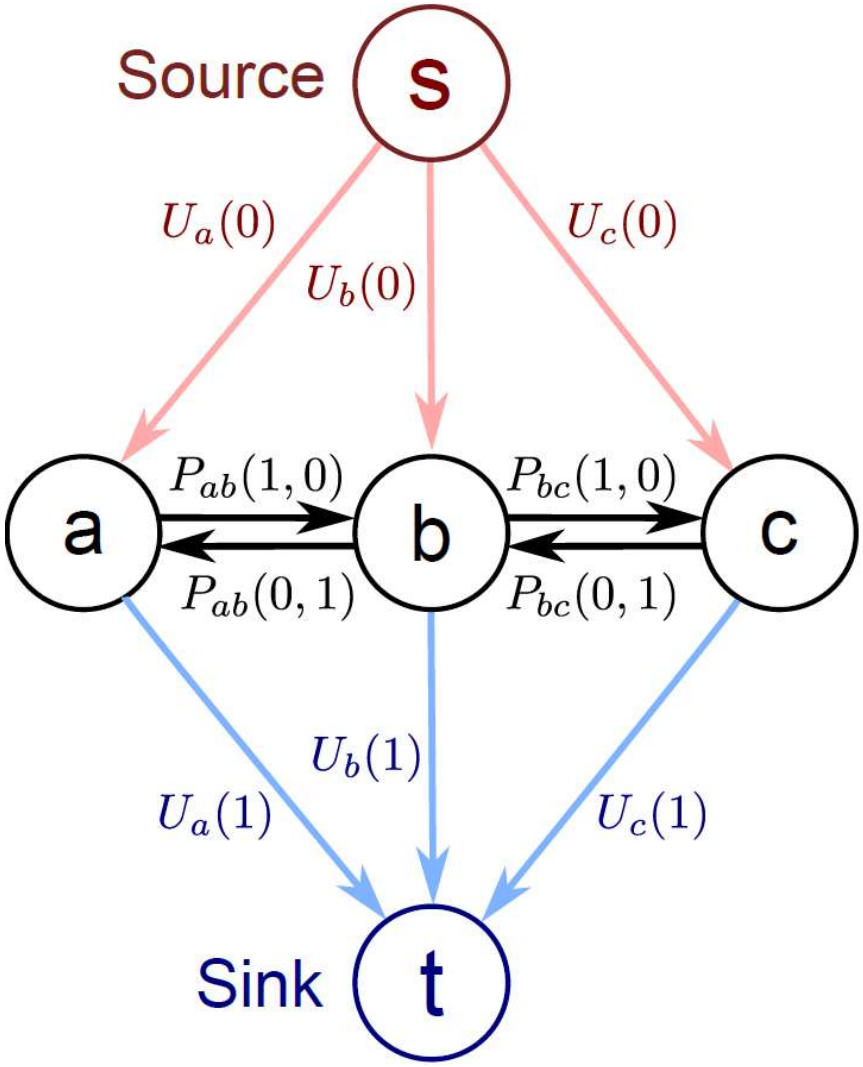


Note that in the minimum cut EITHER the edge connecting to the source will be cut, OR the edge connecting to the sink, but NOT BOTH (unnecessary).

Which determines whether we give that pixel label 1 or label 0.

Now a 1 to 1 mapping between possible labelling and possible minimum cuts

Graph Construction



Now add capacities so that minimum cut, minimizes our cost function

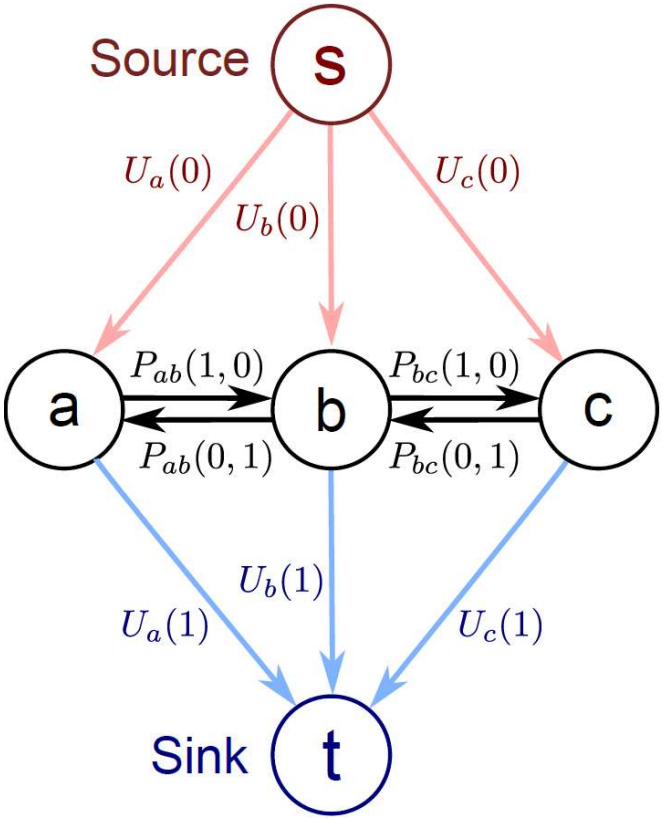
Unary costs U(0), U(1) attached to links to source and sink.

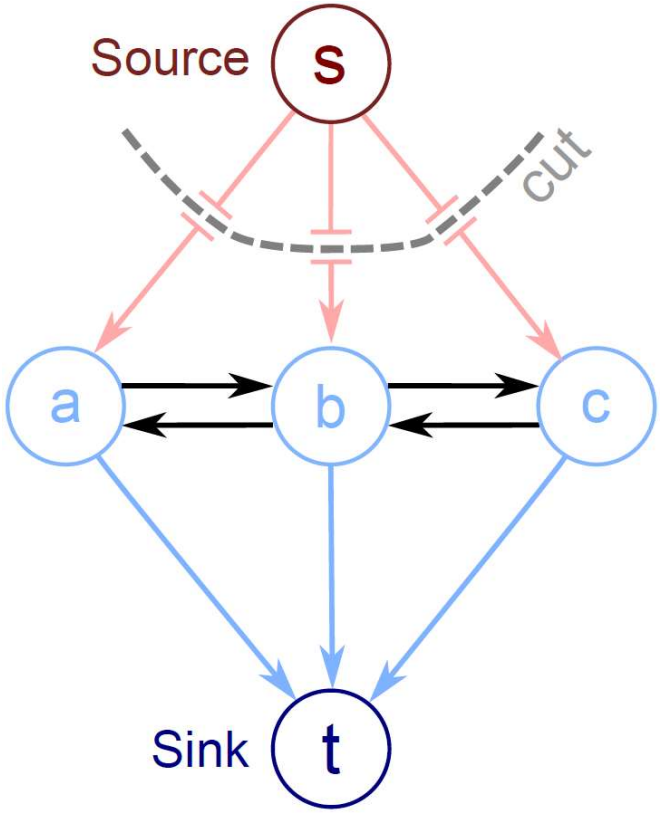
- Either one or the other is paid.

Pairwise costs between pixel nodes as shown.

- Why? Easiest to understand with some worked examples.

Example 1





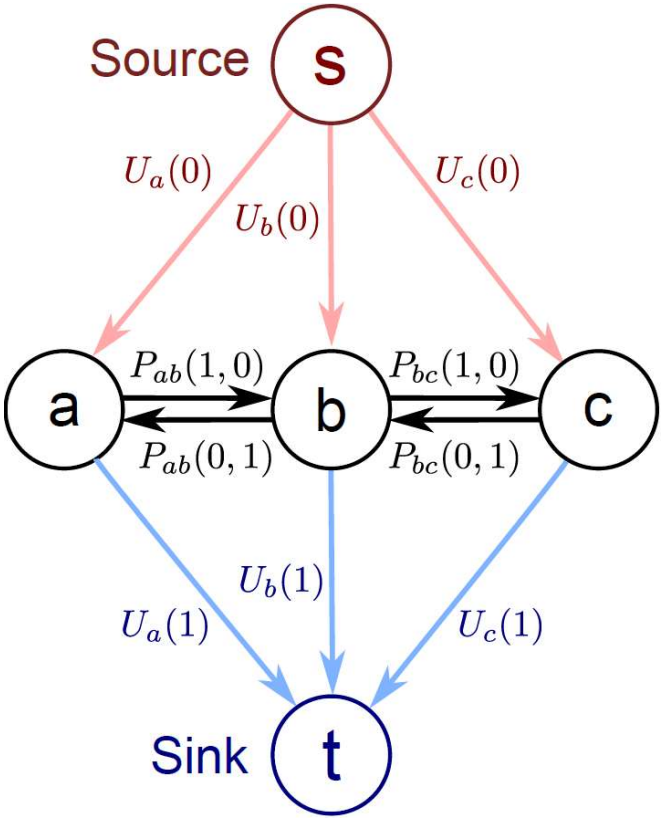
Solution

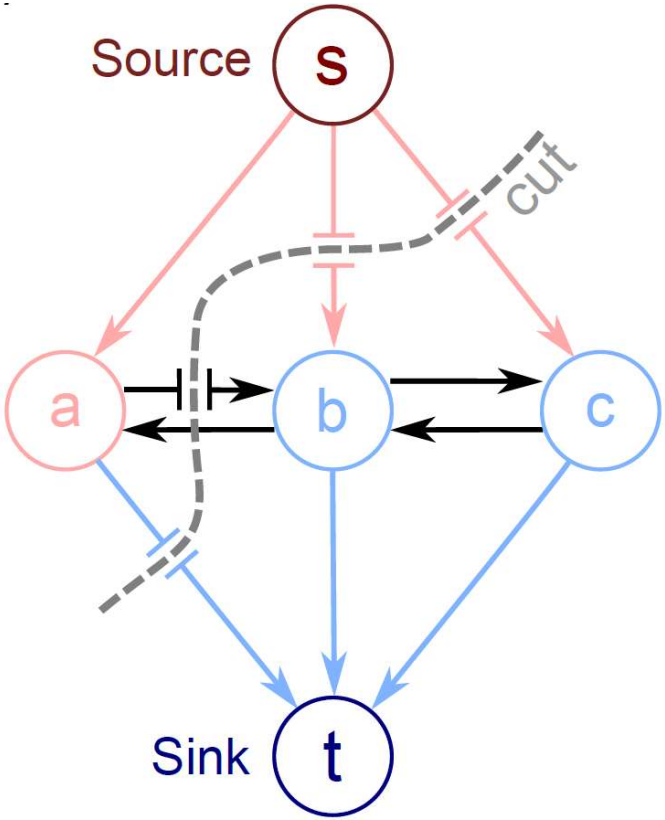
0	0	0

Cost

$$U_a(0) + U_b(0) + U_c(0)$$

Example 2





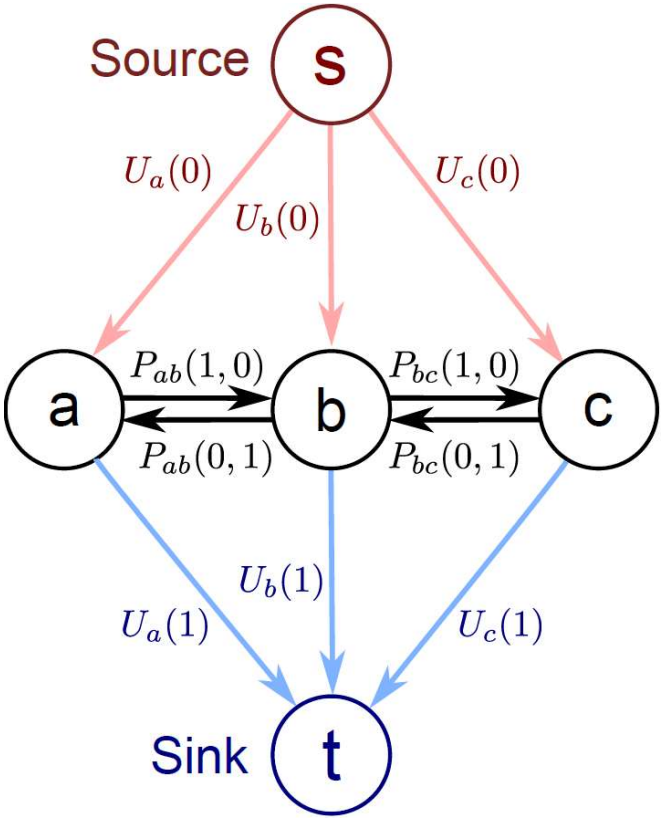
Solution

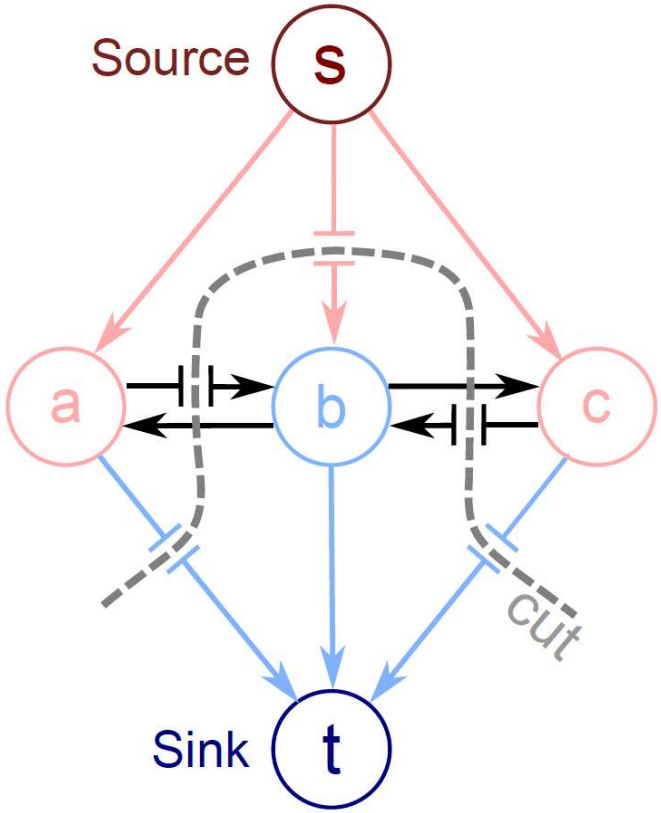
1	0	0

Cost

$$U_a(1) + U_b(0) + U_c(0) + P_{ab}(1,0)$$

Example 3





Solution

1	0	1

Cost

$$U_a(0) + U_b(0) + U_c(0)$$
$$+P_{ab}(1,0) + P_{bc}(0,1)$$

Graph Cuts: Binary MRF

Graph cuts used to optimise this cost function:

$$\arg\min_{y_{1\ldots N}} \sum_{n=1}^{N} U_n(y_n) + \sum_{(m,n)\in\mathcal{C}} P_{m,n}(y_m, y_n)$$

Unary terms
(compatability of data with label y)

Pairwise terms
(compatability of neighboring labels)

Summary of approach

- Associate each possible solution with a minimum cut on a graph
- Set capacities on graph, so cost of cut matches the cost function
- Use augmenting paths to find minimum cut
- This minimizes the cost function and finds the MAP solution

