Time-Space Trade-offs for Triangulations and Voronoi Diagrams

Matias Korman
National institute of informatics.
Tokyo, Japan

Wolfgang Mulzer
Institut für Informatik,
Freie Universität
Berlin. Germany

André van Renssen
National institute of informatics.
Tokyo, Japan

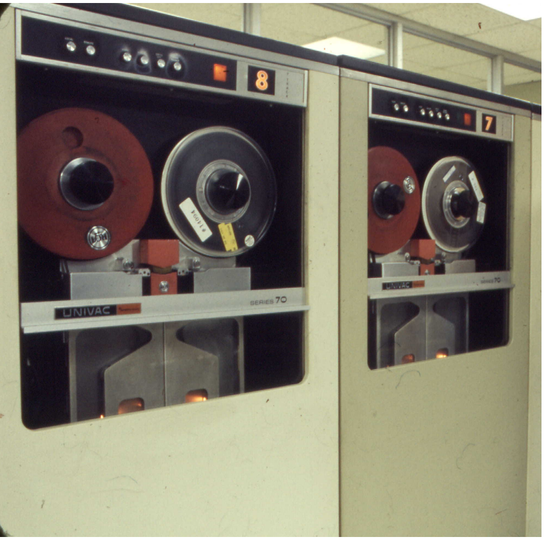
Marcel Roeloffzen
National institute of informatics.
Tokyo, Japan

Paul Seiferth
Institut für Informatik,
Freie Universität
Berlin. Germany

Yannik Stein
Institut für Informatik,
Freie Universität
Berlin. Germany

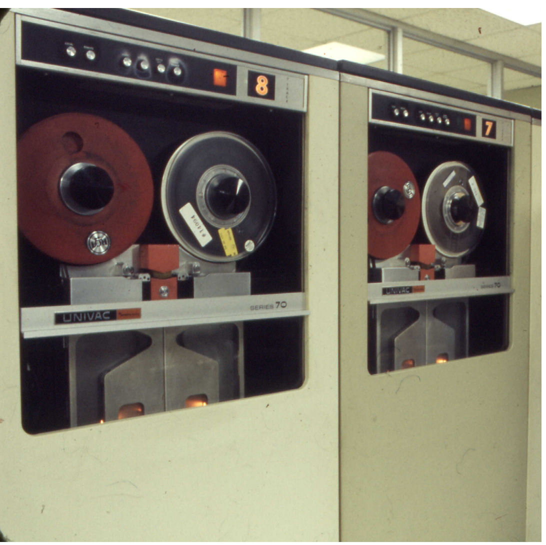
Limited Memory

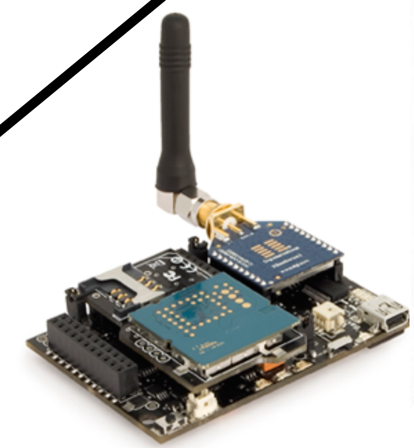
Started in the 70's

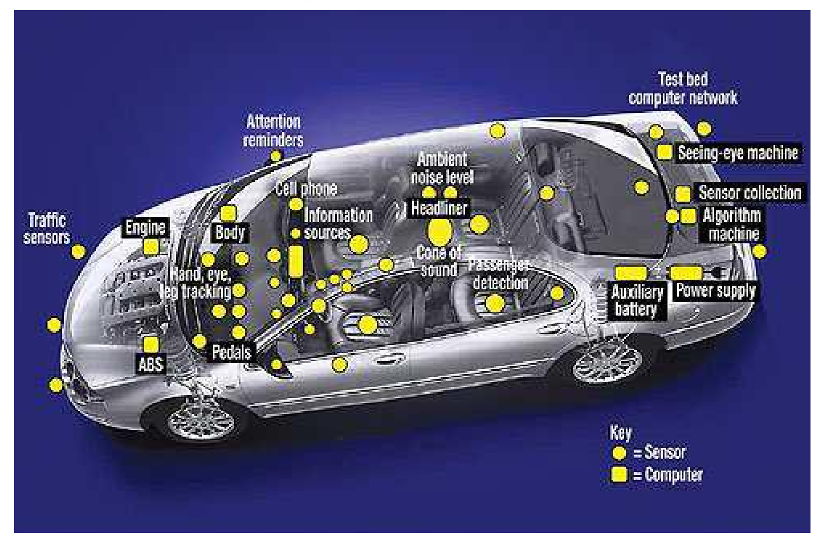


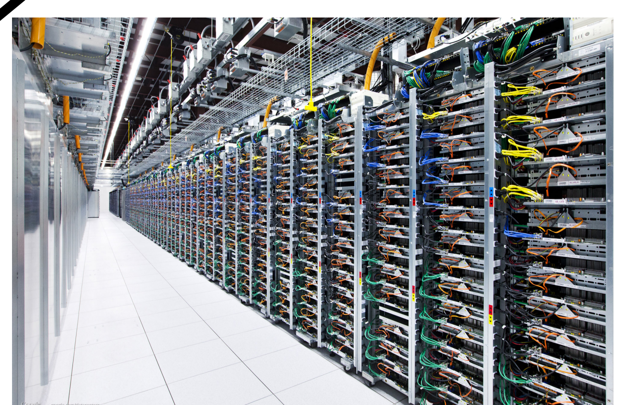
Limited Memory

Started in the 70's





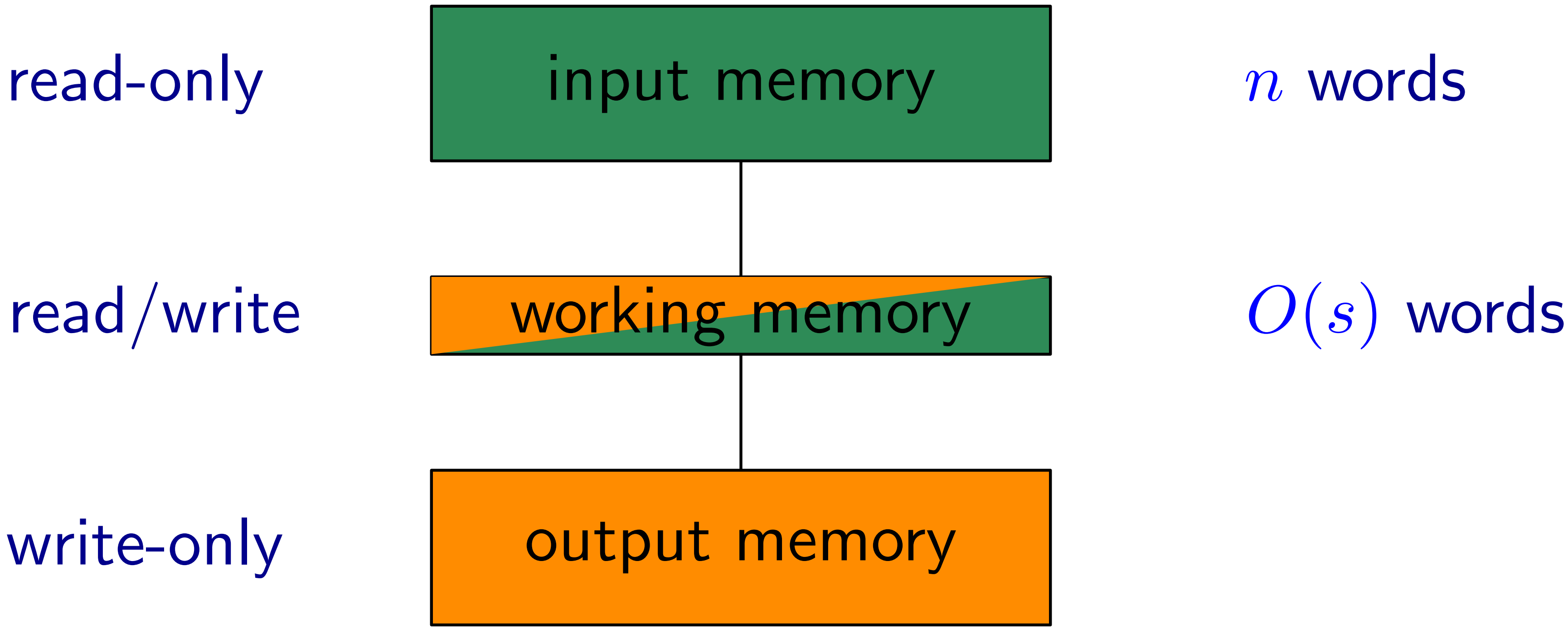




still relevant today

Model

Word RAM with unit costs, parameter s

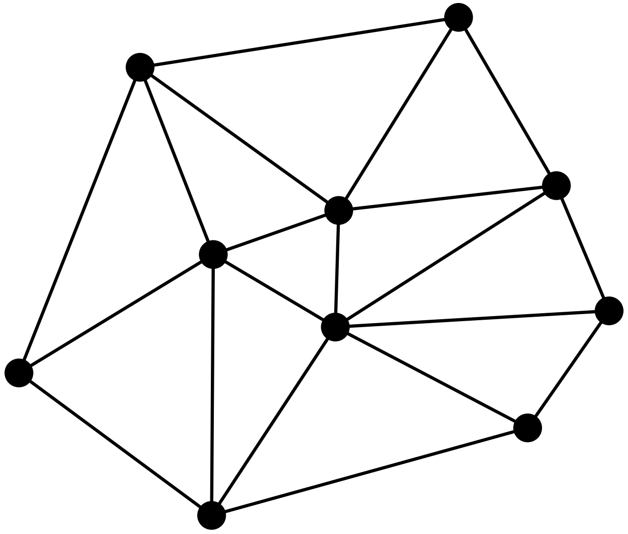


word $= \Omega(\log n)$ bits

Our Results

Input set P of n points in $\mathbb{R}^2$

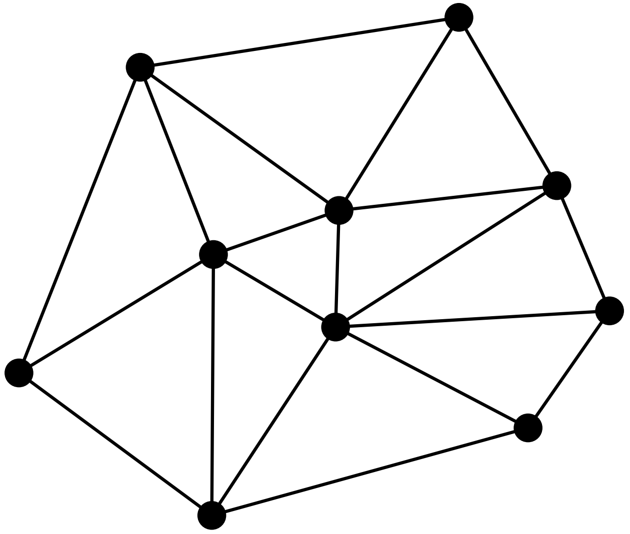
Output edges of a triangulation in arbitrary order



Our Results

Input set P of n points in $\mathbb{R}^2$

Output edges of a triangulation in arbitrary order

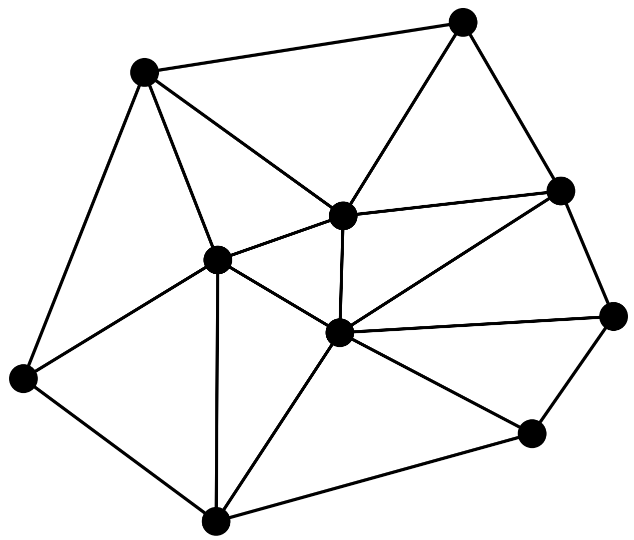


Related Results

- $O(n \log n)$ time with $O(n)$ space
- $O(n^2)$ time with $O(1)$ space [Asano et al. '11]

Our Results

Input set P of n points in $\mathbb{R}^2$
Output edges of a triangulation in arbitrary order



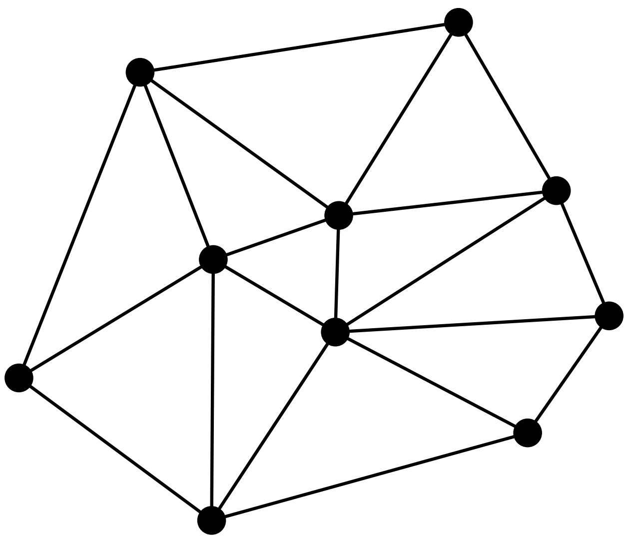
Related Results

- $O(n \log n)$ time with $O(n)$ space
- $O(n^2)$ time with $O(1)$ space
  [Asano et al. '11]

Theorem. Let $P \subset \mathbb{R}^2$ be a set of n points. Then, we can report a triangulation of P in $O(n^2/s + n \log n \log s)$ time using $O(s)$ space.

