

# Algorithms for Topological and Metric Surfaces



Loïc Dubois

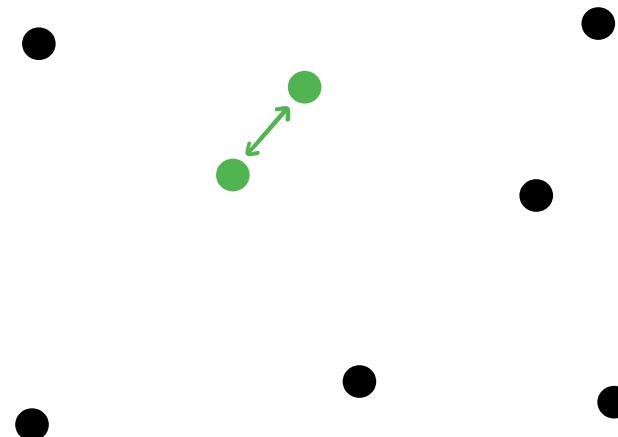
# Computational Geometry

Design algorithms for geometric problems



ex:

given  $n$  points in  $\mathbb{R}^2$ , find two closest points



# Computational Geometry

Design algorithms for geometric problems

This thesis

Focus on surfaces

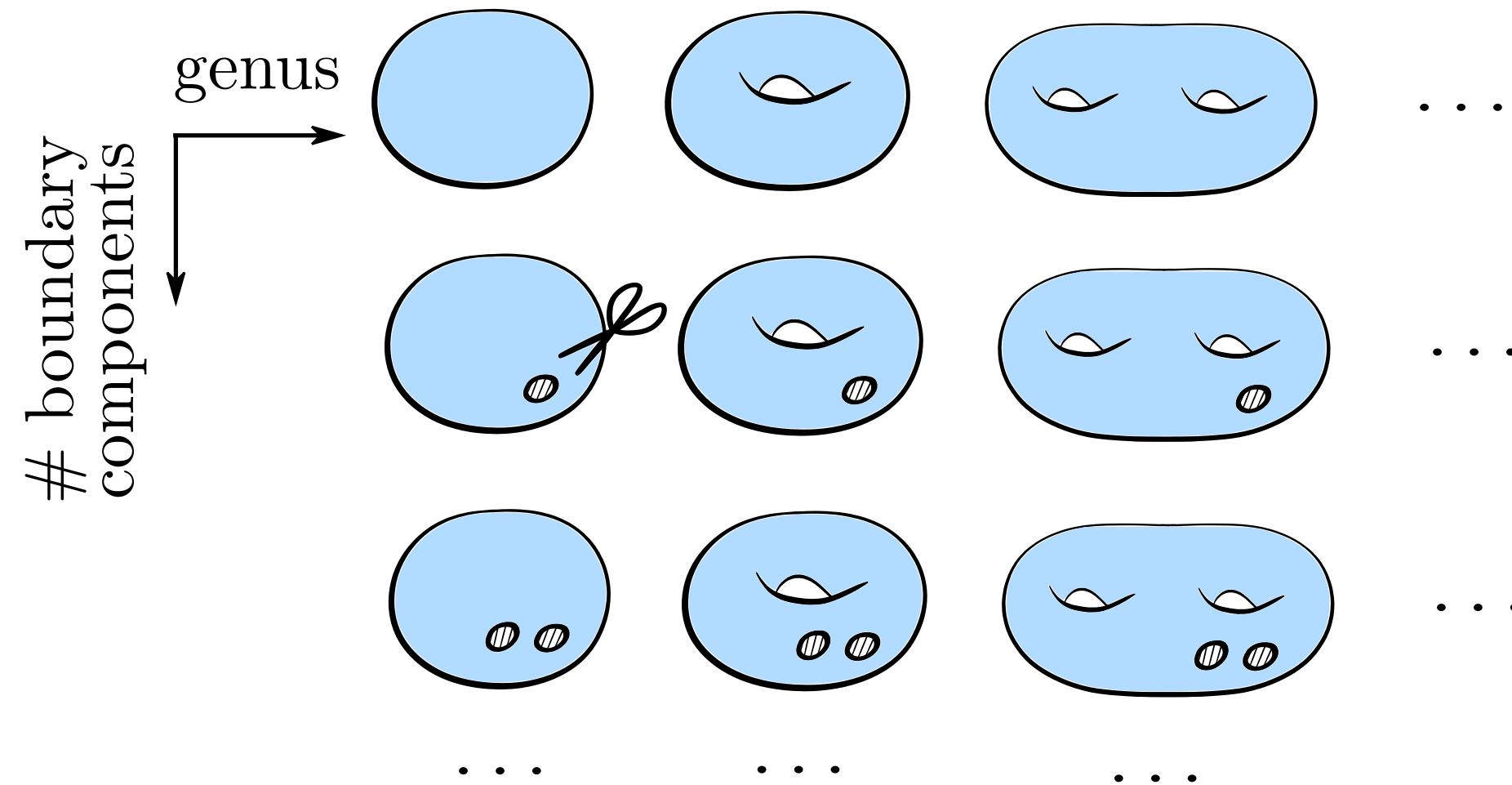


# Topology

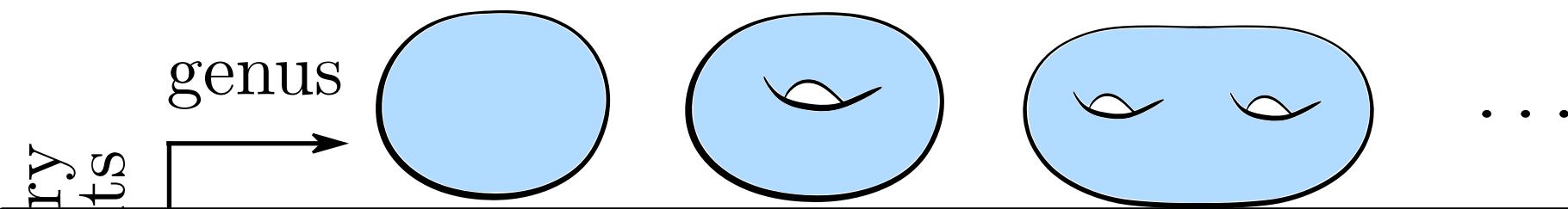


image by Crane and Segerman

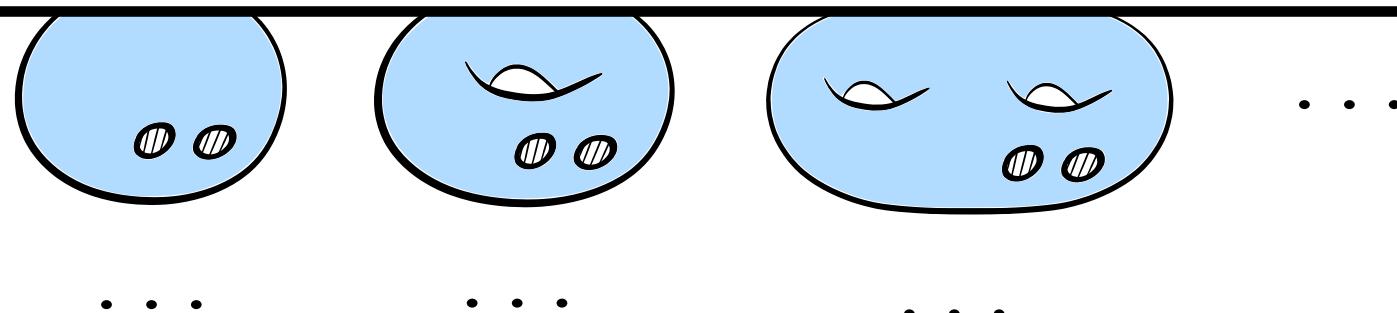
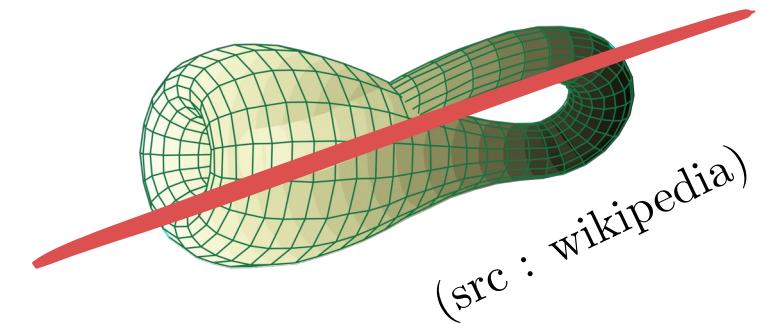
# Topological Surfaces



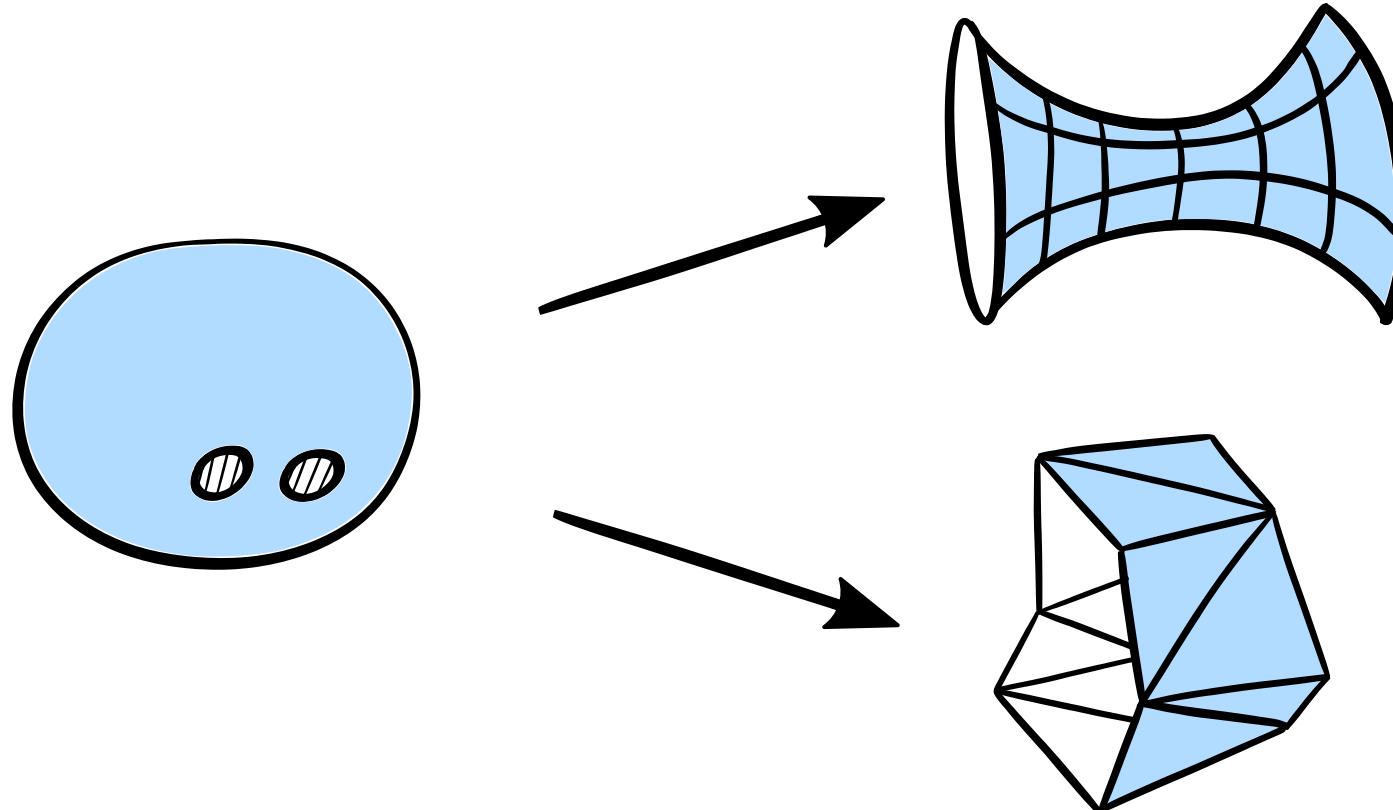
# Topological Surfaces



Only **orientable** surfaces today !



# Metrics on surfaces



# Untangling Graphs

## Computing Delaunay Triangulations

### Other Results and Possible Continuations

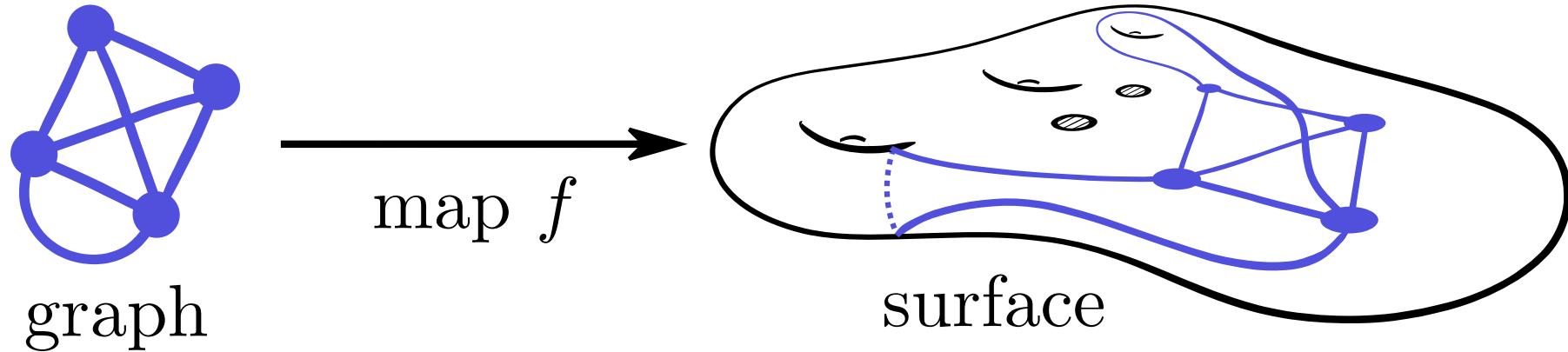
# Untangling Graphs

Computing Delaunay Triangulations

Other Results  
and Possible Continuations

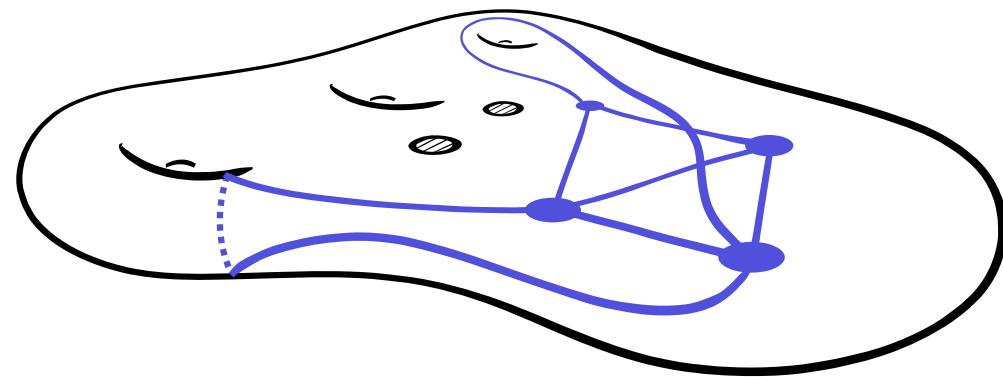
# Problem: untangling graphs

Input:

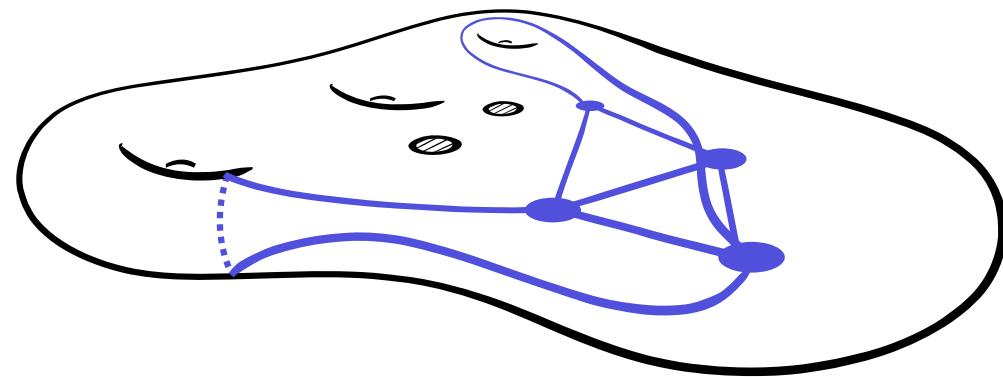


Goal: remove all crossings by deforming  $f$

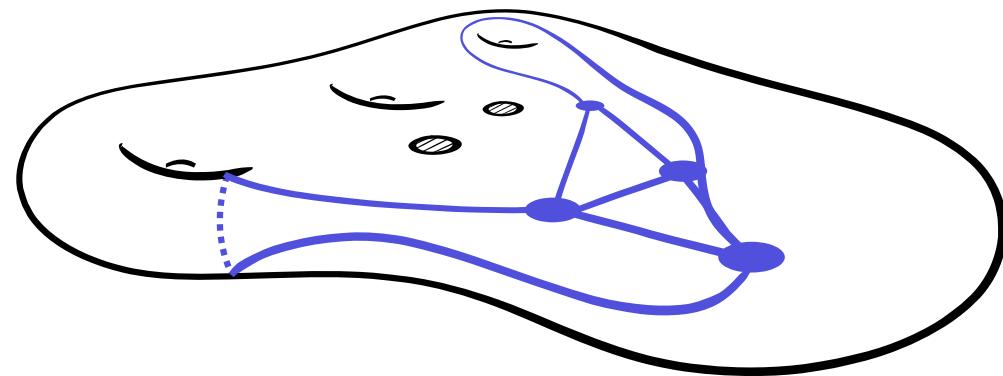
# Problem: untangling graphs



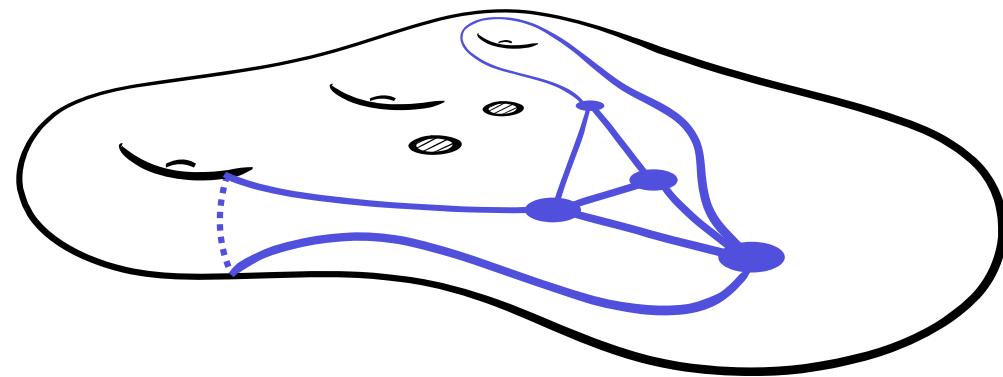
# Problem: untangling graphs



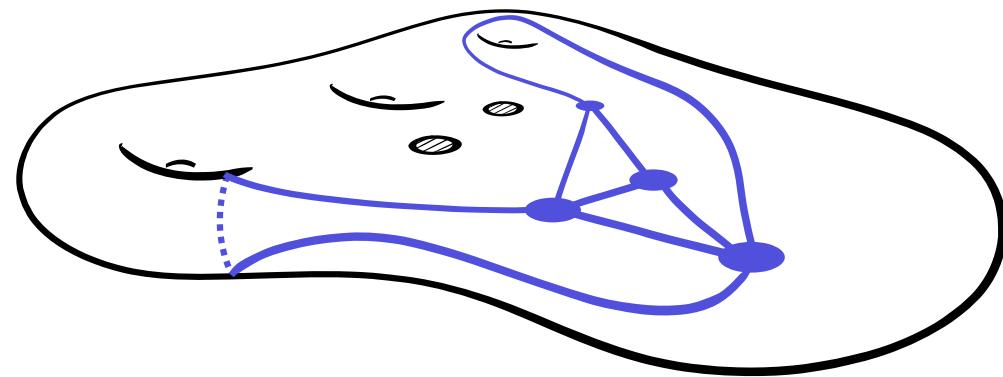
# Problem: untangling graphs



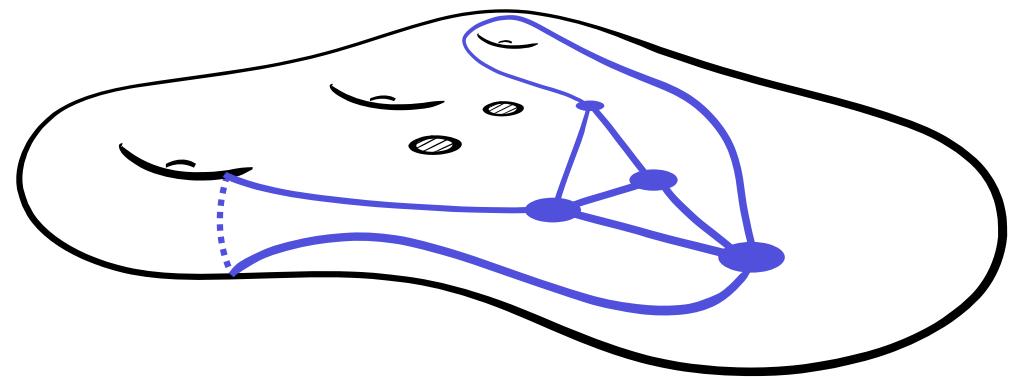
# Problem: untangling graphs



# Problem: untangling graphs



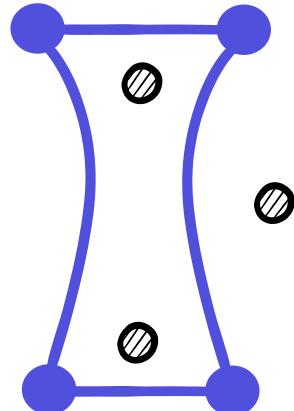
# Problem: untangling graphs

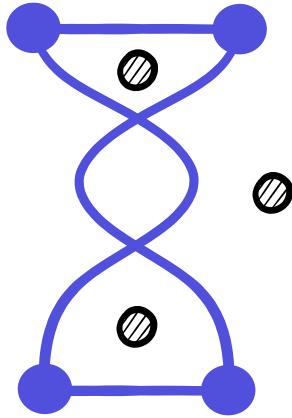


Output: “Yes” (+untangled drawing) or “No”

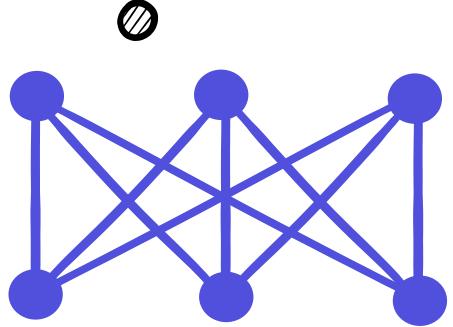
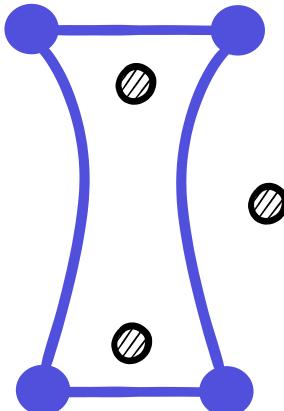


can be  
untangled:

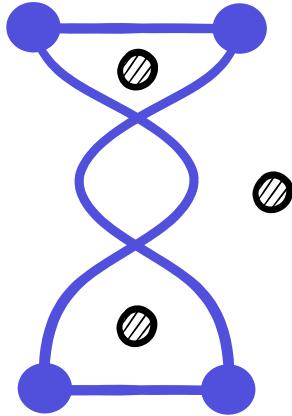




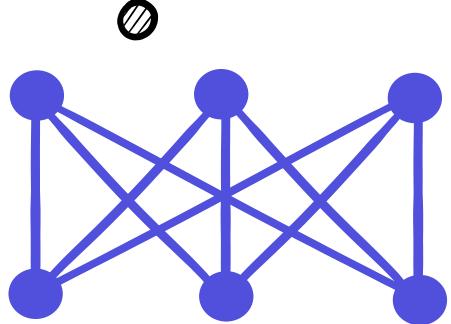
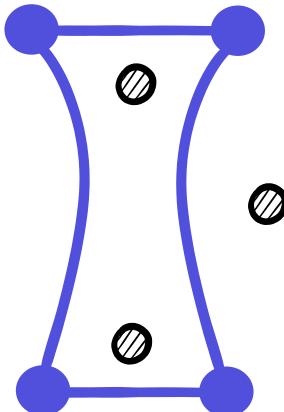
can be  
untangled:



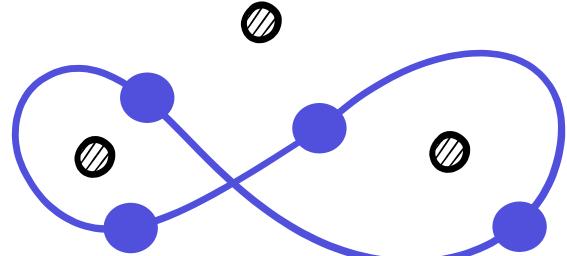
can't be  
untangled



can be  
untangled:



