On Approximate Halfspace Range Counting and Relative ε-Approximations

Boris Aronov[1] Sariel Har-Peled[2] Micha Sharir[3]

[1]Polytechnic University, Brooklyn, USA

[2]UIUC, Illinois, USA

[3]Tel-Aviv University, Israel

June 9, 2007

2: Counting

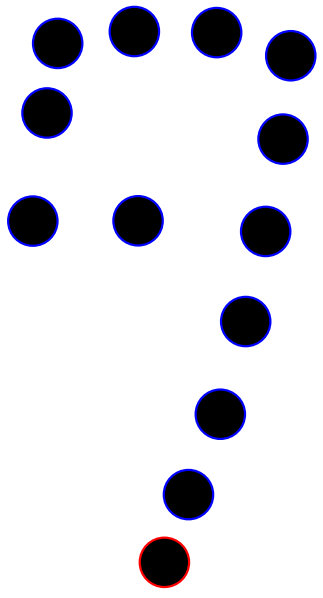
Counting

P: Set of **n** points in $\mathbb{R}^{\mathbf{d}}$

Preprocess **P** for **Counting**

h: query halfspace

Output: $|\mathbf{P} \cap \mathbf{h}|$



2: Counting

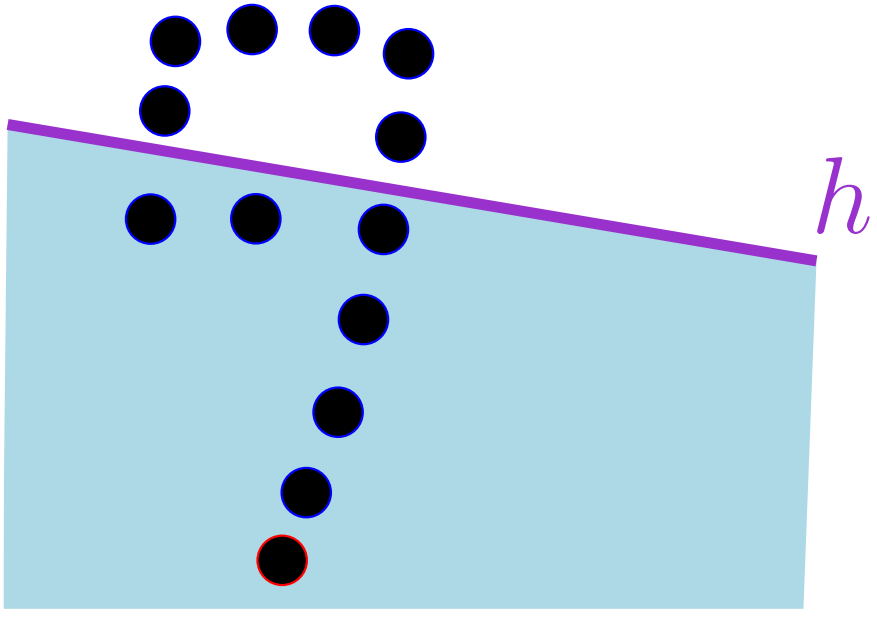
Counting

P: Set of **n** points in $\mathbb{R}^{\mathbf{d}}$

Preprocess **P** for **Counting**

h: query halfspace

Output: $|\mathbf{P} \cap \mathbf{h}|$



2: Counting

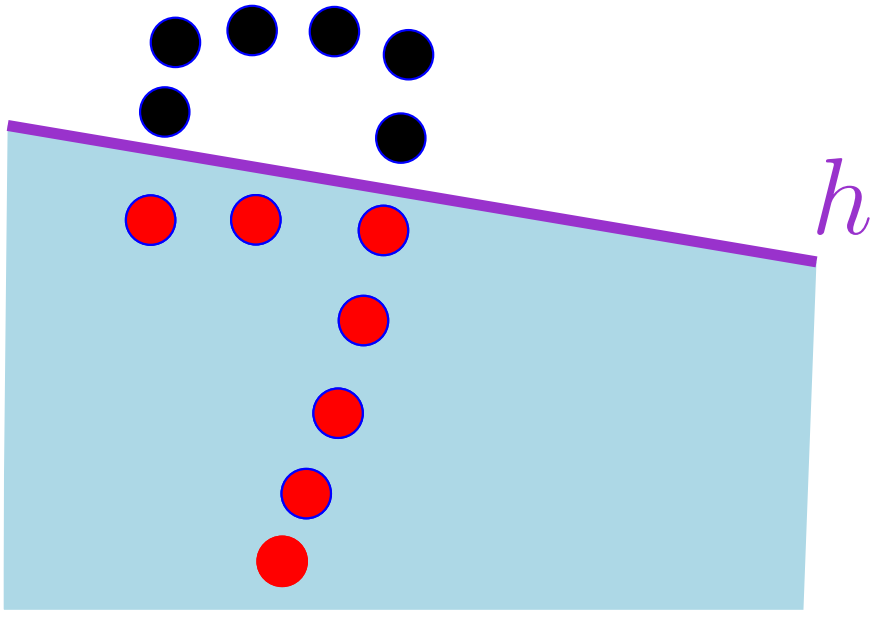
Counting

P: Set of **n** points in $\mathbb{R}^{\mathbf{d}}$

Preprocess **P** for **Counting**

h: query halfspace

Output: $|\mathbf{P} \cap \mathbf{h}|$



3: Everybody knows that the dice are loaded

Already known to the Czech

(and many other nationalities):

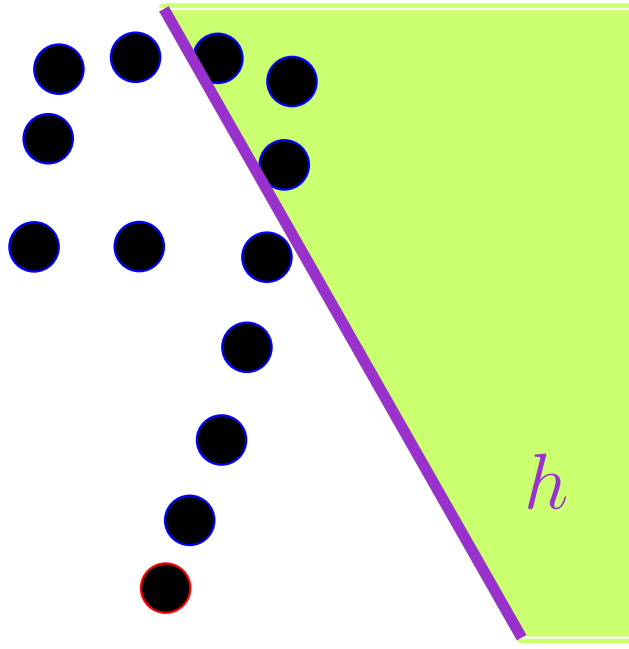
- Storage/preprocessing: $\approx \mathbf{O(n)}$,
- Query time: $\mathbf{O(n^{1-1/d})}$
  via Partition Trees **[Matoušek, 1992a]**

Let's call this **"inefficient"**.

4: What's "Efficient"?

Halfspace Range Reporting / Emptiness

Report $\mathbf{P} \cap \mathbf{h}$ / Is $\mathbf{P} \cap \mathbf{h} = \emptyset$?



4: What's "Efficient"?

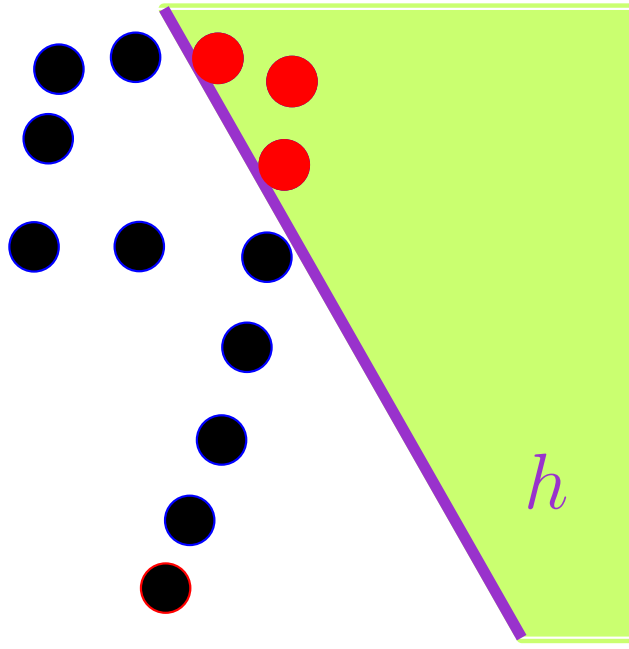
Halfspace Range Reporting / Emptiness

Report $\mathbf{P \cap h}$ / Is $\mathbf{P \cap h = \emptyset}$?

Again, the Czech know how to do it:

[Matoušek, 1992b]:

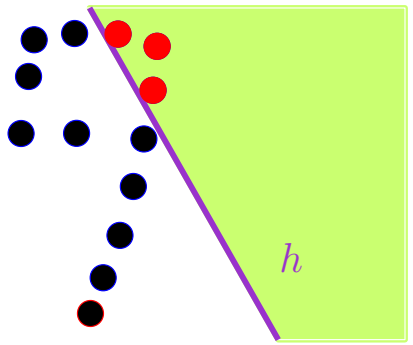
Shallow Partition Trees



4: What's "Efficient"?

Halfspace Range Reporting / Emptiness

Report $\mathbf{P \cap h}$ / Is $\mathbf{P \cap h = \emptyset}$?
Again, the Czech know how to do it:
[Matoušek, 1992b]:
Shallow Partition Trees



Performance

storage/preprocessing: $\approx \mathbf{O(n)}$,
Query time: $\mathbf{O(n^{1-1/\lfloor d/2 \rfloor}(+k))}$
Now this is **efficient**!

