Graph Cuts Approach to the Problems of Image Segmentation

CS7640 Ross Whitaker

(Adapted from Antong Chen)

Outline

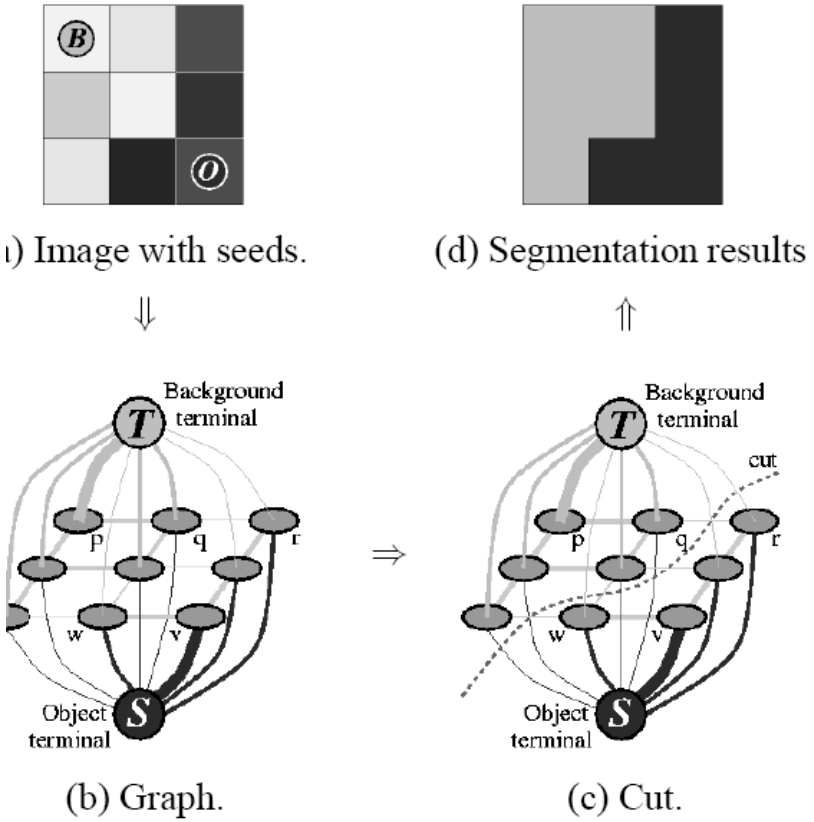
- Image Segmentation and Graph Cuts – Introduction
- Glance at Graph Theory
- Graph Cuts and Max-Flow/Min-Cut Algorithms
- Solving Image Segmentation Problem
- Experimental Examples

Image Segmentation and Graph Cuts – Introduction

- An image segmentation problem can be interpreted as partitioning the image elements (pixels/voxels) into different categories
- A Cut of a graph is a partition of the vertices in the graph into two disjoint subsets
- Constructing a graph with an image, we can solve the segmentation problem using techniques for graph cuts in graph theory

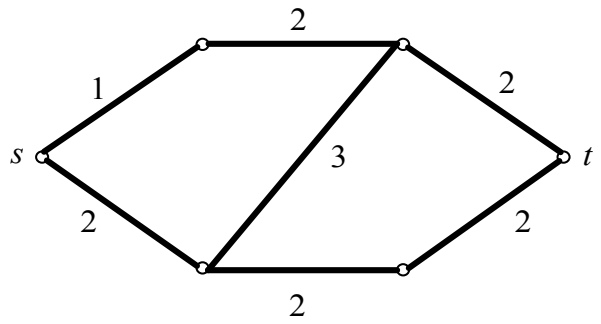
Constructing a Graph from an Image

- Two kinds of vertices
- Two kinds of edges
- Cut - Segmentation



Glance at Graph Theory

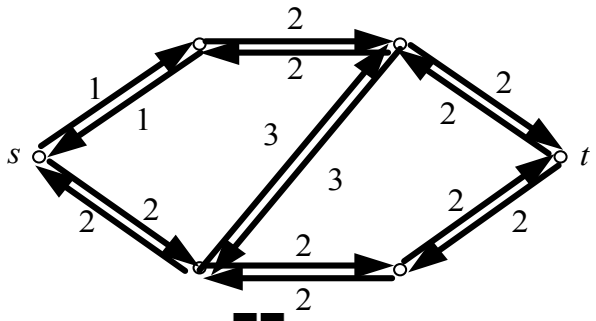
- Undirected Graph
- An undirected graph $G = (V, E)$ is defined as a set of nodes (vertices V) and a set of undirected edges E that connect the nodes.
- Assigning each edge $e \in E$ a weight w_e, the graph becomes an undirected weighted graph.



