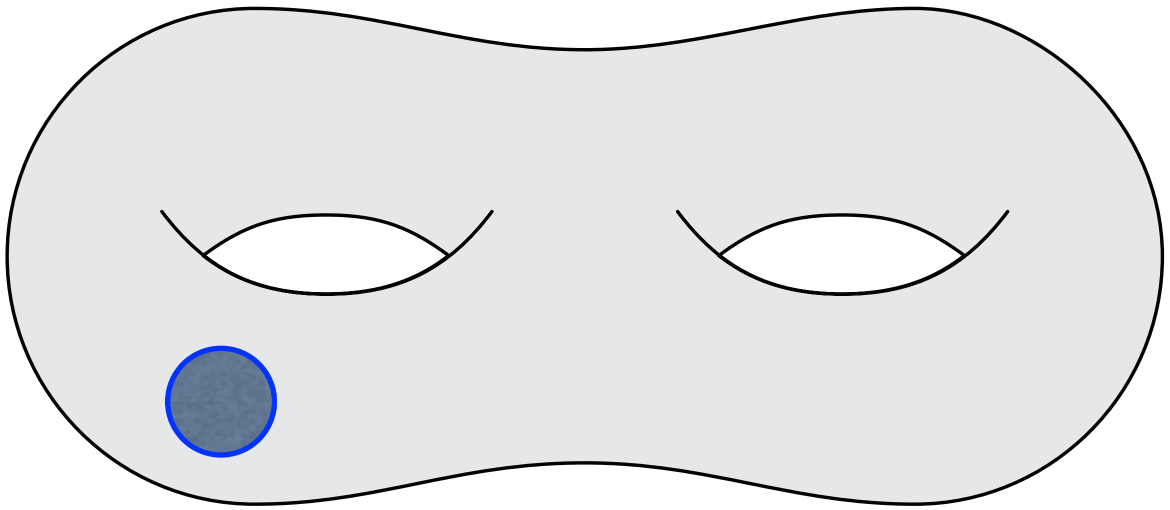
Computing Shortest Non-trivial Cycles in Directed Surface Graphs

Kyle Fox

University of Illinois at Urbana-Champaign

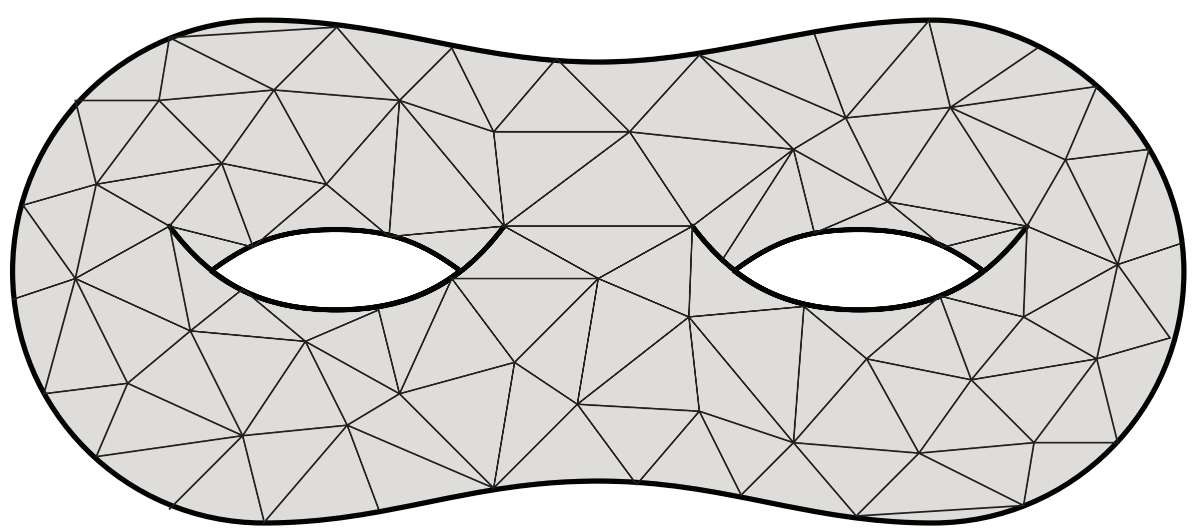
Surfaces

- 2-manifolds (with boundary)
- ***genus g***: max # of disjoint simple cycles whose compliment is connected
  = number of holes
  = number of handles attached to sphere



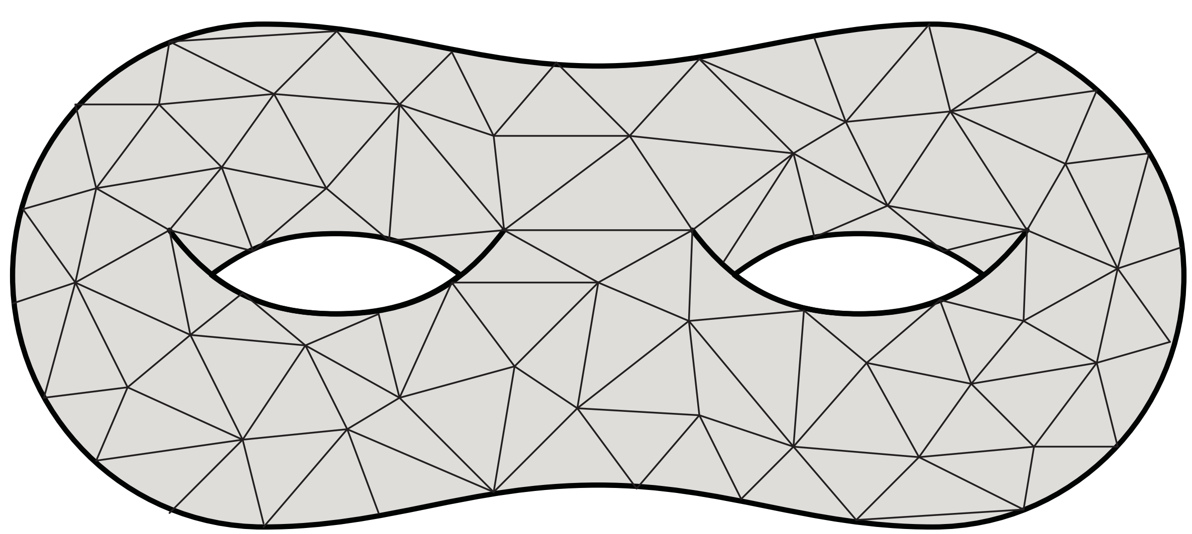
Surface Graphs

- *n* vertices as points
- *m* edges as (mostly) disjoint curves



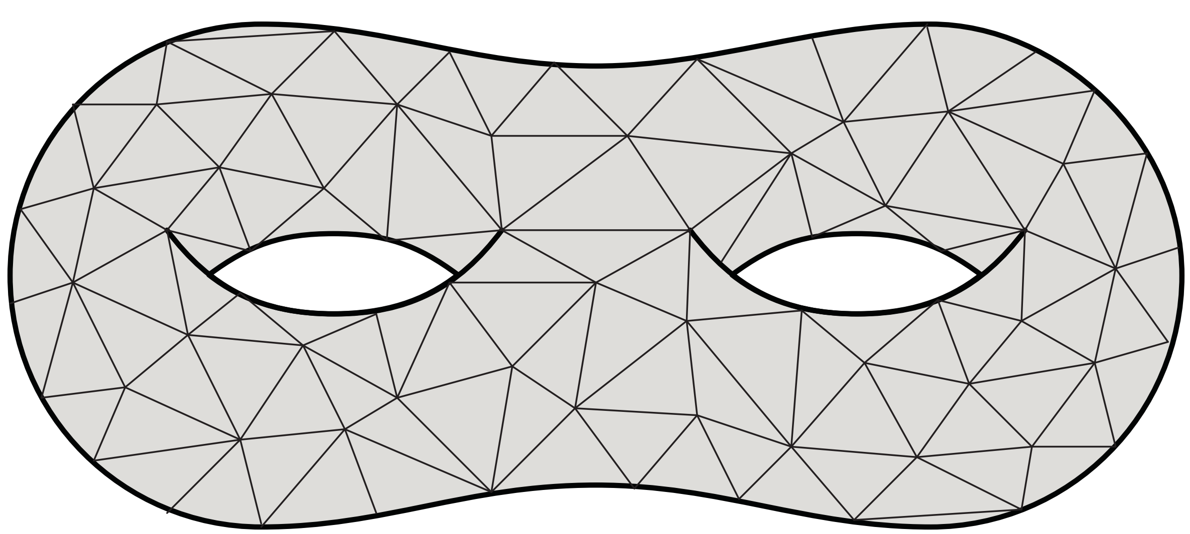
Surface Graphs

- *n* vertices as points
- *m* edges as (mostly) disjoint curves
- Assume $g = O(n)$ and $m = O(n)$

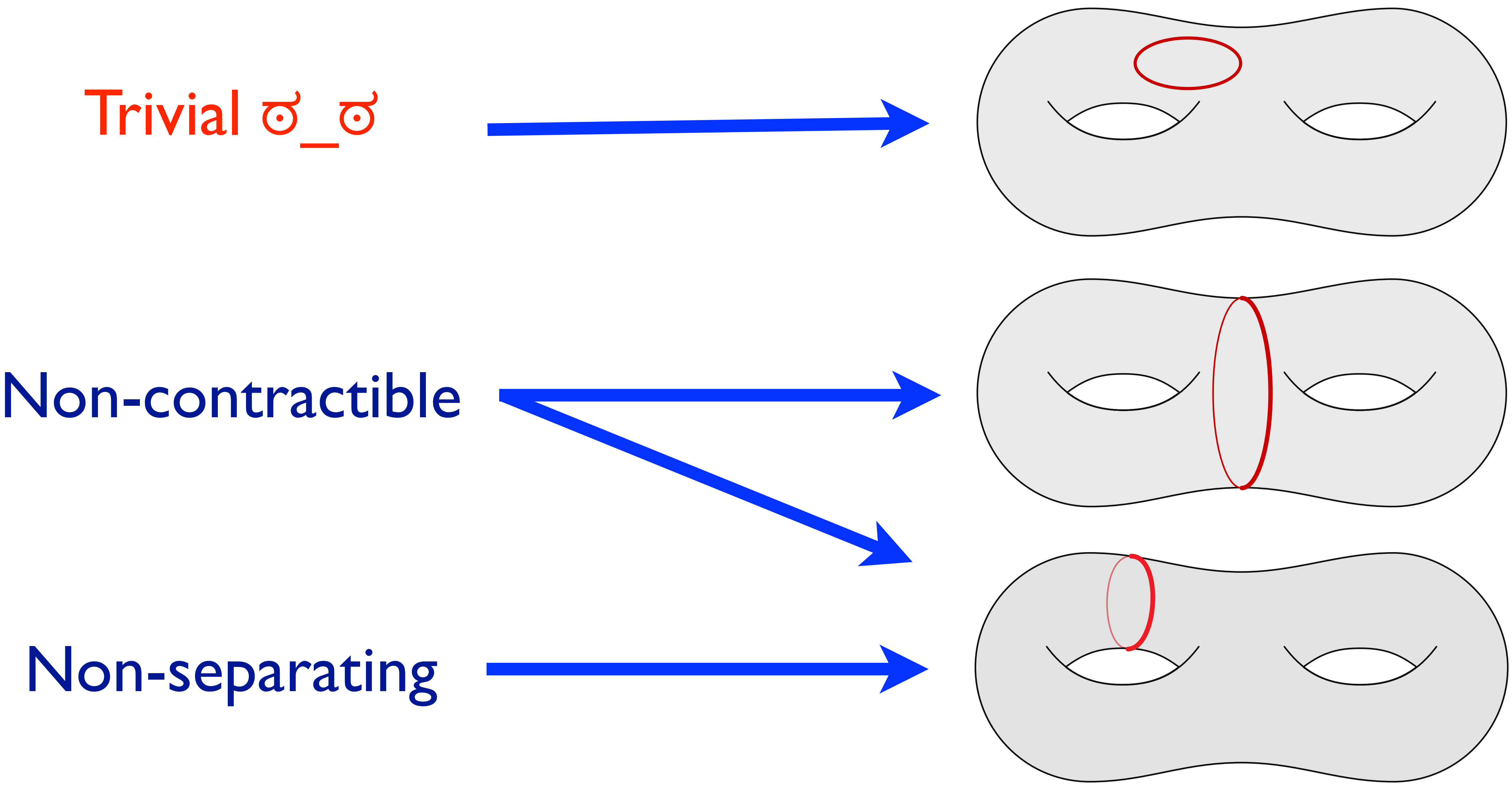


Surface Graphs

- *n* vertices as points
- *m* edges as (mostly) disjoint curves
- Assume $g = O(n)$ and $m = O(n)$
- We want to find non-trivial cycles