4: What's "Efficient"?

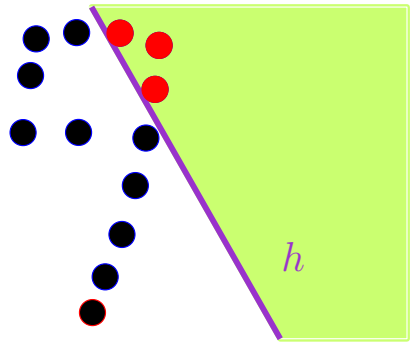
Halfspace Range Reporting / Emptiness

Report $\mathbf{P \cap h}$ / Is $\mathbf{P \cap h = \emptyset}$?

Again, the Czech know how to do it:

[Matoušek, 1992b]:

Shallow Partition Trees



Range Searching Performance

Storage/preprocessing: $\approx \mathbf{O(n)}$,

	efficient Emptiness	inefficient Counting
Query time	$\mathbf{O(n^{1-1/\lfloor d/2 \rfloor}(+k))}$	$\mathbf{O(n^{1-1/d})}$

Interesting for $\mathbf{d = 3}$...

5: But we want COUNTING

...and we want it now, and we don't care how!

Impossible(?)

Naa... use **Approximate Counting:**

Approx counting

Given $\varepsilon > 0$

h: query halfspace

compute **C** s.t.

$$(1-\varepsilon)|\mathbf{P} \cap \mathbf{h}| \leq \mathbf{C} \leq (1+\varepsilon)|\mathbf{P} \cap \mathbf{h}|$$

Relative error = error proportional to $\mathbf{weight(h)}$.

5: But we want COUNTING

...and we want it now, and we don't care how!

Impossible(?)

Naa... use **Approximate Counting:**

Approx counting

Given $\varepsilon > 0$

h: query halfspace

compute **C** s.t.

$$(1-\varepsilon)|\mathbf{P} \cap \mathbf{h}| \leq \mathbf{C} \leq (1+\varepsilon)|\mathbf{P} \cap \mathbf{h}|$$

Relative error = error proportional to **weight(h)**.

5: But we want COUNTING

...and we want it now, and we don't care how!

Impossible(?)

Naa... use **Approximate Counting:**

Approx counting

Given $\varepsilon > 0$

h: query halfspace

compute **C** s.t.

$$(1-\varepsilon)|\mathbf{P} \cap \mathbf{h}| \leq \mathbf{C} \leq (1+\varepsilon)|\mathbf{P} \cap \mathbf{h}|$$

Relative error = error proportional to $\mathbf{weight(h)}$.

6: Approximate Counting

- Tricky: If $|\mathbf{P} \cap \mathbf{h}| < \mathbf{1}/\varepsilon \Rightarrow$ **exact answer**!
- No better than **Emptiness queries**

Goal: Approx Counting

Achieve performance same as emptiness:
Storage/preprocessing $\approx \mathbf{O(n)}$,
Query time: $\approx \mathbf{O}_\varepsilon\left(\mathbf{n}^{1-1/\lfloor \mathbf{d}/2 \rfloor}\right)$

7: Self-competing alternative approaches

[Aronov and Har-Peled, 2005]

Reduction from emptiness to approximate counting.
Estimating $|\mathbf{P} \cap \mathbf{h}|$ using random samples.

[Kaplan and Sharir, 2006]

Estimating $|\mathbf{P} \cap \mathbf{h}|$ using random permutation:
Minimum-rank element in $\mathbf{P} \cap \mathbf{h}$ is a good estimator
(General technique of **[Cohen, 1997]**)

In both cases, repeat $\mathbf{O}\left(\frac{1}{\varepsilon^2}\log \mathbf{n}\right)$ times

To ensure high probability (**Chernoff**)

8: Our result:

- Achieve desired performance by combining the reporting / emptiness data structure (**Shallow partition tree**) with **relative-error approximations** at each node
- More versatile and modular
- Avoids the "Chernoff factor" $\mathbf{O}\left(\frac{1}{\varepsilon^2}\log n\right)$

(Sharper probabilistic analysis)

- Better dependence on ε; Storage independent of ε
- Better control: Lots of opportunities for fine-tuning

(Also a con: Messy recurrences aplenty...)

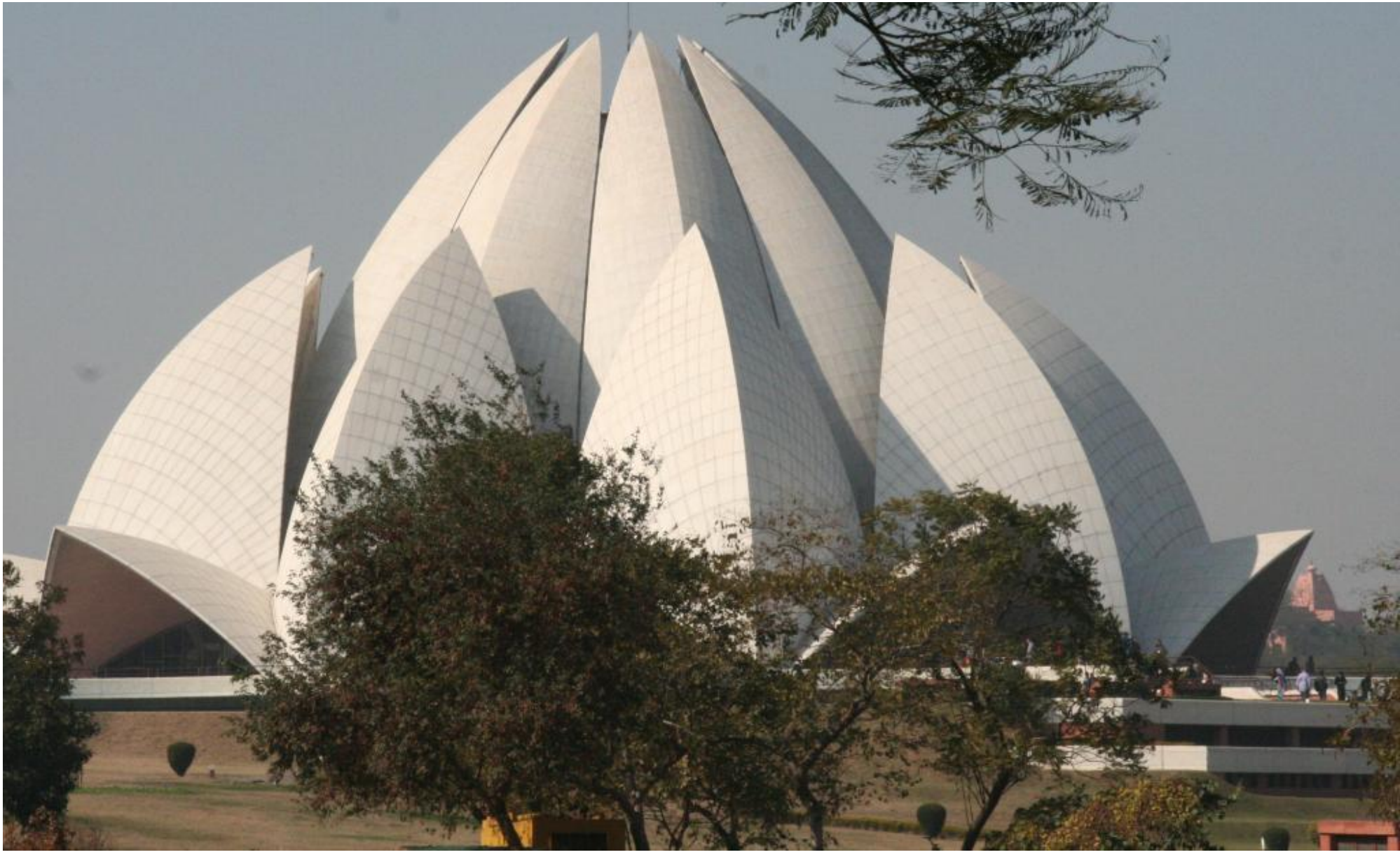
9: Approach (sketch)

- Use the shallow partition tree for emptiness / reporting
- At each node $\mathbf{v}$, store a relative-error approximation $\mathbf{A_v}$ of $\mathbf{P_v}$
- **Query with $\mathbf{h}$**: At a node $\mathbf{v}$ of the tree:
  Recurse if $\mathbf{h}$ crosses only a few children

$$\mathbf{O}\left(\mathbf{r_v^{1-1/\lfloor d/2 \rfloor}}\right)$$

- Otherwise $\mathbf{h}$ is not shallow: $|\mathbf{h} \cap \mathbf{P_v}| > |\mathbf{P_v}|/\mathbf{r_v}$
  $\Rightarrow$ Can use $|\mathbf{A_v} \cap \mathbf{h}|$ to estimate $|\mathbf{P_v} \cap \mathbf{h}|$

And now for something completely different - the middle of the talk



10: Points and Spanning Trees

Good spanning tree

P: **n** points

Find good spanning tree for P?

What is good?

A: Light, small diameter, small crossing number, and carrot.

10: Points and Spanning Trees

Good spanning tree

P: **n** points

Find good spanning tree for P?

What is good?

A: Light, small diameter, small crossing number, and carrot.