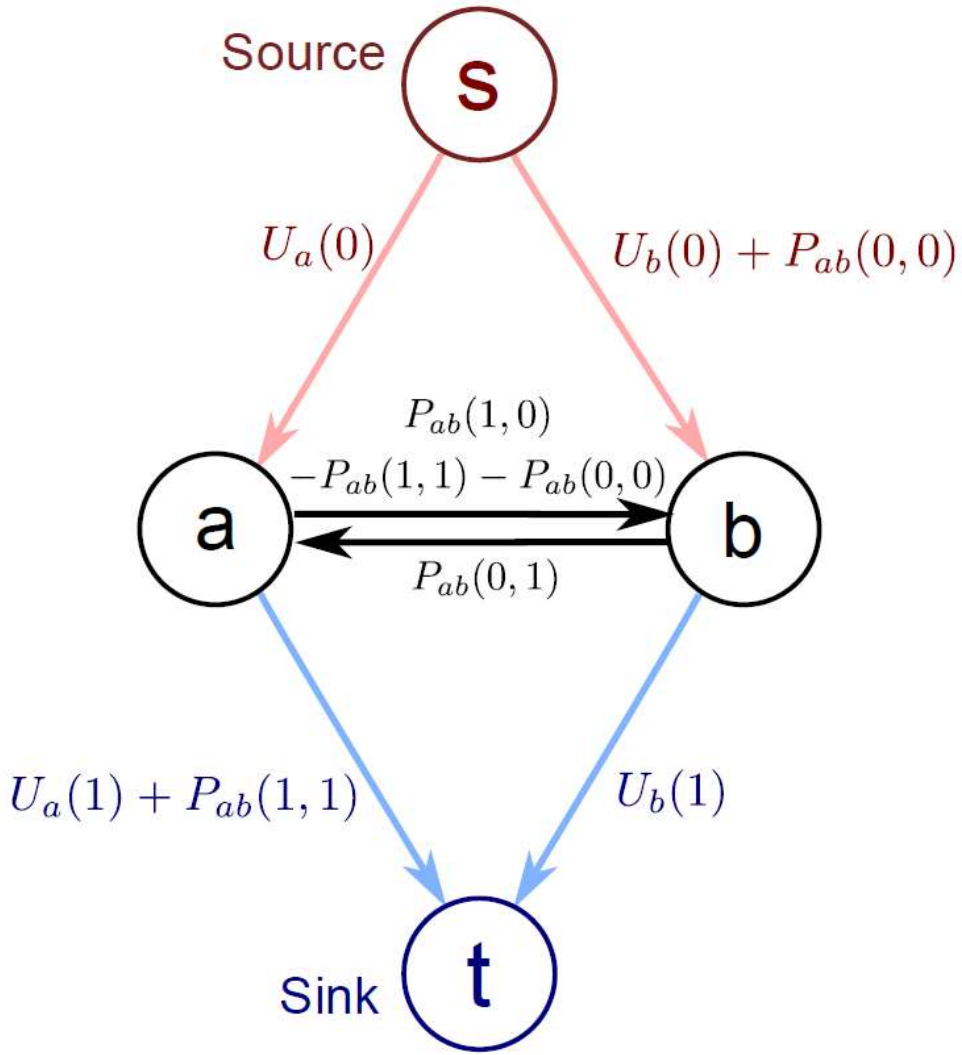
General Pairwise costs

$$P_{m,n}(0,0) = \theta_{00} \quad P_{m,n}(1,0) = \theta_{10}$$
$$P_{m,n}(0,1) = \theta_{01} \quad P_{m,n}(1,1) = \theta_{11}$$

Modify graph to

- Add P(0,0) to edge s-b
  - Implies that solutions 0,0 and 1,0 also pay this cost
- Subtract P(0,0) from edge b-a
  - Solution 1,0 has this cost removed again

Similar approach for P(1,1)

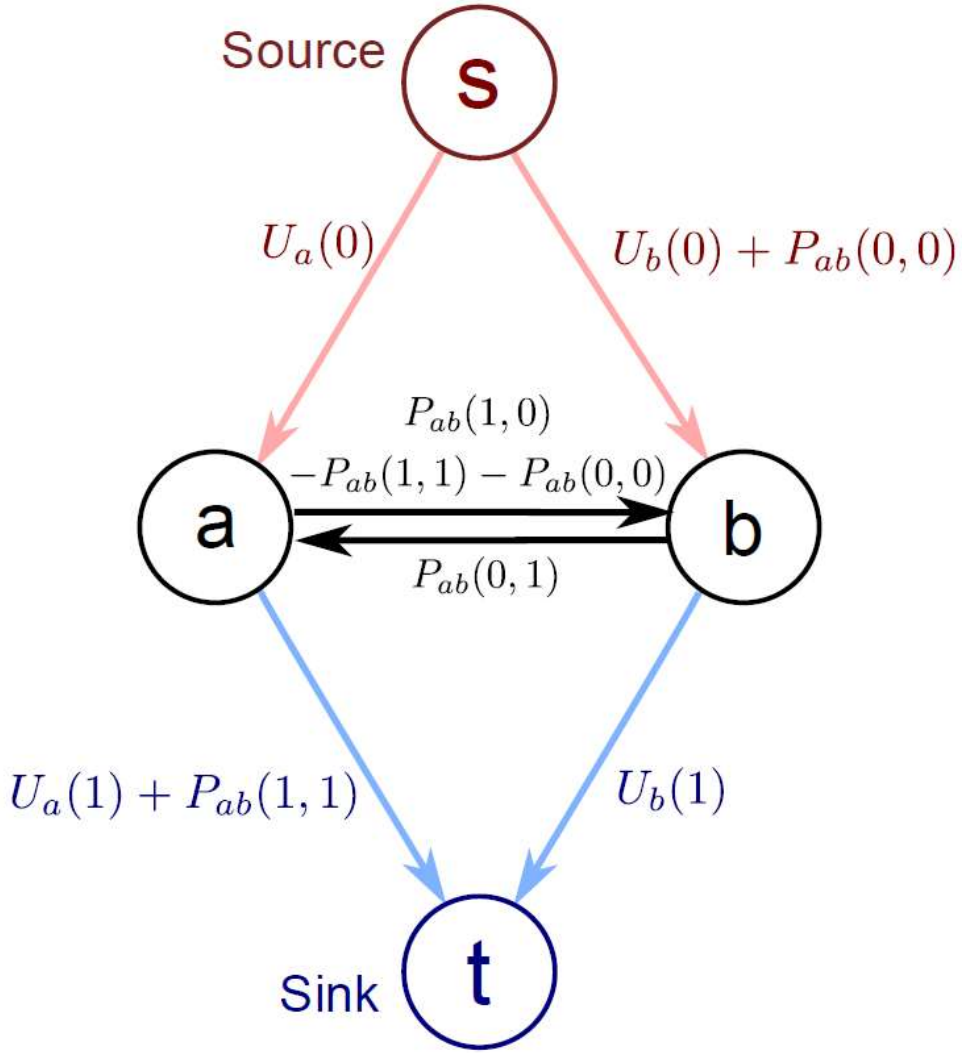


Reparameterization

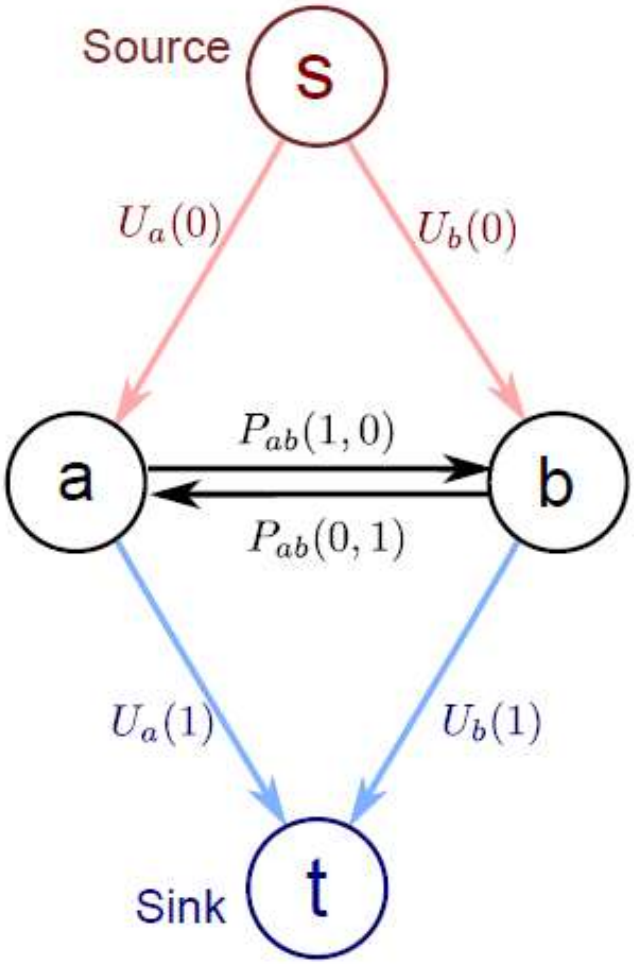
The max-flow / min-cut algorithms require that all of the capacities are non-negative.

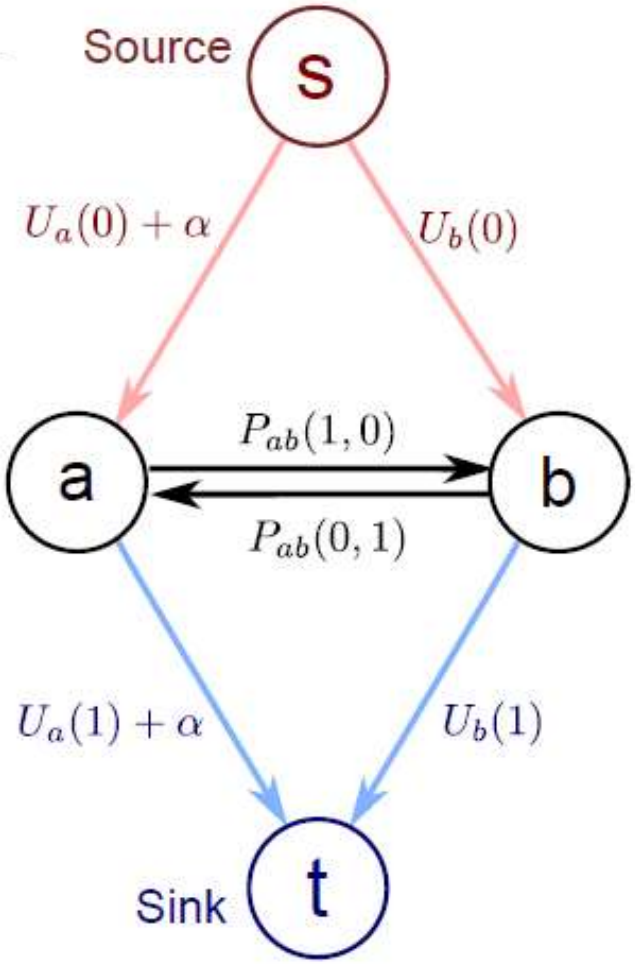
However, because we have a subtraction on edge a-b we cannot guarantee that this will be the case, even if all the original unary and pairwise costs were positive.

The solution to this problem is reparamaterization: find new graph where costs (capacities) are different but choice of minimum solution is the same (usually just by adding a constant to each solution)