Non-trivial Cycles



Finding Short Non-trivial Cycles

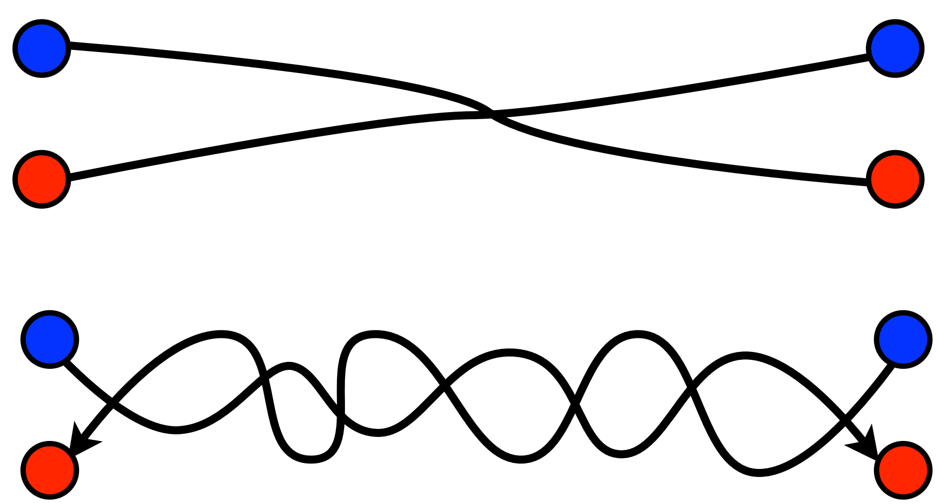
- Want to minimize sum of real edge lengths
- Natural question for surface embedded graphs
- Cutting along non-trivial cycles reduces the complexity of the graph
- Useful for combinatorial optimization, graphics, graph drawing, …

Results (Undirected)

Non-con.	Non-sep.	
$O(n^3)$	$O(n^3)$	[Thomassen '90]
$O(n^2 \log n)$	$O(n^2 \log n)$	[Erickson, Har-peled '04]
$g^{O(g)}\ n^{3/2}$	$O(g^{3/2}\ n^{3/2} \log n + g^{5/2}\ n^{1/2})$	[Cabello, Mohar '07]
$g^{O(g)}\ n \log n$	$g^{O(g)}\ n \log n$	[Kutz '06]
$O(g^2\ n \log n)$	$O(g^2\ n \log n)$	[Cabello, Chambers '06; C, C, Erickson '12]
$g^{O(g)}\ n \log \log n$	$g^{O(g)}\ n \log \log n$	[Italiano, *et al.* '11]

Undirected Edges are Kind

- Walks have the same length as their reversals
- Shortest paths cross at most once
- Neither holds in general for directed graphs



Results (Directed)

Non-con.	Non-sep.	
$O(n^2 \log n)$ and $O(g^{1/2} n^{3/2} \log n)$	$O(n^2 \log n)$ and $O(g^{1/2} n^{3/2} \log n)$	[Cabello, Colin de Verdière, Lazarus '10]
	$2^{O(g)} n \log n$	[Erickson, Nayyeri '11]
$g^{O(g)} n \log n$	$O(g^2 n \log n)$	[Erickson '11]
$O(g^3 n \log n)$		[F '11]

Results (Directed)

Non-con.	Non-sep.	
$O(n^2 \log n)$ and $O(g^{1/2} n^{3/2} \log n)$	$O(n^2 \log n)$ and $O(g^{1/2} n^{3/2} \log n)$	[Cabello, Colin de Verdière, Lazarus '10]
	$2^{O(g)} n \log n$	[Erickson, Nayyeri '11]
$g^{O(g)} n \log n$	$O(g^2 n \log n)$	[Erickson '11]
$O(g^3 n \log n)$		[F '11]

Cabello, Colin de Verdière, and Lazarus

- $O(n^2 \log n)$ time for shortest non-contractible (non-separating) cycle
- Finds one closed walk per vertex based on the 3-path condition

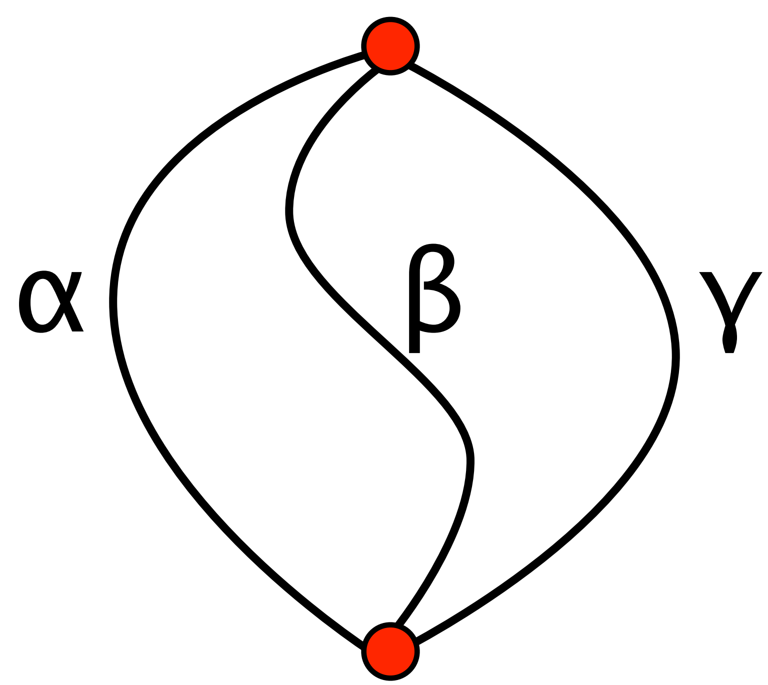
3-path Condition

- Contractible (separating) cycles have these properties:
  1. Their reversals are contractible (separating)

3-path Condition

- Contractible (separating) cycles have these properties:

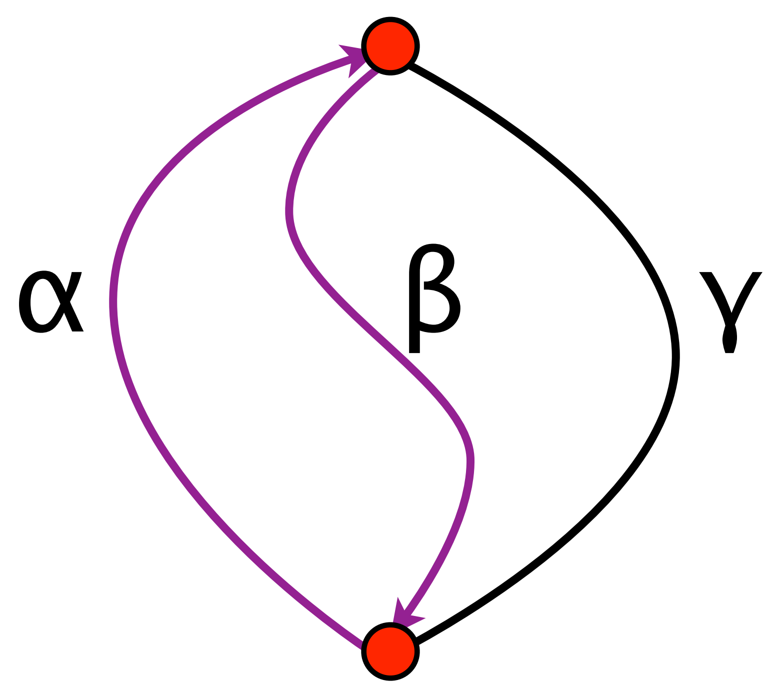
  2. If $\alpha \cdot rev(\beta)$, and $\beta \cdot rev(\gamma)$ are contractible (separating), then $\alpha \cdot rev(\gamma)$ is contractible (separating)



3-path Condition

- Contractible (separating) cycles have these properties:

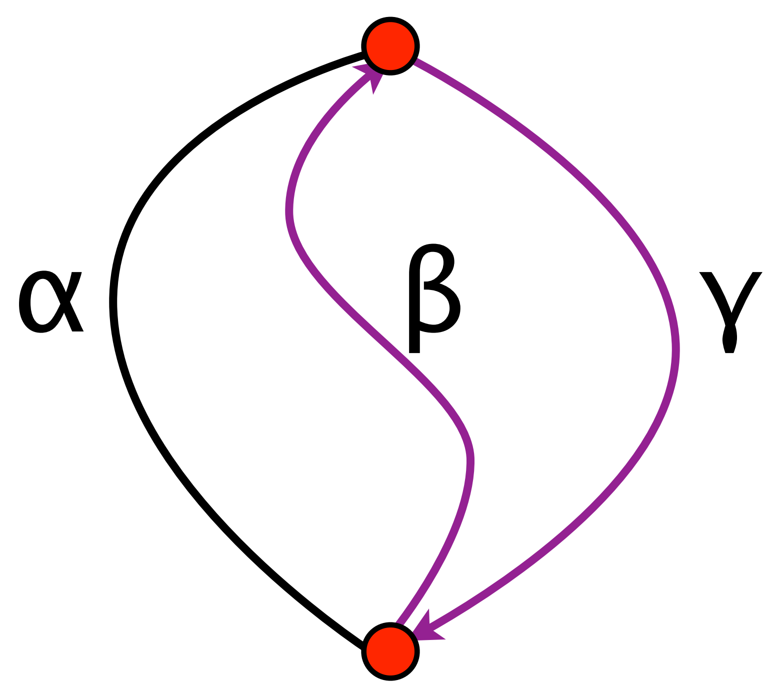
  2. If $\alpha \cdot \mathrm{rev}(\beta)$, and $\beta \cdot \mathrm{rev}(\gamma)$ are contractible (separating), then $\alpha \cdot \mathrm{rev}(\gamma)$ is contractible (separating)



