Airline Scheduling

Design and Analysis of Algorithms
Andrei Bulatov

The Problem

An airline carrier wants to serve certain set of flights

Example:

Boston (6 am) - Washington DC (7 am), San Francisco (2:15pm) - Seattle (3:15pm)

Philadelphia (7 am) - Pittsburg (8 am), Las Vegas (5 pm) - Seattle (6 pm)

Washington DC (8 am) – Los Angeles (11 am)

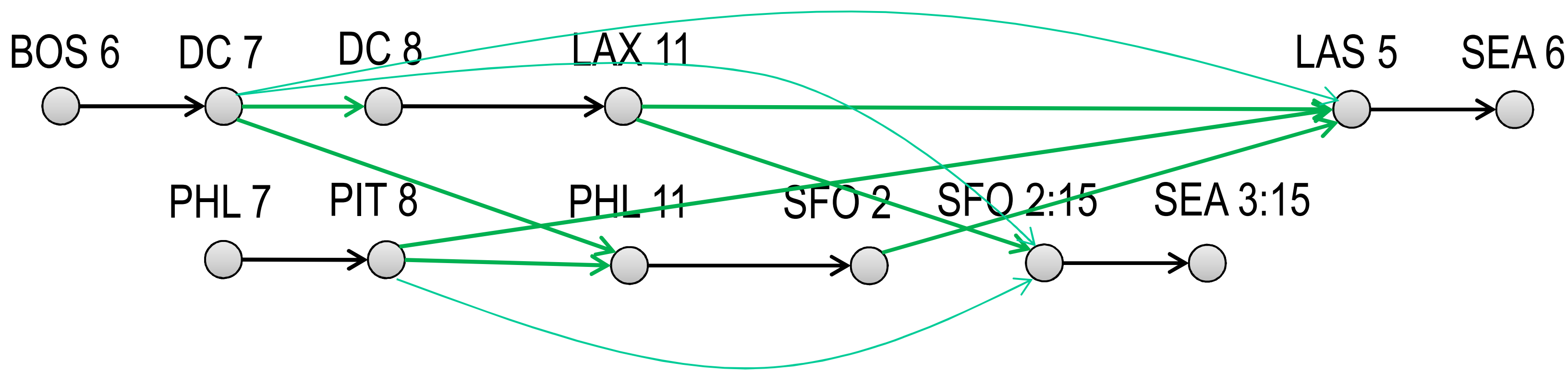
Philadelphia (11 am) - San Francisco (2 pm)

The same plane can be used for flight i and for flight j if

- the destination of i is the same as origin of j and there is enough time for maintenance (say, 1 hour)
- a flight can be added in between that gets the plane from the destination of i to the origin of j with adequate time in between

Formalism

Boston (6 am) - Washington DC (7 am),
Philadelphia (7 am) - Pittsburg (8 am),
Washington DC (8 am) – Los Angeles (11 am),
Philadelphia (11 am) - San Francisco (2 pm)
San Francisco (2:15pm) - Seattle (3:15pm)
Las Vegas (5 pm) - Seattle (6 pm)



Flight j is reachable from flight i if it is possible to use the same plane for flight i, and then later for flight j as well.

(or we can use a different set of rules, it does not matter)

The Problem

The Airline Scheduling Problem

Instance:

A set of flights to serve, and a set of pairs of reachable flights, the allowed number k of planes

Objective:

Is it possible to serve the required flights with k planes

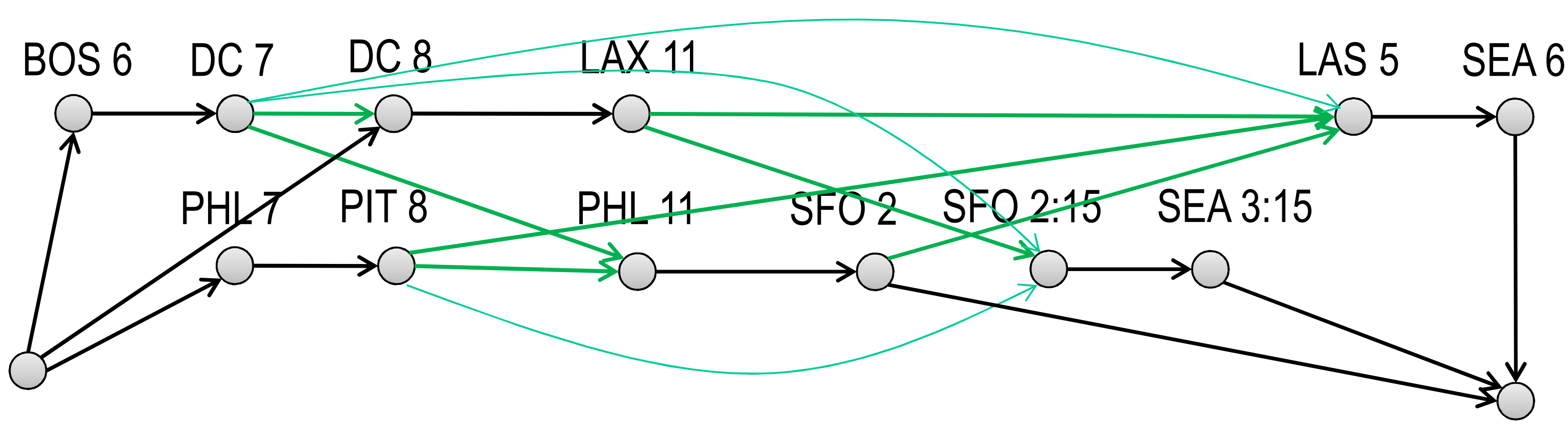
The Idea

Each airplane is represented by a unit of flow

The required flights (arcs) have lower bound 1, and capacity 1

If (u_i, v_i) and (u_j, v_j) are arcs representing required flights i and j, and j is reachable from i, then there is an arc connecting v_i to u_j; we assign this arc capacity 1

Extend the network by adding an external source and sink



Construction

- For each required flight i, the graph G has two nodes u_i and v_i
- G also has a distinct source s and a sink t
- For each i, there is an arc (u_i, v_i) with a lower bound 1 and capacity 1
- For each i and j such that flight j is reachable from flight i, there is an arc (v_i, u_j) with a lower bound 0 and a capacity 1
- For each i there is an arc (s, u_i) with a lower bound 0 and a capacity 1
- For each i there is an arc (v_i, t) with a lower bound 0 and a capacity 1
- There is an arc (s,t) with lower bound 0 and capacity k
- The node s has demand -k, and node t has demand k

The Problem

Theorem

There is a way to perform all flights using at most k planes if and only if there is a feasible circulation in the network G.

Proof

DIY

Image Segmentation

Design and Analysis of Algorithms
Andrei Bulatov

Image Segmentation

The general problem is to separate objects on a digital image

We have pixels and have to decide, which object each pixel belongs to

The problem we solve:

decide whether a pixel belongs to the background or foreground

The decision, where a given pixel belongs to is made taking into account its neighbors

Usually, pixels are arranged in a grid

But our model will allow any other configuration

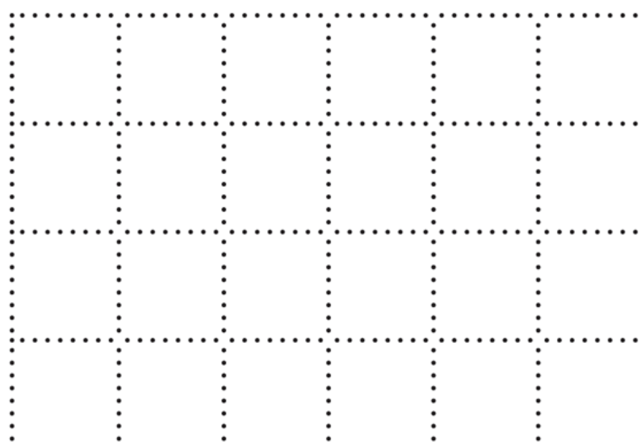


Image Segmentation: Framework

We construct an undirected graph G = (V,E) where V is the set of pixels, and E is the neighborhood relation

For every pixel i there are associated likelihood a_i that it belongs to the foreground, and likelihood b_i that it belongs to the background

The likelihoods can be any non-negative numbers

For a pixel i we tend to label it as a foreground pixel if $a_i > b_i$ and as a background pixel otherwise

