Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für
INFORMATIK I
Effiziente Algorithmen und
wissensbasierte Systeme

Institut für Informatik

Advanced Algorithms

Winter term 2019/20

Lecture 9. Exact Algorithms for Geometric Intersection Graphs

Slides by Paweł Rzążewski, Warsaw University of Technology

Steven Chaplick & Alexander Wolff *Chair for Computer Science I*

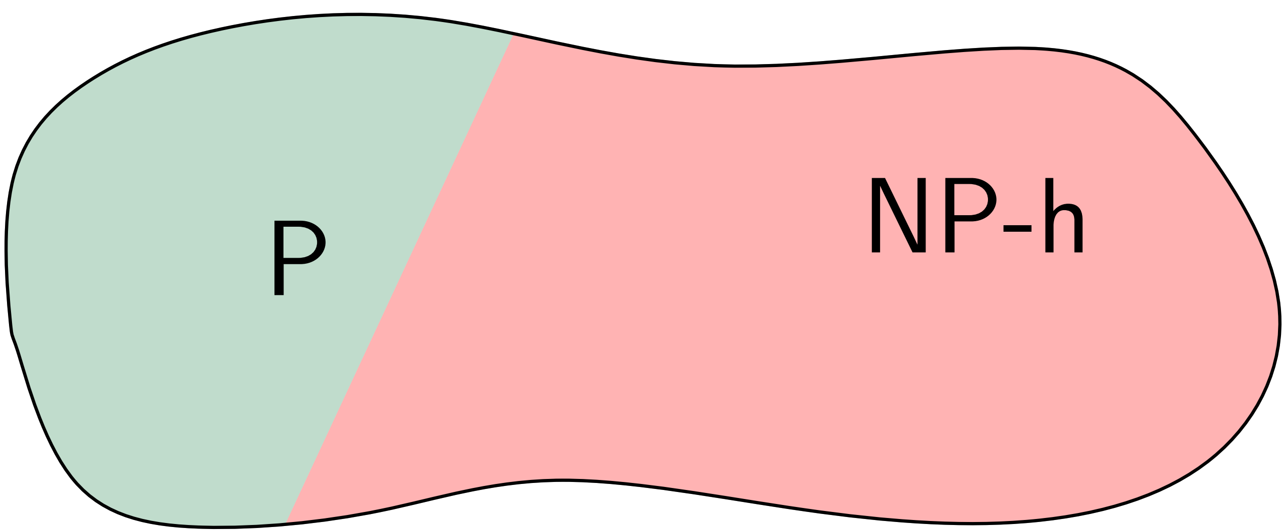
Fine-Grained Complexity and the Exponential-Time Hypothesis

Classical Approach to Complexity Theory

Assuming $P \neq NP$, we partition problems into two sets:

- P (solvable in polynomial time)
  proven by presenting an algorithm

- NP-hard (no polynomial algorithm)
  proven by polynomial reductions



Classical Approach to Complexity Theory

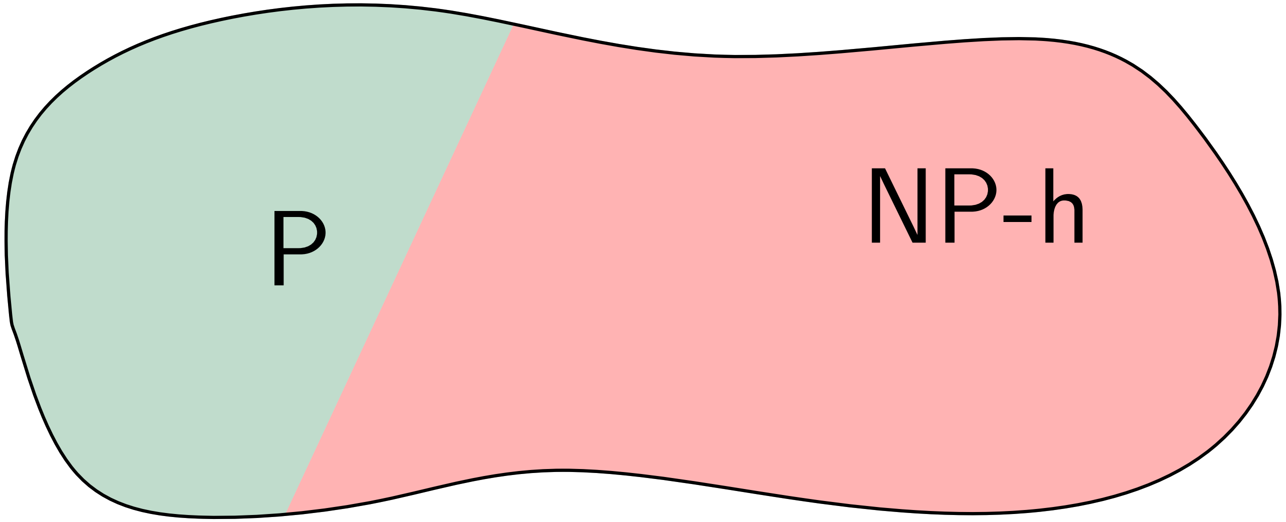
Assuming P $\neq$ NP, we partition problems into two sets:

- P (solvable in polynomial time)
  proven by presenting an algorithm

    worth attention,
    how fast can be solve them?

- NP-hard (no polynomial algorithm)
  proven by polynomial reductions

    hopeless, unsolvable



How Hard Are Hard Problems?

- Hard problems are quite common (even in practice).
- Many new algorithmic techniques

How Hard Are Hard Problems?

- Hard problems are quite common (even in practice).
- Many new algorithmic techniques
- NP-hardness $\rightarrow$ no polynomial algorithm

  but maybe $2^{\mathcal{O}(\sqrt{n})}$?
  or even $2^{\mathcal{O}(\log^2 n)}$?

polynomial time:
$n^c = 2^{c \log n} = 2^{\mathcal{O}(\log n)}$

How Hard Are Hard Problems?

- Hard problems are quite common (even in practice).
- Many new algorithmic techniques
- NP-hardness $\rightarrow$ no polynomial algorithm

  but maybe $2^{\mathcal{O}(\sqrt{n})}$?
  or even $2^{\mathcal{O}(\log^2 n)}$?

polynomial time:
$n^c = 2^{c \log n} = 2^{\mathcal{O}(\log n)}$

Exponential Time Hypothesis (ETH) [Impagliazzo, Paturi, 1999]
There is no algorithm solving $3\text{-}\textsc{Sat}$ with n variables and $\mathcal{O}(n)$ clauses in time $2^{o(n)}$.

How Hard Are Hard Problems?

- Hard problems are quite common (even in practice).
- Many new algorithmic techniques
- NP-hardness $\rightarrow$ no polynomial algorithm

  but maybe $2^{\mathcal{O}(\sqrt{n})}$?
  or even $2^{\mathcal{O}(\log^2 n)}$?

polynomial time:
$n^c = 2^{c \log n} = 2^{\mathcal{O}(\log n)}$

Exponential Time Hypothesis (ETH) [Impagliazzo, Paturi, 1999]
There is no algorithm solving $3\text{-}\textsc{Sat}$ with n variables and $\mathcal{O}(n)$ clauses in time $2^{o(n)}$.

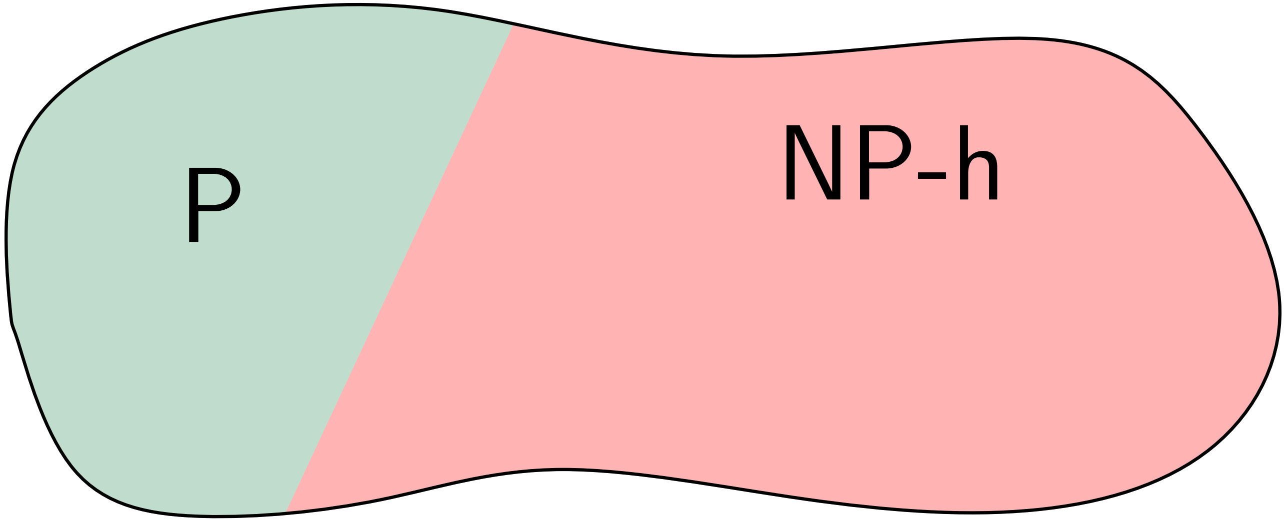
subexponential time: $2^{o(n)}$,
e.g. $2^{\mathcal{O}(n^{0.99})}$ or $2^{\mathcal{O}(n/\log n)}$

A Closer Look

Being a stronger assumption than P $\neq$ NP, ETH allows for a finer analysis:

- P (solvable in polynomial time)

- NP-hard
  (no polynomial
  algorithm)



A Closer Look

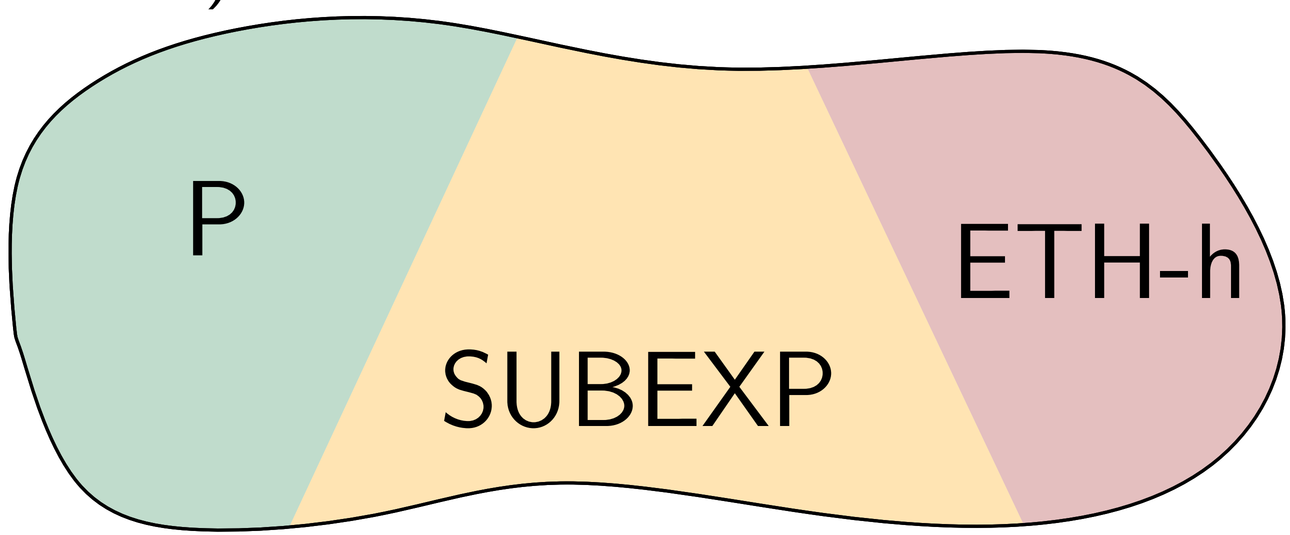
Being a stronger assumption than $P \neq NP$, ETH allows for a finer analysis:

- P (solvable in polynomial time)

SUBEXP (solvable in subexponential time)

- NP-hard (no polynomial algorithm)

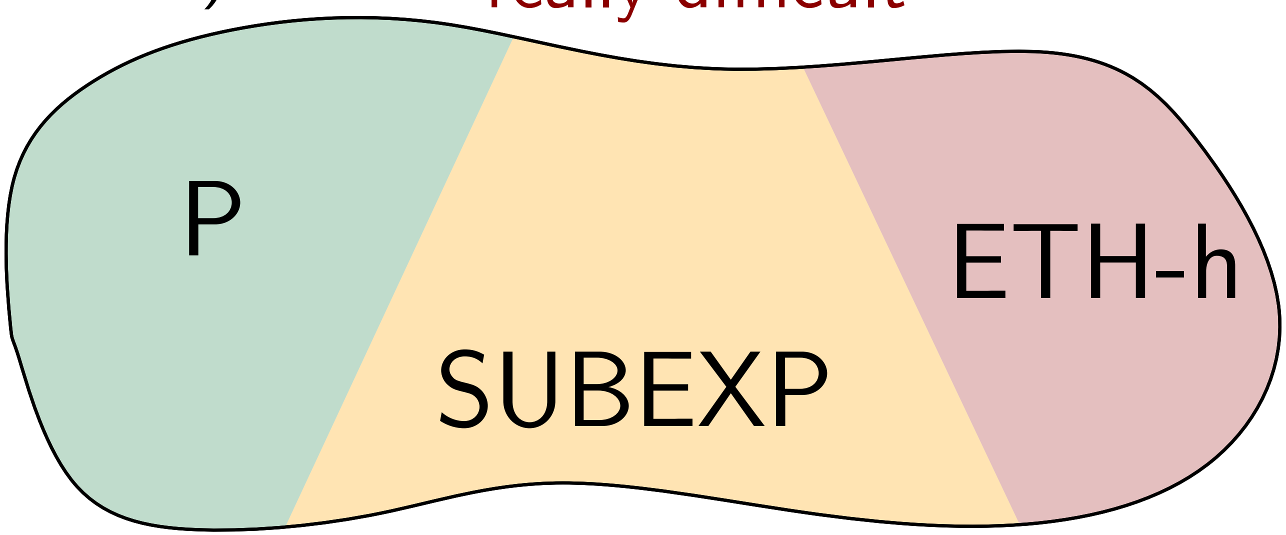
ETH-hard (no subexponential algorithm)



A Closer Look

Being a stronger assumption than P $\neq$ NP, ETH allows for a finer analysis:

- P (solvable in polynomial time)
  easy
- NP-hard (no polynomial algorithm)
  - SUBEXP (solvable in subexponential time)
    not so super-hard
  - ETH-hard (no subexponential algorithm)
    really difficult



Lower Bounds

- Hardness is proven via reductions.
- Start from 3-SAT with n variables and $m = \mathcal{O}(n)$ clauses.
- Construct an instance $\mathcal{I}$ with $N = \mathcal{O}(n^{\alpha})$ vertices.

Lower Bounds

- Hardness is proven via reductions.
- Start from 3-SAT with n variables and $m = \mathcal{O}(n)$ clauses.
- Construct an instance $\mathcal{I}$ with $N = \mathcal{O}(n^{\alpha})$ vertices.

$$\text{Algorithm solving } \mathcal{I} \text{ in time } 2^{o(N^{1/\alpha})}$$

$$\downarrow$$

$$\text{Algorithm solving 3-SAT in time } 2^{o(n)}$$

Lower Bounds

- Hardness is proven via reductions.
- Start from 3-SAT with n variables and $m = \mathcal{O}(n)$ clauses.
- Construct an instance $\mathcal{I}$ with $N = \mathcal{O}(n^{\alpha})$ vertices.

$$\text{Algorithm solving } \mathcal{I} \text{ in time } 2^{o(N^{1/\alpha})}$$

$$\downarrow$$

$$\text{Algorithm solving 3-SAT in time } 2^{o(n)}$$

$$\begin{array}{lllll} \alpha = 1 & \text{(linear reduction)} & \rightarrow & \text{no } 2^{o(n)} \text{ algorithm} \\ \alpha = 2 & \text{(quadratic reduction)} & \rightarrow & \text{no } 2^{o(\sqrt{n})} \text{ algorithm} \end{array}$$

What Can We Hope for?

- Bad news: Assuming the ETH, there are no subexponential algorithms for canonical graph problems.

  3-Coloring, Independent Set, Clique, Dominating Set, Vertex Cover, Hamiltonian Cycle, Max Cut etc.

What Can We Hope for?

- Bad news: Assuming the ETH, there are no subexponential algorithms for canonical graph problems.

  3-Coloring, Independent Set, Clique, Dominating Set, Vertex Cover, Hamiltonian Cycle, Max Cut etc.

Boring!

What Can We Hope for?

- Bad news: Assuming the ETH, there are no subexponential algorithms for canonical graph problems.

  3-Coloring, Independent Set, Clique, Dominating Set, Vertex Cover, Hamiltonian Cycle, Max Cut etc.

  Boring!

- What about restricted classes of graphs?
  Planar graphs?

What Can We Hope for?

- Bad news: Assuming the ETH, there are no subexponential algorithms for canonical graph problems.

  3-Coloring, Independent Set, Clique, Dominating Set, Vertex Cover, Hamiltonian Cycle, Max Cut etc.

  Boring!

- What about restricted classes of graphs?
  Planar graphs?

- Square-root phenomenon: For planar graphs, most canonical problems can be solved in time $2^{\mathcal{O}(\sqrt{n})}$.

  Assuming the ETH, this cannot be improved to $2^{o(\sqrt{n})}$.

What Can We Hope for?

- Bad news: Assuming the ETH, there are no subexponential algorithms for canonical graph problems.

  3-Coloring, Independent Set, Clique, Dominating Set, Vertex Cover, Hamiltonian Cycle, Max Cut etc.

  Boring!

- What about restricted classes of graphs?
  Planar graphs?

- Square-root phenomenon: For planar graphs, most canonical problems can be solved in time $2^{\mathcal{O}(\sqrt{n})}$.

  Assuming the ETH, this cannot be improved to $2^{o(\sqrt{n})}$.

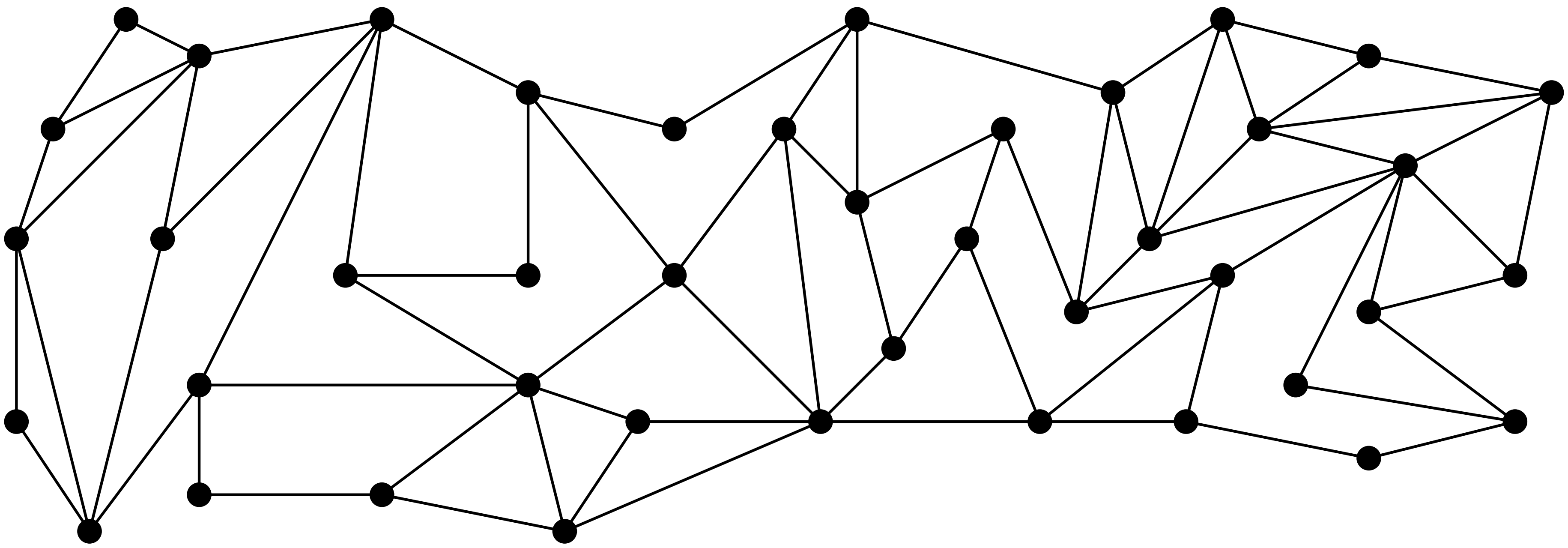
  Still boring!

Subexponential Algorithms for Planar Graphs

Planar separator theorem [Lipton & Tarjan 1979]

Every planar graph has a balanced separator of size $\mathcal{O}(\sqrt{n})$.

