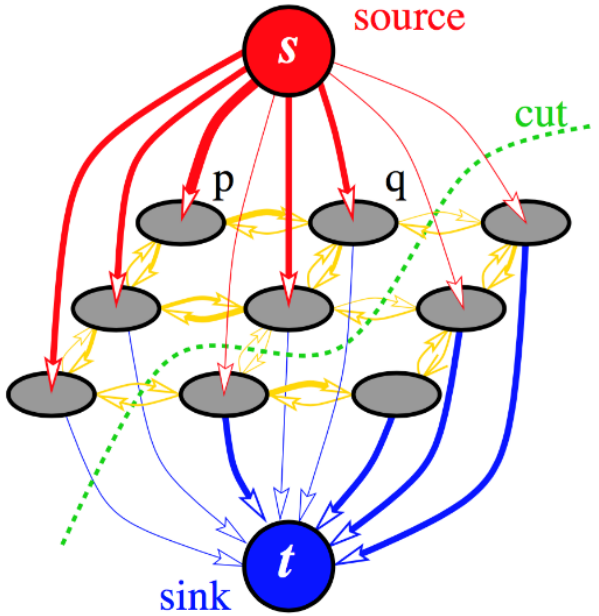
Graph-cuts



Dr. Martin Oswald

Computer Vision and Geometry Group

Overview Graph-cuts

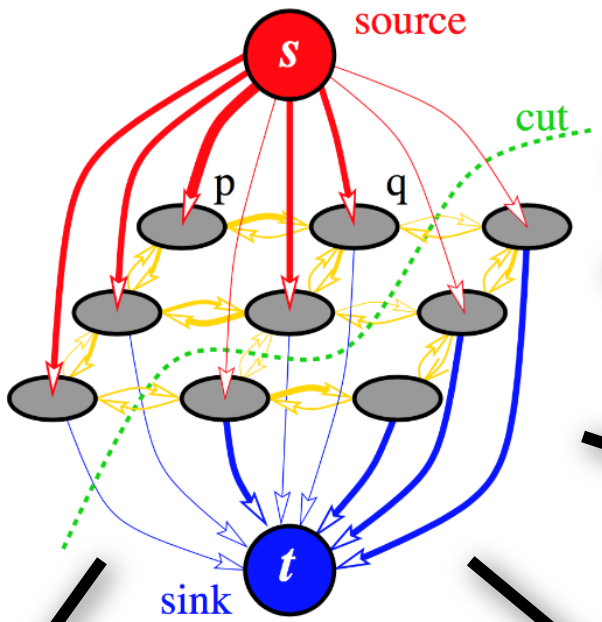


Image Segmentation

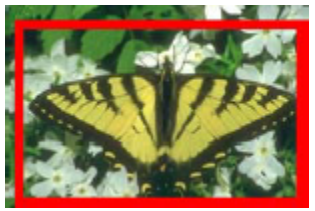
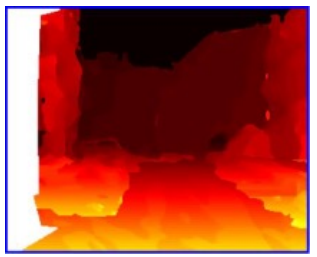
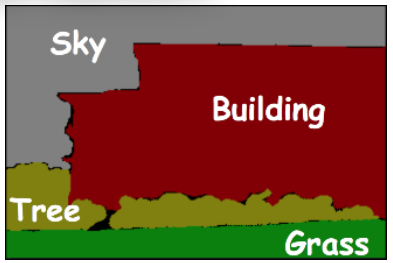


Image Denoising

Depth Estimation



Object Segmentation



3D Reconstruction

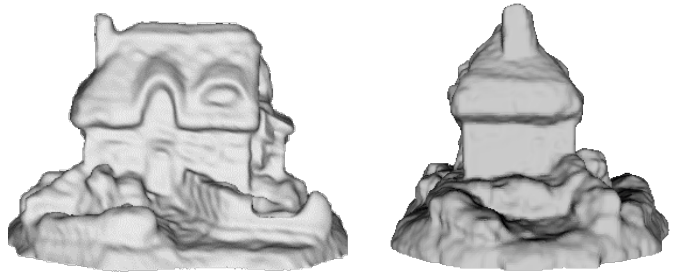


Image Labeling Problems

Assign a label to each image pixel

Geometry Estimation

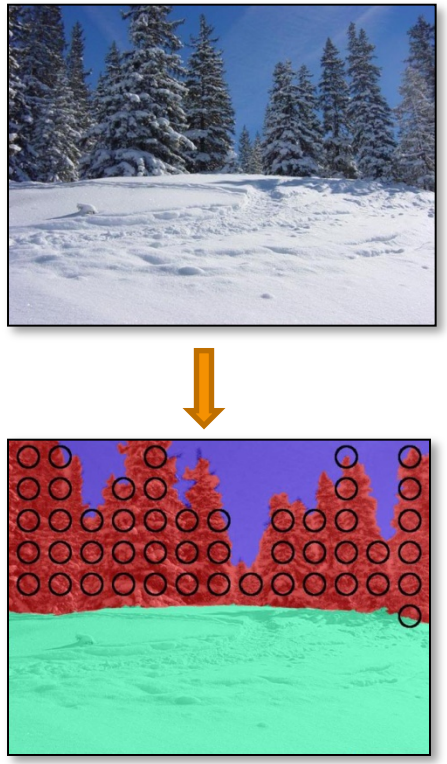
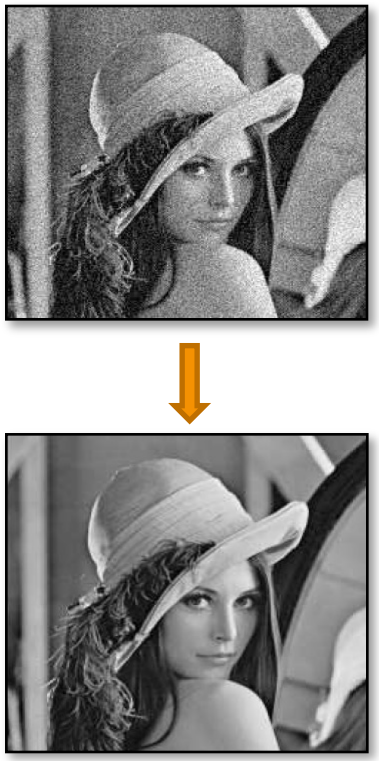
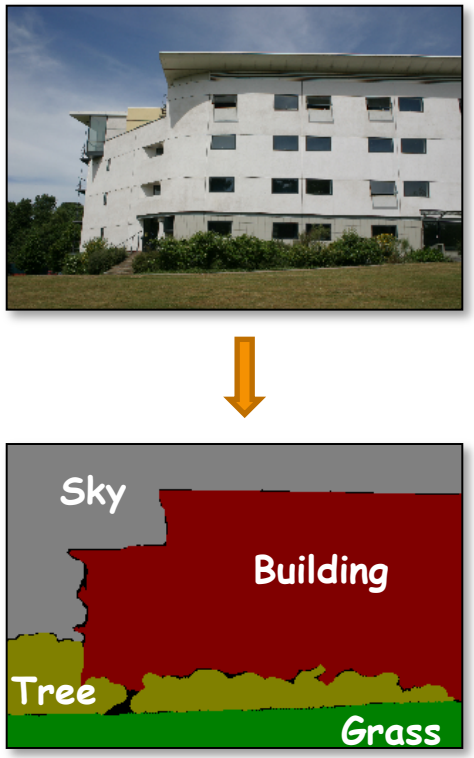


Image Denoising



Object Segmentation

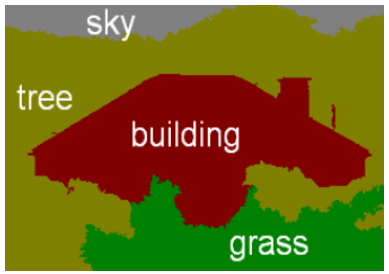


Depth Estimation

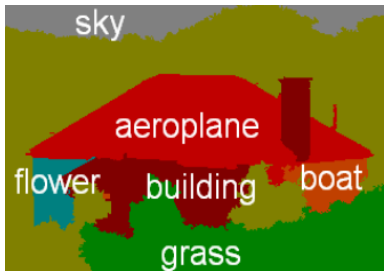
[tutorial slides from Ľubor Ladický]

Image Labeling Problems

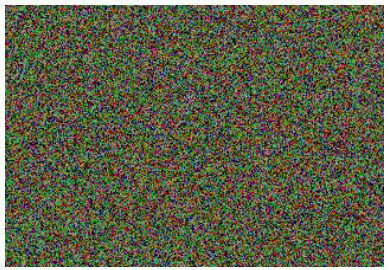
- Labelings are highly structured



Possible labelling



Unprobable labelling

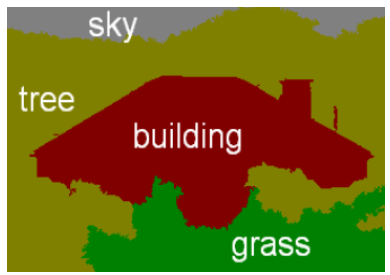


Impossible labelling

[tutorial slides from Ľubor Ladický]

Image Labeling Problems

- Labelings are highly structured
- Labels highly correlated with very complex dependencies



- Neighbouring pixels tend to take the same label
- Low number of connected components
- Classes present may be seen in one image
- Geometric / Location consistency
- Planarity in depth estimation
- … many others (task dependent)

[tutorial slides from Ľubor Ladický]

Image Labeling Problems

- Labelings are highly structured
- Labels highly correlated with very complex dependencies
- Independent label estimation too hard

[tutorial slides from Ľubor Ladický]

Image Labeling Problems

- Labelings are highly structured
- Labels highly correlated with very complex dependencies
- Independent label estimation too hard
- Whole labeling should be formulated as one optimization problem

[tutorial slides from Ľubor Ladický]

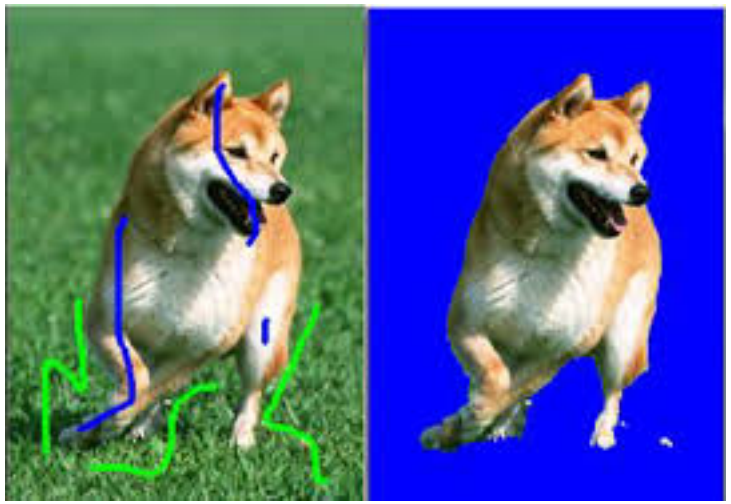
Image Labeling Problems

- Labelings are highly structured
- Labels highly correlated with very complex dependencies
- Independent label estimation too hard
- Whole labeling should be formulated as one optimization problem
- Number of pixels up to millions
  - Hard to train complex dependencies
  - Optimization problem is hard to infer

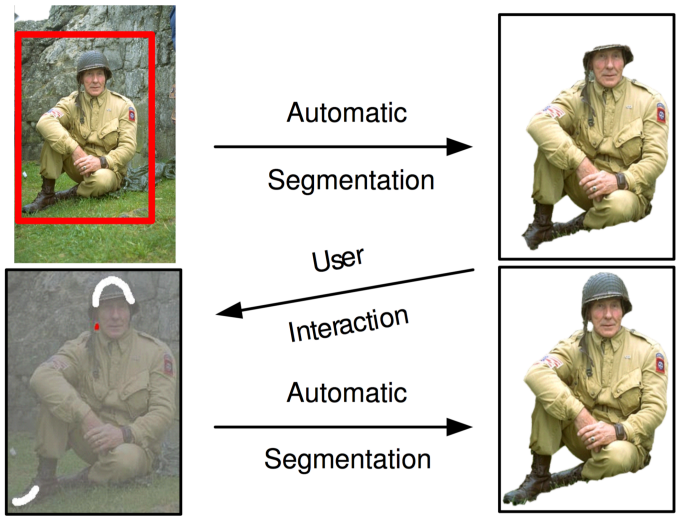
[tutorial slides from Ľubor Ladický]

Image Segmentation

Interactive Graph Cuts, Boykov, Jolly, ICCV 2001



GrabCut – Rother, Kolmogorov, Blake, SIGGRAPH 2004



now part of Powerpoint: Picture Format -> Remove Background

Image Segmentation

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j)$$

Data term **Smoothness term**

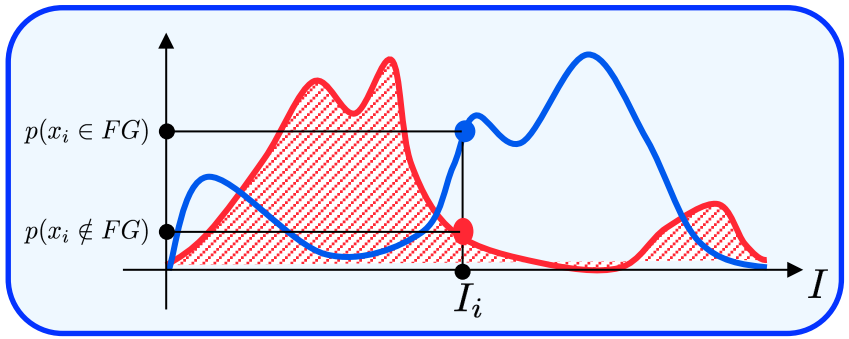
$$x_i = 0 \implies i \in Background$$

$$x_i = 1 \implies i \in Foreground$$

Data term

$$\psi_i(0) = -\log(p(x_i \notin FG))$$

$$\psi_i(1) = -\log(p(x_i \in FG))$$



Estimated using FG / BG color models

Smoothness term

$$\psi_{ij}(x_i, x_j) = K_{ij}\delta(x_i \neq x_j)$$

where $K_{ij} = \lambda_1 + \lambda_2 \exp(-\beta(I_i - I_j)^2)$

Intensity dependent smoothness

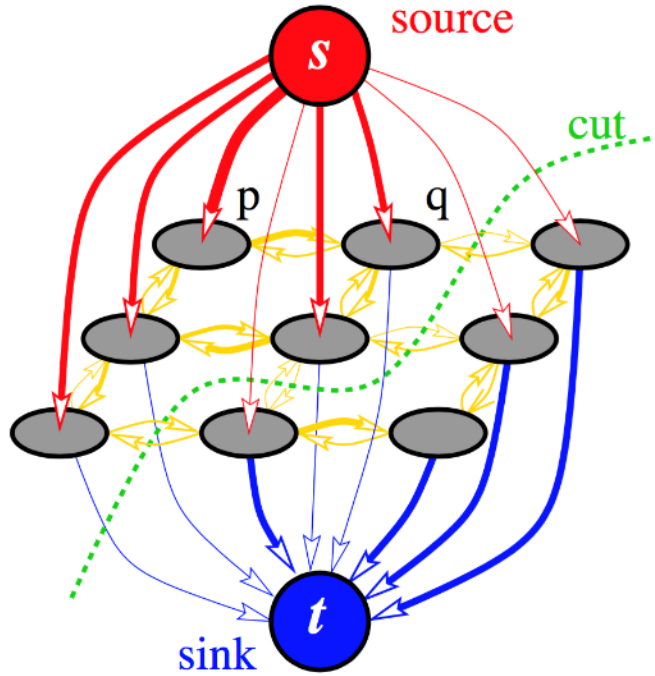
[tutorial slides from Ľubor Ladický]

Image Segmentation

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j)$$

Data term **Smoothness term**

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbf{L}} E(\mathbf{x})$$



How to solve this optimisation problem?

- Transform into min-cut / max-flow problem
- Solve it using min-cut / max-flow algorithm

[tutorial slides from Ľubor Ladický]

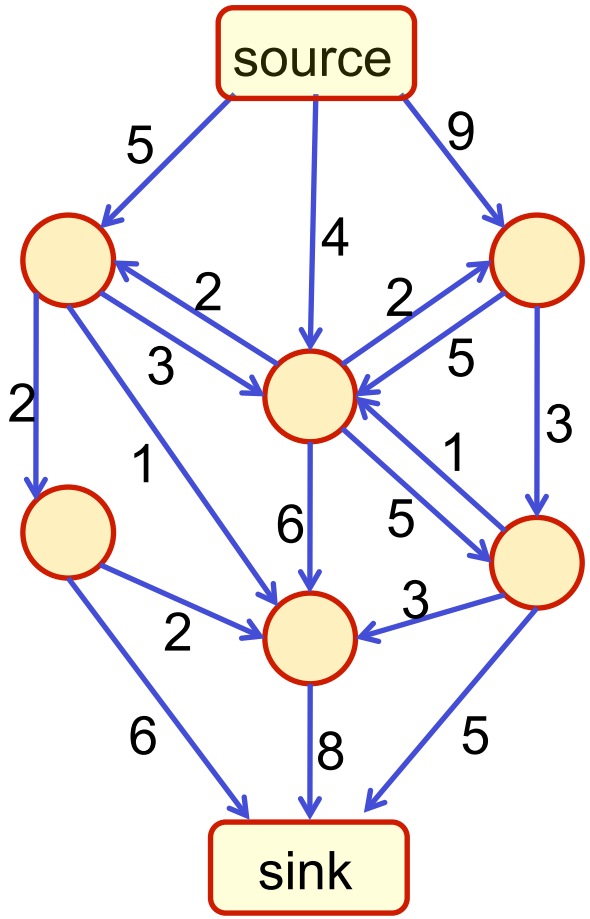
Max-Flow Problem

Task :

Maximize the flow from the source to the sink such that

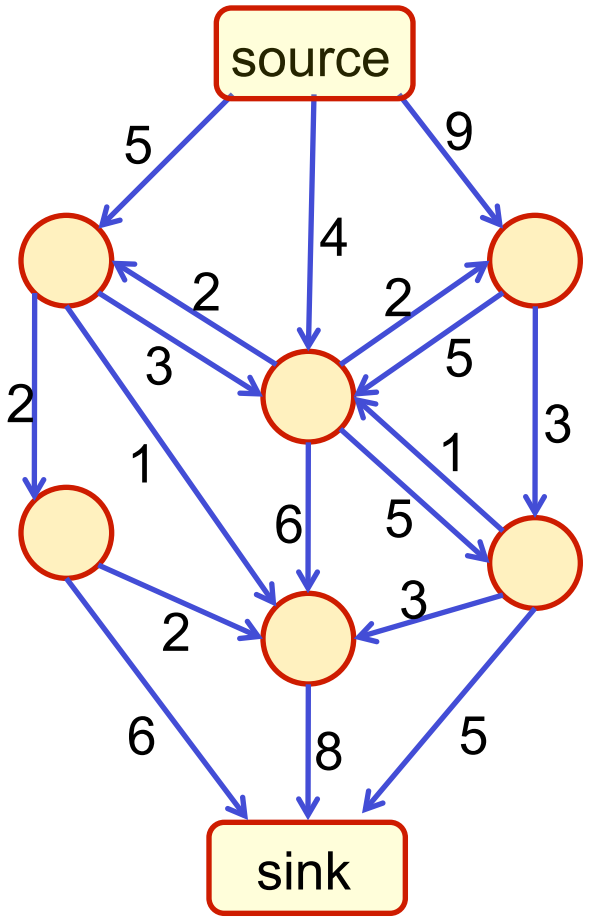
1) The flow it conserved for each node

2) The flow for each pipe does not exceed the capacity



[tutorial slides from Ľubor Ladický]

Max-Flow Problem



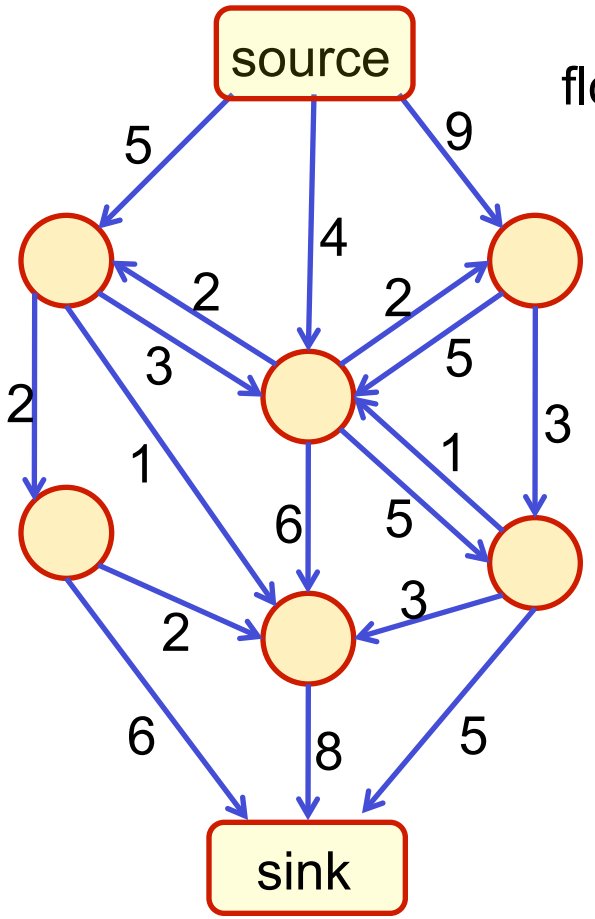
$$\max \sum_{i \in V} f_{si}$$

$$\text{s.t.} \qquad 0 \leq f_{ij} \leq c_{ij}, \qquad \forall (i,j) \in E$$

$$\sum_{j \in N(i)} f_{ji} - f_{ij} = 0, \qquad \forall i \in V \setminus \{s,t\}$$

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow from node i to node j

flow from the source

capacity

set of edges

$$\max \sum_{i \in V} f_{si}$$

$$\text{s.t.} \quad 0 \leq f_{ij} \leq c_{ij}, \qquad \forall (i,j) \in E$$

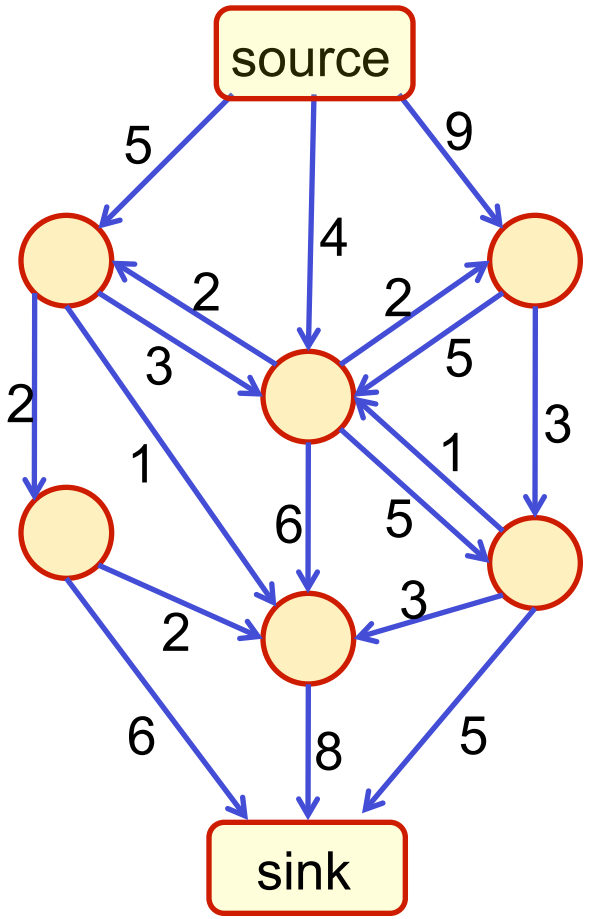
$$\sum_{j \in N(i)} f_{ji} - f_{ij} = 0, \qquad \forall i \in V \setminus \{s,t\}$$

conservation of flow

set of nodes

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

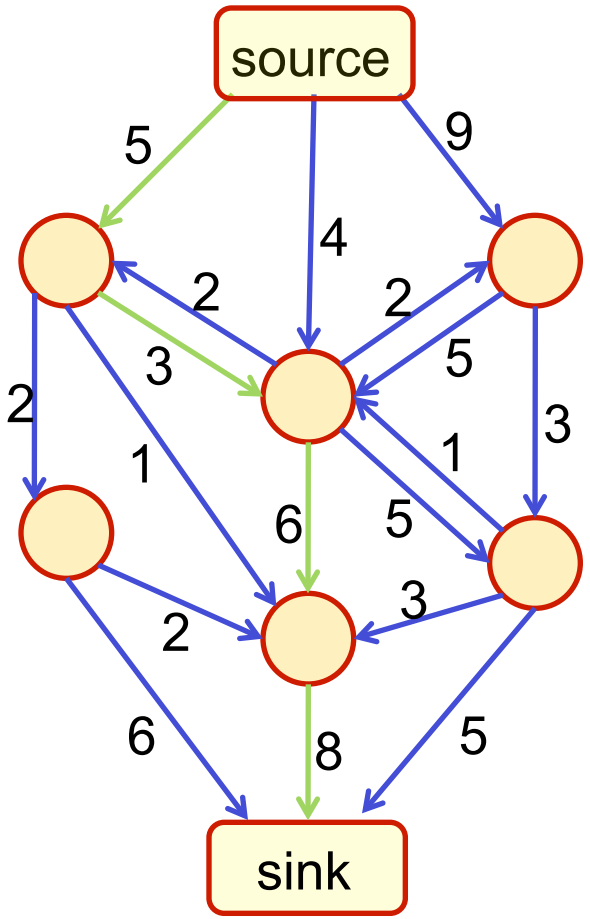
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

