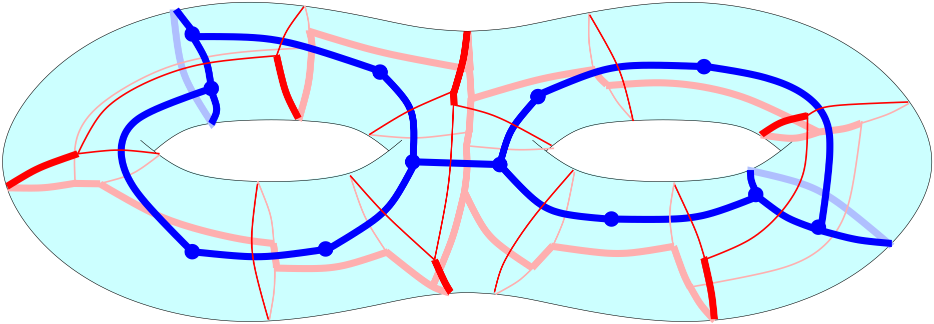
Shortest Cut Graph of a Surface with Prescribed Vertex Set

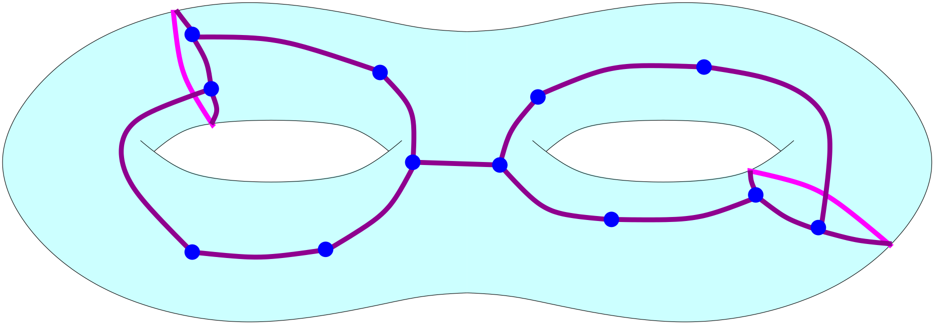
Éric Colin de Verdière
École normale supérieure, CNRS
Paris



The Problem

- Σ: compact surface without boundary;
- cut graph of Σ: graph embedded on Σ that cuts Σ into a topological disk.

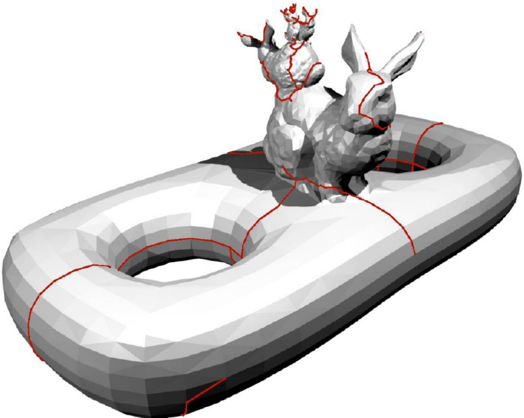
Goal: Given a finite point set P on Σ, compute the shortest cut graph with vertex set exactly P.



Why Cut a Surface into a Disk?

Applications

- Cutting a surface (into one or several planar pieces), to parameterize, put a texture, remesh, or compress it;
- sometimes, one needs to cut along lines of high curvature. The "length" function can be chosen arbitrarily, e.g., in relation to curvature.



[Favreau, Ph.D. thesis, 2009]

Why Cut a Surface into a Disk?

Applications

- Cutting a surface (into one or several planar pieces), to parameterize, put a texture, remesh, or compress it;
- sometimes, one needs to cut along lines of high curvature. The "length" function can be chosen arbitrarily, e.g., in relation to curvature.

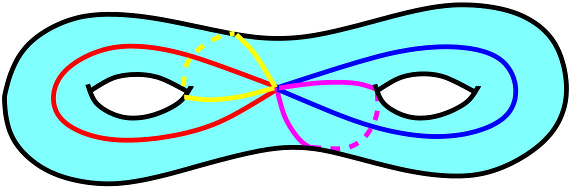
Theory

- Algorithms for nearly-planar graphs: separators, tree decompositions;
- building block for many algorithms to compute shortest graphs on surfaces:
  - shortest splitting cycle [Chambers et al., 2006],
  - shortest curves within a given homotopy class [CdV and Erickson, 2006],
  - minimum cut [Chambers, Erickson, Nayyeri, 2009],
  - shortest non-crossing walks [Erickson and Nayyeri, 2011],
  - edgewidth and facewidth parameters [Cabello, CdV, Lazarus, 2010].

Comparison with Previous Work

Previous work

- [Erickson and Har-Peled, 2004]:
  - computing the shortest cut graph (without fixing vertices) is NP-hard;
  - $O(\log^2 g)$-approximation in small polynomial time.
- [Erickson and Whittlesey, 2006]: fast algorithm to compute the shortest cut graph with one single (given) vertex.
- [CdV, 2003]: polynomial-time algorithm to compute the shortest graph isotopic (with fixed vertices) to a given graph.



This talk

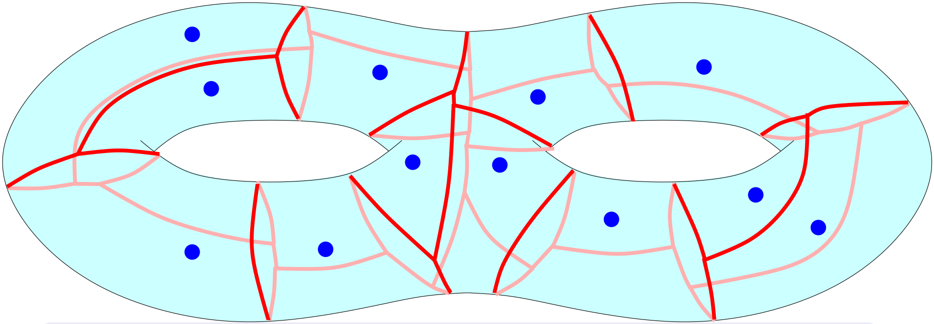
Generalization of the algorithm by [Erickson and Whittlesey]

- arbitrary finite set of vertices,
- possibly non-orientable surfaces.