- also specialized versions, e.g. the separator is a cycle



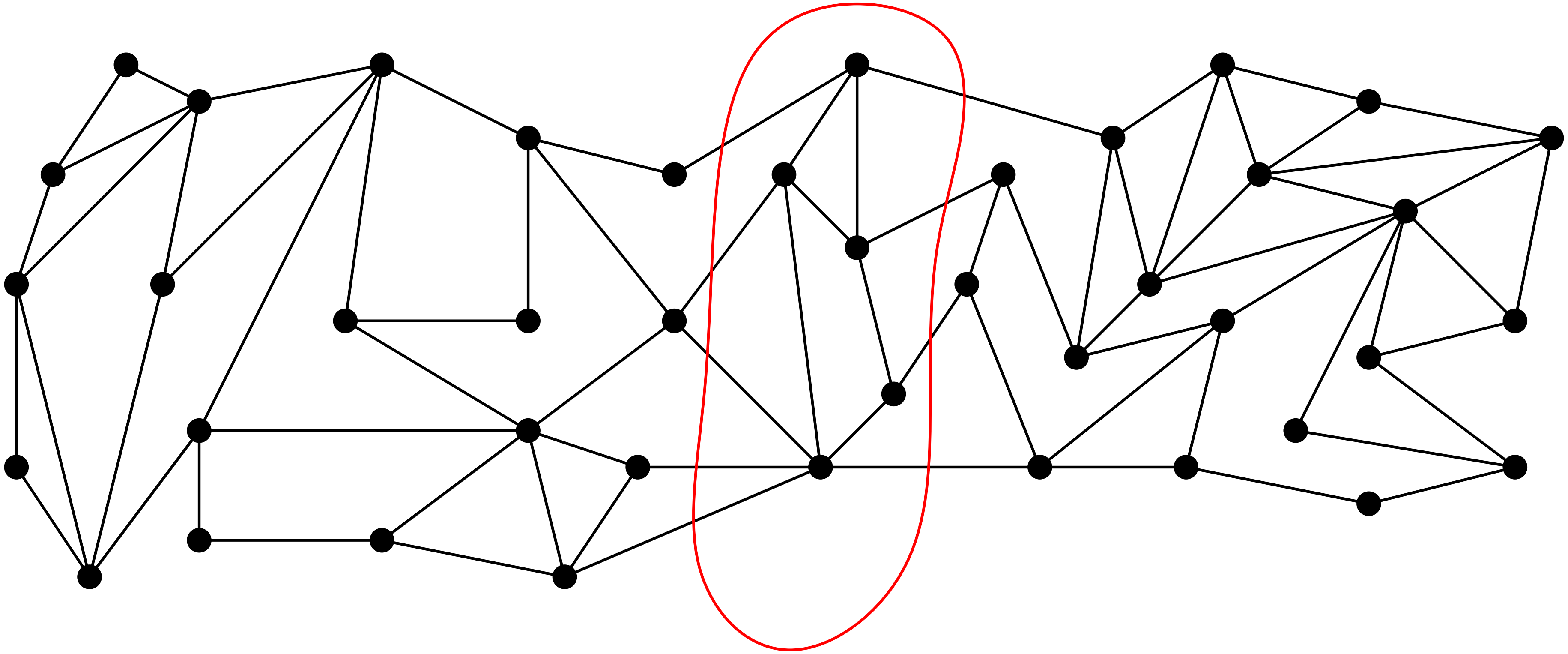
Independent Set

Subexponential Algorithms for Planar Graphs

Planar separator theorem [Lipton & Tarjan 1979]

Every planar graph has a balanced separator of size $\mathcal{O}(\sqrt{n})$.

- also specialized versions, e.g. the separator is a cycle



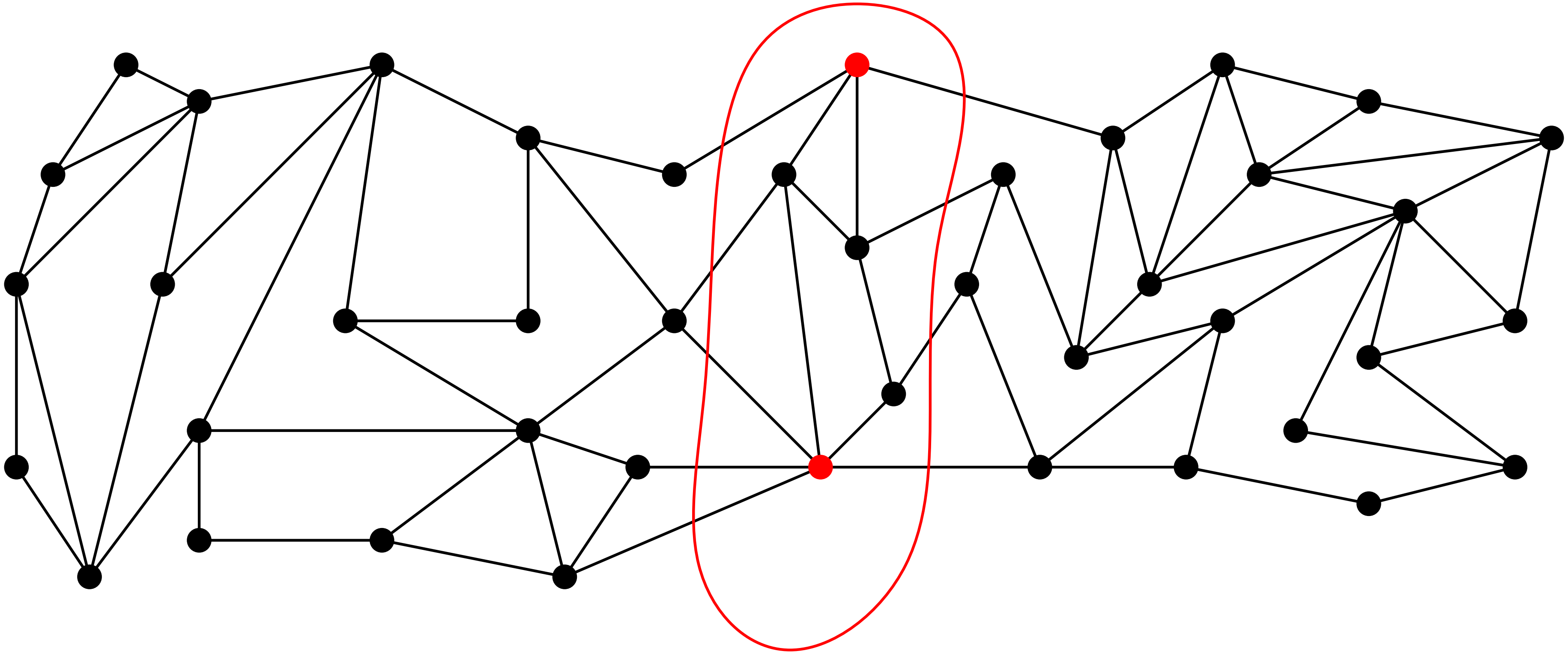
Independent Set

Subexponential Algorithms for Planar Graphs

Planar separator theorem [Lipton & Tarjan 1979]

Every planar graph has a balanced separator of size $\mathcal{O}(\sqrt{n})$.

- also specialized versions, e.g. the separator is a cycle



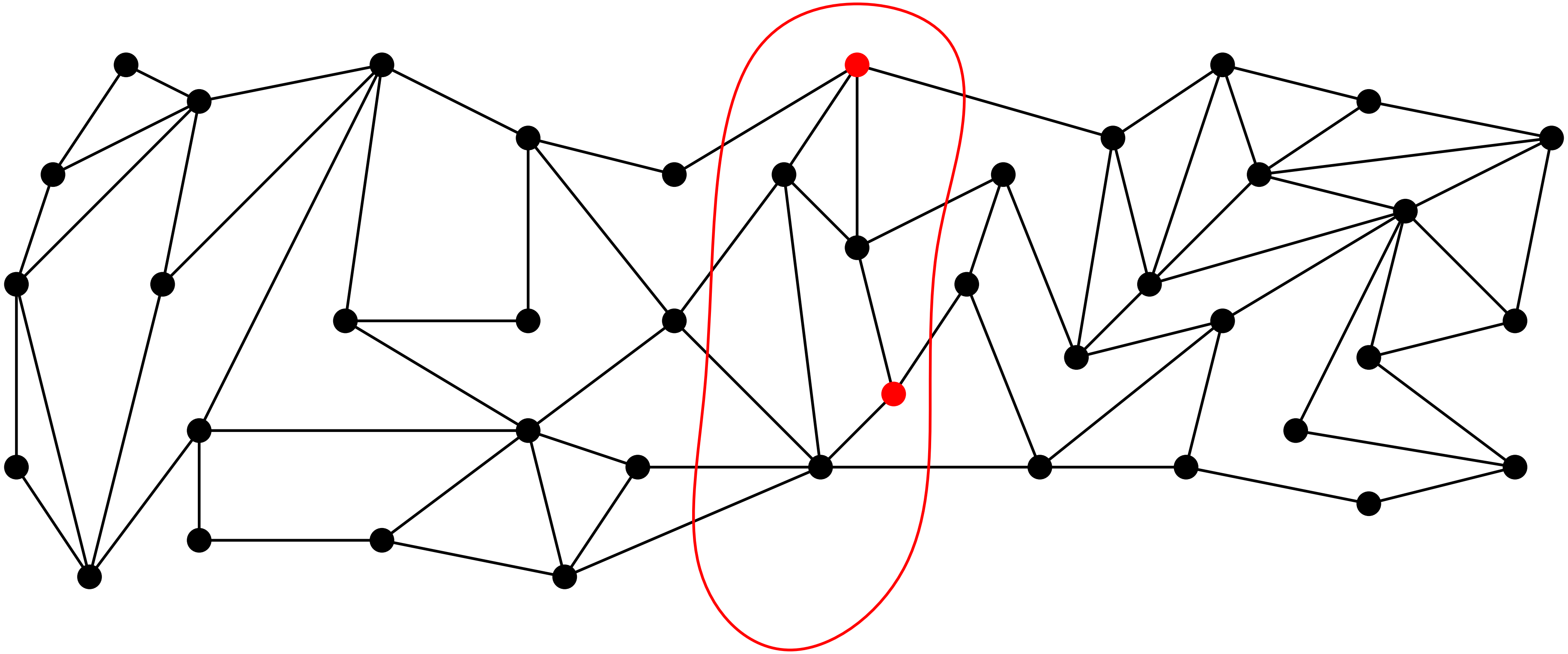
Independent Set

Subexponential Algorithms for Planar Graphs

Planar separator theorem [Lipton & Tarjan 1979]

Every planar graph has a balanced separator of size $\mathcal{O}(\sqrt{n})$.

- also specialized versions, e.g. the separator is a cycle



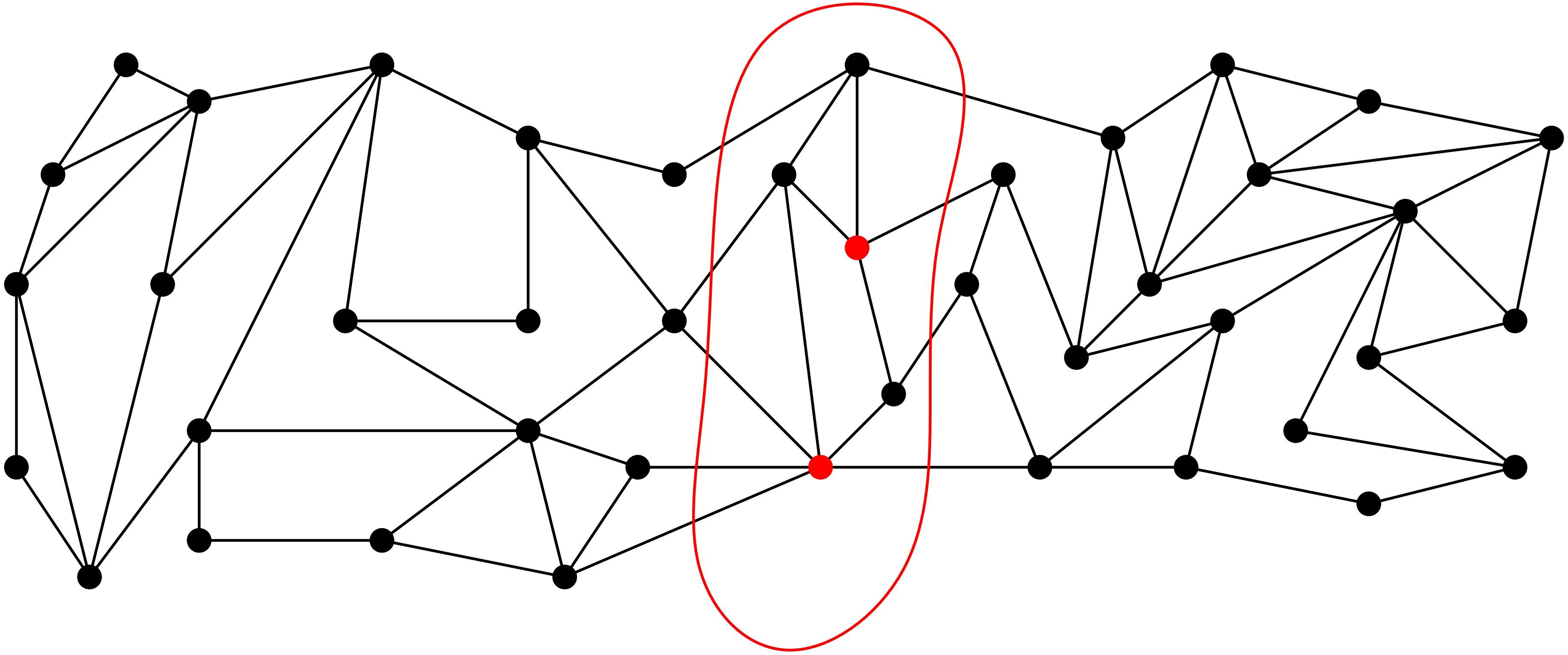
Independent Set

Subexponential Algorithms for Planar Graphs

Planar separator theorem [Lipton & Tarjan 1979]

Every planar graph has a balanced separator of size $\mathcal{O}(\sqrt{n})$.

- also specialized versions, e.g. the separator is a cycle



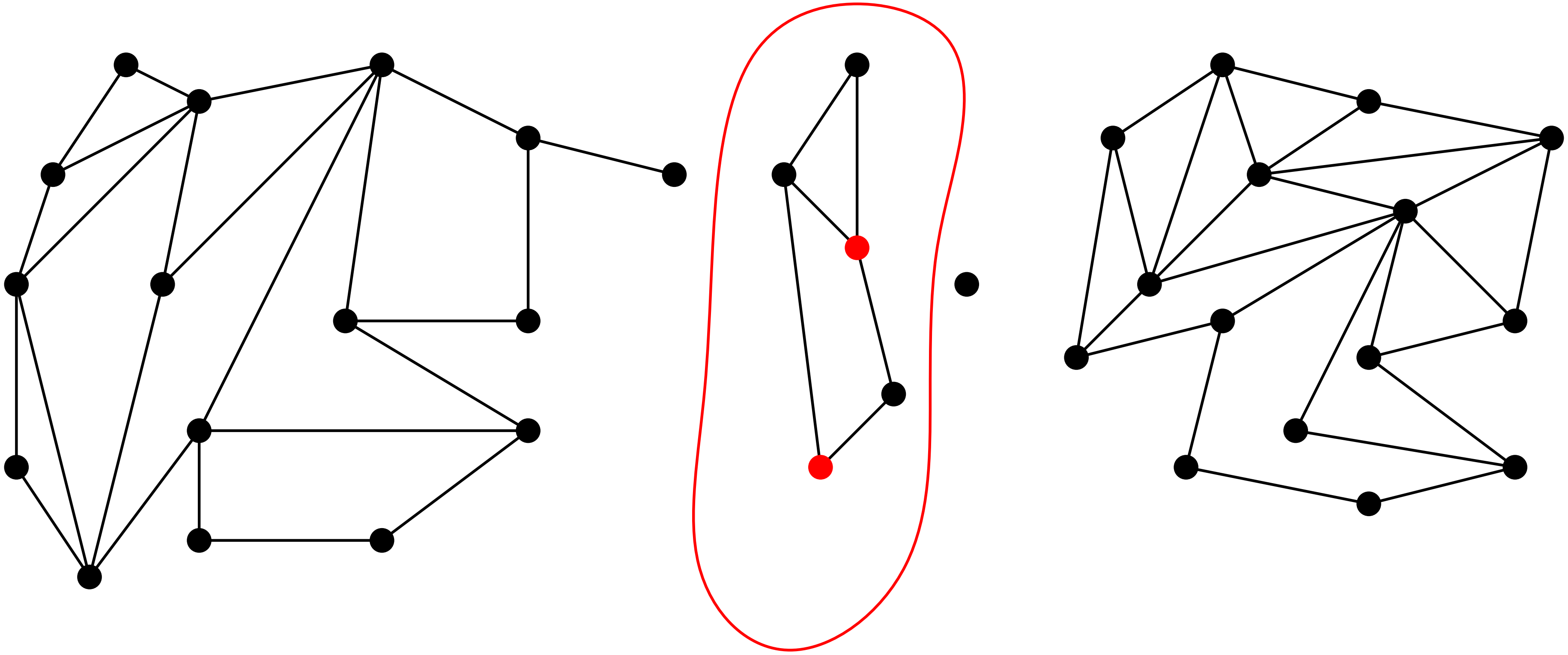
Independent Set

Subexponential Algorithms for Planar Graphs

Planar separator theorem [Lipton & Tarjan 1979]

Every planar graph has a balanced separator of size $\mathcal{O}(\sqrt{n})$.

- also specialized versions, e.g. the separator is a cycle



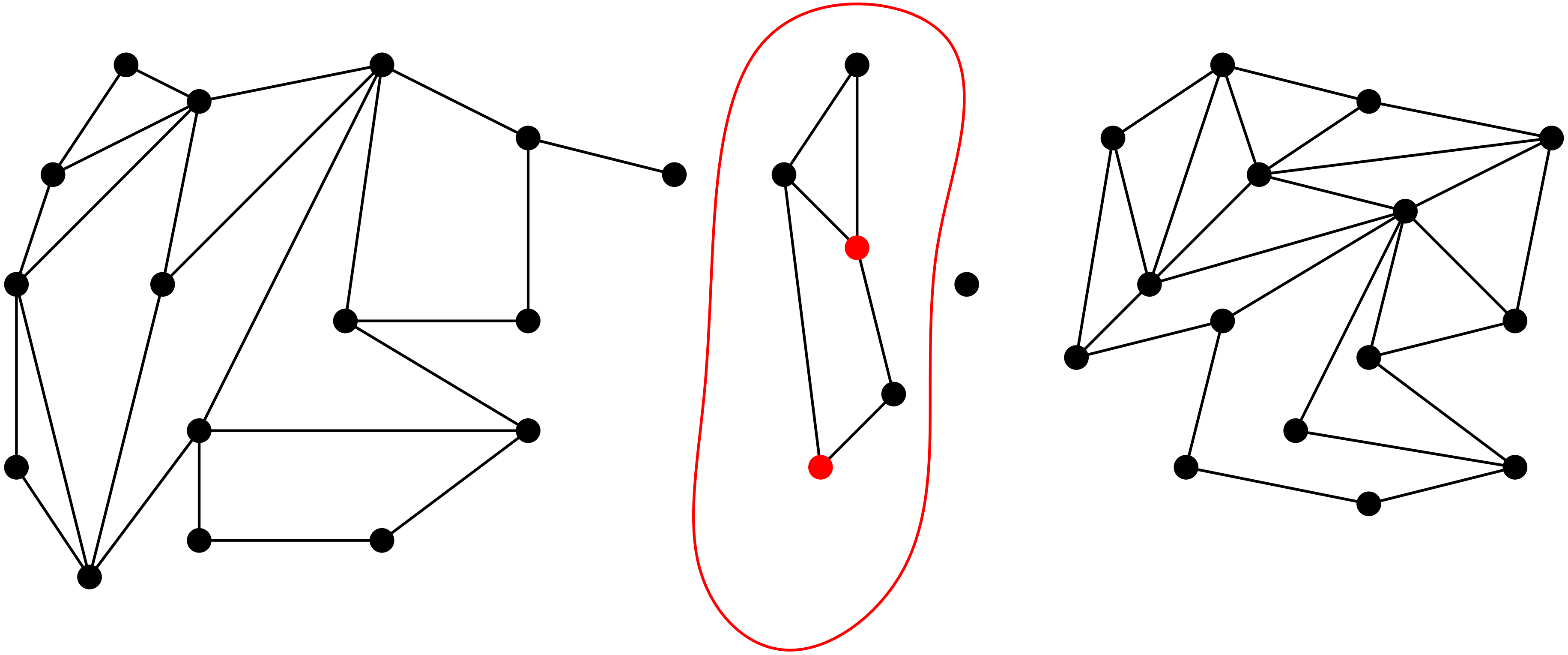
Independent Set

Subexponential Algorithms for Planar Graphs

Planar separator theorem [Lipton & Tarjan 1979]

Every planar graph has a balanced separator of size $\mathcal{O}(\sqrt{n})$.

