Lecture 18: "Applications" of Max Flow-Min Cut Theorem
Date: April 1, 2026 **Scribe:** Lily Renneker

Today we will cover two applications of the Max Flow-Min Cut Theorem introduced last lecture. The first is a mathematical application, specifically using the theorem to prove Menger's Theorem. The second is a practical application, where we will see how the theorem can be used to optimize airline flight planning.

1 Application 1

Let $G = (V, E)$ be a directed graph (*although we will see directed or undirected doesn't matter*). Let $s, t \in V$ be distinct. Two paths, P and P' from s to t are:

1. **edge disjoint** if they do not share an edge
2. **vertex disjoint** if there is no vertex u different from s and t ($u \neq s, t$) such that u is in both P and P'. This is also called internally vertex disjoint.

Definition 1.1. *an s-t **edge cut** is a subset $C \subseteq E$ such that every s-t path contains an edge in C*

Definition 1.2. *an s-t **vertex cut** is a subset $U \subseteq V \setminus \{s, t\}$ such that every s-t path contains a node in U*

Our goal is to prove the min-max relation between disjoint s-t paths and s-t cuts

Theorem 1.3. *(Menger's) Let $G = (V, E)$ be a directed graph, and $s, t \in V$ be distinct. Then:*

1. *the maximum number of edge disjoint s-t paths is the minimum size of an s-t edge cut*
2. *the maximum number of vertex disjoint s-t paths is the minimum size of an s-t vertex cut*

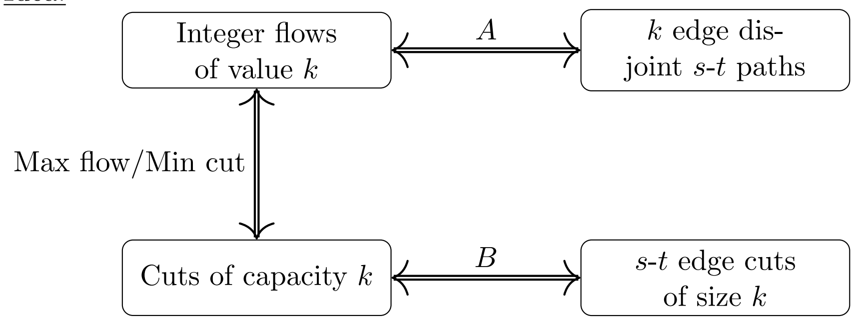
these also hold if G is undirected

Proof. In (1), the maximum number of edge-disjoint s-t paths is less than the minimum cardinality of an s-t edge cut since each of these paths must use a distinct edge in the cut. Through a similar argument, in (2), the maximum number of vertex disjoint s-t paths must be less than or equal to the minimum cardinality of an s-t vertex cut.

It remains to prove the opposite directions for both (1) and (2).

For (1), let $c_e = 1 \ \forall e \in E$, this is our "gadget"

Idea:



A: an integer flow of value k decomposes as k edge disjoint s-t paths: find a unit flow s-t path supported in the flow subtracted from the flow and iterate. This works because $c_e = 1$.

B: Given an s-t edge cut C of size k, let S be the set of nodes reachable from source s in the induced graph $G' = (V, E \setminus C)$, and $T = V \setminus S$. Then, (S, T) is an s-t cut of capacity k.

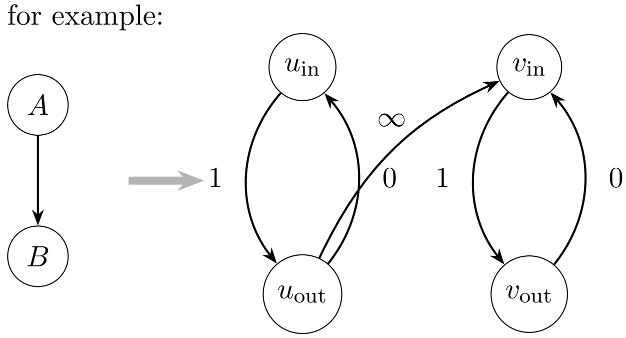
Conversely, if (S, T) is an s-t cut with a capacity $C(S, T) = \sum_{e \in \delta(S,T)} c_e = \sum_{e \in \delta(S,T)} 1 = |\delta(S, T)| = k$

Then $\delta(S, T)$ is an s-t edge cut of size k.

