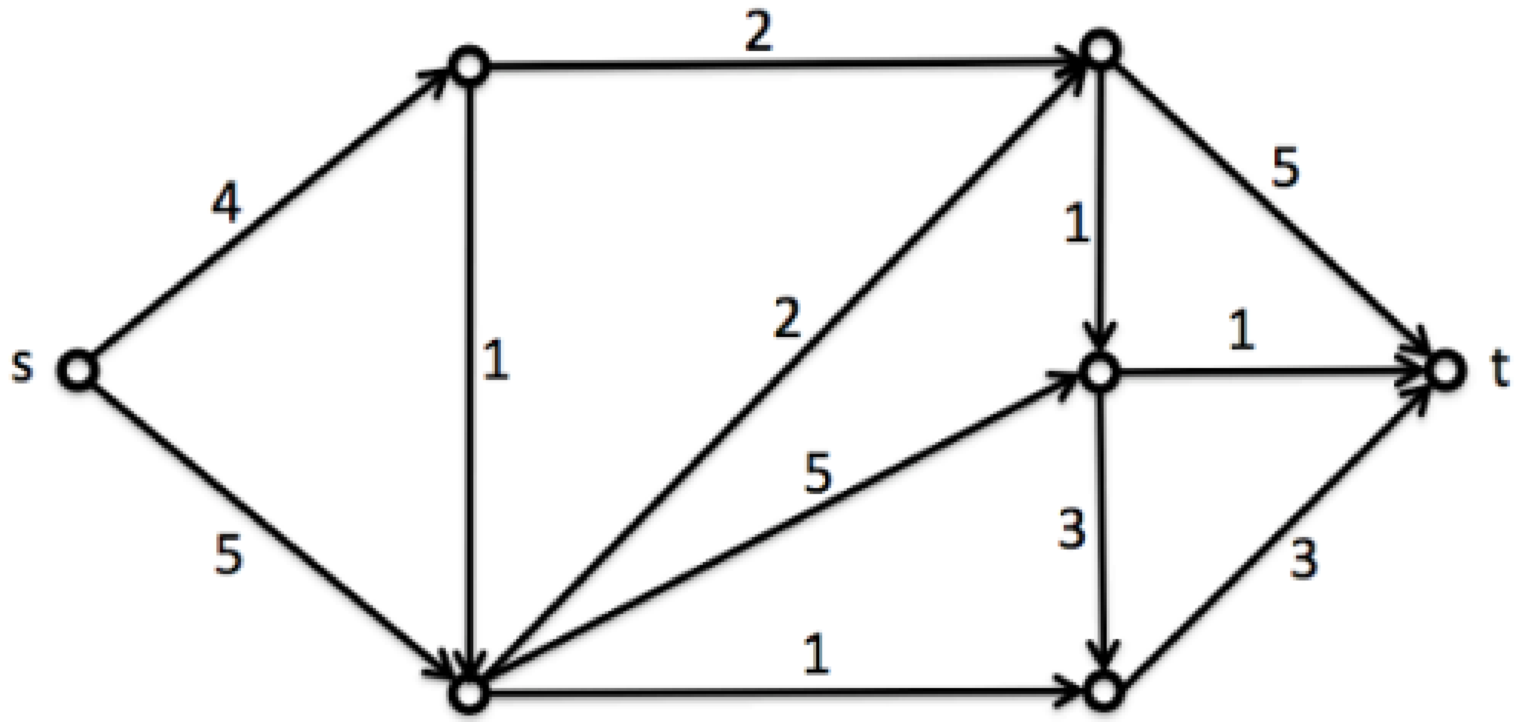
Maximum Flows and Minimum Cuts II

Lecture 6 COMP 610

January 24 2018

Recall: A Capacitated Network

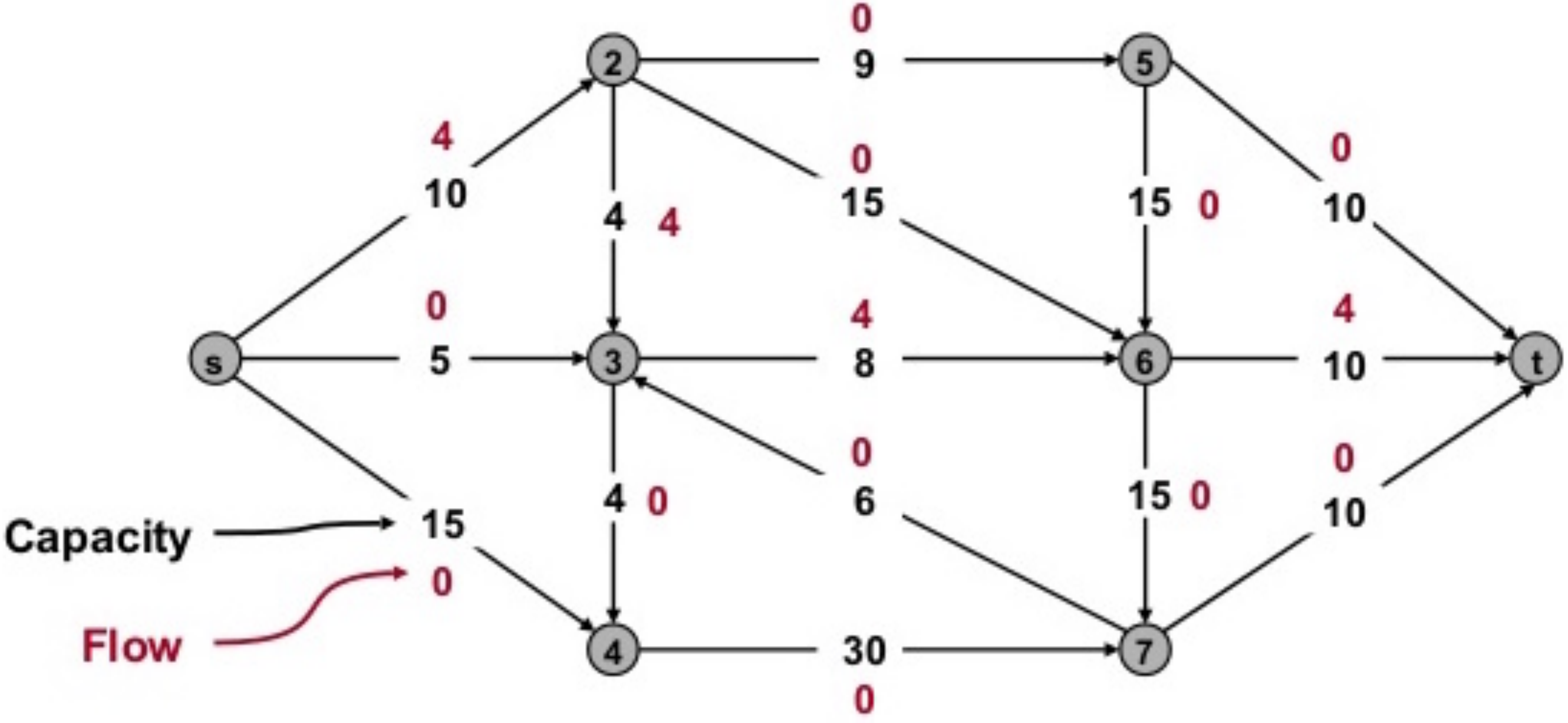
Directed Graph G=(V,E), nonnegative capacity c(e) (sometimes u(e) is used) for each edge e
unique source s and sink t, no edges into s or out of t,



Recall: An s-t Flow

$\forall e \in E$ have $f(e)$, $0 \leq f(e) \leq c(e)$ (capacity constraints)