10: Points and Spanning Trees

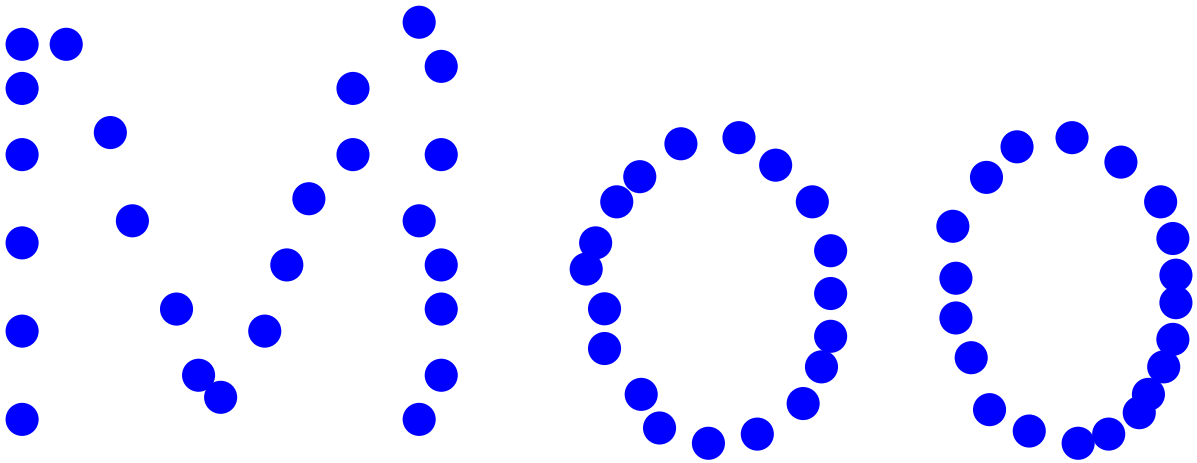
Good spanning tree

P: **n** points

Find good spanning tree for P?

What is good?

A: Light, small diameter, small crossing number, and carrot.



10: Points and Spanning Trees

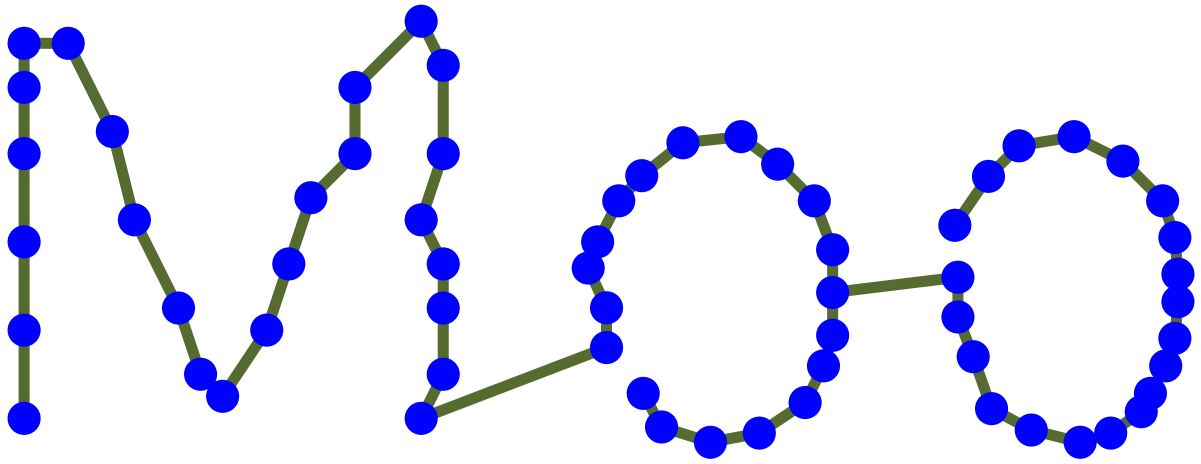
Good spanning tree

P: **n** points

Find good spanning tree for P?

What is good?

A: Light, small diameter, small crossing number, and carrot.



10: Points and Spanning Trees

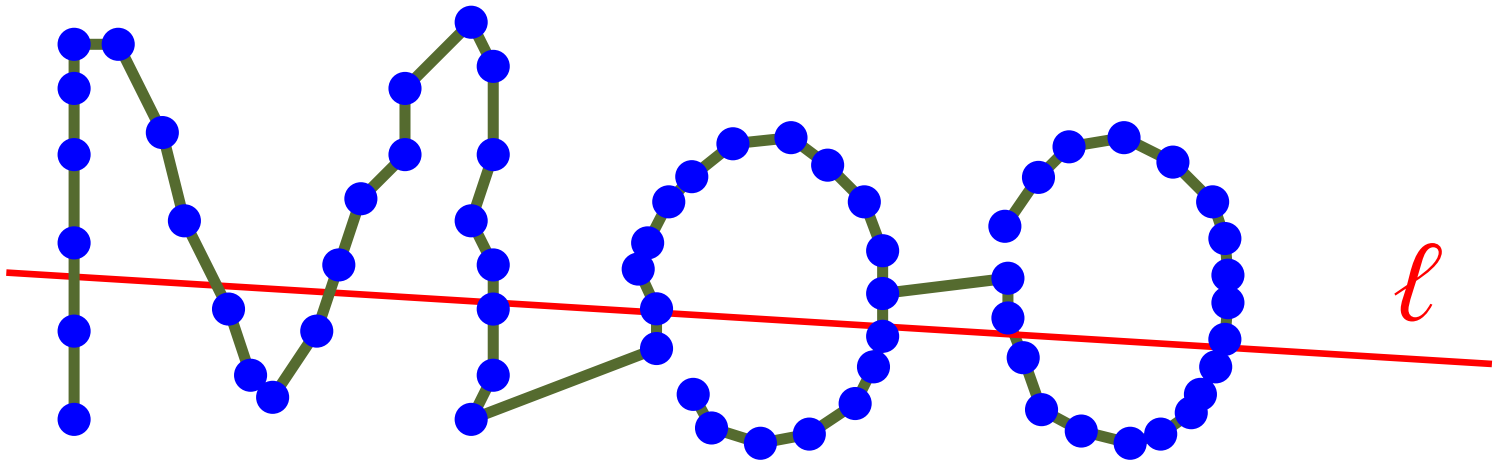
Good spanning tree

P: **n** points

Find good spanning tree for P?

What is good?

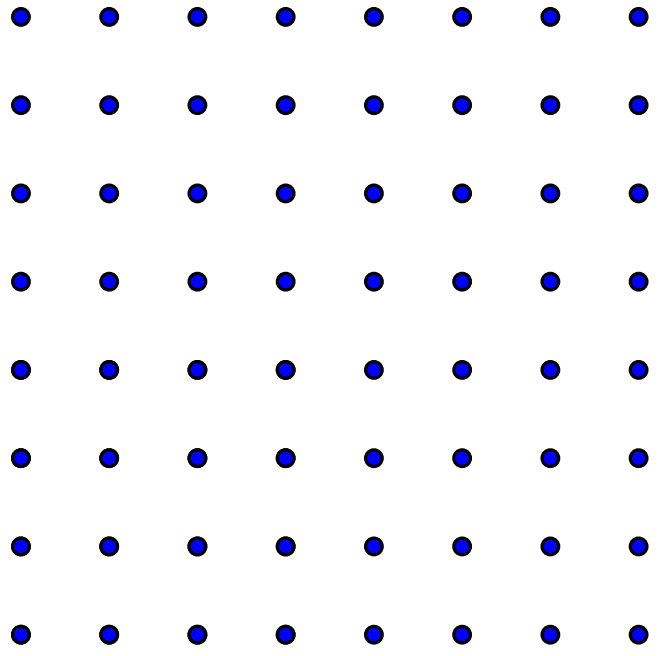
A: Light, small diameter, small crossing number, and carrot.



11: Spanning tree with low crossing number

[Welzl, 1992]

P: **n** points
T: spanning tree for **P**
Such that any line ℓ crosses
$\mathbf{O(\sqrt{n})}$ edges of **T**
Proof: Uses reweighting.
Mysterious.



Intuitively...

All point sets behave like they are on a grid...
[Matoušek, 1992a]: **Partition trees.**

11: Spanning tree with low crossing number

[Welzl, 1992]

P: **n** points

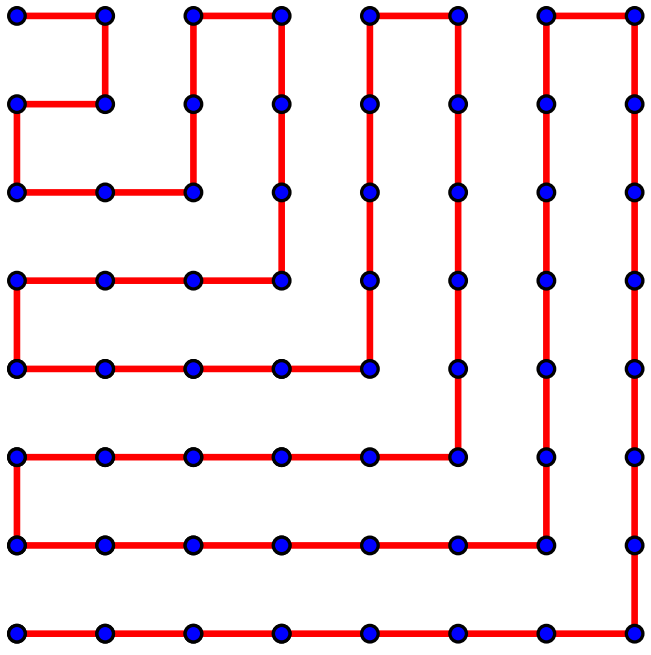
T: spanning tree for **P**

Such that any line ℓ crosses

$\mathbf{O(\sqrt{n})}$ edges of **T**

Proof: Uses reweighting.

Mysterious.



