Time-space Trade-offs for 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

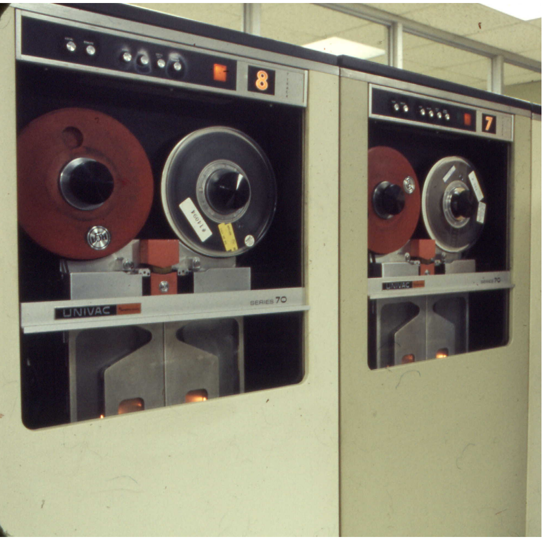
Marcel Roeloffzen
Tohoku University.
Tokyo, Japan

Paul Seiferth
Institut für Informatik,
Freie Universität
Berlin, Germany

Yannik Stein
Institut für Informatik,
Freie Universität
Berlin, Germany

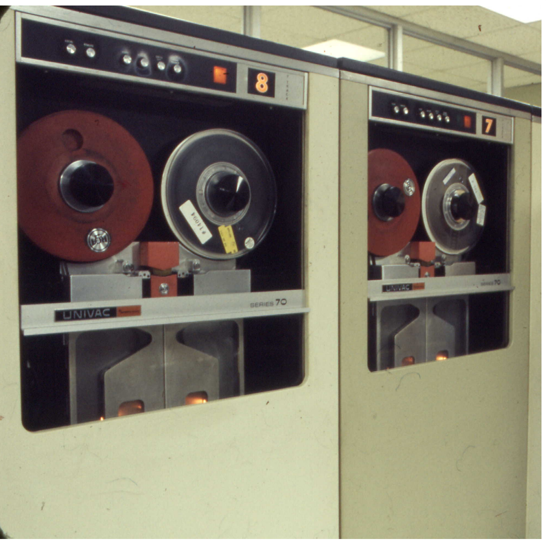
Limited Memory

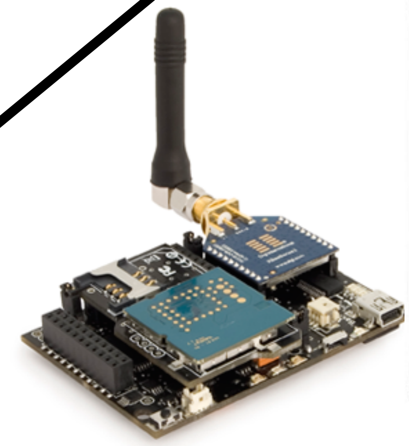
Started in the 70's

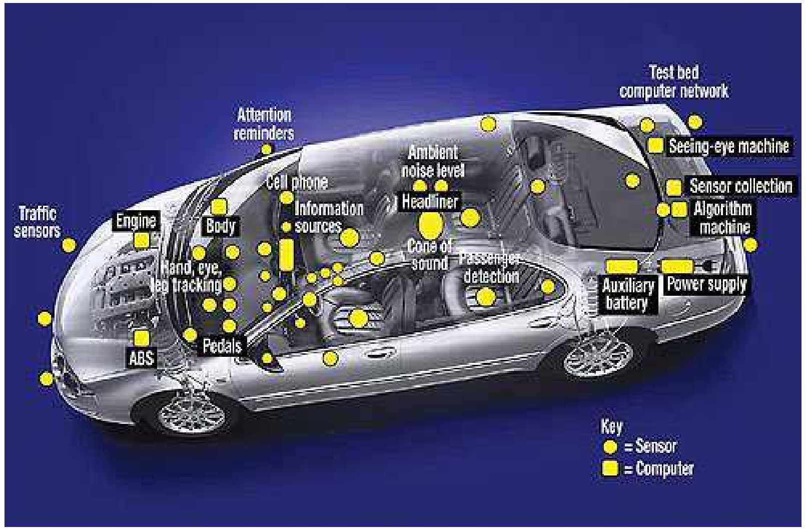


Limited Memory

Started in the 70's



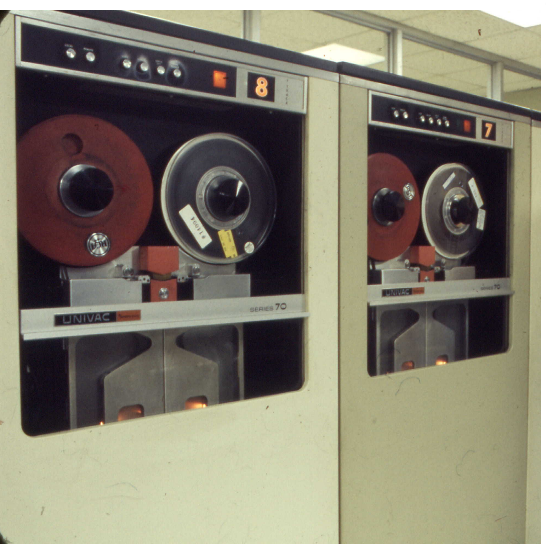




Increased interest recently

Limited Memory

Started in the 70's

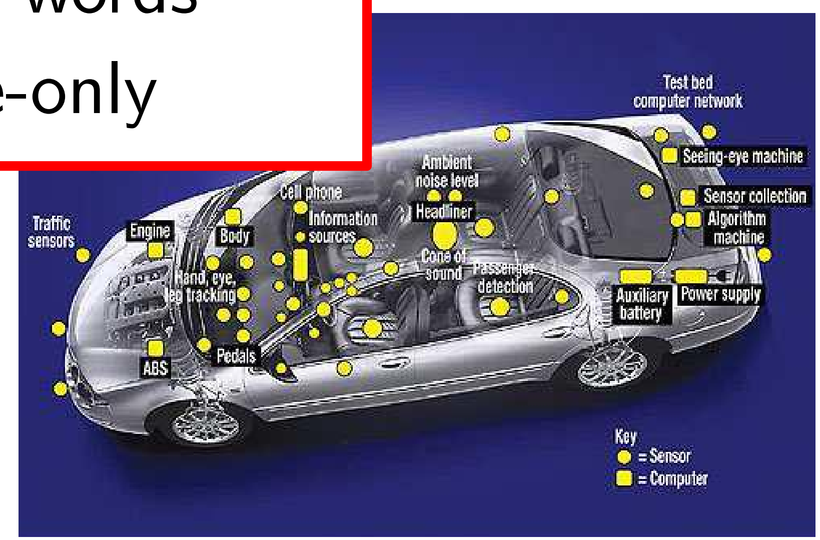


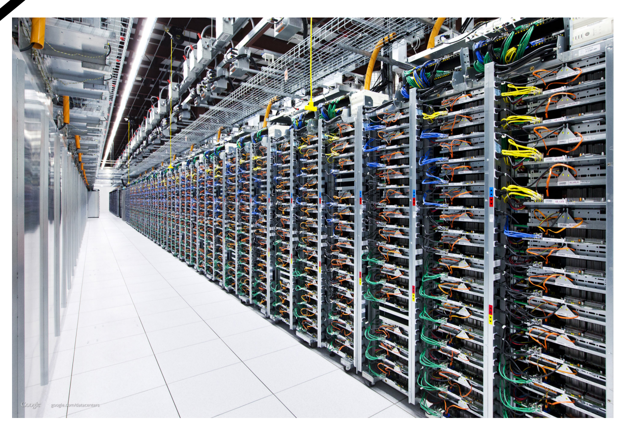
Model

Input: read-only, random-access

Memory: $O(s)$ words

Output: write-only





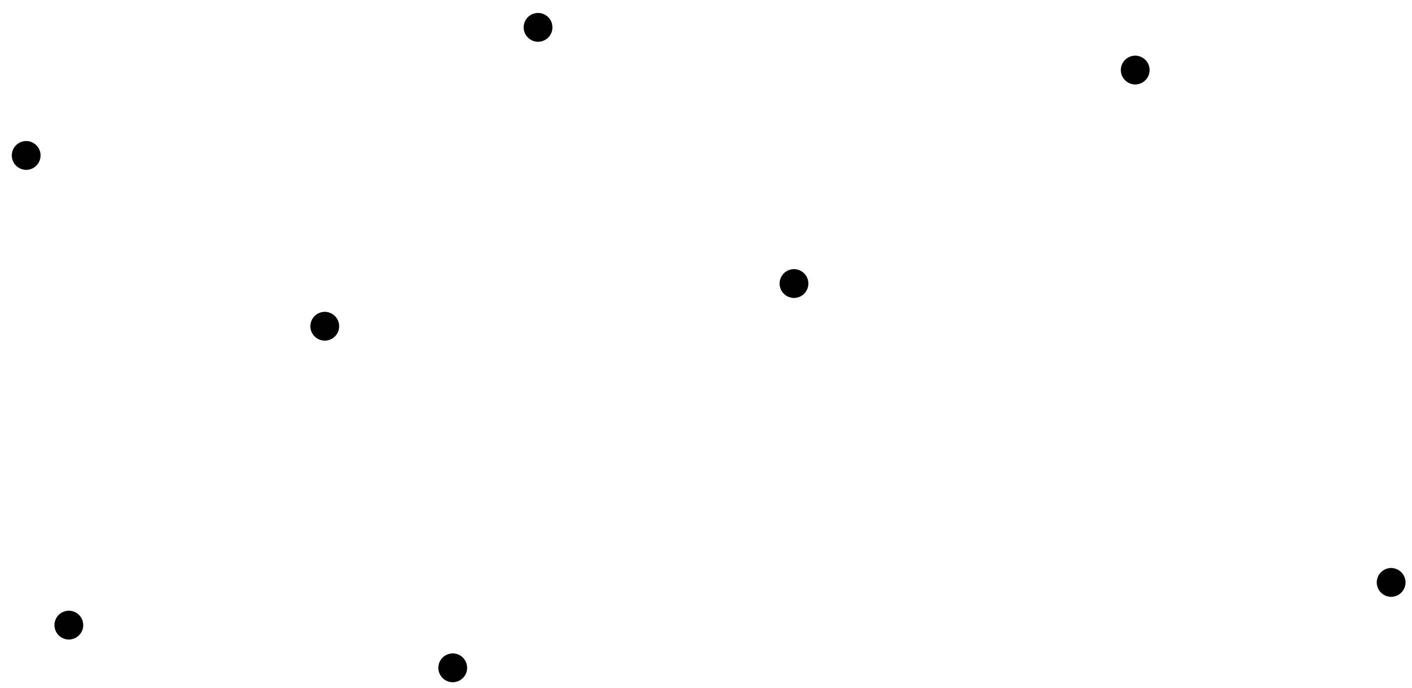
Increased interest recently

Voronoi Diagram

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

Output: Subdivision of $\mathbb{R}^2$, such that each region has a common nearest neighbor in P.

Output format: vertices of Voronoi diagram in arbitrary order

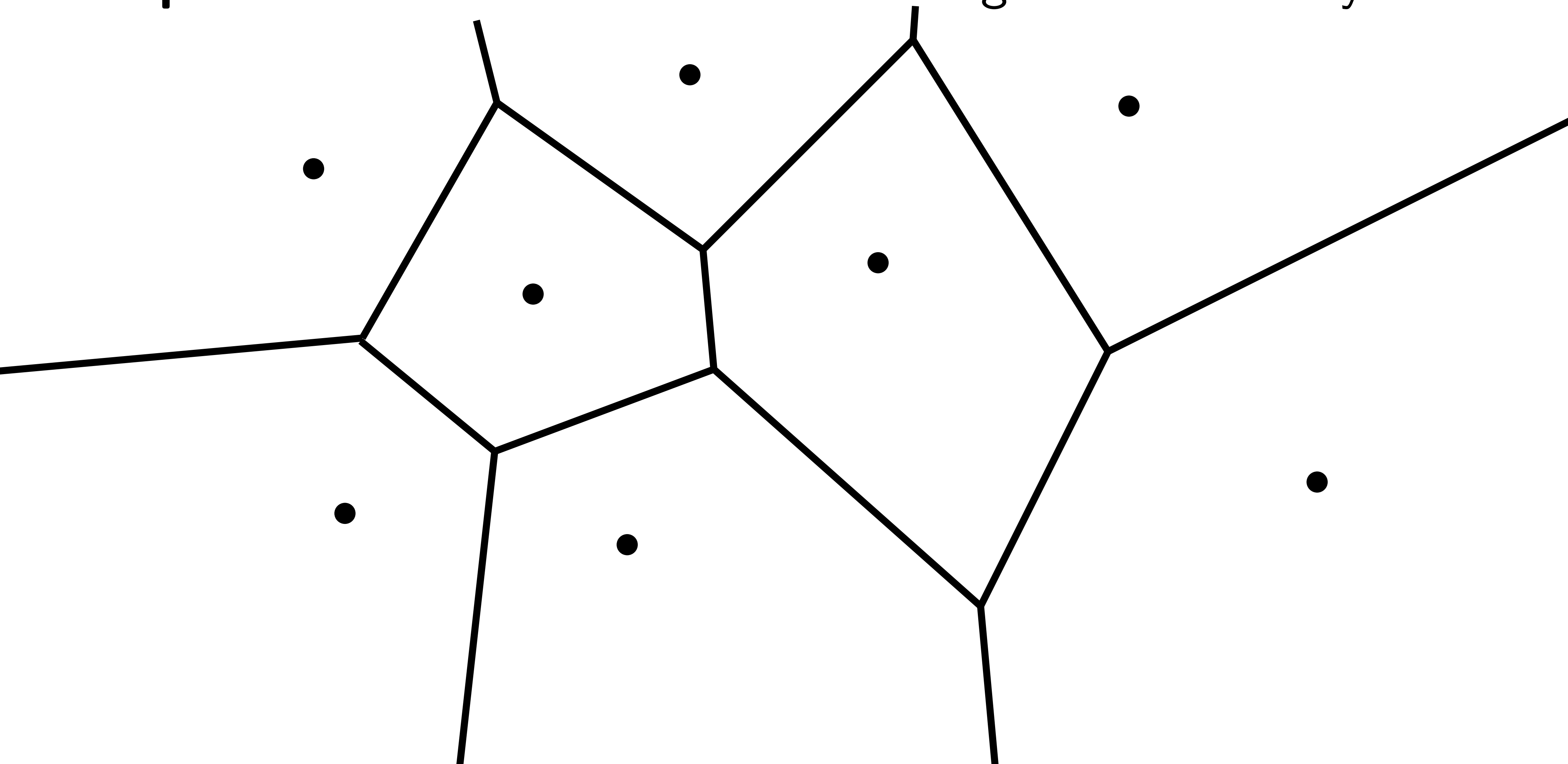


Voronoi Diagram

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

Output: Subdivision of $\mathbb{R}^2$, such that each region has a common nearest neighbor in P.

