Learning with Nets and Meshes

Don Sheehy

Thanks to my collaborators:

Benoit Hudson
Gary Miller
Todd Phillips
Steve Oudot

Point Clouds in low to medium dimensional ambient space.

Rule of thumb: $d!$ or 2^{d^2} is okay but n^d is *not*.

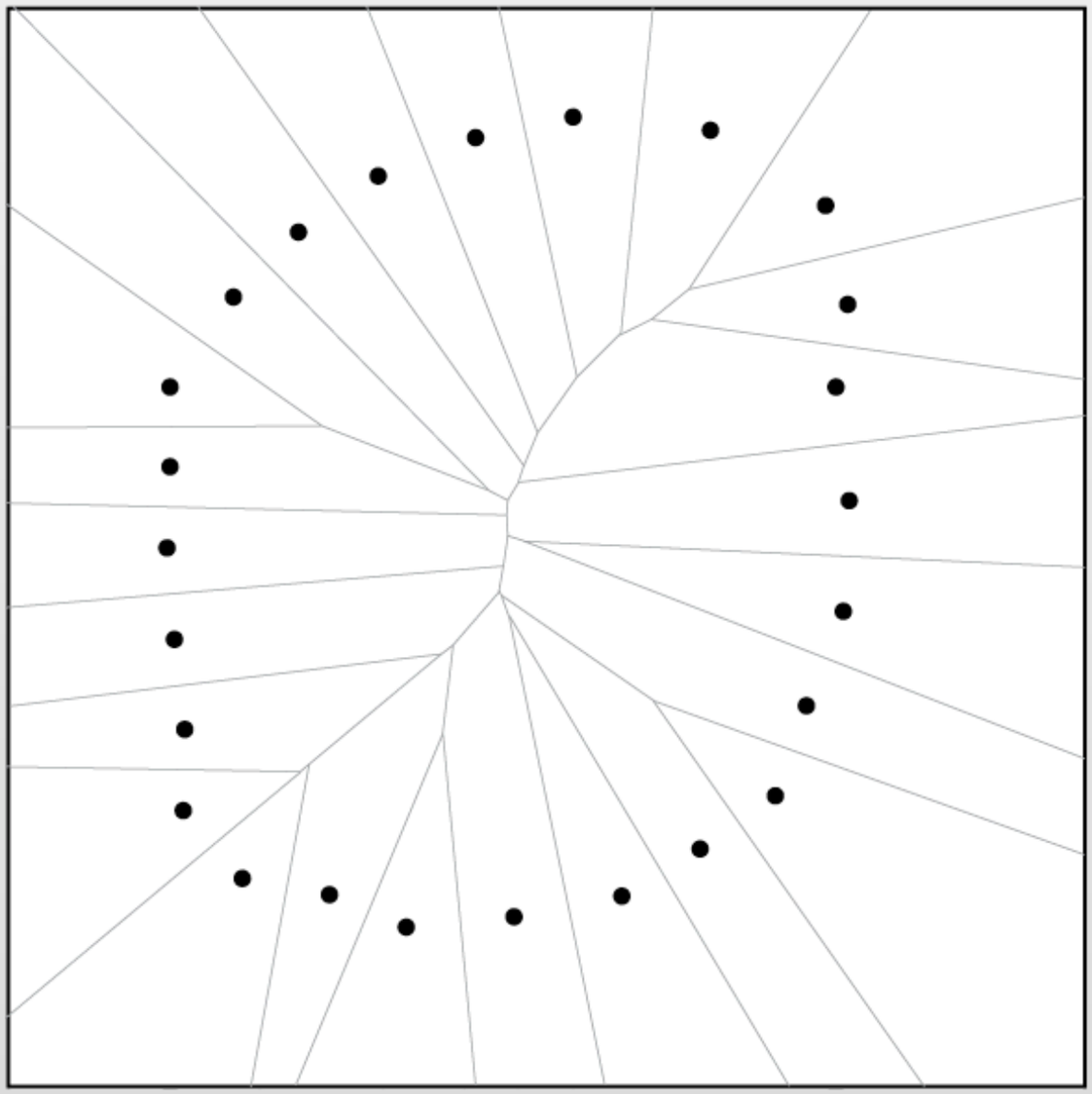
There are many geometric inference problems that could benefit from meshing.

Discretize Space
Approximate Functions
Adapt to density
Describe the space *around* the input.

Meshing

Input: $P \subset \mathbb{R}^d \qquad n = |P|$

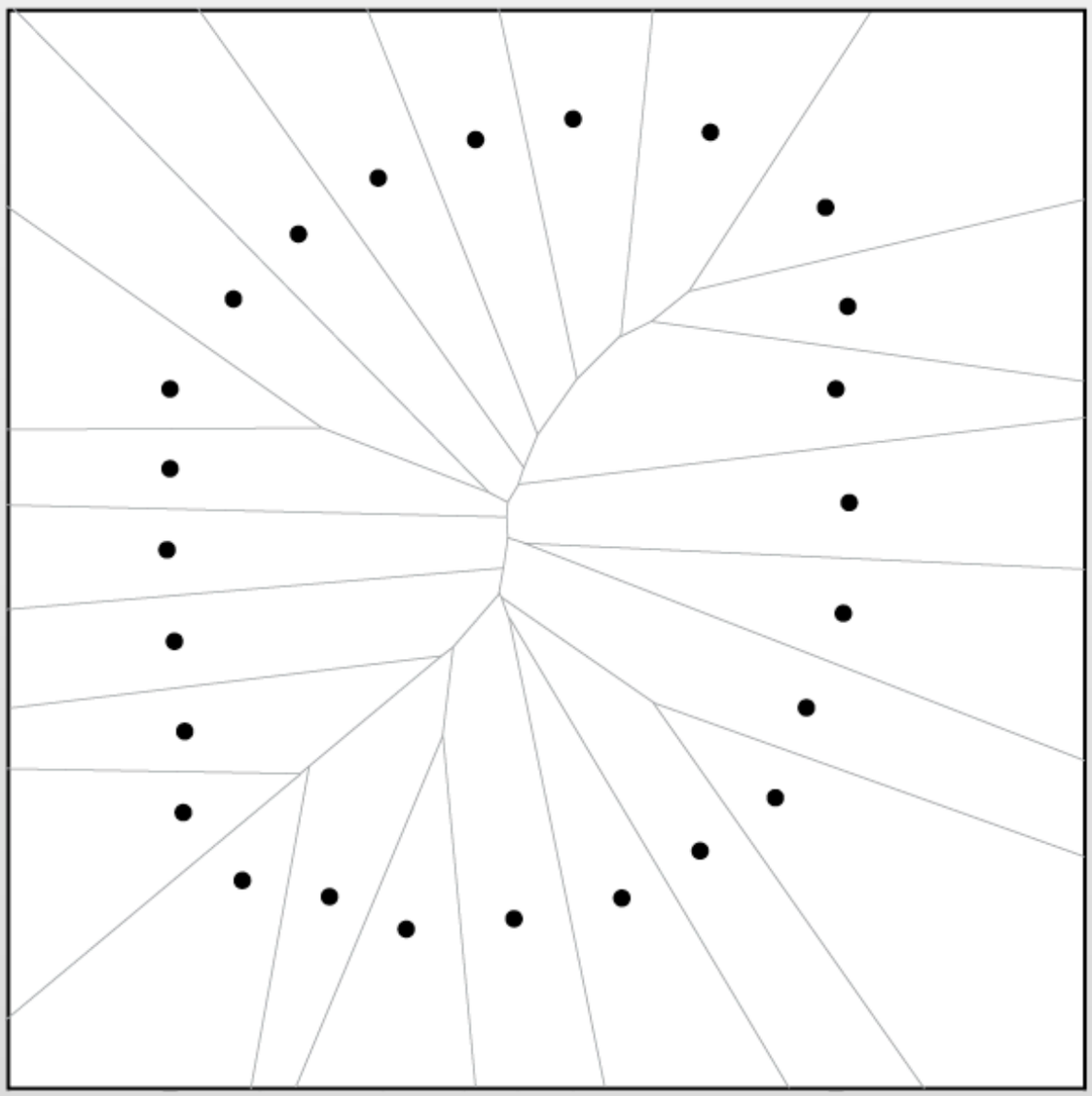
Output: $M \supset P$ with a "nice" Voronoi diagram

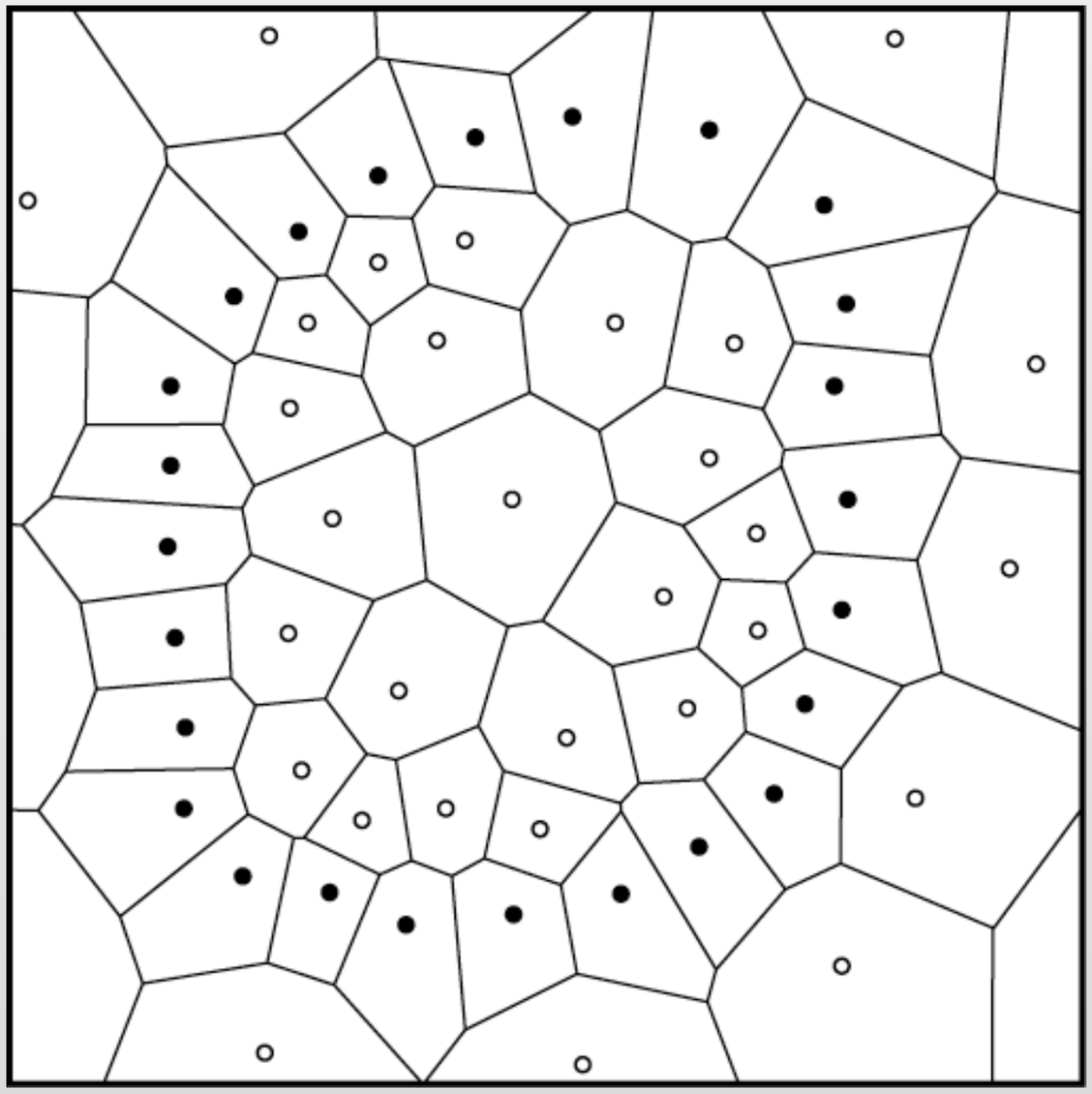


Meshing

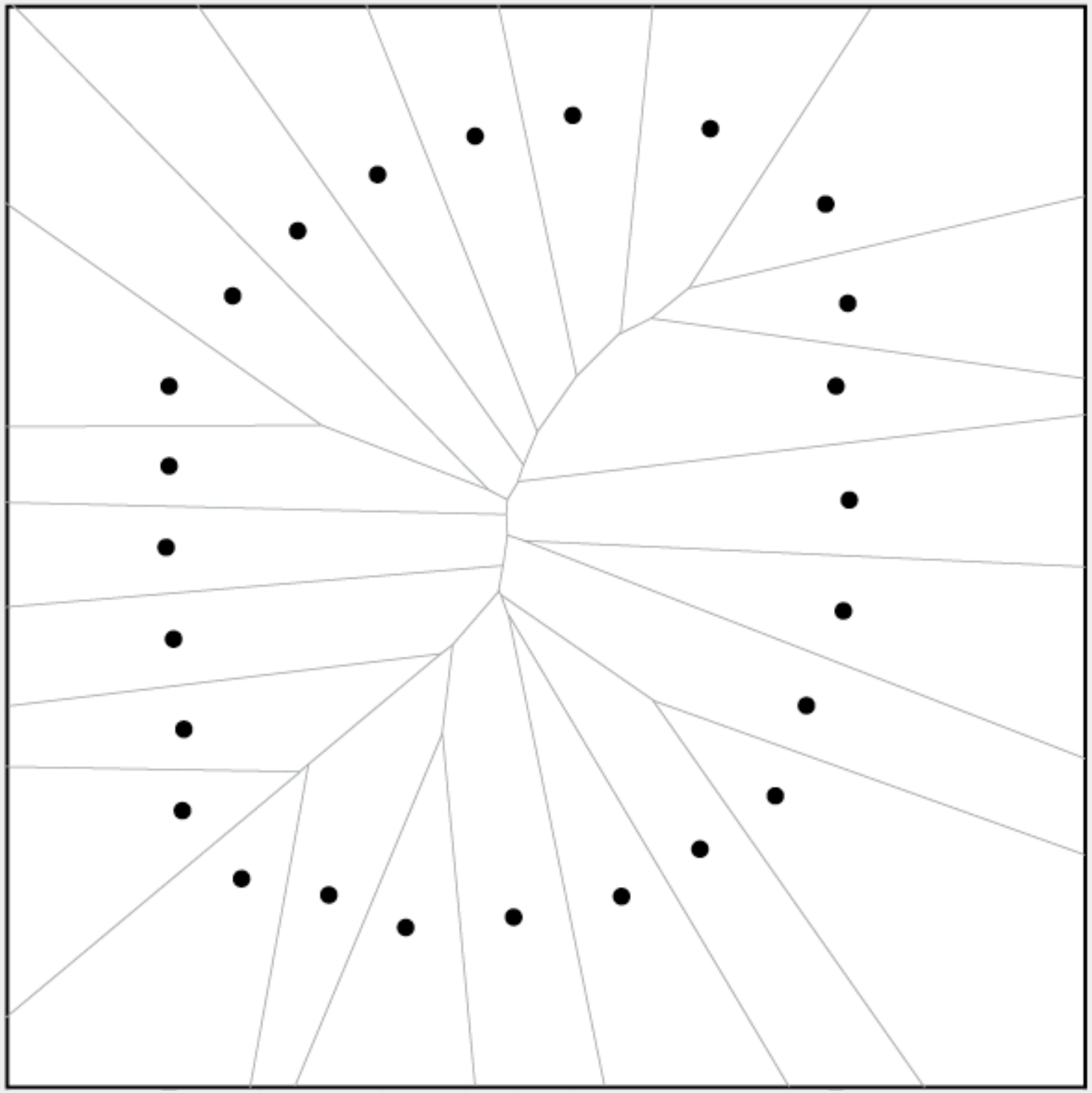
Input: $P \subset \mathbb{R}^d \qquad n = |P|$

Output: $M \supset P$ with a "nice" Voronoi diagram

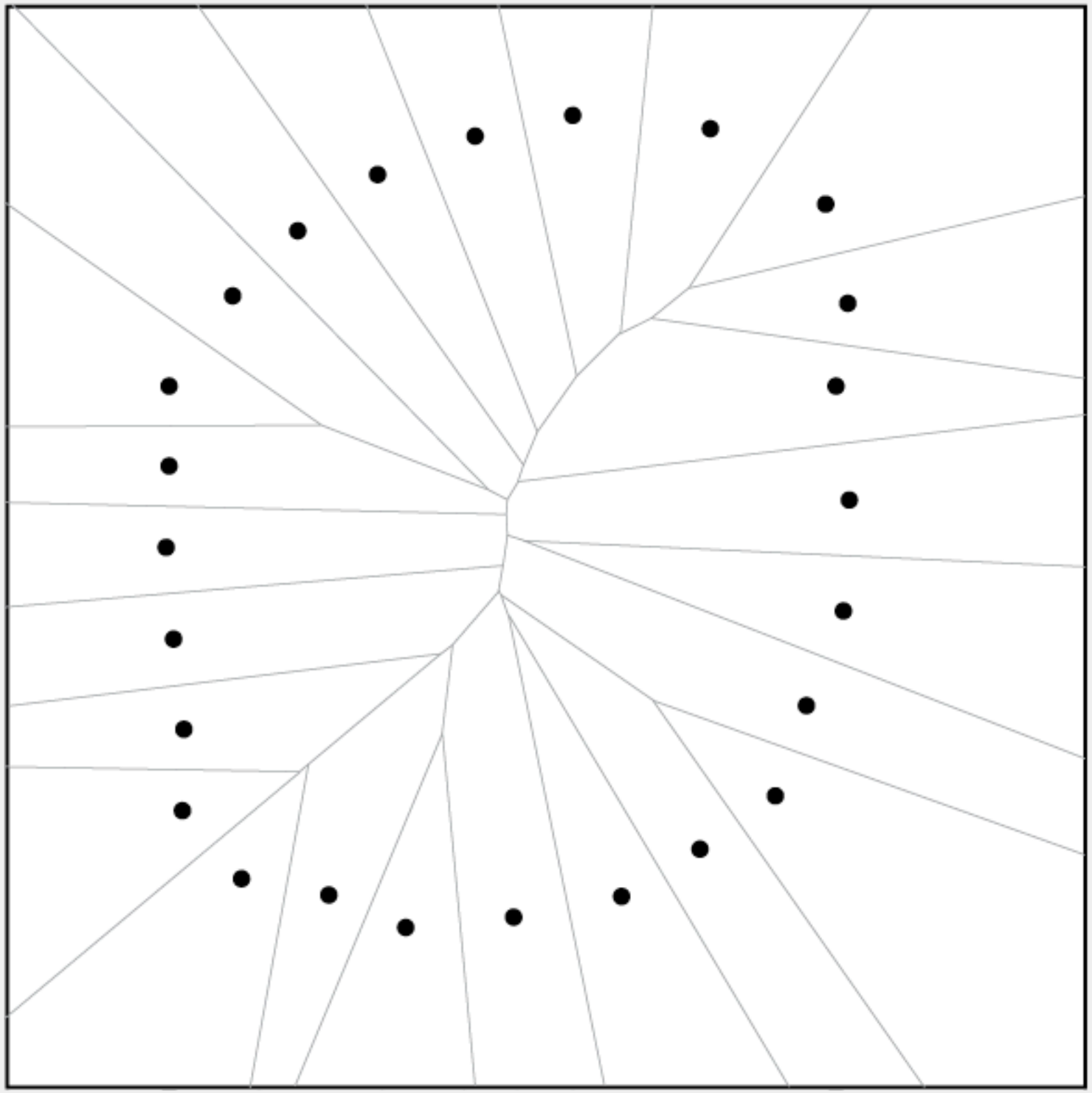




Points, offsets, homology, and persistence.

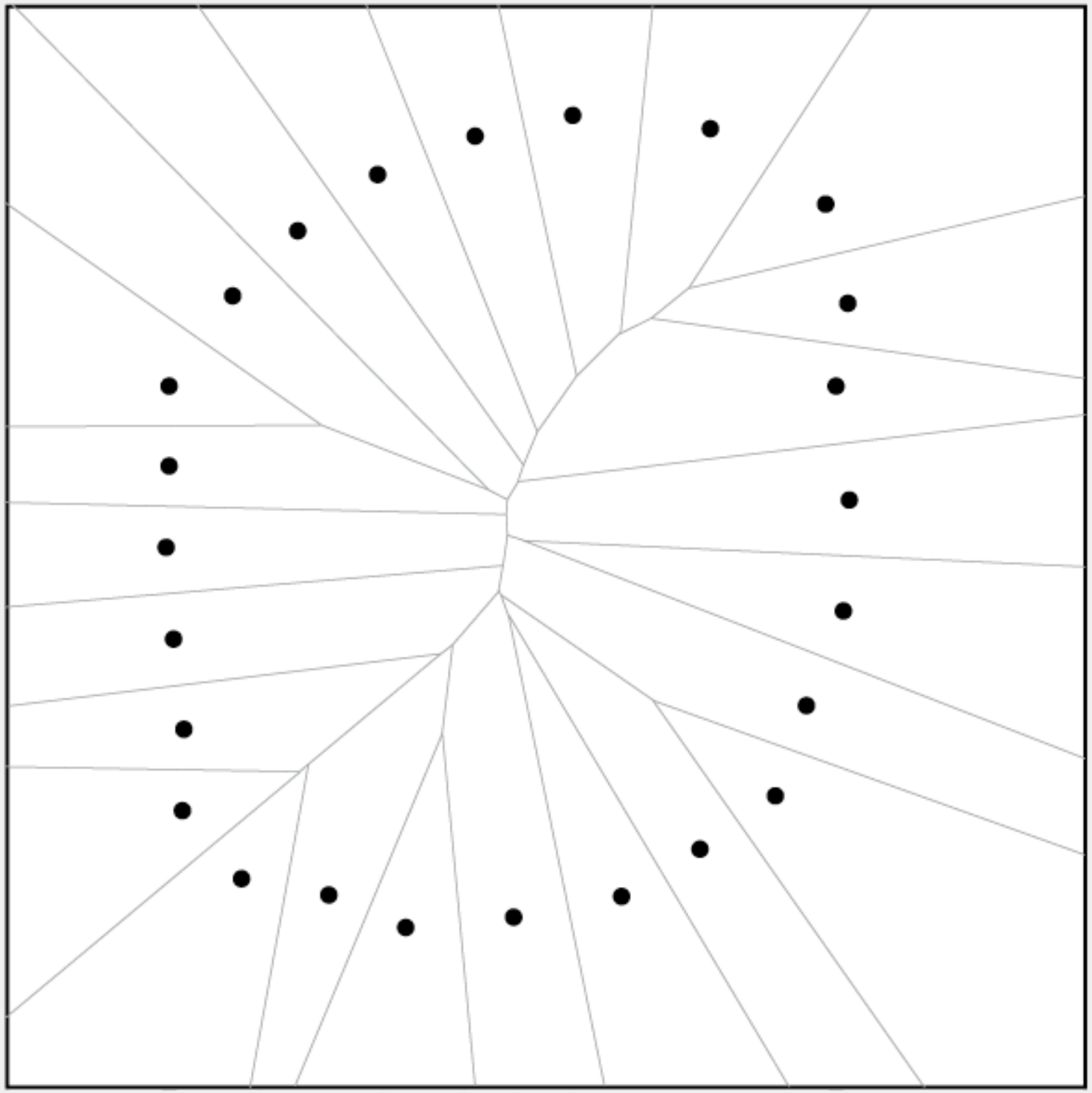


Points, offsets, homology, and persistence.



