Makeup 3:30–5 on 10/15

1 Min-Cost Flow

Many different max-flows in a graph. How compare?

- *cost* $c(e)$ to send a unit of flow on edge e
- find max-flow minimizing $\sum c(e)f(e)$
- costs may be positive or negative!
- note: pushing flow on cost c edge create residual cost $-c$ edge.
- also easy to find min-cost flow of given value v less than max (add bottleneck source edge of capacity v)

Clearly, generalizes max-flow. Also shortest path:

- How send flow 1 unit of flow?
- just use shortest path
- more generally, flow decompose into paths and cycles
- cost of flow is sum of costs of paths and cycles.
- each path costs at most nC (C = max cost)
- cost of flow at most mUC

Min-cost circulation:

- no source or sink
- just find flow satisfying balance everywhere, min-cost
- if satisfy balance everywhere, all flow must be going in circles!
- more formally: circulation can be decomposed into just cycles.
- hard to define in max-flow perspective, but makes sense once allow negative cost arcs.
- reduction to min-cost flow: add disconnected s, t.
- reduction from min-cost flow:
  - add s-t arc of "infinite" capacity, "infinite" negative cost
  - of course, circulation will push max possible through this edge
  - how much can it? max s-t flow
  - so of course, suff to assign capacity equal to max-flow value

  - see later, sufficient to assign cost $-nU$ (good for scaling)

- another reduction from min-cost flow:
  - find any old max-flow f
  - consider min-cost flow f^*
  - difference $f^* - f$ is a circulation (note: diff of two equal capacity flows is a circulation)
  - so find circulation q in G_f.
  - $q + f$ is a flow in G (note: flow+circulation=flow of same capacity)
  - cost is $c(q) + c(f)$
  - so adding min-cost q in G_f yields min-cost flow

Deciding optimality:

- given a max-flow. How decide optimal?
- by above, optimal if min-cost residual circulation is 0
- suppose not. so have negative cost circulation
- decomposes into cycles of flow
- one must have negative cost.
- so, if f nonoptimal, negative cost cycles in G_f
- converse too: if negative cost cycle, have negative cost circulation. So min-cost< 0.

half a lecture

2 Min-Cost Flows

2.1 Optimality Criteria

Recall: flow/circulation optimal iff no residual negative cost cycle.
Reduced-Cost optimality.

- another way to decide optimal
- based on LP duality (see next week)
- given min-cost flow problem
- one way to solve: compute opt flow (command economy)
- alternative: market economy!
- infinite widget *supply* at source s, infinite *demand* for widgets at source t

- let seller at s, buyer at t set prices for stuff
- creates "market" where prices get set at all vertices
- *price* $p(v)$ for widget at vertex v
- costs reflect shipping charges

When is a set of prices "stable"?

- suppose have $p(v), p(w), c(vw)$ such that $p(w) \geq p(v) + c(vw)$
- then shipper can make money by buying at v, shipping to w, and selling.
- *reduced cost* $c_p(vw) = c(vw) + p(v) - p(w)$
- reduced costs reflect "true cost" of shipping on edge
- merchant will want to ship if reduced cost negative.
- this will increase demand at v, raise price: so prices wrong!
- what stops him? no residual capacity!
- a price function is *feasible* for a (residual) graph if no (residual) arc has negative reduced cost

Important observation: prices don't affect overall cost:

- reduced cycle cost same as original cycle cost
- cost of every v, w path changes by $p(v) - p(w)$
- negative cost cycles still negative

Claim: A circulation/flow is optimal if there exists a feasible price function on its residual graph.

- first: price function implies feasible
- recall: optimal iff no negative cost cycles in residual graph
- suppose have feasible price function
- so no negative reduced-cost edges
- so no negative cost cycles under reduced costs
- thus no negative cost cycles under original costs!
- key: cost of cycle unchanged under reduced costs (prices telescope)!
- converse: suppose have optimal flow
- then residual graph has no negative cycles. How find prices?

