Approximation algorithms for geometric intersection graphs

Thomas Erlebach

University *of* Leicester

Based on joint work with:

Christoph Ambühl, Klaus Jansen, Erik Jan van Leeuwen, Matúš Mihaľák, Marc Nunkesser, Eike Seidel

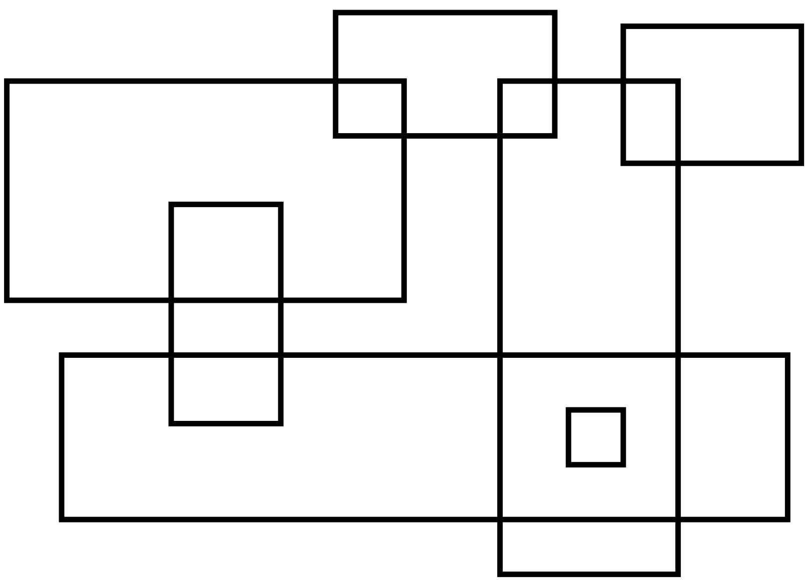
Outline

- Introduction
- Independent sets in disk graphs
- Vertex coloring disk graphs
- Independent sets in rectangle intersection graphs
- Dominating sets in unit disk graphs
- Some open problems

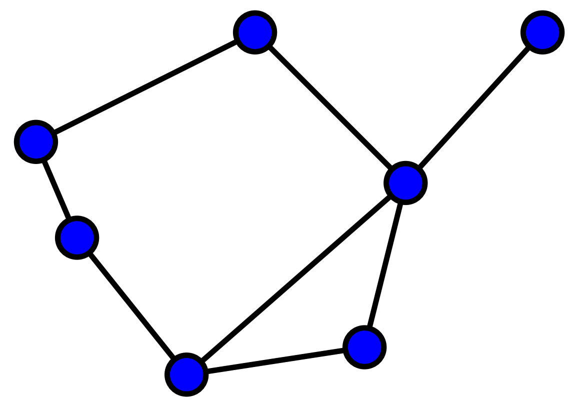
What are geometric intersection graphs?

- **vertices** = geometric objects
- **edges** = non-empty intersection between objects

Example: a rectangle intersection graph



geometric representation



intersection graph

Popular geometric intersection graphs

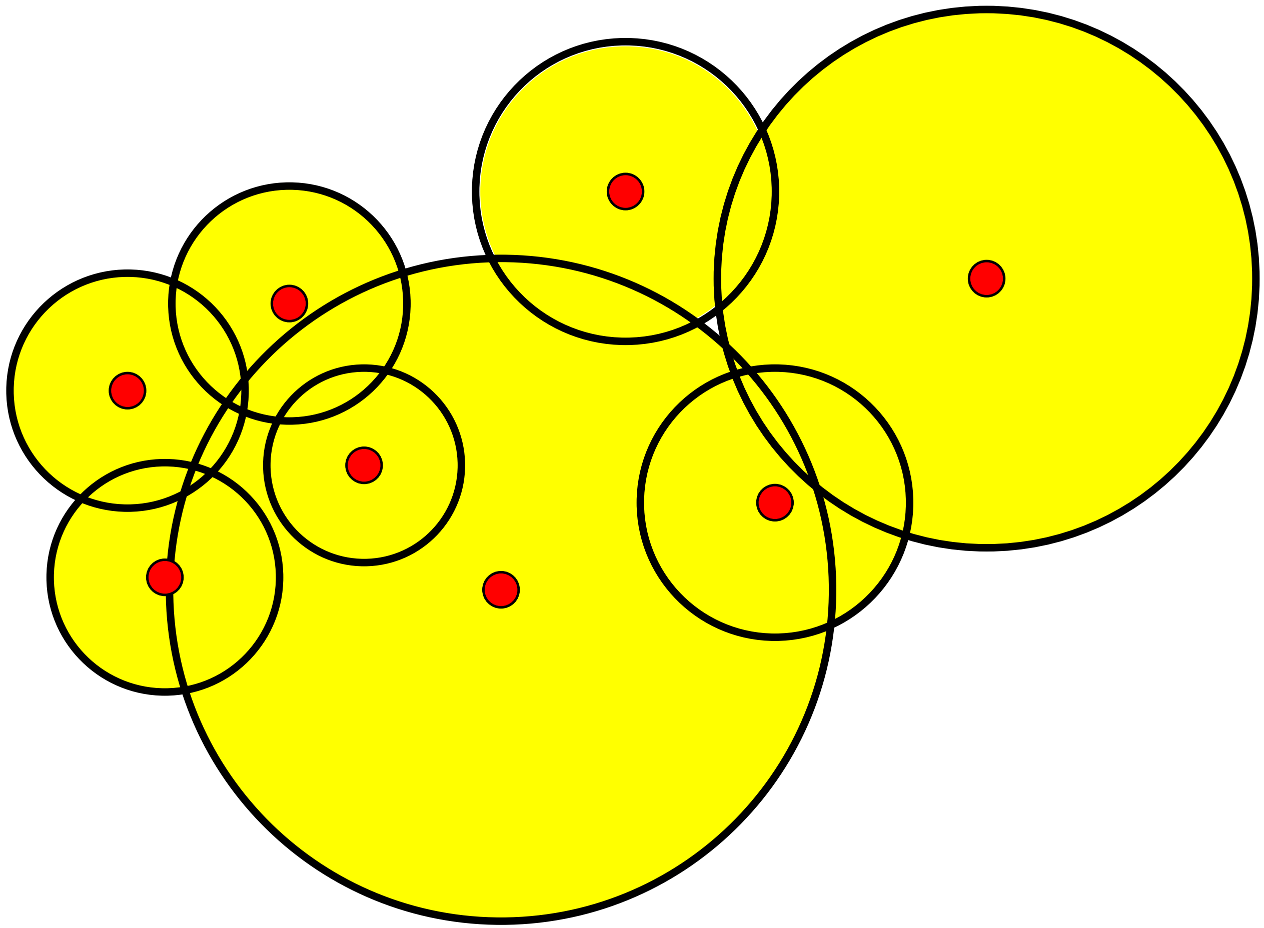
- disks (➜ **disk graphs**), squares
- “fat” objects
- ellipses, rectangles (axis-aligned), arbitrary convex objects
- line segments, curves, higher-dimensional objects

The **recognition problem is typically NP-hard**!!

Some Applications:

- ⇨ Wireless networks (frequency assignment problems)
- ⇨ Map labeling
- ⇨ Resource allocation (e.g. admission control in line networks)

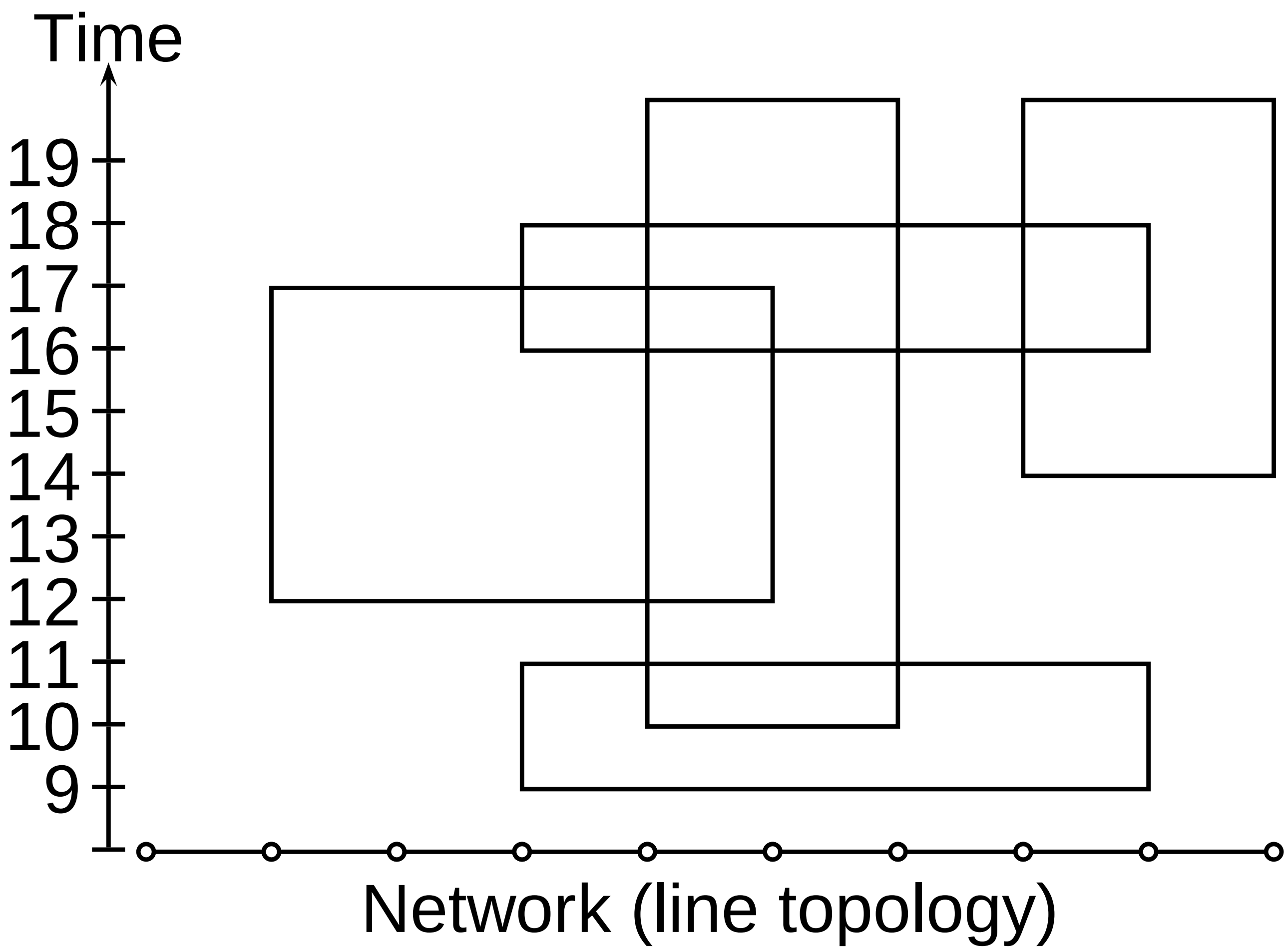
Application: Wireless networks



Application: Map labeling

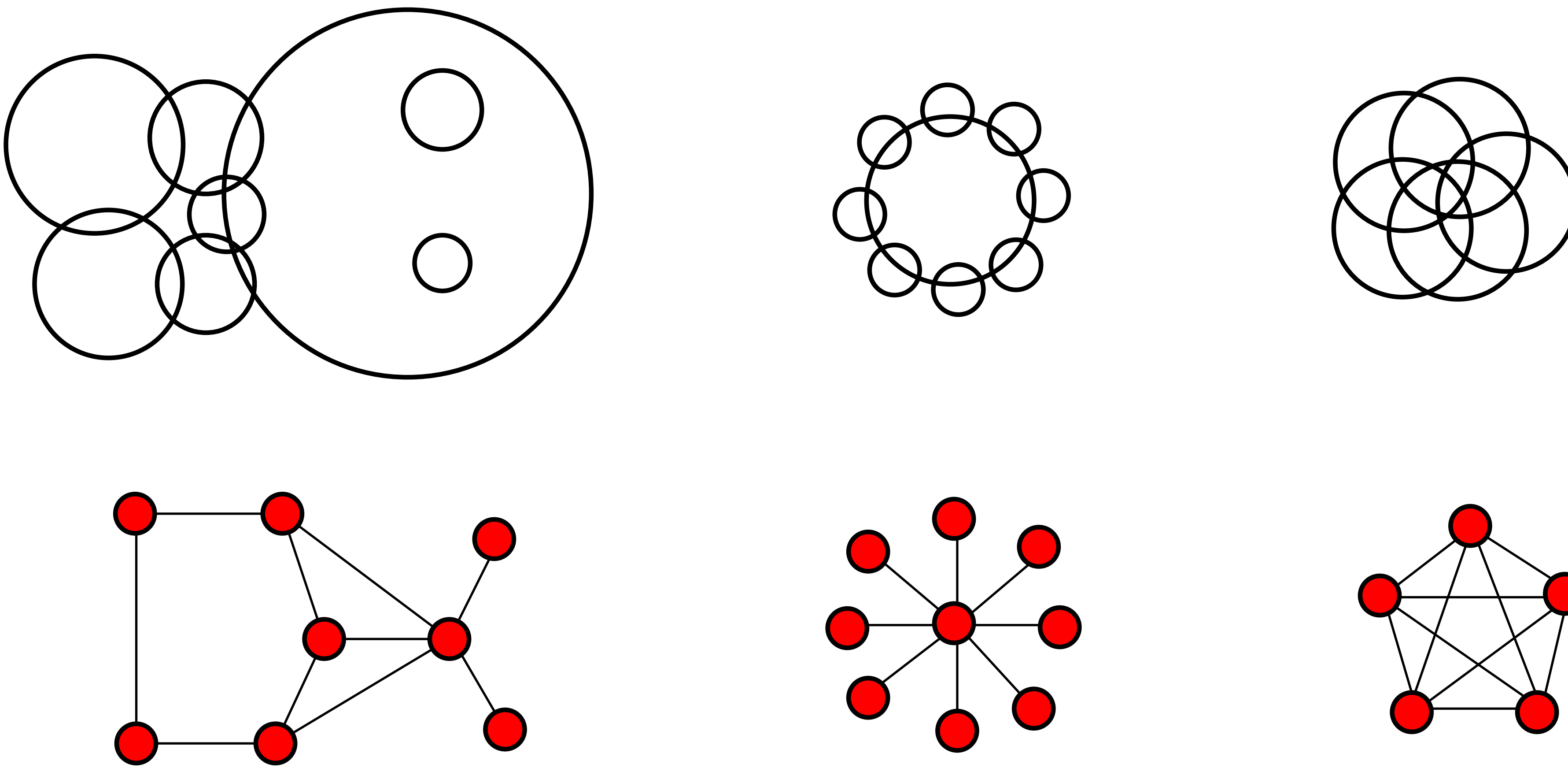
(illustration taken from a paper by van Kreveld, Strijk, Wolff)

Application: Call admission control



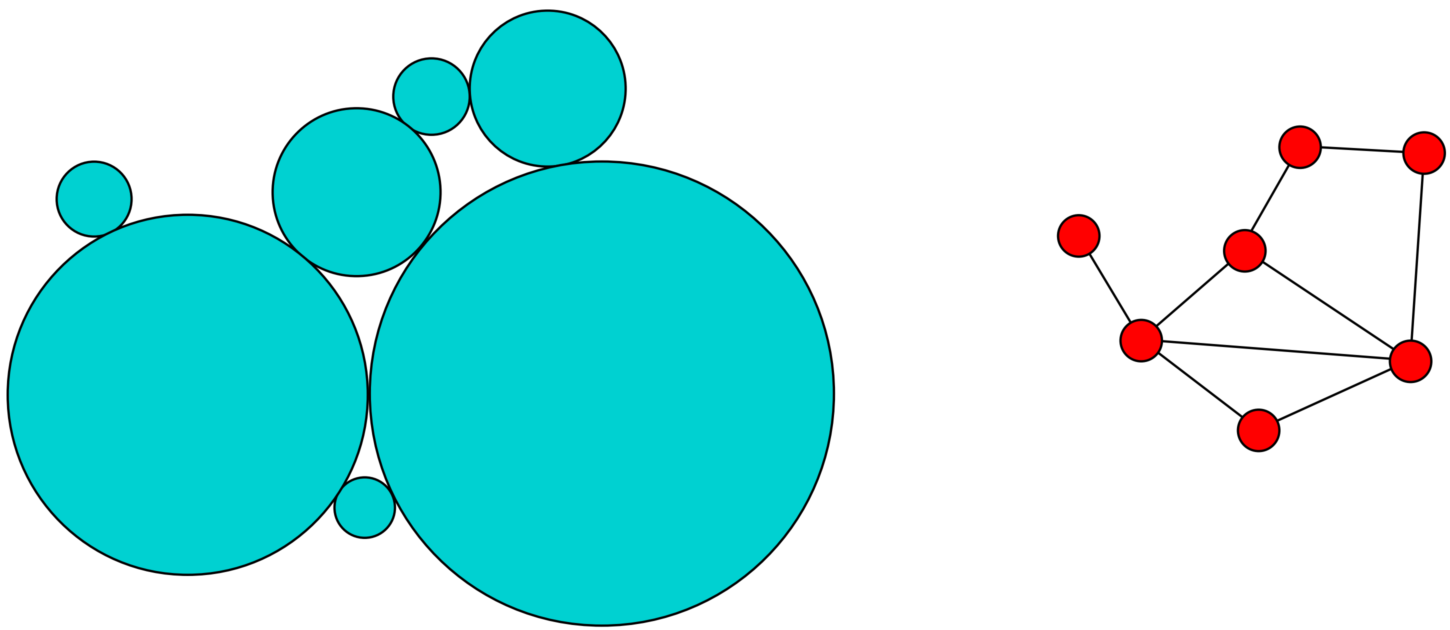
Disk graphs

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



Subclasses of disk graphs

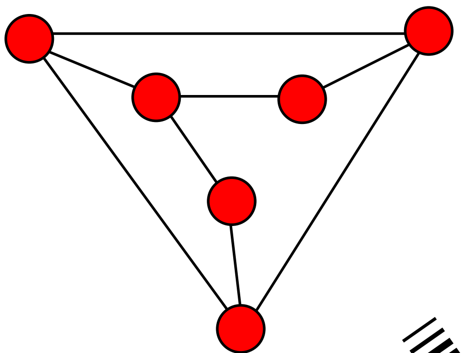
- **Unit disk graphs**: all disks have diameter 1
- **Coin graphs**: touching graphs of disks whose interiors are disjoint



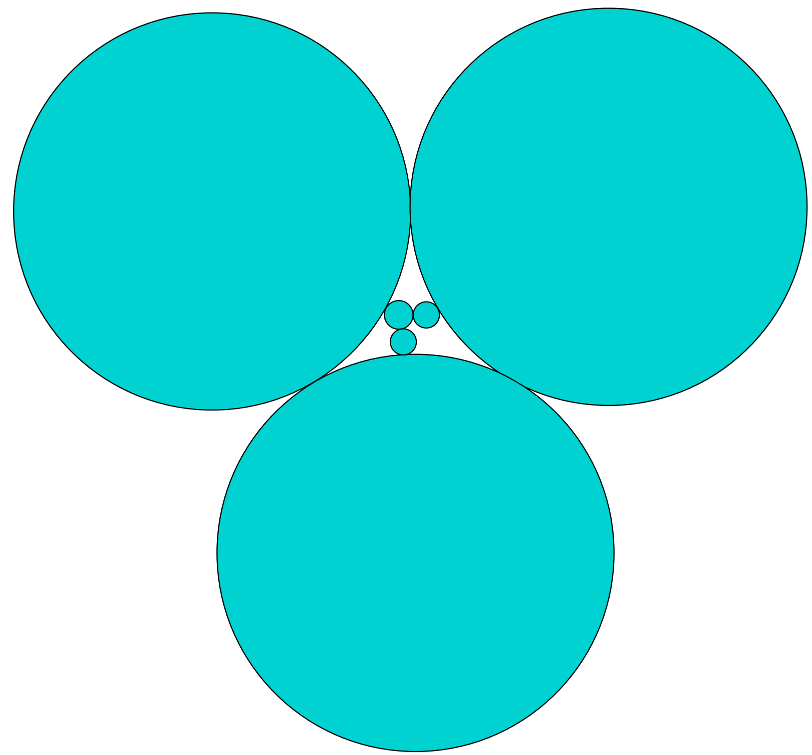
Coin graphs are planar, but surprisingly . . .