Input: $P \subset \mathbb{R}^d$

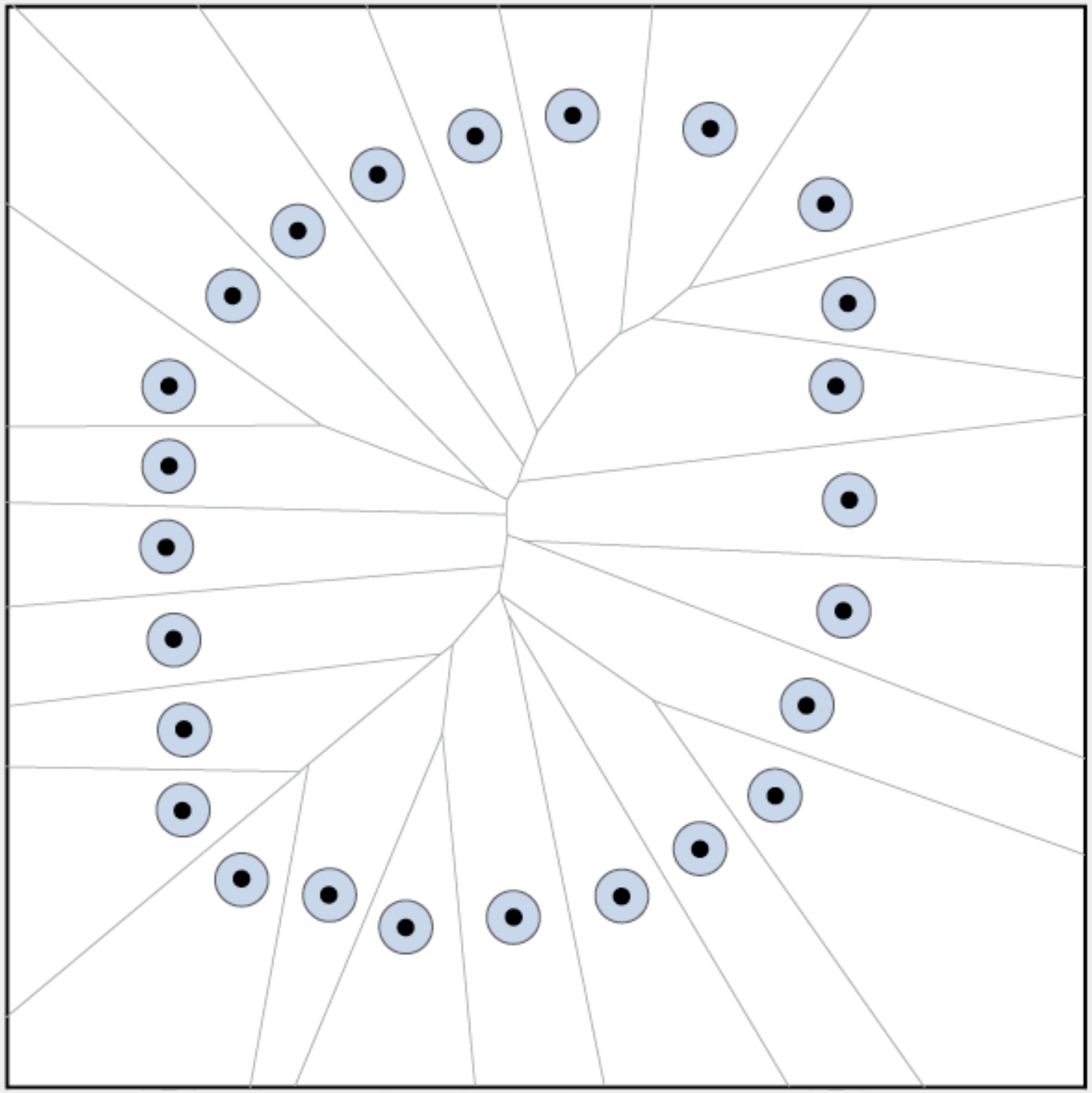
Points, offsets, homology, and persistence.



Input: $P \subset \mathbb{R}^d$

$$P^\alpha = \bigcup_{p \in P} \text{ball}(p, \alpha)$$

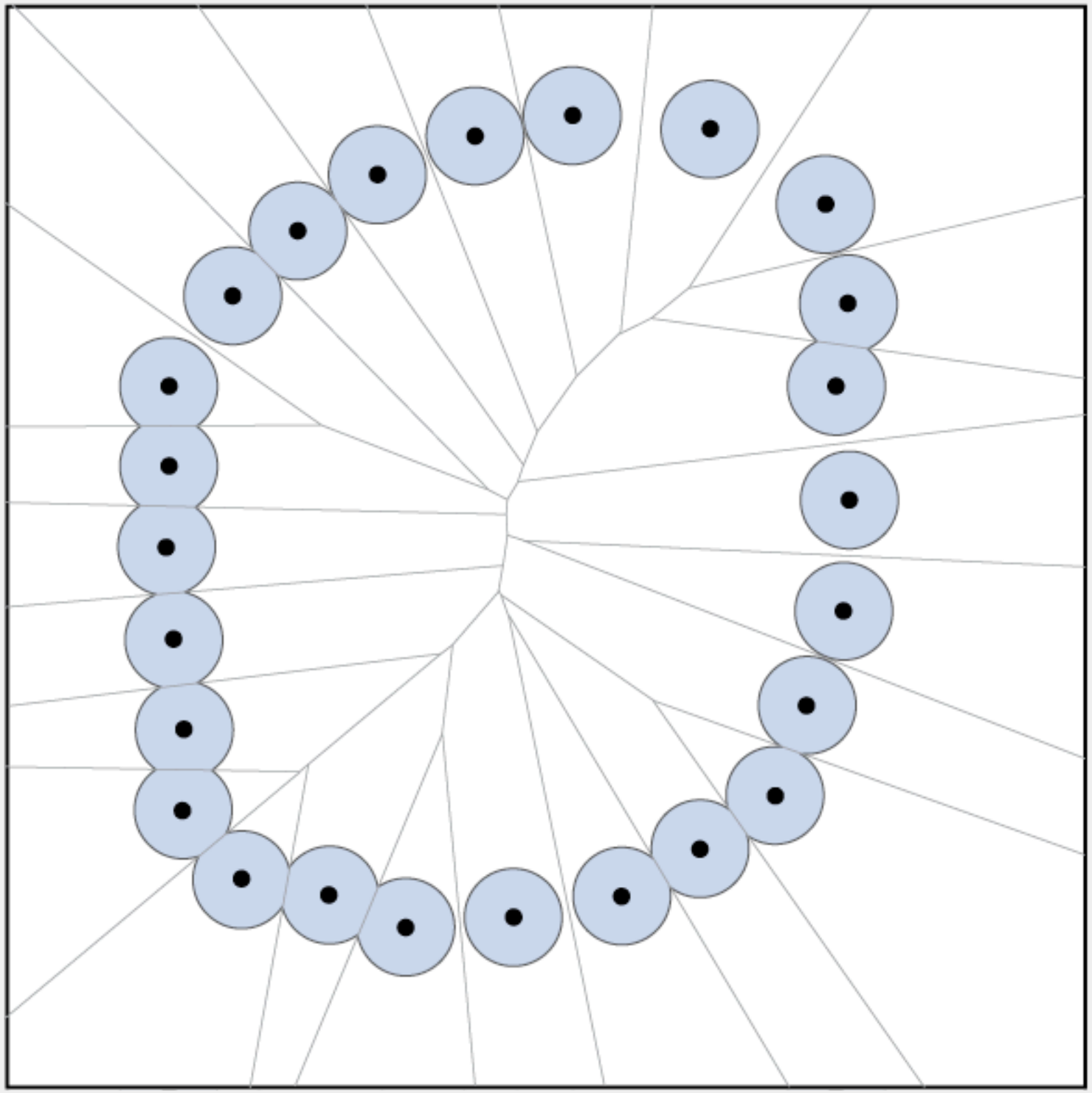
Points, offsets, homology, and persistence.



Input: $P \subset \mathbb{R}^d$

$$P^\alpha = \bigcup_{p \in P} \text{ball}(p, \alpha)$$

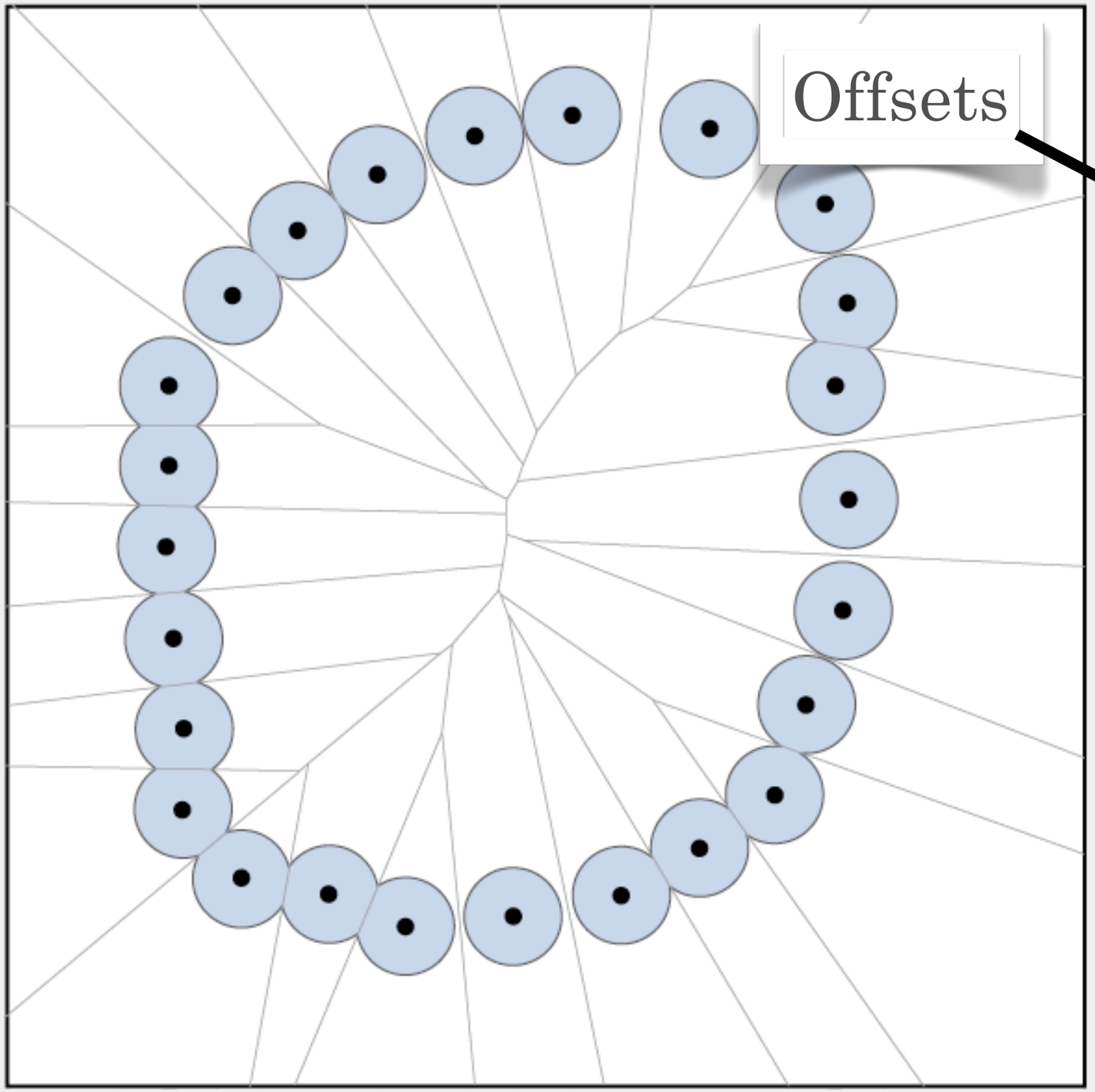
Points, offsets, homology, and persistence.



Input: $P \subset \mathbb{R}^d$

$$P^\alpha = \bigcup_{p \in P} \text{ball}(p, \alpha)$$

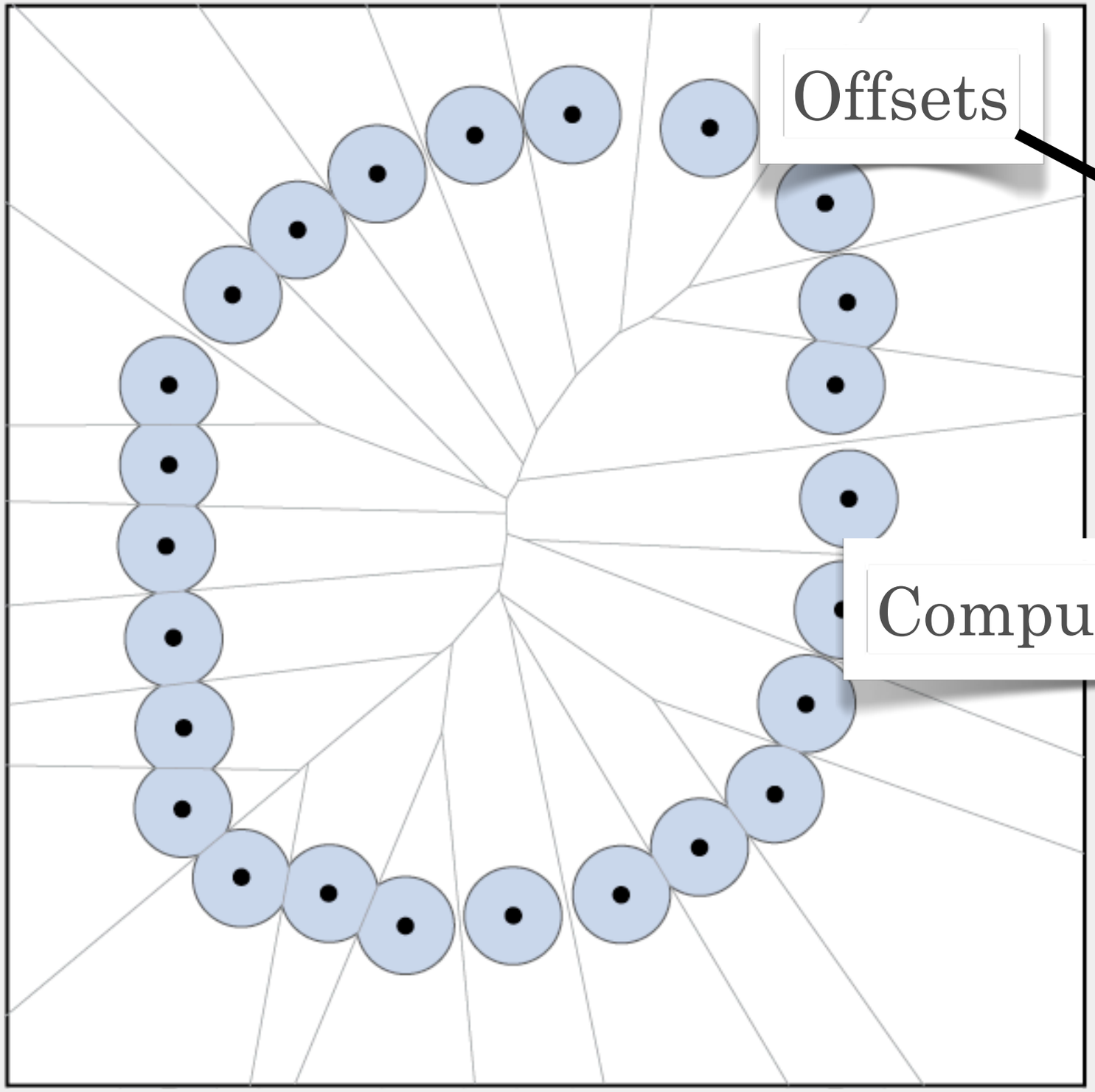
Points, offsets, homology, and persistence.



Input: $P \subset \mathbb{R}^d$

$$P^\alpha = \bigcup_{p \in P} \text{ball}(p, \alpha)$$

Points, offsets, homology, and persistence.

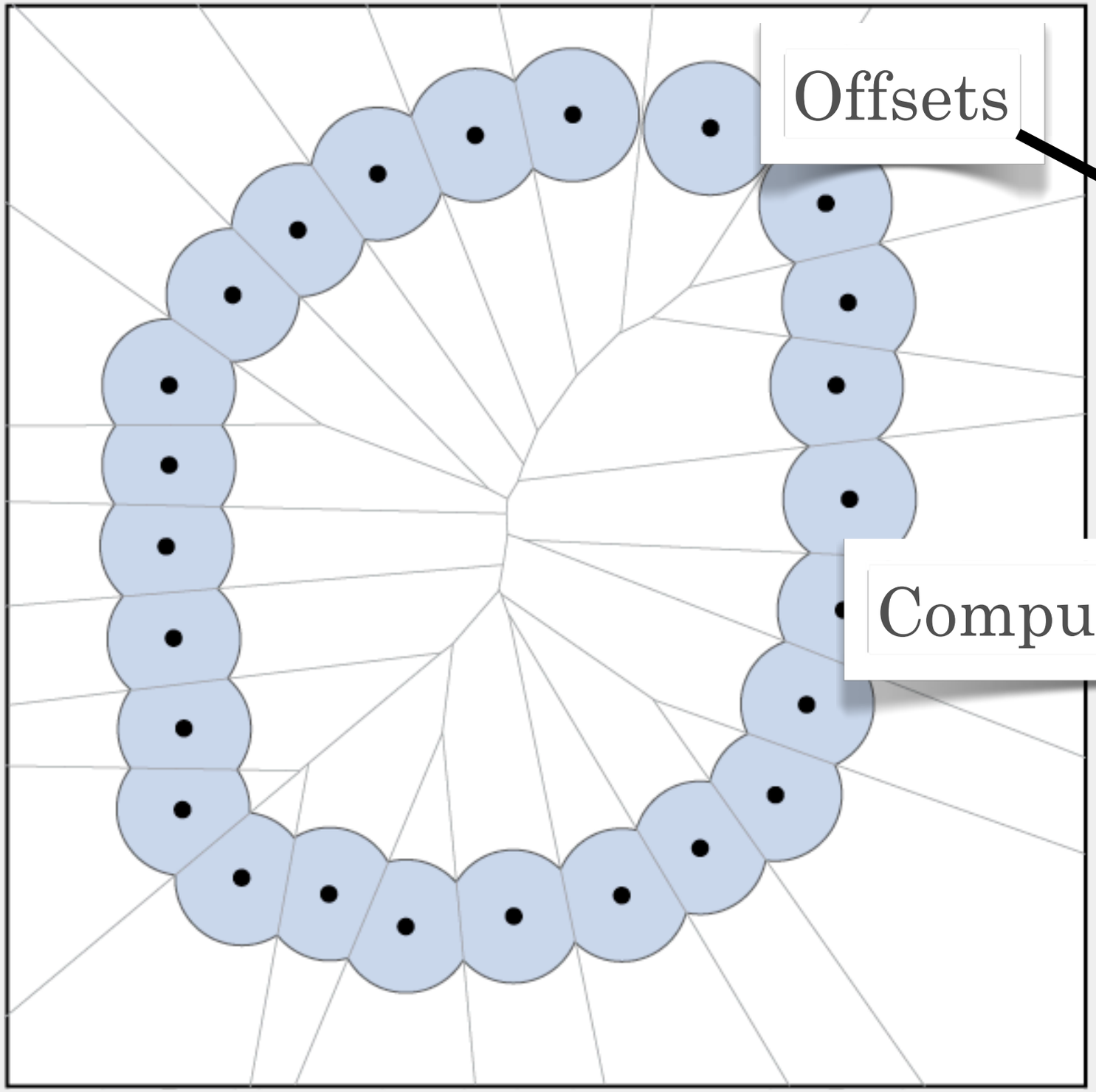


Input: $P \subset \mathbb{R}^d$

$$P^\alpha = \bigcup_{p \in P} \text{ball}(p, \alpha)$$

Compute the **Homology**

Points, offsets, homology, and persistence.

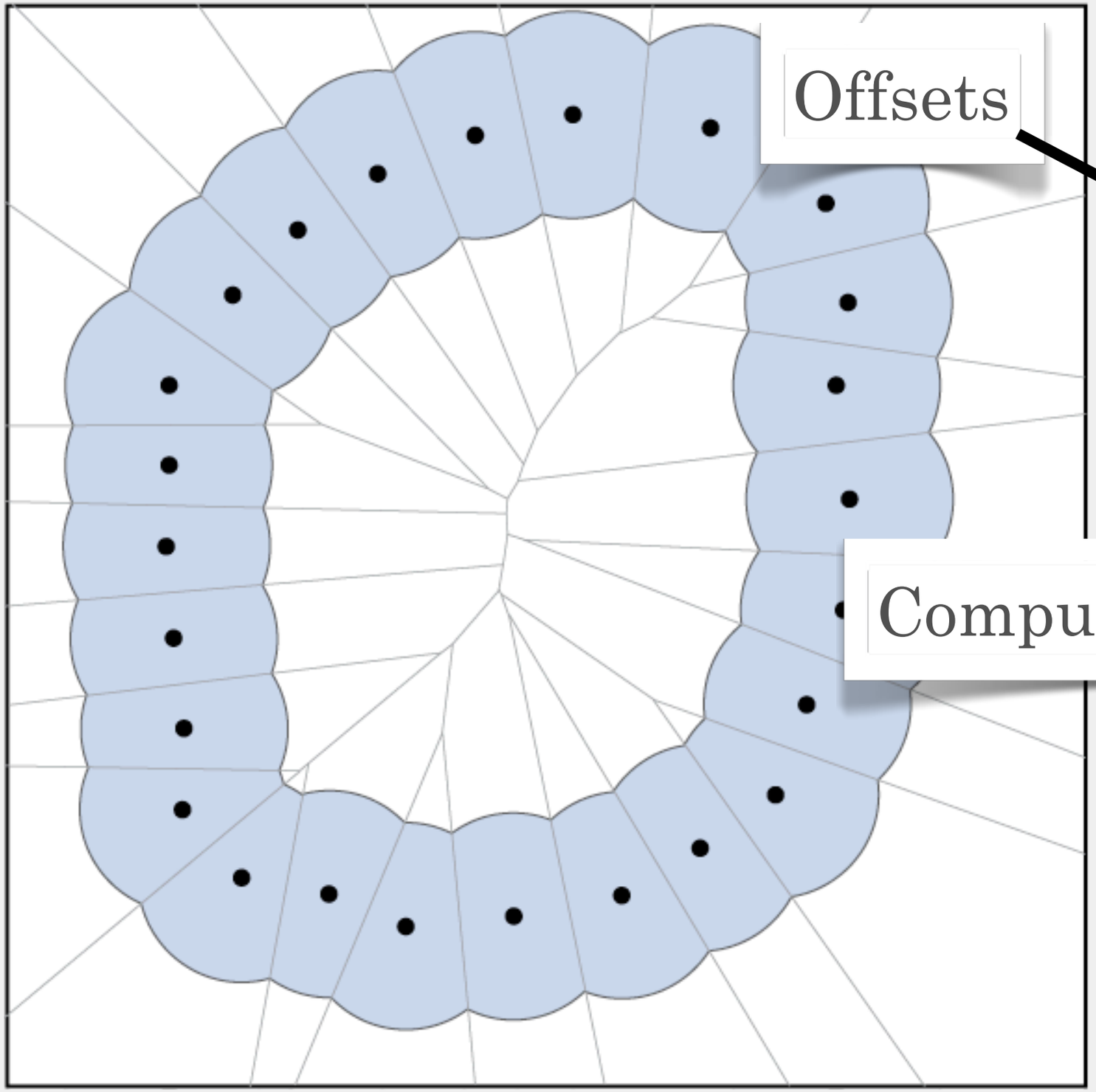


Input: $P \subset \mathbb{R}^d$

$$P^\alpha = \bigcup_{p \in P} \text{ball}(p, \alpha)$$

Compute the **Homology**

Points, offsets, homology, and persistence.