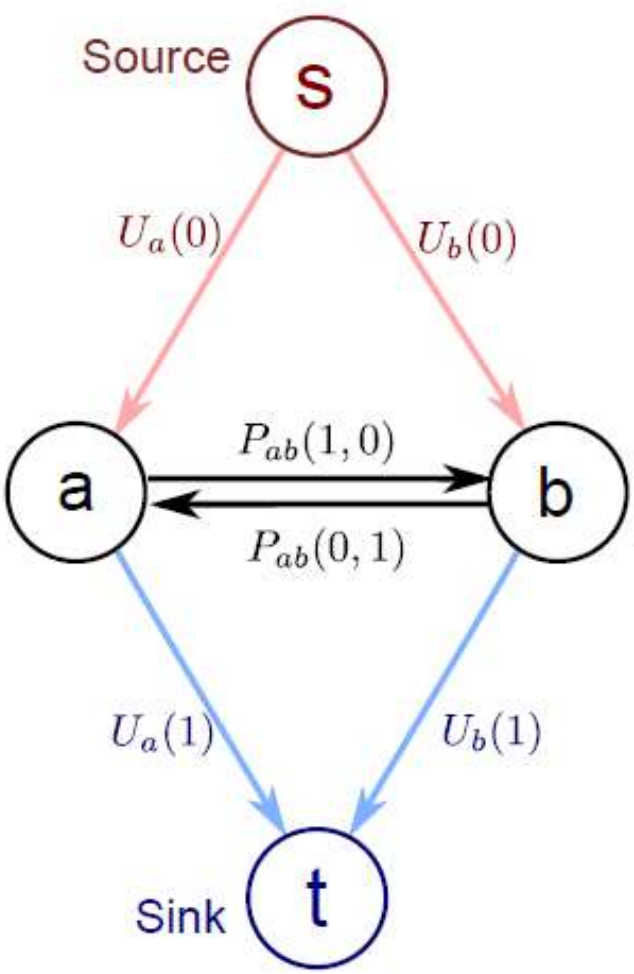
Reparameterization 1

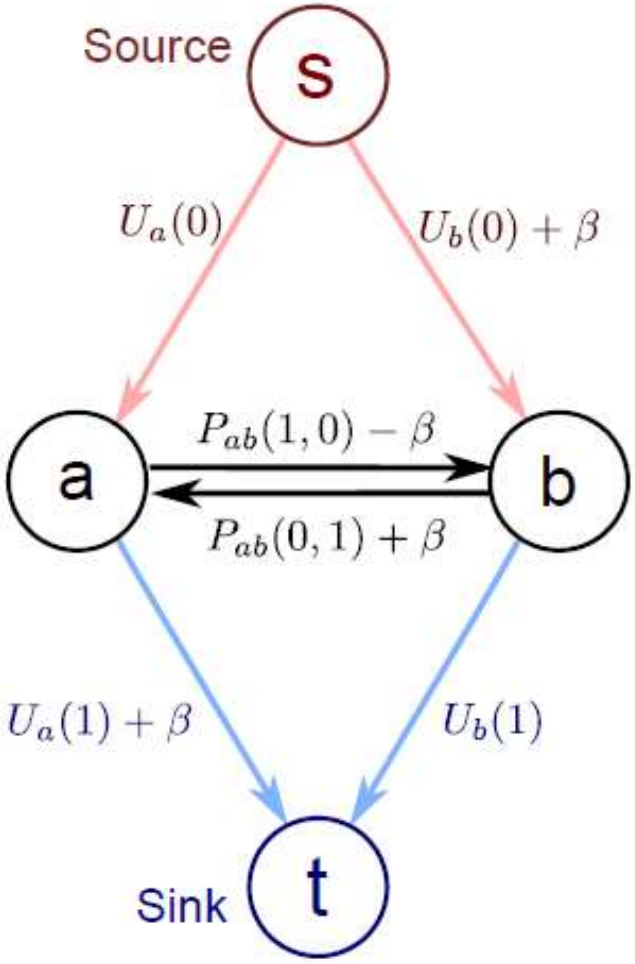




The minimum cut chooses the same links in these two graphs

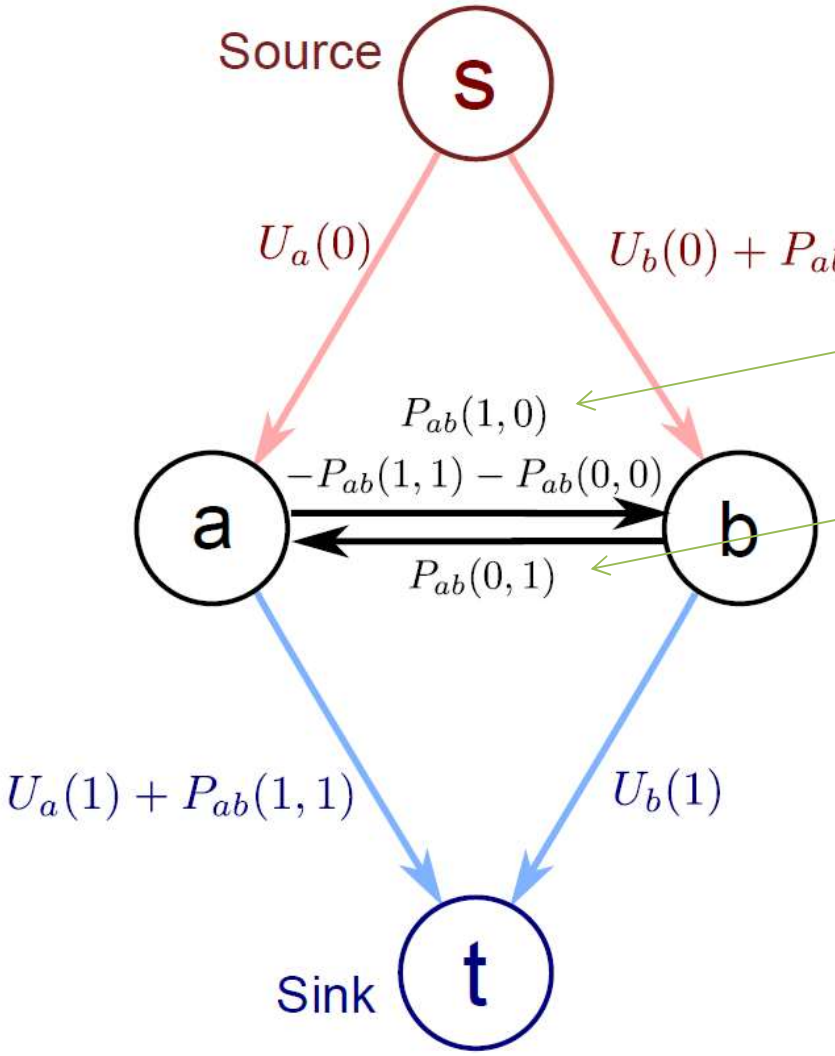
Reparameterization 2





The minimum cut chooses the same links in these two graphs

Submodularity



Subtract constant β

Add constant, β

$$\theta_{01} - \theta_{11} - \theta_{00} - \beta > 0$$

$$\theta_{10} + \beta > 0$$

Adding together implies

$$\theta_{10} + \theta_{01} - \theta_{11} - \theta_{00} > 0$$

Submodularity

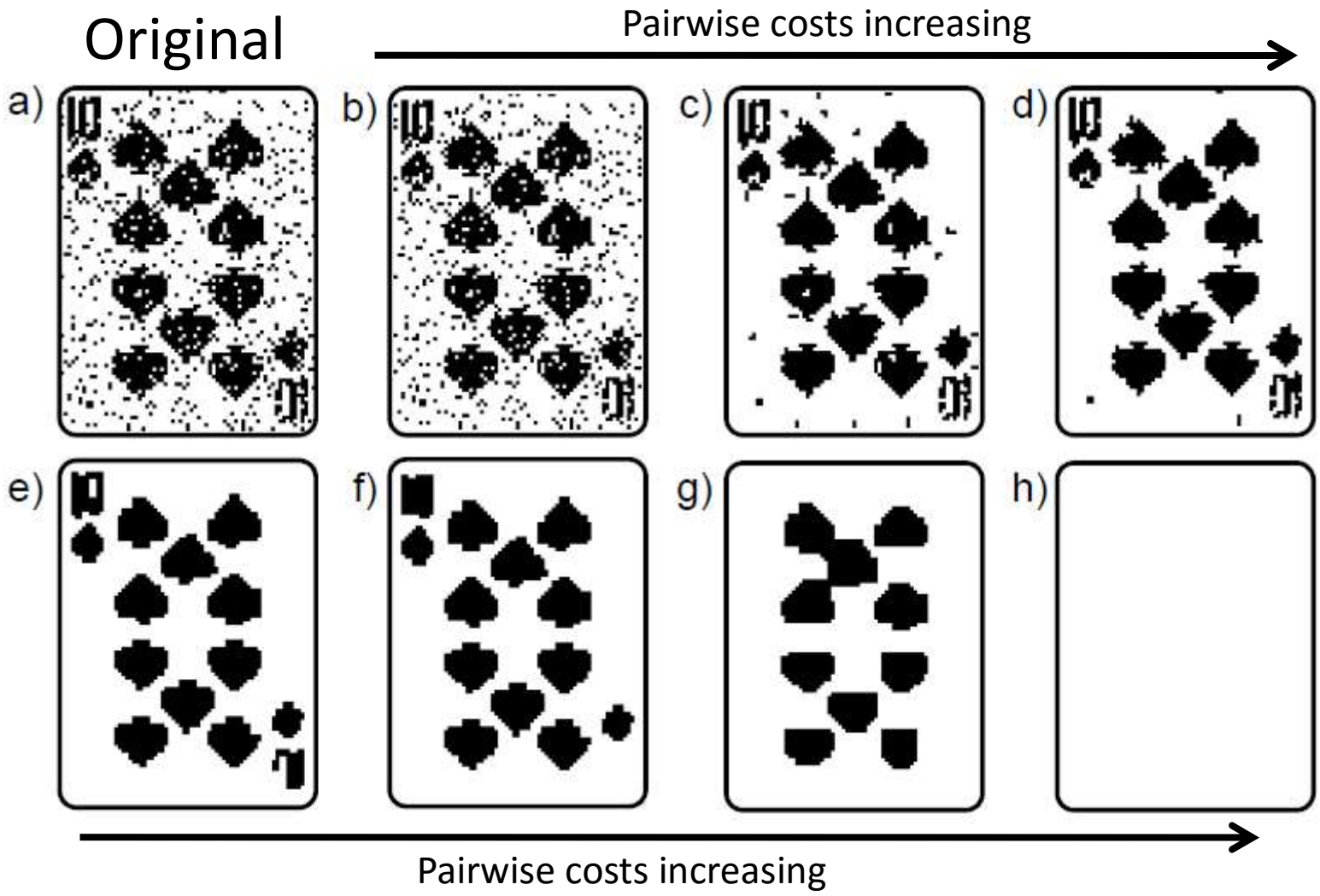
$$\theta_{10} + \theta_{01} - \theta_{11} - \theta_{00} > 0$$

If this condition is obeyed, it is said that the problem is “submodular” and it can be solved in polynomial time.

If it is not obeyed then the problem is NP hard.

Usually it is not a problem as we tend to favour smooth solutions.

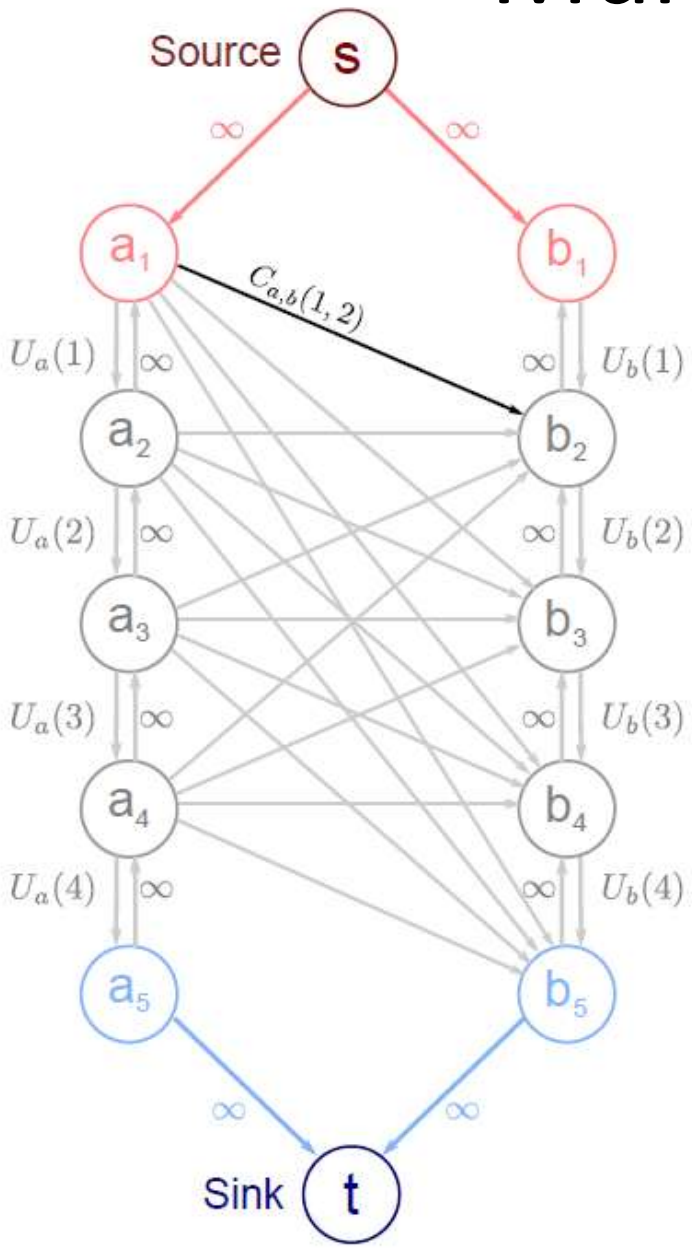
Denoising Results



Plan of Talk

- Denoising problem
- Markov random fields (MRFs)
- Max-flow / min-cut
- Binary MRFs – submodular (exact solution)
- Multi-label MRFs – submodular (exact solution)
- Multi-label MRFs - non-submodular (approximate)

Multiple Labels

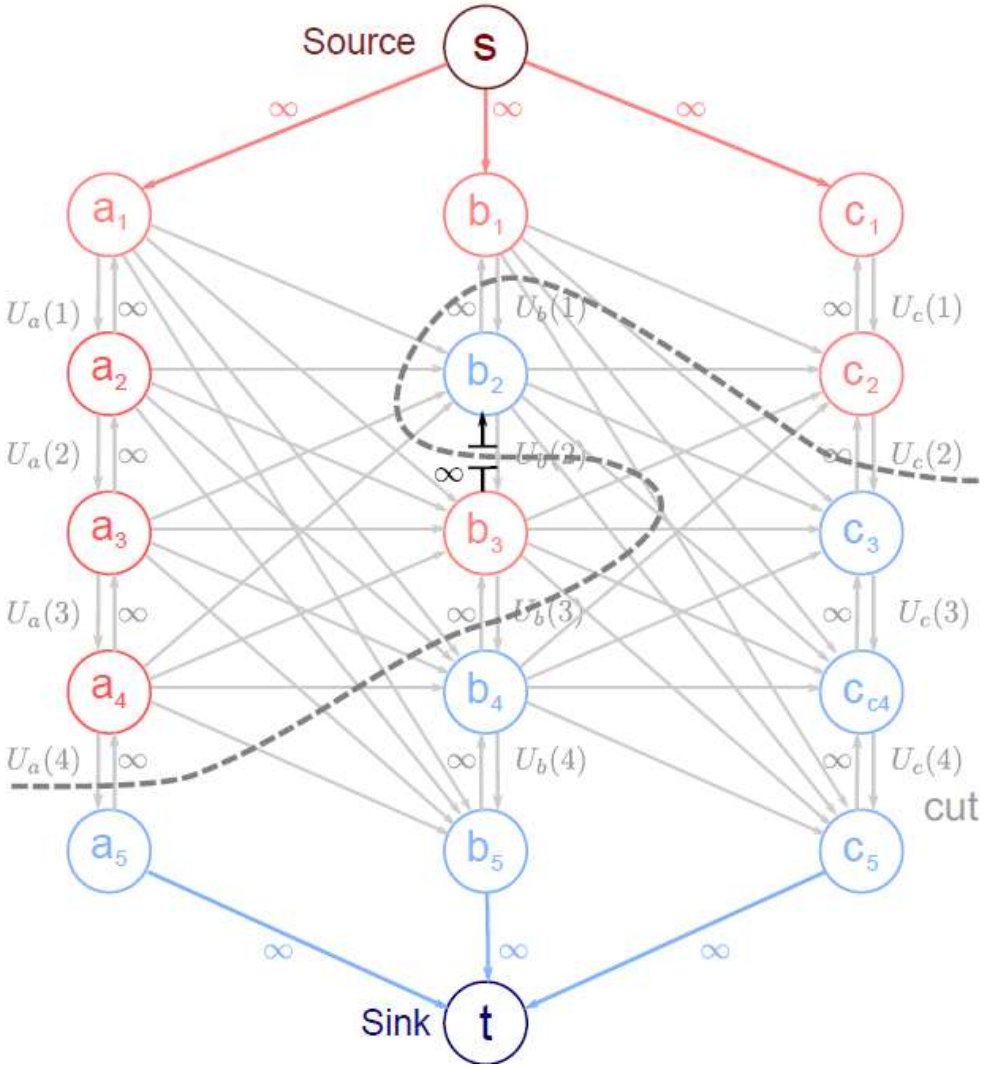


Construction for two pixels (a and b) and four labels (1,2,3,4)

There are 5 nodes for each pixel and 4 edges between them have unary costs for the 4 labels.

One of these edges must be cut in the min-cut solution and the choice will determine which label we assign.

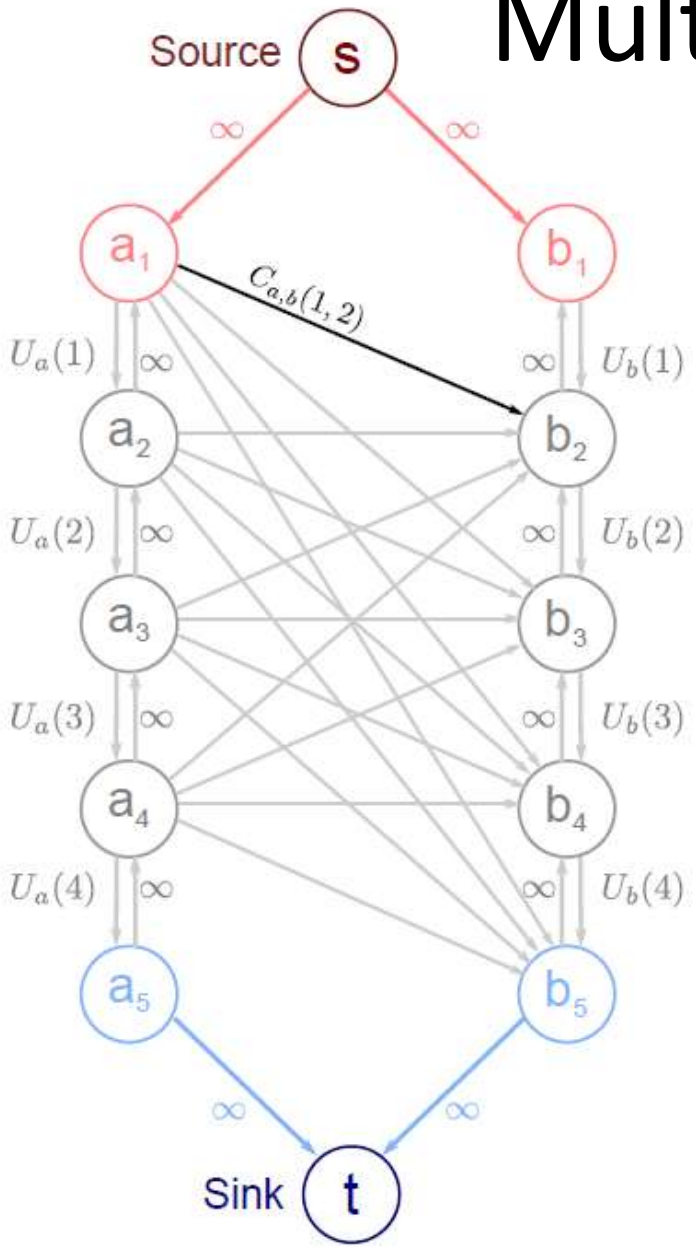
Constraint Edges



The edges with infinite capacity pointing upwards are called constraint edges.

