Maximum Flow Continued

- Let’s pick up from where we left off last time.
- We’re given a plane graph $G = (V, D)$ with two designated vertices $s$ and $t$ along with a capacity function $c : D \rightarrow R^+$ on the darts. We want to compute the maximum $s,t$-flow.
- Venkatesan ['87] shows how to find an $s,t$-flow of value $\lambda > 0$ if one exists:
  - Let $P$ be an arbitrary $s,t$-path. Let $\pi(d) =$
    - $1$ if $d$ in $P$
    - $-1$ if $d$ in $\text{rev}(P)$
    - $0$ otherwise
  - Define the residual capacity of a dart as $c_\lambda(u \rightarrow v) = c(u \rightarrow v) - \lambda \cdot \pi(u \rightarrow v)$. Let $G^*_\lambda$ denote the dual residual network that treats these residual capacities as dart lengths.
  - An $s,t$-flow of value $\lambda$ exists if and only if $G^*_\lambda$ has no negative cycle.
- Suppose there isn’t one. Let $\text{dist}_\lambda(f^*)$ be the distance from $o$ to $f^*$ in $G^*$ with regard to $c_\lambda$ and define $\text{slack}_\lambda(p \rightarrow q) := \text{dist}_\lambda(p) + c_\lambda(p \rightarrow q) - \text{dist}_\lambda(q)$.
- Define $\phi_\lambda(p \rightarrow q)$
  - $:= \text{dist}_\lambda(q) - \text{dist}_\lambda(p) + \lambda \cdot \pi(p \rightarrow q)$
  - $= c(p \rightarrow q) - \text{slack}_\lambda(p \rightarrow q)$.

- How do we find the maximum good value for $\lambda$?
- We’ll use a strategy similar to last Monday’s. We’ll start with $\lambda = 0$ and compute the dual shortest path tree. Then we’ll continuously increase $\lambda$ as we compute so-called parametric shortest paths.
- $\text{dist}_\lambda$ and $\phi_\lambda$ will vary continuously as we increase $\lambda$, but shortest path tree $T_\lambda$ will change at discrete pivots just like before.
- And just like before, we’ll wait until just before some $\text{slack}_\lambda(p \rightarrow q) < 0$, pivot $p \rightarrow q$ into $T_\lambda$, and pivot $\text{pred}(q) \rightarrow q$ out.
- Define $\text{slack'}_\lambda(p \rightarrow q) :=$ derivative of slack for $p \rightarrow q$ and $\text{path}_\lambda(p) :=$ the shortest path in $T_\lambda$ to $p$
- $\text{slack'}_\lambda(p \rightarrow q)$
  - $= \text{dist'}_\lambda(p) + \pi(p \rightarrow q) - \text{dist'}_\lambda(q)$
  - $= -\pi(\text{path}_\lambda(p)) - \pi(p \rightarrow q) - \pi(\text{rev}(\text{path}_\lambda(q)))$
  - $= -\pi(\text{cycle}(T_\lambda, p \rightarrow q))$
  - in $\{-1, 0, 1\}$
Also, we see \( \text{slack}'_\lambda(p \to q) = -\text{slack}'_\lambda(\text{rev}(p \to q)) \)

We’ll call \( p \to q \) active if \( \text{slack}'(p \to q) = -1 \).

But how do we find the active darts?

Let \( L_\lambda = (G \setminus T_\lambda)^* \), the complementary spanning tree of \( G \).

Assuming shortest paths are unique, \( L_\lambda \) is the set of loose edges, those where both darts have \( \text{slack} > 0 \).

\( \text{LP}_\lambda := \text{unique path from } s \text{ to } t \text{ in } L_\lambda \)

Lemma: \( d^\star \) is active if and only if \( d \) in \( \text{LP}_\lambda \)

\( \text{Darts of } T_\lambda \text{ have slack } = 0, \text{ so } \text{slack}' = 0 \text{ for them and their reversals.} \)

\( d^\star \) is active if and only if \( \pi(\text{cycle}(T_\lambda, d^\star)) = 1 \) if and only if \( \text{C}(d) = (\text{cycle}(T_\lambda, d^\star))^* \) is an \( s,t \)-cut

If \( d^\star \) is active, then \( \text{LP}_\lambda \) contains at least one edge of \( \text{C}(d) \). But \( d \) is on the only loose edge of \( \text{C}(d) \), so \( d \) in \( \text{LP}_\lambda \)

If \( d \) in \( \text{LP}_\lambda \), then \( \text{C}(d) \) is an \( s,t \)-cut.

So now the algorithm behaves similar to MSSP.