Glance at Graph Theory

- Directed Graph
  - A directed graph is defined as a set of nodes (vertices V) and a set of ordered set of vertices or directed edges E that connect the nodes
- For an edge e=(u,v), u is called the tail of e, v is called the head of e.
- This edge is different from the edge e'=(v,u)

$$G = (V, E)$$
$$e' = (v, u)$$
$$e = (u, v)$$



Glance at Graph Theory

- A cut is a set of edges $C \subset E$ such that the two terminals become separated on the induced graph

$$G' = (V, E \setminus C)$$

- Denoting a source terminal as s and a sink terminal as t, a cut (S, T) of $G = (V, E)$ is a partition of V into S and T = V \S, such that $t \in T$ and $s \in S$

Graph Cuts and Max-Flow/Min-Cut Algorithms

- A flow network is defined as a directed graph where an edge has a nonnegative capacity
- A flow in G is a real-valued (often integer) function that satisfies the following three properties:
  - Capacity Constraint:
  - For all $u, v \in V, f(u,v) \leq c(u,v)$
  - Skew Symmetry
  - For all $u, v \in V, f(u,v) = -f(v,u)$
  - Flow Conservation
  - For all $u \in (V \setminus \{s,t\}), \sum_{v \in V} f(u,v) = 0$

How to Find the Minimum Cut?

- Theorem:
  - In graph G, the maximum source-to-sink flow possible is equal to the capacity of the minimum cut in G.
  - (L. R. Foulds, Graph Theory Applications, 1992 Springer-Verlag New York Inc., 247-248)

Maximum Flow and Minimum Cut Problem

- Some Concepts
  - If f is a flow, then the net flow across the cut (S, T) is defined to be f (S, T), which is the sum of all edge capacities from S to T subtracted by the sum of all edge capacities from T to S
  - The capacity of the cut (S, T) is c (S, T), which is the sum of the capacities of all edge from S to T
  - A minimum cut is a cut whose capacity is the minimum over all cuts of G.

Algorithms to Solve Max-Flow Problem

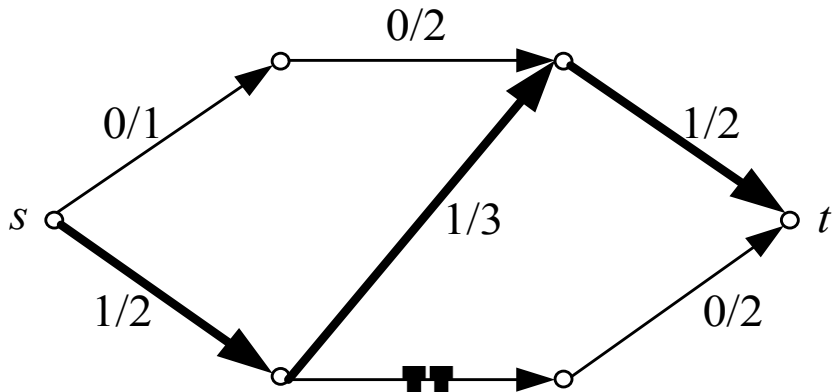
- Ford-Fulkerson Algorithm
- Push-Relabel Algorithm
- New Algorithm by Boykov, etc.

Ford-Fulkerson Algorithm

- Main Operation
- Starting from zero flow, increase the flow gradually by finding a path from s to t along which more flow can be sent, until a max-flow is achieved
- The path for flow to be pushed through is called an augmenting path

Ford-Fulkerson Algorithm

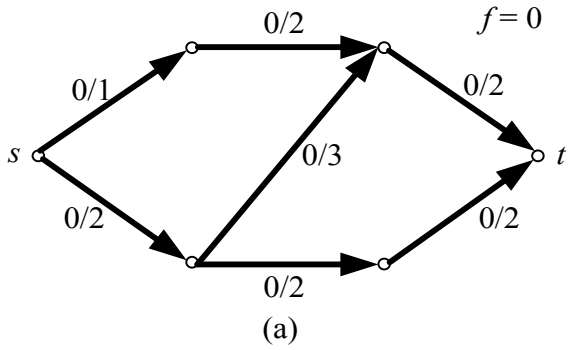
- The Ford-Fulkerson algorithm uses a residual network of flow in order to find the solution
- The residual network is defined as the network of edges containing flow that has already been sent
- In the graph shown below, there is an initial path from the source to the sink, and the middle edge has a total capacity of 3, and a residual capacity of 3-1=2



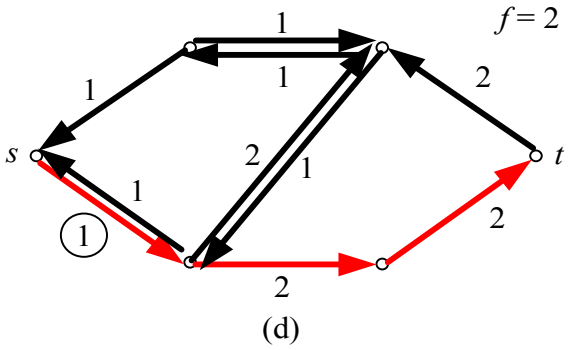
Ford-Fulkerson Algorithm

- Assuming there are two vertices, u and v, let f (u, v) denote the flow between them, c (u, v) be the total capacity, cf (u, v) be the residual capacity, and there should be,
  - cf (u, v) = c (u, v) – f (u, v)
- Given a flow network and a flow f, the residual network of G is $G_f = (V, E_f)$, where $E_f = \{(u,v) \in V \times V : c_f(u,v) > 0\}$
- Given a flow network and a flow f, an augmenting path P is a simple path from s to t in the residual network
- We call the maximum amount by which we can increase the flow on each edge in an augmenting path P the residual capacity of P, given by,
  - cf (P) = min{cf (u, v): (u, v) is on P}

