Graph Cuts

Dr Cian M Scannell, Assistant Professor

Department of Biomedical Engineering, Medical Image Analysis group

Background and motivation

Image segmentation

Goal of segmentation: assign each $p \in P$ to a label $y_p \in L$ such that y is both piecewise smooth and consistent with the observed data.

Energy minimization

Segmentation tasks can be formulated in terms of an energy minimization problem. (You have seen this in 8BB050)

... the process of finding the optimal solution to a problem by defining an "energy function" (also known as a cost function or objective function) that quantifies the "goodness" of a potential solution...

$E(y)$: a function that represents the quality of the segmentation.

So the goal can be achieved by finding the labelling y that minimizes $E(y)$ for a suitable choice of $E(y)$.

Energy minimization

$$E(y) = E_{smooth}(y) + E_{data}(y)$$

Usually:

$$E_{data}(y) = \sum_{p \in P} D_p(y_p)$$

Where D_p measures how appropriate a label is for the pixel p given the observed data.

Energy minimization

$$E(y) = E_{data}(y) + E_{smooth}(y)$$

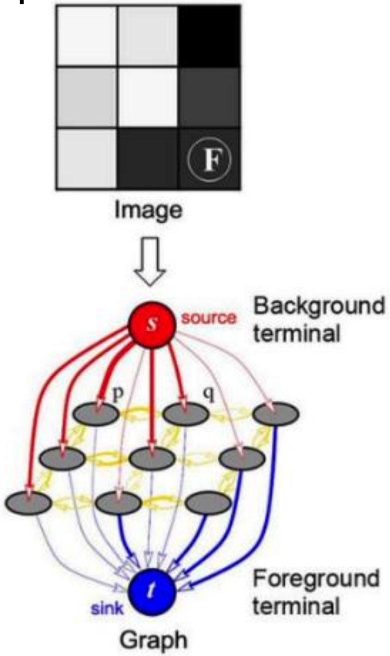
Usually:

$$E_{smooth}(y) = \sum_{\{p,q\} \in N} V_{\{p,q\}}(y_p, y_q)$$

Where $V_{\{p,q\}}$ penalizes differences between the labels of neighbouring pixels.

Graph cuts

Energy minimization problems can be approximated by solving a maximum flow problem in a graph



Aims for today

1) We will derive a cost function in the form:

$$\sum_{p} D_p(y_p) + \sum_{\{p,q\}\in N} V_{\{p,q\}}(y_p, y_q)$$

By formulating the image as a graph

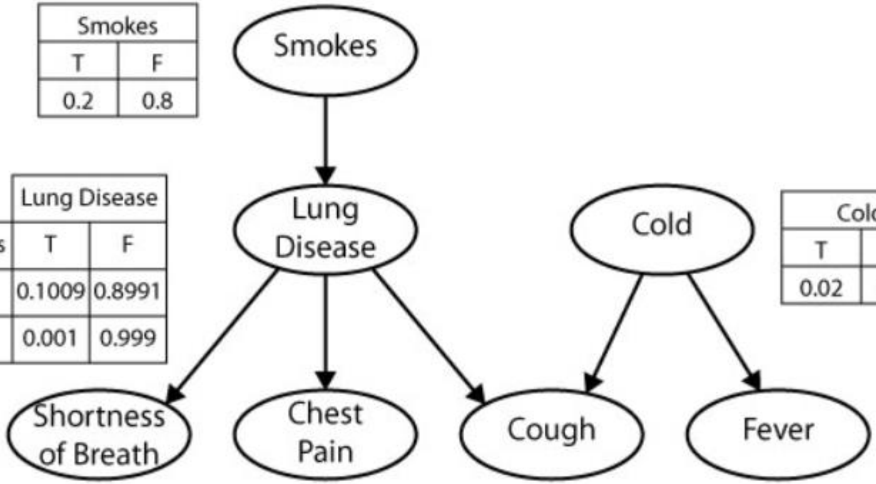
2) We will show that we can optimizing this cost function by solving a min cut-max flow problem on the graph

3) We will learn an algorithm for solving min cut-max flow problems

But first….

Graphs - basics

Probabilistic graphical models



Smokes	
T	F
0.2	0.8

	Lung Disease	
Smokes	T	F
T	0.1009	0.8991
F	0.001	0.999

Cold	
T	F
0.02	0.98

	Shortness of Breath	
Lung Disease	T	F
T	0.208	0.792
F	0.01	0.99

	Chest Pain	
Lung Disease	T	F
T	0.208	0.792
F	0.01	0.99

		Cough	
Lung Disease	Cold	T	F
T	T	0.7525	0.2475
T	F	0.505	0.495
F	T	0.505	0.495
F	F	0.01	0.99

	Fever	
Cold	T	F
T	0.307	0.693
F	0.01	0.99

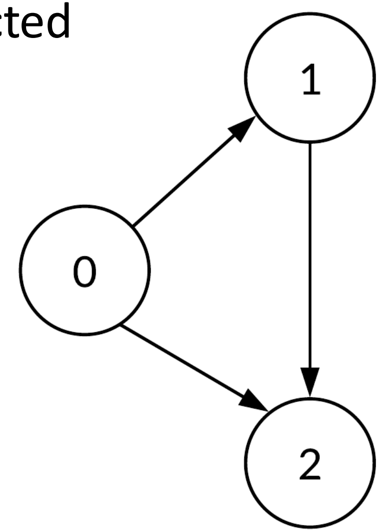
Probabilistic graphical models

$$G = (V, E, w)$$

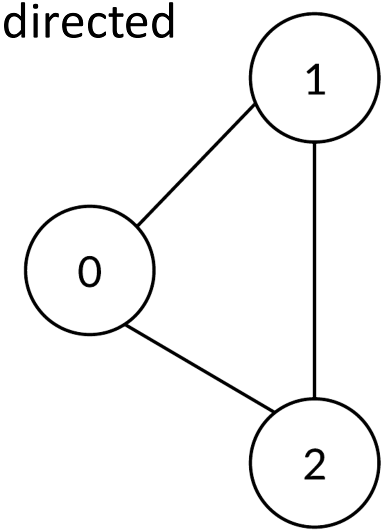
V: Vertices of the graph – index elements of our image grid
E: Edge - express probabilistic relationships between pairs of vertices
w_{pq}: Weights – functions of similarities between vertices p and q

Direction

Directed



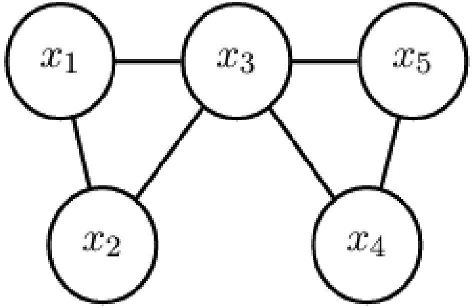
Undirected



Clique

A clique is any fully connected subset of the the graph.
-> a set of nodes where each node is directly connected to every other node in the set.

E.g. $\{x_4\}$,$\{x_1, x_2, x_3\}$, or $\{x_3, x_5\}$. But not $\{x_1, x_3, x_4\}$ or $\{x_2, x_4\}$.



Markov random field (MRF)

A type of probabilistic graphical model represented by an undirected graph

Possesses the Markov property - a node is conditionally independent of all other nodes given its neighbours:

$$P\left(y_p \middle| \boldsymbol{y} \backslash \mathrm{y}_p\right) = P\left(y_p \middle| N_p\right)$$

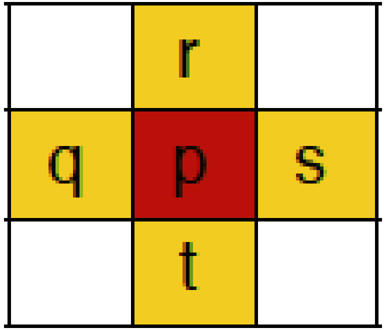
The conditional probability of observing a given variable y_p is independent of all other variables given a neighborhood N_p of p

Representing images as graphs

For images

With images, neighbourhoods consisting of the set of directly neighbouring pixels (above, below, left, right) are often used

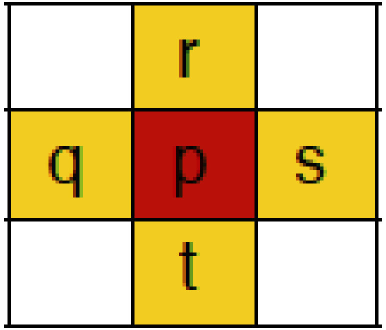
$$E_p = \{\{p,q\},\{p,r\},\{p,s\},\{p,t\}\}$$
$$E_p \subset E$$



Cliques for images

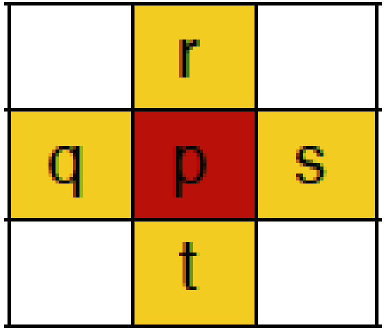
With this neighbourhood system commonly used for images, where each pixel label is dependent on the labels of the pixels above, below, left, right.

