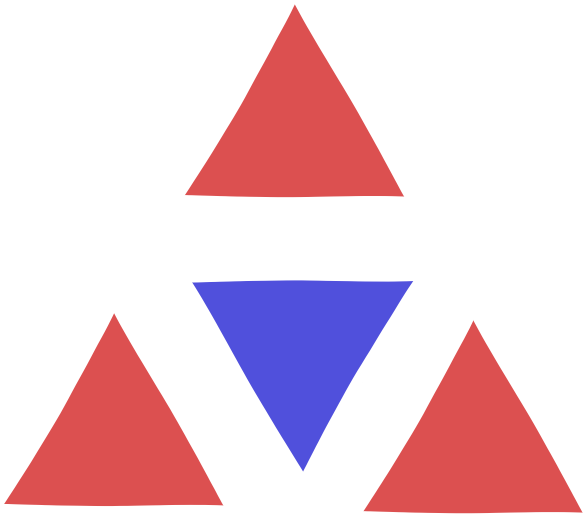
Algorithms for Topological and Metric Surfaces



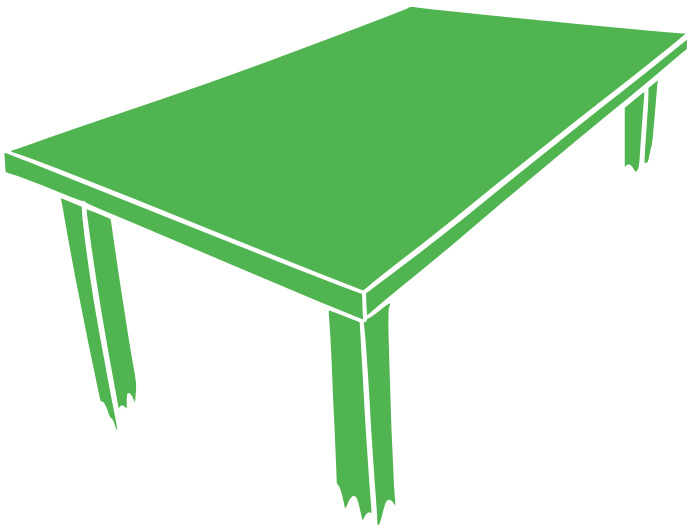
Loïc Dubois

Computational Geometry

Design algorithms for geometric problems

This thesis

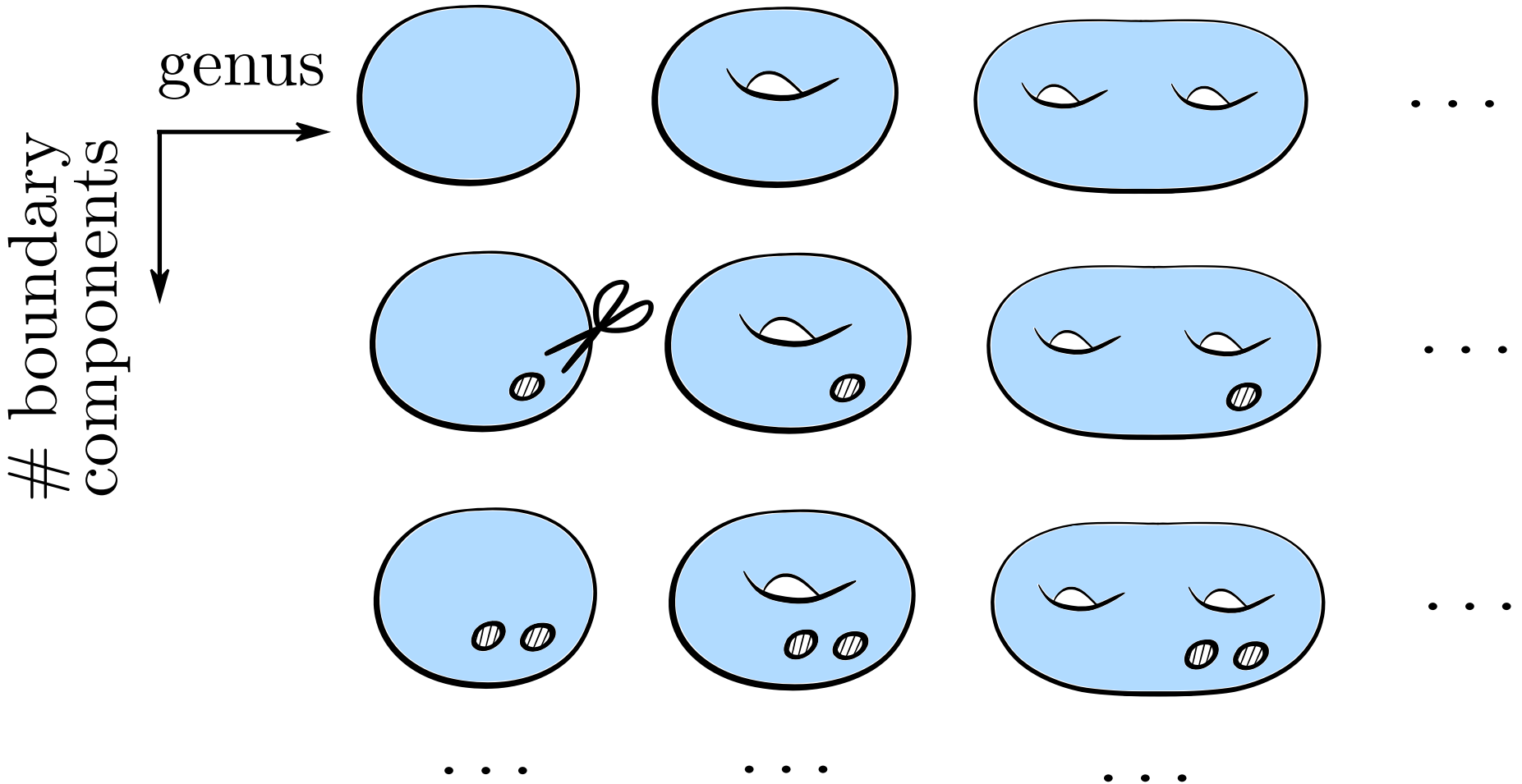
Focus on surfaces



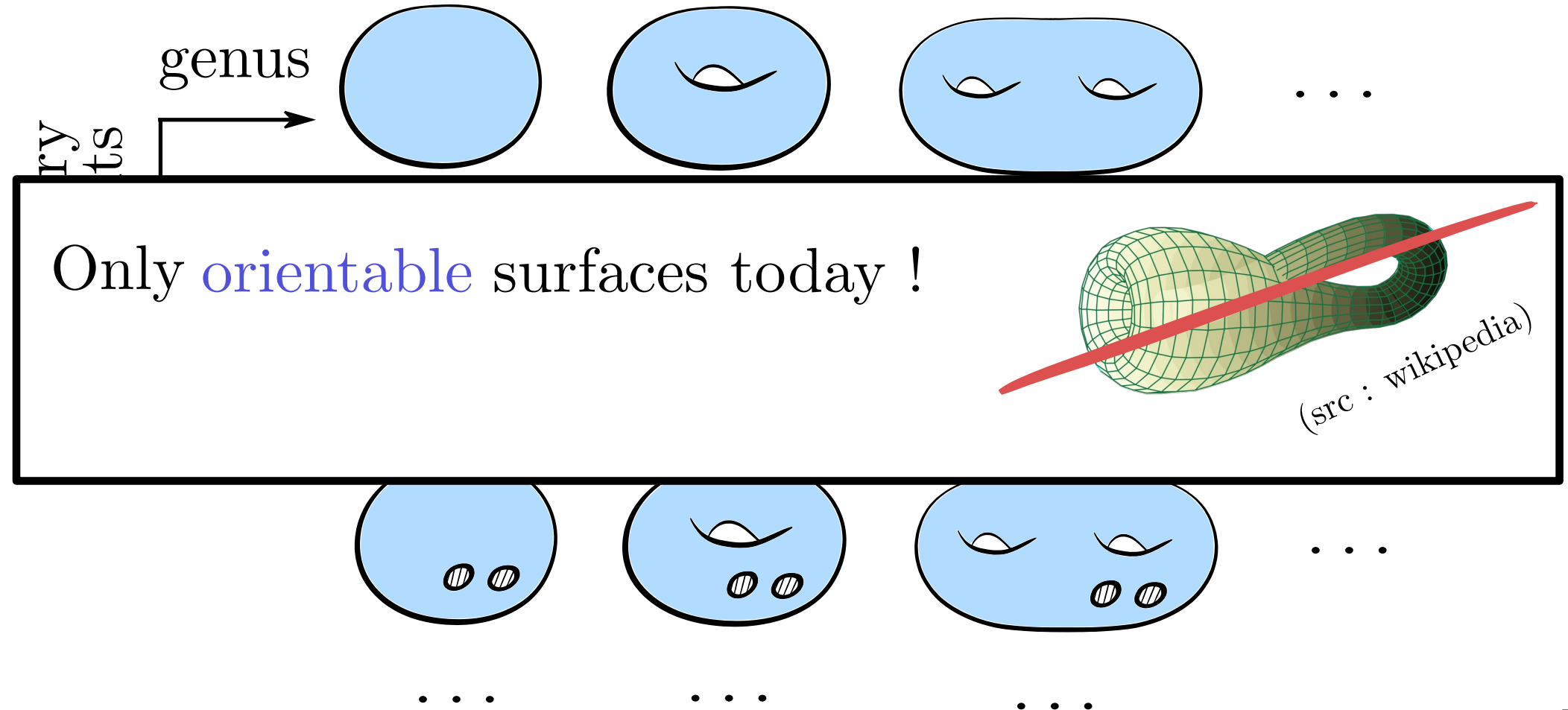
Topology

image by Crane and Segerman

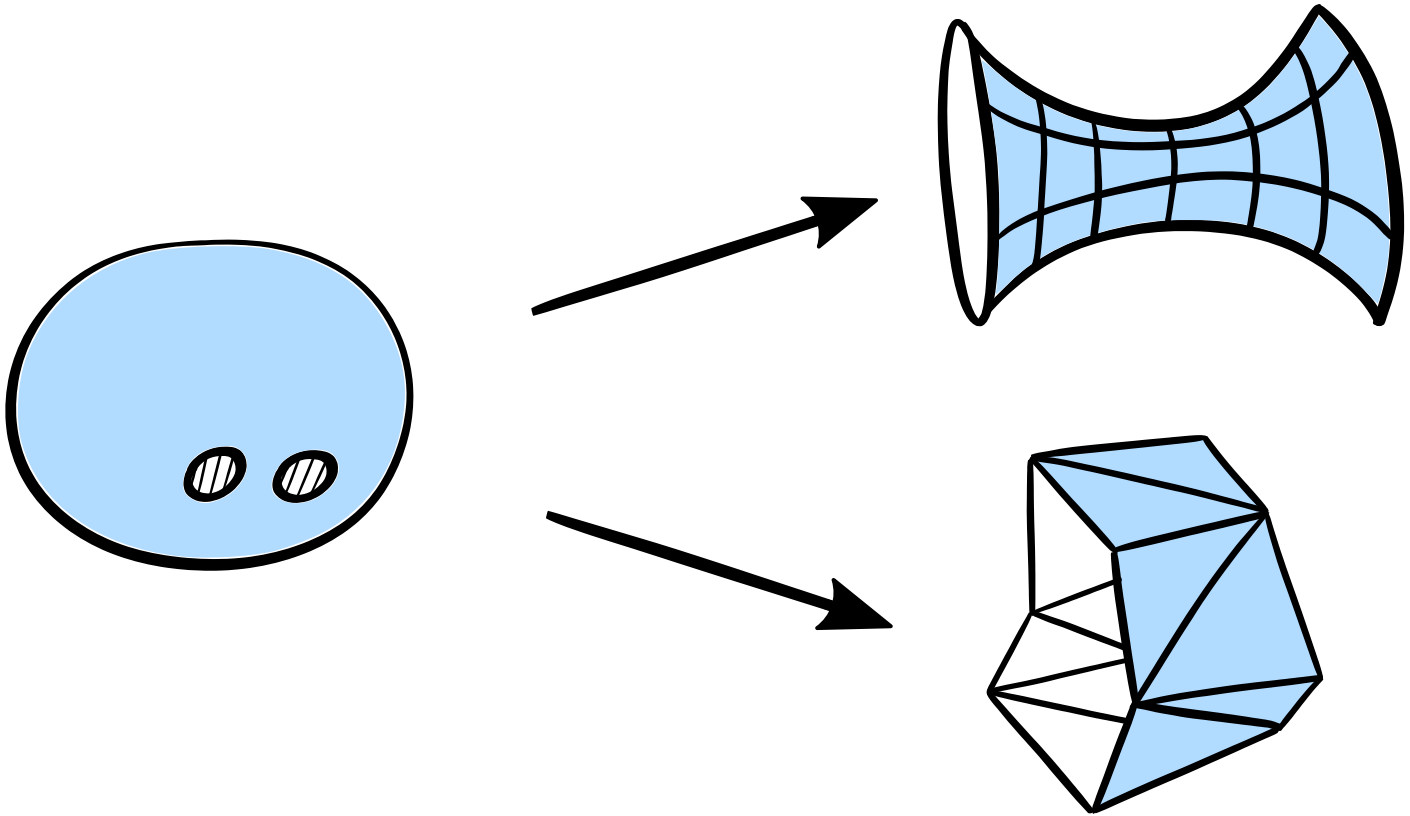
Topological Surfaces



Topological Surfaces



Metrics on surfaces



Untangling Graphs

Computing Delaunay Triangulations

Conclusion

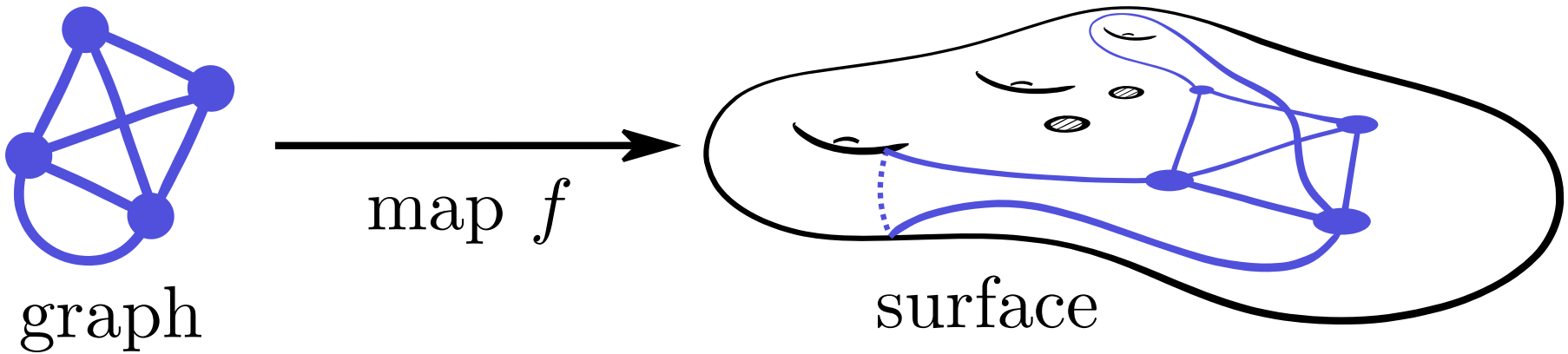
Untangling Graphs

Computing Delaunay Triangulations

Conclusion

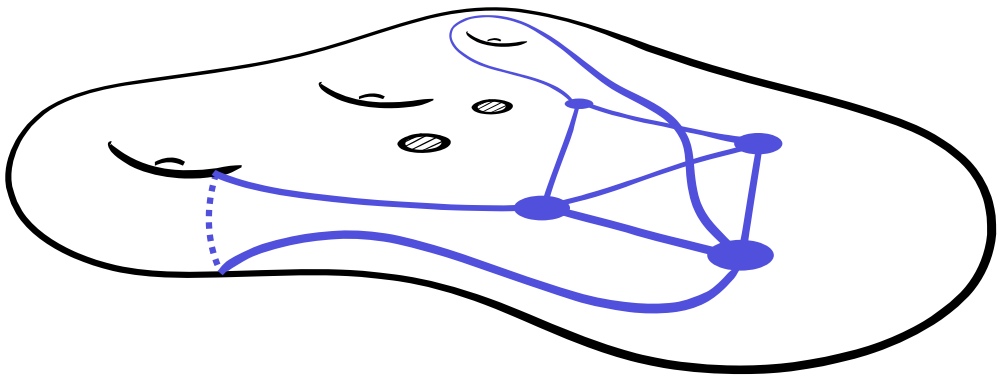
Problem: untangling graphs

Input:

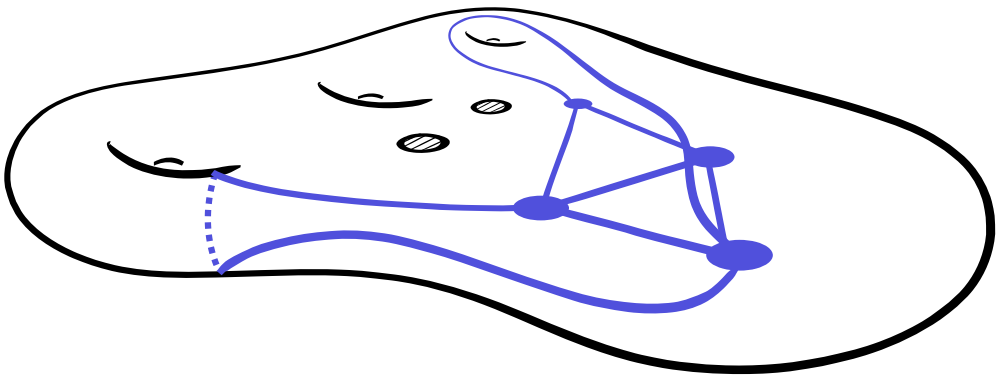


Goal: remove all crossings by deforming f

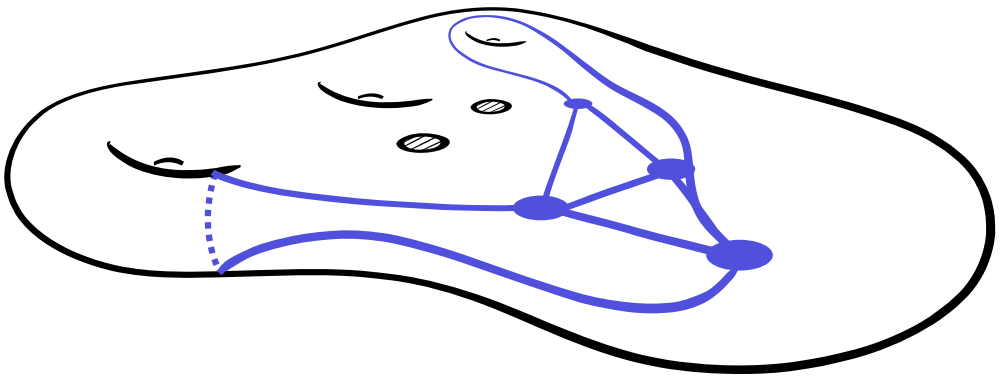
Problem: untangling graphs



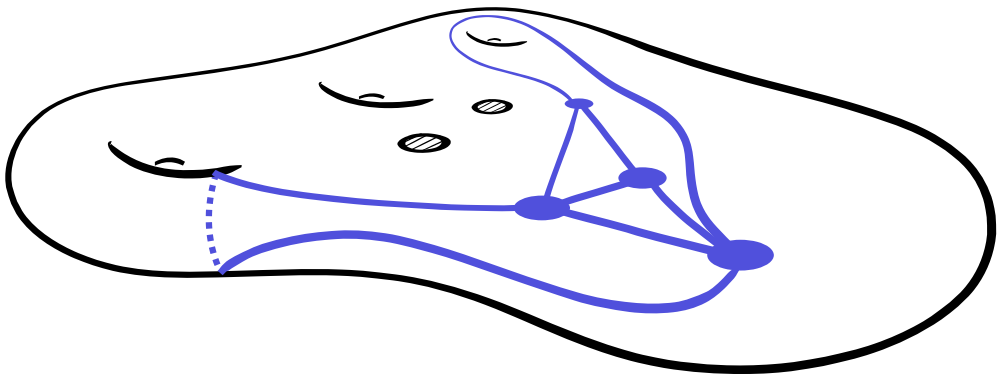
Problem: untangling graphs



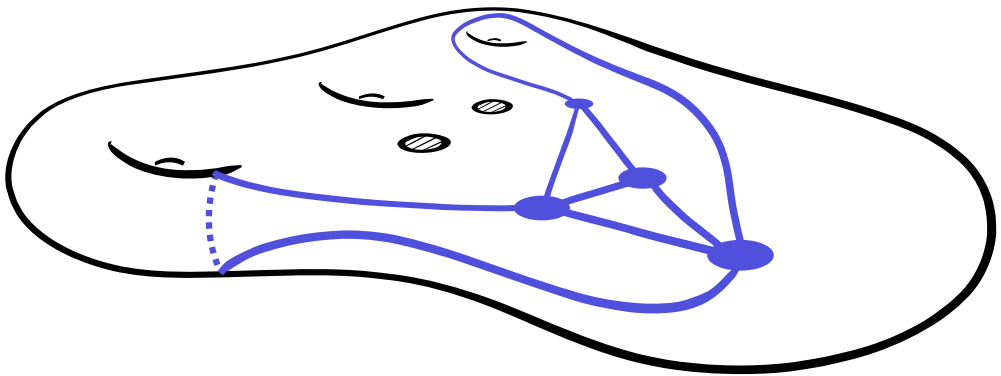
Problem: untangling graphs



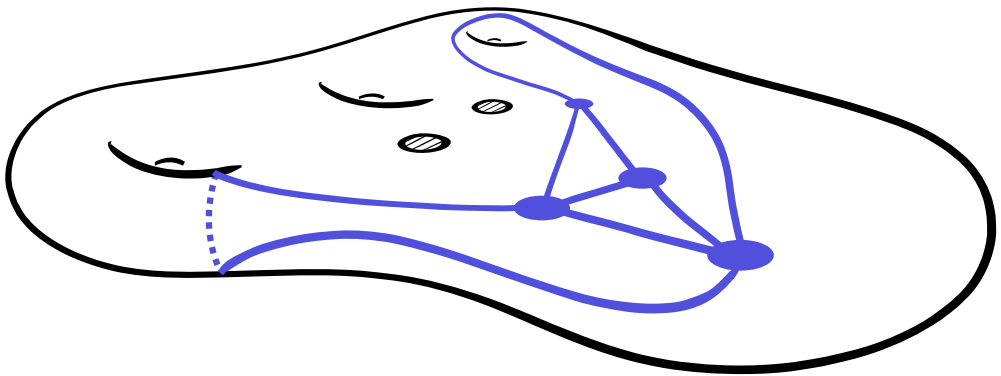
Problem: untangling graphs



Problem: untangling graphs

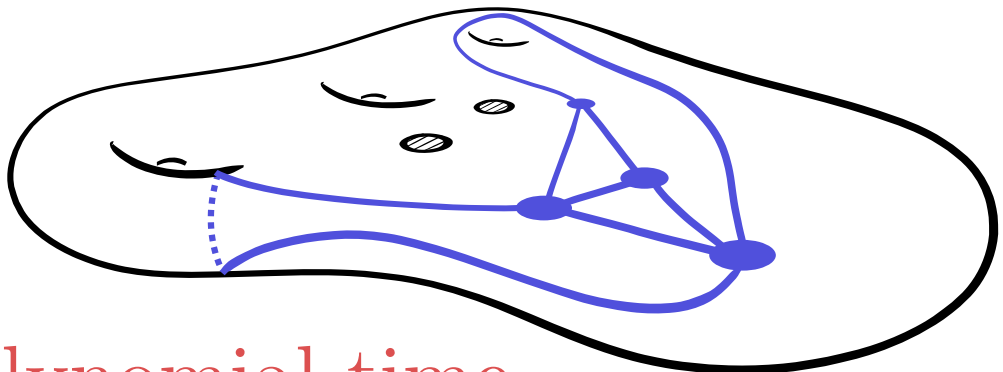


Problem: untangling graphs



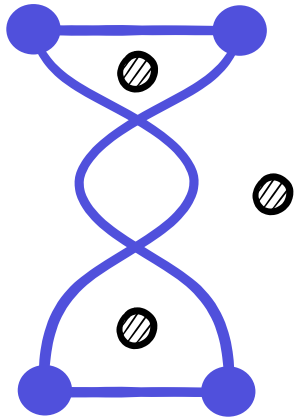
Output: Yes (+ untangled drawing) or No

Problem: untangling graphs

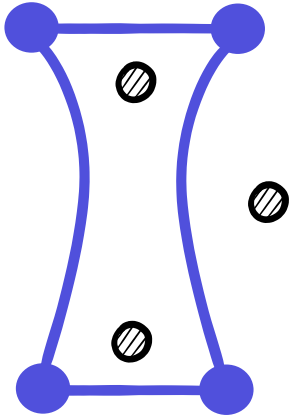


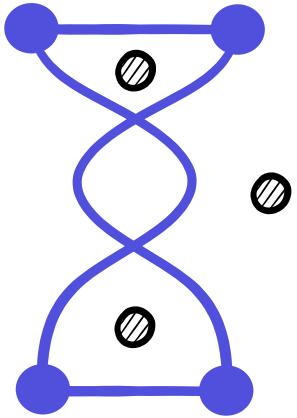
We obtain the first polynomial time algorithms for this problem

Output: Yes (+ untangled drawing) or No

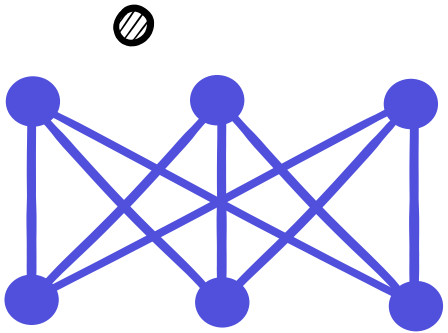


Yes:

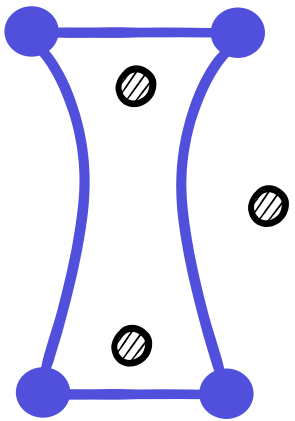


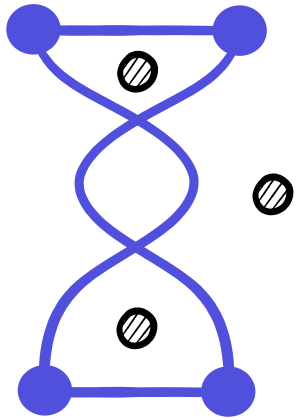


Yes:

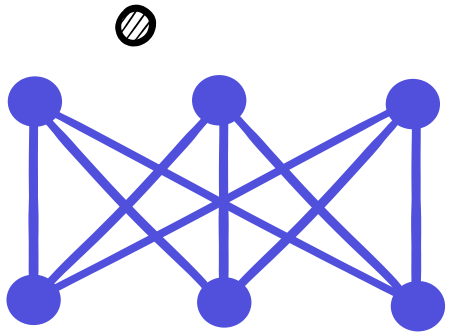


No

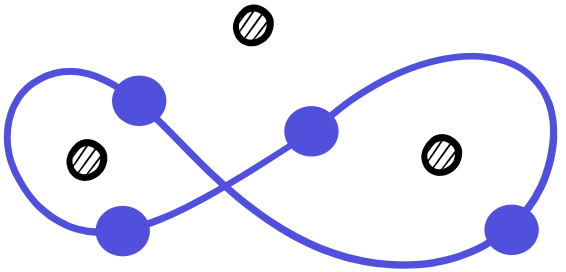




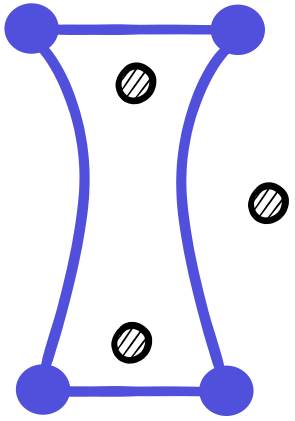
Yes:



No

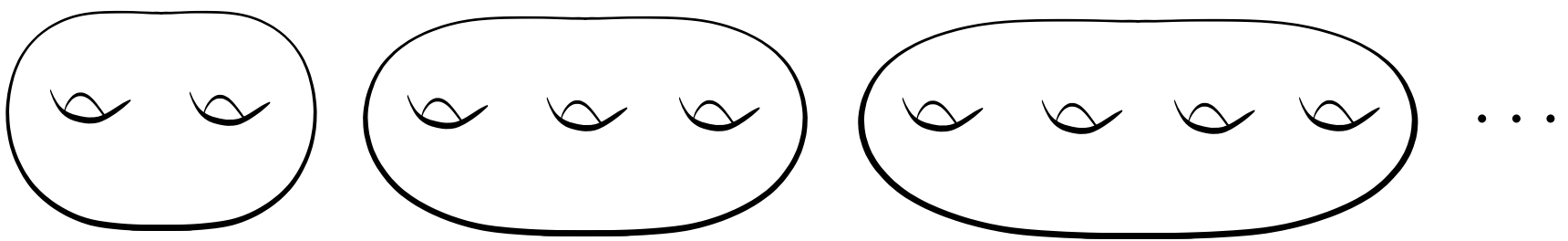


No



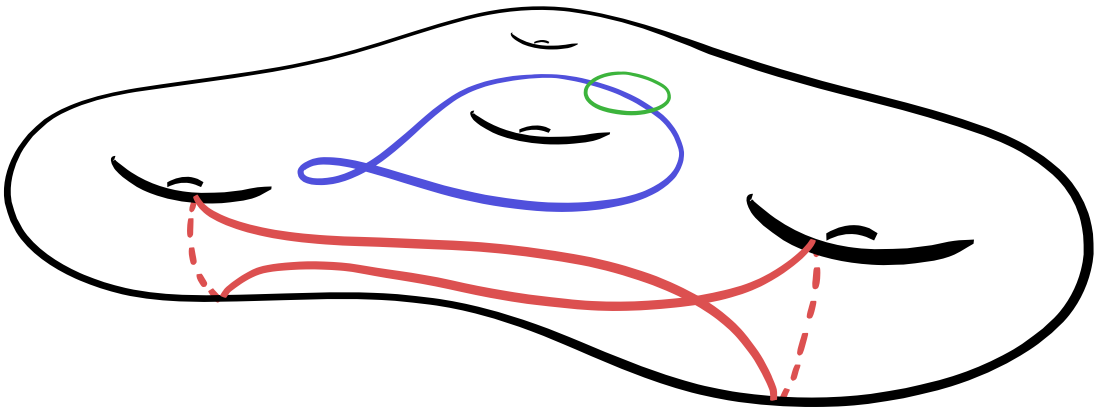
Related works

We focus on surfaces without boundary of genus ≥ 2



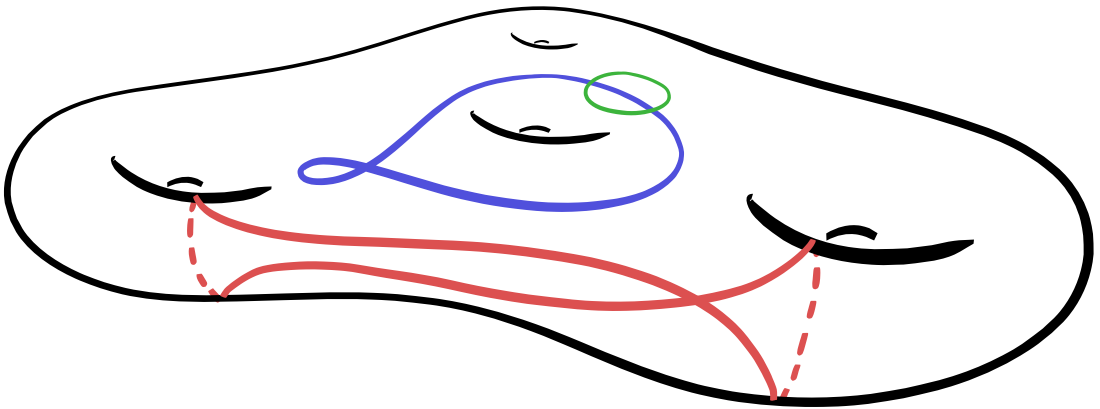
Related problem: making curves cross minimally