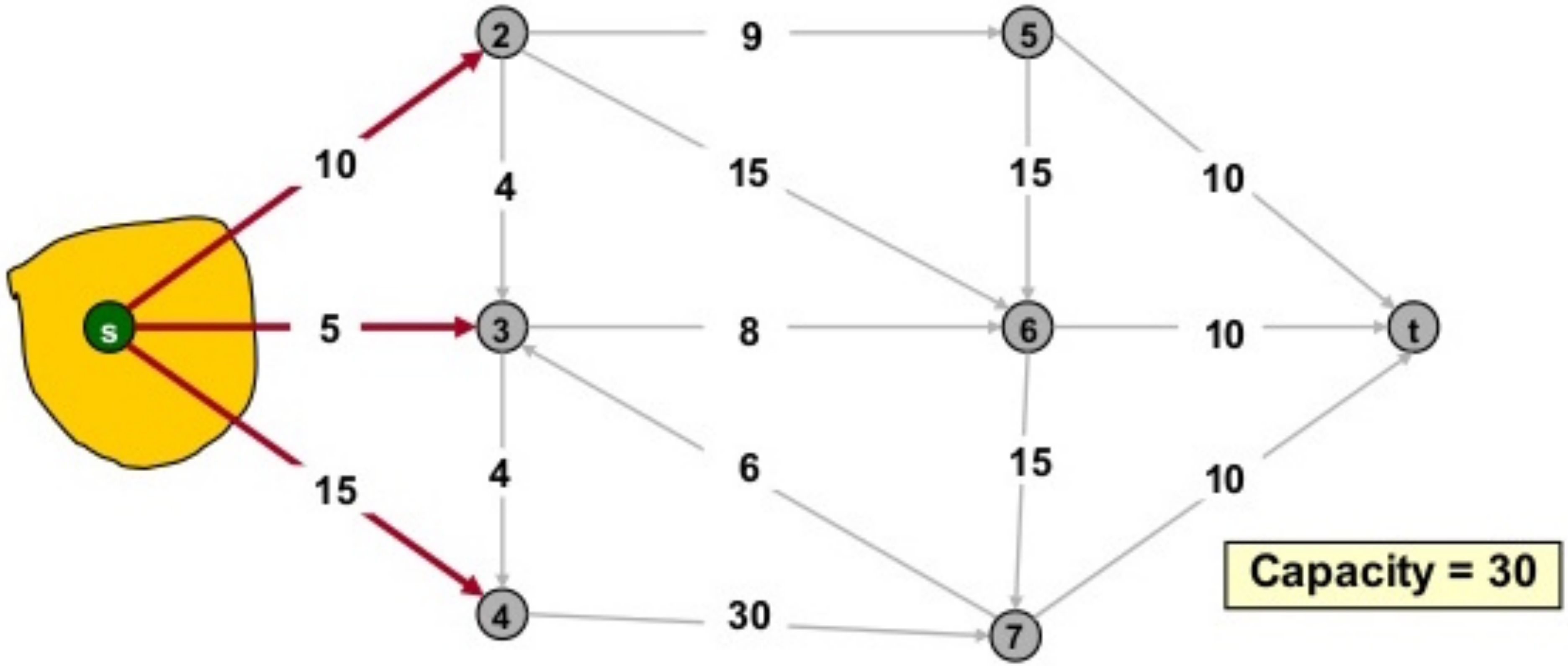
$\forall v \in V$ $\sum_{uv \in E} f(uv) = \sum_{vu \in E} f(uv)$ (flow conservation constraints)



Recall: An s-t Cut

is a partition(A,B) of V with s$\in A, b \in B$.

Its capacity c(A,B)=$\sum_{uv\in E, u\in A, v\in B} c(uv)$



Recall:
The value of a flow and flow across a cut

For all S $\subseteq V$:

$f^{out}(S) = \sum_{uv \in E, u \in S, v \in V-S} f(xy)$

$f^{in}(S) = \sum_{uv \in E, u \in V-S, v \in S} f(xy)$ Type equation here.

$v(f) = f^{out}(\{s\})$

For all s-t cuts (A,B), $v(f) = f^{out}(A) - f^{in}(A) \leq c(A,B)$

Max Flow Min Cut Theorem

For every capacitated network. The maximum volume of an s-t flow is equal to the minimum volume of an s-t cut.

Theorem 7.14 KT

If all the capacities are integer then there is a maximum volume s-t flow f which is integer valued. Can find this flow in $O(v(f)|E|)$ time.