They prevent solutions that cut the chain of edges associated with a pixel more than once (and hence given an ambiguous labelling)

Multiple Labels



Inter-pixel edges have costs defined as:

$$C_{ab}(i,j) =$$

$$P_{ab}(i,j-1) + P_{ab}(i-1,j)$$

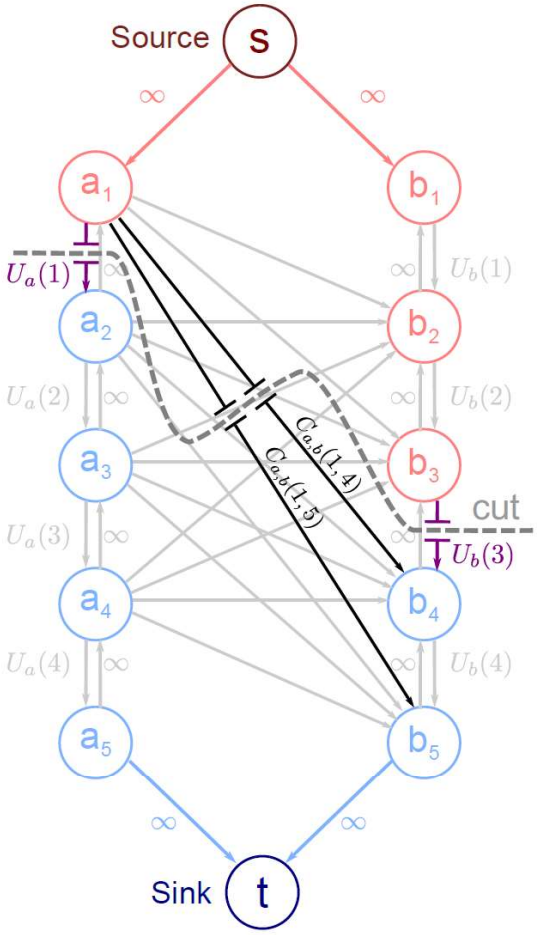
$$- P_{a,b}(i,j) - P_{ab}(i-1,j-1)$$

Superfluous terms :

$$P_{ab}(i,0) = 0 \qquad P_{ab}(i,K+1) = 0$$
$$P_{ab}(0,j) = 0 \qquad P_{ab}(K+1,j) = 0$$

For all i,j where K is number of labels

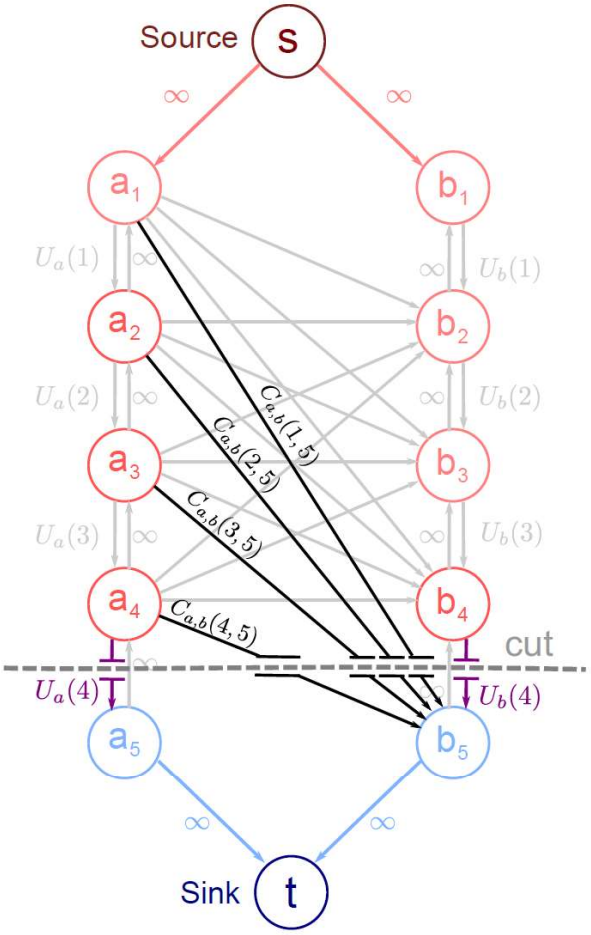
Example Cuts



1	3

Cost

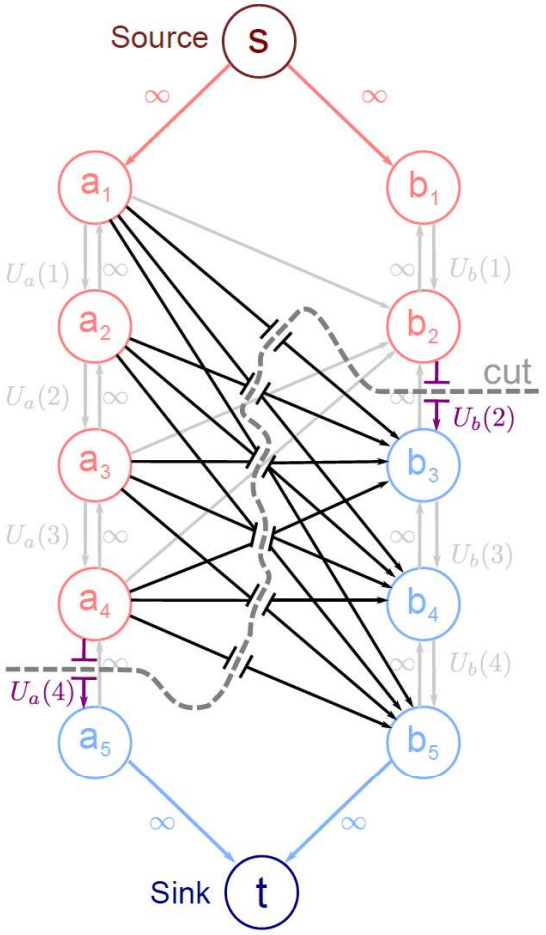
$$U_a(1) + U_b(3) + \sum_{i=1}^{1}\sum_{j=4}^{5} C_{ab}(i,j)$$
$$= U_a(1) + U_b(3) + P_{ab}(1,3)$$



4	4

Cost

$$U_a(4) + U_b(4) + \sum_{i=1}^{4}\sum_{j=5}^{5} C_{ab}(i,j)$$
$$= U_a(4) + U_b(4) + P_{ab}(4,4)$$



4	2

Cost

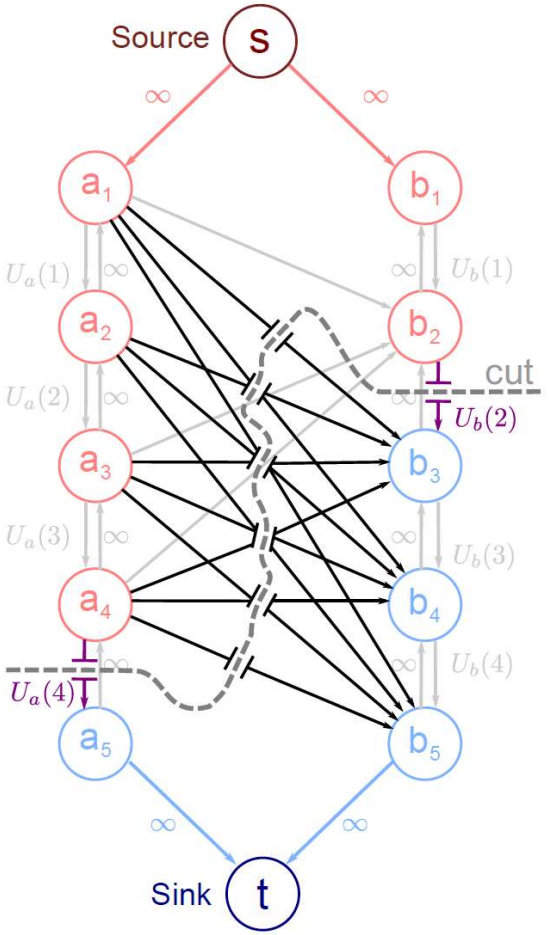
$$U_a(4) + U_b(2) + \sum_{i=1}^{4}\sum_{j=3}^{5} C_{ab}(i,j)$$
$$= U_a(4) + U_b(2) + P_{ab}(4,2)$$

Must cut links from before cut on pixel a to after cut on pixel b.

Pairwise Costs

Must cut links from before cut on pixel a to after cut on pixel b.

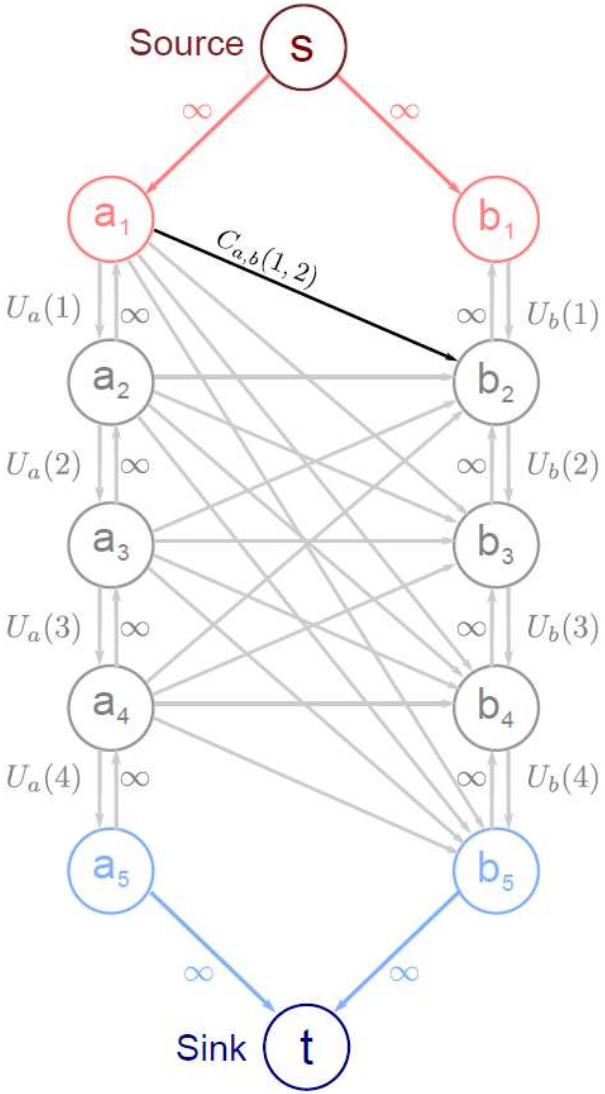
Costs were carefully chosen so that sum of these links gives appropriate pairwise term.

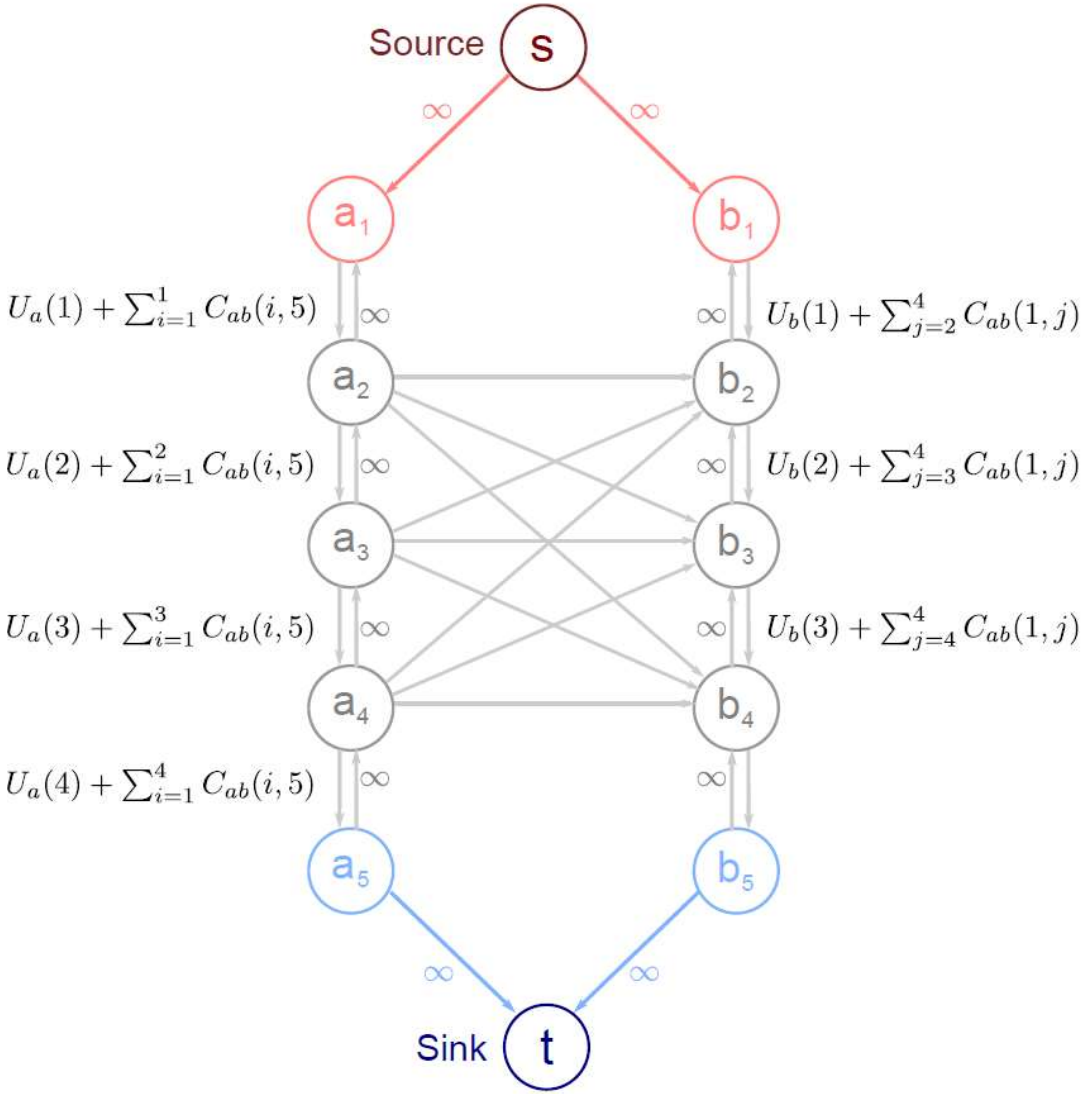


If pixel a takes label I and pixel b takes label J

$$\begin{aligned}\sum_{i=1}^{I}\sum_{j=J+1}^{K+1} C_{ab}(i,j) &= \sum_{i=1}^{I}\sum_{j=J+1}^{K+1} P_{ab}(i,j-1)+P_{ab}(i-1,j)-P_{a,b}(i,j)-P_{ab}(i-1,j-1)\\ &= P_{ab}(I,J)+P_{ab}(0,K+1)-P_{a,b}(I,K+1)-P_{ab}(0,K+1)\\ &= P_{ab}(I,J), \qquad (11.29)\end{aligned}$$

