Deletion in Abstract Voronoi diagrams in Expected Linear Time

Kolja Junginger and **Evanthia Papadopoulou**

Università della Svizzera italiana, Lugano, Switzerland

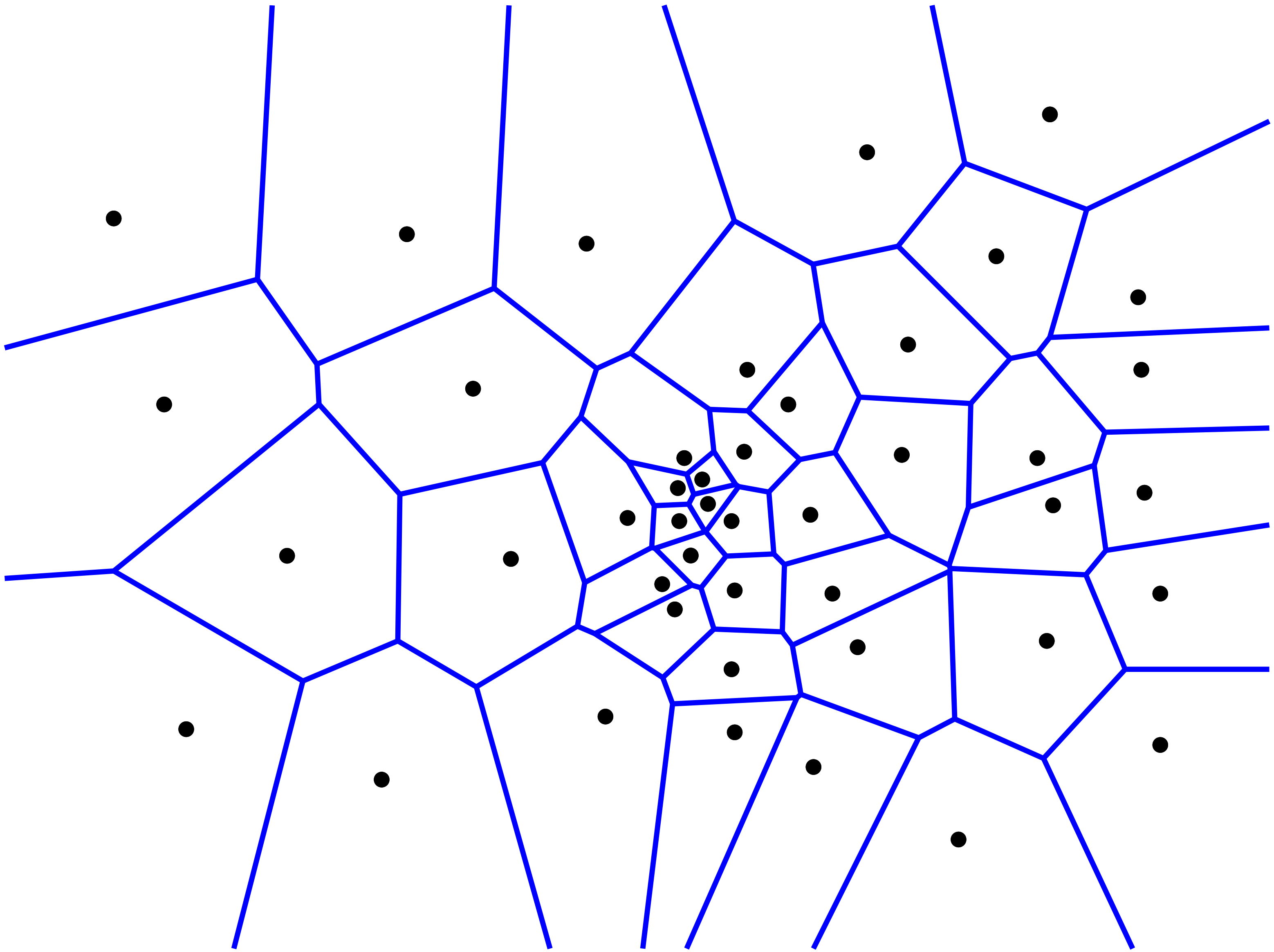
HMI Workshop, June 18-21, 2018

Voronoi diagram of points

$S =$ set of n point sites

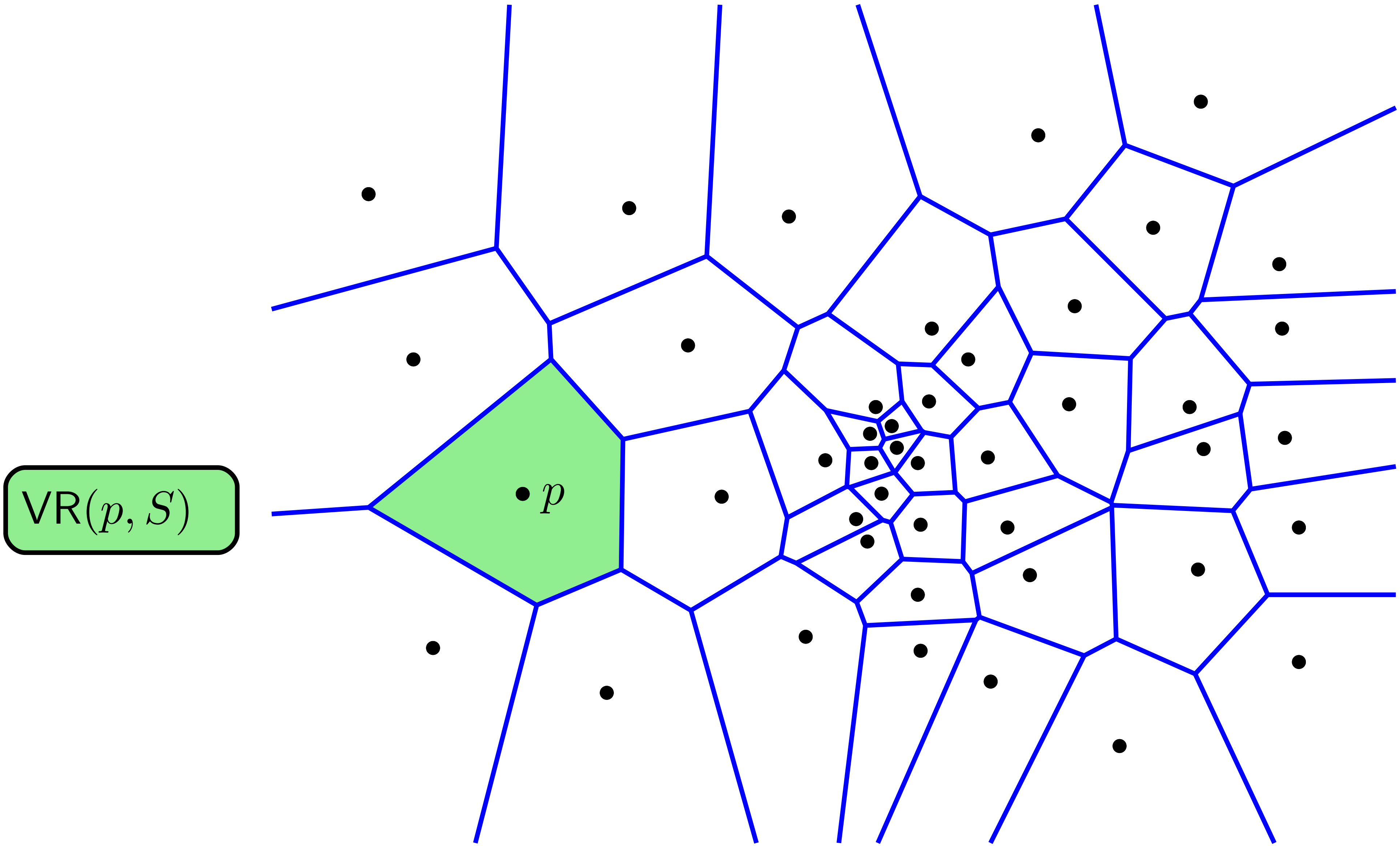
Voronoi diagram of points

S = set of n point sites



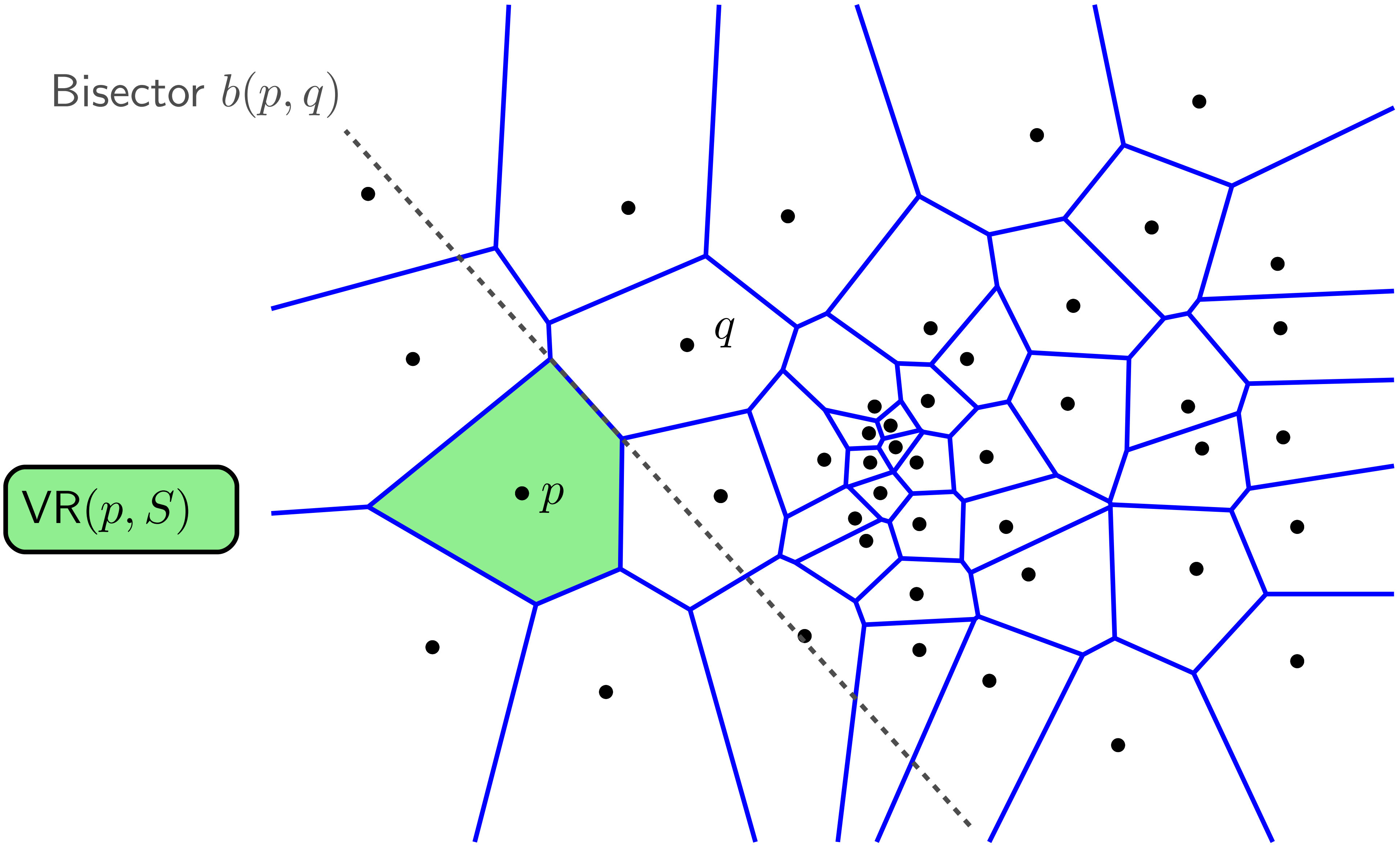
Voronoi diagram of points

S = set of n point sites



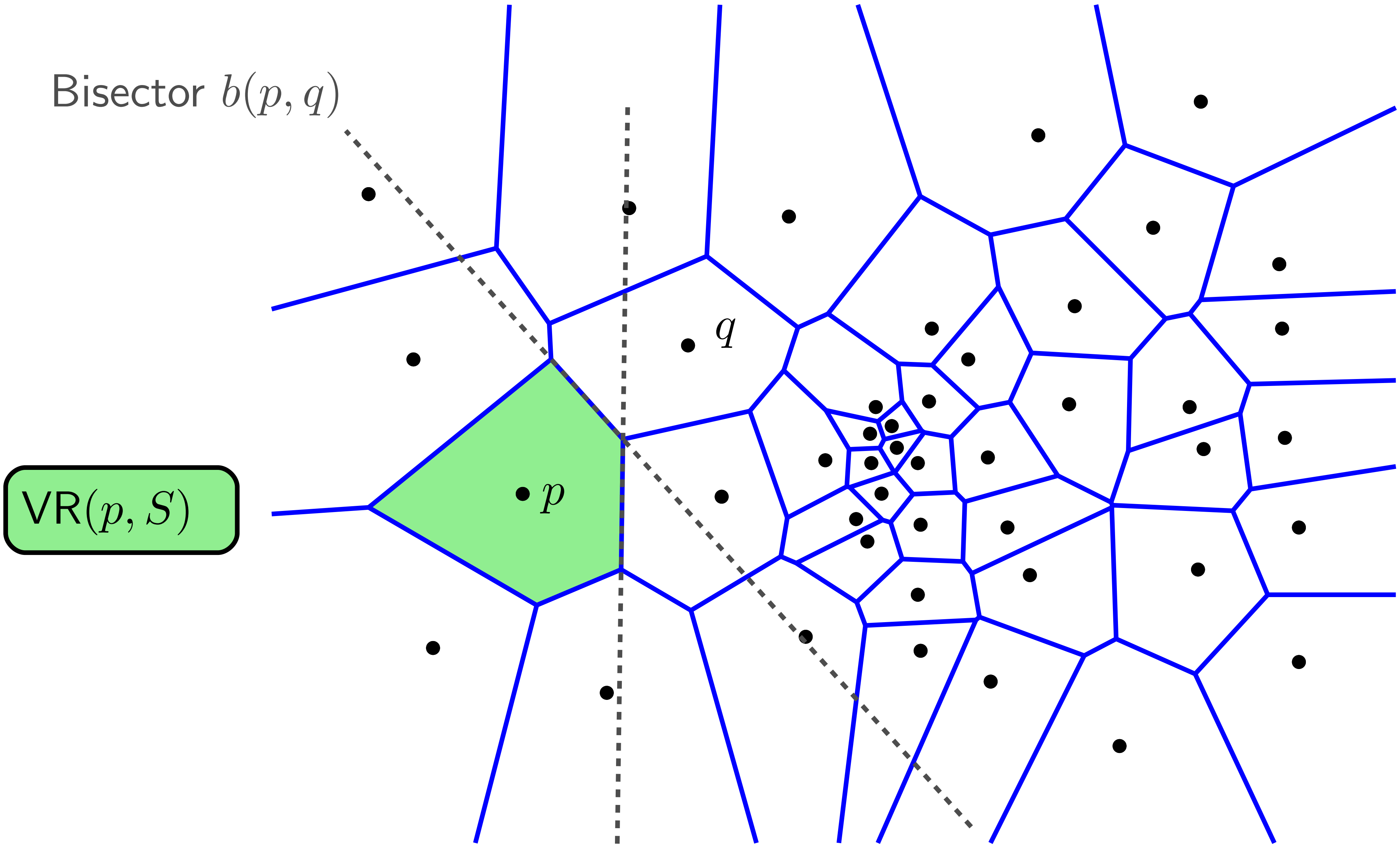
Voronoi diagram of points

S = set of n point sites



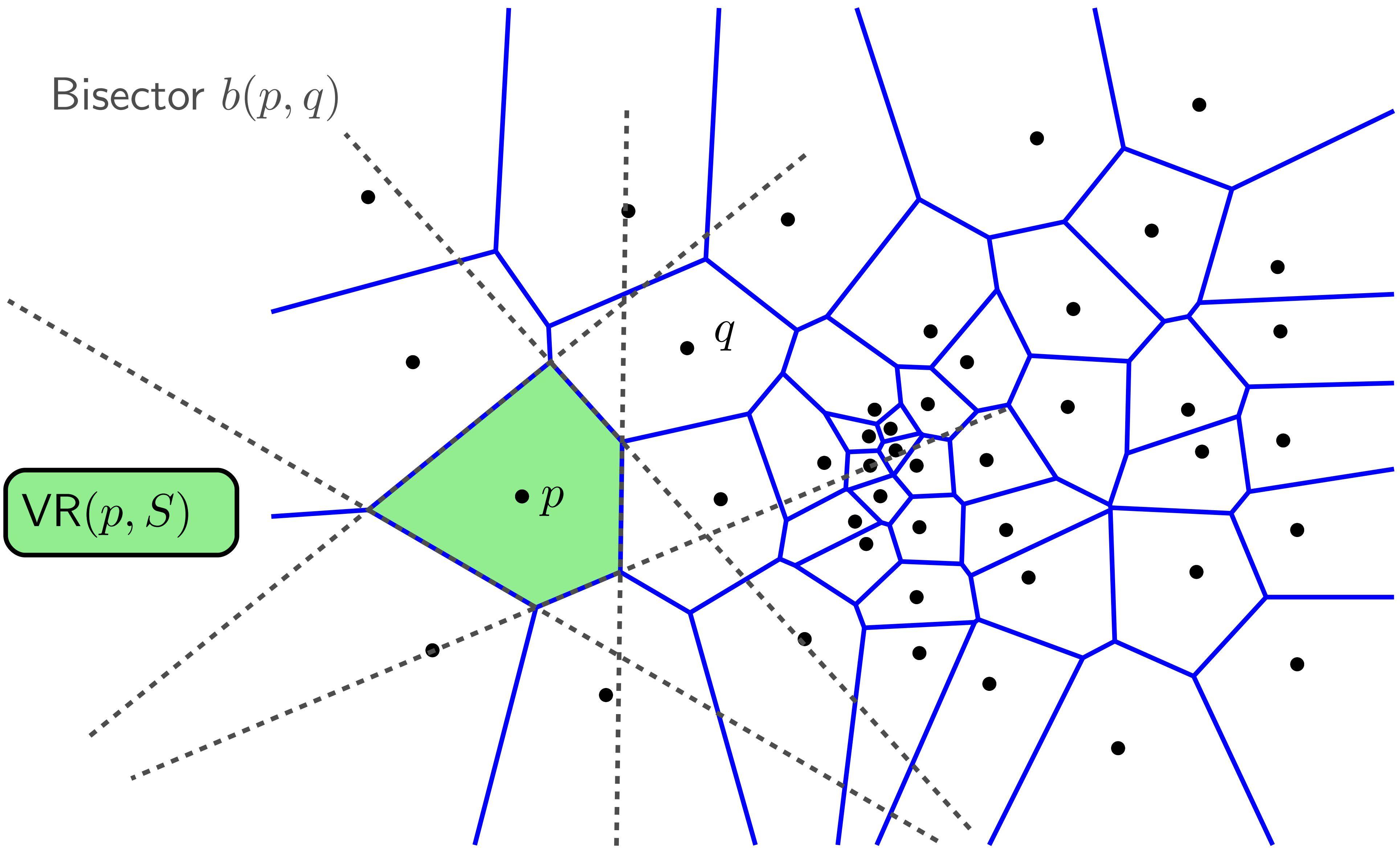
Voronoi diagram of points

S = set of n point sites



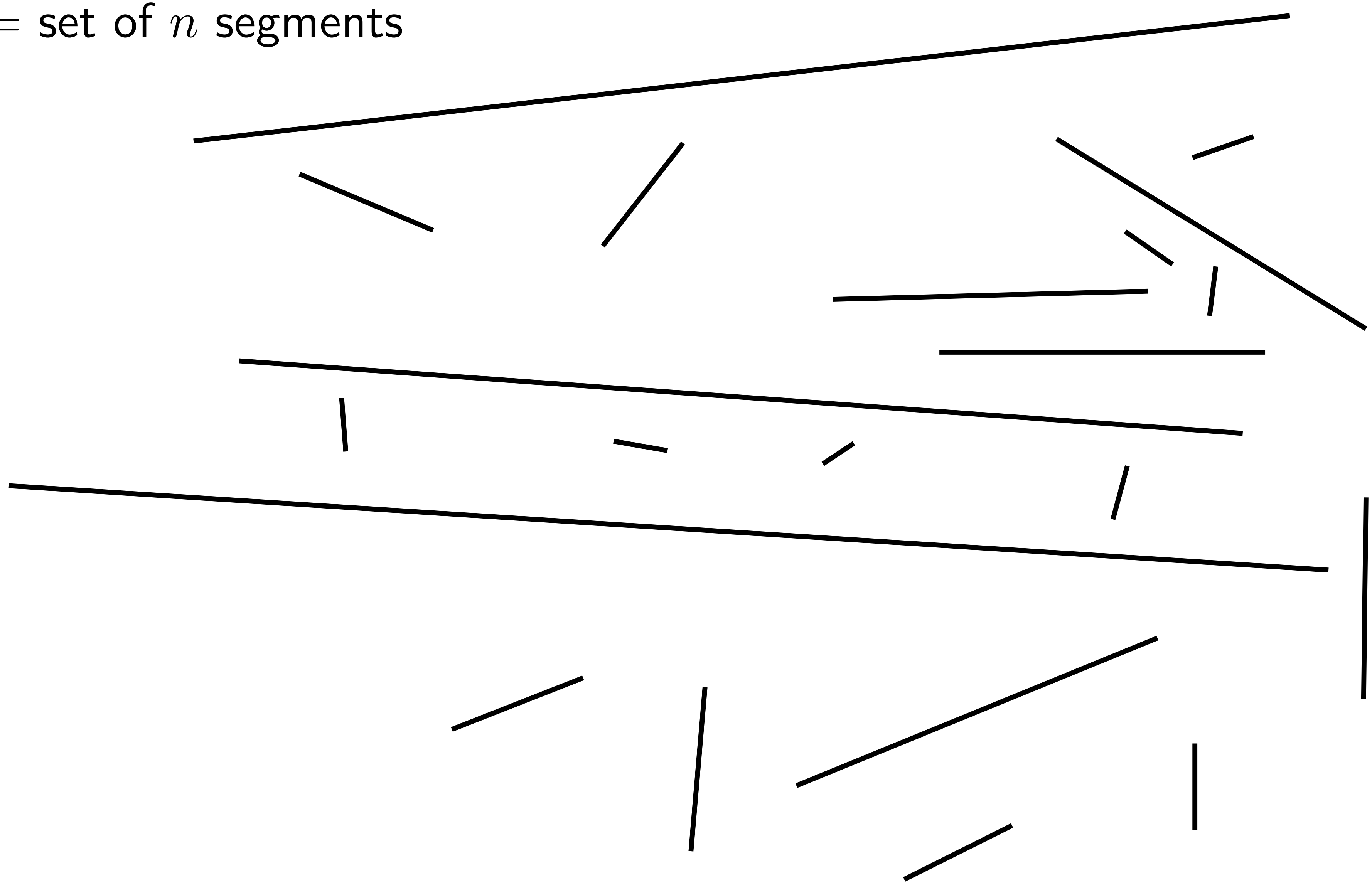
Voronoi diagram of points

S = set of n point sites



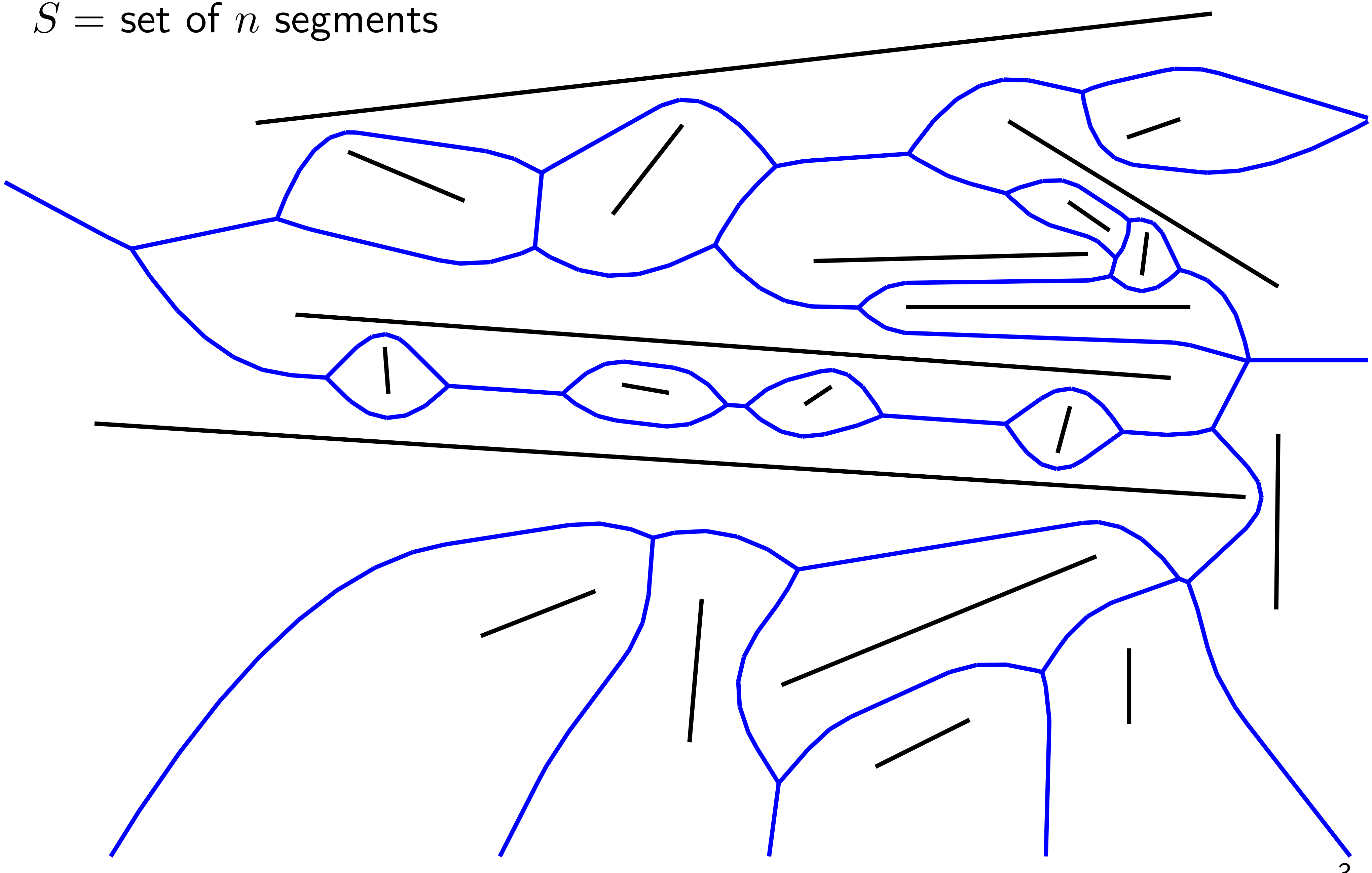
Voronoi diagram of segments

$S =$ set of n segments



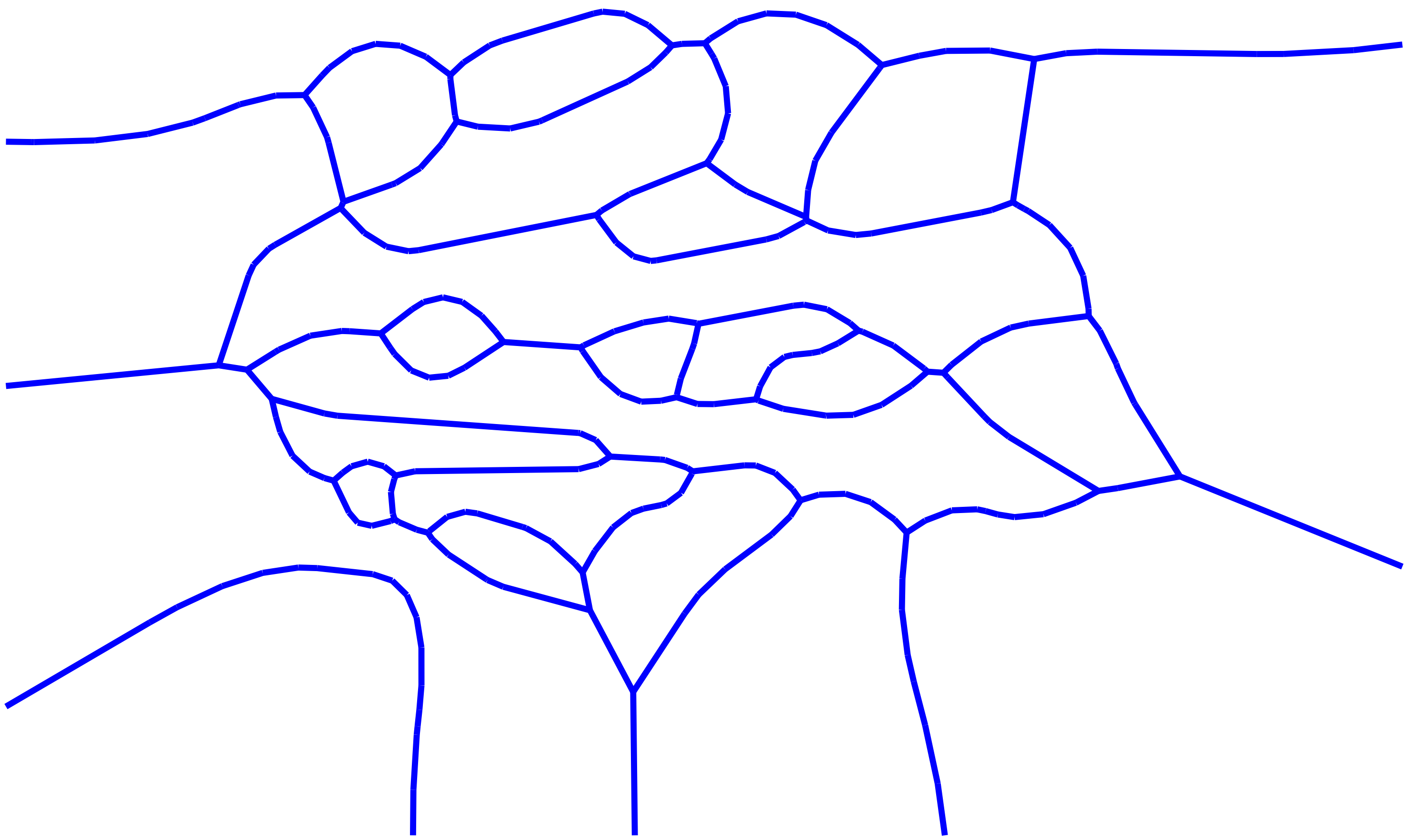
Voronoi diagram of segments

S = set of n segments



Abstract Voronoi diagram

- Offer a unifying framework to many concrete diagrams.
  Defined on bisecting curves satisfying some axioms, rather than sites.

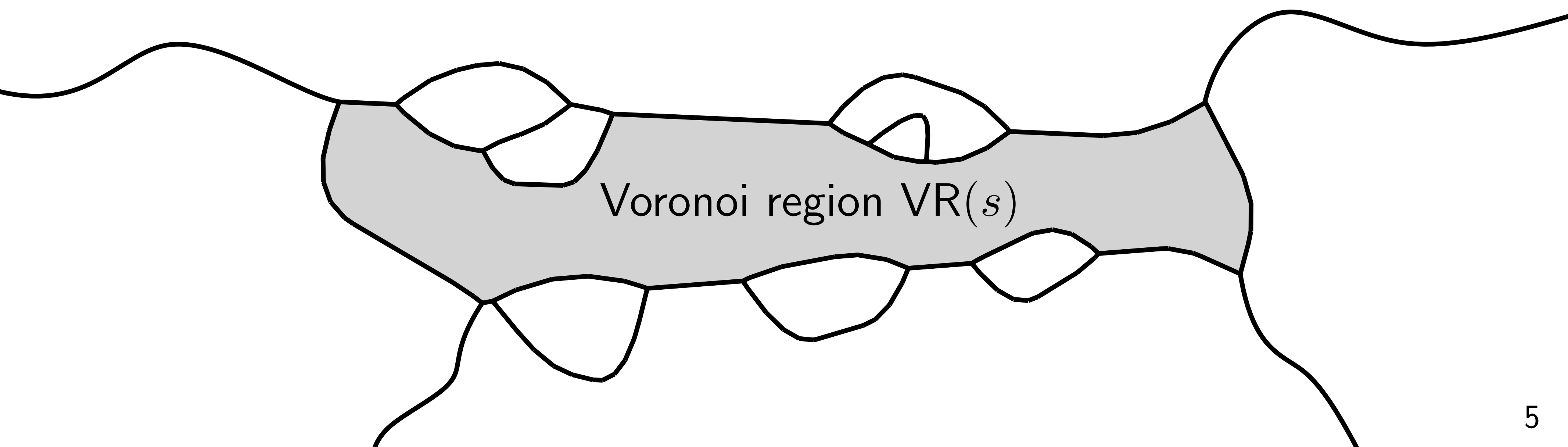


Rolf Klein. *Concrete and Abstract Voronoi Diagrams*. 1989.

The problem

The problem

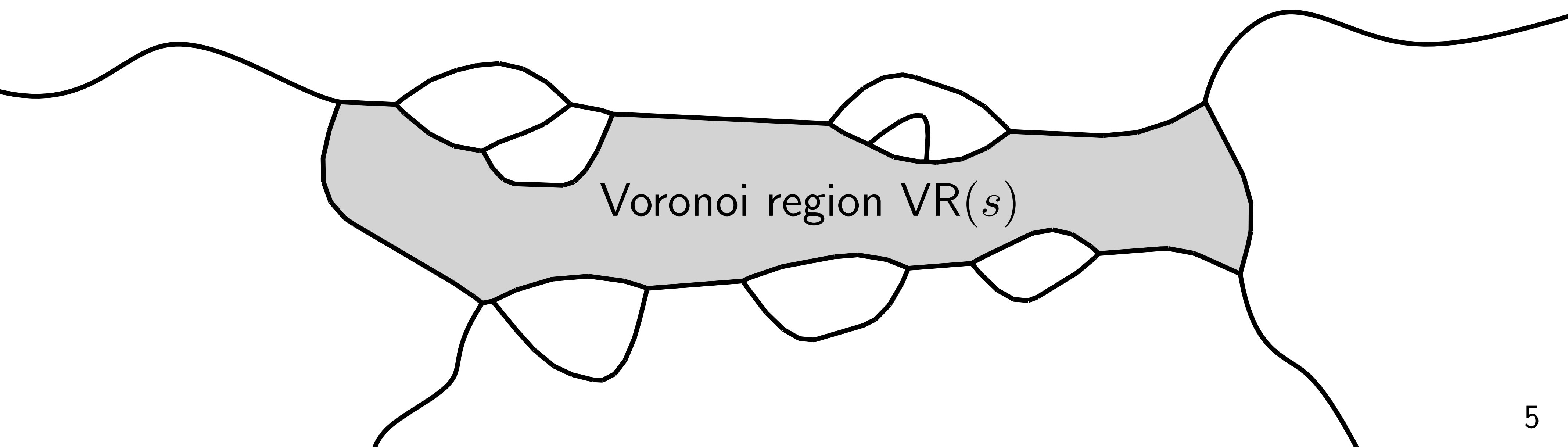
Given: Voronoi diagram $\mathcal{V}(S)$ and a site $s \in S$.



The problem

Given: Voronoi diagram $\mathcal{V}(S)$ and a site $s \in S$.

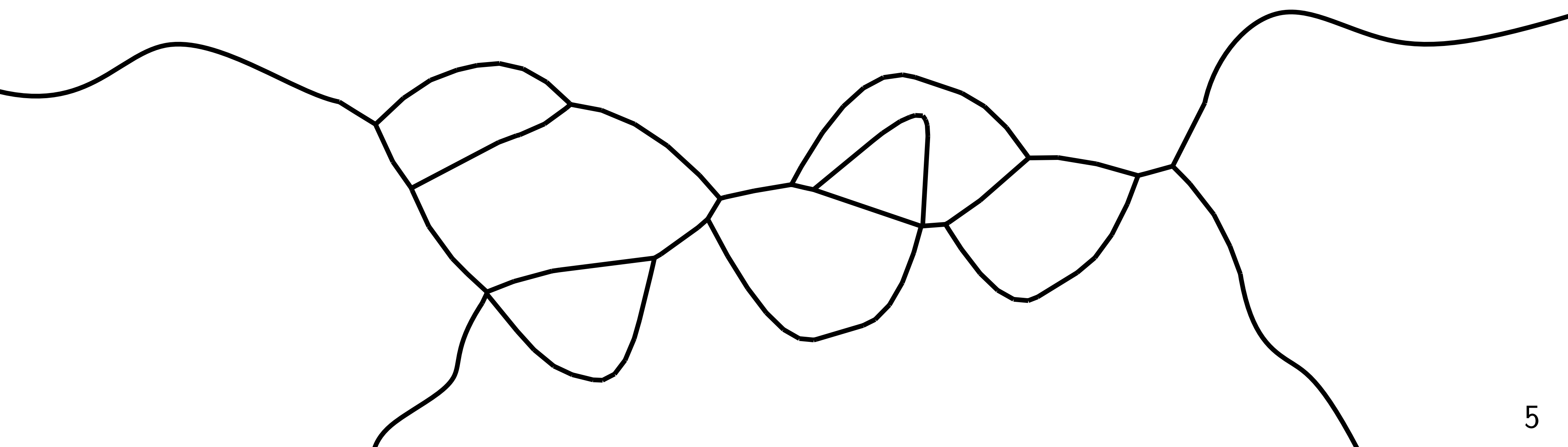
Goal: Update $\mathcal{V}(S)$, after **deletion** of site s, in time **linear** in the number of changes in the diagram.



The problem

Given: Voronoi diagram $\mathcal{V}(S)$ and a site $s \in S$.

Goal: Update $\mathcal{V}(S)$, after **deletion** of site s, in time **linear** in the number of changes in the diagram.



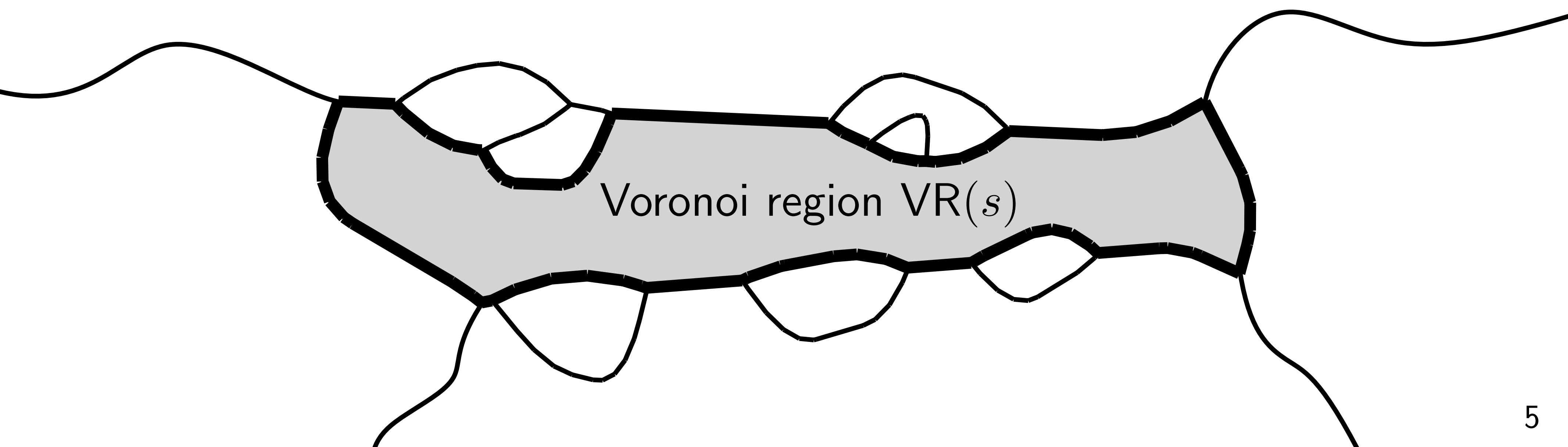
The problem

Given: Voronoi diagram $\mathcal{V}(S)$ and a site $s \in S$.