More natural proof

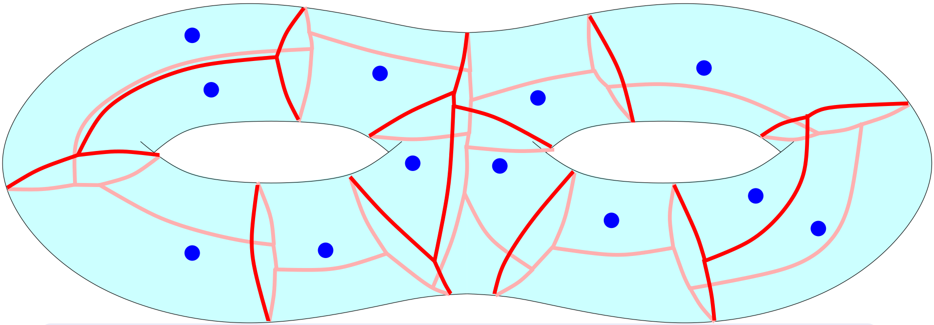
- The algorithm is greedy.
- (Most) optimal greedy algorithms fall within the matroid framework.
- I show that this is indeed the case here (via homology).
- Proof is therefore more natural and simpler.

Algorithm Description



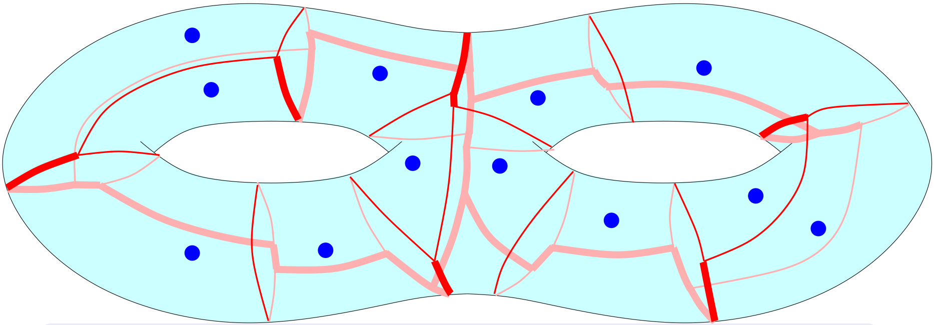
- Simultaneously grow a disk around each point in P. Compute the "Voronoi diagram" $\mathbb{V}$ of P, i.e., the set of points where these disks collide.
- A P-path is a path whose endpoints are in P.
- For each edge e of $\mathbb{V}$, let $\delta(e)$ be the dual "Delaunay edge": the shortest P-path crossing only edge e. Let the weight of e be the length of $\delta(e)$.
- Compute a *maximum* spanning tree T of $\mathbb{V}$ w.r.t. these weights.
- Return $\{\delta(e) \mid e \in \mathbb{V} \setminus T\}$.

Algorithm Description



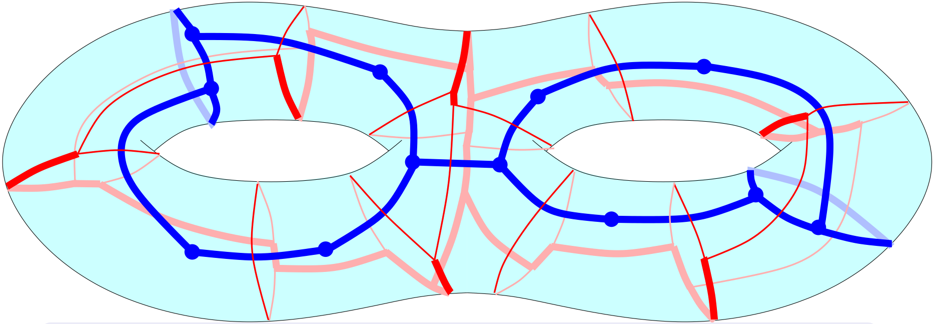
- Simultaneously grow a disk around each point in P. Compute the "Voronoi diagram" $\mathbb{V}$ of P, i.e., the set of points where these disks collide.
- A P-path is a path whose endpoints are in P.
- For each edge e of $\mathbb{V}$, let $\delta(e)$ be the dual "Delaunay edge": the shortest P-path crossing only edge e. Let the weight of e be the length of $\delta(e)$.
- Compute a *maximum* spanning tree T of $\mathbb{V}$ w.r.t. these weights.
- Return $\{\delta(e) \mid e \in \mathbb{V} \setminus T\}$.

Algorithm Description



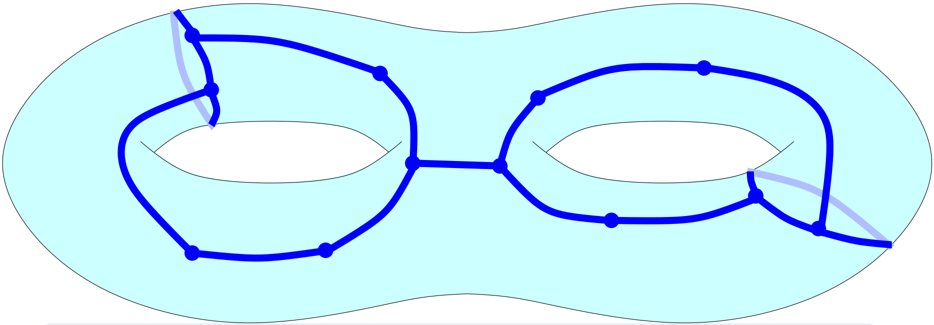
- Simultaneously grow a disk around each point in P. Compute the "Voronoi diagram" $\mathbb{V}$ of P, i.e., the set of points where these disks collide.
- A P-path is a path whose endpoints are in P.
- For each edge e of $\mathbb{V}$, let $\delta(e)$ be the dual "Delaunay edge": the shortest P-path crossing only edge e. Let the weight of e be the length of $\delta(e)$.
- Compute a *maximum* spanning tree T of $\mathbb{V}$ w.r.t. these weights.
- Return $\{\delta(e) \mid e \in \mathbb{V} \setminus T\}$.

