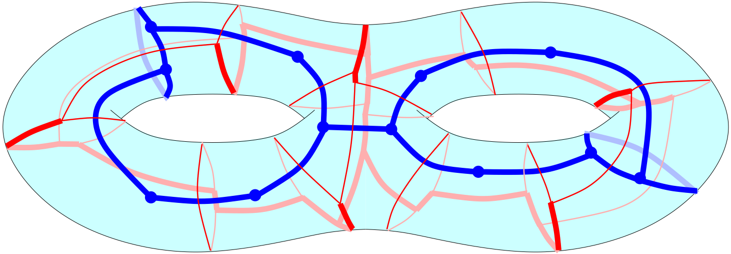
Computational topology of graphs on surfaces

Éric Colin de Verdière

CNRS, LIGM, Université Paris-Est Marne-la-Vallée, France

(joint works by/with many coauthors)



Summer School "Low-Dimensional Geometry and Topology: Discrete & Algorithmic Aspects"

Expected audience

- Graduate students and researchers,
- Computer scientists and mathematicians,
- Communities expected: computational geometry/topology; differential/Riemannian/topological geometry.

Speakers (~ 8 hours of lectures each)

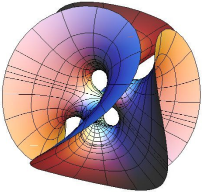
- Jeff Erickson (UIUC),
- Joel Hass (UC Davis).

Shorter talks

about 12 shorter talks will be planned.

When, where

- June 18–22, 2018,
  just after SoCG 2018 in Budapest!
- In the center of Paris (IHP).



Organizing team

Computer scientists and mathematicians at U. Paris-Est Marne-la-Vallée (Labex Bézout): É. Colin de Verdière, X. Goaoc, L. Hauswirth, A. Hubard, S. Sabourau.

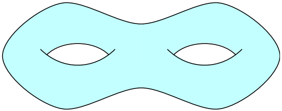
Registration

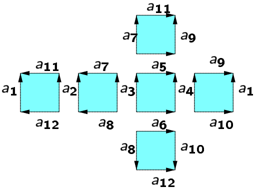
- free but mandatory!
- deadline not before March 15, 2018.

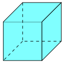
More informations

`http://geomschool2018.univ-mlv.fr/`
Feel free to contact the organizers!

Graphs on surfaces



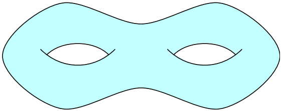


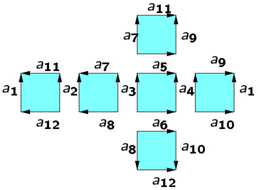


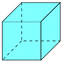
What's a surface? Equivalent definitions:

1. A (compact, connected) 2-manifold.
2. A space obtained by gluing edges of disjoint polygons in pairs.
3. (In the orientable case): A topological space obtained from the sphere by attaching $g \geq 0$ *handles*; g is the genus.

Graphs on surfaces

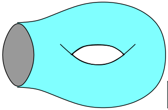
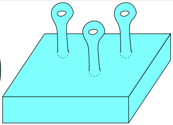
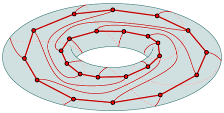


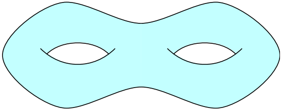


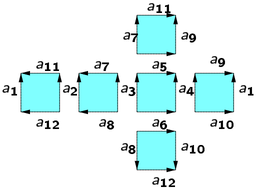


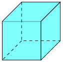
What's a surface? Equivalent definitions:

1. A (compact, connected) 2-manifold.
2. A space obtained by gluing edges of disjoint polygons in pairs.
3. (In the orientable case): A topological space obtained from the sphere by attaching $g \geq 0$ *handles*; g is the genus.

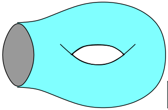


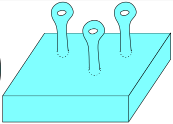


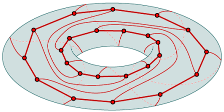


What's a surface? Equivalent definitions:

1. A (compact, connected) 2-manifold.
2. A space obtained by gluing edges of disjoint polygons in pairs.
3. (In the orientable case): A topological space obtained from the sphere by attaching $g \geq 0$ *handles*; g is the genus.







This talk: Graphs embedded (drawn without crossings) on surfaces, in the field of computational topology.

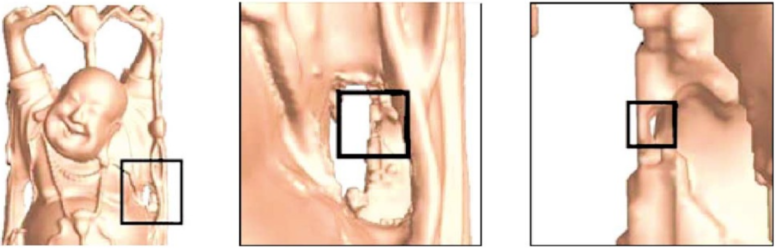
Roadmap of the talk

1. quick survey on topological graphs on surfaces in different fields of mathematics and computer science
2. decision problems: deformations of curves and graphs (homotopy/isotopy)
3. shortest non-contractible closed curves
4. topological decompositions of surfaces
5. other problems solved
6. open problems

1. Topological graphs on surfaces in general

Applications

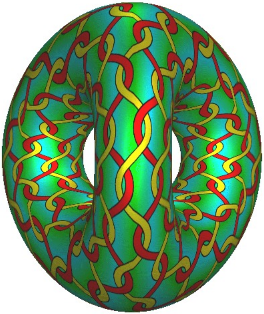
- Topological simplification, remeshing, approximation;



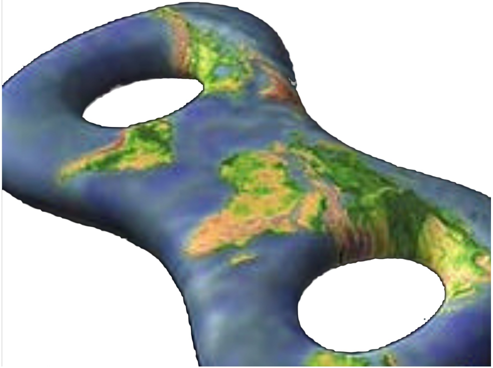
[Wood et al., 2004]

Applications

- Topological simplification, remeshing, approximation;
- parameterization: texture mapping, compression, numerical analysis;

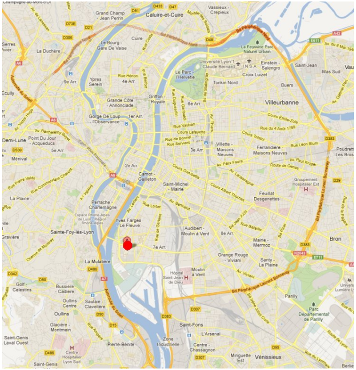


http://www.cs.berkeley.edu/~sequin/CS184/IMGS/mvs.g2.D3.gif



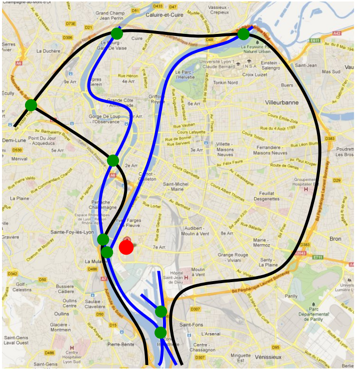
Applications

- Topological simplification, remeshing, approximation;
- parameterization: texture mapping, compression, numerical analysis;
- geographic information systems.



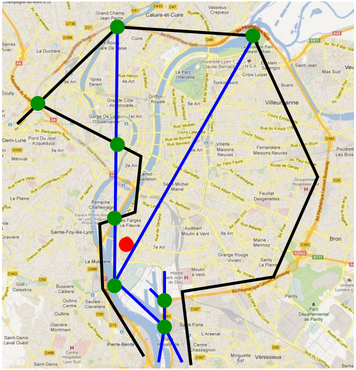
Applications

- Topological simplification, remeshing, approximation;
- parameterization: texture mapping, compression, numerical analysis;
- geographic information systems.



Applications

- Topological simplification, remeshing, approximation;
- parameterization: texture mapping, compression, numerical analysis;
- geographic information systems.



Topology

- Gigantic recent progress in 3-dimensional topology (Poincaré conjecture [Perelman, 2003], . . .);
- "computational" motivations: classify 3-manifolds, decide if a knot is trivial [Haken 1961, Kuperberg 2011, Lackenby 2016, . . .], braids, . . . ;
- algorithms on surface-based structures:
  - algebraic structures (surface groups, mapping class groups, . . .);
  - representation of curves (train tracks, curve complex, pants complex, . . .);
  - deformability of curves to make them disjoint.

Topology

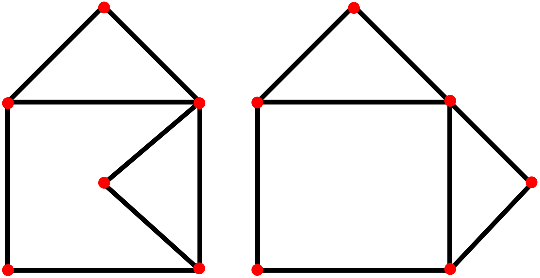
- Gigantic recent progress in 3-dimensional topology (Poincaré conjecture [Perelman, 2003], . . .);
- "computational" motivations: classify 3-manifolds, decide if a knot is trivial [Haken 1961, Kuperberg 2011, Lackenby 2016, . . .], braids, . . . ;
- algorithms on surface-based structures:
  - algebraic structures (surface groups, mapping class groups, . . .);
  - representation of curves (train tracks, curve complex, pants complex, . . .);
  - deformability of curves to make them disjoint.

In computational topology. . .

- algorithmically more precise;
- topologically more elementary;
- more "concrete" problems (?).

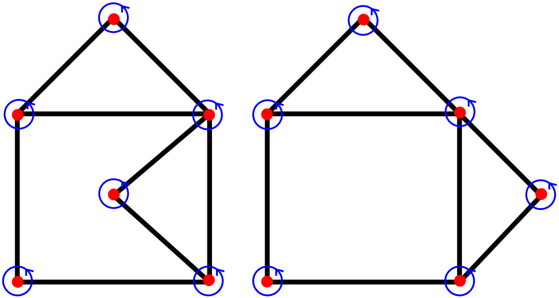
Combinatorial maps (=rotation system)

- A graph is cellularly embedded if its faces are disks.
- Combinatorial maps represent cellular embeddings combinatorially.



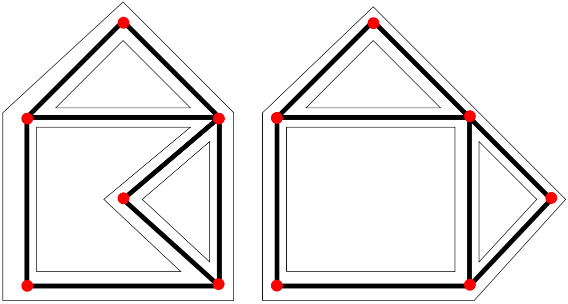
Combinatorial maps (=rotation system)

- A graph is cellularly embedded if its faces are disks.
- Combinatorial maps represent cellular embeddings combinatorially.



Combinatorial maps (=rotation system)

- A graph is cellularly embedded if its faces are disks.
- Combinatorial maps represent cellular embeddings combinatorially.



Enumerative combinatorics

Combinatorial maps (=rotation system)

- A graph is cellularly embedded if its faces are disks.
- Combinatorial maps represent cellular embeddings combinatorially.