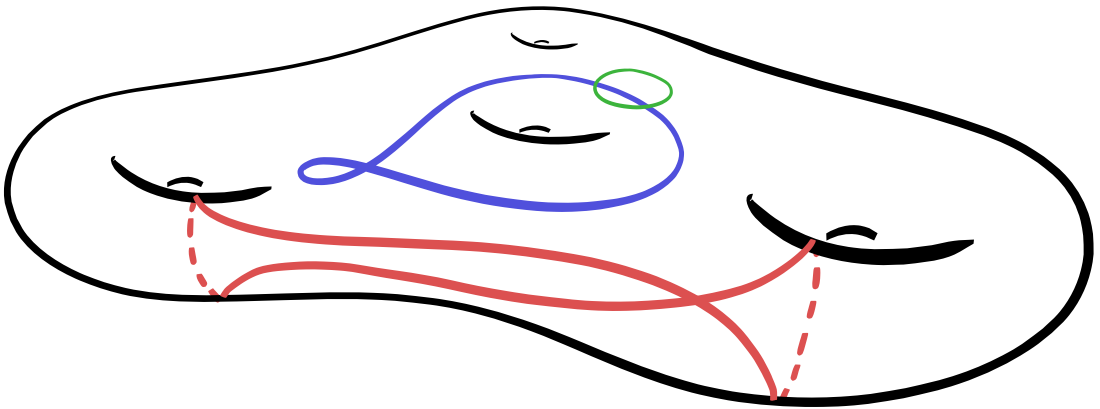
Input: closed curves on a surface

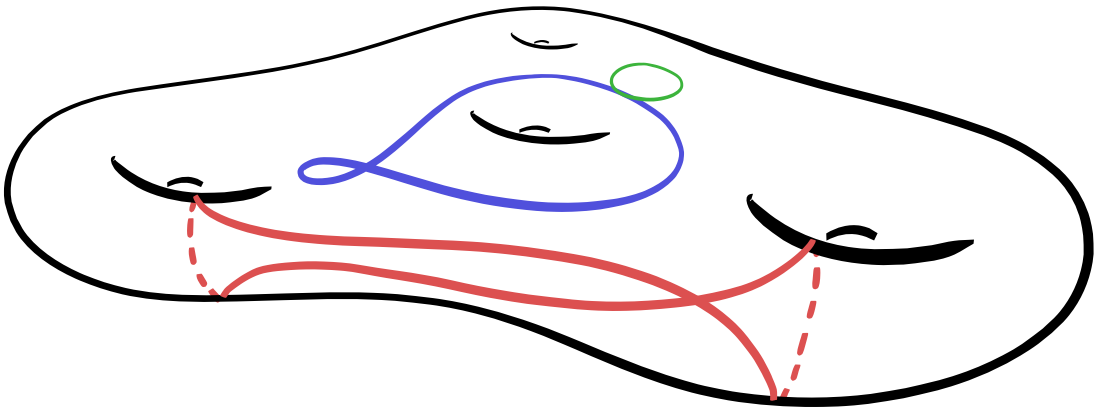


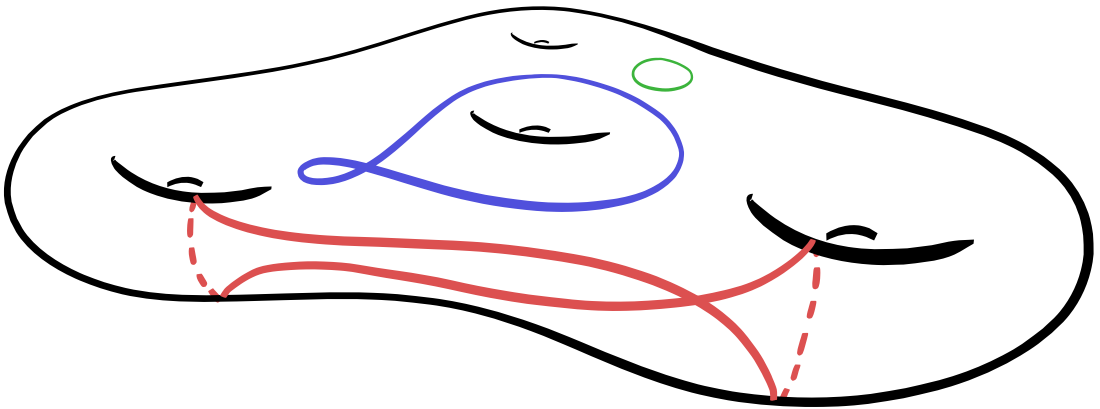
Goal:

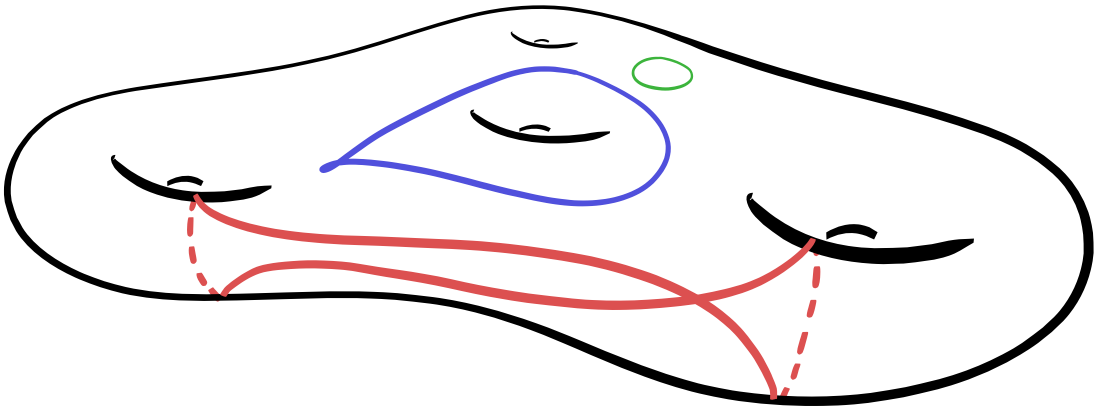
minimize the # crossings by deforming the curves

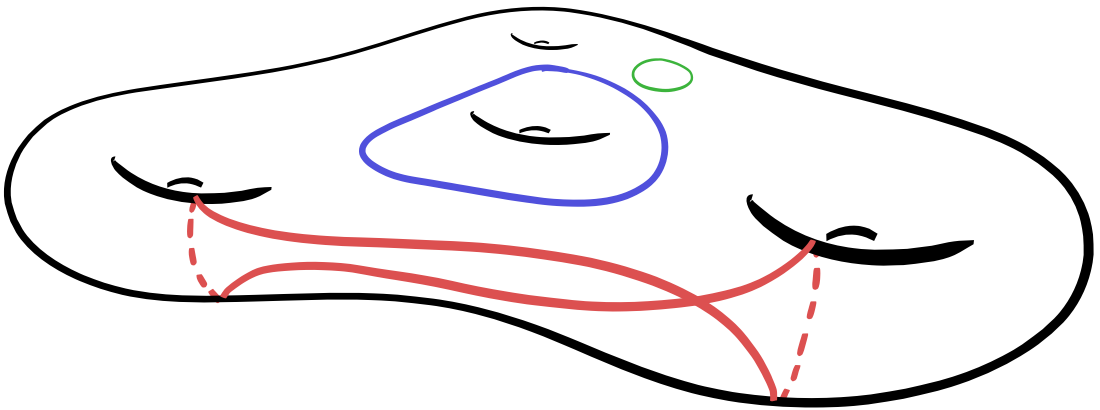


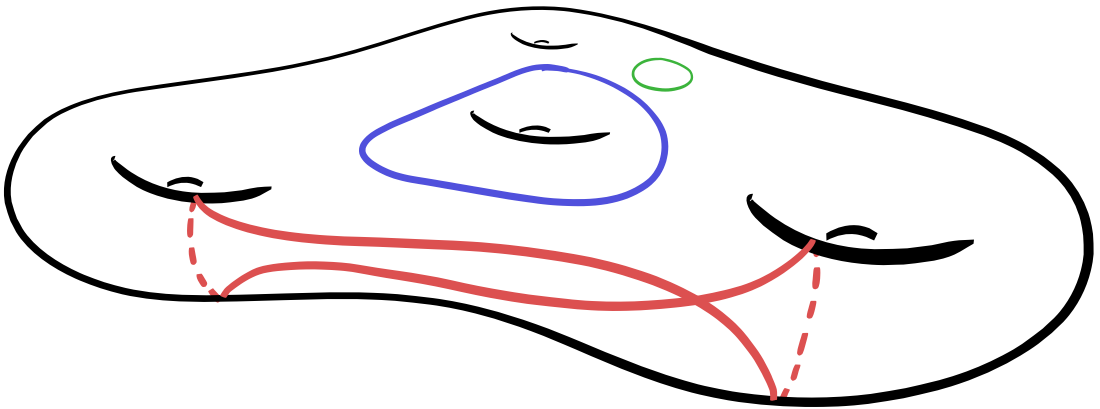












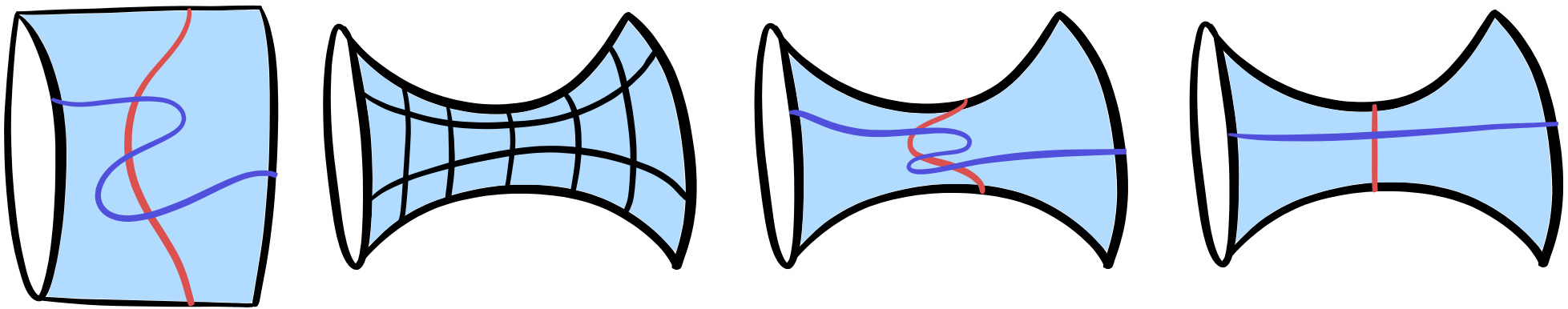
Output: min # of crossings (+ optimal curves)

Many works related to making curves cross minimally!

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

Method for making curves cross minimally

Poincaré, 1905



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

The special shape

negative curvature:

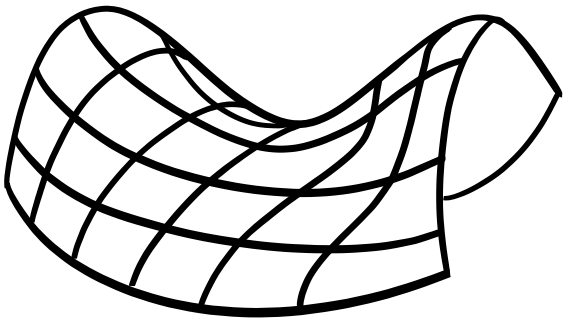


image by Susan Lombardo

almost all surfaces
can be curved negatively

The key property

On a negatively curved surface,

straight curves cross minimally

The key property

On a negatively curved surface,

straight curves cross minimally

(does not hold on all surfaces)

The key property

On a negatively curved surface,

straight curves cross minimally $\Longleftarrow$ ever
into

The key property

On a negatively curved surface,

straight curves cross minimally $\Longleftarrow$ every pat
into a uni

The key property

On a negatively curved surface,

straight curves cross minimally $\Longleftarrow$ every path can be
into a unique strai

The key property

On a negatively curved surface,

ht curves cross minimally $\Longleftarrow$ every path can be deform
into a unique straight pa

The key property

On a negatively curved surface,

cross minimally $\Longleftarrow$ every path can be deformed into a unique straight path

The key property

On a negatively curved surface,

inimally $\Longleftarrow$ every path can be deformed into a unique straight path

The key property

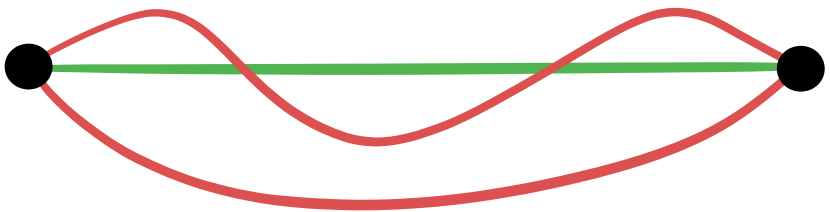
On a negatively curved surface,

every path can be deformed into a unique straight path

The key property

On a negatively curved surface,

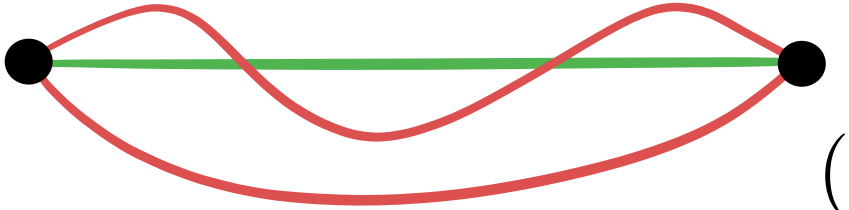
every path can be deformed into a unique straight path



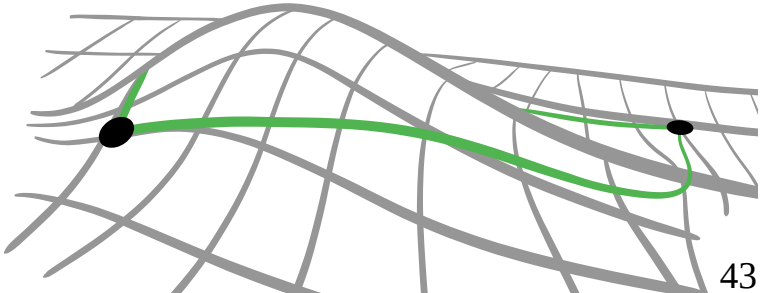
The key property

On a negatively curved surface,

every path can be deformed into a unique straight path

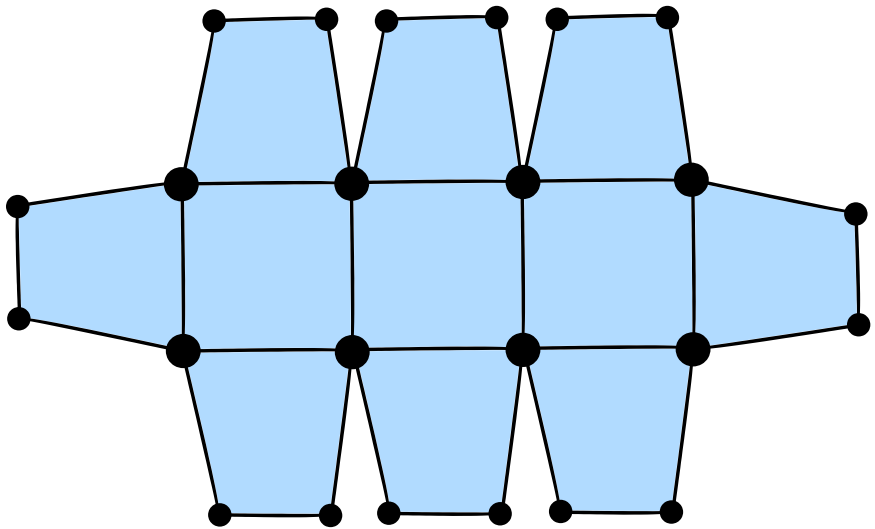


(does not hold on all surfaces)

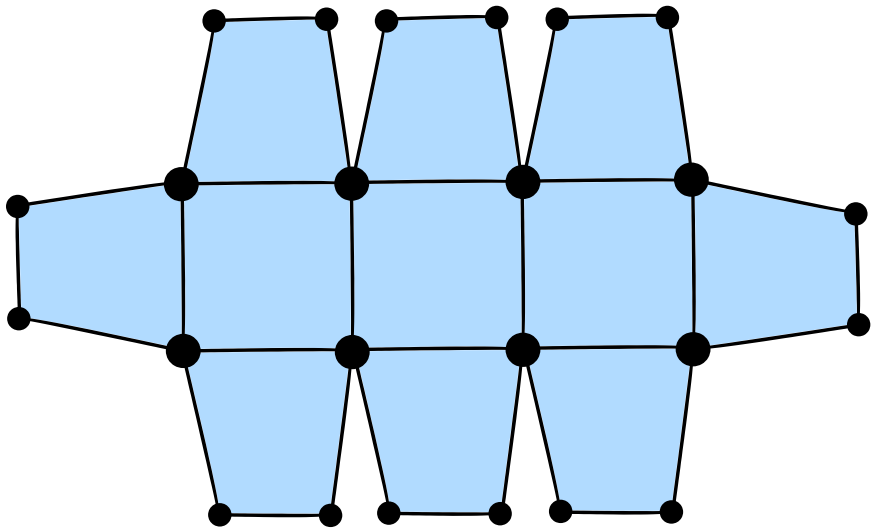


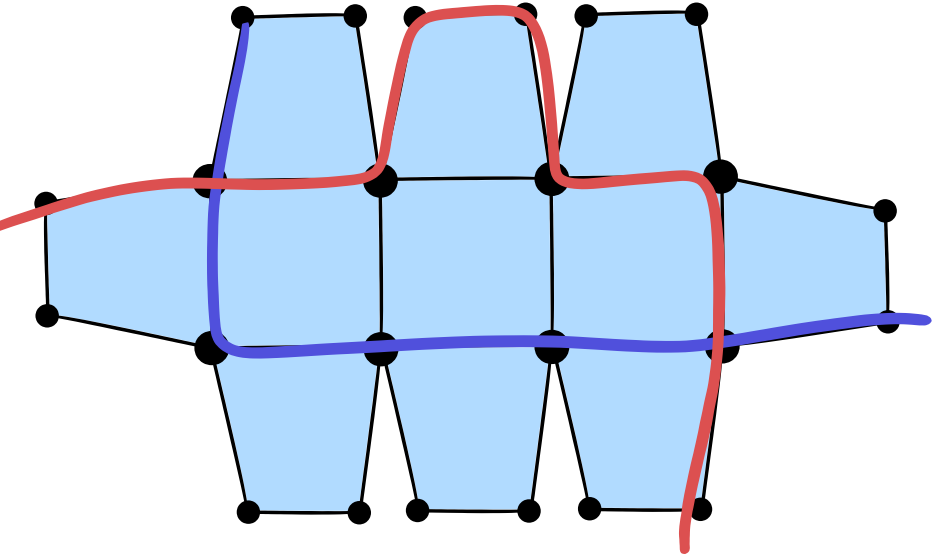
Discrete model of negatively curved surfaces?

Lazarus and Rivaud, 2012
Erickson and Whittlesey, 2013

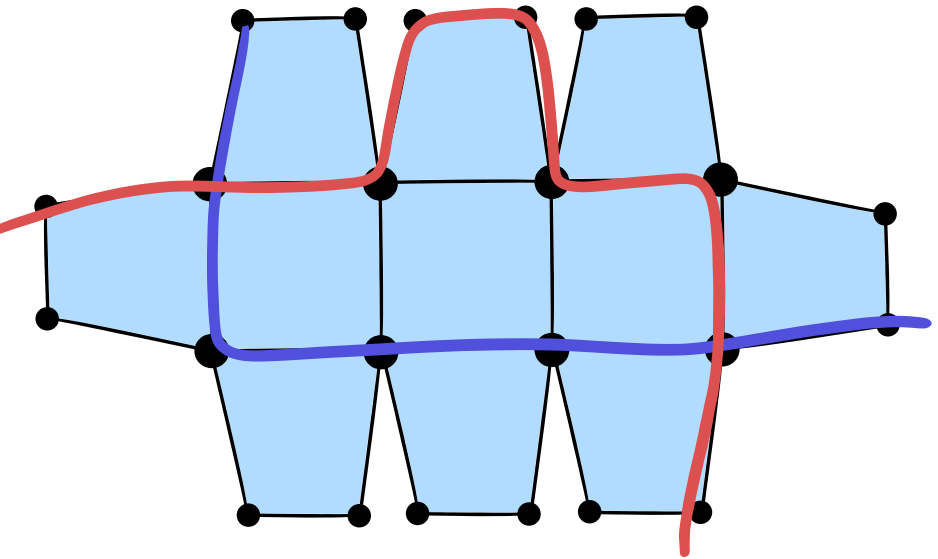


System of quads

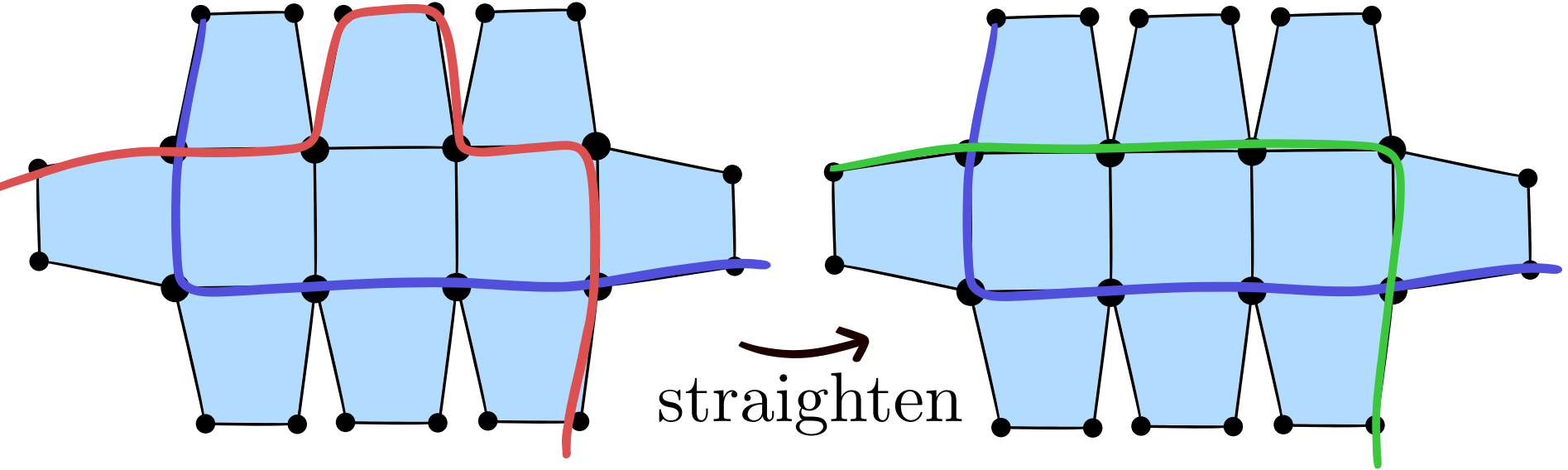




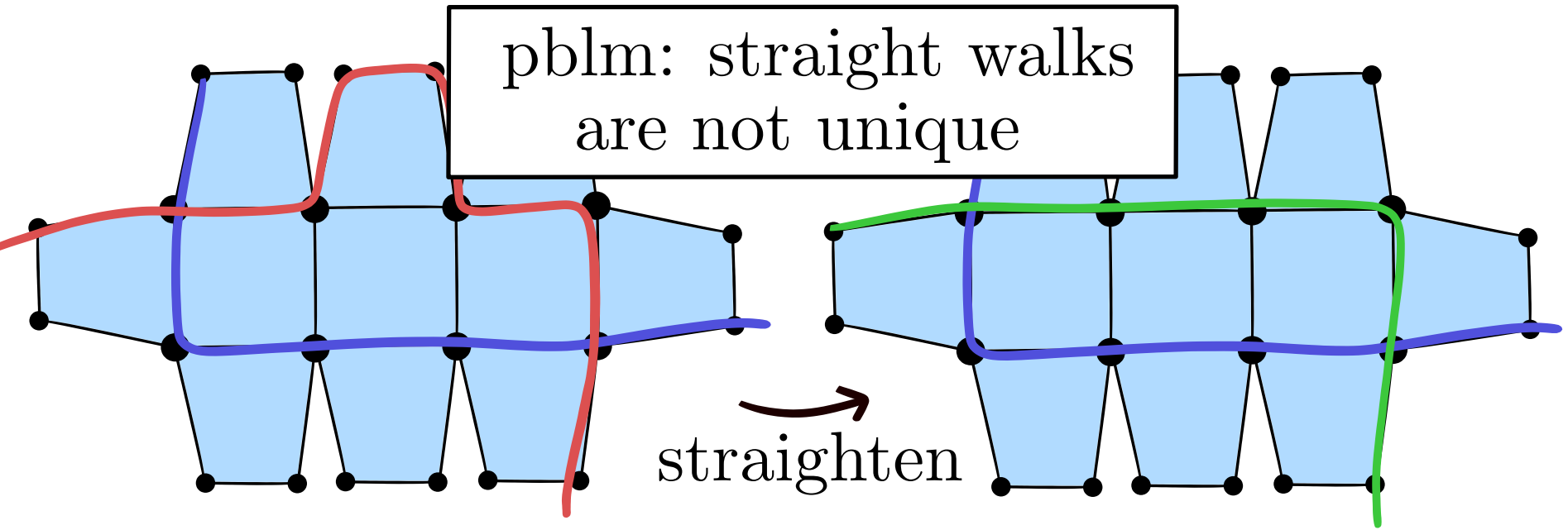
Algo for making curves cross minimally



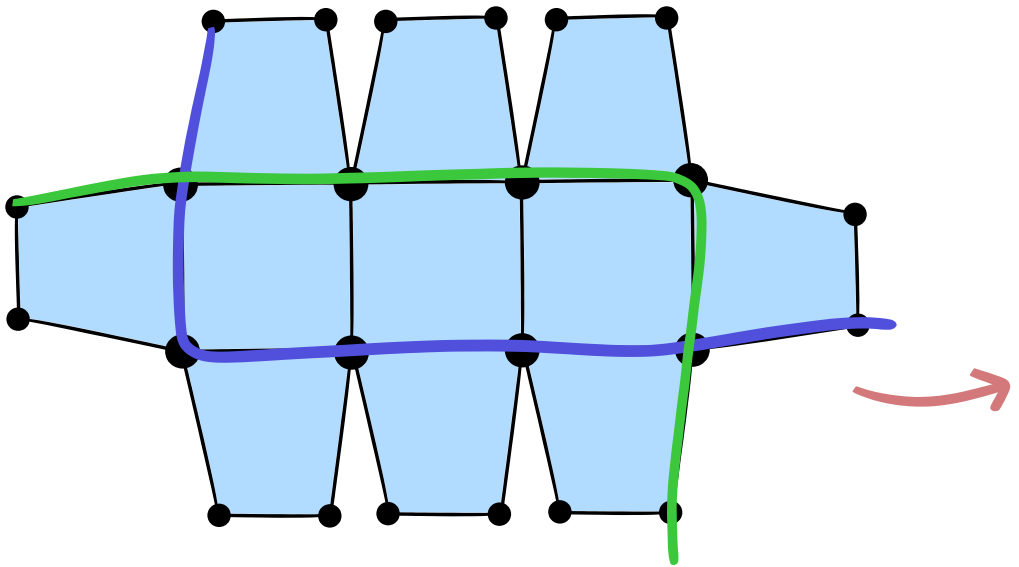
Algo for making curves cross minimally



Algo for making curves cross minimally



Algo for making curves cross minimally



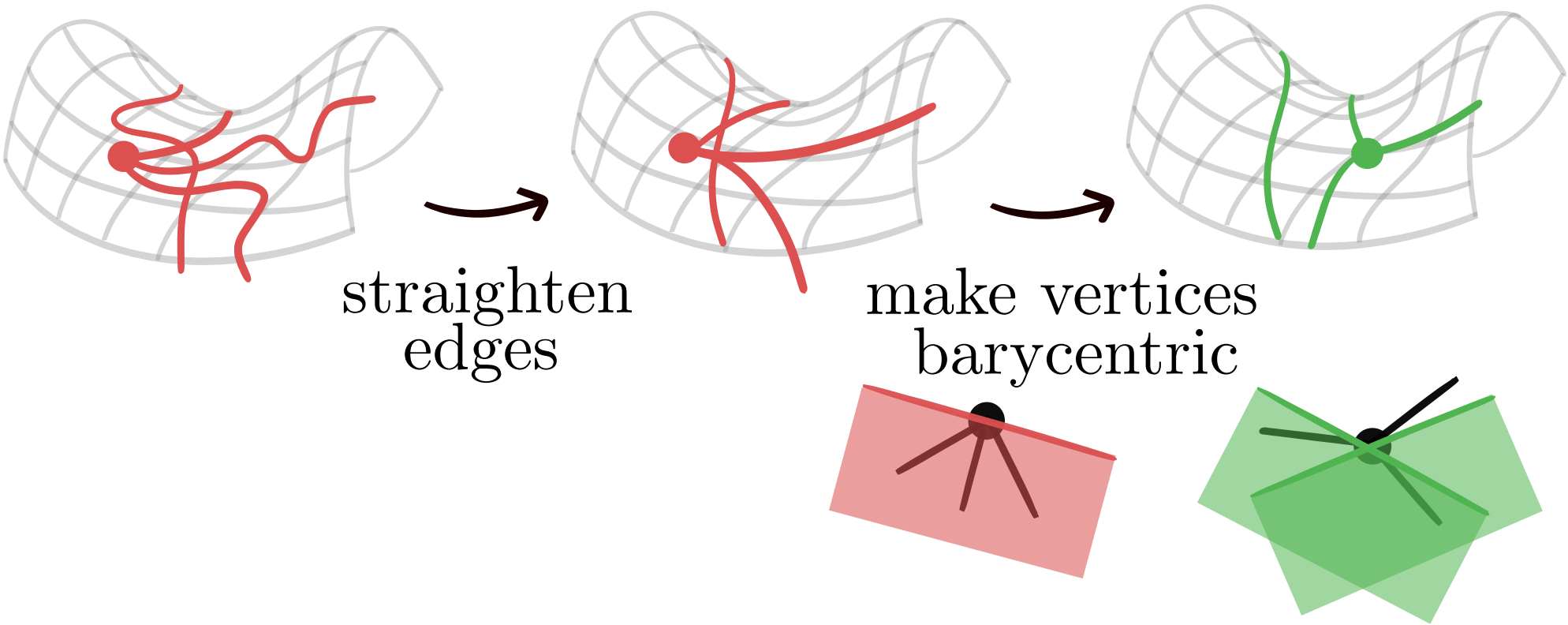
Despré and Lazarus, 2019

What about untangling graphs?

