UNIVERSITÄT BONN

Juergen Gall

Markov Random Fields

MA-INF 2201 - Computer Vision

WS24/25

Organization

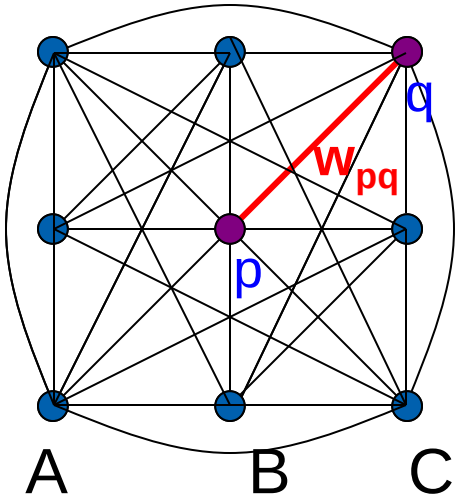
- No lecture on 15.11.
- Next lecture on 19.11.

Literature

Chapter 12 Models for Grids

S. Prince. **Computer Vision: Models, Learning, and Inference.** Cambridge University Press 2012

Segmentation by Graph Cuts



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

Source: S. Seitz

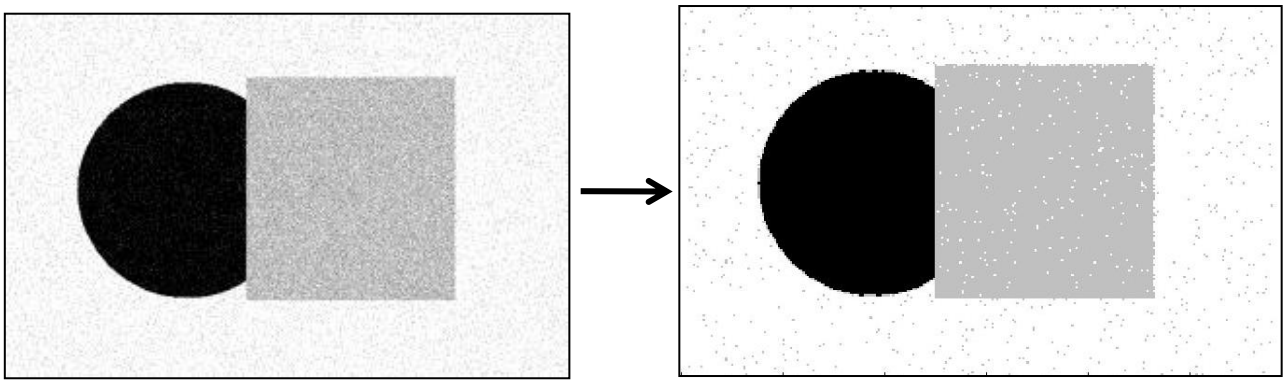
Models for grids

- Consider models with one unknown world state at each pixel in the image – takes the form of a grid.
- Loops in the graphical model, so cannot use dynamic programming
- Define probability distributions that favor certain configurations of world states
  - Called Markov random fields
  - Inference using a set of techniques called graph cuts

Smoothing out cluster assignments

UNIVERSITÄT BONN

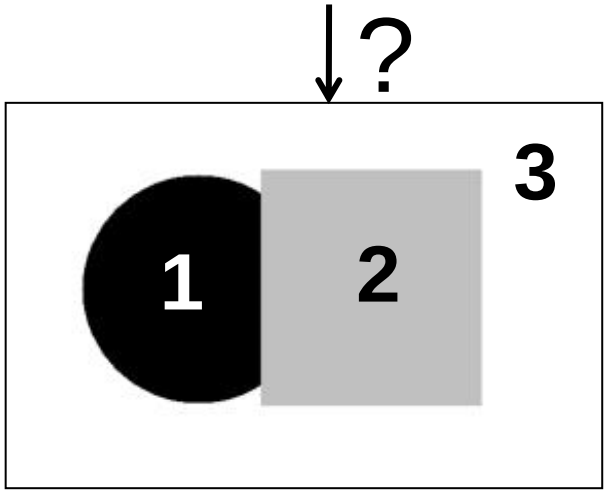
- Assigning a cluster label per pixel may yield outliers:



original

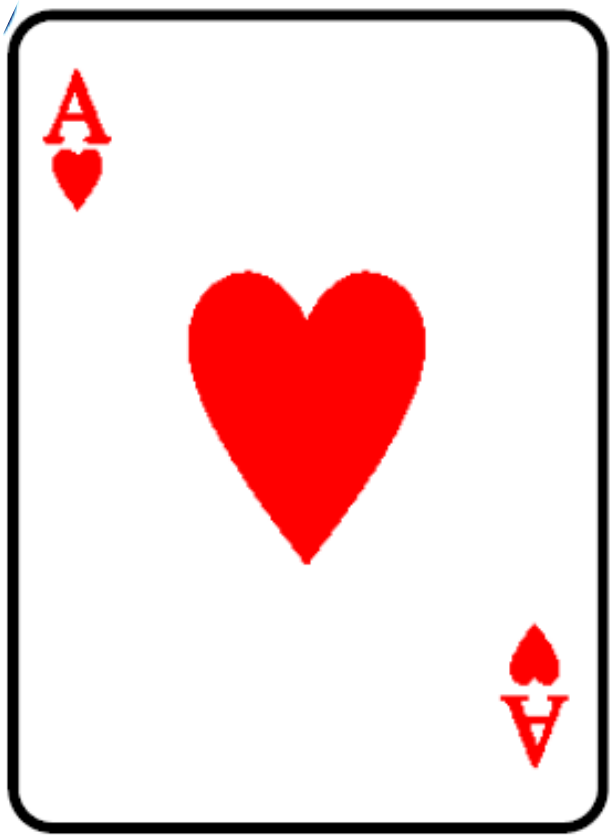
labeled by cluster center's intensity

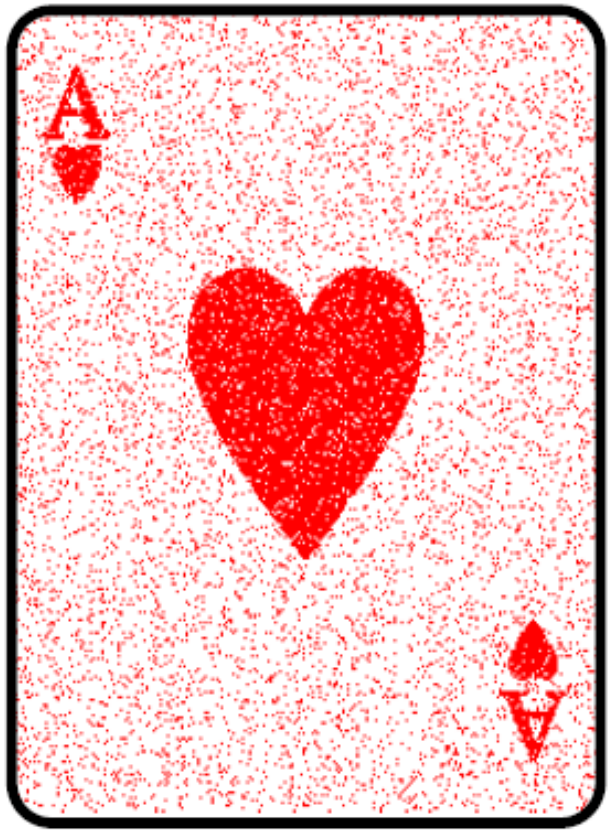
- How to ensure they are spatially smooth?



Source: K. Grauman

Binary Denoising





Before

After

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

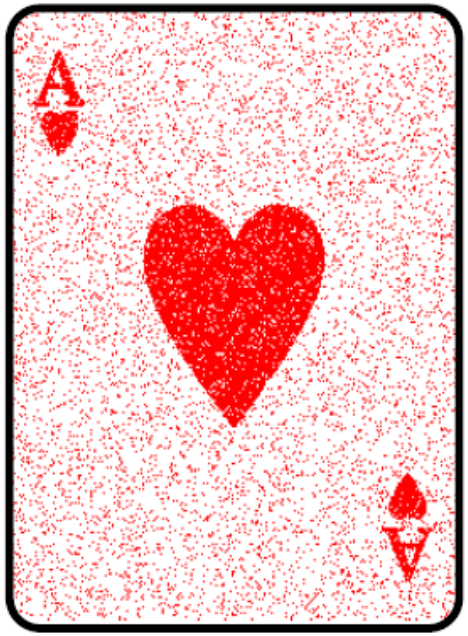
Multi-label Denoising

Before

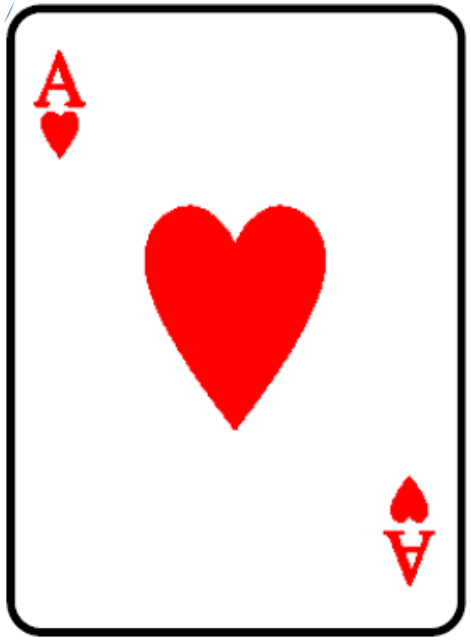
After

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

Denoising goal



Observed Data

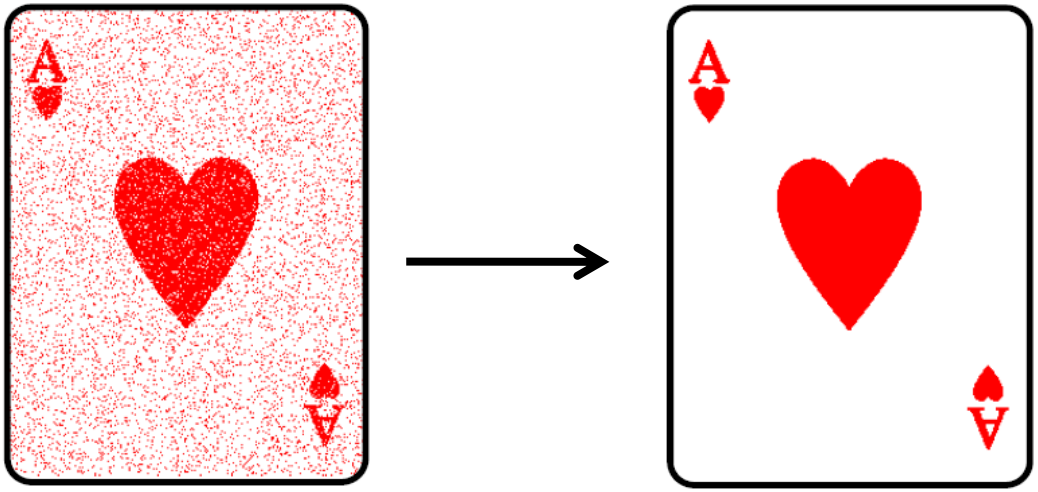


Uncorrupted Image

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

$$\mathbf{w} = \{w_1, w_2 \ldots w_N\}$$

Denoising goal



- Most of the pixels stay the same
- Observed image is not as smooth as original

Now consider pdf over binary images that encourages smoothness – Markov random field

Markov random fields

A Markov random field is formally determined by:

- A set of sites $\mathcal{S} = \{1 \dots N\}$. These will correspond to the N pixel locations.
- A set of random variables $\{w_n\}_{n=1}^{N}$ associated with each of the sites.
- A set of neighbours $\{\mathcal{N}_n\}_{n=1}^{N}$ at each of the N sites.

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

$$Pr(w_n|w_{\mathcal{S}\setminus n}) = Pr(w_n|w_{\mathcal{N}_n}).$$

Markov random fields

$$Pr(\mathbf{w}) = \frac{1}{Z} \prod_{j=1}^{J} \phi_j[\mathbf{w}\mathcal{C}_j]$$

Normalizing constant
(partition function)

Potential function
Returns positive number

Subset of variables
(clique)

Markov random fields

$$Pr(\mathbf{w}) = \frac{1}{Z} \exp\left[-\sum_{j=1}^{J} \psi_j[\mathbf{w}_{\mathcal{C}_j}]\right]$$

Normalizing constant
(partition function)

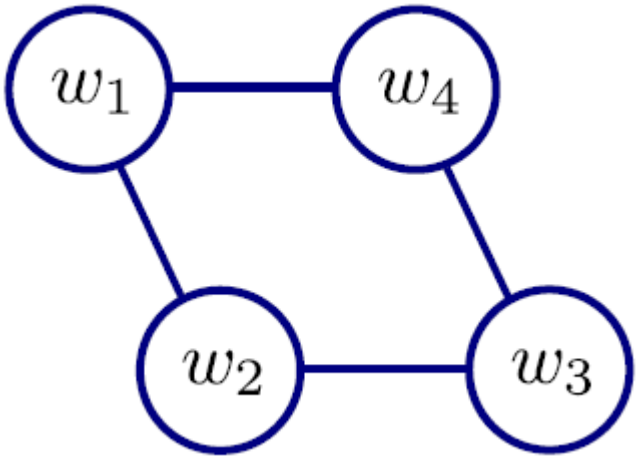
Cost function
Returns any number

Subset of variables
(clique)

Relationship

$$\psi[\bullet] = -\log[\phi[\bullet]]$$

Smoothing Example



$$Pr(\mathbf{w}) = \frac{1}{Z}\phi_{12}(w_1, w_2)\phi_{23}(w_2, w_3)\phi_{34}(w_3, w_4)\phi_{41}(w_4, w_1)$$

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

Smoothing Example

UNIVERSITÄT BONN

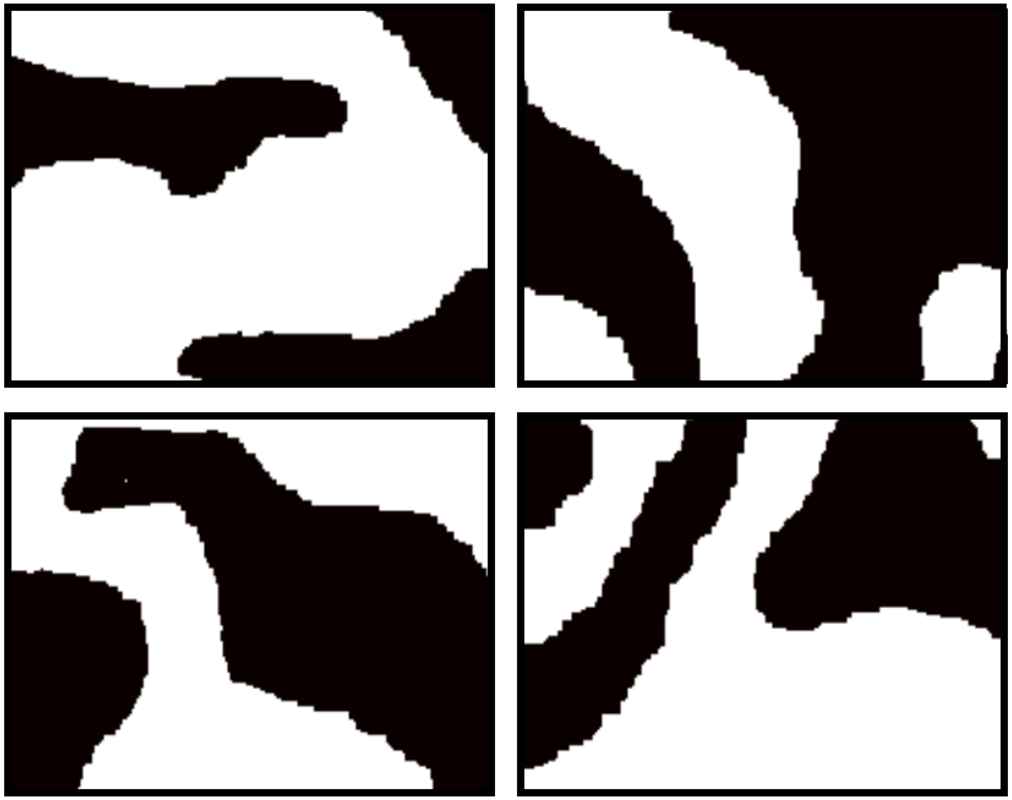
$$Pr(\mathbf{w}) = \frac{1}{Z}\phi_{12}(w_1, w_2)\phi_{23}(w_2, w_3)\phi_{34}(w_3, w_4)\phi_{41}(w_4, w_1)$$

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

$w_{1\ldots4}$	$Pr(w_{1\ldots4})$	$w_{1\ldots4}$	$Pr(w_{1\ldots4})$	$w_{1\ldots4}$	$Pr(w_{1\ldots4})$	$w_{1\ldots4}$	$Pr(w_{1\ldots4})$
0000	0.47176	0100	0.00471	1000	0.00471	1100	0.00471
0001	0.00471	0101	0.00005	1001	0.00471	1101	0.00471
0010	0.00471	0110	0.00471	1010	0.00005	1110	0.00471
0011	0.00471	0111	0.00471	1011	0.00471	1111	0.47176

Smooth solutions (e.g. 0000,1111) have high probability
Z was computed by summing the 16 un-normalized probabilities

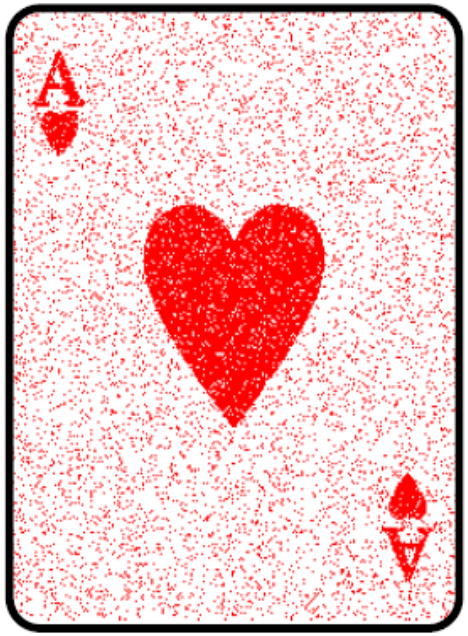
Smoothing Example



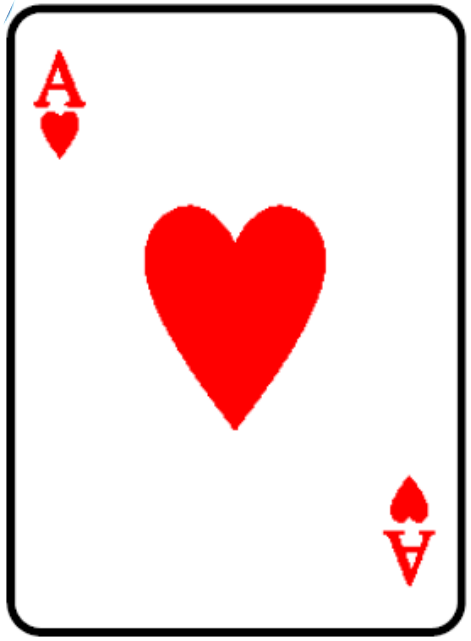
Samples from larger grid -- mostly smooth

Cannot compute partition function Z here - intractable

Denoising goal



Observed Data



Uncorrupted Image

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

$$\mathbf{w} = \{w_1, w_2 \ldots w_N\}$$

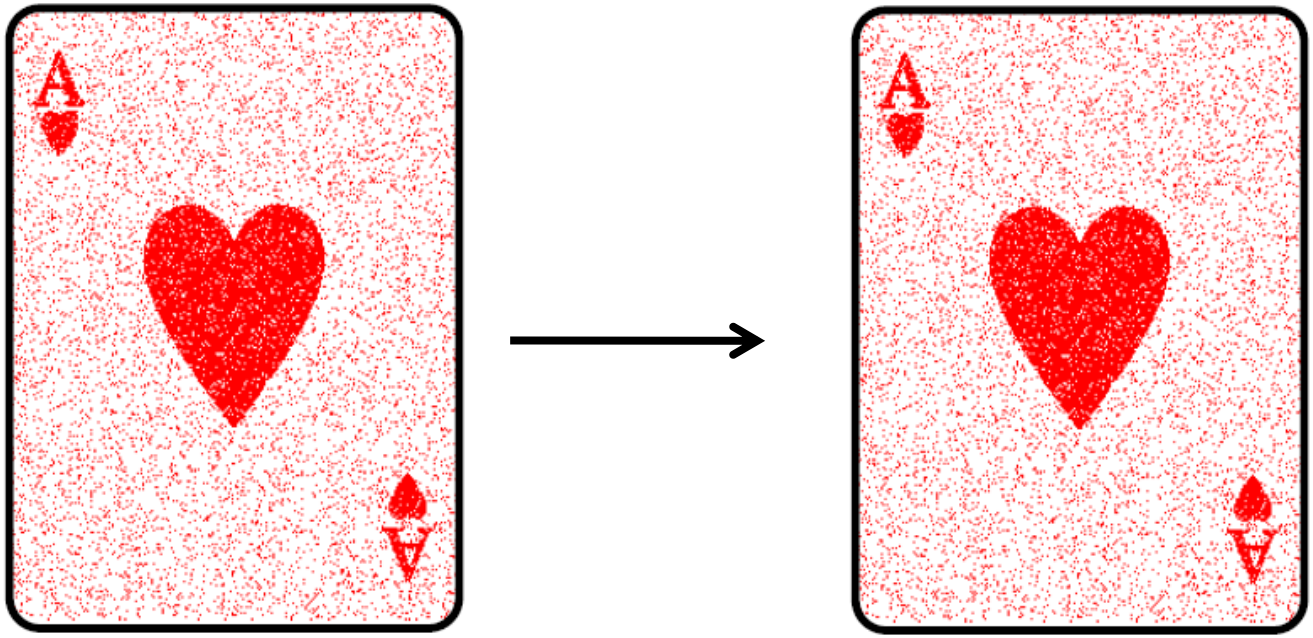
Maximum Likelihood

Fitting: As the name suggests: find the parameters under which the data $\mathbf{x}_{1\ldots I}$ are most likely:

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\left[Pr(\mathbf{x}_{1\ldots I}|\boldsymbol{\theta})\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\left[\prod_{i=1}^{I} Pr(\mathbf{x}_i|\boldsymbol{\theta})\right]\end{aligned}$$

We have assumed that data was independent (hence product)

Denoising goal



Observed Data

Uncorrupted Image

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

$$\mathbf{w} = \{w_1, w_2 \ldots w_N\}$$

Maximum a posteriori (MAP)

UNIVERSITÄT BONN

Fitting

As the name suggests we find the parameters which maximize the posterior probability $Pr(\boldsymbol{\theta}|\mathbf{x}_{1\ldots I})$.

$$
\begin{aligned}
\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[Pr(\boldsymbol{\theta}|\mathbf{x}_{1\ldots I}) \right] \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\frac{Pr(\mathbf{x}_{1\ldots I}|\boldsymbol{\theta}) Pr(\boldsymbol{\theta})}{Pr(\mathbf{x}_{1\ldots I})} \right] \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\frac{\prod_{i=1}^{I} Pr(\mathbf{x}_i|\boldsymbol{\theta}) Pr(\boldsymbol{\theta})}{Pr(\mathbf{x}_{1\ldots I})} \right]
\end{aligned}
$$

Again we have assumed that data was independent

Maximum a posteriori (MAP)

UNIVERSITÄT BONN

Fitting

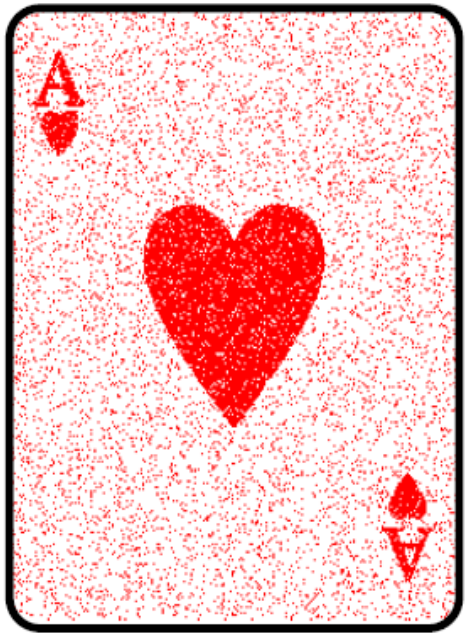
As the name suggests we find the parameters which maximize the posterior probability $Pr(\boldsymbol{\theta}|\mathbf{x}_{1\ldots I})$.

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\frac{\prod_{i=1}^{I} Pr(\mathbf{x}_i|\boldsymbol{\theta}) Pr(\boldsymbol{\theta})}{Pr(\mathbf{x}_{1\ldots I})} \right]$$

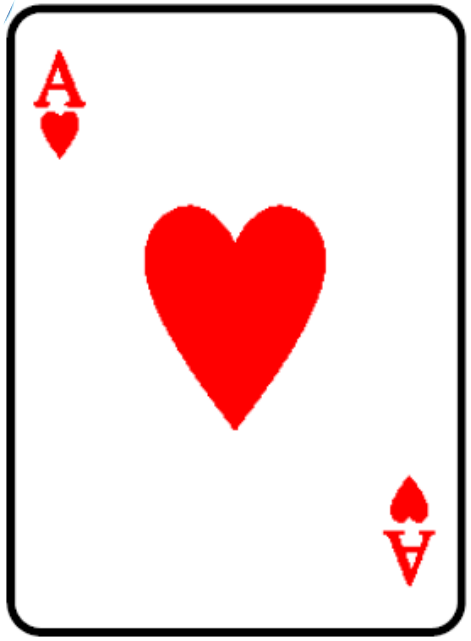
Since the denominator doesn't depend on the parameters we can instead maximize

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\prod_{i=1}^{I} Pr(\mathbf{x}_i|\boldsymbol{\theta}) Pr(\boldsymbol{\theta}) \right]$$

Denoising goal



Observed Data

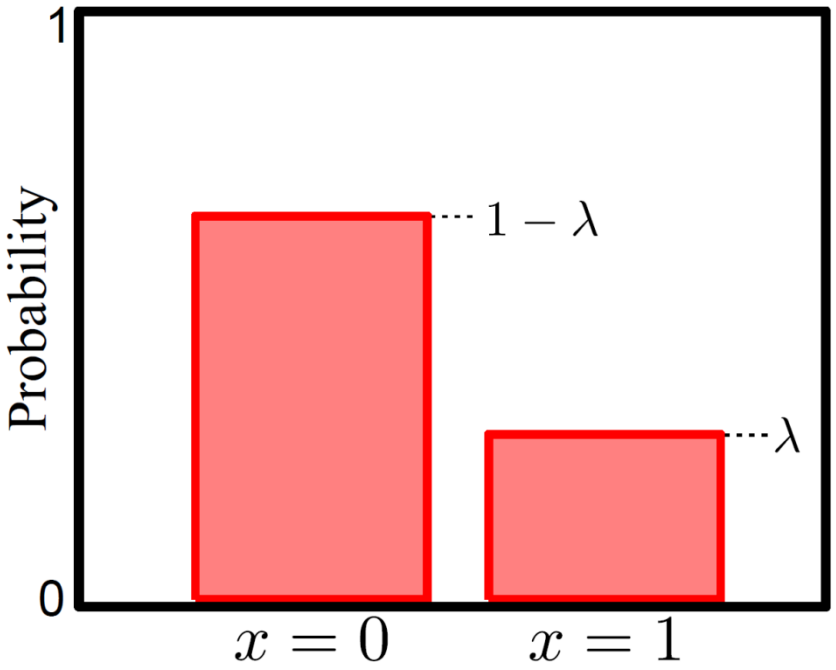


Uncorrupted Image

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

$$\mathbf{w} = \{w_1, w_2 \ldots w_N\}$$

Bernoulli Distribution



$$Pr(x=0) = 1-\lambda$$
$$Pr(x=1) = \lambda.$$

or

$$Pr(x) = \lambda^x(1-\lambda)^{1-x}$$

For short we write:

$$Pr(x) = \text{Bern}_x[\lambda]$$

Bernoulli distribution describes situation where only two possible outcomes y=0/y=1 or failure/success

Takes a single parameter $\lambda \in [0,1]$

Denoising overview

UNIVERSITÄT BONN

Bayes' rule:

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

Likelihoods:

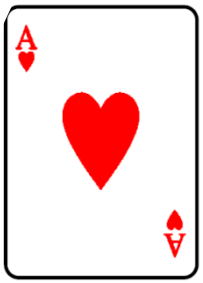
$$\begin{aligned} Pr(x_n|w_n = 0) &= \text{Bern}_{x_n}[\rho] \\ Pr(x_n|w_n = 1) &= \text{Bern}_{x_n}[1-\rho] \end{aligned}$$

Probability of flipping polarity

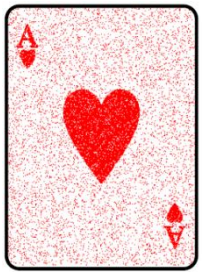
Prior: Markov random field (smoothness)