Intuitively...

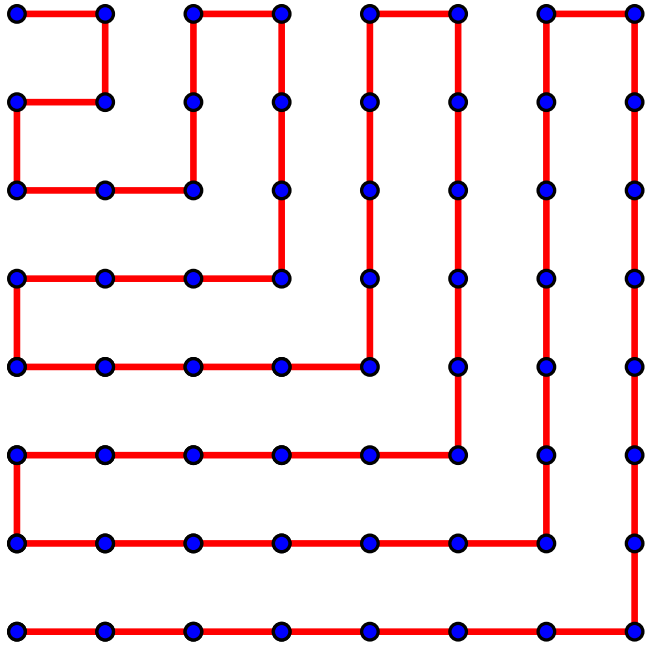
All point sets behave like they are on a grid...

[Matoušek, 1992a]: **Partition trees.**

11: Spanning tree with low crossing number

[Welzl, 1992]

P: **n** points
T: spanning tree for **P**
Such that any line ℓ crosses
$\mathbf{O(\sqrt{n})}$ edges of **T**
Proof: Uses reweighting.
Mysterious.



Intuitively...

All point sets behave like they are on a grid...
[Matoušek, 1992a]: **Partition trees.**

12: Spanning tree with low crossing number

Question

Can Welzl's result be improved?

13: Can Welzl's spanning trees be improved?

Spanning tree with low crossing number - cont'd

Yes!

14: New Result

Spanning tree with relative crossing number

Theorem

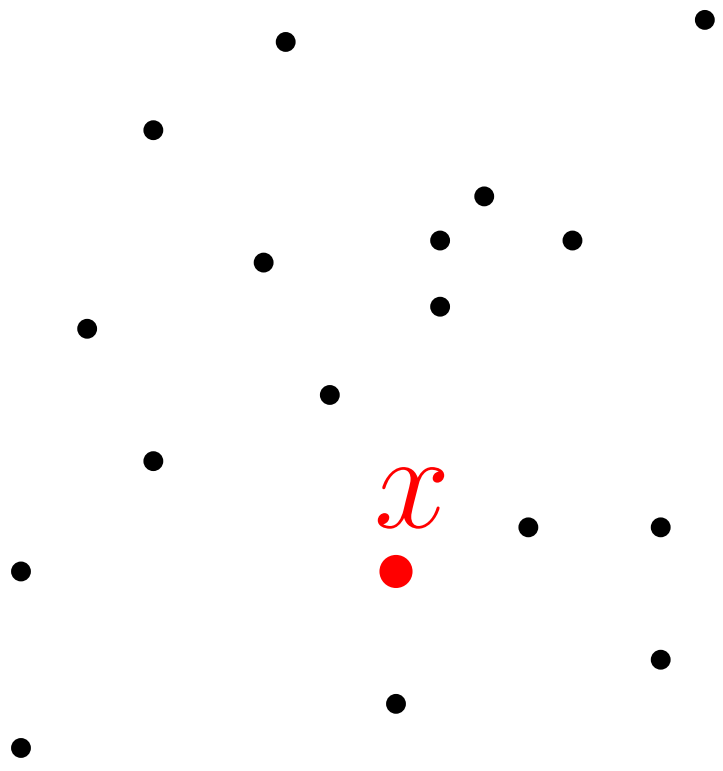
T: *relative spanning tree of* **P** *(set of* **n** *points).*
For any line ℓ *in the plane.*
$\mathbf{k} = \omega(\ell)$: *# points on one side of* ℓ.
$\Rightarrow \ell$ *crosses* $\mathbf{O}\left(\sqrt{\mathbf{k}}\log\frac{\mathbf{n}}{\mathbf{k}}\right)$

15: Half-Space Depth

Let us fix **k**...

Depth Contour

P: **n** points
x: a point.
Half-space depth of **x**:
Min number of points of **P** on a half plane of **x**.
$\mathcal{D}_k$: Contour of depth **k**.

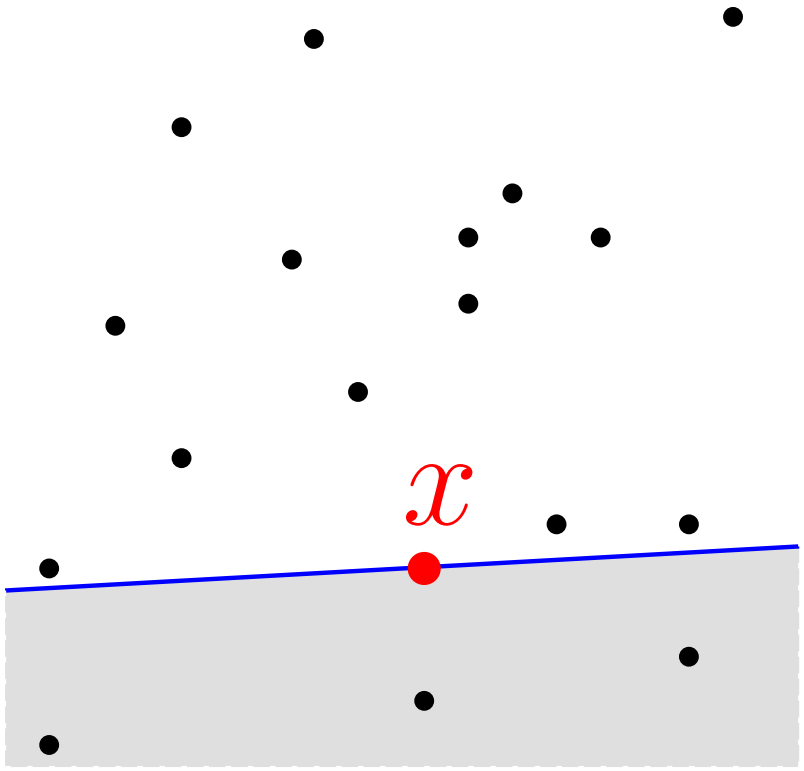


15: Half-Space Depth

Let us fix **k**...

Depth Contour

P: **n** points
x: a point.
Half-space depth of **x**:
Min number of points of **P** on a half plane of **x**.
$\mathcal{D}_k$: Contour of depth **k**.

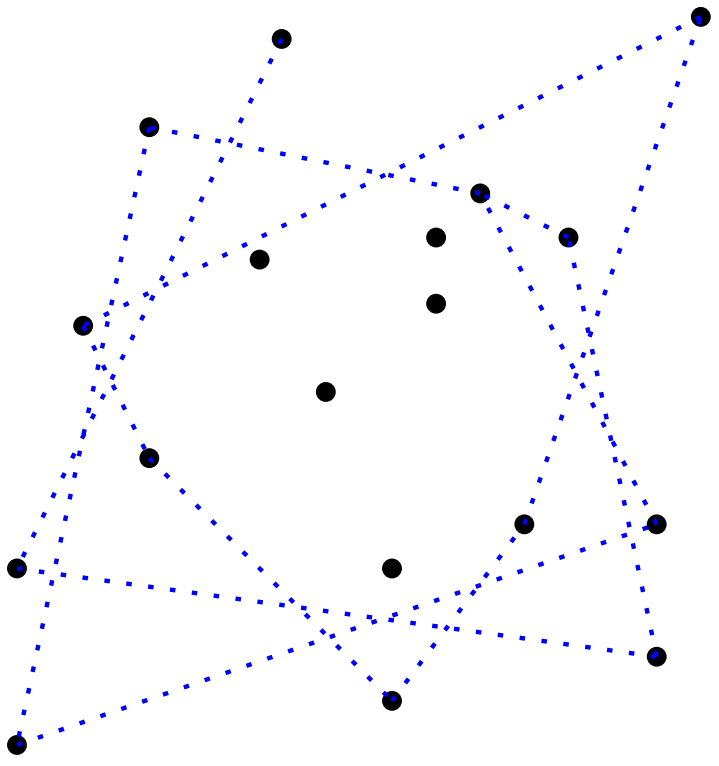


15: Half-Space Depth

Let us fix **k**...

Depth Contour

P: **n** points
x: a point.
Half-space depth of **x**:
Min number of points of **P** on a half plane of **x**.
$\mathcal{D}_k$: Contour of depth **k**.