The maximum clique size is 2



Markov property for images

So, the Markov property says that the label y_p of a particular pixel p is independent of all other pixel labels given the neighbouring pixels N_p



Hammersley-Clifford theorem

Any positive probability distribution that satisfies the Markov properties with respect to an undirected graph can be represented as a Gibbs distribution with potentials defined over the cliques of that graph

➡ Given an MRF it is possible to write the joint distribution over all variables y as a Gibbs distribution

Gibbs distribution

The Gibbs distribution form is:

$$p(y) = (1/Z) * exp(-E(y))$$

Where Z is the normalization constant, $E(y)$ is the energy function

Gibbs factorization

In the context of probabilistic graphical models, Gibbs distributions can be factorized using undirected graphical models, such as Markov Random Fields, using the conditional independencies between variables

Gibbs distributions have factorization properties over cliques

For images

The energy can be decomposed into sum of potential functions over cliques, leading to a sum over the pairwise cliques

$$E(y) = \sum_{c \in C} V_c(y)$$

$$= \sum_{\{p,q\} \in N} V_{\{p,q\}}(y_p, y_q)$$

MAP

But what we are actually interested in is not $p(y)$. To solve the segmentation problem, we want the *maximum a posteriori* (MAP) estimate of the labels given the data:

$$\hat{y} = \mathrm{argmax}_y \; p(y|x)$$

$$= \mathrm{argmax}_y \;\; p(x|y)p(y)$$

$$= \mathrm{argmax}_y \prod_p p(x_p|y_p)p(y_p)$$

$$= \mathrm{argmax}_y \sum_p \log(p(x_p|y_p)) + \log p(y_p)$$

$$= \mathrm{argmax}_y \sum_p \log(p(x_p|y_p)) - \sum_{\{p,q\}\in N} V_{\{p,q\}}(y_p, y_q)$$

$$= \mathrm{argmin}_y - \sum_p \log(p(x_p|y_p)) + \sum_{\{p,q\}\in N} V_{\{p,q\}}(y_p, y_q)$$

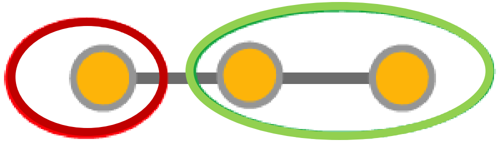
$$= \mathrm{argmin}_y \sum_p D_p(y_p) + \sum_{\{p,q\}\in N} V_{\{p,q\}}(y_p, y_q)$$

This is the form of the cost function we aimed to derive from the start of the lecture

Combination of internal (unary) and external (pairwise) potentials

D_p are the **unary** potentials
$V_{\{p,q\}}$ are the **pairwise** potentials

Think of them as a data term and smoothness prior, respectively



From earlier

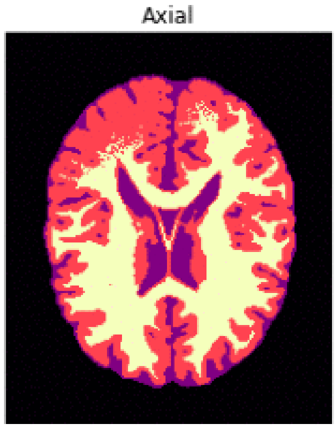
D_p: measures how appropriate a label is for the pixel p given the observed data.

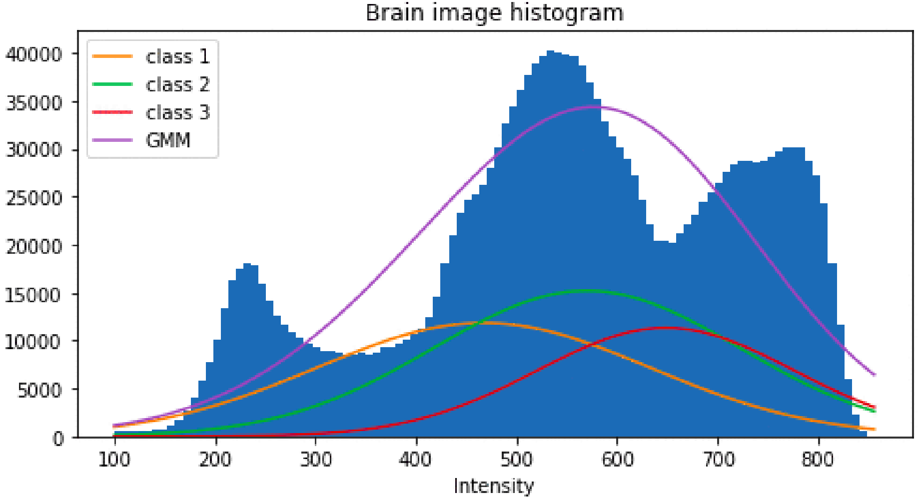
$V_{\{p,q\}}$: penalizes differences between the labels of neighbouring pixels.

Data term

Unary function based only on image information at the voxel.

Can be (for example) an intensity model based on a Gaussian mixture model





Smoothness prior

Pairwise penalty function based on boundary properties between voxels

$$V_{\{p,q\}} = \exp(\frac{-\left(I_p - I_q\right)^2}{\sigma^2}) \,/\, \mathrm{dist}(p,q) \qquad \text{if } y_p \neq y_q$$

Penalizes discontinuities between similar intensities if $\left|I_p - I_q\right| < \sigma$

If voxels are very different, $\left|I_p - I_q\right| > \sigma$, then the penalty is small

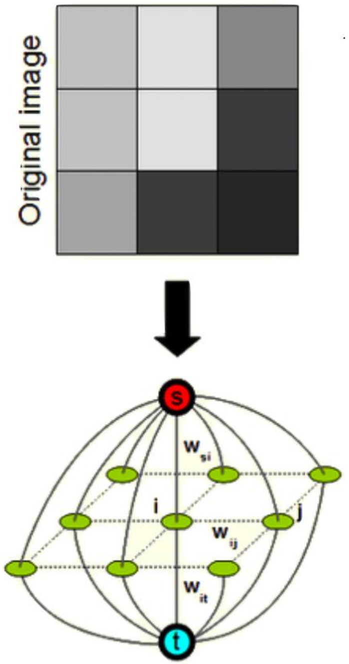
Graph cuts

We can optimize this cost function by solving a min-cut or max-flow problem on the graph

Graph cuts

MRF-based energy function minimization solved as a min-cut or max-flow problem:

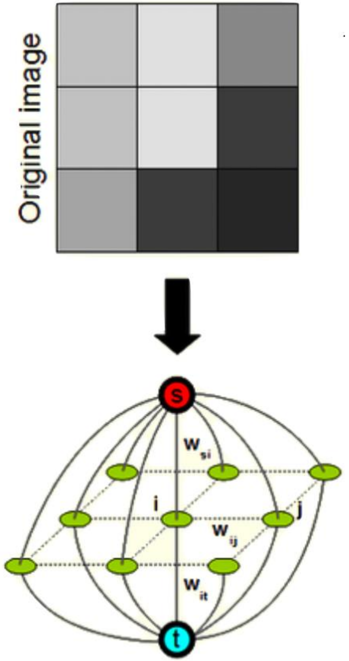
- Image voxels represented as nodes in a graph
- Source and Sink nodes represent foreground and background



Graph cuts

We seek a ‘cut’ that separates the voxels in an optimal way

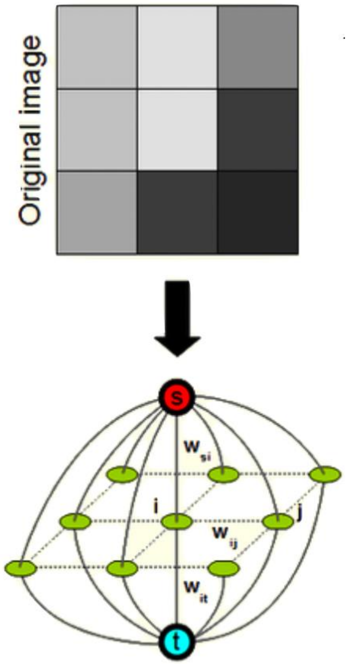
- Maximises the difference in intensities between classes
- Minimises the total weight of edges severed in the cut



Graph cuts

Edge weights:

- Pixel to source/sink: Defined by the data term
- Pixel to pixel : Defined by smoothness term