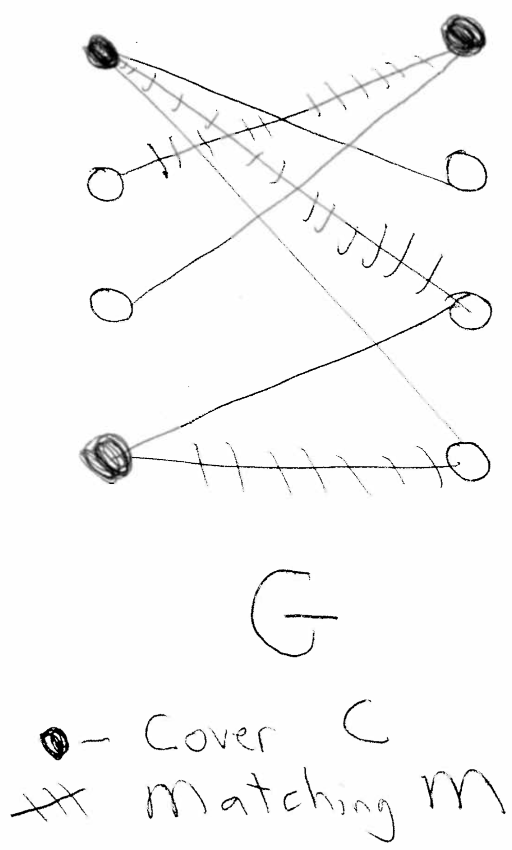
Recall: Matchings and Covers

A matching M is a disjoint set of edges

$C \subseteq V$ is a cover if there is no edge with both endpoints in V-C.

For a matching M and cover C in any G, $|M| \leq |C|$

For Bipartite G:
The Max Size of a Matching=Min Size of a cover



Maximum Number of Edge-Disjoint Paths in a Directed Graph G

V(G')=V(G), E(G')=E(G)-{xs|xs∈ $E(G)\} - \{tx|tx$ ∈E(G)}

Find a integer-valued max value s-t flow f in G'.

Find a set of v(f) edge disjoint paths from s-t in G, only using e if f(e)=1.

Maximum Number of Edge-Disjoint Paths in a Directed Graph G

V(G')=V(G), E(G')=E(G)-{xs|xs∈ $E(G)\} - \{tx|tx$ ∈E(G)}

Find a integer-valued max value s-t flow f in G'.

Find a set of v(f) edge disjoint paths from s-t in G, only using e if f(e)=1.

For any s-t cut (A,B), c(A,B)=|{ab ∈ E|a∈ A, b∈ B}|.

Maximum Number of Edge-Disjoint Paths in a Directed Graph G

V(G')=V(G), E(G')=E(G)-{xs|xs∈ $E(G)\} - \{tx|tx$ ∈E(G)}

Find a integer-valued max value s-t flow f in G'.

Find a set of v(f) edge disjoint paths from s-t in G, only using e if f(e)=1.

For any s-t cut (A,B), c(A,B)=|{ab ∈ E|a∈ A, b∈ B}|.

For any s-t cut (A,B), removing the set {ab ∈ E|a∈ A, b∈ B} separates s from t (i.e. destroys all s-t paths).

Maximum Number of Edge-Disjoint Paths in a Directed Graph G

V(G')=V(G), E(G')=E(G)-{xs|xs∈ $E(G)\} - \{tx|tx$ ∈E(G)}

Find a integer-valued max value s-t flow f in G'.

Find a set of v(f) edge disjoint paths from s-t in G, only using e if f(e)=1.

For any s-t cut (A,B), c(A,B)=|{ab ∈ E|a∈ A, b∈ B}|.

For any s-t cut (A,B), removing the set {ab ∈ E|a∈ A, b∈ B} separates s from t (i.e. destroys all s-t paths).

Menger's Theorem(1927): The maximum size of a set of edge disjoint s-t paths in a directed graph equals the minimum size of a set of edges whose removal separates s from t.

Maximum Number of Edge-Disjoint Paths in an Undirected Graph G

V(G’)=V(G), E(G’)={uv,vu| $\mathrm{uv} \in E(G)\}$ –{ xs|xs$\in E(G)\} - \{tx|tx$ ∈E(G)}

Maximum Number of Edge-Disjoint Paths in an Undirected Graph G

V(G')=V(G), E(G')={uv,vu | $\text{uv} \in E(G)\}$ –{ xs|xs∈ $E(G)\} - \{tx|tx$ ∈E(G)}

A set F of k edge disjoint s-t paths in G yield an s-t flow of value k in G'.

