Image Segmentation

Slides by Carl Kingsford

Apr. 7, 2014

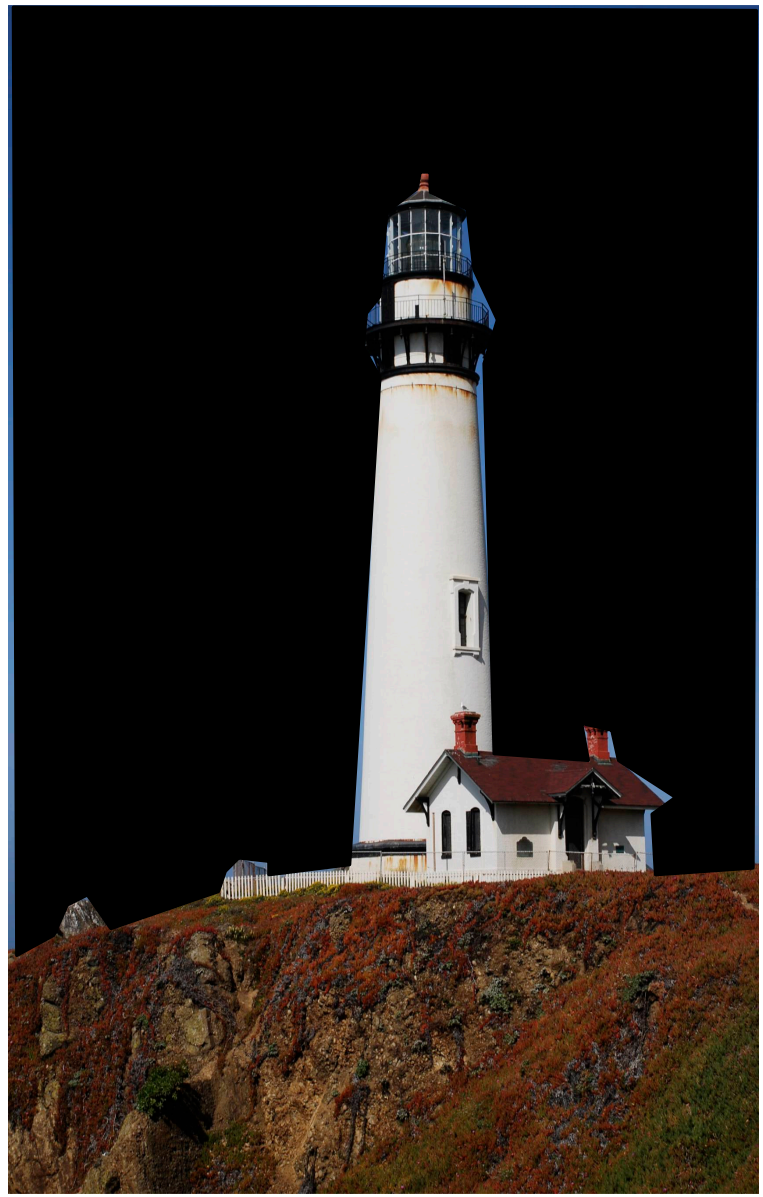
Based on AD 7.10

Image Segmentation

- Given an image, what is foreground and what is background?
- E.g. Hockey puck against the ice, football against the field, missiles against the sky.
- If (x, y) is a foreground pixel, then nearby pixels are also likely foreground.

Image Segmentation

- Given an image, what is foreground and what is background?
- E.g. Hockey puck against the ice, football against the field, missiles against the sky.
- If (x, y) is a foreground pixel, then nearby pixels are also likely foreground.



Domain Knowledge

Different applications will have different rules for identifying foreground and background pixels, e.g.:

> If a pixel is brown, it is more likely to be the football.

Abstract those rules away, and suppose we have 2 nonnegative numbers for each pixel i:

1. a_i = likelihood that pixel i is in foreground.
2. b_i = likelihood that pixel i is in background.

How these values are generated depends on the application.

Clumping

All else equal, if $a_i > b_i$, then we should make i a foreground pixel.

But: we noted that foreground pixels tend to be near one another, and background pixels tend to be near one another.

To represent neighboring pixels, we encode an image by an undirected graph $G = (V, E)$.

- V contains a vertex for each pixel
- E contains an edge between pixels i and j if i and j neighbor each other.