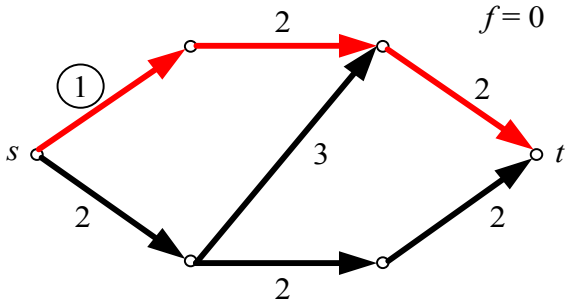
Algorithm Execution Example



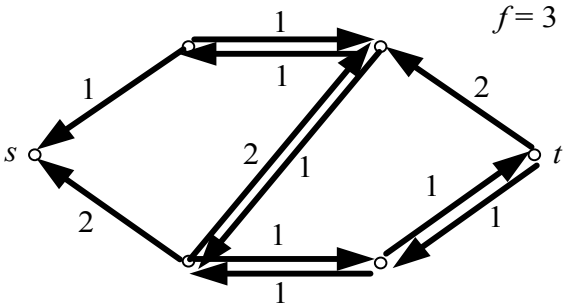
(a)



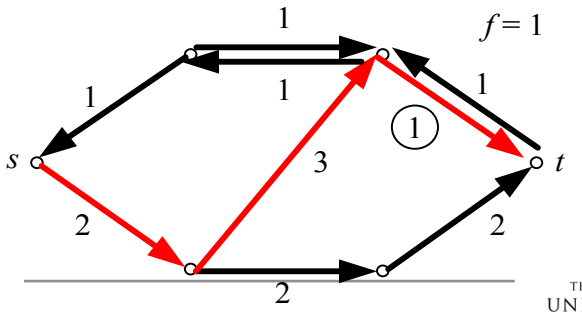
(d)



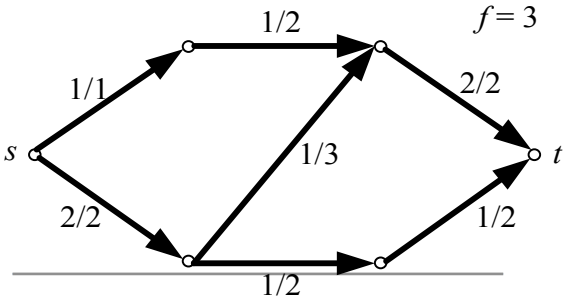
(b)



(e)



(c)



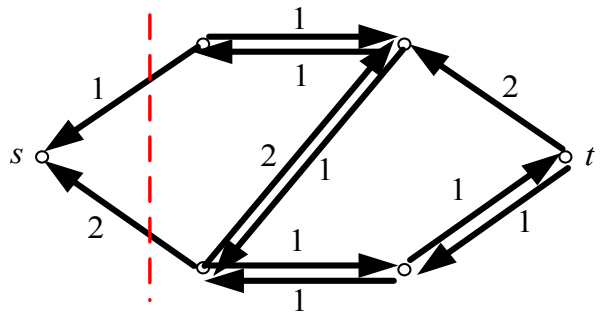
(f)

