Algorithms for Surface Graphs

Jeff Erickson
University of Illinois, Urbana-Champaign

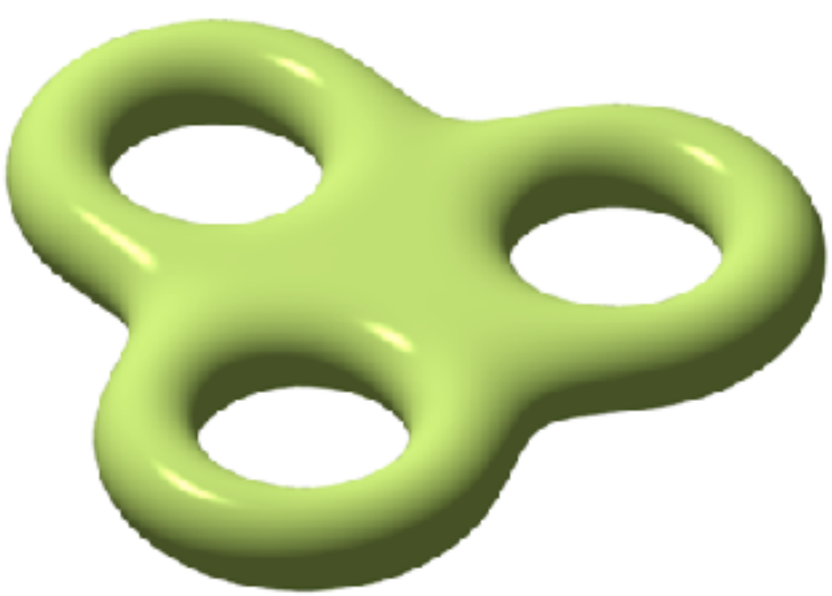
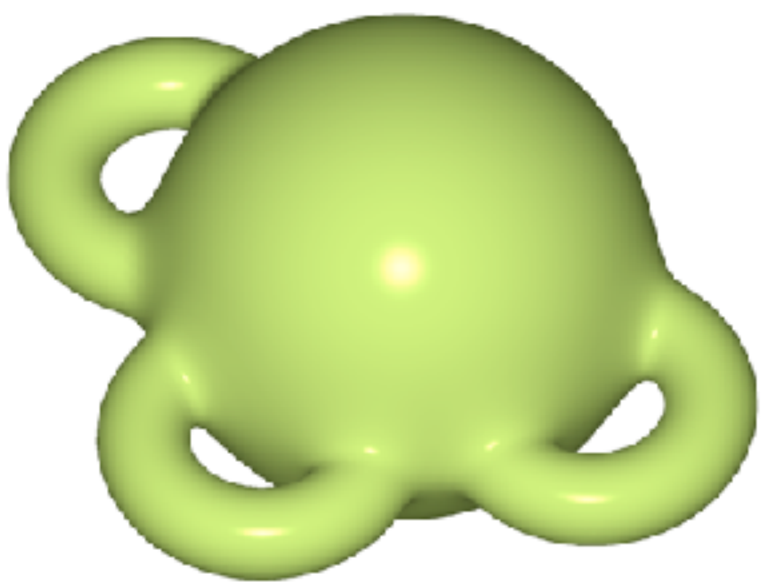
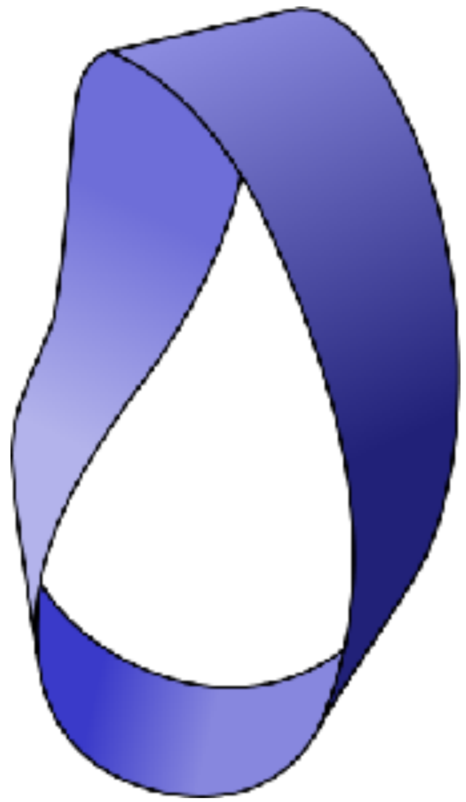
CIMAT, Guanajuato, Mexico
September 11, 2018

Please ask questions!

Surface maps

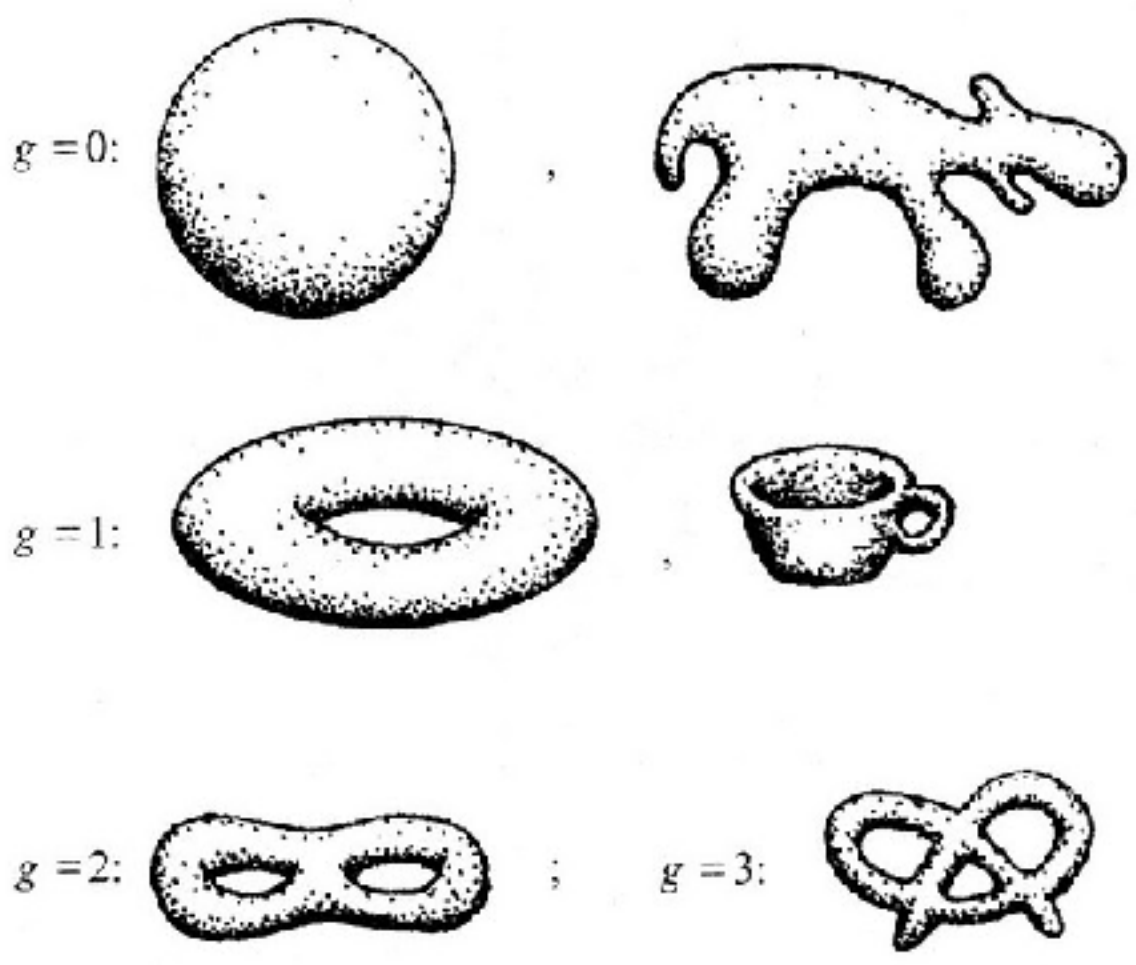
Surfaces = 2-manifolds

- Connected, compact, Hausdorff space in which every point has a neighborhood homeomorphic to the plane.
- An *orientable* surface does not contain a Möbius band.

Surface classification

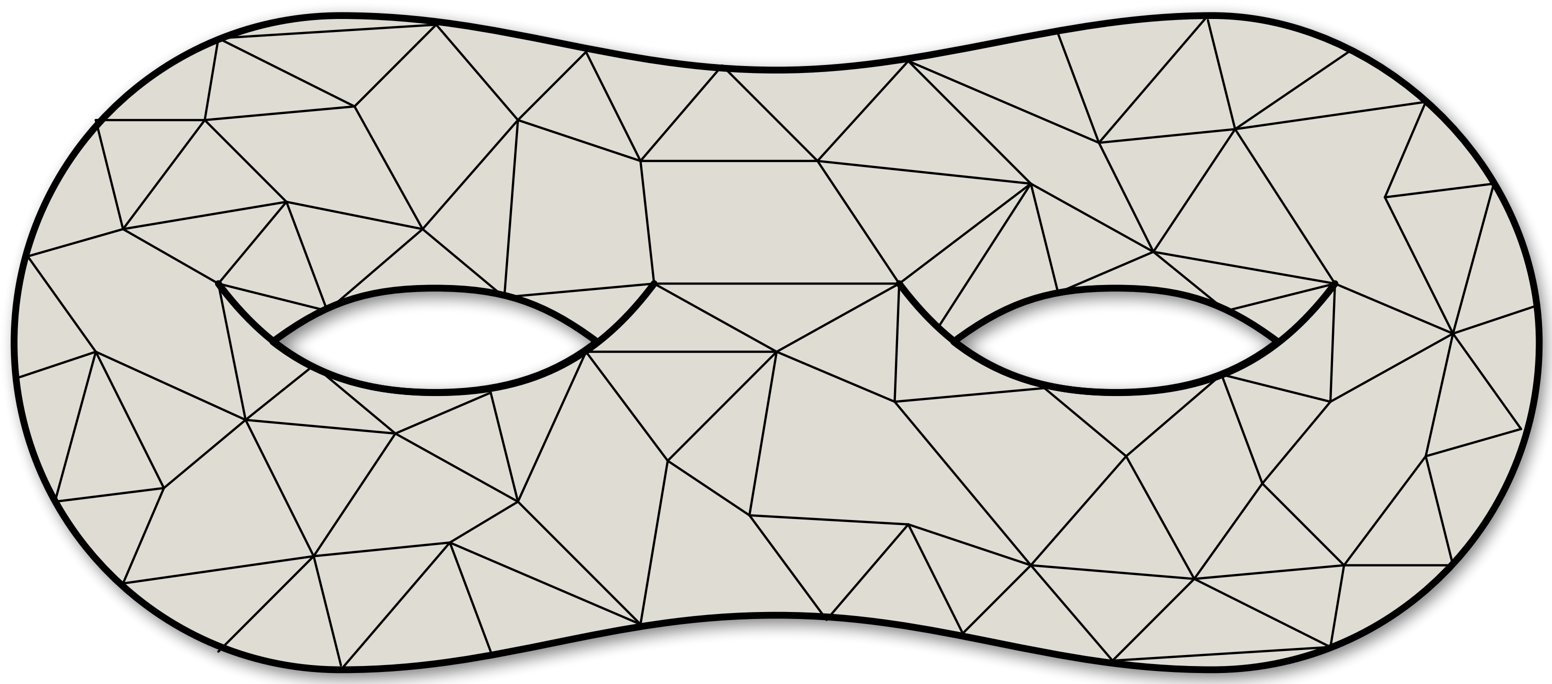
Every orientable surface is homeomorphic to a sphere with g handles, for some integer $g \geq 0$, called its *genus*.



Roger Penrose, The Road to Reality *(2004)*

Surface map

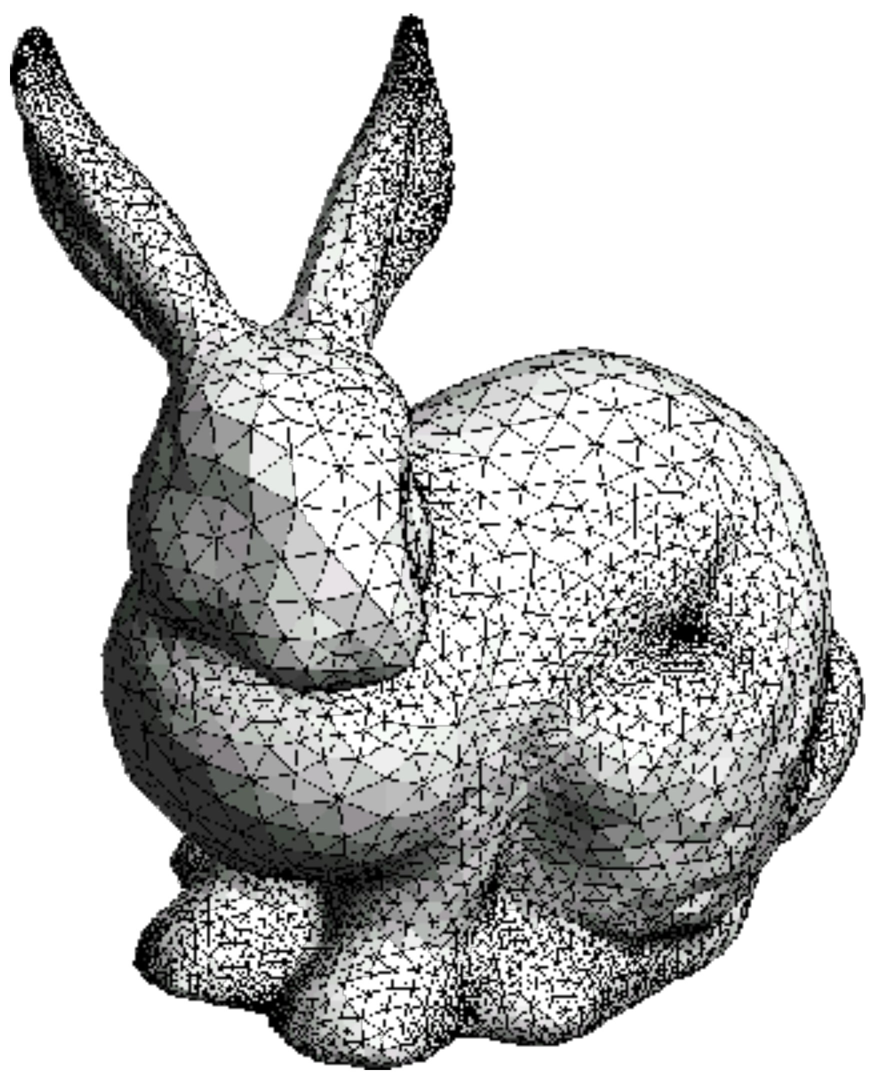
- Graph embedded on a surface so that each face is a disk
- Equivalently: Polygons glued together into a closed surface

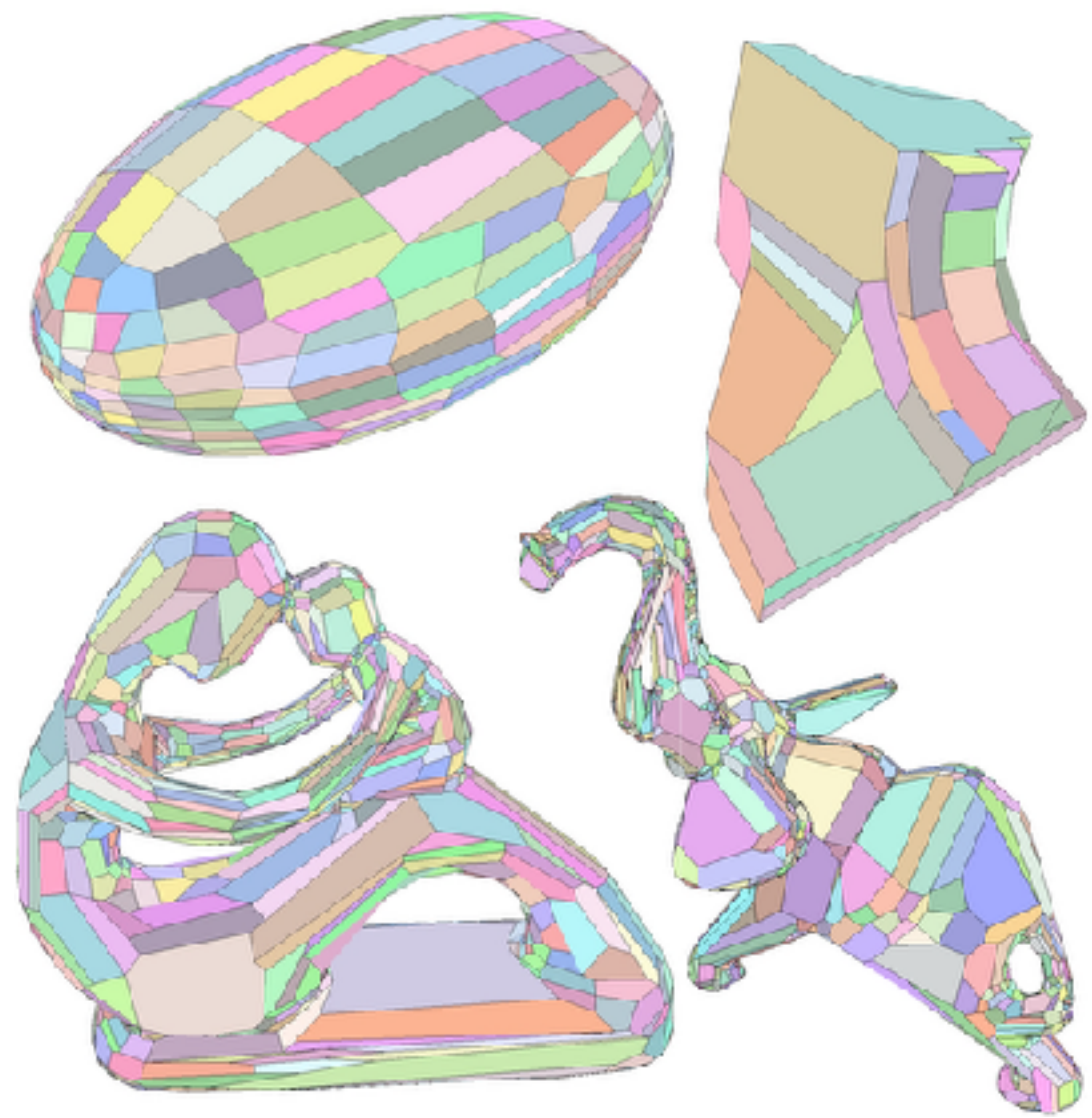


[Riemann 1857; Heawood 1890; Poincaré 1895; Heffter 1898; Dehn Heegaard 1907; Kerékjártó 1923; Radó 1937; Edmonds 1960; Youngs 1963; Gross Tucker 1987 ; Mohar Thomassen 2001; ....]

Surface map

- The standard surface representation in graphics and geometric modeling....

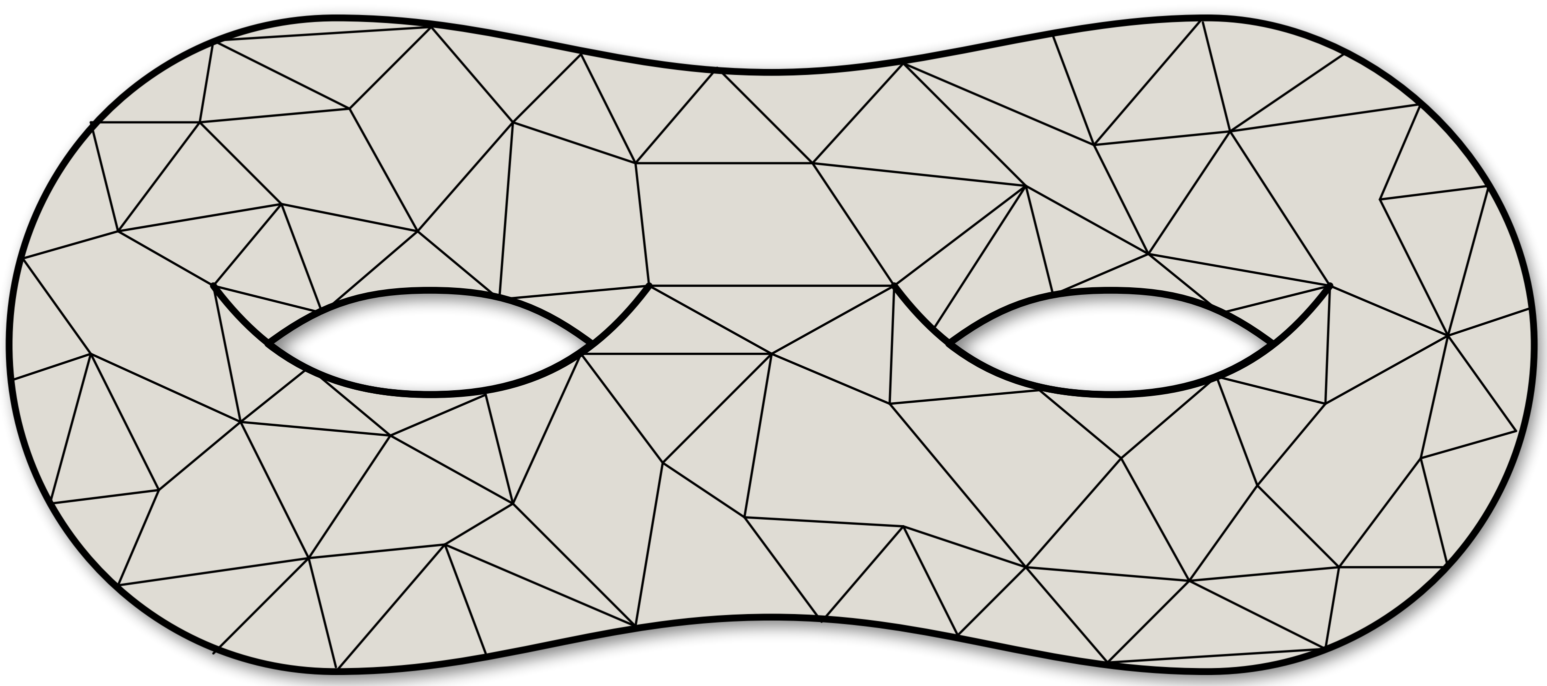




[Pellenard Morvan Alliez ’12]

Surface map

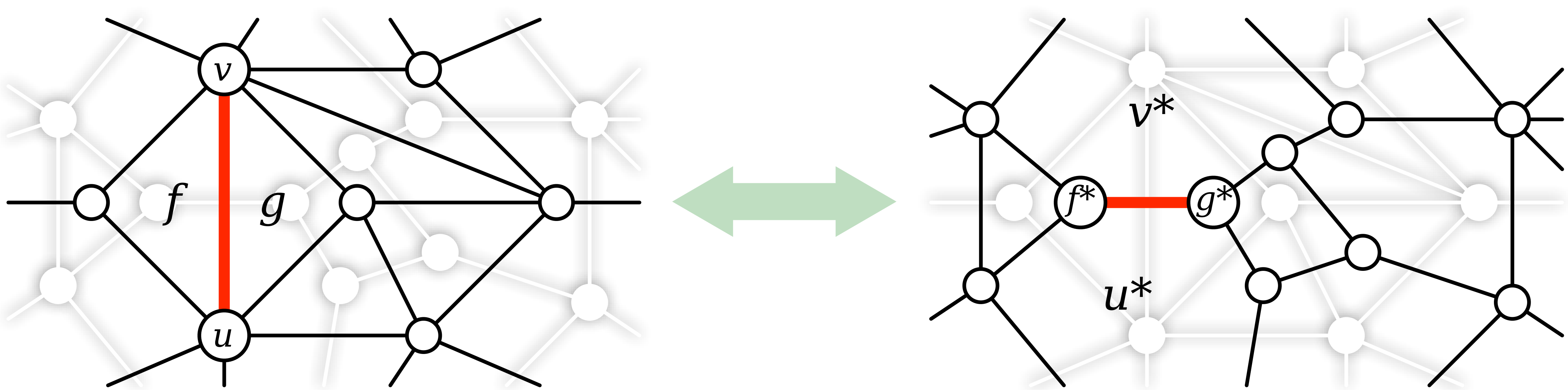
- The standard surface representation in graphics and geometric modeling, *but without vertex coordinates*



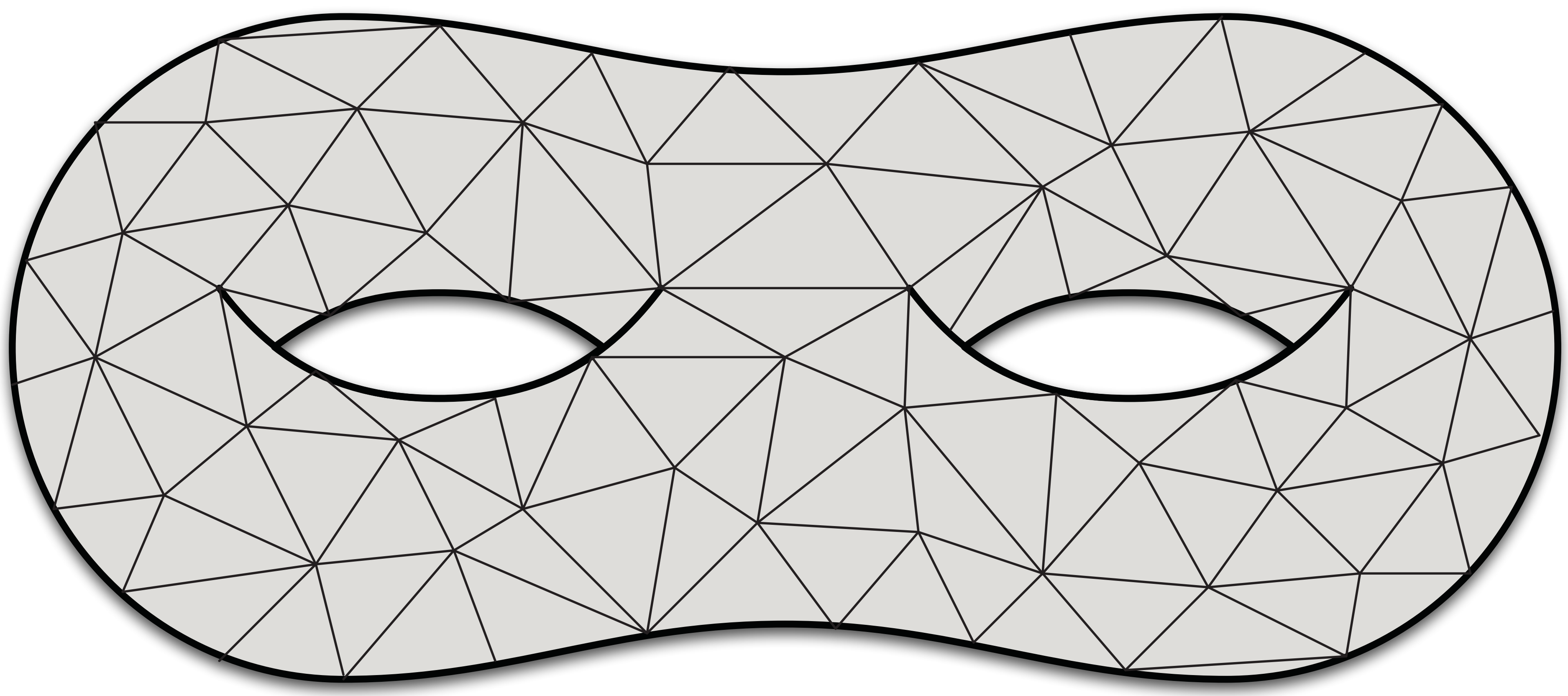
Map duality

Every surface map $\Sigma = (V, E, F)$ has a natural *dual* map $\Sigma^* = (F^*, E^*, V^*)$ on the same surface:

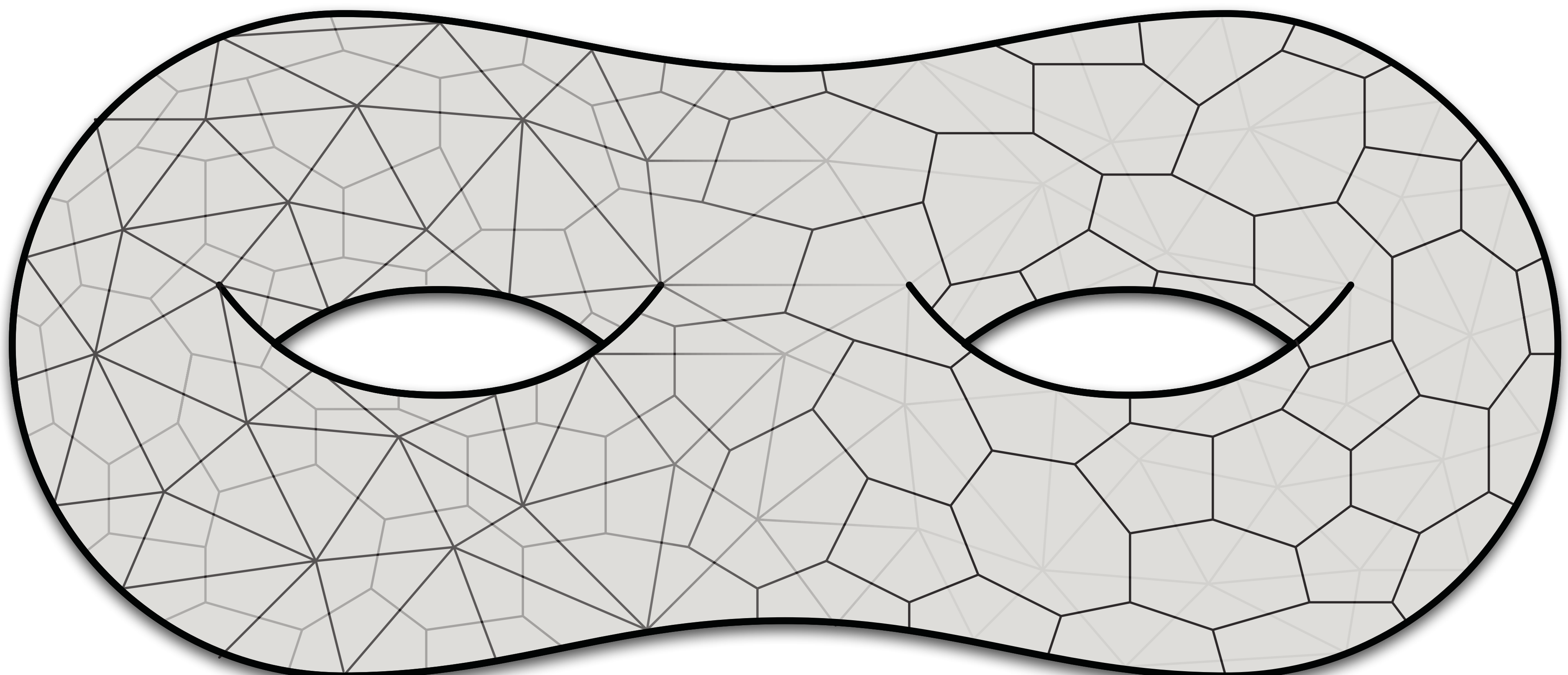
- vertices of Σ^* = faces of Σ
- edges of Σ^* = edges of Σ
- faces of Σ^* = vertices of Σ



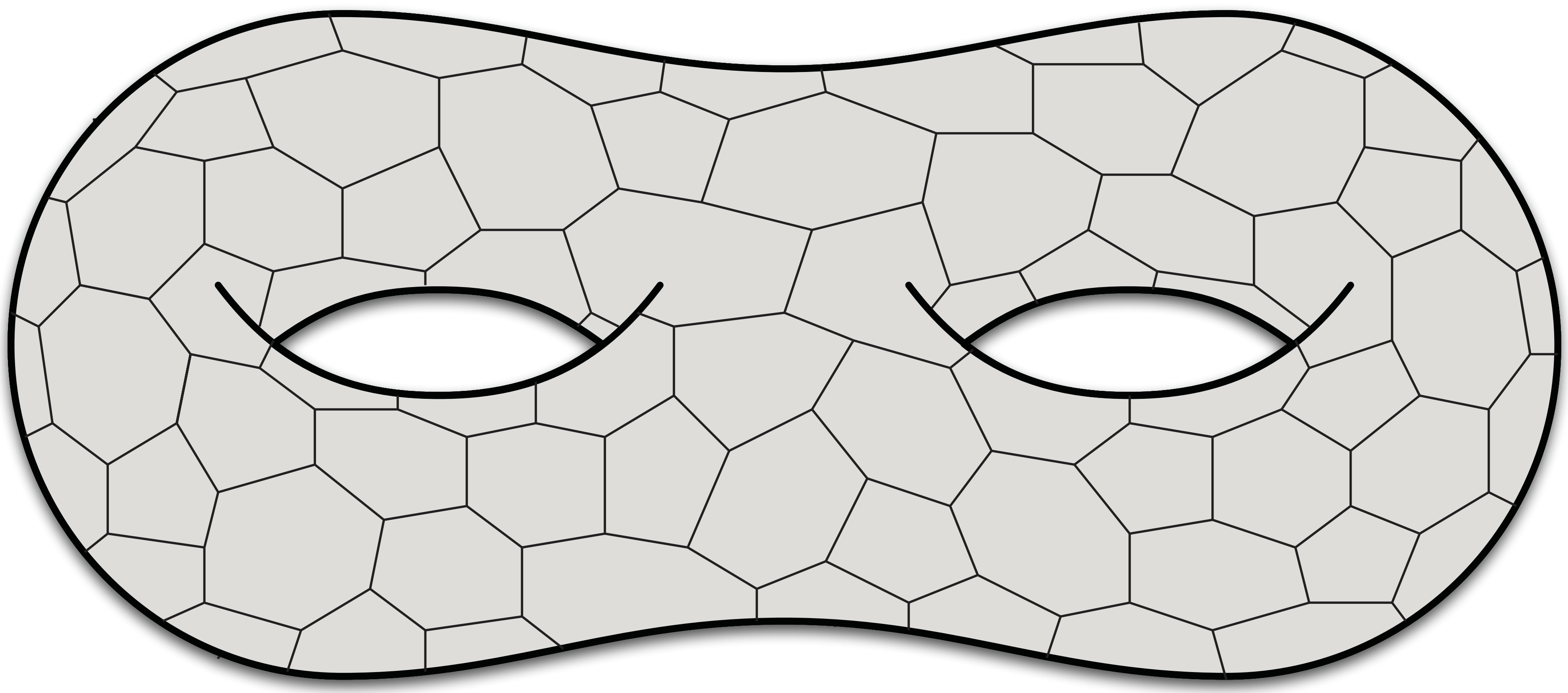
Map duality



Map duality

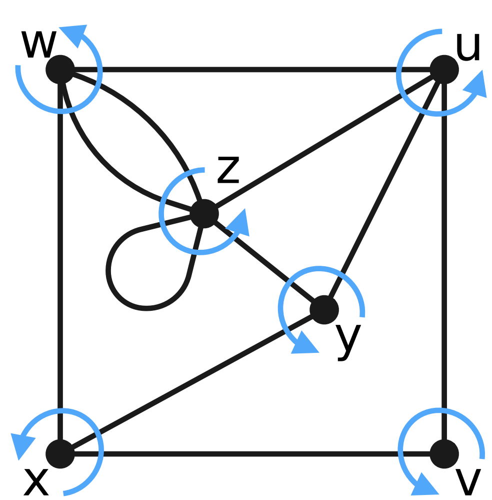


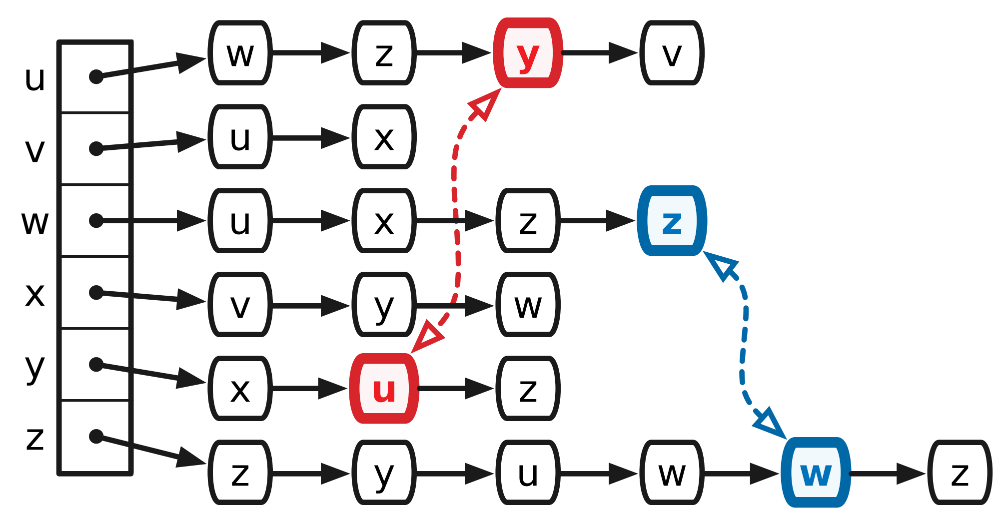
Map duality



Rotation systems

- The cyclic order of edges incident to each vertex completely specifies the surface map (up to homeomorphism).
- We can use standard graph data structures to represent both a surface map Σ and its dual Σ*.