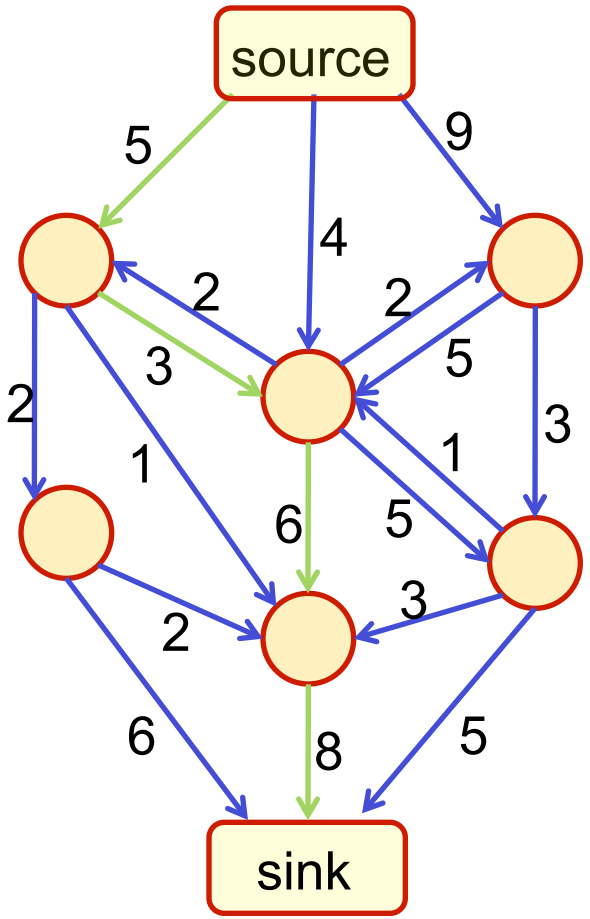
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 3

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

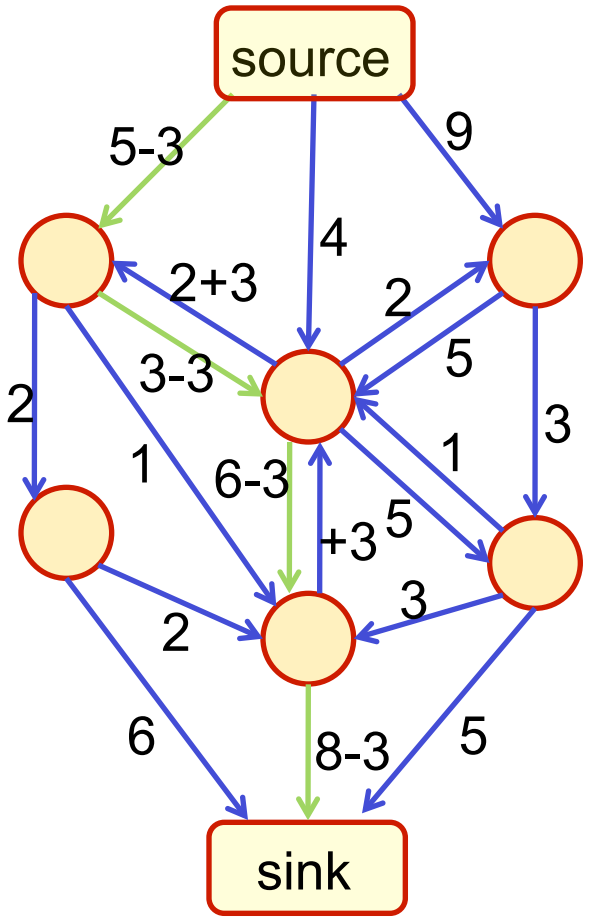
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 3

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

Build the residual graph (“subtract” the flow)

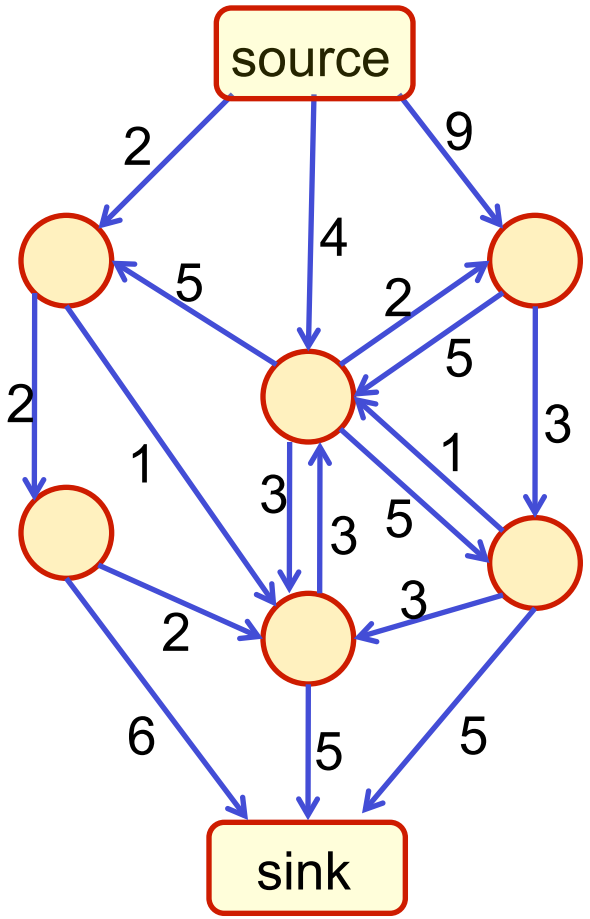
Find the path in the residual graph

End

$$r_{ij} = c_{ij} - f_{ij} + f_{ji}$$

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 3

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

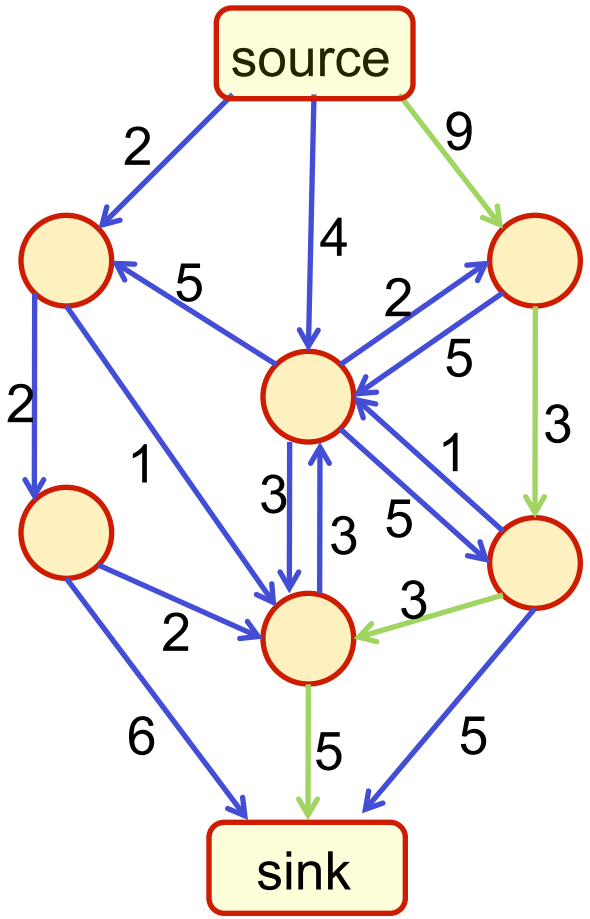
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 3

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

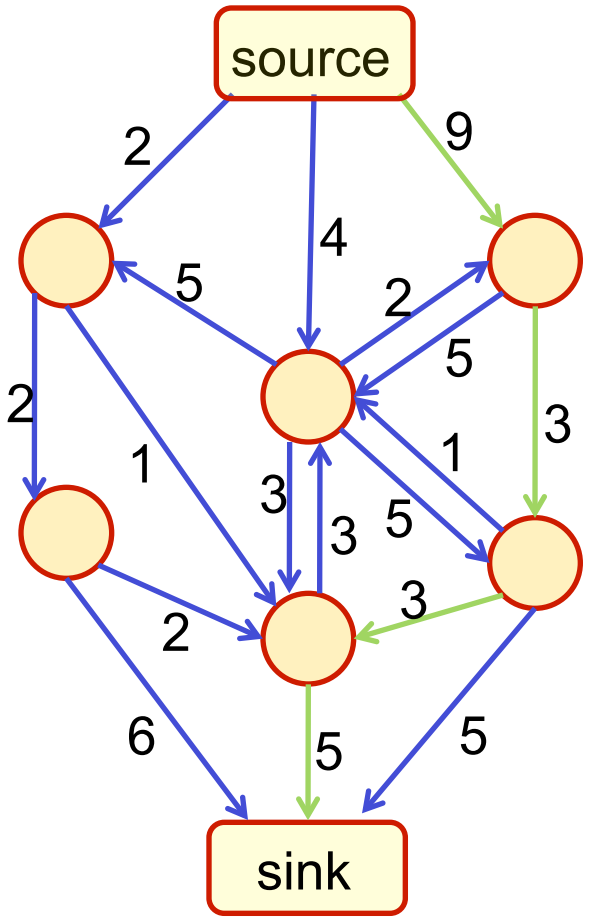
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 6

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

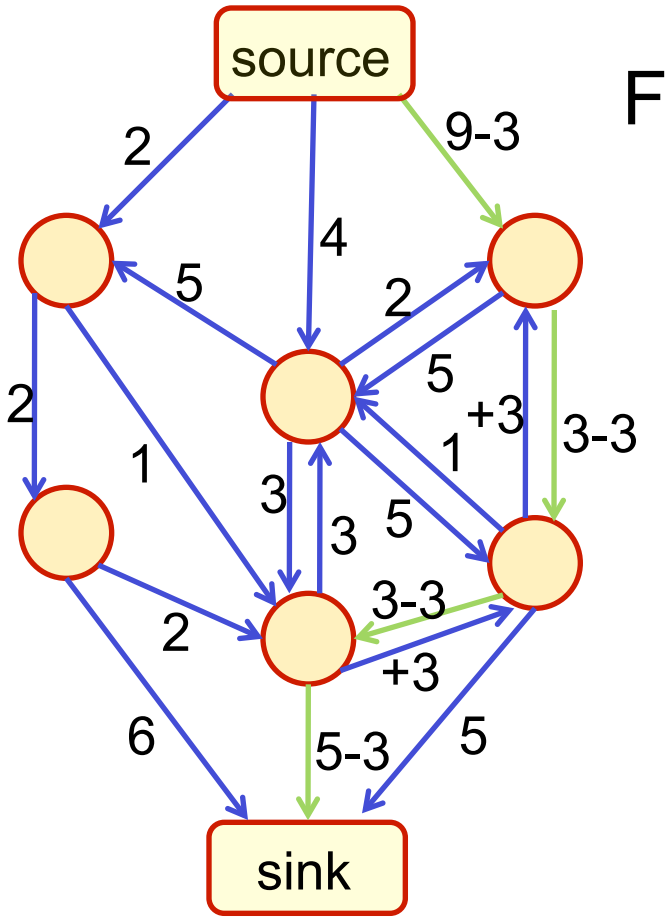
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 6

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

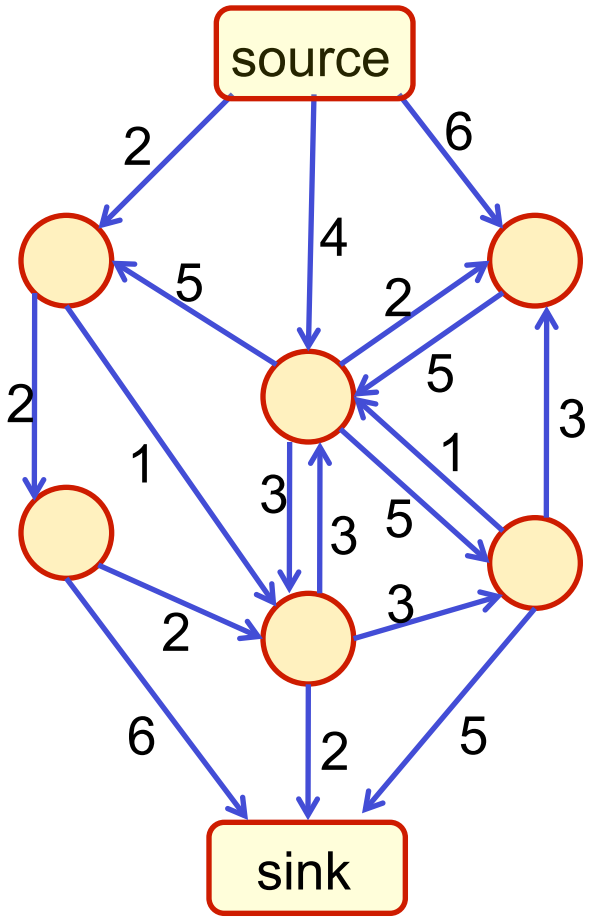
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 6

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

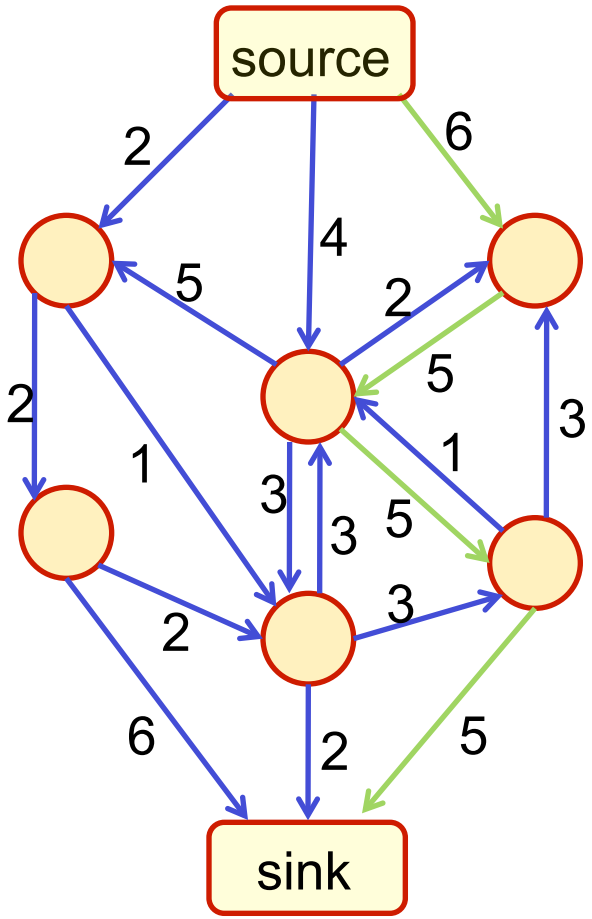
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 6

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

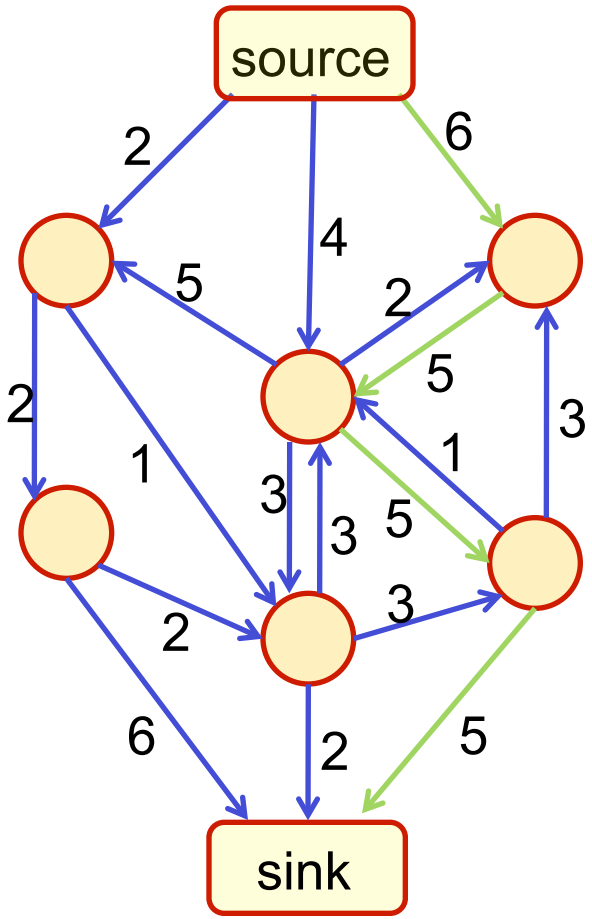
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 11

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

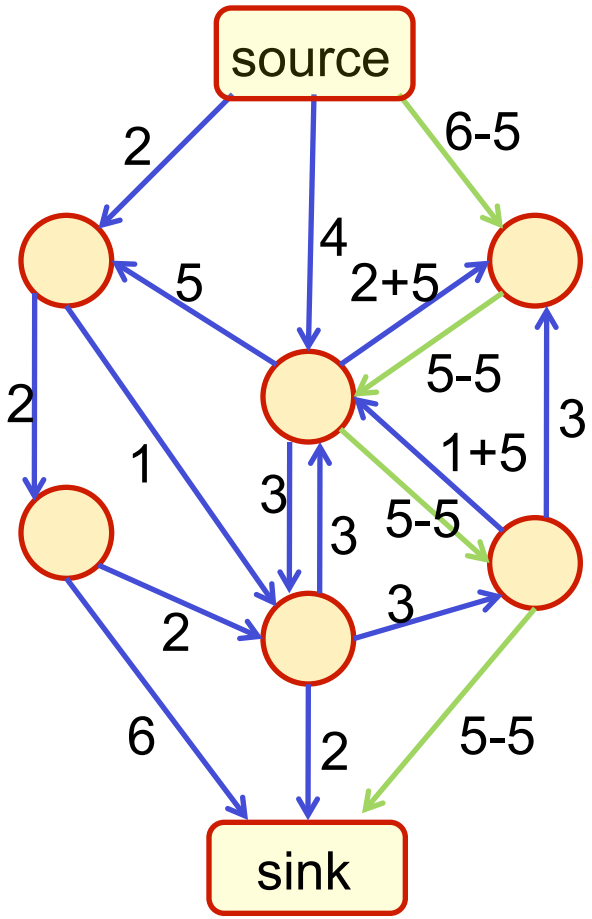
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 11

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

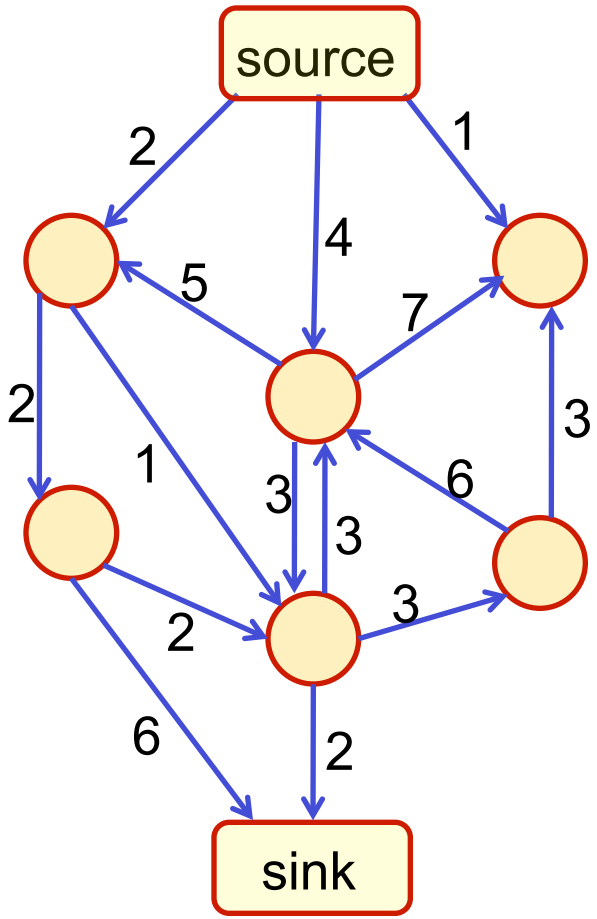
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 11

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

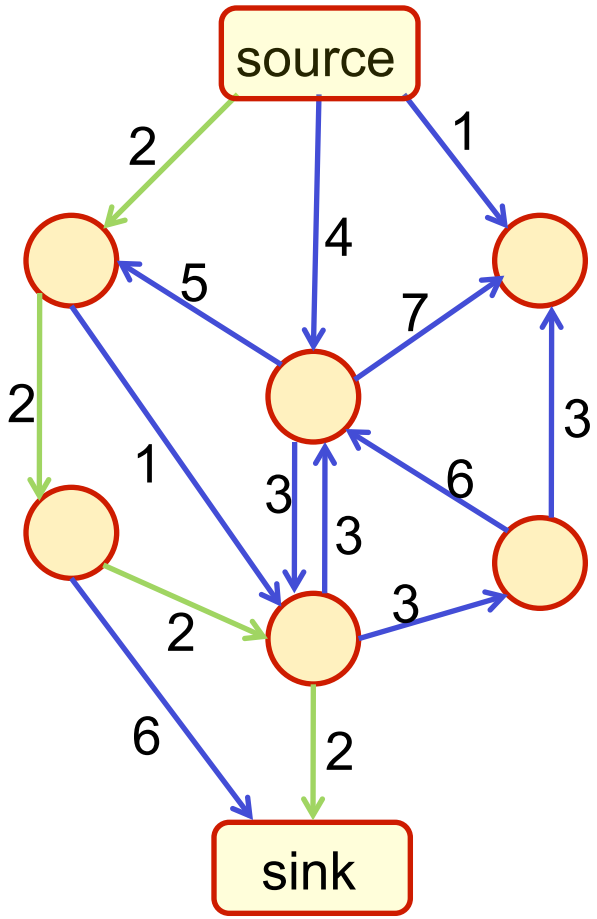
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 11

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

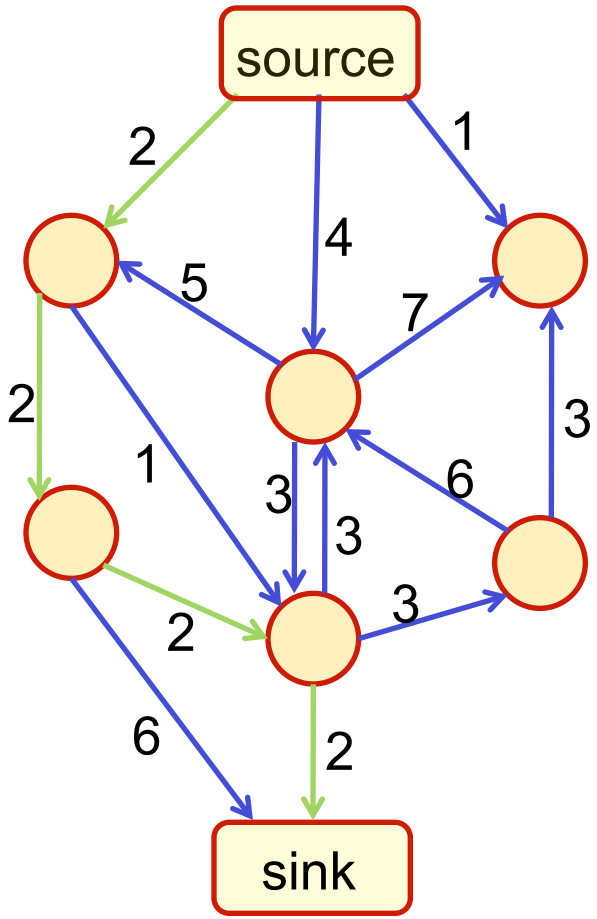
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 13