Our Results

Input set P of n points in $\mathbb{R}^2$
Output edges of a triangulation in arbitrary order



Related Results

- $O(n \log n)$ time with $O(n)$ space
- $O(n^2)$ time with $O(1)$ space [Asano et al. '11]

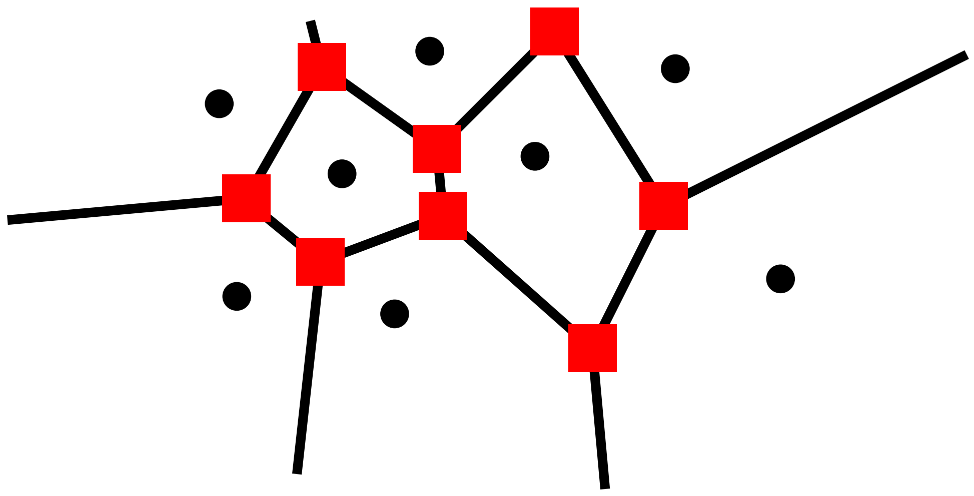
Theorem. Let $P \subset \mathbb{R}^2$ be a set of n points. Then, we can report a triangulation of P in $O(n^2/s + n \log n \log s)$ time using $O(s)$ space.

- $O(1)$ space: $O(n^2)$ time
- $O(n)$ space: $O(n \log^2 n)$ time

Our Results — Continued

Input set P of points in $\mathbb{R}^2$

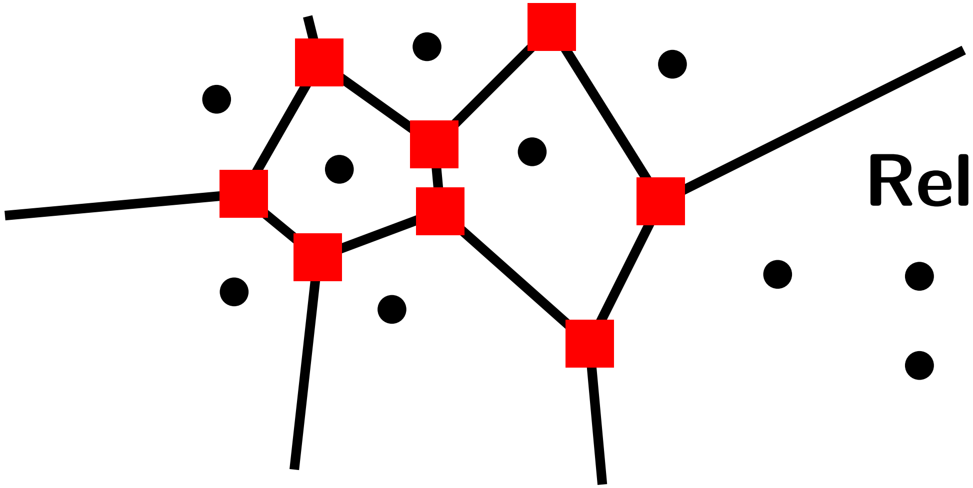
Output vertices of the Voronoi diagram in arbitrary order



Our Results — Continued

Input set P of points in $\mathbb{R}^2$

Output vertices of the Voronoi diagram in arbitrary order

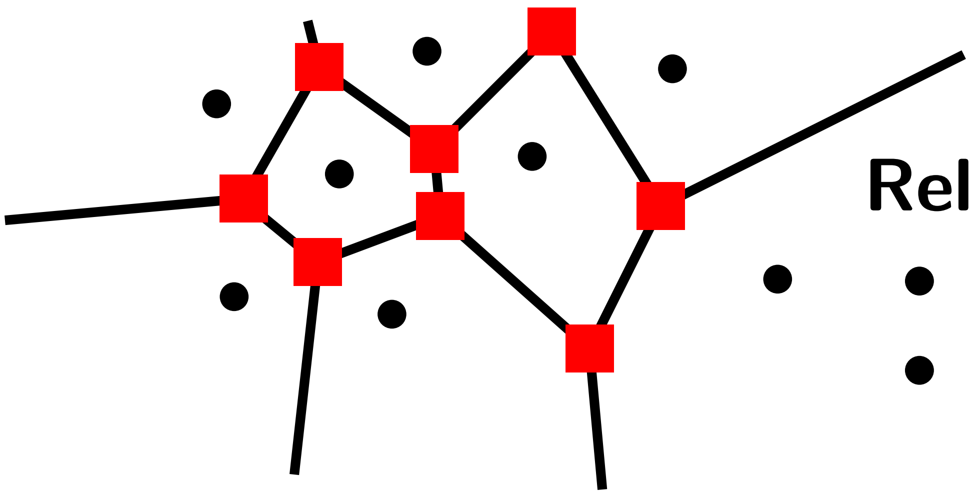


Related Results

- $O(n \log n)$ time with $O(n)$ space
- $O(n^2)$ time with $O(1)$ space [Asano et al. '11]

Our Results — Continued

Input set P of points in $\mathbb{R}^2$
Output vertices of the Voronoi diagram in arbitrary order



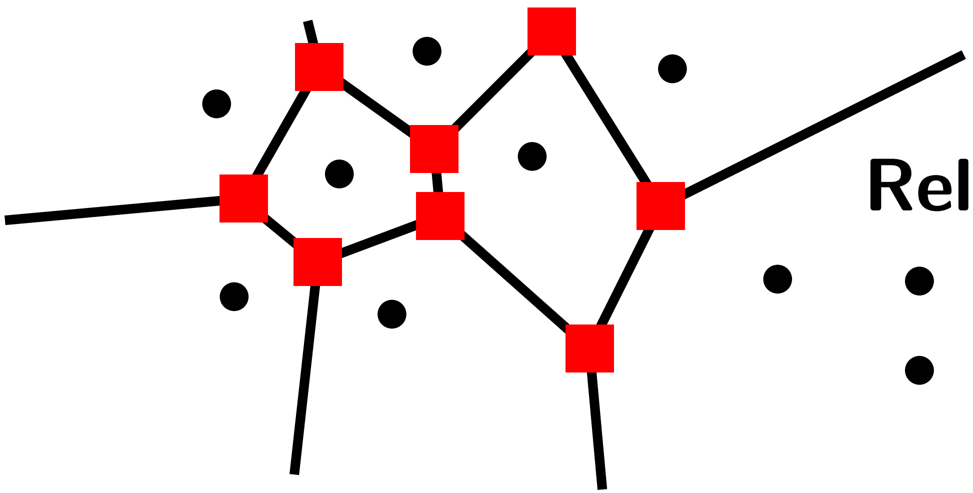
Related Results

- $O(n \log n)$ time with $O(n)$ space
- $O(n^2)$ time with $O(1)$ space
  [Asano et al. '11]

Theorem. Reporting Voronoi diagrams of a set of n points in the plane can be done in $O((n^2/s) \log s + n \log s \log^* s)$ expected time using $O(s)$ space.

Our Results — Continued

Input set P of points in $\mathbb{R}^2$
Output vertices of the Voronoi diagram in arbitrary order



Related Results

- $O(n \log n)$ time with $O(n)$ space
- $O(n^2)$ time with $O(1)$ space
  [Asano et al. '11]

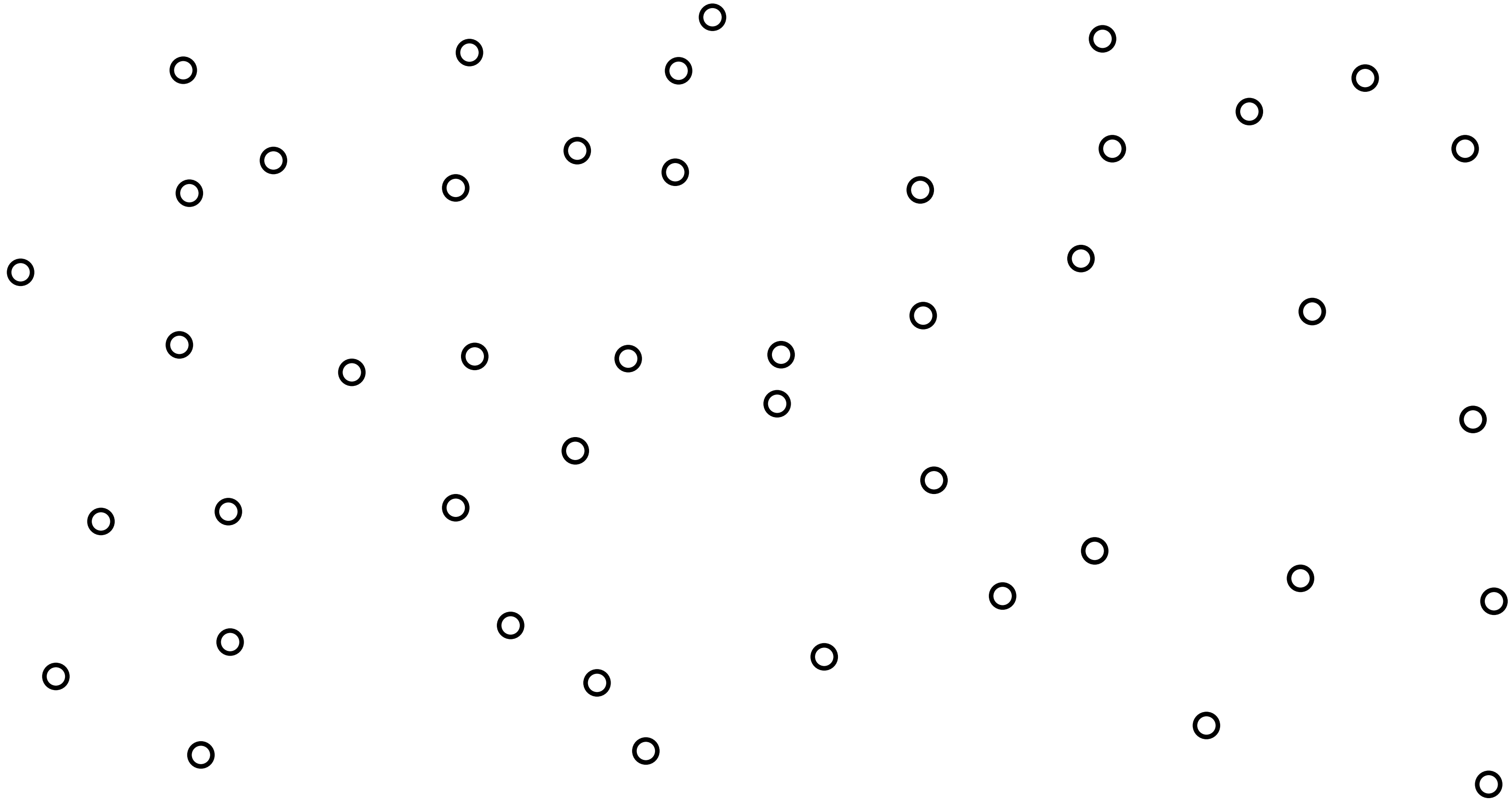
Theorem. Reporting Voronoi diagrams of a set of n points in the plane can be done in $O((n^2/s)\log s + n \log s \log^* s)$ expected time using $O(s)$ space.

- $O(1)$ space: $O(n^2)$ time
- $O(n)$ space: $O(n \log n \log^* n)$ time

Computing $\mathrm{VD}(P)$ in $O(s)$ space — Overview

Phase I: Sampling

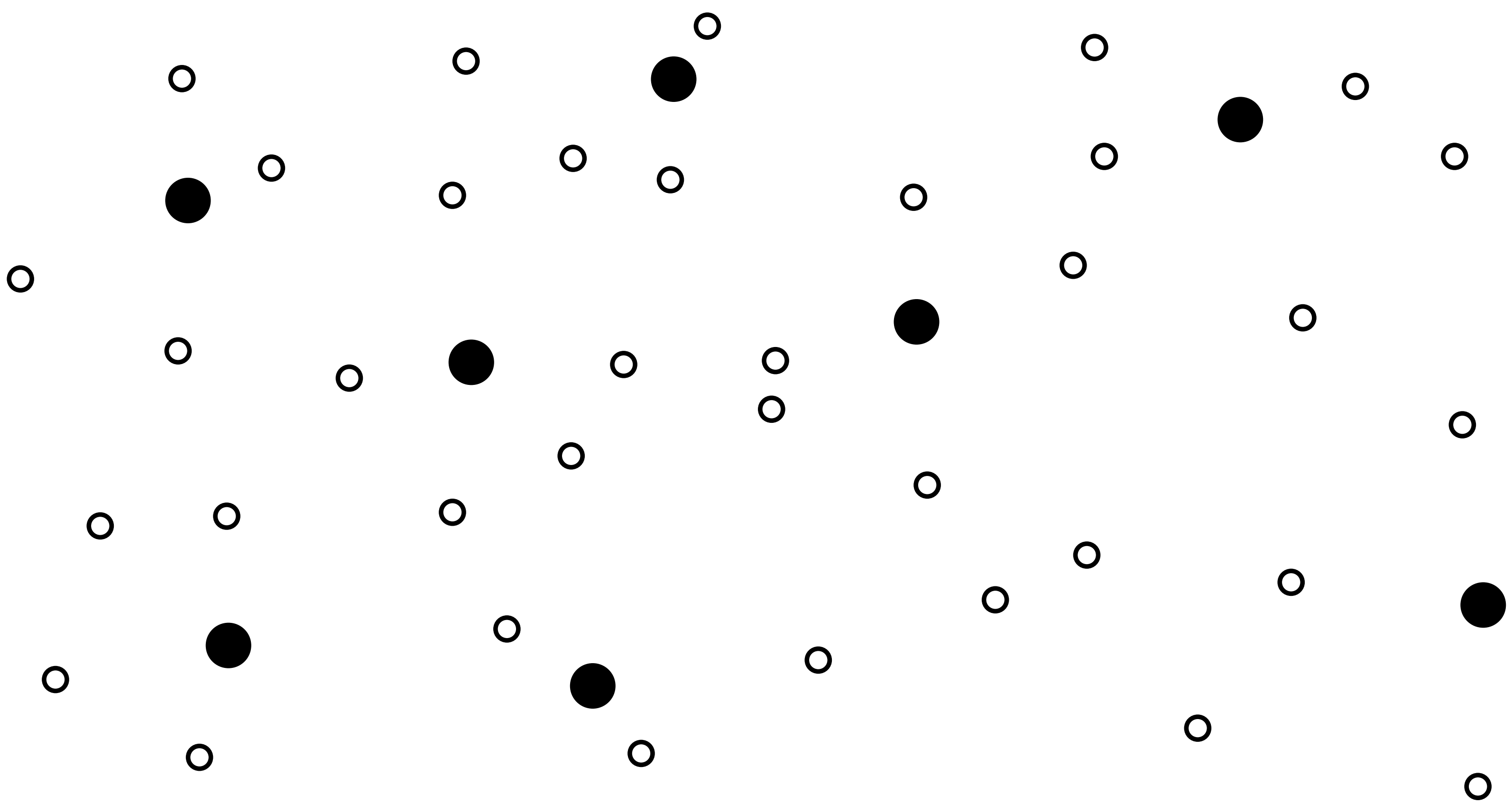
- Take random sample $R \subset P$ of size $O(s)$



