1 Maximum Flow

1.1 Definitions

Tarjan: *Data Structures and Network Algorithms*
Ford and Fulkerson, *Flows in Networks*, 1962 (paper 1956)
Ahuja, Magnanti, Orlin *Network Flows*. Problem: do min-cost.
Problem: in a graph, find a *flow* that is *feasible* and has maximum *value*.
Directed graph, edge *capacities* $u(e)$ or $u(v, w)$. Why not c? reserved for costs, later.
source s, *sink* t
Goal: assign a *flow* value to each edge: (net flow form)

- *skew symmetry:* $f(v, w) = -f(w, v)$
- *conservation:* $\sum_w f(v, w) = 0$ unless $v = s, t$
- *capacity:* $f(e) \leq u(e)$ (flow is *feasible/legal*)

Alternative formulation: (gross flow form)

- *conservation:* $\sum_w f(v, w) = 0$ unless $v = s, t$
- *capacity:* $0 \leq f(e) \leq u(e)$ (flow is *feasible/legal*)

Equivalence: second formulation has "gross flow" g, first has "net flow" $f(v, w) = g(v, w) - g(w, v)$. To go other way, sign of f defines "direction" of flow in g.
We'll focus on net flow model for now.
Flow *value* $|f| = \sum_w f(s, w)$ (in net model).
Water hose intuition. Also routing commodities, messages under bandwidth constraints, etc. Often "per unit time" flows/capacities.
Maximum flow problem: find flow of maximum value.
Path decomposition (another picture):

- claim: any s-t flow can be decomposed into paths with quantities
- proof: induction on number of edges with nonzero flow
- if s has out flow, find an s-t path (why can we? conservation) and kill
- if some vertex has outflow, find a cycle and kill
- corollary: flow into t equals flow out of s (global conservation)

Cuts:

- partition of vertices into 2 groups
- s-t-cut if one has s, other t
- represent as $(S, \overline{S})$ or just S

- $f(S)$ = net flow leaving S
- lemma: for any s-t cut, $f(S) = |f|$ (all cuts carry same flow)
  - suppose move v from $\overline{S}$ to S.
  - adds $f(v, S)$ to crossing flow value (no longer subtracting from net)
  - also add $f(v, \overline{S})$ (adding to net flow value)
  - total change: $f(v, S \cup \overline{S}) = 0$

Flows versus cuts:

- Deduce: $|f| \leq u(S) = \sum_{e \in S \times \overline{S}} u(e)$.
- in other words, max-flow $\leq$ minimum s=t cut value.
- soon, we'll see **equal** (**duality**)
- first, need more machinery.

Residual network.

- Given: flow f in graph G
- define G_f to have capacities $u'_e = u_e - f_e$
- if f feasible, all capacities positive
- Since f_e can be negative, some residual capacities **grow**
- Suppose f' is a feasible flow in G_f
- then $f + f'$ is feasible flow in G of value $f + f'$
  - flow
  - feasible
- Suppose f' is feasible flow in G
- then $f' - f$ is feasible flow in G_f (value $|f'| - |f|$)
- **corollary**: max-flows in G correspond to max-flows in G_f
- Many algorithms for max-flow:
  - find some flow f
  - recurse on G_f

How can we know a flow is maximum?

- check if residual network has 0 max-flow
- augmenting path: s-t path of positive capacity in G_f

- if one exists, not max-flow

Max-flow Min-cut

- Equivalent statements:
  - f is max-flow
  - no augmenting path in G_f
  - $|f| = u(S)$ for some S

Proof:

- if augmenting path, can increase f
- let S be vertices reachable from S in G_f. All outgoing edges have $f(e) = u(e)$
- since $|f| \leq u(S)$, equality implies maximum

1.2 Problem Variants

Multiple sinks
Edge lower bounds
Bipartite matching.
Vertex capacities (HW).

1.3 Applications

Matrix Rounding (flow with lower bounds).
$P\|r_j, pmtn\|C_{\max}$

1.4 Algorithms

1.4.1 Augmenting paths

Can always find one.
If capacities integral, always find an integer.

- So terminate
- Running time $O(mf)$
- **Lemma:** flow integrality
- Polynomial for unit-capacity graphs
- Also terminate for rationals (but not polynomial)
- might not terminate for reals.

1.4.2 max capacity augmenting path

How find one?
Running time:

- recall flow decomposition bound
- Get $1/m$ of flow each time.
- So $m \log f$ flows suffice for integer f

weakly vs. strongly polynomial bounds
Works for rational capacities too.

1.5 If little time, scaling

Algorithm: repeatedly

- Shift in one bit at a time
- Run plain augmenting paths

Analysis:

- before bit shift, some cut has capacity 0
- after bit shift, cut has capacity at most m
- so m augmentations suffice.

Running time: $O(m \log U)$.
Discuss relation to max capacity algorithm.

1.5.1 Shortest Augmenting Path

Strongly polynomial.
Lemma: $(s.i)$ and (i, t) distance nondecreasing.

- Among i that got closer to s
- Consider closest to s (after change)
- i has parent j on new shortest path
- j didn't get closer to s
- so (j, i) path got shorter
- so didn't used to have residual (j, i) edge
- so flow went from i to j
- so j was farther than i from s

- now they swapped places
- but j didn't get closer!
- so i must be farher—contra.

Lemma: at most $mn/2$ augmentations.

- Consider edge (i, j) saturated by augmenting path
- Before used again, must push flow on (j, i)
- In first aug, i was closer than j to s
- In next, j was closer than i
- Since no distances go down, must have increased distance of i.
- only happens n times per edge

Running time: $O(m^2n)$.

1.5.2 Blocking Flows

definition of blocking flow
dinic's alg (alg + proof that s-t dist increases)
at the very end I stated BF time bounds and said they'd see more next time