Bowyer-Watson algorithm, from the Euclidean plane to hyperbolic surfaces.

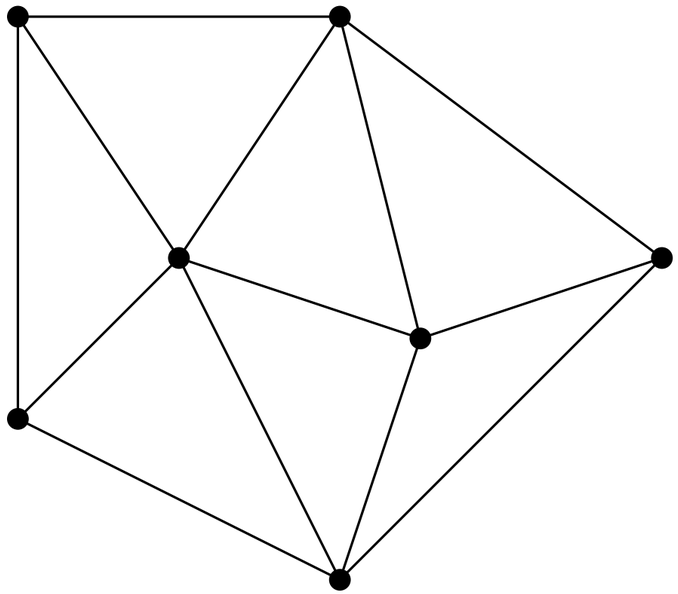
Vincent Despré, Dorian Perrot, Marc Pouget

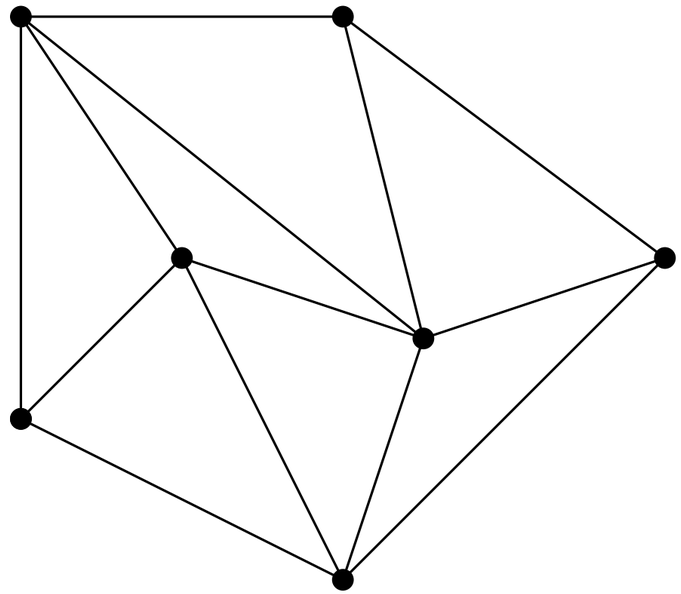
IRIMAS -Mulhouse- 04/2026

- Delaunay triangulation in Euclidean plane
  - BW-algorithm on a torus ($g = 1$)
  - Hyperbolic plane
  - BW-algorithm on the Bolza surface ($g = 2$)
  - BW-algorithm on hyperbolic surface ($g \geq 2$)

Delaunay triangulation

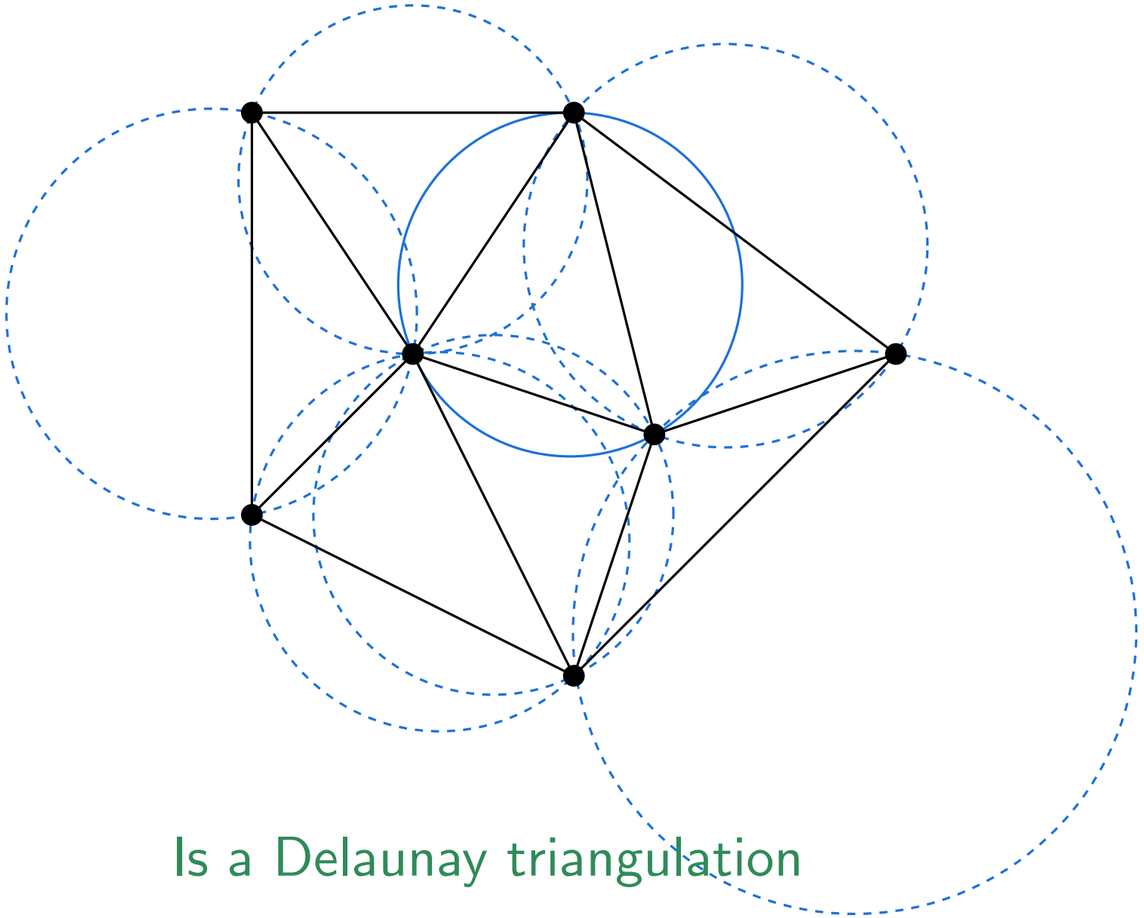
Def : a triangulation in which every circumscribed disk of triangle does not contain any of triangulation's vertices



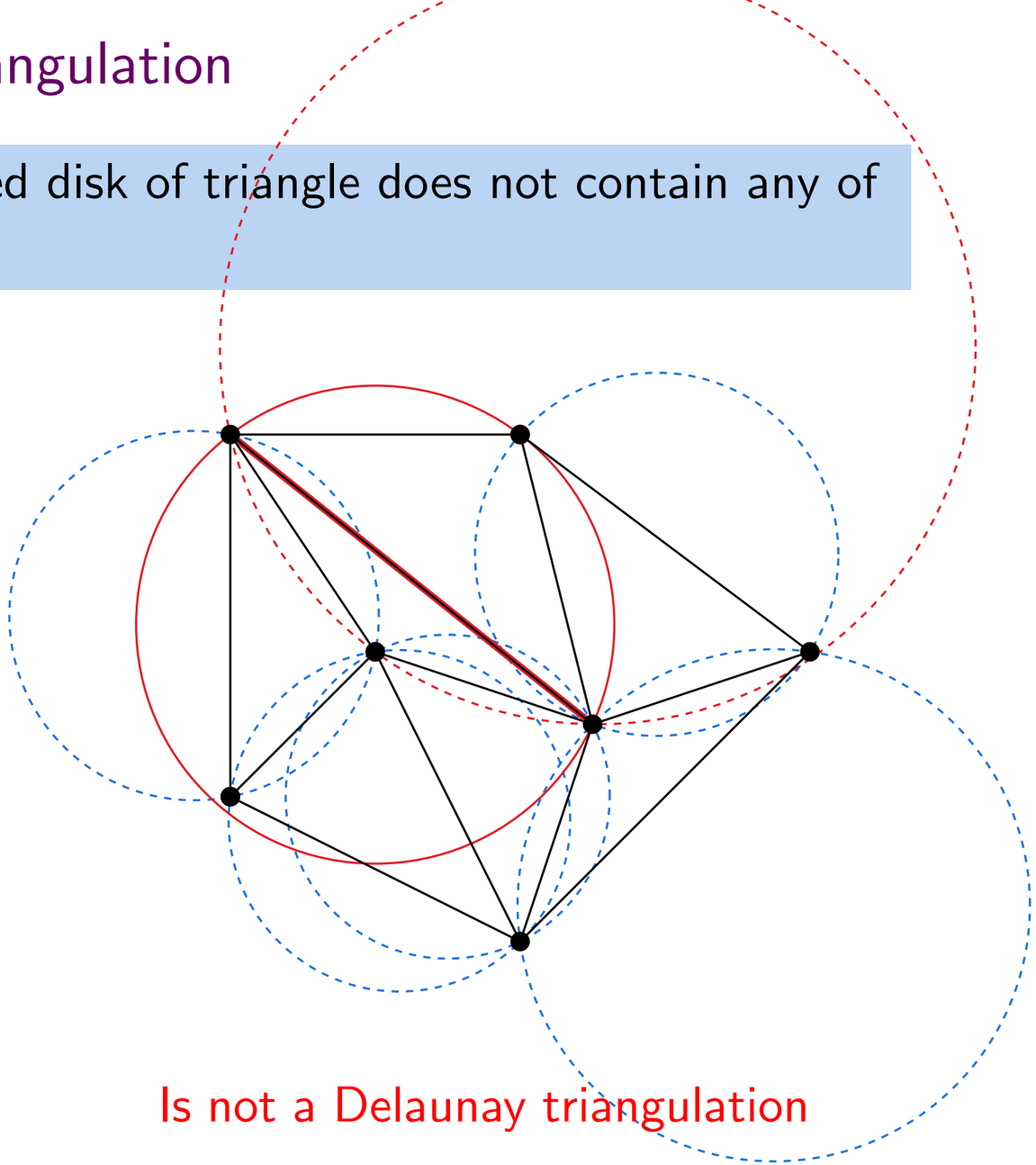


Delaunay triangulation

Def : a triangulation in which every circumscribed disk of triangle does not contain any of triangulation's vertices



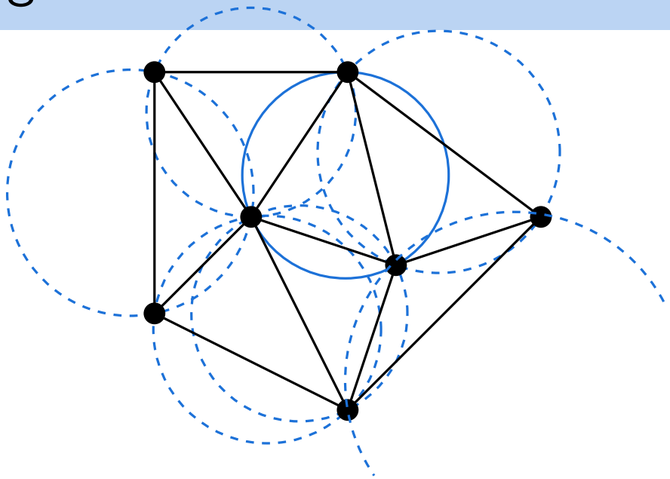
Is a Delaunay triangulation



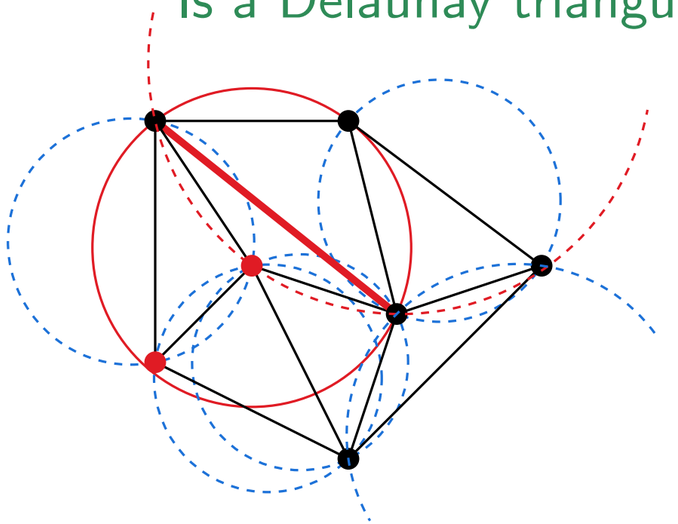
Is not a Delaunay triangulation

Delaunay triangulation

Def : a triangulation in which every circumscribed disk of triangle does not contain any of triangulation's vertices



Is a Delaunay triangulation



Is not a Delaunay triangulation

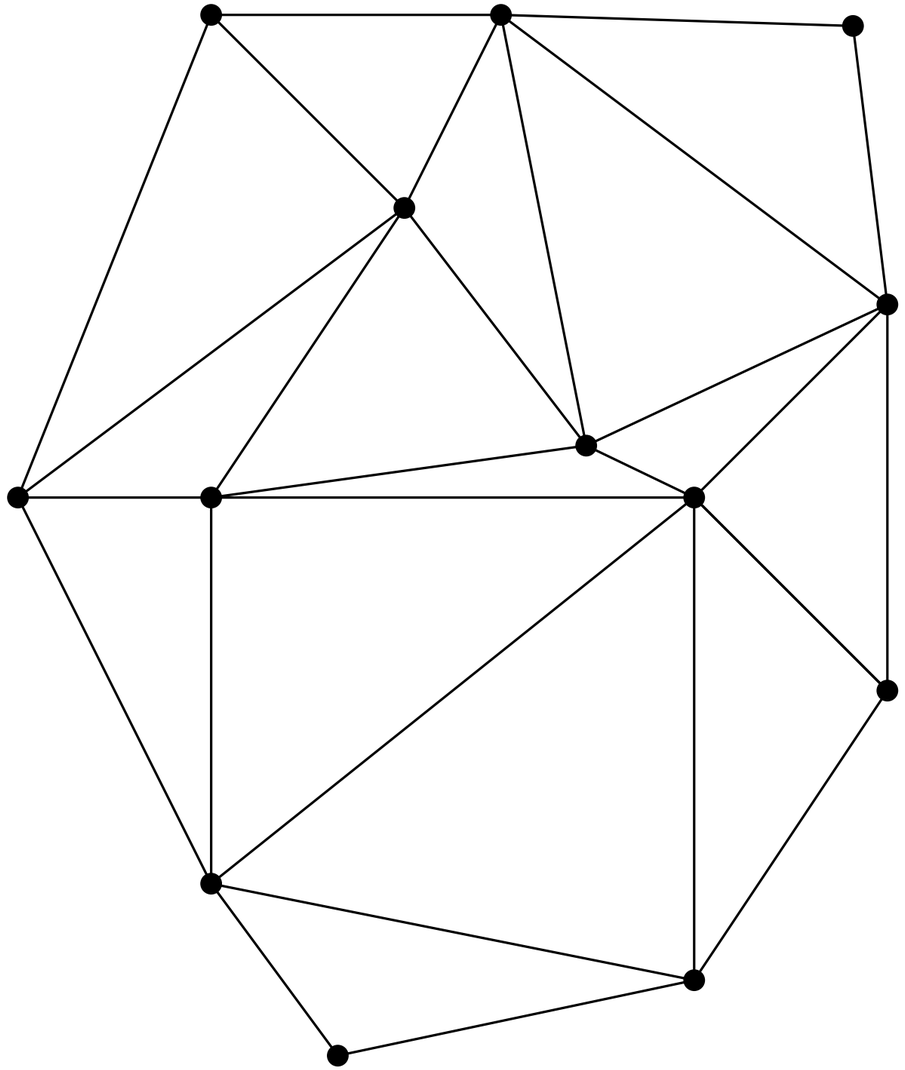
→ Flip algorithm in Euclidean plane [La71].
  → Generalised to a hyperbolic surface [DST24].

→ Divide and conquer [D87]

→ Bowyer-Watson algorithm. [Bo81].
  → Extended to Bolza surface (most symetric surface of genus 2) [IT17]
  ↓
  Generalization of this algorithm to all hyperbolic surfaces.[DDP26]

Flip algorithm

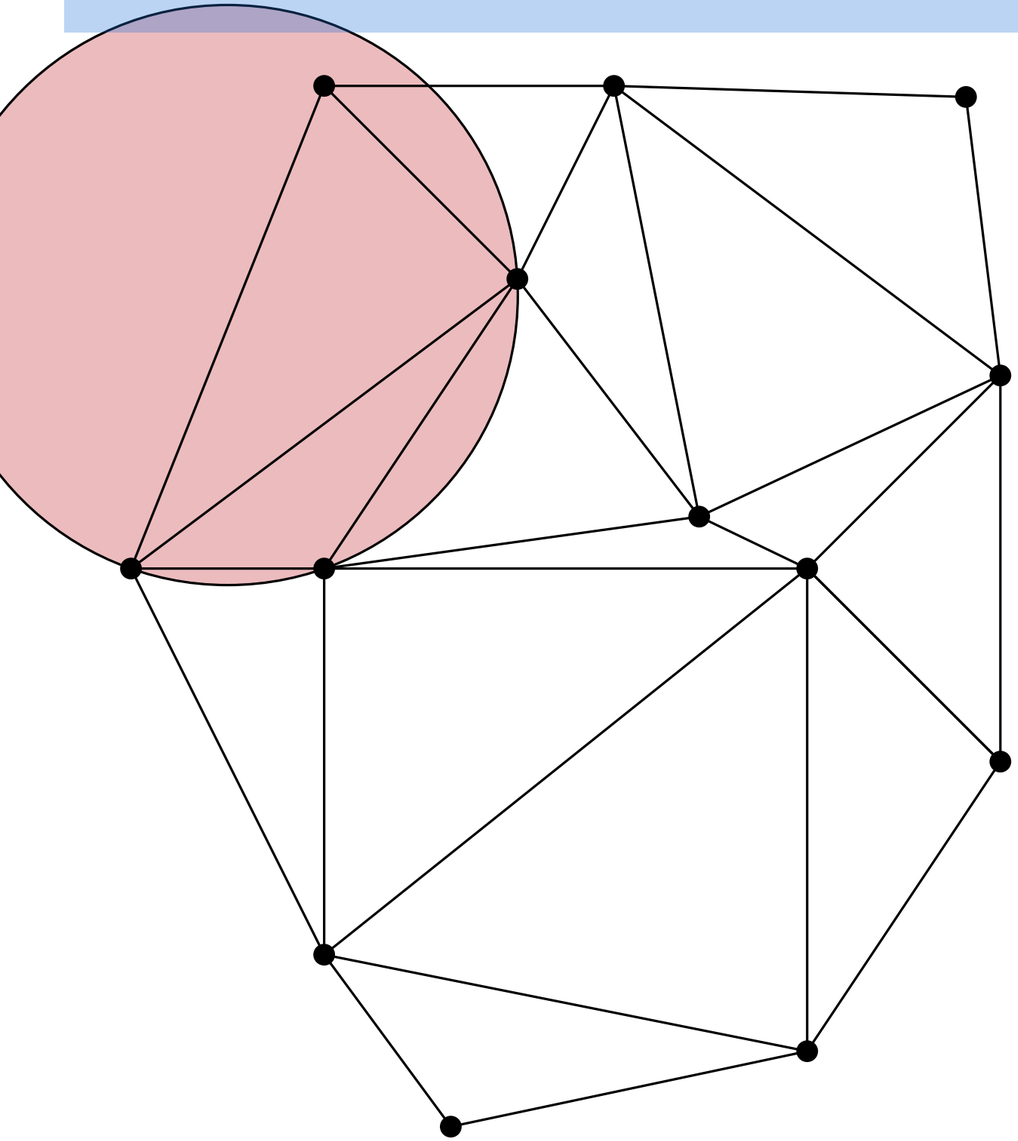
⟶ "improving triangulation" algorithm that computes a Delaunay triangulation



1) Start from a triangulation T

Flip algorithm

→ "improving triangulation" algorithm that computes a Delaunay triangulation



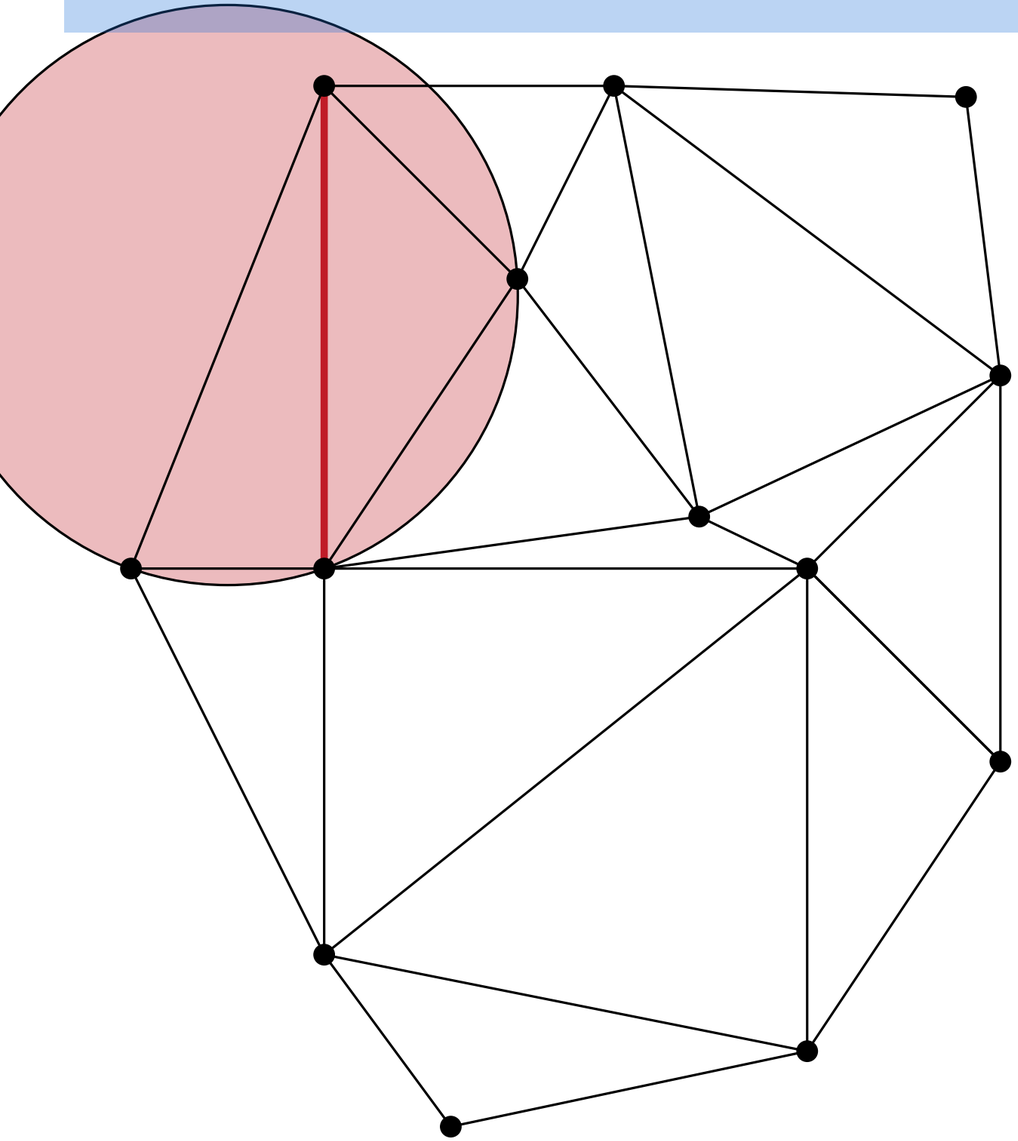
1) Start from a triangulation T

2) While T is not Delaunay :

Find a "bad" edge and flip it

Flip algorithm

→ "improving triangulation" algorithm that computes a Delaunay triangulation



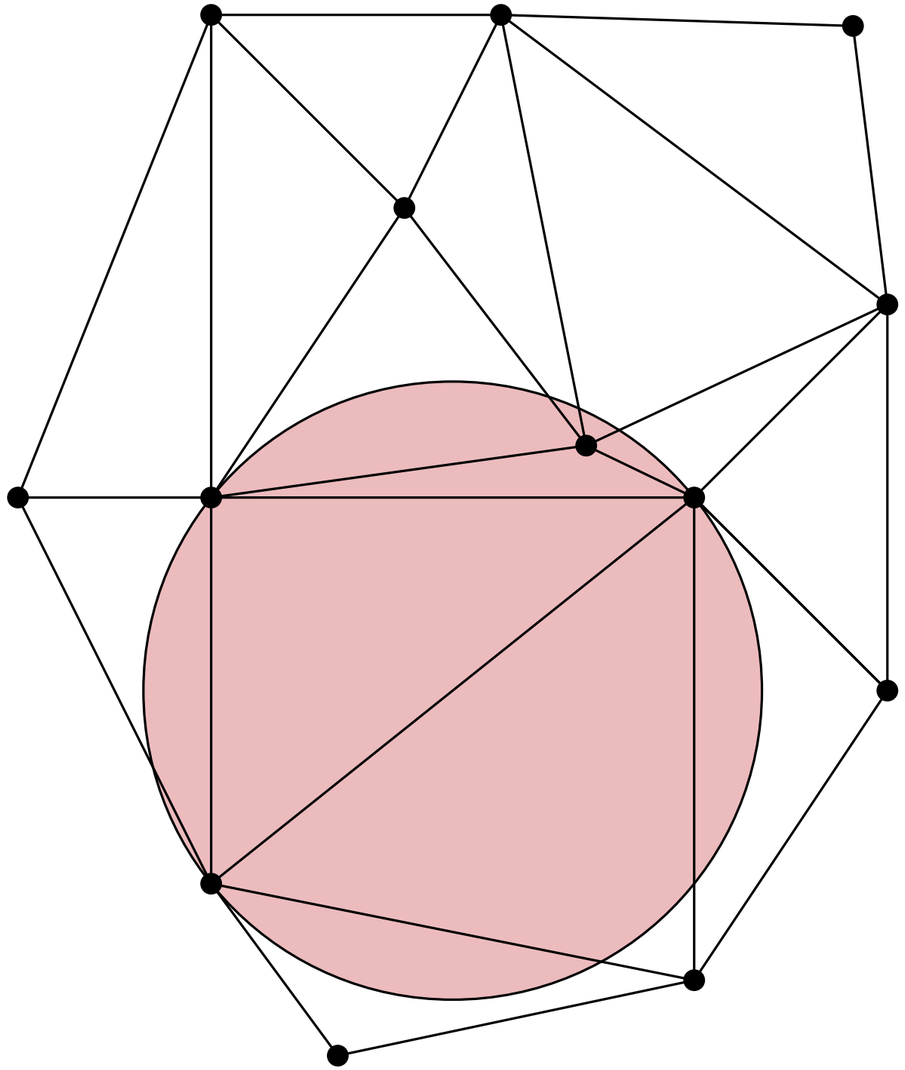
1) Start from a triangulation T

2) While T is not Delaunay :

Find a "bad" edge and flip it

Flip algorithm

→ "improving triangulation" algorithm that computes a Delaunay triangulation



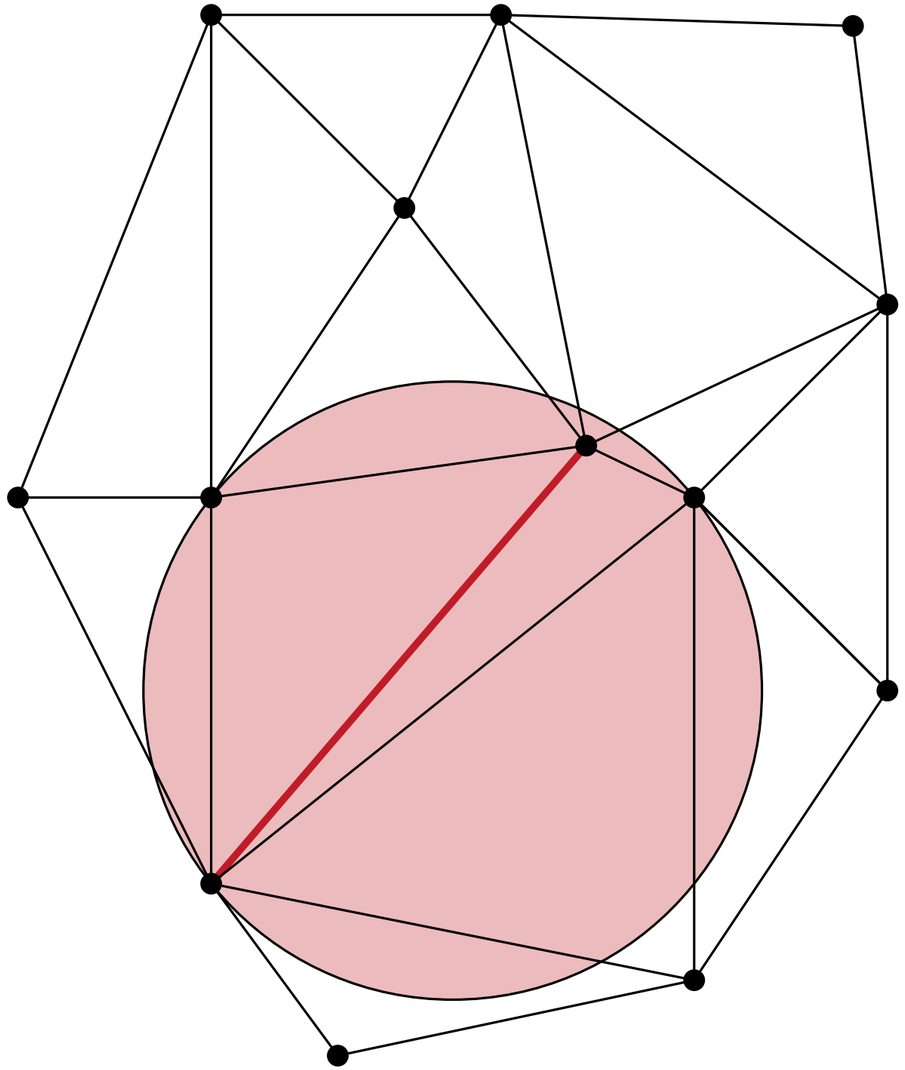
1) Start from a triangulation T

2) While T is not Delaunay :

Find a "bad" edge and flip it

Flip algorithm

→ "improving triangulation" algorithm that computes a Delaunay triangulation



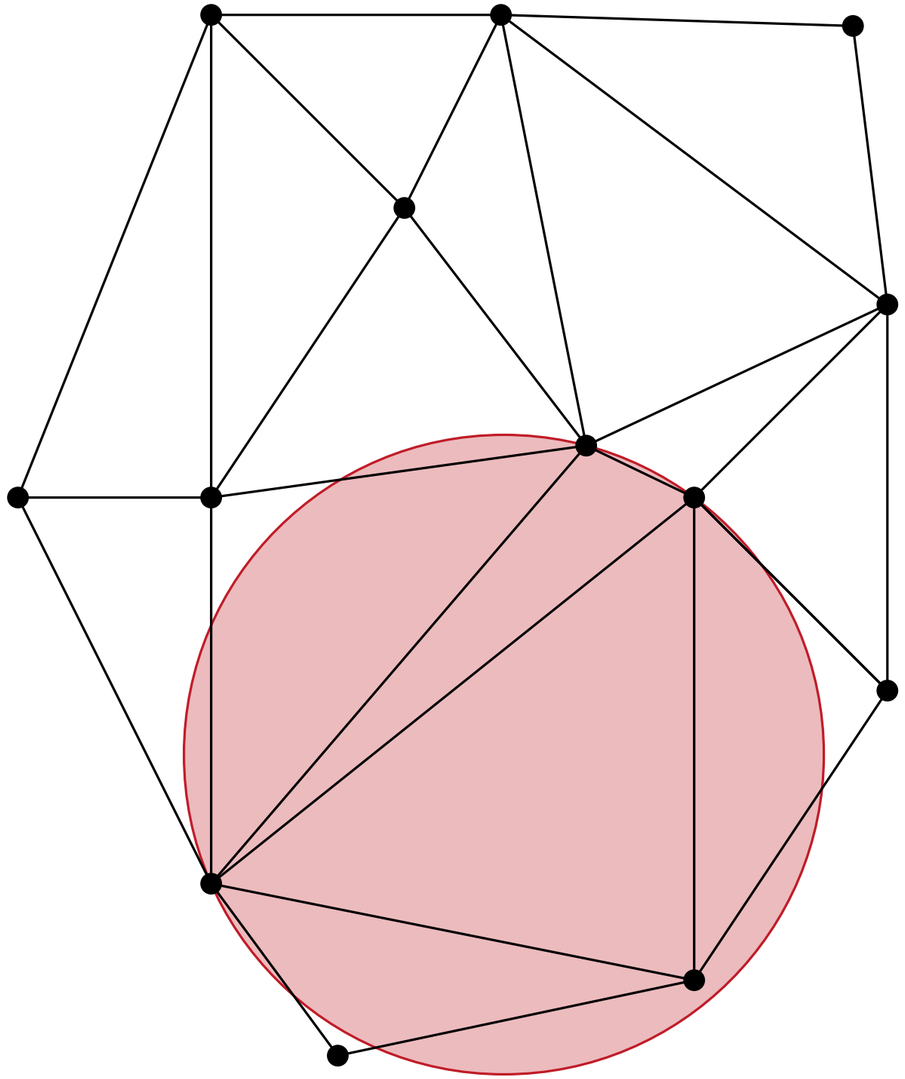
1) Start from a triangulation T

2) While T is not Delaunay :