Computing $\mathrm{VD}(P)$ in $O(s)$ space — Overview

Phase I: Sampling

- Take random sample $R \subset P$ of size $O(s)$



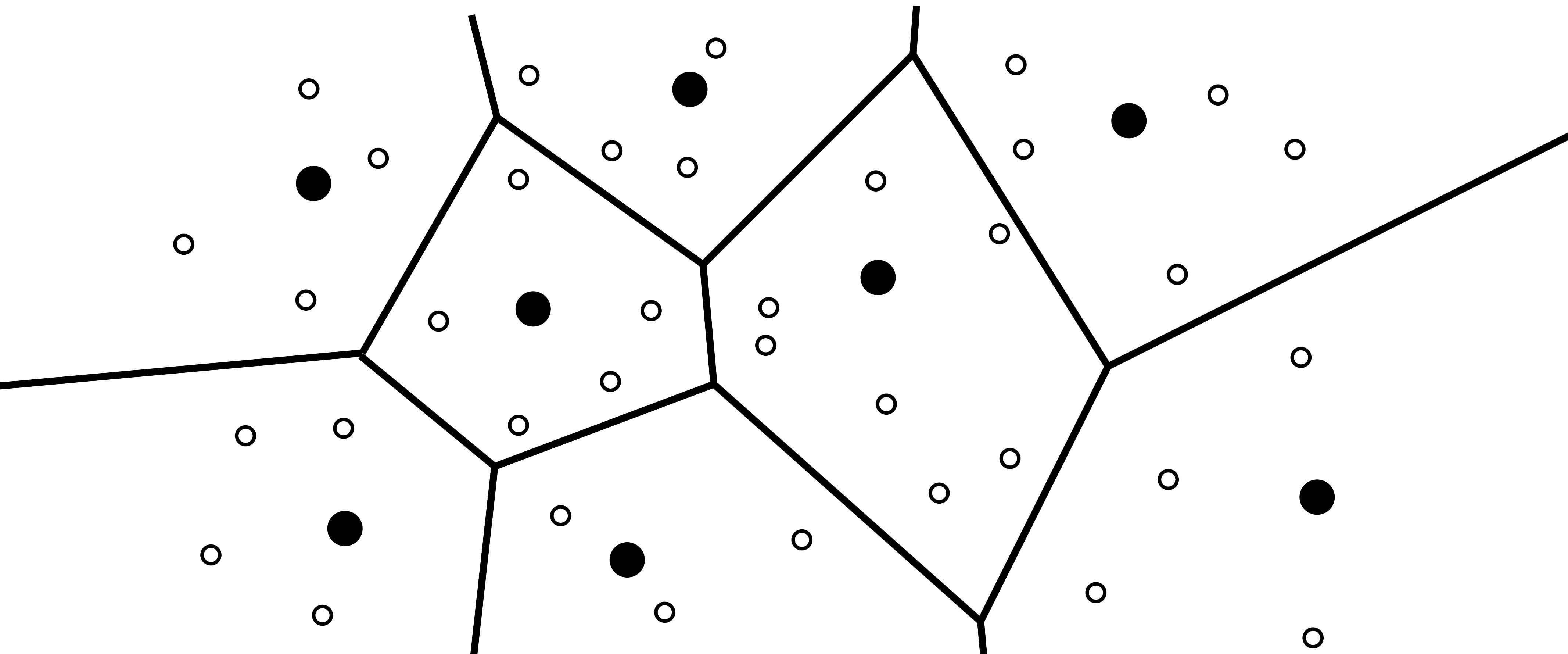
Computing $\mathrm{VD}(P)$ in $O(s)$ space — Overview

Phase I: Sampling

- Take random sample $R \subset P$ of size $O(s)$

Phase II: Compute $\mathrm{VD}(P)$

- Compute $\mathrm{VD}(R)$



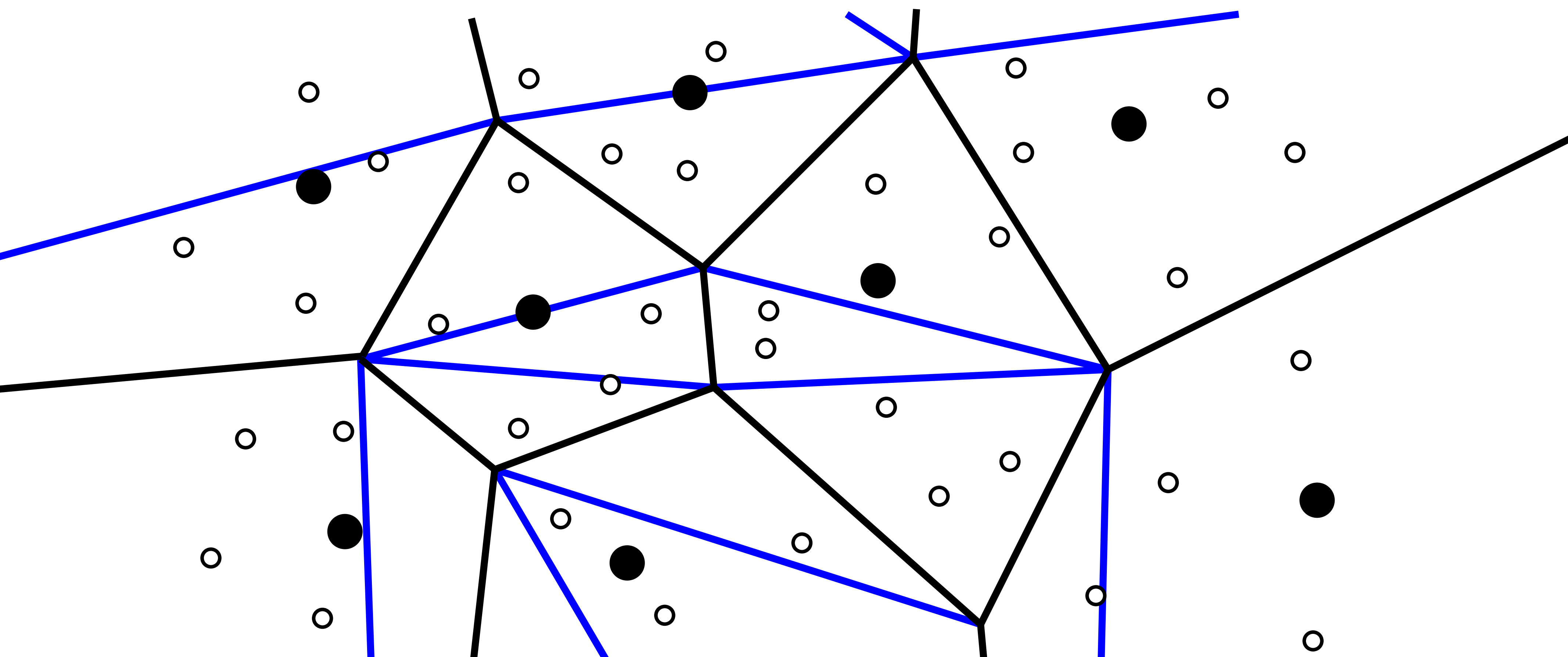
Computing $\mathrm{VD}(P)$ in $O(s)$ space — Overview

Phase I: Sampling

- Take random sample $R \subset P$ of size $O(s)$

Phase II: Compute $\mathrm{VD}(P)$

- Compute $\mathrm{VD}(R)$
- Triangulate cells of $\mathrm{VD}(R)$



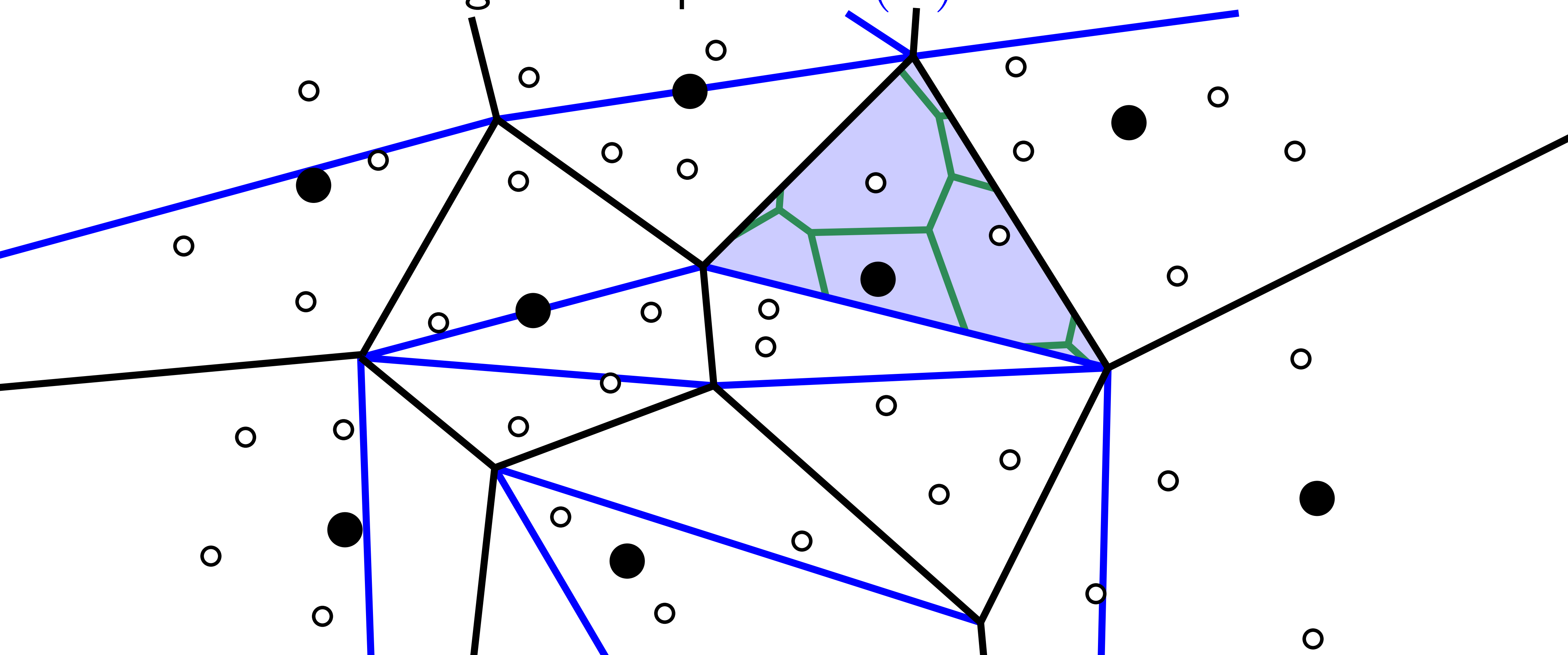
Computing $\mathrm{VD}(P)$ in $O(s)$ space — Overview

Phase I: Sampling

- Take random sample $R \subset P$ of size $O(s)$

Phase II: Compute $\mathrm{VD}(P)$

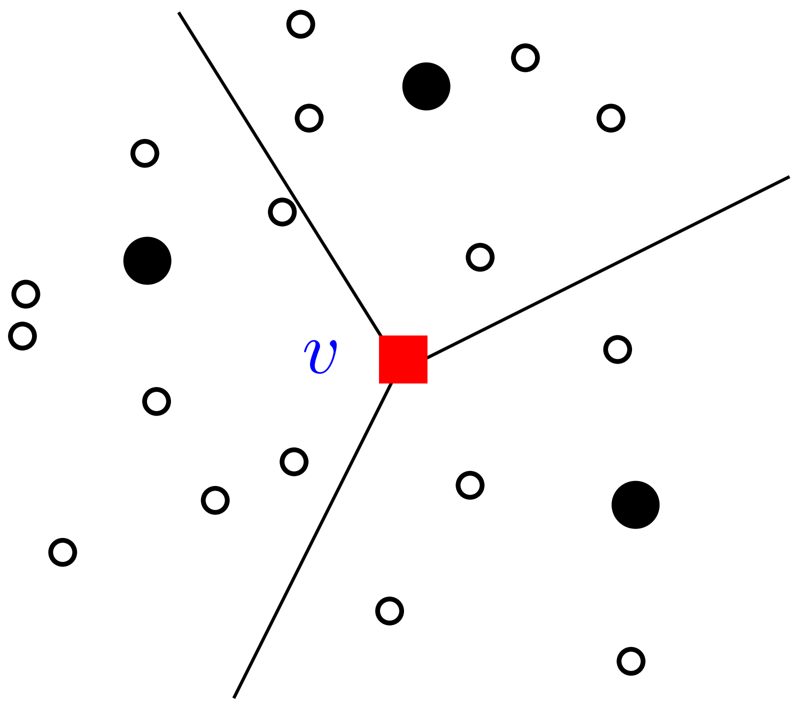
- Compute $\mathrm{VD}(R)$
- Triangulate cells of $\mathrm{VD}(R)$
- For each triangle Δ: report $\mathrm{VD}(P) \cap \Delta$