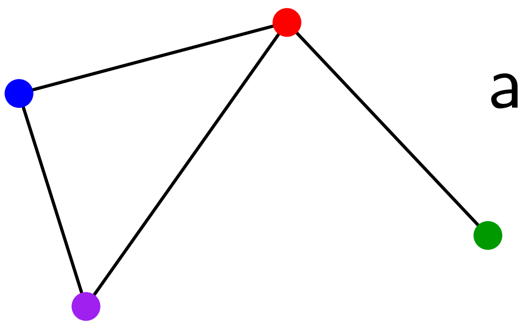
- also specialized versions, e.g. the separator is a cycle



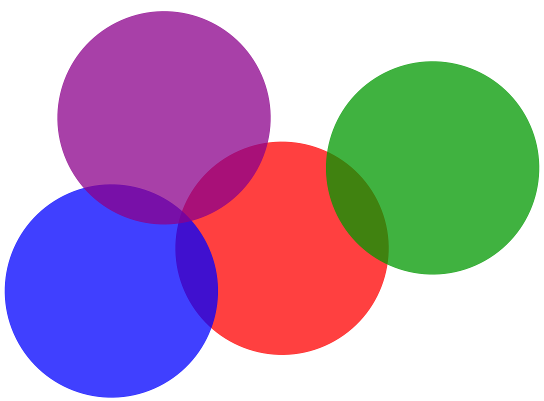
Independent Set

- Divide & conquer yields a $2^{\mathcal{O}(\sqrt{n})}$-time algorithm.

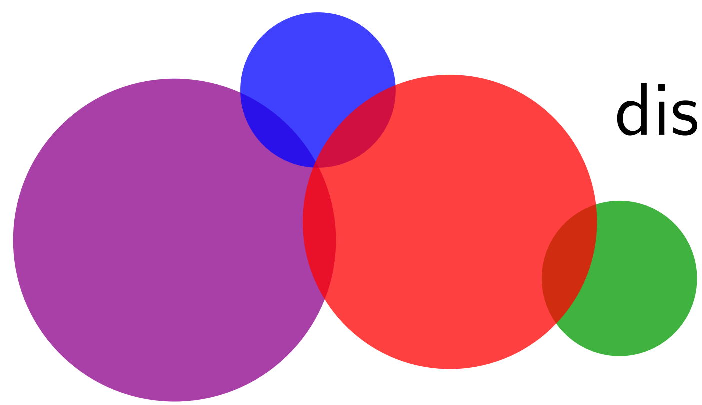
Geometric Intersection Graphs



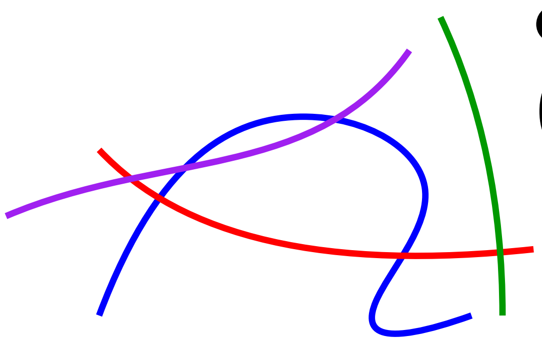
as an intersection graph of:



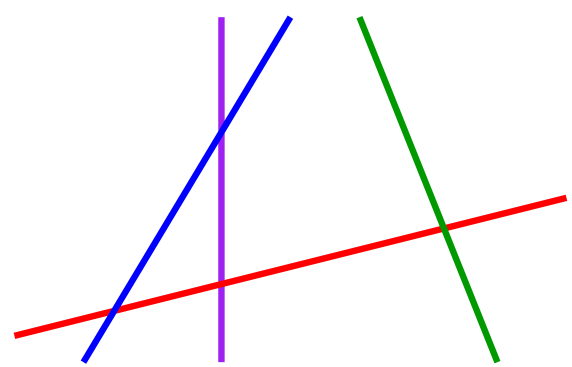
unit disks (UDG)



disks (DG)

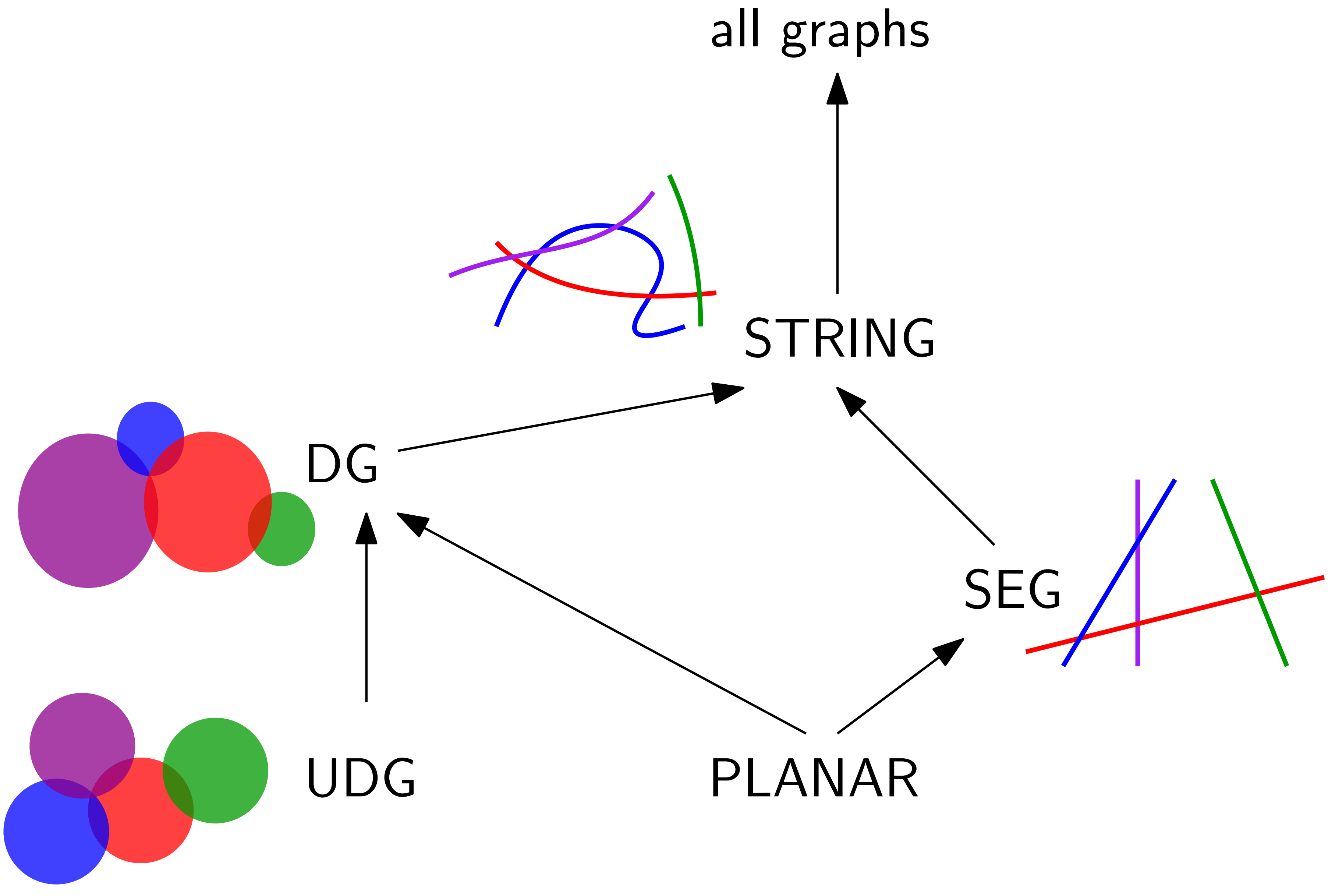


curves (STRING)



segments (SEG)

Relations Between Classes



Separator-Based Algorithms for Disk Intersection Graphs

k-Coloring Disk Graphs

Disk separator theorem [Miller et al. 1997]

A disk intersection graph of with ply $\leq k$ has a balanced separator of size $\mathcal{O}(\sqrt{nk})$.

k-Coloring Disk Graphs

Disk separator theorem [Miller et al. 1997]

A disk intersection graph of with ply $\leq k$ has a balanced separator of size $\mathcal{O}(\sqrt{nk})$.

ply $=$ max number of disks covering a single point

k-Coloring Disk Graphs

Disk separator theorem [Miller et al. 1997]

A disk intersection graph of with ply $\leq k$ has a balanced separator of size $\mathcal{O}(\sqrt{nk})$.

ply $=$ max number of disks covering a single point

k-Coloring of disk graphs

k-Coloring Disk Graphs

Disk separator theorem [Miller et al. 1997]

A disk intersection graph of with ply $\leq k$ has a balanced separator of size $\mathcal{O}(\sqrt{nk})$.

ply = max number of disks covering a single point

k-Coloring of disk graphs

2. ply $\leq k \Rightarrow$ balanced separator S of size $\mathcal{O}(\sqrt{nk})$
3. guess the coloring of S (one of $k^{|S|} = k^{\mathcal{O}(\sqrt{nk})}$ possibilities)
4. recurse using divide & conquer

k-Coloring Disk Graphs

Disk separator theorem [Miller et al. 1997]

A disk intersection graph of with ply $\leq k$ has a balanced separator of size $\mathcal{O}(\sqrt{nk})$.

ply = max number of disks covering a single point

k-Coloring of disk graphs

1. ply $> k \Rightarrow$ clique of size $> k \rightarrow$ return NO
2. ply $\leq k \Rightarrow$ balanced separator S of size $\mathcal{O}(\sqrt{nk})$
3. guess the coloring of S (one of $k^{|S|} = k^{\mathcal{O}(\sqrt{nk})}$ possibilities)
4. recurse using divide & conquer

k-Coloring Disk Graphs

Disk separator theorem [Miller et al. 1997]

A disk intersection graph of with ply $\leq k$ has a balanced separator of size $\mathcal{O}(\sqrt{nk})$.

ply $=$ max number of disks covering a single point

k-Coloring of disk graphs

1. ply $> k \Rightarrow$ clique of size $> k \rightarrow$ return NO
2. ply $\leq k \Rightarrow$ balanced separator S of size $\mathcal{O}(\sqrt{nk})$
3. guess the coloring of S (one of $k^{|S|} = k^{\mathcal{O}(\sqrt{nk})}$ possibilities)
4. recurse using divide & conquer

Theorem: For any fixed k, k-Coloring can be solved in time $2^{\mathcal{O}(\sqrt{n})}$ for disk graphs.

k-Coloring Disk Graphs

Disk separator theorem [Miller et al. 1997]

A disk intersection graph of with ply $\leq k$ has a balanced separator of size $\mathcal{O}(\sqrt{nk})$.

ply = max number of disks covering a single point

k-Coloring of disk graphs

1. ply $> k \Rightarrow$ clique of size $> k \rightarrow$ return NO
2. ply $\leq k \Rightarrow$ balanced separator S of size $\mathcal{O}(\sqrt{nk})$
3. guess the coloring of S (one of $k^{|S|} = k^{\mathcal{O}(\sqrt{nk})}$ possibilities)
4. recurse using divide & conquer

Theorem: For any fixed k, k-Coloring can be solved in time $2^{\mathcal{O}(\sqrt{n})}$ for disk graphs.

Key observation:

Yes-instances of k-Coloring do not have large cliques.

Independent Set in Disk Graphs

- Existence of a large clique does not trivialize the instance...

Independent Set in Disk Graphs

- Existence of a large clique does not trivialize the instance...
- ...but not too much can happen in a clique.

Independent Set in Disk Graphs

- Existence of a large clique does not trivialize the instance...
- ...but not too much can happen in a clique.

Let Q be a clique in G, $|Q| = \tau$.

Independent Set in Disk Graphs

- Existence of a large clique does not trivialize the instance...
- ...but not too much can happen in a clique.

Let Q be a clique in G, $|Q| = \tau$.

- At most one vertex of Q belongs to the optimal solution.

Independent Set in Disk Graphs

- Existence of a large clique does not trivialize the instance...
- ...but not too much can happen in a clique.

Let Q be a clique in G, $|Q| = \tau$.

- At most one vertex of Q belongs to the optimal solution.
- We can branch into $\tau + 1$ instances, each of size $n - \tau$.

Independent Set in Disk Graphs

- Existence of a large clique does not trivialize the instance...
- ...but not too much can happen in a clique.

Let Q be a clique in G, $|Q| = \tau$.

- At most one vertex of Q belongs to the optimal solution.
- We can branch into $\tau + 1$ instances, each of size $n - \tau$.

$$F(n) \leq (\tau + 1) \cdot F(n - \tau)$$

INDEPENDENT SET in Disk Graphs

- Existence of a large clique does not trivialize the instance...
- ...but not too much can happen in a clique.

Let Q be a clique in G, $|Q| = \tau$.

- At most one vertex of Q belongs to the optimal solution.
- We can branch into $\tau + 1$ instances, each of size $n - \tau$.

$$F(n) \le (\tau + 1) \cdot F(n - \tau) \le (\tau + 1)^2 \cdot F(n - 2\tau)$$

Independent Set in Disk Graphs

- Existence of a large clique does not trivialize the instance...
- ...but not too much can happen in a clique.

Let Q be a clique in G, $|Q| = \tau$.

- At most one vertex of Q belongs to the optimal solution.
- We can branch into $\tau + 1$ instances, each of size $n - \tau$.

$$\begin{aligned} F(n) &\leq (\tau+1) \cdot F(n-\tau) \leq (\tau+1)^2 \cdot F(n-2\tau) \\ &\leq \ldots \leq (\tau+1)^{n/\tau} \cdot \mathcal{O}(1) = 2^{\mathcal{O}(n/\tau \cdot \log \tau)} = 2^{\widetilde{\mathcal{O}}(n/\tau)} \end{aligned}$$

Independent Set in Disk Graphs

- Existence of a large clique does not trivialize the instance. . .
- . . . but not too much can happen in a clique.

Let Q be a clique in G, $|Q| = \tau$.

- At most one vertex of Q belongs to the optimal solution.
- We can branch into $\tau + 1$ instances, each of size $n - \tau$.

$$
\begin{aligned}
F(n) &\leq (\tau+1) \cdot F(n-\tau) \leq (\tau+1)^2 \cdot F(n-2\tau) \\
&\leq \dots \leq (\tau+1)^{n/\tau} \cdot \mathcal{O}(1) = 2^{\mathcal{O}(n/\tau \cdot \log \tau)} = 2^{\widetilde{\mathcal{O}}(n/\tau)}
\end{aligned}
$$

$\widetilde{\mathcal{O}}(f(n)) = f(n) \cdot \mathrm{polylog}(n)$

Independent Set in Disk Graphs

- Existence of a large clique does not trivialize the instance...
- ...but not too much can happen in a clique.

Let Q be a clique in G, $|Q| = \tau$.

- At most one vertex of Q belongs to the optimal solution.
- We can branch into $\tau + 1$ instances, each of size $n - \tau$.

$$F(n) \leq (\tau+1) \cdot F(n-\tau) \leq (\tau+1)^2 \cdot F(n-2\tau)$$
$$\leq \ldots \leq (\tau+1)^{n/\tau} \cdot \mathcal{O}(1) = 2^{\mathcal{O}(n/\tau \cdot \log \tau)} = 2^{\widetilde{\mathcal{O}}(n/\tau)}$$

$\widetilde{\mathcal{O}}(f(n)) = f(n) \cdot \mathrm{polylog}(n)$

Algorithm.

Independent Set in Disk Graphs

- Existence of a large clique does not trivialize the instance...
- ...but not too much can happen in a clique.

Let Q be a clique in G, $|Q| = \tau$.

- At most one vertex of Q belongs to the optimal solution.
- We can branch into $\tau + 1$ instances, each of size $n - \tau$.

$$F(n) \leq (\tau+1)\cdot F(n-\tau) \leq (\tau+1)^2 \cdot F(n-2\tau)$$
$$\leq \ldots \leq (\tau+1)^{n/\tau} \cdot \mathcal{O}(1) = 2^{\mathcal{O}(n/\tau \cdot \log \tau)} = 2^{\widetilde{\mathcal{O}}(n/\tau)}$$

$\widetilde{\mathcal{O}}(f(n)) = f(n) \cdot \mathrm{polylog}(n)$

Algorithm.

1. ply $> \tau \Rightarrow$ clique of size $> \tau$, branch $(2^{\widetilde{\mathcal{O}}(n/\tau)})$
2. ply $\leq \tau \Rightarrow$ balanced separator S of size $\mathcal{O}(\sqrt{n\tau})$
3. guess the solution on S (one of $2^{|S|} = 2^{\mathcal{O}(\sqrt{n\tau})}$ possibilities)
4. recurse using divide & conquer $(2^{\mathcal{O}(\sqrt{n\tau})})$

Independent Set in Disk Graphs (cont'd)

- We have two basic steps:
  - branching with complexity $2^{\widetilde{\mathcal{O}}(n/\tau)}$,
  - divide & conquer with complexity $2^{\mathcal{O}(\sqrt{n\tau})}$.

Independent Set in Disk Graphs (cont'd)

- We have two basic steps:
  - branching with complexity $2^{\widetilde{\mathcal{O}}(n/\tau)}$,
  - divide & conquer with complexity $2^{\mathcal{O}(\sqrt{n\tau})}$.
- How to choose the threshold τ?

Independent Set in Disk Graphs (cont'd)

- We have two basic steps:
  - branching with complexity $2^{\widetilde{\mathcal{O}}(n/\tau)}$,
  - divide & conquer with complexity $2^{\mathcal{O}(\sqrt{n\tau})}$.
- How to choose the threshold τ?

$$n/\tau = \sqrt{n\tau}$$

Independent Set in Disk Graphs (cont'd)

- We have two basic steps:
  - branching with complexity $2^{\widetilde{\mathcal{O}}(n/\tau)}$,
  - divide & conquer with complexity $2^{\mathcal{O}(\sqrt{n\tau})}$.
- How to choose the threshold τ?

$$n/\tau = \sqrt{n\tau}$$
$$\tau = n^{1/3}$$

Independent Set in Disk Graphs (cont'd)

- We have two basic steps:
  - branching with complexity $2^{\widetilde{\mathcal{O}}(n/\tau)}$,
  - divide & conquer with complexity $2^{\mathcal{O}(\sqrt{n\tau})}$.
- How to choose the threshold τ?

$$n/\tau = \sqrt{n\tau}$$
$$\tau = n^{1/3}$$

Theorem. Independent Set can be solved in time $2^{\mathcal{O}(n^{2/3})}$ for disk graphs.

Independent Set in Disk Graphs (cont'd)

- We have two basic steps:
  - branching with complexity $2^{\widetilde{\mathcal{O}}(n/\tau)}$,
  - divide & conquer with complexity $2^{\mathcal{O}(\sqrt{n\tau})}$.
- How to choose the threshold τ?

$$n/\tau = \sqrt{n\tau}$$
$$\tau = n^{1/3}$$

Theorem. Independent Set can be solved in time $2^{\mathcal{O}(n^{2/3})}$ for disk graphs.

We can do better; more on this later.

Independent Set in Disk Graphs (cont'd)

- We have two basic steps:
  - branching with complexity $2^{\widetilde{\mathcal{O}}(n/\tau)}$,
  - divide & conquer with complexity $2^{\mathcal{O}(\sqrt{n\tau})}$.
- How to choose the threshold τ?

$$n/\tau = \sqrt{n\tau}$$
$$\tau = n^{1/3}$$

Theorem. Independent Set can be solved in time $2^{\mathcal{O}(n^{2/3})}$ for disk graphs.

We can do better; more on this later.

Also, still quite boring!

Optimality for
Segment and String Graphs

Independent Set in String Graphs

String separator theorem [Matoušek 2014, Lee 2016]

String graphs have balanced separators of size $\mathcal{O}(\sqrt{m})$.

Independent Set in String Graphs

String separator theorem [Matoušek 2014, Lee 2016]

String graphs have balanced separators of size $\mathcal{O}(\sqrt{m})$.

Theorem. [Fox & Pach 2011]

Independ. Set in string graphs can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

Independent Set in String Graphs

String separator theorem [Matoušek 2014, Lee 2016]
String graphs have balanced separators of size $\mathcal{O}(\sqrt{m})$.

Theorem. [Fox & Pach 2011]
Independ. Set in string graphs can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

1. There is a vertex v of degree at least $\tau = n^{1/3} \Rightarrow$ branch!
   - Either discard v or put it into the solution.

     $F(n) \leq F(n-1) + F(n - n^{1/3})$

Independent Set in String Graphs

String separator theorem [Matoušek 2014, Lee 2016]
String graphs have balanced separators of size $\mathcal{O}(\sqrt{m})$.

Theorem. [Fox & Pach 2011]
Independ. Set in string graphs can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

1. There is a vertex v of degree at least $\tau = n^{1/3} \Rightarrow$ branch!
   - Either discard v or put it into the solution.

$$F(n) \leq F(n-1) + F(n - n^{1/3}) \leq F(n-2) + 2 \cdot F(n - n^{1/3})$$

Independent Set in String Graphs

String separator theorem [Matoušek 2014, Lee 2016]
String graphs have balanced separators of size $\mathcal{O}(\sqrt{m})$.