Output format: vertices of Voronoi diagram in arbitrary order

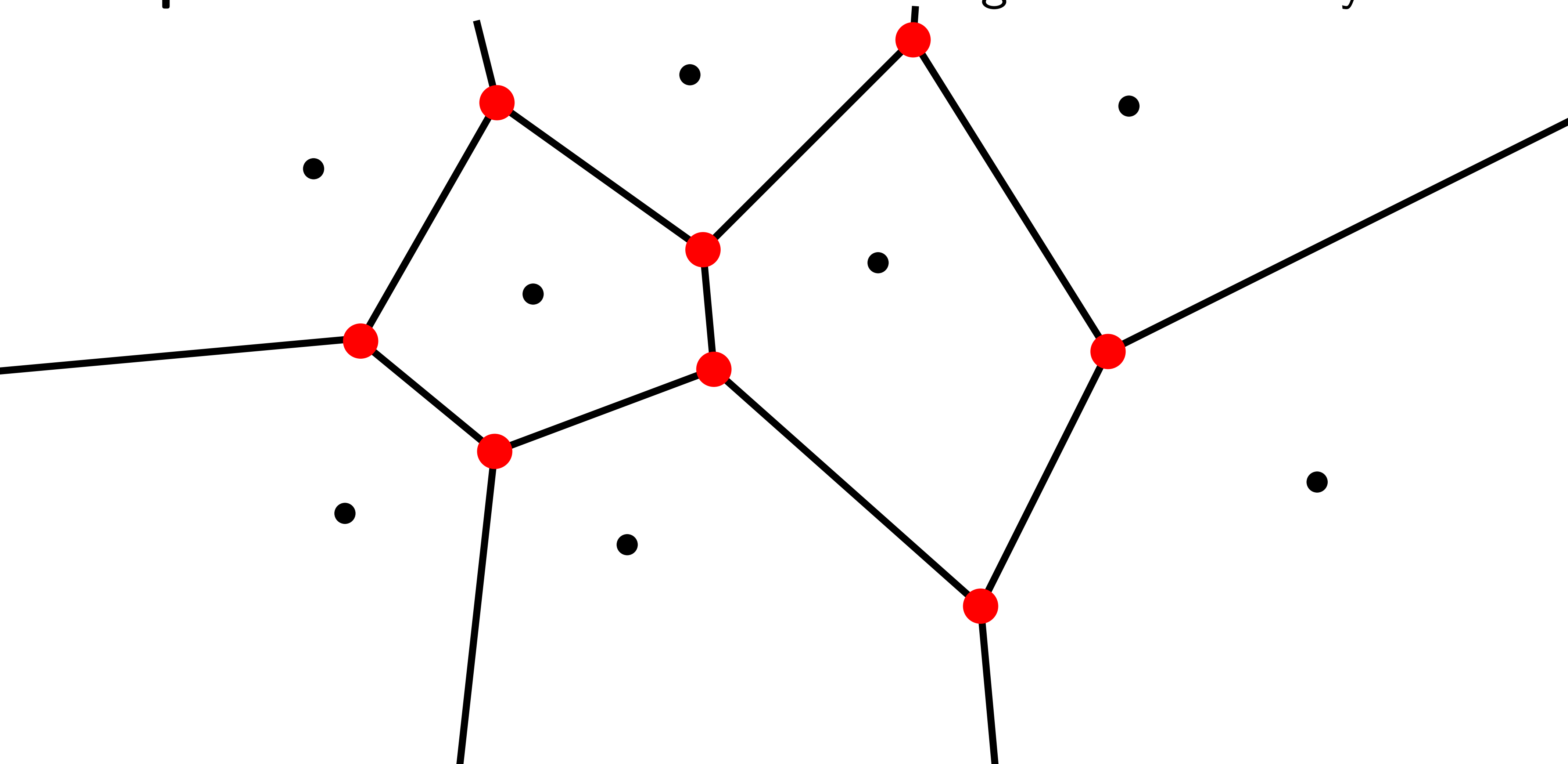


Voronoi Diagram

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

Output: Subdivision of $\mathbb{R}^2$, such that each region has a common nearest neighbor in P.

Output format: vertices of Voronoi diagram in arbitrary order

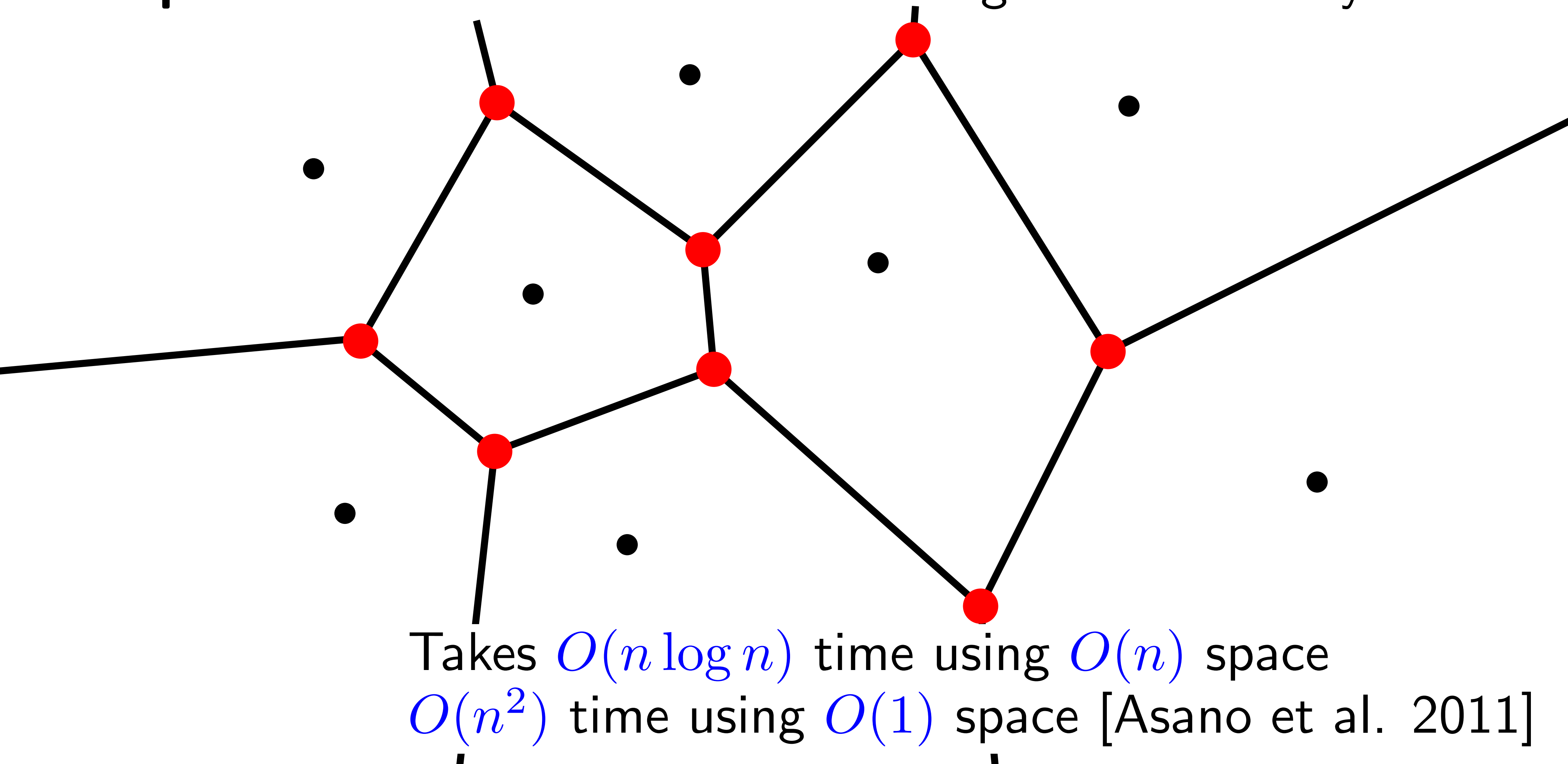


Voronoi Diagram

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

Output: Subdivision of $\mathbb{R}^2$, such that each region has a common nearest neighbor in P.

Output format: vertices of Voronoi diagram in arbitrary order

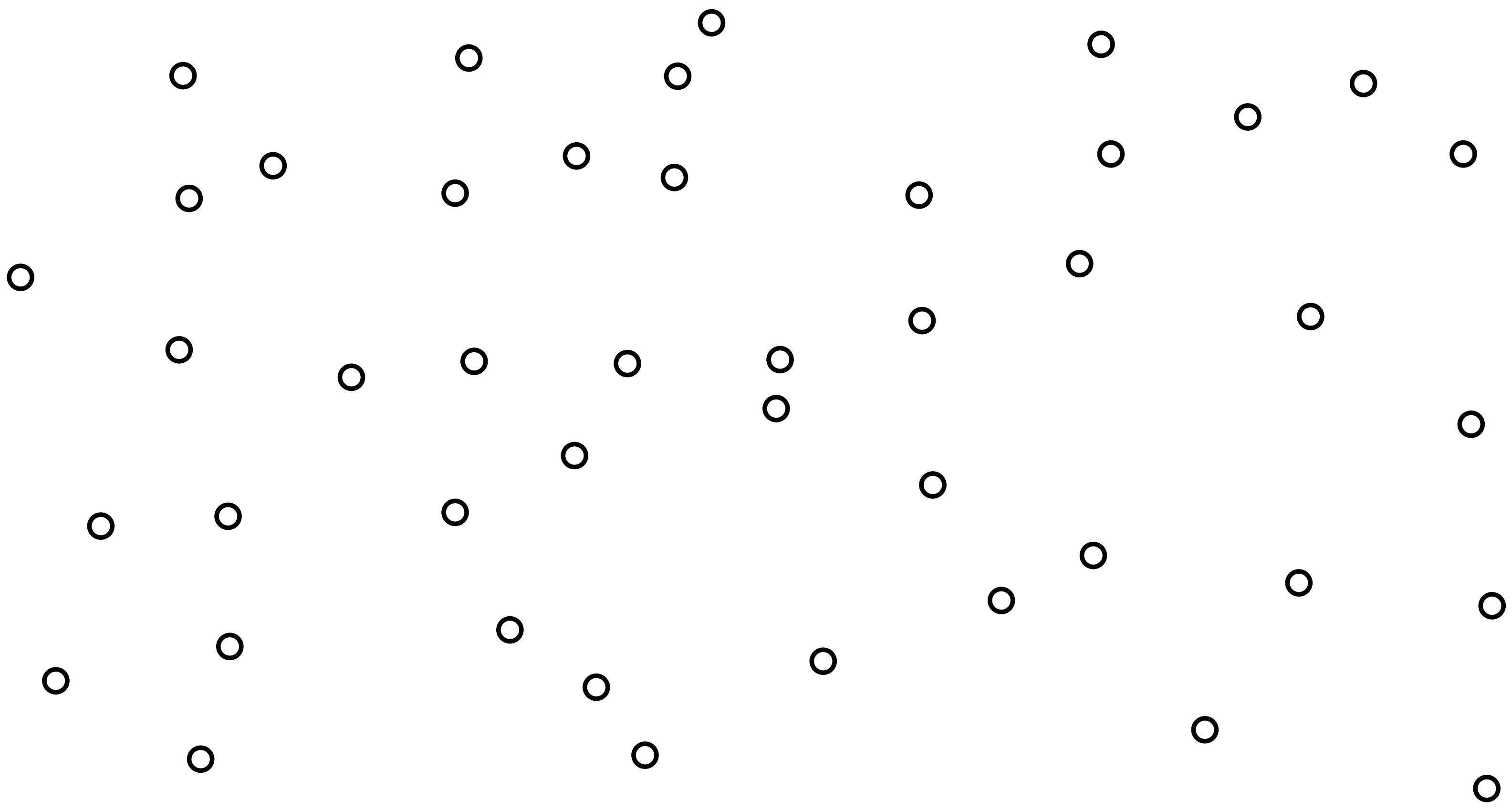


Takes $O(n \log n)$ time using $O(n)$ space

$O(n^2)$ time using $O(1)$ space [Asano et al. 2011]

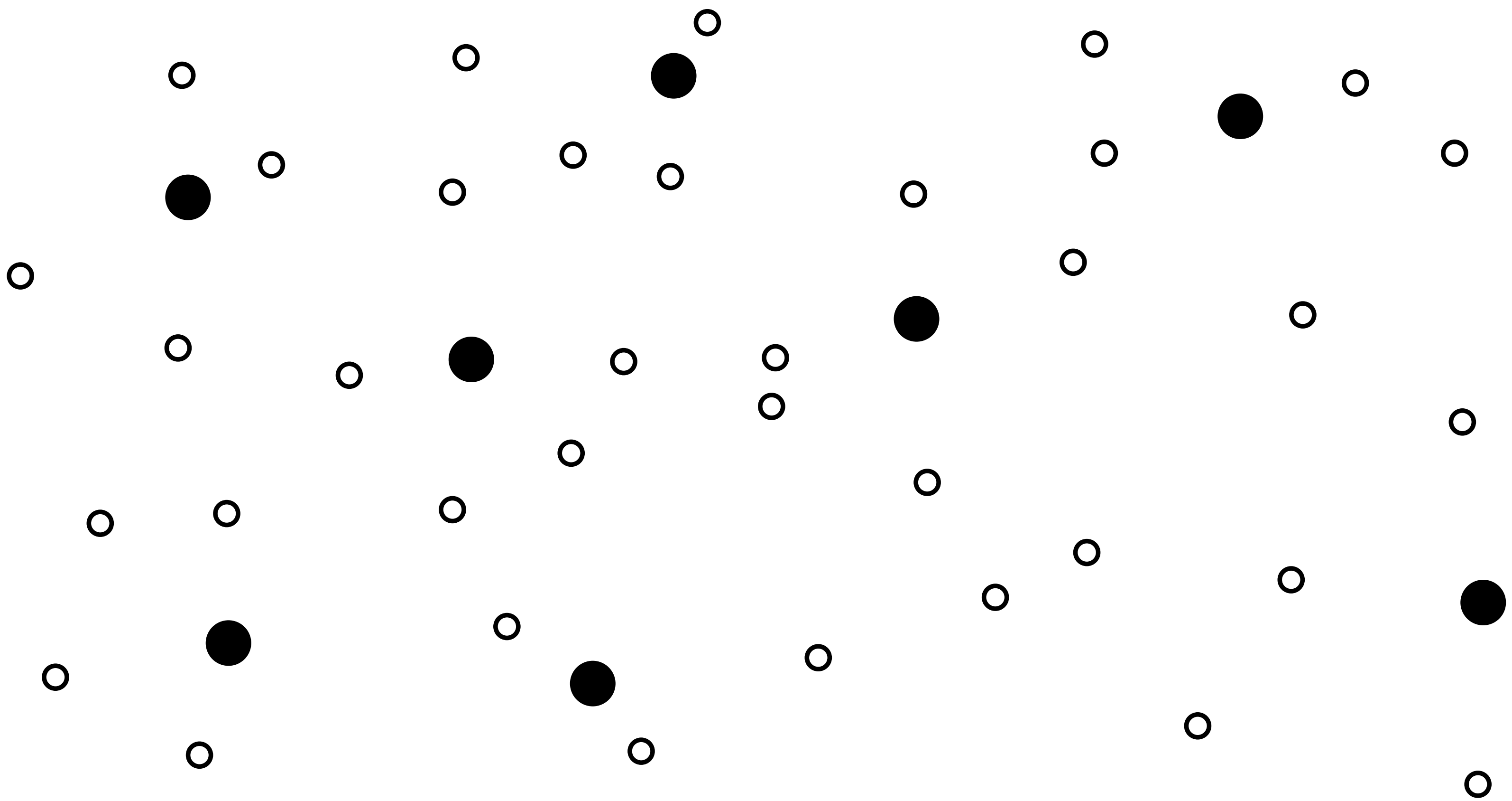
General approach

- Find $R \subset P$ of size $O(s)$
- Compute $VD(R)$
- Triangulate $VD(R)$
- For each triangle, report the vertices of $VD(P)$ inside it