. . . every planar graph is a coin graph

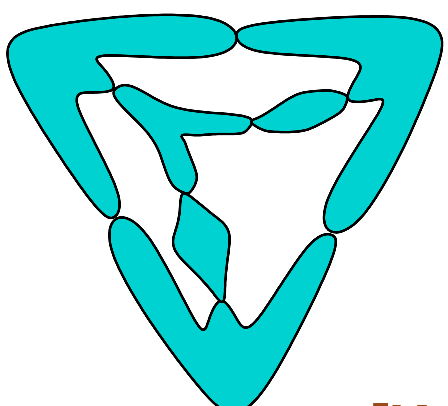
planar graph:



touching graph of disks:



touching graph of “blobs”:

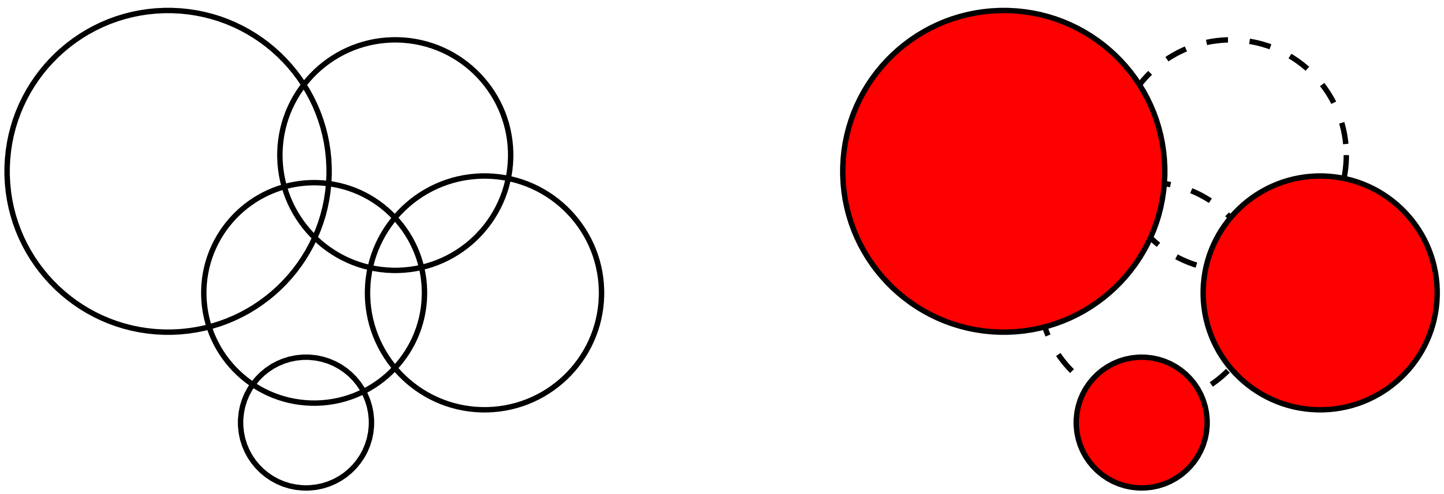


[Koebe, 1936]

Maximum Independent Set

Maximum Independent Set (MIS)

Input: a set $\mathcal{D}$ of disks in the plane
Feasible solution: subset $A \subseteq \mathcal{D}$ of disjoint disks
Goal: maximize $|A|$



In the weighted case (MWIS), each disk is associated with a positive weight.

Approximation algorithms for MIS

An algorithm for MIS is a ρ**-approximation algorithm** if it

- runs in **polynomial time** and
- always outputs an independent set of **size at least** OPT/ρ, 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 $\varepsilon > 0$.

For MWIS, the definitions are analogous.

MIS 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.

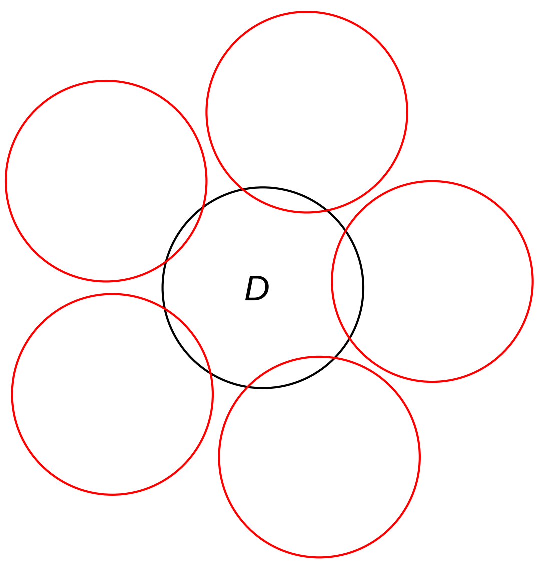
③ Bound the number of disks charged to the same disk in I.

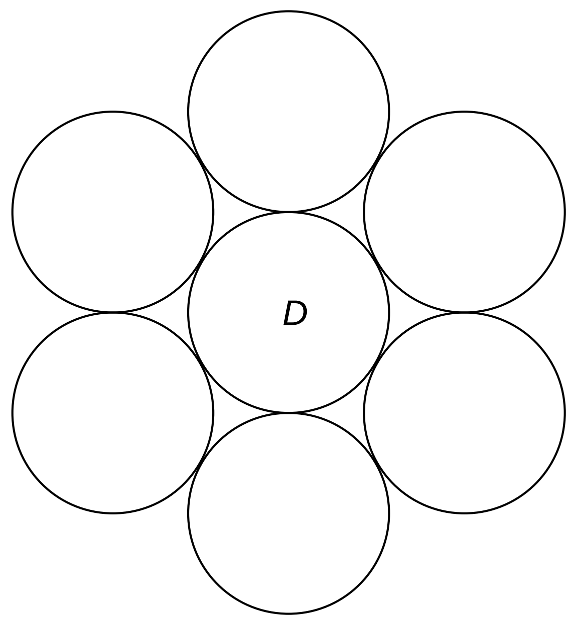
Charging rules for a disk $D \in I^*$:

- 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**.

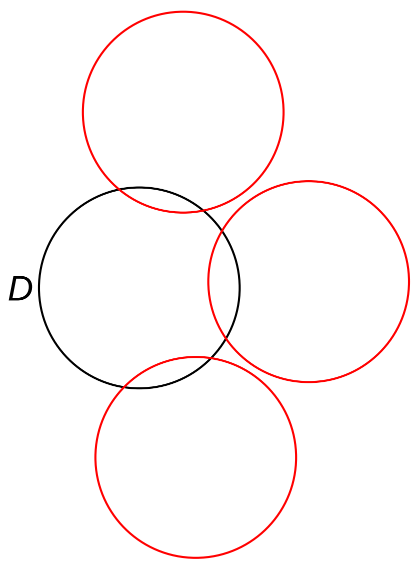
An improved greedy algorithm

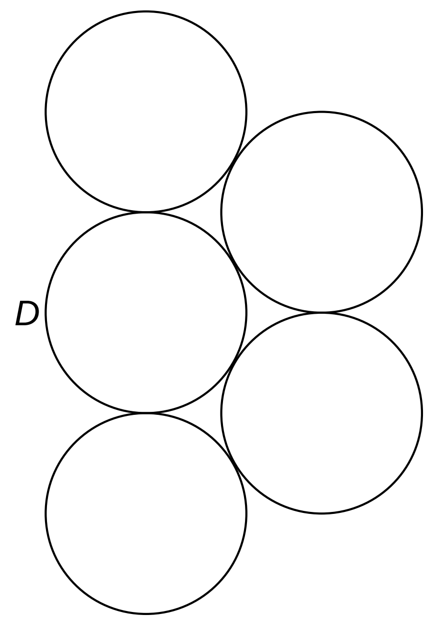
Algorithm LEFTMOST-GREEDY
$I = \emptyset$;
for all given disks D in order of increasing x-value **do**
    if D is disjoint from the disks in I **then**
        $I = I \cup \{D\}$;
return I;

Claim. LEFTMOST-GREEDY is a **3-approximation algorithm** for MIS 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 w/o representation

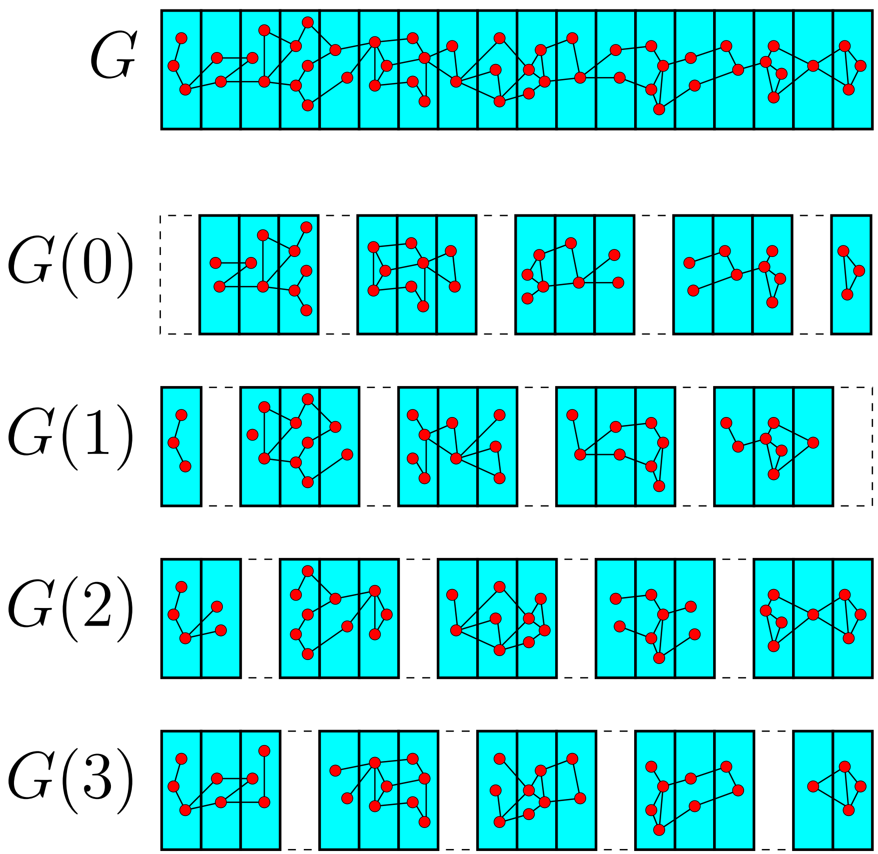
Given a graph $G = (V, E)$ that is the intersection graph of unit disks, the following is a 3**-approximation algorithm for MIS:**

$I = \emptyset$;
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.

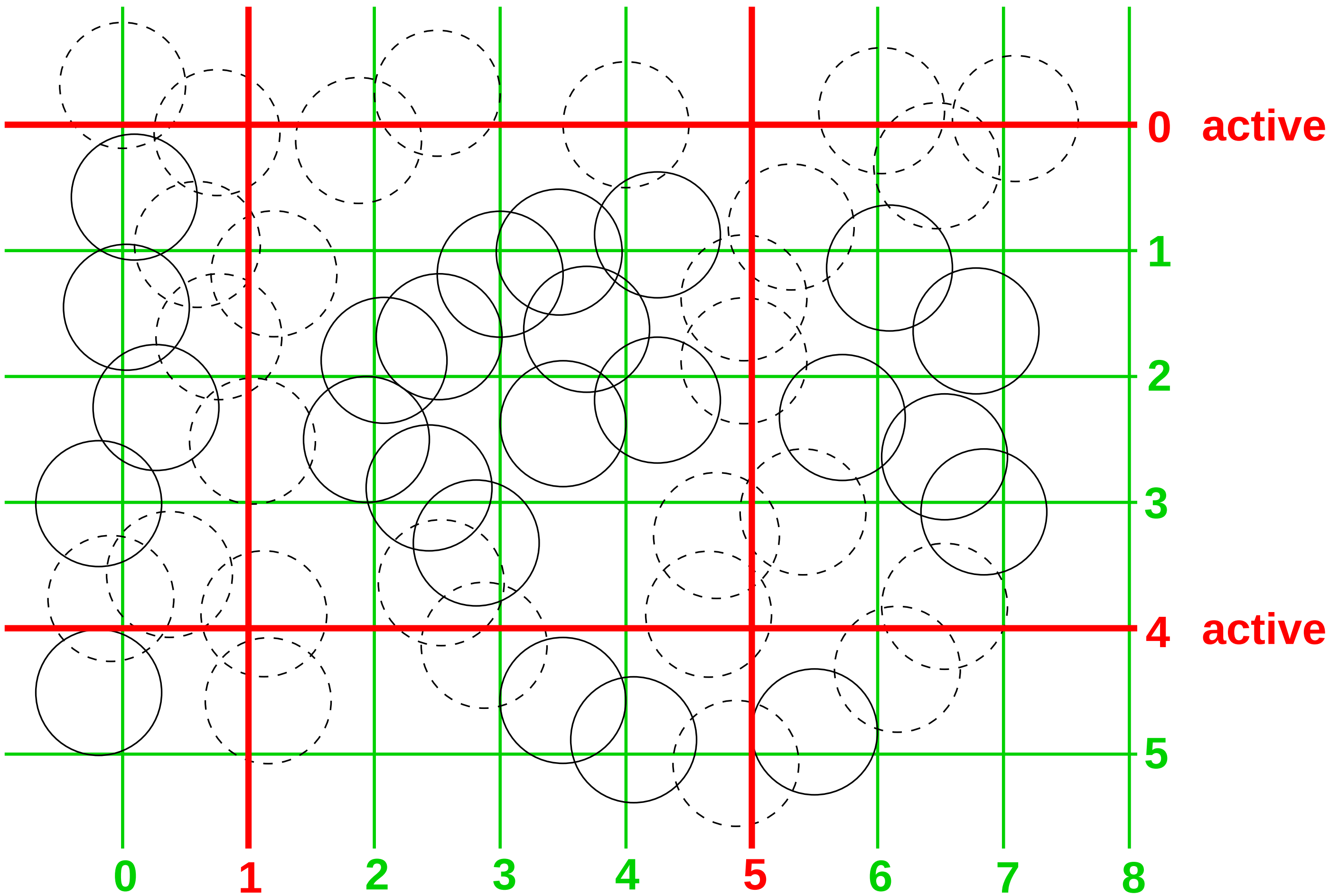
❸ Remove slices equal to ℓ modulo k and compute a maximum independent set in the graph $G(\ell)$, $0 \leq \ell < k$.

❹ Output the largest set found in this way.

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

Shifting for unit disk graphs

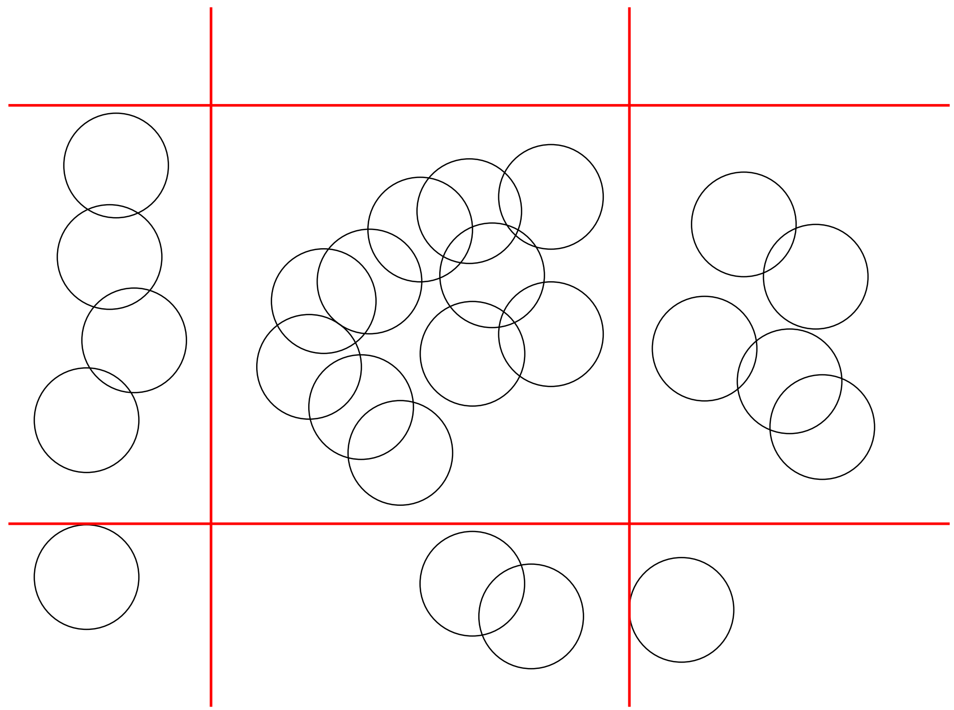
[Hochbaum and Maass, 1985]



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

Solving the Subproblems

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.

PTAS for MIS in unit disk graphs

❶ For $0 \le r, s < k$, 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.

❷ Compute the maximum independent set I_S in each $k \times k$ square S of $\mathcal{D}(r,s)$ by brute-force enumeration.

❸ The union of the sets I_S gives a maximum independent set in $\mathcal{D}(r,s)$.

❹ 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 $\left(1 - \frac{2}{k}\right) \mathrm{OPT}$.

MIS in unit disk graphs: Summary

- $\mathcal{NP}$-hard [Clark, Colbourn, Johnson 1990].
- 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]

Recent related results

- [Nieberg, Hurink, Kern, 2004] PTAS for maximum weight independent set in unit disk graphs without given representation.
- [Marx, 2005] Maximum independent set in unit disk graphs is W[1]-hard. (➠ No FPT algorithm and no EPTAS unless FPT=W[1].)
- [van Leeuwen, 2005] Asymptotic FPTAS for maximum independent set (and various other problems) in unit disk graphs of bounded density.

MIS 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.

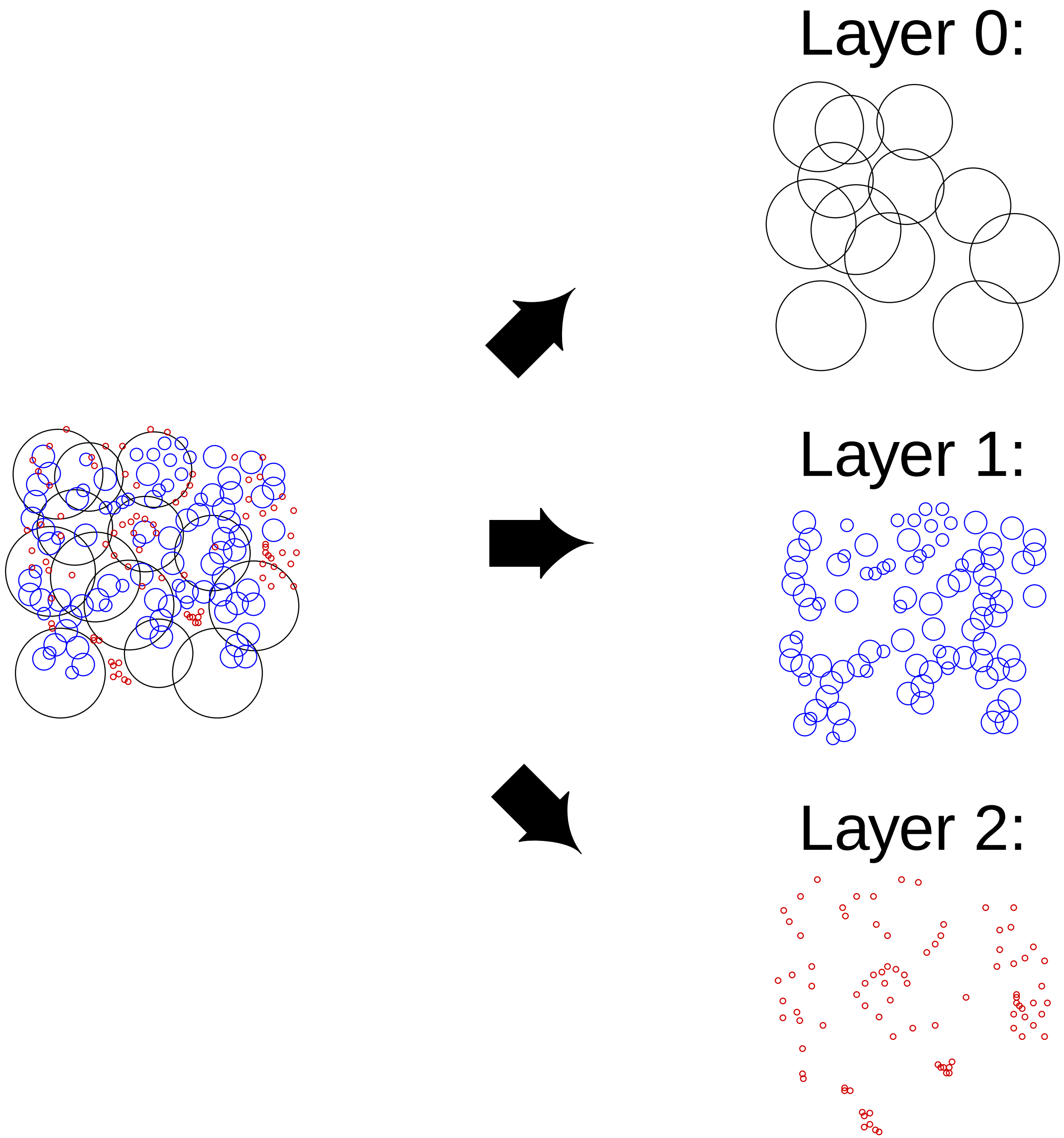
Extending the shifting strategy

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

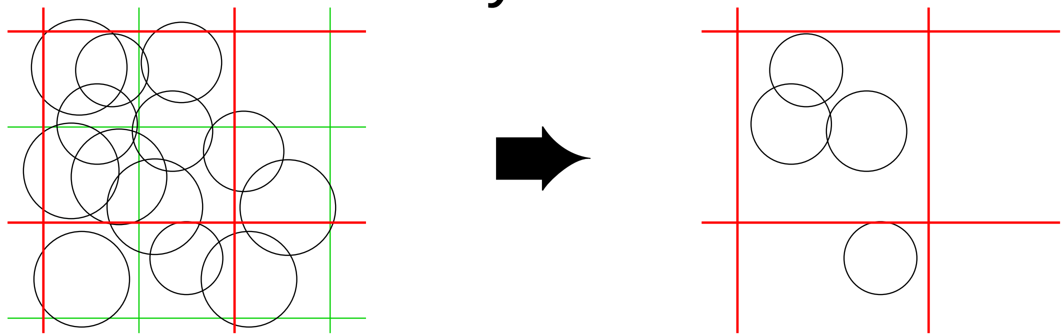
Classification into layers:

- Assume that the largest disk has diameter 1.
- Layer ℓ: disks with diameter d, $\frac{1}{(k+1)^\ell} \geq d > \frac{1}{(k+1)^{\ell+1}}$.
- Lines on layer ℓ are $\frac{1}{(k+1)^\ell}$ apart, every k-th line is active.