Reparameterization





Submodularity

We require the remaining inter-pixel links to be positive so that

$$C_{ab}(i,j) \geq 0$$

or

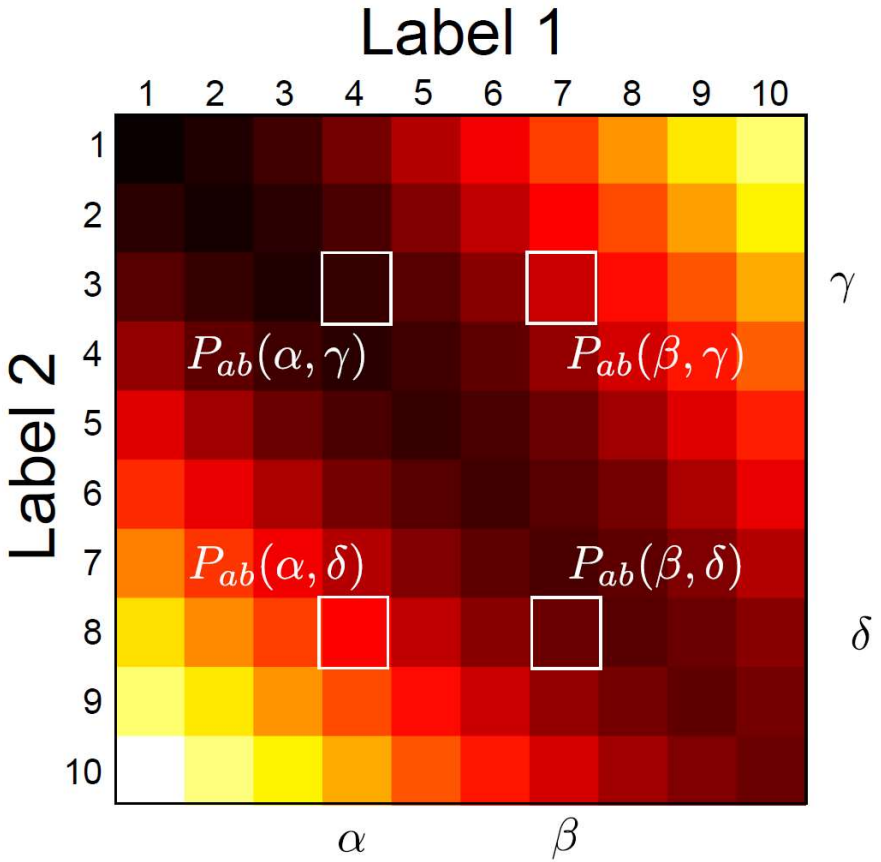
$$P_{ab}(i,j-1) + P_{ab}(i-1,j) - P_{a,b}(i,j) - P_{ab}(i-1,j-1) \geq 0$$

By mathematical induction we can get the more general result

$$P_{ab}(\beta,\gamma) + P_{ab}(\alpha,\delta) - P_{a,b}(\beta,\delta) - P_{ab}(\alpha,\gamma) \geq 0,$$

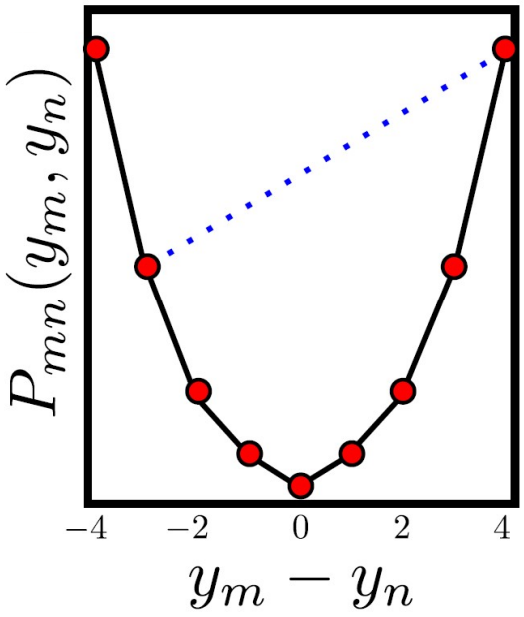
Submodularity

$$P_{ab}(\beta,\gamma) + P_{ab}(\alpha,\delta) - P_{a,b}(\beta,\delta) - P_{ab}(\alpha,\gamma) \geq 0,$$



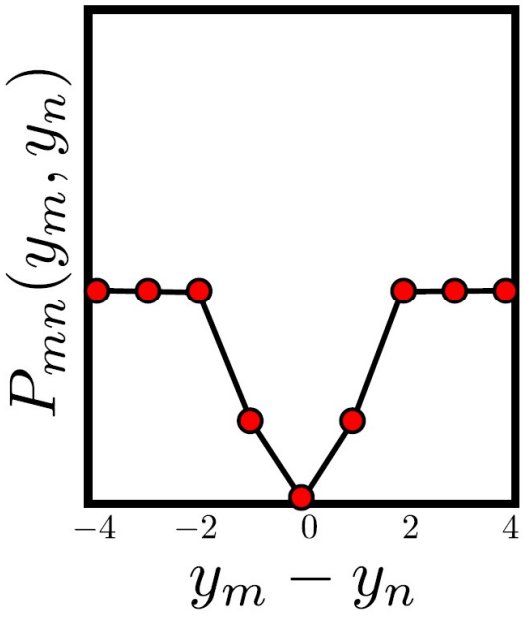
If not submodular then the problem is NP hard.

Convex vs. non-convex costs



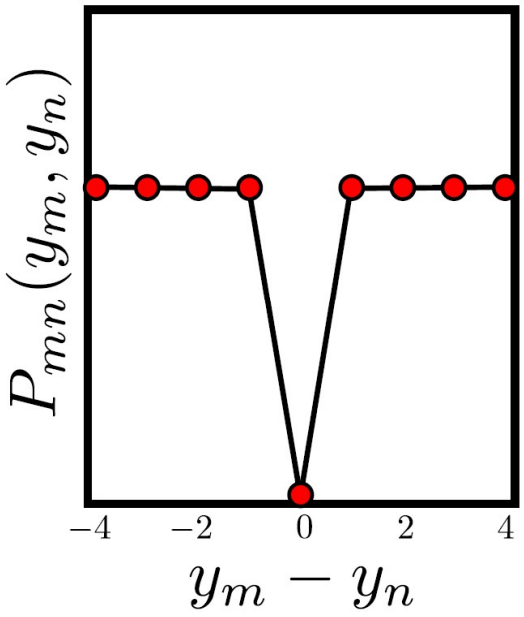
Quadratic

- Convex
- Submodular



Truncated Quadratic

- Not Convex
- Not Submodular

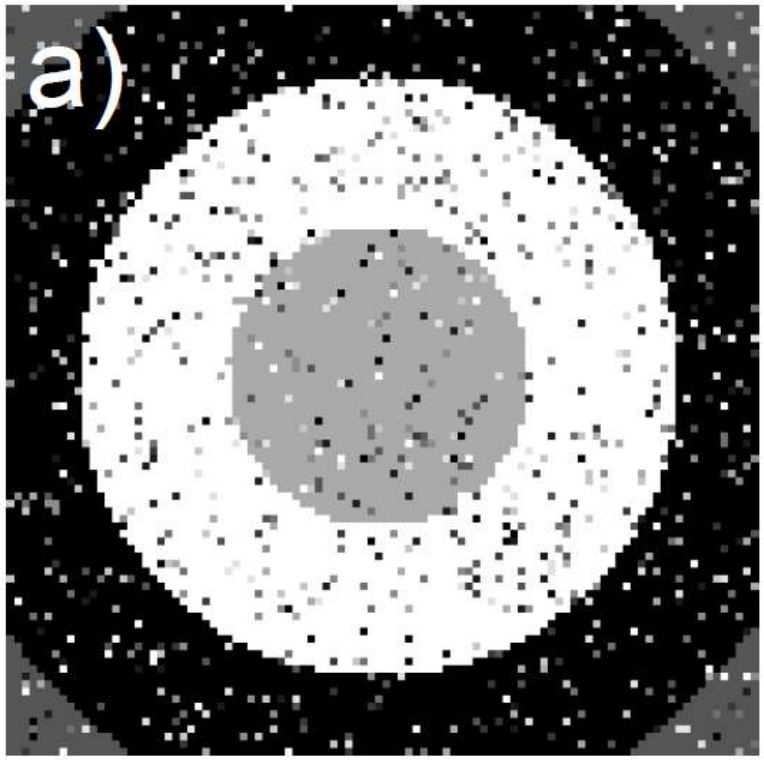


Potts Model

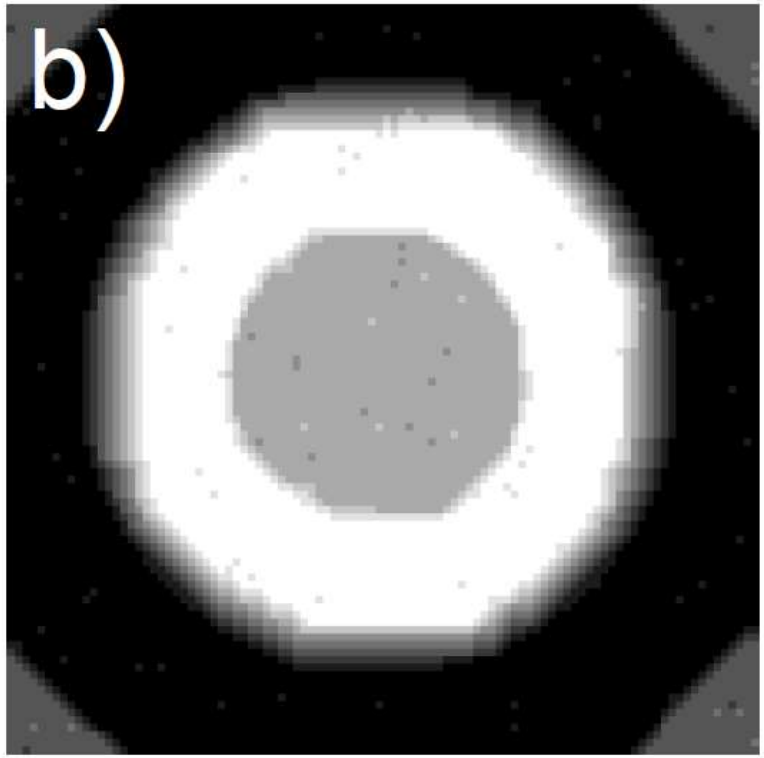
- Not Convex
- Not Submodular

What is wrong with convex costs?

Observed noisy image



Denoised result



- Pay lower price for many small changes than one large one
- Result: blurring at large changes in intensity

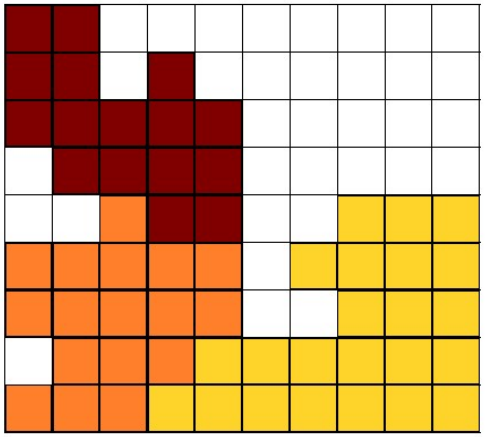
Plan of Talk

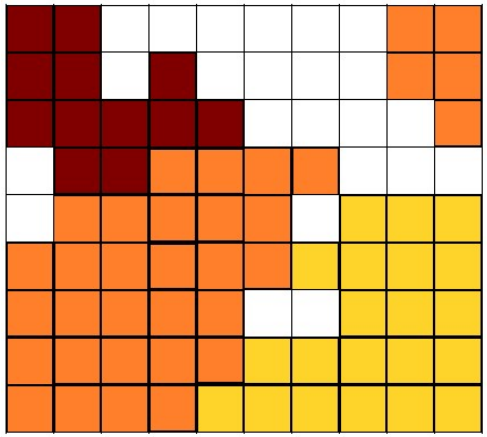
- Denoising problem
- Markov random fields (MRFs)
- Max-flow / min-cut
- Binary MRFs - submodular (exact solution)
- Multi-label MRFs – submodular (exact solution)
- Multi-label MRFs - non-submodular (approximate)

Alpha Expansion Algorithm

- break multilabel problem into a series of binary problems
- at each iteration, pick label α and expand (retain original or change to α)

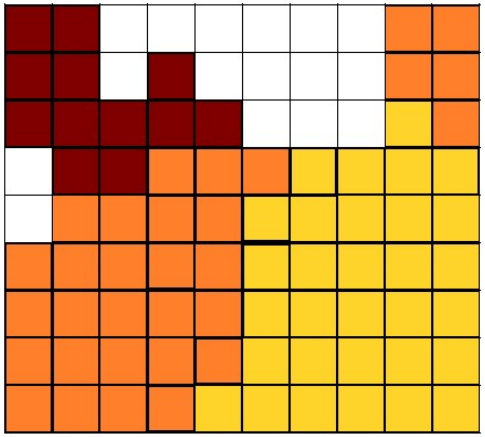
Initial labelling

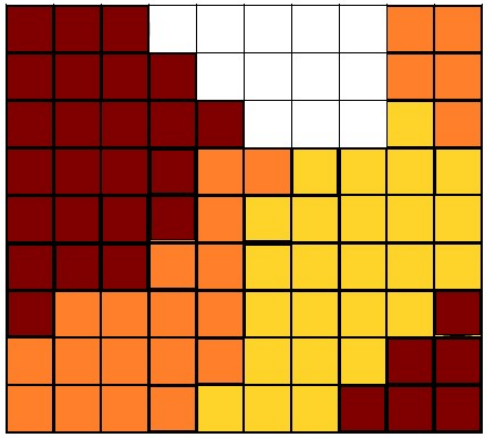




Iteration 1 (orange)

Iteration 2 (yellow)





Iteration 3 (red)

Alpha Expansion Ideas

- For every iteration
  - For every label
  - Expand label using optimal graph cut solution

Co-ordinate descent in label space.