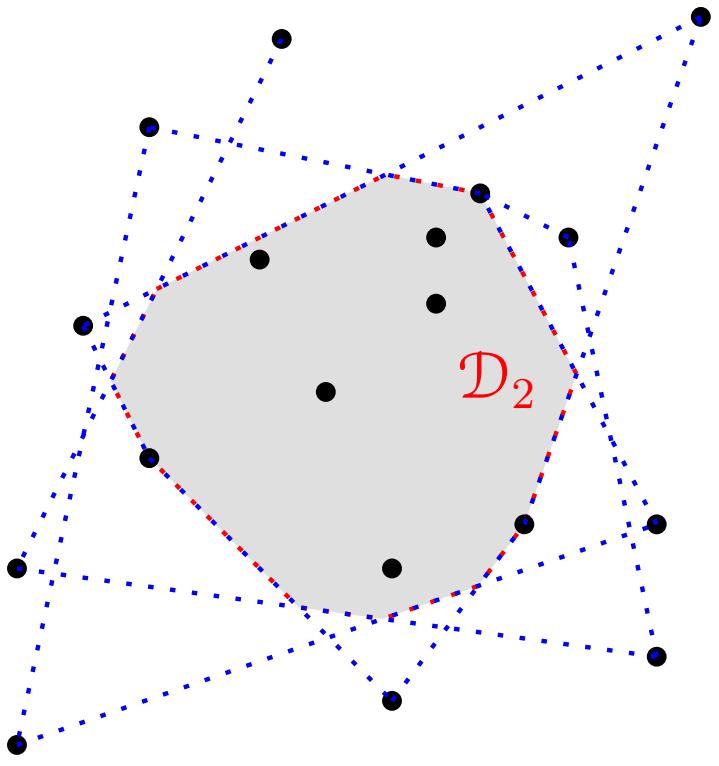


15: Half-Space Depth

Let us fix **k**...

Depth Contour

P: **n** points
x: a point.
Half-space depth of **x**:
Min number of points of **P** on a half plane of **x**.
$\mathcal{D}_k$: Contour of depth **k**.



16: Depth contour

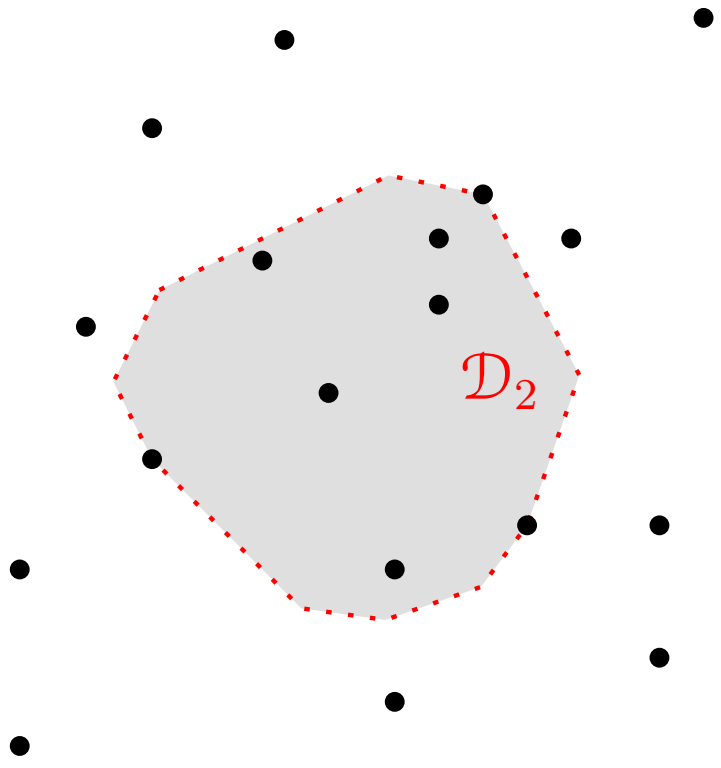
Fix **k**

Observations

ℓ a line

$\left|\ell^+ \cap \mathbf{P}\right| > \mathbf{2k} \quad \Rightarrow \quad \ell$ intersects $\mathcal{D}_\mathbf{k}$

$\left|\ell^+ \cap \mathbf{P}\right| \leq \mathbf{k} \quad \Rightarrow \quad \ell$ avoids $\mathcal{D}_\mathbf{k}$



$\mathcal{D}_\mathbf{k}$ approximates points of depth $\leq \mathbf{k}$.

Find spanning tree for $\mathbf{Q} = \mathbf{P} \setminus \mathcal{D}_\mathbf{k}$

16: Depth contour

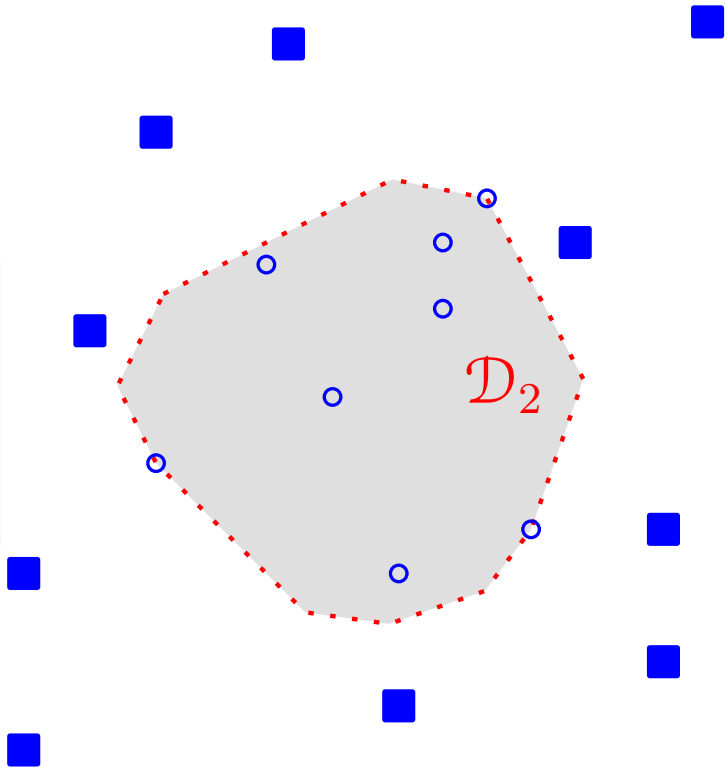
Fix $\mathbf{k}$

Observations

ℓ a line

$\left|\ell^+ \cap \mathbf{P}\right| > \mathbf{2k} \quad \Rightarrow \quad \ell$ intersects $\mathcal{D}_\mathbf{k}$

$\left|\ell^+ \cap \mathbf{P}\right| \leq \mathbf{k} \quad \Rightarrow \quad \ell$ avoids $\mathcal{D}_\mathbf{k}$



$\mathcal{D}_\mathbf{k}$ approximates points of depth $\leq \mathbf{k}$.

Find spanning tree for $\mathbf{Q} = \mathbf{P} \setminus \mathcal{D}_\mathbf{k}$

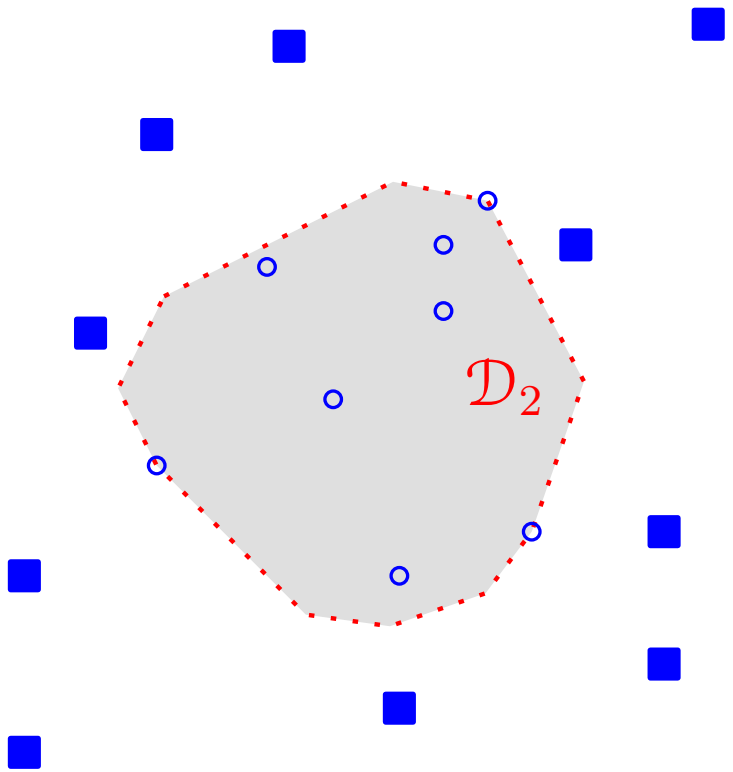
17: Half-Space Depth - cont'd

Idea:

$\mathbf{Q = P \setminus \mathcal{D}_k}$

Partition $\mathbf{Q}$ into sets of size $\mathbf{k}$:

- $\mathbf{S_1, \ldots S_m}$.
- Any line crosses few sets.
- $\mathbf{T_i} \Leftarrow$ good spanning tree for $\mathbf{S_i}$ **[Welzl, 1992]**
- Connect trees together into one big hairy tree.