3-path Condition

- Contractible (separating) cycles have these properties:

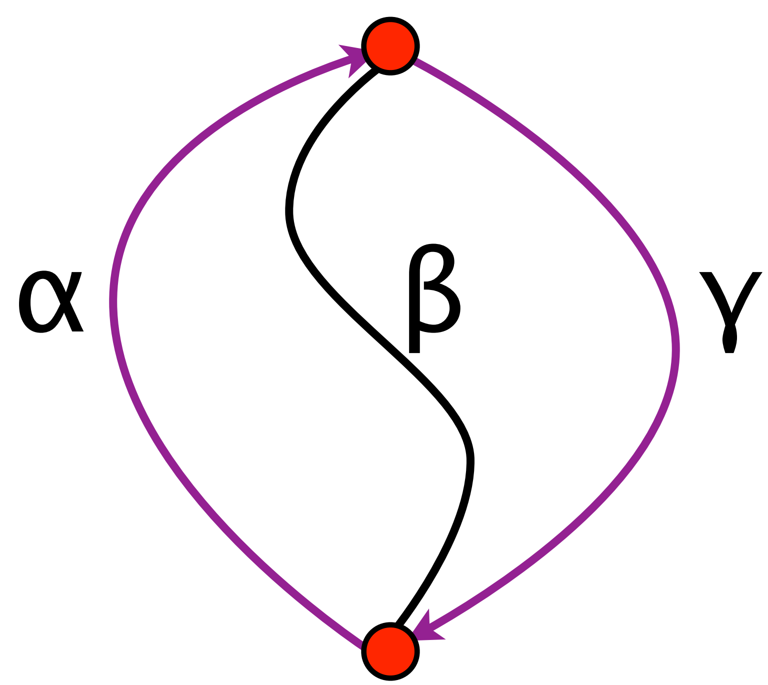
2. If $\alpha \cdot \text{rev}(\beta)$, and $\beta \cdot \text{rev}(\gamma)$ are contractible (separating), then $\alpha \cdot \text{rev}(\gamma)$ is contractible (separating)



3-path Condition

- Contractible (separating) cycles have these properties:

  2. If $\alpha \cdot rev(\beta)$, and $\beta \cdot rev(\gamma)$ are contractible (separating), then $\alpha \cdot rev(\gamma)$ is contractible (separating)

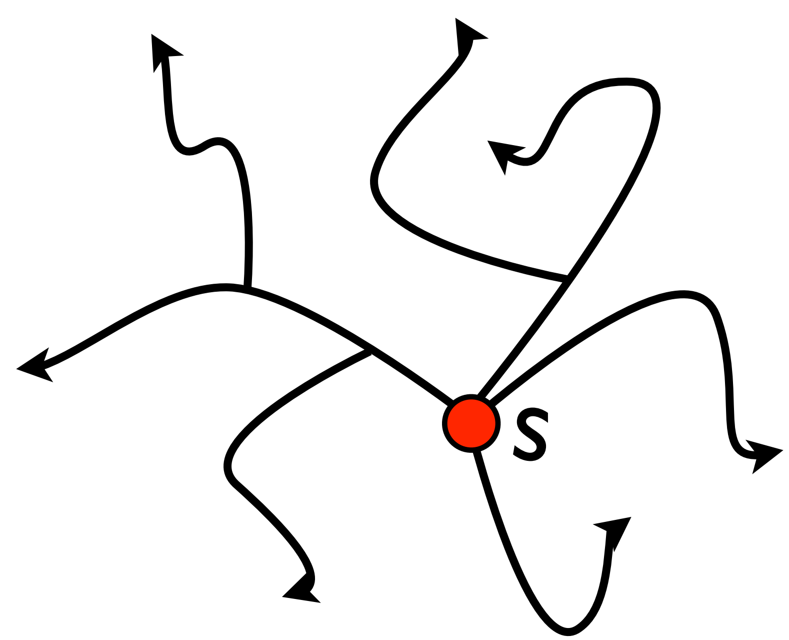


A Generic Algorithm

- Given a family of closed walks with the 3-path condition, we can find the shortest closed walk avoiding that family
- We find the shortest interesting walk based at each vertex
- The idea can be used to find many types of interesting cycles (including non-contractible and non-separating)

Shortest Paths

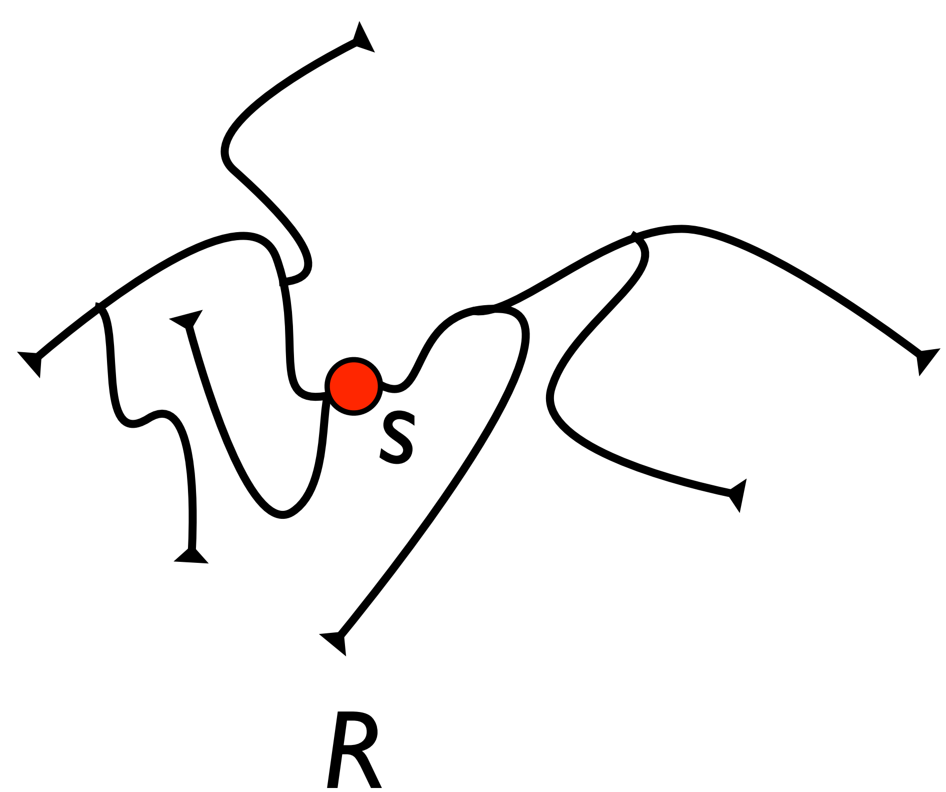
- Let *T* be the shortest path tree with source *s* and *R* be the reverse shortest path tree with target *s*



T

Shortest Paths

- Let *T* be the shortest path tree with source *s* and *R* be the reverse shortest path tree with target *s*

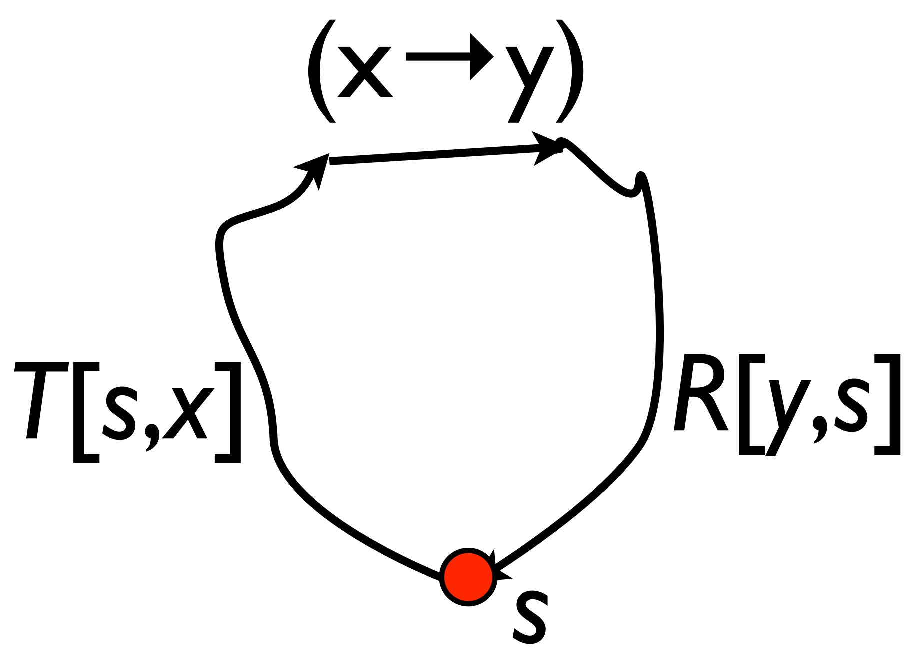


Shortest Paths

- We can compute both T and R in $O(n \log n)$ time using Dijkstra's algorithm

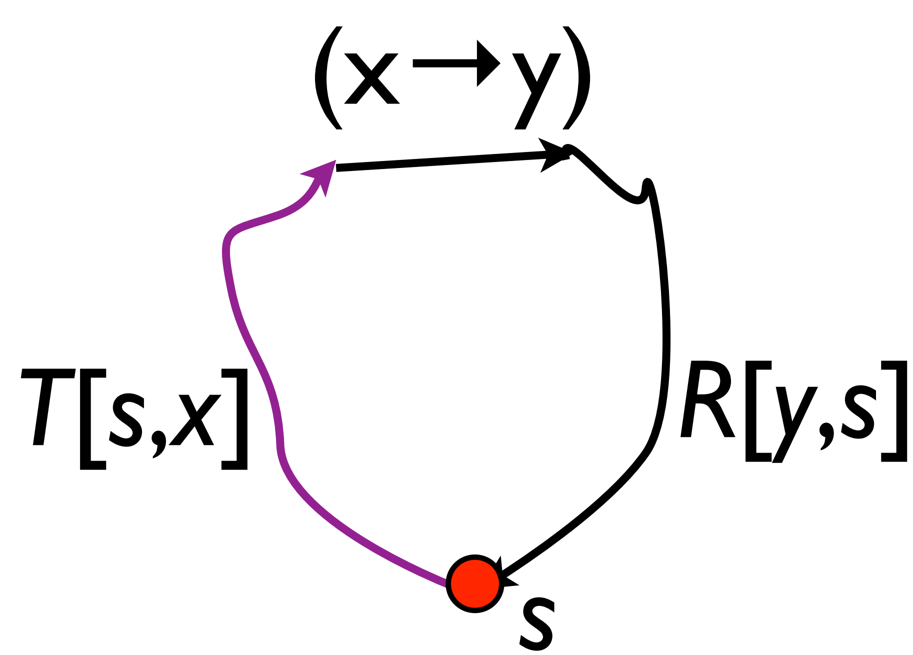
Candidate Walks

- The shortest interesting walk containing *s* follows *T*, takes an edge ($x \rightarrow y$) disjoint from *T*, and then follows *R*