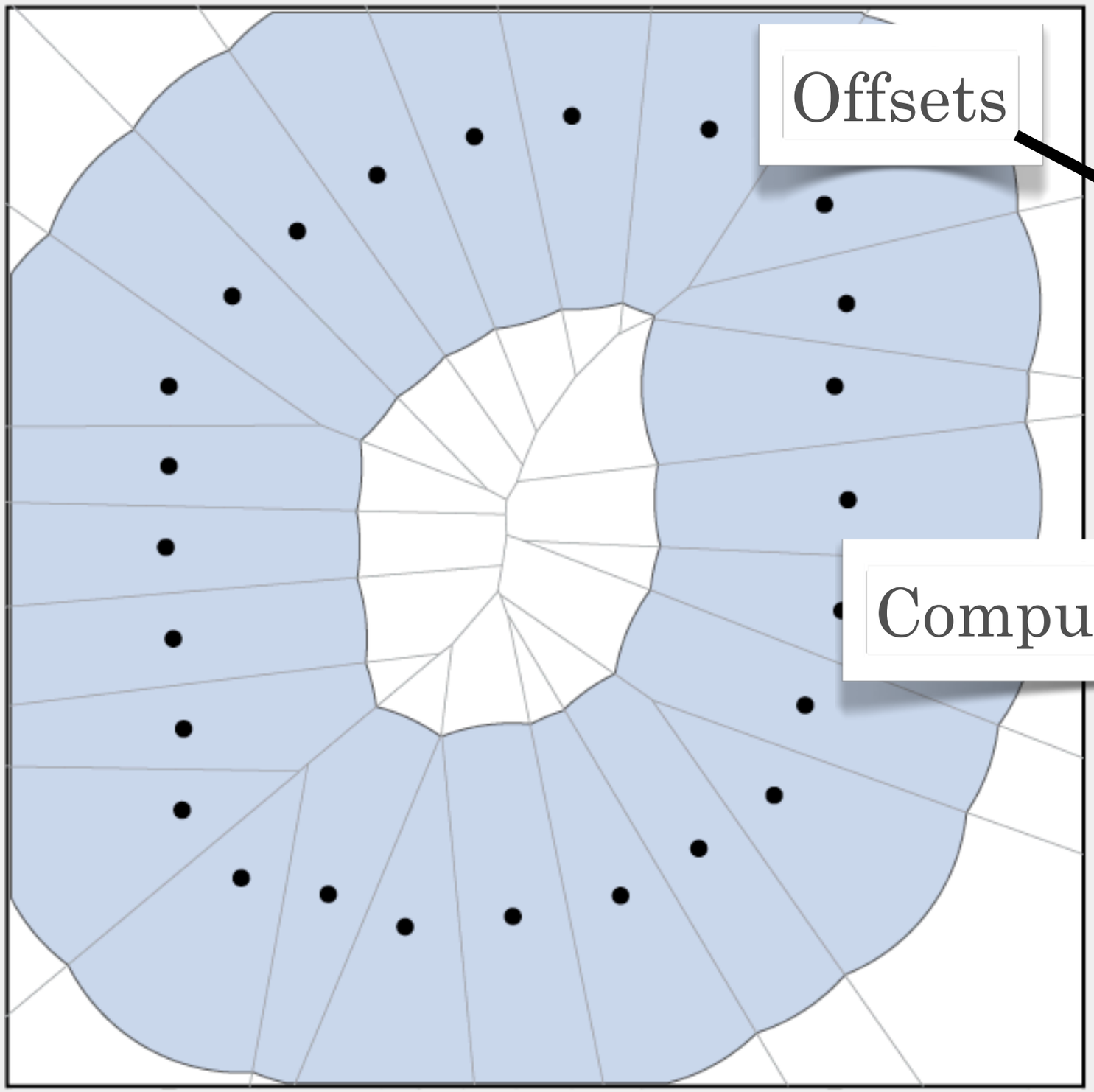


Input: $P \subset \mathbb{R}^d$

$$P^\alpha = \bigcup_{p \in P} \text{ball}(p, \alpha)$$

Compute the **Homology**

Points, offsets, homology, and persistence.

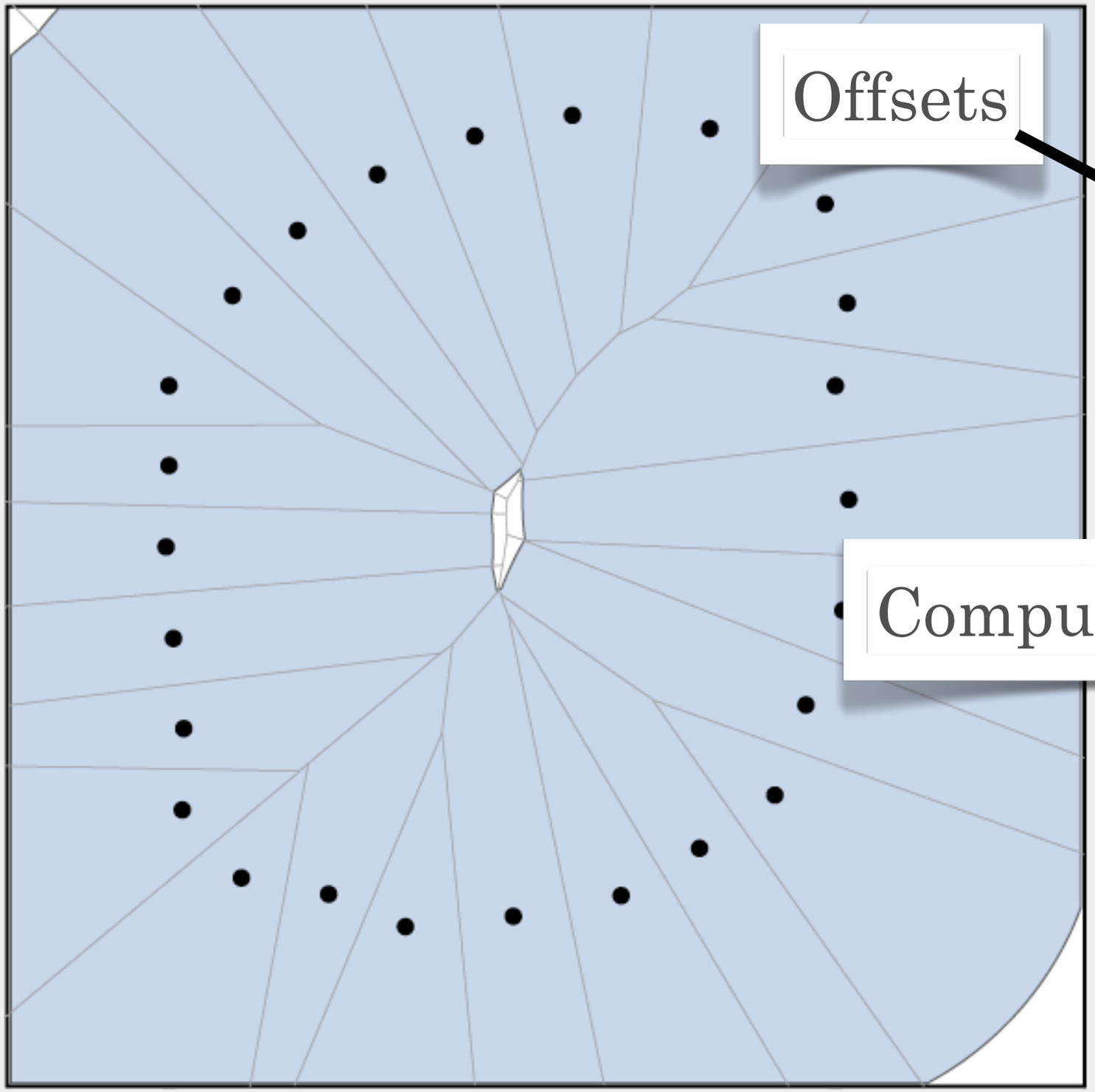


Input: $P \subset \mathbb{R}^d$

$$P^\alpha = \bigcup_{p \in P} \text{ball}(p, \alpha)$$

Compute the **Homology**

Points, offsets, homology, and persistence.

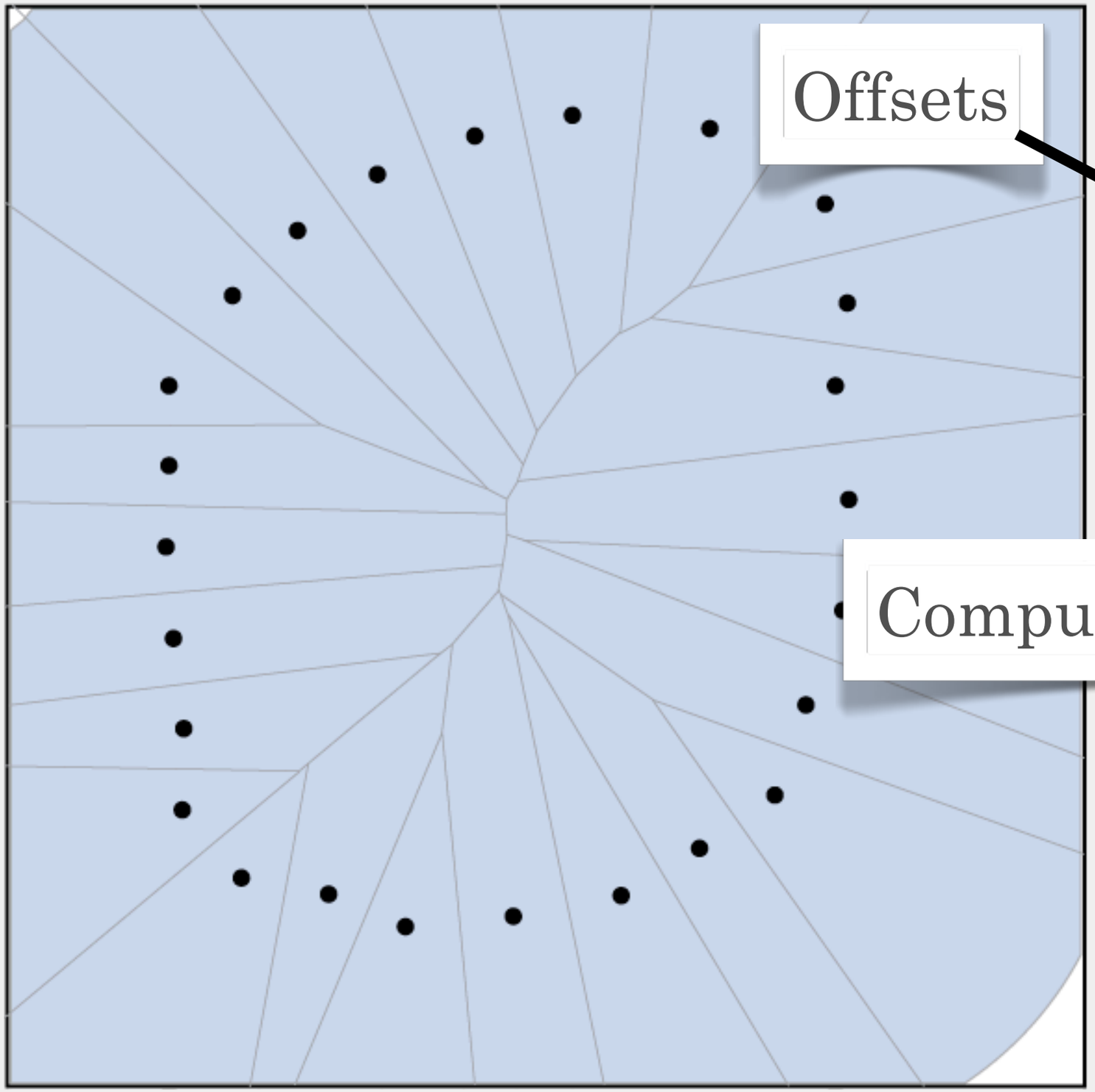


Input: $P \subset \mathbb{R}^d$

$$P^\alpha = \bigcup_{p \in P} \text{ball}(p, \alpha)$$

Compute the **Homology**

Points, offsets, homology, and persistence.

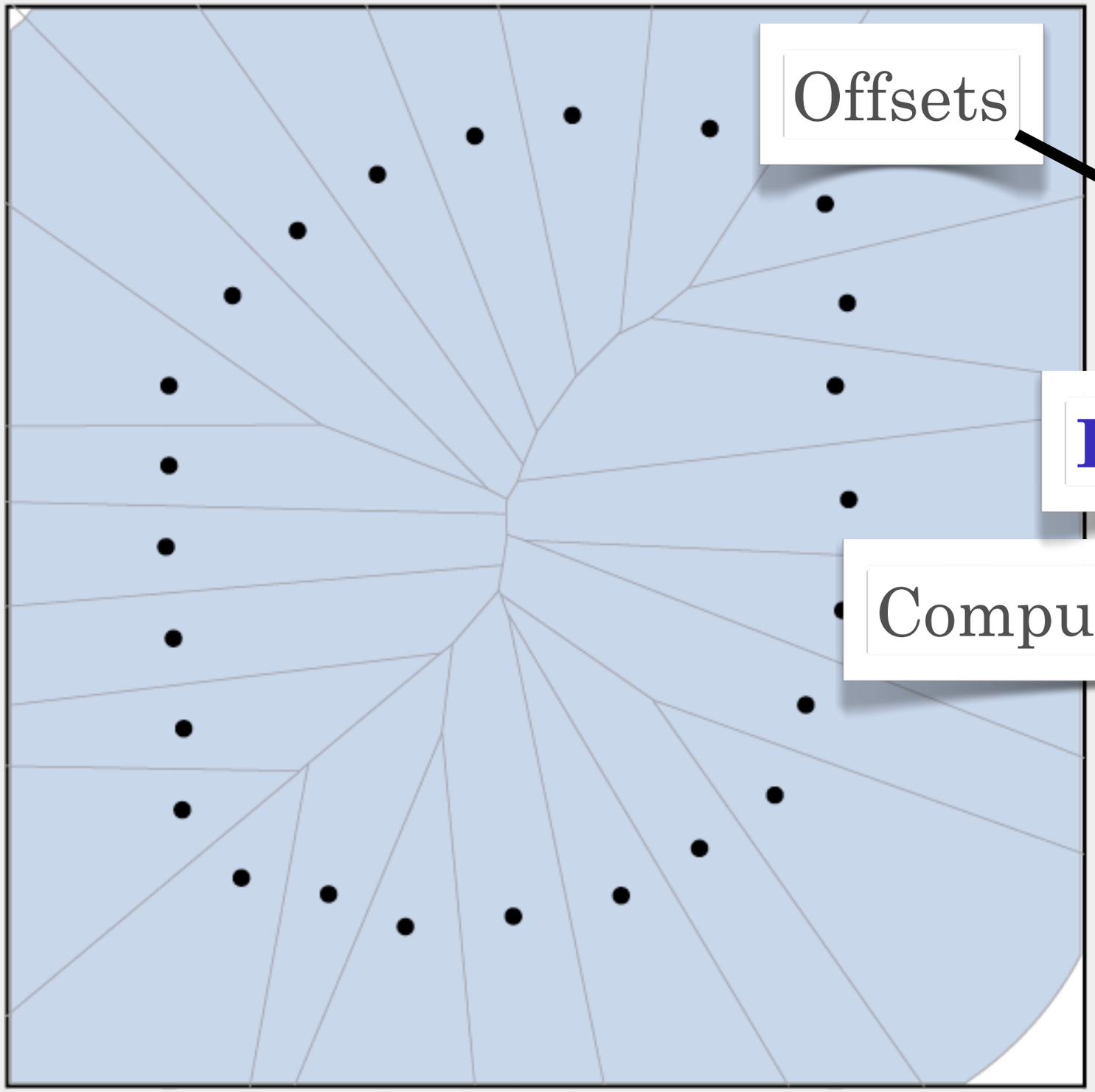


Input: $P \subset \mathbb{R}^d$

$$P^\alpha = \bigcup_{p \in P} \text{ball}(p, \alpha)$$

Compute the **Homology**

Points, offsets, homology, and persistence.



Offsets

Input: $P \subset \mathbb{R}^d$

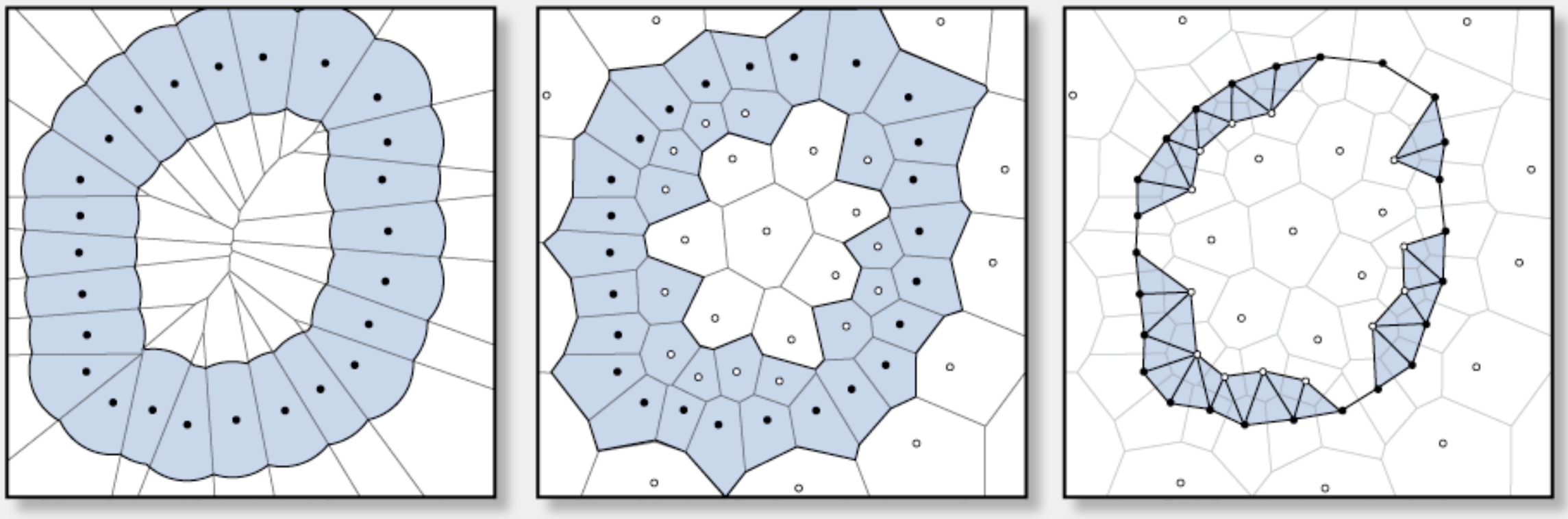
$$P^\alpha = \bigcup_{p \in P} \text{ball}(p, \alpha)$$

Persistent

Compute the **Homology**

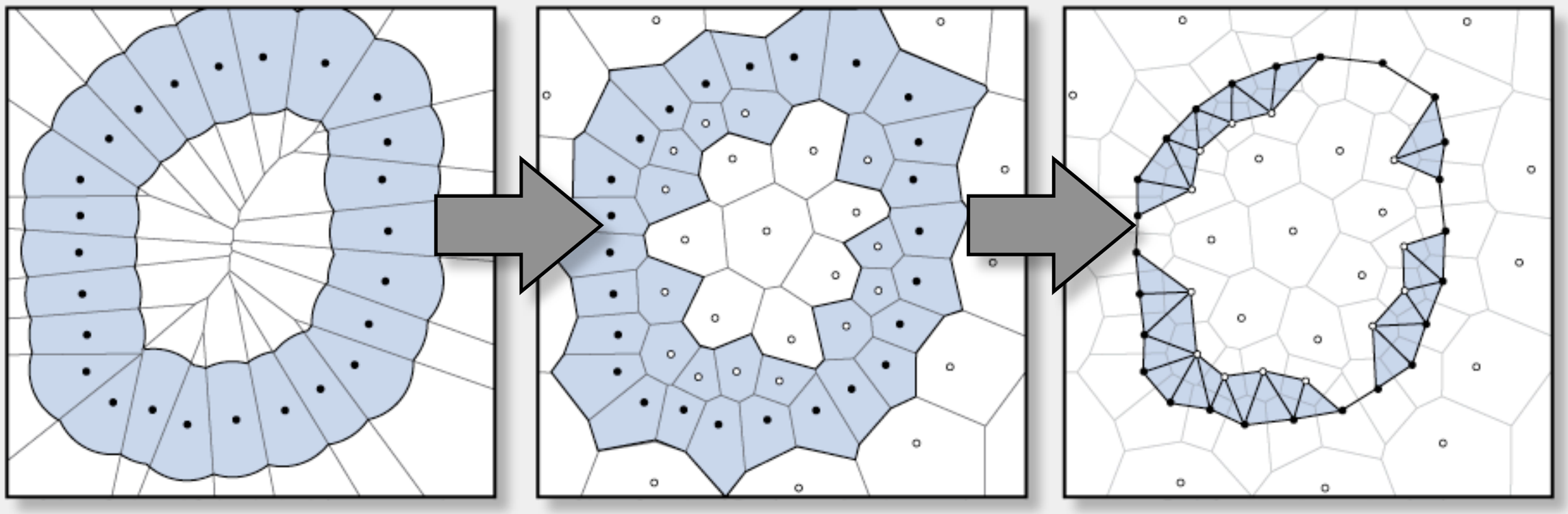
Geometric Approximation

Topologically Equivalent



Geometric Approximation

Topologically Equivalent



Complexity: How big is the mesh?

Complexity: How big is the mesh?

How many Steiner Points?

Complexity: How big is the mesh?

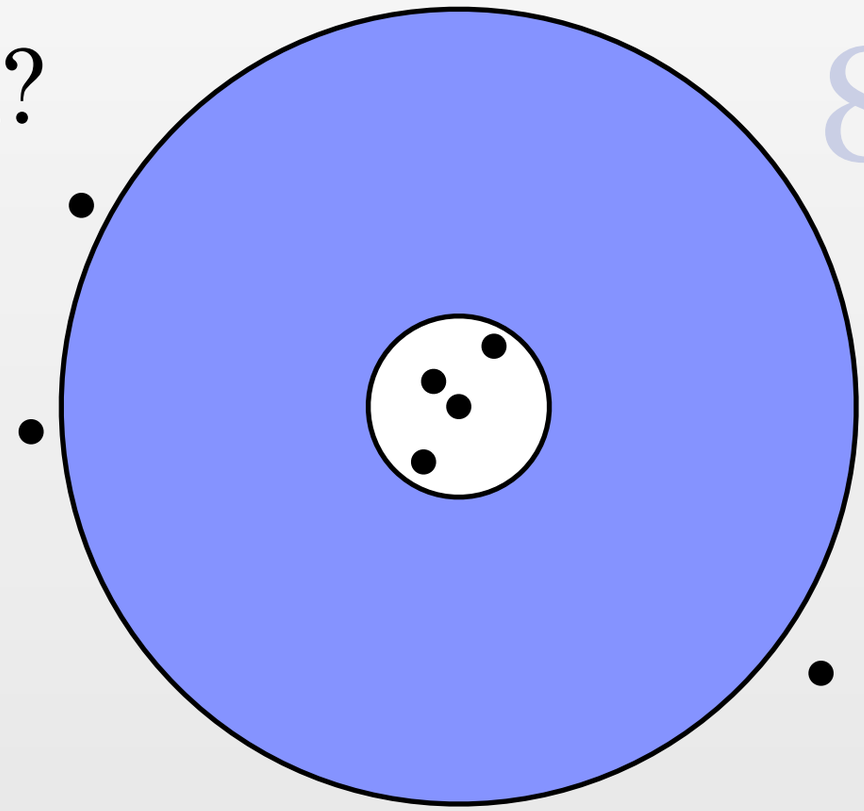
How many Steiner Points?

$$|M| = \int_{\Omega} \frac{1}{\mathrm{lfs}_P(x)^d} dx$$

Complexity: How big is the mesh?