Enumeration

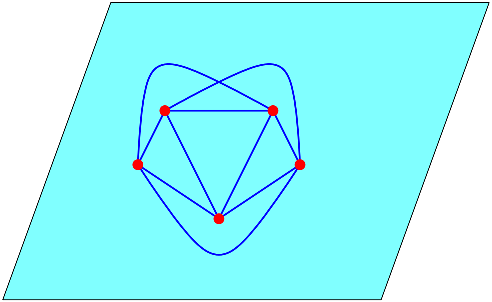
Given g, n, count (exactly or asymptotically) combinatorial maps with genus g and n vertices: rooted / triangulations or quadrangulations / cut graphs / ...

Typical and limit properties

- Properties of a random map, diameter, etc.
- Scaling limits: limits of random maps.

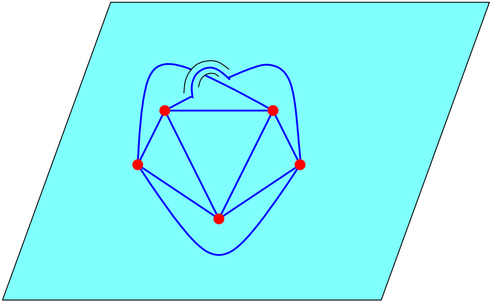
Topological graph theory

Natural generalization of planar graphs: Every graph can be embedded on some surface.



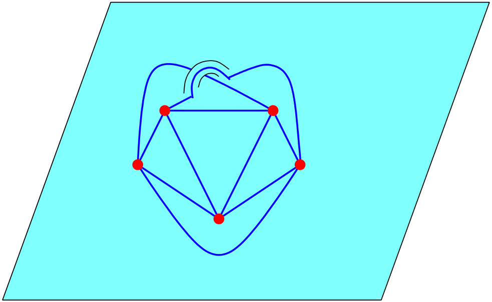
Topological graph theory

Natural generalization of planar graphs: Every graph can be embedded on some surface.



Topological graph theory

Natural generalization of planar graphs: Every graph can be embedded on some surface.



Testing whether a graph with n vertices and edges embeds on a surface of genus g:

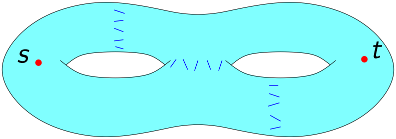
- running time $2^{\text{poly}(g)} \cdot n$ [Mohar, 1996...];
- NP-hard (no polynomial-time algorithm unless P=NP) if g is part of the input;
- space complexity, approximation of the genus, ...

Graph algorithms

General recipe

- Take any graph algorithm problem;
- study it in the specific case where the input graph is embedded in the plane;
- or more generally on a fixed surface.

Examples: Minimum cut, maximum flow, induced cycles, graph isomorphism, minimum multicut, Steiner tree, TSP, ...

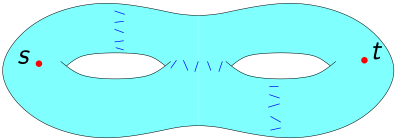


Graph algorithms

General recipe

- Take any graph algorithm problem;
- study it in the specific case where the input graph is embedded in the plane;
- or more generally on a fixed surface.

Examples: Minimum cut, maximum flow, induced cycles, graph isomorphism, minimum multicut, Steiner tree, TSP, . . .



Planar graphs are rather limited (add one edge and you cannot do anything).

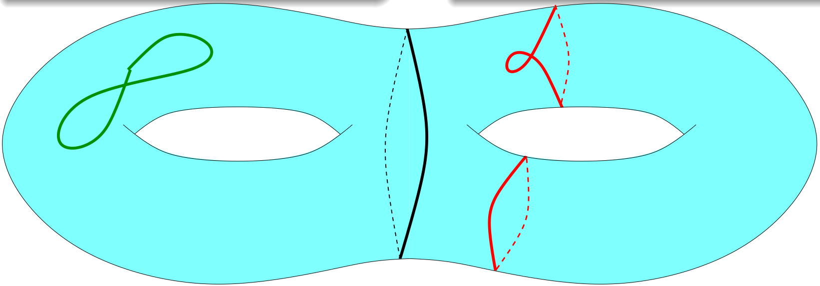
2. Decision problems: homotopy and isotopy

Testing homotopy

Let G be a graph cellularly embedded on $\mathscr{S}$.

Given a closed curve γ in G, decide whether γ is contractible in $\mathscr{S}$ (can be continuously deformed to a point).

Given two closed curves γ and δ in G, decide whether they are freely homotopic in $\mathscr{S}$ (can be deformed one into the other on $\mathscr{S}$).



Testing homotopy

Let G be a graph cellularly embedded on $\mathscr{S}$.

Given a closed curve γ in G, decide whether γ is contractible in $\mathscr{S}$ (can be continuously deformed to a point).

Given two closed curves γ and δ in G, decide whether they are freely homotopic in $\mathscr{S}$ (can be deformed one into the other on $\mathscr{S}$).

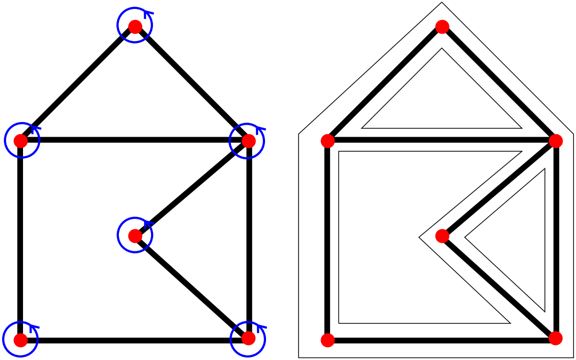
Remarks

- These problems date back to Poincaré (*word problem, conjugacy problem for surface groups*).
- In 1912, Dehn gives a solution, which translates to a polynomial-time algorithm.
- There is more to be said: These problems are solvable in linear time [Dey, Guha 1999][Lazarus, Rivaud, 2012][Erickson, Whittlesey 2013].

Data structures

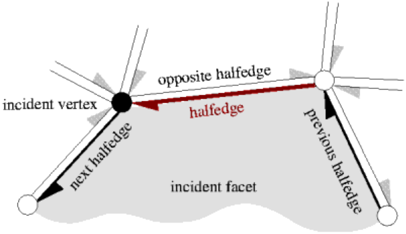
Data structures

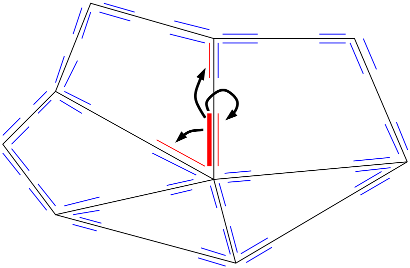
- of size linear in the number n of edges
- allowing to do reasonable operations efficiently:
  - visit the vertices/edges/faces in $O(n)$ time,
  - degree of a face/vertex in $O(\text{degree})$, ...



Data structures

- of size linear in the number n of edges
- allowing to do reasonable operations efficiently:
  - visit the vertices/edges/faces in $O(n)$ time,
  - degree of a face/vertex in $O(\text{degree})$, ...





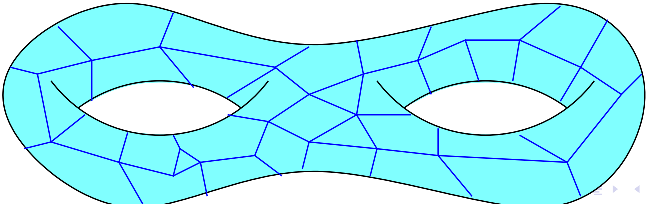
Reduction

[Lazarus, Rivaud, 2012]: WLOG, G has

- two vertices, of degree $4g$,
- $4g$ edges, and
- $2g$ faces, which are quadrilaterals.

Proof

- WLOG, there is a single vertex (edge contractions).
- WLOG, there is a single face (edge deletions).
- By Euler's formula $v - e + f = 2 - 2g$, the graph has $2g$ edges.
- The curves γ (and δ) use edges that were removed, but we can transform them by creating a new vertex.



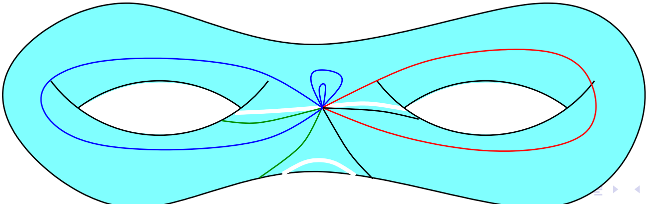
Reduction

[Lazarus, Rivaud, 2012]: WLOG, G has

- two vertices, of degree $4g$,
- $4g$ edges, and
- $2g$ faces, which are quadrilaterals.

Proof

- WLOG, there is a single vertex (edge contractions).
- WLOG, there is a single face (edge deletions).
- By Euler's formula $v - e + f = 2 - 2g$, the graph has $2g$ edges.
- The curves γ (and δ) use edges that were removed, but we can transform them by creating a new vertex.



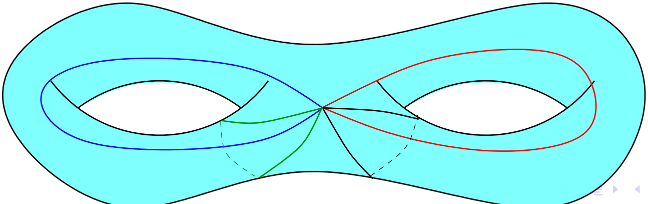
Reduction

[Lazarus, Rivaud, 2012]: WLOG, G has

- two vertices, of degree $4g$,
- $4g$ edges, and
- $2g$ faces, which are quadrilaterals.

Proof

- WLOG, there is a single vertex (edge contractions).
- WLOG, there is a single face (edge deletions).
- By Euler's formula $v - e + f = 2 - 2g$, the graph has $2g$ edges.
- The curves γ (and δ) use edges that were removed, but we can transform them by creating a new vertex.



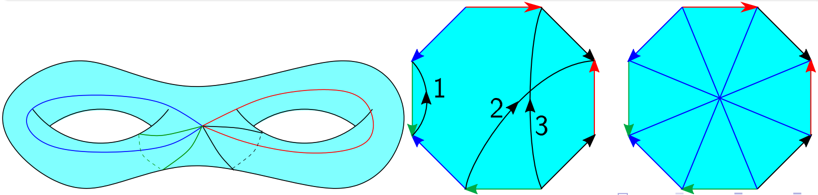
Reduction

[Lazarus, Rivaud, 2012]: WLOG, G has

- two vertices, of degree $4g$,
- $4g$ edges, and
- $2g$ faces, which are quadrilaterals.

Proof

- WLOG, there is a single vertex (edge contractions).
- WLOG, there is a single face (edge deletions).
- By Euler's formula $v - e + f = 2 - 2g$, the graph has $2g$ edges.
- The curves γ (and δ) use edges that were removed, but we can transform them by creating a new vertex.



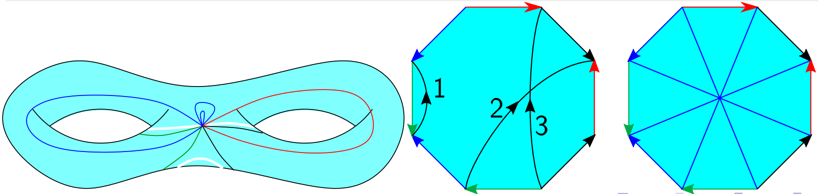
Reduction

[Lazarus, Rivaud, 2012]: WLOG, G has

- two vertices, of degree $4g$,
- $4g$ edges, and
- $2g$ faces, which are quadrilaterals.

Proof

- WLOG, there is a single vertex (edge contractions).
- WLOG, there is a single face (edge deletions).
- By Euler's formula $v - e + f = 2 - 2g$, the graph has $2g$ edges.
- The curves γ (and δ) use edges that were removed, but we can transform them by creating a new vertex.



