6.889	Lecture 20	Nov. 21, 2011

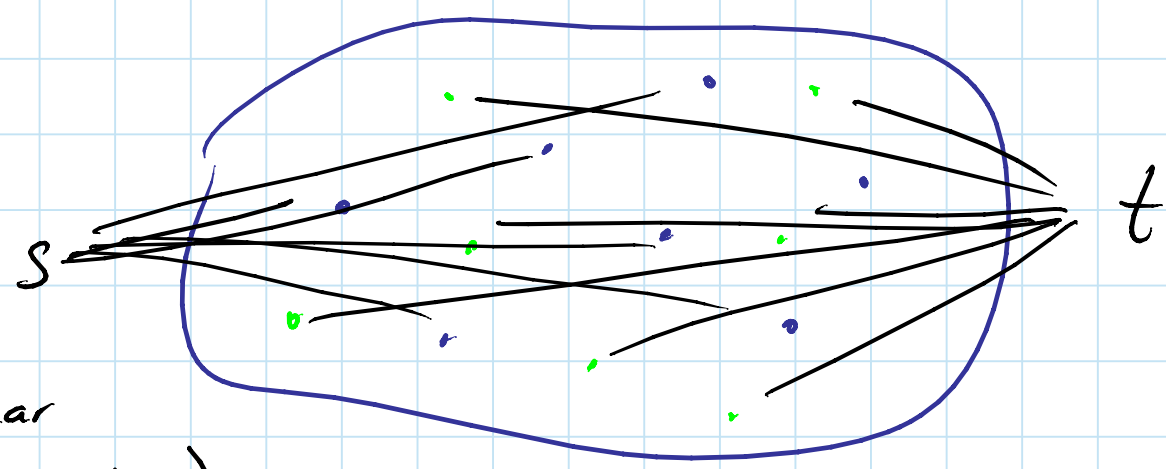
Maximum flow with multiple sources and sinks in directed planar graphs.

Input: $G=(V,A)$, non-negative arc capacities c, $S, T \subset V$

Output: flow assignment f that respects capacities, obeys conservation everywhere except $S \cup T$ and maximizes $\sum_{d:\, tail(d) \in S} f(d)$, the total flow leaving S.

In general graphs there is a simple reduction to max st-flow: connect an artificial source s to all $v \in S$ with ∞-capacity arcs. Similarly, connect all $v \in T$ to an artificial sink t. Compute maximum st-flow.

destroys planarity

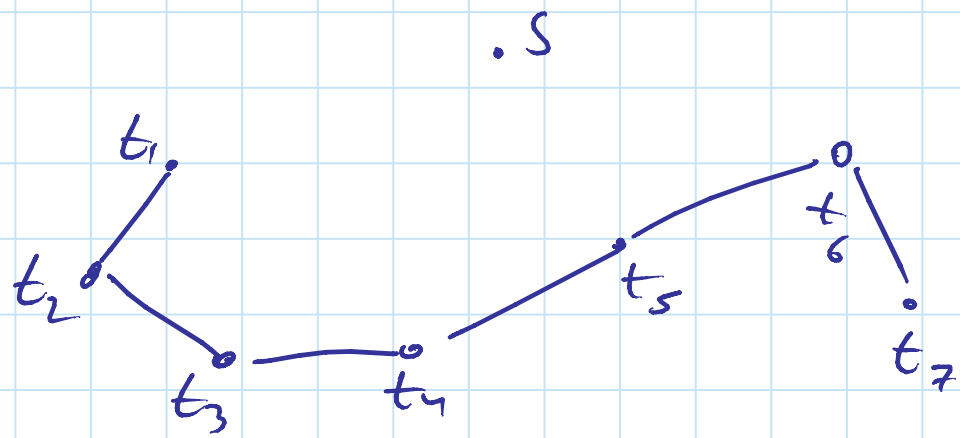


(this is a planar graph with 2 apices)

so cannot use the $O(n \log n)$ algorithm from last lecture.

Warm up:

single source, many sinks, but sinks induce a path P in G.