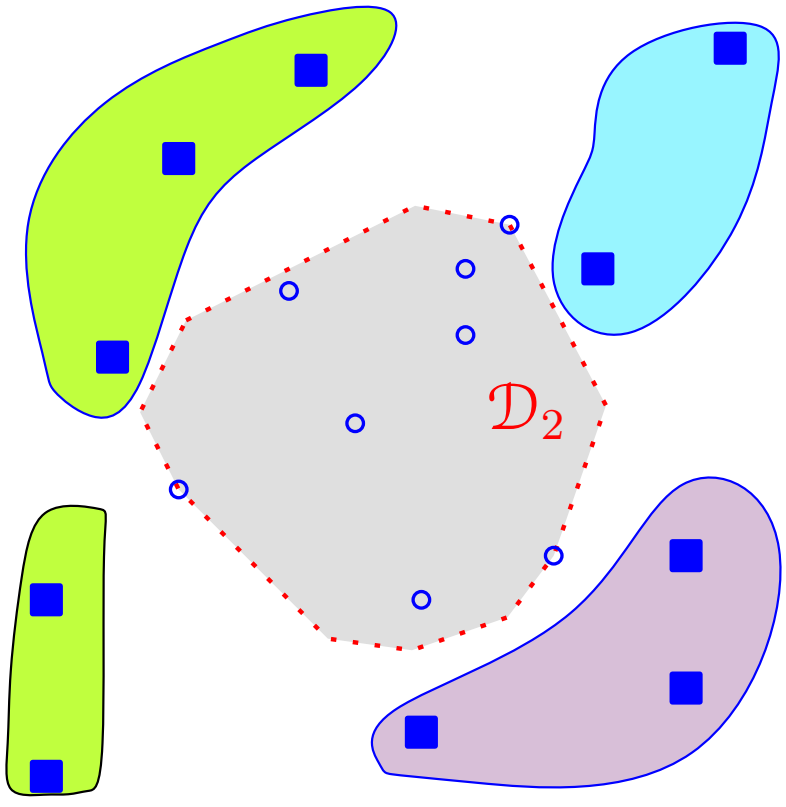


17: Half-Space Depth - cont'd

Idea:

$\mathbf{Q = P \setminus \mathcal{D}_k}$
Partition $\mathbf{Q}$ into sets of size $\mathbf{k}$:

- $\mathbf{S_1, \ldots S_m}$.
- Any line crosses few sets.
- $\mathbf{T_i} \Leftarrow$ good spanning tree for $\mathbf{S_i}$ **[Welzl, 1992]**
- Connect trees together into one big hairy tree.

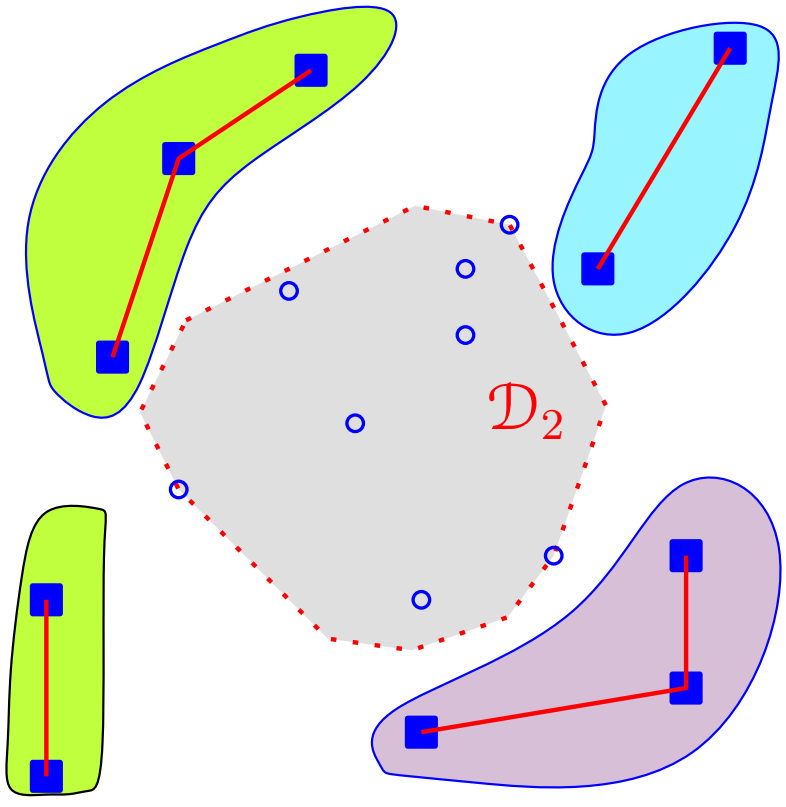


17: Half-Space Depth - cont'd

Idea:

$\mathbf{Q = P \setminus \mathcal{D}_k}$
Partition $\mathbf{Q}$ into sets of size $\mathbf{k}$:

- $\mathbf{S_1, \ldots S_m}$.
- Any line crosses few sets.
- $\mathbf{T_i} \Leftarrow$ good spanning tree for $\mathbf{S_i}$ **[Welzl, 1992]**
- Connect trees together into one big hairy tree.

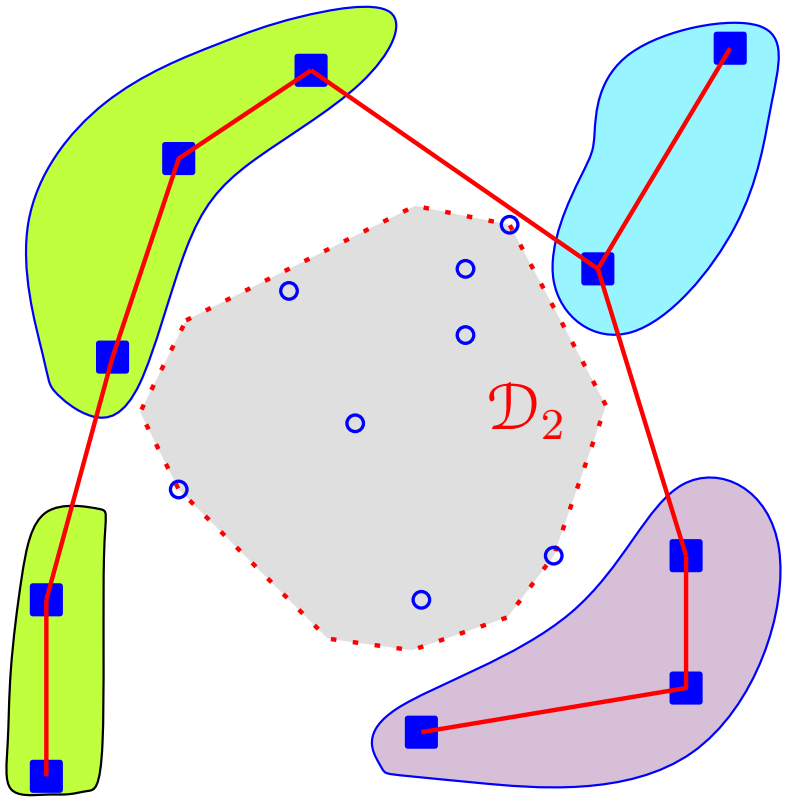


17: Half-Space Depth - cont'd

Idea:

$\mathbf{Q = P \setminus \mathcal{D}_k}$
Partition $\mathbf{Q}$ into sets of size $\mathbf{k}$:

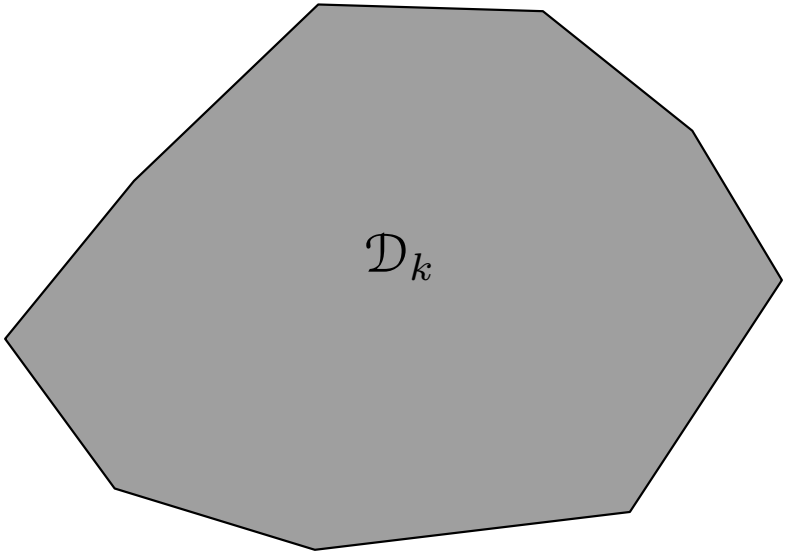
- $\mathbf{S_1, \ldots S_m}$.
- Any line crosses few sets.
- $\mathbf{T_i} \Leftarrow$ good spanning tree for $\mathbf{S_i}$ **[Welzl, 1992]**
- Connect trees together into one big hairy tree.