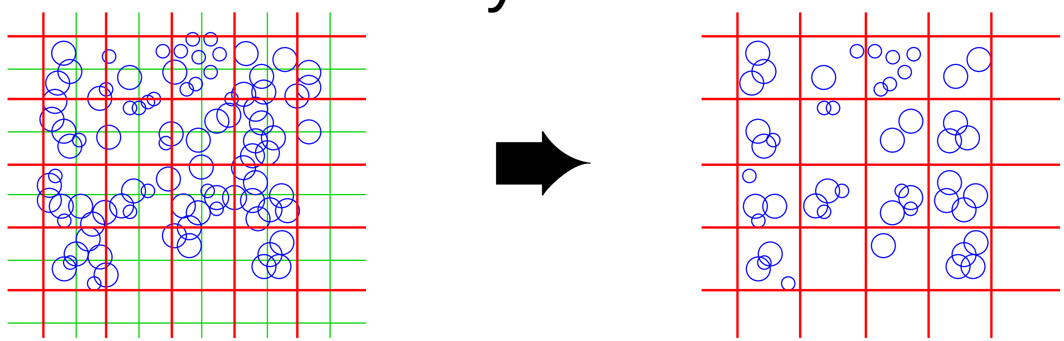
Partition into layers



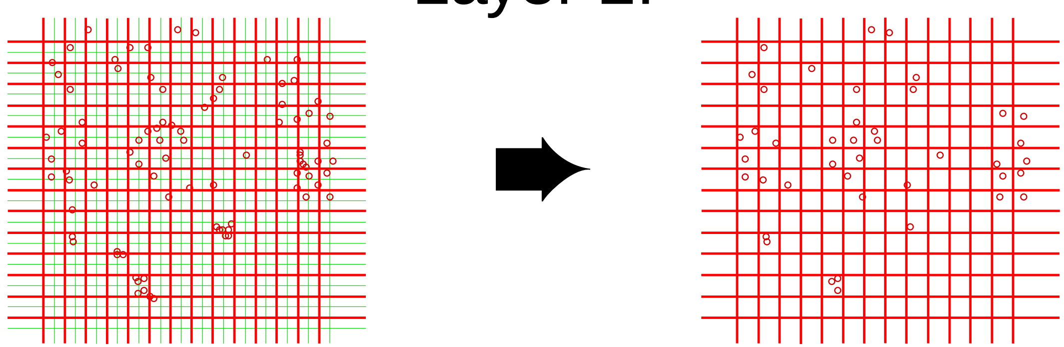
Layer 0:



Layer 1:



Layer 2:

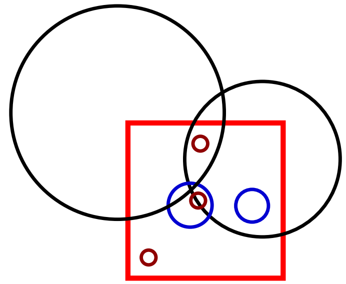


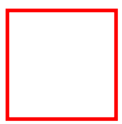
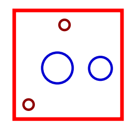
Dynamic programming table

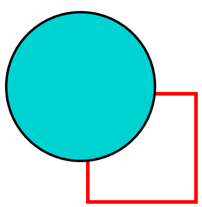
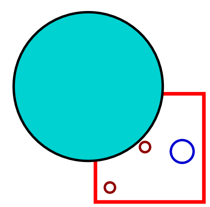
At square S on level ℓ, compute $\textsc{Table}_S$.
If I is an independent set of disks of level $< \ell$ intersecting S, then

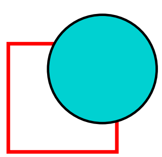
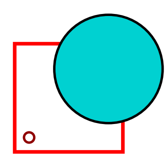
$$\textsc{Table}_S[I] = \begin{cases} \text{size of maximum independent set } I' \\ \text{of disks of level } \geq \ell \text{ in } S \text{ such that} \\ I \cup I' \text{ is an independent set.} \end{cases}$$

Example

S: 

$\textsc{Table}_S$ [] $= 4$ (note )

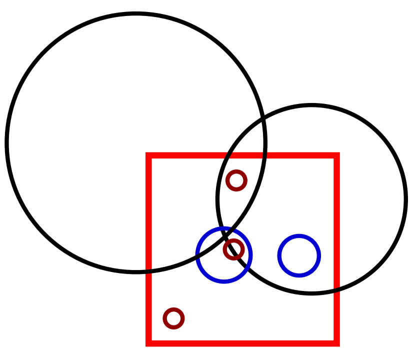
$\textsc{Table}_S$ [] $= 3$ (note )

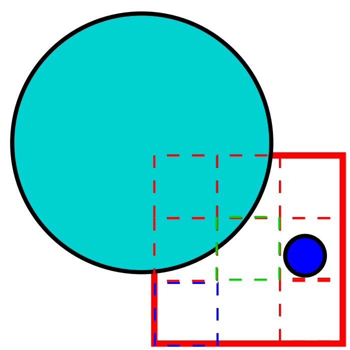
$\textsc{Table}_S$ [] $= 1$ (note )

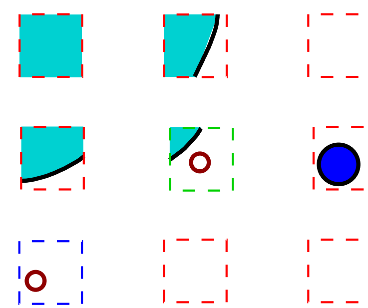
Computing $\textsc{Table}_S$

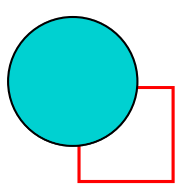
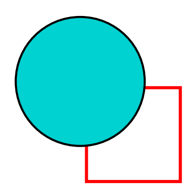
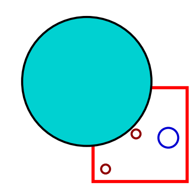
1. Enumerate all $n^{O(k^4)}$ independent sets J of disks of level $\le \ell$ touching S.
2. Look up corresponding entries of $\textsc{Table}_{S'}$ for subsquares of S.
3. Update $\textsc{Table}_S[I]$ for $I = \{D \in J \mid D \text{ has level } < \ell\}$.

Example:

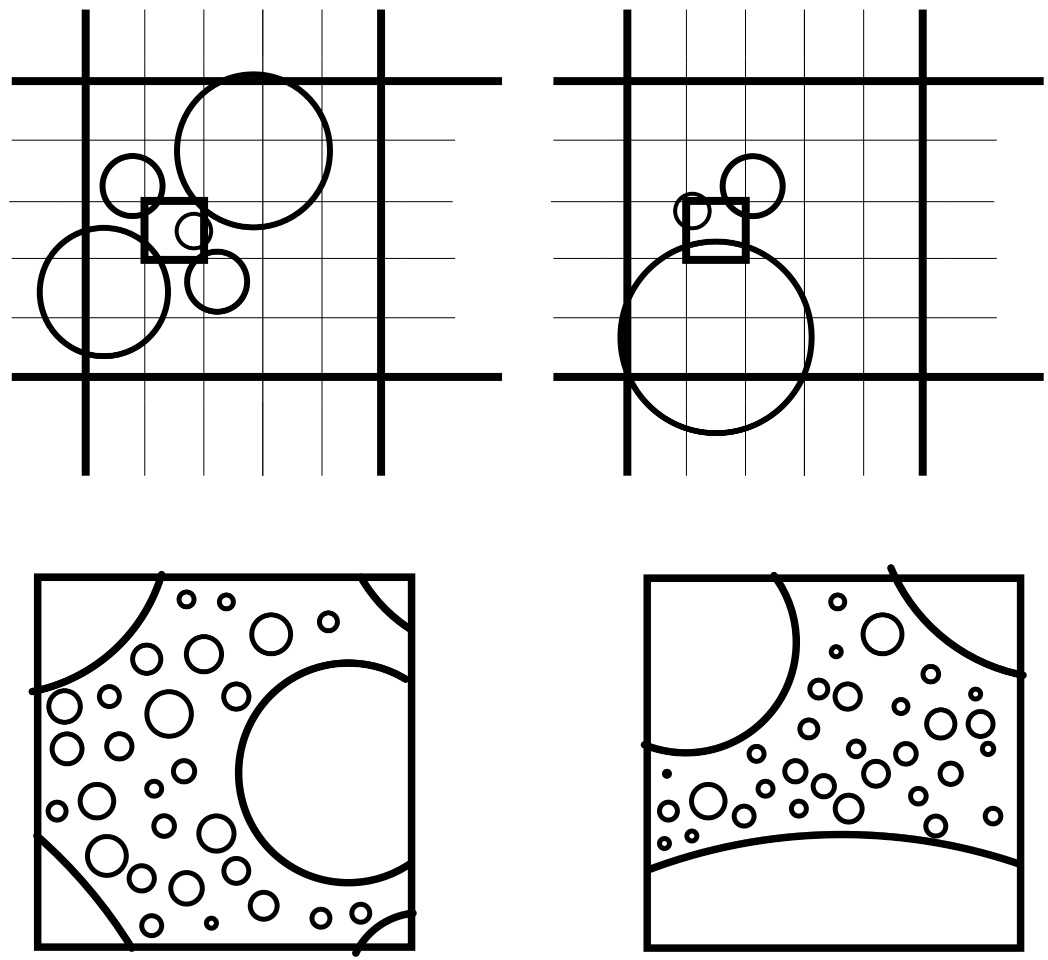
S: 

J: 

Lookups: 

$\Rightarrow$ $\textsc{Table}_S$ $\Big[$  $\Big] = \max \Big\{$ $\textsc{Table}_S$ $\Big[$  $\Big], 3 \Big\}$ (note )

Two more examples for lookups



The PTAS for MIS

❶ For $0 \le r, s < k$, get $\mathcal{D}(r,s)$ from $\mathcal{D}$ by deleting disks that

- ➜ hit a horizontal line equal to r modulo k on their level, or
- ➜ hit a vertical line equal to s modulo k on their level

❷ Compute dynamic programming tables for $\mathcal{D}(r,s)$ in all squares.

❸ The union of $\text{TABLE}_S[\emptyset]$ over all top-level squares gives a maximum independent set in $\mathcal{D}(r,s)$.

❹ 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 $\left(1 - \frac{2}{k}\right)\mathrm{OPT}$.

MIS 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" or fat objects**, also in **higher dimensions.** The PTAS works also for the **weighted version**.

Vertex Coloring

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 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.

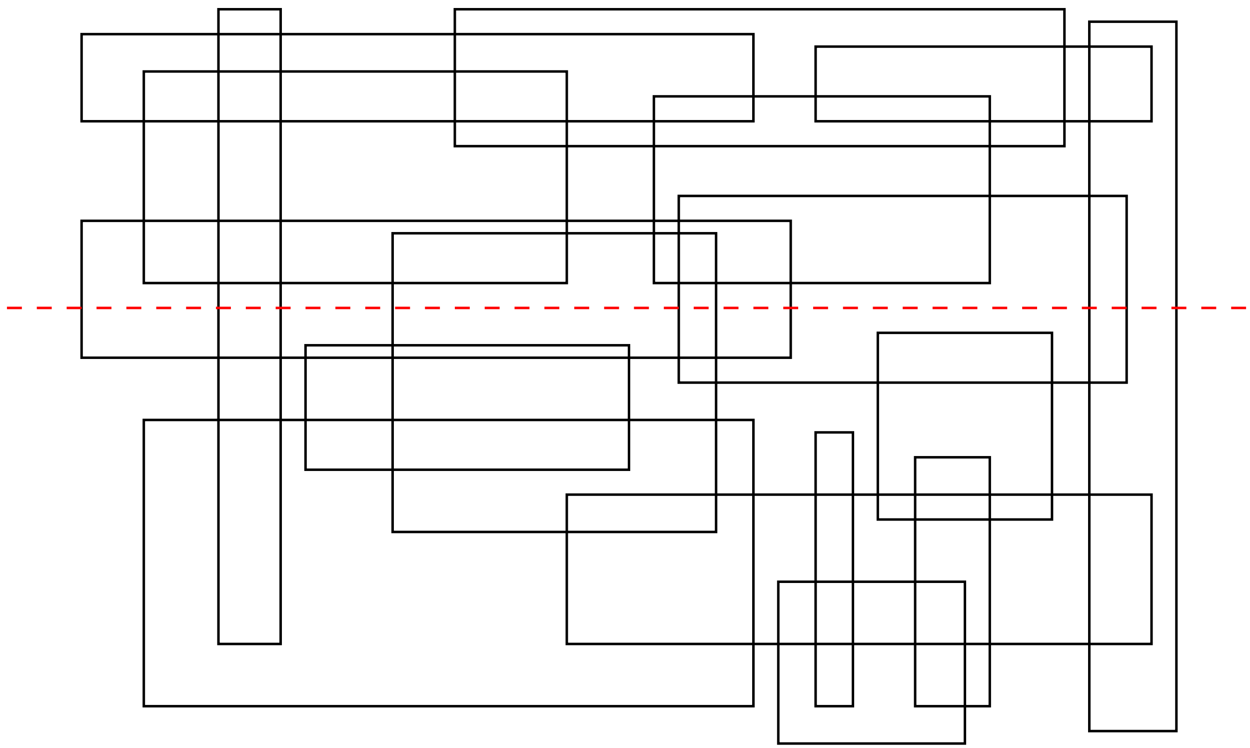
➥ Optimal number of colors OPT is at least $1 + \frac{|N(v)|}{5}$.
➥ $|N(v)| \leq 5 \cdot \mathrm{OPT} - 5$.
➥ So we must also have $D \leq 5\mathrm{OPT} - 5$.

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

Rectangle Intersection Graphs

MIS in Rectangle Graphs

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



Approximation algorithm for rectangles

Algorithm RECTANGLE-APPROX(set of rectangles R)
$\ell =$ stabbing line with at most $|R|/2$ rectangles above and below;
$R_{\mathrm{above}} =$ rectangles above stabbing line;
$R_{\mathrm{below}} =$ rectangles below stabbing line;
$R_{\mathrm{mid}} =$ rectangles intersecting stabbing line;
compute approximations I_1 and I_2 for R_{above} and R_{below} recursively;
compute optimal independent set I_0 for R_{mid};
return the larger of I_0 and $I_1 \cup I_2$;

Analysis of RECTANGLE-APPROX

Theorem 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 ℓ.
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

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

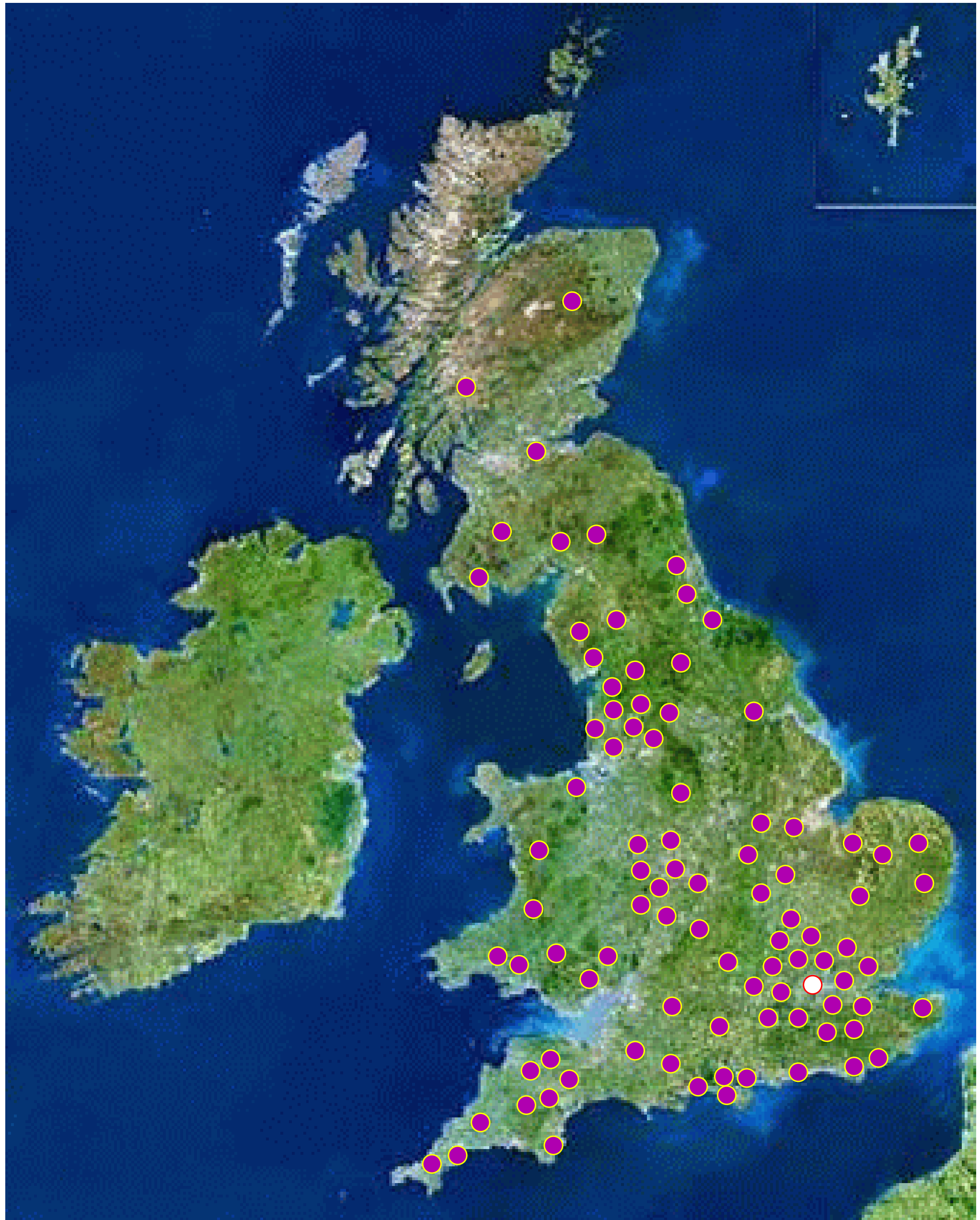
$\square$

MIS in rectangle graphs: Summary

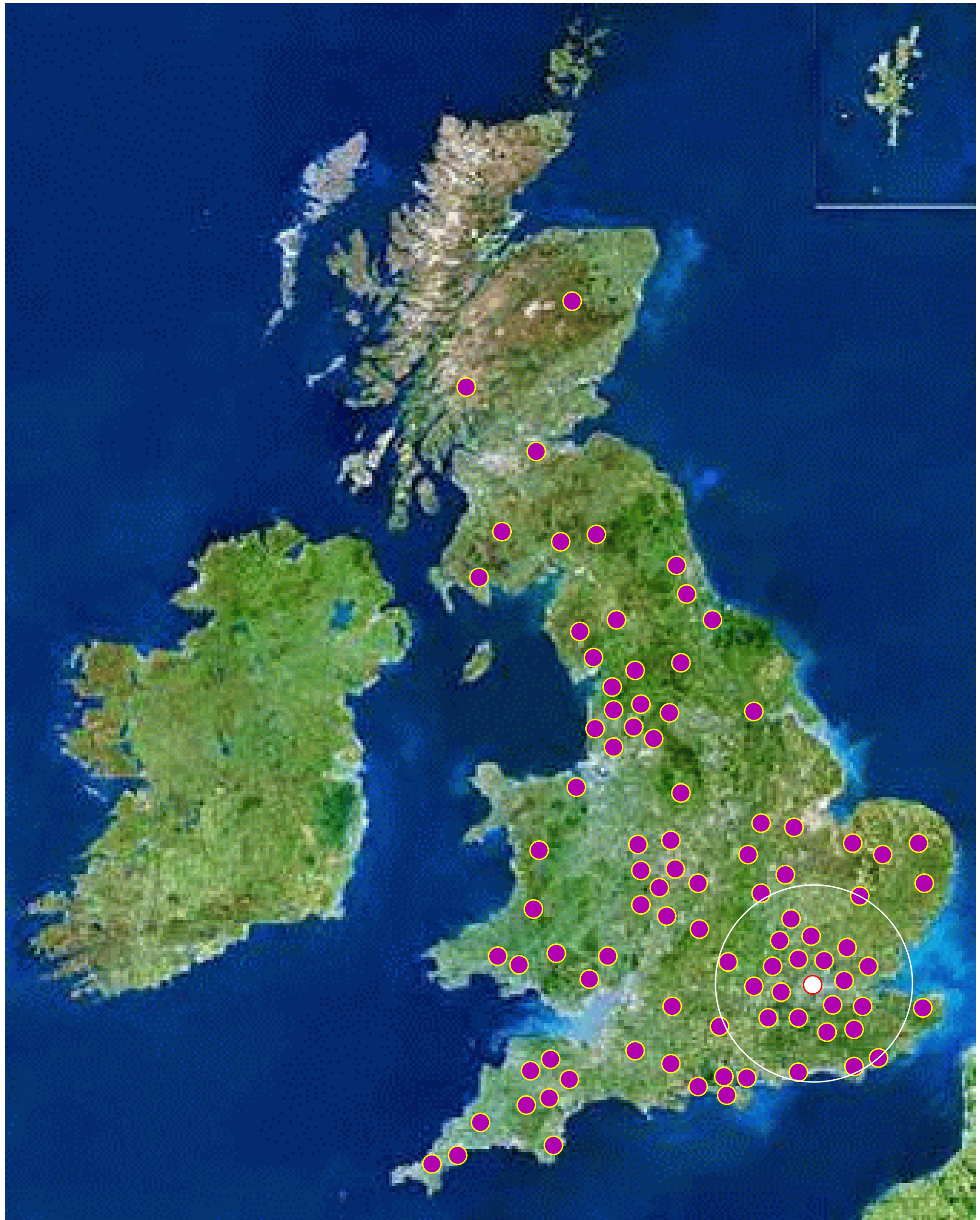
- There is an $O(\log n)$-approximation algorithm (with given representation).
  [Agarwal et al., 1998; Khanna et al. 1998; Nielsen 2000]
- For every constant $c > 0$, there is an approximation algorithm with ratio $1 + \frac{1}{c} \log n$.
  [Berman et al., 2001]
- If all rectangles have the same height, there is a PTAS.
  [Agarwal et al., 1998]