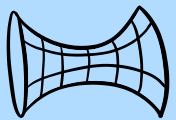
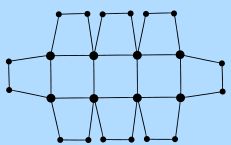
Method for untangling graphs

Tutte, 1963

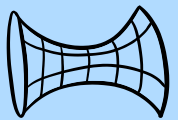
Y. Colin de Verdière, 1991



Summary

	Curves	Graphs
Method	negatively curved surface 	
Algo	system of quads 	

Our results

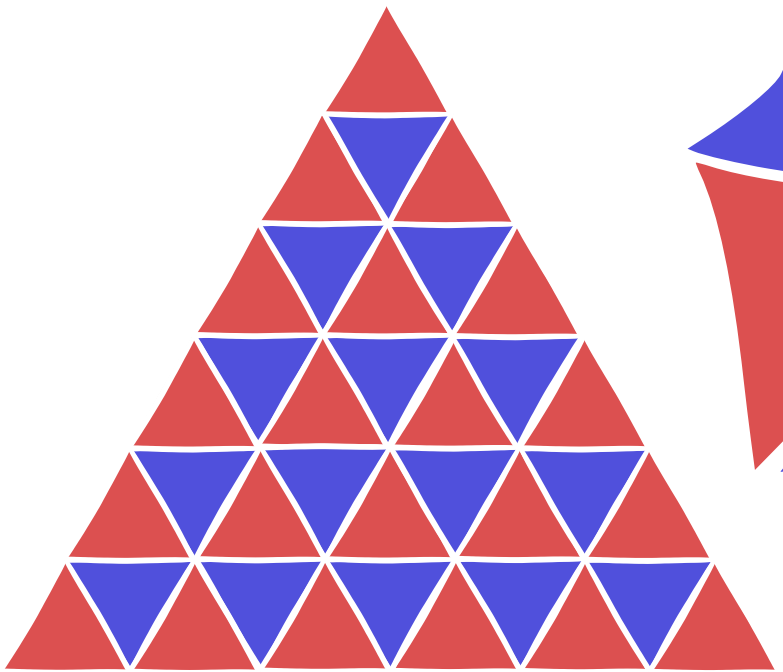
	Curves	Graphs
Method	negatively curved surface 	
Algo	reducing triangulations	
	improved algos for making curves cross minimally	first algos for untangling graphs

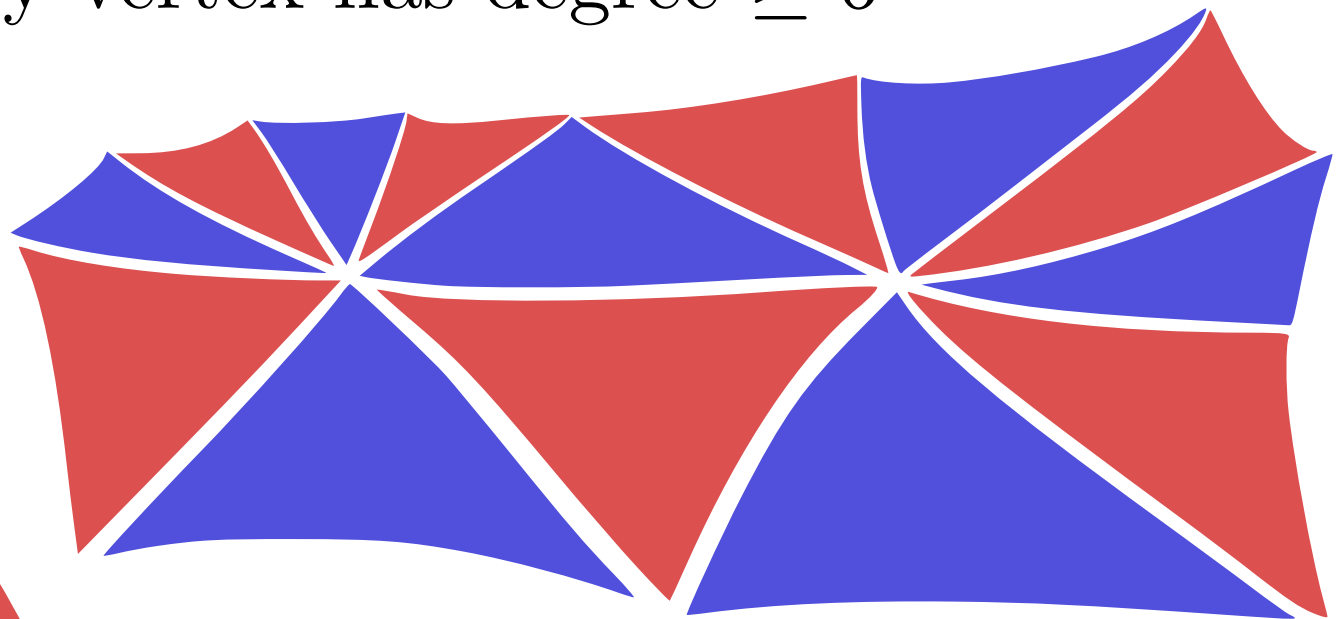
A new tool:

Reducing triangulations

Reducing triangulations

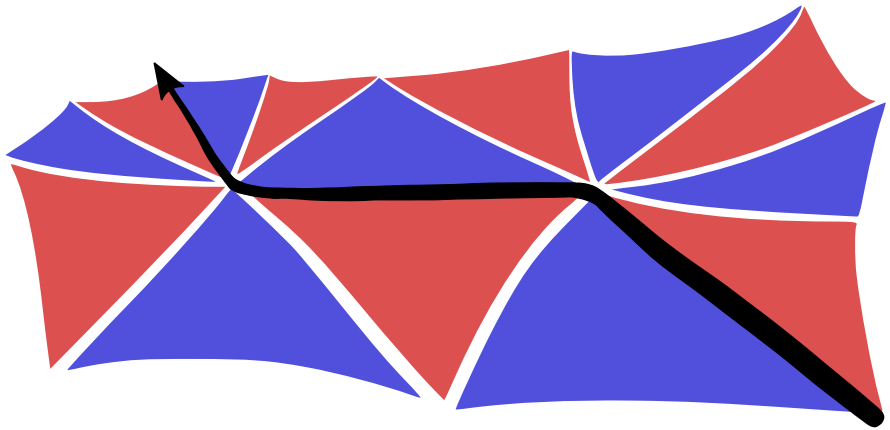
dual is bipartite and
every vertex has degree $\geq 6^*$



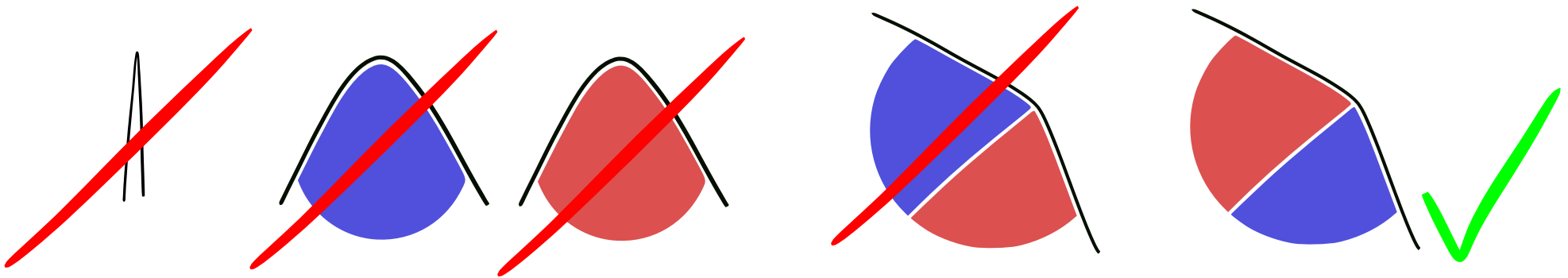


*sometimes 8

Reduced walks

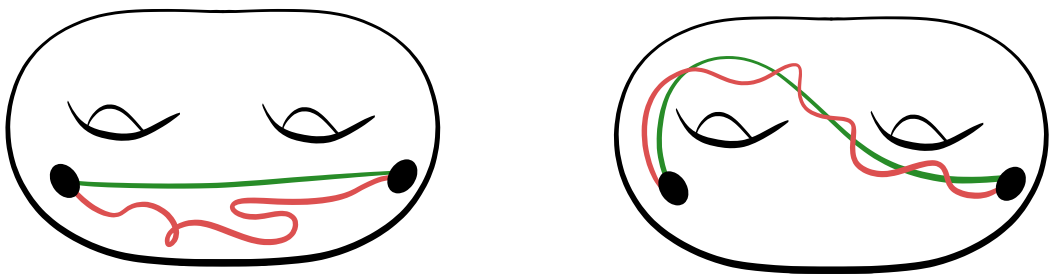


no bad turn



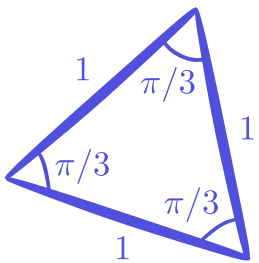
Properties of reduced walks

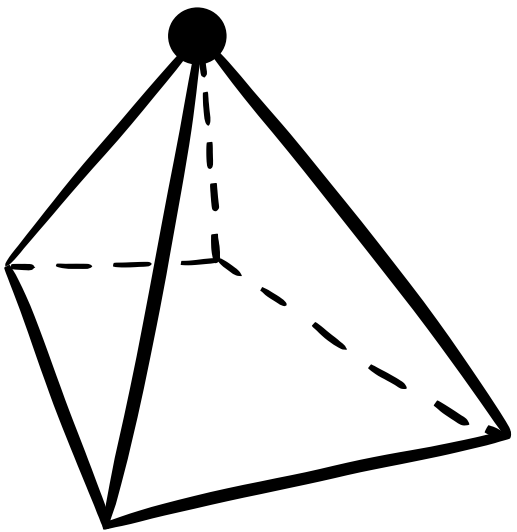
every walk can be deformed into a unique reduced walk, computable in linear time



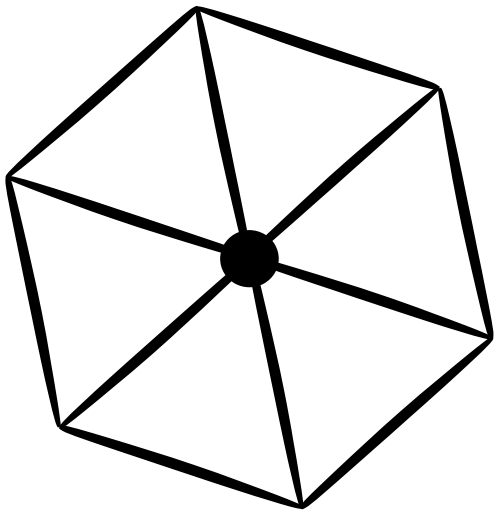
reduced walks are stable upon reversal and subwalk

Analogy:
vertex degree vs. curvature

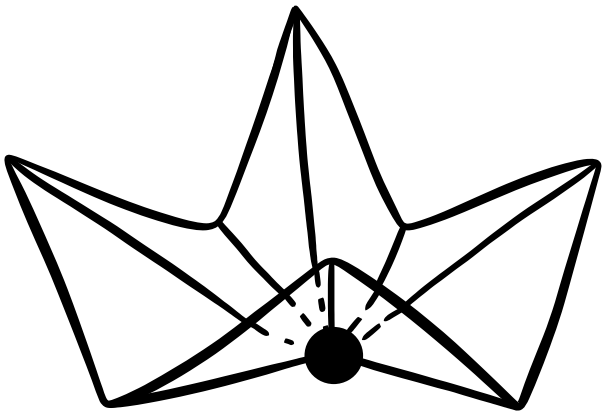




degree < 6

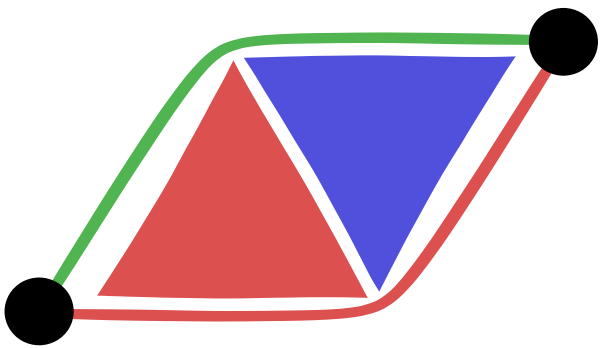


degree 6

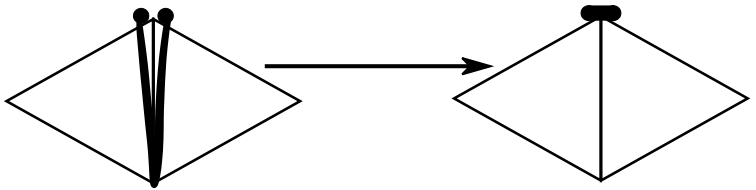


degree > 6

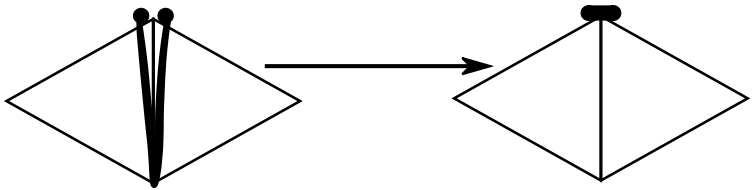
Purpose of the coloring

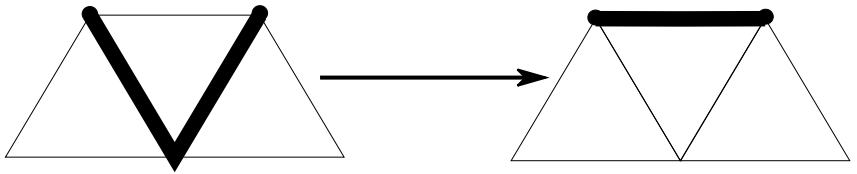


Reducing a walk

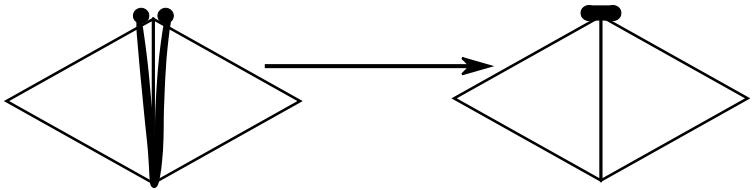


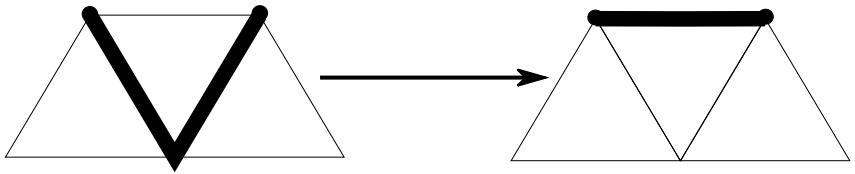
Reducing a walk

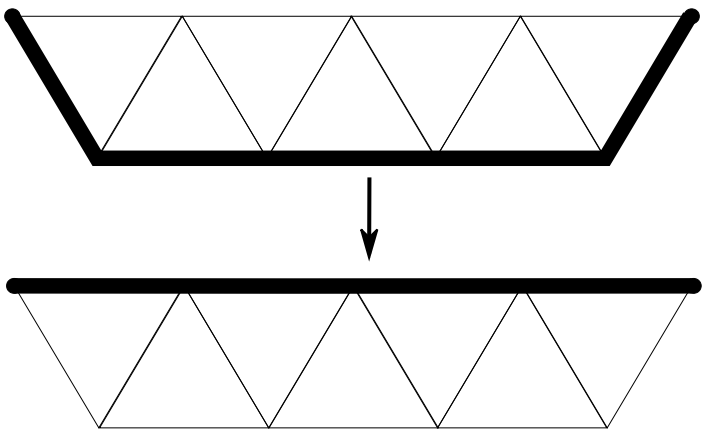




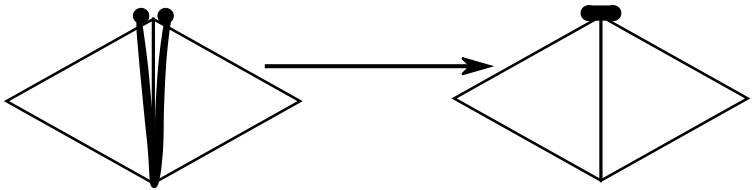
Reducing a walk

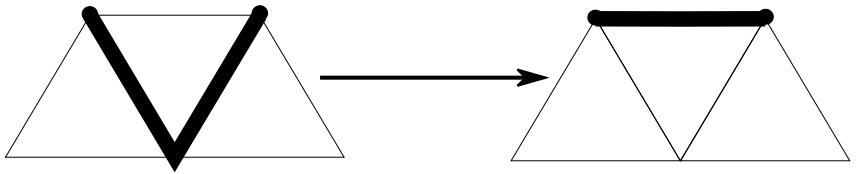


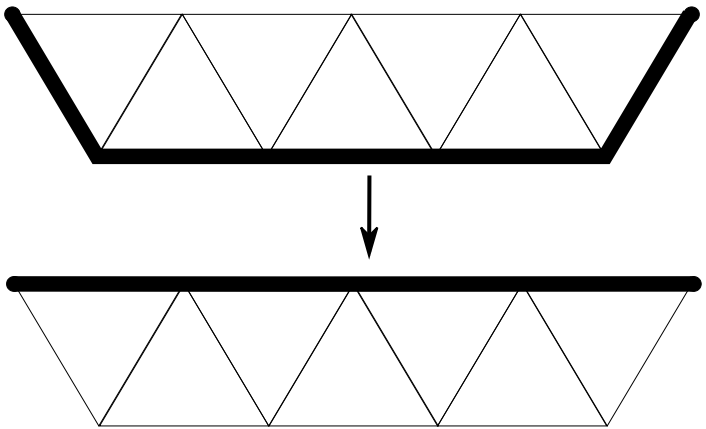


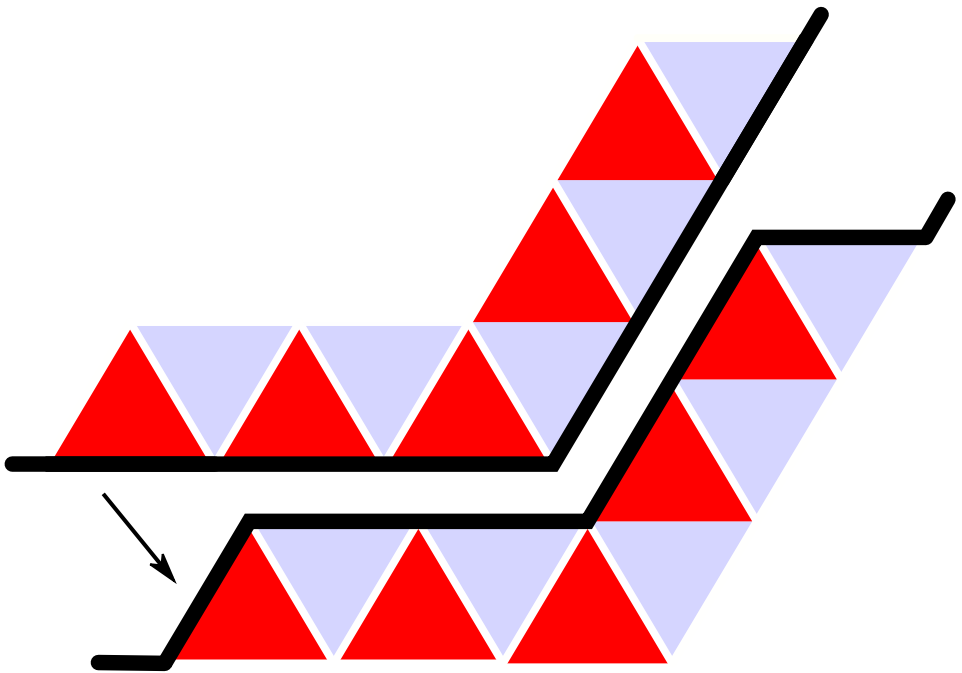


Reducing a walk

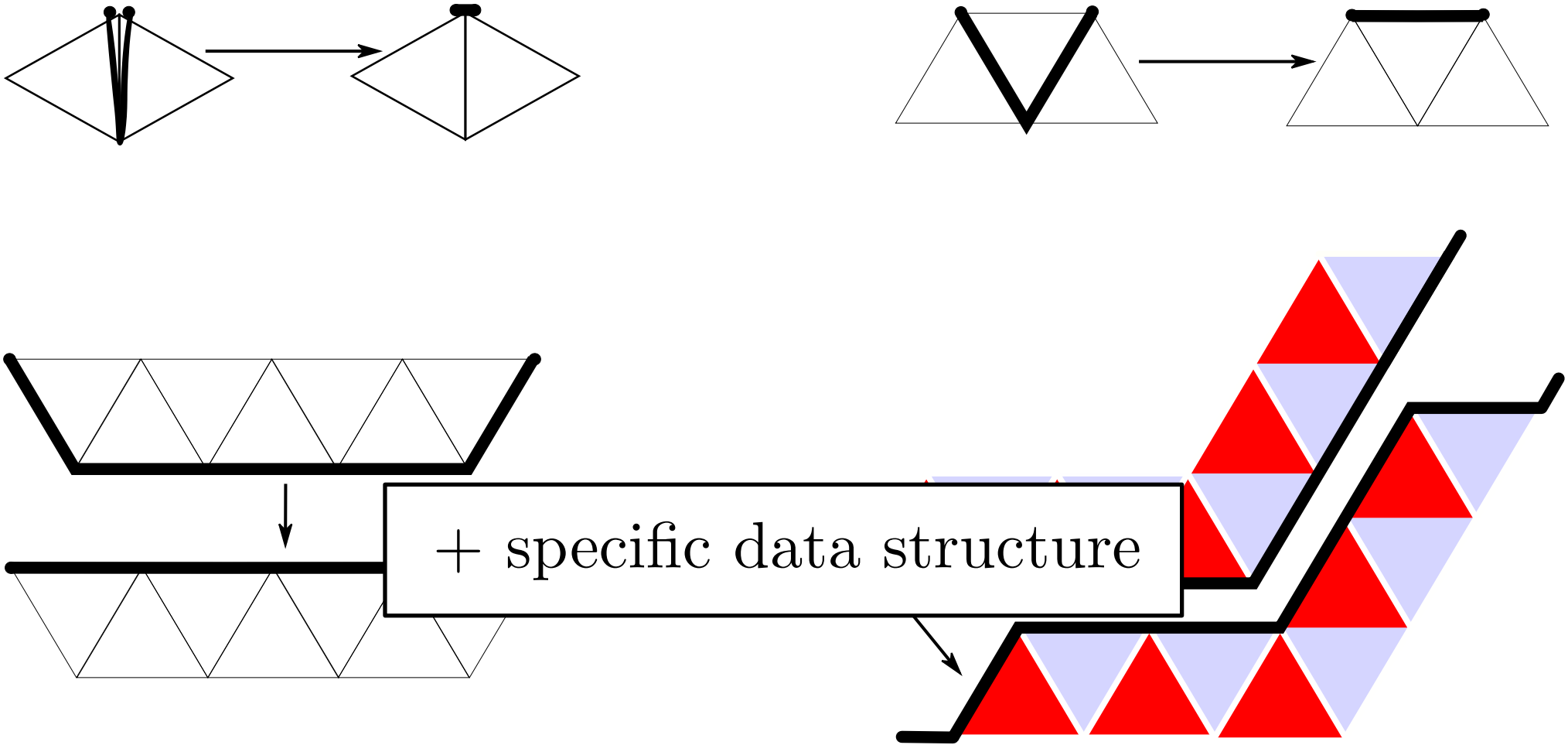




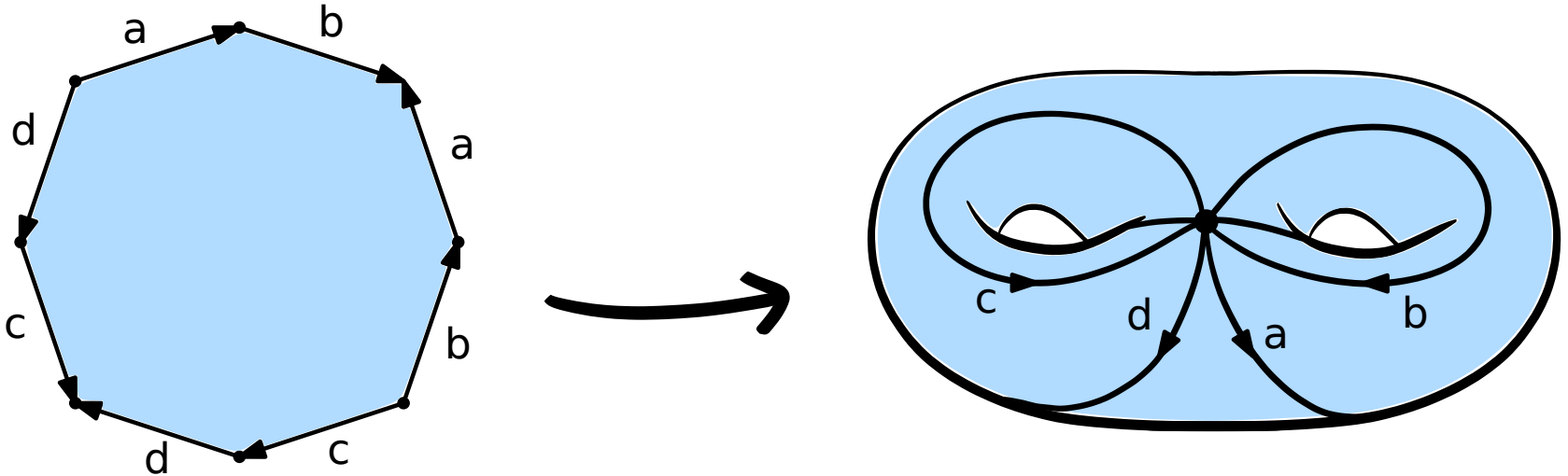




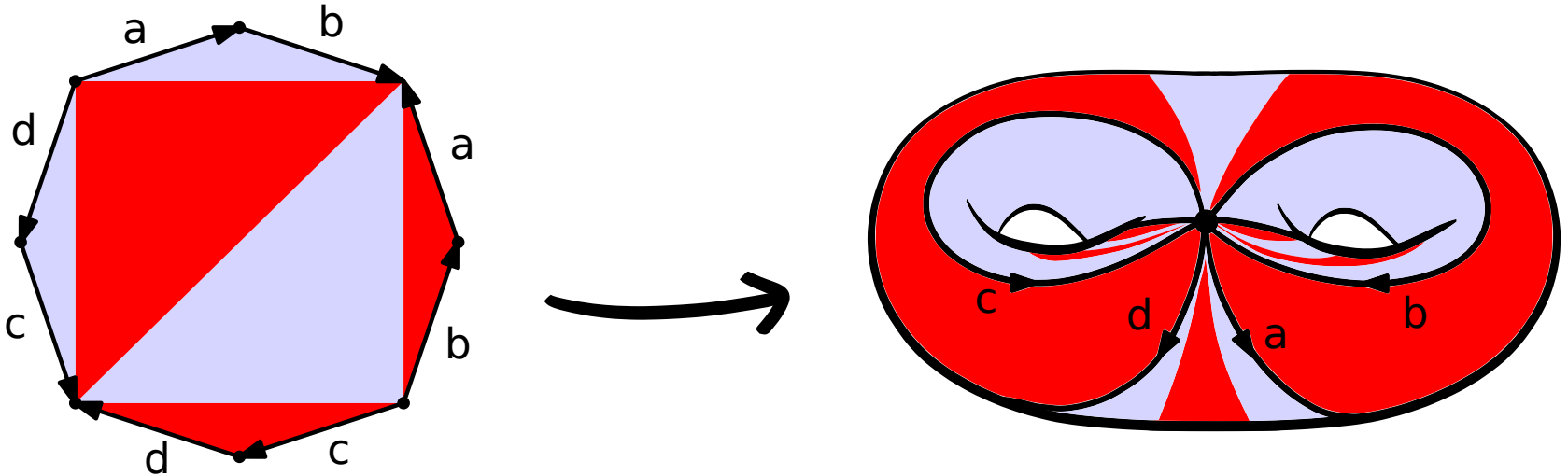
Reducing a walk



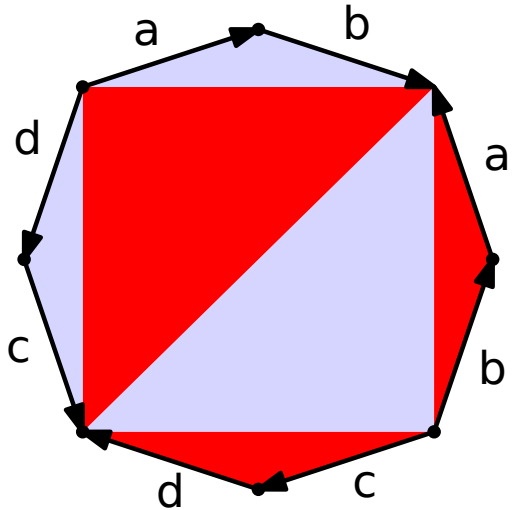
Constructing reducing triangulations



Constructing reducing triangulations

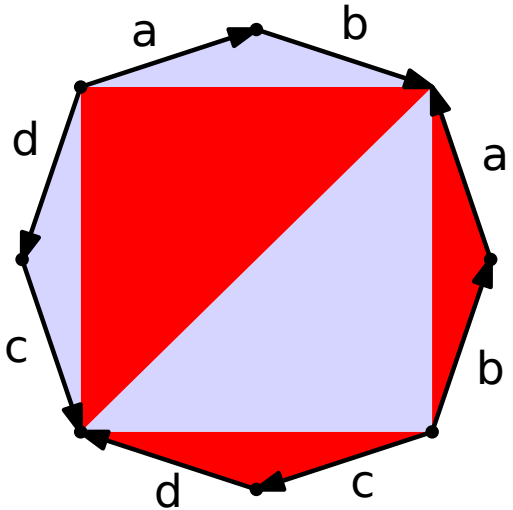


Constructing reducing triangulations



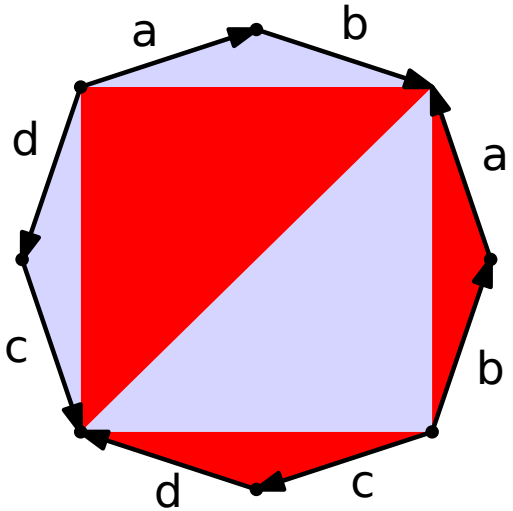
Constructing reducing triangulations

genus 2

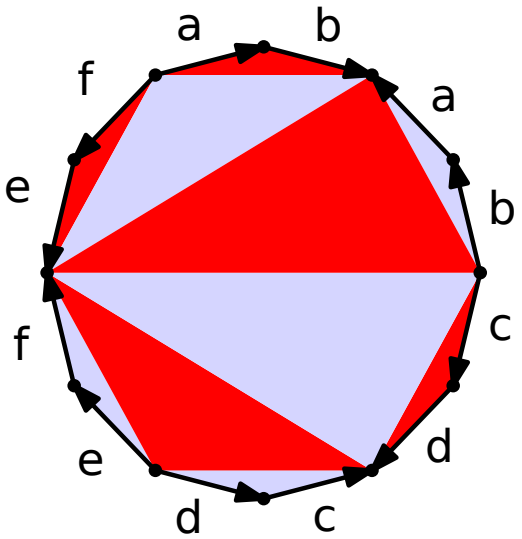


Constructing reducing triangulations

genus 2

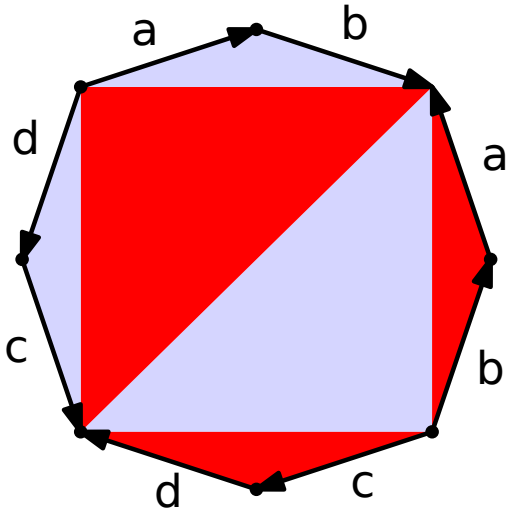


genus 3

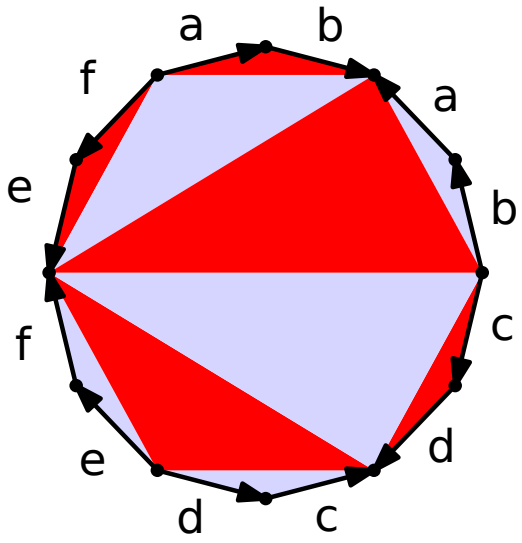


Constructing reducing triangulations

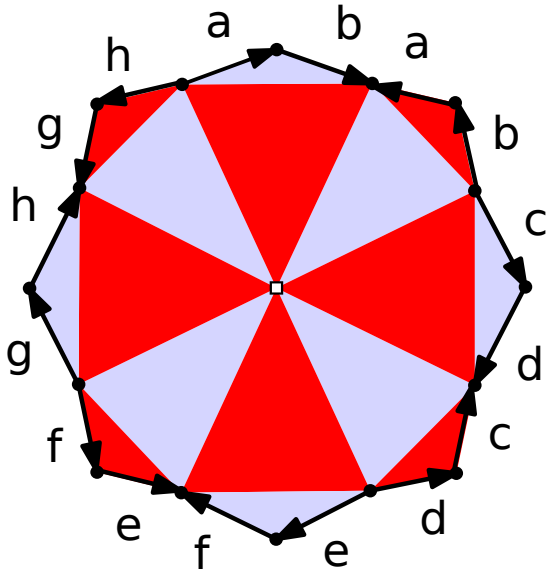
genus 2



genus 3



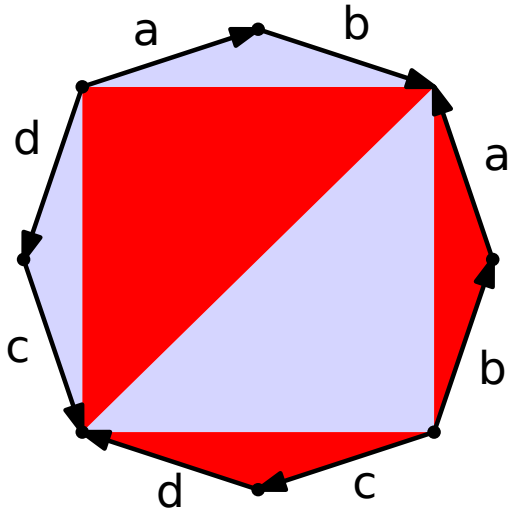
genus 4



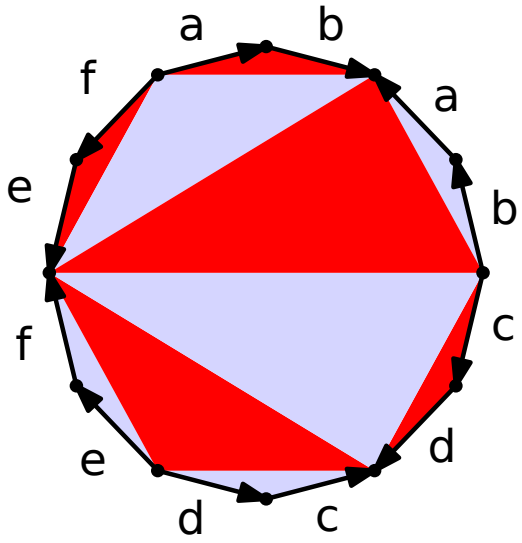
. . .

Constructing reducing triangulations

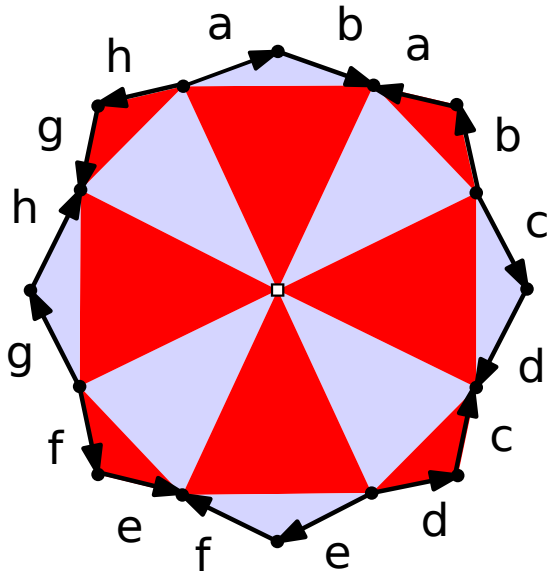
genus 2



genus 3

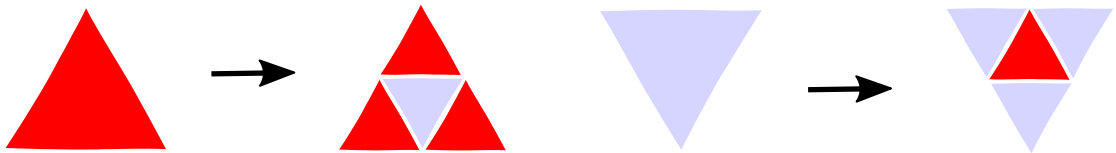


genus 4



. . .

