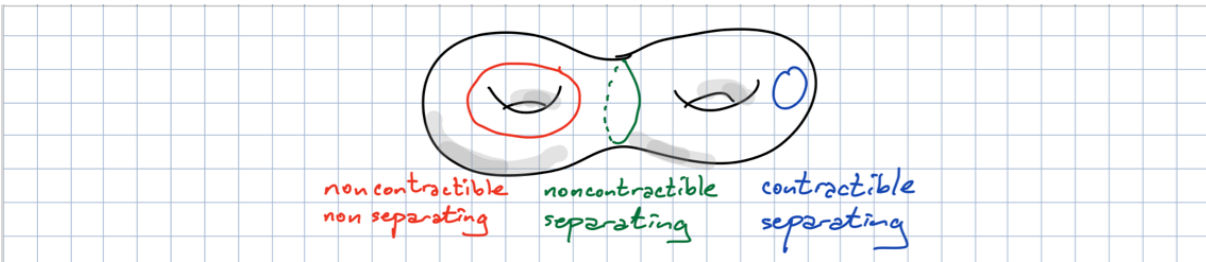
CS 7301.003.20F Lecture 15—October 7, 2020

Main topics are #non-trivial_cycles.

Non-trivial Cycles

- Many algorithms for surface graphs start by cutting the underlying surface to reduce topology:
  - Example: Planar minimum cut
  - Example: Find a cutgraph to act as the seam in a texture map
- Today, we're going to talk about two kinds of cycles that are potentially useful to cut along and algorithms for finding shortest examples of those cycles.
- We'll say a cycle is *trivial* if it is either of the following:
  - contractible—homotopic to a point—if simple, it is the boundary of a disk
  - separating / null-homologous / boundary—technically defined as being *homologous* to the empty cycle; we'll define that later. Also form the boundary of a subset of faces, and if simple, cutting along the cycle leaves the surface disconnected



- Our goal: Give a surface graph with edge weights ell : E ➔ R^+, find the shortest cycle whose image is non-contractible or non-separating. While it won't matter in the end, I'm using the word cycle here the topological sense. The problem itself does not require simplicity.

Thomassen's 3-path Condition

- We'll start with an algorithm by Thomassen ['90].
- Let alpha, beta, and gamma be three paths with the same endpoints.
- Lemma: If alpha dot rev(beta) and beta dot rev(gamma) are contractible (or separating), so is alpha dot rev(gamma).
  - alpha ~= beta ~= gamma in the first case.
  - In the second, the first two cycles must bound some faces, and the third cycle bounds the symmetric difference of the first two sets.
- Any "trivial" class of cycles for which the lemma statement apply are said to have the *3-path condition*.
- Lemma: Let s lie on a shortest non-trivial cycle and let T_s be its shortest path tree. There is

a shortest non-trivial cycle sigma consisting of the path in T_s from s to x, an edge xy, and the path in T_s from y from s.

  - Let sigma be a shortest non-trivial cycle containing s that contains the most edges possible in common with a shortest path out from s.
  - Let x be the endpoint of that shortest path and y the next vertex along sigma.
  - Let alpha be the path in T_s from s to x followed by xy and gamma be the path in gamma from s directly to y.
  - Suppose gamma is not in T_s, and let beta be the path in T_s from s to y. If alpha dot rev(beta) is non-trivial, we are done.
  - Otherwise, beta dot rev(gamma) must be trivial by our definition of sigma and x. Therefore, alpha dot rev(gamma) is also trivial, a contradiction!
- And now we have a simple algorithm:
  - For each vertex s
    - Compute the shortest path tree T_x
    - For each edge e not in T_x
      - Check if cycle(T_x, e) is trivial
  - Return shortest non-trivial cycle found.
- Assuming the graph is sparse (m = O(n)), we can check if a cycle is trivial in O(n) time by slicing it open and seeing what lies on each side. The whole algorithm takes O(n^3) time.