THE UNIVERSITY OF UTAH

SCI
www.sci.utah.edu

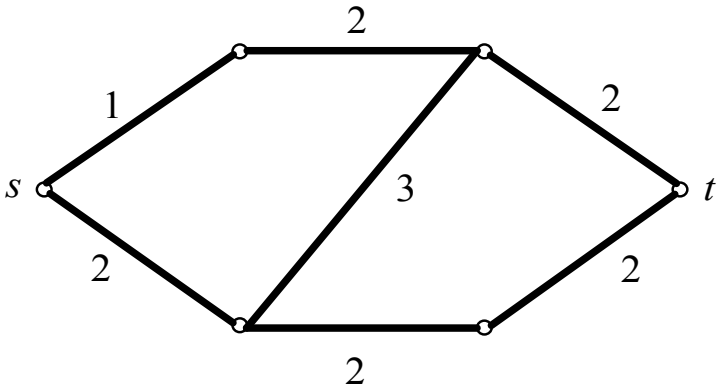
Finding the Min-Cut

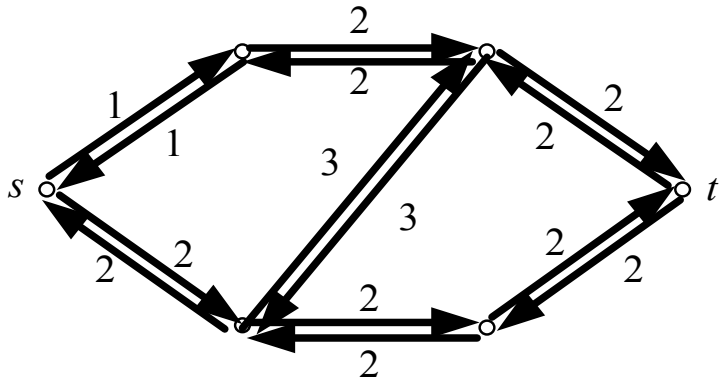
- After the max-flow is found, the minimum cut is determined by
- S = {All vertices reachable from s}
- T = G \ S



Special Case

- As in some applications only undirected graph is constructed, when we want to find the min-cut, we assign two edges with the same capacity to take the place of the original undirected edge





Algorithm Framework

The basic Ford-Fulkerson algorithm

for each edge $(u,v) \in E$

do

$$f(u,v) \leftarrow 0$$

$$f(v,u) \leftarrow 0$$

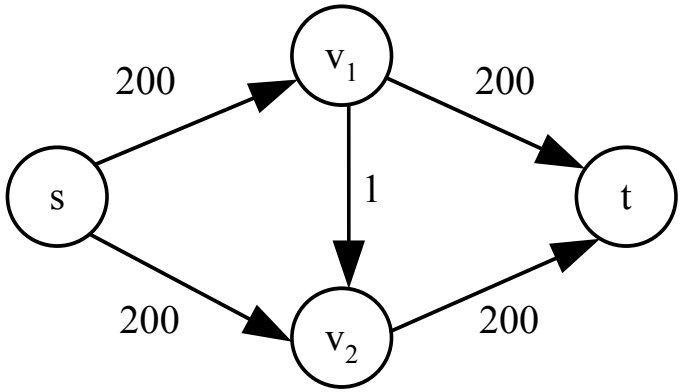
while there exists a path *P* from *s* to *t*
in the residual network G_f
do $c_f(P) \leftarrow \min\{c_f(u, v): (u, v) \text{ is on } P\}$
for each edge (u, v) in *P*
do

$$f(u,v) \leftarrow f(u,v) + c_f(P)$$

$$f(v,u) \leftarrow -f(u,v)$$

Ford-Fulkerson Algorithm Analysis

- Running time of the algorithm depends on how the augmenting path is determined
  - If the searching for augmenting path is realized by a breadth-first search, the algorithm runs in polynomial time of
    - $O(E|f_{max}|)$
- In extreme cases efficiency can be reduced drastically.
  - Below, applying Ford-Fulkerson algorithm needs 400 iterations to get the max flow of 400



Push-Relabel Algorithm

- This algorithm does not maintain a flow conservation rule, there is $\forall u \in V \setminus \{s\}, \exists f(V,u) \geq 0$
- We call the total net flow at a vertex u the excess flow into u, given by $e(u) = f(V,u)$. We say that a vertex $u \in V \setminus \{s,t\}$ is overflowing if $e(u) = f(V,u) > 0$
- Height Function:
  - Let $G = (V,E)$ be a flow network with source s and sink t, let f be a preflow (the flow satisfying the skew symmetry, capacity constraint, and the condition above) in G. A function $h : V \rightarrow \mathbf{N}$ is a height function if h (s) = | V |, h(t) = 0, and for every residual edge $(u,v) \in E_f$

Push-Relabel Algorithm Stages

Initialize-Preflow (*G*, *s*)

for each vertex $u \in V$

do

$h[u] \leftarrow 0$

$e[u] \leftarrow 0$

$h(s) \leftarrow |V|$

for each edge $(u,v) \in E$

do

$f[v,u] \leftarrow 0$

$f[u,v] \leftarrow 0$

for each vertex $u \in Adj(s)$

do

$f[s,u] \leftarrow c(s,u)$

$f(u,s) \leftarrow -c(s,u)$

$e[u] \leftarrow c(s,u)$

$e[s] \leftarrow e[s] - c(s,u)$

Push-Relabel Algorithm Stages

Push (*u*, *v*)
Applied when: *u* is overflowing, $c_f(u,v) > 0$, and $h(u) = h(v) + 1$