Image Graph Example

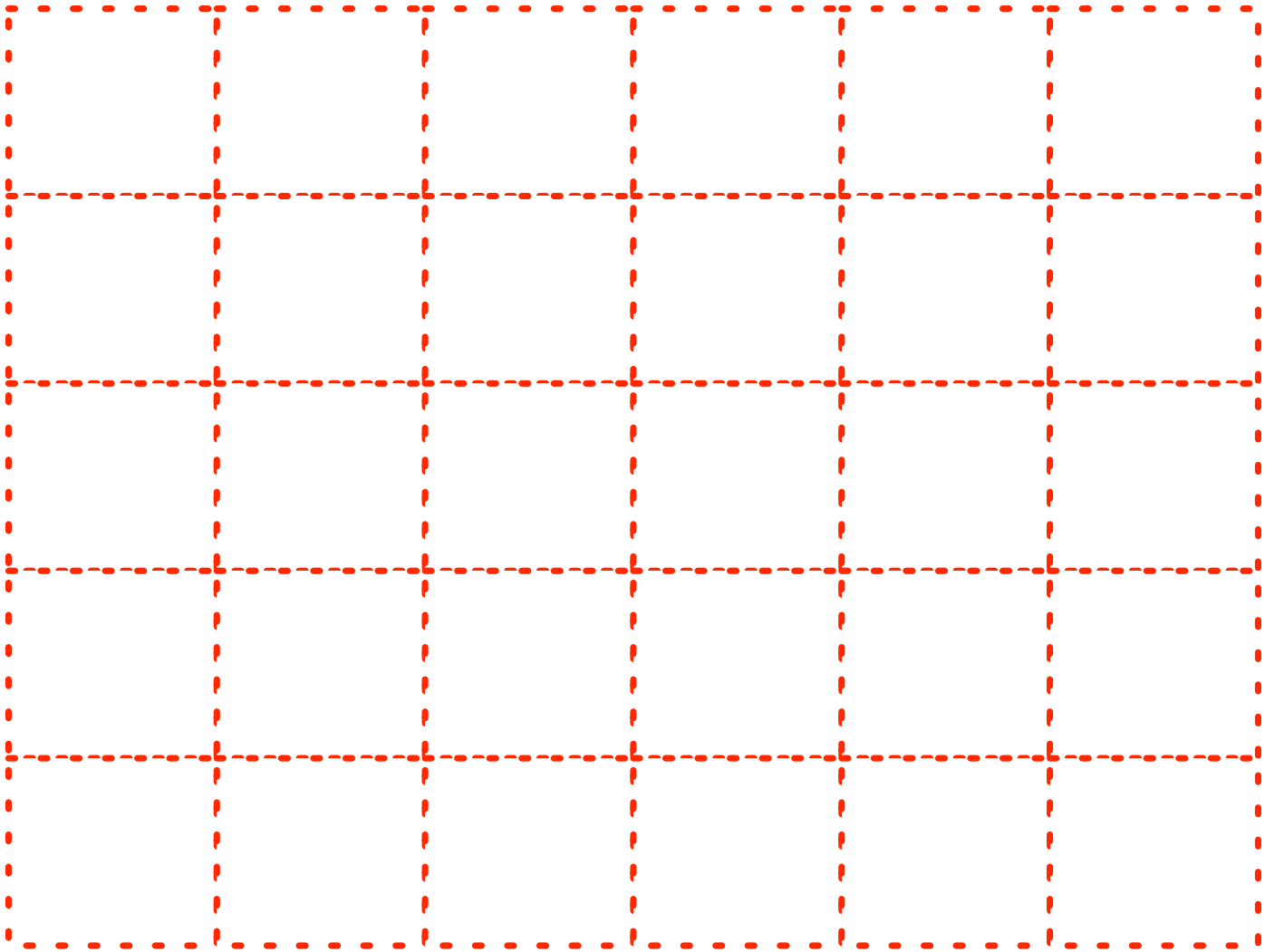


Image Graph Example

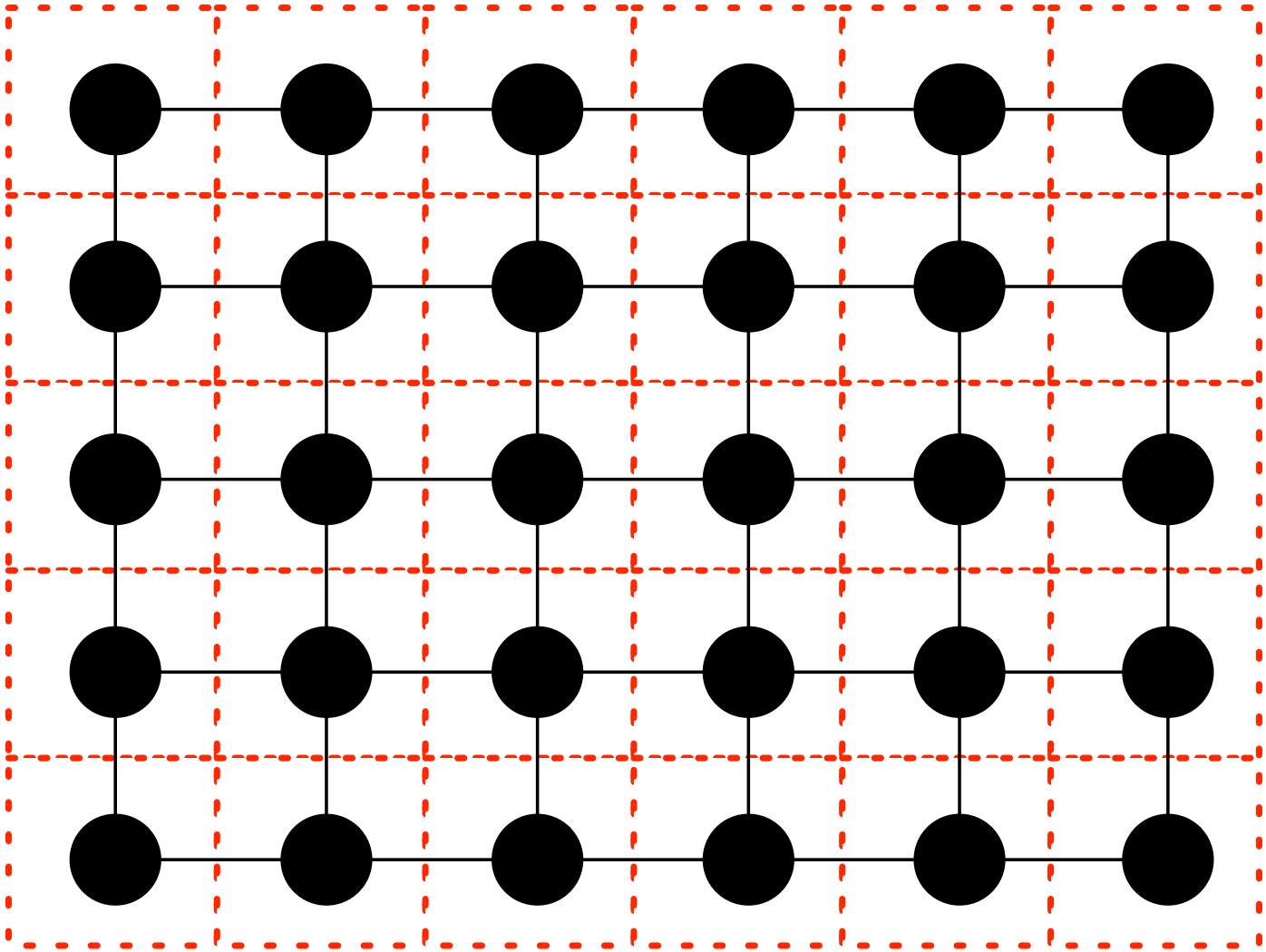


Image Segmentation Problem