  - attach vertex s' with 0-cost edges to everywhere
  - compute shortest residual paths from s
  - well defined because no negative cycles
  - 0-edges guaranteed finite price at all vertices (this why didn't do paths from s)

- define $p(v) = d(s', v)$
- (note: yields residual cost of shipping a unit from s')
- note $p(w) \leq p(v) + c(vw)$
- thus $c_p(vw) \geq 0$ everywhere, as desired.
- **note:** given min-cost flow, can verify optimality via one shortest path computation!
- **note:** Using this computation, all edges on shortest paths from s' have reduced cost exactly 0 (useful for later).

Complementary Slackness:

- what do merchants do? think about reduced costs.
- if reduced cost positive, noone will ship: flow 0 on edge
- if negative, will ship: flow equals capacity on edge
- if 0, don't care: flow arbitrary on edge.
- flow with these properties has *complementary slackness:*, another optimality condition.
- complementary slackness implies optimal, since no residual negative reduced cost arcs
- suppose optimal. assign prices so no residual negative cost arcs. implies any negative reduced cost original arc is saturated, any positive reduced cost arc empty (since reverse would have neg cost)

Complementary slackness is on **original graph**, corresponds to feasible pricing on **residual graph**
Given feasible price function, can find opt flow easy:

- delete positive redued cost arcs (no flow in optimum)
- saturate negative arcs
- creates excesses/deficits at nodes
- ship excesses to deficits on 0-cost arcs

- know this can be done, since optimum does
- do by creating supersource for excesses, supersink for deficits, finding max-flow on 0 arcs

saw converse: given flow, need to compute optimum distances. So min-cost flow really is max-flow plus shortest paths!

- some flow algs use prices implicitly, to prove correctness
- others use explicitly, to guide solution.

3 Algorithms

3.1 Cycle Canceling Algorithm

(Klein)
Cycle canceling algorithm:

- start with any max-flow (or min-cost circulation problem)
- find a negative cost cycle
- push flow around it
- analogue of generic augmenting paths under circulation reduction
- how many times?
  - cost decreases by 2 each push
  - initial cost (in residual graph) 0
  - final cost at least $-mCU$ (why?)
  - so mCU iterations.

How find negative-cost cycle?

- think back to shortest paths
- dijkstra only works for positive edges
- but Floyd, Bellman-Ford work for negative edges too
- **Unless** have negative cost cycle
- then, of course, shortest paths undefined
- however, Floyd/BF will indentify one negative cycle that is wrecking things.
- Floyd $O(n^3)$, BF $O(nm)$.

- fancy scaling algorithm running in $O(m\sqrt{n}\log C)$ also known.

So: time bound of $O(m^2\sqrt{n}CU\log C)$ or $O(nm^2CU)$ time.
Slow, and not even weakly polynomial! Let's do better.
Later, we'll see that cycle canceling is a good idea. But for now, lets take a different approach.
Point A: Timing to here is 1:20

3.2 Shortest Augmenting Path

Special case algorithm:

- unit capacity edges,
- no negative cost cycles.
- so mincost circ 0, but min-cost flow maybe not
- note *capacity* of flow at most n.

Idea: augmenting paths

- natural greedy strategy: what augmenting path to use?
- Repeatedly augment shortest (min-cost) path

Time analysis: $O(mv\log n)$ for flow of value v, so $O(mn\log n)$
Correctness: uses price function.

- key: price function changes *value* of paths, but not which is shortest (proof: telescoping)
- claim: at no time does residual graph have negative cost cycle
- proof by induction
- if currently true, can compute shortest paths from s to define prices
- result: all edges on shortest paths have cost 0
- so augmentation is along cost 0 edges
- create residual arcs, but only cost 0
- so, no negative reduced cost arcs
- so, no negative cost cycles. induction proved.
- so after v iterations, have flow value v of minimum cost.

Note: don't need explicit prices to run

- prices don't change which paths are shortest

- but with good prices, can use Dijkstra's fast algorithm.