Triangles Can Be Handled Locally

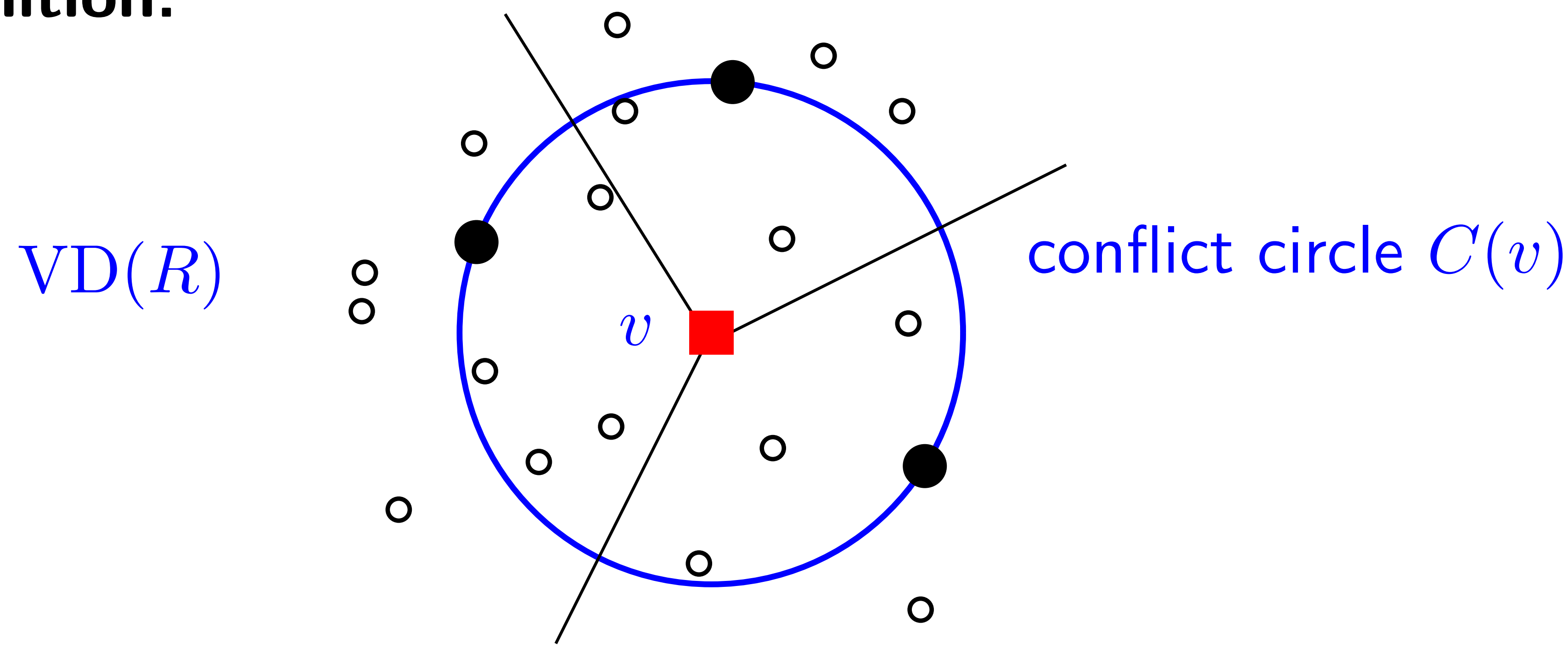
Definition.

$\mathrm{VD}(R)$



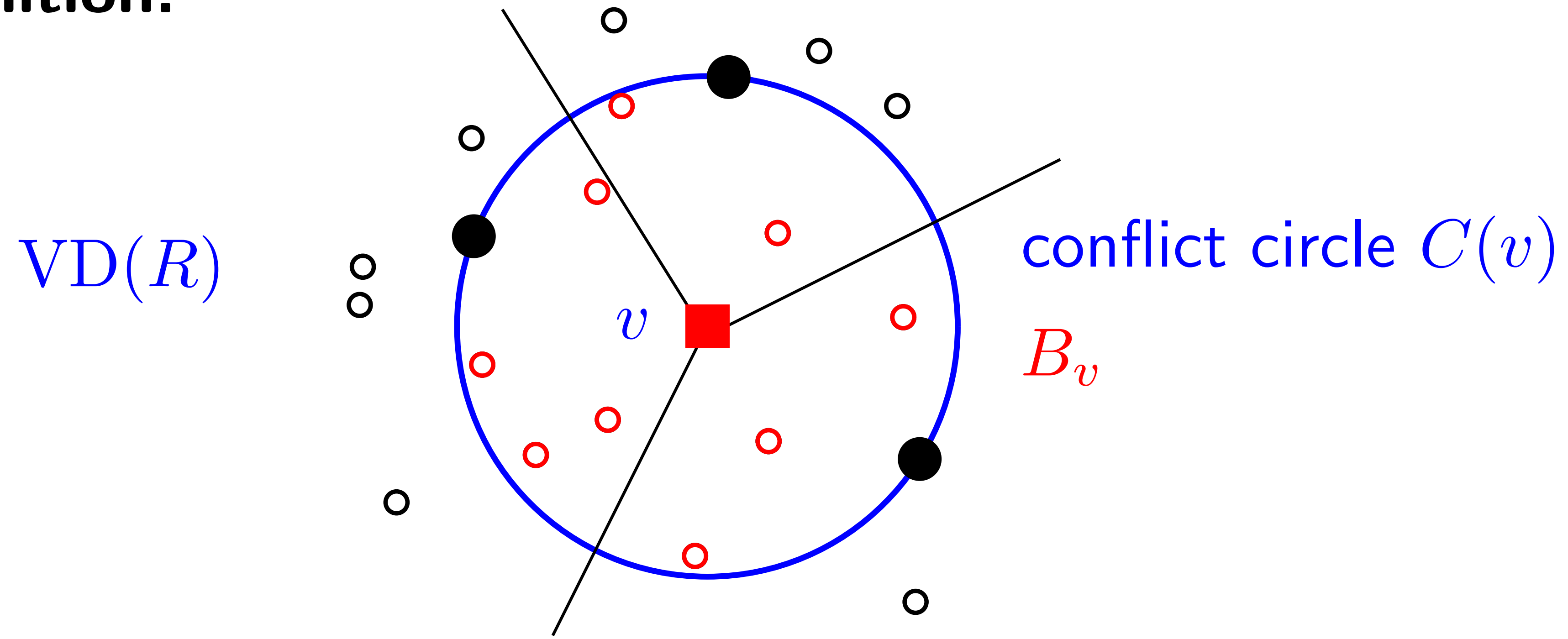
Triangles Can Be Handled Locally

Definition.



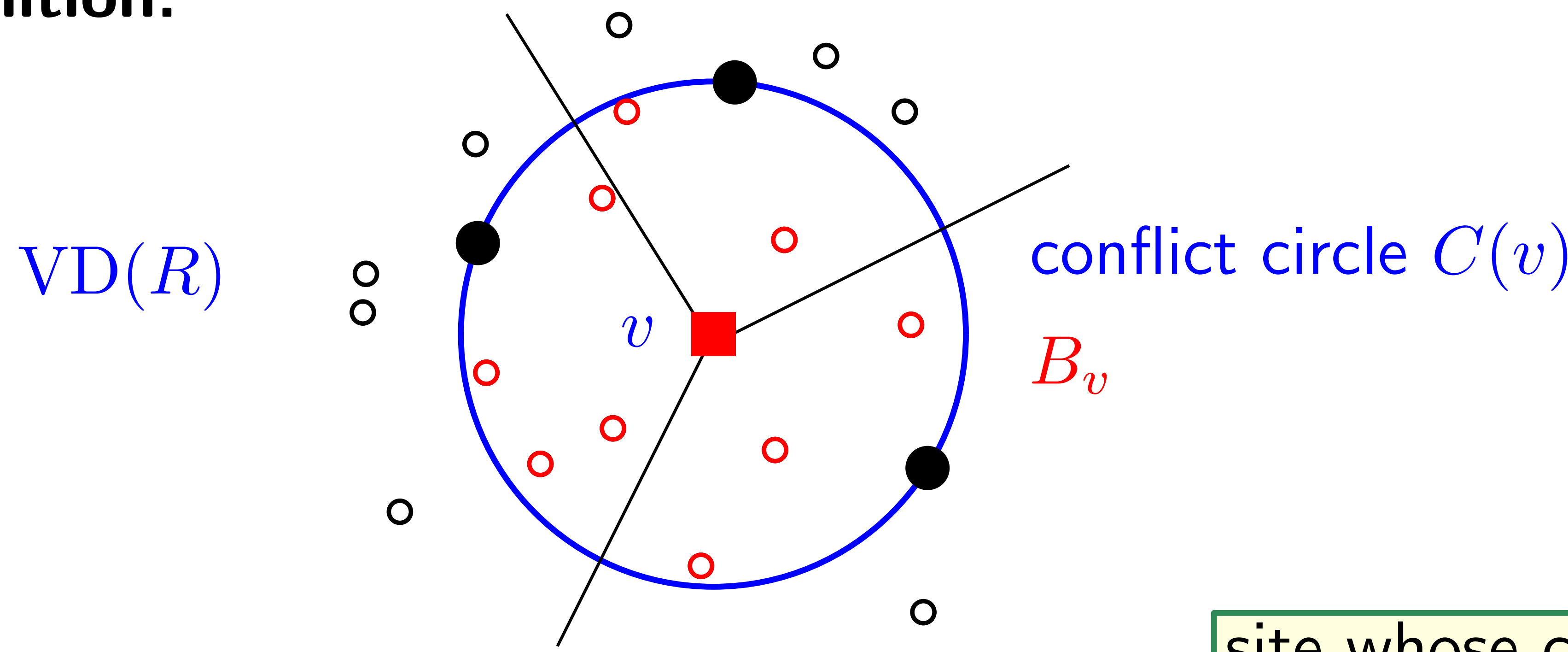
Triangles Can Be Handled Locally

Definition.



Triangles Can Be Handled Locally

Definition.



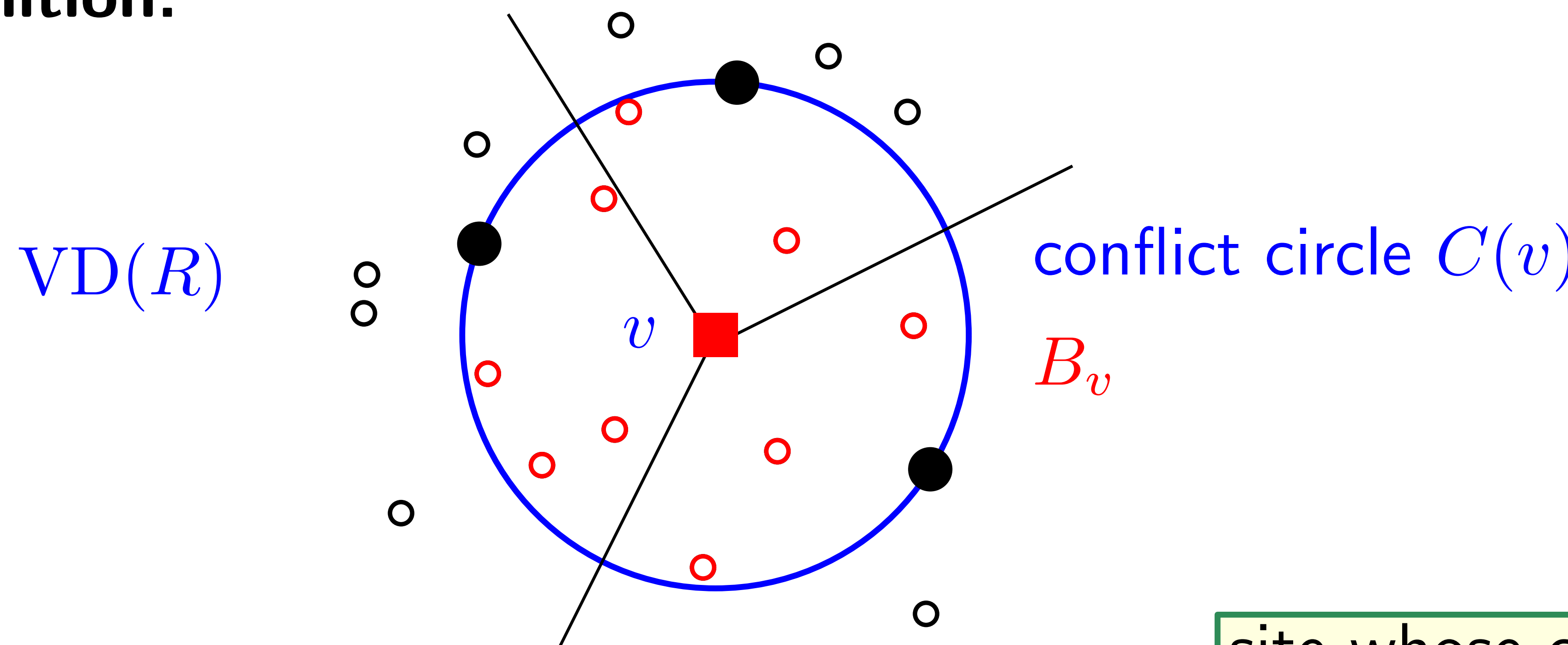
site whose cell contains Δ

Lemma. Let $\Delta = \{v_1, v_2, v_3\}$ be a triangle in the triangulation of $\mathrm{VD}(R)$. Then,

$$\mathrm{VD}(P) \cap \Delta = \mathrm{VD}(\underbrace{B_{v_1} \cup B_{v_2} \cup B_{v_3} \cup \{s\}}_{:=B_\Delta}) \cap \Delta$$

Triangles Can Be Handled Locally

Definition.