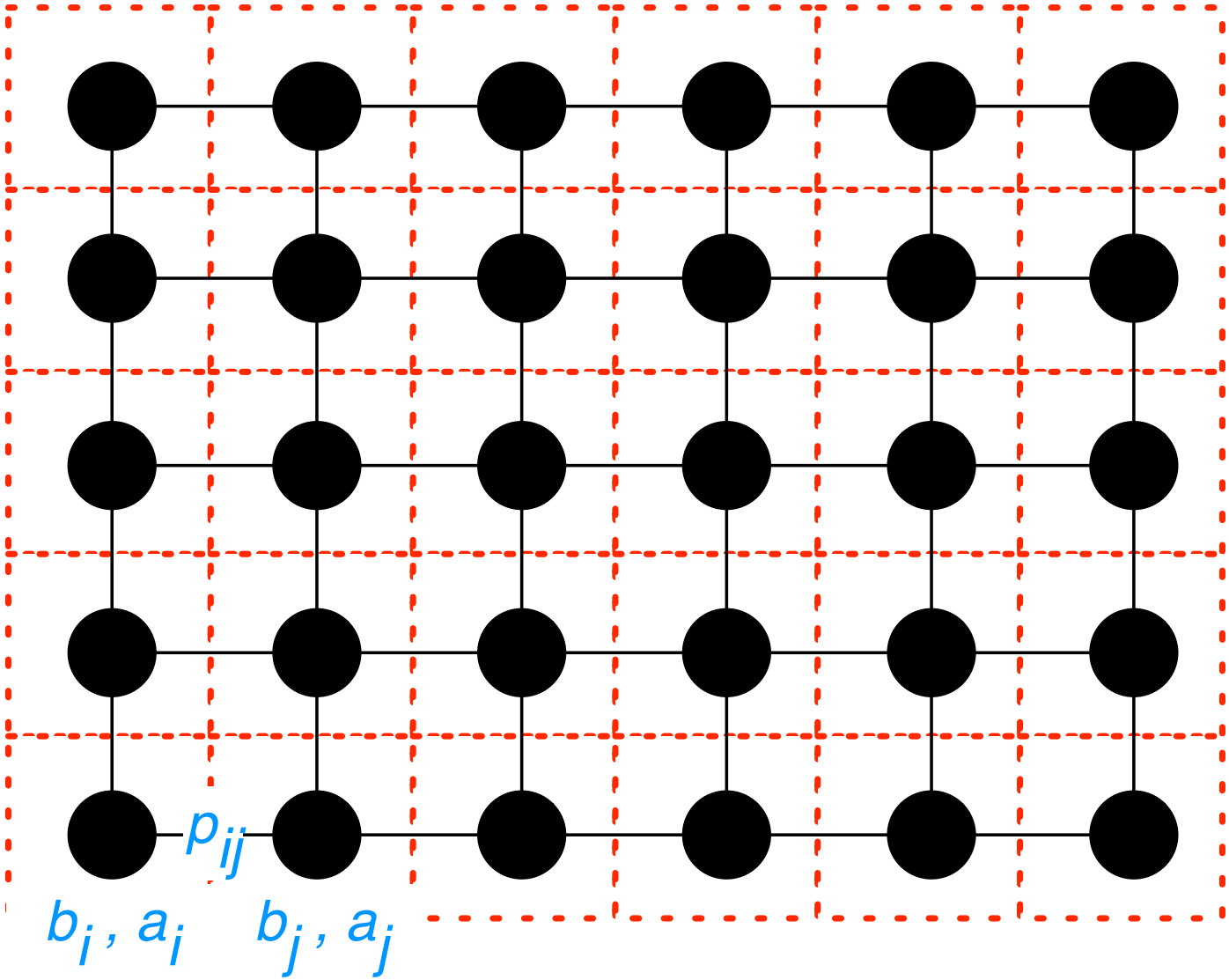
We add new parameters to model separating neighboring pixels. For every neighboring pair of pixels $\{i, j\}$, we have a parameter:

3. p_{ij} = the penalty for putting one of i, j in foreground and the other in the background.

Image Segmentation Problem Partition the set of pixels into two sets A and B to maximize:

$$q(A, B) = \sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{\substack{(i,j) \in E \\ i,j \text{ sep}}} p_{ij} .$$

Image Graph Example



Min Cut?

Image Segmentation: Partition the vertices of the image graph into 2 sets A, B to maximize $q(A, B)$.

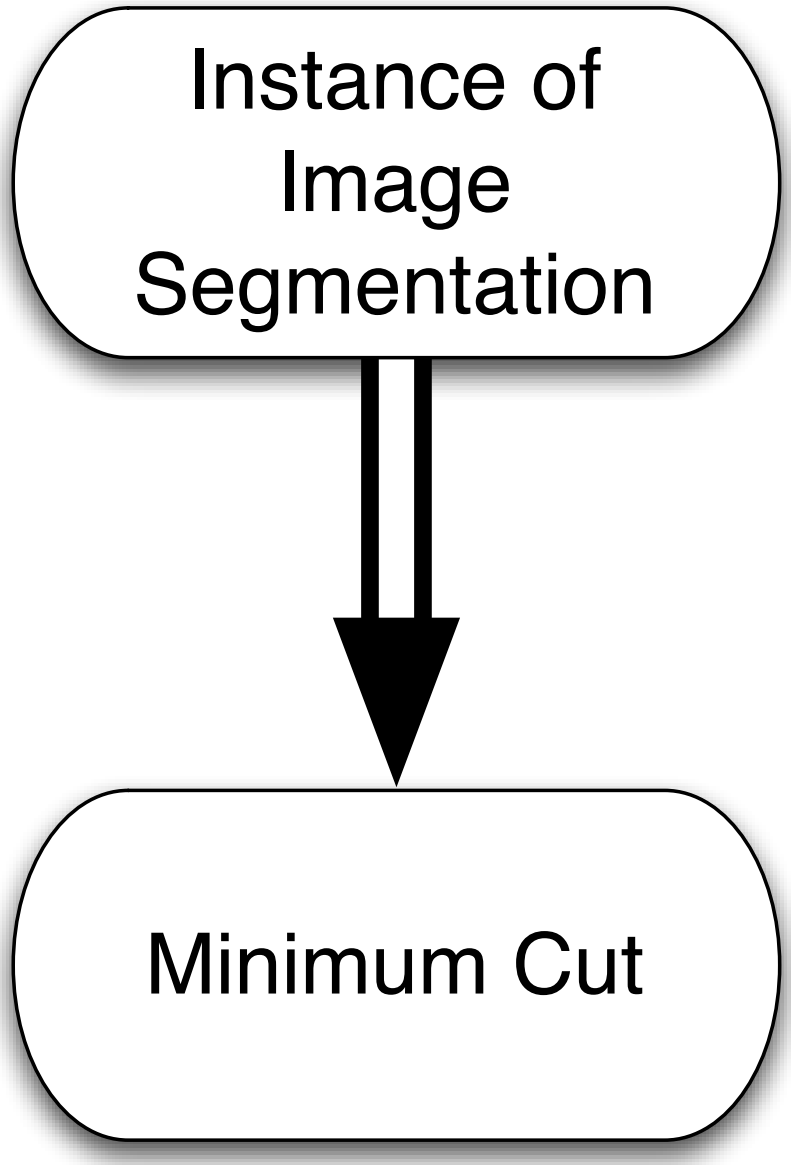
Minimum Cut: Partition the vertices of the a directed graph into 2 sets A, B, with $s \in A$, $t \in B$, to minimize weight of edges crossing from A to B.

Seem similar, but with some differences:

- Maximization vs. minimization
- Image segmentation has no source or sink
- Image segmentation has a more complicated objective function $q(A, B)$ with weights on the nodes
- Undirected vs. directed

Reduction!

- Given an instance of Image Segmentation,
- Create an instance of Minimum Capacity Cut.
- Where the minimum cut can trivially be used to find the solution to the image segmentation.



