

# Approximation algorithms for geometric intersection graphs

Thomas Erlebach

Eidgenössische Technische Hochschule Zürich

`erlebach@tik.ee.ethz.ch`

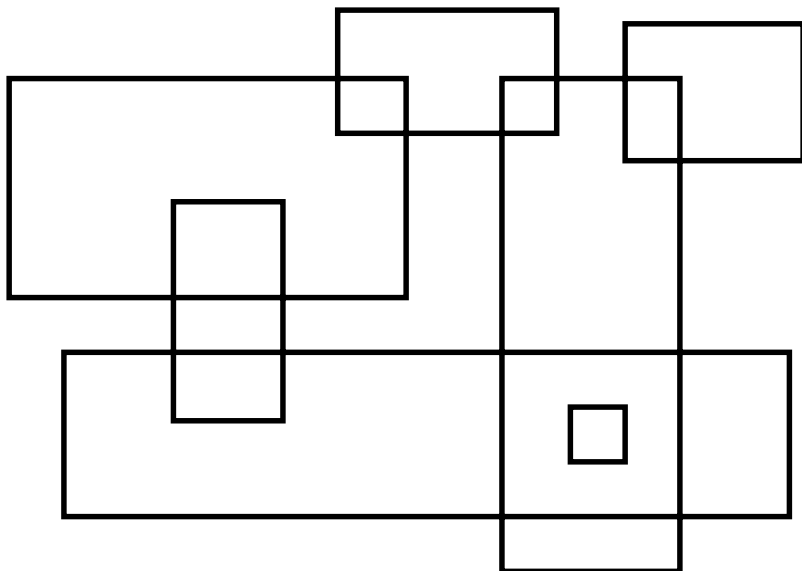
- ➊ Introduction
- ➋ Independent sets in disk graphs
- ➌ Vertex cover in disk graphs
- ➍ Coloring disk graphs
- ➎ Independent sets in rectangle graphs

# What are geometric intersection graphs?

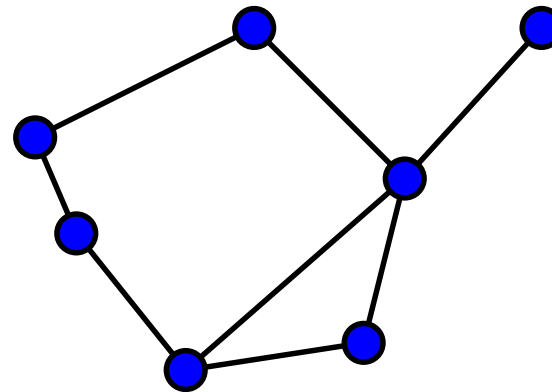
☞ **vertices** = geometric objects

☞ **edges** = non-empty **intersection** between objects

## Example: a rectangle intersection graph



geometric representation



intersection graph

# Popular geometric intersection graphs

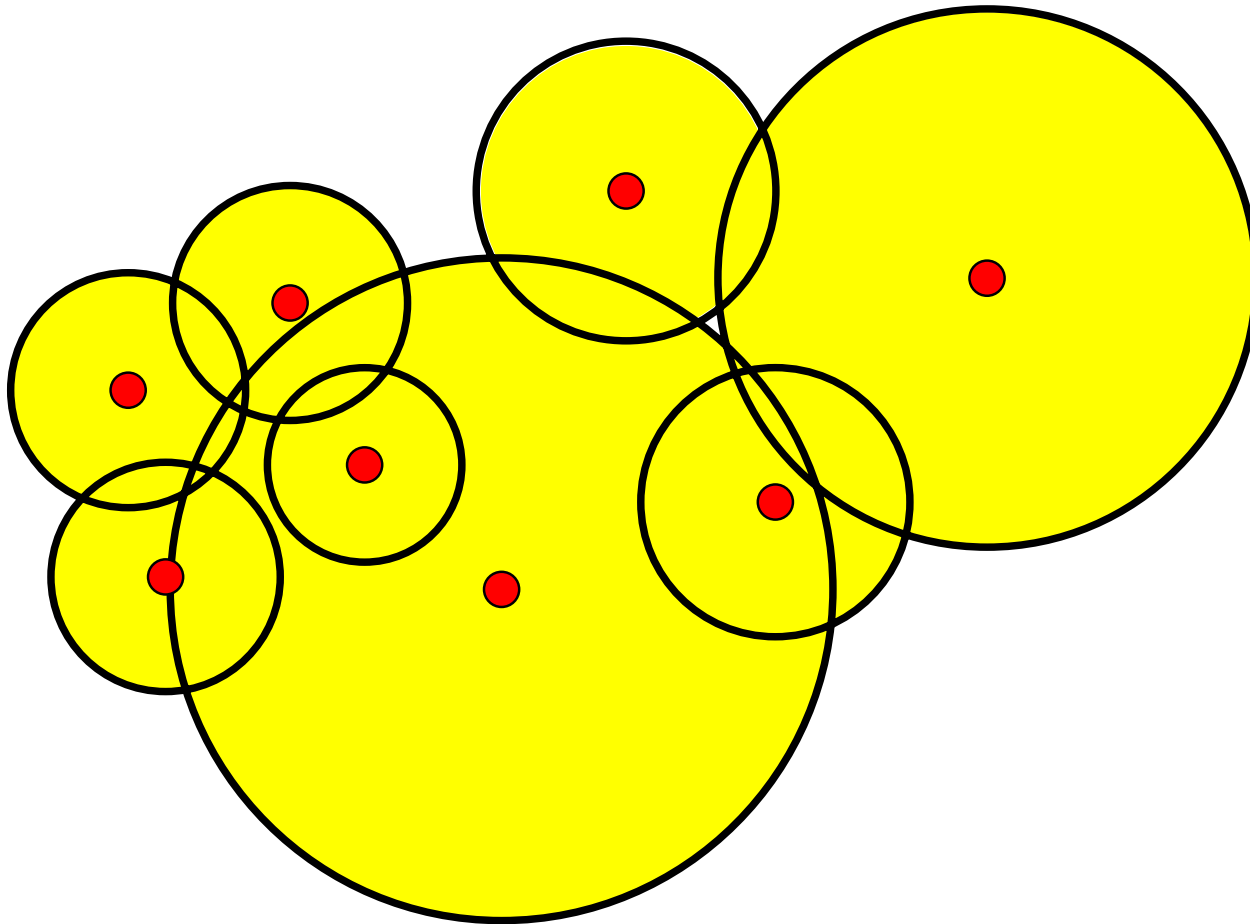
- ☐ disks (→ **disk graphs**), ellipses
- ☐ rectangles, squares (axis-aligned), arbitrary convex objects
- ☐ line segments, curves, higher-dimensional objects

The **recognition problem is usually  $\mathcal{NP}$ -hard!!**

## Some Applications:

- ⇒ Wireless networks (frequency assignment problems)
- ⇒ Map labeling
- ⇒ Admission control in chain networks with advance reservation