Euler's formula

- For *every* map (V, E, F) on the orientable surface of genus g:

$$\boxed{V - E + F = 2 - 2g}$$

- $\chi := 2-2g$ is the *Euler characteristic* of the map / surface.
- In particular, a map is *planar* if and only if $V - E + F = 2$.

[Descartes c.1630 (via Leibniz 1676 (via Foucher de Careil 1859)),
***Euler 1750**, ~~Euler 1753~~, ~~Karsten 1768~~, ~~Meister 1784~~,*
Legendre 1794, Hirsch 1807, l'Huillier 1811, ~~Cauchy 1811~~,
~~Grunert 1827~~, Von Staudt 1847, Cayley 1861, ~~Listing 1861~~, ...]

Easy consequences

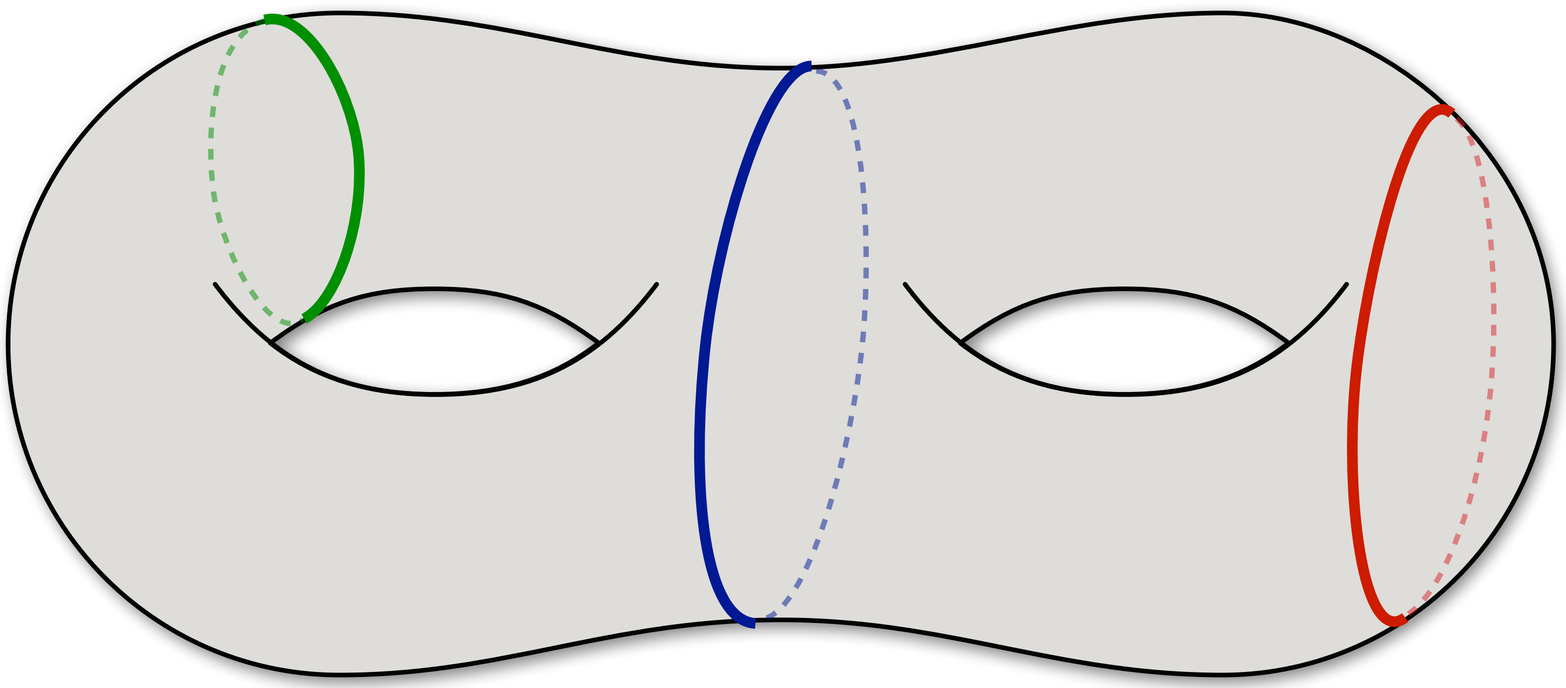
- Surface triangulations: $E = 3V - 6 + 6g$ and $F = 2V - 4 + 4g$
- Simple surface graphs: $E \le 3V - 6 + 6g$ and $F \le 2V - 4 + 4g$
- Typically assume $g = O(V)$, so that $E = O(V)$ and $F = O(V)$
- Every simple surfave graph has vertex of degree $O(1 + g/V)$
  - At most 6 if $g < V/12$
  - Minimum spanning trees in $O(V)$ time if $g = O(V)$

Today's Question

Given a surface Σ, find the shortest *topologically nontrivial* cycle in Σ.

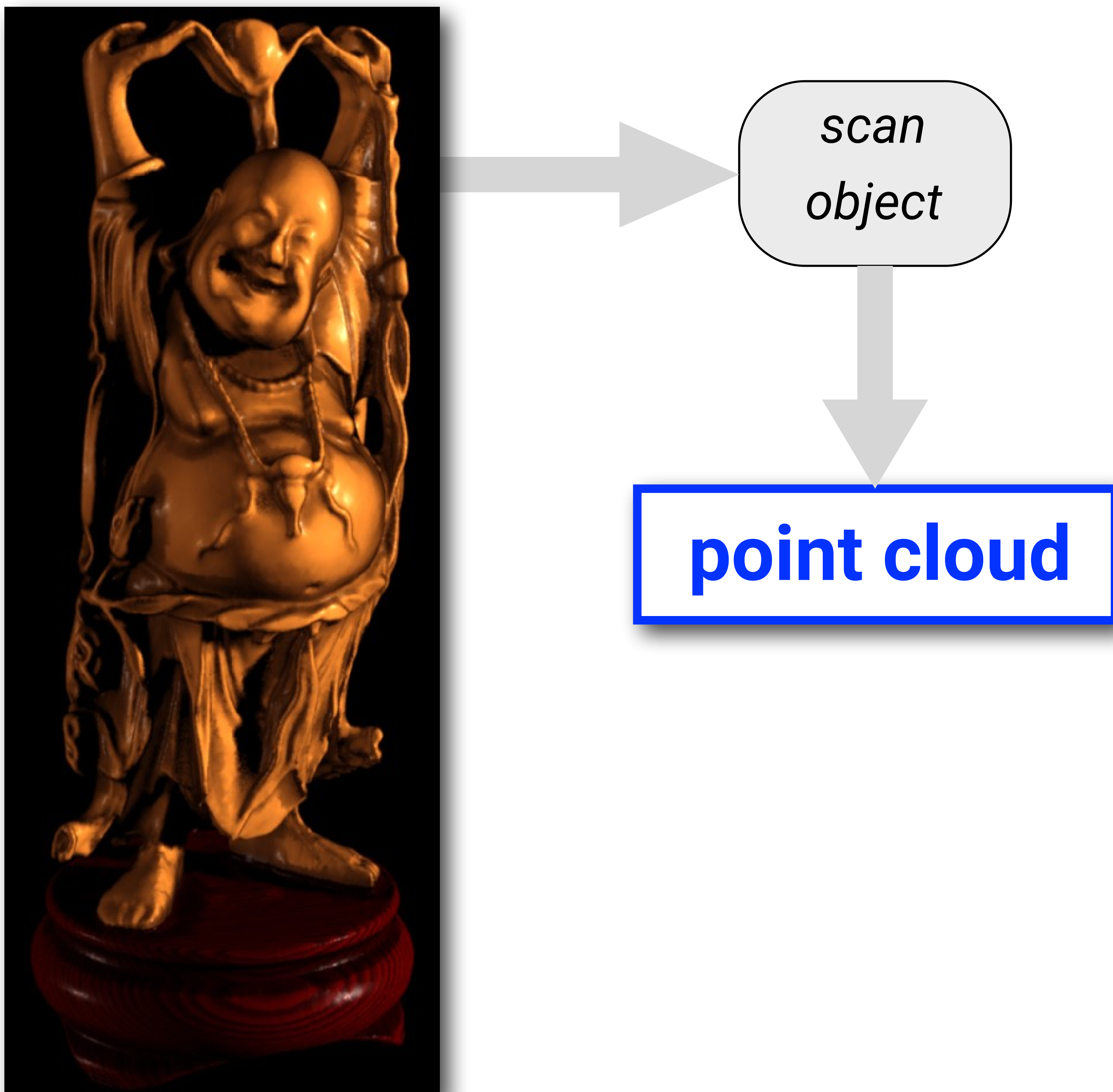
Trivial cycles

- *contractible* = null-homotopic = boundary of a disk
- *separating* = null-homologous = boundary of a subsurface

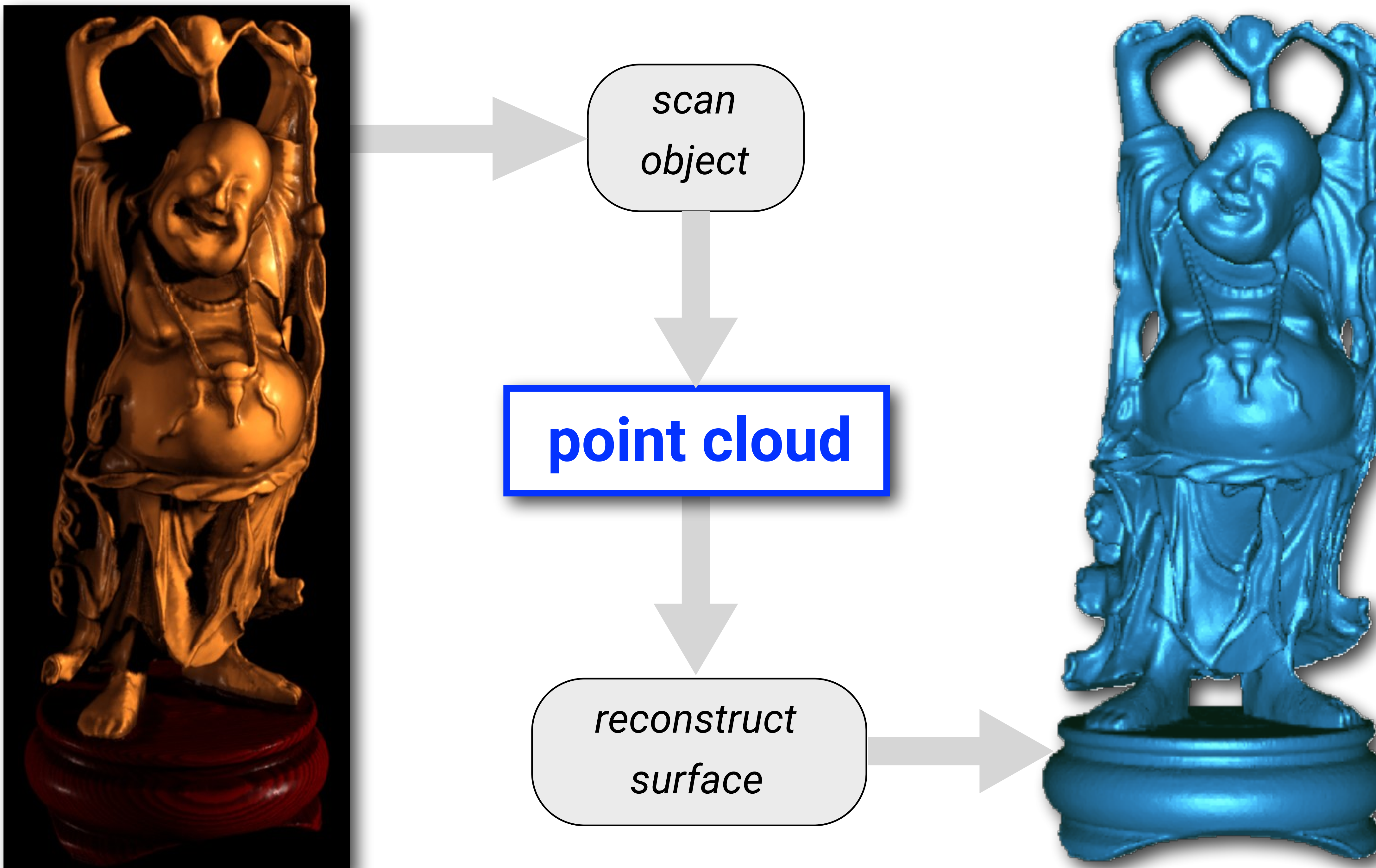


Surface reconstruction

Surface reconstruction

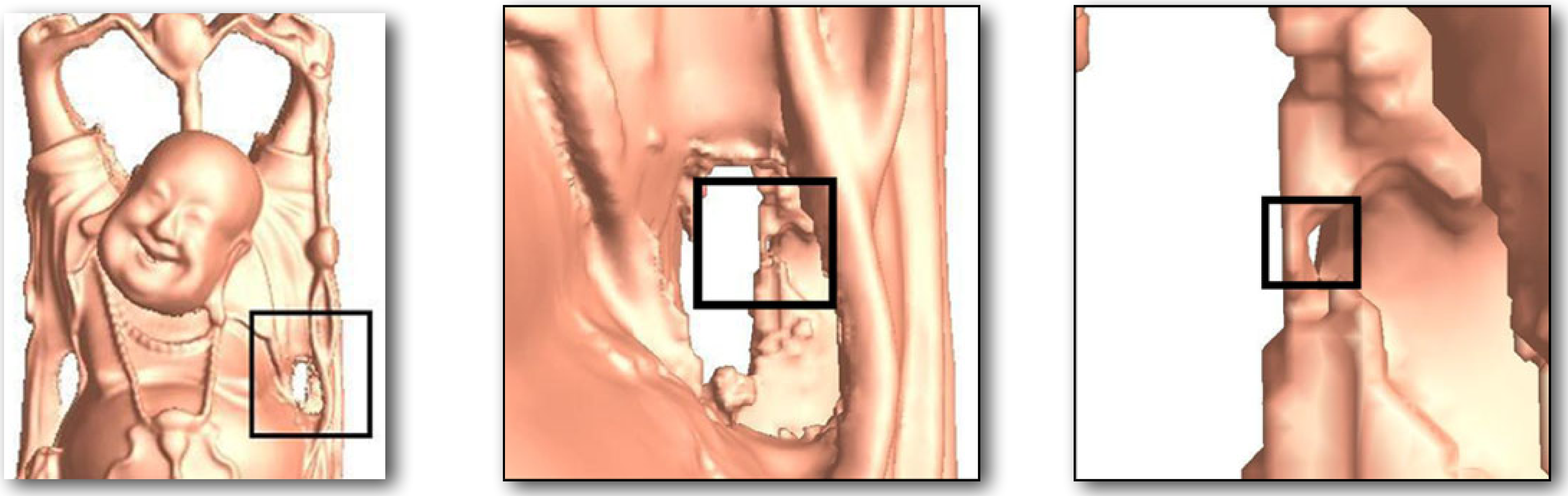


Surface reconstruction



Topological noise

- Measurement errors from the scanning device add extra handles/tunnels to the reconstructed surface.

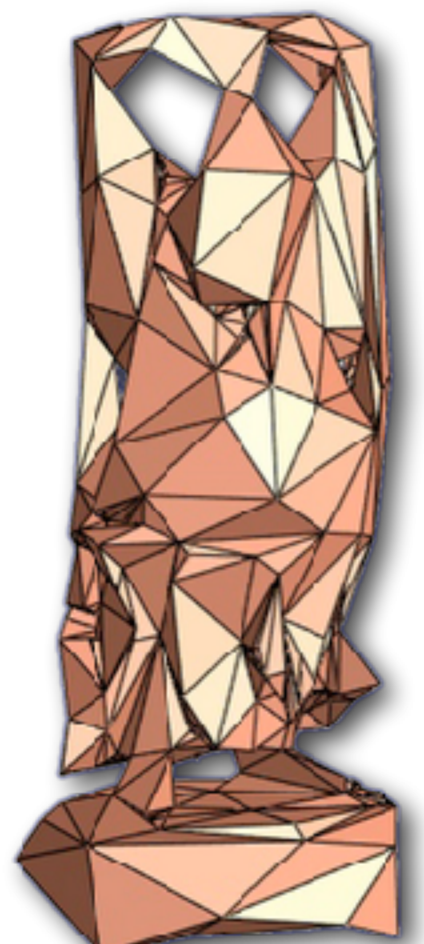


[Wood, Hoppe, Desbrun, Schröder '04]

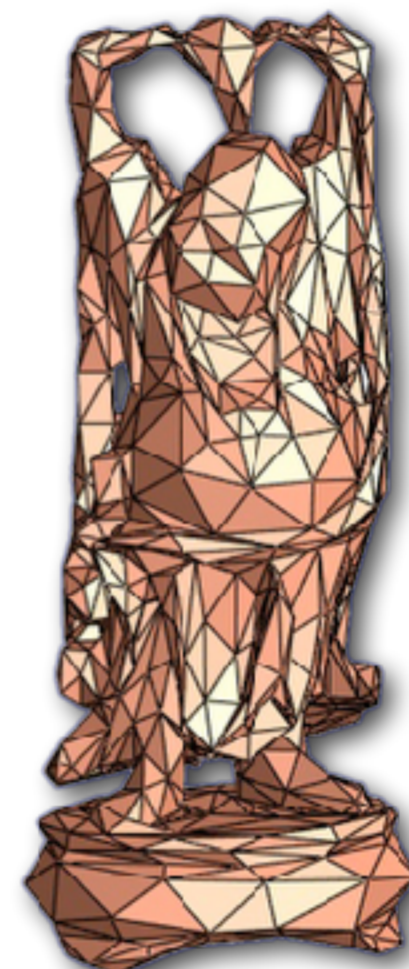
Topological noise

- These extra tunnels make compression difficult.

genus 104



genus 104
50K vertices



genus 6
50K vertices

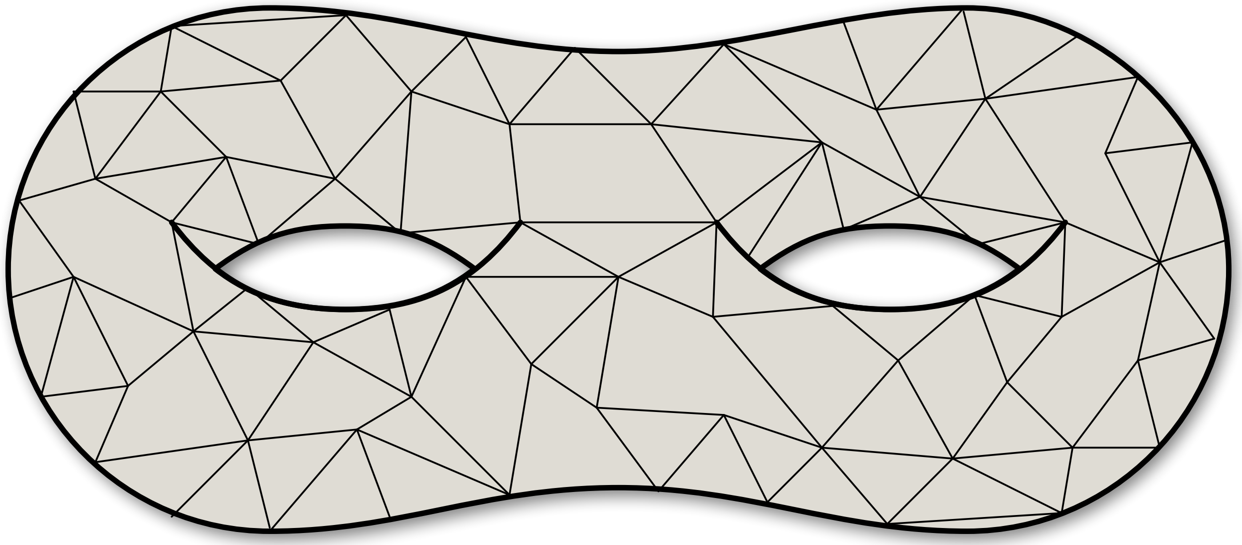
[Wood, Hoppe, Desbrun, Schröder ’04]

Connections

- Length of shortest noncontractible cycle
  - *systole* *[Loewner '49] [Pu '52] ... [Gromov 83] ...*
  - *representativity* *[Robertson, Seymour 87]*
  - *edge-width* *[Thomassen 90; Mohar, Thomassen 99]*
- First step of *many* other topological graph algorithms
- Related to broader problems in topological data analysis
  - Coverage analysis of ad-hoc/sensor networks
  - Identifying (un)important topological features in high-dimensional data sets

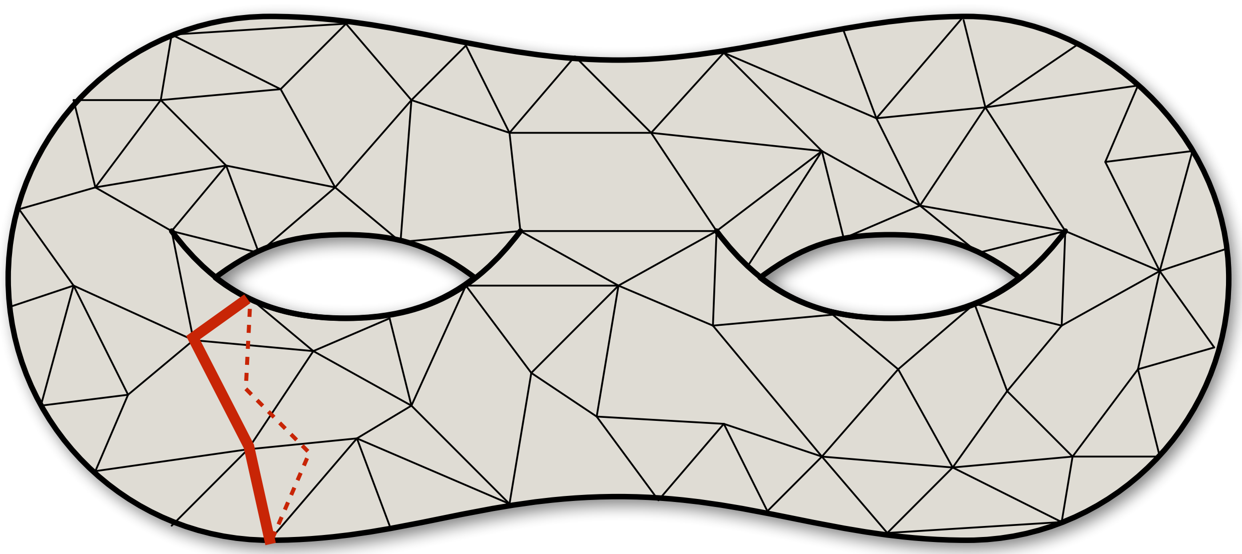
“Given”?

- Input:
  - Orientable surface map Σ with complexity n and genus g.
  - Length $\ell(e) \geq 0$ for every edge of Σ



“Given”?

- Input:
  - Orientable surface map Σ with complexity n and genus g.
  - Length $\ell(e) \geq 0$ for every edge of Σ



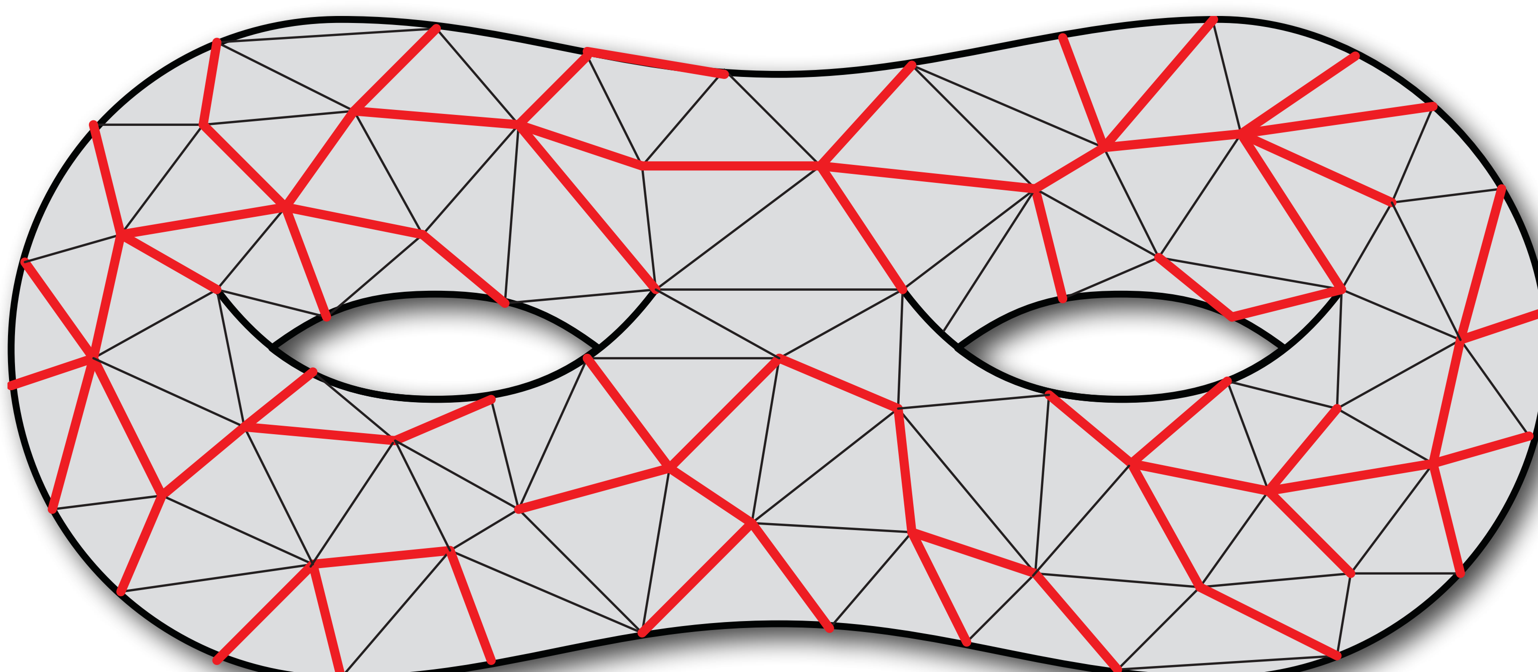
- Output:
  - Minimum-length cycle in the graph of Σ that is noncontractible or nonseparating in Σ.

Tree-cotree structures

Tree-cotree decomposition

A partition of the edges into *three* disjoint subsets:

- A spanning tree ***T***
- A spanning cotree ***C*** — C^* is a spanning tree of G^*
- Leftover edges ***L*** $:= E \setminus (C \cup T)$ — Euler's formula implies $|L| = 2g$

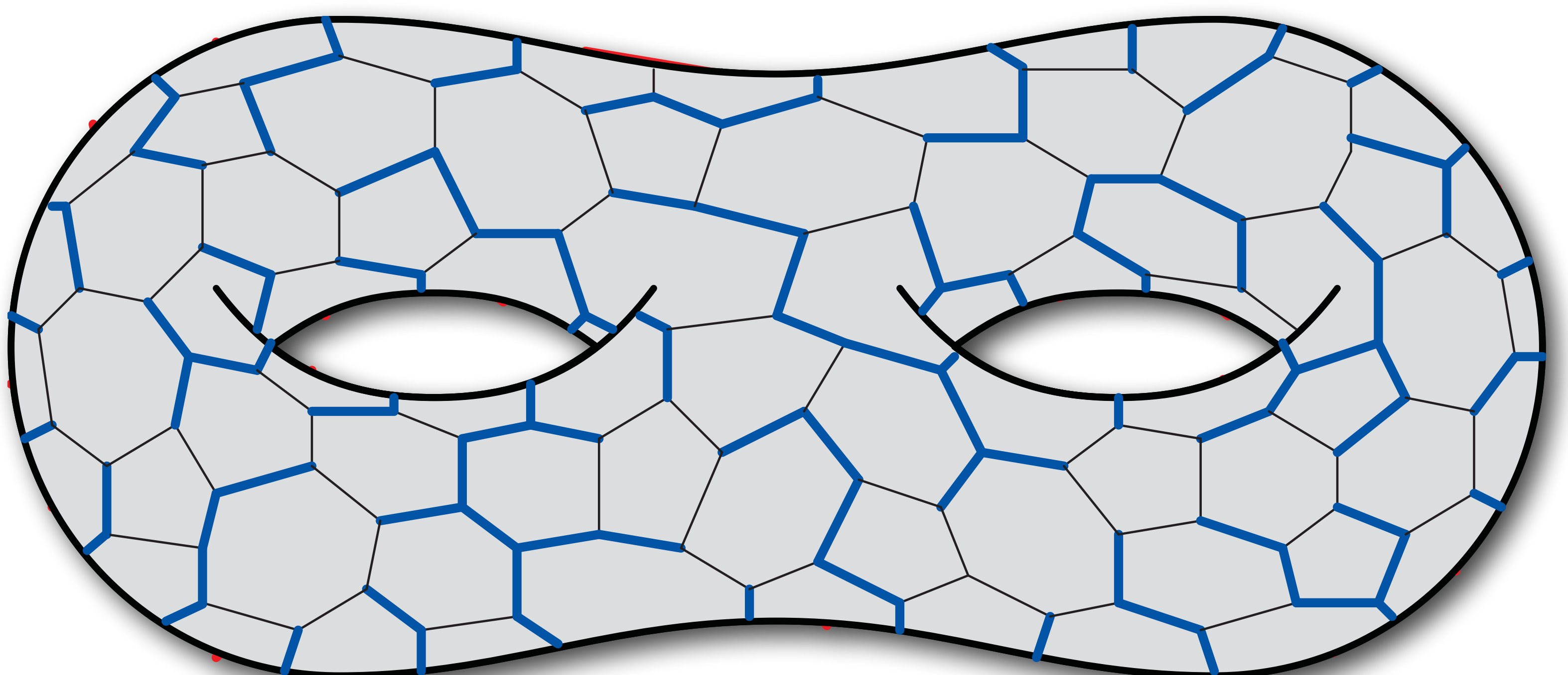


[von Staudt 1847; Dehn 1936; Biggs 1971; Eppstein 2003]

Tree-cotree decomposition

A partition of the edges into *three* disjoint subsets:

- A spanning tree ***T***
- A spanning cotree ***C*** — C^* is a spanning tree of G^*
- Leftover edges ***L*** := $E \setminus (C \cup T)$ — Euler's formula implies $|L| = 2g$

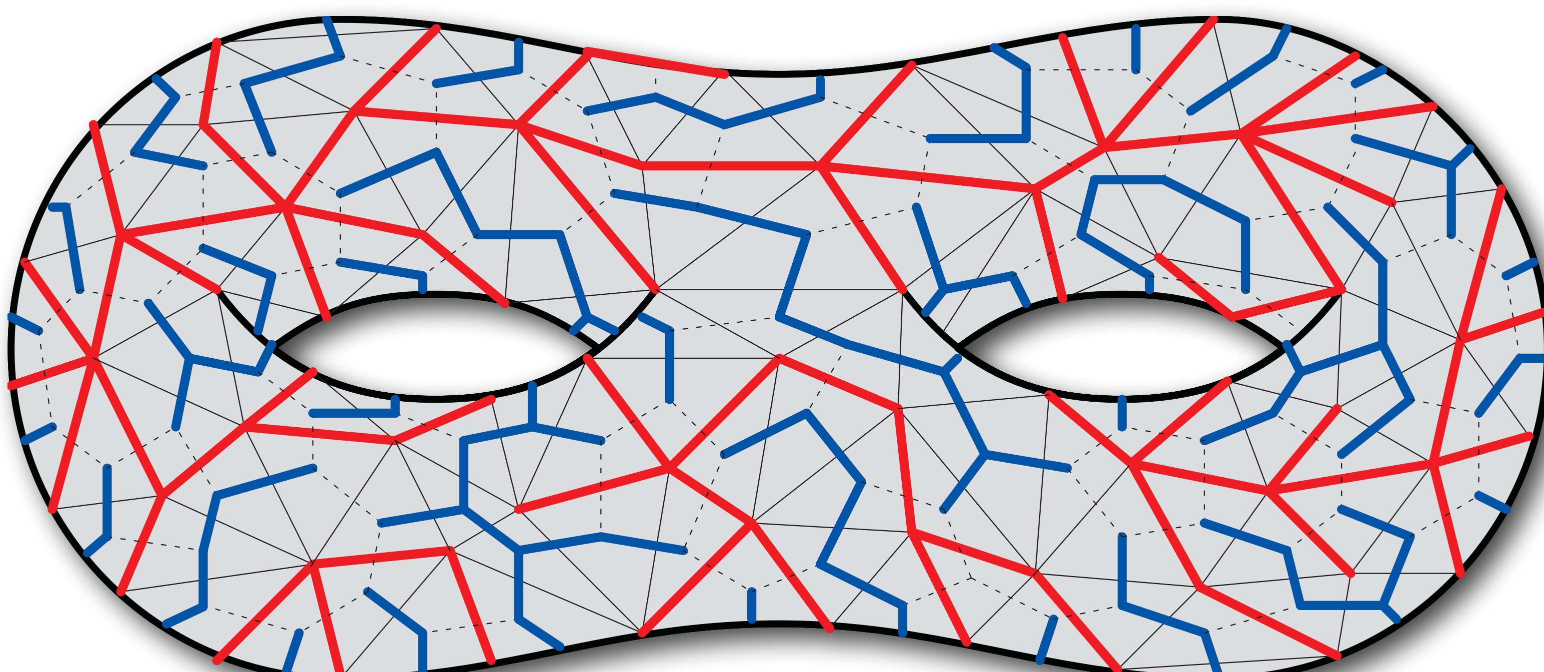


[von Staudt 1847; Dehn 1936; Biggs 1971; Eppstein 2003]

Tree-cotree decomposition

A partition of the edges into *three* disjoint subsets:

- A spanning tree ***T***
- A spanning cotree ***C*** — C^* is a spanning tree of G^*
- Leftover edges ***L*** := $E \setminus (C \cup T)$ — Euler's formula implies $|L| = 2g$

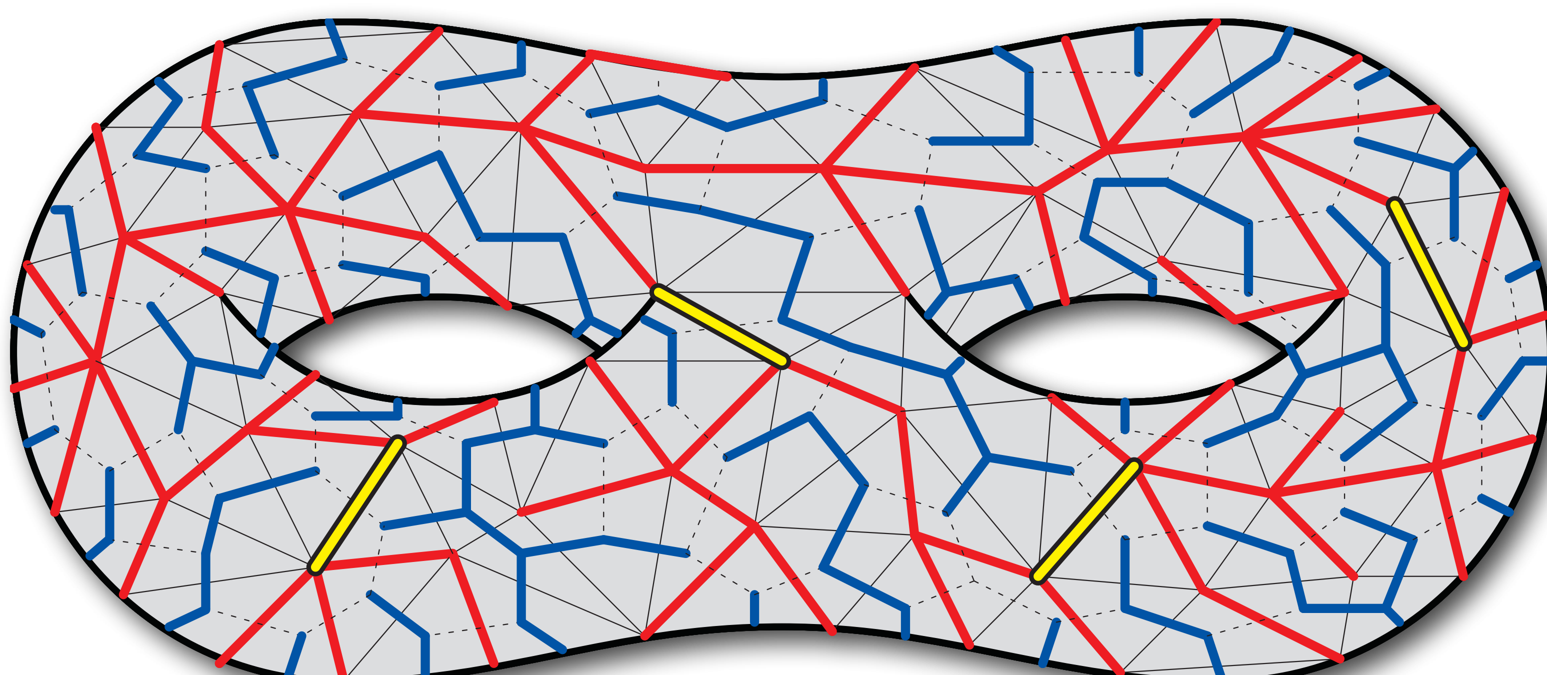


[von Staudt 1847; Dehn 1936; Biggs 1971; Eppstein 2003]

Tree-cotree decomposition

A partition of the edges into *three* disjoint subsets:

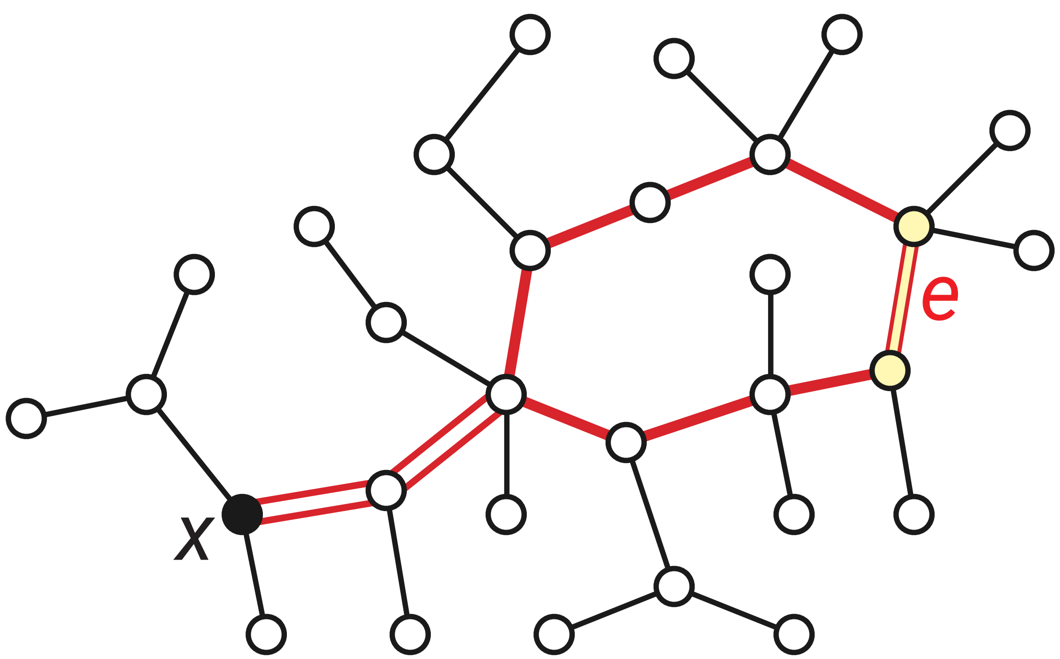
- A spanning tree ***T***
- A spanning cotree ***C*** — C^* is a spanning tree of G^*
- Leftover edges ***L*** $:= E \setminus (C \cup T)$ — Euler's formula implies $|L| = 2g$

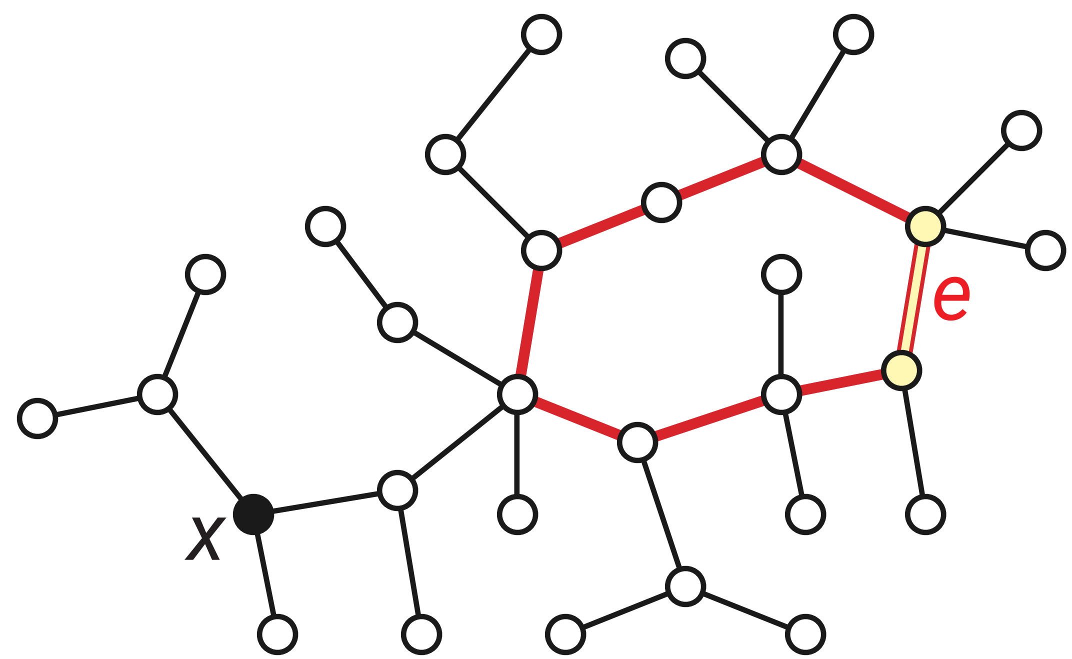


[von Staudt 1847; Dehn 1936; Biggs 1971; Eppstein 2003]

Fundamental loops and cycles

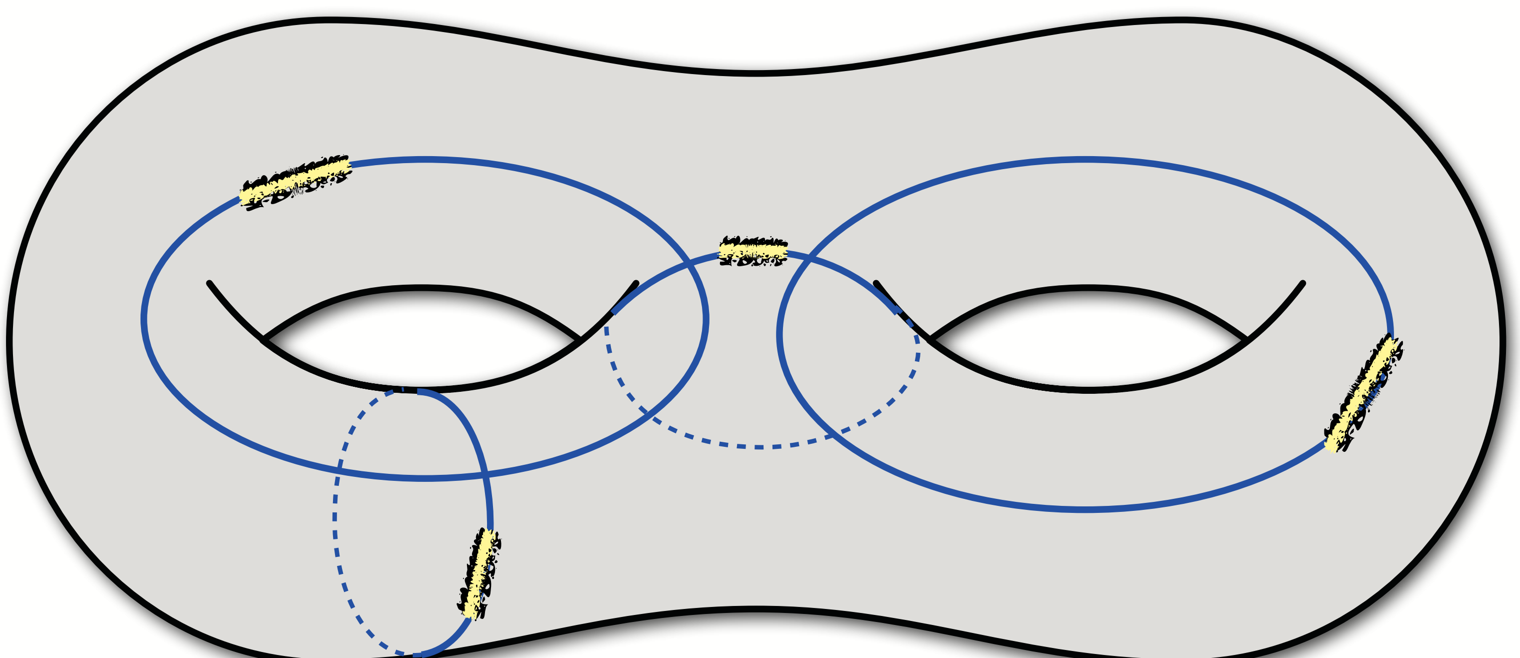
- Fix a tree-cotree decomposition (T, L, C) and a *basepoint* x.
- Each nontree edge e defines a *fundamental loop* $loop(T, e)$
- Each nontree edge e defines a *fundamental cycle* $cycle(T, e)$





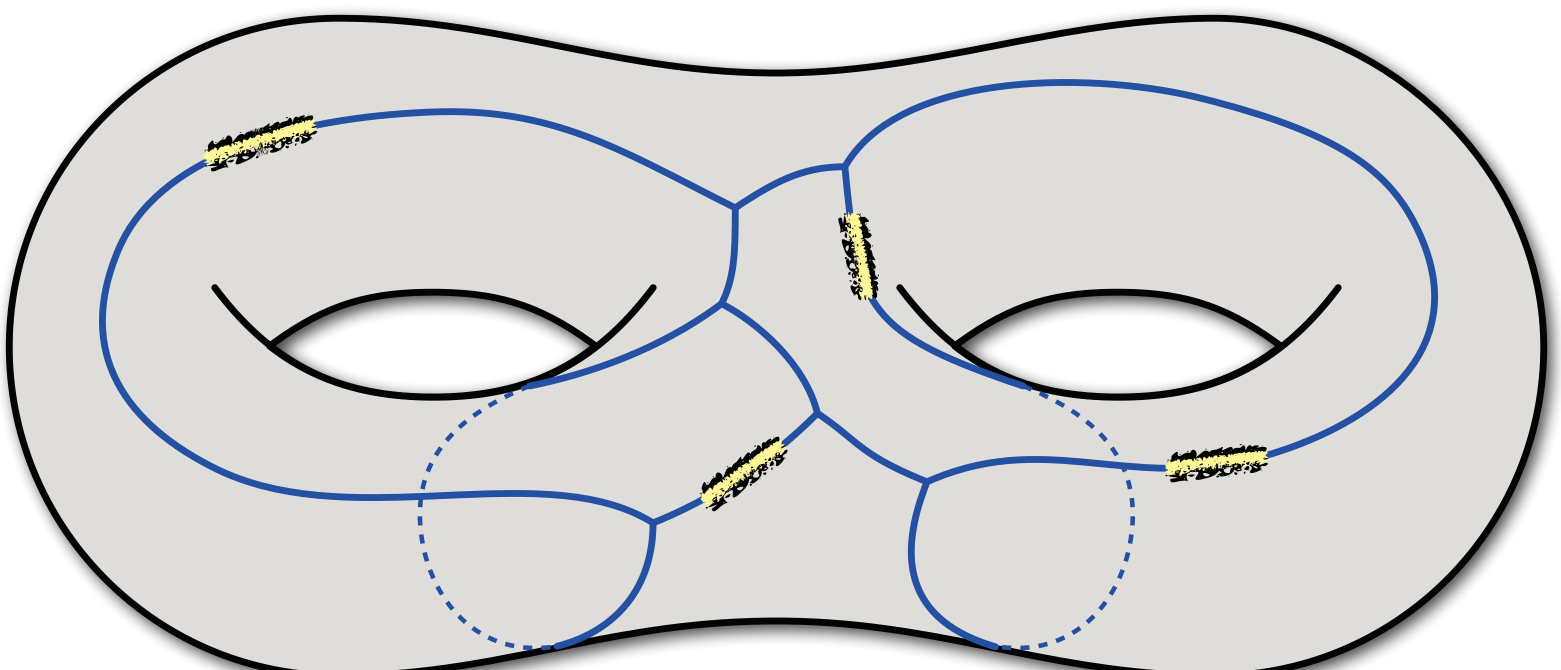
Tree-cotree structures

- *System of cycles* $\{cycle(T, e) \mid e \in L\}$
  - $2g$ simple cycles
  - Basis for the first homology group $H_1(\Sigma)$



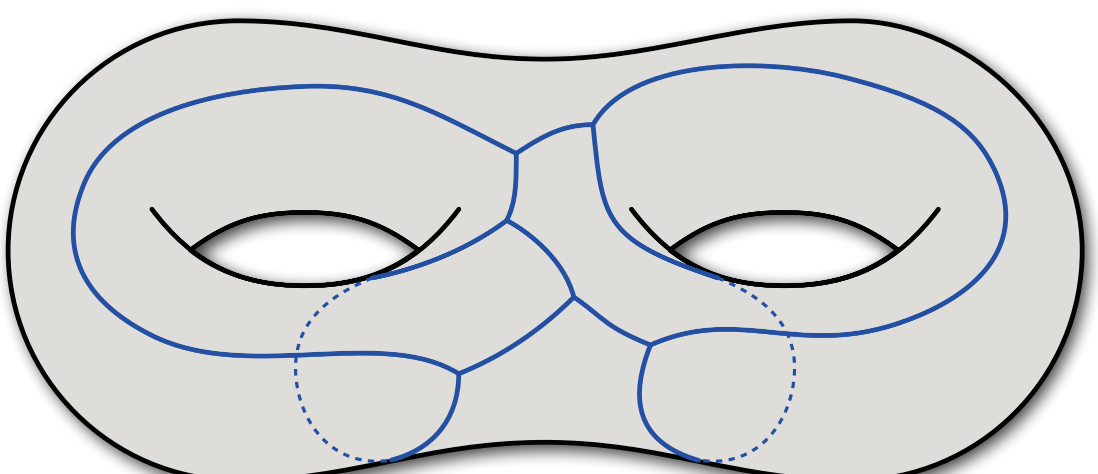
Tree-cotree structures

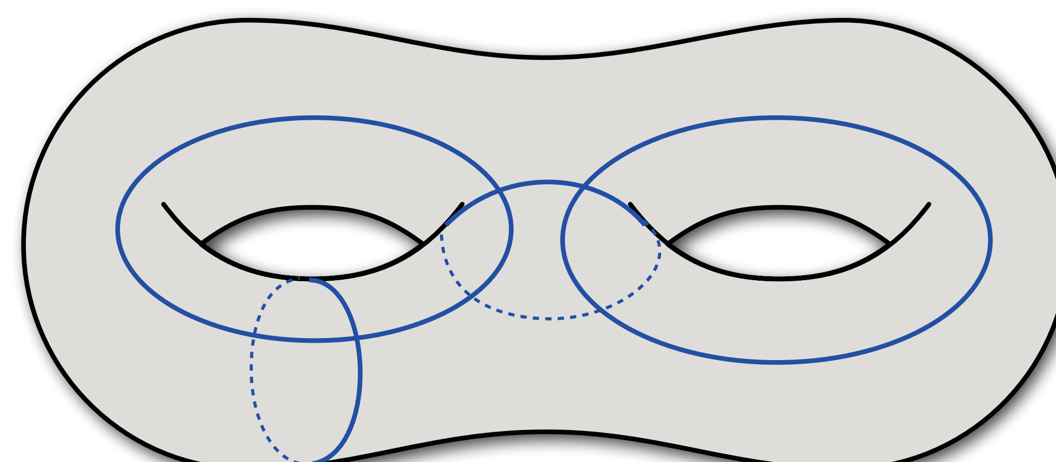
- *Cut graph* $T \cup L = \Sigma \backslash C$
- Remove degree-1 vertices ⇒ *reduced cut graph*
  - Minimal subgraph with one face
  - Composed of at most $3g$ *cut paths* meeting at most $2g$ *branch points*



Tree-cotree structures

- Often useful to build these structures *in the dual map Σ**.
- Every *noncontractible* cycle in Σ crosses every (dual) reduced cut graph at least once.
- Every *nonseparating* cycle in Σ crosses at least one cycle in every (dual) system of cycles.



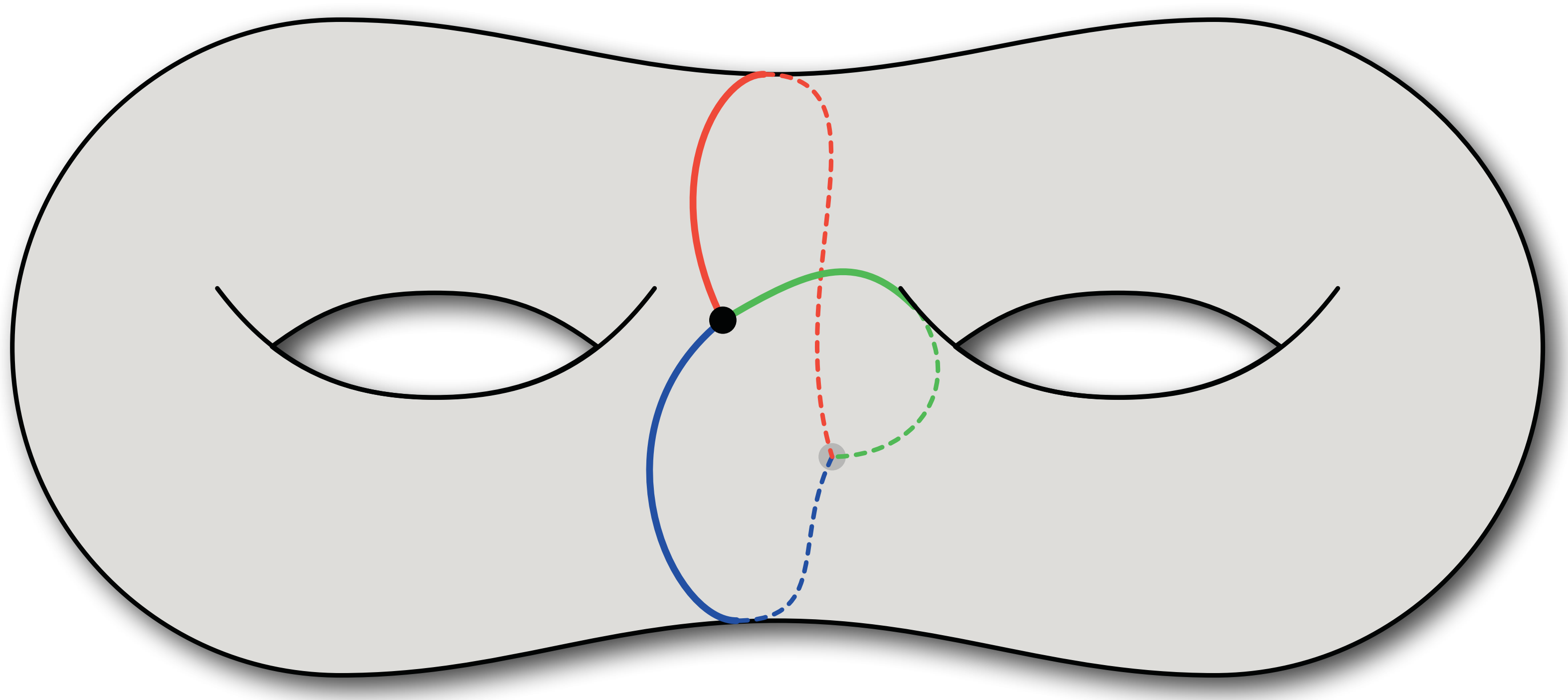


Shortest nontrivial cycles

Three-path condition

[Thomassen 1990]

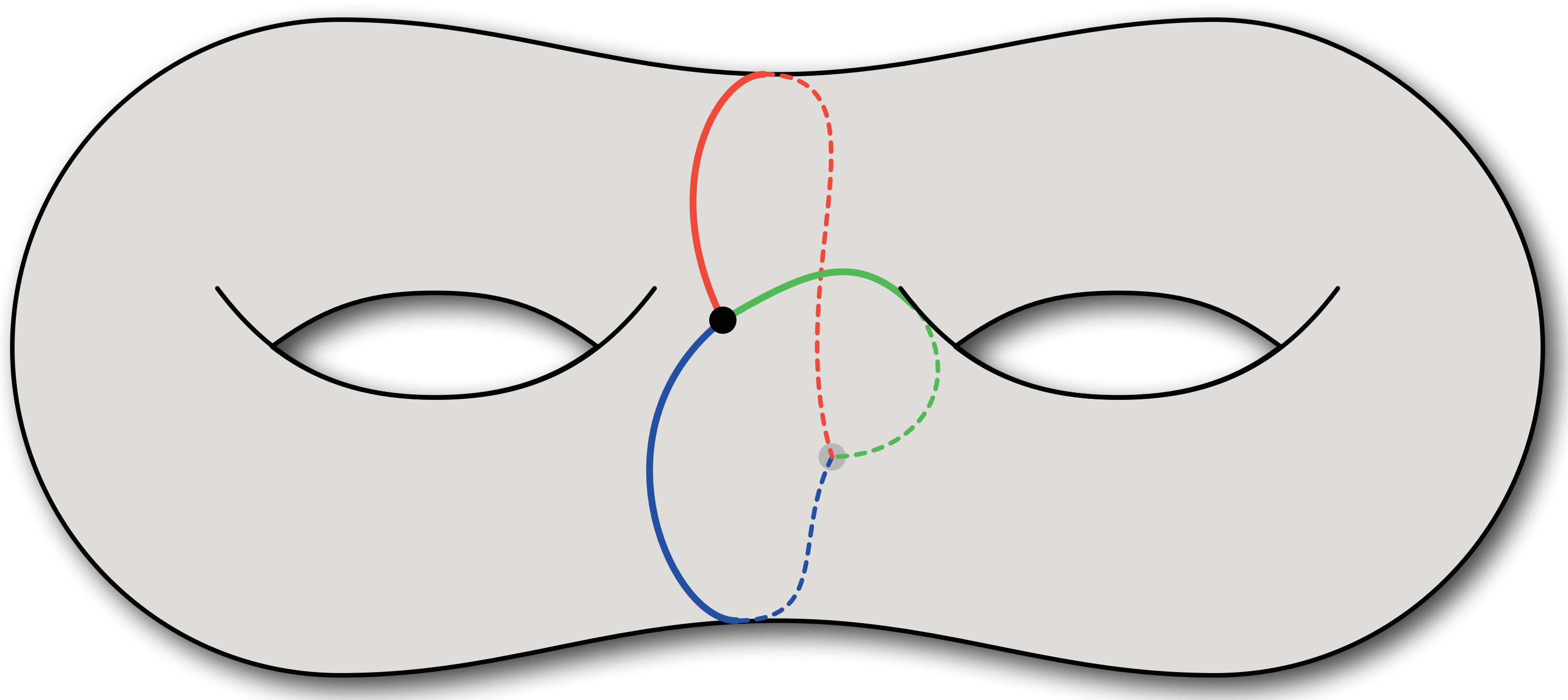
- Any three paths with same endpoints define three cycles.
- If any two of these cycles are trivial, so is the third.



Three-path condition

[Thomassen 1990]

- The shortest nontrivial cycle consists of two shortest paths between any pair of antipodal points.
- Otherwise, the actual shortest path would create a shorter nontrivial cycle.



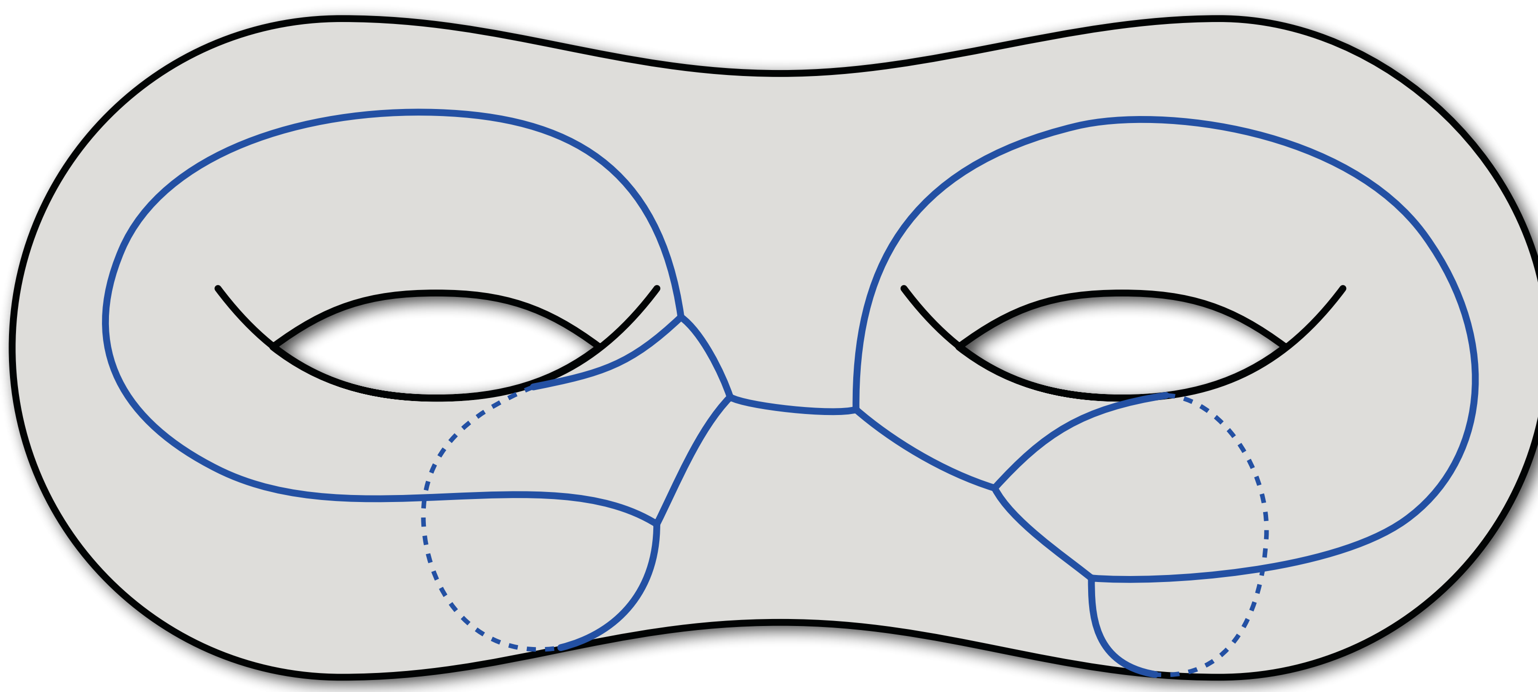
Greedy tree-cotree decomposition

- Assume edges have non-negative lengths $\ell(e) \geq 0$
- T = *shortest-path* tree in Σ with arbitrary source vertex x
- C^* = *maximum* spanning tree of Σ^* where $w(e^*) = \ell(loop(T,e))$
- Computable in $O(n \log n)$ time using textbook algorithms.
  - $O(n)$ time if all lengths = 1
  - $O(n)$ time if $g = O(n^{1-\varepsilon})$ *[Henzinger et al. '97]*

[Eppstein 2003, Erickson Whittlesey 2005]

Shortest nontrivial loops

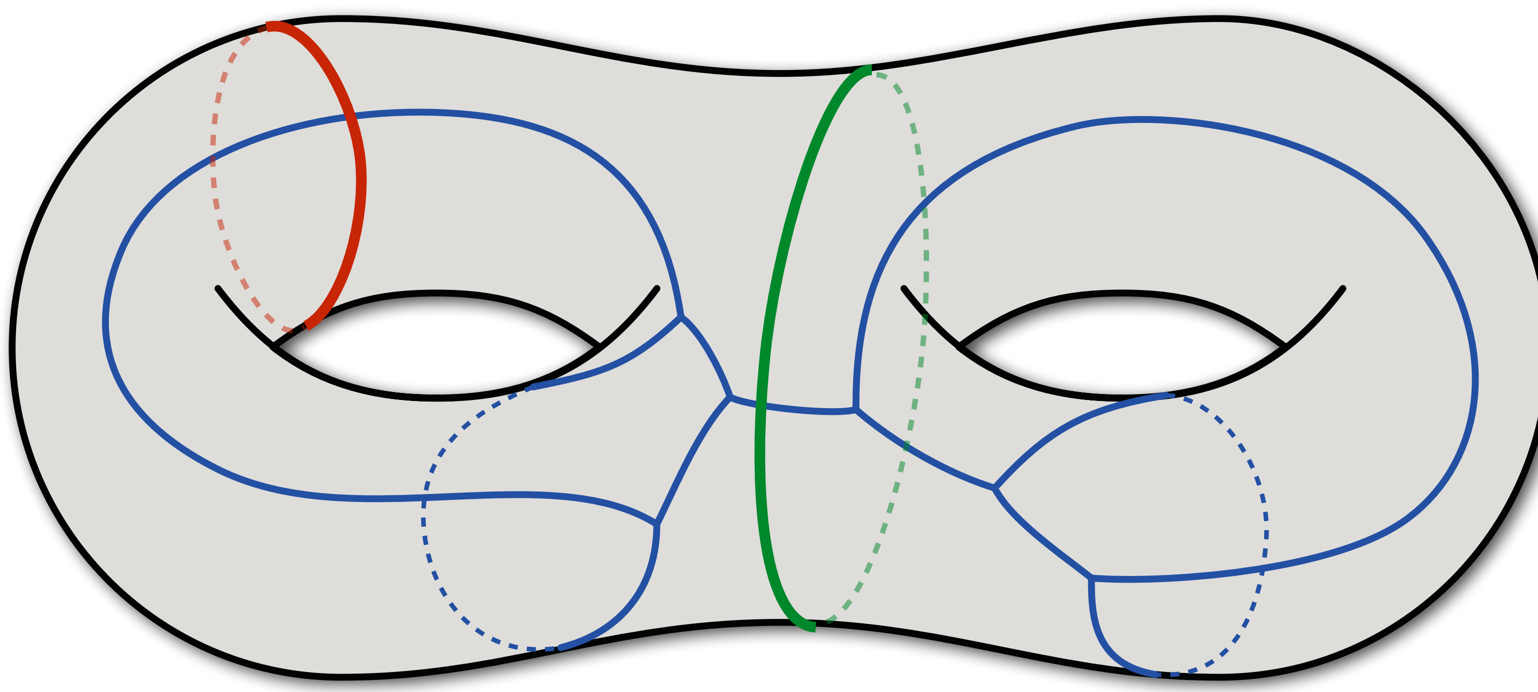
- Build greedy tree-cotree decomposition (T, L, C) based at x.
- Build *dual* cut graph $X^* = L^* \cup C^*$
- Reduce X^* to get R^*



[Erickson Har-Peled 2005]

Shortest nontrivial loops

- 3-path condition $\Rightarrow$ We want *loop*(T, e) for some $e \notin T$
- *loop*(T, e) is *noncontractible* iff $e^* \in R^*$
- *loop*(T, e) is *nonseparating* iff $R^* \backslash e^*$ is connected



[Erickson Har-Peled 2005]

[Cabello, Colin de Verdière, Lazarus 2010]

Shortest non-trivial cycle

[Erickson Har-Peled 2005]

- For each basepoint: $O(n \log n)$ time.
- Try all possible basepoints: $O(n^2 \log n)$ time.

Shortest non-trivial cycle

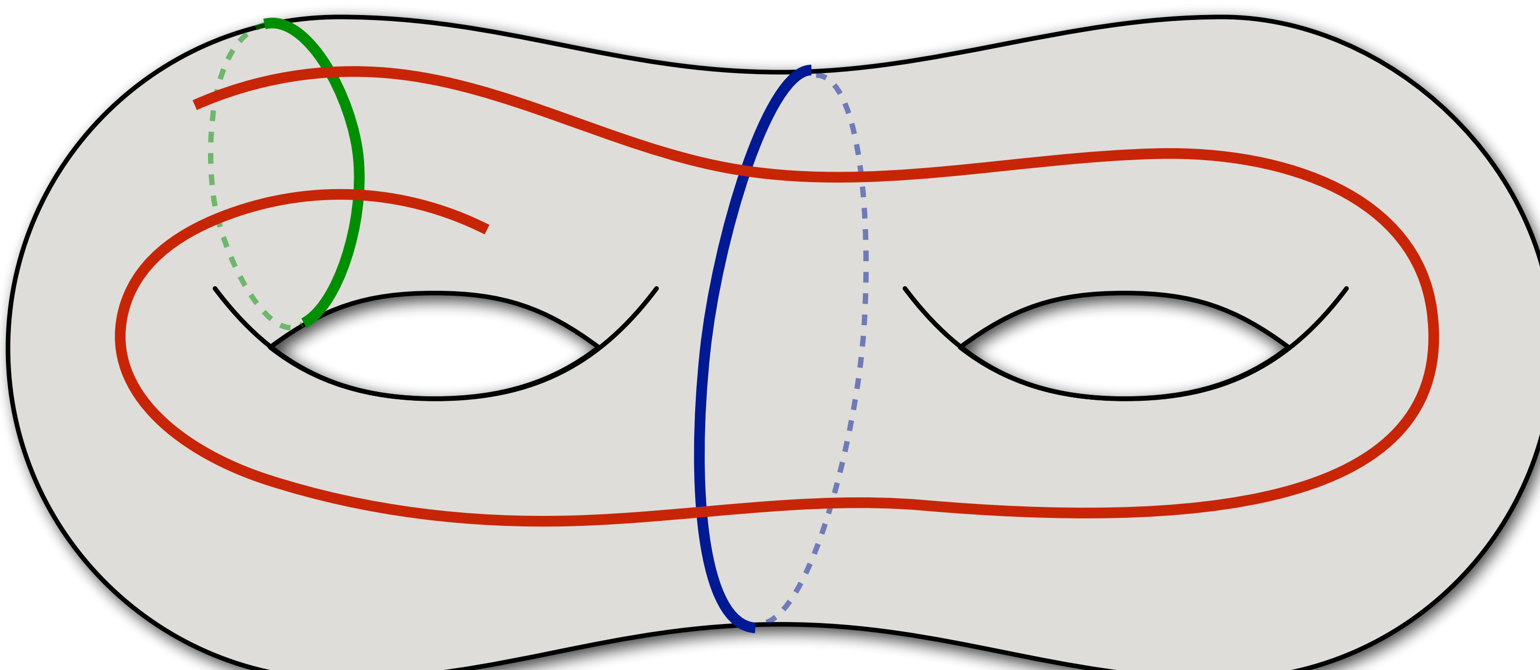
[Erickson Har-Peled 2005]

- For each basepoint: $O(n \log n)$ time.
- Try all possible basepoints: $O(n^2 \log n)$ time.

- *This is the fastest algorithm known.*
  - Significant improvement would also improve the best time to compute the *girth* of a sparse graph: $O(n^2)$ = BFS at each vertex *[Itai Rodeh 1978]*
  - Computing the girth of a *dense* graph is at least as hard as all-pairs shortest paths and boolean matrix multiplication. *[Vassilevska Williams, Williams 2010]*

One-cross lemmas

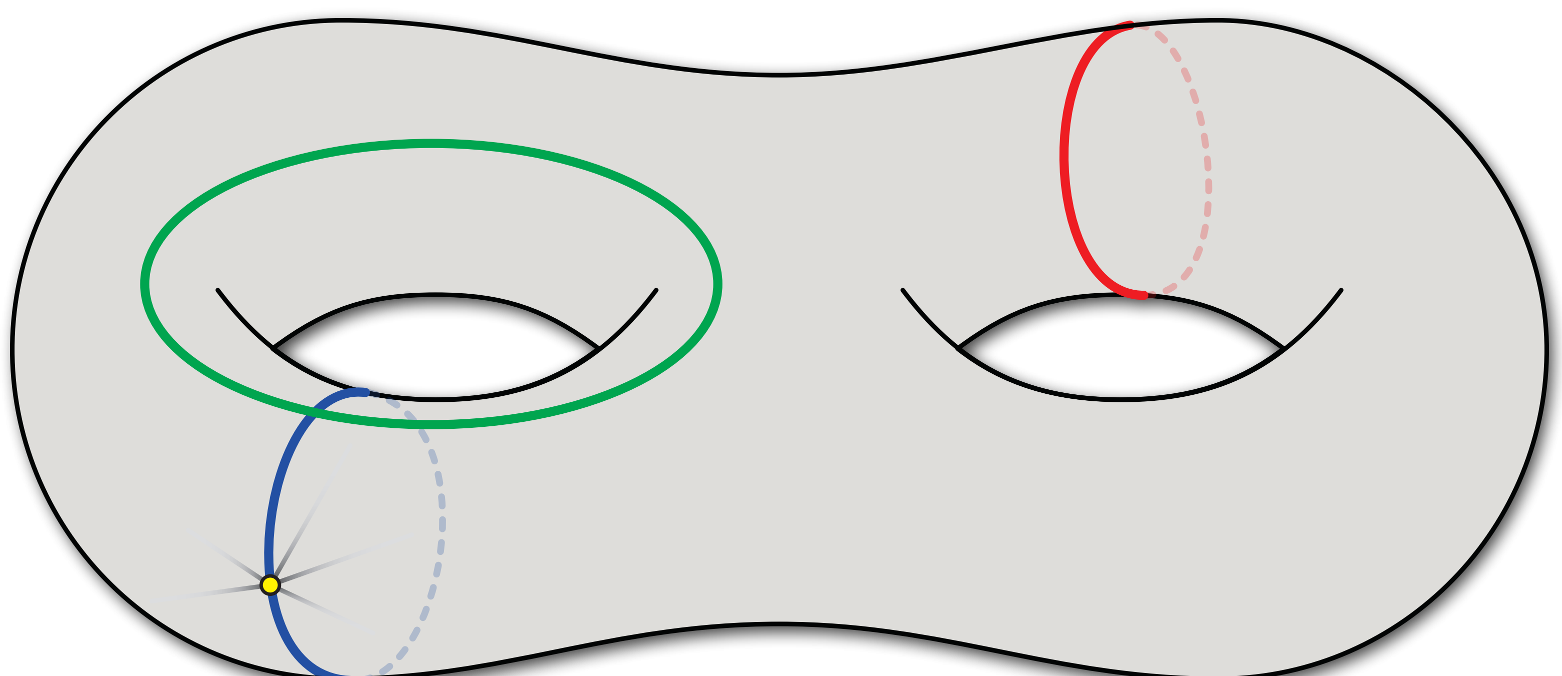
- The shortest nontrivial cycle crosses any shortest path at most once
- Otherwise, we could find a shorter nontrivial cycle!