How many Steiner Points?

$$|M| = \int_{\Omega} \frac{1}{\mathrm{lfs}_P(x)^d} dx$$

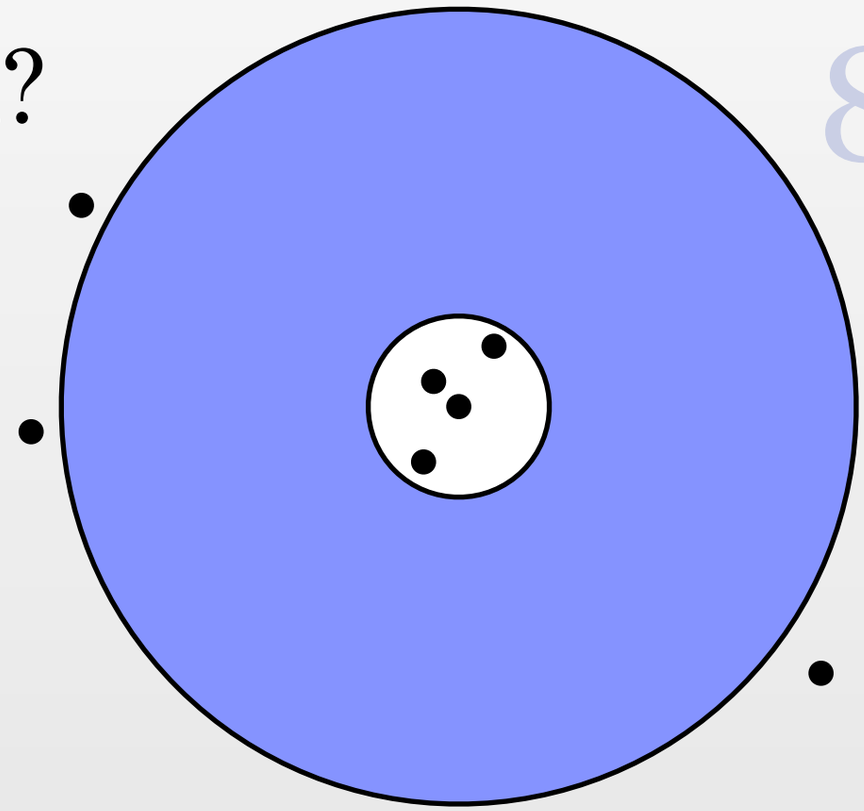


$|M| = O(n)$ as long as there are no big empty annuli with 2 or more points inside [MPS08].

Complexity: How big is the mesh?

How many Steiner Points?

$$|M| = \int_{\Omega} \frac{1}{\mathrm{lfs}_P(x)^d} dx$$



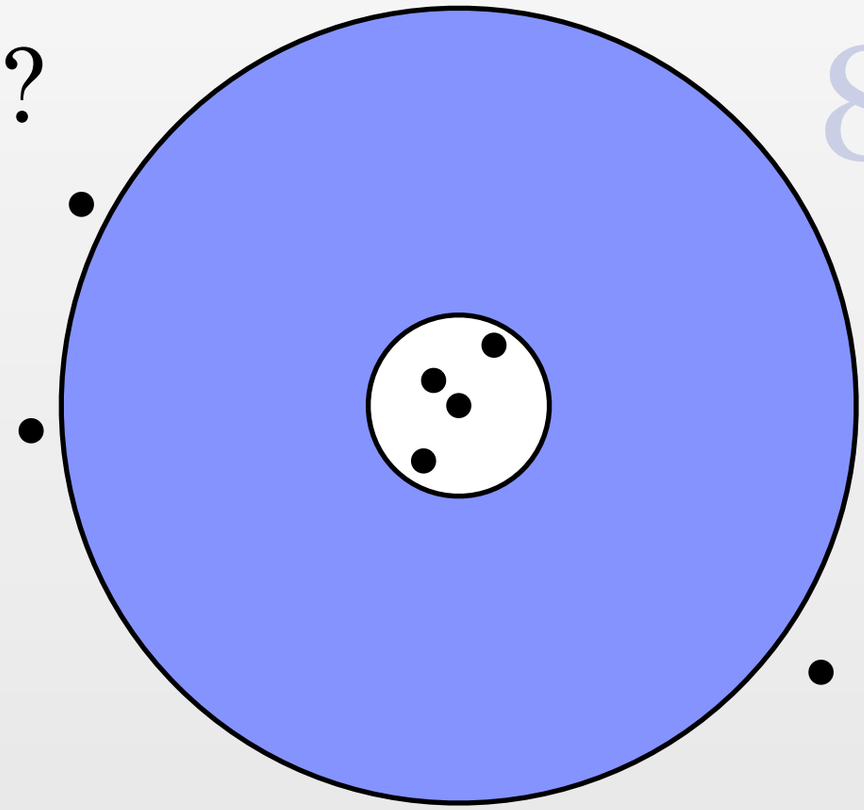
$|M| = O(n)$ as long as there are no big empty annuli with 2 or more points inside [MPS08].

How many simplices?

Complexity: How big is the mesh?

How many Steiner Points?

$$|M| = \int_\Omega \frac{1}{\mathrm{lfs}_P(x)^d} dx$$



$|M| = O(n)$ as long as there are no big empty annuli with 2 or more points inside [MPS08].

How many simplices?

Only $O(|M|)$ simplices.
Compare to $|M|^{\lceil d/2 \rceil}$ for general Delaunay triangulations.
Constants depend on aspect ratio.

Complexity: How hard is it to compute a mesh?

Complexity: How hard is it to compute a mesh?

$$O(n \log n + |M|)$$

Complexity: How hard is it to compute a mesh?

$$O(n \log n + |M|)$$

Point Location

Complexity: How hard is it to compute a mesh?

$$O(n \log n + |M|)$$

Point Location

Output Sensitive

Complexity: How hard is it to compute a mesh?

$$O(n \log n + |M|)$$

Point Location

Output Sensitive

This is optimal in the comparison model.

Complexity: How hard is it to compute a mesh?

$$O(n \log n + |M|)$$

Point Location

Output Sensitive

This is optimal in the comparison model.

$$O(n \log n)$$

Complexity: How hard is it to compute a mesh?

$$O(n \log n + |M|)$$

Point Location

Output Sensitive

This is optimal in the comparison model.

$$O(n \log n)$$

Compute a *hierarchical quality* mesh.

Complexity: How hard is it to compute a mesh?

$$O(n \log n + |M|)$$

Point Location

Output Sensitive

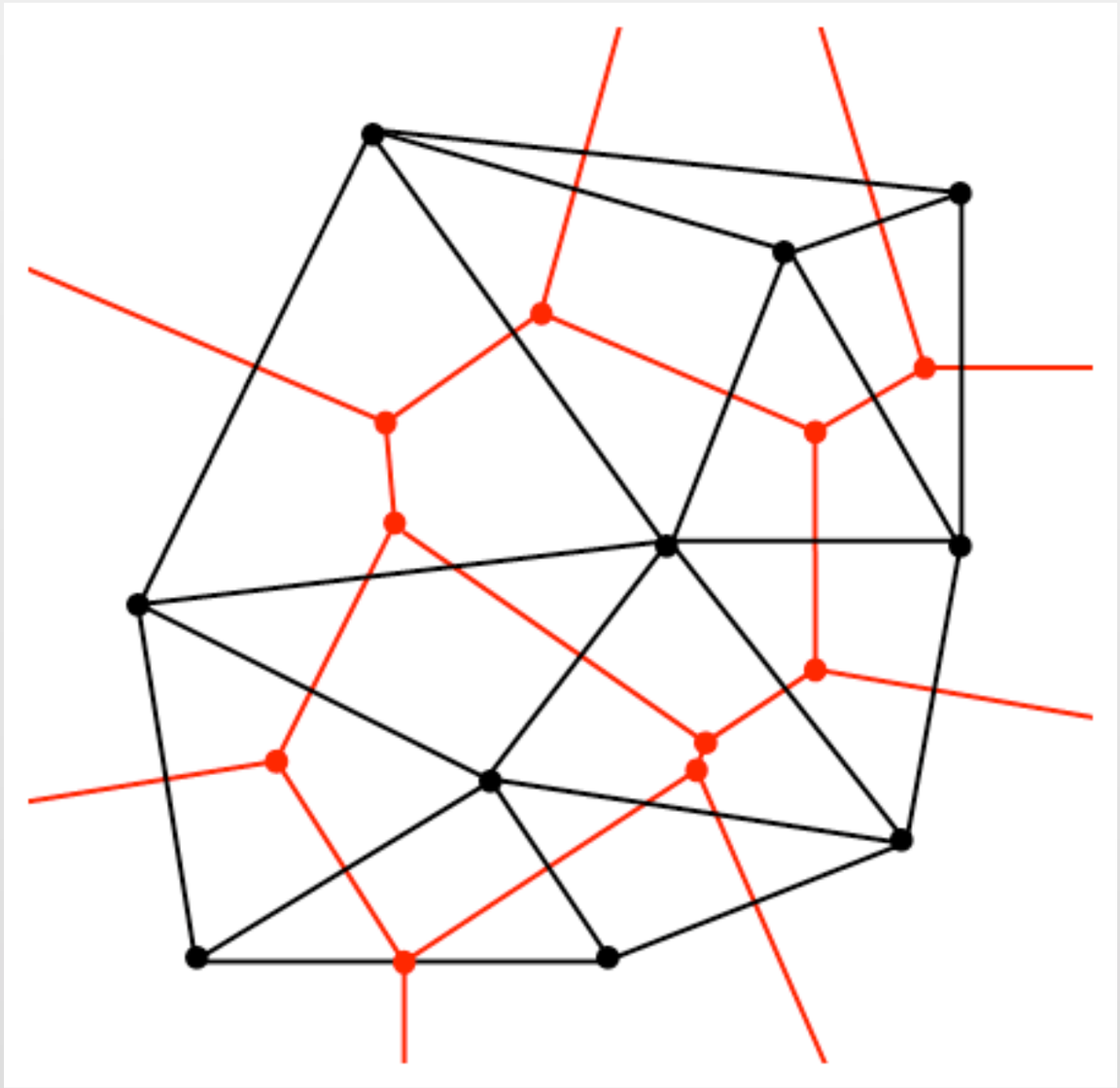
This is optimal in the comparison model.

$$O(n \log n) \quad + \quad O(|M|)$$

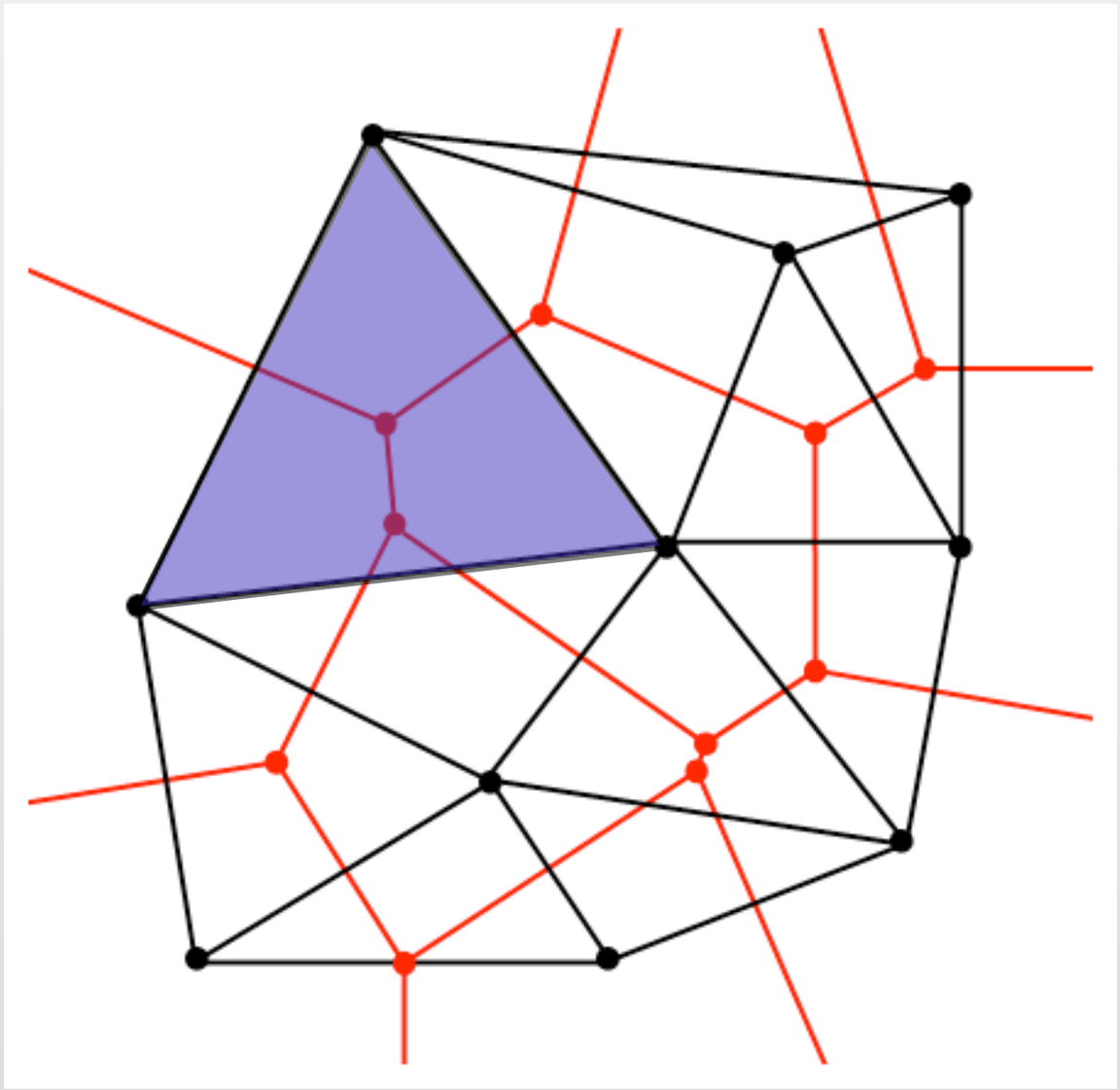
Compute a *hierarchical quality* mesh.

Finishing post-process.

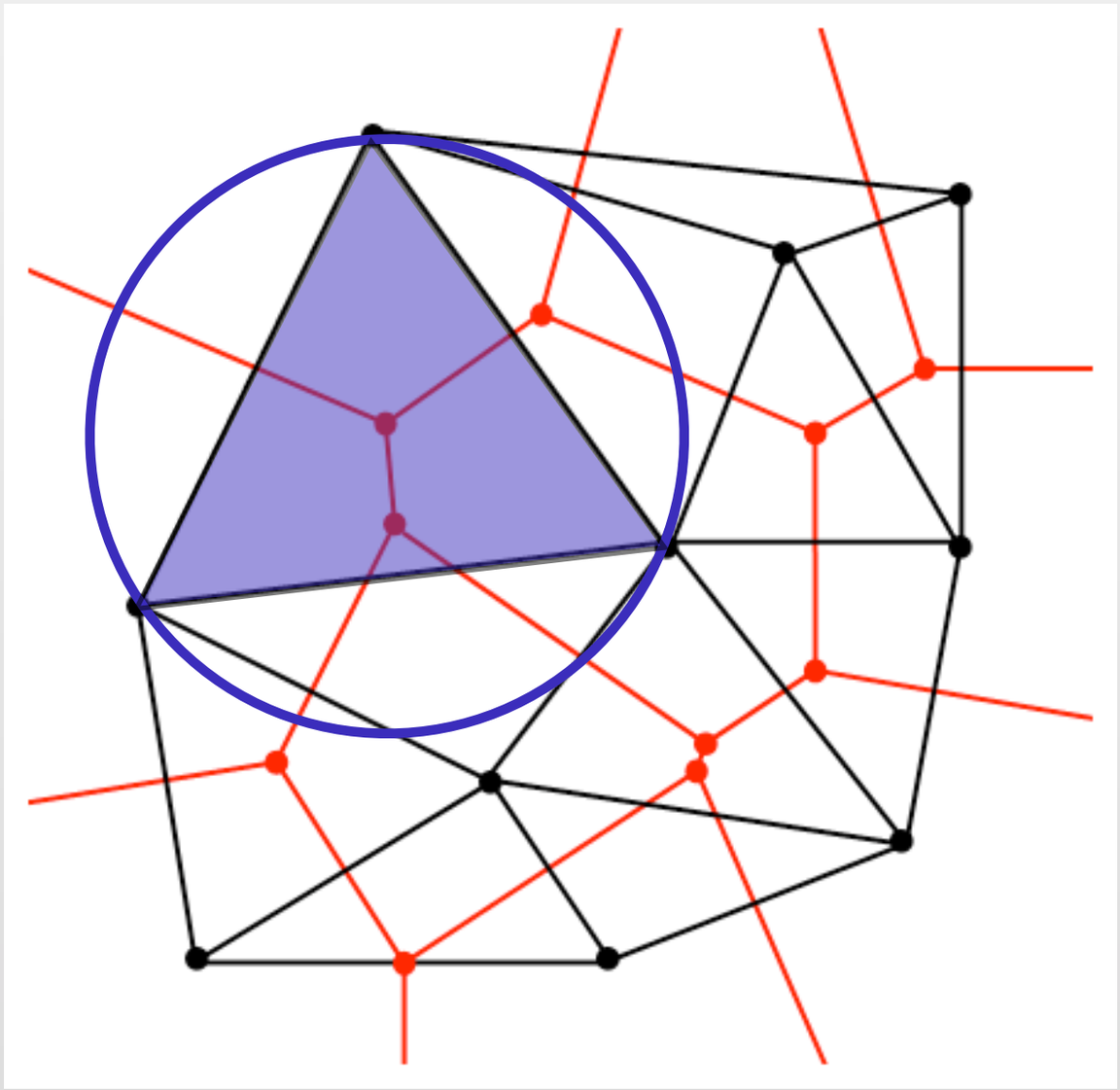
The Delaunay Triangulation is the dual of the Voronoi Diagram.



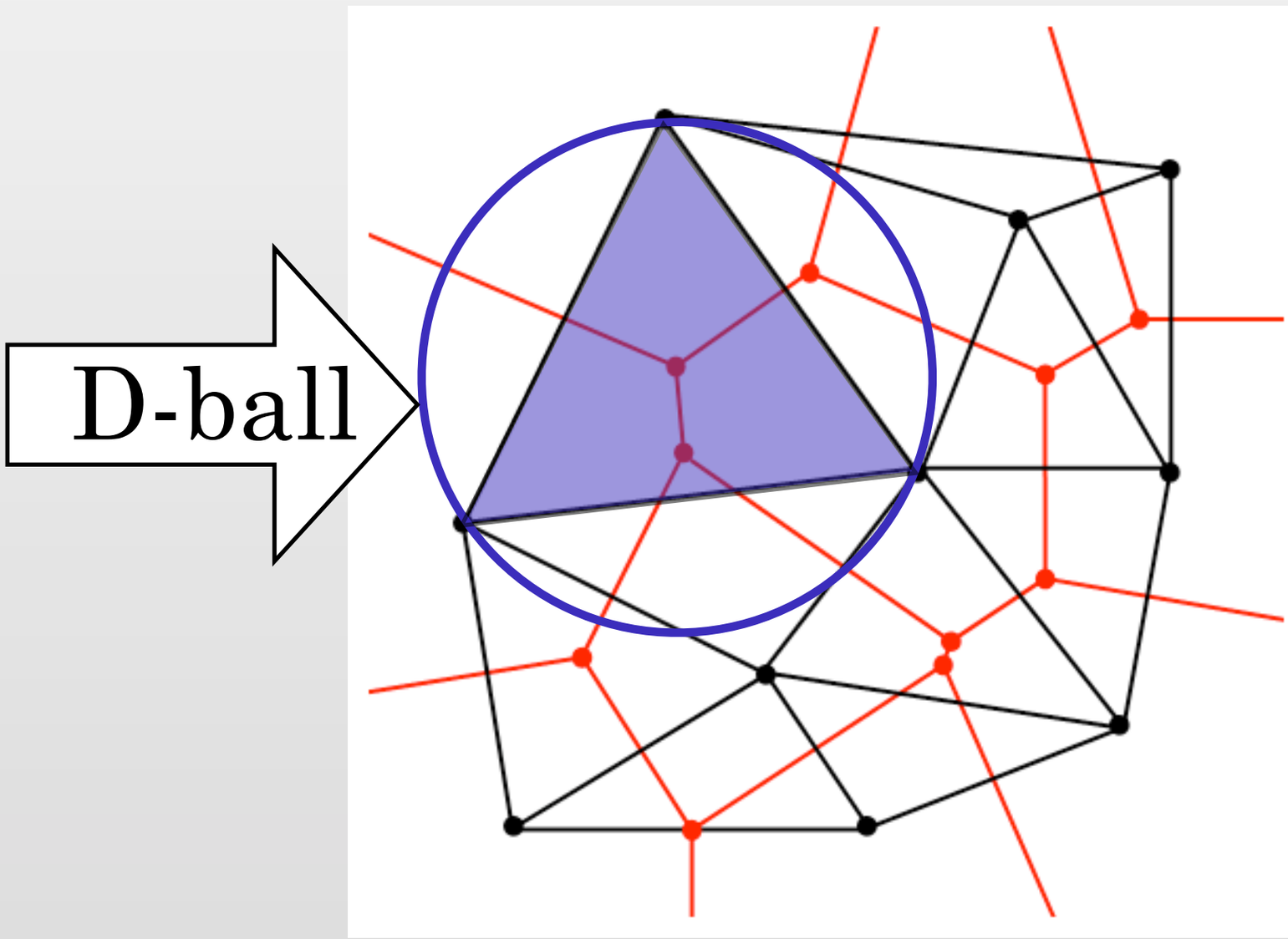
The Delaunay Triangulation is the dual of the Voronoi Diagram.



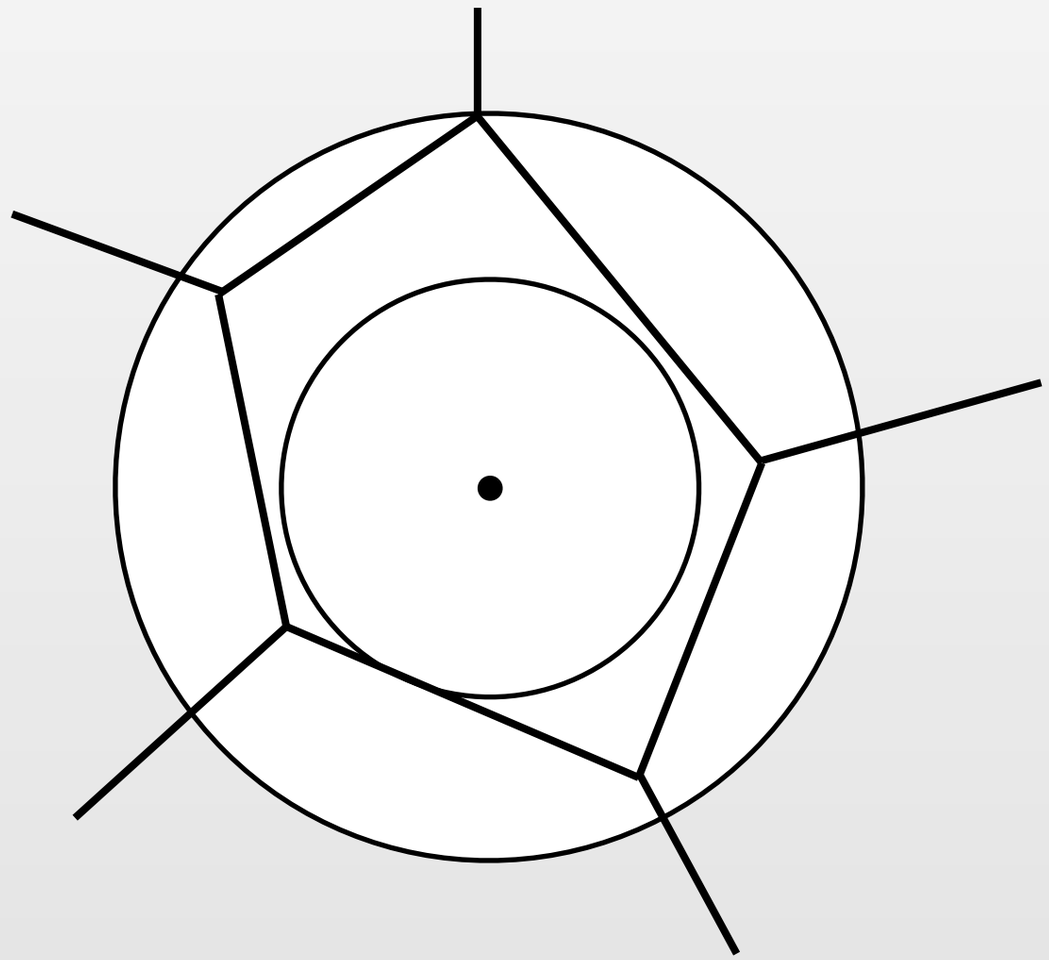
The Delaunay Triangulation is the dual of the Voronoi Diagram.