MAP Inference: Graph cuts

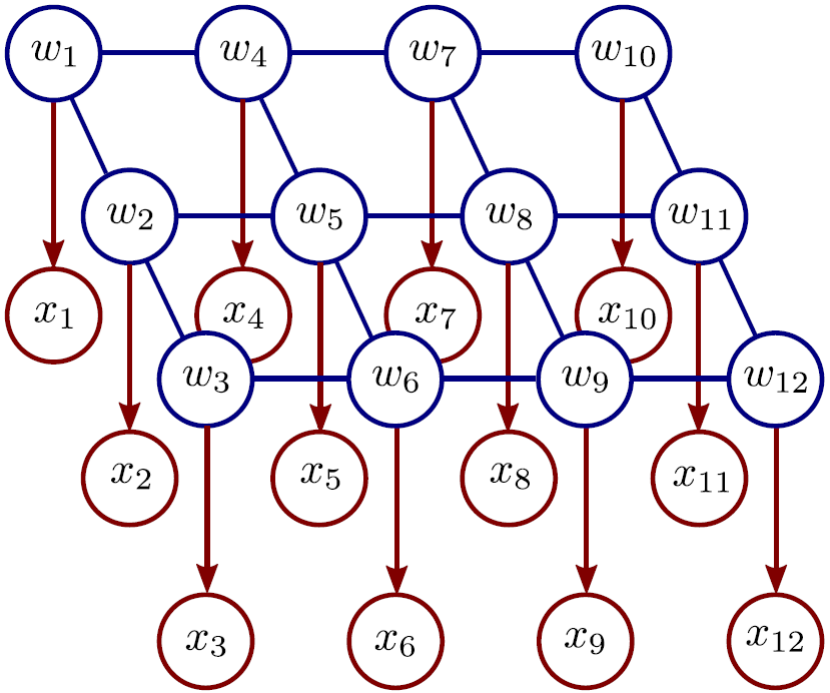
Denoising with MRFs



Original image, **w**



Observed image, **x**



MRF Prior (pairwise cliques)

$$Pr(w_{1\ldots N}) = \frac{1}{Z}\exp\left[-\sum_{(m,n)\in\mathcal{C}}\psi[w_m, w_n, \boldsymbol{\theta}]\right]$$

Likelihoods

$$Pr(x_n|w_n = 0) = \text{Bern}_{x_n}[\rho]$$
$$Pr(x_n|w_n = 1) = \text{Bern}_{x_n}[1-\rho]$$

Inference : $$Pr(w_{1\ldots N}|x_{1\ldots N}) = \frac{\prod_{n=1}^{N} Pr(x_n|w_n)Pr(w_{1\ldots N})}{Pr(x_{1\ldots N})}$$

MAP inference

$$
\begin{aligned}
\hat{w}_{1\ldots N} &= \underset{w_{1\ldots N}}{\operatorname{argmax}} \left[Pr(w_{1\ldots N} | \mathbf{x}_{1\ldots N}) \right] \\
&= \underset{w_{1\ldots N}}{\operatorname{argmax}} \left[\prod_{n=1}^{N} Pr(x_n | w_n) Pr(w_{1\ldots N}) \right] \\
&= \underset{w_{1\ldots N}}{\operatorname{argmax}} \left[\sum_{n=1}^{N} \log[Pr(x_n | w_n)] + \log[Pr(w_{1\ldots N})] \right] \\
&= \underset{w_{1\ldots N}}{\operatorname{argmax}} \left[\sum_{n=1}^{N} \log[Pr(x_n | w_n)] - \sum_{(m,n)\in\mathcal{C}} \psi[w_m, w_n, \boldsymbol{\theta}] \right] \\
&= \underset{w_{1\ldots N}}{\operatorname{argmin}} \left[\sum_{n=1}^{N} -\log[Pr(x_n | w_n)] + \sum_{(m,n)\in\mathcal{C}} \psi[w_m, w_n, \boldsymbol{\theta}] \right] \\
&= \underset{w_{1\ldots N}}{\operatorname{argmin}} \left[\sum_{n=1}^{N} U_n(w_n) + \sum_{(m,n)\in\mathcal{C}} P_{mn}(w_m, w_n) \right],
\end{aligned}
$$

$$
Pr(w_{1\ldots N}) = \frac{1}{Z} \exp\left[- \sum_{(m,n)\in\mathcal{C}} \psi[w_m, w_n, \boldsymbol{\theta}] \right]
$$

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:

$$\underset{w_{1\ldots N}}{\operatorname{argmin}} \left[\sum_{n=1}^{N} U_n(w_n) + \sum_{(m,n)\in\mathcal{C}} P_{mn}(w_m, w_n) \right]$$

Unary terms
(compatability of data with label y)

Pairwise terms
(compatability of neighboring labels)

Three main cases:

- binary MRFs (i.e. $w_i \in \{0, 1\}$) where the costs for different combinations of adjacent labels are "submodular". Exact MAP inference is tractable here.
- multi-label MRFs (i.e. $w_i \in \{1, 2 \ldots, K\}$) 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:

$$\underset{w_{1\ldots N}}{\operatorname{argmin}} \left[\sum_{n=1}^{N} U_n(w_n) + \sum_{(m,n)\in\mathcal{C}} P_{mn}(w_m, w_n) \right]$$

Unary terms
(compatability of data with label y)

Pairwise terms
(compatability of neighboring labels)

Approach:

Convert minimization into the form of a standard CS problem,

MAXIMUM FLOW or MINIMUM CUT ON A GRAPH

Polynomial-time methods for solving this problem are known

Solvers

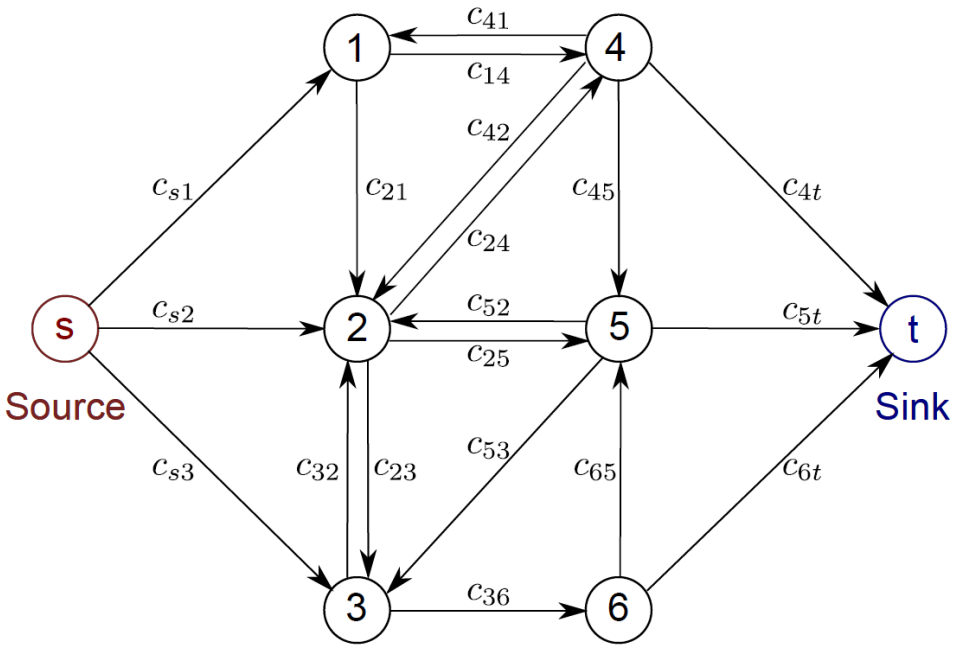
UNIVERSITÄT BONN

- OpenCV: grabCut uses GCGraph
- http://pub.ist.ac.at/~vnk/software.html
- http://vision.csd.uwo.ca/code/

An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Computer Vision. Yuri Boykov and Vladimir Kolmogorov. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004.

Fast approximate energy minimization via graph cuts. Yuri Boykov, Olga Veksler, Ramin Zabih. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001.

Max-Flow Problem

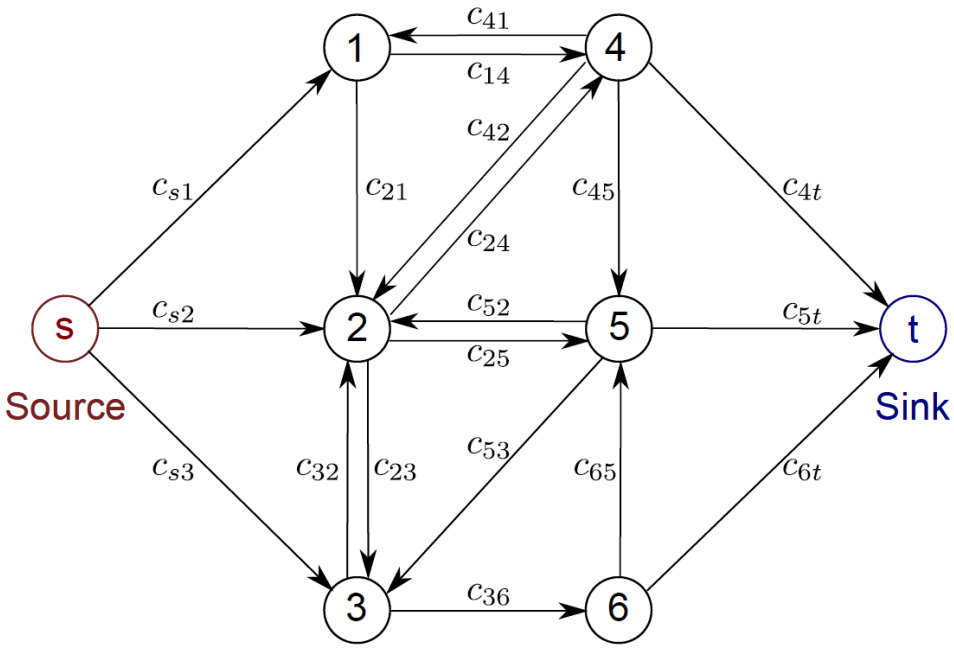


Goal:

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.

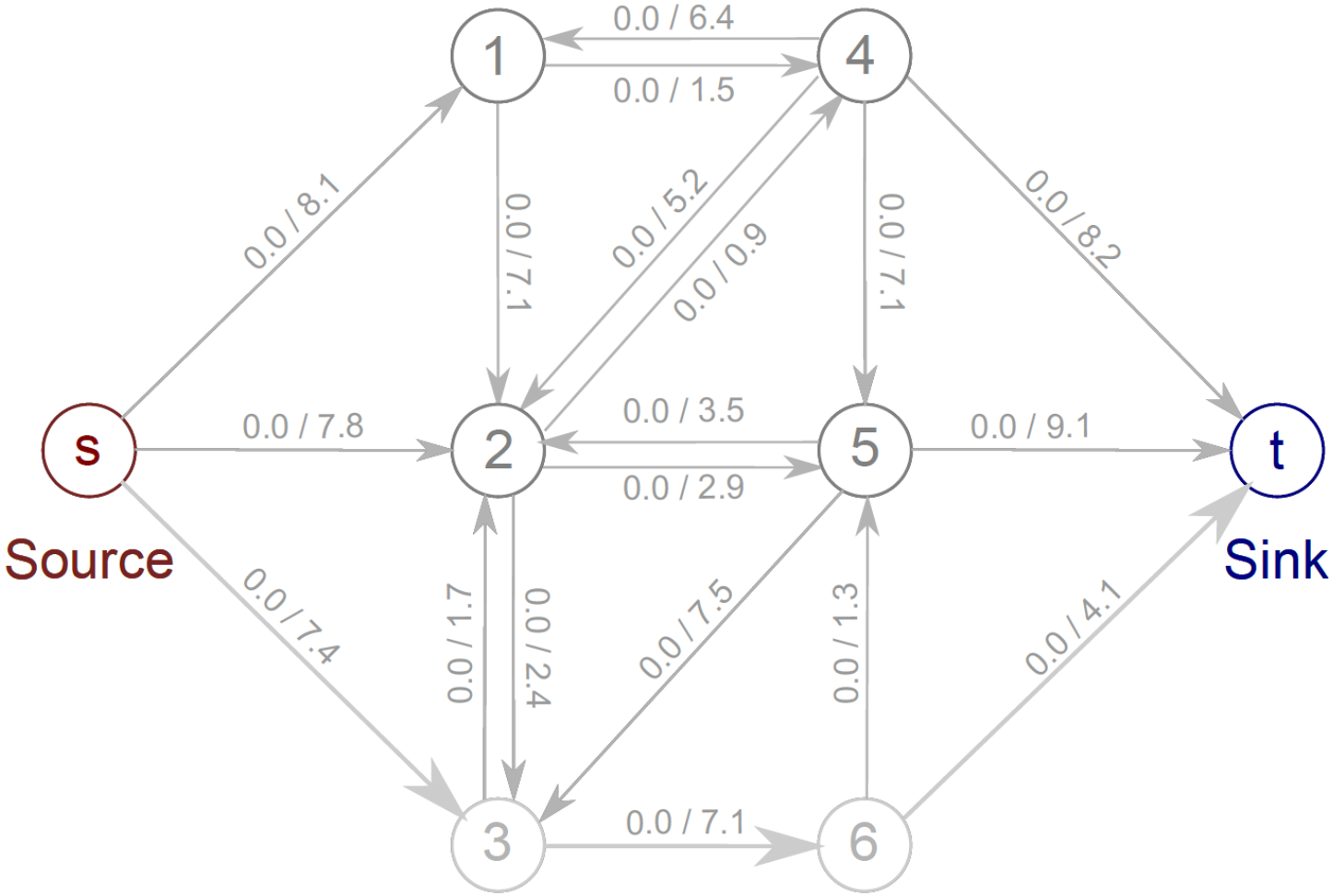
Saturated Edges



When we are pushing the maximum amount of flow:

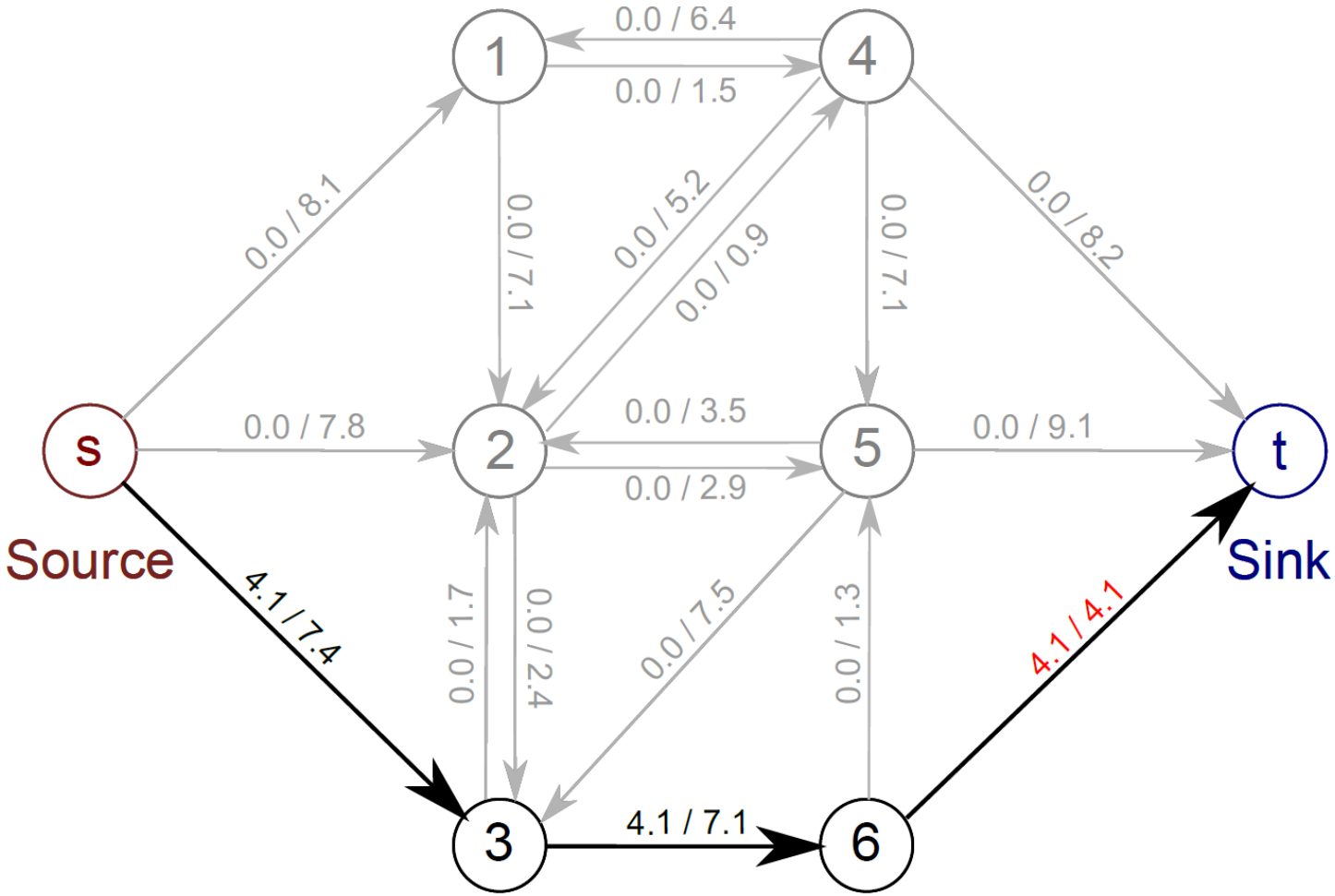
- There must be at least one saturated edge on any path from source to sink (otherwise we could push more flow)
- The set of saturated edges hence separate the source and sink

Augmenting Paths



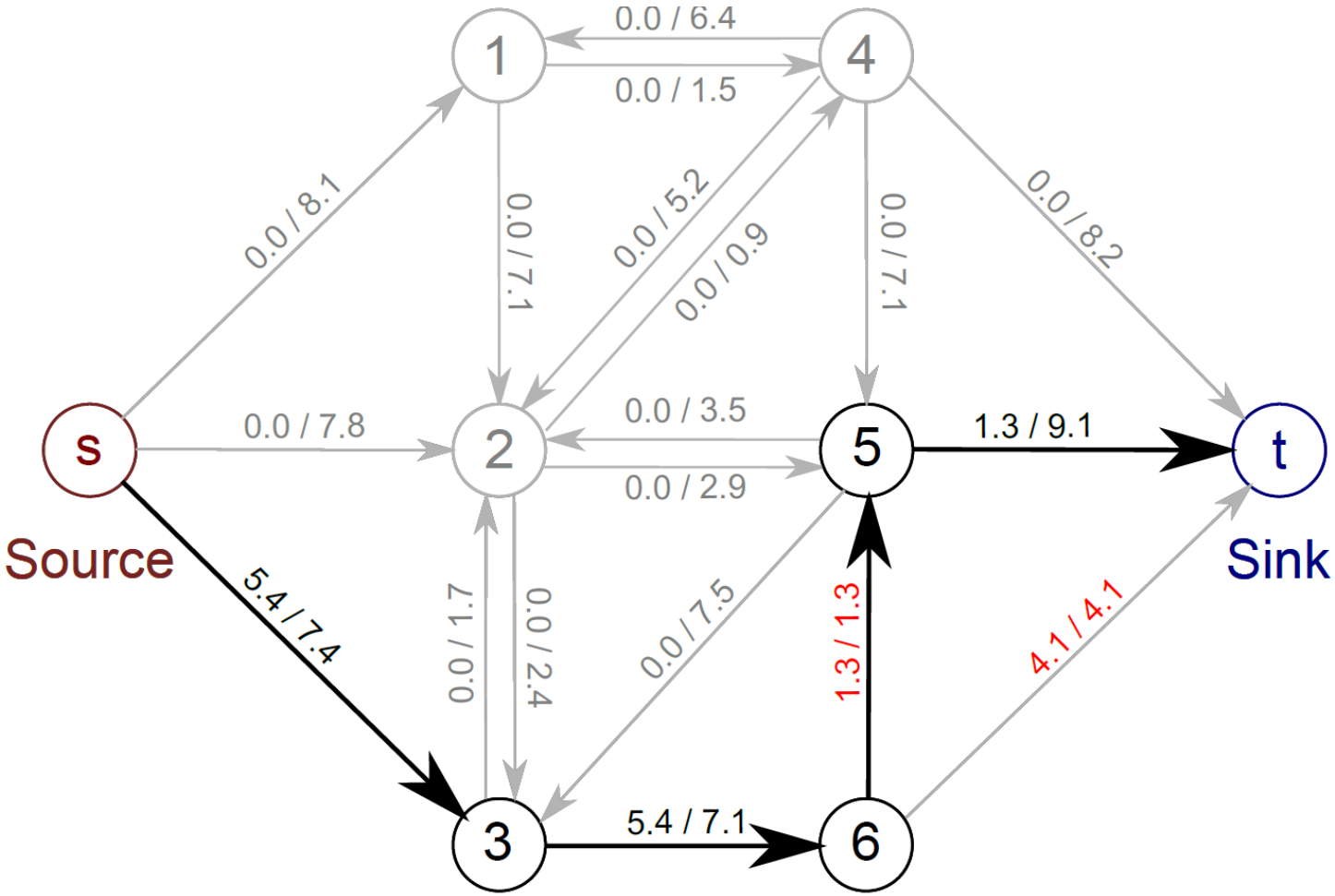
Two numbers represent: current flow / total capacity

Augmenting Paths



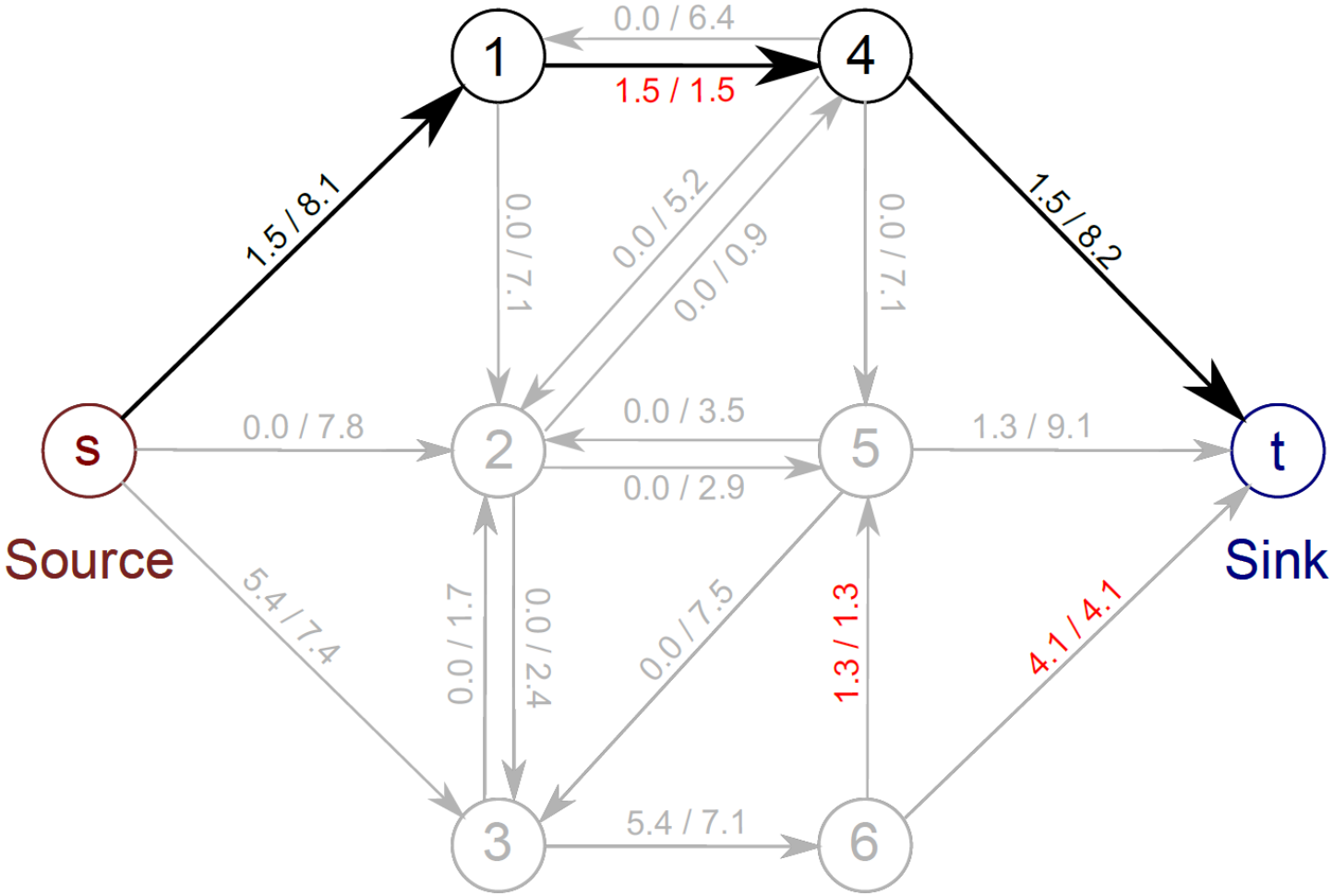
Choose any route from source to sink with spare capacity, and push as much flow as you can. One edge (here 6-t) will saturate.

Augmenting Paths



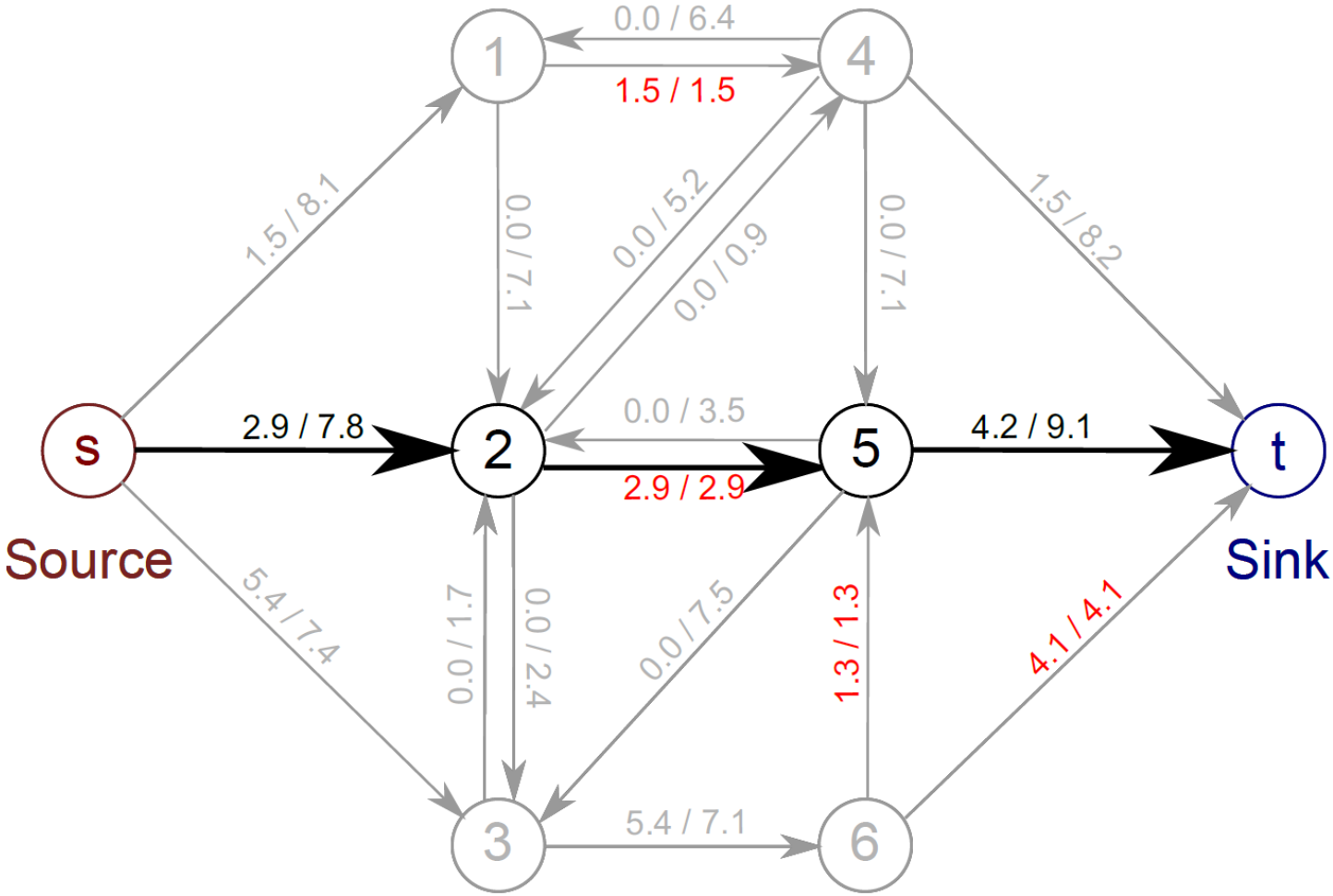
Choose another route, respecting remaining capacity. This time edge 6-5 saturates.