Maximum Number of Edge-Disjoint Paths in an Undirected Graph G

V(G')=V(G), E(G')={uv,vu| $\mathrm{uv} \in E(G)\}$ –{ xs|xs∈ $E(G)\} - \{tx|tx$ ∈E(G)}

A set F of k edge disjoint s-t paths in G yield an s-t flow of value k in G'.

As before, there is an algorithm which given an integer-valued flow f of value k in G' yields a set F of k disjoint s-t paths in G'.

Maximum Number of Edge-Disjoint Paths in an Undirected Graph G

V(G')=V(G), E(G')={uv,vu| $\mathrm{uv} \in E(G)\}$ −{ xs|xs∈ $E(G)\} - \{tx|tx$ ∈E(G)}

A set F of k edge disjoint s-t paths in G yield an s-t flow of value k in G'.

As before, there is an algorithm which given an integer-valued flow f of value k in G' yields a set F of k disjoint s-t paths in G'.

Each such path yields an undirected path in G. These are disjoint except that the path using uv in G' may intersect with the path using vu in G'.

Maximum Number of Edge-Disjoint Paths in an Undirected Graph G

V(G')=V(G), E(G')={uv,vu| $\mathrm{uv} \in E(G)\}$ –{ xs|xs∈ $E(G)\} - \{tx|tx$ ∈E(G)}

A set F of k edge disjoint s-t paths in G yield an s-t flow of value k in G'.

As before, there is an algorithm which given an integer-valued flow f of value k in G' yields a set F of k disjoint s-t paths in G'.

Each such path yields an undirected path in G. These are disjoint except that the path using uv in G' may intersect with the path using vu in G'.

How do we handle this????

Maximum Number of Edge-Disjoint Paths in an Undirected Graph G

V(G')=V(G), E(G')={uv,vu | uv $\in E(G)\}$ –{ xs|xs$\in E(G)\} - \{tx|tx$ $\in$E(G)}

A set F of k edge disjoint s-t paths in G yield an s-t flow of value k in G'.

As before, there is an algorithm which given an integer-valued flow f of value k in G' yields a set F of k disjoint s-t paths in G'.

Each such path yields an undirected path in G. These are disjoint except that the path using uv in G' may intersect with the path using vu in G'.

How do we handle this????

For all uv $\in E(G)$, if f(uv)=f(vu)=1, set f(uv)=f(vu)=0. Then find F.

Airline Scheduling

Airline Scheduling

Set of flights: (Origin, Destination,Departure Time, Arrival Time)

Airline Scheduling

Set of flights: (Origin, Destination,Departure Time, Arrival Time)

Flight j is reachable from flight i if between the arrival time for flight i and the departure time for flight j there is time to due all scheduled maintenance and cleaning, and relocate from the destination of flight i to the origin of flight j.

Airline Scheduling

Set of flights: (Origin, Destination,Departure Time, Arrival Time)

Flight j is reachable from flight i if between the arrival time for flight i and the departure time for flight j there is time to due all scheduled maintenance and cleaning, and relocate from the destination of flight i to the origin of flight j.

We want to find out if k planes can handle all the flights.

Airline Scheduling Via Edge Disjoint Paths

Airline Scheduling Via Edge Disjoint Paths

Build a digraph with two vertices u_i and v_i for each flight.

There is an edge from u_i to v_i for every i.

Airline Scheduling Via Edge Disjoint Paths

Build a digraph with two vertices u_i and v_i for each flight.

There is an edge from u_i to v_i for every i.

There is an edge from v_i to u_j if flight j is reachable from flight i.

Airline Scheduling Via Edge Disjoint Paths

Build a digraph with two vertices u_i and v_i for each flight.

There is an edge from u_i to v_i for every i.

There is an edge from v_i to u_j if flight j is reachable from flight i.

There is a source s and an edge from s to every u_i.

There is a sink t and an edge from every v_i to t.

Airline Scheduling Via Edge Disjoint Paths

Build a digraph with two vertices u_i and v_i for each flight.

There is an edge from u_i to v_i for every i.

There is an edge from v_i to u_j if flight j is reachable from flight i.

There is a source s and an edge from s to every u_i.

There is a sink t and an edge from every v_i to t.

A path P from s to t points out a set of flights which can be handled by one plane (those for which u_iv_i is an edge of P).