However, the label also depends on labels of its neighbors

It is regulated by a separation penalty $p_{ij} \geq 0$ for one of i and j in the foreground, and the other in the background

The Problem

The Image Segmentation Problem

Instance:

A set of pixels with likelihoods and separation penalties

Objective:

Find an optimal labeling, that is, a partition of the set of pixels into sets A and B (foreground and background) so as to maximize

$$q(A,B) = \sum_{i \in A} a_i + \sum_{i \in B} b_i - \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}| = 1}} p_{ij}$$

Algorithm Ideas

Observe that the problem is similar to finding a minimal cut

Difficulties:

(1) Need to maximize, rather than minimize

(2) No source and sink

(3) Have to deal with values assigned to vertices

(4) The graph is undirected

Algorithm Ideas (cntd)

(1) Need to maximize, rather than minimize

Let $Q = \sum_{i} (a_i + b_i)$

Then $\sum_{i \in A} a_i + \sum_{i \in B} b_i = Q - \sum_{i \in A} b_i - \sum_{i \in B} a_i$

So $q(A,B) = Q - \sum_{i \in A} b_i - \sum_{i \in B} a_i - \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}| = 1}} p_{ij}$

Thus instead of maximizing q(A,B) we can minimize

$$q'(A,B) = \sum_{i \in A} b_i + \sum_{i \in B} a_i + \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}| = 1}} p_{ij}$$

Algorithm Ideas (cntd)

(2) No source and sink

Add an external source, s, and sink, t

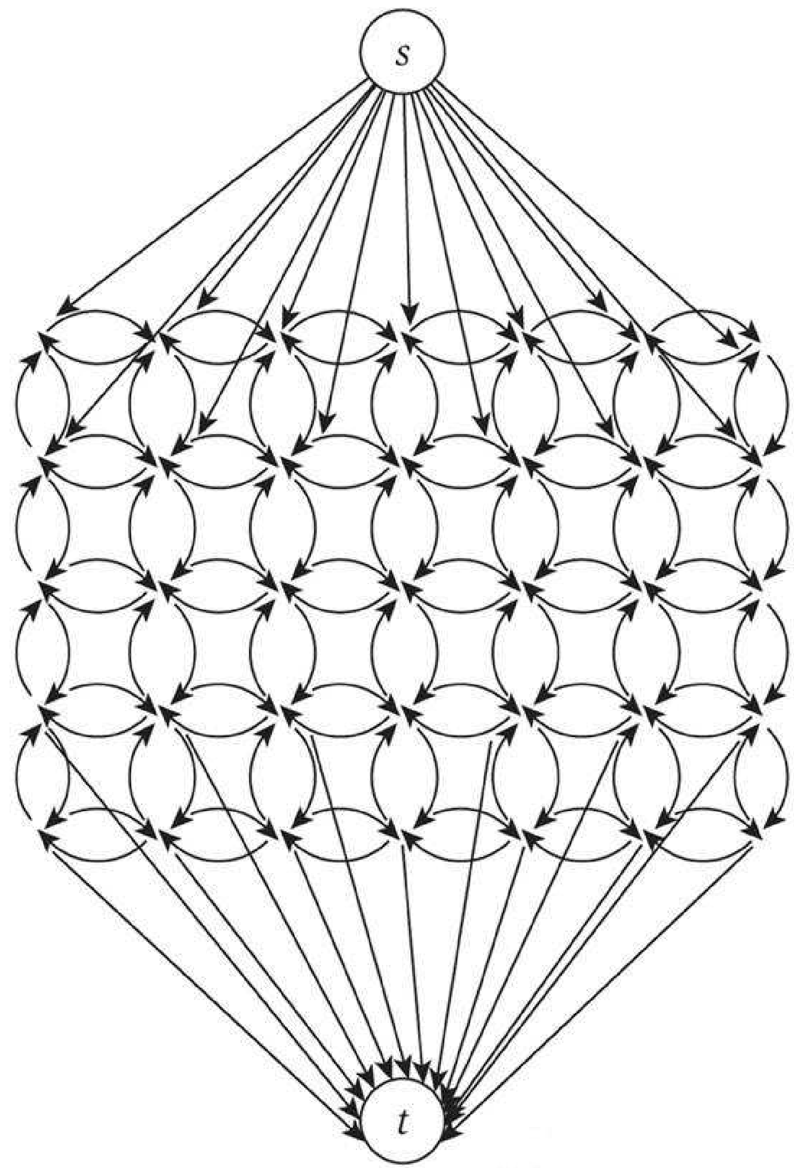
(3) Have to deal with values assigned to vertices