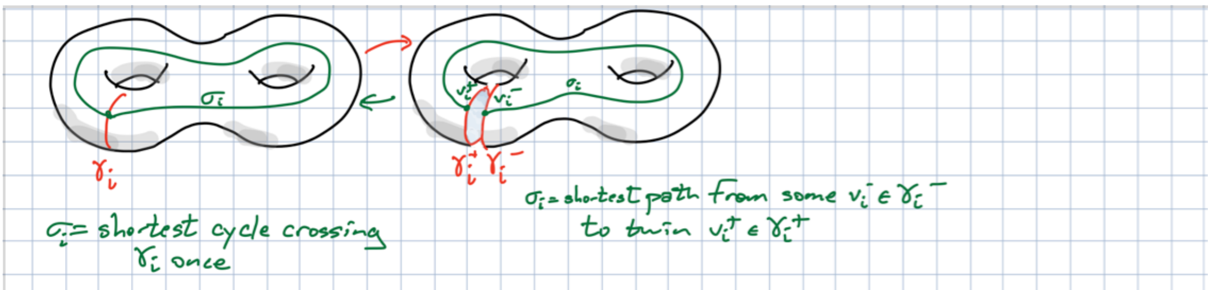
Greedy Tree-cotree Decomposition

- But can we do it faster?
- Consider the following tree-cotree decomposition (T_s, L, C):
  - T_s : shortest path tree rooted at some vertex s
    - Can compute in O(n log n) time using Dijkstra or O(n) time using Henzinger et al.
  - C^* : maximum spanning tree of (G \ T_s)^* where w(e^*) := length(loop(T, e))
    - This length counts repeated edges on the loop twice. We can compute each weight in constant time using the distances from s.
  - L := E \ (C cup T_s)
- Define the *dual cut graph* K^* = C^* cup L^*. It's a subgraph of G^ * with one dual face.
- Create the reduced dual cut graph R^* by repeatedly removing degree-1 vertices (hair) from K^*.
- Lemma:
  - cycle(T_s, e) is separating <=> if K^* \ e^* is disconnected <=> e^* is a bridge in K^*
  - cycle(T_s, e) is contractible <=> it is separating and one component of K^* \ e^* is a tree <=> e^* is a hair in K^* <=> e^* not in R^*
- We can find all hairs in O(n) time. All bridges too [Tarjan '74].
- So we really only need O(n) time to find shortest non-trivial cycle through each vertex.

That's O(n^2) time overall [Erickson-Har-Peled '03; Cabello at al. '16].

Faster via MSSP

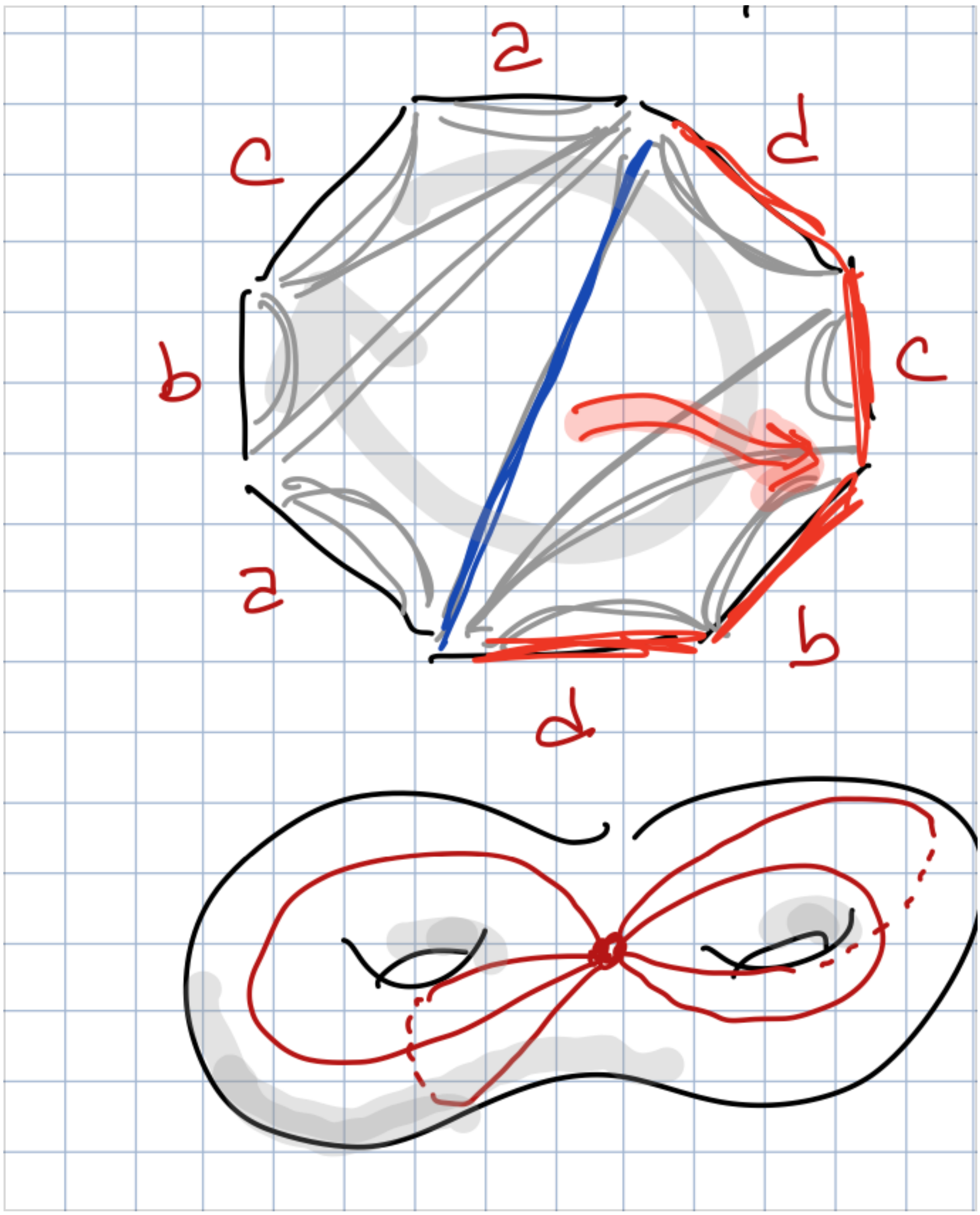
- Let Ell = {loop(T_s, e) | e in L} for any vertex s.
- Ell is a system of loops (and that's true for any tree-cotree decomposition).
- In this case, it's actually the shortest system of loop based at s [Erickson, Whitttlesey '05; Colin d Verdi`ere '10] and is often called the *greedy system of loops*.
- Let Gamma = {cycle(T_s, e) | e in L}. Gamma is a *greedy system of cycles*.
- Using a similar exchange argument to what we saw earlier, you can show that for any shortest path, there is a shortest non-trivial cycle that doesn't cross it.
- But Gamma is especially useful for one of the problems we're focusing on today.
- Lemma:
  - Every non-separating cycle crosses some cycle in Gamma at least once. (Proven using homology).
  - Some shortest non-separating cycle crosses each cycle in Gamma at most once. (Proven using exchange arguments.)
- So, here's what we'll do.
- Compute the greedy cycles Gamma = {gamma_1, …, gamma_{2g}}
- For i ← 1 to 2g
  - Find the shortest cycle sigma_i that crosses gamma_i exactly once.