Theorem. [Fox & Pach 2011]
Independ. Set in string graphs can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

1. There is a vertex v of degree at least $\tau = n^{1/3} \Rightarrow$ branch!
   - Either discard v or put it into the solution.

$$F(n) \leq F(n-1) + F(n - n^{1/3}) \leq F(n-2) + 2 \cdot F(n - n^{1/3})$$
$$\leq ... \leq (n^{1/3} + 1) \cdot F(n - n^{1/3}) \leq$$

Independent Set in String Graphs

String separator theorem [Matoušek 2014, Lee 2016]
String graphs have balanced separators of size $\mathcal{O}(\sqrt{m})$.

Theorem. [Fox & Pach 2011]
Independ. Set in string graphs can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

1. There is a vertex v of degree at least $\tau = n^{1/3} \Rightarrow$ branch!
   - Either discard v or put it into the solution.

$$F(n) \leq F(n-1) + F(n - n^{1/3}) \leq F(n-2) + 2 \cdot F(n - n^{1/3})$$
$$\leq \ldots \leq (n^{1/3}+1) \cdot F(n - n^{1/3}) \leq (n^{1/3}+1)^{n^{2/3}} =$$

Independent Set in String Graphs

String separator theorem [Matoušek 2014, Lee 2016]

String graphs have balanced separators of size $\mathcal{O}(\sqrt{m})$.

Theorem. [Fox & Pach 2011]

Independ. Set in string graphs can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

1. There is a vertex v of degree at least $\tau = n^{1/3} \Rightarrow$ branch!
   - Either discard v or put it into the solution.

$$F(n) \leq F(n-1) + F(n - n^{1/3}) \leq F(n-2) + 2 \cdot F(n - n^{1/3})$$
$$\leq \dots \leq (n^{1/3} + 1) \cdot F(n - n^{1/3}) \leq (n^{1/3} + 1)^{n^{2/3}} = 2^{\widetilde{\mathcal{O}}(n^{2/3})}$$

Independent Set in String Graphs

String separator theorem [Matoušek 2014, Lee 2016]
String graphs have balanced separators of size $\mathcal{O}(\sqrt{m})$.

Theorem. [Fox & Pach 2011]
Independ. Set in string graphs can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

1. There is a vertex v of degree at least $\tau = n^{1/3} \Rightarrow$ branch!
   - Either discard v or put it into the solution.

$$F(n) \leq F(n-1) + F(n-n^{1/3}) \leq F(n-2) + 2 \cdot F(n-n^{1/3})$$
$$\leq \ldots \leq (n^{1/3}+1) \cdot F(n-n^{1/3}) \leq (n^{1/3}+1)^{n^{2/3}} = 2^{\widetilde{\mathcal{O}}(n^{2/3})}$$

time complexity $2^{\widetilde{\mathcal{O}}(n^{2/3})}$

Independent Set in String Graphs

String separator theorem [Matoušek 2014, Lee 2016]
String graphs have balanced separators of size $\mathcal{O}(\sqrt{m})$.

Theorem. [Fox & Pach 2011]
Independ. Set in string graphs can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

1. There is a vertex v of degree at least $\tau = n^{1/3} \Rightarrow$ branch!
   - Either discard v or put it into the solution.

$$
\begin{aligned}
F(n) &\leq F(n-1) + F(n-n^{1/3}) \leq F(n-2) + 2 \cdot F(n-n^{1/3}) \\
&\leq \ldots \leq (n^{1/3}+1) \cdot F(n-n^{1/3}) \leq (n^{1/3}+1)^{n^{2/3}} = 2^{\widetilde{\mathcal{O}}(n^{2/3})}
\end{aligned}
$$

time complexity $2^{\widetilde{\mathcal{O}}(n^{2/3})}$

2. $m \leq n^{4/3} \Rightarrow$

Independent Set in String Graphs

String separator theorem [Matoušek 2014, Lee 2016]
String graphs have balanced separators of size $\mathcal{O}(\sqrt{m})$.

Theorem. [Fox & Pach 2011]
Independ. Set in string graphs can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

1. There is a vertex v of degree at least $\tau = n^{1/3} \Rightarrow$ branch!
   - Either discard v or put it into the solution.

$$F(n) \leq F(n-1) + F(n-n^{1/3}) \leq F(n-2) + 2 \cdot F(n-n^{1/3})$$
$$\leq \ldots \leq (n^{1/3}+1) \cdot F(n-n^{1/3}) \leq (n^{1/3}+1)^{n^{2/3}} = 2^{\widetilde{\mathcal{O}}(n^{2/3})}$$

time complexity $2^{\widetilde{\mathcal{O}}(n^{2/3})}$

2. $m \leq n^{4/3} \Rightarrow$ balanced separator of size $\mathcal{O}(n^{2/3})$

INDEPENDENT SET in String Graphs

String separator theorem [Matoušek 2014, Lee 2016]
String graphs have balanced separators of size $\mathcal{O}(\sqrt{m})$.

Theorem. [Fox & Pach 2011]
INDEPEND. SET in string graphs can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

1. There is a vertex v of degree at least $\tau = n^{1/3} \Rightarrow$ branch!
   - Either discard v or put it into the solution.

$$F(n) \leq F(n-1) + F(n-n^{1/3}) \leq F(n-2) + 2 \cdot F(n-n^{1/3})$$
$$\leq \ldots \leq (n^{1/3}+1) \cdot F(n-n^{1/3}) \leq (n^{1/3}+1)^{n^{2/3}} = 2^{\widetilde{\mathcal{O}}(n^{2/3})}$$

time complexity $2^{\widetilde{\mathcal{O}}(n^{2/3})}$

2. $m \leq n^{4/3} \Rightarrow$ balanced separator of size $\mathcal{O}(n^{2/3})$
   - Guess the solution on S and recurse

Independent Set in String Graphs

String separator theorem [Matoušek 2014, Lee 2016]
String graphs have balanced separators of size $\mathcal{O}(\sqrt{m})$.

Theorem. [Fox & Pach 2011]
Independ. Set in string graphs can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

1. There is a vertex v of degree at least $\tau = n^{1/3} \Rightarrow$ branch!
   - Either discard v or put it into the solution.

$$F(n) \leq F(n-1) + F(n - n^{1/3}) \leq F(n-2) + 2 \cdot F(n - n^{1/3})$$
$$\leq \dots \leq (n^{1/3}+1) \cdot F(n - n^{1/3}) \leq (n^{1/3}+1)^{n^{2/3}} = 2^{\widetilde{\mathcal{O}}(n^{2/3})}$$

time complexity $2^{\widetilde{\mathcal{O}}(n^{2/3})}$

2. $m \leq n^{4/3} \Rightarrow$ balanced separator of size $\mathcal{O}(n^{2/3})$
   - Guess the solution on S and recurse $\Rightarrow 2^{\widetilde{\mathcal{O}}(n^{2/3})}$ time

3-Coloring

1. There is a vertex v of degree at least $\tau = n^{1/3}$ $\Rightarrow$???
   - guessing a color for v does not mean we can discard $N(v)$!

2. $m \leq n^{4/3} \Rightarrow$ balanced separator of size $\mathcal{O}(n^{2/3})$
   - Guess the solution on S and recurse $\Rightarrow 2^{\widetilde{\mathcal{O}}(n^{2/3})}$ time.

3-COLORING

1. There is a vertex v of degree at least $\tau = n^{1/3}$
   - Consider LIST 3-COLORING: lists are subsets of $\{1, 2, 3\}$.

2. $m \leq n^{4/3} \Rightarrow$ balanced separator of size $\mathcal{O}(n^{2/3})$
   - Guess the solution on S and recurse $\Rightarrow 2^{\widetilde{\mathcal{O}}(n^{2/3})}$ time.

3-Coloring

1. There is a vertex v of degree at least $\tau = n^{1/3}$
   - Consider List 3-Coloring: lists are subsets of $\{1, 2, 3\}$.
   - We can get rid of vertices with one-element lists.
   - Possible lists are $\{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$.

2. $m \leq n^{4/3} \Rightarrow$ balanced separator of size $\mathcal{O}(n^{2/3})$
   - Guess the solution on S and recurse $\Rightarrow 2^{\widetilde{\mathcal{O}}(n^{2/3})}$ time.

3-Coloring

1. There is a vertex v of degree at least $\tau = n^{1/3}$
   - Consider List 3-Coloring: lists are subsets of $\{1, 2, 3\}$.
   - We can get rid of vertices with one-element lists.
   - Possible lists are $\{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$.
   - At least $n^{1/3}/4$ neighbors of v have the same list L.
   - There is a color c shared by L and $L(v)$.
   - Branch: either v gets color c or not.

2. $m \leq n^{4/3} \Rightarrow$ balanced separator of size $\mathcal{O}(n^{2/3})$
   - Guess the solution on S and recurse $\Rightarrow 2^{\widetilde{\mathcal{O}}(n^{2/3})}$ time.

3-Coloring

1. There is a vertex v of degree at least $\tau = n^{1/3}$
   - Consider List 3-Coloring: lists are subsets of $\{1, 2, 3\}$.
   - We can get rid of vertices with one-element lists.
   - Possible lists are $\{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$.
   - At least $n^{1/3}/4$ neighbors of v have the same list L.
   - There is a color c shared by L and $L(v)$.
   - Branch: either v gets color c or not.
   - $N =$ total size of all lists $\Rightarrow N \leq$

2. $m \leq n^{4/3} \Rightarrow$ balanced separator of size $\mathcal{O}(n^{2/3})$
   - Guess the solution on S and recurse $\Rightarrow 2^{\widetilde{\mathcal{O}}(n^{2/3})}$ time.

3-Coloring

1. There is a vertex v of degree at least $\tau = n^{1/3}$
   - Consider List 3-Coloring: lists are subsets of $\{1,2,3\}$.
   - We can get rid of vertices with one-element lists.
   - Possible lists are $\{1,2\},\{1,3\},\{2,3\},\{1,2,3\}$.
   - At least $n^{1/3}/4$ neighbors of v have the same list L.
   - There is a color c shared by L and $L(v)$.
   - Branch: either v gets color c or not.
   - $N =$ total size of all lists $\Rightarrow N \leq 3n$.
2. $m \leq n^{4/3} \Rightarrow$ balanced separator of size $\mathcal{O}(n^{2/3})$
   - Guess the solution on S and recurse $\Rightarrow 2^{\widetilde{\mathcal{O}}(n^{2/3})}$ time.

3-Coloring

1. There is a vertex v of degree at least $\tau = n^{1/3}$
   - Consider List 3-Coloring: lists are subsets of $\{1,2,3\}$.
   - We can get rid of vertices with one-element lists.
   - Possible lists are $\{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}$.
   - At least $n^{1/3}/4$ neighbors of v have the same list L.
   - There is a color c shared by L and $L(v)$.
   - Branch: either v gets color c or not.
   - $N =$ total size of all lists $\Rightarrow N \leq 3n$.

   $F(N) \leq$

2. $m \leq n^{4/3} \Rightarrow$ balanced separator of size $\mathcal{O}(n^{2/3})$
   - Guess the solution on S and recurse $\Rightarrow 2^{\widetilde{\mathcal{O}}(n^{2/3})}$ time.

3-Coloring

1. There is a vertex v of degree at least $\tau = n^{1/3}$
   - Consider List 3-Coloring: lists are subsets of $\{1, 2, 3\}$.
   - We can get rid of vertices with one-element lists.
   - Possible lists are $\{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$.
   - At least $n^{1/3}/4$ neighbors of v have the same list L.
   - There is a color c shared by L and $L(v)$.
   - Branch: either v gets color c or not.
   - $N =$ total size of all lists $\Rightarrow N \leq 3n$.

   $$F(N) \leq F(N-1) + F(N - n^{1/3}/4) \leq$$
2. $m \leq n^{4/3} \Rightarrow$ balanced separator of size $\mathcal{O}(n^{2/3})$
   - Guess the solution on S and recurse $\Rightarrow 2^{\widetilde{\mathcal{O}}(n^{2/3})}$ time.

3-Coloring

1. There is a vertex v of degree at least $\tau = n^{1/3}$
   - Consider List 3-Coloring: lists are subsets of $\{1, 2, 3\}$.
   - We can get rid of vertices with one-element lists.
   - Possible lists are $\{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$.
   - At least $n^{1/3}/4$ neighbors of v have the same list L.
   - There is a color c shared by L and $L(v)$.
   - Branch: either v gets color c or not.
   - $N =$ total size of all lists $\Rightarrow N \le 3n$.

   $$F(N) \le F(N-1) + F(N - n^{1/3}/4) \le 2^{\widetilde{\mathcal{O}}(N^{2/3})} =$$

2. $m \le n^{4/3} \Rightarrow$ balanced separator of size $\mathcal{O}(n^{2/3})$
   - Guess the solution on S and recurse $\Rightarrow 2^{\widetilde{\mathcal{O}}(n^{2/3})}$ time.

3-Coloring

1. There is a vertex v of degree at least $\tau = n^{1/3}$
   - Consider List 3-Coloring: lists are subsets of $\{1, 2, 3\}$.
   - We can get rid of vertices with one-element lists.
   - Possible lists are $\{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$.
   - At least $n^{1/3}/4$ neighbors of v have the same list L.
   - There is a color c shared by L and $L(v)$.
   - Branch: either v gets color c or not.
   - $N =$ total size of all lists $\Rightarrow N \leq 3n$.

   $$F(N) \leq F(N-1) + F(N - n^{1/3}/4) \leq 2^{\widetilde{\mathcal{O}}(N^{2/3})} = 2^{\widetilde{\mathcal{O}}(n^{2/3})}$$

2. $m \leq n^{4/3} \Rightarrow$ balanced separator of size $\mathcal{O}(n^{2/3})$
   - Guess the solution on S and recurse $\Rightarrow 2^{\widetilde{\mathcal{O}}(n^{2/3})}$ time.

3-Coloring

1. There is a vertex v of degree at least $\tau = n^{1/3}$
   - Consider List 3-Coloring: lists are subsets of $\{1, 2, 3\}$.
   - We can get rid of vertices with one-element lists.
   - Possible lists are $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$, $\{1, 2, 3\}$.
   - At least $n^{1/3}/4$ neighbors of v have the same list L.
   - There is a color c shared by L and $L(v)$.
   - Branch: either v gets color c or not.
   - $N =$ total size of all lists $\Rightarrow N \leq 3n$.

   $$F(N) \leq F(N-1) + F(N - n^{1/3}/4) \leq 2^{\widetilde{\mathcal{O}}(N^{2/3})} = 2^{\widetilde{\mathcal{O}}(n^{2/3})}$$

2. $m \leq n^{4/3} \Rightarrow$ balanced separator of size $\mathcal{O}(n^{2/3})$
   - Guess the solution on S and recurse $\Rightarrow 2^{\widetilde{\mathcal{O}}(n^{2/3})}$ time.

What about 4-Coloring?

- The second step (divide & conquer) works.

- In List 4-Coloring lists are subsets of $\{1, 2, 3, 4\}$.
- We can get rid of vertices with one-element lists.
- Possible lists are $\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \ldots, \{1,2,3,4\}$.

What about 4-Coloring?

- The second step (divide & conquer) works.

- In List 4-Coloring lists are subsets of $\{1, 2, 3, 4\}$.
- We can get rid of vertices with one-element lists.
- Possible lists are $\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \ldots, \{1,2,3,4\}$.

- If a large-degree vertex v has list $\{1, 2\}$ and almost all of its neighbors have lists $\{3, 4\}$, we don't know what to do!

What about 4-Coloring?

- The second step (divide & conquer) works.

- In List 4-Coloring lists are subsets of $\{1,2,3,4\}$.
- We can get rid of vertices with one-element lists.
- Possible lists are
  $\{1,2\}$, $\{1,3\}$, $\{1,4\}$, $\{2,3\}$, $\{2,4\}$, $\{3,4\}$, $\ldots$,{1,2,3,4}.

- If a large-degree vertex v has list $\{1,2\}$ and almost all of its neighbors have lists $\{3,4\}$, we don't know what to do!

- These edges are meaningless for coloring, why not just remove them?

What about 4-Coloring?

- The second step (divide & conquer) works.
- In List 4-Coloring lists are subsets of $\{1, 2, 3, 4\}$.
- We can get rid of vertices with one-element lists.
- Possible lists are $\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \ldots, \{1,2,3,4\}$.
- If a large-degree vertex v has list $\{1, 2\}$ and almost all of its neighbors have lists $\{3, 4\}$, we don't know what to do!
- These edges are meaningless for coloring, why not just remove them?

The resulting graph might not be a string graph :-(

What about 4-Coloring?

- The second step (divide & conquer) works.

- In List 4-Coloring lists are subsets of $\{1, 2, 3, 4\}$.
- We can get rid of vertices with one-element lists.
- Possible lists are
  $\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \ldots, \{1,2,3,4\}$.

- If a large-degree vertex v has list $\{1, 2\}$ and almost all of its neighbors have lists $\{3, 4\}$, we don't know what to do!

- These edges are meaningless for coloring, why not just remove them?

The resulting graph might not be a string graph :-(

$\Rightarrow$ We cannot use the separator theorem!

k-Coloring of String Graphs

Theorem [Bonnet & Rz. 2018] k-Coloring for string graphs:

1. for $k = 3$, can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$,
2. for $k \geq 4$, cannot be solved in time $2^{o(n)}$ (under the ETH).

k-Coloring of String Graphs

Theorem [Bonnet & Rz. 2018] k-Coloring for string graphs:

1. for $k = 3$, can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$,
2. for $k \geq 4$, cannot be solved in time $2^{o(n)}$ (under the ETH).

- Let's try to show hardness for List 4-Coloring.

What do we know about the constructed instance G?

k-Coloring of String Graphs

Theorem [Bonnet & Rz. 2018] k-Coloring for string graphs:

1. for $k = 3$, can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$,
2. for $k \geq 4$, cannot be solved in time $2^{o(n)}$ (under the ETH).

- Let's try to show hardness for List 4-Coloring.

What do we know about the constructed instance G?

- Must have $\Theta(n^2)$ edges –

k-Coloring of String Graphs

Theorem [Bonnet & Rz. 2018] k-Coloring for string graphs:

1. for $k = 3$, can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$,
2. for $k \geq 4$, cannot be solved in time $2^{o(n)}$ (under the ETH).

- Let's try to show hardness for List 4-Coloring.

What do we know about the constructed instance G?

- Must have $\Theta(n^2)$ edges –

otherwise we get a sublinear separator.

k-Coloring of String Graphs

Theorem [Bonnet & Rz. 2018] k-Coloring for string graphs:

1. for $k = 3$, can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$,
2. for $k \geq 4$, cannot be solved in time $2^{o(n)}$ (under the ETH).

- Let's try to show hardness for List 4-Coloring.

What do we know about the constructed instance G?

- Must have $\Theta(n^2)$ edges –

  otherwise we get a sublinear separator.

- For (almost) every large-degree vertex v, (almost) each of its neighbors has a totally disjoint list of colors –

k-Coloring of String Graphs

Theorem [Bonnet & Rz. 2018] k-Coloring for string graphs:

1. for $k = 3$, can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$,
2. for $k \geq 4$, cannot be solved in time $2^{o(n)}$ (under the ETH).

- Let's try to show hardness for List 4-Coloring.

What do we know about the constructed instance G?

- Must have $\Theta(n^2)$ edges –

  otherwise we get a sublinear separator.

- For (almost) every large-degree vertex v, (almost) each of its neighbors has a totally disjoint list of colors –

  otherwise we can branch effectively.

k-Coloring of String Graphs

Theorem [Bonnet & Rz. 2018] k-Coloring for string graphs:

1. for $k = 3$, can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$,
2. for $k \geq 4$, cannot be solved in time $2^{o(n)}$ (under the ETH).

- Let's try to show hardness for List 4-Coloring.

What do we know about the constructed instance G?

- Must have $\Theta(n^2)$ edges –

  otherwise we get a sublinear separator.

- For (almost) every large-degree vertex v, (almost) each of its neighbors has a totally disjoint list of colors –

  otherwise we can branch effectively.

Even though G is dense, almost all its edges are meaningless!