Augmenting Paths



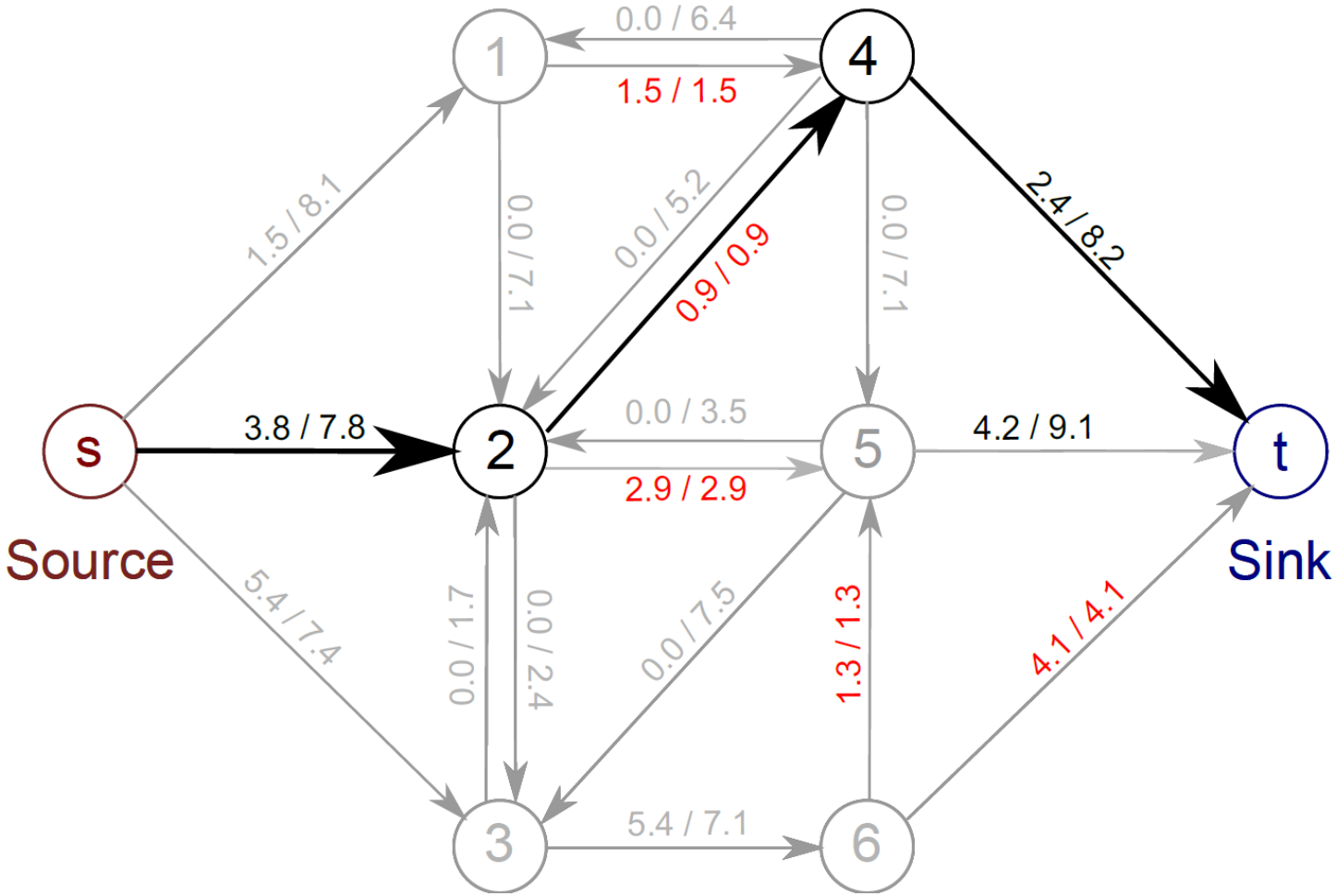
A third route. Edge 1-4 saturates

Augmenting Paths



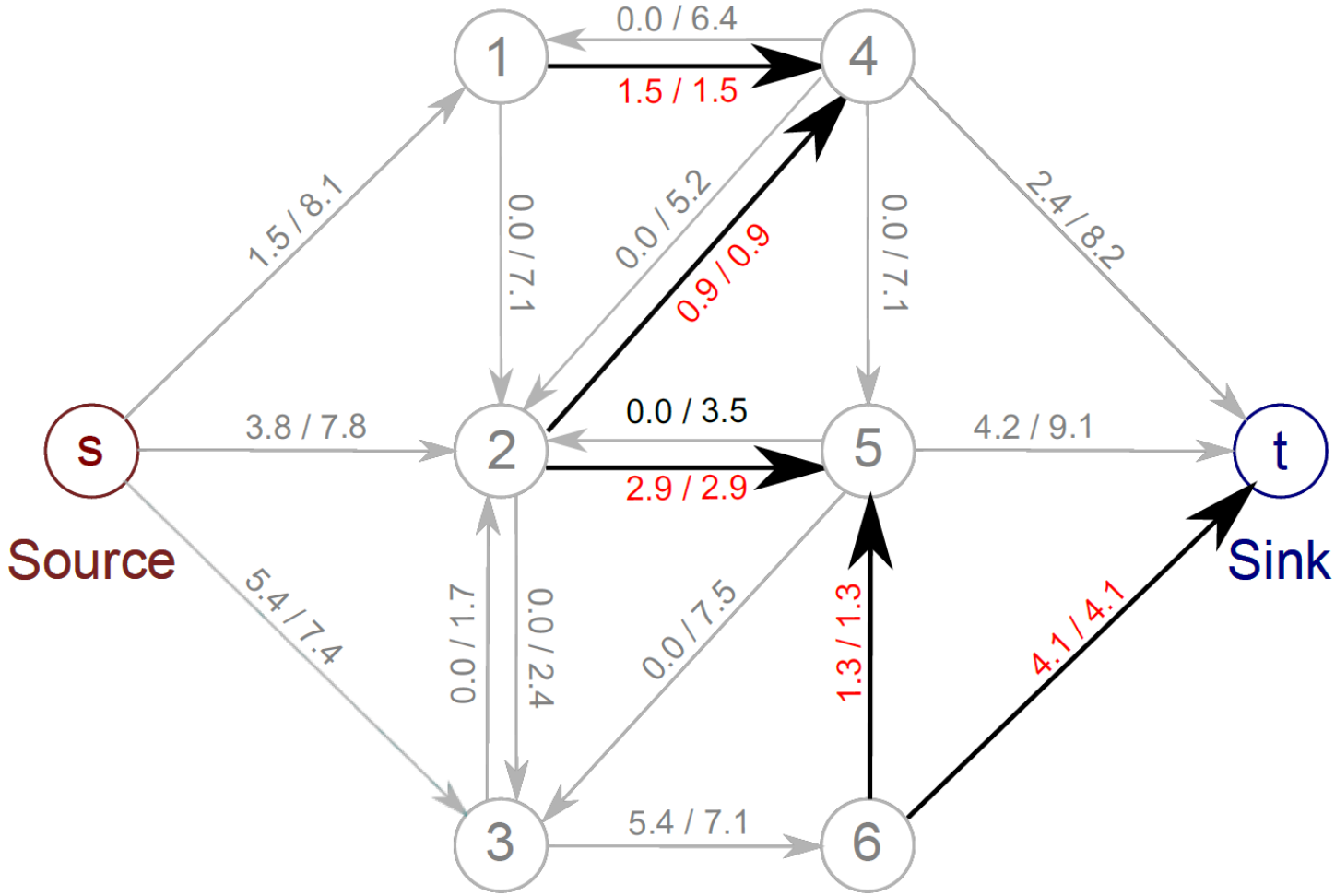
A fourth route. Edge 2-5 saturates

Augmenting Paths



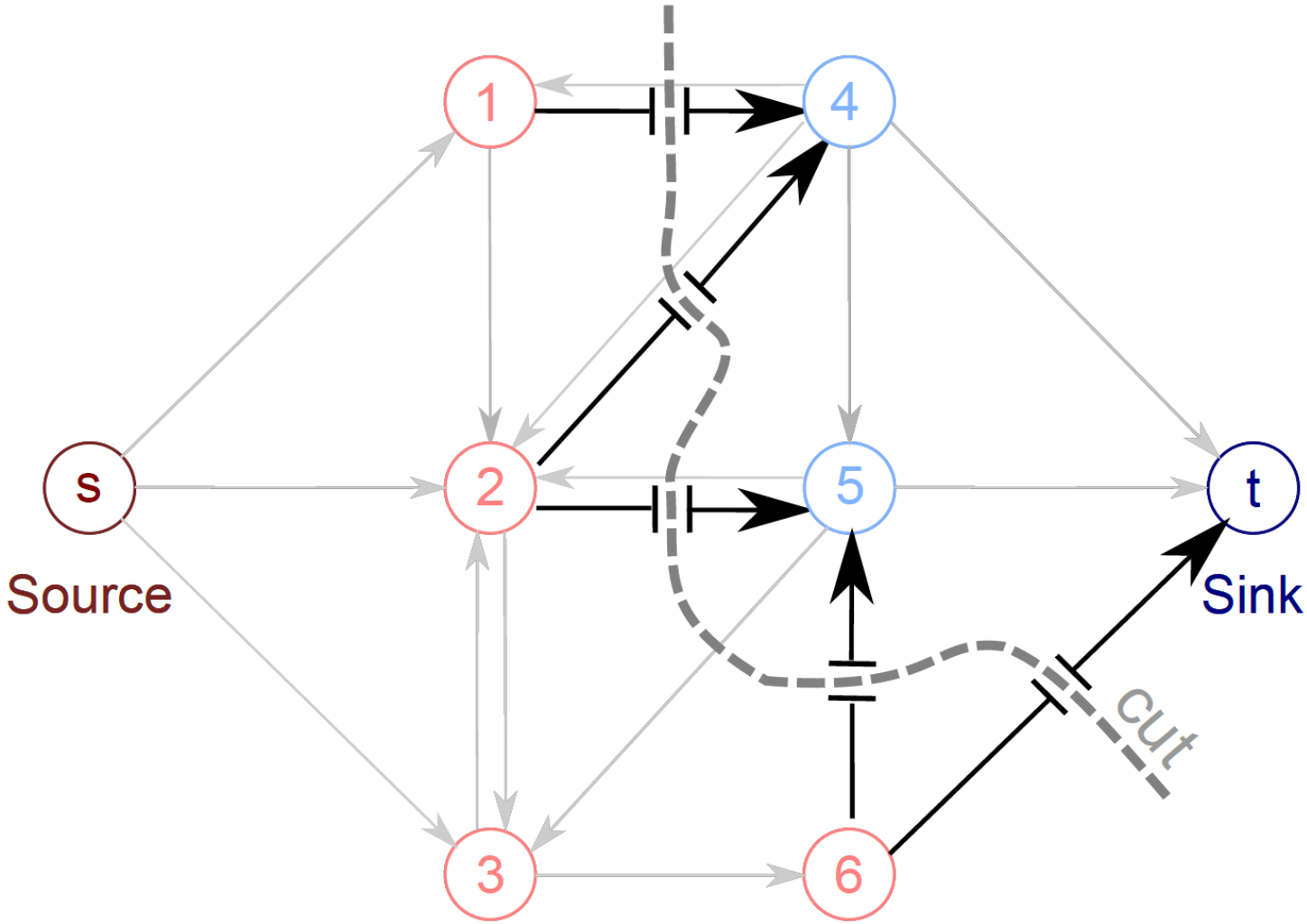
A fifth route. Edge 2-4 saturates

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.

Graph Cuts: Binary MRF

UNIVERSITÄT BONN

Graph cuts used to optimise this cost function:

$$\arg\min_{w_{1\ldots N}} \sum_{n=1}^{N} U_n(w_n) + \sum_{(m,n)\in\mathcal{C}} P_{mn}(w_m, w_n),$$

Unary terms
(compatability of data with label w)

Pairwise terms
(compatability of neighboring labels)

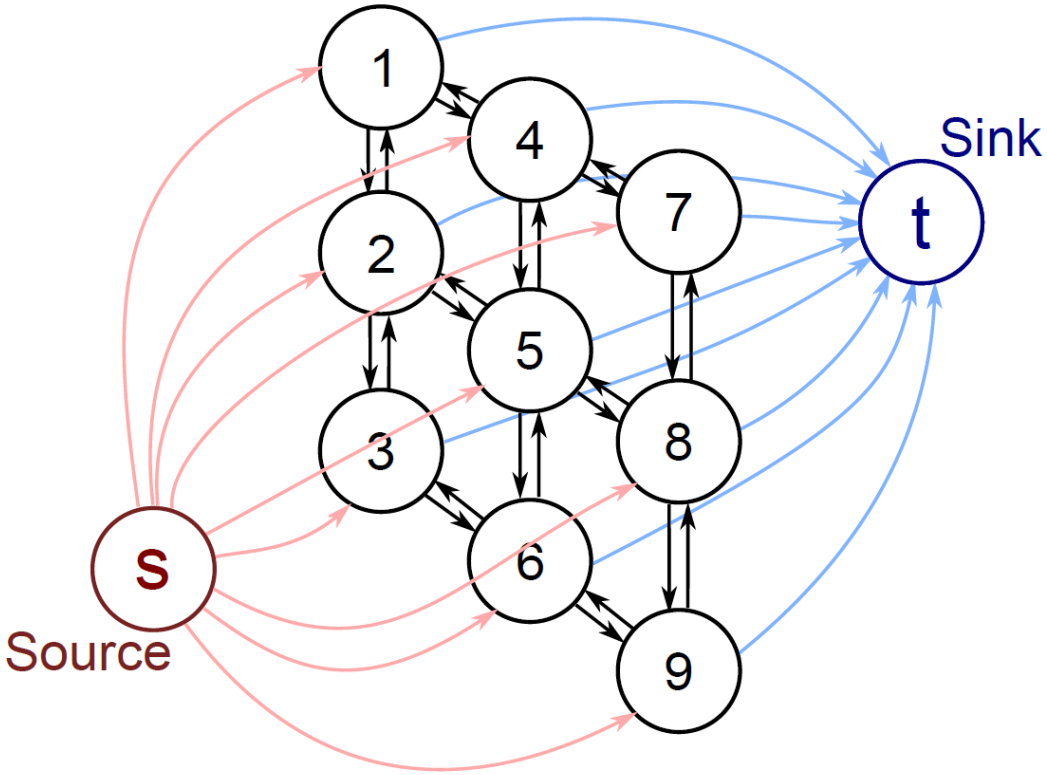
First work with binary case (i.e. True label w is 0 or 1)

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

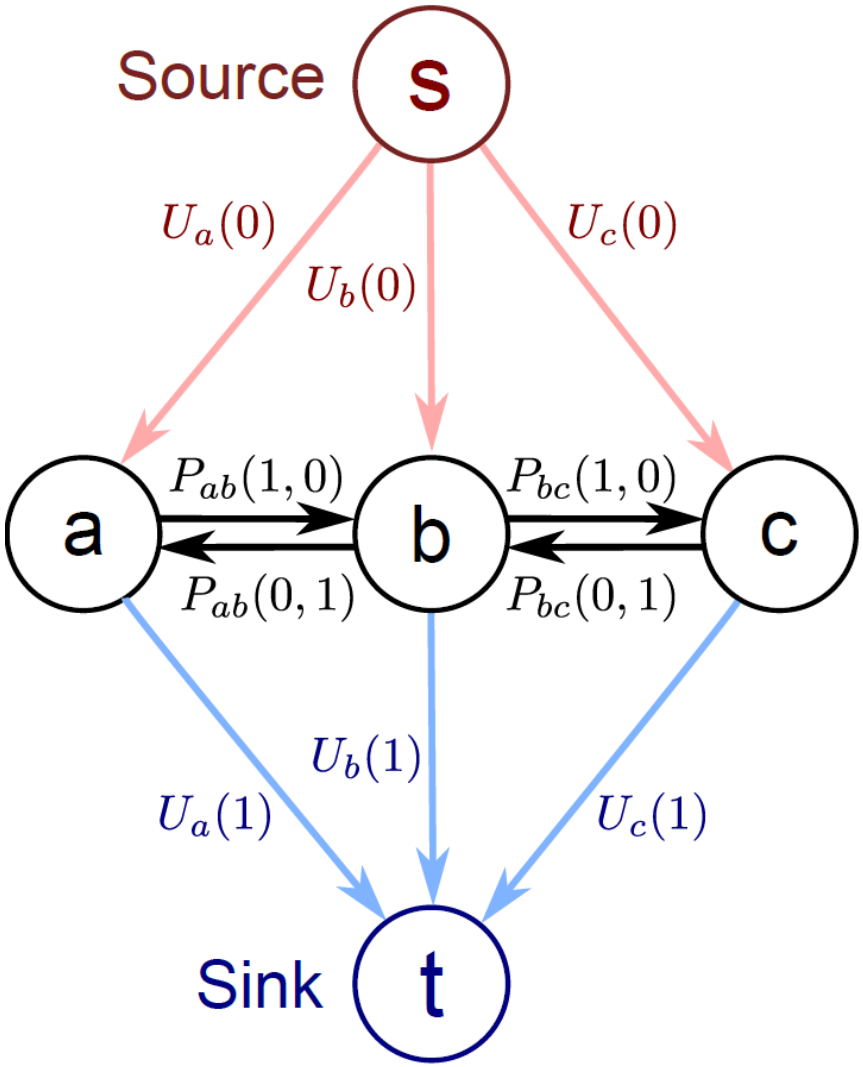


Note that in the minimum cut EITHER the edge connecting to the source will be cut, OR the edge connecting to the sink, but NOT BOTH (unnecessary).

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

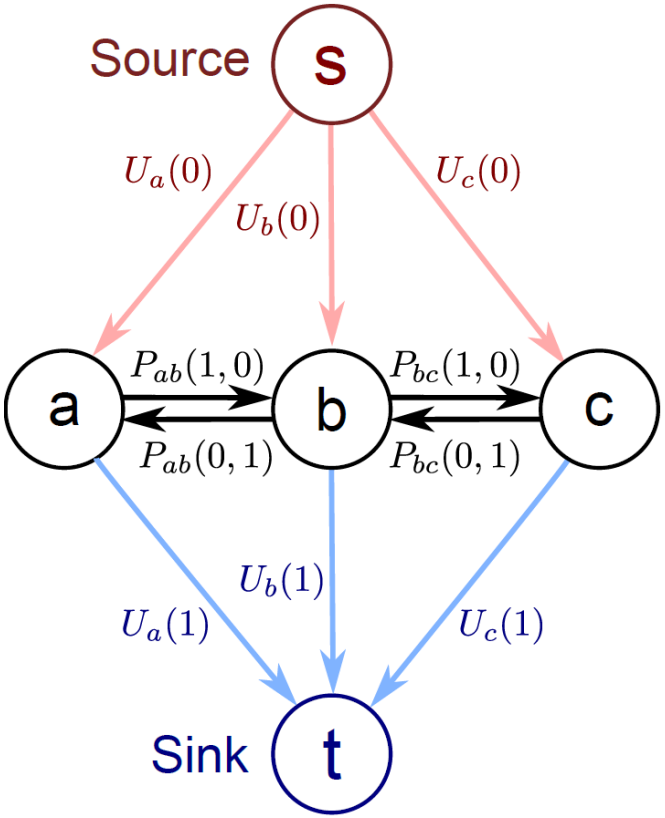
Unary costs U(0), U(1) attached to links to source and sink.

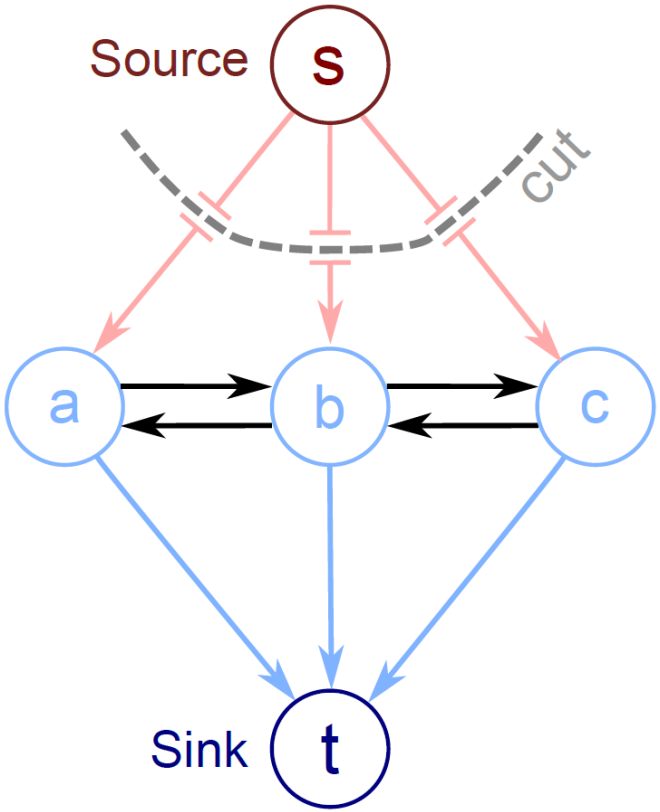
- Either one or the other is paid.

Pairwise costs between pixel nodes as shown.

- Why? Easiest to understand with some worked examples.

Example 1





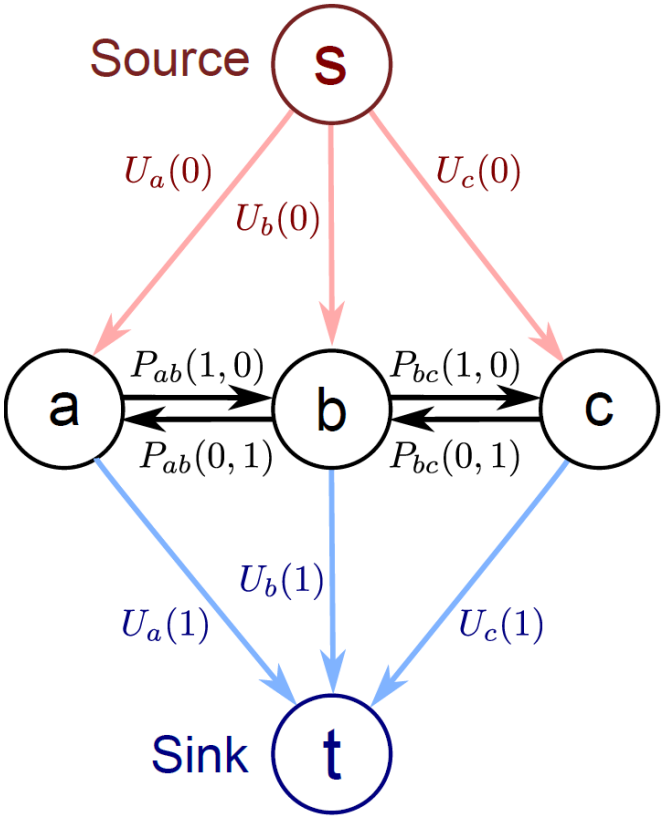
Solution

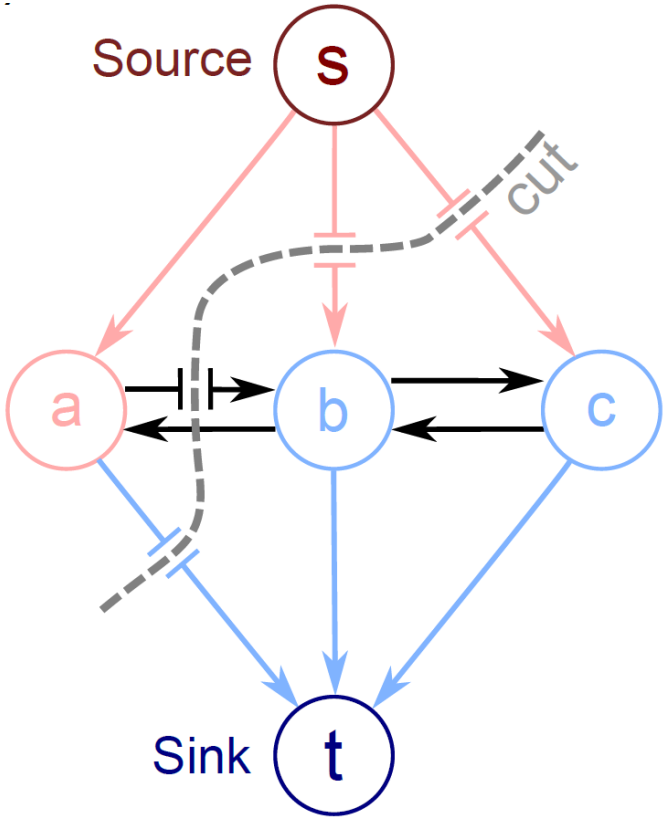
0	0	0

Cost

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

Example 2





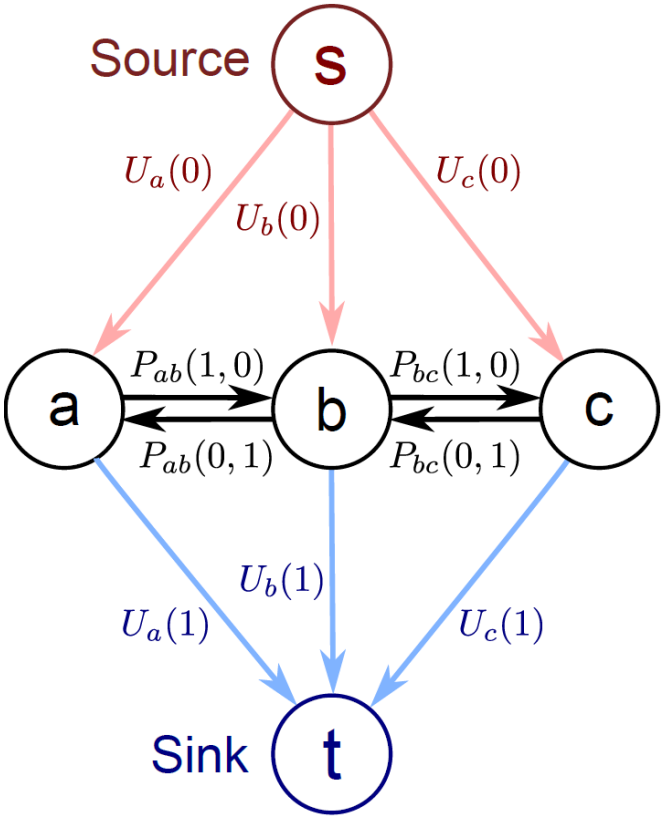
Solution

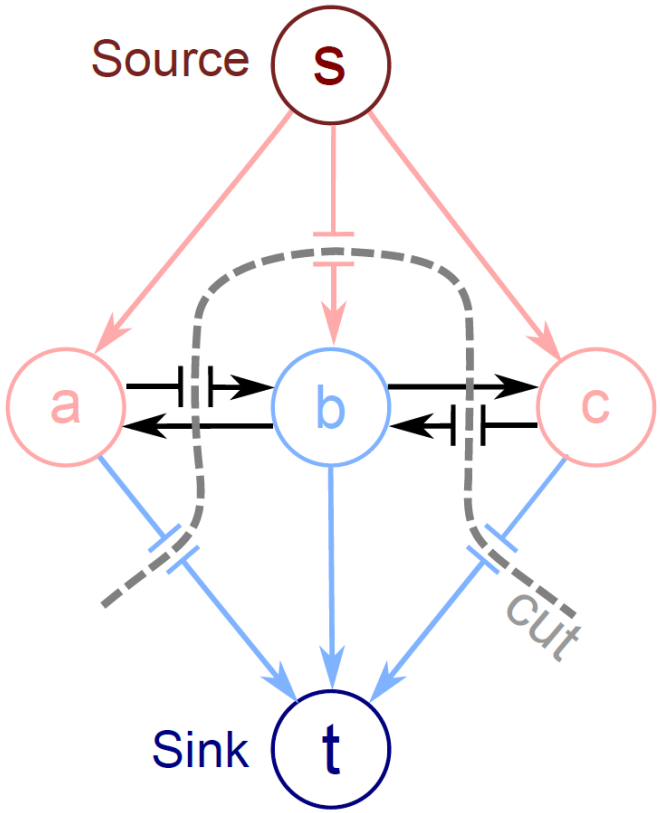
1	0	0

Cost

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

Example 3



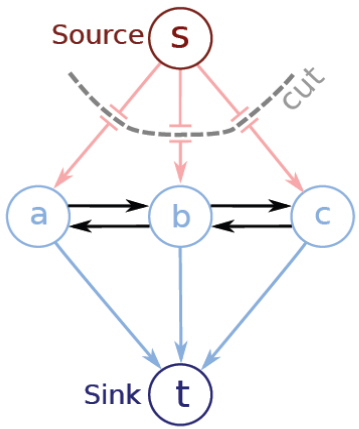


Solution

1	0	1

Cost

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

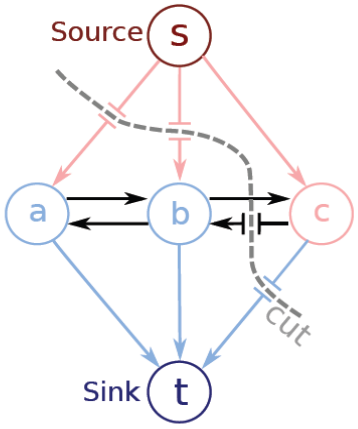


Solution

0	0	0

Cost

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

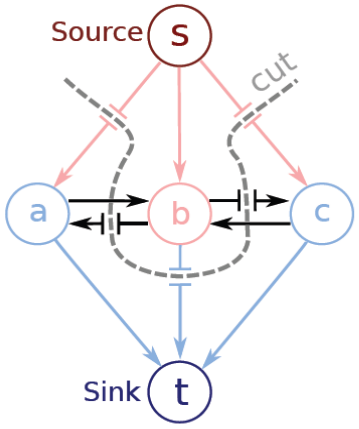


Solution

0	0	1

Cost

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

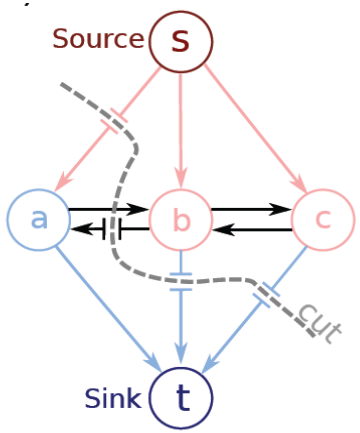


Solution

0	1	0

Cost

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

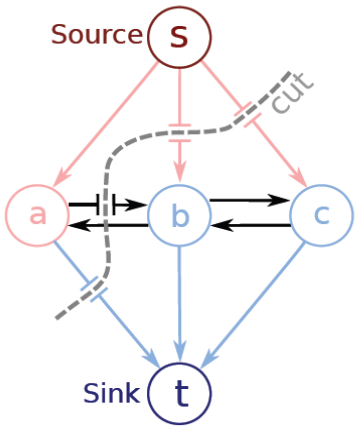


Solution

0	1	1

Cost

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

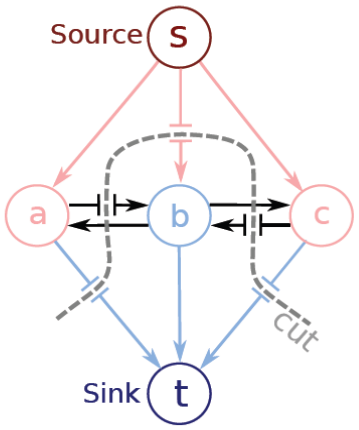


Solution

1	0	0

Cost

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

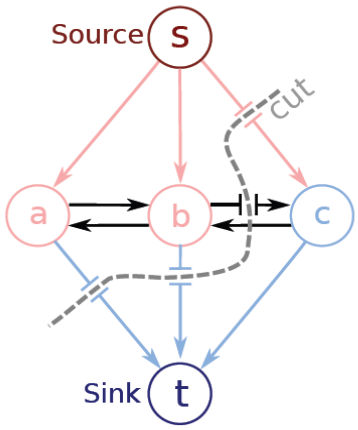


Solution

1	0	1

Cost

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

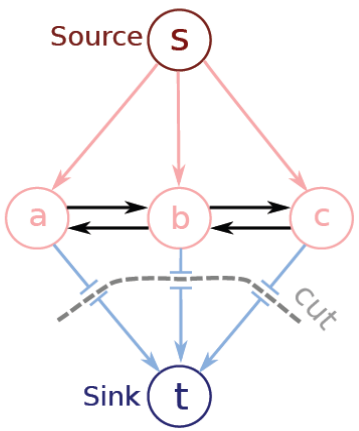


Solution

1	1	0

Cost

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



Solution

1	1	1

Cost

$$U_a(1) + U_b(1) + U_c(1)$$

Graph Cuts: Binary MRF

Graph cuts used to optimise this cost function:

$$\arg\min_{w_{1\ldots N}} \sum_{n=1}^{N} U_n(w_n) + \sum_{(m,n)\in\mathcal{C}} P_{mn}(w_m, w_n),$$