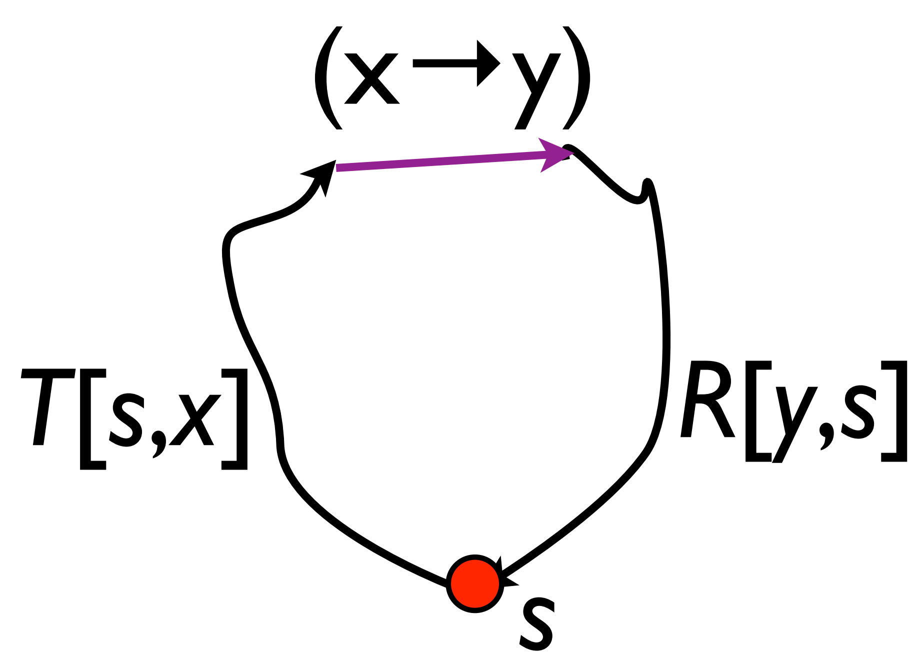
Candidate Walks

- The shortest interesting walk containing *s* follows *T*, takes an edge ($x \rightarrow y$) disjoint from *T*, and then follows *R*



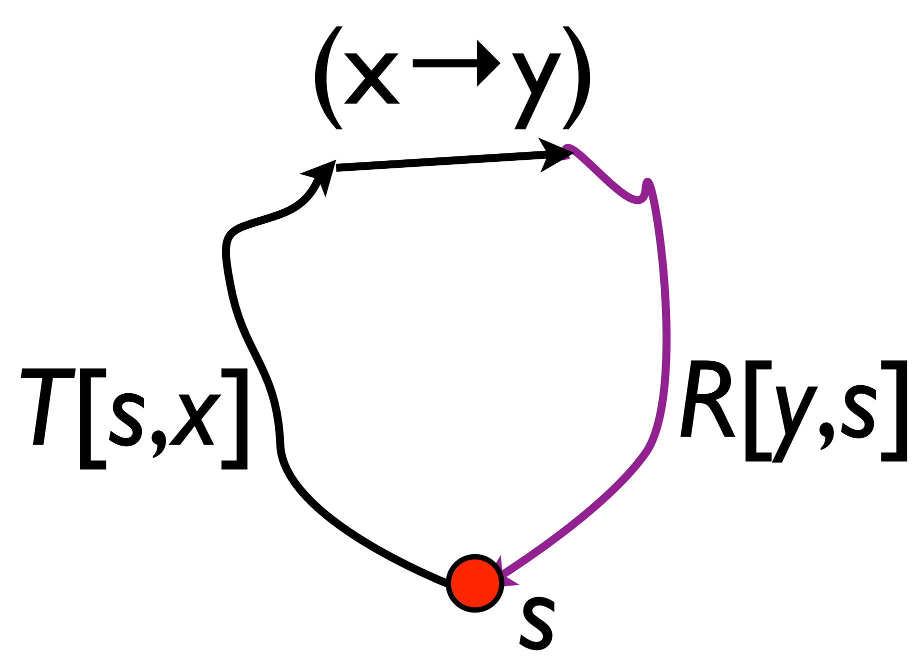
Candidate Walks

- The shortest interesting walk containing *s* follows *T*, takes an edge ($x \rightarrow y$) disjoint from *T*, and then follows *R*



Candidate Walks

- The shortest interesting walk containing *s* follows *T*, takes an edge ($x \rightarrow y$) disjoint from *T*, and then follows *R*

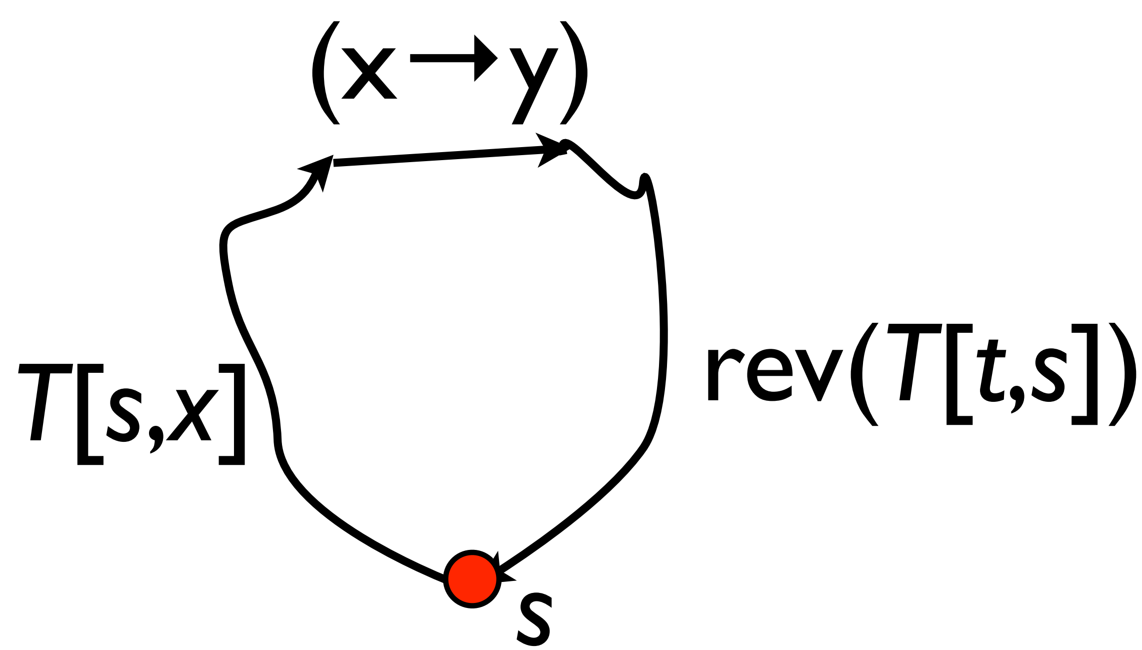


Check All Edges

- Check each edge $(x \to y)$ to see if T, R, and $(x \to y)$ make an interesting walk
- If we spend $\tau(n)$ time per edge, we can find the shortest interesting walk through s in $O(n\,\tau(n) + n \log n)$ time
- We can find the shortest interesting cycle in $O(n^2\,\tau(n) + n^2 \log n)$ time by finding walks through each vertex

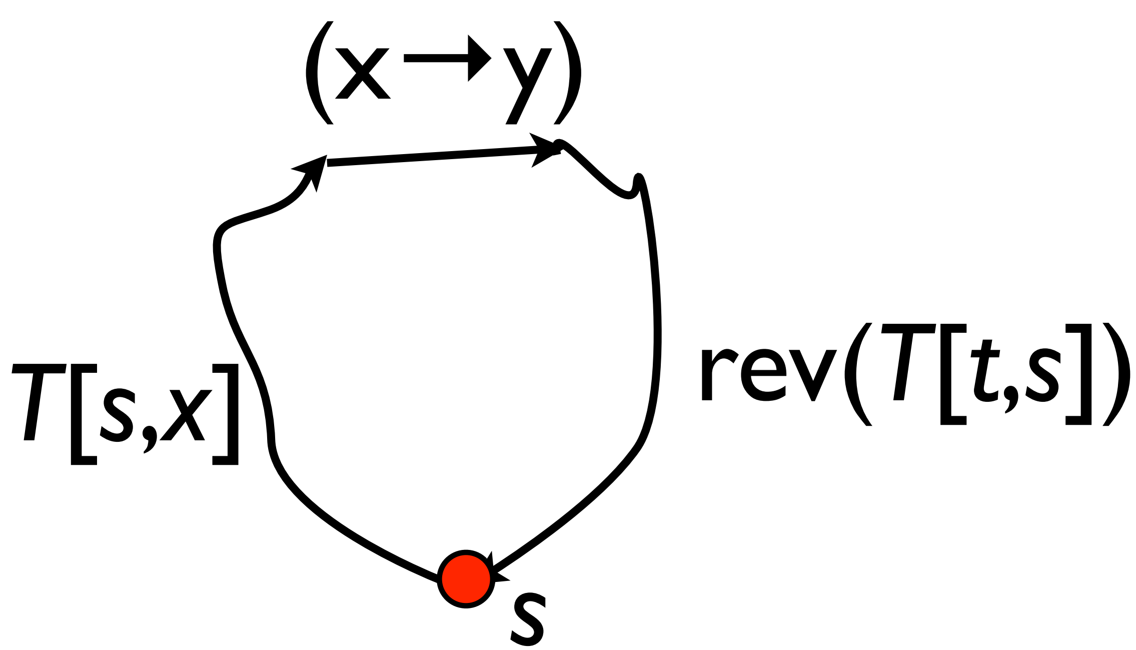
Fast Edge Checking

- The edge ($x \rightarrow y$) we want makes an interesting undirected cycle with T



Fast Edge Checking

- We can play tricks with the dual graph to check if each of these cycles is non-contractible or non-separating in constant time



Fast Edge Checking

- Fast edge checking lets us find the shortest non-contractible or non-separating cycle in $O(n^2 \log n)$ time