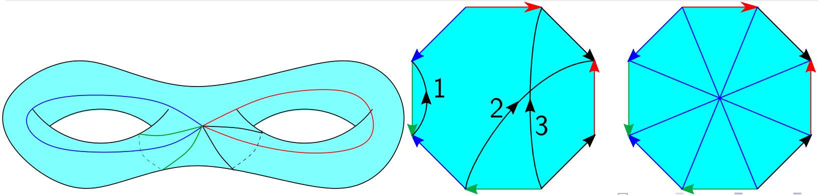
Reduction

[Lazarus, Rivaud, 2012]: WLOG, G has

- two vertices, of degree $4g$,
- $4g$ edges, and
- $2g$ faces, which are quadrilaterals.

Proof

- WLOG, there is a single vertex (edge contractions).
- WLOG, there is a single face (edge deletions).
- By Euler's formula $v - e + f = 2 - 2g$, the graph has $2g$ edges.
- The curves γ (and δ) use edges that were removed, but we can transform them by creating a new vertex.

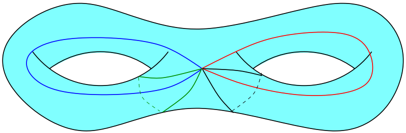


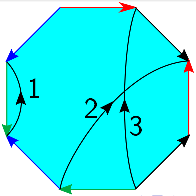
[Lazarus, Rivaud, 2012]: WLOG, G has

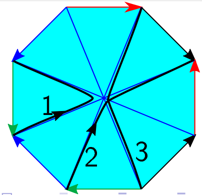
- two vertices, of degree $4g$,
- $4g$ edges, and
- $2g$ faces, which are quadrilaterals.

Proof

- WLOG, there is a single vertex (edge contractions).
- WLOG, there is a single face (edge deletions).
- By Euler's formula $v - e + f = 2 - 2g$, the graph has $2g$ edges.
- The curves γ (and δ) use edges that were removed, but we can transform them by creating a new vertex.







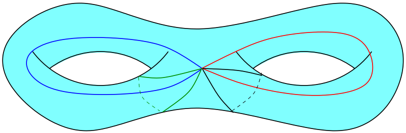
Reduction

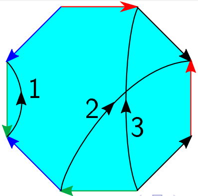
[Lazarus, Rivaud, 2012]: WLOG, G has

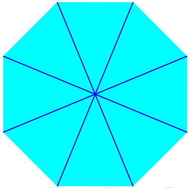
- two vertices, of degree $4g$,
- $4g$ edges, and
- $2g$ faces, which are quadrilaterals.

Proof

- WLOG, there is a single vertex (edge contractions).
- WLOG, there is a single face (edge deletions).
- By Euler's formula $v - e + f = 2 - 2g$, the graph has $2g$ edges.
- The curves γ (and δ) use edges that were removed, but we can transform them by creating a new vertex.



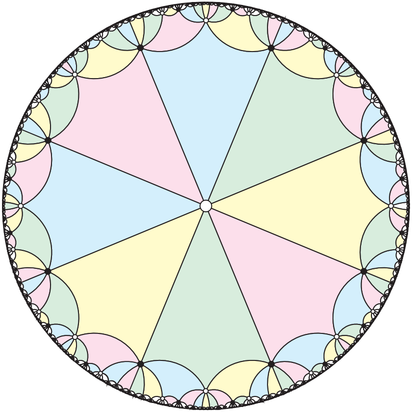


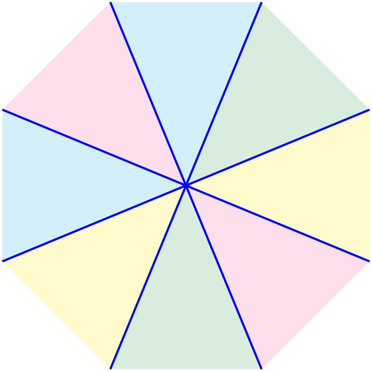


Universal cover $\tilde{\mathscr{S}}$

A regular tiling of squares meeting $4g$ at a vertex:

- Every path in $\mathscr{S}$ lifts to a path in $\tilde{\mathscr{S}}$;
- a closed curve is contractible in $\mathscr{S}$ iff it lifts to a closed curve.

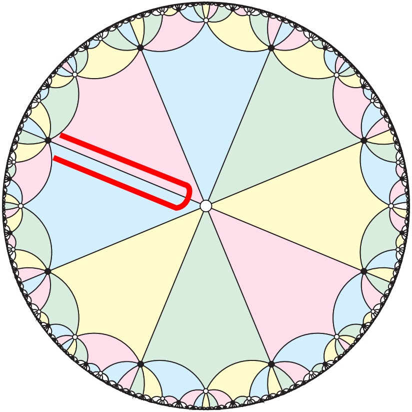




Universal cover $\tilde{\mathscr{S}}$

A regular tiling of squares meeting $4g$ at a vertex:

- Every path in $\mathscr{S}$ lifts to a path in $\tilde{\mathscr{S}}$;
- a closed curve is contractible in $\mathscr{S}$ iff it lifts to a closed curve.



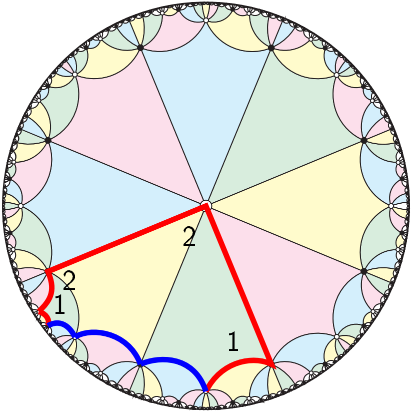
Result from geometric group theory [Gersten, Short, 1990]

In this tiling, every non-trivial closed curve has either a spur or a bracket.

Universal cover $\tilde{\mathscr{S}}$

A regular tiling of squares meeting $4g$ at a vertex:

- Every path in $\mathscr{S}$ lifts to a path in $\tilde{\mathscr{S}}$;
- a closed curve is contractible in $\mathscr{S}$ iff it lifts to a closed curve.



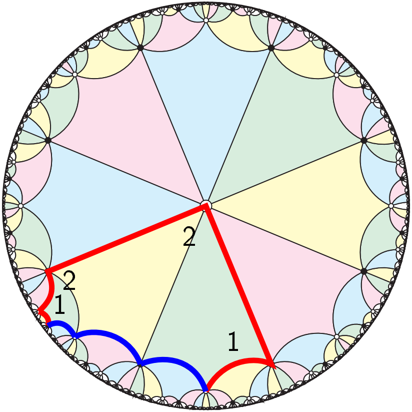
Result from geometric group theory [Gersten, Short, 1990]

In this tiling, every non-trivial closed curve has either a spur or a bracket.

Universal cover $\tilde{\mathscr{S}}$

A regular tiling of squares meeting $4g$ at a vertex:

- Every path in $\mathscr{S}$ lifts to a path in $\tilde{\mathscr{S}}$;
- a closed curve is contractible in $\mathscr{S}$ iff it lifts to a closed curve.



Result from geometric group theory [Gersten, Short, 1990]

In this tiling, every non-trivial closed curve has either a spur or a bracket.

Algorithm [Erickson, Whittlesey 2013]

Remove iteratively spurs and brackets whenever possible!

Extension: Geometric intersection numbers

The game

- Given a curve γ, move it continuously (by a homotopy) to minimize its number of crossings.
- Given two curves γ and δ, move them continuously (by a homotopy) to minimize the number of crossings between them.

Result [Despré, Lazarus, 2017]

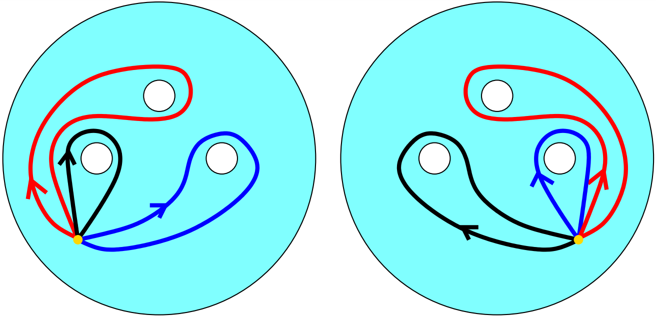
Computing these numbers is doable in near-linear time.

Proof

Similar spirit (ask Vincent at the coffee break).

Testing isotopy

An isotopy of an embedded curve or graph is a homotopy (deformation) that remains crossing-free at all times.



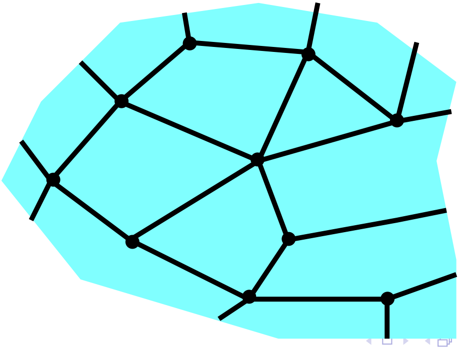
Problem

Given an abstract graph G embedded in two different ways, G_1 and G_2, on $\mathscr{S}$, does there exist a continuous family of embeddings between G_1 and G_2?

This is possible in linear in the input size [CdV, de Mesmay, 2014].

Storing graphs on surfaces

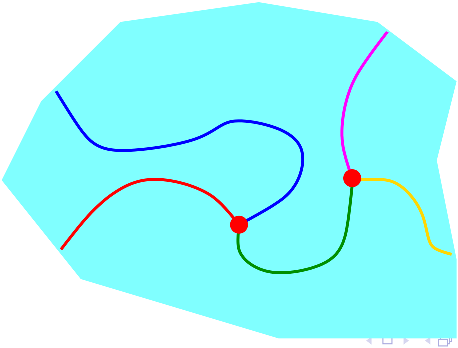
- Let M be a fixed graph (cellularly) embedded on $\mathscr{S}$.
- The graphs G_1 and G_2 are in general position with respect to M.
- We store the combinatorial map of the overlay of M and G_1, and similarly the overlay of M and G_2.



Data structures for storing graphs on surfaces

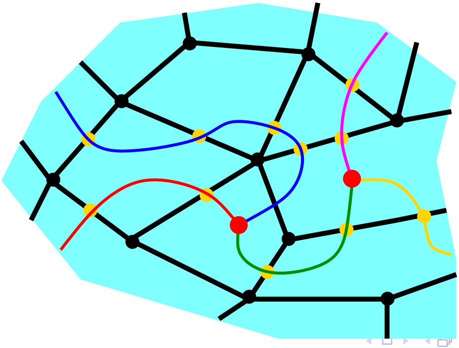
Storing graphs on surfaces

- Let M be a fixed graph (cellularly) embedded on $\mathscr{S}$.
- The graphs G_1 and G_2 are in general position with respect to M.
- We store the combinatorial map of the overlay of M and G_1, and similarly the overlay of M and G_2.

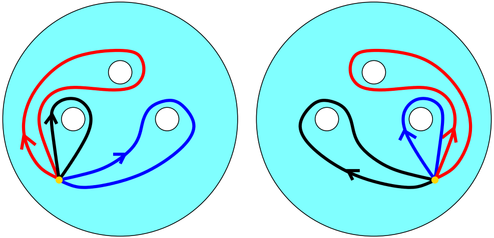


Storing graphs on surfaces

- Let M be a fixed graph (cellularly) embedded on $\mathscr{S}$.
- The graphs G_1 and G_2 are in general position with respect to M.
- We store the combinatorial map of the overlay of M and G_1, and similarly the overlay of M and G_2.



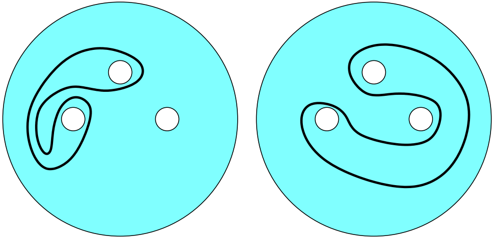
Proof sketch



Some clearly necessary conditions that turn out to be sufficient [Ladegaillerie, 1984]

1. Oriented homeomorphism of $\mathscr{S}$ mapping G_1 to G_2;
2. each cycle in G_1 is homotopic to its counterpart in G_2.