One-cross lemmas

[Cabello Mojar 2005]

- Let γ^* be the shortest *nonseparating* cycle, and let γ be any cycle in a greedy system of cycles.
- Then γ^* and γ cross *at most* once.

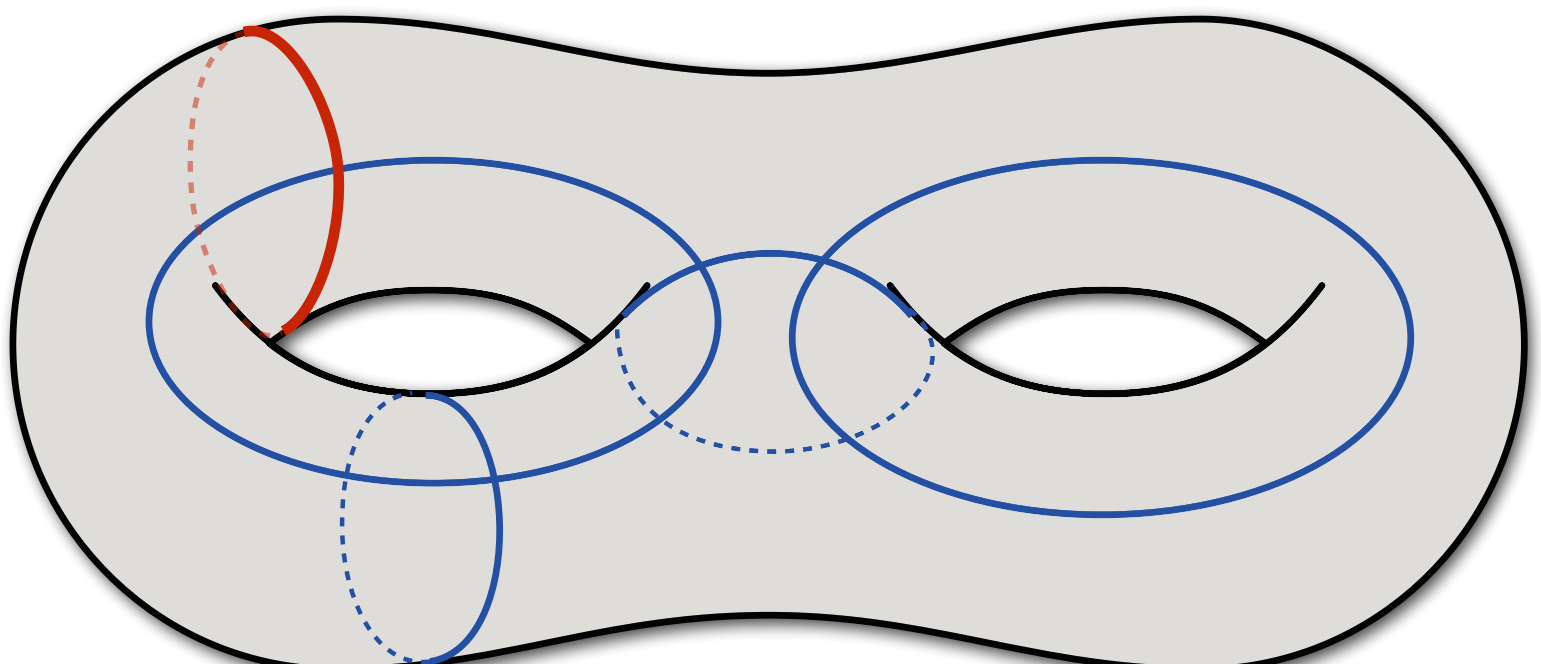


Faster algorithm

[Cabello Chambers 2007]

To compute the shortest *nonseparating* cycle:

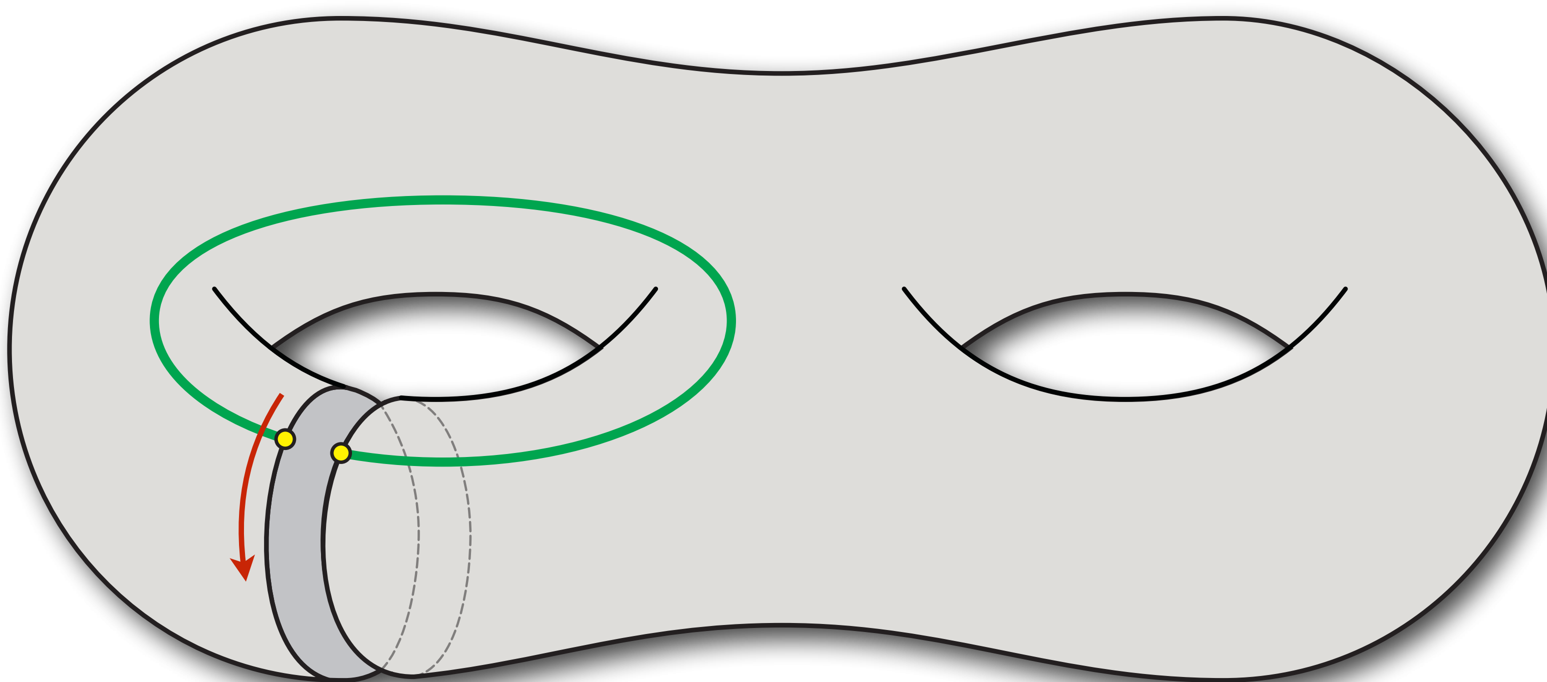
- Compute a greedy system of cycles $\gamma_1, \gamma_2, \ldots, \gamma_{2g}$
- For each i, find the shortest cycle that crosses γ_i *exactly* once



Faster algorithm

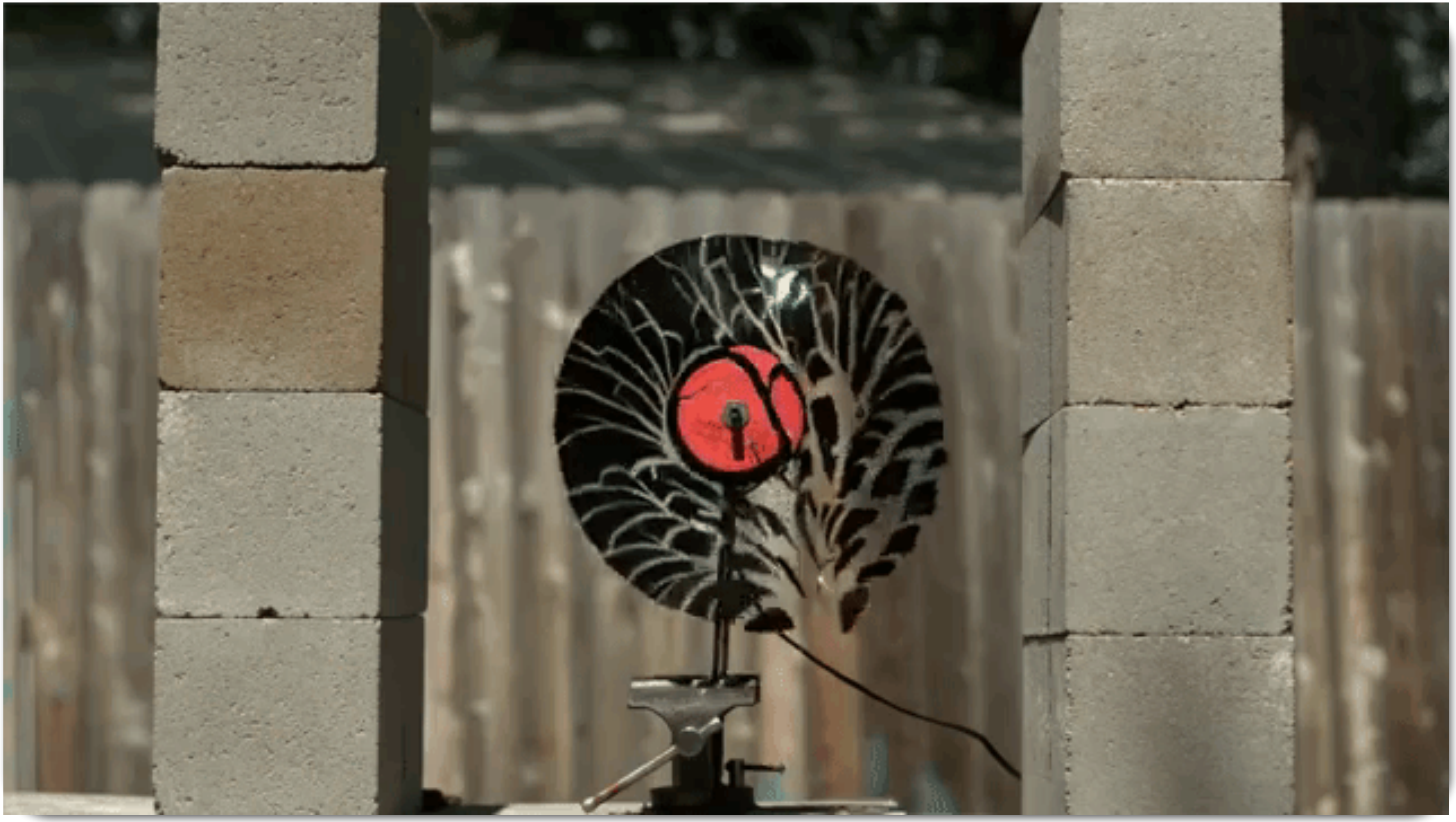
[Cabello Chambers 2007]

- To find the shortest cycle that crosses γ_i exactly once:
  - Cut the surface open along γ_i. Resulting surface Σ✂γ_i has two copies of γ on its boundary.
  - Find the *shortest path* in Σ✂γ_i between the clones of each vertex of γ_i



Please ask questions!

Multiple-Source Shortest Paths



[Free Gruchy (“Slow-Mo Guys”) 2018]

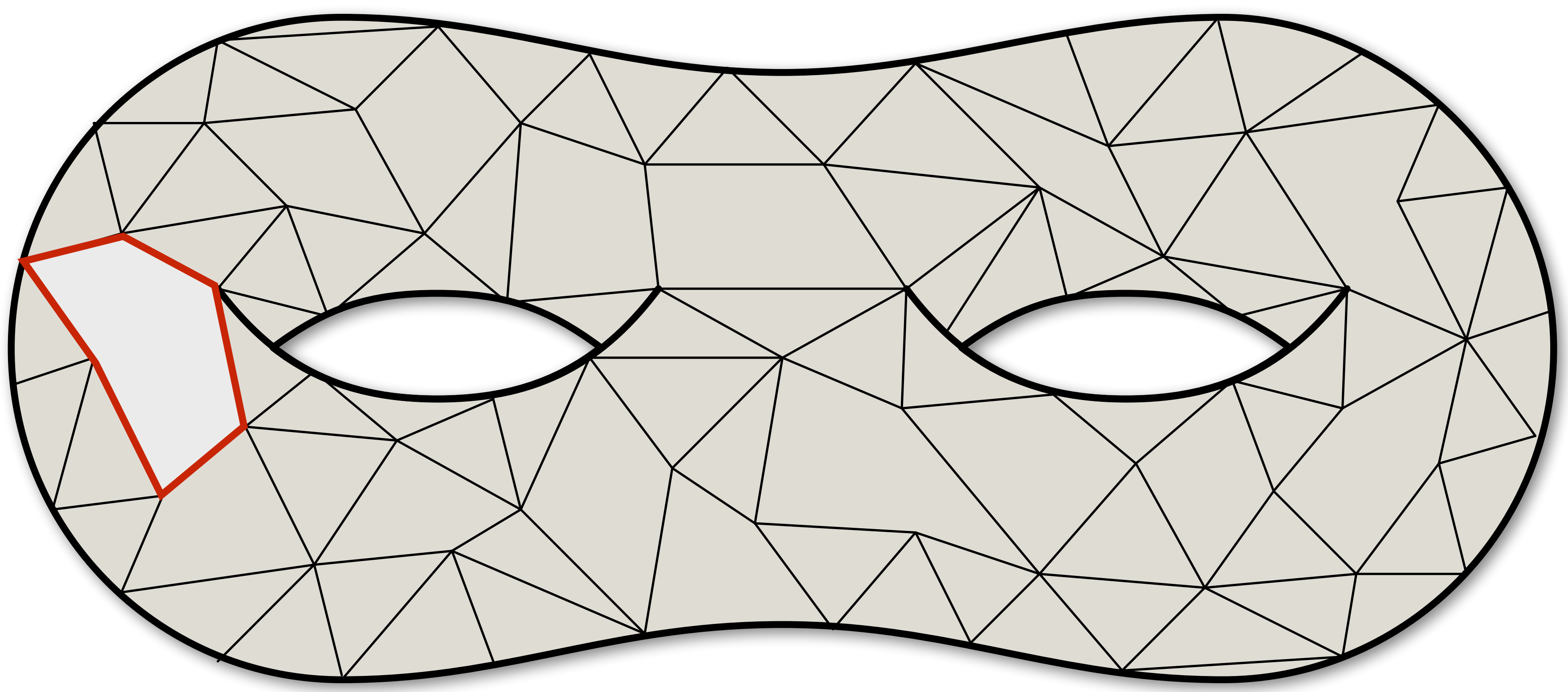
Multiple-Source Shortest Paths

[Free Gruchy (“Slow-Mo Guys”) 2018]

Multiple-Source Shortest Paths

[Klein 2005]

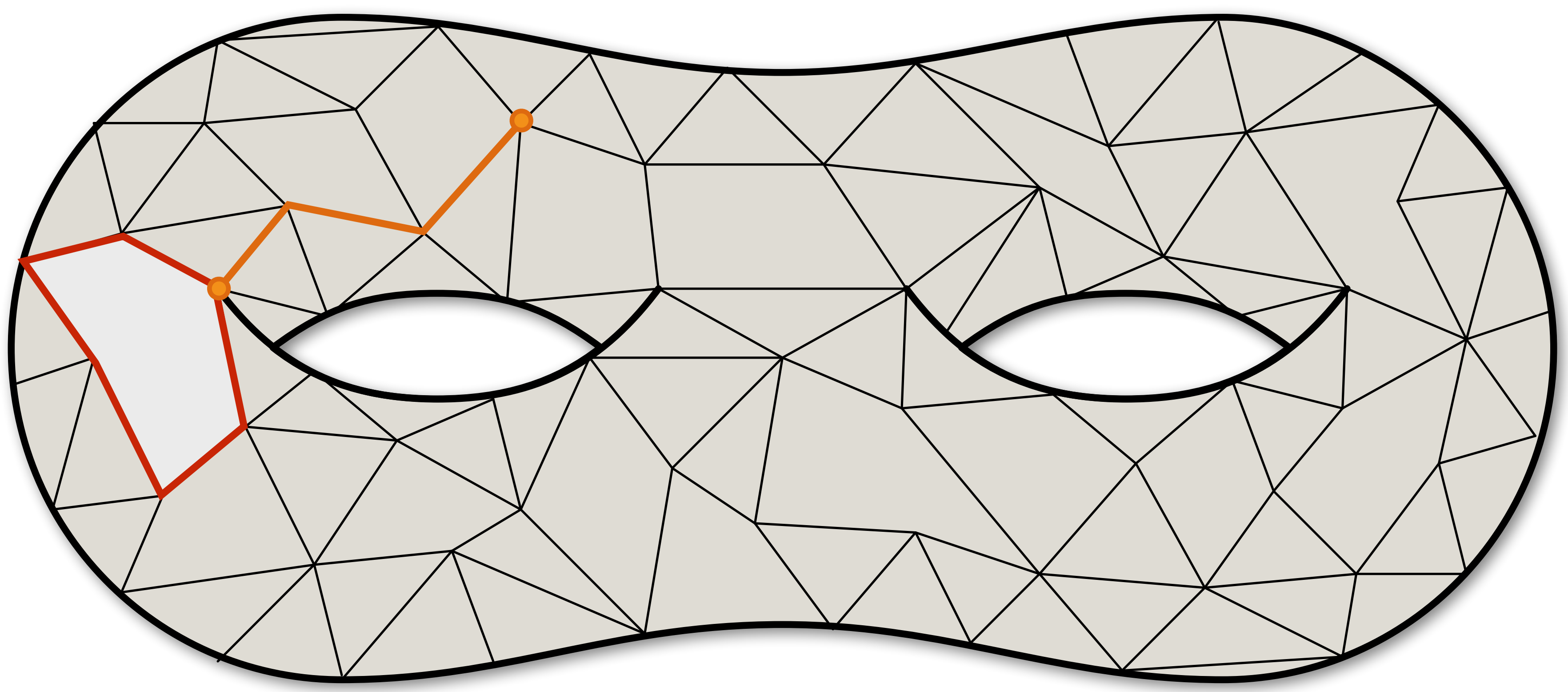
- Compute shortest paths between many pairs of vertices, with one vertex of each pair on a fixed “outer” face.



Multiple-Source Shortest Paths

[Klein 2005]

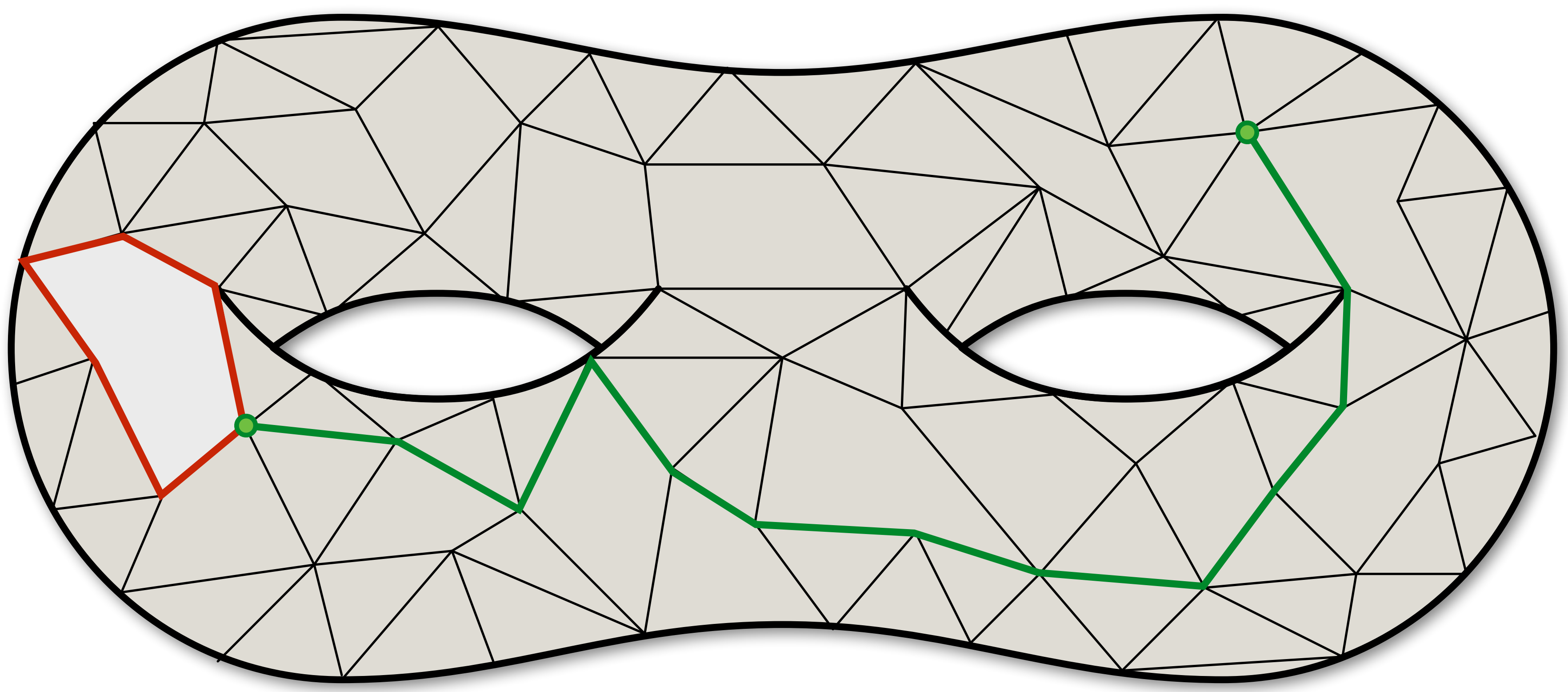
- Compute shortest paths between many pairs of vertices, with one vertex of each pair on a fixed “outer” face.



Multiple-Source Shortest Paths

[Klein 2005]

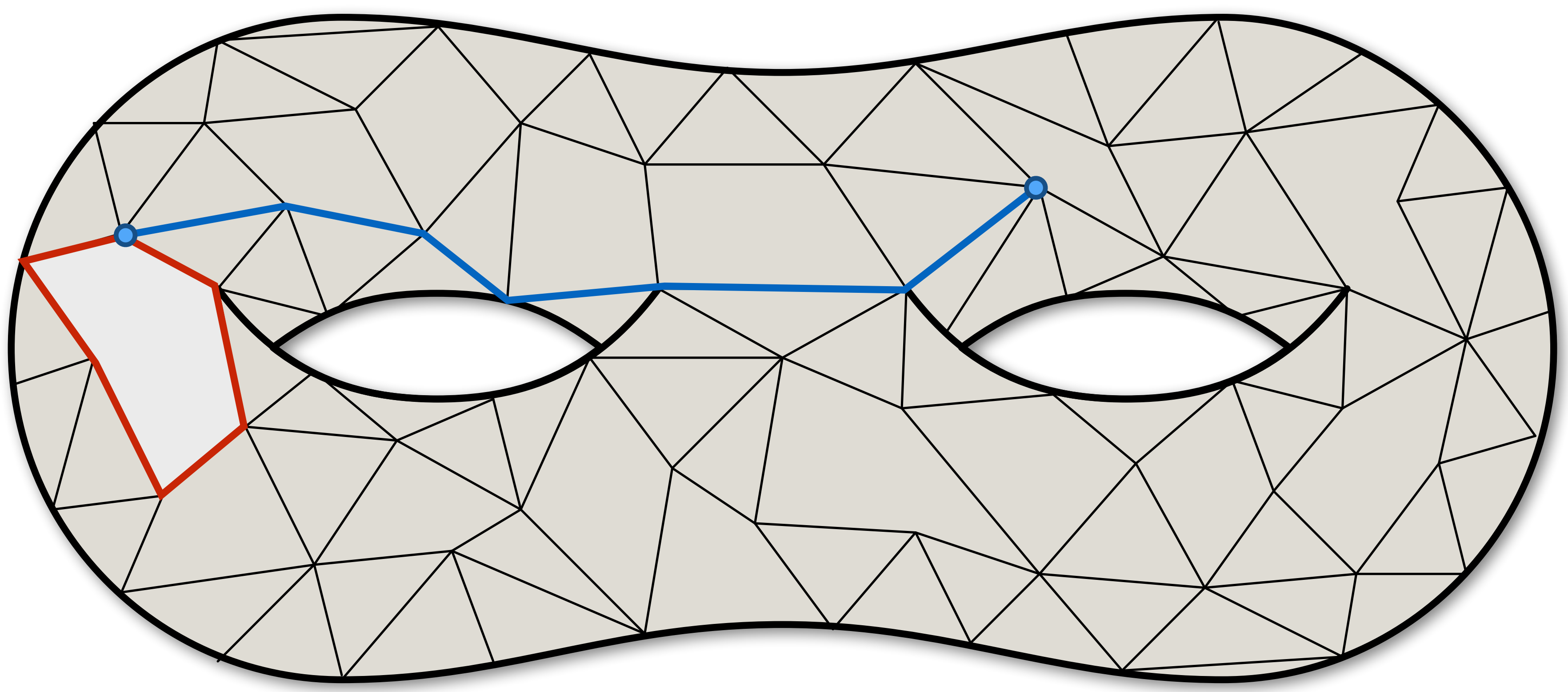
- Compute shortest paths between many pairs of vertices, with one vertex of each pair on a fixed “outer” face.



Multiple-Source Shortest Paths

[Klein 2005]

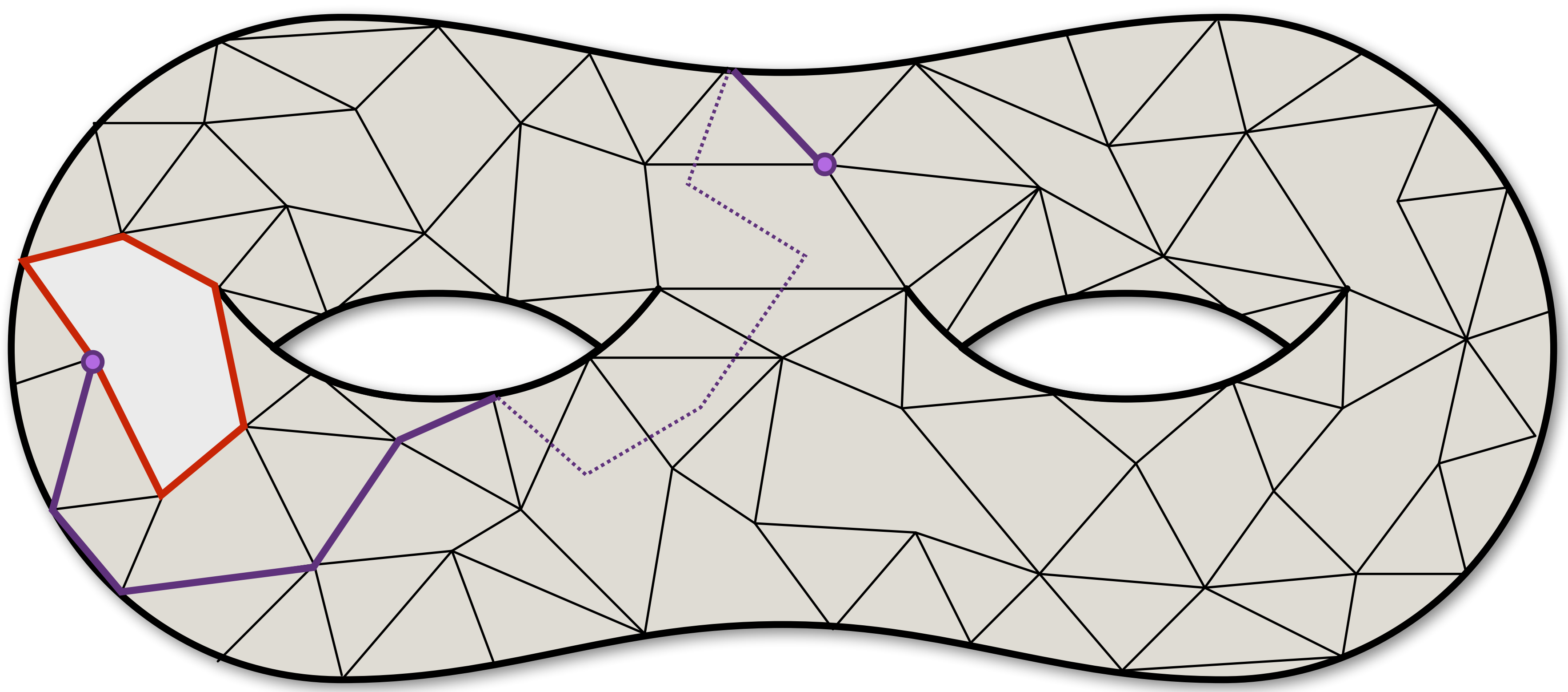
- Compute shortest paths between many pairs of vertices, with one vertex of each pair on a fixed “outer” face.



Multiple-Source Shortest Paths

[Klein 2005]

- Compute shortest paths between many pairs of vertices, with one vertex of each pair on a fixed “outer” face.



Naïve algorithm

- For each boundary vertex s, compute the shortest-path tree rooted at s in $O(n \log n)$ time. *[Dijkstra 1956]*
- The overall algorithm runs in $O(n^2 \log n)$ time.
- But in fact, we can (implicitly) compute ***all*** such distances in just $O(gn \log n)$ time.

Faster algorithm

[Cabello Chambers Erickson 2013]

To compute the shortest nonseparating cycle:

- Compute a greedy tree-cotree decomposition
- Compute a greedy system of cycles $\gamma_1, \gamma_2, \ldots, \gamma_{2g}$
- For each i, find the shortest cycle that crosses γ_i *exactly* once, in $O(gn \log n)$ time via MSSP

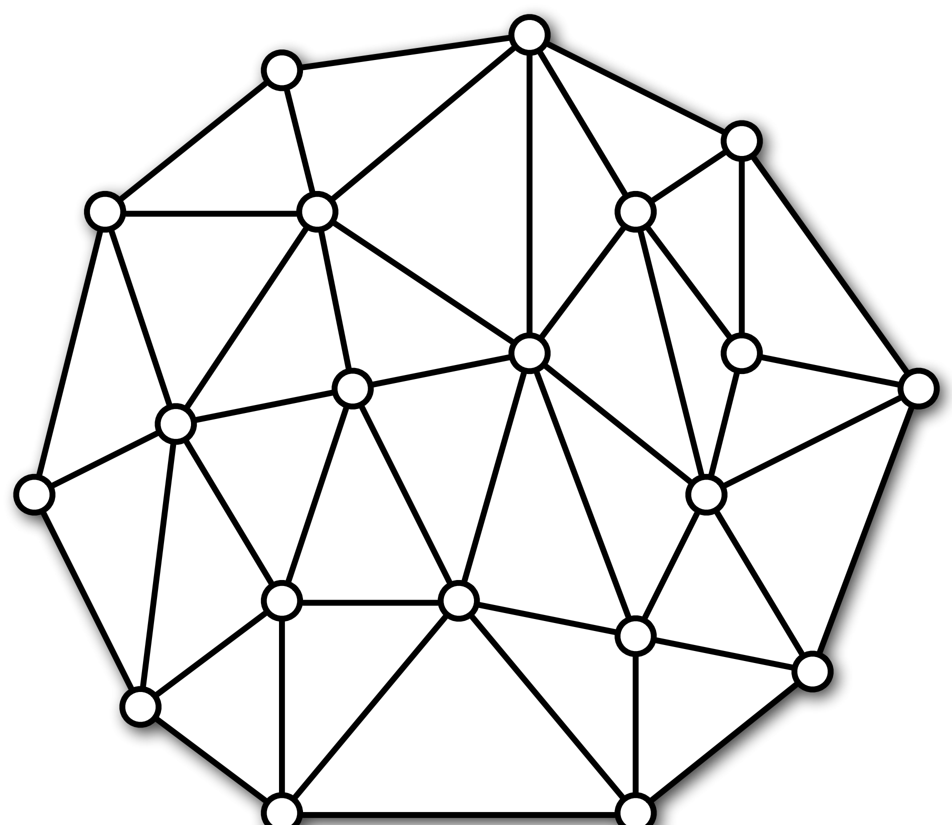
- Overall algorithm runs in $O(g^2 n \log n)$ time

- This is the *fastest algorithm known* in terms of both n and g.

Planar MSSP

[Klein 2005]

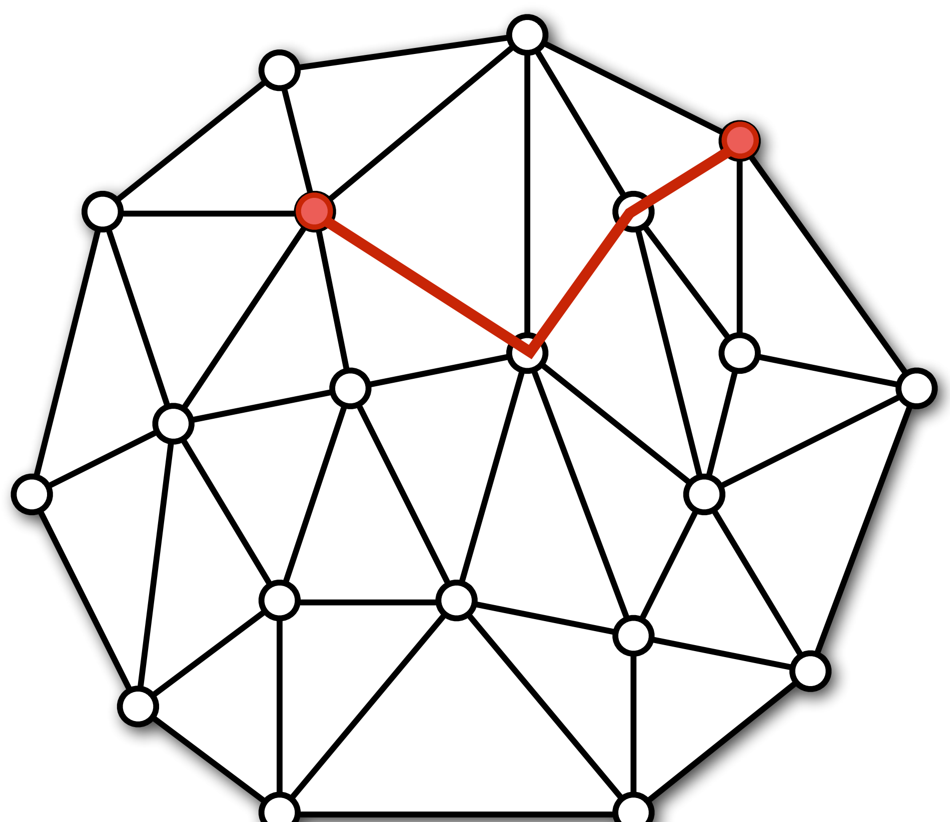
- Let's start with the simplest possible setting.
- Implicitly compute shortest paths in a *plane* graph G from every boundary vertex to every other vertex.



Planar MSSP

[Klein 2005]

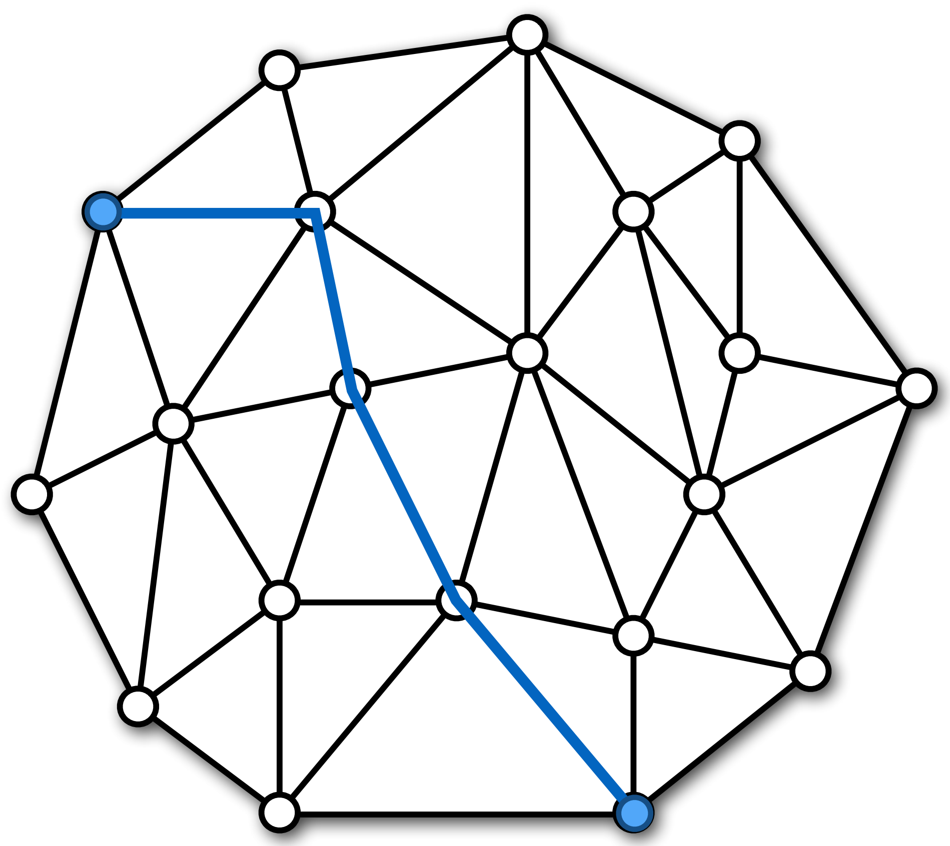
- Let's start with the simplest possible setting.
- Implicitly compute shortest paths in a *plane* graph G from every boundary vertex to every other vertex.