Unary terms (compatability of data with label w)

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

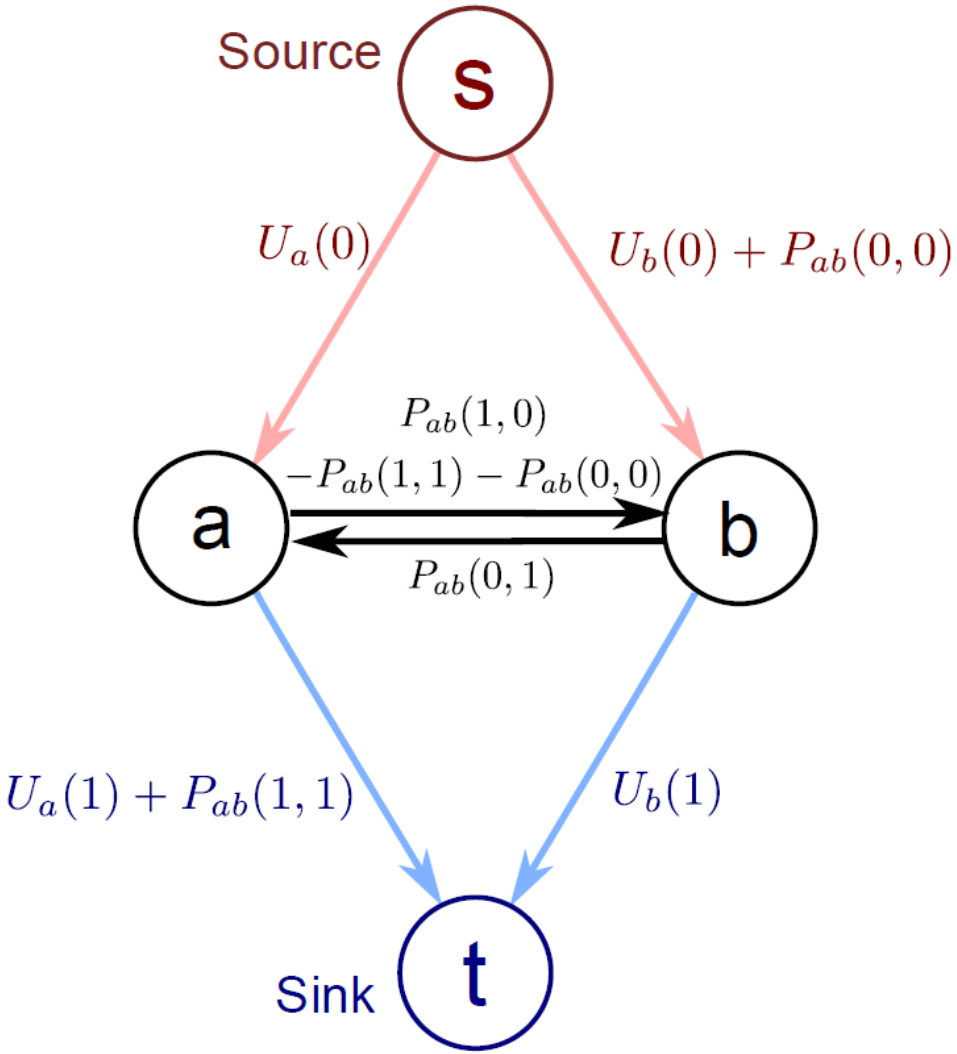
General Pairwise costs

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

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

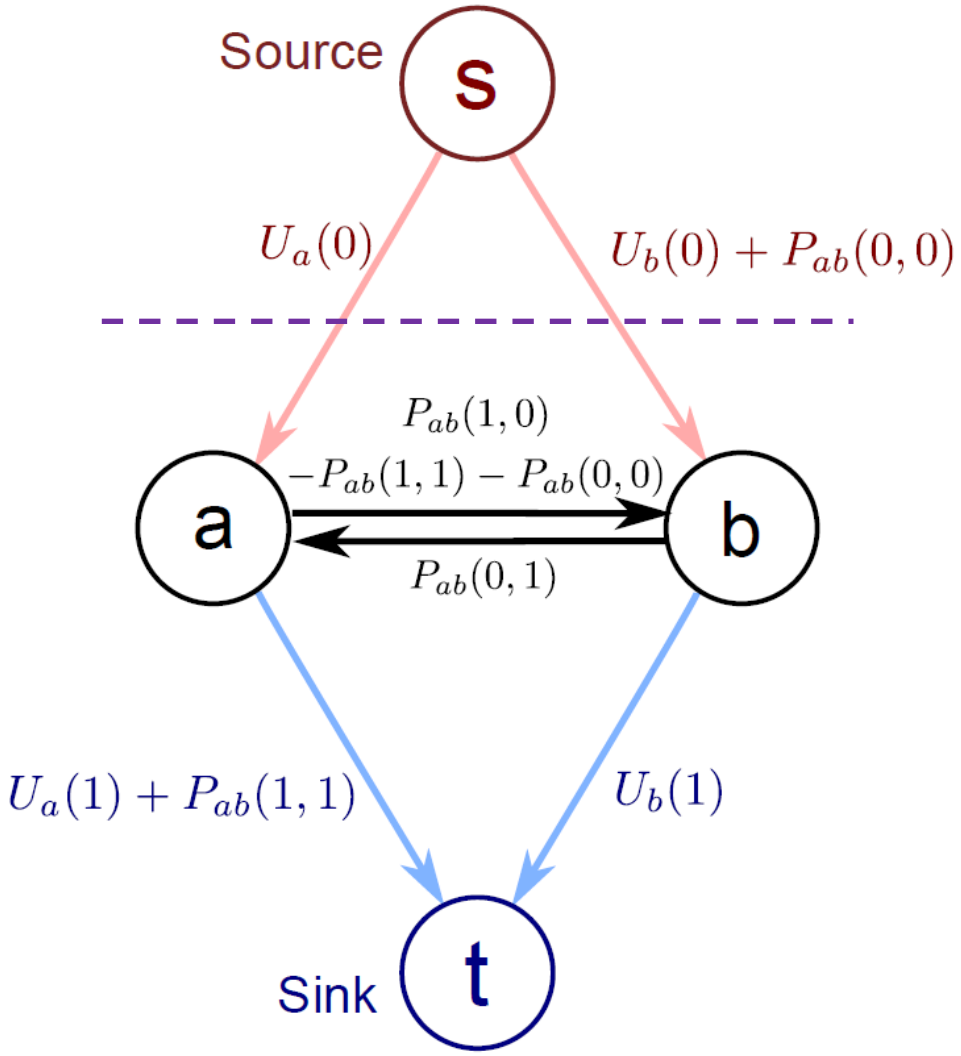
Similar approach for P(1,1)



General Pairwise costs

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

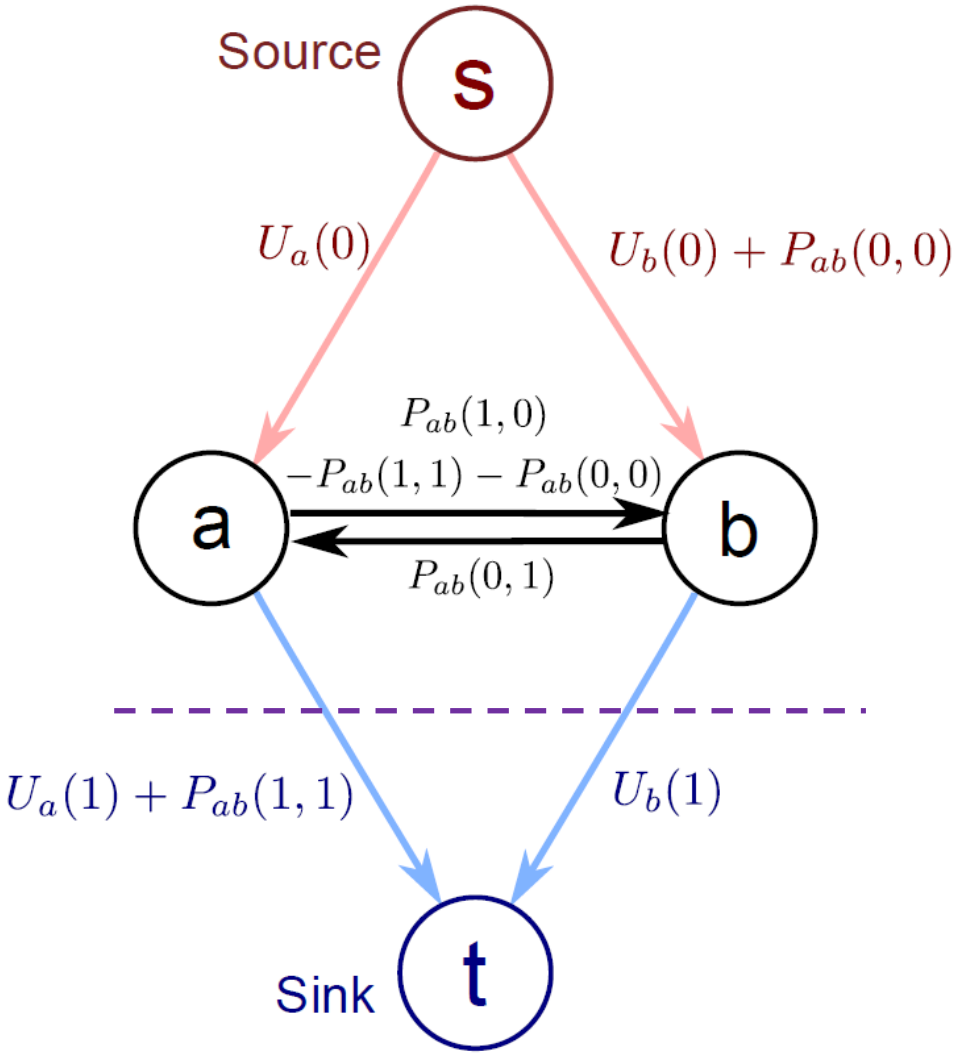
Label: 0 0
Cost: U(0)+P(0,0)+U(0)



General Pairwise costs

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

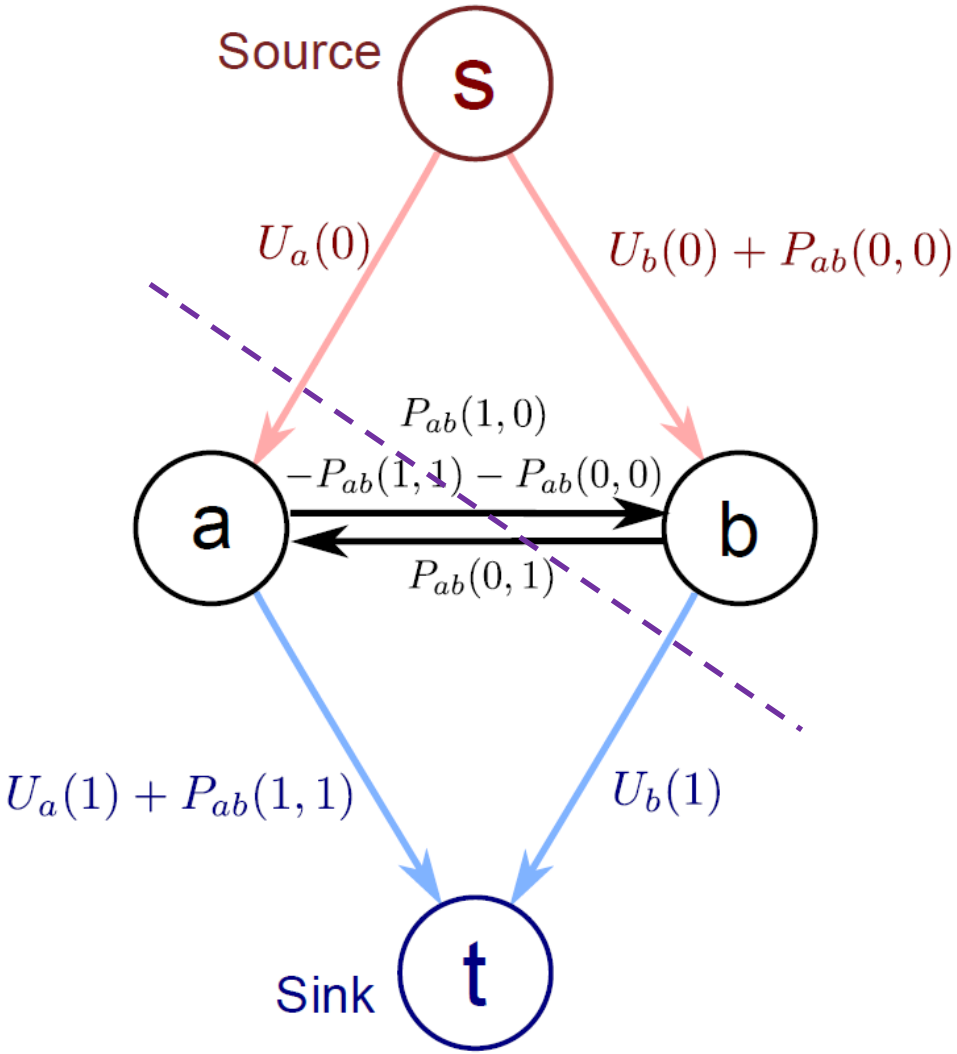
Label: 1 1
Cost: U(1)+P(1,1)+U(1)



General Pairwise costs

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

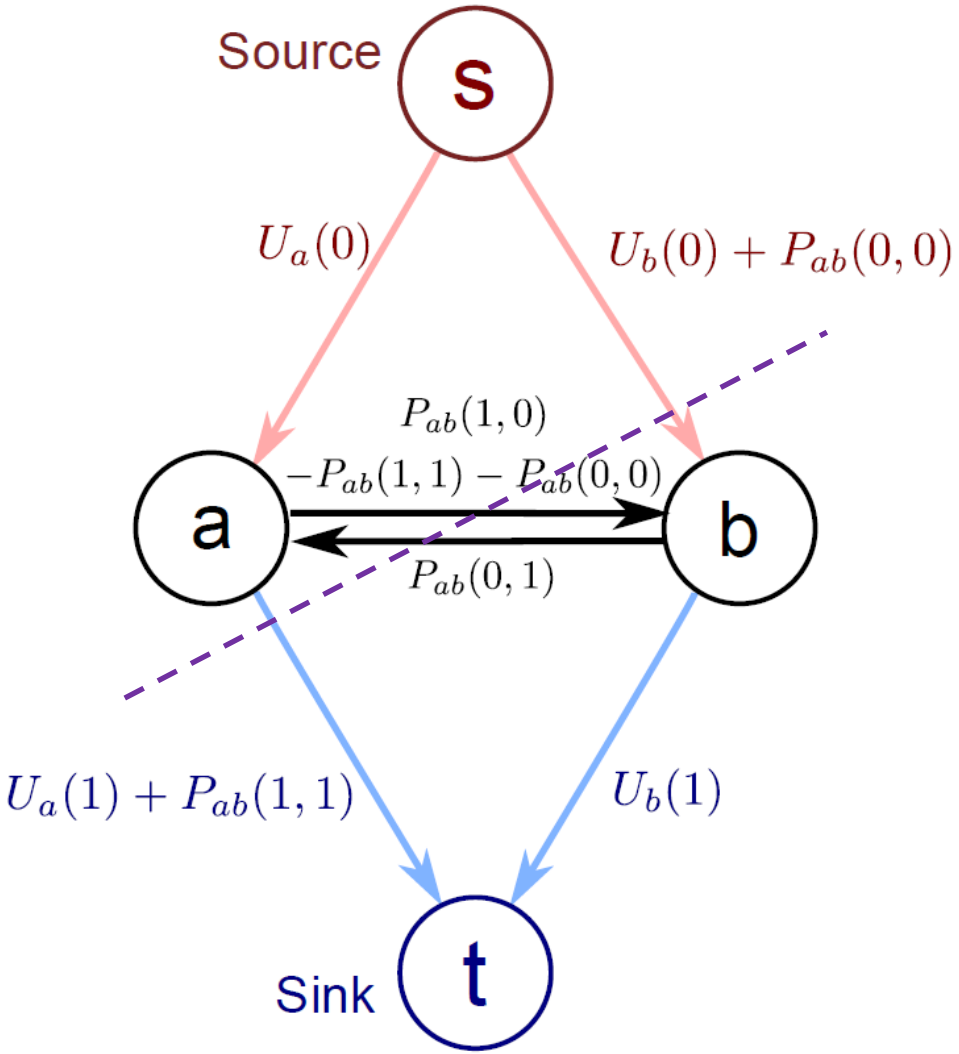
Label: 0 1
Cost: U(0)+P(0,1)+U(1)



General Pairwise costs

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

Label: 1 0
Cost:
(U(1)+P(1,1))
+ (P(1,0)-P(1,1)-P(0,0))
+ (U(0) + P(0,0))
= U(1) + P(1,0) + U(0)

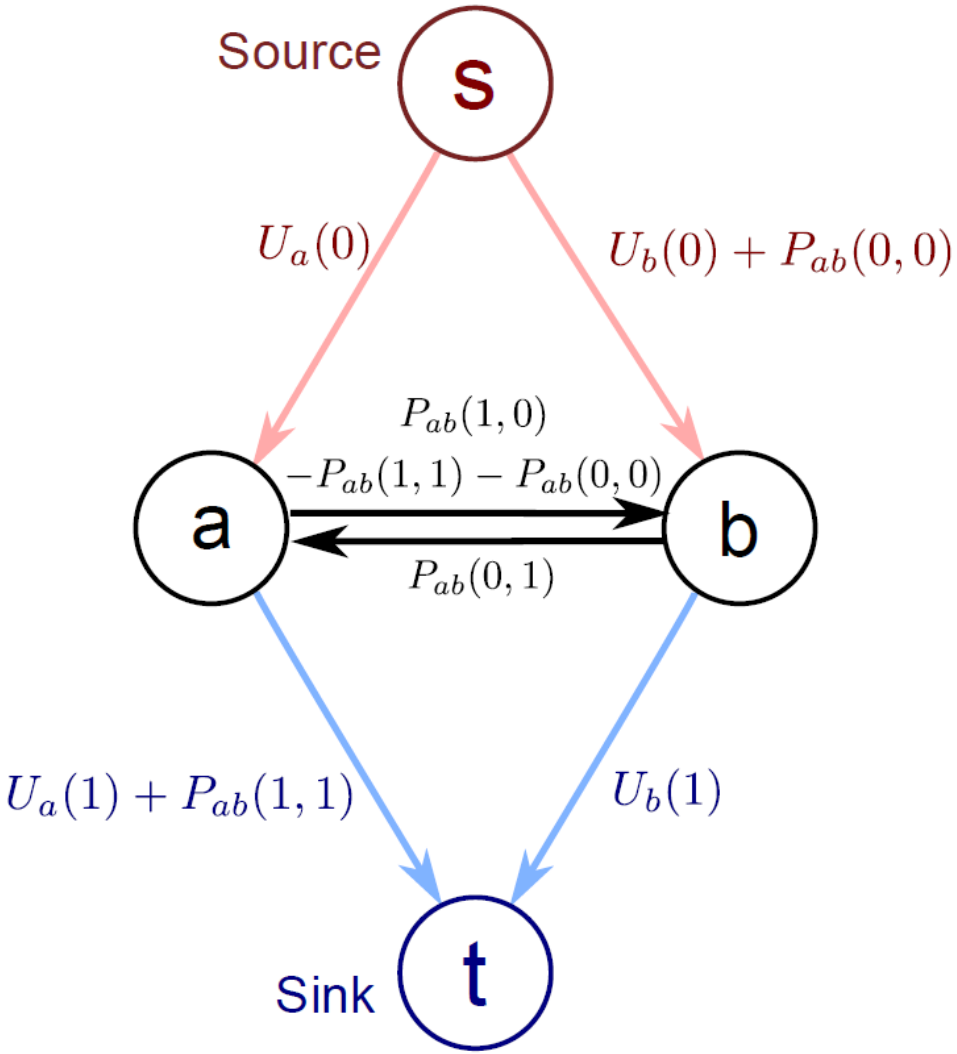


Reparameterization

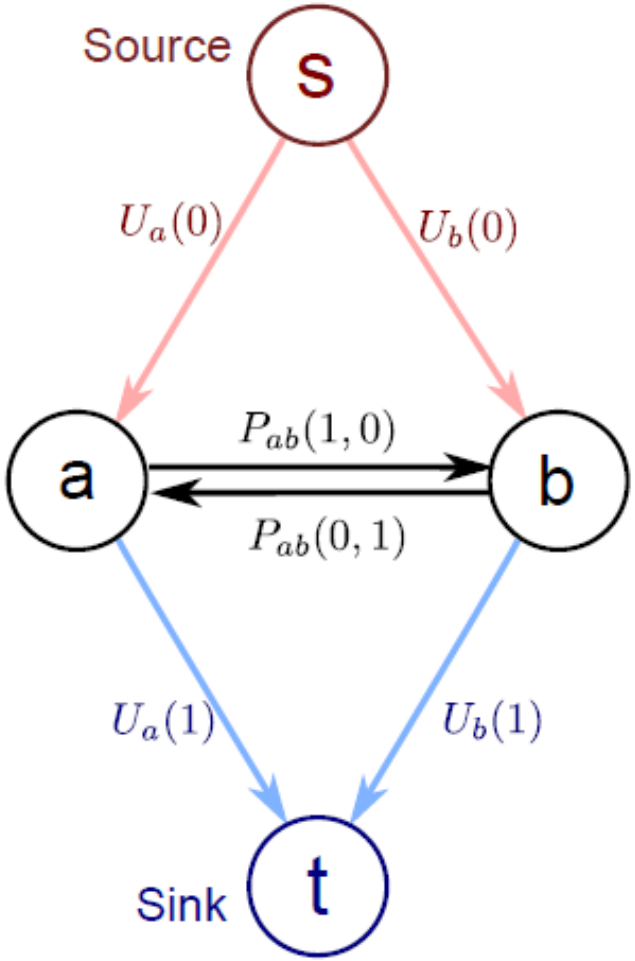
The max-flow / min-cut algorithms require that all of the capacities are non-negative.

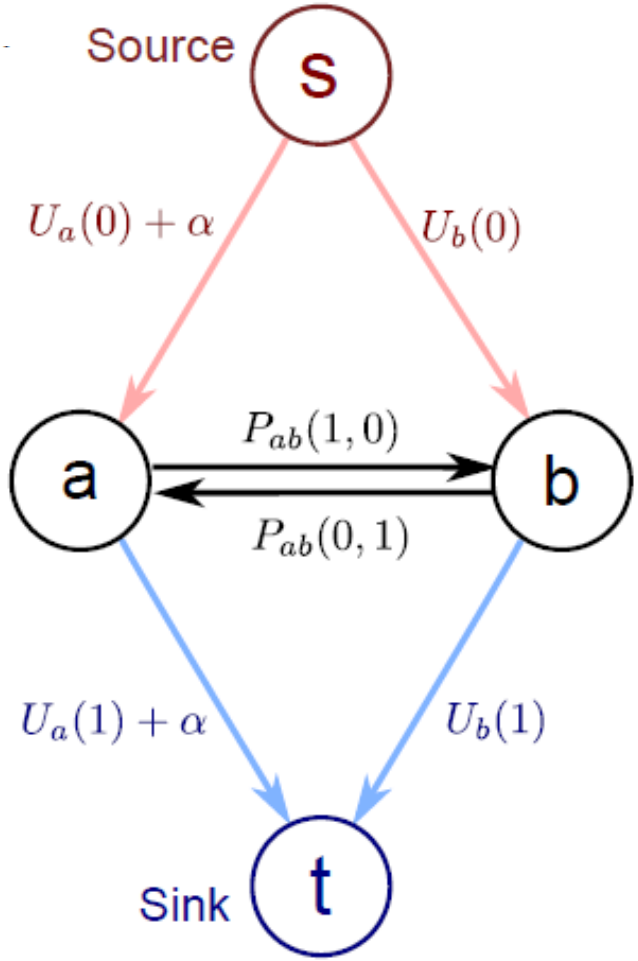
However, because we have a subtraction on edge a-b we cannot guarantee that this will be the case, even if all the original unary and pairwise costs were positive.