Lemma. Let $\Delta = \{v_1, v_2, v_3\}$ be a triangle in the triangulation of $\mathrm{VD}(R)$. Then,

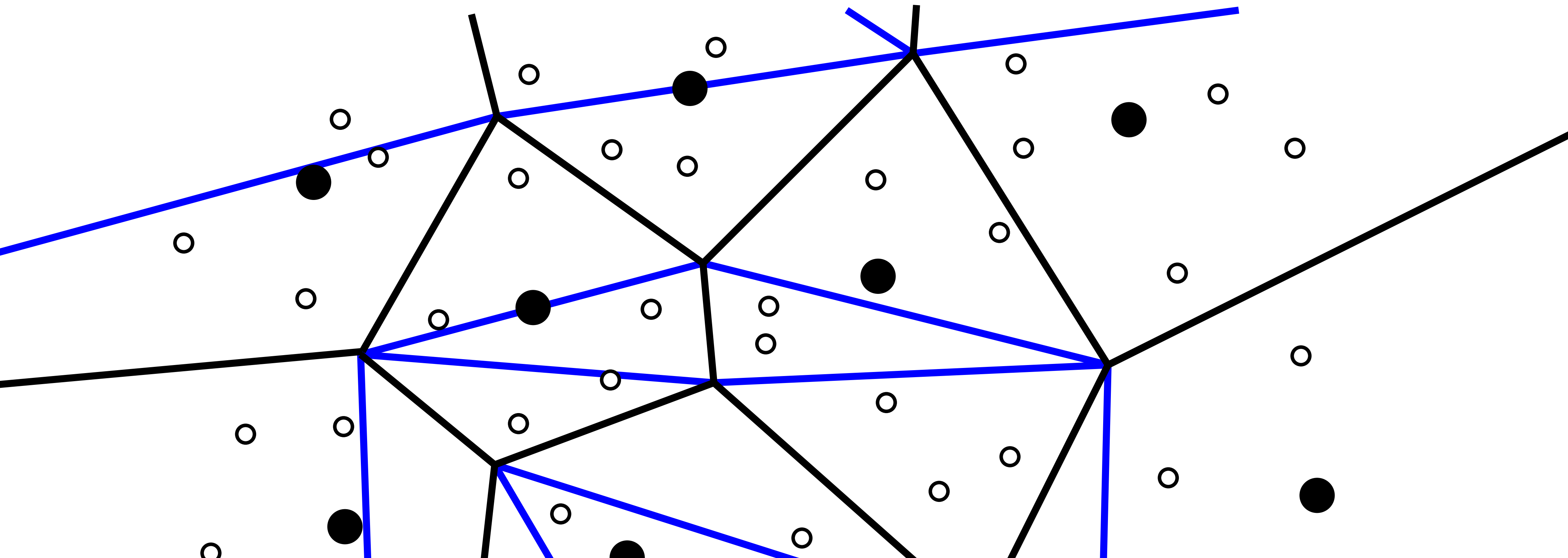
site whose cell contains Δ

$$\mathrm{VD}(P) \cap \Delta = \mathrm{VD}(\underbrace{B_{v_1} \cup B_{v_2} \cup B_{v_3} \cup \{s\}}_{:=B_\Delta}) \cap \Delta$$

$O(s)$ triangles $\rightarrow$ want $B_\Delta = O(n/s)$ for all triangles Δ

Phase II: Compute $\mathrm{VD}(P)$

Assumption: for each Δ: $B_\Delta = O(n/s)$

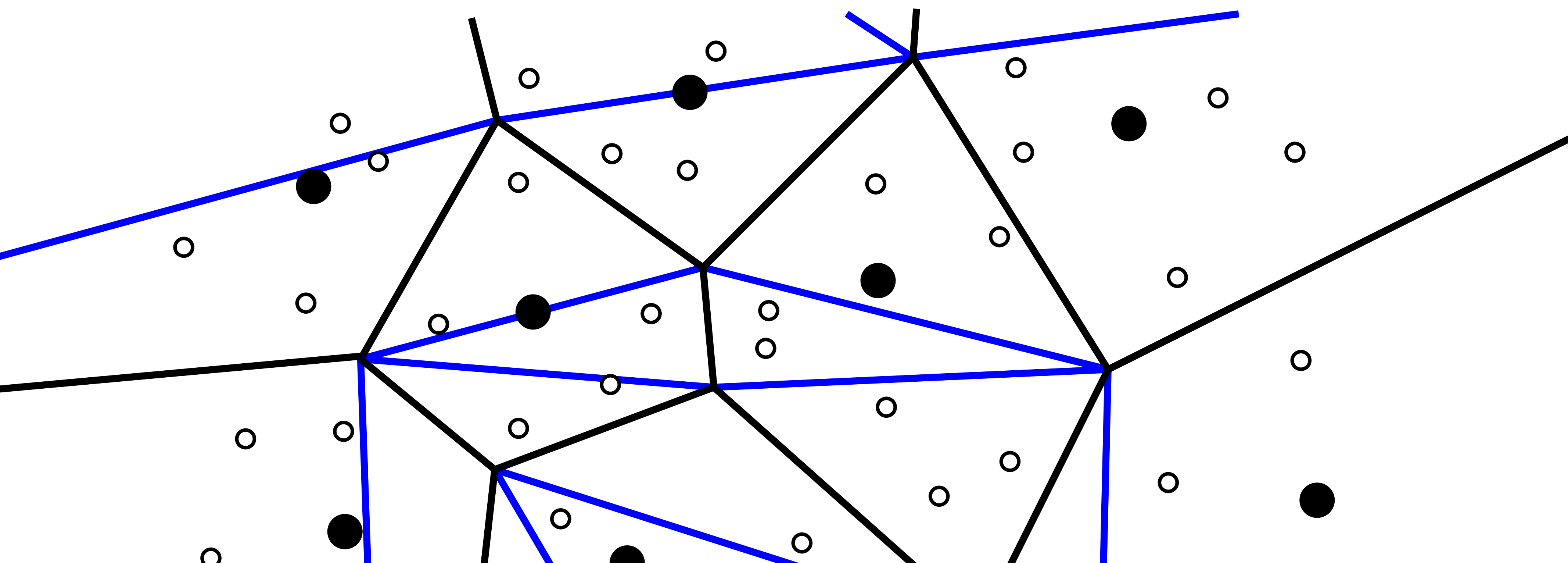


Phase II: Compute $\mathrm{VD}(P)$

Assumption: for each Δ: $B_\Delta = O(n/s)$

Run in parallel for each Δ the algorithm by Asano et al.

- $O(n^2/s^2)$ time and $O(1)$ space

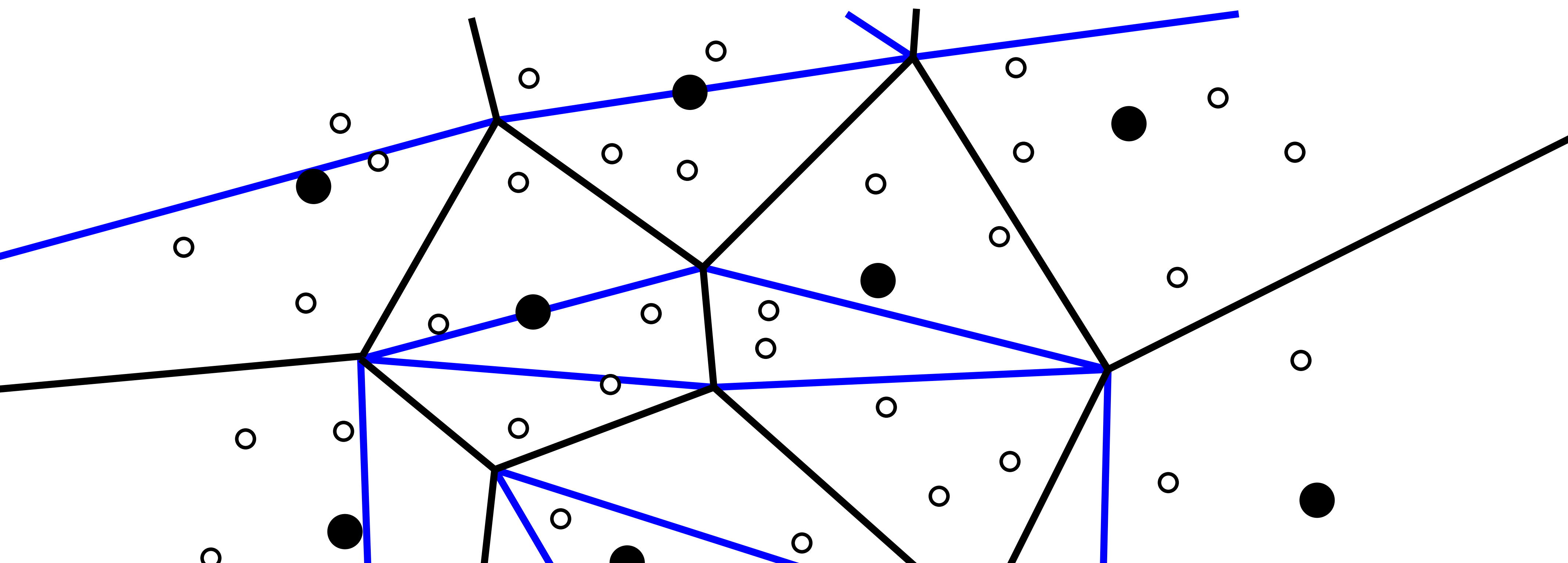


Phase II: Compute $\mathrm{VD}(P)$

Assumption: for each Δ: $B_\Delta = O(n/s)$

Run in parallel for each Δ the algorithm by Asano et al.

- $O(n^2/s^2)$ time and $O(1)$ space
- requires $O(n/s)$ scans of B_Δ

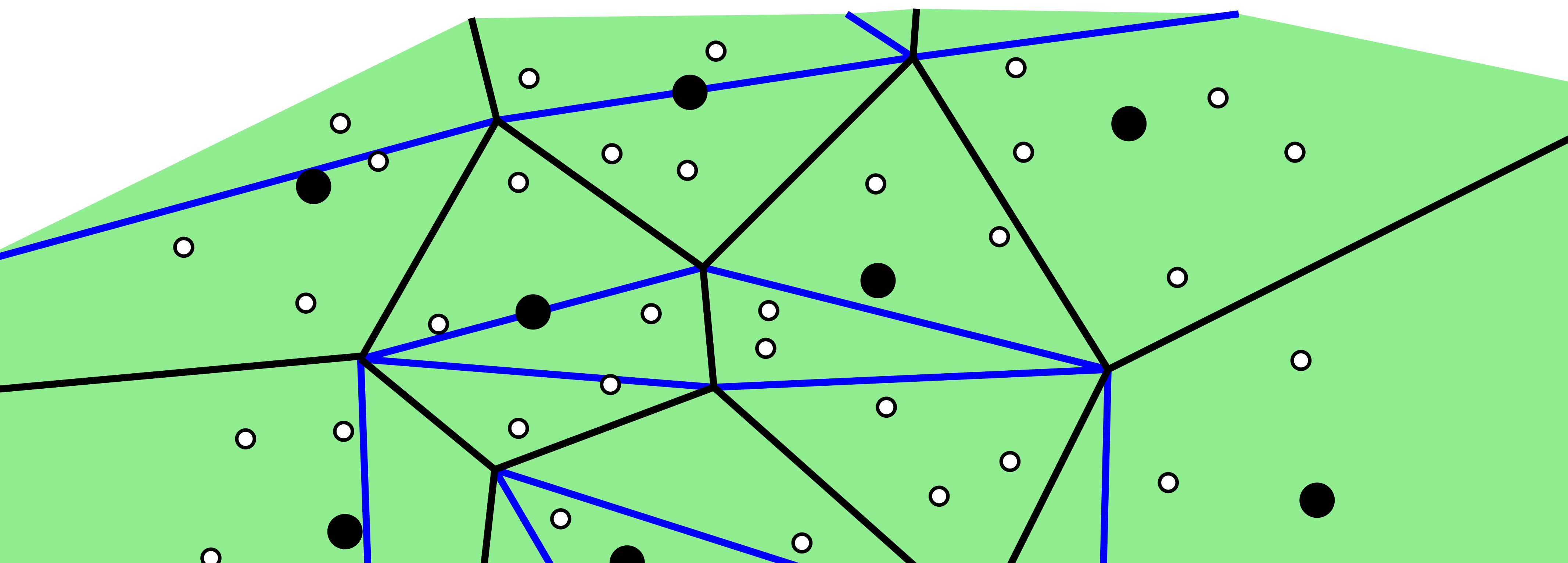


Phase II: Compute $\mathrm{VD}(P)$

Assumption: for each Δ: $B_\Delta = O(n/s)$

Run in parallel for each Δ the algorithm by Asano et al.

- $O(n^2/s^2)$ time and $O(1)$ space
- requires $O(n/s)$ scans of B_Δ

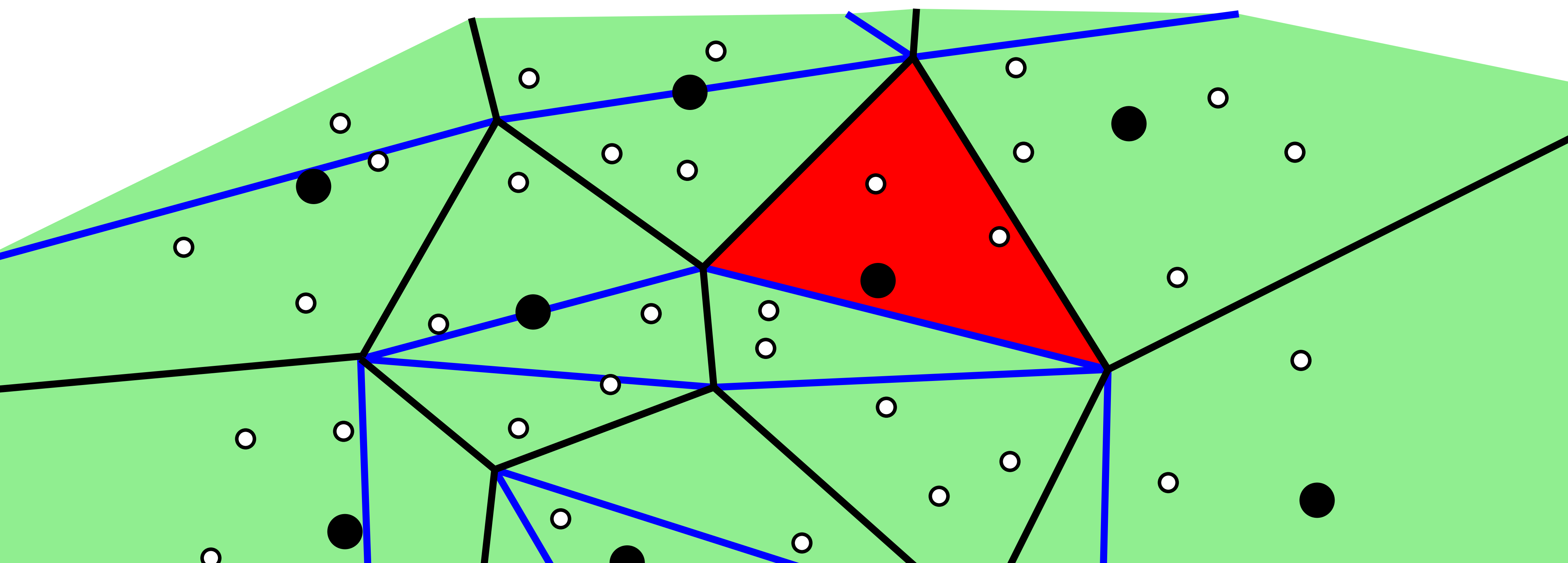


Phase II: Compute $\mathrm{VD}(P)$

Assumption: for each Δ: $B_\Delta = O(n/s)$

Run in parallel for each Δ the algorithm by Asano et al.

