ORIE 6330 Notes

Shawn Ong

September 2020

1 Lecture 1 (9/2/20)

What;s nice about network flows? Rich set of applications, beautiful algorithms, nice analyses, practical implementation

Max $s-t$ flows:

- Input directed graph $G=(V,A)$, capacities $u(i,j)\forall(i,j)\in A$, source $s\in V$, sink $t\in V$
- Goal: find flow f maximizing net flow out of source s

Definition 1. A $s-t$ flow f is $f:A\to\mathbb{R}_{\geq 0}$ s.t.:

(i) $0\leq f(i,j)\leq u(i,j)$ (capacity constraint)

(ii) $\sum_{k:(k,i)\in A} f(k,i)=\sum_{k:(i,k)\in A} f(i,k)$

(iii) if $(i,j)\in A$, then $(j,i)\in A$ and $f(j,i)=-f(i,j)$. (skew symmetry)

To make the last point work, we set $u(j,i)=0$. Adding skew symmetry allows us to rewrite first condition as $f(i,j)\leq u(i,j)$.

value of a flow is $|f|$, amt of flow out of s.

Definition 2. An $s-t$ <u>flow</u> is $f:A\to\mathbb{R}$ such that:

(i) $f(i,j)\leq u(i,j)$ (capacity constraint)

(ii) $\sum_{k:(i,k)\in A} f(i,k)=0$ (flow conservation)

(iii) $f(i,j)=-f(j,i)$ (skew symmetry)

The value of a flow is $|f|\equiv\sum_{k:(s,k})f(s,k)$.

<u>Def:</u> *An $s-t$ cut is $S\subseteq V, s\in S, t\notin S$.*

Take $\delta^+(S)=\{(i,j)\in A: i\in S, j\notin S\}, \delta^-(S)=\{(i,j)\in A: i\notin S, j\in S$.

Definition 3. The capacity of cut S is $u(\delta^+(S))=\sum_{(i,j)\in\delta^+(S)} u(i,j)$

Lemma 4. *For any $s-t$ flow f, $s-t$ cut S, $|f| \leq u(\delta^+(S))$.*

Proof.

$$\begin{aligned}|f| &= \sum_{k:(s,k)\in A} f(s,k) + 0 = \sum_{k:(s,k)\in A} f(s,k) + \sum_{i\in S-\{s\}} \sum_{j:(i,j)\in A} f(i,j) \\ &= \sum_{i\in S} \left(\sum_{j\in S:(i,j)\in A} f(i,j) + \sum_{j\notin S:(i,j)\in A} f(i,j) \right) \\ &= \sum_{i\in S, j\notin S:(i,j)\in A} f(i,j)\end{aligned}$$

(first term vanishes because $f(i,j) = -f(j,i)$. □

Corollary 5. *If we have $s-t$ flow f, $s-t$ cut S such t hat $f(i,j) = u(i,j) \forall (i,j) \in \delta^+(S)$, then $|f| = u(\delta^+(S))$.*

Definition 6. If $f(i,j) = u(i,j)$, then (i,j) is saturated.

Definition 7. The minimum $s-t$ cut S is S^* that minimizes $u(\delta^+(S))$ over all $s-t$ cuts.

Reminiscent of a weak duality statement.

Theorem 8. *(Ford,Fulkerson 1955): the value of a maximum flow is equal to the capacity of a minimum $s-t$ cut.*

Definition 9. Given a flow f, the residual graph $G_f = (V, A)$ has residual capacities $u_f(i,j) = u(i,j) - f(i,j)$.

NOTE: This definition includes the induced edges from skew-symmetry. Residual capacities indicate how much additional flow can be added to/removed from an edge.

Definition 10. An $s-t$ path in G_f in which all arcs have positive residual capacity is an augmenting path.

Given flow f, graph G_f, augmenting path P, $\delta = \min_{(i,j)\in P} u_f(i,j) > 0$. Then take:

$$f'(i,j) = \begin{cases} f(i,j) + \delta & \forall (i,j) \in P \\ f(i,j) - \delta & \forall (j,i) \in P \\ f(i,j) & \text{otherwise} \end{cases}$$

Call this *pushing* δ flow along P. We first check whether this is still a flow:

- Skew-symmetry: $f'(i,j) = f(i,j) + \delta = -f(j,i) + \delta = -(f(j,i) - \delta) = -f'(j,i)$; other cases easy.
- Capacity constraint: $f'(i,j) = f(i,j) + \delta \leq f(i,j) + u_f(i,j) = u(i,j)$
- Flow conservation: total flow in/out remains unchanged b/c of paired edges.

After pushing δ, we have $|f'| = |f| + \delta$.

Theorem 11. *TFAE:*

1. *f is a maximum flow.*
2. *There is no augmenting path P in G_f.*
3. *There is an $s-t$ cut S s.t. $|f| = u(\delta^+(S))$.*

Proof. (1) $\implies$ (2): Just showed contrapositive (if augmenting path exists, can push more flow).

(2) $\implies$ (3): Take S to be the vertices reachable from s via positive residual capacities. Note that $t \notin S$ because no augmenting paths exist. Then for $(i,j) \in A, i \in S, j \notin S$, we have $f(i,j) = u(i,j)$. Follows from corollary.

(3) $\implies$ (1): Since $|f| \leq u(\delta^+(S))$ for any $s-t$ cut S, equality implies max flow and min cut. □

2 Lecture 2 (9/7/20)

Algorithm 1 Ford-Fulkerson algorithm

```
f ← 0
while ∃ augmenting path in G_f do
    Push flow along P
    Update f
end while
Return f
```

Integrality property: If all capacities $u(i,j)$ are integer, then there is a max flow f such that all $f(i,j)$ are integers. Moreover, this flow can be found via our algorithm (invariant is maintained by alg).

How efficient is the algorithm? Take $U = \max_{(i,j)\in A} u(i,j), m = |E|, n = |V|$. Easy upper bound on flow value mU. Since flow value increases by at least 1 each time, this corresponds to an upper bound on the number of iterations of the algorithm.

If runtime of finding augmenting path/updating is $O(m)$, then overall runtime would be $O(m^2U)$.

Definition 12. An algorithm runs in pseudopolynomial time if the number of steps can be bounded by a polynomial when input is encoded in unary.

Definition 13. An algorithm runs in strongly polynomial time if the runtime can be bounded by a polynomial in the number of input data items, e.g. $poly(n,m)$.

So $O(m^2U)$ is not poly-time, but it is pseudopolynomial. Example of something "normal" poly-time is $O(mn \log U)$.

Now consider the densest subgraph problem.

- Input $G = (V, E)$ (undirected).
- Let $E(S)$ = set of edges with both endpoints in S.
- Let $\delta(S)$ = set of edges with exactly one endpoint in S.
- Let d_i denote the degree of i.

Note: $\sum_{i \in S} d_i = 2E(S) + \delta(S)$. We wish to find $S \subseteq V$ maximizing $\frac{|E(S)|}{|S|}$; let D^* denote the maximum density and S^* be a set of density D^*.

Lemma 14. *Max flow value of graph described below is mn iff $\gamma \geq D^*$.*

Proof. Suppose $\gamma \geq D^*$. We already know that mn is an upper bound on the size of flow. Computation of capacity of $|S|$ shows that any other cut has capacity $mn + 2|S| \left(\gamma - \frac{|E(S)|}{|S|}\right) \geq mn$. Hence max flow is exactly mn.

Suppose $\gamma < D^*$. Then $s - t$ cut $\{s\} \cup S^*$ has capacity $mn + 2|S| \left(\gamma - \frac{|E(S^*)|}{|S^*|}\right)$. But then $\gamma - \frac{|E(S^*)|}{|S^*|} < 0$, so this cut has strictly lower capacity and max flow is strictly less than mn. □

Theorem 15. *(Goldberg 1984) Densest subgraph can be found in $O(\log(n))$ max flow computations.*

Proof. Idea: Try to guess density of densest subgraph, γ. Run max flow to determine if γ too high or too low. Then use binary search to find actual D^*.

Make graph by adding s, t. Add edges $s \to v$ for all $v \in V$ with capacity m, bidirectional edges with capacity 1 between i and j whenever $\{i, j\} \in E$, and edges $v \to t$ with capacities $m + 2\gamma - d_k$.

Easy upper bound on value of this flow is nm. Note that $\{s\}$ is an $s - t$ cut of capacity mn; capacity of the $s - t$ cut $\{s\} \cup S$ is:

$$\begin{aligned} m|V - S| + |\delta(S)| + \sum_{i \in S}(m + 2\gamma - d_i) &= m(n - |S|) + |\delta(S)| + m|S| + 2\gamma|S| - \sum_{i \in S} d_i \\ &= mn + |\delta(S)| + 2\gamma|S| - (2|E(S)| + |\delta(S)|) \\ &= mn + 2|S| \left(\gamma - \frac{|E(S)|}{|S|}\right) \end{aligned}$$

Rest of proof follows from lemma + binary search. □

Corollary 16. *Let $D' < D^*$ be the second largest density, so that for all $S \subseteq V$ either*

$\frac{|E(S)|}{|S|} \leq D'$ *or* $\frac{|E(S)|}{|S|} = D^*$. *Then if* $D' \leq \gamma < D^*$, *and* $\{s\} \cup X$ *is a min* $s-t$ *cut, then* X *is a maximum densest subgraph.*

Proof. For any optimal S^*, capacity of $s-t$ cut $\{s\} \cup S^* < mn$. For any set S of density $\frac{|E(S)|}{|S|} \leq D' \leq \gamma$, capacity of cut $\{s\} \cup S$ is $\geq mn$. So we will find a min cut. □

But how can we figure out when we have $D' \leq \gamma < D^*$? Denote the set of possible densities $Z = \frac{m'}{n'} : 0 \leq m' \leq m, 1 \leq n' \leq n\}$. Let Δ be the smallest possible positive difference between two elements of Z.

$$\Delta = \frac{m_1}{n_1} - \frac{m_2}{n_2} = \frac{m_1 n_2 - m_2 n_1}{n_1 n_2} \geq \frac{1}{n^2}$$

So if we can identify an interval of size at most $\frac{1}{n^2}$, we can use the above result to produce our algorithm.

Algorithm 2 Densest subgraph

procedure EUCLID(a, b) ▷ The g.c.d. of a and b
  $\ell \leftarrow 0, u \leftarrow m, X \leftarrow \emptyset$
  while $u - \ell \geq \frac{1}{n^2}$ **do** ▷ We have the answer if r is 0
    Compute max flow f
    Find min $s-t$ cut $\{s\} \cup S$ with guess $\gamma = \frac{1}{2}(u + \ell)$
    if $|f| = mn$ **then**
      $u = \frac{1}{2}(u + \ell)$
    else
      $\ell \leftarrow \frac{1}{2}(u + \ell)$
      $X \leftarrow S$
    end if
  end while
  return X ▷ X has maximum density
end procedure

How many calls to max flow? $O\left(\lg \frac{m}{1/n^2}\right) = O(\log mn^2) = O(\log n)$.

3 Lecture 3 (9/9/20)

How do we make the algorithm (originally $O(mU)$ iterations) more efficient? Idea: choose the most improving alternating path, i.e. augmenting path P that maximizes amount of flow we can push (min residual capacity). Want to maximize: $\min_{(i,j)\in P} u_f(i,j) = u(i,j) - f(i,j)$.

Idea: We can modify Dijkstra's alg for shortest paths to find this in $O(m + n \log n)$.

Idea: Sort edges in descending order of residual capacity. Add edges in until $s \rightsquigarrow t$ is connected.

Flow decomposition: Write $f = f' + f''$ (or $f = f' - f''$) if $f(i,j) = f'(i,j) + f''(i,j)$ for all edges (or taking differences). Note that if both f', f'' obey flow conservation and skew symmetry, then f does too; $|f| = |f'| + |f''|$.

Lemma 17. *Flow decomposition: For $s-t$ flow f, $\exists s-t$ flows $f_1, \ldots, f_p$ with $p \leq m$ such that $f = \sum_{i=1}^{p} f_i$ and each f_i has positive flow only on an $s-t$ path or a cycle.*

Proof. (Sketch): Suppose $\exists (i,j)$ with $f(i,j) > 0$. Then if $j \neq t, \exists k$ such that $f(j,k) > 0$ (flow conservation); likewise for $i \neq s, \exists h$ s.t. $f(h,i) > 0$. Repeated extending gives either a cycle C or $s-t$ path P of positive flow. Suppose the latter; then take $\delta = \min_{(i,j)\ in P} f(i,j) > 0$. We have:

$$f_P(i,j) = \begin{cases} \delta & (i,j) \in P \\ -\delta & (j,i) \in P \\ 0 & \text{otherwise} \end{cases}$$

Claim: $f - f_P$ is a flow; not very difficult. Also note that one fewer edge has positive flow. Remainder follows by induction. □

Lemma 18. *Let f be an $s-t$ flow in G, and f^* a maximum flow in G. Then the max flow in G_f has value $|f^*| - |f$.*

Proof. Claim that $f' = f^* - f$ is a feasible flow in G_f. Already know that f' obeys flow conservation, skew-symmetry since f^* and f do as well. So only remains to show that capacity constraints satisfied:

$$f'(i,j) = f^*(i,j) - f(i,j) \leq u(i,j) - f(i,j) = u_f(i,j)$$

Now consider min $s-t$ cut S in G. So $|f^*| = u(\delta^+(S))$, so that $f^*(i,j) = u(i,j)$ for all $(i,j) \in \delta^+(S)$. Then S is also a min $s-t$ cut in G_f, since $f'(i,j) = f^*(i,j) - f(i,j) = u_f(i,j)$ for all $(i,j) \in \delta^+(S)$.

So f' is a max flow in G_f, S is min $s-t$ cut, and $f^* = f' + f$. □

Lemma 19. *Residual capacity of most improving augmenting path is at least $\frac{1}{m}(|f^*| - |f|)$.*

Proof. Value of max flow in G_f is $|f^*| - |f|$. Can decompose max flow in G_f into at most m flows f_i. Largest augmenting path must then have value at least $\frac{1}{m}(|f^*| - |f|)$. □

Theorem 20. *If capacities are integers, then the most improving path alg takes $O(m \ln(mU))$ iterations. In particular, this gives a poly-time algorithm.*

Proof. We had $|f*| \leq mU$ as an upper bound. Let $f^{(k)}$ indicate the flow f after k iterations. Note that $|f^{(1)}| \geq |f| + \frac{1}{m}(|f^*| - |f|)$. Rearranging this gives:

$$\begin{aligned} |f^*| - |f^{(1)}| &\leq (1 - \frac{1}{m})(|f^*| - |f|) \\ &< e^{-1/m}(|f^*| - |f|) \end{aligned}$$

Repeating this gives in general:

$$|f^*| - |f^{(k)}| < e^{-k/m}(|f^*| - |f|)$$

So it suffices to choose $k = m \ln(mU)$ to get:

$$|f^*| - |f^{(k)}| < \frac{1}{mU}(|f^*| - |f|) \leq \frac{1}{mU}(mU - 0) \leq 1$$

By integrality (all of our augmenting paths have integer values), this suffices to show that $f^{(k)}$ is a max flow. □

Instead of looking for the best augmenting path, does it suffice to find one that is "good enough"?