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)



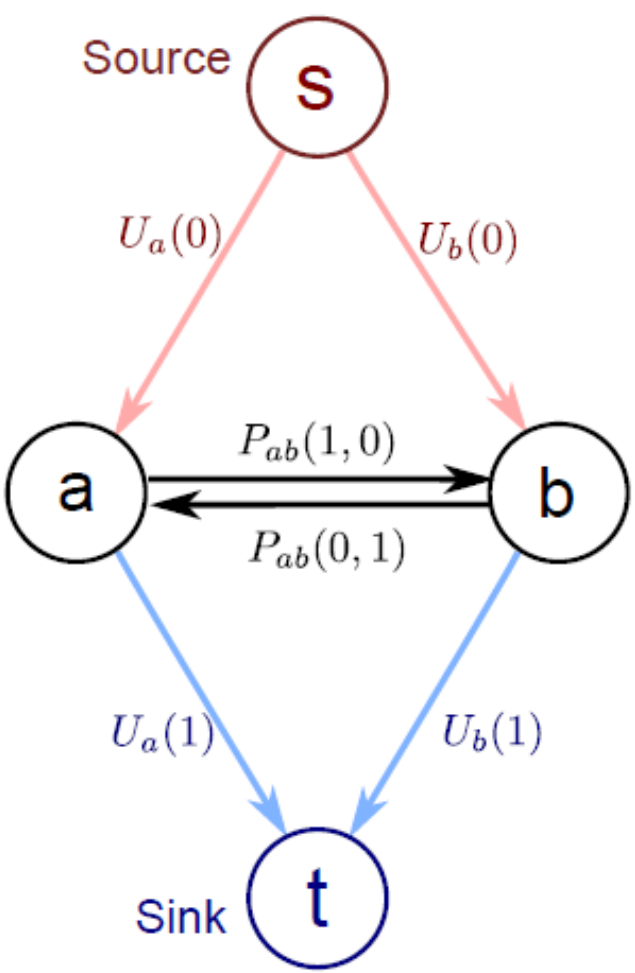
Reparameterization 1

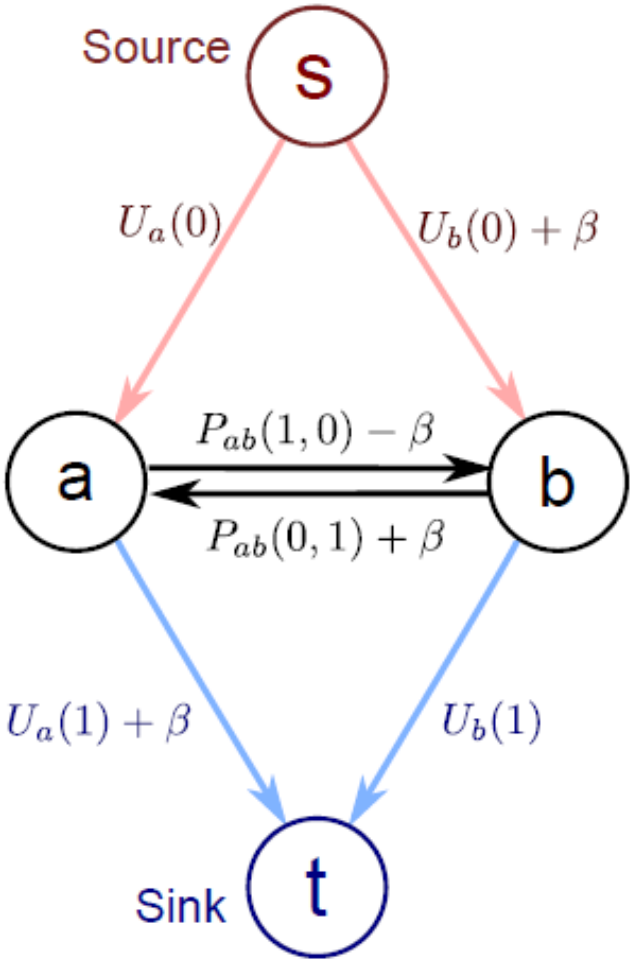




The minimum cut chooses the same links in these two graphs

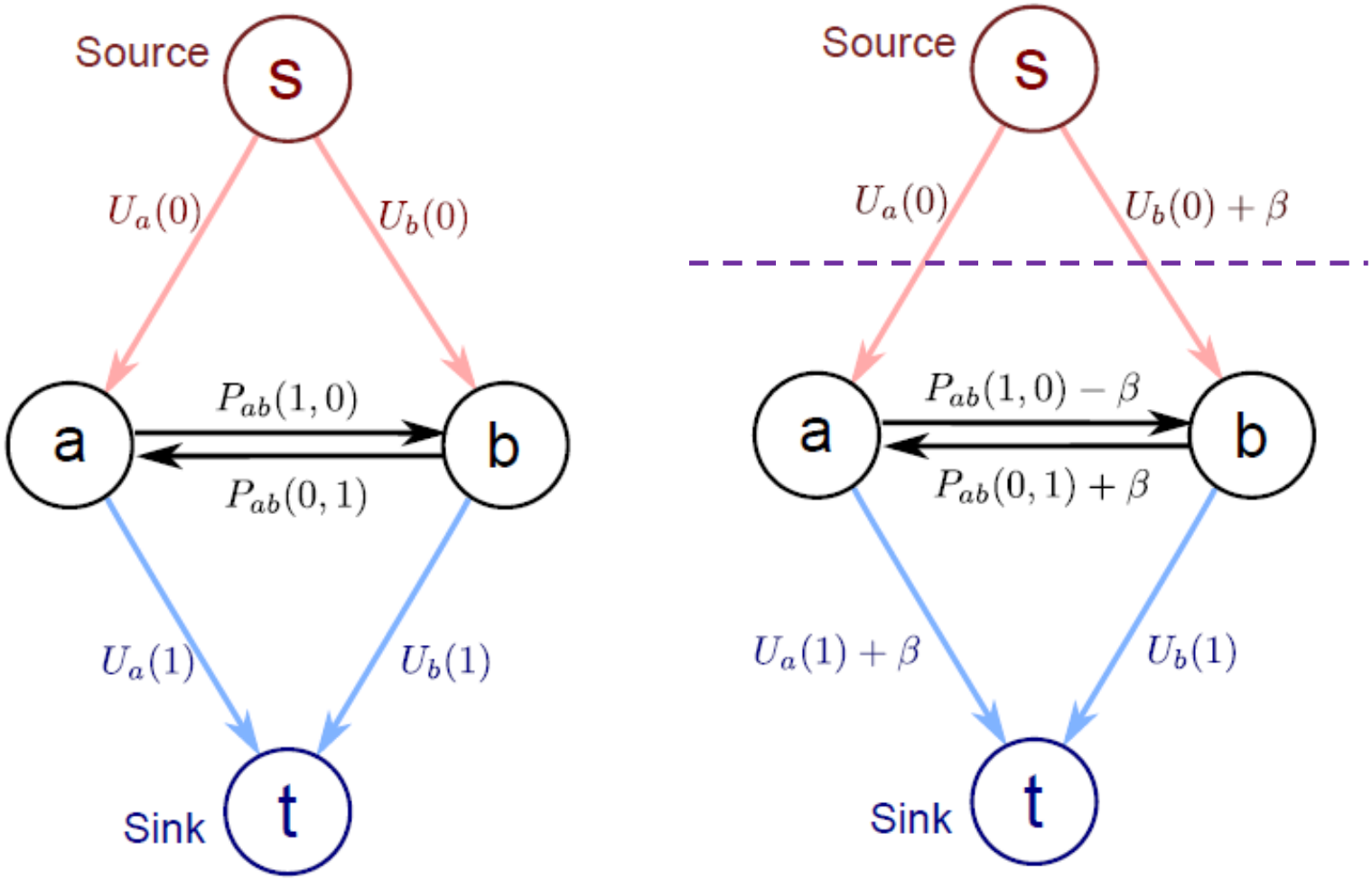
Reparameterization 2





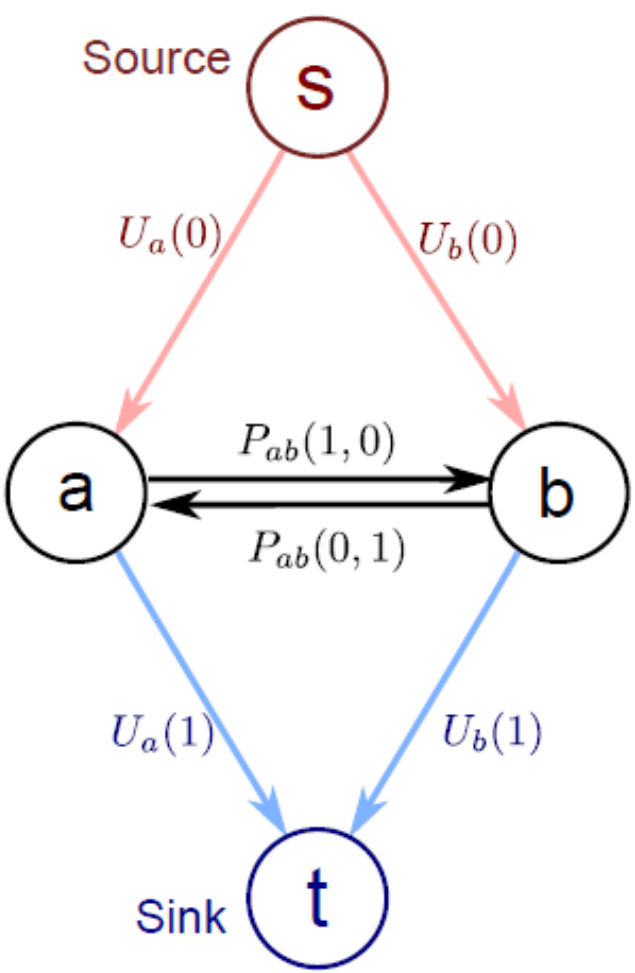
The minimum cut chooses the same links in these two graphs

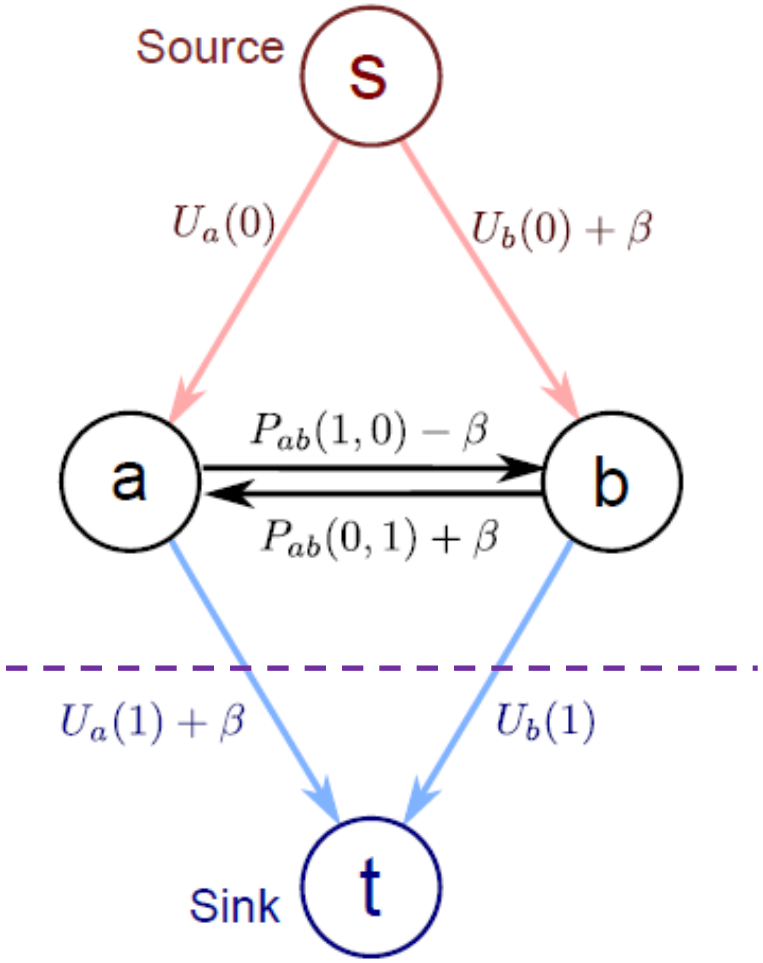
Reparameterization 2



The minimum cut chooses the same links in these two graphs

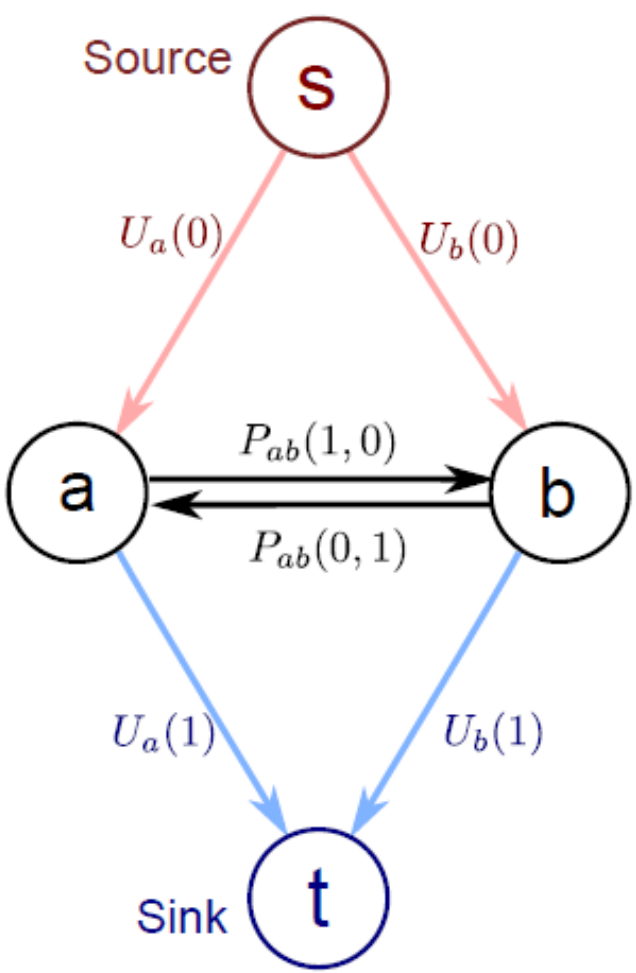
Reparameterization 2

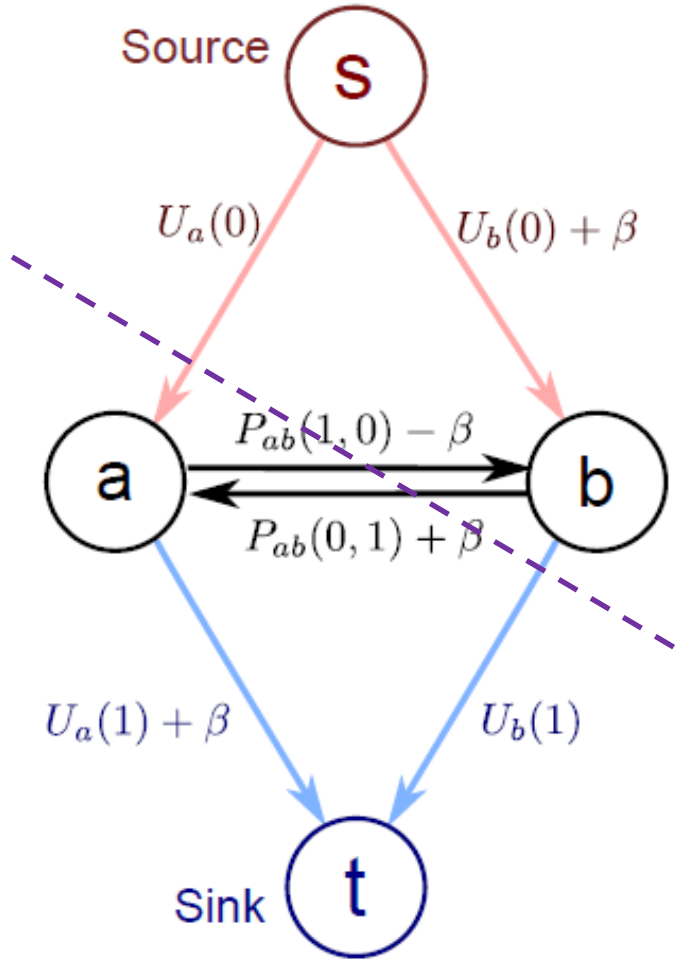




The minimum cut chooses the same links in these two graphs

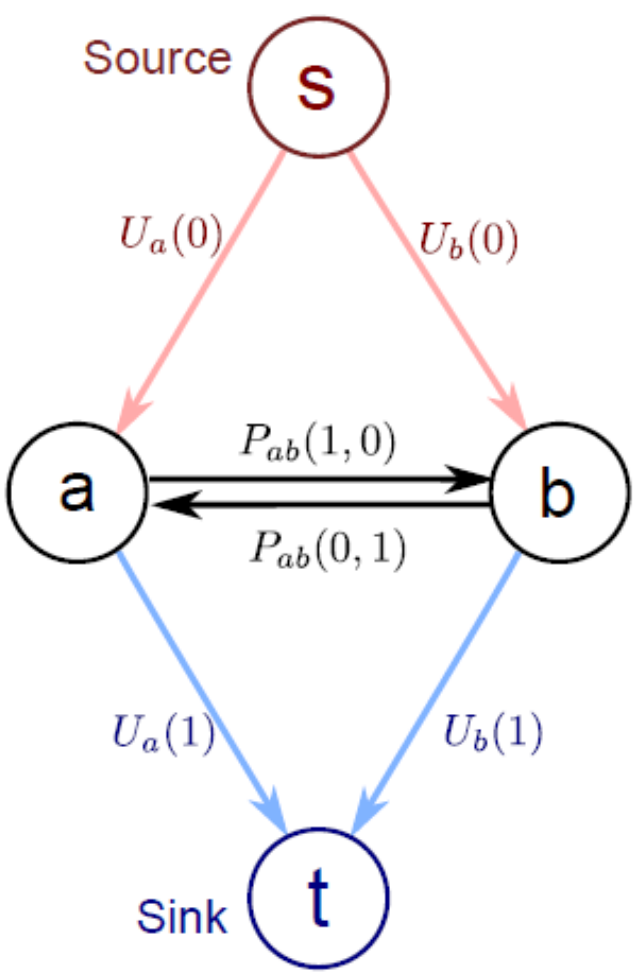
Reparameterization 2

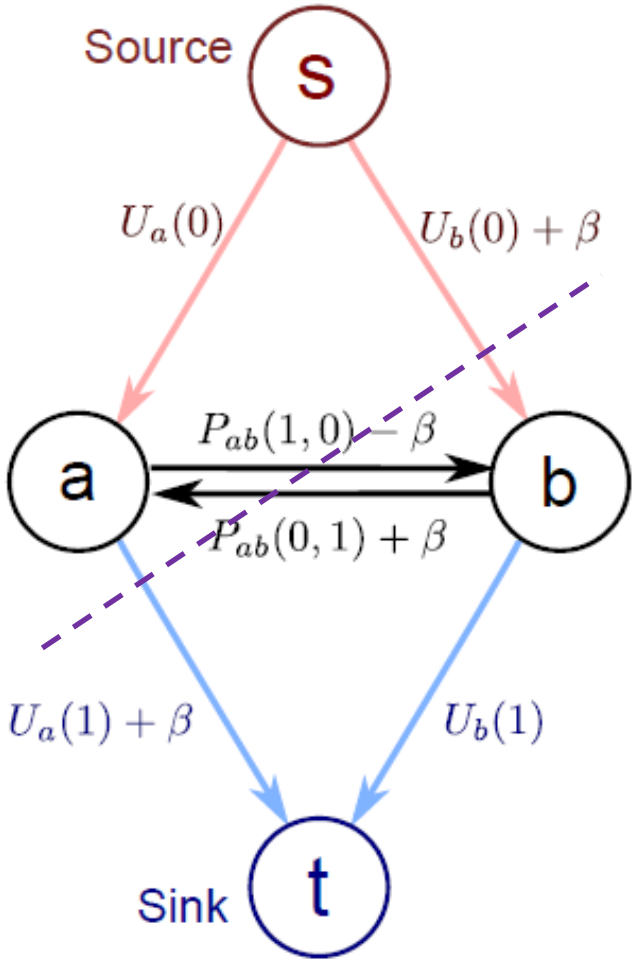




The minimum cut chooses the same links in these two graphs

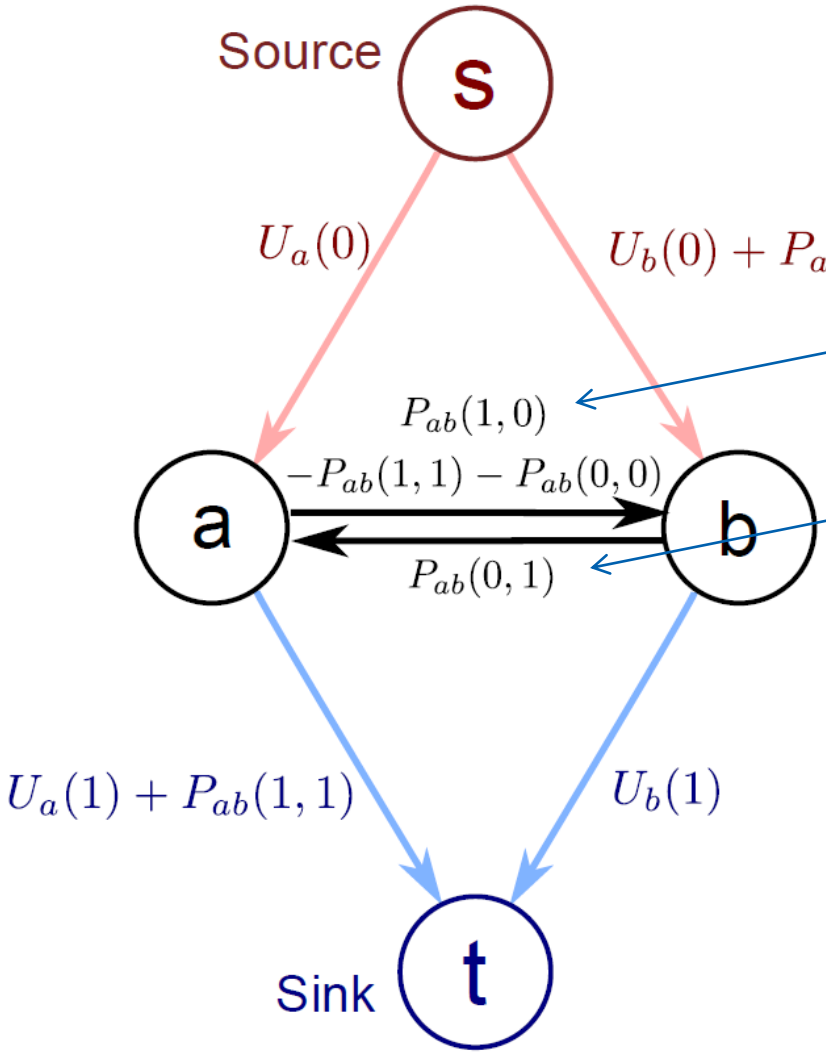
Reparameterization 2





The minimum cut chooses the same links in these two graphs

Submodularity



Subtract constant β

Add constant, β

$$\theta_{10} - \theta_{11} - \theta_{00} - \beta \geq 0$$

$$\theta_{01} + \beta \geq 0$$

Adding together implies

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

Submodularity

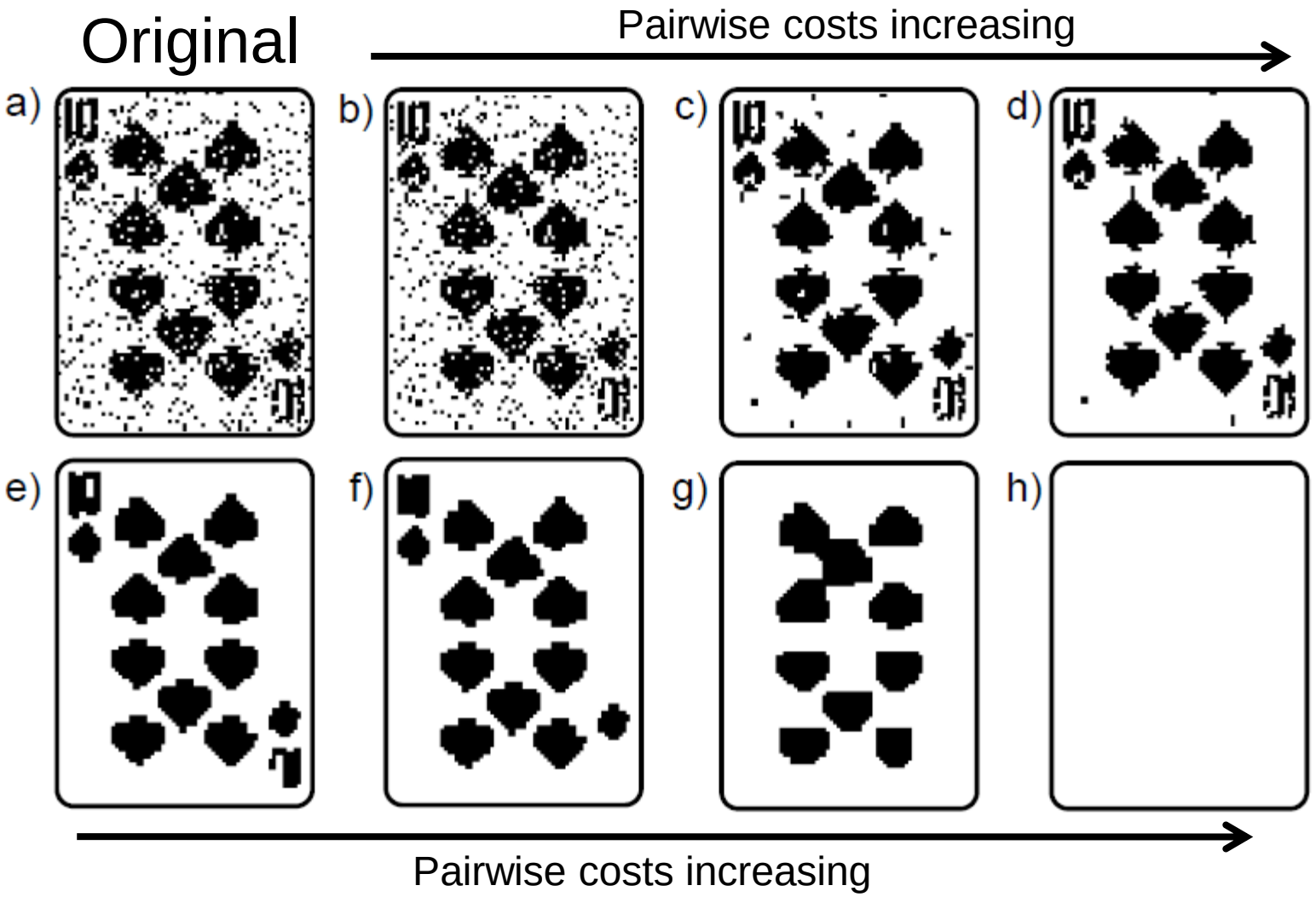
$$\theta_{10} + \theta_{01} - \theta_{11} - \theta_{00} \geq 0$$

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

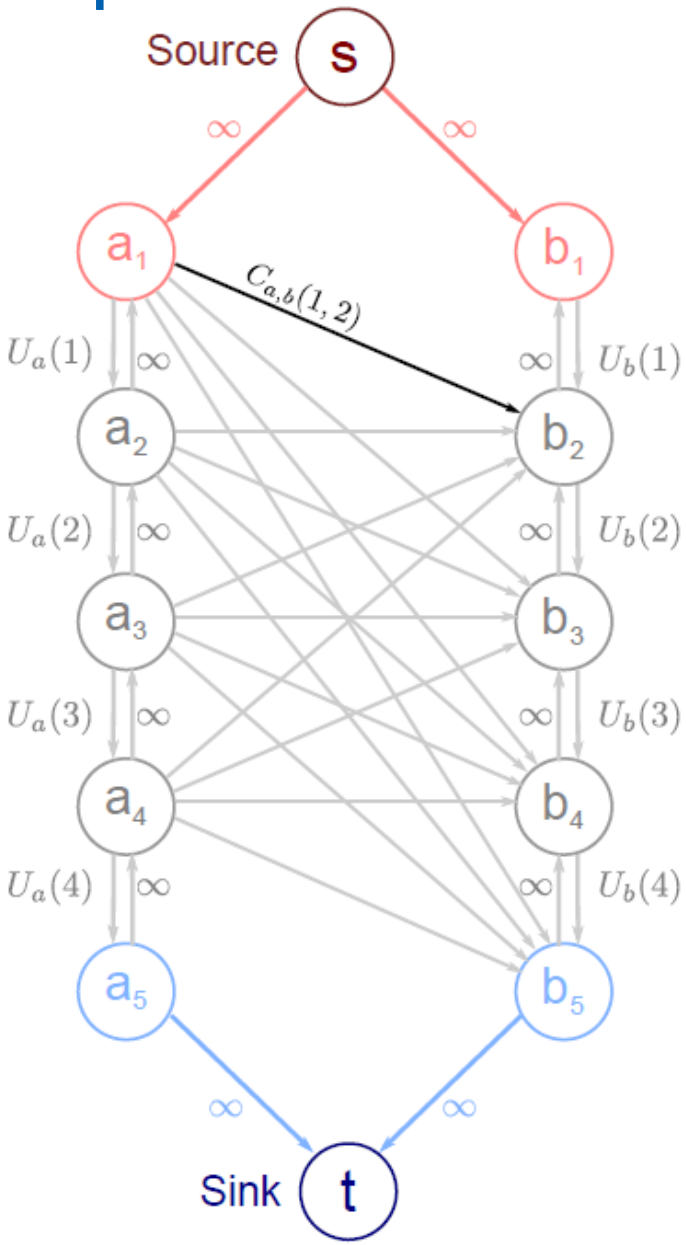
If it is not obeyed then the problem is NP hard.