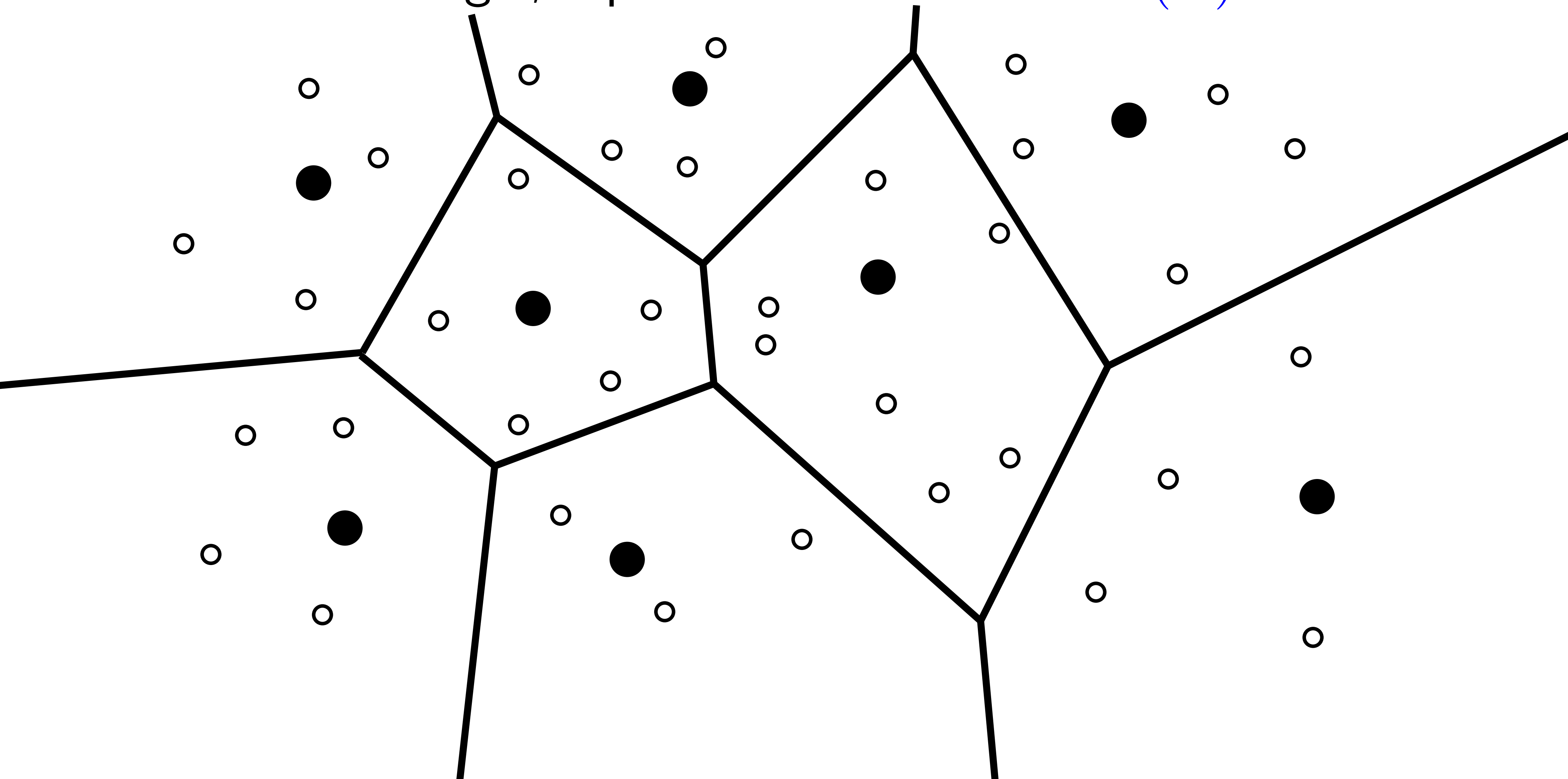
General approach

- Find $R \subset P$ of size $O(s)$
- Compute $VD(R)$
- Triangulate $VD(R)$
- For each triangle, report the vertices of $VD(P)$ inside it



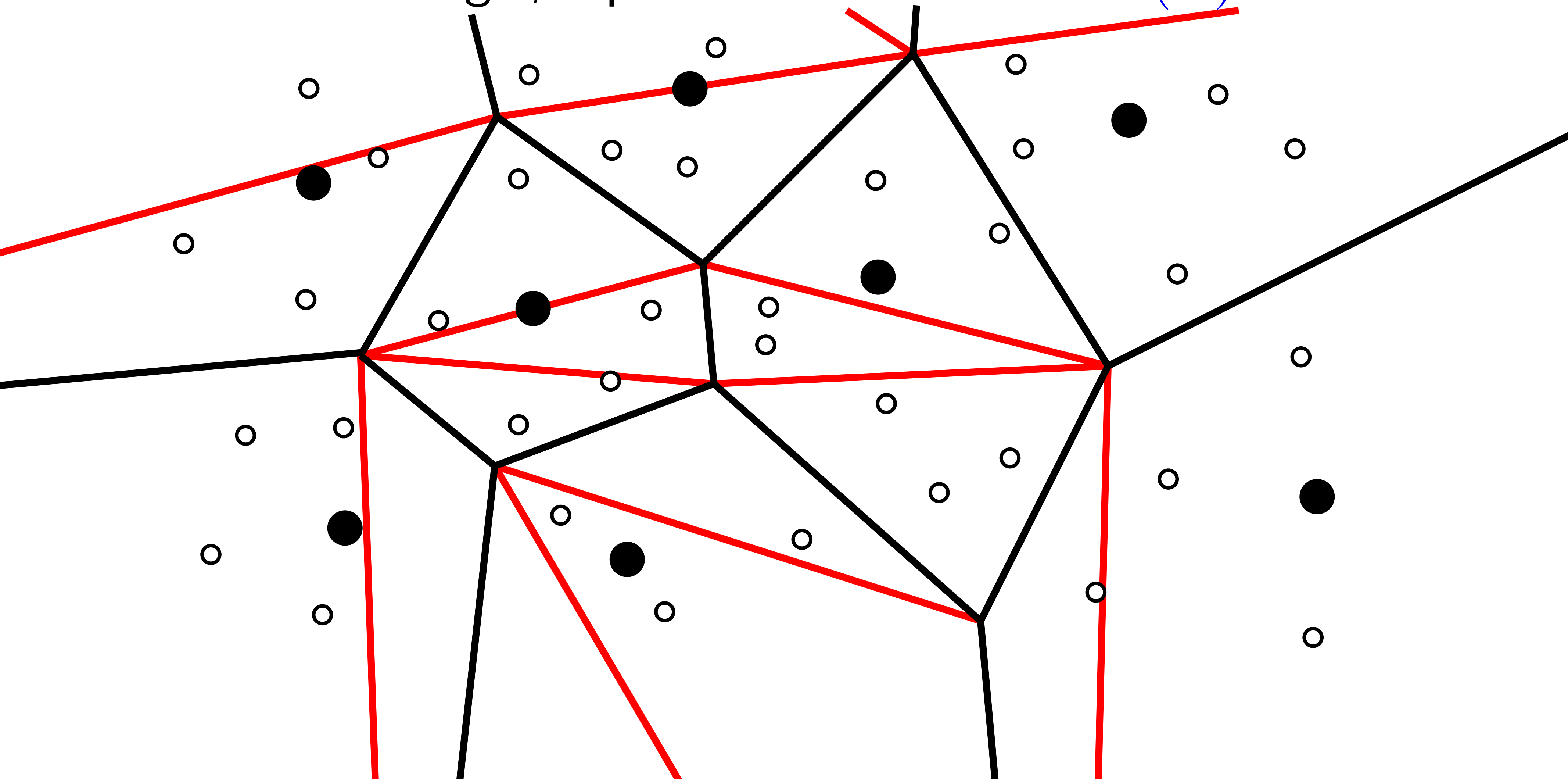
General approach

- Find $R \subset P$ of size $O(s)$
- Compute $VD(R)$
- Triangulate $VD(R)$
- For each triangle, report the vertices of $VD(P)$ inside it



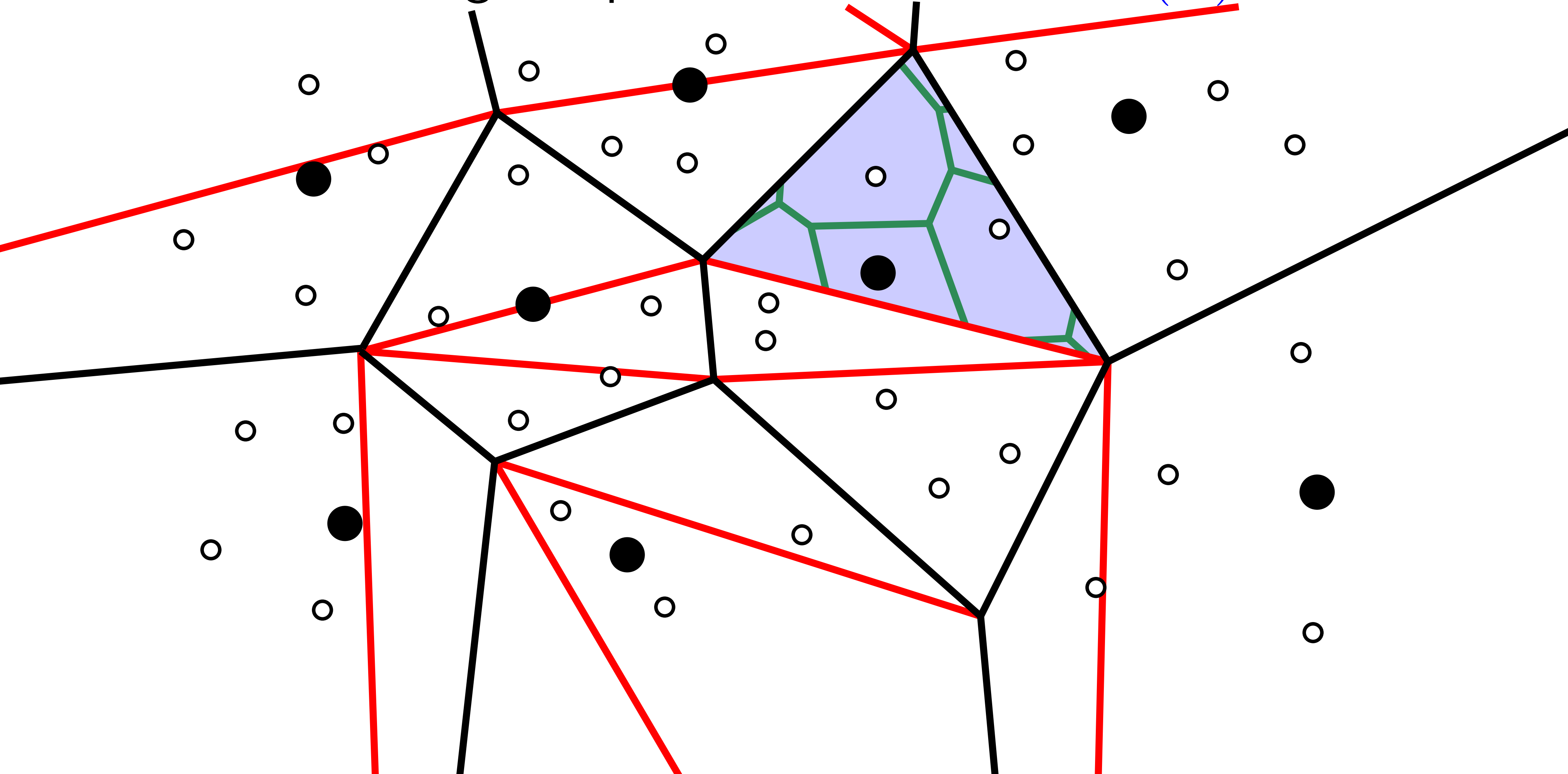
General approach

- Find $R \subset P$ of size $O(s)$
- Compute $VD(R)$
- Triangulate $VD(R)$
- For each triangle, report the vertices of $VD(P)$ inside it



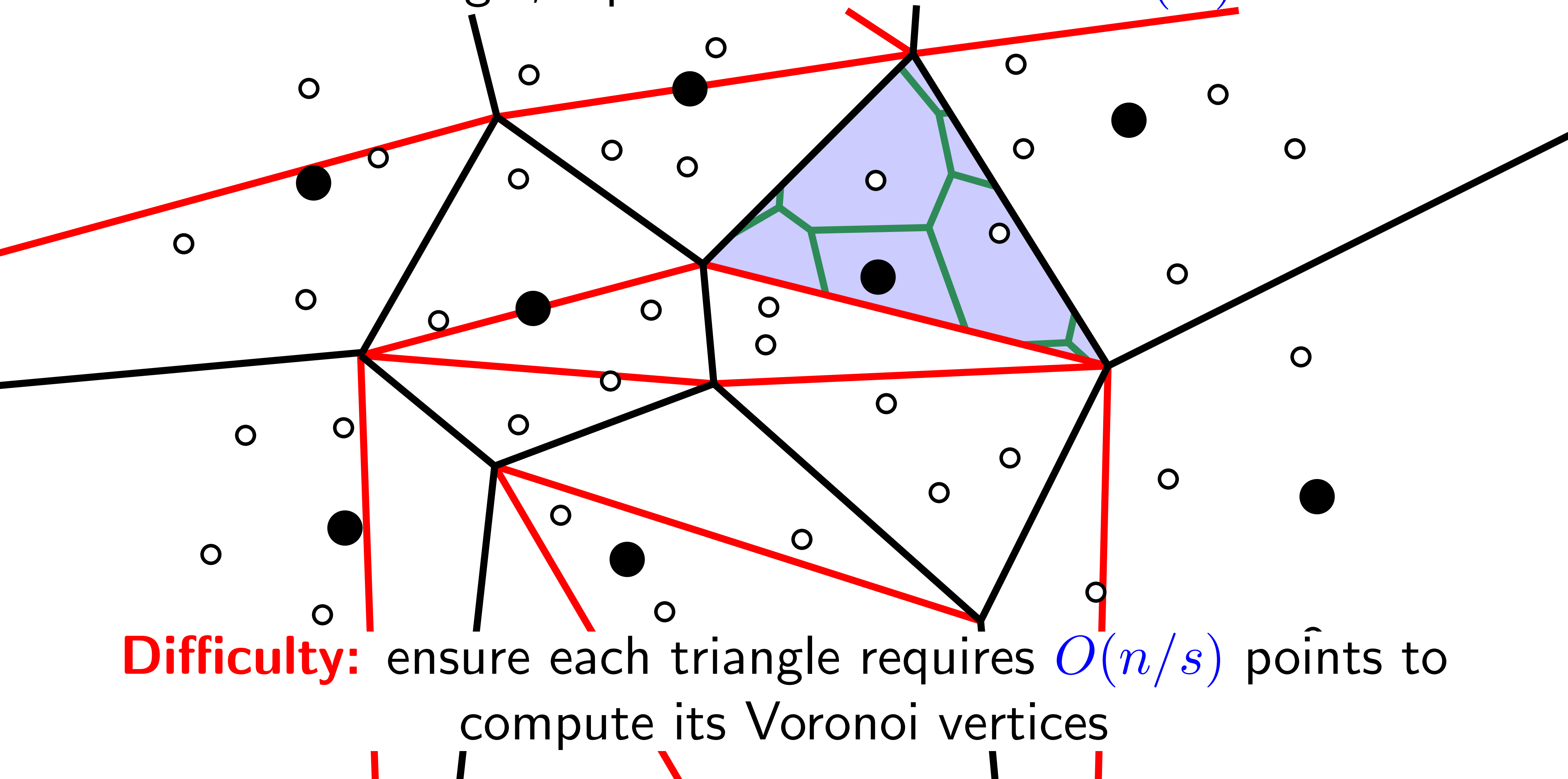
General approach

- Find $R \subset P$ of size $O(s)$
- Compute $VD(R)$
- Triangulate $VD(R)$
- For each triangle, report the vertices of $VD(P)$ inside it



General approach

- Find $R \subset P$ of size $O(s)$
- Compute $VD(R)$
- Triangulate $VD(R)$
- For each triangle, report the vertices of $VD(P)$ inside it



Difficulty: ensure each triangle requires $O(n/s)$ points to compute its Voronoi vertices