Goal: Update $\mathcal{V}(S)$, after **deletion** of site s, in time **linear** in the number of changes in the diagram.

Given: Voronoi region VR(s).

Goal: Compute $\mathcal{V}(S \setminus s)$ within VR(s), in time **linear** (randomized or deterministic) in the complexity of the region's boundary



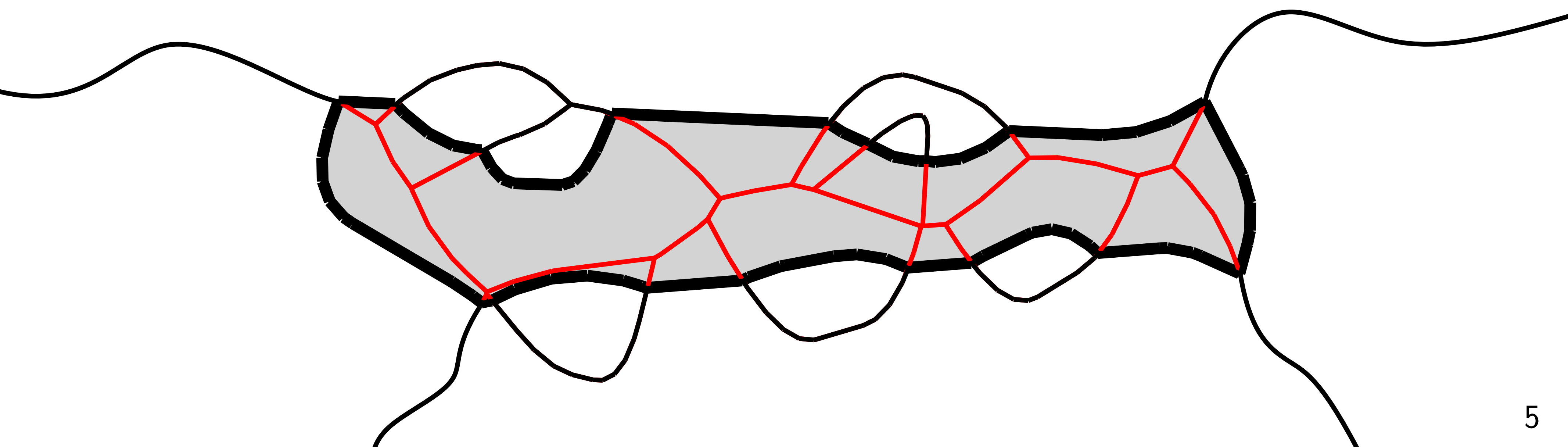
The problem

Given: Voronoi diagram $\mathcal{V}(S)$ and a site $s \in S$.

Goal: Update $\mathcal{V}(S)$, after **deletion** of site s, in time **linear** in the number of changes in the diagram.

Given: Voronoi region VR(s).

Goal: Compute $\mathcal{V}(S \setminus s)$ within VR(s), in time **linear** (randomized or deterministic) in the complexity of the region's boundary



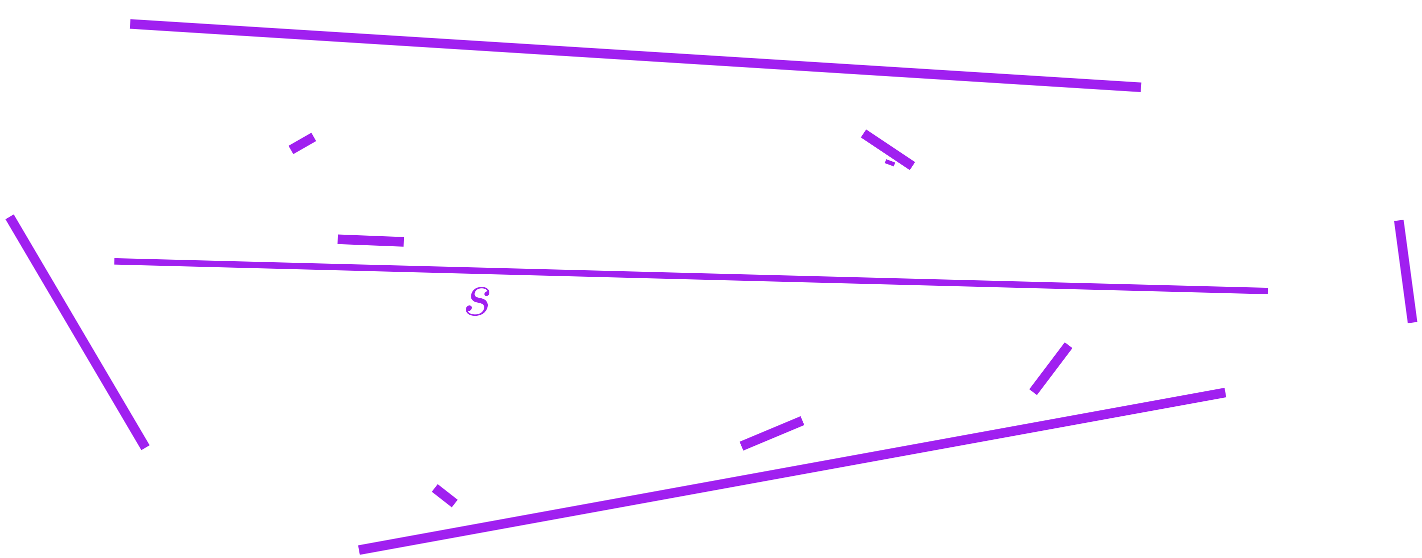
The problem

Given: Voronoi diagram $\mathcal{V}(S)$ and a site $s \in S$.

Goal: Update $\mathcal{V}(S)$, after **deletion** of site s, in time **linear** in the number of changes in the diagram.

Given: Voronoi region $\mathrm{VR}(s)$.

Goal: Compute $\mathcal{V}(S \setminus s)$ within $\mathrm{VR}(s)$, in time **linear** (randomized or deterministic) in the complexity of the region's boundary.



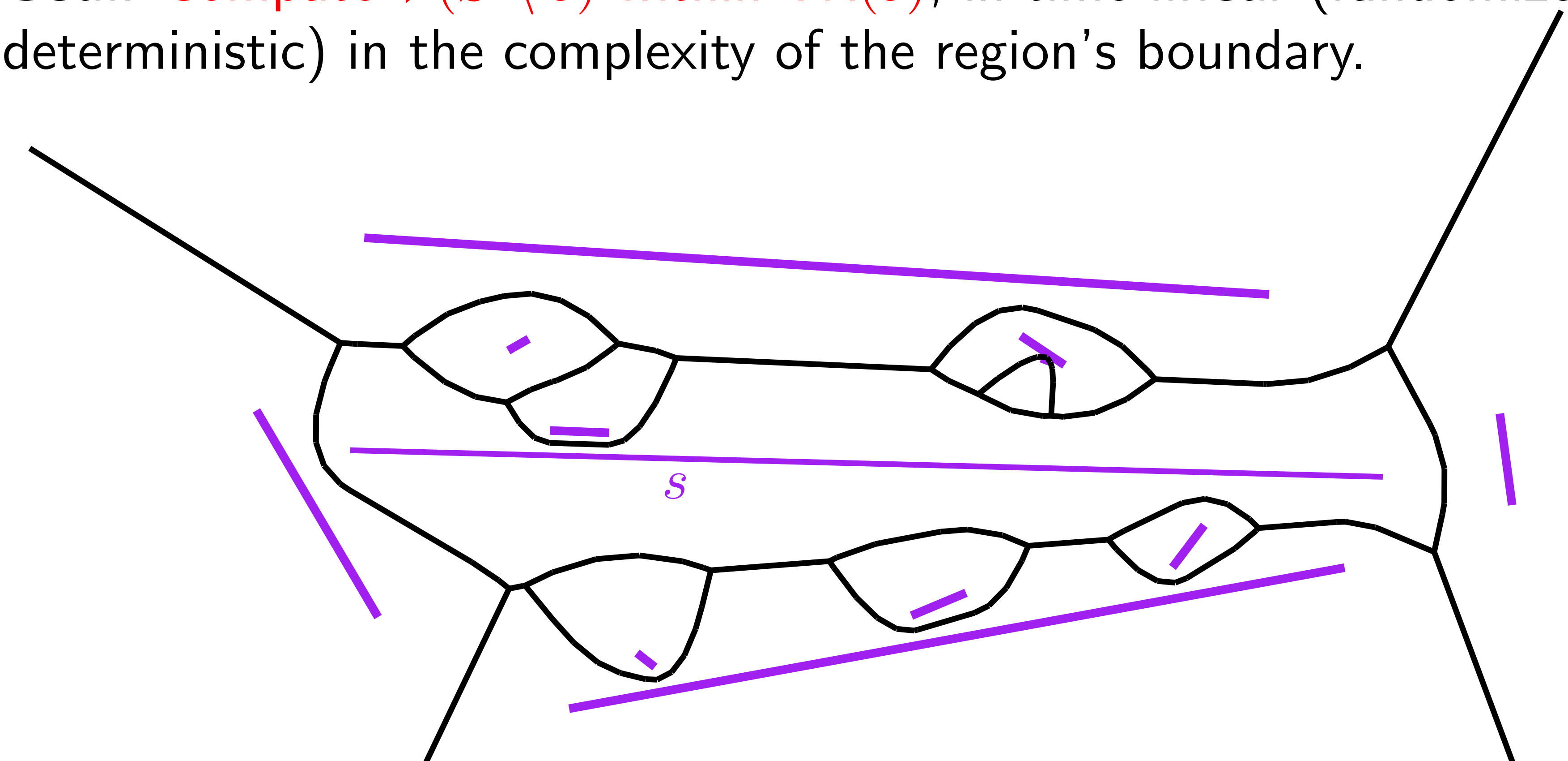
The problem

Given: Voronoi diagram $\mathcal{V}(S)$ and a site $s \in S$.

Goal: Update $\mathcal{V}(S)$, after **deletion** of site s, in time **linear** in the number of changes in the diagram.

Given: Voronoi region VR(s).

Goal: Compute $\mathcal{V}(S \setminus s)$ within VR(s), in time **linear** (randomized or deterministic) in the complexity of the region's boundary.



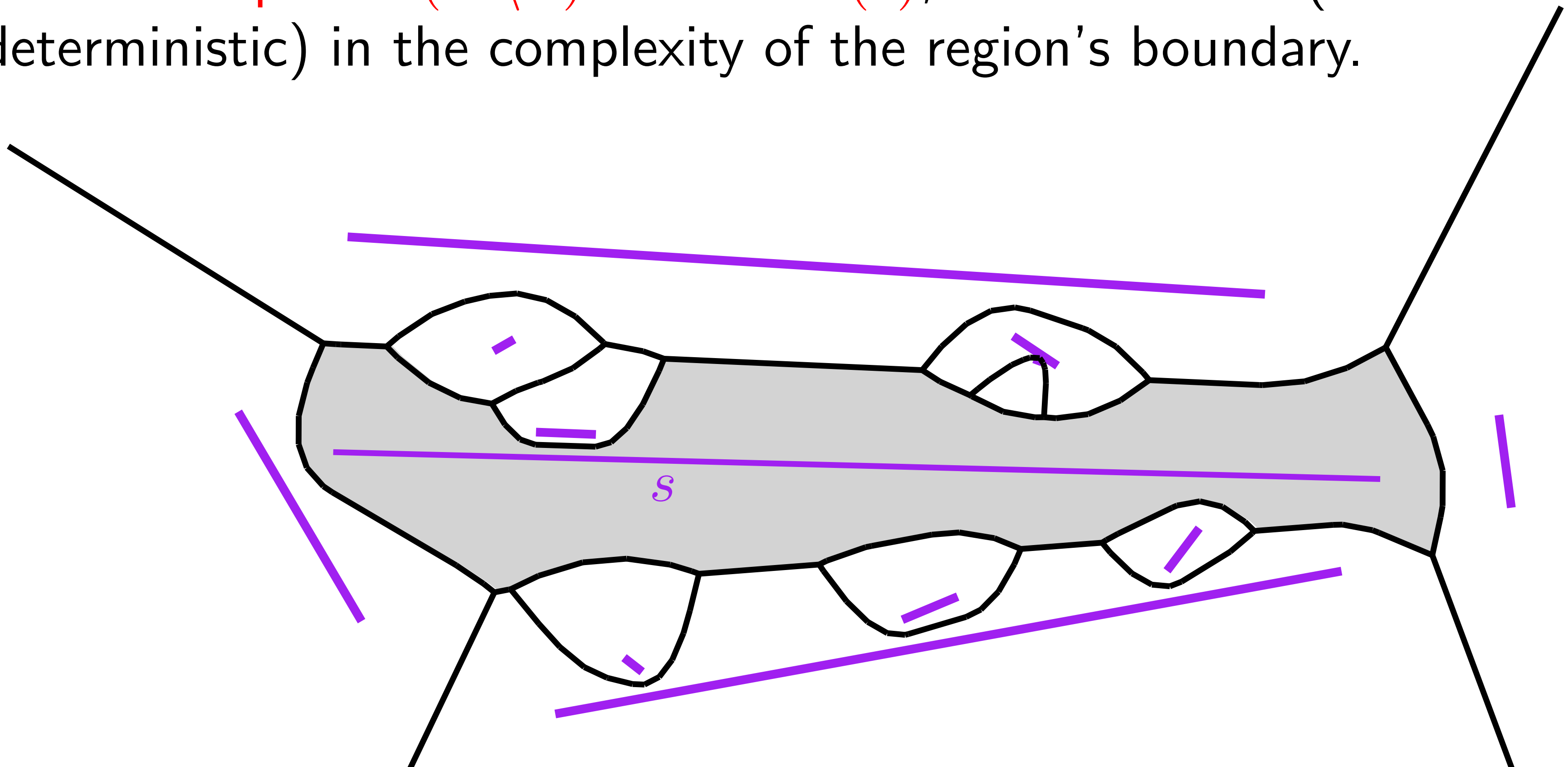
The problem

Given: Voronoi diagram $\mathcal{V}(S)$ and a site $s \in S$.

Goal: Update $\mathcal{V}(S)$, after **deletion** of site s, in time **linear** in the number of changes in the diagram.

Given: Voronoi region VR(s).

Goal: Compute $\mathcal{V}(S \setminus s)$ within VR(s), in time **linear** (randomized or deterministic) in the complexity of the region's boundary.



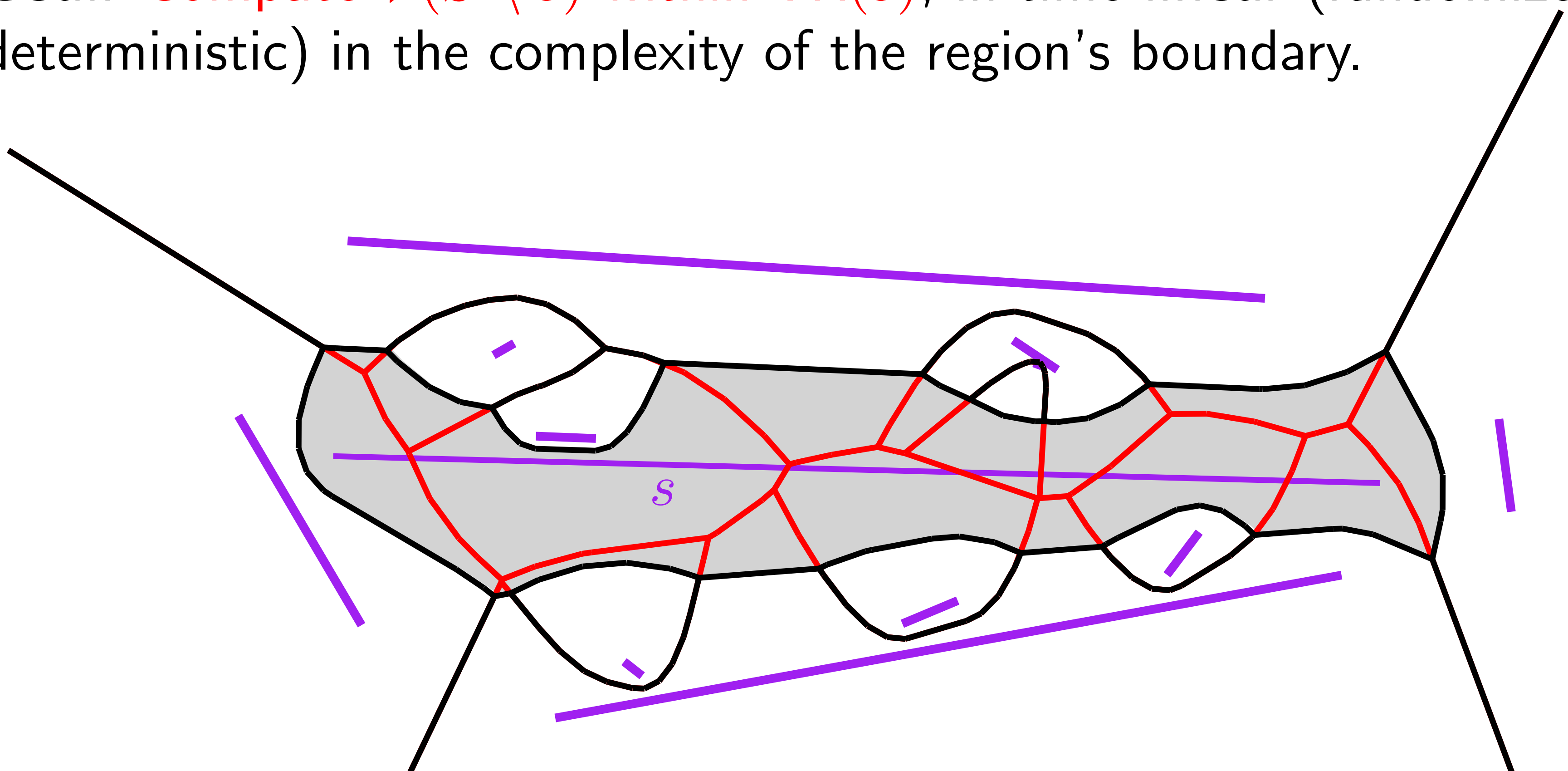
The problem

Given: Voronoi diagram $\mathcal{V}(S)$ and a site $s \in S$.

Goal: Update $\mathcal{V}(S)$, after **deletion** of site s, in time **linear** in the number of changes in the diagram.

Given: Voronoi region VR(s).

Goal: Compute $\mathcal{V}(S \setminus s)$ within VR(s), in time **linear** (randomized or deterministic) in the complexity of the region's boundary.



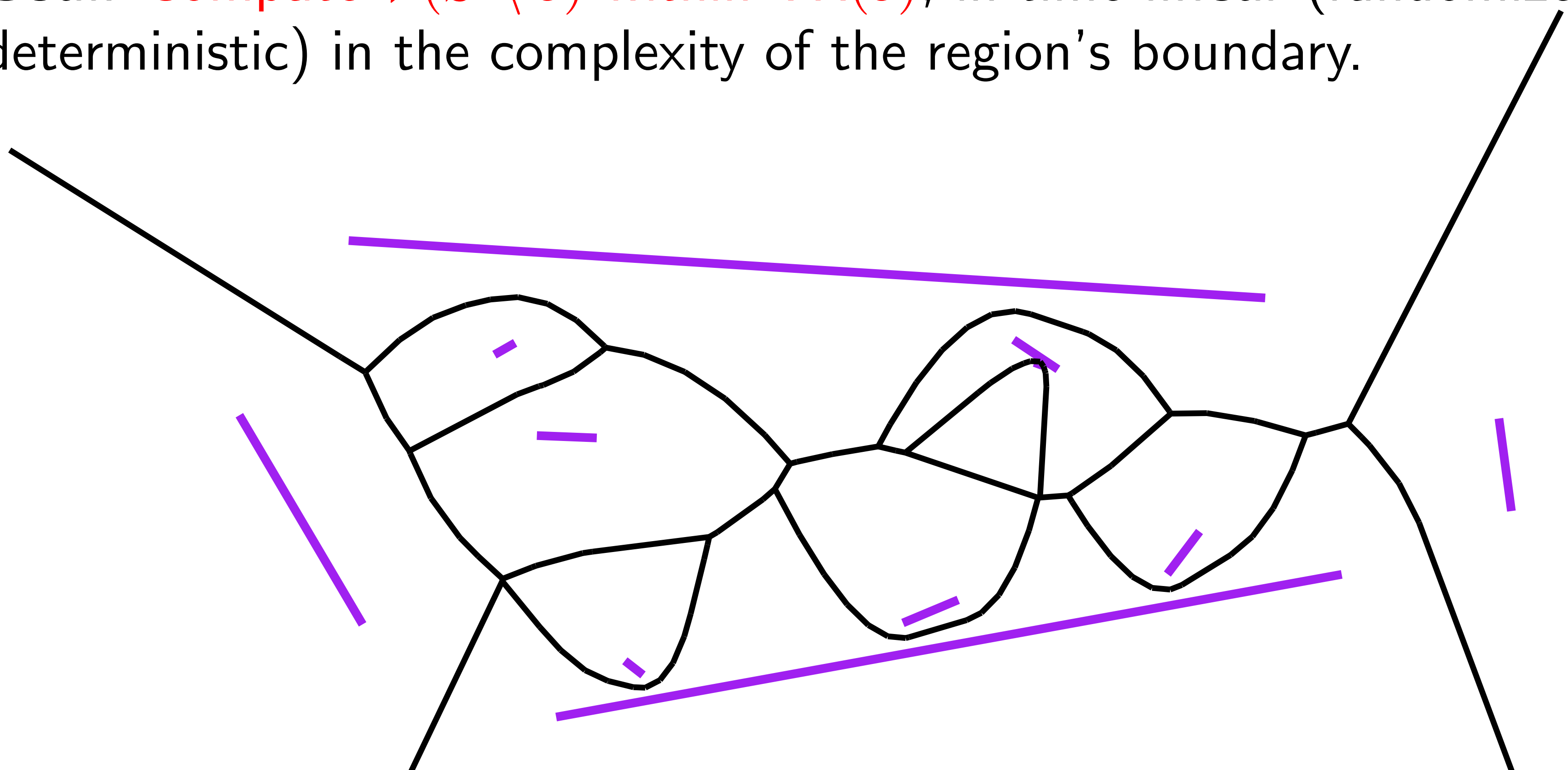
The problem

Given: Voronoi diagram $\mathcal{V}(S)$ and a site $s \in S$.

Goal: Update $\mathcal{V}(S)$, after **deletion** of site s, in time **linear** in the number of changes in the diagram.

Given: Voronoi region VR(s).

Goal: Compute $\mathcal{V}(S \setminus s)$ within VR(s), in time **linear** (randomized or deterministic) in the complexity of the region's boundary.



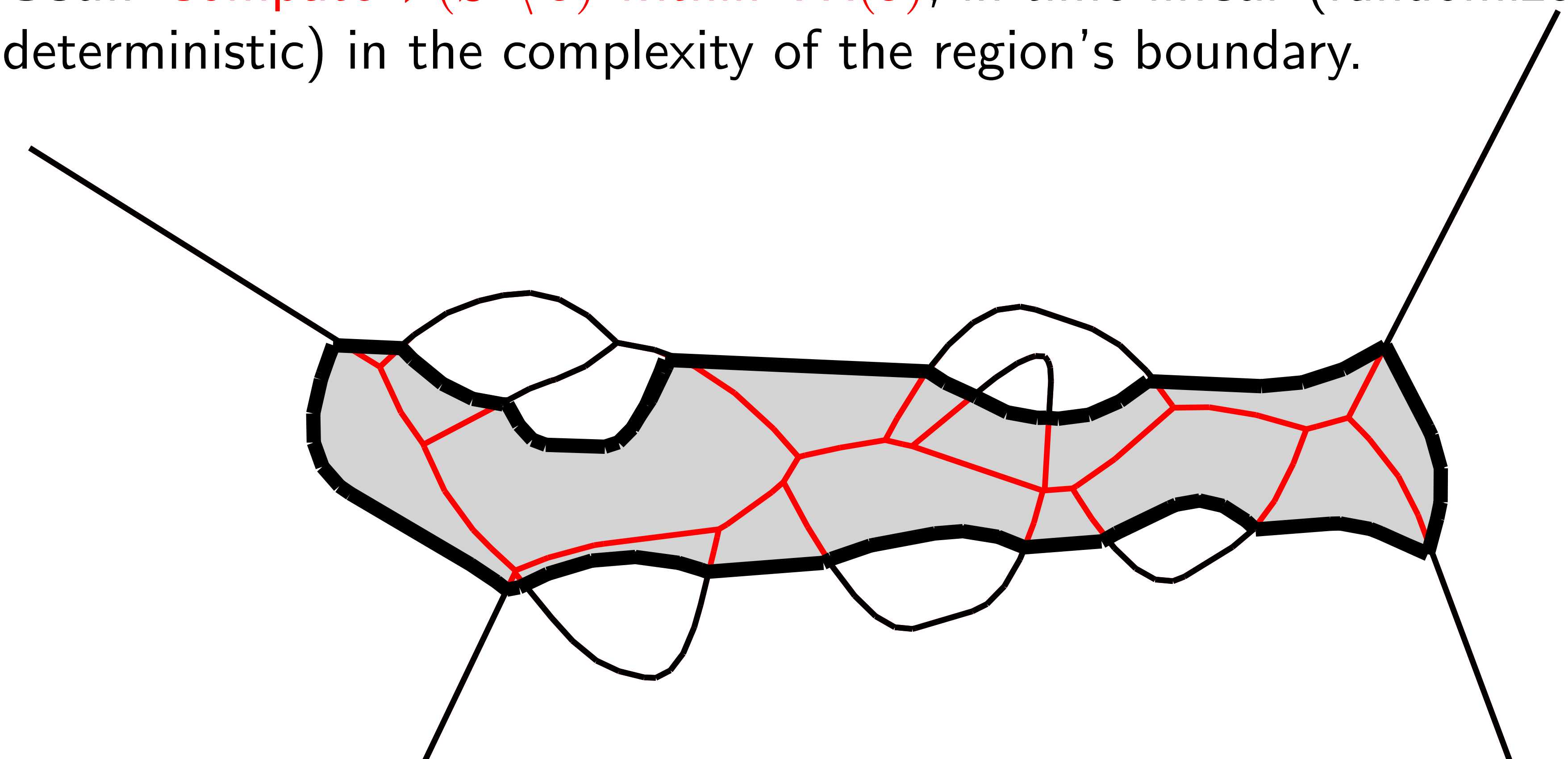
The problem

Given: Voronoi diagram $\mathcal{V}(S)$ and a site $s \in S$.

Goal: Update $\mathcal{V}(S)$, after **deletion** of site s, in time **linear** in the number of changes in the diagram.

Given: Voronoi region VR(s).

Goal: Compute $\mathcal{V}(S \setminus s)$ within VR(s), in time **linear** (randomized or deterministic) in the complexity of the region's boundary.



Problem history

- Problem is **solved for points** in deterministic linear time.
  [Aggarwal, Guibas, Saxe, Shore, DCG 1989]

Problem history

- Problem is **solved for points** in deterministic linear time.
  [Aggarwal, Guibas, Saxe, Shore, DCG 1989]

- Simpler **expected linear-time** algorithm (for points).
  [Chew, 1990]

Problem history

- Problem is **solved for points** in deterministic linear time.
  [Aggarwal, Guibas, Saxe, Shore, DCG 1989]

- Simpler **expected linear-time** algorithm (for points).
  [Chew, 1990]

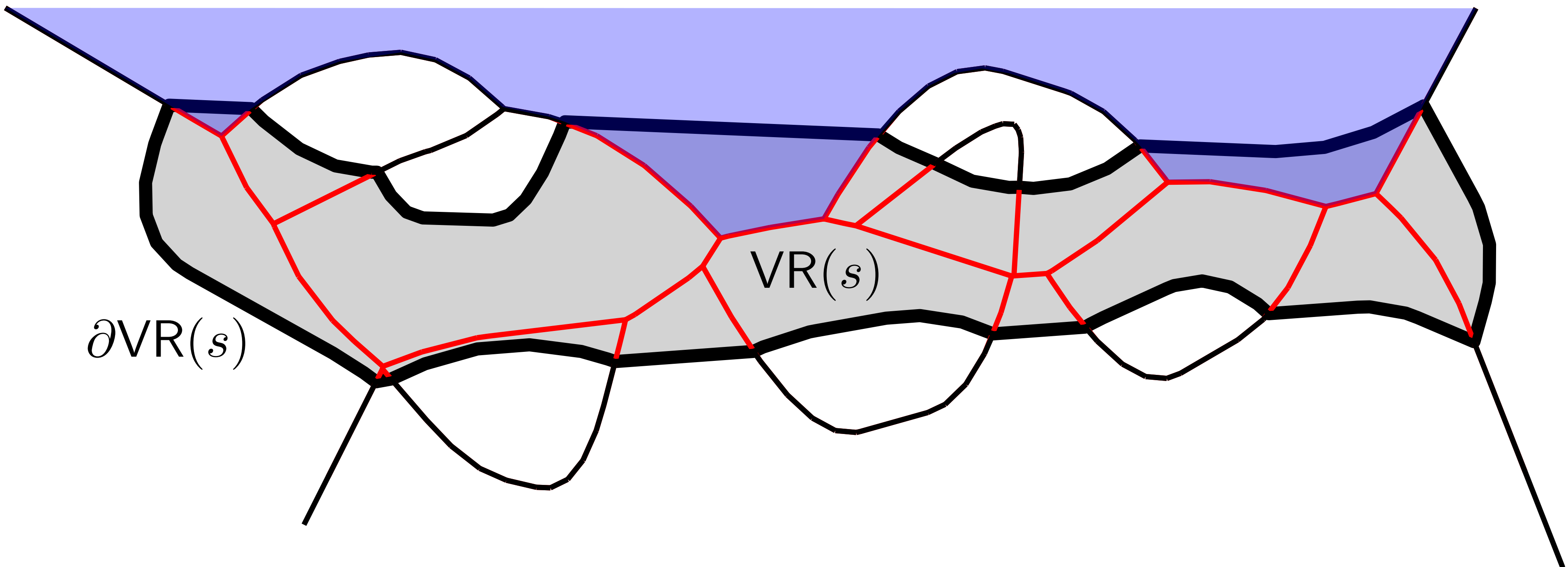
- **Open** since then for other sites (line segments, circles etc.).

- **Open** for **abstract Voronoi diagrams** (AVDs).

What is difficult?

For abstract Voronoi diagrams and non-point sites (line segments, circles):

The Voronoi region of **one site** can have **multiple faces** within $\mathrm{VR}(s)$.
– The sites along $\partial\mathrm{VR}(s)$ can **repeat**.
(AVDs: $\partial\mathrm{VR}(s)$ is a Davenport-Schinzel sequence of order 2.)



Linear-time Voronoi algorithms

- The Voronoi diagram of points in **convex position**.
  [Aggarwal, Guibas, Saxe, Shor, DCG 1989]

Linear-time Voronoi algorithms

- The Voronoi diagram of points in **convex position**.
  [Aggarwal, Guibas, Saxe, Shor, DCG 1989]
  – Update the Voronoi diagram of points, after **deletion** of one site.
  – The **farthest Voronoi diagram** of points, given their convex hull.
  – Update the **order-**$(k+1)$ **diagram** within an order-k region.

Linear-time Voronoi algorithms

- The Voronoi diagram of points in **convex position**.
  [Aggarwal, Guibas, Saxe, Shor, DCG 1989]
  - Update the Voronoi diagram of points, after **deletion** of one site.
  - The **farthest Voronoi diagram** of points, given their convex hull.
  - Update the **order-**$(k+1)$ **diagram** within an order-k region.

- Simpler expected linear-time algorithm for the same problems.
  [Chew, 1989]

Linear-time Voronoi algorithms

- The Voronoi diagram of points in **convex position**.
  [Aggarwal, Guibas, Saxe, Shor, DCG 1989]
  - Update the Voronoi diagram of points, after **deletion** of one site.
  - The **farthest Voronoi diagram** of points, given their convex hull.
  - Update the **order-**$(k+1)$ **diagram** within an order-k region.

- Simpler expected linear-time algorithm for the same problems.
  [Chew, 1989]

- **Hamiltonian abstract Voronoi diagrams** [Klein, Lingas, ISAAC 1994]
  Given a Hamiltonian curve, that visits **every region exactly once** and intersects **each bisector exactly once**.

Linear-time Voronoi algorithms

- The Voronoi diagram of points in **convex position**.
  [Aggarwal, Guibas, Saxe, Shor, DCG 1989]
  – Update the Voronoi diagram of points, after **deletion** of one site.
  – The **farthest Voronoi diagram** of points, given their convex hull.
  – Update the **order-**$(k+1)$ **diagram** within an order-k region.

- Simpler expected linear-time algorithm for the same problems.
  [Chew, 1989]

- **Hamiltonian abstract Voronoi diagrams** [Klein, Lingas, ISAAC 1994]
  Given a Hamiltonian curve, that visits **every region exactly once** and intersects **each bisector exactly once**.

- **Forest-Like abstract Voronoi Diagrams** [Bohler, Klein, Liu, CCCG 2014]
  Similar conditions, where no region can have multiple faces.

Linear-time Voronoi algorithms

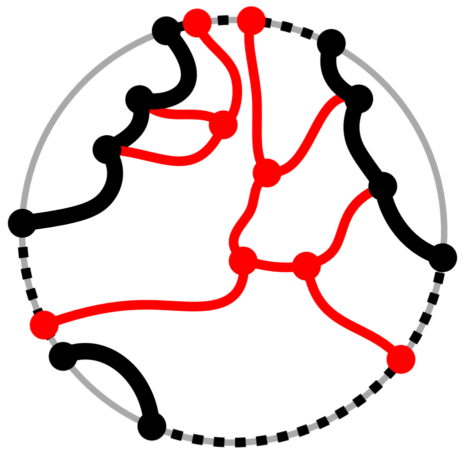
- The **medial axis** of a simple polygon.
  [Chin, Snoeyink, Wang, DCG 1999]

Linear-time Voronoi algorithms

- The **medial axis** of a simple polygon.
  [Chin, Snoeyink, Wang, DCG 1999]

- **Expected** linear-time algorithm for the **farthest-segment** Voronoi diagram, after the sequence of faces at infinity is known.
  [Khramtcova, Papadopoulou, ISAAC 2015]

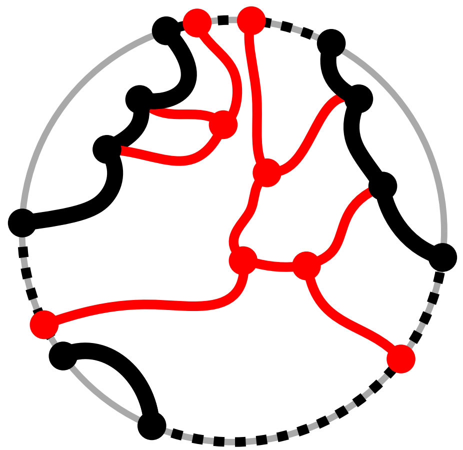
Our results



Introduce **Voronoi-like diagrams**

– **relaxed** version of a Voronoi diagram (easier to compute)

Our results

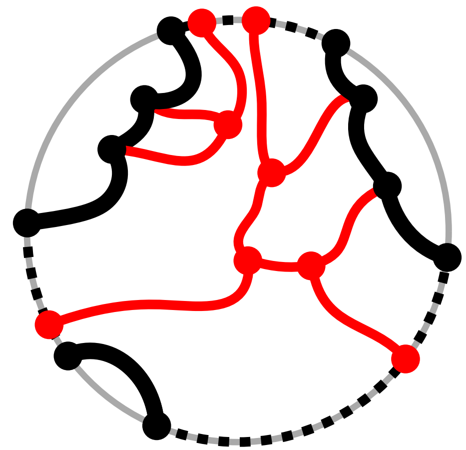


Introduce **Voronoi-like diagrams**

– **relaxed** version of a Voronoi diagram (easier to compute)

A simple, **randomized incremental** algorithm for updating abstract Voronoi diagrams after **deletion** of one site **in expected linear time**.

Our results



Introduce **Voronoi-like diagrams**

– **relaxed** version of a Voronoi diagram
(easier to compute)

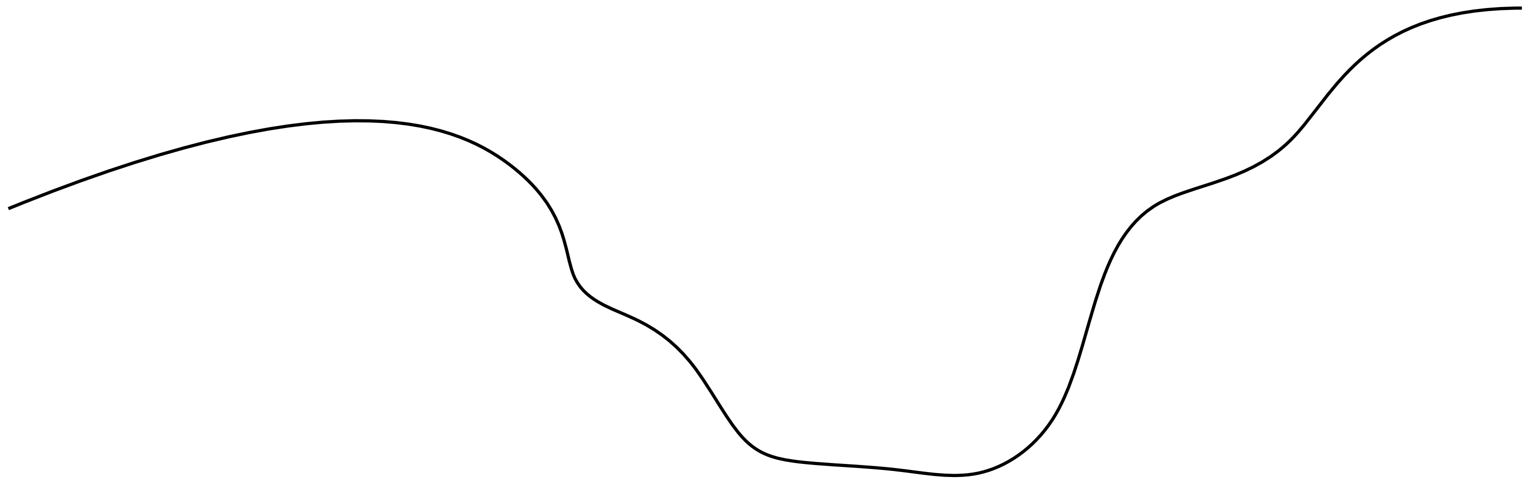
A simple, **randomized incremental** algorithm for updating abstract Voronoi diagrams after **deletion** of one site **in expected linear time**.