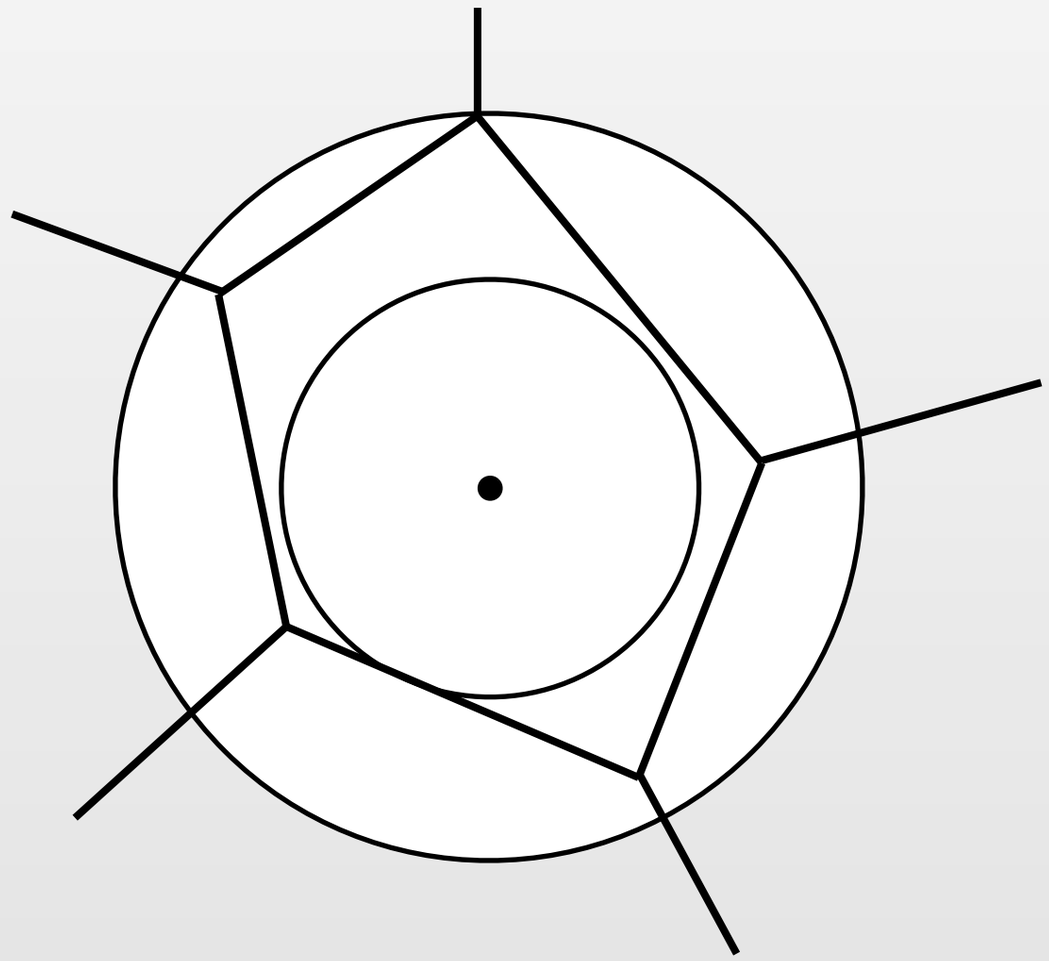
The Delaunay Triangulation is the dual of the Voronoi Diagram.



Quality Meshes

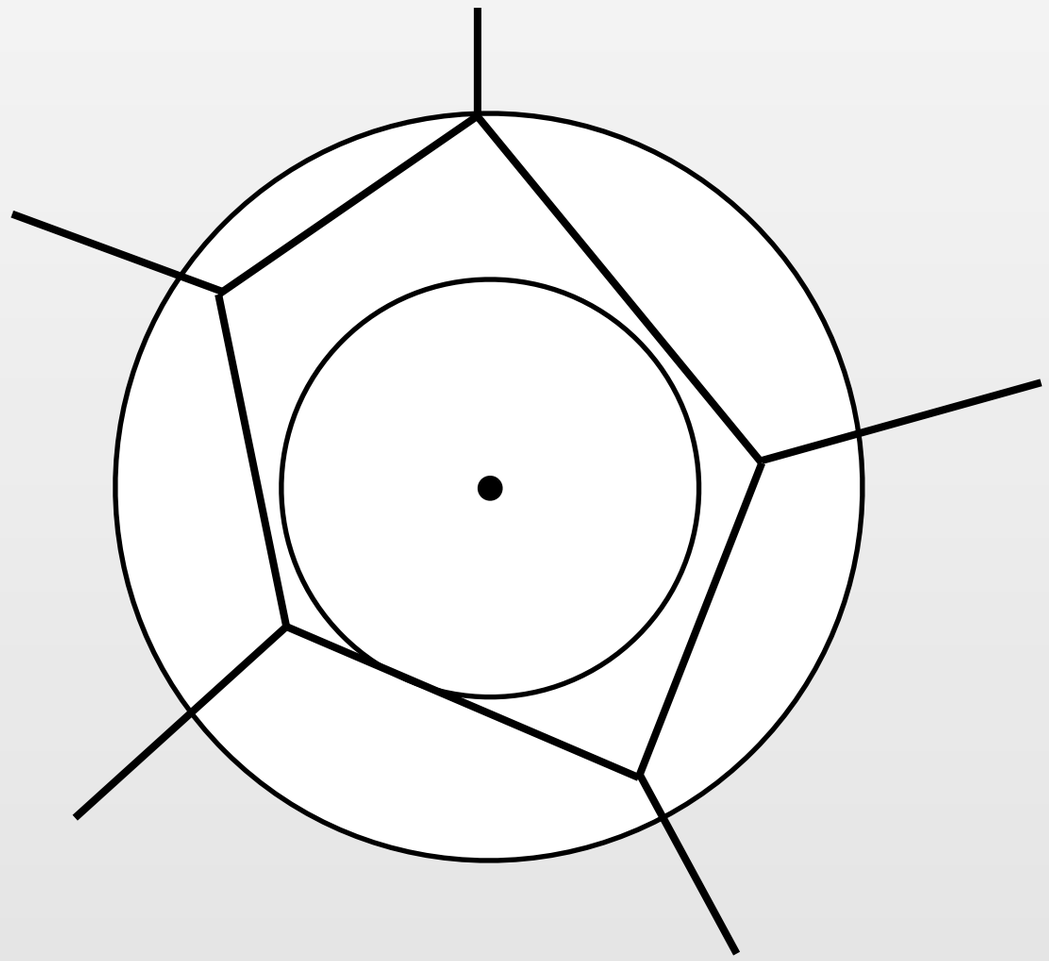


Quality Meshes



D-Balls are Constant Ply

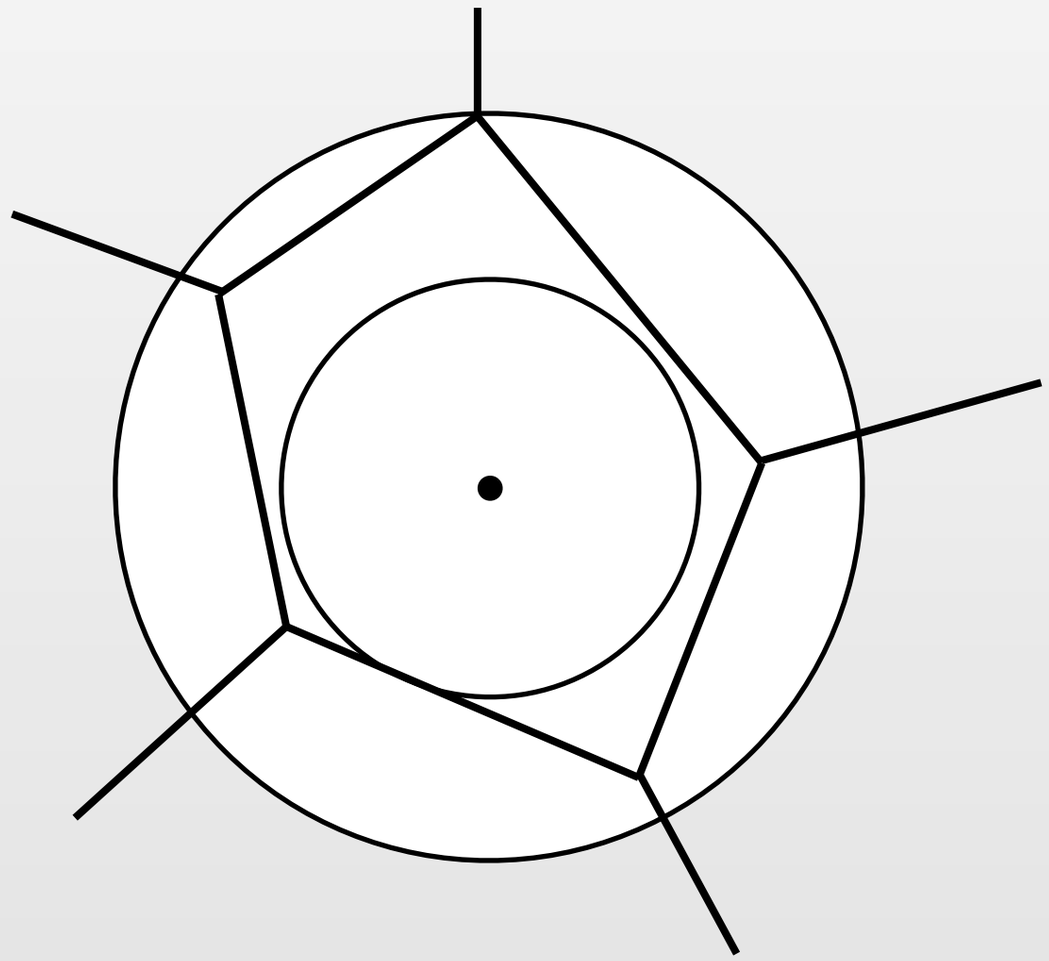
Quality Meshes



D-Balls are Constant Ply

Total Complexity is linear in |M|

Quality Meshes

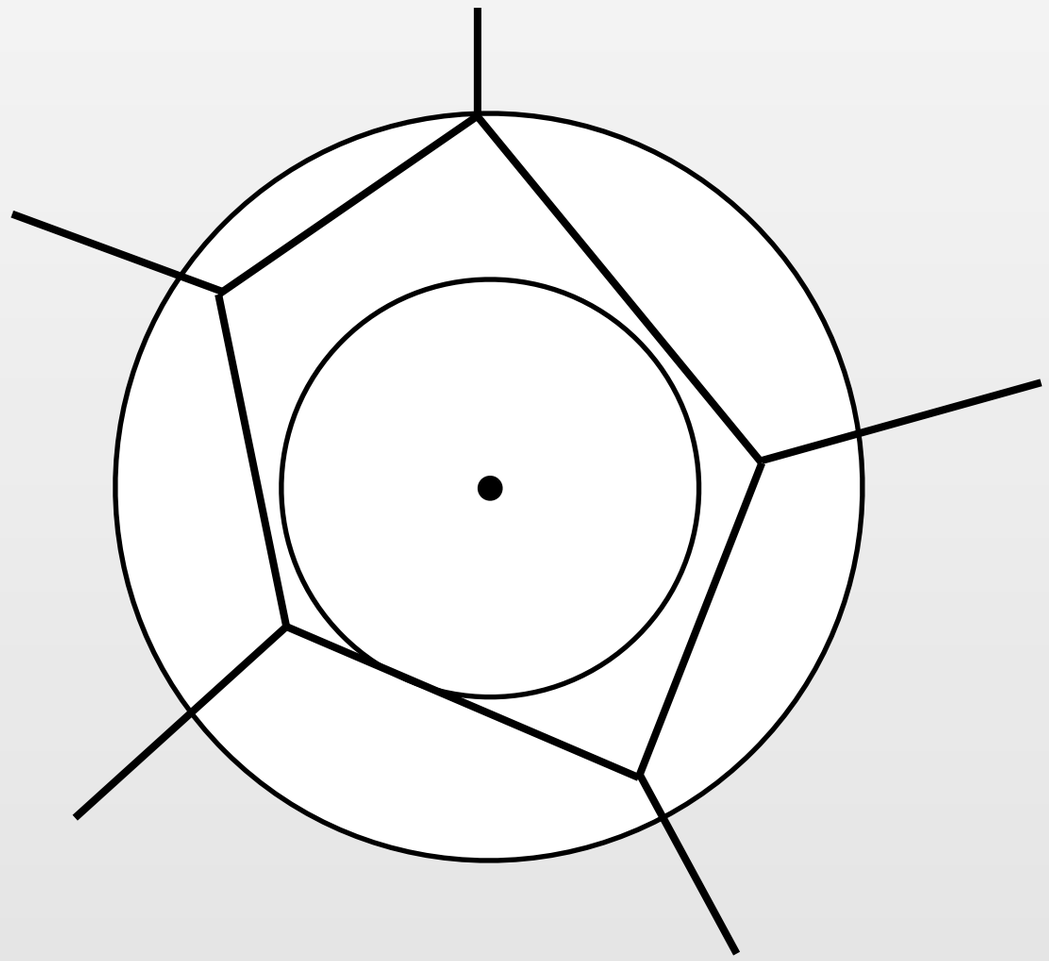


D-Balls are Constant Ply

Total Complexity is linear in |M|

No D-ball intersects more than O(1) others.

Quality Meshes



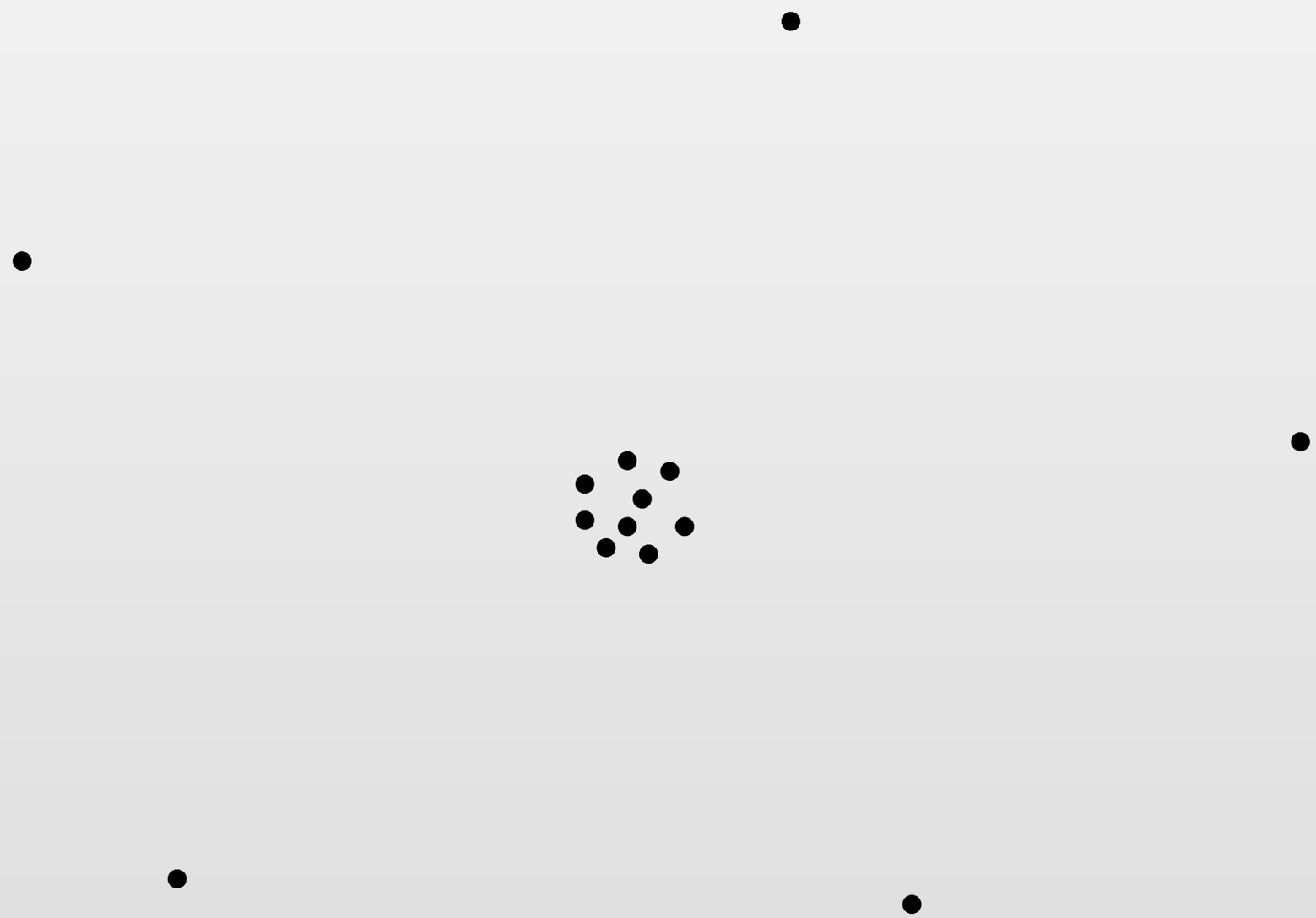
D-Balls are Constant Ply

Total Complexity is linear in |M|

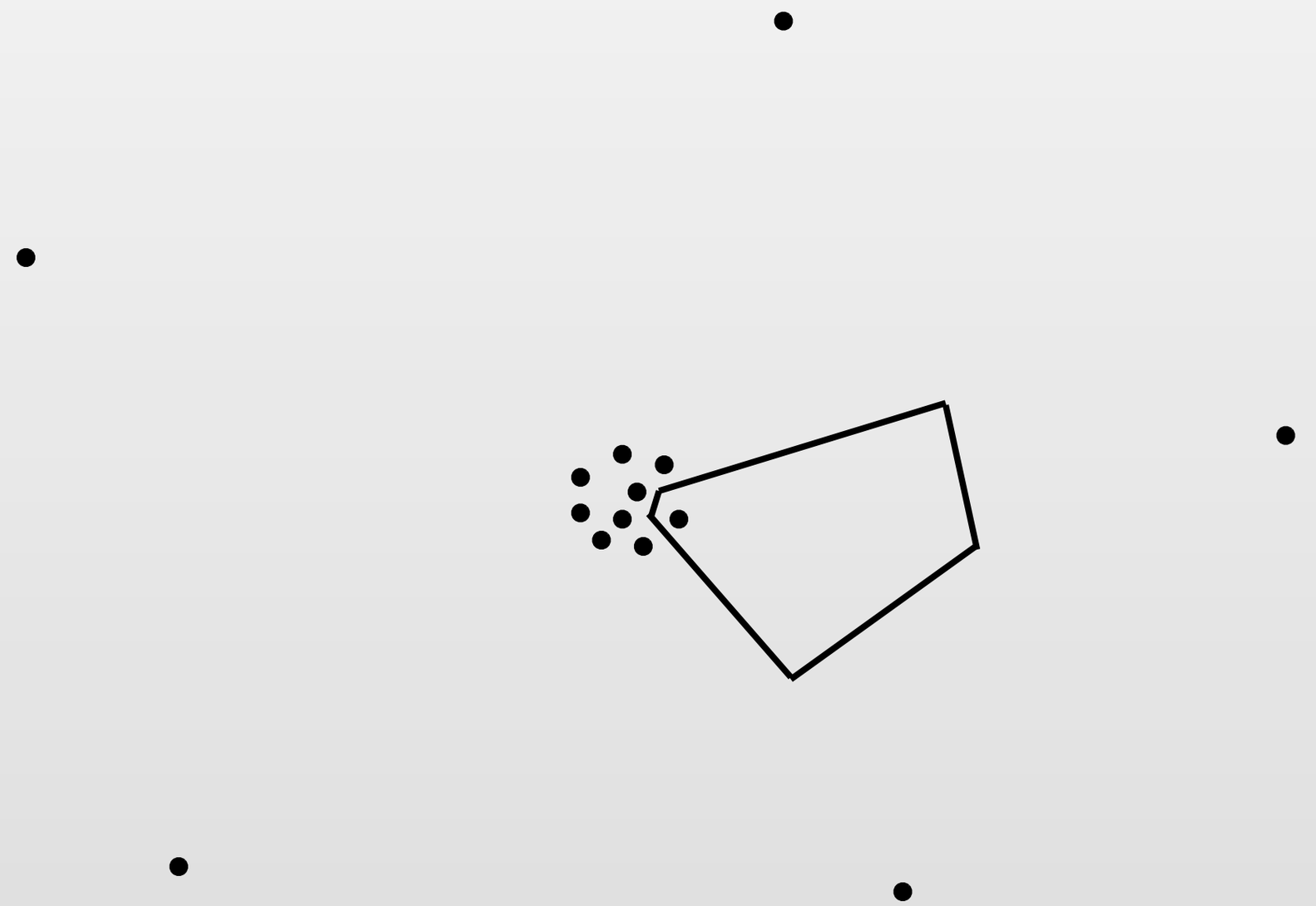
No D-ball intersects more than O(1) others.

Refine poor-quality cells by adding a Steiner point at its farthest vertex.

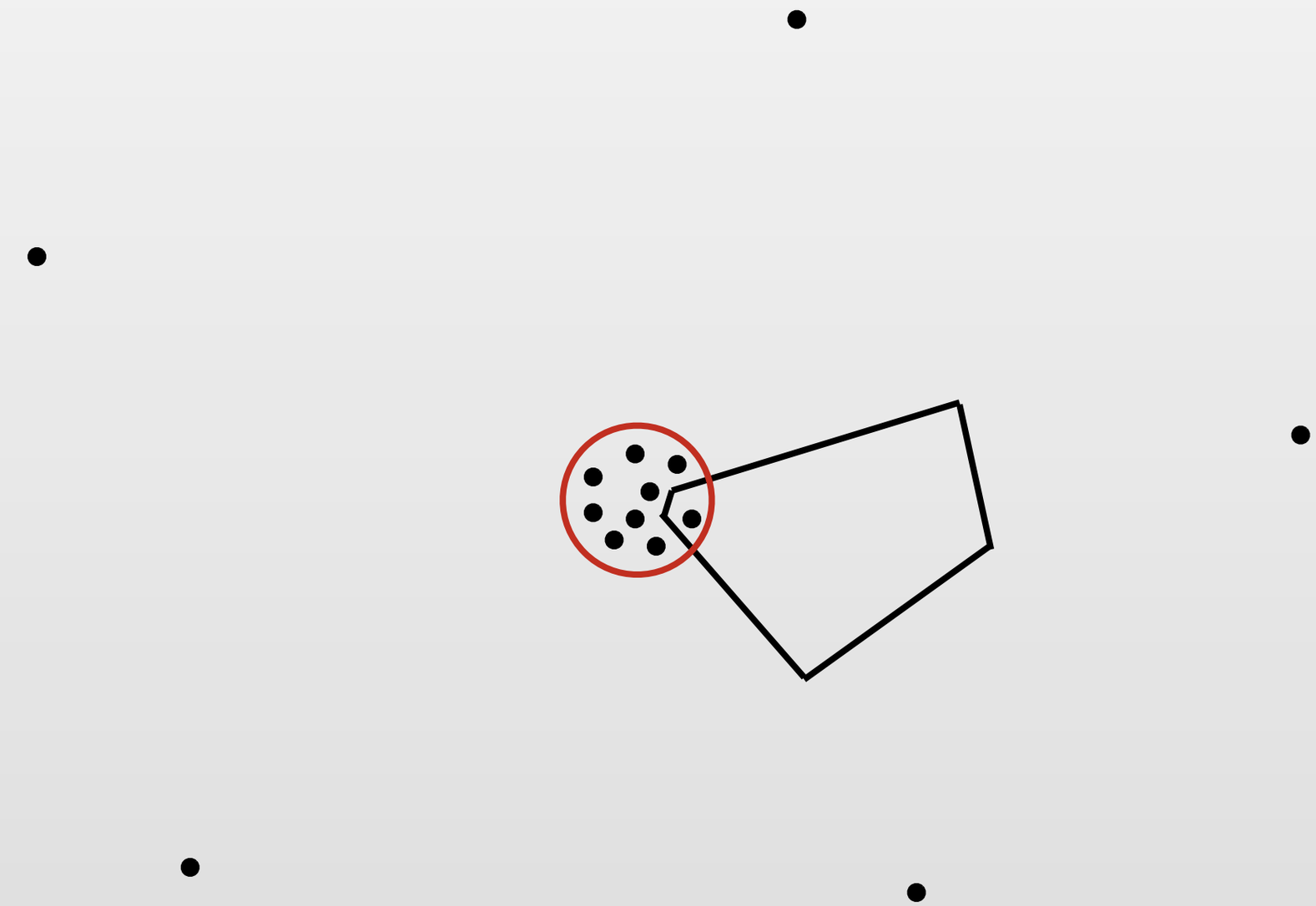
We replace *quality* with *hierarchical quality*.