As we increase \( \lambda \), \( \text{slack}_\lambda(d) \) decreases for all \( d \) in \( \text{LP}_\lambda \) and increases for all \( d \) in \( \text{rev}(\text{LP}_\lambda) \).

Equivalently, \( \phi_\lambda(d) \) increases along \( \text{LP}_\lambda \) and decreases along \( \text{rev}(\text{LP}_\lambda) \).

It’s like we’re pushing flow from \( s \) to \( t \) along \( \text{LP}_\lambda \).

Pivot \( d \) into \( T_\lambda \) when \( \text{slack}_\lambda(d) = 0 \). It’s like we saturated the dart.

Pivot \( \text{pred}(q) \to q \) out to \( T_\lambda \) at the same time. It’s like we made a new augmenting path!

Eventually, we do a pivot that creates a directed cycle of slack 0 darts.

But that means each of those darts is saturated. We found the minimum \( s,t \)-cut and the maximum \( s,t \)-flow!

Pseudocode:

- Create initial dual shortest path tree \( T_0 \)
- Maintain primal spanning tree \( L \)
- While \( L \) is connected
  - \( \text{LP} \leftrightarrow \text{path from } s \text{ to } t \text{ in } L \)
  - \( (p \to q) \leftrightarrow \text{min slack edge in } \text{LP}^* \)
  - decrease slacks along \( \text{LP} \) by \( \text{slack}(p \to q) \). Increase slacks along \( \text{rev}(\text{LP}) \) by the same amount.
  - delete \( (p \to q)^* \) from \( L \)
  - insert \( (\text{pred}(q) \to q)^* \) into \( L \)
  - \( \text{pred}(q) \leftrightarrow p \)
- \( \phi \leftrightarrow c - \text{slack} \)
Analysis

- Using dynamic forests, each step of the algorithm can be implemented in $O(\log n)$ time. Our running time therefore depends on the number of pivots.
- Let $\text{path}_i$ denote the shortest walk from $o$ to $q$ s.t. $\pi(\text{path}_i(q)) = i$.
- We see $\text{dist}_\lambda(q) = \min_i (c_\lambda(\text{path}_i(q))$.
- We can also observe that whenever $\text{path}_\lambda(q)$ changes, $\pi(\text{path}_\lambda(q))$ increases by 1, because $\pi(\text{cycle}(T, p \to q)) = 1$ if $p \to q$ is pivoting in.
- So now we want to know, for which $i$ is $p \to q$ in $\text{path}_i(q)$?
- Imagine removing faces $s^*$ and $t^*$ from the plane. We now have a sphere with two boundary, also known as an annulus.
- We’ll define something called the universal cover as follows: Imagine cutting along that path $P$ from earlier, turning our annulus into a disk. Now make a doubly infinite sequence $..., G^*_{-1}, G^*_{0}, G^*_1, G^*_2, ...$ of copies of $G^*$ and paste them together along them respective copies of $P$.
- Formally, its a plane graph $Gbar^* = (Vbar^*, Ebar^*)$ where $Vbar^* = \{p_i \mid p \in V^* \text{ and } i \in \mathbb{Z}\}$ and $Ebar^* = \{p_i \to q_{i + \pi(p \to q)) \mid p \to q \in E^*\}. We also have dart costs $c(p_i \to q_j) = c(p \to q)$.
- Each vertex $p_i$ is a lift of $p$ to $Gbar^*$. There is a projection map $\omega bar(Gbar^* \to G^*)$ that drops subscripts so $\omega bar(p_i) = p$ and $\omega bar(p_i \to q_j) = p \to q$.
- The preimage $\omega bar^{-1}(P_i)$ for any path $P_i$ in $G^*$ is a doubly-infinite set of paths in $Gbar^*$ called the lifts of $P_i$. If $P_i$ starts at $p$ and ends at $q$, then for any $i$, there is a lift of $P_i$ from $P_i$ to $q_{i + \pi(P_i)}$.
- $s^*$ and $t^*$ lift to two unbounded faces $sbar^*$ and $tbar^*$, and every other face lists to an infinite sequence of faces.

So now for the punchline. Each $\text{path}_i(q)$ in $G^*$ lifts to a shortest path to $q_0$ from $o_{-i}$.
- The $i$ for which $p \to q$ is in $\text{path}_i(q)$ are the set of $i$ for which the shortest path from $o_{-i}$ goes through $p_{-\pi(p \to q)} \to q_0$.
- But we saw on Monday that these $i$ are contiguous!
- So $p \to q$ pivots into $T_\lambda$ at most once, and it leaves at most once.
- Which implies $O(n)$ pivots.
• Which implies an $O(n \log n)$ running time!
• Some quick notes:
  • Erickson [’10] described this formulation of the algorithm based on parametric shortest paths.
  • His algorithm is essentially identical to one by Borradaile and Klein [’09]. However, they describe things mostly in the primal graph, sending flow along LP each iteration to always have a “leftmost flow” of each value lambda. Their analysis is much more complicated, because they focus on how often you can saturate each dart with this primal flow.