– adapt to the **farthest abstract Voronoi diagram**,
after the sequence of its faces at infinity is known.

Overview

- Define abstract Voronoi diagrams (AVDs).
- Define **Voronoi-like diagrams**.
- Properties of Voronoi-like diagrams.
- Define an **insertion** operation on Voronoi-like diagrams.
- Sketch a randomized incremental algorithm.

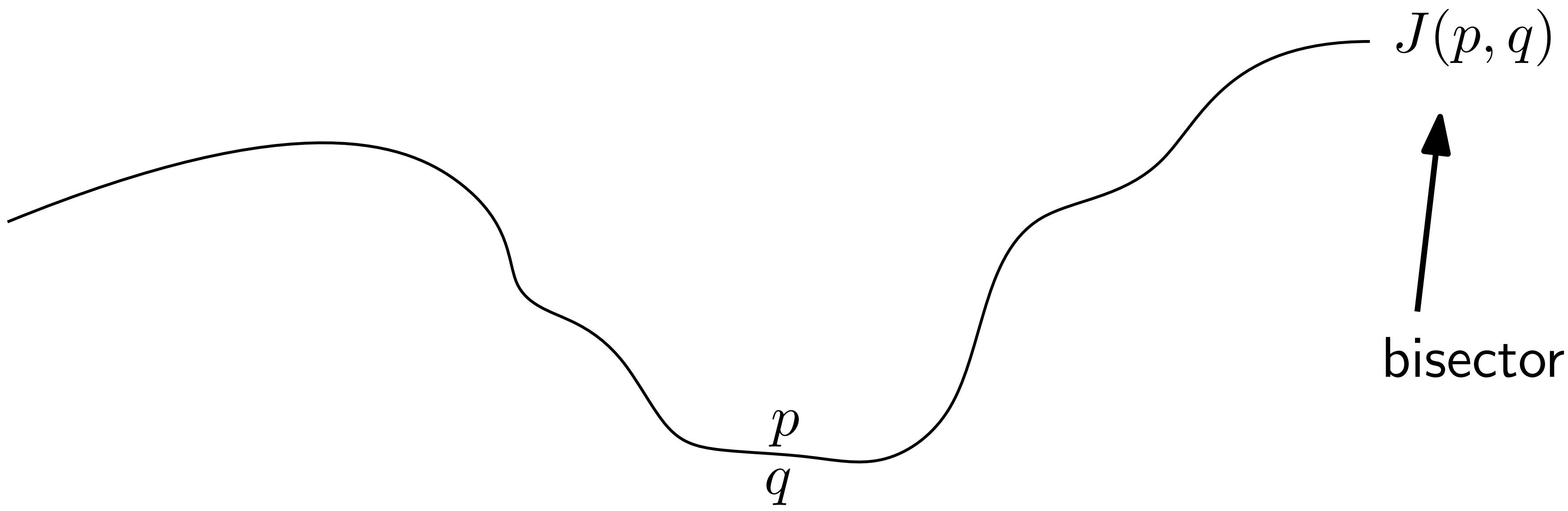
Abstract Voronoi diagrams



Rolf Klein. *Concrete and Abstract Voronoi Diagrams*. 1989.

Abstract Voronoi diagrams

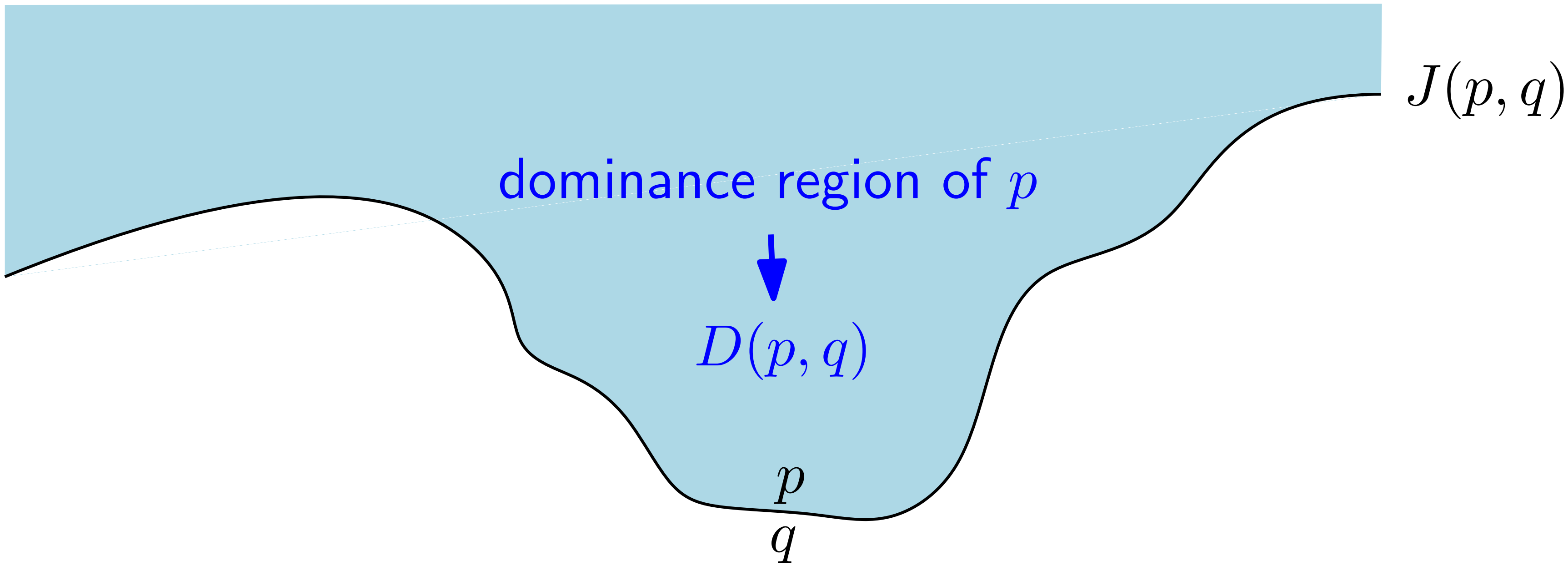
S abstract sites, $n = |S|$.



Rolf Klein. *Concrete and Abstract Voronoi Diagrams*. 1989.

Abstract Voronoi diagrams

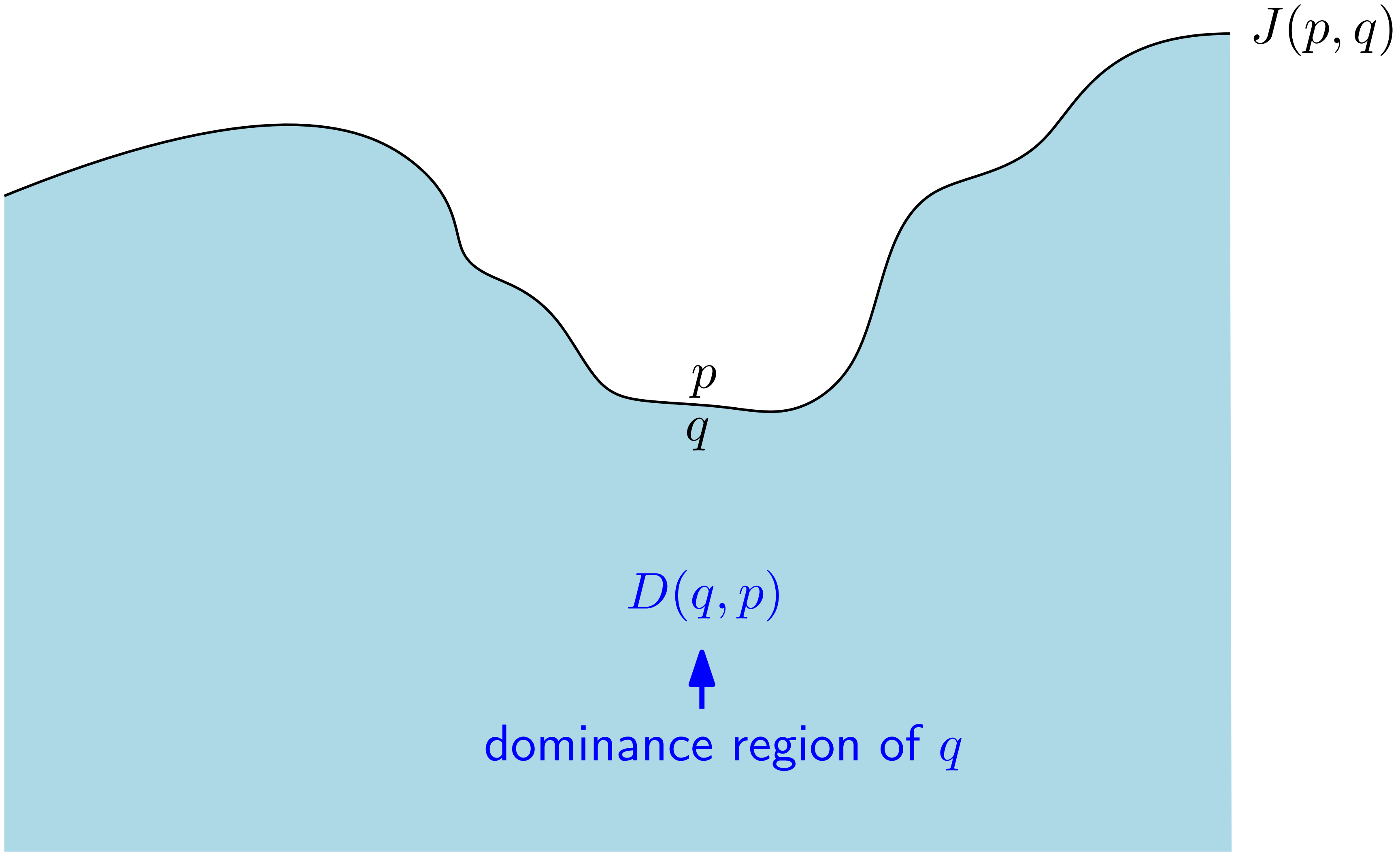
S abstract sites, $n = |S|$.



Rolf Klein. *Concrete and Abstract Voronoi Diagrams*. 1989.

Abstract Voronoi diagrams

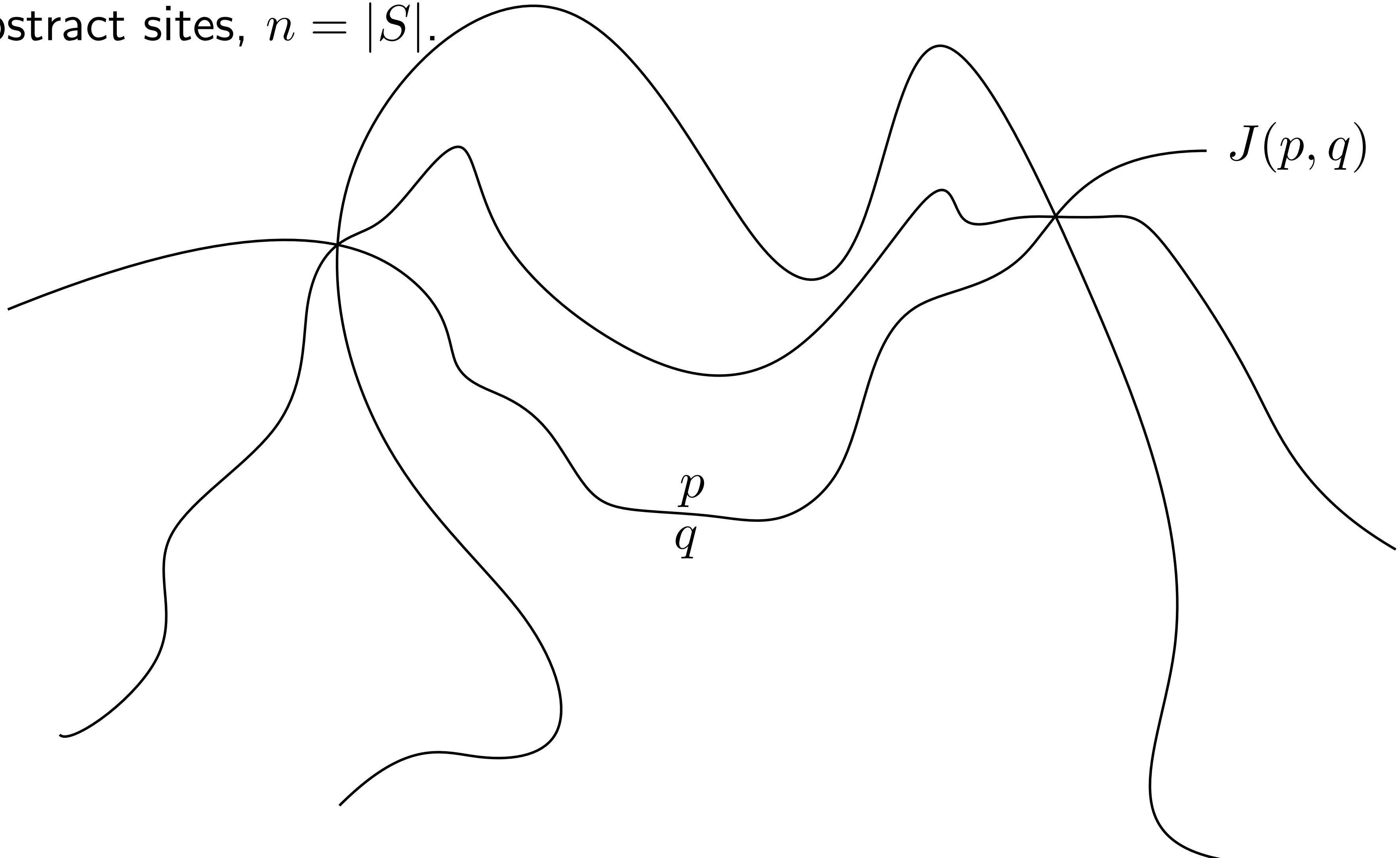
S abstract sites, $n = |S|$.



Rolf Klein. *Concrete and Abstract Voronoi Diagrams*. 1989.

Abstract Voronoi diagrams

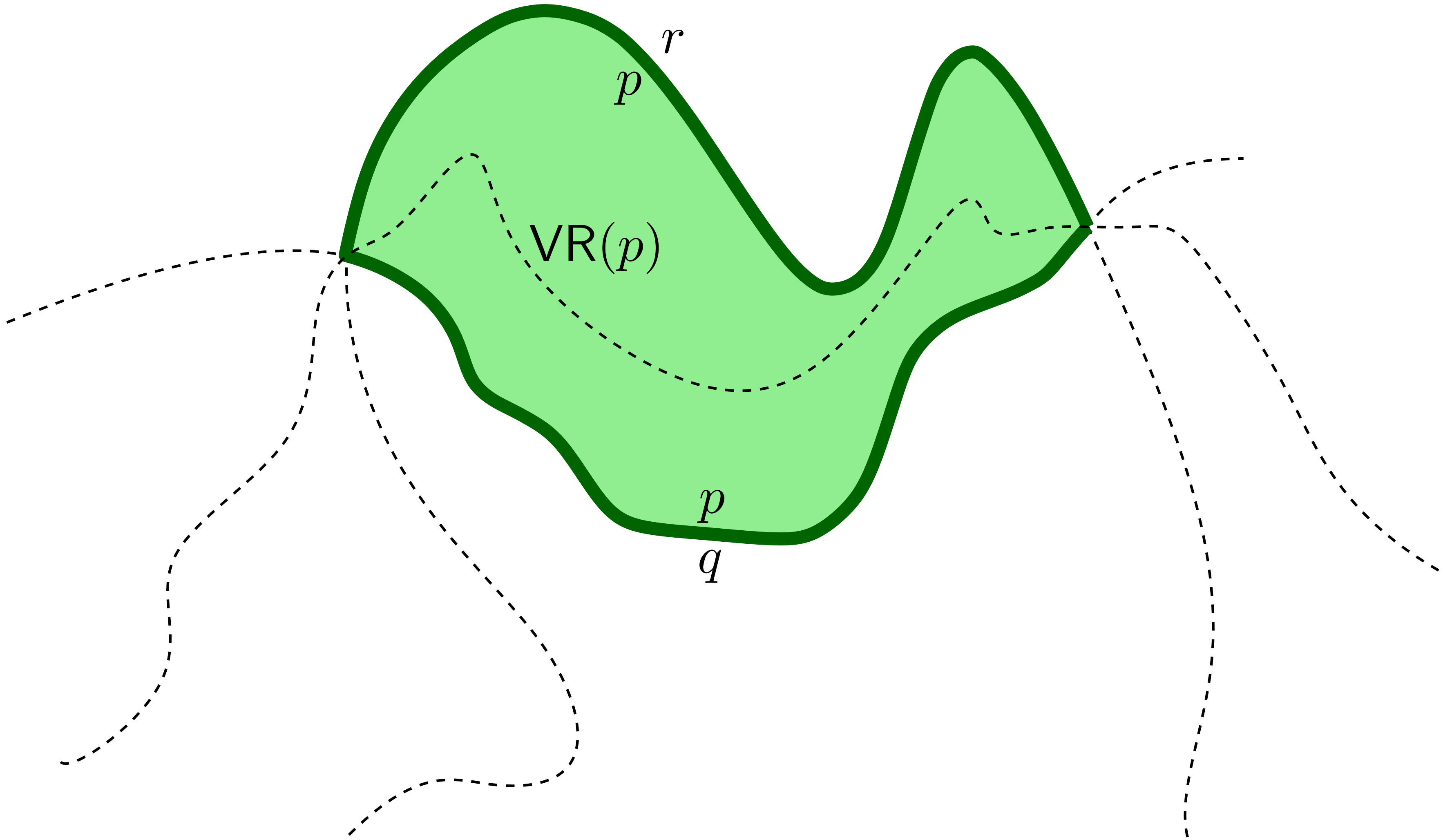
S abstract sites, $n = |S|$.



Given a set of bisectors $\mathcal{J} := \{J(p,q) : p \neq q \in S\}$.

Rolf Klein. *Concrete and Abstract Voronoi Diagrams*. 1989.

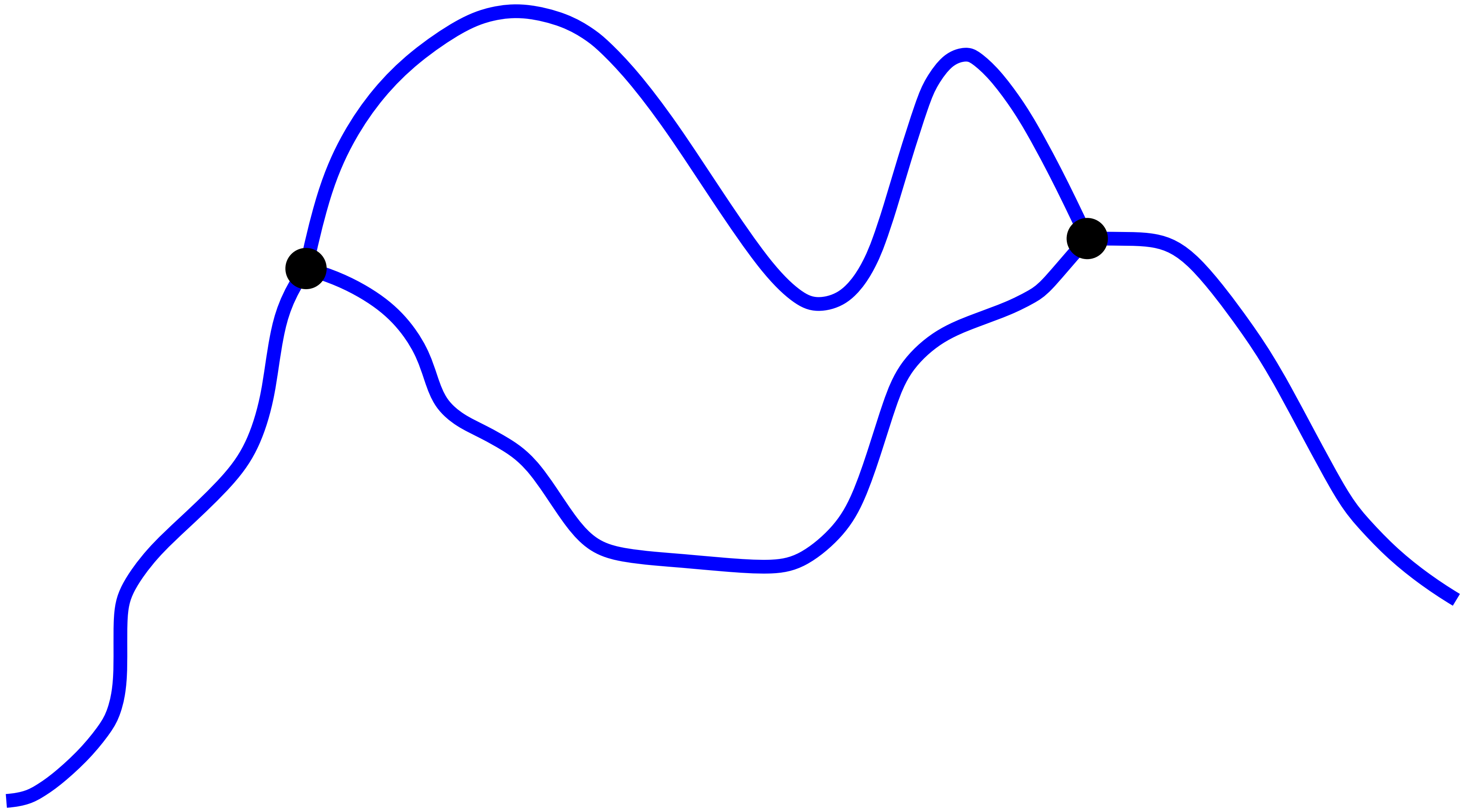
Abstract Voronoi diagrams



Voronoi region: $\boxed{\mathrm{VR}(p) = \bigcap_{q \in S \setminus \{p\}} D(p,q)}$

Rolf Klein. *Concrete and Abstract Voronoi Diagrams*. 1989.

Abstract Voronoi diagrams



Voronoi diagram: $\boxed{\mathcal{V}(S) = \mathbb{R}^2 \setminus \bigcup_{p \in S} \mathsf{VR}(p, S)}$

Rolf Klein. *Concrete and Abstract Voronoi Diagrams*. 1989.

Admissible bisector system

Given $\mathcal{J} := \{J(p,q) : p \neq q \in S\}$.

For every $S' \subseteq S$:

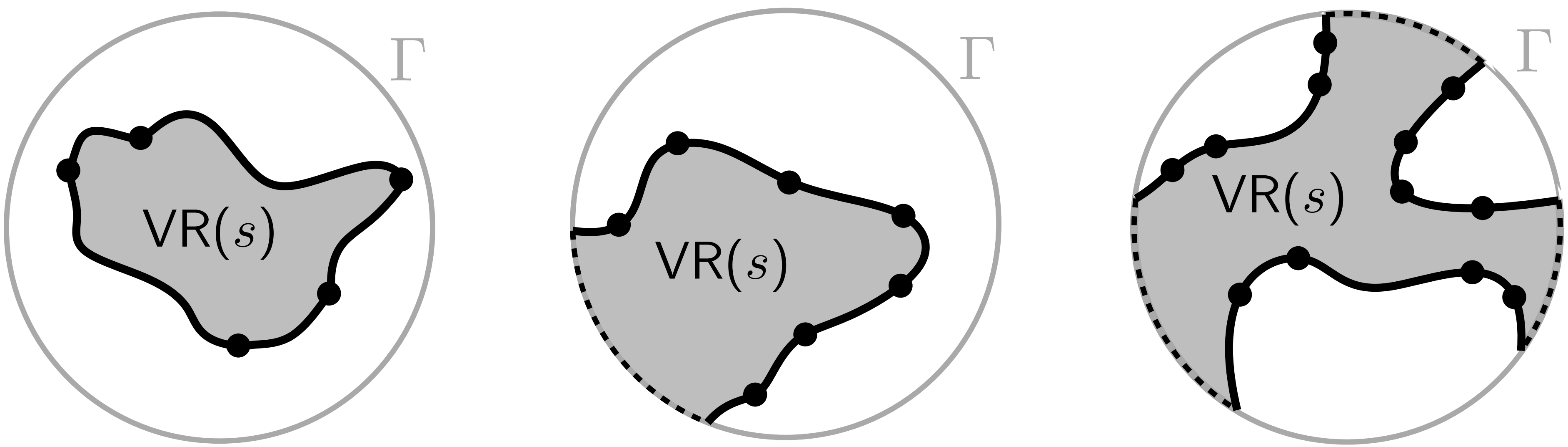
- (A1) Voronoi regions are non-empty and connected.
- (A2) Voronoi regions cover the plane.
- (A3) Bisectors are unbounded Jordan curves.
- (A4) Transversal and finite # intersections.

Rolf Klein. *Concrete and Abstract Voronoi Diagrams*. 1989.

Admissible bisector system

- For simplicity we always assume a big circle Γ, containing all intersections.
  We restrict all computations in the interior of Γ.

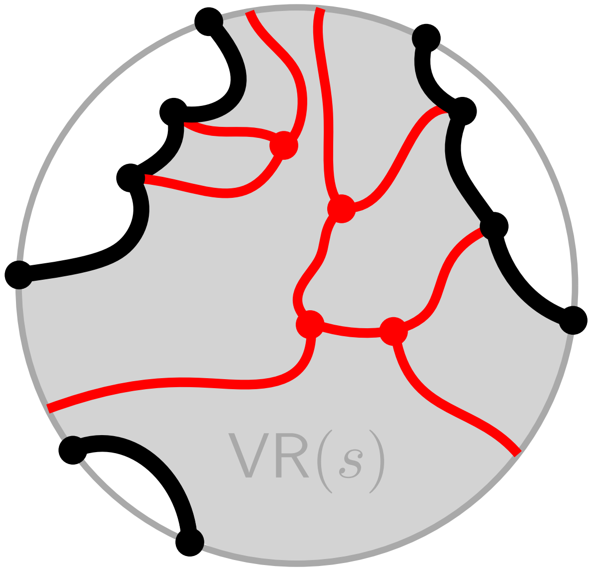
- VR(s) can be bounded, unbounded, and have several openings to infinity (Γ-arcs).



Rolf Klein. *Concrete and Abstract Voronoi Diagrams*. 1989.

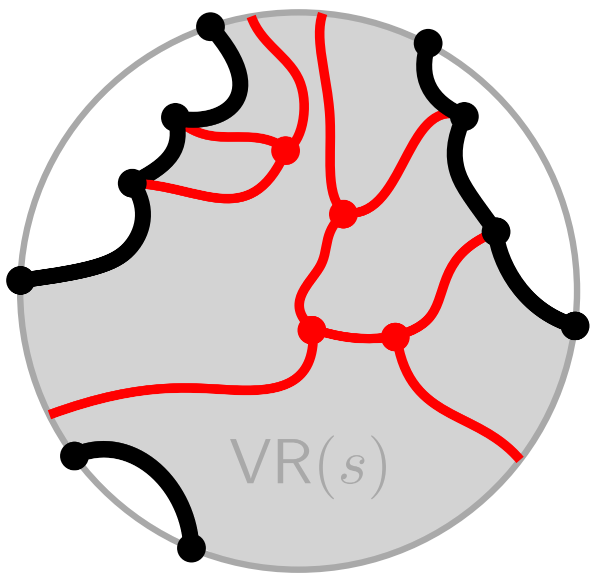
Site deletion

Problem: Compute $\mathcal{V}(S \setminus s) \cap \mathsf{VR}(s)$ (within $\mathsf{VR}(s)$).



Site deletion

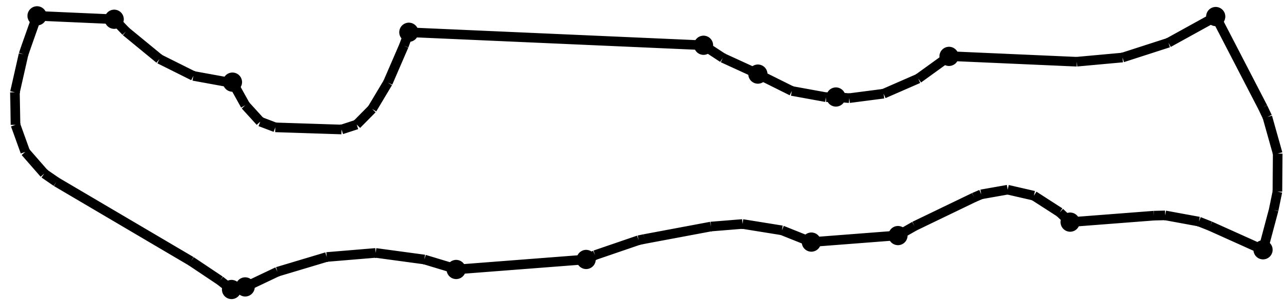
Problem: Compute $\mathcal{V}(S \setminus s) \cap \mathsf{VR}(s)$ (within $\mathsf{VR}(s)$).



Lemma: $\mathcal{V}(S \setminus s) \cap \mathsf{VR}(s)$ is a **forest** with **one face per Voronoi edge** of $\partial\mathsf{VR}(s)$.

Voronoi regions of arcs

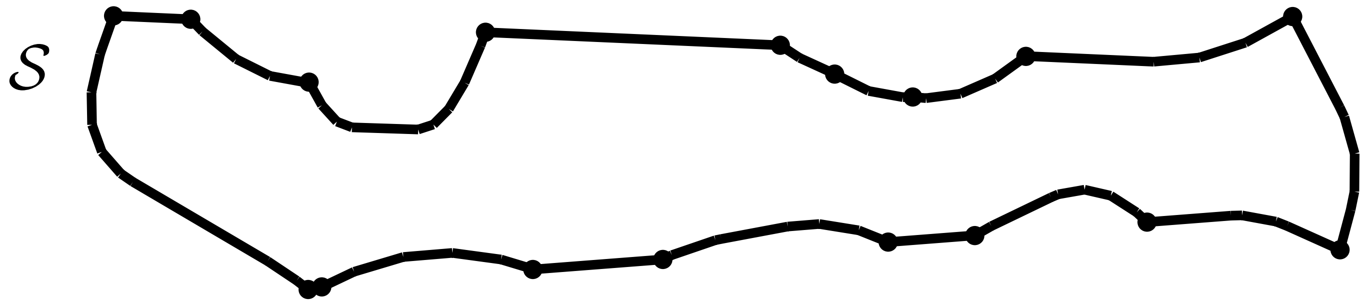
Idea: Treat the boundary arcs (Voronoi edges) of $\mathrm{VR}(s)$ as **sites**.



Voronoi regions of arcs

Idea: Treat the boundary arcs (Voronoi edges) of $\mathrm{VR}(s)$ as **sites**.

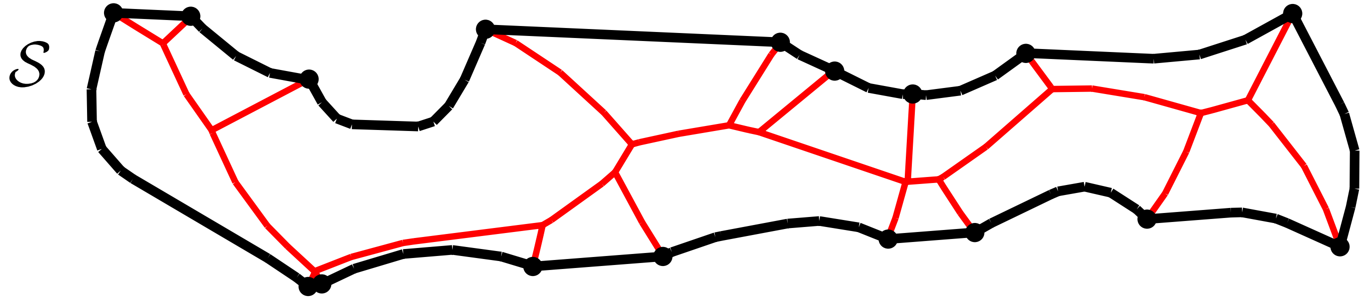
- Denote these arcs by $\mathcal{S}$.



Voronoi regions of arcs

Idea: Treat the boundary arcs (Voronoi edges) of VR(s) as **sites**.

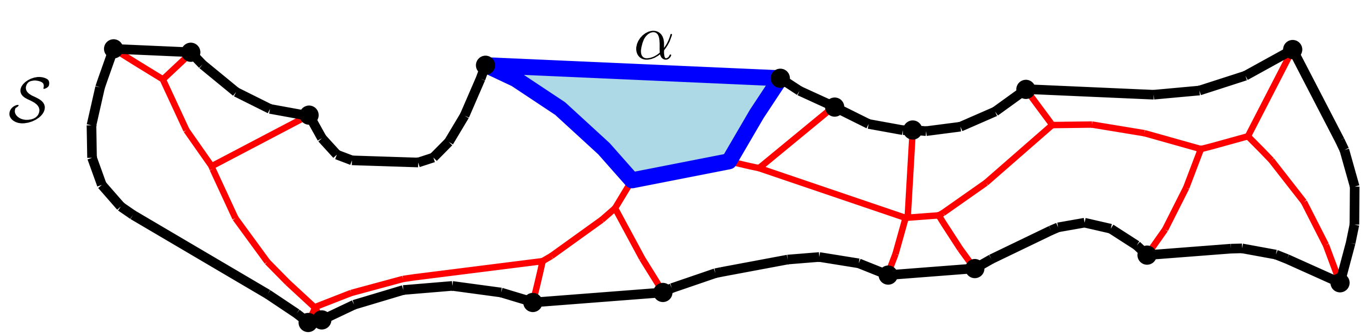
- Denote these arcs by $\mathcal{S}$.
- Voronoi diagram of $\mathcal{S}$ is $\mathcal{V}(\mathcal{S}) = \mathcal{V}(S \setminus s) \cap \text{VR}(s)$.



Voronoi regions of arcs

Idea: Treat the boundary arcs (Voronoi edges) of VR(s) as **sites**.

- Denote these arcs by $\mathcal{S}$.
- Voronoi diagram of $\mathcal{S}$ is $\mathcal{V}(\mathcal{S}) = \mathcal{V}(S \setminus s) \cap \mathrm{VR}(s)$.
- For an arc $\alpha \in \mathcal{S}$, assign VR(α) = face of $\mathcal{V}(\mathcal{S})$ incident to α.

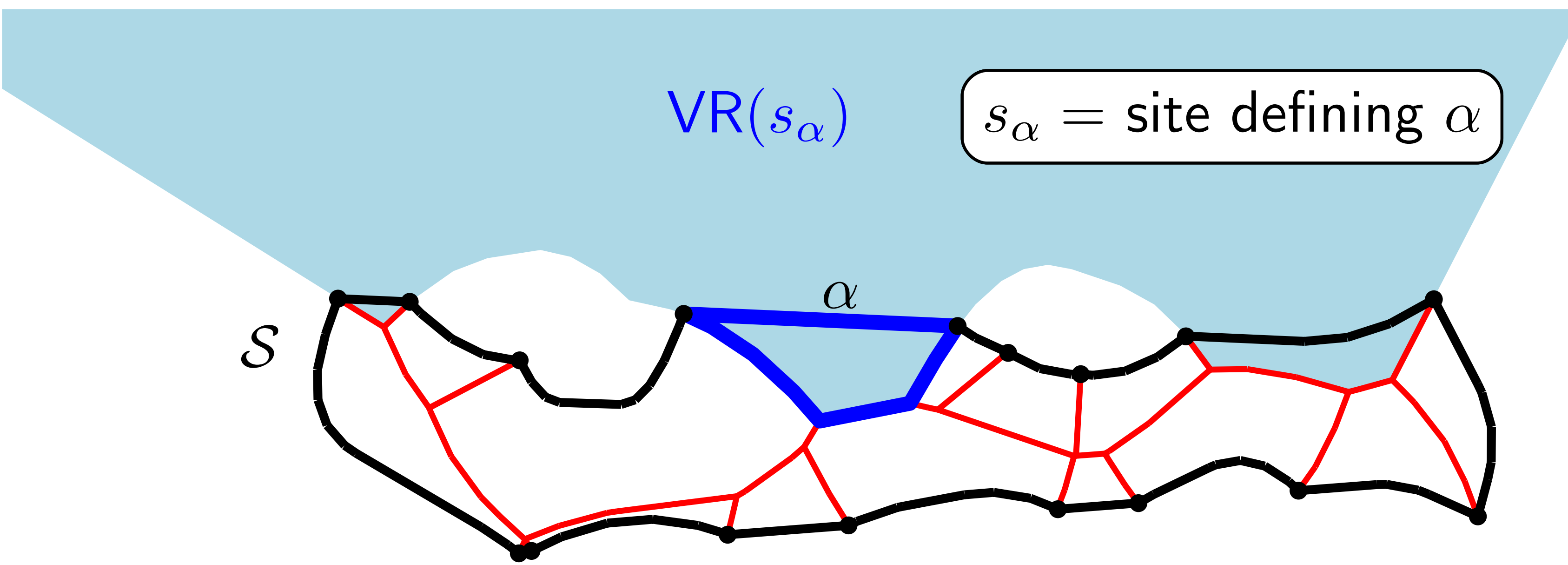


Voronoi regions of arcs

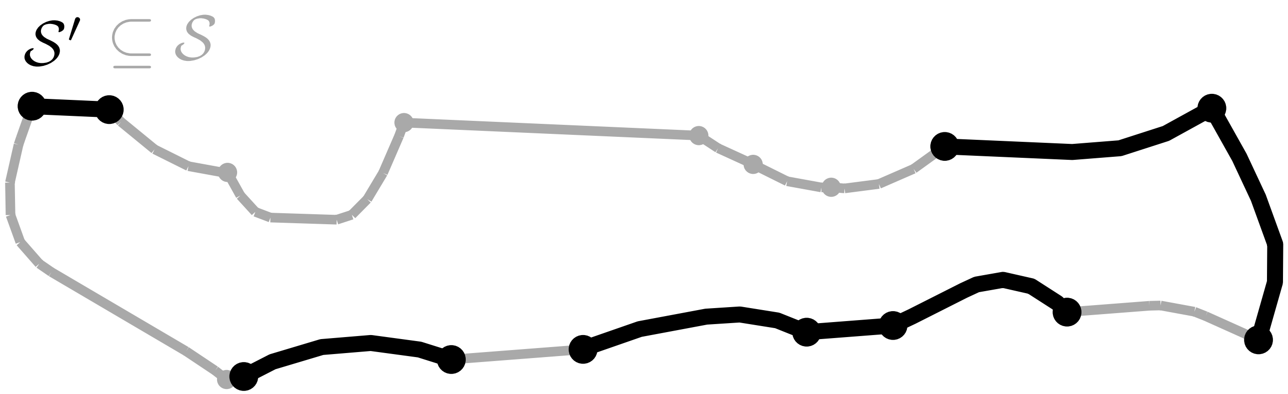
Idea: Treat the boundary arcs (Voronoi edges) of VR(s) as **sites**.

- Denote these arcs by $\mathcal{S}$.
- Voronoi diagram of $\mathcal{S}$ is $\mathcal{V}(\mathcal{S}) = \mathcal{V}(S \setminus s) \cap \mathrm{VR}(s)$.
- For an arc $\alpha \in \mathcal{S}$, assign VR(α) = face of $\mathcal{V}(\mathcal{S})$ incident to α.

Site s_α can have $\Theta(n)$ faces within VR(s).
Treat each face independently (different arc).