   Find a "bad" edge and flip it

Flip algorithm

→ "improving triangulation" algorithm that computes a Delaunay triangulation



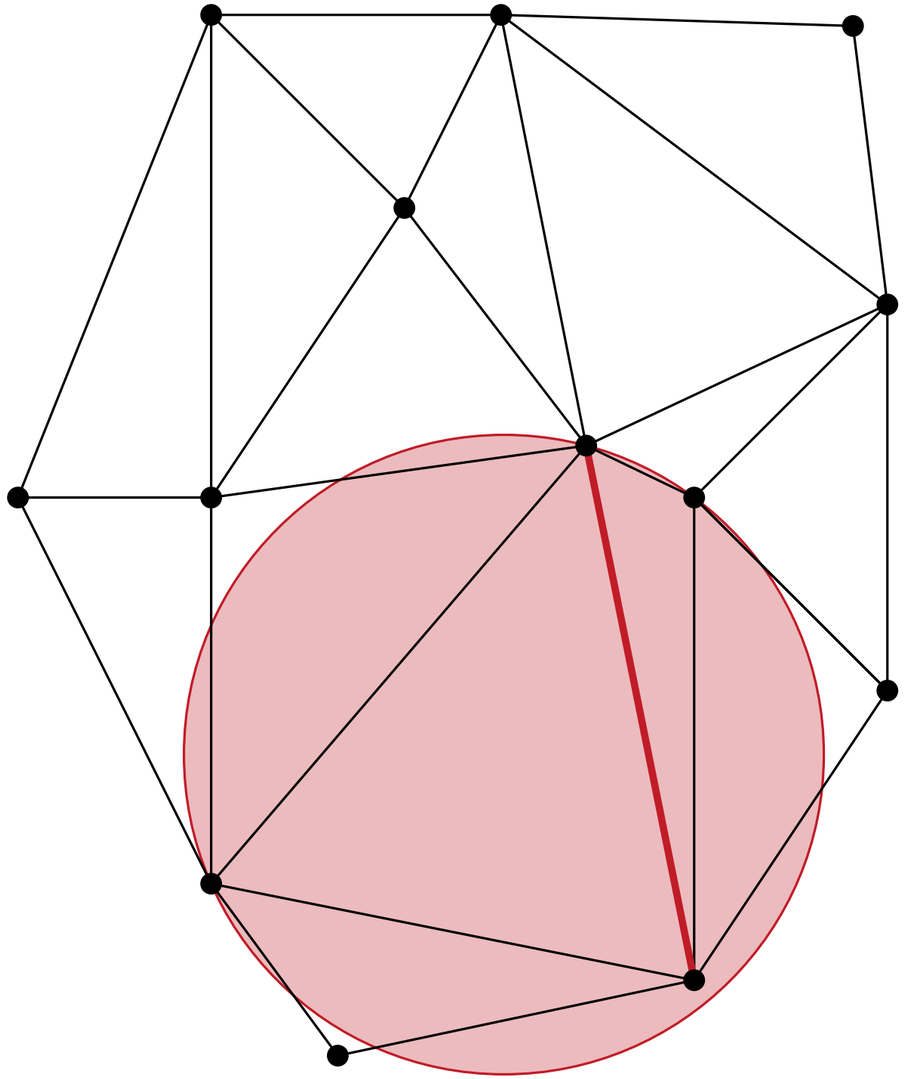
1) Start from a triangulation T

2) While T is not Delaunay :

Find a "bad" edge and flip it

Flip algorithm

→ "improving triangulation" algorithm that computes a Delaunay triangulation



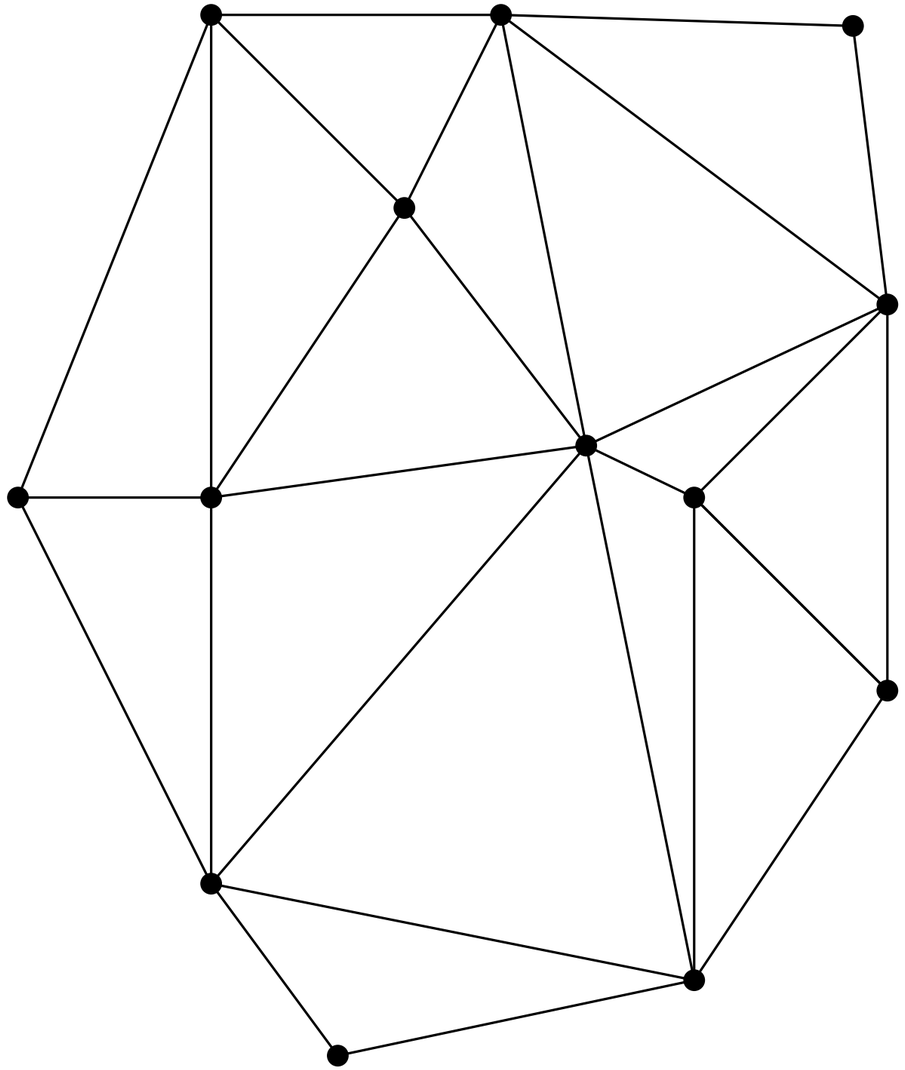
1) Start from a triangulation T

2) While T is not Delaunay :

Find a "bad" edge and flip it

Flip algorithm

⟶ "improving triangulation" algorithm that computes a Delaunay triangulation



1) Start from a triangulation T

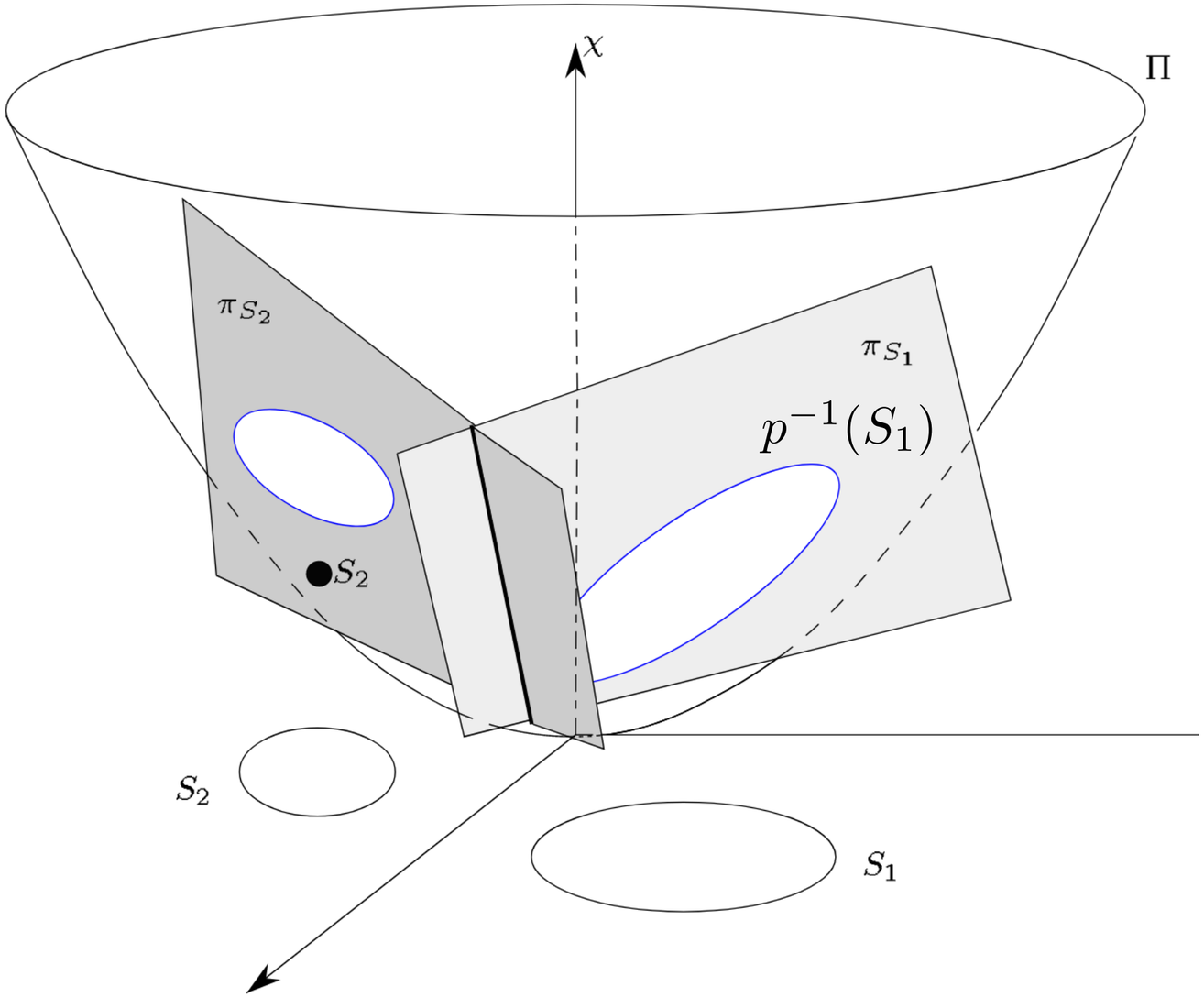
2) While T is not Delaunay :

   Find a "bad" edge and flip it

3) Return T

Delaunay triangulation and hyperboloid

Delaunay triangulation = lower convex hull of points on a paraboloid [Bro79]



Paraboloid Π:

$$x^2 + y^2 - z = 0$$

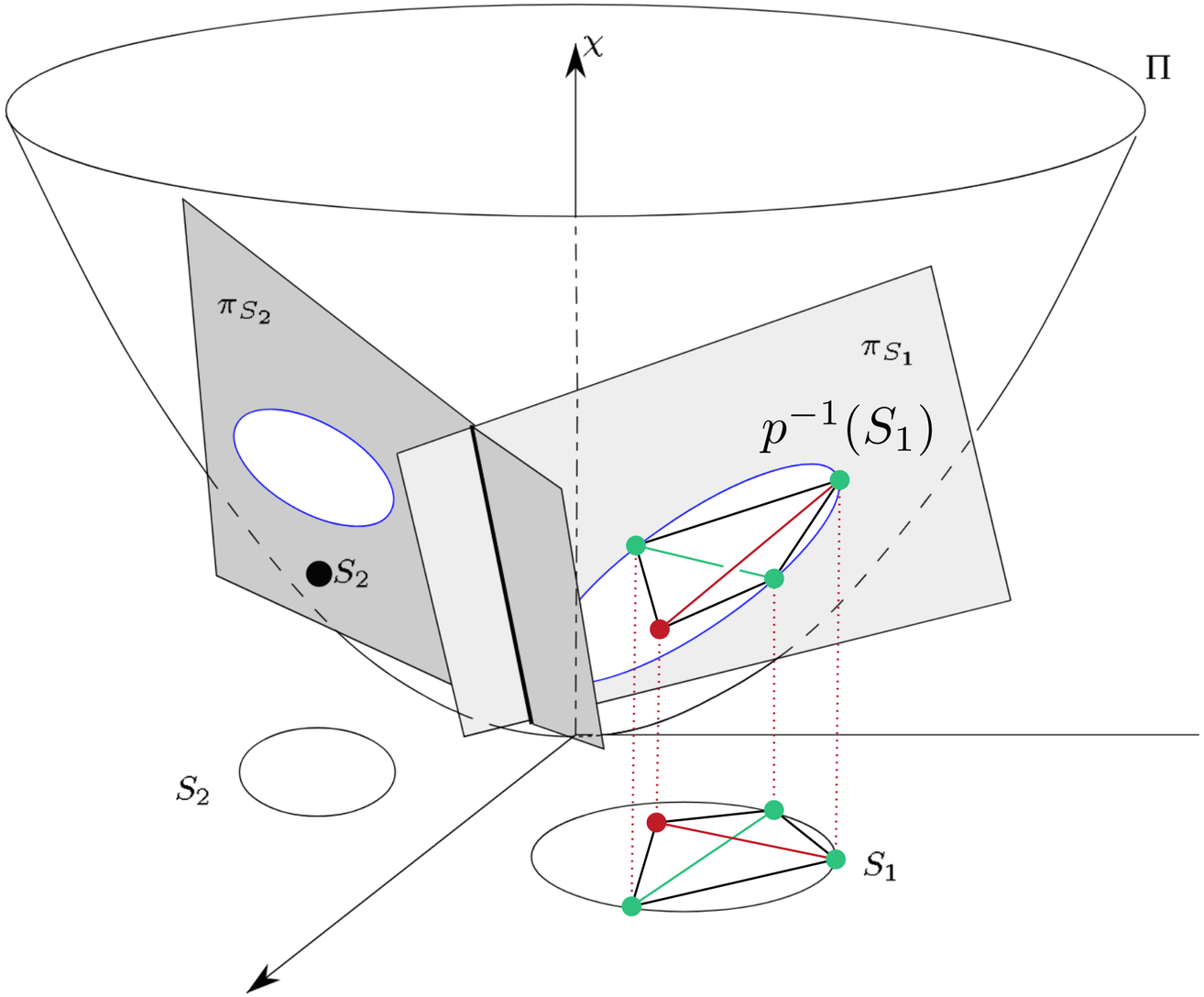
Projection p:

$$p : (x, y, x^2 + y^2) \in \Pi \mapsto (x, y, 0)$$

Prop: $p^{-1}(\text{circle}) = \Pi \cap$ a hyperplane

Delaunay triangulation and hyperboloid

Delaunay triangulation = lower convex hull of points on a paraboloid [Bro79]



Paraboloid Π:

$$x^2 + y^2 - z = 0$$

Projection p:

$$p : (x, y, x^2 + y^2) \in \Pi \mapsto (x, y, 0)$$

Prop: $p^{-1}(\text{circle}) = \Pi \cap$ a hyperplane

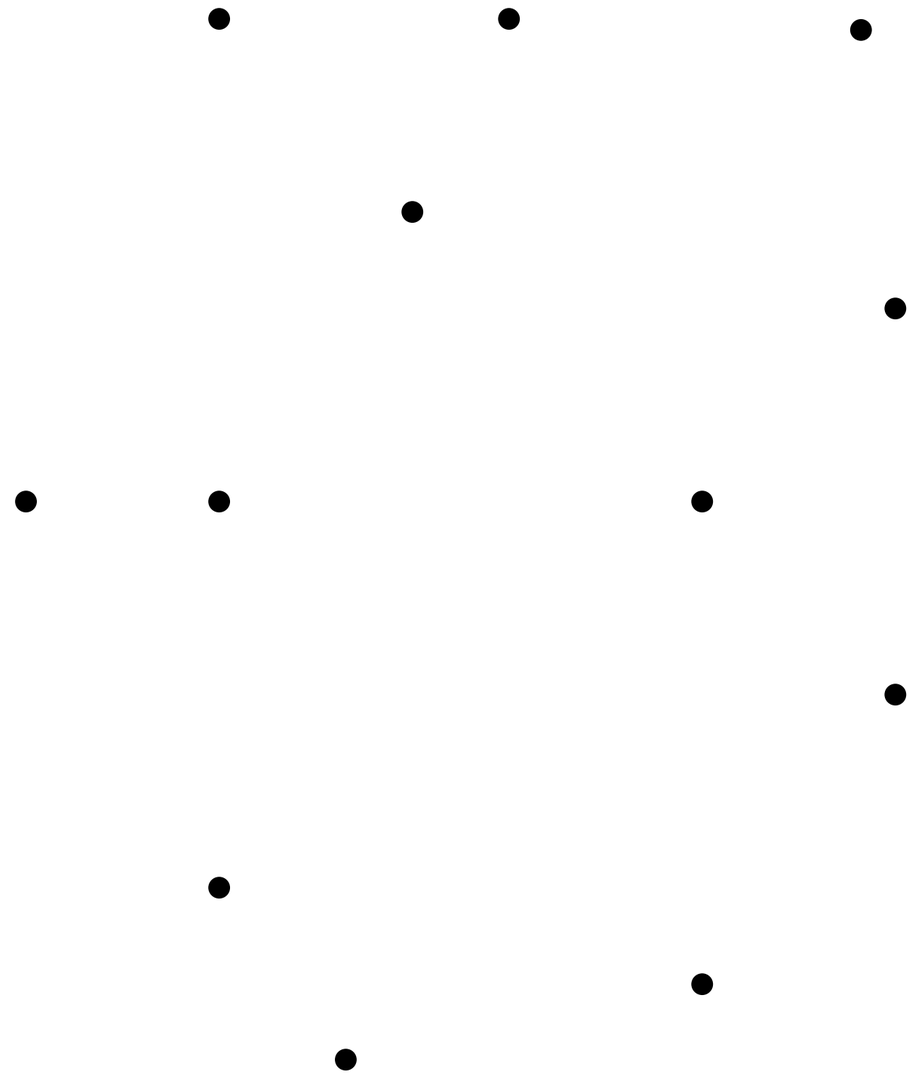
Prop: Let V be a set of points in the plane, let C be the lower convex hull of $p^{-1}(V)$. Then $\text{DL}(V) = p(C)$.

→ Delaunay triangulation exists

→ Flip algorithm is correct

Divide and conquer algorithm

→ Divide and conquer algorithm that computes a Delaunay triangulation



Recursion principle :

Function DL(V) :

If $|V| \leq 3$:

Return convhull(V)

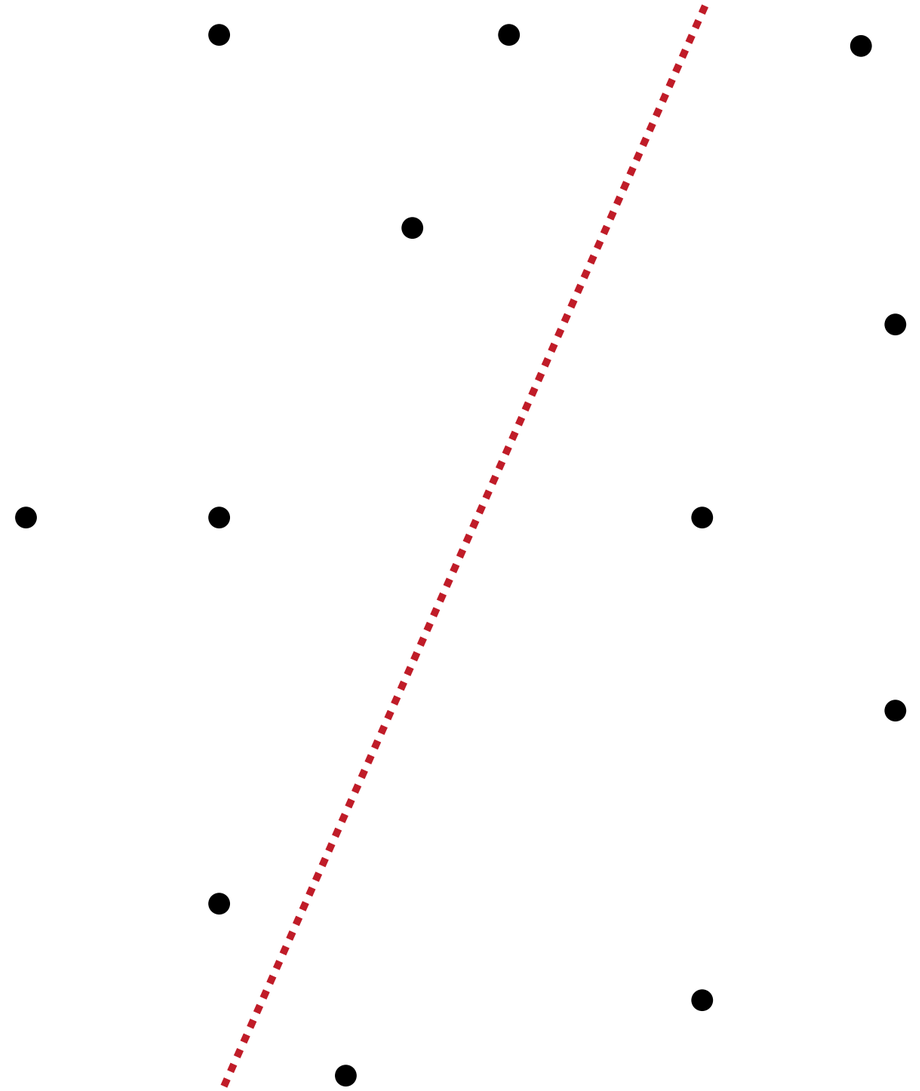
Else :

$V = V' \cup V''$

Return Merge(DL(V'), DL(V''))

Divide and conquer algorithm

→ Divide and conquer algorithm that computes a Delaunay triangulation



Recursion principle :

Function DL(V) :

If $|V| \leq 3$:

Return convhull(V)

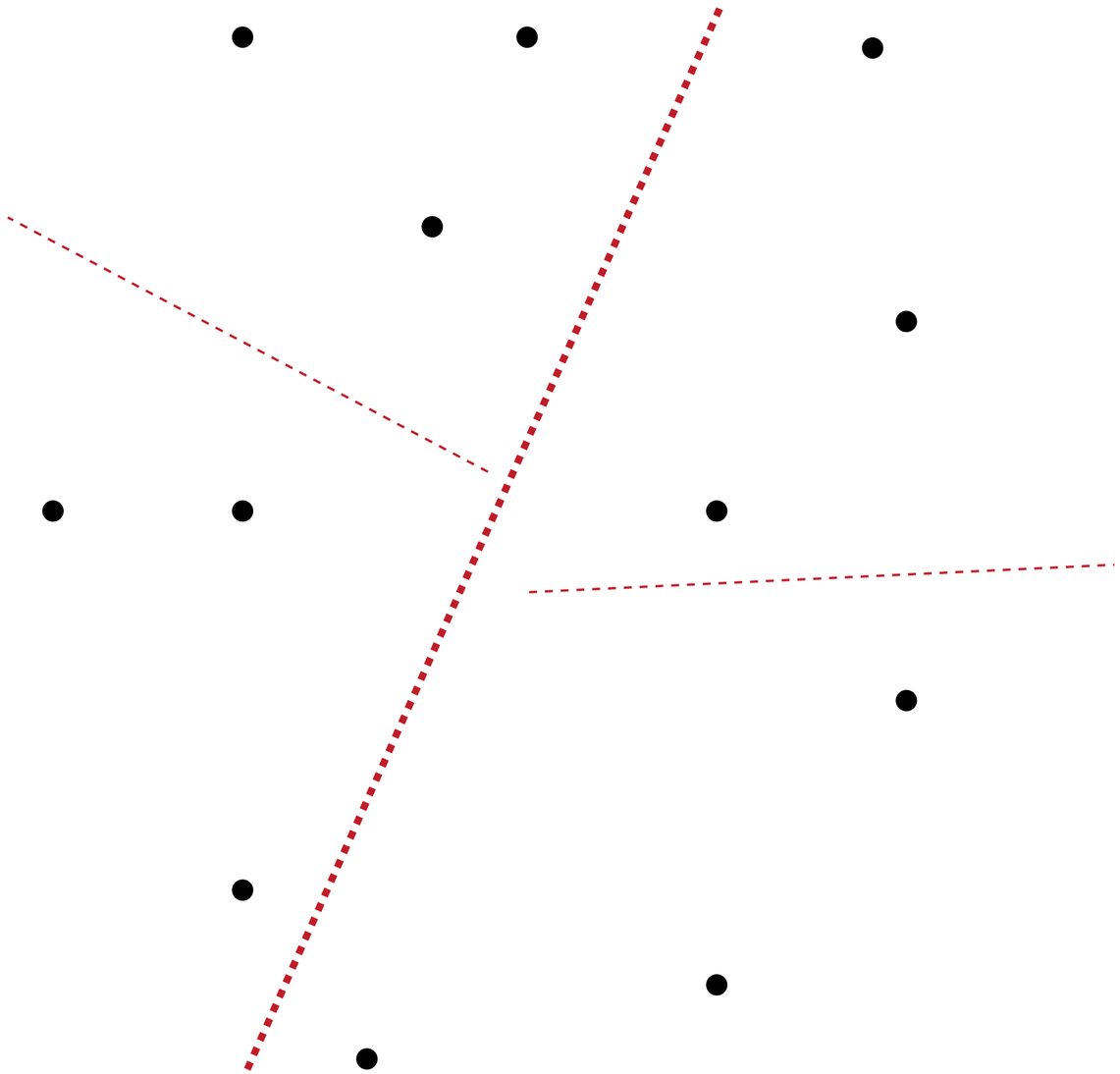
Else :

$V = V' \cup V''$

Return Merge(DL(V'), DL(V''))

Divide and conquer algorithm

→ Divide and conquer algorithm that computes a Delaunay triangulation



Recursion principle :

Function DL(V) :

If $|V| \leq 3$:

Return convhull(V)

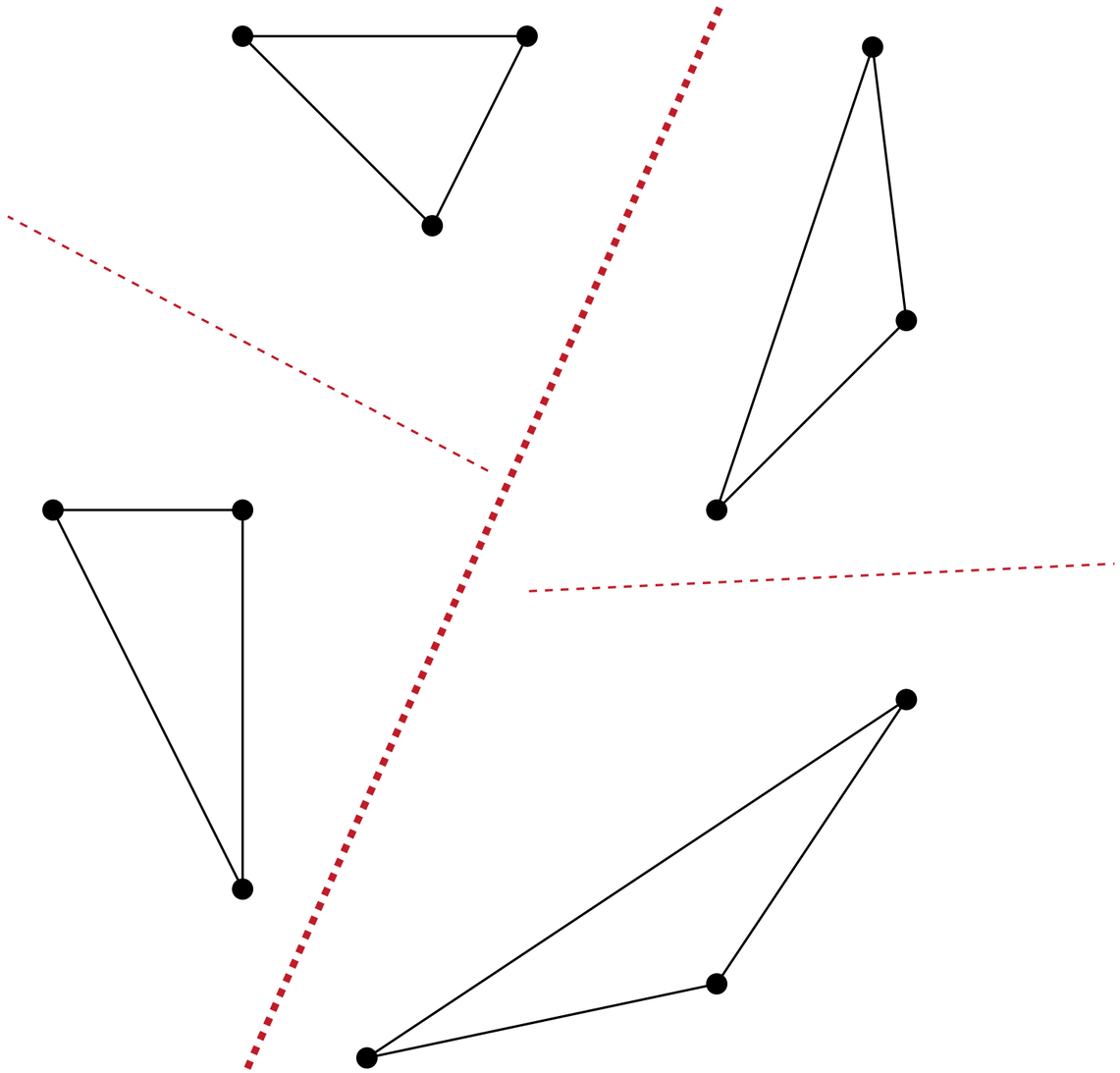
Else :

$V = V' \cup V''$

Return Merge(DL(V'), DL(V''))

Divide and conquer algorithm

→ Divide and conquer algorithm that computes a Delaunay triangulation



Recursion principle :

Function DL(V) :

If $|V| \leq 3$:

Return convhull(V)

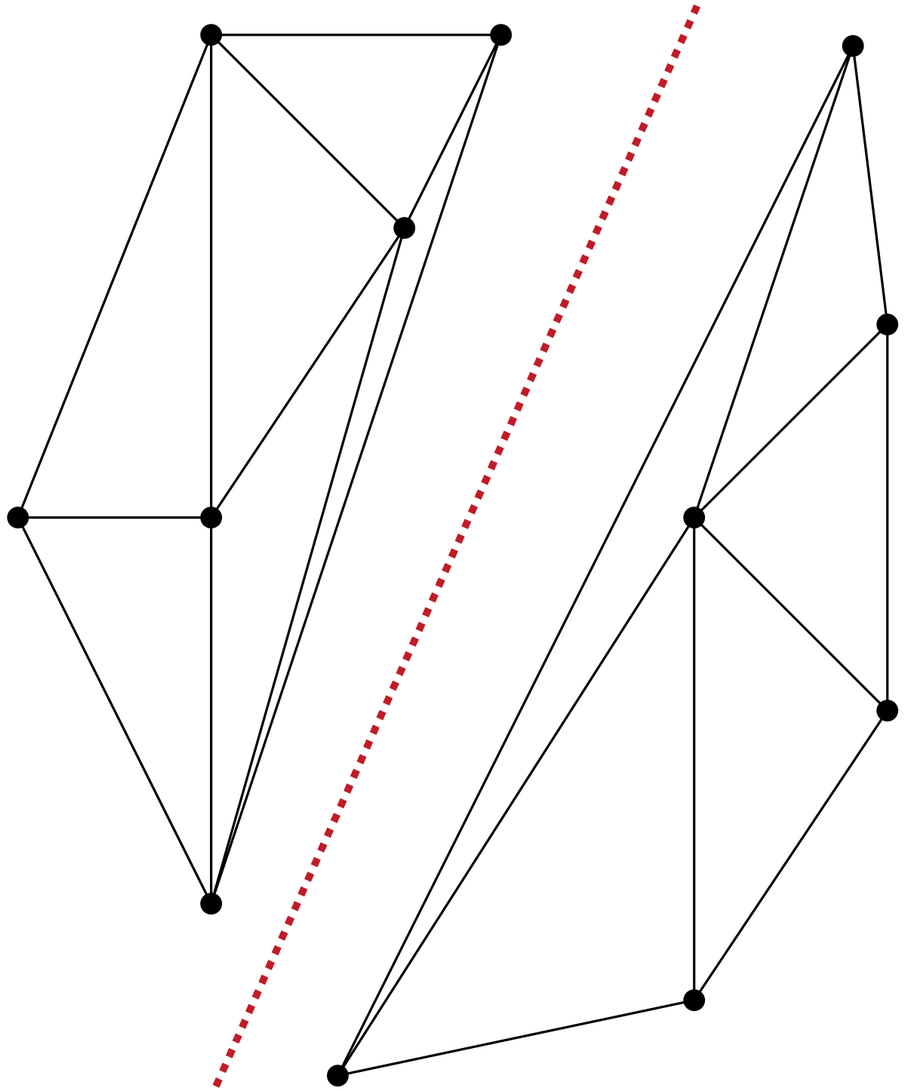
Else :

$V = V' \cup V''$

Return Merge(DL(V'), DL(V''))

Divide and conquer algorithm

→ Divide and conquer algorithm that computes a Delaunay triangulation



Recursion principle :

Function DL(V) :

If $|V| \leq 3$:

Return convhull(V)