Idea: Δ-scaling parameter. Consider $G_f(\Delta)$ = graph of all arcs with residual capacity $\geq \Delta$. Any $s-t$ path in this graph allows us to push flow at least Δ; if no such path exists, then we must be close to max flow.

Algorithm 3 Δ-scaling

```
1: procedure EUCLID(a, b)
2:     f ← 0; Δ ← 2^⌊lg U⌋
3:     while Δ ≥ 1 do
4:         while ∃ s − t path P in G_f(Δ) do
5:             Push flow on P
6:             Update f
7:         end while
8:         Δ ← Δ/2
9:     end while
10:    Return f
11: end procedure
```

Call a Δ-scaling phase one iteration of outer loop for fixed value of Δ.

Lemma 21. *At the start of Δ-scaling phase, value of flow in $G_f \leq 2m\Delta$.*

Proof. At start, $f = 0, \Delta \geq U/2$, with max flow value $\leq mU$.

At the end of a Δ-scaling phase, no $s-t$ path exists in $G_f(\Delta)$, i.e. $\exists s-t$ cut S s.t. for all $(i,j) \in S$, $u_f(i,j) < \Delta$. This implies $u_f(\delta^+(S)) < m\Delta$.

So at the start of next Δ-scaling phase, $u_f(\delta^+(S)) < 2m\Delta$.

□

Lemma 22. *Each Δ-scaling phase only has at most $2m$ iterations.*

Proof. Each iteration of Δ-scaling phase increases flow value by at least Δ. Max flow in $G_f = |f^*| - |f| \leq 2m\Delta$. If the phase is not over after $2m$ iterations, we have $|f^{(2m+1)}| \geq |f| + (2m+1)\Delta \geq |f^*|$ □

Theorem 23. *If capacities are all integers, then alg takes at most $O(m \ln U)$ iterations.*

Proof. Each inner loop is at most $2m$ iterations; the outer loop has at most $O(\lg U)$ iterations. Again, stop when $\Delta < 1$ by integrality. □

4 Lecture 4 (9/14)

Augmenting path algorithms: Maintain a feasible flow; eventually get $s-t$ cut. Today, look at different type of alg: Infeasible flow at every step, maintaining $s-t$ cut throughout; end up with feasible flow.

Push/relabel alg: Maintain an $s-t$ preflow:

- $f(i,j) \leq u(i,j)$ for all $(i,j) \in A$
- $f(i,j) = -f(j,i)$ for all $(i,j) \in A$
- $\sum_{k:(k,i)\in A} f(k,i) \geq 0$ for all $i \neq s$.

Last condition: can't have more flow coming in than going out. Keep track of excess of preflow f at i:

$$e_f(i) = \sum_{k:(k,i)\in A} f(k,i)$$

If $e_f(i) = 0 \forall i \neq s,t$, then f is actually a flow.

Idea: Push any positive excess to sink on shortest paths to t. Maintain a valid distance labeling $d : V \to \mathbb{Z}$ such that:

- $d(s) = n$
- $d(t) = 0$
- $d(i) \leq d(j) + 1$ for all (i,j) with $u_f(i,j) > 0$.

Idea: If there is a path P in residual graph from i to t, then $d(i) \leq |P|$ (follows basically from applying third condition to all vertices in path).

Lemma 24. *There are no augmenting paths in G_f.*

Proof. Suppose $s-t$ path P exists. Then $d(s) \leq d(t) + P \leq 0 + (n-1) = n-1$. But $d(s) = n$. □

We push excess to nodes that we know are close to sink. So push excess from i to j only if $d(i) = d(j) + 1$. What happens if $d(i) \leq d(j) \forall (i,j) : u_f(i,j) > 0$? Infer that the distance $d(i)$ is too small, and relabel i.

`Push(i,j)`:

$\delta \leftarrow \min(u_f(i,j), e_f(i))$

$f(i,j) \leftarrow f(i,j) + \delta, f(j,i) = f(j,i) - \delta$

`Relabel(i)`:

$d(i) \leftarrow \min\{d(j) + 1 : u_f(i,j) > 0\}$

Algorithm 4 Push-relabel (Goldberg, Tarjan 1988)

procedure PUSHRELABEL(G)
  $f \leftarrow 0$
  $f(s,j) \leftarrow u(s,j), f(j,s) \leftarrow -u(s,j), \forall (s,j) \in A$
  $d(s) \leftarrow n, d(i) \leftarrow 0, \forall i \neq s$
  while $\exists$ active i (i.e. $e_f(i) > 0, i \neq s,t$ **do**
    if $\exists j : u_f(i,j) > 0$ and $d(i) = d(j) + 1$ **then** `Push(i,j)`
    else `Relabel(i)`
    end if
  end while
  Return f
end procedure

Lemma 25. *The algorithm maintains a preflow.*

Proof. (Sketch): Initially, if $(s,j) \in A$, then $e_f(j) = u(s,j) \geq 0$, $e_f(j) = 0$ for all other j.

If we push δ on (i,j), then:

$$e_{f'}(i) = e_f(i) - \delta \geq 0, e_{f'}(j) = e_f(j) + \delta \geq 0$$

□

Lemma 26. *The algorithm maintains a valid distance labelling.*

Proof. Initially, this is trivial for all (i,j) except for (s,j) since $d(s) = n, d(j) = 0$. No issue because no positive residual capacity on the arcs out of s.

After relabel: Maintains labelling by construction.

After push: possible to have $u_f(j,i) = 0$ before pushing on (i,j), but $u_f(j,i) > 0$ afterwards. But by virtue of choosing this arc, we needed $d(i) = d(j) + 1 \implies d(j) = d(i) - 1 \leq d(i) + 1$. □

Theorem 27. *If the algorithm terminates, then f is a max flow.*

Proof. If it terminates, f is a flow. But since no augmenting path exists, this must be a max flow. □

It remains to show that (1) this algorithm does terminate, and (2) it does so in poly-time.

Lemma 28. *If $e_f(i) > 0$ for $i \neq t$, then $\exists$ a path $i \rightsquigarrow s$ on arcs of positive residual capacity.*

Proof. Let S be the nodes reachable from i via arcs of positive residual capacity. Assume $s \notin S$. Then for any $(j,k) \in \delta^+(S)$, must have $f(j,k) = u(j,k)$. Now we have $f(k,j) = -f(j,k) = -u(j,k) \leq 0$. Consider:

$$\sum_{j \in S} e_f(j) = \sum_{j \in S} \sum_{k:(k,j) \in A} f(k,j) = \sum_{j \in S} \left(\sum_{k \in S:(j,k) \in A} f(k,j) + \sum_{k \notin S:(j,k) \in A} f(k,j) \right)$$

But the first term cancels by skew-symmetry, and we have:

$$\sum_{j \in S, k \notin S:(k,j) \in A} f(k,j) \leq 0$$

But f is a preflow, so $e_f(j) \geq 0$ for all $j \neq s$. In particular, if $s \notin S$, we have $e_f(j) = 0$ for all $j \in S$; but $i \in S$, violating our initial assumption that $e_f(i) > 0$.

□

Lemma 29. *For all $i \in V$, $d(i) \leq 2n - 1$, i.e. the distance labels never get "too big."*

Proof. We only relabel i if $e_f(i) > 0$, i.e. $\exists$ path P of arcs of positive residual capacity. Then :

$$d(i) \leq d(s) + |P| \leq n + (n-1) = 2n - 1$$

□

Lemma 30. *The push-relabel algorithm runs at most $2n^2$ executions of relabel.*

Proof. For $i \neq s$, we always have $d(i) \leq 2n - 1$, and distance labels never decrease. Each relabel increases $d(i)$ by at least 1, so at most $2n^2$ relabels.

For pushes, we consider whether the push is saturating.

A Push operation is saturating with $\delta = u_f(i,j)$ if after `push(i,j)`, $f(i,j) = u(i,j)$; nonsaturating otherwise (then $\delta = e_f(i)$).

We claim that there are at most mn saturating pushes. To see this, pick $(i,j) \in A$. To push from i to j, need $d(i) = d(j) + 1$. Then to do another saturating push on (i,j), we need to have pushed from j to i. this requires:

$$d'(j) = d'(i) + 1 \geq d(i) + 1 = (d(j) + 1) + 1 = d(j) + 2$$

Hence there are at most n saturating pushes from i to j.

Consider now the number of nonsaturating pushes. We claim that there are at most $4n^2m$ nonsaturating pushes. We use a potential function + amortized analysis to measure this.

$$\Phi = \sum_{\text{active } i} d(i)$$

Initially, $\Phi = 0$. Claim: Nonsaturating push on (i, j) makes Φ go down by at least 1. Such a push makes i inactive; it may make j active, but $d(i) = d(j) + 1$ in this case. So such a push always decreases Φ.

When can Φ increase?

- Saturating push on (i, j) can add $d(j) \leq 2n - 1$ for each saturating push.
- Relabel increase by 1

Total increase in Φ is bounded by:

$$\Phi \leq nm(2n - 1) + 2n^2 \leq 4n^2m \text{ if } m \geq n$$

The second term corresponds to an upper bound on the sum of distance labels.

□

5 Lecture 5 (9/16)

Theorem 31. *Push/relabel takes $O(n^2m)$ push/relabel operations and $O(n^2m)$ time.*

Proof. Sketch: Each operation takes average constant time. Time taken per relabel on i is just outdegree of i. □

Note: This is our first strongly polynomial-time alg for max flow.

We can improve this by being a bit more careful when we choose our paths.

Highest label $d^* = \max_{\text{active } i} d_i$; do pushes only on i such that $d(i) = d^*$.

Lemma 32. *Highest label algorithm has $O(\sqrt{m}n^2)$ nonsaturating pushes.*

Proof. Let K be a parameter (eventually we take $K = \sqrt{m}$). Take $N(i) = \{j \in V : d(j) \leq d(i)\}$ and the potential function:

$$\Phi = \frac{1}{K} \sum_{\text{active } i} |N(i)|$$

Initially $\Phi \leq n^2/K$ (could be up to n active nodes all with label 0, all $|N(i)| \leq n$). Increases in Φ arise from:

- Relabel i can increase Φ by as much as n/K since $|N(i)|$ can go up by at most n.
- Saturating push on (i, j) can increase Φ by as much as n/K by making an inactive node active.

Consider phases, i.e. periods where d^* remains constant. How many phases occur?

Phase ends at d^* increase, which occurs due to relabel of i. Increase in d^* is at most change in i. Total amount of increase in $d^* \leq$ total change in distance labels which is $O(n^2)$.

Total number of times we can decrease $d^* \leq$ total amount we can increase $d^* = O(n^2)$. So number of phases is also $O(n^2)$. Call a phase short if $< K$ nonsaturating pushes; long otherwise. So number of nonsaturating pushes from short phases is then $O(n^2K)$.

Suppose $\geq K$ nonsaturating pushes, i.e. we are in a long phase. But then there must have been $\geq K$ nodes with $d(i) = d^*$, so Φ must bedcrease by $(|N(i)| - |N(j)|)/K \geq 1$ for each nonsaturating push.

So total number of nonsaturating pushes in long pushes is at most:

$$\text{Initial } \Phi + \text{ total increase in } \Phi = O\left(\frac{n^2}{K} + \frac{mn^2}{K}\right) = O\left(\frac{mn^2}{K}\right)$$

Total number of nonsaturating pushes is then $O\left(n^2K + \frac{mn^2}{K}\right) = O(n^2\sqrt{m})$ when $K = \sqrt{m}$.

□

Lemma 33. *Let f be an $s-t$ preflow, S an $s-t$ cut, $e_f(j) = 0 \forall j \notin S, j \neq t$, and $u_f(i,j) = 0 \forall (i,j) \in \delta^+(S)$. Then S is a min $s-t$ cut.*

Proof. (Sketch). Suppose we keep running push-relabel at this stage.

- Can we push on $(i,j) \in \delta^+(S)$? No; this edge is saturated.
- Can we push on (j,i)? No $e_f(j) = 0$ (j is saturated).

These properties will be maintaned throughout execution of the algorithm; in the end, $u_f(i,j) = 0$ for all $(i,j) \in \delta^+(S)$ at end of alg. But by earlier lemma, since f will be a flow at end of alg, S must be a min $s-t$ cut. □

So we can use this as another termination condition.

Lemma 34. *We can convert this preflow f to a flow in $O(mn)$ time. (Pf. omitted)*

How can we check when such a state is reached? We modify the definition of active nodes. Call i active only if $e_f(i) > 0$ and $d(i) < n$; stop if no active nodes. Let S be all nodes that cannot reach t via arcs of positive residual capacity; then S is a min $s-t$ cut. This is despite the fact that f is not a flow yet (can still have positive excess on nodes).

Why? By definition, $(i,j) \in \delta^+(S) \implies u_f(i,j) = 0$. Now suppose that $e_f(i) > 0$; since we terminated, must have $d(i) \geq n$. Claim that $i \in S$. Otherwise, there is a path P from i to t on arcs of positive residual capacity; but the max label for any such path is $n-1$.

6 Lecture 6 (9/21)

We're interested in global min-cut problem. Input: directed graph $G = (V, A)$, capacities $u(i,j) \geq 0 \forall (i,j) \in A$. Goal: Find $S \subseteq V, S \neq \emptyset$ that minimizes $u(\delta^+(S))$.

Naive alg: For optimal cut S^*, there must be some $s \in S^*$ and $t \notin S^*$. Can try all $n(n-1)$ possibilities for (s,t) and find min $s-t$ cut for each.

We define a min s-cut to be a minimum global cut such that $s \in S$. Claim: we can find a min s-cut with $n-1$ min $s-t$ cut computations. Just try all other possible vertices as t; one of them must work.

Claim: We can solve the global min-cut problem in only 2 min s-cut computations (and hence $2(n-1)$ min $s-t$ cut computations).

Idea: pick some $s \in V$. Either it's in S^* or it's not.

1. Find min s-cut S.
2. Reverse all arcs in G, run min s-cut computation.
3. Return smaller of 2.

The second computation finds S' with $s \in S'$ that minimizes $u(\delta^-(S)) = u(\delta^+(V-S))$.

Claim: If we can find a min $X-t$ cut (i.e. cut S such that $X \subseteq S, t \notin S$) for any $X-t$, then we ca nfind a min s-cut.

Algorithm 5 Min s-cut

```
1: procedure MinSCut(G)
2:     X ← {s}
3:     while X ≠ V do
4:         Pick t ∈ V − X
5:         Compute min X − t cut S_t
6:         X ← X ∪ {t}
7:     end while
8:     Return smallest S_t
9: end procedure
```

Let S^* be a min s-cut, and let $t*$ be the first t such that $t \notin S^*$. In this iteration, $X \subseteq S^*$ and $t \notin S^*$. So S^* is a min $s-t$ cut and the algorithm must find an s-cut of the same capacity.

We define an X-preflow to be like a preflow, but can have $e_f(i) < 0$ for $i \in X$.