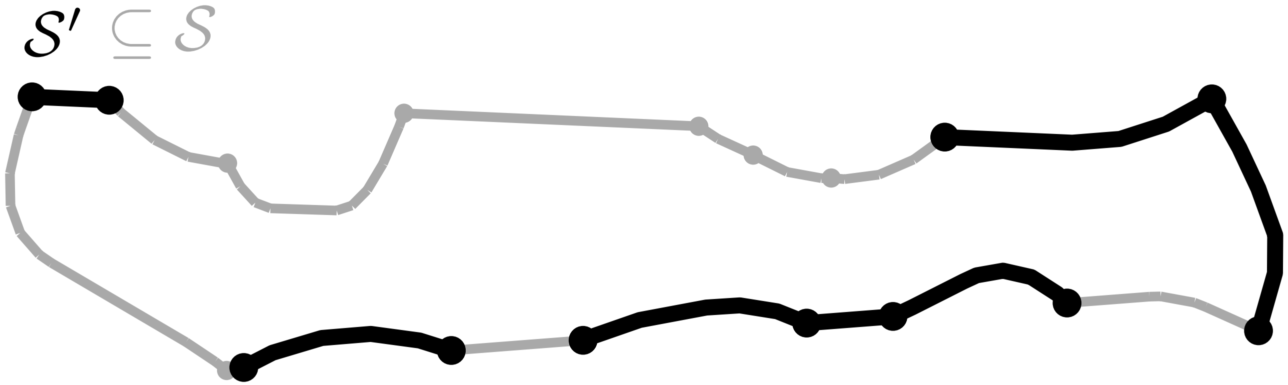


Wish: Voronoi diagram of a **subset of arcs** $\mathcal{S}' \subseteq \mathcal{S}$.
But that does not exist. ↯



Wish: Voronoi diagram of a **subset of arcs** $\mathcal{S}' \subseteq \mathcal{S}$.
But that does not exist. ↯

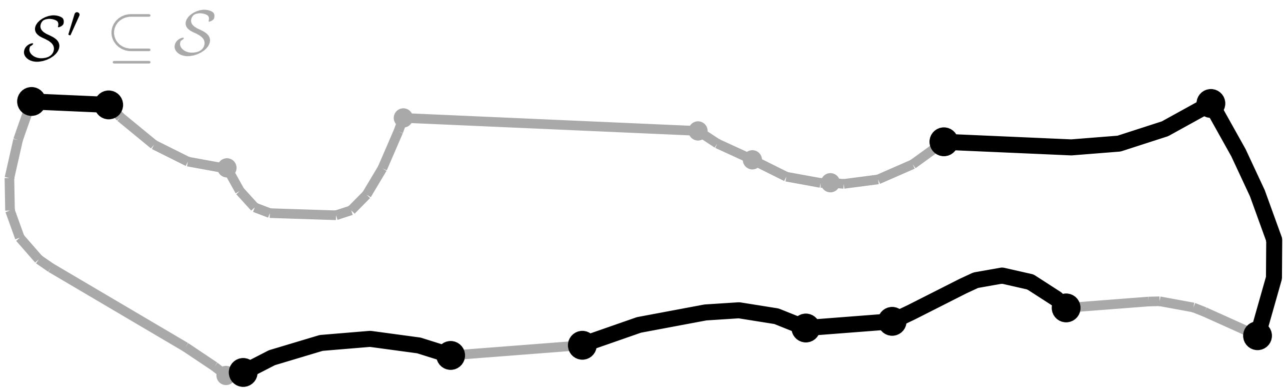
Instead we define a **Voronoi-like diagram**
for a subset of arcs $\mathcal{S}' \subseteq \mathcal{S}$.



Wish: Voronoi diagram of a **subset of arcs** $\mathcal{S}' \subseteq \mathcal{S}$.
But that does not exist. ↯

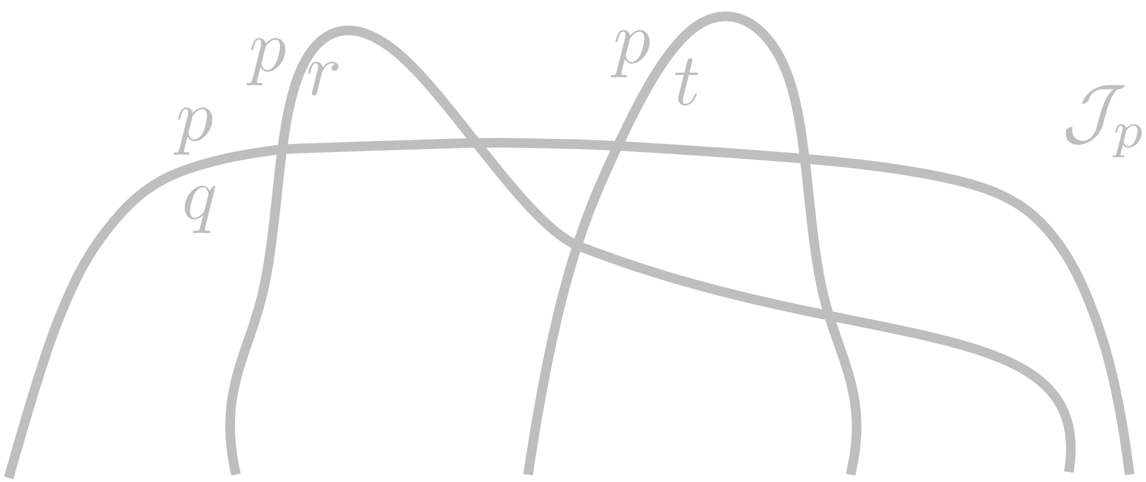
Instead we define a **Voronoi-like diagram**
for a subset of arcs $\mathcal{S}' \subseteq \mathcal{S}$.

Next: Definitions...



p-monotone paths

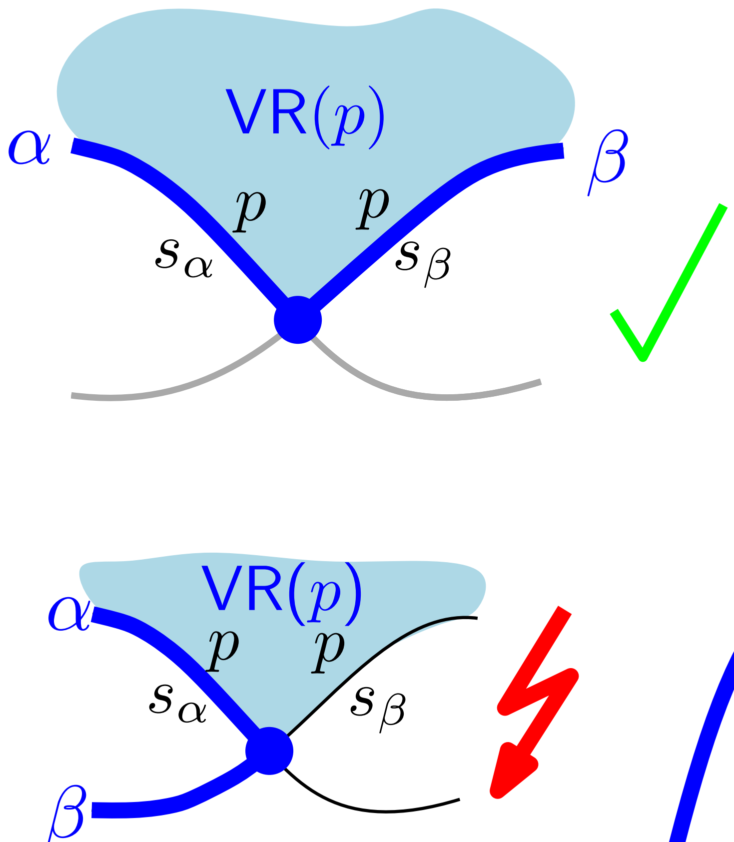
Let $p \in S$ be a site. Let $\mathcal{J}_p$ be the arrangement of all p-related bisectors.

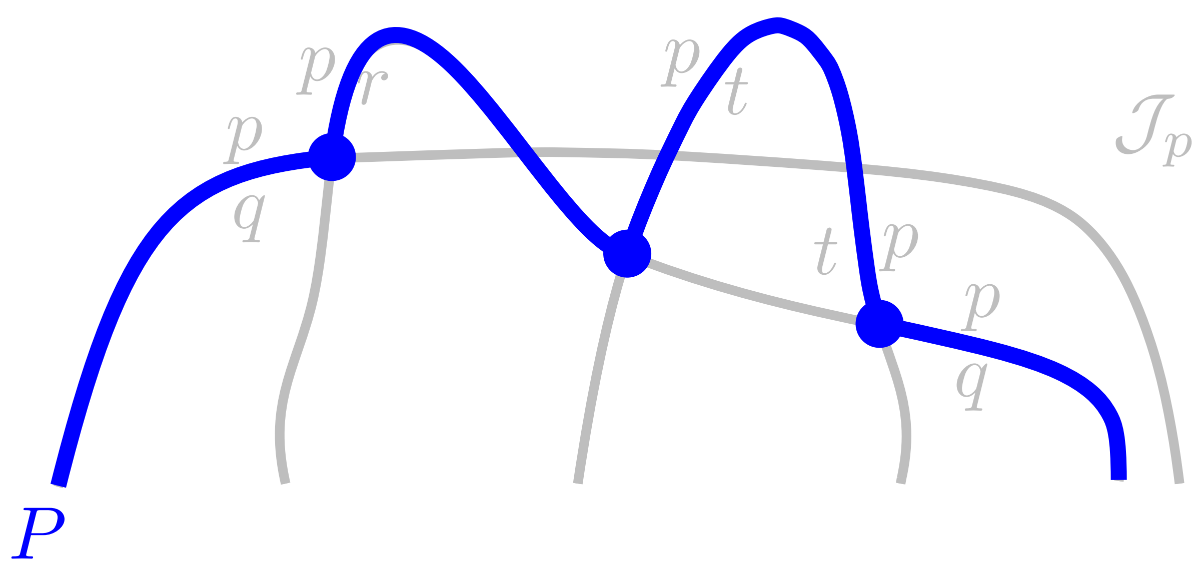


p-monotone paths

Let $p \in S$ be a site. Let $\mathcal{J}_p$ be the arrangement of all p-related bisectors.

A path in the arrangement $\mathcal{J}_p$ is p-**monotone**, if any two adjacent edges α, β coincide locally with the Voronoi edges of $\mathsf{VR}(p, \{p, s_\alpha, s_\beta\})$.



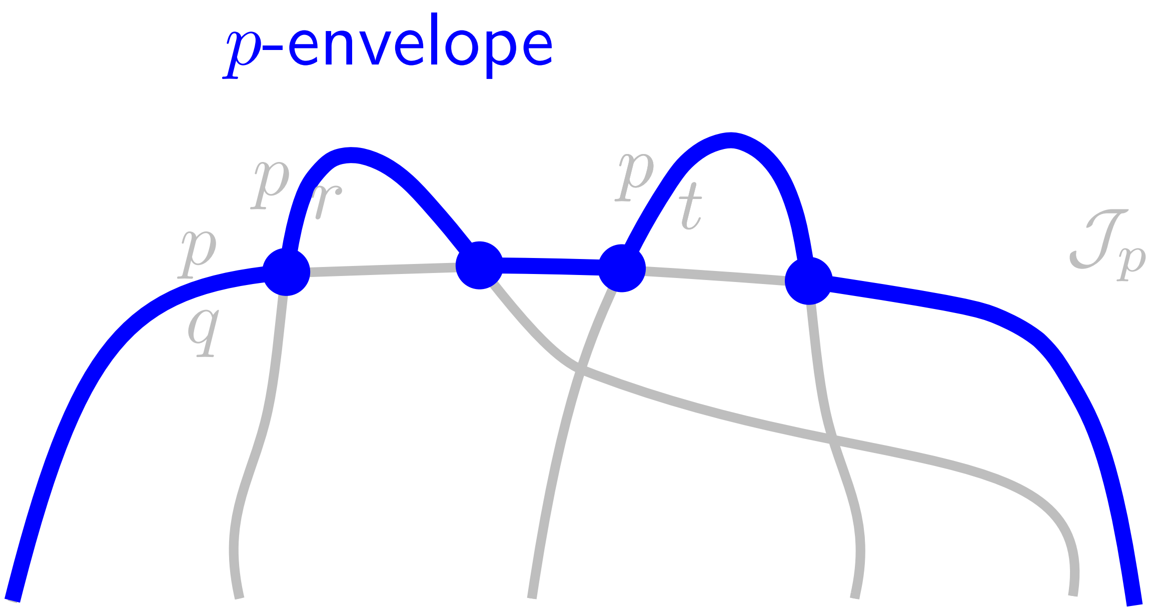


p-monotone paths

Let $p \in S$ be a site. Let $\mathcal{J}_p$ be the arrangement of all p-related bisectors.

A path in the arrangement $\mathcal{J}_p$ is p-**monotone**, if any two adjacent edges α, β coincide locally with the Voronoi edges of $\mathrm{VR}(p, \{p, s_\alpha, s_\beta\})$.

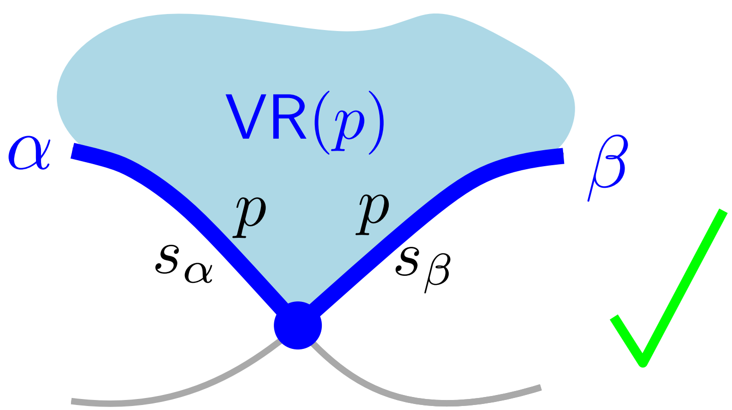
A path in $\mathcal{J}_p$ is the p-**envelope**, if it is the boundary of $\mathrm{VR}(p)$

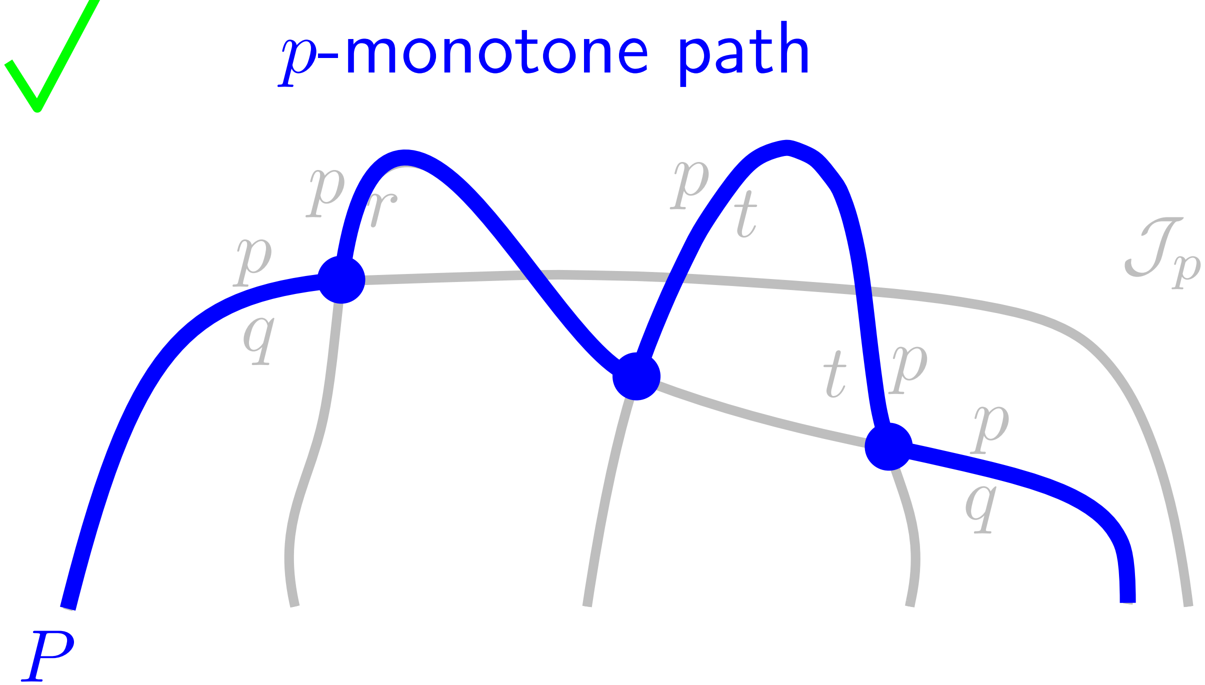


p-monotone paths

Let $p \in S$ be a site. Let $\mathcal{J}_p$ be the arrangement of all p-related bisectors.

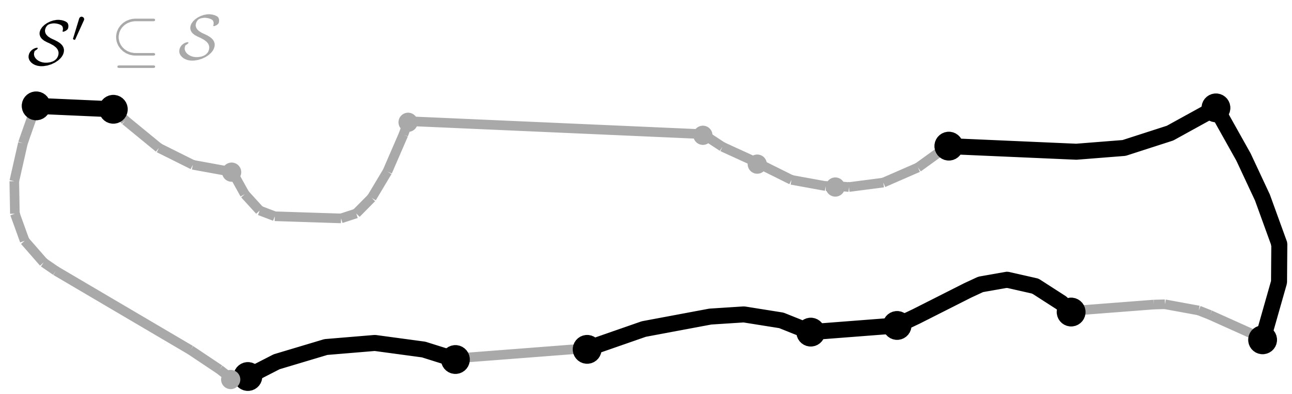
A path in the arrangement $\mathcal{J}_p$ is p-**monotone**, if any two adjacent edges α, β coincide locally with the Voronoi edges of $\mathrm{VR}(p, \{p, s_\alpha, s_\beta\})$.





Boundary curve

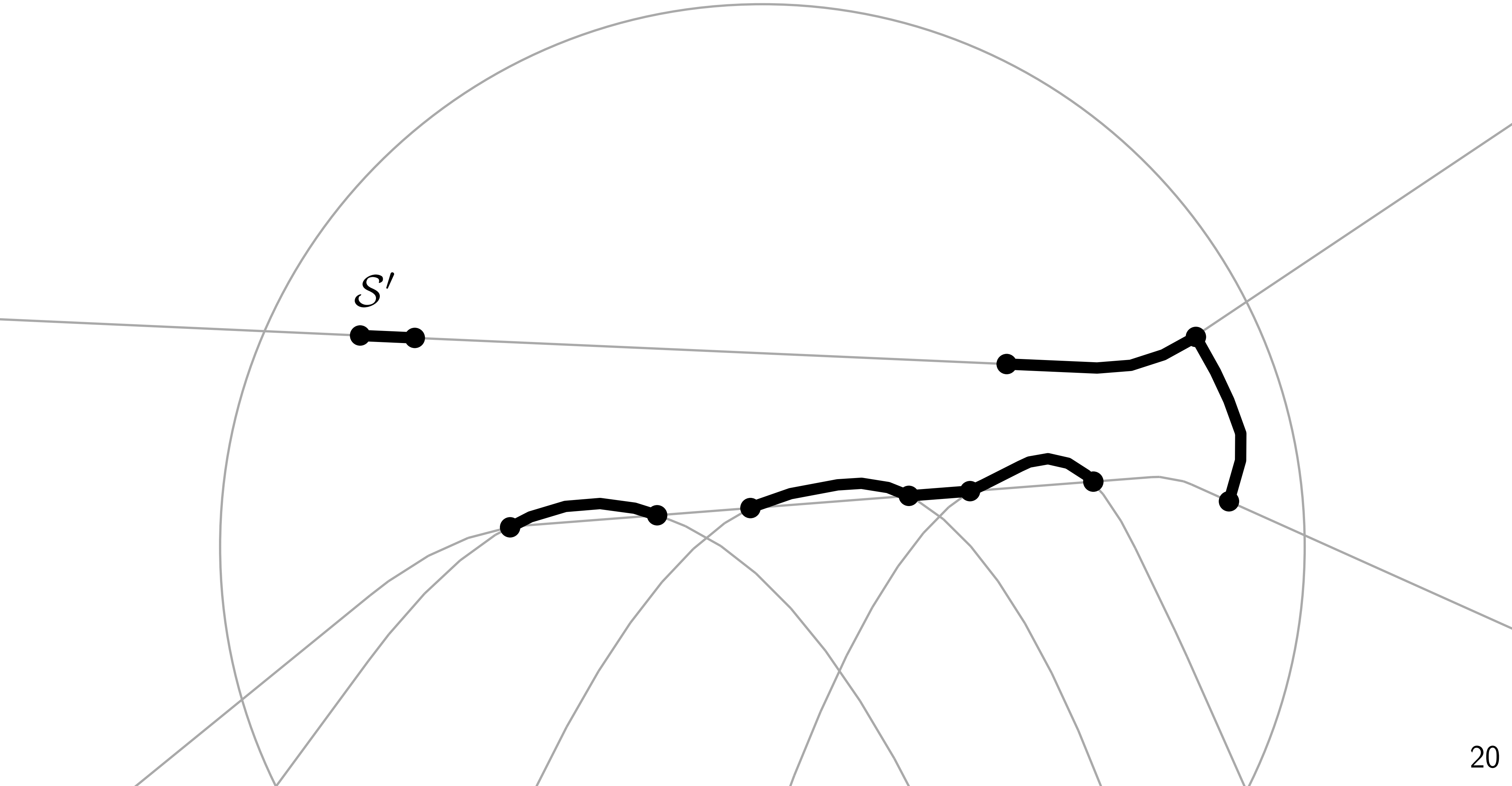
Let $\mathcal{S}' \subseteq \mathcal{S} =$ boundary arcs (Voronoi edges) along $\partial\mathrm{VR}(s)$.



Boundary curve

Let $\mathcal{S}' \subseteq \mathcal{S}$ = boundary arcs (Voronoi edges) along ∂VR(s).

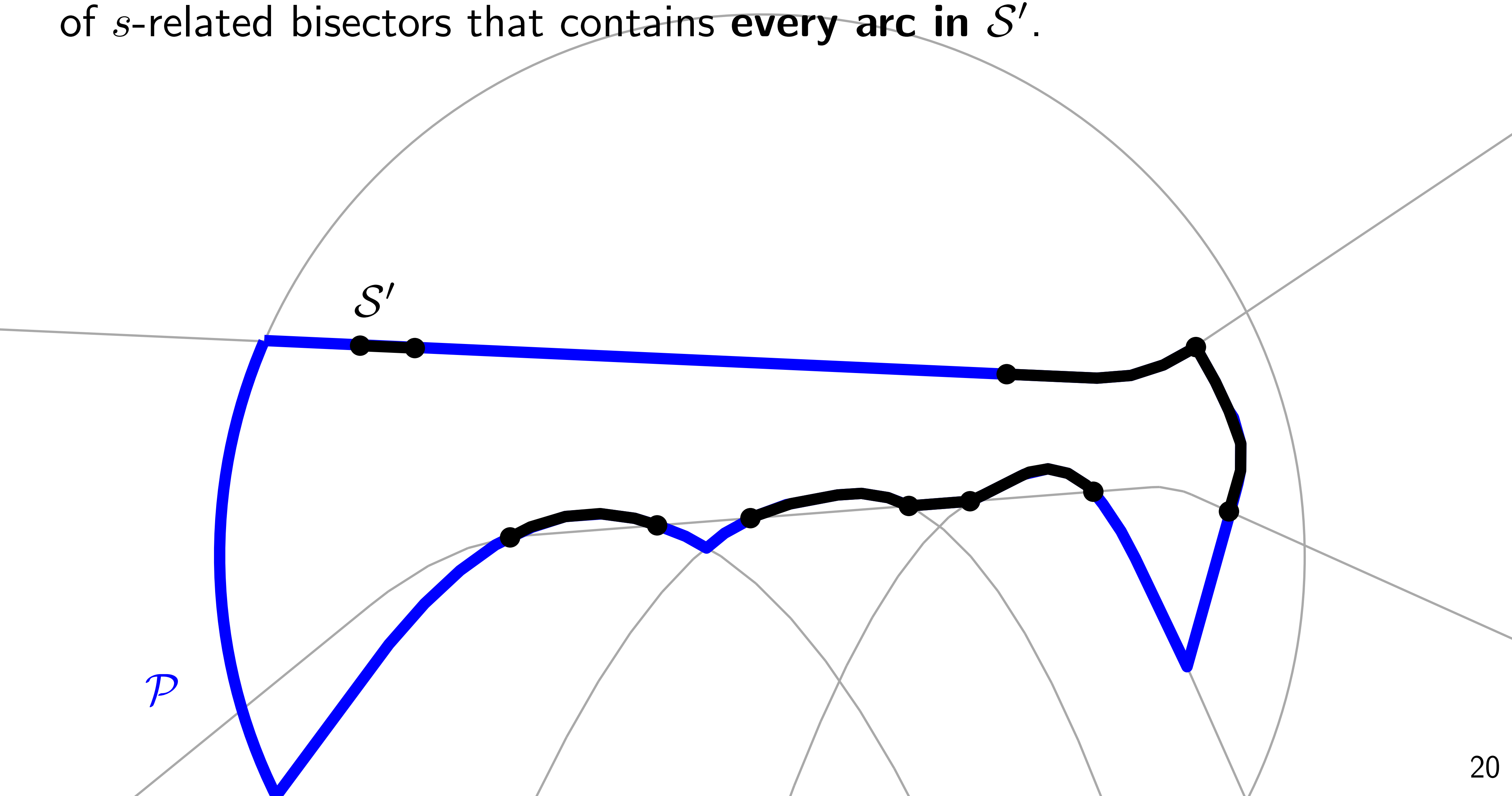
Consider the arrangement of all s-related bisectors of arcs in $\mathcal{S}'$.



Boundary curve

Let $\mathcal{S}' \subseteq \mathcal{S}$ = boundary arcs (Voronoi edges) along ∂VR(s).

A **boundary curve** $\mathcal{P}$ for $\mathcal{S}'$ is an s-**monotone** path in the arrangement of s-related bisectors that contains **every arc in** $\mathcal{S}'$.

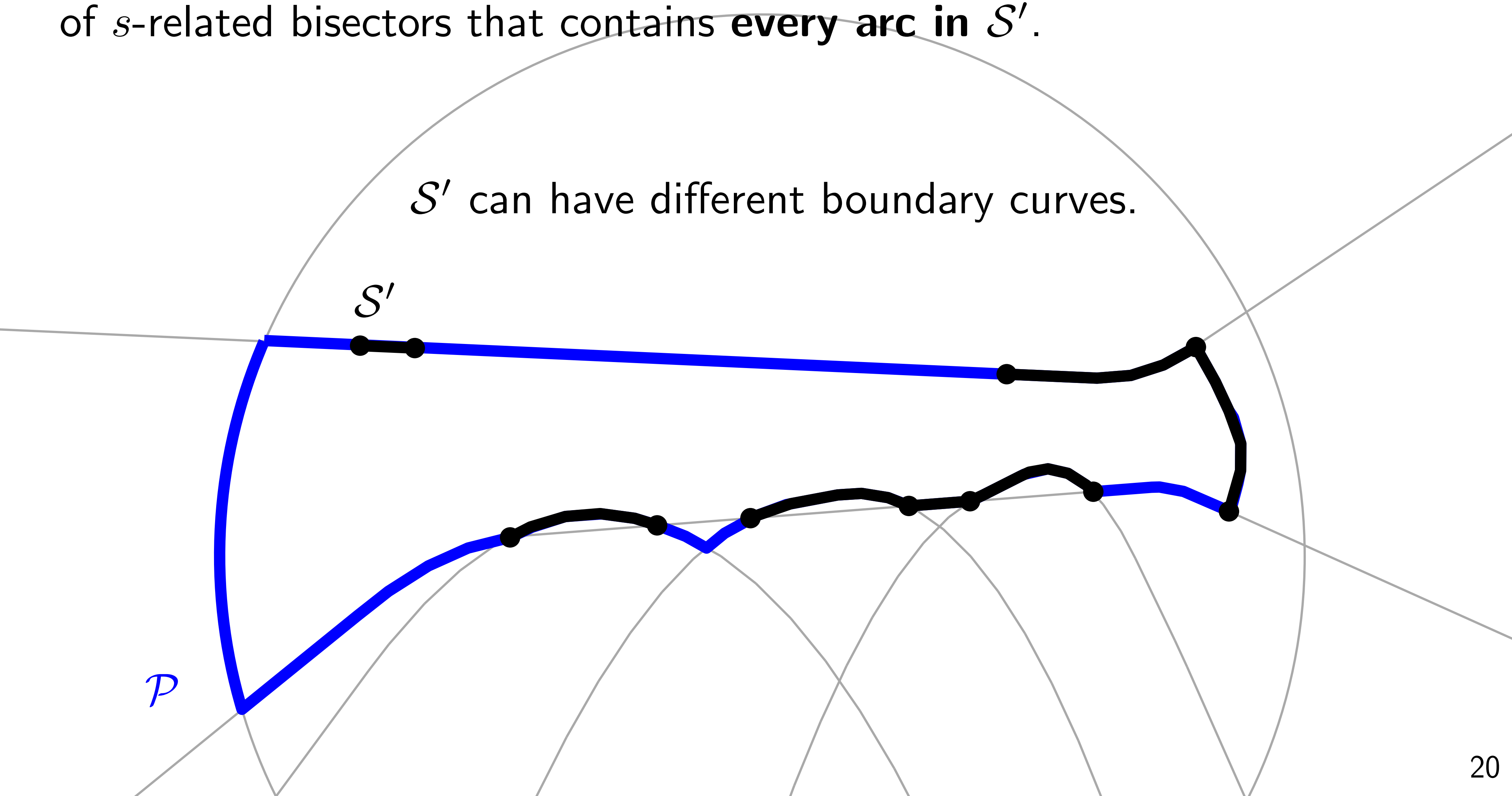


Boundary curve

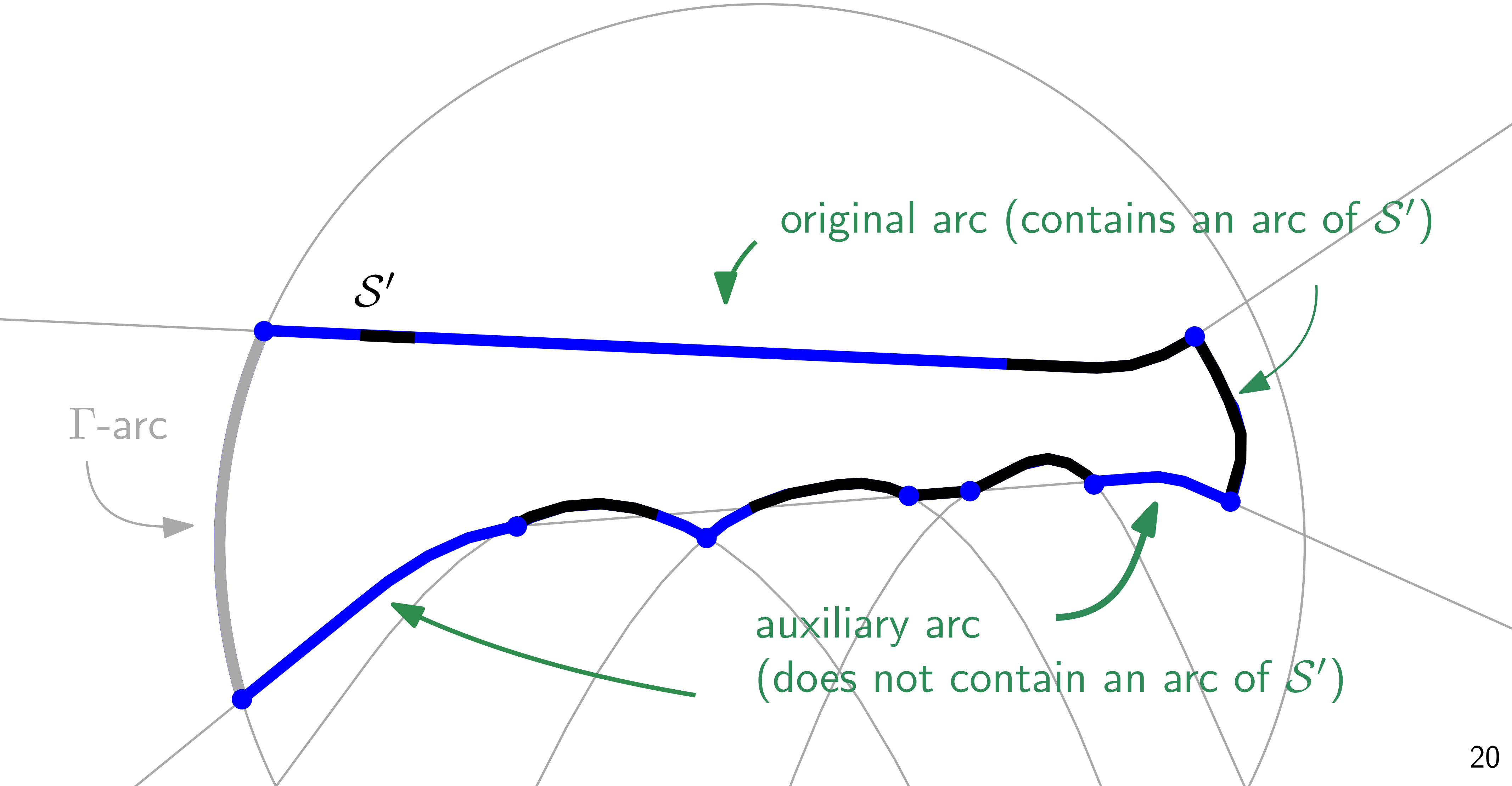
Let $\mathcal{S}' \subseteq \mathcal{S}$ = boundary arcs (Voronoi edges) along ∂VR(s).

A **boundary curve** $\mathcal{P}$ for $\mathcal{S}'$ is an s-**monotone** path in the arrangement of s-related bisectors that contains **every arc in** $\mathcal{S}'$.

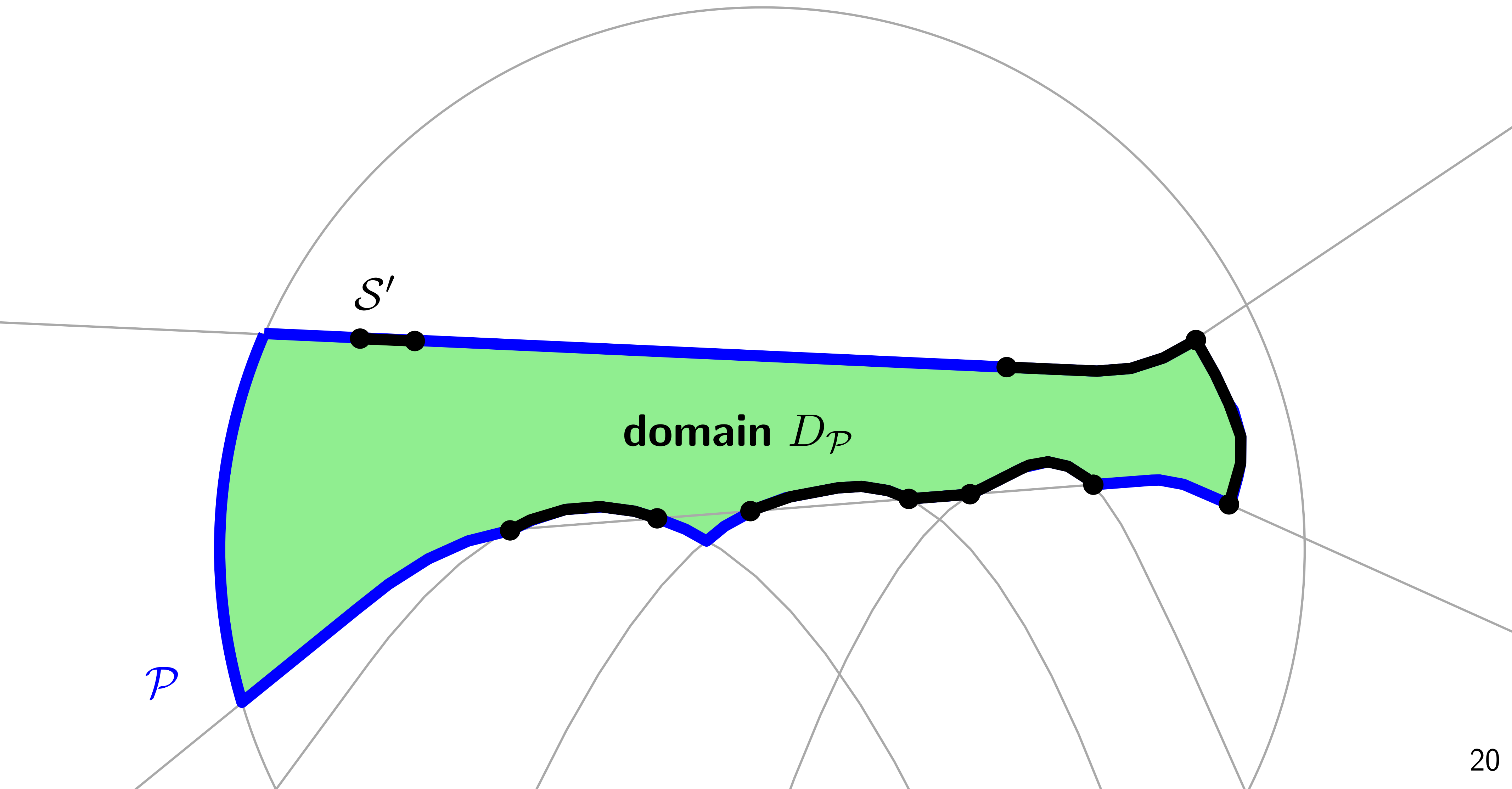
$\mathcal{S}'$ can have different boundary curves.