Action:
Push $d_f(u,v) = \min(e[u], c_f(u,v))$ units of flow from *u* to *v*:

$$d_f(u,v) \leftarrow \min(e[u], c_f(u,v))$$

$$f[u,v] \leftarrow f[u,v] + d_f(u,v)$$

$$f[v,u] \leftarrow -f[u,v]$$

$$e[u] \leftarrow e[u] - d_f(u,v)$$

$$e[v] \leftarrow e[v] + d_f(u,v)$$

Push-Relabel Algorithm Stages

Relable (*u*)
Applied when: *u* is overflowing and for all $v \in V$ such that

$(u,v) \in E_f$, we have $h(u) \leq h(v)$

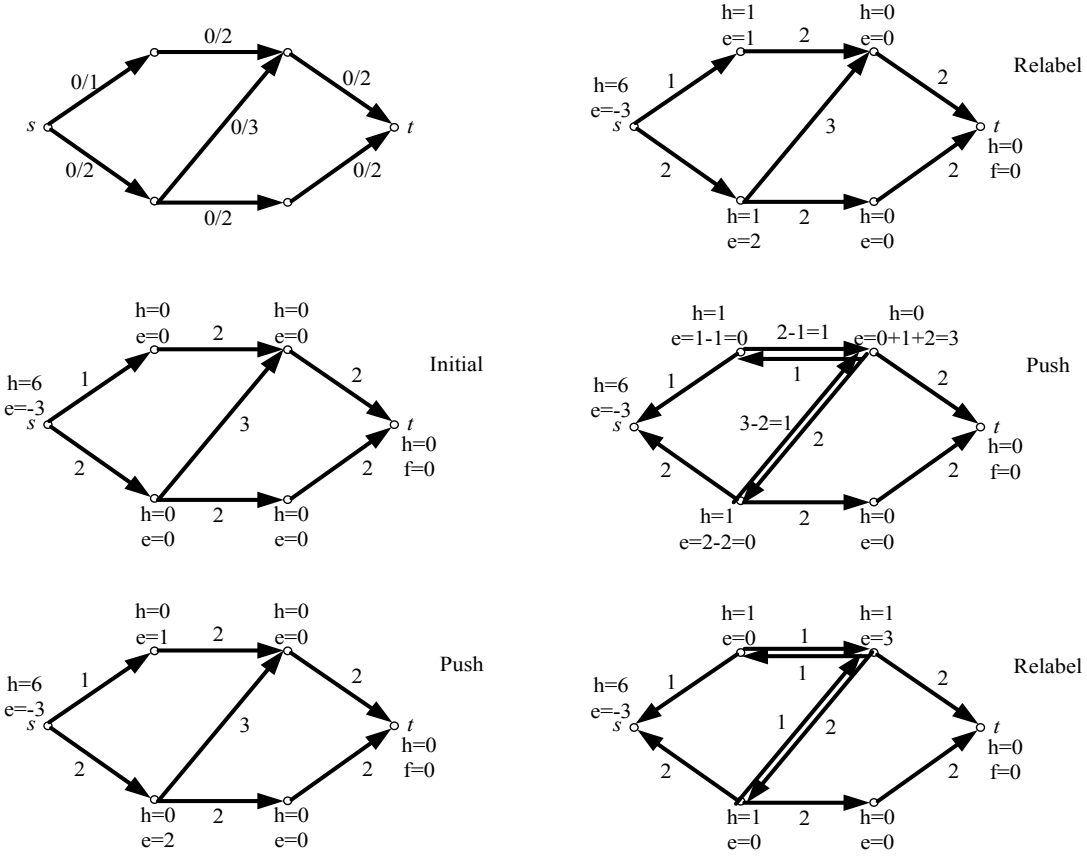
Action: Increase the height of u:

$$h(u) \leftarrow 1 + \min(h(v) : (u,v) \in E_f)$$

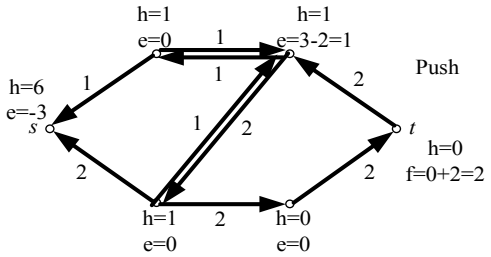
Push-Relabel Algorithm Framework

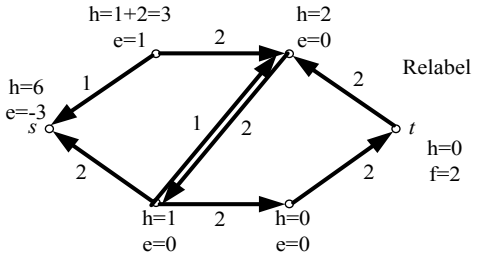
- Generic-Push-Relabel (G)
  - Initialize-Preflow (G, s)
    - while there exists an applicable push or relabel operation do
      - select an applicable push or relabel operation and perform it