Algorithm Description



- Simultaneously grow a disk around each point in P. Compute the "Voronoi diagram" $\mathbb{V}$ of P, i.e., the set of points where these disks collide.
- A P-path is a path whose endpoints are in P.
- For each edge e of $\mathbb{V}$, let $\delta(e)$ be the dual "Delaunay edge": the shortest P-path crossing only edge e. Let the weight of e be the length of $\delta(e)$.
- Compute a *max*imum spanning tree T of $\mathbb{V}$ w.r.t. these weights.
- Return $\{\delta(e) \mid e \in \mathbb{V} \setminus T\}$.

Algorithm Description



- Simultaneously grow a disk around each point in P. Compute the "Voronoi diagram" $\mathbb{V}$ of P, i.e., the set of points where these disks collide.
- A P-path is a path whose endpoints are in P.
- For each edge e of $\mathbb{V}$, let $\delta(e)$ be the dual "Delaunay edge": the shortest P-path crossing only edge e. Let the weight of e be the length of $\delta(e)$.
- Compute a *max*imum spanning tree T of $\mathbb{V}$ w.r.t. these weights.
- Return $\{\delta(e) \mid e \in \mathbb{V} \setminus T\}$.

Trivial Example: The Case of the Sphere

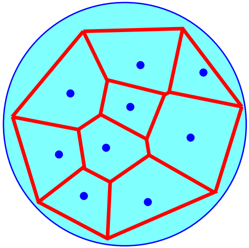
Problem reformulation

- Σ = Euclidean plane $\mathbb{R}^2$ with point at ∞ (sphere);
- We actually want to compute the minimum spanning tree of P.

Well-known: $\mathrm{MST}(P) \subseteq \mathrm{Gab}(P) \subseteq \mathrm{Del}(P)$.

What does the algorithm?

- computes a *max*imum spanning tree of the Voronoi diagram, where the weight of a Voronoi edge is the length of its dual "Delaunay" edge;
- the dual of the complement is the *min*imum spanning tree of the Delaunay triangulation, i.e., the *min*imum spanning tree of P.



Trivial Example: The Case of the Sphere

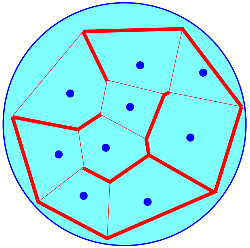
Problem reformulation

- $\Sigma =$ Euclidean plane $\mathbb{R}^2$ with point at ∞ (sphere);
- We actually want to compute the minimum spanning tree of P.

Well-known: $\mathrm{MST}(P) \subseteq \mathrm{Gab}(P) \subseteq \mathrm{Del}(P)$.

What does the algorithm?

- computes a *max*imum spanning tree of the Voronoi diagram, where the weight of a Voronoi edge is the length of its dual "Delaunay" edge;
- the dual of the complement is the *min*imum spanning tree of the Delaunay triangulation, i.e., the *min*imum spanning tree of P.



Trivial Example: The Case of the Sphere

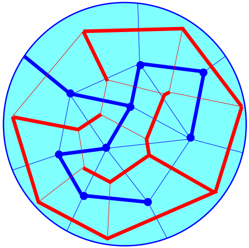
Problem reformulation

- Σ = Euclidean plane $\mathbb{R}^2$ with point at ∞ (sphere);
- We actually want to compute the minimum spanning tree of P.

Well-known: $\mathrm{MST}(P) \subseteq \mathrm{Gab}(P) \subseteq \mathrm{Del}(P)$.

What does the algorithm?

- computes a *max*imum spanning tree of the Voronoi diagram, where the weight of a Voronoi edge is the length of its dual "Delaunay" edge;
- the dual of the complement is the *min*imum spanning tree of the Delaunay triangulation, i.e., the *min*imum spanning tree of P.



Trivial Example: The Case of the Sphere

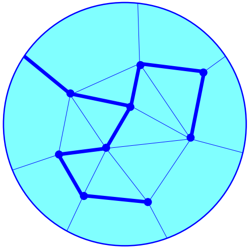
Problem reformulation

- Σ = Euclidean plane $\mathbb{R}^2$ with point at ∞ (sphere);
- We actually want to compute the minimum spanning tree of P.

Well-known: $\mathrm{MST}(P) \subseteq \mathrm{Gab}(P) \subseteq \mathrm{Del}(P)$.

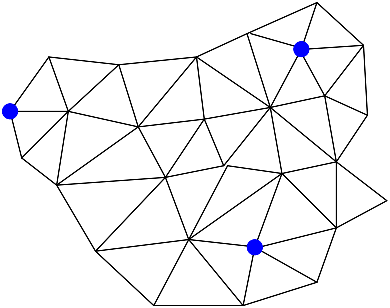
What does the algorithm?

- computes a *max*imum spanning tree of the Voronoi diagram, where the weight of a Voronoi edge is the length of its dual "Delaunay" edge;
- the dual of the complement is the *min*imum spanning tree of the Delaunay triangulation, i.e., the *min*imum spanning tree of P.



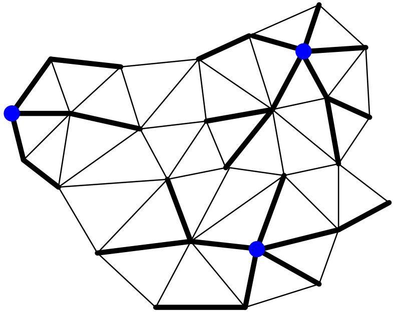
Metric: Combinatorial Surfaces

- A fixed weighted graph G on the surface gives the metric.
- The P-paths of the cut graph are restricted to lie in G and to be non-crossing. (All the $\delta(e)$ are non-crossing.)



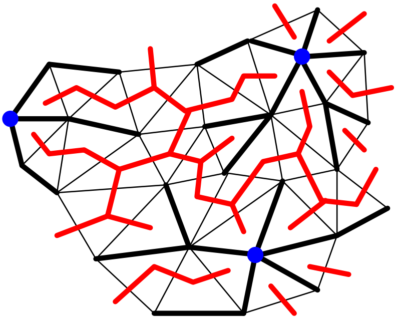
Metric: Combinatorial Surfaces

- A fixed weighted graph G on the surface gives the metric.
- The P-paths of the cut graph are restricted to lie in G and to be non-crossing. (All the $\delta(e)$ are non-crossing.)