Take an X-valid distance labeling for preflow X to be d s.t.:

- $d(j) = n$ for all $j \in X$
- $d(i) \leq d(j) + 1$ for all $(i,j) : u_f(i,j) > 0$
- $d(t) \leq |X| - 1$

Lemma 35. *For X-preflow f with X-valid distance labels d, there is no path in G_f from any $i \in X$ to t on arcs of positive residual capacity.*

Proof. Suppose otherwise; WLOG P starts from i, the only node of X in P. Then $d(i) \leq d(t) + |P| \leq |X| - 1 + n - |X|$, same contradiction as previously. □

Lemma 36. *Key Lemma: Suppose we have X-preflow f, S an $X - t$ cut s.t. $\forall (i,j) \in \delta^+(S), u_f(i,j) = 0$ and $e_f(i) = 0, \forall i \notin S, i \neq t$. Then S is a min $X - t$ cut.*

Proof is again along the same lines as for $s - t$ cut.

We define distance level k to be $B(k) = \{i \in V : d(i) = k\}$. It is empty if $B(k) = \emptyset$. It is a cut level if $\forall i \in B(k), \forall (i,j) : u_f(i,j) > 0$, we have $d(i) \leq d(j)$.

Trivially, any empty distance level is also a cut level.

Main idea of algorithm: Look at sets $S(k) = \{i \in V : d(i) \geq k\}$ and find a set $S(k)$ such that the Key Lemma applies. Then $S(k)$ is a min $X - t$ cut.

Lemma 37. *If k is a cut level, then $\forall (i,j) \in \delta^+(S(k))$, $u_f(i,j) = 0$.*

Proof. By definition of $S(k)$, for all $(i,j) \in \delta^+(S(k))$ satisfies $d(i) \geq k$ and $d(j) < k$. We have two cases:

- $d(i) = k$. Then since k is a cut level and $d(i) > d(j)$, it must be the case that $u_f(i,j) = 0$.
- $d(i) > k$. Then since d is X-valid, must have $d(i) \leq d(j) + 1$. But then $d(j) \geq k$, which means that $j \in S(k)$, a contradiction.

□

As a corollary, if k is a cut level for $d(t) < k \leq n$ and $e_f(i) = 0, \forall i \notin S(k), i \neq t$ then $S(k)$ is a min $X - t$ cut.

We can modify push-relabel framework to solve for $X - t$ cuts in only 1 iteration.

Note that the looping doesn't have multiple executions of push-relabel; rather, it picks up where it left off at the previous iteration. To analyze runtime, we prove a few ideas:

Lemma 38. *The non-empty distance levels are consecutive (not including distance level n).*

Proof. Starts off true since all nodes other than s have $d(i) = 0$. Suppose distance k becomes empty. Let i be the last node in $B(k)$. Then either i was relabeled, or i was the sink and added to X. The former is impossible since we never relabel the last vertex in a distance level. The latter does not create non-consecutive gap because we chose sink to be of minimal distance label.

Other potential issue: We added node to empty $B(k)$ from relabeling. But in this case, it must receive this label from some $j \in B(k-1)$, so no non-consecutive levels arise. □

A brief sketch of the rest of proof (the proofs are omitted for time):

Algorithm 6 Hao-Orlin (1994)

procedure HAOORLIN(G)
  $X \leftarrow \{s\}$
  pick $t \in V - X, f \leftarrow 0$
  $d(s) \leftarrow n, d(i) \leftarrow 0, \forall i \neq s$
  Saturate all arcs out of s
  $\ell \leftarrow n - 1$ ▷ ℓ is a cut level
  while $X \neq V$ **do**
    Run push/relabel except only select i for pushes/relabels if $d(i) < \ell$
    If you want to relabel i and $|B(d(i))| = 1$, then don't relabel and set $\ell \leftarrow d(i)$
    If relabel of i makes $d(i) \geq \ell$, then $\ell \leftarrow n - 1$.
    $S_t \leftarrow S(\ell)$, $X \leftarrow X \cup \{t\}$, $d(t) \leftarrow n$
    Saturate all arcs out of t
    Pick some $t \in V - X$ of minimal distance label
    if $d(t) \geq \ell$ **then** $\ell \leftarrow n - 1$
    end if
  end while
  Return smallest S_t
end procedure

Lemma 39. *We always have $d(t) \leq |X| - 1$.*

Lemma 40. *If $i \notin X, d(i) \leq n - 2$.*

Proof. Follows from the nonempty distance levels being consecutive. □

Lemma 41. *Throughout execution of the algorithm, ℓ is a cut level.*

Lemma 42. *Number of relabels is $O(n^2)$. Number of saturating pushes is $O(mn)$. Number of nonsaturating pushes is $O(n^2m)$. So overall runtime is same as push-relabel.*

So we can find a min s-cut in $O(n^2m)$ time. If we use fancier data structures, can get this down to $O(nm \log(n^2/m))$ time.

7 Lecture 7 (9/23)

Global min cuts in undirected graphs. Input $G = (V, E)$ undirected, capacities $u(i, j) \geq 0, \forall (i, j) \in E$.

Goal: Find $S \subseteq V, S \neq \emptyset$ minimizing $u(\delta(S)) = \sum_{(i,j)\in\delta(S)} u(i, j)$, where $\delta(S)$ is set of edges with exactly one endpoint in S.

We can do this naively by first just making every undirected edge a bidirectional edge with same capacity both directions.

Claim: Can solve in $n-1$ min $s-t$ cut comps. Capacity of S is same as $V-S$, so we can just pick any source and try all other vertices as sinks.

Now define $\delta(A,B) = \{(i,j) \in E : i \in A, j \in B\}$, $\delta(A,v)$ likewise.

Take the MA ordering ("max adjacency"):

procedure MAOrdering(G)
    Pick $v_1 \in V$
    $W_1 \leftarrow \{v_1\}, k \leftarrow 1$
    while $k < n$ **do**
        Choose $v_{k+1} \in V - W_k$ to maximize $u(\delta(W_k, v_{k+1}))$
        $W_{k+1} \leftarrow W_k \cup \{v_{k+1}\}$
        $k \leftarrow k+1$
    end while
end procedure

Claim: This alg can be implemented in $O(m+n\log n)$ time (similar to Dijkstra's when using fib heap).

Lemma 43. *For MA ordering $v_1, \ldots, v_n$, $\{v_n\}$ is a min $v_{n-1} - v_n$ cut.*

Proof skipped while we see why this lemma is useful. Either there is some global min cut S^* that is a $v_n - v_{n-1}$ cut or not. If so, then $u(\delta(v_n)) \leq u(\delta(S^*))$ (since it's a $v_n - v_{n-1}$ cut) and $u(\delta(S^*)) \leq u(\delta(v_n))$ (since S^* is a global min cut); then $\{v_n\}$ is a global min cut. If not, then $v_{n-1}, v_n \in S^*$ or in $V - S^*$ for any global min-cut S^*. So contract v_n, v_{n-1} into one vertex v', with $u(v', w) = u(v_n, w) + u(v_{n-1}, w)$. This preserves the capacity of all global min-cuts.

These two cases give us enough to find a min cut: Either $\{v_n\}$ is one and we are finished; otherwise it isn't, so we contract and run again.

Algorithm 7 MA ordering min-cut

procedure MAOrderMinCut(G)
    `val` $\leftarrow \infty, S^* \leftarrow \emptyset, \ell \leftarrow n$
    while $\ell > 1$ **do**
        Compute MA ordering $v_1, v_2, \ldots, v_\ell$
        if $u(\delta(\{v_\ell\})) <$ `val` **then**
            `val` $\leftarrow u(\delta(v_\ell))$
            $S^* \leftarrow$ all nodes contracted into v_ℓ
        end if
        Contract $v_{\ell-1}, v_\ell$; update capacities; $\ell \leftarrow \ell - 1$
    end while
    Return S^*
end procedure

Theorem 44. *The algorithm finds a global mincut in $O(n(m + n\log n))$ time.*

Proof. (PROOF OF MAIN LEMMA): Let C be a min $v_{n-1}-v_n$ cut. Take $W_k = \{v_1, \ldots, v_k\}$, $E_k = E(W_k)$, i.e. all edges with both endpoints in W_k. Say u is separated from v if $u \in C, v \notin C$ or vice versa.

We claim that if v_{k-1} is separated from v_k, then $u(\delta(W_{k-1}, v_k)) \leq u(\delta(C) \cap E_k)$. When $k = n$, this gives $u(\delta(v_n)) = u(\delta(W_{n-1}, v_n)) \leq u(\delta(C) \cap E_n) = u(\delta(C))$. But $u(\delta(v_n))$ is a $v_n - v_{n-1}$ cut, so this implies equality; then $\{v_n\}$ is itself a min $v_n - v_{n-1}$ cut.

Prove this claim by induction. BC: let k be minimal s.t. v_k is separated from v_{k-1}. WLOG, $v_k \notin C$ (and all predecessors in C).

$$u(\delta(W_{k-1}, v_k)) = u(\delta(C) \cap E_k)$$

Now suppose that this holds for all v_j separated from v_{j-1} for $j < k$; and that v_k is separated from v_{k-1}. Pick $j < k$ to be maximal such that v_j is separated from v_{j-1}.

Edges from v_k to W_{k-1} either go to W_{j-1} or to vertices in $W_{k-1} - W_{j-1}$.

$$u(\delta(W_{k-1}, v_k)) = u(\delta(W_{j-1}, v_k)) + u(\delta(W_{k-1} - W_{j-1}, v_k))$$

The first term is at most $u(\delta(W_{j-1}, v_j))$ since we picked v_j to minimize this quantity in the alg. For the latter term, recall that we chose v_j to be such that everything after it is on the same side of C.

$$\begin{aligned} &\leq u(\delta(W_{j-1}, v_j)) + u(\delta(C) \cap (E_k - E_j)) \\ &\leq u(\delta(C) \cap E_j) + u(\delta(C) \cap (E_k - E_j)) \\ &= u(\delta(C) \cap E_k) \end{aligned}$$

□

Note: This algorithm may be generalized to find minimizers of symmetric submodular functions.

8 Lecture 8 (9/28)

Here we see one of the most straightforward applications of randomization:

Algorithm 8 Random Contraction algorithm (David Karger 1993)

```
procedure RandomContraction(G)
    while |V| > 2 do
        Pick edge (i, j) at random with prob ∝ u(i, j)
        Contract (i, j), update capacities
    end while
    Return S*
end procedure
```

Let S^* be a global min cut with $\lambda^* = u(\delta(S^*))$; Take $W = \sum_{(i,j)\in E} u(i,j)$. Pick edge (i,j) with prob $u(i,j)/W$.

We say that S^* survives a contradiction if chosen edge $(i,j) \notin \delta(S^*)$. Probability S^* survives first contraction is $1 - \frac{\lambda^*}{W}$. Since $u(\delta(i)) \geq \lambda^*$ for all $i \in V$ and $W = \frac{1}{2}\sum_{i\in V} u(\delta(i))$, we have $W \geq \frac{n}{2}\lambda^*$. It follows that the probability S^* survives first contraction is:

$$1 - \frac{\lambda^*}{W} \geq 1 - \frac{\lambda^*}{n\lambda^*/2} = 1 - \frac{2}{n}$$

Let W_k be the random variable corresponding to the total capacity of the graph after k contractions. We have $W_k \geq (n-k)\lambda^*/2$, so the probability S^* survives the k^{th} contraction given that it survived the first $k-1$:

$$1 - \frac{\lambda^*}{W_{k-1}} \geq 1 - \frac{2}{n-k+1}$$

Probability of S^* surviving all contractions:

$$\geq \prod_{k=1}^{n-2}\left(1 - \frac{2}{n-k+1}\right) = \prod_{k=1}^{n-2}\frac{n-k-1}{n-k+1} = \prod_{\ell=3}^{n}\frac{\ell-2}{\ell} = \frac{1}{\binom{n}{2}}$$

Claim: We can implement Random Contraction alg in $O(n^2)$ time.

Theorem 45. *We can find a global min cut with high probability in $O(n^4 \ln n)$ time.*

Proof. Run alg $k = c\binom{n}{2}\ln n$ times. The probability that S^* is never returned is:

$$\left(1 - \frac{1}{\binom{n}{2}}\right)^{c\binom{n}{2}\ln n} \leq e^{-c\ln n} = \frac{1}{n^c}$$

by independence of runs. The running time is $n^2 k = O(n^4 \ln n)$. □

Theorem 46. *(Karger, Stein 1996) Can find S^* with probability $\Omega(1/\log n)$ in $O(n^2 \log n)$ time. Then repeated application allows for finding S^* w.h.p. in $O(n^2 \log^3 n)$ time.*

Define $p_n \equiv 1 - \frac{2}{n}$. Consider the following algs:

`Alg1`

procedure RandomContraction(G)
  Pick k with prob $p_n(1-p_n)^{k-1}$
  for $n \leftarrow 1$ to k **do**
    $G_i \leftarrow$ result of contracting one edge
    $S_i \leftarrow$ `Alg1`(G_i)
  end for
  Return smallest S_i found
end procedure

`Alg2`

procedure RandomContraction(G)
  Pick $r \in [0,1]$ uniformly at random
  Let G' be the result of random contraction
  $S_1 \leftarrow$ `Alg2`(G')
  if $r \leq p_n$ **then**
    Return S_1
  else
    $S_2 \leftarrow$ `Alg2`(G)
    Return smaller of S_1, S_2.
  end if
end procedure

Lemma 47. *Expected number of recursive calls in `Alg1` in which S^* survives is at least 1.*

Proof. For geometric distribution, $\sum_{k=1}^{\infty} k p_n (1-p_n)^{k-1} = \frac{1}{p_n}$.

Each call prob of surviving is at least p_n. So expected number of calls that S^* survives is at least:

$$\sum_{k=1}^{\infty} (k p_n) p_n (1-p_n)^{k-1} = p_n \sum_{k=1}^{\infty} k p_n (1-p_n)^{k-1} = 1$$

□

"Goldilocks" value of 1: too small means we won't find the cut; too large means that recursive tree will blow up exponentially.

Lemma 48. *Probability that S^* is returned by alg is at least $\frac{1}{2H_n - 2}$.*

Proof. Let $P(n)$ be the probability that S^* is returned by alg on a graph with n nodes. We also assume that the prob of no edge in S^* being contracted on graph with k nodes is exactly

p_k (to simplify analysis). Note then that $P(n)$ is a lower bound on the actual probability. Clearly $P(2) = 1$.

$$P(n) = \underbrace{p_n}_{\text{prob. of 1 call}} \cdot \underbrace{p_n}_{\text{prob } S^* \text{ survives}} \cdot P(n-1) + \underbrace{(1-p_n)}_{\text{prob of 2 calls}} (1 - \underbrace{(1-P(n))}_{\text{prob. 2nd call fail}} \underbrace{(1-p_nP(n-1))}_{\text{prob 1st call fail}})$$

Which we can simplify to:

$$p_nP(n) = p_nP(n-1) - p_n(1-p_n)P(n-1)P(n)$$

Divide by $P(n)P(n-1)$:

$$\begin{aligned}
\frac{1}{P(n-1)} &= \frac{1}{P(n)} - \frac{2}{n} \\
\frac{1}{P(n)} &= \frac{1}{P(n-1)} + \frac{2}{n} \\
&= 2H_n - 3 + \frac{1}{P(2)} \\
&= 2H_n - 2
\end{aligned}$$

□

Lemma 49. *Expected running time is $O(n^2 \log n)$.*

Proof. Let $T(n)$ be the expected running time on n nodes. Then we have the recursion:

$$\begin{aligned}
T(n) &= T(n-1) + (1-p_n)T(n) + O(n) \\
p_nT(n) &= T(n-1) + O(n)
\end{aligned}$$

Substitute in $p_n = \frac{n-2}{n}$

$$\begin{aligned}
\frac{1}{n}T(n) &= \frac{1}{n-2}T(n-1) + O\left(\frac{n}{n-2}\right) \\
\frac{1}{n(n-1)}T(n) &= \frac{1}{(n-1)(n-2)}T(n-1) + O\left(\frac{n}{(n-1)(n-2)}\right)
\end{aligned}$$

Use $R(n) = \frac{1}{n(n-1)}T(n)$:

$$\begin{aligned}
R(n) &= R(n-1) + O\left(\frac{1}{n}\right) \\
&= O(H_n) = O(\log n)
\end{aligned}$$

It follows that $T(n) = O(n^2 \log n)$. □

State-of-the-art algorithms:

Korger ('96) came up with $O(m \log^n)$ randomized algorithm.