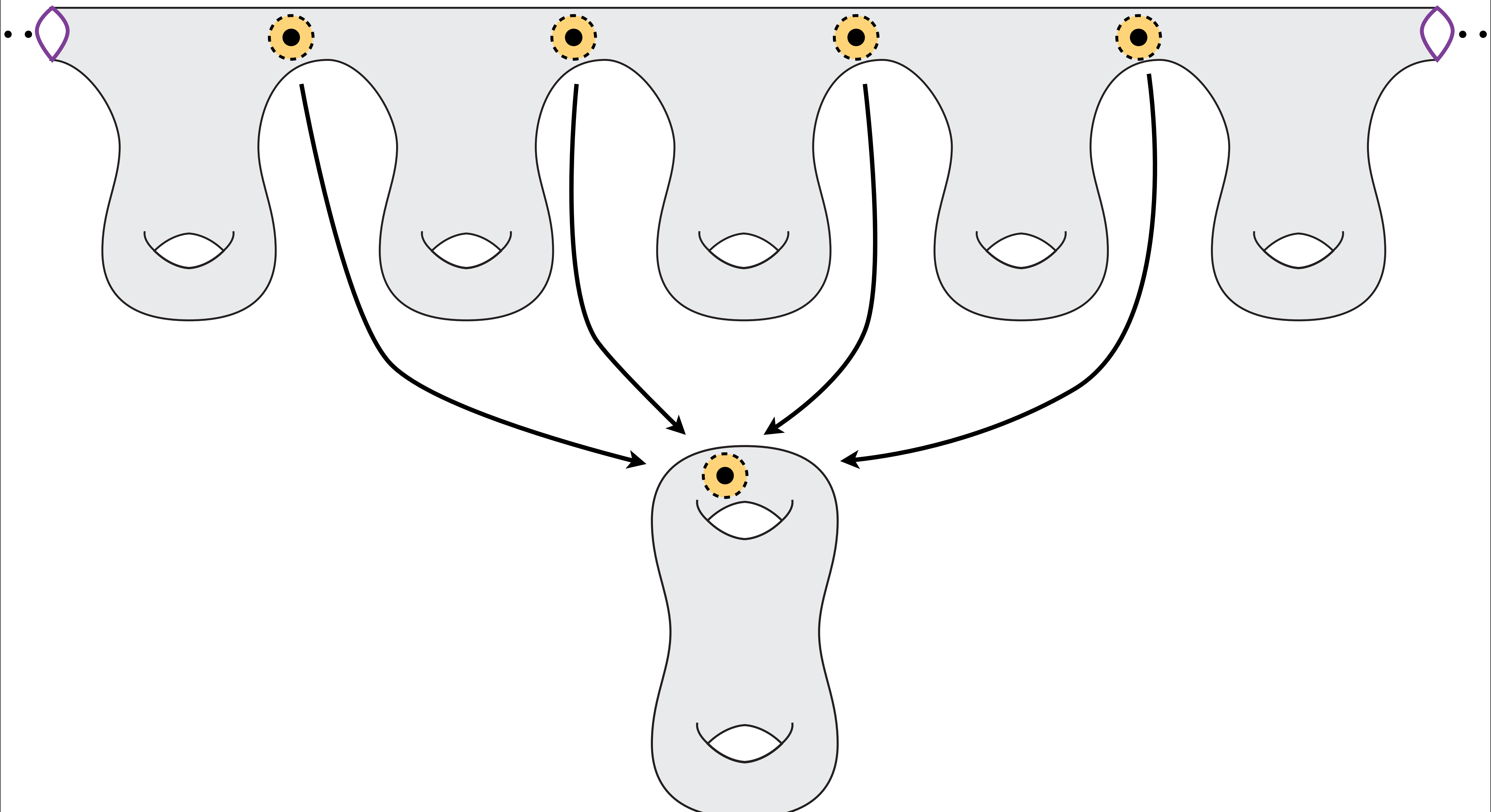
Erickson

- $O(g^2 \, n \log n)$ time for shortest non-separating cycle
- Lifts graph to several finite covering spaces

Covering Spaces

- Each point *x* in the original space lies in an open neighborhood *U* such that one or more open neighborhoods in the covering space have a homeomorphism to *U*

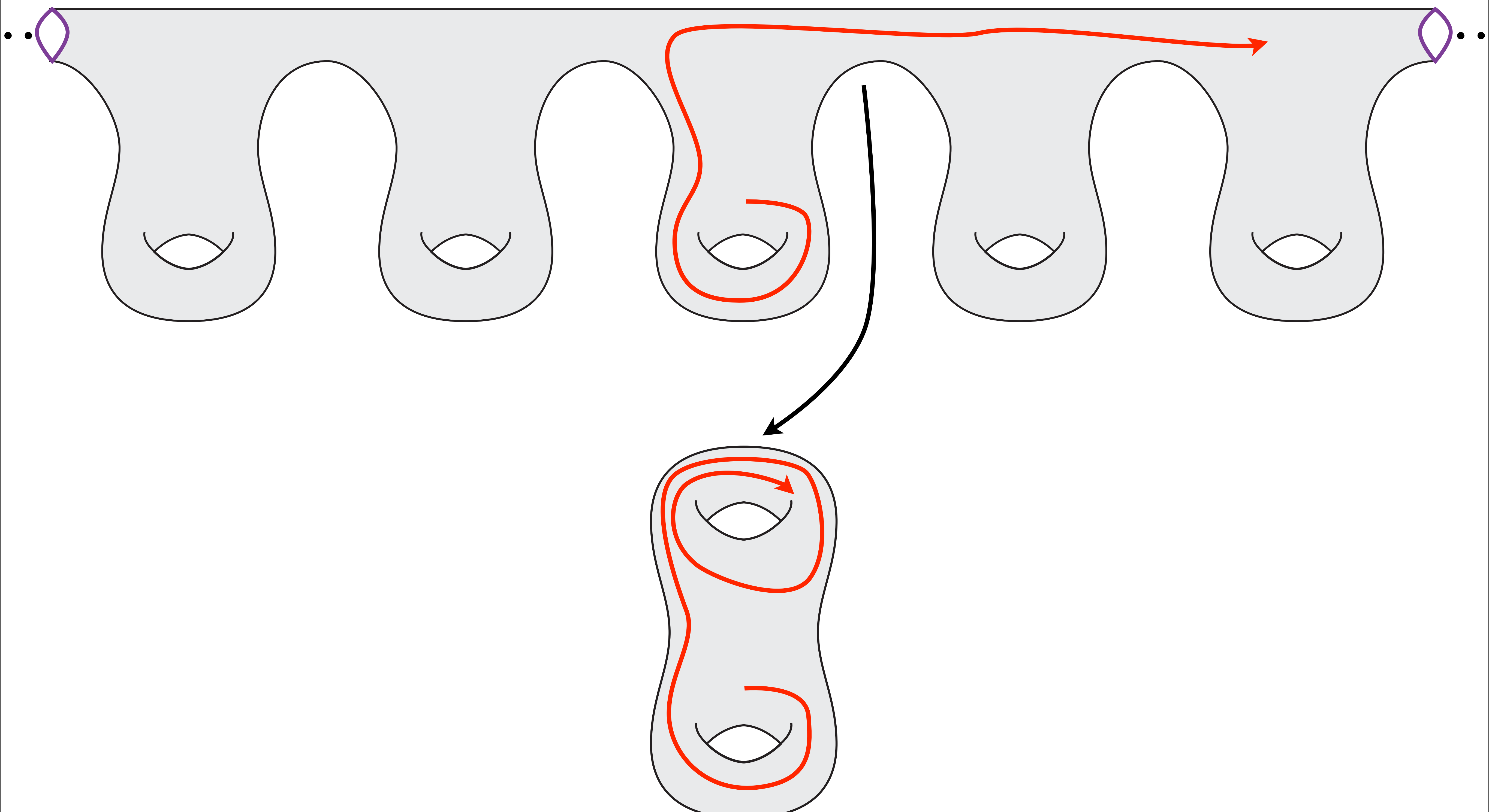
Covering Spaces



Lifts and Projections

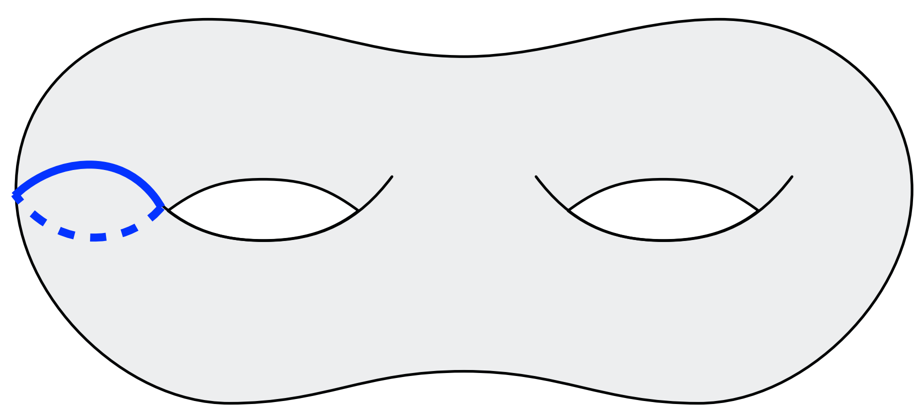
- Any walk on the original surface has at most one lift to the covering space that begins on a particular point
- Each walk in the covering space projects to a walk in the original space