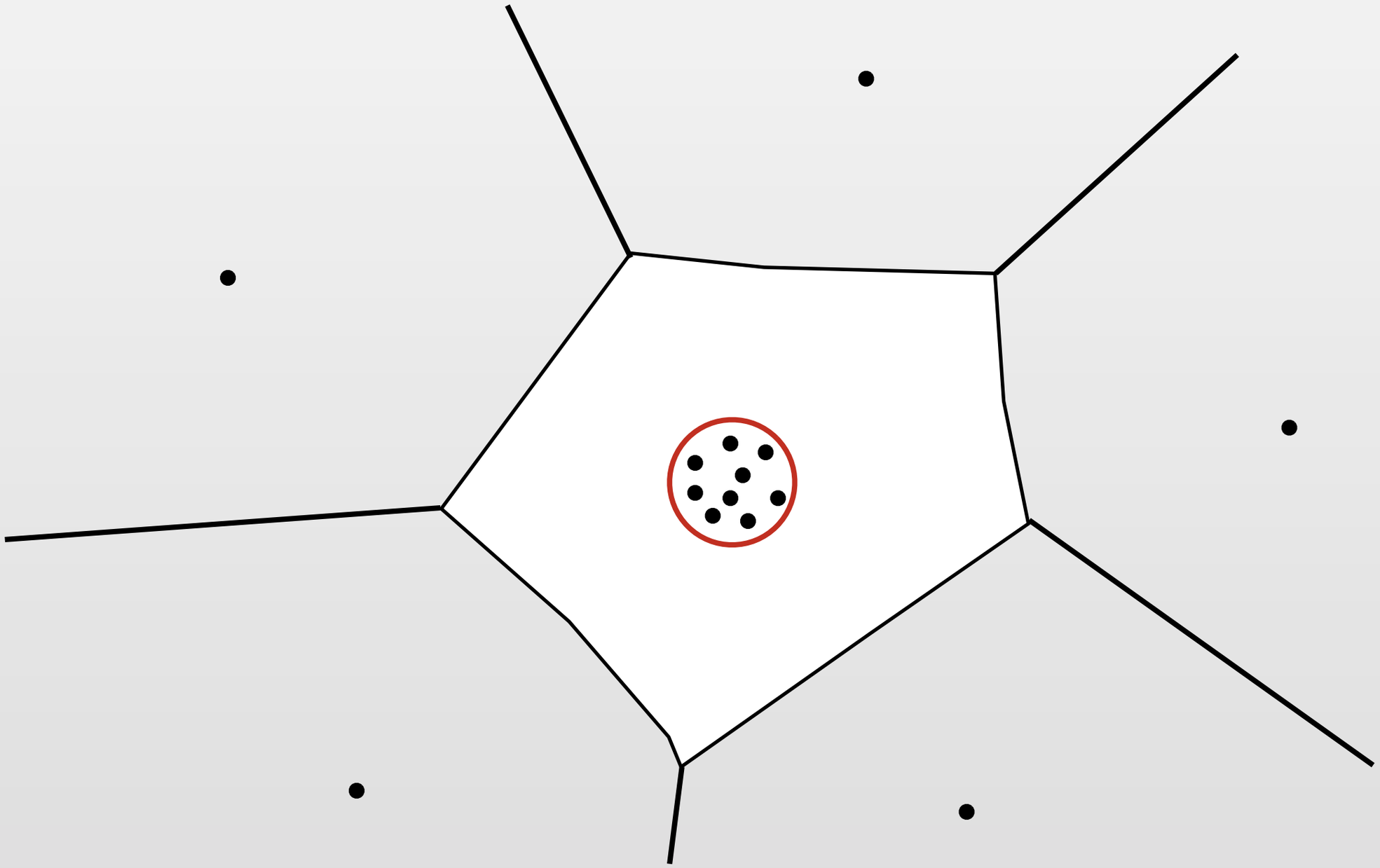
We replace *quality* with *hierarchical quality*.



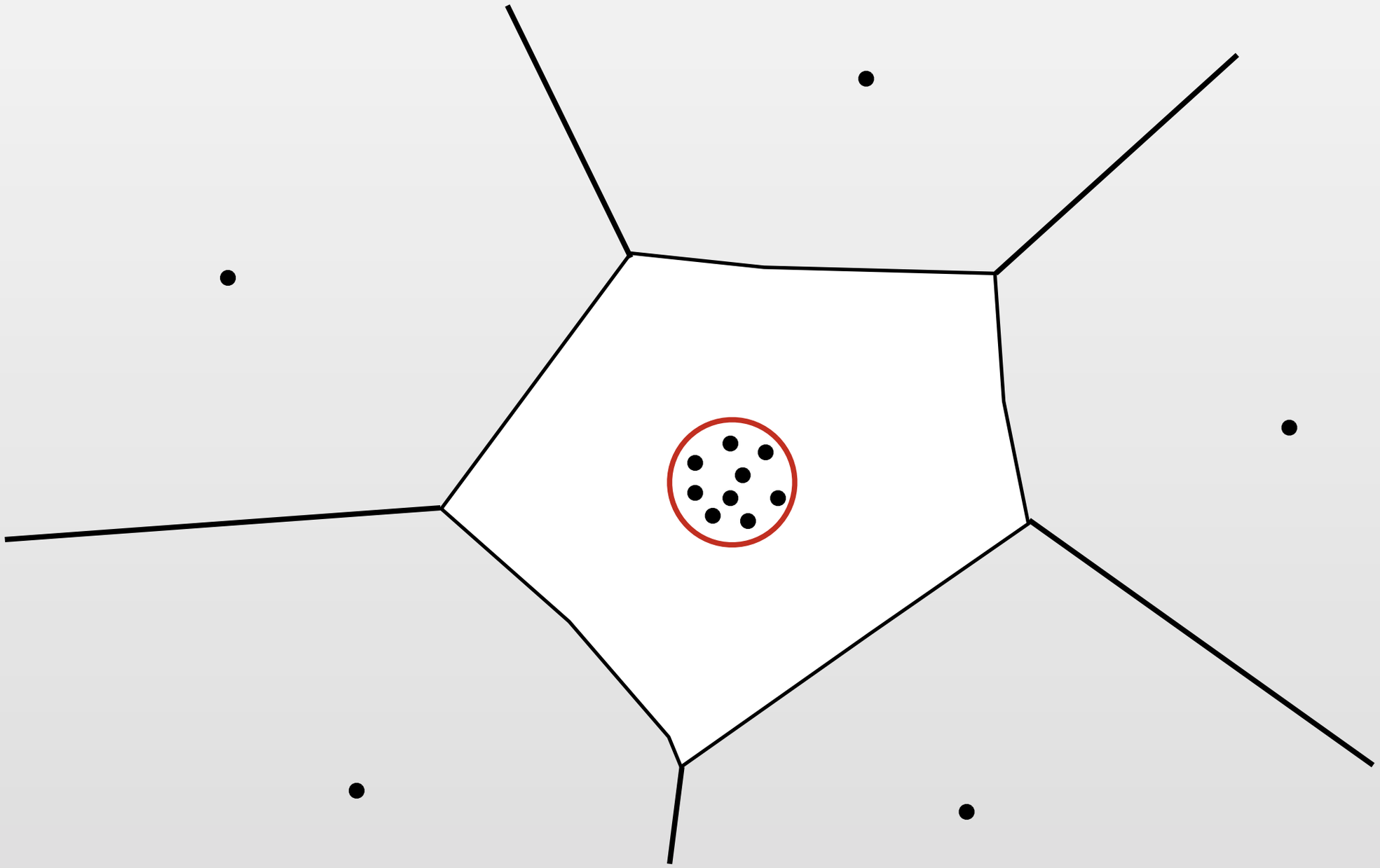
We replace *quality* with *hierarchical quality*.



We replace *quality* with *hierarchical quality*.

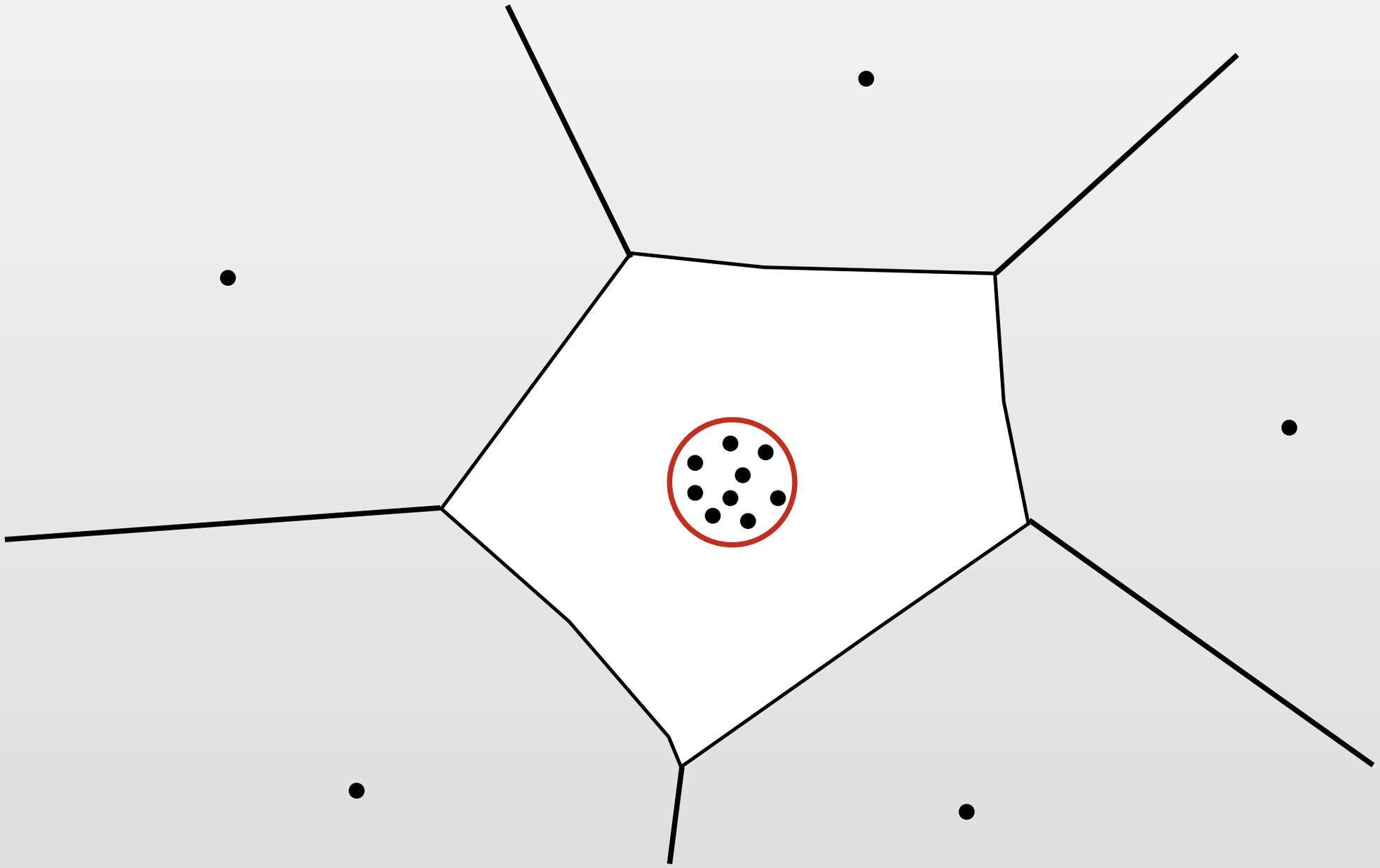


We replace *quality* with *hierarchical quality*.



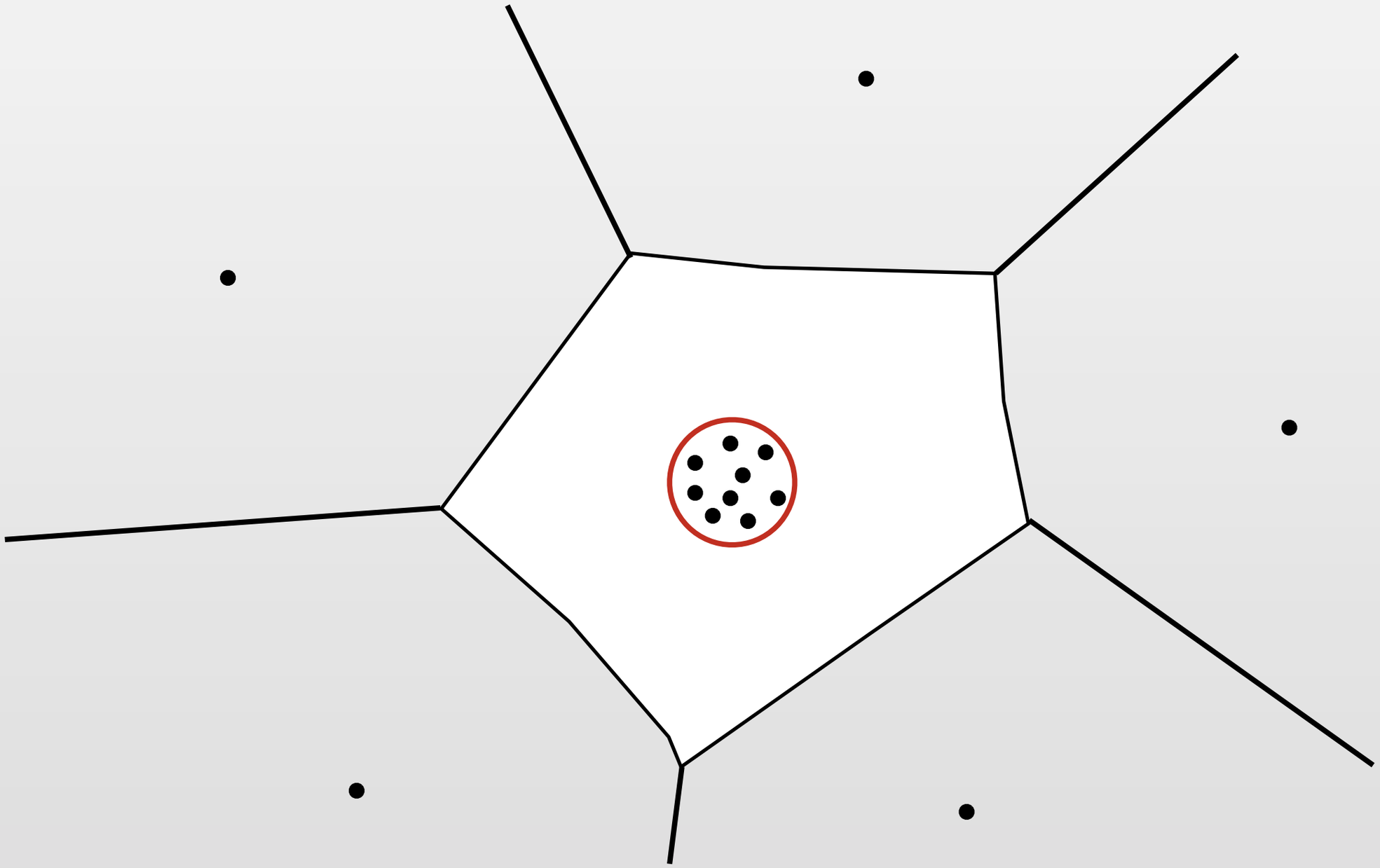
Inside the cage: Old definition of quality.

We replace *quality* with *hierarchical quality*.



Inside the cage: Old definition of quality.
Outside: Treat the whole cage as a single object.

We replace *quality* with *hierarchical quality*.



Inside the cage: Old definition of quality.
Outside: Treat the whole cage as a single object.

All the same properties as quality meshes: ply, degree, etc.

The sparse meshing model:

The sparse meshing model:

1. Build a Voronoi diagram incrementally.

The sparse meshing model:

1. Build a Voronoi diagram incrementally.
2. Interleave input and Steiner point insertions.

The sparse meshing model:

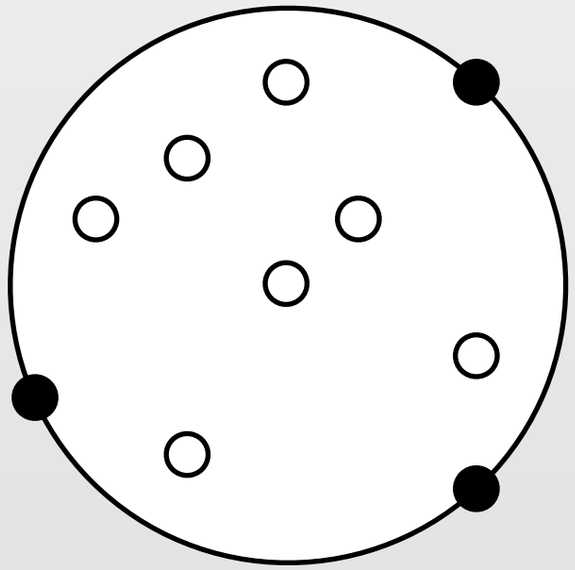
1. Build a Voronoi diagram incrementally.
2. Interleave input and Steiner point insertions.
3. Recover quality after each input point.

The sparse meshing model:

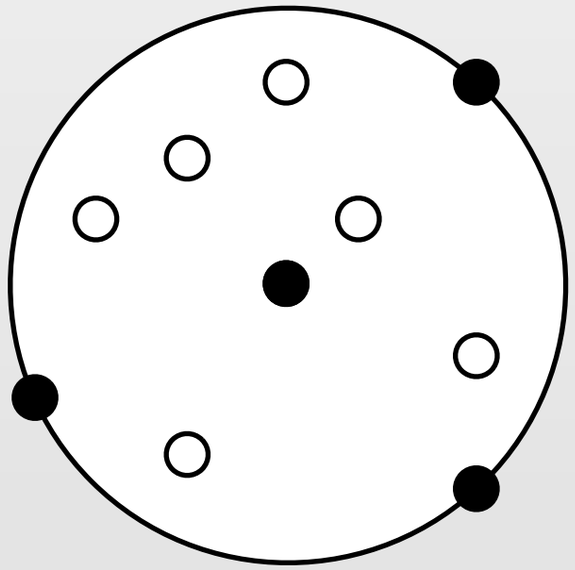
1. Build a Voronoi diagram incrementally.
2. Interleave input and Steiner point insertions.
3. Recover quality after each input point.

This is how we avoid the worst-case Voronoi bounds.

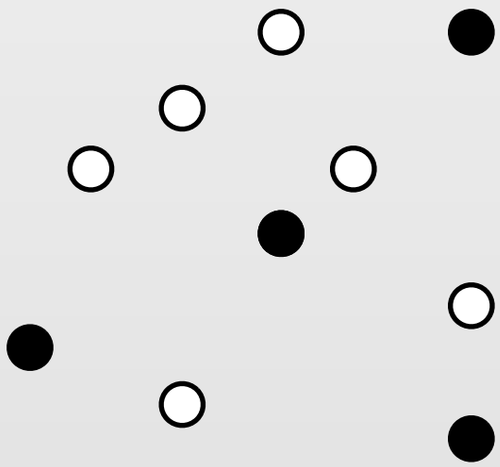
Idea: Store the uninserted points in the D-balls.
When the balls change, make local updates.



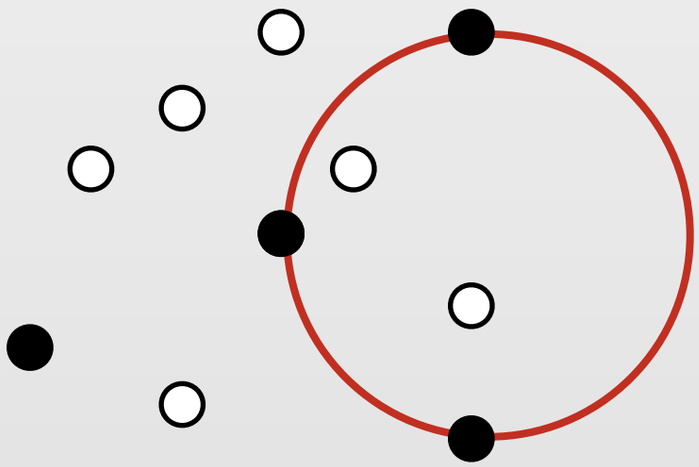
Idea: Store the uninserted points in the D-balls.
When the balls change, make local updates.



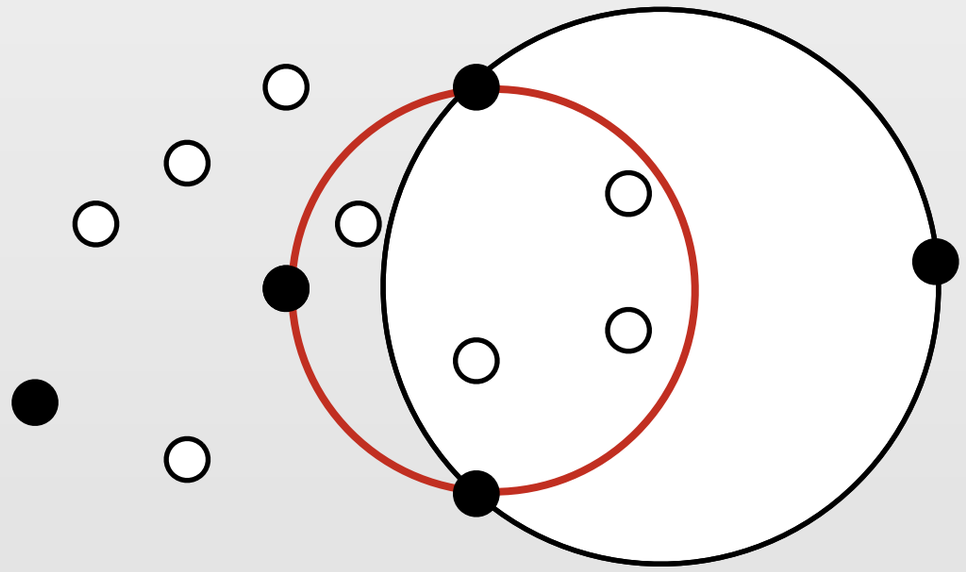
Idea: Store the uninserted points in the D-balls.
When the balls change, make local updates.