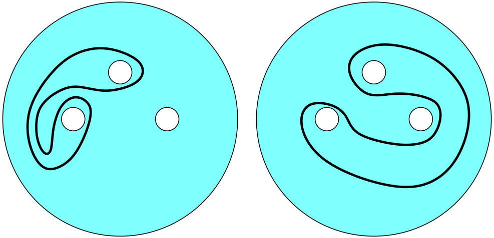
Proof sketch



Some clearly necessary conditions that turn out to be sufficient
[Ladegaillerie, 1984]

1. Oriented homeomorphism of $\mathscr{S}$ mapping G_1 to G_2;
2. each cycle in G_1 is homotopic to its counterpart in G_2.

Proof sketch



Some clearly necessary conditions that turn out to be sufficient [Ladegaillerie, 1984]

1. Oriented homeomorphism of $\mathscr{S}$ mapping G_1 to G_2;
2. each cycle in G_1 is homotopic to its counterpart in G_2.

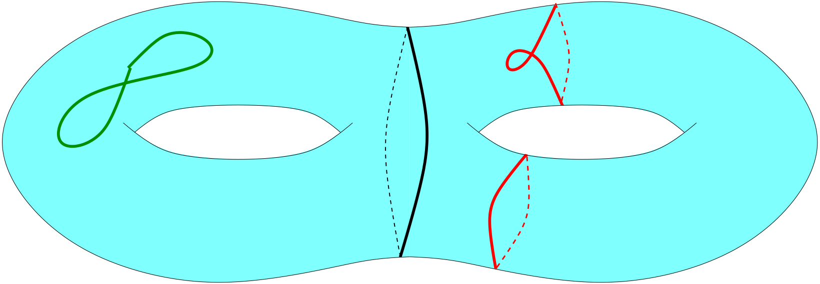
- algorithmically: 1 easy, 2 as before;
- difficulty: small family of cycles for 2;
- tools: universal cover, hyperbolic geometry, Reidemeister moves, [Ringel, 1955], [de Graaf and Schrijver, 1997], ...

3. Shortest non-contractible closed curves

Goal

Problem

Compute a shortest non-contractible closed curve (a.k.a. systole, a.k.a. edge-width).



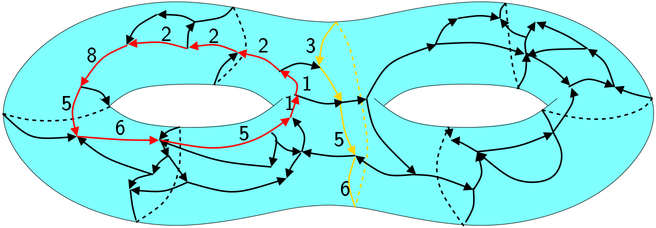
Remark

Similar algorithms for shortest non-separating closed curve.

In a graph cellularly embedded on $\mathscr{S}$: many results!

n: complexity of G g: genus k: output size

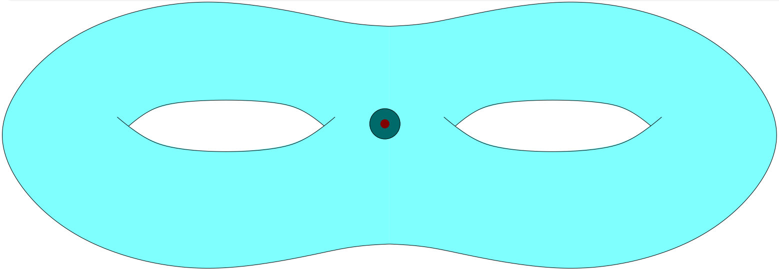
	directed	directed
weighted	$\boldsymbol{O(n^2 \log n)}$ [Erickson–Har-Peled'04] $O(g^{3/2} n^{3/2} \log n)$ [Cabello–Mohar'07] $g^{O(g)} n \log n$ [Kutz'06] $O(g^3 n \log n)$ [Cabello–Chambers'07] $\boldsymbol{O(g^2 n \log n)}$ [Cabello–Chambers–Erickson'13] $\boldsymbol{2^{O(g)} n \log \log n}$ [Fox'13] $\boldsymbol{O(gn \log n)}$ for 2-approx [Erickson–Har-Peled'04]	$O(n^2 \log n)$ [Cab–CdV–Laz'10] $\boldsymbol{O(g^{1/2} n^{3/2} \log n)}$ [Cab–CdV–Laz'10] $g^{O(g)} n \log n$ [Erickson'11] $\boldsymbol{O(g^3 n \log n)}$ [Fox'13]
unweighted	$O(n^3)$ [Thomassen'90] $\boldsymbol{O(n^2)}$ [Cab–CdV–Laz'10] $\boldsymbol{O(gnk)}$ [Cab–CdV–Laz'10] $\boldsymbol{O(gn/\varepsilon)}$ for $(1+\varepsilon)$-approx [Cab–CdV–Laz'16]	$\boldsymbol{O(n^2)}$ [Cab–CdV–Laz'10] $\boldsymbol{O(gnk)}$ [Cab–CdV–Laz'16]



Cut locus

Intermediate step

Let us compute a shortest closed curve passing through a fixed basepoint b.



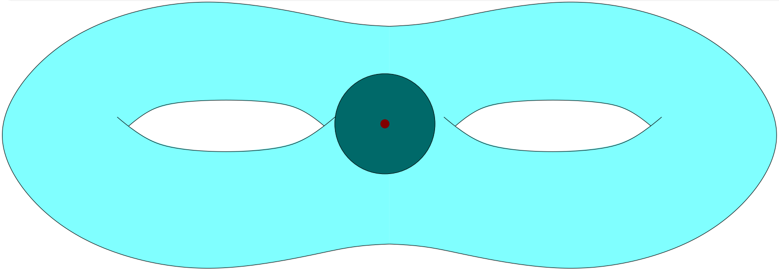
Construction

- Grow a disk around b; the cut locus C is the set of points where the disk self-collides.
- Formally, it is the (closure of the) set of points with several shortest paths to b.
- $\mathscr{S} \setminus C$ is (homeomorphic to) an open disk.

Cut locus

Intermediate step

Let us compute a shortest closed curve passing through a fixed basepoint b.



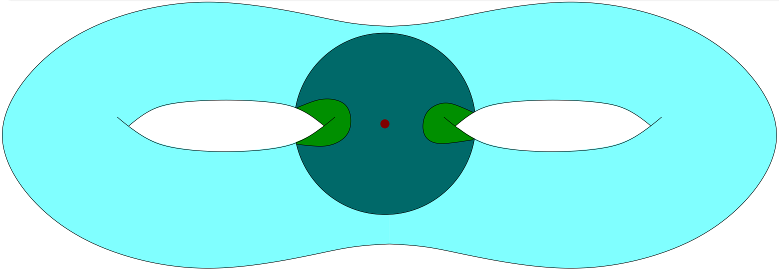
Construction

- Grow a disk around b; the cut locus C is the set of points where the disk self-collides.
- Formally, it is the (closure of the) set of points with several shortest paths to b.
- $\mathscr{S} \setminus C$ is (homeomorphic to) an open disk.

Cut locus

Intermediate step

Let us compute a shortest closed curve passing through a fixed basepoint b.



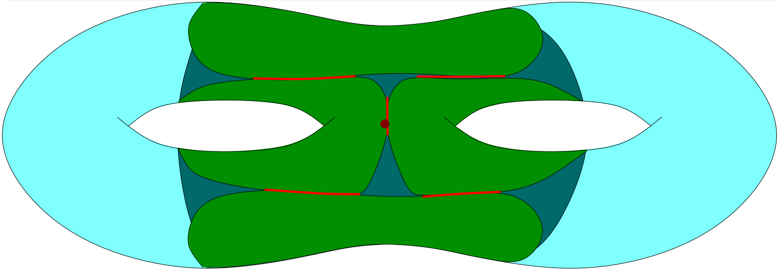
Construction

- Grow a disk around b; the cut locus C is the set of points where the disk self-collides.
- Formally, it is the (closure of the) set of points with several shortest paths to b.
- $\mathscr{S} \setminus C$ is (homeomorphic to) an open disk.

Cut locus

Intermediate step

Let us compute a shortest closed curve passing through a fixed basepoint b.



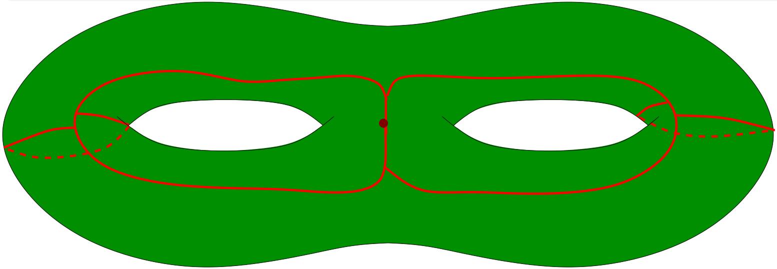
Construction

- Grow a disk around b; the cut locus C is the set of points where the disk self-collides.
- Formally, it is the (closure of the) set of points with several shortest paths to b.
- $\mathscr{S} \setminus C$ is (homeomorphic to) an open disk.

Cut locus

Intermediate step

Let us compute a shortest closed curve passing through a fixed basepoint b.



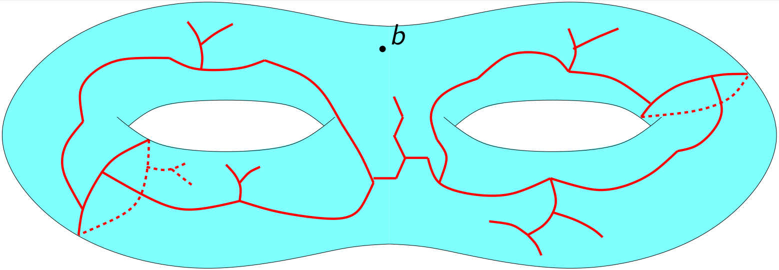
Construction

- Grow a disk around b; the cut locus C is the set of points where the disk self-collides.
- Formally, it is the (closure of the) set of points with several shortest paths to b.
- $\mathscr{S} \setminus C$ is (homeomorphic to) an open disk.

Cut locus

Intermediate step

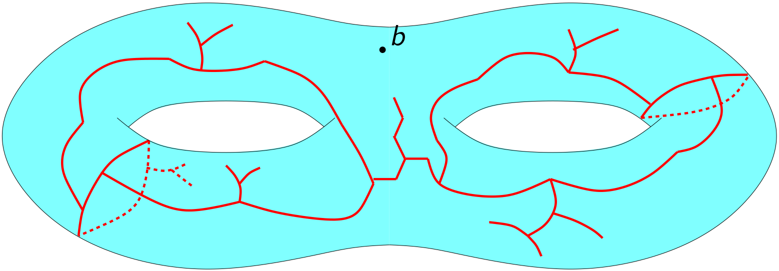
Let us compute a shortest closed curve passing through a fixed basepoint b.



Construction

- Grow a disk around b; the cut locus C is the set of points where the disk self-collides.
- Formally, it is the (closure of the) set of points with several shortest paths to b.
- $\mathscr{S} \setminus C$ is (homeomorphic to) an open disk.

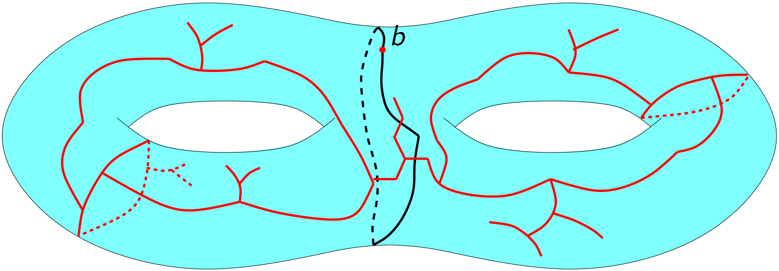
Generic algorithm



Lemma

- A shortest non-contractible loop based at b crosses the cut locus C exactly once;
- and is a shortest loop among those crossing an edge e of C such that no connected component of $C - e$ is a tree.