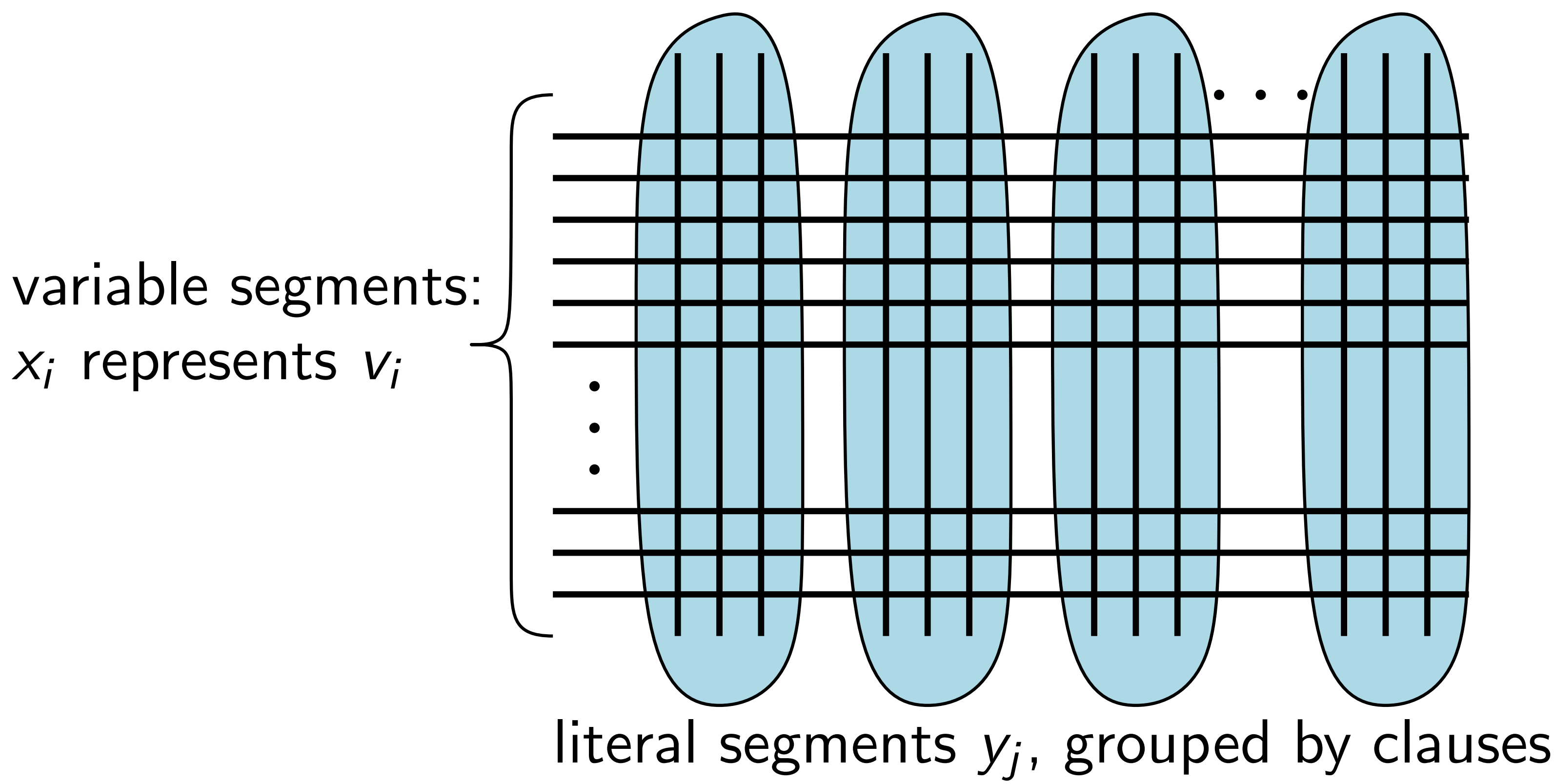
Hardness of List 4-Coloring

- Reduce from 3-Sat with n variables and $m = \mathcal{O}(n)$ clauses.
- Variables: $v_1, v_2 \ldots, v_n$, clauses: $C_1, C_2, \ldots, C_m$
- We show hardness even for segment graphs.

Hardness of List 4-Coloring

- Reduce from 3-Sat with n variables and $m = \mathcal{O}(n)$ clauses.
- Variables: $v_1, v_2 \ldots, v_n$, clauses: $C_1, C_2, \ldots, C_m$
- We show hardness even for segment graphs.

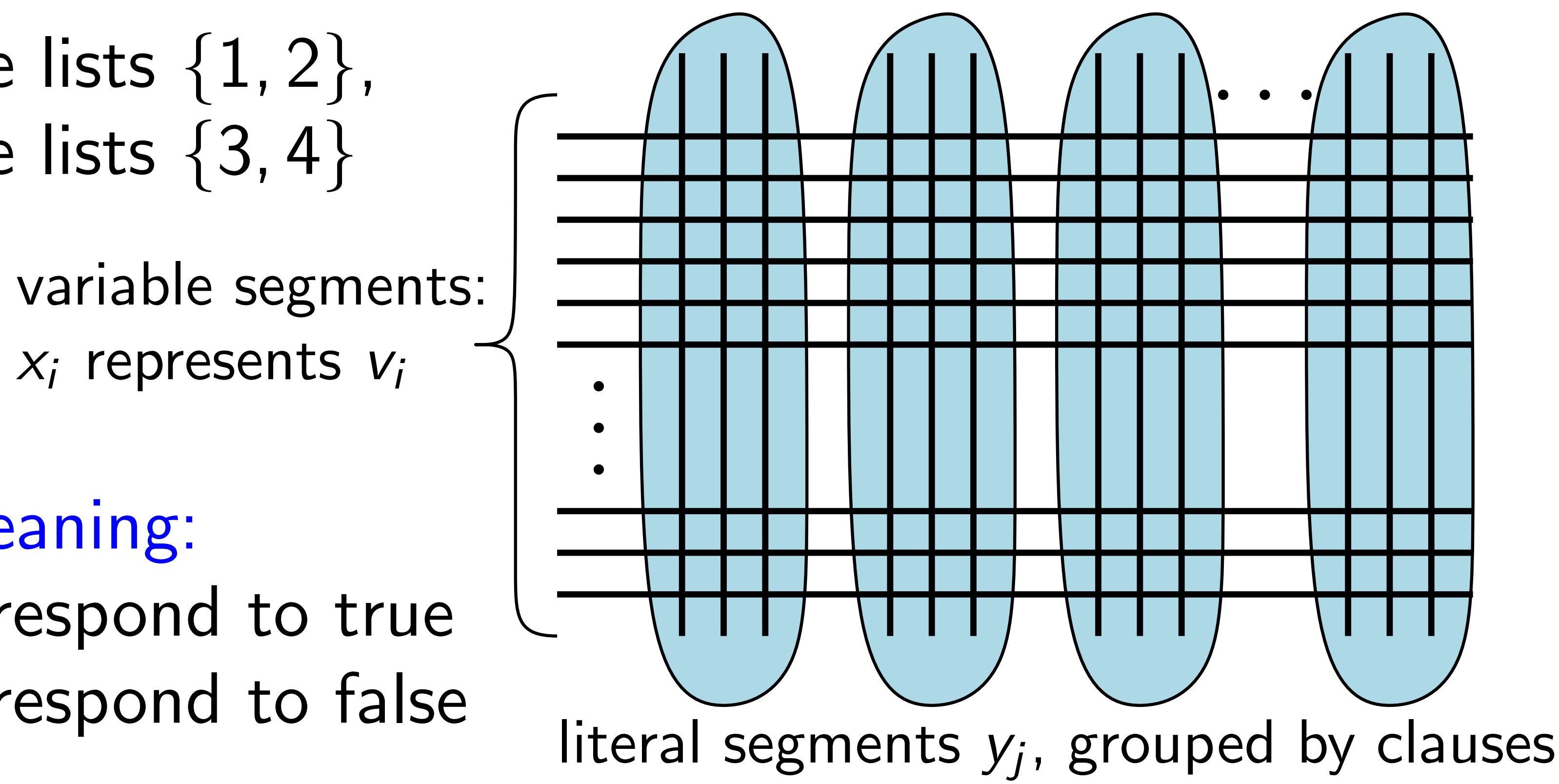
- We introduce a grid-like structure of variable segments (x_i) and literal segments (y_j)



Hardness of LIST 4-COLORING

- Reduce from 3-SAT with n variables and $m = \mathcal{O}(n)$ clauses.
- Variables: $v_1, v_2 \ldots, v_n$, clauses: $C_1, C_2, \ldots, C_m$
- We show hardness even for segment graphs.

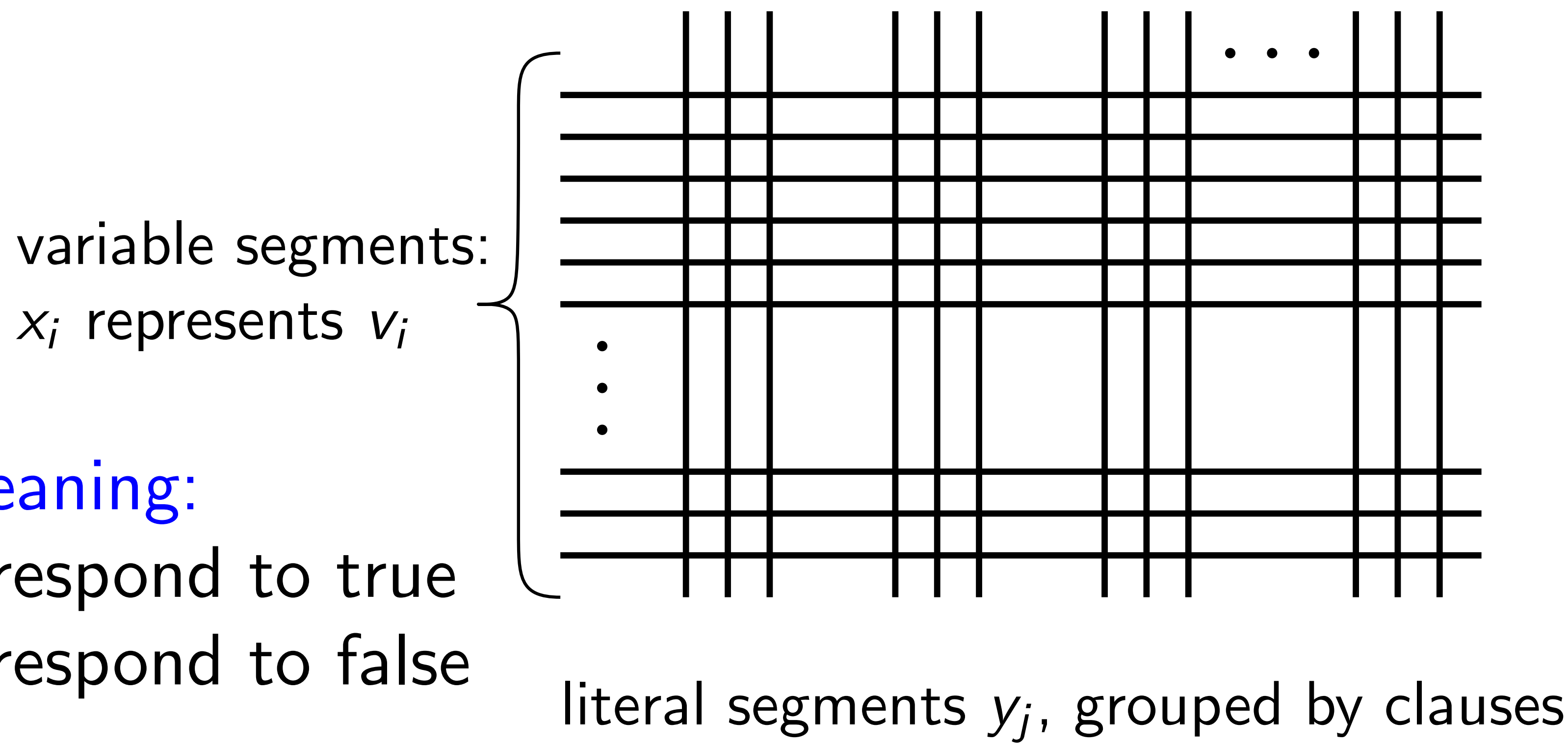
- We introduce a grid-like structure of variable segments (x_i) and literal segments (y_j)
- x_i's have lists $\{1, 2\}$, y_i's have lists $\{3, 4\}$



Intended meaning:
1 and 3 correspond to true
2 and 4 correspond to false

Hardness of List 4-Coloring (cont'd)

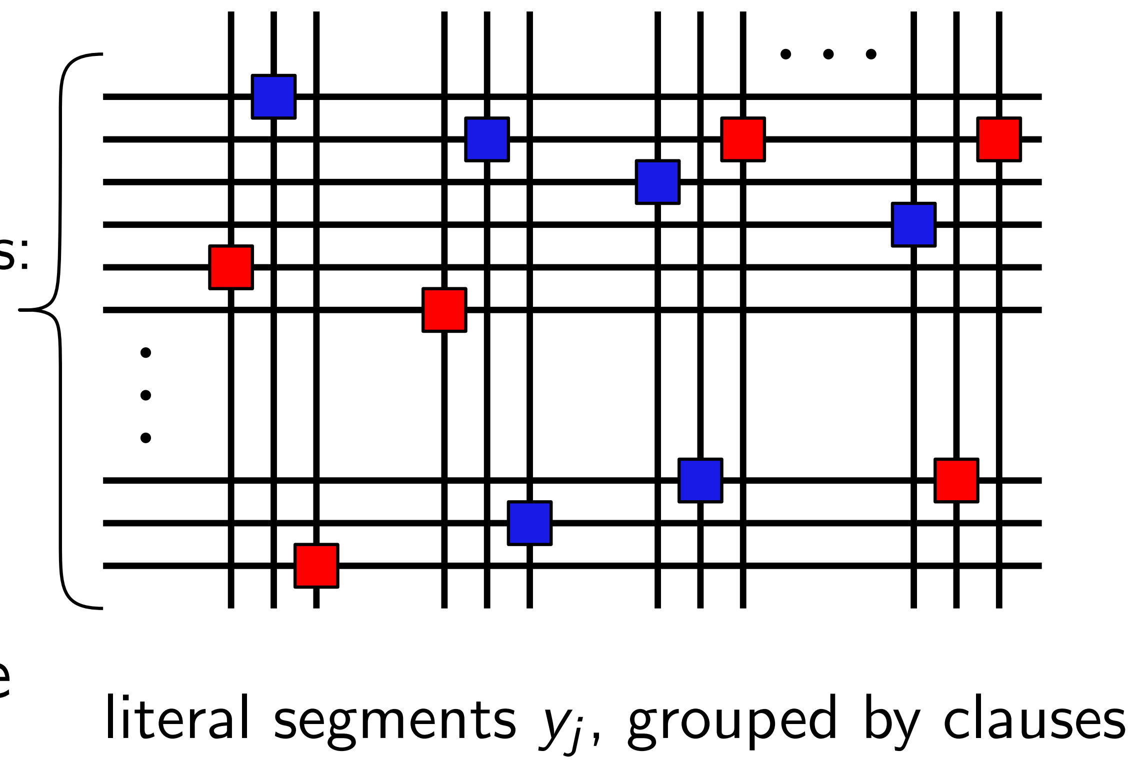
- Consistency of colorings:
  Segments x_i and y_j that correspond to the same variable...



Intended meaning:
1 and 3 correspond to true
2 and 4 correspond to false

Hardness of List 4-Coloring (cont'd)

- Consistency of colorings:
  Segments x_i and y_j that correspond to the same variable...
  – positive occurrence: x_i gets color 1 iff y_j gets color 3
  – negative occurrence: x_i gets color 1 iff y_j gets color 4



Intended meaning:
1 and 3 correspond to true
2 and 4 correspond to false

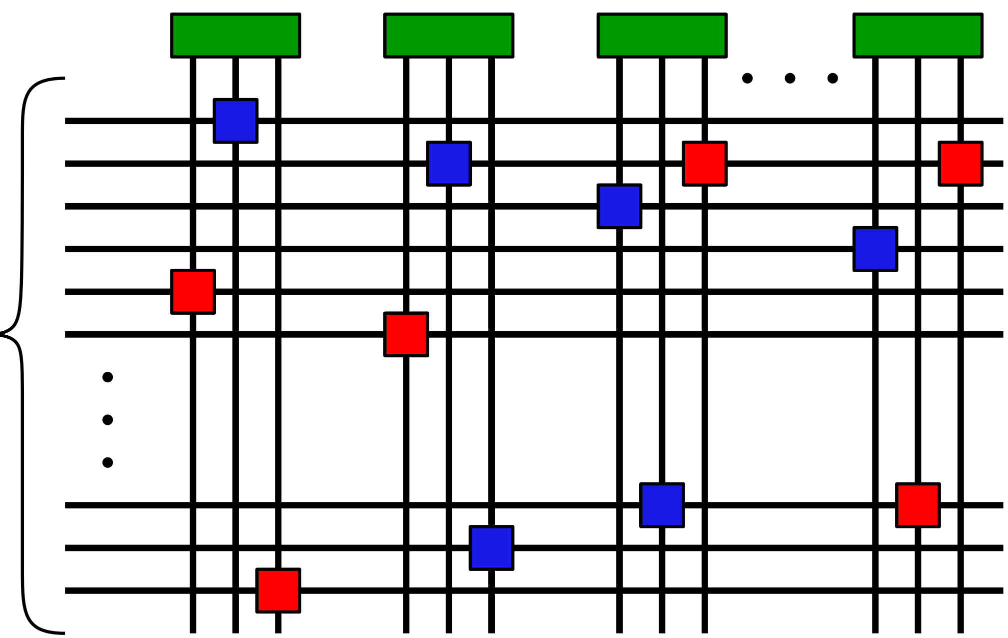
Hardness of List 4-Coloring (cont'd)

- Consistency of colorings:
  Segments x_i and y_j that correspond to the same variable...
  – positive occurrence: x_i gets color 1 iff y_j gets color 3
  – negative occurrence: x_i gets color 1 iff y_j gets color 4

- Satisfiability

  at least one of y's must be colored 3

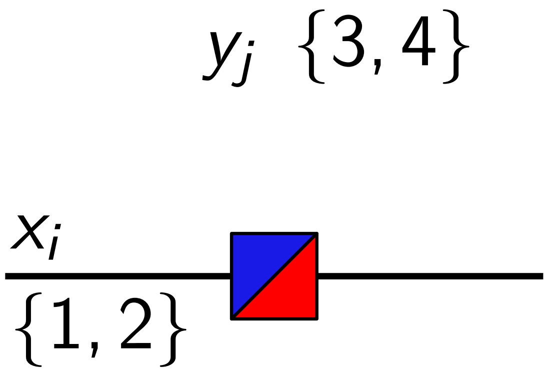
variable segments:
x_i represents v_i



literal segments y_j, grouped by clauses

Intended meaning:
1 and 3 correspond to true
2 and 4 correspond to false

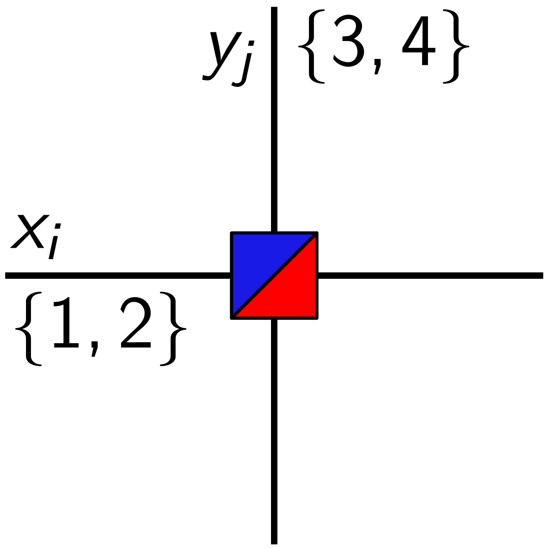
Consistency Gadgets

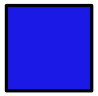


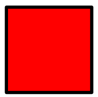
x_i gets color 1 iff y_j gets color 3

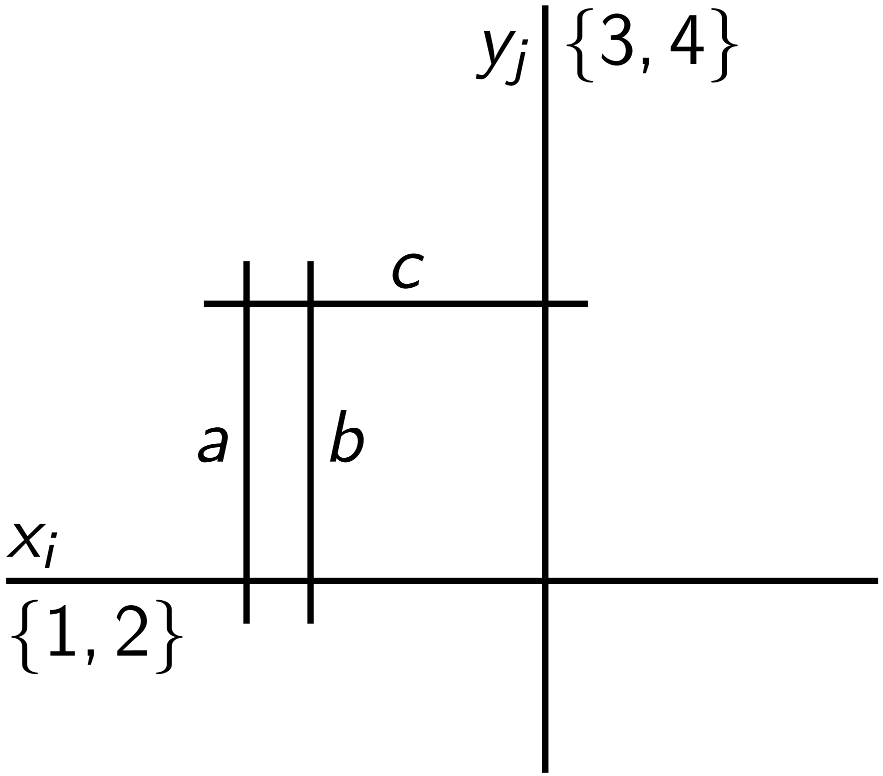
x_i gets color 1 iff y_j gets color 4

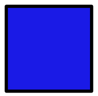
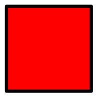
Consistency Gadgets



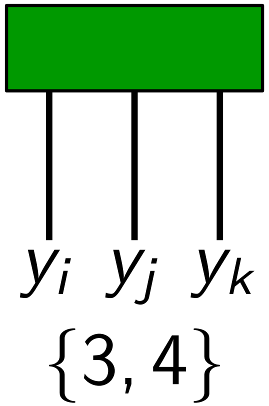
 x_i gets color 1 iff y_j gets color 3

 x_i gets color 1 iff y_j gets color 4



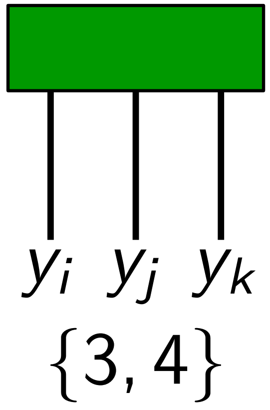
		
a	$1, 3$	$1, 4$
b	$2, 4$	$2, 3$
c	$3, 4$	$3, 4$

Satisfiability Gadget

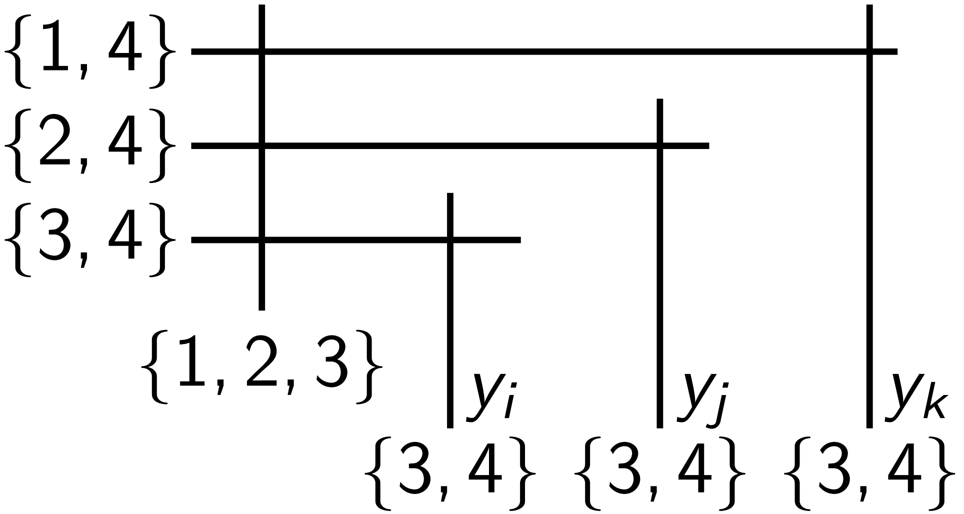


At least one of u_i, y_j, y_k must get color 3.