Computing the vertices

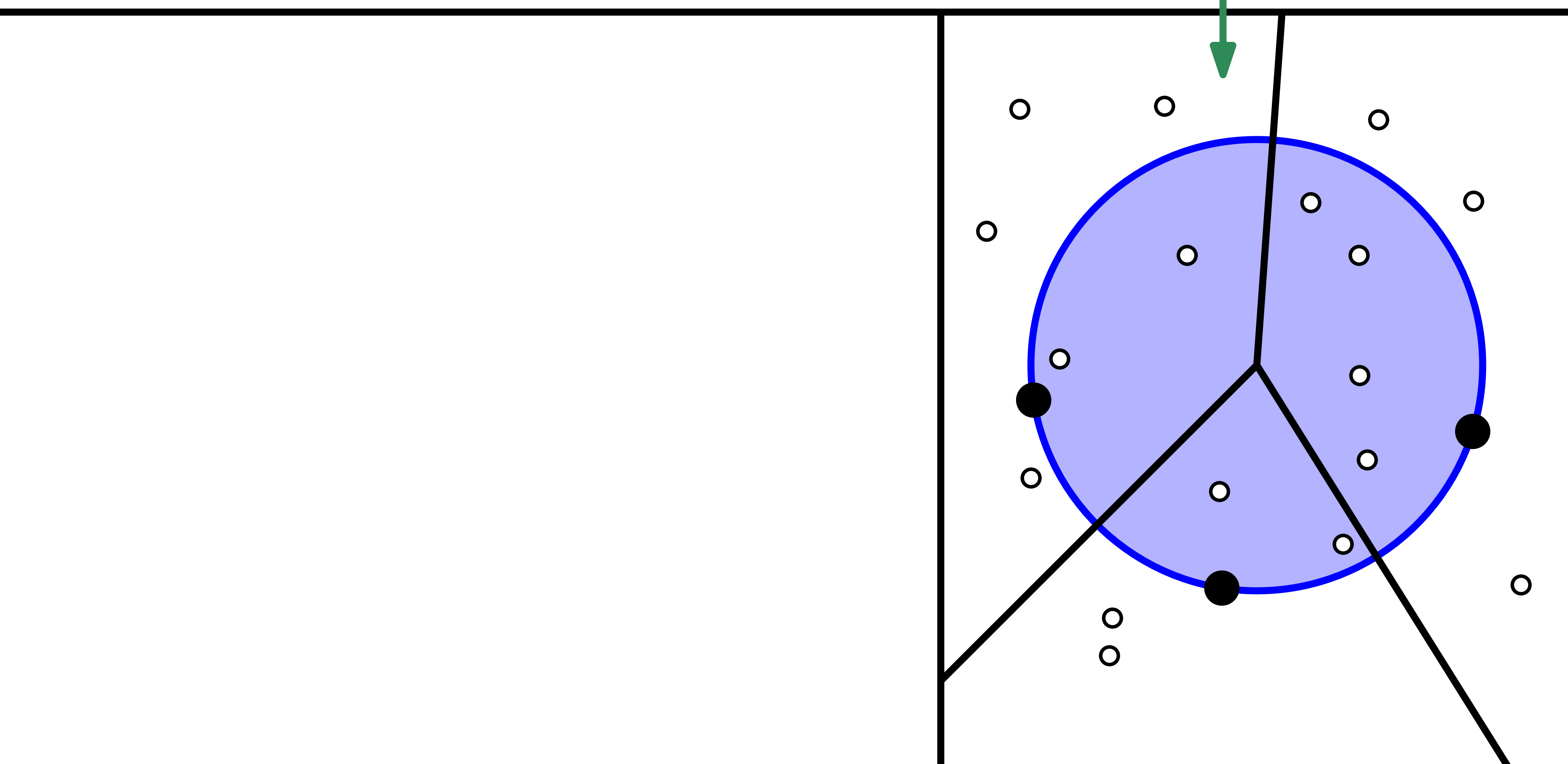
Given $R_2 \subset P$ such that:

- $|R_2| = O(s)$
- each vertex $v \in VD(R_2)$ has conflict set B_v with $|B_v| = O(n/s)$.

Computing the vertices

Given $R_2 \subset P$ such that:

- $|R_2| = O(s)$
- each vertex $v \in VD(R_2)$ has conflict set B_v with $|B_v| = O(n/s)$.

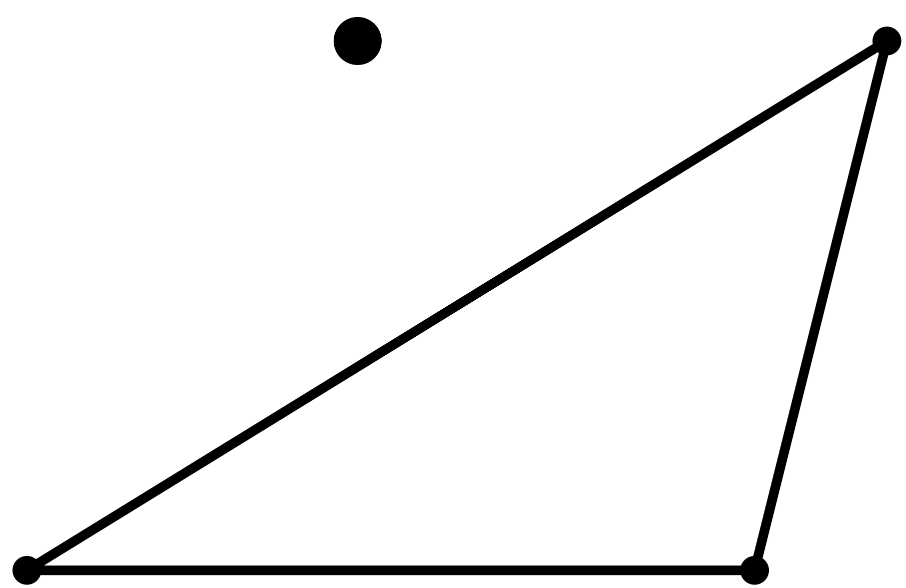


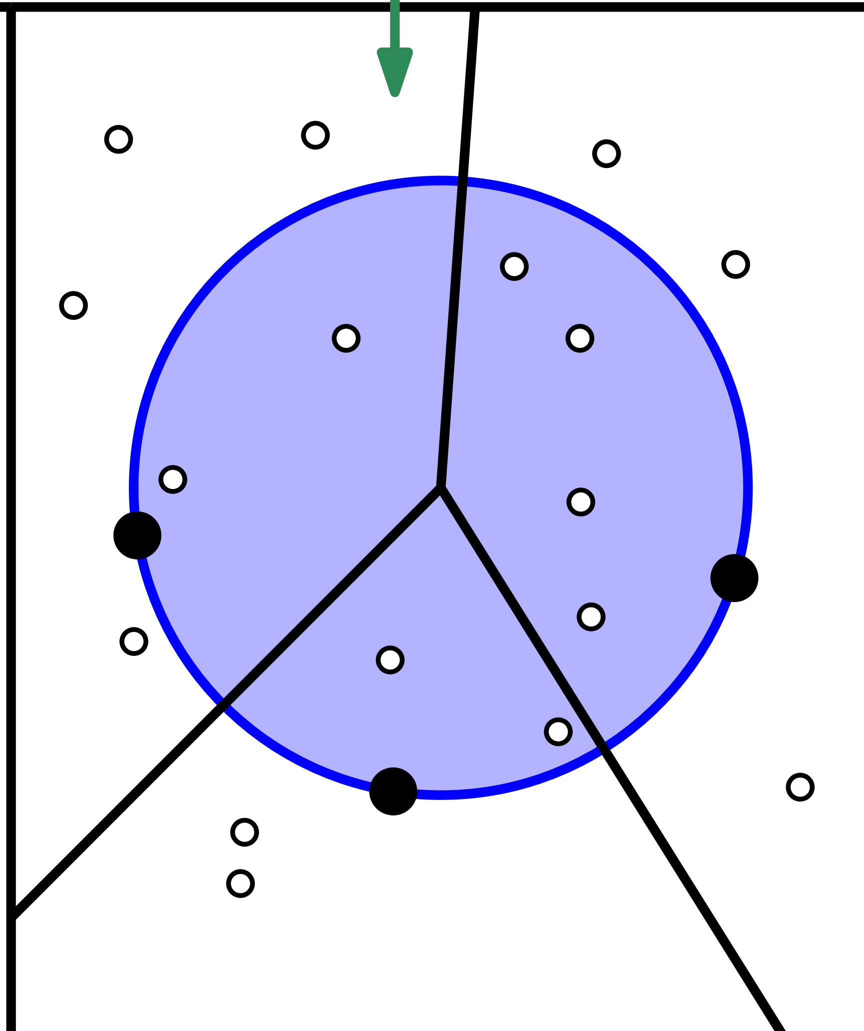
Computing the vertices

Given $R_2 \subset P$ such that:

- $|R_2| = O(s)$
- each vertex $v \in VD(R_2)$ has conflict set B_v with $|B_v| = O(n/s)$.

To report Voronoi vertices in $\Delta = \{v_1, v_2, v_3\} \subseteq R_2$ only consider points in $B_{v_1}, B_{v_2}, B_{v_3}$



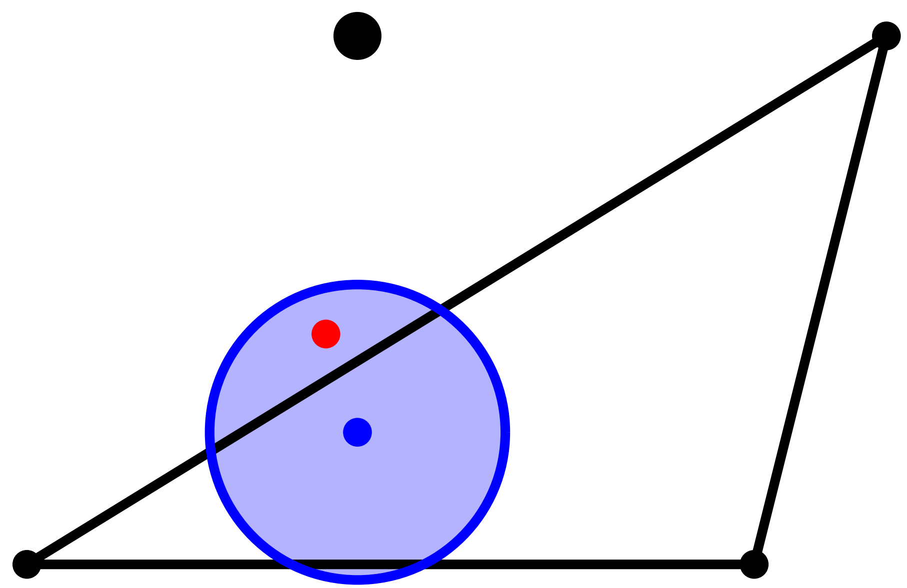


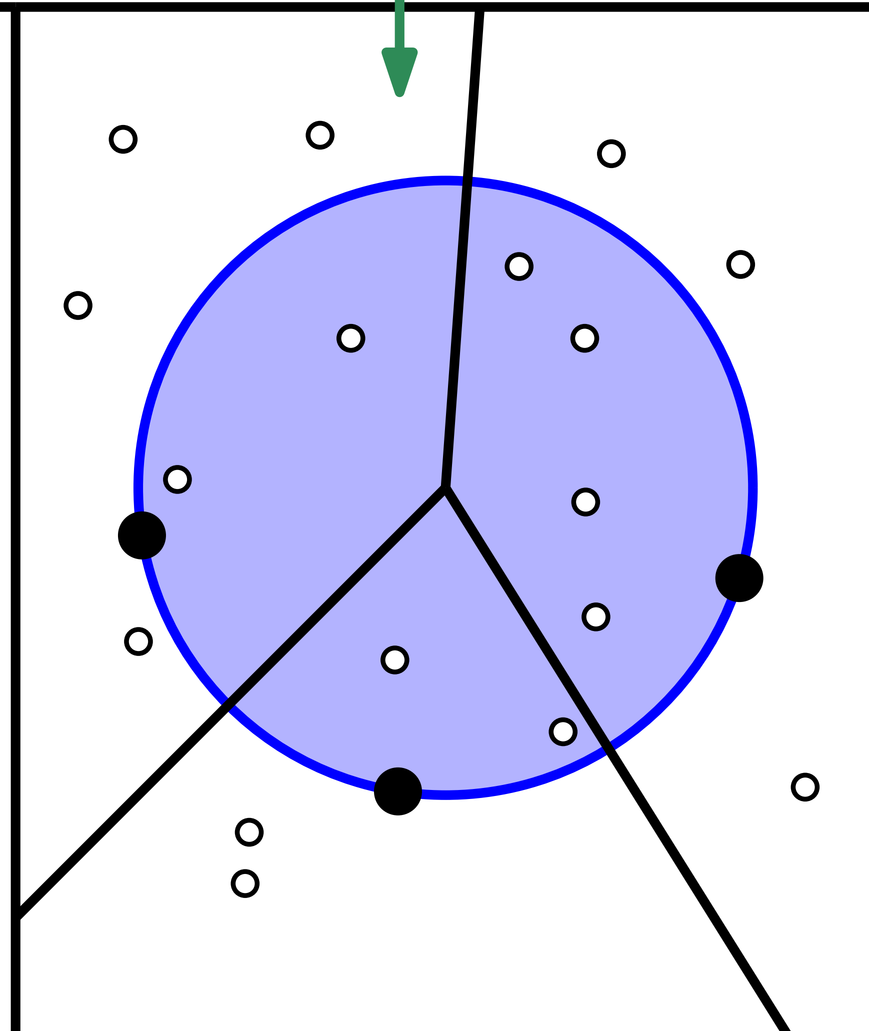
Computing the vertices

Given $R_2 \subset P$ such that:

- $|R_2| = O(s)$
- each vertex $v \in VD(R_2)$ has conflict set B_v with $|B_v| = O(n/s)$.

To report Voronoi vertices in $\Delta = \{v_1, v_2, v_3\} \subseteq R_2$ only consider points in $B_{v_1}, B_{v_2}, B_{v_3}$



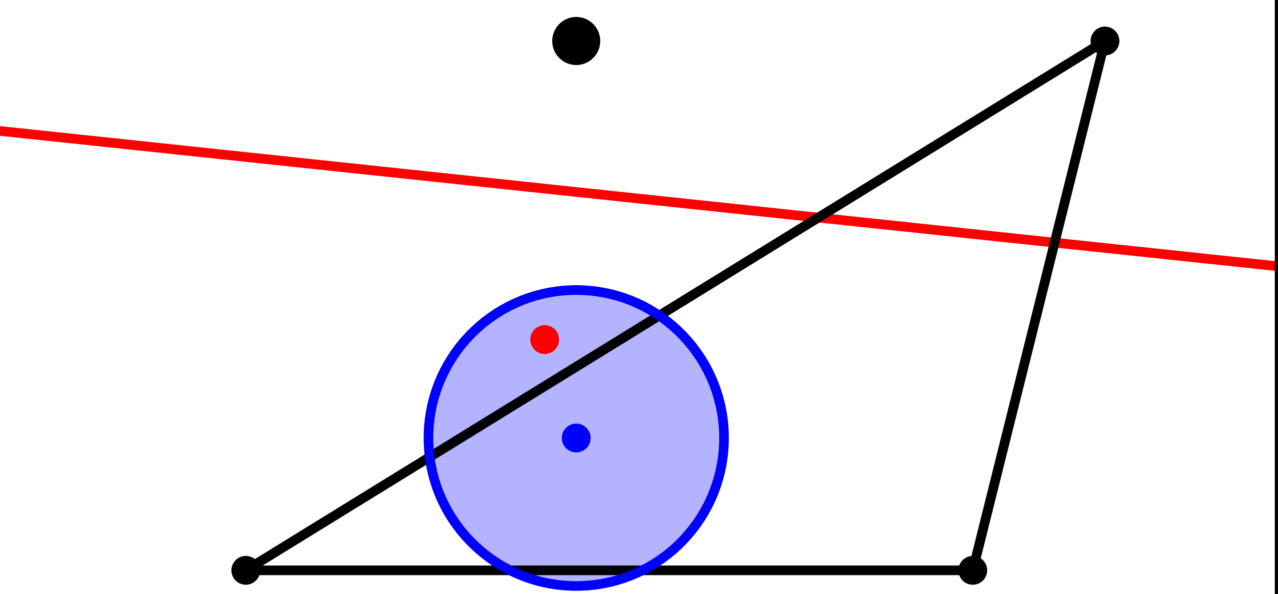


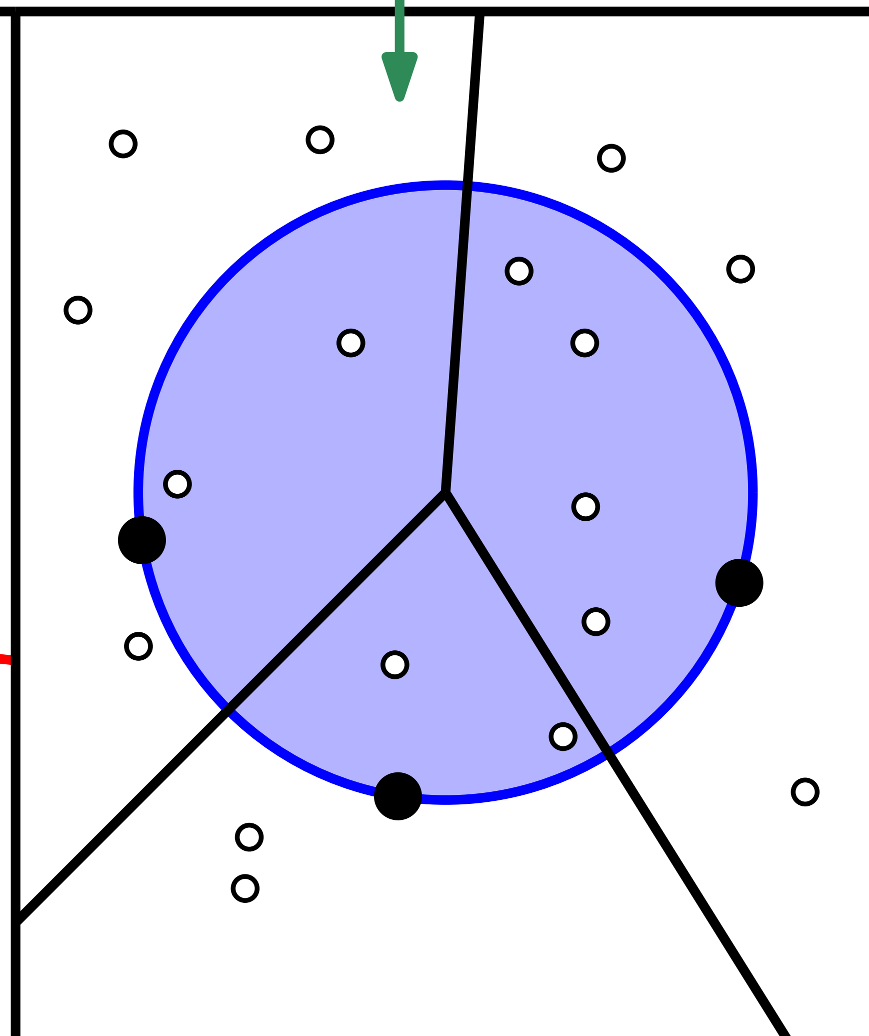
Computing the vertices

Given $R_2 \subset P$ such that:

- $|R_2| = O(s)$
- each vertex $v \in VD(R_2)$ has conflict set B_v with $|B_v| = O(n/s)$.