Boundary curve

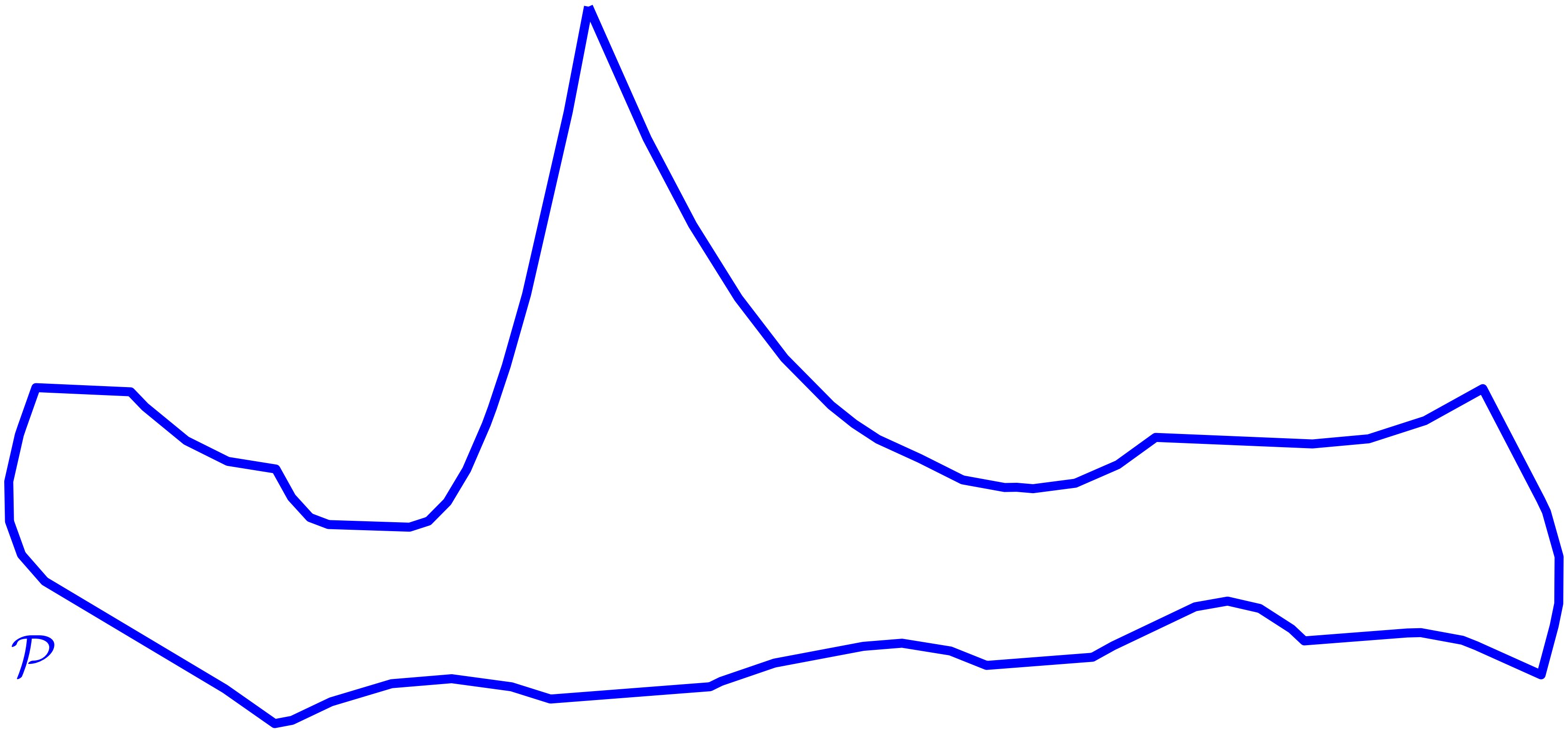


Boundary curve



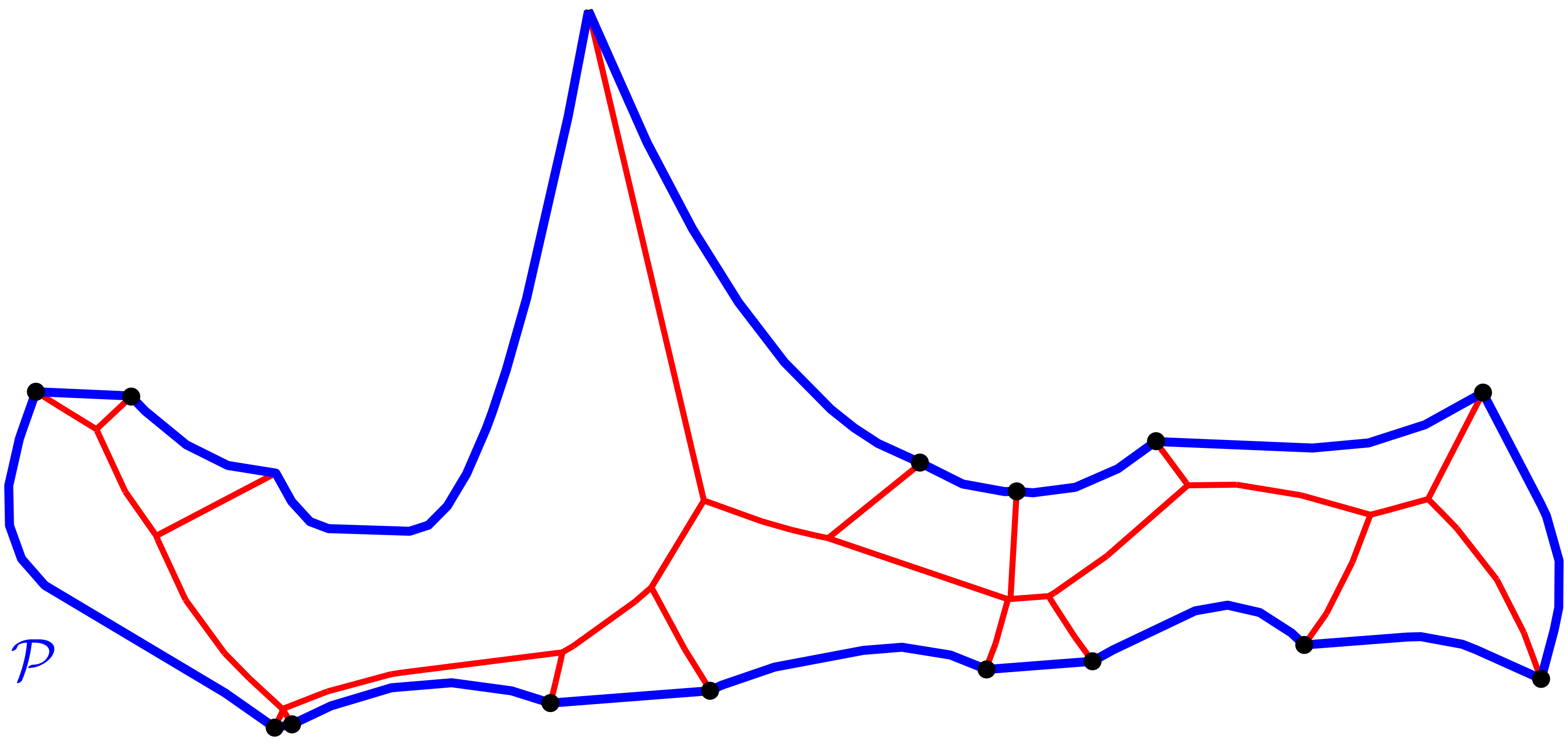
Voronoi-like diagram

Definition: Given a boundary curve $\mathcal{P}$, the Voronoi-like diagram $\mathcal{V}_l(\mathcal{P})$ is a subdivision of the domain $D_{\mathcal{P}}$ such that:



Voronoi-like diagram

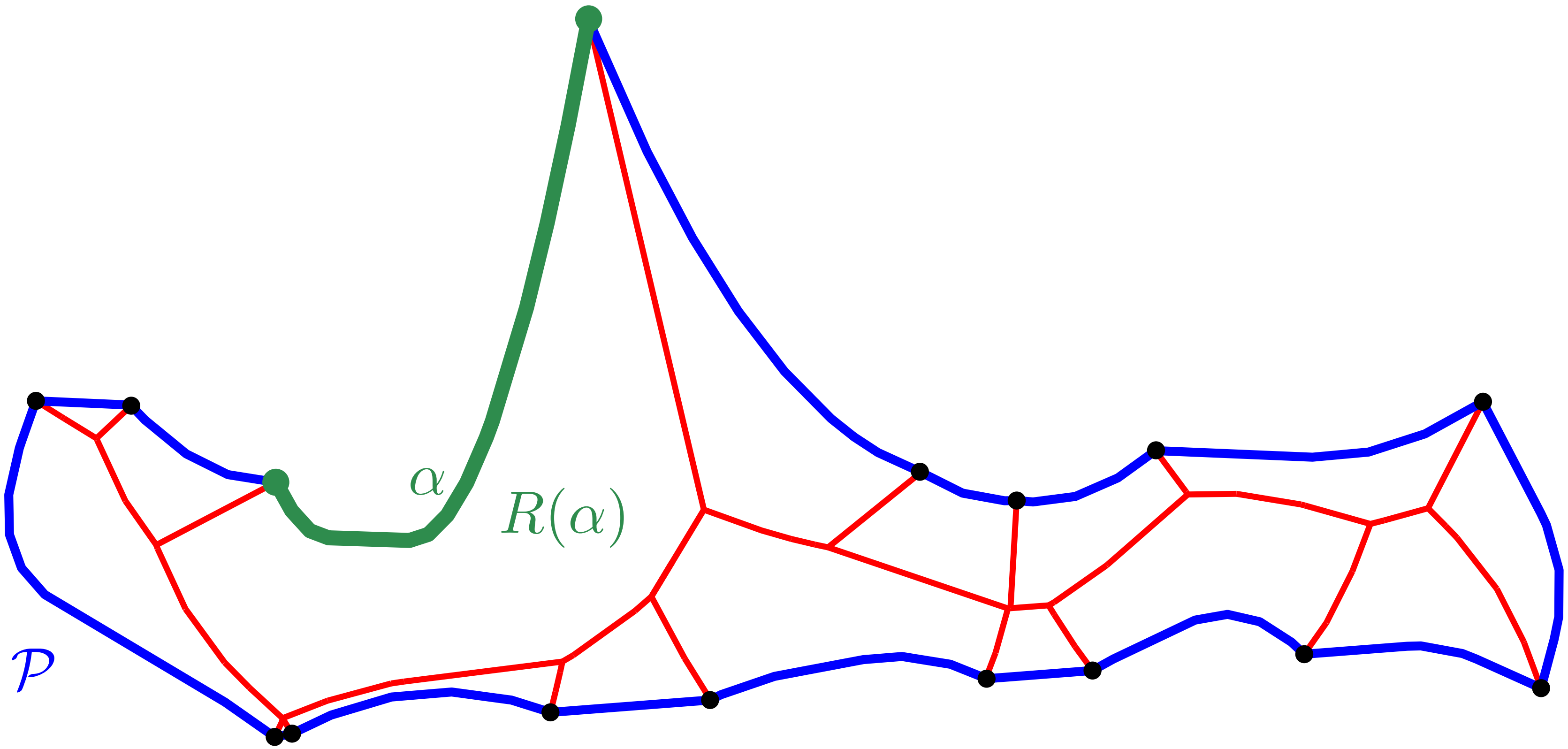
Definition: Given a boundary curve $\mathcal{P}$, the Voronoi-like diagram $\mathcal{V}_l(\mathcal{P})$ is a subdivision of the domain $D_{\mathcal{P}}$ such that:



Voronoi-like diagram

Definition: Given a boundary curve $\mathcal{P}$, the Voronoi-like diagram $\mathcal{V}_l(\mathcal{P})$ is a subdivision of the domain $D_{\mathcal{P}}$ such that:

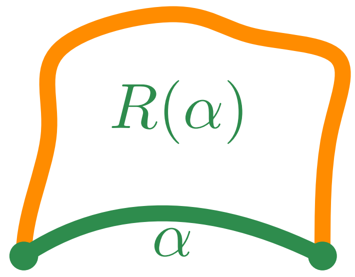
- Each boundary arc $\alpha \in \mathcal{P}$ has one region $R(\alpha)$.

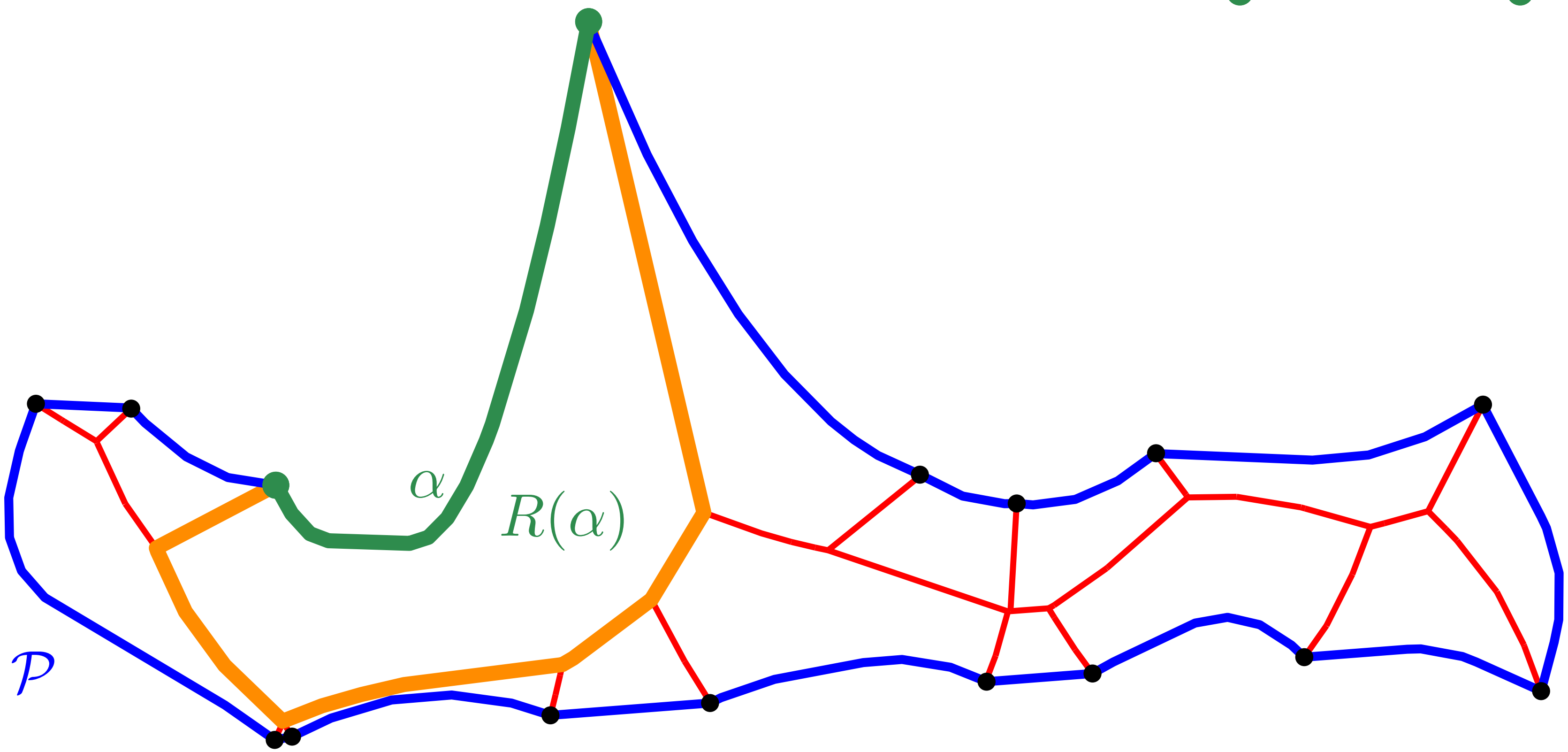


Voronoi-like diagram

Definition: Given a boundary curve $\mathcal{P}$, the Voronoi-like diagram $\mathcal{V}_l(\mathcal{P})$ is a subdivision of the domain $D_{\mathcal{P}}$ such that:

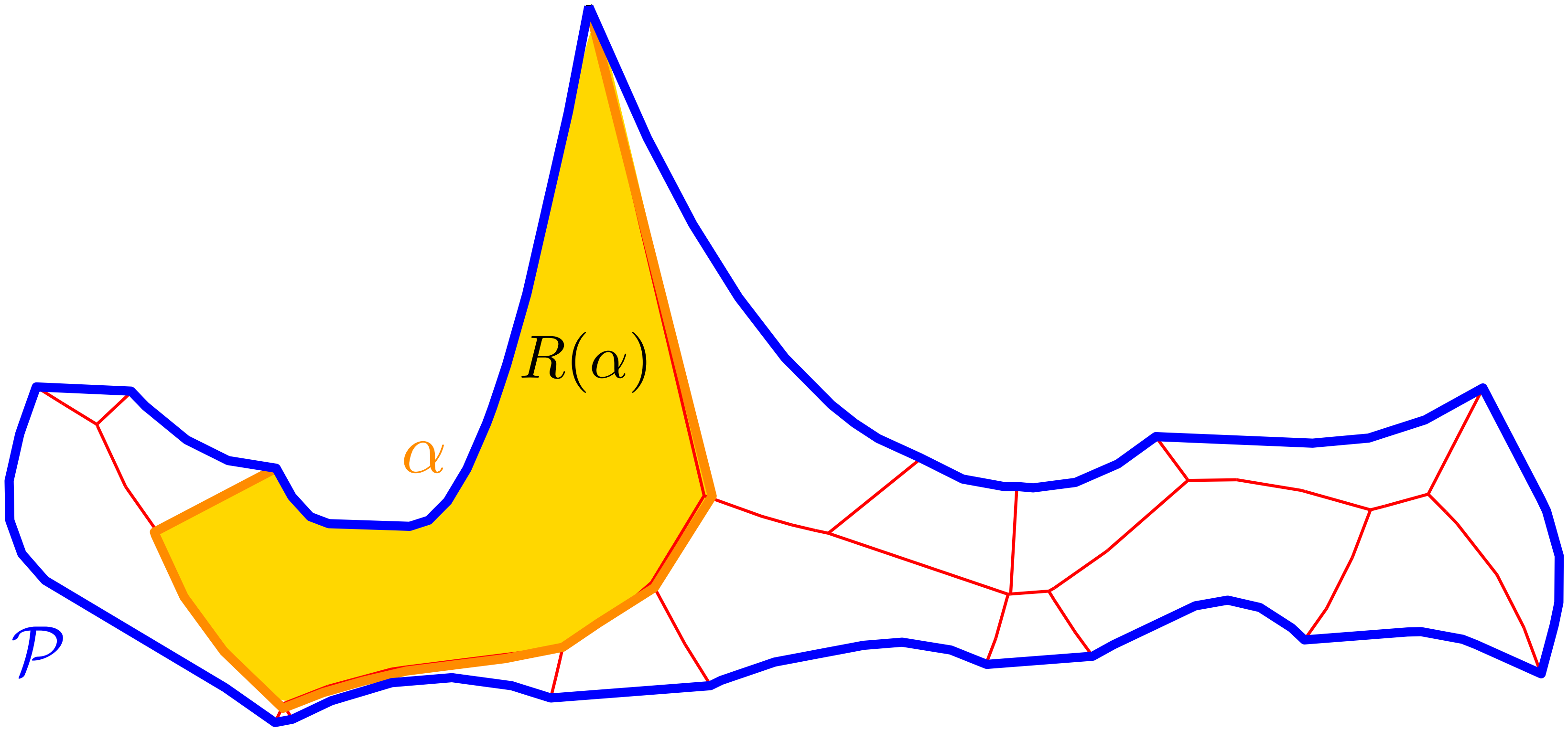
- Each boundary arc $\alpha \in \mathcal{P}$ has one region $R(\alpha)$.
- $\partial R(\alpha)$ is an s_α-monotone path plus α.





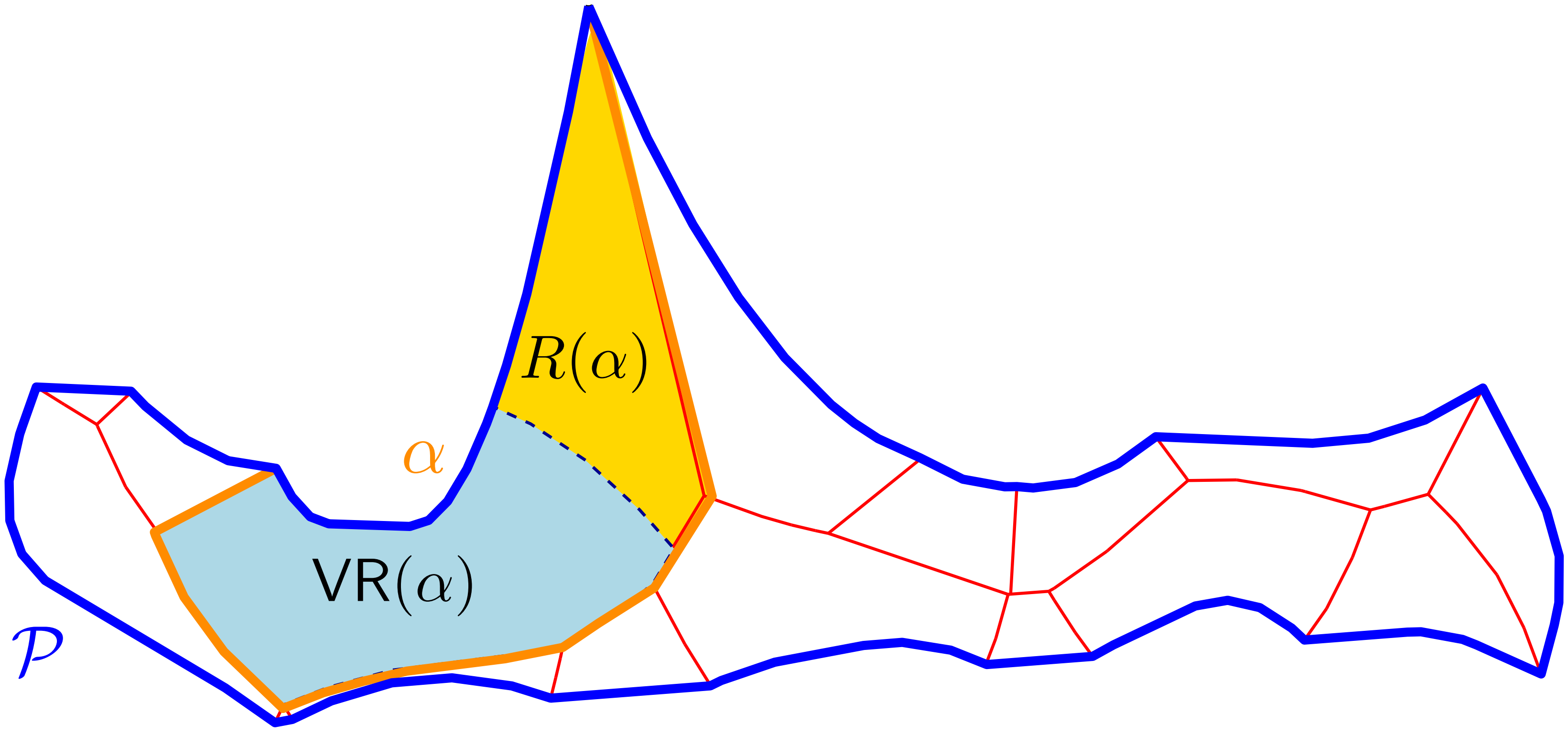
Properties of Voronoi-like diagrams

- Voronoi-like regions are supersets of the real Voronoi regions.



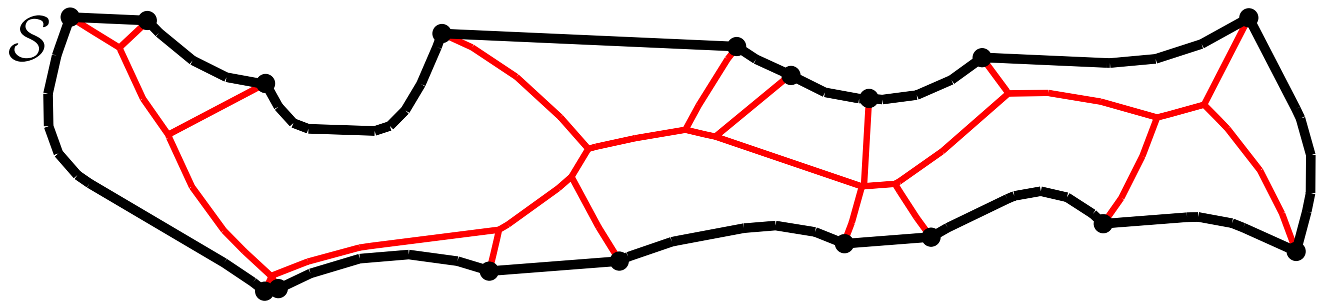
Properties of Voronoi-like diagrams

- Voronoi-like regions are supersets of the real Voronoi regions.



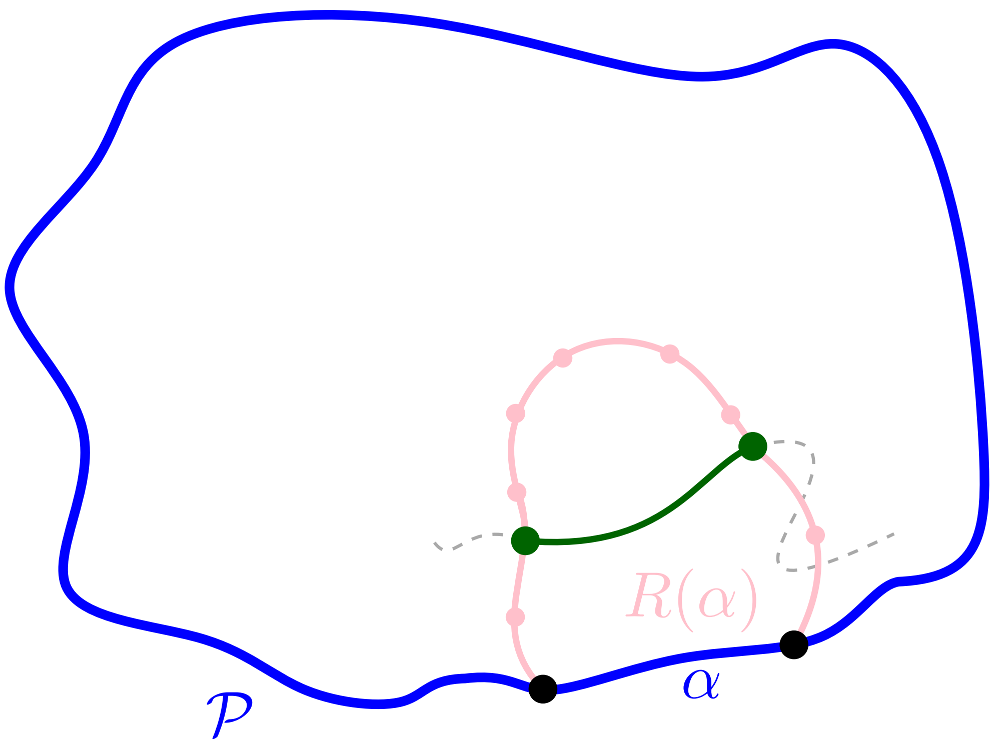
Properties of Voronoi-like diagrams

- Voronoi-like regions are supersets of the real Voronoi regions.
- For all arcs $\mathcal{S}$, $\mathcal{V}_l(\mathcal{S})$ equals the real diagram $\mathcal{V}(\mathcal{S}) = \mathcal{V}(\mathcal{S} \setminus s) \cap \mathsf{VR}(s)$.



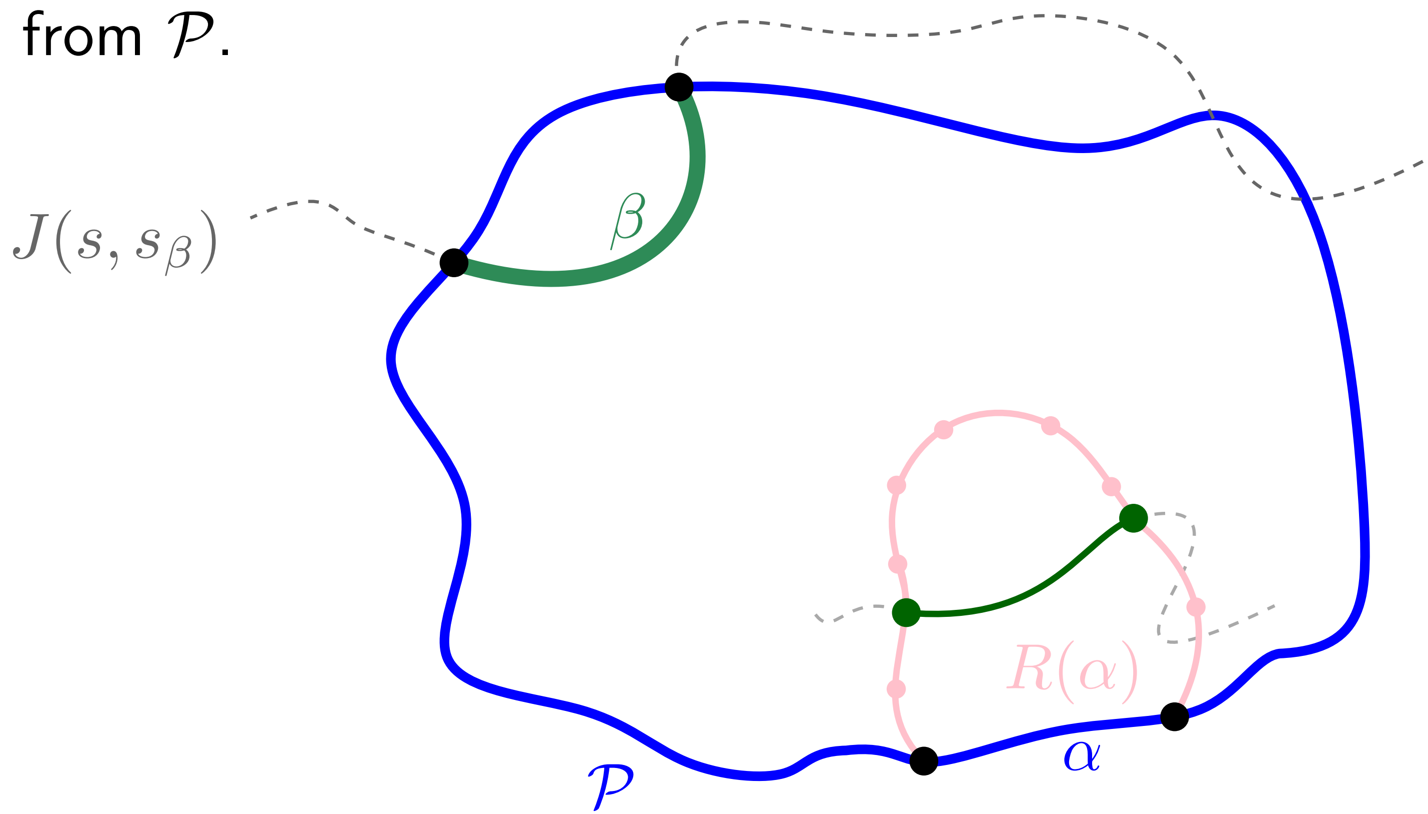
Properties of Voronoi-like diagrams

- Voronoi-like regions are supersets of the real Voronoi regions.
- For all arcs $\mathcal{S}$, $\mathcal{V}_l(\mathcal{S})$ equals the real diagram $\mathcal{V}(\mathcal{S}) = \mathcal{V}(S \setminus s) \cap \mathrm{VR}(s)$.
- **Missing arc lemma:**
  Suppose an α-related bisector appears within $R(\alpha)$. Then there is an arc β "missing" from $\mathcal{P}$.



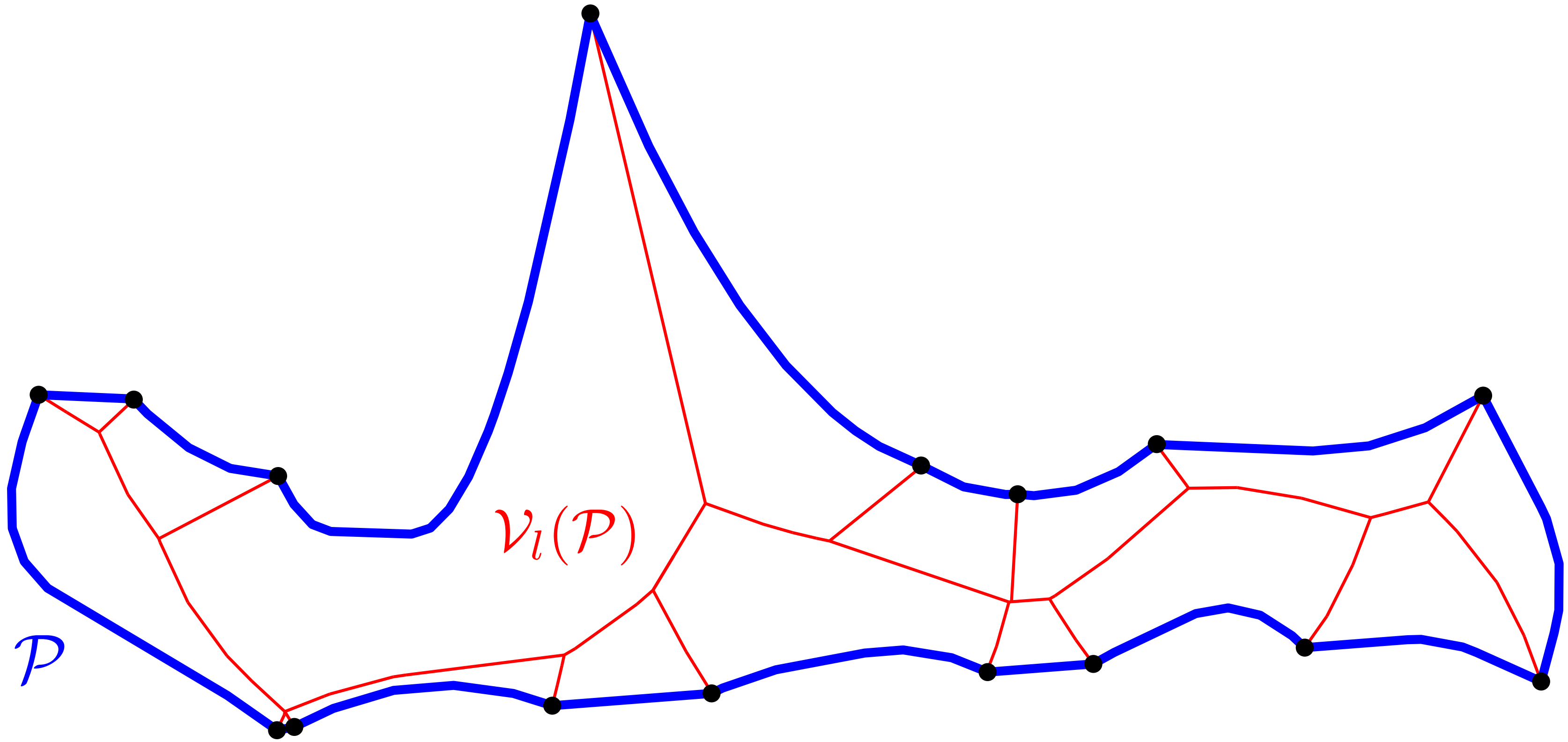
Properties of Voronoi-like diagrams

- Voronoi-like regions are supersets of the real Voronoi regions.
- For all arcs $\mathcal{S}$, $\mathcal{V}_l(\mathcal{S})$ equals the real diagram $\mathcal{V}(\mathcal{S}) = \mathcal{V}(S \setminus s) \cap \mathrm{VR}(s)$.
- **Missing arc lemma:**
  Suppose an α-related bisector appears within $R(\alpha)$. Then there is an arc β "missing" from $\mathcal{P}$.



Uniqueness of Voronoi-like diagrams

Theorem:
The Voronoi-like diagram $\mathcal{V}_l(\mathcal{P})$ of a boundary curve $\mathcal{P}$ is **unique**.



No monotonicity property

Voronoi-like regions do not have the standard monotonicity property of real Voronoi regions:

$$\text{Voronoi diagram: } S' \subseteq S \Rightarrow \mathsf{VR}(p, S) \subseteq \mathsf{VR}(p, S')$$

$$\text{Voronoi-like diagram: } \mathcal{S}' \subseteq \mathcal{S} \not\Rightarrow R(\alpha, \mathcal{S}) \subseteq R(\alpha, \mathcal{S}')$$

No monotonicity property

Voronoi-like regions do not have the standard monotonicity property of real Voronoi regions:

Voronoi diagram: $S' \subseteq S \Rightarrow \mathsf{VR}(p, S) \subseteq \mathsf{VR}(p, S')$

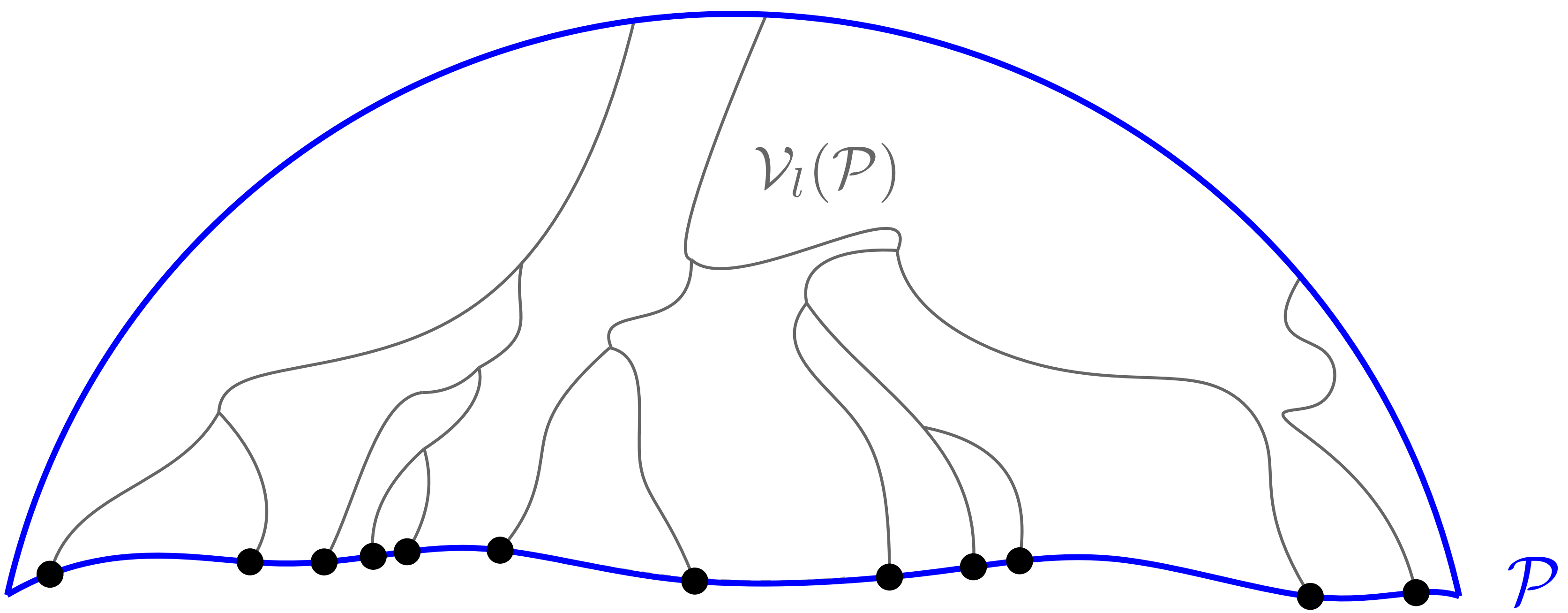
Voronoi-like diagram: $\mathcal{S}' \subseteq \mathcal{S} \not\Rightarrow R(\alpha, \mathcal{S}) \subseteq R(\alpha, \mathcal{S}')$

In proofs, use missing-arc lemma instead

Arc insertion

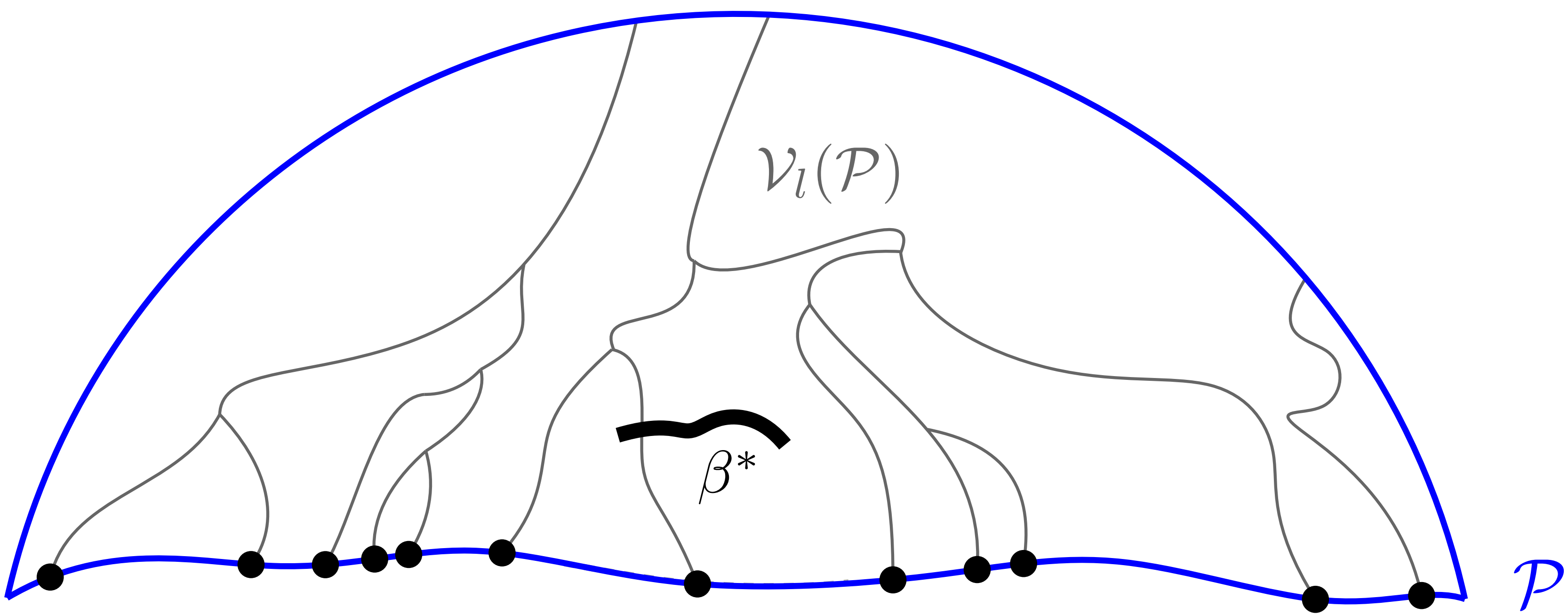
Arc insertion

Problem: Given a boundary curve $\mathcal{P}$ for $\mathcal{S}' \subset \mathcal{S}$ and its Voronoi-like diagram $\mathcal{V}_l(\mathcal{P})$, **insert** arc $\beta^* \in \mathcal{S} \setminus \mathcal{S}'$, prolong $\beta^* \subseteq \beta$, compute $R(\beta)$, and update the diagram to $\mathcal{V}_l(\mathcal{P}) \oplus \beta$.