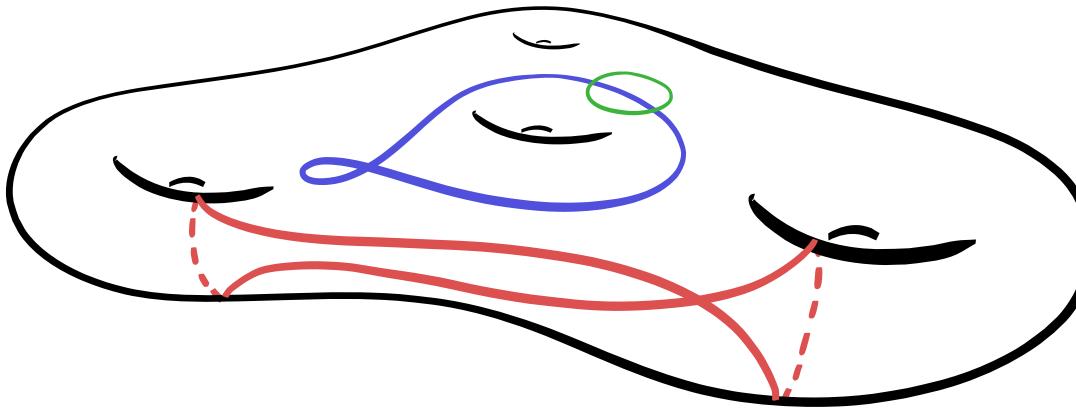
can't be  
untangled



can't be  
untangled

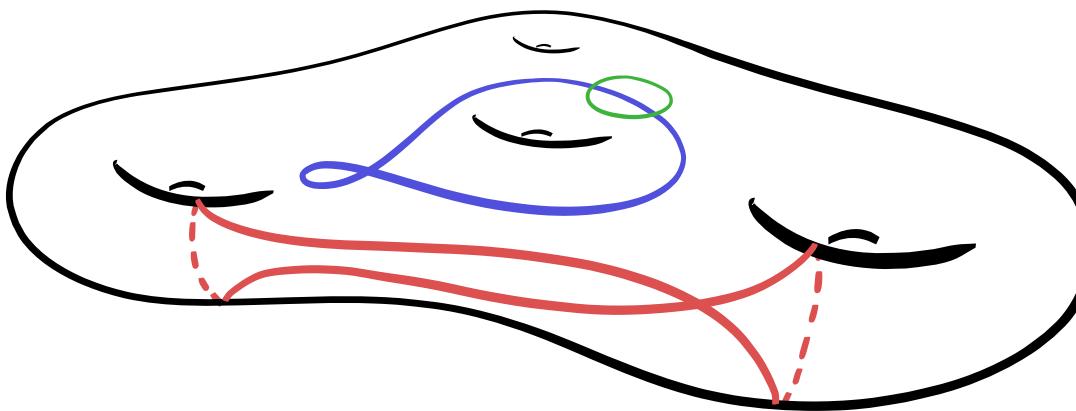
# Related problem: untangling curves

Input: closed curves on surface

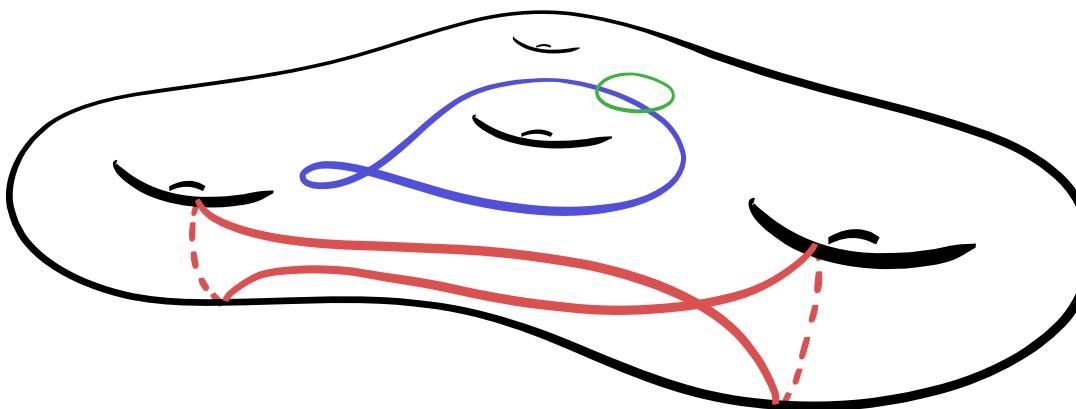


Goal: minimize the crossings by deforming the curves

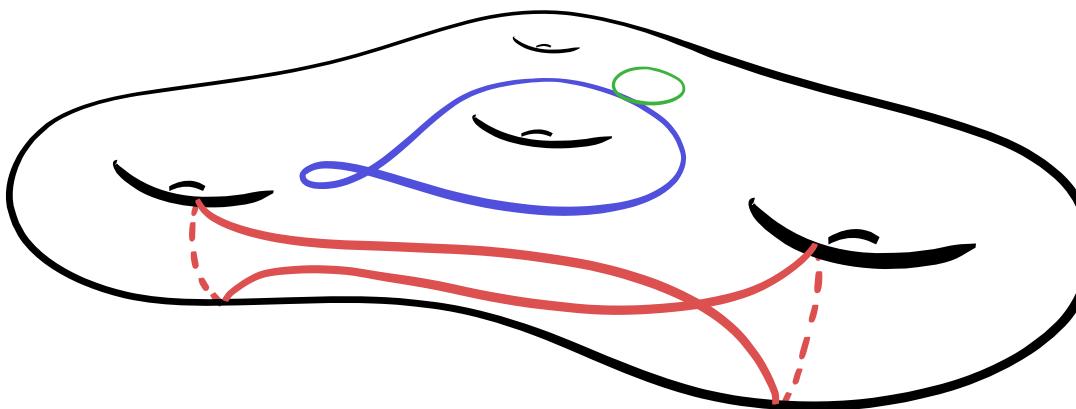
# Related problem: untangling curves



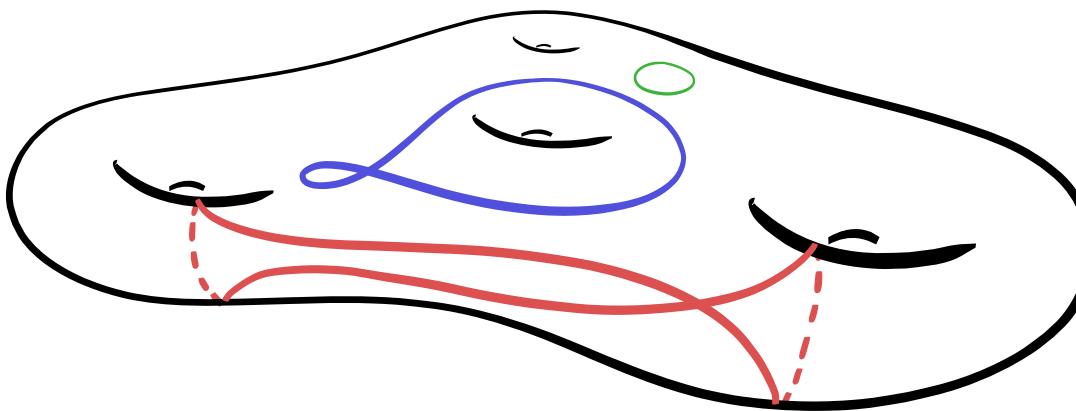
# Related problem: untangling curves



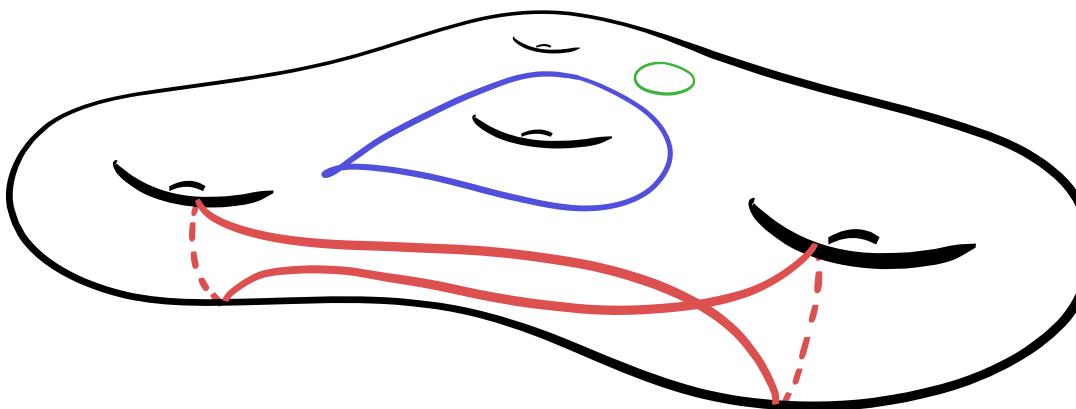
# Related problem: untangling curves



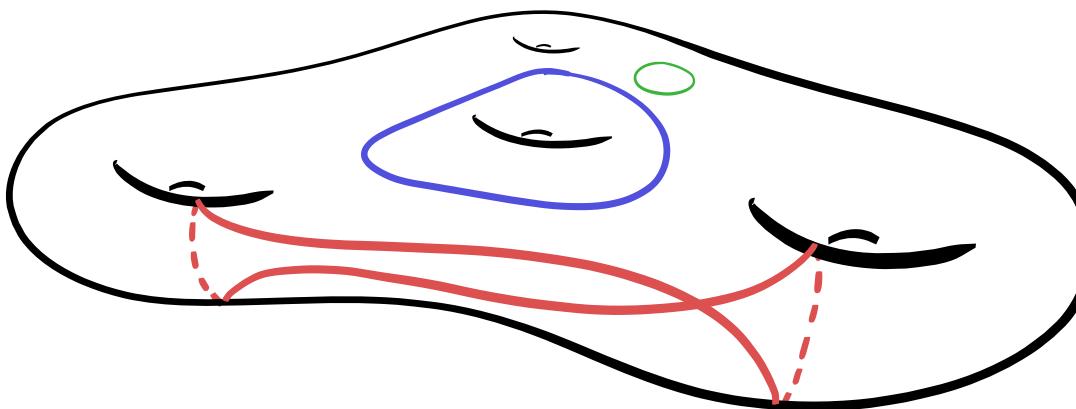
# Related problem: untangling curves



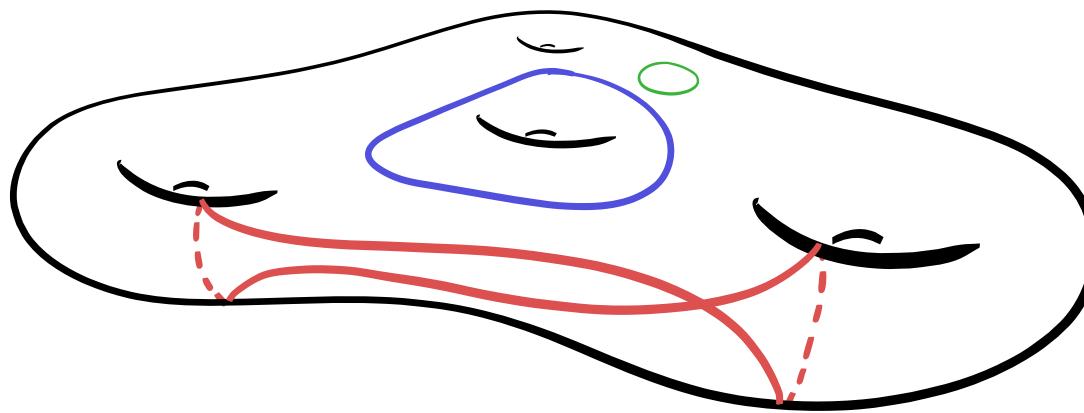
# Related problem: untangling curves



# Related problem: untangling curves



# Related problem: untangling curves



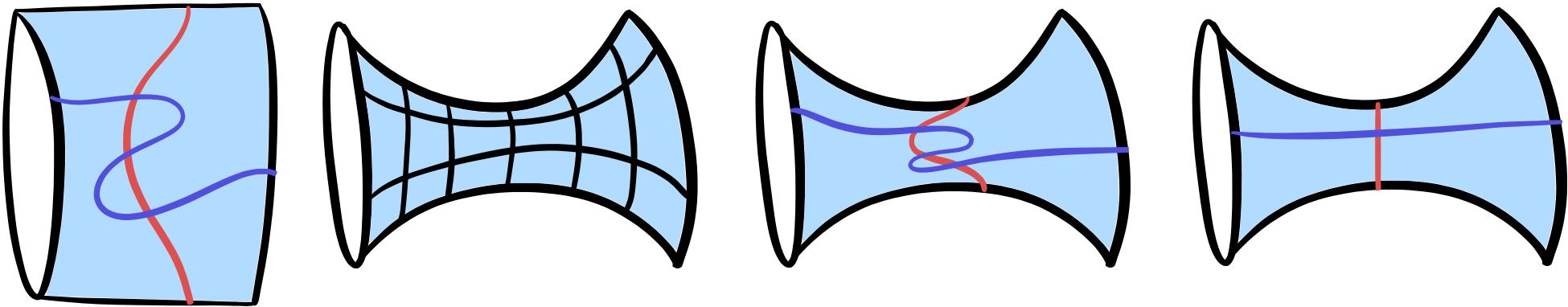
Output:  $\min \#$  of crossings (+ optimal curves)

# Many previous works!

- |                         |                               |
|-------------------------|-------------------------------|
| Poincaré, 1905          | de Graaf and Schrijver, 1987  |
| Dehn, 1911              | Dynnikov, 2002                |
| Dehn, 1912              | Paterson, 2002                |
| Reinhart, 1962          | Gonçalves et al., 2005        |
| Zieschang, 1965         | Schaefer et al., 2008         |
| Chillingworth, 1969     | Lazarus and Rivaud, 2012      |
| Zieschang, 1969         | Erickson and Whittlesey, 2013 |
| Chillingworth, 1971     | Arettines, 2015               |
| Turaev, 1979            | Chang et al., 2018            |
| Birman and Series, 1984 | Despré and Lazarus, 2019      |
| Cohen and Lustig, 1984  | Fulek and Tóth, 2020          |
| Hass and Scott, 1985    | Chang and de Mesmay, 2022     |
| Lustig, 1987            | Lackenby, 2024                |
- · ·

# Method for untangling curves

Poincaré, 1905



1. give special shape to surface
2. straighten the curves

# Negatively curved surfaces

surfaces that look like  
that everywhere:

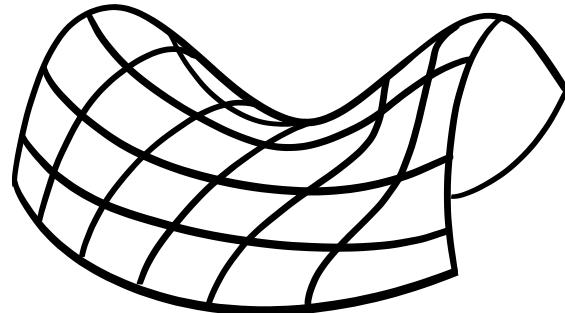


image by Susan Lombardo

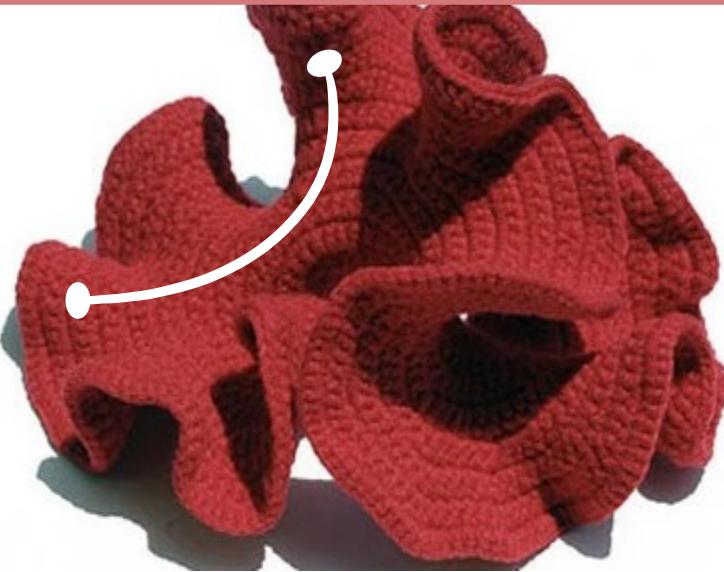
almost all surfaces  
can be curved negatively

# Key property

every path can be deformed  
into a unique straight path

(homotopy)

straightening  
removes  
crossings



(specific to these  
surfaces)