To report Voronoi vertices in $\Delta = \{v_1, v_2, v_3\} \subseteq R_2$ only consider points in $B_{v_1}, B_{v_2}, B_{v_3}$



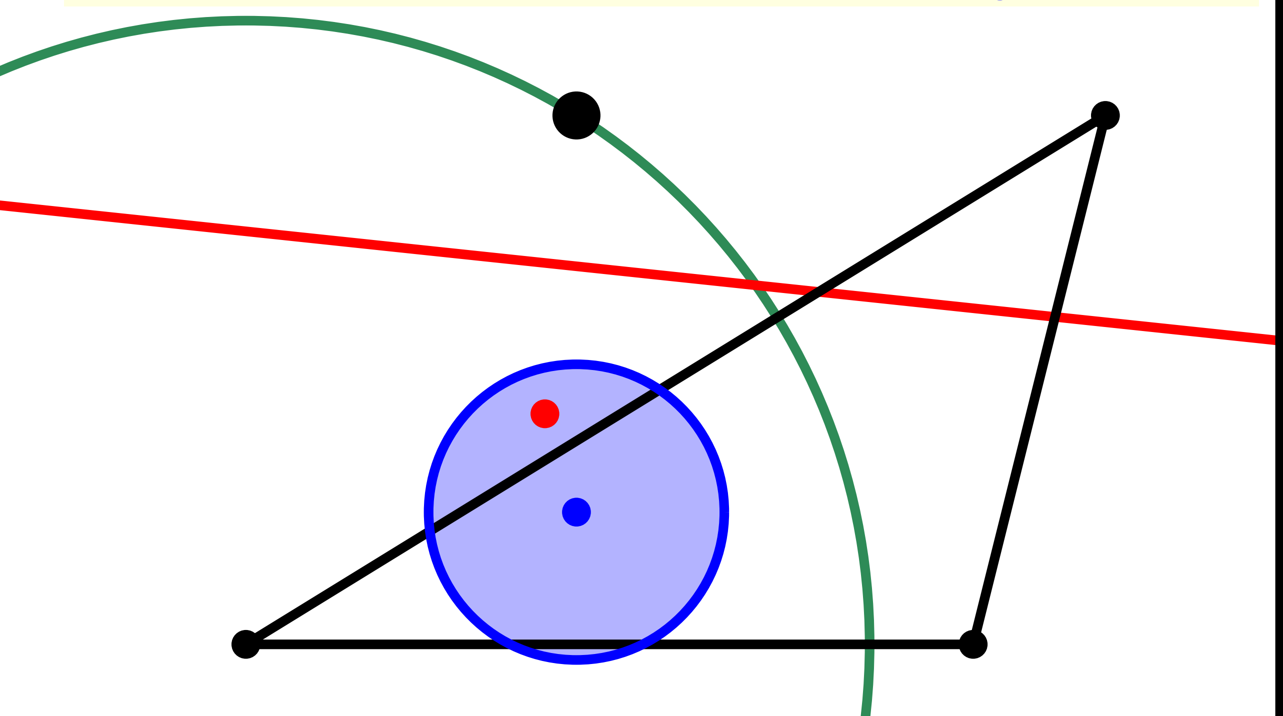


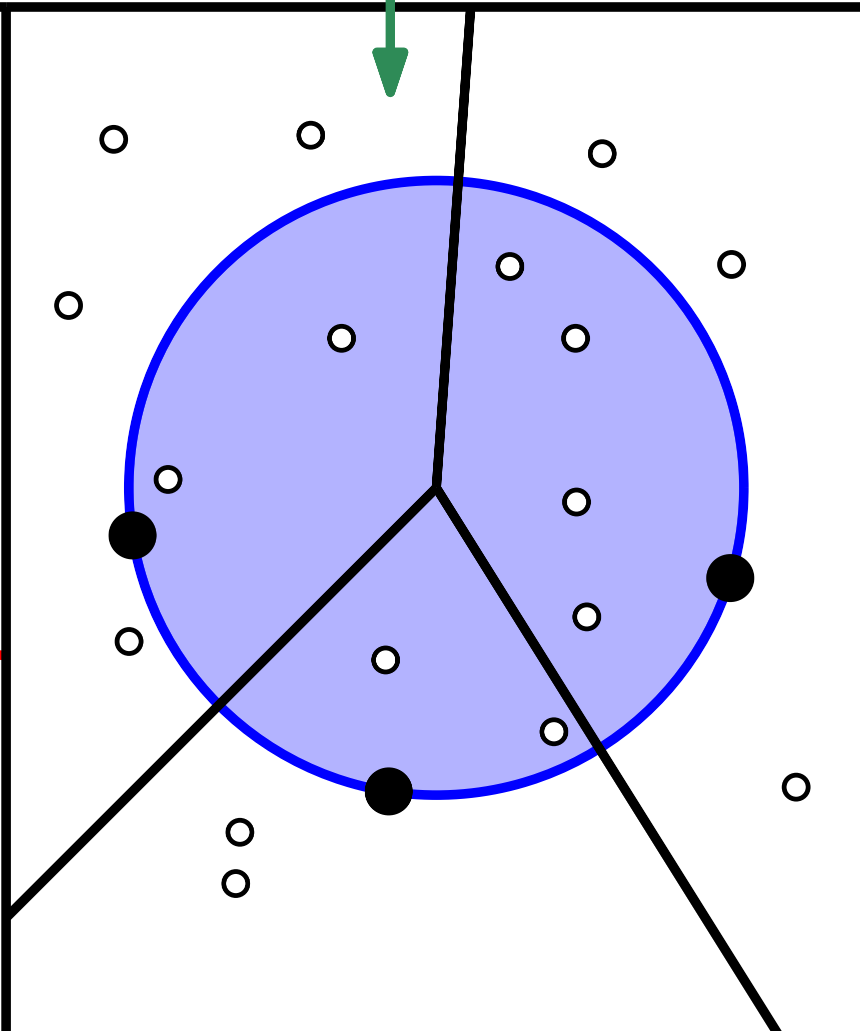
Computing the vertices

Given $R_2 \subset P$ such that:

- $|R_2| = O(s)$
- each vertex $v \in VD(R_2)$ has conflict set B_v with $|B_v| = O(n/s)$.

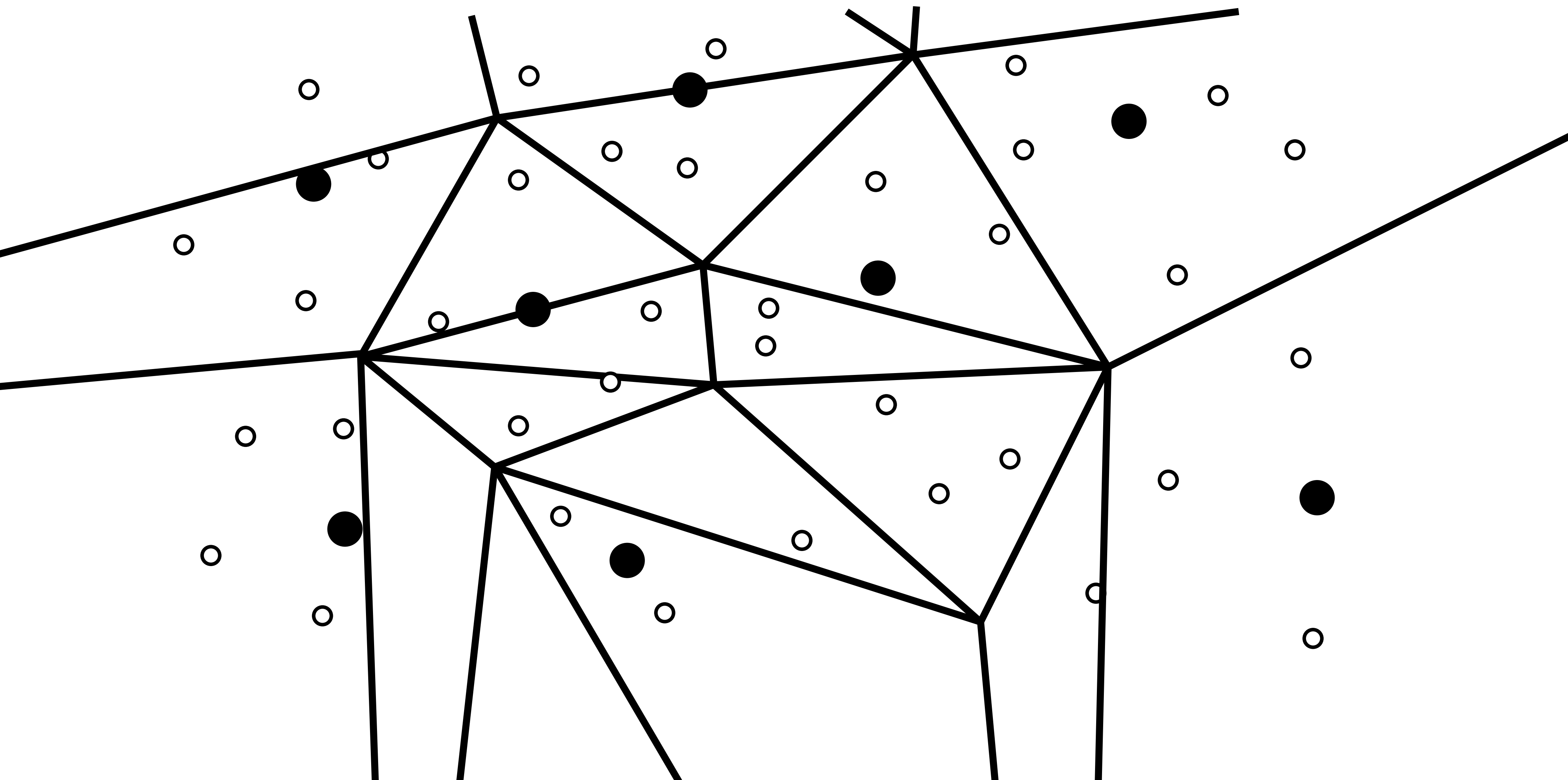
To report Voronoi vertices in $\Delta = \{v_1, v_2, v_3\} \subseteq R_2$ only consider points in $B_{v_1}, B_{v_2}, B_{v_3}$





Computing the Voronoi vertices

Problem: $O(s)$ voronoi diagrams of size $O(n/s)$ to compute

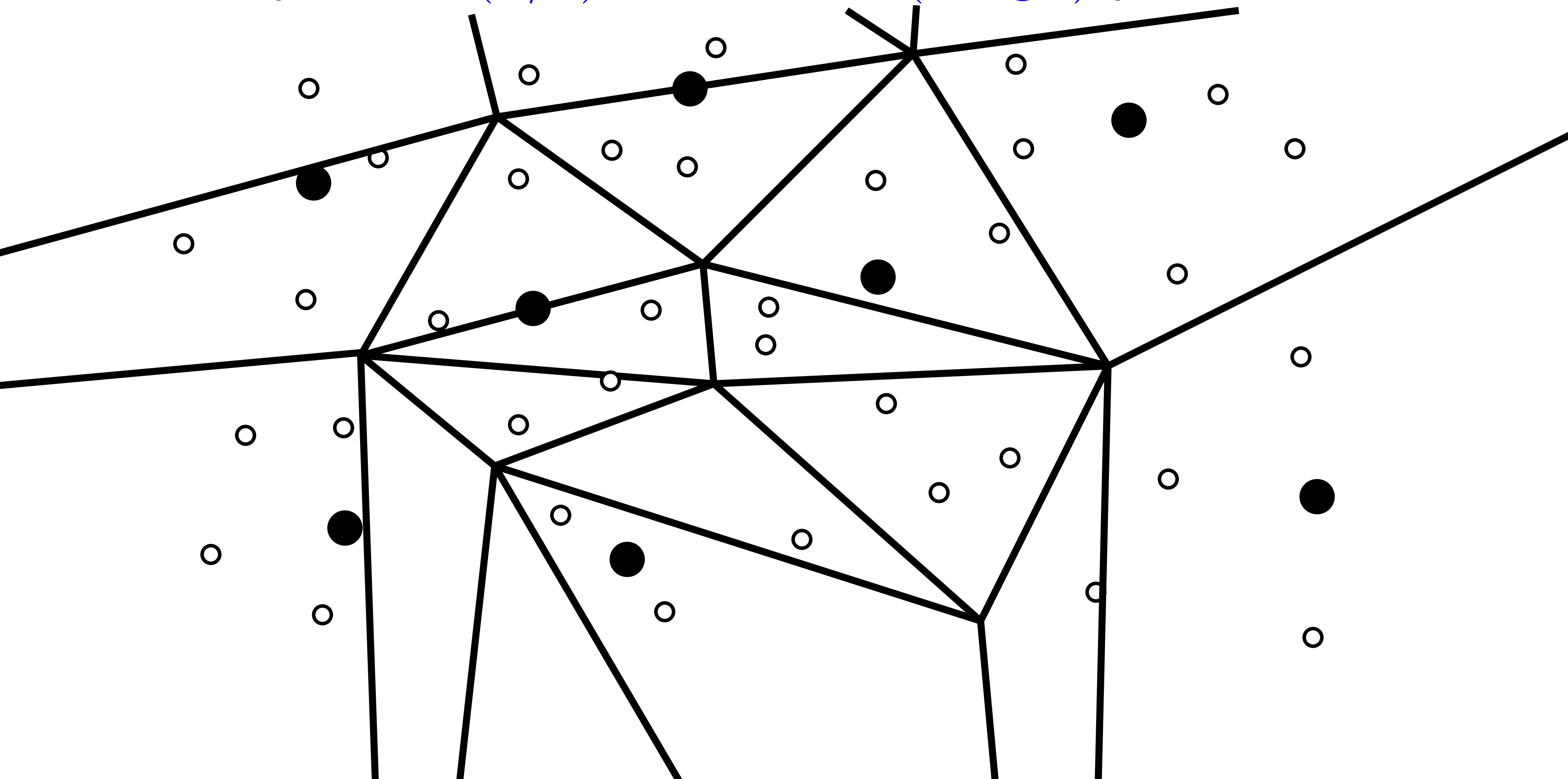


Computing the Voronoi vertices

Problem: $O(s)$ voronoi diagrams of size $O(n/s)$ to compute

Solution: Use $O(1)$-memory algorithm on each triangle:

- Allocate each triangle $O(1)$ memory
- Scan points $O(n/s)$ times and $O(n \log s)$ per scan