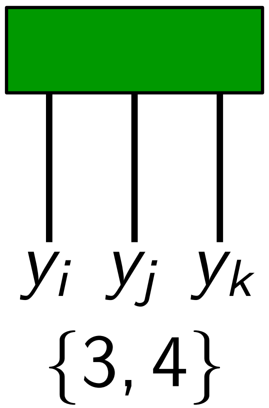
Satisfiability Gadget



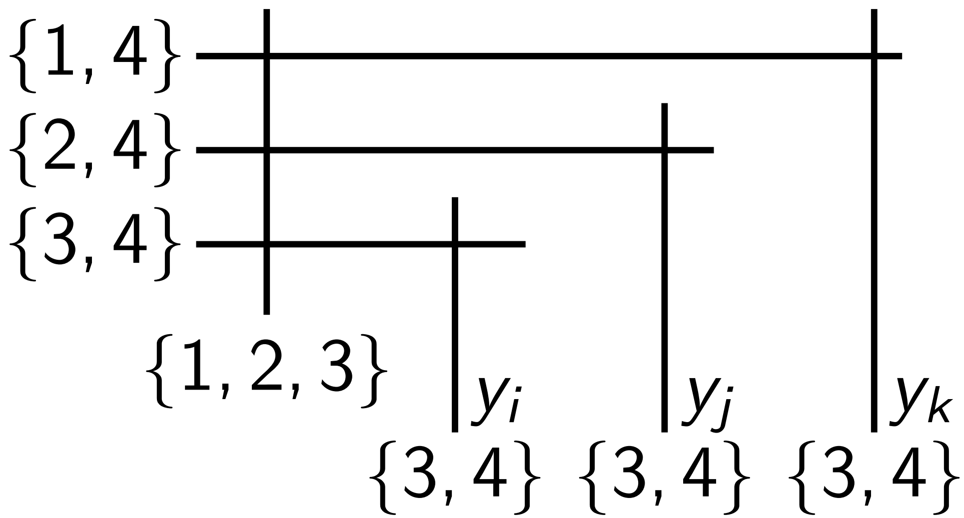
At least one of u_i, y_j, y_k must get color 3.



Satisfiability Gadget



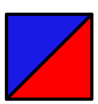
At least one of u_i, y_j, y_k must get color 3.

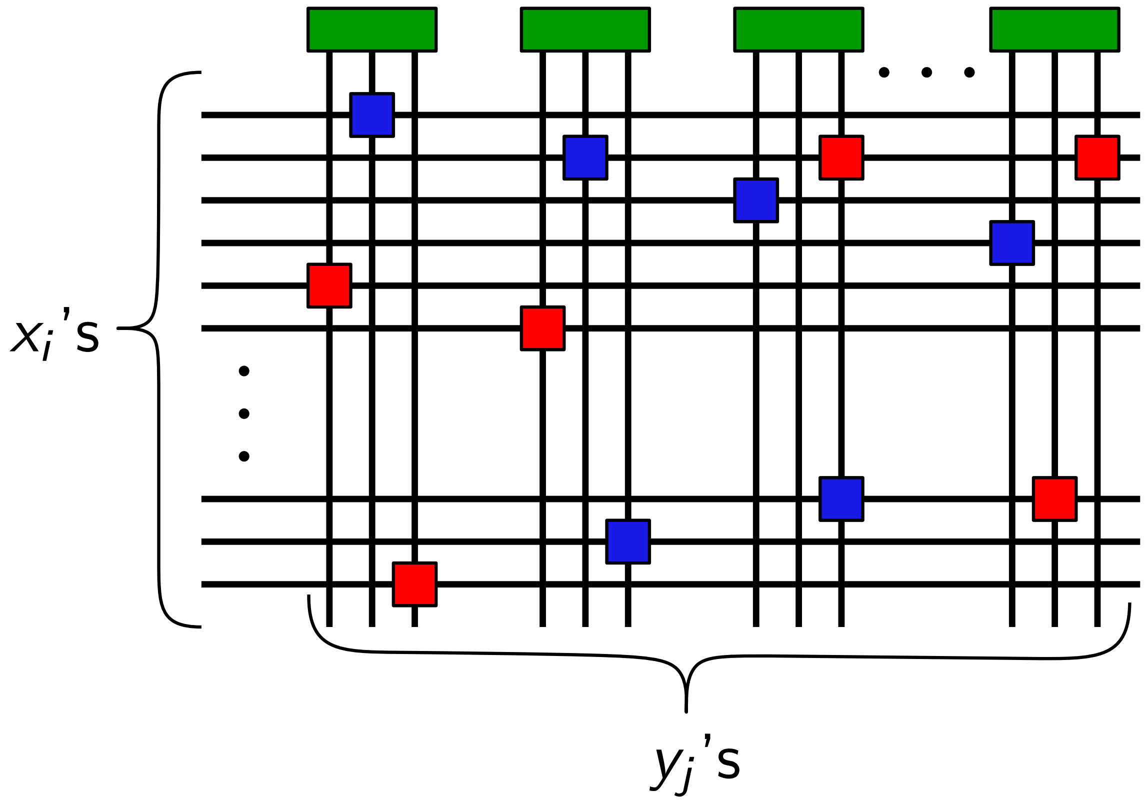


- Note that there are segments with 3-element lists – if all lists have ≤ 2 elements, then the problem is in P!

Wrap-Up

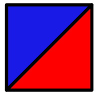
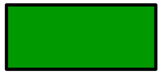
- We've reduced from 3-SAT with n variables and $m = \mathcal{O}(n)$ clauses.
- How many segments do we have?

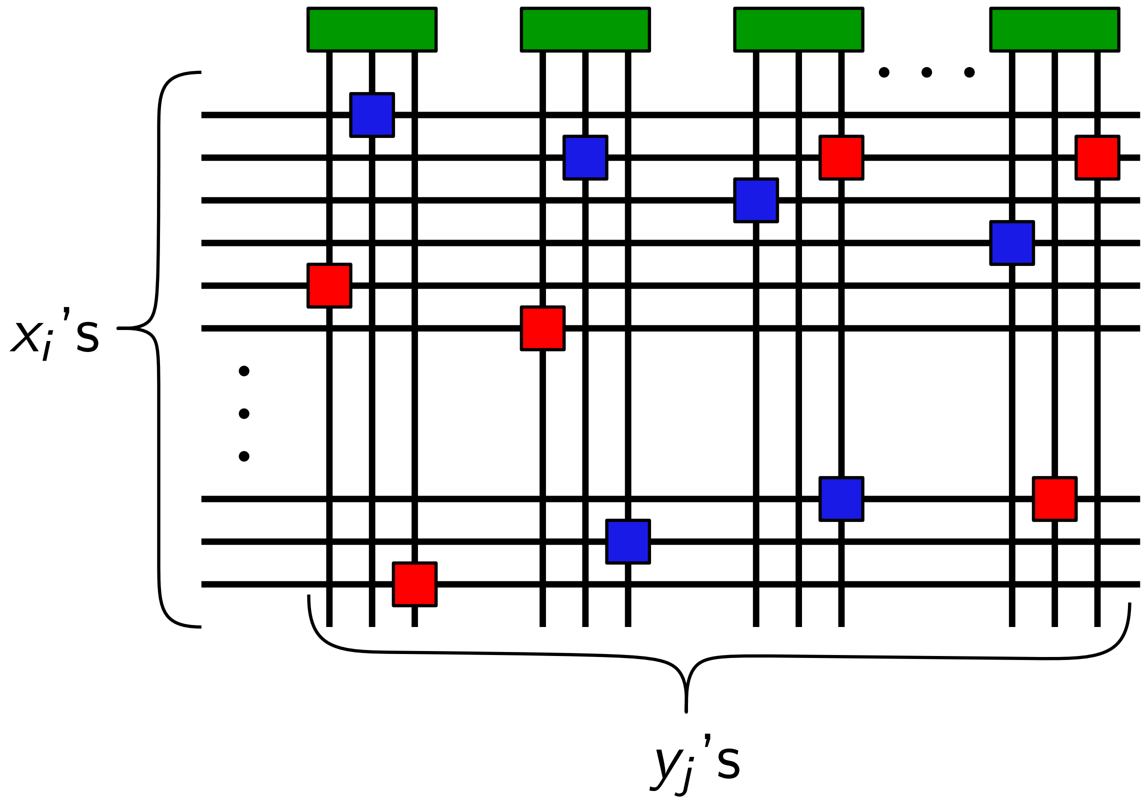
x_i's	
y_j's	
	
	
total	



Wrap-Up

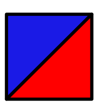
- We've reduced from 3-Sat with n variables and $m = \mathcal{O}(n)$ clauses.
- How many segments do we have?

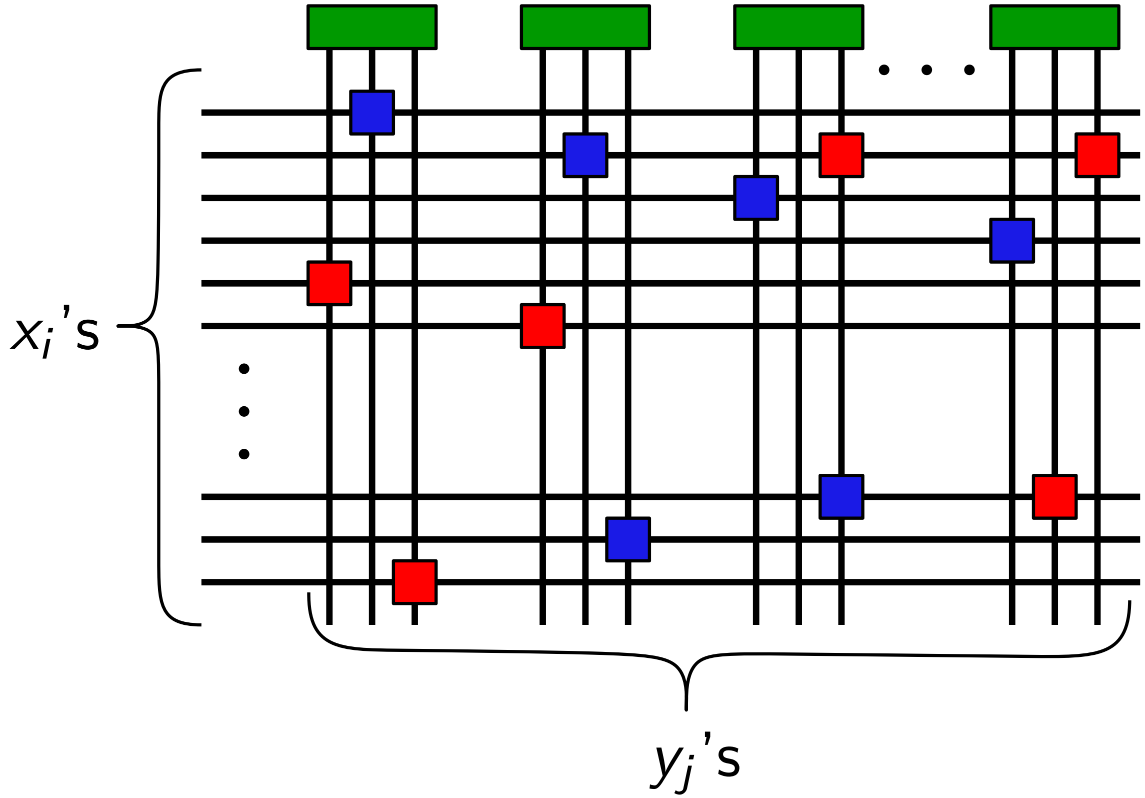
x_i's	n
y_j's	$3m$
	$3m \times 3$
	$m \times 4$
total	$n + 16m$ $\in \mathcal{O}(n)$



Wrap-Up

- We've reduced from 3-SAT with n variables and $m = \mathcal{O}(n)$ clauses.
- How many segments do we have?

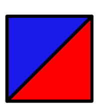
x_i's	n
y_j's	$3m$
	$3m \times 3$
	$m \times 4$
total	$n + 16m \in \mathcal{O}(n)$

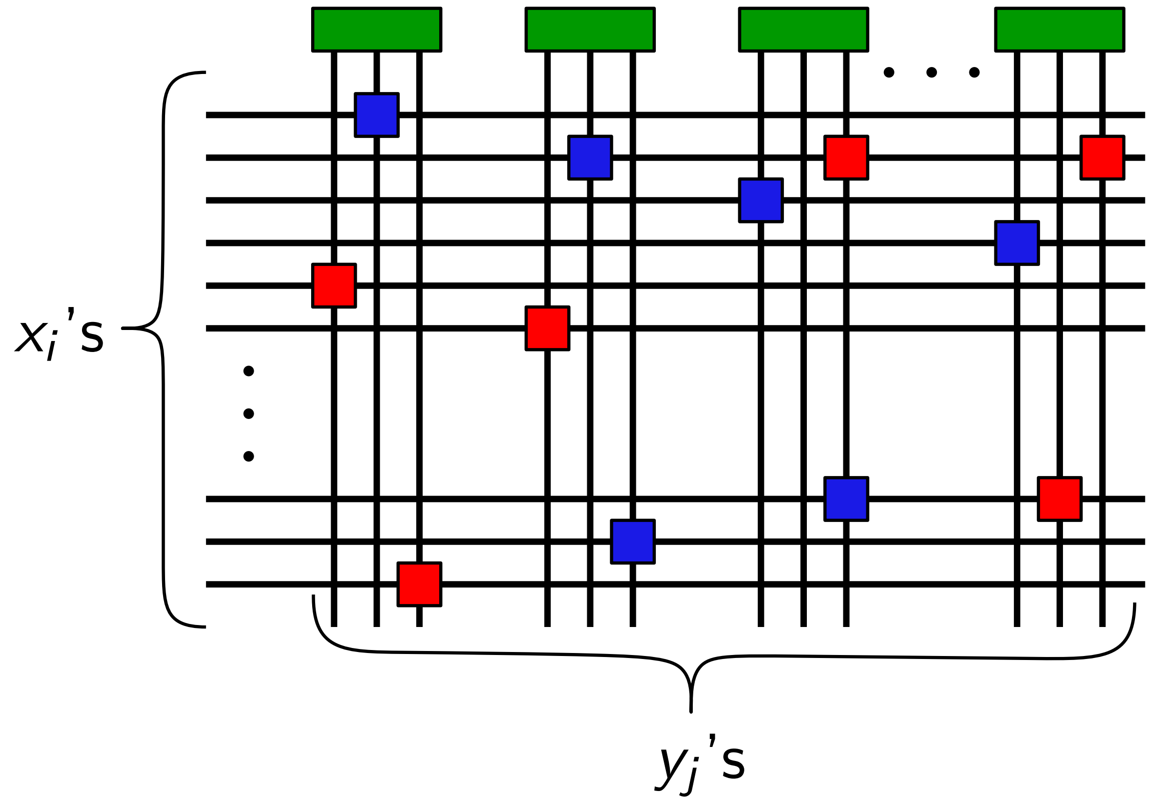


- Assume we could LIST 4-COLOR segment graphs with N vertices in time $2^{o(N)}$.

Wrap-Up

- We've reduced from 3-Sat with n variables and $m = \mathcal{O}(n)$ clauses.
- How many segments do we have?

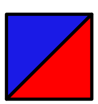
x_i's	n
y_j's	$3m$
	$3m \times 3$
	$m \times 4$
total	$n + 16m$ $\in \mathcal{O}(n)$

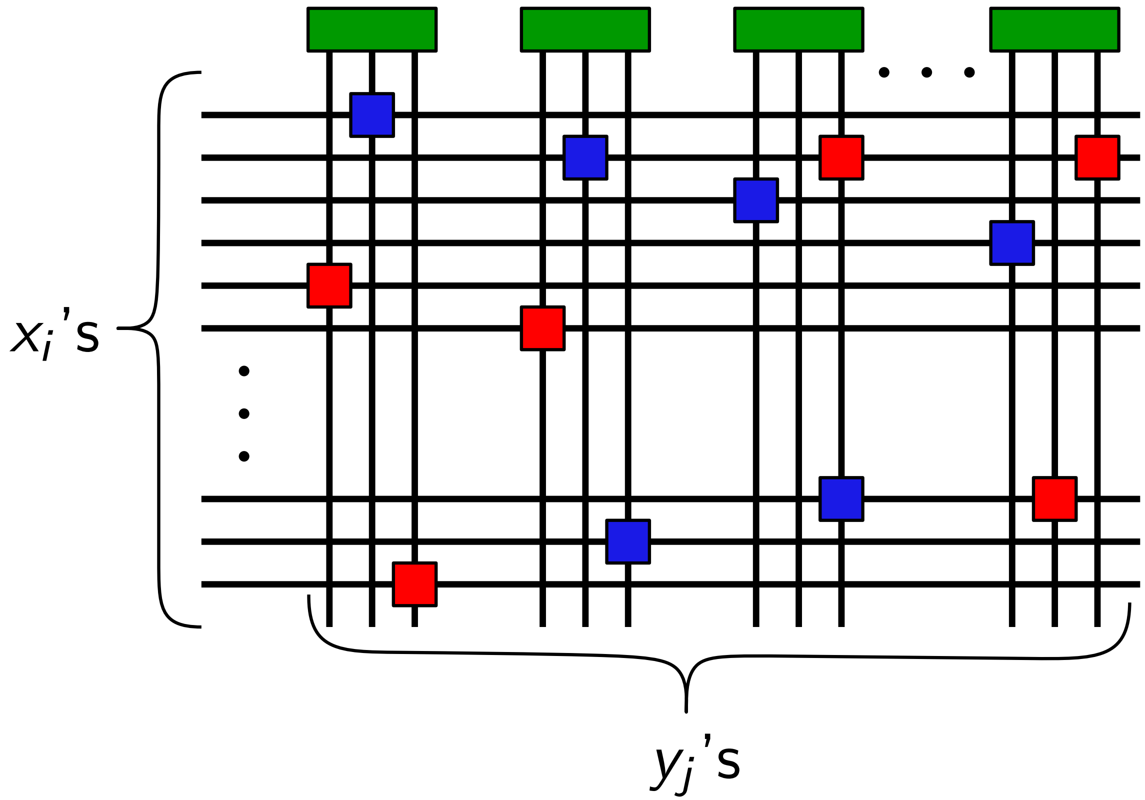


- Assume we could List 4-Color segment graphs with N vertices in time $2^{o(N)}$.
  $\Rightarrow$ Could solve 3-Sat in time $2^{o(n)}$.