Airline Scheduling Via Edge Disjoint Paths

Build a digraph with two vertices u_i and v_i for each flight.

There is an edge from u_i to v_i for every i.

There is an edge from v_i to u_j if flight j is reachable from flight i.

There is a source s and an edge from s to every u_i.

There is a sink t and an edge from every v_i to t.

A path P from s to t points out a set of flights which can be handled by one plane (those for which u_iv_i is an edge of P).

We can schedule the flights using k planes if there is a set of at most k edge-disjoint paths of G using the edge u_iv_i for every flight i.

Airline Scheduling Via Bipartite Matching

For any schedule using exactly k planes to handle f flights, the corresponding k edge disjoint paths use k edges out of s, k edges into t, the f edges of the form $u_i v_i$, and f-k edges of the form $v_i u_j$ where flight j is reachable from flight i.

Airline Scheduling Via Bipartite Matching

For any schedule using exactly k planes to handle f flights, the corresponding k edge disjoint paths use k edges out of s, k edges into t, the f edges of the form $u_i v_i$, and f-k edges of the form $v_i u_j$ where flight j is reachable from flight i.

These last f-k edges form a matching.

Airline Scheduling Via Bipartite Matching

For any schedule using exactly k planes to handle f flights, the corresponding k edge disjoint paths use k edges out of s, k edges into t, the f edges of the form u_iv_i, and f-k edges of the form v_iu_j where flight j is reachable from flight i.

These last f-k edges form a matching.

The desired k paths exist precisely if there is a matching of size f-k in the graph G' whose edge set is $\{v_ju_i \mid$ flight i is reachable from flight j$\}$.

Airline Scheduling Via Bipartite Matching

For any schedule using exactly k planes to handle f flights, the corresponding k edge disjoint paths use k edges out of s, k edges into t, the f edges of the form $u_i v_{i,}$ and f-k edges of the form $v_i u_j$ where flight j is reachable from flight i.

These last f-k edges form a matching.

The desired k paths exist precisely if there is a matching of size f-k in the graph G' whose edge set is $\{v_j u_i |$ flight i is reachable from flight j$\}$.

So the number of planes needed is the minimum k for which there is a matching of size n-k in G'. Given the matching we can find the schedule easily, so the time to find an optimal schedule is O(f^3).

Sequential Scheduling Via Bipartite Matching

Image Segmentation Graph.

Graph G=(V,E). V is the set of pixels, E joins neighbours.