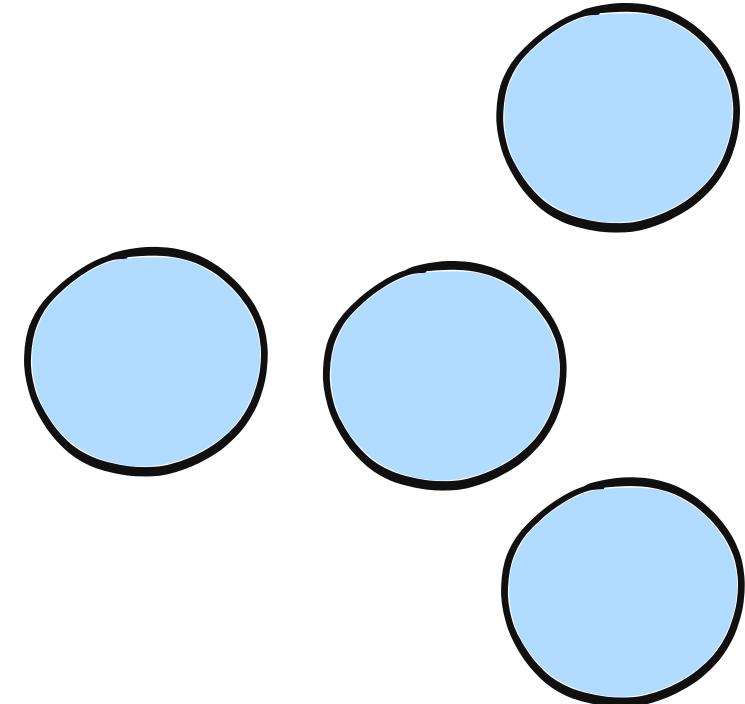


# Algorithms for untangling curves

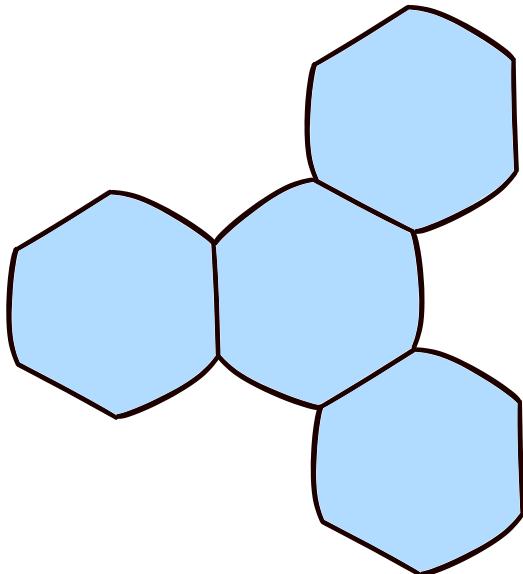
Birman and Series, 1984

Fulek and Tóth, 2020

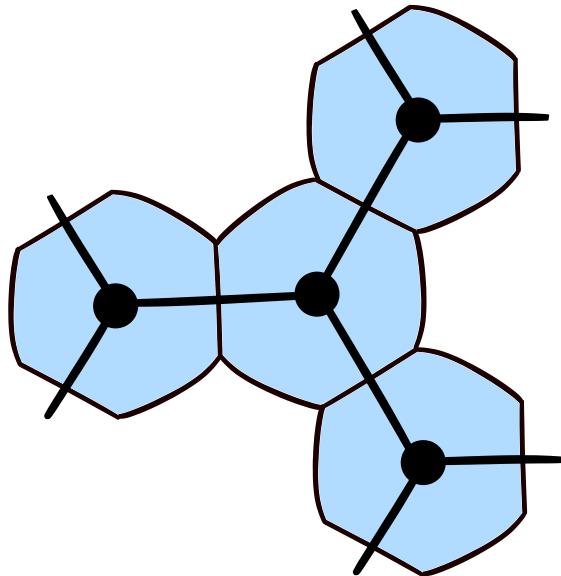
# Algorithms for untangling curves



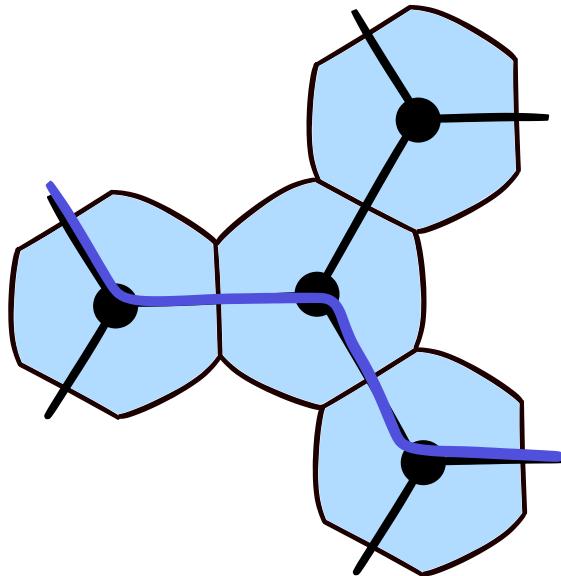
# Algorithms for untangling curves



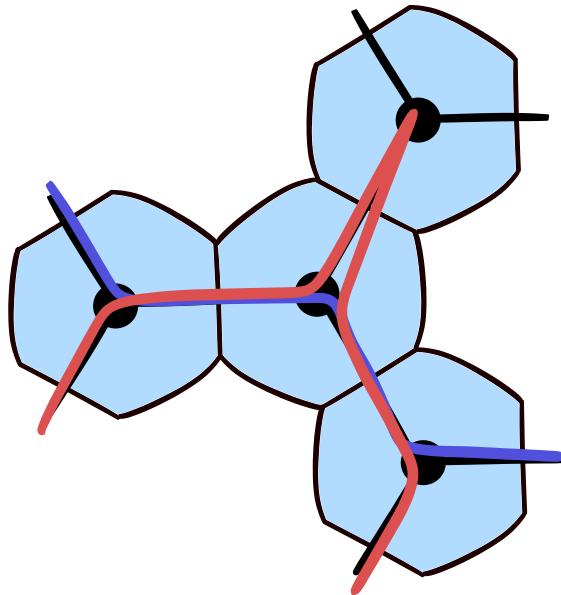
# Algorithms for untangling curves



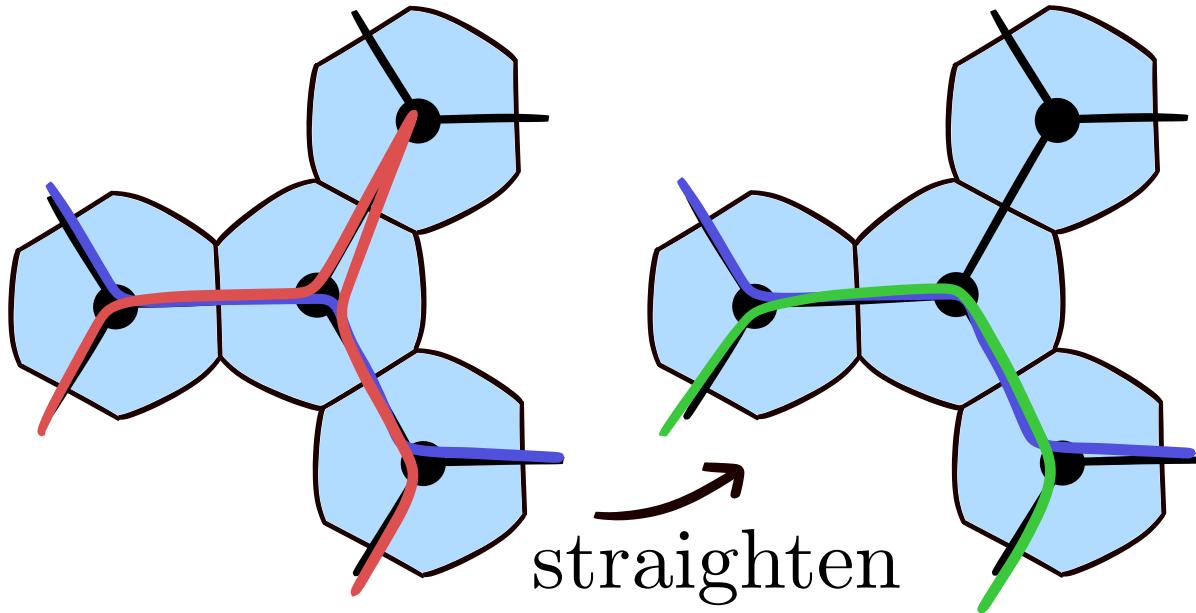
# Algorithms for untangling curves



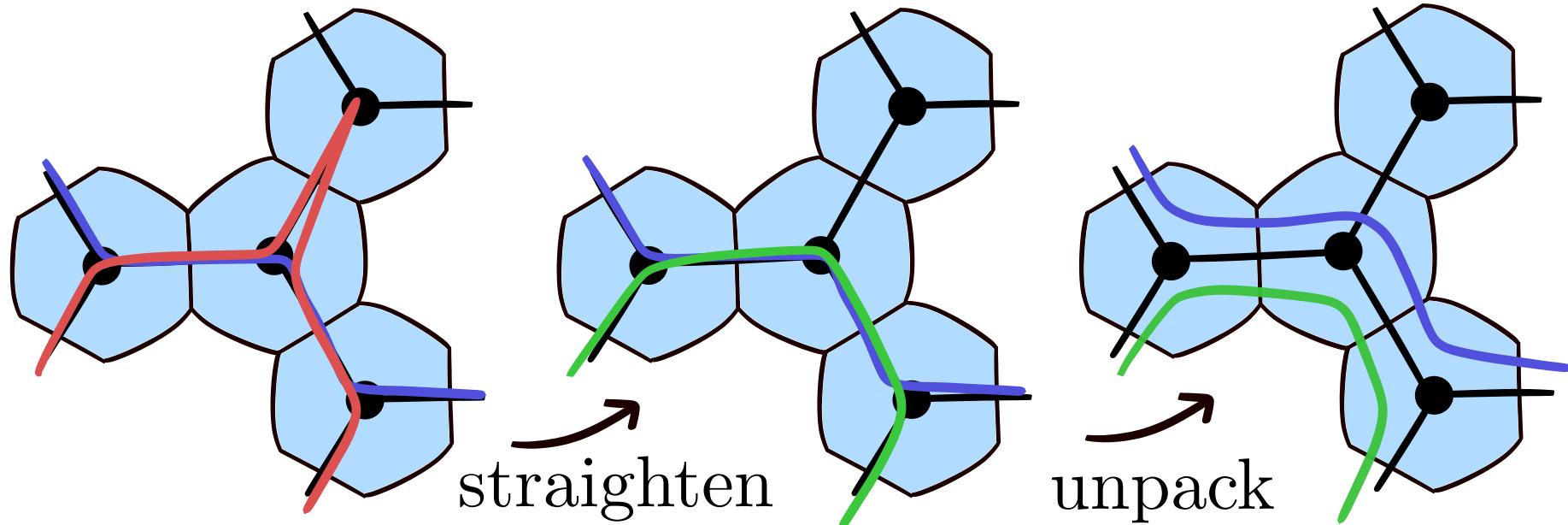
# Algorithms for untangling curves



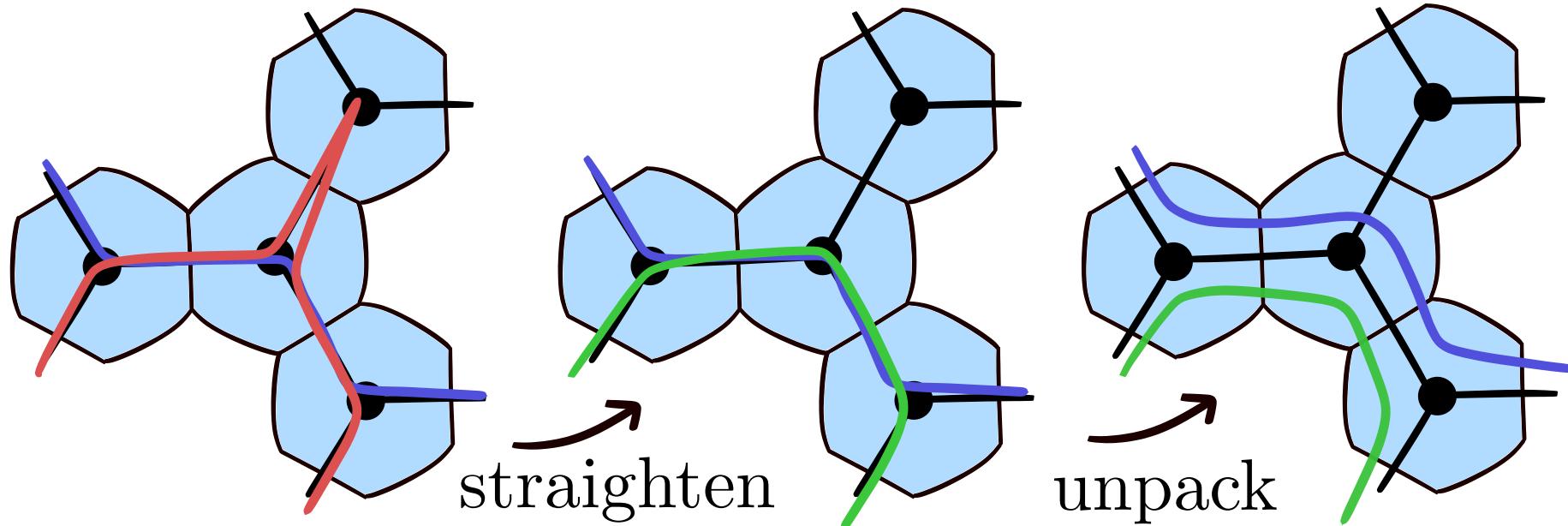
# Algorithms for untangling curves



# Algorithms for untangling curves

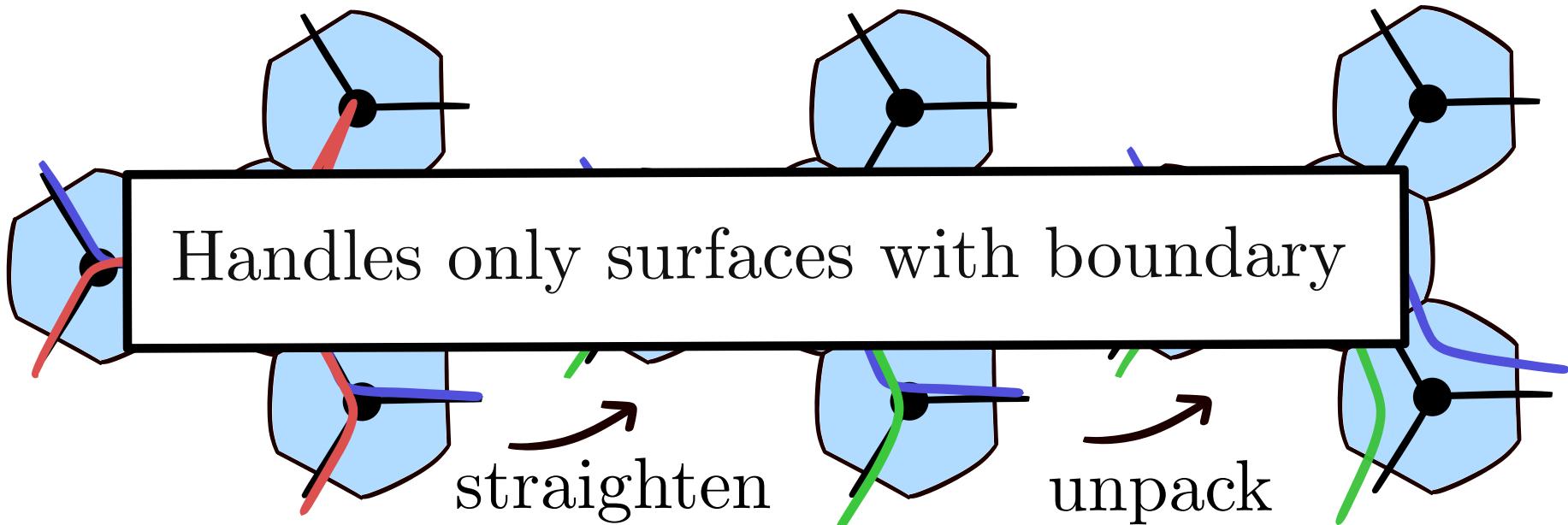


# Algorithms for untangling curves



every walk can be deformed  
into a unique straight walk

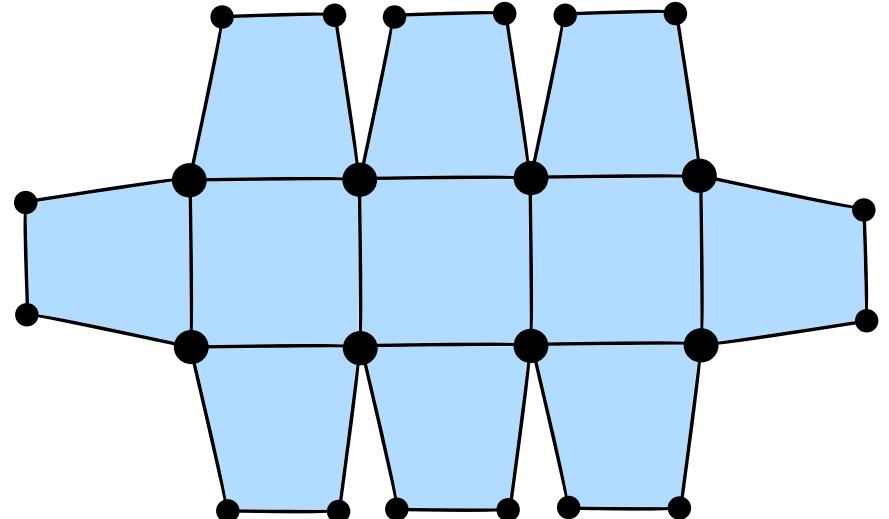
# Algorithms for untangling curves



every walk can be deformed  
into a unique straight walk

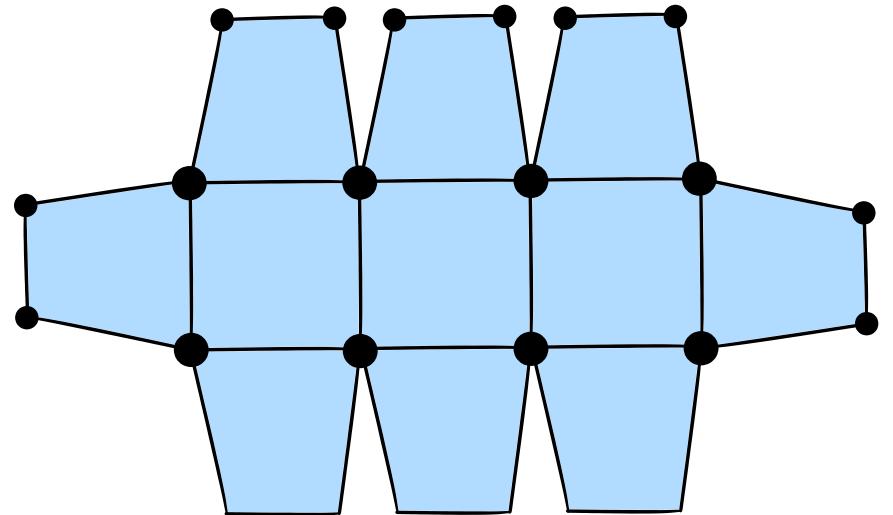
# Algorithms for untangling curves

Lazarus and Rivaud, 2012  
Erickson and Whittlesey, 2013

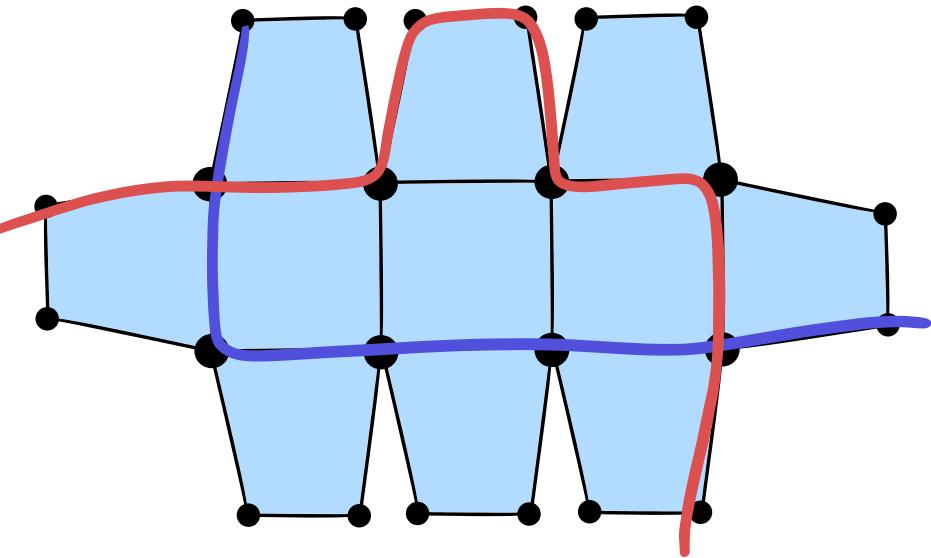


Model for surfaces  
without boundary

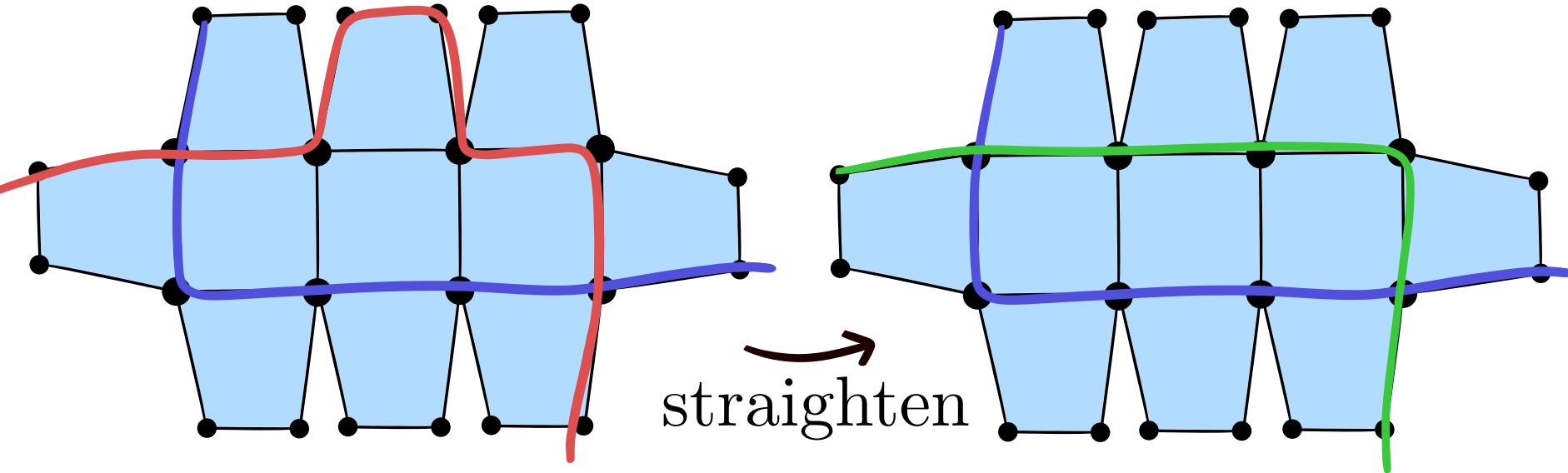
# Algorithms for untangling curves



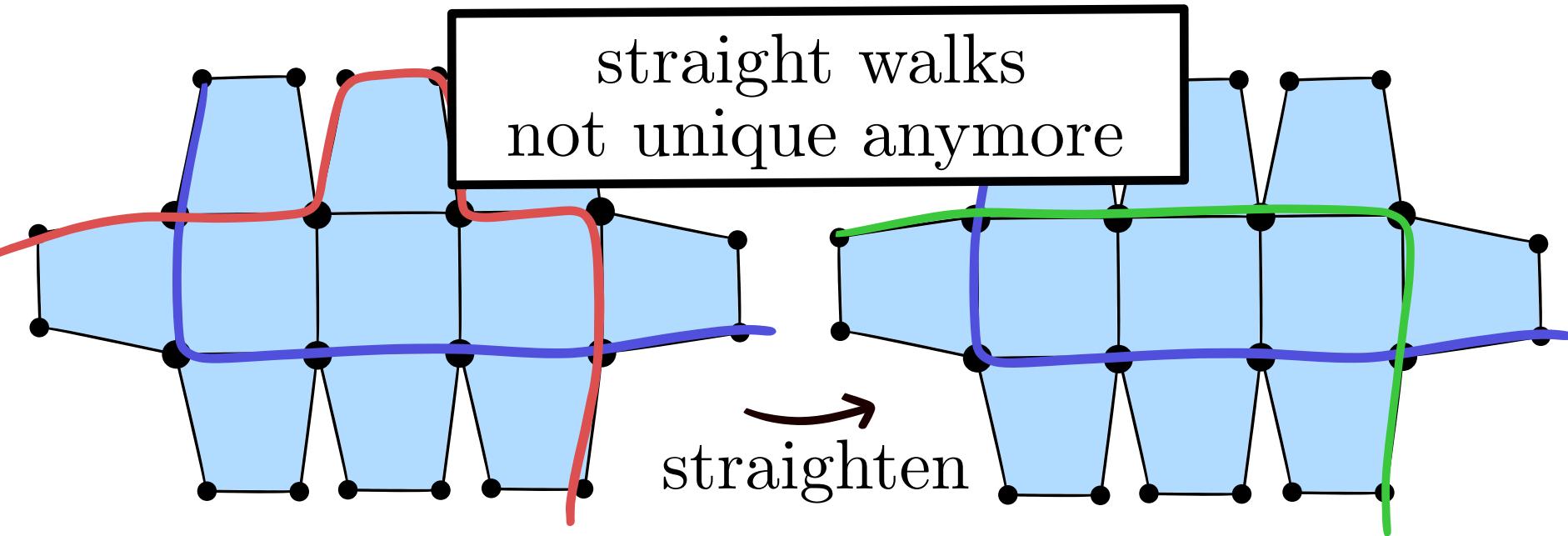
# Algorithms for untangling curves



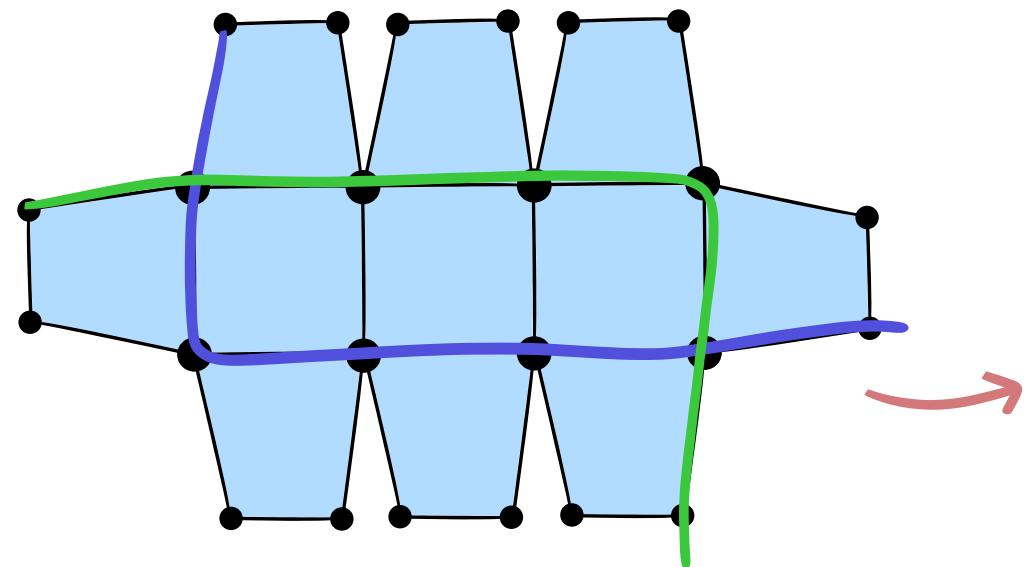
# Algorithms for untangling curves



# Algorithms for untangling curves



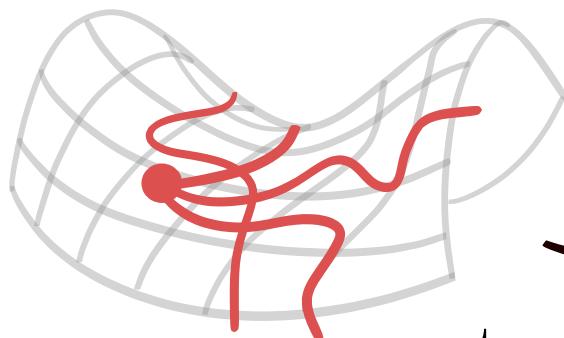
# Algorithms for untangling curves



Despré and Lazarus, 2019

# Method for untangling graphs

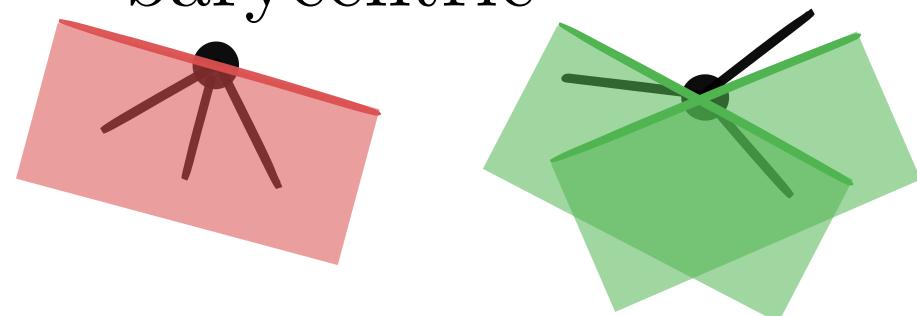
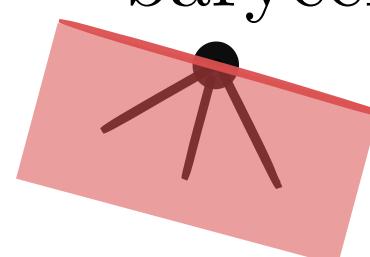
Tutte, 1963



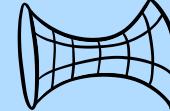
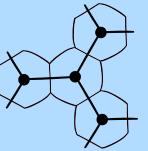
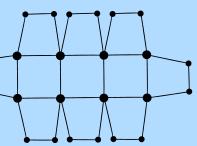
straighten  
edges



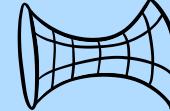
make vertices  
barycentric



# Straightening to untangle: summary

	Curves	Graphs
Method	negatively curved surface	
Algos	patch system 	system of quads 

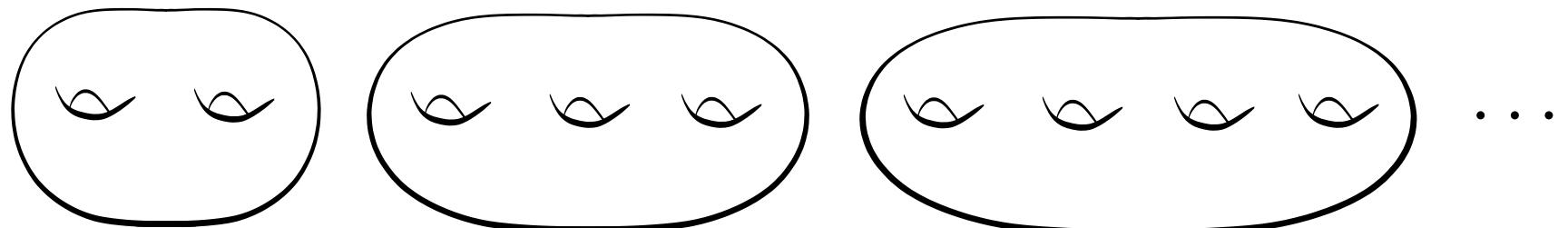
# Our results

	Curves	Graphs
Method	negatively curved surface 	
Algos	reducing triangulations improved algos for untangling curves	first algos for untangling graphs discrete analogue of Tutte embeddings

A new tool:

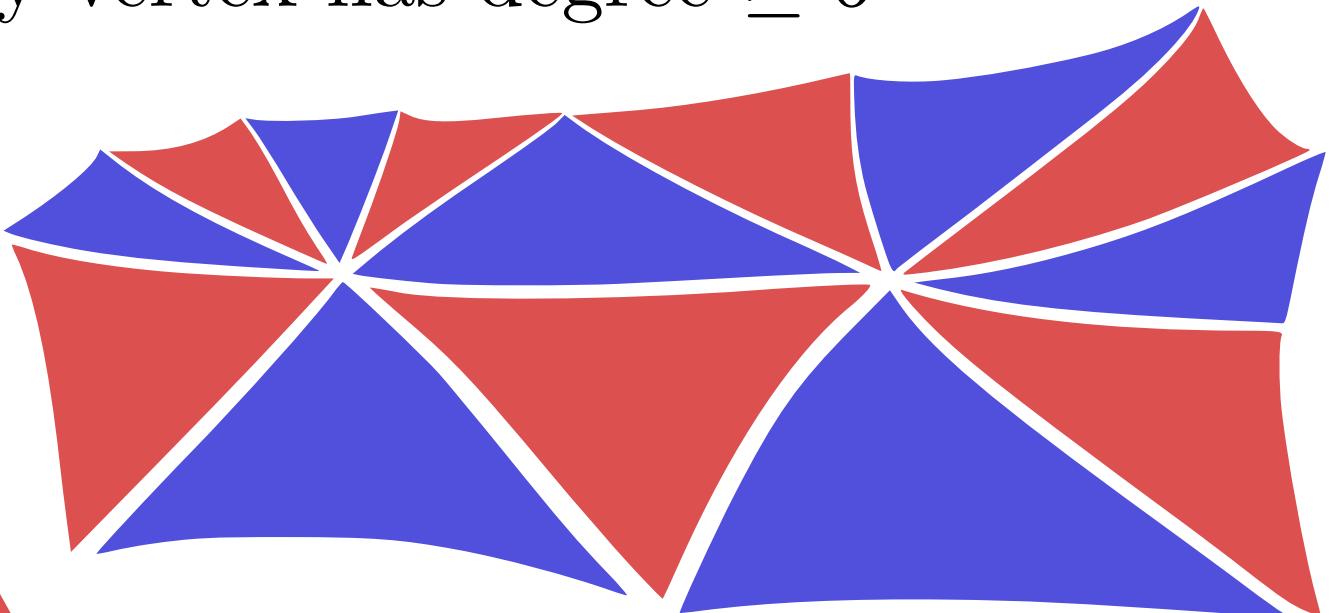
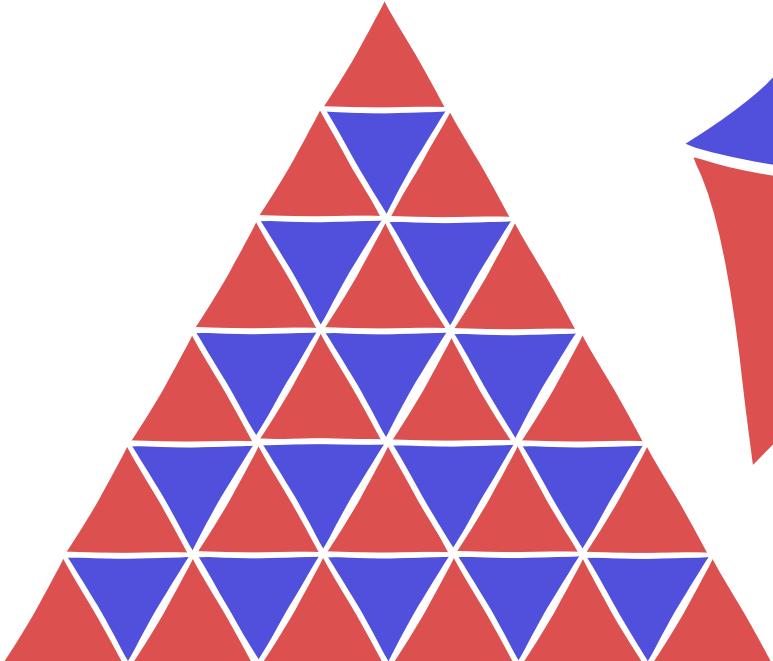
# Reducing triangulations

We focus on surfaces without boundary  
of genus  $\geq 2$

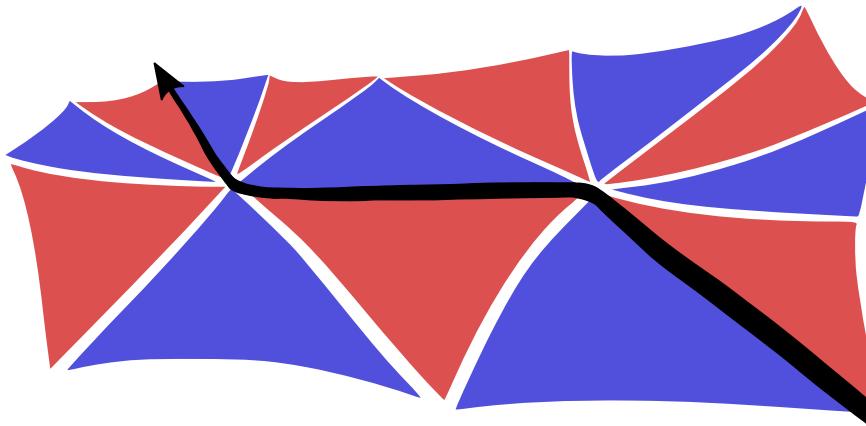


# Reducing triangulations

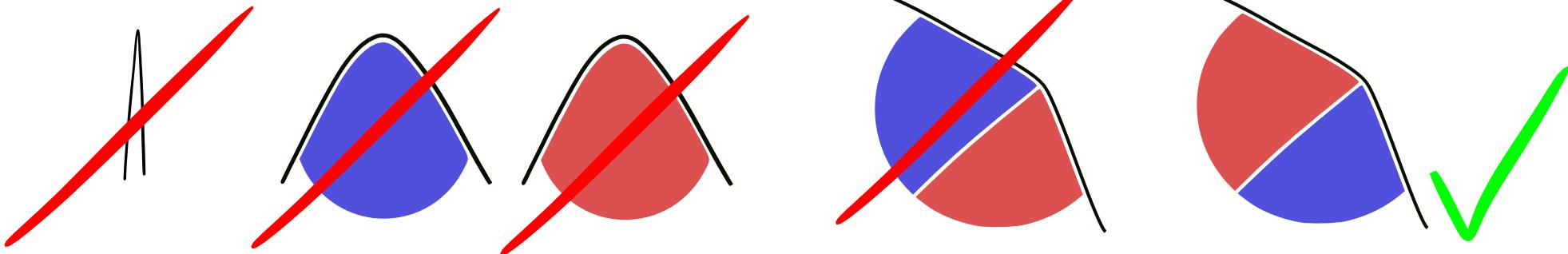
dual is bipartite and  
every vertex has degree  $\geq 6$



# Reduced walks

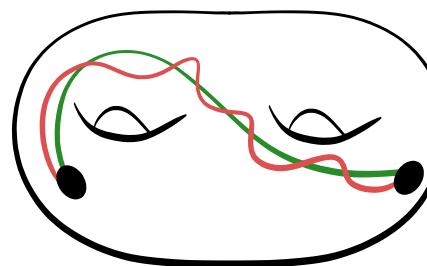
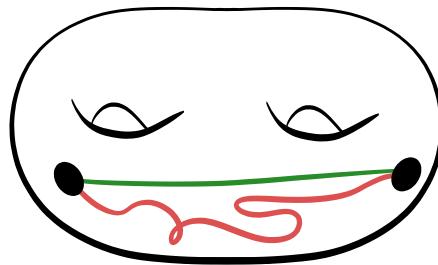


no bad turn



# Properties of reduced walks

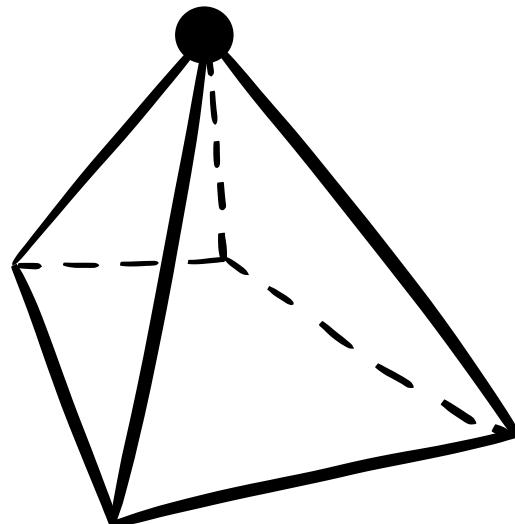
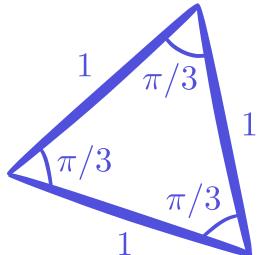
every walk can be deformed into a unique reduced walk, computable in linear time



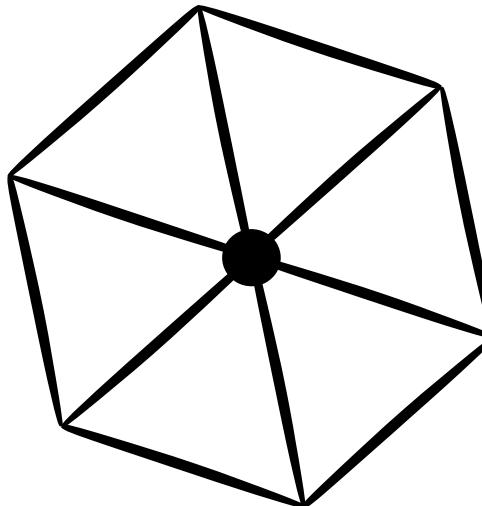
reduced walks are stable upon reversal and subwalk

# Analogy: vertex degree vs. curvature

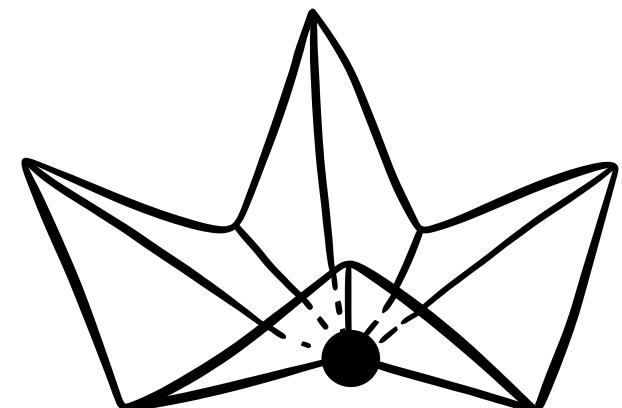
---



degree  $< 6$



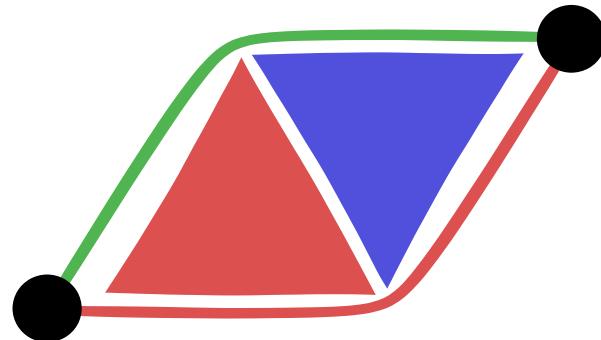
degree 6



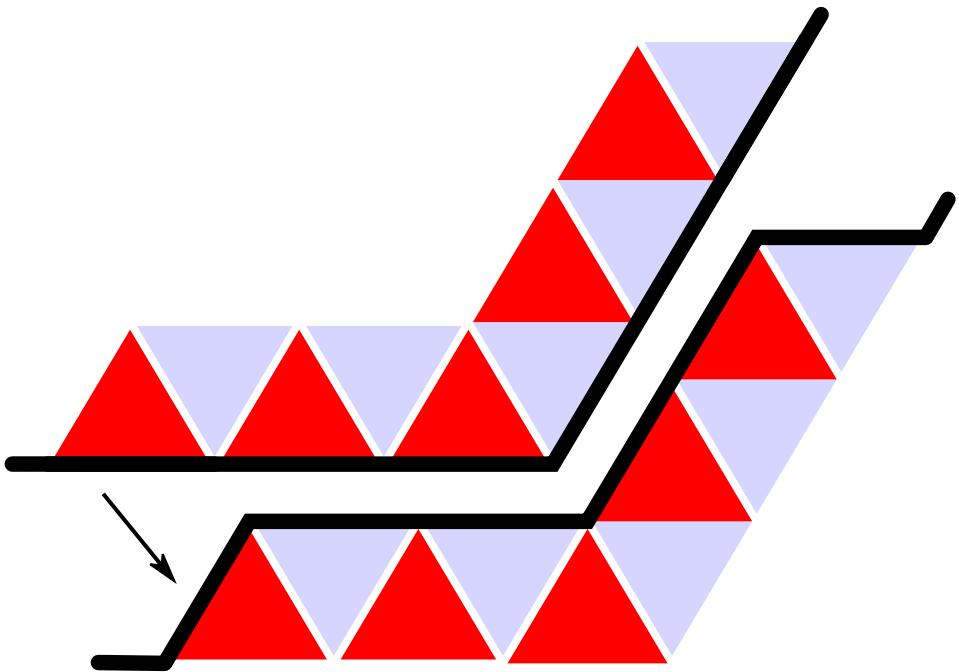
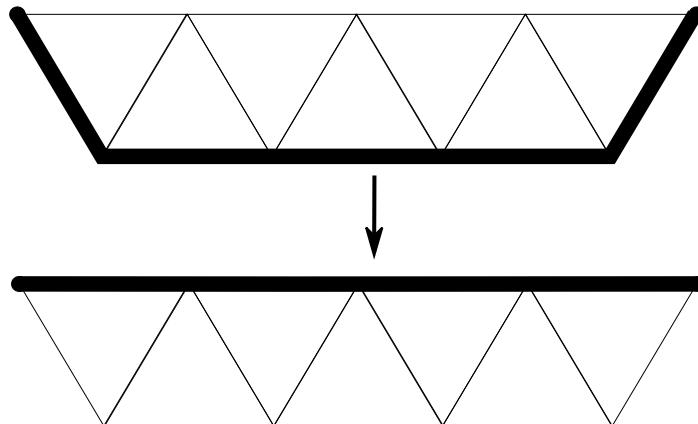
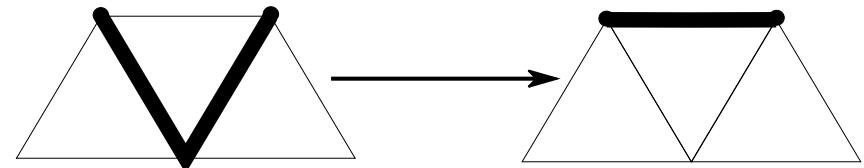
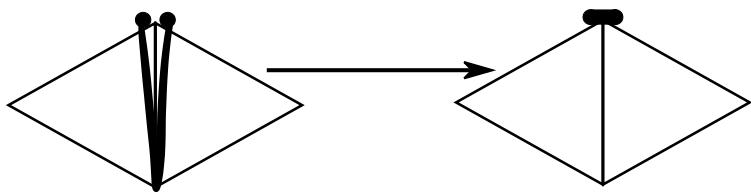
degree  $> 6$

## Purpose of the coloring

---



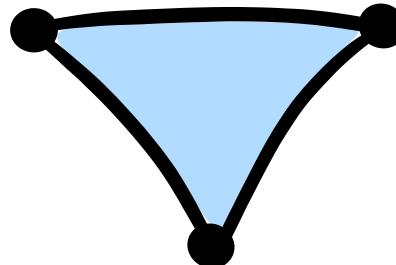
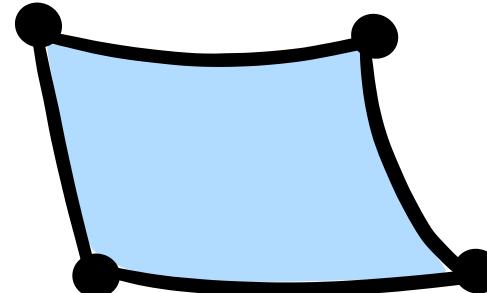
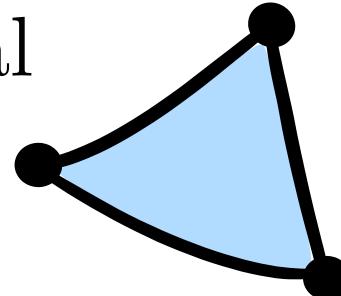
# Reducing a walk



Untangling graphs using  
reducing triangulations

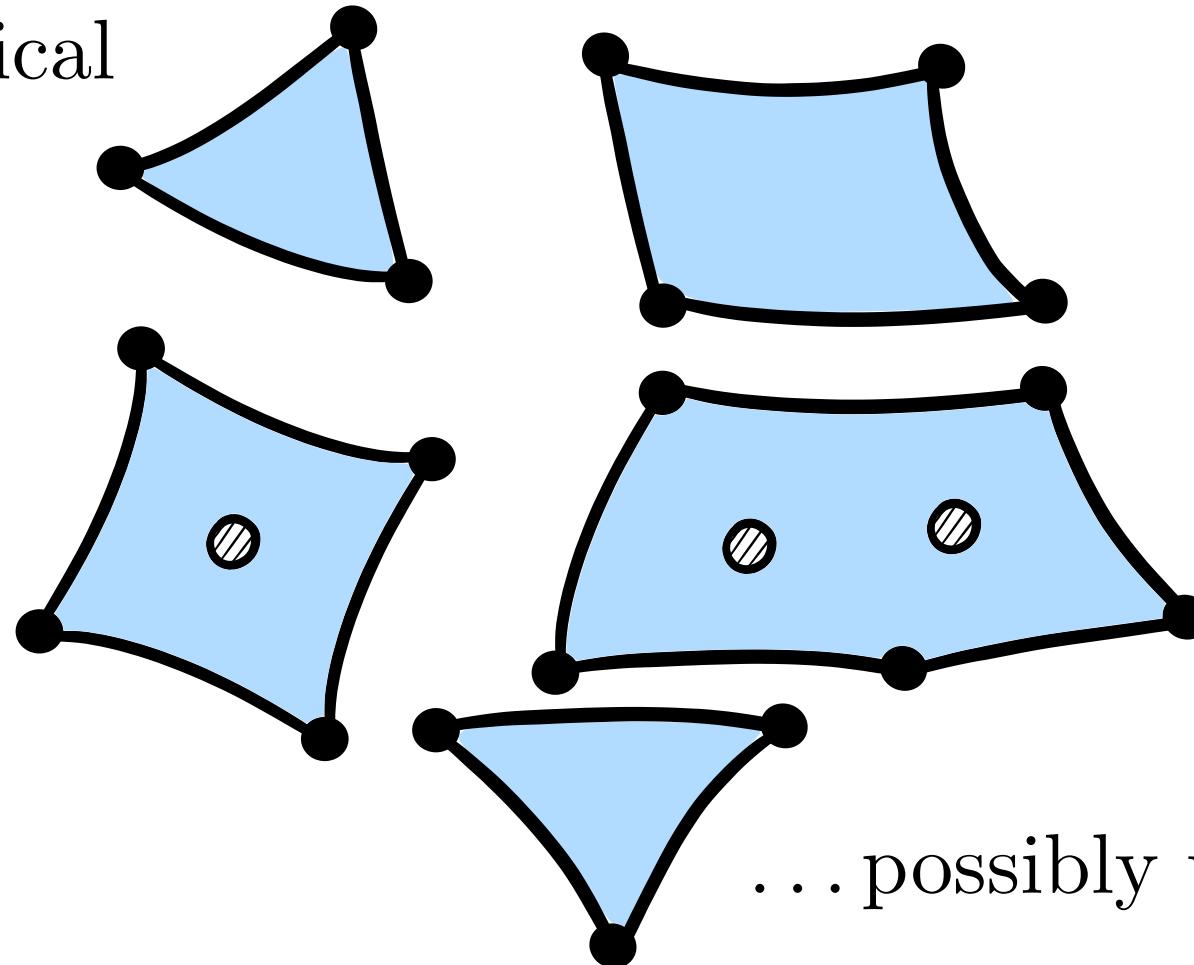
# Input

take topological  
polygons...