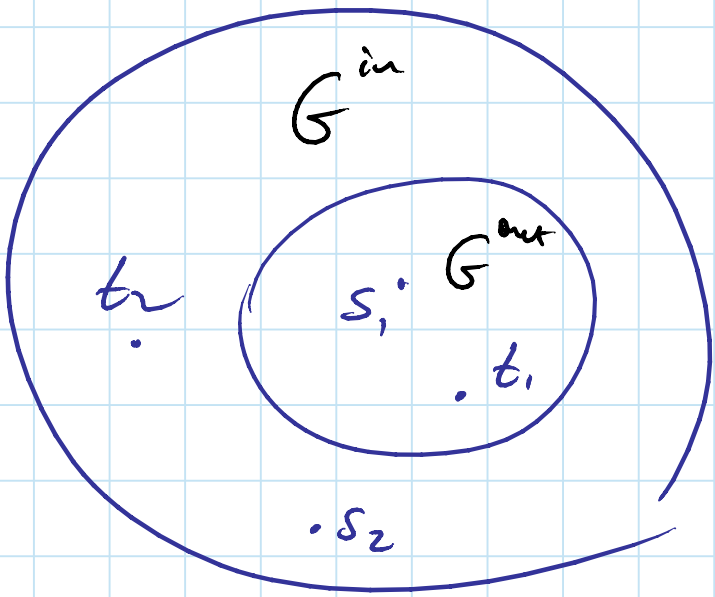
make capacities of edges of P infinite
compute max flow from s to t_1
don't use flow on any edge of P.

Another view: contract all edges of P, now there's just a single sink.

Recursive approach:

try this: - Find a cycle separator C
- solve recursively inside and outside C.

$\downarrow$ G^{in} $\qquad$ $\downarrow$ G^{out}



Problem: this pushes flow from s_1 to t_1 and s_2 to t_2 but how about s_1 to t_2 and s_2 to t_1?

notation: $A \xrightarrow{G} B$ if $\exists$ residual path from A to B in G.

$A \overset{G}{\not\to} B$ otherwise

second attempt: think in terms of saturating residual paths. We want to saturate (eliminate) all residual paths from any source to any sink. Any such path between $s_i \in G^{in}$ and $t_j \in G^{out}$ crosses C, so recursively solve a more general problem:

find a flow f in G^i s.t.

$$S \overset{G_f^i}{\not\longrightarrow} T$$

$$S \overset{G_f^i}{\not\longrightarrow} C$$

$$C \overset{G_f^i}{\not\longrightarrow} T$$

in terms of flow, this means: push max flow in G^i from S to T, from S to C and from C to T.

This guarantees $S \not\to T$ in the entire graph!

<u>problem:</u> conservation is violated on nodes of C.

We call a flow assignment that respects capacities (but possibly violates conservation) a pseudoflow.

The inflow at v (also called excess flow at v) is:

$$\sum_{d:\ \mathrm{head}(d)=v} f(d).$$

Inflow can be positive or negative, zero when f obeys conservation at v.

maximality criterion:

A st-flow f is maximal if $s \overset{G_f}{\not\to} t$.

when can we say that a pseudoflow is maximal?

Think of sources as having $+\infty$ excess
Think of sinks as having $-\infty$ excess
V^+ (V^-): non terminals with positive (negative) excess.

Say a pseudoflow f is maximum if $+ \overset{G_f}{\not\to} -$,
where $+ = S \cup V^+$
$- = T \cup V^-$

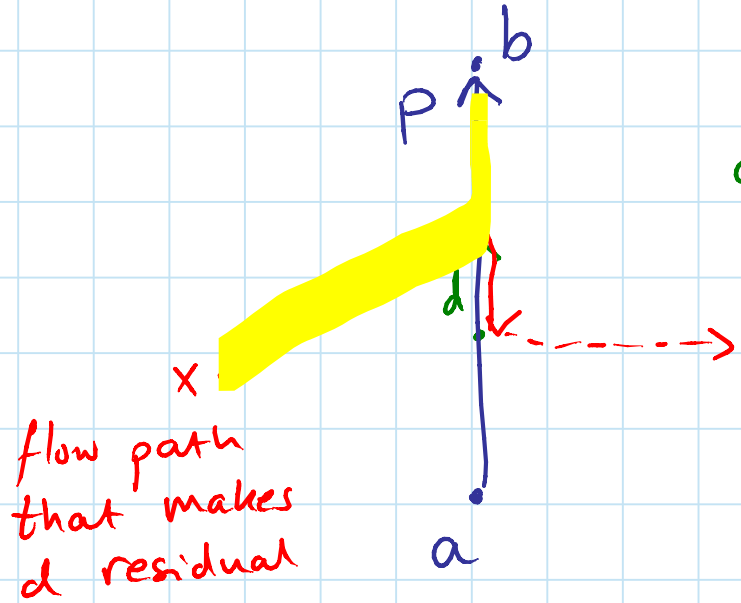
f is maximum in the sence that we can efficiently convert it into a maximum flow that does obey conservation.