Minimum Dominating Set

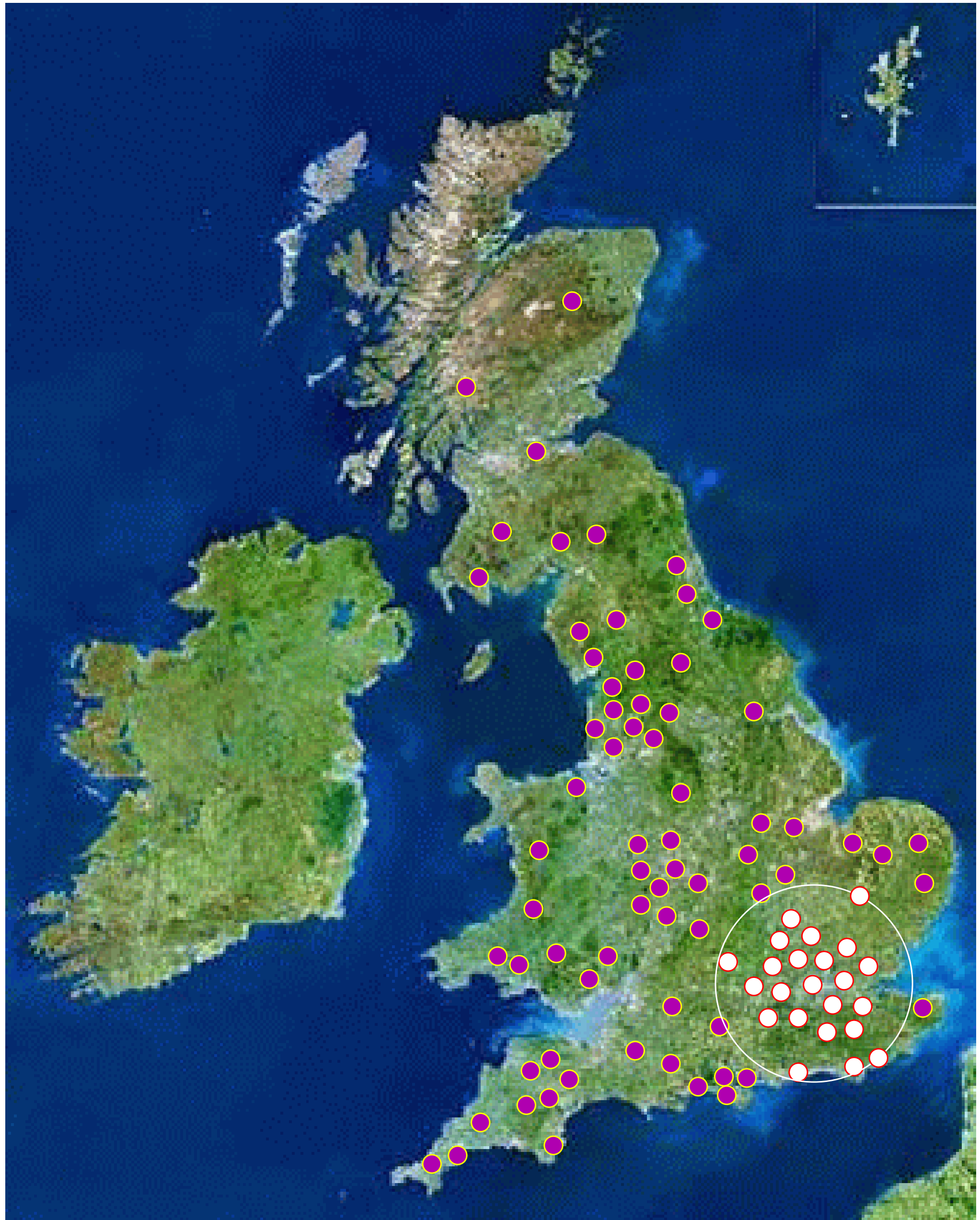
Flooding an Ad-Hoc Network



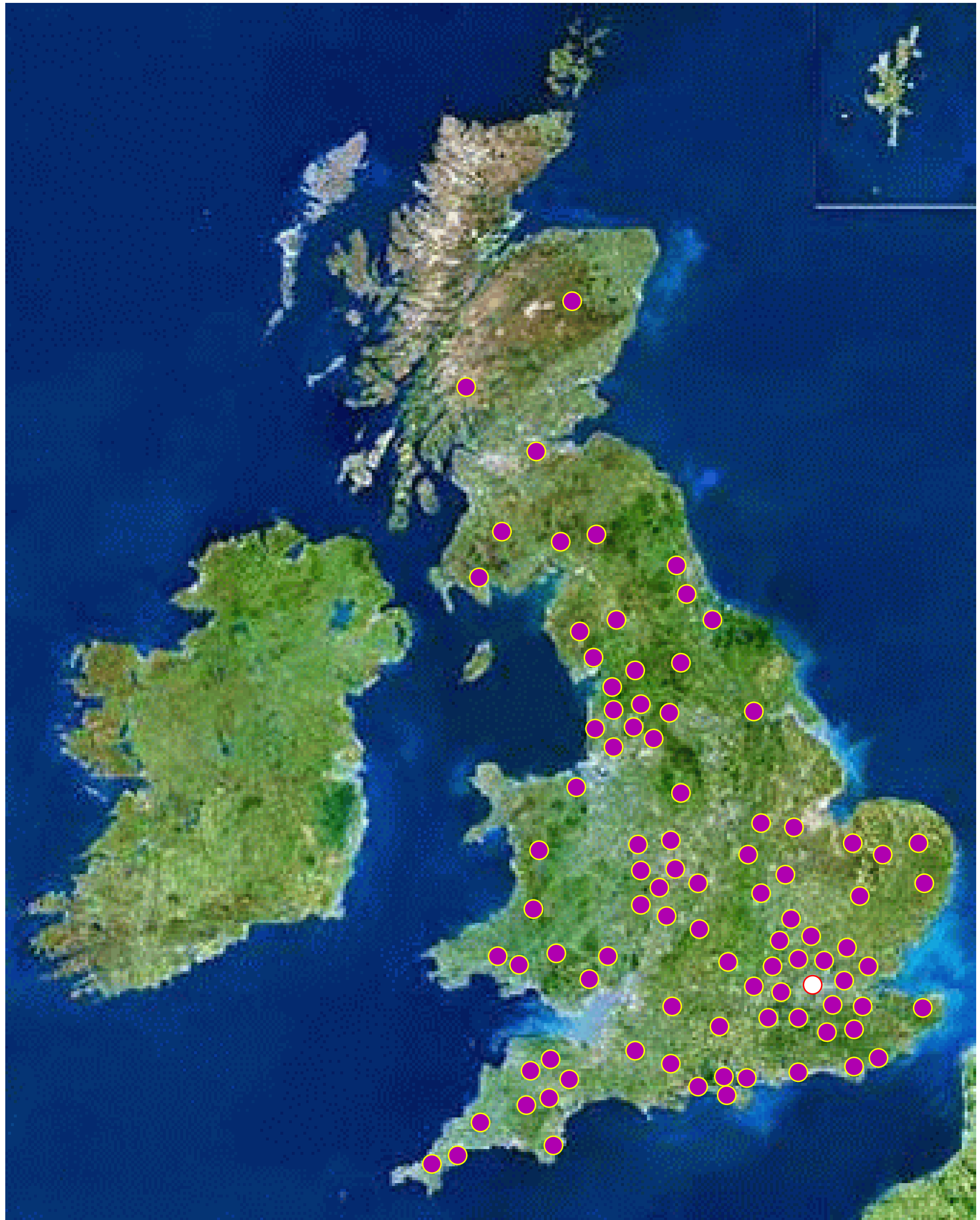
Flooding an Ad-Hoc Network



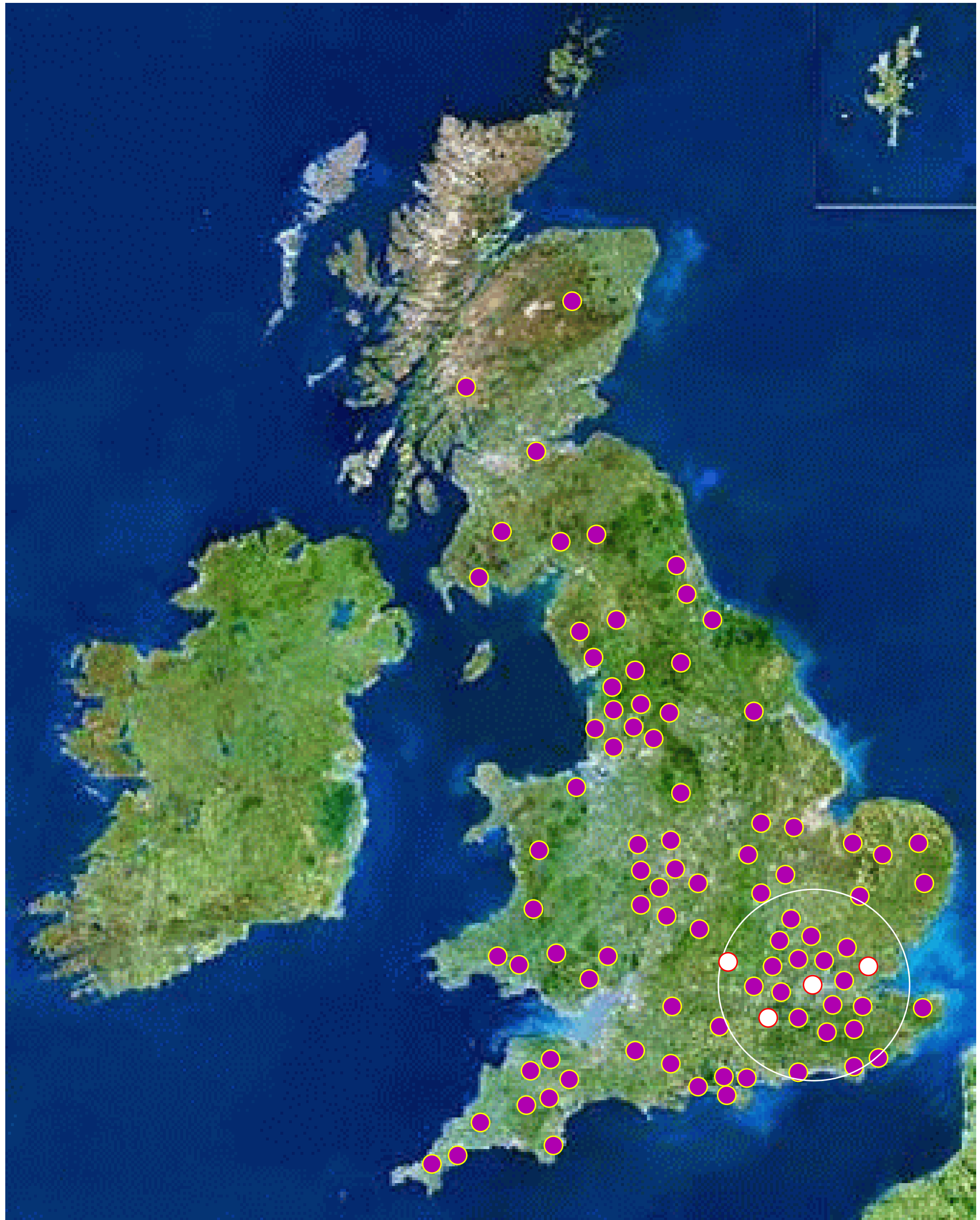
Flooding an Ad-Hoc Network



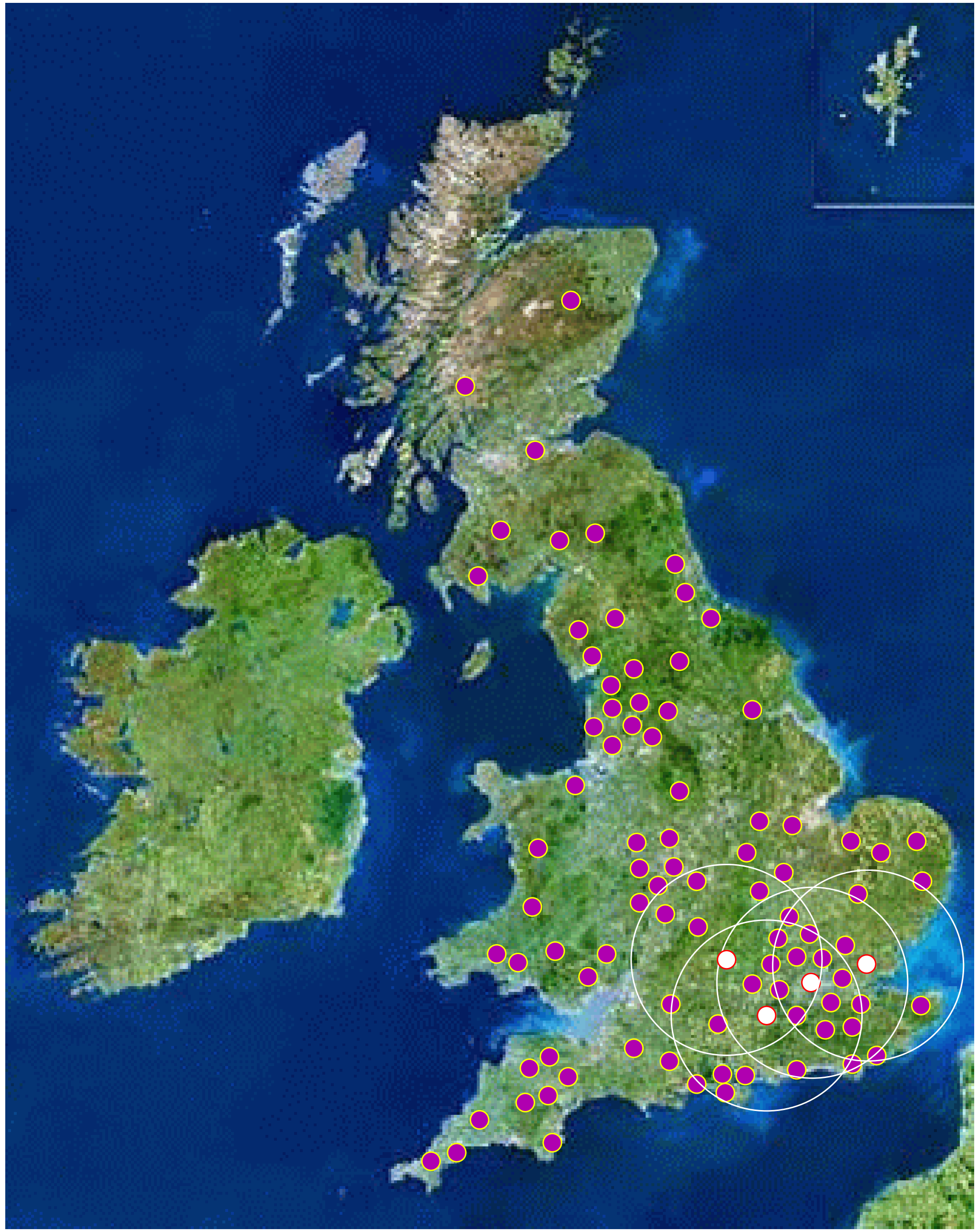
Efficient Flooding



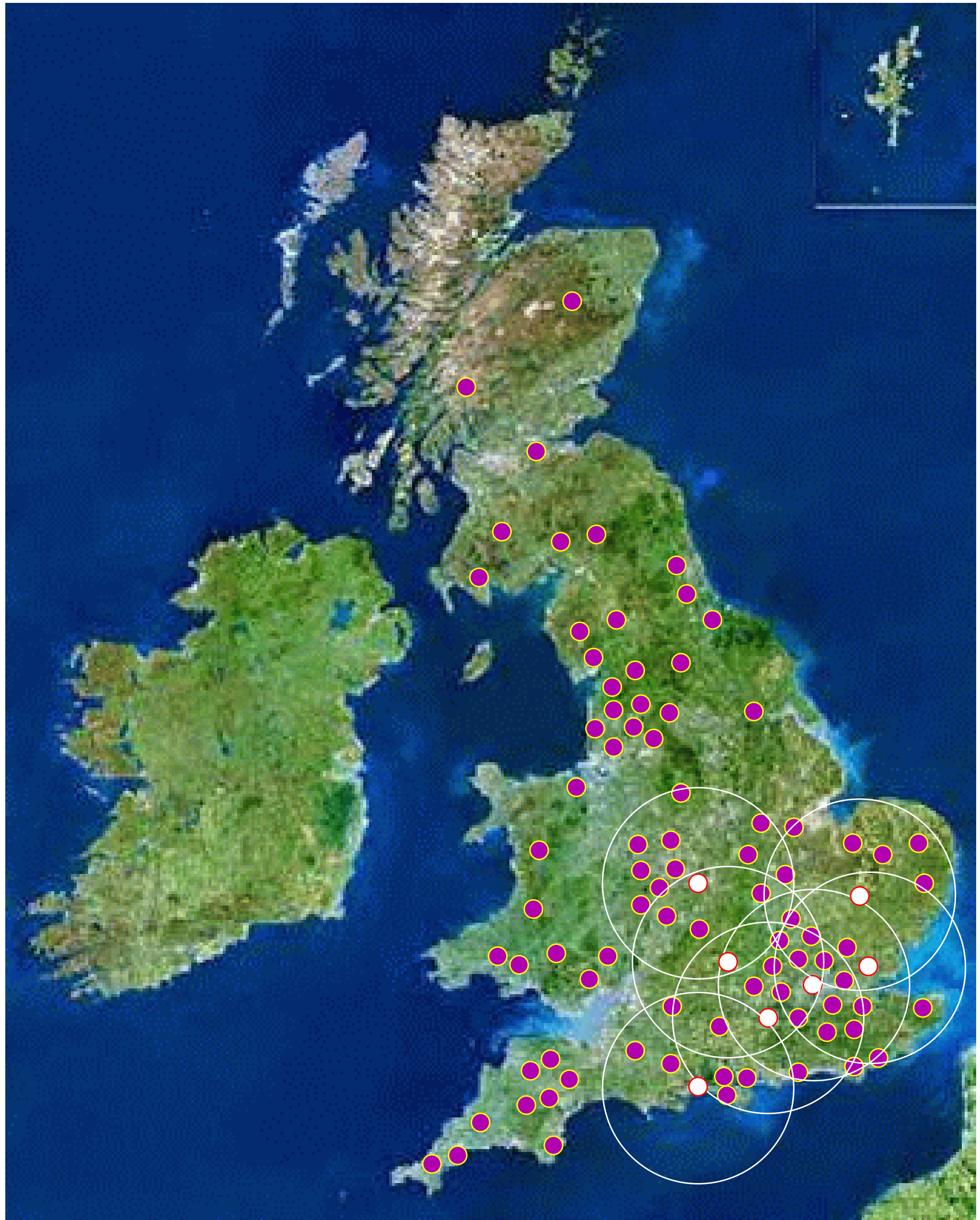
Efficient Flooding



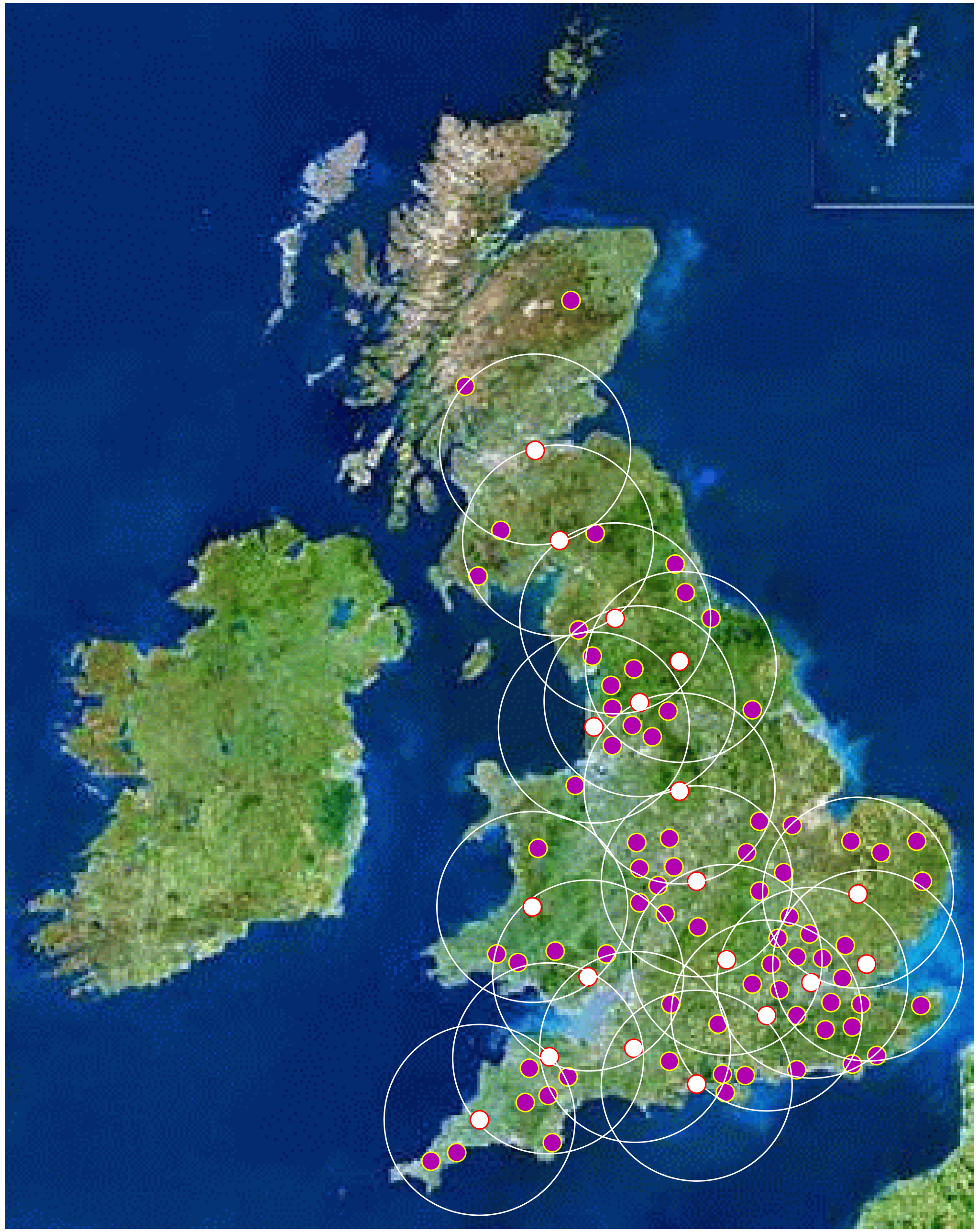
Efficient Flooding



Efficient Flooding



Efficient Flooding

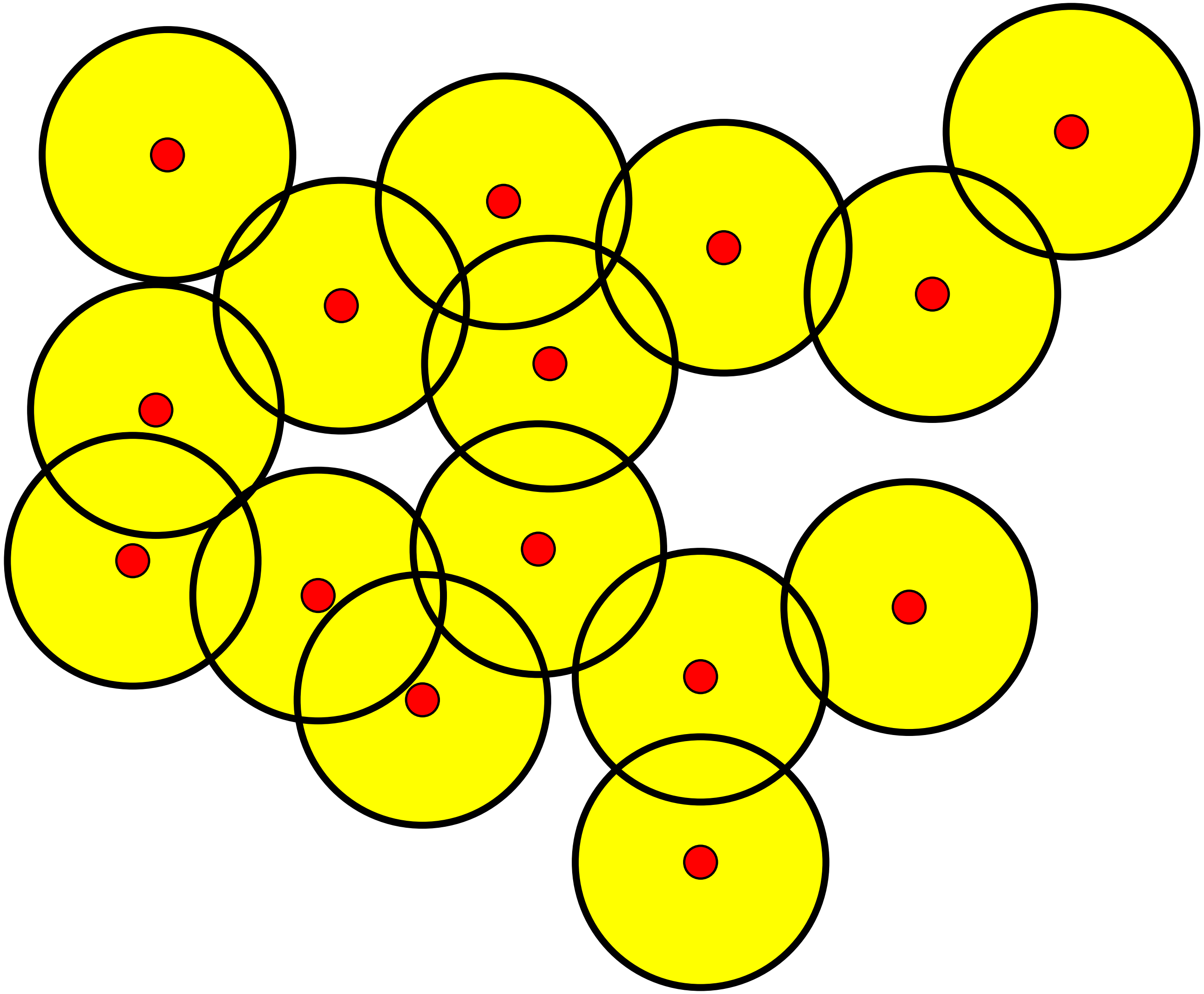


Routing Backbone

- For efficient flooding, we want to find a small subset of the nodes that can reach all other nodes. That subset is then the **routing backbone**. [Guha and Khuller, 1999]
- We can model the network as a graph.
  - Simple model: **Unit Disk Graph**
    Two nodes can reach each other if their distance is at most d, for some fixed value d.

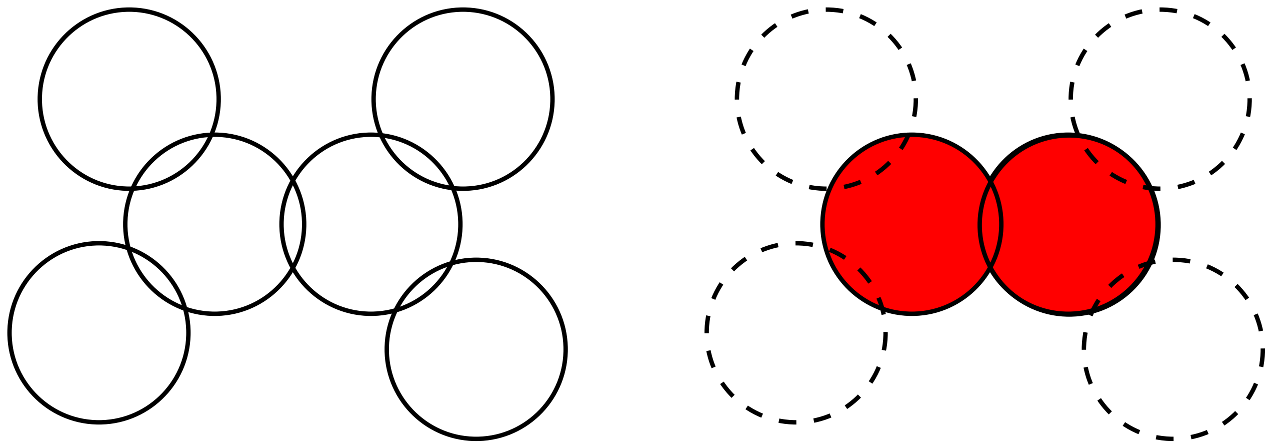
    Each node corresponds to a unit disk, and there is an edge between two nodes if the disks intersect.
- The problem of identifying a small routing backbone then becomes the minimum (connected) dominating set problem in unit disk graphs.

Unit Disk Graph



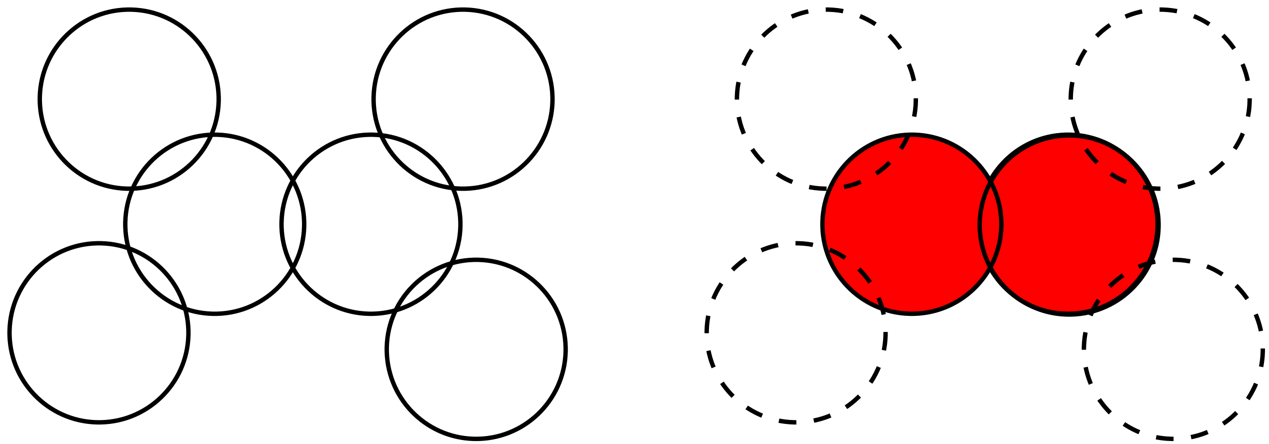
Minimum Dominating Set (MDS)

Input: a set $\mathcal{D}$ of unit disks in the plane
Feasible solution: subset $A \subseteq \mathcal{D}$ that dominates all disks
Goal: minimize $|A|$



Minimum Dominating Set (MDS)

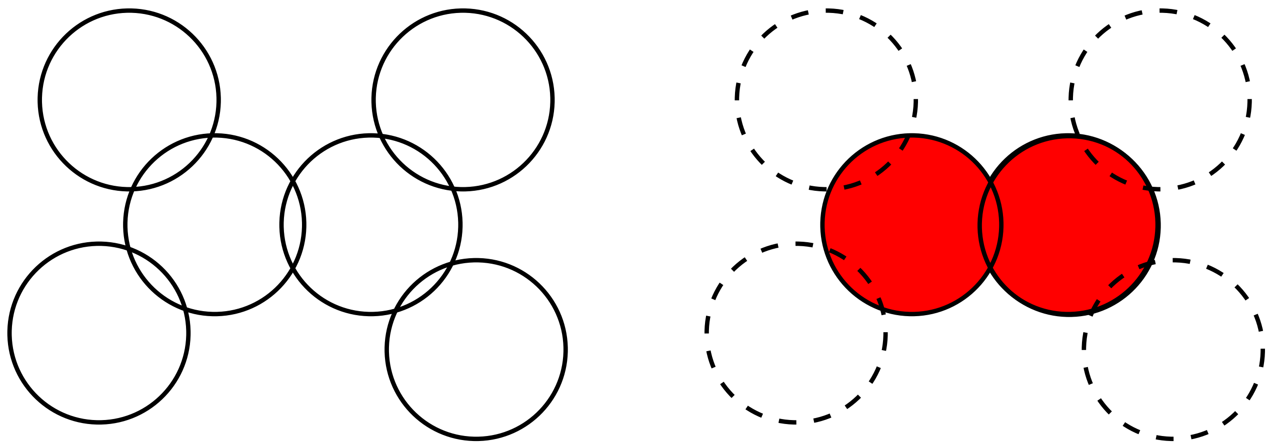