# Input

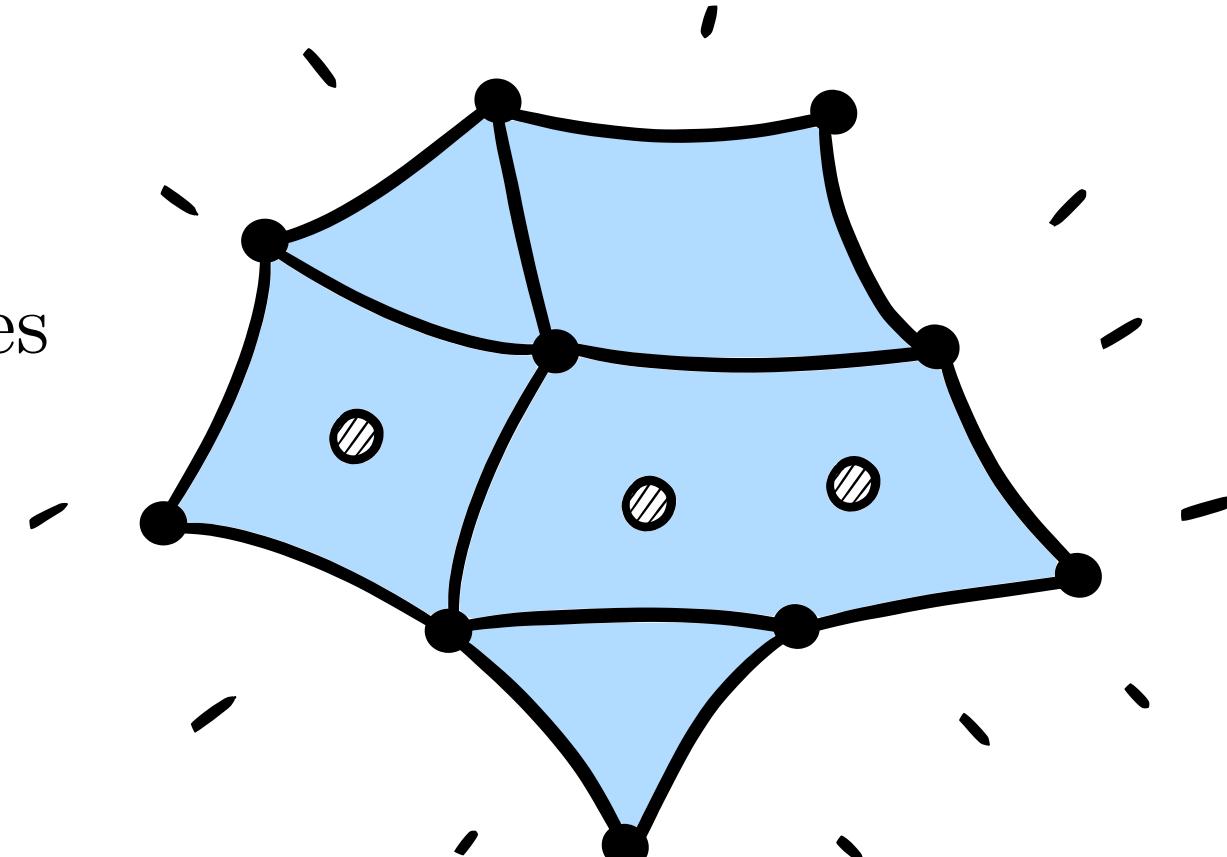
take topological  
polygons...



... possibly with holes

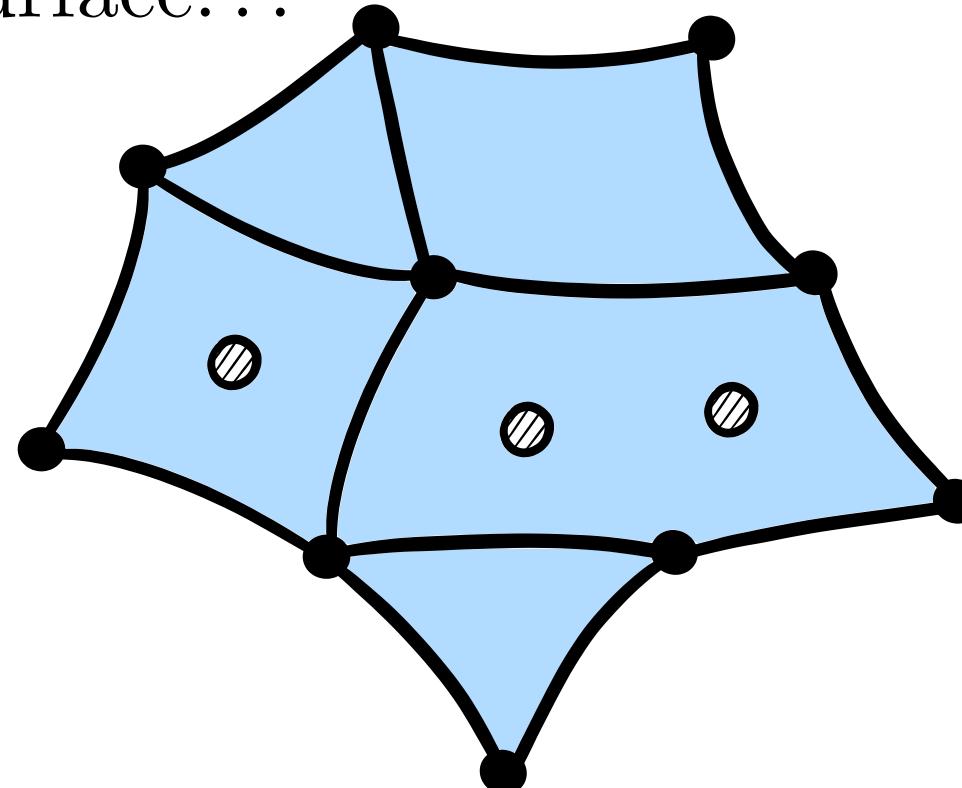
# Input

glue edges



# Input

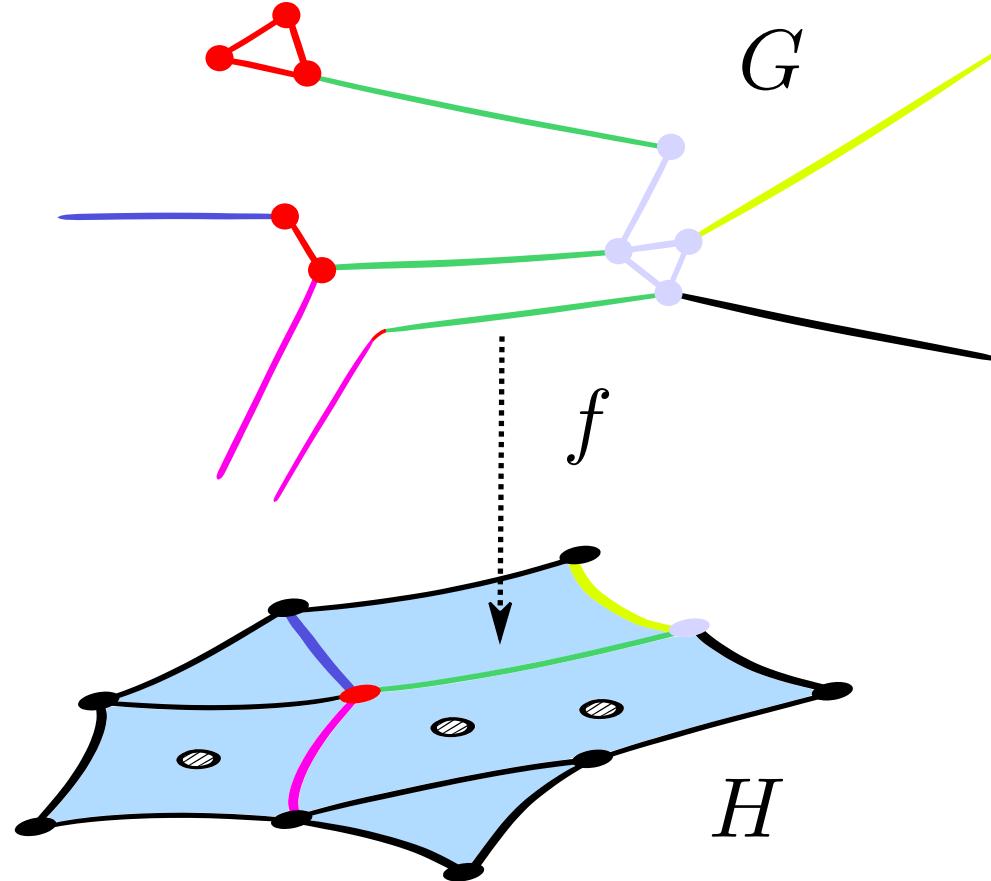
that encodes the surface . . .



. . . and a graph  $H$  on it

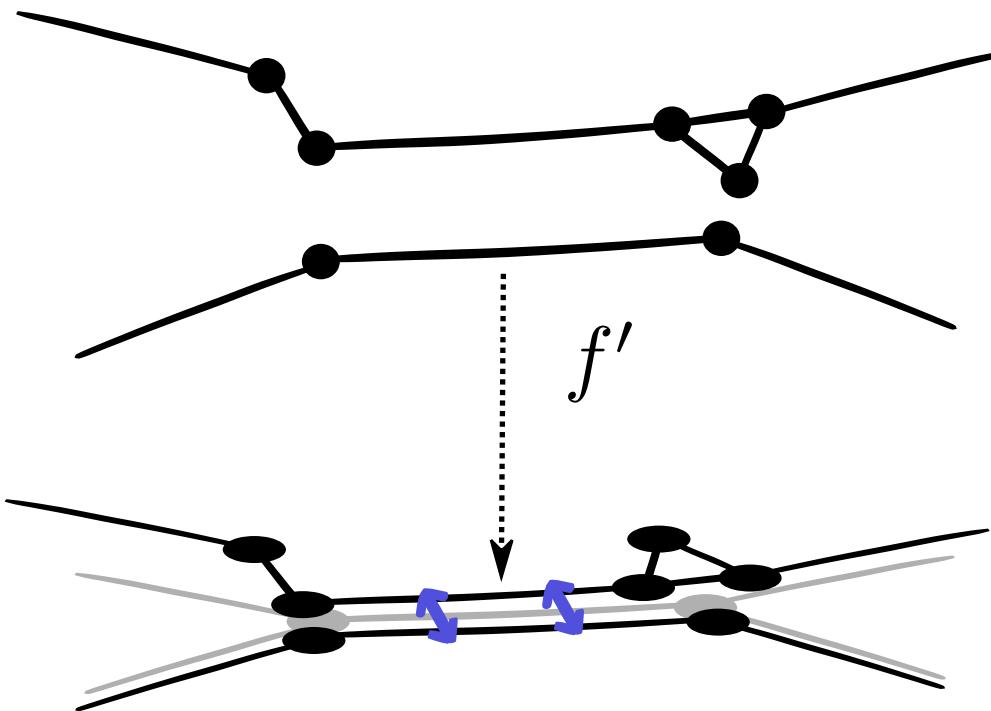
# Input

draw  $G$  in  $H$



# Output

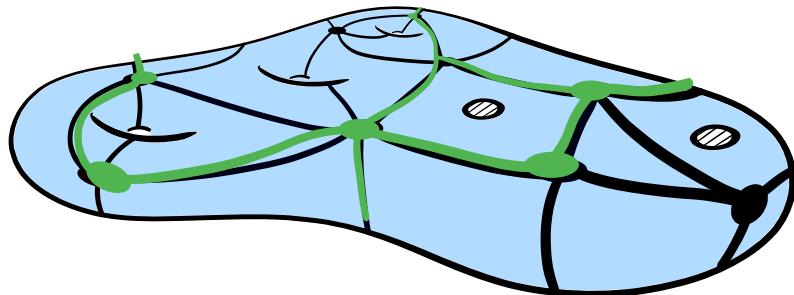
weak embedding: drawing  $f'$  that can be untangled by infinitesimal perturbation



Akitaya, Fulek, and  
Tóth, 2019

algo to determine if  
 $f'$  is weak embedding,  
and if so to perturb  $f'$

# Result



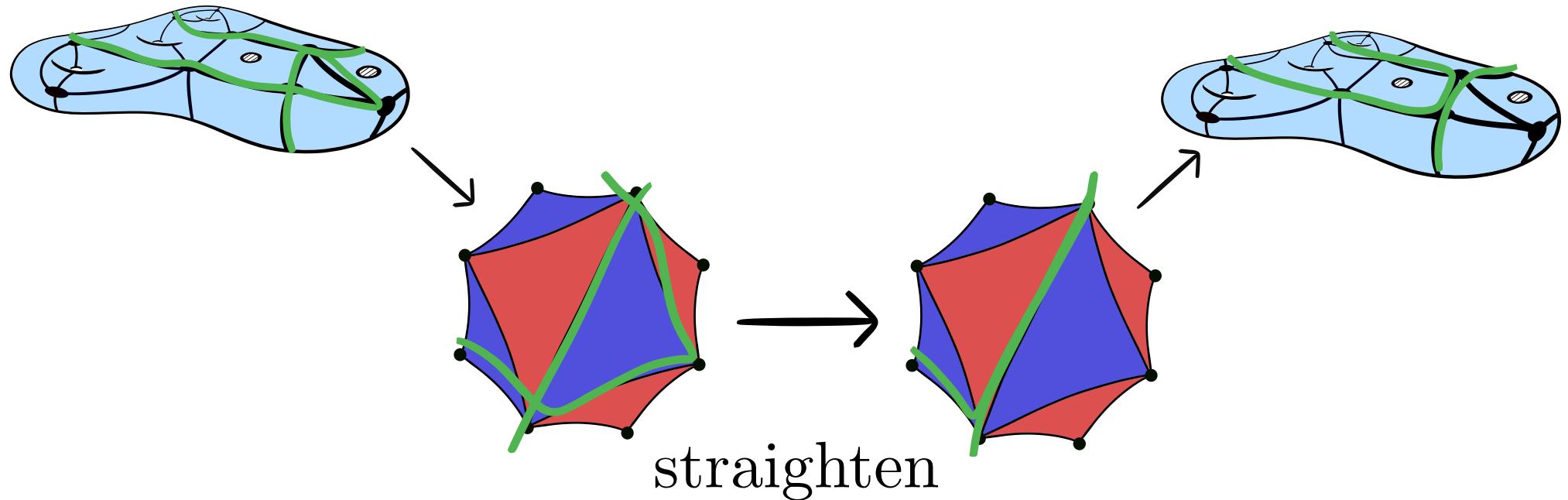
$n$  : # times  $f$  uses an edge  
or vertex of  $H$   
 $m$  : # vertices and edges of  $H$   
 $s$  : genus + # holes

Colin de Verdière, Despré, D., 2023

We can decide if  $f$  can be untangled,  
in  $O(m + s^2 n \log(s n))$  time.

If so, we can compute a weak embedding homotopic  
to  $f$  in additional  $O(s^2 m n^2)$  time.

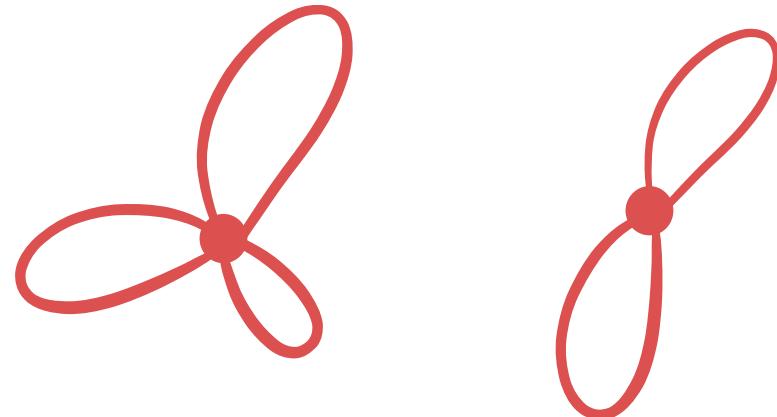
# Algorithm overview



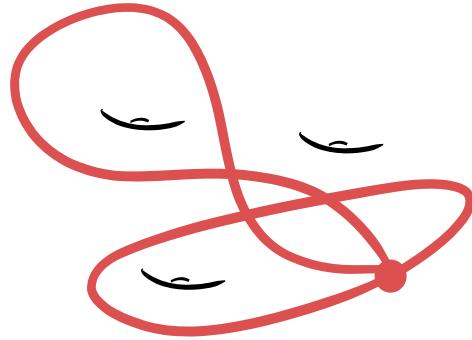
# Straightening a graph

*loop graph*

# Straightening a ~~graph~~

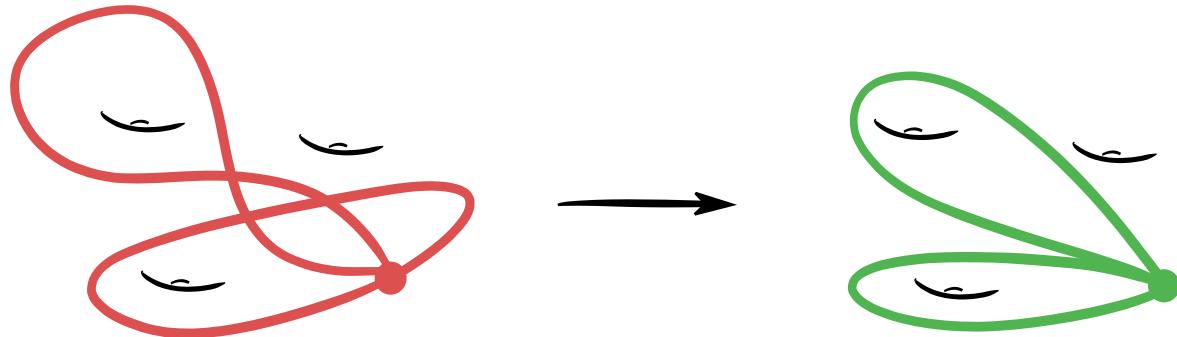


# Straightening a loop graph



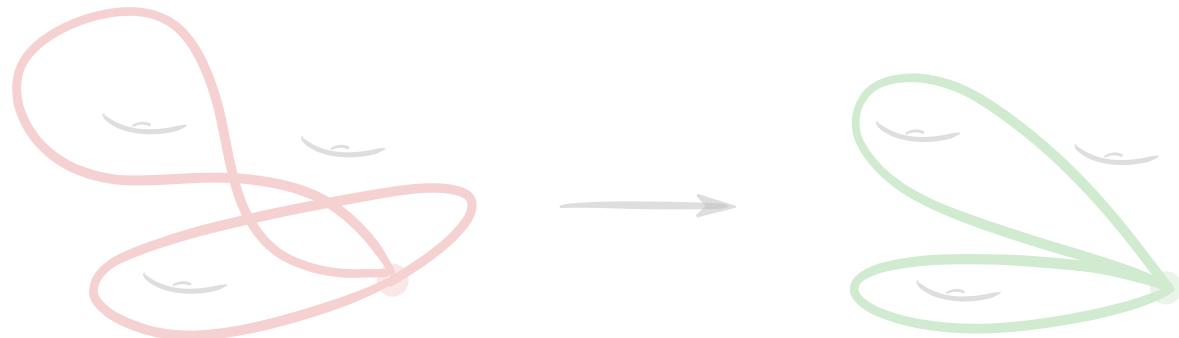
# Straightening a loop graph

First attempt: reduce the loops, vertex fixed

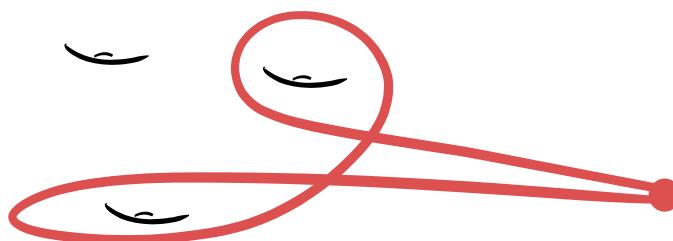


# Straightening a loop graph

First attempt: reduce the loops, vertex fixed



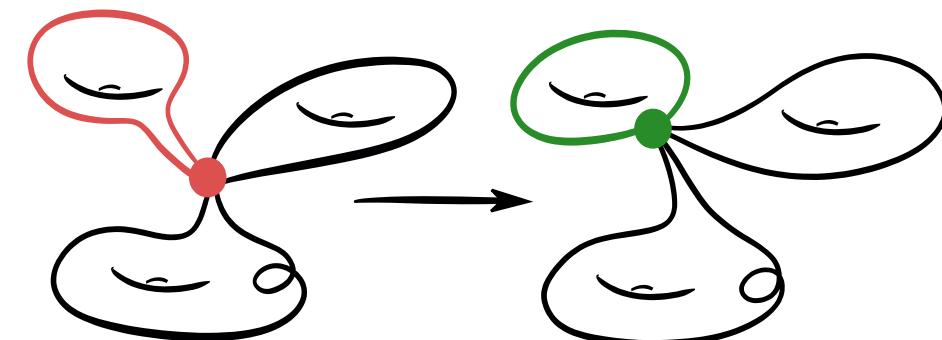
Problem:



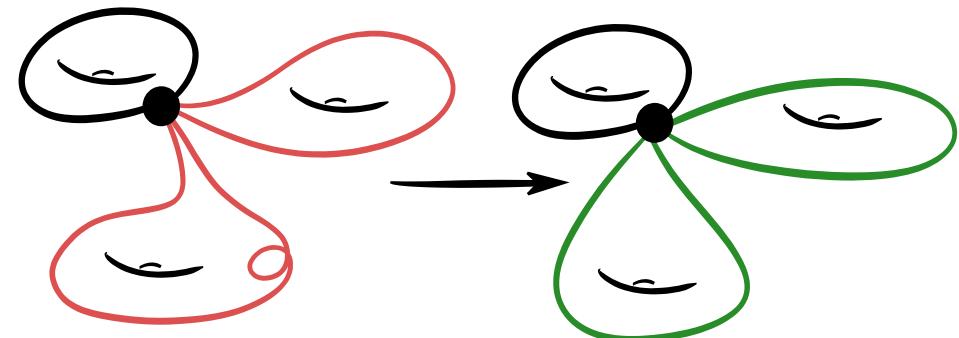
→ vertex must move

# Straightening a loop graph

Solution:



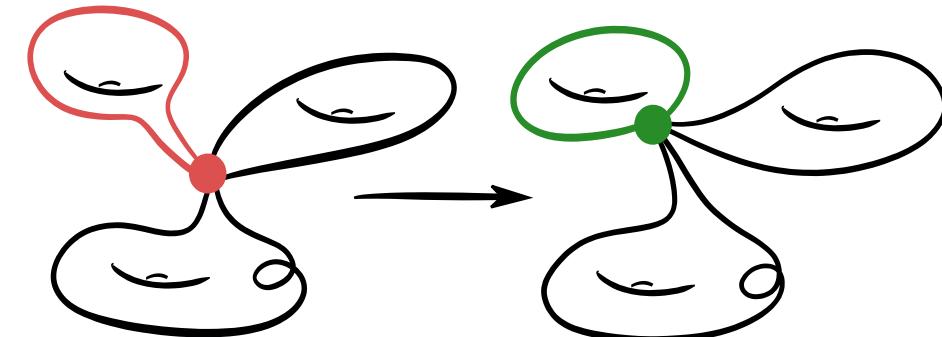
1. Reduce 1 loop cyclically



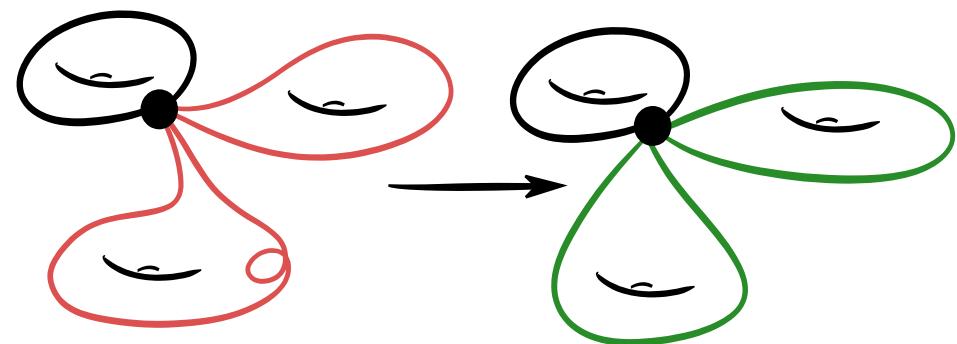
2. Reduce the other loops linearly

# Straightening a loop graph

A straightened loop graph is a weak embedding  
or cannot be untangled



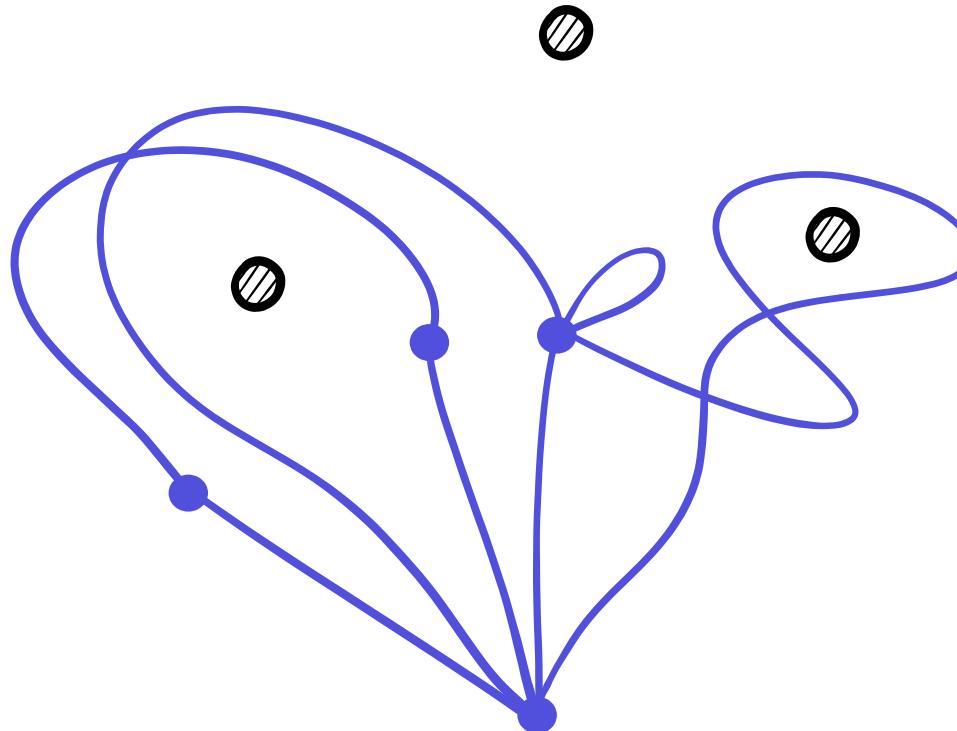
1. Reduce 1 loop cyclically



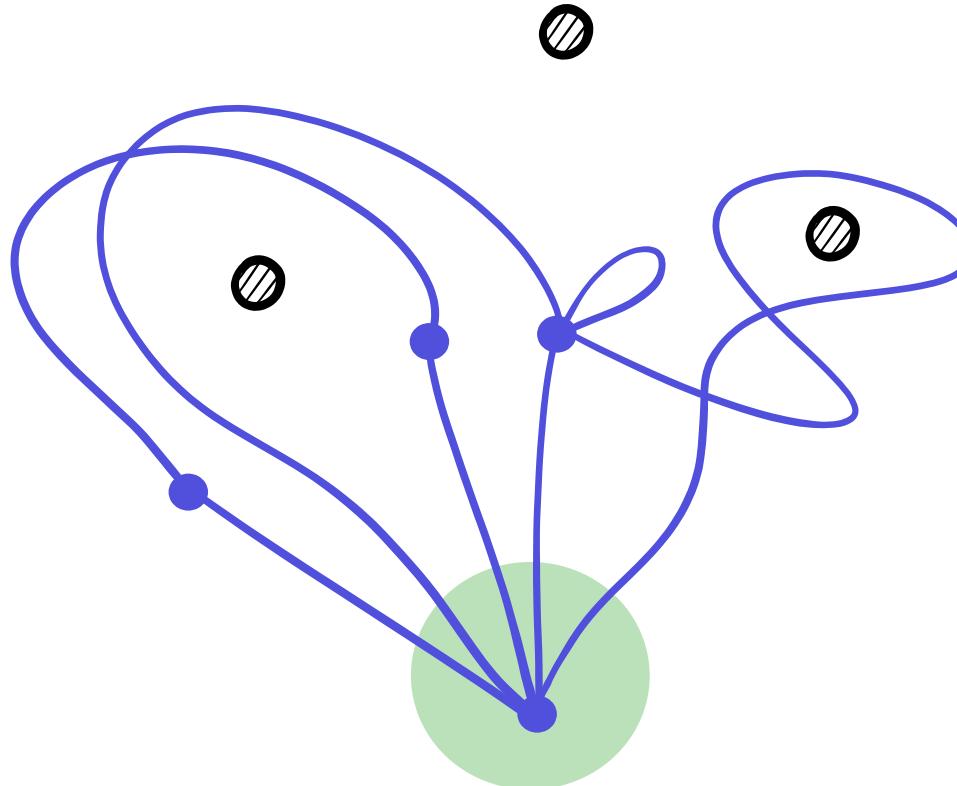
2. Reduce the other loops linearly

# Straightening a graph

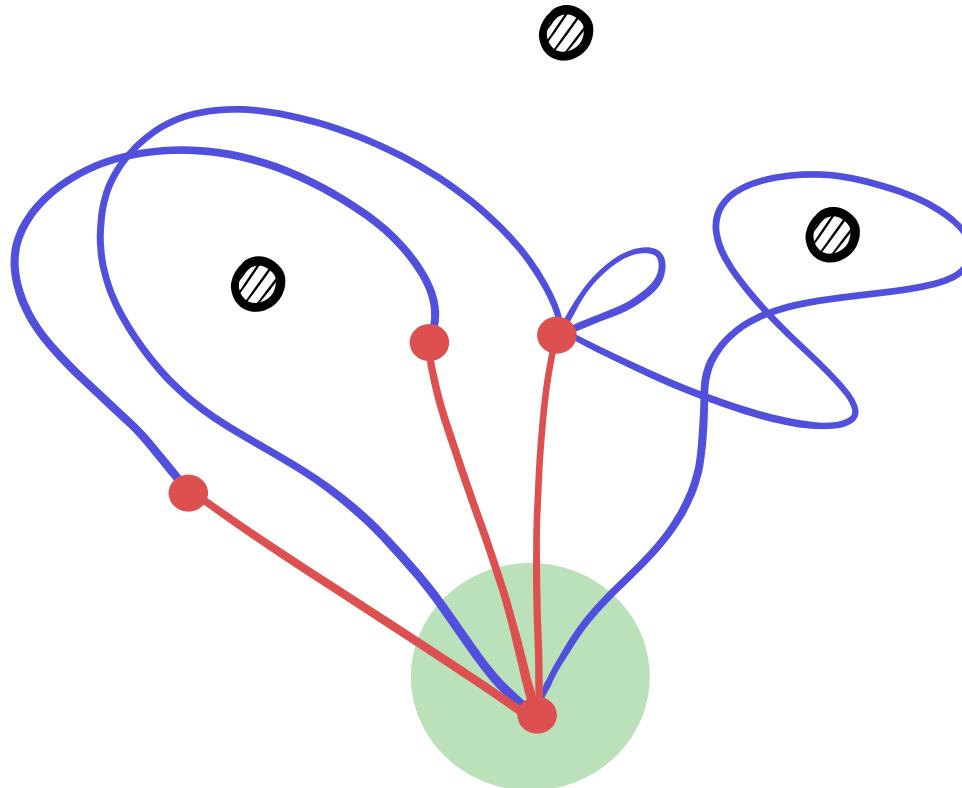
# Straightening a graph



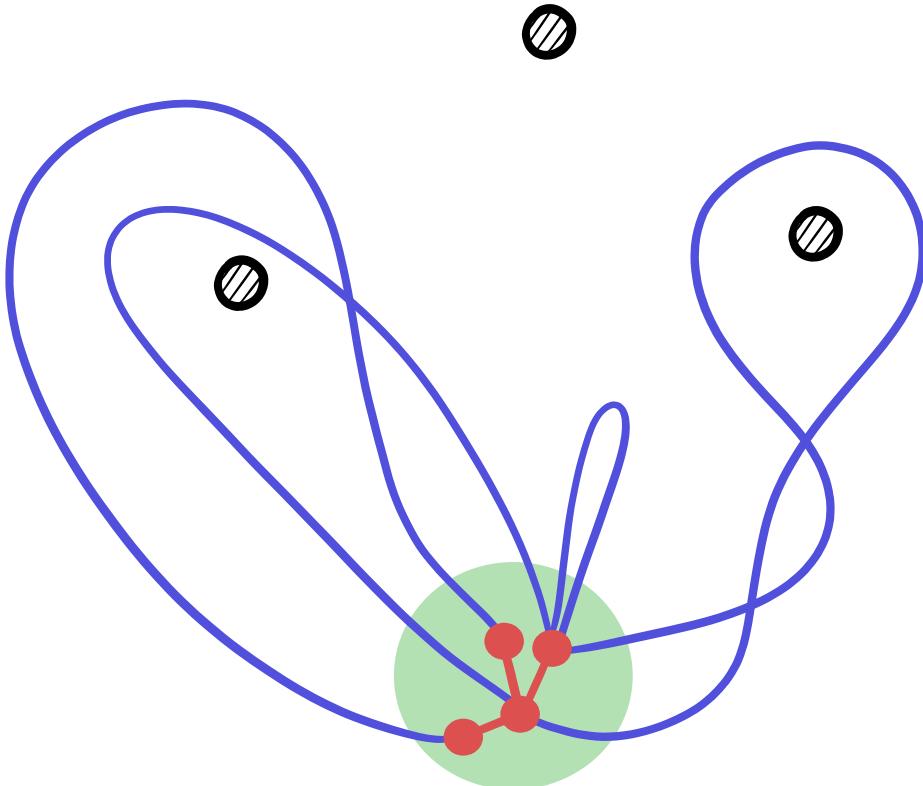
# Straightening a graph



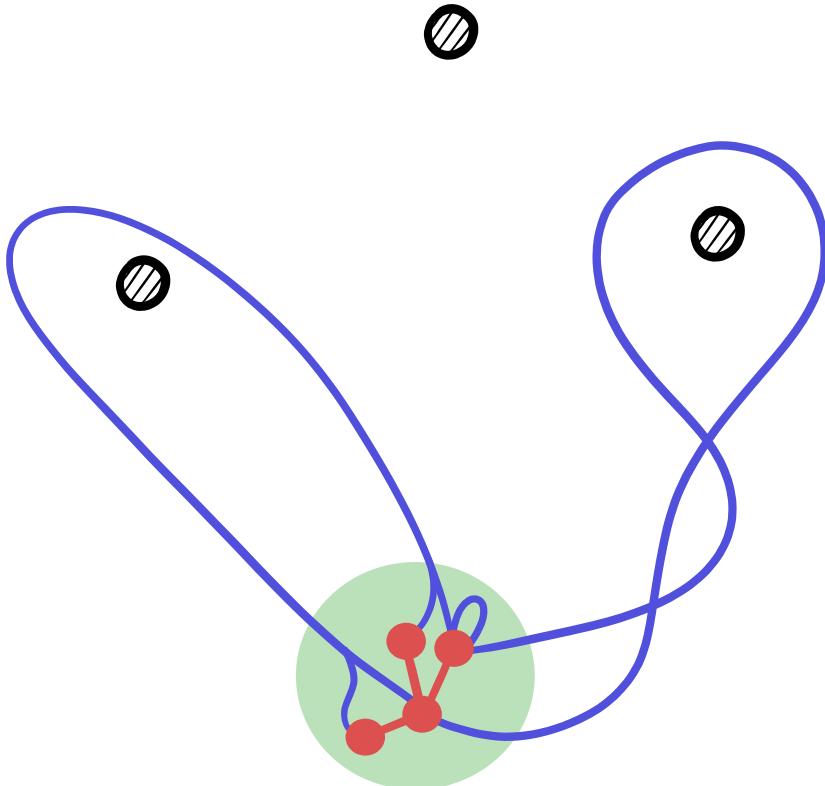
# Straightening a graph



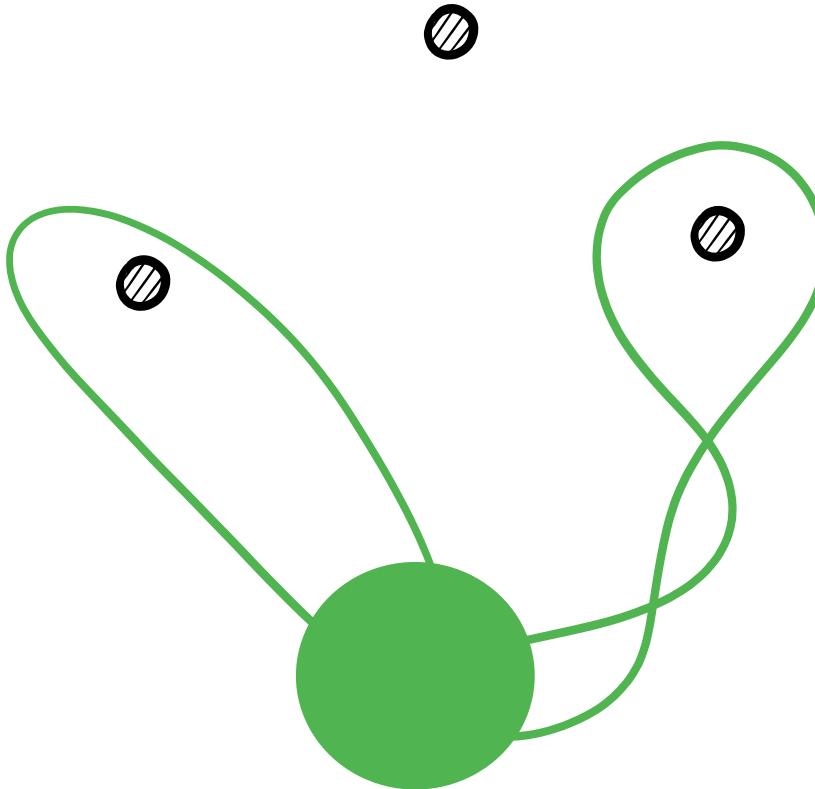
# Straightening a graph



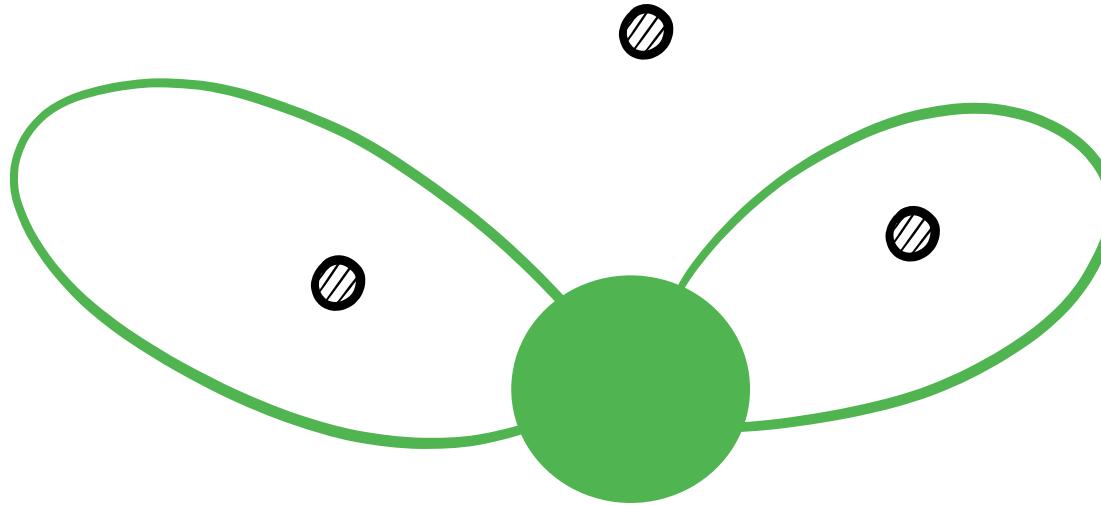
# Straightening a graph



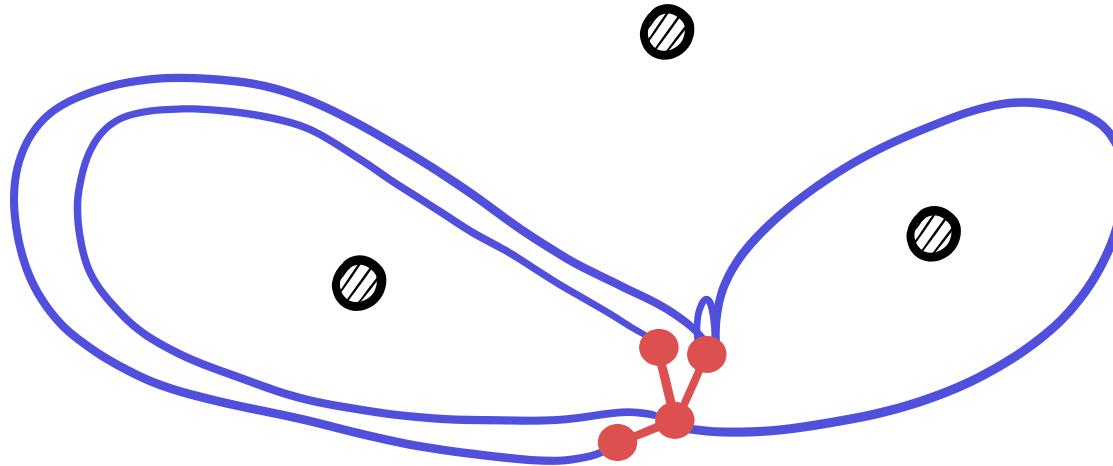
# Straightening a graph



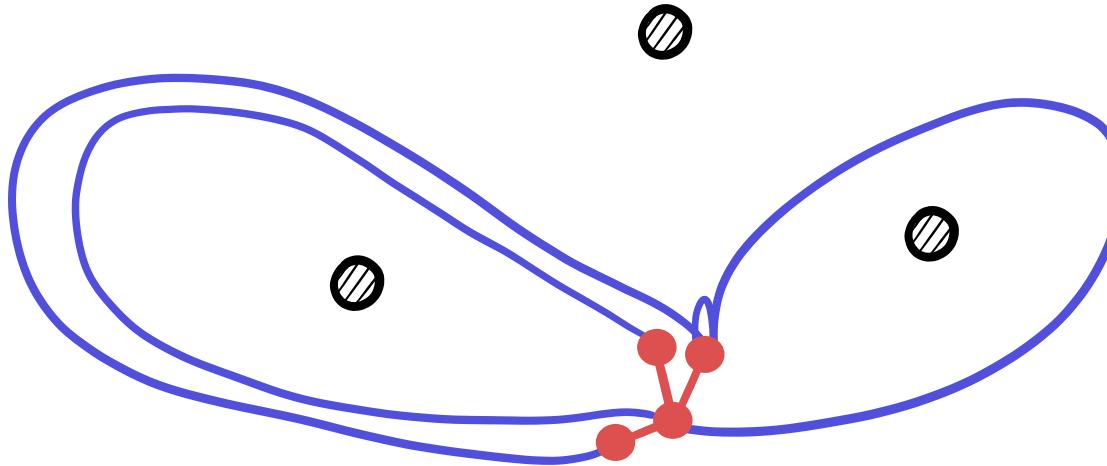
# Straightening a graph



# Straightening a graph



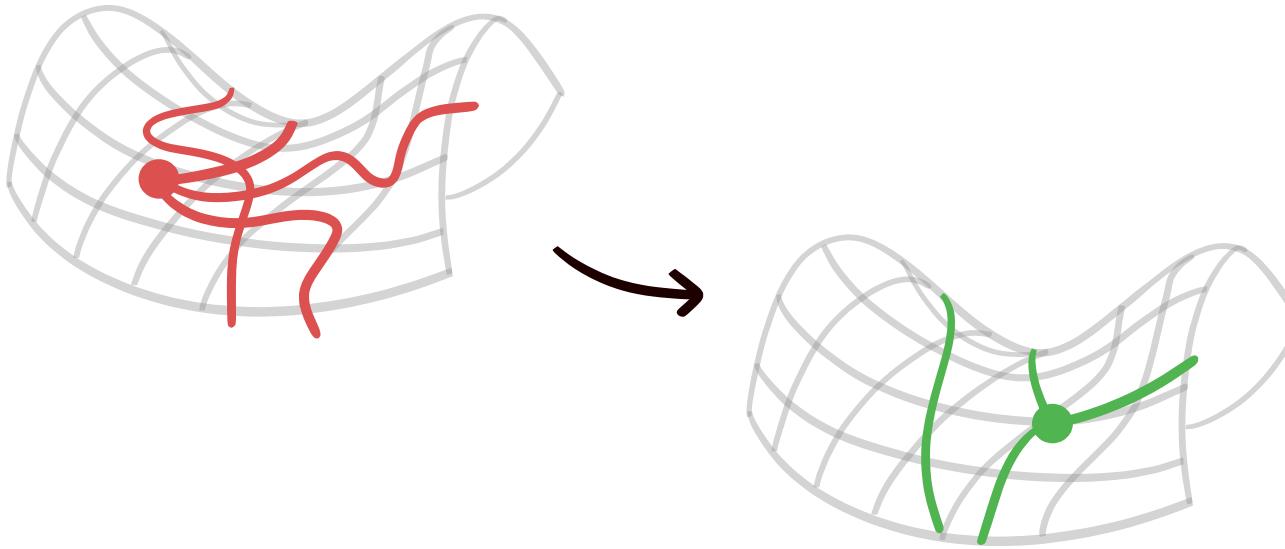
# Straightening a graph



A straightened graph is a weak embedding  
or cannot be untangled

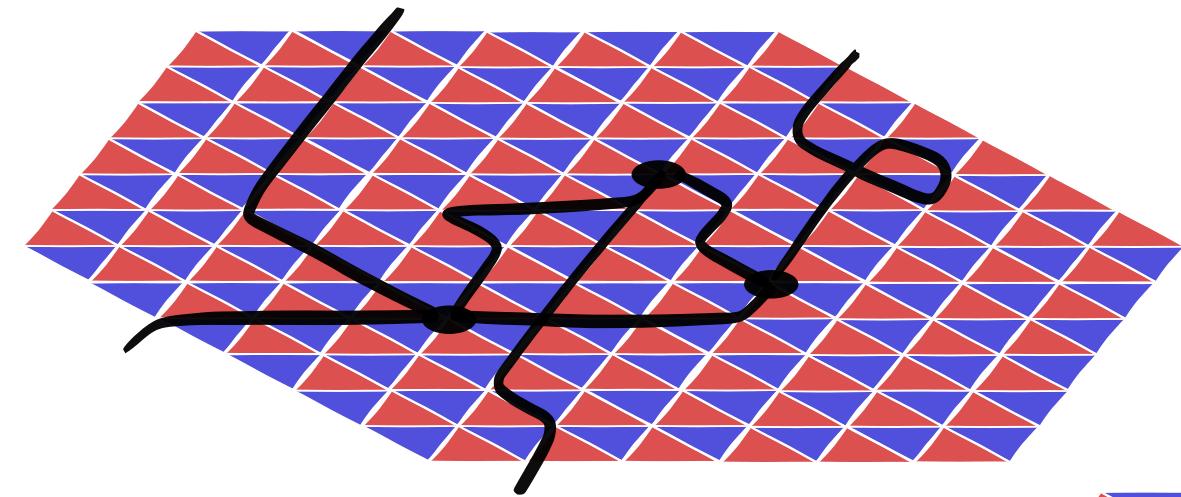
Discrete analogue of  
Tutte embeddings

# Recall: Tutte embeddings

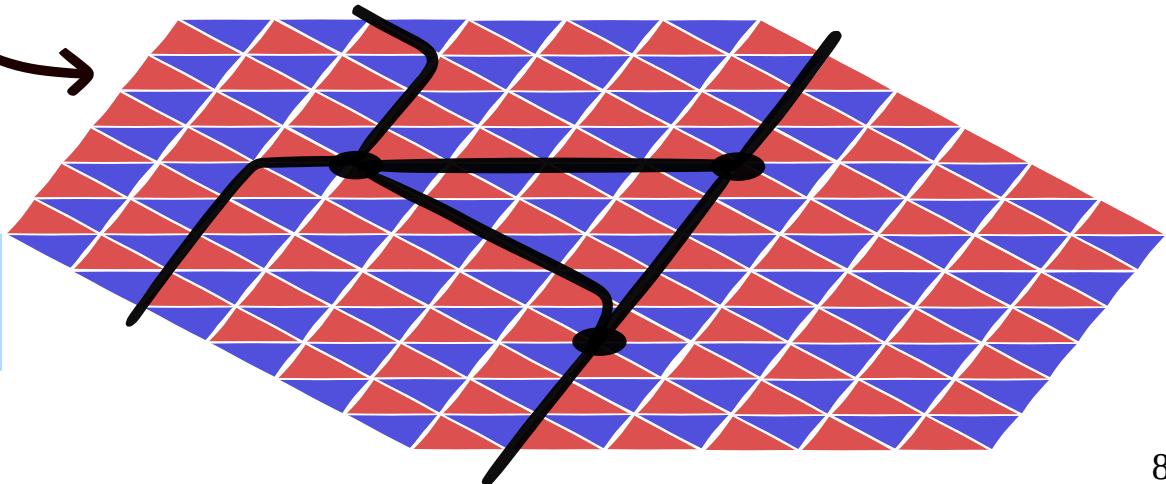


edges straight and  
vertices “barycentric”

# Harmonious drawings

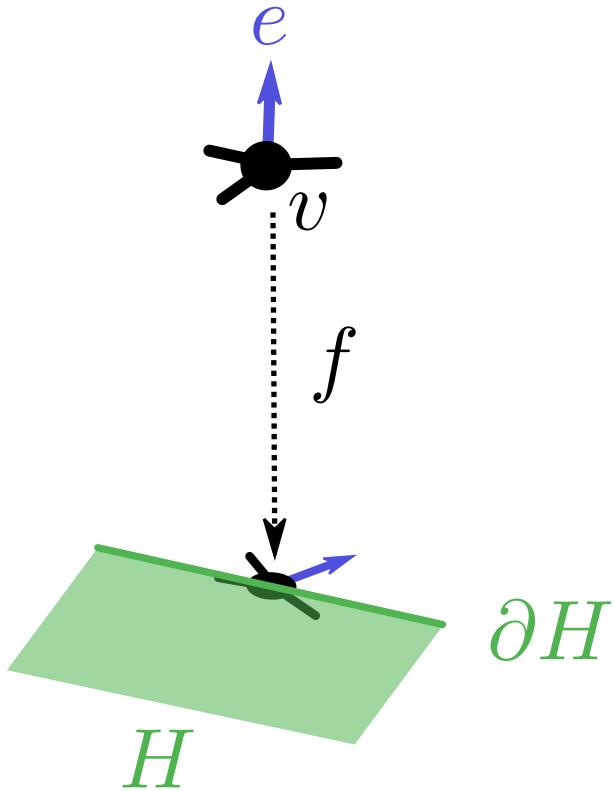


edges reduced and  
vertices “barycentric”



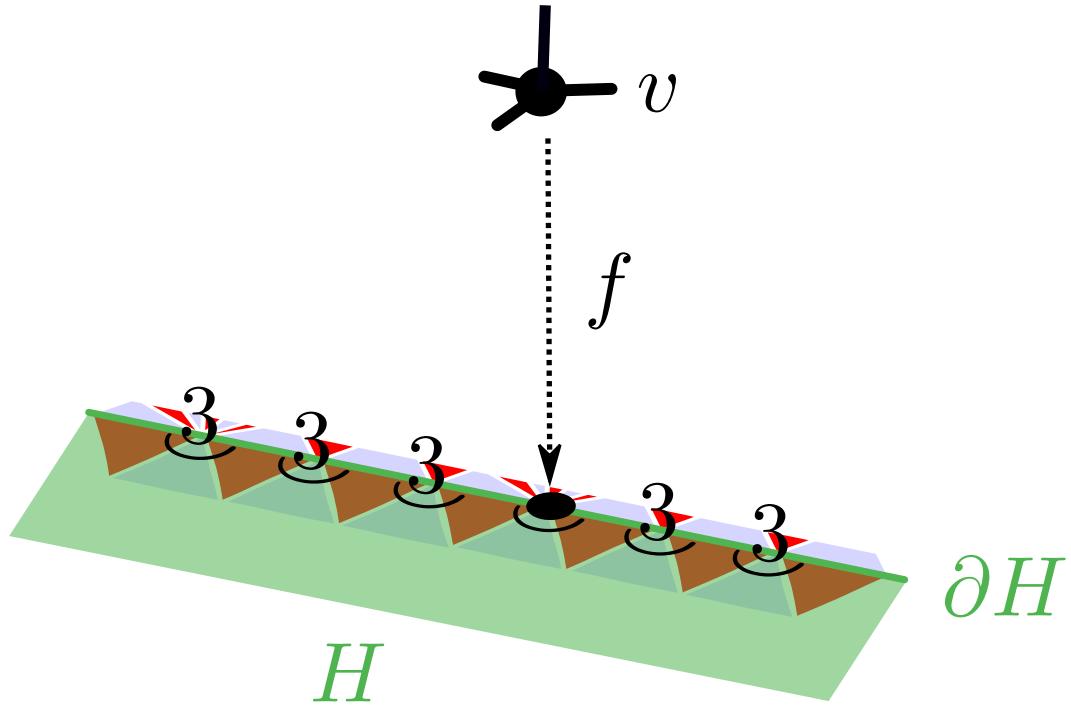
vertices “barycentric”

$$\begin{aligned} f(v) \in \partial H \\ \implies \exists e \quad f(e) \text{ escapes } H \end{aligned}$$



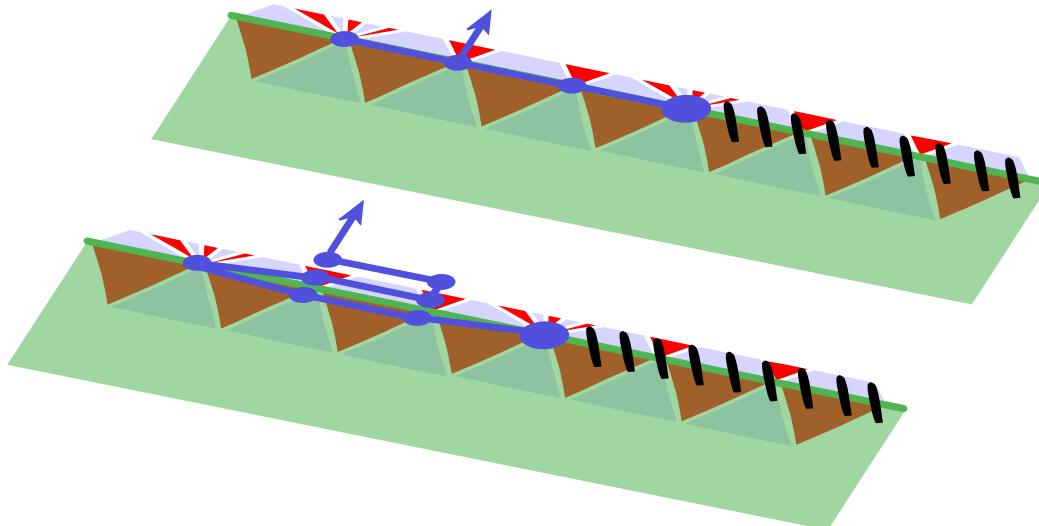
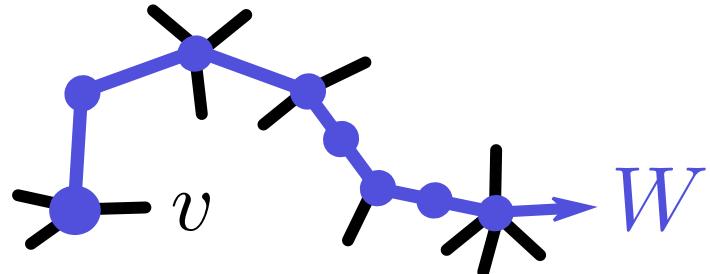
vertices “barycentric”

$$f(v) \in \partial H$$

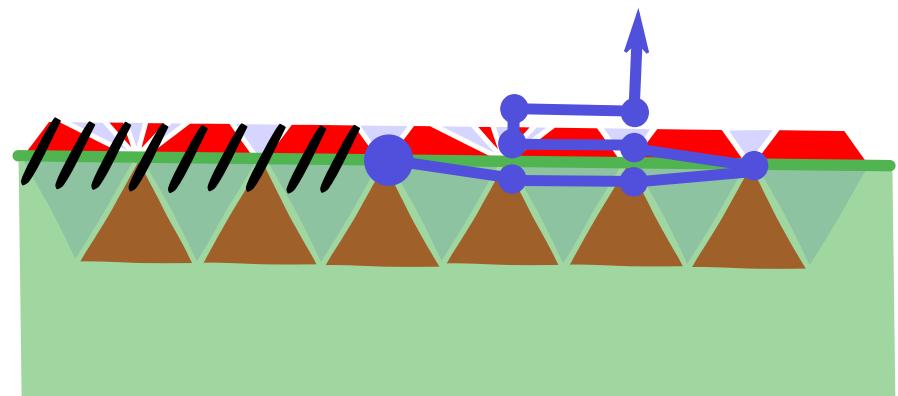
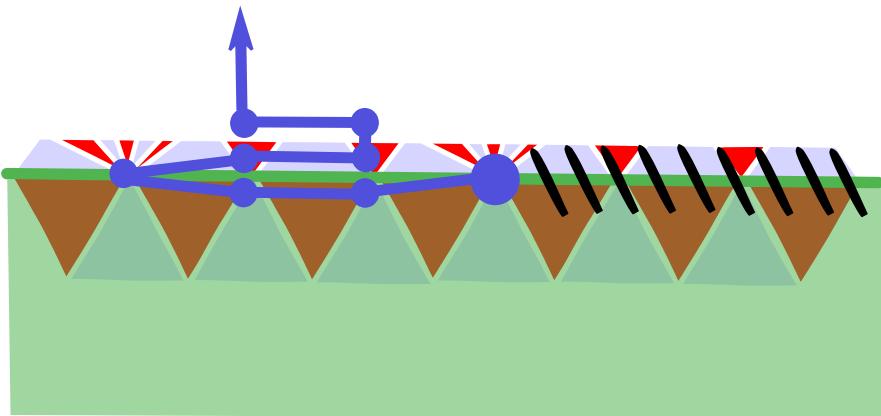