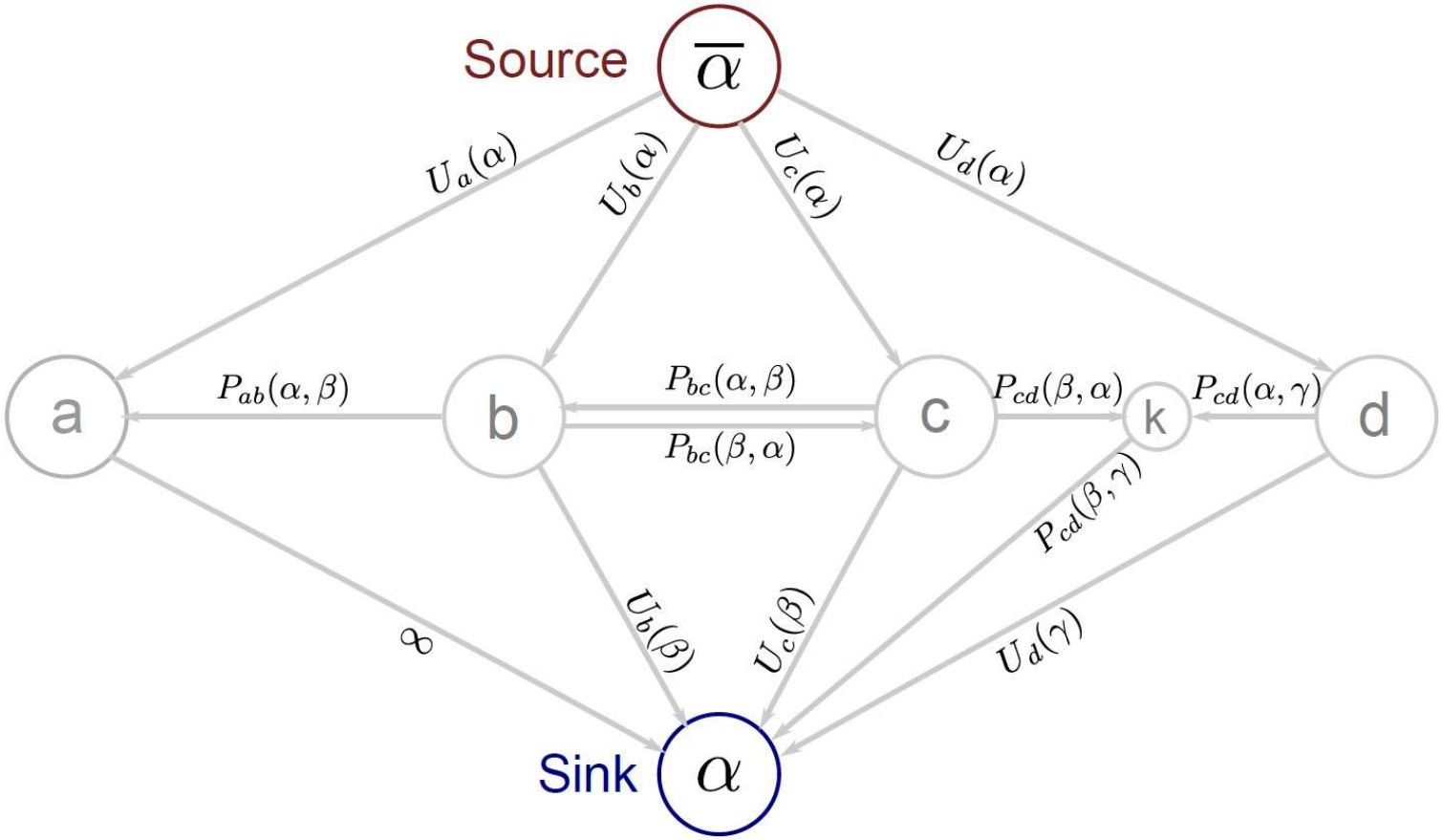
Each step optimal, but overall global maximum not guaranteed

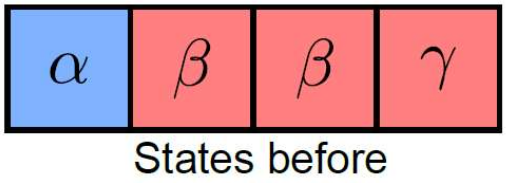
Proved to be within a factor of 2 of global optimum.

Requires that pairwise costs form a metric:

$$\begin{aligned} P(\alpha,\beta) = \quad & 0 \quad \Leftrightarrow \alpha = \beta \\ P(\alpha,\beta) \quad = \quad & P(\beta,\alpha) > 0 \\ P(\alpha,\beta) \quad \leq \quad & P(\alpha,\gamma) + P(\gamma,\beta) \end{aligned}$$

Alpha Expansion Construction

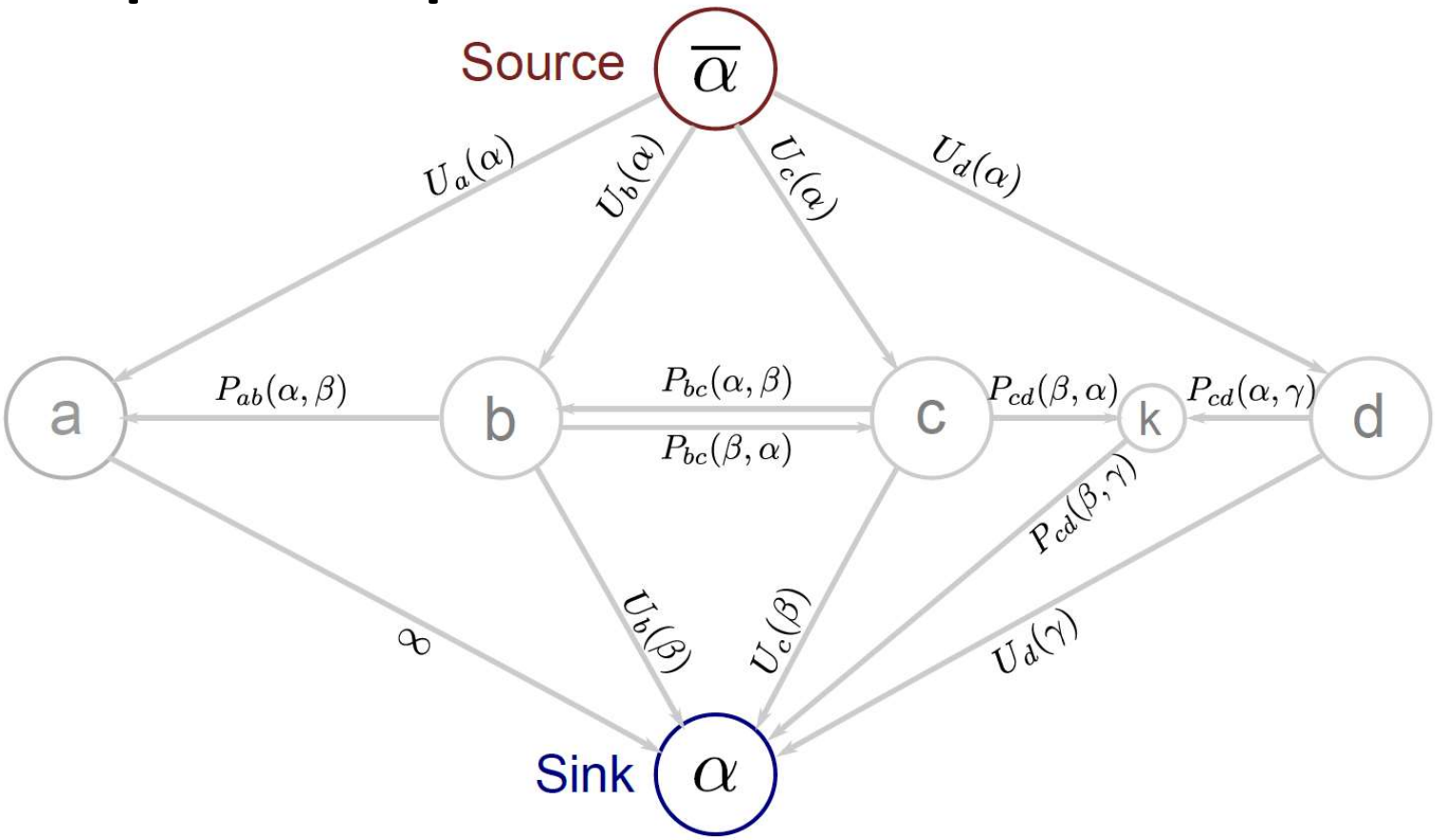


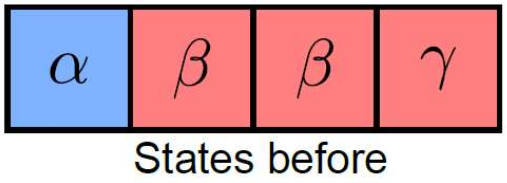


Binary graph cut – either cut link to source (assigned to α) or to sink (retain current label)

Unary costs attached to links between source, sink and pixel nodes appropriately.

Alpha Expansion Construction

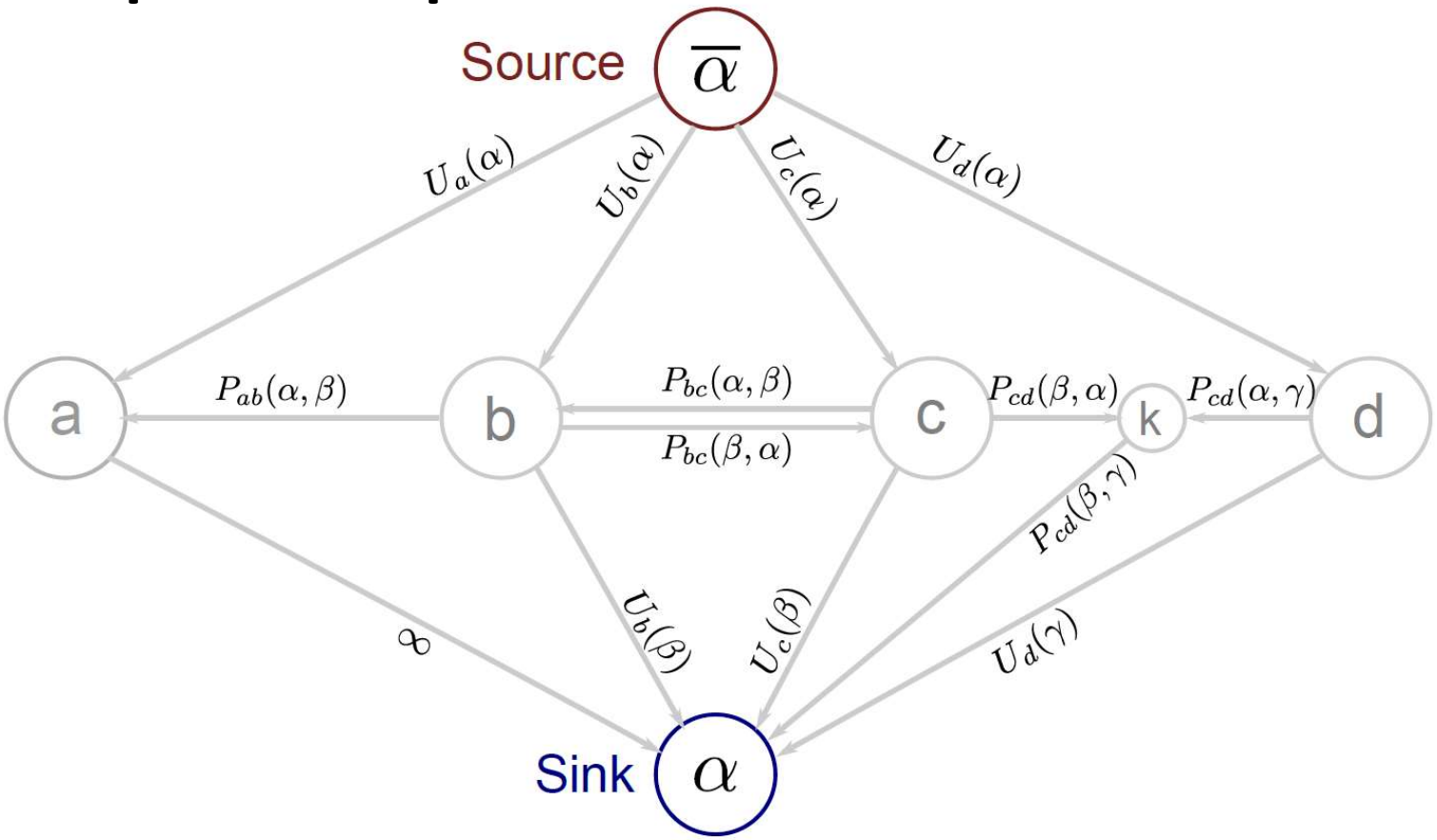


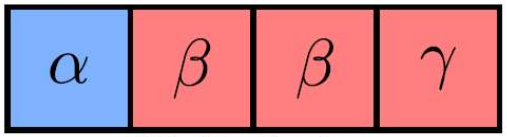


Graph is dynamic. Structure of inter-pixel links depends on α and the choice of labels.

There are four cases.

Alpha Expansion Construction

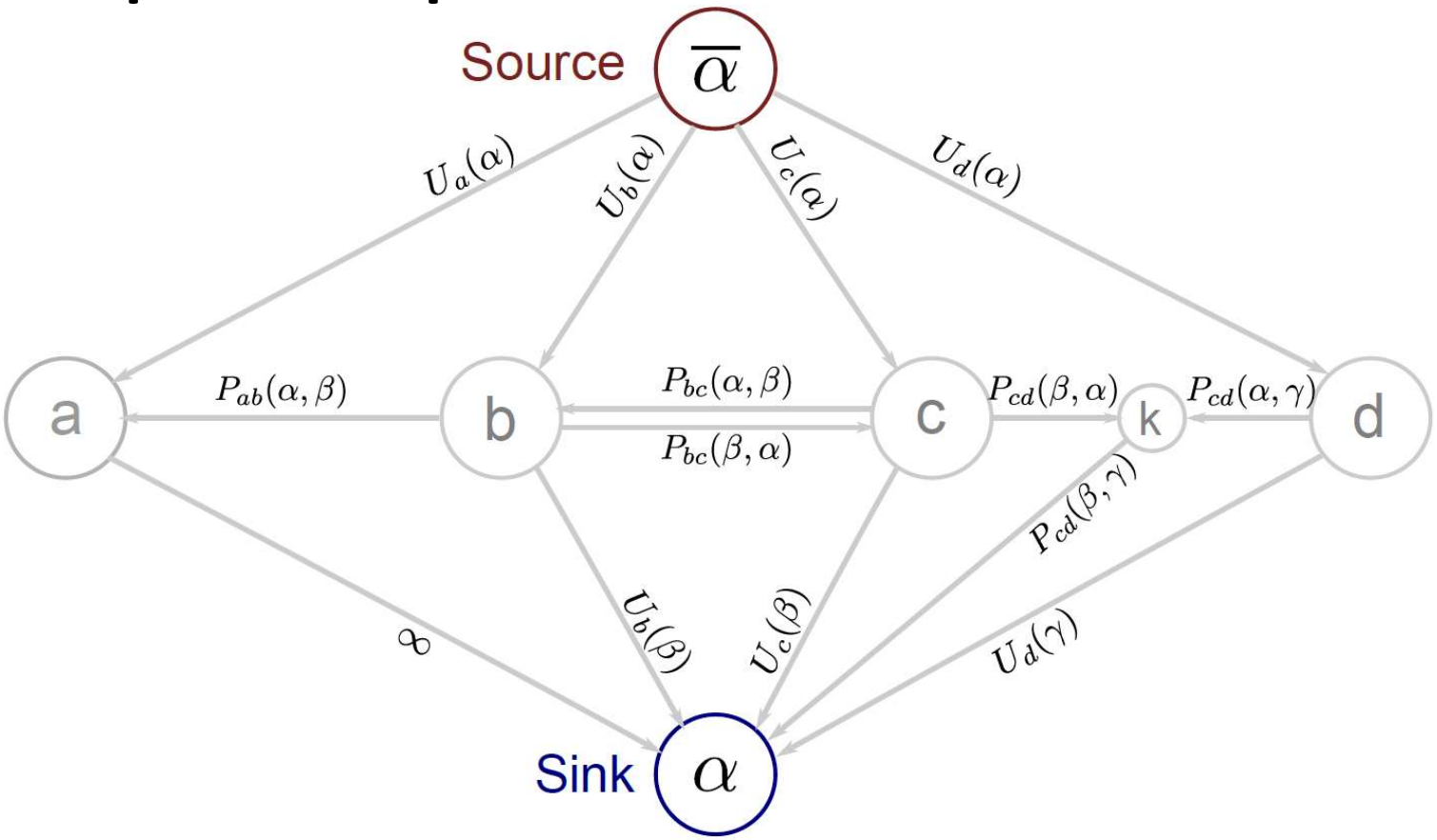


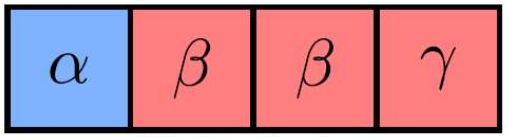


Case 1:

Adjacent pixels both have label α already.
Pairwise cost is zero – no need for extra edges.