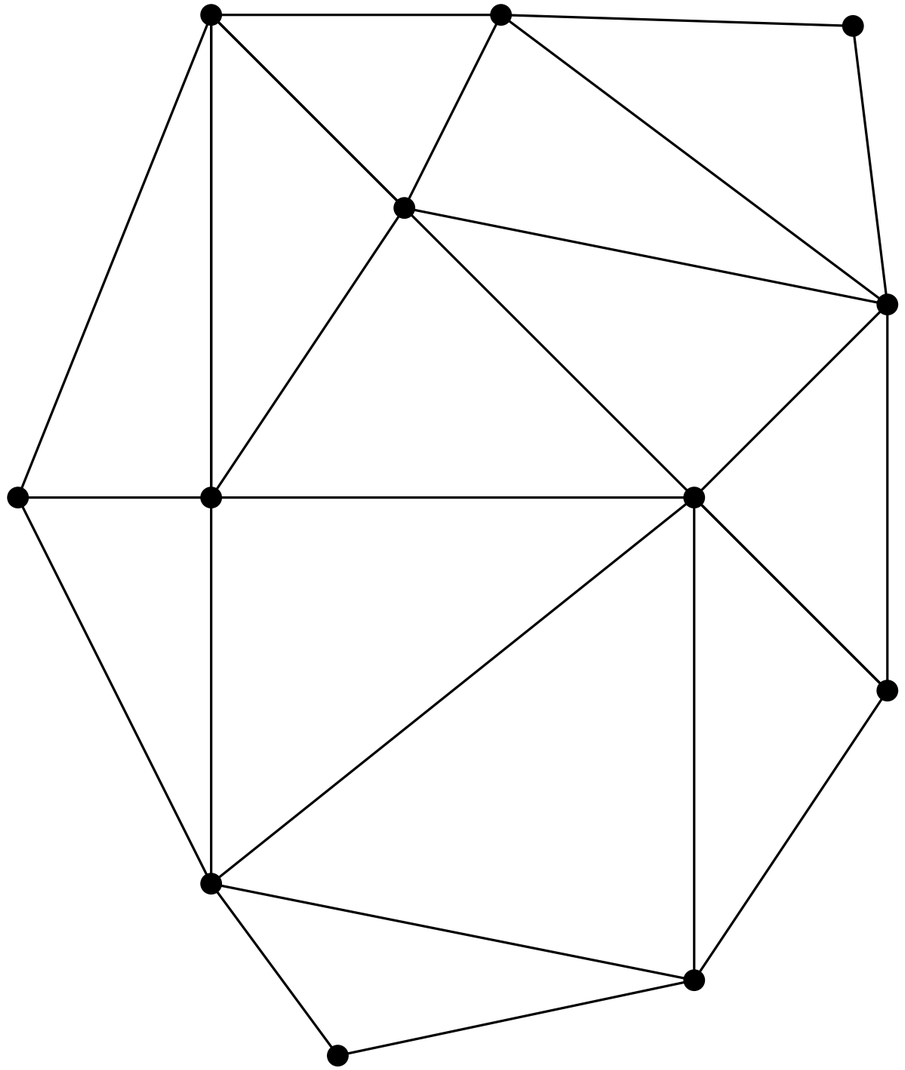
Else :

$V = V' \cup V''$

Return Merge(DL(V'), DL(V''))

Divide and conquer algorithm

→ Divide and conquer algorithm that computes a Delaunay triangulation



Recursion principle :

Function DL(V) :

If $|V| \leq 3$:

Return convhull(V)

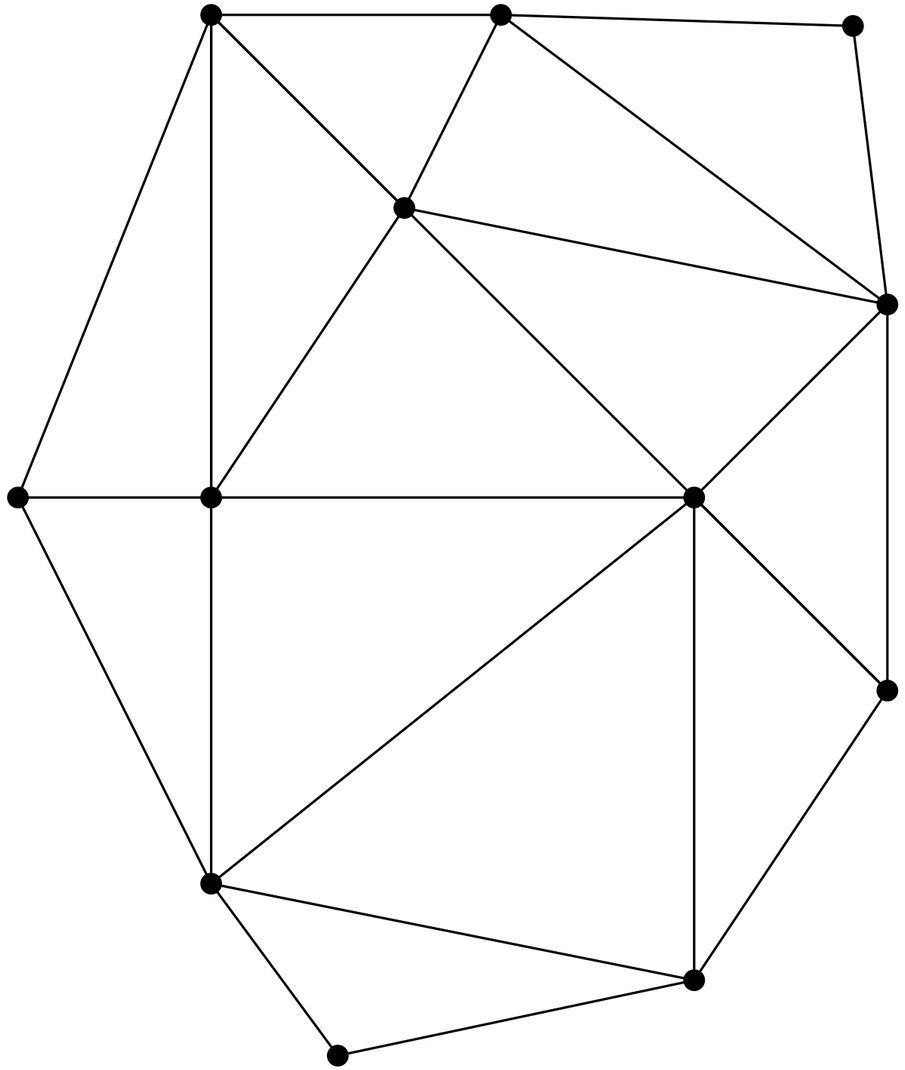
Else :

$V = V' \cup V''$

Return Merge(DL(V'), DL(V''))

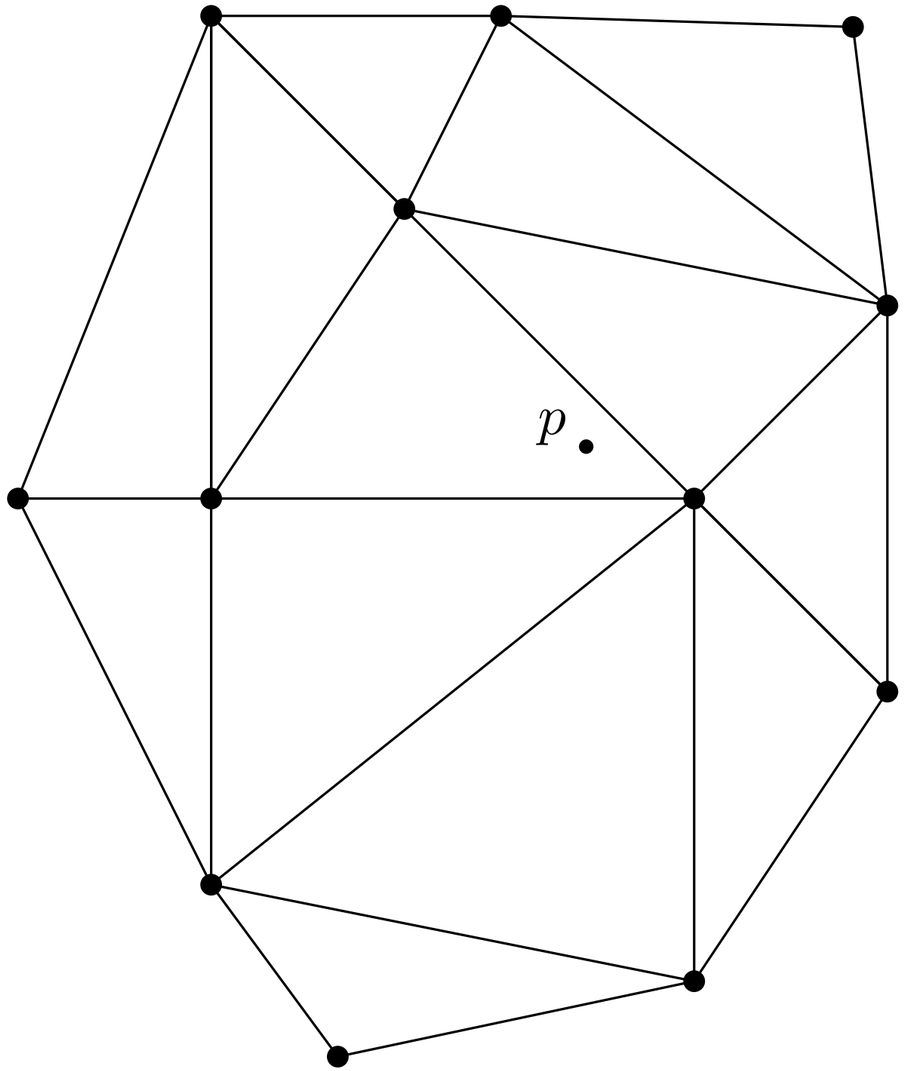
Bowyer-Watson (BW) algorithm

→ Incremental algorithm that computes a Delaunay triangulation



Bowyer-Watson (BW) algorithm

→ Incremental algorithm that computes a Delaunay triangulation

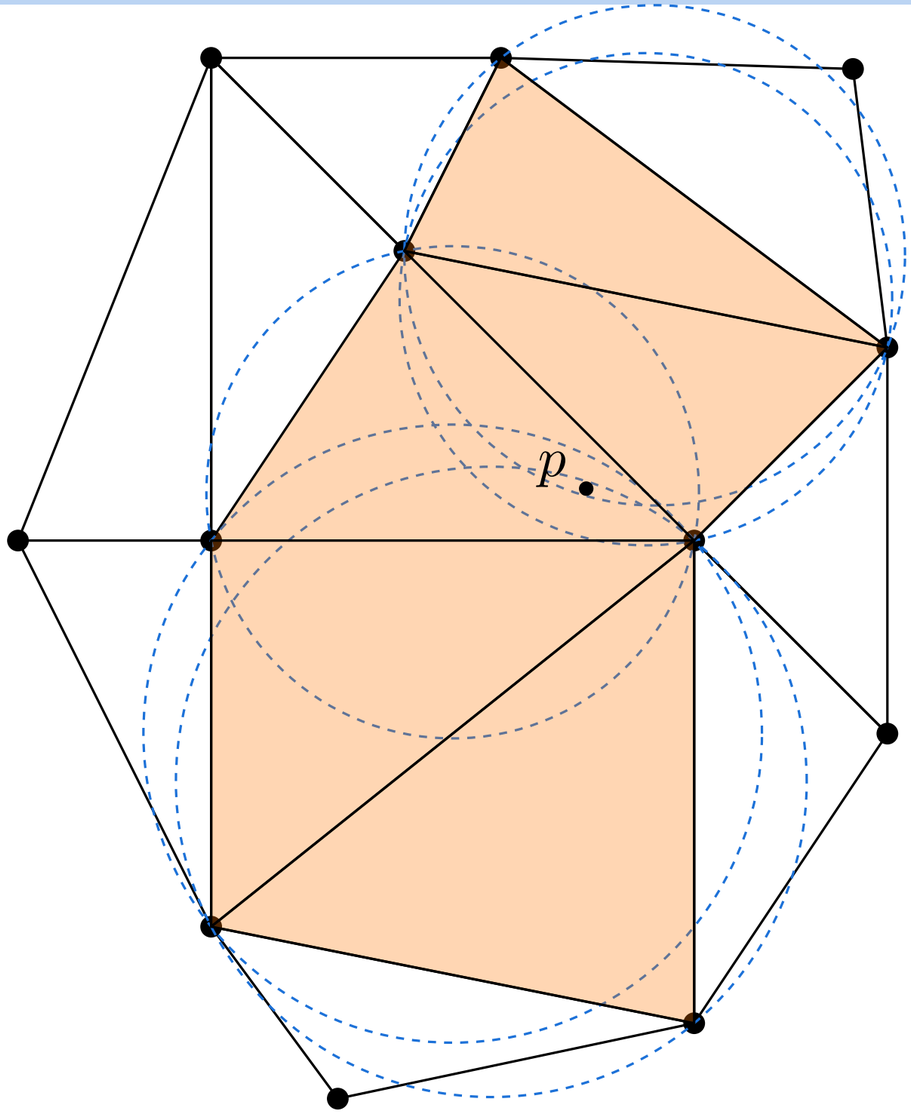


Insertion step :

1) Insert a point p

Bowyer-Watson (BW) algorithm

→ Incremental algorithm that computes a Delaunay triangulation



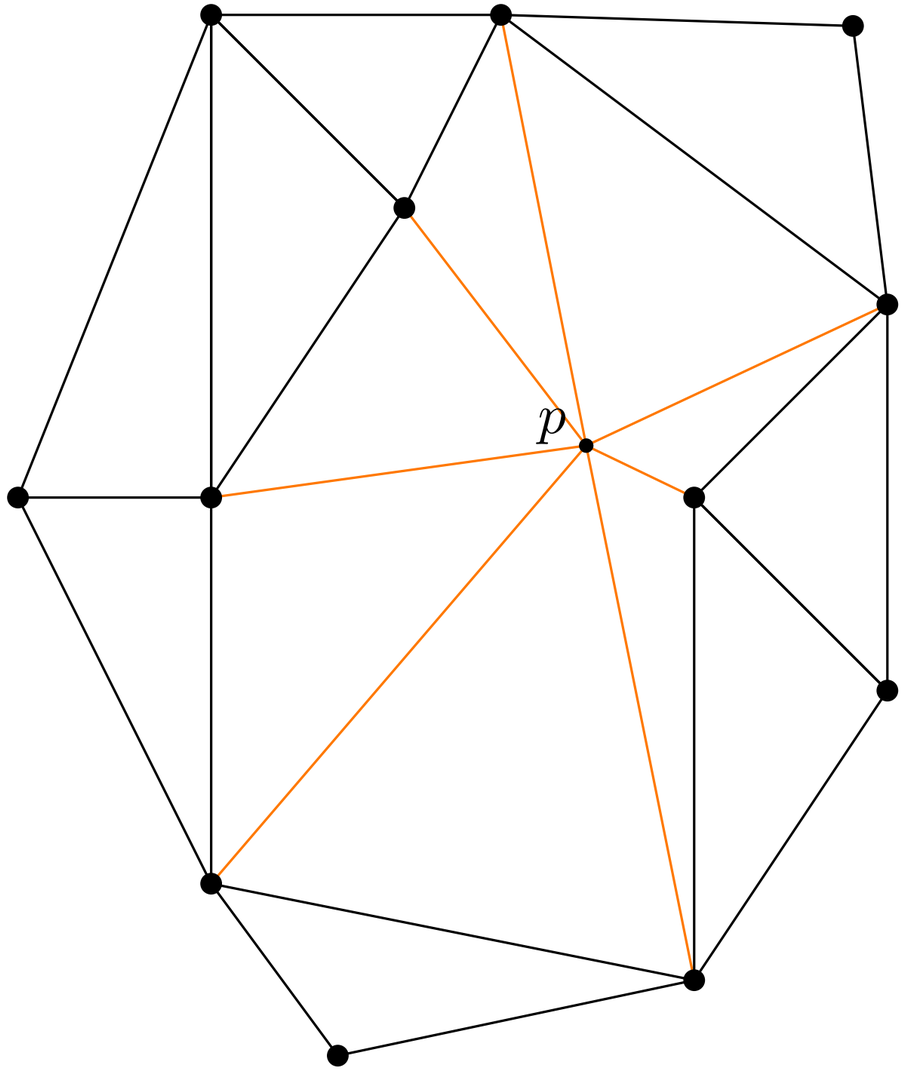
Insertion step :

1) Insert a point p

2) Compute triangles whose circumscribed disk contains p : the conflict zone.

Bowyer-Watson (BW) algorithm

→ Incremental algorithm that computes a Delaunay triangulation



Insertion step :

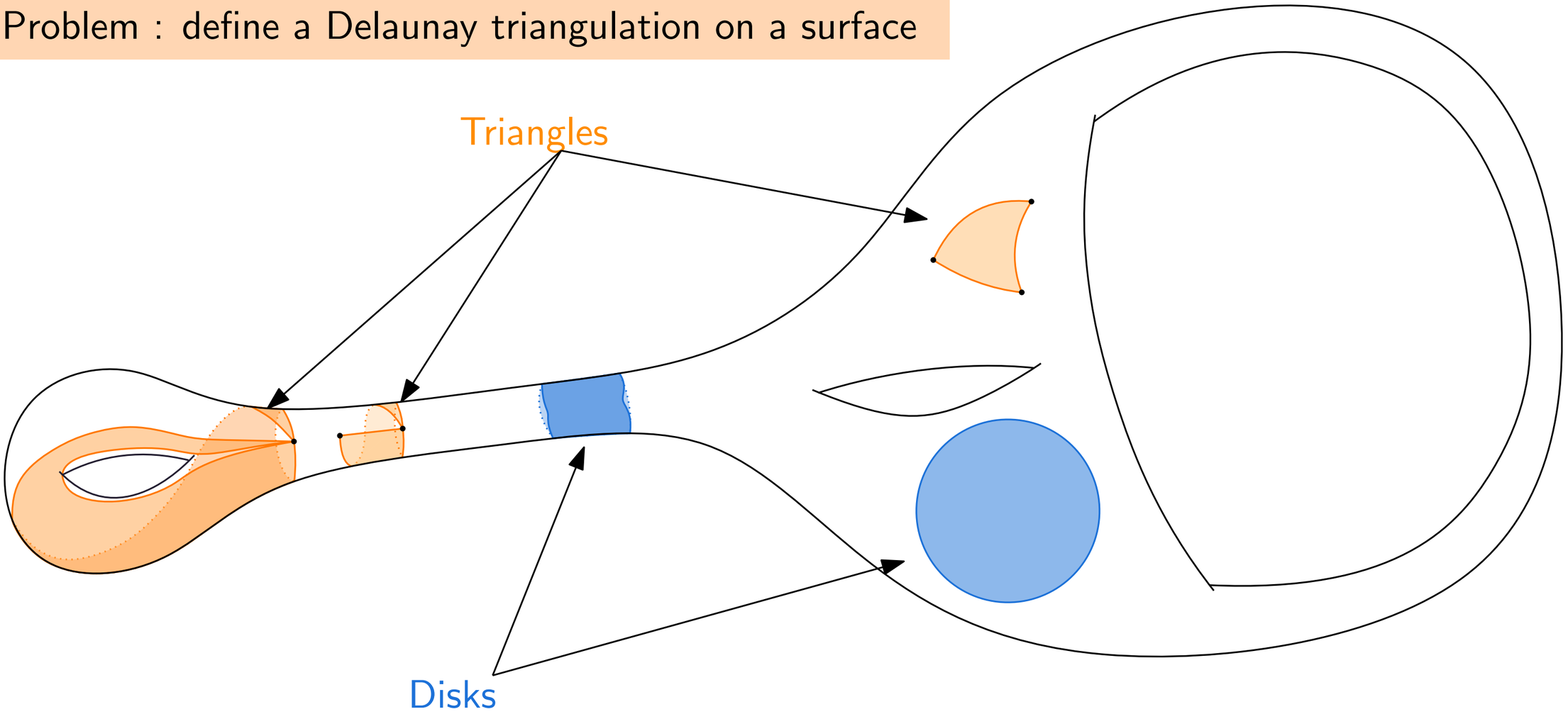
1) Insert a point p

2) Compute triangles whose circumscribed disk contains p : the conflict zone.

3) Connect p to vertices of these triangles.

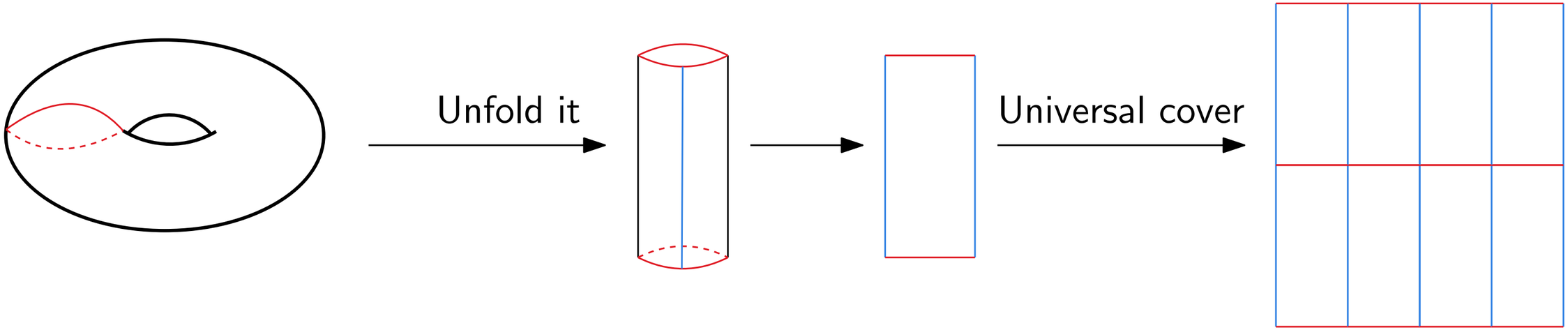
Topological problem : weird triangles and disks

Problem : define a Delaunay triangulation on a surface

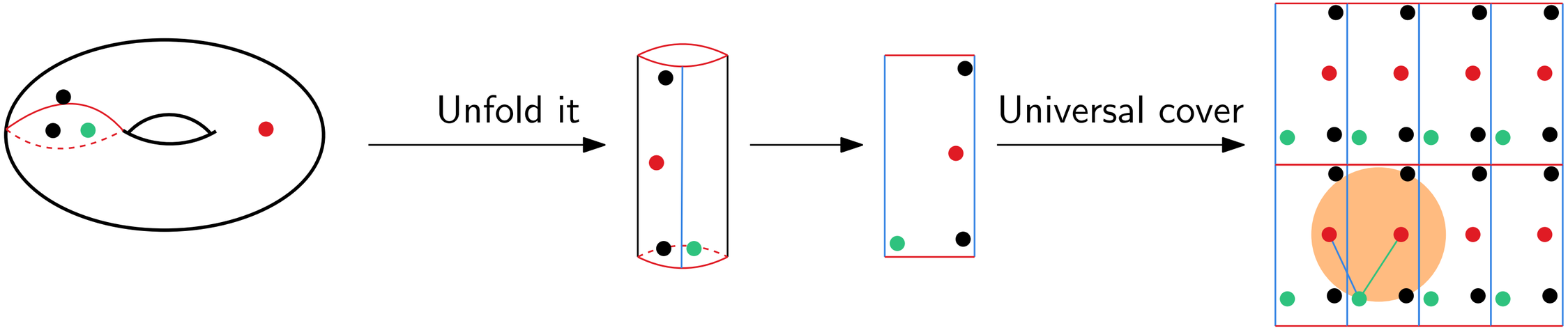


- ▷ Delaunay triangulation in Euclidean plane
- ▶ BW-algorithm on a torus ($g = 1$)
- ▷ Hyperbolic plane
- ▷ BW-algorithm on the Bolza surface ($g = 2$)
- ▷ BW-algorithm on hyperbolic surface ($g \geq 2$)

BW-algorithm on a flat Torus

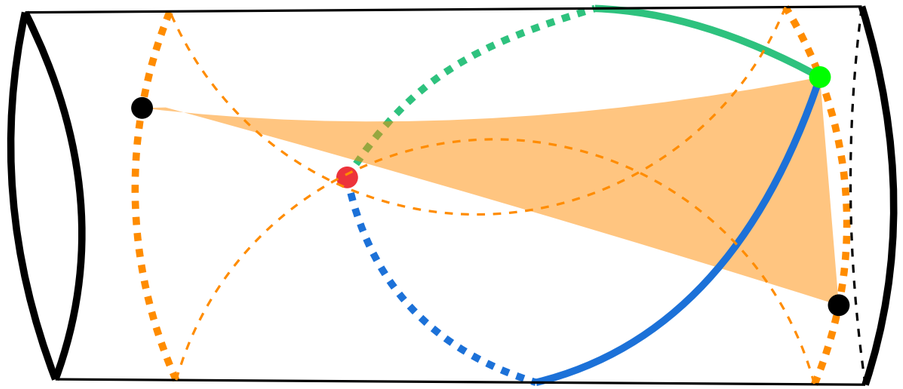


BW-algorithm on a flat Torus

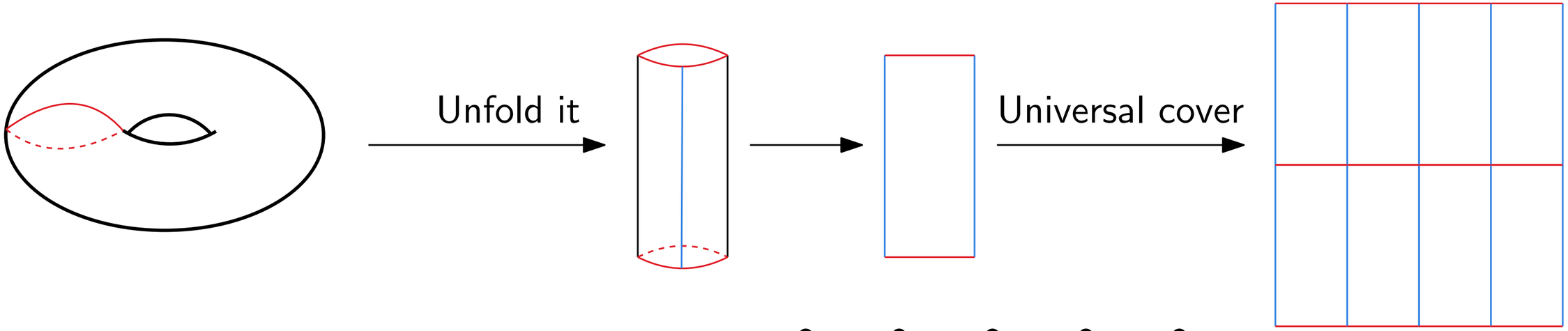


Avoid non-embedded disk!

↳ Which path ?

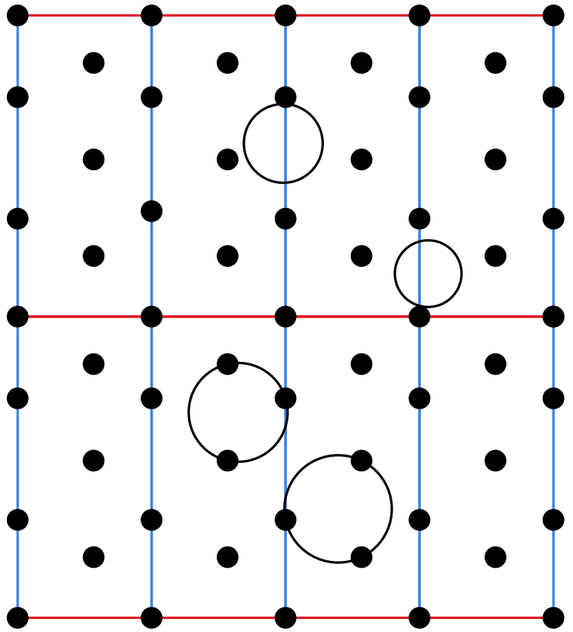


BW-algorithm on a flat Torus



1) Add some "dummy" points forming a "ε-net" to avoid non-embedded disk.

2) Continue the insertion with the BW-algorithm.



- ▷ Delaunay triangulation in Euclidean plane
- ▷ BW-algorithm on a torus ($g = 1$)
- ▶ Hyperbolic plane
- ▷ BW-algorithm on the Bolza surface ($g = 2$)
- ▷ BW-algorithm on hyperbolic surface ($g \geq 2$)

Disks models of hyperbolic plane

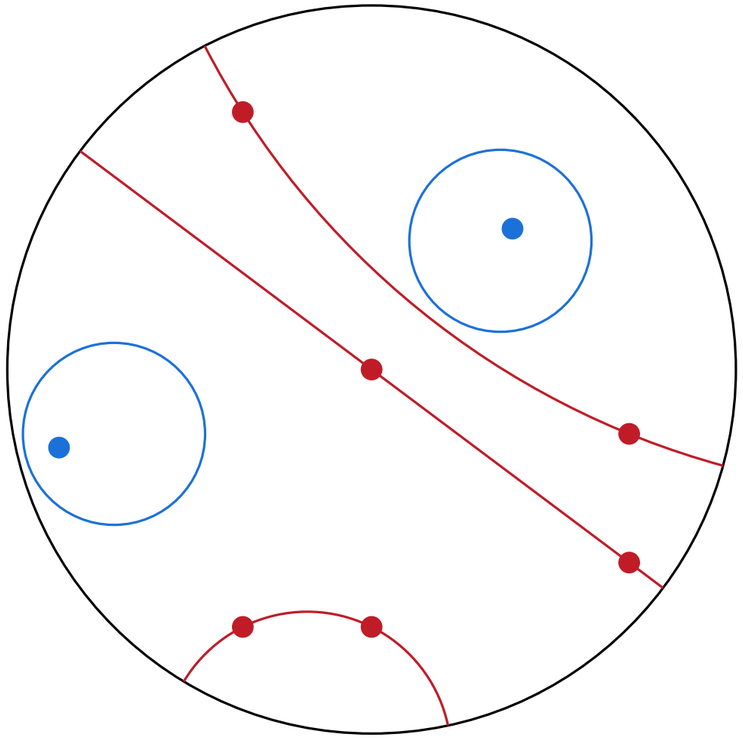
Poincaré disk model

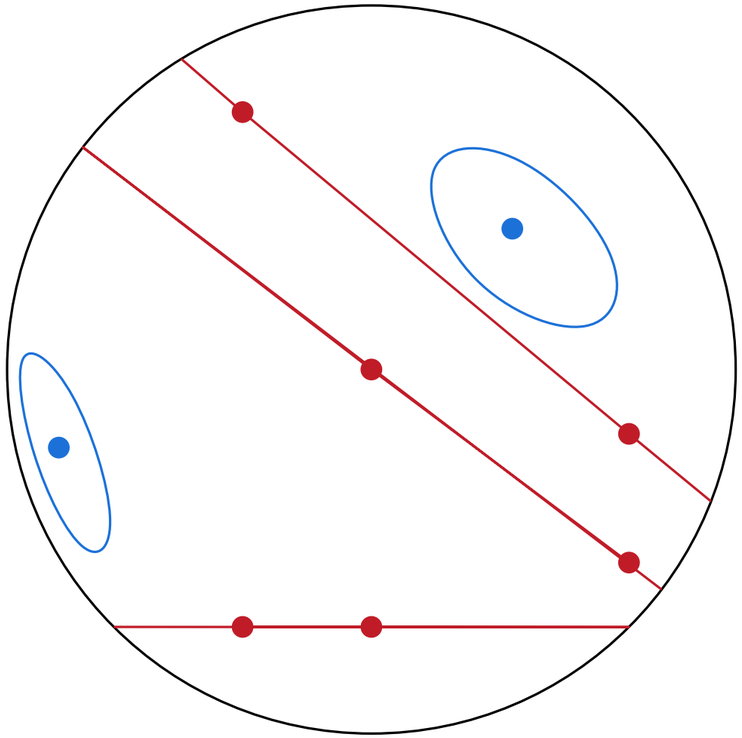
- circle = Euclidean circle
- line = circle wich is orthogonal to the unit circle
- preserves angles

Do your Choice!

Klein-Beltrami disk model

- circle = Euclidean ellipse
- line = Euclidean line
- (Useful for the hyperbolic extended plane model)





Disks models of hyperbolic plane

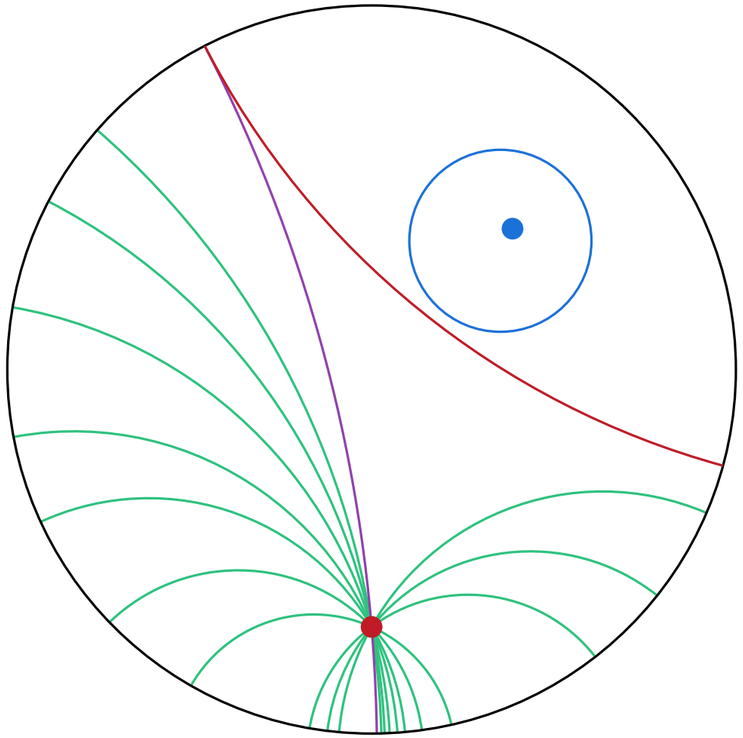
Poincaré disk model

- circle = Euclidean circle
- line = circle wich is orthogonal to the unit circle
- preserves angles

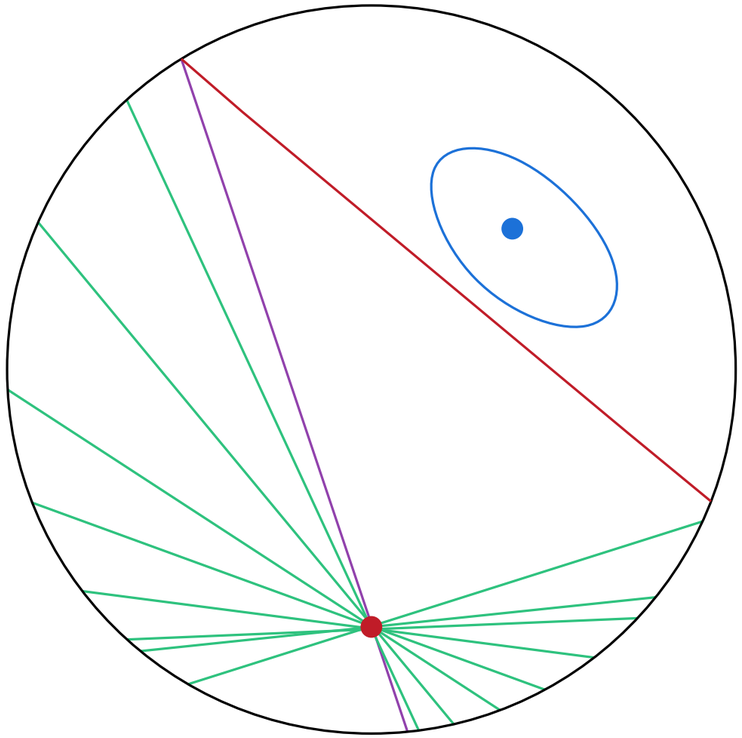
Do your Choice!