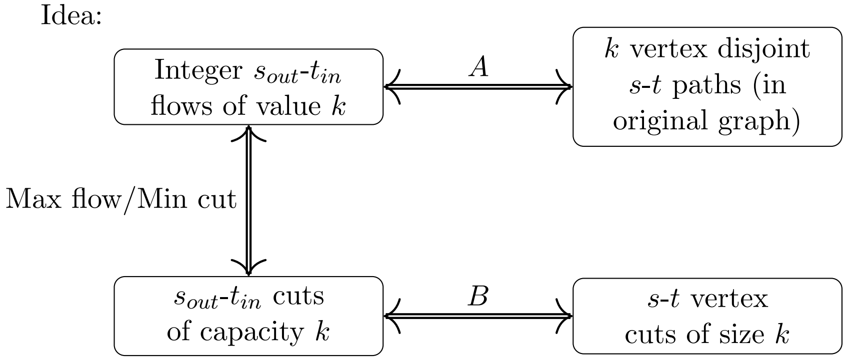
For (2), the gadget is a bit more involved. For each node in the graph, $u \in V$ form two new nodes: u_{in} and u_{out} with $c(u_{in}, u_{out}) = 1$ and $c(u_{out}, u_{in}) = 0 \ \forall u \in V$.

Additionally, each $e \in (u, v) \in E$ in the original graph becomes an edge (u_{out}, v_{in}) in the new graph with capacity, $c(u_{out}, v_{in}) = \infty$

for example:



Idea:



A: Same argument for 1A (edge-disjoint paths case). Only edges from u_{in} is to u_{out} and you use that node.

B: Minimum cuts have finite capacity, meaning we do not use any infinite capacity edges fro the cuts. So, in any finite capacity s_{out}-t_{in} cut (S, T), say of capacity k, only edges of the form (u_{in}, u_{out}) can appear in $\delta(S, T)$.

$U = \{u \in V : (u_{in}, u_{out}) \in \delta(S, T)\}$ forms an s-t vertex cut with size k. The reverse construction is the same.

Finally, for the undirected graph case, we can bidirect G and apply the same proof ☐

2 Application 1: Airline Scheduling

Supposed you want to operate the flights:

Departure city	Departure time		Arrival city	Arrival time
BOS	6:00 AM	→	DCA	7:00 AM
DCA	8:00 AM	→	LAX	11:00 AM
PHL	7:00 AM	→	PIT	8:00 AM
DEN	5:00 PM	→	SFO	6:00 PM
SFO	2:15 PM	→	SEA	3:15 PM
PHL	11:00 AM	→	SFO	2:00 PM

There are m flight segments you need to operate, but you only have k planes (presumably $m > k$). More generally, is it possible to operate all of these flights with your fleet size. How do we reuse planes for different flight segments?

We can turn this into a flow problem (sort of, after some complicated gadgets).

Definition 2.1. *a circulation with demands* $\{d_u\}_{u \in G} \in \mathbb{R}_{\geq 0}$*, capacities* $c : E \to \mathbb{R}_{\geq 0}$*, and lower bounds* $l : E \to \mathbb{R}$ *such that* $l_e \leq c_e \ \forall e \in E$*, is a vector* f *such that:*

1. $\forall e \in E$, $l_e \leq f_e \leq c_e$
2. $f(\delta^-(u)) - f(\delta^+(u)) = d_{u_1} \ \forall u \in V$

Circulations generalize flows:

In flows, $l_e = 0 \ \forall e \in E$ and

$$d_u = \begin{cases} \lambda & \text{if } u = t \\ -\lambda & \text{if } u = s \\ 0 & \text{otherwise} \end{cases}$$