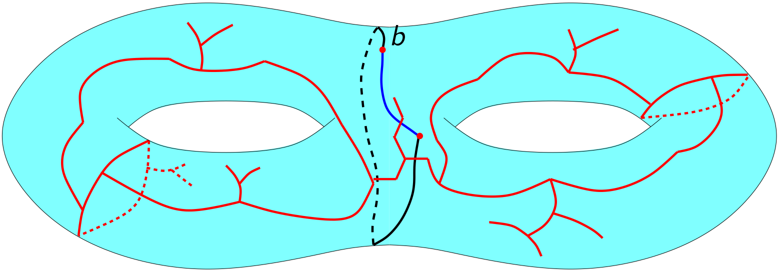
Generic algorithm



Lemma

- A shortest non-contractible loop based at b crosses the cut locus C exactly once;
- and is a shortest loop among those crossing an edge e of C such that no connected component of $C - e$ is a tree.

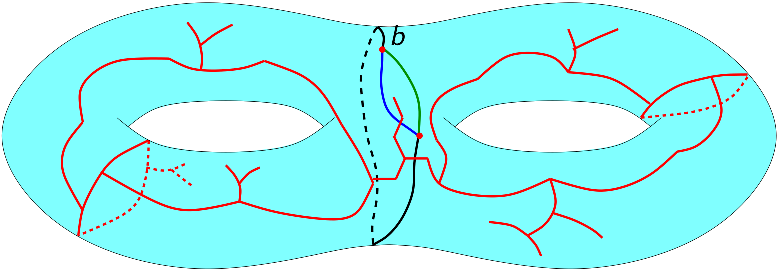
Generic algorithm



Lemma

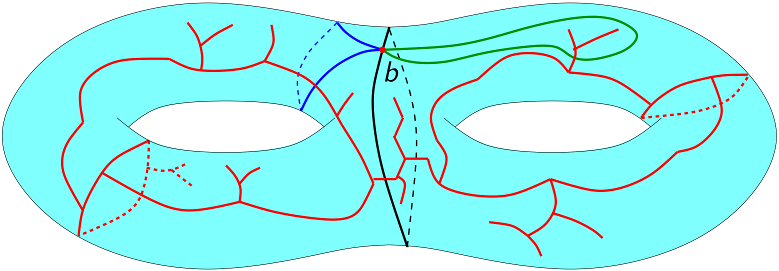
- A shortest non-contractible loop based at b crosses the cut locus C exactly once;
- and is a shortest loop among those crossing an edge e of C such that no connected component of $C - e$ is a tree.

Generic algorithm



Lemma

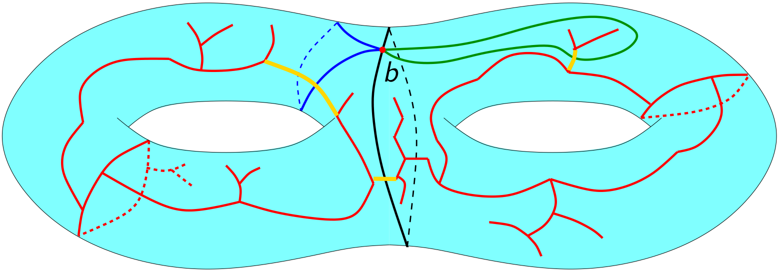
- A shortest non-contractible loop based at b crosses the cut locus C exactly once;
- and is a shortest loop among those crossing an edge e of C such that no connected component of $C - e$ is a tree.



Lemma

- A shortest non-contractible loop based at b crosses the cut locus C exactly once;
- and is a shortest loop among those crossing an edge e of C such that no connected component of $C - e$ is a tree.

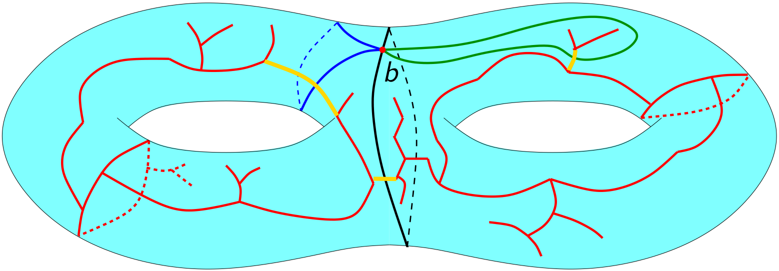
Generic algorithm



Lemma

- A shortest non-contractible loop based at b crosses the cut locus C exactly once;
- and is a shortest loop among those crossing an edge e of C such that no connected component of $C - e$ is a tree.

Generic algorithm



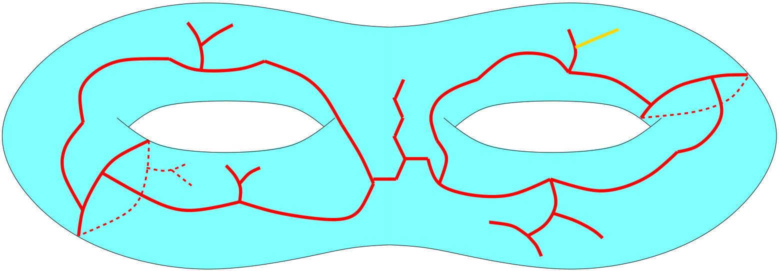
Lemma

- A shortest non-contractible loop based at b crosses the cut locus C exactly once;
- and is a shortest loop among those crossing an edge e of C such that no connected component of $C - e$ is a tree.

Generic algorithm [Erickson and Whittlesey, 2005] [CdV, 2009]

- Prune C, removing the edges e such that one connected component of $C - e$ is a tree. Let C' be the pruned cut locus.
- return the shortest loop from b crossing C' exactly once.

Generic algorithm



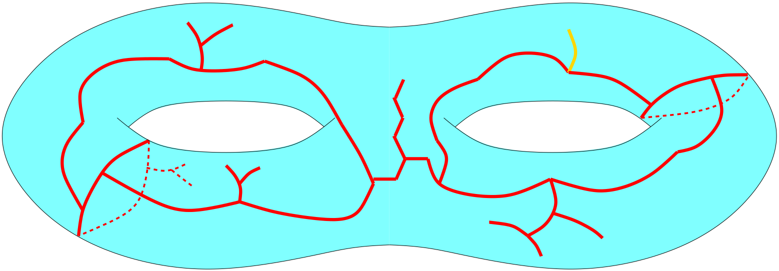
Lemma

- A shortest non-contractible loop based at b crosses the cut locus C exactly once;
- and is a shortest loop among those crossing an edge e of C such that no connected component of $C - e$ is a tree.

Generic algorithm [Erickson and Whittlesey, 2005] [CdV, 2009]

- Prune C, removing the edges e such that one connected component of $C - e$ is a tree. Let C' be the pruned cut locus.
- return the shortest loop from b crossing C' exactly once.

Generic algorithm

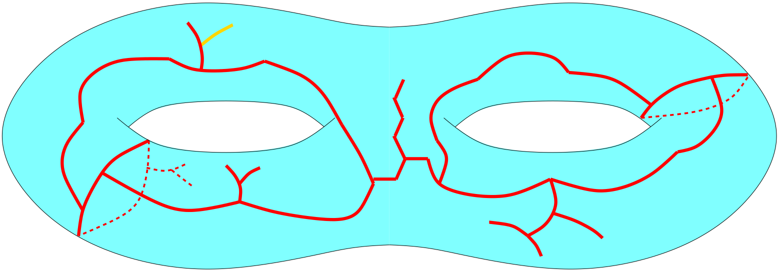


Lemma

- A shortest non-contractible loop based at b crosses the cut locus C exactly once;
- and is a shortest loop among those crossing an edge e of C such that no connected component of $C - e$ is a tree.

Generic algorithm [Erickson and Whittlesey, 2005] [CdV, 2009]

- Prune C, removing the edges e such that one connected component of $C - e$ is a tree. Let C' be the pruned cut locus.
- return the shortest loop from b crossing C' exactly once.



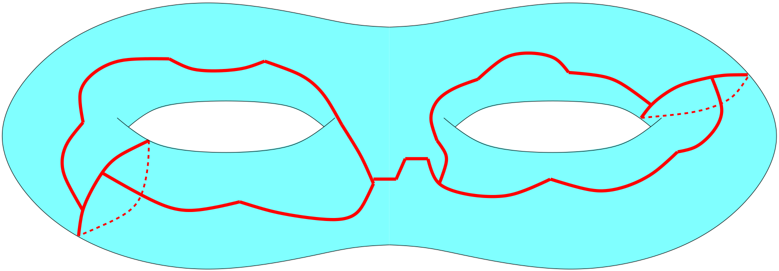
Lemma

- A shortest non-contractible loop based at b crosses the cut locus C exactly once;
- and is a shortest loop among those crossing an edge e of C such that no connected component of $C - e$ is a tree.

Generic algorithm [Erickson and Whittlesey, 2005] [CdV, 2009]

- Prune C, removing the edges e such that one connected component of $C - e$ is a tree. Let C' be the pruned cut locus.
- return the shortest loop from b crossing C' exactly once.

Generic algorithm

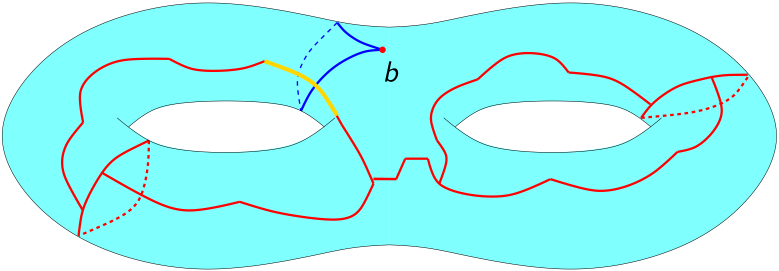


Lemma

- A shortest non-contractible loop based at b crosses the cut locus C exactly once;
- and is a shortest loop among those crossing an edge e of C such that no connected component of $C - e$ is a tree.

Generic algorithm [Erickson and Whittlesey, 2005] [CdV, 2009]

- Prune C, removing the edges e such that one connected component of $C - e$ is a tree. Let C' be the pruned cut locus.
- return the shortest loop from b crossing C' exactly once.



Lemma

- A shortest non-contractible loop based at b crosses the cut locus C exactly once;
- and is a shortest loop among those crossing an edge e of C such that no connected component of $C - e$ is a tree.

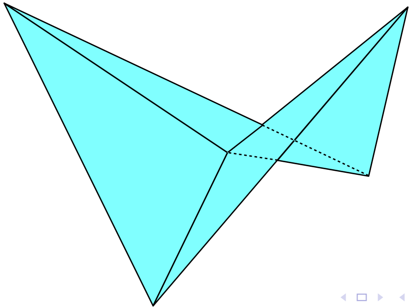
Generic algorithm [Erickson and Whittlesey, 2005] [CdV, 2009]

- Prune C, removing the edges e such that one connected component of $C - e$ is a tree. Let C' be the pruned cut locus.
- return the shortest loop from b crossing C' exactly once.

Polyhedral surfaces

Everything relies on the computation of the cut locus!

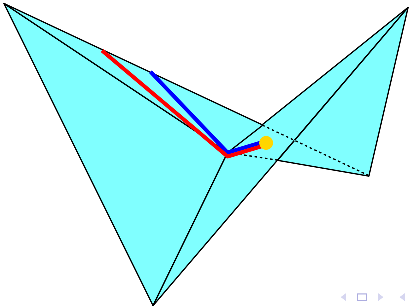
- [Chen and Han, 1996]: $O(n^2)$, where n is the number of triangles (or the total complexity of the polygons);
- Thus, algorithm with running-time $O(n^2)$ when b is fixed. But the loop is not necessarily simple, it may "run along itself".