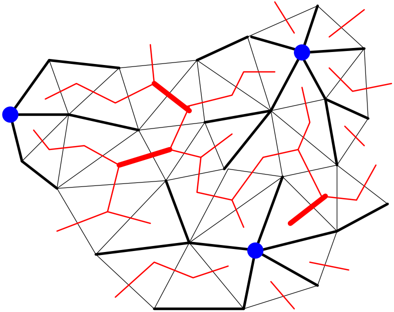


Metric: Combinatorial Surfaces

- A fixed weighted graph G on the surface gives the metric.
- The P-paths of the cut graph are restricted to lie in G and to be non-crossing. (All the $\delta(e)$ are non-crossing.)

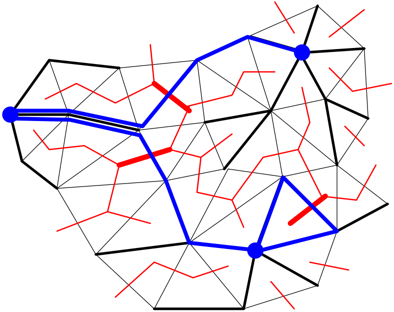


- A fixed weighted graph G on the surface gives the metric.
- The P-paths of the cut graph are restricted to lie in G and to be non-crossing. (All the $\delta(e)$ are non-crossing.)



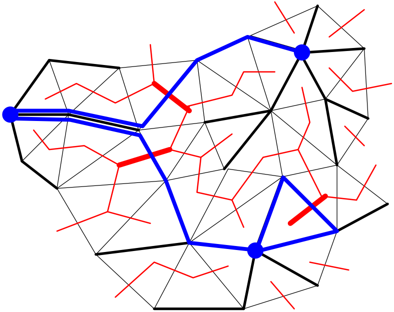
Metric: Combinatorial Surfaces

- A fixed weighted graph G on the surface gives the metric.
- The P-paths of the cut graph are restricted to lie in G and to be non-crossing. (All the $\delta(e)$ are non-crossing.)



Metric: Combinatorial Surfaces

- A fixed weighted graph G on the surface gives the metric.
- The P-paths of the cut graph are restricted to lie in G and to be non-crossing. (All the $\delta(e)$ are non-crossing.)



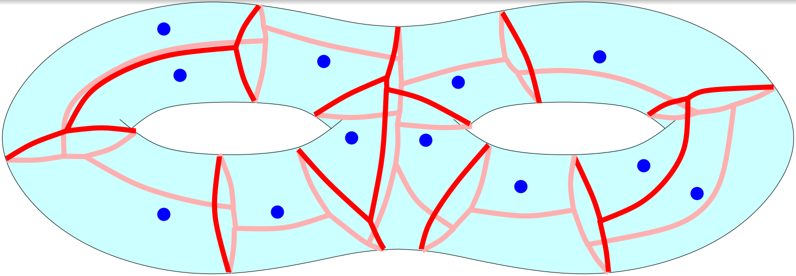
Running-time

$O(n \log n)$ plus size of the output: $O((g + |P|)n)$
where g is the genus of Σ and n is the complexity of G.

Greedy View of the Algorithm

Lemma

Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.

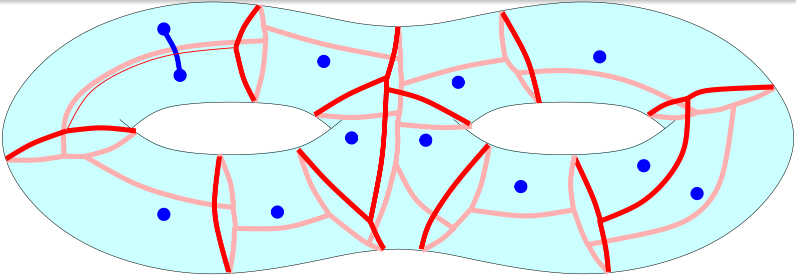


- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Greedy View of the Algorithm

Lemma

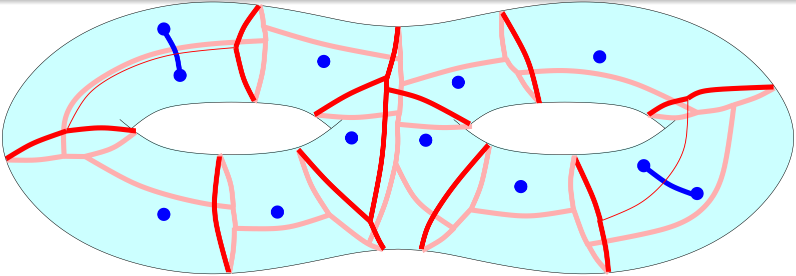
Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.



- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Lemma

Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.

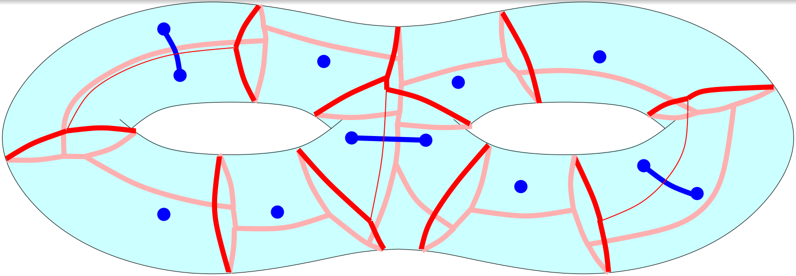


- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Greedy View of the Algorithm

Lemma

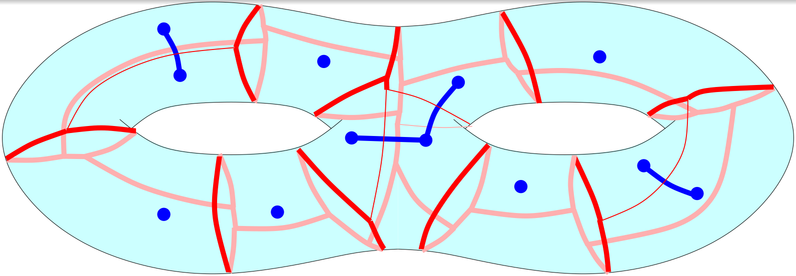
Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.



- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Lemma

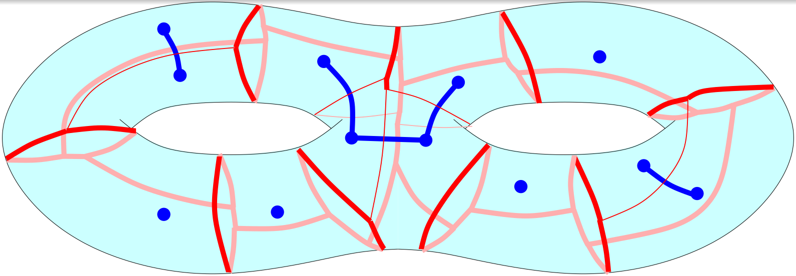
Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.



- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Lemma

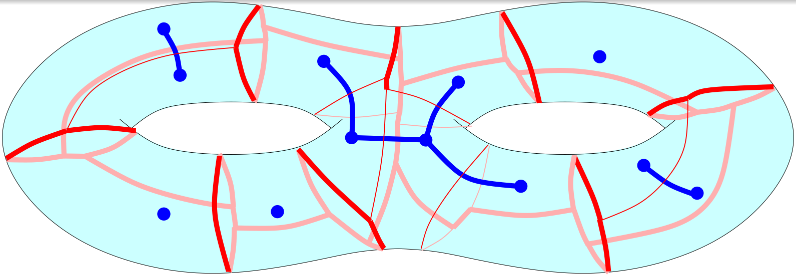
Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.



- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Lemma

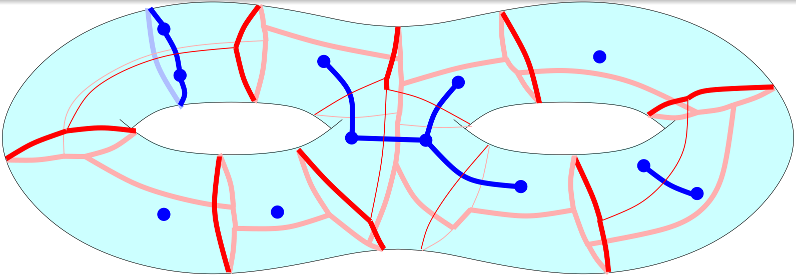
Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.



- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Lemma

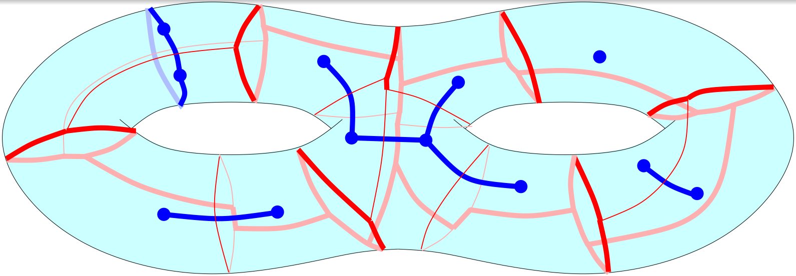
Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.



- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Lemma

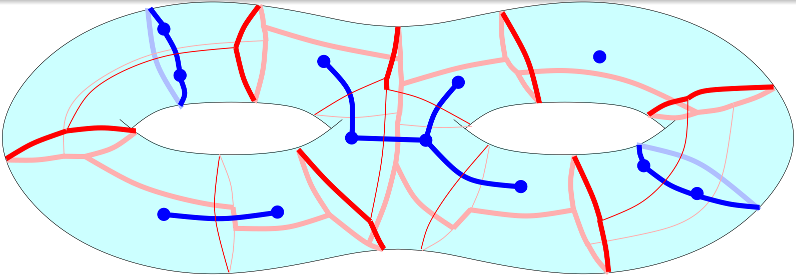
Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.



- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Lemma

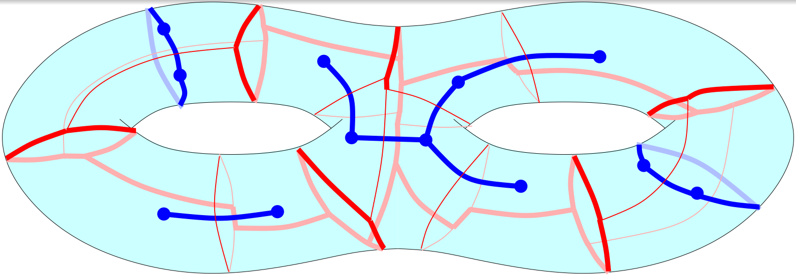
Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.



- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Lemma

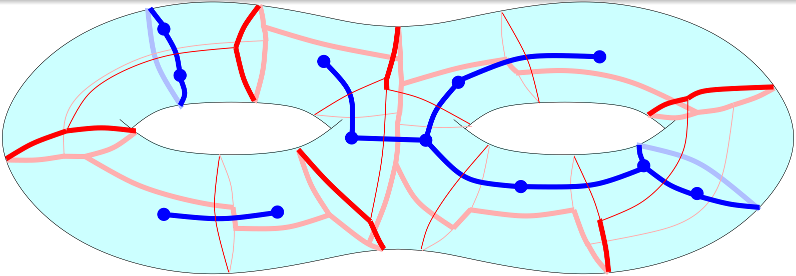
Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.



- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Lemma

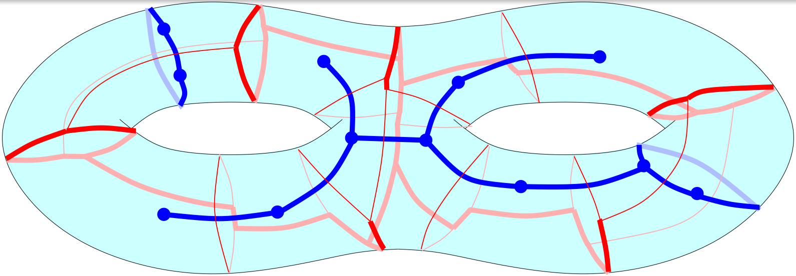
Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.



- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Lemma

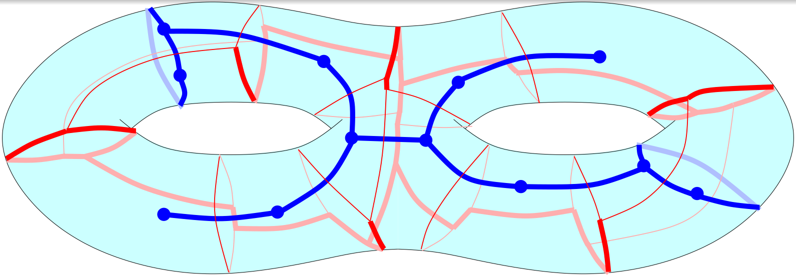
Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.



- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Lemma

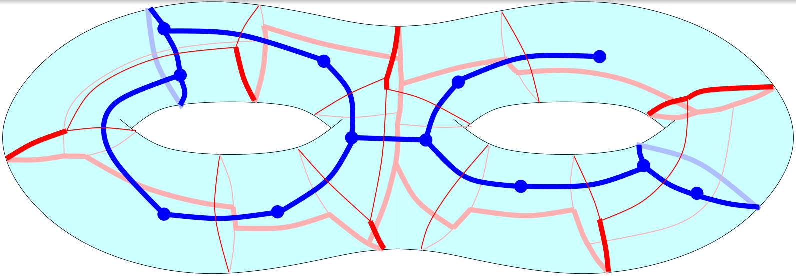
Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.



- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Lemma

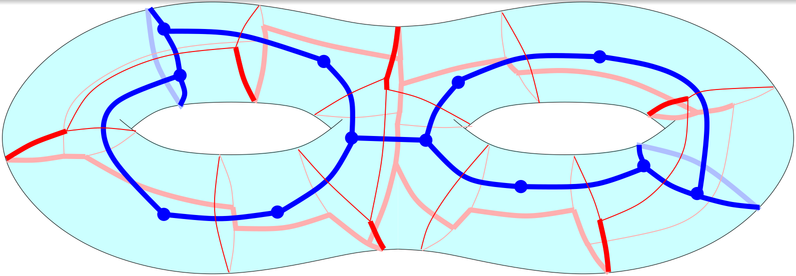
Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.



- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Lemma

Alternate way of computing the same result: iteratively add to a set S the shortest non-disconnecting Delaunay P-path p such that $S \cup \{p\}$ leaves Σ connected.



- To compute a maximum spanning tree of $\mathbb{V}$: iteratively remove minimum-weight non-disconnecting edges in $\mathbb{V}$.
- Each time we remove an edge e to $\mathbb{V}$, we add the dual Delaunay edge $\delta(e)$ to a set S.
- $\Sigma \setminus S$ is connected $\Leftrightarrow$ $\mathbb{V}$ is connected. $\square$

Minimum-weight basis of a vector space

Let V be a vector space; let $w : V \to \mathbb{Z}_+$ be a weight function.
The greedy algorithm computes a minimum-weight basis of V:

- start with $A = \emptyset$;
- whenever possible, iteratively add to A a minimum-weight element linearly independent with A.

Minimum-weight basis of a vector space

Let V be a vector space; let $w : V \to \mathbb{Z}_+$ be a weight function. The greedy algorithm computes a minimum-weight basis of V:

- start with $A = \emptyset$;
- whenever possible, iteratively add to A a minimum-weight element linearly independent with A.

Proof

- Greedy basis $G = \{g_1, \ldots, g_k\}$ in increasing order of weight,
- optimal basis $O = \{o_1, \ldots, o_k\}$ in increasing order of weight.

If $w(O) < w(G)$, then $w(o_i) < w(g_i)$ for some i:

$$\begin{array}{l} G = \{\ g_1 \le g_2 \le \ldots \le g_{i-1} \le g_i\ \ \le \ldots \le g_k\ \} \\ \hphantom{G = \{\ g_1 \le g_2 \le \ldots \le g_{i-1} \le\ } \vee \\ O = \{\ o_1 \le o_2 \le \ldots \le o_{i-1} \le o_i\ \ \le \ldots \le o_k\ \}. \end{array}$$

At the ith step, one of $\{o_1, \ldots, o_i\}$ would be chosen instead of g_i, contradiction! $\square$

Preliminaries on Matroids and Greedy Algorithms

Minimum-weight basis of a vector space

Let V be a vector space; let $w : V \to \mathbb{Z}_+$ be a weight function. The greedy algorithm computes a minimum-weight basis of V:

- start with $A = \emptyset$;
- whenever possible, iteratively add to A a minimum-weight element linearly independent with A.

More generally...

A matroid is a pair (V, I) where V is a set and I is a non-empty set of finite subsets of V such that:

- if $A \subseteq B \in I$, then $A \in I$;
- if $A, B \in I$ and $|A| < |B|$, then $A \cup \{z\} \in I$ for some $z \in B \setminus A$.

Given $w : V \to \mathbb{Z}_+$, the greedy algorithm finds a minimum-weight, inclusionwise maximal element of I.

Strategy of the Proof: the Hidden Matroid

Formally: 1-dimensional singular homology on surfaces, over $\mathbb{Z}/2\mathbb{Z}$, relatively to P.

Rough sketch of proof (simplification)

1. We exhibit a matroid with base space the set of P-paths.
2. Let S be a set of non-crossing P-paths. Then S is an independent set iff it does not disconnect Σ.
   (In general, an independent set may contain crossing P-paths.)
3. Let S be an independent set, and p be the shortest P-path such that $S \cup \{p\}$ is independent. Then p is Delaunay.

This proves the result. . .

- The shortest maximal independent set can be computed greedily;
- the paths computed are Delaunay, and therefore non-crossing;
- thus the algorithm is identical to the previous one, and computes a shortest cut graph. $\square$

Strategy of the Proof: the Hidden Matroid

Formally: 1-dimensional singular homology on surfaces, over $\mathbb{Z}/2\mathbb{Z}$, relatively to P.

Rough sketch of proof (simplification)

1. We exhibit a matroid with base space the set of P-paths.
2. Let S be a set of non-crossing P-paths. Then S is an independent set iff it does not disconnect Σ.
   (In general, an independent set may contain crossing P-paths.)
3. Let S be an independent set, and p be the shortest P-path such that $S \cup \{p\}$ is independent. Then p is Delaunay.

This proves the result...

- The shortest maximal independent set can be computed greedily;
- the paths computed are Delaunay, and therefore non-crossing;
- thus the algorithm is identical to the previous one, and computes a shortest cut graph. $\square$

Strategy of the Proof: the Hidden Matroid

Formally: 1-dimensional singular homology on surfaces, over $\mathbb{Z}/2\mathbb{Z}$, relatively to P.