Input: a set $\mathcal{D}$ of unit disks in the plane
Feasible solution: subset $A \subseteq \mathcal{D}$ that dominates all disks
Goal: minimize $|A|$



In the weighted case (MWDS), each disk is associated with a positive weight.

Minimum Dominating Set (MDS)

Input: a set $\mathcal{D}$ of unit disks in the plane
Feasible solution: subset $A \subseteq \mathcal{D}$ that dominates all disks
Goal: minimize $|A|$



In the weighted case (MWDS), each disk is associated with a positive weight.

For Minimum (Weight) Connected Dominating Set (MCDS/MWCDS), the dominating set must induce a connected subgraph.

Approximation Algorithms

An algorithm for MWDS is a ρ-approximation algorithm if it runs in polynomial time and always outputs a solution of weight at most $\rho \cdot \mathrm{OPT}$, where OPT is the weight of an optimal solution.

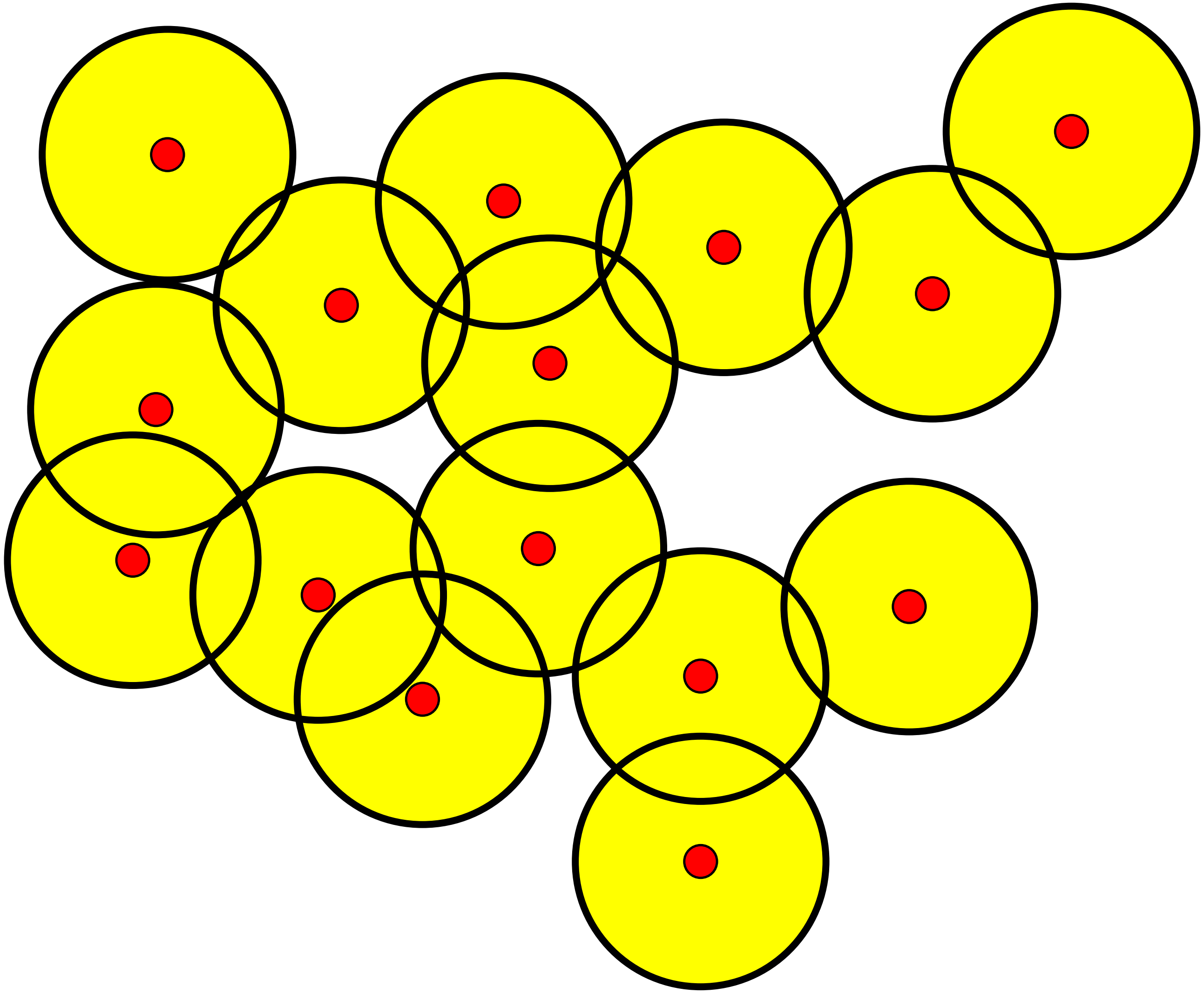
A polynomial-time approximation scheme (PTAS) is a family of algorithms containing a $(1 + \varepsilon)$-approximation algorithm for every fixed $\varepsilon > 0$.

Remark: In practice, we are interested in distributed algorithms with fast running-time and good performance in realistic scenarios.

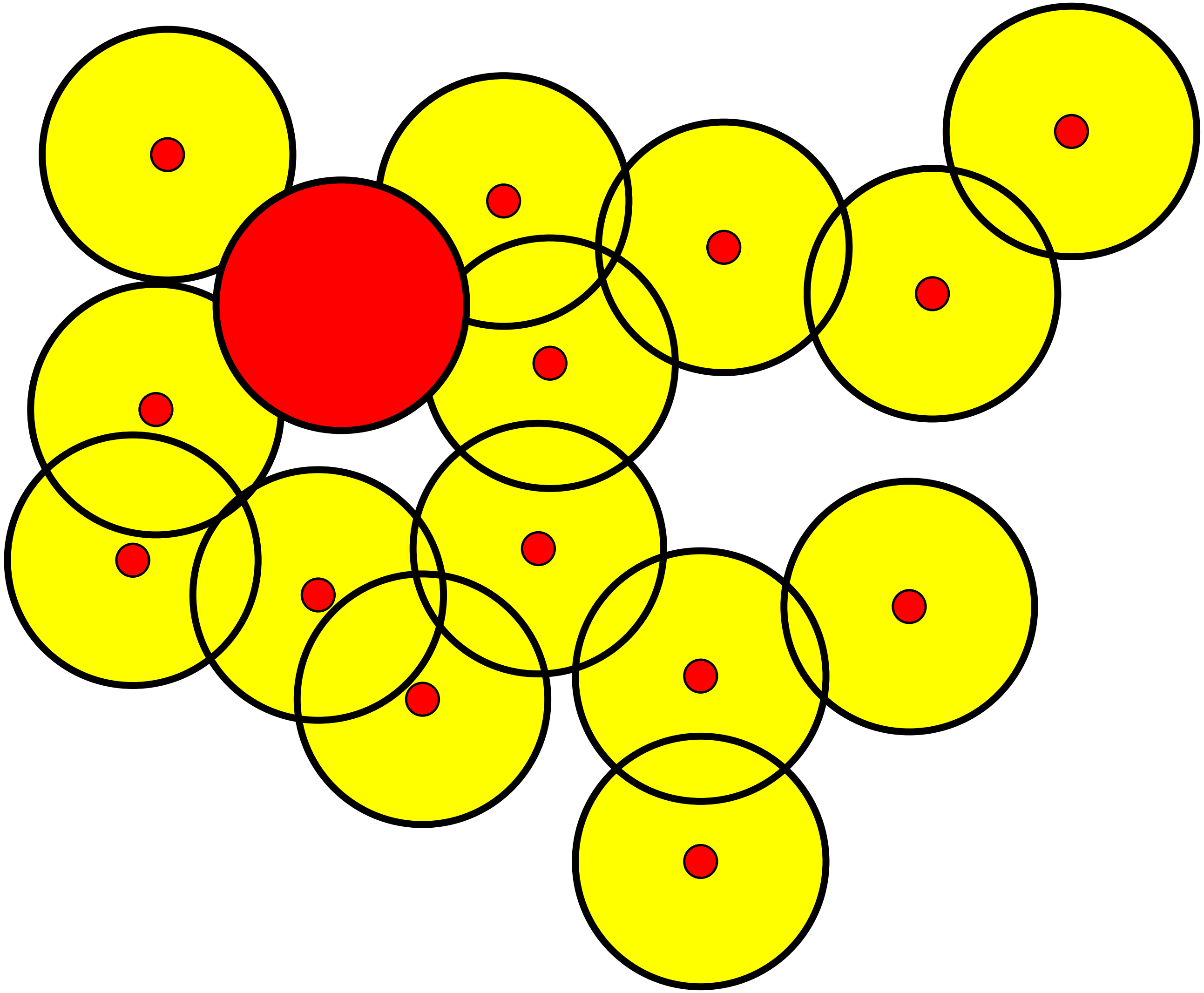
A simple algorithm for MDS

- Initialise $\mathcal{U}$ as the empty set.
- Repeat until no disk left:
  - pick an arbitrary disk D
  - insert D into the set $\mathcal{U}$
  - delete the disk D and all its neighbours from the instance
- Output the set $\mathcal{U}$ as dominating set