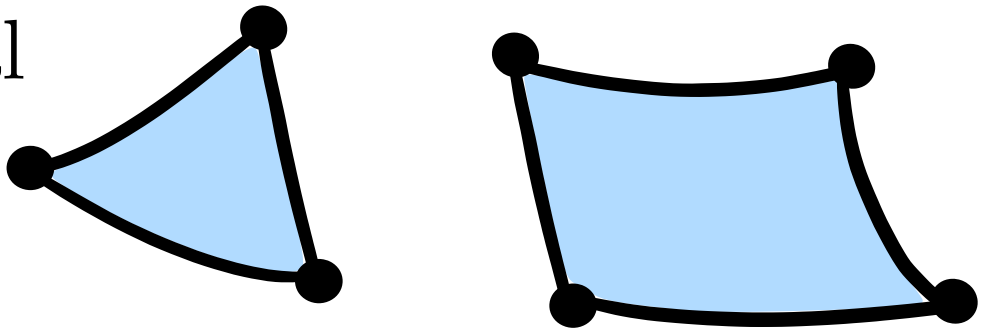
then subdivide at will:

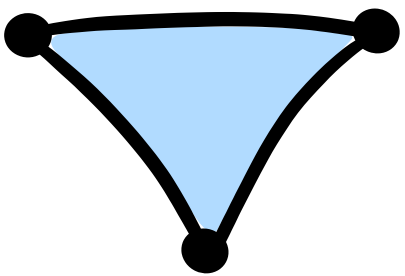


Untangling graphs using reducing triangulations

Input

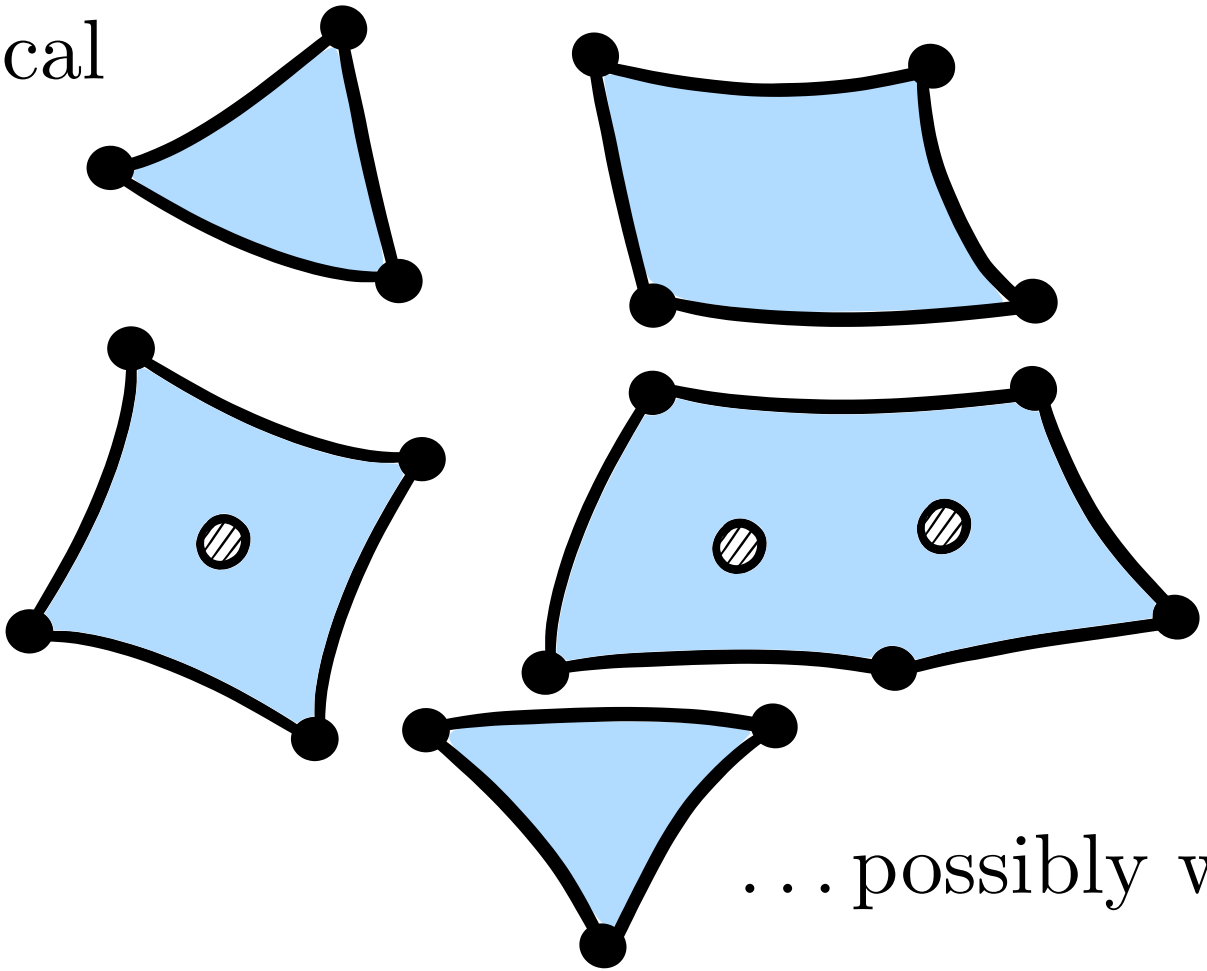
take topological polygons...





Input

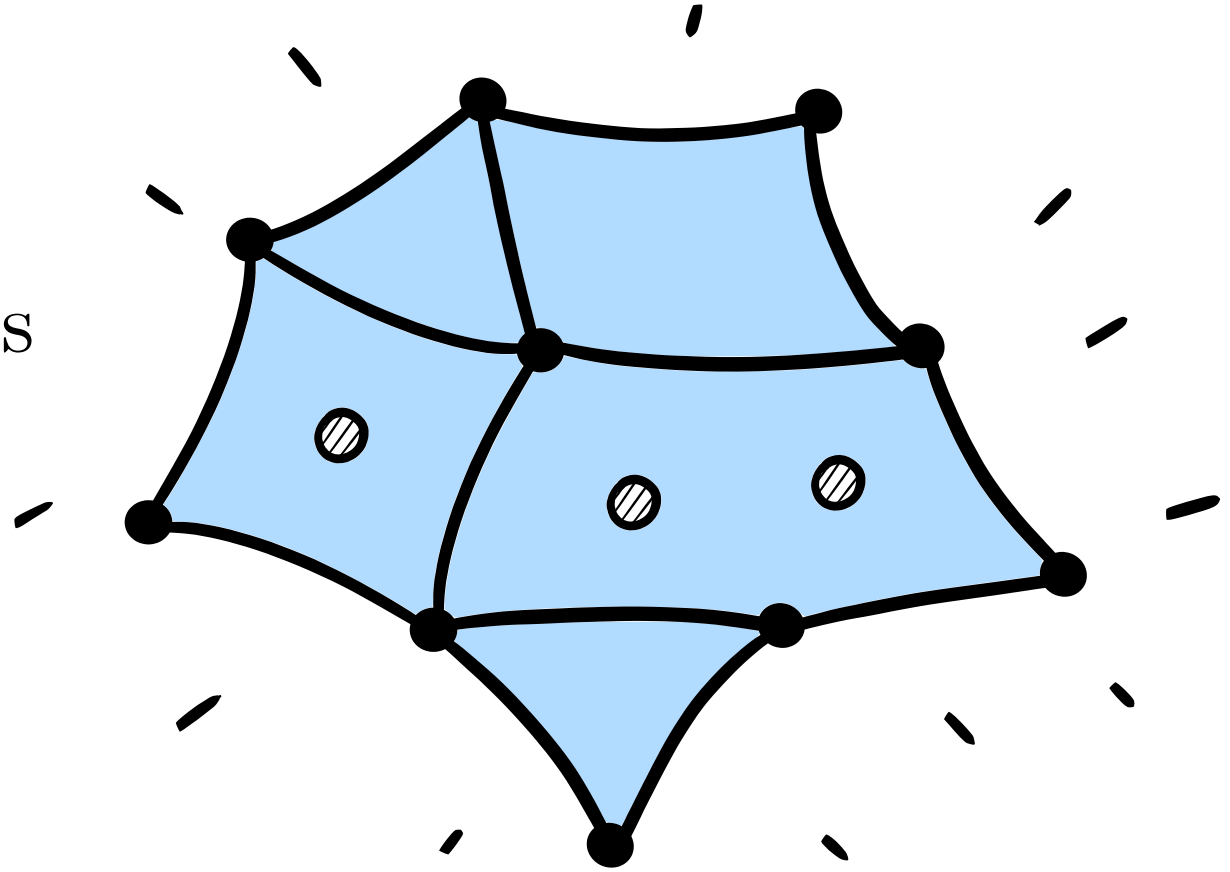
take topological polygons...



...possibly with holes

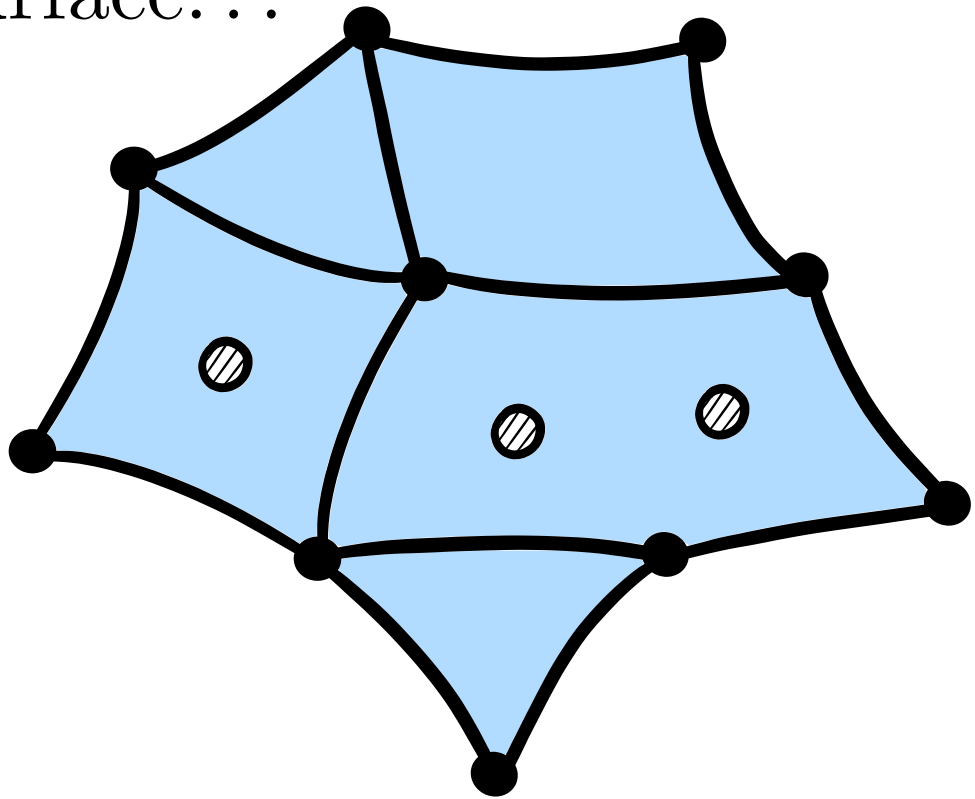
Input

glue edges



Input

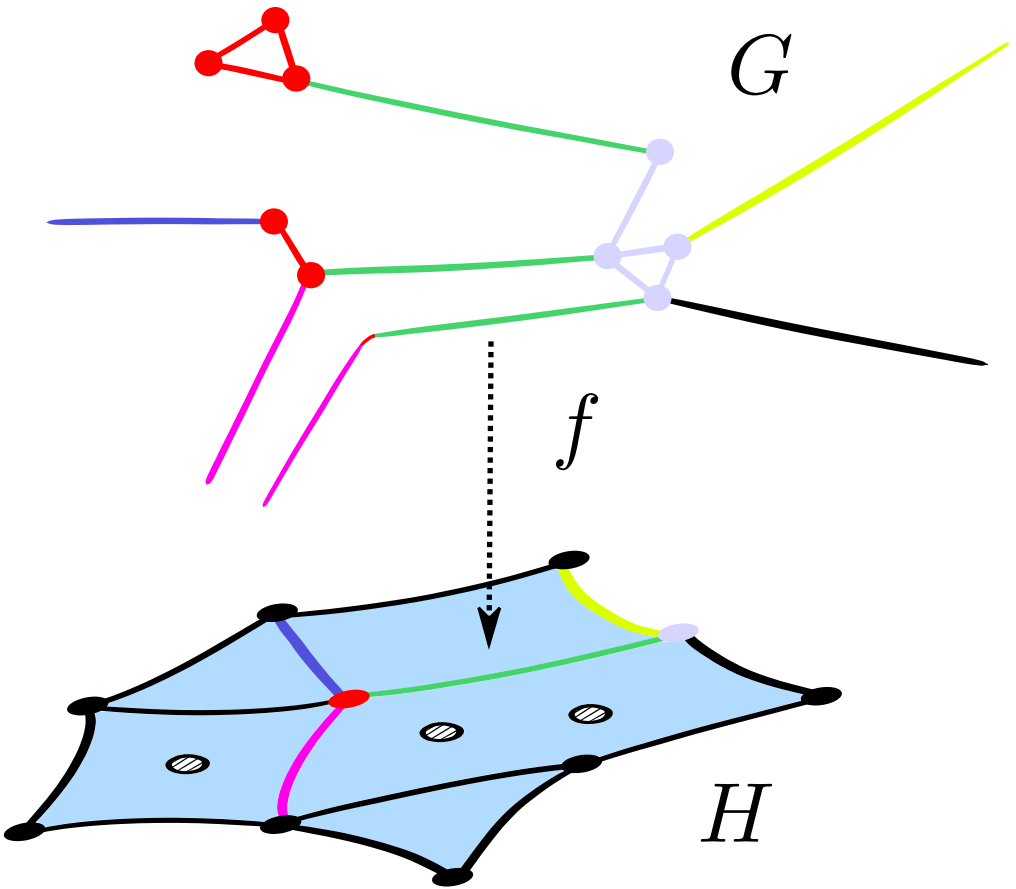
that encodes the surface...



...and a graph H on it

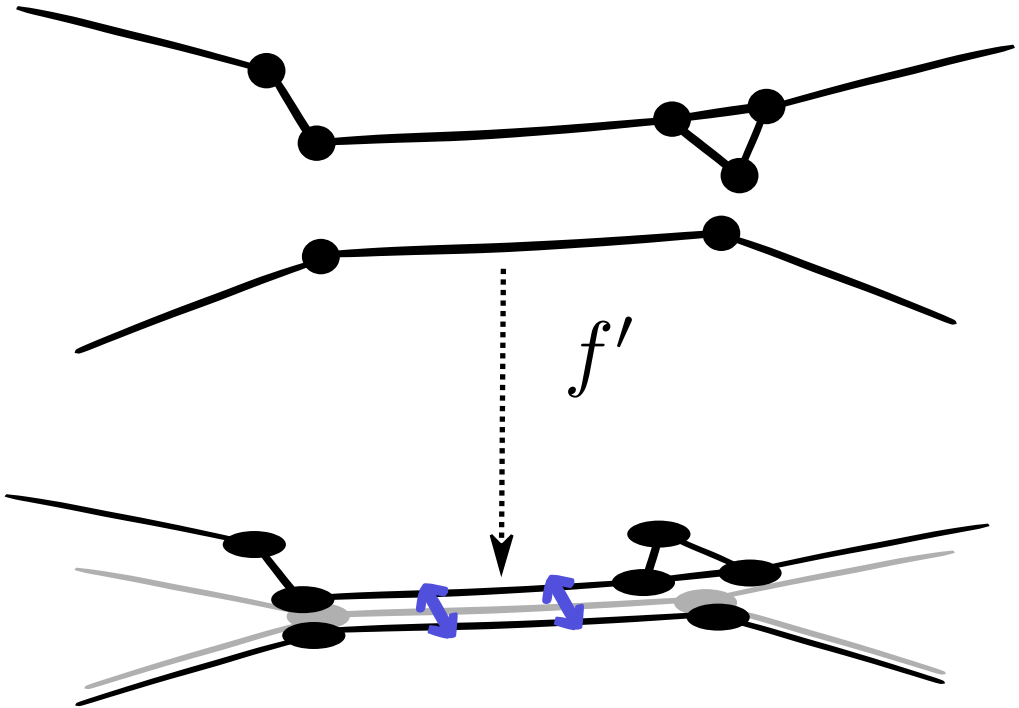
Input

draw G in H



Output

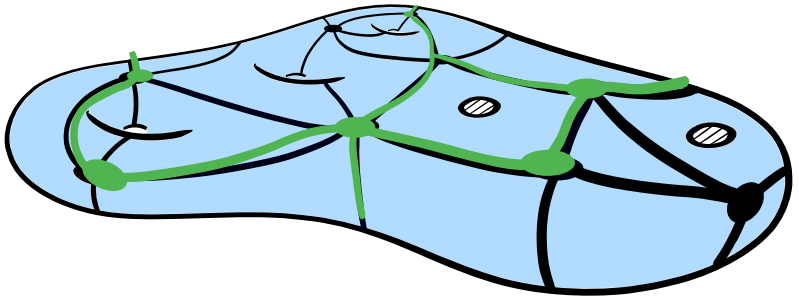
weak embedding: drawing f' that can be untangled by infinitesimal perturbation



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

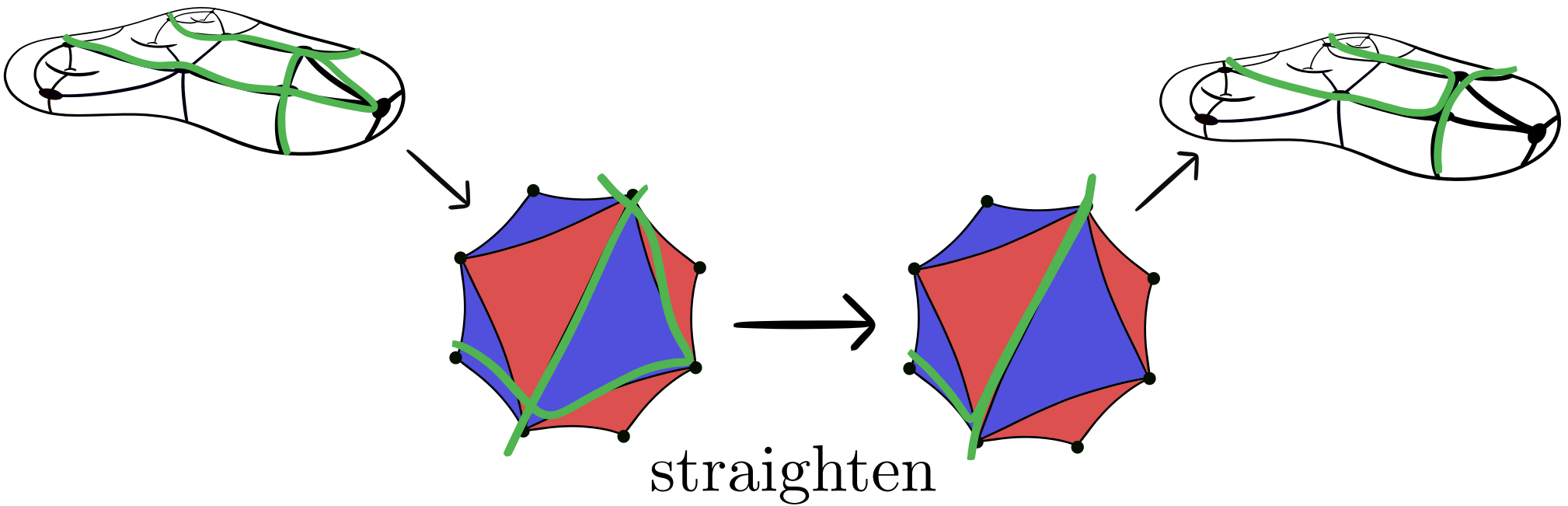
s : genus + # holes

Colin de Verdière, Despré, D., 2023

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

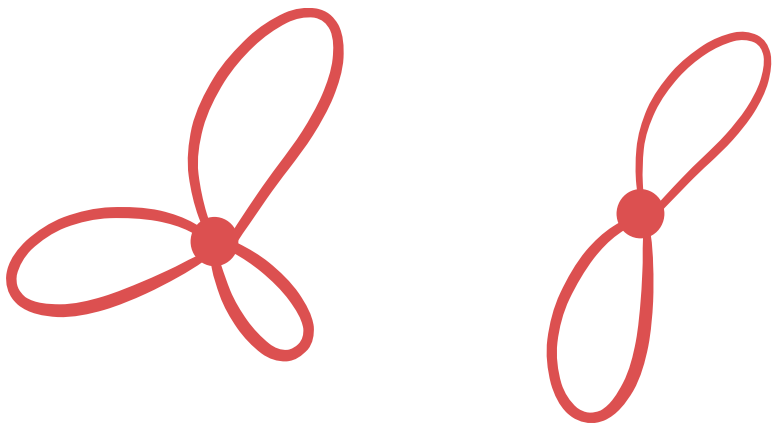
If so, we can compute a weak embedding homotopic to f in additional $O(s^2 m\, n^2)$ time.

Algorithm overview



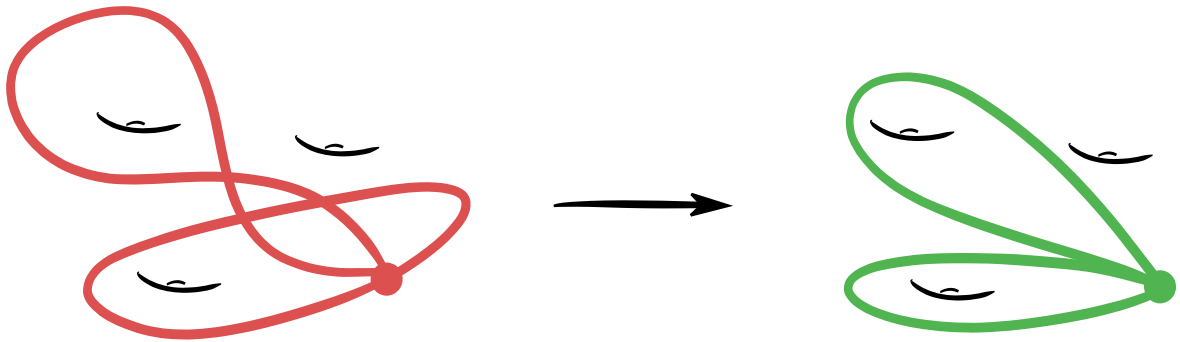
Straightening a graph

Straightening a ~~graph~~ loop graph



Straightening a loop graph

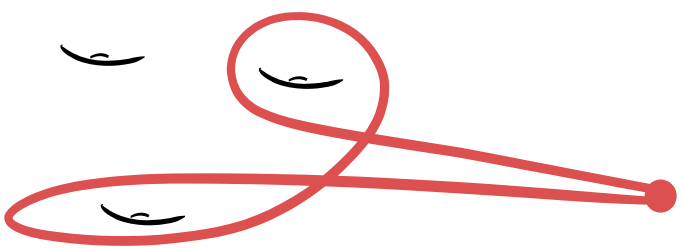
First attempt: reduce the loops, vertex fixed



Straightening a loop graph

First attempt: reduce the loops, vertex fixed

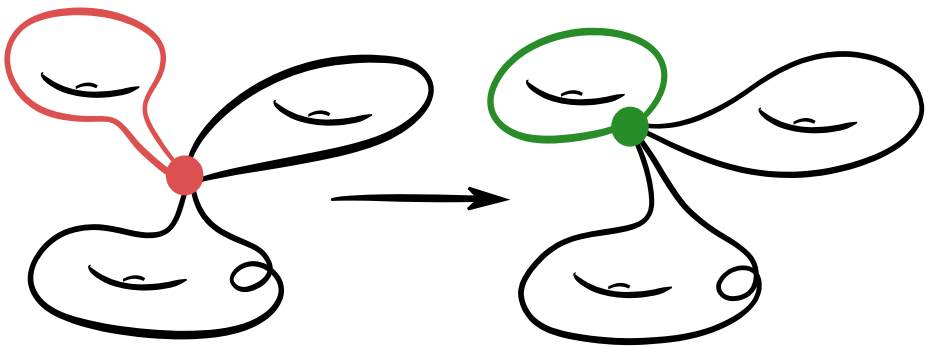
Problem:



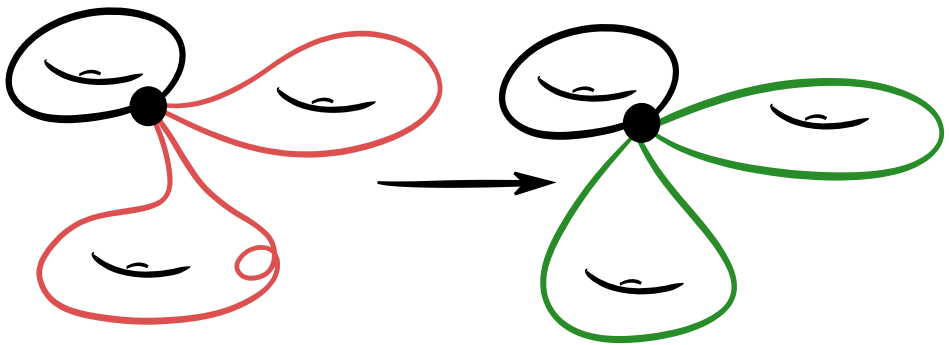
→ vertex must move

Straightening a loop graph

Solution:



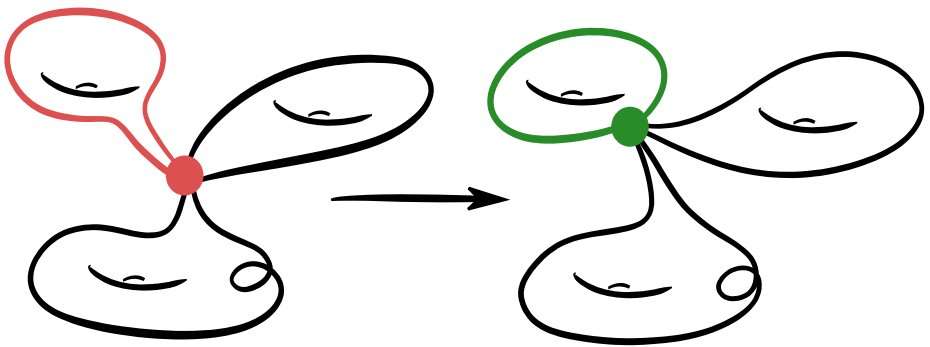
1. Reduce 1 loop cyclically



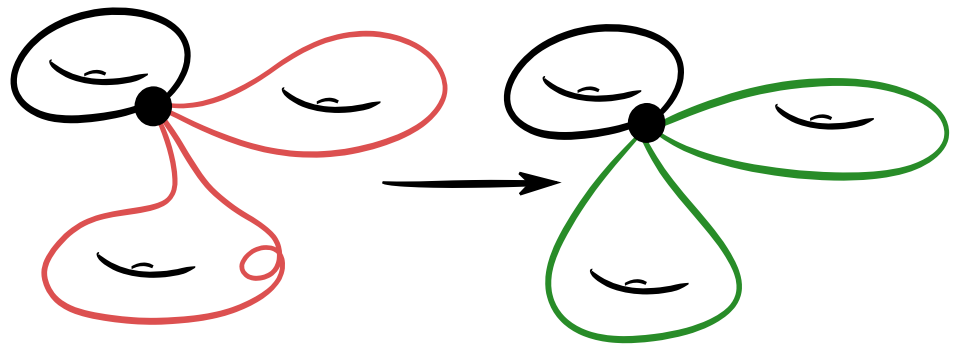
2. Reduce the other loops linearly

Straightening a loop graph

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



1. Reduce 1 loop cyclically

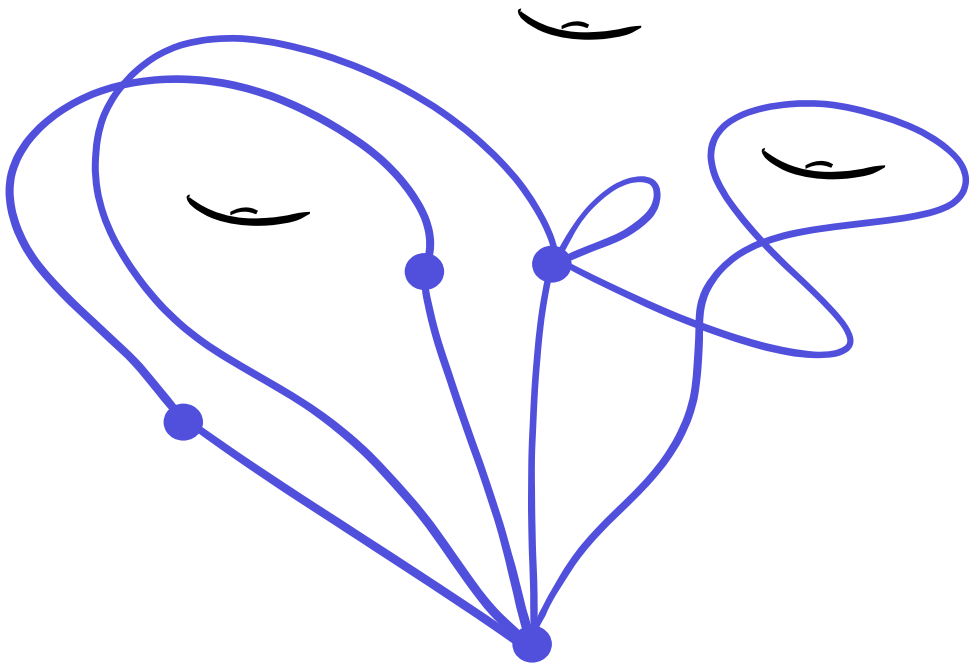


2. Reduce the other loops linearly

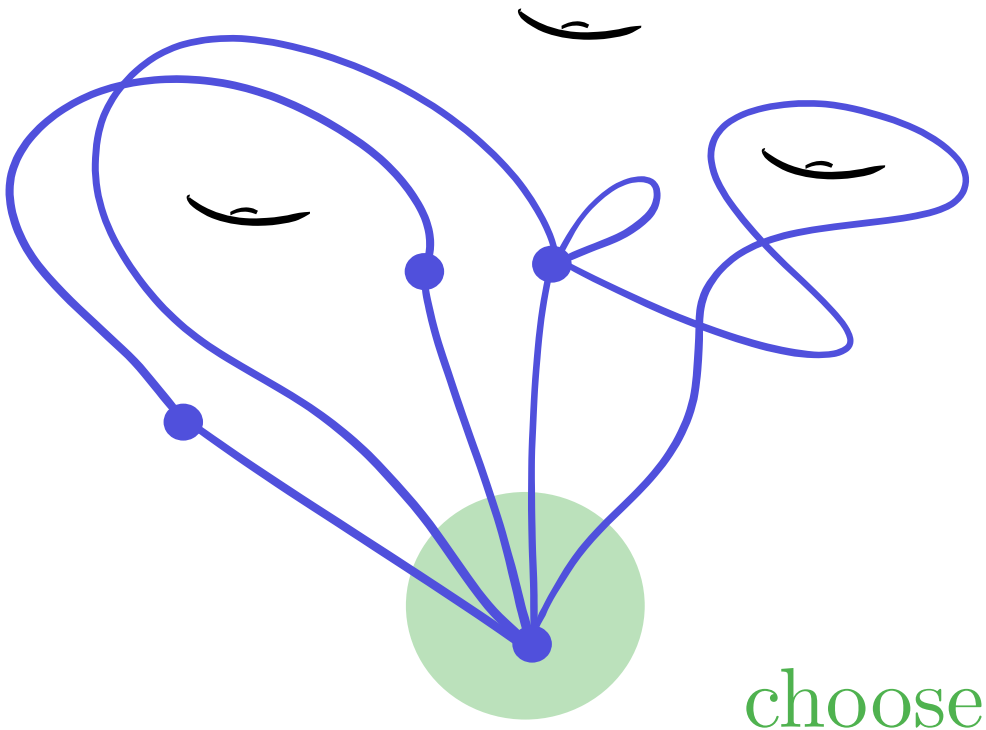
Straightening a graph

Straightening a graph

(connected, say)