Rough sketch of proof (simplification)

1. We exhibit a matroid with base space the set of P-paths.
2. Let S be a set of non-crossing P-paths. Then S is an independent set iff it does not disconnect Σ.
   (In general, an independent set may contain crossing P-paths.)
3. Let S be an independent set, and p be the shortest P-path such that $S \cup \{p\}$ is independent. Then p is Delaunay.

This proves the result...

- The shortest maximal independent set can be computed greedily;
- the paths computed are Delaunay, and therefore non-crossing;
- thus the algorithm is identical to the previous one, and computes a shortest cut graph. $\square$

- Z: $\mathbb{Z}/2\mathbb{Z}$-vector space of all sets of P-paths.
- $B \subseteq Z$: $p_1 + \ldots + p_k \in B$ iff there exist P-paths $q_1, \ldots, q_\ell$ such that the concatenation of the paths

  $$\{p_1, \ldots, p_k, \quad q_1, q_1, \; q_2, q_2, \ldots, q_\ell, q_\ell\},$$

  in some order, possibly after replacing some paths by their reversals, is a contractible cycle.
  B is a vector space included in Z.
- $z_1, \ldots, z_m \in Z$ are dependent if some non-trivial linear combination (sum) among the z_i is in B.
- Z is a matroid!

- Z: $\mathbb{Z}/2\mathbb{Z}$-vector space of all sets of P-paths.
- $B \subseteq Z$: $p_1 + \ldots + p_k \in B$ iff there exist P-paths $q_1, \ldots, q_\ell$ such that the concatenation of the paths

  $$\{p_1, \ldots, p_k, \quad q_1, q_1, \;\; q_2, q_2, \ldots, q_\ell, q_\ell\},$$

  in some order, possibly after replacing some paths by their reversals, is a contractible cycle.
  B is a vector space included in Z.
- $z_1, \ldots, z_m \in Z$ are dependent if some non-trivial linear combination (sum) among the z_i is in B.
- Z is a matroid!

- Z: $\mathbb{Z}/2\mathbb{Z}$-vector space of all sets of P-paths.
- $B \subseteq Z$: $p_1 + \ldots + p_k \in B$ iff there exist P-paths $q_1, \ldots, q_\ell$ such that the concatenation of the paths
$$\{p_1, \ldots, p_k, \quad q_1, q_1, \; q_2, q_2, \ldots, q_\ell, q_\ell\},$$
in some order, possibly after replacing some paths by their reversals, is a contractible cycle.
B is a vector space included in Z.
- $z_1, \ldots, z_m \in Z$ are dependent if some non-trivial linear combination (sum) among the z_i is in B.
- Z is a matroid!

- Z: $\mathbb{Z}/2\mathbb{Z}$-vector space of all sets of P-paths.
- $B \subseteq Z$: $p_1 + \ldots + p_k \in B$ iff there exist P-paths $q_1, \ldots, q_\ell$ such that the concatenation of the paths

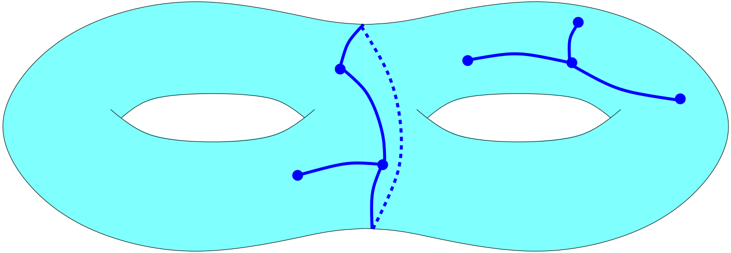
  $$\{p_1, \ldots, p_k, \quad q_1, q_1, \; q_2, q_2, \ldots, q_\ell, q_\ell\},$$

  in some order, possibly after replacing some paths by their reversals, is a contractible cycle.
  B is a vector space included in Z.
- $z_1, \ldots, z_m \in Z$ are dependent if some non-trivial linear combination (sum) among the z_i is in B.
- Z is a matroid!

Lemma

Let $\{p_1, \ldots, p_k\}$ be a set of non-crossing P-paths. Then $\{p_1, \ldots, p_k\}$ is an independent set iff it does not disconnect Σ.

Proof of $\Rightarrow$ by contraposition.



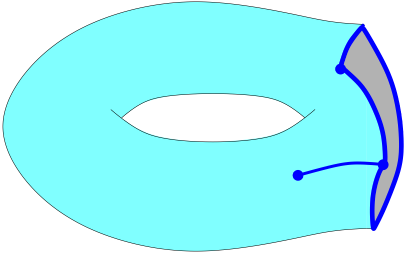
Recall:

- $\{z_1, \ldots, z_m\} \subseteq Z$ are dependent iff some non-trivial sum is in B.
- $p_1 + \ldots + p_k \in B$ iff there exist P-paths $q_1, \ldots, q_\ell$ such that the concatenation of the paths $\{p_1, \ldots, p_k, \quad q_1, q_1, \ q_2, q_2, \ldots, q_\ell, q_\ell\}$ in some order, possibly after replacing some paths by their reversals, is a contractible cycle.

Lemma

Let $\{p_1, \ldots, p_k\}$ be a set of non-crossing P-paths. Then $\{p_1, \ldots, p_k\}$ is an independent set iff it does not disconnect Σ.

Proof of $\Rightarrow$ by contraposition.



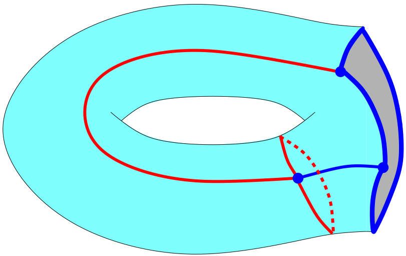
Recall:

- $\{z_1, \ldots, z_m\} \subseteq Z$ are dependent iff some non-trivial sum is in B.
- $p_1 + \ldots + p_k \in B$ iff there exist P-paths $q_1, \ldots, q_\ell$ such that the concatenation of the paths $\{p_1, \ldots, p_k, \quad q_1, q_1, \; q_2, q_2, \ldots, q_\ell, q_\ell\}$ in some order, possibly after replacing some paths by their reversals, is a contractible cycle.