Kawarabayashi, Thorup (STOC '15) has $O(m \log^{12} n)$ deterministic for unit capacity graphs.

Henzinger, Rao, Wang (SODA '17) has $O(m \log^2 n \log\log^2 n)$.

Li, Panigrahi (FOCS '20) has $O(m^{1+\varepsilon} + (\log^{O(1/\varepsilon^4)} n)MF)$ (max flow comps) for general capacity.

Next time: keep moving towards global min-cut.

9 Lecture 9 (9/30)

Gomory-Hu trees: Give representation of ALL min $s-t$ cuts in an undirected graph (sometimes called cut-equivalent tree).

Given $G = (V, E)$ undirected, $u(i, j) \geq 0, \forall (i, j) \in E$. A Gomory-Hu tree is a spanning tree of G with labels $\ell(e)$ for all $e \in T$. Let $S(e)$ denote the cut in T when e is removed. For all $s, t \in V$, let P be the unique $s-t$ path in T., and let $e \in T$ be the edge of minimum label $\ell(e)$. Then $\ell(e) = u(\delta(S(e)))$ is the capacity of min $s-t$ cut, and $S(e)$ is a min $s-t$ cut.

Lemma 50. *Suppose for all edges $e = (i, j) \in T$, $\ell(e) = u(\delta(S(e)))$ and $S(e)$ is a min $i-j$ cut. Then T is a Gomory-Hu tree.*

Proof. Suppose $\ell(e) = u(\delta(S(e)))$ for all $e = (i, j) \in T$. Pick any $s, t \in V$ and let $s = v_1, v_2, \ldots, v_k = t$ be the unique $s-t$ path in T. Let $e_i = (v_i, v_{i+1})$. Define $c(p, q)$ to be the capacity of minimum $p-q$ cut in G (symmetric so $c(p, q) = c(q, p)$).

We now show that $c(s, t) \leq \min_{i=1,\ldots,k-1} c(v_i, v_{i-1})$; showing the reverse inequality gives equality. The former holds since each edge $e_i = (v_i, v_{i+1})$ has $S(e_i)$ an $s-t$ cut. For the other direction, let S^* be a min $s-t$ cut; suppose that $s \in S^*$. Some For some i, we have $v_i \in S^*, v_{i+1} \notin S^*$. Then S^* is a $v_i - v_{i+1}$ cut and is minimal, so $c(v_i, v_{i+1}) \leq c(s, t)$.

□

How do we construct Gomory-Hu trees? First think about some ideas for min-cuts. Will use the fact that $u(\delta(S))$ is symmetric, submodular (both proven via counting arguments):

Submodular:

$$u(\delta(S)) + u(\delta(T)) \geq u(\delta(S \cup T)) + u(\delta(S \cap T))$$

Symmetric submodular:

$$u(\delta(S)) + u(\delta(T)) \geq u(\delta(S - T)) + u(\delta(T - S))$$

Lemma 51. $\forall r, s, t \in V$, $c(s, t) \geq \min(c(r, s), c(r, t))$

Proof. Let S be a min $s-t$ cut, $s \in S, t \notin S$. If $r \in S$, then S is an $r-t$ cut so $c(r,t) \leq c(s,t)$. Otherwise, $c(r,s) \leq c(s,t)$ (same argument). □

Corollary 52. *The min of $c(r,s)$, $c(s,t)$, and $c(r,t)$ is unique.*

Proof. WLOG, $c(s,t)$ is unique min; then above lemma does not hold. □

Key Lemma:

Lemma 53. *If R is a min $r-s$ cut with $r \in R$, and S a min $s-t$ cut with $s \in S$, assuming $t \notin R$:*

(1) If $r \in S$, then $R \cap S$ is a min $r-s$ cut, $R \cup S$ is a min $s-t$ cut.

(2) If $r \notin S$, then $R - S$ is a min $r-s$ cut, $S - R$ is a min $s-t$ cut, and $c(r,t) = c(r,s)$.

Proof. By submodularity and symmetric submodularity:

$$c(r,s) + c(s,t) = u(\delta(R)) + u(\delta(S)) \geq u(\delta(S \cup R)) + u(\delta(R \cap S))$$

$$c(r,s) + c(s,t) = u(\delta(R)) + u(\delta(S)) \geq u(\delta(S - R)) + u(\delta(R - S))$$

In case (1), we know $R \cap S$ is an $r-s$ cut. We also know $R \cup S$ is an $s-t$ cut. So the inequality implies that each of these are minimal cuts.

For (2), $R - S$ is an $r-s$ cut and $S - R$ is an $s-t$ cut; by similar argument as above, they are both min-cuts. Also note that R is an $r-t$ cut so $c(r,t) \leq c(r,s)$. S is an $s-r$ cut so $c(r,s) \leq c(s,t)$. This implies equality since min not unique (from corollary).

□

We introduce the algorithm; the general idea is to maintain a partition of the vertices and update as we go.

Lemma 54. *At the end of each iteration, for each $e = (r_i, r_j) \in T$, $S(e)$ is am in $r_i - r_j$ cut and $\ell(e) = c(r_i, r_j)$.*

Proof. Initially, r and t are the only r_i and r_j; basically follows directly from alg.

Suppose it holds at end of previous iteration. Algorithm next picks $V_i, t \in V_i, t \neq r_i$ and X a min $r_i - t$ cut with $r_i \in X$.

Two possibilities: $r_j \in X$ or $r_j \notin X$. We'll apply the Key Lemma with $s = r_i, t = t, r = r_j$; $R = S(e), S = X$.

Case 1: $r_j \in X$. Then $S(e) \cup X$ is a min $r_i - t$ cut.

Case 2: $r_j \notin X$. Then $X - S(e)$ is a min $r_i - t$ cut and $c(r_j, t) = c(r_j, r_i)$.

□

Algorithm 9 ????

procedure GomoryHu(G)
  $\mathcal{P} \leftarrow \{V\}; T \leftarrow \emptyset$
  Pick $r \in V$ as representative of V
  while $\exists V_i \in \mathcal{P}$ s.t. $|V_i| \geq 2$: **do**
    Pick $t \in V_i$, $t \neq r_i$ (representative of V_i)
    Compute min $r_i - t$ cut X with $r_i \in X$
    for $(r_i, r_j) \in T$ **do**
      if $r_j \notin X$ **then** replace (r_i, r_j) with (r_j, t)
      end if
    end for
    $\mathcal{P} \leftarrow \mathcal{P} - \{V_i\} \cup \{V_i \cap X\} \cup \{V_i - X\}$
    r_i is representative of $V_i \cap X$ and t is representative of $V_i - X$
    Add (r_i, t) to T with label $u(\delta(X))$
  end while
  Return S^*
end procedure

10 Lecture 10 (10/5)

The augmenting path algorithms we've seen before push flow on the shortest augmenting path. New idea: Send flow simultaneously on all shortest augmenting paths. Say a flow f is blocking if for each $s-t$ path of arcs with positive capacity, there is some $(i,j) \in P$ that is saturated $f(i,j) = u(i,j)$. Note: blocking flows are not necessarily max flows (augmenting paths can exist because residual graph adds in backwards edges).

For flow f, compute shortest path distance $d(i)$ from i to t on positive residual arcs. Observe that if (i,j) is on a shortest $s-t$ path, then $d(i) = d(j) + 1$. Additionally, it is always true that $d(i) \leq d(j) + 1$ if $u_f(i,j) > 0$. Let $\hat{A} = \{(i,j) : d(i) = d(j) + 1, u_f(i,j) > 0\}$ be the admissible arcs (residual capacity > 0 and on shortest $s-t$ path). Send flow on all shortest paths in $G_f \Rightarrow$ Compute a blocking flow in $\hat{A}$. We show that each time this happens, $d(s)$ increases by at least one.

Claim: Algorithm maintains a flow f. (Sketch): Flow conservation and skew-symmetry follow from f, f' being flows; capacity constraints from $\hat{u}$.

Lemma 55. *Distance $d(s)$ increases by at least one in each iteration.*

Proof. Let f, d be flow and distance labels in one iteration of the alg; f', d' in the next iteration. Let P be a shortest augmenting path in $G_{f'}$. Want to show that $d'(s) = |P| > d(s)$. We show:

(1) $d(i) \leq d(j) + 1, \forall (i,j) \in P$.

(2) $d(i) \leq d(j)$ for some $(i,j) \in P$.

Algorithm 10 Dinitz's algorithm

procedure DINITZ(G)
  $f \leftarrow 0$
  while $\exists$ augmenting path in G_f: **do**
    Compute distances $d(i)$ using arcs of positive residual cap.
    $\hat{A} \leftarrow \{(i,j) : d(i) = d(j) + 1, u_f(i,j) > 0\}$
    $\hat{u}(i,j) \leftarrow \begin{cases} u_f(i,j) & \text{if } (i,j) \in \hat{A} \\ 0 & \text{otherwise} \end{cases}$
    Compute blocking flow $\hat{f}$ in G with capacities $\hat{u}$
    $f \leftarrow f + \hat{f}$
  end while
  Return f
end procedure

If these hold, then:

$$d'(s) = |P| - \sum_{(i,j)\in P} 1 > \sum_{(i,j)\in P} (d(i) - d(j)) = d(s) - d(t) = d(s)$$

(1): Note that $(i,j) \in P$ means that $u_{f'}(i,j) > 0$. Then either it had positive residual capacity before: $u_f(i,j) > 0 \implies d(i) \leq d(j) + 1$. Otherwise $u_f(i,j) = 0$ and flow was pushed backwards. But then (j,i) must be admissible, so $d(j) = d(i) + 1$ and we are done.

(2): Claim that impossible to have $\forall (i,j) \in P$, $d(i) = d(j) + 1$ and $u_f(i,j) > 0$. This would imply that $P \subseteq \hat{A}$; by properties of blocking flow, some arc (i,j) must have been saturated. so $\exists (i,j) \in P$ such that either: $d(i) \leq d(j)$ or $u_f(i,j) = 0$. First condition proves the claim; second condition implies that we must have pushed flow back through (j,i) since $u_{f'}(i,j) > 0$. So $d(j) = d(i) + 1$ and we are done.

□

This implies that there are at most n iterations of the alg.

Lemma 56. *If there are no cycles of positive capacity arcs, can compute a blocking flow in $O(nm)$ time. Idea: Search for path to t; if you find one, push flow until edge is blocked. Otherwise, you reach vertex that has no outgoing edges of positive capacity so can trim that vertex. Each path takes $O(n)$ to find; you remove one edge each time. Can be reduced to $O(m \log n)$ by using dynamic trees (Sleator, Tarjan 1980).*

Theorem 57. *Dinitz's algorithm can find a max flow in $O(mn \log n)$ time.*

Proof. Alg has n iterations. Within each iteration, take $O(m)$ time to compute $d(i)$ and $O(m \log n)$ time to compute blocking flow. Note that there's no cycles in $\hat{A}$ due to the restriction on distance labels. □

Special case: Unit capacity graphs $u(i,j) \in \{0,1\} \forall (i,j) \in A$. Recall the notion of distance level $B(k) = \{i \in V : d(i) = k\}$ and the $s-t$ cut $S(k) = \{i \in V : d(i) \geq k\}$. We define Λ to be $\min(\sqrt{m}, 2n^{2/3})$.

Lemma 58. *Dinitz's algorithm takes $O(\Lambda)$ iterations for unit capacity graphs.*

Proof. We first show it takes $O(\sqrt{m})$ iterations. After $\sqrt{m}$ iterations, $d(s) \geq \sqrt{m}$. We must have $\sqrt{m}$ non-empty distinct distance levels. Note that edges cannot advance more than a single distance level. Consider cuts $S(1), S(2), \ldots S(m)$. There must be some k, $1 \leq k \leq \sqrt{m}$ such that at most $\sqrt{m}$ arcs from distance level k to $k-1$.

But a cut in the residual graph of size at most $\sqrt{m}$ implies the max flow has size at most $\sqrt{m}$ greater than the current flow; since each iteration increases max flow, there are at most $\sqrt{m}$ more iterations.

To show $O(n^{2/3})$ iterations, we apply similar proof. After $2n^{2/3}$ iterations, exist non-empty distinct distance levels $0, 1, \ldots, 2n^{2/3}$. Some k must satisfy $|B(k)|, |B(k-1)| \leq n^{1/3}$. But then at most $n^{2/3}$ arcs between them; rest of proof is same as above.

□

Corollary 59. *In Λ iterations, $\exists k$ such that $|\delta^+(S(k))| \leq \Lambda$. (For next time)*

Theorem 60. *Can find max flow in unit capacity graphs in $O(\Lambda m)$ time.*

11 Lecture 11 (10/7)

Last time: Λ iterations of Dinitz's alg finds cut with size $< \Lambda$. Today: Try to apply this same notion to general capacity graphs. Goldberg-Rao (1998) runs in $O(\Lambda(m \log n)(\log(mU)))$ time.

Idea: Suppose that we can ensure that all arcs from distance level k to $k-1$ have residual capacity $\leq \Delta$. Then after Λ blocking flows, $\exists k$ such that:

$$|\delta^+(S(k))| \leq \Lambda \implies u_f(\delta^+(S(k))) \leq \Lambda\Delta$$

Then reduce Δ by a factor of 2 and repeat. This implies cut size in residual graph decreases by a factor of 2 every Λ iterations. Initially, min cut in G_f has capacity at most mU so we need $O(\Lambda \log(mU))$ iterations of finding blocking flows.

Idea: Set $\ell(i,j) = \begin{cases} 1 & \text{if } u_f(i,j) \leq \Delta \\ 0 & \text{otherwise} \end{cases}$

When computing $d(i)$ = distance of i to t, use the lengths $\ell(i,j)$. Then alter the definition of admissible to hold if $u_f(i,j) > 0$ and $d(i) = d(j) + \ell(i,j)$.

Now consider what should happen in execution of blocking flows alg. What problems could this cause?

1. Does $d(s)$ still increase for each blocking flow computation?
2. If admissible arcs are no longer acyclic, cannot use the fast algorithms we already know.