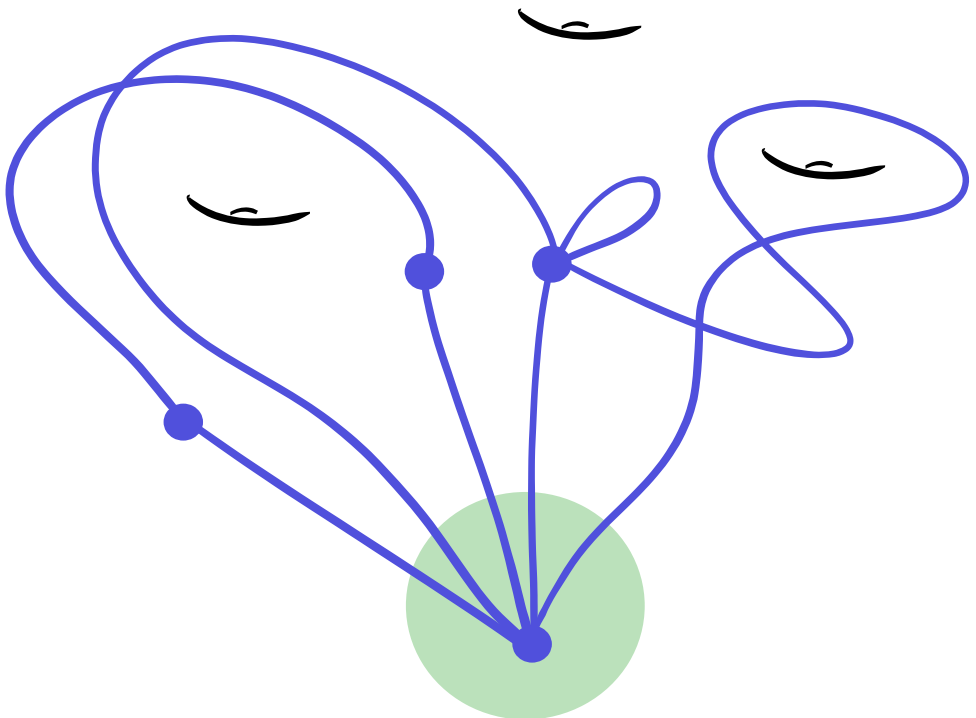


Straightening a graph

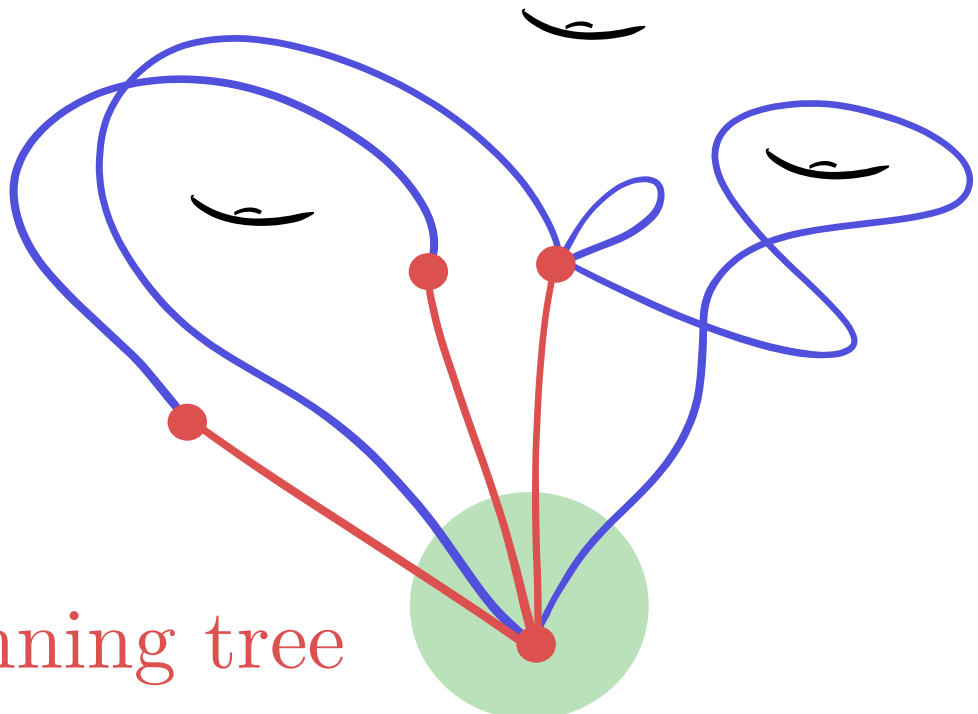


choose a base vertex

Straightening a graph

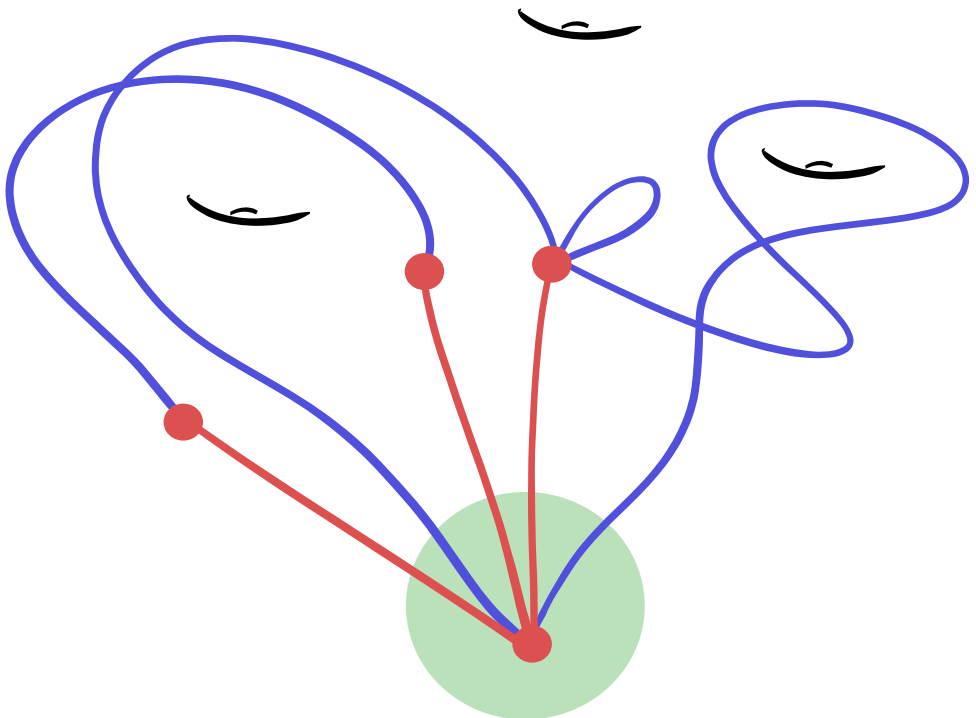


Straightening a graph



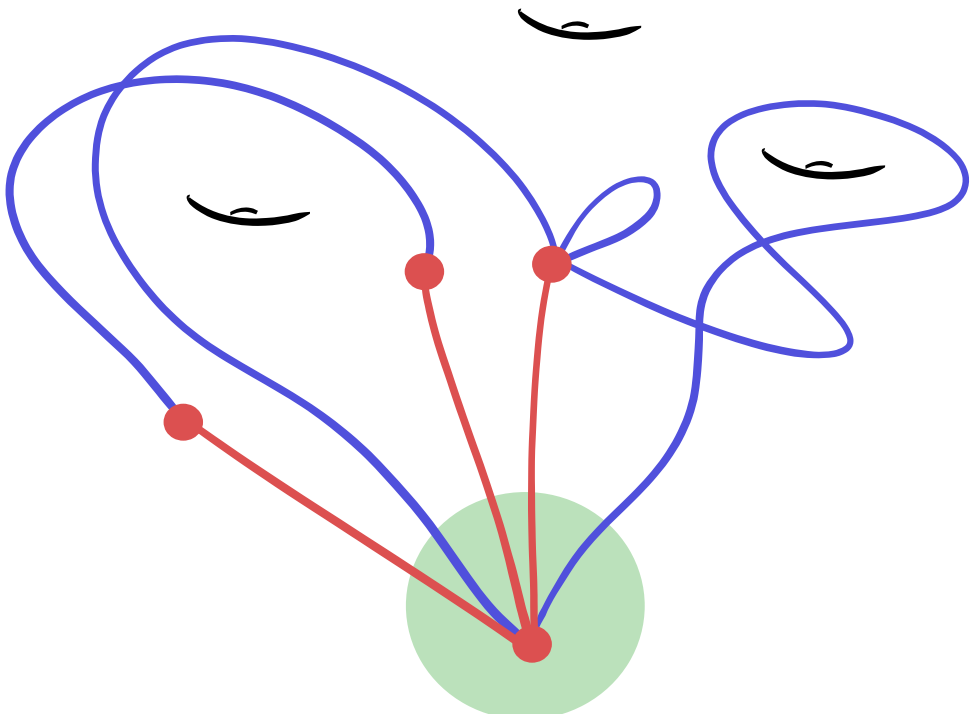
choose a spanning tree

Straightening a graph



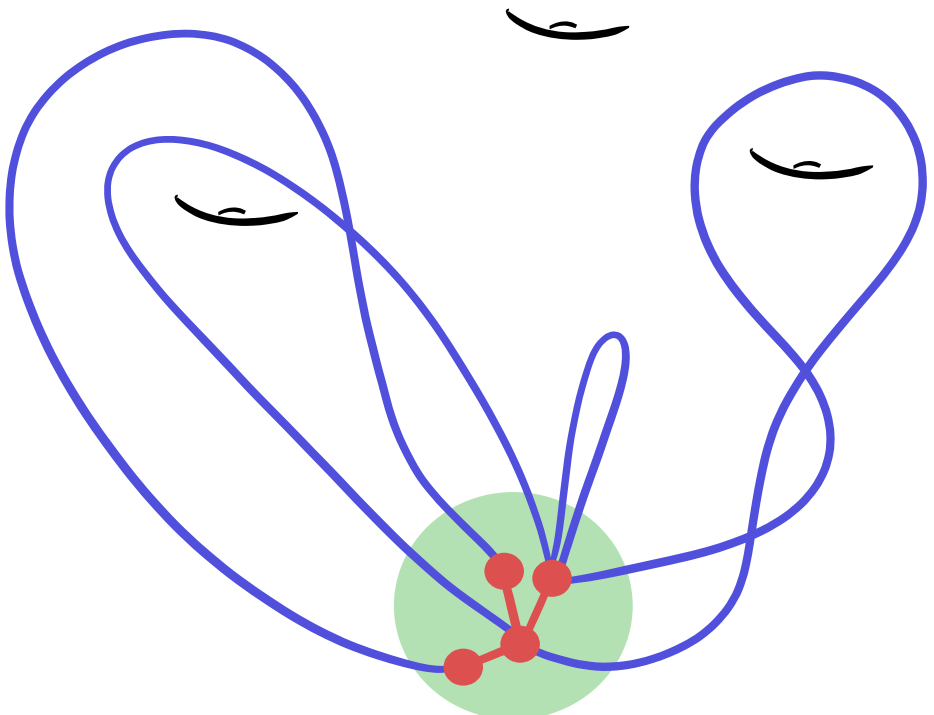
Straightening a graph

contract the spanning tree

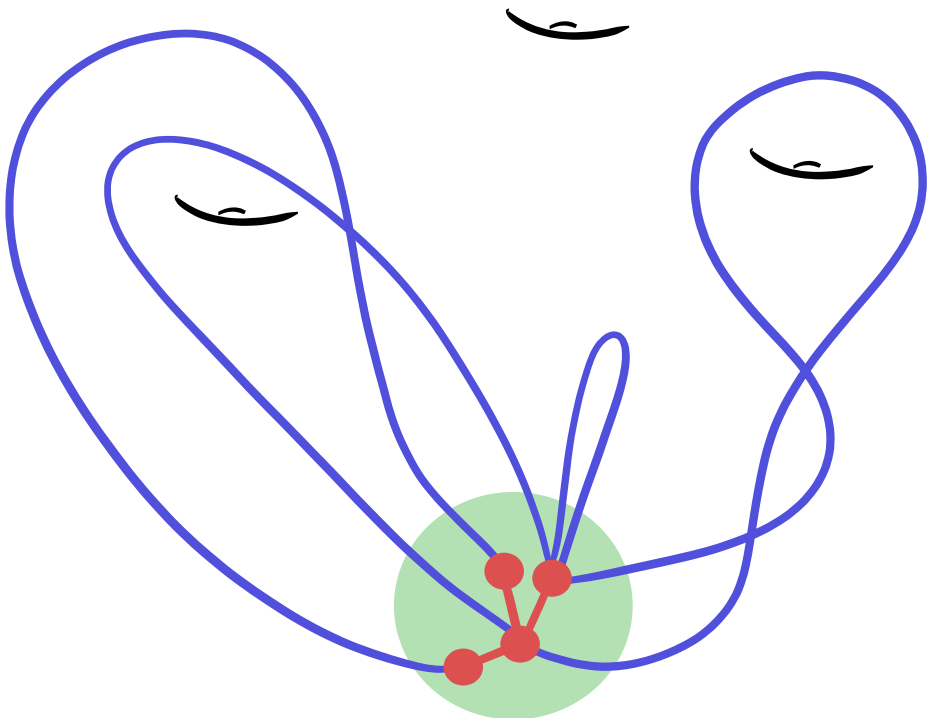


Straightening a graph

contract the spanning tree

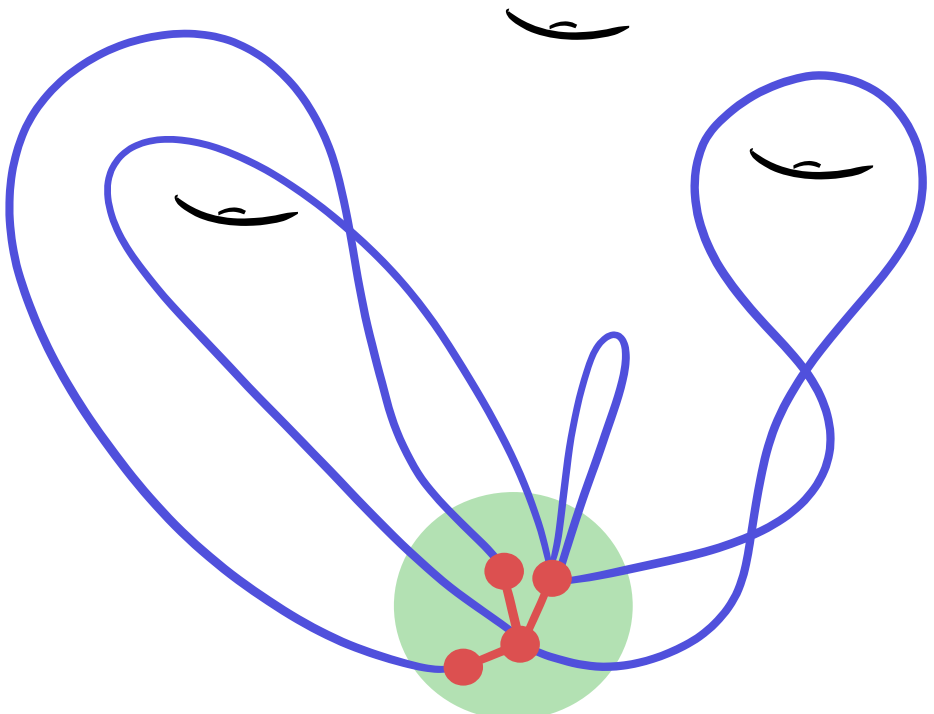


Straightening a graph



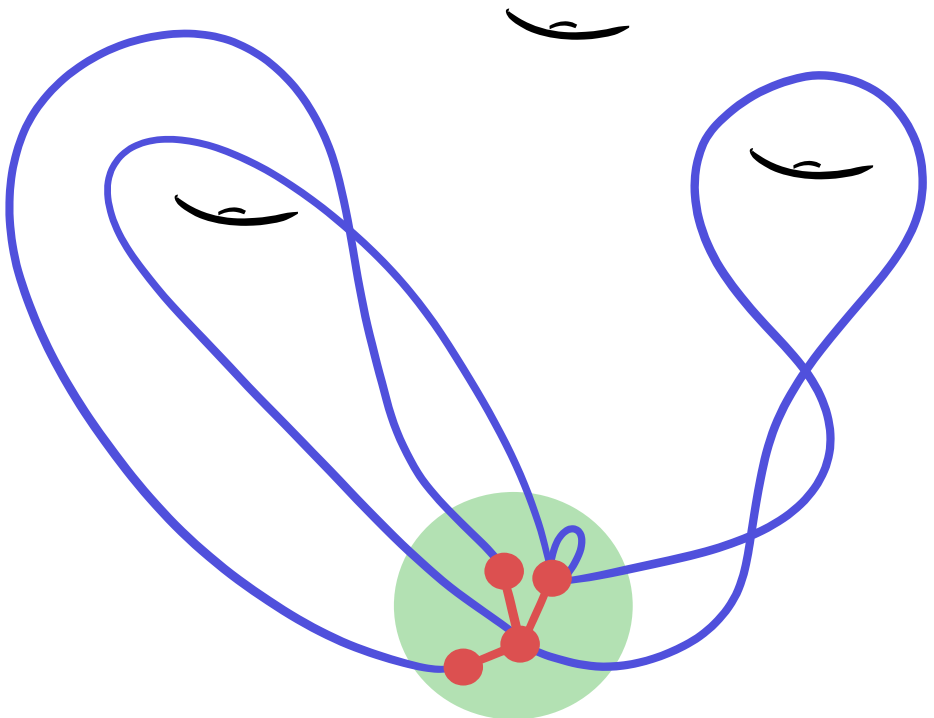
Straightening a graph

contract and bundle the other edges when possible



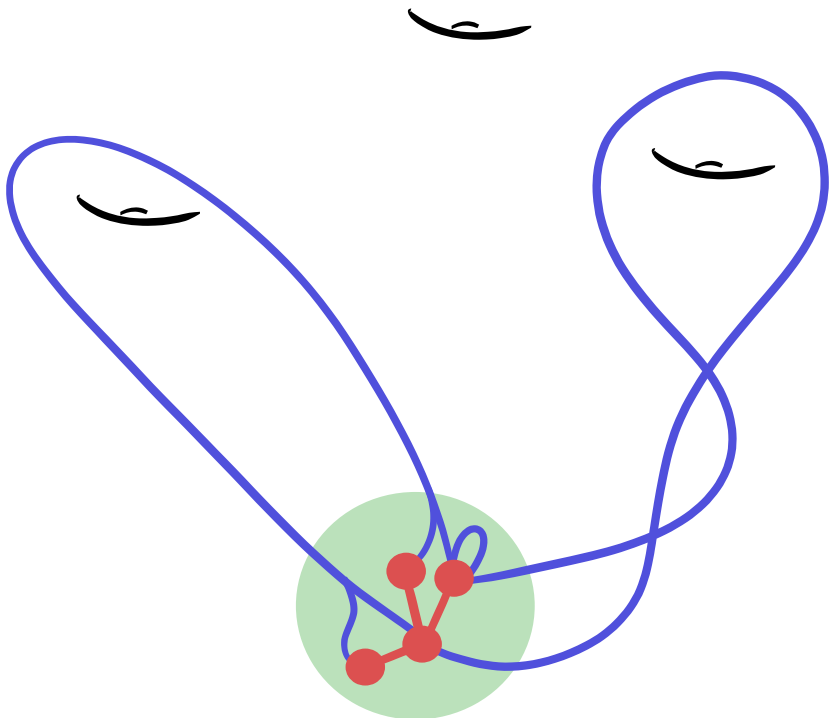
Straightening a graph

contract and bundle the other edges when possible

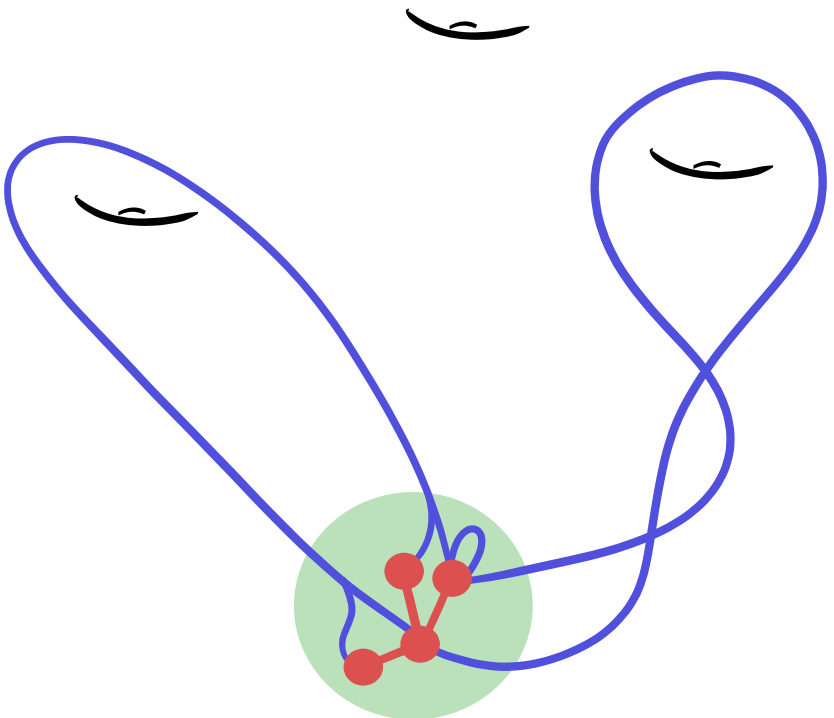


Straightening a graph

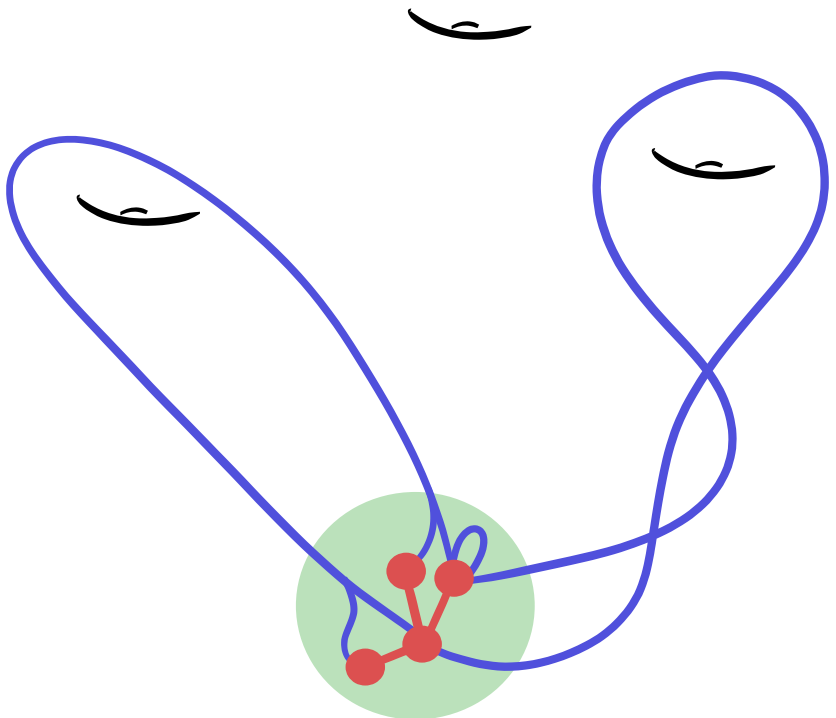
contract and bundle the other edges when possible



