Algorithm Design and Analysis

Minimum-cost Flows

Roadmap for today

- Another flow problem, ***minimum-cost flows***
- The ***cheapest augmenting paths*** algorithm
- The ***cycle cancelling*** algorithm

Network Flow recap recap

- A ***flow network*** is a directed graph with:
  - ***capacities*** $c(u,v)$
  - A ***source vertex*** s and ***sink vertex*** t
- A ***flow*** is an assignment of values to edges:
  - ***Capacity constraint:*** $0 \leq f(u,v) \leq c(u,v)$
  - ***Conservation constraint:*** "flow in = flow out" for all vertices except s, t

$$\sum_{v \in V} f(u,v) = \sum_{v \in V} f(v,u)$$

- The value of a flow is the net flow out of the source (can prove via conservation that is = net flow into sink)

Network Flow recap

- The ***maximum flow*** problem is to find a flow of maximum value
- We learned the ***Ford-Fulkerson*** algorithm:
  - Define the ***residual network:***

$$c_f(u,v) = c(u,v) - f(u,v)$$

$$c_f(v,u) = c(v,u) + f(u,v)$$

  Then the algorithm is:

  while there exists a path in the residual network:
    add flow to that path.

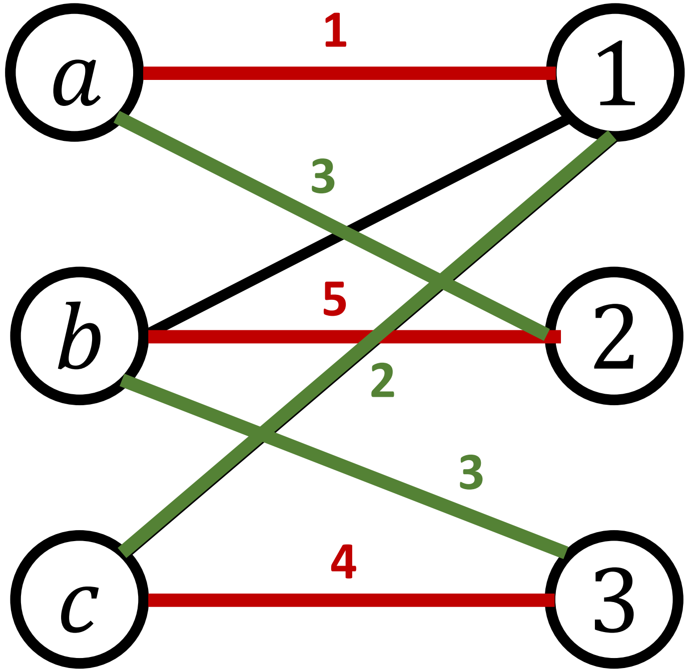
- Last lecture we saw ***Edmonds-Karp*** and ***Dinic's***.

Motivation

- There can be multiple maximum flows in a particular network

- What if we want to preference some over others?

- Example: Bipartite matching allows us to find whether a matching is possible. If there are multiple, can we also have preferences so that we get the "best" matching?



Minimum-cost flows

- We consider the same setting as before: A directed graph with capacities.
- Edges now also have ***costs***. Edge e costs $\$(e)$
- The cost of an edge is **per unit of flow**. The total cost is

$$\sum_{e \in E} \$(e) \cdot f(e)$$

- ***Goal***: Find maximum flow of minimum cost
- ***Note***: Other variants of the problem exist. E.g., you might want the minimum possible cost, regardless of the flow value (not maximum)

Assumptions

- Negative costs are allowed!
- Negative cycles are also allowed!!
  - However, some algorithms don't work.
  - Assume that there is no infinite capacity negative cycle (or the cost is $-\infty$)

The residual network

- The residual network is a powerful tool. Let's keep using it

- If $f(u,v) > 0$, we define the *residual capacities and residual costs*

$$
\begin{aligned}
c_f(u,v) &= c(u,v) - f(u,v) \qquad & \$_f(u,v) &= \$(u,v) \\
c_f(v,u) &= c(v,u) + f(u,v) \qquad & \$_f(v,u) &= -\$(u,v)
\end{aligned}
$$

- Note: If the input graph contains anti-parallel edges (u,v) and (v,u), then we need to have $\$(u,v) = -\$(v,u)$

An augmenting path algorithm

- Ford-Fulkerson finds a maximum flow (ignoring costs completely)
- What is a natural way to choose the augmenting paths?
- Find a ***cheapest augmenting path***.
- Use Bellman-Ford to find the augmenting paths (why not Dijkstra?)
- Requires no negative cycles in the input network!
- Assume integer capacities as well for termination

Does it work?