## Application: Wireless networks

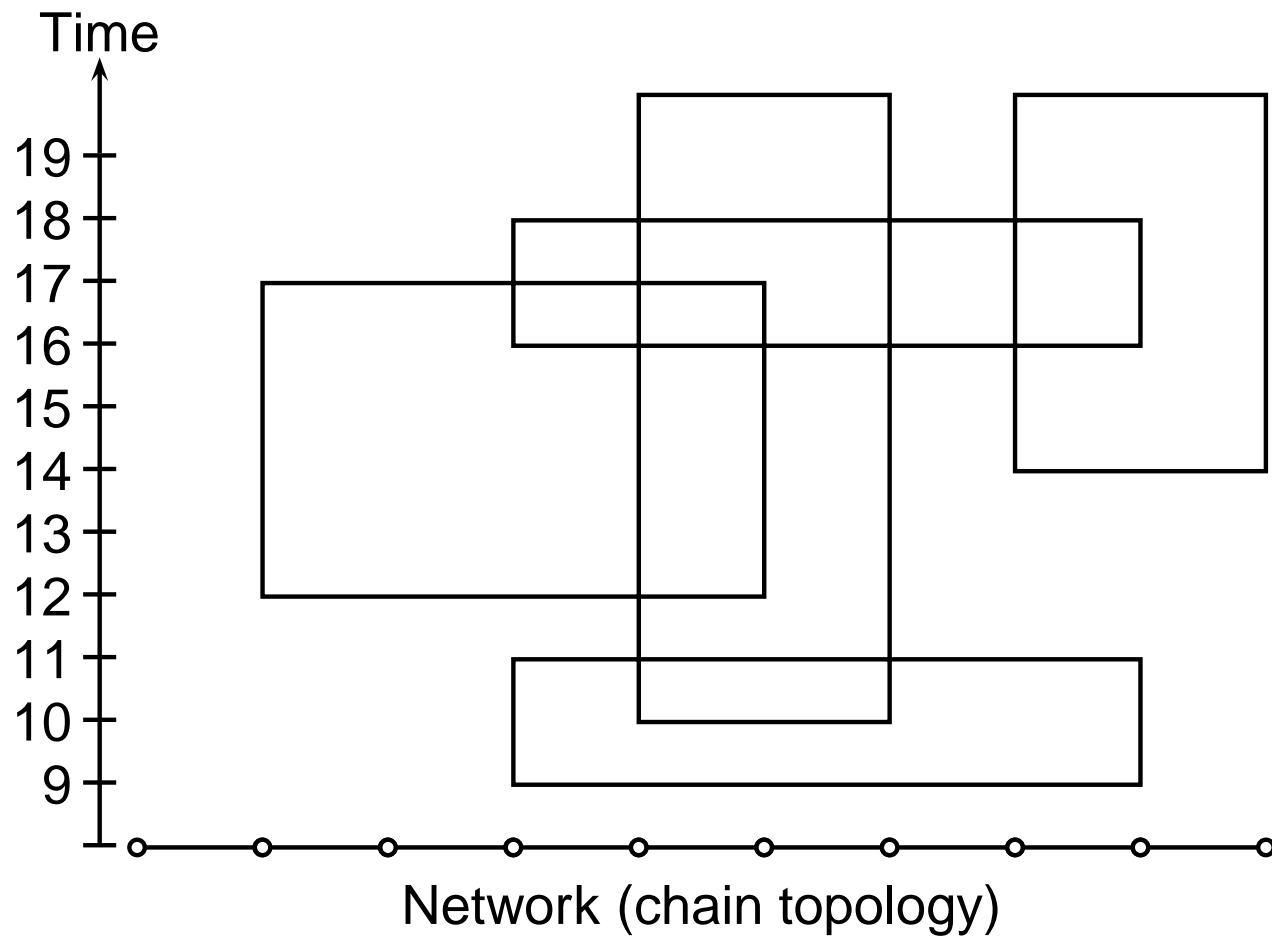


## Application: Map labeling



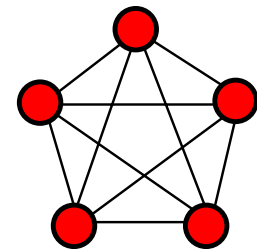
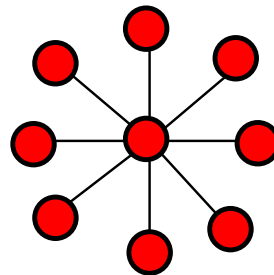
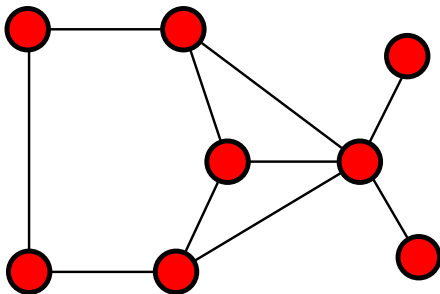
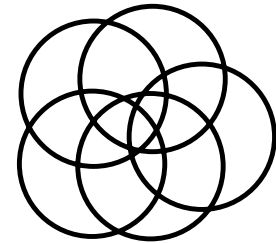
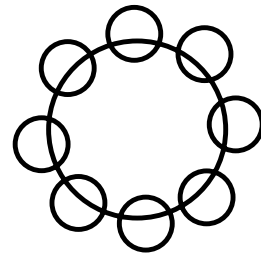
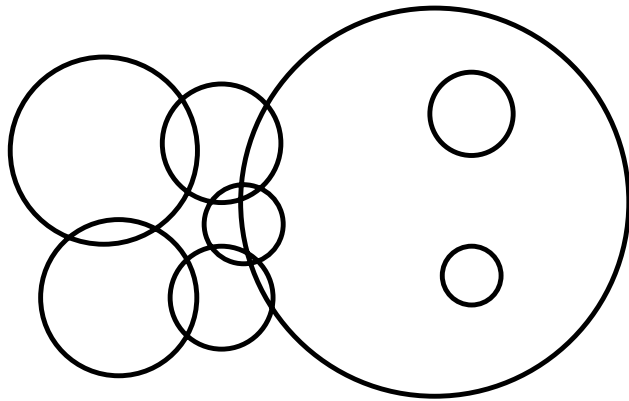
(taken from a paper by van Kreveld, Strijk, Wolff)

## Application: Call admission control



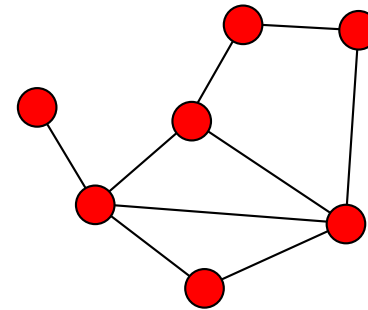
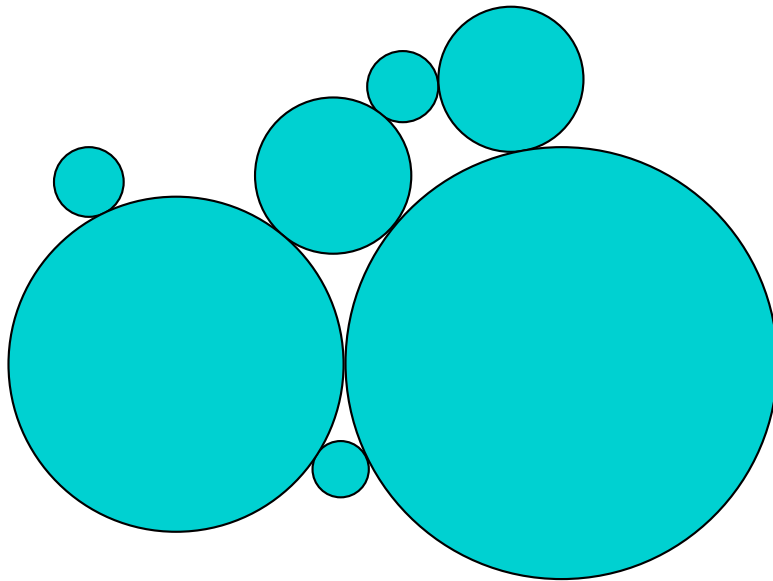
# Disk graphs

...are intersection graphs of disks in the plane:



## Subclasses of disk graphs

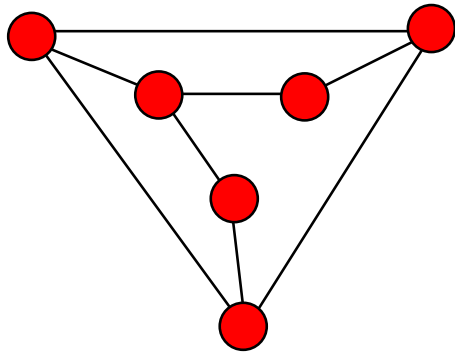
- ✿ **Unit disk graphs:** all disks have diameter 1
- ✿ **Coin graphs:** touching graphs of disks whose interiors are disjoint



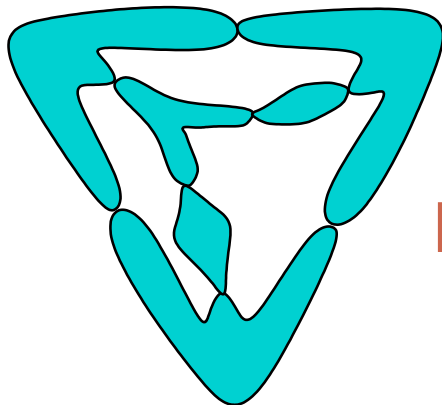
All coin graphs are planar, but surprisingly ...

# ... every planar graph is a coin graph

planar graph:

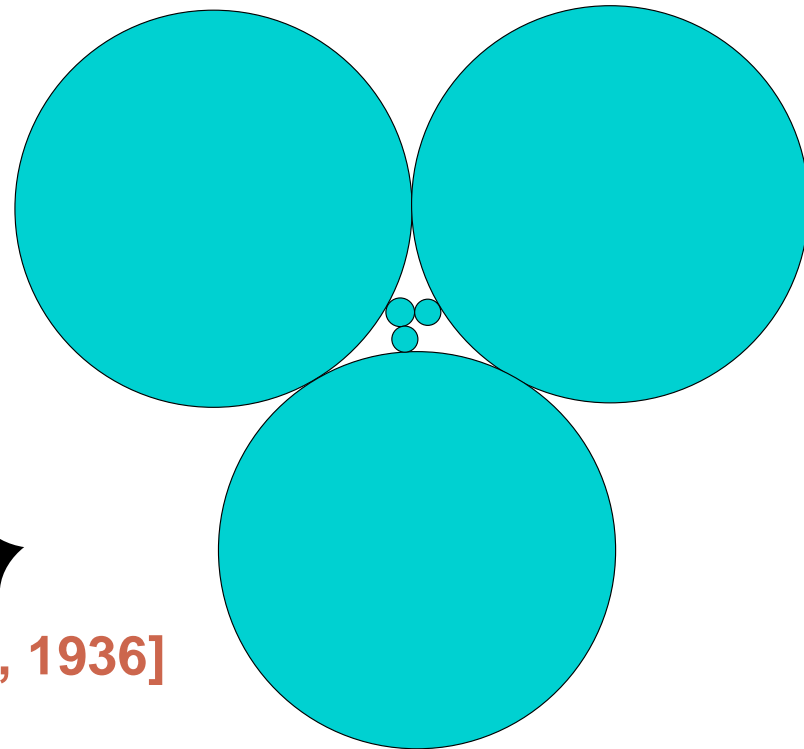


touching graph of "blobs":