Polyhedral surfaces

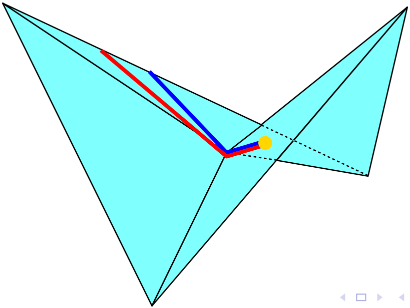
Everything relies on the computation of the cut locus!

- [Chen and Han, 1996]: $O(n^2)$, where n is the number of triangles (or the total complexity of the polygons);
- Thus, algorithm with running-time $O(n^2)$ when b is fixed. But the loop is not necessarily simple, it may "run along itself".



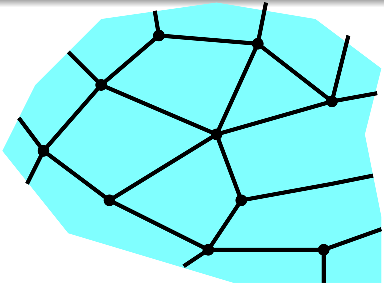
Everything relies on the computation of the cut locus!

- [Chen and Han, 1996]: $O(n^2)$, where n is the number of triangles (or the total complexity of the polygons);
- Thus, algorithm with running-time $O(n^2)$ when b is fixed. But the loop is not necessarily simple, it may "run along itself".
- Observation: A shortest non-contractible closed curve passes through a vertex. Thus, a shortest non-contractible loop (without fixing b) can be computed in $O(n^3)$.



Cross-metric surfaces [CdV, Erickson, 2006]

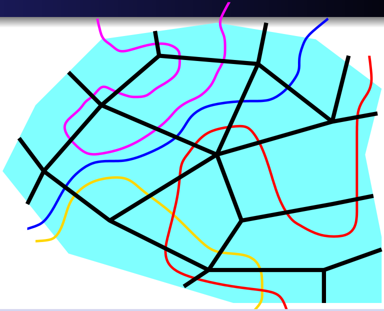
A discretization of metric surfaces, suitable for many purposes.



Storing curves on surfaces

- Let M be an edge-weighted graph (cellularly) embedded on $\mathscr{S}$.

Cross-metric surfaces



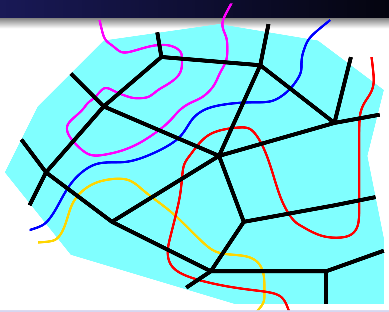
Cross-metric surfaces [CdV, Erickson, 2006]

A discretization of metric surfaces, suitable for many purposes.

Storing curves on surfaces

- Let M be an edge-weighted graph (cellularly) embedded on $\mathscr{S}$.
- Curves are in general position with respect to M.

Cross-metric surfaces



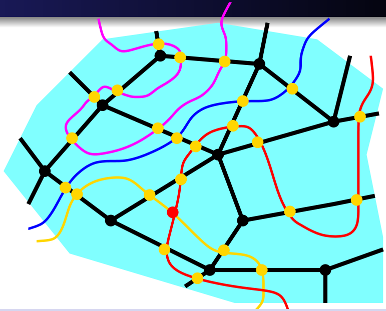
Cross-metric surfaces [CdV, Erickson, 2006]

A discretization of metric surfaces, suitable for many purposes.

Storing curves on surfaces

- Let M be an edge-weighted graph (cellularly) embedded on $\mathscr{S}$.
- Curves are in general position with respect to M.
- The length of a curve is, by definition, the sum of the weights of the edges of M crossed by that curve.

Cross-metric surfaces



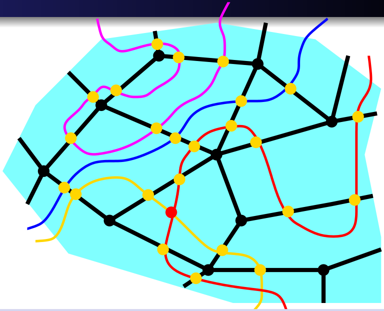
Cross-metric surfaces [CdV, Erickson, 2006]

A discretization of metric surfaces, suitable for many purposes.

Storing curves on surfaces

- Let M be an edge-weighted graph (cellularly) embedded on $\mathscr{S}$.
- Curves are in general position with respect to M.
- The length of a curve is, by definition, the sum of the weights of the edges of M crossed by that curve.
- We store the combinatorial map of the overlay of M and the curves.

Cross-metric surfaces



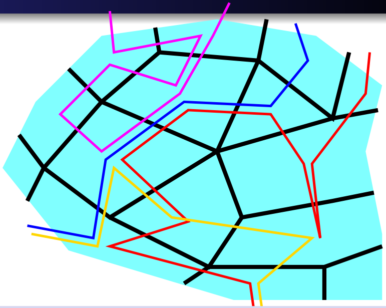
Cross-metric surfaces [CdV, Erickson, 2006]

A discretization of metric surfaces, suitable for many purposes.

Storing curves on surfaces

- Let M be an edge-weighted graph (cellularly) embedded on $\mathscr{S}$.
- Curves are in general position with respect to M.
- The length of a curve is, by definition, the sum of the weights of the edges of M crossed by that curve.
- We store the combinatorial map of the overlay of M and the curves.
- Refinement of walks stored in the dual graph M^*. Easy to compute, e.g., shortest paths (Dijkstra in M^*).

Cross-metric surfaces



Cross-metric surfaces [CdV, Erickson, 2006]

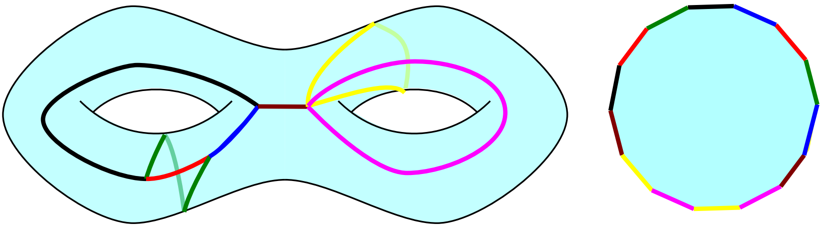
A discretization of metric surfaces, suitable for many purposes.

Storing curves on surfaces

- Let M be an edge-weighted graph (cellularly) embedded on $\mathscr{S}$.
- Curves are in general position with respect to M.
- The length of a curve is, by definition, the sum of the weights of the edges of M crossed by that curve.
- We store the combinatorial map of the overlay of M and the curves.
- Refinement of walks stored in the dual graph M^*. Easy to compute, e.g., shortest paths (Dijkstra in M^*).

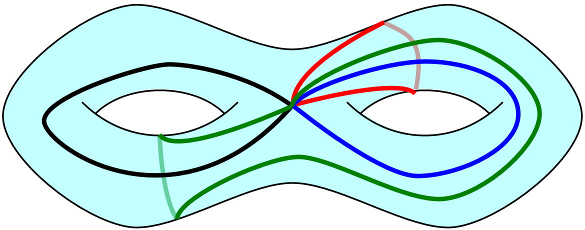
4. Topological decompositions of surfaces

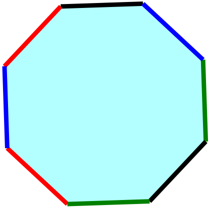
Cut graphs



- cut graph: a graph that cuts $\mathscr{S}$ into a disk.
- system of loops: a one-vertex cut graph.
- canonical system of loops: a one-vertex cut graph in which the loops appear in canonical order.

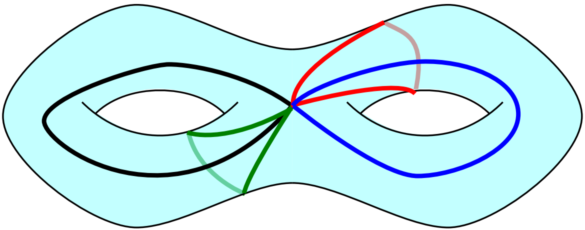
Cut graphs

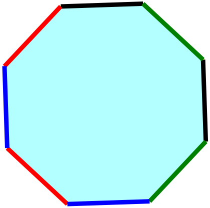




- cut graph: a graph that cuts $\mathscr{S}$ into a disk.
- system of loops: a one-vertex cut graph.
- canonical system of loops: a one-vertex cut graph in which the loops appear in canonical order.

Cut graphs

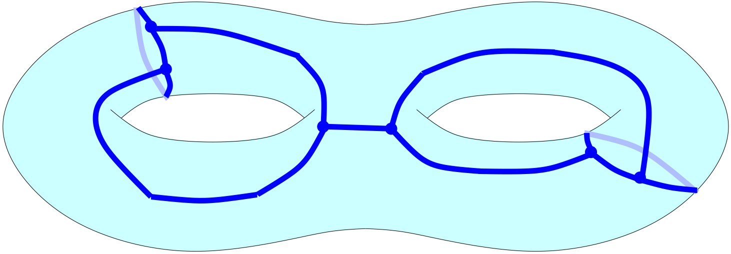




- cut graph: a graph that cuts $\mathscr{S}$ into a disk.
- system of loops: a one-vertex cut graph.
- canonical system of loops: a one-vertex cut graph in which the loops appear in canonical order.

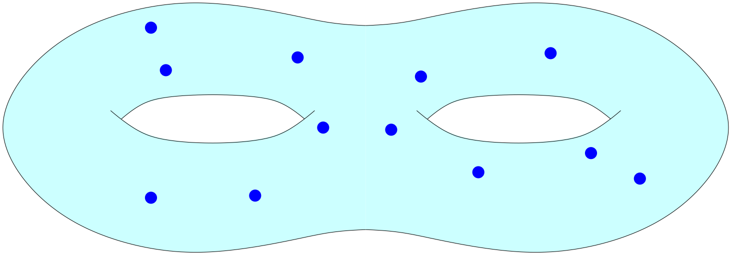
Cut graphs

- Computing a cut graph: easy!
- Shortest cut graph:
  - NP-hard in general [Erickson, Har-Peled, 2004];
  - ε-approximation in $f(g, \varepsilon) \cdot n^3$ [Cohen-Addad and de Mesmay, 2015];
  - easy if one wishes to compute the shortest cut graph with specified vertex set P: doable in $O(n \log n + gn + |P|)$ time [CdV, 2010];
  - in particular, allows to compute the shortest system of loops [Erickson and Whittlesey, 2006].



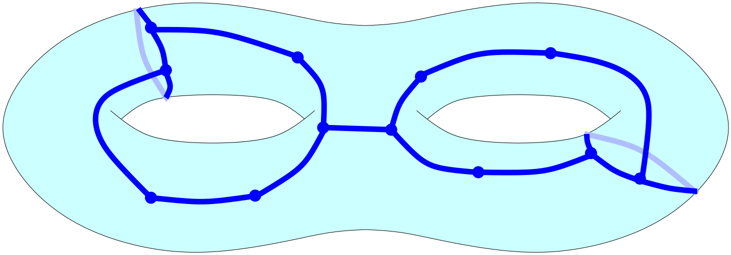
Cut graphs

- Computing a cut graph: easy!
- Shortest cut graph:
  - NP-hard in general [Erickson, Har-Peled, 2004];
  - ε-approximation in $f(g, \varepsilon) \cdot n^3$ [Cohen-Addad and de Mesmay, 2015];
  - easy if one wishes to compute the shortest cut graph with specified vertex set P: doable in $O(n \log n + gn + |P|)$ time [CdV, 2010];
  - in particular, allows to compute the shortest system of loops [Erickson and Whittlesey, 2006].