Klein-Beltrami disk model

- circle = Euclidean ellipse
- line = Euclidean line
- (Useful for the hyperbolic extended plane model)



Infinite number of parallel lines to the red one passing through a point

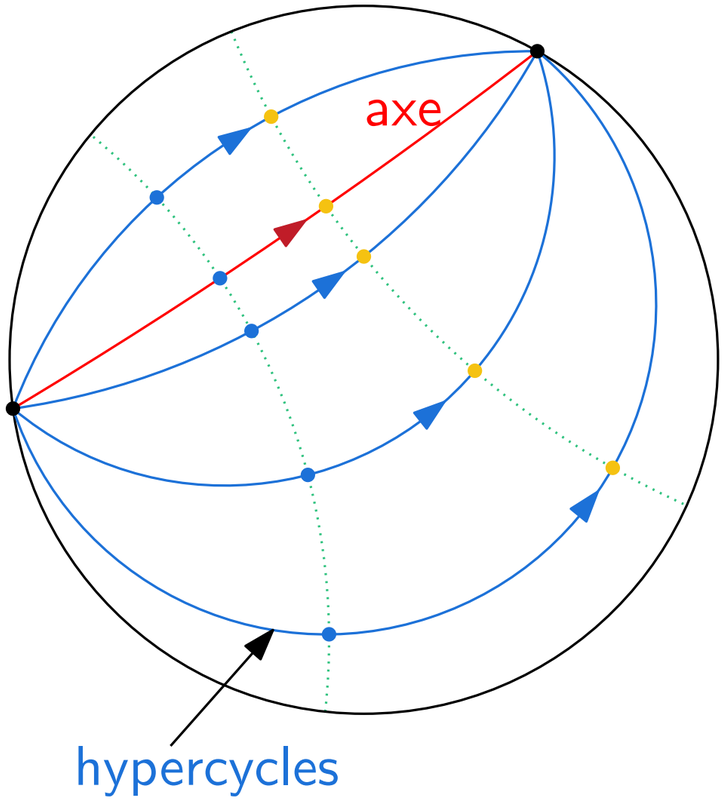


Triangles in hyperbolic plane

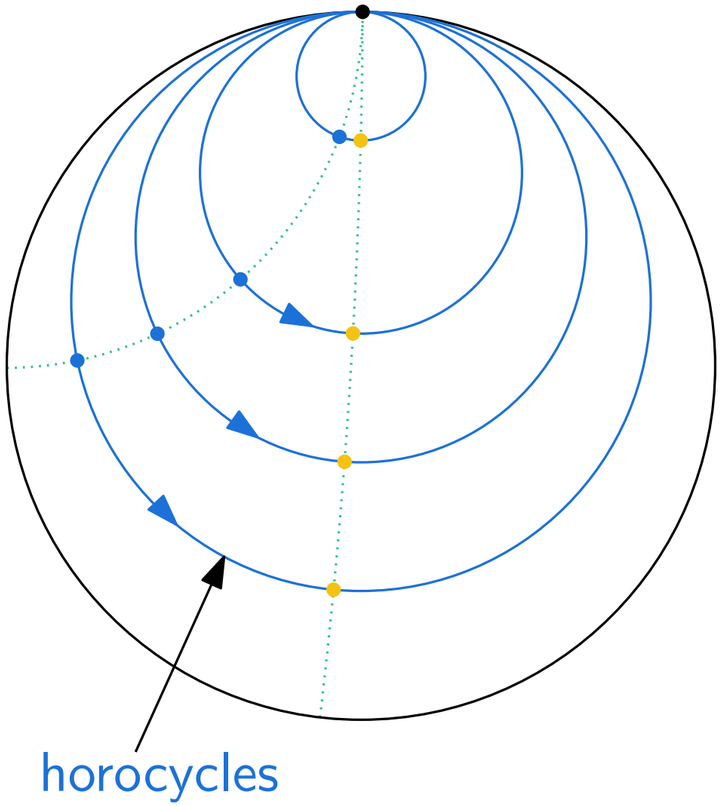
Prop : $\mathrm{Biholom}(D) = \{z \mapsto \lambda \frac{\rho - z}{1 - \overline{\rho} z} \mid \lambda \in U, \rho \in D\}$

3 types of orientation-preserving isometries.

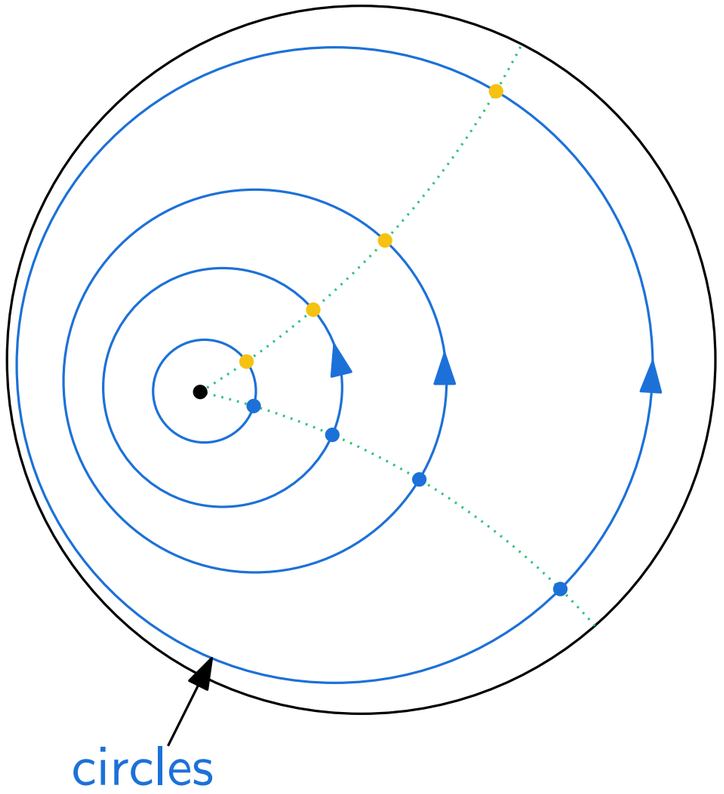
Hyperbolique



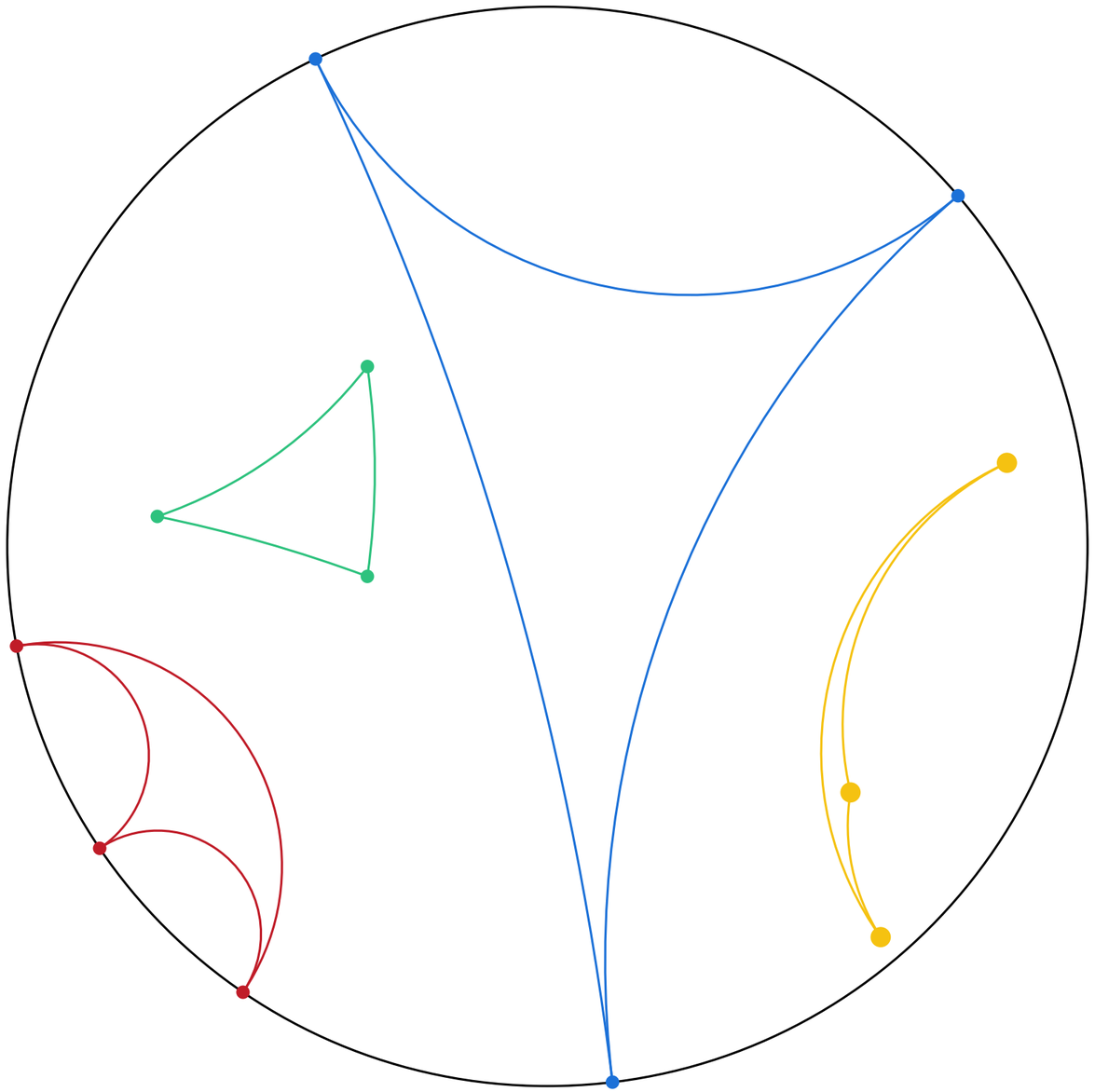
Parabolique



Elliptique

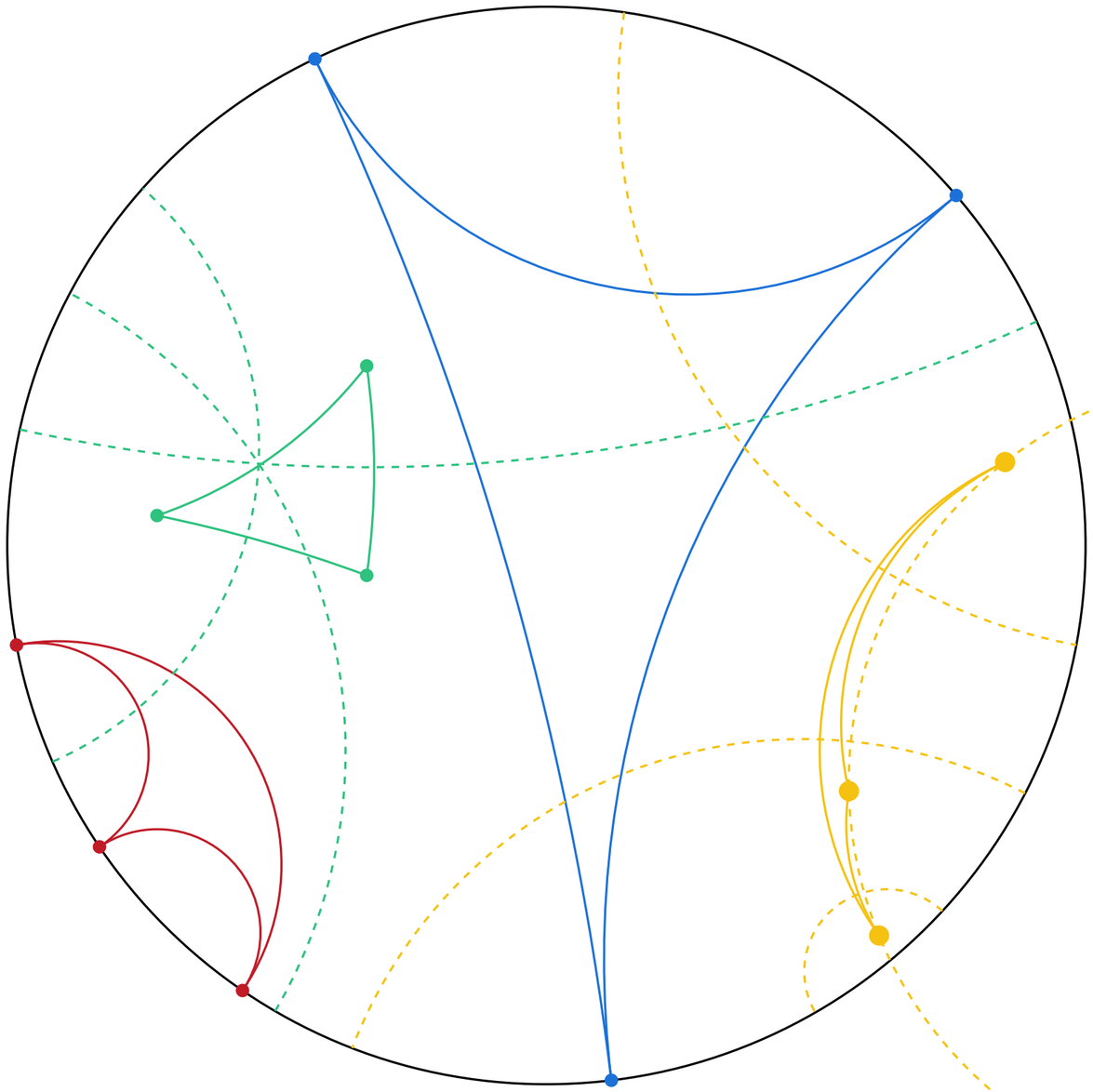


Triangles in hyperbolic plane



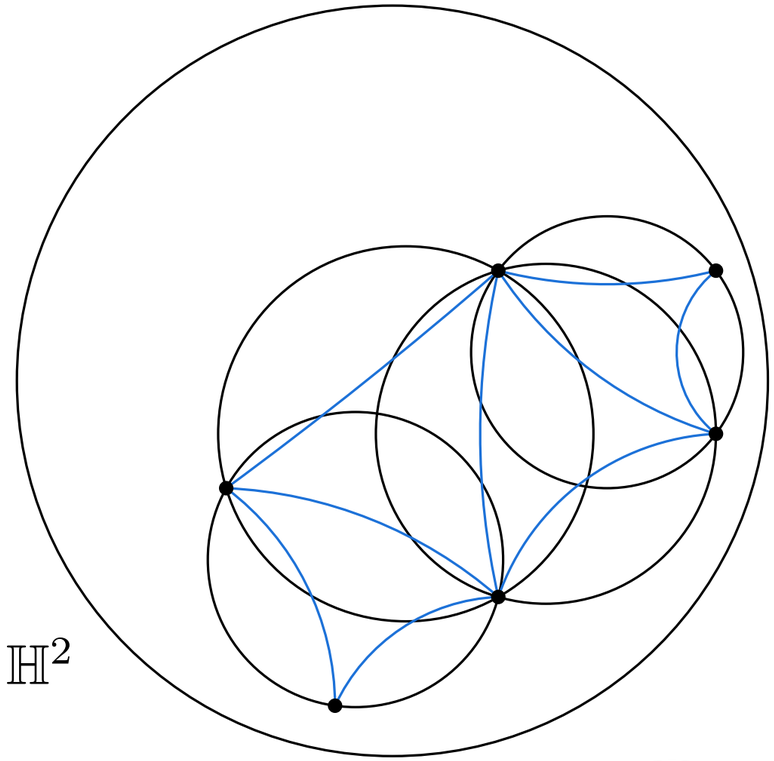
- Area(triangle) $= \pi - \sum$ angles

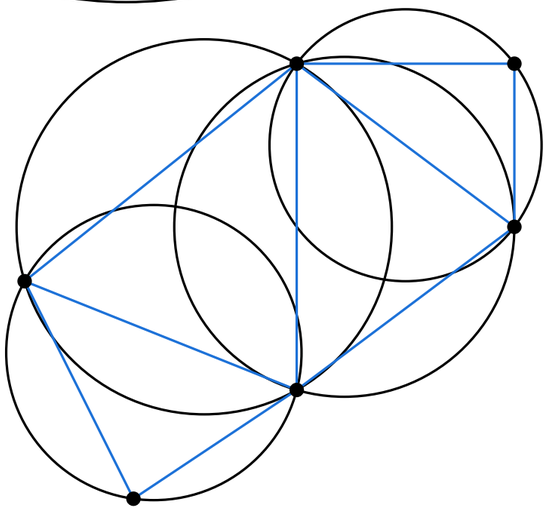
Triangles in hyperbolic plane



- Area(triangle) $= \pi - \sum$ angles
- Circumcircle may not exist!

Triangles in hyperbolic plane

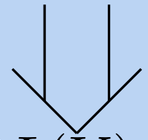




Plane

- Area(triangle) $= \pi - \sum$ angles
- Circumcircle may not exist!

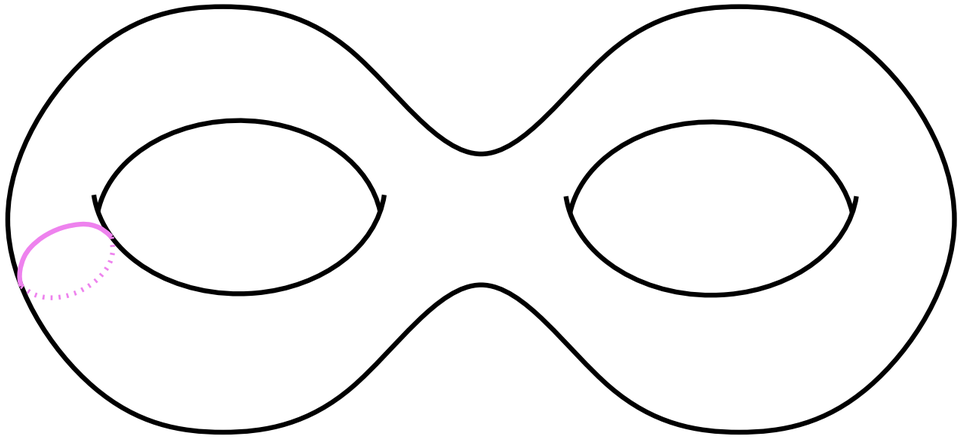
Existence of $DL(V)$ in Euclidean plane



Existence of $DL(V)$ in hyperbolic plane

- ▷ Delaunay triangulation in Euclidean plane
- ▷ BW-algorithm on a torus ($g = 1$)
- ▷ Hyperbolic plane
- ▶ BW-algorithm on the Bolza surface ($g = 2$)
- ▷ BW-algorithm on hyperbolic surface ($g \geq 2$)

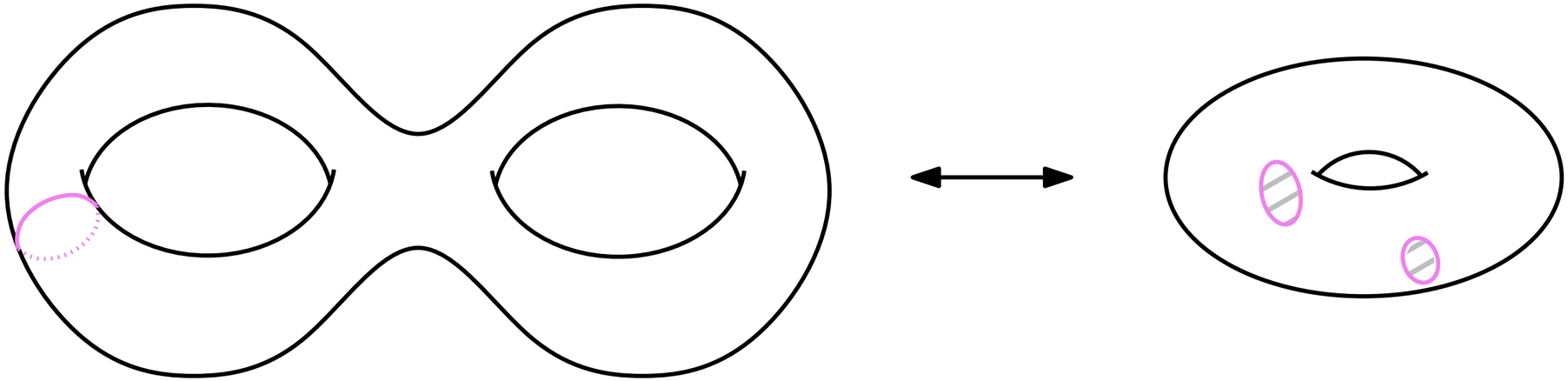
Unfolding the Bolza surface



hyperbolic surface: a surface endowed with a Riemannian metric of constant negative curvature.

Bolza surface: a hyperbolic surface with the simplest possible topology (genus 2).

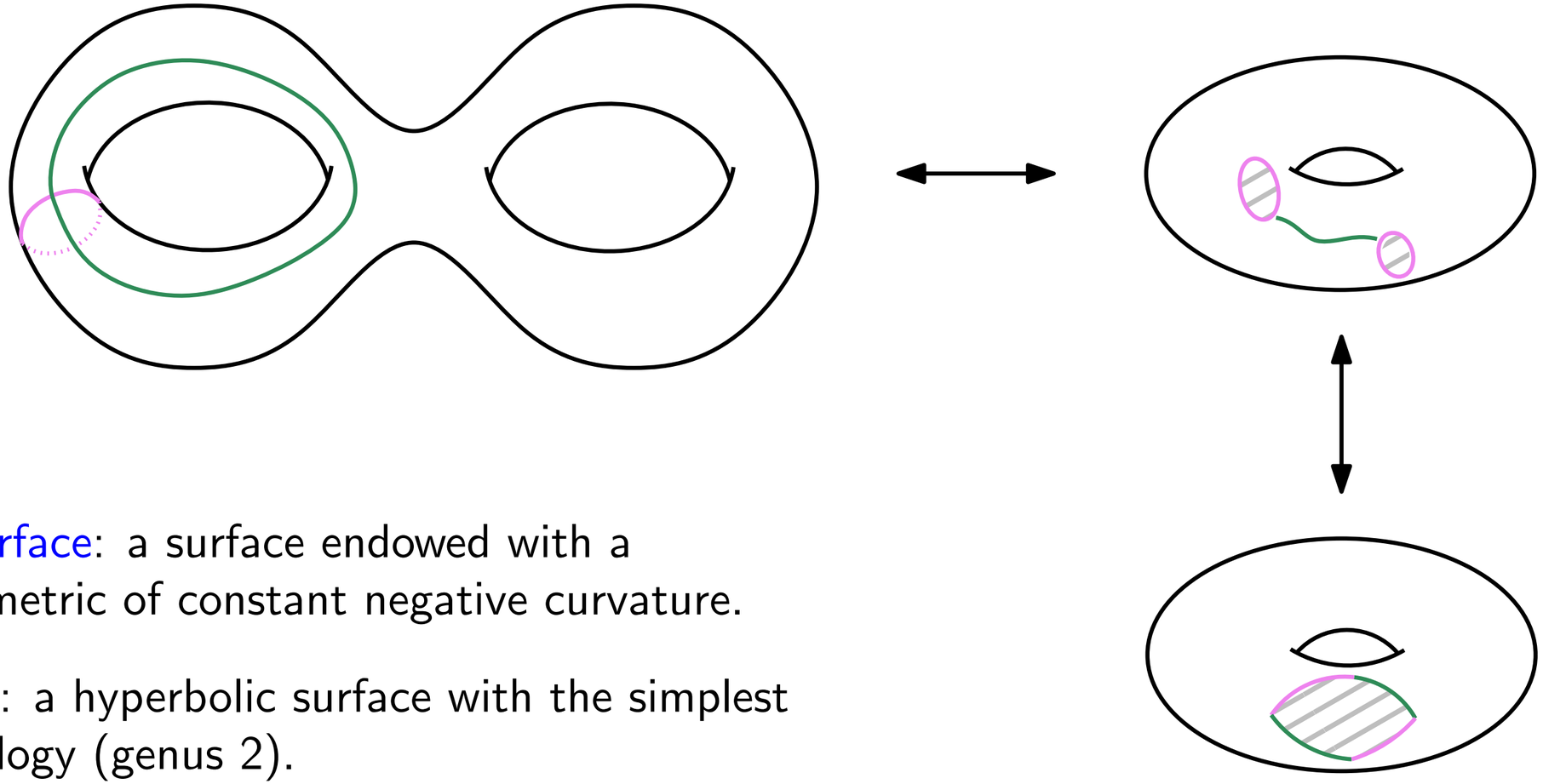
Unfolding the Bolza surface



hyperbolic surface: a surface endowed with a Riemannian metric of constant negative curvature.

Bolza surface: a hyperbolic surface with the simplest possible topology (genus 2).

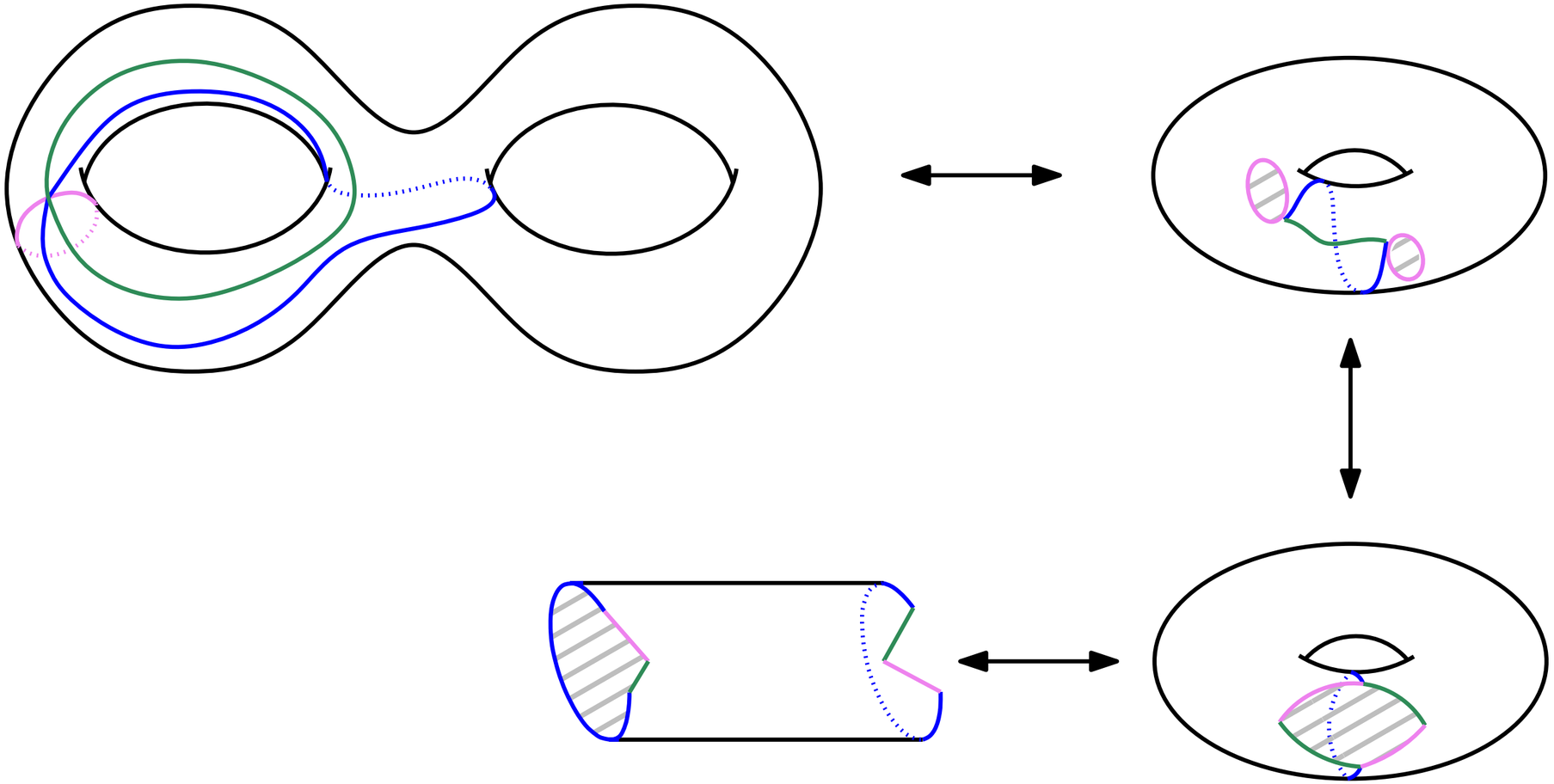
Unfolding the Bolza surface



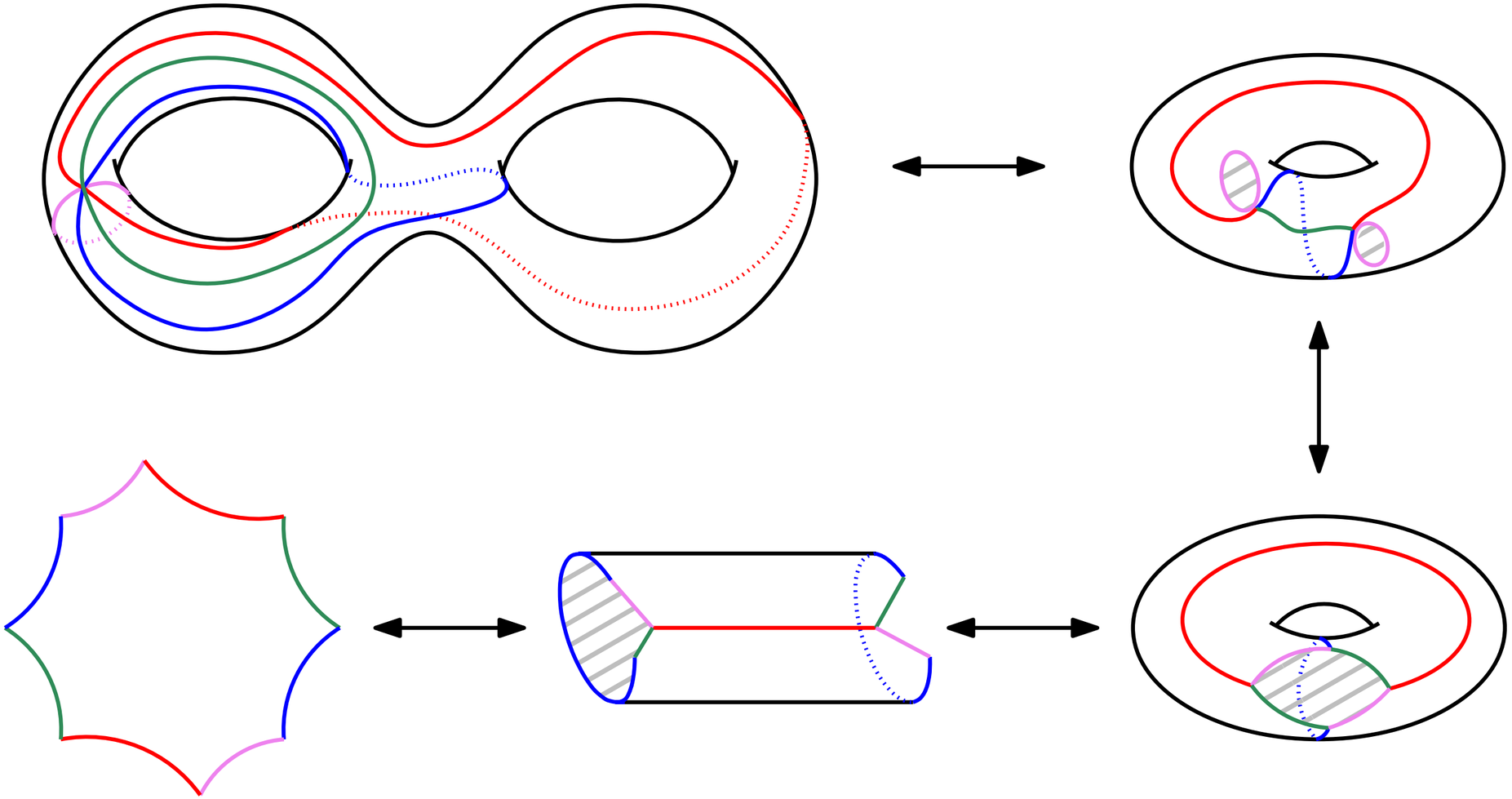
hyperbolic surface: a surface endowed with a Riemannian metric of constant negative curvature.

Bolza surface: a hyperbolic surface with the simplest possible topology (genus 2).