[Koebe, 1936]

touching graph of disks:

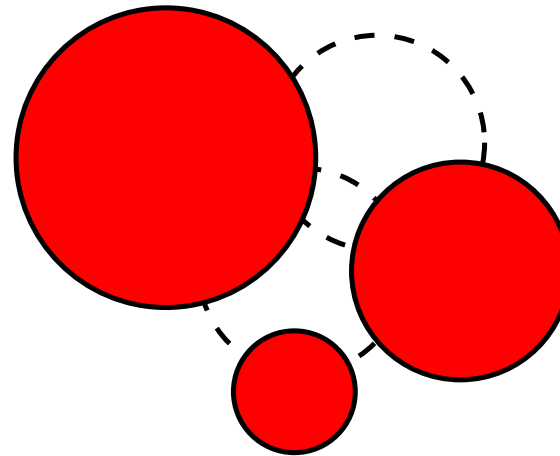
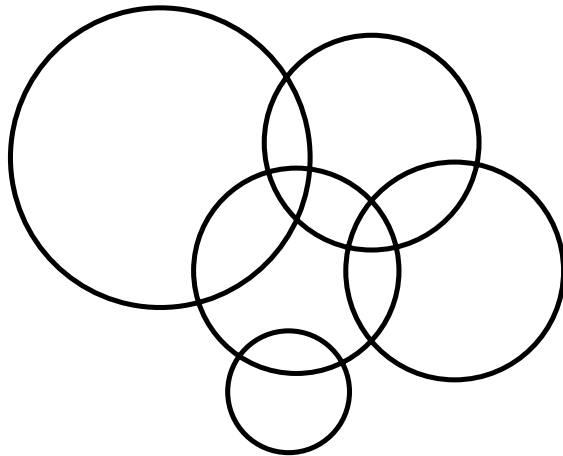


# The problem MAXINDEPENDENTSET

**Input:** a set  $\mathcal{D}$  of disks in the plane

**Feasible solution:** subset  $A \subseteq \mathcal{D}$  of disjoint disks

**Goal:** maximize  $|A|$



# Approximation algorithms for MAXINDEPENDENTSET

An algorithm for MAXINDEPENDENTSET is a  $\rho$ -**approximation algorithm** if it

- runs in **polynomial time** and
- always outputs an independent set of **size at least**  $OPT/\rho$ , where  $OPT$  is the size of the optimal independent set.

A **polynomial-time approximation scheme (PTAS)** is a family of  $(1 + \varepsilon)$ -approximation algorithms for every constant  $\varepsilon > 0$ .

# MAXINDEPENDENTSET in unit disk graphs

The problem is  $\mathcal{NP}$ -hard [Clark, Colbourn, Johnson'90].

Let's try the **greedy algorithm**:

## Algorithm GREEDY

$I = \emptyset$ ;

**for** all given disks  $D$  **do**

**if**  $D$  is disjoint from the disks in  $I$  **then**

$I = I \cup \{D\}$ ;

**return**  $I$ ;

## Analysis of the greedy algorithm

- ① Compare the greedy solution  $I$  with the optimal solution  $I^*$ .
- ② “Charge” every disk in  $I^*$  to a disk in  $I$ .
- ③ Bound the number of disks charged to the same disk in  $I$ .

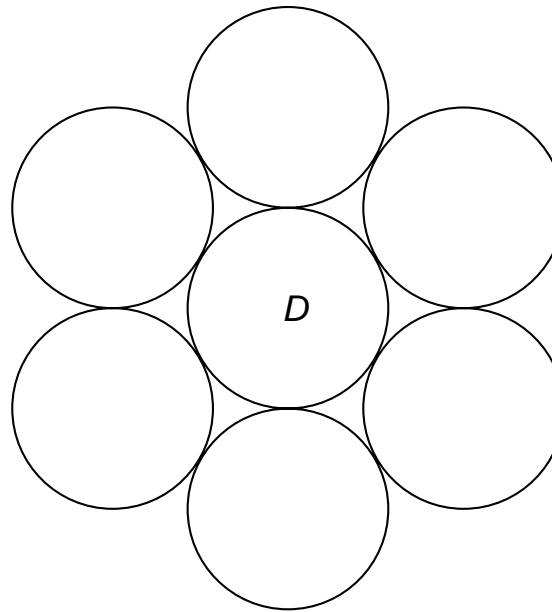
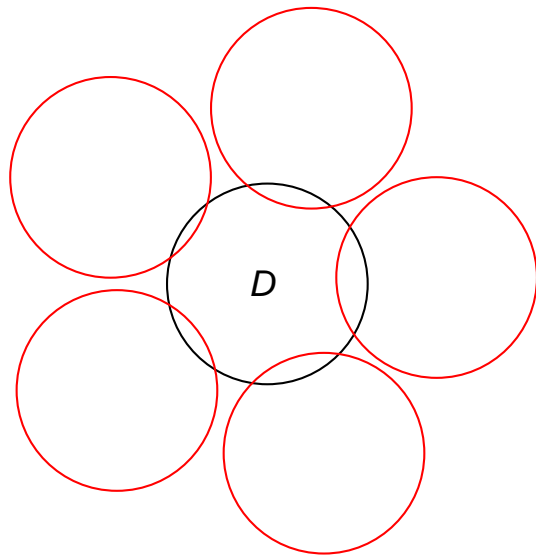
### Charging rules for a disk $D \in I^*$ :

- ⇒ If  $D$  is in  $I$ , charge  $D$  to itself.
- ⇒ If  $D$  is not in  $I$ , then charge it to any disk that intersects  $D$  and was accepted by GREEDY before it processed  $D$ .

## How often can a disk $D$ in $I$ be charged?

If  $D$  is also in  $I^*$ ,  $D$  is charged only once.

If  $D$  is not in  $I^*$ , it is charged by disks in  $I^*$  that intersect  $D$ . These disks are disjoint, so there can be at most 5 such disks:



➔  $|I^*| \leq 5|I|$  and **GREEDY is a 5-approximation algorithm.**

## An improved greedy algorithm

### Algorithm LEFTMOST-GREEDY

$I = \emptyset$ ;

**for** all given disks  $D$  **in order of increasing  $x$ -value** **do**

**if**  $D$  is disjoint from the disks in  $I$  **then**

$I = I \cup \{D\}$ ;

**return**  $I$ ;

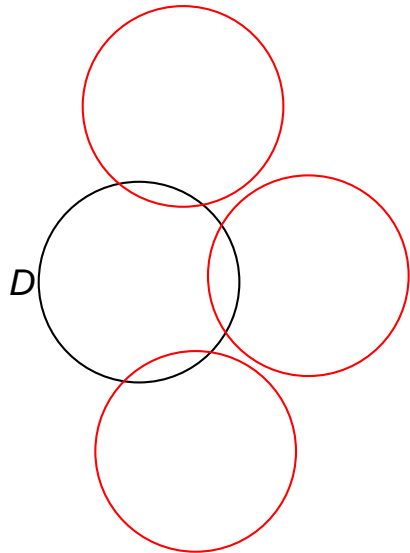
**Claim.** LEFTMOST-GREEDY is a **3-approximation algorithm** for MAXINDEPENDENTSET in unit disk graphs.

## Analysis of LEFTMOST-GREEDY

Use the same charging argument.

**Note:** A disk  $D$  in  $I$  receives charge from disks in  $I^*$  that are processed **after**  $D$  by LEFTMOST-GREEDY.

Therefore, each disk is charged at most three times:



## Do we need the representation?

**GREEDY did not need to know the representation, but what about LEFTMOST-GREEDY?**

For getting ratio 3 we needed only the following:

When a disk  $D$  is selected, the disks intersecting  $D$  that are processed later contain at most three disjoint disks.