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

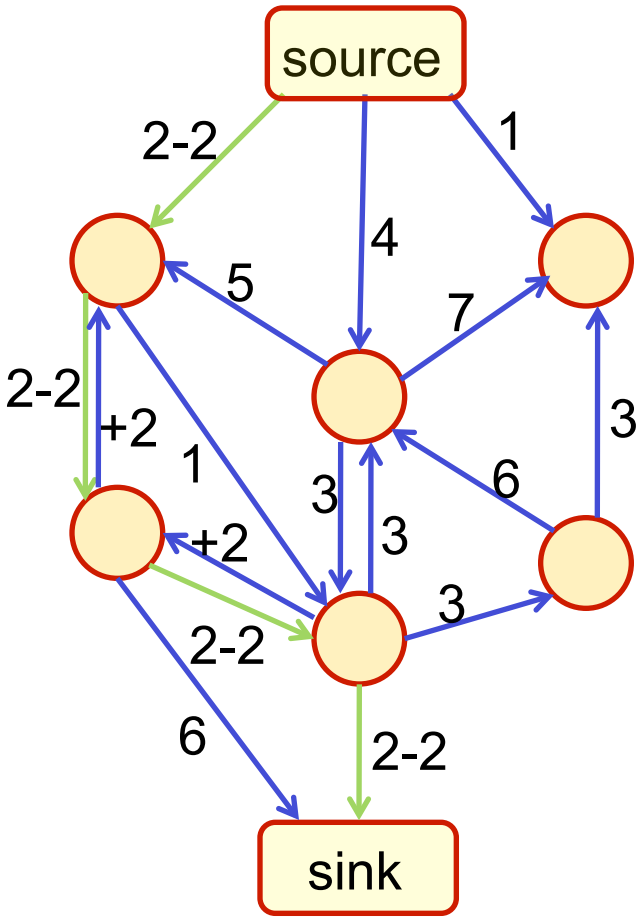
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 13

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

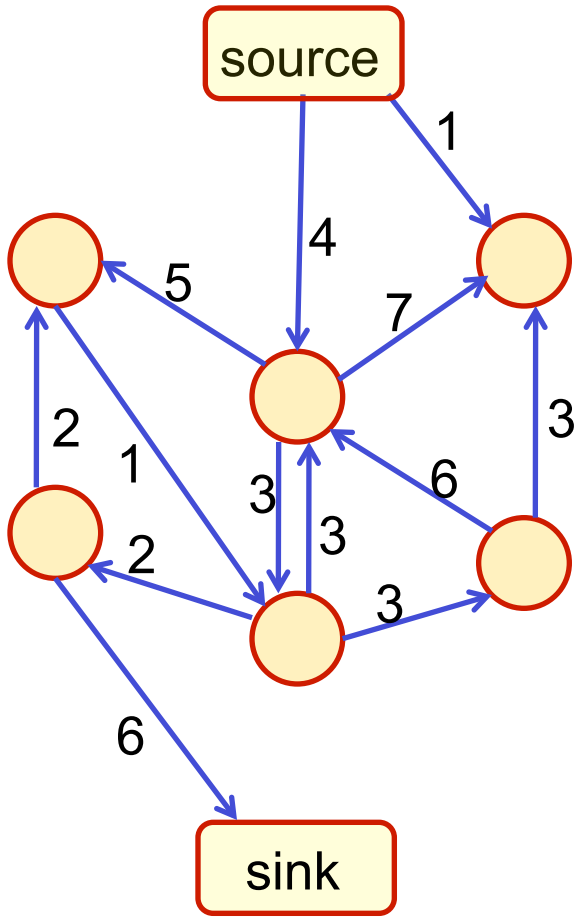
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 13

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

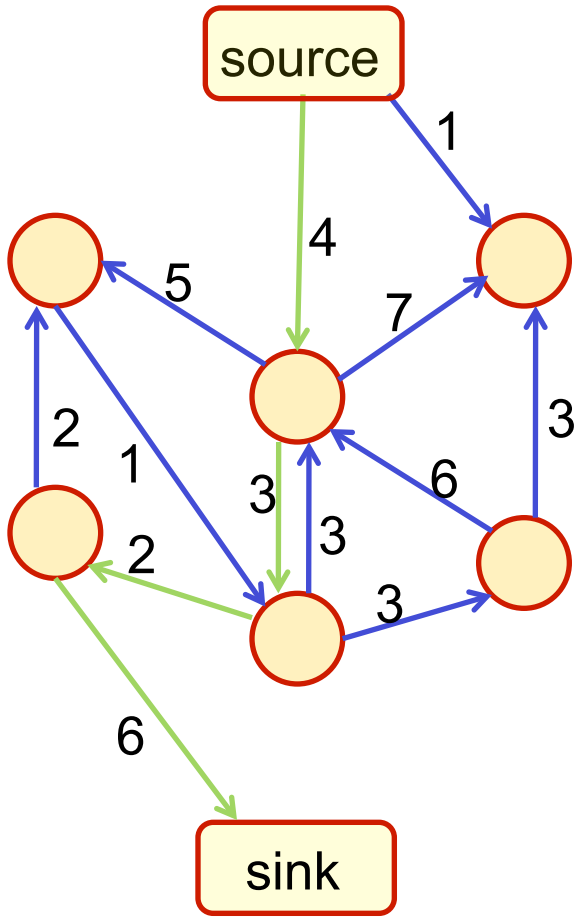
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 13

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

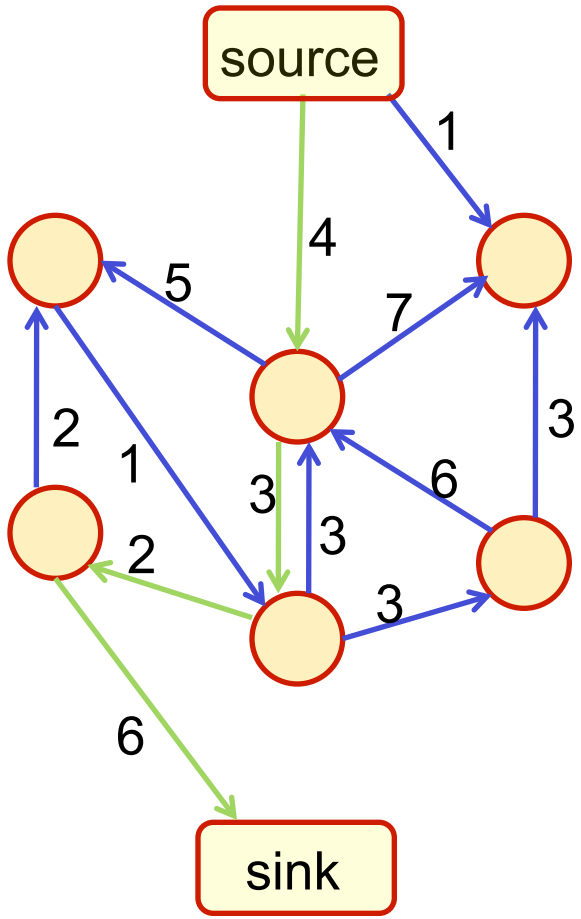
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 15

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

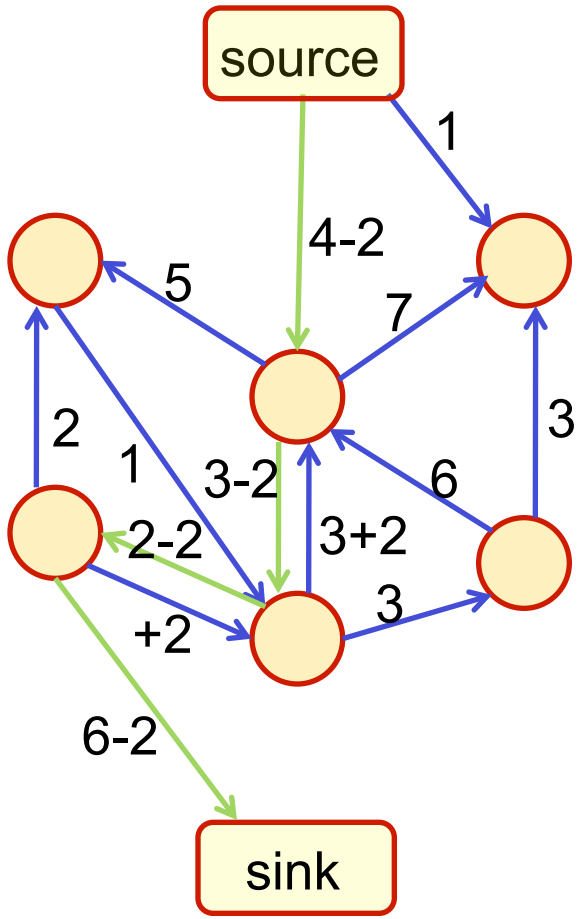
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 15

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

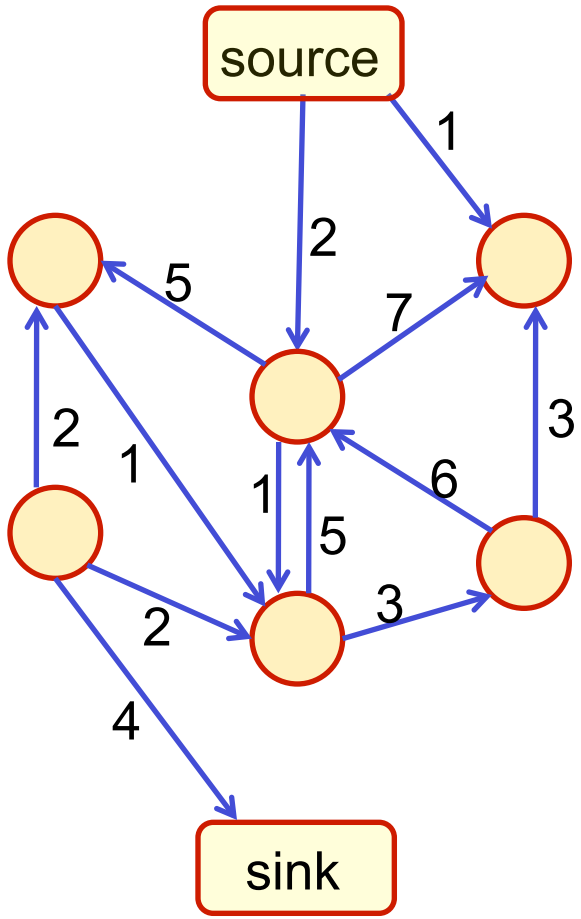
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 15

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

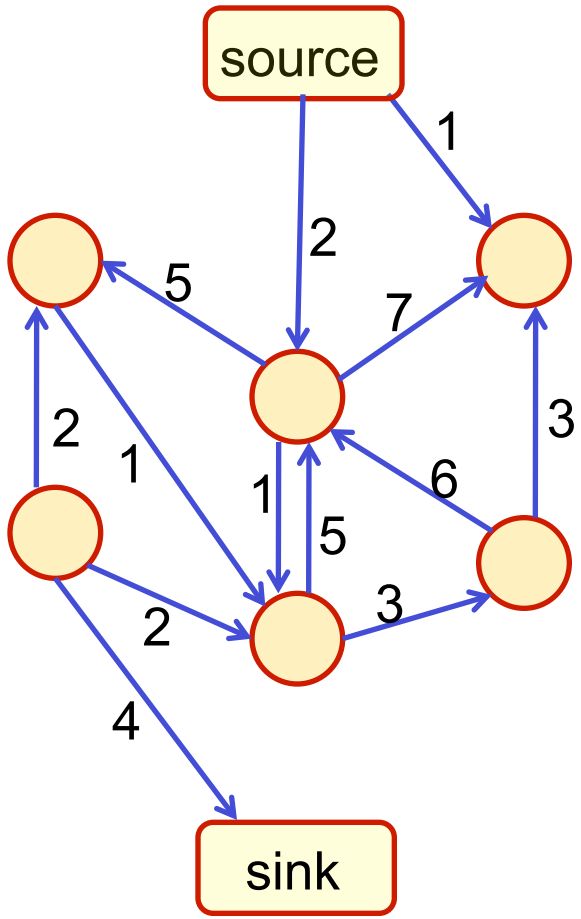
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 15

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

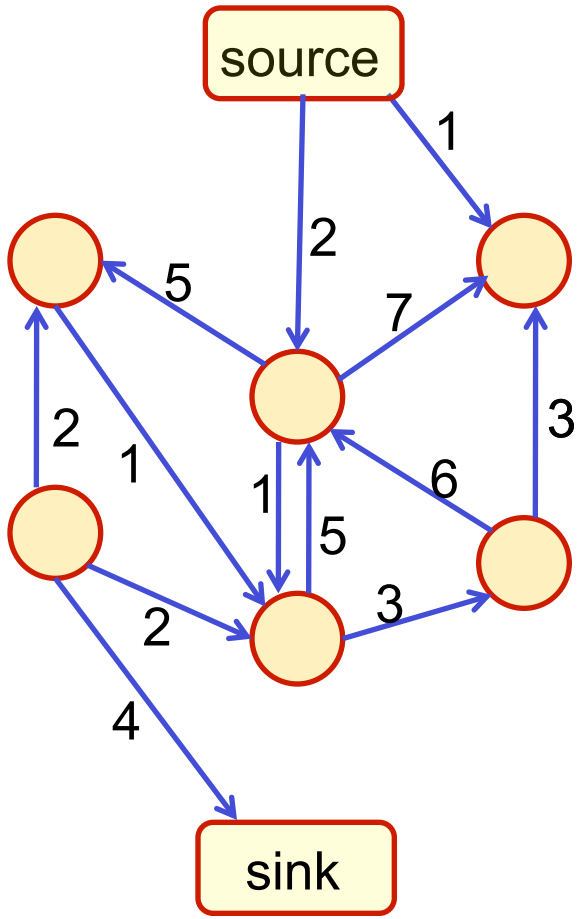
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



flow = 15

Ford & Fulkerson algorithm (1956)

Find the path from source to sink

While (path exists)

flow += maximum capacity in the path

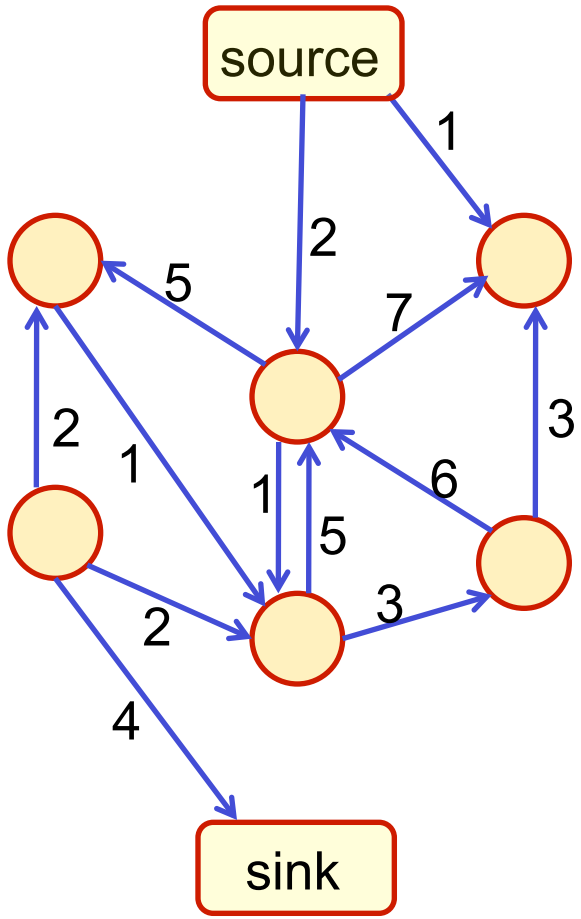
Build the residual graph (“subtract” the flow)

Find the path in the residual graph

End

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



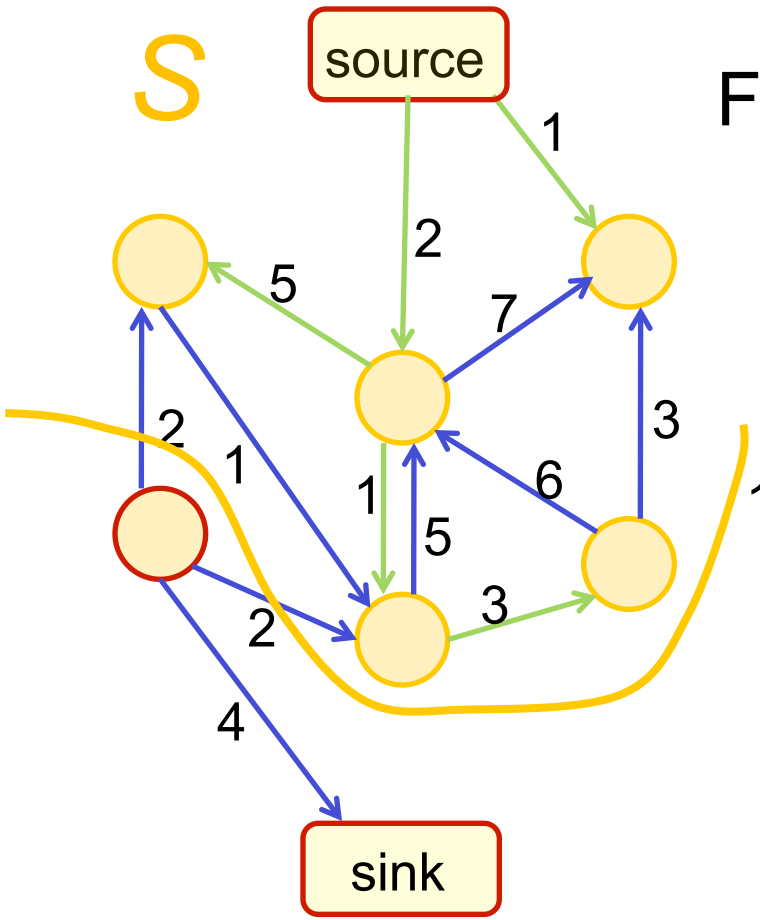
Ford & Fulkerson algorithm (1956)

Why is the solution globally optimal ?

flow = 15

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



Ford & Fulkerson algorithm (1956)

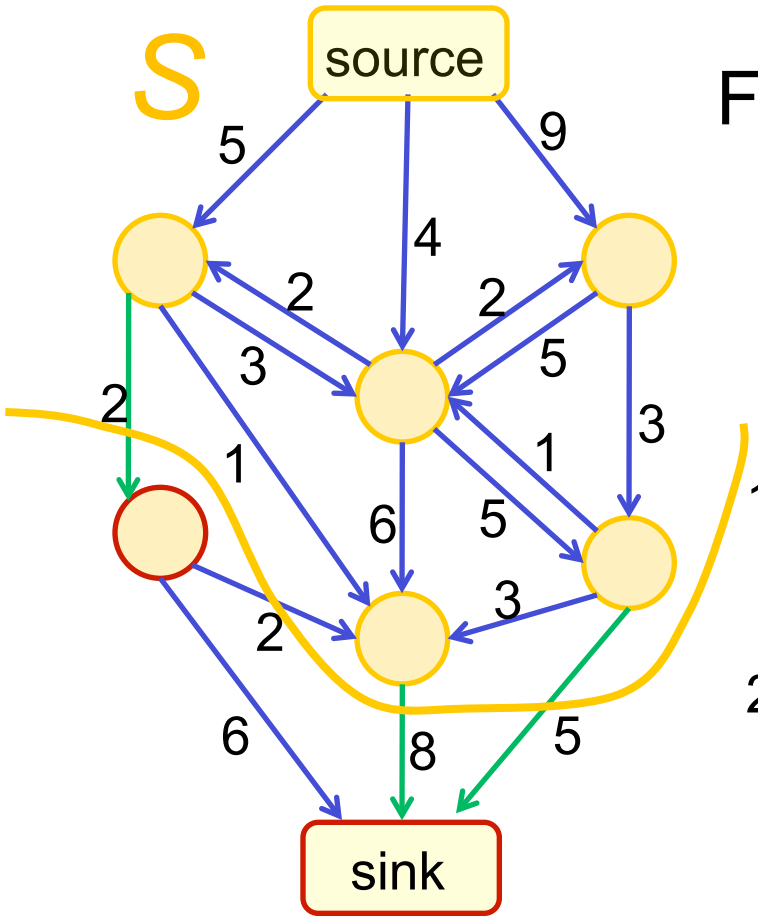
Why is the solution globally optimal ?

1. Let S be the set of reachable nodes in the residual graph

flow = 15

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



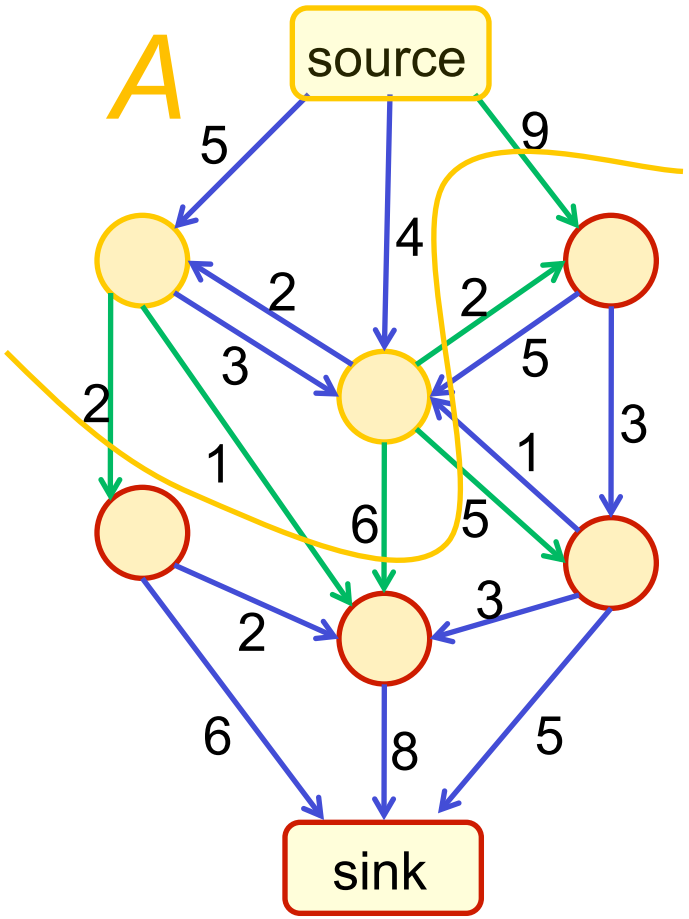
Ford & Fulkerson algorithm (1956)

Why is the solution globally optimal ?

1. Let S be the set of reachable nodes in the residual graph
2. The flow from S to V - S equals to the sum of capacities from S to V – S

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



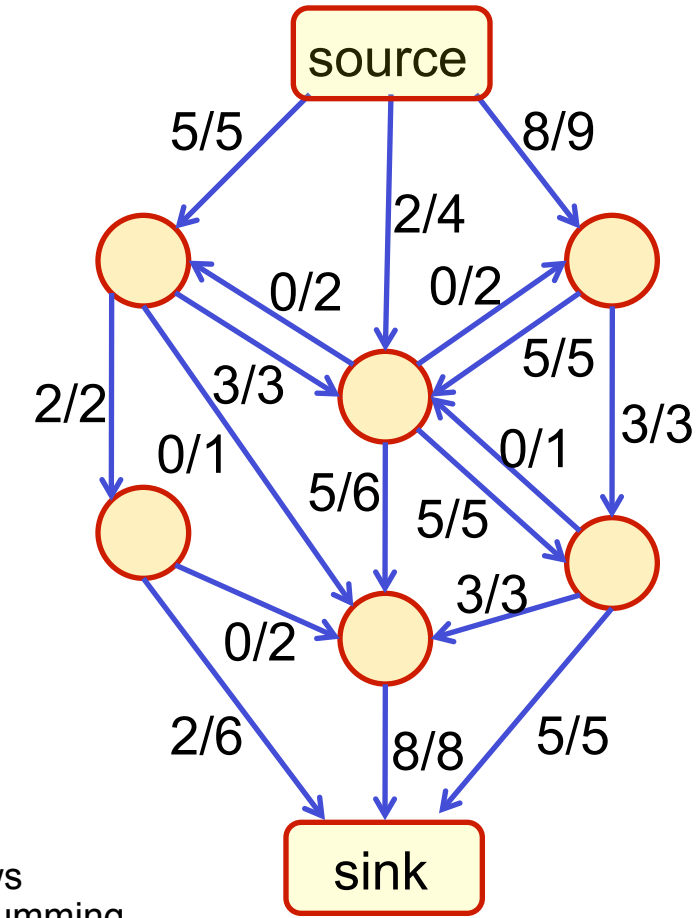
Ford & Fulkerson algorithm (1956)

Why is the solution globally optimal ?

1. Let S be the set of reachable nodes in the residual graph
2. The flow from S to V - S equals to the sum of capacities from S to V – S
3. The flow from any A to V - A is upper bounded by the sum of capacities from A to V – A

[tutorial slides from Ľubor Ladický]

Max-Flow Problem



Individual flows obtained by summing up all paths

flow = 15

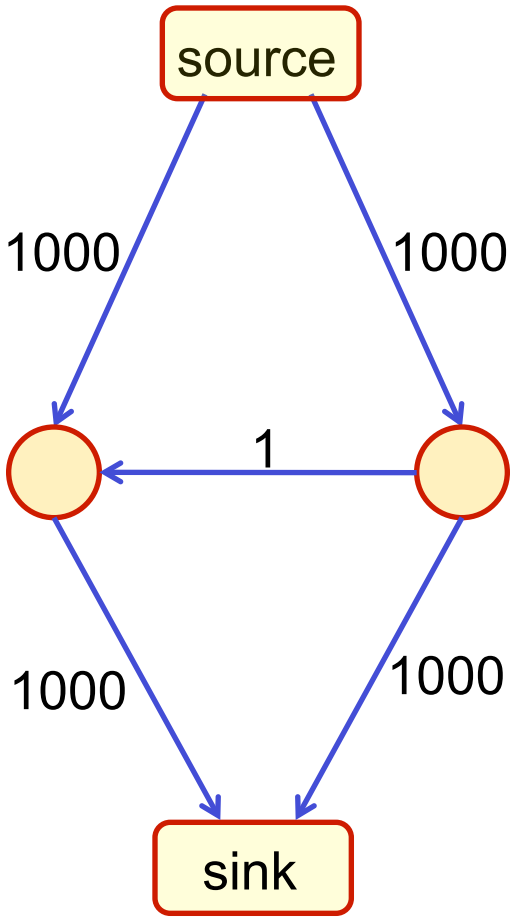
Ford & Fulkerson algorithm (1956)

Why is the solution globally optimal ?