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

Denoising Results



Multiple Labels

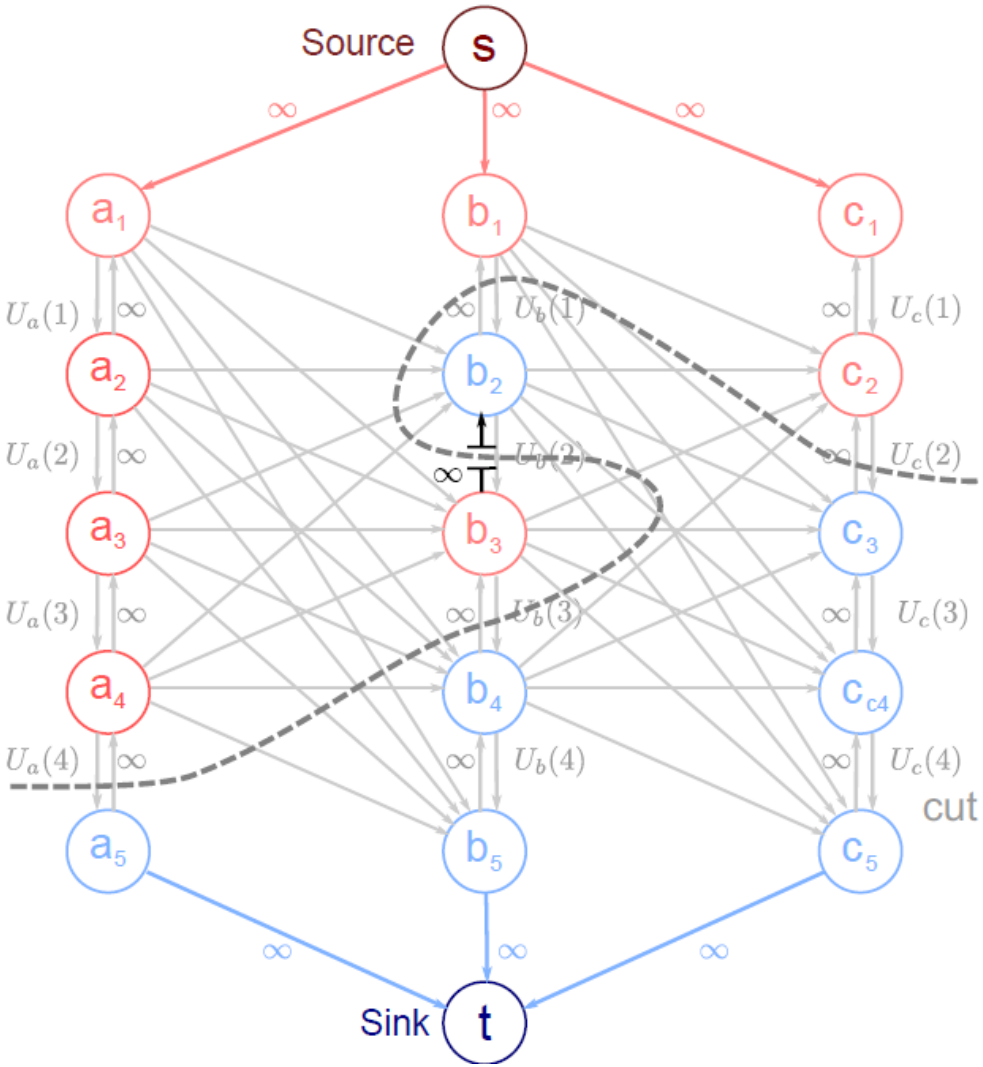


Construction for two pixels (a and b) and four labels (1,2,3,4)

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.

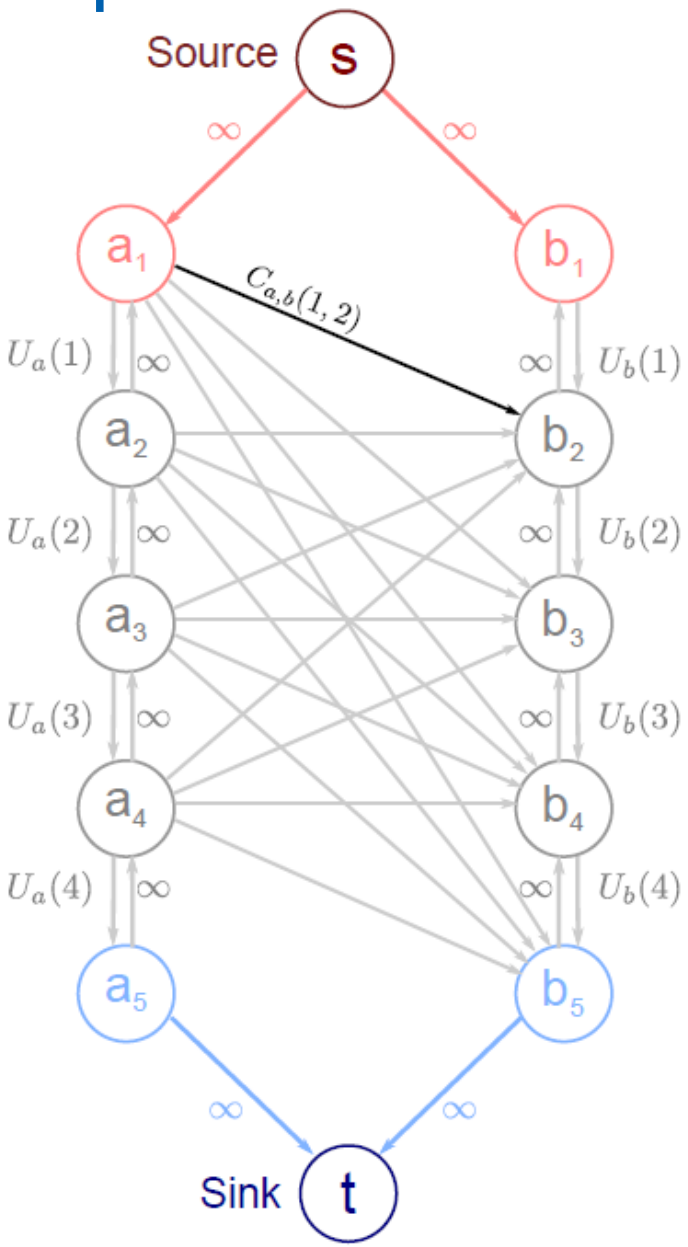
Constraint Edges



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:

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

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

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

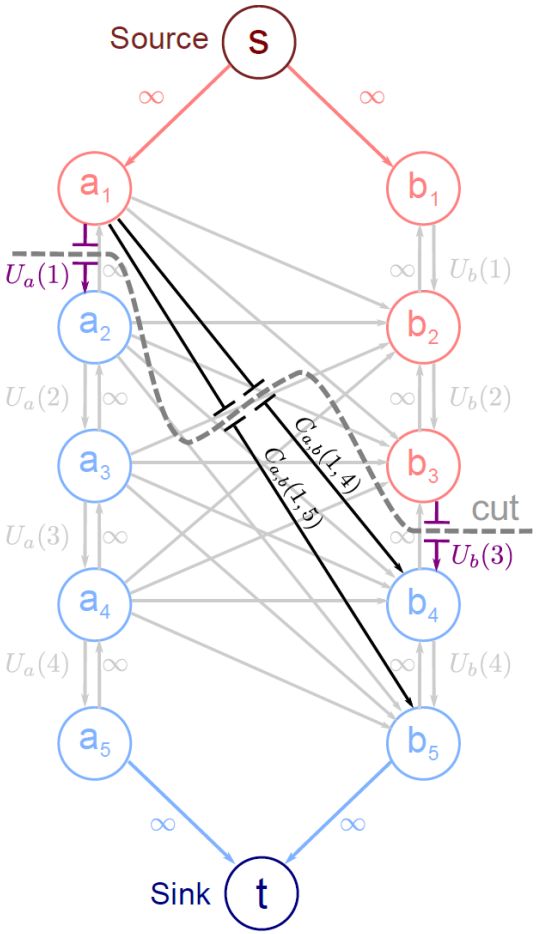
Superfluous terms :

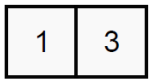
$$P_{ab}(i,0) = 0 \qquad P_{ab}(i,K+1) = 0$$

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

For all i,j where K is number of labels

Example Cuts





Cost

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

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

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

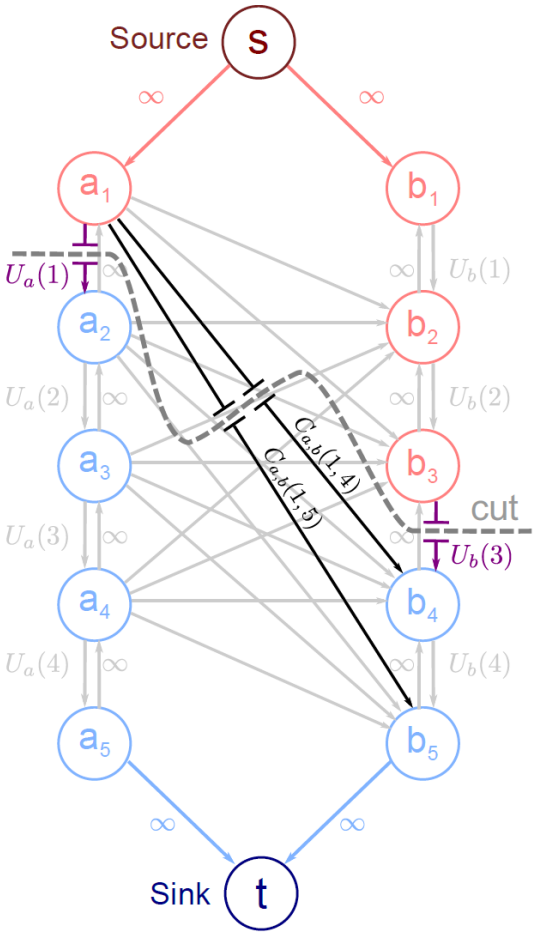
$$- \; P_{ab}(i,j) - P_{ab}(i-1,j-1)$$

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

Cost:
U(1) + U(3) + C(1,4) + C(1,5)
= U(1) + U(3)
+ (P(1,3) + P(0,4) – P(1,4) – P(0,3))
+ (P(1,4) + P(0,5) – P(1,5) – P(0,4))
= U(1) + U(3) + P(1,3)

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

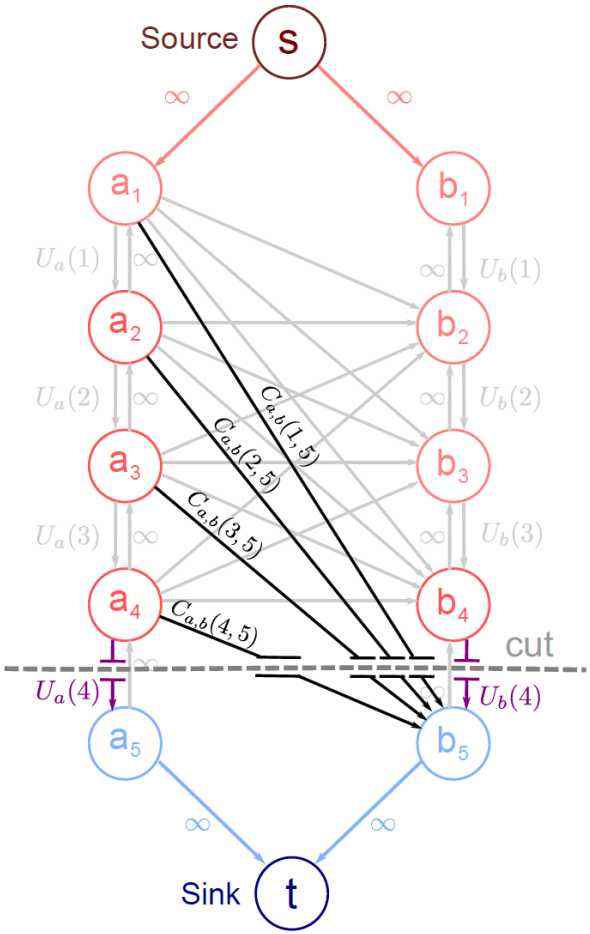
Example Cuts



1	3

Cost

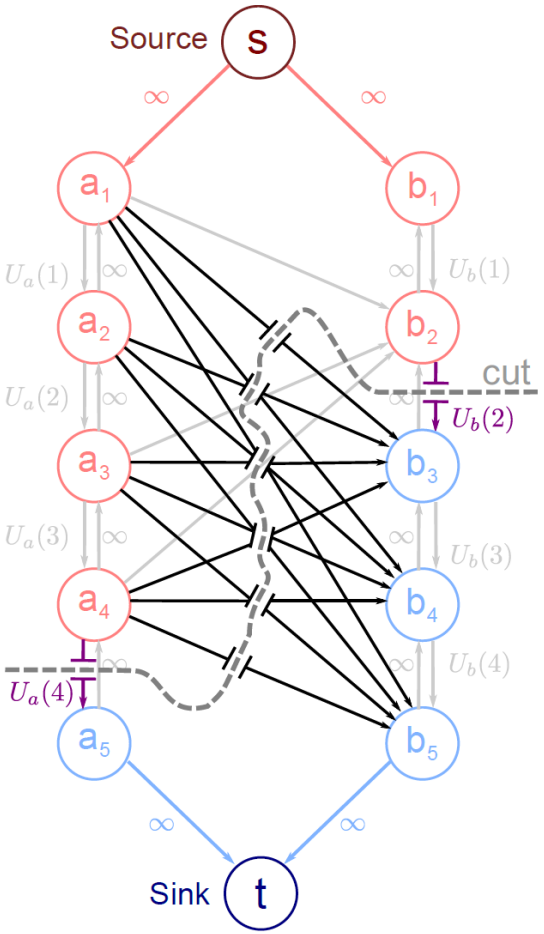
$$U_a(1) + U_b(3) + \sum_{i=1}^{1}\sum_{j=4}^{5} C_{ab}(i,j)$$
$$= U_a(1) + U_b(3) + P_{ab}(1,3)$$



4	4

Cost

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



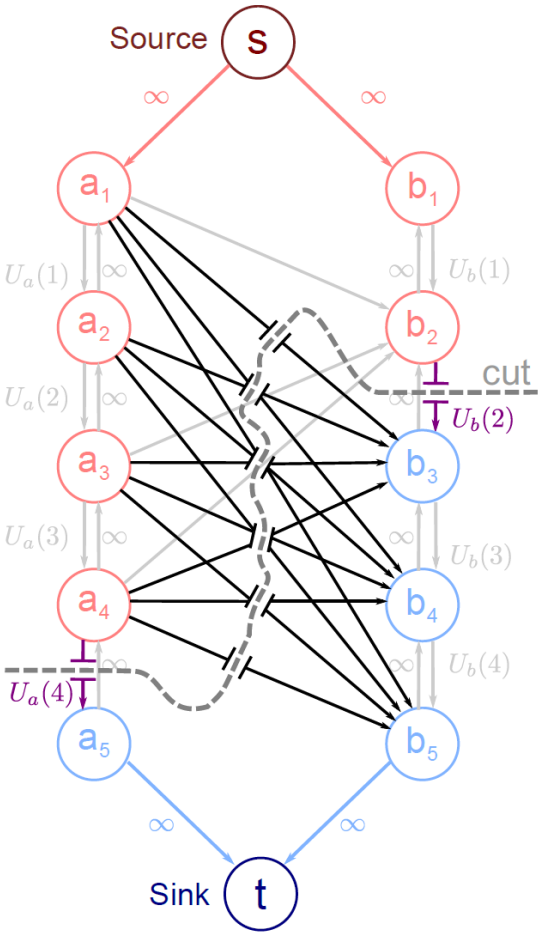
4	2

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



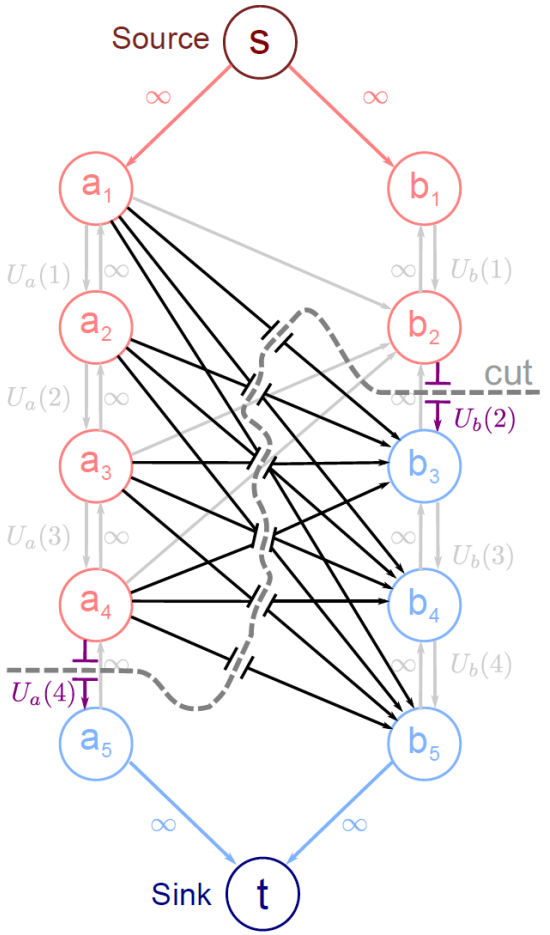
Must cut links from before cut on pixel a to after cut on pixel b.

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*

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

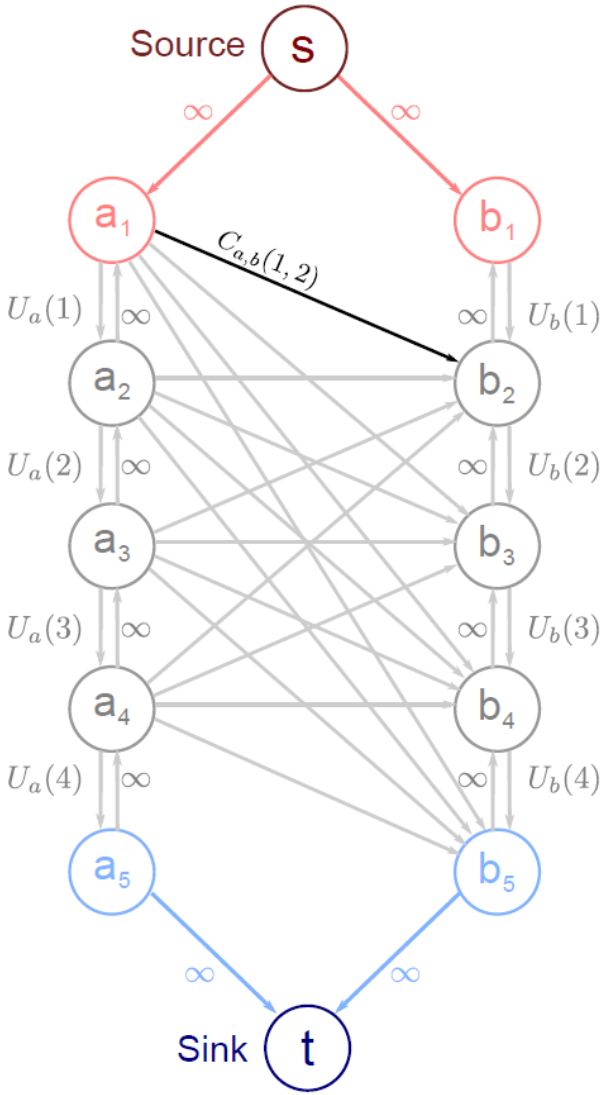
Pairwise Costs

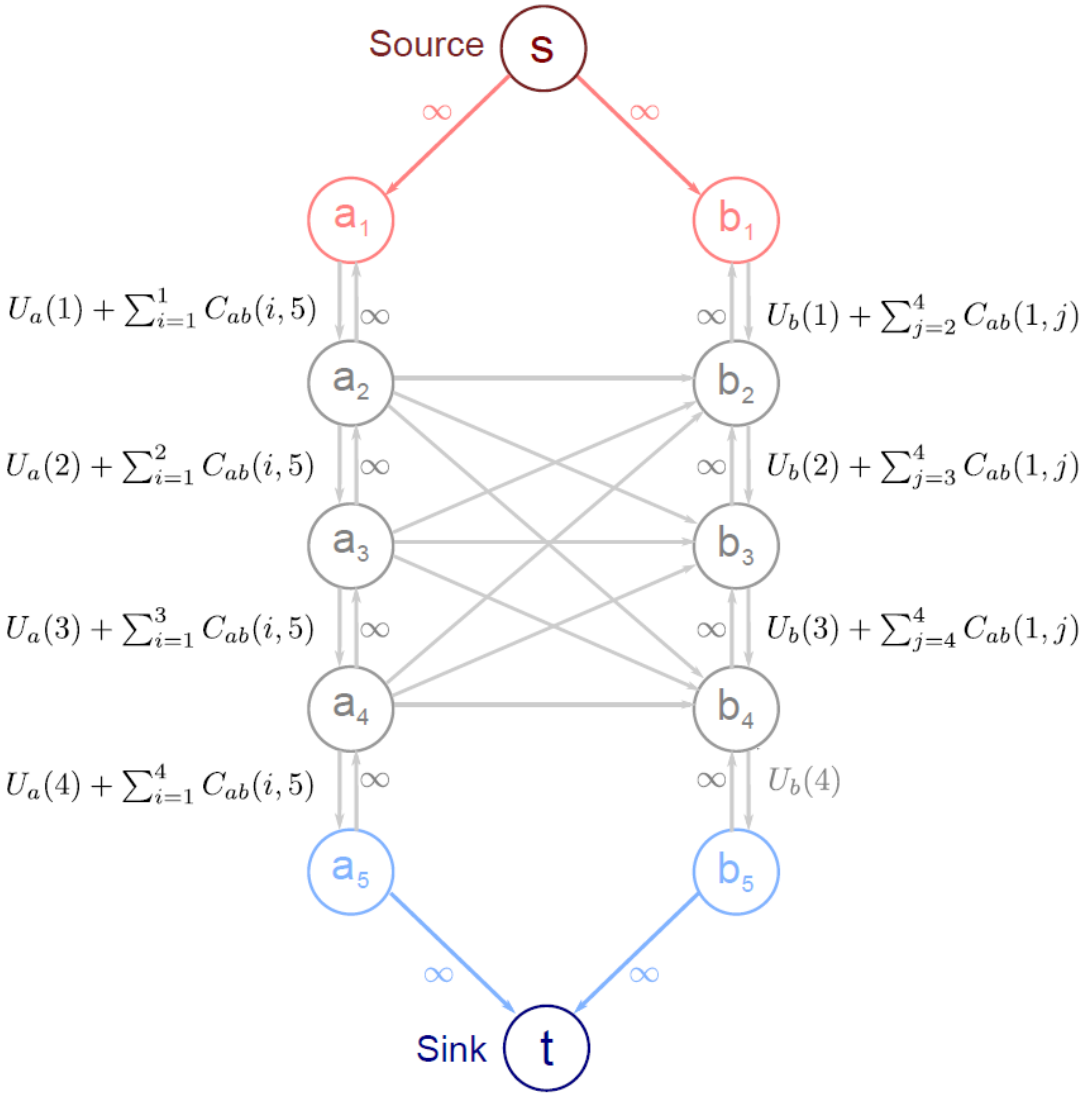


If pixel a takes label I and pixel b takes label J

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

Reparameterization





Submodularity

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

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

or

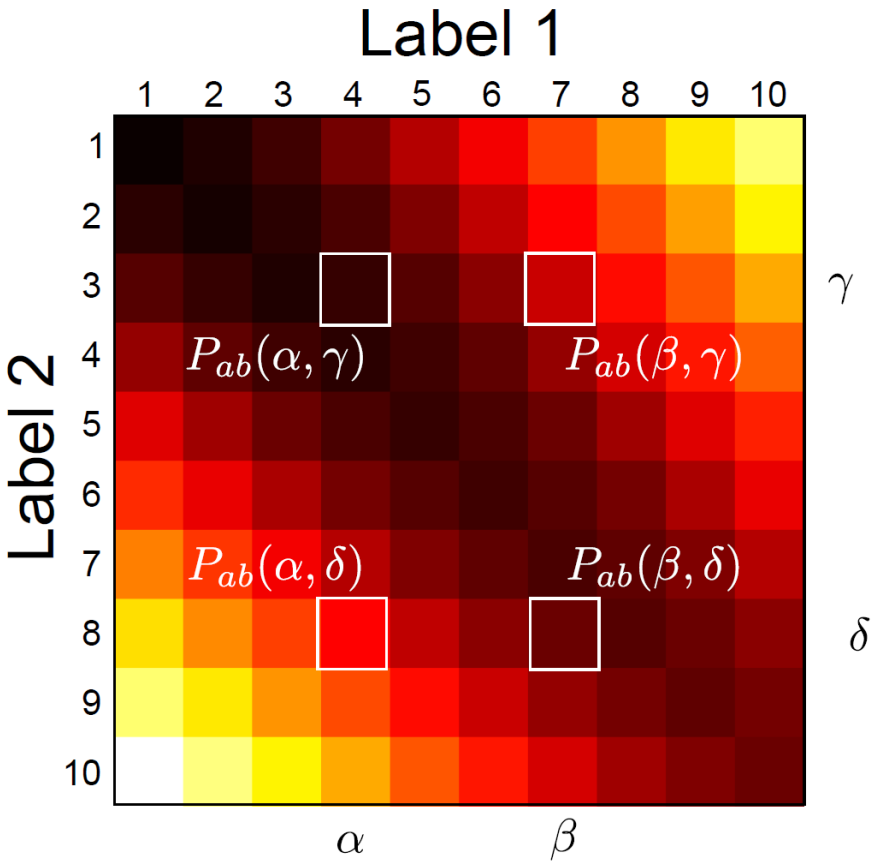
$$P_{ab}(i,j-1) + P_{ab}(i-1,j) - P_{ab}(i,j) - P_{ab}(i-1,j-1) \geq 0$$

By mathematical induction we can get the more general result for $\beta > \alpha$ and $\delta > \gamma$:

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

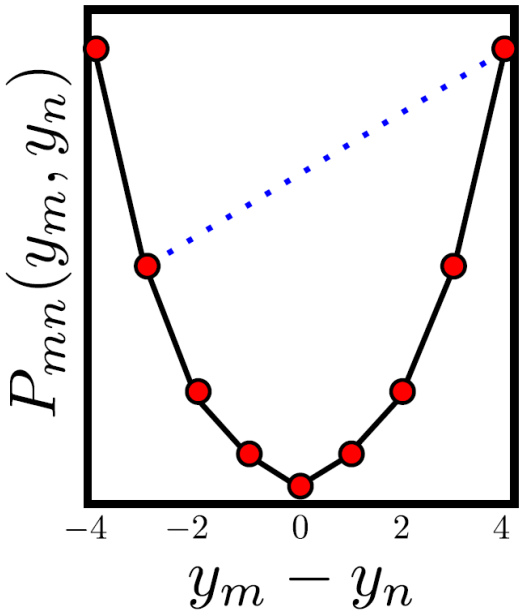
Submodularity

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



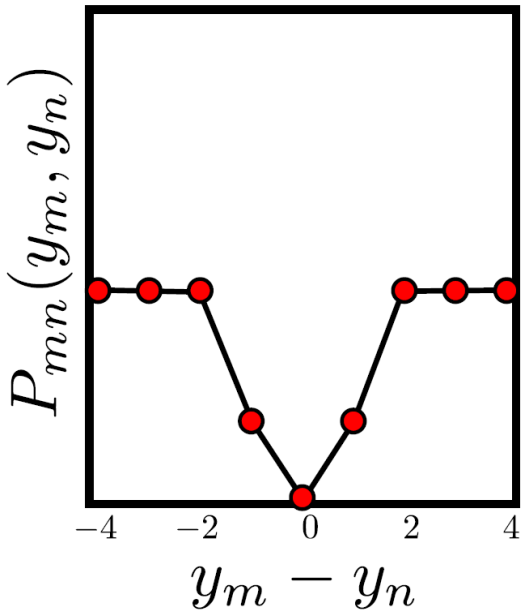
If not submodular, then the problem is NP hard.