**Surface Maps**

• It’s finally time to leave the plane. Let’s do that by going back to something we saw before.
• Recall a rotation system can be described as a triple of permutations $\text{succ}$, $\text{rev}$, and $\text{next}$ from darts to darts (Erickson now calls these $\text{vnext}$, $\text{rev}$, and $\text{fnext}$ so make it easier to remember which is which. Maybe we should start doing that too?)
  • $\text{rev}$ is an involution w/o fixed points
  • $\text{fnext}$ [i.e. next] = $\text{rev} \circ \text{vnext}$
  • orbits of $\text{vnext}$ go ccw around vertices at head
  • orbits of $\text{rev}$ are edges
  • orbits of $\text{fnext}$ go cw around faces to right
• $V - E + F = 2$ in connected planar graphs, but what if they equal something else?
• We still have an embedding… it’s just not planar!
• Imagine every face (orbit of $\text{fnext}$) as a polygon. Label the sides of these polygons with the names of distinct darts. This construction is called the *polygonal schema* of the embedding.
• Each edge appears twice on the boundary of the polygons.
• Glue the polygons together at their corresponding darts by identifying each $\text{rev}(d)$ with the reversal of dart $d$.

![Diagram of surface maps](image)

• If you look at a sufficiently small neighborhood around each point after gluing
you’ll see these neighborhoods are all homeomorphic to the plane.

- What we’ve created is an orientable compact 2-manifold. Otherwise known as a surface.

- There are infinitely many of these things, and they’re distinguished (up to homeomorphism) by their genus.

- Intuitively, the genus is the number of handles you glue onto a sphere to get the surface, but sometimes it’s hard to tell what the handles are.

- Formally, the genus is the maximum number of disjoint simple closed curves on the surface whose complement is still connected.

Trees, Co-trees, and Formulas

- Remember, if G is planar and T is a spanning tree, then \((E \setminus T)^*\) is a spanning tree of \(G^*\). But that’s no longer true when genus > 0.

- An isthmus is any edge with the same face on both sides. We may have isthmuses now that are not bridges!

- We can contract any edge that’s not a loop without messing with the faces. By duality, we can delete any edge separating distinct faces.

- Take any rotation system, and let \(g\) be the genus of the underlying surface.

- Suppose we contract non-loops until there’s a single vertex. The \(V - 1\) edges we contracted form a spanning tree \(T\), but the faces remain intact.

- Now delete non-isthmuses until only one face remains. The \(F - 1\) deleted edges form a dual spanning tree.

- But now there are \(L\) leftover edges that are all both loops and isthmuses. Counting the edges, we see \(V - E + F = 2 - L\).

- These remaining edges form what is called a system of loops, a surface map with one vertex (the basepoint) and one face homeomorphic to an open disk.

- But how are \(L\) and \(g\) related? To find out, we’re going to do something previously forbidden. We’re going to contract a loop!
Let's recall how to modify the rotation system to contract a non-loop edge.

\[ \text{vnext}(\text{vprev}(d)) \rightarrow \text{vnext}((\text{rev}(d))) \text{ and } \text{vprev}(\text{vnext}(d)) \rightarrow \text{vprev}(\text{rev}(d)) \]

But if we do those operations to a loop, we split the vertex.

And if we delete an isthmus, by duality we split a face.

So what would this look like on the actual surface?

Well, you can't get from one side of the loop to the other anymore, so it's like we cut a handle. Here's another way to imagine what happened:

The result is that we now have genus \( g - 1 \) but still only one face.

The face uses both vertices, so there is at least one non-loop edge. Contract it as well.

Now we have a new system of \( L - 2 \) loops on a surface of genus \( g - 1 \).

Now we just need to figure out some base cases, and we'll be done.

If \( L = 0 \), then we have the trivial map of a single vertex on a sphere.

If \( L = 1 \), then we have two darts \( d \) and \( \text{rev}(d) \).

- So \( \text{vnext}(d) = \text{rev}(d) \), implying \( \text{fnext} \) is the identity function. But that means we have two faces. There is no system of one loop (on an orientable surface).
- So by induction, we conclude \( V - E + F = 2 - 2g \).