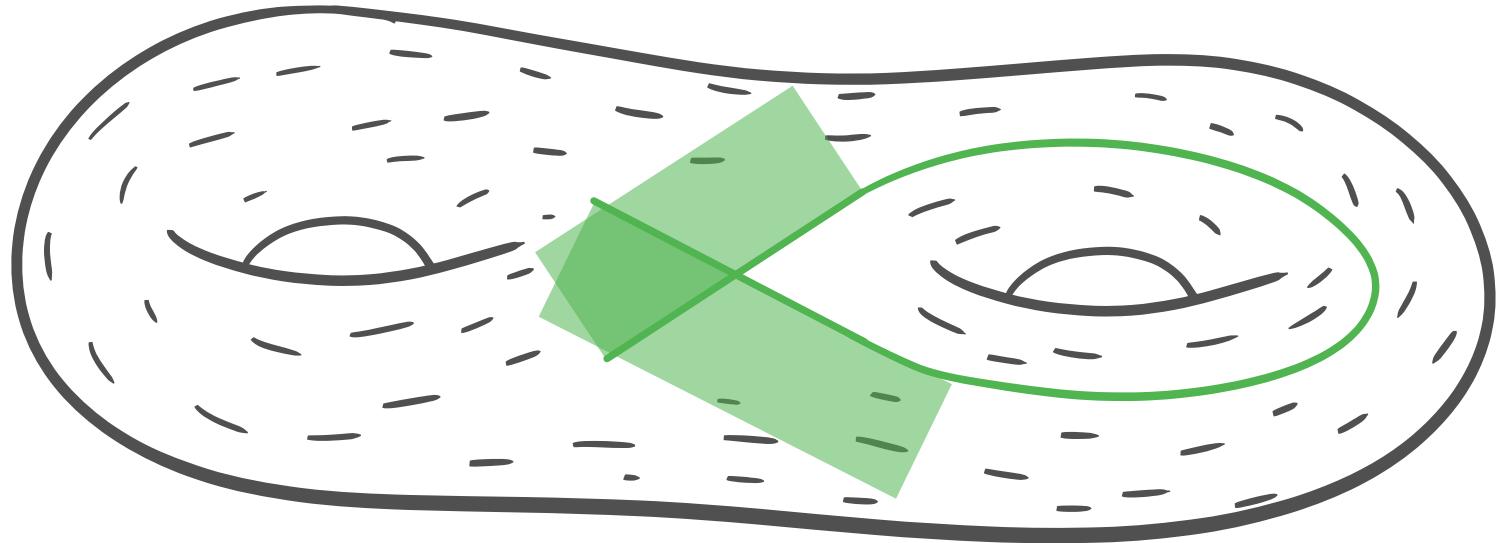
vertices “barycentric”

$$f(v) \in \partial H \\ \xrightarrow{} \\ \exists W f \circ W \text{“escapes”} H$$



where to escape depends on the coloring





this definition generalizes to surfaces

# Results

graph  $G$

reducing triangulation  $T$  with  $m$  edges

$f : G \rightarrow T^1$  of size  $n$

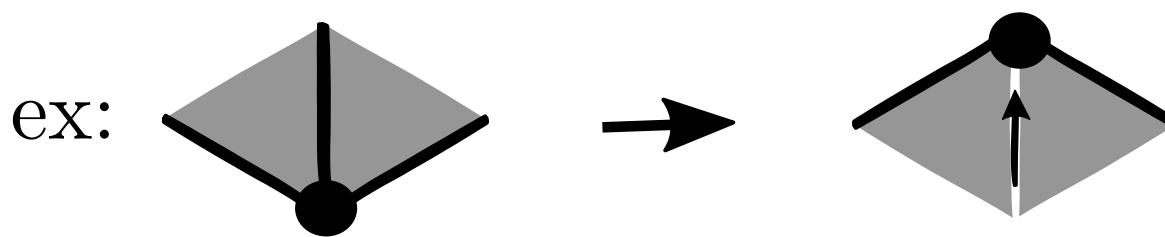
Definition of *harmonious* drawings

$f$  harmonious and  $f$  can be untangled  
 $\Rightarrow f$  weak embedding

Algo to make  $f$  harmonious in  $O((m+n)^2 n^2)$  time,  
without increasing any edge length

# Harmonizing a drawing

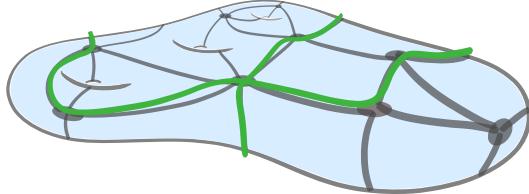
- 1 we define monotonic moves to apply to  $f$



- 2 some moves do not seem to decrease any potential so we combine the moves carefully

# Untangling curves

# Related work

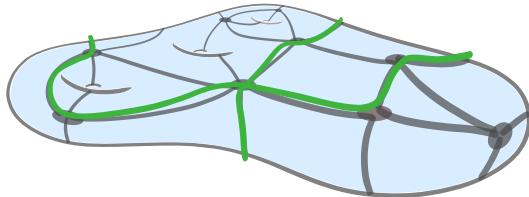


closed walks of total length  $n$   
on a graph of size  $m$

Despré, Lazarus, 2019

- Put a single curve in minimal position in  $O(m + n^4)$  time
- Compute the min. nb. of crossings in  $O(m + n^2)$  time

# Result



closed walks of total length  $n$   
on a graph of size  $m$

simpler algos and proofs!

$$m^3n + mn \log(mn)$$

D., 2024

- Put a single curve in minimal position in  $O(\cancel{m} + \cancel{n}^4)$  time
- Compute the min. nb. of crossings in  $O(\cancel{m} + \cancel{n}^2)$  time

$$m^2 + mn \log(mn)$$

# Untangling Graphs

Computing Delaunay Triangulations

Other Results  
and Possible Continuations

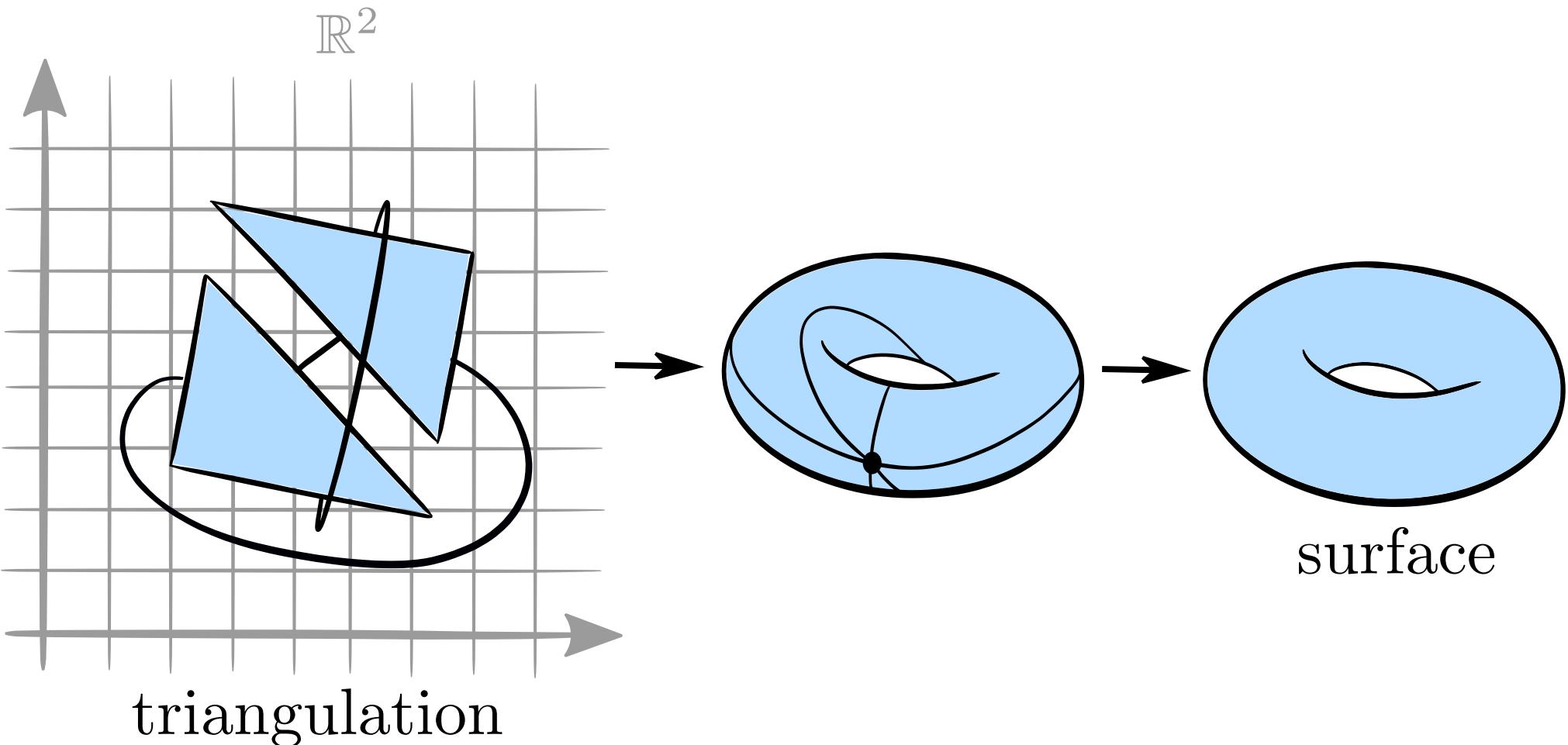
# Untangling Graphs

Computing Delaunay Triangulations

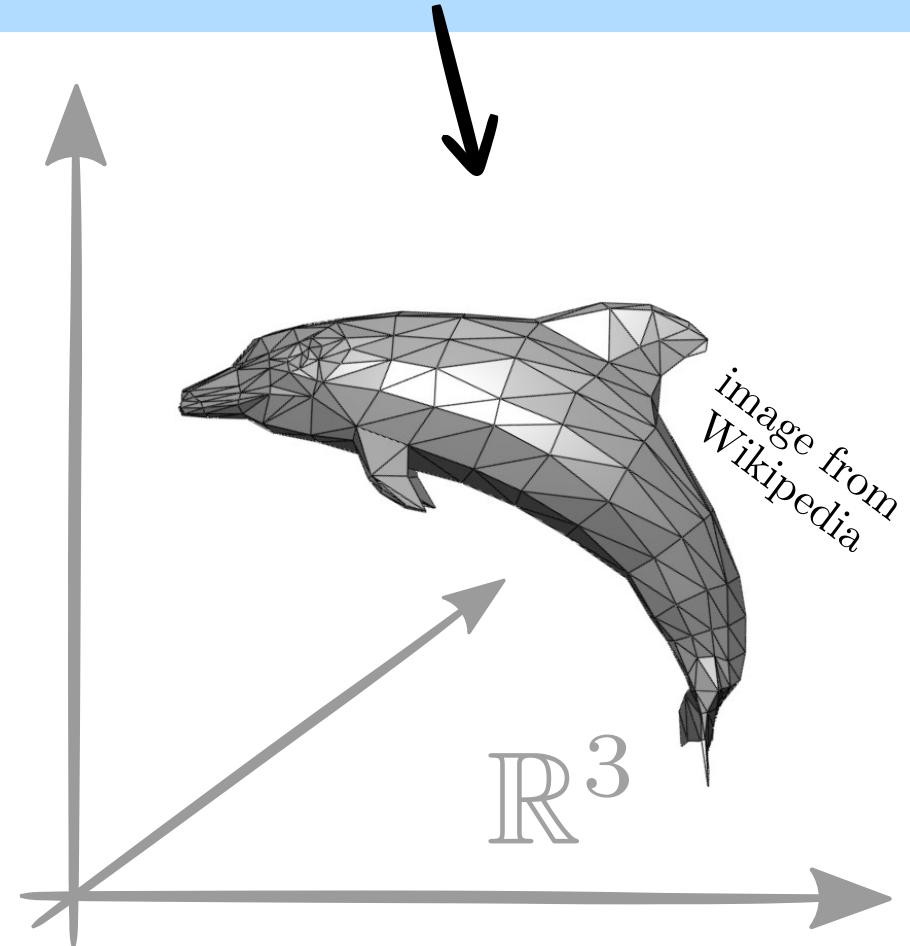
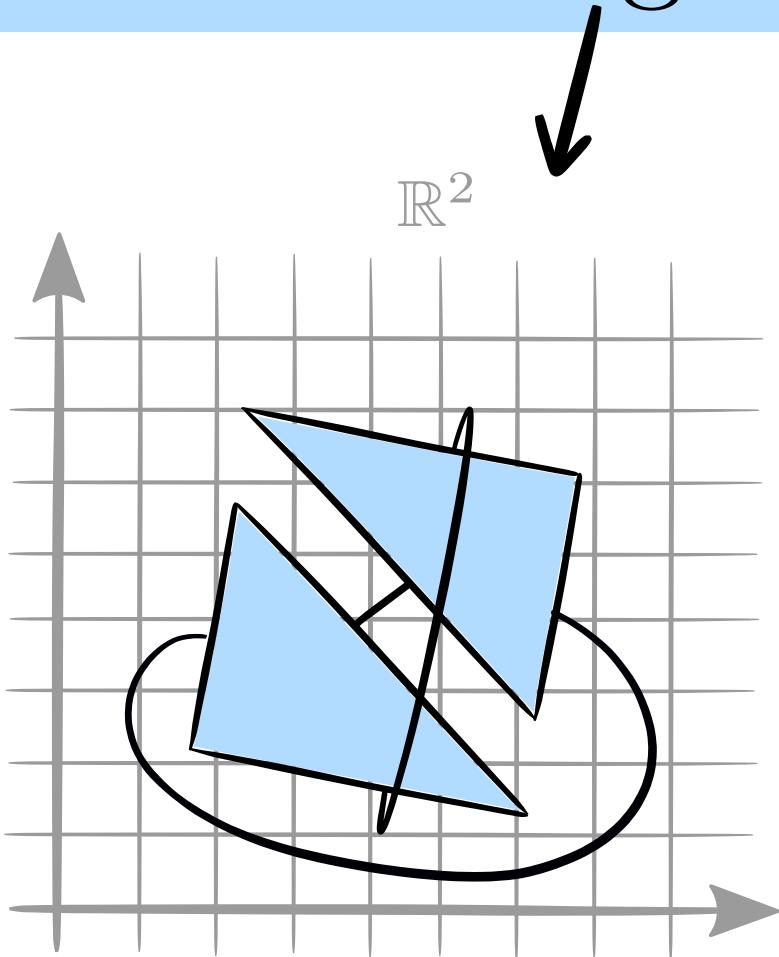
Other Results  
and Possible Continuations

# Triangulations of polyhedral surfaces

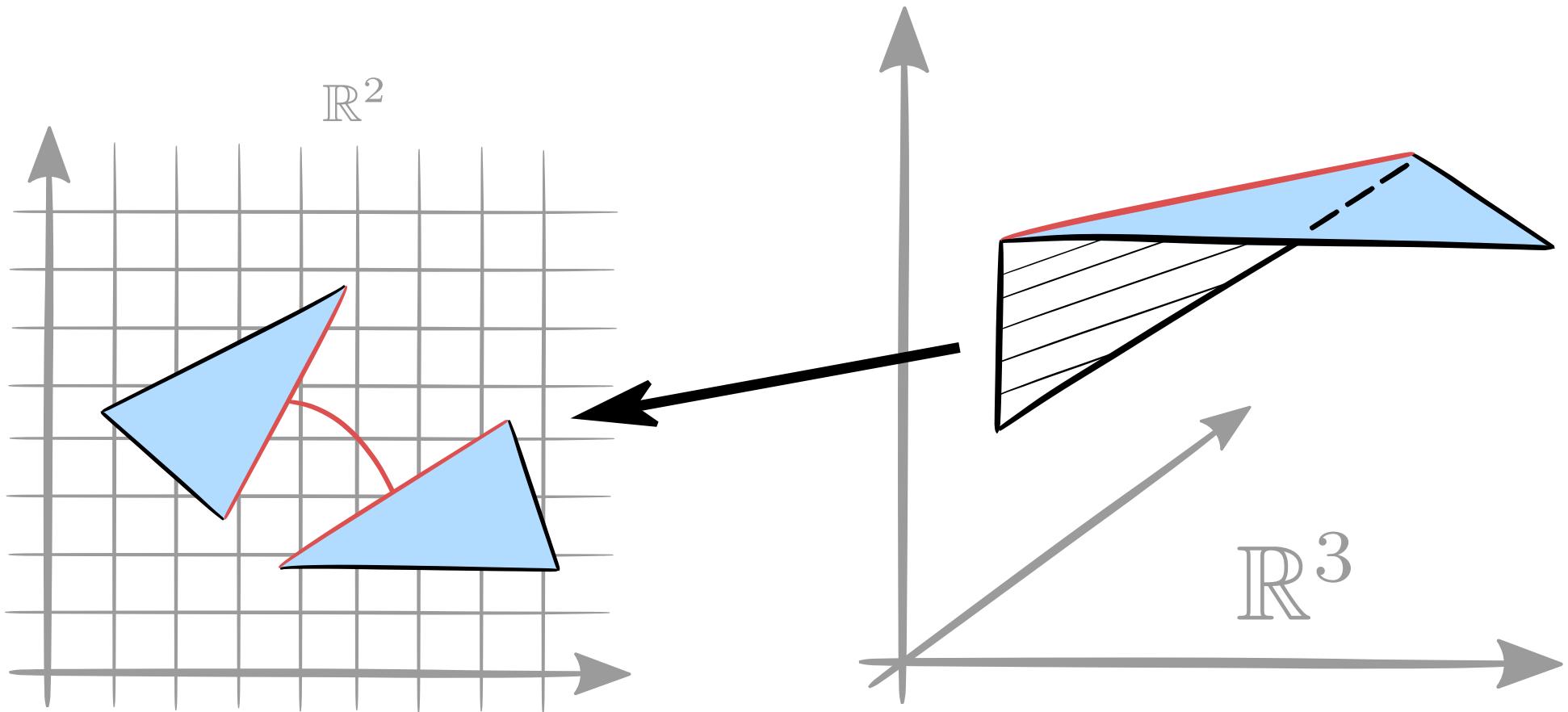
# Triangulation of polyhedral surface



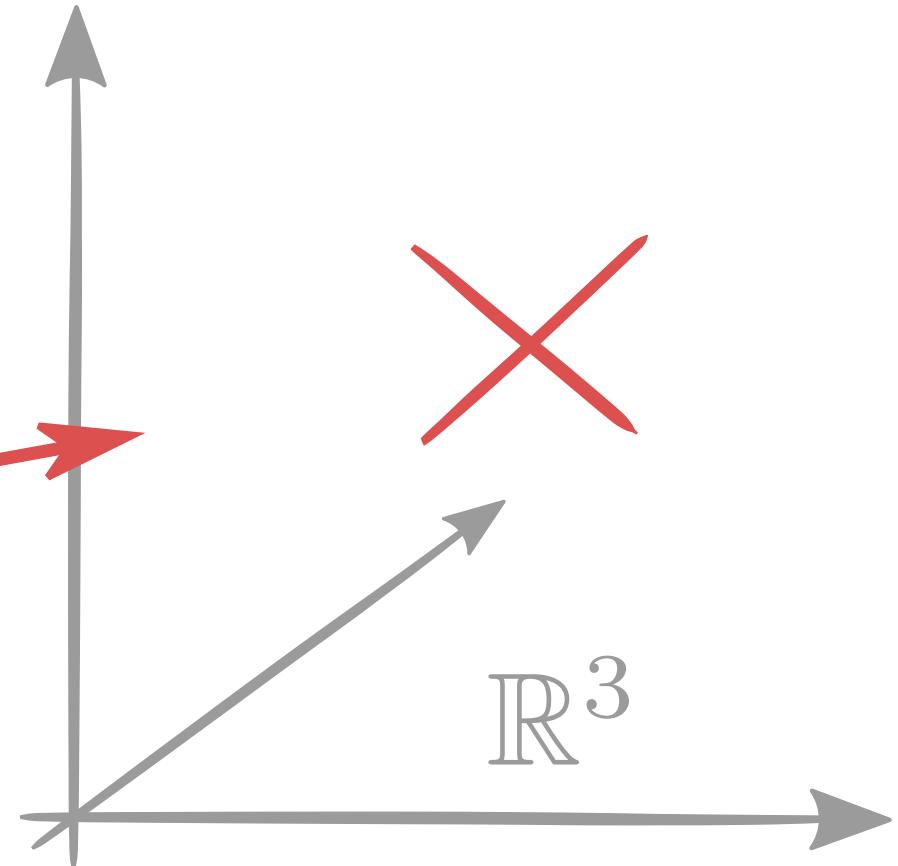
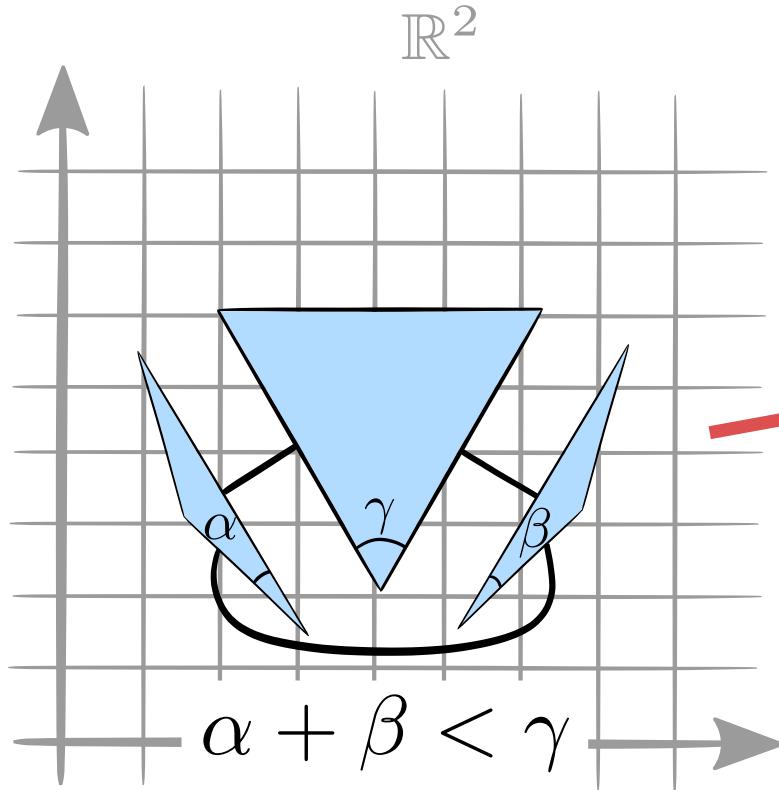
# Triangulation vs. Mesh



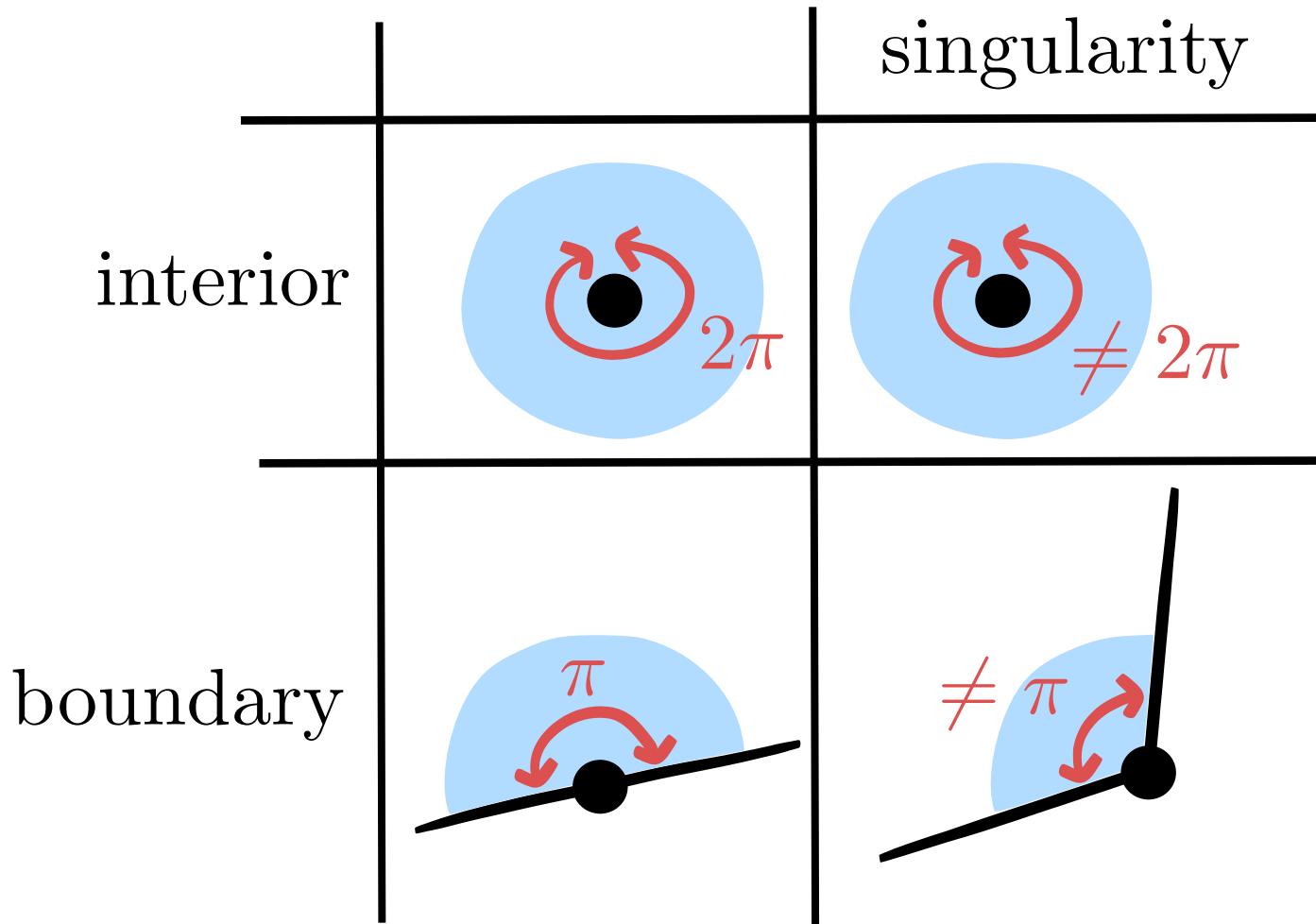
Every mesh gives a triangulation



Every mesh gives a triangulation  
but converse is false!