↳ We can still get ratio 3 if we can identify **a disk whose neighborhood does not contain four disjoint disks!**

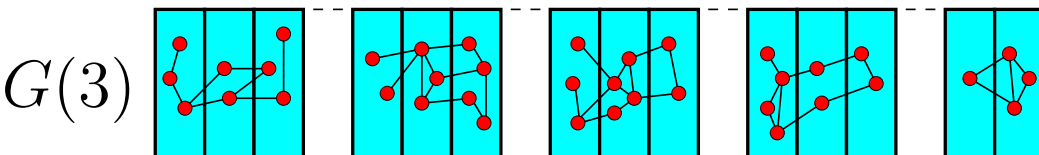
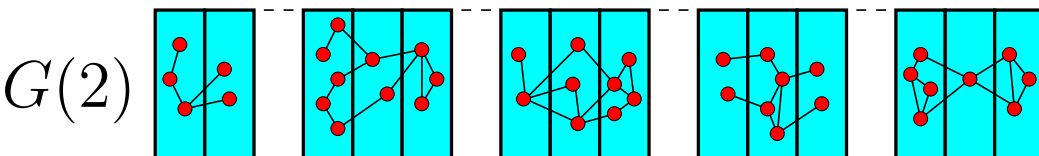
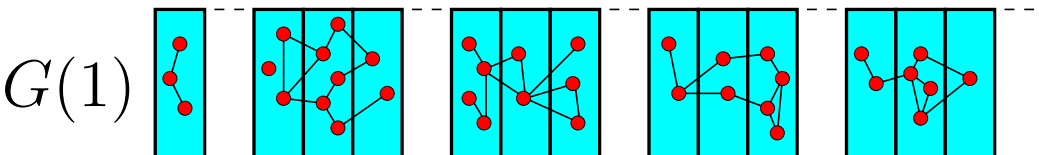
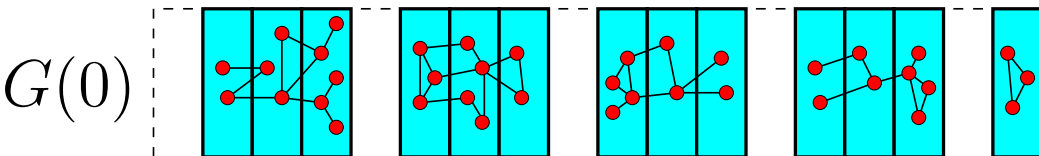
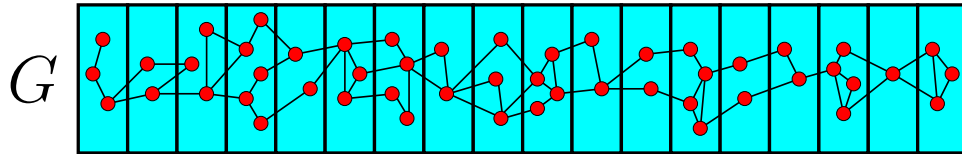
## LEFTMOST-GREEDY without given representation

Given a graph  $G = (V, E)$  that is the intersection graph of unit disks, the following is a **3-approximation algorithm for MAXINDEPENDENTSET**:

```
 $I = \emptyset;$   
repeat  
   $v =$  a vertex whose neighborhood does not have 4 independent vertices;  
   $I = I \cup \{v\};$   
  delete  $v$  and its neighbors from the graph;  
until the graph is empty;  
return  $I;$ 
```

The vertex  $v$  can be found in  $O(|V|^5)$  time.

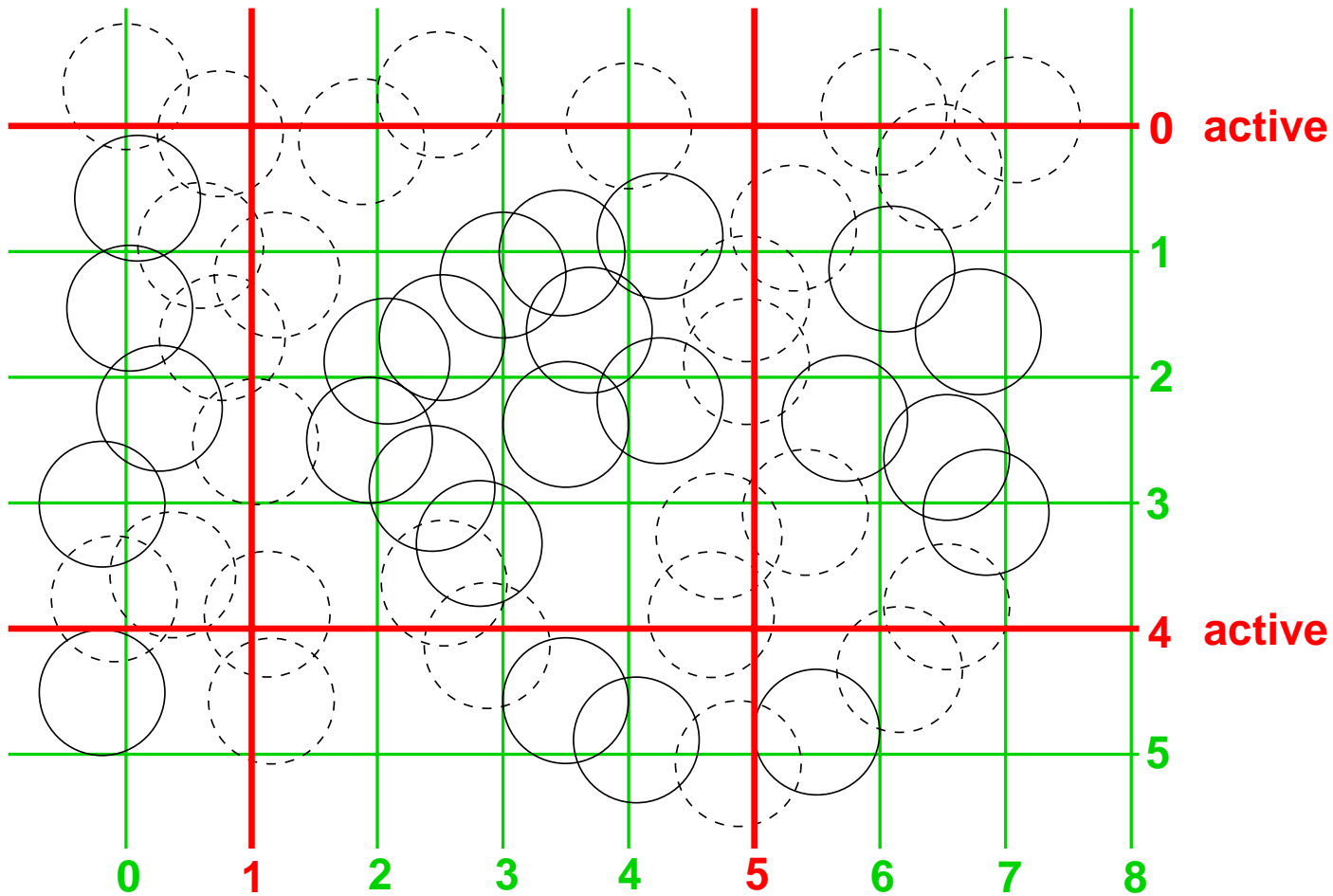
# The shifting strategy [Baker, 1984; Hochbaum and Maass, 1985]



- ❶ Partition graph into **slices**.
- ❷ Let  $k > 0$  be a fixed integer.
- ❸ Remove slices equal to  $\ell$  modulo  $k$  and compute a maximum independent set in the graph  $G(\ell)$ ,  $0 \leq \ell < k$ .
- ❹ Output the largest set found in this way.

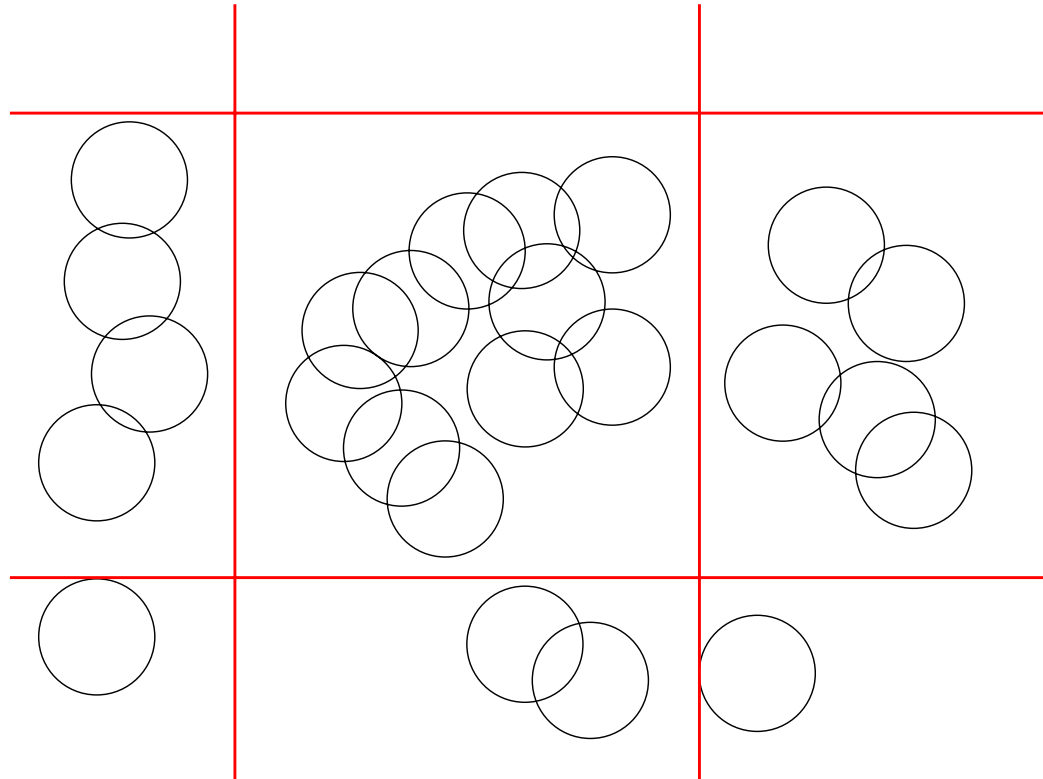
The largest of these sets contains at least  $(1 - \frac{1}{k}) OPT$  vertices.