Unfolding the Bolza surface

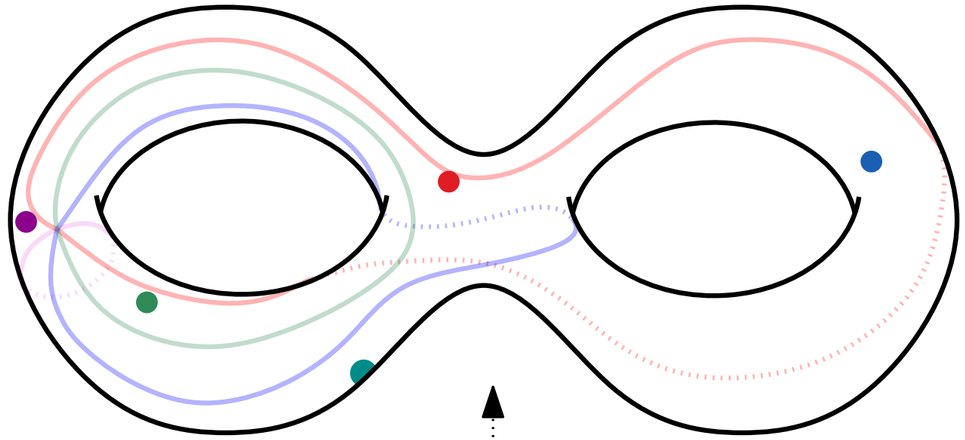


Unfolding the Bolza surface

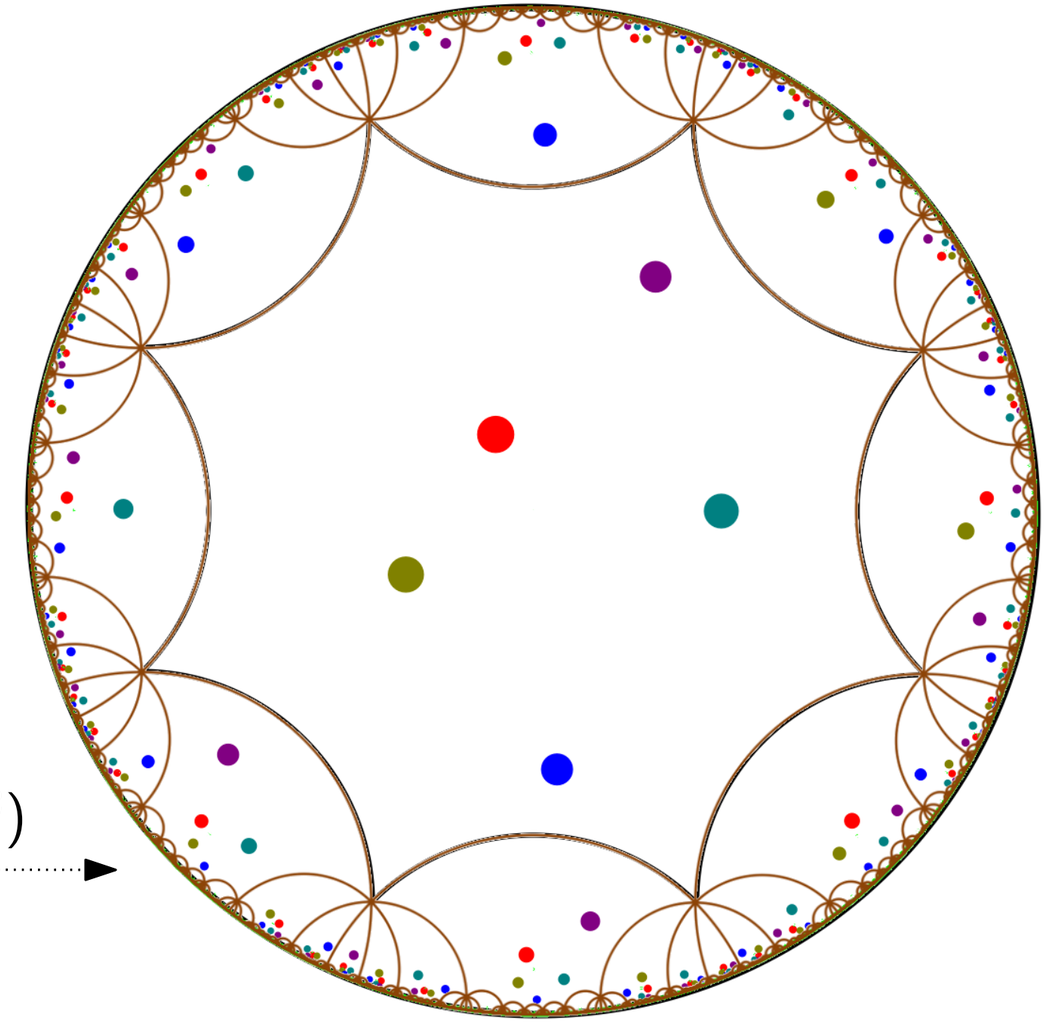


Fundamental polygon

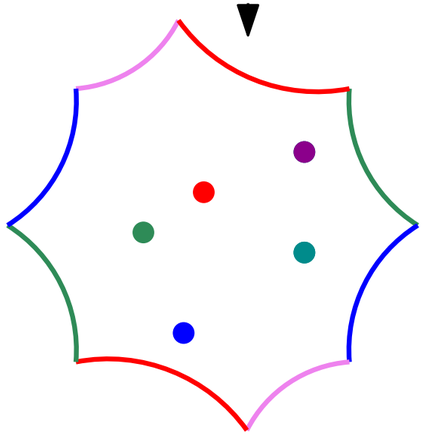
Unfolding the Bolza surface



Universal cover



"Unfolding" the surface
(Fundamental polygon)



Hyperbolic plane ($\mathbb{H}^2$)

tesselation

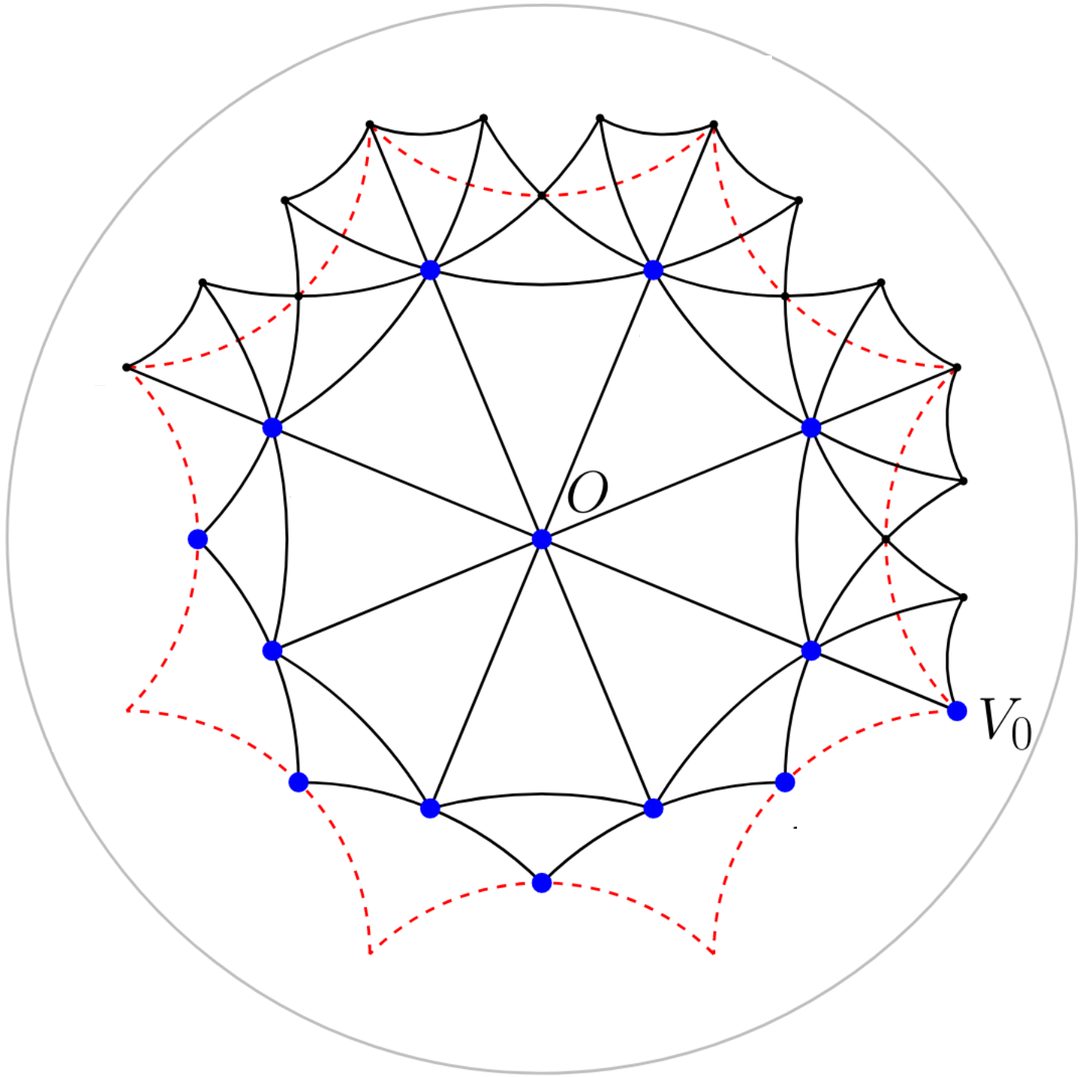
BW-algorithm on the Bolza surface

Results in [IT17]:

Add 14 dummy points such that $sys(M) > 2\delta$ where δ is the radius of the largest disks that do not contain any dummy point.

Use

- Knowledge of the systole
- A good fondamental domain
- Symetries of the surfaces



Summary of BW-algorithm on surface

Keys ideas:

- Initialization by adding points
- Use the universal cover -plane or $\mathbb{H}^2$-

For Torus and Bolza we used:

- Knowledge of the systole
- A good fondamental domain
- Symetries of the surfaces

For any surface S of genus ≥ 2:

- Initialization by adding points
- Use the universal cover $\mathbb{H}^2$

Adapted settings:

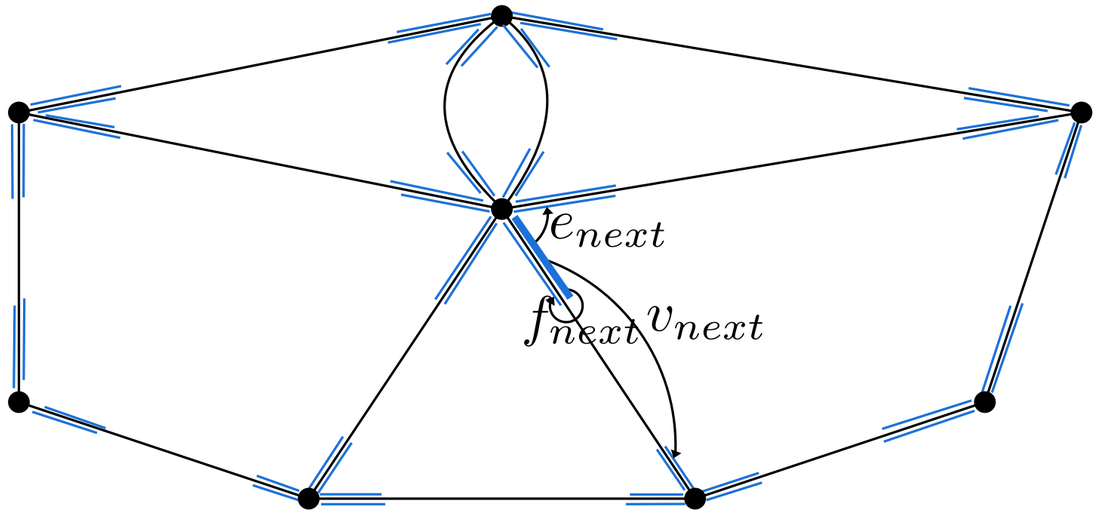
- Decompose S in "thin" and "thick" parts
- Compute a Dirichlet domain
- Use points satisfying a "density" property.

- ▷ Delaunay triangulation in Euclidean plane
- ▷ BW-algorithm on a torus ($g = 1$)
- ▷ Hyperbolic plane
- ▷ BW-algorithm on the Bolza surface ($g = 2$)
- ▶ BW-algorithm on the Bolza surface ($g = 2$)

Data structure

A Flag is a triple (v, e, f) (a vertex, an edge, a face)

Combinatorial map = set of flags + 3 involutions $(v_{next}, e_{next}, f_{next})$



- v_{next} change only the vertex incidence.
- e_{next} change only the edge incidence.
- f_{next} change only the face incidence.

Data structure

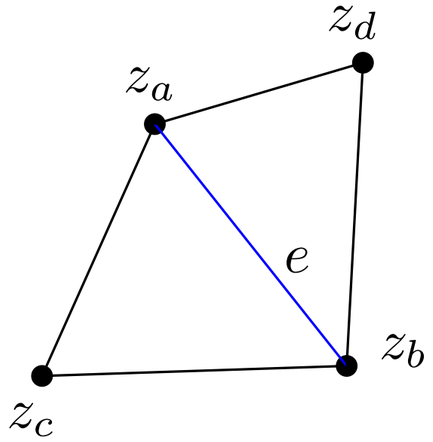
A Flag is a triple (v, e, f) (a vertex, an edge, a face)

Combinatorial map = set of flags + 3 involutions $(v_{next}, e_{next}, f_{next})$

To encode the geometry, we attach at each edge e a complex number ζ_e.to reconstruct the triangles.

1) Fix a triangle $z_a z_c z_b$.

2) Glue the triangle $z_a z_b z_d$ such that $\frac{(z_c - z_a)(z_d - z_b)}{(z_c - z_b)(z_d - z_a)} = \zeta_{[z_a, z_b]}$.



Data structure

A Flag is a triple (v, e, f) (a vertex, an edge, a face)

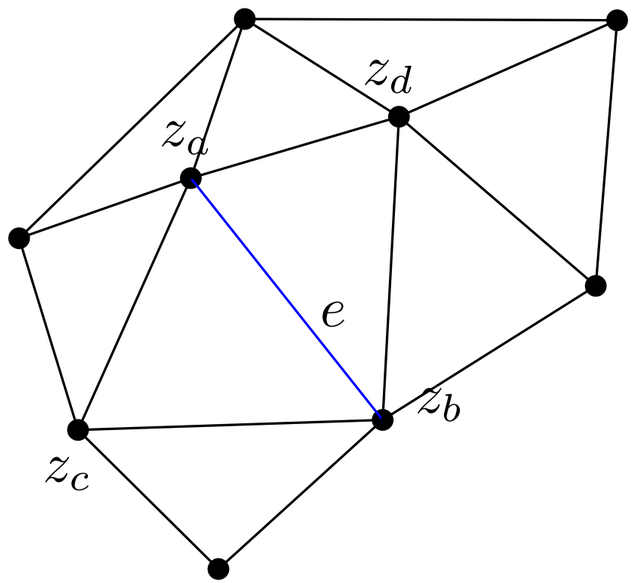
Combinatorial map = set of flags + 3 involutions $(v_{next}, e_{next}, f_{next})$

To encode the geometry, we attach at each edge e a complex number ζ_e.to reconstruct the triangles.

1) Fix a triangle $z_a z_c z_b$.

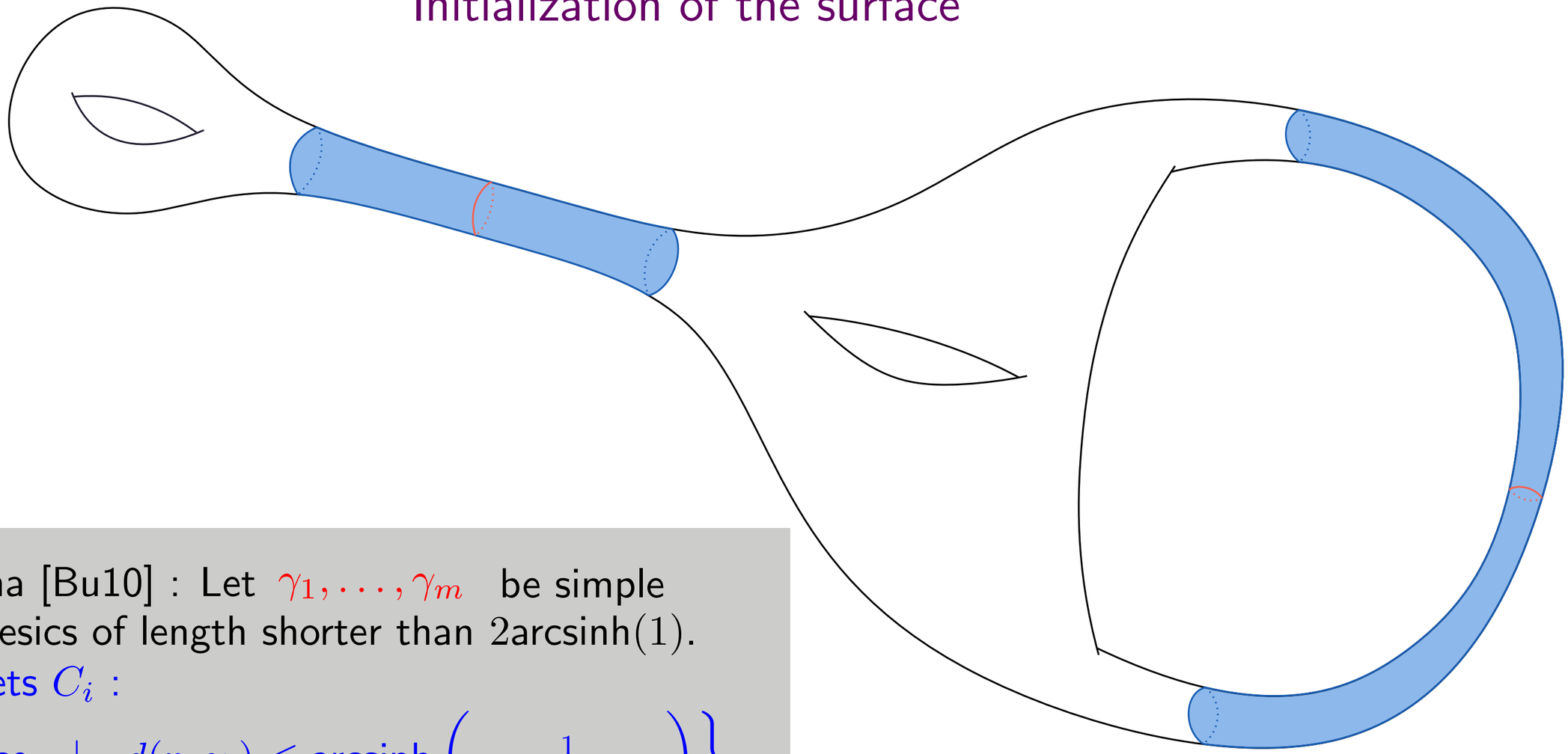
2) Glue the triangle $z_a z_b z_d$ such that $\frac{(z_c - z_a)(z_d - z_b)}{(z_c - z_b)(z_d - z_a)} = \zeta_{[z_a, z_b]}$.

3) And so on...



Initialization of the surface

Initialization of the surface

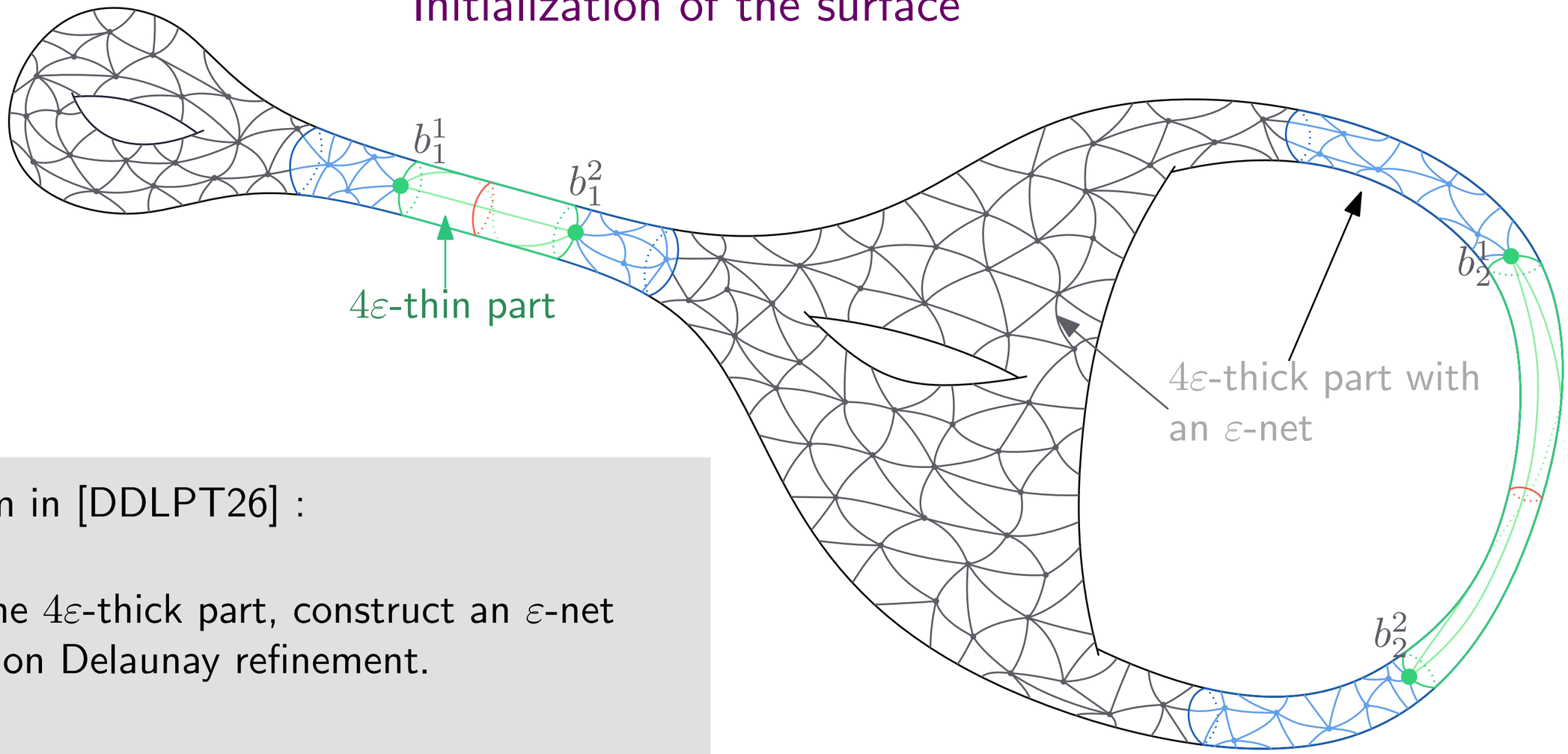


Collar lemma [Bu10] : Let $\gamma_1, \dots, \gamma_m$ be simple closed geodesics of length shorter than $2\text{arcsinh}(1)$. Then the sets C_i :

$$\left\{ p \in \text{Surface} \quad | \quad d(p, \gamma_i) \leq \text{arcsinh}\left(\frac{1}{\sinh\left(\frac{1}{2} l(\gamma_i)\right)} \right) \right\}$$

are pairwise disjoint and homeomorphic to a cylinder.

Initialization of the surface

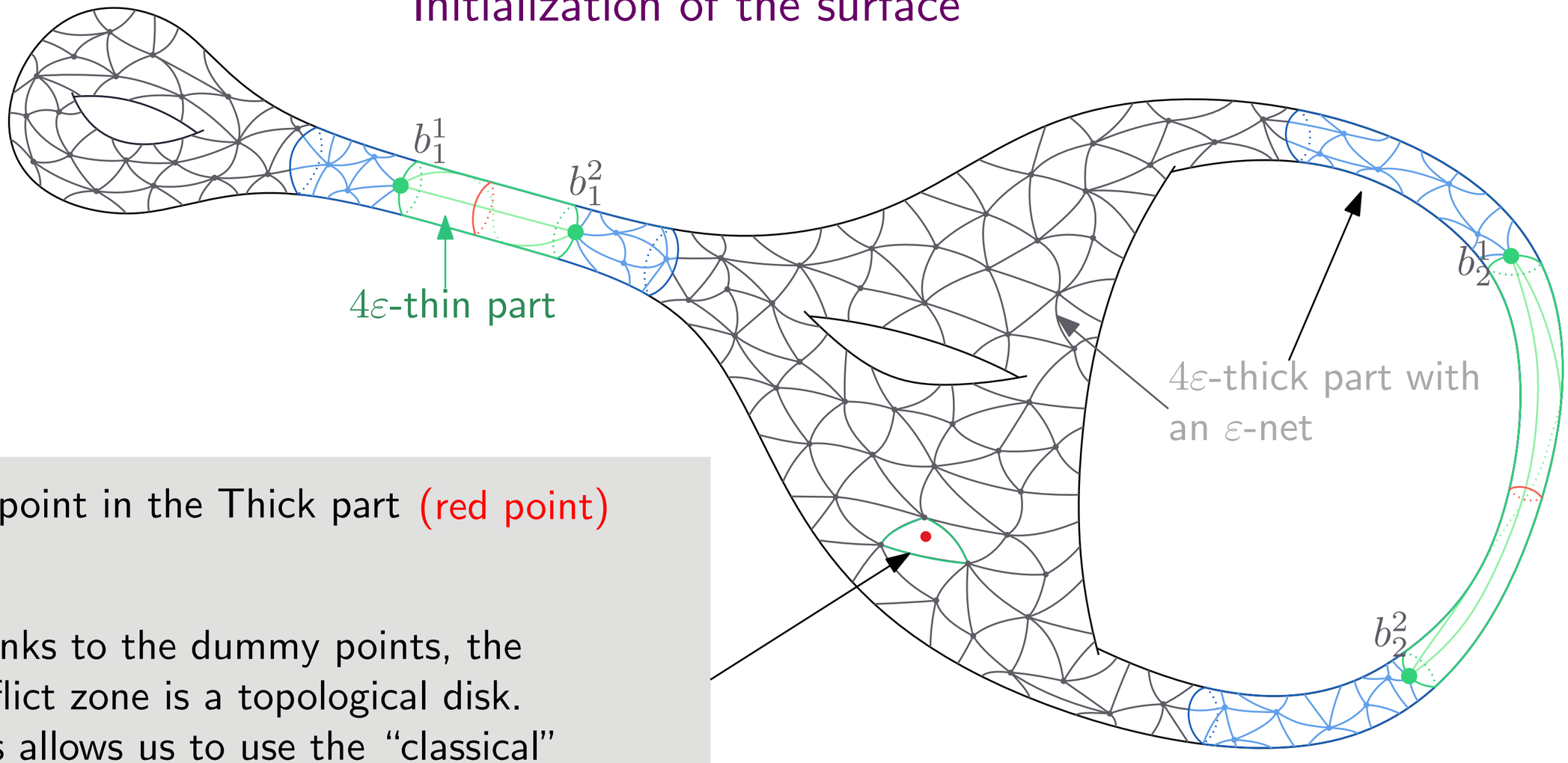


Algorithm in [DDLPT26] :

- In the 4ε-thick part, construct an ε-net based on Delaunay refinement.

- Detect the collars (4ε-thin parts) and place points $b_{1,2}^i$ on their borders.

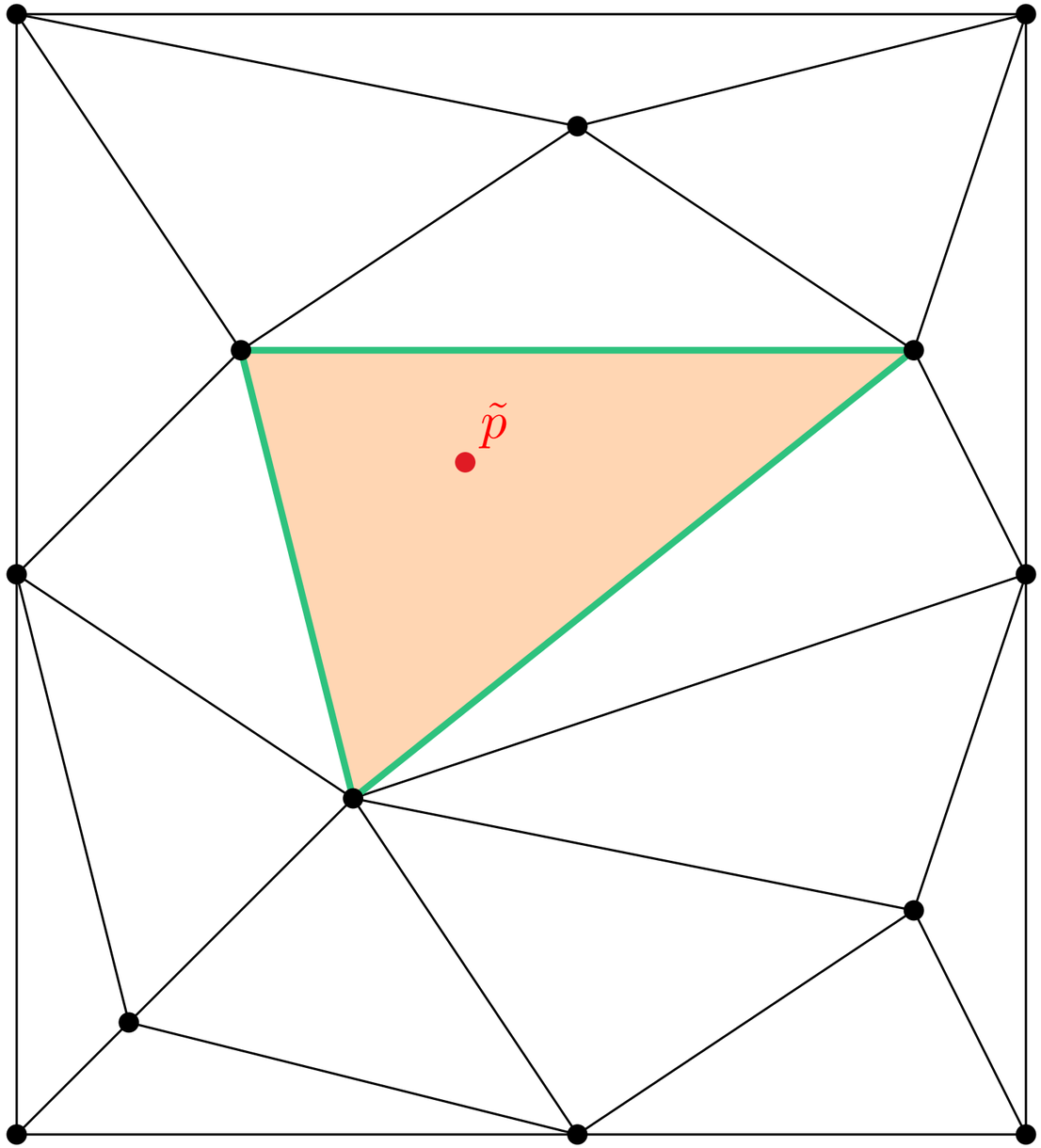
Initialization of the surface



Insert a point in the Thick part (red point)

Thanks to the dummy points, the conflict zone is a topological disk. This allows us to use the "classical" BW-algorithm.

Insertion in 4ε-thick part

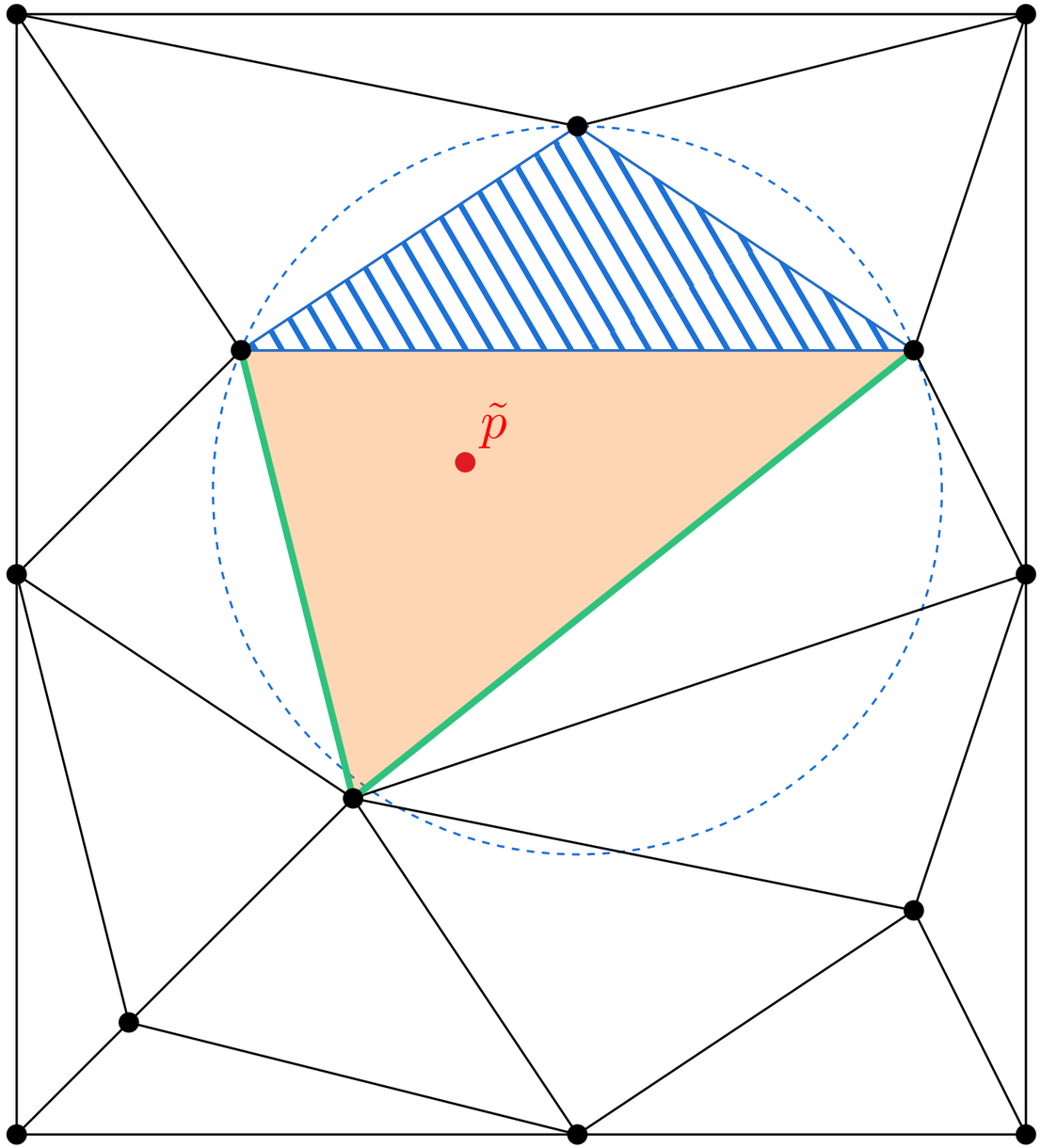


Processing in the universal cover

→ Find a triangle which contains $\widetilde{p}$.

→ Start a DFS to find conflict zone.