- We need two things:
  - ***Question 1:*** Does the algorithm actually terminate?
  - ***Question 2:*** Does it give a minimum-cost flow?

To answer Question 1, we need to prove that G_f never contains a negative-cost cycle! (Or the cheapest path would be undefined).

A powerful lemma

Theorem: Given a network G and flow f such that G_f contains no negative-cost cycles, if we augment a cheapest path, then the result still has no negative-cost cycles.

Lemma: Augmenting a cheapest path does not decrease ~~increase~~ the cost of the cheapest $s - t$ path in the residual network.

Lemma: Augmenting a cheapest path does not ~~increase~~ decrease the cost of the cheapest $s - t$ path in the residual network.

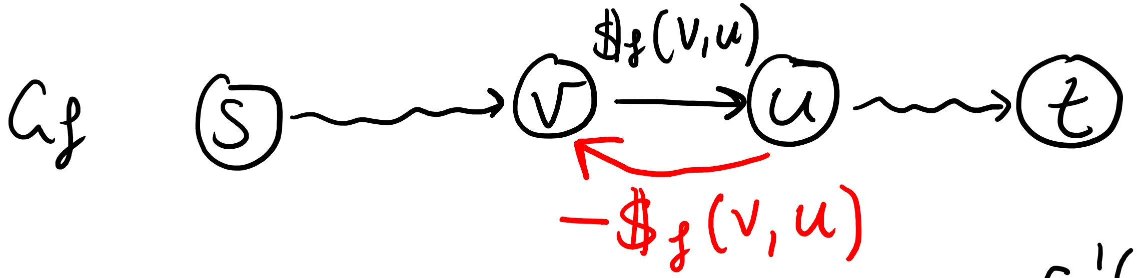
$c(v)$ = cost of $s \to v$ path in G_f $\quad c'(v) = \ldots$ in $G_{f'}$

$$\underline{c'(v) = c'(u) + \$_{f'}(u,v)}$$

$$\underline{c'(v) \geq c(u) + \$_{f'}(u,v)}$$

If $\$_{f'}(u,v) = \$_f(u,v) \Rightarrow c'(v) \geq c(u) + \$_f(u,v)$

If cost didn't change $\qquad \geq c(v)$



$$\$_{f'}(u,v) = -\$_f(v,u)$$

$$c'(v) \geq \underline{c(u) - \$_f(v,u)} = c(v)$$

$$c(u) = c(v) + \$_f(v,u)$$

$$\underline{c(u) - \$_f(v,u)} = c(v)$$

$\square$

Does it work?

- We need two things:
  - ~~***Question 1:*** Does the algorithm actually terminate?~~
  - ***Question 2:*** Does it give a minimum-cost flow?

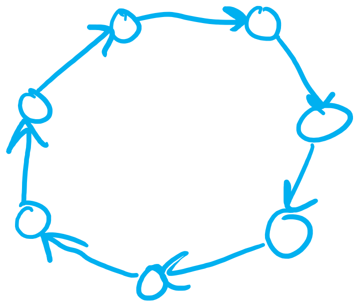
To answer Question 2, we need some more definitions!

Optimality criteria

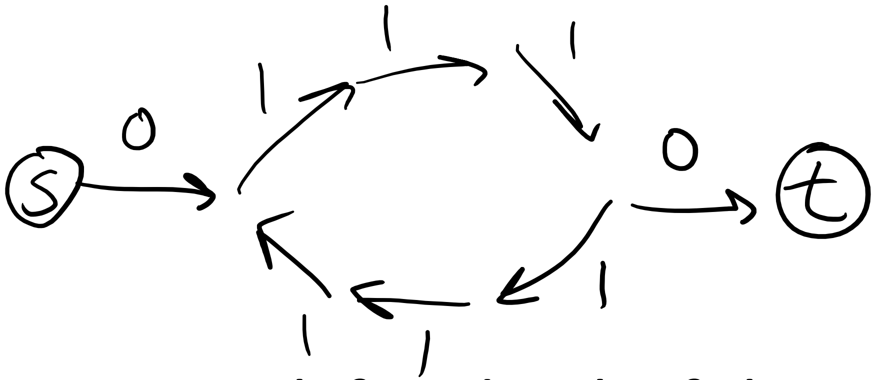
Definition (cost optimal): A flow is ***cost optimal*** if it is the cheapest possible flow of all flows of the same value.

