II. Network flows

- *Network*
  - Directed graph $G = (V, E)$
  - *Source* $s \in V$, *sink* $t \in V$
  - *Edge capacities* $\text{cap} : E \to \mathbb{R}_+ = \{x \in \mathbb{R} \mid x \geq 0\}$
- *Flow:* $f : E \to \mathbb{R}_+$ satisfying
  1. Flow conservation constraints
  $$\sum_{e:\text{target}(e)=v} f(e) = \sum_{e:\text{source}(e)=v} f(e), \text{ for all } v \in V \setminus \{s, t\}$$
  2. Capacity constraints
  $$0 \leq f(e) \leq \text{cap}(e), \text{ for all } e \in E$$

Maximum flow problem

- *Excess* at node v: $\text{excess}(v) = \sum_{e:\text{target}(e)=v} f(e) - \sum_{e:\text{source}(e)=v} f(e)$
- If f is a flow, then $\text{excess}(v) = 0$, for all $v \in V \setminus \{s, t\}$.
- *Value* of a flow: $\text{val}(f) \stackrel{\text{def}}{=} \text{excess}(t)$
- **Maximum flow problem:**
$$\max\{\text{val}(f) \mid f \text{ is a flow in } G\}$$
- Can be seen as a linear programming problem.

Maximum flow problem (2)

Lemma
If f is a flow, then $\text{excess}(t) = -\text{excess}(s)$.

Proof: We have
$$\text{excess}(s) + \text{excess}(t) = \sum_{v \in V} \text{excess}(v) = 0.$$

- First "=": $\text{excess}(v) = 0$, for $v \in V \setminus \{s, t\}$
- Second "=": For any edge $e = (v, w)$, the flow through e appears twice in the sum, positively in $\text{excess}(w)$ and negatively in $\text{excess}(v)$.

Cuts

- A *cut* is a partition (S, T) of V, i.e., $T = V \setminus S$.
- (S, T) is an (s, t)-*cut* if $s \in S$ and $t \in T$.
- *Capacity* of the cut (S, T)
$$\text{cap}(S, T) = \sum_{E \cap (S \times T)} \text{cap}(e)$$
- A cut is *saturated by f* if $f(e) = \text{cap}(e)$, for all $e \in E \cap (S \times T)$, and $f(e) = 0$, for all $e \in E \cap (T \times S)$.

Cuts (2)

Lemma
If f is a flow and (S,T) an (s,t)-cut, then

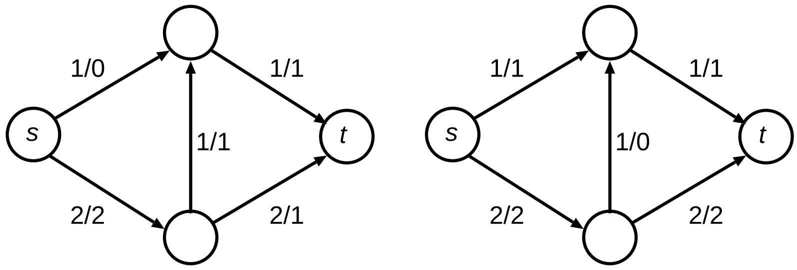
$$\text{val}(f) = \sum_{e \in E \cap (S \times T)} f(e) - \sum_{e \in E \cap (T \times S)} f(e) \leq \text{cap}(S,T).$$

If (S,T) is saturated by f, then $\text{val}(f) = \text{cap}(S,T)$.

Proof: We have

$$\begin{aligned} \text{val}(f) &= -\text{excess}(s) = -\sum_{u \in S} \text{excess}(u) = \sum_{e \in E \cap (S \times T)} f(e) - \sum_{e \in E \cap (T \times S)} f(e) \\ &\leq \sum_{e \in E \cap (S \times T)} \text{cap}(e) = \text{cap}(S) \end{aligned}$$

For a saturated cut, the inequality is an equality.



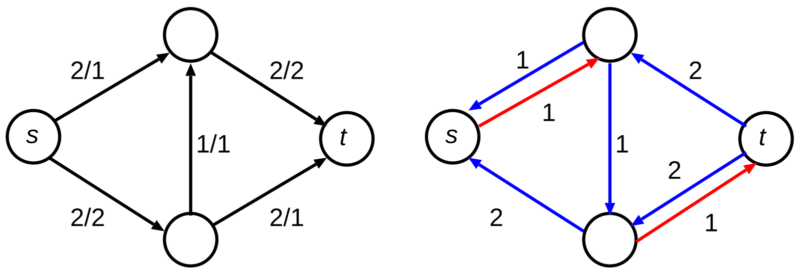
Remarks

- A saturated cut proves the optimality of a flow.
- To show: for every maximal flow there is a saturated cut proving its optimality.