Insertion in 4ε-thick part

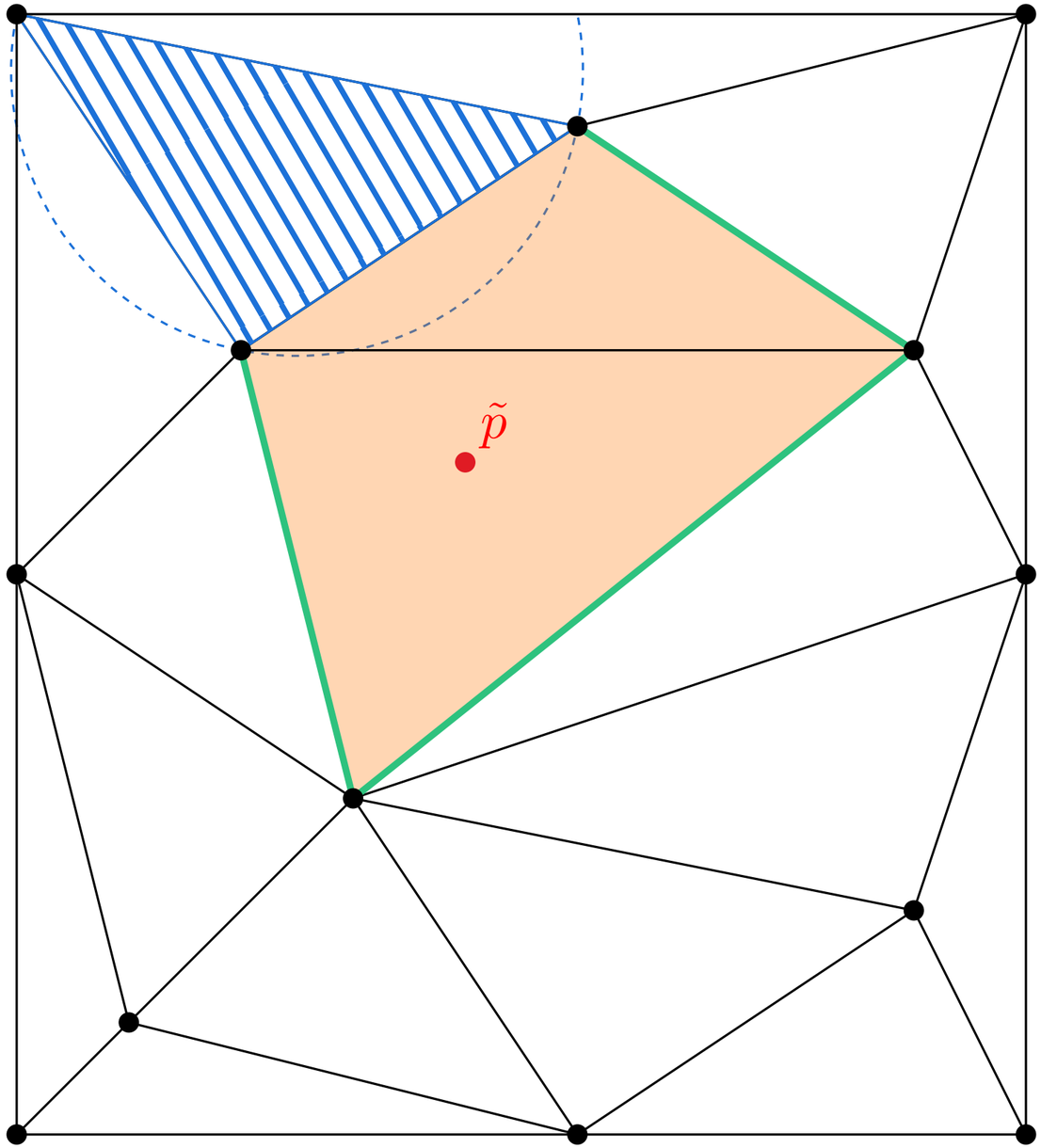


Processing in the universal cover

$\longrightarrow$ Conflict with $\widetilde{p}$.

$\longrightarrow$ Push the two other sides in the stack.

Insertion in 4ε-thick part

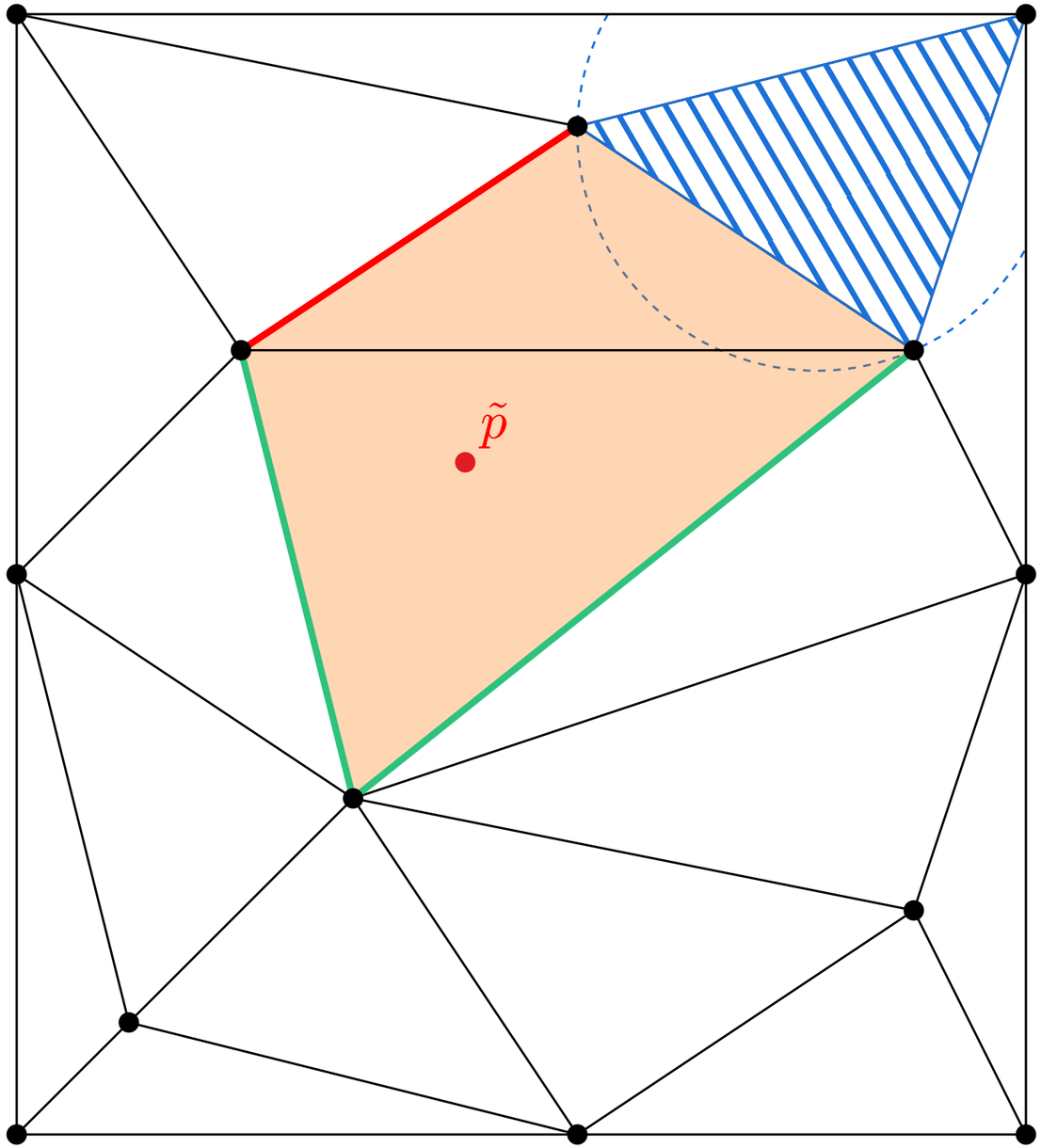


Processing in the universal cover

→ Not in conflict with $\widetilde{p}$.

→ The side in commun with the conflict zone belongs to the border.

Insertion in 4ε-thick part

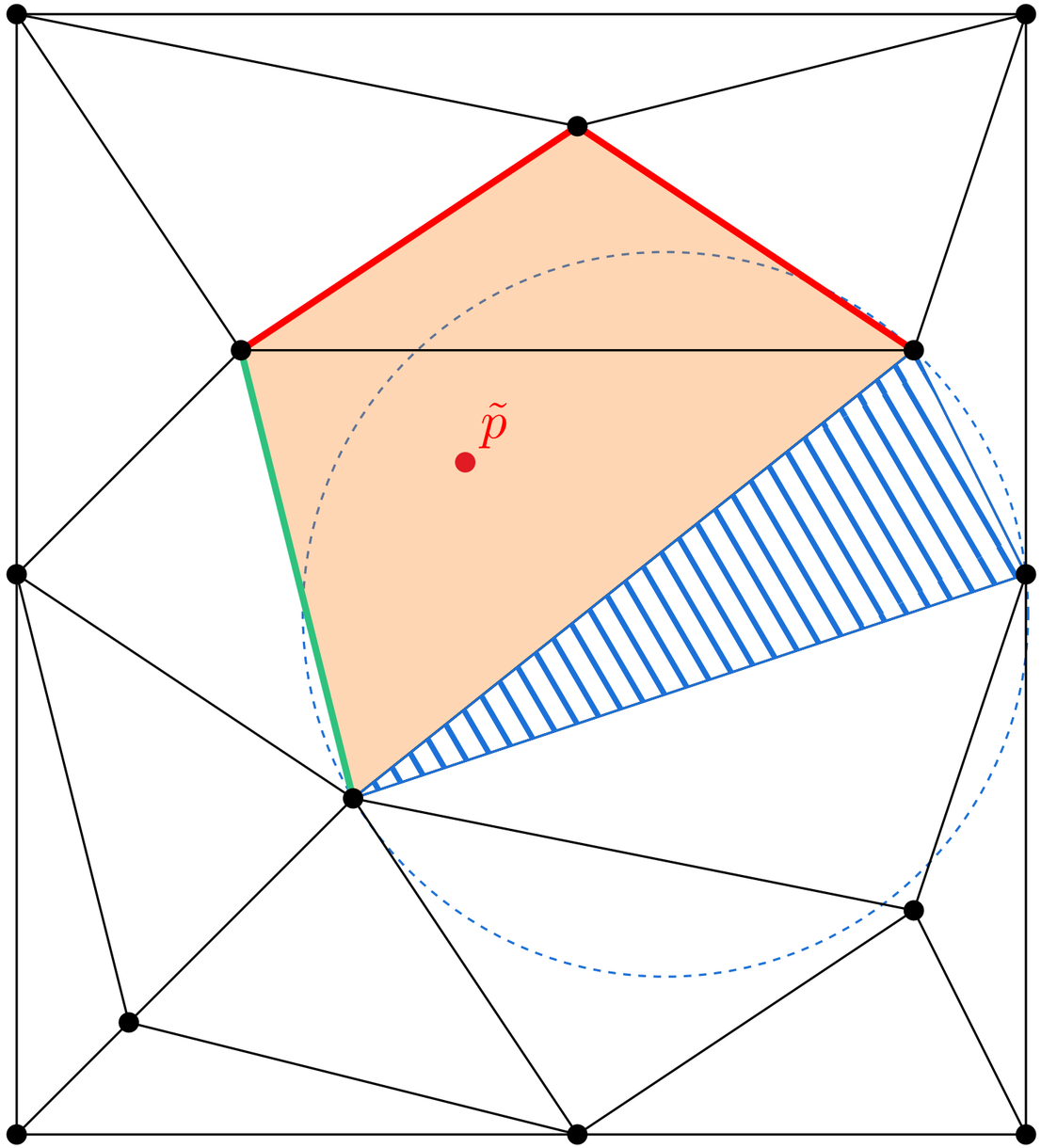


Processing in the universal cover

⟶ Not in conflict with $\widetilde{p}$.

⟶ The side in commun with the conflict zone belongs to the border.

Insertion in 4ε-thick part

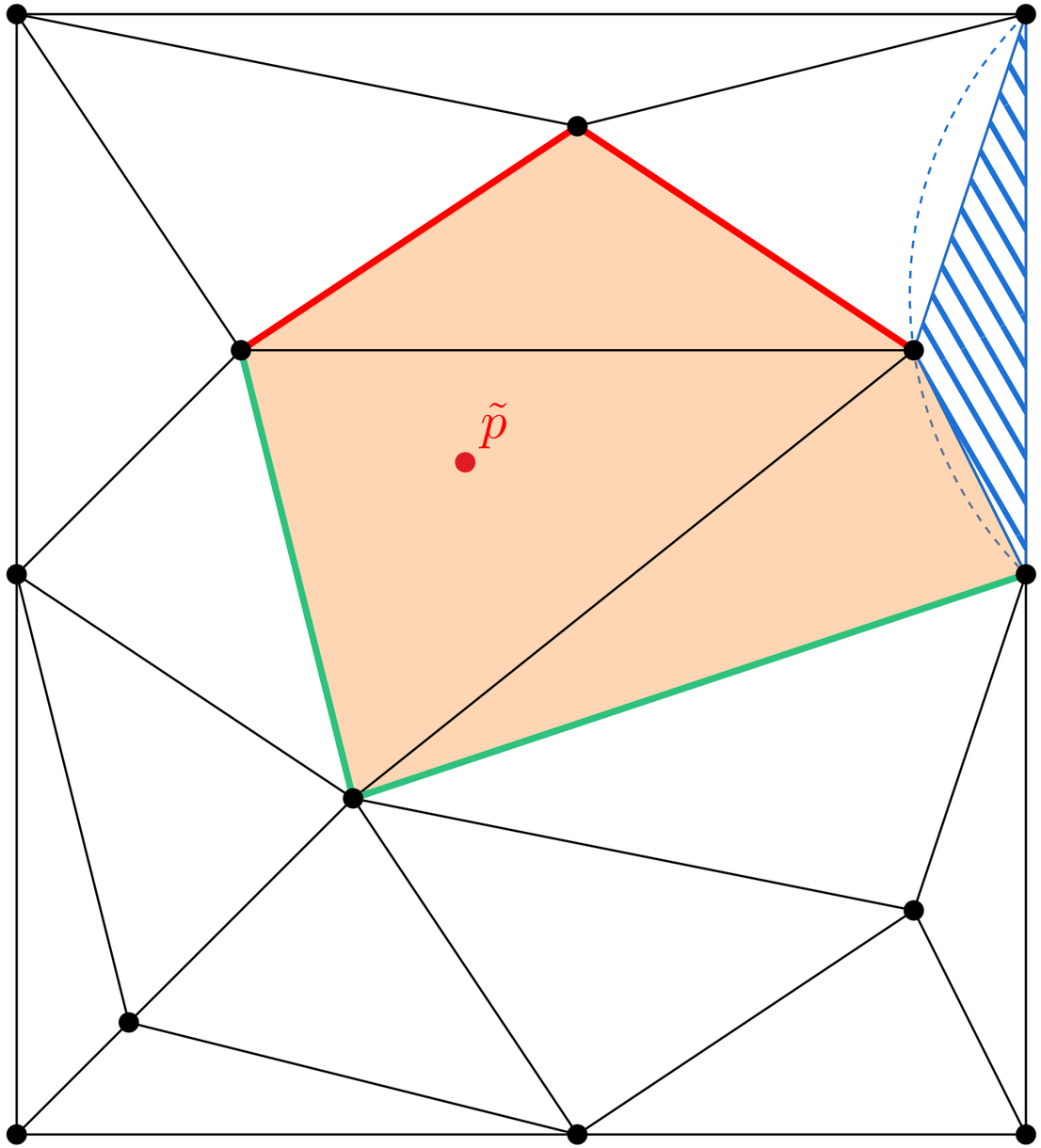


Processing in the universal cover

$\longrightarrow$ Conflict with $\widetilde{p}$.

$\longrightarrow$ Push the two other sides in the stack.

Insertion in 4ε-thick part

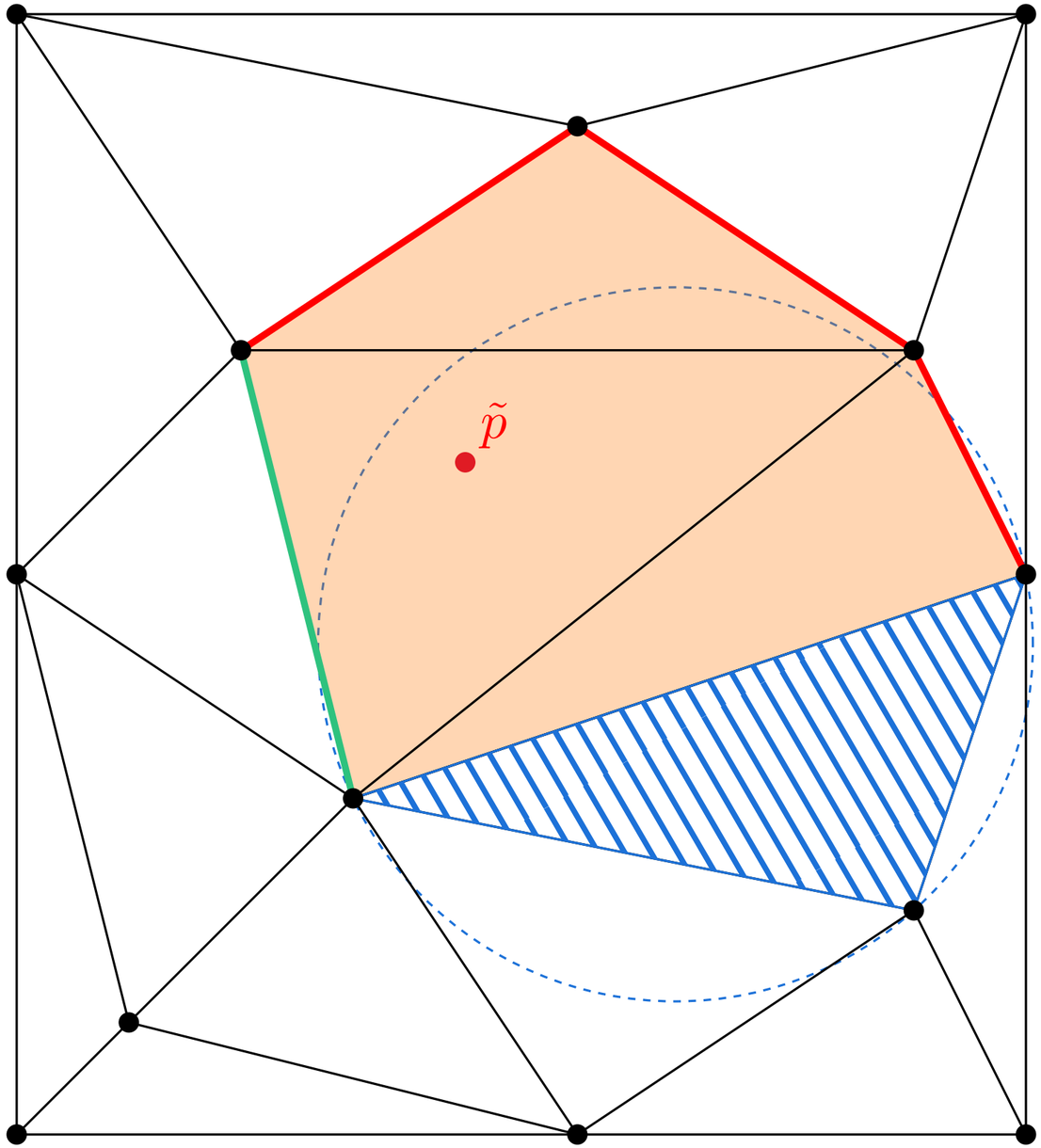


Processing in the universal cover

⟶ Not in conflict with $\widetilde{p}$.

⟶ The side in commun with the conflict zone belongs to the border.

Insertion in 4ε-thick part

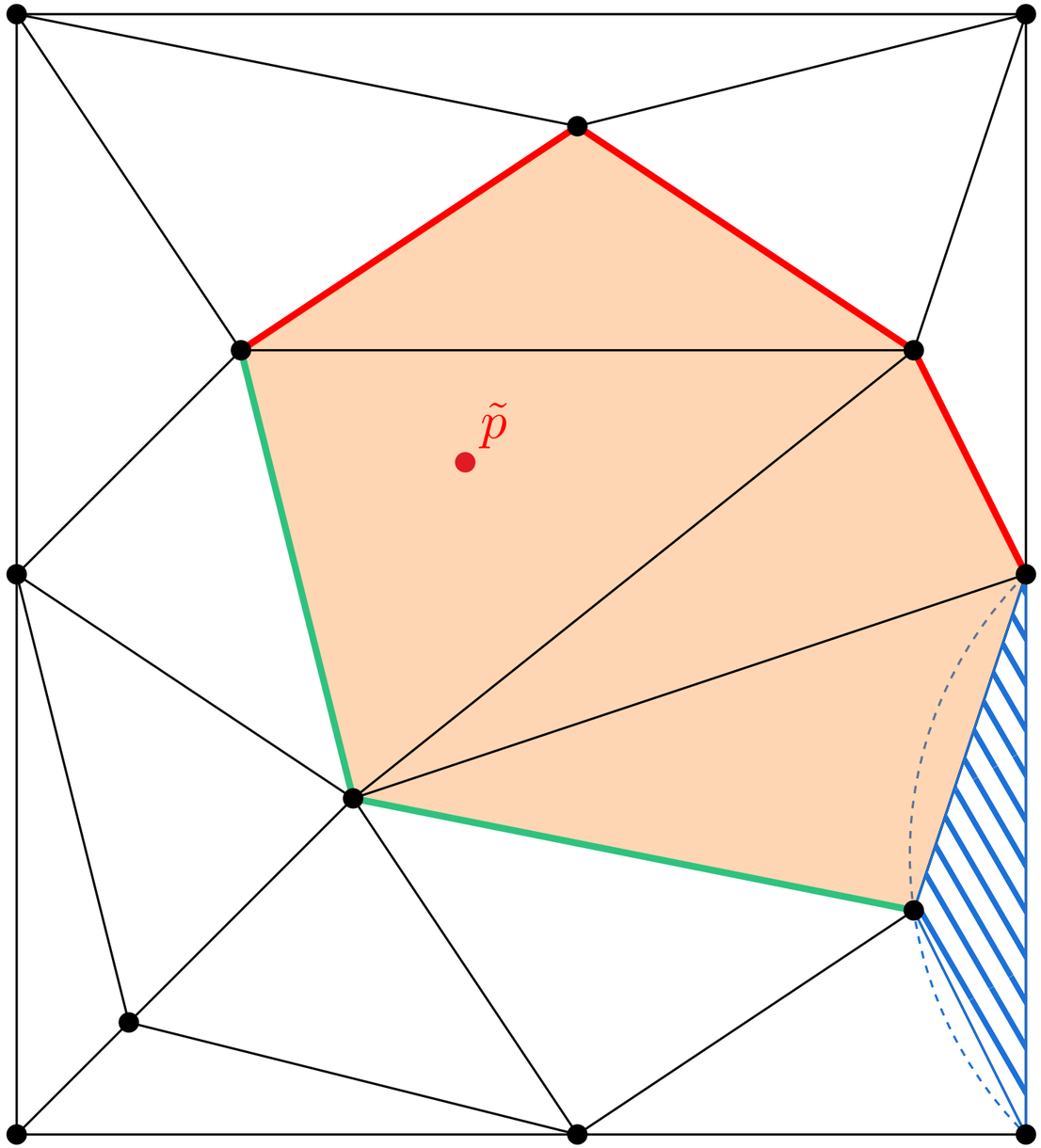


Processing in the universal cover

⟶ Conflict with $\widetilde{p}$.

⟶ Push the two other sides in the stack.

Insertion in 4ε-thick part

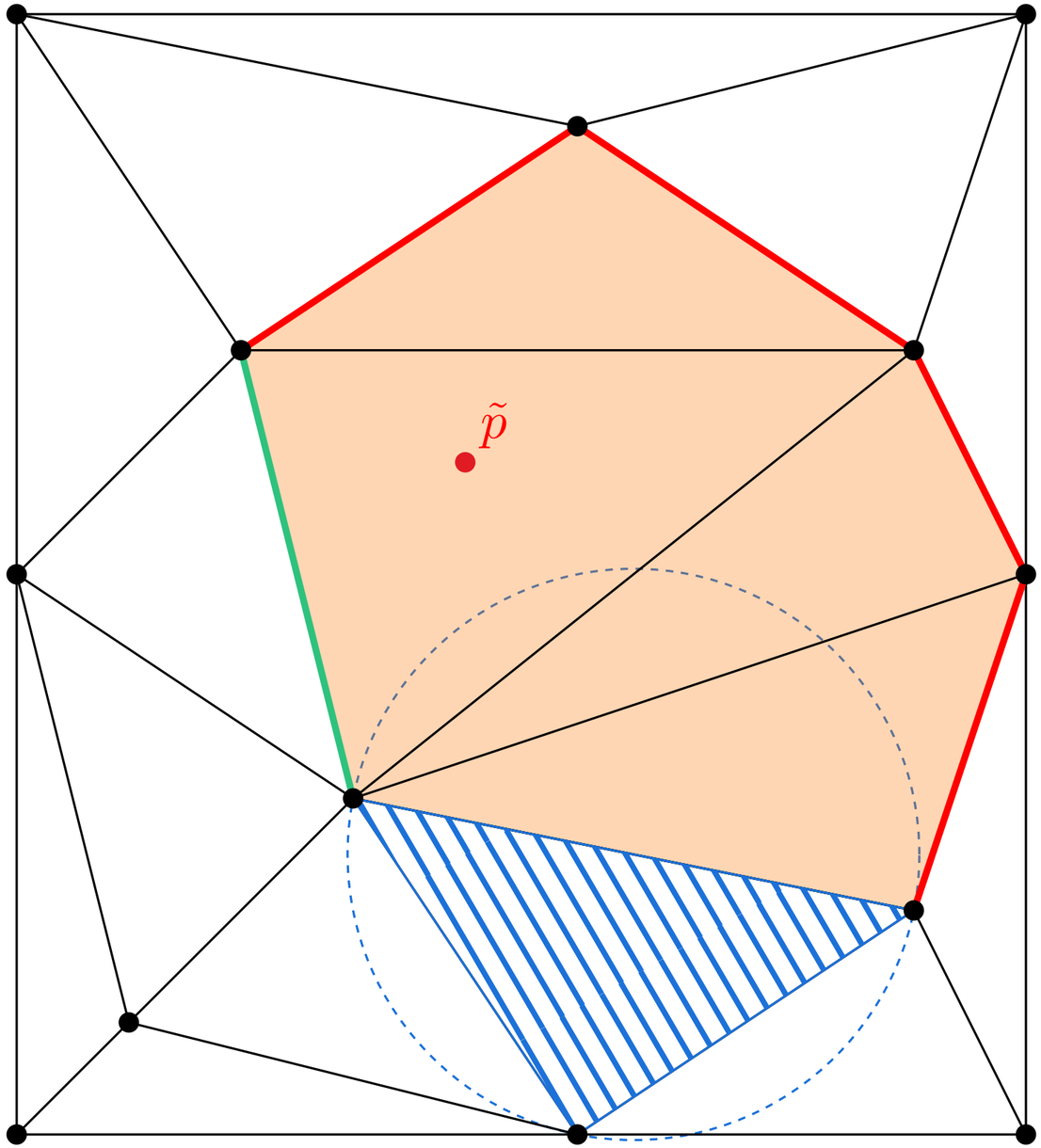


Processing in the universal cover

⟶ Not in conflict with $\widetilde{p}$.

⟶ The side in commun with the conflict zone belongs to the border.

Insertion in 4ε-thick part

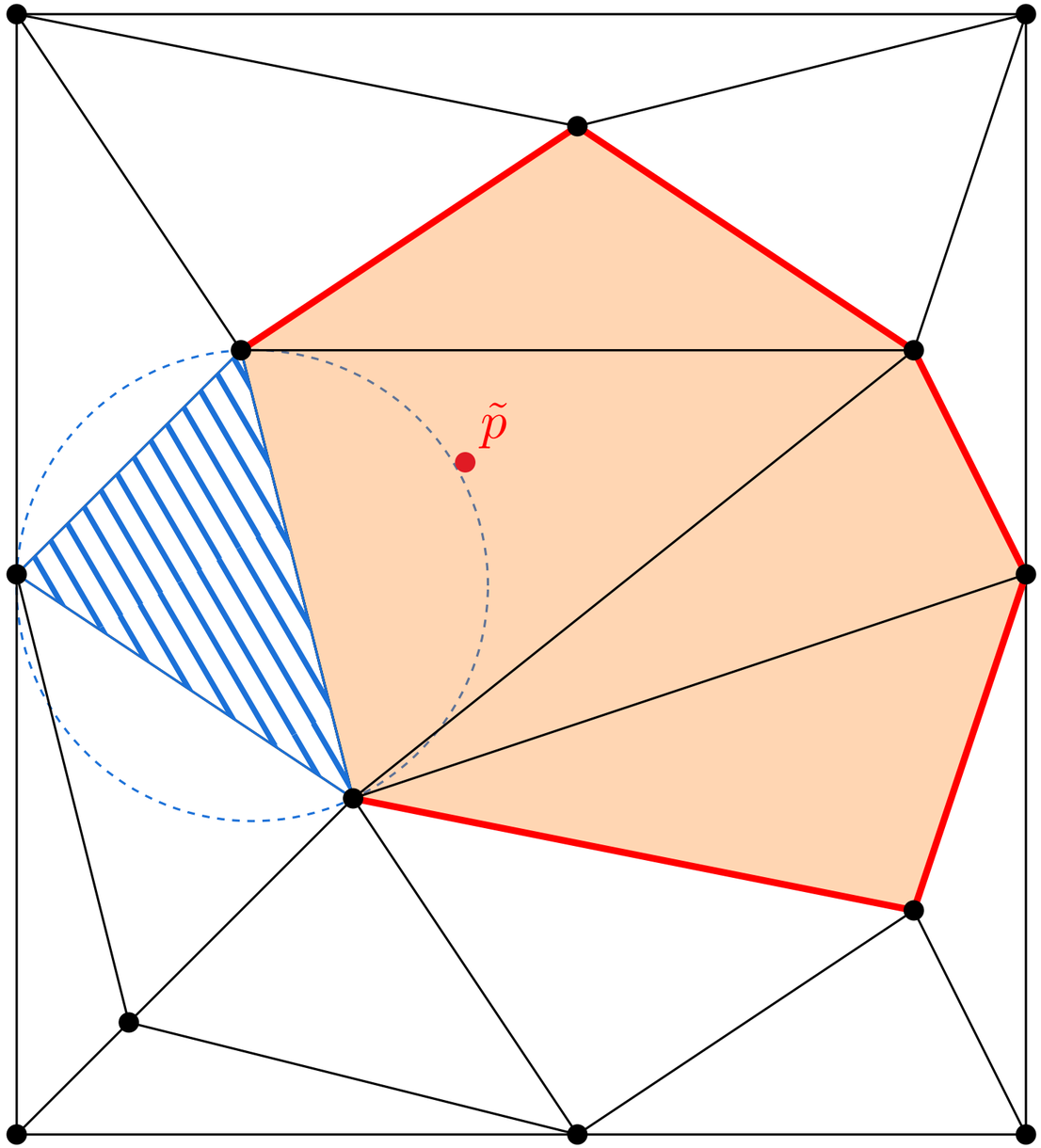


Processing in the universal cover

⟶ Not in conflict with $\widetilde{p}$.

⟶ The side in commun with the conflict zone belongs to the border.

Insertion in 4ε-thick part

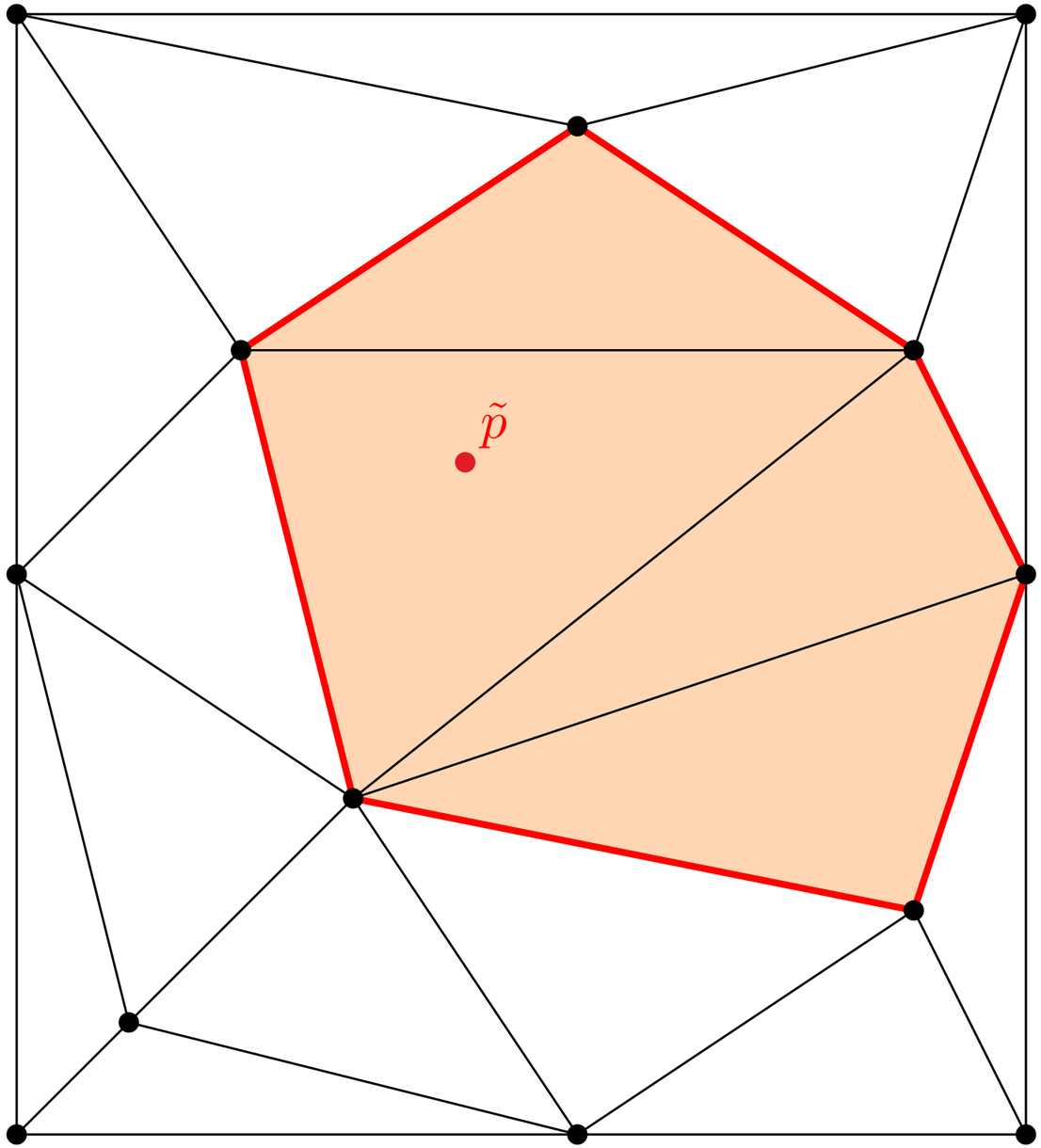


Processing in the universal cover

→ Not in conflict with $\widetilde{p}$.

→ The side in commun with the conflict zone belongs to the border.