Lemma

Let $\{p_1, \ldots, p_k\}$ be a set of non-crossing P-paths. Then $\{p_1, \ldots, p_k\}$ is an independent set iff it does not disconnect Σ.

Proof of $\Rightarrow$ by contraposition.



Recall:

- $\{z_1, \ldots, z_m\} \subseteq Z$ are dependent iff some non-trivial sum is in B.
- $p_1 + \ldots + p_k \in B$ iff there exist P-paths $q_1, \ldots, q_\ell$ such that the concatenation of the paths $\{p_1, \ldots, p_k, \quad q_1, q_1, \; q_2, q_2, \ldots, q_\ell, q_\ell\}$ in some order, possibly after replacing some paths by their reversals, is a contractible cycle.

Lemma

Let S be an independent set, and z be the shortest set of P-paths such that $S \cup \{z\}$ is independent. Then z is a single Delaunay path.

Proof: Linear algebra!

- *Recall:* T dependent $\Leftrightarrow$ some non-trivial linear combination of elements in T is in the vector space B.
- Thus $S \cup \{y\}$ is independent $\Leftrightarrow$ y is *not* a linear combination of elements in S and B.
- *Single path:* Assume $z = p_1 + \ldots + p_k$. Then for some i, $S \cup \{p_i\}$ is independent. Since p_i is no longer than z, we have $z = p_i$.
- *Single Delaunay path:* Similar, using the fact that $\delta(e_i)$ is the shortest path crossing e_i:
  Assume z crosses the edges $e_1, \ldots, e_k$ of $\mathbb{V}$. Then $z - (\delta(e_1) + \ldots + \delta(e_k))$ is in B. So for some i, $S \cup \{\delta(e_i)\}$ is independent. Since $\delta(e_i)$ is no longer than z, we have $z = \delta(e_i)$.

Lemma

Let S be an independent set, and z be the shortest set of P-paths such that $S \cup \{z\}$ is independent. Then z is a single Delaunay path.

Proof: Linear algebra!

- *Recall:* T dependent $\Leftrightarrow$ some non-trivial linear combination of elements in T is in the vector space B.
- Thus $S \cup \{y\}$ is independent $\Leftrightarrow$ y is *not* a linear combination of elements in S and B.
- *Single path:* Assume $z = p_1 + \ldots + p_k$. Then for some i, $S \cup \{p_i\}$ is independent. Since p_i is no longer than z, we have $z = p_i$.
- *Single Delaunay path:* Similar, using the fact that $\delta(e_i)$ is the shortest path crossing e_i:
  Assume z crosses the edges $e_1, \ldots, e_k$ of $\mathbb{V}$. Then $z - (\delta(e_1) + \ldots + \delta(e_k))$ is in B. So for some i, $S \cup \{\delta(e_i)\}$ is independent. Since $\delta(e_i)$ is no longer than z, we have $z = \delta(e_i)$.

Lemma

Let S be an independent set, and z be the shortest set of P-paths such that $S \cup \{z\}$ is independent. Then z is a single Delaunay path.

Proof: Linear algebra!

- *Recall:* T dependent $\Leftrightarrow$ some non-trivial linear combination of elements in T is in the vector space B.
- Thus $S \cup \{y\}$ is independent $\Leftrightarrow$ y is *not* a linear combination of elements in S and B.

- *Single path:* Assume $z = p_1 + \ldots + p_k$. Then for some i, $S \cup \{p_i\}$ is independent. Since p_i is no longer than z, we have $z = p_i$.
- *Single Delaunay path:* Similar, using the fact that $\delta(e_i)$ is the shortest path crossing e_i:
  Assume z crosses the edges $e_1, \ldots, e_k$ of $\mathbb{V}$. Then $z - (\delta(e_1) + \ldots + \delta(e_k))$ is in B. So for some i, $S \cup \{\delta(e_i)\}$ is independent. Since $\delta(e_i)$ is no longer than z, we have $z = \delta(e_i)$.

3 Delaunay Property

Lemma

Let S be an independent set, and z be the shortest set of P-paths such that $S \cup \{z\}$ is independent. Then z is a single Delaunay path.

Proof: Linear algebra!

- *Recall:* T dependent $\Leftrightarrow$ some non-trivial linear combination of elements in T is in the vector space B.
- Thus $S \cup \{y\}$ is independent $\Leftrightarrow$ y is *not* a linear combination of elements in S and B.

- *Single path:* Assume $z = p_1 + \ldots + p_k$. Then for some i, $S \cup \{p_i\}$ is independent. Since p_i is no longer than z, we have $z = p_i$.
- *Single Delaunay path:* Similar, using the fact that $\delta(e_i)$ is the shortest path crossing e_i:
  Assume z crosses the edges $e_1, \ldots, e_k$ of $\mathbb{V}$. Then $z - (\delta(e_1) + \ldots + \delta(e_k))$ is in B. So for some i, $S \cup \{\delta(e_i)\}$ is independent. Since $\delta(e_i)$ is no longer than z, we have $z = \delta(e_i)$.

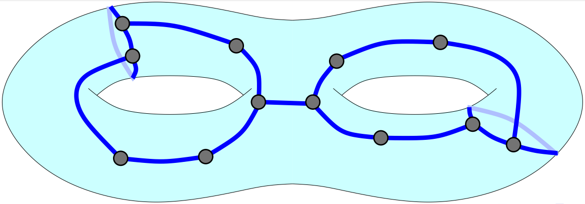
Remark

Every output P-path is:

- a shortest homotopic path, and
- the concatenation of two shortest paths plus an edge.

Extensions

- This algorithm also solves the following problem: Given a surface with boundary, compute the shortest set of simple disjoint arcs that cut the surface into a disk.
- Extension to PL surfaces (non-crossing paths) and smooth (Riemannian) surfaces.



Conclusion

Remark

Every output P-path is:

- a shortest homotopic path, and
- the concatenation of two shortest paths plus an edge.

Extensions

- This algorithm also solves the following problem: Given a surface with boundary, compute the shortest set of simple disjoint arcs that cut the surface into a disk.
- Extension to PL surfaces (non-crossing paths) and smooth (Riemannian) surfaces.

Naïve question (or open problem???)

How to compute the shortest (=minimum-weight) triangulation of a surface with given vertex set? This is NP-hard in the Euclidean plane [Mulzer, Rote, 2006]; how hard is it in the combinatorial surface model?