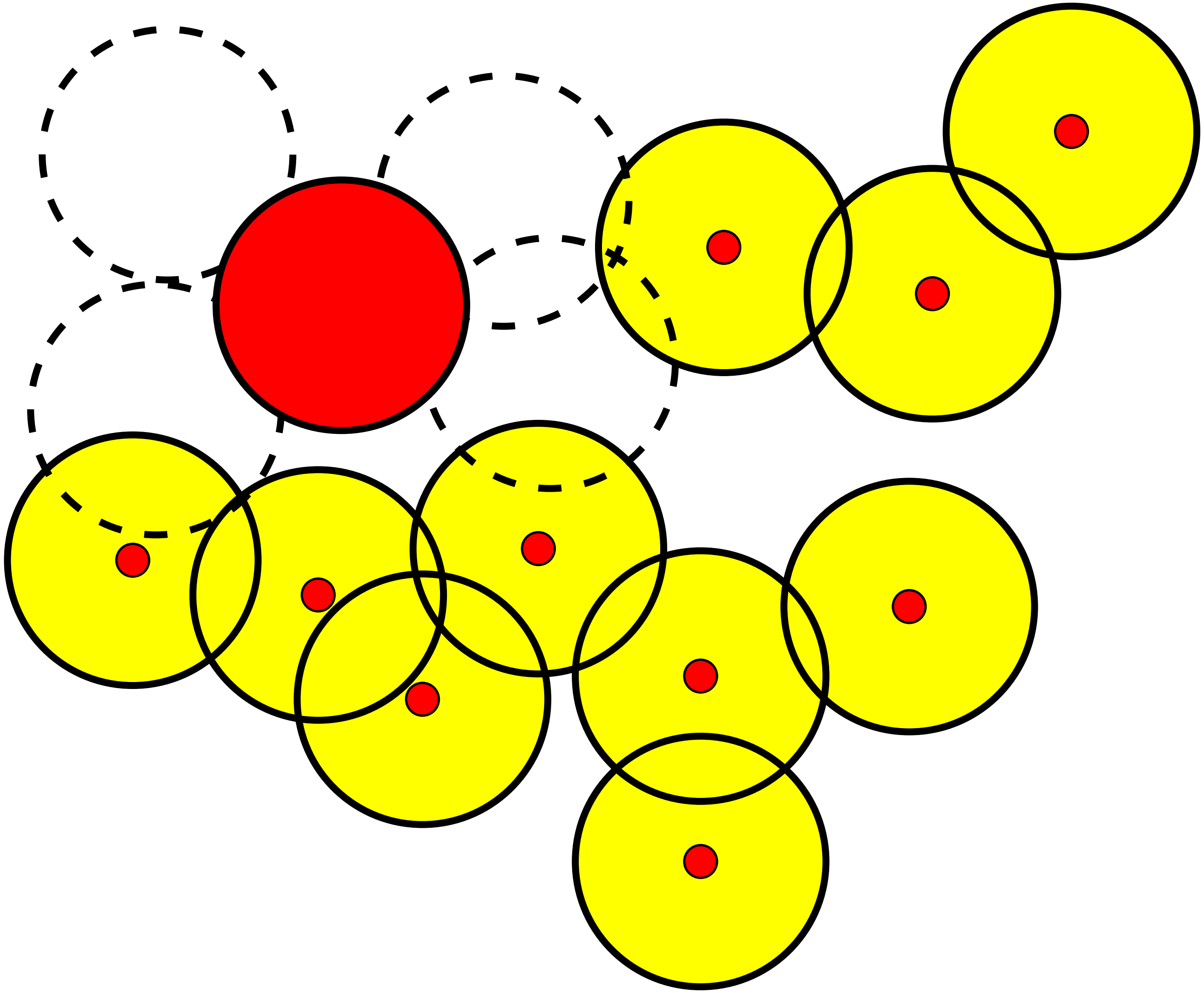
Example run



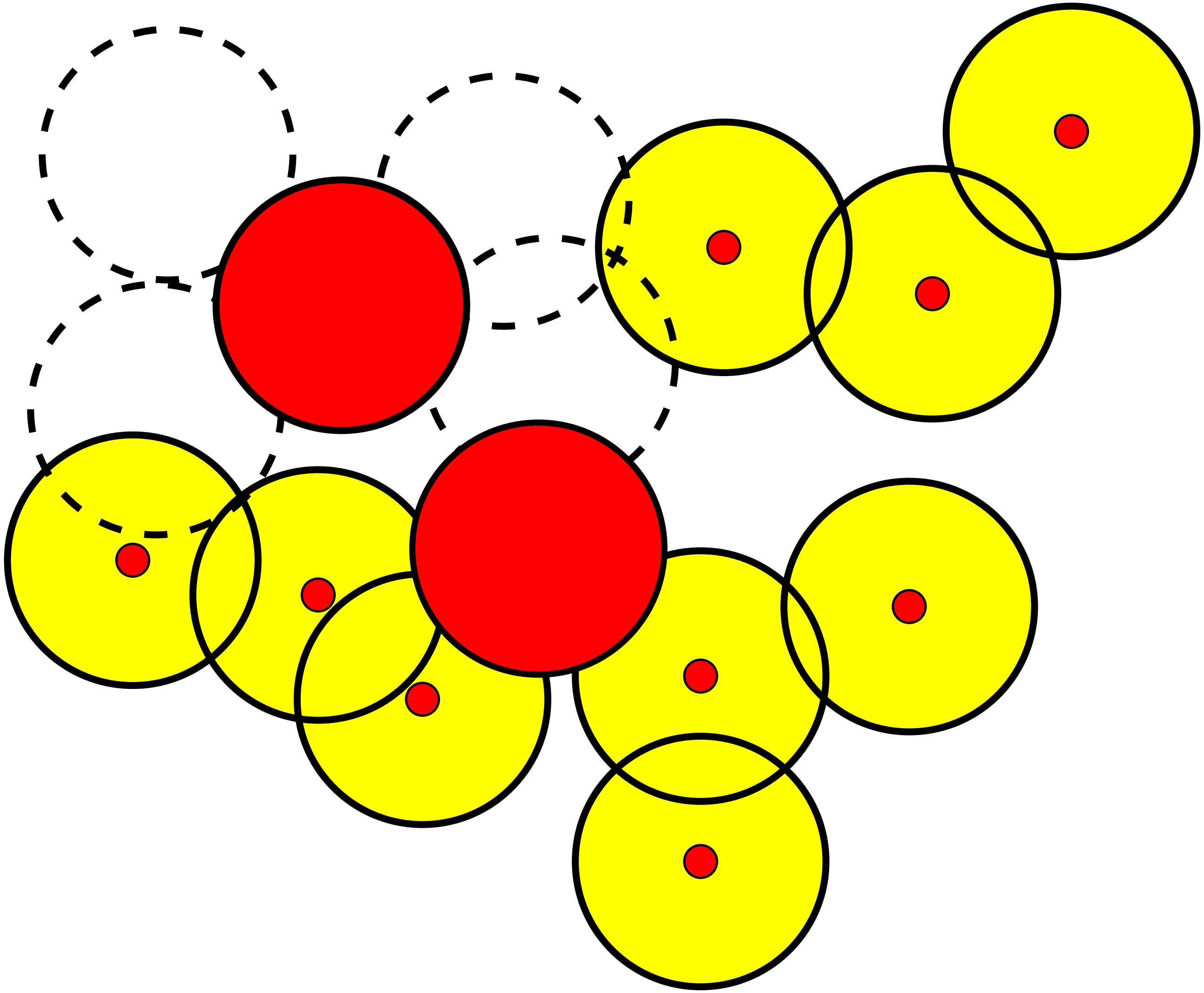
Example run



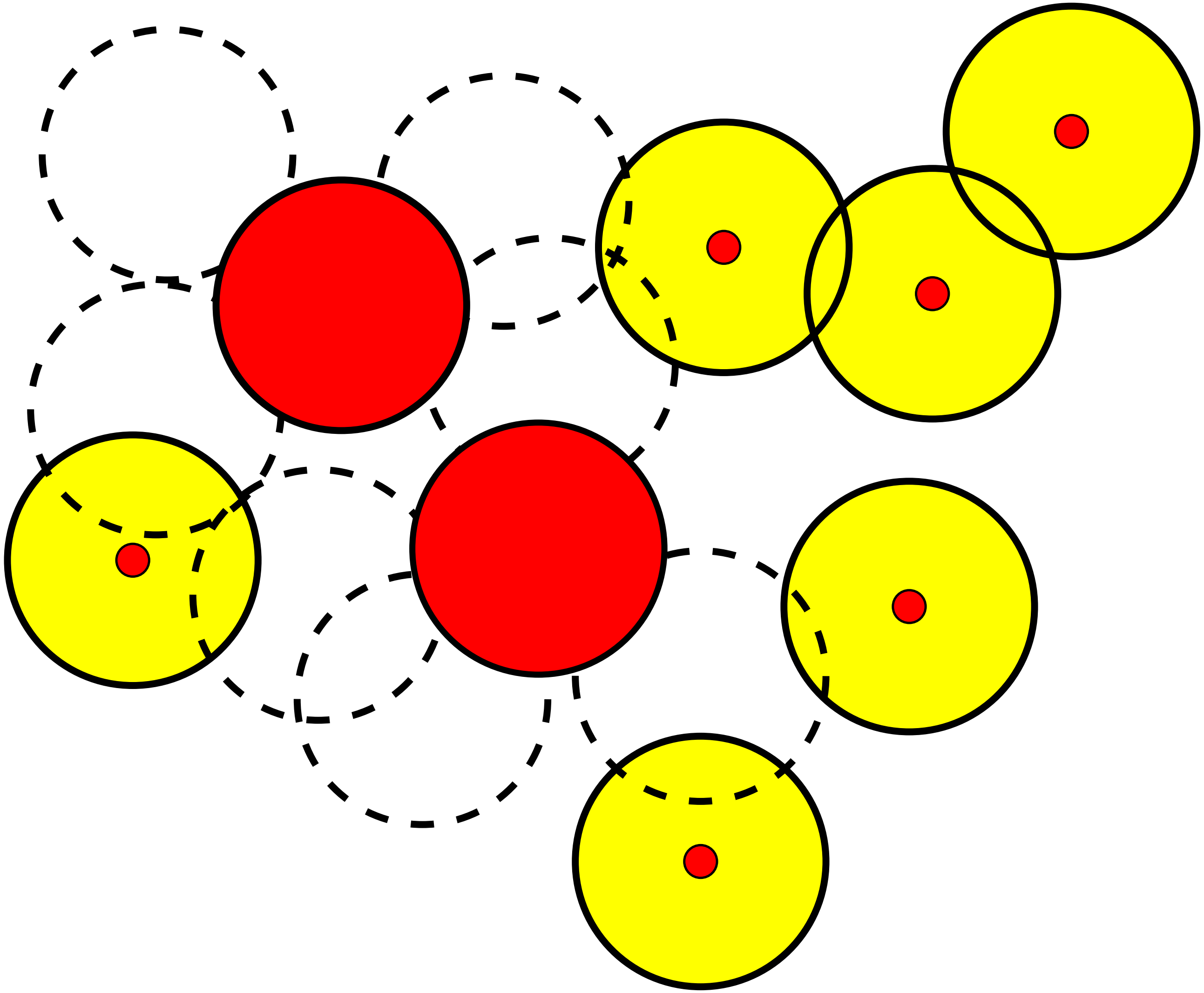
Example run



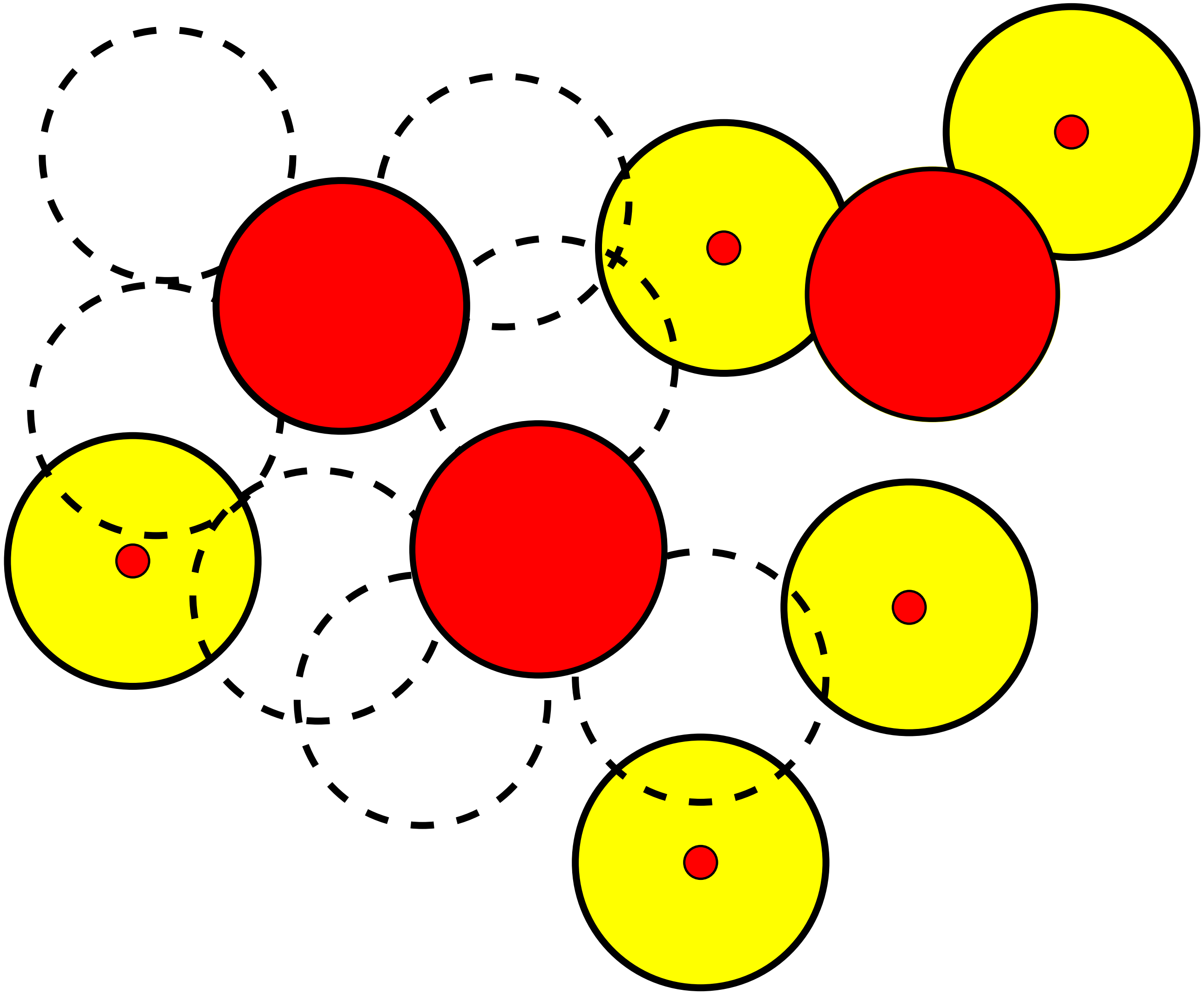
Example run



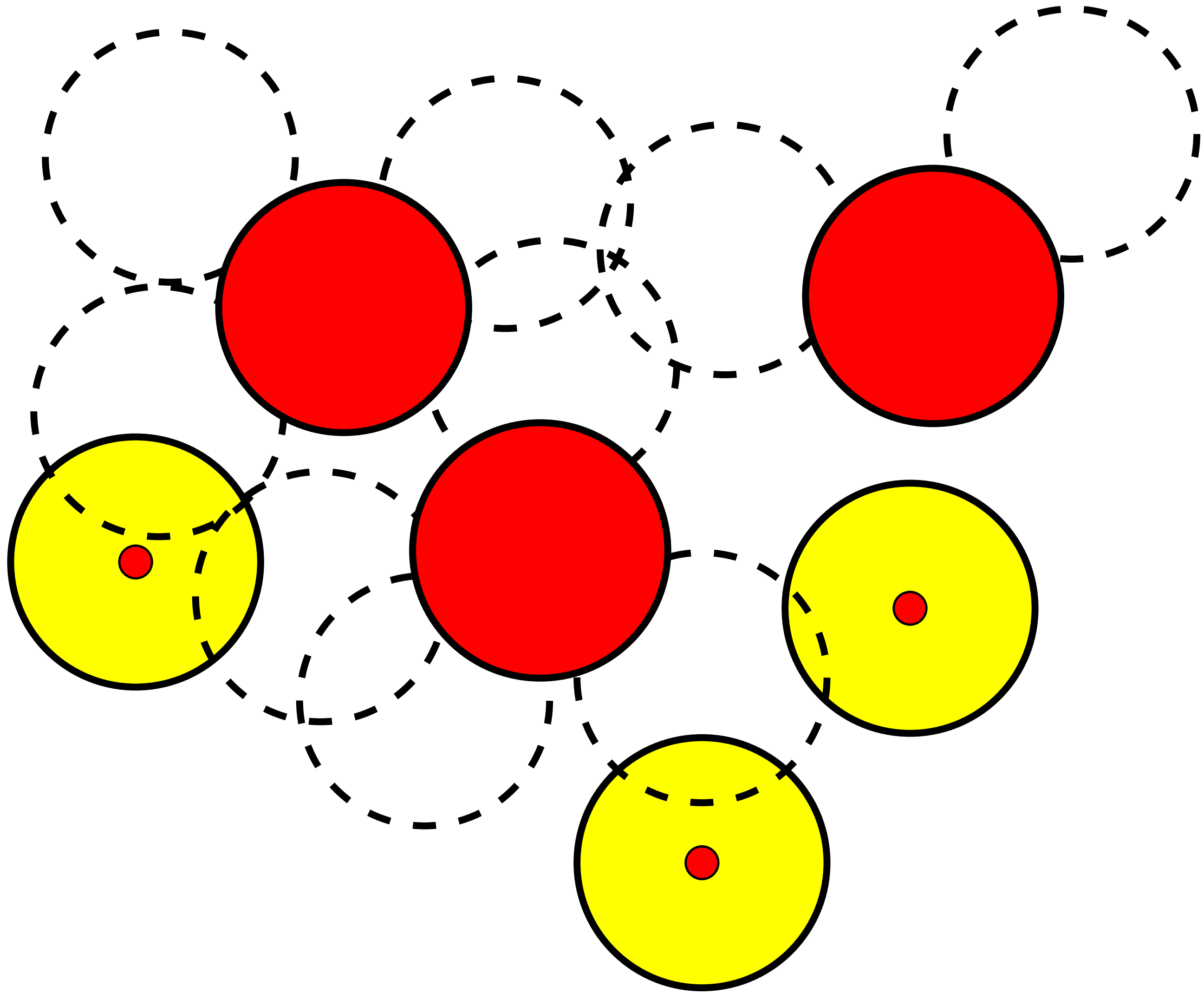
Example run



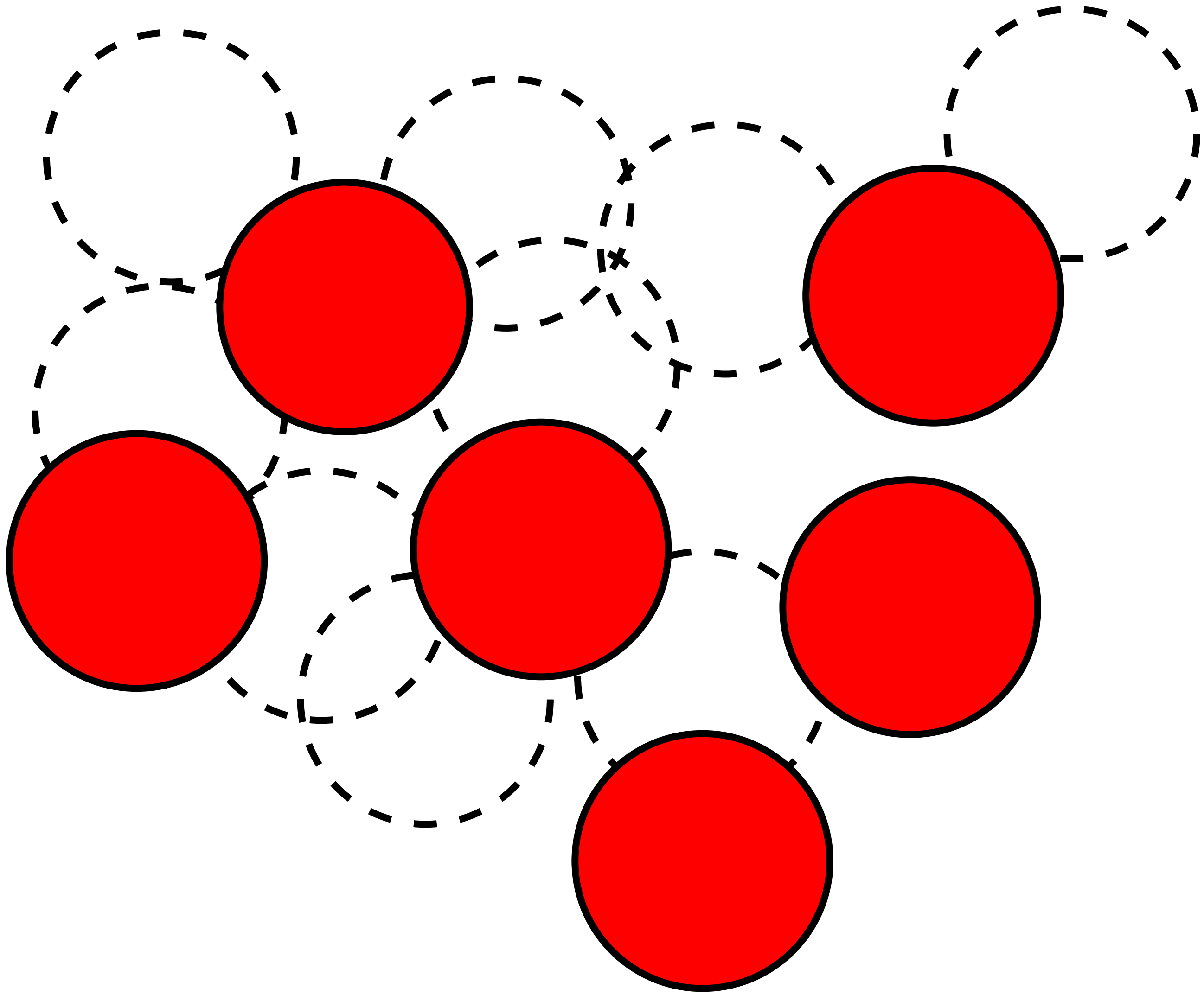
Example run



Example run



Example run



Analysis of the algorithm

- How much worse than the optimal dominating set can the solution produced by this algorithm be?