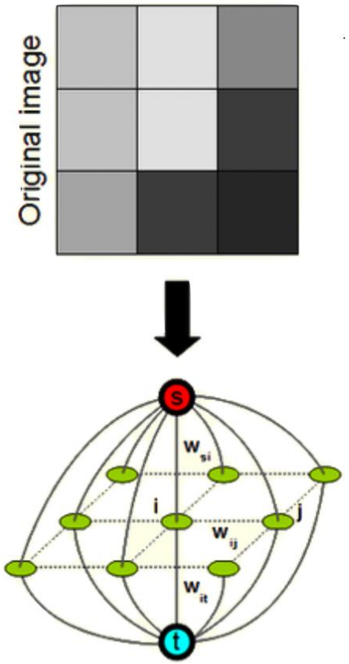
Graph cuts

$$E(\boldsymbol{y}) = \sum_{p} D_p(y_p) + \sum_{\{p,q\}\in N} V_{\{p,q\}}(y_p, y_q)$$

$$w_{pq} = V_{\{p,q\}}(y_s, y_t)$$

$$w_{ps} = D_p(y_s)$$

$$w_{qt} = D_q(y_t)$$

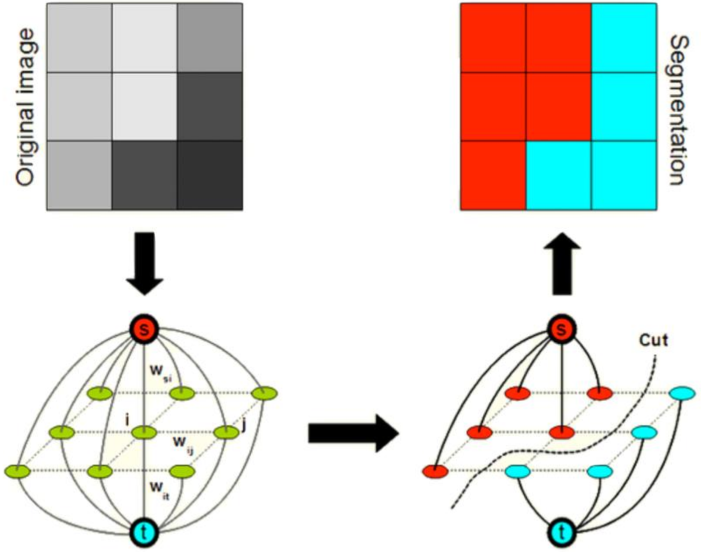


Graph cuts

The goal is to "cut" the graph into segments that represent different objects or regions.

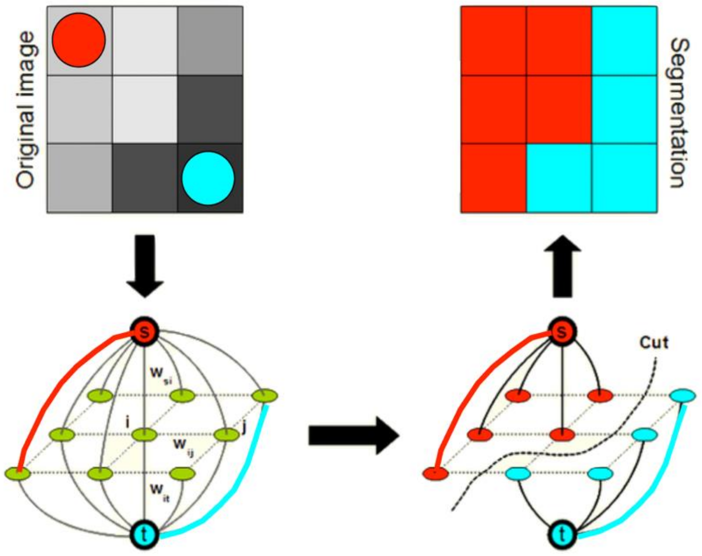
The disconnected subgraphs made by the cut will be each associated to the same label

Graph cuts



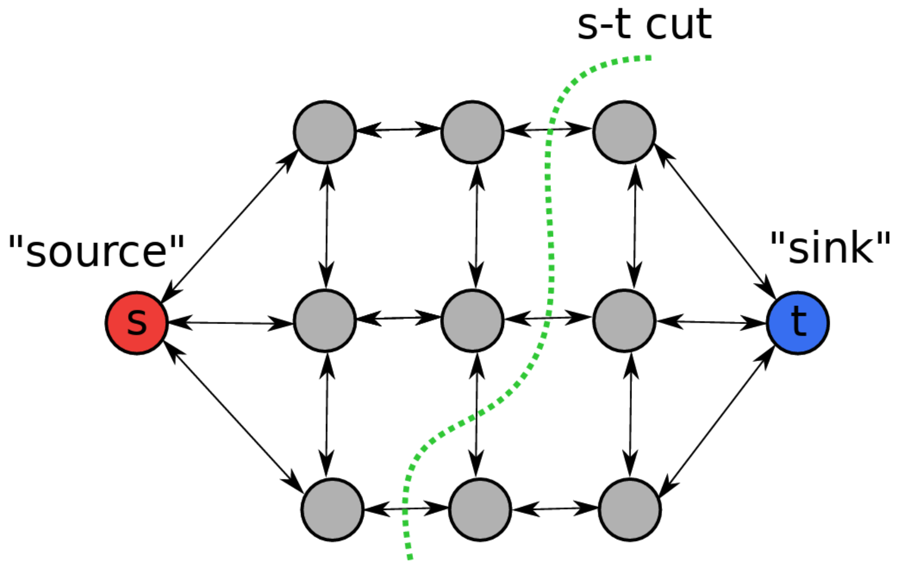
Graph cuts

Constrained forced by adding links to source/sink with infinite weight



Algorithm

We want the cut to find the smallest total weight of the edges which if removed would disconnect the source from the sink

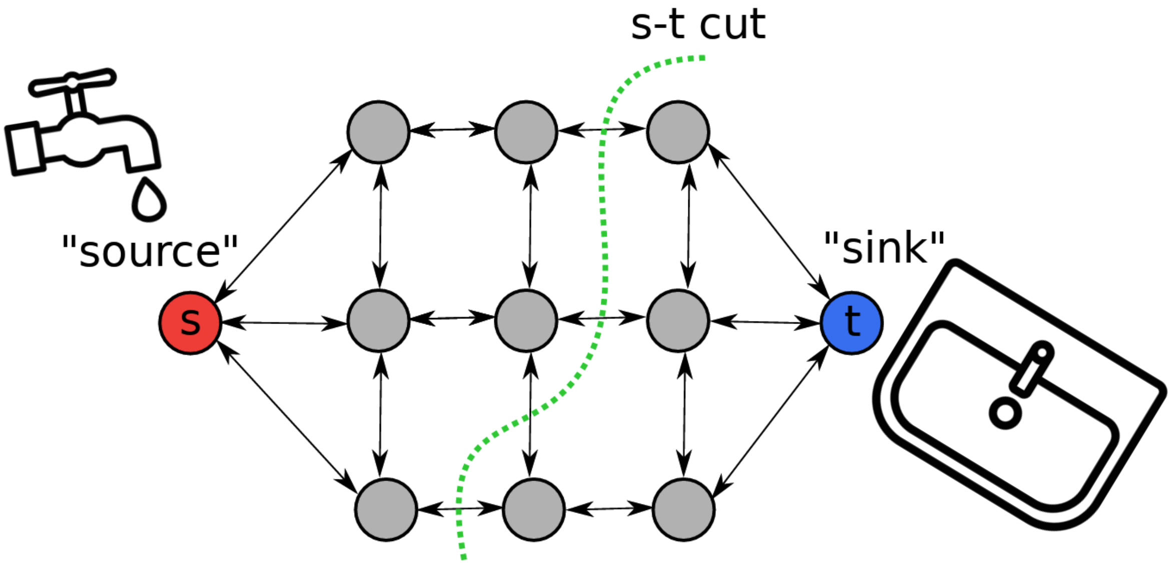


Algorithm

The min-cut algorithm then finds the lowest energy way to partition the graph (and thus the image pixels) into two segments (e.g., foreground and background) by severing the "cheapest" set of edges that separate the source from the sink.

Algorithm

We want the cut to find the smallest total weight of the edges which if removed would disconnect the source from the sink



Flow network

Source: has only outgoing edges

Sink: has only incoming edges

Capacity: each edge has a non-negative capacity – the maximum amount of flow it can carry

A flow network is directed (from source to sink)