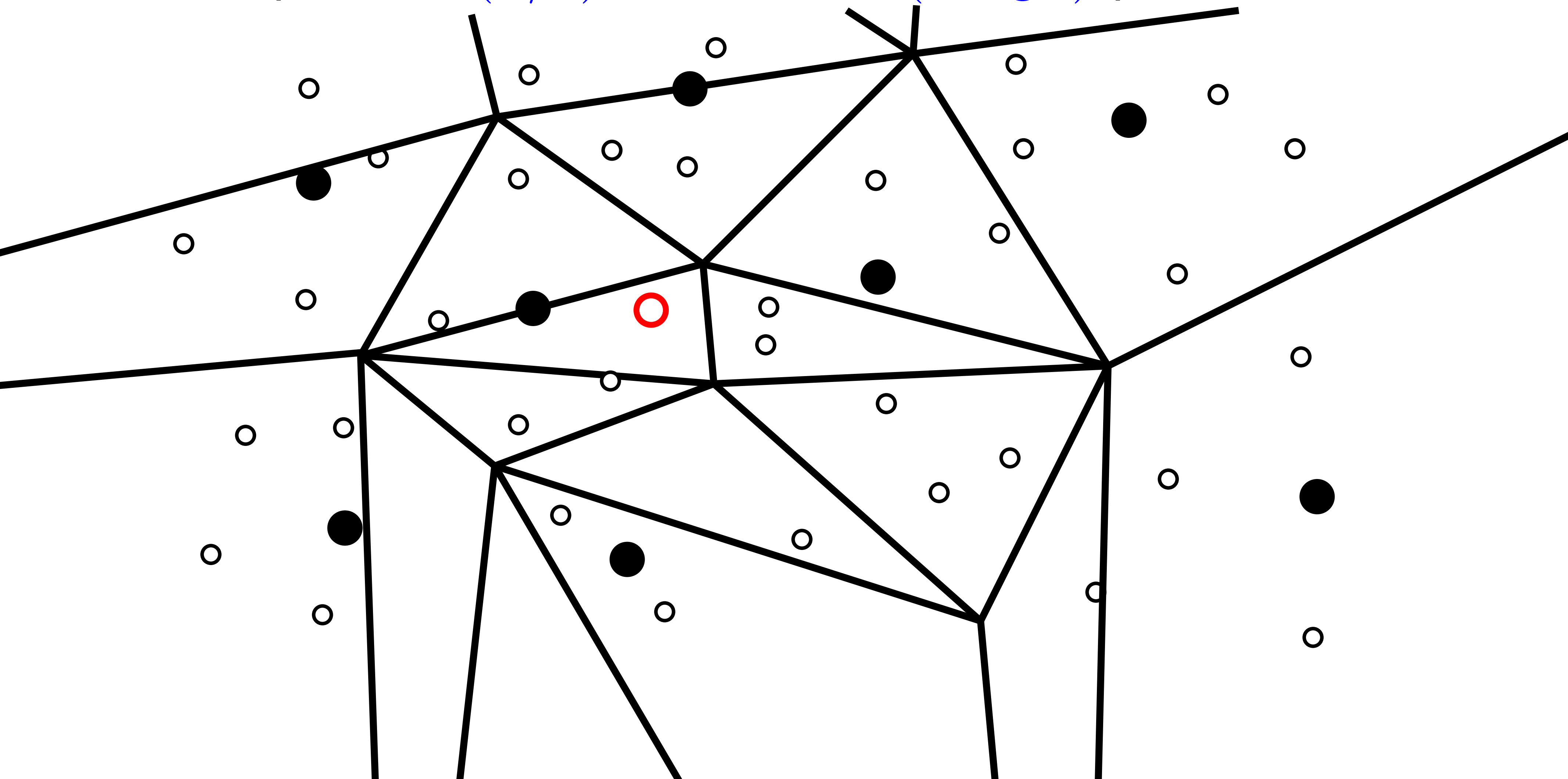


Computing the Voronoi vertices

Problem: $O(s)$ voronoi diagrams of size $O(n/s)$ to compute

Solution: Use $O(1)$-memory algorithm on each triangle:

- Allocate each triangle $O(1)$ memory
- Scan points $O(n/s)$ times and $O(n \log s)$ per scan

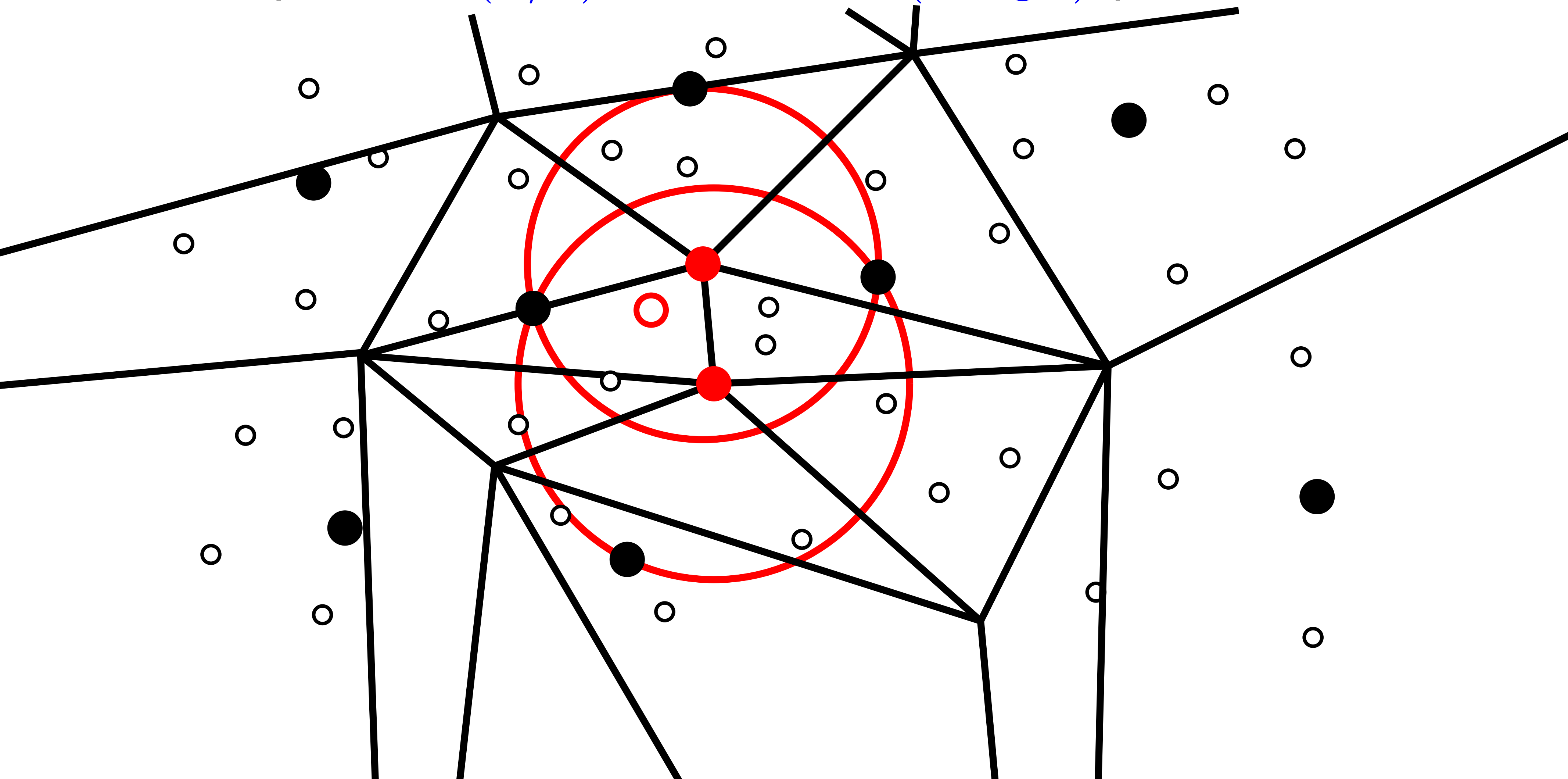


Computing the Voronoi vertices

Problem: $O(s)$ voronoi diagrams of size $O(n/s)$ to compute

Solution: Use $O(1)$-memory algorithm on each triangle:

- Allocate each triangle $O(1)$ memory
- Scan points $O(n/s)$ times and $O(n \log s)$ per scan

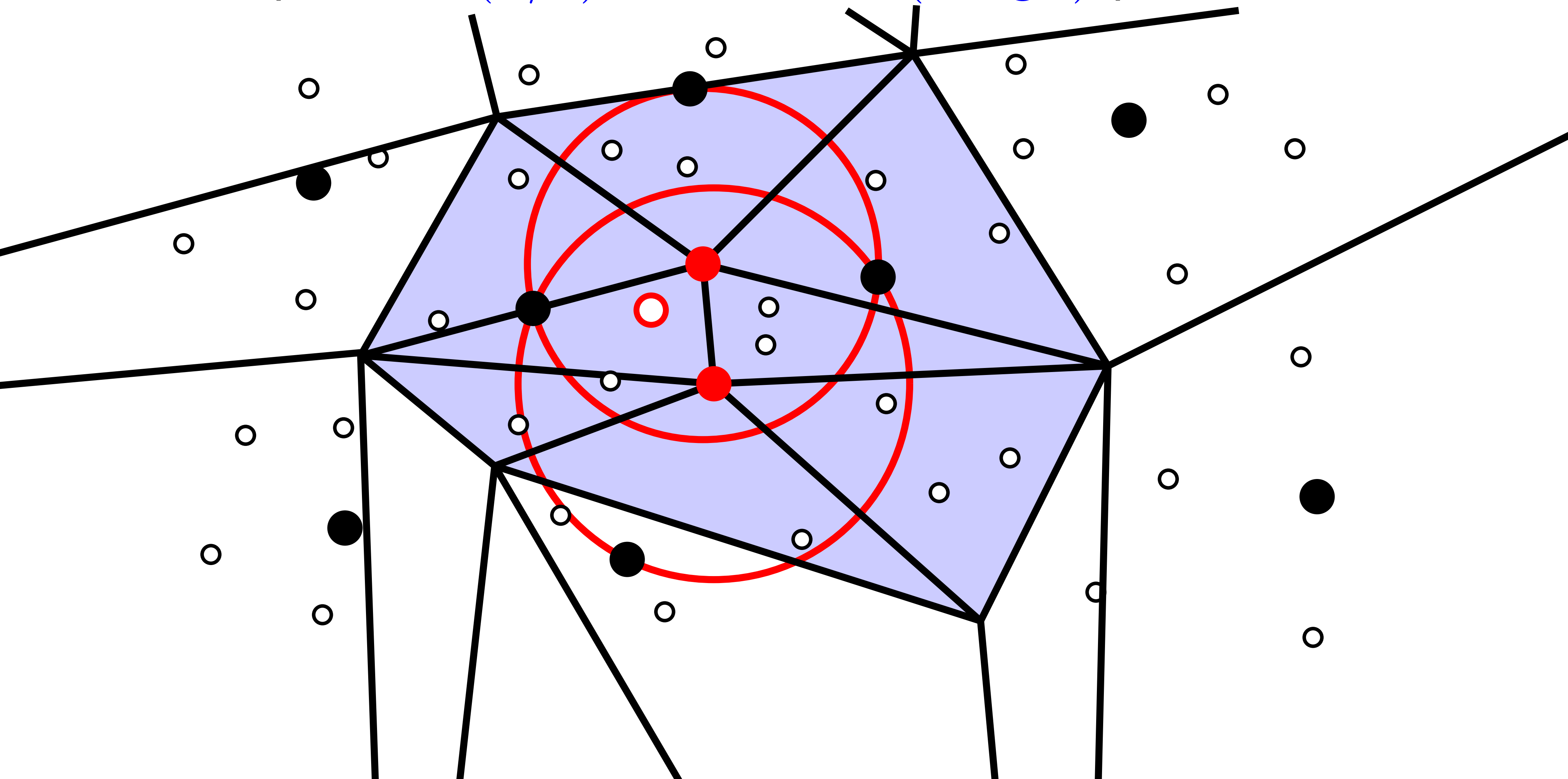


Computing the Voronoi vertices

Problem: $O(s)$ voronoi diagrams of size $O(n/s)$ to compute

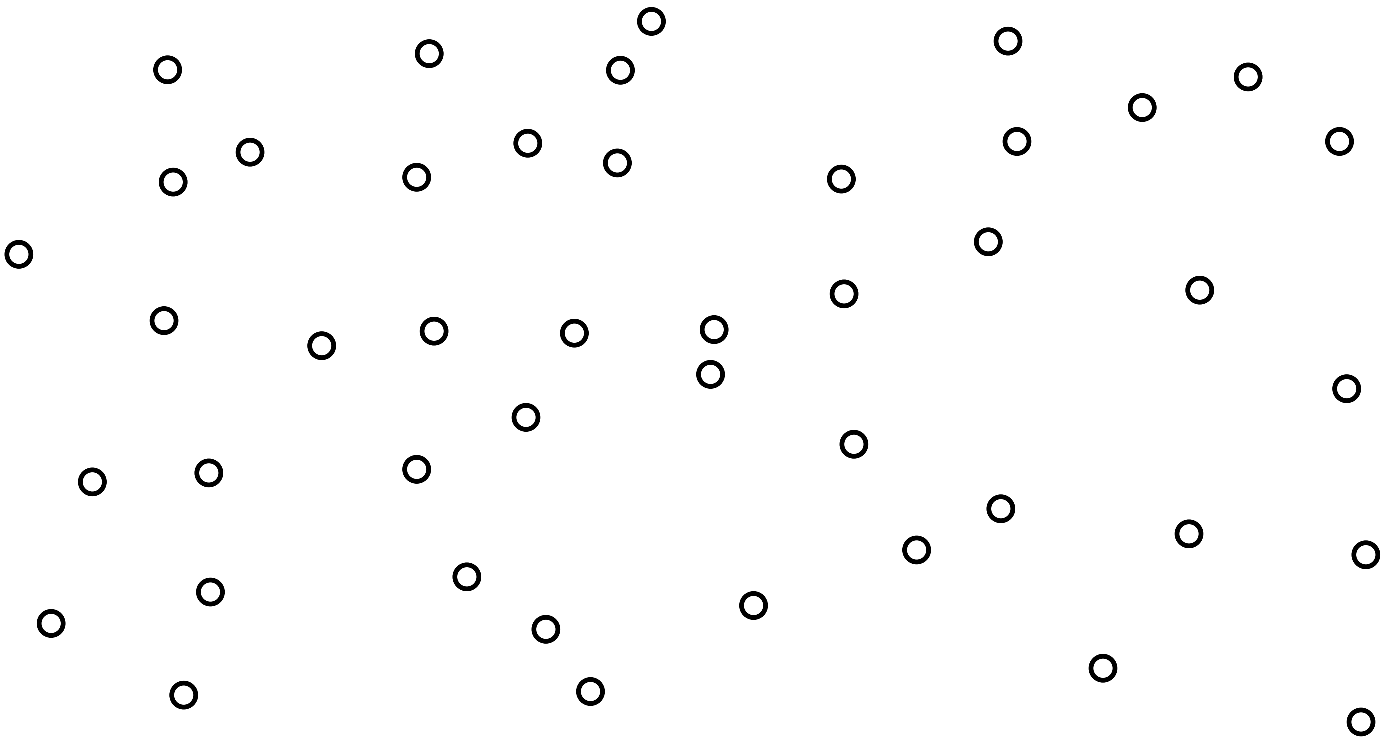
Solution: Use $O(1)$-memory algorithm on each triangle:

- Allocate each triangle $O(1)$ memory
- Scan points $O(n/s)$ times and $O(n \log s)$ per scan