1) Make f acyclic (see problem set)
2) in reverse topological order, unpush excess flow entering V^+
3) in topological order, unpush excess flow leaving V^-

when done, f obeys conservation and $S \overset{G_f}{\not\to} T$.
why?
Lemma: Let η be a flow with source set X. Let A, B be two disjoint sets of nodes.

$$A \cup X \overset{G}{\not\to} B \Rightarrow A \cup X \overset{G_\eta}{\not\to} B$$

Proof: By contradiction: P: residual a-to-b path after push, $a \in A \cup X$, $b \in B$



d: last dart of P that was not residual before push

so highlighted path was residual before push, a contradiction $\square$

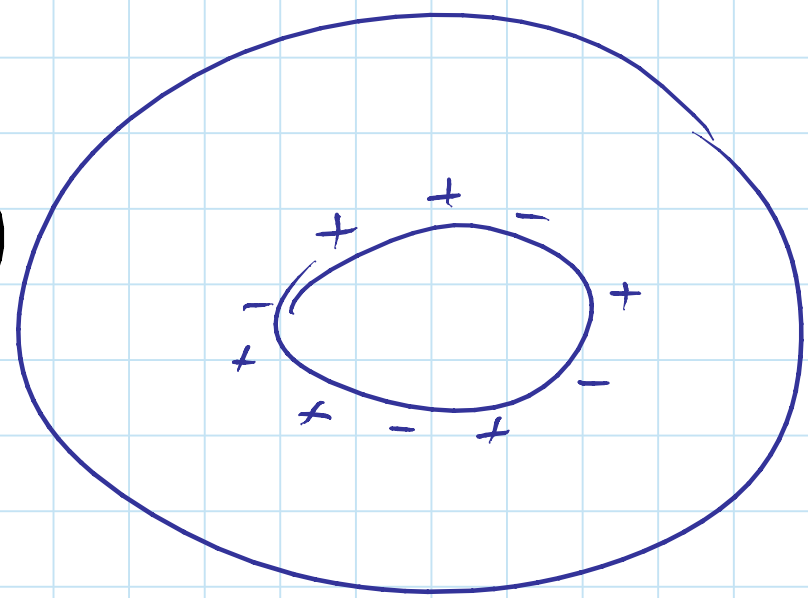
Back to our recursive algorithm:

after recursive calls:

① $S \overset{G_f}{\not\longrightarrow} T$

② $S \overset{G_f}{\not\longrightarrow} C$ (so $S \not\leftrightarrow V^-$)

③ $C \overset{G_f}{\not\longrightarrow} T$ (so $V^+ \not\leftrightarrow T$)



but maybe $V^+ \longrightarrow V^-$

we want to find flow f' in G_f with sources and sinks on C s.t. in addition, ④ $V^+ \overset{G_{f+f'}}{\not\longrightarrow} V^-$

Cycle fixing procedure: first solve in $O(n^{3/2})$, then in $\tilde{O}(n)$

Handle nodes on C one after the other in cyclic order. $v_1, v_2, \dots, v_k$ $\quad k = O(\sqrt{n})$

Suppose v_i has positive excess $q > 0$
push as much of the excess as possible to $v_{i+1}, \dots v_k$ by:

1) set capacities of edges of C between $v_{i+1}, \dots, v_k$ to ∞
2) push at most q units of flow from v_i to v_{i+1} C using SP in the dual.
3) reset capacities of edges of C between $v_{i+1}, \dots, v_k$ and unpush flow on these edges

at each step, either we get rid of all the excess flow, or of all residual paths between v_i and v_j $(j > i)$

after all iterations, $V^+ \not\longrightarrow V^-$ as desired

running time: k iterations, each takes $O(n)$ for computing SP in the dual.
so $O(kn) = O(n^{3/2})$

Flow representation:

at iteration i, in step ② we compute shortest paths in dual of residual graph and use them as face prices $\varphi_i(\cdot)$ that induce a circulation

$$\theta_i(d) = \varphi_i(\text{head}(d^*)) - \varphi_i(\text{tail}(d^*))$$

since we want a $v_i v_{i+1}$ flow rather than a circulation we do not want to push θ_i on the edge $v_i v_{i+1}$. In fact, in step ③ we do not want to push θ_i on any edge $v_j v_{j+1}$ for $j = i+1, \dots, k-1$

at each step flow can be decomposed into

a) circulation θ_i represented by potentials φ_i

b) a flow η_i defined by

$$\eta_i(d) = \begin{cases} \varphi_i(\text{tail}(d^*)) - \varphi_i(\text{head}(d^*)) & \text{if } d = v_j v_{j+1} \text{ for } i \le j < k \\ 0 & \text{otherwise} \end{cases}$$

so can be represented by just i, φ_i

so total flow before iteration i is represented by:

(a) the flow f we started with (from recursion)

(b) face potentials $\varphi = \sum_{j<i} \varphi_j$

(c) flow on edges of C $\eta = \sum_{j<i} \eta_j$

given φ, η how do we compute φ_i, η_i?

need to compute SP in dual of residual graph w.r.t. flow represented by φ, η

lengths in dual are residual capacities. for $d^* \notin C$ this is:

$$c(d) - f(d) + \varphi(\text{tail}(d^*)) - \varphi(\text{head}(d^*))$$

which is the reduced length of $c - f$ w.r.t. φ.

for $d^* \in C$ it is

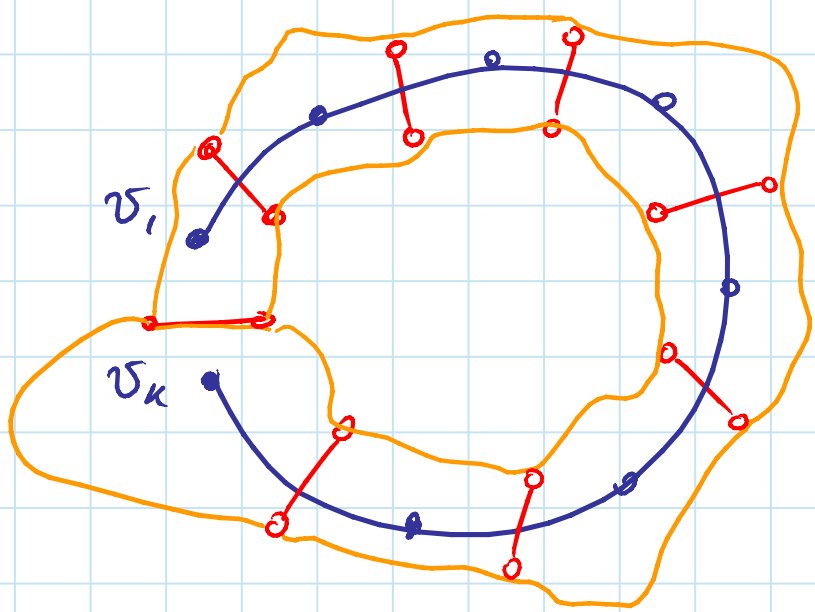
$$c(d) - f(d) + \varphi(\text{tail}(d^*)) - \varphi(\text{head}(d^*)) - \eta(d)$$

Claim: it suffices to maintain just η and the values of φ for faces that are endpoints of dual darts of C

Proof: We need to show that given η and the values of φ for just these faces, we can compute η_i and the values of φ_i for just these faces.

η_i only depends on these values of φ_i.

φ_i are the distances in the dual of the residual graph.



Let $H = G^* \setminus C^*$. precompute the dense distance graph between $V(C^*)$ in H before first iteration (i.e., w.r.t. lengths $c-\gamma$)

Since at any iteration, lengths of darts in H are just reduced lengths w.r.t. φ, shortest paths in H do not change, and the distances in the DDG only change by the difference of φ on their endpoints, which are exactly the endpoints of darts of C^*.

φ_i on the endpoints of C^* are the distances in G^* from some endpoint of C^* to all the other endpoints of C^*. Such SP can be decomposed into darts of C^* and edges of the DDG, so we have all the info required to compute φ_i.

How fast can we compute φ_i ?

Can be done with a hybrid of FR-Dijkstra and regular Dijkstra - edges of the DDG are relaxed implicitly using the FR data structure
edges of C^* are relaxed explicitly using a regular heap (FR's global heap)

time required is $O(\sqrt{n}\log^2 n + \sqrt{n}\log n)$

FR

explicitly relaxing edges of C

note: - we cannot afford to update the lengths of edges of DDG to reduce lengths (there are $O(n)$ edges). However, the reduced length of any DDG edge can be computed in constant time from φ whenever FR-Dijkstra requires one.