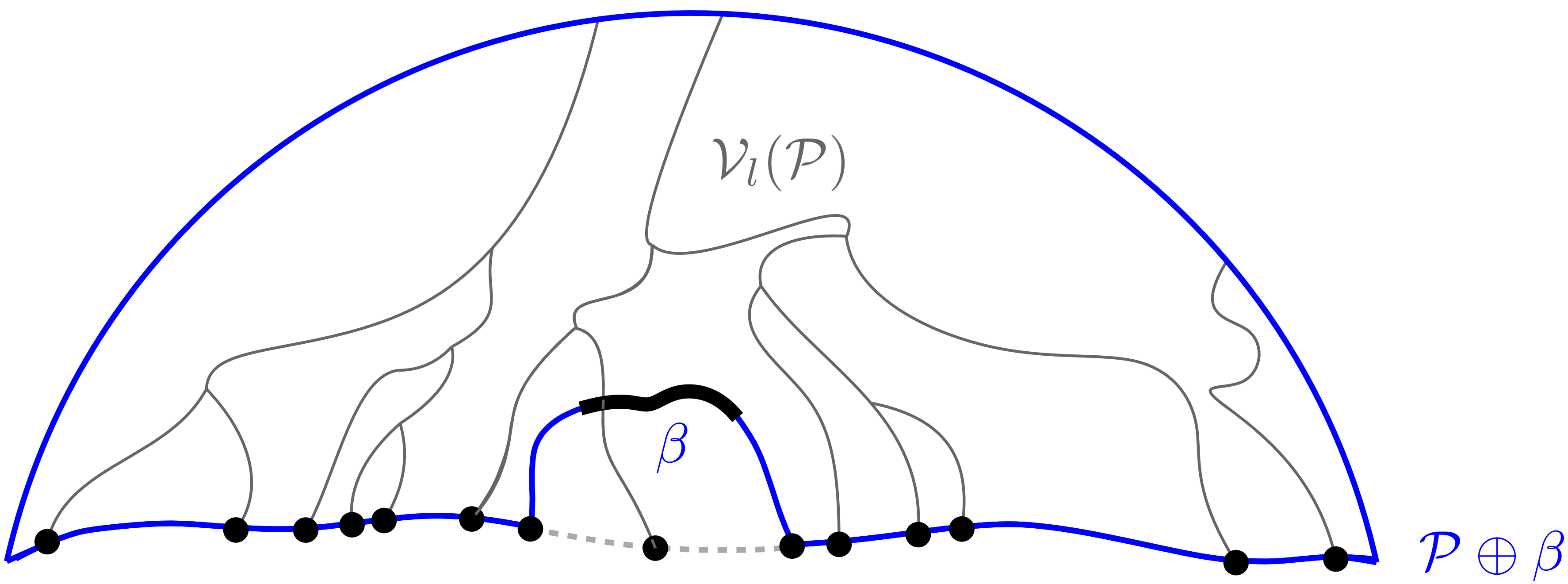
Arc insertion

Problem: Given a boundary curve $\mathcal{P}$ for $\mathcal{S}' \subset \mathcal{S}$ and its Voronoi-like diagram $\mathcal{V}_l(\mathcal{P})$, **insert** arc $\beta^* \in \mathcal{S} \setminus \mathcal{S}'$, prolong $\beta^* \subseteq \beta$, compute $R(\beta)$, and update the diagram to $\mathcal{V}_l(\mathcal{P}) \oplus \beta$.



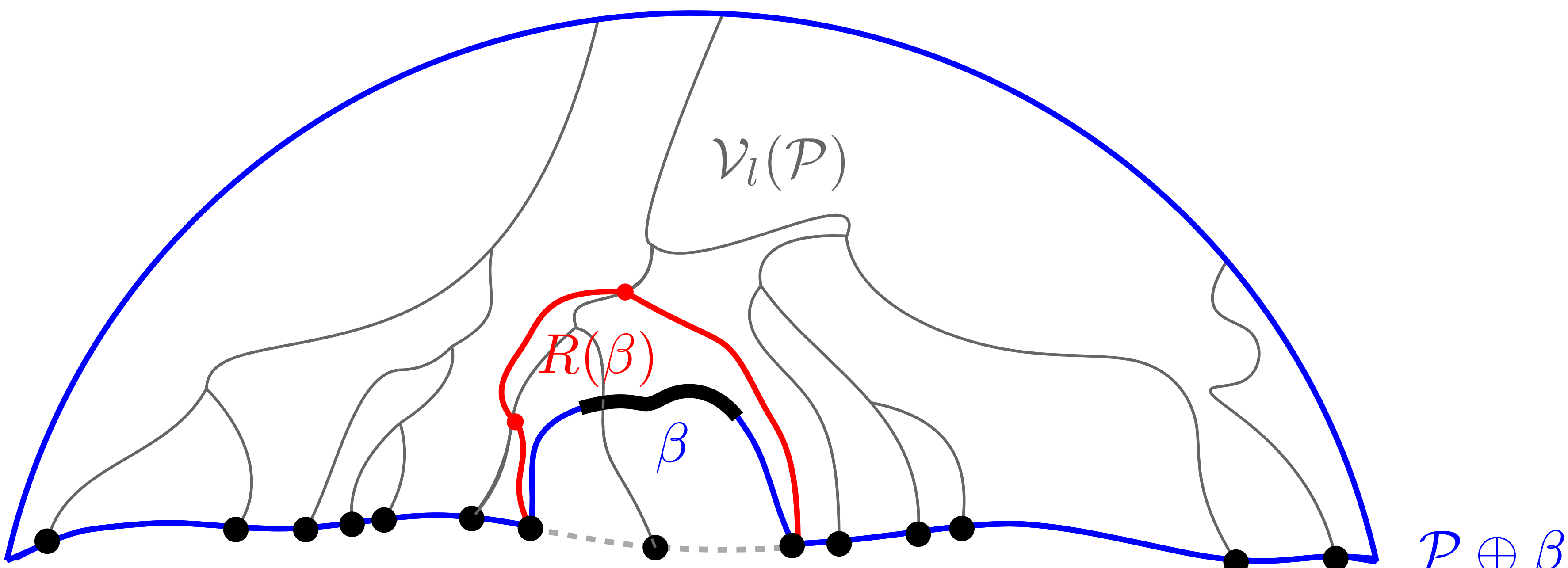
Arc insertion

Problem: Given a boundary curve $\mathcal{P}$ for $\mathcal{S}' \subset \mathcal{S}$ and its Voronoi-like diagram $\mathcal{V}_l(\mathcal{P})$, **insert** arc $\beta^* \in \mathcal{S} \setminus \mathcal{S}'$, prolong $\beta^* \subseteq \beta$, compute $R(\beta)$, and update the diagram to $\mathcal{V}_l(\mathcal{P}) \oplus \beta$.



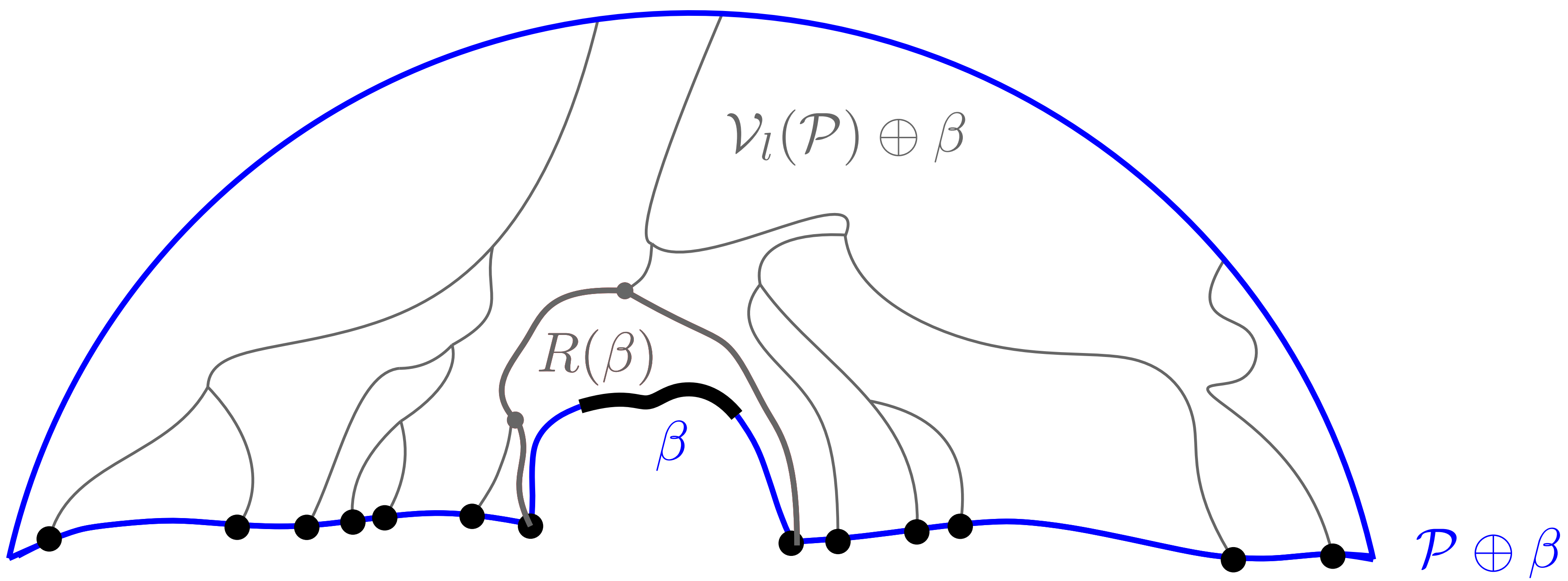
Arc insertion

Problem: Given a boundary curve $\mathcal{P}$ for $\mathcal{S}' \subset \mathcal{S}$ and its Voronoi-like diagram $\mathcal{V}_l(\mathcal{P})$, **insert** arc $\beta^* \in \mathcal{S} \setminus \mathcal{S}'$, prolong $\beta^* \subseteq \beta$, compute $R(\beta)$, and update the diagram to $\mathcal{V}_l(\mathcal{P}) \oplus \beta$.



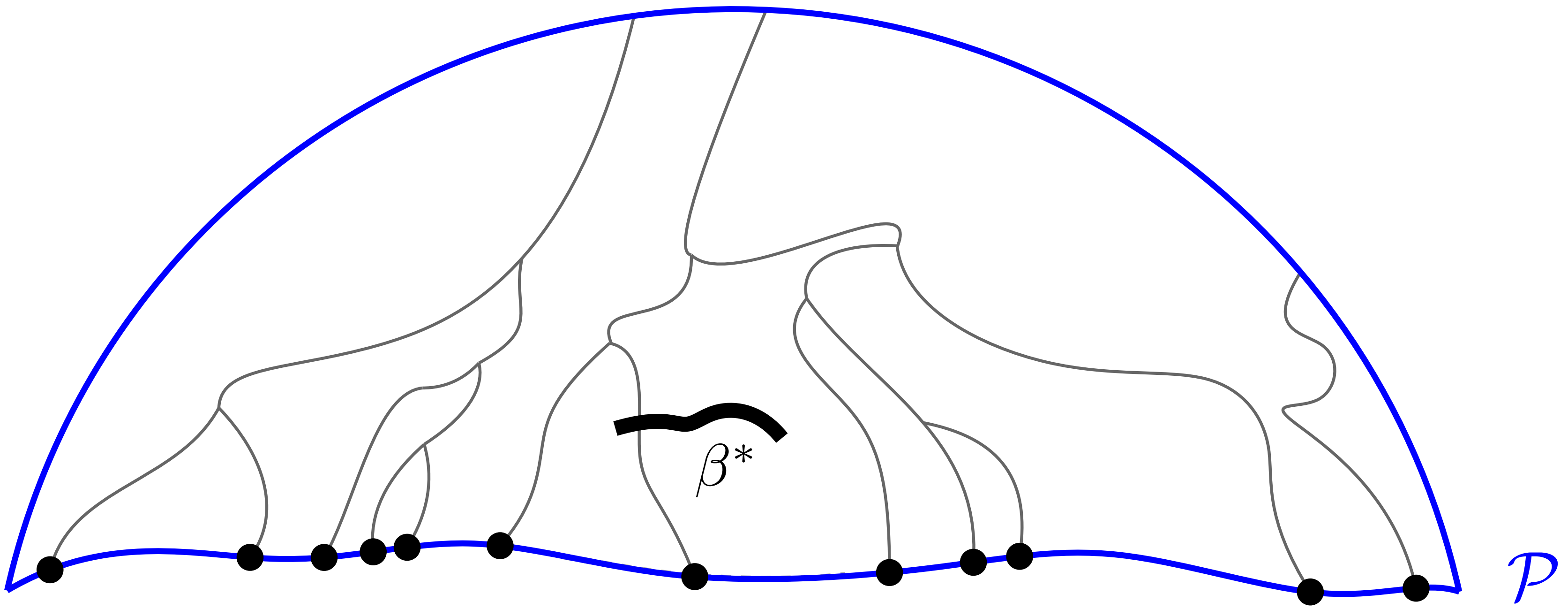
Arc insertion

Problem: Given a boundary curve $\mathcal{P}$ for $\mathcal{S}' \subset \mathcal{S}$ and its Voronoi-like diagram $\mathcal{V}_l(\mathcal{P})$, **insert** arc $\beta^* \in \mathcal{S} \setminus \mathcal{S}'$, prolong $\beta^* \subseteq \beta$, compute $R(\beta)$, and update the diagram to $\mathcal{V}_l(\mathcal{P}) \oplus \beta$.



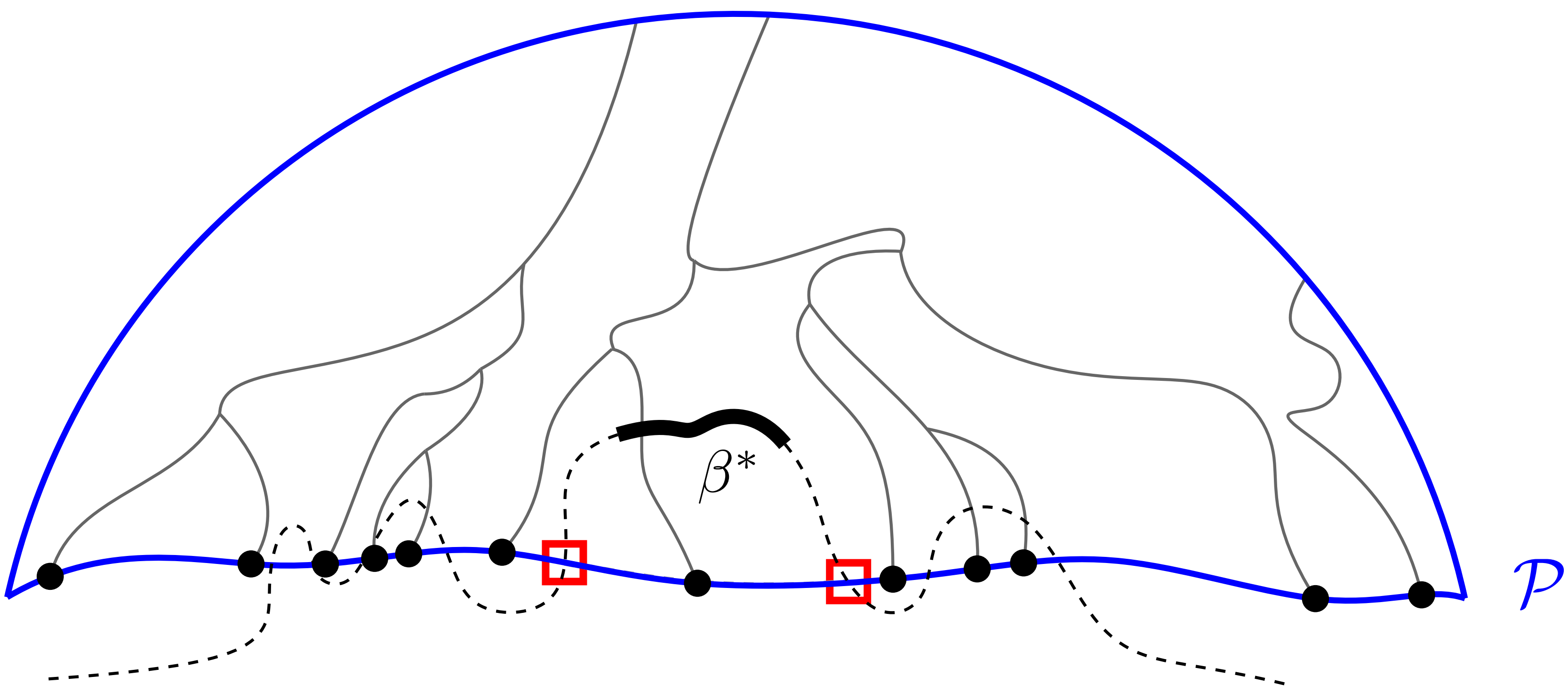
Arc insertion

- Compute the boundary curve $\mathcal{P} \oplus \beta$ containing β ($\beta^* \subseteq \beta$).



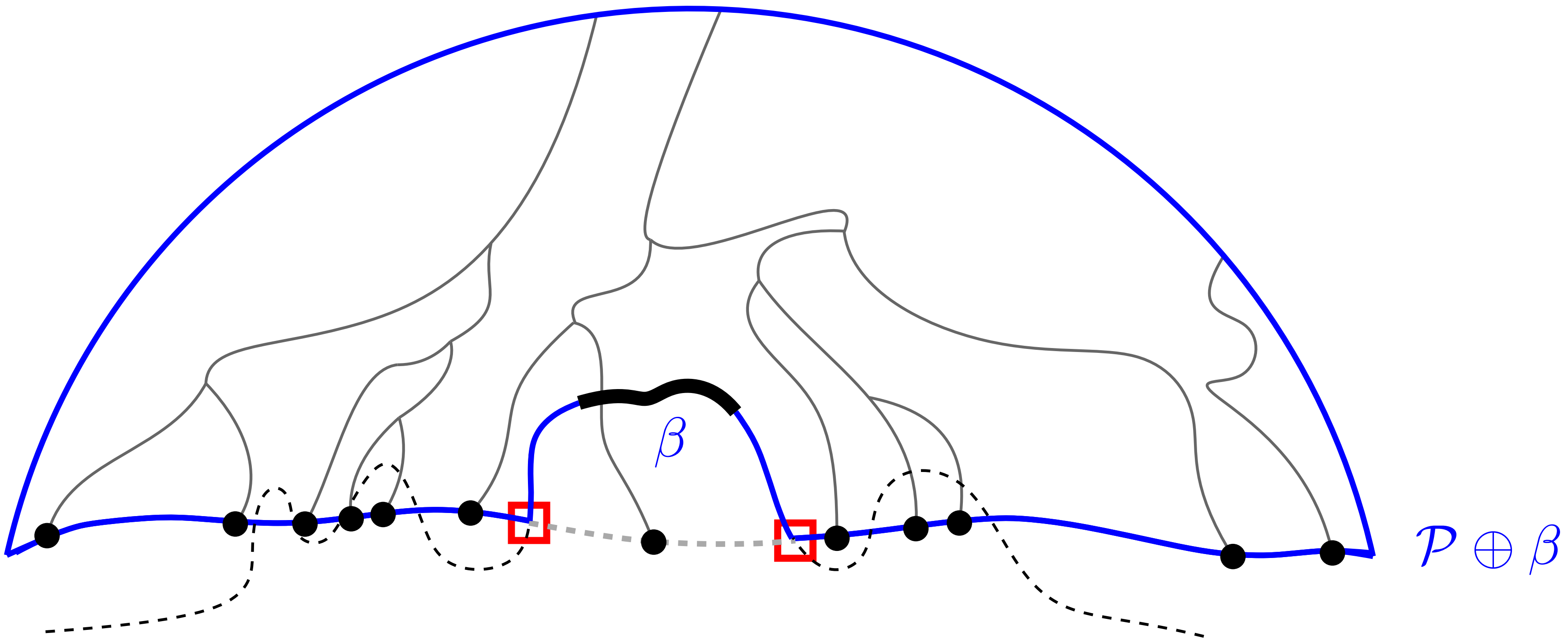
Arc insertion

- Compute the boundary curve $\mathcal{P} \oplus \beta$ containing β ($\beta^* \subseteq \beta$).



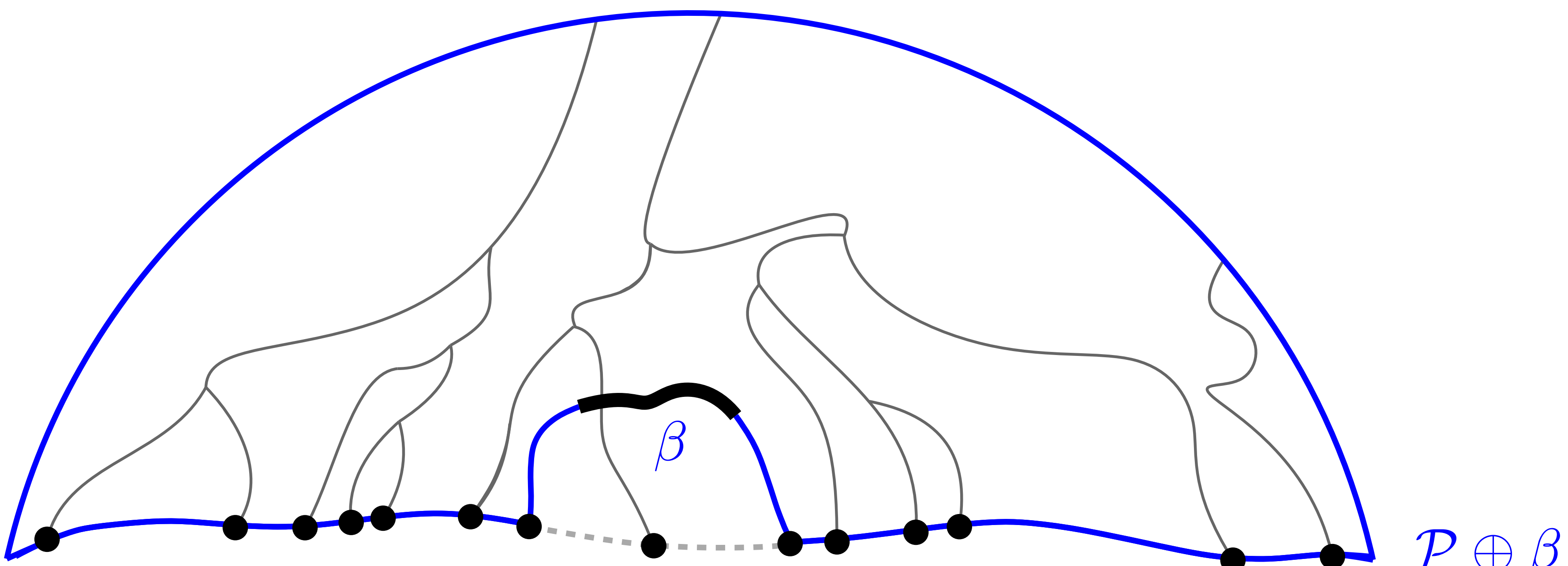
Arc insertion

- Compute the boundary curve $\mathcal{P} \oplus \beta$ containing β ($\beta^* \subseteq \beta$).



Arc insertion

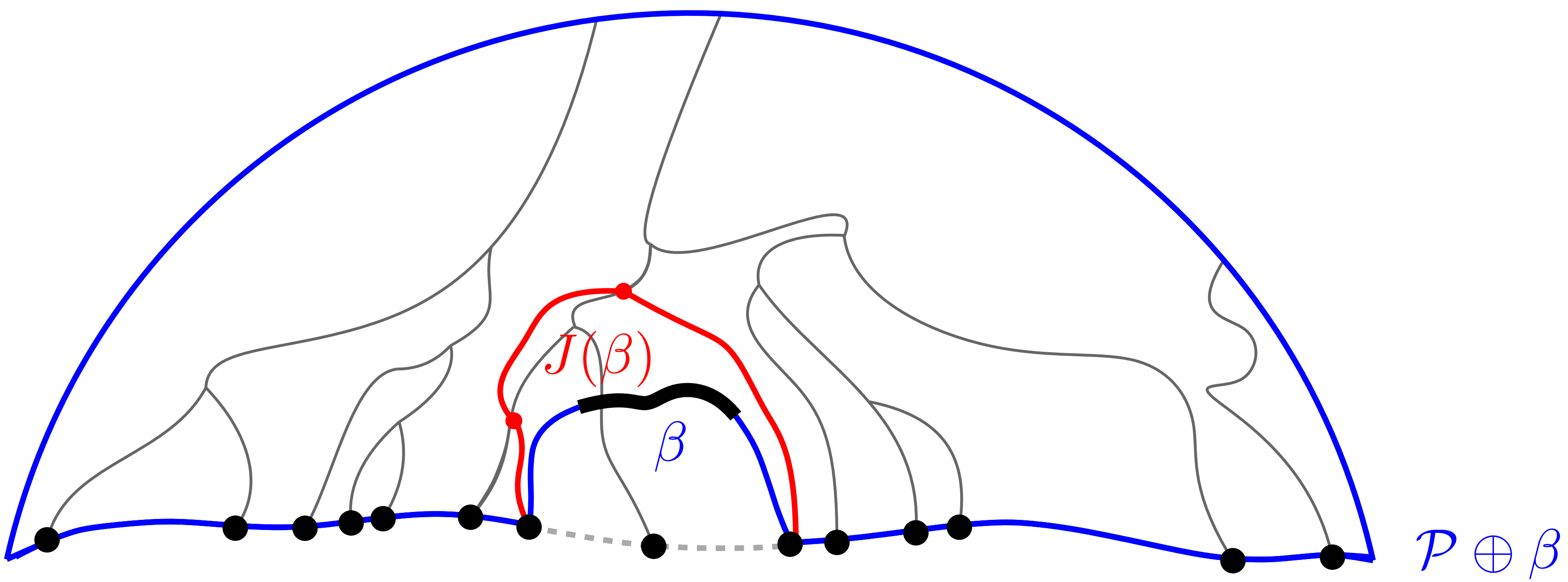
- Compute the boundary curve $\mathcal{P} \oplus \beta$ containing β ($\beta^* \subseteq \beta$).



Arc insertion

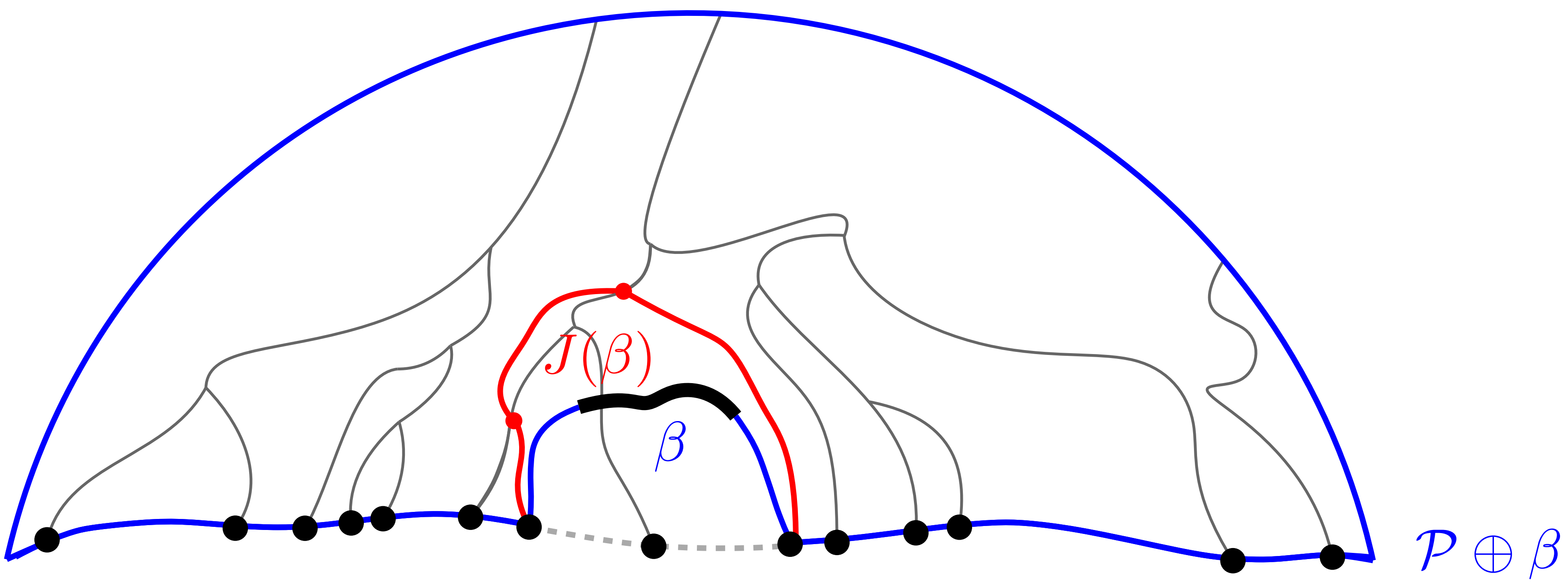
- Compute the boundary curve $\mathcal{P} \oplus \beta$ containing β ($\beta^* \subseteq \beta$).

- Compute the merge curve $J(\beta)$; it defines region $R(\beta)$.



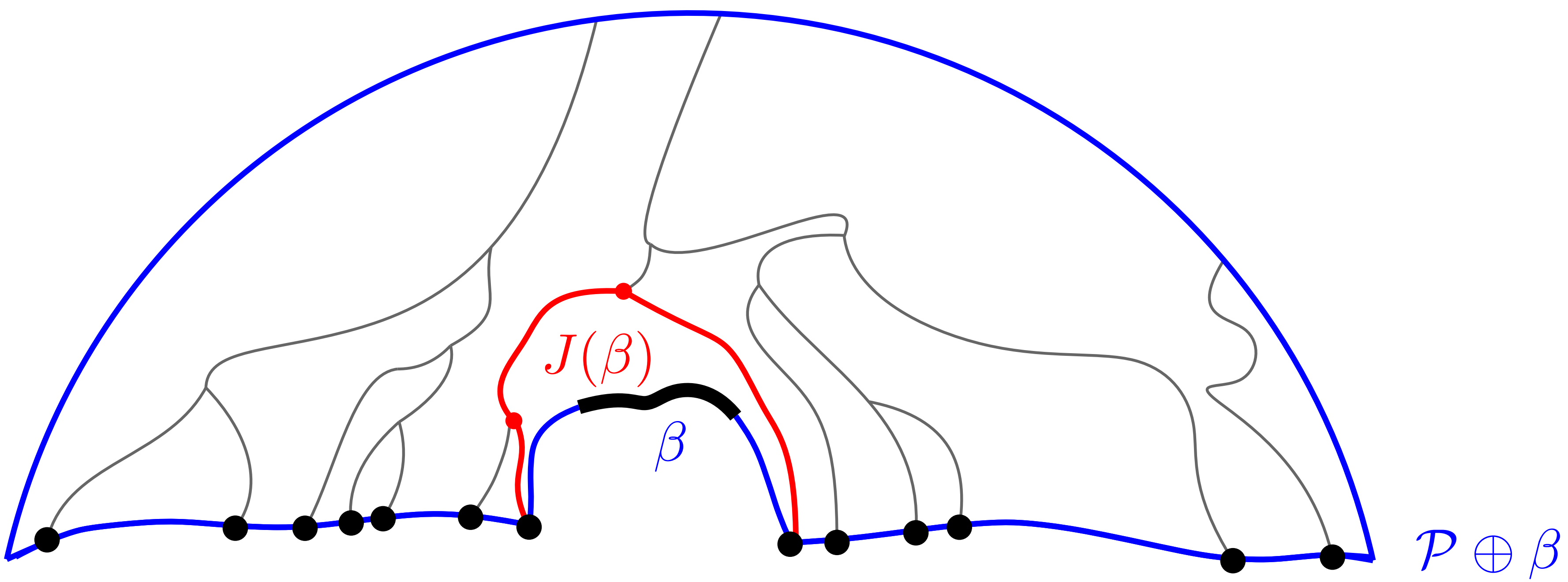
Arc insertion

- Compute the boundary curve $\mathcal{P} \oplus \beta$ containing β ($\beta^* \subseteq \beta$).
- Compute the merge curve $J(\beta)$; it defines region $R(\beta)$.
- Insert $R(\beta)$ in $\mathcal{V}_l(\mathcal{P})$ and derive $\mathcal{V}_l(\mathcal{P}) \oplus \beta$:



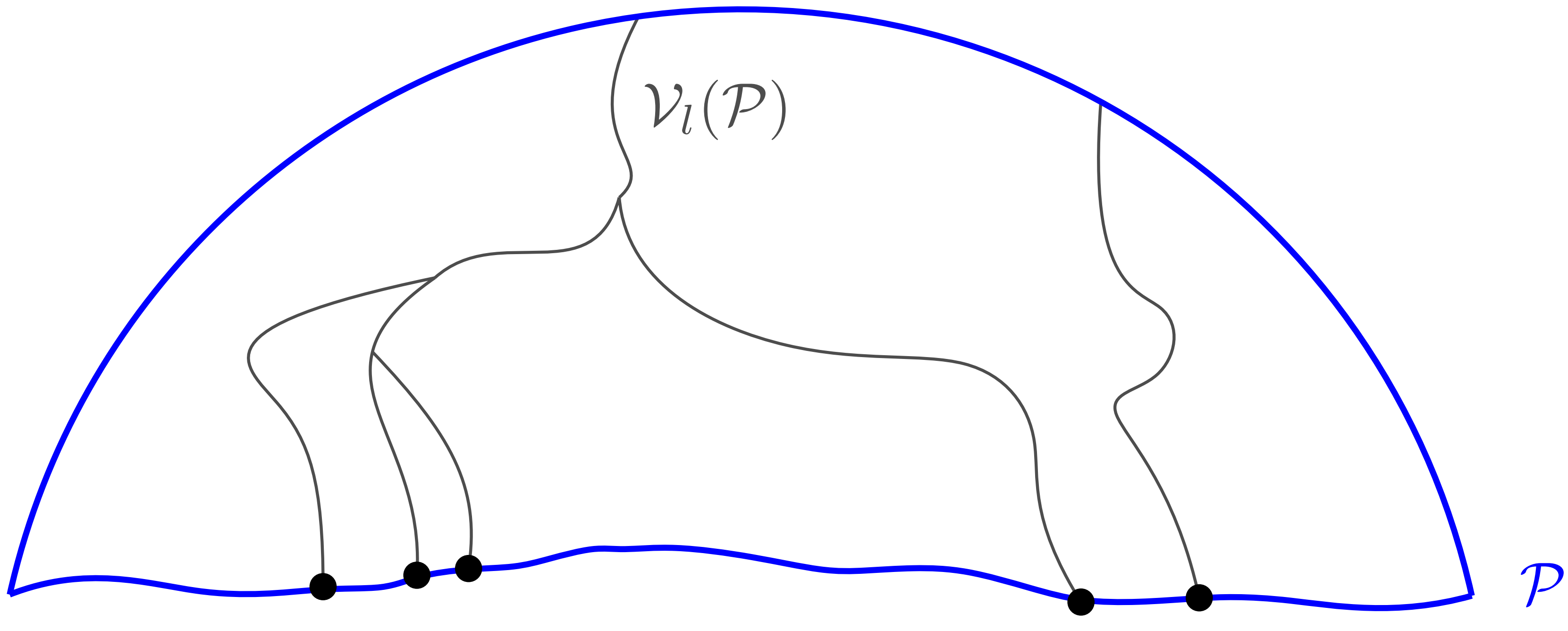
Arc insertion

- Compute the boundary curve $\mathcal{P} \oplus \beta$ containing β ($\beta^* \subseteq \beta$).
- Compute the merge curve $J(\beta)$; it defines region $R(\beta)$.
- Insert $R(\beta)$ in $\mathcal{V}_l(\mathcal{P})$ and derive $\mathcal{V}_l(\mathcal{P}) \oplus \beta$:



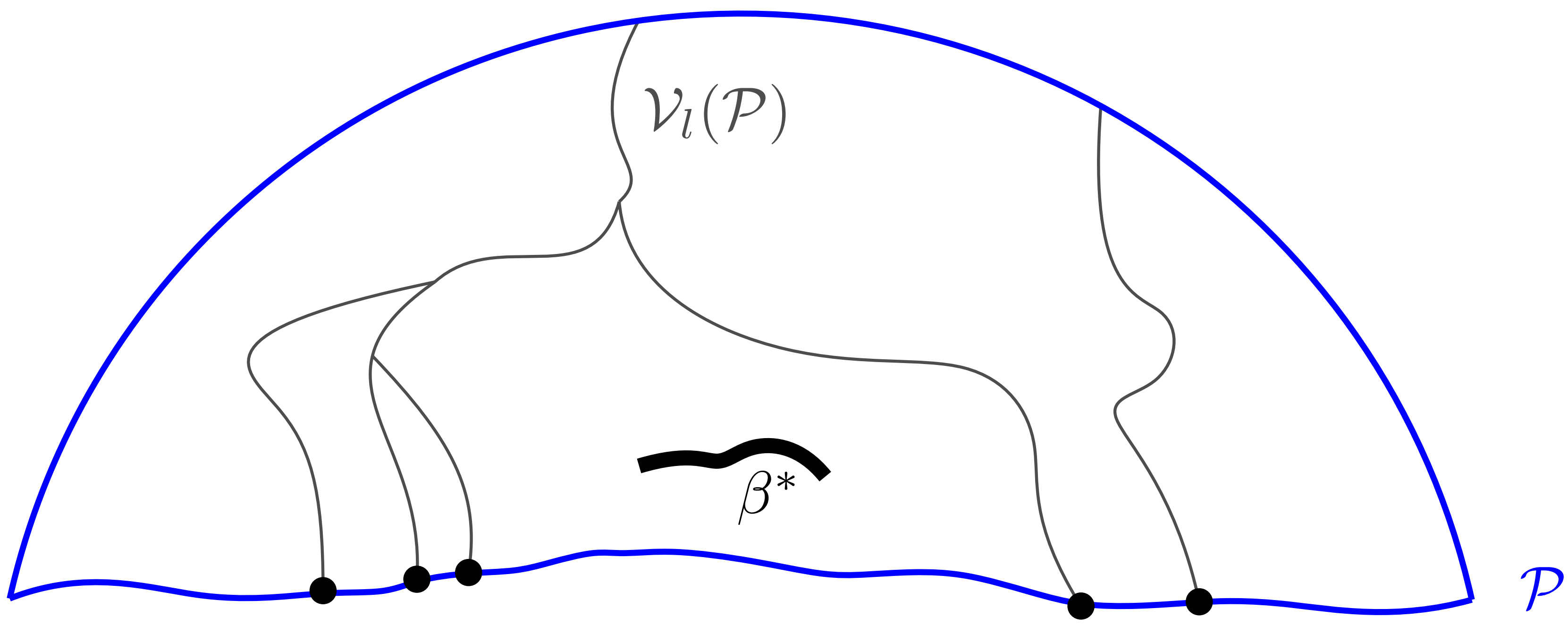
Insertion of β splits an arc

When inserting β, a face (and its arc) may split in two, creating a new auxiliary arc (γ') that was not in $\mathcal{S}$.