We first consider problem 2. Shrink SCC's of length zero admissible arcs into single node; then run blocking flow alg. So this would eliminate cycles of positive capacity nodes and we apply blocking flow alg. How do we route the flow in original graph?

All arcs within the component have large capacity (at least Δ). Pick a root r in the component, build intree of arcs going to r and outtree of arcs leaving r. Use intree to route all incoming flow to r and outtree to route all outgoing flow from r. If we restrict overall flow to $\Delta/4$, then use at most $\Delta/2$ capacity on each arc (all of which have capacity $\geq \Delta$).

Change our goals: At each step, either compute a blocking flow, or find a flow of value $\leq \Delta/4$.

Algorithm 11 Goldberg-Rao algorithm (Take 1)

procedure GoldbergRao(G)
  $F \leftarrow mU$; $f \leftarrow 0$
  while $F \geq 1$ **do**
    $\Delta \leftarrow \frac{F}{2\Lambda}$
    repeat
      $\ell(i,j) \leftarrow \begin{cases} 1 & \text{if } u_f(i,j) \leq \Delta \\ 0 & \text{otherwise} \end{cases}$
      Compute distances $d(i)$ to t using lengths $\ell(i,j)$
      $\hat{A} \leftarrow \{(i,j) \in A : d(i) = d(j) + \ell(i,j), u_f(i,j) > 0\}$
      Shrink SCCs of $\hat{A}$
      Find either flow $\hat{f}$ of value $\Delta/4$ or blocking flow $\hat{f}$
      Convert $\hat{f}$ into flow in original graph
      $f \leftarrow f + \hat{f}$
    until 5Λ iterations
    $F \leftarrow F/2$
  end while
  Return f
end procedure

Lemma 61. *F is an upper bound on the value of max flow in G_f.*

Proof. By induction. True initially since mU bounds any flow. After $S\Lambda$ iterations, either:

(1) In 4Λ iterations, flow increases by $\Delta/4$. So total increase $4\Lambda(\Delta/4) = \Lambda\Delta = F/2$. Then remaining flow in $G_f \leq F - F/2 = F/2$.

(2) We found a blocking flow in Λ iterations. Then $\exists$ cut in G_f of capacity $\leq \Lambda\Delta \leq F/2$ so that the value of max flow in $G_f \leq F/2$.

So $O(\log mU)$ iterations of dividing F by 2. In each of these, we run blocking flow alg $O(\Lambda)$

times so that the overall runtime is $O(\Lambda(m \log n)(\log mU))$.

□

Lemma 62. *If we find a blocking flow, then $d(s)$ increases. (This solves problem 1)*

Proof. Let f, d, ℓ be the flow, distances, lengths from previous iteration and f', d', ℓ' from the next iteration. WTS that for any shortest augmenting path P in $G_{f'}$:

(a) $\forall (i,j) \in P, d(i) \leq d(j) + \ell'(i,j)$.

(b) $\exists (i,j) \in P, d(i) < d(j) + \ell'(i,j)$.

Then $d'(s) = \sum_{(i,j)\in P} \ell'(i,j) > \sum_{(i,j)\in P}(d(i) - d(j)) = d(s) - d(t) = d(t)$.

To prove (a), start by proving that $d(i) \leq d(j) + \ell(i,j), \forall (i,j) \in P$. If $u_f(i,j) > 0$, then follows from validity of d. Otherwise, $u_f(i,j) = 0$ and flow must have been pushed backwards along (j,i). But this requires that $d(j) = d(i) + \ell(j,i) \implies d(i) \leq d(j) \leq d(j) + \ell(i,j)$.

When could it be the case that $d(i) \leq d(j) + \ell(i,j)$ but not $d(i) \leq d(j) + \ell'(i,j)$? Need $d(i) = d(j) + 1$, $\ell(i,j) = 1$ and $\ell'(i,j) = 0$. But if $d(i) = d(j) + 1$, the edge (j,i) is not admissible so we cannot push flow along it. Then $f(i,j) \leq f'(i,j) \implies u_f(i,j) \geq u_{f'}(i,j) \geq \Delta$. This implies $\ell(i,j) = 0$, a contradiction.

Now it remains to prove (b). But it turns out that this does not actually go through. Goldberg and Rao modify the alg slightly to make sure things work out. They introduce special arcs to make case (b) go through. An arc is special if $\Delta/2 \leq u_f(i,j) \leq \Delta, u_f(j,i) \geq \Delta, d(i) = d(j)$. Then set $\ell(i,j) = 0$.

□

New fastest algs:

$O(mn)$ (Orlin 2013) $\tilde{O}(m\sqrt{n} \log U)$ (Lee-Sidford 2014)

$\tilde{O}(m^{1/2}U^{1/2} \log U)$ (??? 2016)

$\tilde{O}(m\sqrt{n} \log U)$ (Liu-Sidford 2020)

12 Lecture 12 (10/12)

Min-cost circulation problem.

Input: $G = (V, A)$, costs $c(i,j), \forall (i,j) \in A$, capacities $u(i,j), \forall (i,j) \in A$, and lower bounds $\ell(i,j), \forall (i,j) \in A$.

Goal: find a circulation F that minimizes $c(f) = \sum_{(i,j)\in A} c(i,j)f(i,j)$.

A circulation is $f : A \to \mathbb{R}^{\geq 0}$ s.t.:

1. for all $(i,j) \in A$, $\ell(i,j) \le f(i,j) \le u(i,j)$ (capacity constraints)
2. for all $i \in V$, $\sum_{k:(i,k)\in A} f(i,k) = \sum_{k:(k,i)\in A} f(k,i)$ (flow conservation)

Related to min-cost flow problem (equivalent problems).

Input: $G = (V, A)$, costs $c(i,j), \forall (i,j) \in A$, capacities $u(i,j), \forall (i,j) \in A$, and demands/supplies $b(i), \forall i \in V$.

Goal: Find a flow that minimizes $\sum_{(i,j)\in A} c(i,j)f(i,j)$ s.t. $0 \le f(i,j) \le u(i,j)$ and $b(i) = \sum_{k:(i,k)\in A} f(i,k) - \sum_{k:(k,i)\in A} f(k,i)$. Note that in order for this to be possible, we need $\sum_{i\in V} b(i) = 0$.

To convert to min-cost circulation problem, add edges from s to vertices with $b(i) > 0$ and edges from $b(j) < 0$ to s with cost 0 and $\ell(s,i) = u(s,i) = b(i)$ and likewise for (j,s).

Reduction in the opposite direction also straightforward (omitted for time).

Now go back to considering circulations. We can think about the capacity constraints differently by using skew symmetry. Instead of $\ell(i,j) \le f(i,j) \le u(i,j)$, take $u(j,i) = -\ell(i,j)$. Then skew symmetry requires $f(i,j) = -f(j,i)$ so that $f(j,i) \le u(j,i)$ iff $f(i,j) \ge \ell(i,j)$ (so we are only considering upper bounds). Cost of the backward edge should cancel out: $c(i,j) = -c(j,i)$.

New def: A <u>circulation</u> $f : A \to \mathbb{R}$ satisfies:

1. $f(i,j) \le u(i,j), \forall (i,j) \in A$ (capacity constraint)
2. $f(i,j) = -f(j,i), \forall (i,j) \in A$ (skew symmetry)
3. $\sum_{k:(k,i)\in A} f(k,i) = 0, \forall i \in V$ (flow conservation)

Note that we redefine $c(f) = \frac{1}{2}\sum_{(i,j)\in A} c(i,j)f(i,j)$ since we count flow from backwards edges as well, with $c(i,j)f(i,j) + (-c(i,j))(-f(i,j)) = 2c(i,j)f(i,j)$.

Define the <u>residual graph</u> $G_f = (V, A)$ in much the same manner as before, with residual capacities $u_f(i,j) = u(i,j) - f(i,j) \ge 0$. Take $A_f = \{(i,j) \in A : u_f(i,j) > 0\}$ to be positive residual capacity arcs. Analogy to augmenting paths: negative-cost cycles $\Gamma \subseteq A_f$ so that we can push more flow along the cycle. Cost is negative if $c(\Gamma) = \sum_{(i,j)\in\Gamma} c(i,j) < 0$. Let $\delta = \min_{(i,j)\in\Gamma} u_f(i,j) > 0$. Set:

$$f'(i,j) = \begin{cases} f(i,j) + \delta & \text{if } (i,j) \in \Gamma \\ f(i,j) - \delta & \text{if } (j,i) \in \Gamma \\ f(i,j) & \text{otherwise} \end{cases}$$

This operation is denoted "pushing flow on/around Γ" or Cancel Γ. Note that f' obeys flow conservation, skew symmetry, and capacity constraints. Also not hard to show that $c(f') = c(f) + \delta c(\Gamma) < c(f)$, so that we get a better circulation.

We now introduce a dual notion (like min-cut for flows) to show that we can always find a

negative-cost cycle if f is not already a min-cost circulation. Define node potentials/prices to be $p : V \to \mathbb{R}$ and reduced cost of arcs w.r.t. p are $c_p(i,j) = c(i,j) + p(i) - p(j)$.

Note that $c_p(i,j) = -c_p(j,i)$, and also that $c_p(\Gamma) = c(\Gamma)$ (potentials telescope).

Lemma 63. *For potentials p, circulation f, $c(f) = c_p(f)$.*

Proof. Start with reduced cost:

$$\begin{aligned}
c_p(f) &= \frac{1}{2} \sum_{(i,j)\in A} c_p(i,j) f(i,j) \\
&= \frac{1}{2} \sum_{(i,j)\in A} (c(i,j) + p(i) - p(j)) f(i,j) \\
&= c(f) + \frac{1}{2} \sum_{(i,j)\in A} (p(i) - p(j)) f(i,j) \\
&= c(f) + \sum_{i\in V} p(i) \left(\sum_{k:(i,k)\in A} f(i,k) - \sum_{k:(k,i)\in A} f(k,i) \right) \\
&= c(f) \qquad \text{(flow conservation)}
\end{aligned}$$

□

Theorem 64. *TFAE:*

1. *f is a min-cost circulation*
2. *There is no neg-cost cycle in A_f*
3. *There are potentials p such that $c_p(i,j) \geq 0$ for all $(i,j) \in A_f$*

Proof. (1) $\implies$ (2): Done above (contrapositive).

(2) $\implies$ (3): Add s with edges to everything in V with cost 0. Define $p(i)$ to be the length/cost of shortest $s - i$ path using arcs in A_f and costs $c(i,j)$. Note that $p(j) \leq p(i) + c(i,j)$ for all $(i,j) \in A_f$. So $c(i,j) + p(i) - p(j) \geq 0$ and $c_p(i,j) \geq 0$. Note that condition (2) is required so that $p(i)$ is well-defined (otherwise we can just repeatedly traverse negative-cost cycle).

(3) $\implies$ (1): Let $\tilde{f}$ be any other circulation. WTS $c(\tilde{f}) \geq c(f)$. Consider $f' = \tilde{f} - f$; it obeys flow conservation + skew symmetry from $\tilde{f}$ and f. Also note that $f'(i,j) = \tilde{f}(i,j) - f(i,j) \leq u(i,j) - f(i,j) = u_f(i,j)$.

So f' is a circulation in G_f. Notice that $f'(i,j) > 0 \implies u_f(i,j) > 0 \implies (i,j) \in A_f$. We

can write:

$$\begin{aligned} c(\tilde{f}) - c(f) &= c(\tilde{f} - f) \\ &= c(f') \\ &= c_p(f') \\ &= \frac{1}{2} \sum_{(i,j) \in A} c_p(i,j) f'(i,j) \\ &= \sum_{(i,j) \in A : f'(i,j) > 0} c_p(i,j) f'(i,j) \\ &\geq 0 \end{aligned}$$

Since costs are all nonnegative and the sum is over positive flow edges. So $c(\tilde{f}) \geq c(f)$.

□

13 Lecture 13 (10/19)

Algorithm 12 Neg-cost cycle canceling (Klein '67).

```
1: procedure NegCostCycleCancel(G)
2:     Let f be any feasible circulation
3:     while ∃ neg-cost cycle Γ in G_f do
4:         Cancel Γ, update f
5:     end while
6:     Return f
7: end procedure
```

How do we find a feasible circulation? Can use max-flow (hw 1). Integrality property: If all ℓ, u are integers, then there exists an integer min-cost circulation (essentialyl same proof as integrality for max flows).

If costs are integers, then this is a pseudopolynomial-time alg. Let $U = \max_{(i,j) \in A} u(i,j)$ and $C = \max_{(i,j) \in A} |c(i,j)|$. Max cost of circulation is at most mCU and min cost is at least $-mCU$. If costs are integer, then for any neg-cost cycle Γ, $c(\Gamma) \leq -1$. So we change cost by at least 1 in each iteration, with $O(mCU)$ iterations overall. Claim that it takes $O(mn)$ time to find a neg-cost cycle, so overall runtime is $O(m^2 nCU)$ overall runtime.

Let the mean cost of cycle Γ be $\frac{c(\Gamma)}{|\Gamma|}$; let $\mu(f)$ be the min-mean cost in G_f; $\mu(f) := \min_{\Gamma \subseteq A_f} \frac{c(\Gamma)}{|\Gamma|}$. Note that $\mu_f < 0$ iff there is a neg-cost cycle in A_f; we can find $\mu(f)$ and Γ in $O(mn)$ time.

Min-mean cycle canceling (Goldberg, Tarjan '89). Basically the same as negative-cost cycle canceling except you pick min-mean cost cycle.

We say a circulation f is ε-optimal if $\exists$ potentials p s.t. $c_p(i,j) \geq -\varepsilon \forall (i,j) \in A_f$.

Observation: Any circulation is C-optimal (use 0 potentials). Additionally, any 0-optimal circulation satisfies condition (3) from last time, and is therefore a min-cost circulation.

Lemma 65. *If all costs are integer and f is ε-optimal for $\varepsilon < \frac{1}{n}$, then f is min-cost circulation.*

Proof. If hypothesis true, then $\exists$ potentials p s.t. $c_p(i,j) \geq -\varepsilon > -\frac{1}{n} \forall (i,j) \in A_f$. So consider any simple cycle $\Gamma \subseteq A_f$. Then $c(\Gamma) = c_f(\Gamma) > -\frac{1}{n}|\Gamma| \geq -1$. But if the costs are integer, this means that $c(\Gamma) \geq 0$. □

Take $\varepsilon(f) = \min \varepsilon$ such that f is ε-optimal.

Lemma 66. *If f is not a min-cost circulation, then $\varepsilon(f) = -\mu(f)$.*

Proof. First show that $\mu(f) \geq -\varepsilon(f)$. Note that $\exists$ potentials p s.t. $c_p(i,j) \geq -\varepsilon(f) \forall (i,j) \in A_f$. Let Γ be min-mean cost cycle. Then:

$$\mu(f) = \frac{c(\Gamma)}{|\Gamma|} = \frac{c_p(\Gamma)}{|\Gamma|} \geq -\frac{\varepsilon(f)|\Gamma|}{|\Gamma|} = -\varepsilon(f)$$

Now show that $\mu(f) \leq -\varepsilon(f)$. Set $\bar{c}(i,j) = c(i,j) - \mu(f) \forall (i,j) \in A_f$. For any cycle $\Gamma \subseteq A_f$:

$$\bar{c}(\Gamma) = c(\Gamma) - |\Gamma|\mu(f) \geq c(\Gamma) - |\Gamma|\frac{c(\Gamma)}{|\Gamma|} \geq 0$$

Take $p(i) =$ shortest path from s to i using costs $\bar{c}(i,j)$ on arcs in A_f. Note that $p(j) \leq p(i) + \bar{c}(i,j) = p(i) + c(i,j) - \mu(f)$, for all $(i,j) \in A_f$. Then $c_p(i,j) = c(i,j) + p(i) - p(j) \geq -\mu(f)$ for all $(i,j) \in A_f$. So f is $-\mu(f)$ optimal and $\varepsilon(f) \leq -\mu(f)$.