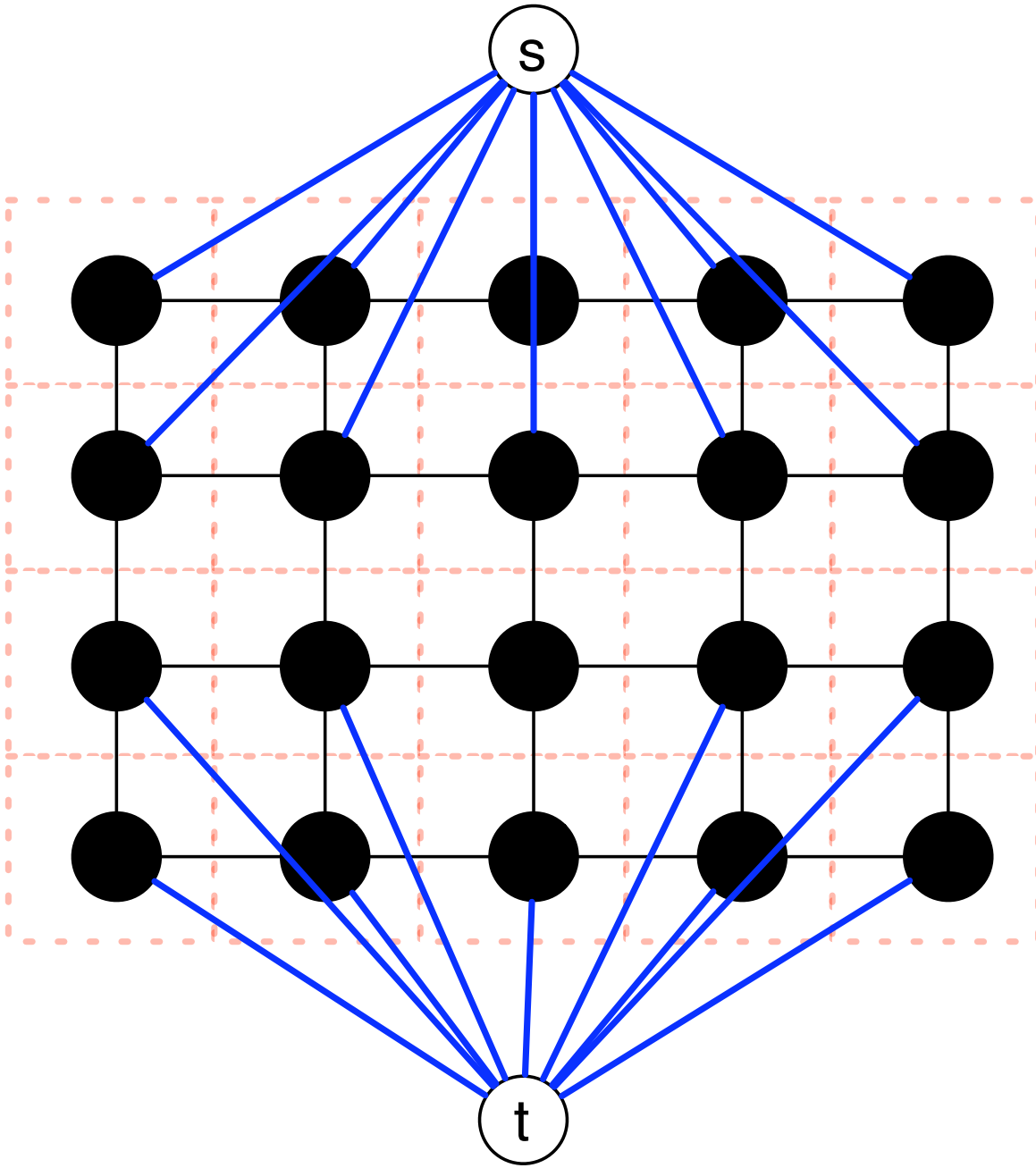
Missing Source and Sink?

We add:

- a source s with an edge (s, u) for every vertex u.
- a sink t with an edge (u, t) for every vertex u.

We will see: s will represent the foreground, and t will represent the background.

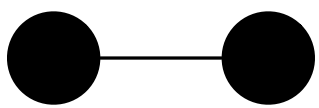
Source and Sink Example



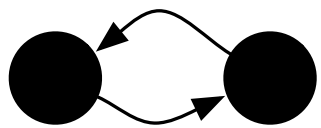
Directed Edges

We convert the currently undirected graph into a directed graph:

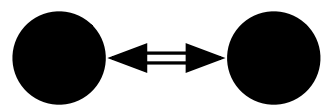
- Edges adjacent to s are directed so they leave s.
- Edges adjacent to t are directed so they enter t.
- All other edges are replaced by 2, anti-parallel edges:



A single undirected edge becomes...