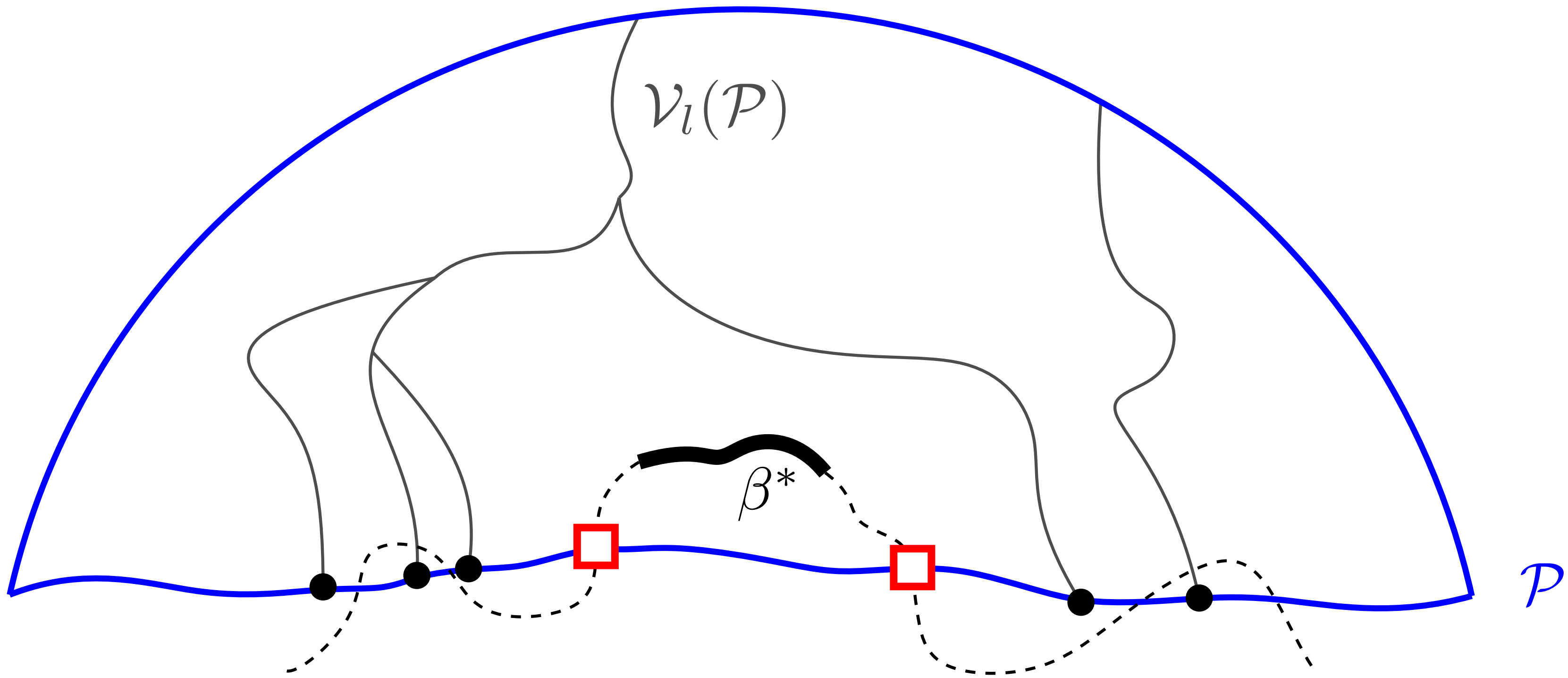
Insertion of β splits an arc

When inserting β, a face (and its arc) may split in two, creating a new auxiliary arc (γ') that was not in $\mathcal{S}$.



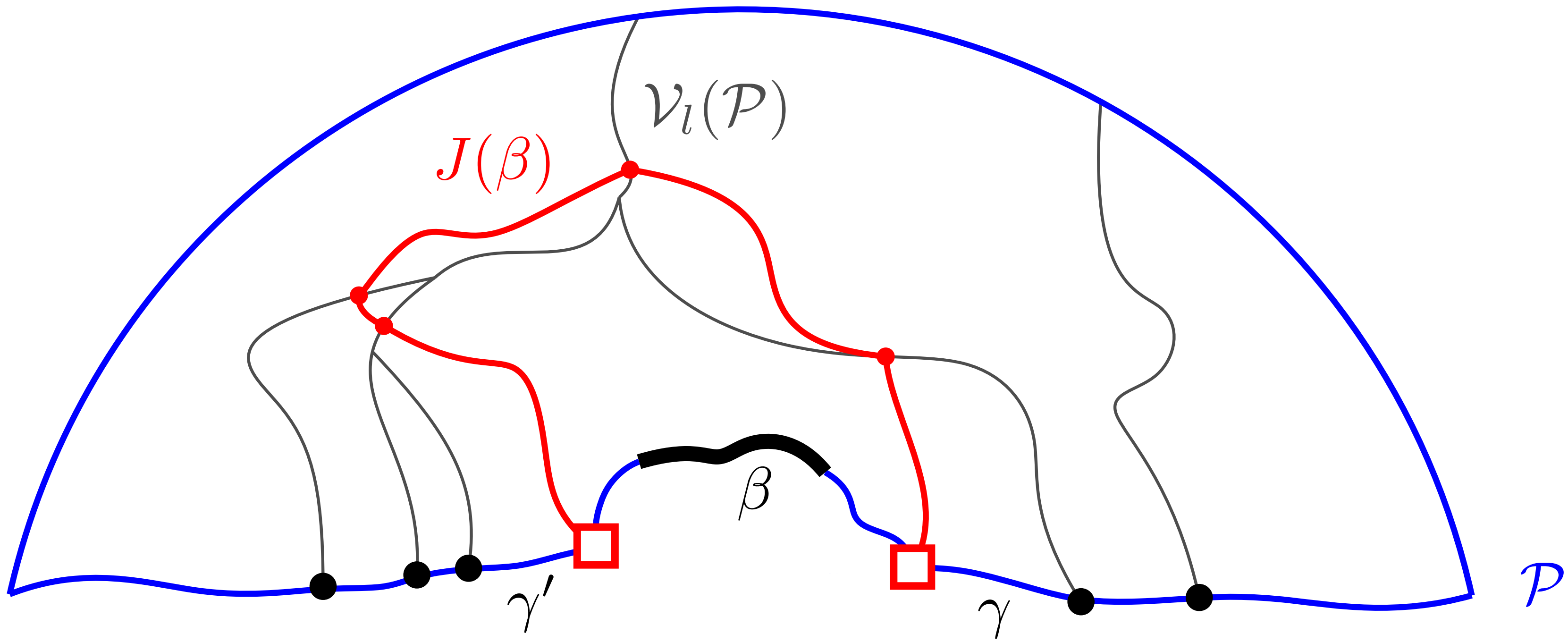
Insertion of β splits an arc

When inserting β, a face (and its arc) may split in two, creating a new auxiliary arc (γ') that was not in $\mathcal{S}$.



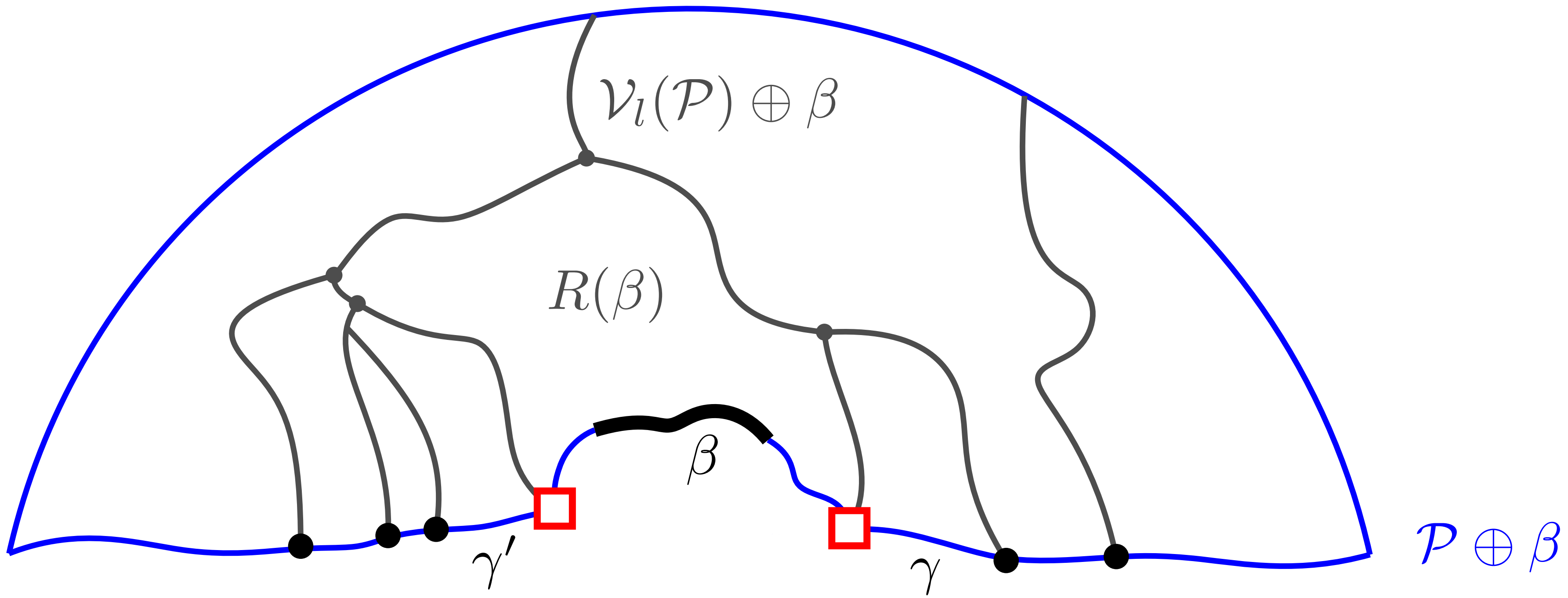
Insertion of β splits an arc

When inserting β, a face (and its arc) may split in two, creating a new auxiliary arc (γ') that was not in $\mathcal{S}$.



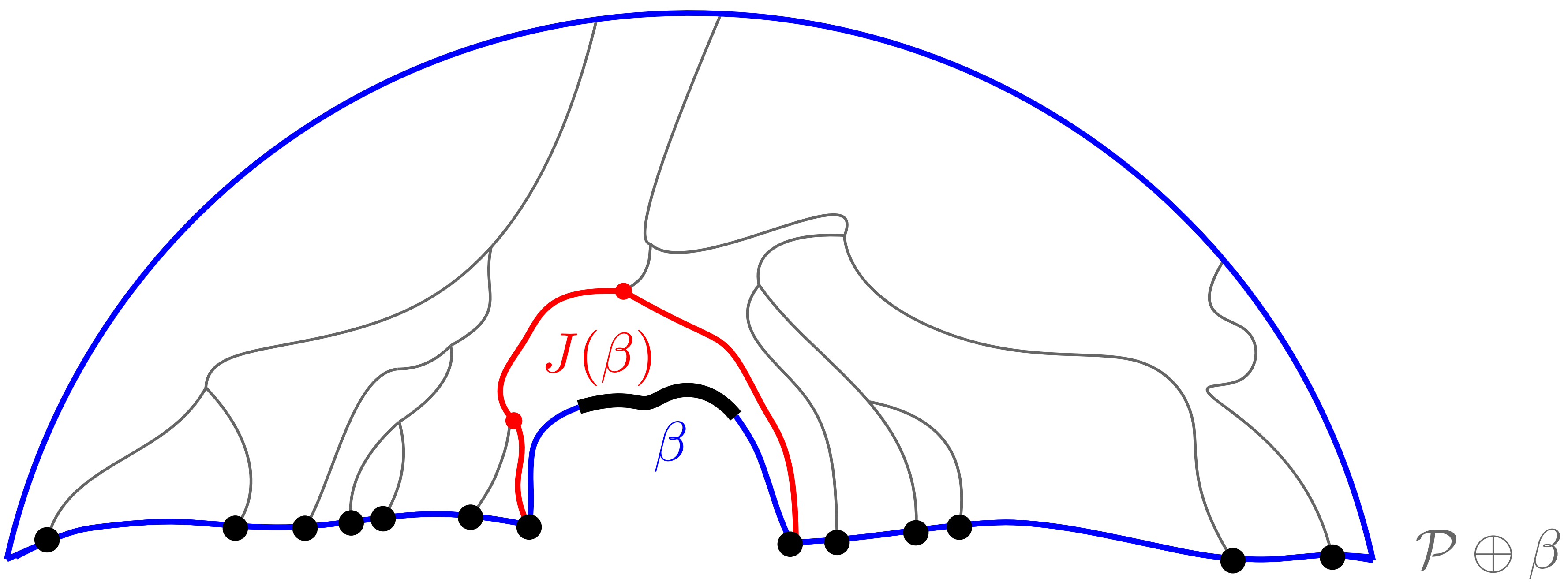
Insertion of β splits an arc

When inserting β, a face (and its arc) may split in two, creating a new auxiliary arc (γ') that was not in $\mathcal{S}$.



Arc insertion

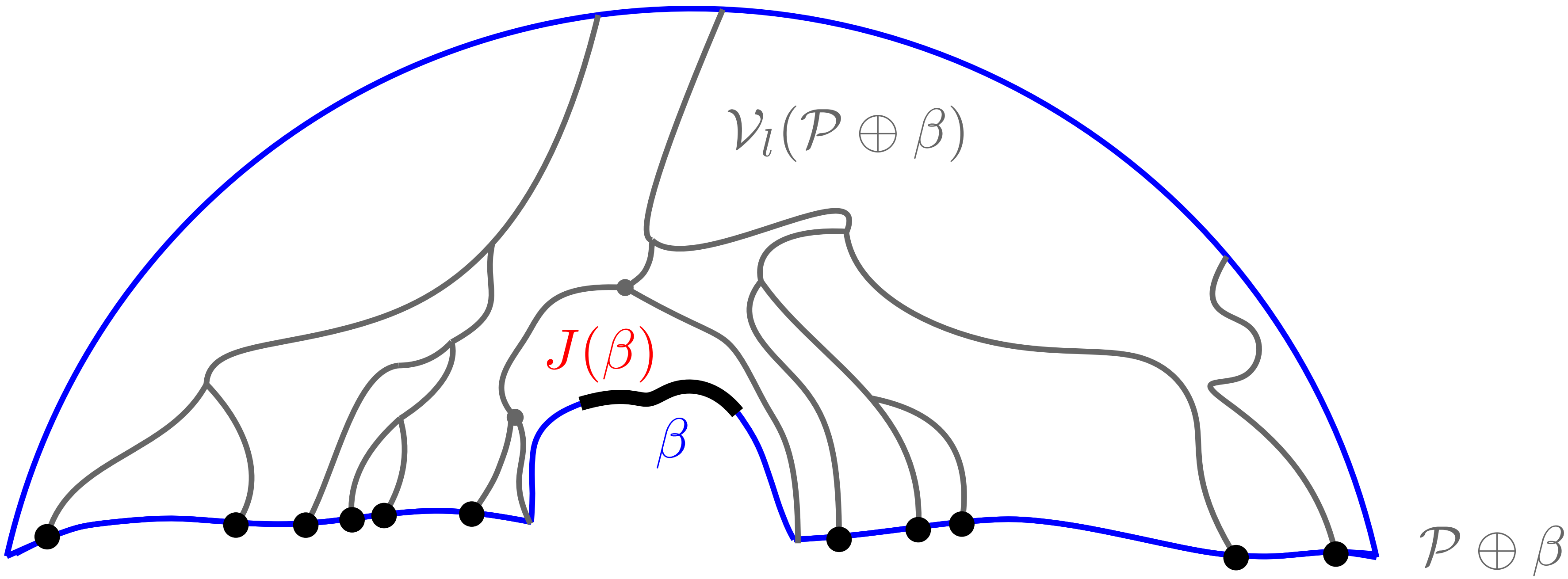
Theorem: The merge curve $J(\beta)$ is an s_β-monotone path.



Arc insertion

Theorem: The merge curve $J(\beta)$ is an s_β-monotone path.

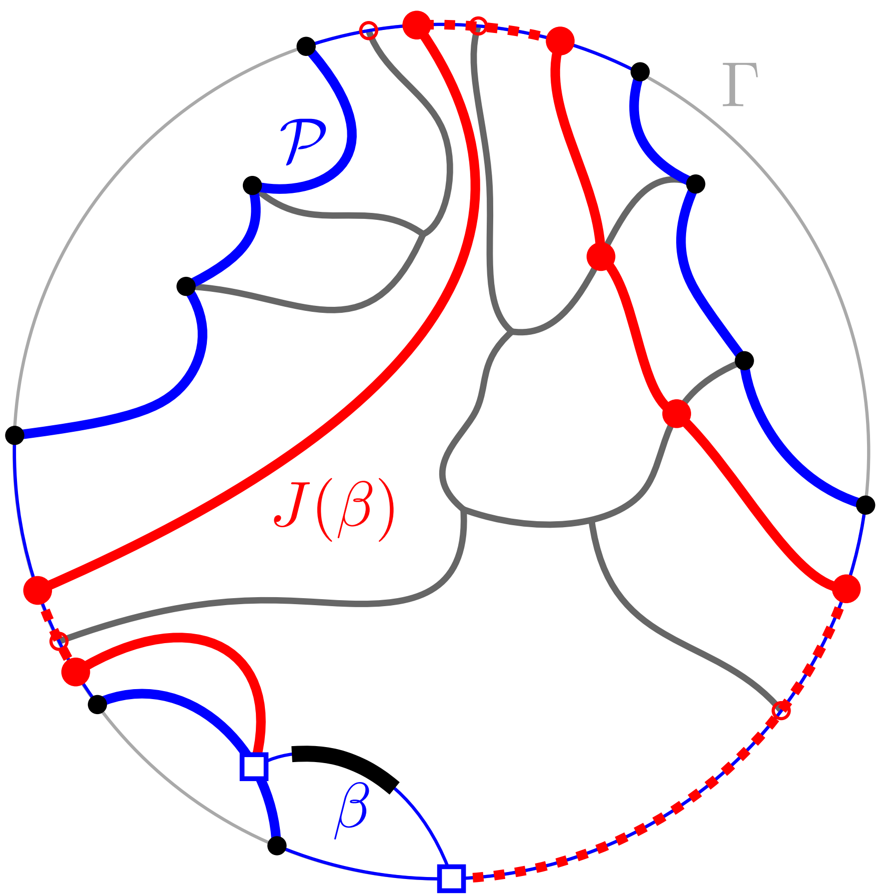
Theorem: $\mathcal{V}_l(\mathcal{P}) \oplus \beta$ is the Voronoi-like diagram, $\mathcal{V}_l(\mathcal{P} \oplus \beta)$.



Proof sketch

Theorem: The merge curve $J(\beta)$ is an s_β-monotone path.

Use a bi-directional induction starting at the two endpoints of β. Show:

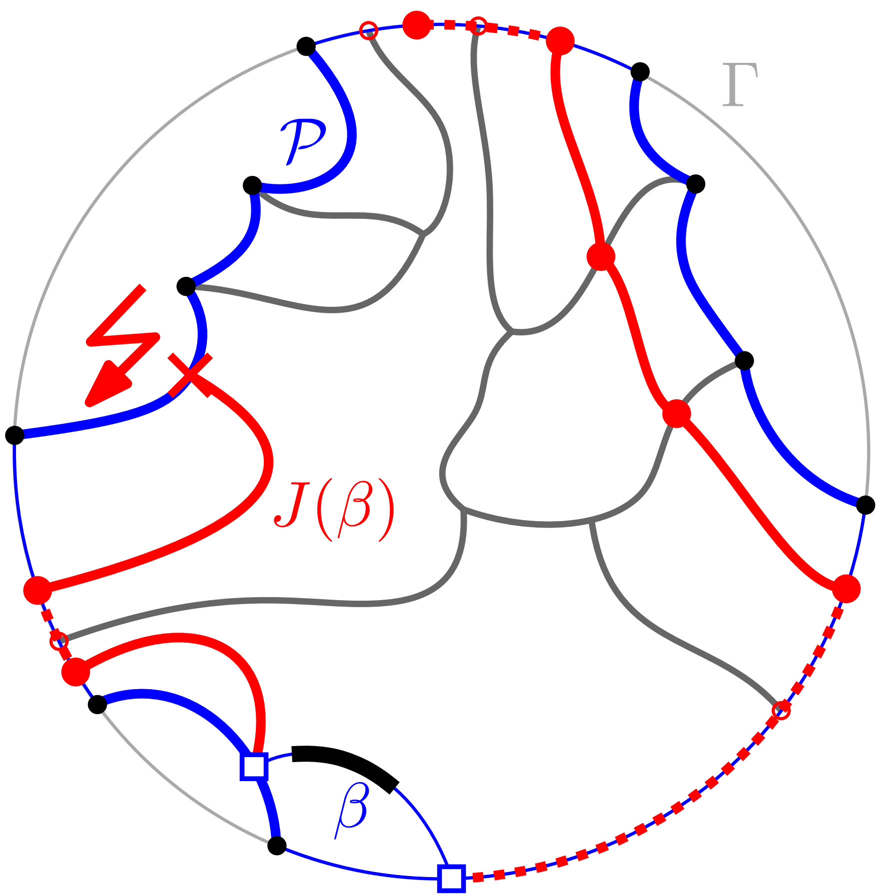


Proof sketch

Theorem: The merge curve $J(\beta)$ is an s_β-monotone path.

Use a bi-directional induction starting at the two endpoints of β. Show:

- $J(\beta)$ cannot hit a boundary arc.

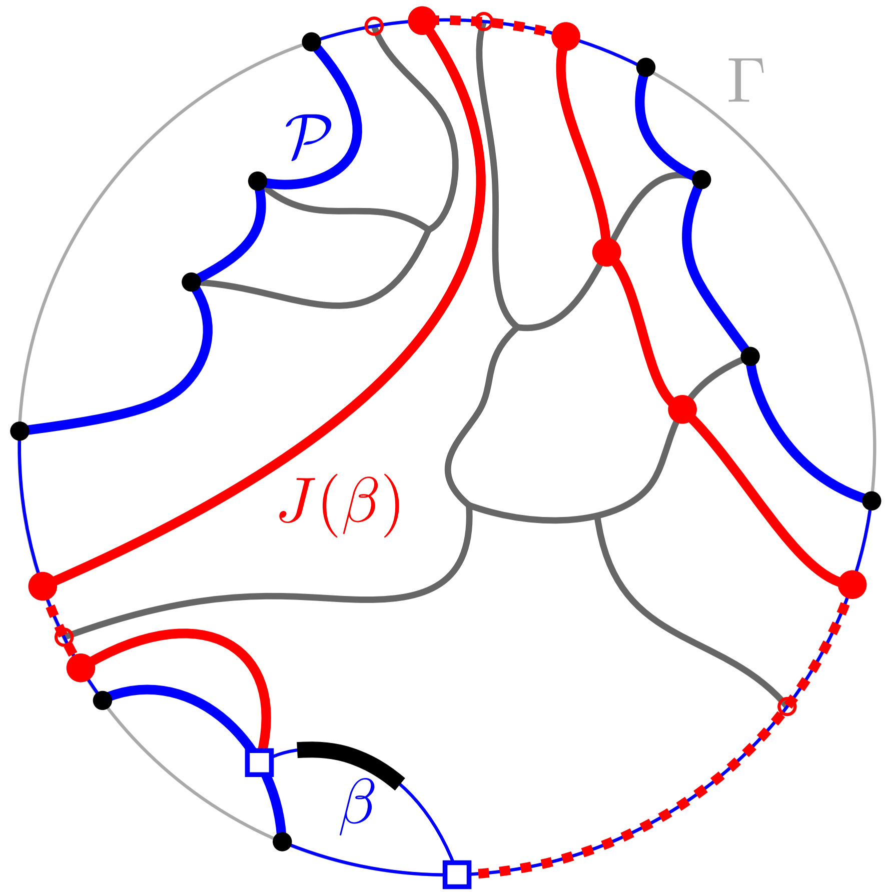


Proof sketch

Theorem: The merge curve $J(\beta)$ is an s_β-monotone path.

Use a bi-directional induction starting at the two endpoints of β. Show:

- $J(\beta)$ cannot hit a boundary arc.
- $J(\beta)$ cannot get stuck on Γ.

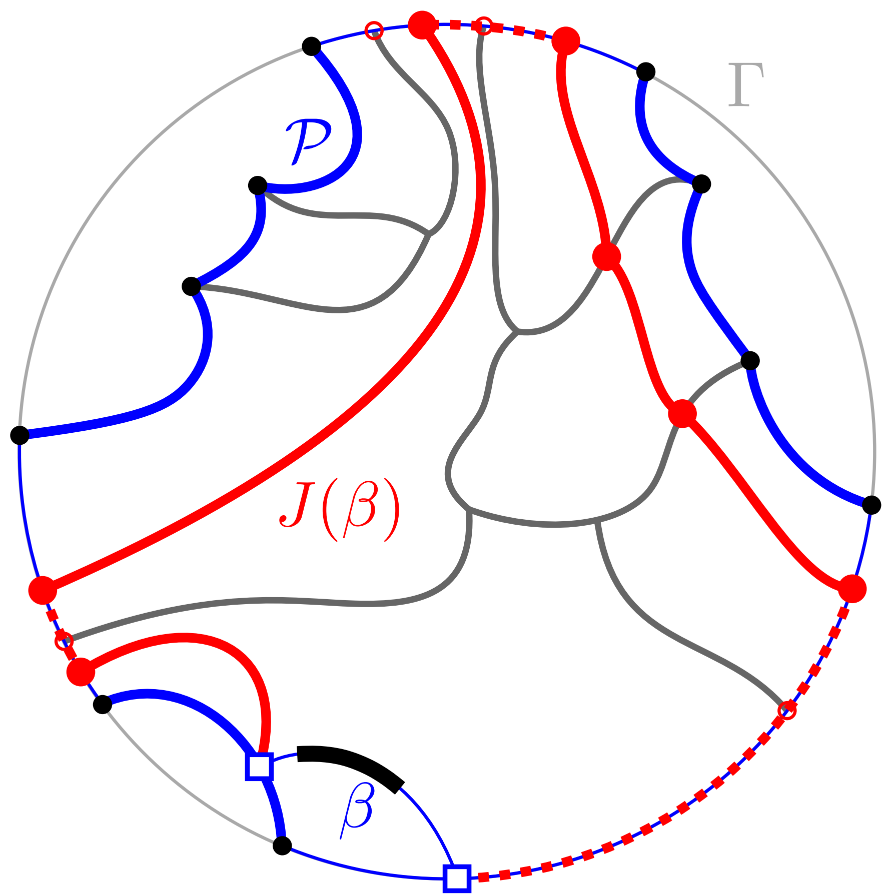


Proof sketch

Theorem: The merge curve $J(\beta)$ is an s_β-monotone path.

Use a bi-directional induction starting at the two endpoints of β. Show:

- $J(\beta)$ cannot hit a boundary arc.
- $J(\beta)$ cannot get stuck on Γ.
- $J(\beta)$ can visit a Voronoi-like region at most once.

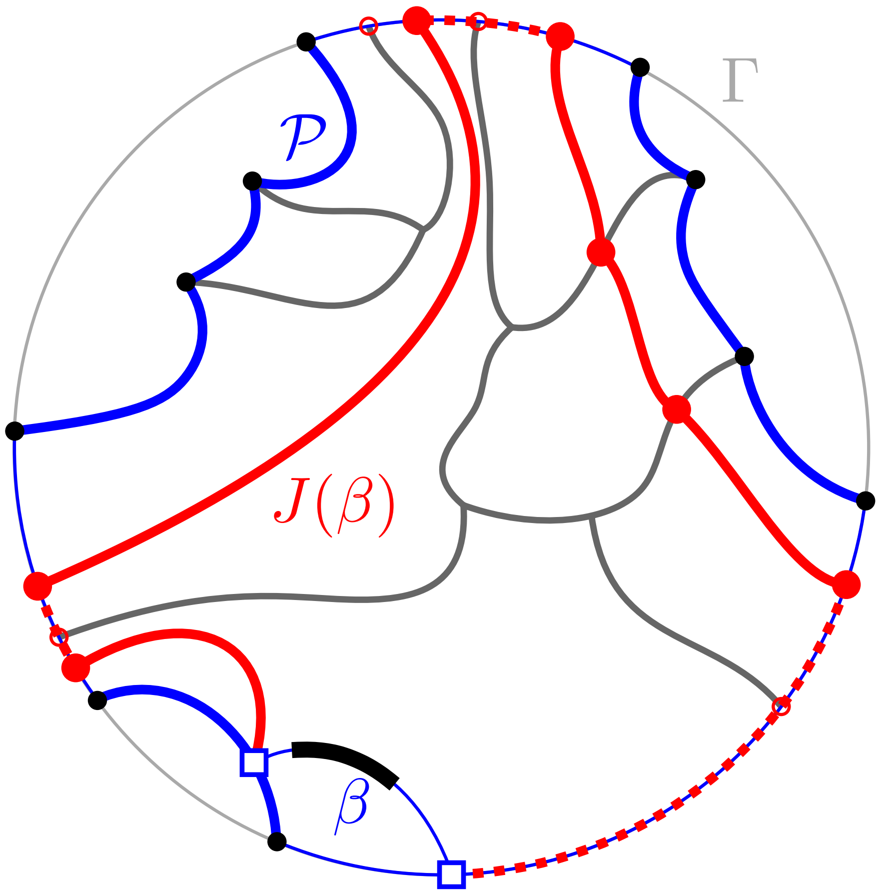


Proof sketch

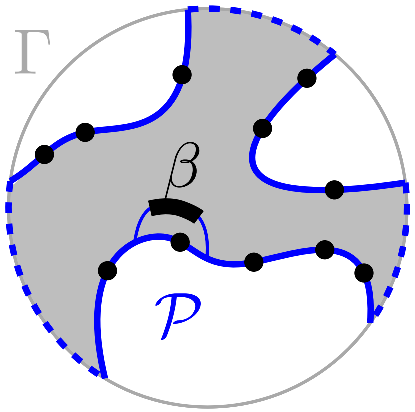
Theorem: The merge curve $J(\beta)$ is an s_β-monotone path.

Use a bi-directional induction starting at the two endpoints of β. Show:

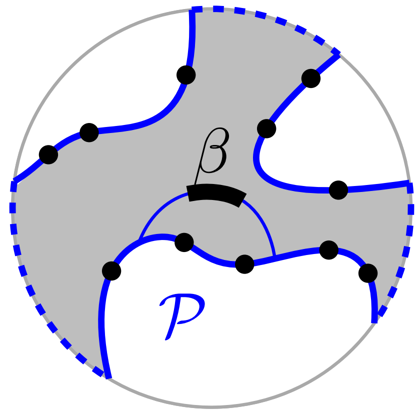
- $J(\beta)$ cannot hit a boundary arc.
- $J(\beta)$ cannot get stuck on Γ.
- $J(\beta)$ can visit a Voronoi-like region at most once.
- At the end, the 2 brunches of $J(\beta)$ meet in the same region



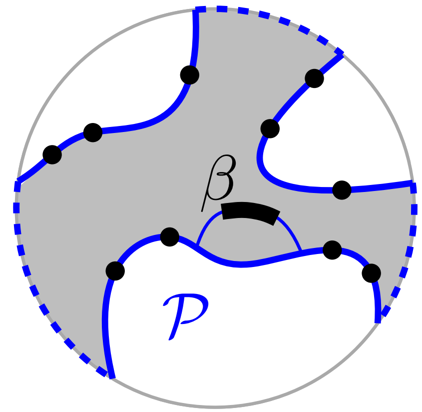
Possibilities for inserting β in $\mathcal{P}$



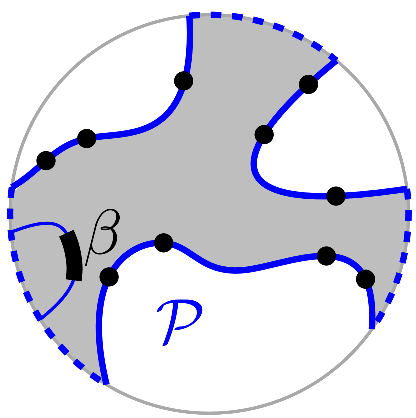
(a) Ordinary.



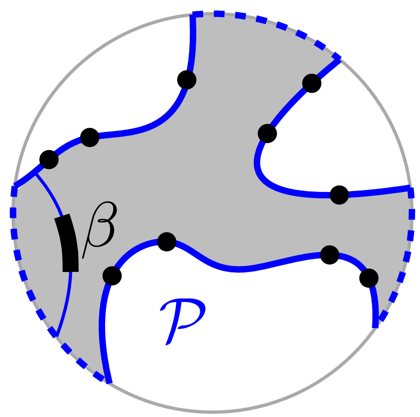
(b) Delete arc.



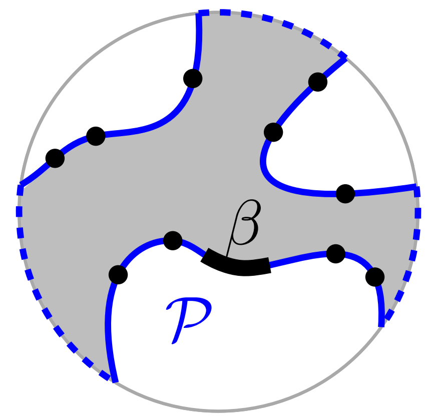
(c) Split arc.



(d) Split Γ-arc.



(e) Shrink Γ-arc.



(f) Trivial.

A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.