Analysis of the algorithm

- How much worse than the optimal dominating set can the solution produced by this algorithm be?
- The set $\mathcal{U}$ output by the algorithm consists of disjoint disks.

Analysis of the algorithm

- How much worse than the optimal dominating set can the solution produced by this algorithm be?
- The set $\mathcal{U}$ output by the algorithm consists of disjoint disks.
- The optimal solution also needs to dominate all disks in $\mathcal{U}$.

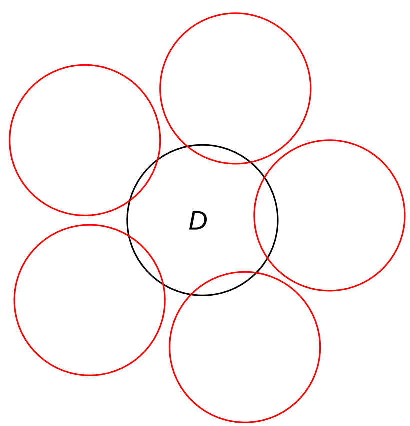
Analysis of the algorithm

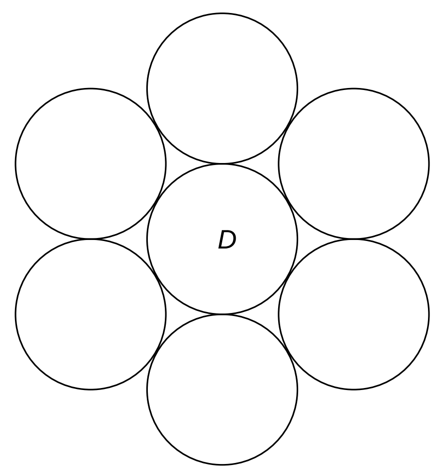
- How much worse than the optimal dominating set can the solution produced by this algorithm be?
- The set $\mathcal{U}$ output by the algorithm consists of disjoint disks.
- The optimal solution also needs to dominate all disks in $\mathcal{U}$.
- How many disks in $\mathcal{U}$ can one disk D from the optimal solution dominate?

Analysis of the algorithm

- How much worse than the optimal dominating set can the solution produced by this algorithm be?
- The set $\mathcal{U}$ output by the algorithm consists of disjoint disks.
- The optimal solution also needs to dominate all disks in $\mathcal{U}$.
- How many disks in $\mathcal{U}$ can one disk D from the optimal solution dominate?

At most 5:





Simple approximation results

The algorithm outputs the set $|\mathcal{U}|$, and the optimal solution has size at least $|\mathcal{U}|/5$.

Simple approximation results

The algorithm outputs the set $|\mathcal{U}|$, and the optimal solution has size at least $|\mathcal{U}|/5$.

Theorem (Marathe et al., 1992)
This simple greedy algorithm is a 5-approximation algorithm for MDS in unit disk graphs.

Simple approximation results

The algorithm outputs the set $|\mathcal{U}|$, and the optimal solution has size at least $|\mathcal{U}|/5$.

Theorem (Marathe et al., 1992)
This simple greedy algorithm is a 5-approximation algorithm for MDS in unit disk graphs.

Theorem (Marathe et al., 1992)
There is a simple 10-approximation algorithm for MCDS in unit disk graphs.

Simple approximation results

The algorithm outputs the set $|\mathcal{U}|$, and the optimal solution has size at least $|\mathcal{U}|/5$.

Theorem (Marathe et al., 1992)
This simple greedy algorithm is a 5-approximation algorithm for MDS in unit disk graphs.

Theorem (Marathe et al., 1992)
There is a simple 10-approximation algorithm for MCDS in unit disk graphs.

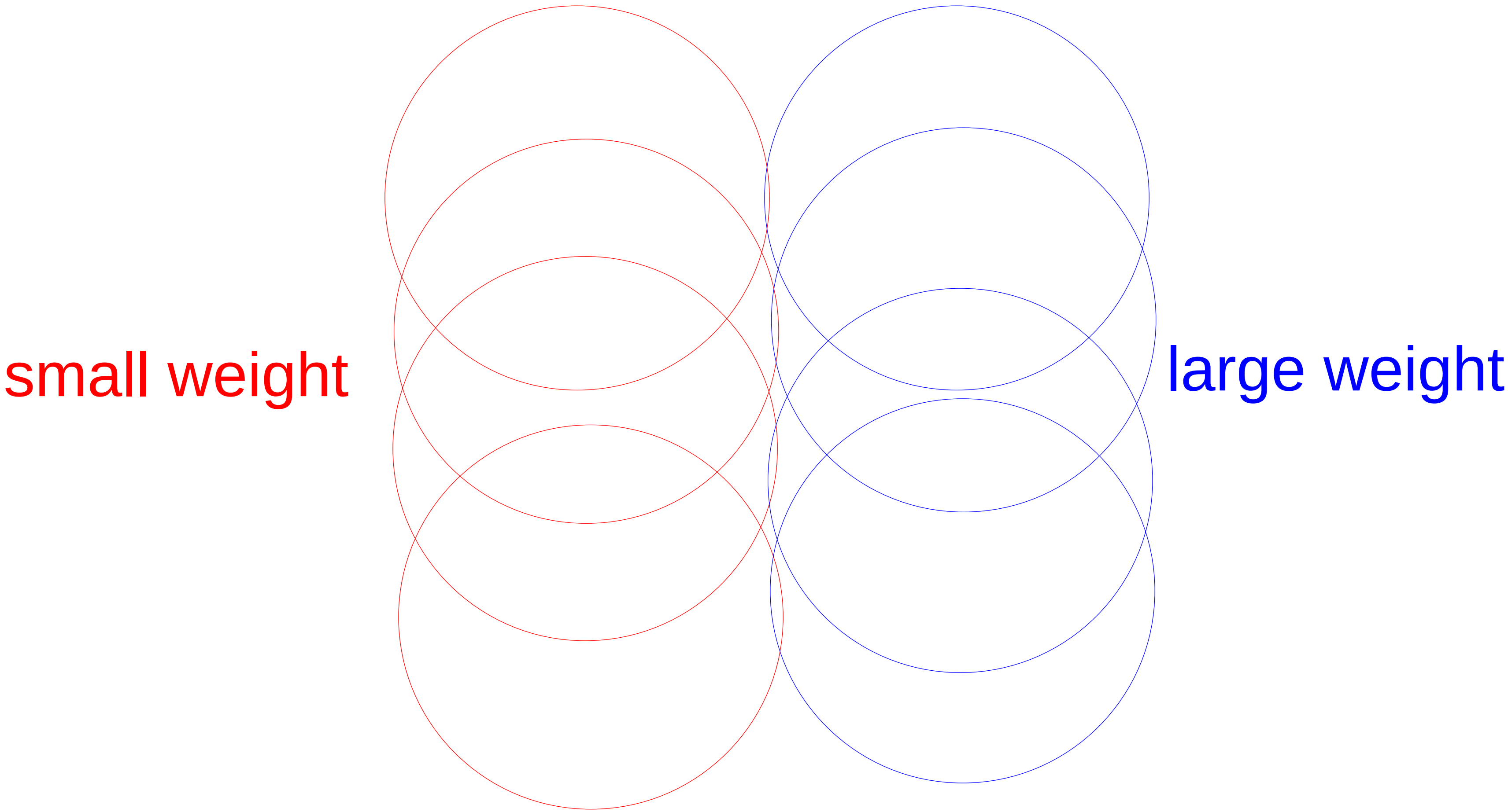
Remark: There are also fast distributed approximation algorithms for dominating set problems.
(Kuhn & Wattenhofer, 2005)

Known dom. set approximations

- In **arbitrary graphs**, ratio $\Theta(\log n)$ is best possible (unless $P = NP$) for MDS, MWDS, MCDS and MWCDS. [Feige '96; Arora and Sudan '97; Guha and Khuller '99]
- For **MDS in unit disk graphs**, a PTAS can be obtained using the shifting strategy [Hunt III et al., 1994]:
  - Any maximal independent set is a dominating set.
  - Therefore, the smallest dominating set in a constant-size square can be found in polynomial time by enumeration.
- PTAS for **MDS in unit disk graphs without representation** [Nieberg and Hurink, 2005]
- PTAS for **MCDS in unit disk graphs** [Cheng et al., 2003]
- **Question:** MWDS and MWCDS in unit disk graphs?

Shifting strategy doesn't seem to work

MWDS can be arbitrarily large for unit disks in an area of constant size:



➠ Brute-force enumeration does no longer work.

Constant-Factor Approximation

Theorem (Ambühl, E, Mihal'ák, Nunkesser, 2006) There is a constant-factor approximation algorithm for MWDS in unit disk graphs.

Ideas:

- Partition the plane into unit squares and solve the problem for each square separately.
- In each square, reduce the problem to the problem of covering points with weighted disks.
- Use enumeration techniques (guess properties of OPT) and dynamic programming to solve the latter problem.

The constant factor is 72.

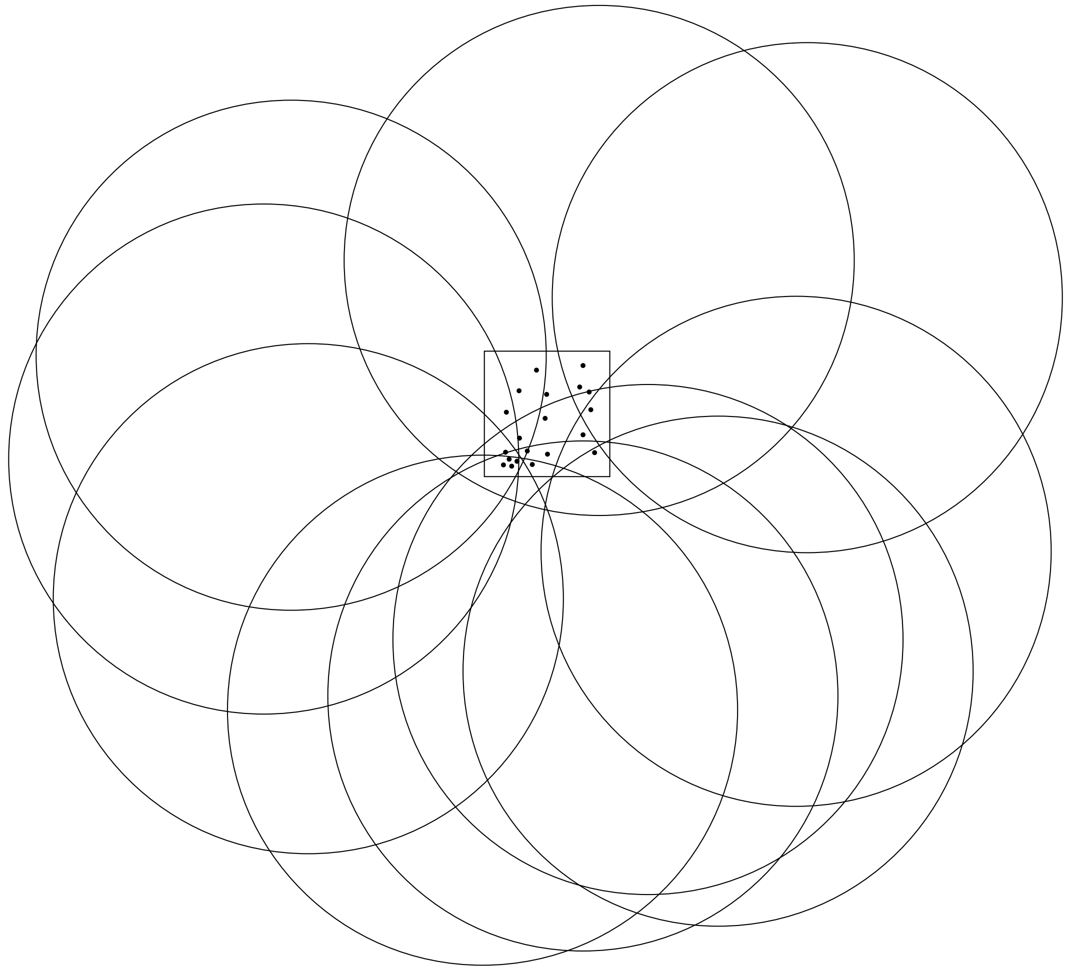
The subproblem for each square

- Find a dominating set for the square:
  - Let $\mathcal{D}_S$ denote the set of disks with center in a 1×1 square S.
  - Let $N(\mathcal{D}_S)$ denote the disks in $\mathcal{D}_S$ and their neighbors.
  - **Task:** Find a minimum weight set of disks in $N(\mathcal{D}_S)$ that dominates all disks in $\mathcal{D}_S$.

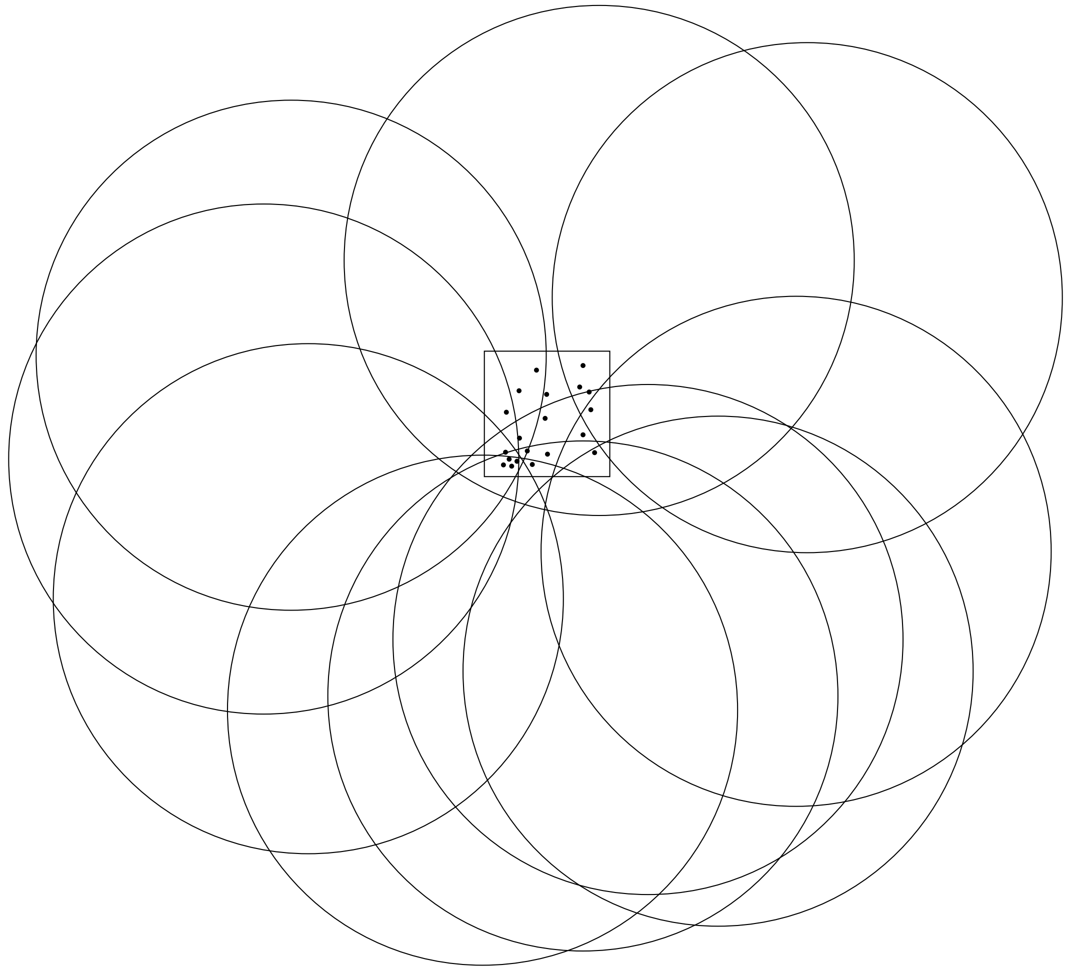
The subproblem for each square

- Find a dominating set for the square:
  - Let $\mathcal{D}_S$ denote the set of disks with center in a 1×1 square S.
  - Let $N(\mathcal{D}_S)$ denote the disks in $\mathcal{D}_S$ and their neighbors.
  - **Task:** Find a minimum weight set of disks in $N(\mathcal{D}_S)$ that dominates all disks in $\mathcal{D}_S$.
- Reduces (by guessing the max weight of a disk in OPT$_S$) to covering points in a square with weighted disks:
  - Let P be a set of points in a $\frac{1}{2} \times \frac{1}{2}$ square S.
  - Let $\mathcal{D}$ be a set of weighted unit disks covering P.
  - **Task:** Find a minimum weight set of disks in $\mathcal{D}$ that covers all points in P.

Covering points by weighted disks



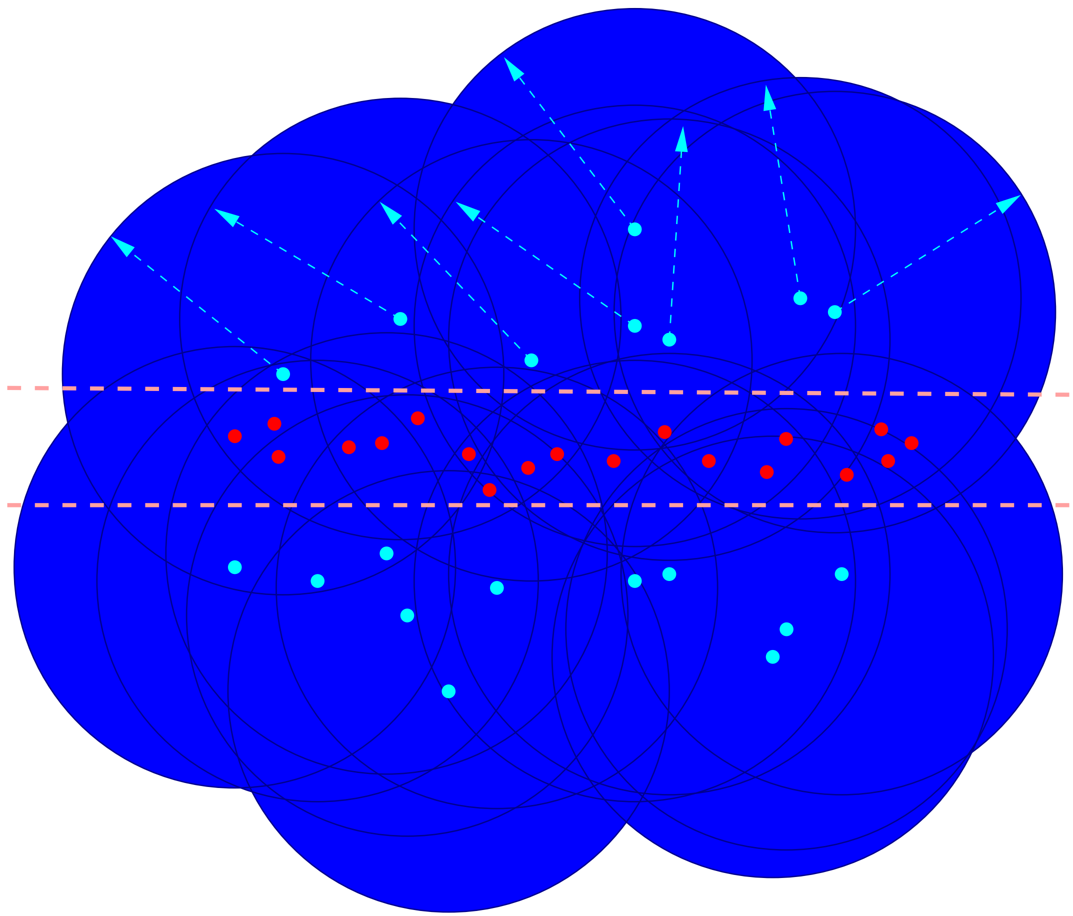
Covering points by weighted disks



Remark. $O(1)$-approximation algorithms are known for unweighted disk cover [Brönninmann and Goodrich, 1995].