Wrap-Up

- We've reduced from 3-SAT with n variables and $m = \mathcal{O}(n)$ clauses.
- How many segments do we have?

x_i's	n
y_j's	$3m$
	$3m \times 3$
	$m \times 4$
total	$n + 16m \in \mathcal{O}(n)$



- Assume we could LIST 4-COLOR segment graphs with N vertices in time $2^{o(N)}$.
  $\Rightarrow$ Could solve 3-SAT in time $2^{o(n)}$.
  $\Rightarrow$ ETH would fail.

– End of Lecture –

Feedback Vertex Set in string graphs

- remove the minimum number vertices to destroy all cycles
- if we have a small separator, the divide & conquer works
- what if we have a vertex of large degree?

Feedback Vertex Set in string graphs

- remove the minimum number vertices to destroy all cycles
- if we have a small separator, the divide & conquer works
- what if we have a vertex of large degree?

Theorem [Lee, 2016].
String graphs with no subgraph $K_{t,t}$ have $\mathcal{O}(n \cdot t \log t)$ edges.

Feedback Vertex Set in string graphs

- remove the minimum number vertices to destroy all cycles
- if we have a small separator, the divide & conquer works
- what if we have a vertex of large degree?

Theorem [Lee, 2016].
String graphs with no subgraph $K_{t,t}$ have $\mathcal{O}(n \cdot t \log t)$ edges.

- combining with the separator of size $\mathcal{O}(\sqrt{m})$, we get

Corollary. Every string graph either has a biclique $K_{t,t}$ or a balanced separator of size $\widetilde{\mathcal{O}}(\sqrt{n \cdot t})$.

Feedback Vertex Set in string graphs

Corollary. Every string graph either has a biclique $K_{t,t}$ or a balanced separator of size $\widetilde{\mathcal{O}}(\sqrt{n \cdot t})$.

- set $t = n^{1/3}$

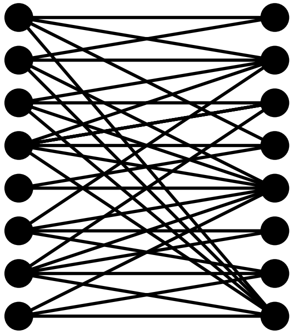
Feedback Vertex Set in string graphs

Corollary. Every string graph either has a biclique $K_{t,t}$ or a balanced separator of size $\widetilde{\mathcal{O}}(\sqrt{n \cdot t})$.

- set $t = n^{1/3}$

1. if there are at least $\widetilde{\Omega}(n^{4/3})$ edges

- there is a biclique $K_{n^{1/3},n^{1/3}}$ for $t = n^{1/3}$, classes A and B



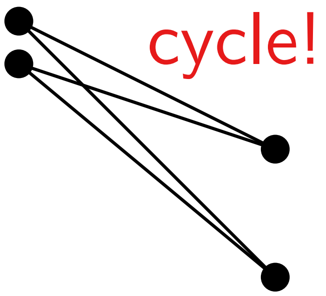
Feedback Vertex Set in string graphs

Corollary. Every string graph either has a biclique $K_{t,t}$ or a balanced separator of size $\widetilde{\mathcal{O}}(\sqrt{n \cdot t})$.

- set $t = n^{1/3}$

1. if there are at least $\widetilde{\Omega}(n^{4/3})$ edges

- there is a biclique $K_{n^{1/3},n^{1/3}}$ for $t = n^{1/3}$, classes A and B
- we must remove all but one vertex from A or B



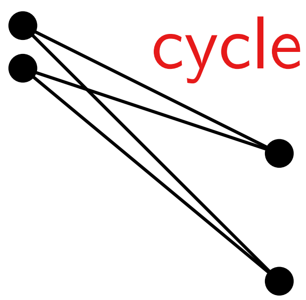
Feedback Vertex Set in string graphs

Corollary. Every string graph either has a biclique $K_{t,t}$ or a balanced separator of size $\widetilde{\mathcal{O}}(\sqrt{n \cdot t})$.

- set $t = n^{1/3}$

1. if there are at least $\widetilde{\Omega}(n^{4/3})$ edges

- there is a biclique $K_{n^{1/3},n^{1/3}}$ for $t = n^{1/3}$, classes A and B
- we must remove all but one vertex from A or B
- branch: we select a class (2 ways) and a vertex ($n^{1/3}$ ways) that might survive



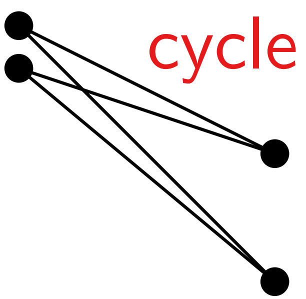
$$F(n) \leq 2n^{1/3} \cdot F(n - n^{1/3}) \leq 2^{\widetilde{\mathcal{O}}(n^{2/3})}$$

Feedback Vertex Set in string graphs

Corollary. Every string graph either has a biclique $K_{t,t}$ or a balanced separator of size $\widetilde{\mathcal{O}}(\sqrt{n \cdot t})$.

- set $t = n^{1/3}$

1. if there are at least $\widetilde{\Omega}(n^{4/3})$ edges
   - there is a biclique $K_{n^{1/3},n^{1/3}}$ for $t = n^{1/3}$, classes A and B
   - we must remove all but one vertex from A or B
   - branch: we select a class (2 ways) and a vertex ($n^{1/3}$ ways) that might survive



$$F(n) \le 2n^{1/3} \cdot F(n - n^{1/3}) \le 2^{\widetilde{\mathcal{O}}(n^{2/3})}$$

2. otherwise there is a balanced separator of size $\widetilde{\mathcal{O}}(n^{2/3}) \to$ divide & conquer works in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$

total running time is $2^{\widetilde{\mathcal{O}}(n^{2/3})}$

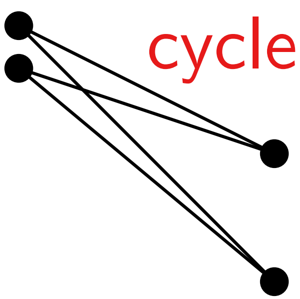
Feedback Vertex Set in string graphs

Corollary. Every string graph either has a biclique $K_{t,t}$ or a balanced separator of size $\widetilde{\mathcal{O}}(\sqrt{n \cdot t})$.

- set $t = n^{1/3}$

1. if there are at least $\widetilde{\Omega}(n^{4/3})$ edges

- there is a biclique $K_{n^{1/3},n^{1/3}}$ for $t = n^{1/3}$, classes A and B
- we must remove all but one vertex from A or B
- branch: we select a class (2 ways) and a vertex ($n^{1/3}$ ways) that might survive



$$F(n) \le 2n^{1/3} \cdot F(n - n^{1/3}) \le 2^{\widetilde{\mathcal{O}}(n^{2/3})}$$

2. otherwise there is a balanced separator of size $\widetilde{\mathcal{O}}(n^{2/3}) \to$ divide & conquer works in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$

total running time is $2^{\widetilde{\mathcal{O}}(n^{2/3})}$

- But no $2^{o(n)}$ algorithm for Odd Cycle Transversal

A detour: the need of representation and robust algorithms

Finding geometric representations

- How fast can we find representations?

Finding geometric representations

- How fast can we find representations?
- Bad news: it is NP-hard to recognize string graphs, segment graphs [Kratochvíl, Matoušek, early 90s], (U) DGs [Breu, Kirkpatrick, '98, Kratochvíl, Hliněný, '01]

Finding geometric representations

- How fast can we find representations?
- Bad news: it is NP-hard to recognize string graphs, segment graphs [Kratochvíl, Matoušek, early 90s], (U) DGs [Breu, Kirkpatrick, '98, Kratochvíl, Hliněný, '01]

- NP-complete? Given a representation, you can verify it.

Finding geometric representations

- How fast can we find representations?
- Bad news: it is NP-hard to recognize string graphs, segment graphs [Kratochvíl, Matoušek, early 90s], (U) DGs [Breu, Kirkpatrick, '98, Kratochvíl, Hliněný, '01]

- NP-complete? Given a representation, you can verify it.
- Bad news: there are n-vertex string graphs, whose every representation requires $2^{\Omega(n)}$ crossing points [KM]
- Bad news: there are n-vertex segment graphs, whose every representation requires coordinates with $2^{\Omega(n)}$ digits [KM]

Finding geometric representations

- How fast can we find representations?
- Bad news: it is NP-hard to recognize string graphs, segment graphs [Kratochvíl, Matoušek, early 90s], (U) DGs [Breu, Kirkpatrick, '98, Kratochvíl, Hliněný, '01]

- NP-complete? Given a representation, you can verify it.
- Bad news: there are n-vertex string graphs, whose every representation requires $2^{\Omega(n)}$ crossing points [KM]
- Bad news: there are n-vertex segment graphs, whose every representation requires coordinates with $2^{\Omega(n)}$ digits [KM]

- is it even decidable? (yes, a non-trivial argument by Tarski)

Finding geometric representations

- How fast can we find representations?
- Bad news: it is NP-hard to recognize string graphs, segment graphs [Kratochvíl, Matoušek, early 90s], (U) DGs [Breu, Kirkpatrick, '98, Kratochvíl, Hliněný, '01]

- NP-complete? Given a representation, you can verify it.
- Bad news: there are n-vertex string graphs, whose every representation requires $2^{\Omega(n)}$ crossing points [KM]
- Bad news: there are n-vertex segment graphs, whose every representation requires coordinates with $2^{\Omega(n)}$ digits [KM]

- is it even decidable? (yes, a non-trivial argument by Tarski)

Theorem [Schaefer, Sedgewick, Štefankovič, '03].
Recognizing string graphs is in NP.

Recognizing segment graphs

- What about segment graphs? Any non-trivial witness?

Theorem [Schaefer, Štefankovič, '17].
Recognizing segment graphs is in $\exists\mathbb{R}$-complete.

Recognizing segment graphs

- What about segment graphs? Any non-trivial witness?

Theorem [Schaefer, Štefankovič, '17].
Recognizing segment graphs is in $\exists\mathbb{R}$-complete.

NP = class of problems
polynomially equivalent to SAT.

SAT: decide if a formula is **true**
$\exists x_1 \exists x_2 \ldots \exists x_n \; \Phi(x_1, \ldots, x_n)$

x_i's are **boolean**,
Φ is quantifier-free and uses
$\wedge, \vee, \neg, =, \rightarrow$

Recognizing segment graphs

- What about segment graphs? Any non-trivial witness?

Theorem [Schaefer, Štefankovič, '17].
Recognizing segment graphs is in $\exists\mathbb{R}$-complete.

NP $=$ class of problems polynomially equivalent to SAT.	$\exists\mathbb{R}$ – class of problems polynomially equivalent to ETR.
SAT: decide if a formula is **true** $\exists x_1 \exists x_2 \dots \exists x_n\ \Phi(x_1, \dots, x_n)$	ETR: decide is a formula is **true** $\exists x_1 \exists x_2 \dots \exists x_n\ \Phi(x_1, \dots, x_n)$
x_i's are **boolean**, Φ is quantifier-free and uses $\wedge, \vee, \neg, =, \rightarrow$	x_i's are **reals**, Φ is quantifier-free and uses $\wedge, \vee, \neg, =, \rightarrow, >, +, -, \times$ (in $\mathbb{R}$)

Recognizing segment graphs

- What about segment graphs? Any non-trivial witness?

Theorem [Schaefer, Štefankovič, '17].
Recognizing segment graphs is in $\exists\mathbb{R}$-complete.

NP $=$ class of problems polynomially equivalent to SAT.	$\exists\mathbb{R}$ – class of problems polynomially equivalent to ETR.
SAT: decide if a formula is **true** $\exists x_1 \exists x_2 \ldots \exists x_n \; \Phi(x_1, \ldots, x_n)$	ETR: decide is a formula is **true** $\exists x_1 \exists x_2 \ldots \exists x_n \; \Phi(x_1, \ldots, x_n)$
x_i's are **boolean**, Φ is quantifier-free and uses $\wedge, \vee, \neg, =, \rightarrow$	x_i's are **reals**, Φ is quantifier-free and uses $\wedge, \vee, \neg, =, \rightarrow, >, +, -, \times$ (in $\mathbb{R}$)

- a strong indication that the problem is not in NP!
- similar for unit disk graphs [Kang, Müller, '12]

What about our algorithms?

Independent Set in disk graphs

1. ply $> n^{1/3} \rightarrow$ a clique of size $> n^{1/3}$, branch
2. ply $\leq n^{1/3} \rightarrow$ a balanced separator S of size $\mathcal{O}(n^{2/3})$
3. guess the solution on S
4. recurse using divide & conquer

Total running time: $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

- where do we need a representation?

What about our algorithms?

Independent Set in disk graphs

1. ply $> n^{1/3} \rightarrow$ a clique of size $> n^{1/3}$, branch
2. ply $\leq n^{1/3} \rightarrow$ a balanced separator S of size $\mathcal{O}(n^{2/3})$
3. guess the solution on S
4. recurse using divide & conquer

Total running time: $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

- where do we need a representation?

What about our algorithms?

INDEPENDENT SET in disk graphs

1. ply $> n^{1/3} \to$ a clique of size $> n^{1/3}$, branch
2. ply $\leq n^{1/3} \to$ a balanced separator S of size $\mathcal{O}(n^{2/3})$
3. guess the solution on S
4. recurse using divide & conquer

Total running time: $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

- where do we need a representation?
- enumerating all possibilities takes time $n^{n^{2/3}} = 2^{\widetilde{\mathcal{O}}(n^{2/3})}$

What about our algorithms?

INDEPENDENT SET in disk graphs

1. if we find a clique of size $> n^{1/3}$, branch
2. otherwise, find a balanced separator S of size $\mathcal{O}(n^{2/3})$
3. guess the solution on S
4. recurse using divide & conquer

Total running time: $2^{\widetilde{\mathcal{O}}(n^{2/3})} + 2^{\widetilde{\mathcal{O}}(n^{2/3})} = 2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

- where do we need a representation?
- enumerating all possibilities takes time $n^{n^{2/3}} = 2^{\widetilde{\mathcal{O}}(n^{2/3})}$
- we do not really need a representation!

Robust algorithms

- An algorithm is robust, if it either
    - computes the correct solution, or
    - correctly concludes that the input does not belong to the right class (here: disk graphs)
- notion introduced by Spinrad

Robust algorithms

- An algorithm is robust, if it either
  - computes the correct solution, or
  - correctly concludes that the input does not belong to the right class (here: disk graphs)
- notion introduced by Spinrad
- it's not really an algorithm for disk graphs, but for the class $\mathcal{X} =$graphs with balanced separators of size $\mathcal{O}(\sqrt{n \cdot \omega(G)})$
- disk graphs $\subseteq \mathcal{X}$

Robust algorithms

- An algorithm is robust, if it either
    - computes the correct solution, or
    - correctly concludes that the input does not belong to the right class (here: disk graphs)
- notion introduced by Spinrad
- it's not really an algorithm for disk graphs, but for the class $\mathcal{X} =$graphs with balanced separators of size $\mathcal{O}(\sqrt{n \cdot \omega(G)})$
- disk graphs $\subseteq \mathcal{X}$
- on the other hand, our hardness results hold even if a geometric representation is given

When large cliques do not help

CLIQUE in disk graphs

- CLIQUE is polynomially solvable in UDG [Clark et al., 1990]
- the complexity for DG is open

- the existence of a large clique does not make the problem any easier!

Clique in disk graphs

- Clique is polynomially solvable in UDG [Clark et al., 1990]
- the complexity for DG is open
- the existence of a large clique does not make the problem any easier!
- we need to make our hands dirty and look at the properties of geometric representations
- by some epsilon-perturbation we can assume that no three centers are aligned

Notation: vertex v_i is represented by a disk with the center c_i

C_4's in disk graphs

Simple observation.

In any disk representation of of C_4 with vertices v_1, v_2, v_3, v_4:

the line $\ell(c_2 c_4)$ crosses the segment $c_1 c_3$, or

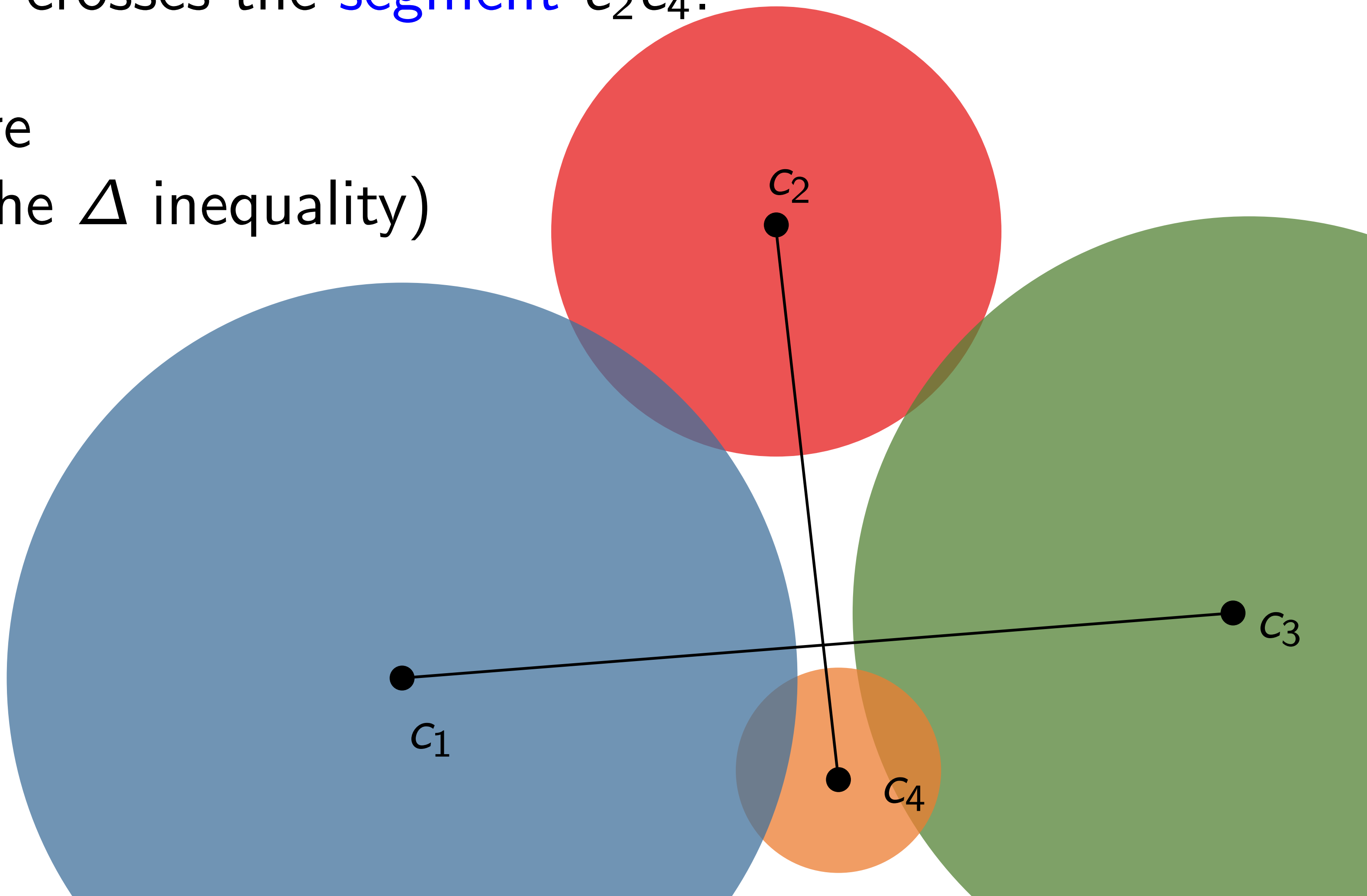
the line $\ell(c_1 c_3)$ crosses the segment $c_2 c_4$.

C_4's in disk graphs

Simple observation.

In any disk representation of of C_4 with vertices v_1, v_2, v_3, v_4:
the line $\ell(c_2 c_4)$ crosses the segment $c_1 c_3$, or
the line $\ell(c_1 c_3)$ crosses the segment $c_2 c_4$.