□

Given a circulation f, let $f^{(k)}$ be the circulation after k iterations.

Lemma 67. $\varepsilon(f^{(m)}) \leq \left(1 - \frac{1}{n}\right) \varepsilon(f)$

We reserve the proof of the lemma for now, and directly prove the following theorem:

Theorem 68. *Min-mean cycle canceling with integer costs takes $O(mn \ln(nC))$ iterations and therefore $O(m^2 n^2 \ln(nC))$ time. (Goldberg, Tarjan '89)*

Proof. Our initial circulation is always C-optimal. After $k = mn \ln(nC)$ iterations, the lemma says:

$$\varepsilon(f^{(k)}) \leq \left(1 - \frac{1}{n}\right)^{n \ln(nC)} \varepsilon(f) < e^{-\ln(nC)} C = \frac{1}{nC} \cdot C = \frac{1}{n}$$

But since $f^{(k)}$ is ε-optimal for $\varepsilon < \frac{1}{n}$ and costs are integer, $f^{(k)}$ is min-cost. □

We now prove the lemma.

Proof. Let f be a circulation, Γ our cycle to cancel, and f' the resulting circulation. There exist potentials p s.t. $c_p(i,j) \geq -\varepsilon(f)$ for all $(i,j) \in A_f$.

$$\mu(f) = \frac{c_p(\Gamma)}{|\Gamma|} \geq \frac{-\varepsilon(f)|\Gamma|}{|\Gamma|} = -\varepsilon(f)$$

But $\mu(f) = -\varepsilon(f)$, so $c_p(i,j) = -\varepsilon(f)$ for all $(i,j) \in \Gamma$. We now show that $c_p(i,j) \geq -\varepsilon(f)$ for all $(i,j) \in A_{f'}$ (the new circulation). If $(i,j) \in A_{f'}$, then either:

- $(i,j) \in A_f$ so that $c_p(i,j) \geq -\varepsilon(f)$.
- $(i,j) \notin A_f$, so we must have pushed flow in (j,i). But $(j,i) \in \Gamma$ implies $c_p(j,i) = -\varepsilon(f)$ so $c_p(i,j) = -c_p(j,i) = \varepsilon(f) \geq 0$.

Note that this implies $\varepsilon(f^{(1)}) \leq \varepsilon(f)$. More generally, when we cancel Γ s.t. $\forall (i,j) \in \Gamma, -\varepsilon(f) \leq c_p(i,j) \leq 0$, we only add arcs to $A_{f'}$ satisfying $c_p(i,j) \geq 0$. Consider having proceeded m iterations. Two cases:

1. For all Γ canceled, $c_p(i,j) < 0, \forall (i,j) \in \Gamma$. But each iteration can only add arcs with positive reduced cost to $A_{f'}$ and must saturate an arc (removing it from A_f) with $c_p(i,j) < 0$. After m iterations, all arcs with positive residual capacity have nonnegative reduced costs, and no negative cycles remain (so $f^{(m)}$ is a min-cost circulation).
2. In some iteration k, cancel Γ with $c_p(i,j) = 0$ for some edge in Γ. Then:

$$-\varepsilon(f^{(m)}) > -e(f^{(k)}) = \mu(f^{(k)}) = \frac{c_p(\Gamma)}{|\Gamma|} \geq \frac{-\varepsilon(f)(|\Gamma|-1)}{|\Gamma|} \geq \left(1 - \frac{1}{n}\right)(-\varepsilon(f))$$

   Dividing through the negative sign gives the desired inequality.

□

14 Lecture 14 (10/21)

We now consider a different runtime analysis of the min-mean cost cycle canceling alg. We aim to instead achieve a strongly polynomial runtime (eliminating the reliance on C). Historical significance: still unknown whether LP can be solved in strongly polynomial time; Tardos (1985) showed strongly polynomial runtime for this special problem.

Idea: Show that after a certain number of iterations, flow on a new arc becomes fixed. We say an arc (i,j) is $\underline{\varepsilon\text{-fixed}}$ if flow $f(i,j)$ is the same for all ε-optimal flows.

Lemma 69. *For any circulation f, any $S \subseteq V$, $S \neq \emptyset$, we have $\sum_{(k,\ell)\in\delta^+(S)} f(k,\ell) = 0$.*

Proof. By definition of flow conservation, $\sum_{k:(k,i)\in A} f(k,i) = 0$ for any i. So sum over all

vertices not in S.

$$\begin{aligned} 0 &= \sum_{i \notin S} \sum_{k:(k,i)\in A} f(k,i) \\ &= \sum_{i \notin S} \left(\sum_{k \in S:(k,i)\in A} f(k,i) + \sum_{k \notin S:(k,i)\in A} f(k,i) \right) \end{aligned}$$

The latter sum cancels out to 0 by skew symmetry.

$$= \sum_{(k,i)\in\delta^+(S)} f(k,i)$$

□

Lemma 70. *Let $\varepsilon > 0$, f a circ, p potentials s.t. f is ε-optimal w.r.t. p. If $c_p(i,j) \leq -2n\varepsilon$, then (i,j) is ε-fixed.*

Idea/proof sketch: Suppose $\exists$ ε-optimal f' such that $f'(i,j) \neq f(i,j)$. Then $\exists \Gamma \subseteq A_{f'}$ including (i,j). Then

$$-\varepsilon(f) = \mu(f') \leq \frac{c(\Gamma)}{|\Gamma|} < -\varepsilon$$

so that $\varepsilon(f') > \varepsilon$ and f' is not ε-optimal, contradiction.

Proof. Suppose that (i,j) is not ε-fixed, so that $\exists f'$ which is ε-optimal with $f'(i,j) \neq f(i,j)$. Since $c_p(i,j) \leq -2n\varepsilon < \varepsilon$ and f is ε-optimal w.r.t. p, $(i,j) \notin A_f$. This implies $u(i,j) = f(i,j) > f'(i,j)$.

Consider $A_< := \{(k,\ell) : f'(k,\ell) < f(k,\ell)\}$. We want to show that $\exists \Gamma \subseteq A_<$ such that $(i,j) \in \Gamma$. Note that $f'(k,\ell) < f(k,\ell) \leq u(k,\ell) \implies A_< \subseteq A_{f'}$.

Let $S = \{\text{vertices reachable from } j \text{ using arcs in } A_<\}$. WTS $i \in S$, so that (i,j) is in some cycle $\Gamma \subseteq A_<$. Suppose $i \notin S$. By above lemma, $\sum_{(k,\ell)\in\delta^+(S)} f(k,\ell) = 0$ and $\sum_{(k,\ell)\in\delta^+(S)} f'(k,\ell) = 0$. Subtracting these:

$$\sum_{(k,\ell)\in\delta^+(S)} (f(k,\ell) - f'(k,\ell)) = 0$$

But $f'(i,j) < f(i,j)$, so that $f'(j,i) > f(j,i)$ by skew-symmetry. By assumption, $(j,i) \in \delta^+(S)$. So (j,i)'s contribution to the sum is negative, and there must be some $(k,\ell) \in \delta^+(S)$ with $f(k,\ell) > f'(k,\ell)$. But then $k \in S, \ell \notin S$ and $(k,\ell) \in A_<$ requires $\ell \in S$ since we can also reach k from ℓ. So $i \in S$ and we have our desired cycle $\Gamma \subseteq A_< \subseteq A_{f'}$ including (i,j).

For any $(k,\ell) \in \gamma$, we have $f'(k,\ell) < f(k,\ell)$ and therefore $u(\ell,k) \geq f'(\ell,k) > f(\ell,k) \implies (\ell,k) \in A_f$. But then $c_p(\ell,k) \geq -\varepsilon$ so that $c_p(k,\ell) \leq \varepsilon$. Then for Γ we have:

$$\frac{c(\Gamma)}{|\Gamma|} = \frac{c_p(\Gamma)}{|\Gamma|} = \frac{1}{|\Gamma|}\left(c_p(i,j) + \sum_{(k,\ell)\in\Gamma\setminus\{i,j\}} c_p(k,\ell) \right) \leq \frac{1}{|\Gamma|}(-2n\varepsilon + (|\Gamma|-1)\varepsilon) < -\varepsilon$$

Then $u(f') \leq \frac{c(\Gamma)}{|\Gamma|} < -\varepsilon$. But $\varepsilon(f') = -\mu(f') > \varepsilon$. So then f' is not ε-optimal, a contradiction.

□

Lemma 71. *Let f and f' be ciruclations such that $\varepsilon(f') \leq \varepsilon(f)/2n$ and f is not min-cost. Then there are strictly more $\varepsilon(f')$-fixed arcs than $\varepsilon(f)$-fixed arcs.*

Proof. Since $\varepsilon(f') < \varepsilon(f)$, any $\varepsilon(f)$-fixed arc is also $\varepsilon(f')$-fixed; only need to show that there is some $\varepsilon(f')$-fixed arc that is not $\varepsilon(f)$-fixed. Let p be potentials s.t. f is $\varepsilon(f)$-optimal. Note that f is not min-cost, so let Γ be a min-mean cost cycle in G_f. Then:

$$-\varepsilon(f) = \mu(f) = \frac{c_p(\Gamma)}{|\Gamma|} \geq \frac{-\varepsilon(f)|\Gamma|}{|\Gamma|} = -\varepsilon(f)$$

So that $c_p(i,j) = -\varepsilon(f)$ for all edges in Γ. Note that none of these arcs are $\varepsilon(f)$-fixed, since cancelling Γ will modify flow on all of these arcs and resulting flow is still $\varepsilon(f)$-optimal.

Now let f' be $\varepsilon(f')$-optimal w.r.t. potentials p'. Consider the same cycle Γ.

$$\frac{c_{p'}(\Gamma)}{|\Gamma|} = \mu(f') = -\varepsilon(f) \leq -2n\varepsilon(f')$$

So there must exist some $(i,j) \in \Gamma$ with $c_{p'}(i,j) \leq -2n\varepsilon(f')$. Then (i,j) is $\varepsilon(f')$-fixed from previous lemma but not $\varepsilon(f)$-fixed, as desired. □

Theorem 72. *Min-mean cost cycle canceling takes $O(m^2 n \ln n)$ iterations and $O(m^3 n^2 \ln n)$ time.*

Proof. Claim a new arc is fixed every $k = mn \ln 2n$ iterations. Pick any circulation f; then $\varepsilon(f^{(k)}) \leq \left(1 - \frac{1}{n}\right)^{n \ln 2n} \varepsilon(f) < e^{-\ln(2n)}\varepsilon(f) = \frac{\varepsilon(f)}{2n}$. Applying the lemma gives a new arc fixed, so that there are at most km iterations. □

15 Lecture 15 (10/26)

Min-cost circulations: Wallacher's algorithm.

Idea: Find neg-cost cycle in A_f whose cancellation decreases cost by as much as possible.

$$\min_{\Gamma \subseteq A_f} c(\Gamma) \cdot \min_{(i,j)\in\Gamma} u_f(i,j)$$

Claim: NP-hard by reduction from Ham cycle (just set all costs to -1 and capacities to 1). So instead, we try to minimize:

$$\min_{\Gamma \subseteq A_f} \frac{c(\Gamma)}{\min_{(i,j)\in\Gamma} \frac{1}{u_f(i,j)}}$$

Instead, take the cycle that minimizes:

$$\beta(f) = \min_{\Gamma \subseteq A_f} \frac{c(\Gamma)}{\sum_{(i,j)\in\Gamma} \frac{1}{u_f(i,j)}}$$

Lemma 73. *Given circulation f, $\exists f_1, \dots f_\ell, \ell \leq m$ such that $f = \sum_{i=1}^{\ell} f_i$, $c(f) = \sum_{i=1}^{\ell} c(f_i)$ and f_i has positive flow only on arcs in a simple cycle.*

Proof. Similar to lemma for flows. Find a cycle by extending edges; pull the cycle out. Canceling saturates an edge, so can only happen up to m times. □

Lemma 74. *Let f^* be a min-cost circulation, f a circulation that is not min-cost. Then $\beta(f) \leq \frac{1}{m}(c(f^*) - c(f))$.*

Proof. Note that $f^* - f$ is a feasible circulation in G_f. Use flow decomposition lemma on $f^* - f$ in G_f. Get $f_1, \dots, f_\ell$ with $\ell \leq m$; let Γ_i be the cycle associated with f_i and let $\delta(i)$ be amt of flow on Γ_i, so $c(f) = \sum_i \delta(i) c(f_i)$.

Let $\beta(\Gamma) = \frac{c(\Gamma)}{\sum_{(i,j)\in\Gamma} \frac{1}{u_f(i,j)}}$.

$$\begin{aligned}
c(f^*) - c(f) &= \sum_{k=1}^{\ell} c(f_k) \\
&= \sum_{k=1}^{\ell} \delta_k c(\Gamma_k) \\
&= \sum_{k=1}^{\ell} \delta_k \beta(\Gamma_k) \sum_{(i,j)\in\Gamma_k} \frac{1}{u_f(\Gamma_k)} \\
&\geq \beta(f) \sum_{k=1}^{\ell} \delta_k \sum_{(i,j)\in\Gamma_k} \frac{1}{u_f(\Gamma_k)} \\
&= \beta(f) \sum_{(i,j)\in A_f} \frac{1}{u_f(i,j)} \sum_{k=1}^{\ell} \delta_k \\
&\geq m\beta(f)
\end{aligned}$$

□

Corollary 75. *Suppose costs c, capacities u are integer. Then if $\beta(f) > -\frac{1}{m}$, f is a min-cost circulation.*

Proof. f, f^* must be integer flows. This also holds for their costs. Previous lemma implies $-\frac{1}{m} < \beta(f) \leq \frac{1}{m}(c(f^*) - c(f))$ so that their costs must match. □

Algorithm 13 Wallacher's Algorithm

```
procedure WALLACHER(G)
    Let f be any feasible circulation
    while β(f) ≤ −1/m do
        Let Γ ⊆ A_f such that β(f) = β(Γ)
        Cancel Γ, Update f
    end while
    Return f
end procedure
```

This gives us the foundation behind our algorithm.