1. Let S be the set of reachable nodes in the residual graph
2. The flow from S to V - S equals to the sum of capacities from S to V – S
3. The flow from any A to V - A is upper bounded by the sum of capacities from A to V – A
4. The solution is globally optimal

[tutorial slides from Ľubor Ladický]

Path Selection

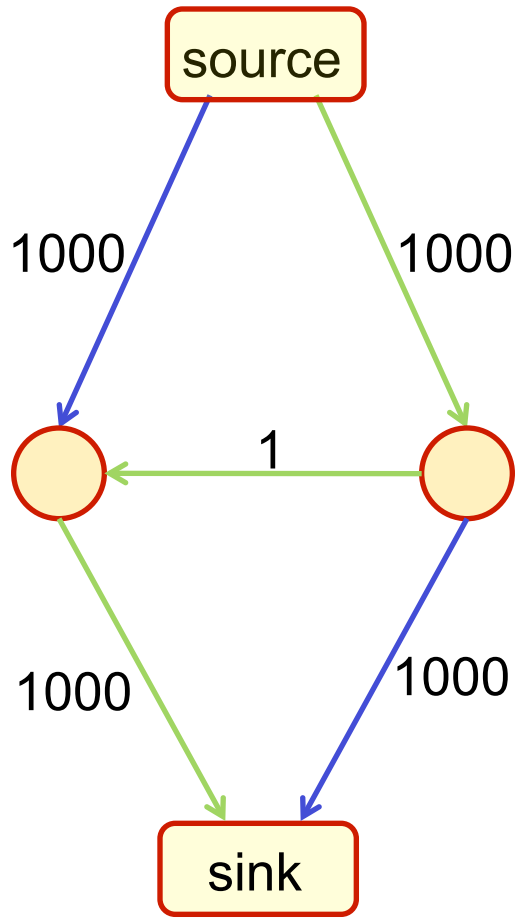


Ford & Fulkerson algorithm (1956)

Order does matter

[tutorial slides from Ľubor Ladický]

Path Selection

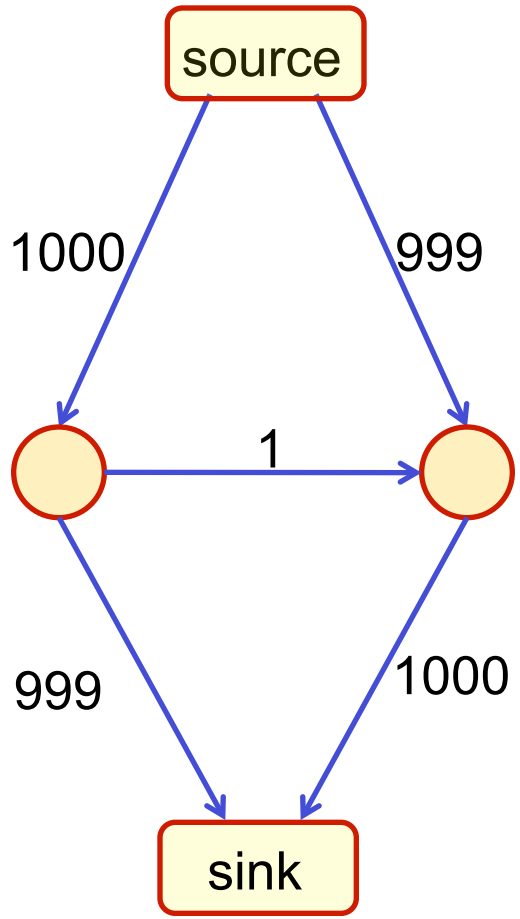


Ford & Fulkerson algorithm (1956)

Order does matter

[tutorial slides from Ľubor Ladický]

Path Selection

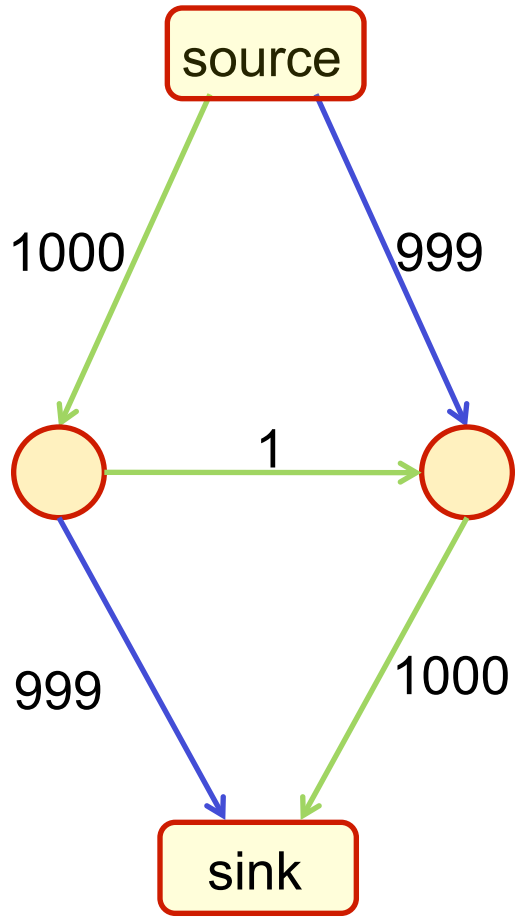


Ford & Fulkerson algorithm (1956)

Order does matter

[tutorial slides from Ľubor Ladický]

Path Selection

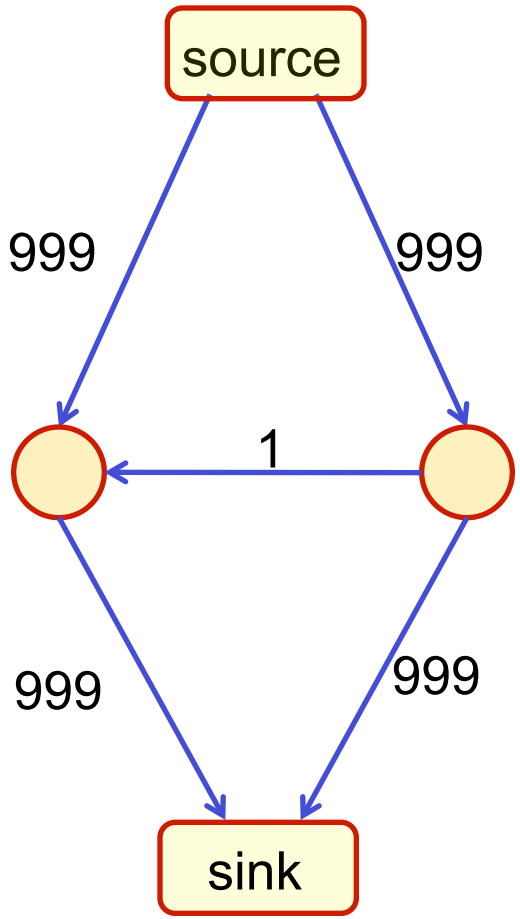


Ford & Fulkerson algorithm (1956)

Order does matter

[tutorial slides from Ľubor Ladický]

Path Selection

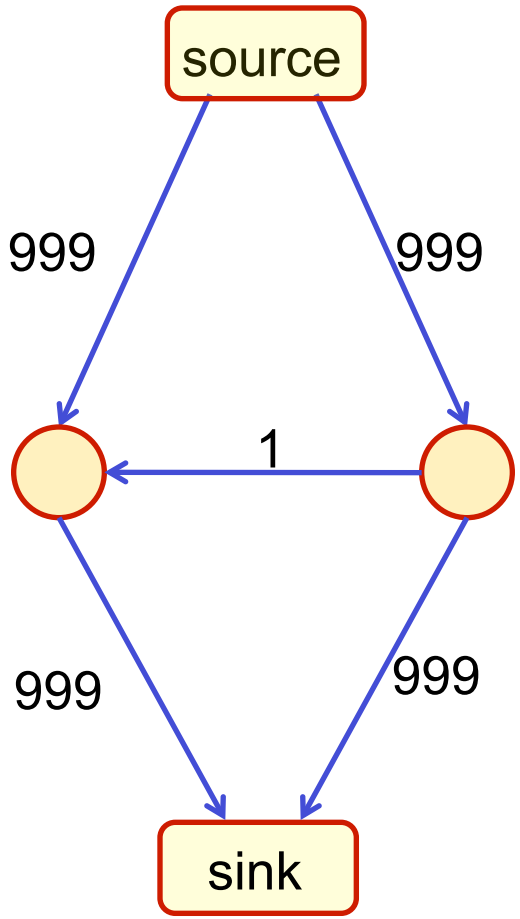


Ford & Fulkerson algorithm (1956)

Order does matter

[tutorial slides from Ľubor Ladický]

Path Selection



Ford & Fulkerson algorithm (1956)

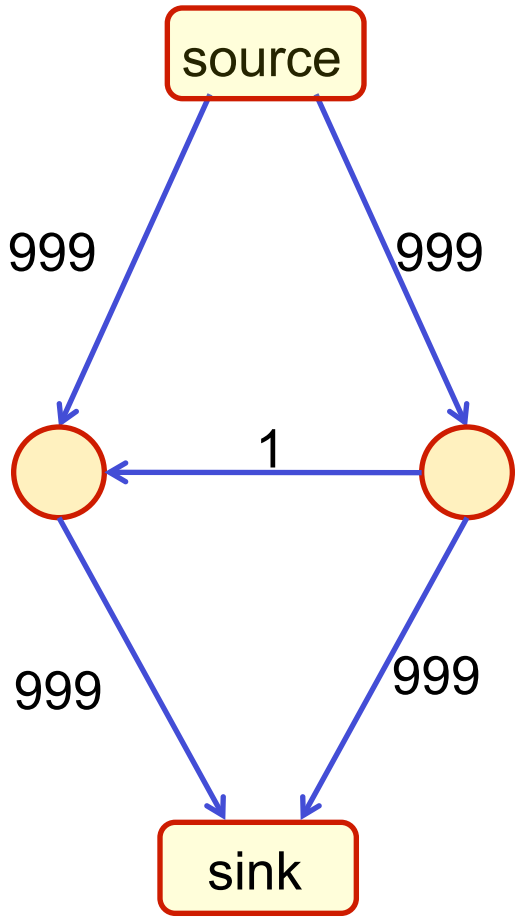
Order does matter

- Standard algorithm not polynomial
- Breath first leads to $O(VE^2)$
  - Path found in O(E)
  - At least one edge gets saturated
  - The saturated edge distance to the source has to increase and is at most V leading to O(VE)

(Edmonds & Karp, Dinic)

[tutorial slides from Ľubor Ladický]

Path Selection



Ford & Fulkerson algorithm (1956)

Order does matter

- Standard algorithm not polynomial
- Breath first leads to $O(VE^2)$

  (Edmonds & Karp)
- Various methods use different algorithms to find the path and vary in complexity

[tutorial slides from Ľubor Ladický]

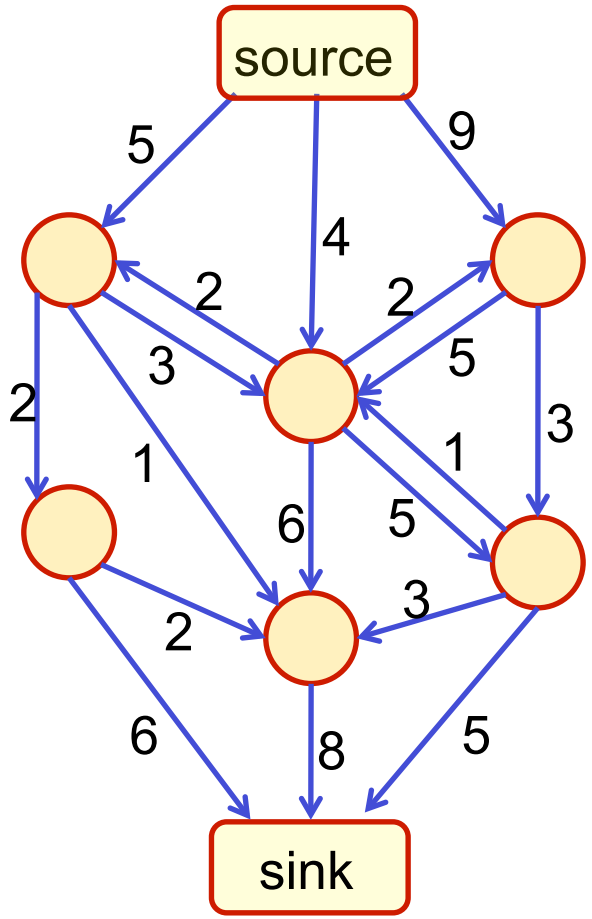
Min-Cut Problem

Task :

Minimize the cost of the cut

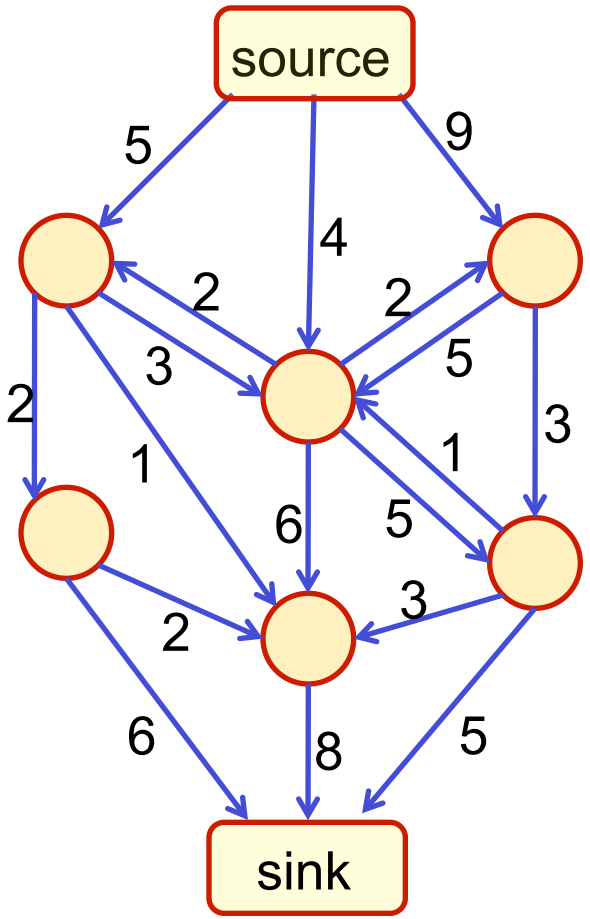
1) Each node is either assigned to the source S or sink T

2) The cost of the edge (i, j) is taken if (i ∈ S) and (j ∈ T)



[tutorial slides from Ľubor Ladický]

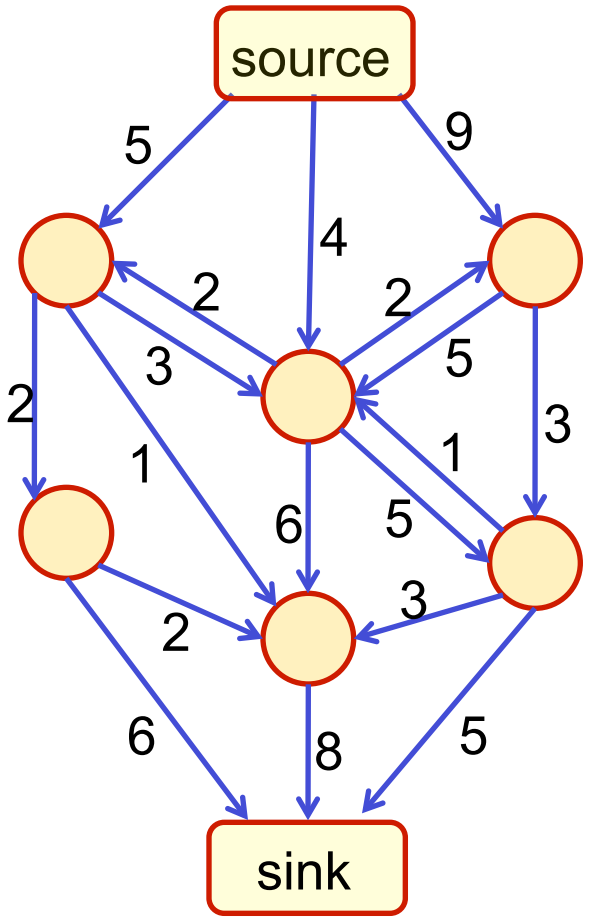
Min-Cut Problem

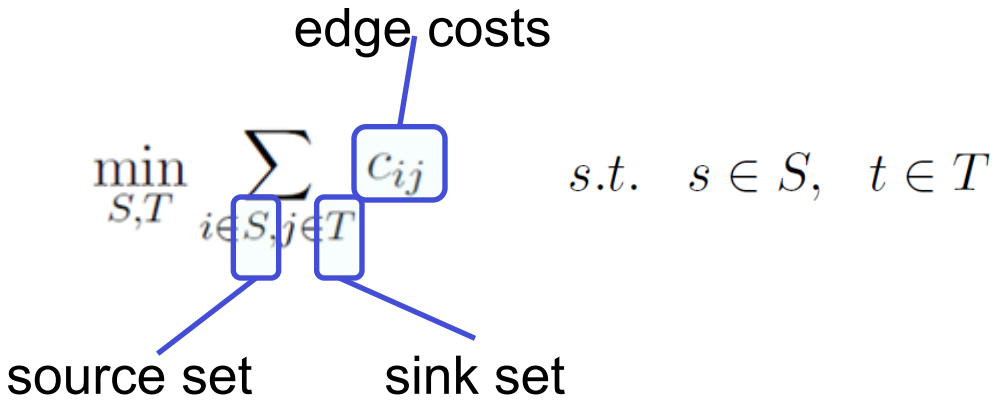


$$\min_{S,T} \sum_{i \in S, j \in T} c_{ij}$$

[tutorial slides from Ľubor Ladický]

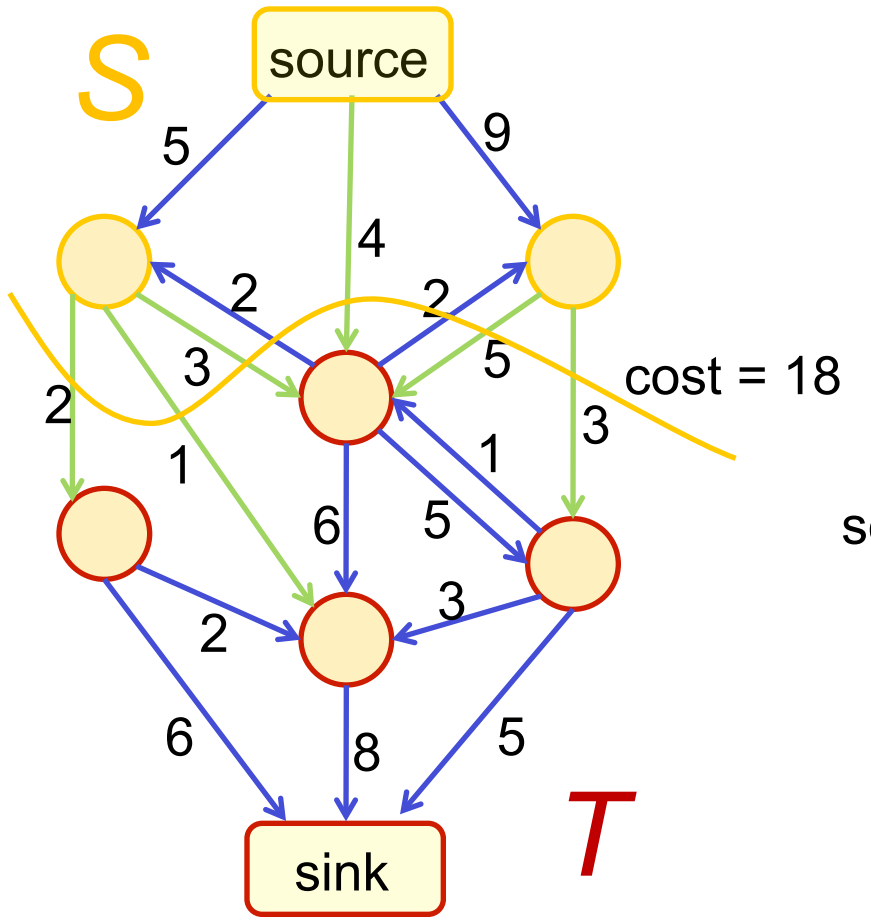
Min-Cut Problem





[tutorial slides from Ľubor Ladický]

Min-Cut Problem



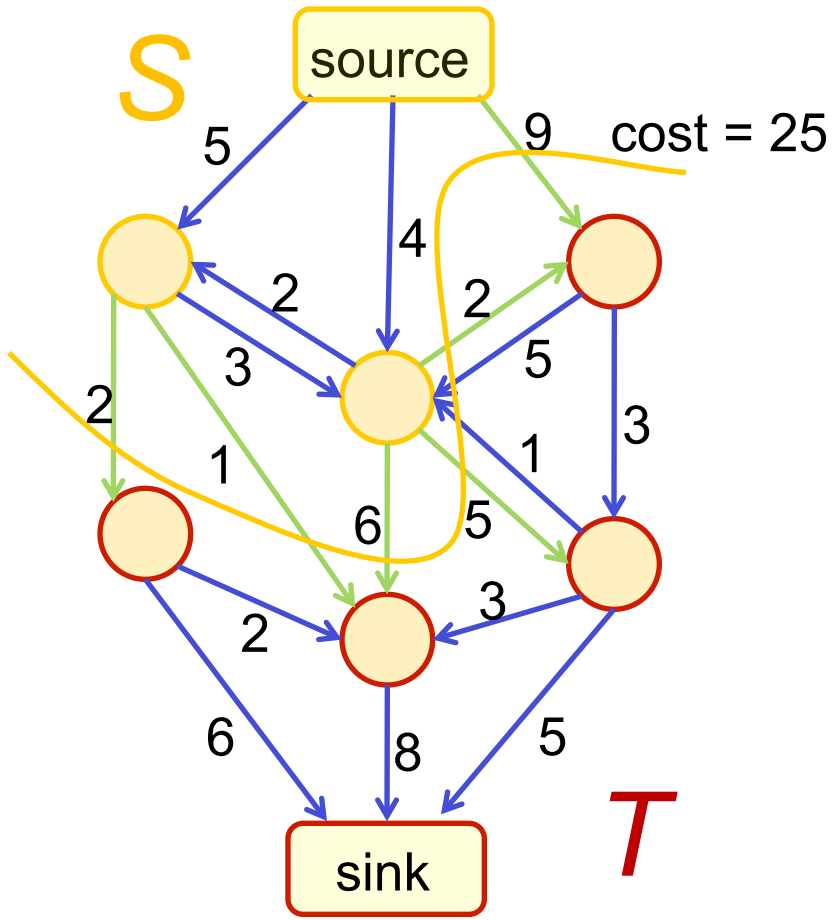
edge costs

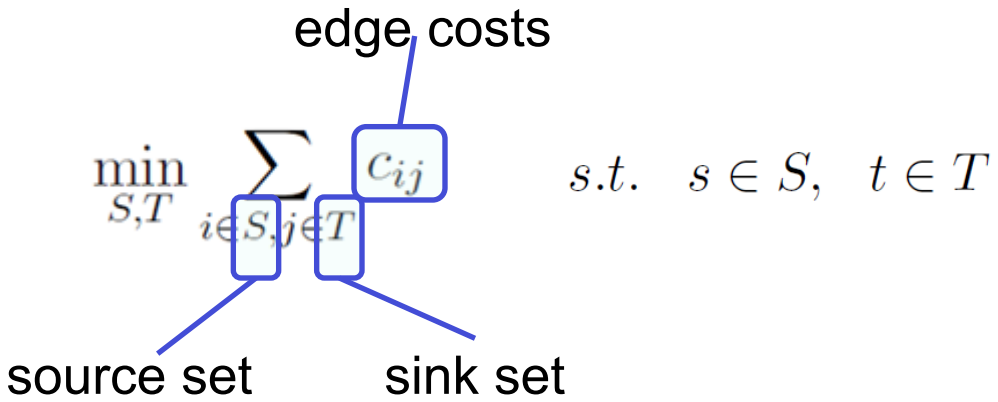
$$\min_{S,T} \sum_{i \in S, j \in T} c_{ij} \qquad s.t. \quad s \in S, \quad t \in T$$

source set sink set

[tutorial slides from Ľubor Ladický]