- Recall that a flow is not maximum if and only if there exists an augmenting path
- It would be great if there was an analogy for costs: A flow is not cost optimal if and only if there exists…

Augmenting cycles!



Cost optimality



Theorem (cost optimality): A flow f is cost optimal if and only if there are no negative-cost cycles in G_f

(The cost of a cycle is the sum of the costs of the edges in the cycle)

Exists negative cycle ⇒ not cost optimal

If I augment a negative cycle, cost ↓

Cost optimality

Not cost optimal* ⇒ *exists a negative cycle

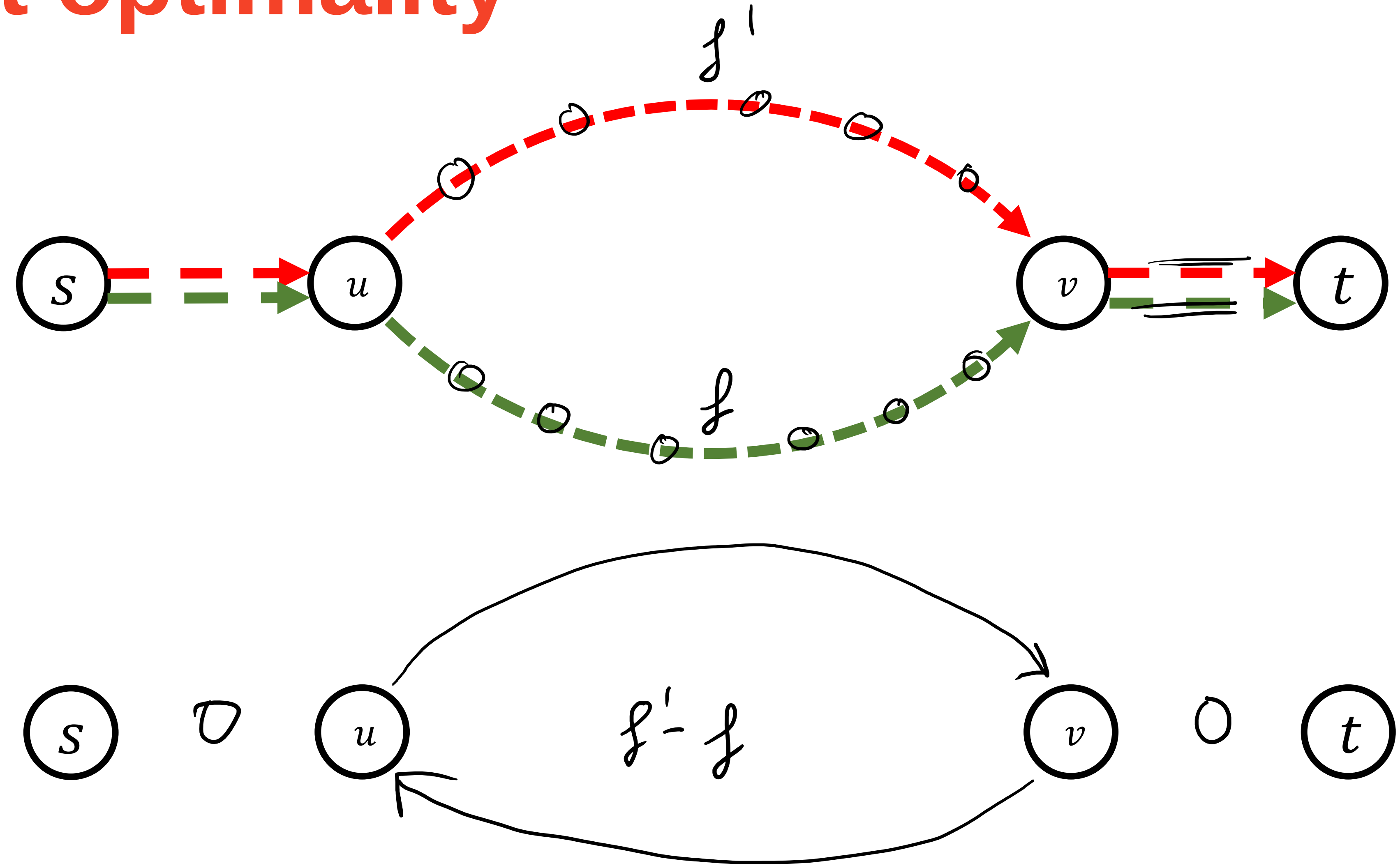
Since f is not cost optimal, there exists f' of **same value but lower cost**