Convex vs. non-convex costs



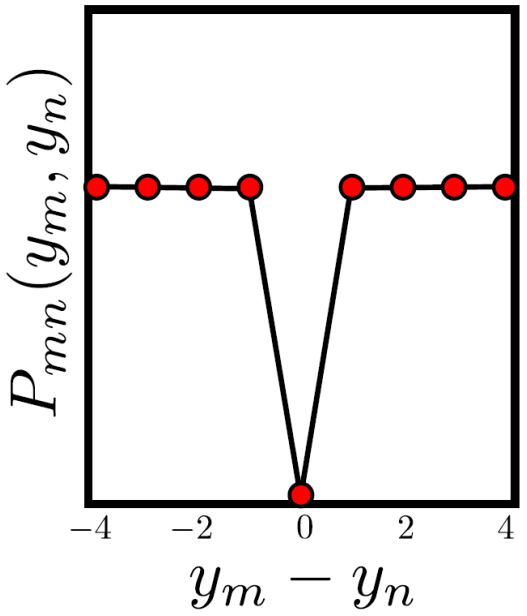
Quadratic
- Convex
- Submodular

$$\kappa(w_m - w_n)^2$$



Truncated Quadratic
- Not Convex
- Not Submodular

$$\min(\kappa_1, \kappa_2(w_m - w_n)^2)$$

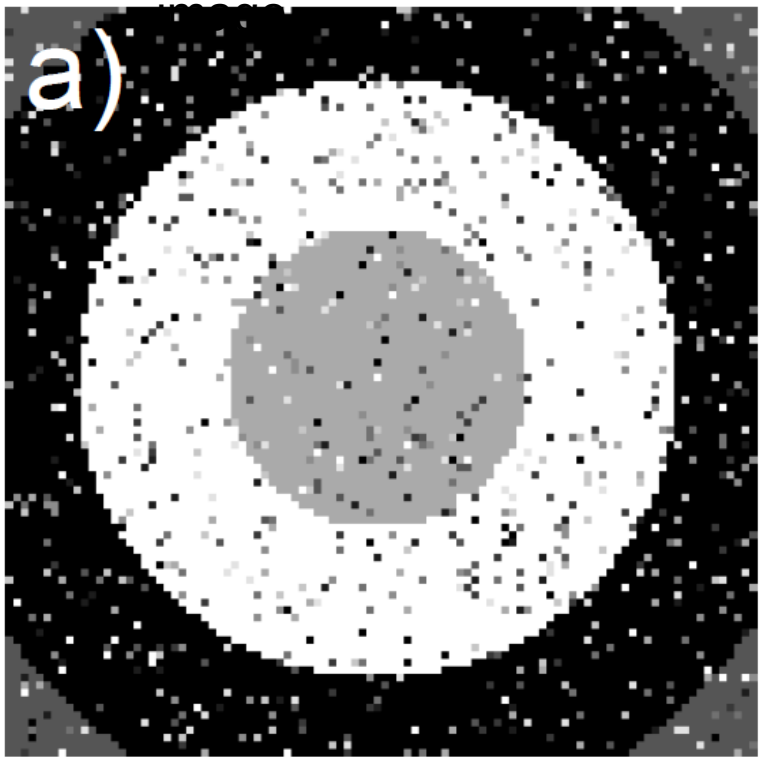


Potts Model
- Not Convex
- Not Submodular

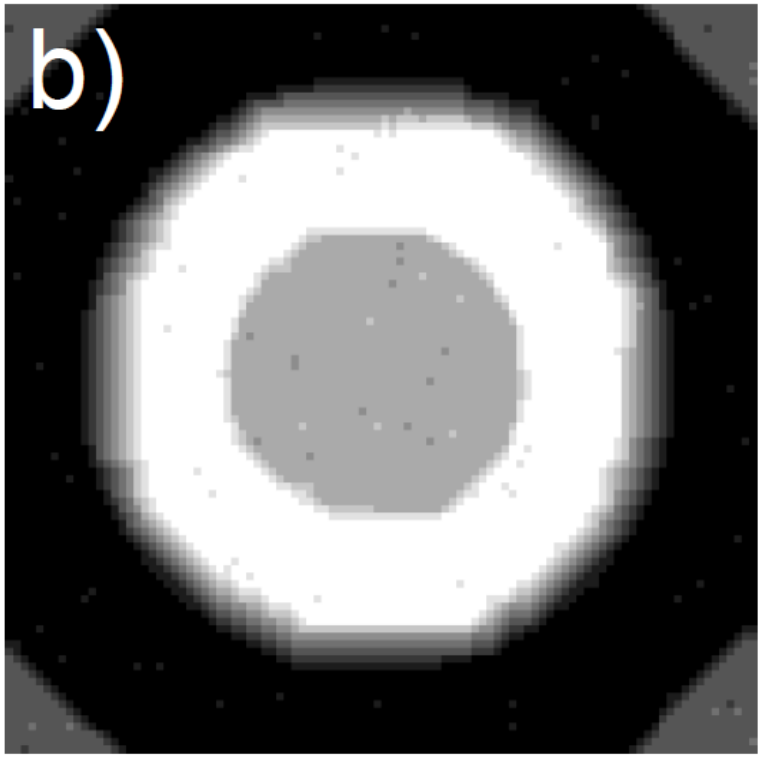
$$\kappa(1 - \delta(w_m - w_n))$$

What is wrong with convex costs?

Observed noisy image



Denoised result



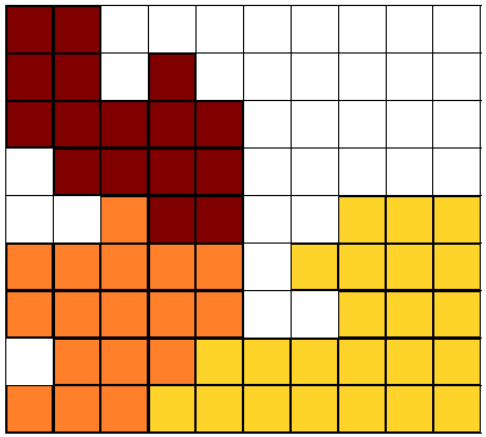
- Pay lower price for many small changes than one large one
- Result: blurring at large changes in intensity

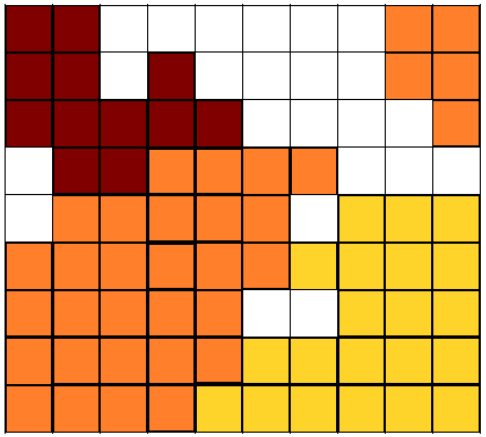
Alpha Expansion Algorithm

UNIVERSITÄT BONN

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

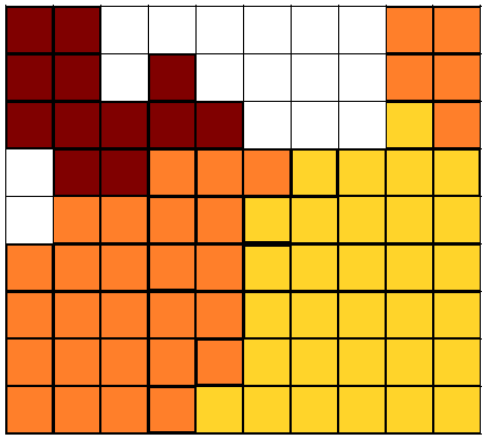
Initial labelling

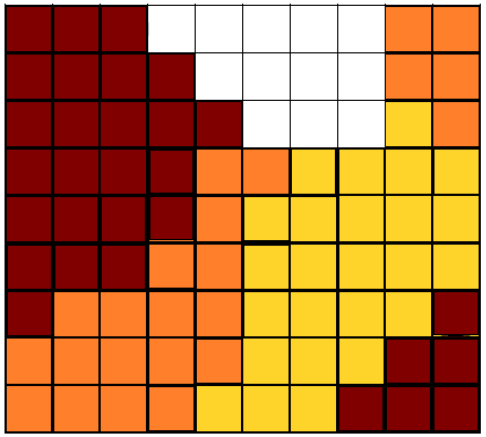




Iteration 1 (orange)

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.

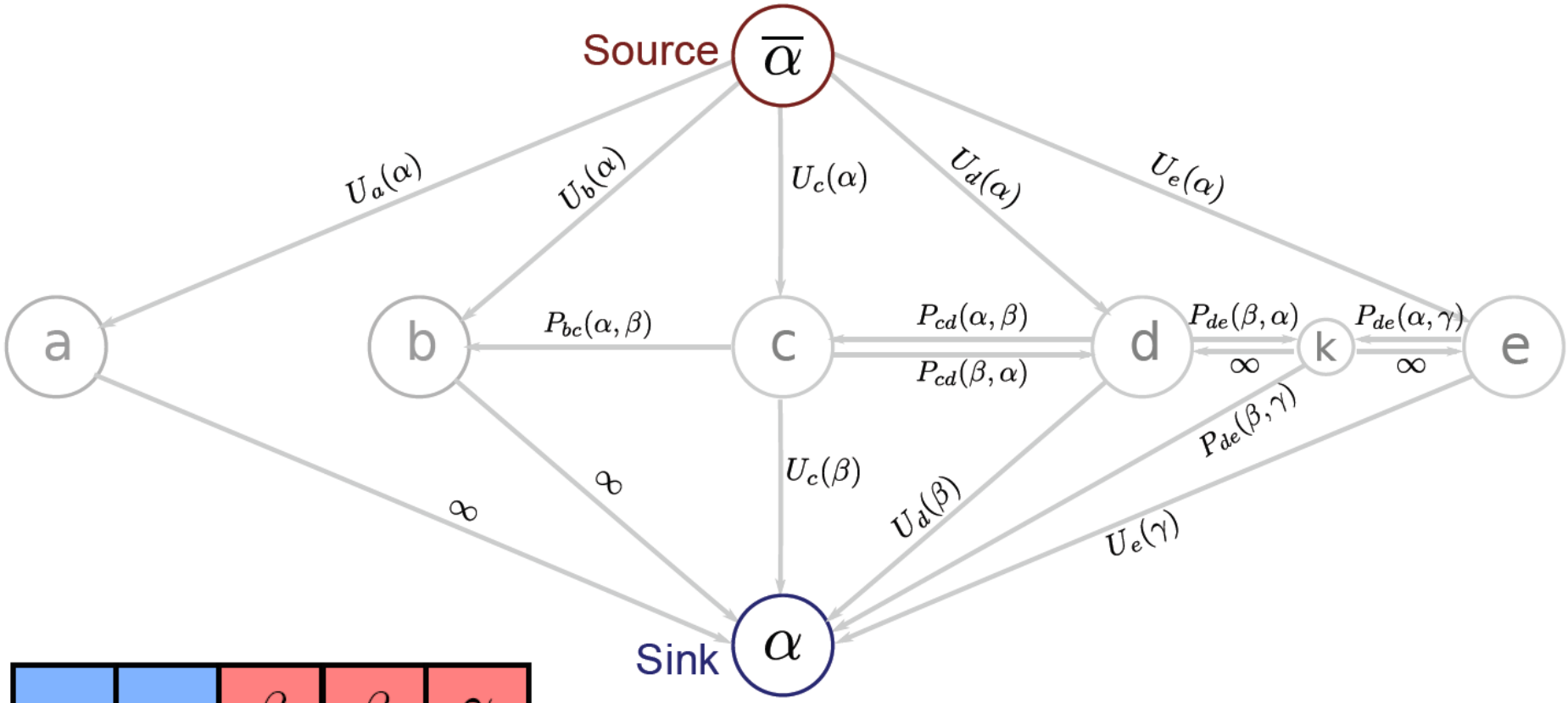
Each step optimal, but overall global maximum not guaranteed

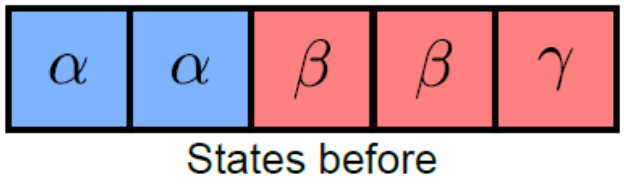
Proved to be within a factor of 2 of global optimum.

Requires that pairwise costs form a metric:

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

Alpha Expansion Construction

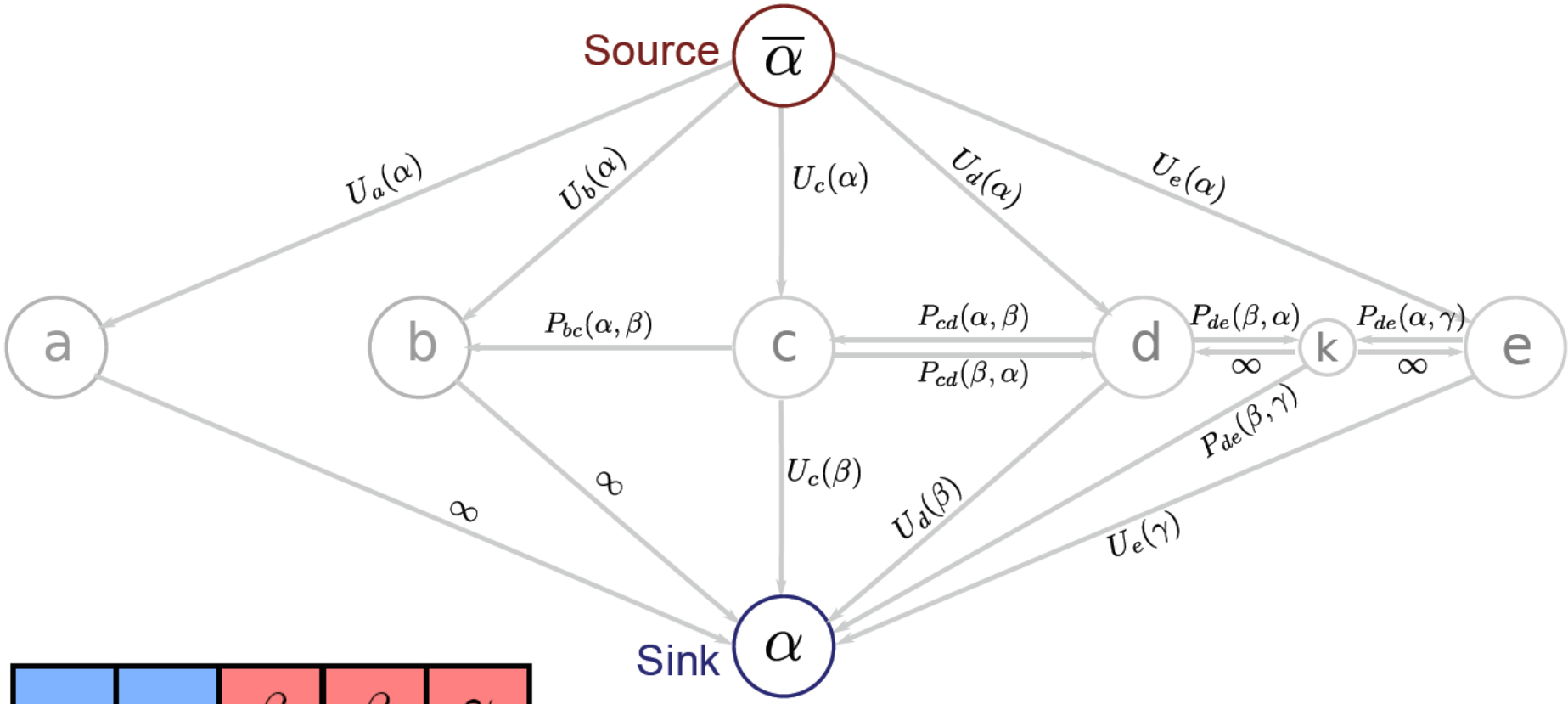


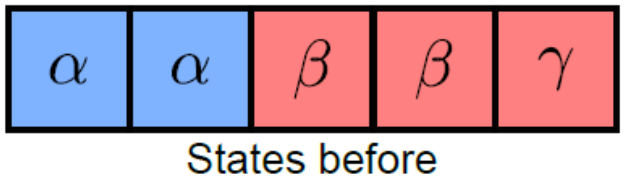


Binary graph cut – either cut link to source (assigned to α) or to sink (retain current label)

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

Alpha Expansion Construction

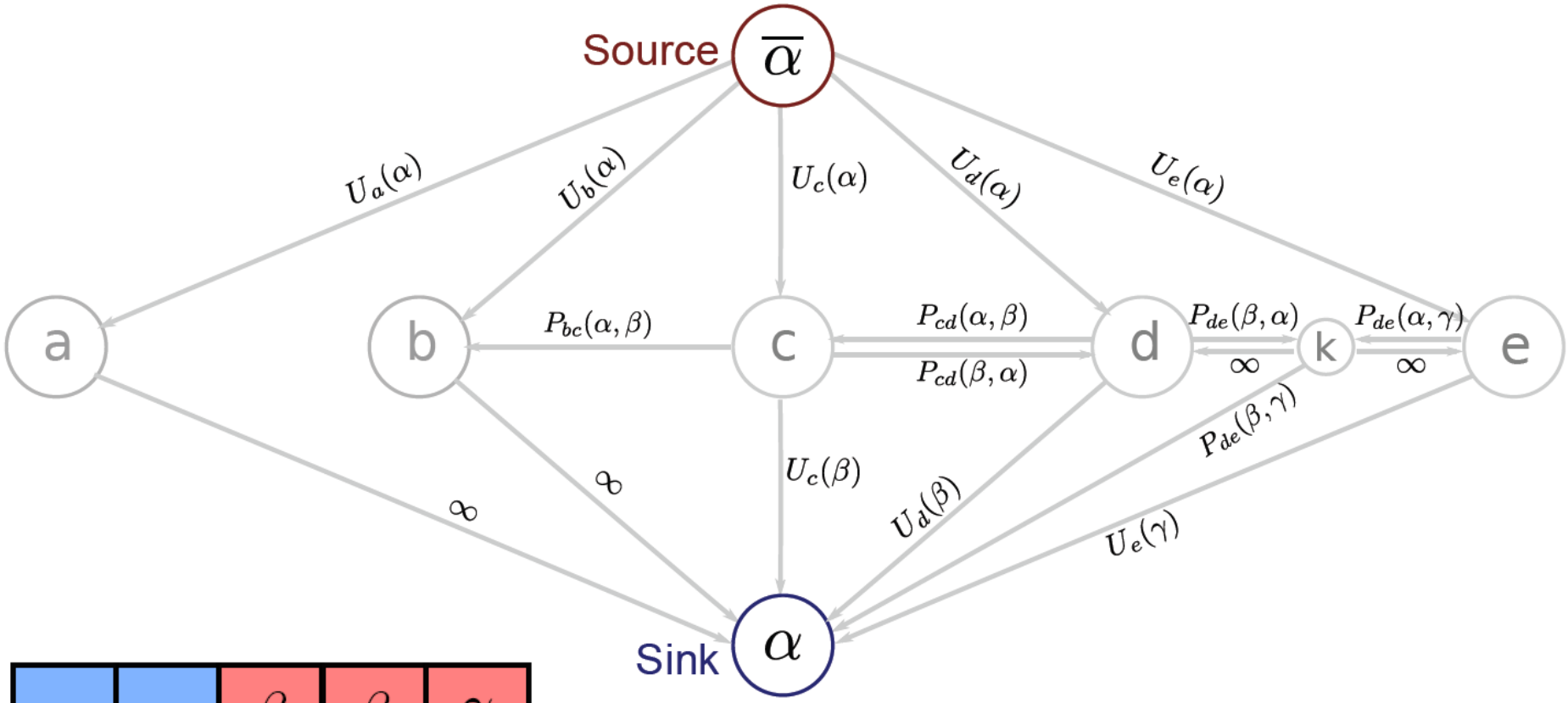


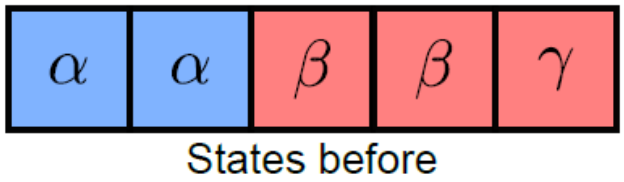


Graph is dynamic. Structure of inter-pixel links depends on α and the choice of labels.

There are four cases.

Alpha Expansion Construction

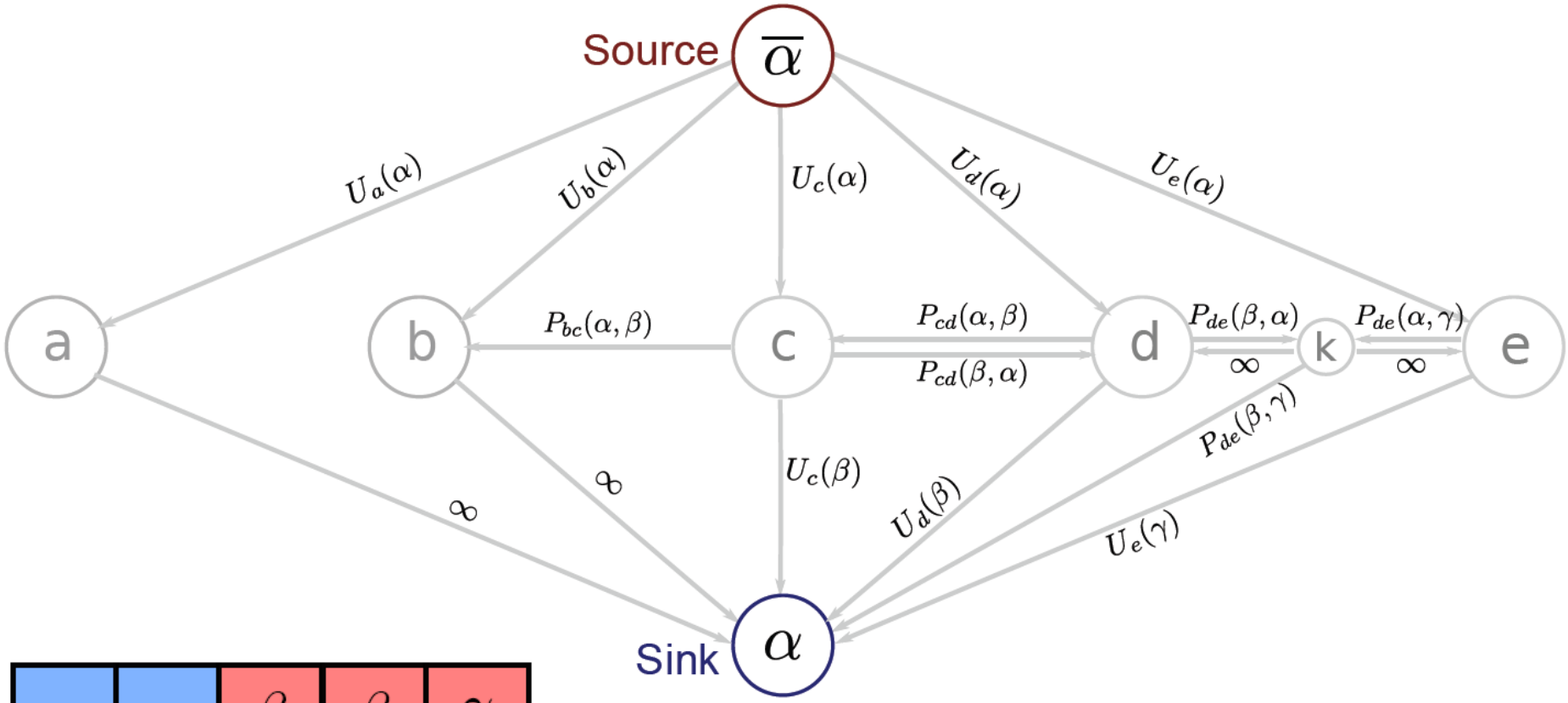


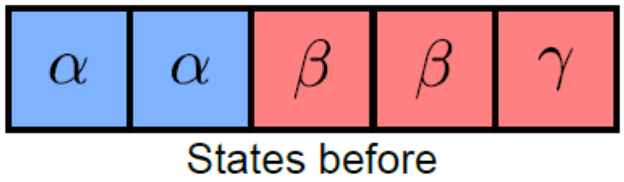


Case 1:

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

Alpha Expansion Construction

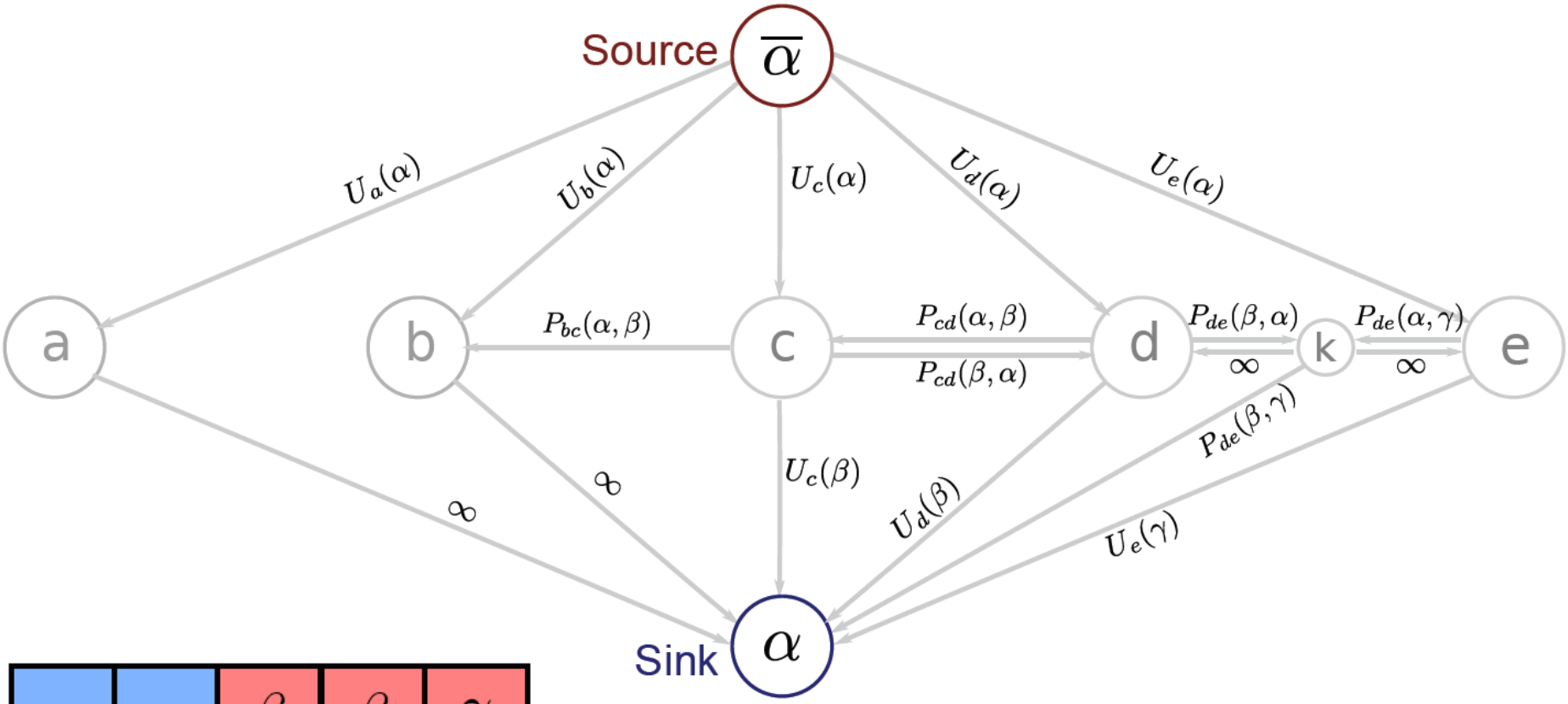


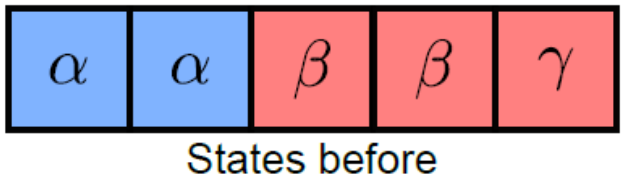


Case 2: Adjacent pixels are α,β.
Result either

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

Alpha Expansion Construction

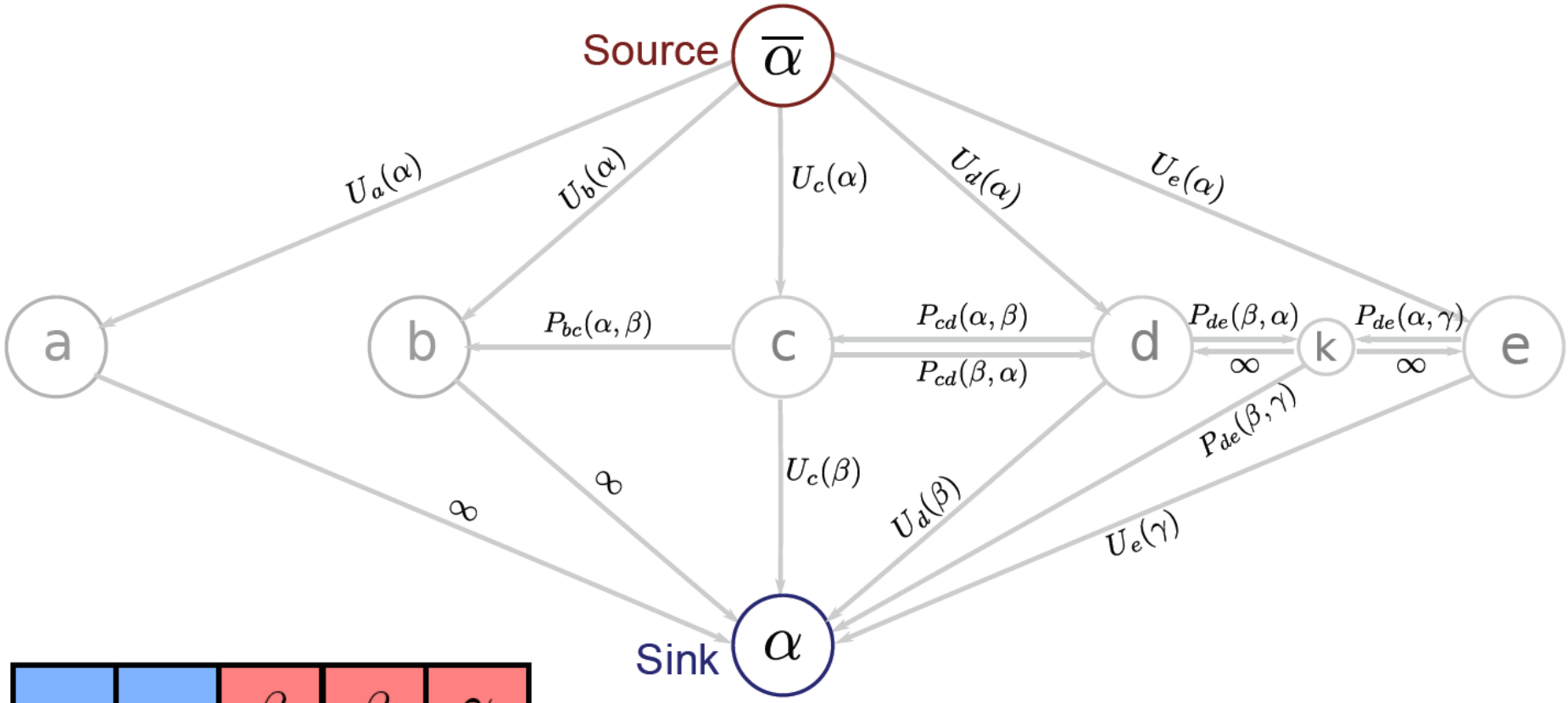


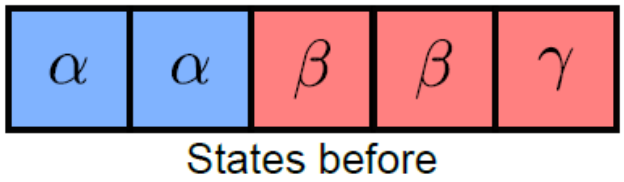


Case 3: Adjacent pixels are β,β. Result either

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

Alpha Expansion Construction

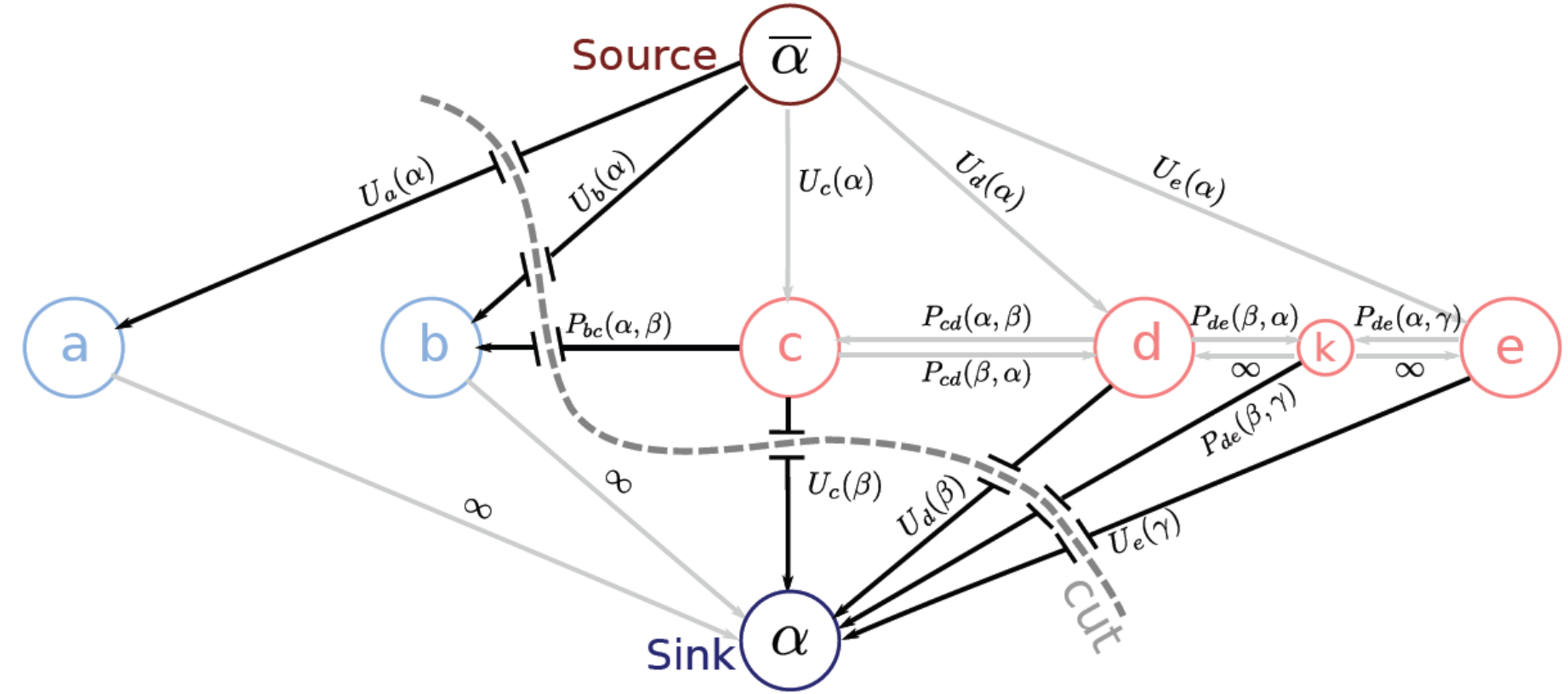


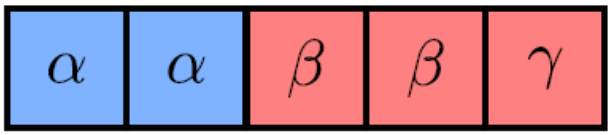


Case 4: Adjacent pixels are β,γ. Result either

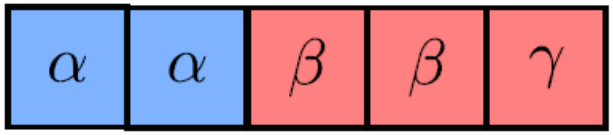
- β,γ (P(β,γ), add new edge).
- α,γ (P(α,γ), add new edge).
- β,α (P(β,α), add new edge).
- α,α (no cost and no new edge).

Example Cut 1





States before

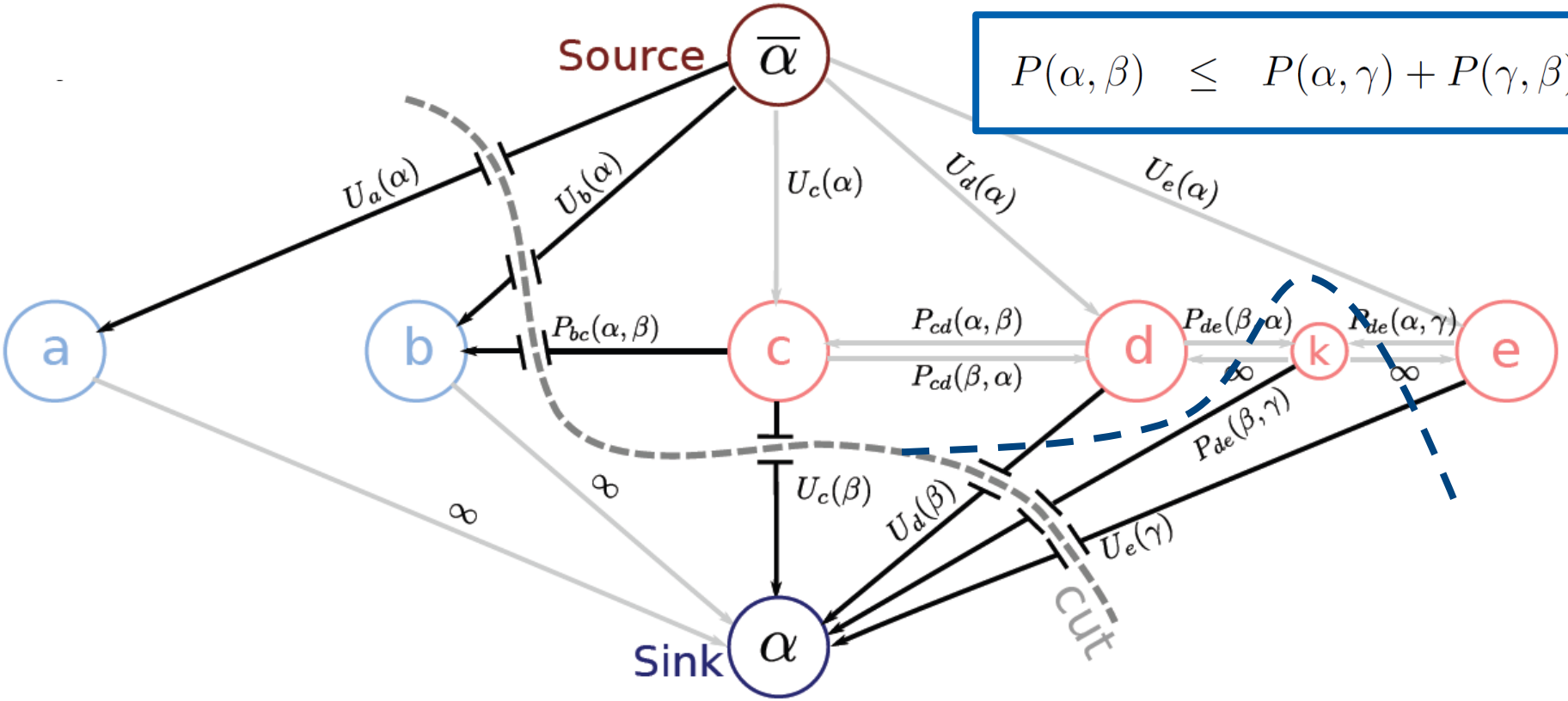


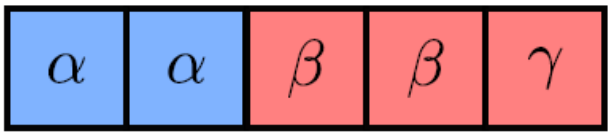
States after

Cost

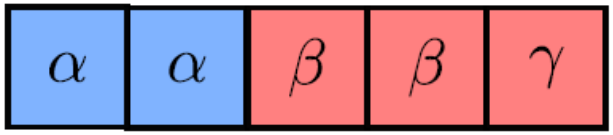
$$U_a(\alpha) + U_b(\alpha) + U_c(\beta) + U_d(\beta)$$
$$+U_e(\gamma) + P_{bc}(\alpha,\beta) + P_{de}(\beta,\gamma)$$

Example Cut 1





States before

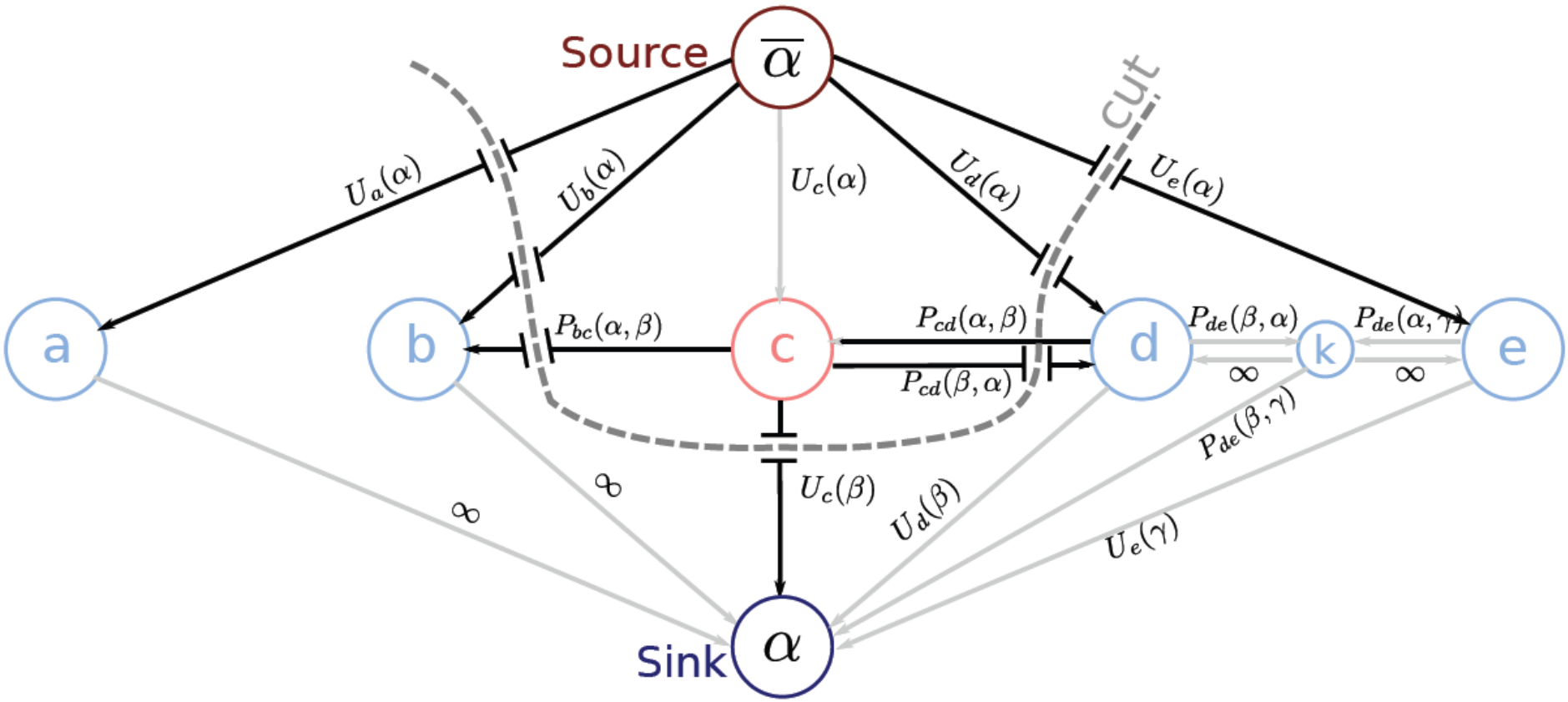


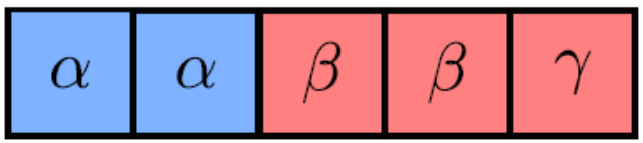
States after

Cost

$$U_a(\alpha) + U_b(\alpha) + U_c(\beta) + U_d(\beta)$$
$$+U_e(\gamma) + P_{bc}(\alpha,\beta) + P_{de}(\beta,\gamma)$$

Example Cut 2





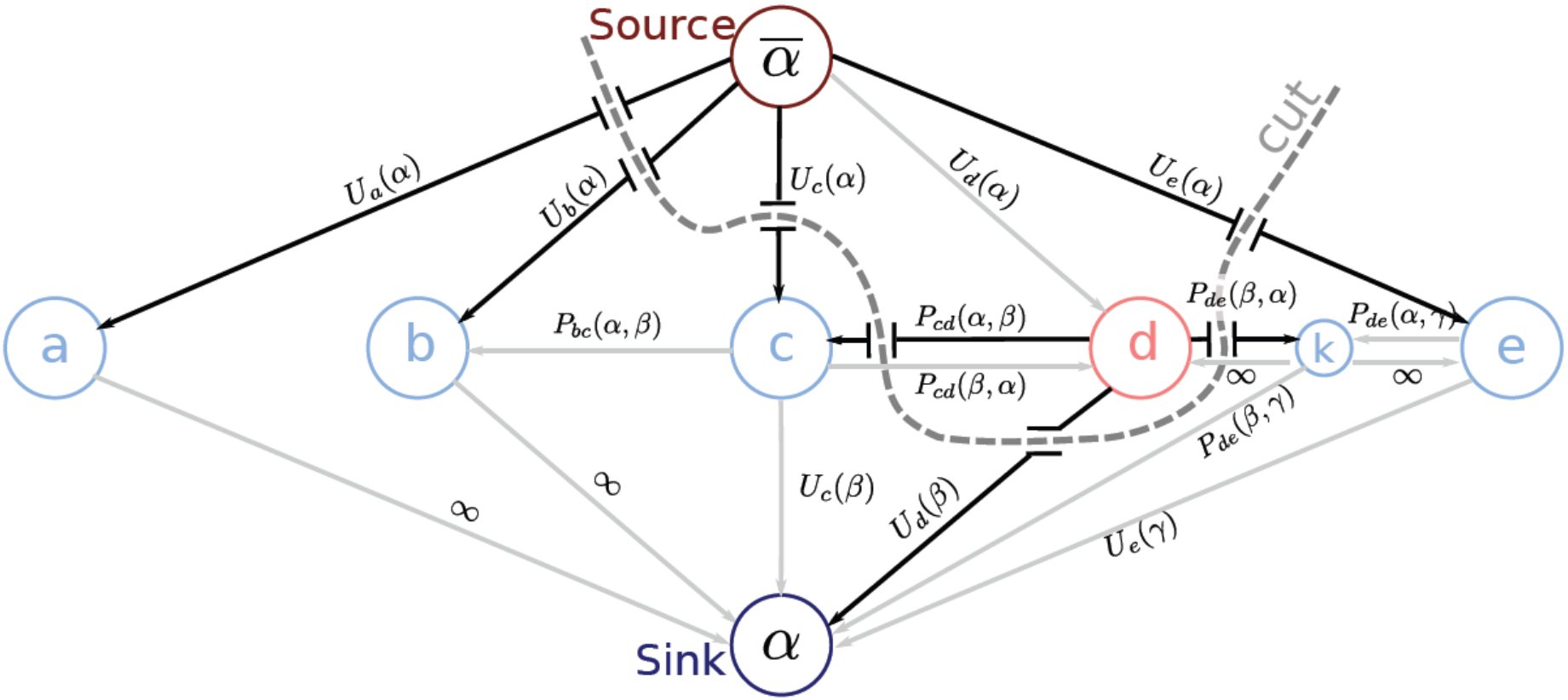
States before

States after

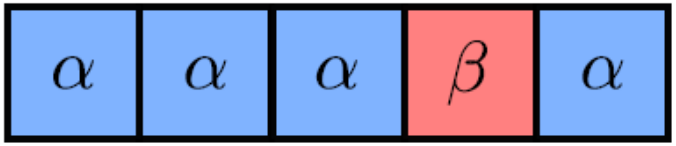
Cost

$$U_a(\alpha) + U_b(\alpha) + U_c(\beta) + U_d(\alpha)$$
$$+U_e(\alpha) + P_{bc}(\alpha,\beta) + P_{cd}(\beta,\alpha)$$

Example Cut 3



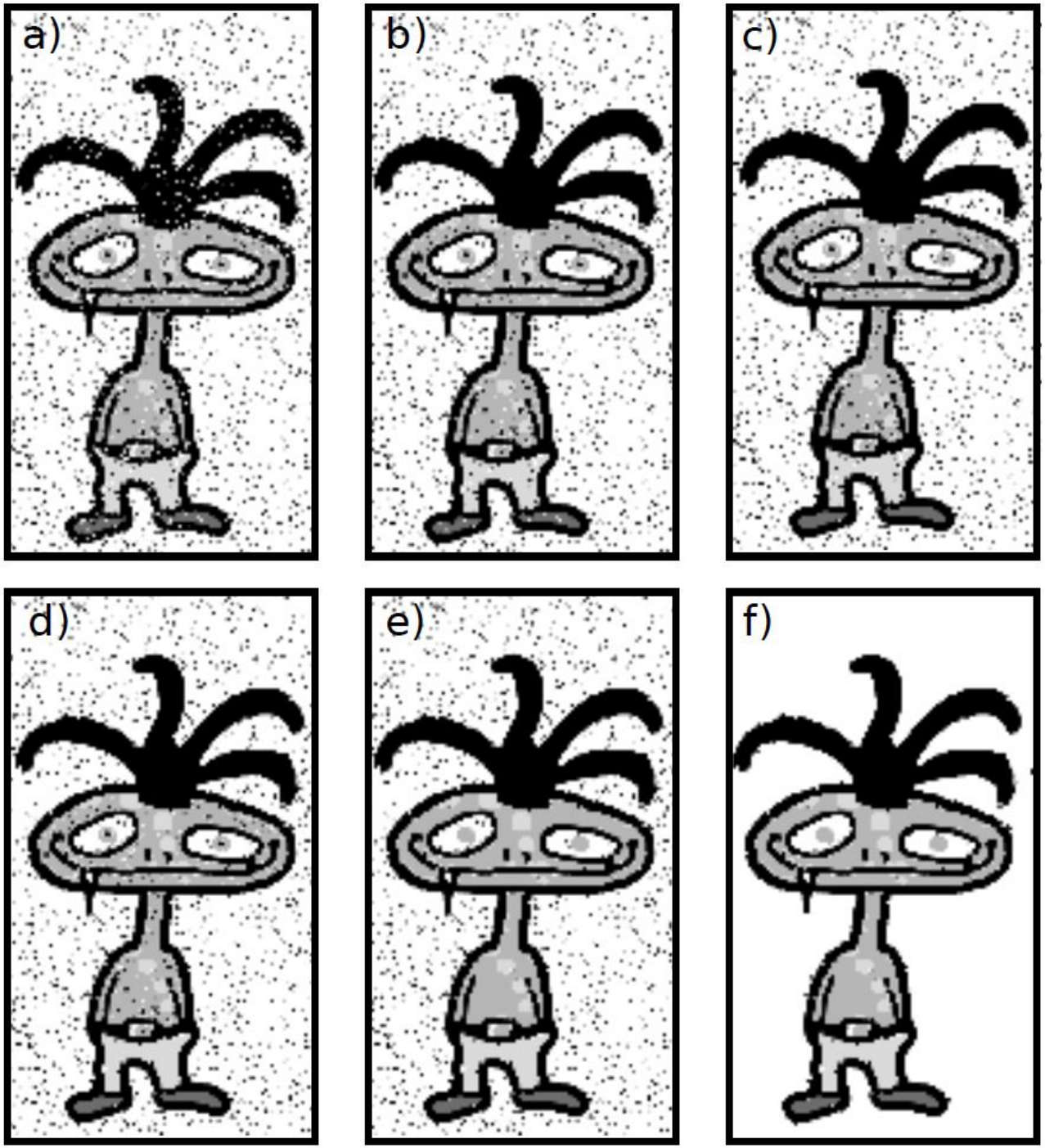
States before



States after

Cost

$$U_a(\alpha) + U_b(\alpha) + U_c(\alpha) + U_d(\beta)$$
$$+U_e(\alpha) + P_{cd}(\alpha,\beta) + P_{de}(\beta,\alpha)$$

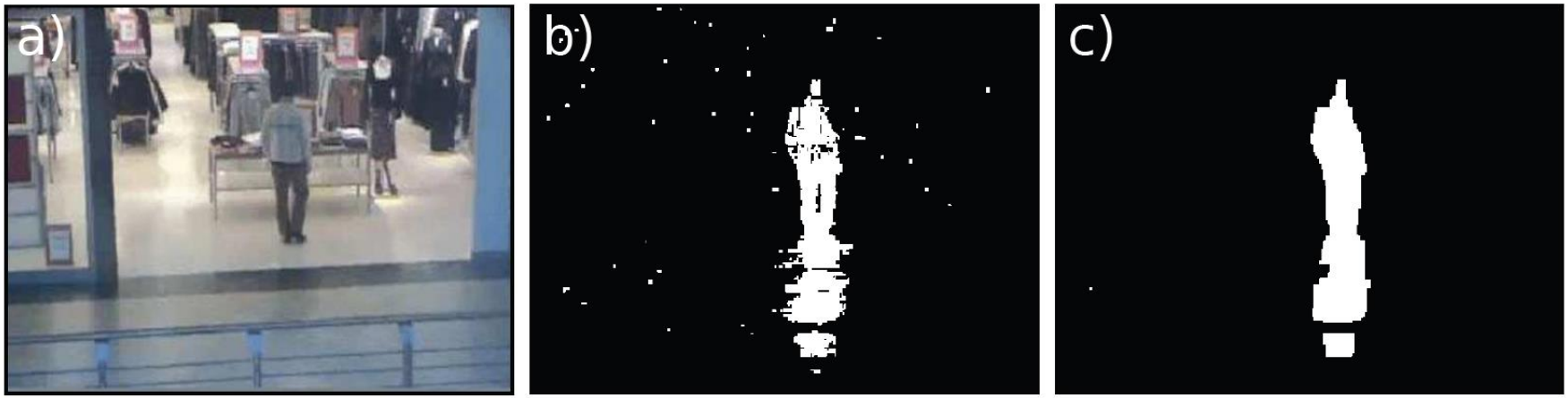


a) Observed image
b) Hair
c) Boots
d) Trousers
e) Skin
f) Background

Applications: Background subtraction

Model background distribution of each pixel by Gaussian

$$Pr(\mathbf{x}_n|w=0) = \mathrm{Norm}_{\mathbf{x}_n}[\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n]$$
$$Pr(\mathbf{x}_n|w=1) = \kappa,$$



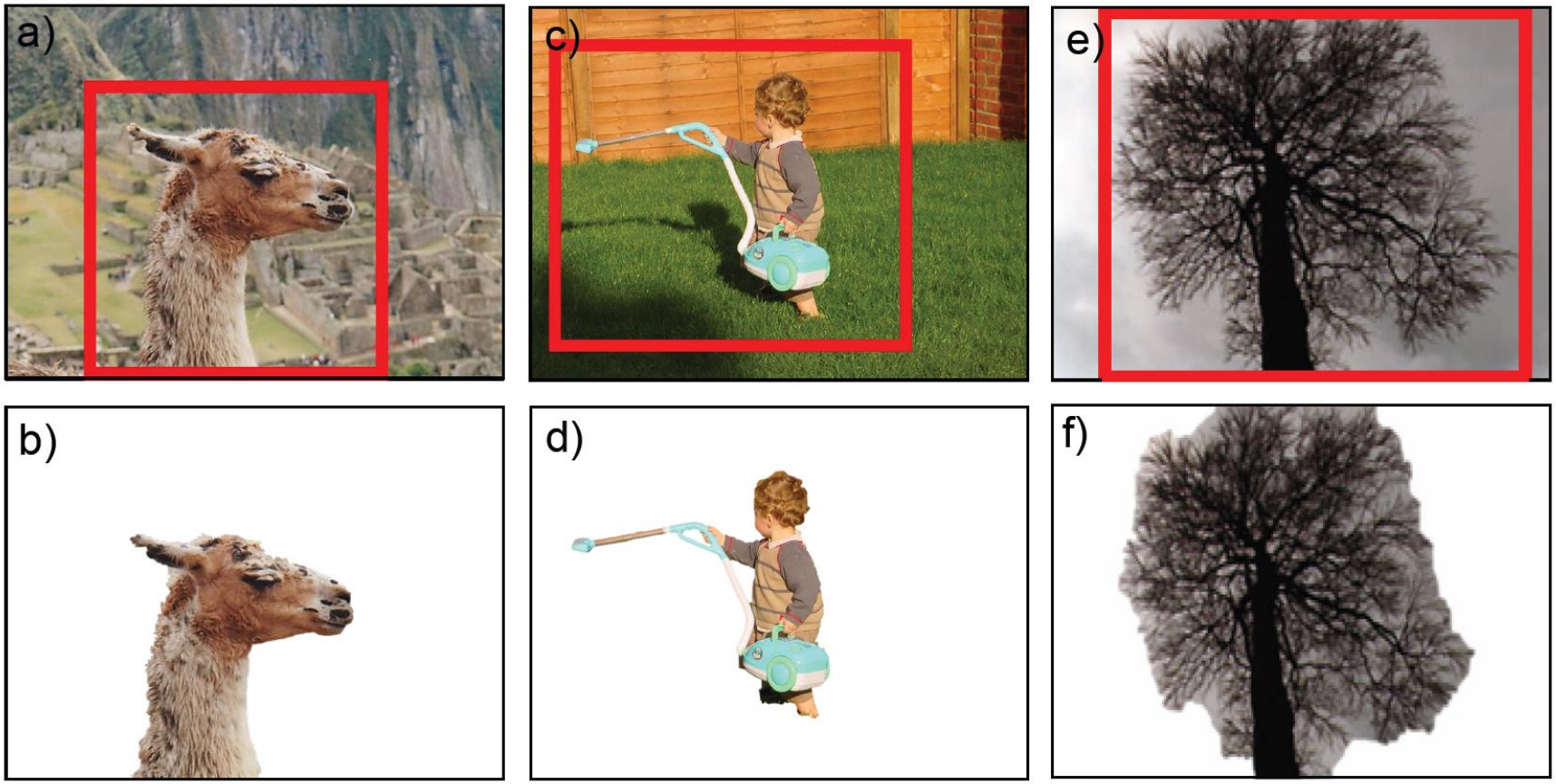
Test image | Pixel-wise classification | MRF

Applications: Interactive Segmentation

UNIVERSITÄT BONN

Grab cut

$$Pr(\mathbf{x}_n|w=j) = \sum_{k=1}^{K} \lambda_{jk} \mathrm{Norm}_{\mathbf{x}_n}[\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}]$$



Applications

UNIVERSITÄT BONN

Stereo vision


UNIVERSITÄT BONN