# Shifting for unit disk graphs [Hochbaum and Maass, 1985]



Remove disks hitting **active lines** (and shift active lines).

Active lines partition the plane into squares that can be considered independently:



↳ Compute maximum independent set  $I$  in each square by **brute-force enumeration**. Since  $|I| = O(k^2)$ , **time**  $n^{O(k^2)}$  suffices.

## The PTAS for MAXINDEPENDENTSET

- 1 For  $0 \leq r, s < k$ , get  $\mathcal{D}(r, s)$  from  $\mathcal{D}$  by deleting disks that
    - hit a horizontal line equal to  $r$  modulo  $k$  or
    - hit a vertical line equal to  $s$  modulo  $k$ .
  - 2 Compute the maximum independent set  $I_S$  in each  $k \times k$  square  $S$  of  $\mathcal{D}(r, s)$  by brute-force enumeration.
  - 3 The union of the sets  $I_S$  gives a **maximum independent set in  $\mathcal{D}(r, s)$** .
  - 4 **Output the largest independent set** obtained in this way.
- 

**Running-time:**  $n^{O(k^2)}$  for  $n$  disks. (Can be improved to  $n^{O(k)}$ .)

**Approximation:** Computed solution has size **at least**  $(1 - \frac{2}{k}) OPT$ .

## MAXINDSET in unit disk graphs: Summary

- ▶▶▶ GREEDY gives a 5-approximation.  
[Marathe et al., 1995]
- ▶▶▶ LEFTMOST-GREEDY gives a 3-approximation. There is a variant that does not need the representation.  
[Marathe et al., 1995]
- ▶▶▶ The shifting strategy gives a PTAS. It needs the representation.  
[Hochbaum and Maass, 1985; Hunt III et al., 1998]

## MAXINDSET in general disk graphs

- ❖ The approximation ratio of GREEDY is only  $|V| - 1$ .
- ❖ But it helps to process the disks in the right order:

### Algorithm SMALLEST-GREEDY

$I = \emptyset$ ;

**for** all given disks  $D$  **in order of increasing diameter** **do**

**if**  $D$  is disjoint from the disks in  $I$  **then**

$I = I \cup \{D\}$ ;

**return**  $I$ ;

## Analysis of SMALLEST-GREEDY

Again, charge disks in the optimal solution  $I^*$  to disks in the solution  $I$  computed by the algorithm.

↳ Every disk  $D$  in  $I$  receives charge only from disks in  $I^*$  that intersect  $D$  and were processed after  $D$ . There can be **at most five such disks**.

**SMALLEST-GREEDY is a 5-approximation algorithm.**

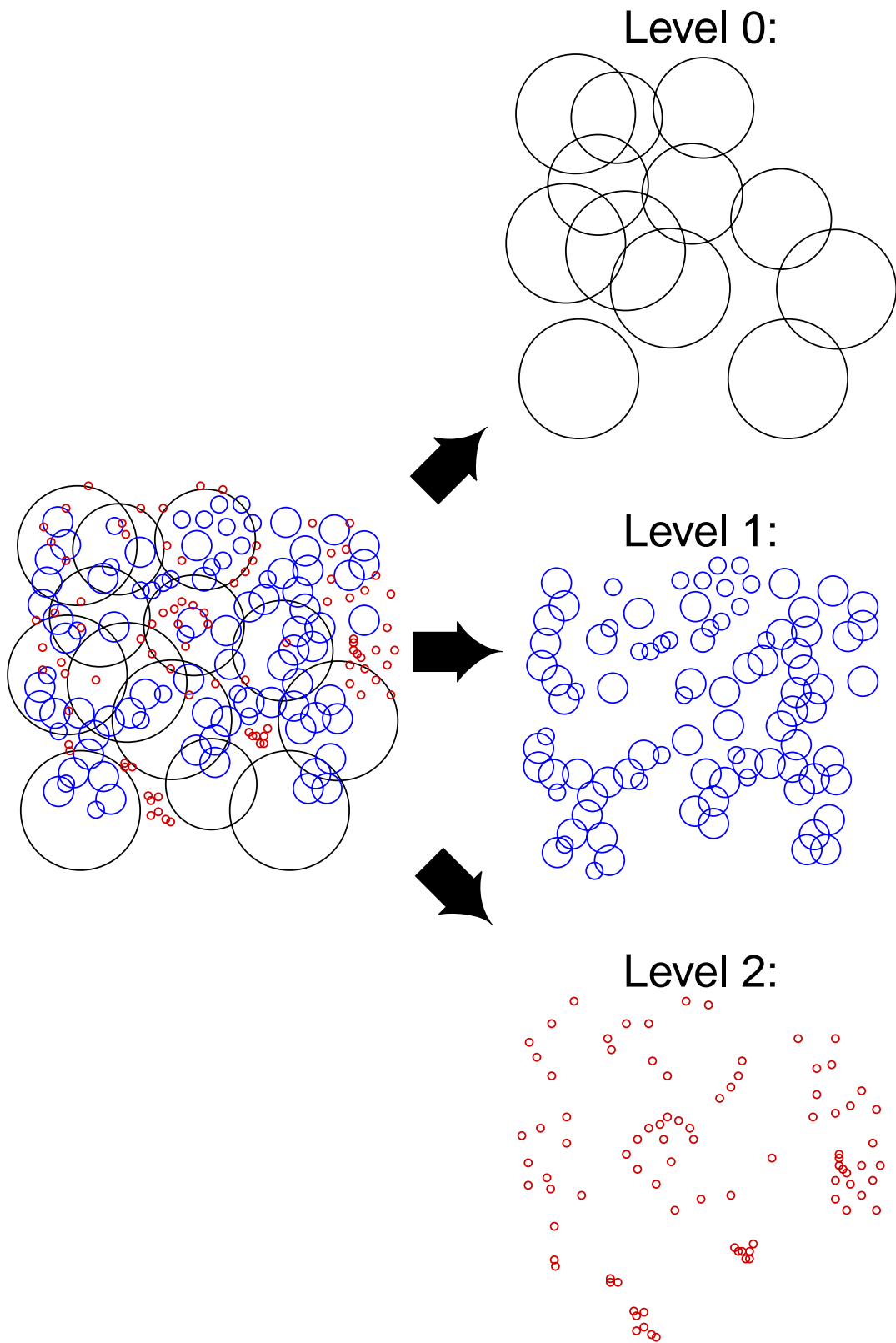
If the representation is not given: Find a vertex whose neighborhood does not contain an independent set of size 6, select it, and delete its neighbors.

## Extending the shifting strategy to disk graphs

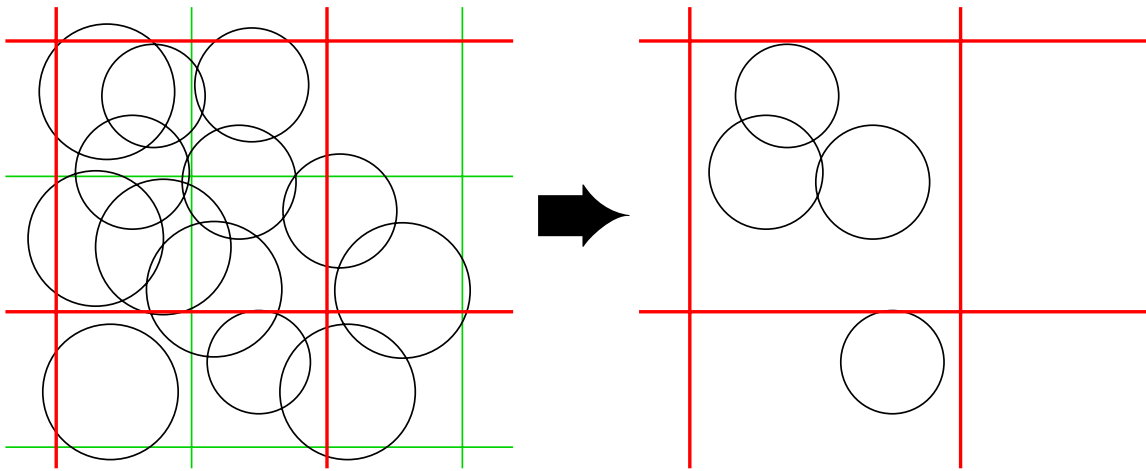
- ① Classify the disks into **levels** according to their sizes.
- ② Use the shifting strategy **on all levels simultaneously**.
- ③ After removing all disks that hit active lines, use **dynamic programming** to compute a maximum independent set.