Planar MSSP

[Klein 2005]

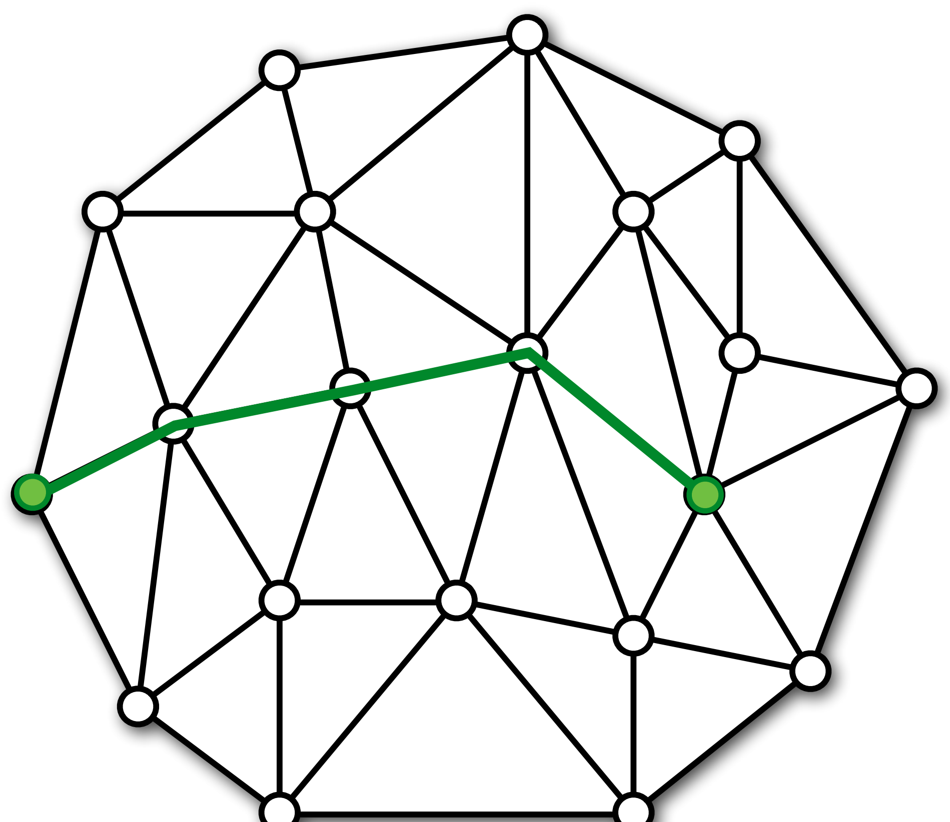
- Let's start with the simplest possible setting.
- Implicitly compute shortest paths in a *plane* graph G from every boundary vertex to every other vertex.



Planar MSSP

[Klein 2005]

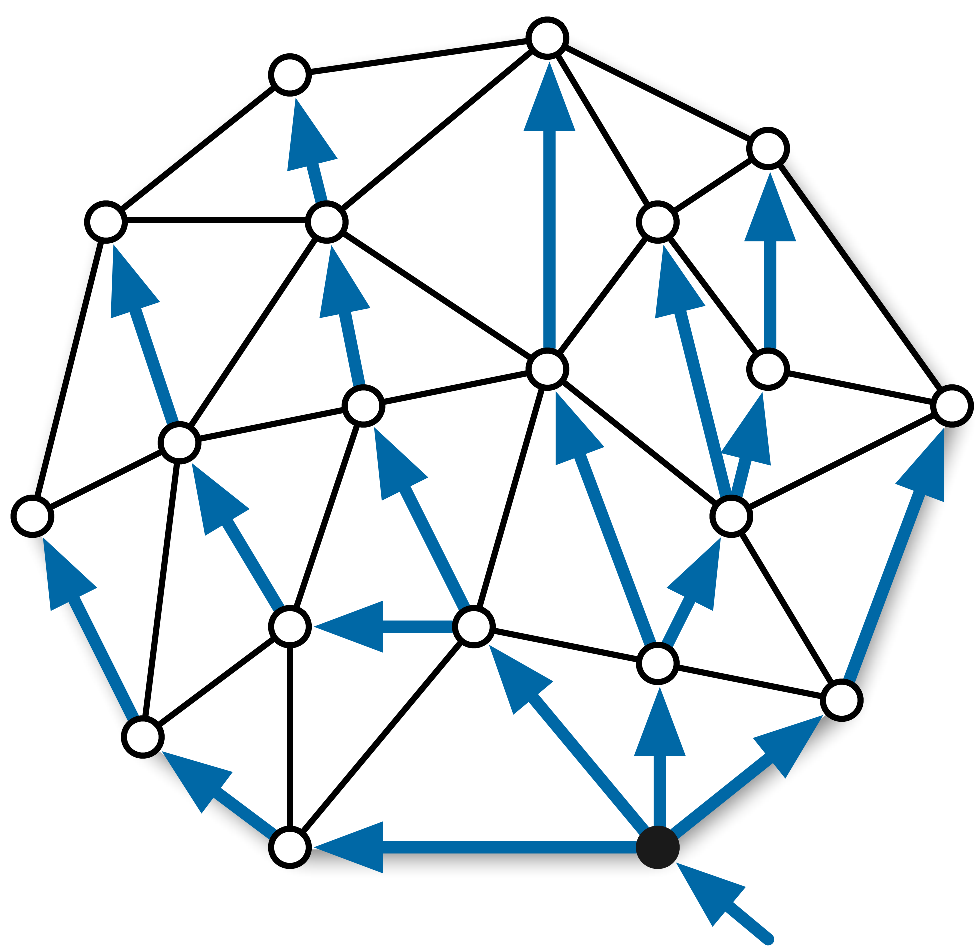
- Let's start with the simplest possible setting.
- Implicitly compute shortest paths in a *plane* graph G from every boundary vertex to every other vertex.



Planar MSSP

[Klein 2005]

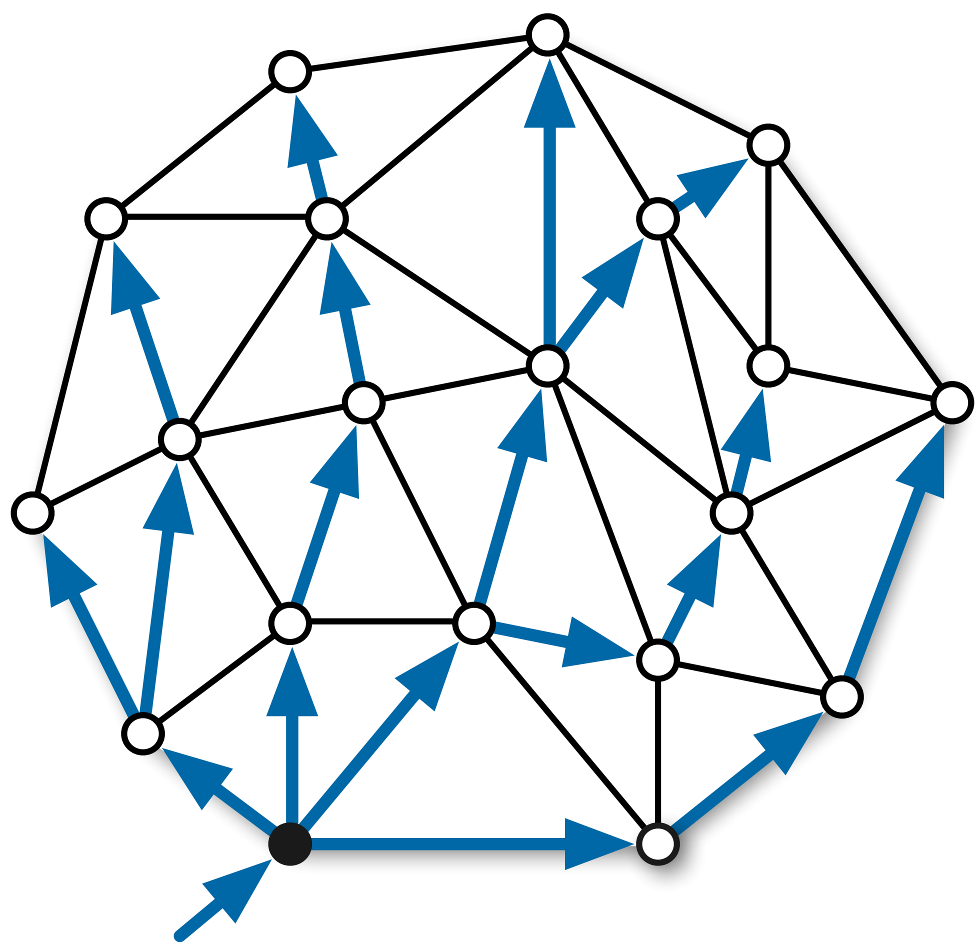
- Intuitively, we want the shortest-path tree rooted at every boundary vertex.



Planar MSSP

[Klein 2005]

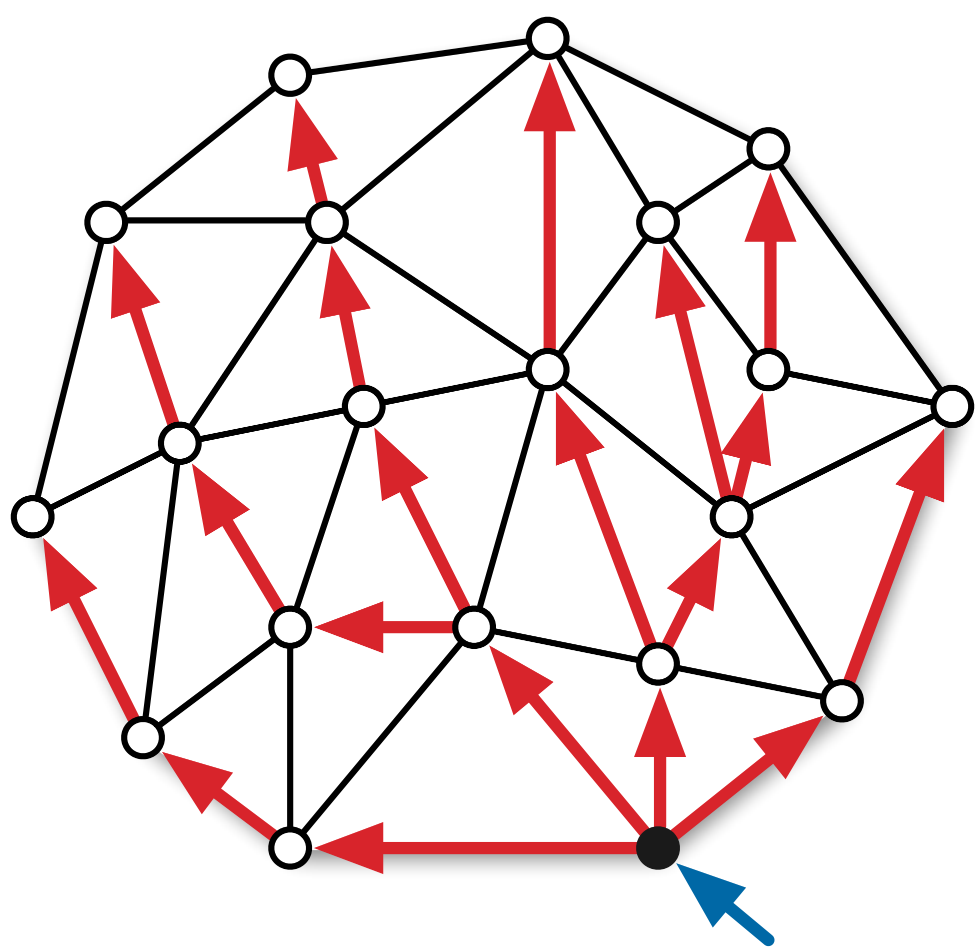
- Intuitively, we want the shortest-path tree rooted at every boundary vertex.



Planar MSSP

[Klein 2005]

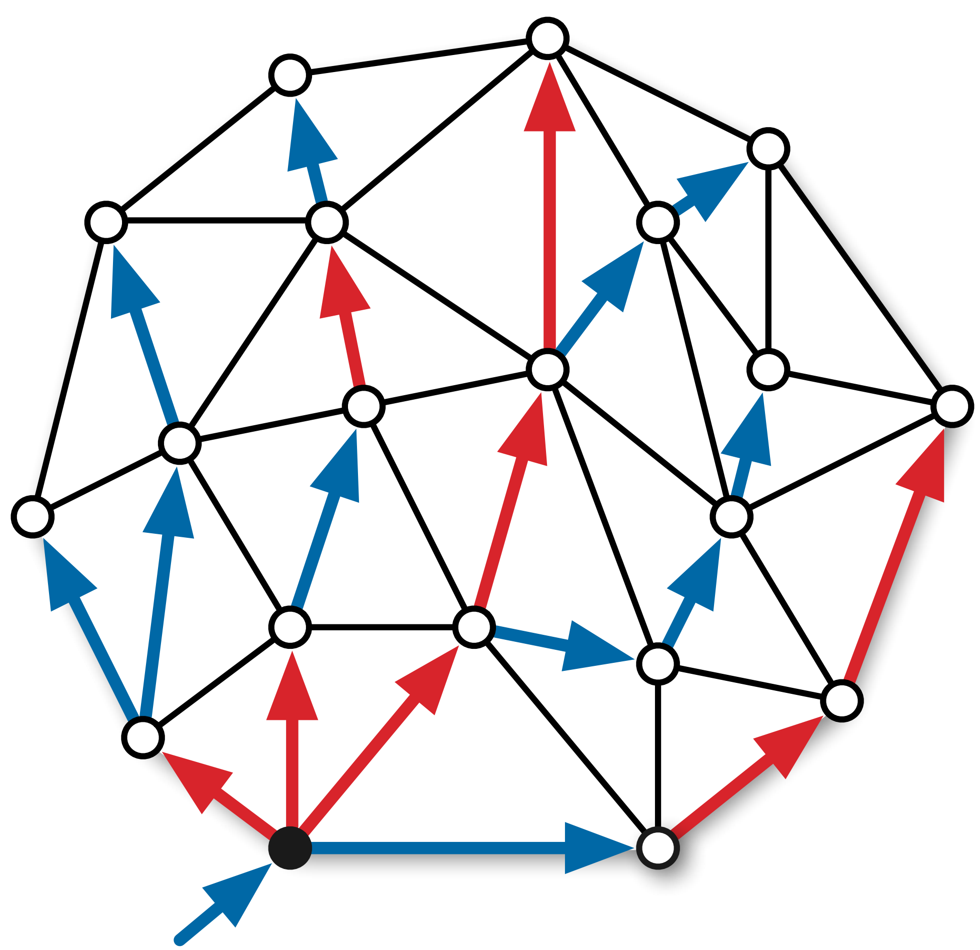
- In fact, we only need to compute the *first* shortest-path tree, followed by *changes* from each tree to the next.



Planar MSSP

[Klein 2005]

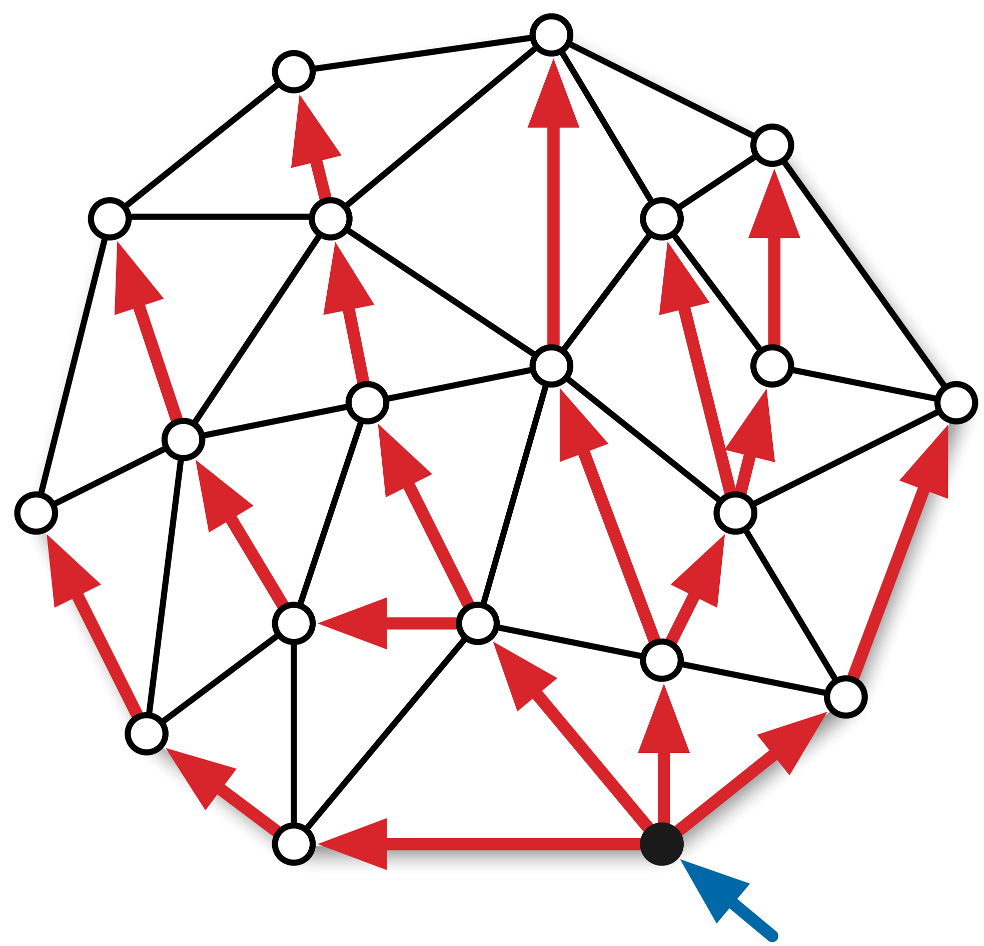
- In fact, we only need to compute the *first* shortest-path tree, followed by *changes* from each tree to the next.



Planar MSSP

[Klein 2005]

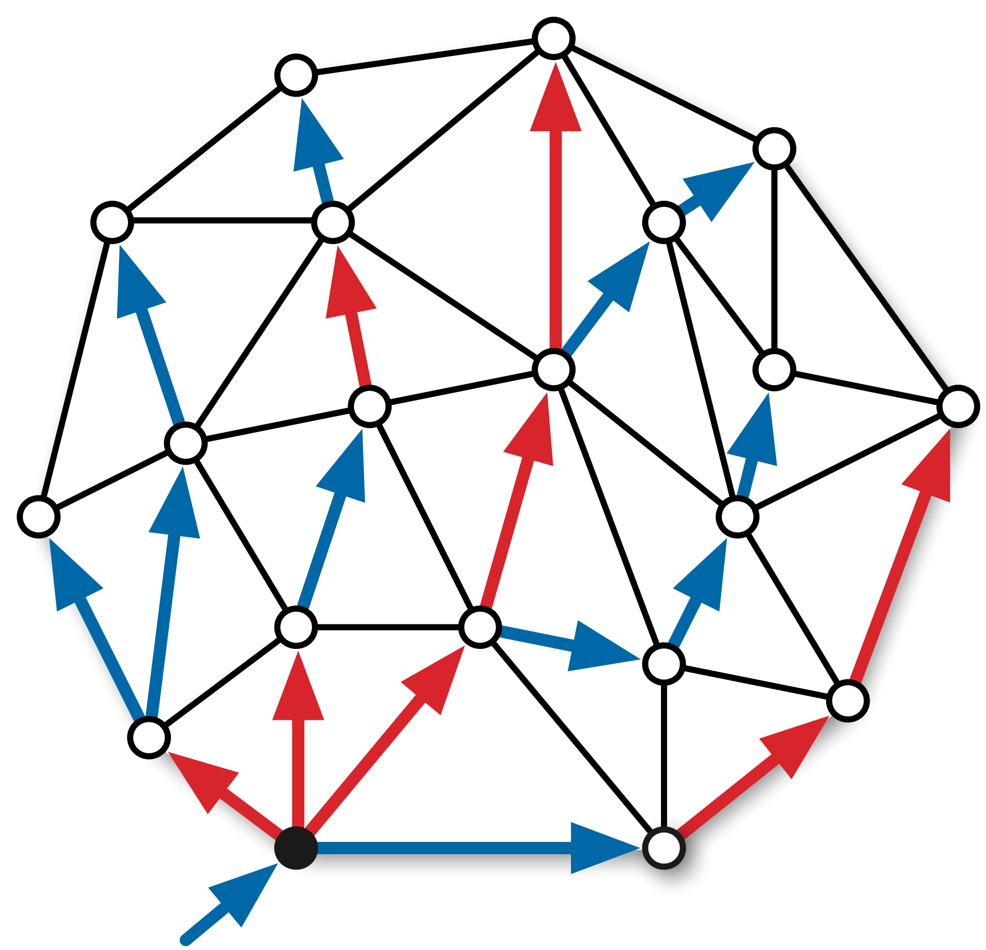
- In fact, we only need to compute the *first* shortest-path tree, followed by *changes* from each tree to the next.



Planar MSSP

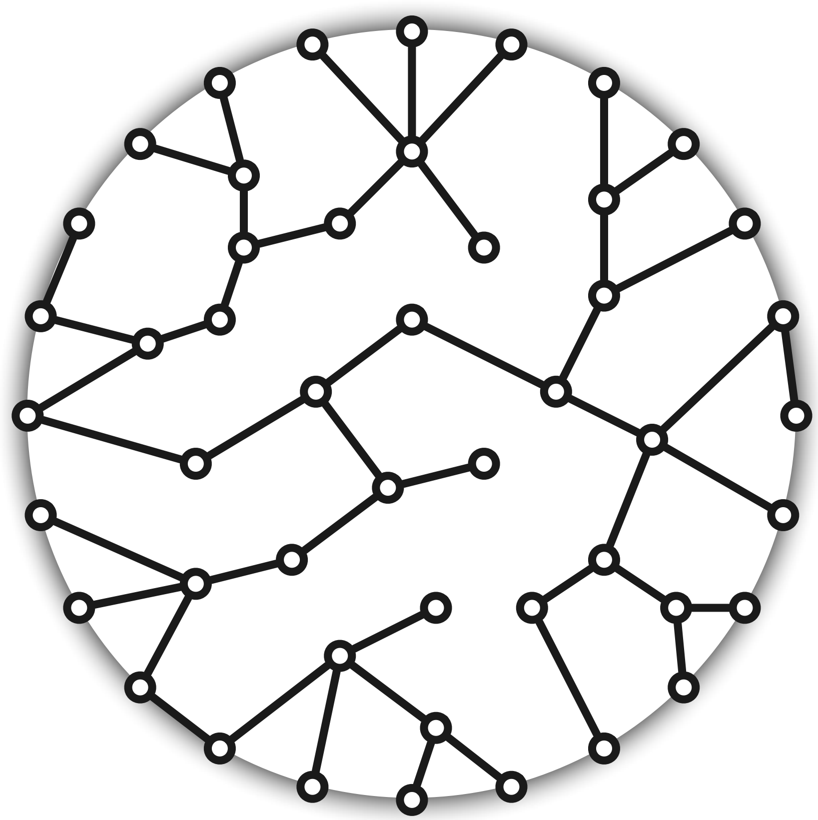
[Klein 2005]

- In fact, we only need to compute the *first* shortest-path tree, followed by *changes* from each tree to the next.



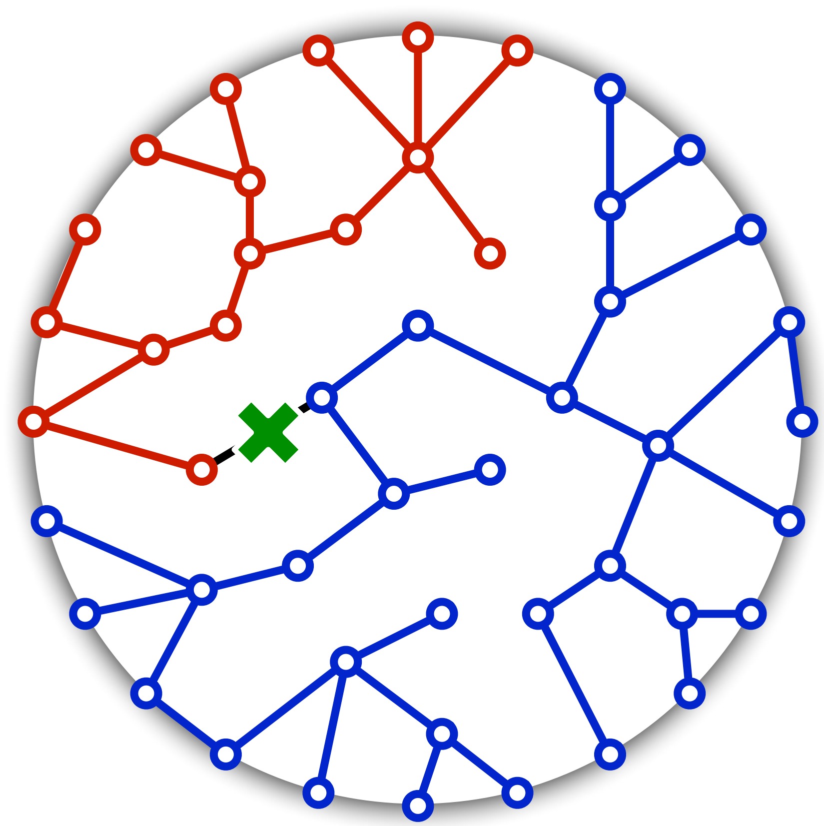
The disk-tree lemma

- Let T be any tree embedded on a closed disk. Vertices of T subdivide the boundary of the disk into intervals.
- Deleting any edge splits T into two subtrees R and B.
- At most two intervals have one end in R and the other in B.



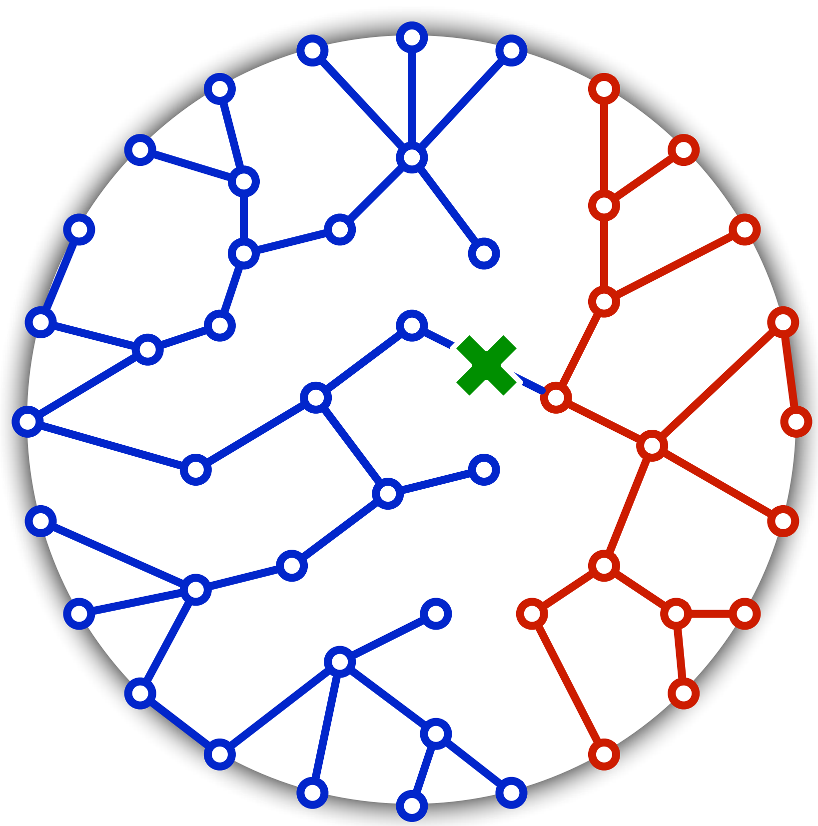
The disk-tree lemma

- Let T be any tree embedded on a closed disk. Vertices of T subdivide the boundary of the disk into intervals.
- Deleting any edge splits T into two subtrees R and B.
- At most two intervals have one end in R and the other in B.



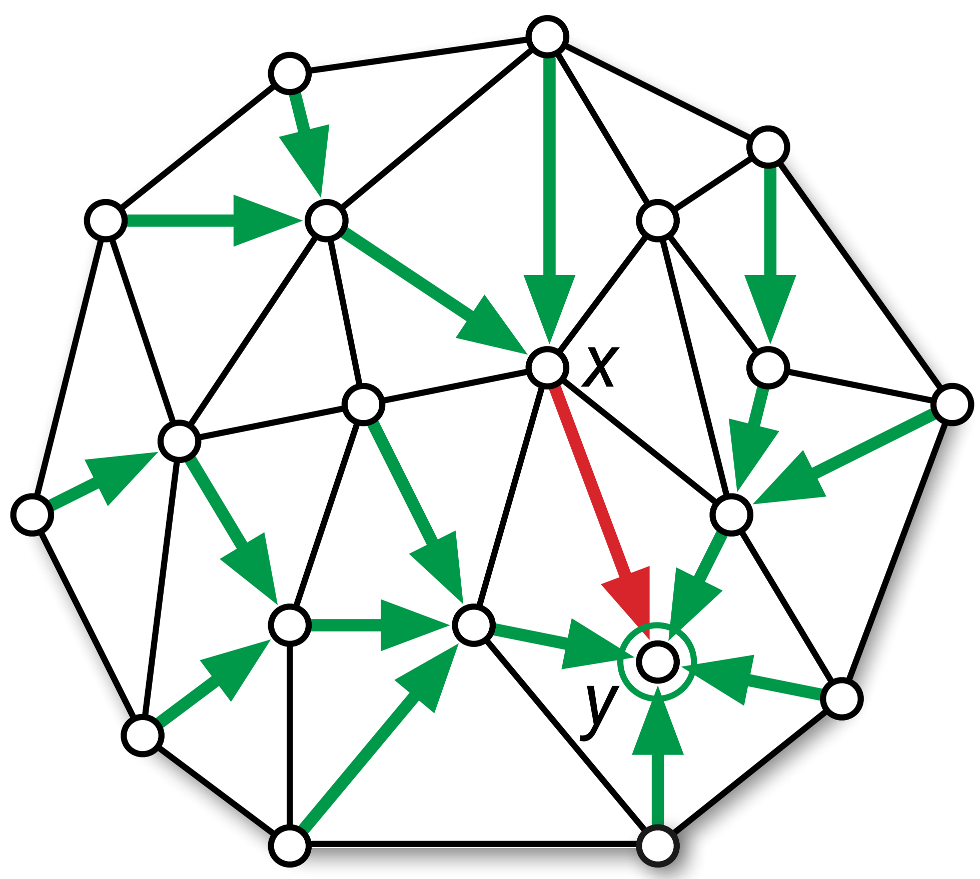
The disk-tree lemma

- Let T be any tree embedded on a closed disk. Vertices of T subdivide the boundary of the disk into intervals.
- Deleting any edge splits T into two subtrees R and B.
- At most two intervals have one end in R and the other in B.



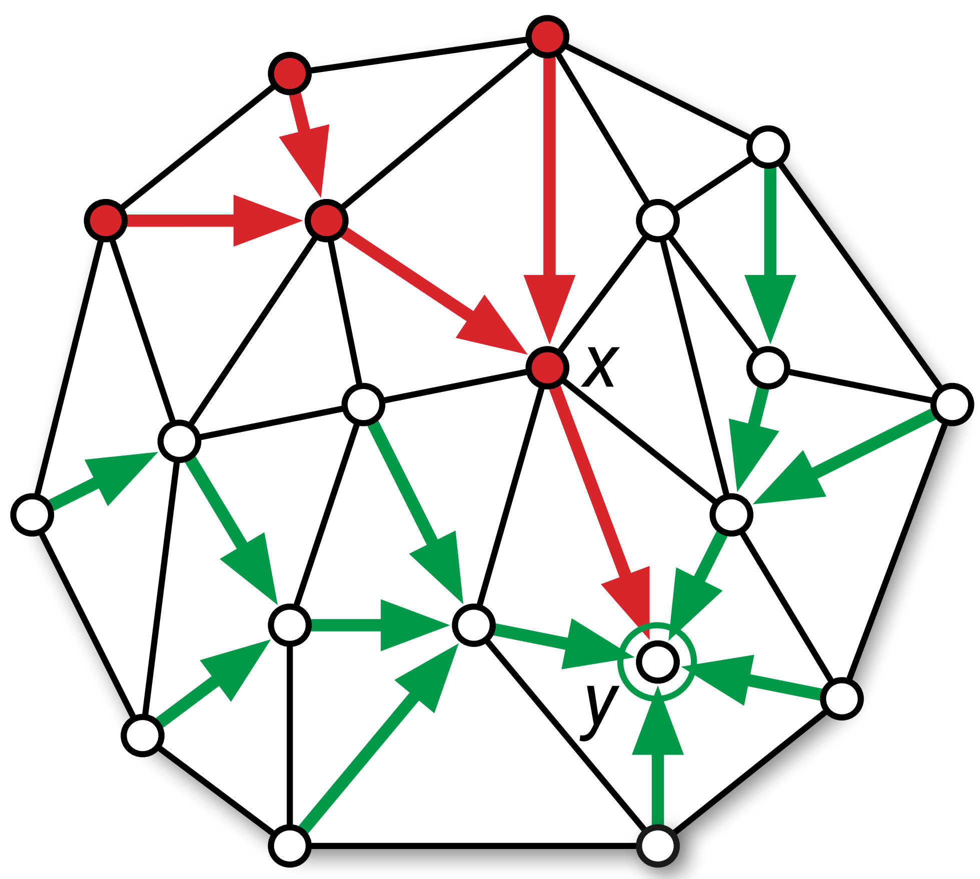
Number of pivots

- Each directed edge $x \to y$ pivots in *at most once*.
    - Consider the tree of shortest paths *ending at* y.



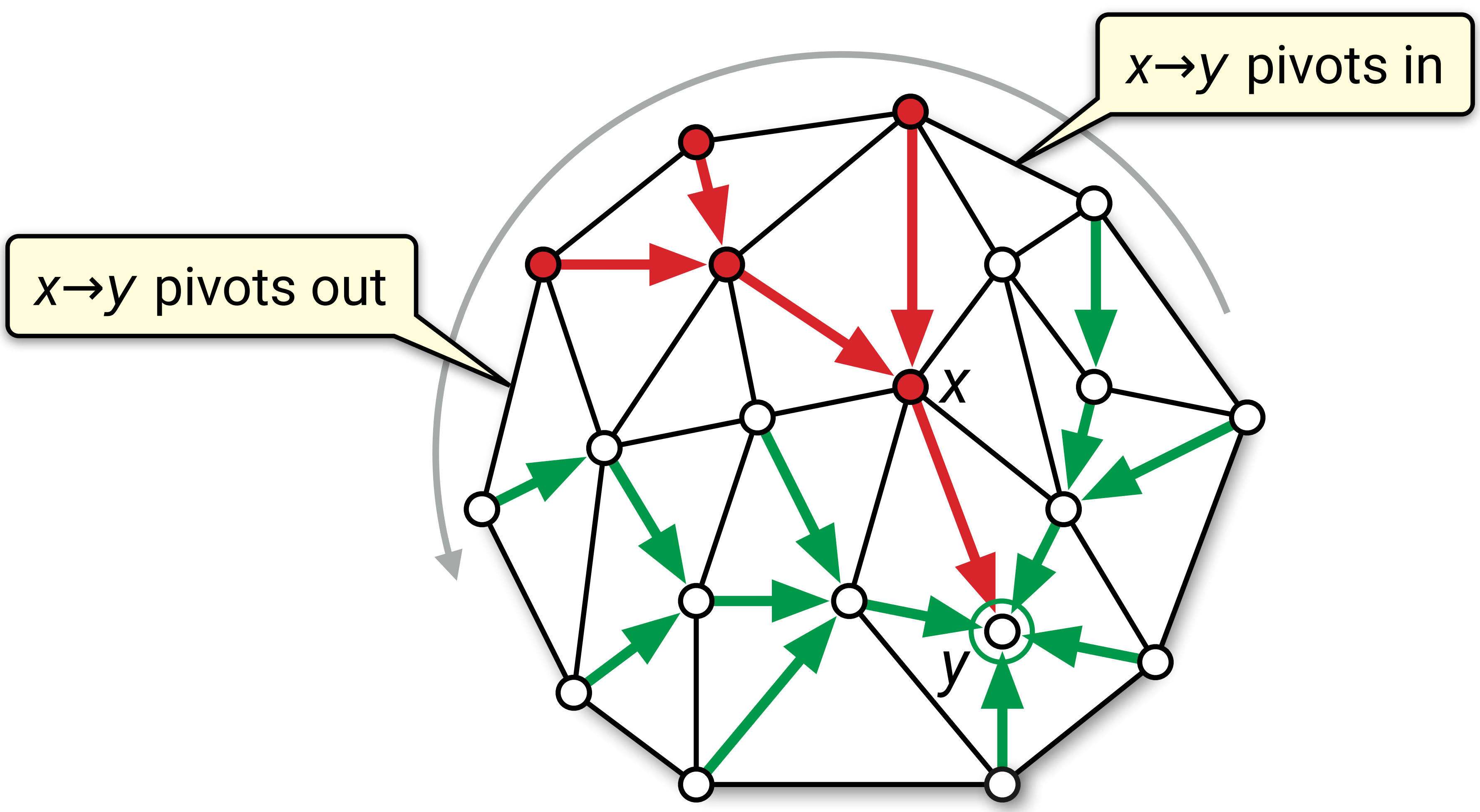
Number of pivots

- Each directed edge $x \to y$ pivots in *at most once*.
  - Consider the tree of shortest paths *ending at* y.