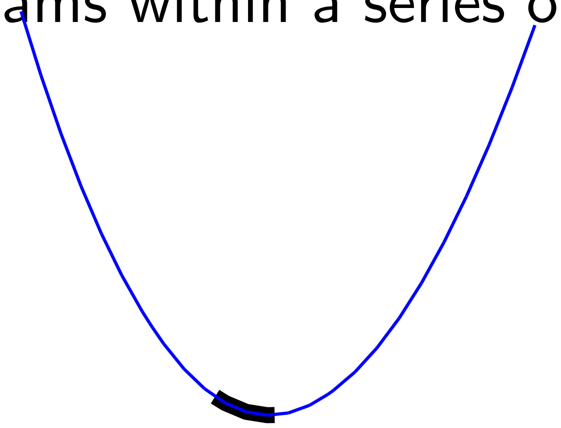
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



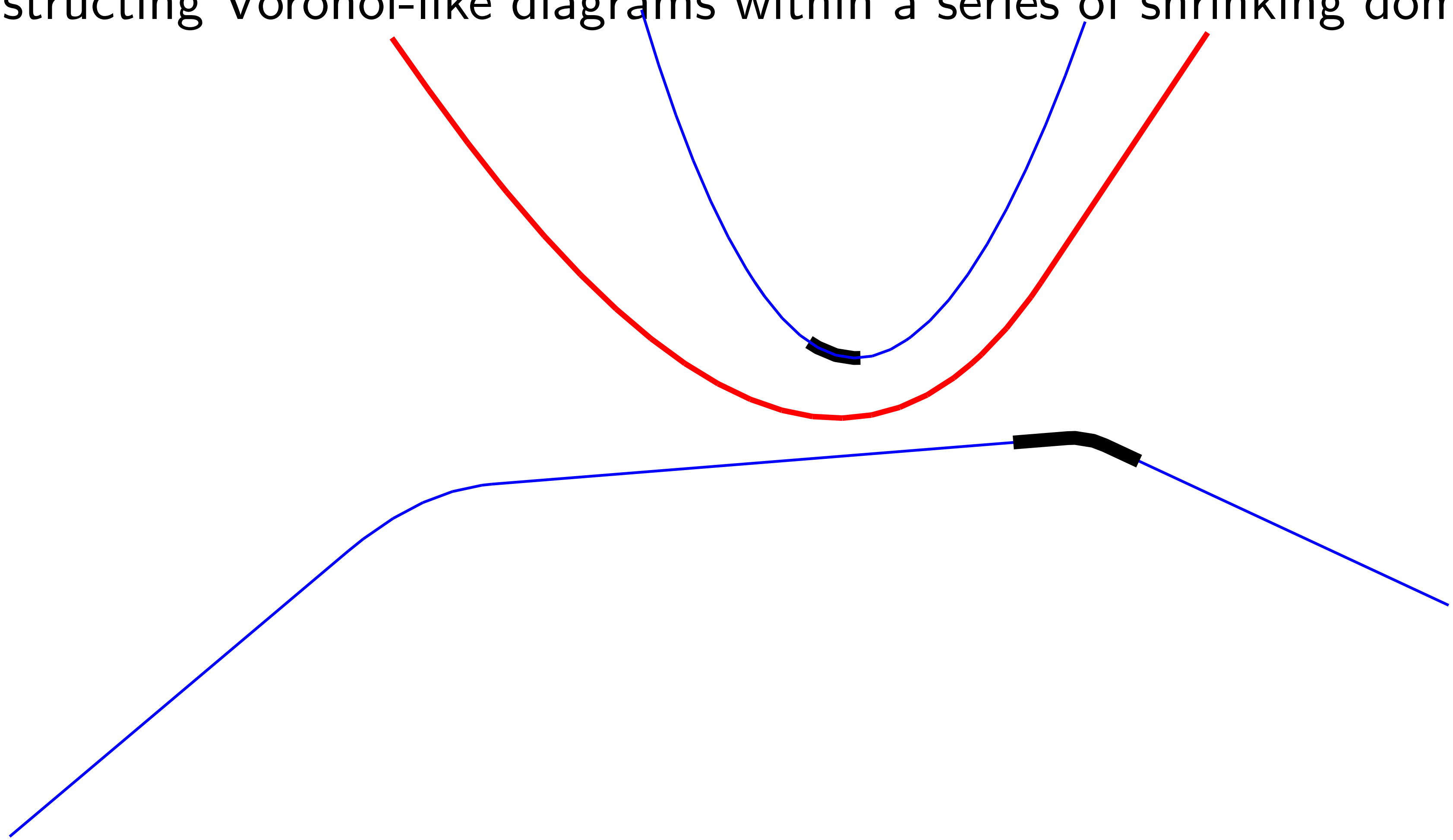
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



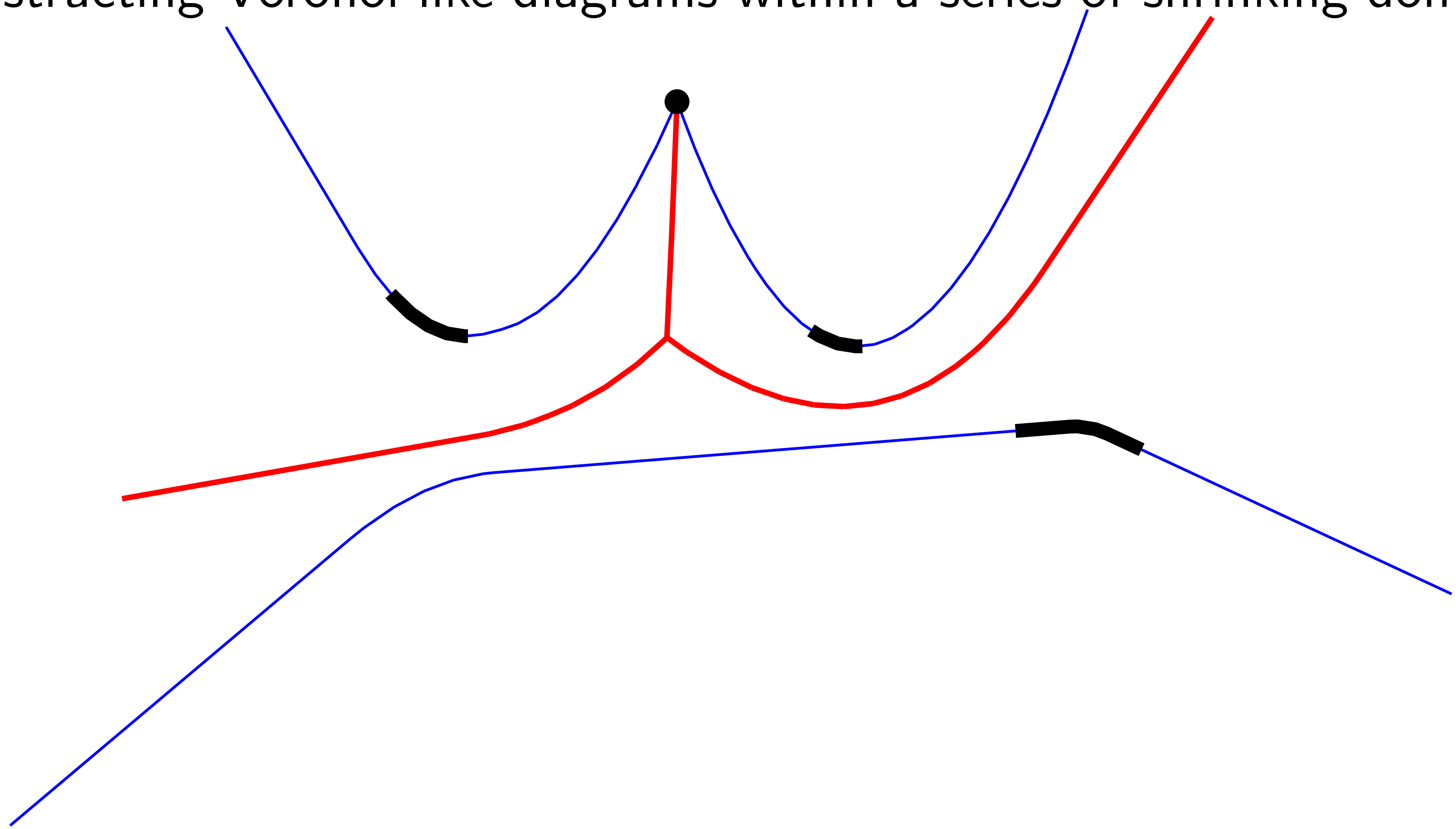
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



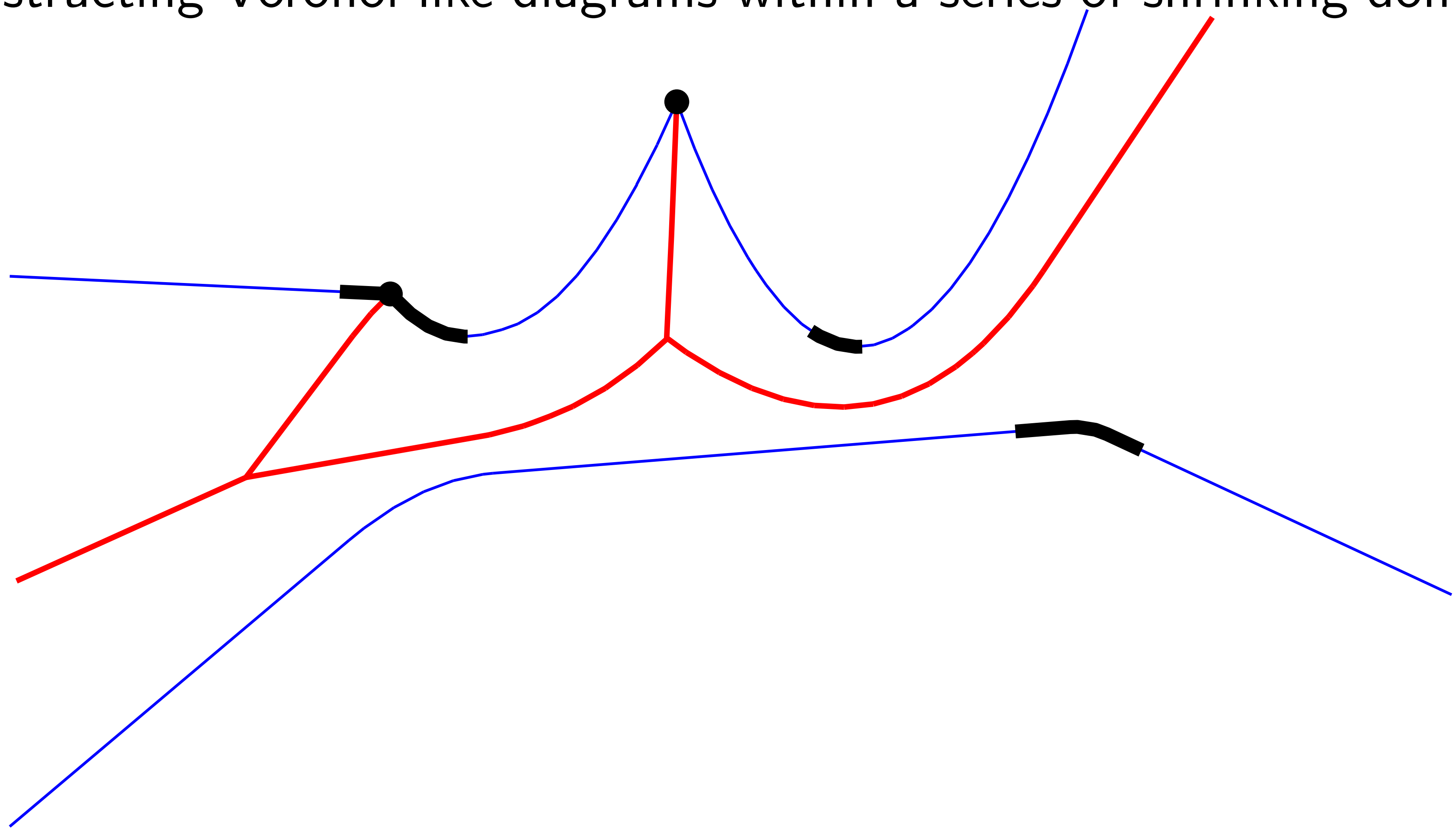
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



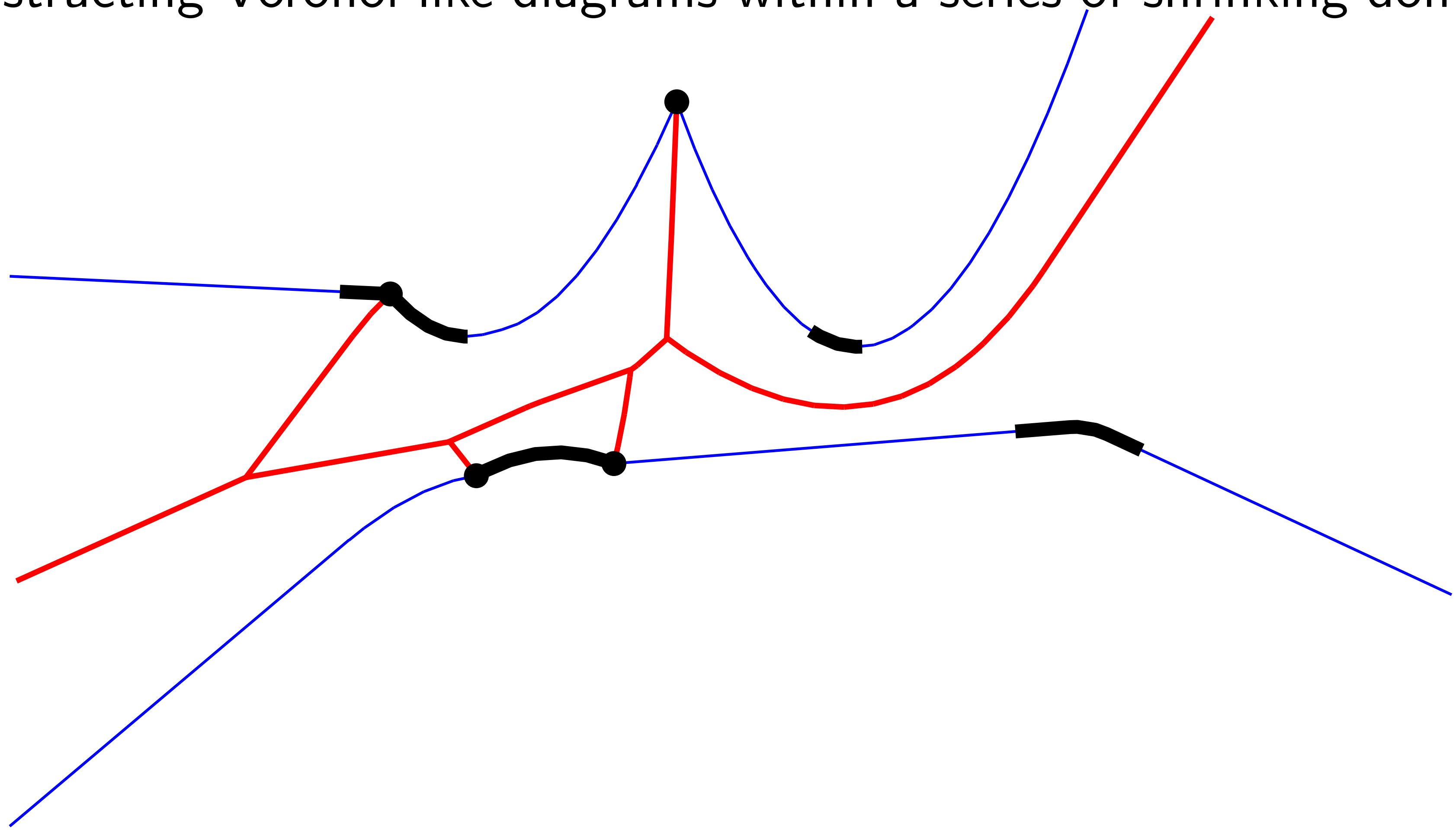
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



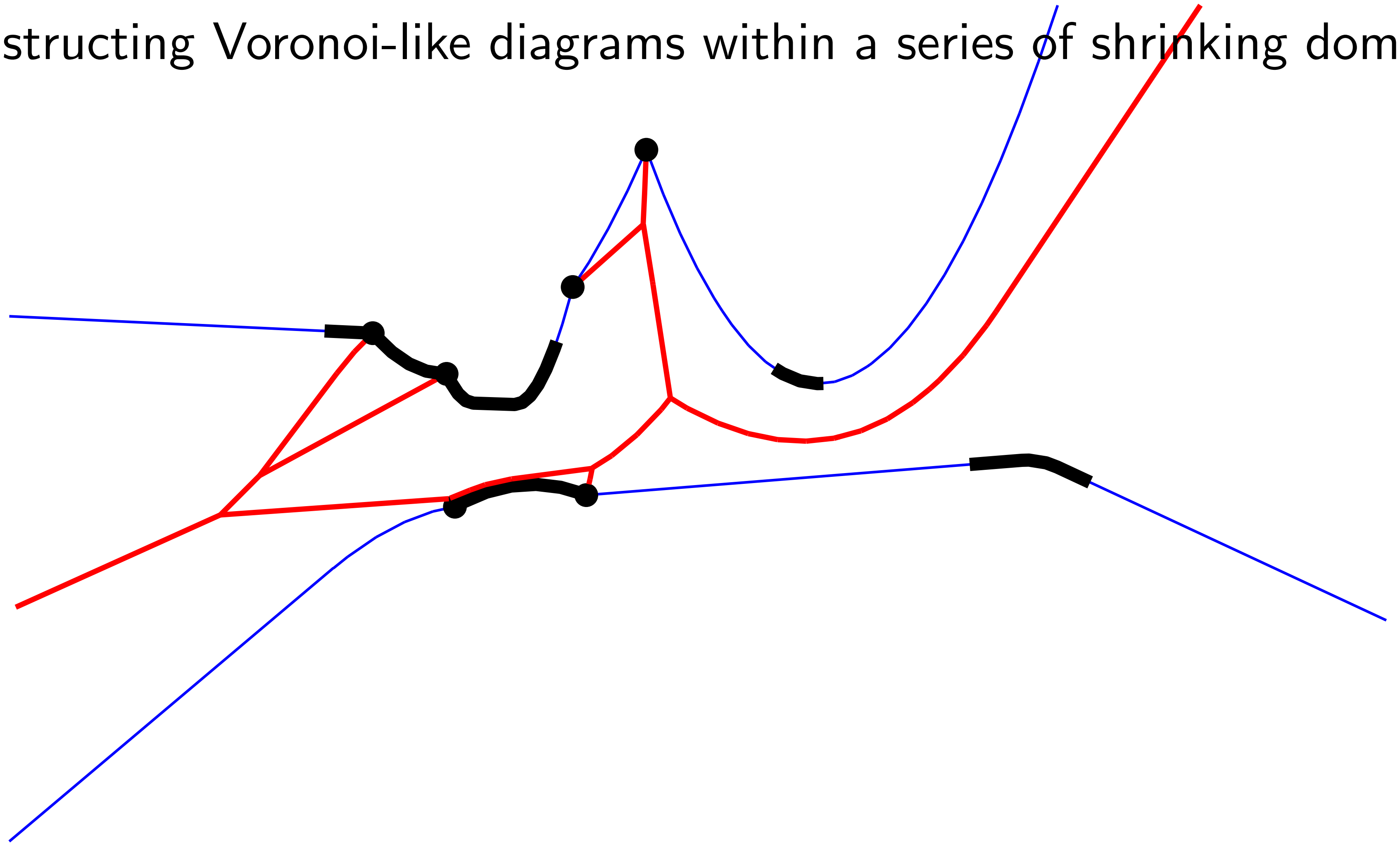
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



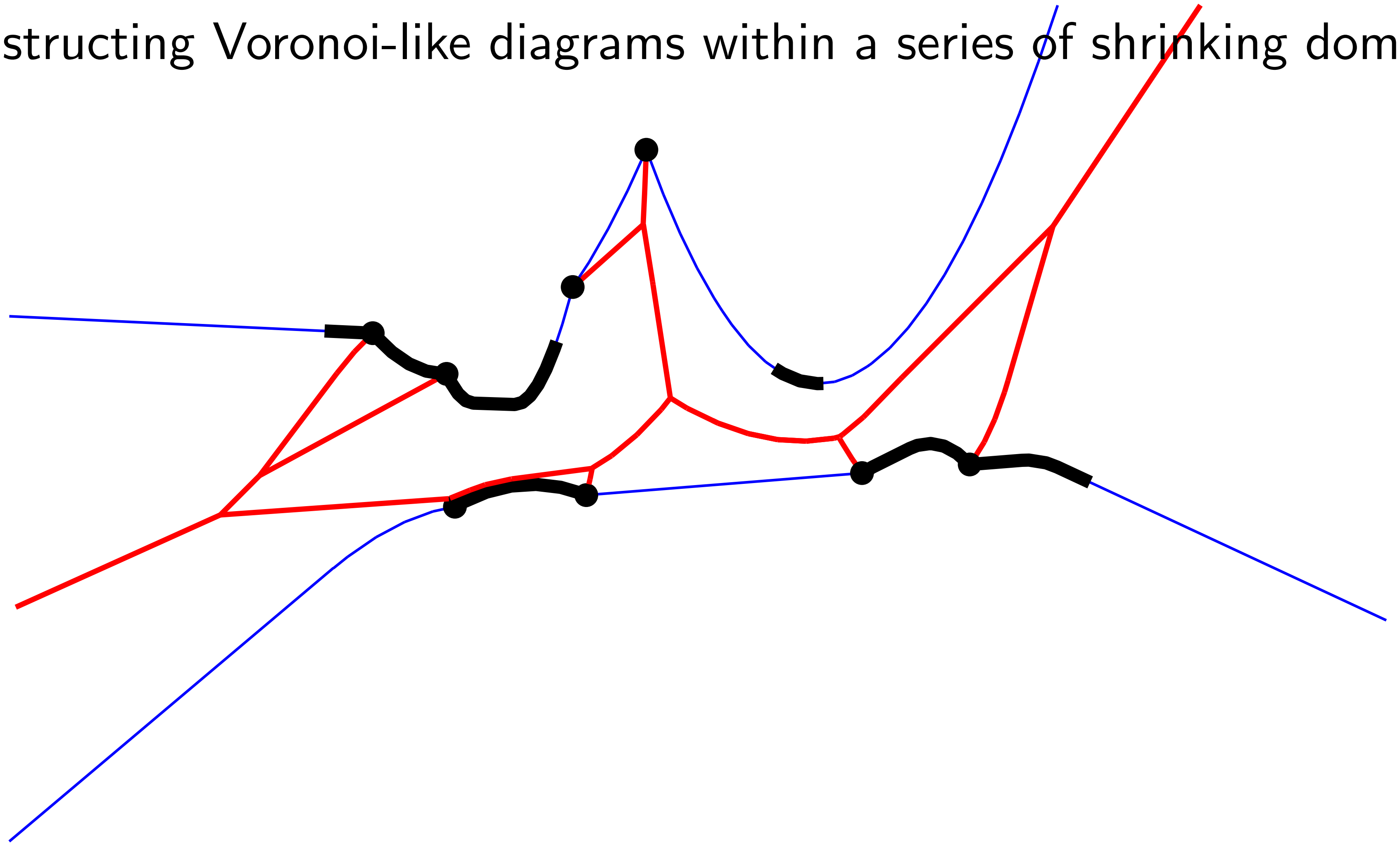
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



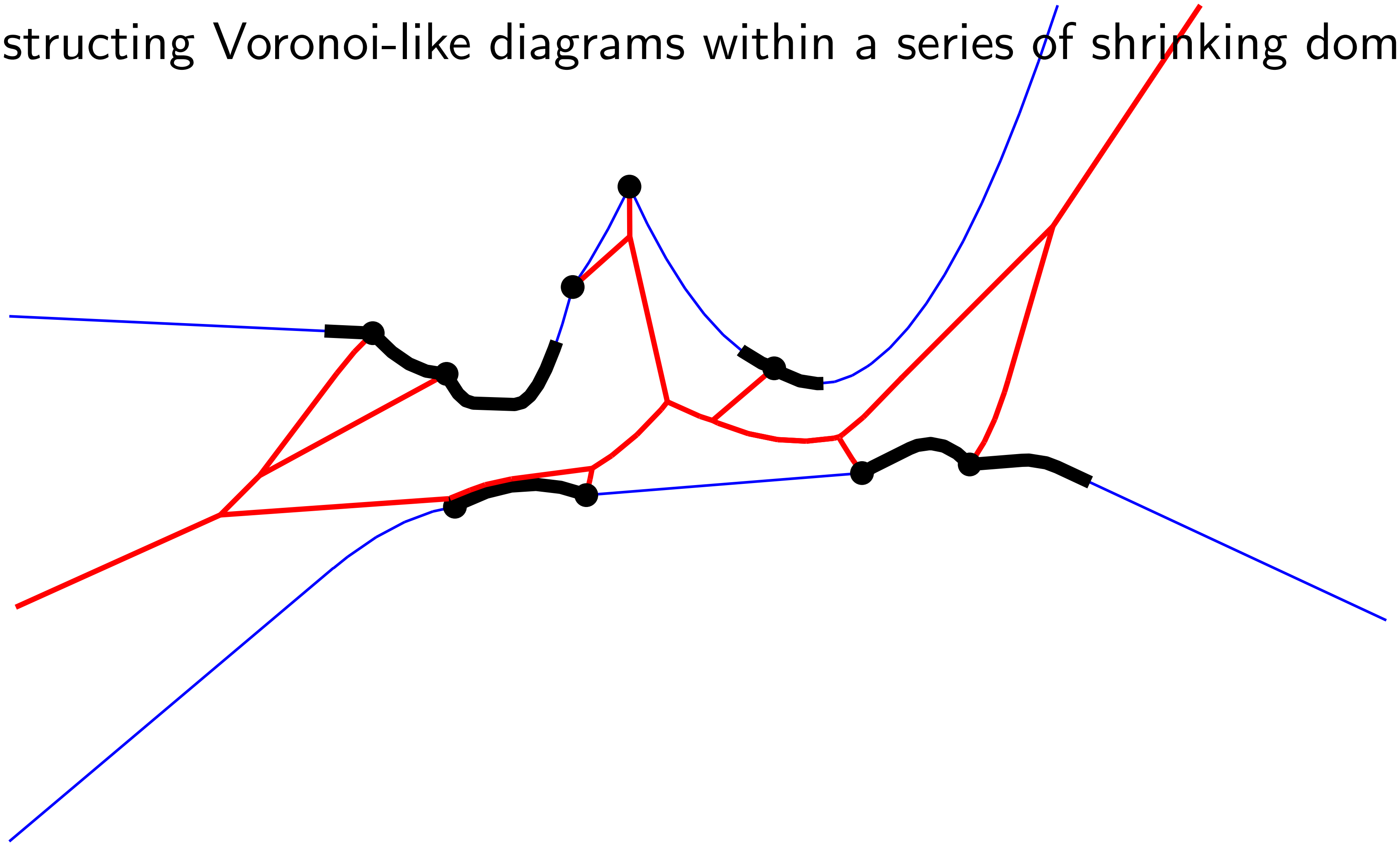
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



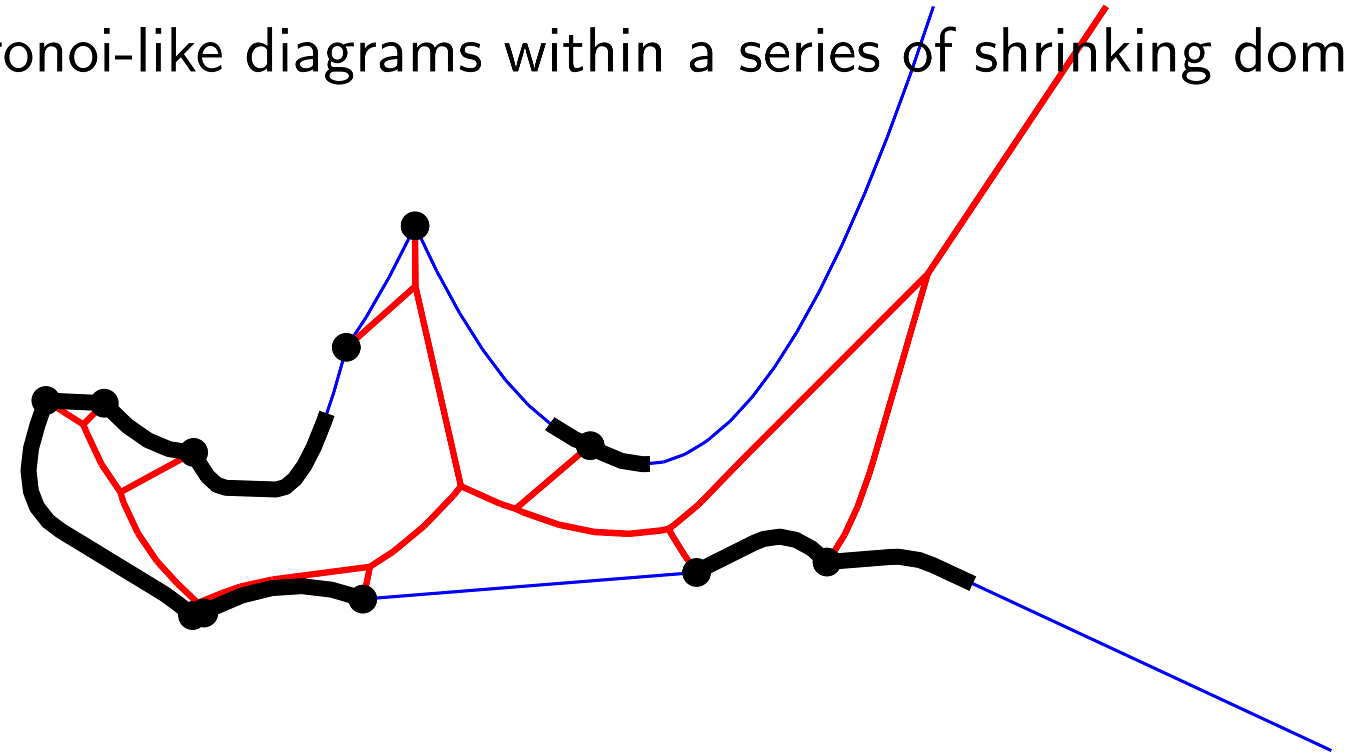
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



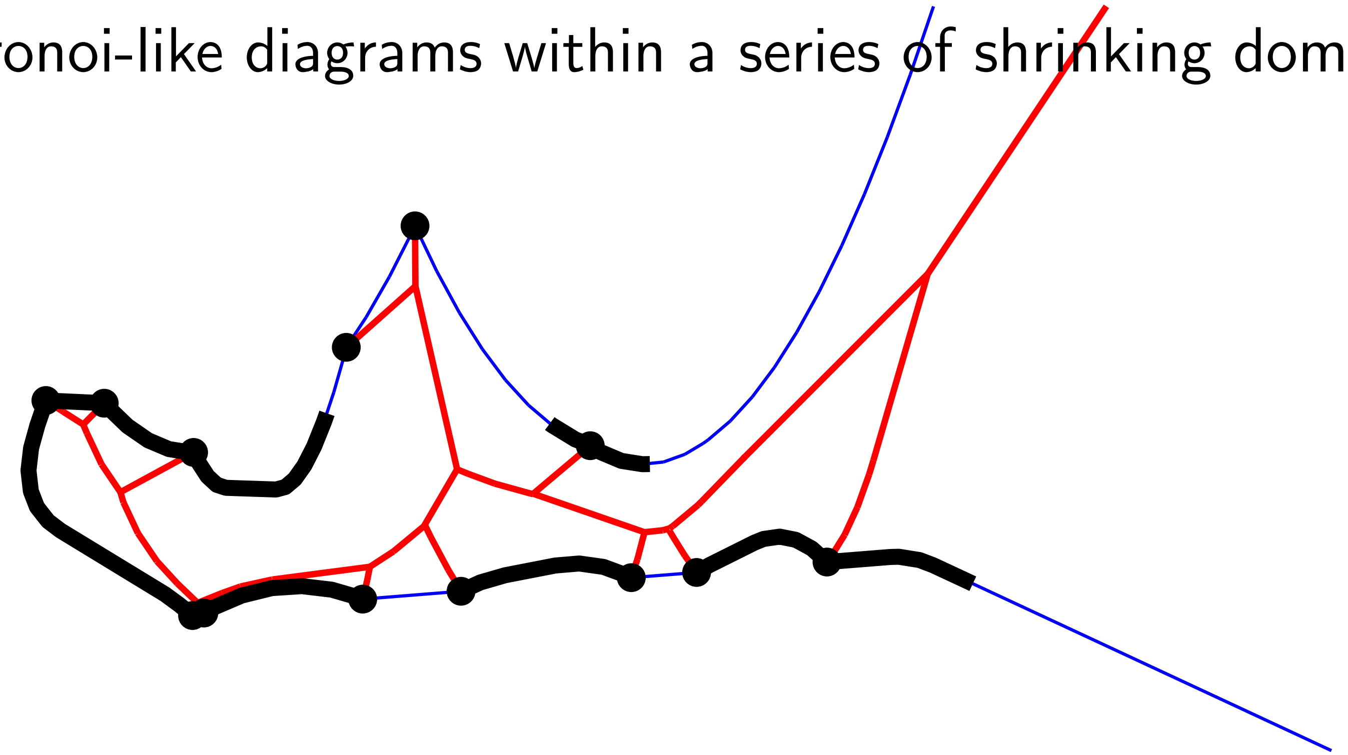
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



Because of auxiliary arcs, when we insert an arc, its neighbors need not be the ones of Phase 1. Need to trace auxiliary arcs (expected constant).

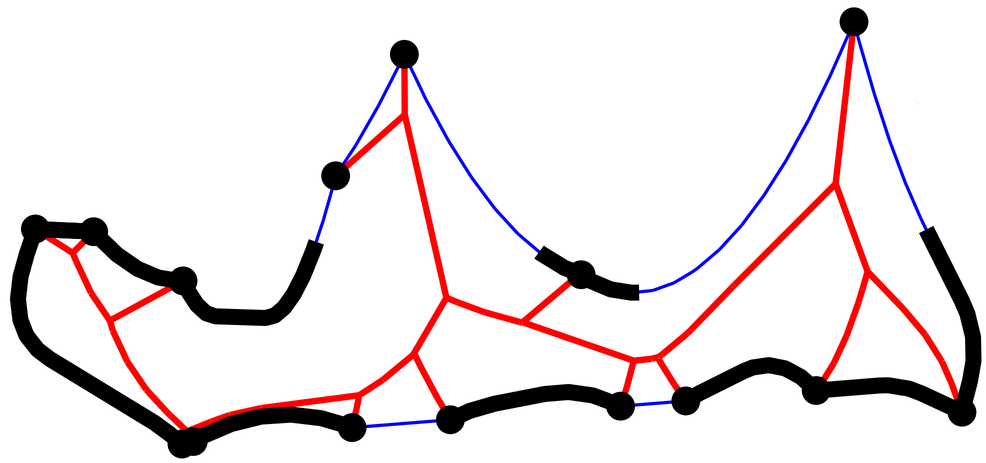
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



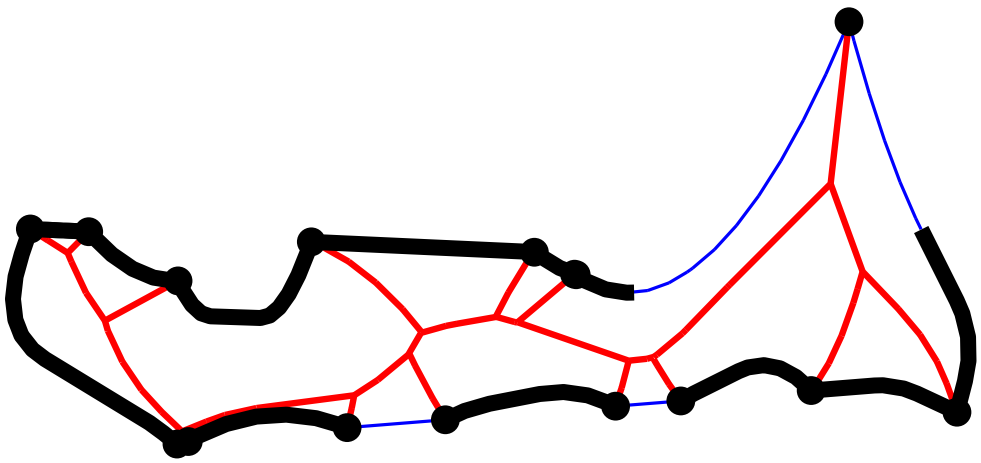
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



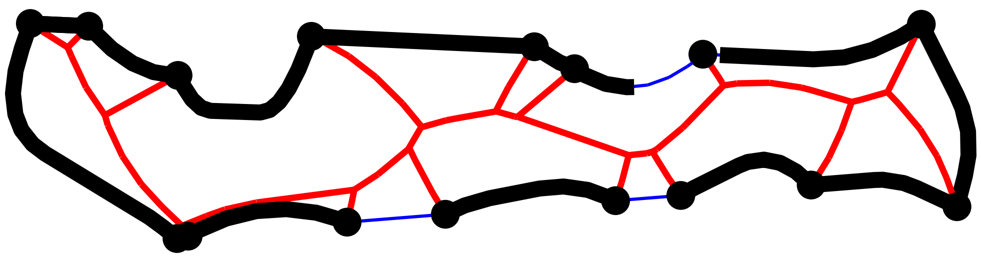
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



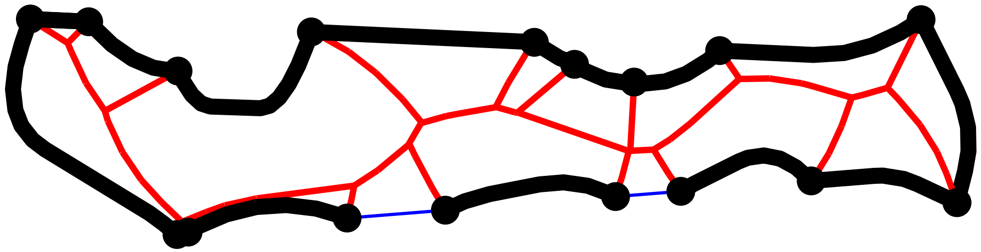
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



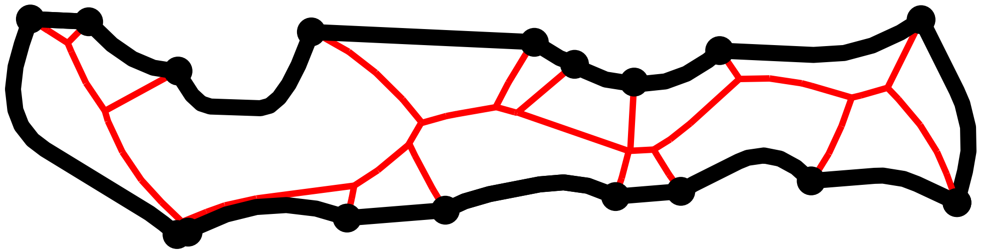
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



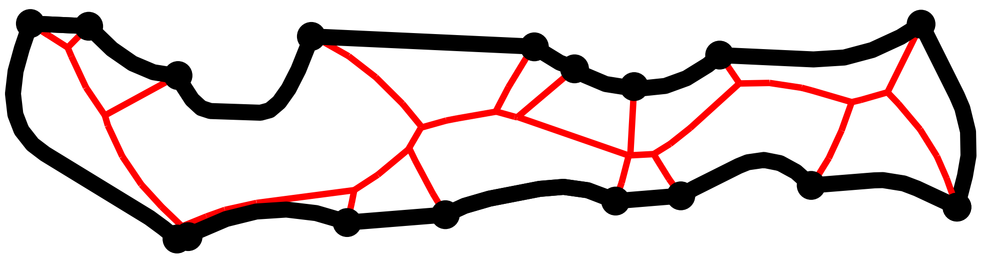
A randomized incremental algorithm

Consider a random permutation of the arcs $\mathcal{S}$.

Phase 1: Delete arcs from $\mathcal{S}$, recording their neighbors at time of deletion (inspired by Chew).

Phase 2: Insert the arcs in $\mathcal{S}$ in reverse order one by one.

...constructing Voronoi-like diagrams within a series of shrinking domains.



In the end we obtain $\mathcal{V}_l(\mathcal{S}) = \mathcal{V}(\mathcal{S}) = \mathcal{V}(S \setminus s) \cap \mathrm{VR}(s)$.

Università della Svizzera italiana

Our main result

Theorem:
Given $\mathcal{V}(S)$ and a site $s \in S$, the Voronoi diagram $\mathcal{V}(S \setminus \{s\})$ can be computed in expected time linear in the complexity of $\partial\mathsf{VR}(s)$.

Our main result

Theorem:
Given $\mathcal{V}(S)$ and a site $s \in S$, the Voronoi diagram $\mathcal{V}(S \setminus \{s\})$ can be computed in expected time linear in the complexity of ∂VR(s).

In the **proof** we use:

- **Lemma:** The boundary curve at step i consists of at most $2i$ arcs.
- **Lemma:** The expected number of arcs that are visited during one insertion is constant.
- Backward analysis, similar to line segments [Khramtcova, Papadopoulou, ISAAC 2015]

Future work / Open problems

- Deterministic linear-time algorithm for deletion in abstract Voronoi diagrams ?

  Currently, we are exploring using Voronoi-like diagrams in the framework of [Aggarwal, Guibas, Saxe and Shor, DCG 1989].

Future work / Open problems

- Deterministic linear-time algorithm for deletion in abstract Voronoi diagrams ?

  Currently, we are exploring using Voronoi-like diagrams in the framework of [Aggarwal, Guibas, Saxe and Shor, DCG 1989].

- Other uses of Voronoi-like diagrams...

Future work / Open problems

- Deterministic linear-time algorithm for deletion in abstract Voronoi diagrams ?

  Currently, we are exploring using Voronoi-like diagrams in the framework of [Aggarwal, Guibas, Saxe and Shor, DCG 1989].

- Other uses of Voronoi-like diagrams...

Thank you for your attention!