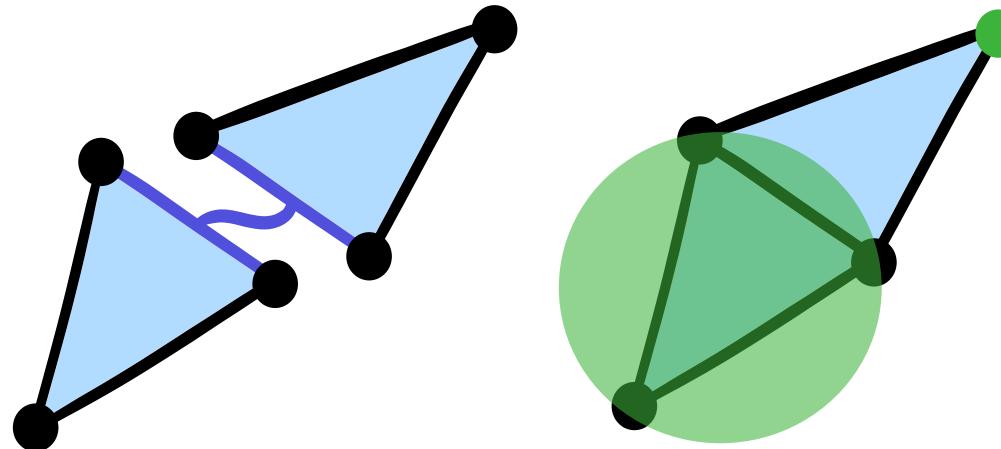
# Types of points on the surface



# Problem

# Delaunay triangulation

triangulation in which  
every edge is Delaunay



# The Delaunay triangulation

Generically, every surface has a **unique**  
Delaunay triangulation  
with given vertex set\*

\*finite, non-empty, containing the singularities

# Problem

Given triangulation  $T$ , and vertex set  $V^*$ ,  
compute the Delaunay triangulation of the  
surface of  $T$  whose vertex set is  $V$

$*$ usually the vertex set of  $T$  or  
the singularities of the surface of  $T$

# Motivations

- isometry testing
- shortest paths

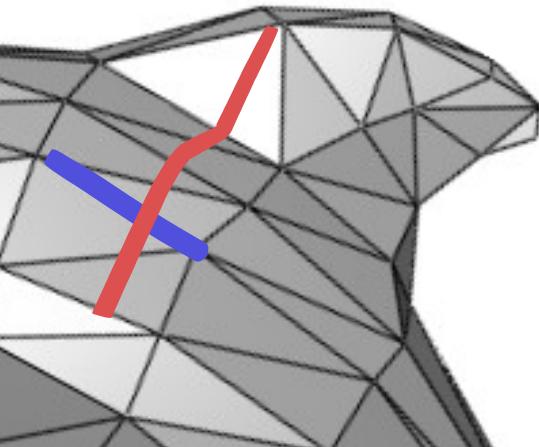
- shortest paths

# Shortest paths on meshes

On a mesh  $M$  with  $n$  triangles...

a shortest path cannot cross an edge twice

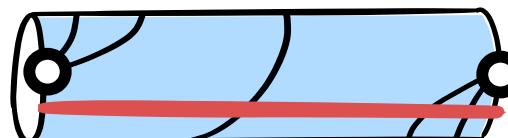
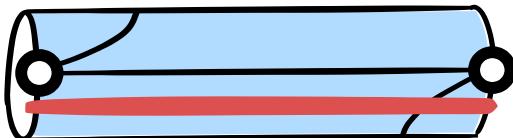
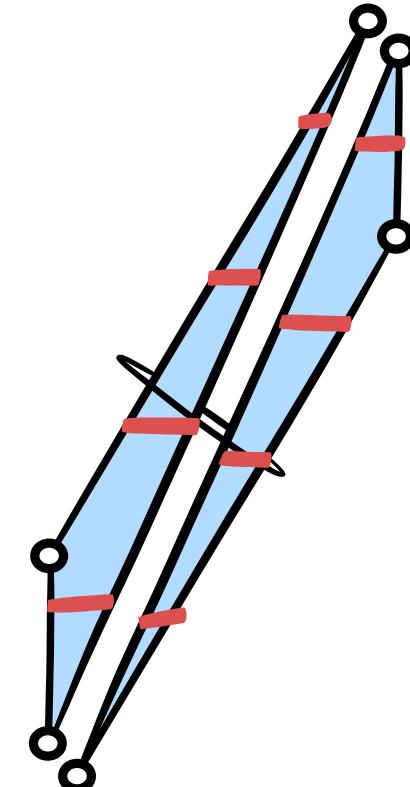
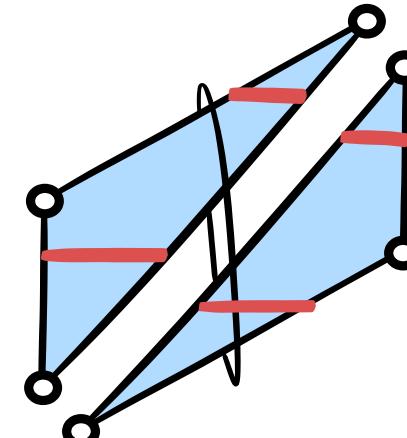
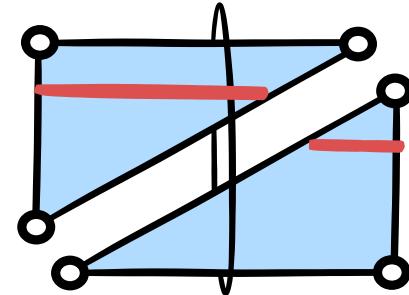
→ shortest path can be computed in  $O(f(n))$  time



Mitchel, Mount, Papadimitriou, 1987

# Shortest paths on triangulations

they can cross edges arbitrarily many times

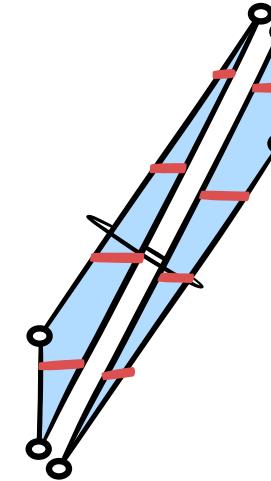


Erickson, 2006

# Shortest paths on triangulations

Löffler, Ophelders, Staals, Silveira, 2023

happiness  $h$ : max number of times  
a shortest path visits a triangle



→ shortest path can be computed in  $O(f(n, h))$  time

# Shortest paths on triangulations

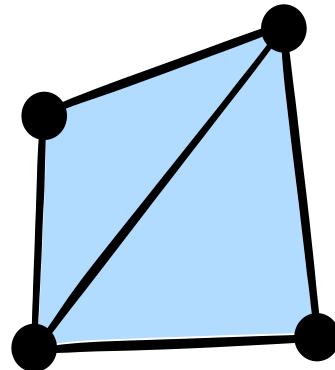
Löffler, Ophelders, Staals, Silveira, 2023

Delaunay triangulations have happiness  $O(1)$

# Existing algorithms

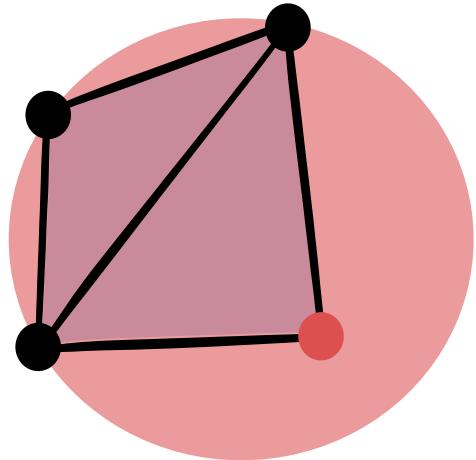
# Delaunay flips algorithm

Delaunay flip:



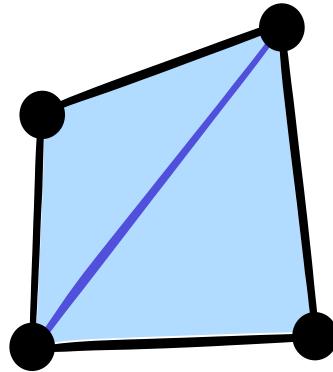
# Delaunay flips algorithm

Delaunay flip:



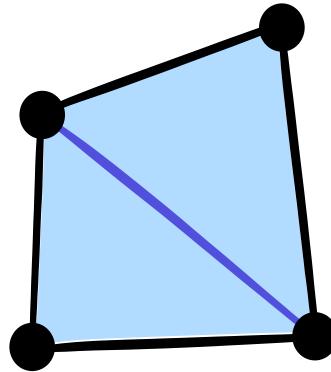
# Delaunay flips algorithm

Delaunay flip:



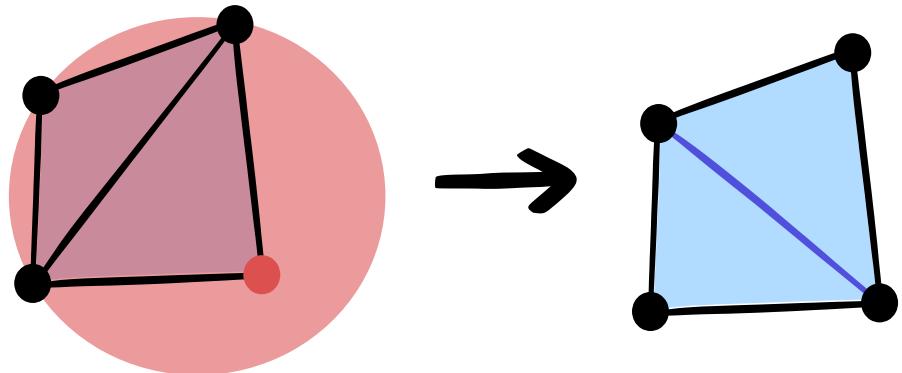
# Delaunay flips algorithm

Delaunay flip:



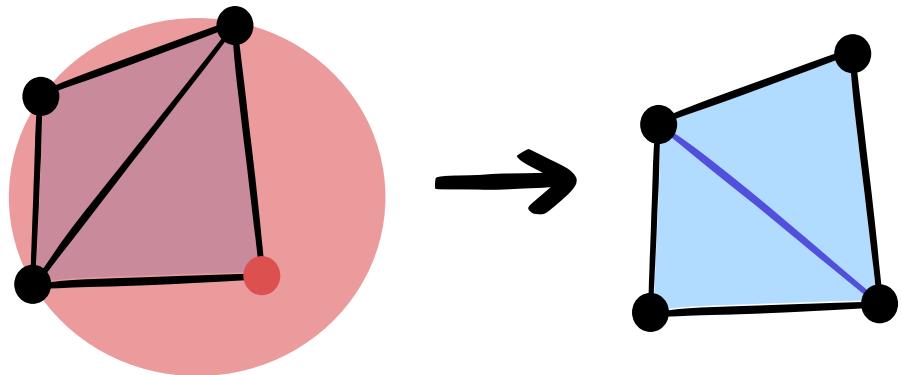
# Delaunay flips algorithm

Delaunay flip:



# Delaunay flips algorithm

Delaunay flip:

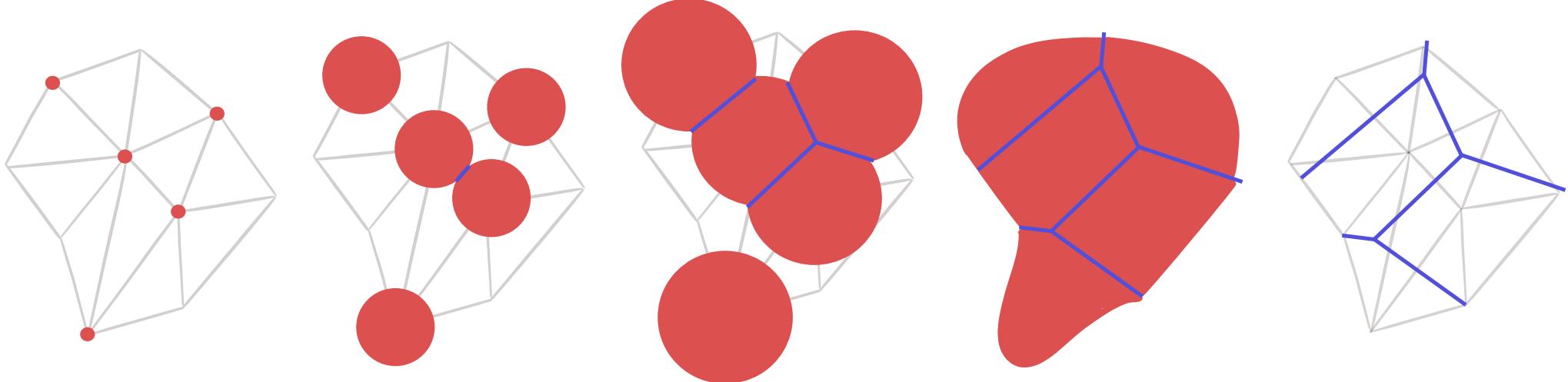


Algorithm:

apply Delaunay flips greedily as long as you can

# Other method

compute the **Voronoi diagram**  
by propagating **waves**

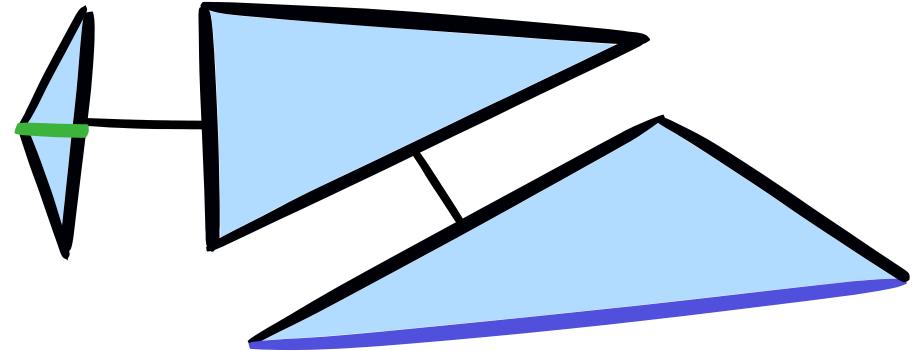


then derive Delaunay from it

# Result

# Result

aspect ratio =  
 $\frac{\text{maximum side length}}{\text{minimum height}}$

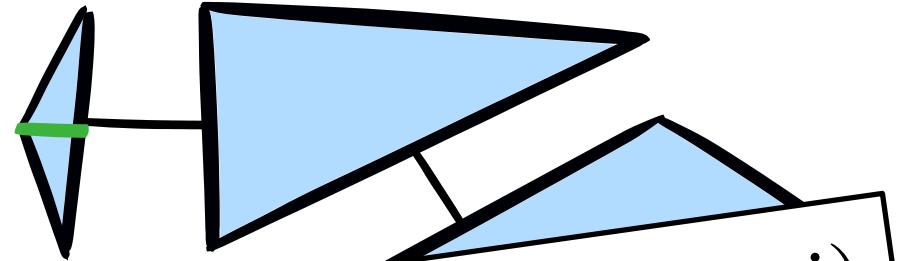


D. 2025

Given triangulation  $T$  of  $n$  triangles, of aspect ratio  $r$ , whose surface has no boundary, we can compute Delaunay in  $O(n^3 \log^2(n) \cdot \log^4(r))$  time

# Result

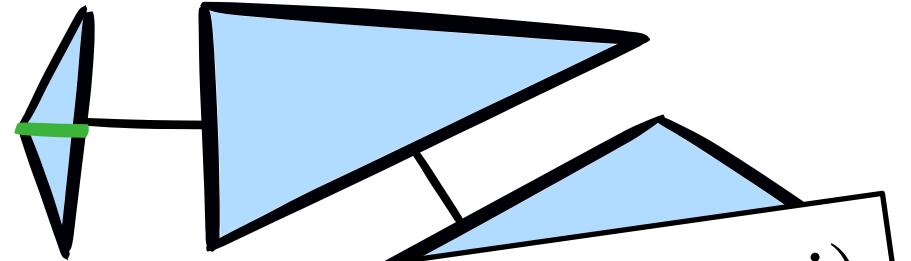
aspect ratio =  
 $\frac{\text{maximum side length}}{\text{minimum height}}$



D. S. G. Previous algorithms (Delaunay flips and Voronoi) achieved  $O(\text{Poly}(n, r))$  time for surfaces with boundaries. If the surface has no boundary, we can compute Delaunay in  $O(n^3 \log^2(n) \cdot \log^4(r))$  time.

# Result

aspect ratio =  
maximum side length  
minimum height



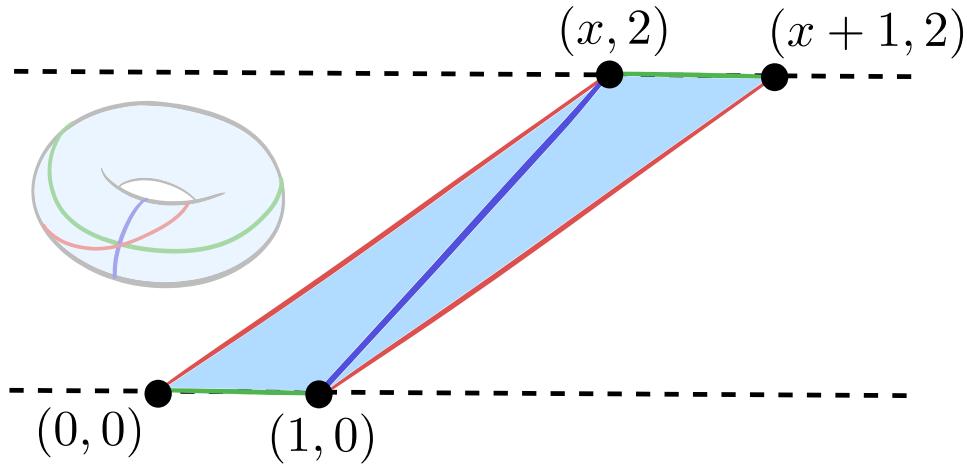
D. So  
G. Previous algorithms (Delaunay flips and Voronoi)  
achieved  $O(\text{Poly}(n, r))$   
surface has no boundary, we can compute  
Delaunay in  $O(n^3 \log^2(n) \cdot \log^4(r))$  time

Now backed by a lower bound!

# Lower bound

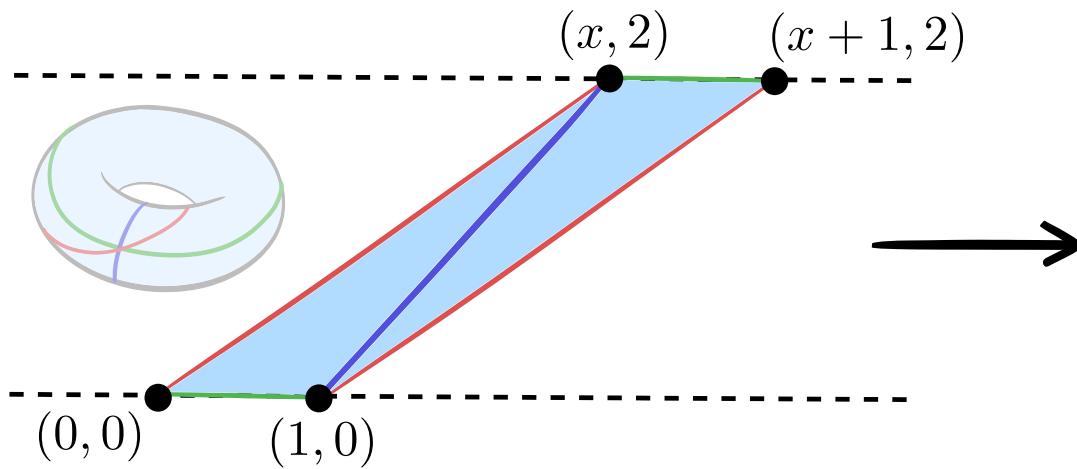
# Lower bound

Input

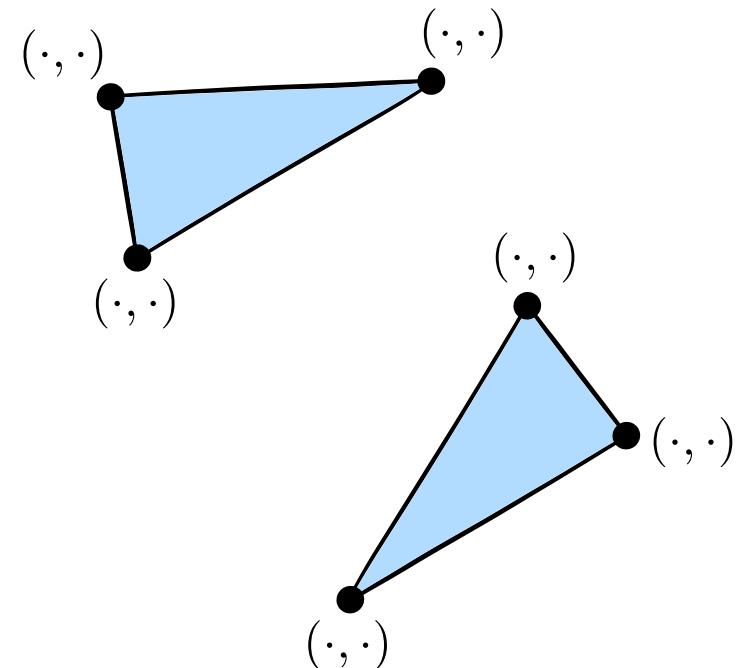


# Lower bound

Input

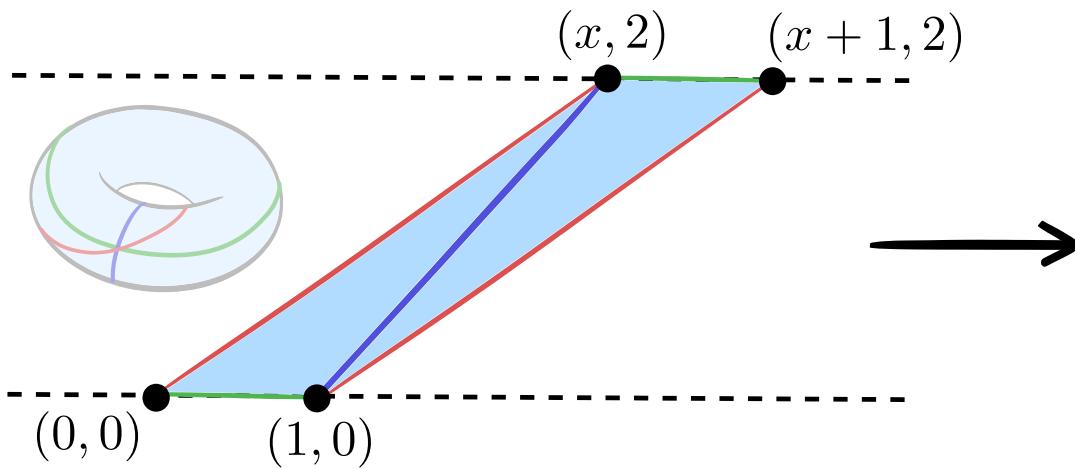


Output: Delaunay

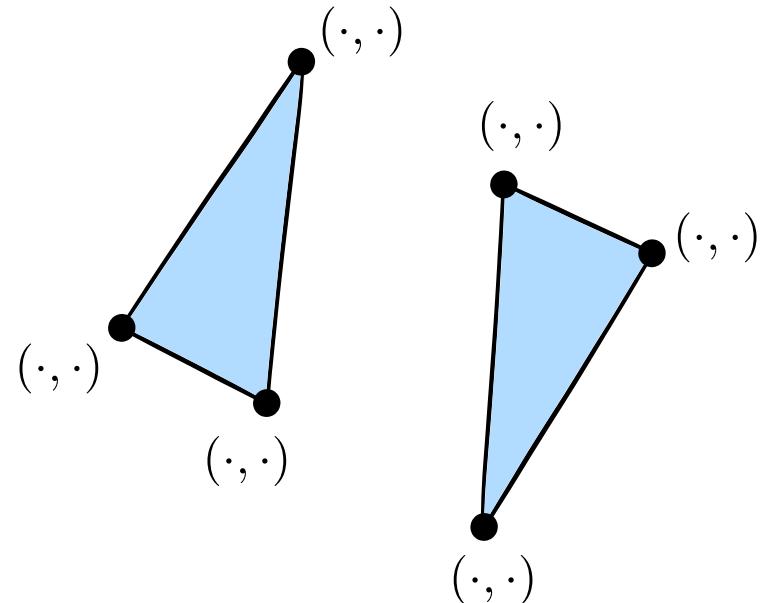


# Lower bound

Input

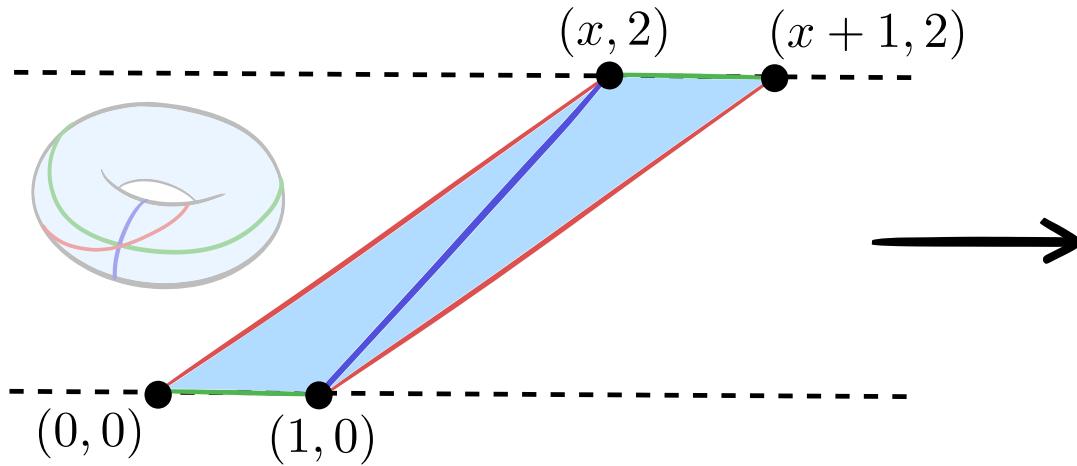


Output: Delaunay

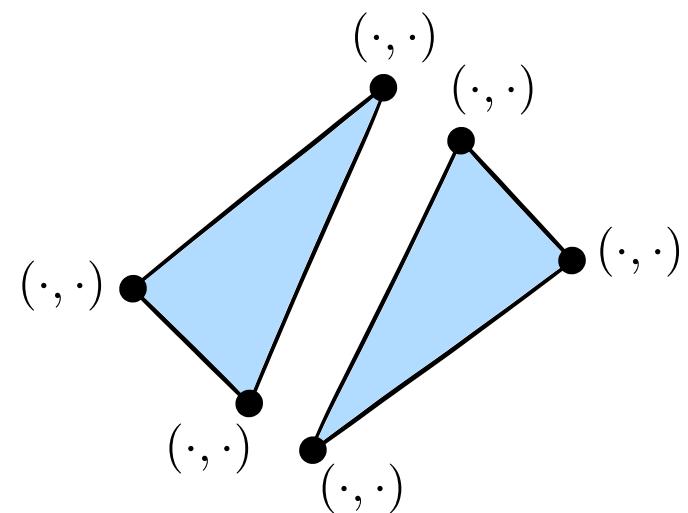


# Lower bound

Input

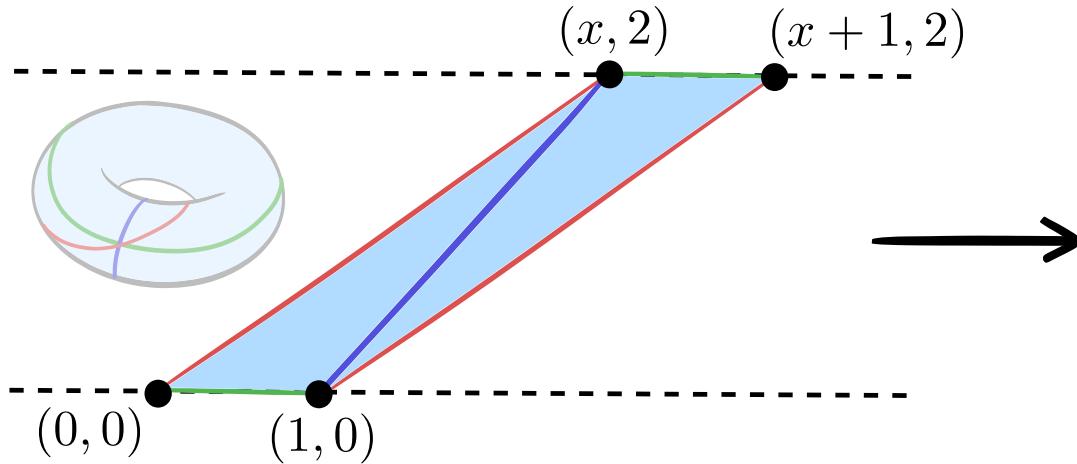


Output: Delaunay

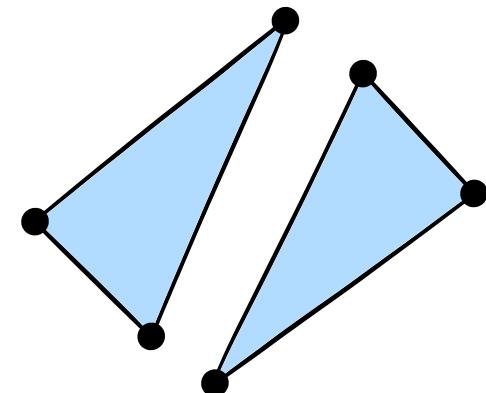


# Lower bound

Input

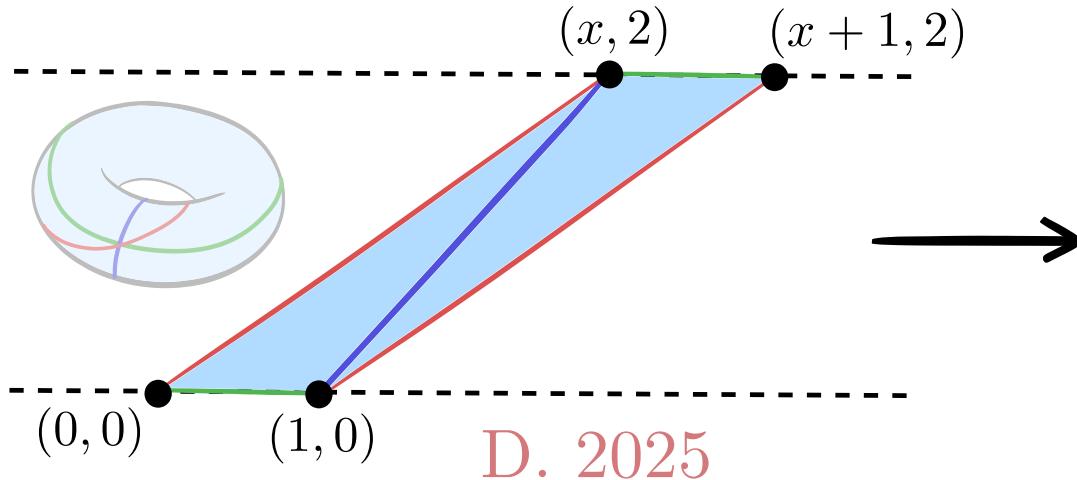


Output: Delaunay

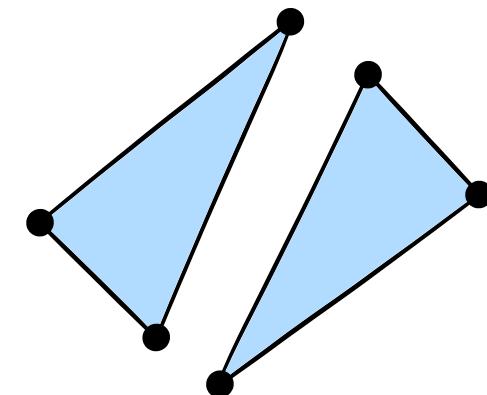


# Lower bound

Input



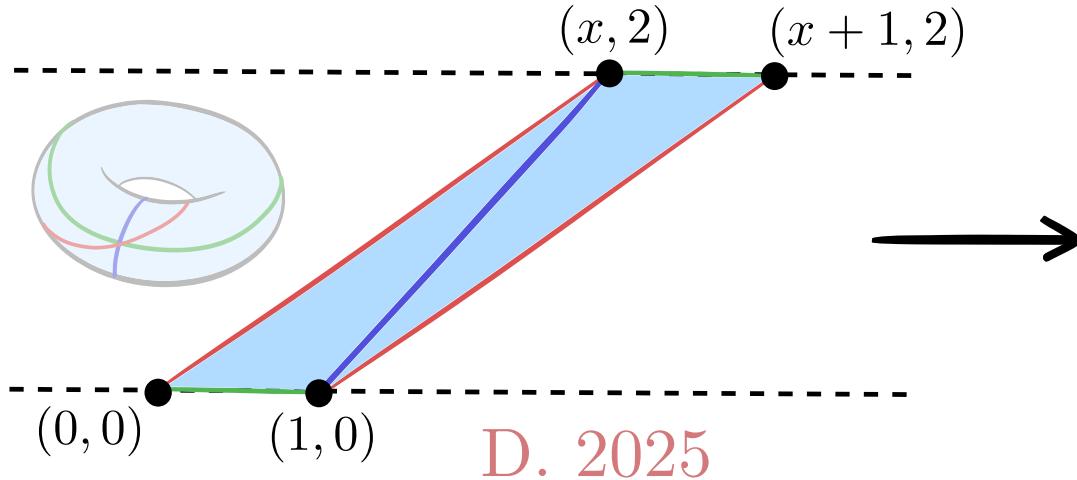
Output: Delaunay



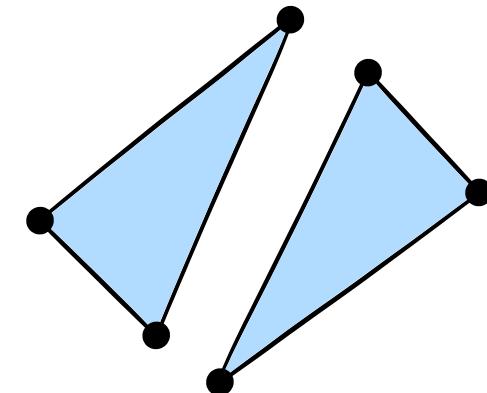
No Real RAM algo can compute  
Delaunay from  $x$  in  $o(\log x)$  time