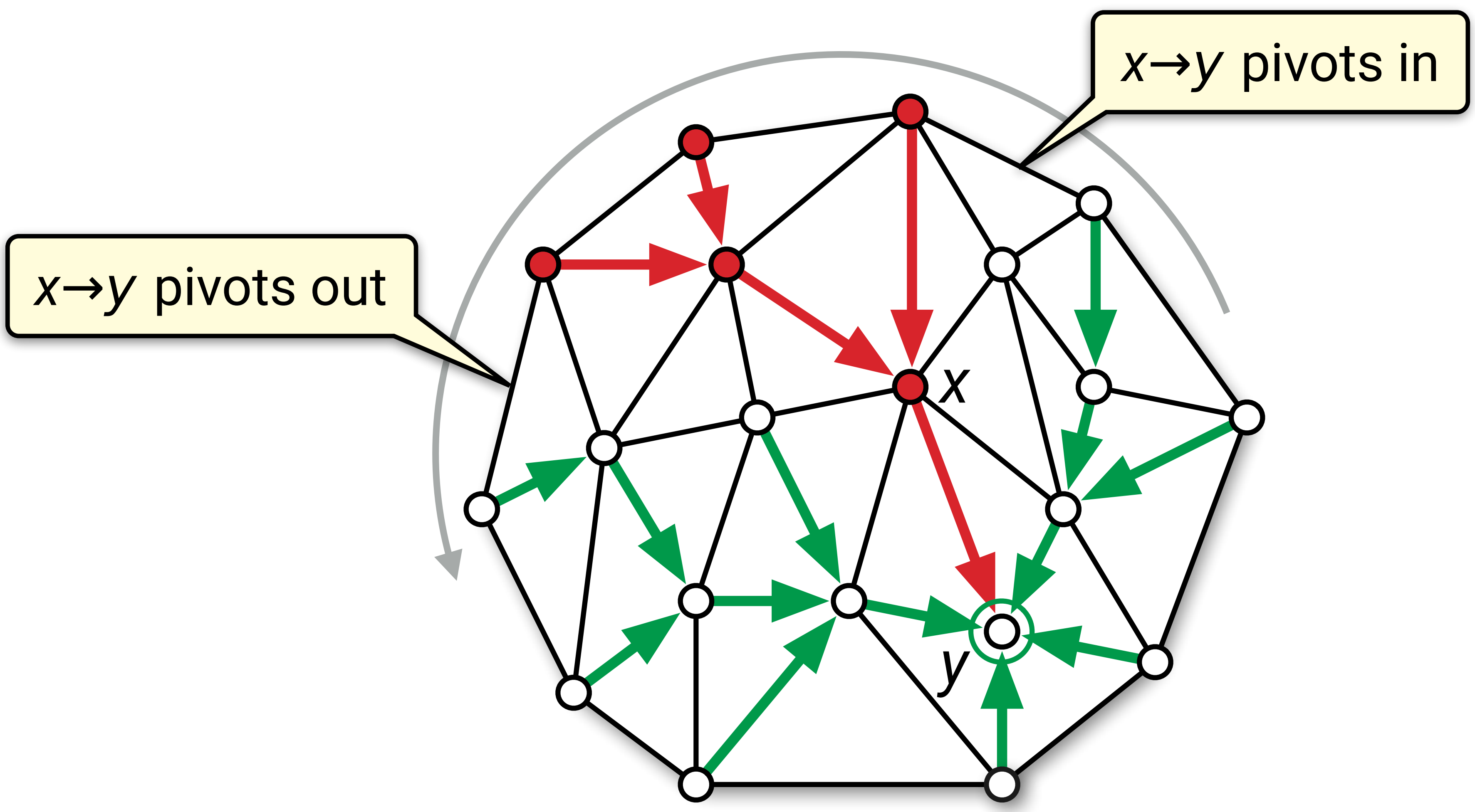
Number of pivots

- Each directed edge $x \rightarrow y$ pivots in *at most once*.
  - Consider the tree of shortest paths *ending at* y.



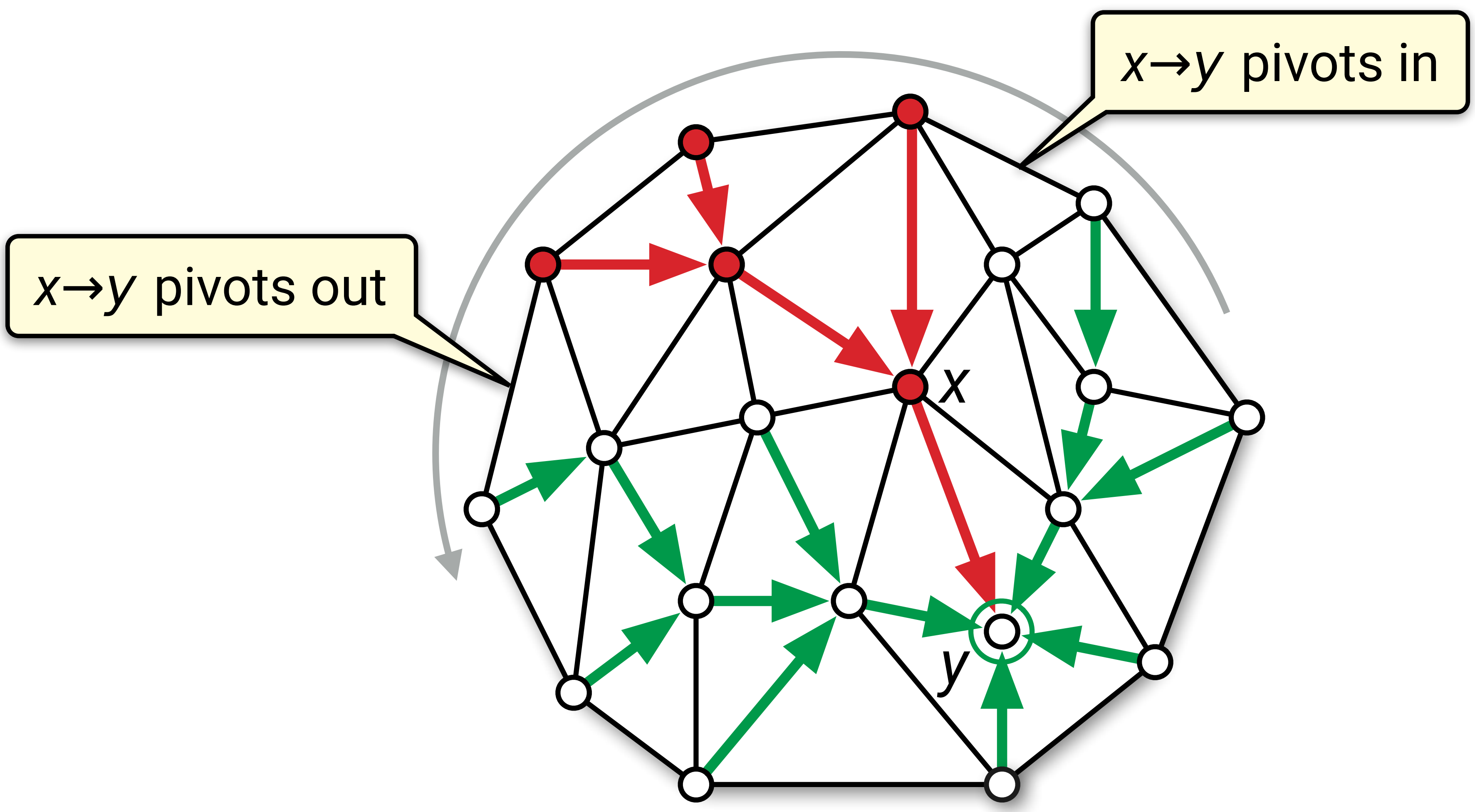
Number of pivots

- So the overall number of pivots is only O(n)!



Number of pivots

- So the overall number of pivots is only O(n)!
- But how do we find these pivots *quickly*?

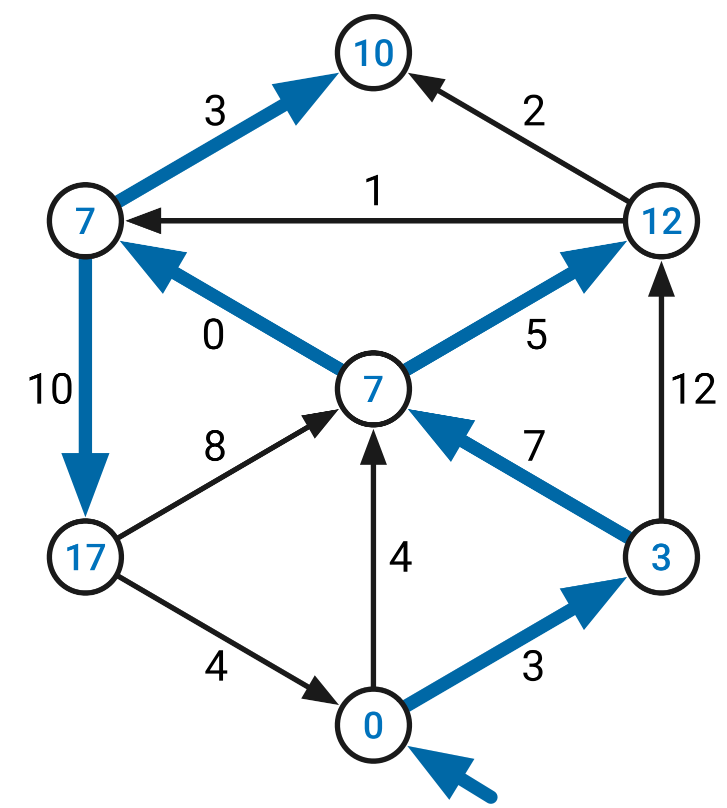


Please ask questions!

How shortest paths work

[Ford 1956]

- Input:
  - Directed graph G = (V, E)
  - length $\ell(u\rightarrow v)$ for each edge $u\rightarrow v$
  - *A source* vertex s.

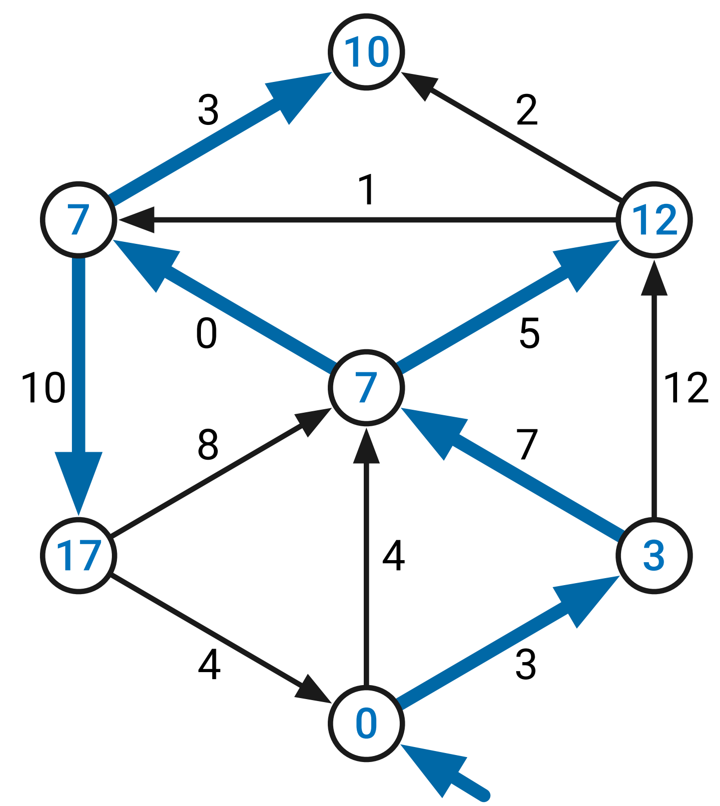


- Each vertex v maintains two values:
  - $dist(v)$ is the length of some path from s to v
  - $pred(v)$ is the next-to-last vertex of that path from s to v.

How shortest paths work

[Ford 1956]

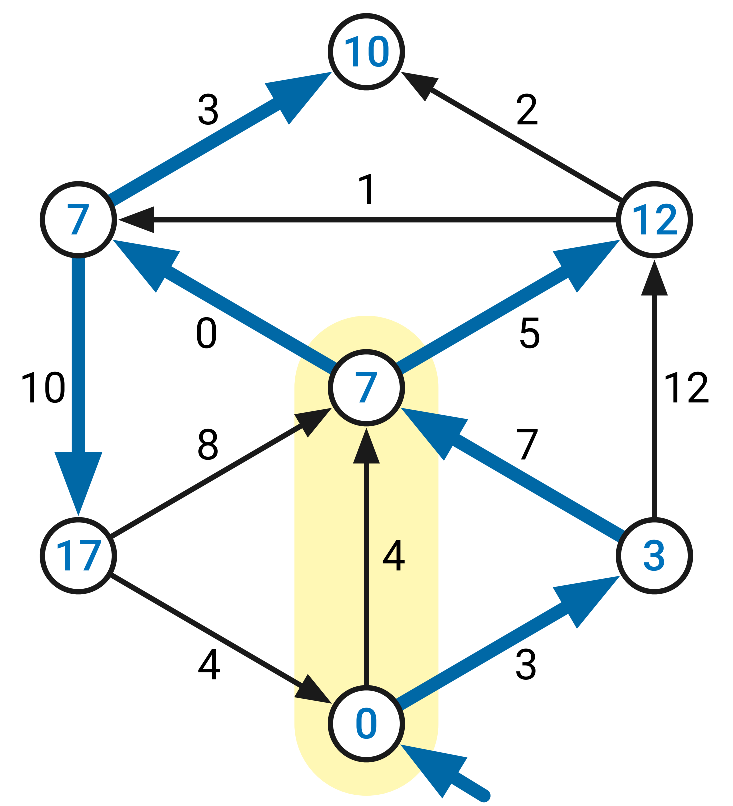
- Edge $u \rightarrow v$ is *tense* iff $dist(v) \geq dist(u) + \ell(u \rightarrow v)$.



How shortest paths work

[Ford 1956]

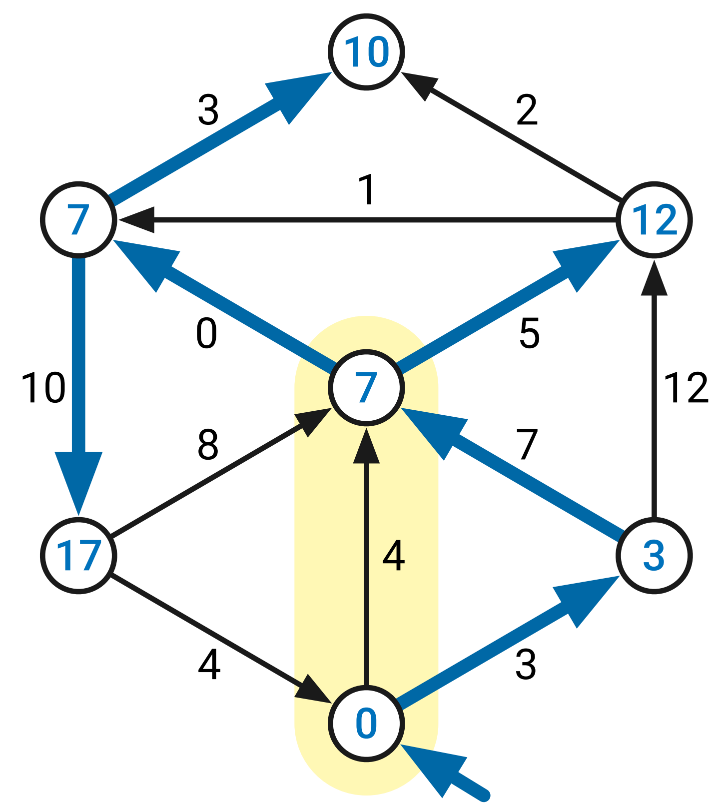
- Edge $u \to v$ is *tense* iff $dist(v) \geq dist(u) + \ell(u \to v)$.



How shortest paths work

[Ford 1956]

- Edge $u \rightarrow v$ is *tense* iff $dist(v) \geq dist(u) + \ell(u \rightarrow v)$.

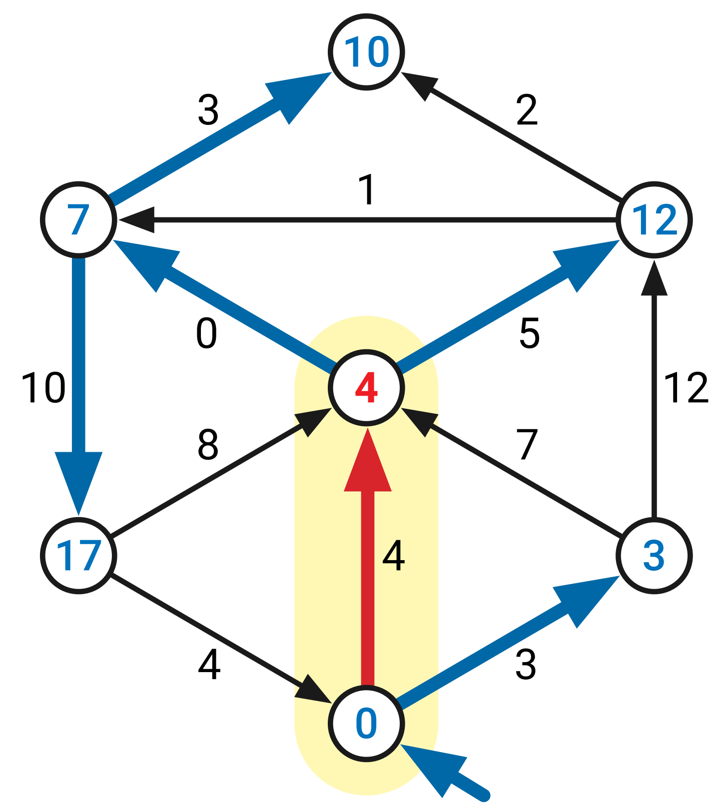


- To *relax* $u \rightarrow v$, set $dist(v) = dist(u) + \ell(u \rightarrow v)$ and $pred(v) = u$

How shortest paths work

[Ford 1956]

- Edge $u \to v$ is *tense* iff $dist(v) \geq dist(u) + \ell(u \to v)$.

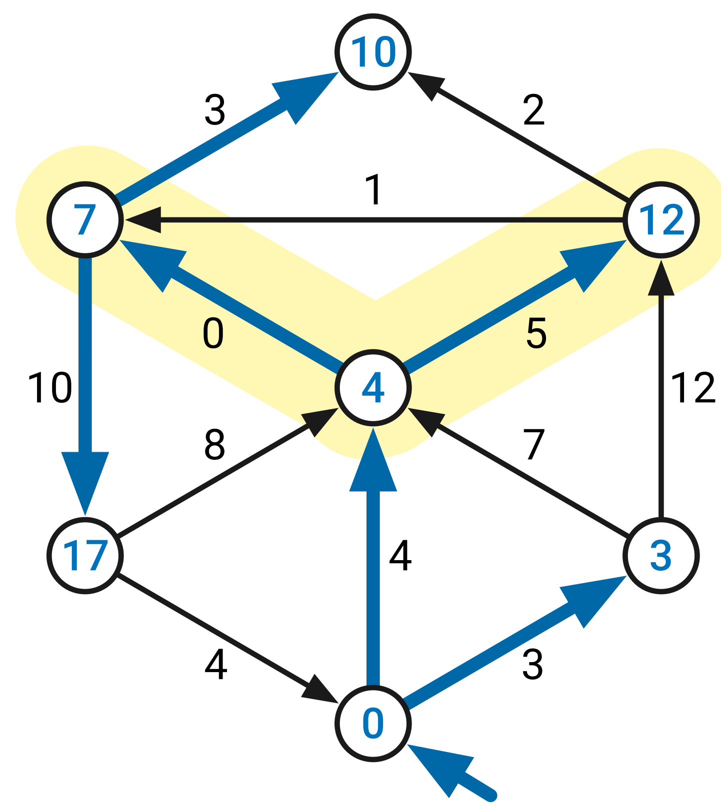


- To *relax* $u \to v$, set $dist(v) = dist(u) + \ell(u \to v)$ and $pred(v) = u$

How shortest paths work

[Ford 1956]

- Edge $u \rightarrow v$ is *tense* iff $dist(v) \geq dist(u) + \ell(u \rightarrow v)$.

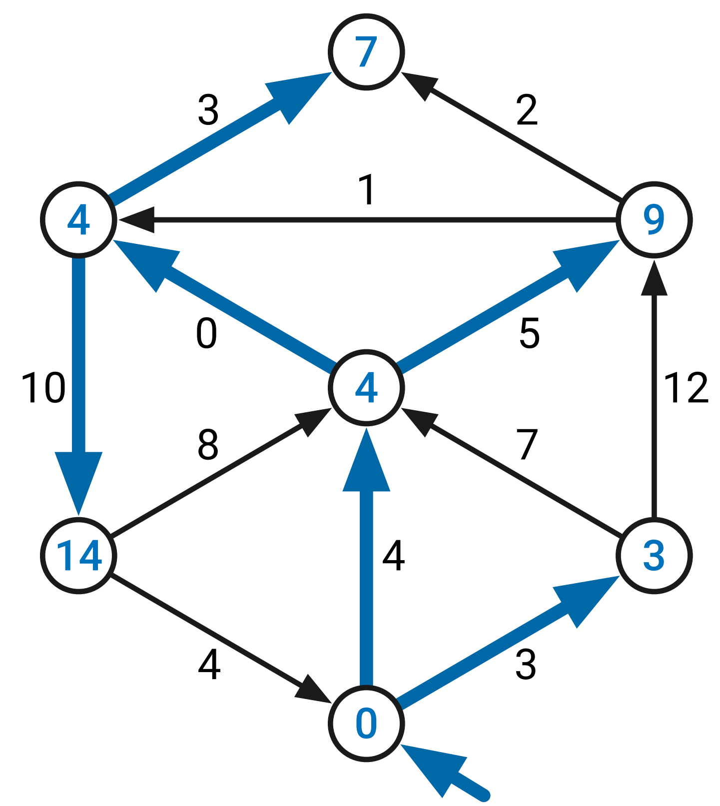


- To *relax* $u \rightarrow v$, set $dist(v) = dist(u) + \ell(u \rightarrow v)$ and $pred(v) = u$

How shortest paths work

[Ford 1956]

- Edge *u*→*v* is *tense* iff *dist*(*v*) ≥ *dist*(*u*) + *ℓ*(*u*→*v*).



- If *no* edges are tense, then *dist*(*v*) is the length of the *shortest* path from *s* to *v*, for every vertex *v*.

Back to MSSP

[Cabello Chambers Erickson 2013]

- Maintain the shortest path tree rooted at a point s that is moving *continuously* around the outer face.
- Also maintain the *slack* of each edge $u{\to}v$:

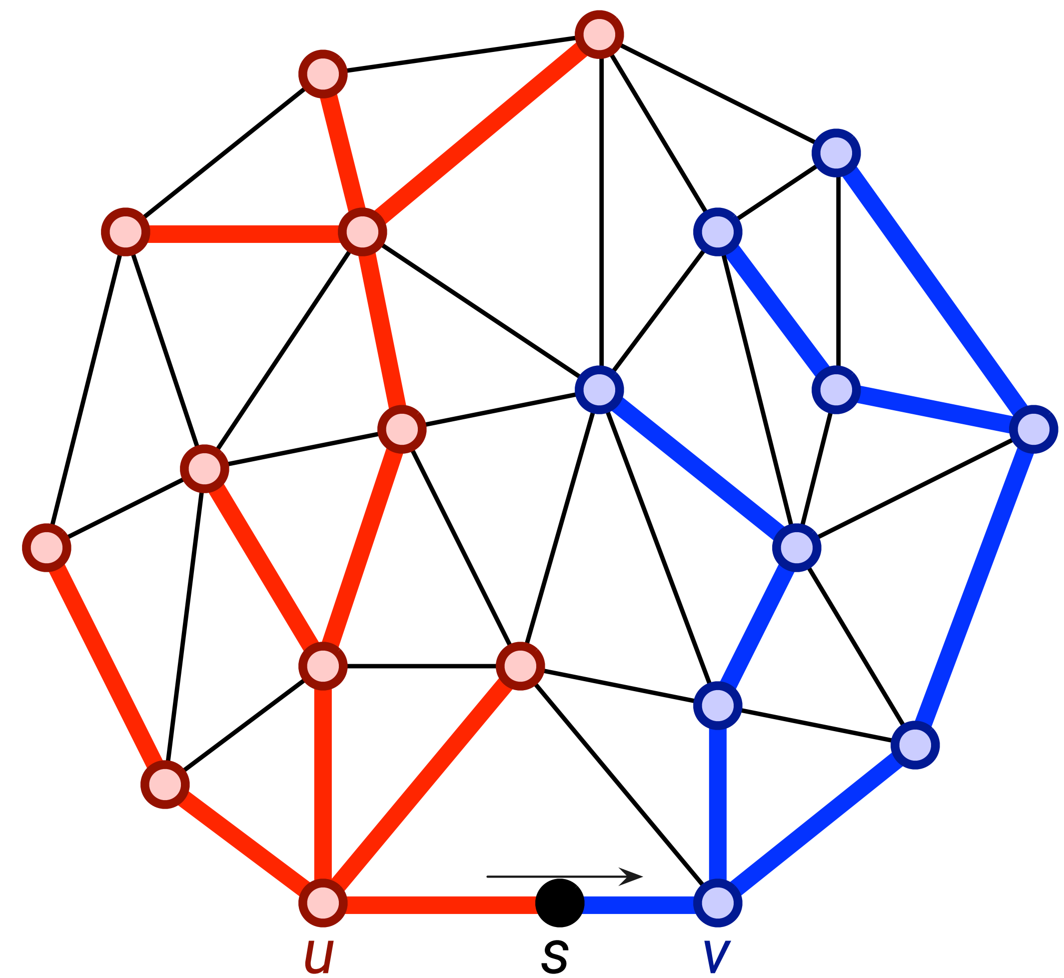
$$slack(u{\to}v) := dist(u) + \ell(u{\to}v) - dist(v)$$

- Distances and slacks change continuously with s, but in a controlled manner.
- The shortest path tree is correct as long as $slack(u{\to}v)>0$ for every edge $u{\to}v$.

Distance and slack changes

[Doppler 1842]
[Fizeau 1848]

- Red: dist growing
- Blue: dist shrinking

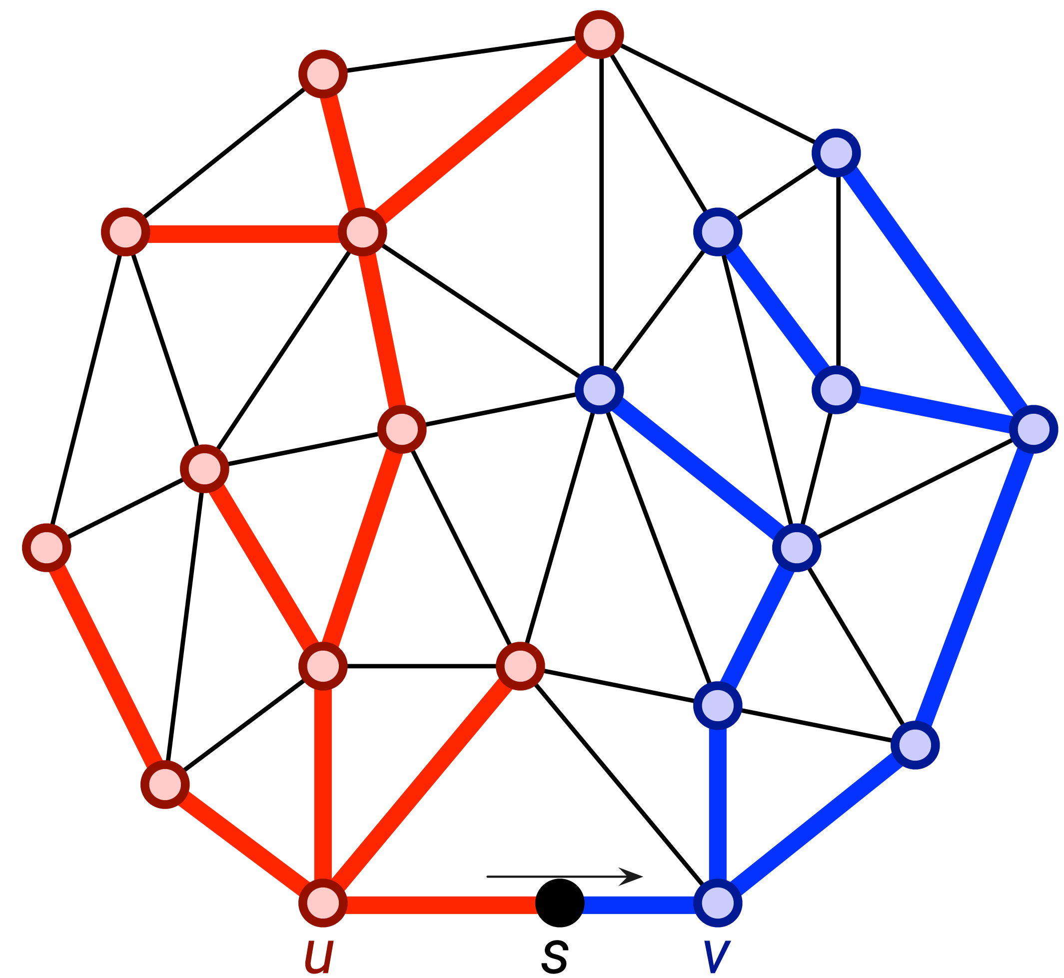


Distance and slack changes

[Doppler 1842]
[Fizeau 1848]

- Red: dist growing
- Blue: dist shrinking

- Red→red: slack constant
- Blue→blue: slack constant
- Red→blue: slack growing
- Blue→red: slack shrinking

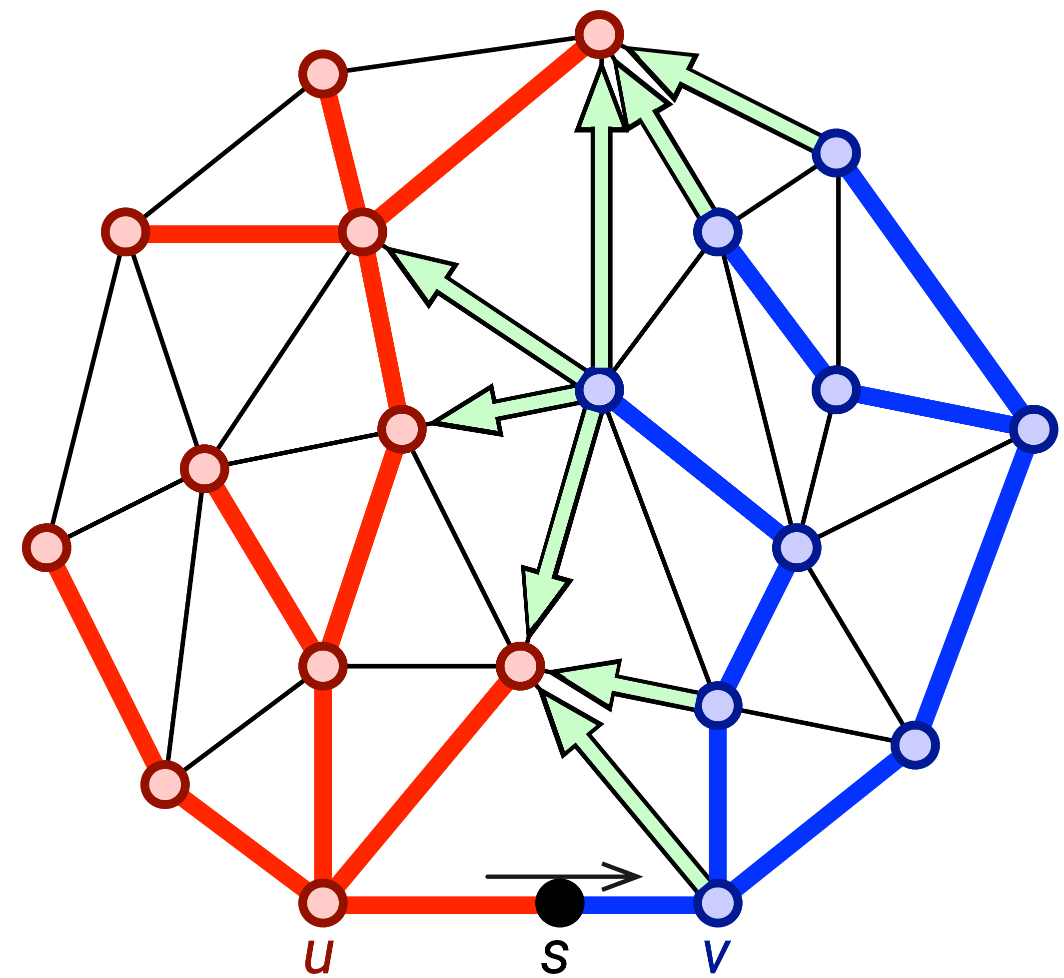


Distance and slack changes

[Doppler 1842]
[Fizeau 1848]

- Red: dist growing
- Blue: dist shrinking

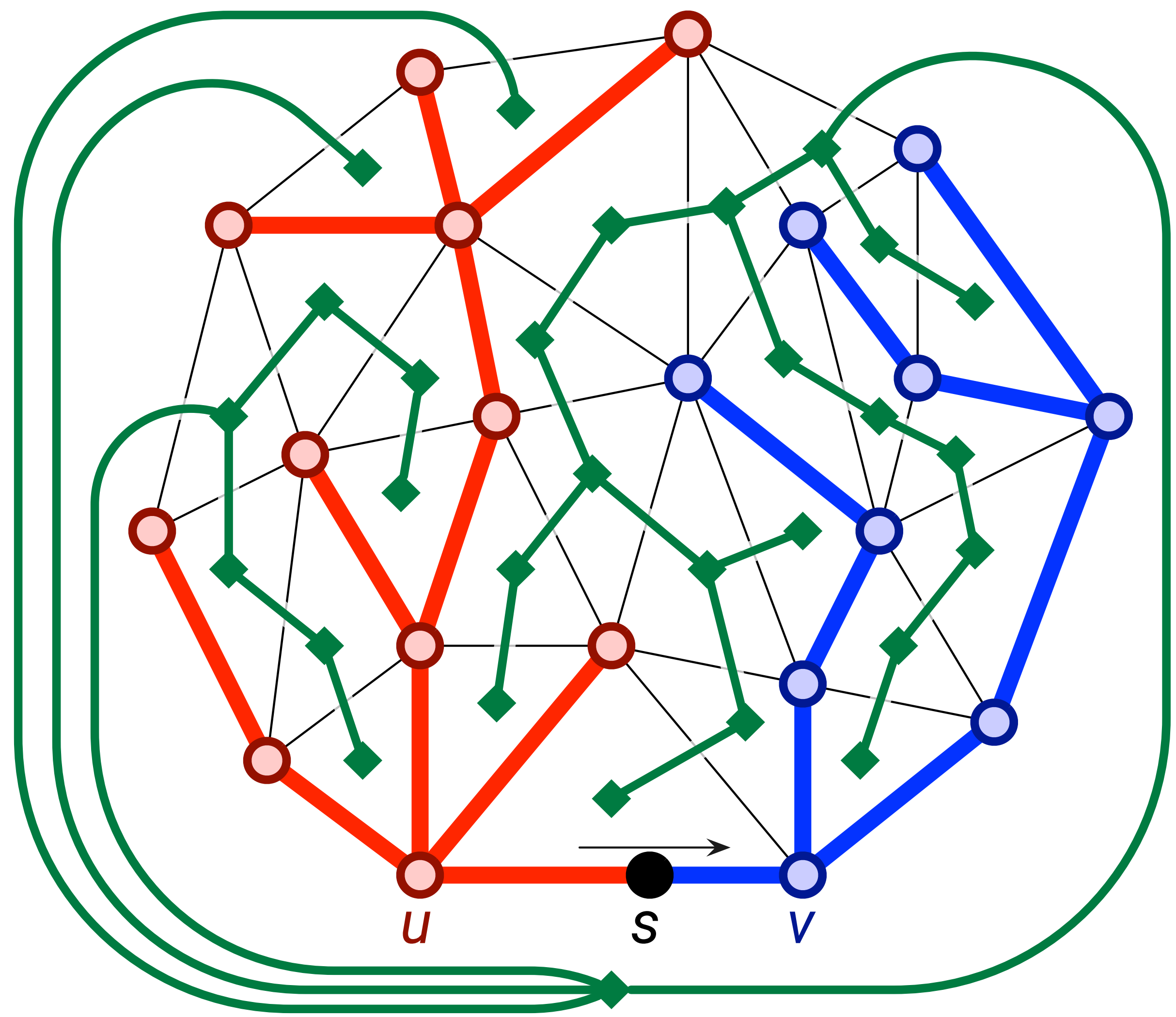
- Red→red: slack constant
- Blue→blue: slack constant
- Red→blue: slack growing
- Blue→red: slack shrinking
  - *active* edges



Tree-cotree decomposition

[von Staudt 1847]
[Whitney 1932]
[Dehn 1936]

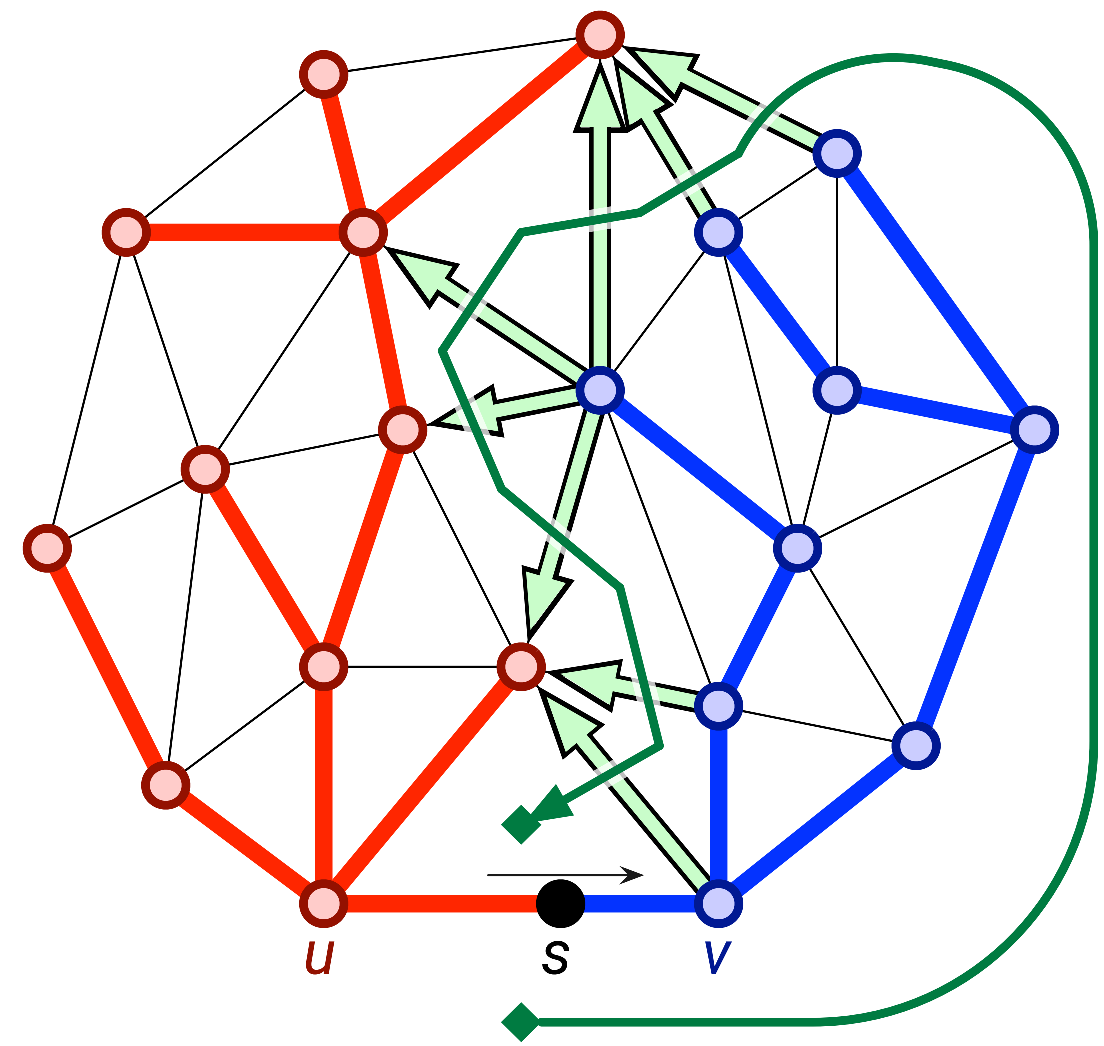
- Complementary dual spanning tree $C^* = (G \backslash T)^*$
- Red and blue subtrees are separated by a path in C^*
- Active edges are dual to edges in this path.