Covering Spaces



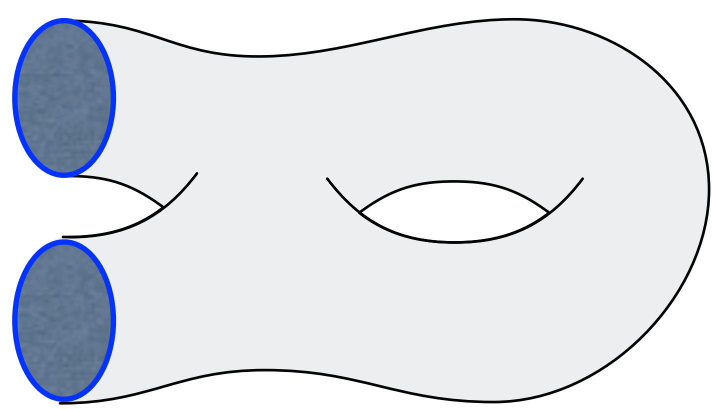
Cyclic Double Cover

- Let λ be any non-separating cycle



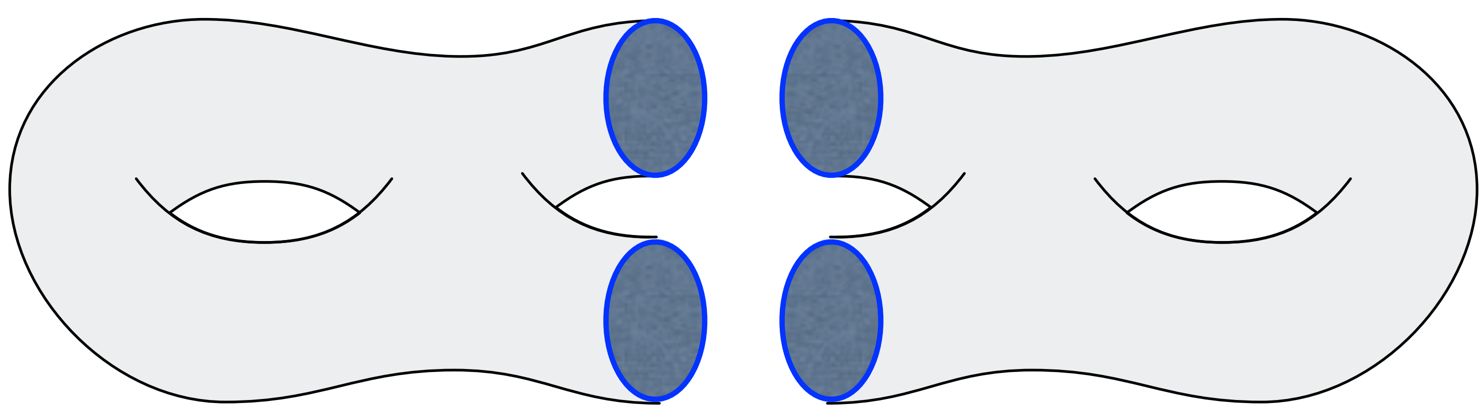
Cyclic Double Cover

- Cut the surface along λ



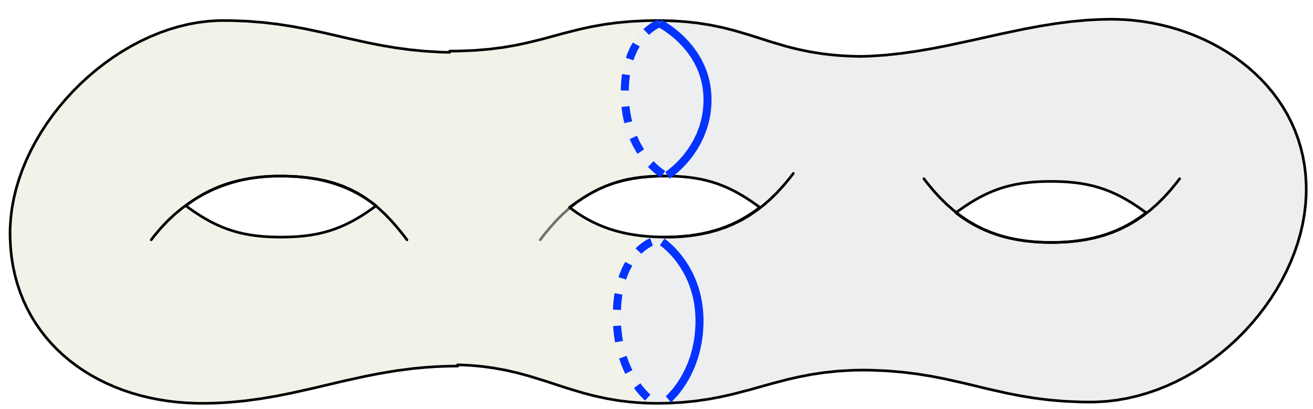
Cyclic Double Cover

- Make two copies of the cut surface



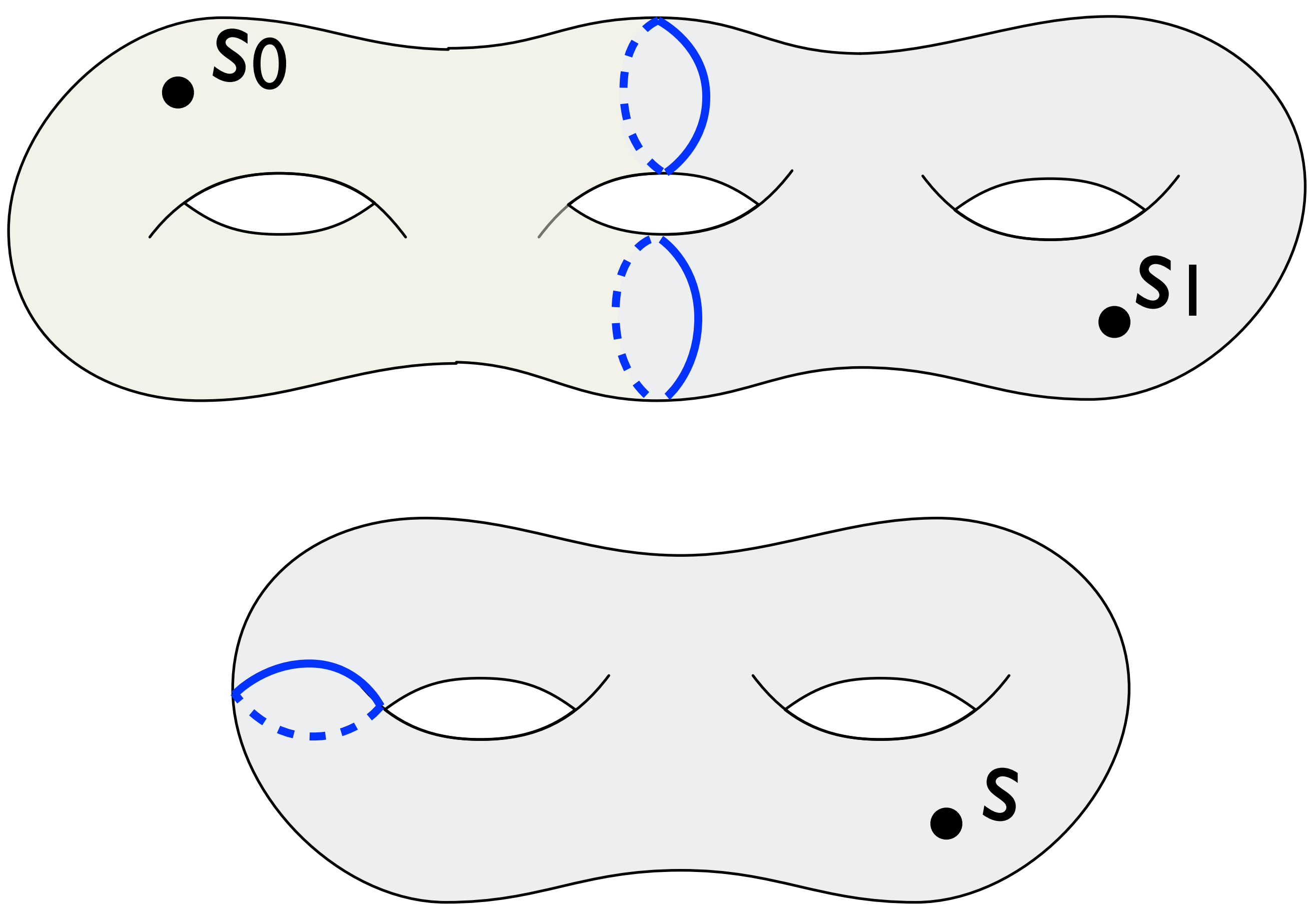
Cyclic Double Cover

- Glue cut spaces together by identifying their copies of λ



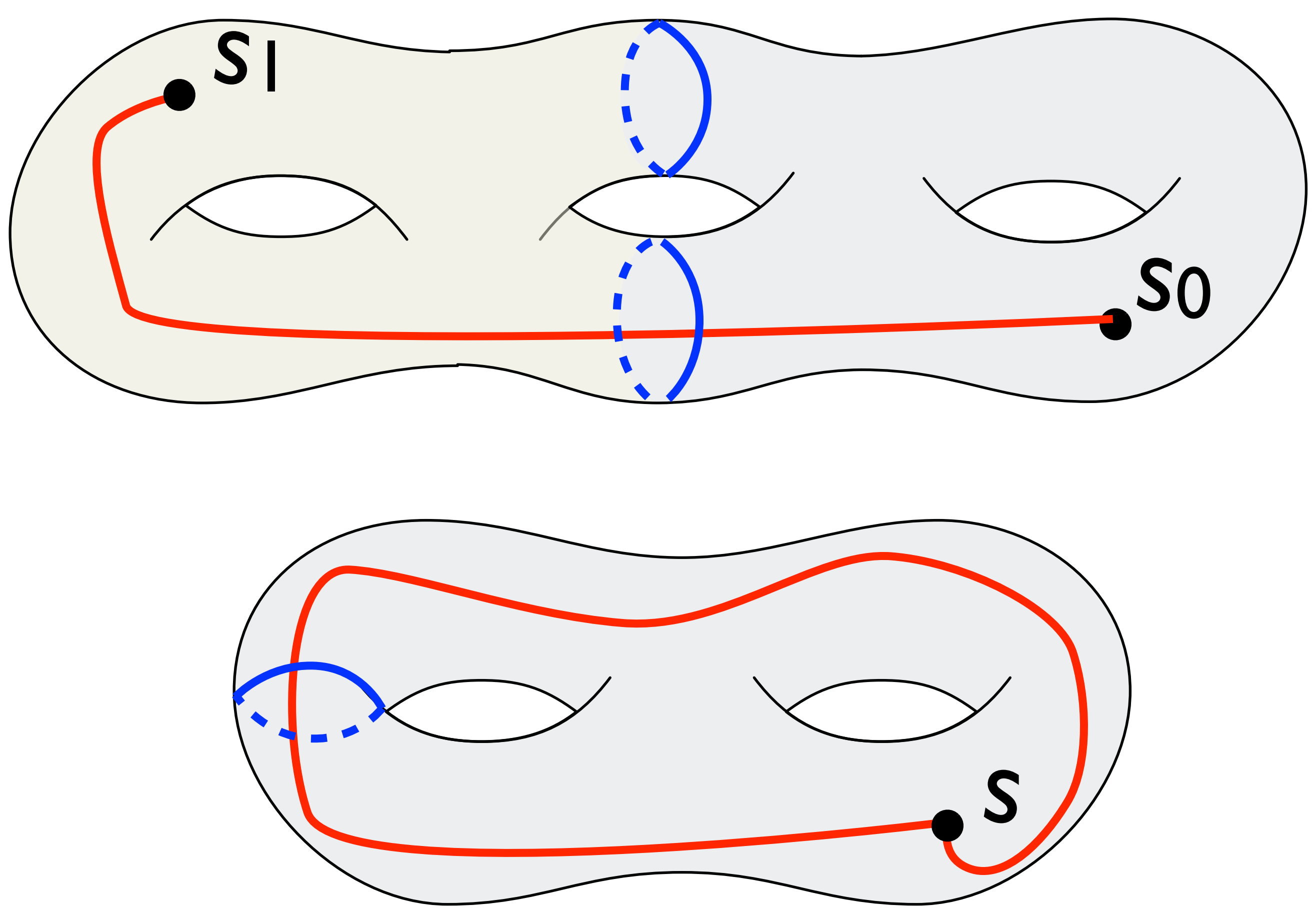
Crossing λ

- Let s be a vertex on the original surface with copies s_0 and s_1 in the cover



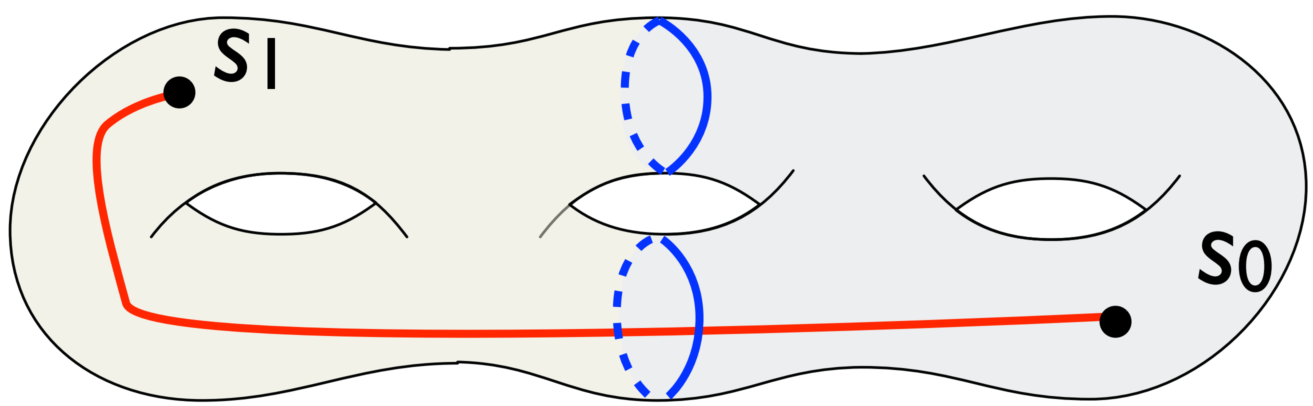
Crossing λ

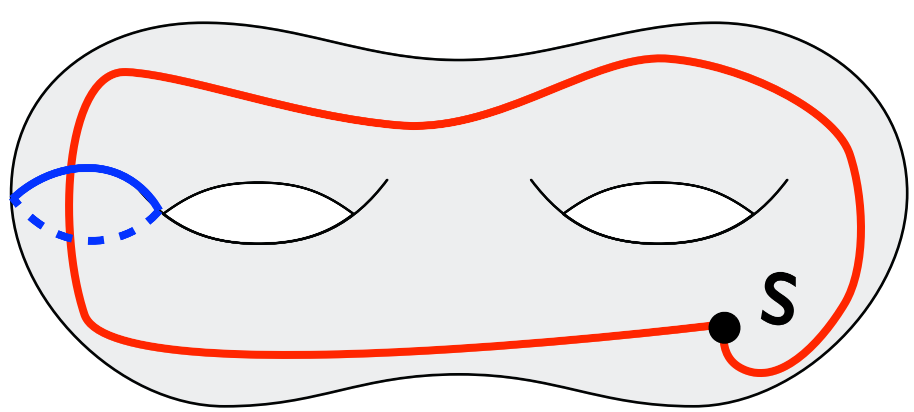
- A closed walk γ based at s lifts to a walk from s_0 to s_1 if and only if γ crosses λ an odd number of times