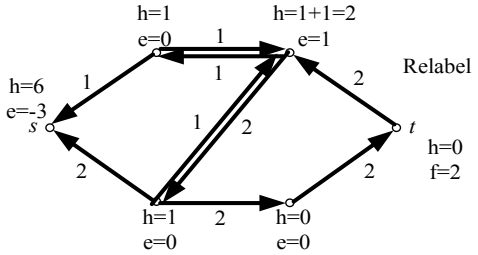
Push-Relabel Algorithm Example

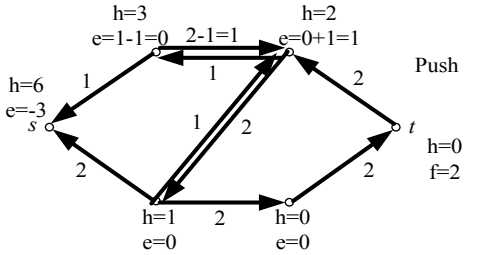


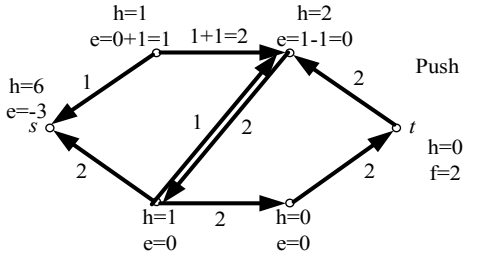
Push-Relabel Algorithm Example

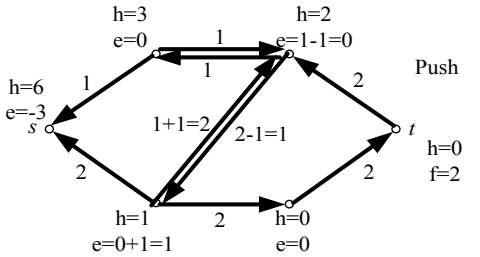




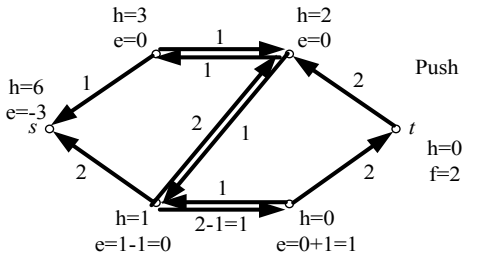


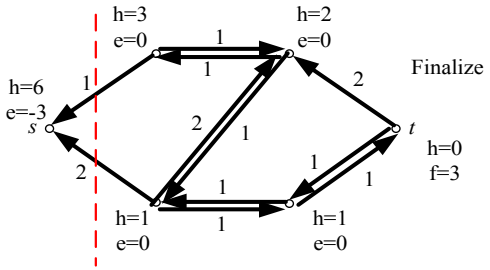


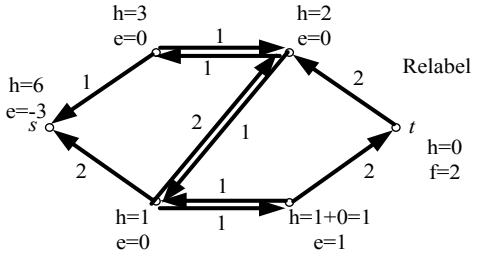


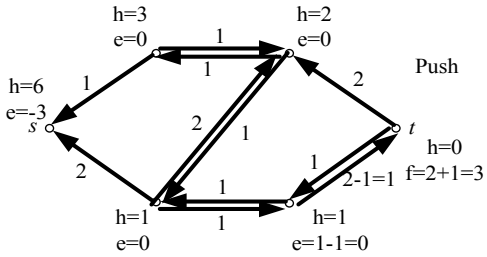


Push-Relabel Algorithm Example









Push-Relabel Algorithm Analysis

- The algorithm complexity is bounded to O(V2E)
- Improved algorithms using highest-level selection rule and FIFO-rule (queue based selection rule) can reduce the complexity to

$$O(V^2\sqrt{E})$$

- Optimizing the program by using bucket data structure based on any of the improved algorithms can yield O(VE)

Solving Image Segmentation Problem

- Data element set P representing the image pixels/ voxels
- Neighborhood system as a set N representing all pairs {p,q} of neighboring elements in P (ordered or unordered)
- A be a vector specifying the assignment of pixel p in P, each Ap can be either in the background or the object