Tree-cotree decomposition

[von Staudt 1847]
[Whitney 1932]
[Dehn 1936]

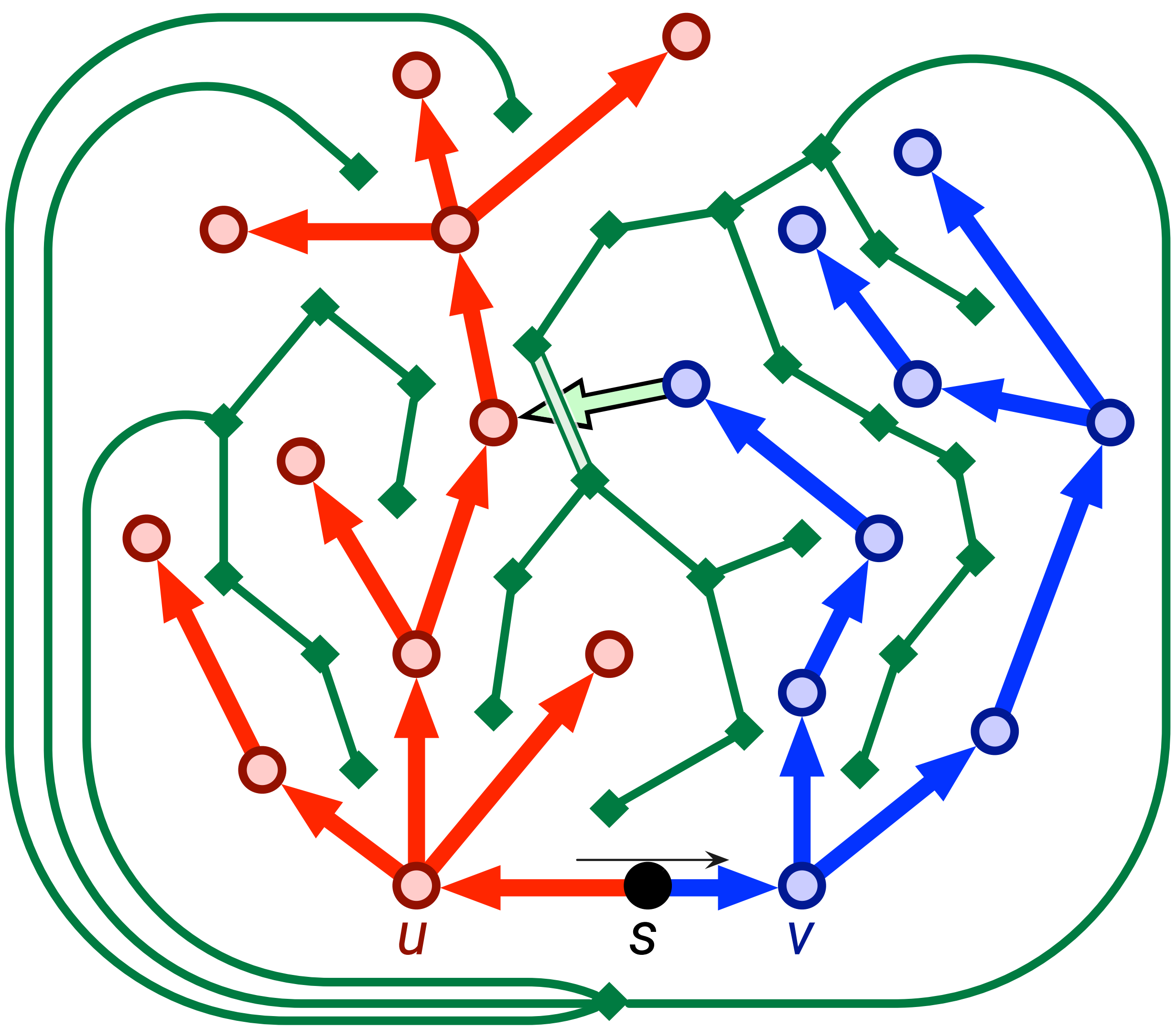
- Complementary dual spanning tree $C^* = (G \backslash T)^*$
- Red and blue subtrees are separated by a path in C^*
- Active edges are dual to edges in this path.



Pivot

[Ford 1956]

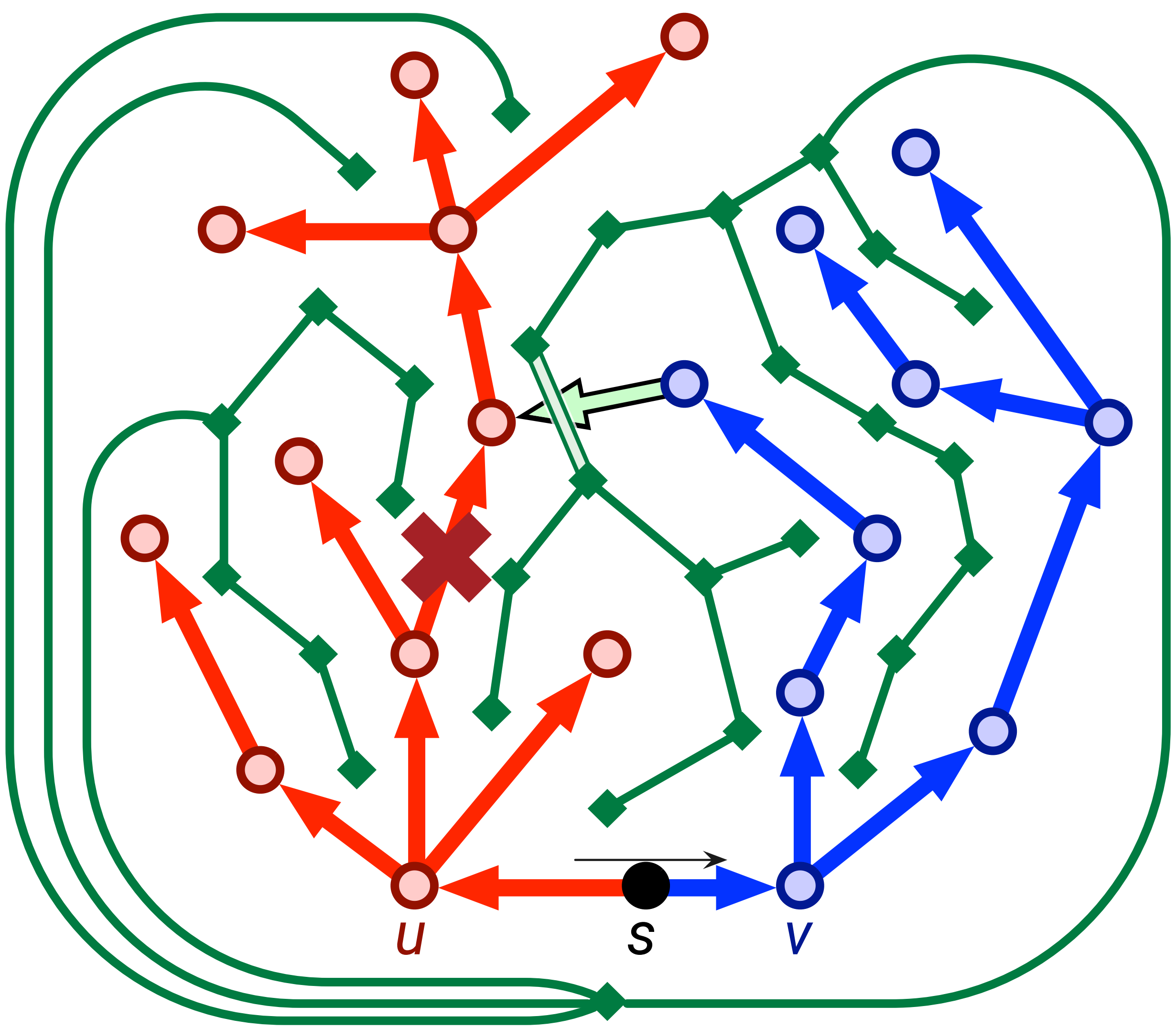
- When *slack*(*u*→*v*) becomes 0, relax *u*→*v*
  - Delete *pred*(*v*)→*v* from *T*
  - Insert *u*→*v* into *T*.
  - Delete (*u*→*v*)* from *C**.
  - Insert (*pred*(*v*)→*v*)* into *C**
  - Set *pred*(*u*) := *v*



Pivot

[Ford 1956]

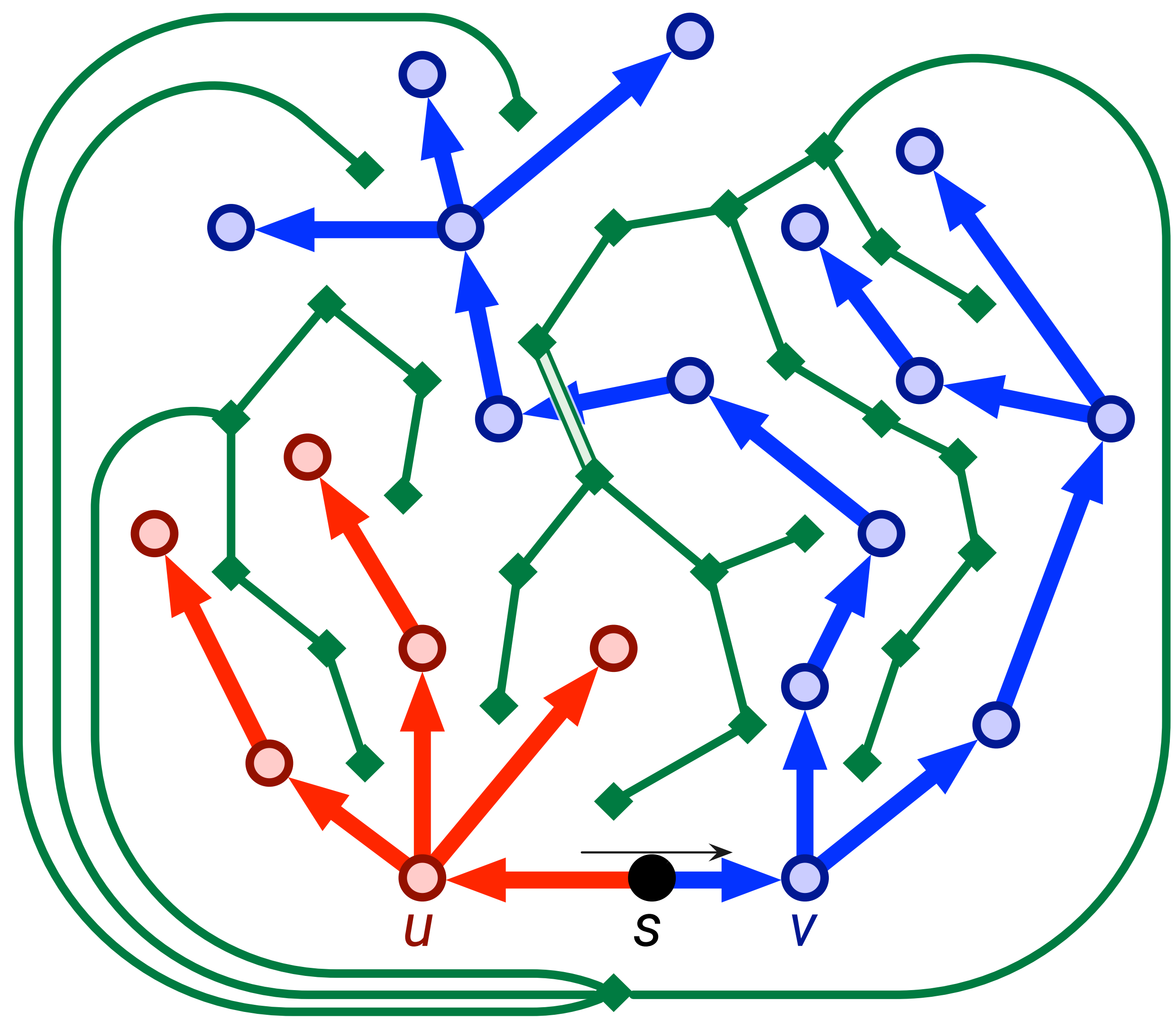
- When *slack*(u→v) becomes 0, relax u→v
  - Delete *pred*(v)→v from T
  - Insert u→v into T.
  - Delete (u→v)* from C*.
  - Insert (*pred*(v)→v)* into C*
  - Set *pred*(u) := v



Pivot

[Ford 1956]

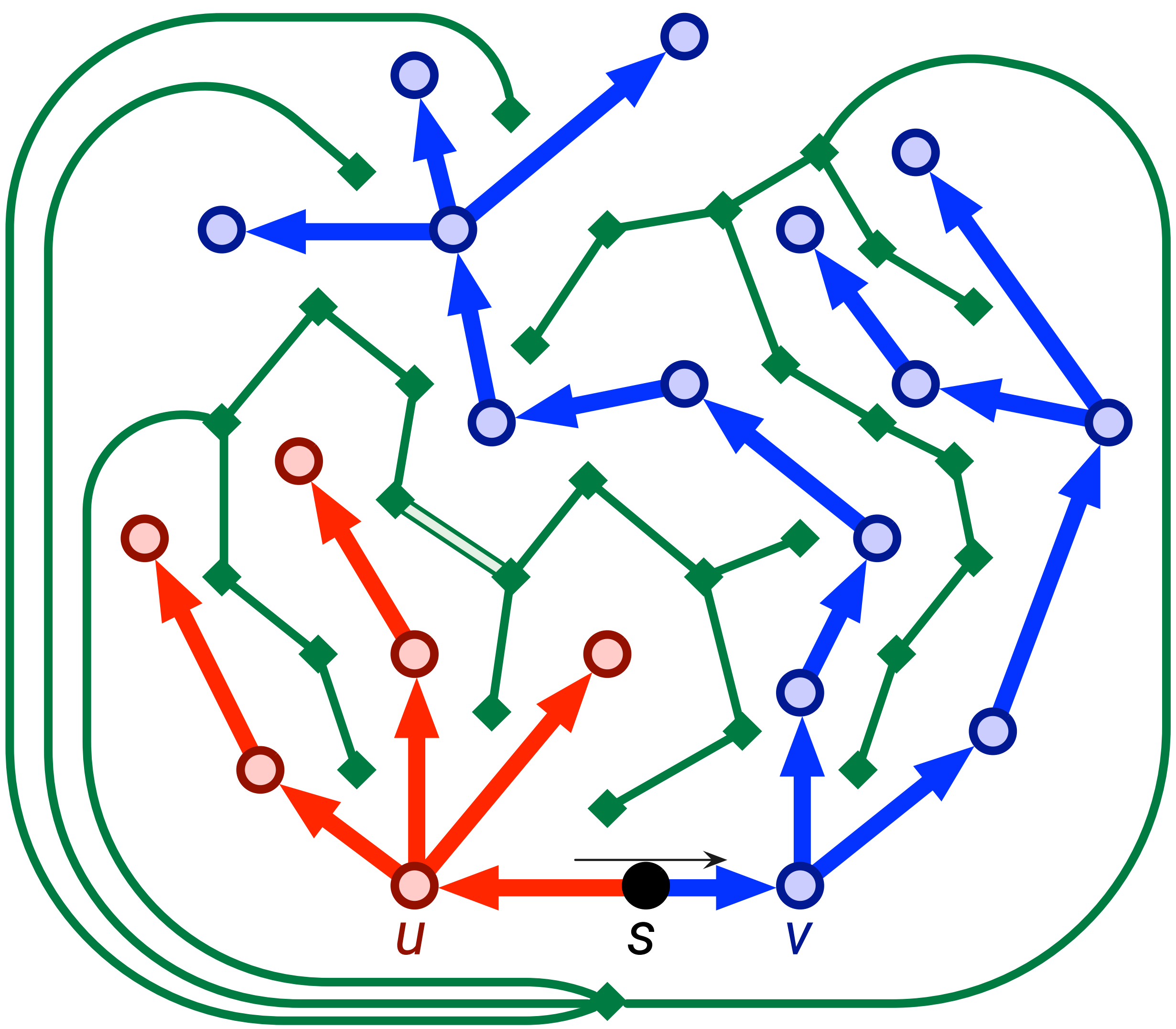
- When *slack*(u→v) becomes 0, relax u→v
  - Delete *pred*(v)→v from T
  - Insert u→v into T.
  - Delete (u→v)* from C*.
  - Insert (*pred*(v)→v)* into C*
  - Set *pred*(u) := v



Pivot

[Ford 1956]

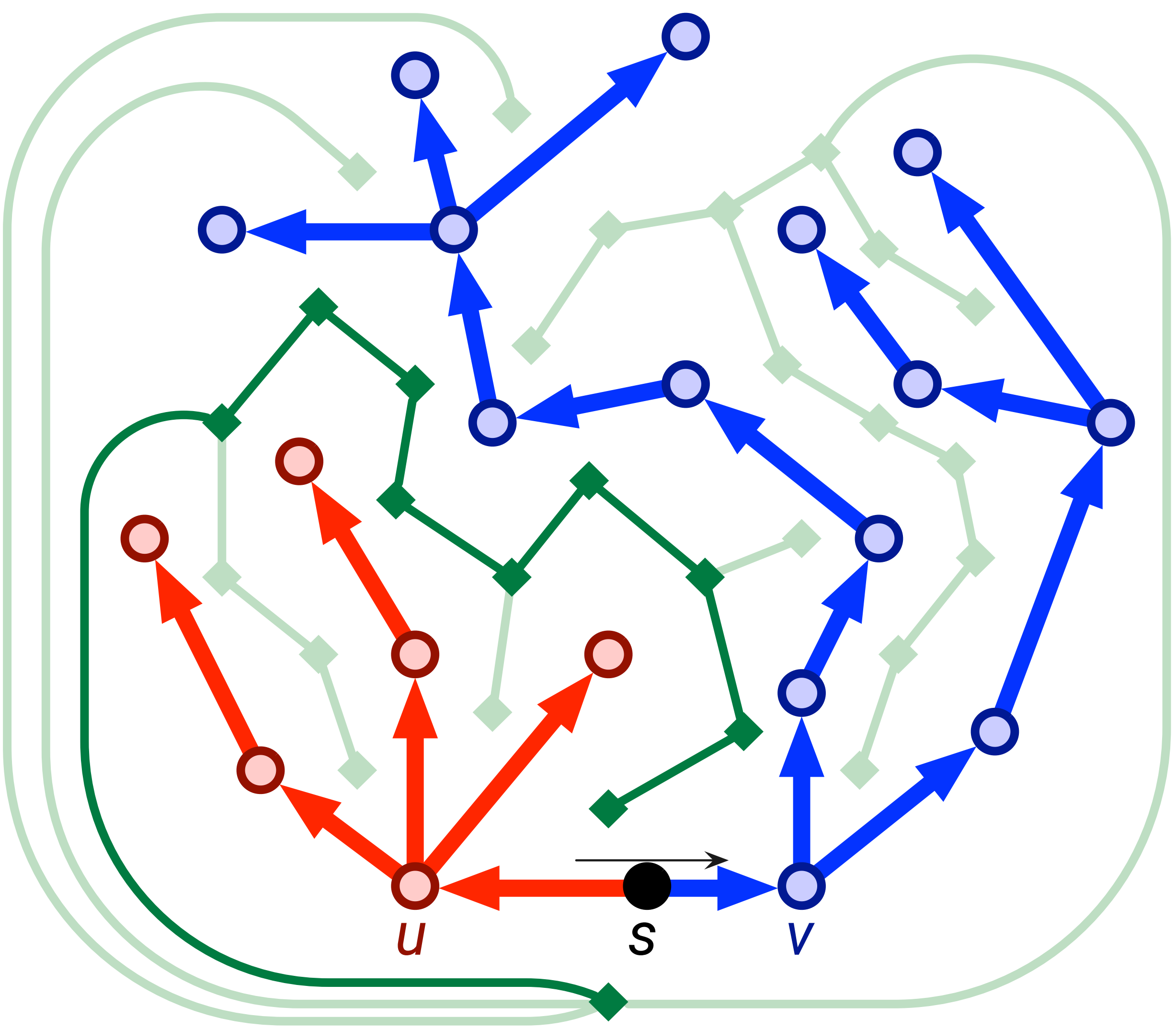
- When *slack*(*u*→*v*) becomes 0, relax *u*→*v*
  - Delete *pred*(*v*)→*v* from *T*
  - Insert *u*→*v* into *T*.
  - Delete (*u*→*v*)* from *C**.
  - Insert (*pred*(*v*)→*v*)* into *C**
  - Set *pred*(*u*) := *v*



Pivot

[Ford 1956]

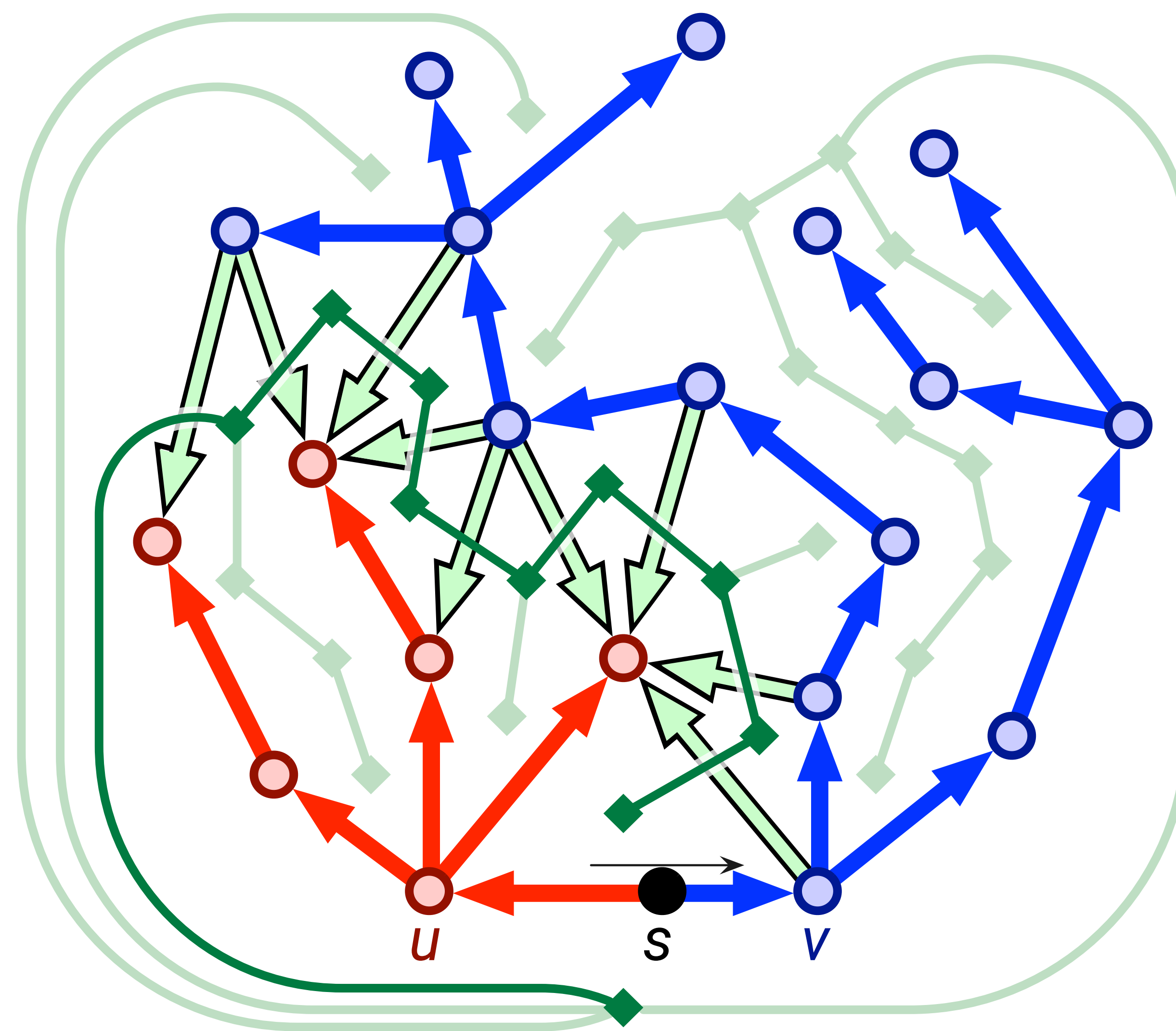
- When *slack*(*u*→*v*) becomes 0, relax *u*→*v*
  - Delete *pred*(*v*)→*v* from *T*
  - Insert *u*→*v* into *T*.
  - Delete (*u*→*v*)* from *C**.
  - Insert (*pred*(*v*)→*v*)* into *C**
  - Set *pred*(*u*) := *v*



Pivot

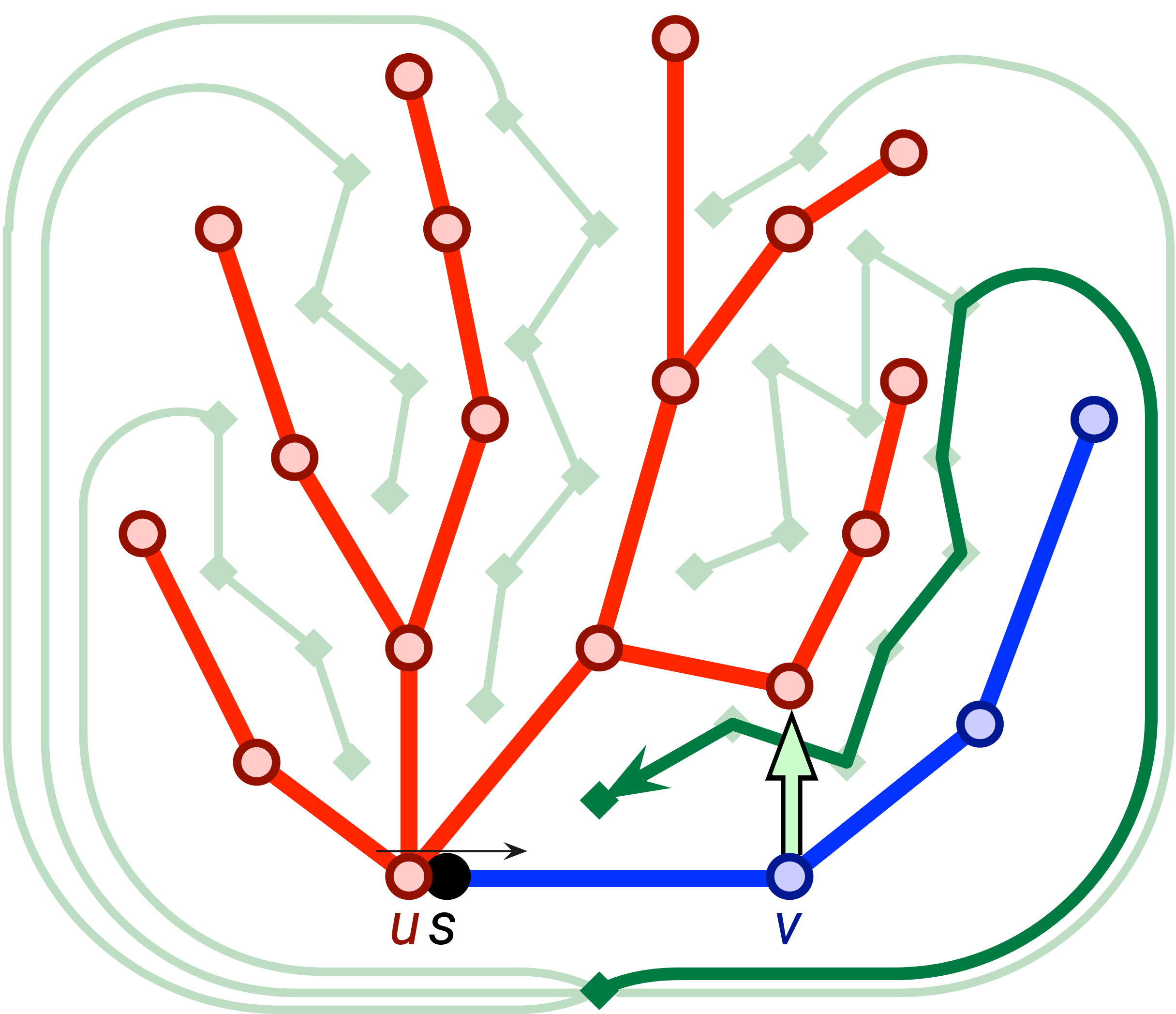
[Ford 1956]

- When *slack*(*u*→*v*) becomes 0, relax *u*→*v*
  - Delete *pred*(*v*)→*v* from *T*
  - Insert *u*→*v* into *T*.
  - Delete (*u*→*v*)* from *C**.
  - Insert (*pred*(*v*)→*v*)* into *C**
  - Set *pred*(*u*) := *v*



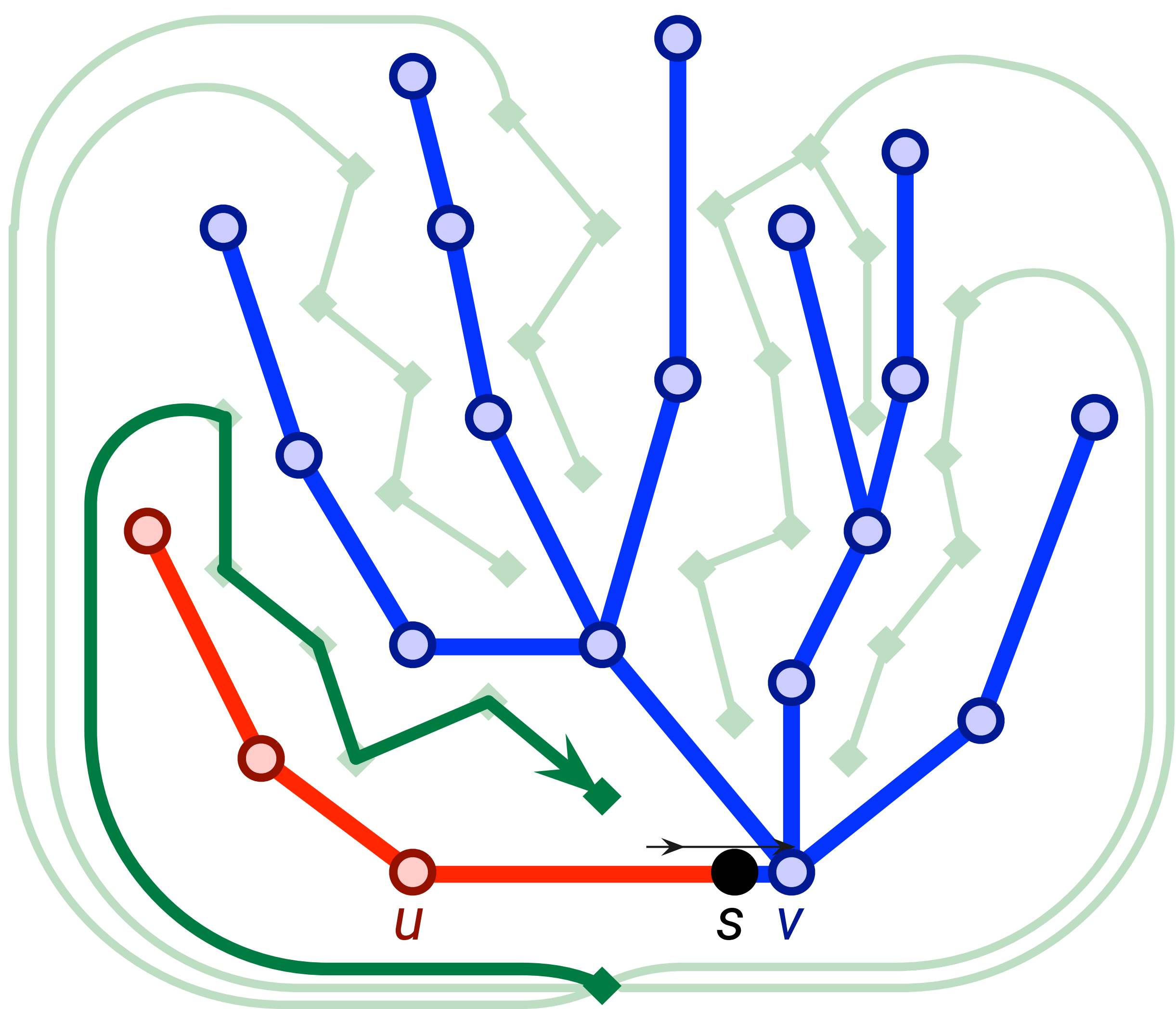
Pivots

- Vertices can only change from red to blue.
- So any edge that pivots into T stays in T.



Pivots

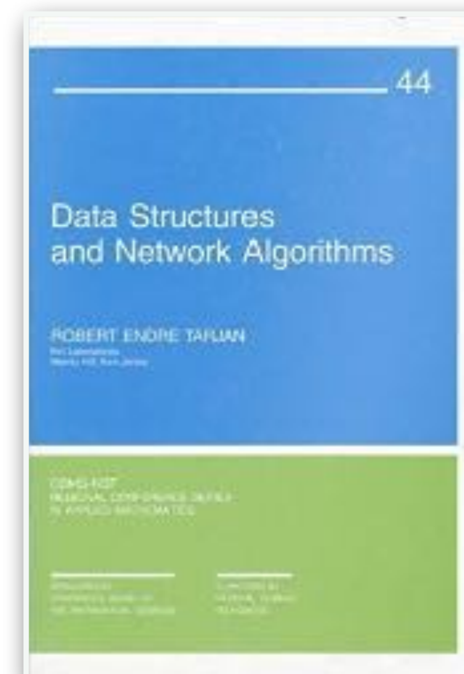
- Vertices can only change from red to blue.
- So any edge that pivots into T stays in T.