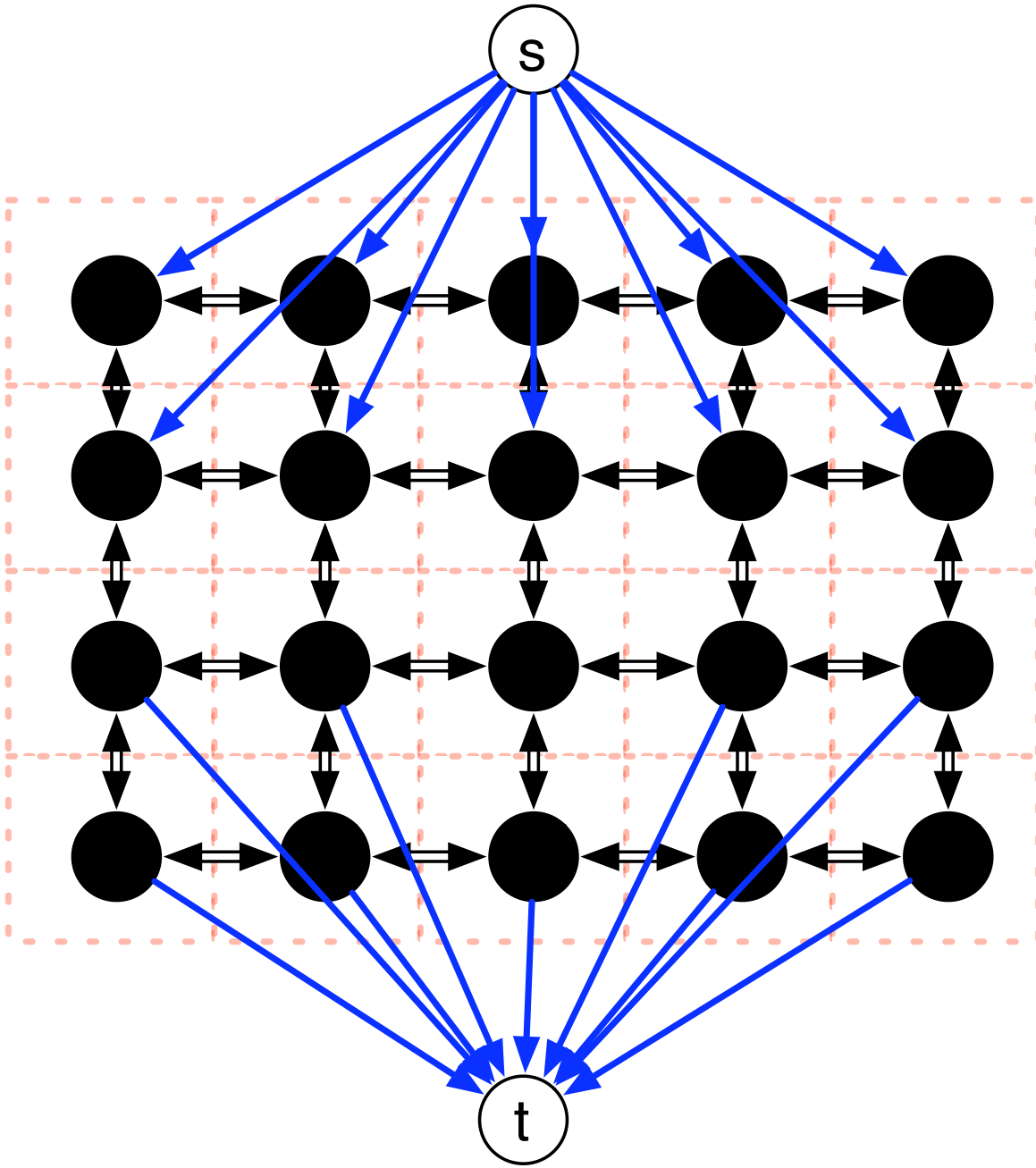


2 anti-parallel edges



Sometimes draw this way to save space

Directed Edges Example



What have we done?

Given an image graph $G = (V, E)$ where V represent the pixels and E connect neighboring pixels, we produced a new graph $G' = (V', E')$ by:

1. $V' = V \cup \{s, t\} \leftarrow$ added source and sink.

2. E' edges from s to every pixel and from every pixel to t plus edges (i, j) and (j, i) between each pair of pixels.

Last issue: how do we handle the parameters a_i, b_i, p_{ij} and minimization vs. maximization?

Maximization vs. minimization

Let $Q = \sum_i (a_i + b_i)$.

Note that:

$$\sum_{i \in A} a_i \; + \; \sum_{j \in B} b_j = Q - \sum_{i \in A} b_i \; - \; \sum_{j \in B} a_j$$

Our old objective function was to maximize:

$$q(A, B) = \sum_{i \in A} a_i \; + \; \sum_{j \in B} b_j \; - \; \sum_{\substack{(i,j) \in E \\ i, j \text{ sep}}} p_{ij}$$

By above, this equals:

$$q(A, B) = Q - \sum_{i \in A} b_i \; - \; \sum_{j \in B} a_j \; - \; \sum_{\substack{(i,j) \in E \\ i, j \text{ sep}}} p_{ij}$$

Maximization vs. Minimization

We want to maximize:

$$q(A,B) = Q - \sum_{i \in A} b_i \; - \sum_{j \in B} a_j \; - \sum_{\substack{(i,j) \in E \\ i,j \text{ sep}}} p_{ij}$$