Alpha Expansion Construction

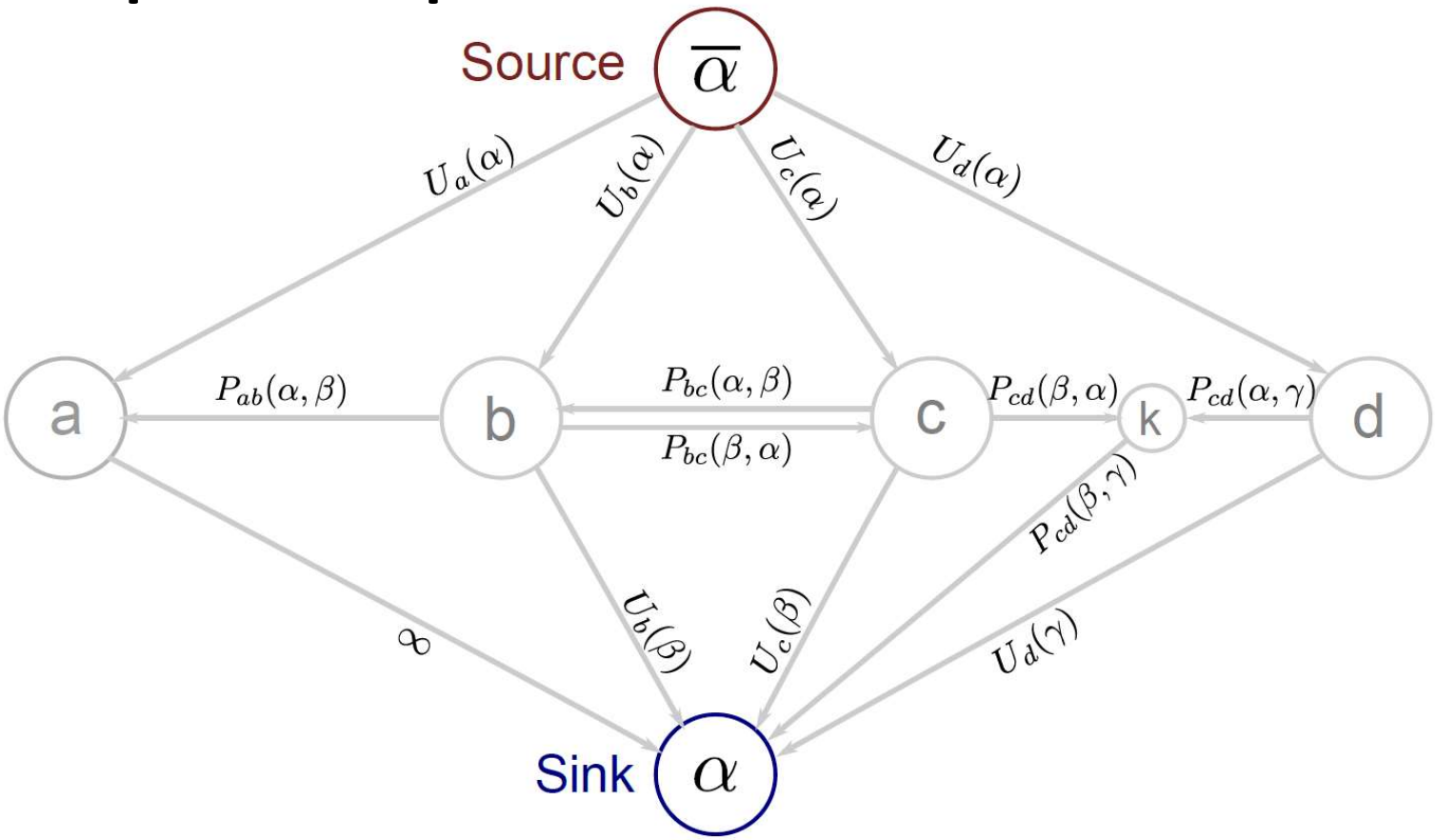


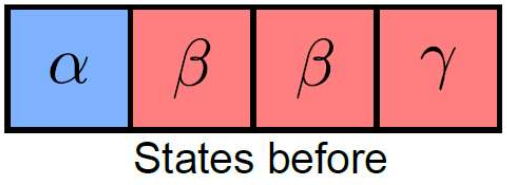


Case 2: Adjacent pixels are α,β.
Result either

- α,α (no cost and no new edge).
- α,β (P(α,β), add new edge).

Alpha Expansion Construction

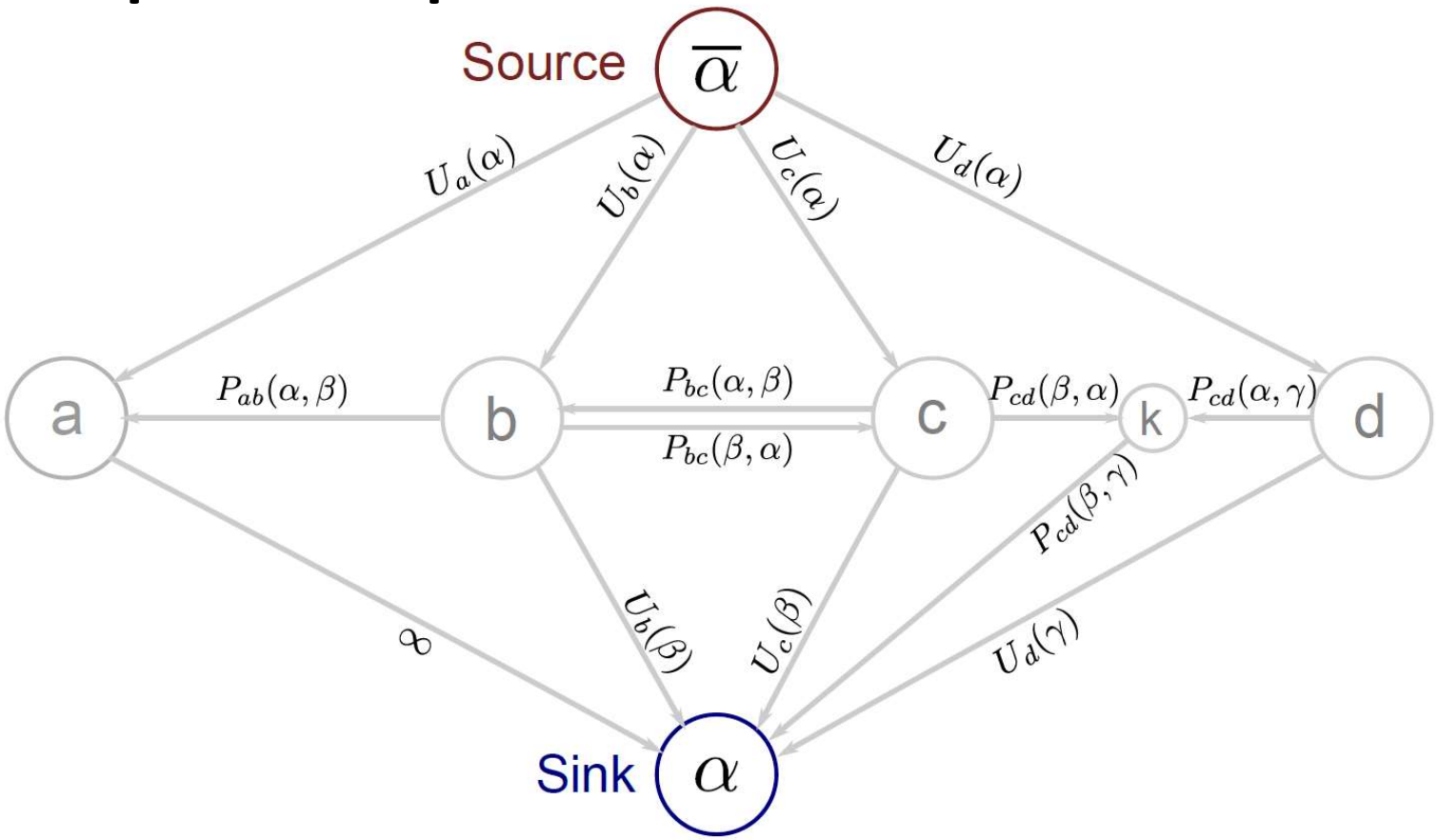


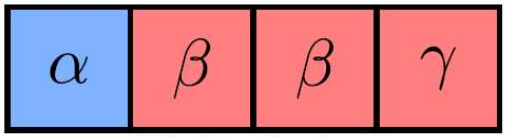


Case 3: Adjacent pixels are β,β. Result either

- β,β (no cost and no new edge).
- α,β (P(α,β), add new edge).
- β,α (P(β,α), add new edge).

Alpha Expansion Construction

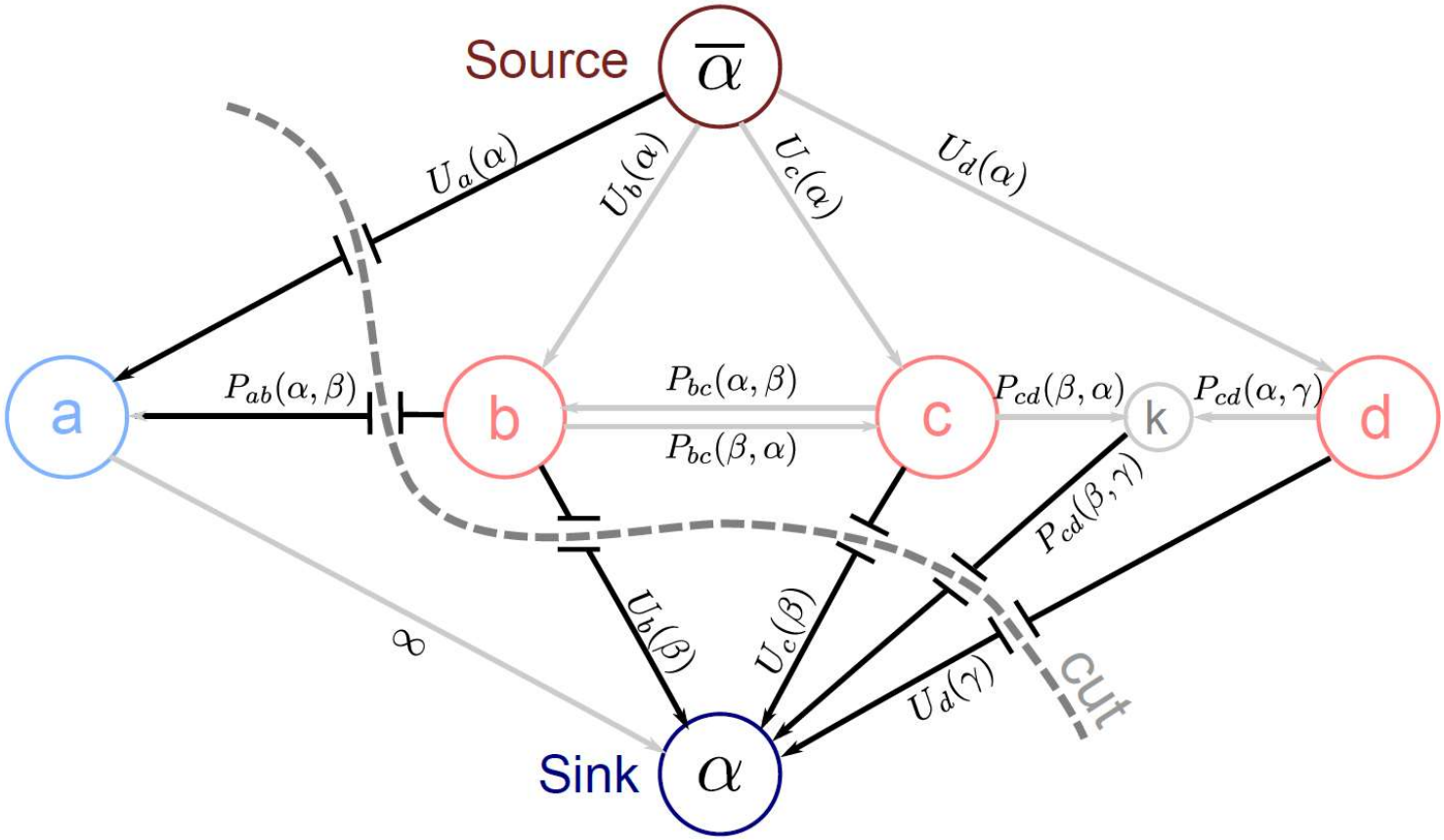


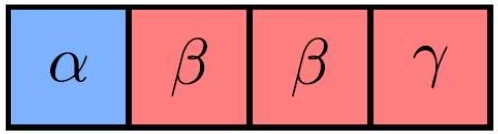


Case 4: Adjacent pixels are β,γ. Result either

- β,γ (P(β,γ), add new edge).
- α,γ (P(α,γ), add new edge).
- β,α (P(β,α), add new edge).
- α,α (no cost and no new edge).