Straightening a graph

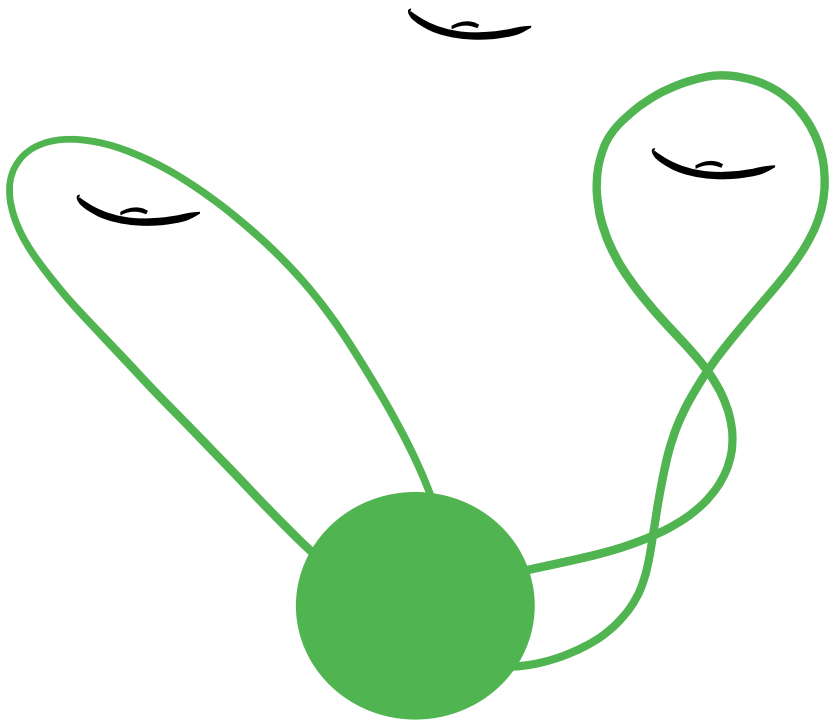


Straightening a graph



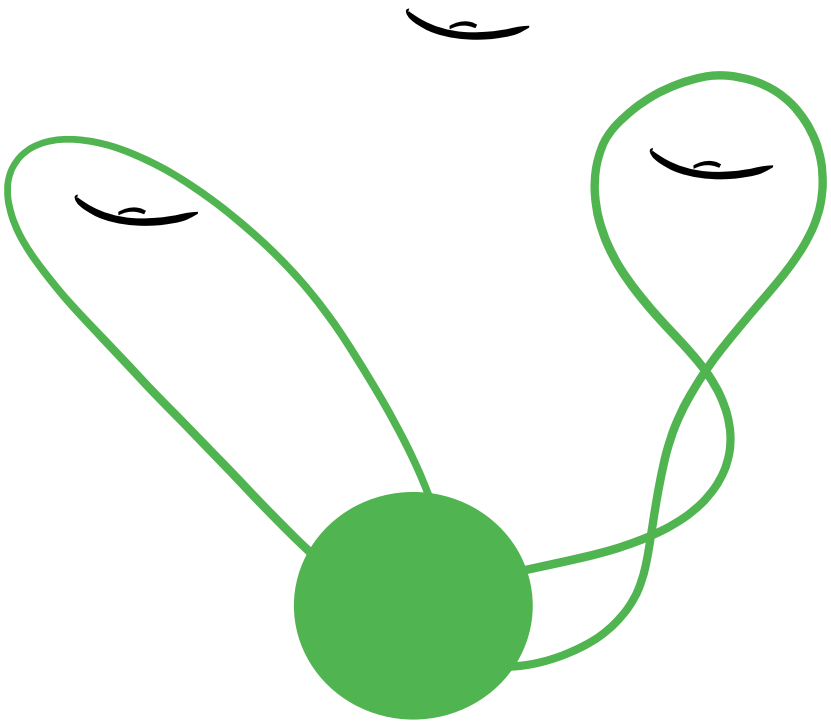
consider the associated loop graph

Straightening a graph



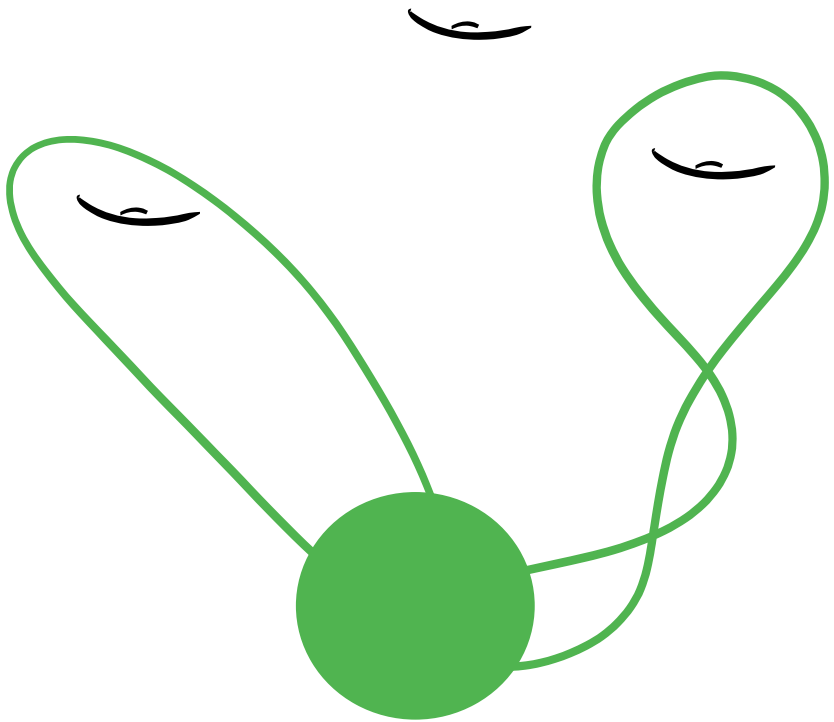
consider the associated loop graph

Straightening a graph



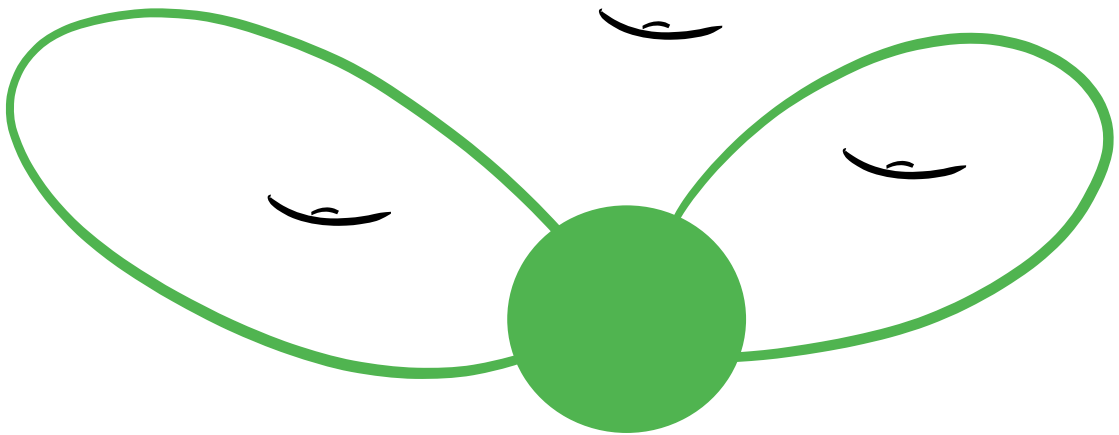
Straightening a graph

straighten the loop graph

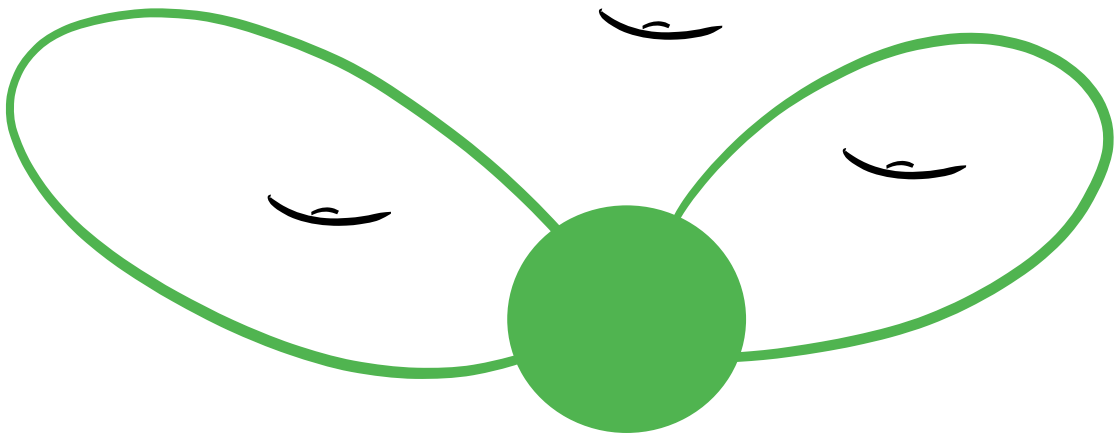


Straightening a graph

straighten the loop graph

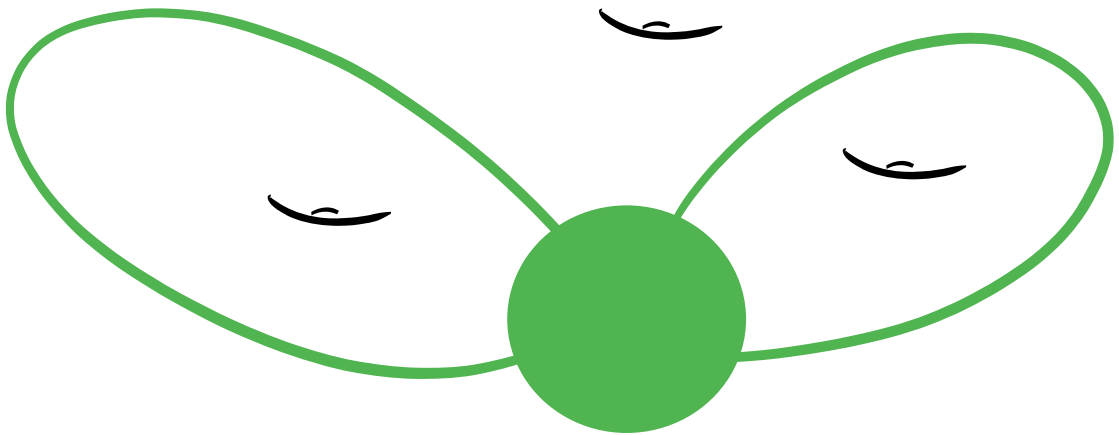


Straightening a graph



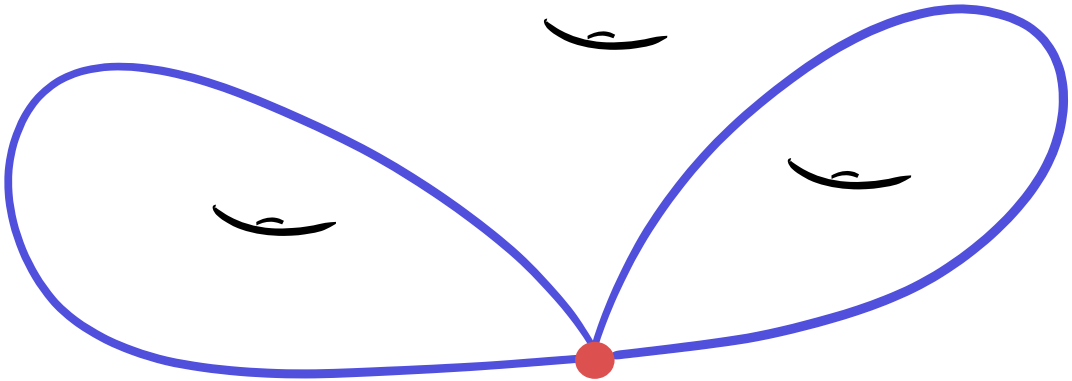
Straightening a graph

forget about the loop graph

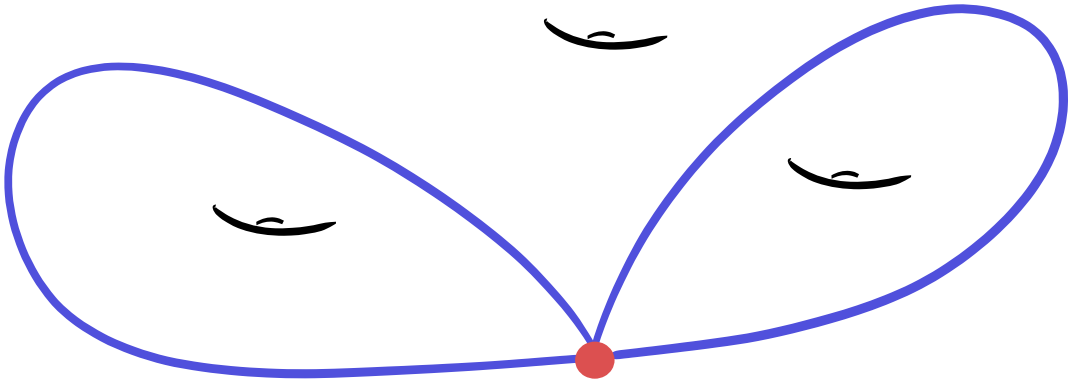


Straightening a graph

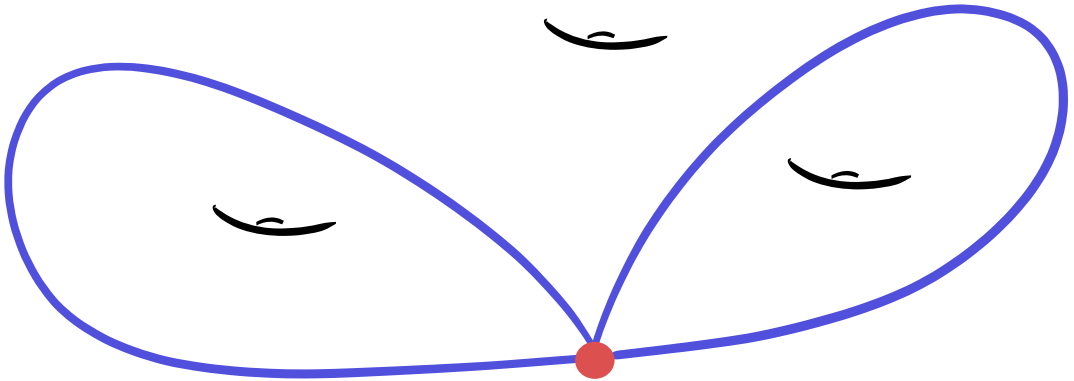
forget about the loop graph



Straightening a graph

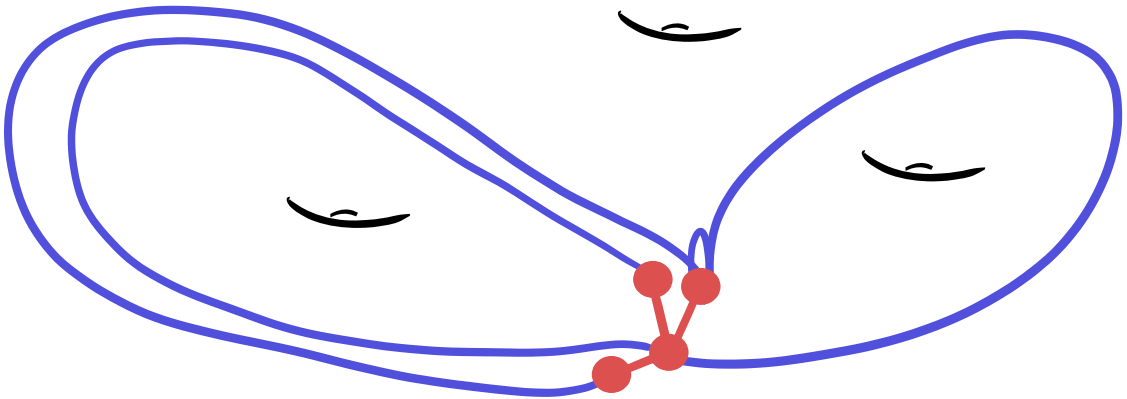


Straightening a graph



A straightened graph is a weak embedding or cannot be untangled

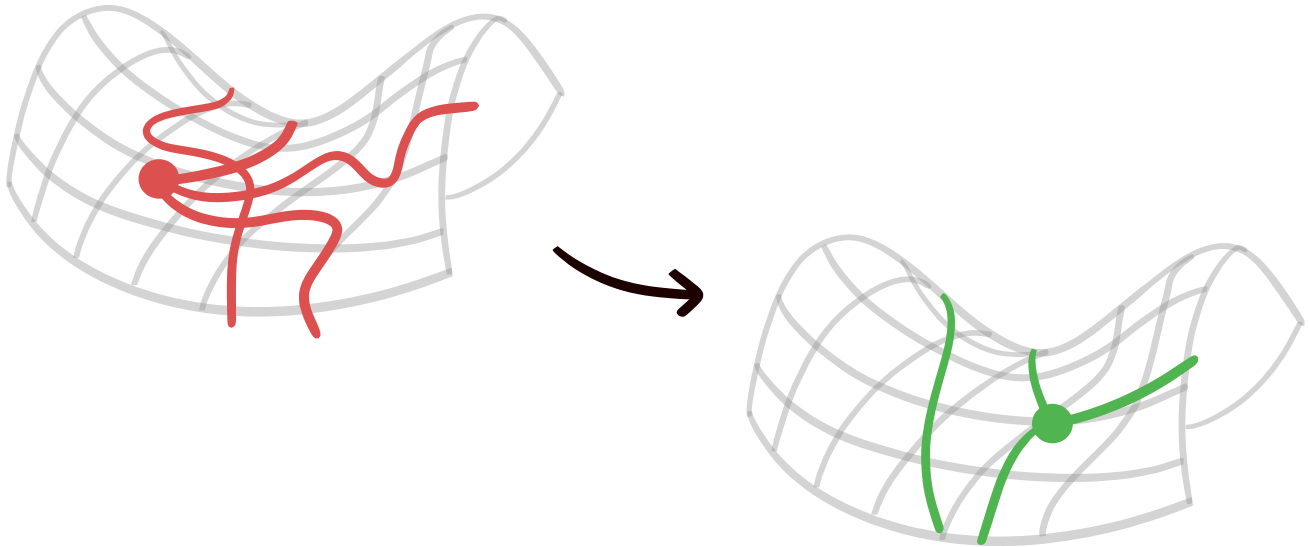
Straightening a graph



A straightened graph is a weak embedding or cannot be untangled

Discrete analogue of Tutte embeddings

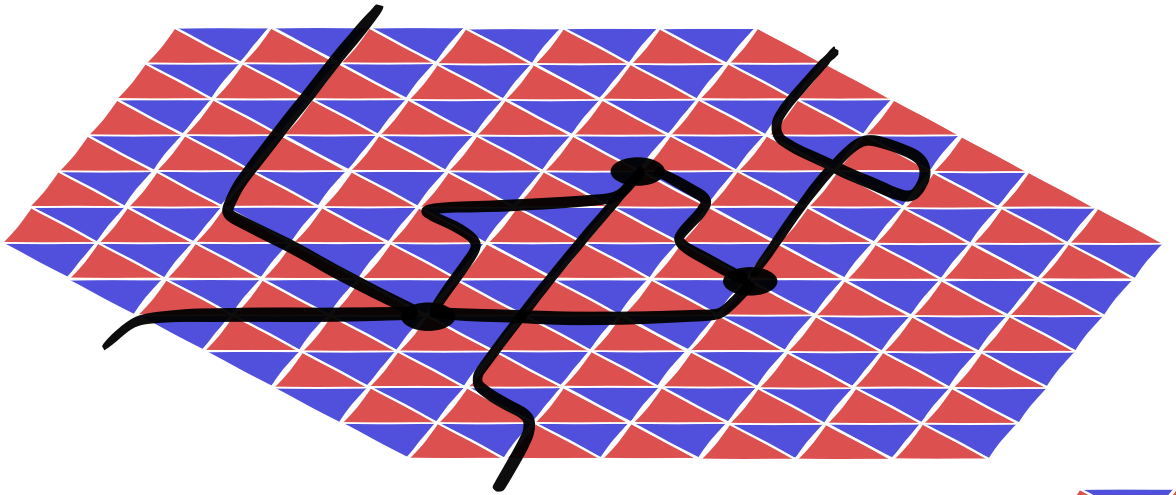
Recall: Tutte embeddings

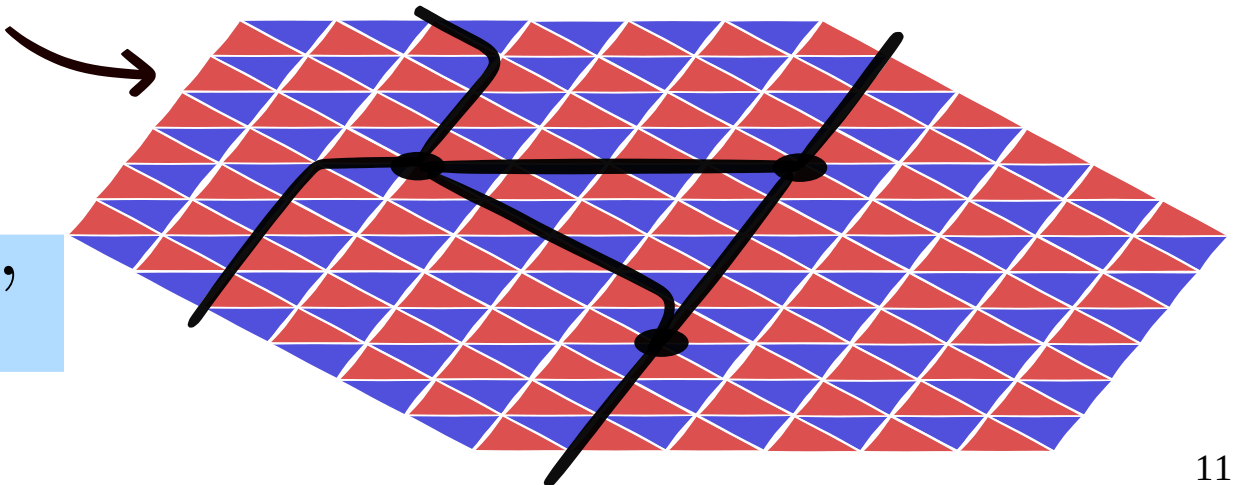


edges straight and

vertices "barycentric"

Harmonious drawings



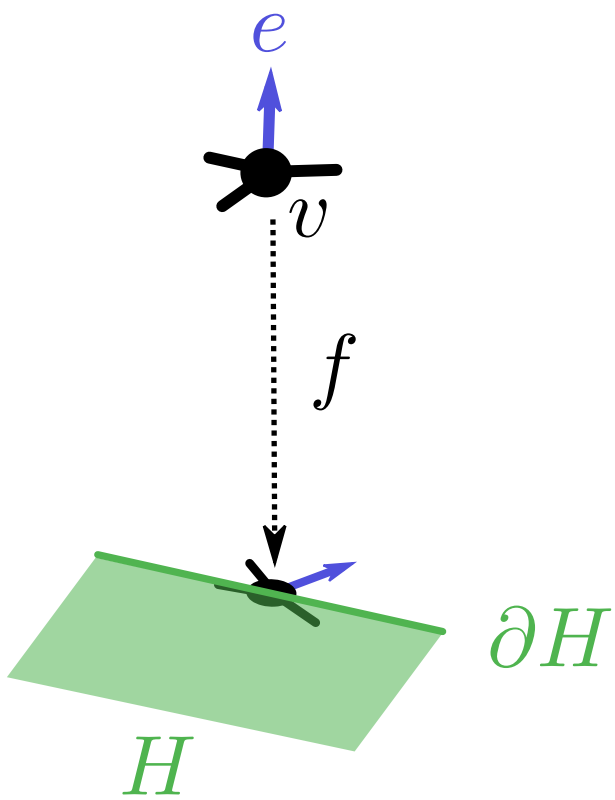


edges reduced and

vertices “barycentric”

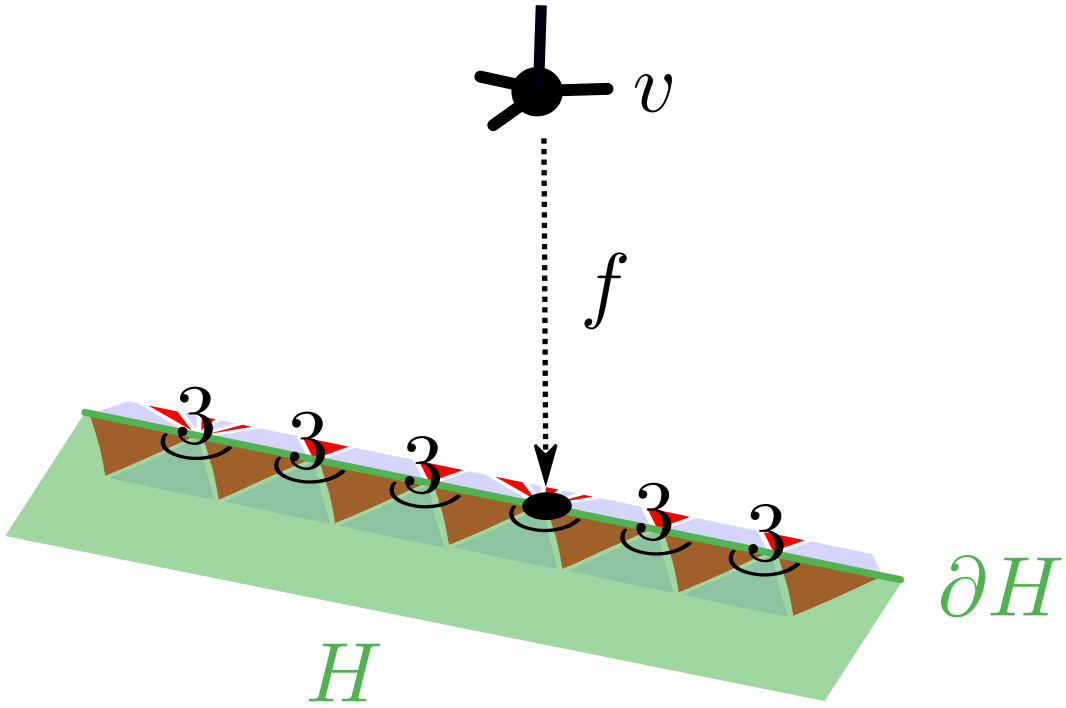
vertices "barycentric"

$$f(v) \in \partial H$$
$$\Longrightarrow$$
$$\exists e \; f(e) \text{ escapes } H$$



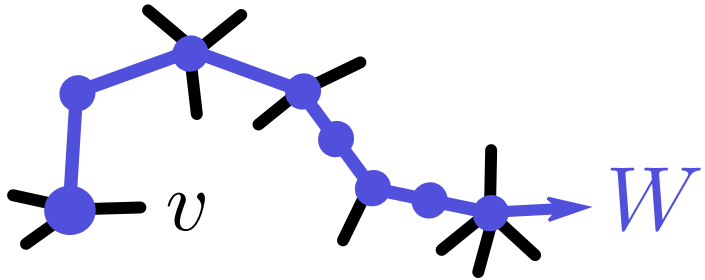
vertices "barycentric"

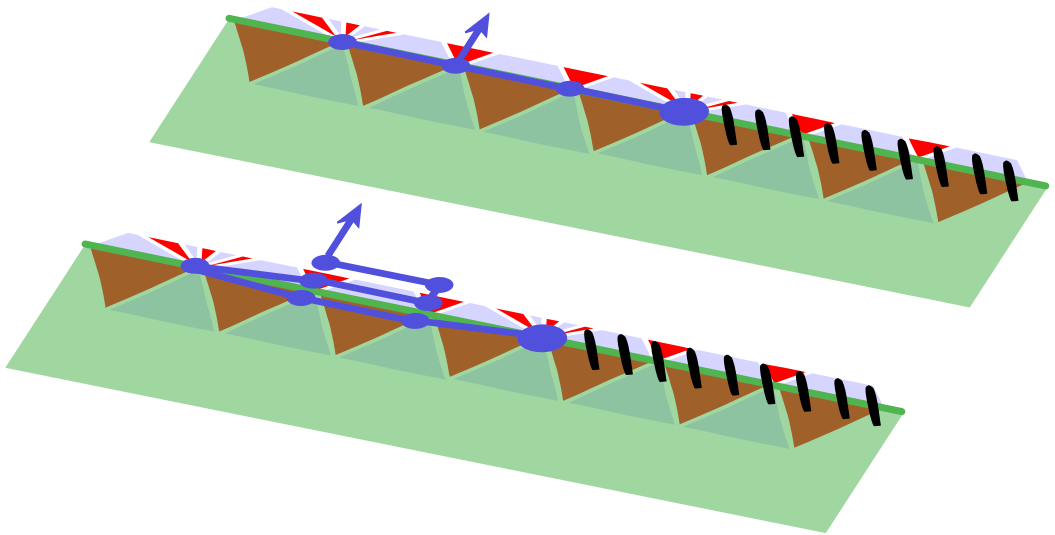
$f(v) \in \partial H$



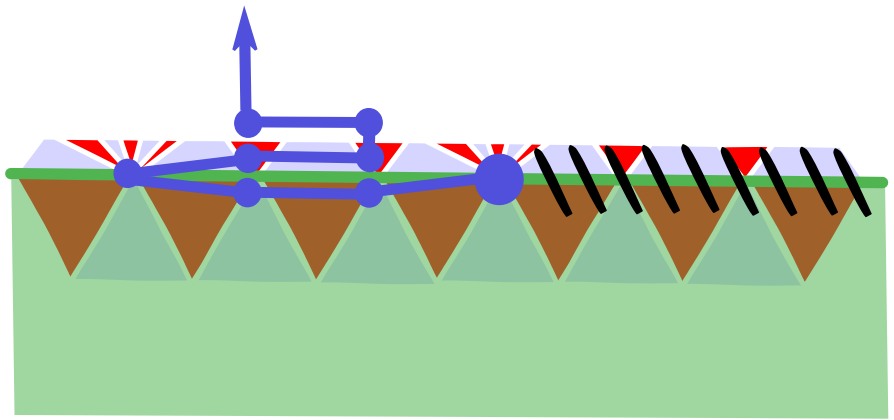
vertices "barycentric"

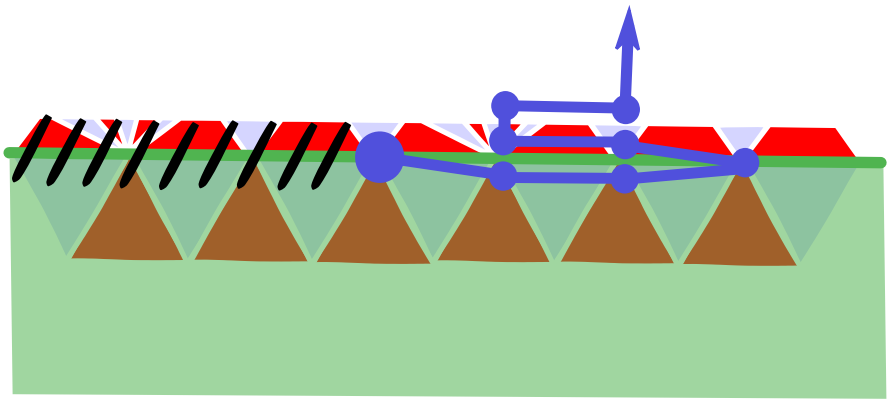
$$f(v) \in \partial H$$
$$\Longrightarrow$$
$$\exists W \; f \circ W \text{“escapes”} H$$

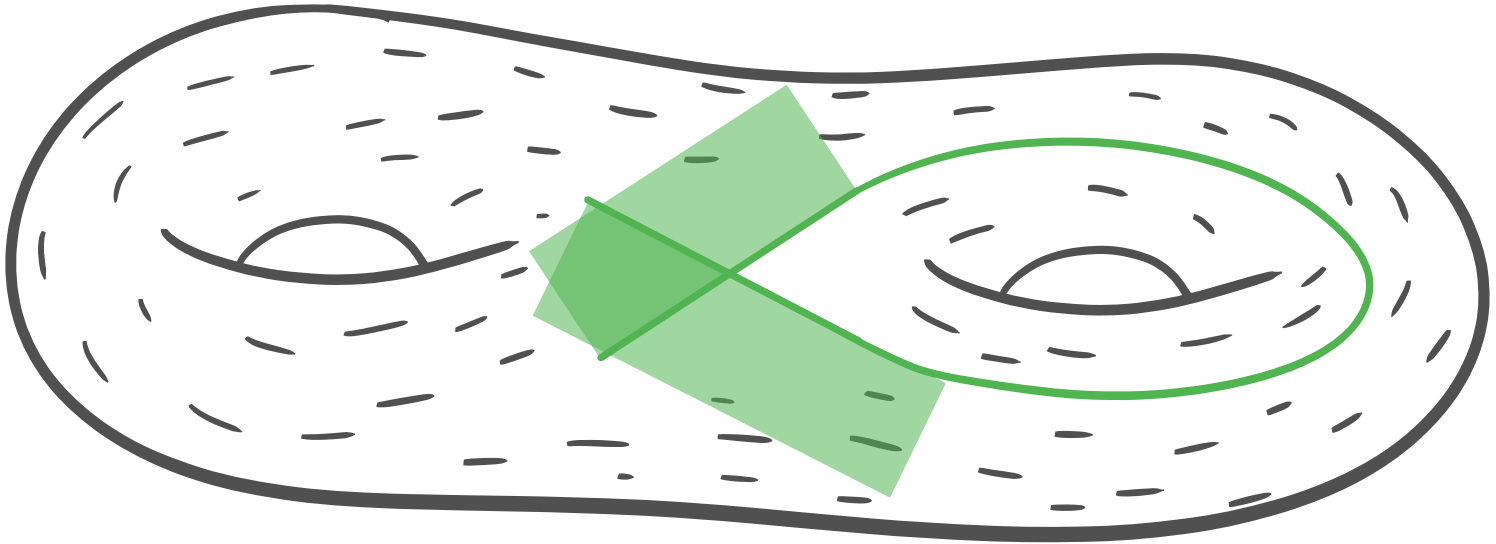




where to escape depends on the coloring







this definition generalizes to surfaces

Results

graph G
reducing triangulation T with m edges
$f : G \to T^1$ of size n

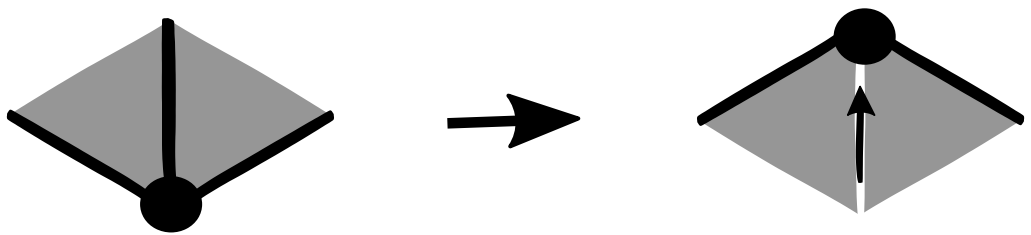
Definition of *harmonious* drawings

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

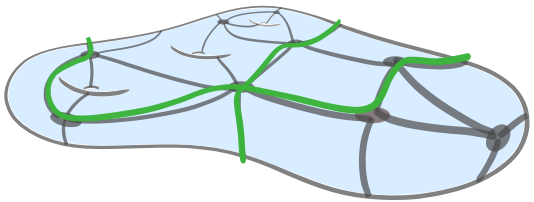
1. we define monotonic moves to apply to f

ex: 

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

Making curves cross minimally

Related work

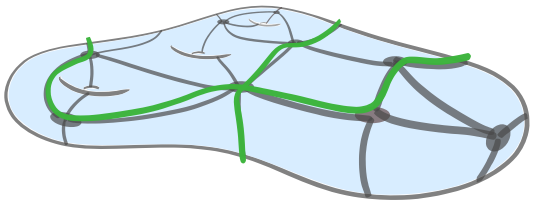


closed walks of total length n
on a graph of size m

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!