18: Half-Space Depth - cont'd

Construction

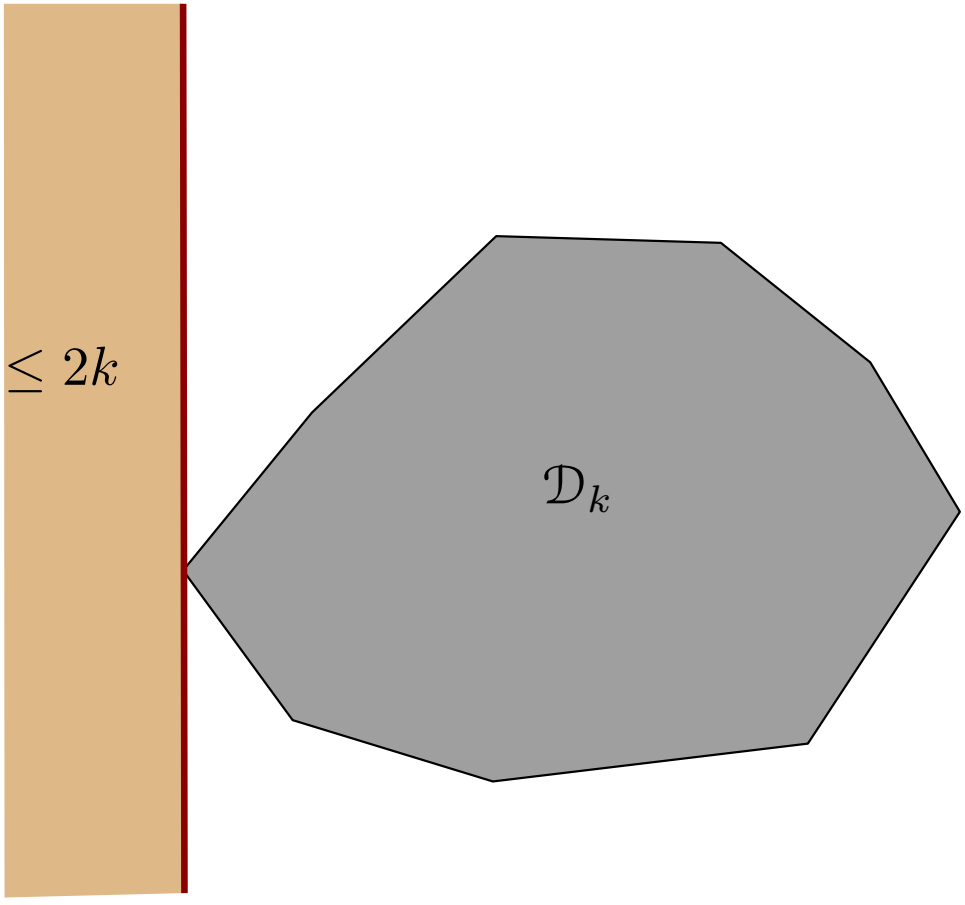
- How to generate sets?
- Dobkin-Kirkpatrik....
- Any line crosses $\mathbf{O\left(\log \frac{n}{k}\right)}$ sets.
- Any line crosses $\mathbf{O\left(\sqrt{k}\log(n/k)\right)}$ edges spanning tree.



18: Half-Space Depth - cont'd

Construction

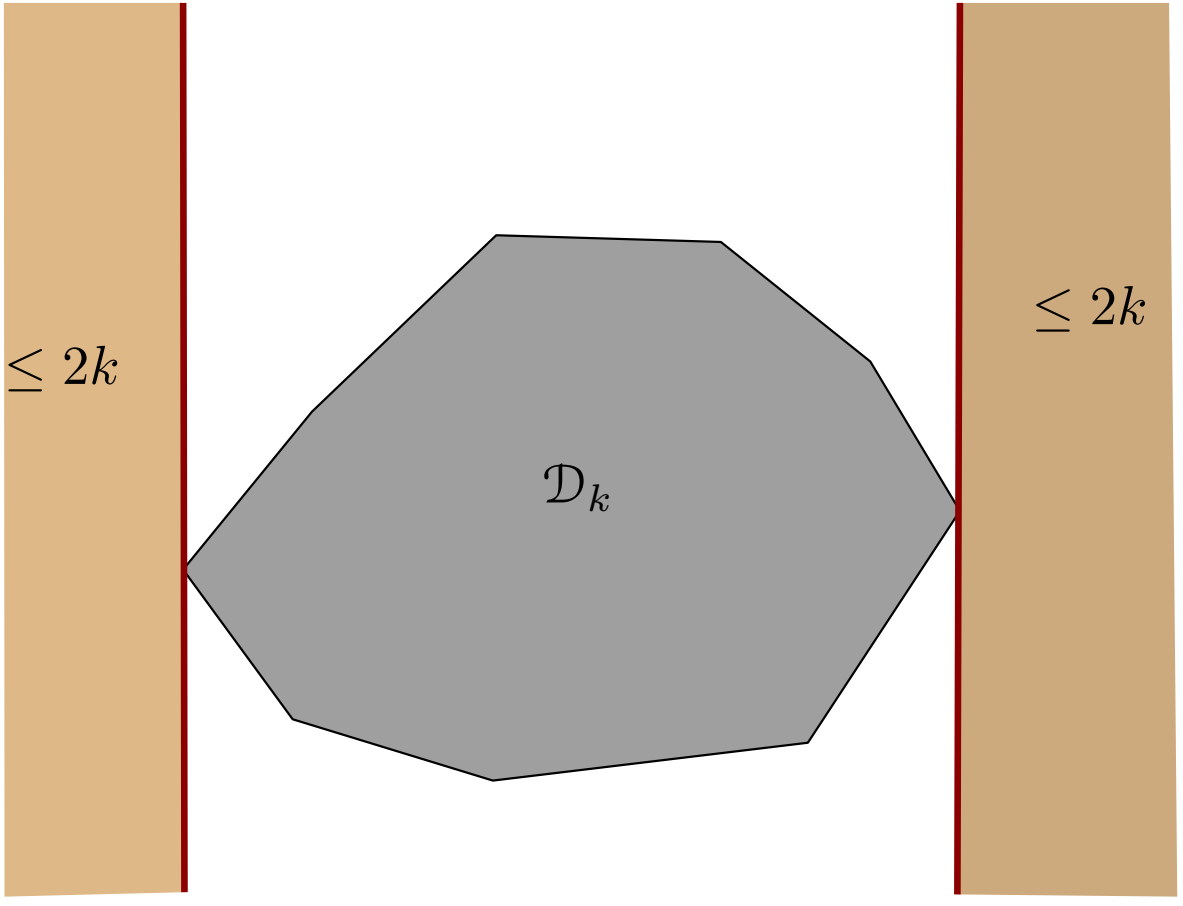
- How to generate sets?
- Dobkin-Kirkpatrik....
- Any line crosses $\mathbf{O\left(\log \frac{n}{k}\right)}$ sets.
- Any line crosses $\mathbf{O\left(\sqrt{k}\log(n/k)\right)}$ edges spanning tree.



18: Half-Space Depth - cont'd

Construction

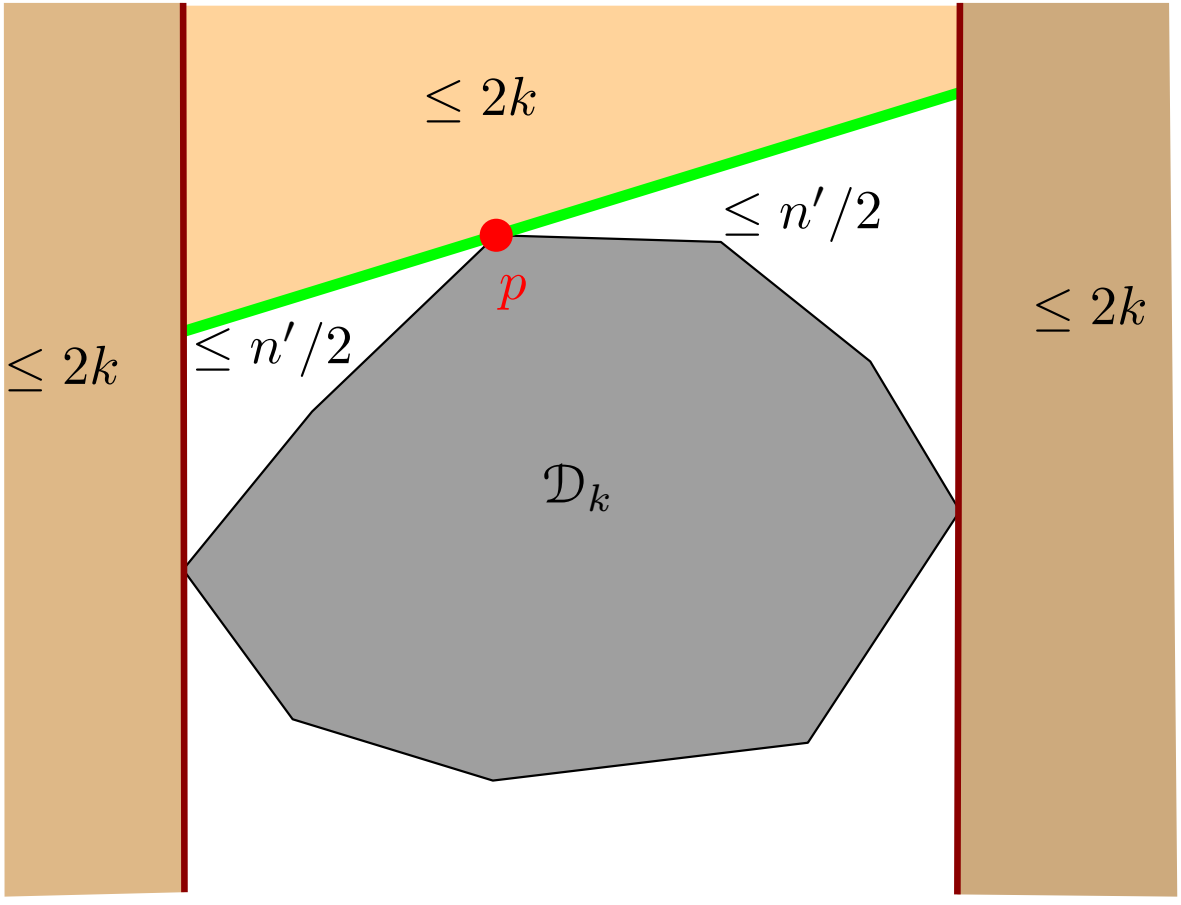
- How to generate sets?
- Dobkin-Kirkpatrik....
- Any line crosses $\mathbf{O\left(\log \frac{n}{k}\right)}$ sets.
- Any line crosses $\mathbf{O\left(\sqrt{k}\log(n/k)\right)}$ edges spanning tree.