Define $(f' - f)$

If $f'(u,v) \geq f(u,v) \longrightarrow (f'-f)(u,v) = f'(u,v) - f(u,v)$

$f'(u,v) < f(u,v) \longrightarrow (f'-f)(v,u) = f(u,v) - f'(u,v)$

Cost optimality



Cost optimality

- $f' - f$ is called a *circulation*. It's a collections of cycles, a flow of value zero.

Claim: $f' - f$ is feasible in G_f

$$f'(e) - f(e) \leq \underline{c(e) - f(e) = c_f(e)}$$

$$f(u,v) - f'(u,v) \leq \underline{f(u,v) + c(u,v) = c_f(v,u)}$$

$$\mathrm{cost}(f' - f) = \mathrm{cost}(f') - \mathrm{cost}(f) < 0 \qquad \square$$

Completing the analysis

- Cheapest augmenting paths never creates a negative cycle
- Therefore, the flow is always cost optimal
- Augmenting until there are no augmenting paths implies the flow is maximum.
- Therefore, it outputs a ***minimum-cost maximum flow***

Runtime: $O(nmF)$

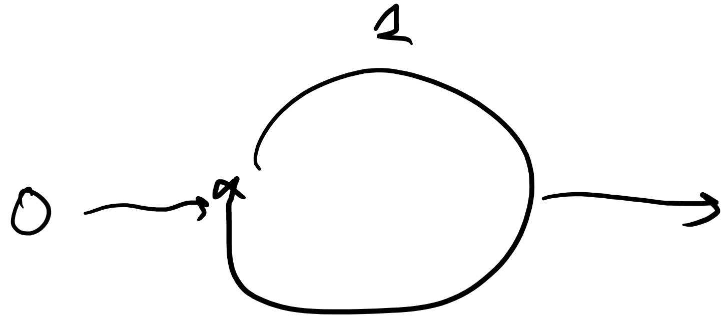
Another algorithm

- Cheapest augmenting paths started with a flow that was cost optimal (the zero flow) then incrementally made it more maximum
- Can we start with a maximum flow and incrementally make it cheaper?

Theorem (cost optimality): A flow f is cost optimal if and only if there are no negative-cost cycles in G_f

(The cost of a cycle is the sum of the costs of the edges in the cycle)

Cycle cancelling



find a maximum flow (e.g., Ford-Fulkerson, or Dinic’s)
while there exists a negative cost cycle in G_f
    augment flow along the negative cost cycle

- Works when the input has negative-cost cycles!
- Assumes integer costs

Runtime: $\overset{C}{\overset{\shortparallel}{\max \$}} \cdot \overset{U}{\overset{\shortparallel}{\max c}} \cdot m \cdot mn$

$= O(m^2 n \cdot C U)$

Summary of today

- The ***minimum-cost flow problem***, and two algorithms
- ***Cheapest augmenting paths***
  - Ford-Fulkerson but always use cheapest cost augmenting path
  - Works for integer-capacity, negative-cycle-free networks
  - Runs in $O(nmF)$ worst-case time
- ***Cycle cancelling***
  - Find max flow and use augmenting cycles until no negative cycles left
  - Works for integer-cost networks. Negative cycles permitted!
  - Runs in $O(nm^2UC)$ worst-case time
- Polynomial-time algorithms exist, but not covered today