- FR-Dijkstra requires a RMQ structure that, for every node x in left side of bipartite instance, returns $\min_{i_- \leq i \leq i_+} d(x, y_i)$ for any i_-, i_+.
  These RMQ structures need to be reconstructed whenever the price function changes.
  Existing constructions are too slow.
  Can be constructed in $O(\sqrt{n} \log^2 n)$ by exploiting the Monge property [Kaplan, Mozes, Nussbaum, Sharir, SODA 2012]

We saw that we can compute η_i and the values of φ_i at the endpoints of C^* in $O(\sqrt{n}\log^2 n)$ time, So all iterations take $O(n\log^2 n)$

After all iterations are done, we have the explicit part of the flow η and we know the circulation only on C. We need the circulation in the entire graph.

we know that $\exists$ circulation θ s.t. $\theta+\eta$ is the flow f' we're looking for. Find θ using one SP in $G^*_{f+\eta}$ in $O(n\log^2 n)$ ($f+\eta$ does not respect capacities

(In fact, this last step can be done using just non-negative length using one more iteration of FR-Dijkstra and an additional regular Dijkstra computation - See problem set)

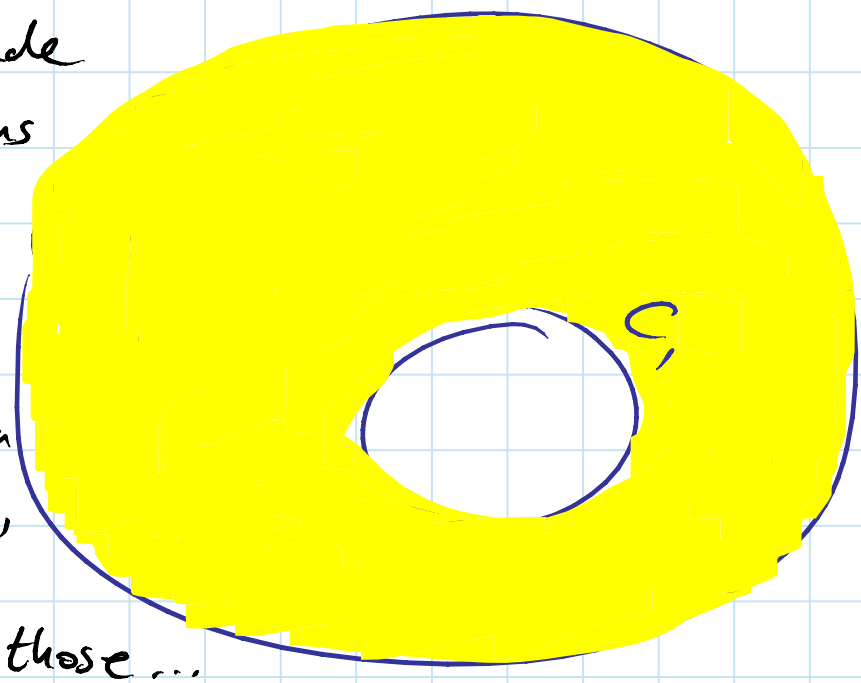
Finally we've established that we can convert the recursive solution f into a maximum pseudoflow $f+f'$ in $O(n\log^2 n)$ time. Convert into a maximum flow in additional $O(n)$ time as discussed in the beginning of lecture.

$O(\log n)$ recursive levels, each in $O(n \lg^2 n)$

so total running time is $O(n \log^3 n)$.

Back to high level recursive solution.
We said we were solving a more complicated problem, but only showed how to find max flow.

Consider the situation inside the recursive call.
Suppose we've just made sure no residual paths between + and − on C_2.
There might still be residual paths between + on C_2 and − on C_1 (or vice versa).
Need to eliminate those...



Solution: send excess flow from +'s on C_1 to C_2 and from C_2 to −'s on C_1. Can be done using st max-flow followed by one more cycle fixing procedure.

How many cycles can there be in a single piece deep in the recursion tree?

$\Omega(\log n)$ if we are not careful.

However, can make sure only a (small) constant of cycles by alternating the weight function in the application of Miller's cycle separator between number of nodes and number of cycles (same idea as in the problem set on r-divisions with a constant number of holes per piece).

Since pushing flow from C to each of the other cycles can be done in $O(n \log^2 n)$ time, if we make sure the number of cycles is constant, the total running time is still $O(n \log^3 n)$.

So exact recursive problem is:

Input: G, S, T, capacities c, a set of constant number of cycles $\{C_i\}$

Output: A pseudoflow f obeying conservation everywhere except $S, T, \{C_i\}$ s.t.

$$S \xrightarrow{G_f} T \quad S \xrightarrow{G_f} \{C_i\} \quad \{C_i\} \xrightarrow{G_f} T$$

Ref:

Borradaile, Klein, Mozes, Nussbaum, Wulff-Nilsen, FOCS 2011.