### Classification into levels:

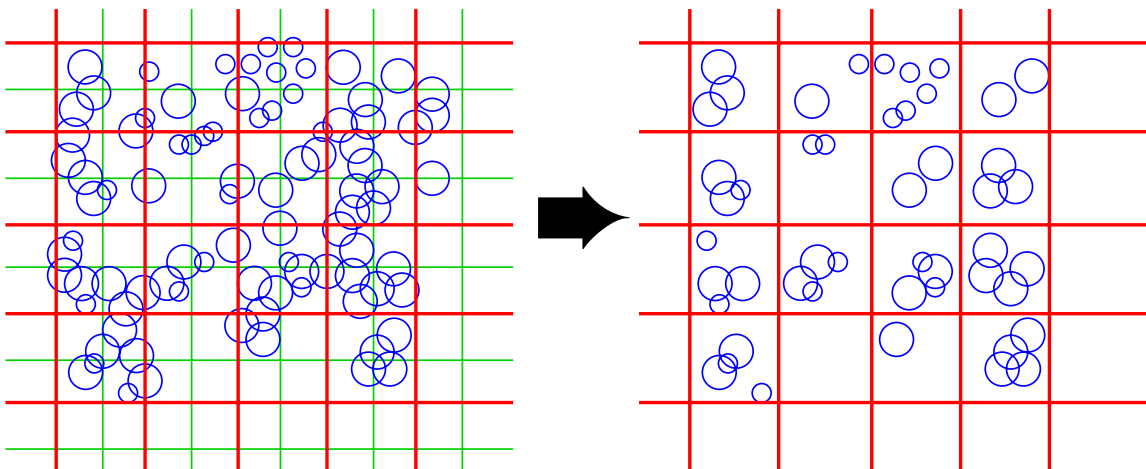
- Assume that the largest disk has diameter 1.
- **Level  $\ell$** : all disks whose diameter  $d$  satisfies  $\frac{1}{(k+1)^\ell} \geq d > \frac{1}{(k+1)^{\ell+1}}$ .
- Lines on level  $\ell$  are  $\frac{1}{(k+1)^\ell}$  **apart**, every  $k$ -th line is **active**.



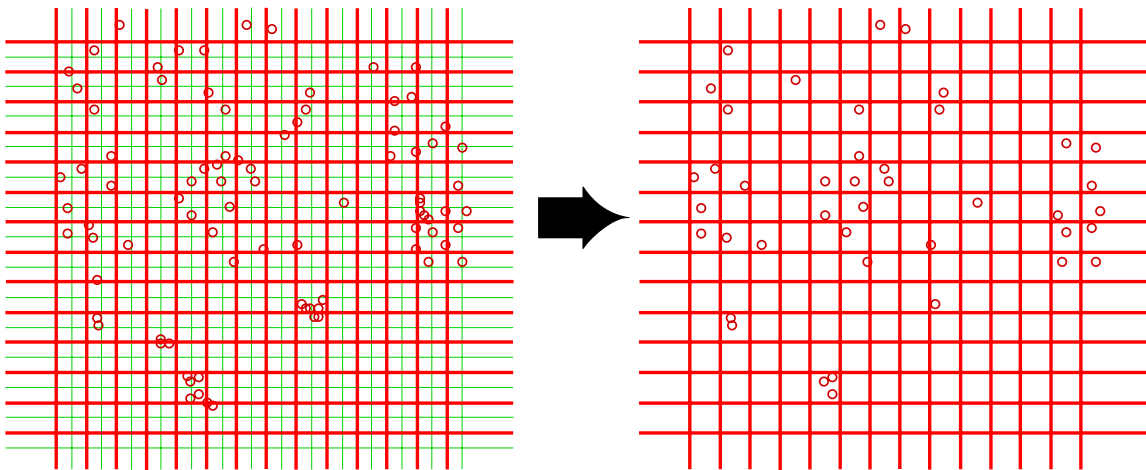
Level 0:



Level 1:



Level 2:

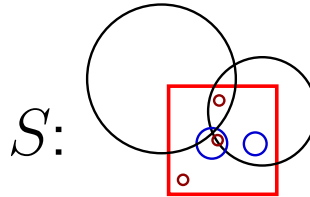


# Dynamic programming table

At square  $S$  on level  $\ell$ , compute  $\text{TABLE}_S$ .

If  $I$  is an independent set of disks of level  $< \ell$  intersecting  $S$ , then

$$\text{TABLE}_S[I] = \begin{cases} \text{size of maximum independent set } I' \\ \text{of disks of level } \geq \ell \text{ in } S \text{ such that} \\ I \cup I' \text{ is an independent set.} \end{cases}$$



$$\text{TABLE}_S \left[ \begin{array}{|c|} \hline \square \\ \hline \end{array} \right] = 4 \quad (\text{note } \begin{array}{|c|} \hline \circ \\ \hline \circ \quad \circ \\ \hline \circ \end{array} )$$

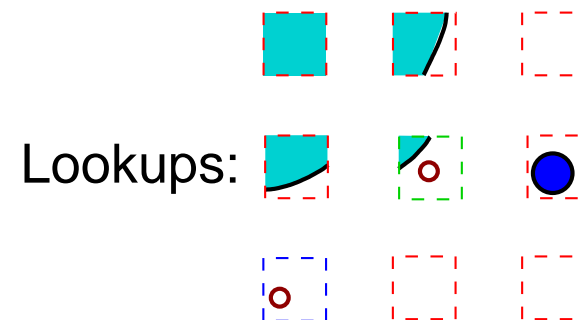
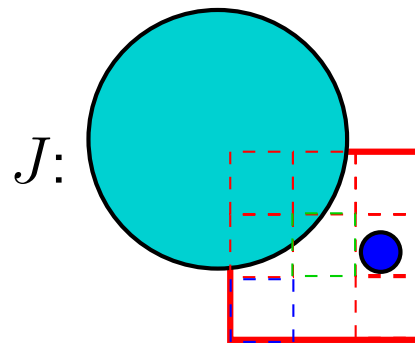
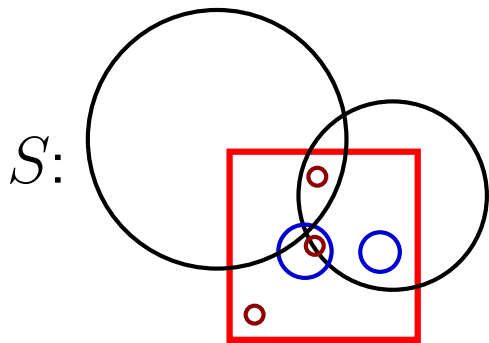
$$\text{TABLE}_S \left[ \begin{array}{|c|} \hline \bigcirc \\ \hline \end{array} \right] = 3 \quad (\text{note } \begin{array}{|c|} \hline \bigcirc \\ \hline \circ \quad \circ \\ \hline \circ \end{array} )$$

$$\text{TABLE}_S \left[ \begin{array}{|c|} \hline \bigcirc \\ \hline \end{array} \right] = 1 \quad (\text{note } \begin{array}{|c|} \hline \bigcirc \\ \hline \circ \end{array} )$$

# Computing $TABLE_S$

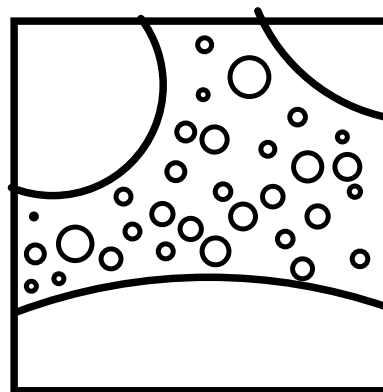
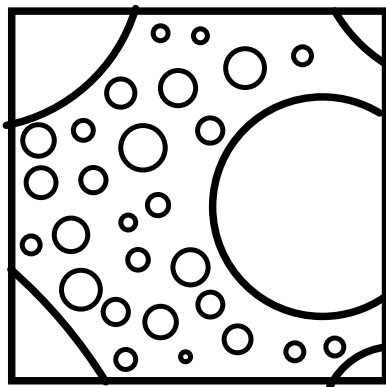
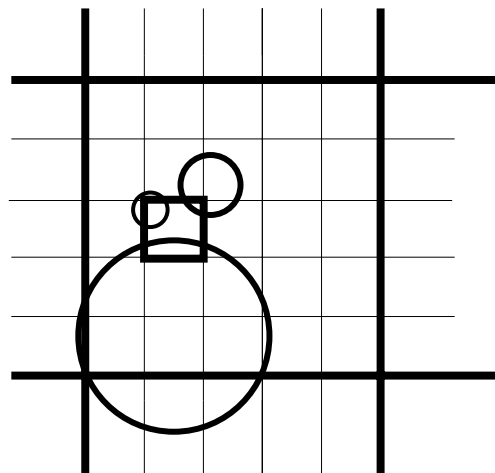
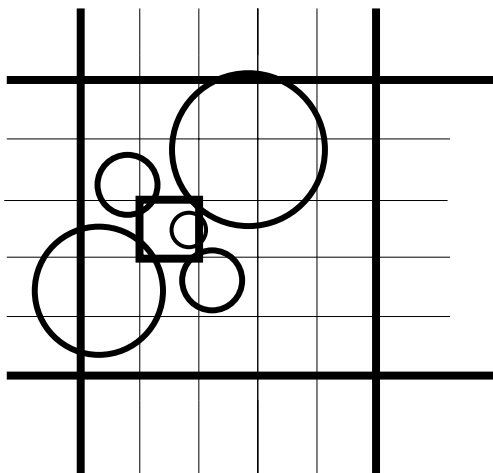
1. **Enumerate** all  $n^{O(k^4)}$  independent sets  $J$  of disks of level  $\leq \ell$  touching  $S$ .
2. **Look up** corresponding entries of  $TABLE_{S'}$  for subsquares of  $S$ .
3. Update  $TABLE_S[I]$  for  $I = \{D \in J \mid D \text{ has level} < \ell\}$ .