We use the external source and sink
Assign capacity a_i to edges (s,i),
and capacity b_i to edges (i,t)

Algorithm Ideas (cntd)

(4) The graph is undirected

Replace each undirected edge with two directed arcs and assign capacity p_{ij} to both

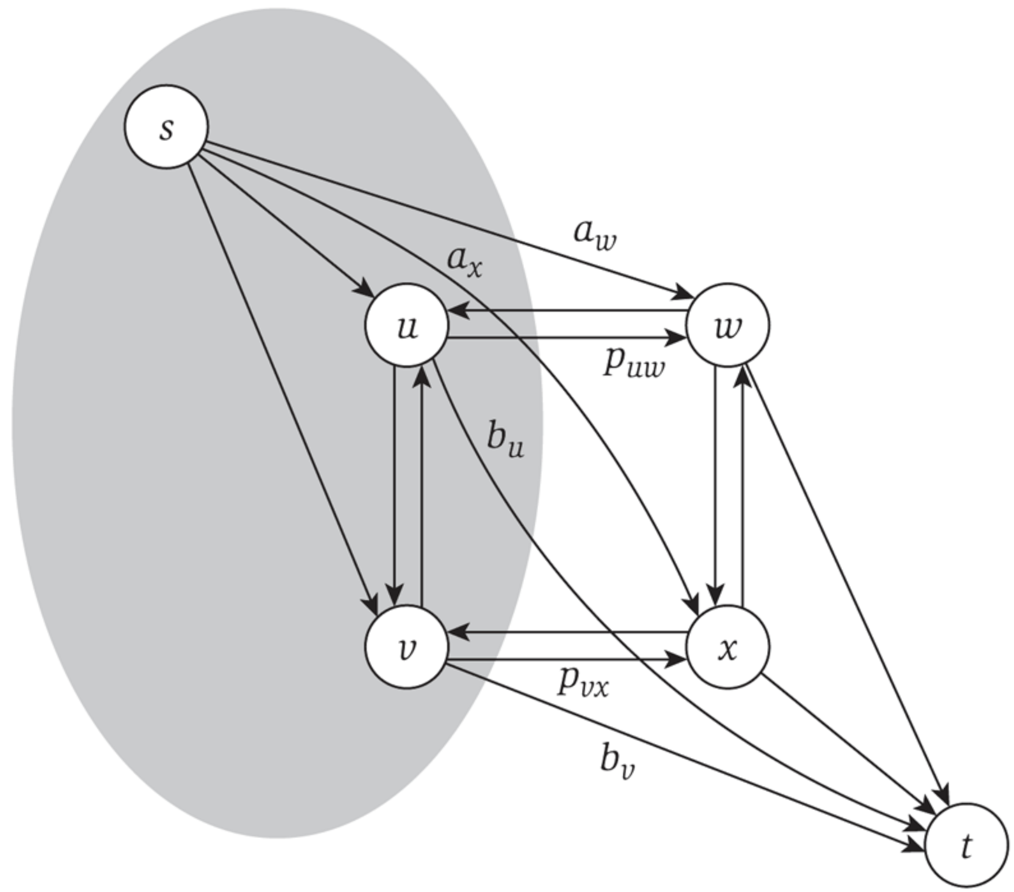


Cuts vs. Segmentation

A cut (A ∪ {s}, B ∪ {t}) corresponds to partition (A,B) of the original graph

The capacity c(A,B) is contributed by

- edges (s,j), where j ∈ B; this edge contributes a_i
- edges (i,t), where i ∈ A; this edge contributes b_i
- edges (i,j), where i ∈ A and j ∈ B; this edge contributes p_{ij}



The Result

Theorem

The solution to the Segmentation Problem can be obtained by a minimum-cut algorithm in the graph G' constructed above. For a minimum cut (A',B'), the partition (A,B) obtained by deleting s and t maximizes the segmentation value q(A,B)