$$A = (A_1, A_2, \ldots, A_p, \ldots, A_{|P|})$$

- A defines a segmentation of P

Cost of Segmentation

$$E(A) = \lambda \cdot R(A) + B(A)$$

$$R(A) = \sum_{p \in P} R_p(A_p)$$

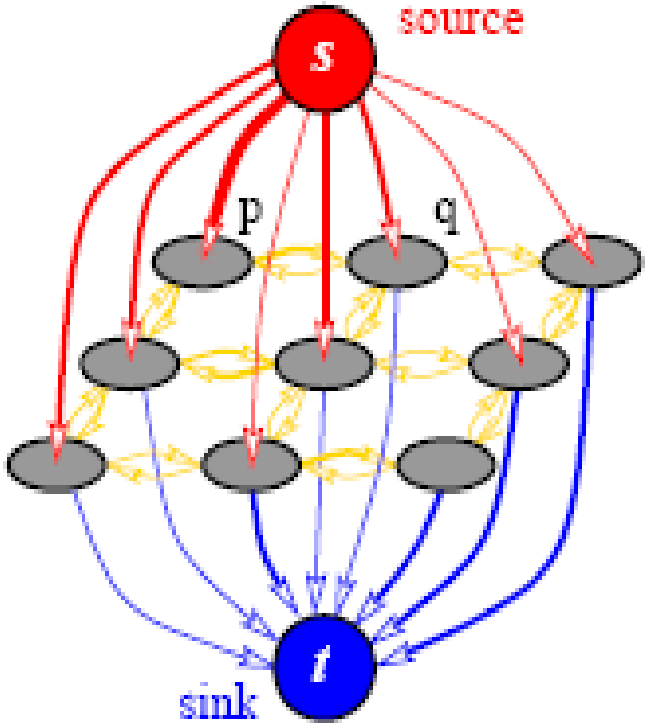
$$B(A) = \sum_{p \in P} \sum_{\{p,q\} \in N} B_{\{p,q\}} \cdot \delta(A_p, A_q)$$

$$\delta(A_p, A_q) = \begin{cases} 1 & A_p \neq A_q \\ 0 & A_p = A_q \end{cases}$$

- R(A) defines the penalties for assigning Ap to object or background, which are Rp("obj") and Rp("bkg")
- B(A) describes the boundary properties of the segmentation, B{p,q}is large when p and q are similar, it is close to 0 when p and q are very different

Graph Construction

Edge	Weight	Condition
n-link $\{p, q\}$	$B_{\{p, q\}}$	$\{p, q\} \in N$
t-link $\{p, s\}$	$\lambda \cdot R_p$ (“bkg”)	$p \in P, p \underline{\in} O \cup B$
	K	$p \in O$
	0	$p \in B$
t-link $\{p, t\}$	$\lambda \cdot R_p$ (“obj”)	$p \in P, p \underline{\in} O \cup B$
	0	$p \in O$
	K	$p \in B$



n-link Construction

$$B_{\{p,q\}} \propto \exp\left(-\frac{(I_p - I_q)^2}{2\sigma^2}\right) \cdot \frac{1}{dist(p,q)}$$

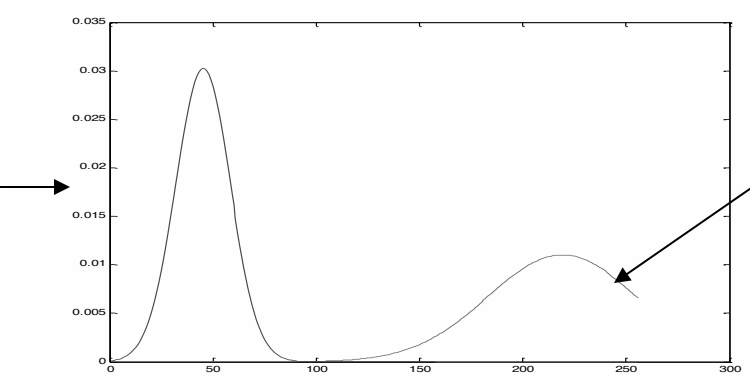
- Ip and Iq are the intensities of pixel p and q
- σ sets the penalty of discontinuities between pixels of similar intensities, when | Ip– Iq| < σ the penalty is large, when | Ip– Iq| > σ the penalty is small
- dist(p, q) penalizes the distance between p and q, and generally when they are in the neighborhood system this term is one

t-link Construction

- Rp("obj") = –ln Pr(Ip | O)
- Rp("bkg") = –ln Pr(Ip | B)