This is the same as minimizing:

$$q'(A,B) = \sum_{i \in A} b_i \; + \sum_{j \in B} a_j \; + \sum_{\substack{(i,j) \in E \\ i,j \text{ sep}}} p_{ij}$$

Parameters

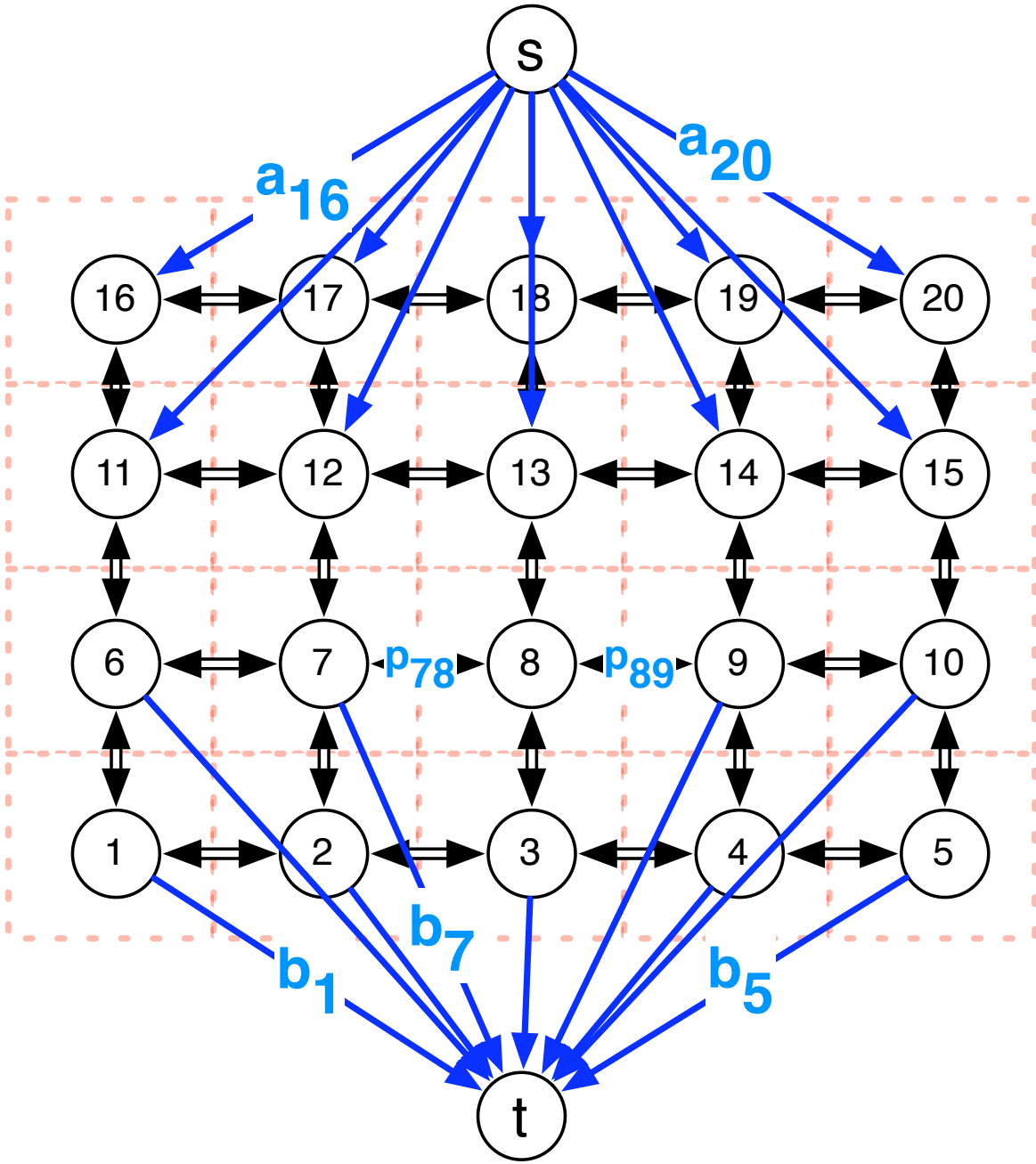
We use the parameters a_i, b_i and p_{ij} as weights on the various edges:

- Edges between two pixels i and j get weight p_{ij}.
- Edge (s, i) gets weight a_i.
- Edge (j, t) get weight b_j.

The intuition here is that the capacity of an $s - t$ cut (A, B) will equal the the quantity we are trying to minimize. (We'll see why.)

Therefore, if we find the min cut, we will find the best partition into foreground and background.