## Example:



$$\Rightarrow TABLE_S \left[ \begin{array}{c} \text{Cyan Circle} \\ \text{Red Square} \end{array} \right] = \max \left\{ TABLE_S \left[ \begin{array}{c} \text{Cyan Circle} \\ \text{Red Square} \end{array} \right], 3 \right\} \quad (\text{note } \begin{array}{c} \text{Cyan Circle} \\ \text{Red Square} \end{array} )$$

## Two more examples for lookups



## The PTAS for MAXINDEPENDENTSET

- 1 For  $0 \leq r, s < k$ , get  $\mathcal{D}(r, s)$  from  $\mathcal{D}$  by deleting disks that
    - hit a horizontal line equal to  $r \bmod k$  on their level, or
    - hit a vertical line equal to  $s \bmod k$  on their level
  - 2 Compute **dynamic programming tables** for  $\mathcal{D}(r, s)$  in all squares.
  - 3 The union of  $\text{TABLE}_S[\emptyset]$  over all top-level squares gives a **maximum independent set** in  $\mathcal{D}(r, s)$ .
  - 4 **Output the largest independent set** obtained in this way.
- 

**Running-time:**  $n^{O(k^4)}$  for  $n$  disks. (Can be improved to  $n^{O(k^2)}$ .)

**Approximation:** Computed solution has size **at least**  $(1 - \frac{2}{k}) OPT$ .

## MAXINDSET in disk graphs: Summary

⇒ SMALLEST-GREEDY is a 5-approximation algorithm. There is a variant that does not need the representation.

[Marathe et al., 1995]

⇒ The shifting strategy combined with dynamic programming gives a PTAS. It needs the representation.

[E, Jansen, Seidel'01:  $n^{O(k^2)}$ ; Chan'01:  $n^{O(k)}$ ]

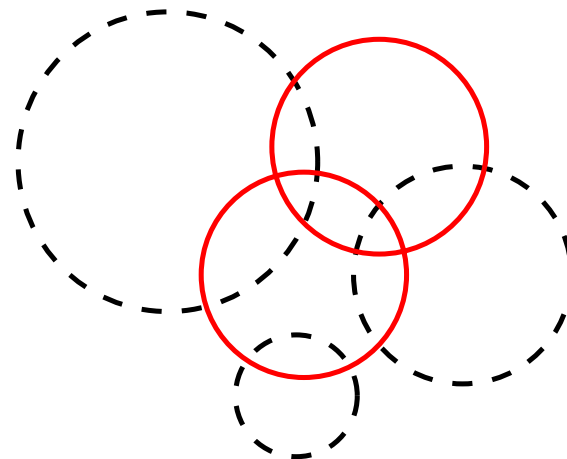
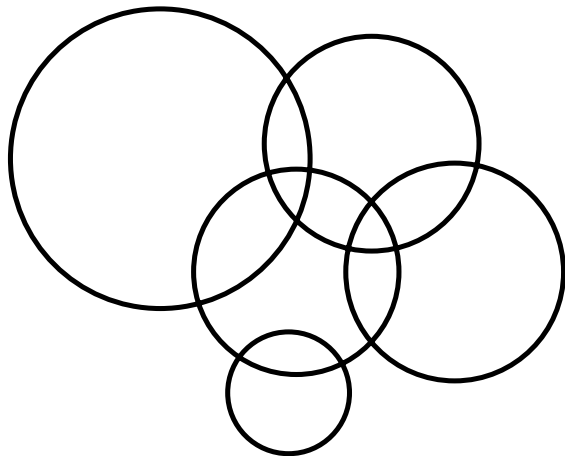
**Note:** These results can be adapted to **squares, regular polygons and other “disk-like” objects**, also in **higher dimensions**. The PTAS works also for the **weighted version**.

## The problem MINVERTEXCOVER

**Input:** a set  $\mathcal{D}$  of disks in the plane

**Feasible solution:** subset  $C \subseteq \mathcal{D}$  of disks such that, for any  $D_1, D_2 \in \mathcal{D}$ ,  
 $D_1 \cap D_2 \neq \emptyset \Rightarrow D_1 \in C$  or  $D_2 \in C$ .

**Goal:** minimize  $|C|$



## Approximation algorithms for MINVERTEXCOVER

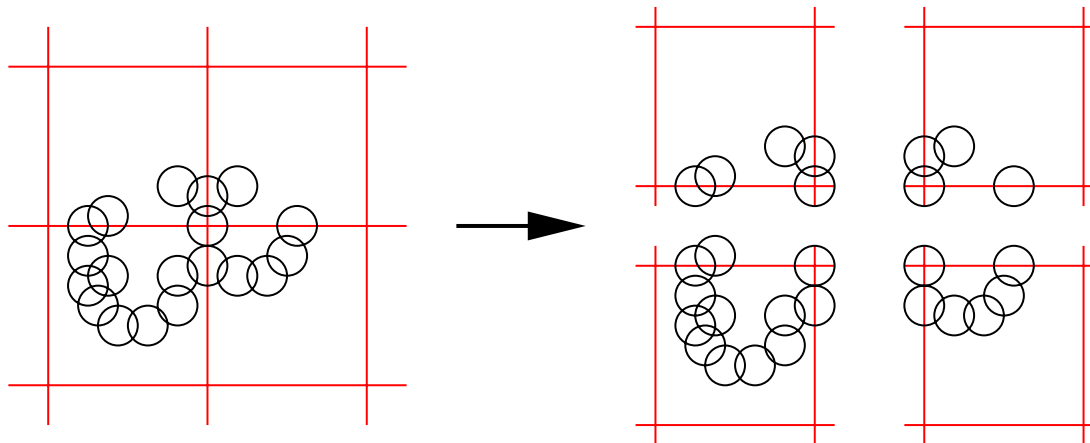
An algorithm for MINVERTEXCOVER is a  $\rho$ -approximation algorithm if it

- runs in **polynomial time** and
- always outputs a vertex cover of **size at most**  $\rho \cdot OPT$ , where  $OPT$  is the size of the optimal vertex cover.

A **polynomial-time approximation scheme (PTAS)** is a family of  $(1 + \varepsilon)$ -approximation algorithms for every constant  $\varepsilon > 0$ .

## Approximating MINVERTEXCOVER

- **Fact:**  $I$  is an independent set  $\Leftrightarrow \mathcal{D} \setminus I$  is a vertex cover
- To approximate MINVERTEXCOVER in unit disk graphs, we can again use the **shifting strategy**.
- Disks that hit an active line are considered in **all squares that they intersect** (at most 4 squares).



## PTAS for MINVERTEXCOVER in unit disk graphs

- 1 For  $0 \leq r, s < k$ , partition the plane into squares via
  - horizontal lines equal to  $r \bmod k$  and
  - vertical lines equal to  $s \bmod k$ .
- 2 Compute the minimum vertex cover  $C_S$  among the disks intersecting each  $k \times k$  square  $S$  by computing a maximum independent set and taking the complement.
- 3 The union of the sets  $C_S$  gives a **candidate vertex cover** (for each  $(r,s)$ ).
- 4 **Output the smallest vertex cover** obtained in this way.

---

**Running-time:**  $n^{O(k^2)}$  for  $n$  disks. (Can be improved to  $n^{O(k)}$ .)

## Analysis of PTAS for MINVERTEXCOVER

- ▶ Let  $C^*$  be an optimum vertex cover.
- ▶ For  $0 \leq r, s < k$  let  $C^*(r, s)$  be the disks intersecting active lines for  $(r, s)$  and let  $\mathcal{S}(r, s)$  be the set of all  $k \times k$  squares determined by these active lines.
- ▶ For a  $k \times k$ -square  $S$ , let  $C_S^*$  be the disks in  $C^*$  intersecting  $S$  and let  $OPT(S)$  be the optimum vertex cover of the disks intersecting  $S$ .