18: Half-Space Depth - cont'd

Construction

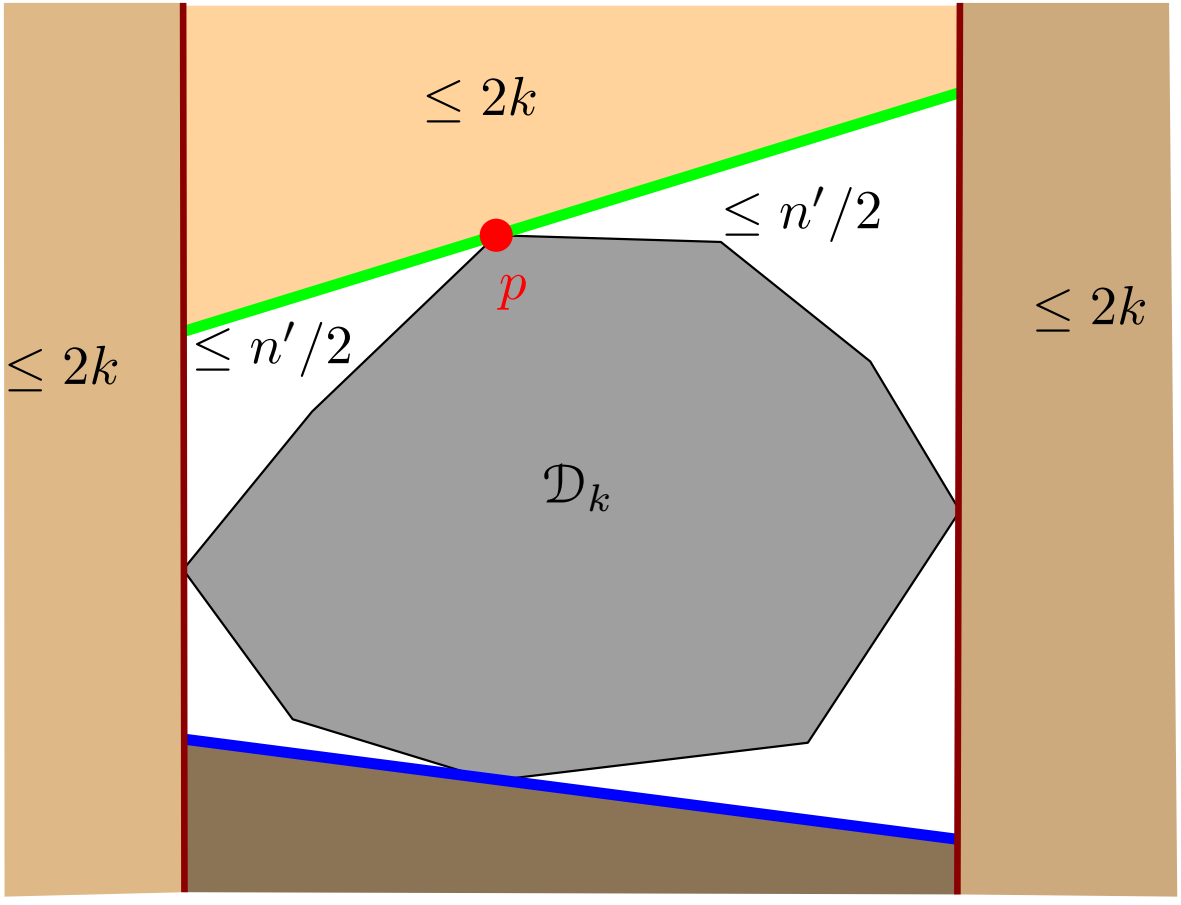
- How to generate sets?
- Dobkin-Kirkpatrik....
- Any line crosses $\mathbf{O\left(\log \frac{n}{k}\right)}$ sets.
- Any line crosses $\mathbf{O\left(\sqrt{k}\log(n/k)\right)}$ edges spanning tree.



18: Half-Space Depth - cont'd

Construction

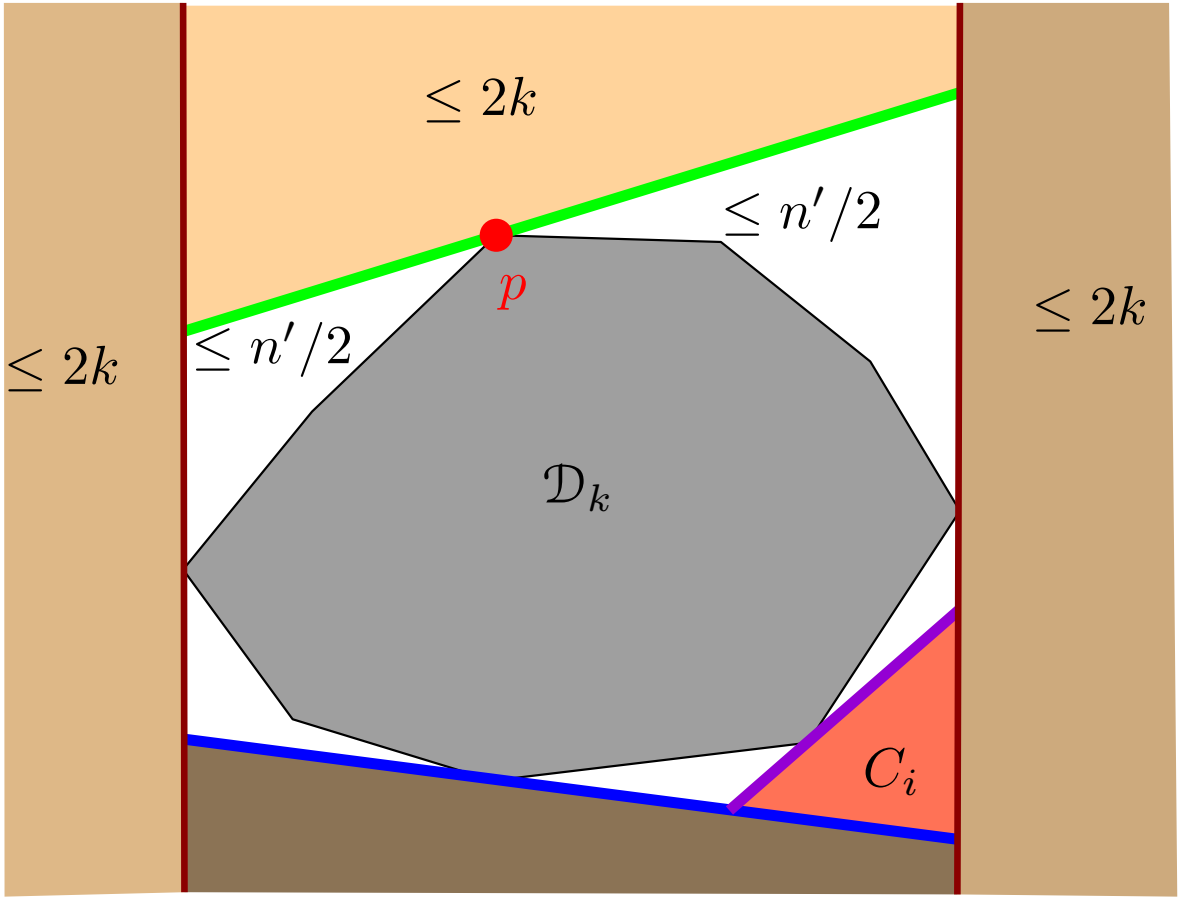
- How to generate sets?
- Dobkin-Kirkpatrik....
- Any line crosses $\mathbf{O\left(\log \frac{n}{k}\right)}$ sets.
- Any line crosses $\mathbf{O\left(\sqrt{k}\log(n/k)\right)}$ edges spanning tree.



18: Half-Space Depth - cont'd

Construction

- How to generate sets?
- Dobkin-Kirkpatrik....
- Any line crosses $\mathbf{O\left(\log \frac{n}{k}\right)}$ sets.
- Any line crosses $\mathbf{O\left(\sqrt{k}\log(n/k)\right)}$ edges spanning tree.



19: If k is not known...

Onion peeling.

Algorithm

Set $k_i = 2^i$. $P_0 = P$, and $i = 1$.
$Q_i = P_{i-1} \setminus \mathcal{D}(P_{i-1}, k_i)$
$T_i \Leftarrow$ spanning tree of Q_i.
$P_i = P_{i-1} \setminus Q_i$.

$T =$ connect $T_1, T_2 \ldots$ together.
Crossing number of T, for a line of weight k:

$$\sum_{i}^{\lg k} O\left(\sqrt{2^i \log(n/2^i)}\right) = O(\sqrt{k \log(n/k)}),$$

20: Conclusions

- A lot of other results in the paper.
- **Open:** Extend sensitive spanning tree to higher dimensions.
- **Open:** Find an alt alg/proof of the Welzl **[Welzl, 1992]** result.
  (Related to partitions.)
- **Open:** Better understanding of discrepancy.


Aronov, B. and Har-Peled, S. (2005).
On approximating the depth and related problems.
In *Proc. 16th ACM-SIAM Sympos. Discrete Algorithms*, pages 886–894.


Cohen, E. (1997).
Size-estimation framework with applications to transitive closure and reachability.
J. Comput. Sys. Sci., 55(3):441–453.


Kaplan, H. and Sharir, M. (2006).
Randomized incremental constructions of three-dimensional convex hulls and planar voronoi diagrams, and approximate range counting.
In *Proc. 17th ACM-SIAM Sympos. Discrete Algorithms*, pages 484–493.


Matoušek, J. (1992a).
Efficient partition trees.

Discrete Comput. Geom., 8:315–334.


Matoušek, J. (1992b).
Reporting points in halfspaces.
Comput. Geom. Theory Appl., 2(3):169–186.


Welzl, E. (1992).
On spanning trees with low crossing numbers.
In *Data Structures and Efficient Algorithms, Final Report on the DFG Special Joint Initiative*, volume 594 of *Lect. Notes in Comp. Sci.*, pages 233–249. Springer-Verlag.