Crossing λ

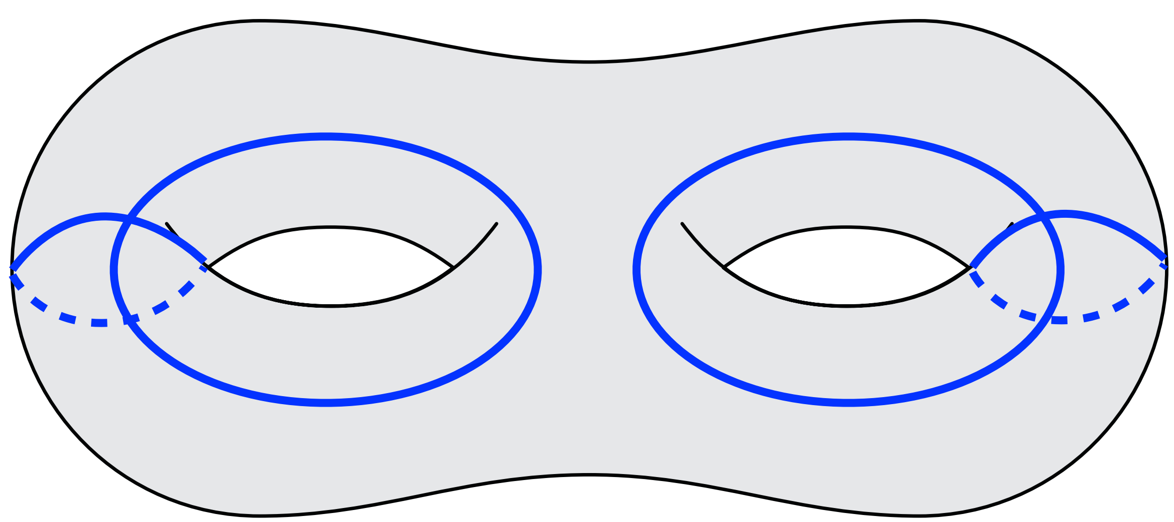
- The shortest closed walk based at s crossing λ an odd number of times is a shortest walk from s_0 to s_1





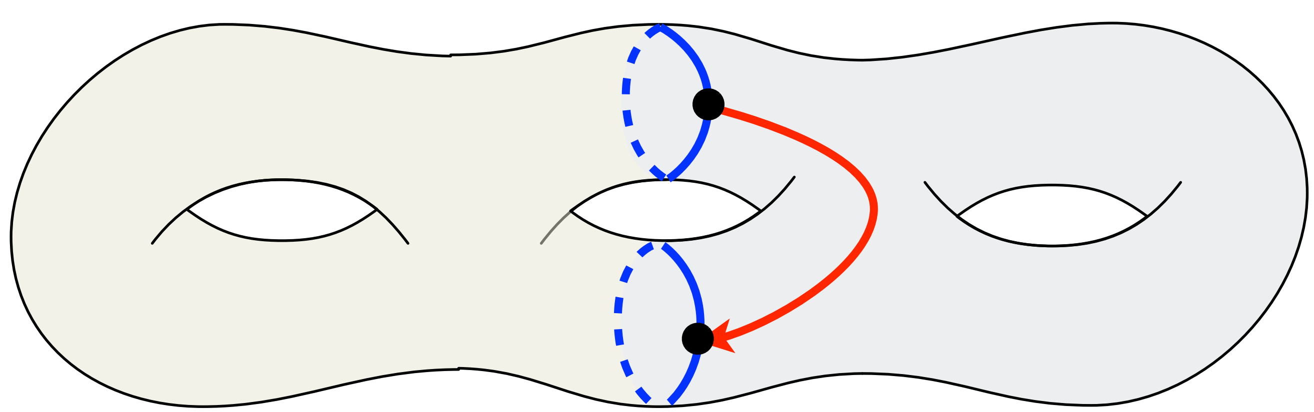
System of Cycles

- Compute a system of 2g non-separating cycles from shortest paths in $O(n \log n + g\,n)$ time
- Any non-separating cycle crosses at least one cycle in the system an odd number of times



Search the Double Cover

- For each cycle λ in the system, build the cyclic double cover
- For each vertex s on λ, compute a shortest path from s_0 to s_1 and return the shortest walk found

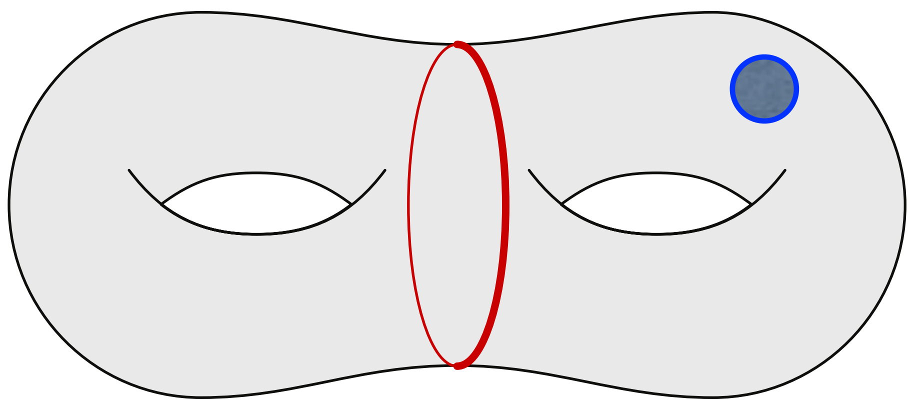


Running Time

- Can search for shortest paths in $O(g\, n \log n)$ time per double cover [Cabello, Chambers, Erickson '12]
- $O(g^2\, n \log n)$ time spent searching $2g$ covers

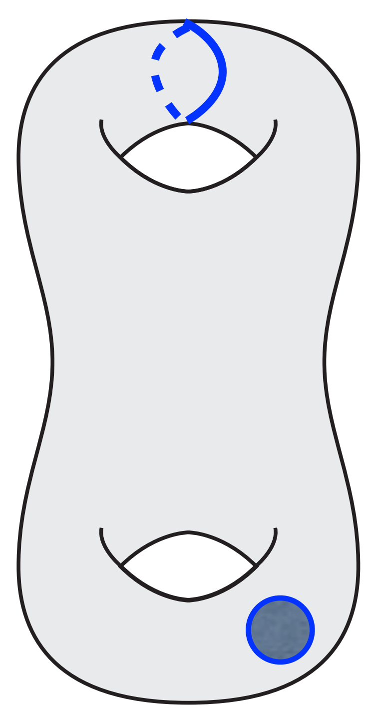
Non-contractible Cycles

- New algorithm to compute the shortest non-contractible cycle in $O(g^3 n \log n)$ time
- Lifts graph to a different covering space than Erickson
- Presentation assumes the cycle is separating and the surface has exactly one boundary cycle



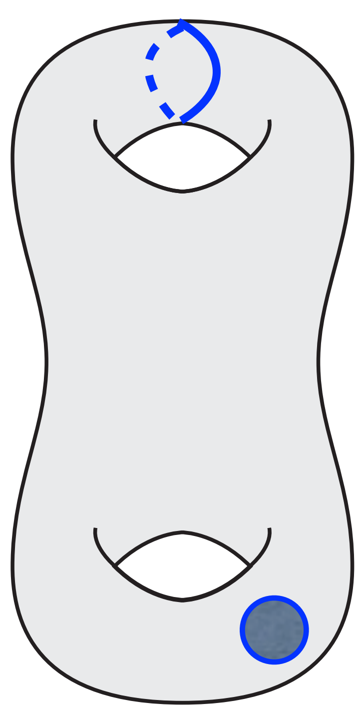
Infinite Cyclic Cover

- Let λ be any non-separating cycle

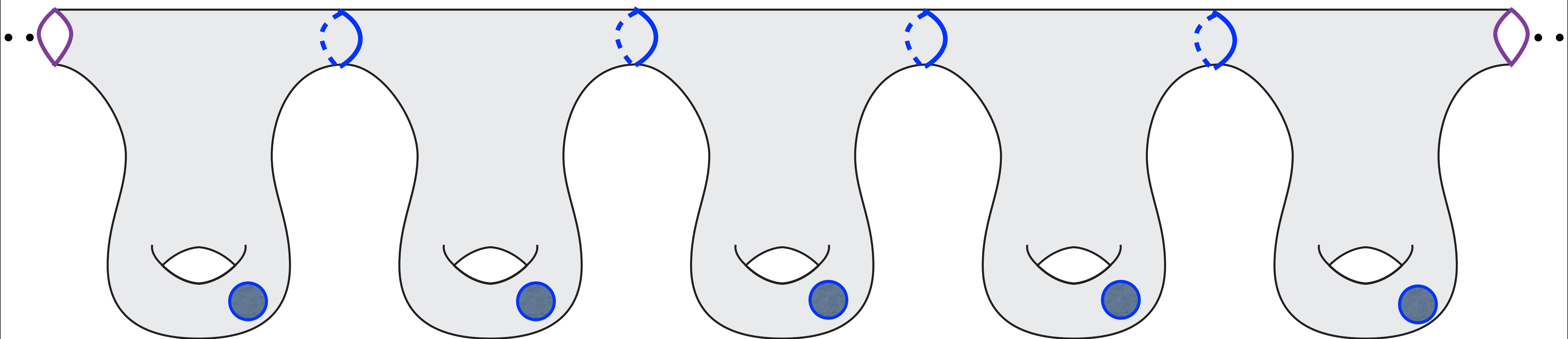


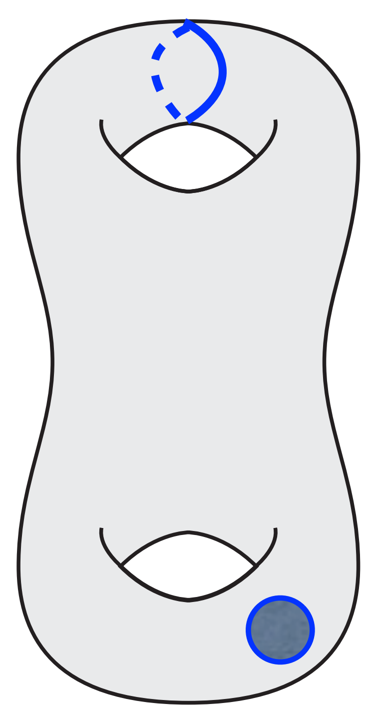
Infinite Cyclic Cover

- Cut the surface along λ, and glue an infinite number of copies together along λ