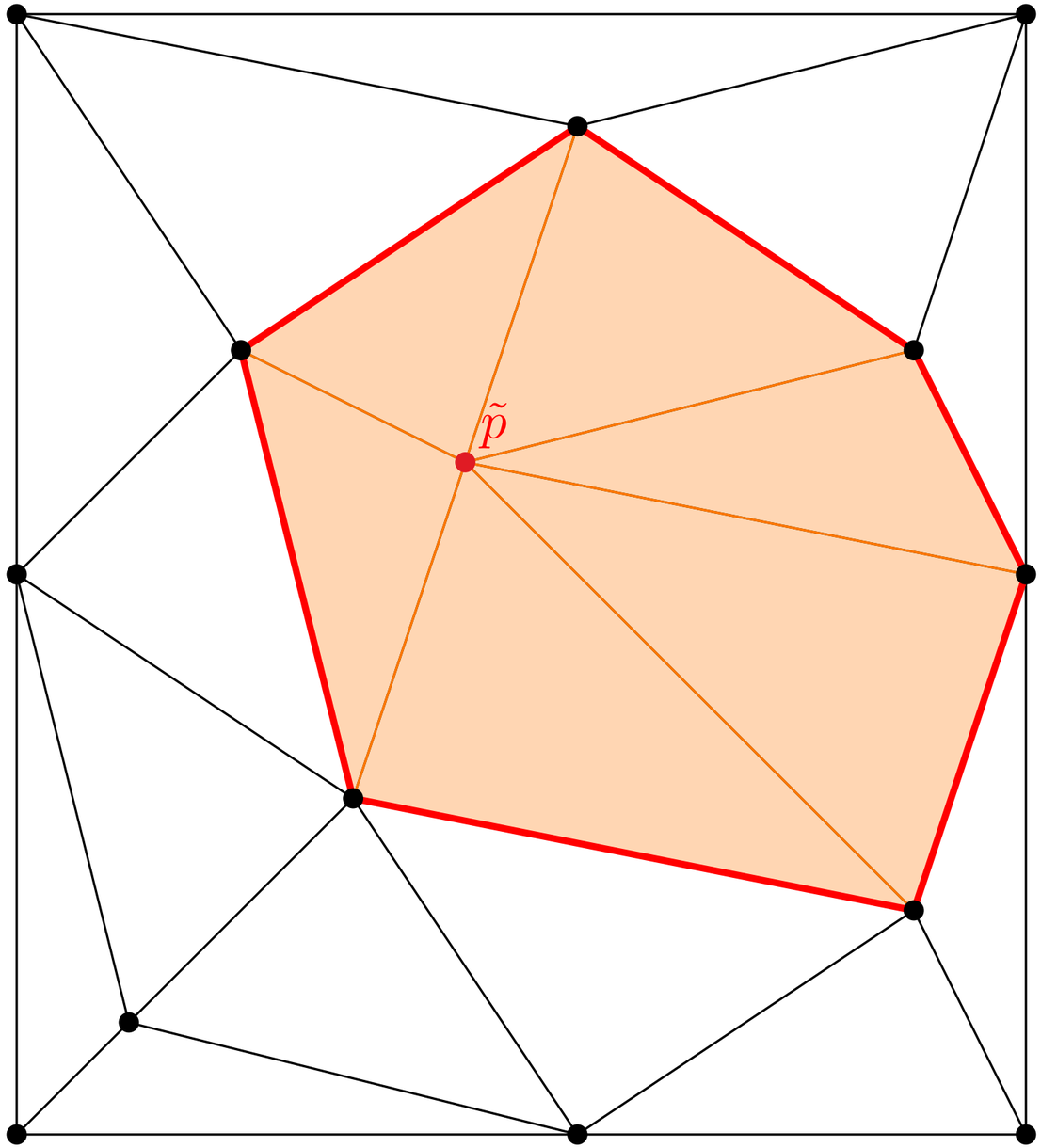
Insertion in 4ε-thick part



Processing in the universal cover

$\longrightarrow$ End of the BFS, we find the conflict zone.

Insertion in 4ε-thick part

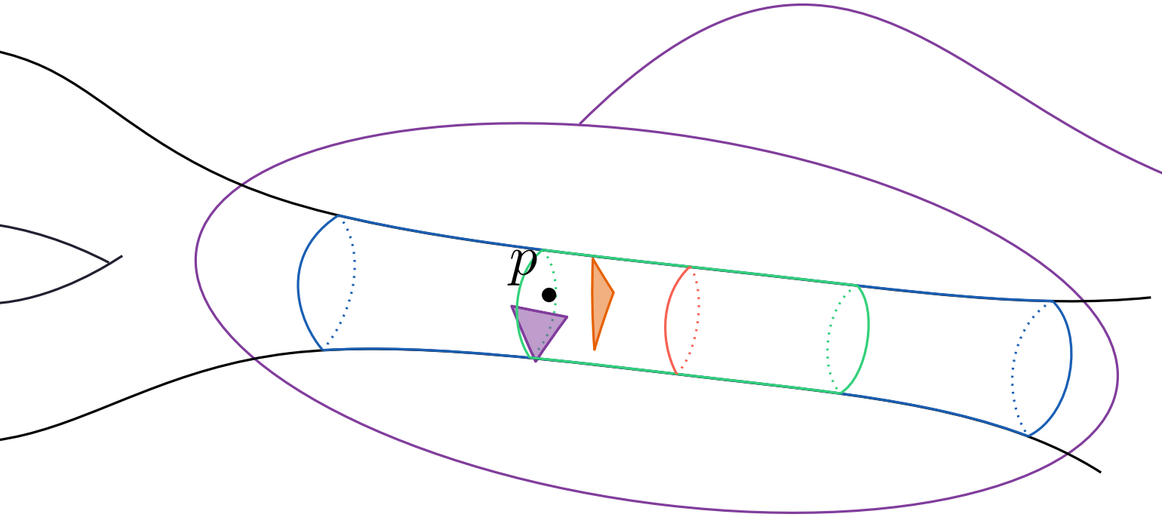


Processing in the universal cover

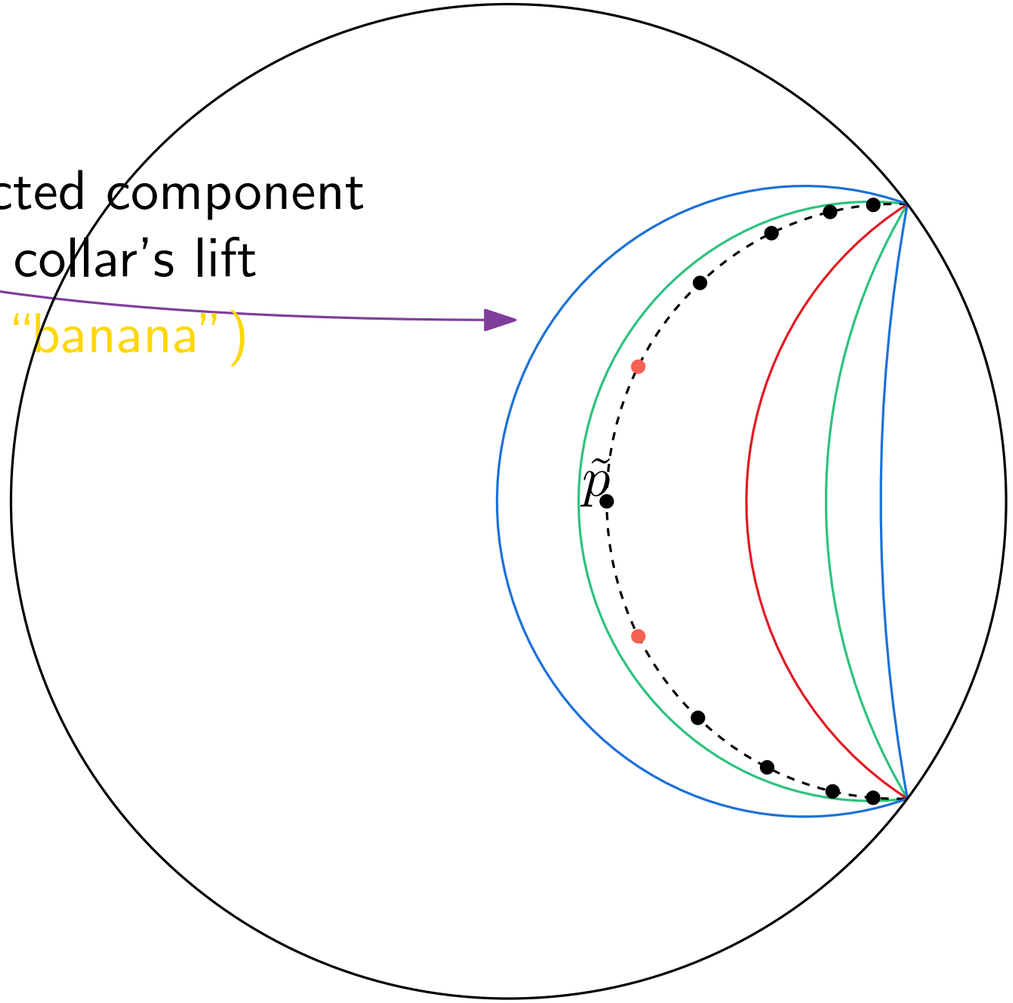
→ Remove all triangles in the conflict zone.

→ Connect $\widetilde{p}$ to the border's vertices.

4ε-thin part : lift of a collar

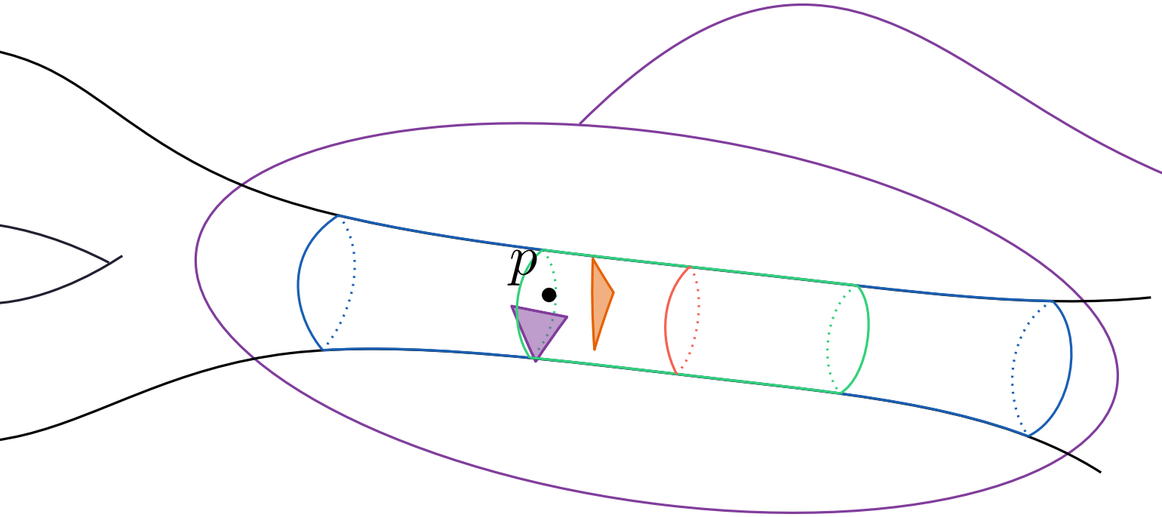


A connected component
of a collar's lift
(a "banana")

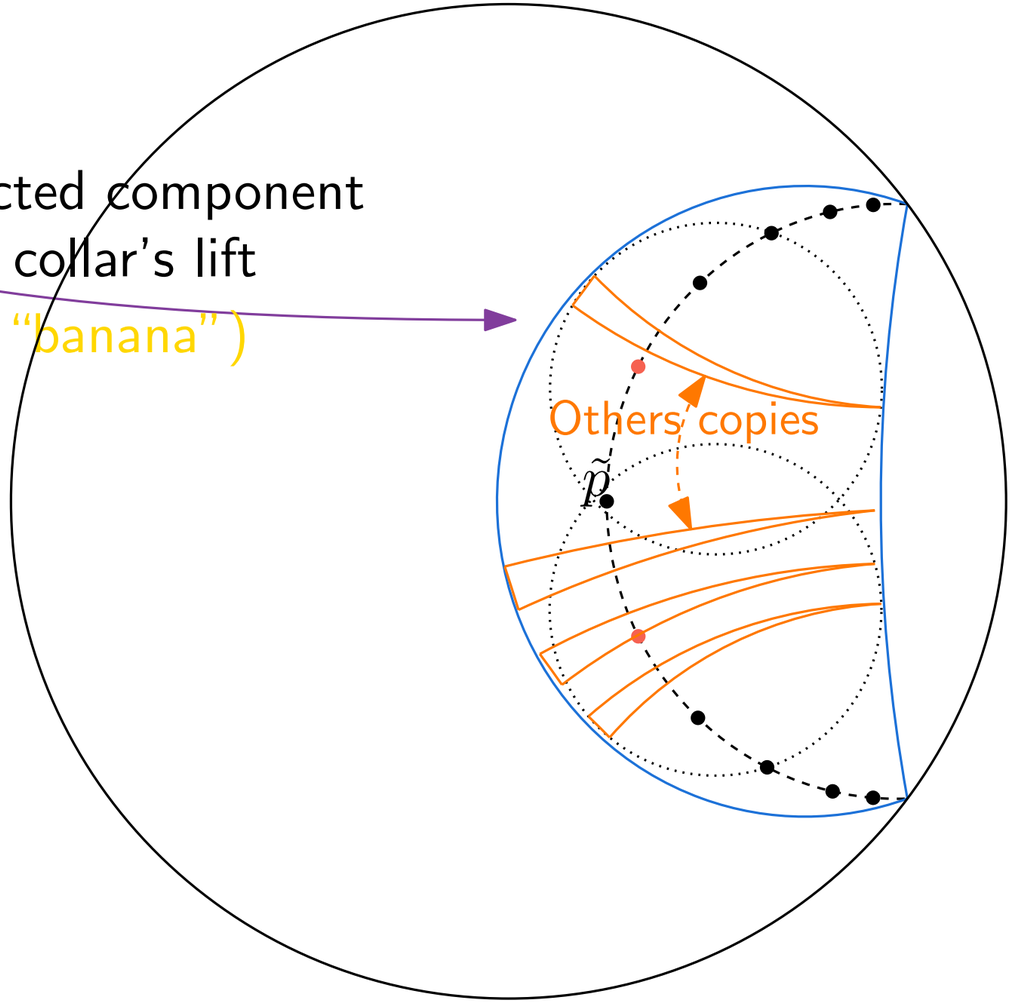


$\mathbb{H}^2$, universal cover

4ε-thin part : lift of a collar



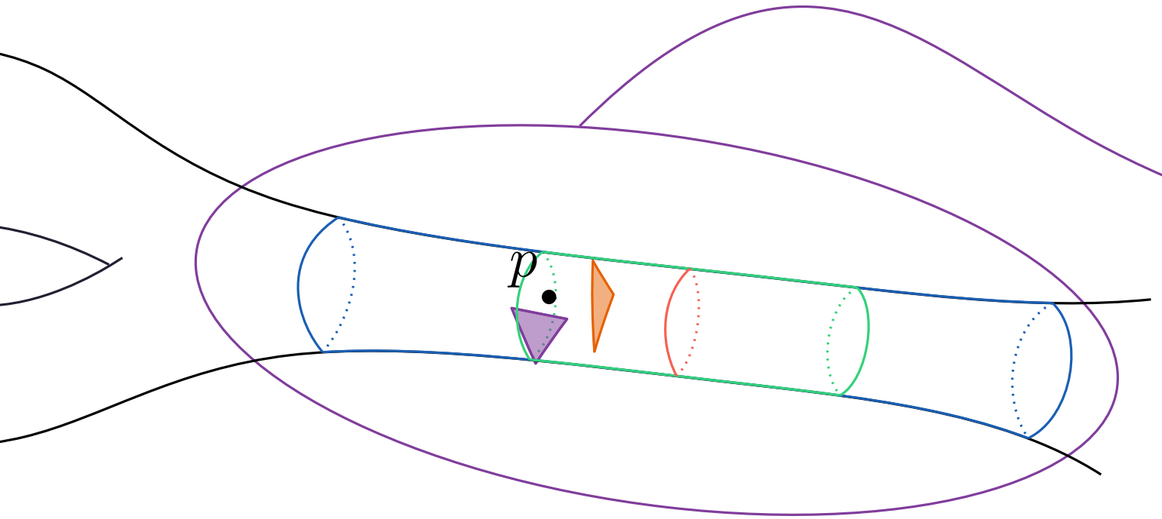
A connected component
of a collar's lift
(a "banana")



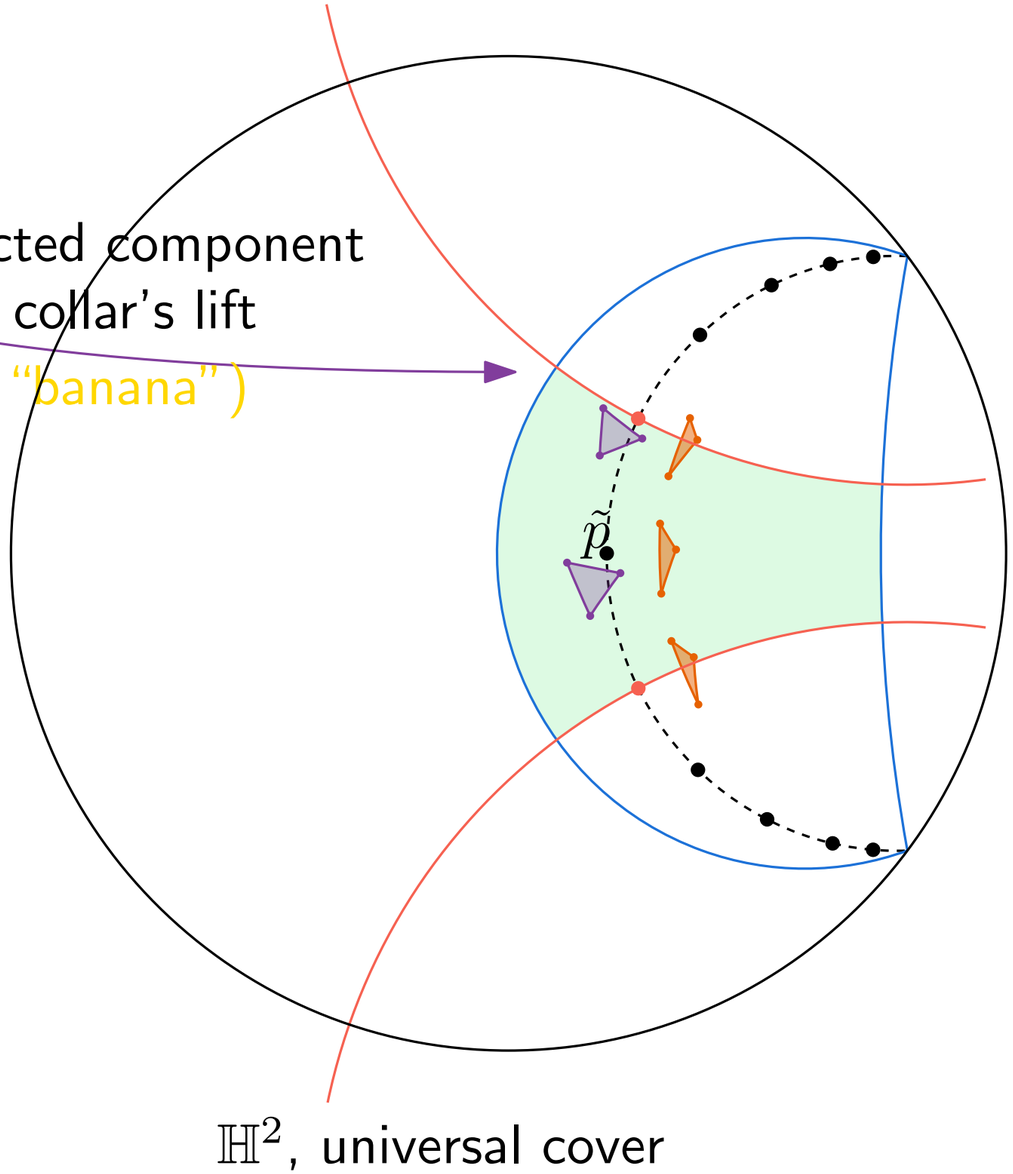
Prop : If p is in the 4ε-thin part,
then the conflict zone is included in the collar.

$\mathbb{H}^2$, universal cover

4ε-thin part : lift of a collar



A connected component
of a collar's lift
(a "banana")

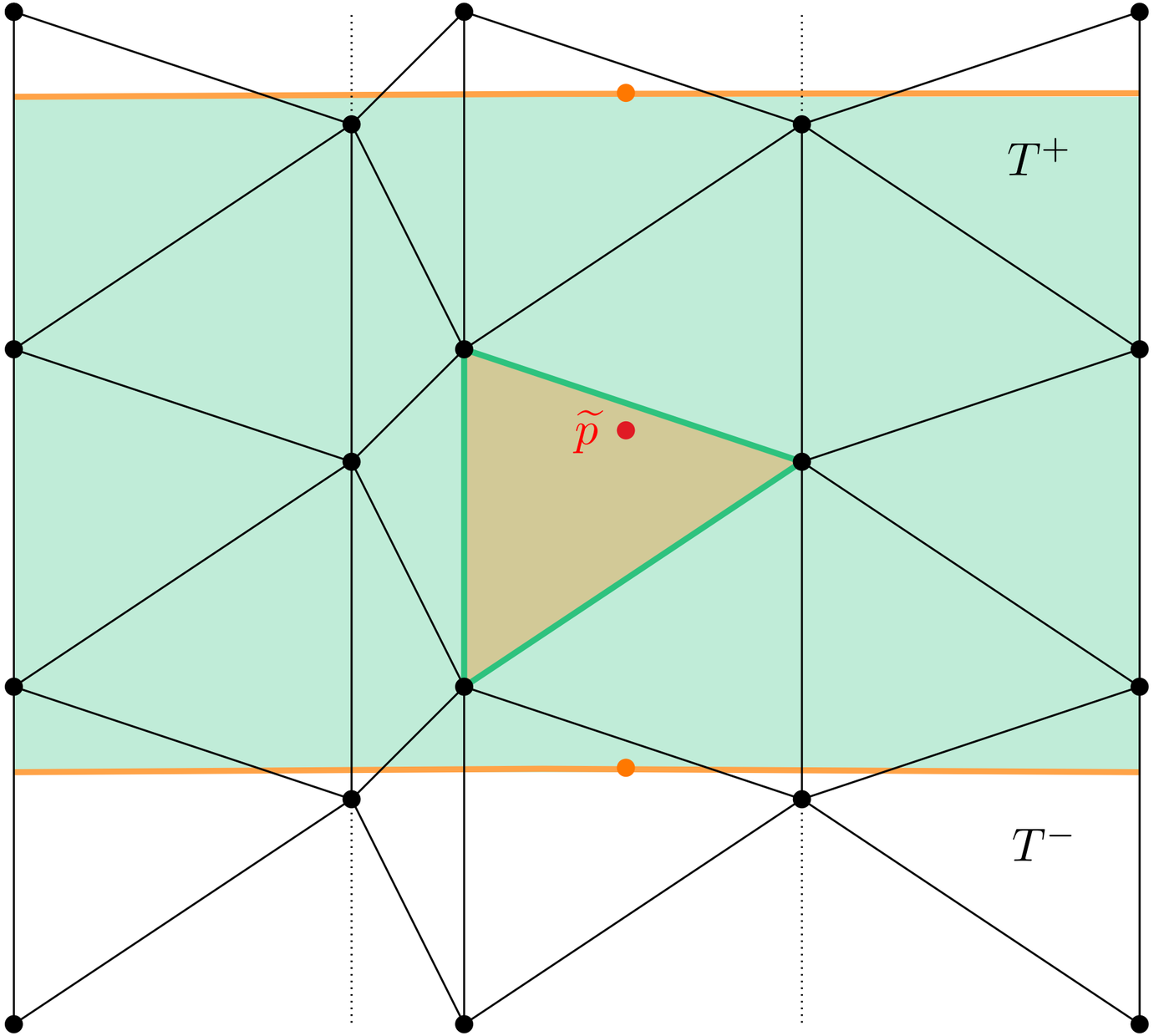


$\mathbb{H}^2$, universal cover

Prop : If p is in the 4ε-thin part,
then the conflict zone is included in the collar.

Restrict the conflict zone to triangles
having a vertex in the green region.

Insertion in the 4ε-thin part

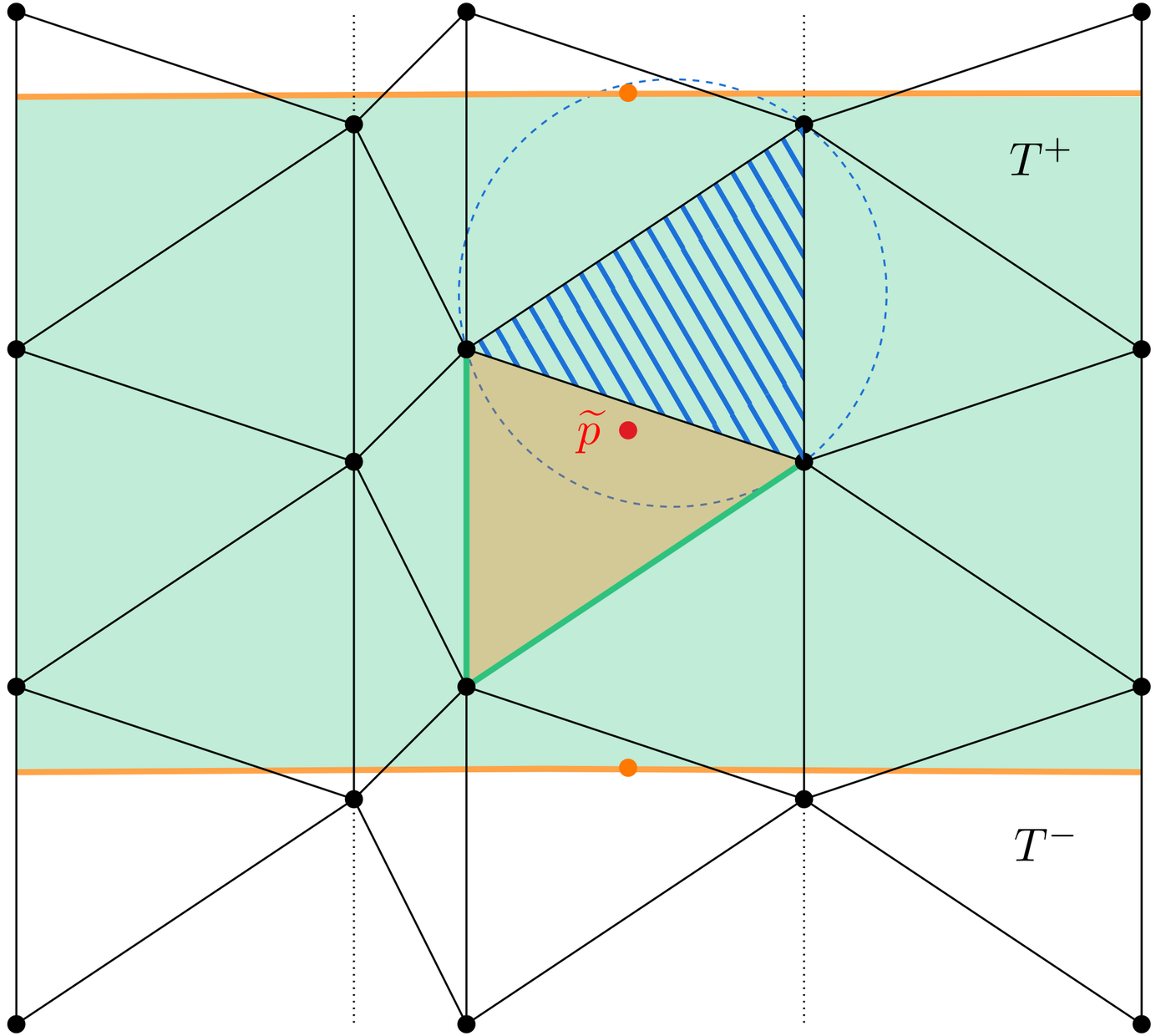


Processing in a banana

→ Find a triangle which contains $\widetilde{p}$.

→ Start a DFS to find conflict zone.

Insertion in the 4ε-thin part

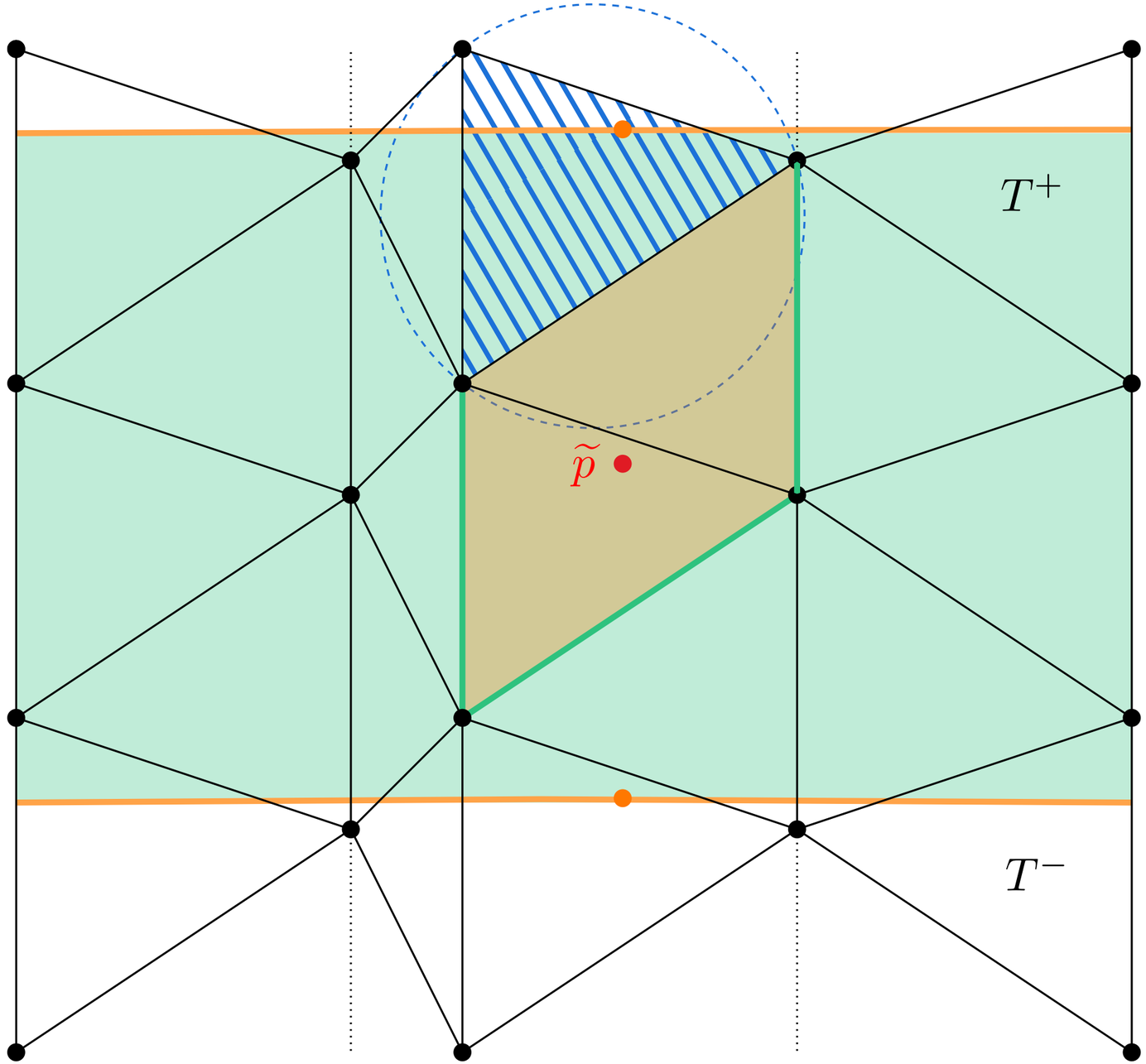


Processing in a banana

→ Conflict with $\widetilde{p}$ **and** the vertices are between the two lines.

→ Push the two other sides in the stack.

Insertion in the 4ε-thin part

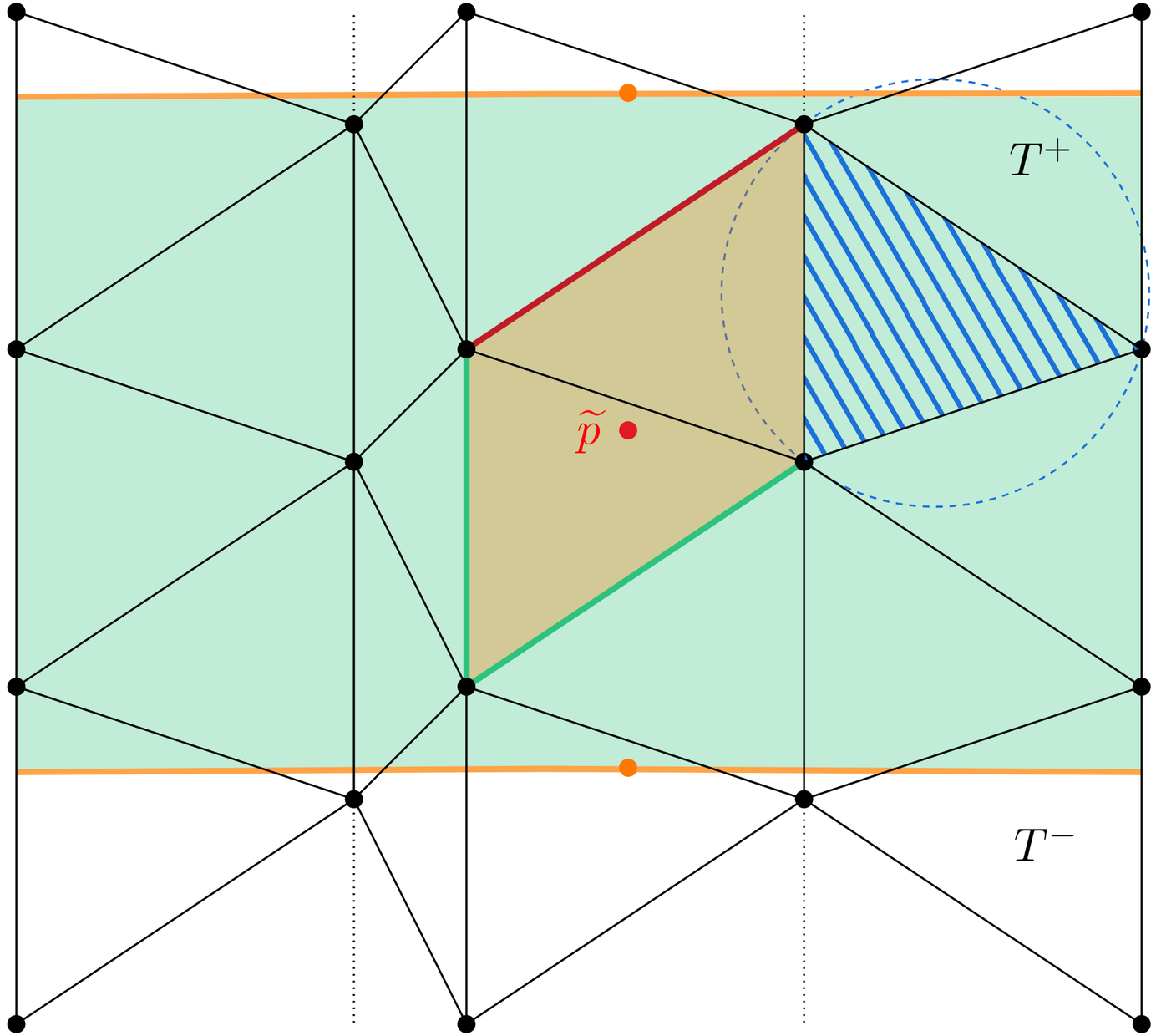


Processing in a banana

→ Not in conflict with $\widetilde{p}$.