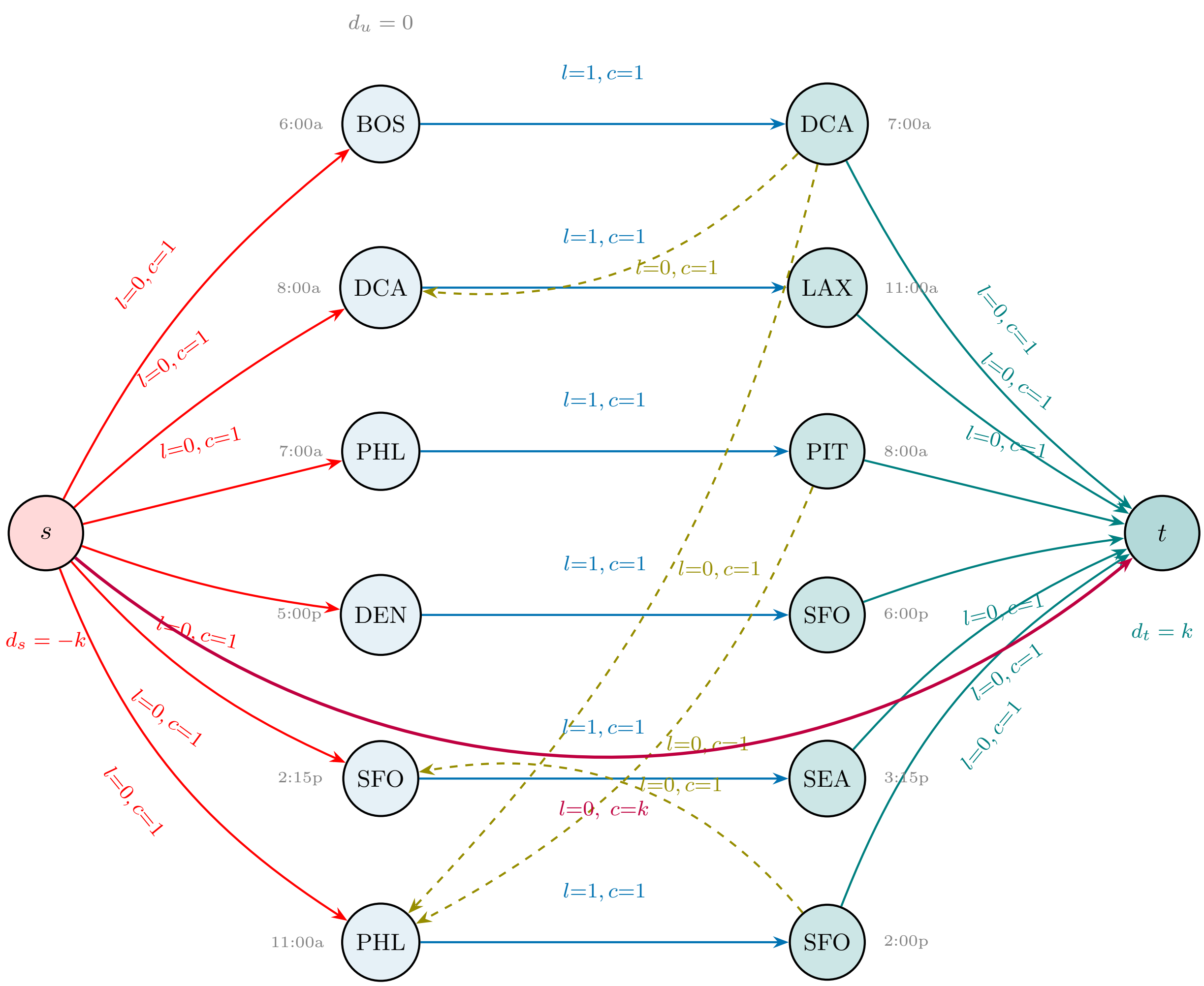
where λ is a flow value.

There are gadgets to cast the existence of a circulation as given as a questions about the existence of flows of a given value. So, if you know about flows, you know about circulations.

Idea (and subsequent diagram):

1. set capacity = 1 and lower bound = 1 for all edges (for each required flight segment

2. add flight destination/origin compatibility
3. add a super source s and connect all required flight origins with $l_e = 0$ and $c_e = 1$
4. add a super target t and connect it to all required flight destinations with $l_e = 0$ and $c_e = 1$
5. add an edge s, t with with $l_{s,t} = 0$ and $c_{s,t} = k$
6. finally, $d_s = -k$, $d_t = k$, and $d_u = 0$ otherwise.



Once the gadgets are set up, then we can use the algorithms from last class to solve.