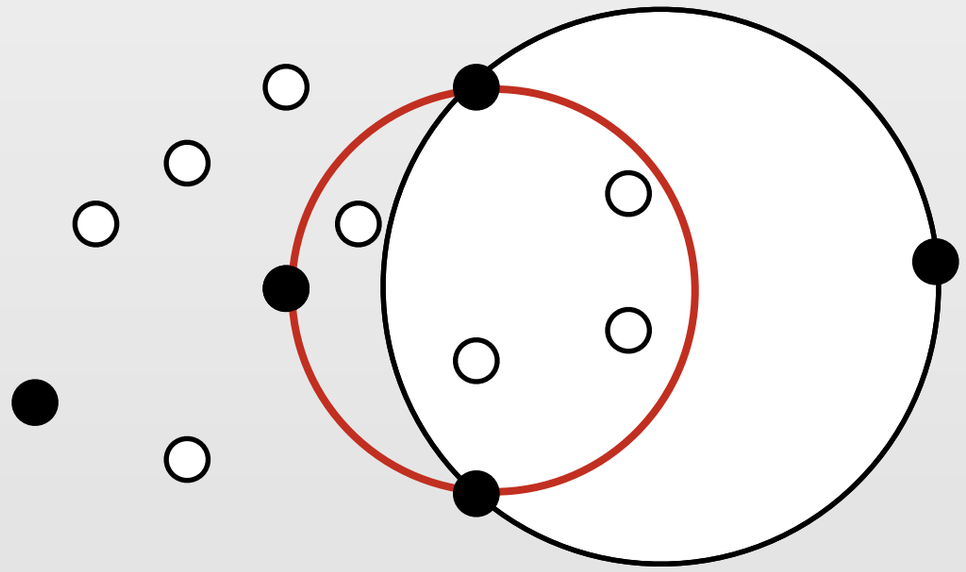
Idea: Store the uninserted points in the D-balls.
When the balls change, make local updates.



Idea: Store the uninserted points in the D-balls.
When the balls change, make local updates.

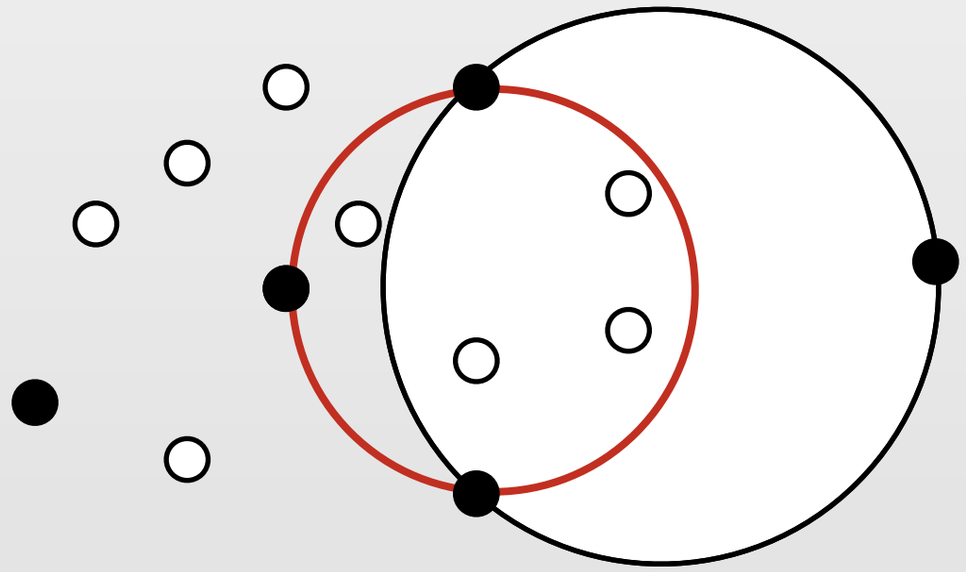


Idea: Store the uninserted points in the D-balls.
When the balls change, make local updates.



Lemma (HMP06)**.** *A point is touched at most a constant number of times before the radius of the largest D-ball containing it goes down by a factor of* 2.

Idea: Store the uninserted points in the D-balls.
When the balls change, make local updates.



Lemma (HMP06)**.** *A point is touched at most a constant number of times before the radius of the largest D-ball containing it goes down by a factor of* 2*.*

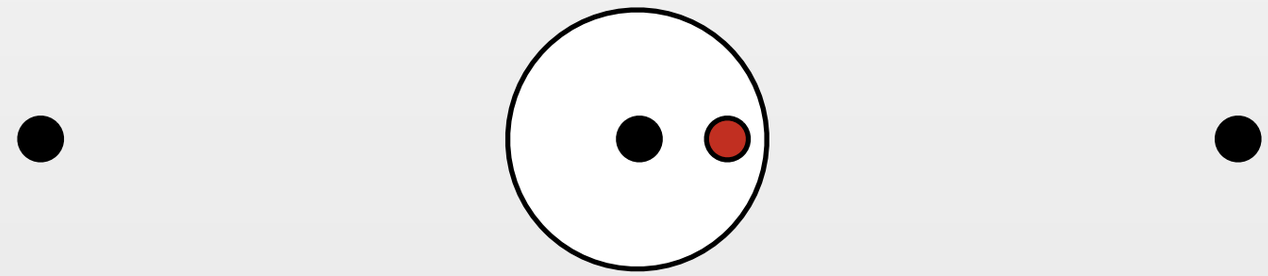
Geometric Divide and Conquer: $O(n \log \Delta)$

With hierarchical meshes, we can add inputs in any order!

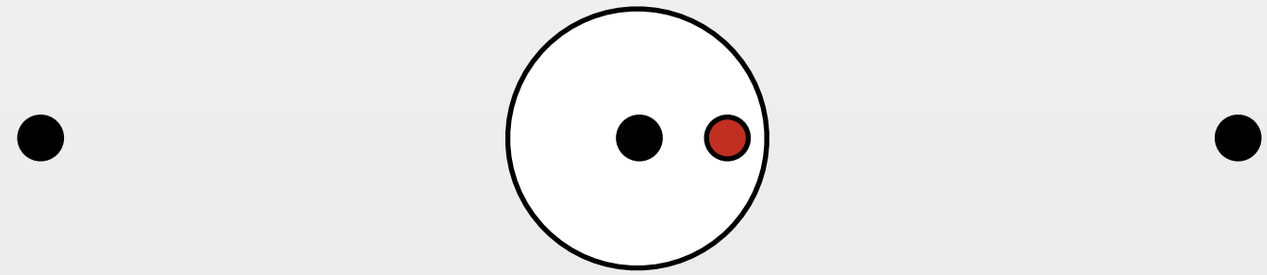
• • •

With hierarchical meshes, we can add inputs in any order!

With hierarchical meshes, we can add inputs in any order!

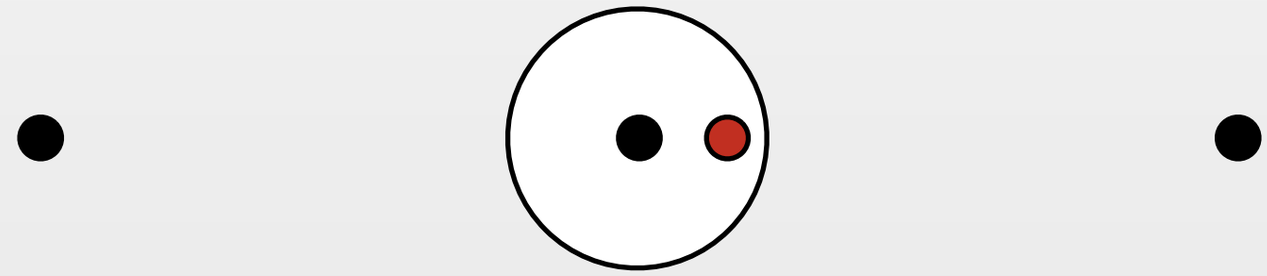


With hierarchical meshes, we can add inputs in any order!



Goal: Combinatorial Divide and Conquer

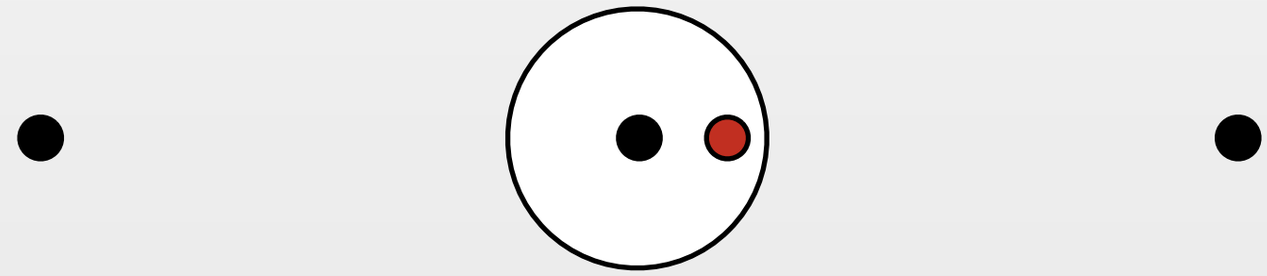
With hierarchical meshes, we can add inputs in any order!



Goal: Combinatorial Divide and Conquer

Old: Progress was decreasing the radius by a factor of 2.

With hierarchical meshes, we can add inputs in any order!



Goal: Combinatorial Divide and Conquer

Old: Progress was decreasing the radius by a factor of 2.

New: Decrease number of points in a ball by a factor of 2.

Range Nets

“Nets catch everything that’s big.”

Range Nets

"Nets catch everything that's big."

Definition. *A* ***range space*** *is a pair* (X, R)*, where* X *is a set (the vertices) and* R *is a collection of subsets (the ranges).*

Range Nets

"Nets catch everything that's big."

Definition. *A* ***range space*** *is a pair* (X, R)*, where* X *is a set (the vertices) and* R *is a collection of subsets (the ranges).*

For us $X = P$ and R is the set of open balls.

Range Nets

"Nets catch everything that's big."

Definition. *A* ***range space*** *is a pair* (X, R)*, where* X *is a set (the vertices) and* R *is a collection of subsets (the ranges).*

For us $X = P$ and R is the set of open balls.

Definition. *Given a range space* (X, R)*, a set* $N \subset X$ *is a* ***range space*** ε***-net*** *if for all ranges* $r \in R$ *that contain at least* $\varepsilon|X|$ *vertices,* r *contains a vertex from* N*.*

Range Nets

"Nets catch everything that's big."

Definition. *A* ***range space*** *is a pair* (X, R), *where* X *is a set (the vertices) and* R *is a collection of subsets (the ranges).*

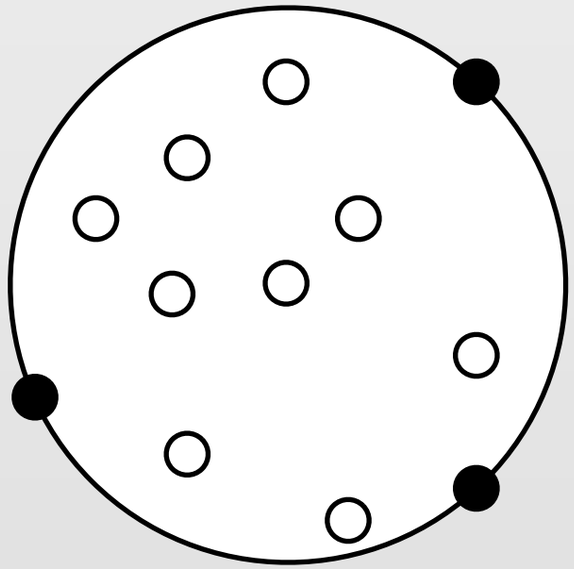
For us $X = P$ and R is the set of open balls.

Definition. *Given a range space* (X, R), *a set* $N \subset X$ *is a* ***range space*** ε***-net*** *if for all ranges* $r \in R$ *that contain at least* $\varepsilon|X|$ *vertices,* r *contains a vertex from* N.

Theorem: [Chazelle & Matousek 96] For ε, d fixed constants, ε-nets of size *O(1)* can be computed in *O(n)* deterministic time.

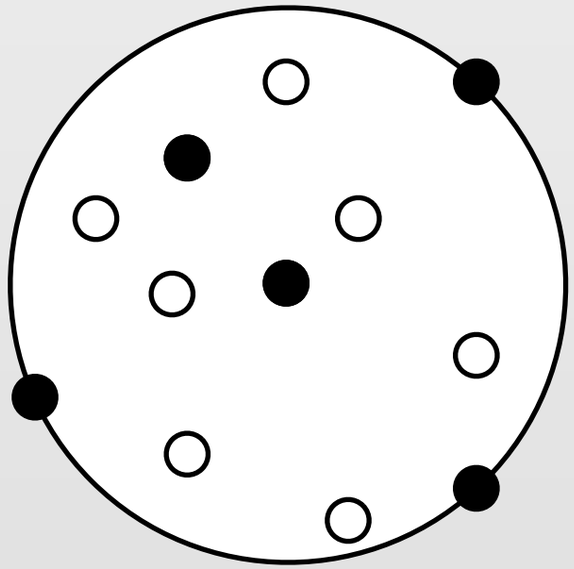
Ordering the inputs

For each D-Ball, select a 1/2d-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



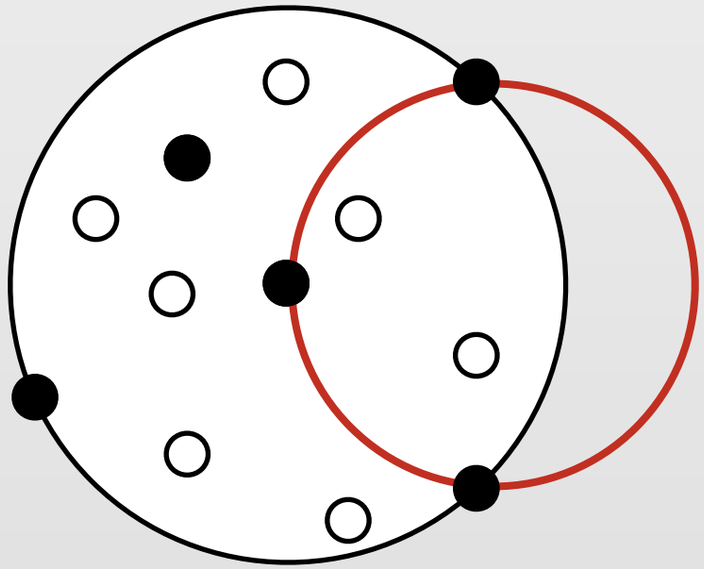
Ordering the inputs

For each D-Ball, select a 1/2d-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



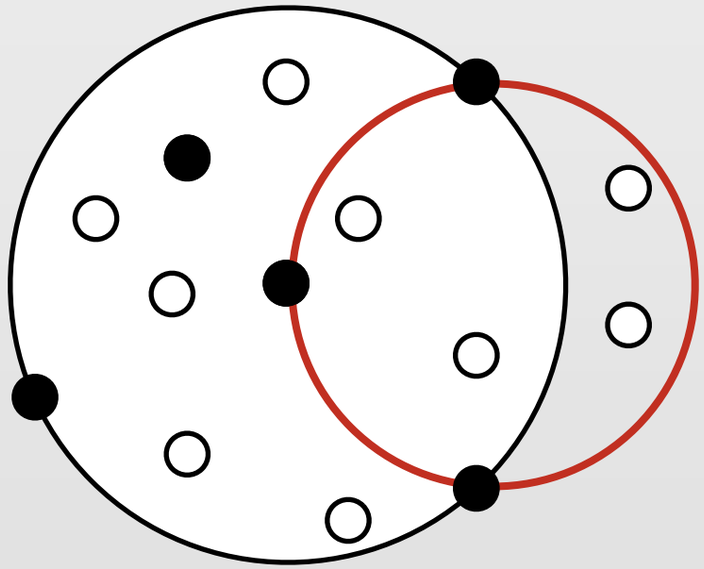
Ordering the inputs

For each D-Ball, select a 1/2d-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



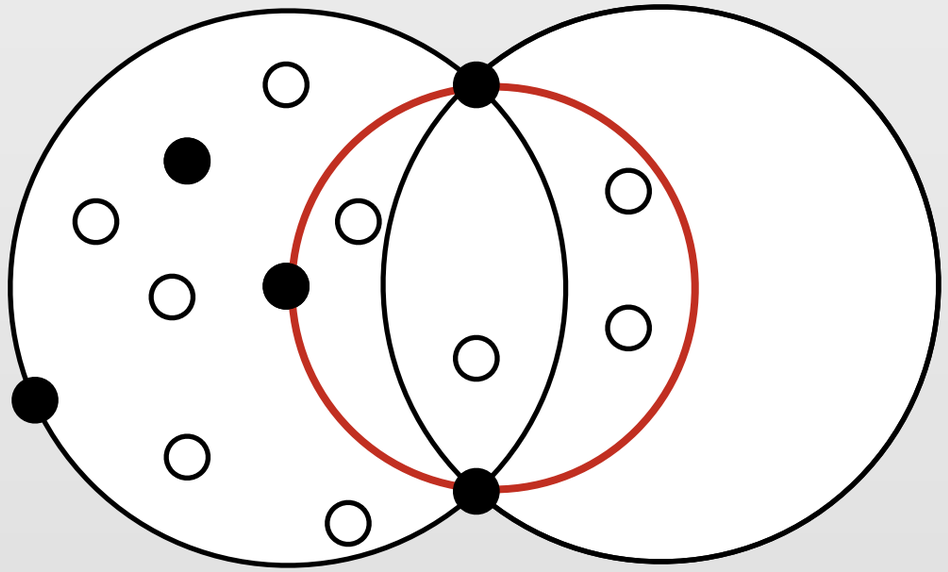
Ordering the inputs

For each D-Ball, select a 1/2d-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



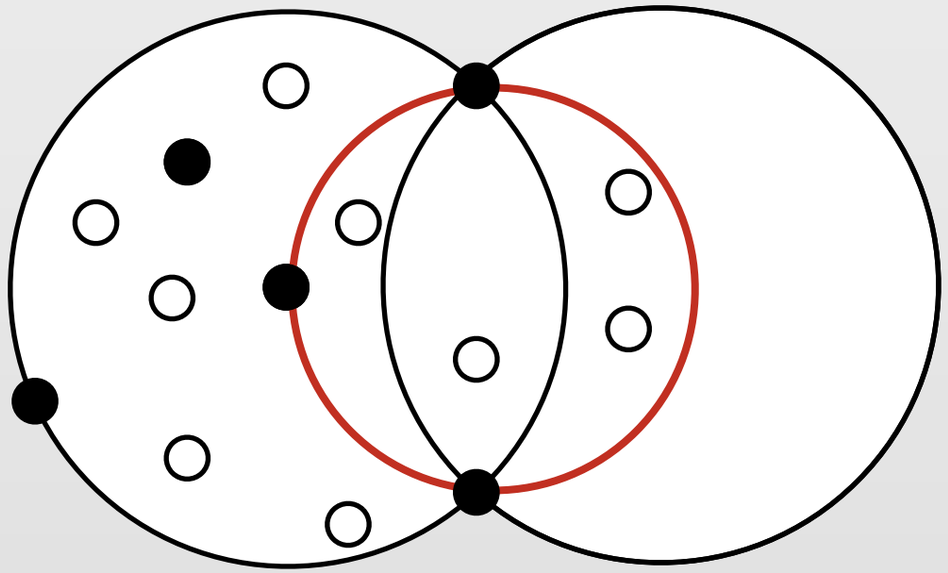
Ordering the inputs

For each D-Ball, select a 1/2d-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



Ordering the inputs

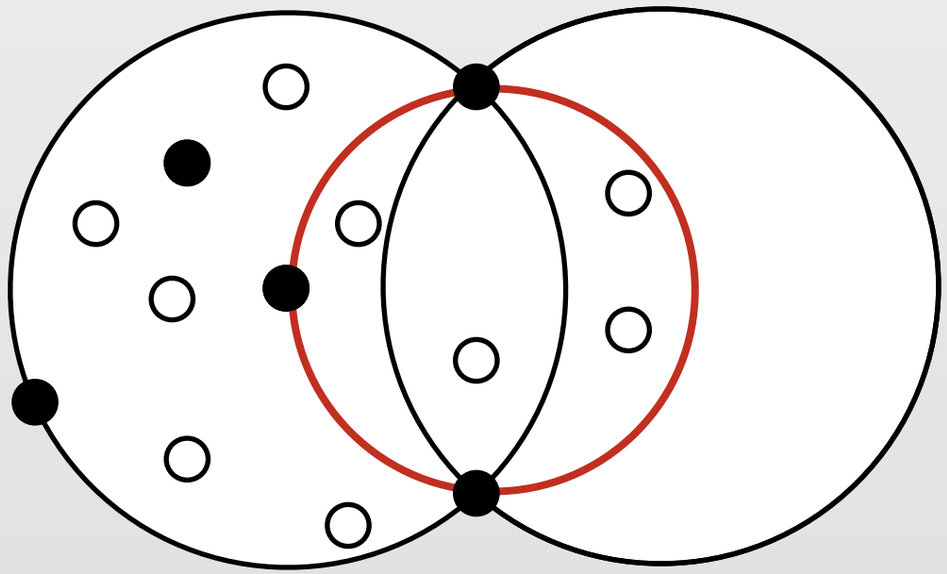
For each D-Ball, select a 1/2d-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



Lemma. *Let M be a set of vertices. If an open ball B contains no points of M, then B is contained in the union of d D-balls of M.*

Ordering the inputs

For each D-Ball, select a 1/2d-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



log(n) Rounds

Lemma. *Let M be a set of vertices. If an open ball B contains no points of M, then B is contained in the union of d D-balls of M.*

Amortized Cost of a Round is O(n)

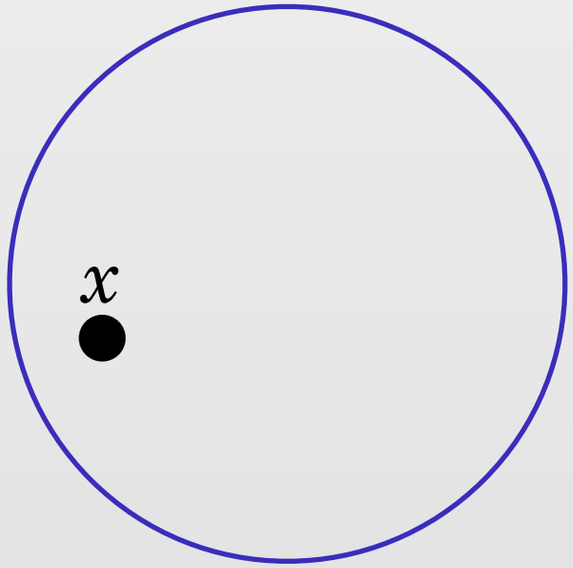
Watch an uninserted point x.
Claim: x only gets touched *O(1)* times per round.

Amortized Cost of a Round is O(n)

Watch an uninserted point x.
Claim: x only gets touched *O(1)* times per round.

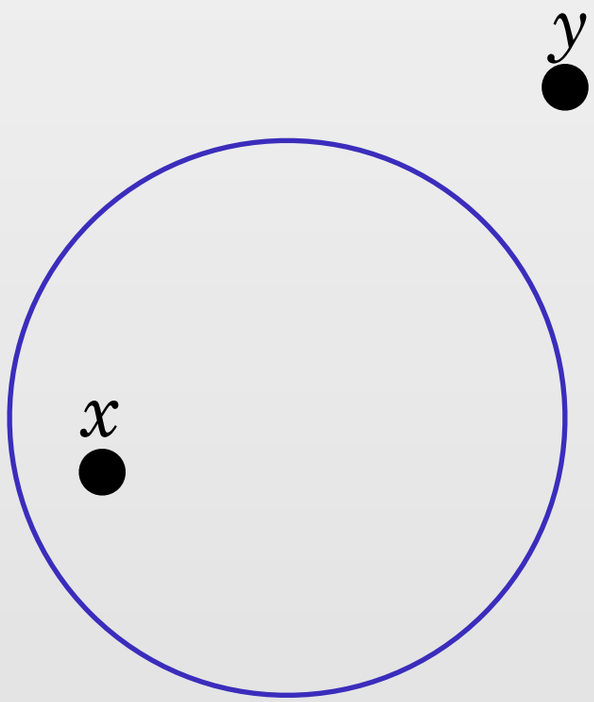
Amortized Cost of a Round is O(n)

Watch an uninserted point x.
Claim: x only gets touched *O(1)* times per round.