Weights Example



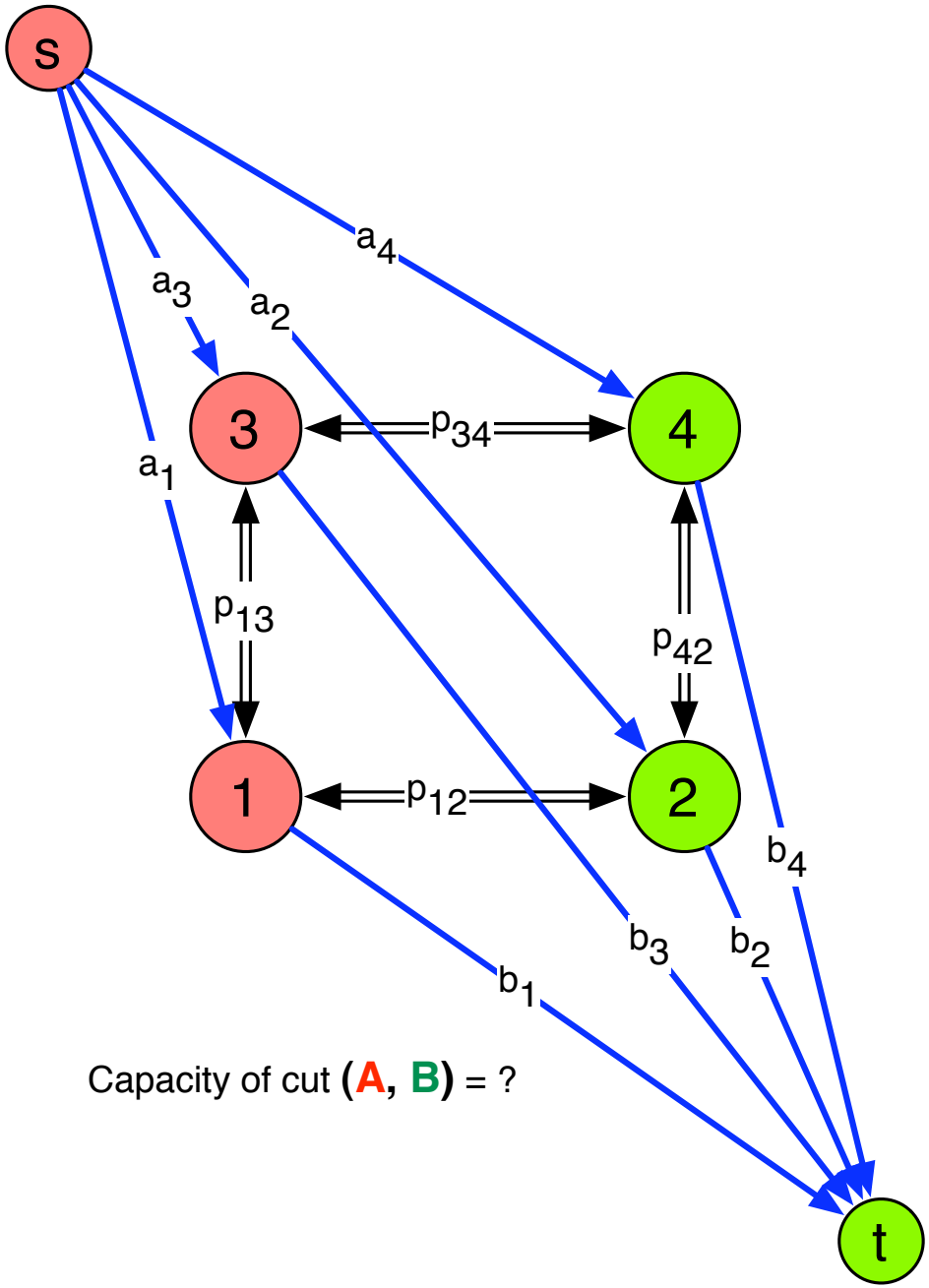
Capacity of a cut = quality of partition

We've designed the graph so that the capacity of an s-t cut (A,B) equals the quality of the partition defined by taking:

- A to be the set of foreground pixels (plus s)
- B to be the set of background pixels (plus t)

Why is this?

This is why:



Capacity of cut **(A, B)** = ?

Cut Edges

$$\text{minimize } q'(A,B) = \sum_{i \in A} b_i + \sum_{j \in B} a_j + \sum_{\substack{(i,j) \in E \\ i,j \text{ sep}}} p_{ij}$$

The edges of any cut (A, B) of our graph can be divided into 3 groups:

- Edges (i, t), where $i \in A$: this edge contributes b_i to the capacity (and putting i into the foreground costs us b_i).
- Edges (s, j), where $j \in B$: this edge contributes a_j to the capacity (and putting j into the background costs us a_j).
- Edges (i, j), where $i \in A, j \in B$: contributes p_{ij} to the capacity (and separating i and j costs us p_{ij}).

Summary

We've shown that for a cut (A, B) in G':

$$\text{capacity}(A, B) = \sum_{i \in A} b_i + \sum_{j \in B} a_j + \sum_{\substack{(i,j) \in E \\ i,\, j \text{ sep}}} p_{ij} = q'(A, B)$$

Hence, finding the minimum capacity cut in our modified graph gives us the partition into foreground A and background B that maximizes $q(A, B)$.

We can find that best partition using a minimum-cut algorithm on the modified graph G' and deleting s and t from the cut it returns.