# Lower bound

Input



Output: Delaunay

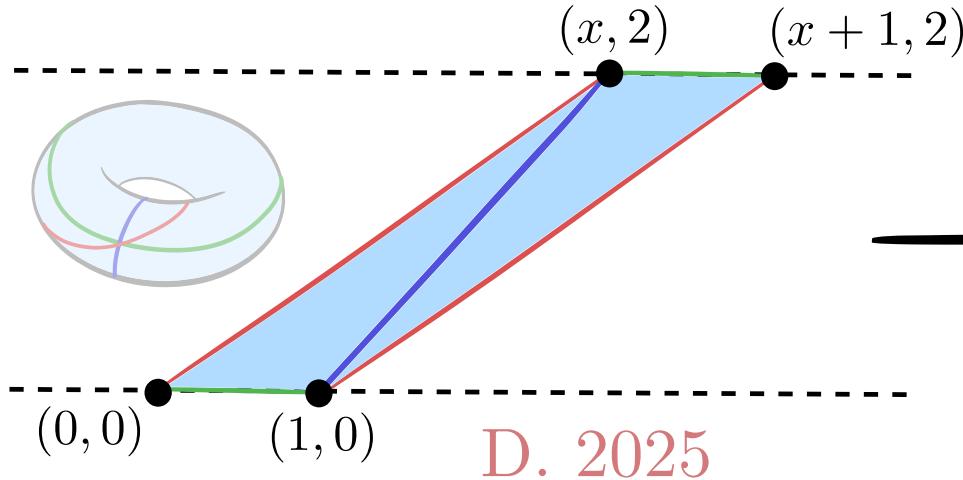


No Real RAM algo can compute  
Delaunay from  $x$  in  $o(\log x)$  time

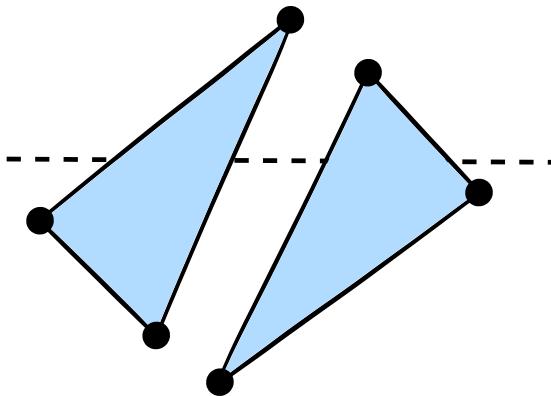
Otherwise we could compute  $\lfloor x \rfloor$  from  $x$  in  $o(\log x)$  time

# Lower bound

Input



Output: Delaunay

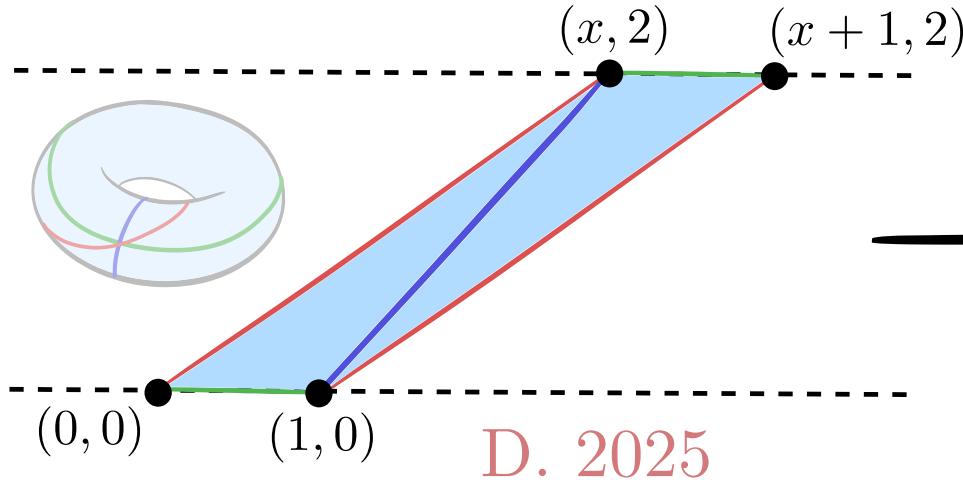


No Real RAM algo can compute  
Delaunay from  $x$  in  $o(\log x)$  time

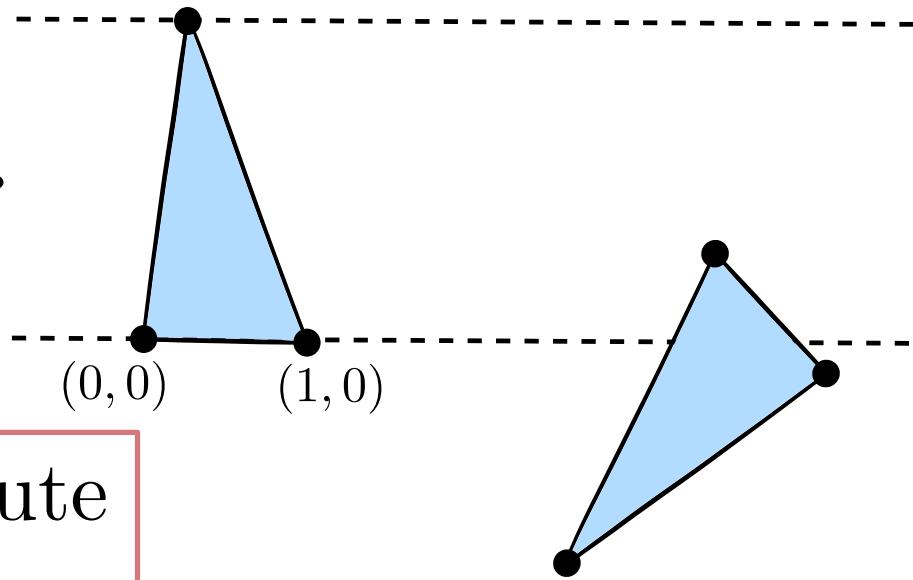
Otherwise we could compute  $\lfloor x \rfloor$  from  $x$  in  $o(\log x)$  time

# Lower bound

Input



Output: Delaunay

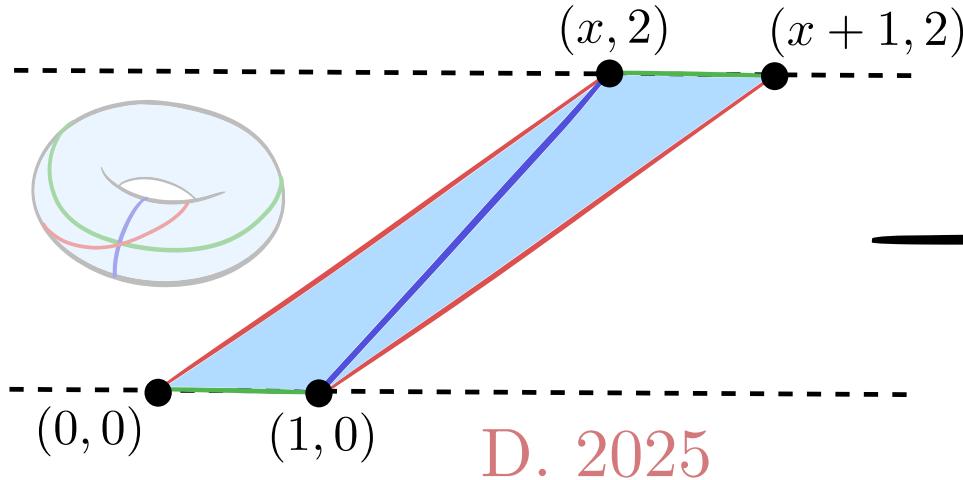


No Real RAM algo can compute  
Delaunay from  $x$  in  $o(\log x)$  time

Otherwise we could compute  $\lfloor x \rfloor$  from  $x$  in  $o(\log x)$  time

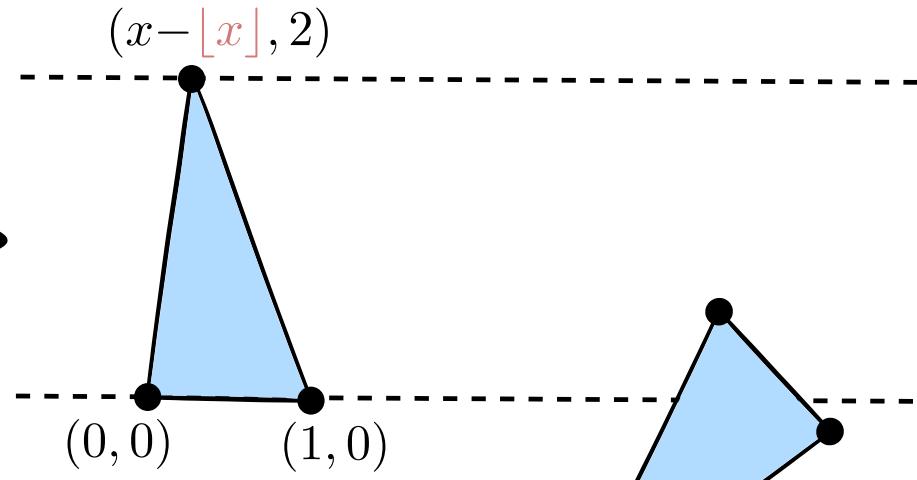
# Lower bound

Input



D. 2025

Output: Delaunay



No Real RAM algo can compute  
Delaunay from  $x$  in  $o(\log x)$  time

Otherwise we could compute  $\lfloor x \rfloor$  from  $x$  in  $o(\log x)$  time

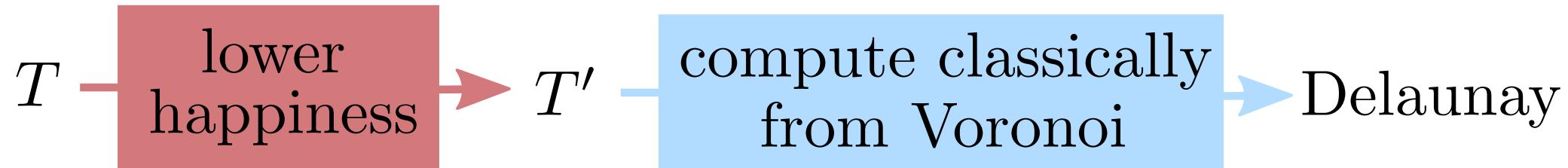
# Algorithm overview

# Algorithm



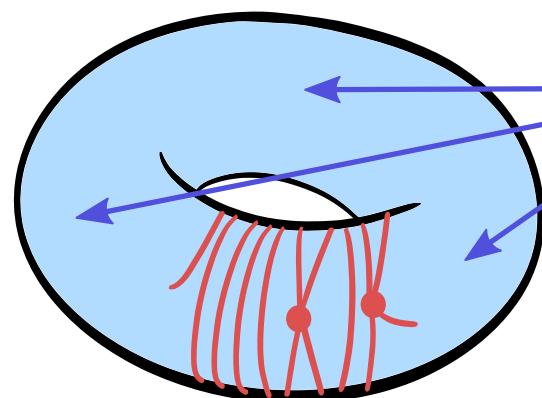
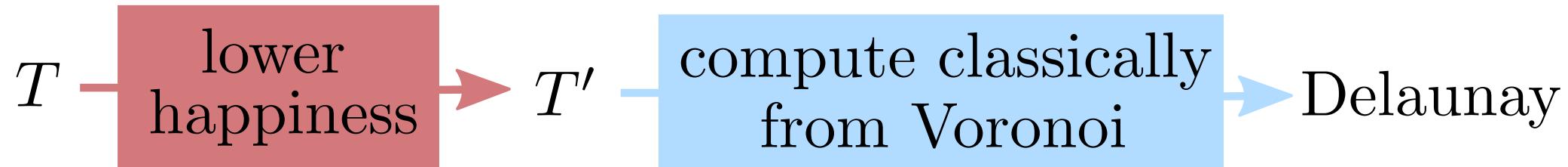
# Algorithm

D. 2025



# Algorithm

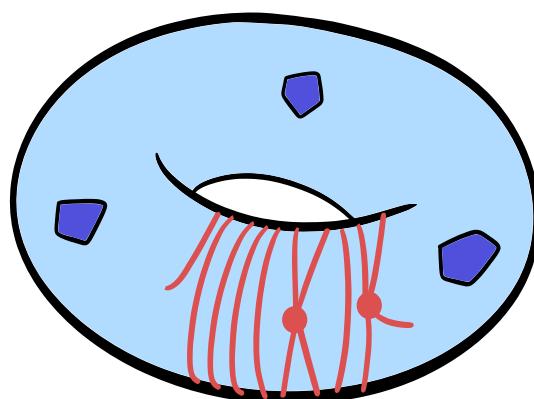
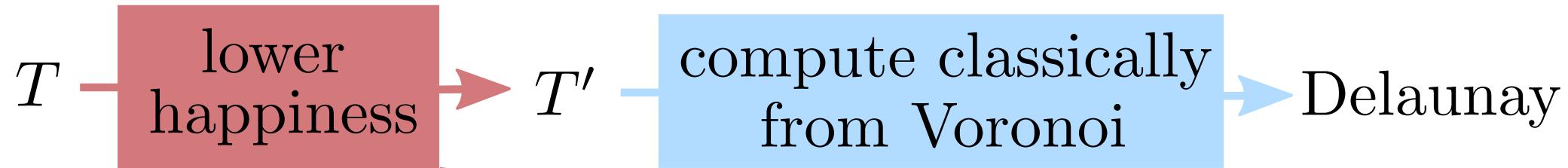
D. 2025



consider the singularities

# Algorithm

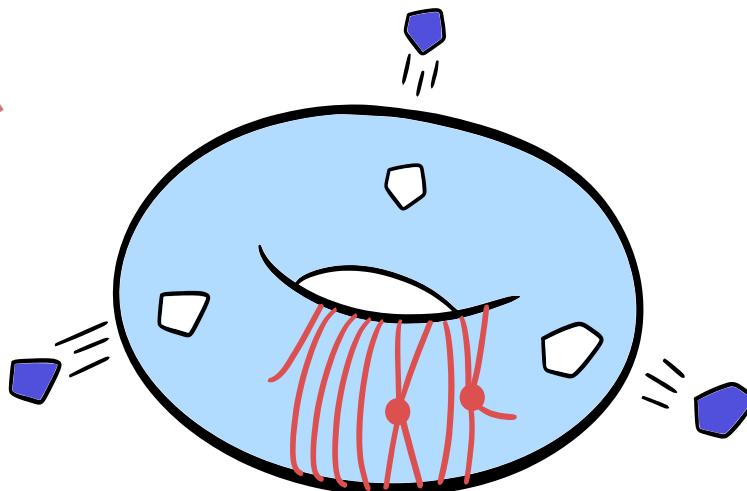
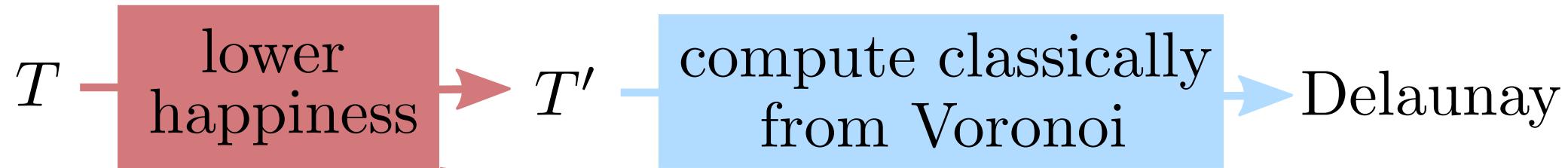
D. 2025



cut out caps around the singularities

# Algorithm

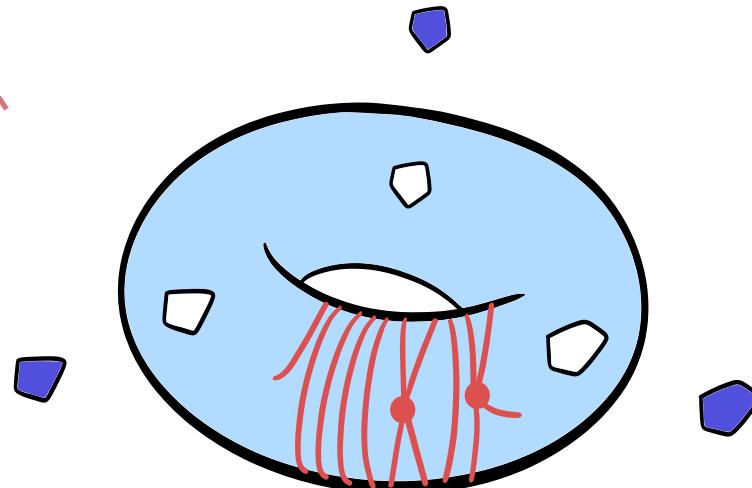
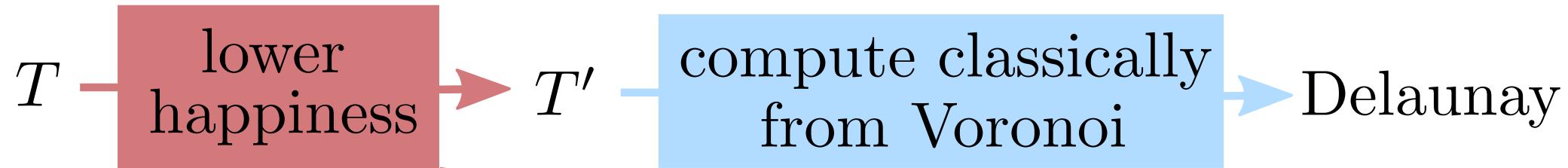
D. 2025



cut out caps around the singularities

# Algorithm

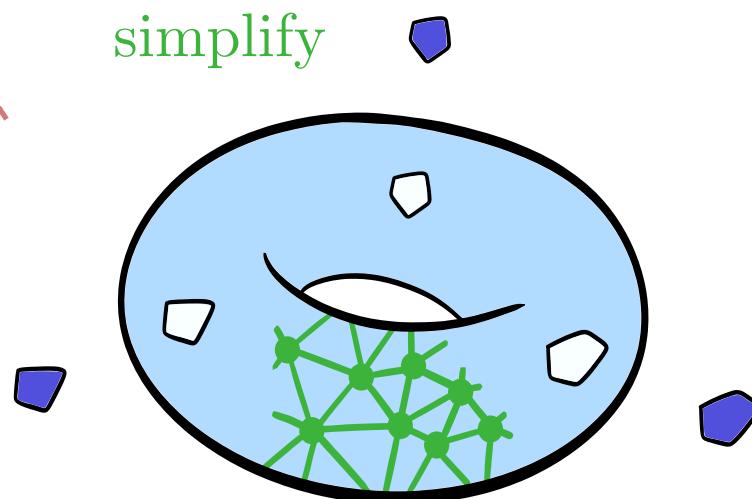
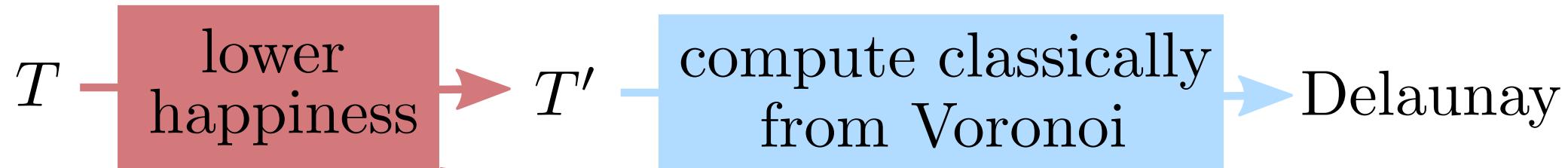
D. 2025



cut out caps around the singularities

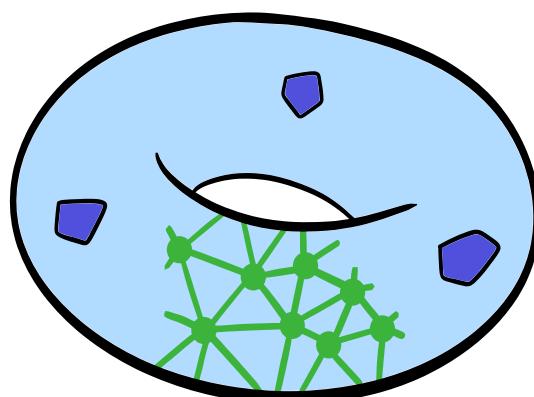
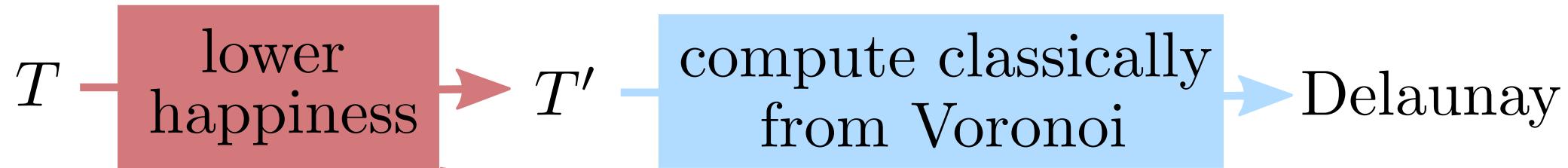
# Algorithm

D. 2025



# Algorithm

D. 2025



put the caps back

# Simplification algorithm

Tuned combination of elementary operations, like

- inserting vertices in edges
- inserting edges in faces
- deleting vertices

repeated many times

some simplify the geometry,  
others decrease # vertices

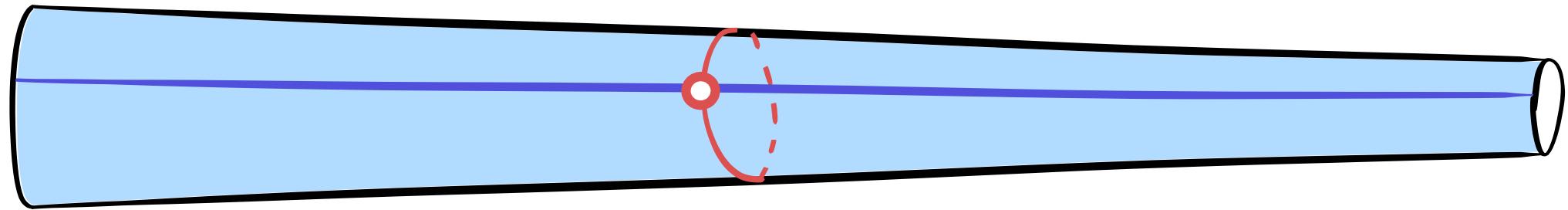
# Analysis

Show that during execution:

1. # vertices stays bounded
2. Geometry gets simpler and simpler



Enclosure



# Untangling Graphs

Computing Delaunay Triangulations

Other Results  
and Possible Continuations

# Untangling Graphs

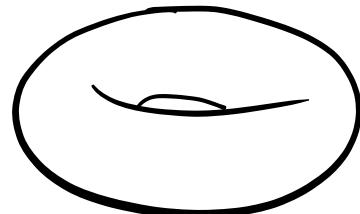
## Computing Delaunay Triangulations

Other Results  
and Possible Continuations

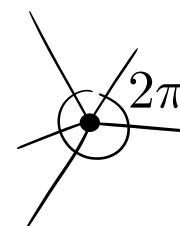
# Other results

D., 2022-23

Upper bound for # Delaunay flips on flat tori,  
tight up to constant factor



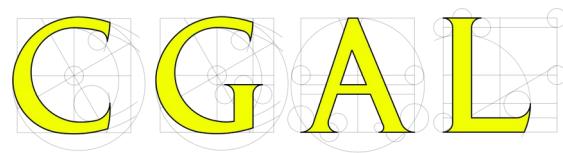
topological shape of a torus



$2\pi$  around each vertex

# Other results

Despré, D., Pouget, and Teillaud, 2025



package for computing with  
hyperbolic surfaces

generation (genus 2 only)

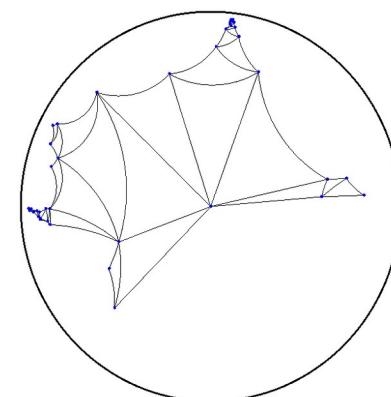
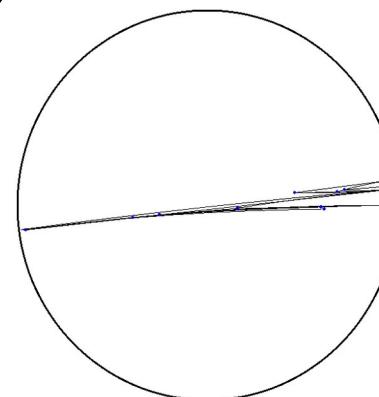


Triangulated hyperbolic  
surface

Delaunay flip



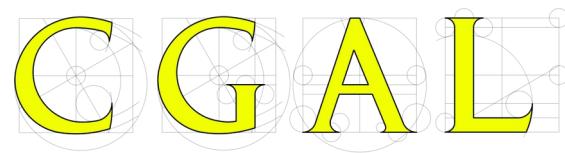
visualization



# Other results

Despré, D., Pouget, and Teillaud

*Exact computations!*  
package for computing  
hyperbolic surfaces



generation (genus 2 only)

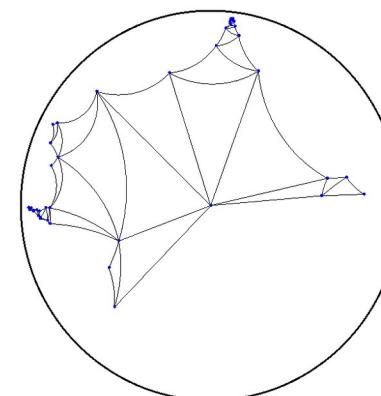
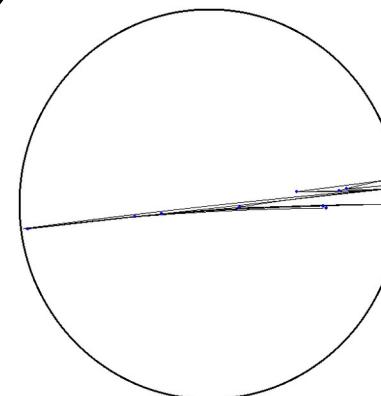


Triangulated hyperbolic  
surface

Delaunay flip

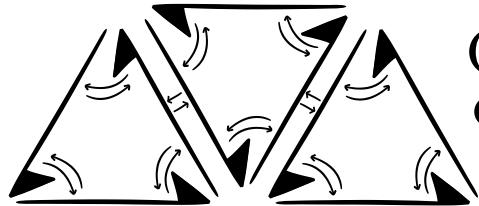


visualization



Triangulated hyperbolic  
surface

# Combinatorics

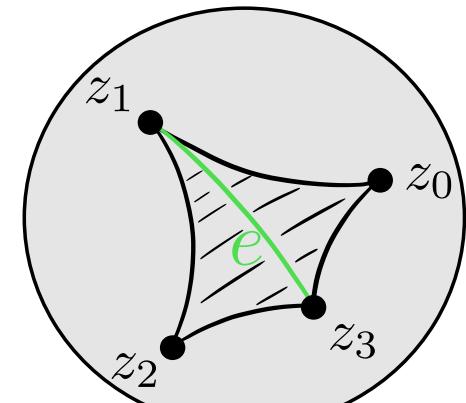


CGAL  
combinatorial maps

Triangulated hyperbolic  
surface

# Geometry

- each edge is decorated with a complex number (cross ratio)



$$e \rightarrow \frac{(z_3 - z_1)(z_4 - z_2)}{(z_3 - z_2)(z_4 - z_1)}$$

# Possible continuations

- generation of hyperbolic surfaces of genus  $\geq 3$
- what is the complexity of Delaunay flips algo?
- certifying that a drawing cannot be untangled
- untangling by homotopy moves
- extension to non orientable surfaces
- what is the complexity of untangling?
- minimizing crossings of graphs by homotopy
- how unique reducing triangulations are?