- $O(n^2/s^2)$ time and $O(1)$ space
- requires $O(n/s)$ scans of B_Δ

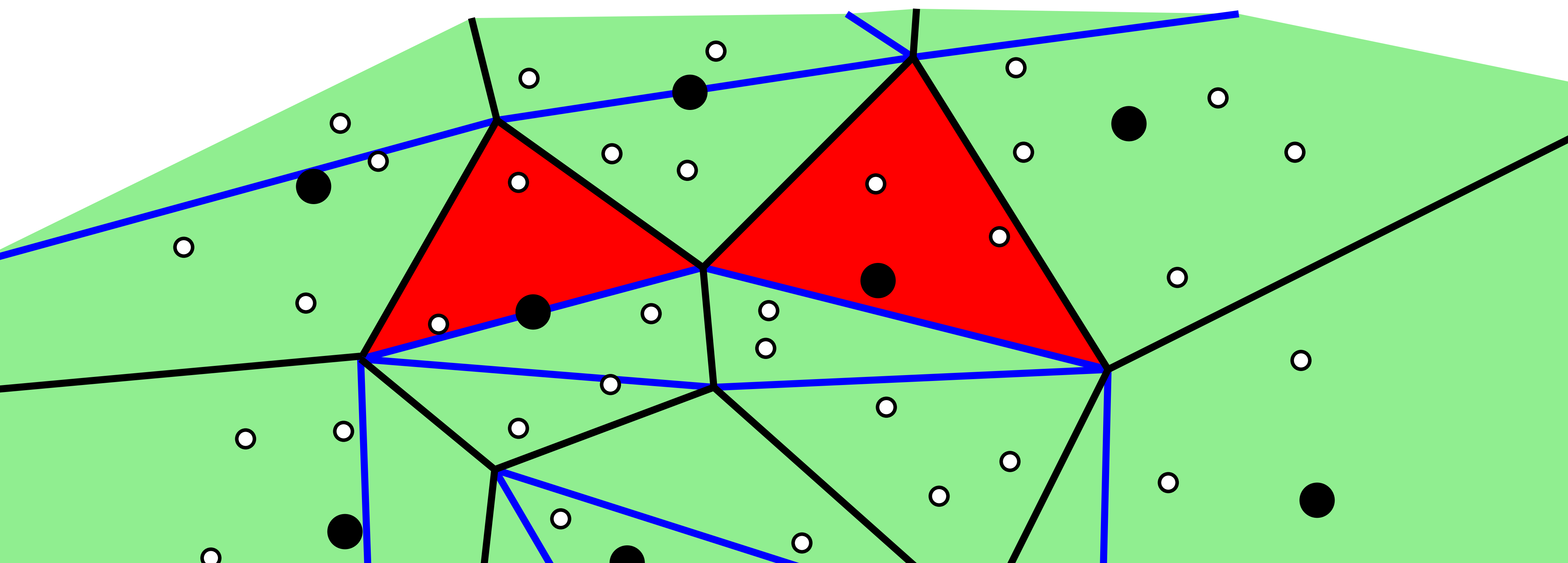


Phase II: Compute $\mathrm{VD}(P)$

Assumption: for each Δ: $B_\Delta = O(n/s)$

Run in parallel for each Δ the algorithm by Asano et al.

- $O(n^2/s^2)$ time and $O(1)$ space
- requires $O(n/s)$ scans of B_Δ



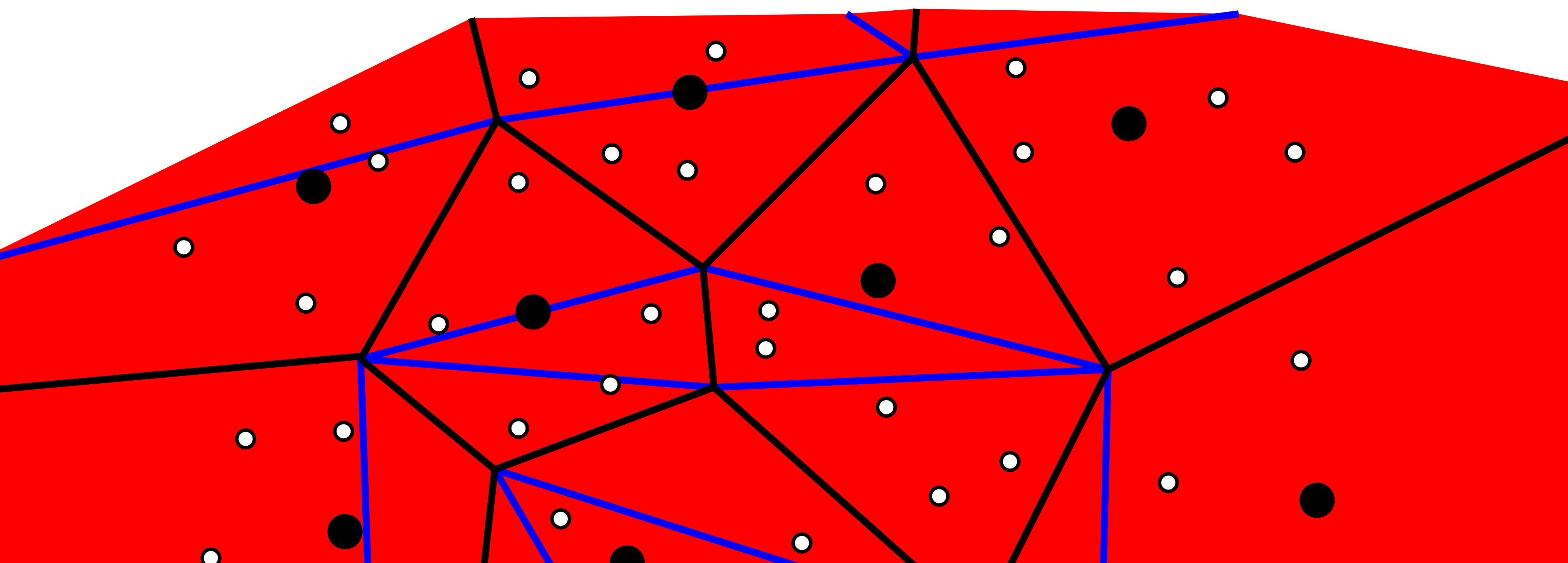
Phase II: Compute $\mathrm{VD}(P)$

Assumption: for each Δ: $B_\Delta = O(n/s)$

Run in parallel for each Δ the algorithm by Asano et al.

- $O(n^2/s^2)$ time and $O(1)$ space
- requires $O(n/s)$ scans of B_Δ

All parallel instances want to read: do full scan



Phase II: Compute $\mathrm{VD}(P)$

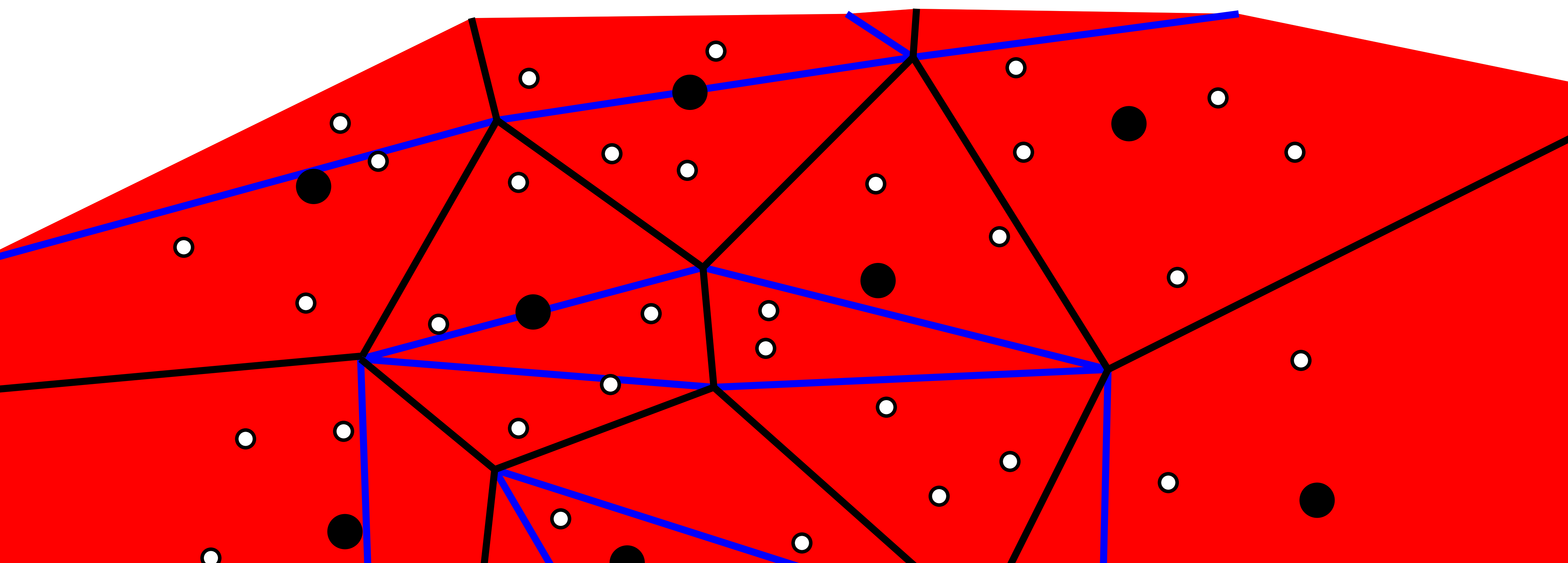
Assumption: for each Δ: $B_\Delta = O(n/s)$

Run in parallel for each Δ the algorithm by Asano et al.

- $O(n^2/s^2)$ time and $O(1)$ space
- requires $O(n/s)$ scans of B_Δ

All parallel instances want to read: do full scan

- For each input point p:
  - determine all conflict sets $B_\Delta : p \in B_\Delta$ by point loc.



Phase II: Compute $\mathrm{VD}(P)$

Assumption: for each Δ: $B_\Delta = O(n/s)$

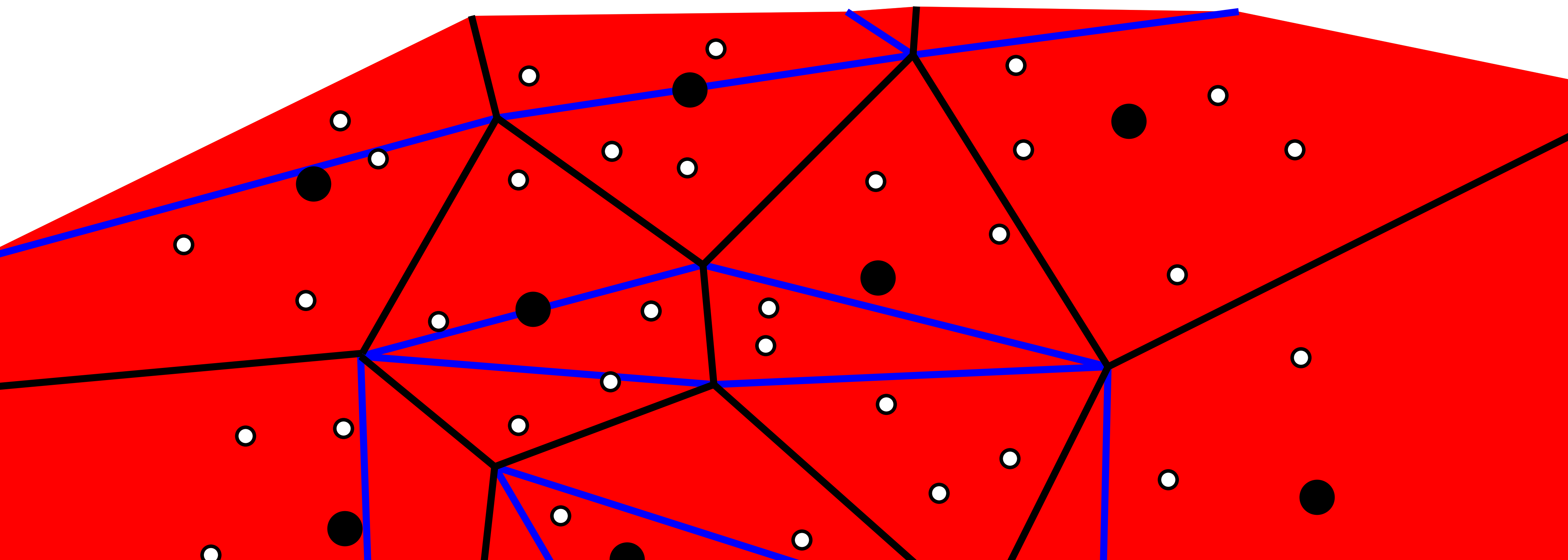
Run in parallel for each Δ the algorithm by Asano et al.

- $O(n^2/s^2)$ time and $O(1)$ space
- requires $O(n/s)$ scans of B_Δ

All parallel instances want to read: do full scan

$\boxed{O(n \log s) \text{ time}}$

- For each input point p:
  - determine all conflict sets $B_\Delta : p \in B_\Delta$ by point loc.