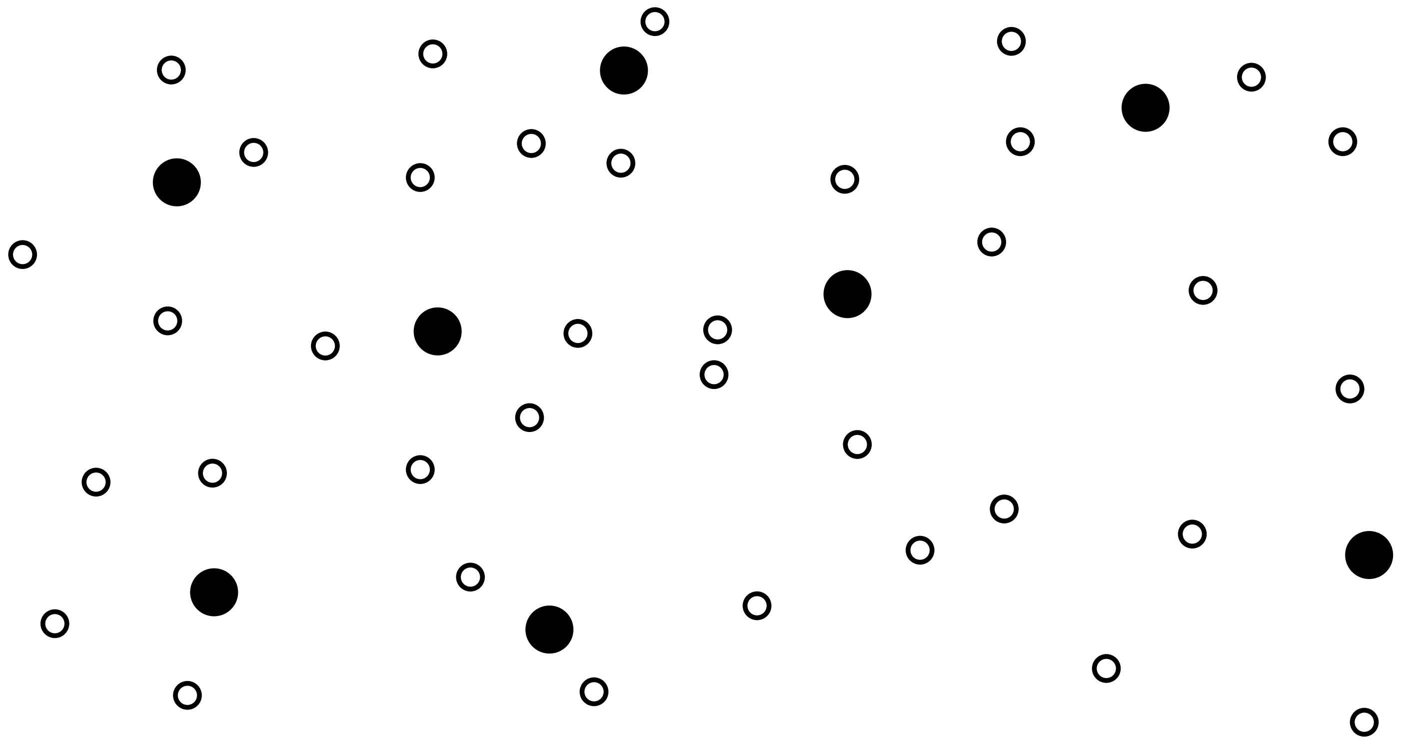
Computing R_2

- Take random set $R \subset P$ of size $\Theta(s)$
- Compute and triangulate $VD(R)$
- Compute sizes of conflict sets B_v for $v \in VD(R)$



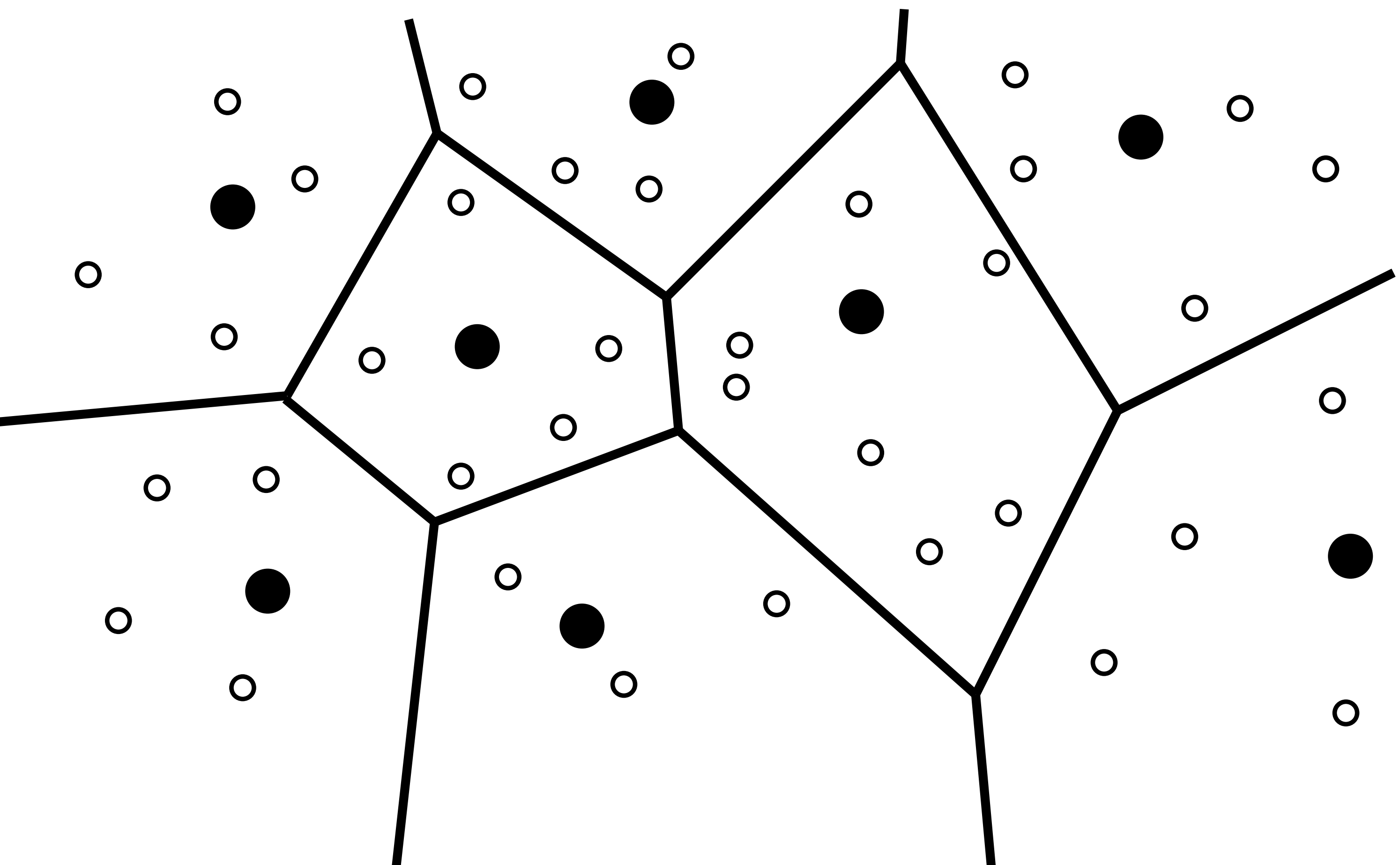
Computing R_2

- Take random set $R \subset P$ of size $\Theta(s)$
- Compute and triangulate $VD(R)$
- Compute sizes of conflict sets B_v for $v \in VD(R)$



Computing R_2

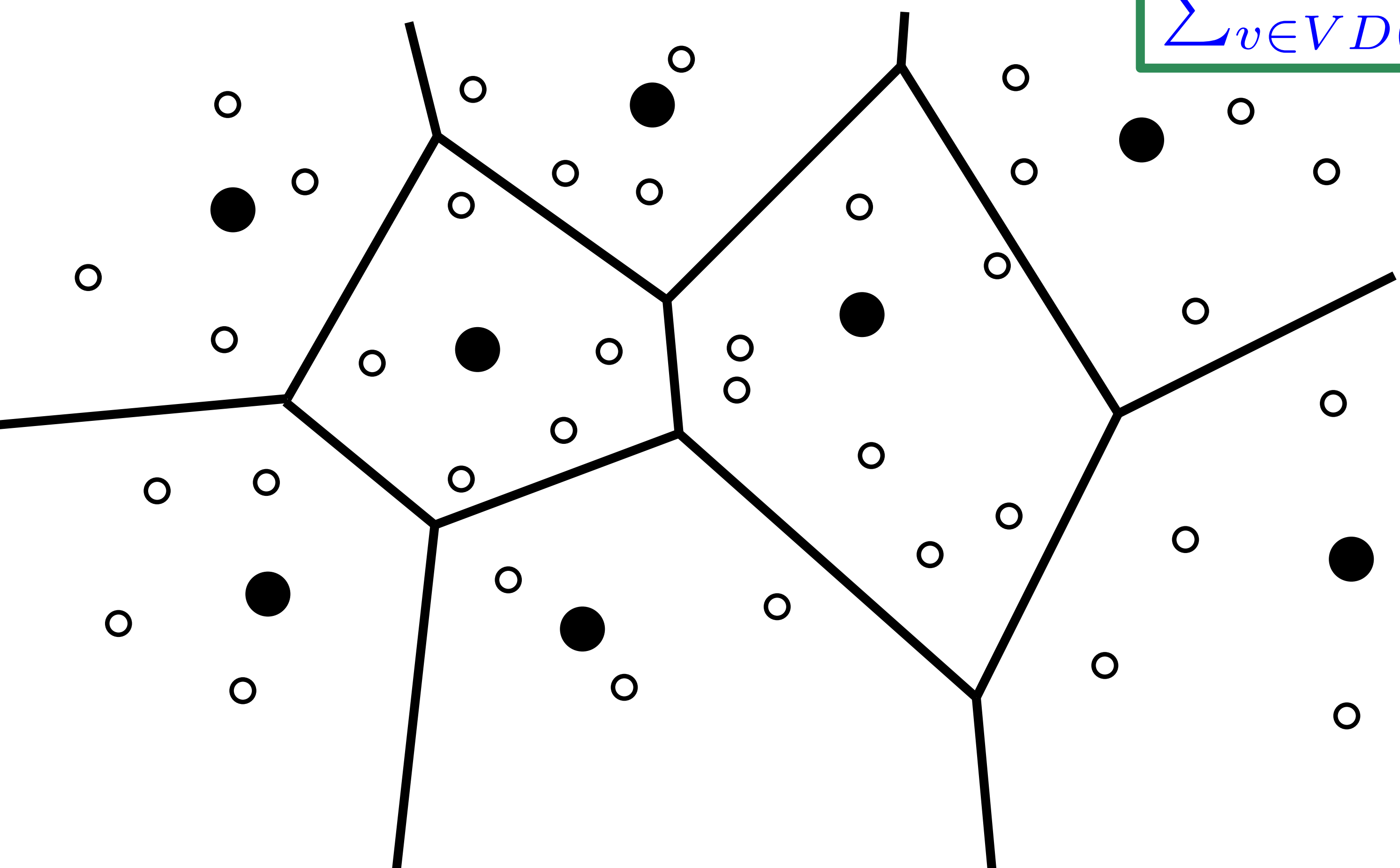
- Take random set $R \subset P$ of size $\Theta(s)$
- Compute and triangulate $VD(R)$
- Compute sizes of conflict sets B_v for $v \in VD(R)$



Computing R_2

- Take random set $R \subset P$ of size $\Theta(s)$
- Compute and triangulate $VD(R)$
- Compute sizes of conflict sets B_v for $v \in VD(R)$
- If size of conflict sets is too large, restart

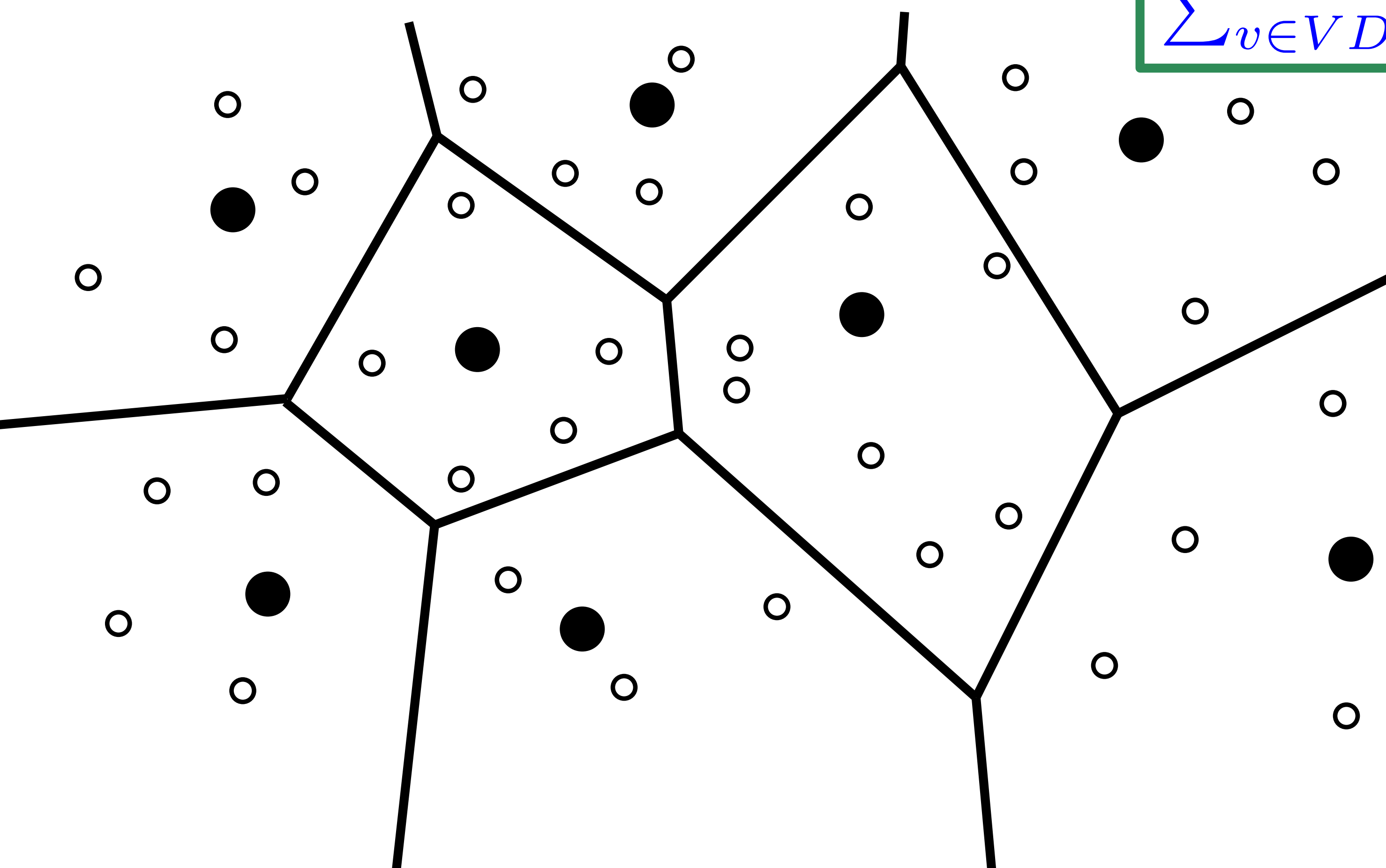
$$\boxed{\begin{gathered} t_v = |B_v| \cdot s/n \\ \sum_{v \in VD(R)} t_v \log t_v = O(s) \end{gathered}}$$