Maximum flow

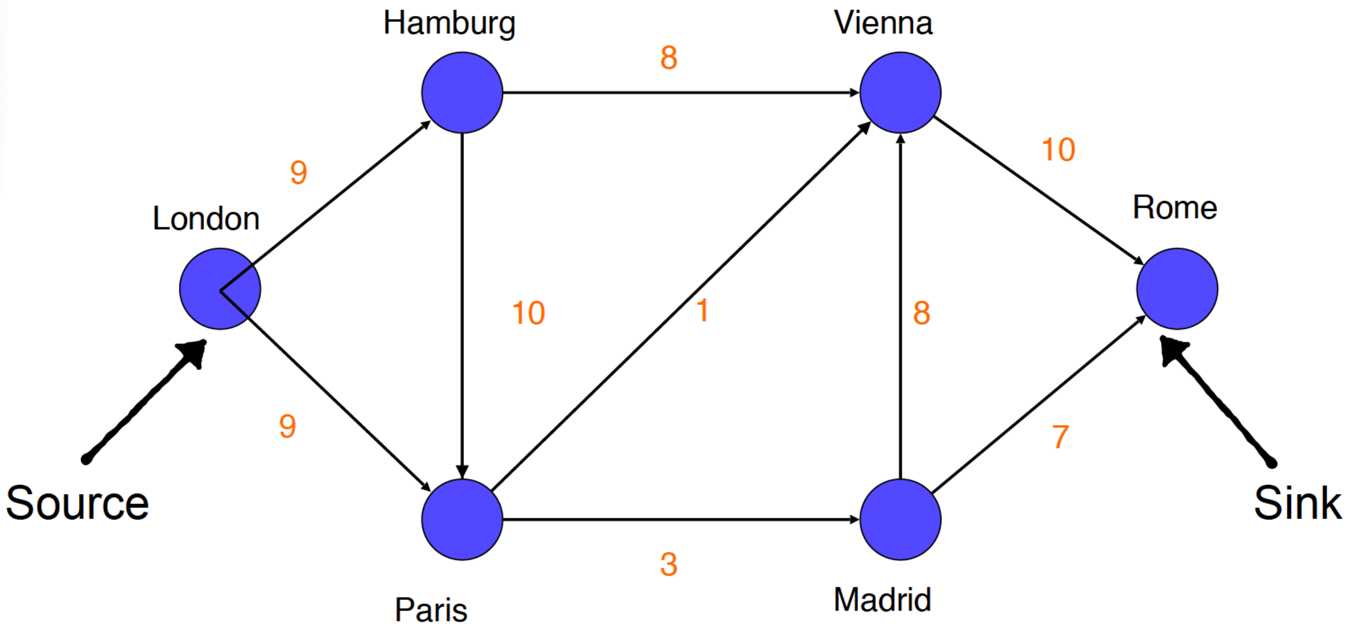
The *maximum flow* problem is to maximise the total amount of flow from source to sink (with each edge limited by its capacity)

Ford-Fulkerson theorem

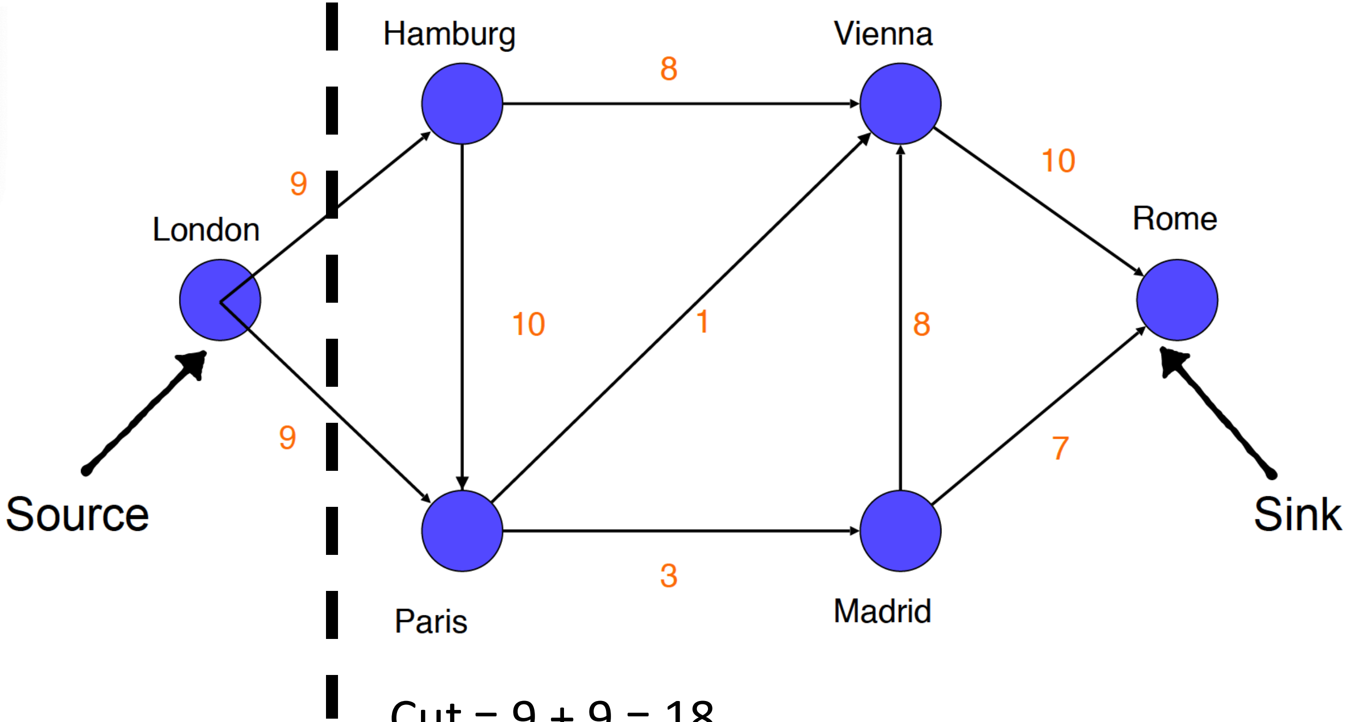
The maximum value of a s-t flow on G is equal to the minimum capacity of an s-t cut.

So, we can compute minimum s-t cuts by computing maximum flow.