Infinite Cyclic Cover

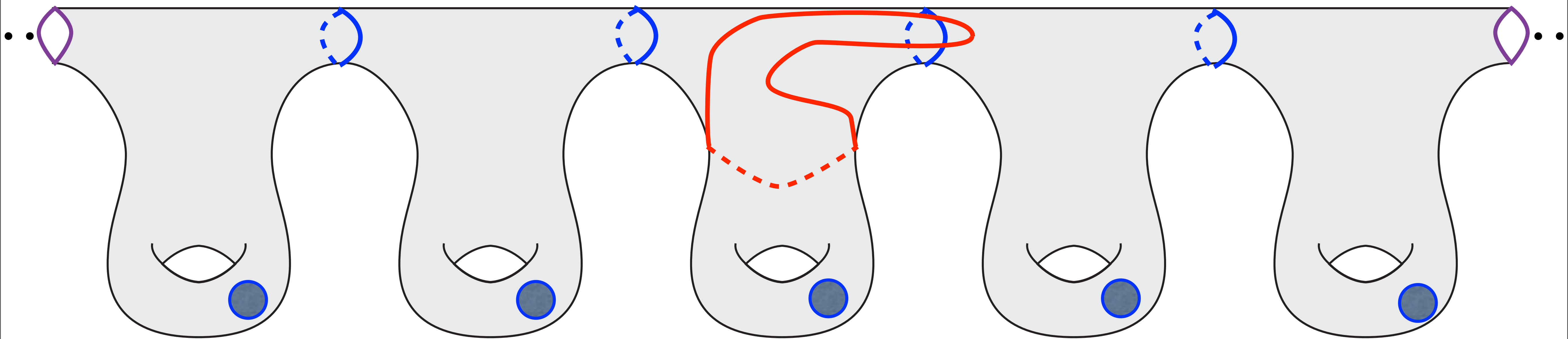


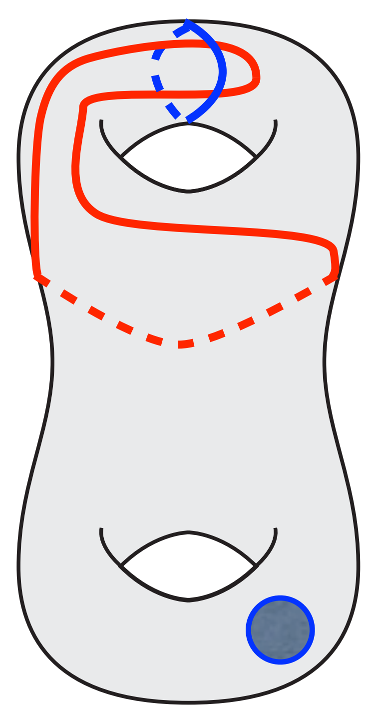


Cycles in the Cover

- A cycle γ lifts to a cycle if and only if it crosses λ left to right the same number of times as it crosses right to left
- Any separating cycle lifts to a cycle
- The shortest non-contractible cycle lifts to a cycle

Cycles in the Cover

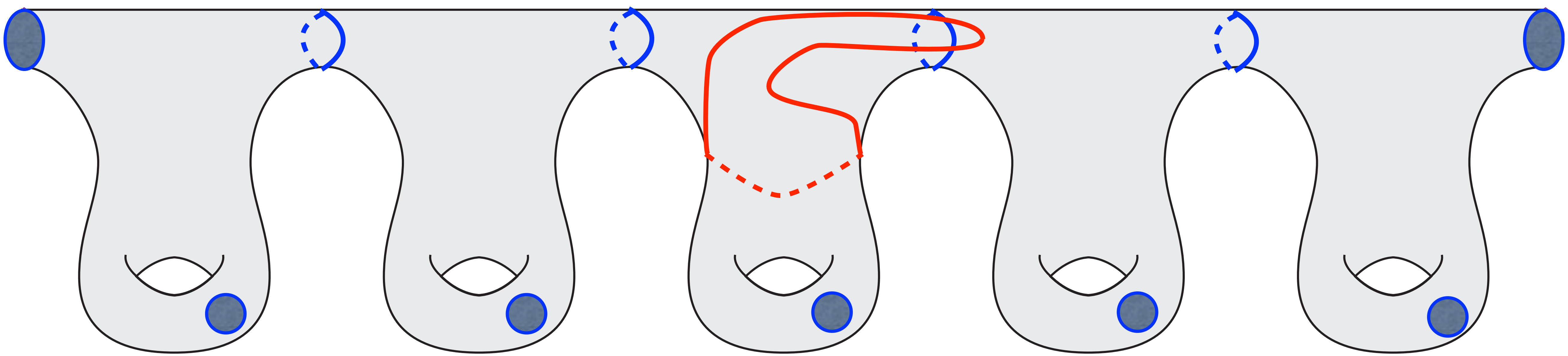


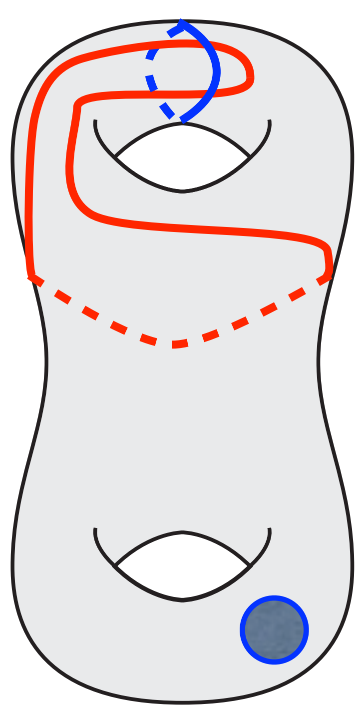


Path Intersections

- The shortest non-contractible cycle intersects at most 2 lifts of any shortest path [Erickson '11]
- We only need 5 copies of the original surface in the cover if we cut along a cycle made from shortest paths
- Leave boundaries at the ends of the left and rightmost copies

Cycles in the Cover





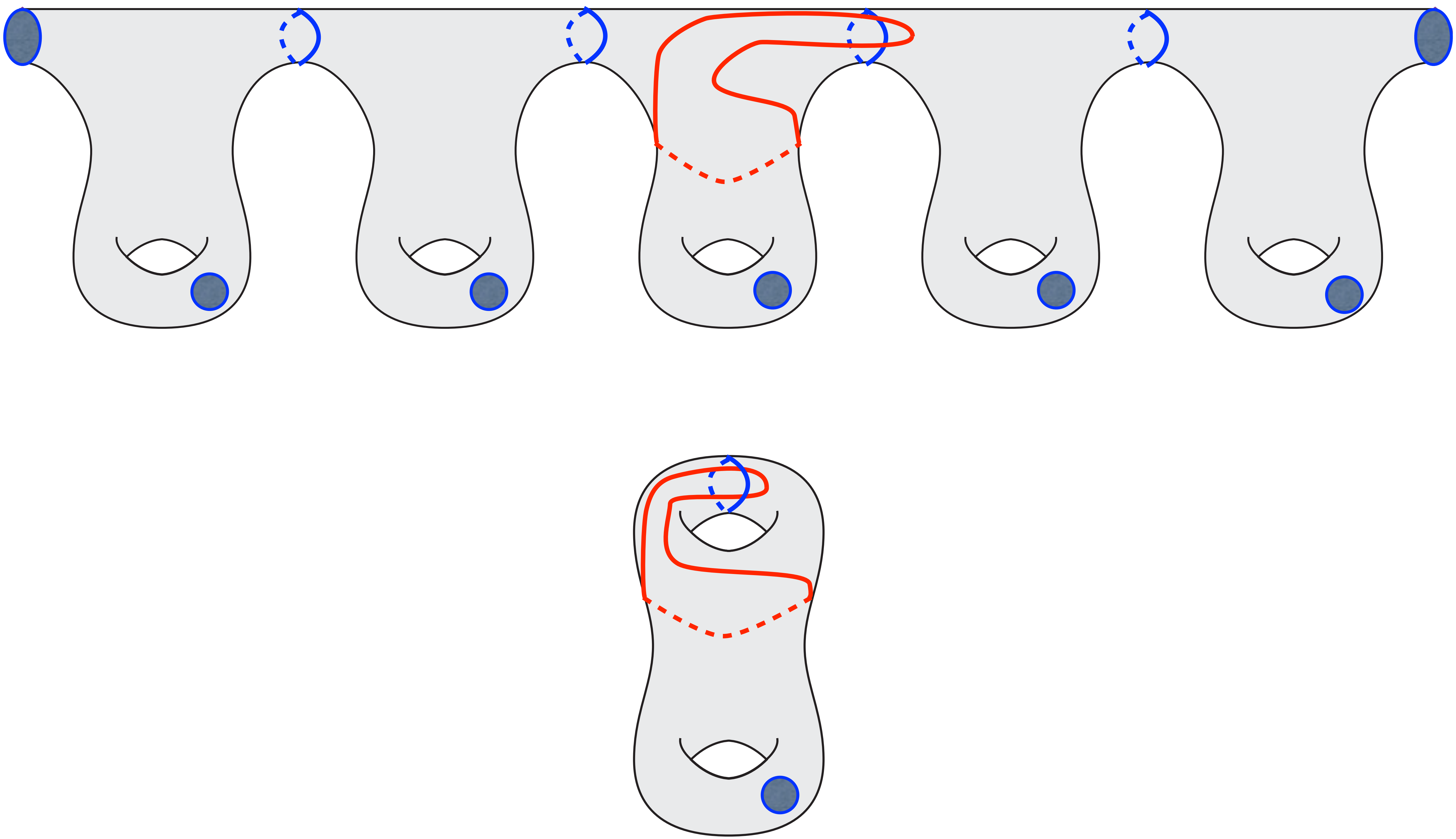
The Lifted Cycle

- The shortest non-contractible cycle in the original surface is the shortest non-contractible cycle in the cover
- We need to carefully choose the cycle we cut along so that we avoid trying to solve the same problem

Separating Boundary

- (Again) compute a system of $2g$ non-separating cycles from shortest paths in $O(n \log n + g\, n)$ time
- At least one of the infinite cyclic covers built from the cycles lifts the shortest non-contractible cycle to the shortest non-separating cycle or one that separates a pair of boundary

Separating Boundary



Search the Covers

- For each cycle λ in the system, build a subset of the infinite cyclic cover
- Find the shortest cycle that is either non-separating or separates a pair of boundary using (a modified version of) Erickson's algorithm

Running Time

- Can search for short cycles in $O(g^2\, n \log n)$ time per covering space
- $O(g^3\, n \log n)$ time spent searching $2g$ covers

Conclusion

- We sketched three algorithms for finding non-trivial cycles in surface embedded graphs
- The first runs in quadratic time, but its speed does not depend on the genus
- The other two run in near-linear time for fixed genus

Thank you