Phase II: Compute $\mathrm{VD}(P)$

Assumption: for each Δ: $B_\Delta = O(n/s)$

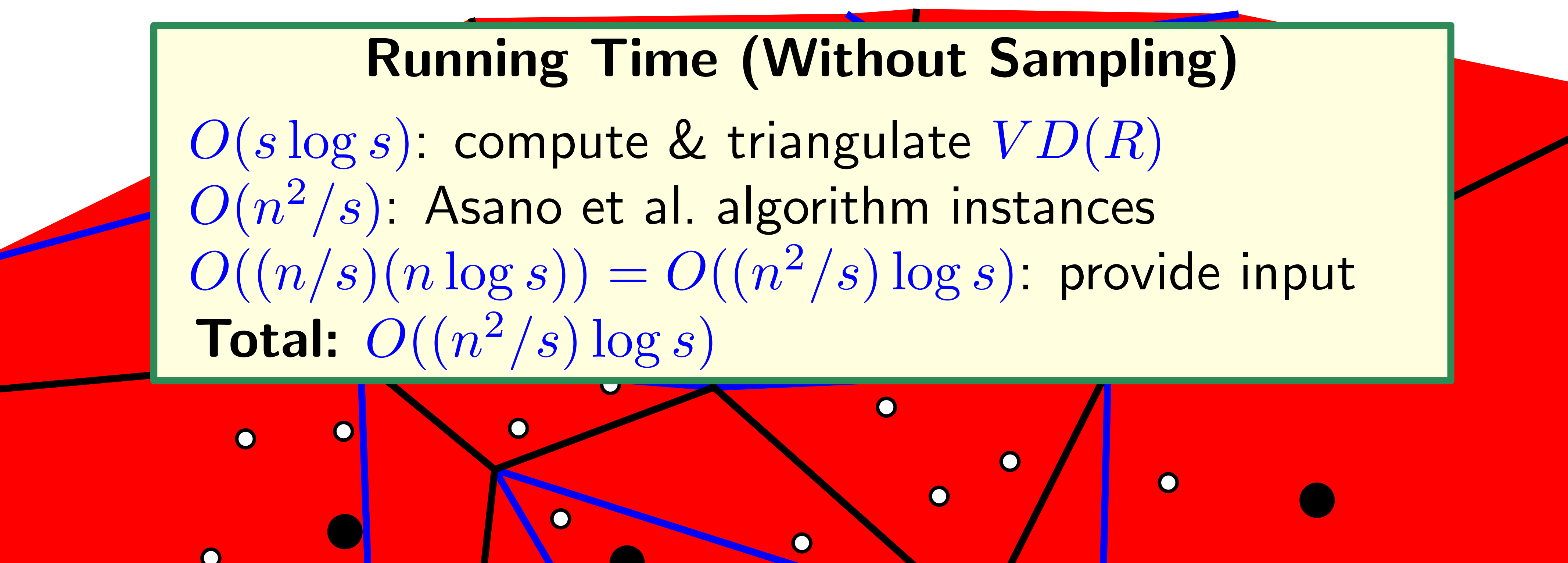
Run in parallel for each Δ the algorithm by Asano et al.

- $O(n^2/s^2)$ time and $O(1)$ space
- requires $O(n/s)$ scans of B_Δ

All parallel instances want to read: do full scan

$O(n \log s)$ time

- For each input point p:
  - determine all conflict sets $B_\Delta : p \in B_\Delta$ by point loc.



Running Time (Without Sampling)

$O(s \log s)$: compute & triangulate $VD(R)$

$O(n^2/s)$: Asano et al. algorithm instances

$O((n/s)(n \log s)) = O((n^2/s) \log s)$: provide input

Total: $O((n^2/s) \log s)$

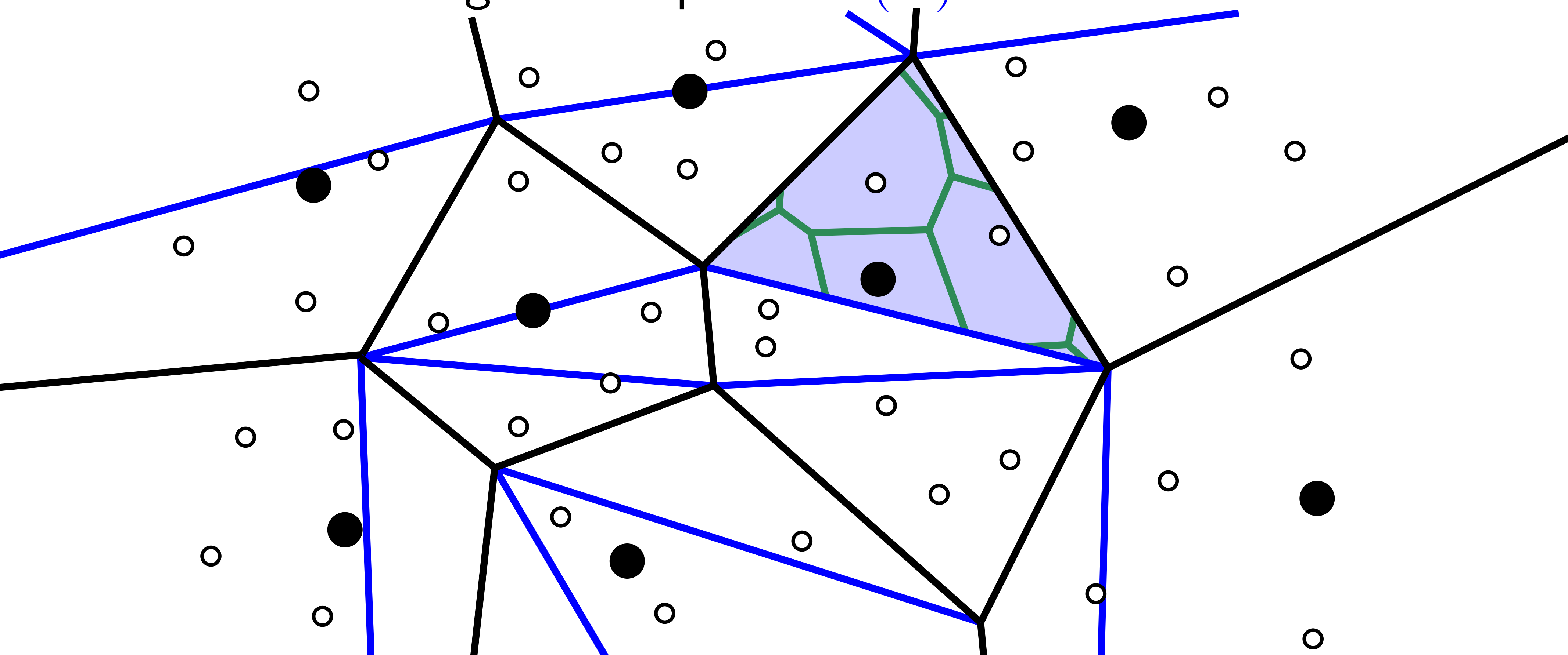
Computing $\mathrm{VD}(P)$ in $O(s)$ space — Overview

Phase I: Sampling

- Take random sample $R \subset P$ of size $O(s)$

Phase II: Compute $\mathrm{VD}(P)$

- Compute $\mathrm{VD}(R)$
- Triangulate cells of $\mathrm{VD}(R)$
- For each triangle Δ: report $\mathrm{VD}(P) \cap \Delta$



Phase I: Sampling — Overview

Want: for each Δ: $B_\Delta = O(n/s)$

Phase I: Sampling — Overview

Want: for each Δ: $B_\Delta = O(n/s)$

- Take random sample $R \subseteq P$ of size s $\boxed{O(n) \text{ time}}$

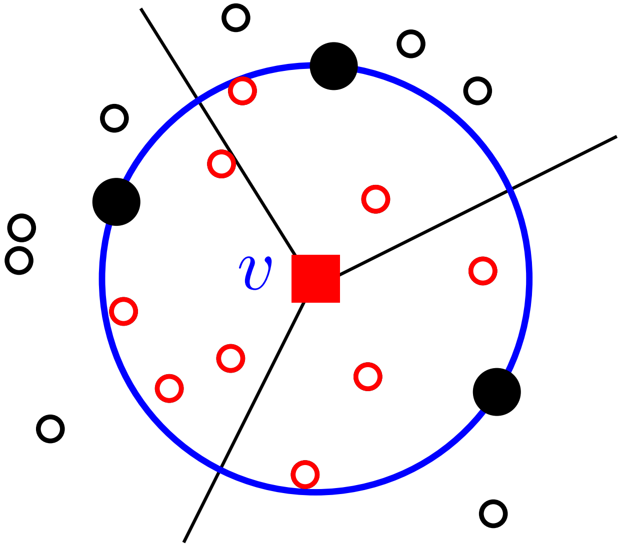
Phase I: Sampling — Overview

Want: for each Δ: $B_\Delta = O(n/s)$

- Take random sample $R \subseteq P$ of size s $\boxed{O(n) \text{ time}}$

- For each vertex $v \in \mathrm{VD}(R)$ compute its *excess* t_v

$$t_v = \frac{|B_v|}{n/s}$$

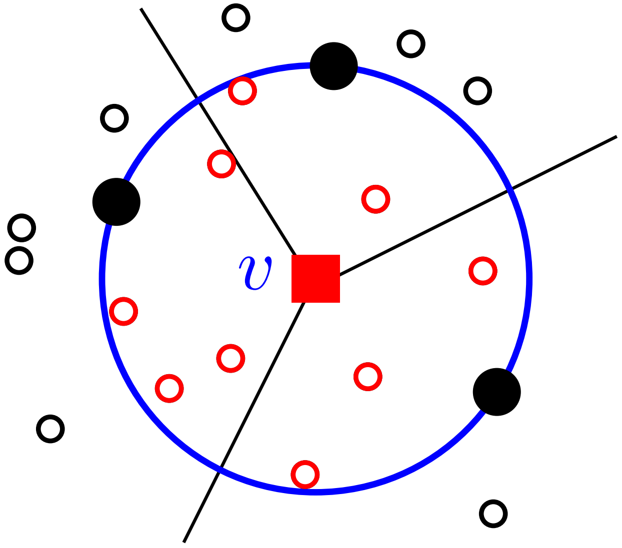


Phase I: Sampling — Overview

Want: for each Δ: $B_\Delta = O(n/s)$

- Take random sample $R \subseteq P$ of size s $\boxed{O(n) \text{ time}}$
- For each vertex $v \in \mathrm{VD}(R)$ compute its *excess* t_v

$$t_v = \frac{|B_v|}{n/s}$$



- counter per vertex $v \in \mathrm{VD}(R)$
- For each input point p:
  - determine all $v \in \mathrm{VD}(R) : p \in B_v$ by point local. and increase counters

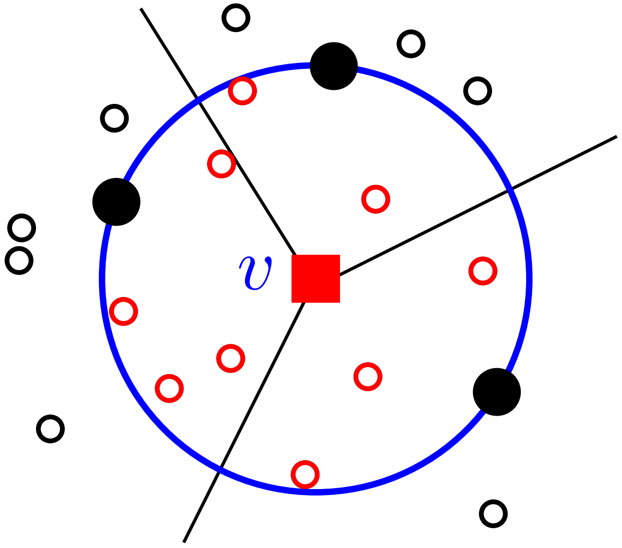
Phase I: Sampling — Overview

Want: for each Δ: $B_\Delta = O(n/s)$

- Take random sample $R \subseteq P$ of size s $\boxed{O(n) \text{ time}}$
- For each vertex $v \in \mathrm{VD}(R)$ compute its *excess* t_v

$$t_v = \frac{|B_v|}{n/s}$$

$\boxed{O(n \log s) \text{ time}}$



- counter per vertex $v \in \mathrm{VD}(R)$
- For each input point p:
  - determine all $v \in \mathrm{VD}(R) : p \in B_v$ by point local. and increase counters

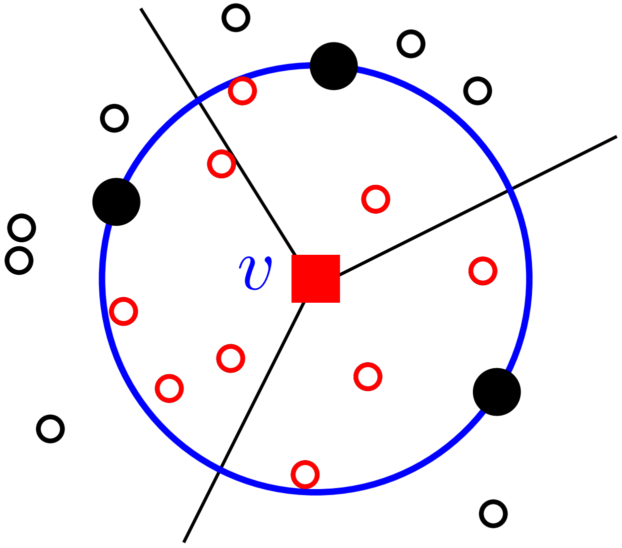
Phase I: Sampling — Overview

Want: for each Δ: $B_\Delta = O(n/s)$

- Take random sample $R \subseteq P$ of size s $\boxed{O(n) \text{ time}}$

- For each vertex $v \in \mathrm{VD}(R)$ compute its *excess* t_v

$$t_v = \frac{|B_v|}{n/s} \qquad \boxed{O(n \log s) \text{ time}}$$

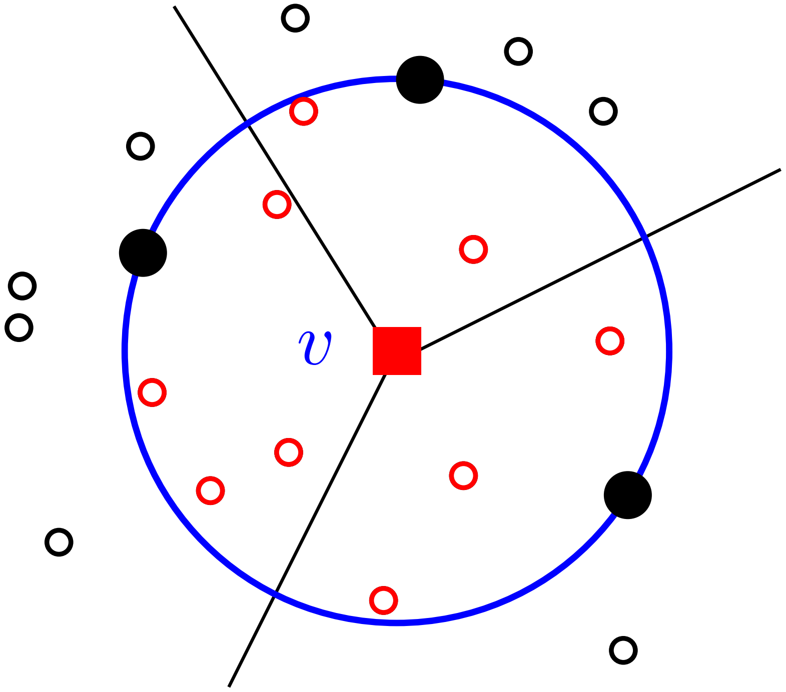


- counter per vertex $v \in \mathrm{VD}(R)$
- For each input point p:
  - determine all $v \in \mathrm{VD}(R) : p \in B_v$ by point local. and increase counters