Candidate vertex cover computed by the algorithm for  $(r,s)$  has size

$$\begin{aligned} \left| \bigcup_{S \in \mathcal{S}(r,s)} OPT(S) \right| &\leq \sum_{S \in \mathcal{S}(r,s)} |OPT(S)| \\ &\leq \sum_{S \in \mathcal{S}(r,s)} |C^*(S)| \\ &\leq 3|C^*(r,s)| + |C^*| \end{aligned}$$

For some choice of  $(r,s)$ :

$\Rightarrow$  at most  $\frac{1}{k}|C^*|$  disks of  $C^*$  intersect vertical active lines

$\Rightarrow$  at most  $\frac{1}{k}|C^*|$  disks of  $C^*$  intersect horizontal active lines

For this choice, we have  $|C^*(r,s)| \leq \frac{2}{k}|C^*|$ .

$\blackrightarrow$  Solution has size **at most**  $\left(1 + \frac{6}{k}\right) |C^*|$  for some choice of  $(r,s)$ .

## MINVERTEXCOVER in disk graphs: Summary

- ▶▶▶ PTAS for **unit disk graphs** using the shifting strategy (needs the representation). [Hunt III et al., 1994]
- ▶▶▶  $\frac{3}{2}$ -approximation algorithm for **general disk graphs** (not needing the representation). [Malesińska, 1997]
- ▶▶▶ PTAS for **general disk graphs** using the shifting strategy and dynamic programming (needs the representation).  
[E, Jansen, Seidel'01]

---

**Note:** PTASs adapt to **squares, regular polygons etc.**, also in **higher dimensions**. Results hold for the **weighted version** as well.

## Coloring disk graphs

**Goal:** Assign a minimum number of colors to the disks such that intersecting disks get different colors!

**Algorithm SMALLEST-DEGREE-LAST(graph  $G$ )**

$v$  = a vertex with minimum degree in  $G$ ;

color  $G \setminus \{v\}$  recursively;

assign  $v$  the smallest available color;

**Observation.** Let  $D$  be the maximum degree of a vertex  $v$  at the time it was colored. Then the algorithm needs at most  $D + 1$  colors.

## Analysis of SMALLEST-DEGREE-LAST for disk graphs

Let  $v$  be the vertex corresponding to the smallest disk.

Let  $N(v)$  be the set of neighbors of  $v$ .

**Note:** At most 5 disks in  $N(v)$  can get the same color.

↳ Optimal number of colors  $OPT$  is at least  $1 + \frac{|N(v)|}{5}$ .

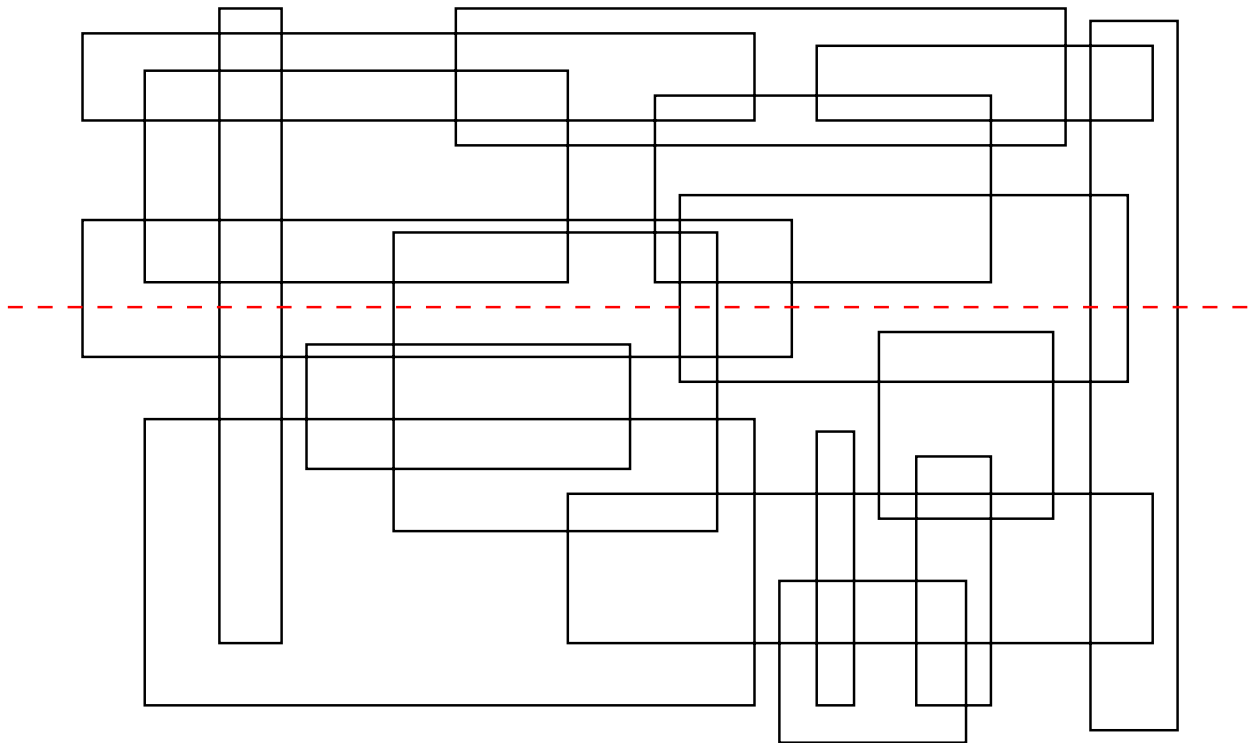
↳  $|N(v)| \leq 5 \cdot OPT - 5$ .

↳ So we must also have  $D \leq 5OPT - 5$ .

**The SMALLEST-DEGREE-LAST algorithm colors any disk graph with at most  $5OPT - 4$  colors.** [Marathe et al. 1995; Gräf 1995]

# MAXINDEPENDENTSET in Rectangle Graphs

★ **Idea:** find a “stabbing line” with at most half of the rectangles above and below.



## An approximation algorithm for rectangles

**Algorithm RECTANGLE-APPROX**(set of rectangles  $R$ )

$\ell$  = stabbing line with at most  $|R|/2$  rectangles above and below;

$R_{\text{above}}$  = rectangles above stabbing line;

$R_{\text{below}}$  = rectangles below stabbing line;

$R_{\text{mid}}$  = rectangles intersecting stabbing line;

compute approximations  $I_1$  and  $I_2$  for  $R_{\text{above}}$  and  $R_{\text{below}}$  recursively;

compute optimal independent set  $I_0$  for  $R_{\text{mid}}$ ;

**return** the larger of  $I_0$  and  $I_1 \cup I_2$ ;

## Analysis of RECTANGLE-APPROX

The algorithm achieves approximation ratio  $\log n$  for  $n$  rectangles.

**Proof.** by induction on the number of rectangles.

Let  $I^*$  be an optimal independent set.

Let  $I_0^*$ ,  $I_1^*$ ,  $I_2^*$  be the rectangles in  $I^*$  that are on, above, below  $\ell$ .

**Case 1:**  $|I_0^*|$  is at least  $|I^*| / \log n$ .

Algorithm outputs a set of size at least

$$|I_0| \geq |I_0^*| \geq \frac{|I^*|}{\log n}.$$

**Case 2:**  $|I_0^*|$  is smaller than  $|I^*|/\log n$ .

The algorithm outputs a set of size at least

$$\begin{aligned}
 |I_1 \cup I_2| &\geq \frac{OPT(R_{\text{above}})}{\log |R_{\text{above}}|} + \frac{OPT(R_{\text{below}})}{\log |R_{\text{below}}|} \\
 &\geq \frac{OPT(R_{\text{above}})}{(\log n) - 1} + \frac{OPT(R_{\text{below}})}{(\log n) - 1} \\
 &\geq \frac{|I_1^*| + |I_2^*|}{(\log n) - 1} = \frac{|I^*| - |I_0^*|}{(\log n) - 1} \\
 &\geq \frac{|I^*| \cdot \left(1 - \frac{1}{\log n}\right)}{(\log n) - 1} = \frac{|I^*|}{\log n}
 \end{aligned}$$

□

## MAXINDSET in rectangle graphs: Summary

⇒ There is an  $O(\log n)$ -approximation algorithm (with given representation).

[Agarwal et al., 1998; Khanna et al. 1998; Nielsen 2000]

⇒ For every constant  $c > 0$ , there is an approximation algorithm with ratio  $1 + \frac{1}{c} \log n$ .

[Berman et al., 2001]

⇒ If all rectangles have the same height, there is a PTAS.

[Agarwal et al., 1998]

## Some open problems

- ❑ **MAXCLIQUE** in disk graphs: polynomial or  $\mathcal{NP}$ -hard?  
(polynomial for unit disk graphs)
- ❑ **MINDOMINATINGSET** in disk graphs:  
Is there an  $O(1)$ -approximation or PTAS ?
- ❑ Intersection graphs of **axis-aligned rectangles**:  
Is there an  $O(1)$ -approximation or PTAS for MAXINDEPENDENTSET?  
Can they be colored with  $O(\omega)$  colors, where  $\omega$  is the clique size?  
(best known upper bound:  $O(\omega^2)$  colors.)