Fast implementation

[Sleator Tarjan 1983]
⋮
[Tarjan Werneck 2005]

- We maintain T and C^* in *dynamic forest* data structures that support the following operations in $O(\log n)$ amortized time:
  - Remove and insert edges:
    - Cut(uv), Link(u,v)
  - Maintain distances at vertices of T:
    - GetNodeValue(v), AddSubtree(Δ, v)
  - Maintain slacks at edges of C^*:
    - GetDartValue($u \to v$), AddPath(Δ, u, v), MinPath(u, v)



- So we can identify and execute each pivot in $O(\log n)$ amortized time.

Planar MSSP summary

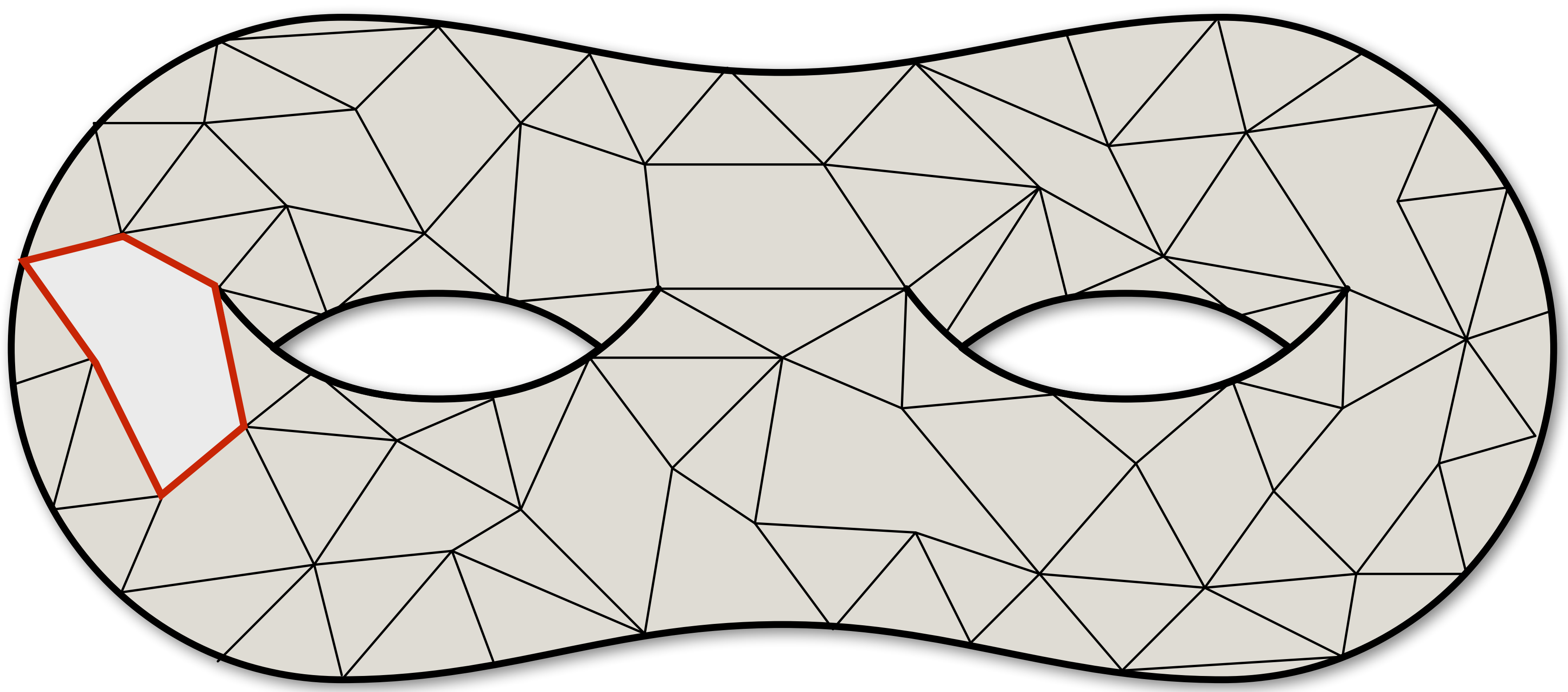
[Klein 2005]

- We can (implicitly) compute distances from every boundary vertex to every vertex in any planar map in $O(n \log n)$ time!
- More accurately: Given k vertex pairs, where one vertex of each pair is on the boundary, we can compute those k shortest-path distances in $O(n \log n + k \log n)$ time.

Please ask questions!

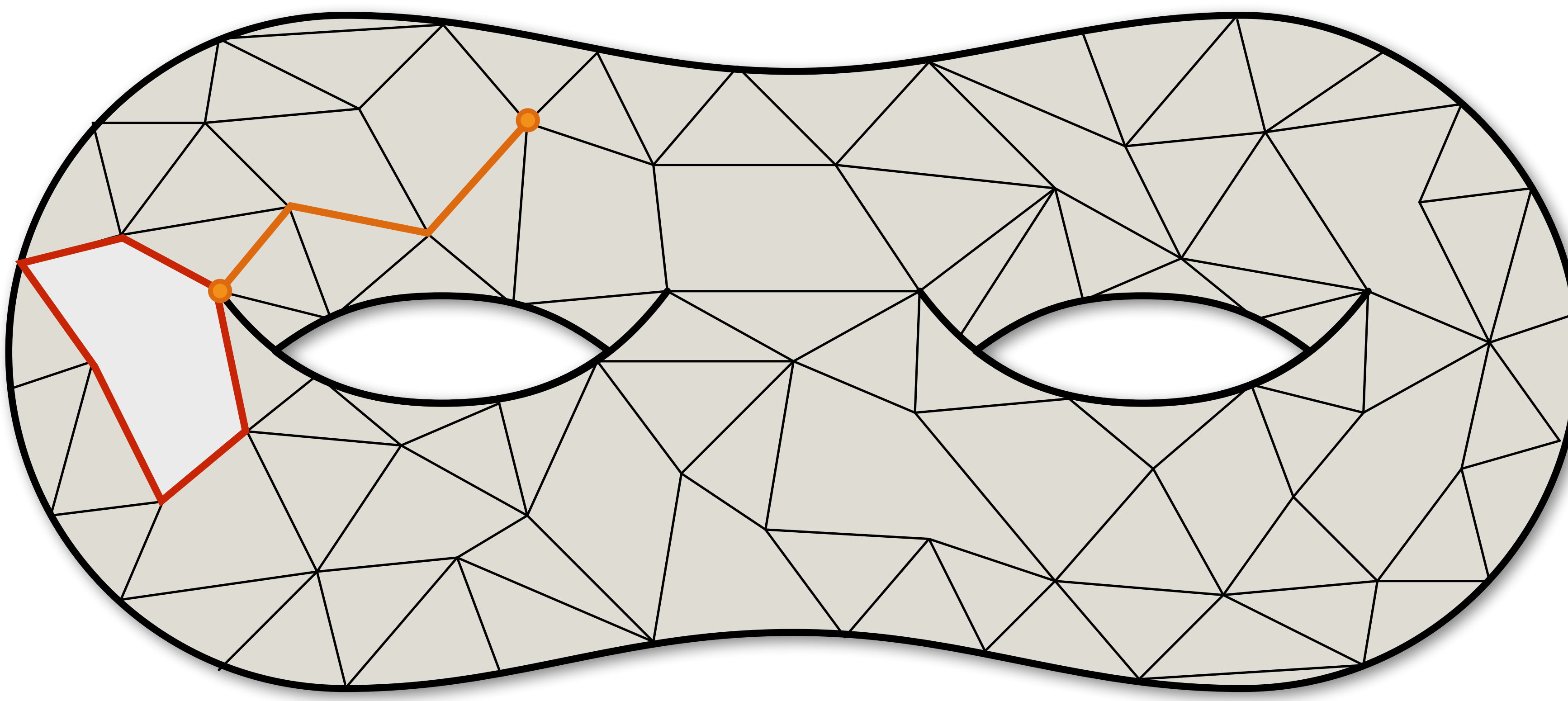
Back to surfaces

- Let Σ be any surface map with genus g. Fix a face f of Σ.
- We want to compute the shortest path trees rooted at every vertex of some “outer” face f.



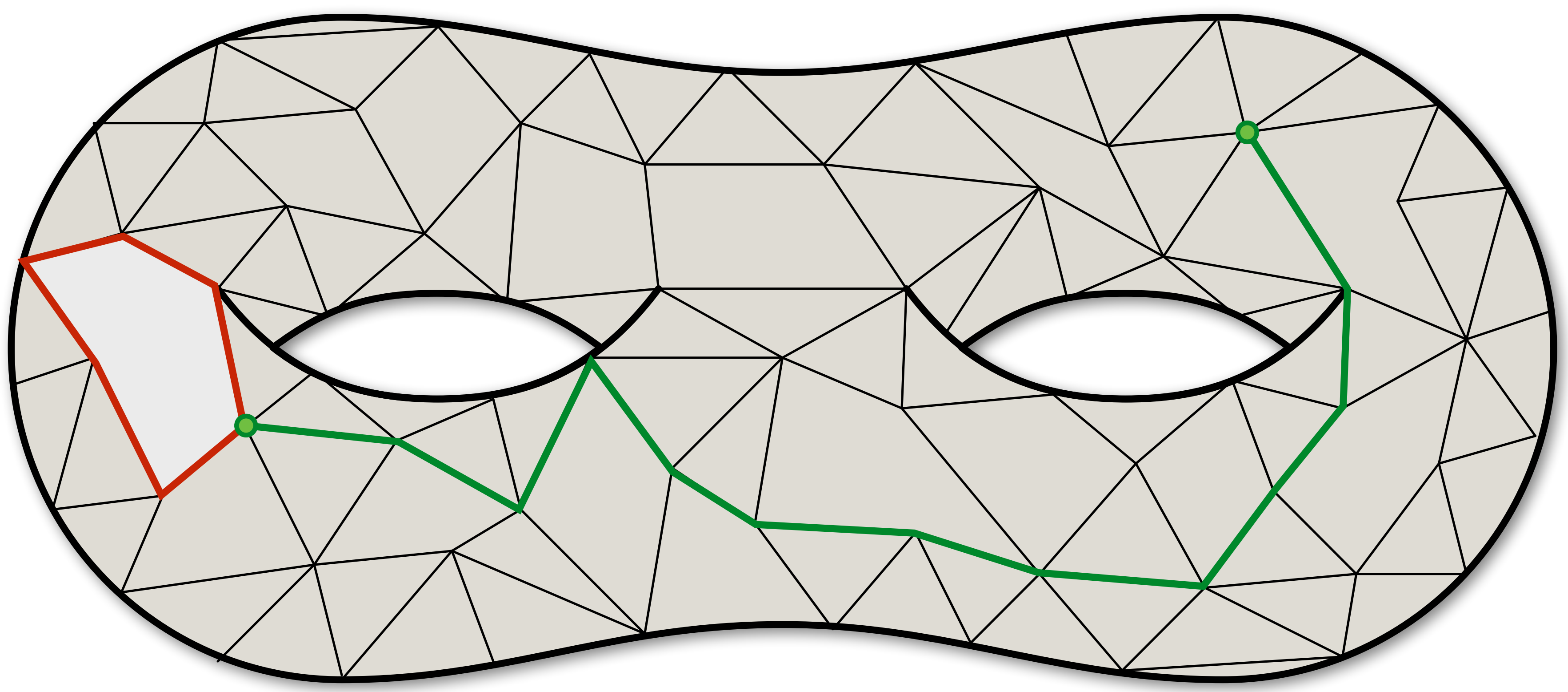
Back to surfaces

- Let Σ be any surface map with genus *g*. Fix a face *f* of Σ.
- We want to compute the shortest path trees rooted at every vertex of some "outer" face *f*.



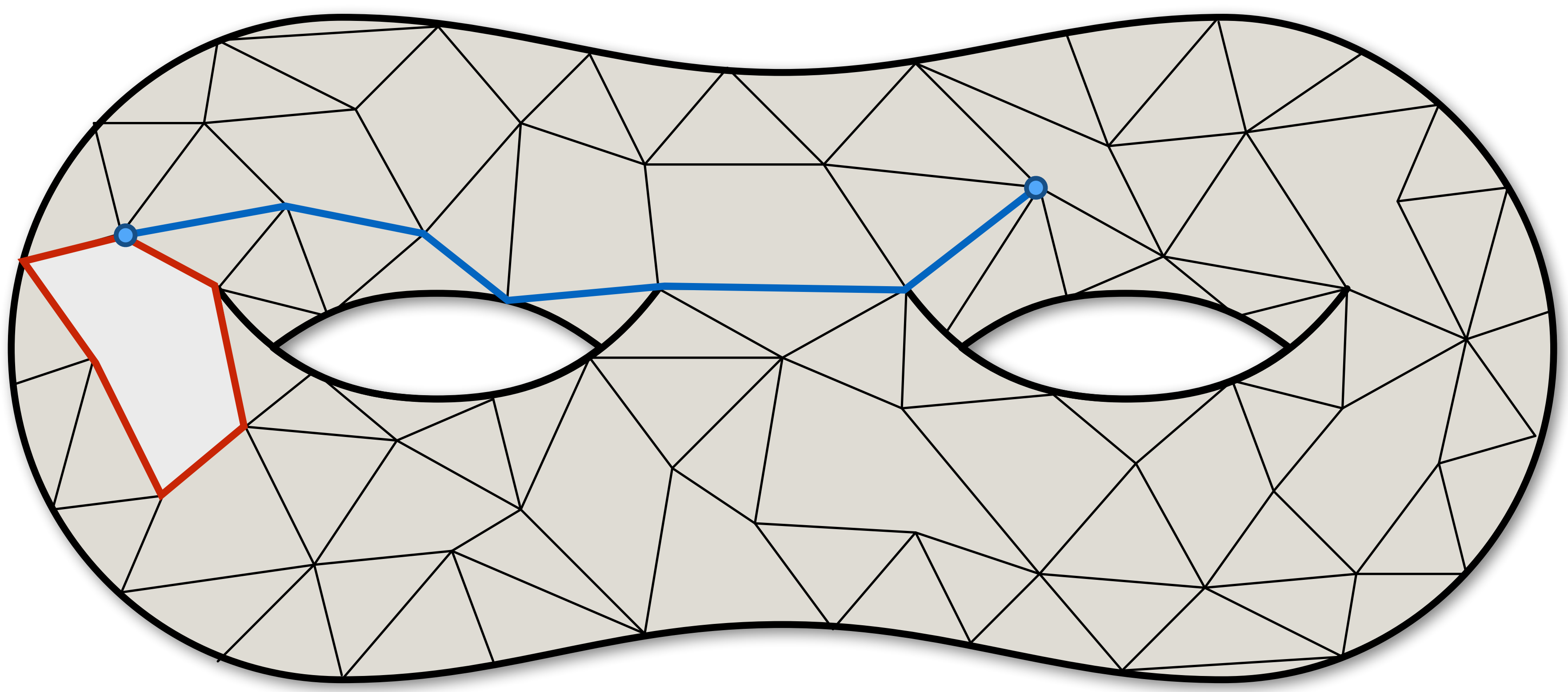
Back to surfaces

- Let Σ be any surface map with genus g. Fix a face f of Σ.
- We want to compute the shortest path trees rooted at every vertex of some "outer" face f.



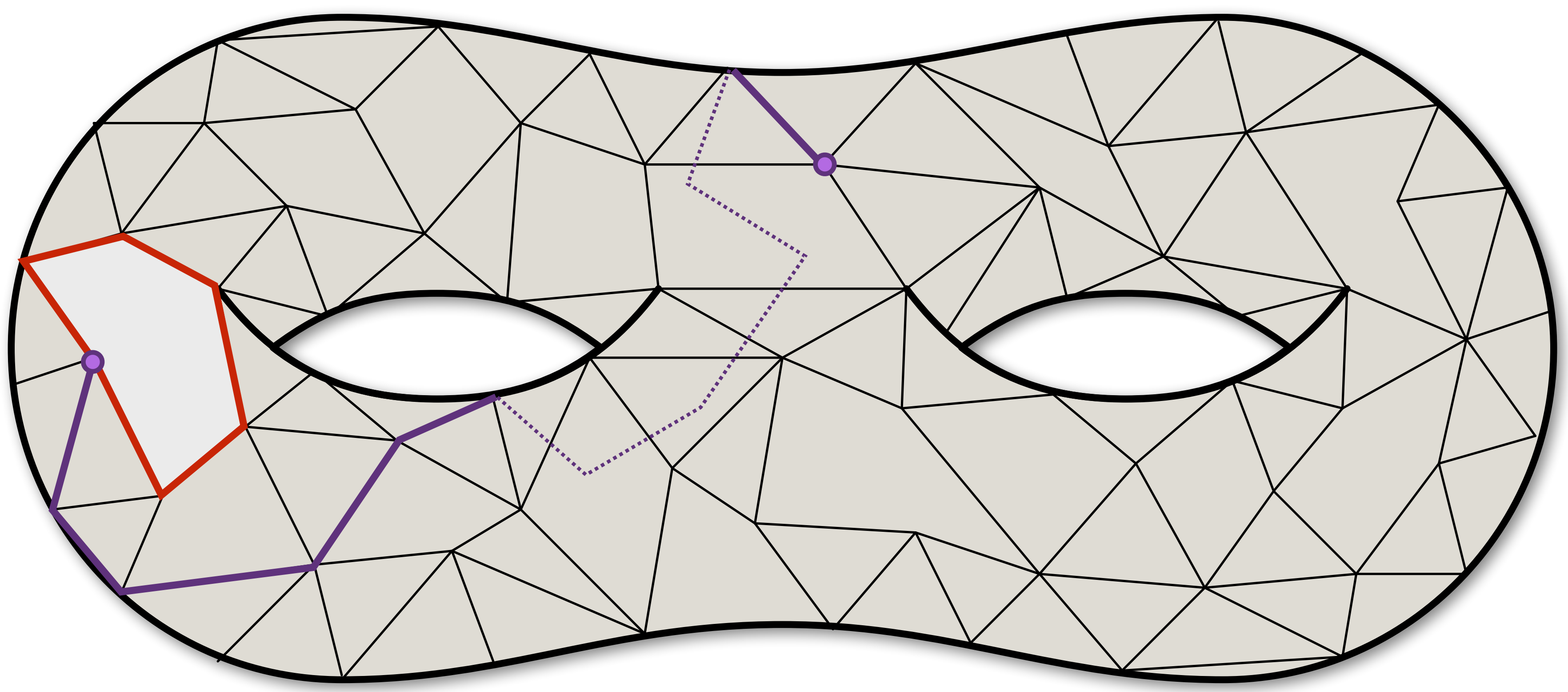
Back to surfaces

- Let Σ be any surface map with genus g. Fix a face f of Σ.
- We want to compute the shortest path trees rooted at every vertex of some "outer" face f.



Back to surfaces

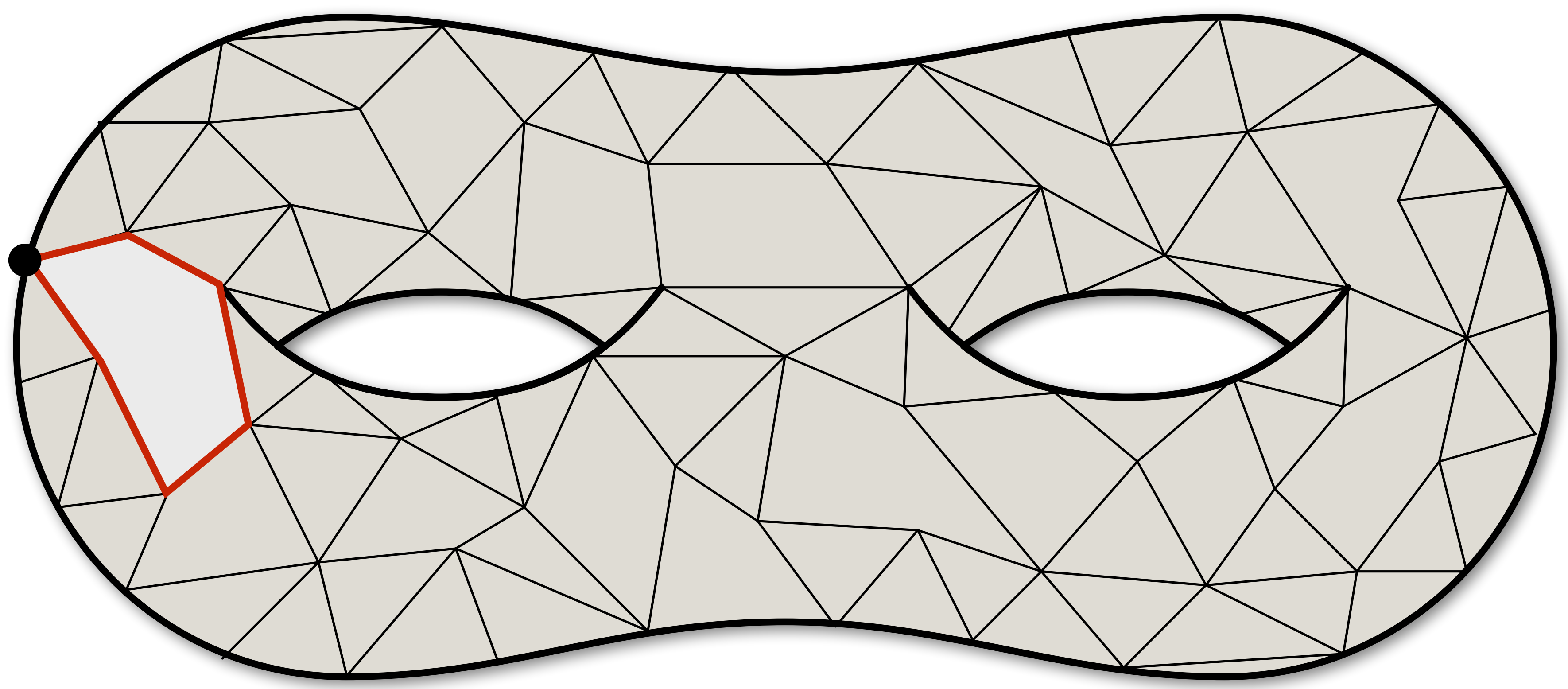
- Let Σ be any surface map with genus g. Fix a face f of Σ.
- We want to compute the shortest path trees rooted at every vertex of some “outer” face f.



Same strategy!

[Cabello Chambers Erickson 2013]

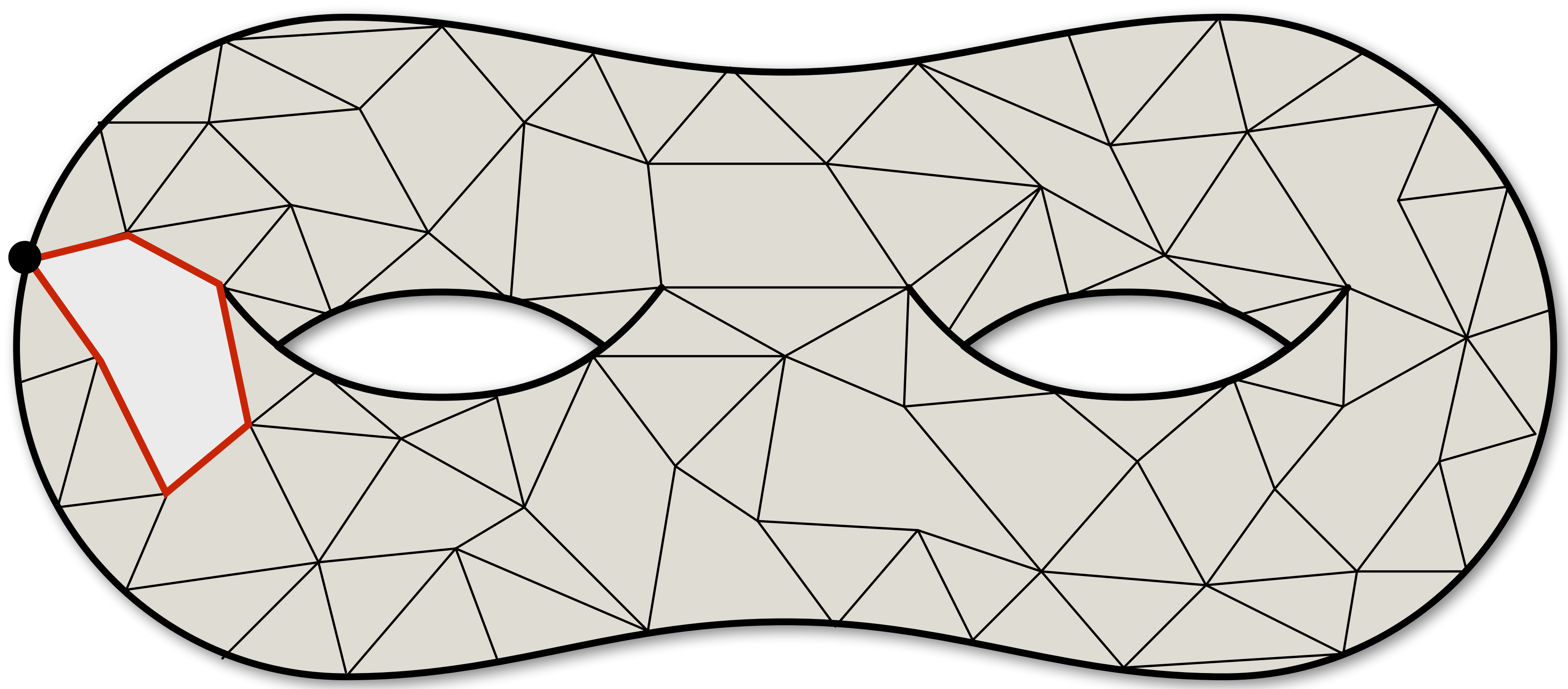
- Move a point *s continously* around *f*, maintaining both the shortest-path tree rooted at s and the complementary slacks. Whenever a non-tree edge becomes tense, relax it.



Same strategy!

[Cabello Chambers Erickson 2013]

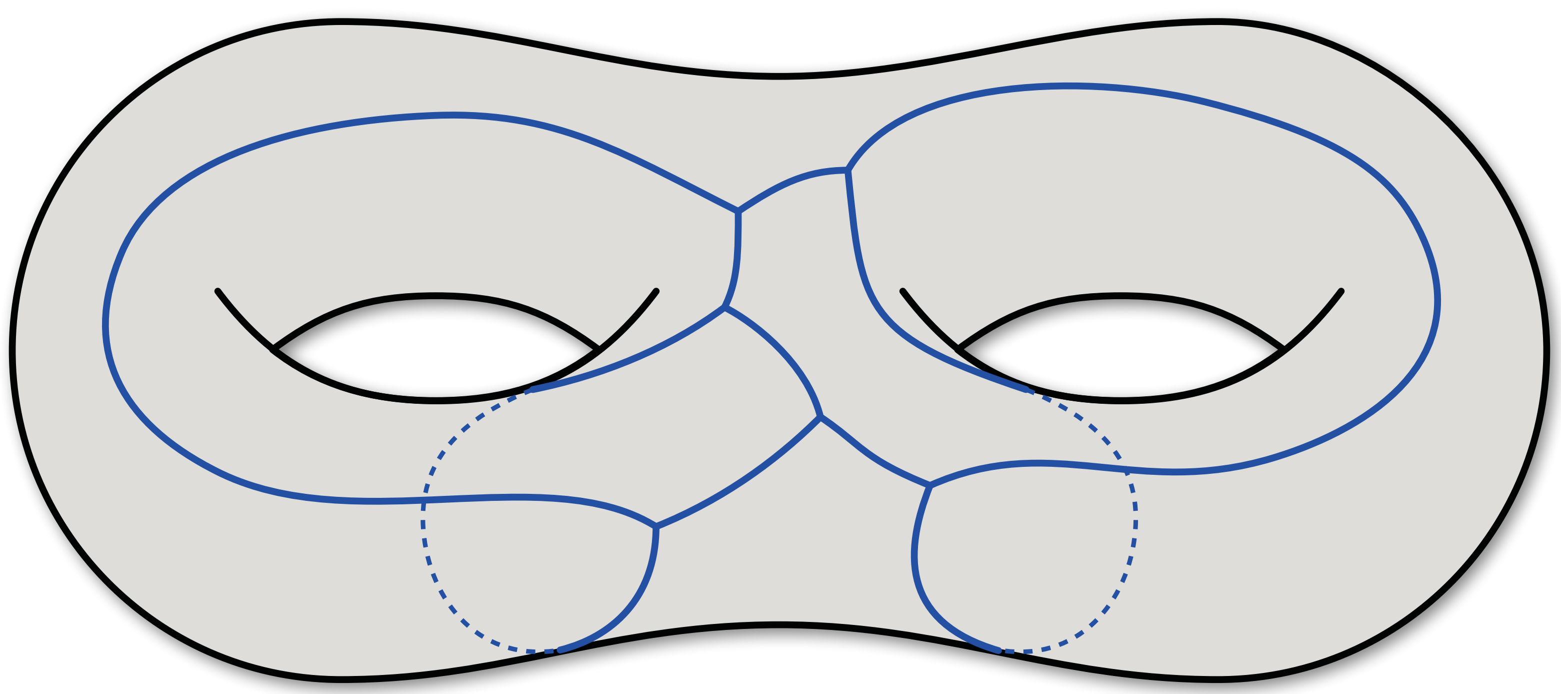
- Move a point *s continously* around *f*, maintaining both the shortest-path tree rooted at s and the complementary slacks. Whenever a non-tree edge becomes tense, relax it.



Complementary grove

[Cabello Chambers Erickson 2013]

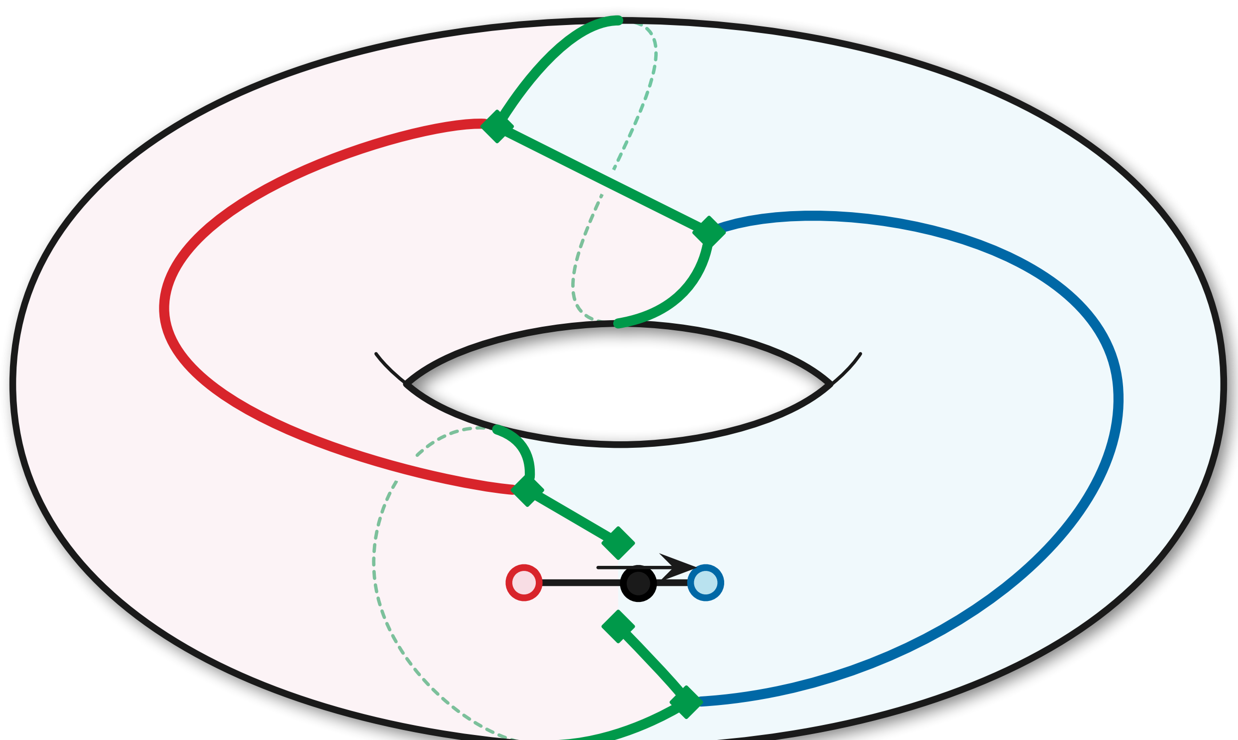
- The dual *cut graph* $X^* = (G \backslash T)^*$ is no longer a spanning tree!
- *Grove decomposition*: partition X^* into $6g$ subtrees of G^*.
  - Each subtree contains one dual cut path and all attached “hair”
  - Maintain each subtree in its own dynamic forest data structure



Where are the pivots?

[Cabello Chambers Erickson 2013]

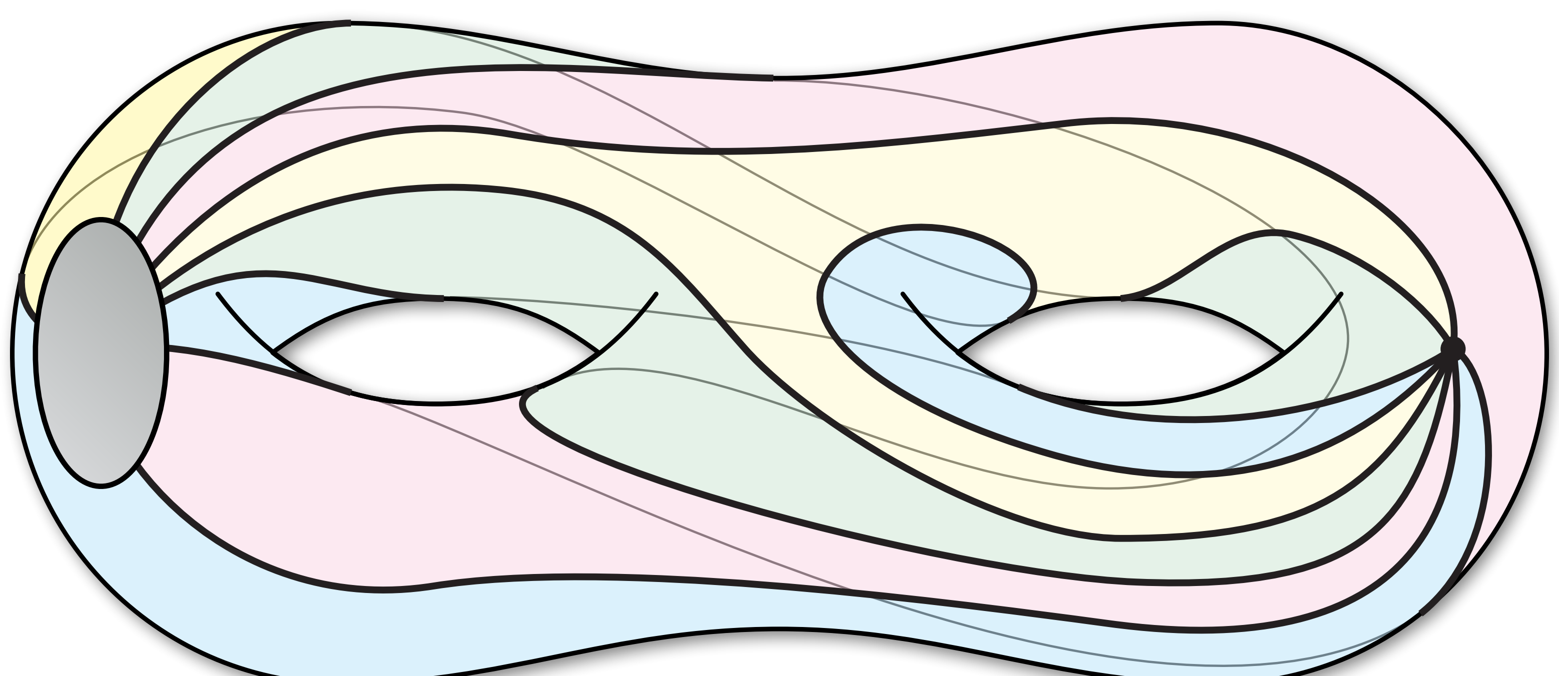
- All *active* edges are dual to edges in some dual cut path.
- We can find and execute each pivot using O(g) dynamic forest operations = O(g log n) amortized time.



How many pivots?

[Cabello Chambers Erickson 2013]

- Each directed edge pivots into T at most $4g$ times.
  - Generalization of disk-tree lemma
  - $4g$ = max # disjoint non-homotopic paths between two points in Σ
- So the total number of pivots is $O(gn)$



Summary

[Cabello Chambers Erickson 2013]
[Fox Erickson Lkhamsuren 2018]

- Given any surface map Σ with complexity n and genus g, with non-negatively weighted edges, and a face f...
- We can (implicitly) compute shortest-path distances from every vertex of f to every vertex of Σ in $O(gn \log n)$ time
  - with high probability
  - or in $O(gn \log^2 n)$ worst-case
  - or in $O(g^2 n \log n)$ worst-case
- So we can compute shortest nontrivial cycles in $O(g^2 n \log n)$ time

Thank you!

[Free Gruchy ("Slow-Mo Guys") 2018]

Thank you!

[Free Gruchy ("Slow-Mo Guys") 2018]