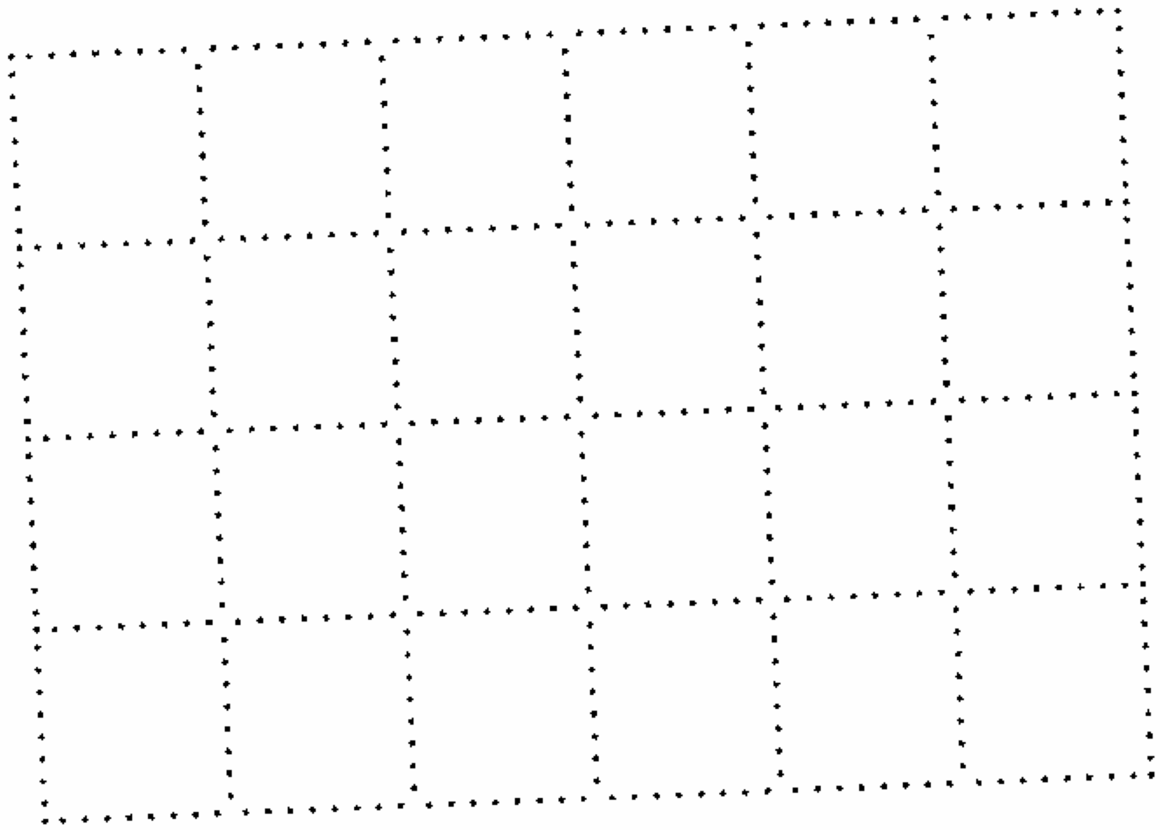


Image Segmentation Problem

- For each pixel I, have: a_i-likelihood i is in the foreground and
  b_i-likelihood i is in the background.

Image Segmentation Problem

- For each pixel I, have: a_i-likelihood i is in the foreground and
  b_i-likelihood i is in the background.
- For each edge ij of the Image Segmentation graph have separation penalty p_{ij} incurred if we make different choices for i and j.

Image Segmentation Problem

- For each pixel I, have: a_i-likelihood i is in the foreground and

  b_i-likelihood i is in the background.
- For each edge ij of the Image Segmentation graph have separation penalty p_{ij} incurred if we make different choices for i and j.
- We want to find a partition of the pixels into a set F of foreground pixels and a set B of background pixels so as to maximize:

$$\sum_{i \in F} a_i + \sum_{i \in B} b_i - \sum_{ij \in E, i \in F, j \in B} p_{ij}$$

Image Segmentation Problem

- For each pixel I, have: a_i-likelihood i is in the foreground and

  b_i-likelihood i is in the background.
- For each edge ij of the Image Segmentation graph have separation penalty p_{ij} incurred if we make different choices for i and j.
- We want to find a partition of the pixels into a set F of foreground pixels and a set B of background pixels so as to maximize:

$$\sum_{i\in F} a_i + \sum_{i\in B} b_i - \sum_{ij\in E, i\in F, j\in B} p_{ij}$$

- This is the same as minimizing

$$\sum_{i\in B} a_i + \sum_{i\in F} b_i + \sum_{ij\in E, i\in F, j\in B} p_{ij}$$

Image Segmentation Via Min Cut

- Construct a capacited network whose vertex set is the pixels, s and t.
- Add an edge s_i for every pixel i with capacity a_i
- Add an edge ti for every pixel i with capacity b_i
- For every two neighbouring pixels ij, add edges ij and ji both with capacity p_{ij}

Image Segmentation Via Min Cut

- Construct a capacited network whose vertex set is the pixels, s and t.
- Add an edge s_i for every pixel i with capacity a_i
- Add an edge ti for every pixel i with capacity b_i
- For every two neighbouring pixels ij, add edges ij and ji both with capacity p_{ij}
- There is a bijection between s-t cuts and choices for the foreground F and background. The cut corresponding to (F,B) is (s+F,t+B).

Image Segmentation Via Min Cut

- Construct a capacited network whose vertex set is the pixels, s and t.
- Add an edge s_i for every pixel i with capacity a_i
- Add an edge ti for every pixel i with capacity b_i
- For every two neighbouring pixels ij, add edges ij and ji both with capacity p_{ij}
- There is a bijection between s-t cuts and choices for the foreground F and background. The cut corresponding to (F,B) is (s+F,t+B).
- We are looking for the minimum capacity cut.