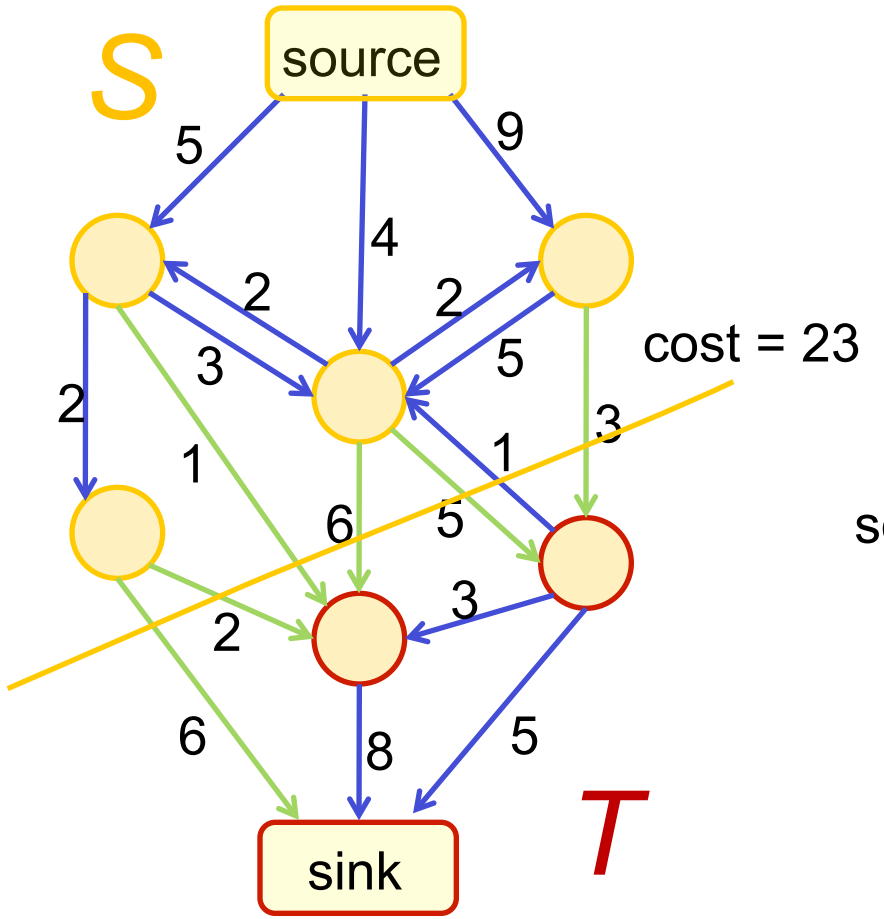
Min-Cut Problem





[tutorial slides from Ľubor Ladický]

Min-Cut Problem



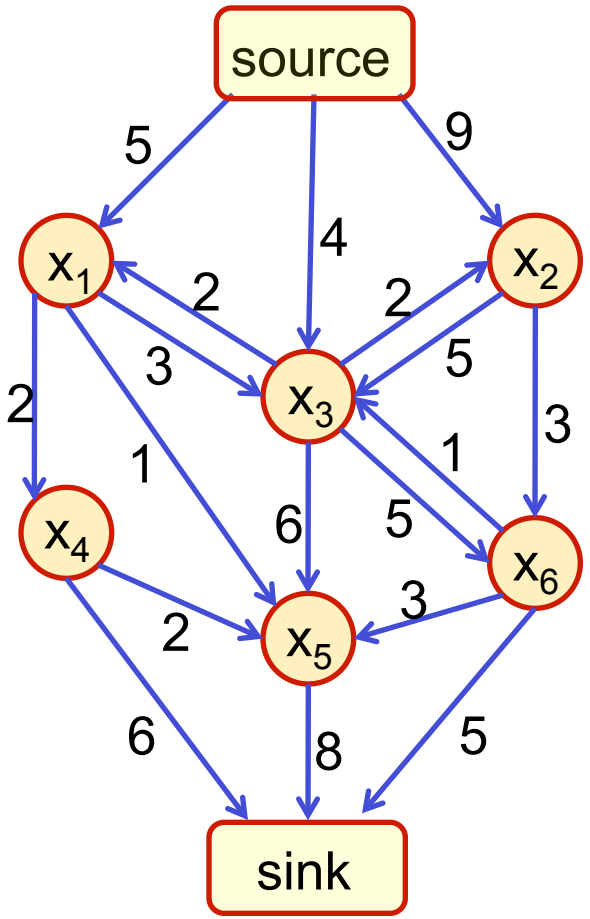
edge costs

$$\min_{S,T} \sum_{i \in S, j \in T} c_{ij} \qquad s.t. \quad s \in S, \quad t \in T$$

source set sink set

[tutorial slides from Ľubor Ladický]

Min-Cut Problem



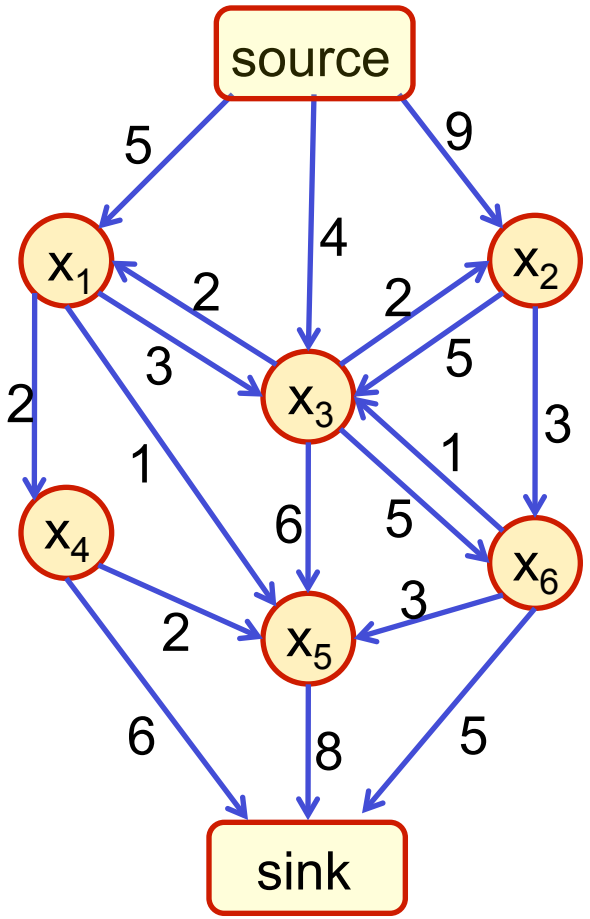
$$\min_{\mathbf{x}} \sum_{(i,j)\in E} c_{ij}(1-x_i)x_j$$

$$s.t. \quad x_s = 0, \quad x_t = 1$$

$$x_i = 0 \implies x_i \in S \qquad x_i = 1 \implies x_i \in T$$

[tutorial slides from Ľubor Ladický]

Min-Cut Problem



$$\min_{\mathbf{x}} \sum_{(i,j)\in E} c_{ij}(1-x_i)x_j$$

$$s.t. \quad x_s = 0, \quad x_t = 1$$

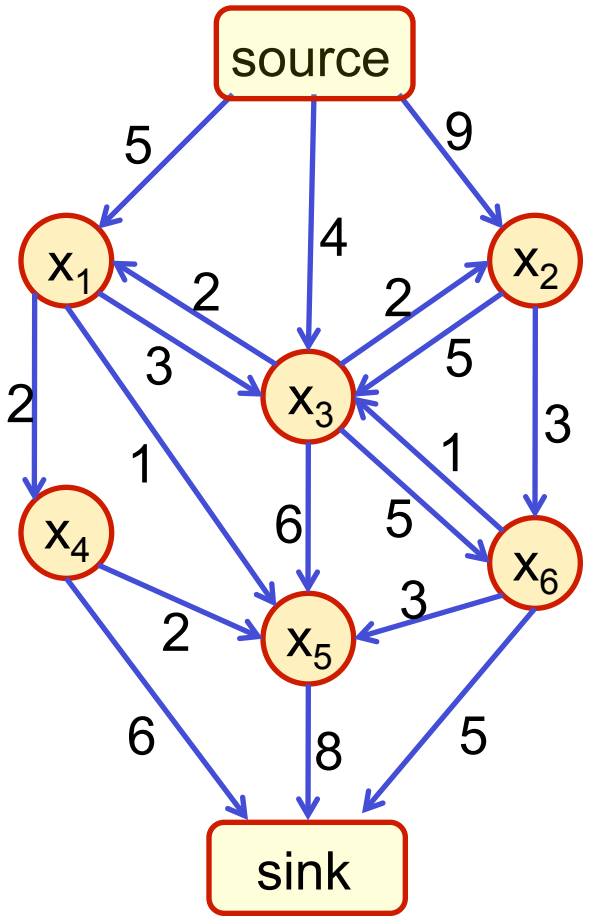
transformable into Linear program (LP)

$$\min_{\mathbf{x},\mathbf{d}} \quad c_{ij}d_{ij}$$

$$s.t. \quad d_{ij} \geq x_j - x_i \qquad d_{ij} \geq 0$$
$$x_s = 0 \qquad x_t = 1$$

[tutorial slides from Ľubor Ladický]

Min-Cut Problem



$$\min_{\mathbf{x}} \sum_{(i,j)\in E} c_{ij}(1-x_i)x_j$$

$$s.t. \quad x_s = 0, \quad x_t = 1$$

transformable into Linear program (LP)

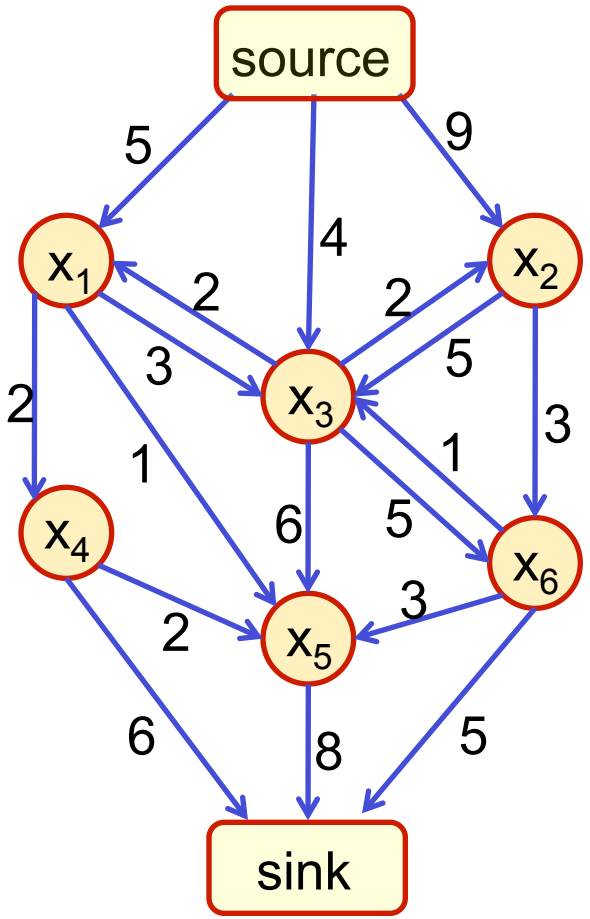
$$\min_{\mathbf{x},\mathbf{d}} \quad c_{ij}d_{ij}$$

$$s.t. \quad d_{ij} \geq x_j - x_i \qquad d_{ij} \geq 0$$
$$x_s = 0 \qquad x_t = 1$$

Dual to max-flow problem

[tutorial slides from Ľubor Ladický]

Min-Cut Problem



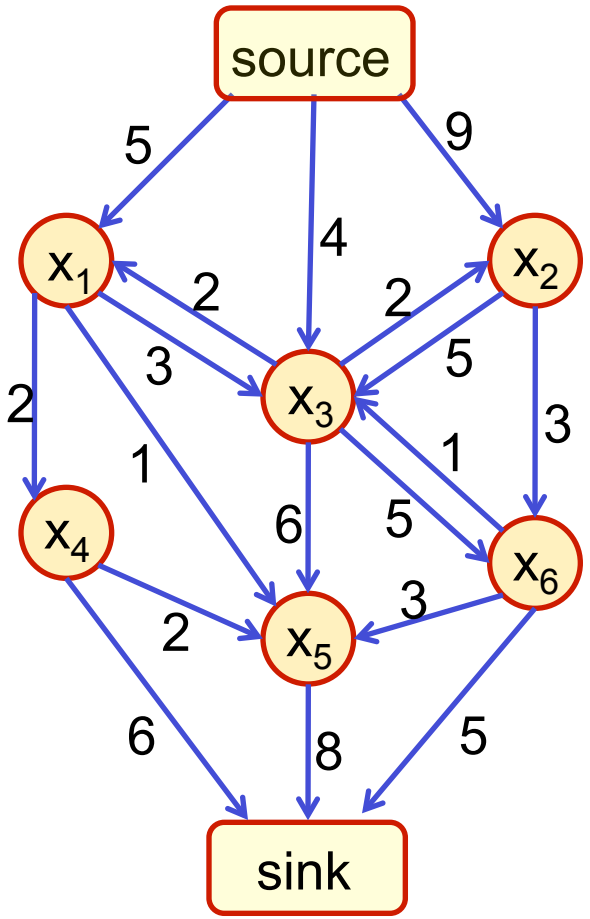
After the substitution of the constraints :

$$\min_{\mathbf{x}} \sum_{(s,i)\in E} c_{si} x_i + \sum_{(i,t)\in E} c_{it}(1-x_i) + \sum_{(i,j)\in E, i,j\notin\{s,t\}} c_{ij}(1-x_i)x_j$$

$$x_i = 0 \implies x_i \in S \qquad x_i = 1 \implies x_i \in T$$

[tutorial slides from Ľubor Ladický]

Min-Cut Problem



source edges sink edges

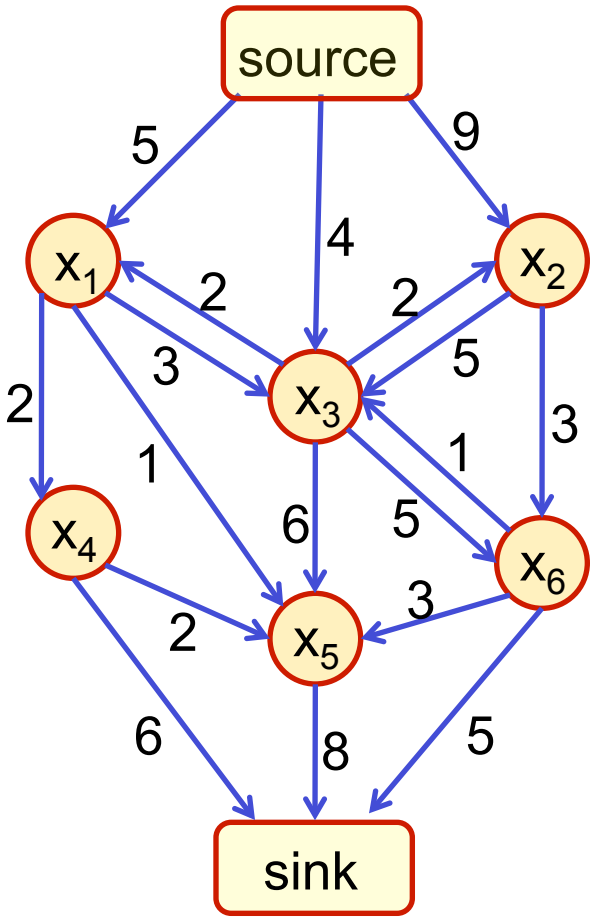
$$\min_{\mathbf{x}} \sum_{(s,i)\in E} c_{si} x_i + \sum_{(i,t)\in E} c_{it}(1-x_i) + \sum_{(i,j)\in E, i,j\notin\{s,t\}} c_{ij}(1-x_i)x_j$$

$x_i = 0 \Longrightarrow x_i \in T$ $\quad$ $x_i = 1 \Longrightarrow x_i \in S$

Edges between variables

[tutorial slides from Ľubor Ladický]

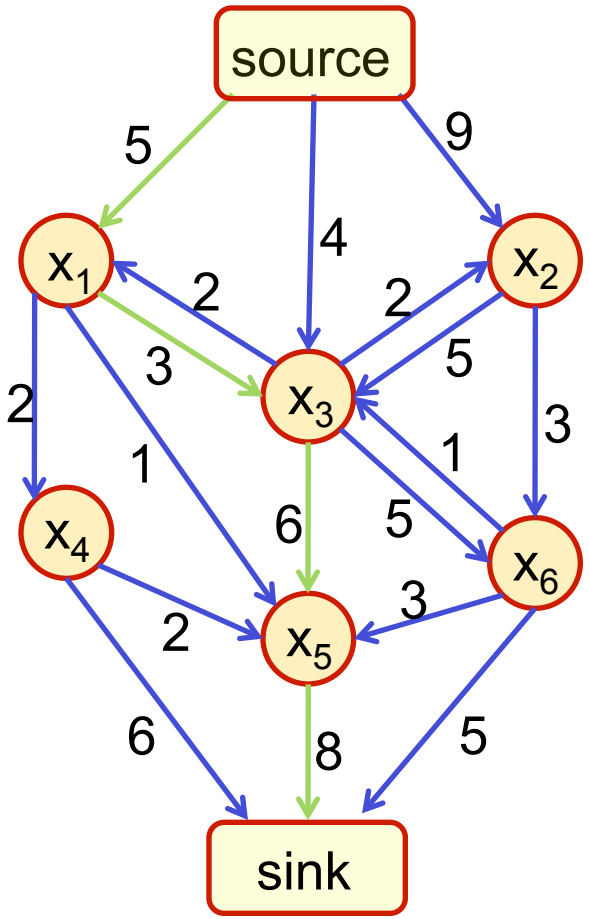
Min-Cut Problem



$$C(\boldsymbol{x}) = 5x_1 + 9x_2 + 4x_3 + 3x_3(1-x_1) + 2x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1)$$
$$+ 1x_5(1-x_1) + 6x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6)$$
$$+ 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4)$$
$$+ 8(1-x_5) + 5(1-x_6)$$

[tutorial slides from Ľubor Ladický]

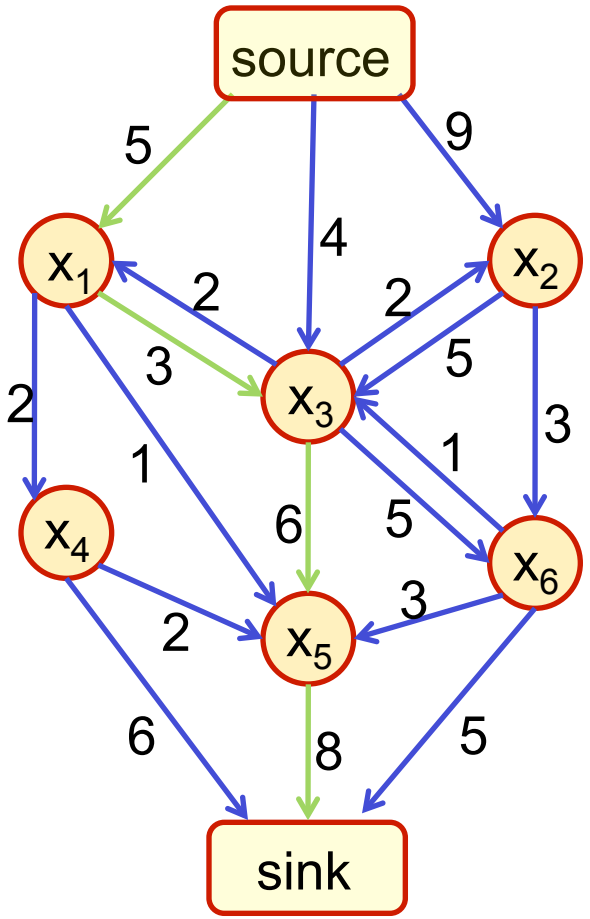
Min-Cut Problem



$$C(\mathbf{x}) = 5x_1 + 9x_2 + 4x_3 + 3x_3(1-x_1) + 2x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1)$$
$$+ 1x_5(1-x_1) + 6x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6)$$
$$+ 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4)$$
$$+ 8(1-x_5) + 5(1-x_6)$$

[tutorial slides from Ľubor Ladický]

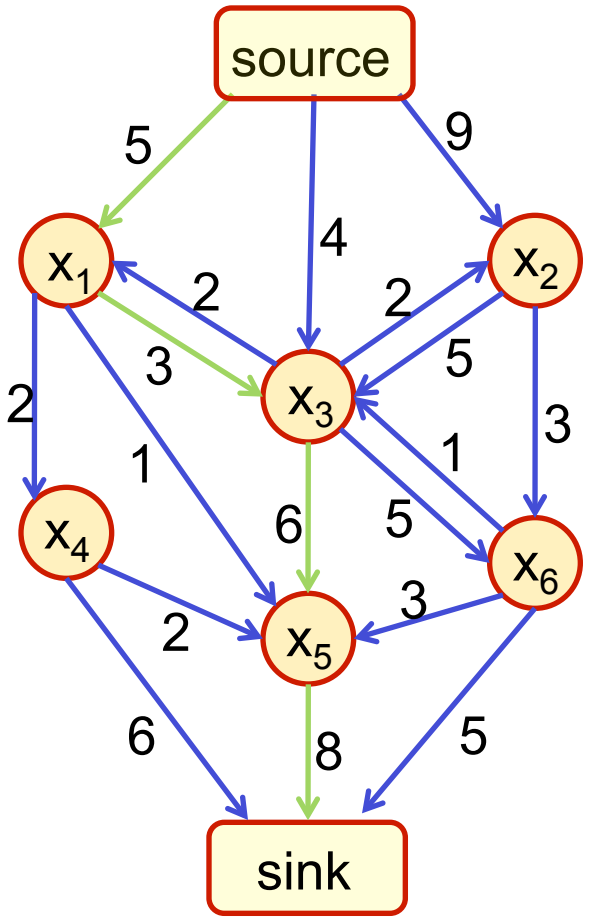
Min-Cut Problem



$$C(\boldsymbol{x}) = 2x_1 + 9x_2 + 4x_3 + 2x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1)$$
$$+ 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6)$$
$$+ 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4)$$
$$+ 5(1-x_5) + 5(1-x_6)$$
$$+ 3x_1 + 3x_3(1-x_1) + 3x_5(1-x_3) + 3(1-x_5)$$

[tutorial slides from Ľubor Ladický]

Min-Cut Problem

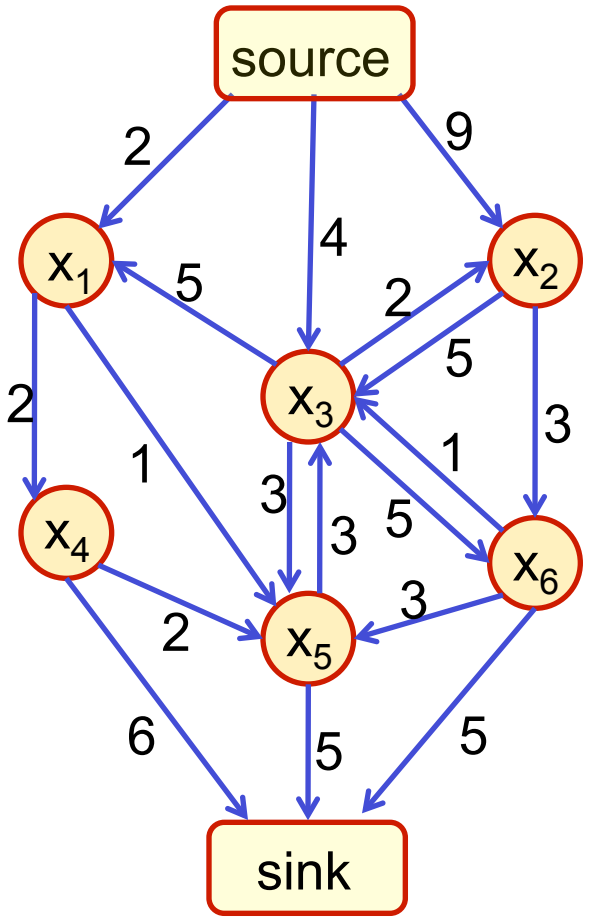


$$C(\boldsymbol{x}) = 2x_1 + 9x_2 + 4x_3 + 2x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1)$$
$$+ 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6)$$
$$+ 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4)$$
$$+ 5(1-x_5) + 5(1-x_6)$$
$$+ 3x_1 + 3x_3(1-x_1) + 3x_5(1-x_3) + 3(1-x_5)$$

$$3x_1 + 3x_3(1-x_1) + 3x_5(1-x_3) + 3(1-x_5)$$
$$=$$
$$3 + 3x_1(1-x_3) + 3x_3(1-x_5)$$

[tutorial slides from Ľubor Ladický]