Limitations:

- unit capacity
- no negative cycles
- (so can't do standard mincost circulation reduction)

Devoted substantial time to review of reduced cost optimality, complementary slackness, and shortest augmenting path.

4 Scaling Min-Cost flow

4.1 Capacity Scaling

notice: above algorithm has running time depending on *value* of flow. Doesn't actually care about unit capacity.
Scale in capacity bits one at a time.

- in scaling step, double capacities, possibly add 1
- double flow values
- changes residual capacities by at most 1
- so some negative cost arcs might get capacity 1
- how fix?
- saturate all negative capacity arcs (creates excess, deficits)
- find min-cost flow to ship excesses to deficits (know one exists)
- total excess $O(m)$
- so $O(m)$ shortest augmenting paths solve
- time $O(m^2 \log n)$
- $O(\log U)$ phases
- total $O(m^2 \log n \log U)$.

4.2 Cost Scaling

skip this. takes 15 minutes to survey. just mention exists.
Idea:

- Recall goal: flow/price function such that all residual arcs have nonnegative cost
- On cost scaling, allow negative arcs that are almost nonresidual
- now, allow residual arcs that are almost nonnegative.
- a flow/price-function is ϵ-*optimal* if every residual arc has cost at least $-\epsilon$

Outer loop:

- scaling phase reduces ϵ to $\epsilon/2$
- Any flow is C-optimal (gives starting point)
- problem: may get nonintegral prices. when can we stop?
- (note with flow, knew stayed integral so 1/2-optimal was optimal.
- claim: an $1/n$-optimal flow is optimal
- proof: no negative cycle. cycle cost unchanged, was originally an integer.

Scaling phase. Many different approaches, we'll start with one based on push relabel.

- start by saturating all negative cost arcs. Creates excesses, deficits.
- preserve $\epsilon/2$-optimality (by juggling flow, prices)
- uses pushes to send excesses to deficits
- need notion of "downhill"
- An arc is *admissible* if negative reduced cost
- pushes are only on admissible arcs
- if an active vertex has no admissible arc, *relabel* it.
- reduce its price by $\epsilon/2$
- decreases cost of outgoing arcs, increases incoming
- since no outgoing arc started negative, all outgoing arcs $> \epsilon/2$, so still $\epsilon/2$-optimal.
- no incoming admissible arcs!
- clearly, so long as have excess, can push or relabel

Claim: no vertex relabeled more than $3n$ times.

- Initially, had pushed flow from supersink t to supersource s
- then did some push of active flow
- so can think of current situation as a preflow with source t
- decompose into paths from t ending at current excesses.
- consider path P from t to v
- since started ϵ-optimal, cost of path at least $-\epsilon n$
- so reverse path originally had cost at most ϵn
- this is a residual path in current graph
- so (by $\epsilon/2$-optimality) has cost at least $-\epsilon/2n$
- each relabel of v decreased path cost by $\epsilon/2$
- so only $3n$ relabels.

Claim: $O(nm)$ saturating pushes
Claim: admissible network acyclic
Claim: $O(n^2m)$ nonsaturating

- let $g(v)$ be number of nodes admissibly reachable from v
- $\phi = \sum g(v)$ over active v
- saturating push increases by n (makes dest active)
- relabel increases by n (v can reach more stuff, but noone can reach v)
- nonsaturating push (v, w) decreases by 1 ($g(w) < g(v)$)

Cleverer methods give $O(n^3)$.
Other approaches (eg double-scaling) gives $O(nm)$ per scaling phase.
Finish remarks on min-cost flow.

- Strongly polynomial algorithms exist.
  - Tardos 1985
  - minimum mean-cost cycle
  - reducing ϵ-optimality
  - "fixing" arcs of very high reduced cost
  - best running running time roughly $O(m^2)$
  - best scaling time (double scaling) $O(mn \log\log U \log C)$.

bf Point B. From point A is 1:00