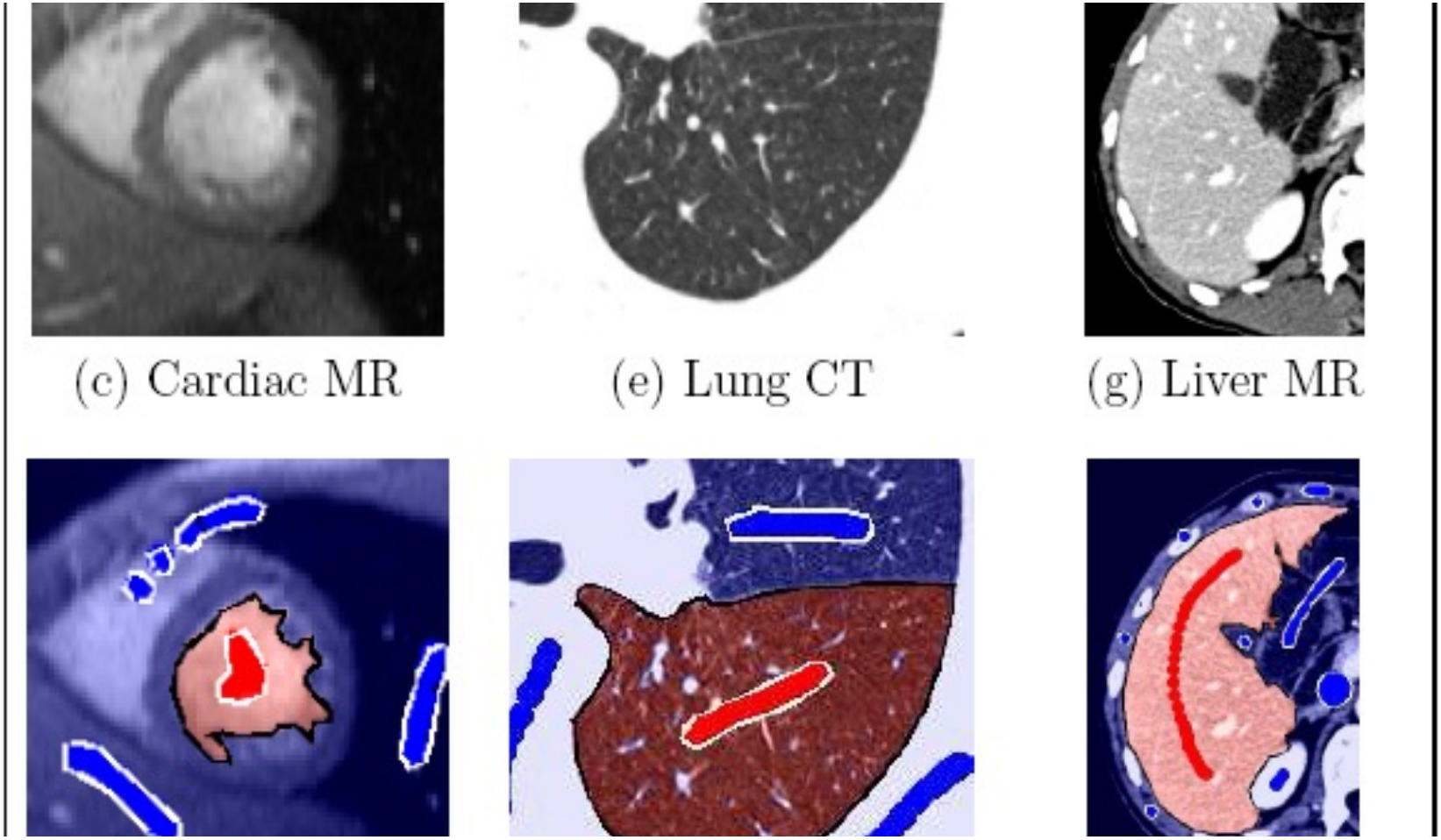
$$K = 1 + \max_{p \in P} \left(\sum_{q:\{p,q\} \in N} B_{\{p,q\}} \right)$$

Estimated Background Intensity Distribution Based on Background Seeds



Estimated Object Intensity Distribution Based on Background Seeds

Experimental Data



(c) Cardiac MR (e) Lung CT (g) Liver MR

Reference

- Yuri. Boykov and Marie-Pierre Jolly, "Interactive Graph Cuts for Optimal Boundary & Regiion Segmentation of Objects in N-D Images", In Proceeding of "International Conference on Computer Vision", Volume I, 105-112, July 2001
- Yuri. Boykov and Vladimir Kolmogorov, "An Experiment Comparison of Min-Cut / Max-Flow Algorithms for Energy Minimization in Vision", IEEE Transactions on PAMI, 26 (9): 1124-1137, September 2004
- Yuri. Boykov and Vladimir Kolmogorov, "Computing Geodesic and Minimal Surfaces via Graph Cuts", In Proceeding of "International Conference on Computer Vision", Volume II, 26-33, October 2003
- Vladimir Kolmogorov and Ramin Zabih, "What Energy Functions can be Minimized via Graph Cuts?", IEEE Transactions on PAMI, 26 (2): 147-159, February 2004
- Y. Boykov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts," IEEE Transactions on PAMI, 23 (11): 1222-1239, November 2004
- Sudipta Sinha, "Graph Cut Algorithms in Vision, Graphics and Machine learning, An Integrated Paper", UNC Chapel Hill, November 2004
- Yuri Boykov and Olga Veksler, "Graph Cuts in Vision and Graphics: Theories and Applications", Chapter 5 of The Handbook of Mathematical Models in Computer Vision, 79-96, Springer, 2005
- Thomas Cormen, Charles Leiserson, Ronald Rivest and Clifford Stein, "Maximum Flow", Chapter 26 of Introduction to algorithms, second edition, 643-698, McGraw-Hill, 2005
- L. R. Floulds, "An Introduction to Transportation Networks", Section 12.5.3 of Graph Theory Applications, 246-256, Springer, 1992
- L. Ford and D. Fulkerson, Flows in Networks, Princeton University Press, 1962.
- Andrew V. Goldberg and Robert E. Tarjan, "A New Approach to the Maximum-Flow Problem", Journal of the Association for Computing Machinery, 35(4):921–940, October 1988
- E. A. Dinic, "Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation", Soviet Math. Dokl., 11:1277–1280, 1970