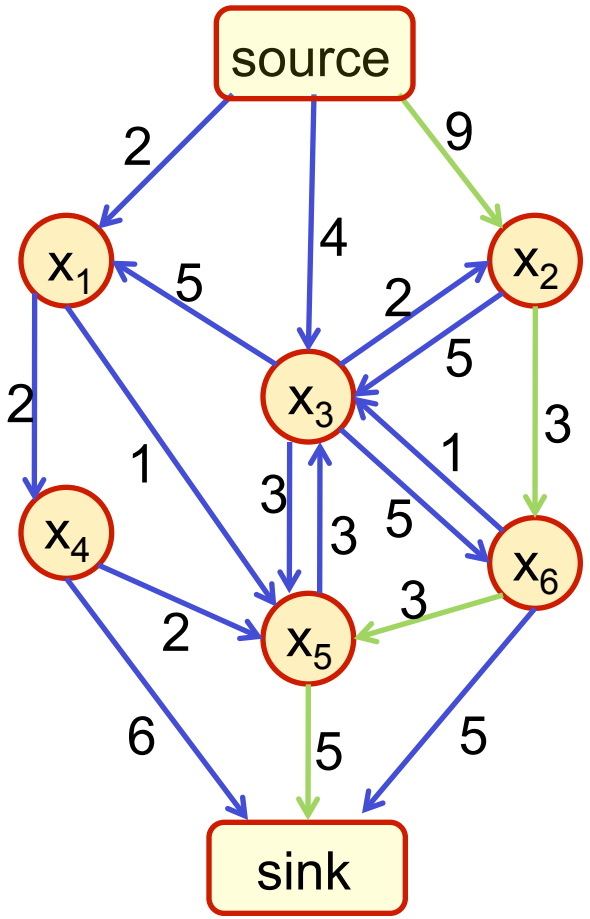
Min-Cut Problem



$$C(\boldsymbol{x}) = 3 + 2x_1 + 9x_2 + 4x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1)$$
$$+ 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6)$$
$$+ 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4)$$
$$+ 5(1-x_5) + 5(1-x_6) + 3x_3(1-x_5)$$

[tutorial slides from Ľubor Ladický]

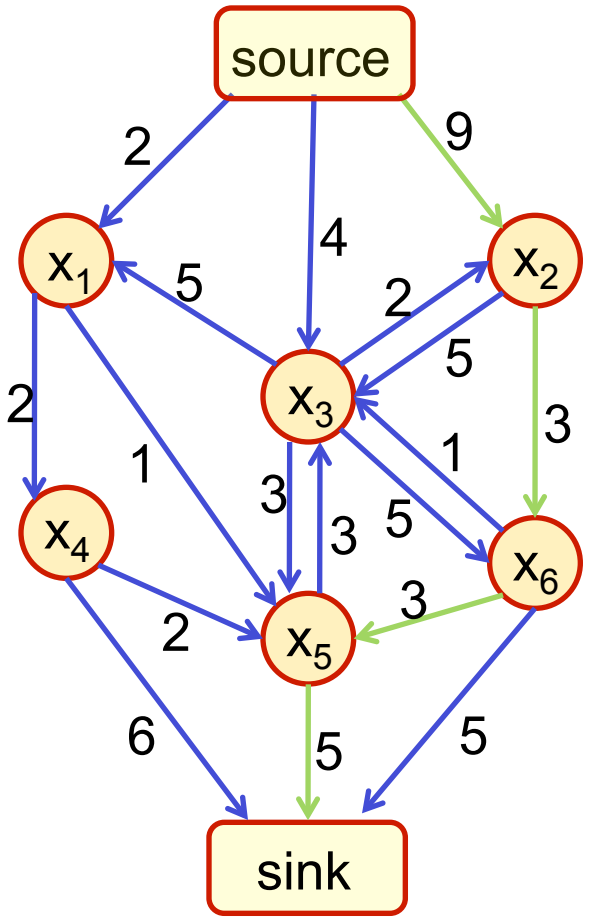
Min-Cut Problem



$$C(\boldsymbol{x}) = 3 + 2x_1 + 9x_2 + 4x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1)$$
$$+ 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6)$$
$$+ 3x_6(1-x_2) + 2x_4(1-x_5) + 3x_6(1-x_5) + 6(1-x_4)$$
$$+ 5(1-x_5) + 5(1-x_6) + 3x_3(1-x_5)$$

[tutorial slides from Ľubor Ladický]

Min-Cut Problem

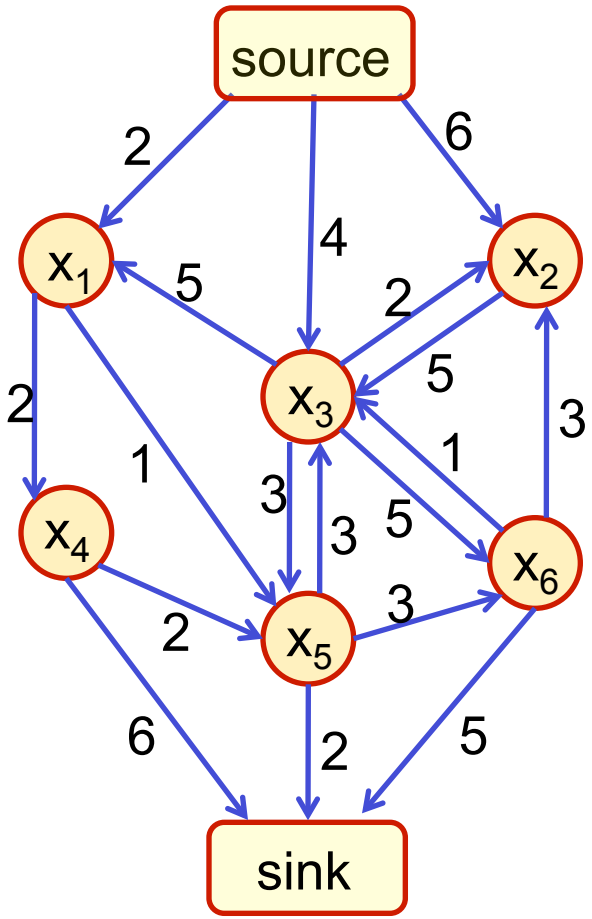


$$C(\boldsymbol{x}) = 3 + 2x_1 + 6x_2 + 4x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1)$$
$$+ 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6)$$
$$+ 2x_5(1-x_4) + 6(1-x_4)$$
$$+ 2(1-x_5) + 5(1-x_6) + 3x_3(1-x_5)$$
$$+ 3x_2 + 3x_6(1-x_2) + 3x_5(1-x_6) + 3(1-x_5)$$

$$3x_2 + 3x_6(1-x_2) + 3x_5(1-x_6) + 3(1-x_5)$$
$$=$$
$$3 + 3x_2(1-x_6) + 3x_6(1-x_5)$$

[tutorial slides from Ľubor Ladický]

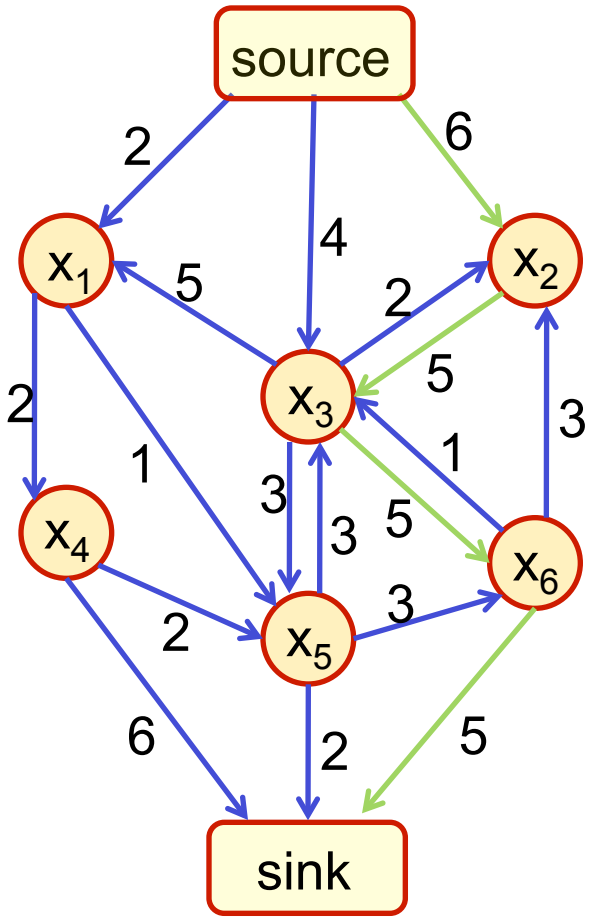
Min-Cut Problem



$$C(\boldsymbol{x}) = 6 + 2x_1 + 6x_2 + 4x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1)$$
$$+ 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6)$$
$$+ 2x_5(1-x_4) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)$$
$$+ 2(1-x_5) + 5(1-x_6) + 3x_3(1-x_5)$$

[tutorial slides from Ľubor Ladický]

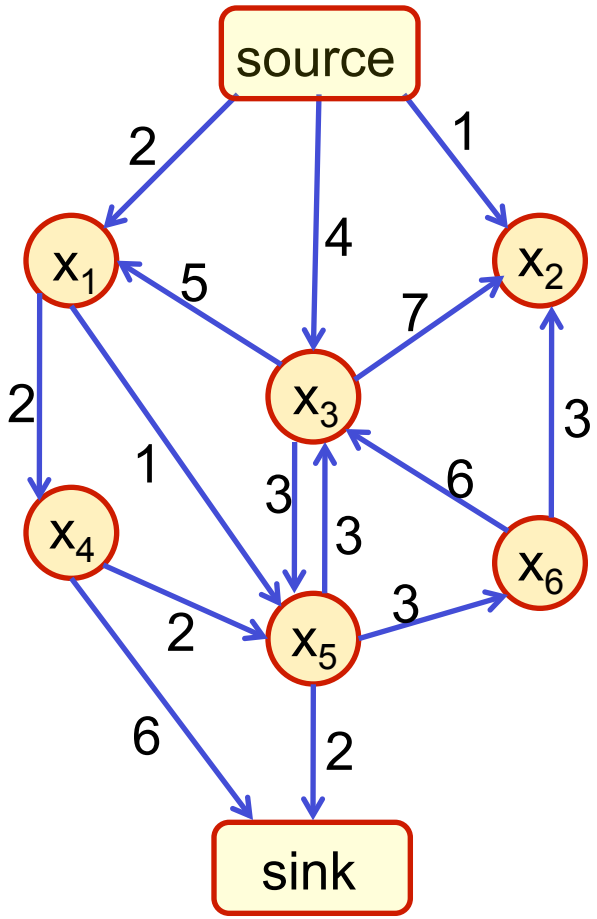
Min-Cut Problem



$$C(\boldsymbol{x}) = 6 + 2x_1 + 6x_2 + 4x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 2x_2(1-x_3) + 5x_3(1-x_2) + 2x_4(1-x_1)$$
$$+ 1x_5(1-x_1) + 3x_5(1-x_3) + 5x_6(1-x_3) + 1x_3(1-x_6)$$
$$+ 2x_5(1-x_4) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)$$
$$+ 2(1-x_5) + 5(1-x_6) + 3x_3(1-x_5)$$

[tutorial slides from Ľubor Ladický]

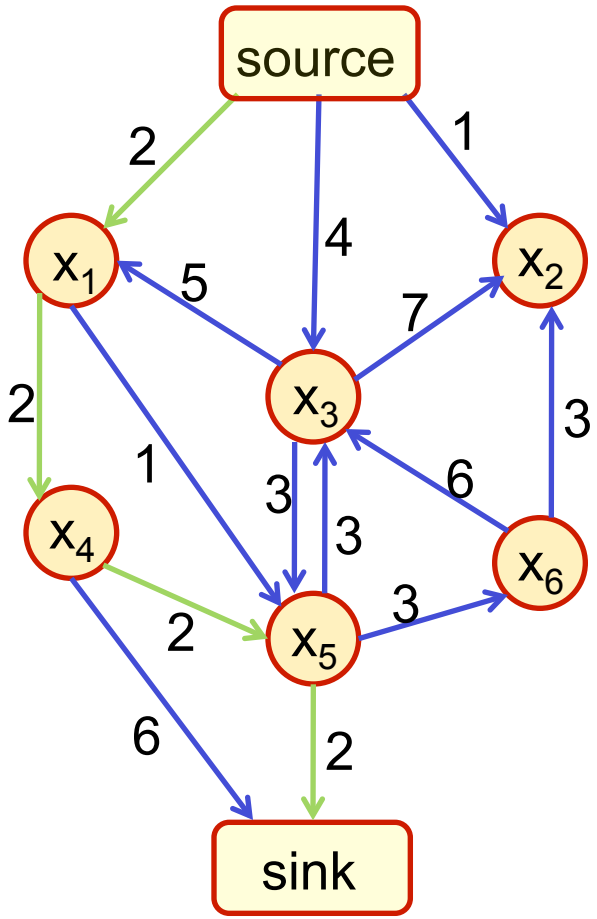
Min-Cut Problem



$$C(\boldsymbol{x}) = 11 + 2x_1 + 1x_2 + 4x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_4(1-x_1)$$
$$+ 1x_5(1-x_1) + 3x_5(1-x_3) + 6x_3(1-x_6)$$
$$+ 2x_5(1-x_4) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)$$
$$+ 2(1-x_5) + 3x_3(1-x_5)$$

[tutorial slides from Ľubor Ladický]

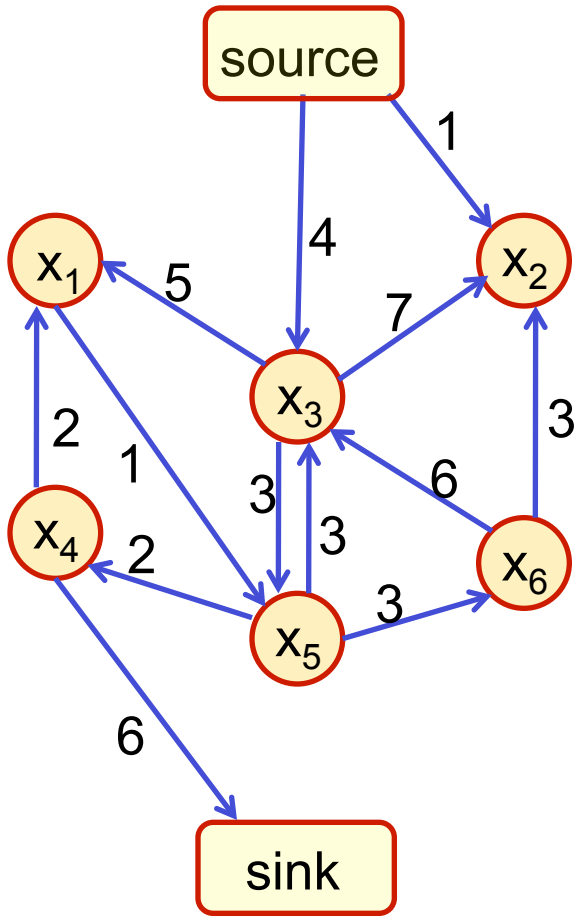
Min-Cut Problem



$$C(\boldsymbol{x}) = 11 + 2x_1 + 1x_2 + 4x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_4(1-x_1)$$
$$+ 1x_5(1-x_1) + 3x_5(1-x_3) + 6x_3(1-x_6)$$
$$+ 2x_5(1-x_4) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)$$
$$+ 2(1-x_5) + 3x_3(1-x_5)$$

[tutorial slides from Ľubor Ladický]

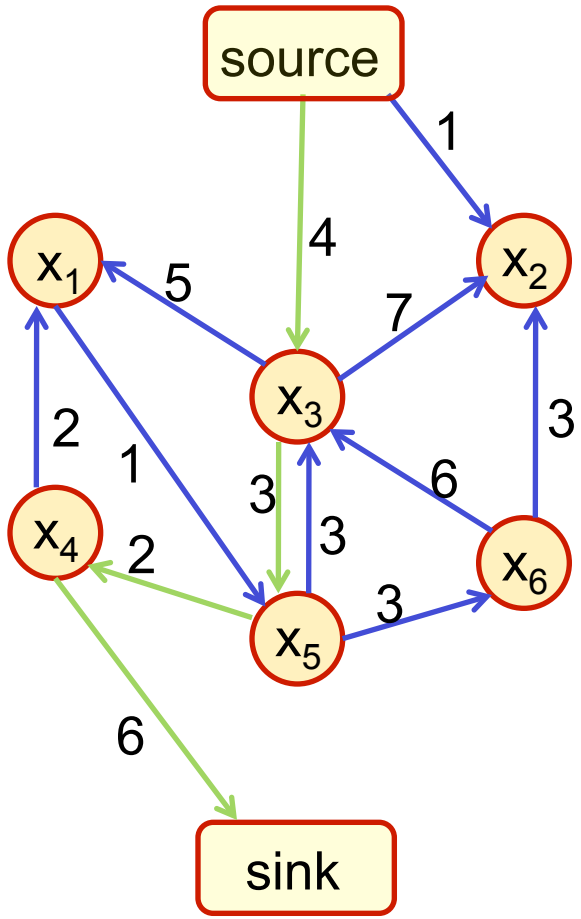
Min-Cut Problem



$$C(\boldsymbol{x}) = 13 + 1x_2 + 4x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4)$$
$$+ 1x_5(1-x_1) + 3x_5(1-x_3) + 6x_3(1-x_6)$$
$$+ 2x_4(1-x_5) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)$$
$$+ 3x_3(1-x_5)$$

[tutorial slides from Ľubor Ladický]

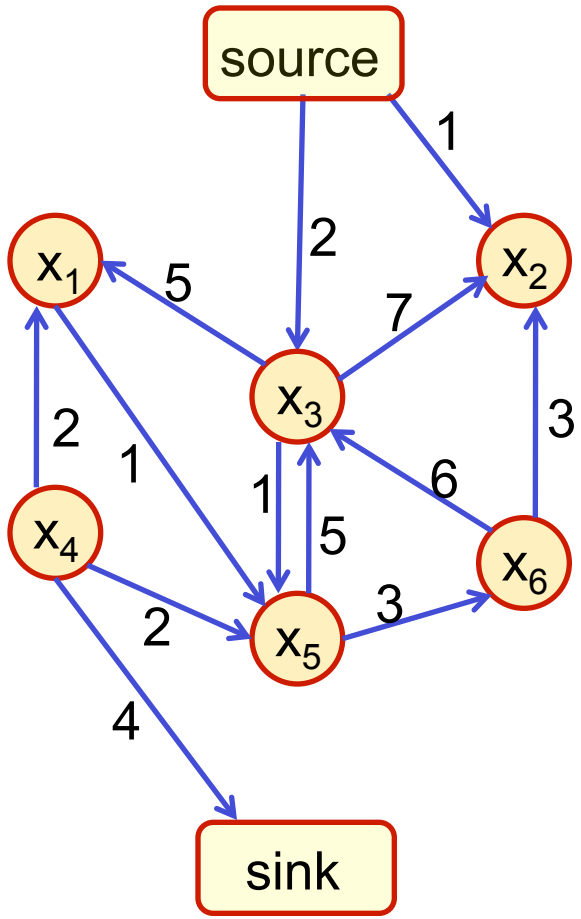
Min-Cut Problem



$$C(\boldsymbol{x}) = 13 + 1x_2 + 2x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4)$$
$$+ 1x_5(1-x_1) + 3x_5(1-x_3) + 6x_3(1-x_6)$$
$$+ 2x_4(1-x_5) + 6(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)$$
$$+ 3x_3(1-x_5)$$

[tutorial slides from Ľubor Ladický]

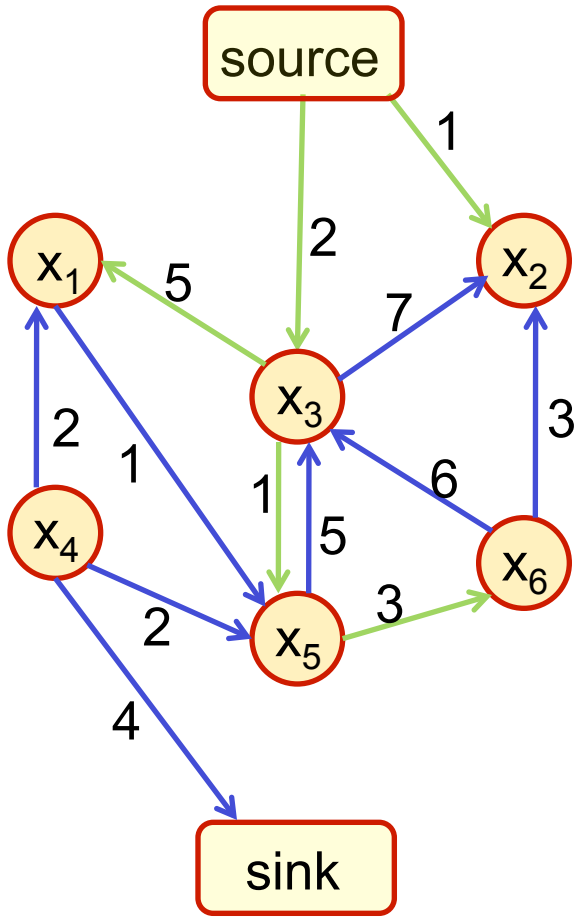
Min-Cut Problem



$$C(\boldsymbol{x}) = 15 + 1x_2 + 4x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4)$$
$$+ 1x_5(1-x_1) + 6x_3(1-x_6) + 6x_3(1-x_5)$$
$$+ 2x_5(1-x_4) + 4(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)$$

[tutorial slides from Ľubor Ladický]

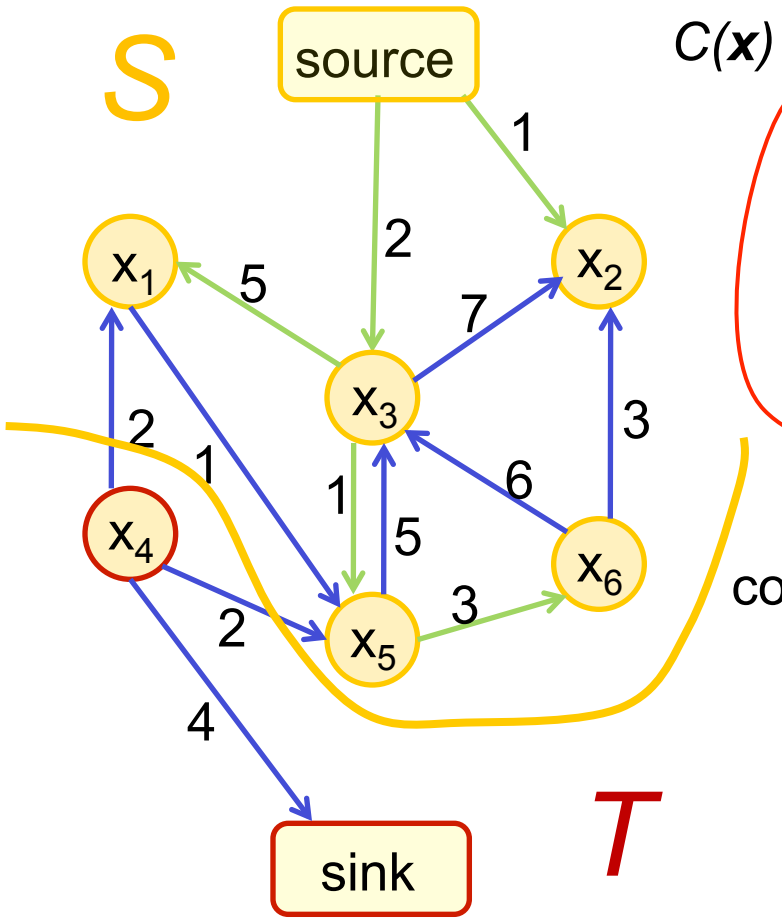
Min-Cut Problem



$$C(\boldsymbol{x}) = 15 + 1x_2 + 4x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4)$$
$$+ 1x_5(1-x_1) + 6x_3(1-x_6) + 6x_3(1-x_5)$$
$$+ 2x_5(1-x_4) + 4(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)$$

[tutorial slides from Ľubor Ladický]