Computing R_2

- Take random set $R \subset P$ of size $\Theta(s)$
- Compute and triangulate $VD(R)$
- Compute sizes of conflict sets B_v for $v \in VD(R)$
- If size of conflict sets is too large, restart

$$t_v = |B_v| \cdot s/n$$
$$\sum_{v \in VD(R)} t_v \log t_v = O(s)$$

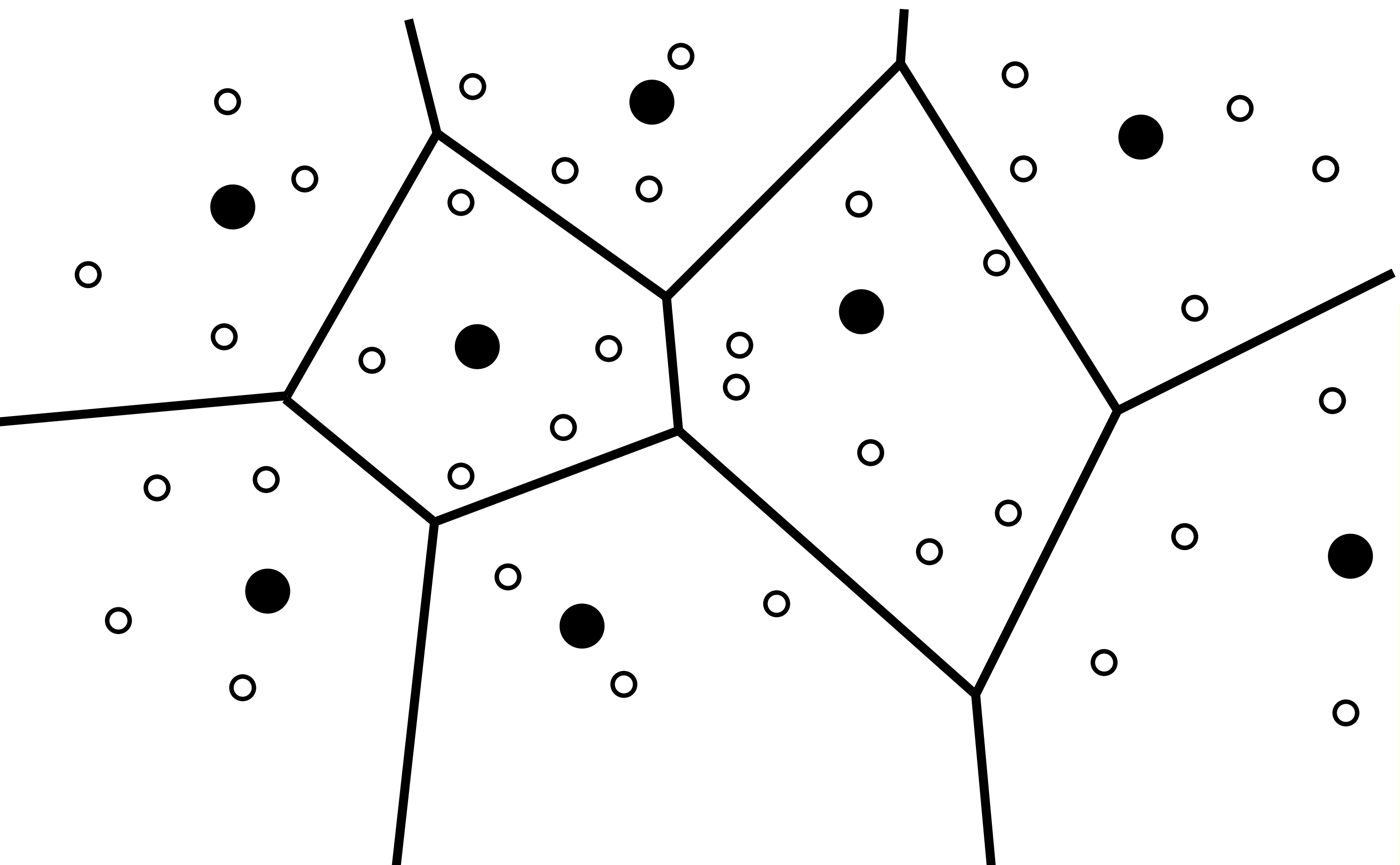


time to sample:
$O(n + s \log s)$
count conflict size:
$O(n \log s)$
total:
$O(n \log s)$

Computing R_2

Problem: For some $v \in VD(R)$ we may have $B_v \gg n/s$

Solution:
- Sample $\Theta(t_v \log t_v)$ extra points from B_v for any $v \in VD(R)$ with $t_v \geq 2$.
- Recompute conflict sizes
- Continue sampling in large conflict sets

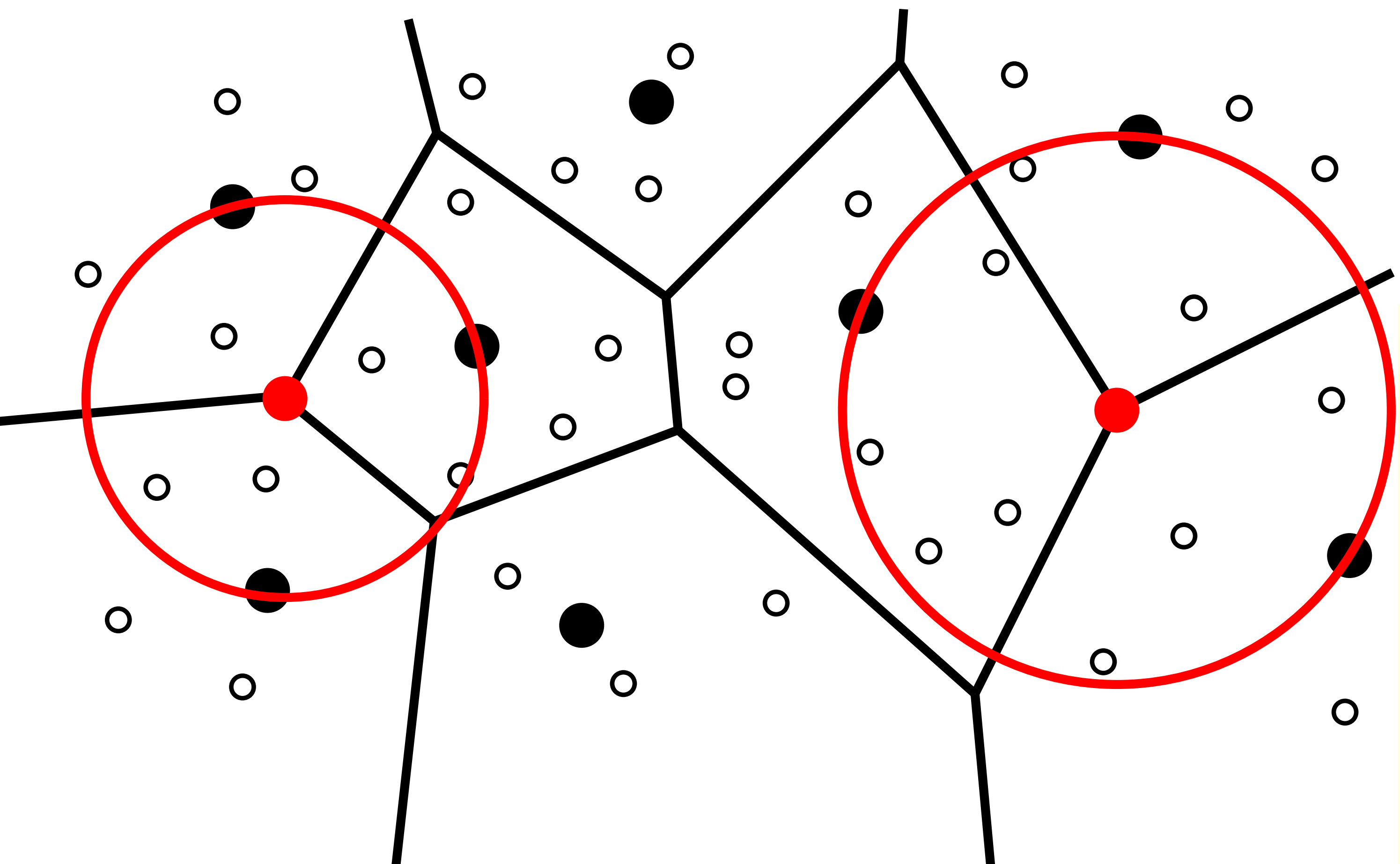


1 sampling round: $O(n \log s + s \log s)$
expected #rounds: $O(\log^* s)$
total: $O(n \log s \log^* s)$

Computing R_2

Problem: For some $v \in VD(R)$ we may have $B_v \gg n/s$

Solution:
- Sample $\Theta(t_v \log t_v)$ extra points from B_v for any $v \in VD(R)$ with $t_v \geq 2$.
- Recompute conflict sizes
- Continue sampling in large conflict sets

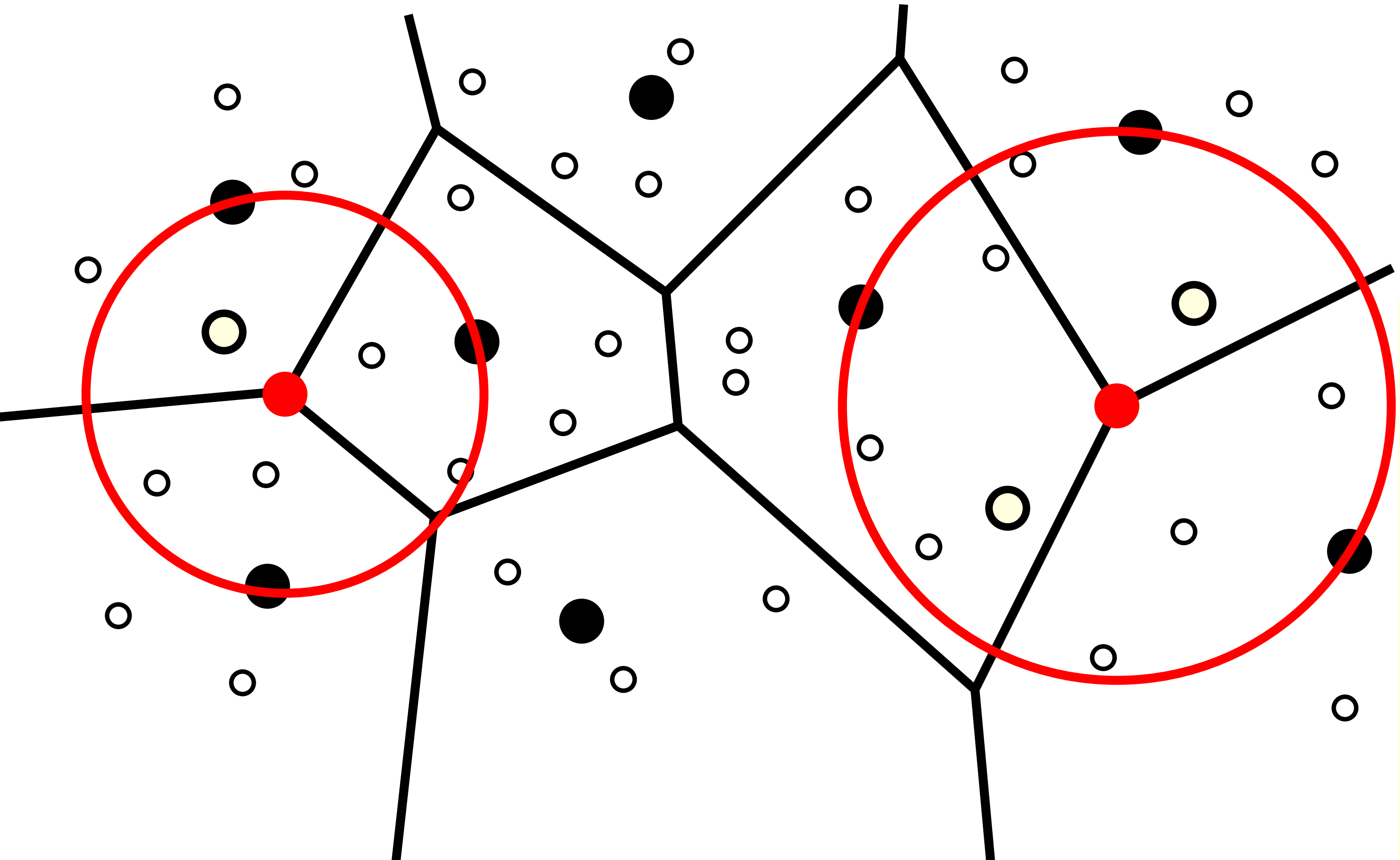


1 sampling round:
$O(n \log s + s \log s)$
expected #rounds:
$O(\log^* s)$
total:
$O(n \log s \log^* s)$

Computing R_2

Problem: For some $v \in VD(R)$ we may have $B_v \gg n/s$

Solution:
- Sample $\Theta(t_v \log t_v)$ extra points from B_v for any $v \in VD(R)$ with $t_v \geq 2$.
- Recompute conflict sizes
- Continue sampling in large conflict sets

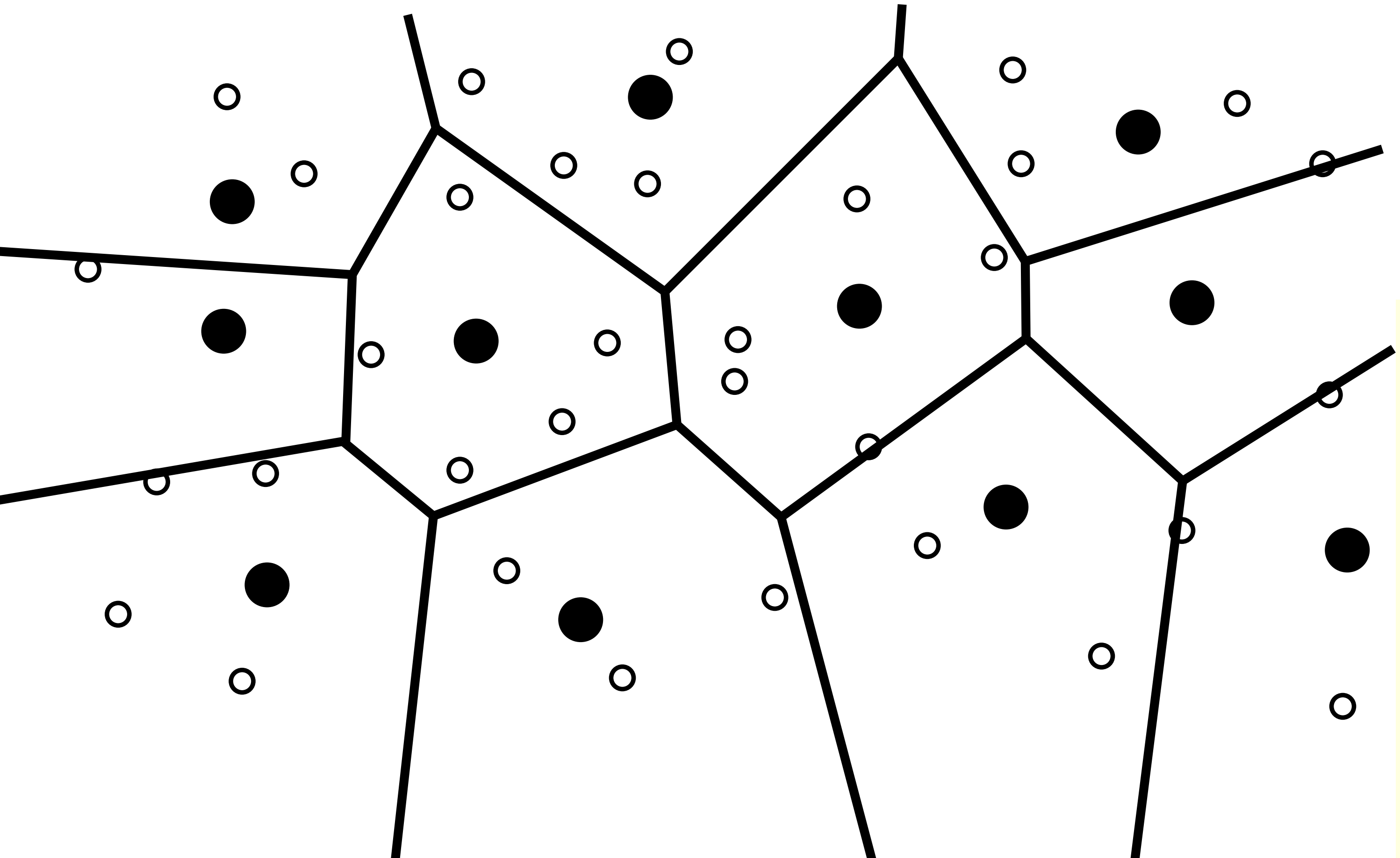


1 sampling round: $O(n \log s + s \log s)$
expected #rounds: $O(\log^* s)$
total: $O(n \log s \log^* s)$

Computing R_2

Problem: For some $v \in VD(R)$ we may have $B_v \gg n/s$

Solution:
- Sample $\Theta(t_v \log t_v)$ extra points from B_v for any $v \in VD(R)$ with $t_v \geq 2$.
- Recompute conflict sizes
- Continue sampling in large conflict sets



1 sampling round:
$O(n \log s + s \log s)$
expected #rounds:
$O(\log^* s)$
total:
$O(n \log s \log^* s)$

Putting it together

Computing R_2: expected $O(n \log s \log^* s)$

Computing for each triangle: expected $O((n^2/s) \log s)$

$$\Downarrow$$

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

(Almost optimal for both linear and constant memory)

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