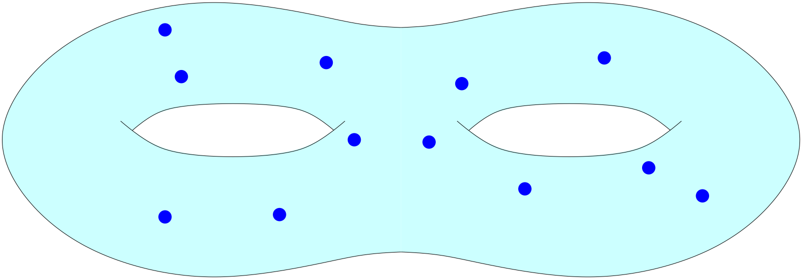


Cut graphs

- Computing a cut graph: easy!
- Shortest cut graph:
  - NP-hard in general [Erickson, Har-Peled, 2004];
  - ε-approximation in $f(g,\varepsilon) \cdot n^3$ [Cohen-Addad and de Mesmay, 2015];
  - easy if one wishes to compute the shortest cut graph with specified vertex set P: doable in $O(n \log n + gn + |P|)$ time [CdV, 2010];
  - in particular, allows to compute the shortest system of loops [Erickson and Whittlesey, 2006].



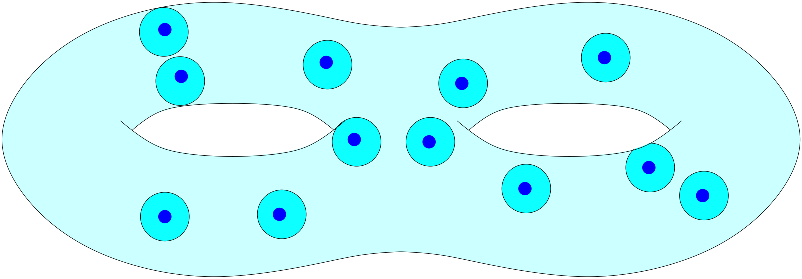
The cut locus w.r.t. P



The cut locus, a.k.a. “Voronoi” diagram

- Grow disks around each point of P simultaneously;

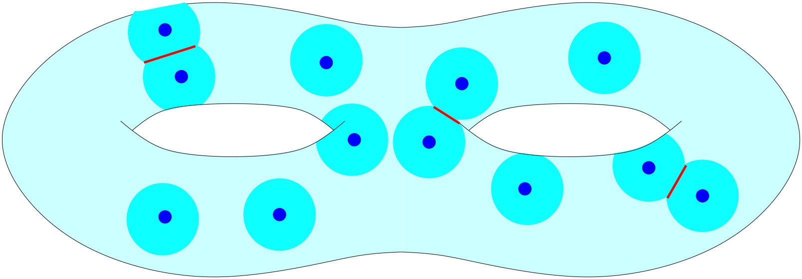
The cut locus w.r.t. P



The cut locus, a.k.a. “Voronoi” diagram

- Grow disks around each point of P simultaneously;

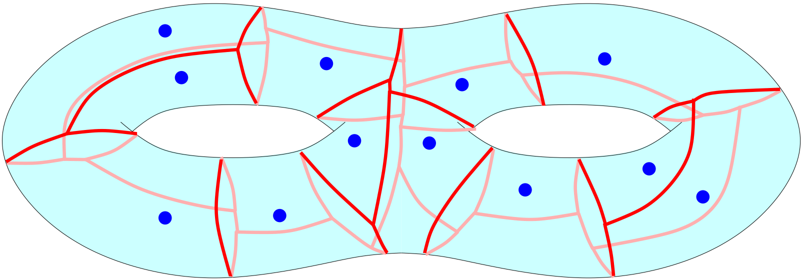
The cut locus w.r.t. P



The cut locus, a.k.a. "Voronoi" diagram

- Grow disks around each point of P simultaneously;
- when disks (self-)collide, stop growing and draw the boundary;

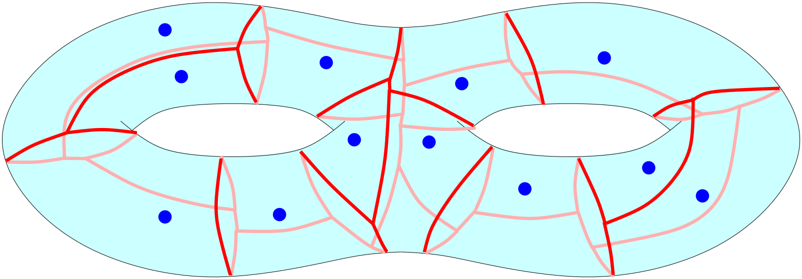
The cut locus w.r.t. P



The cut locus, a.k.a. "Voronoi" diagram

- Grow disks around each point of P simultaneously;
- when disks (self-)collide, stop growing and draw the boundary;

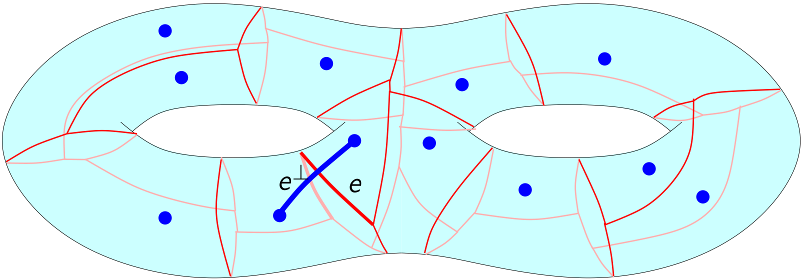
The cut locus w.r.t. P



The cut locus, a.k.a. “Voronoi” diagram

- Grow disks around each point of P simultaneously;
- when disks (self-)collide, stop growing and draw the boundary;
- the cut locus C is the set of all boundaries.

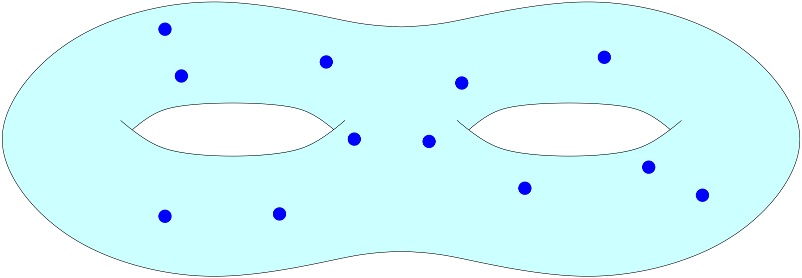
The cut locus w.r.t. P



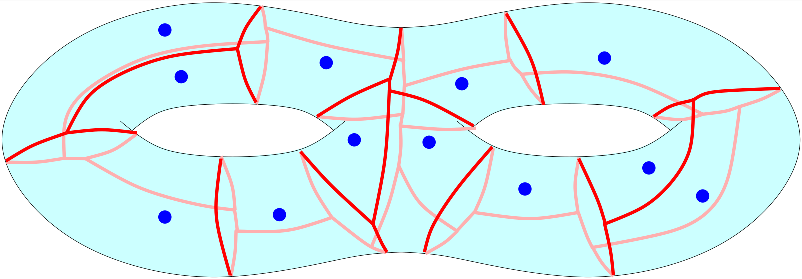
The cut locus, a.k.a. "Voronoi" diagram

- Grow disks around each point of P simultaneously;
- when disks (self-)collide, stop growing and draw the boundary;
- the cut locus C is the set of all boundaries.
- Given an edge e of C, let $e^\perp$ be a "Delaunay" shortest path with endpoints in P that crosses e and no other edge of C.

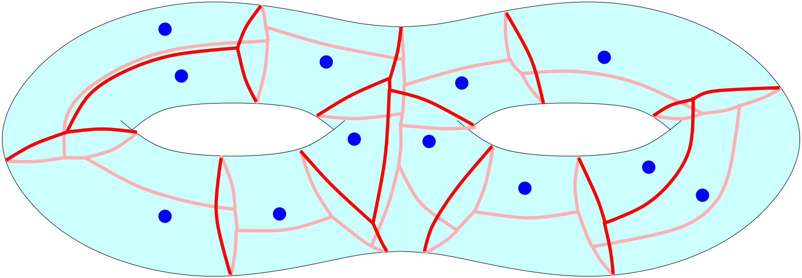
Algorithm sketch



Algorithm sketch

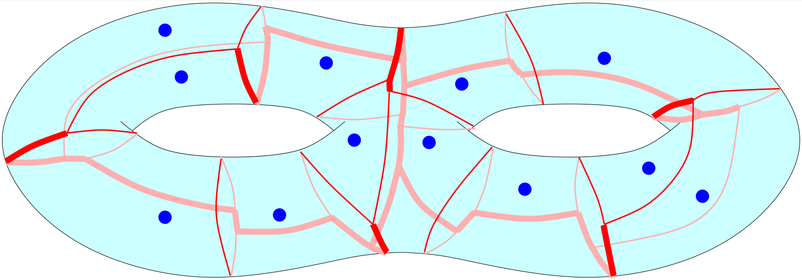


Algorithm sketch



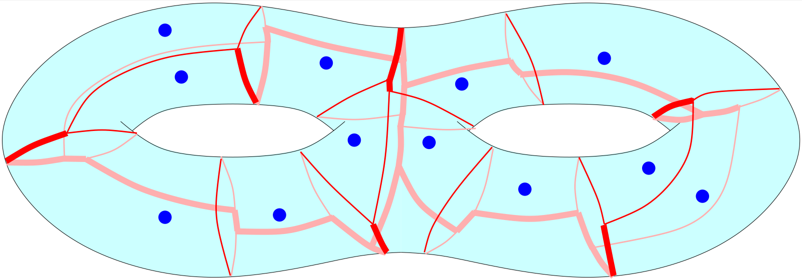
- compute a spanning tree T of C;

Algorithm sketch



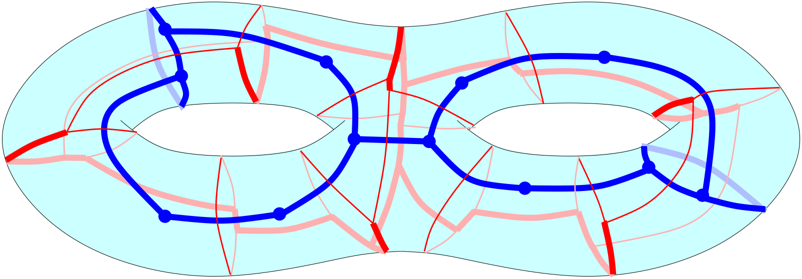
- compute a spanning tree T of C;

Algorithm sketch



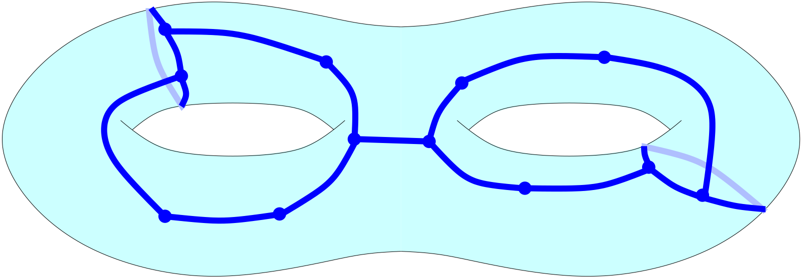
- compute a spanning tree T of C;
- return $K := (E(C) - T)^{\perp}$ ("Delaunay" edges of complement).

Algorithm sketch



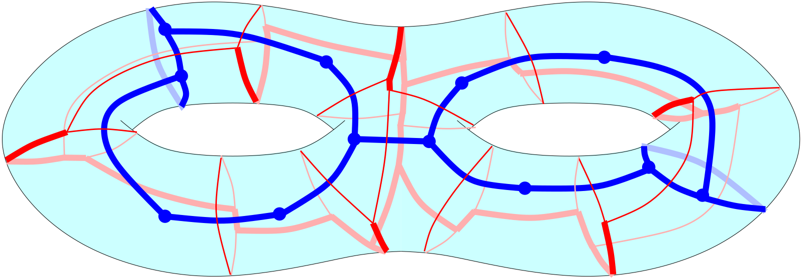
- compute a spanning tree T of C;
- return $K := (E(C) - T)^{\perp}$ ("Delaunay" edges of complement).