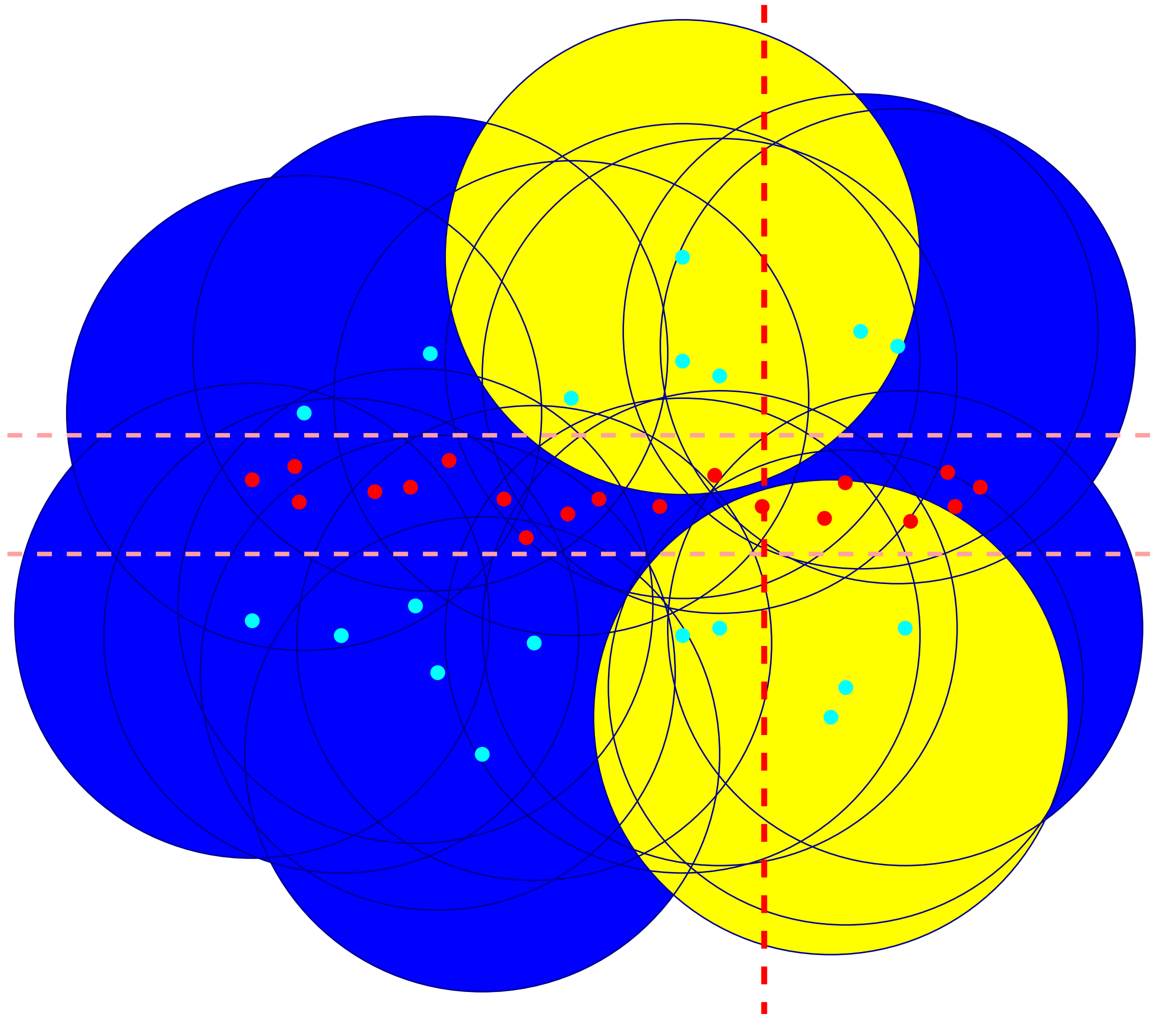
Polynomial-time solvable subproblem

- Given a set of points in a strip, and a set of weighted unit disks with centers outside the strip, compute a minimum weight set of disks covering the points.



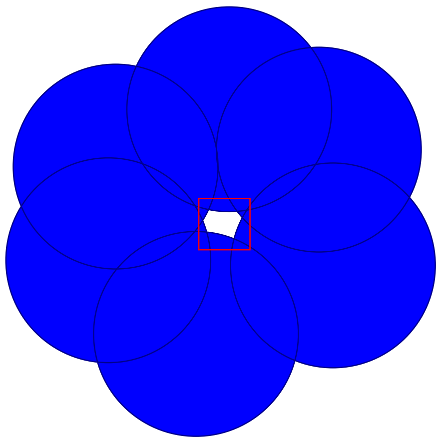
Dynamic programming

- Vertical sweepline, table entry for every pair of disks that could be on the lower and upper envelope:

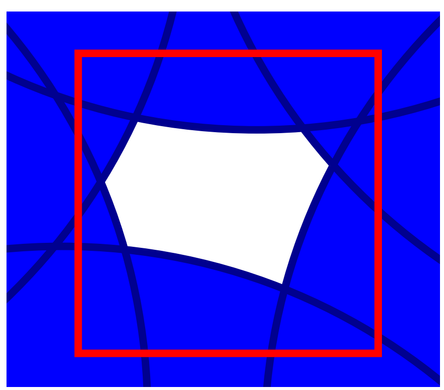


Main cases: One hole or many holes

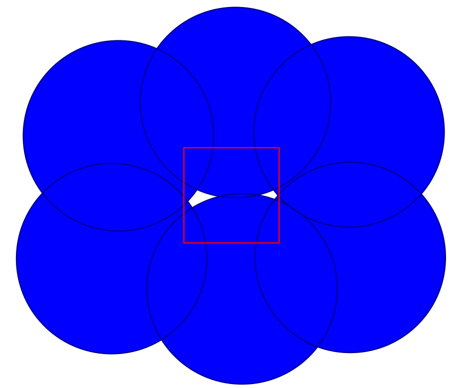
One-hole case:



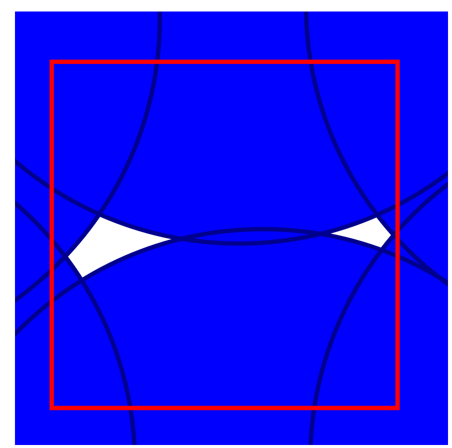
Enlarged:



Many-holes case:

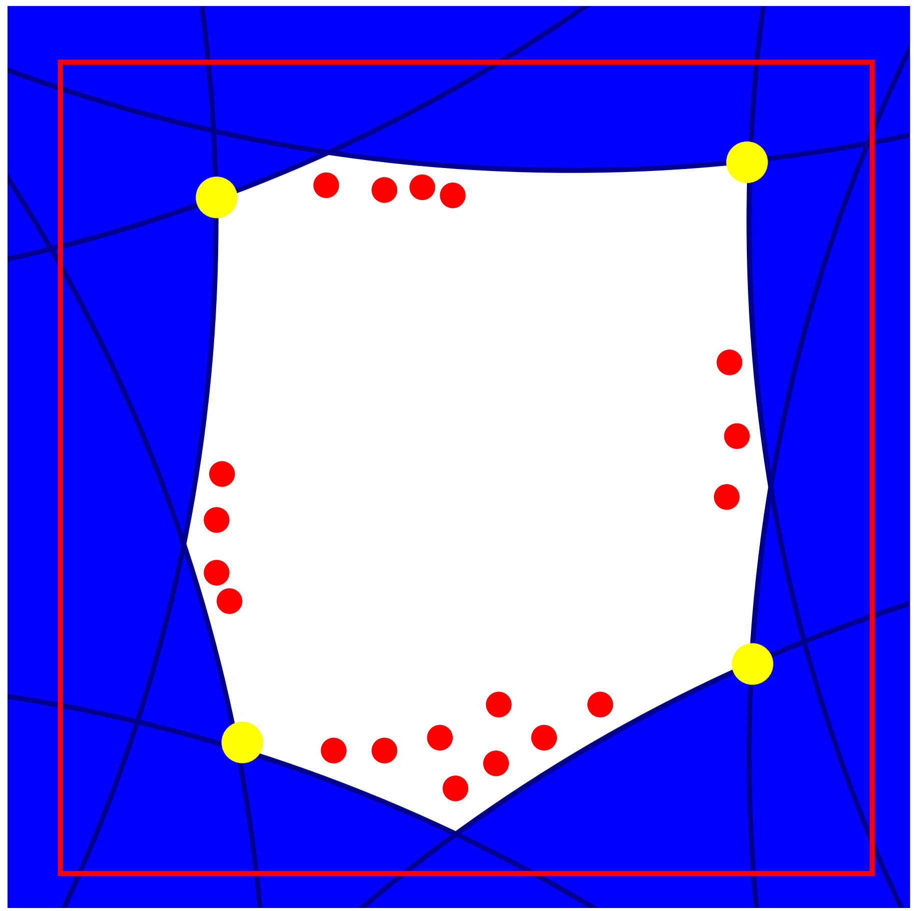


Enlarged:



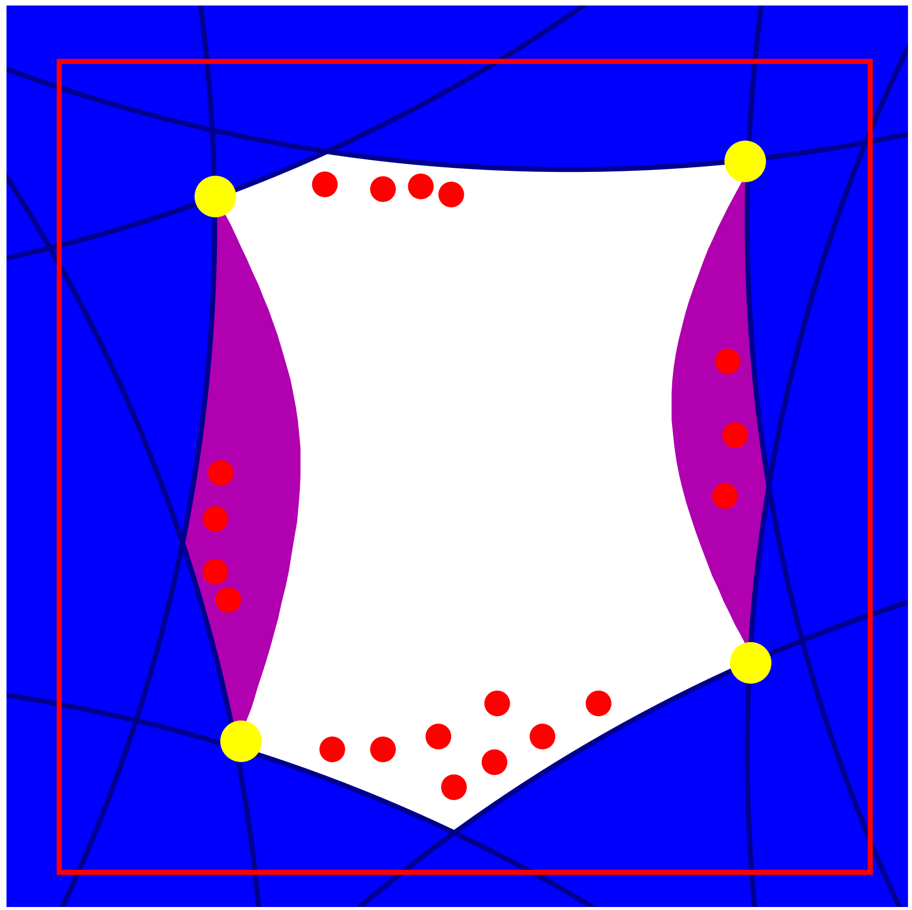
Sketch of the one-hole case

Step 1: Guess the four “corner points” of the optimal solution (each of them is defined by two disks).



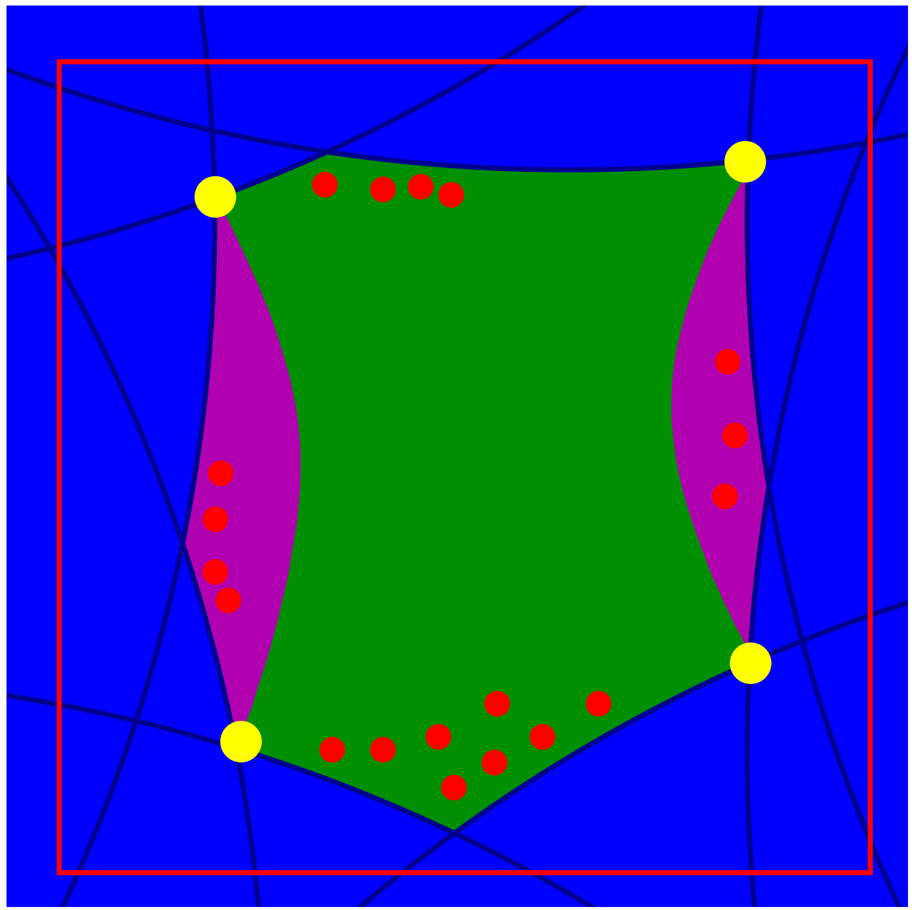
Sketch of the one-hole case

Step 2: Two regions that can only be covered with disks whose centers are to the left or right of the square.



Sketch of the one-hole case

Step 3: Remaining area can only be covered with disks whose centers are above or below the square.



Summary: MWDS in unit disk graphs

- Partition the plane into unit squares and solve the problem for each square separately. (We lose a constant factor compared to OPT.)
- For each square, reduce the weighted dominating set problem to a weighted disk cover problem.
- Distinguish one-hole case and many-holes case.
- In each case, we have a 2-approximation or optimal algorithm for covering points in the square with weighted unit disks.
- This implies the constant-factor approximation algorithm for MWDS in unit disk graphs.

Weighted Connected Dominating Sets

Theorem. There is a constant-factor approximation algorithm for MWCDS in unit disk graphs.

Algorithm Sketch:

- First, compute an $O(1)$-approximate MWDS D.
- Build auxiliary graph H with a vertex for each component of D, and weighted edges corresponding to paths with at most two internal vertices.
- Compute a minimum spanning tree of H and add the disks corresponding to its edges to D.

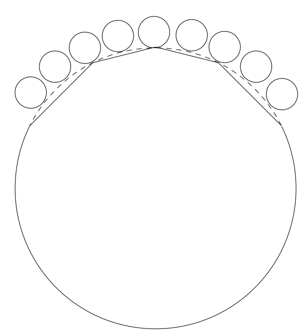
We can show: The total weight of the disks added to D is at most $17 \cdot \mathrm{OPT}$, where OPT is the weight of a minimum weight connected dominating set. The overall approximation ratio is then $72 + 17 = 89$.

Further results on MDS and MWDS

Theorem. [E, van Leeuwen 2007/2008] For disk graphs with bounded ply, there is a $(3+\varepsilon)$-approximation algorithm for MWDS. For intersection graphs of r-regular polygons, there is an $O(r^2)$-approximation algorithm for MDS.

Theorem. [E, van Leeuwen 2007/2008] For rectangle intersection graphs, MDS is APX-hard.

Theorem. [E, van Leeuwen 2007/2008] For intersection graphs of convex fat objects, MDS cannot be approximated with ratio $o(\log n)$ unless $P = NP$.



Open Problems

Disk graphs

- Improve running-time and/or approximation ratio for MWDS in unit disk graphs.
- Is there a PTAS for MDS in disk graphs with bounded ply?
- What is the best possible approximation ratio for minimum dominating set in general disk graphs:
  - Is there an $O(1)$-approximation algorithm or even a PTAS?
  - Is the problem APX-hard?
- What is the complexity of the maximum clique problem in disk graphs?
  (polynomial for unit disk graphs [Clark et al., 1990], NP-hard for ellipses [Ambühl, Wagner 2002])

Rectangle intersection graphs

- What is the best possible approximation ratio for maximum independent set?
  - Known: For every $c > 0$, there is an approximation algorithm with ratio $1 + \frac{1}{c} \log n$. [Berman et al., 2001]
  - Known: If all rectangles have the same height, there is a PTAS. [Agarwal et al., 1998]
- Can we achieve approximation ratio $o(\log n)$ for MDS and MWDS?
- Can rectangle intersection graphs be **colored** with $O(\omega)$ colors, where ω is the clique number?
  (best known upper bound: $O(\omega^2)$ colors [Asplund and Grünbaum, 1960])

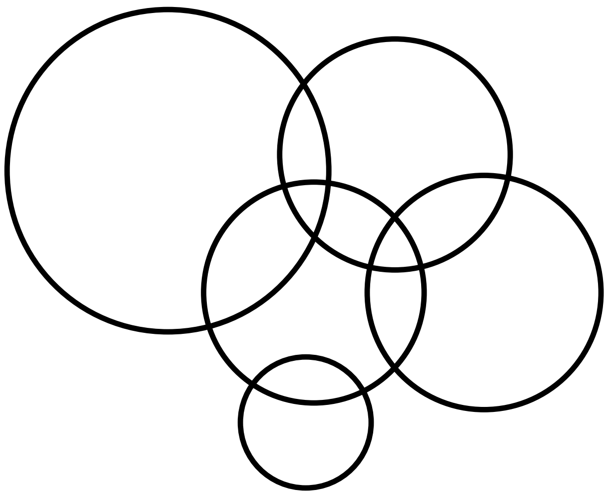
Thank you!

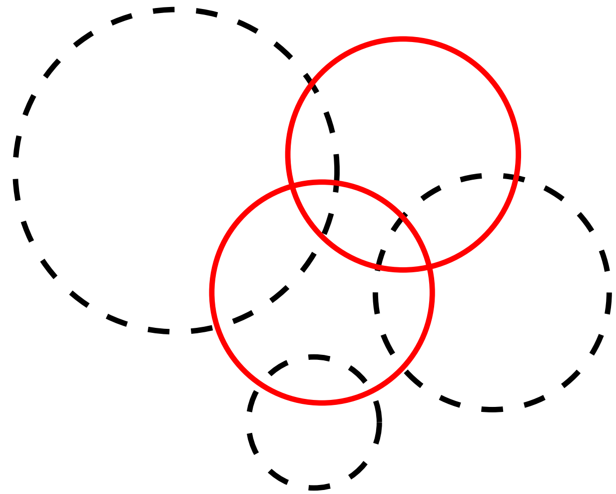
Appendix

Minimum Vertex Cover

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 $|C|$





Approximating MinVertexCover

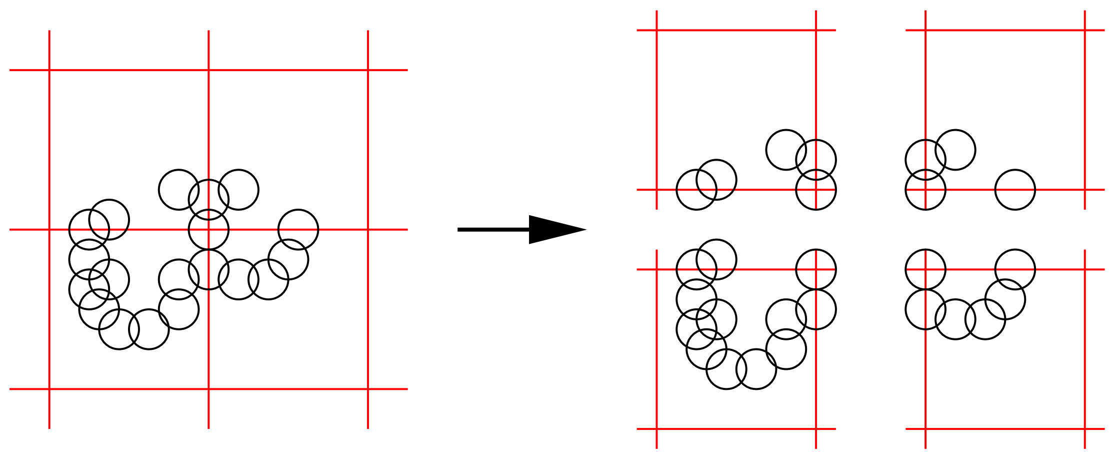
An algorithm for MinVertexCover is a **ρ-approximation algorithm** if it

- runs in **polynomial time** and
- always outputs a vertex cover of **size at most $\rho \cdot \mathrm{OPT}$**, 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 $\varepsilon > 0$.

PTAS idea for 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: MinVertexCover in unit disk graphs

1. For $0 \leq r, s < k$, partition the plane into squares via
   - horizontal lines equal to r modulo k and
   - vertical lines equal to s modulo 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 $\mathrm{OPT}(S)$ be the optimum vertex cover of the disks intersecting S.

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

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

For some choice of (r,s):

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

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

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

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

MinVC 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: PTAS adapts to **squares, regular polygons etc.**, also in **higher dimensions.** Result holds for the **weighted version** as well.