D., 2024

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

- 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

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

Untangling Graphs

Computing Delaunay Triangulations

Conclusion

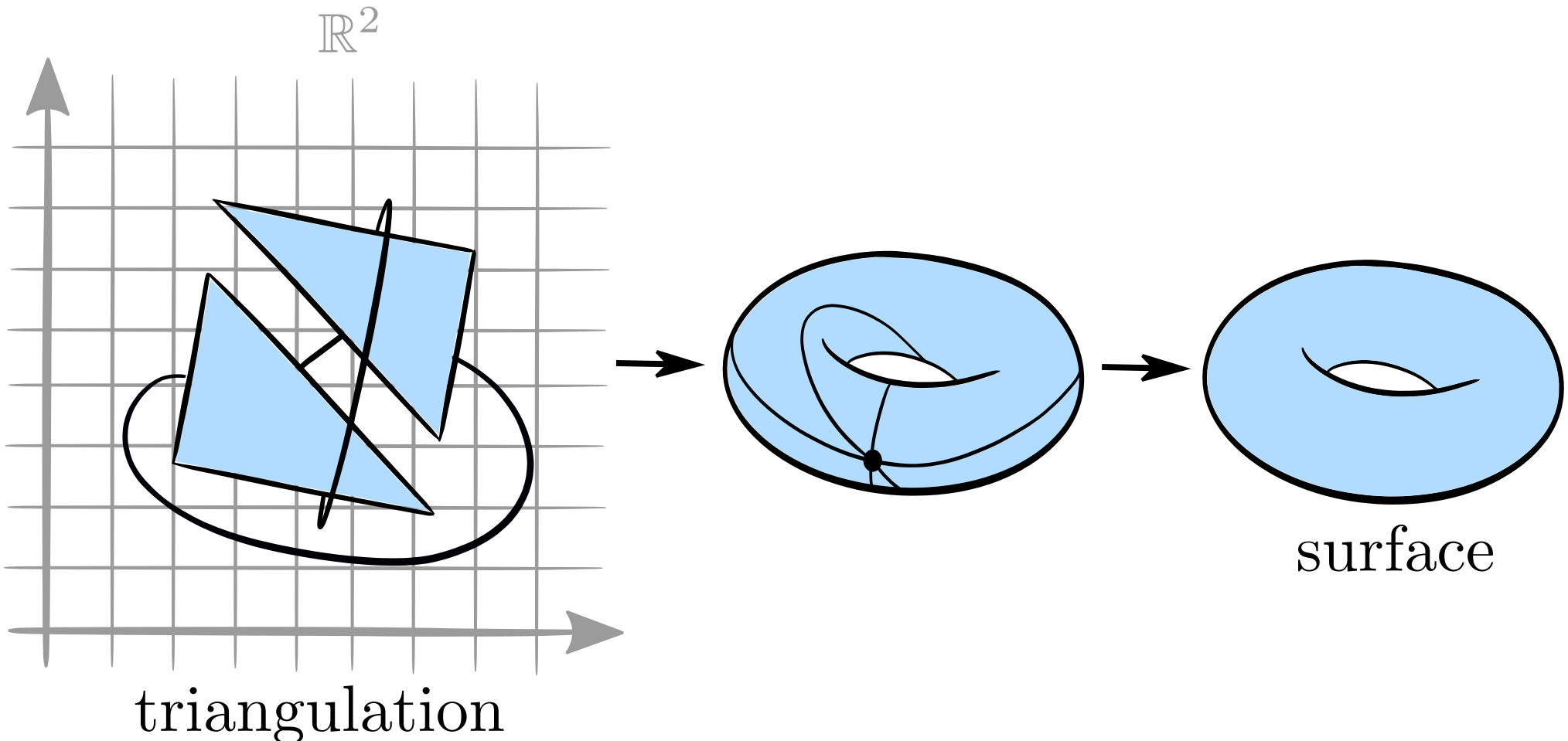
Untangling Graphs

Computing Delaunay Triangulations

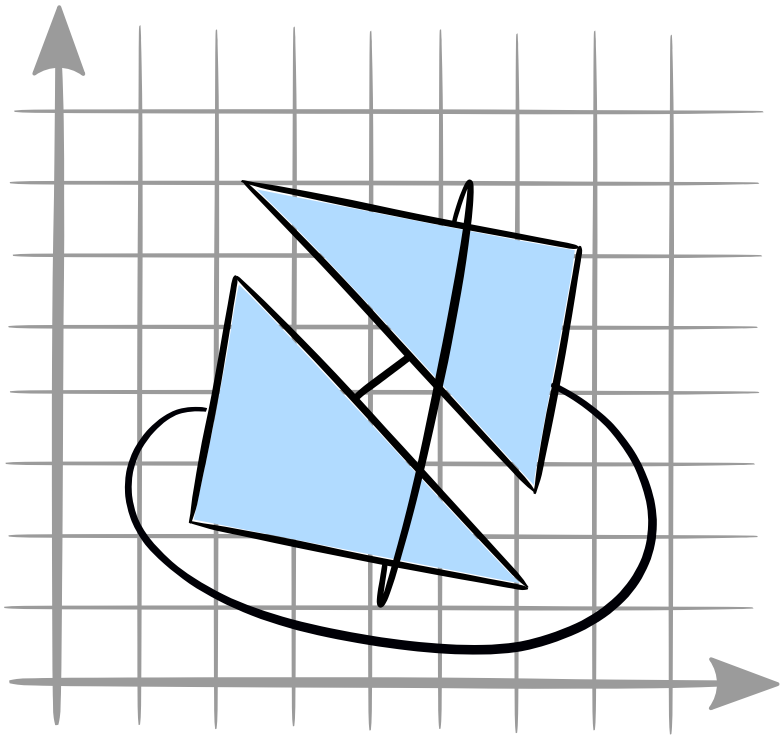
Conclusion

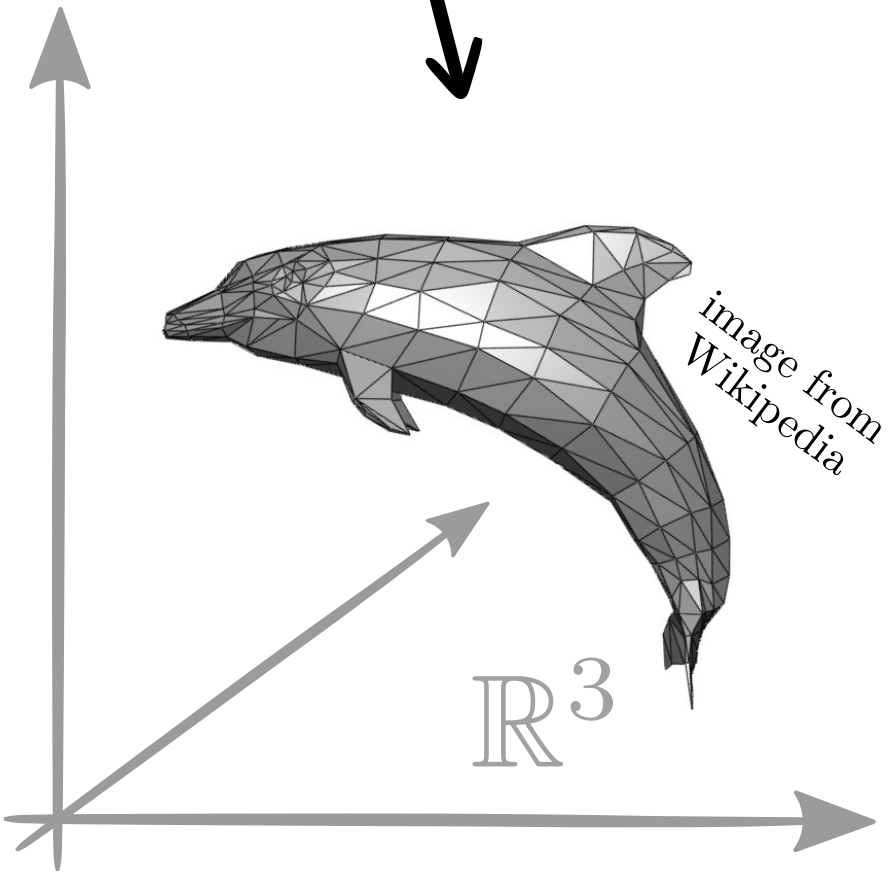
Triangulations of polyhedral surfaces

Triangulation of polyhedral surface

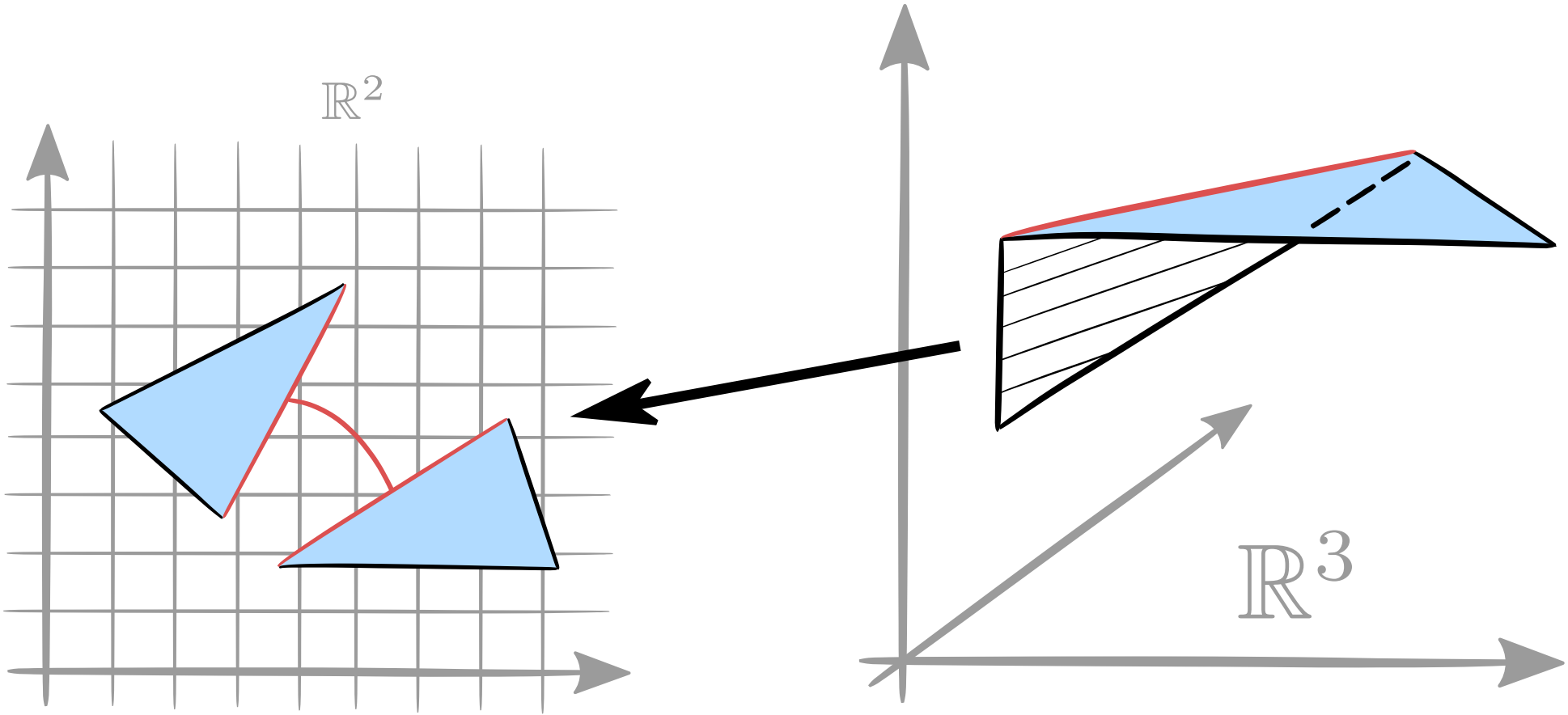


Triangulation vs. Mesh



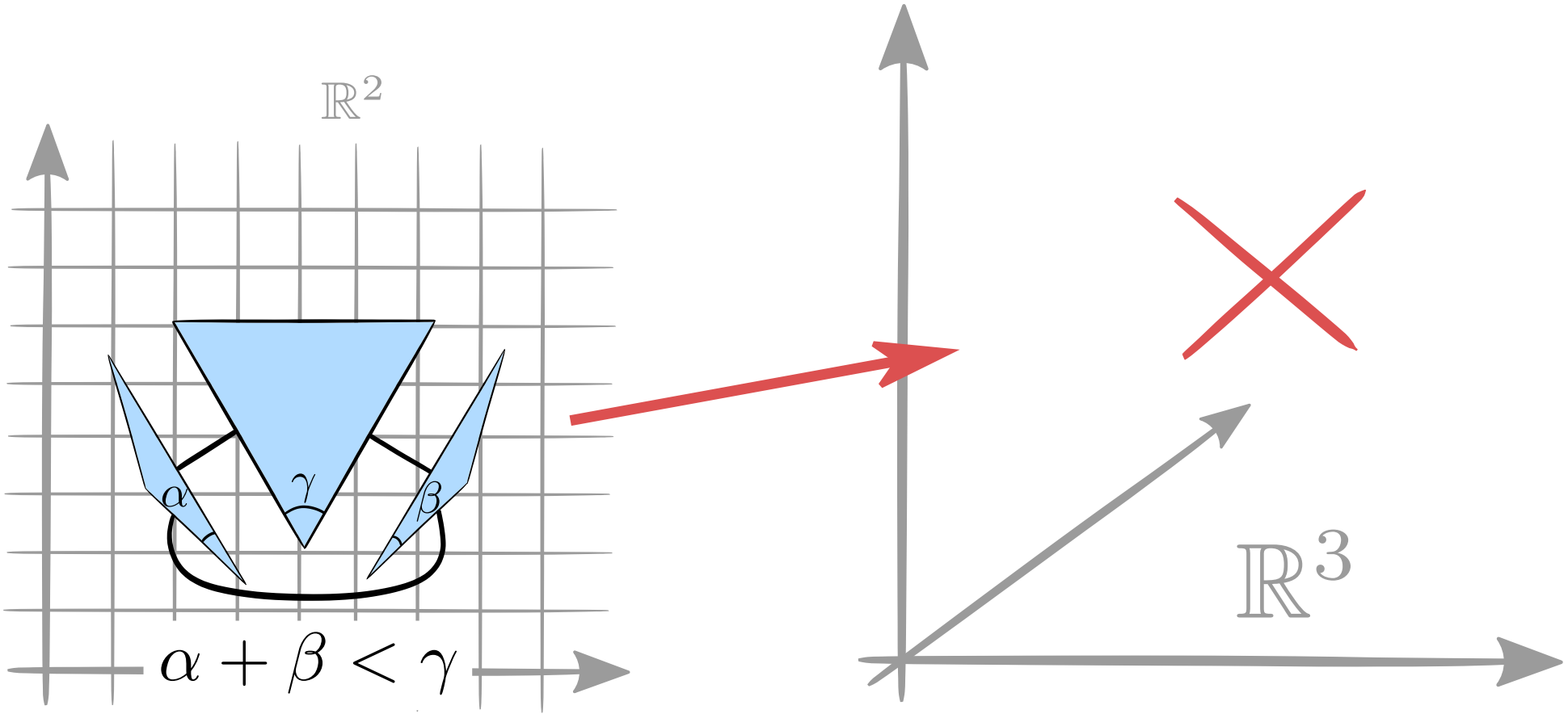


Every mesh gives a triangulation

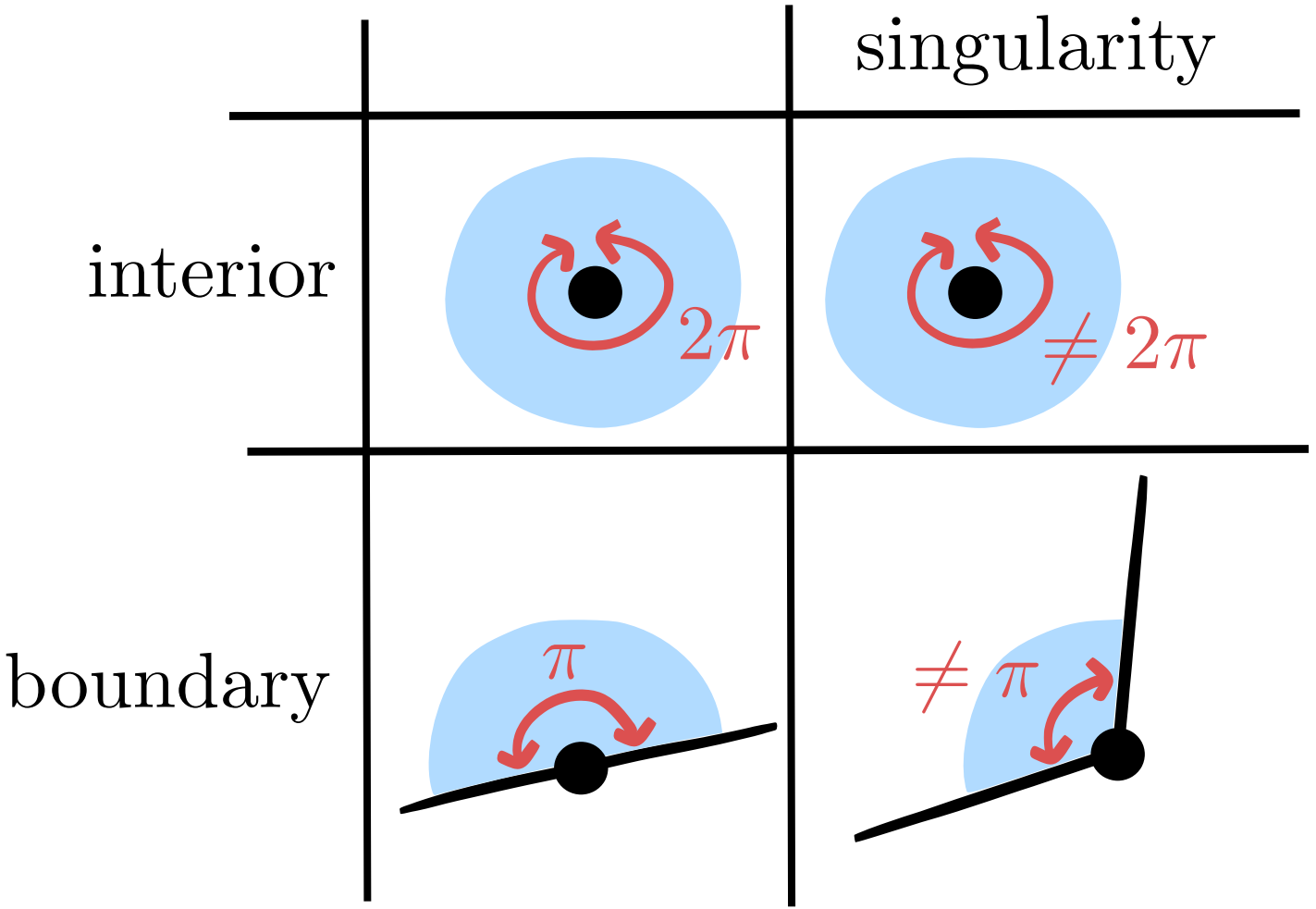


Every mesh gives a triangulation

but converse is false!



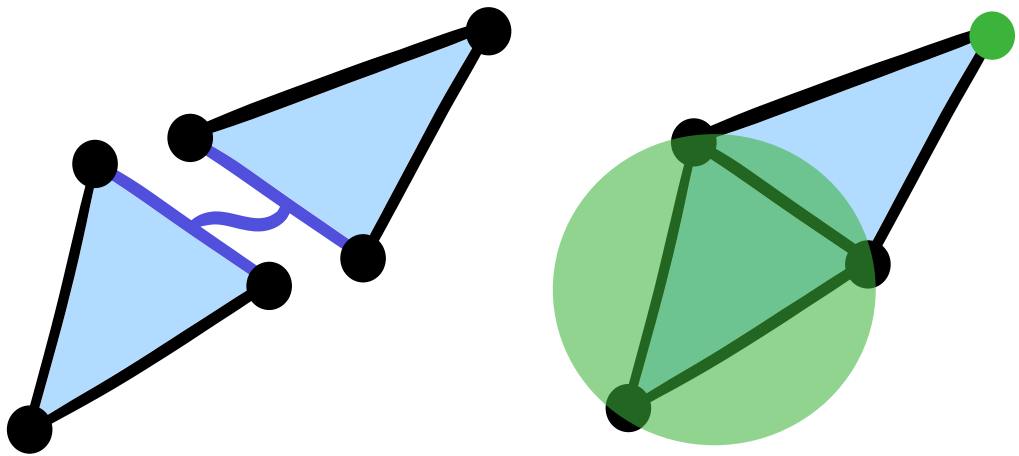
Types of points on the surface



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

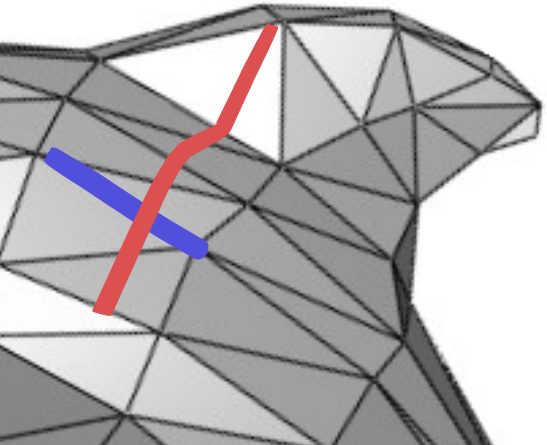
- shortest paths

Shortest paths on meshes

On a mesh M with n triangles...

a shortest path cannot cross an edge twice

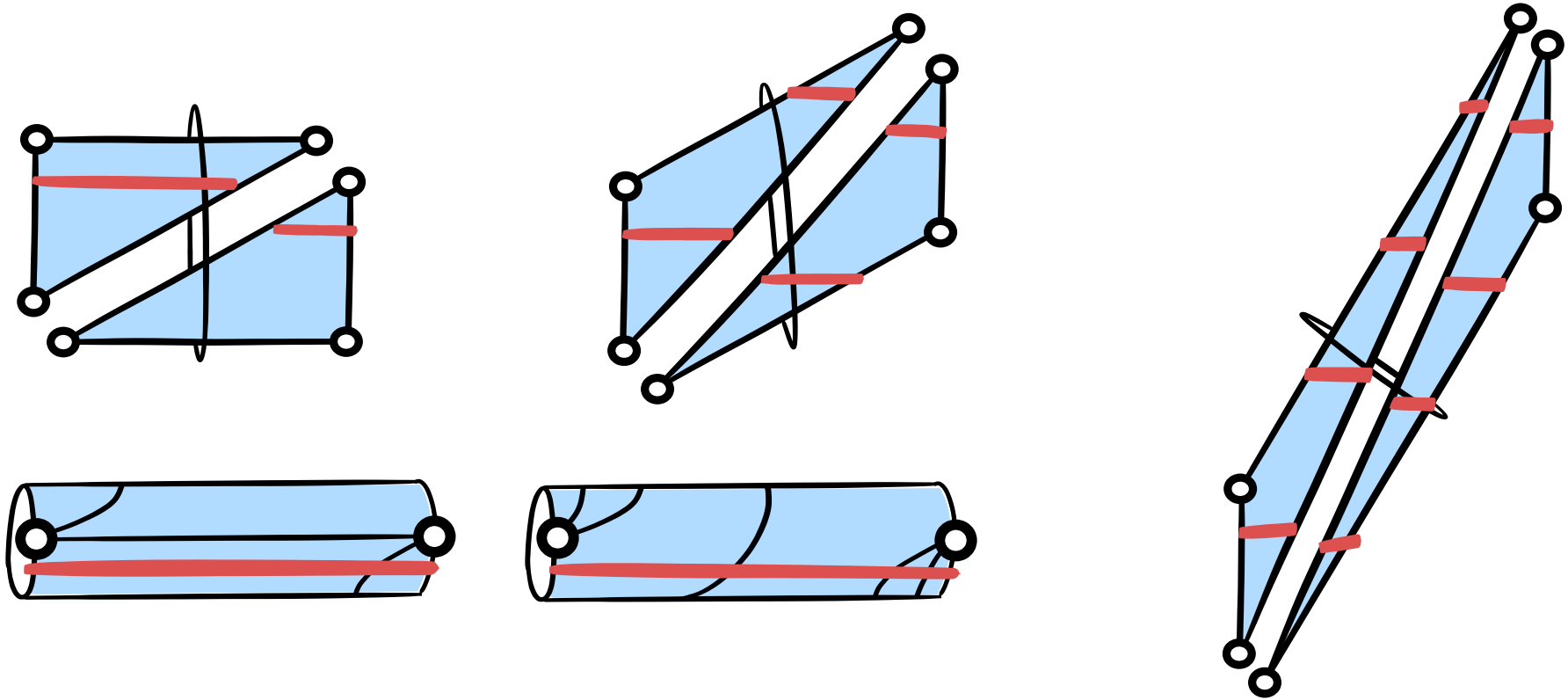
$\rightarrow$ shortest path can be computed in $O(f(n))$ time



Mitchel, Mount, Papadimitriou, 1987

Shortest paths on triangulations

they can cross edges arbitrarily many times

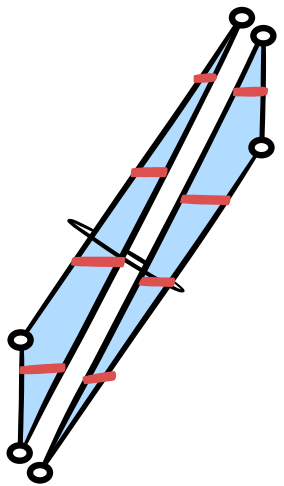


Erickson, 2006

Shortest paths on triangulations

Löffler, Ophelders, Staals, Silveira, 2023

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



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

Shortest paths on triangulations

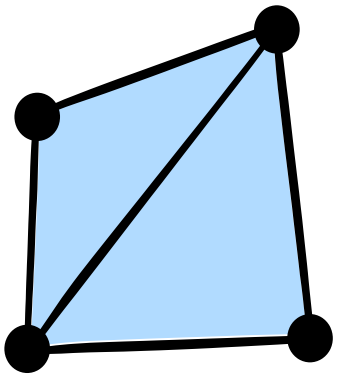
Löffler, Ophelders, Staals, Silveira, 2023

Delaunay triangulations have happiness $O(1)$

Exisiting algorithms

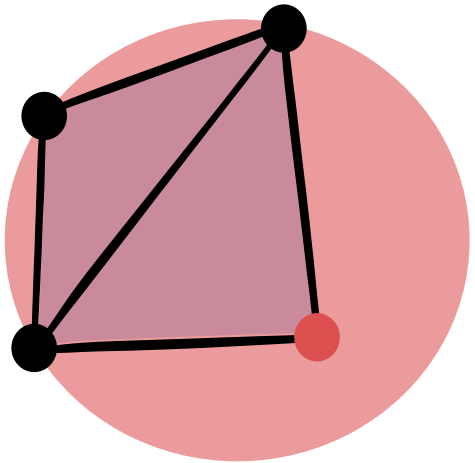
Delaunay flips algorithm

Delaunay flip:



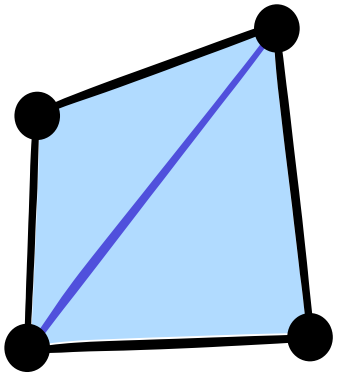
Delaunay flips algorithm

Delaunay flip:



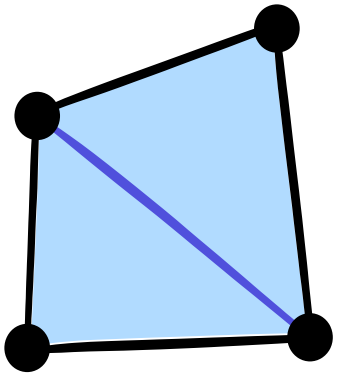
Delaunay flips algorithm

Delaunay flip:



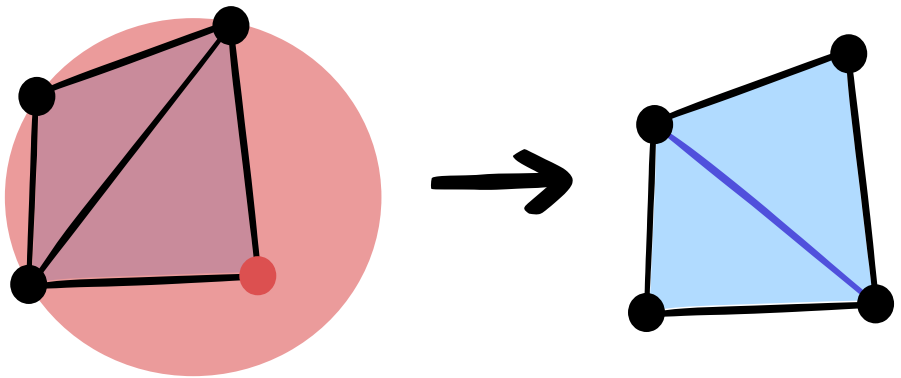
Delaunay flips algorithm

Delaunay flip:



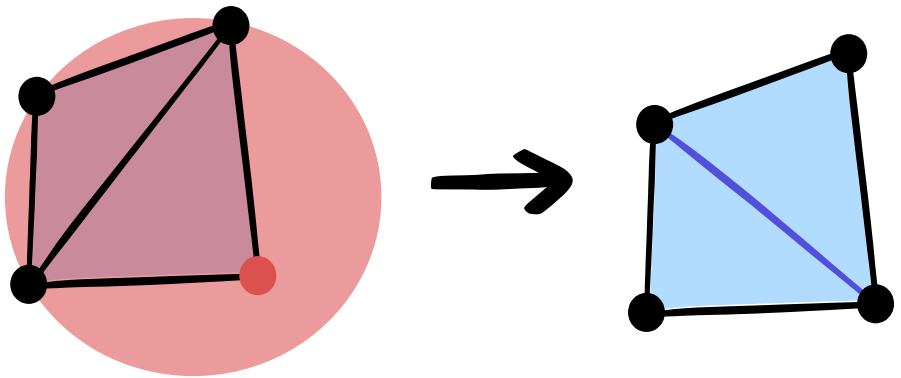
Delaunay flips algorithm

Delaunay flip:



Delaunay flips algorithm

Delaunay flip:

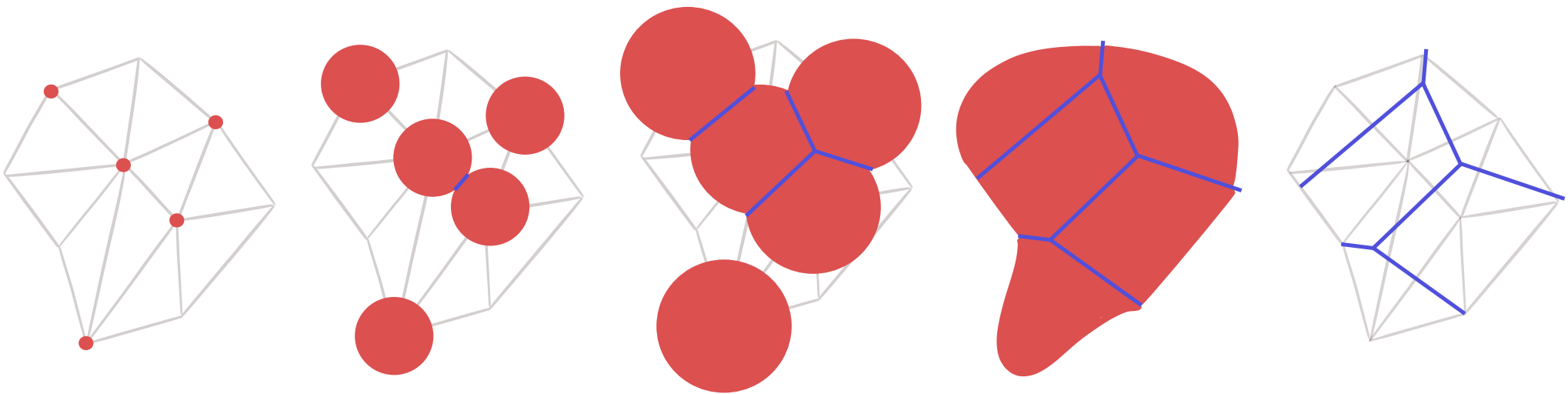


Algorithm:

apply Delaunay flips greedily as long as you can

Other method

compute the Voronoi diagram
by propagating waves

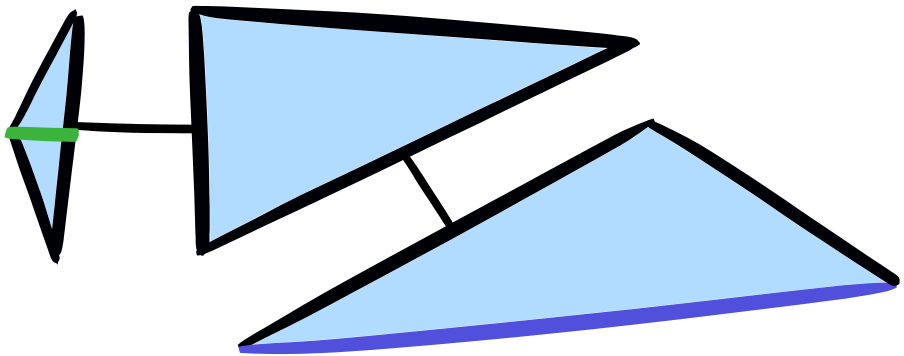


then derive Delaunay from it

Result

Result

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

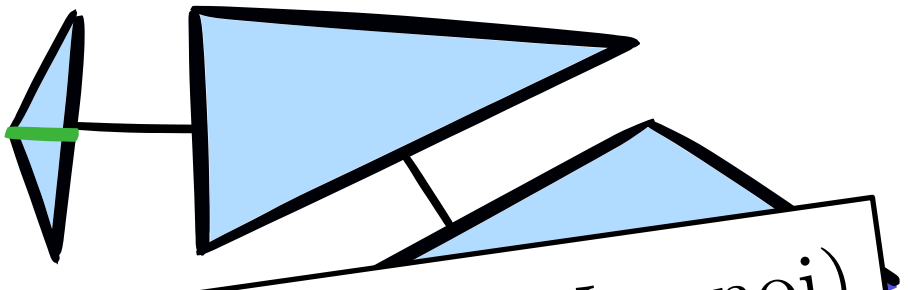


D. 2025

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

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



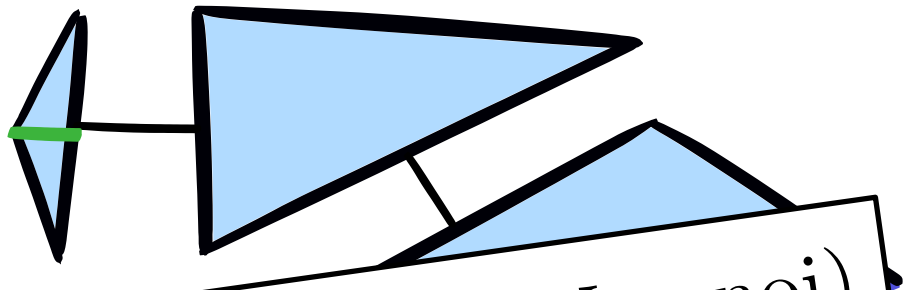
Previous algorithms (Delaunay flips and Voronoi) achieved no better than $O(\text{Poly}(n, r))$

D

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

Result

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



Previous algorithms (Delaunay fli… and Voronoi)
achieved no better than $O(P$…

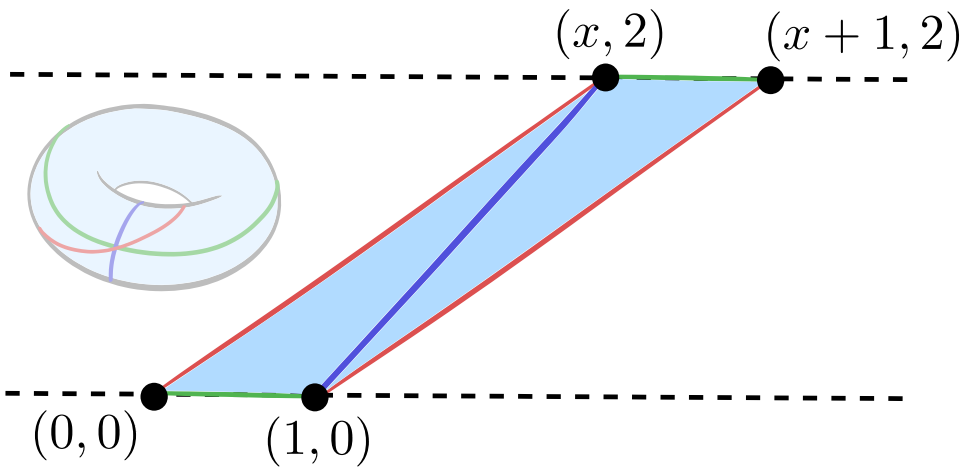
Now backed by a lower bound!

…surface has no boundary, we can compu…
Delaunay in $O(n^3 \log^2(n) \cdot \log^4(r))$ time