Algorithm sketch



- compute a spanning tree T of C;
- return $K := (E(C) - T)^{\perp}$ ("Delaunay" edges of complement).

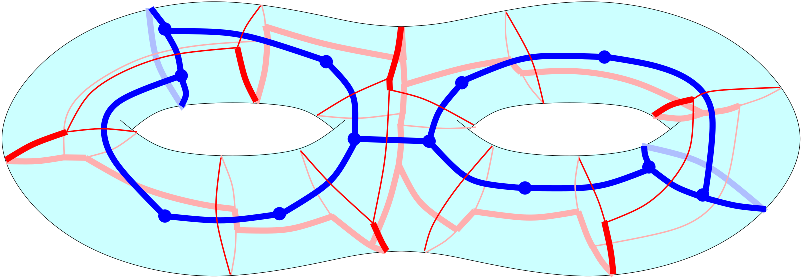
Algorithm sketch



- compute a spanning tree T of C;
- return $K := (E(C) - T)^{\perp}$ ("Delaunay" edges of complement).

If $w(e) :=$ length of $e^{\perp}$, and T is a maximum spanning tree w.r.t. w, then K is a shortest cut graph with vertex set P.

Algorithm sketch

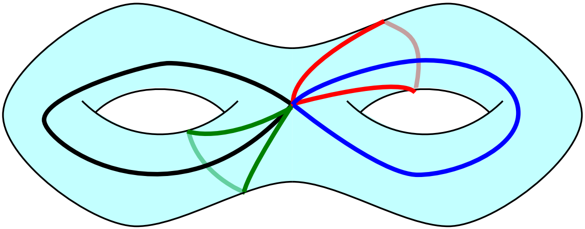


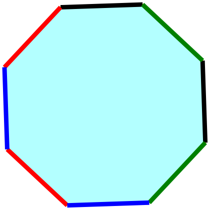
- compute a spanning tree T of C;
- return $K := (E(C) - T)^{\perp}$ ("Delaunay" edges of complement).

If $w(e) :=$ length of $e^{\perp}$, and T is a maximum spanning tree w.r.t. w, then K is a shortest cut graph with vertex set P.

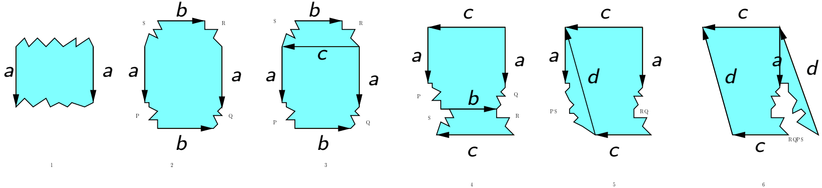
It suffices to prove that each edge of the shortest cut graph is of the form $e^{\perp}$. Proof idea: shortest cut graph = shortest basis of 1-dimensional homology of $\mathscr{S}$ relatively to P.

Canonical system of loops

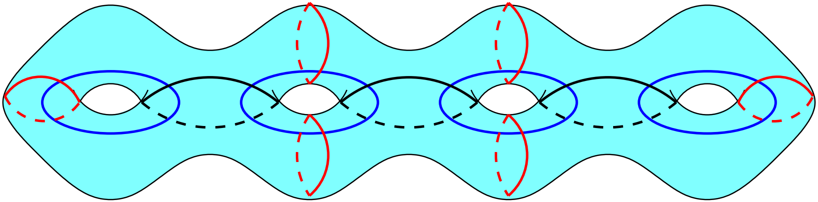




- Shortest: open!
- Some canonical system of loops with $O(gn)$ complexity: doable in $O(gn)$ time [Lazarus, Pocchiola, Vegter, Verroust, 2001].



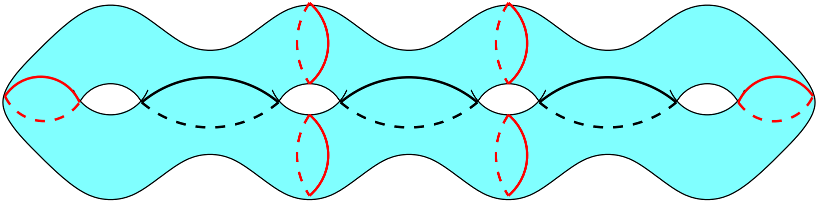
Octagonal decomposition



- Shortest: open!
- Some octagonal decomposition
  - with $O(gn)$ complexity
  - such that each closed curve is as short as possible in its homotopy class

doable in $O(gn \log n)$ time [CdV, Erickson, 2010].

Octagonal decomposition



- Shortest: open!
- Some octagonal decomposition
    - with $O(gn)$ complexity
    - such that each closed curve is as short as possible in its homotopy class

  doable in $O(gn \log n)$ time [CdV, Erickson, 2010].

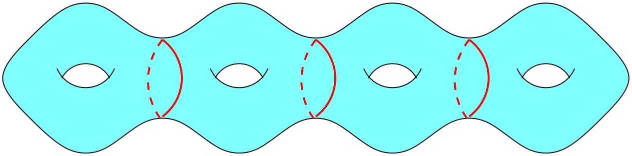
Contains a pants decomposition.

5. Other problems solved

Other curves

- shortest splitting closed curve (separating but non-contractible)

  → *crosses each shortest path $O(g)$ times; each loop of the shortest system of loops is the concatenation of two shortest paths.*



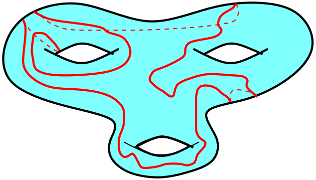
Other curves

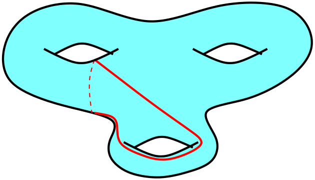
- shortest splitting closed curve (separating but non-contractible)

  → *crosses each shortest path $O(g)$ times; each loop of the shortest system of loops is the concatenation of two shortest paths.*

- shortest path homotopic to a given path;
  shortest closed curve freely homotopic to a given closed curve

  → *octagonal decomposition lifts, in the universal cover, to a regular tiling; defines a region of the universal cover to be explored.*





More generally. . .

These building blocks apply to seemingly unrelated problems:

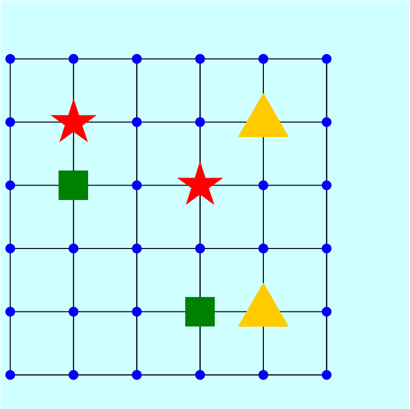
- topological graph theory: crossing number of graphs [Kawarabayashi, Reed, 2007];
- algorithms for planar graphs: maximum flow [Erickson, 2010], shortest non-crossing paths [Erickson, Nayyeri, 2009], multicut [CdV, 2015], [Cohen-Addad, CdV, de Mesmay, 2018?].
- algorithms for surface-embedded graphs: minimum cut [Chambers, Erickson, Nayyeri, 2009], maximum flow [Chambers, Erickson, Nayyeri, 2009].

various models:

- the plane with polygonal obstacles;
- polyhedral surfaces;
- disjoint curves in graphs;
- normal curves.

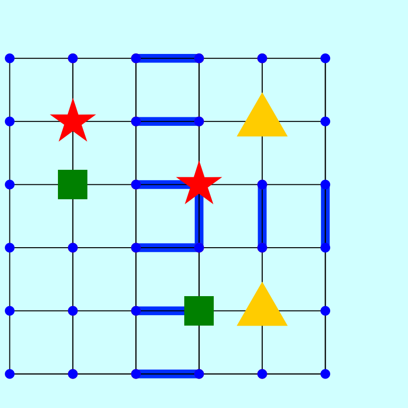
Example: multicut problem

- Input: $G = (V, E)$: a graph; pairs of vertices, called terminals.
- Output: $E' \subseteq E$ of minimum weight such that: after removing E', there is no path in G connecting the two vertices of a pair.



Example: multicut problem

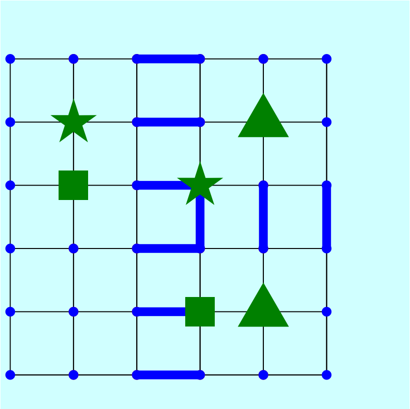
- Input: $G = (V, E)$: a graph; pairs of vertices, called terminals.
- Output: $E' \subseteq E$ of minimum weight such that: after removing E', there is no path in G connecting the two vertices of a pair.



Example: multicut problem

- Input: $G = (V, E)$: a graph; pairs of vertices, called terminals.
- Output: $E' \subseteq E$ of minimum weight such that: after removing E', there is no path in G connecting the two vertices of a pair.

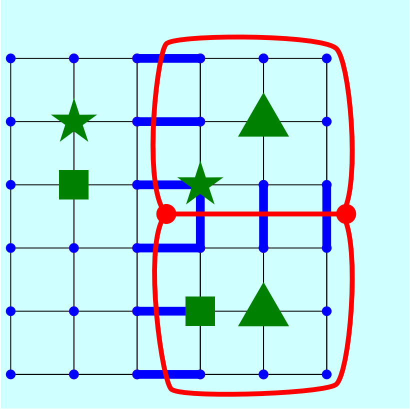
Reformulation: compute a "shortest" (in the cross-metric sense) graph in general position w.r.t. G and separating each pair of terminals.



Example: multicut problem

- Input: $G = (V, E)$: a graph; pairs of vertices, called terminals.
- Output: $E' \subseteq E$ of minimum weight such that: after removing E', there is no path in G connecting the two vertices of a pair.

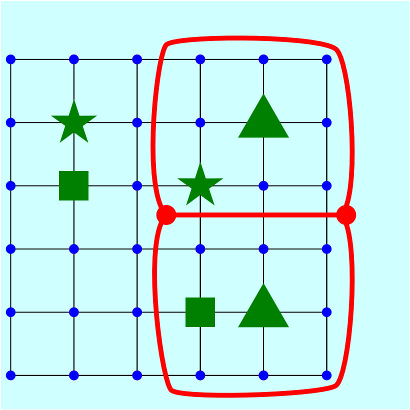
Reformulation: compute a "shortest" (in the cross-metric sense) graph in general position w.r.t. G and separating each pair of terminals.



Example: multicut problem

- Input: $G = (V, E)$: a graph; pairs of vertices, called terminals.
- Output: $E' \subseteq E$ of minimum weight such that: after removing E', there is no path in G connecting the two vertices of a pair.

Reformulation: compute a "shortest" (in the cross-metric sense) graph in general position w.r.t. G and separating each pair of terminals.



6. Open problems

Open problems

- Shortest decompositions (pants decomposition / octagonal decomposition / canonical system of loops);
- shortest graph embedding (possibly fixing vertices / homotopy / isotopy / combinatorial map);
- a conjecture by Negami: Given two graphs G and H embeddable on a fixed surface, can we embed them so that they cross at most $c \cdot |E(G)| \cdot |E(H)|$ times (for some absolute constant c)?

Thanks for your attention!

Questions?

1. 1. Topological graphs on surfaces in general
2. 2. Decision problems: homotopy and isotopy
3. 3. Shortest non-contractible closed curves
4. 4. Topological decompositions of surfaces
5. 5. Other problems solved
6. 6. Open problems