Min-Cut Problem



min cut

$$C(\boldsymbol{x}) = 15 + 1x_2 + 4x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4)$$
$$+ 1x_5(1-x_1) + 6x_3(1-x_6) + 6x_3(1-x_5)$$
$$+ 2x_5(1-x_4) + 4(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)$$

cost = 0

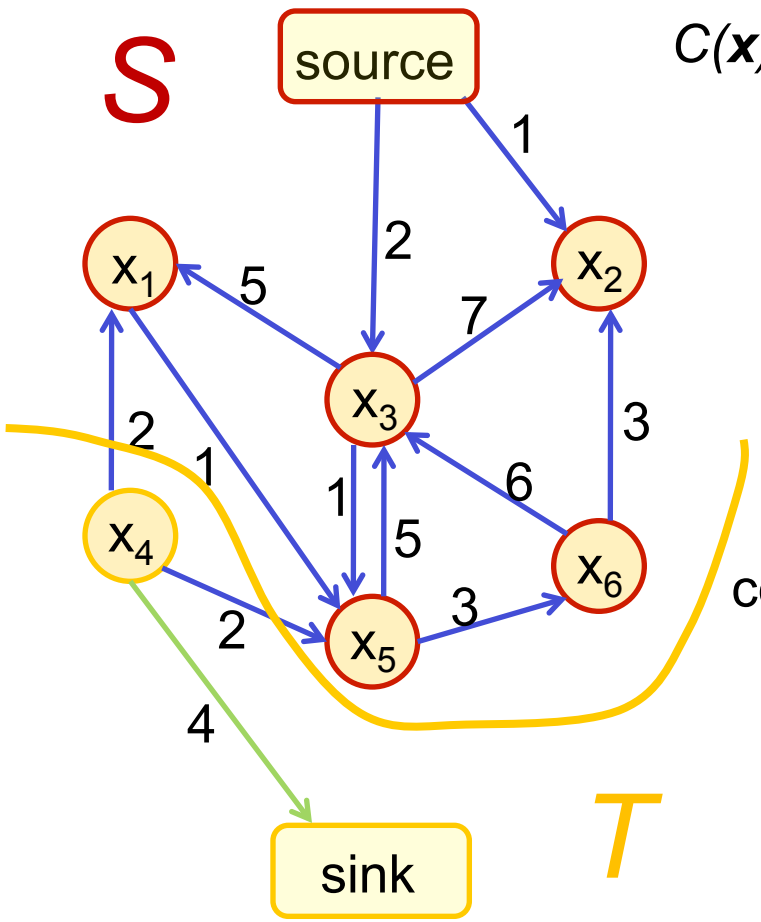
cost = 0

- All coefficients positive
- Must be global minimum

S – set of reachable nodes from s

[tutorial slides from Ľubor Ladický]

Min-Cut Problem



min cut

$$C(\boldsymbol{x}) = 15 + 1x_2 + 4x_3 + 5x_1(1-x_3)$$
$$+ 3x_3(1-x_1) + 7x_2(1-x_3) + 2x_1(1-x_4)$$
$$+ 1x_5(1-x_1) + 6x_3(1-x_6) + 6x_3(1-x_5)$$
$$+ 2x_5(1-x_4) + 4(1-x_4) + 3x_2(1-x_6) + 3x_6(1-x_5)$$

cost = 0

- All coefficients positive
- Must be global minimum

T – set of nodes that can reach t (not necessarily the same)

[tutorial slides from Ľubor Ladický]

Markov / Conditional Random Field

- Markov / Conditional Random fields model conditional dependencies between random variables
- Each variable is conditionally independent of all other variables given its neighbors
- Posterior probability of the labeling *x* given data *D* is :

$$\Pr(\mathbf{x}|\mathbf{D}) = \frac{1}{Z}\exp\Big(-\sum_{c\in\mathcal{C}}\psi_c(\mathbf{x}_c)\Big)$$

partition function — cliques — potential functions

- Energy of the labeling is defined as:

$$E(\mathbf{x}) = -\log\Pr(\mathbf{x}|\mathbf{D}) - \log Z = \sum_{c\in\mathcal{C}}\psi_c(\mathbf{x}_c)$$

[tutorial slides from Ľubor Ladický]

Markov / Conditional Random Field

- The most probable (Max a Posteriori (MAP)) labelling is defined as:

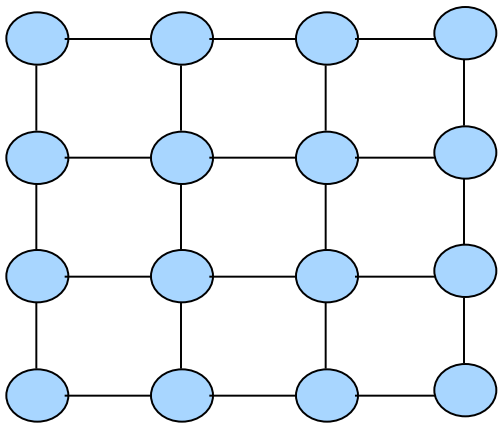
$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathbf{L}} \Pr(\mathbf{x}|\mathbf{D}) = \arg\min_{\mathbf{x}\in\mathbf{L}} E(\mathbf{x})$$

- The only distinction (MRF vs. CRF) is that in the CRF the conditional dependencies between variables depend also on the data
- Typically we define an energy first and then pretend there is an underlying probabilistic distribution there, but there isn't really (Psssst, don't tell anyone)

[tutorial slides from Ľubor Ladický]

Pairwise CRF models

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j)$$



What is the most general pairwise energy solvable using GraphCut ?

Submodular !

[tutorial slides from Ľubor Ladický]

Graph-cut based Inference

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j)$$

Required conditions for graph-cut based inference:

- submodularity: An energy / potential is called **submodular** iff for every pair of variables :

$$E(0,0,\bar{\mathbf{x}}_{ij}) + E(1,1,\bar{\mathbf{x}}_{ij}) \leq E(0,1,\bar{\mathbf{x}}_{ij}) + E(1,0,\bar{\mathbf{x}}_{ij})$$

- pairwise potentials: positiveness $\psi_{ij}(x_i, x_j) = \psi_{ji}(x_j, x_i) \geq 0$

  (optional) metric $\psi_{ij}(x_i, x_j) \leq \psi_{ik}(x_i, x_k) + \psi_{kj}(x_k, x_j)$

some algorithms (e.g. alpha-expansion) require the metric condition, others don't (e.g. alpha-beta swap)

[tutorial slides from Ľubor Ladický]

Graph-cut Applications

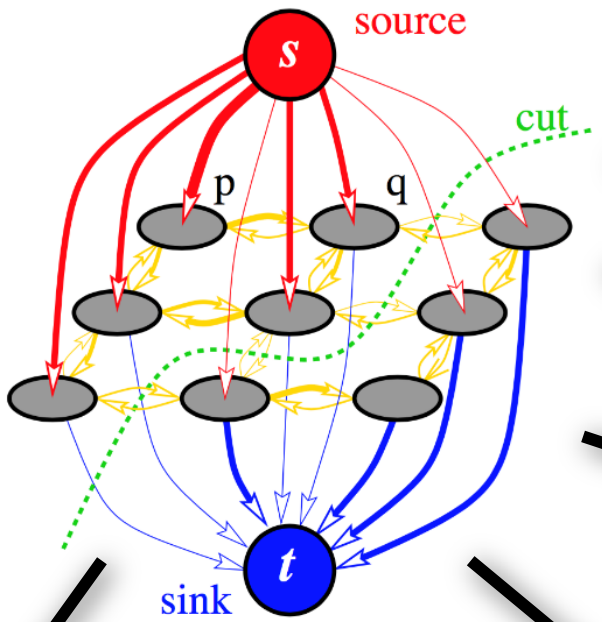


Image Segmentation

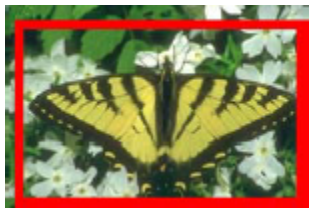
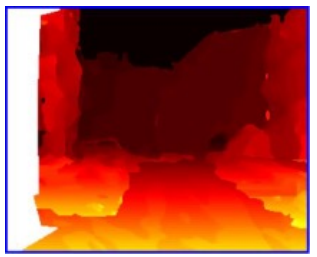
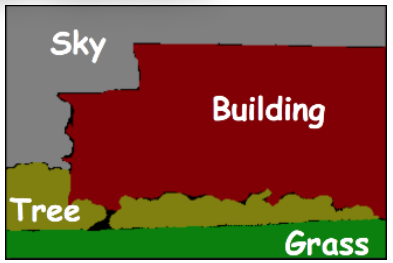


Image Denoising

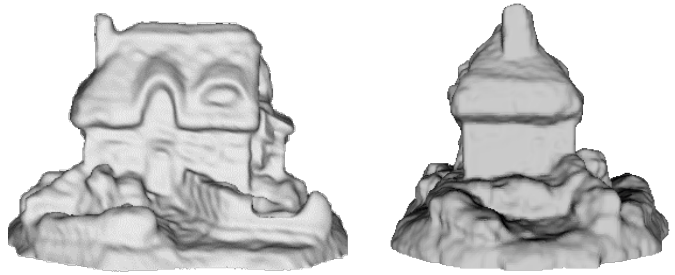
Depth Estimation



Object Segmentation

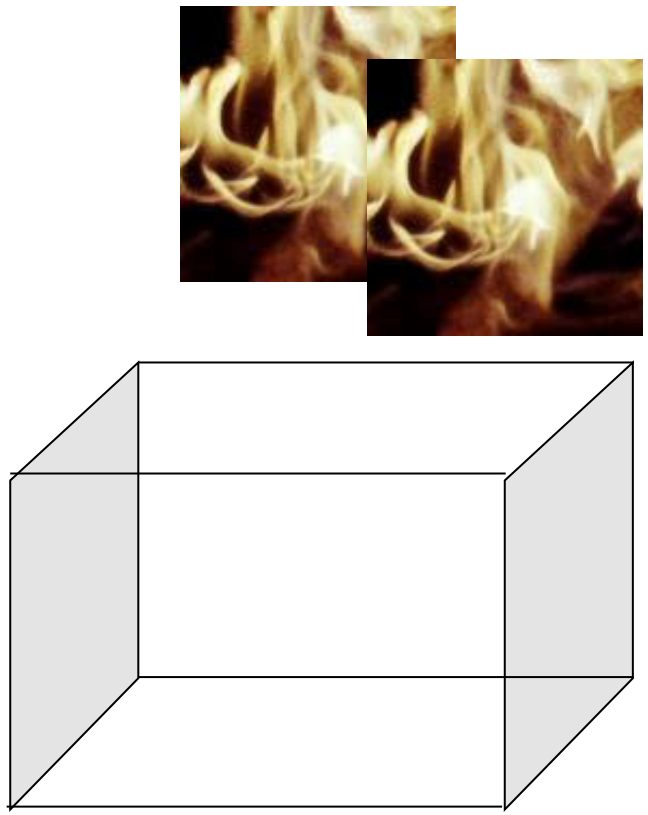


3D Reconstruction

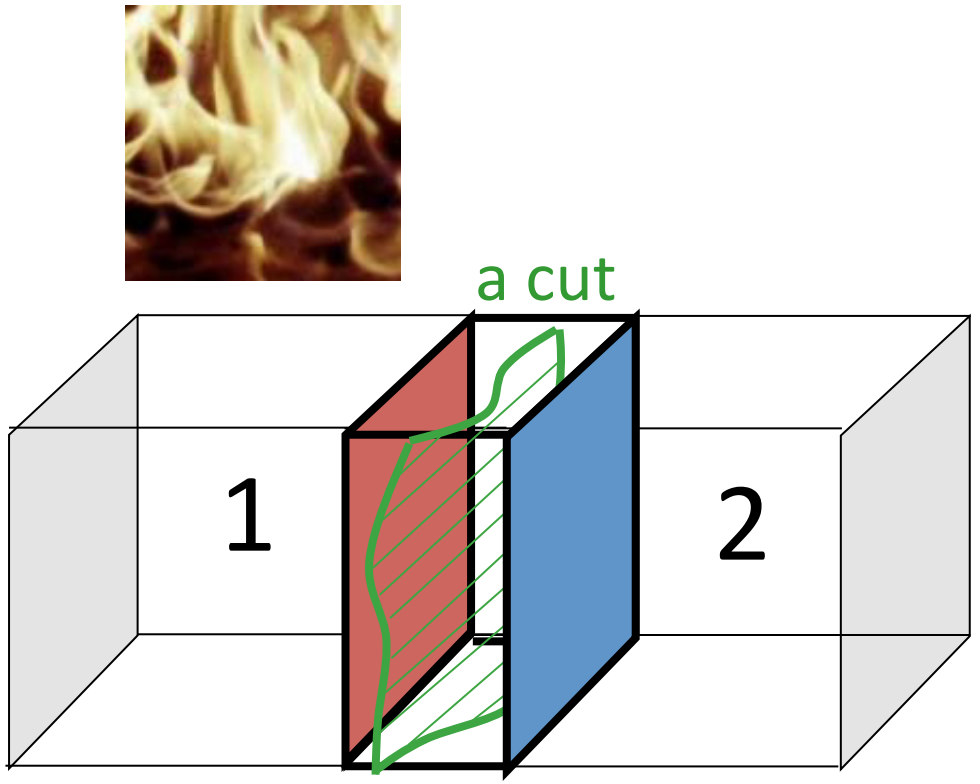


Graph-cuts for Video Textures

Graphcut Textures - Kwatra, Schodl, Essa, Bobick, SIGGRAPH 2003



Short video clip

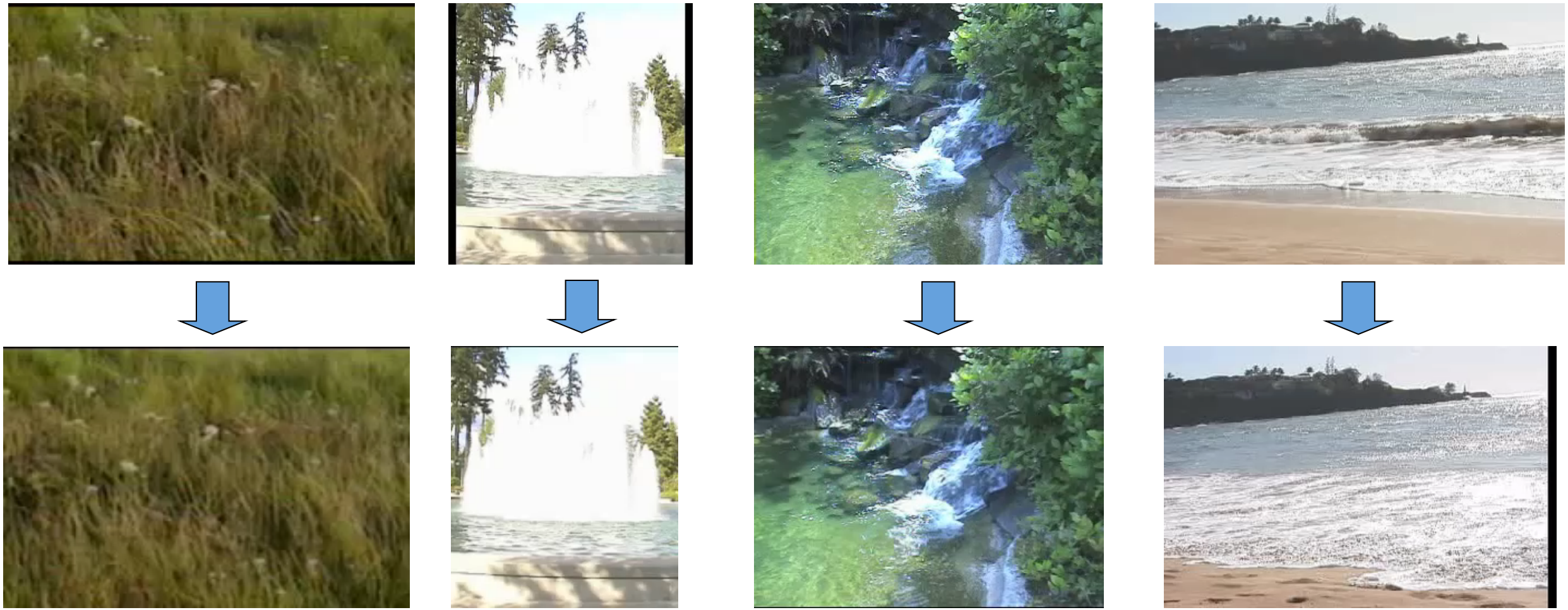


Long video clip

[ECCV'06 tutorial slides from Boykov, Cremers, Kolmogorov]

Graph-cuts for Video Textures

Graphcut Textures - Kwatra, Schödl, Essa, Bobick, SIGGRAPH 2003



Graph-cuts for Texture Synthesis

Graphcut Textures - Kwatra, Schödl, Essa, Bobick, SIGGRAPH 2003

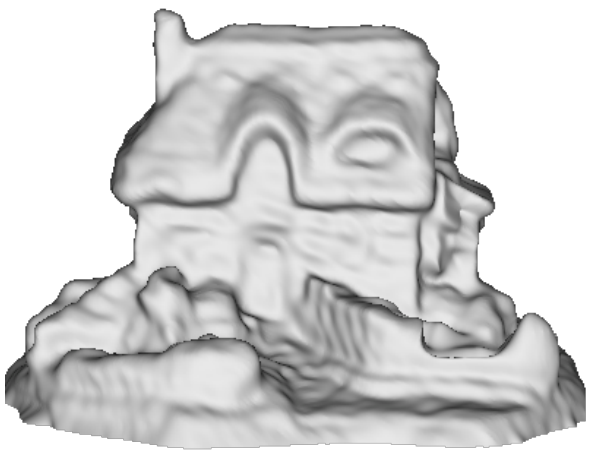





Multi-view Reconstruction

Calibrated images of Lambertian scene

[Vogiatzis, Torr, Cippola, CVPR 2005]

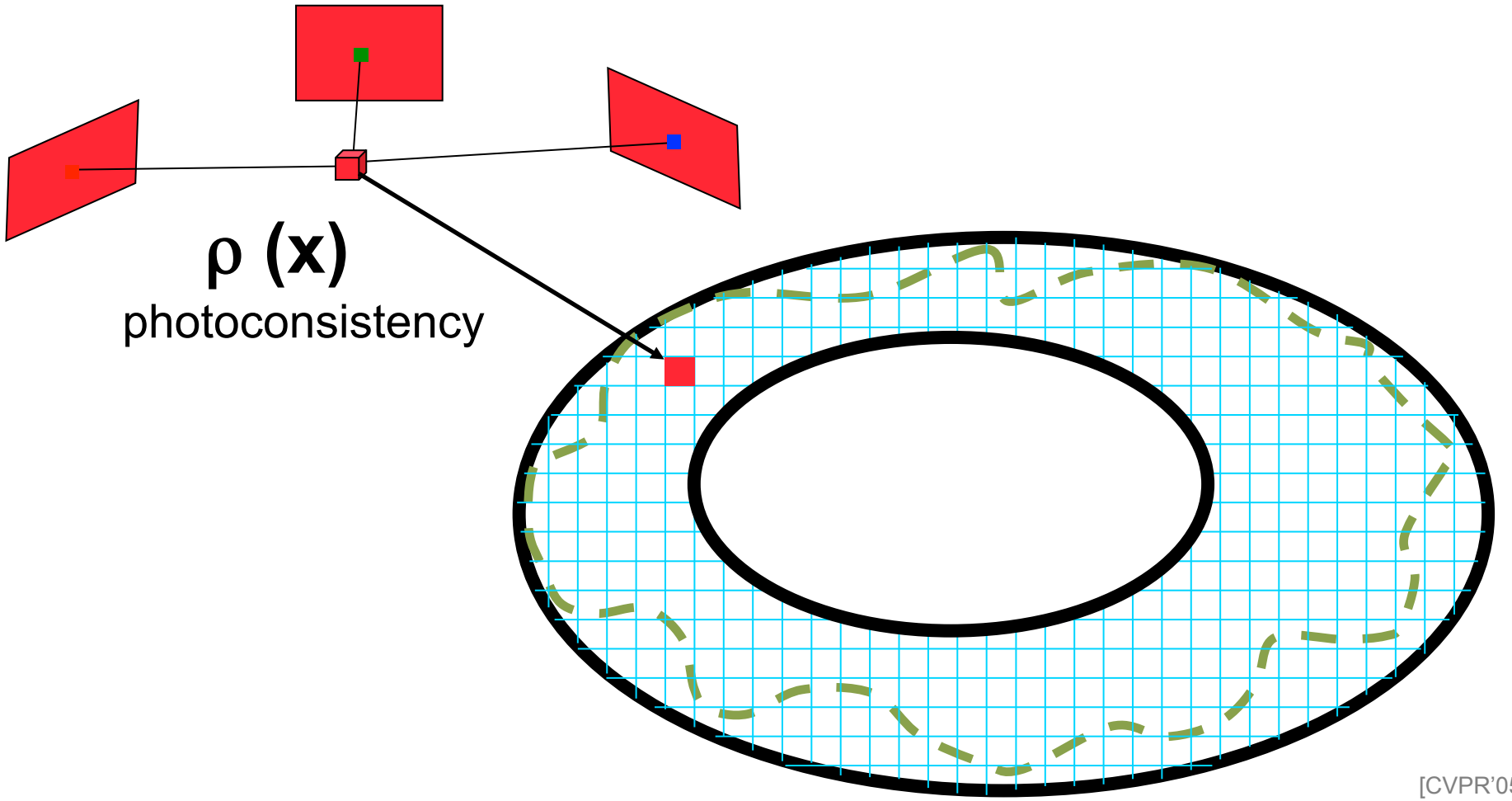


3D model of scene

[CVPR'05 slides from Vogiatzis, Torr, Cippola]

Multi-view Reconstruction

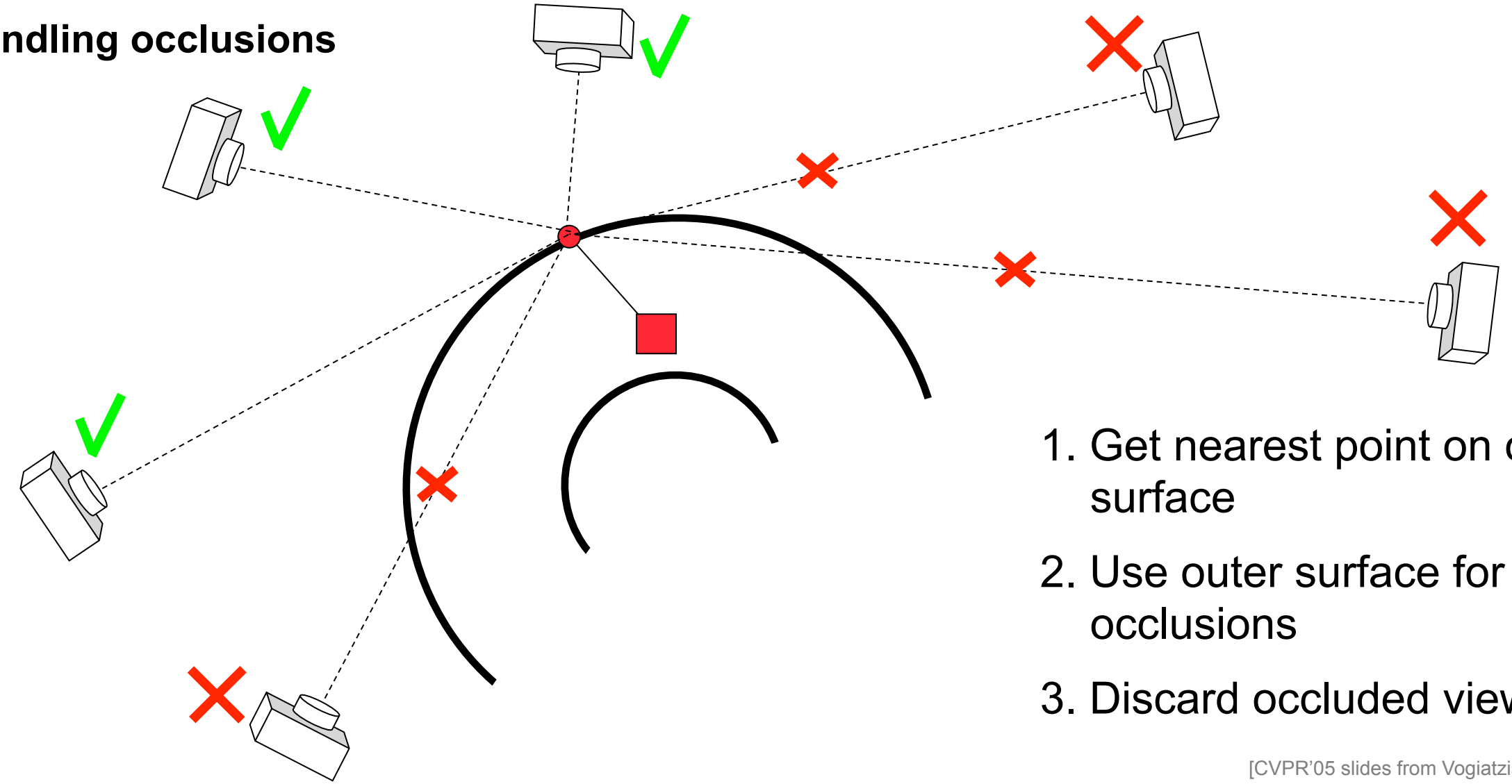
Estimating photoconsistency within a narrow band.