- And to do that last step, we'll use a generalized multiple-source shortest paths algorithm of Cabello, Chambers, and Erickson ['13].
- As before, it finds all shortest path distances from sources on a single face. In our case, we use gamma_i^- as our face.
- I won't go into all the arguments, but there are now O(gn) pivots total and we can spend O(g log n) time per pivot for O(g^2 n log n) time doing MSSP. With more care, we can spend O(g n log n) time doing MSSP.
- So we can compute sigma_i in O(g n log n) time, leading to a shortest non-separating cycle in O(g^2 n log n) time total.
- There's another algorithm for shortest non-contractible cycle that also performs O(g) MSSP computations, but it's quite a bit more complicated.

Faster, Faster!

- OK, but what if g is really small. Then what?
- Again, let Ell = {loop(T_s, e) | e in L} for any vertex s.
- Kutz ['06] showed there is a shortest non-trivial cycle that
  1. crosses each loop of Ell at most twice and
  2. never crosses a loop then immediately turns around to cross the same loop from the other side, forming a *curl*
- Each cycle sigma has a *signed crossing sequence* with regard to Ell. Starting from any point on sigma, we record what order *and from what direction* we cross the loops of Ell. Two cycles with the same signed crossing sequence are homotopic (and therefore both trivial or non-trivial). The converse IS NOT TRUE, and testing whether two paths or cycles are homotopic on a surface is surprisingly subtle.
- So, the shortest non-trivial cycle must have a crossing sequence of length O(g), where each character of the sequence takes on one of O(g) values. There are g^O(g) such sequences.
- Here's our new strategy:
  - For each signed crossing sequence of length O(g) and no curls
    - If an arbitrary cycle with that sequence is non-trivial
      - Compute the shortest cycle with that sequence
- Here's how we'll compute that shortest cycle:
- Cut the surface along Ell. It unfolds into a single 4g-gon with two sides per loop.



- Now, make O(g) copies of this 4g-gon and glue pairs of them together. The i-1st and ith pairs are glued along the ith loop in the sequence and the final copy is glued to the 0th again along the final loop in the sequence.
- There are no curls, so each polygon side is shared by either one or two copies of the 4g-gon. We still have a surface. In particular, we had a single boundary component until we glued the 0th and final copies, so we end up with a planar surface with two boundary: an annulus.
- Now we want the shortest cycle separating the two boundary in a graph of complexity O(gn). But we already solved this problem! Run Italiano et al. in O(gn log log n) time!

- We loop over $g^{O(g)}$ crossing sequences, so the whole algorithm takes $g^{O(g)} n \log \log n$ time total.
- With a bit more work, we can reduce our search to cover only $2^{O(g)}$ crossing sequences for $2^{O(g)} n \log \log n$ time total [F ’13].