- For vertices v with large excess:
  - sample additional points from B_v

Sampling from Conflict Sets

- For vertices v with large excess:
  $t_v \geq 2$
  - sample additional $O(t_v \log t_v)$ points R_v from B_v

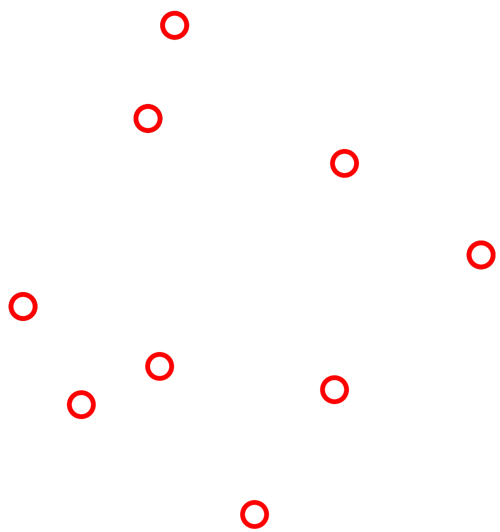


Sampling from Conflict Sets

- For vertices v with large excess:

  $t_v \geq 2$

  - sample additional $O(t_v \log t_v)$ points R_v from B_v

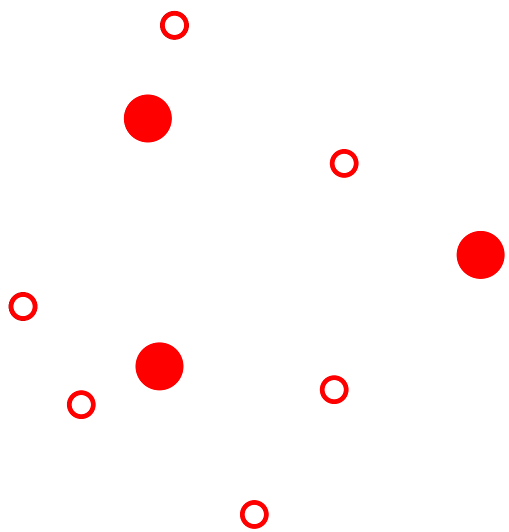


Sampling from Conflict Sets

- For vertices v with large excess:

  $t_v \geq 2$

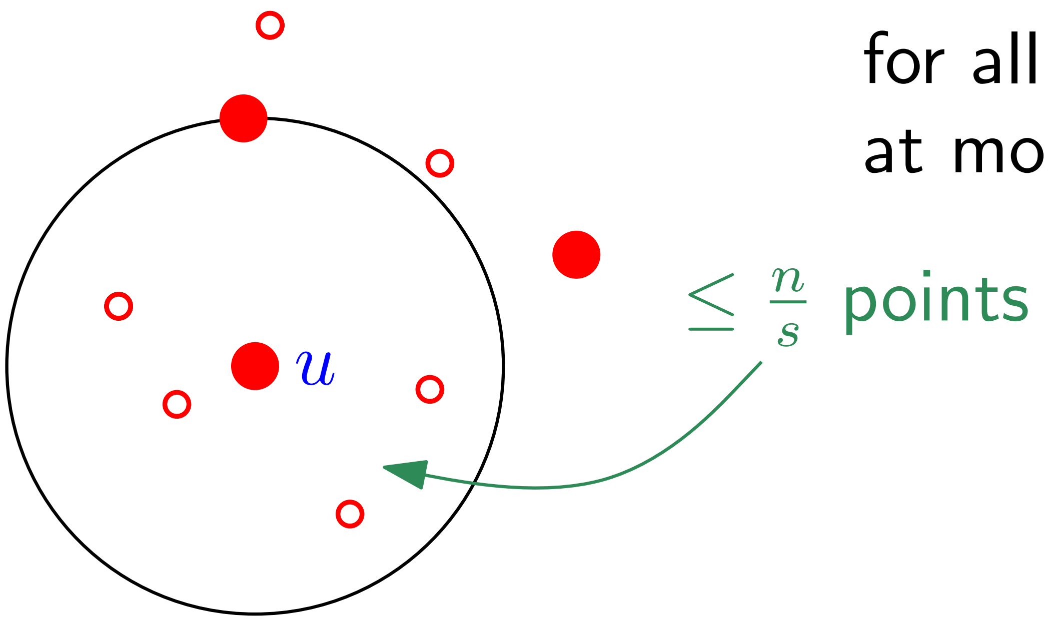
  - sample additional $O(t_v \log t_v)$ points R_v from B_v



Sampling from Conflict Sets

- For vertices v with large excess: $\rightarrow t_v \geq 2$
  - sample additional $O(t_v \log t_v)$ points R_v from B_v

Call a sample R_v *good* iff for all $u \in R_v$: B_u contains at most $\frac{n}{s}$ points from B_v



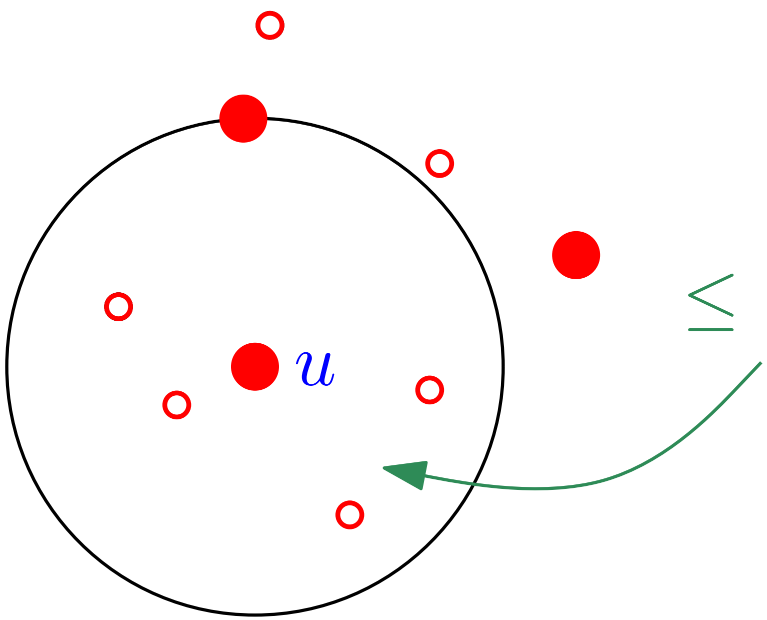
Sampling from Conflict Sets

- For vertices v with large excess:

  $t_v \geq 2$

  - sample additional $O(t_v \log t_v)$ points R_v from B_v

Call a sample R_v *good* iff for all $u \in R_v$: B_u contains at most $\frac{n}{s}$ points from B_v



Lemma. Let $R' = R \cup \{R_v \mid t_v \geq 2,\ R_v \text{ is a good sample}\}$. Then, for all triangles Δ in the triangulation of $\mathrm{VD}(R')$ we have $B_\Delta = O(n/s)$.

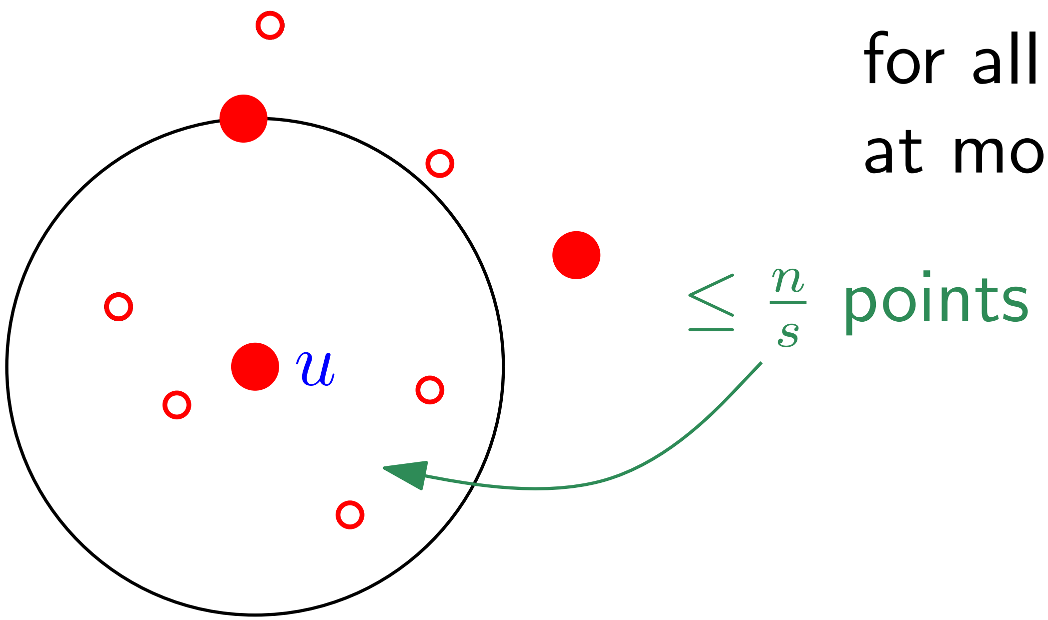
Sampling from Conflict Sets

- For vertices v with large excess:

  $t_v \geq 2$

  - sample additional $O(t_v \log t_v)$ points R_v from B_v

Call a sample R_v *good* iff for all $u \in R_v$: B_u contains at most $\frac{n}{s}$ points from B_v



Lemma. Let $R' = R \cup \{R_v \mid t_v \geq 2,\ R_v \text{ is a good sample}\}$. Then, for all triangles Δ in the triangulation of $\mathrm{VD}(R')$ we have $B_\Delta = O(n/s)$.

Lemma. $\mathbf{E}[\sum_{v \in \mathrm{VD}(R)} |R_v|] = O(s)$

Sampling Efficiently from Conflict Sets

- For vertices v with large excess:

  $\rightarrow t_v \geq 2$

  - sample additional $O(t_v \log t_v)$ points R_v from B_v

Call a sample R_v *good* iff for all $u \in R_v : \ |B_u \cap B_v| \leq \frac{n}{s}$

Sampling Efficiently from Conflict Sets

- For vertices v with large excess:

  $\rightarrow t_v \geq 2$

  - sample additional $O(t_v \log t_v)$ points R_v from B_v

Call a sample R_v *good* iff for all $u \in R_v$: $|B_u \cap B_v| \leq \frac{n}{s}$

Lemma. A sample is good with probability at least $3/4$.

Sampling Efficiently from Conflict Sets

- For vertices v with large excess:

  $\hookrightarrow t_v \geq 2$

  - sample additional $O(t_v \log t_v)$ points R_v from B_v

Call a sample R_v *good* iff for all $u \in R_v : \; |B_u \cap B_v| \leq \frac{n}{s}$

Lemma. A sample is good with probability at least $3/4$.

Sample in Rounds

1. **Round:** take one sample per vertex $v : t_v \geq 2$

   $\Pr[\text{half of the samples are good}] \geq 1/2$

Sampling Efficiently from Conflict Sets

- For vertices v with large excess:

  $\hookrightarrow t_v \geq 2$

  - sample additional $O(t_v \log t_v)$ points R_v from B_v

Call a sample R_v *good* iff for all $u \in R_v : \; |B_u \cap B_v| \leq \frac{n}{s}$

Lemma. A sample is good with probability at least $3/4$.

Sample in Rounds

1. Round: take one sample per vertex $v : t_v \geq 2$

$\Pr[\text{half of the samples are good}] \geq 1/2$

2. Round: take two samples per remaining vertex

$\vdots$

Sampling Efficiently from Conflict Sets

- For vertices v with large excess:

  $\longrightarrow t_v \geq 2$

  - sample additional $O(t_v \log t_v)$ points R_v from B_v

Call a sample R_v *good* iff for all $u \in R_v : \ |B_u \cap B_v| \leq \frac{n}{s}$

Lemma. A sample is good with probability at least $3/4$.

Sample in Rounds

1. **Round:** take one sample per vertex $v : t_v \geq 2$

   $\Pr[\text{half of the samples are good}] \geq 1/2$

2. **Round:** take two samples per remaining vertex

Expected #Rounds: $O(\log^* s)$
One Round: Sampling & Checking Samples $O(n \log s)$
Total: $O(n \log s \log^* s)$

Putting it together

Phase I: Computing R': $O(n \log s \log^* s)$ expected time

Phase II: Computing $VD(P)$: $O((n^2/s) \log s)$ time

Putting it together

Phase I: Computing R': $O(n \log s \log^* s)$ expected time

Phase II: Computing $VD(P)$: $O((n^2/s) \log s)$ time

$$\Downarrow$$

Theorem. Reporting Voronoi diagrams of a set of n points in the plane can be done in $O((n^2/s) \log s + n \log s \log^* s)$ expected time using $O(s)$ space.

Putting it together

Phase I: Computing R': $O(n \log s \log^* s)$ expected time

Phase II: Computing $VD(P)$: $O((n^2/s) \log s)$ time

$$\Downarrow$$

Theorem. Reporting Voronoi diagrams of a set of n points in the plane can be done in $O((n^2/s) \log s + n \log s \log^* s)$ expected time using $O(s)$ space.

Open Problem: Can we do the same in worst-case time?