→ The side in commun with the conflict zone belongs to the border.

Insertion in the 4ε-thin part

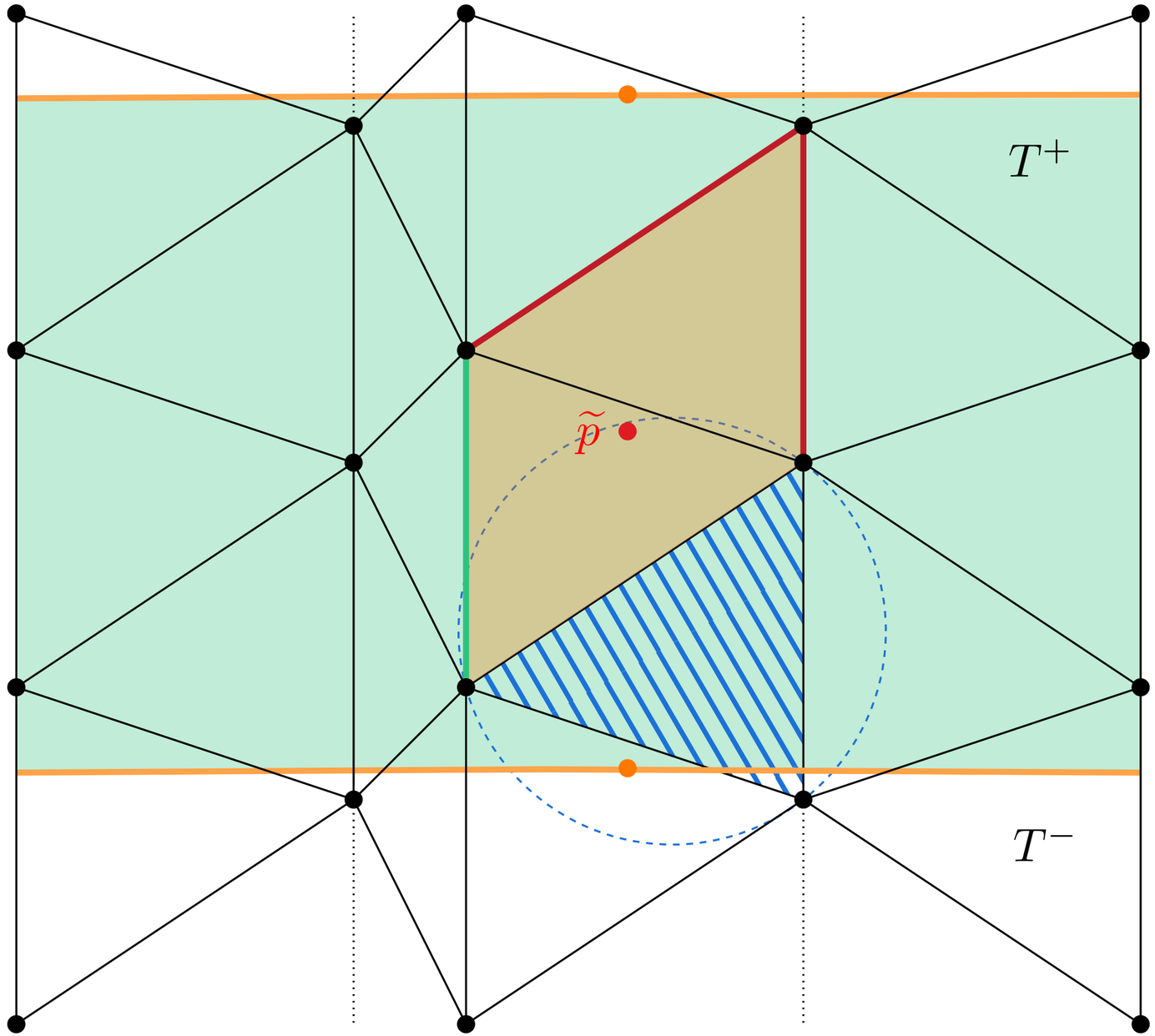


Processing in a banana

→ Not in conflict with $\widetilde{p}$.

→ The side in commun with the conflict zone belongs to the border.

Insertion in the 4ε-thin part

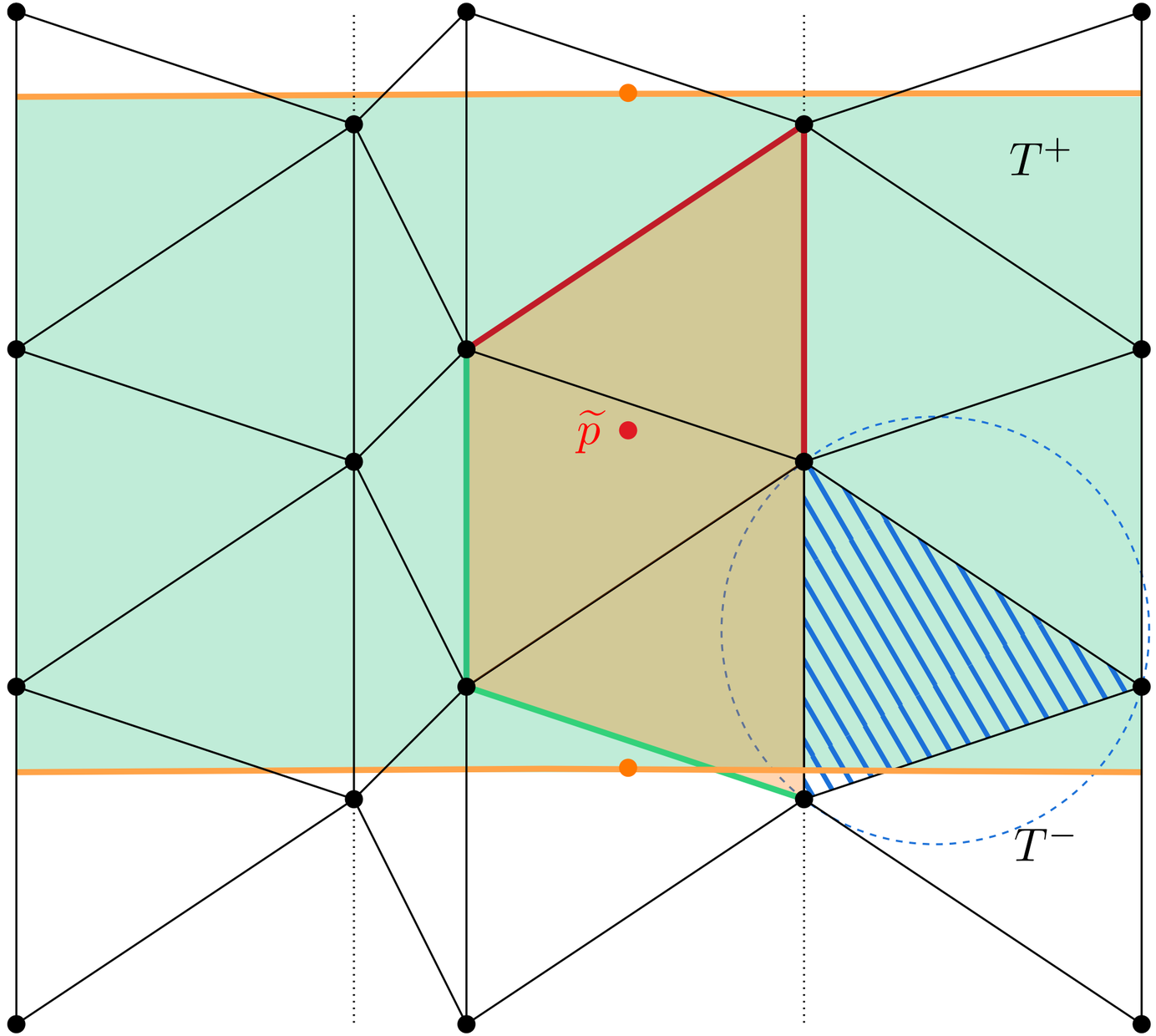


Processing in a banana

→ Conflict with $\widetilde{p}$ **and** has a vertex between the two lines.

→ Push the two other sides in the stack.

Insertion in the 4ε-thin part

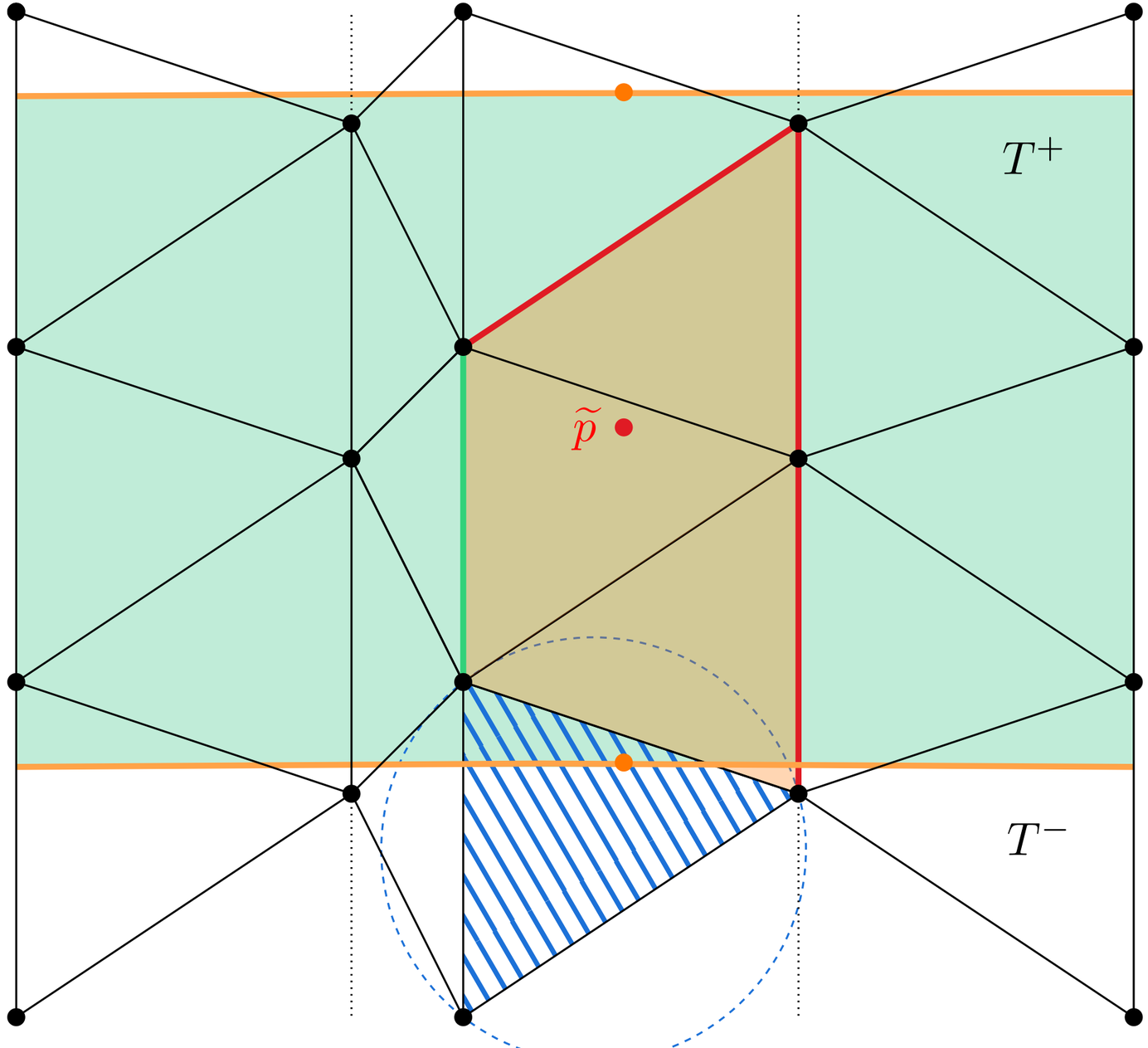


Processing in a banana

→ Not in conflict with $\widetilde{p}$.

→ The side in commun with the conflict zone belongs to the border.

Insertion in the 4ε-thin part

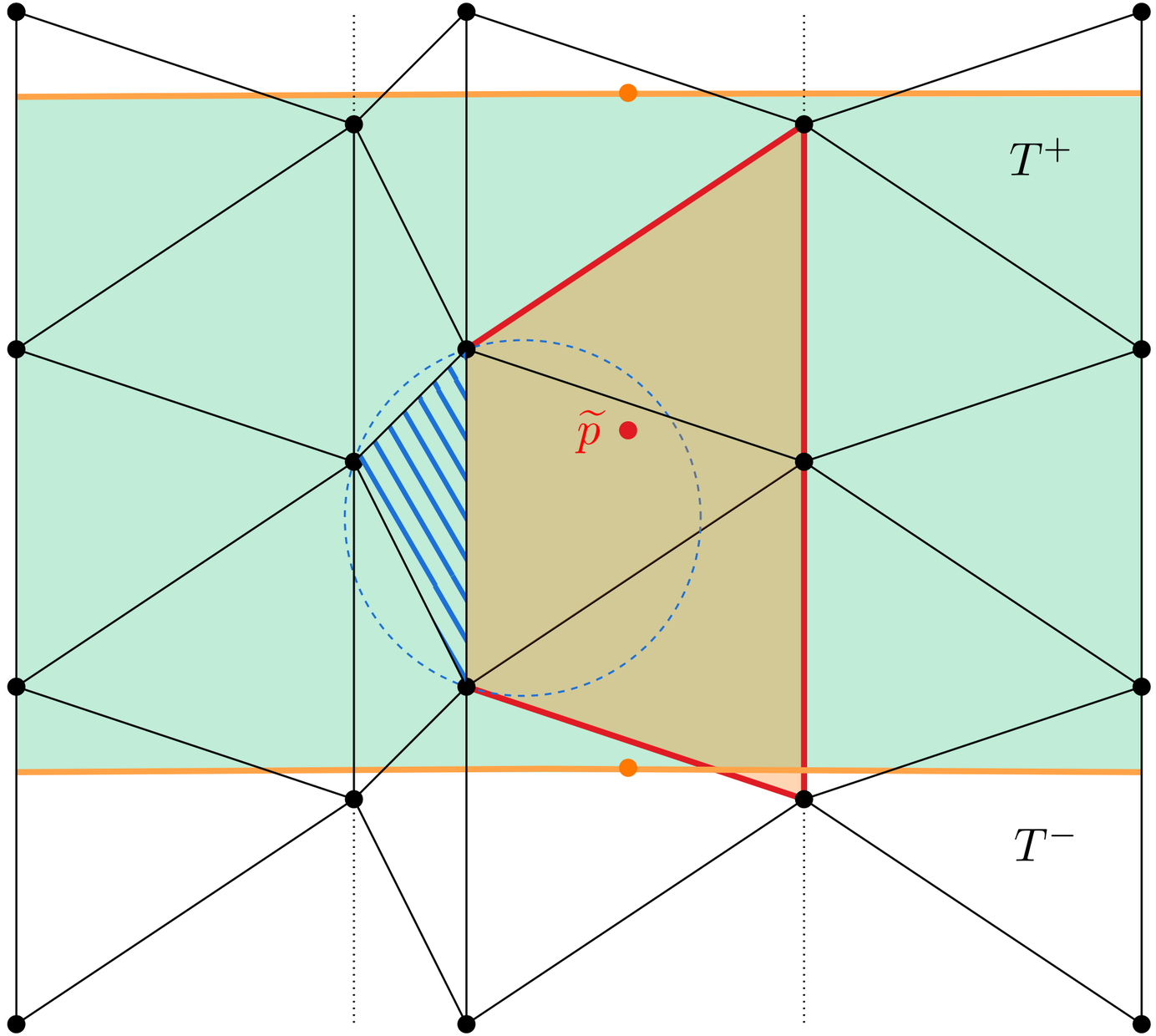


Processing in a banana

→ Not in conflict with $\widetilde{p}$.

→ The side in commun with the conflict zone belongs to the border.

Insertion in the 4ε-thin part

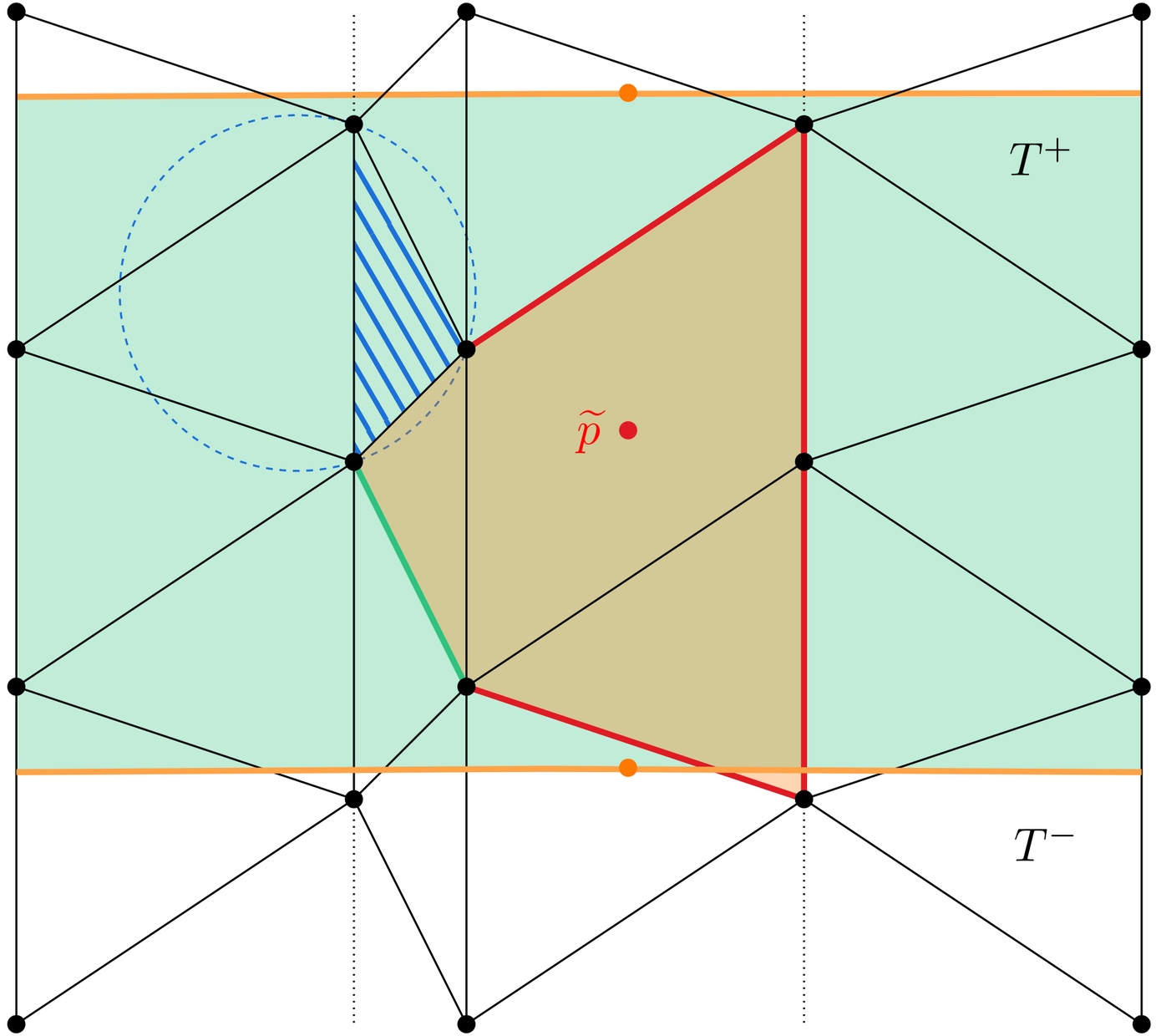


Processing in a banana

→ Conflict with $\widetilde{p}$ **and** has a vertex between the two lines.

→ Push the two other sides in the stack.

Insertion in the 4ε-thin part

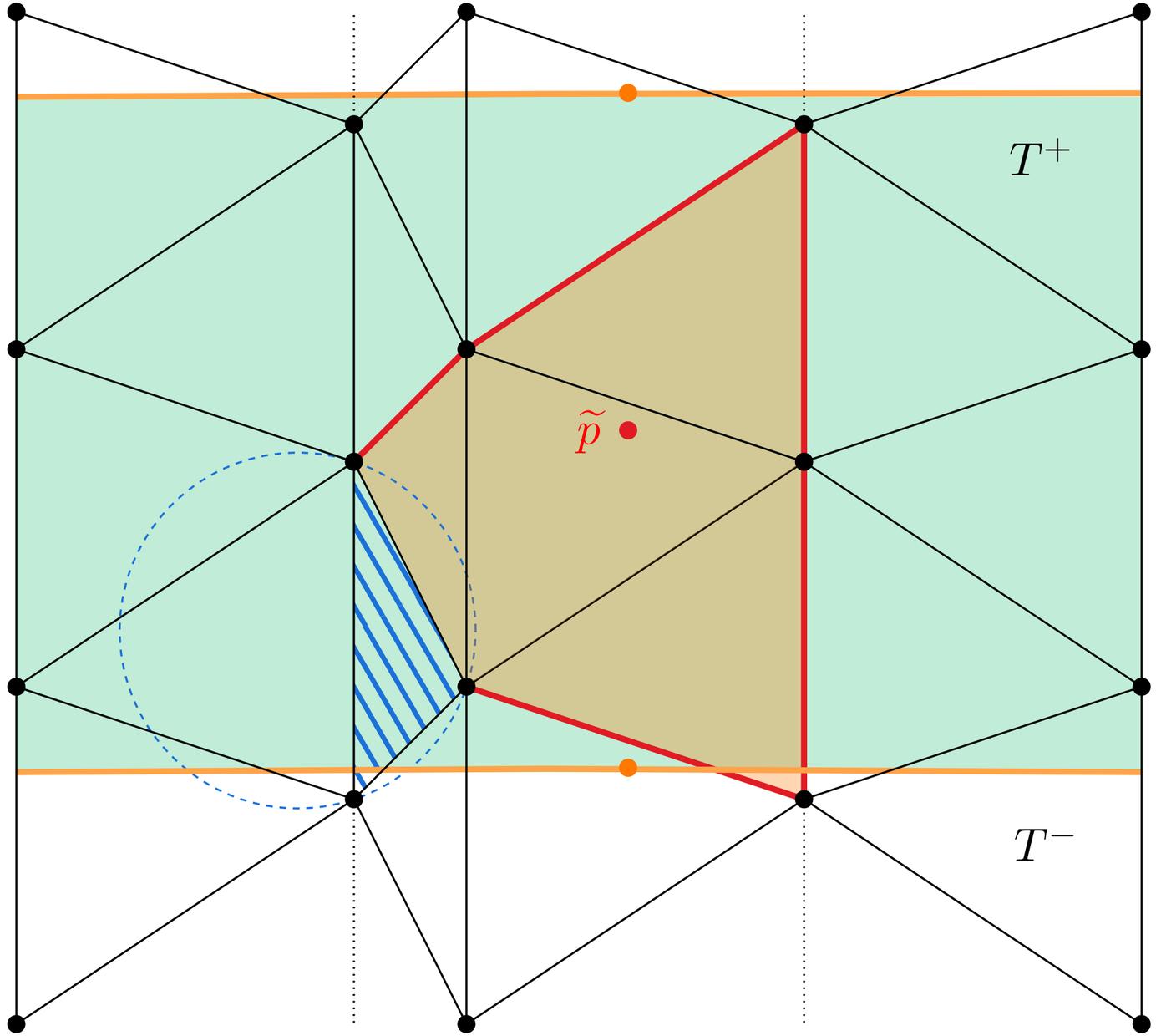


Processing in a banana

→ Not in conflict with $\widetilde{p}$.

→ The side in commun with the conflict zone belongs to the border.

Insertion in the 4ε-thin part

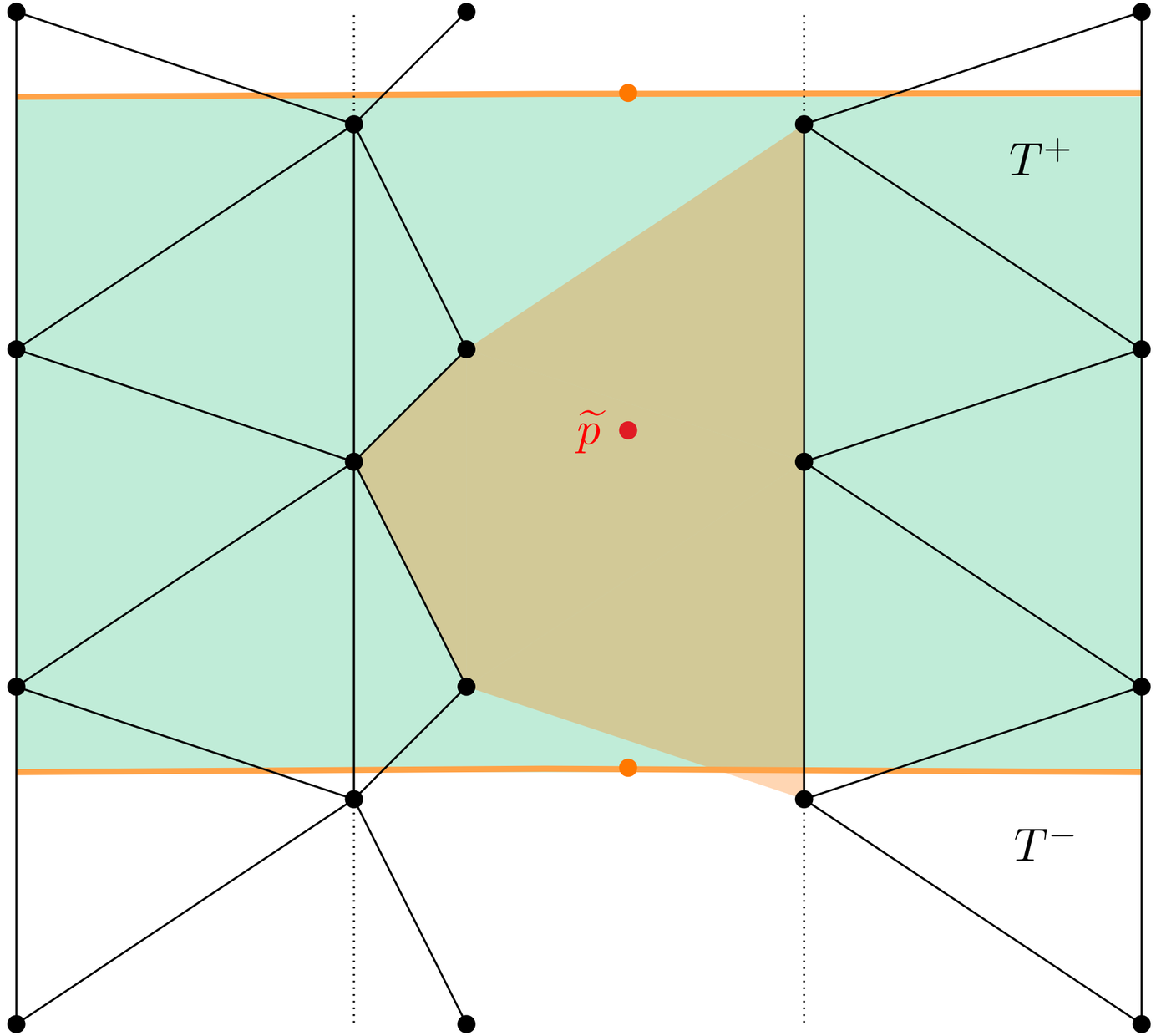


Processing in a banana

→ Not in conflict with $\widetilde{p}$.

→ The side in commun with the conflict zone belongs to the border.

Insertion in the 4ε-thin part

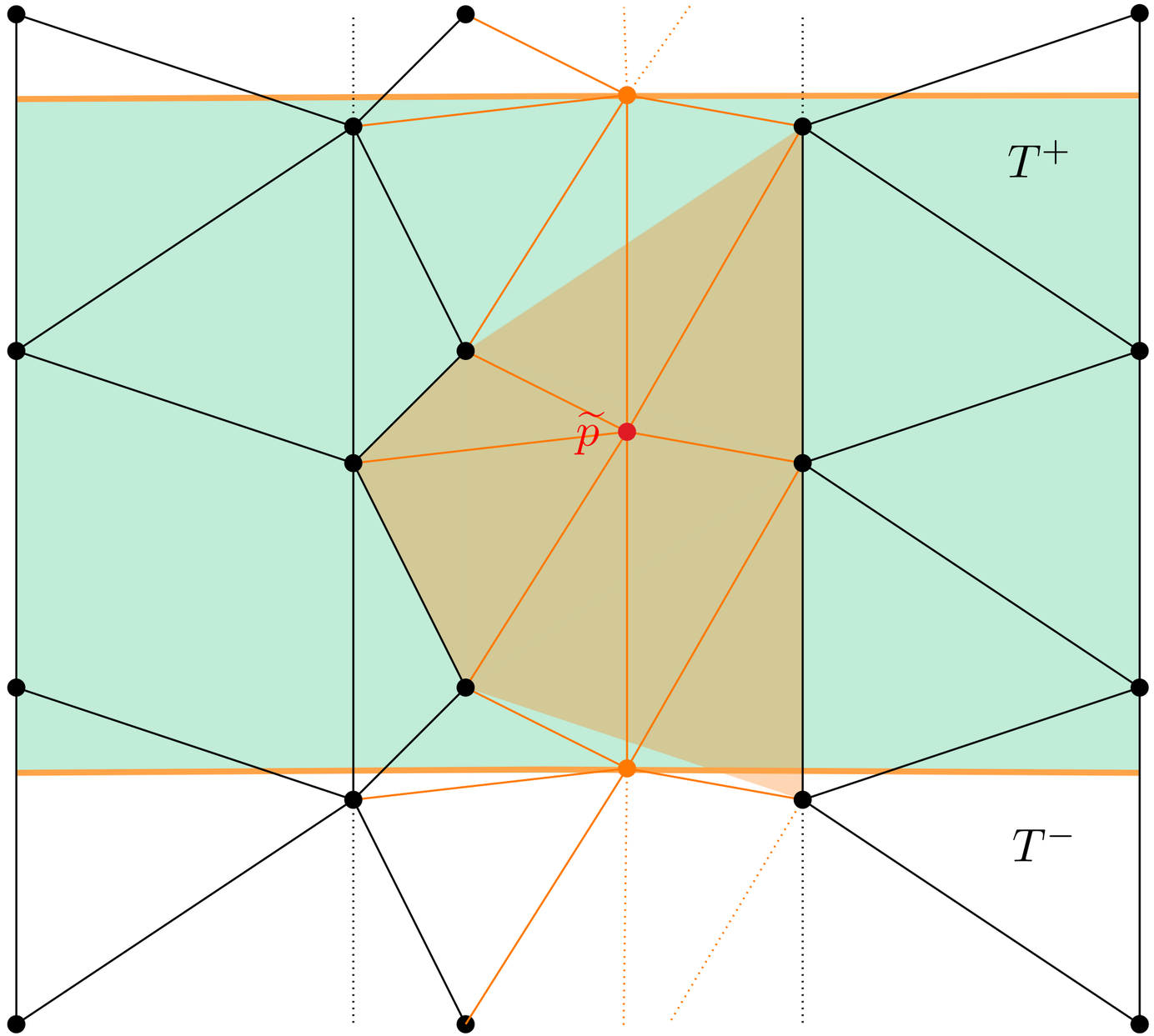


Processing in a banana

→ End of the BFS, we find the restricted conflict zone.

→ Remove all triangles (and their copies) of the restricted conflict zone.

Insertion in the 4ε-thin part



Processing in a banana

→ Connect the border's vertices to $\widetilde{p}$.

→ Check if new triangles are in conflict with two nearest copies of $\widetilde{p}$ and adapt the triangulation.

References

- About Delaunay triangulation and algorithms :
  - [Be08] M.de Berg et al. Computational Geometry: Algorithms and Applications
  - [La71]C.Lawson. "Transforming triangulations"
  - [Bro79]K.Q. Brown. Voronoi diagrams from convex hulls.
  - [D87] A. Dwyer: "A faster divide-and-conquer algorithm for constructing Delaunay triangulations".
- About Delaunay triangulation on hyperbolic surfaces :
  - [DST24] V. Despré, J-M. Schlenker, and M. Teillaud. "Flipping Geometric Triangulations on Hyperbolic Surfaces"
  - [IT17] I. Iordanov and M. Teillaud : "Implementing Delaunay Triangulations of the Bolza Surface"
- The original Bowyer's article :
  - [Bo81] A. Bowyer. "Computing Dirichlet tessellations"
- The surface initialization algorithm can be found in :
  - [DDLPT26](preprint) V. Delecroix, V. Despré, H. Parlier, C. Lanuel, and M. Teillaud. "Computing an ε-net of a closed hyperbolic surface"
- Collar's lemma :
  - [Bu10] P. Buser. Geometry and Spectra of Compact Riemann Surfaces
- Our article :
  - [DPP](preprint) V. Deprés, D. Perrot, M. Pouget. "Bowyer-Watson Algorithm for Delaunay Triangulation on Hyperbolic Surfaces".