Lower bound

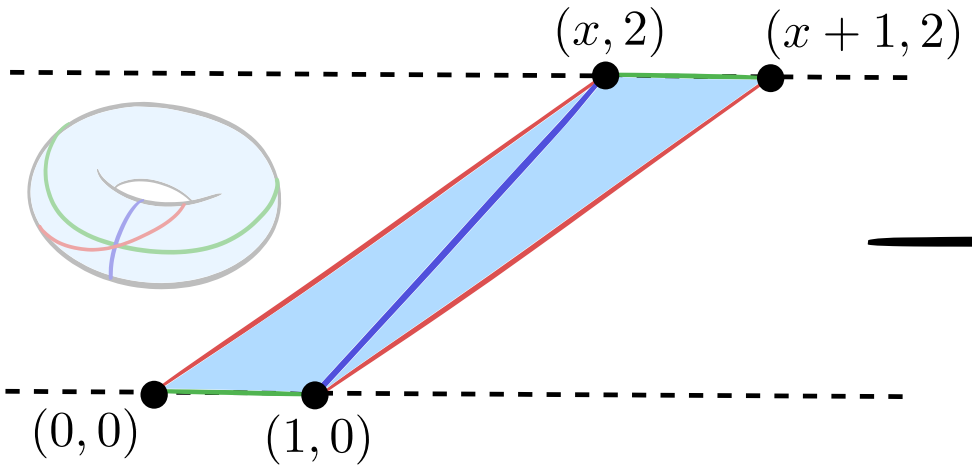
Lower bound

Input

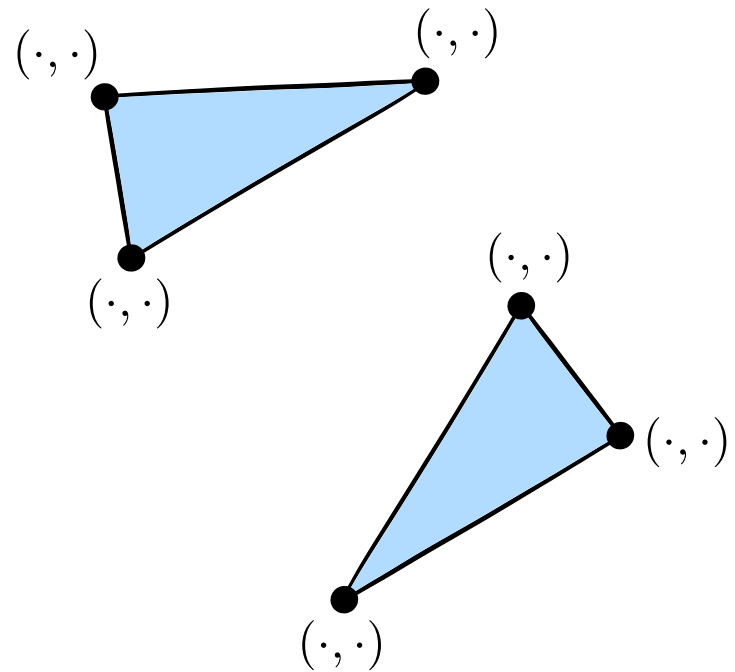


Lower bound

Input

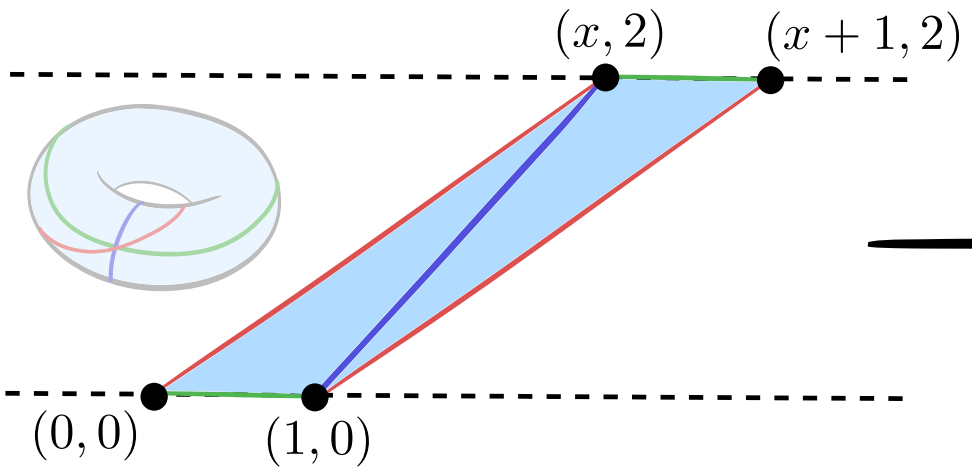


Output: Delaunay

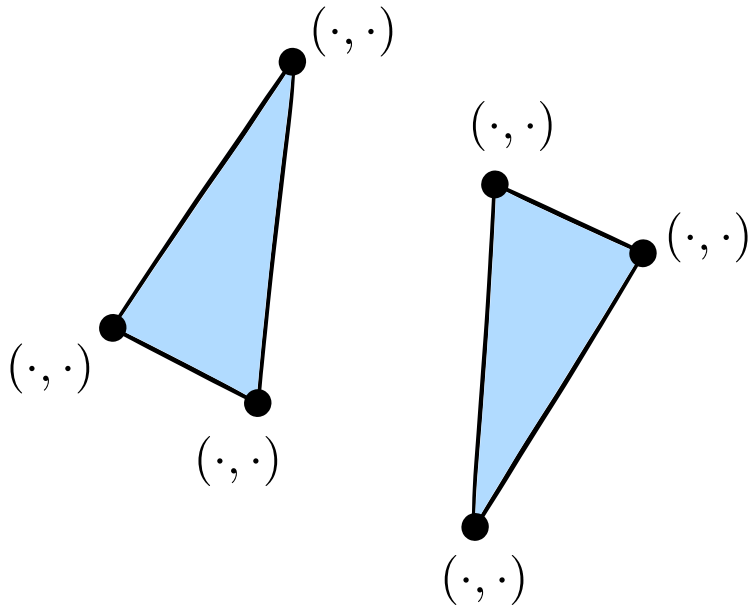


Lower bound

Input



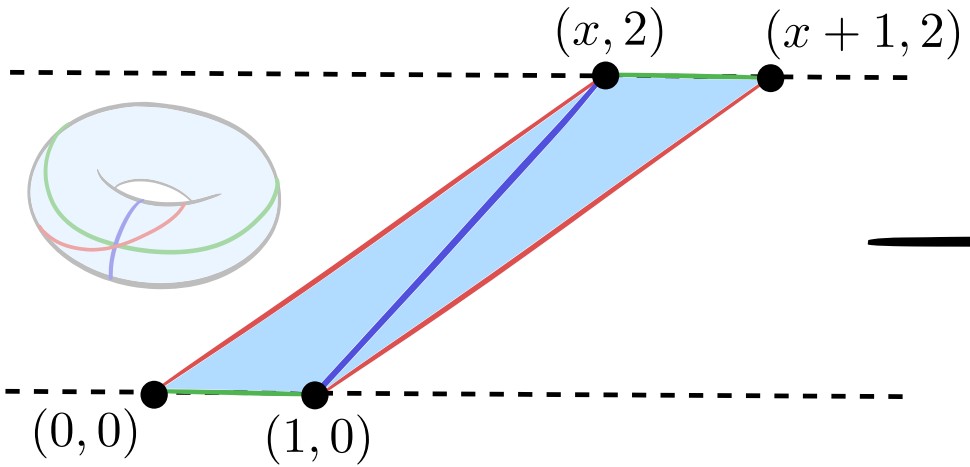
Output: Delaunay

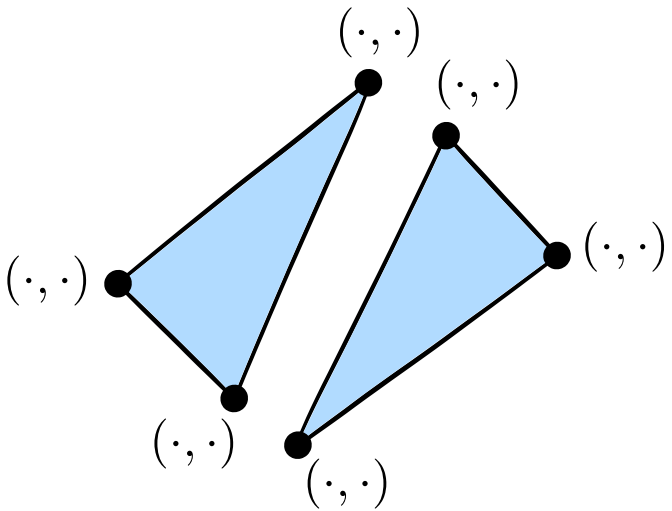


Lower bound

Input

Output: Delaunay

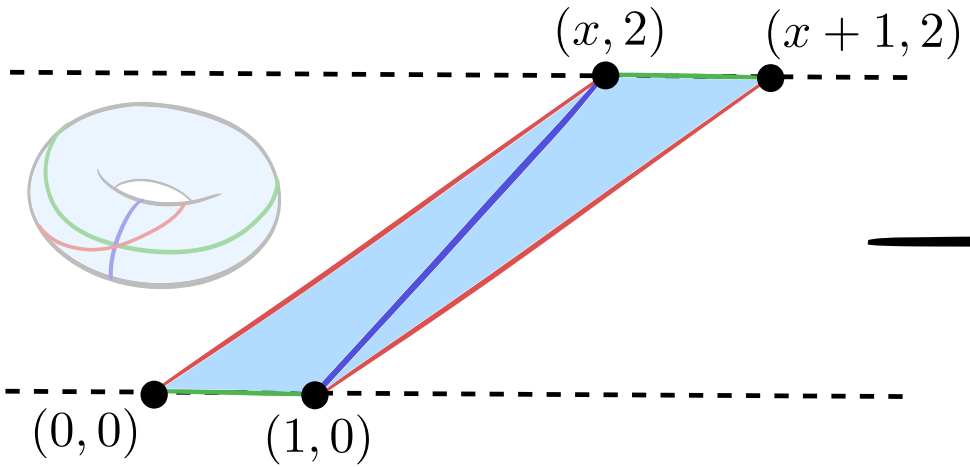


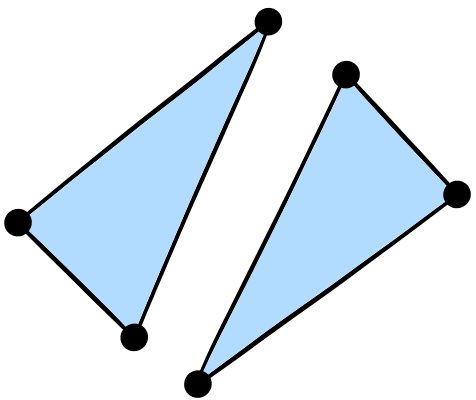


Lower bound

Input

Output: Delaunay

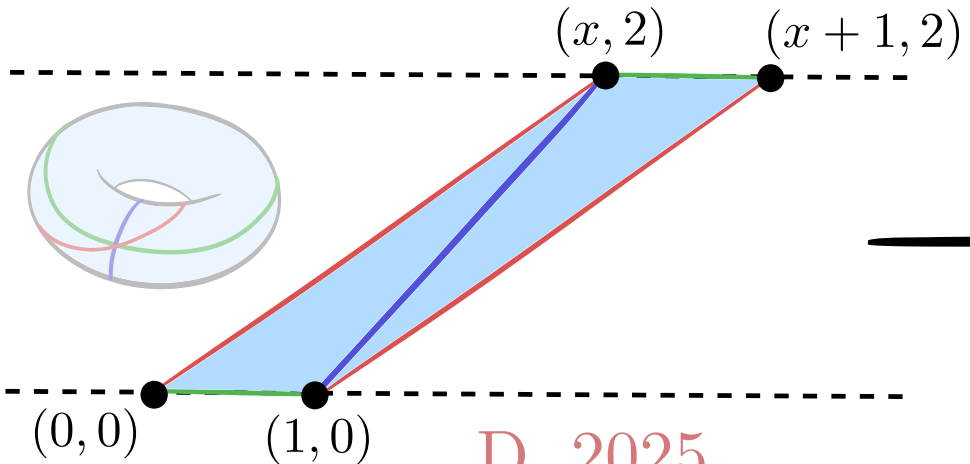




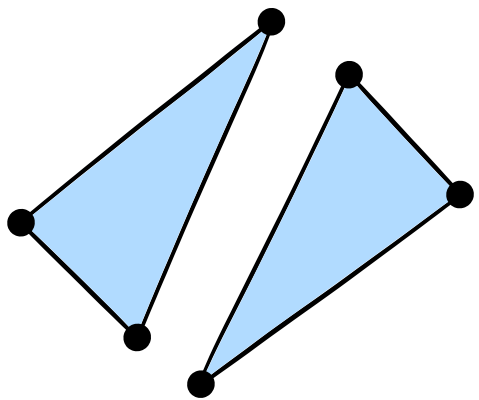
Lower bound

Input

Output: Delaunay



D. 2025

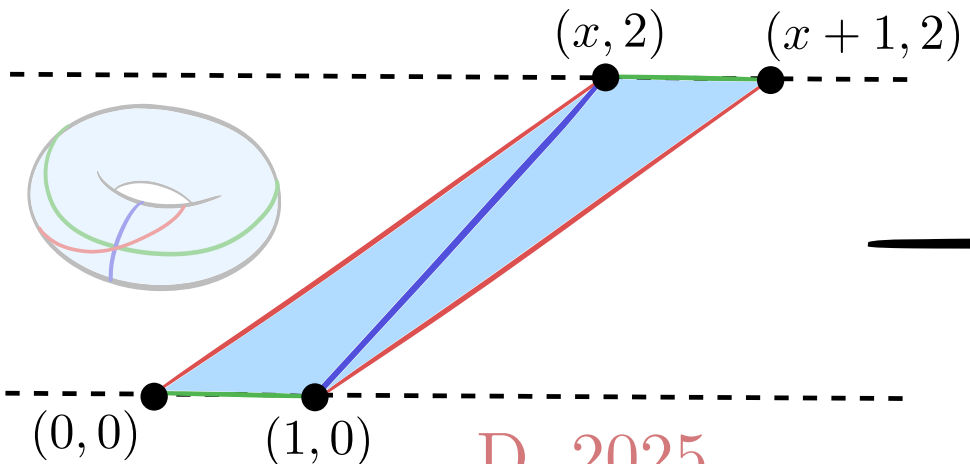


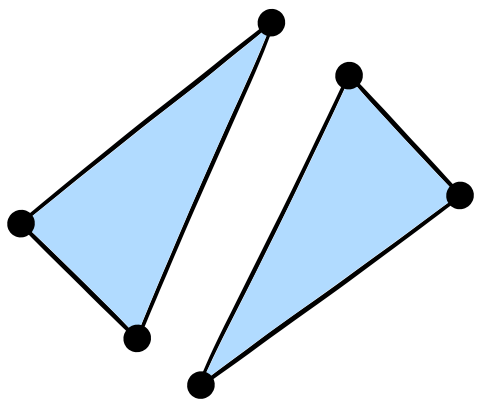
No Real RAM algo can compute Delaunay from x in $o(\log x)$ time

Lower bound

Input

Output: Delaunay





D. 2025

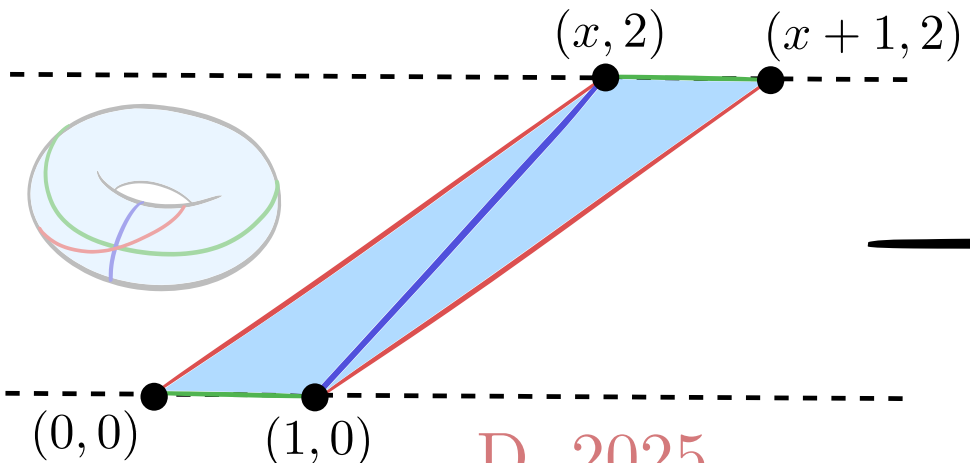
No Real RAM algo can compute Delaunay from x in $o(\log x)$ time

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

Lower bound

Input

Output: Delaunay



D. 2025

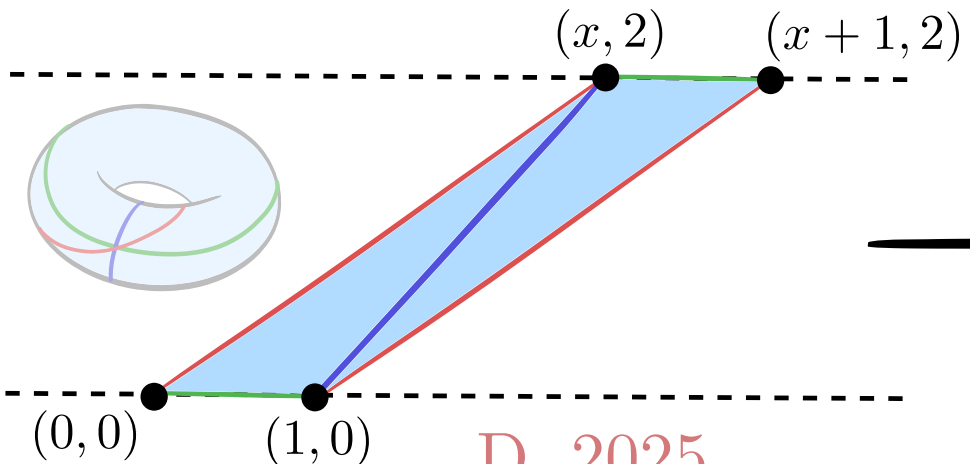
No Real RAM algo can compute Delaunay from x in $o(\log x)$ time

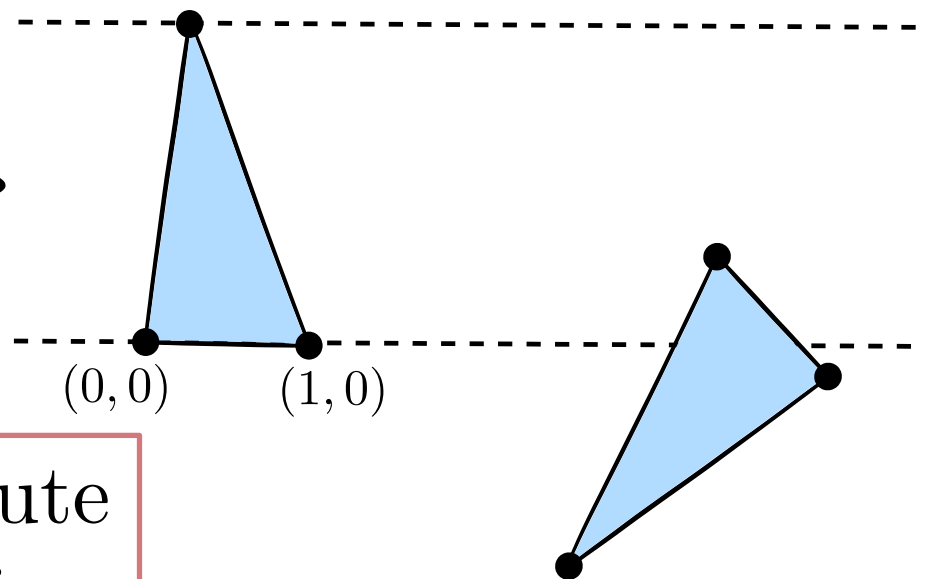
↪ Otherwise we could compute $\lfloor x \rfloor$ from x in $o(\log x)$ time

Lower bound

Input

Output: Delaunay





D. 2025

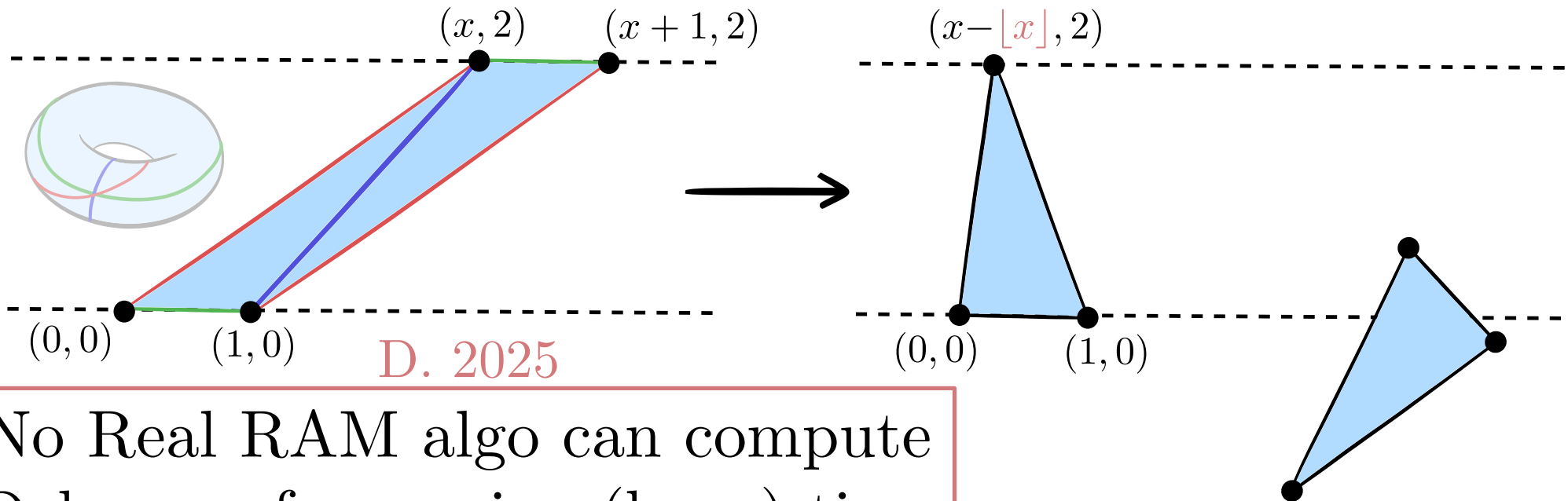
No Real RAM algo can compute Delaunay from x in $o(\log x)$ time

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

Lower bound

Input

Output: Delaunay



D. 2025

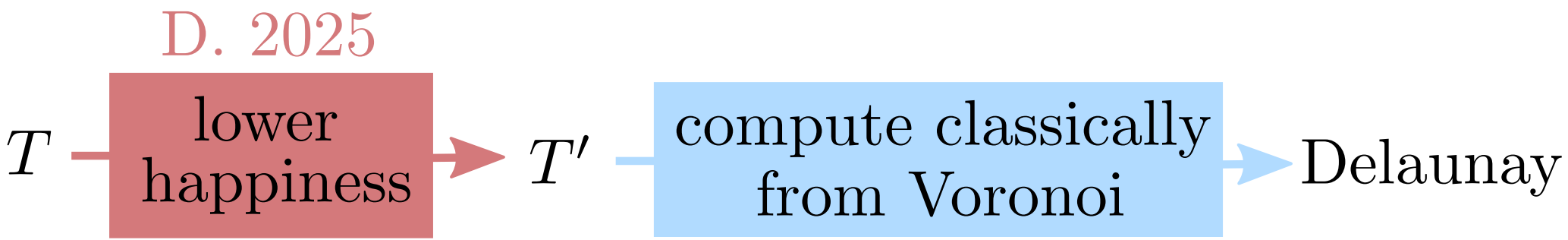
No Real RAM algo can compute Delaunay from x in $o(\log x)$ time

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

Algorithm overview

Algorithm

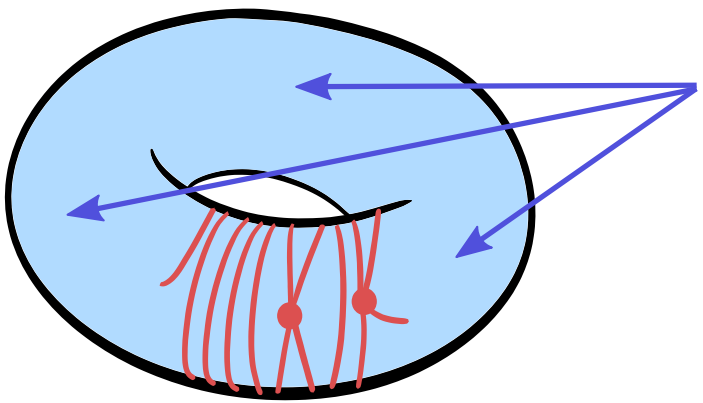
Algorithm


D. 2025
T
lower happiness
T'
compute classically from Voronoi
Delaunay

Algorithm

D. 2025

T → lower happiness → T' → compute classically from Voronoi → Delaunay

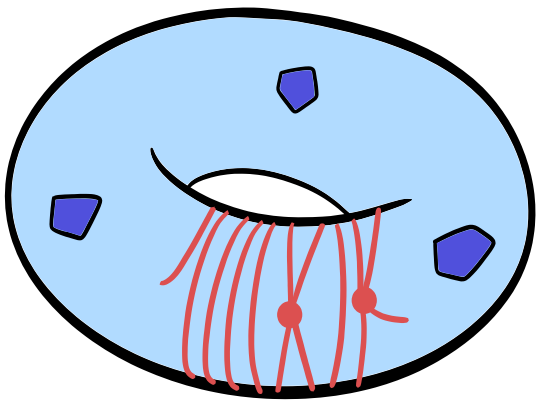


consider the singularities

Algorithm

D. 2025

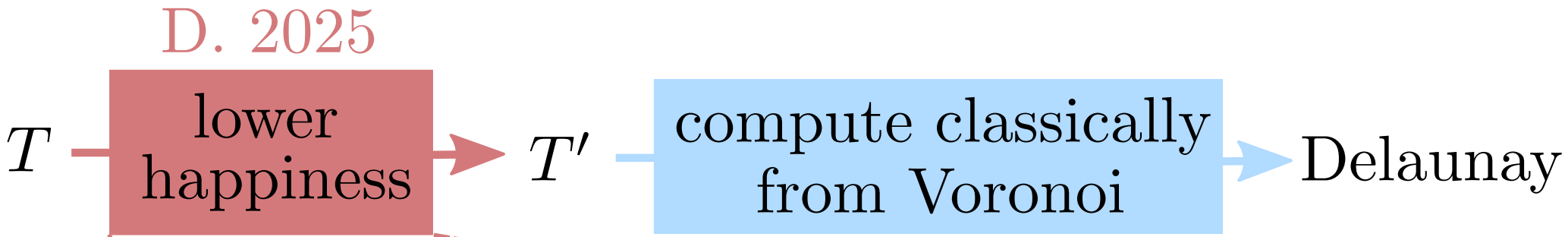
T → lower happiness → T' → compute classically from Voronoi → Delaunay

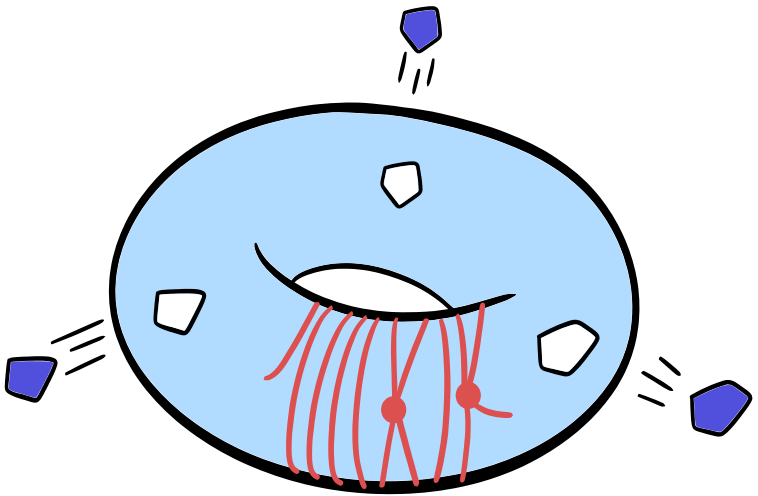


cut out caps around the singularities

Algorithm

D. 2025



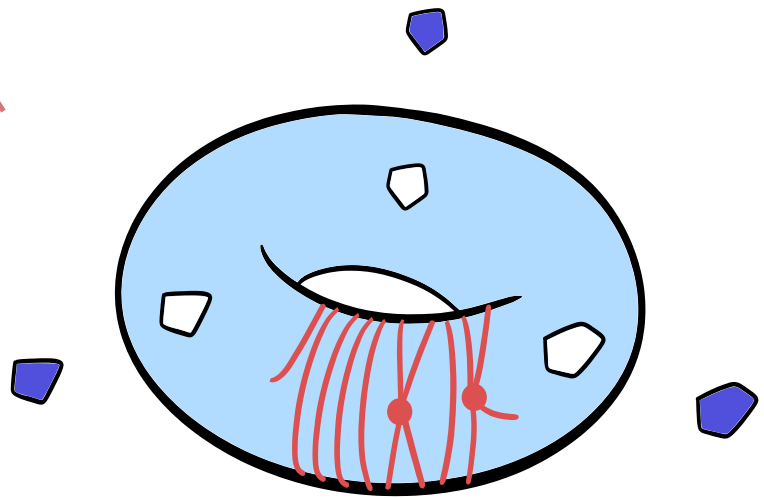


cut out caps around the singularities

Algorithm

D. 2025

T → lower happiness → T' → compute classically from Voronoi → Delaunay

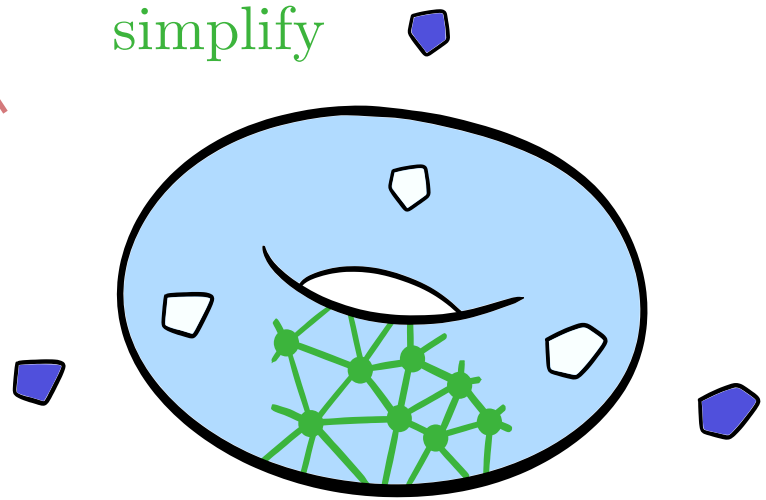


cut out caps around the singularities

Algorithm

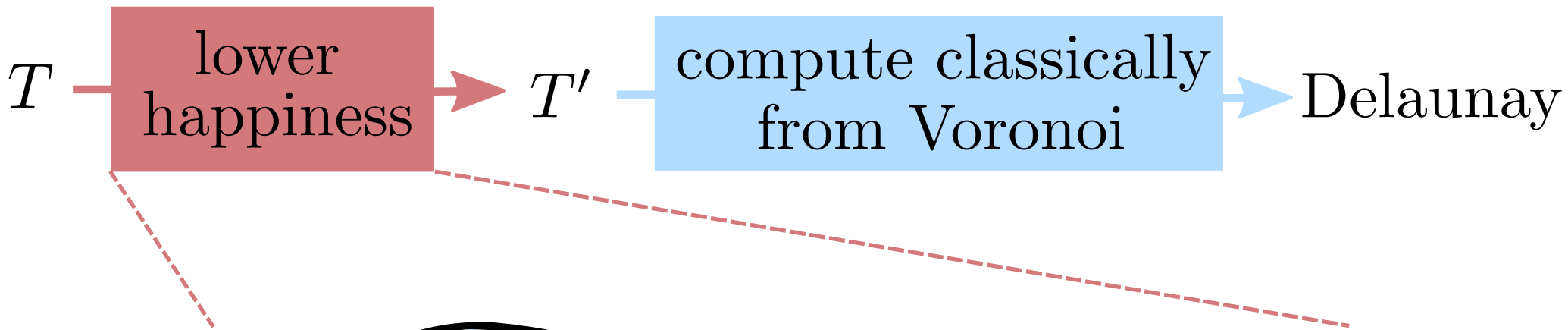
D. 2025

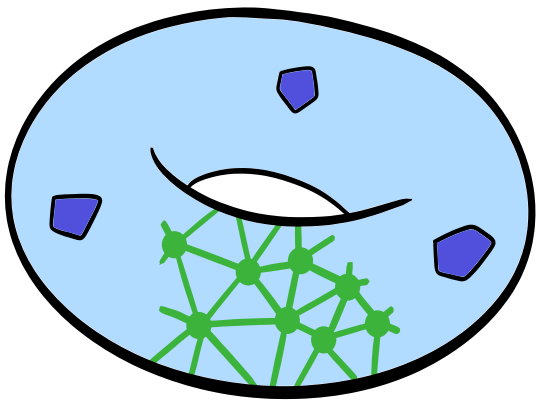
T → lower happiness → T' → compute classically from Voronoi → Delaunay



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

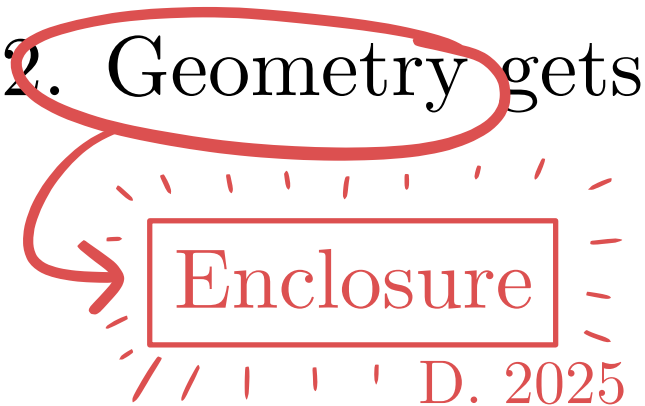
repeated many times

some simplify the geometry,
others decrease # vertices

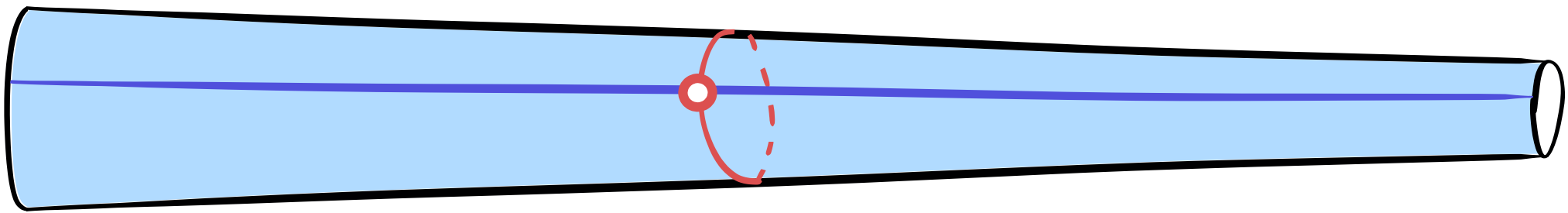
Analysis

Show that during execution:

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



Enclosure



Untangling Graphs

Computing Delaunay Triangulations

Conclusion

Untangling Graphs

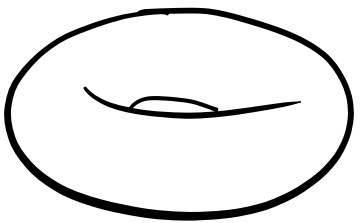
Computing Delaunay Triangulations

Conclusion

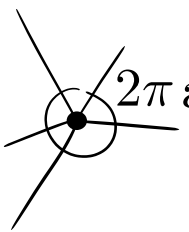
Other results

D., 2022-23

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



topological shape of a torus



2π around each vertex

Other results

Despré, D., Pouget, and Teillaud, 2025

CGAL package for computing with hyperbolic surfaces

generation (genus 2 only)

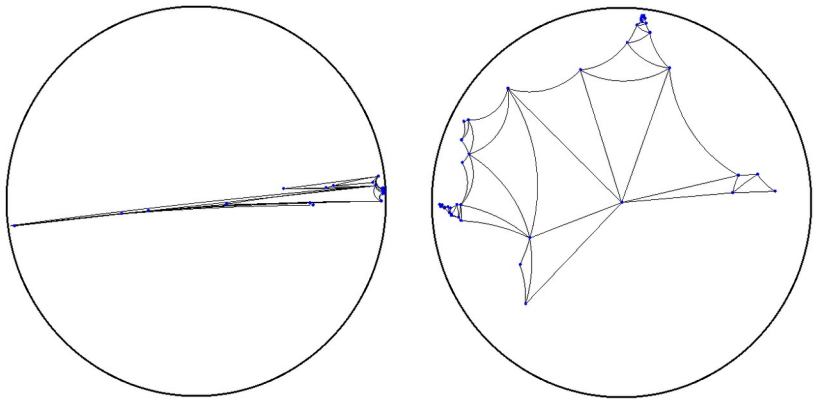
↓

Triangulated hyperbolic surface

↓

visualization

Delaunay flip



Other results

Exact computations!

Despré, D., Pouget, and Te

CGAL package for computing hyperbolic surfaces

generation (genus 2 only)

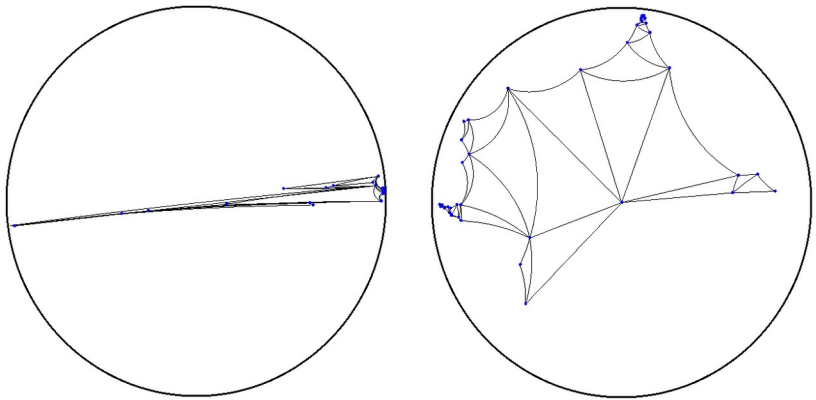
↓

Triangulated hyperbolic surface

↓

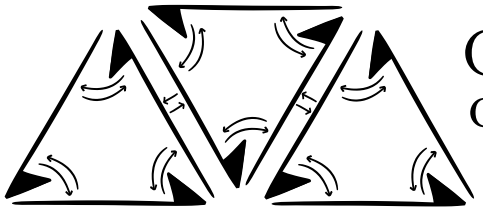
visualization

Delaunay flip



Triangulated hyperbolic surface

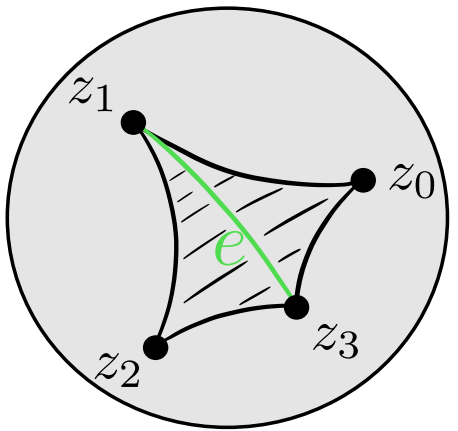
Combinatorics



CGAL
combinatorial maps

Geometry

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



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

Triangulated hyperbolic surface

Possible continuations

- generation of hyperbolic surfaces of genus ≥ 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?