[CVPR'05 slides from Vogiatzis, Torr, Cippola]

Multi-view Reconstruction

Handling occlusions

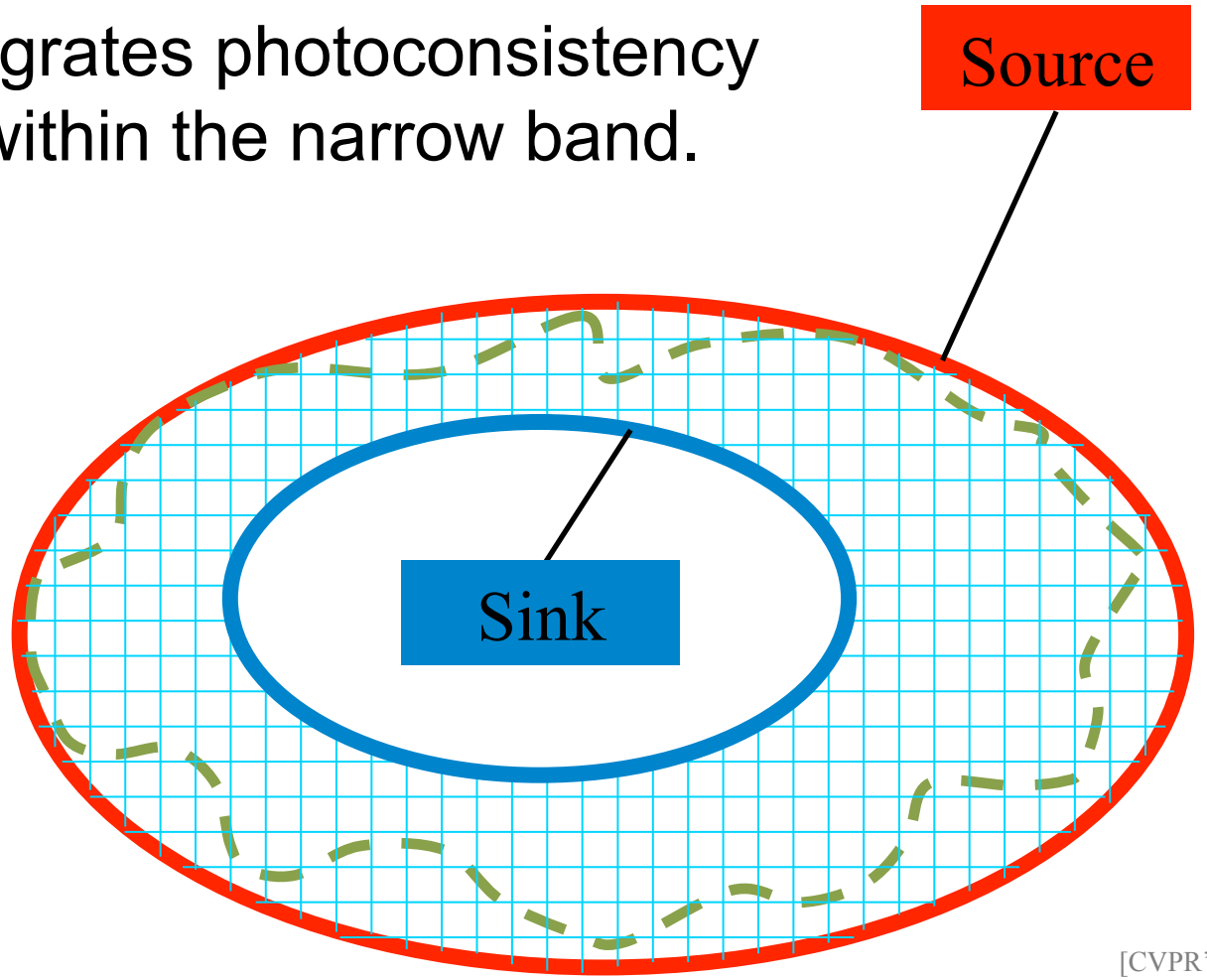


1. Get nearest point on outer surface
2. Use outer surface for occlusions
3. Discard occluded views

[CVPR'05 slides from Vogiatzis, Torr, Cippola]

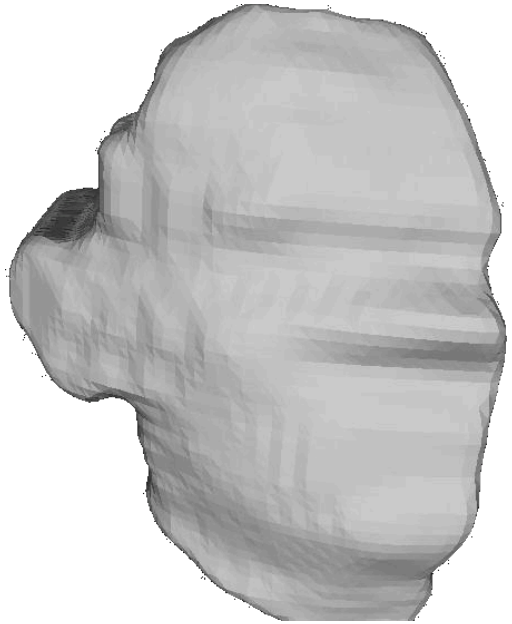
Multi-view Reconstruction

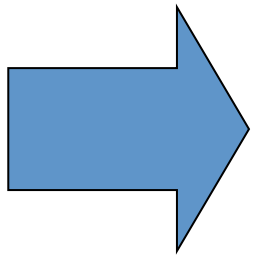
The cost of the cut integrates photoconsistency over the whole space within the narrow band.

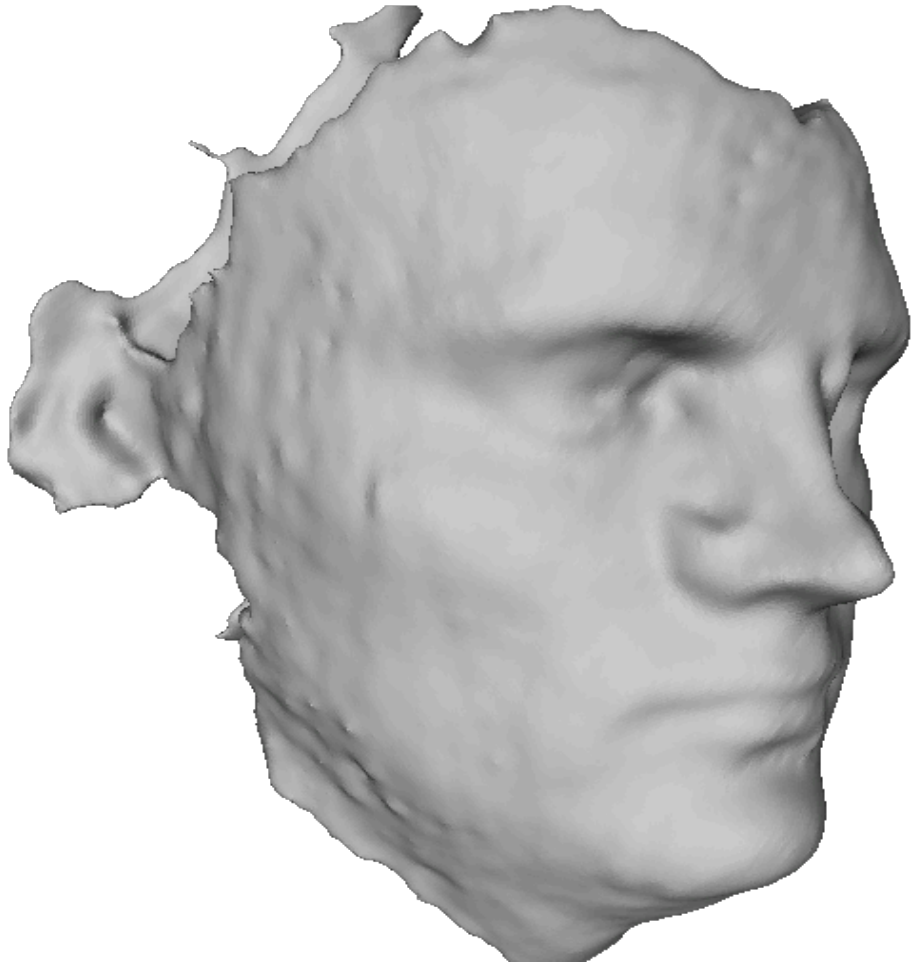


[CVPR'05 slides from Vogiatzis, Torr, Cippola]

Multi-view Reconstruction







visual hull (silhouettes)
narrow band for graph-cut

best photo-consistent surface

[CVPR'05 slides from Vogiatzis, Torr, Cippola]

Multi-label Graph-cuts

For solving non-binary / multi-way / multi-label graph-cuts, the problem can be transformed into solving a **sequence of binary graph-cuts** to find a **local optimum.**

Two algorithms were proposed by Boykov, Veksler, Zabih, PAMI 2001:

- Alpha-beta-swap (requires only semi-metric)
- Alpha-expansion (requires metric condition)

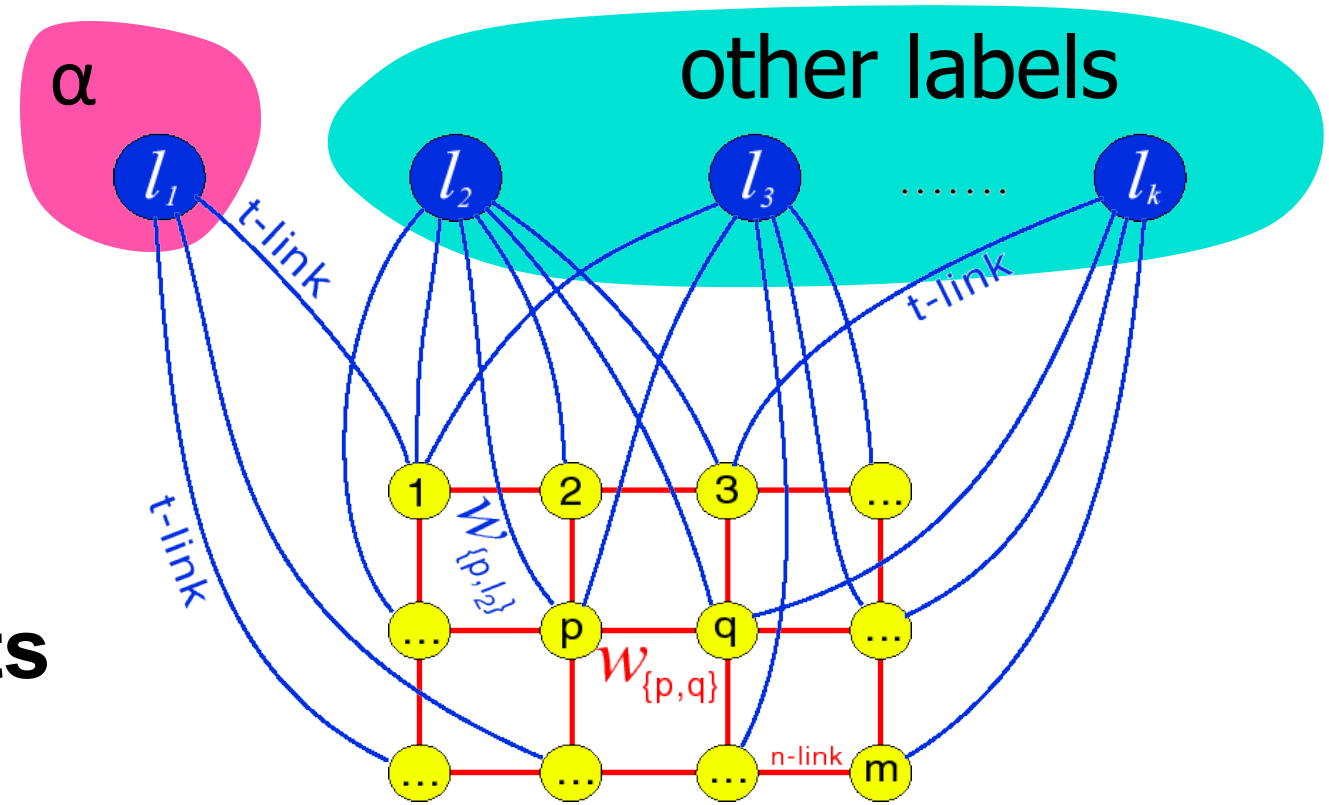
Alpha-expansion algorithm

[Y. Boykov, O. Veksler, R. Zabih, PAMI 2001]

Idea: Break problem into smaller sub-problems.

Multi-way cut

Sequence of binary s-t cuts



[ECCV'06 tutorial slides from Boykov, Cremers, Kolmogorov]

Alpha-expansion algorithm

[Y. Boykov, O. Veksler, R. Zabih, PAMI 2001]

1. Start with any initial solution
2. For each label "a" in any (e.g. random) order

    1. *Compute optimal* a*-expansion move (s-t graph cut) that most decreases E*
    2. *Decline the move if there is no energy decrease*

3. *Stop when no expansion move would decrease energy*

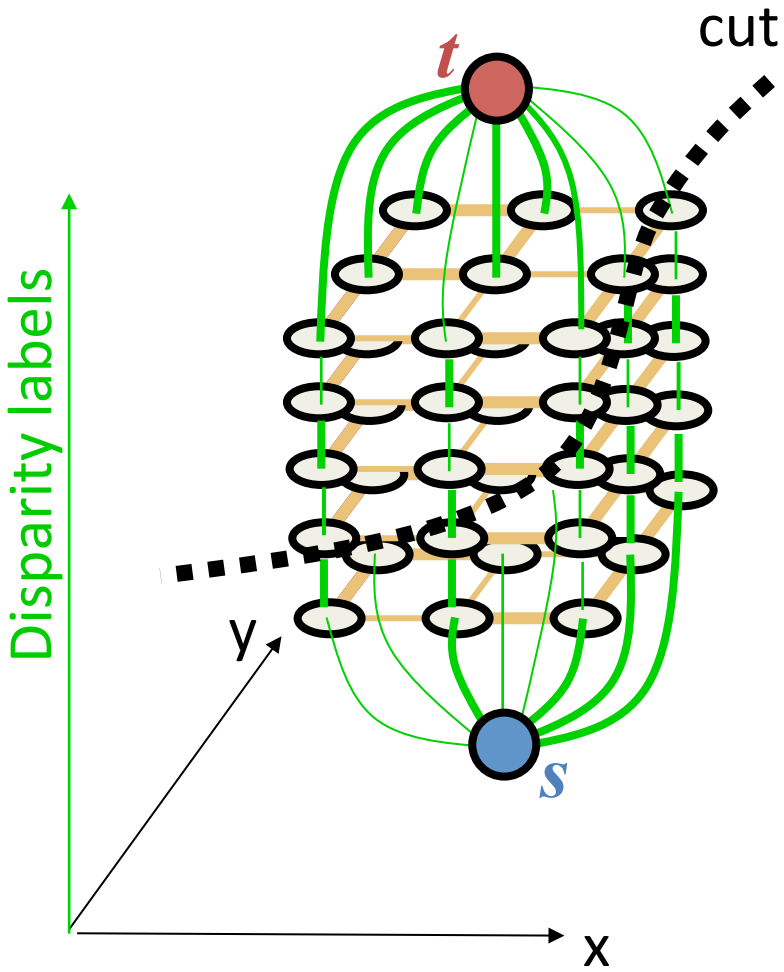
Note: Different orders of expansion moves might lead to different solutions (local minimum).

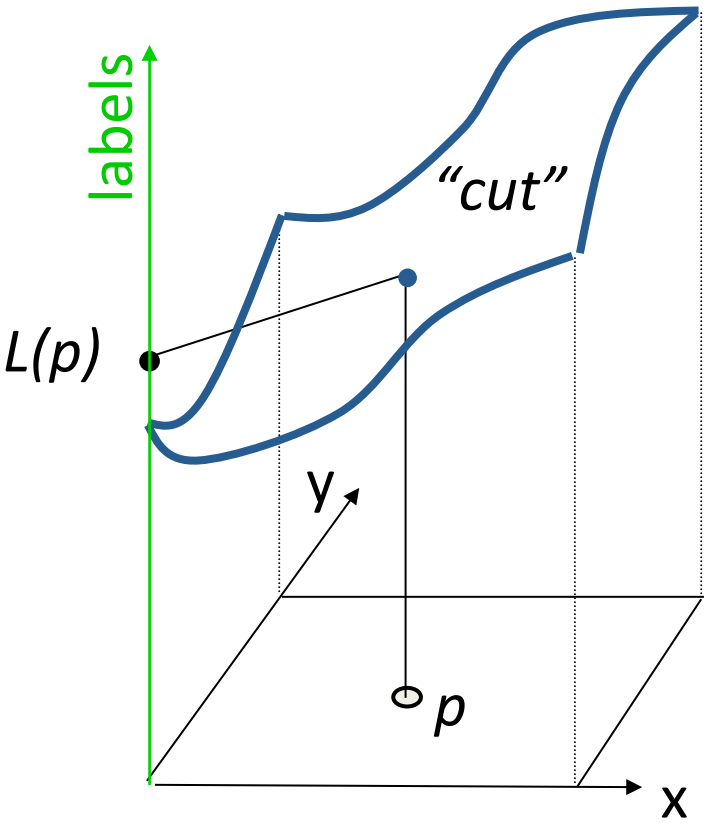
[ECCV'06 tutorial slides from Boykov, Cremers, Kolmogorov]

Example: Stereo Reconstruction

depth map

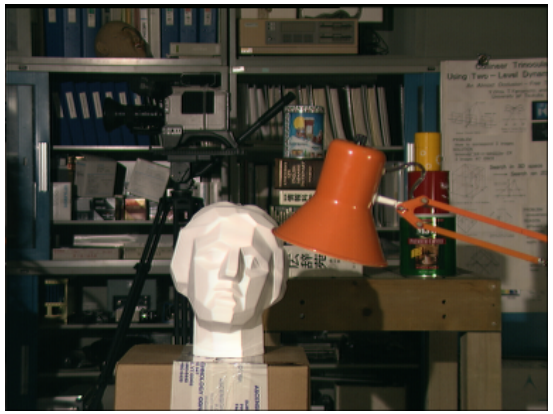
[Roy, Cox, ICCV 1998]

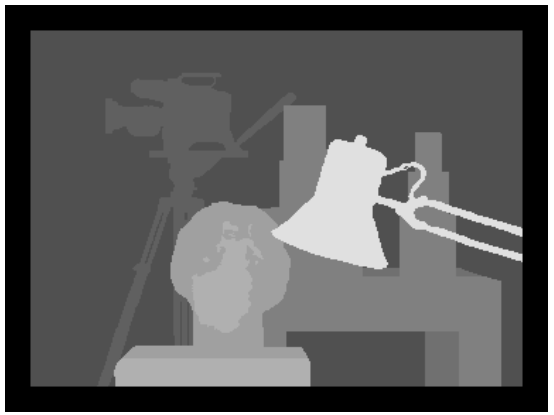




[ECCV'06 tutorial slides from Boykov, Cremers, Kolmogorov]

Example: Stereo Reconstruction





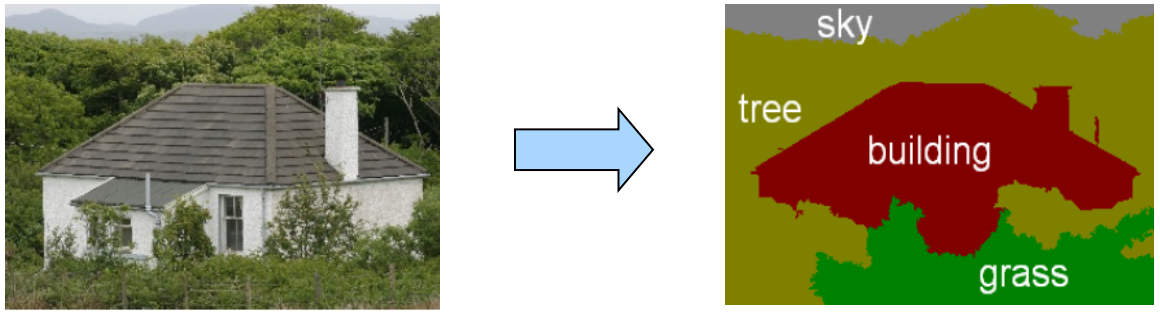
initial solution

- -expansion
- -expansion
- -expansion
- -expansion
- -expansion
- -expansion
- -expansion

For each move we choose the expansion the gives the largest energy decrease.

[ECCV'06 tutorial slides from Boykov, Cremers, Kolmogorov]

Example: Object-class Segmentation



$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j)$$

Data term **Smoothness term**

Data term Discriminatively trained classifier

Smoothness term

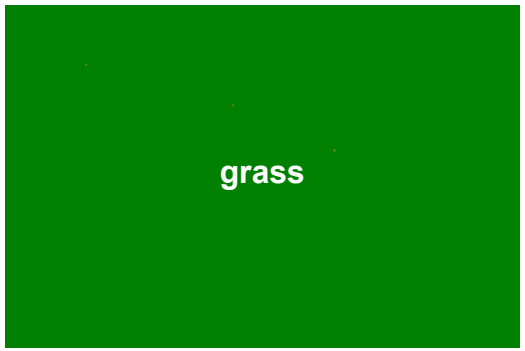
$$\psi_{ij}(x_i, x_j) = K_{ij}\delta(x_i \neq x_j)$$

where $K_{ij} = \lambda_1 + \lambda_2 \exp(-\beta(I_i - I_j)^2)$

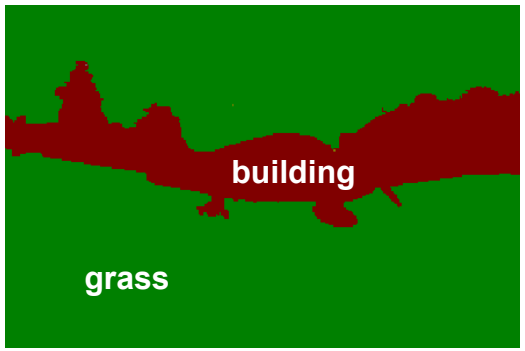
[tutorial slides from Ľubor Ladický]

Example: Object-class Segmentation

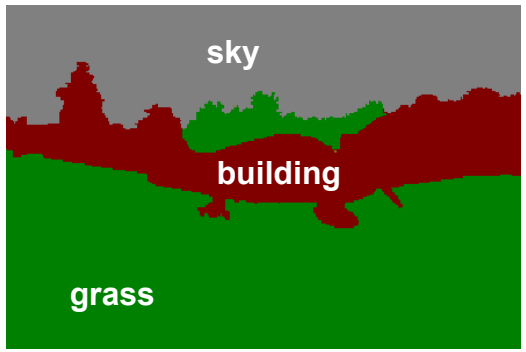
Original Image



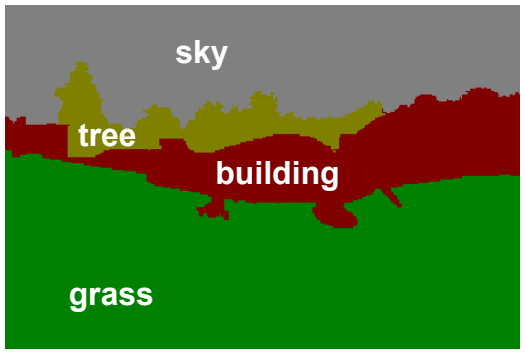
Initial solution



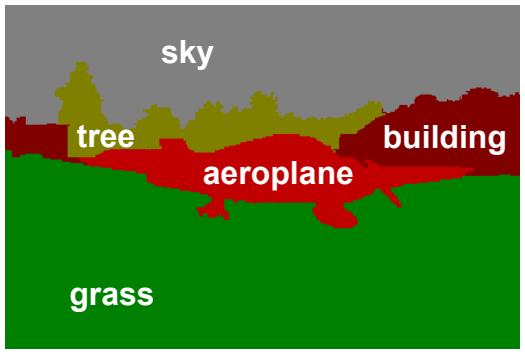
Building expansion



Sky expansion



Tree expansion



Final Solution

[tutorial slides from Ľubor Ladický]

Questions?

References

- S. Roy, I.J. Cox, "*A Maximum-Flow Formulation of the N-camera Stereo Correspondence Problem*", ICCV 1998
- Y. Boykov, O. Veksler, R. Zabih, "*Fast Approximate Energy Minimization via Graph Cuts*", PAMI 2001
- Y. Boykov and M. Jolly. "*Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images.*", In IEEE International Conference on Computer Vision (ICCV), 2001
- V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "*Graphcut Textures: Image and Video Synthesis Using Graph Cuts*", ACM Transactions on Graphics, SIGGRAPH 2003
- V. Kolmogorov and R. Zabih, "*What energy functions can be minimized via graph cuts?*", PAMI 2004
- Y. Boykov and V. Kolmogorov, "*An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision*", PAMI 2004
- C. Rother, V. Kolmogorov, and A. Blake. "*GrabCut: Interactive foreground extraction using iterated graph cuts.*" In ACM SIGGRAPH (ACM Transactions on Graphics), 2004.
- G. Vogiatzis, P.H.S. Torr, R. Cipolla, "*Multi-view Stereo via Volumetric Graph-cuts.*", CVPR 2005
- Ford-Fulkerson Algorithm Applet: http://web.stanford.edu/~ashishg/msande212/MaxFlowJavaDemo/Java%20Applet%20Demos%20of%20Ford-Fulkerson's%20Algorithm.htm