Ford-Fulkerson method by example

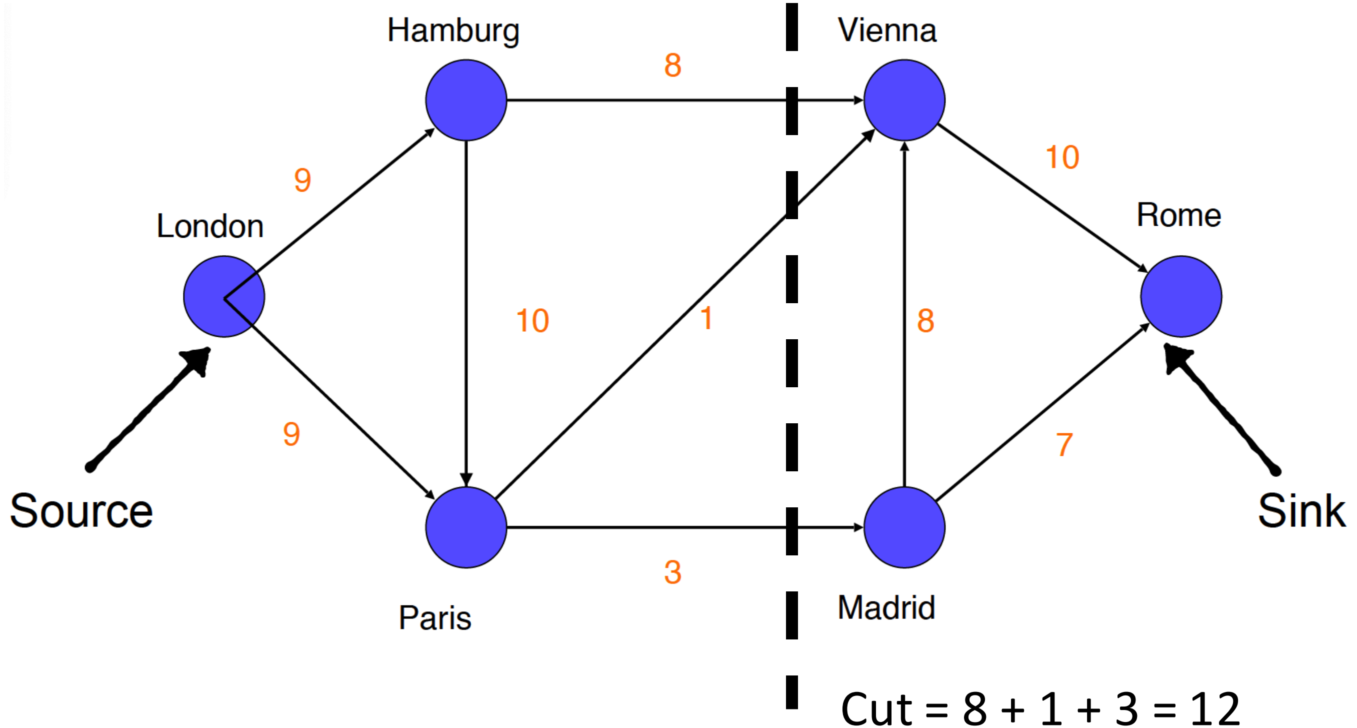


Ford-Fulkerson method by example



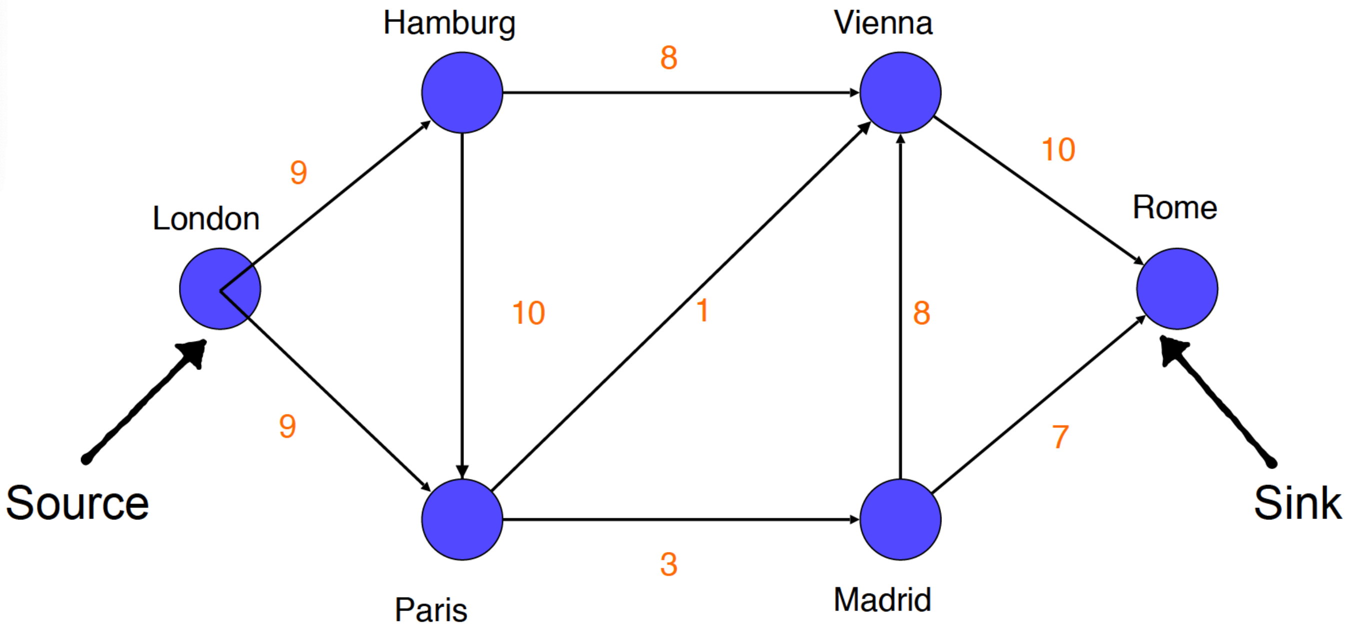
Cut = 9 + 9 = 18

Ford-Fulkerson method by example



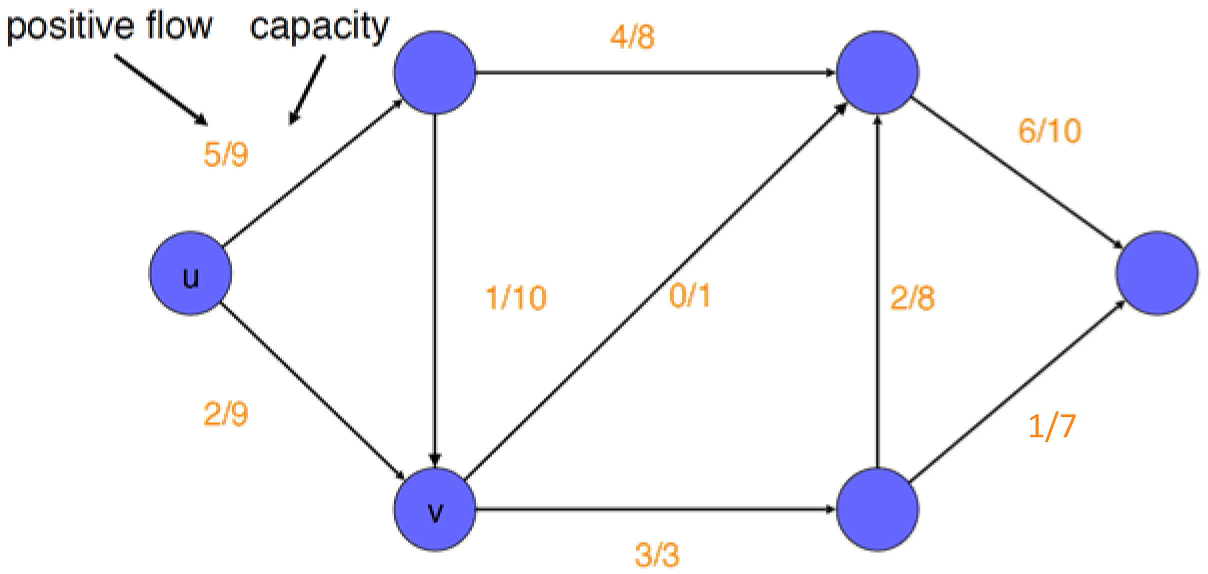
Cut = 8 + 1 + 3 = 12

Max-flow problem



Assign flow to edges so as to: Equalize inflow and outflow at every intermediate vertex and maximize flow sent from s to t.

Flow example

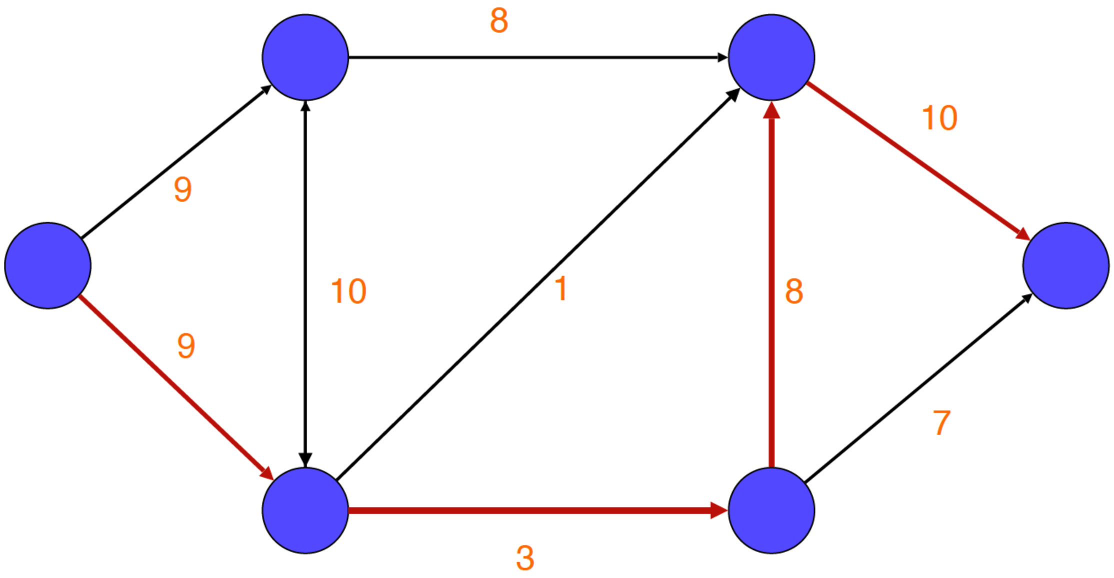


Total flow s to t: 7

Assign flow to edges so as to: Equalize inflow and outflow at every intermediate vertex and maximize flow sent from s to t.

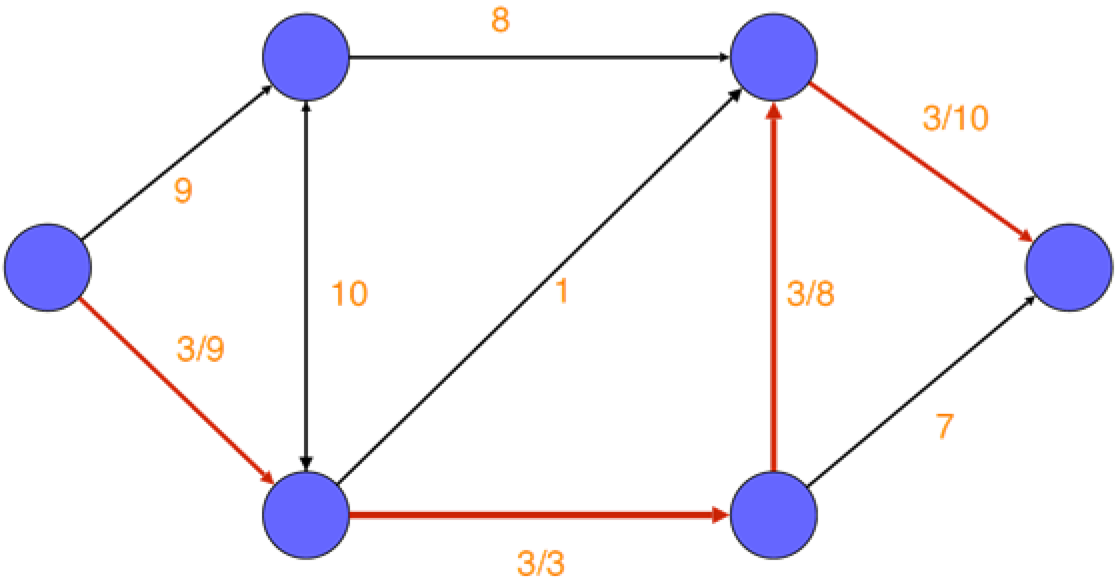
Max-flow Min-cut

Start by assigning flows:



Max-flow Min-cut

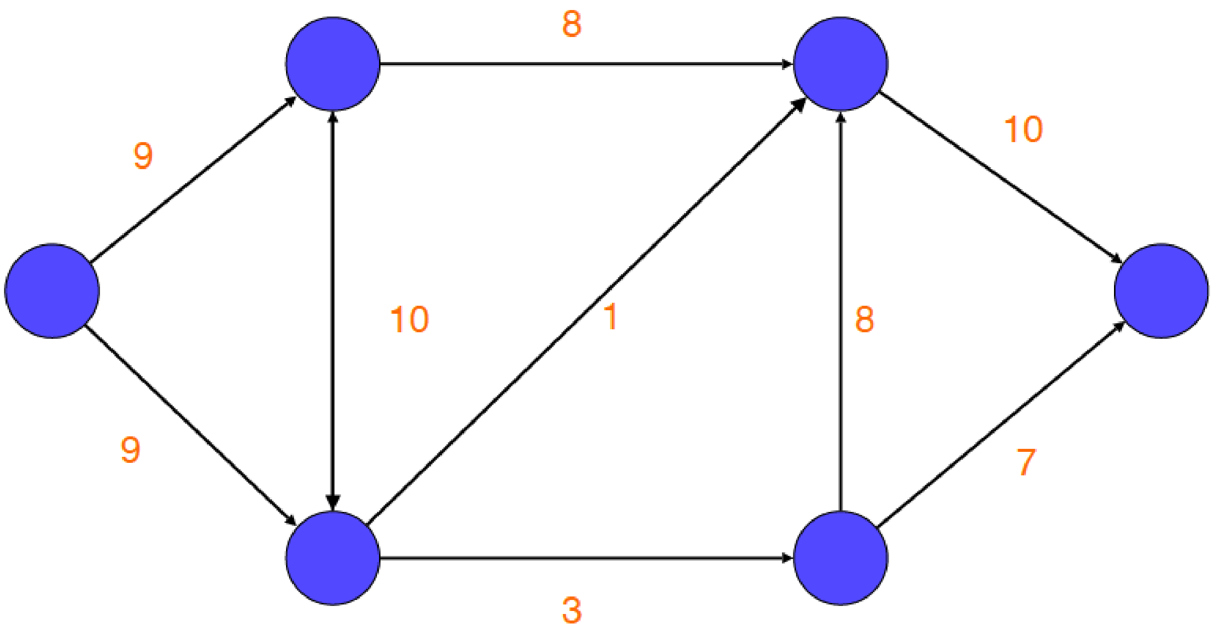
Flow through any path is limited by the strength of the lowest capacity edge



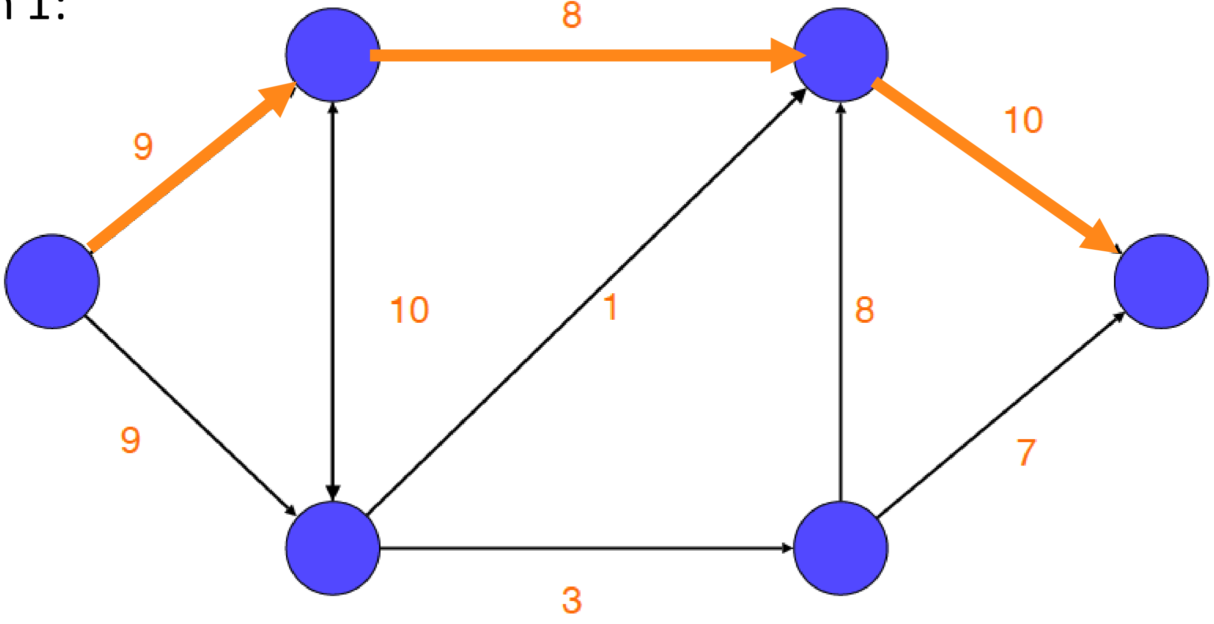
IMAG/e TU/e

Algorithm

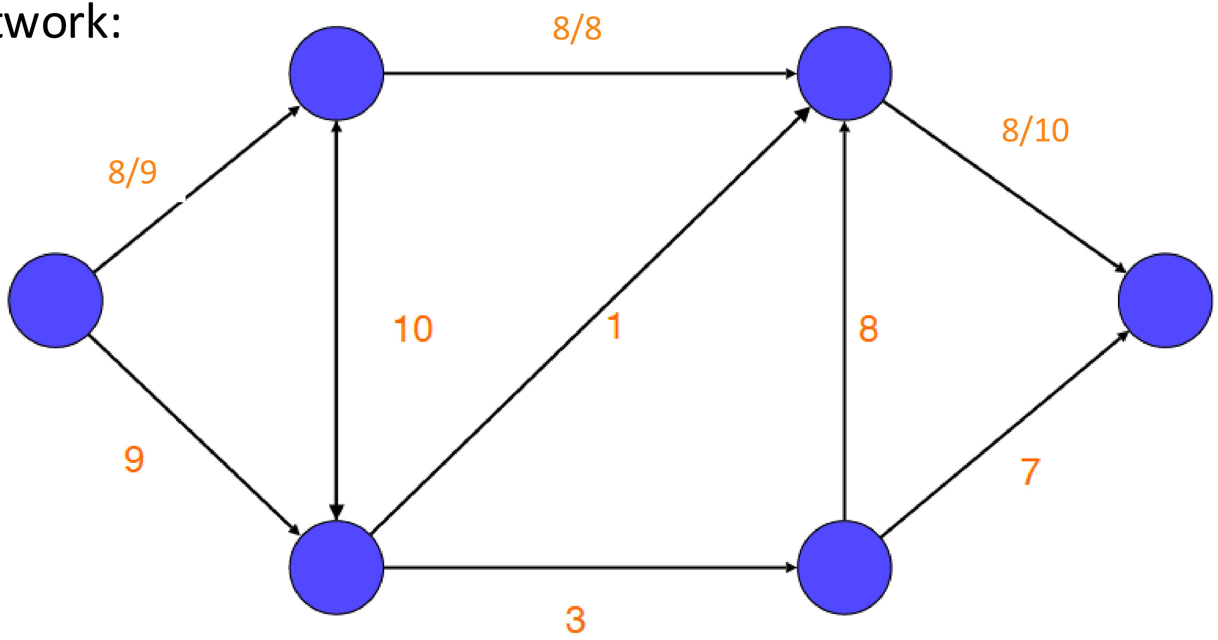
1. Create flow network (G):
   - Start by assigning zero flow to all edges
2. Generate a residual graph
   - Only containing edges that can have more flow
3. Augment paths
   - Find more example paths
   - Identify bottleneck residual capacity
   - Subtract this flow capacity from all edges in that path
   - Repeat until no more augmenting paths can be found


8
9
10
10
1
8
9
7
3

Trial path 1:

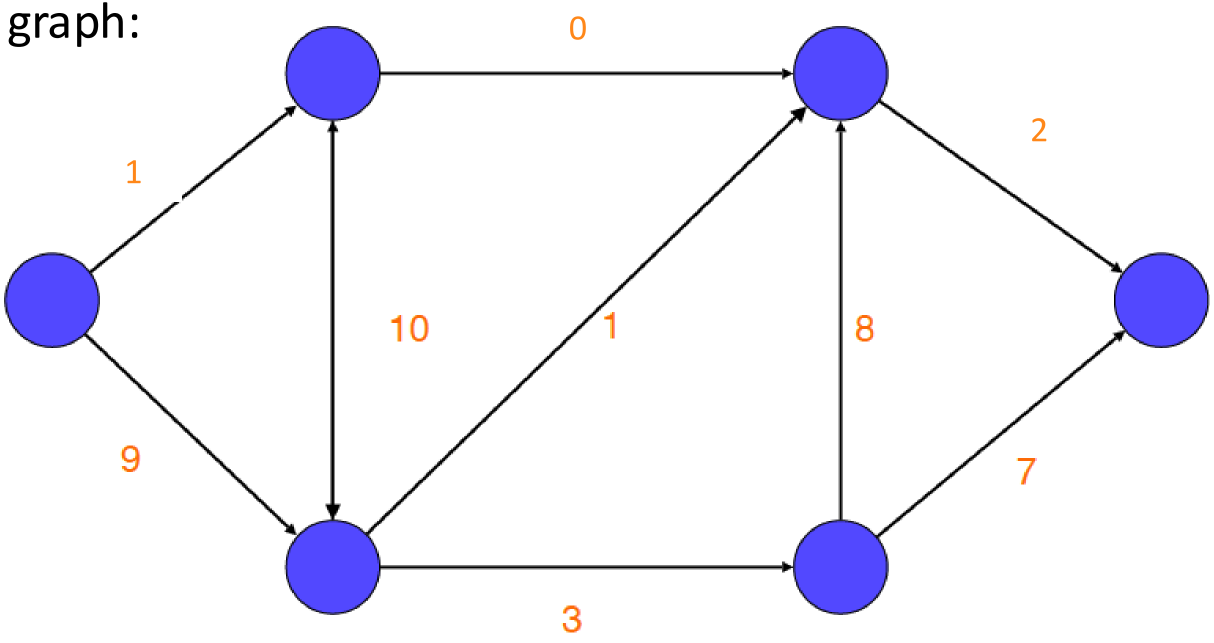


Flow network:



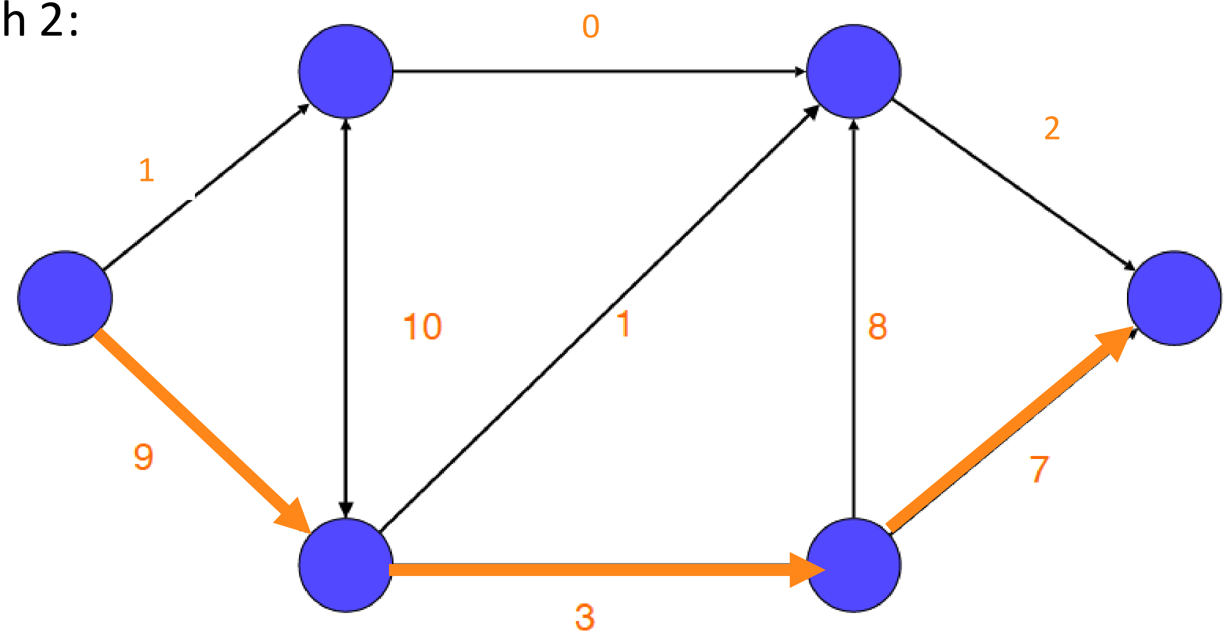
Total flow s to t: 8

Residual graph:



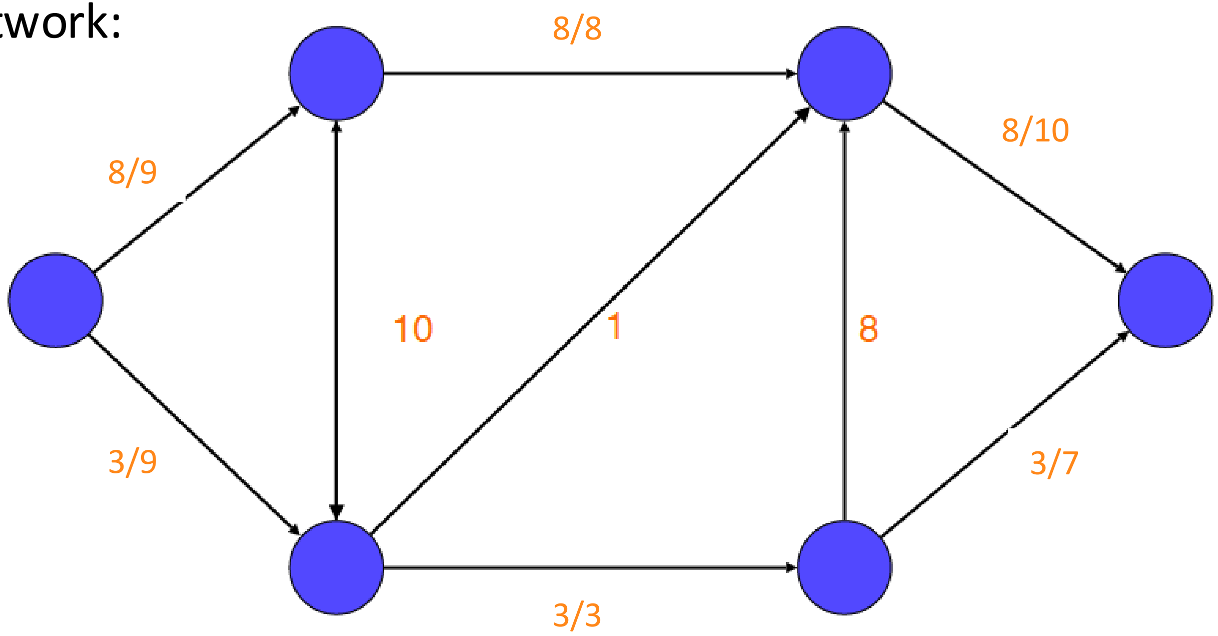
Total flow s to t: 8

Trial path 2:



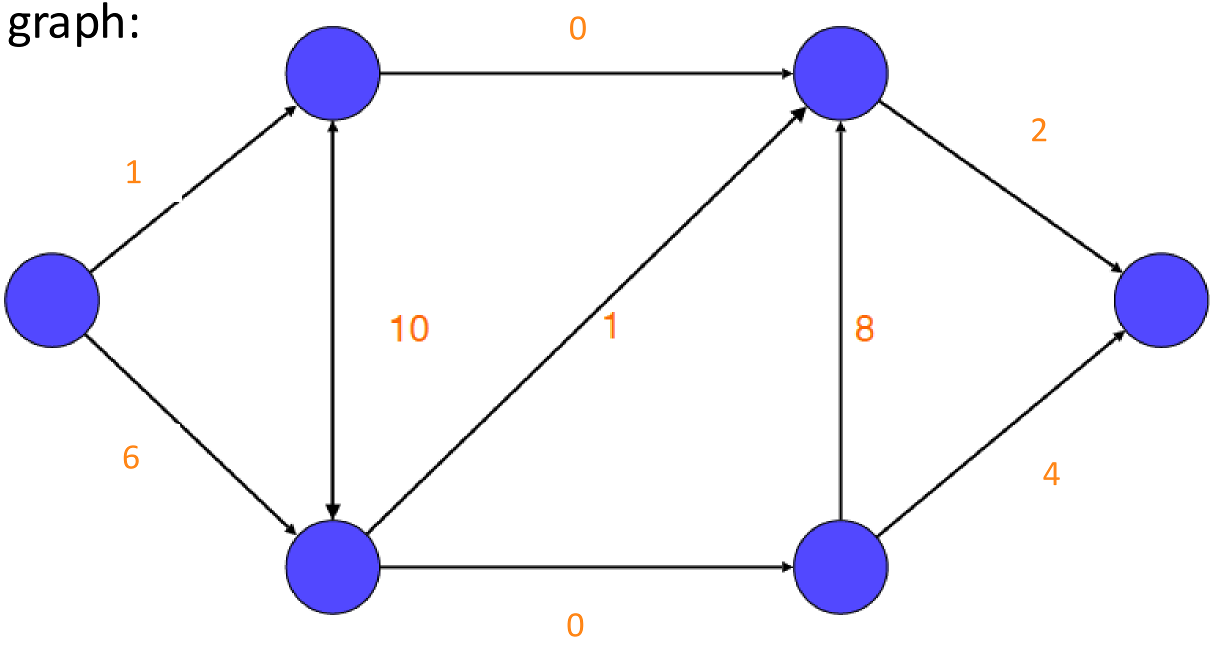
Total flow s to t: 8

Flow network:



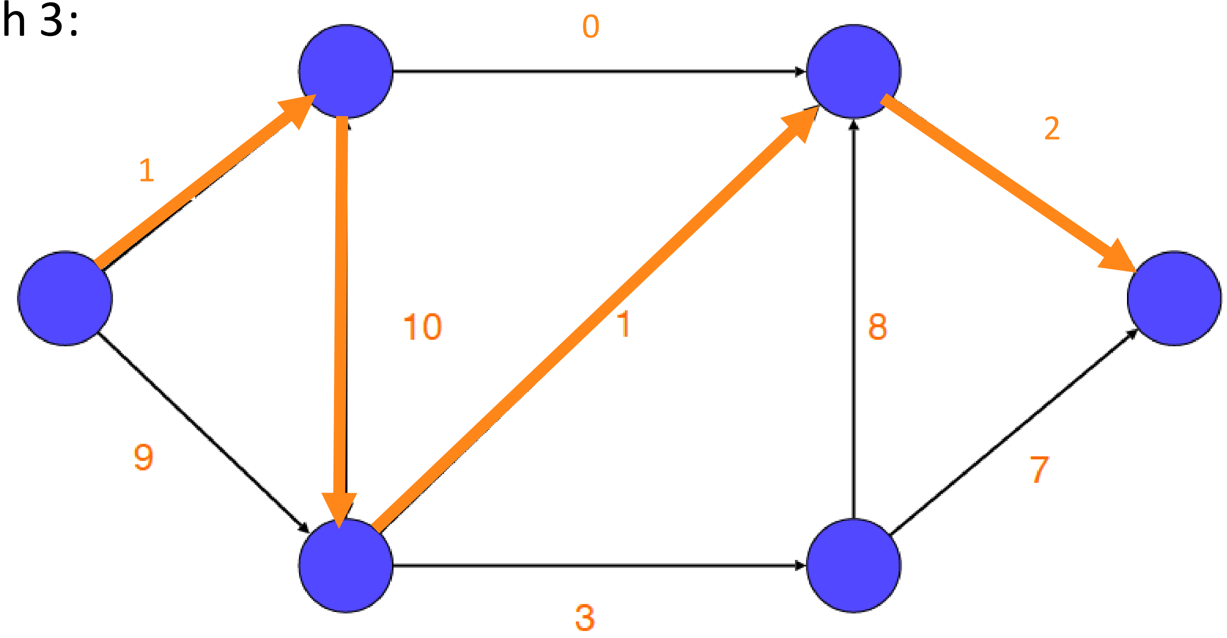
Total flow s to t: 8+3

Residual graph:



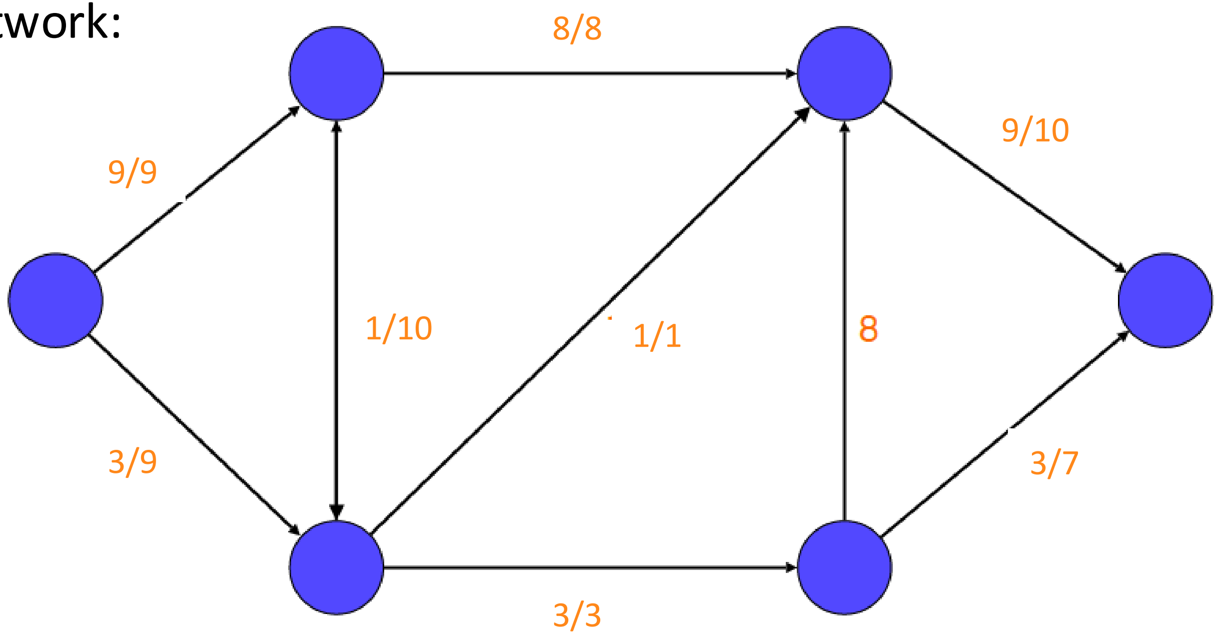
Total flow s to t: 8+3

Trial path 3:



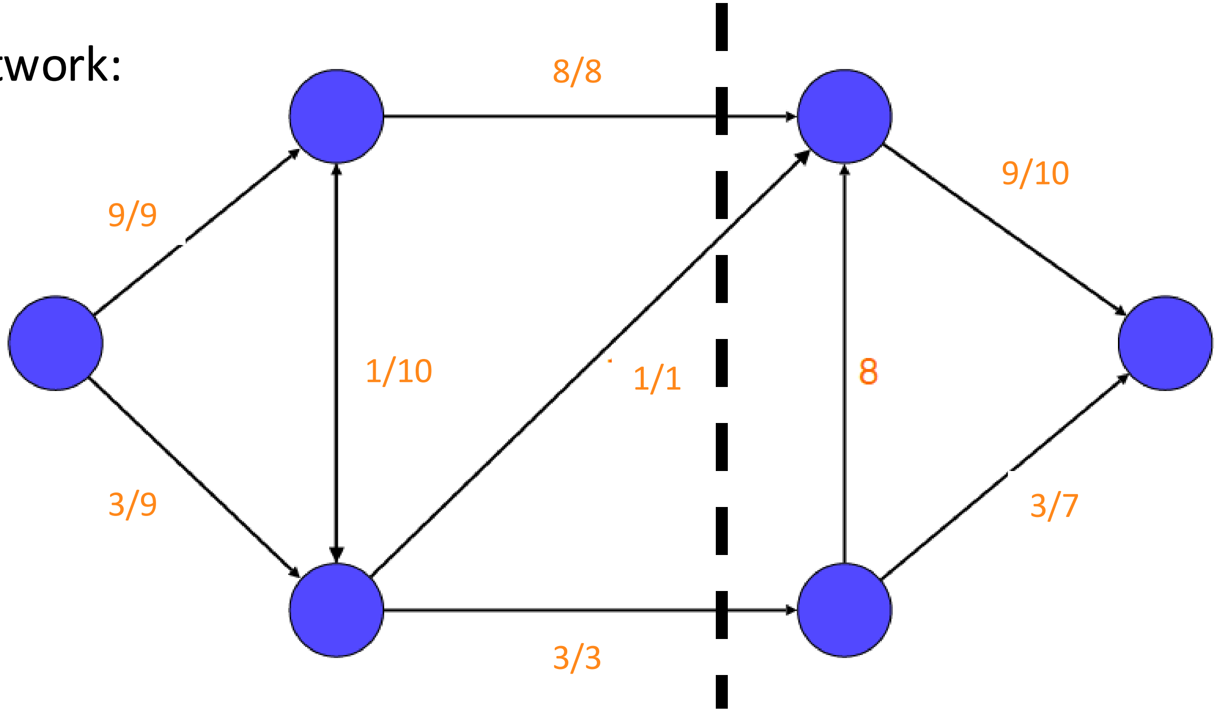
Total flow s to t: 8+3

Flow network:



Total flow s to t: 8+3+1

Flow network:



Max flow: 8+3+1
= min cut

Graph cuts: Properties

Advantages:

– Largely unsupervised

Can be initialised by a few example foreground and background samples

– Highly generalisable

Data likelihood can be anything

Disadvantages:

– Iterates over binary label combinations

compute time can explode for large label sets

Example questions

1) Assume that a 2D image of 3 x 3 pixels is being segmented using graph-cuts. Sketch the graphical representation of the segmentation problem.

2) Define the energy function that is minimised by graph-cuts. Explain the purpose of the different terms in the energy function and explain how the edges in your graphical representation in 2a) are related to the energy function.

See additional example (including backward paths):
https://youtu.be/Tl90tNtKvxs?t=40

Questions

c.m.scannell@tue.nl