Lemma 76. *Let f be a circulation that is not min cost, and let f' be the result of canceling any $\Gamma \subseteq A_f$. Then $c(f') - c(f) \leq \beta(\Gamma)$.*

Proof. Let $\delta = \min_{(i,j)\in\Gamma} u_f(i,j)$, i.e. amount of flow pushed on Γ. Then:

$$c(f') - c(f) = \delta(c(\Gamma)) = \delta\beta(\Gamma) \sum_{(i,j)\in\Gamma} \frac{1}{u_f(i,j)} \leq \beta(\Gamma)$$

□

Corollary 77. *If $\beta(\Gamma) = \beta(f)$, then $c(f')-c(f) \leq \beta(\Gamma) \leq \frac{1}{m}(c(f^*)-c(f))$. So $c(f')-c(f^*) \leq \left(1-\frac{1}{m}\right)(c(f)-c(f^*))$.*

This form lends itself to the standard method of repeated application:

Theorem 78. *Suppose costs, capacities are integers. Then Wallacher's algorithm takes $O(m\ln(mCU))$ iterations, each of which takes $O(mn\ln(mCU))$ time (we won't prove this runtime). This gives overall runtime $O(m^2n\ln^2(mCU))$ time.*

Proof. If $f^{(k)}$ is circulation after k iterations, then $c(f^{(k)}) - c(f^*) \leq \left(1-\frac{1}{m}\right)^k (c(f)-c(f^*))$. For any circulation f, $c(f) - c(f^*) \leq 2mCU$. So after $k = m\ln(2mCU)$ iterations, we have:

$$c(f^{(k)}) - c(f^*) \leq \left(1-\frac{1}{m}\right)^{m\ln(2mCU)} (c(f)-c(f^*)) < e^{-\ln(2mCU)} 2mCU = 1$$

Since costs, capacities integer, this implies optimality. □

Now we work on improving runtime. Instead of finding the most-improving cycle, we search for a "good enough" cycle (similar to the Δ-scaling we did for max flows). We maintain an estimate $\hat{\beta}$ on value of $\beta(f)$. for circulation f, consider costs $\bar{c}(i,j) = c(i,j) - \frac{\hat{\beta}}{u_f(i,j)}$. Then for $\Gamma \subseteq A_f$, $\bar{c}(\Gamma) < 0$ iff $c(\Gamma) < \hat{\beta}\sum_{(i,j)\in\Gamma} \frac{1}{u_f(i,j)}$, i.e. $\hat{\beta} > \beta\Gamma$. This leads to the below algorithm:

Define a $\hat{\beta}$-scaling phase to be a sequence of iterations in which $\hat{\beta}$ has the same value.

Algorithm 14 Scaling Wallacher's Algorithm

```
procedure ScalingWallacher(G)
    Let f be any feasible circulation
    β̂ ← −CU
    while β̂ ≤ −1/(2m) do
        if ∃Γ ⊆ A_f s.t. ĉ(Γ) < 0 then
            Cancel Γ, update f
        else
            β̂ ← β̂/2
        end if
    end while
    Return f
end procedure
```

Lemma 79. *There are at most $2m$ iterations per $\hat{\beta}$-scaling phase.*

Proof. First show that $\hat{\beta} \leq \beta(f)/2$ at the start of any $\hat{\beta}$-scaling phase. Initially, $\hat{\beta} = -CU$. For any cycle Γ, we have:

$$\bar{c}(\Gamma) = c(\Gamma) + CU \sum_{(i,j)\in\Gamma} \frac{1}{u_f(i,j)} \geq c(\Gamma) + C|\Gamma| \geq 0$$

So $\hat{\beta} \leq \beta(\Gamma)$ for all Γ, and $\hat{\beta} \leq \beta(f) \leq \beta(f)/2$.

We now consider the end of any $\hat{\beta}$-scaling phase. At this point, $\bar{C}(\Gamma) \geq 0$ for all $\Gamma \subseteq A_f$ (so $\hat{\beta} \leq \beta(f)$). We divide $\hat{\beta}$ by 2 to start a new $\hat{\beta}$-scaling phase, so now $\hat{\beta} \leq \beta(f)/2$.

Let f be circulation at the start of $\hat{\beta}$-scaling phase, f^* a min-cost circulation. Then:

$$\hat{\beta} \leq \beta(f)/2 \leq \frac{1}{2m}(c(f^*) - c(f))$$

If we cancel a cycle Γ, the cost of circulation changes by $\beta(\Gamma) < \hat{\beta}$ (since $\bar{c}(\Gamma) < 0$). But we also know $\hat{\beta} \leq \frac{1}{2m}(c(f^*) - c(f))$. So there can be at most $2m$ iterations in each $\hat{\beta}$-scaling phase.

□

Theorem 80. *Scaling Wallacher takes $O(m \lg(mCU))$ iterations, for overall $O(m^2 n \ln(mCU))$ time.*

Proof. Initially, $\hat{\beta} = -CU$ and we stop when $\hat{\beta} > -\frac{1}{2m}$, for a total of $O(\log_2(mCU))$ $\hat{\beta}$-scaling phases, each of which contains at most $2m$ iterations. We may stop when $\hat{\beta} > -\frac{1}{2m}$, as this implies $\beta f > -\frac{1}{m}$ from above proof. □

Algorithm 15 Successive Approximation (Godlberg, Tarjan '90)

procedure SUCCESSIVEAPPROXIMATION(G)
  Let f be any feasible circulation
  $\varepsilon \leftarrow C, p(i) \leftarrow 0 \forall i \in V$
  while $\varepsilon \geq 1/n$ **do**
    $\varepsilon \leftarrow \varepsilon/2$
    $(f, p) \leftarrow$ FINDεOPTCIRC(f, ε, p)
  end while
  Return f
end procedure

16 Lecture 16 (10/28)

Recall that if f ε-optimal for $\varepsilon < 1/n$ with costs, caps integer, then f is min-cost.

Idea: FINDεOPTCIRC takes a 2ε-optimal circulation f (w.r.t. p) as input and produces ε-optimal circulation f' (w.r.t. p') as output. Same argument as before shows that we only need $O(\lg(nC))$ iterations.

Algorithm 16 Strongly polynomial successive approximation

procedure STRONGPOLYSUCCESSIVEAPPROXIMATION(G)
  Let f be any feasible circulation
  while true **do**
    Compute $\varepsilon(f)$, potentials p s.t. f is $\varepsilon(f)$-optimal w.r.t. p
    if $\varepsilon(f) > 0$ **then**
      $\varepsilon \leftarrow \varepsilon(f)/2$
      $(f, p) \leftarrow$ FINDεOPTCIRC(f, ε, p)
    else
      Return f
    end if
  end while
end procedure

Lemma 81. *The algorithm takes $O(m \log n)$ iterations.*

Proof. Pick any iteration with circ f. After $\log_2(2n)$ iterations later, we must have f' s.t. $\varepsilon(f') \leq \varepsilon(f)/2n$. But we proved that this implies at least one more fixed arc. So after $m \log_2(2n)$ iterations, all arcs are fixed and we must have a min-cost circ. □

But how do we actually implement the FINDεOPTCIRC subroutine? Define a pseudoflow $f : A \to \mathbb{R}$ to be s.t.:

- $f(i, j) = -f(j, i), \forall (i, j) \in A$
- $f(i, j) \leq u(i, j), \forall (i, j) \in A$

Note that we do not enforce flow conservation (like when we introduced preflows). Define the excess $e_f(i)$ to be the net flow into i, i.e. $\sum_{k:(k,i)\in A} f(k,i)$. If $e_f(i) < 0$, then we call this a deficit. Note the distinction between preflow: we do not require excesses to be nonnegative.

Our plan is as follows: Use the 2ε-optimal circulation to get an ε-optimal pseudoflow f, which we then transform into an ε-optimal circulation. Idea: Set $f(i,j) = u(i,j) \forall (i,j) \in A_f$ satisfying $c_p(i,j) < 0$.

MISSED THIS SECTION, FILL IN LATER

What if i is active, but no admissible $(i,j) \in A_f$ out of i? Then $c_p(i,j) \geq 0$ for all $(i,j) \in A_f$. Then we can relabel i: Set $p(i) = \max_{(i,j)\in A_f}(p(j) - c(i,j) - \varepsilon)$. Note that $p(i) \geq p(j) - c(i,j) - \varepsilon$ so that $c(i,j) + p(i) - p(j) \geq \varepsilon$, i.e. we maintain ε-optimality (with equality in at least one case).

Observe that we must have reduced $p(i)$ by at least ε, since $c_p(i,j) \geq 0$ before and $c_p(i,j) = -\varepsilon$ after relabel. Additionally, $c_p(k,i) \geq -\varepsilon$ for all $(k,i) \in A_f$ before relabel. After relabel, we must have $c_p(k,i) \geq 0$ since we decreased $p(i)$ by at least ε, i.e. no admissible arcs enter i.

procedure PUSH(i,j)
    $\delta \leftarrow \min(e_f(i), u_f(i,j))$
    $f(i,j) \leftarrow f(i,j) + \delta, f(j,i) \leftarrow f(j,i) - \delta$
end procedure

procedure RELABEL(i)
    $p(i) \leftarrow \max_{(i,j)\in A_f}(p(j) - c(i,j) - \varepsilon)$
end procedure

Algorithm 17 Push-relabel ε-Optimal Circulation

procedure FINDεOPTCIRC(G)
    For all $(i,j) \in A_f$, if $c_p(i,j) < 0$ then $f(i,j) \leftarrow u(i,j)$ and $f(j,i) \leftarrow u(j,i)$.
    while $\exists$ active i $(e_f(i) > 0)$ **do**
        if $\exists j$ such that (i,j) admissible, i.e. $u_f(i,j) > 0, c_p(i,j) < 0$ **then**
            PUSH(i,j)
        else
            RELABEL(i)
        end if
    end while
    Return f, p
end procedure

Lemma 82. *Let f be a pseudoflow, f' a circulation. For any i s.t. $e_f(i) > 0$, there exists a path $P \subseteq A_f$ to a node j such that $e_f(j) < 0$. Furthermore, $\forall (k,\ell) \in P, \exists (\ell,k) \in A'_f$ (all reverse edges in P have positive residual capacity w.r.t. f').*

Proof. We find $P \subseteq A_< = \{(k,\ell) \in A_f : f(k,\ell) < f'(k,\ell)$. Note that $f(k,\ell) < f'(k,\ell) \leq$

$u(k,\ell) \implies (k,\ell) \in A_f$ and $f'(\ell,k) < f(\ell,k) \leq u(\ell,k) \implies (\ell,k) \in A_{f'}$ (so that we get these conditions for free).

Pick i s.t. $e_f(i) > 0$. Let S be the set of vertices reachable from i in $A_<$. We want to show $\exists j \in S$ s.t. $e_f(j) < 0$. Take:

$$\begin{aligned}
-\sum_{k\in S} e_f(k) &= -\sum_{k\in S}\sum_{\ell:(\ell,k)\in A} f(\ell,k) \\
&= \sum_{k\in S}\sum_{\ell:(k,\ell)\in A} f(k,\ell) \\
&= \sum_{k\in S}\left(\sum_{\ell\in S:(k,\ell)\in A} f(k,\ell) + \sum_{\ell\notin S:(k,\ell)\in A} f(k,\ell)\right) \\
&= \sum_{k\in S}\sum_{\ell\notin S:(k,\ell)\in A} f(k,\ell) \\
&= \sum_{(k,\ell)\in\delta^+(S)} f(k,\ell)
\end{aligned}$$

Observe: $(k,\ell) \in \delta^+(S)$ requires $(k,\ell) \notin A_<$, i.e. $f(k,\ell) \geq f'(k,\ell)$. Applying over all excesses:

$$-\sum_{k\in S} e_f(k) = \sum_{(k,\ell)\in\delta^+(S)} f(k,\ell) \geq \sum_{(k,\ell)\in\delta^+(S)} f'(k,\ell) = 0$$

Last equality by lemma on circulations. But $i \in S$ with positive excess; since the sum of excesses of S is nonpositive, some vertex in S must have a deficit.

□

Lemma 83. *For any i, $p(i)$ decreases by at most $3n\varepsilon$.*

Proof. Let f', p' be 2ε-optimal circulation, potentials on input and f, p be ε-optimal pseudoflow, potentials at some point in alg. If $p(i)$ changes, $e_f(i) > 0 \implies \exists j$ s.t. $e_f(j) < 0$ and $i \to j$ path P exists with $P \subseteq A_f$ and P' (reverse path) exists $P' \subseteq A_{f'}$. So $(k,\ell) \in P \implies (k,\ell) \in A_f \implies c_p(k,\ell) \geq -\varepsilon$.

Then:

$$-\varepsilon|P| \leq \sum_{(k,\ell)\in P} c_p(k,\ell) = \sum_{(k,\ell)\in P} (c(k,\ell) + p(k) - p(\ell)) = p(i) - p(j) + \sum_{(k,\ell)\in P} c(k,\ell)$$

Same idea on the reverse path shows

$$-2\varepsilon|P'| \leq p'(j) - p'(i) + \sum_{(k,\ell)\in P} c(\ell,k)$$

Add these inequalities together:

$$-3\varepsilon|P| \leq p(i) - p(i') + p'(j) - p(j) = p(i) - p'(i)$$

Because j has a deficit now; it must have also had one beforehand (since we never create a new deficit). □

Corollary 84. *Number of relabels is $O(n^2)$*

Proof. Each vertex relabeled at most $3n$ times. Relabels decrease $p(i)$ by at least ε and $p(i)$ decreases by at most $3n\varepsilon$ over the course of the algorithm, so $\leq 3n^2$ overall relabels. □

17 Lecture 17 (11/2)

We analyze pushes like before: saturating pushes push $\delta = u_f(i,j)$ flow on (i,j) and non-saturating pushes push $\delta = e_f(i) < u_f(i,j)$.

Lemma 85. *Number of saturating pushes is $O(mn)$.*

Proof. Pick any arc (i,j); argue that we can only have $O(n)$ saturating pushes. Initially, $c_p(i,j) \geq 0$ so we need to relabel i before any pushes on (i,j). After a saturating push, we have to push on (j,i) to make (i,j) admissible again. But then (j,i) has to be admissible, which requires $c_p(j,i) < 0$. Then $c_p(i,j) > 0$, which implies that we must relabel i before we can push on (i,j) again. So each edge has at most $3n$ saturating pushes, for at most $3mn$ overall. □

Lemma 86. *The set of admissible arcs is always acyclic.*

Proof. By induction. True initially since no admissible arcs.

Consider pushes and relabels. A push on (i,j) can make (i,j) not admissible (can't create a cycle) and cannot make (j,i) admissible since $c_p(i,j) < 0$. So no new edges are introduced, and no new cycle can exist.

Now consider a relabel of i. We know from above that no admissible arcs enter i so we can't create a cycle. □

Lemma 87. *Number of nonsaturating pushes is $O(mn^2)$.*

Proof. Let $\Phi(i)$ be the number of vertices reachable from i on admissible arcs (always at least 1). Take $\Phi = \sum_{i \text{ active}} \Phi(i)$. Initially, no admissible arcs so $\Phi \leq n$. At the end, no active nodes so $\Phi = 0$ (and always bounded below by 0). How can Φ change?

- Saturating pushes on (i,j) can increase Φ by at most n each time by making j active.
- Relabeling i makes i active, so can add up to n new reachable nodes from i. Note also that i has no incoming admissible arcs so other values of $\Phi(j)$ are unaffected.
- Nonsaturating push on (i,j) changes Φ by at most $\Phi(j) - \Phi(i)$, since i is made inactive. But we know $\Phi(i) > \Phi(j)$ since i can reach j (we are pushing, so admissible arc) and

everything j reaches, but j cannot reach i (acyclic by lemma above). This implies that Φ decreases by at least 1 for each nonsaturating push.