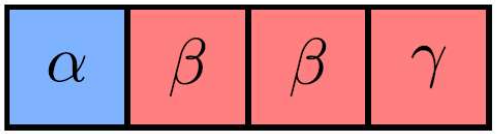
Example Cut 1





States before

Cost

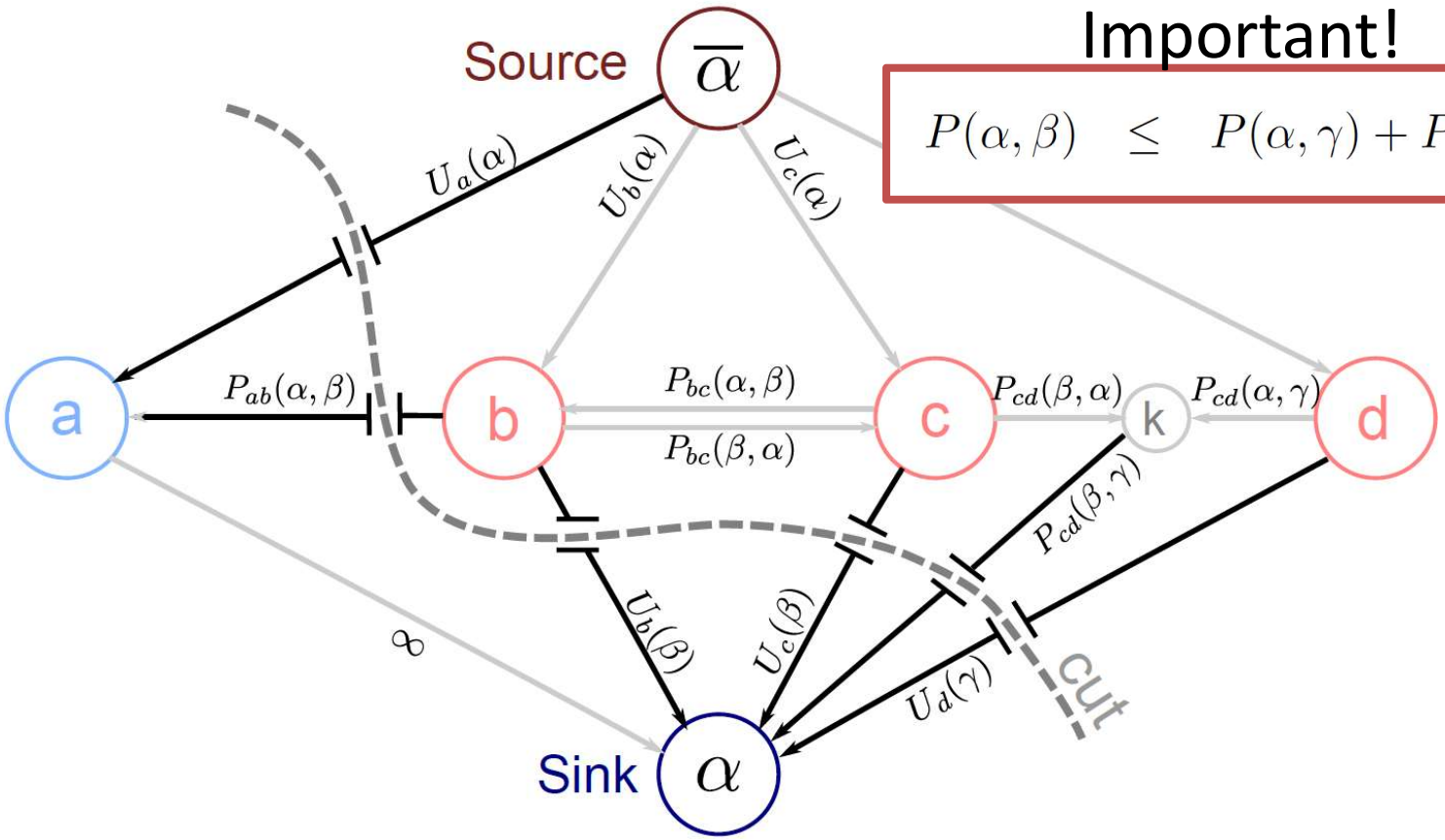


States after

$$U_a(\alpha) + U_b(\beta) + U_c(\beta) + U_d(\gamma)$$

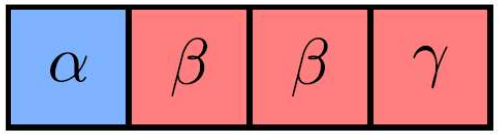
$$P_{ab}(\alpha,\beta) + P_{bc}(\beta\gamma)$$

Example Cut 1

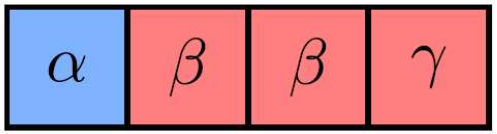


Important!

$$P(\alpha,\beta) \leq P(\alpha,\gamma) + P(\gamma,\beta)$$



States before



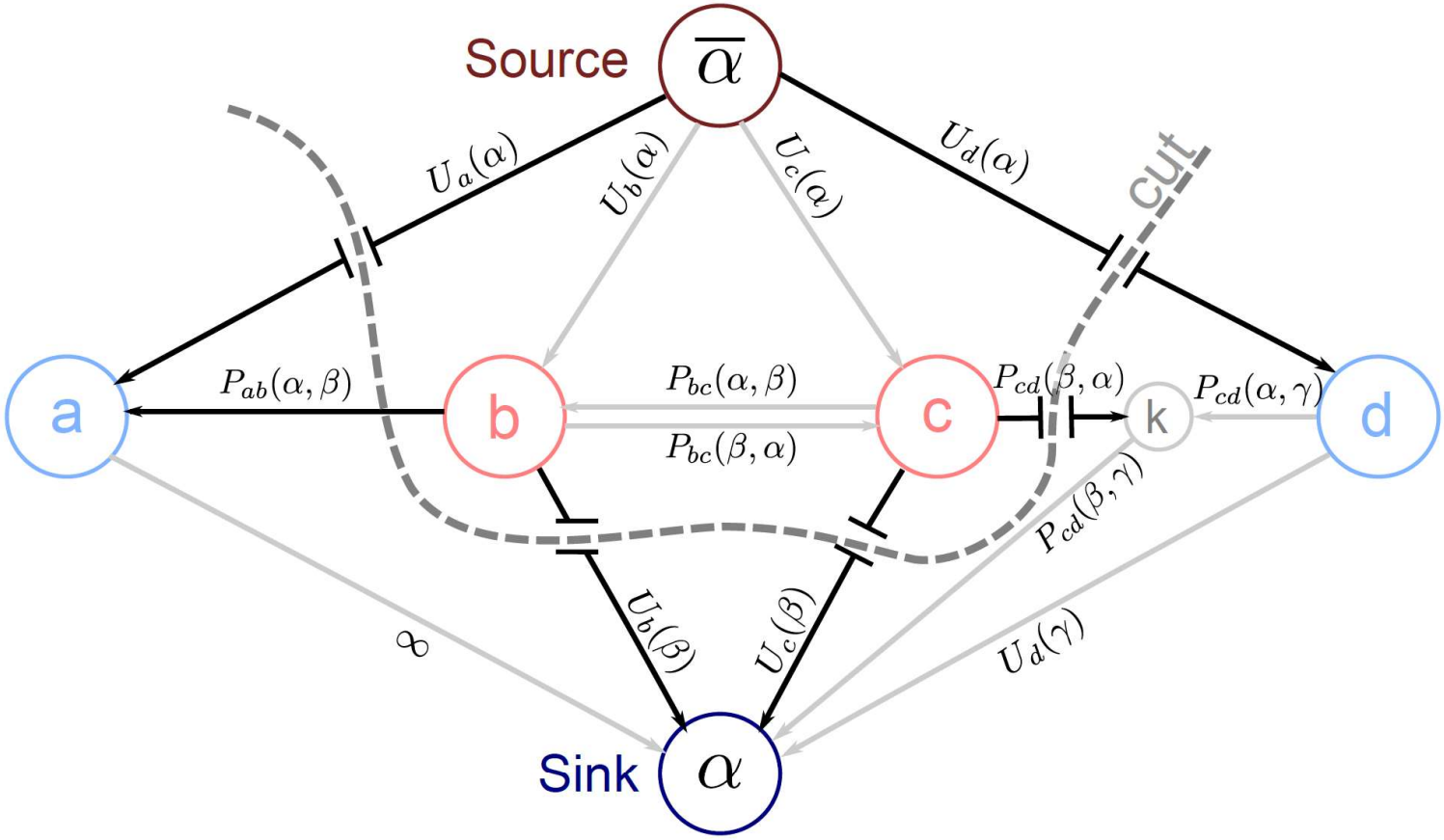
States after

Cost

$$U_a(\alpha) + U_b(\beta) + U_c(\beta) + U_d(\gamma)$$

$$P_{ab}(\alpha,\beta) + P_{bc}(\beta\gamma)$$

Example Cut 2



α	β	β	γ

States before

α	β	β	α

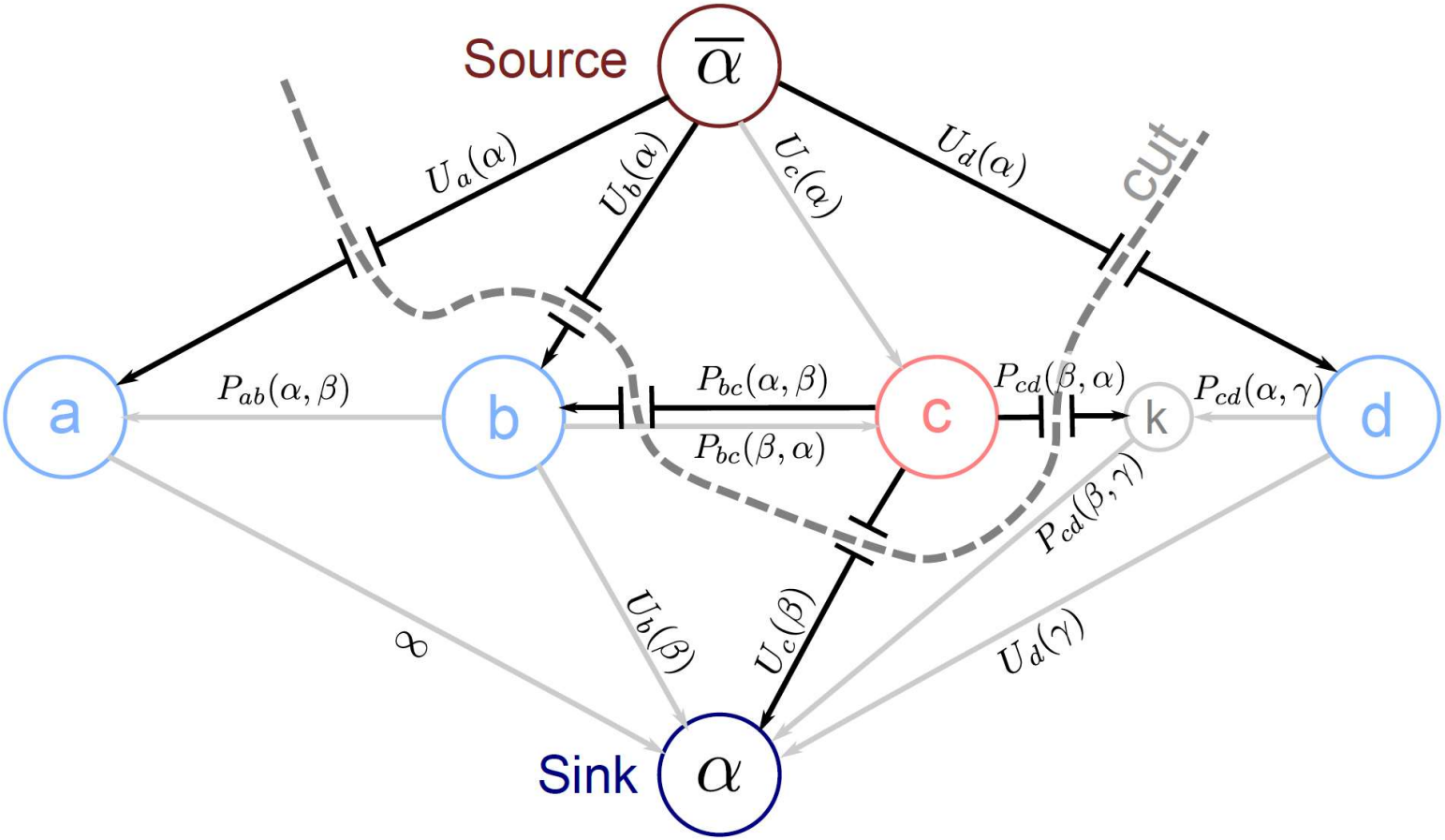
States after

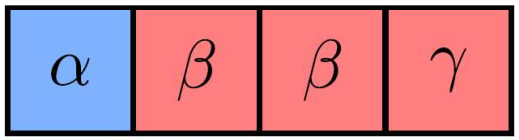
Cost

$$U_a(\alpha) + U_b(\beta) + U_c(\beta) + U_d(\alpha)$$

$$P_{ab}(\alpha,\beta) + P_{bc}(\beta\alpha)$$

Example Cut 3



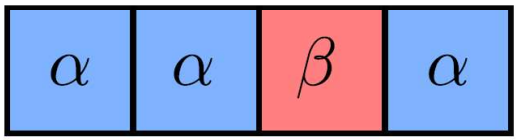


States before

Cost

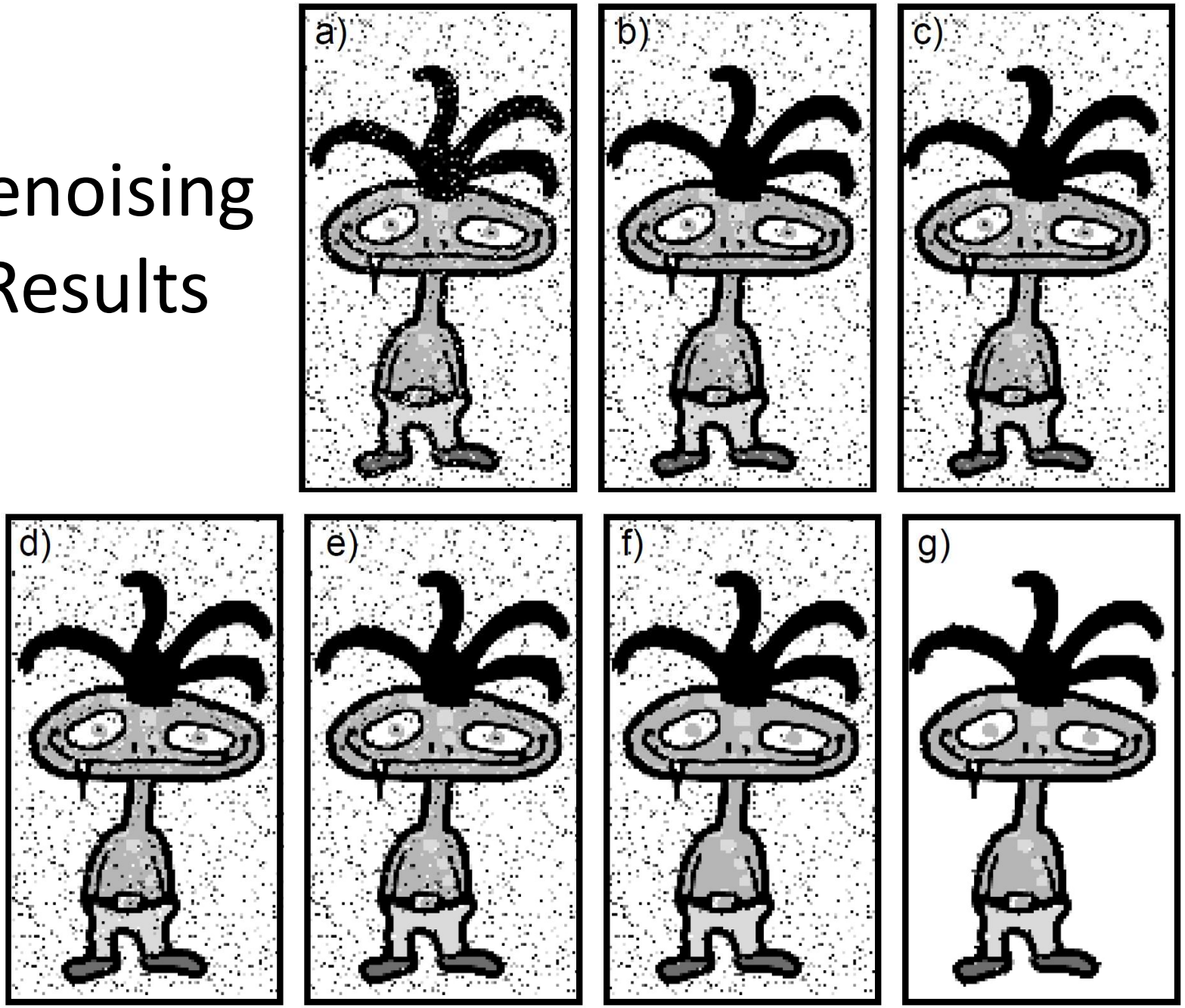
$$U_a(\alpha) + U_b(\alpha) + U_c(\beta) + U_d(\alpha)$$

$$P_{bc}(\alpha,\beta) + P_{bc}(\beta\alpha)$$



States after

Denoising Results



Conclusions

- Graph cuts help you find the MAP solution in models with pairwise MRF priors (also CRFs!)
  - Exact solution in binary case if submodular
  - Exact solution in multi-label case if submodular
  - Approximate solution in multi-label case if a metric

Contact me if you want slides or notes

s.prince@cs.ucl.ac.uk