Residual network

The *residual network* G_f for a flow f in $G = (V,E)$ indicates the capacity unused by f. It is defined as follows:

- G_f has the same node set as G.
- For every edge $e = (v,w)$ in G, there are up to two edges e' and e'' in G_f:
  1. if $f(e) < \text{cap}(e)$, there is an edge $e' = (v,w)$ in G_f with *residual capacity* $r(e') = \text{cap}(e) - f(e)$.
  2. if $f(e) > 0$, there is an edge $e'' = (w,v)$ in G_f with residual capacity $r(e'') = f(e)$.



Theorem
Let f be an (s,t)-flow, let G_f be the residual network w.r.t. f, and let S be the set of all nodes reachable from s in G_f.

1. If $t \in S$, then f is not maximum.
2. If $t \notin S$, then $(S, V \setminus S)$ is a saturated cut and f is maximum.

Proof

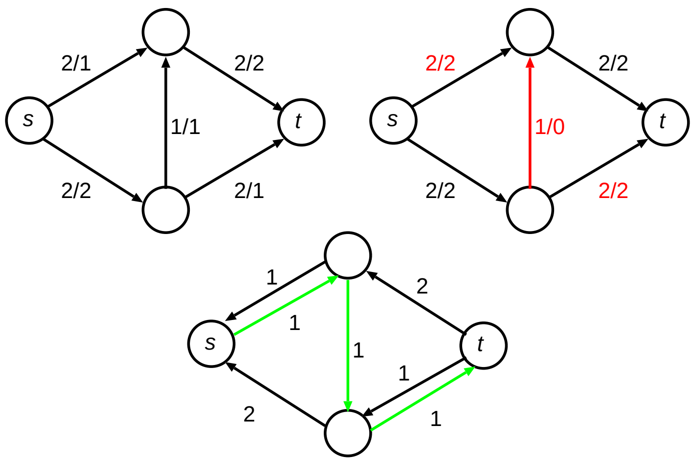
If t is reachable from s in G_f, then f is not maximal.

- Let P be a path from s to t in G_f.
- Let δ be the minimum residual capacity of an edge in P.
  By definition, $r(e) > 0$, for all edges e in G_f. Therefore, $\delta > 0$.
- Construct a flow f' of value $\text{val}(f) + \delta$:

$$f'(e) = \begin{cases} f(e) + \delta, & \text{if } e' \in P \\ f(e) - \delta, & \text{if } e'' \in P \\ f(e), & \text{if neither } e' \text{ nor } e'' \text{ belongs to } P. \end{cases}$$

- f' is a flow and $\text{val}(f') = \text{val}(f) + \delta$.

Example



If t is not reachable from s in G_f, then f is maximal.

- Let S be the set of nodes reachable from s in G_f, and let $T = V \setminus S$.
- There is no edge (v, w) in G_f with $v \in S$ and $w \in T$.
- By the definition of G_f:
  - $f(e) = \text{cap}(e)$, for any $e \in E \cap (S \times T)$, and
  - $f(e) = 0$, for any $e \in E \cap (T \times S)$.
- Thus S is saturated and, by the Lemma, f is maximal.

Ford-Fulkerson Algorithm (1955)

1. Start with the zero flow, i.e., $f(e) = 0$, for all $e \in E$.
2. Construct the residual network G_f.
3. Check whether t is reachable from s in G_f.

- if not, stop.
- if yes, increase the flow along an *augmenting path*, and iterate.

Analysis

- Let $|V| = n$ and $|E| = m$.
- Each iteration takes time $O(n+m)$.
- If capacities are arbitrary reals, the algorithm may run forever.

Integer capacities

- Suppose capacities are integers, bounded by C.
- $v^* \stackrel{\text{def}}{=}$ value of maximum flow $\leq Cn$.
- All flows constructed are integral (proof by induction).
- Every augmentation increases flow value by at least 1.
- Running time $O((n+m)v^*) \rightsquigarrow$ *pseudo-polynomial algorithm*

Edmonds-Karp Algorithm (1972)

- Compute a *shortest* augmenting path, i.e. with a minimum number of arcs.
- Apply breadth-first search (or Dijkstra's algorithm).
- Number of iterations is bound by nm, leads to an $O(nm^2)$ maximum flow algorithm.
- Works also for irrational capacities.

More efficient algorithms exist:

- $O(n^2 m)$ (Dinic, Push-Relabel)
- $O(n^3)$ (FIFO Push-Relabel, ...)

Max-Flow Min-Cut Theorem

Theorem (Ford-Fulkerson 1954)
For a network (V, E, s, t) with capacities $\text{cap} : E \to \mathbb{R}_+$ the maximum value of a flow is equal to the minimum capacity of an (s, t)-cut:

$$\max\{\text{val}(f) \mid f \text{ is a flow}\} = \min\{\text{cap}(S, T) \mid (S, T) \text{ is an } (s, t)\text{-cut}\}$$

Corollary
For integer capacities $\text{cap} : E \to \mathbb{Z}_+$, there exists an integer-valued maximum flow $f : E \to \mathbb{Z}_+$.