Total amount Φ can increase is at most $n+n(3n^2+3mn)$ so the total number of nonsaturating pushes is at most this total increase in Φ and is $O(mn^2)$. □

So the FindεOptCirc subroutine runs in $O(mn^2)$ time. With more care, this can be improved to $O(mn\log(n^2/m))$.

Theorem 88. *(Goldberg, Tarjan 1990) Successive approximation finds a min-cost circulation in $O(mn^2 \cdot \min(\log(nC), m\log n))$ time. This can be improved with the more optimal subroutine.*

Note that this is a strongly polynomial runtime. How does this compare with the state-of-the-art?

- Orlin (1993): $O(m\log n(m + n\log n))$
- Lee, Sidford (2014) $\tilde{O}(m\sqrt{n}\log^{O(1)}(CU))$

In practice, successive approximation runs quickly. So does network simplex (which we will see later), despite not being poly-time in general.

18 Lecture 18 (11/4)

Specialization of the Simplex Method to min-cost circulations.

(Primal) network simplex maintains a spanning tree T, a feasible circulation f, and potentials p such that:

1. If both $u_f(i,j) > 0, u_f(j,i) > 0$, then $\{i,j\} \in T$.
2. If $\{i,j\} \in T$, then $c_p(i,j) = c_p(j,i) = 0$.

Given T, easy to compute p s.t. condition 2 holds. Pick arbitrary root r and set $p(r) = 0$; then just fill in by BFS.

Given feasible circulation f, compute circulation f' and spanning tree T to satisfy the first condition, with $c(f') \le c(f)$. Consider $E = \{\{i,j\} : u_f(i,j) > 0, u_f(j,i) > 0\}$. If it has a cycle, one of the directions is non-positive-cost, so can cancel to saturate an edge. If there's no cycles, we can just add edges to span.

Each nontree arc $(i,j) \in A_f$ defines a basic cycle $\Gamma(i,j)$, the cycle formed by adding (i,j) into T. Observe that $c_p(\Gamma(i,j)) < 0$ iff $\overline{c_p(i,j) < 0}$ (all costs in T are 0).

If we cancel $\Gamma(i,j)$, then some $(k,\ell) \in \Gamma(i,j)$ is saturated (perhaps $(k,\ell) = (i,j)$); add $\{i,j\}$ to T and remove $\{k,\ell\}$ from T. We say that $\{i,j\}$ enters T and $\{k,\ell\}$ leaves T; pivot on (i,j).

Note that at termination, no negative-cost cycles can exist in A_f (all edges in T are 0-cost

Algorithm 18 Network Simplex

```
1: procedure FindεOptCirc(G)
2:     Let f be a feasible circulation
3:     Find f, T, p obeying our properties (1), (2)
4:     while ∃ nontree arc (i, j) ∈ A_f with c_p(i, j) < 0 do
5:         Cancel Γ(i, j)
6:         Update f, T, p
7:     end while
8:     Return f
9: end procedure
```

w.r.t. p and no negative cost edges outside it exist), so we have a min-cost circulation.

(Zadeh '73) Natural pivot rules lead to exponential number of pivots.

(Orlin '97) Can choose a pivot each iteration so that at most $O(nm \cdot \min(\log(nC), m \log n))$ pivots (not similarity to successive approximation runtime). Each pivot takes $O(n)$ amortized time, for $O(n^2 m \cdot \min(\log(nC), m \log n))$ overall.

Research question: Are there other poly-time pivot rules? Can show that if we can cancel $\Gamma(i,j)$ s.t. $c_p(\Gamma(i,j)) \geq 0$, then other poly-time pivot rules exist.

Typical problem: max $s-t$ flow problem over time, (aka max dynamic flow). Inputs:

- Directed $G = (V, A)$
- Capacities $u(i,j) \geq 0$ for all $(i,j) \in A$, integer
- Source $s \in V$, sink $t \in V$
- Transit time $\tau(i,j) \geq 0$ for all $(i,j) \in A$, integer
- Time bound T

Idea: Putting unit of flow in i at time t, arrives at j at time $t + \tau(i,j)$. $u(i,j)$ limits the amount of flow that can be put into the arc at any given time, e.g. the rate of the flow.

Goal: Find max amount of flow that can be sent from s starting at time 0 to arrive at sink t by time T.

We introduce the notion of a time-expanded network: make copies of each vertex for each time, $v(0), \ldots, v(T)$. Arcs go from vertices $i(t), j(t + \tau(i,j))$ with capacity $u(i,j)$ and "holdover arcs" $i(t)$ to $i(t+1)$ (flow can remain at the same vertex). Then just solve max flow problem from $s(0)$ to $t(T)$. Problem: Not poly-time, since network is exponential in size of input (nT vertices).

Suppose we have $s-t$ path P such that $\tau(P) \equiv \sum_{(i,j)\in P} \tau(i,j)$ and let $u = \min_{(i,j)\in P} u(i,j)$. Can send u units of flow along P at time $0, 1, 2, \ldots, T - \tau(P)$ (and guarantee flow reaches t). Call a flow over time along a path like this a temporally repeated flow.

Lemma 89. *Given a standard $s-t$ flow f, decomposition of f into flows $f_1, \ldots, f_\ell$ each on a $s-t$ path $P_1, \ldots, P_\ell$ s.t. $\tau(P_i) \leq T$ for $1 \leq i \leq \ell$, then the value of $s-t$ flow over time by temporally repeating these paths is:*

$$(T+1)|f| - \sum_{(i,j)\in A} \tau(i,j) f(i,j)$$

Proof. We first show that this temporal repetition leads to a valid flow over time. At any time t, how much flow is entering (i,j)? Every path P_k using (i,j) sends $f_k(i,j)$ flow; capacity constraints satisfied since they hold for f.

How much flow is sent over time? Each path P_k sends $|f_k|$ flow at each timestep $0, 1, \ldots, T - \tau(P_k)$:

$$\begin{aligned}
\sum_{k=1}^{\ell} |f_k|(T+1-\tau(P_k)) &= \sum_{k=1}^{\ell} |f_k| T + 1 - \sum_{k=1}^{\ell} |f_k| \tau(P_k)) \\
&= (T+1)|f| - \sum_{k=1}^{\ell} |f_k| \sum_{(i,j)\in P_k} \tau(i,j) \\
&= (T+1)|f| - \sum_{(i,j)\in A} \tau(i,j) \sum_{k:(i,j)\in P_k} |f_k| \\
&= (T+1)|f| - \sum_{(i,j)\in A} \tau(i,j) f(i,j)
\end{aligned}$$

□

19 Lecture 19 (11/9)

Continuing from last time: How can we find a flow to maximize this value? Maximizing $(T+1)|f| - \sum_{(i,j)\in A} \tau(i,j) f(i,j)$ is equivalent to minimizing $\sum_{(i,j)\in A} \tau(i,j) f(i,j) - (T+1)|f|$. We find a min-cost circulation to minimize this value. Create a new instance (V, A') where $c(i,j) = \tau(i,j)$, $u(i,j) = u(i,j)$, $c(j,i) = -\tau(i,j)$, $u(j,i) = 0$ and $c(t,s) = -T+1, u(t,s) = \infty, c(s,t) = T+1, u(s,t) = 0$.

Basic idea: The (t,s) edge accounts for the $-(T+1)|f|$ term. Thinking about negative-cost cycling: Intuition is that this $s-t$ edge is always going to be part of any negative-cost cycle, so any cancellation and change in flow must push along this edge.

But it's not altogether clear that the best temporally repeating flow is also the best flow over time (perhaps we can do better by changing what we do at different times). In fact, we can prove that it is indeed optimal:

Theorem 90. *(Ford, Fulkerson '62) The value of the max $s-t$ flow over time equals the value of the maximum temporally repeated flow.*

Proof. We consider again the time-expanded network. Consider the min $s(0)-t(T)$ cut. We show that the max temporally repeated flow has the same value as the capacity of this cut.

Recall that f is min-cost iff $\exists p$ such that $c_p(i,j) \geq 0, \forall (i,j) \in A_f$. If the reduced cost is negative, we must have $f(i,j) = u(i,j)$. The cost of the circulation is $c(f) = c_p(f) = \frac{1}{2}\sum_{(i,j)\in A'} c(i,j)_p f(i,j) = \sum_{(i,j)\in A': c_p(i,j)<0} c_p(i,j)f(i,j)$ (we take out the positive reduced cost arcs). But this is simply $\sum_{(i,j)\in A} \tau(i,j)f(i,j) - (T+1)f(t,s)$.

Assume $f(t,s) > 0$ so that circulation has negative cost, and let S be a cut in the time-expanded network formed by taking:

$$S = \{i(\theta) : i \in V, p(i) - p(s) \leq \theta\}$$

So for $x = p(i) - p(s)$, we get nodes $i(x), \ldots, i(T)$.

Note that $s(0) \in S$, but $t(T)$ is not. We assumed $f(t,s) > 0$ and hence $f(s,t) < 0$, both have positive residual capacity. So $c_p(t,s) \geq 0$ and $c_p(s,t) \geq 0$; by skew symmetry, they must both be 0. Then $0 = c_p(t,s) = c(t,s) + p(t) - p(s) \implies p(t) - p(s) = T + 1$. This in turn demonstrates that all $t(x) \notin S$ and S is an $s(0) - t(T)$ cut.

We now consider its capacity. Note that no holdover arcs appear in $\delta^+(S)$. For any (i,j), we get a copy of (i,j) in $\delta^+(S)$ for each time θ such that $p(i)-p(s) \leq \theta, p(j)-p(s) > \theta+\tau(i,j)$. Capacity of the cut is:

$$\sum_{(i,j)\in A} u(i,j) \cdot \max\{0, p(j) - p(s) - \tau(i,j) - (p(i) - p(s))\}$$

This simplifies to:

$$\sum_{(i,j)\in A} u(i,j) \cdot \max\{0, p(j) - p(i) - \tau(i,j)\}$$

But recall that we took $p(j)-p(i)-\tau(i,j)$ to be the negative reduced cost $-c_p(i,j)$. Finally, $u(i,j) = f(i,j)$ for negative reduced cost edges. This gives exactly the value of the max temporally repeated flow:

$$(T+1)|f| - \sum_{(i,j)\in A} \tau(i,j)f(i,j)$$

□

Problems get hard fast:

- Quickest transshipment: poly-time computable (Hoppe, Tardos) but complicated
- Quickest min-cost flow is NP-hard
- Quickeest multicommodity flow is NP-hard

20 Lecture 20 (11/11)

Generalized flow. Idea: Want to model "lossy" flow or flow where transformations occur in arcs. Each arc has a gain factor γ which modifies the amount of flow out based on multiplying by flow in. Example: graph of currency conversion rates.

Input: Directed graph $G = (V, A)$ with $(j, i) \in A$ iff $(i, j) \in A$. Capacities $u(i, j) \geq 0$ for all $(i, j) \in A$, integers. Gains $\gamma(i, j)$ for all $(i, j) \in A$, ratio of integers. We additionally assume $\gamma(j, i) = 1/\gamma(i, j)$. Also assume all integers bounded by B, and we have a sink $t \in V$ (not necessarily a source).

Take a generalized pseudoflow $f : A \to \mathbb{R}$ to be such that:

- $f(i, j) \leq u(i, j)$ for all $(i, j) \in A$
- $f(i, j) = -\gamma(j, i)f(j, i)$

We define the excess $e_f(i) \equiv -\sum_{k:(i,k)\in A} f(i, k)$, the net flow entering i.

A flow is a generalized pseudoflow such that $e_f(i) \geq 0$ for all $i \in V$. A flow is proper if $e_f(i) = 0$ for all $i \in V, i \neq t$.

Goal: Find a (proper) flow that maximizes $e_f(t) \equiv |f|$.

Define the residual graph as before: $G_f = (V, A)$ with $u_f(i, j) = u(i, j) - f(i, j)$ and $A_f = \{(i, j) : u_f(i, j) > 0\}$. But we also introduce a new notion. A labeling function $\mu : V \to \mathbb{R}_{\geq 0}$ is s.t. $t = 1$; intuitively corresponds to a change in units of measurement at a node ($\mu(i) =$ new units/old units).

Changing labels affects other quantities: e.g. conversion rates and capacities.

- Capacities: $u^\mu(i, j) = u(i, j)\mu(i)$.
- Gains: $\gamma^\mu(i, j) = \gamma(i, j)\mu(j)/\mu(i)$. (note that this preserves our skew-symmetry)
- Flows: $f^\mu(i, j) = f(i, j)\mu(i)$.
- Excesses: $e_f^\mu(i) = e_f(i)\mu(i)$

How does relabeling affect the value of the flow? Since $\mu(t) = 1$, the excess at t is preserved under relabeling $|f^\mu| = f$.

The gain of a path P is $\gamma(P) = \prod_{(i,j)\in P} \gamma(i, j)$; same thing for cycle.

A cycle is flow generating if $\gamma(C) > 1$; in the case of currency conversion, this gives arbitrage opportunity. If $\gamma(C) < 1$, then it is flow absorbing.

We say μ is the canonical labeling if $\mu(i) = \max_{i-t \text{ paths } P} \gamma(P)$ (if no such paths exist, set $\mu(i) = 0$). Can turn this into a shortest path problem by setting $c(i, j) = -\log \gamma(i, j)$; summing costs is same as multiplying gain factors, and minimize instead of minimize. To compute these labels, need to have no negative-cost cycles reaching t.

Given a flow-generating cycle C that can reach t, we can push δ flow around the path, end up with $\gamma(C)\delta$ flow after one loop, and send the extra $(\gamma(C)-1)\delta$ flow towards t. This preserves flow conservation, and allows us to have a sink without a source. Such a path is called a generalized augmenting path (GAP), i.e. a flow-generating cycle C in A_f with a path P in A_f from a node in C to t.

Theorem 91. *TFA for a proper flow f:*

(1) f is a maximum generalized proper flow.

(2) There are no GAPs in A_f.

(3) $\exists$ labeling μ s.t. $\gamma^\mu(i,j) \leq 1$ for all $(i,j) \in A_f$.

Proof. We showed $\neg(2) \implies \neg(1)$, since given a GAP, we can increase the flow into sink t.

We show (2) $\implies$ (3) by computing canonical labels given assumption (2). Let S be nodes that can reach t on arcs in A_f. Compute the canonical labels $\mu(i)$ for all $i \in S$; since no GAPs exist, no flow-generating cycles in S. Then shortest path algs work.

Note that for any $i,j \in S$, $\mu(i) \geq \gamma(i,j)\mu(j)$. So $(i,j) \in A_f \implies \frac{\gamma(i,j)\mu(j)}{\mu(i)} \leq 1 \iff \delta^\mu(i,j) \leq 1$.

Other cases:

- $i \notin S, j \in S$ (cannot happen for edge in A_f).
- $i \notin S, j \notin S$ so both labels 0; we can take the ratio to be 1 by convention.
- $i \in S, j \notin S$ has $\gamma(i,j)\mu(j)/\mu(i) = 0 \leq 1$.

Last implication will be covered in additional video.

□