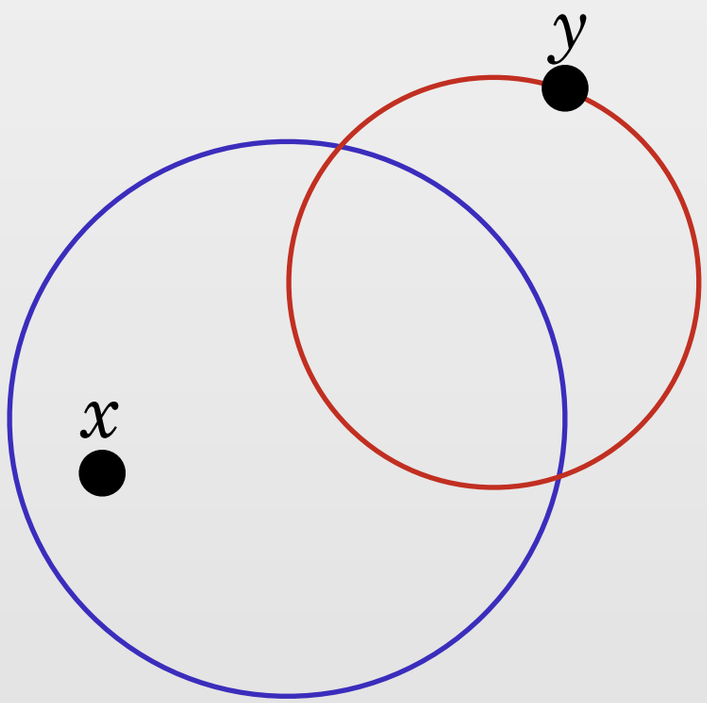
Amortized Cost of a Round is O(n)

Watch an uninserted point x.
Claim: x only gets touched *O(1)* times per round.



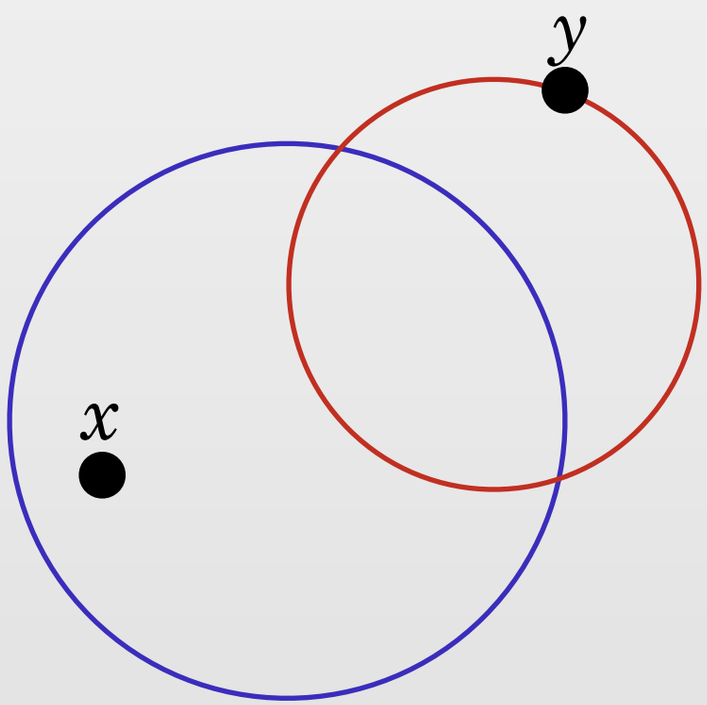
Amortized Cost of a Round is O(n)

Watch an uninserted point x.
Claim: x only gets touched *O(1)* times per round.



Amortized Cost of a Round is O(n)

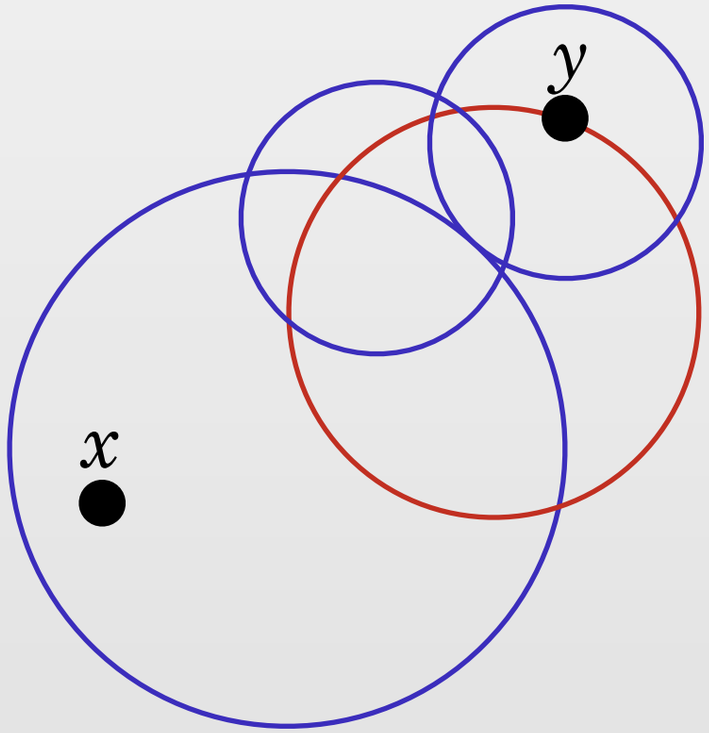
Watch an uninserted point x.
Claim: x only gets touched *O(1)* times per round.



y touches x => y is "close" to x
close = 2 hops among D-balls

Amortized Cost of a Round is O(n)

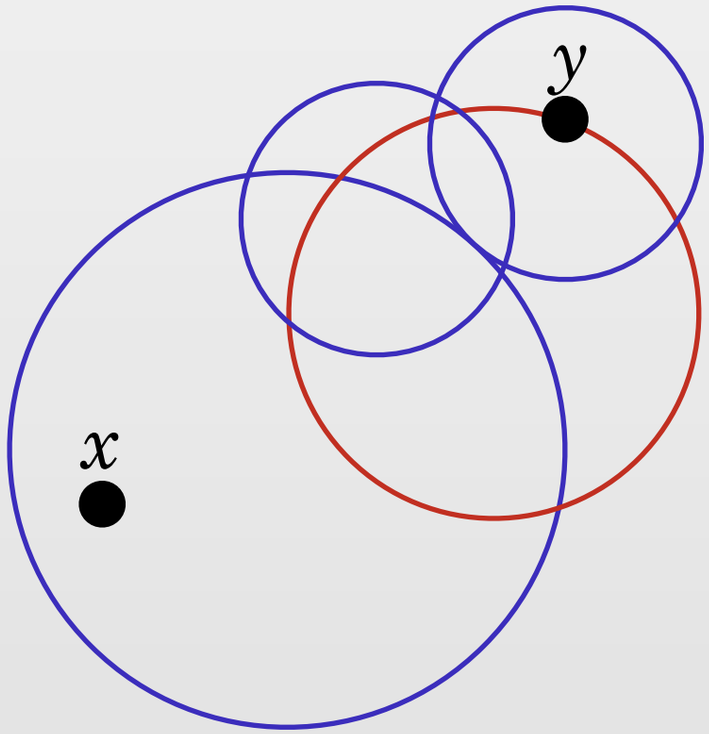
Watch an uninserted point x.
Claim: x only gets touched *O(1)* times per round.



y touches x => y is "close" to x
close = 2 hops among D-balls

Amortized Cost of a Round is O(n)

Watch an uninserted point x.
Claim: x only gets touched *O(1)* times per round.

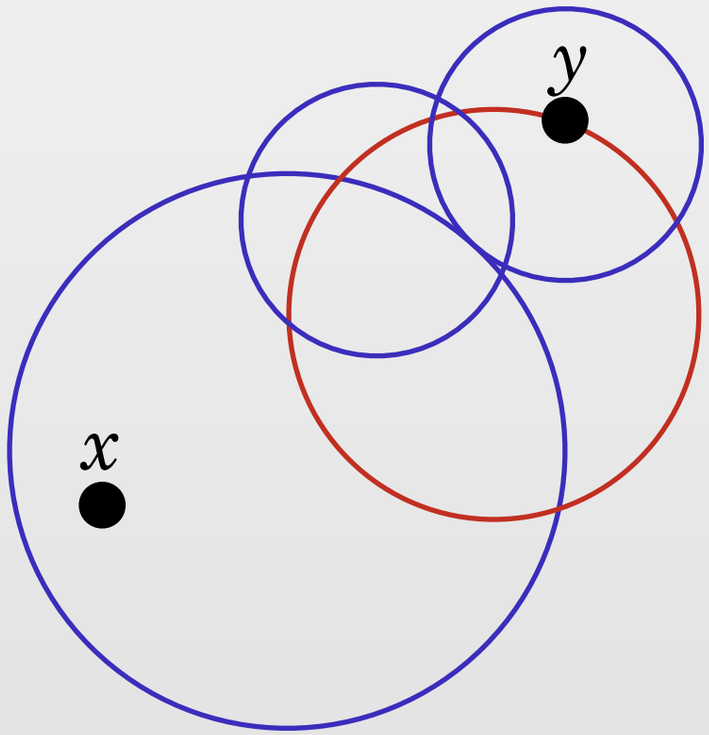


y touches x => y is "close" to x
close = 2 hops among D-balls

Only *O(1)* D-balls are within 2 hops.

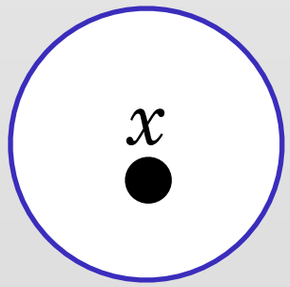
Amortized Cost of a Round is O(n)

Watch an uninserted point x.
Claim: x only gets touched *O(1)* times per round.



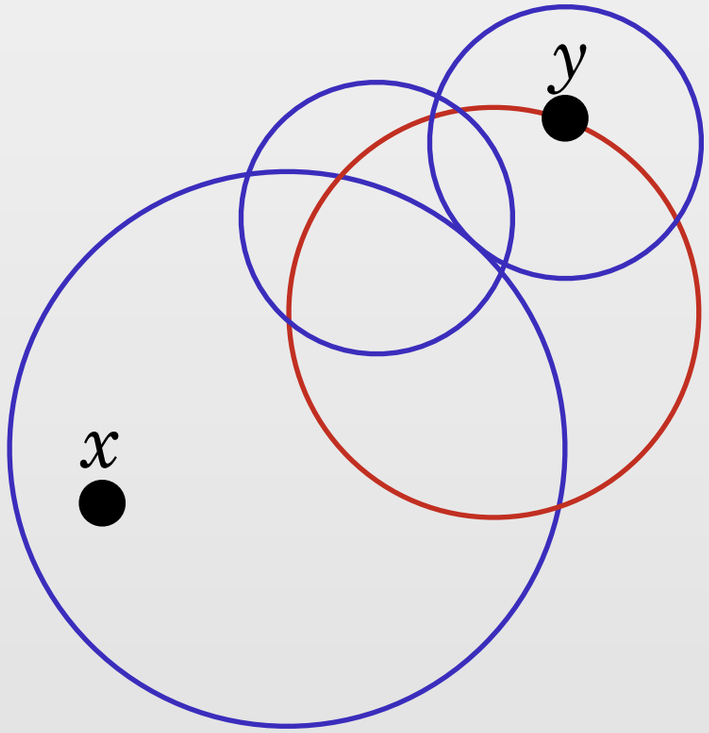
y touches x => y is "close" to x
close = 2 hops among D-balls

Only *O(1)* D-balls are within 2 hops.



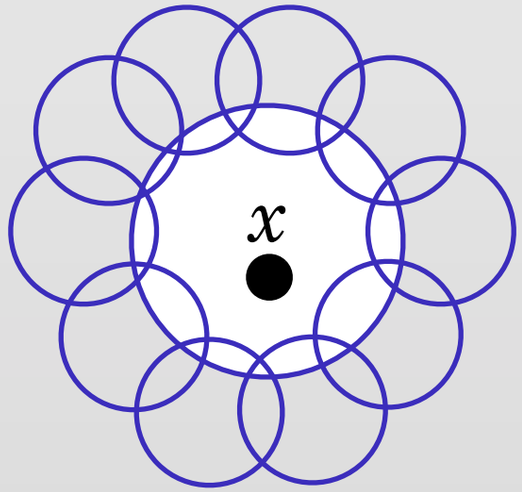
Amortized Cost of a Round is O(n)

Watch an uninserted point x.
Claim: x only gets touched *O(1)* times per round.



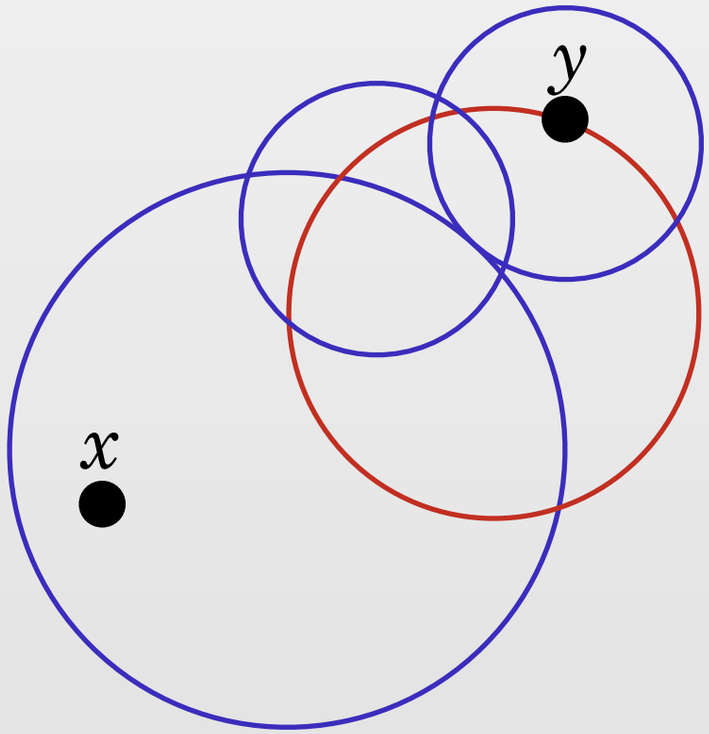
y touches x => y is "close" to x
close = 2 hops among D-balls

Only *O(1)* D-balls are within 2 hops.



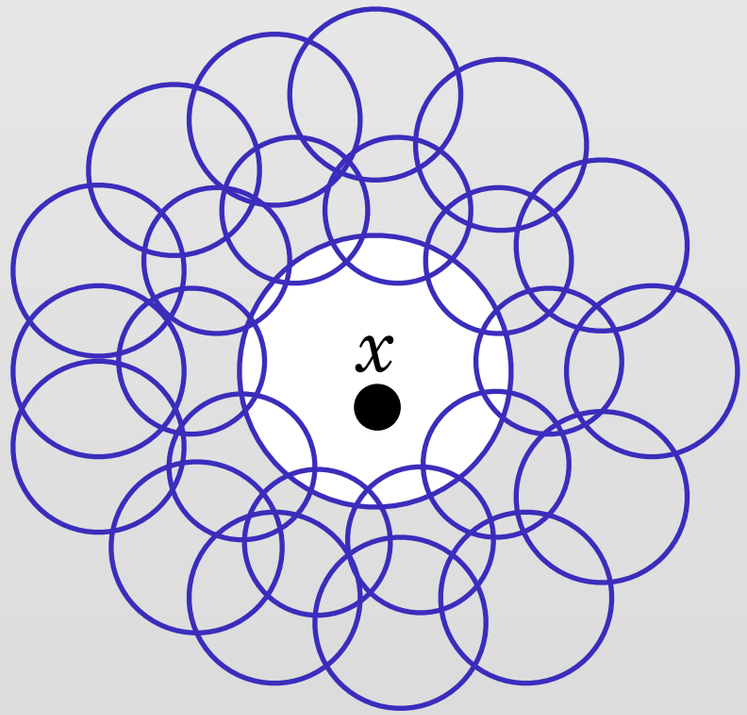
Amortized Cost of a Round is O(n)

Watch an uninserted point x.
Claim: x only gets touched *O(1)* times per round.



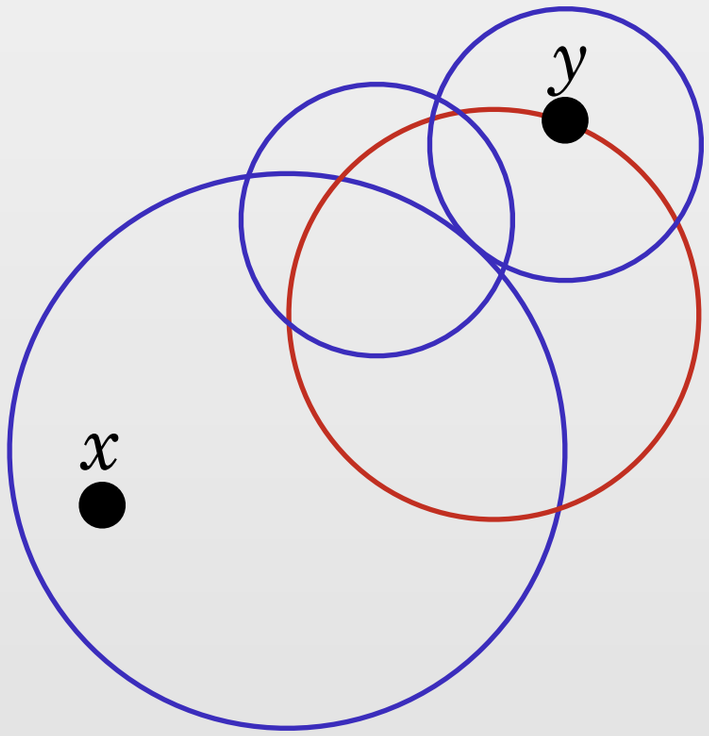
y touches x => y is "close" to x
close = 2 hops among D-balls

Only *O(1)* D-balls are within 2 hops.



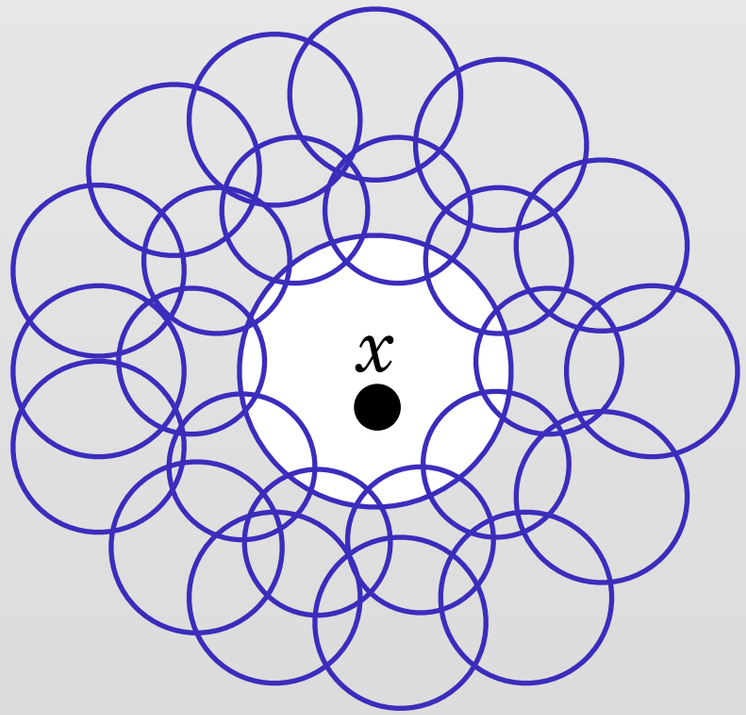
Amortized Cost of a Round is O(n)

Watch an uninserted point x.
Claim: x only gets touched *O(1)* times per round.



y touches x => y is "close" to x
close = 2 hops among D-balls

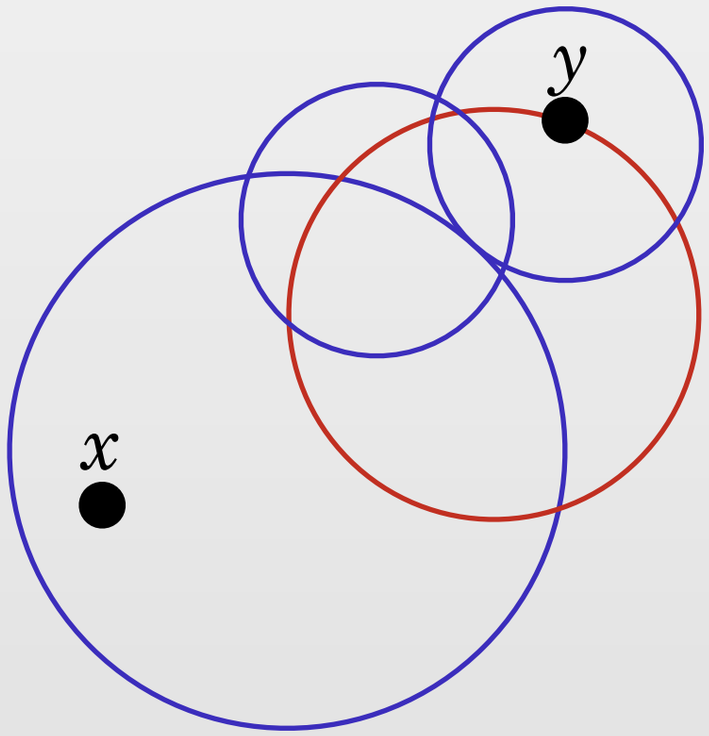
Only *O(1)* D-balls are within 2 hops.



Only *O(1)* points are added to any D-ball in a round.

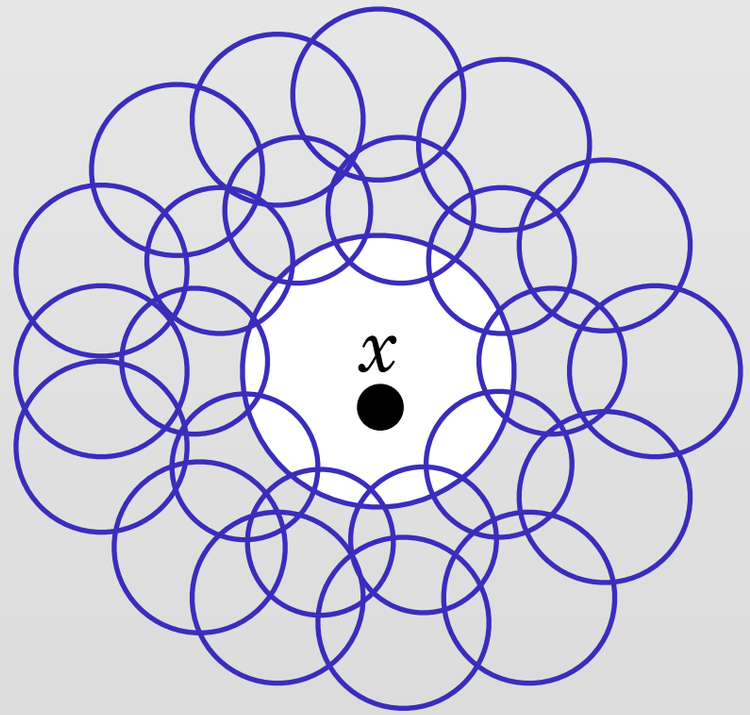
Amortized Cost of a Round is O(n)

Watch an uninserted point x.
Claim: x only gets touched *O(1)* times per round.



y touches x => y is "close" to x
close = 2 hops among D-balls

Only *O(1)* D-balls are within 2 hops.



Only *O(1)* points are added to any D-ball in a round.

O(n) total work per round.

Some Takeaways

Some Takeaways

Voronoi refinement meshes give structure to points, especially with respect to the ambient space.

Some Takeaways

Voronoi refinement meshes give structure to points, especially with respect to the ambient space.

For d not too big, meshing is fast ($O(n \log n)$).

Some Takeaways

Voronoi refinement meshes give structure to points, especially with respect to the ambient space.

For d not too big, meshing is fast ($O(n \log n)$).

There are many possible new applications, waiting to be discovered (i.e. any time you would have used a Voronoi diagram).

Thanks.