Proof by picture
(follows from the Δ inequality)

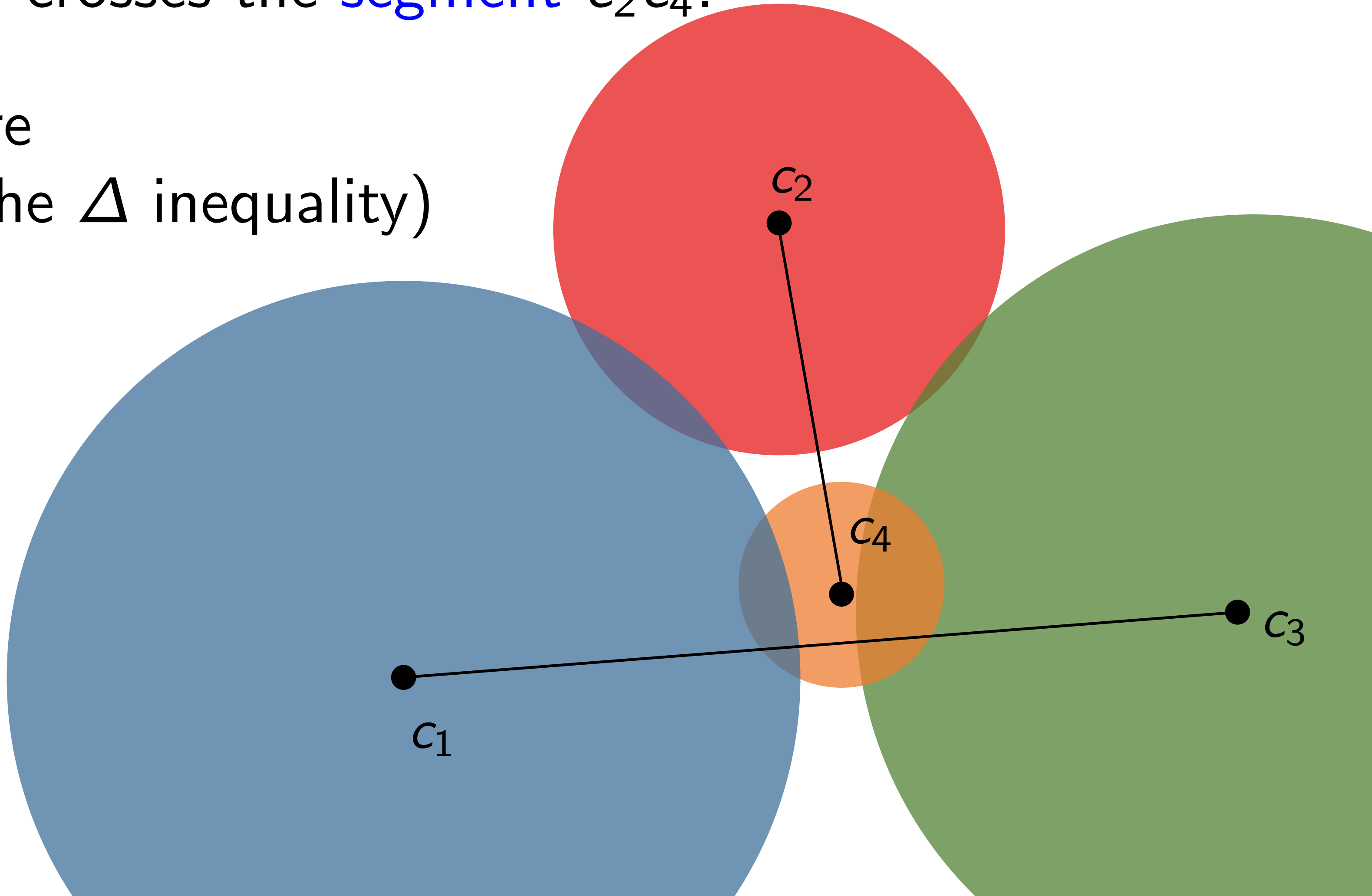


C_4's in disk graphs

Simple observation.

In any disk representation of of C_4 with vertices v_1, v_2, v_3, v_4:
the line $\ell(c_2 c_4)$ crosses the segment $c_1 c_3$, or
the line $\ell(c_1 c_3)$ crosses the segment $c_2 c_4$.

Proof by picture
(follows from the Δ inequality)

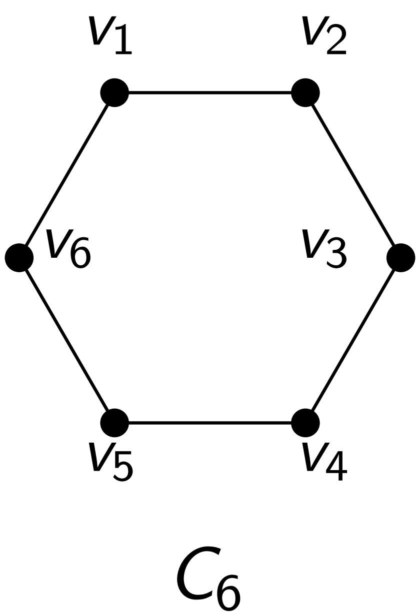


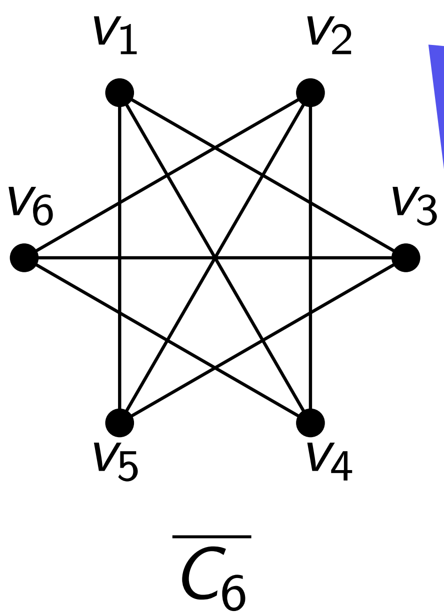
Non-disk graphs

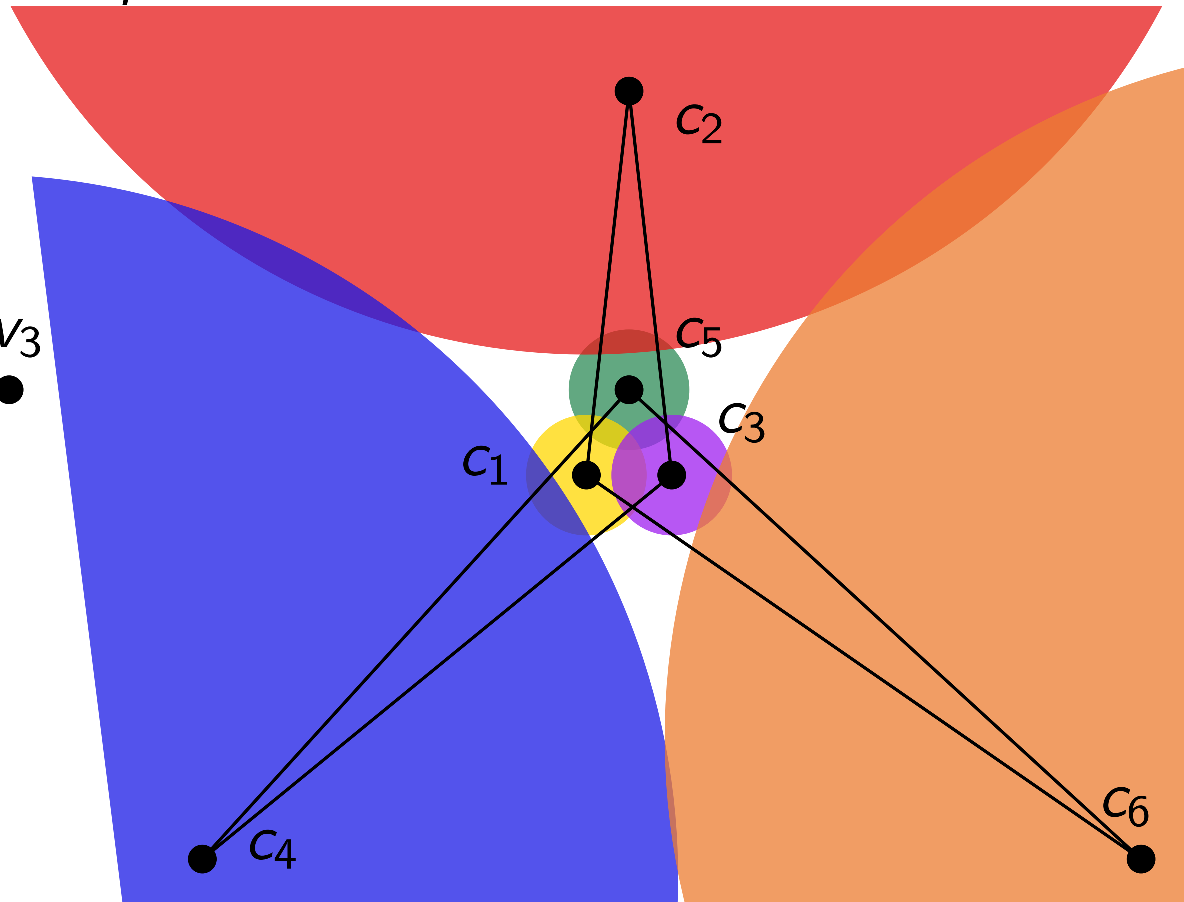
Observation [Bonnet, Giannopoulos, Kim, Rz. Sikora, 2018].
For odd p, q, the graph $G = \overline{C_p + C_q}$ is not a disk graph.

Proof by contradiction.

- suppose there is a representation
- let $S_1, \ldots, S_p$ and $S'_1, \ldots, S'_q$ be segments of the co-cycles







Non-disk graphs

Observation [Bonnet, Giannopoulos, Kim, Rz. Sikora, 2018].
For odd p, q, the graph $G = \overline{C_p + C_q}$ is not a disk graph.

Proof by contradiction.

- suppose there is a representation
- let $S_1, \ldots, S_p$ and $S'_1, \ldots, S'_q$ be segments of the co-cycles
- every S_i and every S'_j correspond to $2K_2$ in $\overline{G}$
  $\rightarrow$ their endpoints induce a C_4 in G
  $\rightarrow \ell(S_i)$ crosses S_j or $\ell(S_j)$ crosses S_j

Non-disk graphs

Observation [Bonnet, Giannopoulos, Kim, Rz. Sikora, 2018].
For odd p, q, the graph $G = \overline{C_p + C_q}$ is not a disk graph.

Proof by contradiction.

- suppose there is a representation
- let $S_1, \ldots, S_p$ and $S'_1, \ldots, S'_q$ be segments of the co-cycles
- $(\star)$: for every i, j either $\ell(S_i)$ crosses S_j or $\ell(S_j)$ crosses S_j

Non-disk graphs

Observation [Bonnet, Giannopoulos, Kim, Rz. Sikora, 2018].
For odd p, q, the graph $G = \overline{C_p + C_q}$ is not a disk graph.

Proof by contradiction.

- suppose there is a representation
- let $S_1, \dots, S_p$ and $S'_1, \dots, S'_q$ be segments of the co-cycles
- $(\star)$: for every i, j either $\ell(S_i)$ crosses S_j or $\ell(S_j)$ crosses S_j
- define: $a_i =$ number of S'_j's intersected by $\ell(S_i)$
  $b_i =$ number of $\ell(S'_j)$'s intersected by S_i
  $c_i =$ number of S'_j's intersected by S_i

$\sum_{i=1}^{p}(a_i + b_i - c_i) =$ number of pairs i, j satisfying $(\star)$ $= pq$

Non-disk graphs

Observation [Bonnet, Giannopoulos, Kim, Rz. Sikora, 2018].
For odd p, q, the graph $G = \overline{C_p + C_q}$ is not a disk graph.

Proof by contradiction.

- suppose there is a representation
- let $S_1, \ldots, S_p$ and $S'_1, \ldots, S'_q$ be segments of the co-cycles
- $(\star)$: for every i, j either $\ell(S_i)$ crosses S_j or $\ell(S_j)$ crosses S_j
- define: $a_i =$ number of S'_j's intersected by $\ell(S_i)$
  $b_i =$ number of $\ell(S'_j)$'s intersected by S_i
  $c_i =$ number of S'_j's intersected by S_i

$\sum_{i=1}^{p}(a_i + b_i - c_i) =$ number of pairs i, j satisfying $(\star)$ $= pq$

- $a_i = \#$ of points where a line crosses a closed curve: even

Non-disk graphs

Observation [Bonnet, Giannopoulos, Kim, Rz. Sikora, 2018].
For odd p, q, the graph $G = \overline{C_p + C_q}$ is not a disk graph.

Proof by contradiction.

- suppose there is a representation
- let $S_1, \ldots, S_p$ and $S'_1, \ldots, S'_q$ be segments of the co-cycles
- $(\star)$: for every i, j either $\ell(S_i)$ crosses S_j or $\ell(S_j)$ crosses S_j
- define: $a_i =$ number of S'_j's intersected by $\ell(S_i)$
  $b_i =$ number of $\ell(S'_j)$'s intersected by S_i
  $c_i =$ number of S'_j's intersected by S_i

$\sum_{i=1}^{p}(a_i + b_i - c_i) =$ number of pairs i, j satisfying $(\star)$ $= pq$

- $a_i = \#$ of points where a line crosses a closed curve: even
- $\sum_{i=1}^{p} b_i = \sum_{i=j}^{q} a'_j$: also even

Non-disk graphs

Observation [Bonnet, Giannopoulos, Kim, Rz. Sikora, 2018].
For odd p, q, the graph $G = \overline{C_p + C_q}$ is not a disk graph.

Proof by contradiction.

- suppose there is a representation
- let $S_1, \dots, S_p$ and $S'_1, \dots, S'_q$ be segments of the co-cycles
- $(\star)$: for every i, j either $\ell(S_i)$ crosses S_j or $\ell(S_j)$ crosses S_j
- define: $a_i =$ number of S'_j's intersected by $\ell(S_i)$
  $b_i =$ number of $\ell(S'_j)$'s intersected by S_i
  $c_i =$ number of S'_j's intersected by S_i

$\sum_{i=1}^{p}(a_i + b_i - c_i) =$ number of pairs i, j satisfying $(\star)$ $= pq$

- $a_i = \#$ of points where a line crosses a closed curve: even
- $\sum_{i=1}^{p} b_i = \sum_{i=j}^{q} a'_j$: also even
- $c_i = \#$ of intersection points of two closed curves: even

Non-disk graphs

Observation [Bonnet, Giannopoulos, Kim, Rz. Sikora, 2018].
For odd p, q, the graph $G = \overline{C_p + C_q}$ is not a disk graph.

Proof by contradiction.

- suppose there is a representation
- let $S_1, \ldots, S_p$ and $S'_1, \ldots, S'_q$ be segments of the co-cycles
- $(\star)$: for every i, j either $\ell(S_i)$ crosses S_j or $\ell(S_j)$ crosses S_j
- define: $a_i =$ number of S'_j's intersected by $\ell(S_i)$
  $b_i =$ number of $\ell(S'_j)$'s intersected by S_i
  $c_i =$ number of S'_j's intersected by S_i

$\sum_{i=1}^{p}(a_i + b_i - c_i) =$ number of pairs i, j satisfying $(\star)$ $= pq$

- $a_i = \#$ of points where a line crosses a closed curve: even
- $\sum_{i=1}^{p} b_i = \sum_{i=j}^{q} a'_j$: also even
- $c_i = \#$ of intersection points of two closed curves: even
- $\sum_{i=1}^{p}(a_i + b_i - c_i) = pq$ is even $\rightarrow$ contradiction $\square$

$\textsc{Clique}$ for disk graphs

Observation [Bonnet, Giannopoulos, Kim, Rz. Sikora, 2018].
For odd p, q, the graph $G = \overline{C_p + C_q}$ is not a disk graph.

Theorem [Györi, Kostochka, Łuczak, '97].
If odd girth is at least δn, then there is X, such that $|X| = \widetilde{\mathcal{O}}(1/\delta)$ and $G - X$ is bipartite.

Clique for disk graphs

Observation [Bonnet, Giannopoulos, Kim, Rz. Sikora, 2018].
For odd p, q, the graph $G = \overline{C_p + C_q}$ is not a disk graph.

Theorem [Györi, Kostochka, Łuczak, '97].
If odd girth is at least δn, then there is X, such that $|X| = \widetilde{\mathcal{O}}(1/\delta)$ and $G - X$ is bipartite.

$$\textsc{Clique} \text{ in } G \equiv \textsc{Independent Set} \text{ in } \overline{G}$$

Clique for disk graphs

Observation [Bonnet, Giannopoulos, Kim, Rz. Sikora, 2018].
For odd p, q, the graph $G = \overline{C_p + C_q}$ is not a disk graph.

Theorem [Györi, Kostochka, Łuczak, '97].
If odd girth is at least δn, then there is X, such that $|X| = \widetilde{\mathcal{O}}(1/\delta)$ and $G - X$ is bipartite.

$$\textsc{Clique} \text{ in } G \equiv \textsc{Independent Set} \text{ in } \overline{G}$$

Independent Set in a co-disk graph:

1. vertex of degree at least $n^{1/3} \to$ branching
2. no odd cycle of length $< n^{1/3} \to$
   there is $|X| = \mathcal{O}(n^{2/3})$ and $G - X$ bipartite
3. odd C of length $\leq n^{1/3}$ and $\Delta \leq n^{1/3} \to$
   $|N[C]| \leq n^{2/3}$ and $G - N[C]$ is bipartite

Clique for disk graphs

Observation [Bonnet, Giannopoulos, Kim, Rz. Sikora, 2018].
For odd p, q, the graph $G = \overline{C_p + C_q}$ is not a disk graph.

Theorem [Györi, Kostochka, Łuczak, '97].
If odd girth is at least δn, then there is X, such that $|X| = \widetilde{\mathcal{O}}(1/\delta)$ and $G - X$ is bipartite.

$$\text{Clique in } G \equiv \text{Independent Set in } \overline{G}$$

Independent Set in a co-disk graph:

1. vertex of degree at least $n^{1/3} \rightarrow$ branching
2. no odd cycle of length $< n^{1/3} \rightarrow$ there is $|X| = \mathcal{O}(n^{2/3})$ and $G - X$ bipartite
3. odd C of length $\leq n^{1/3}$ and $\Delta \leq n^{1/3} \rightarrow$ $|N[C]| \leq n^{2/3}$ and $G - N[C]$ is bipartite

$2^{\widetilde{\mathcal{O}}(n^{2/3})}$ (for item 1)

guess the solution on X or $N[C]$ and finish in poly time (for items 2 and 3)

Clique for disk graphs

Observation [Bonnet, Giannopoulos, Kim, Rz. Sikora, 2018].
For odd p, q, the graph $G = \overline{C_p + C_q}$ is not a disk graph.

Theorem [Györi, Kostochka, Łuczak, '97].
If odd girth is at least δn, then there is X, such that $|X| = \widetilde{\mathcal{O}}(1/\delta)$ and $G - X$ is bipartite.

$$\text{Clique in } G \equiv \text{Independent Set in } \overline{G}$$

Independent Set in a co-disk graph:

1. vertex of degree at least $n^{1/3} \to$ branching
2. no odd cycle of length $< n^{1/3} \to$ there is $|X| = \mathcal{O}(n^{2/3})$ and $G - X$ bipartite
3. odd C of length $\leq n^{1/3}$ and $\Delta \leq n^{1/3} \to$ $|N[C]| \leq n^{2/3}$ and $G - N[C]$ is bipartite

$2^{\widetilde{\mathcal{O}}(n^{2/3})}$

guess the solution on X or $N[C]$ and finish in poly time

Theorem [BGKRzS '18].
Clique in disk graphs can be solved in time $2^{\widetilde{\mathcal{O}}(n^{2/3})}$.

Open problem: MAX CUT in disk graphs

- partition vertices into two sets, to maximize the number of crossing edges
- NP-hard on unit disk graphs, reduction is quadratic $\rightarrow$ no $2^{o(\sqrt{n})}$algorithm
- is there a subexponential algorithm?

Open problem: MAX CUT in disk graphs

- partition vertices into two sets, to maximize the number of crossing edges
- NP-hard on unit disk graphs, reduction is quadratic $\rightarrow$ no $2^{o(\sqrt{n})}$algorithm
- is there a subexponential algorithm?
- Warning: edge-weighted version has no subexponential algorithm on complete graphs!

Open problem: Max Cut in disk graphs

- partition vertices into two sets, to maximize the number of crossing edges
- NP-hard on unit disk graphs, reduction is quadratic $\rightarrow$ no $2^{o(\sqrt{n})}$algorithm
- is there a subexponential algorithm?
- Warning: edge-weighted version has no subexponential algorithm on complete graphs!
- complexity even unclear for (unit) interval graphs

Episode 2: parameterized algorithms

Geometric separators

k-Independent Set in unit disk graphs

- is there an independent set of size at least k?
- are there k disjoint disks?

k-Independent Set in unit disk graphs

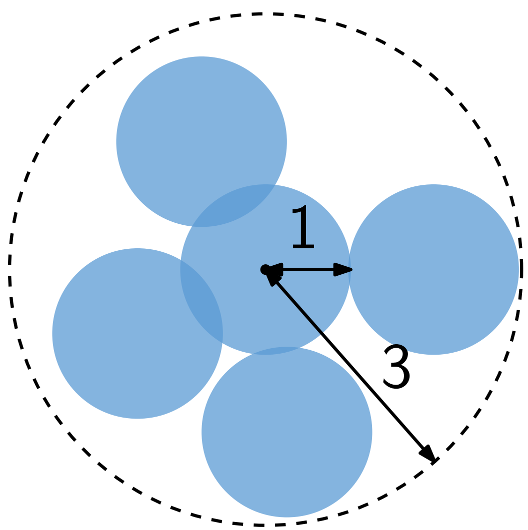
- is there an independent set of size at least k?
- are there k disjoint disks?

- a solution should take some space:
  if total area is $< k \cdot \pi$, then NO

k-Independent Set in unit disk graphs

- is there an independent set of size at least k?
- are there k disjoint disks?

- a solution should take some space:
  if total area is $< k \cdot \pi$, then NO
- large area implies that a greedy algorithm works:
  if total area is $\geq k \cdot 9 \cdot \pi$, then YES

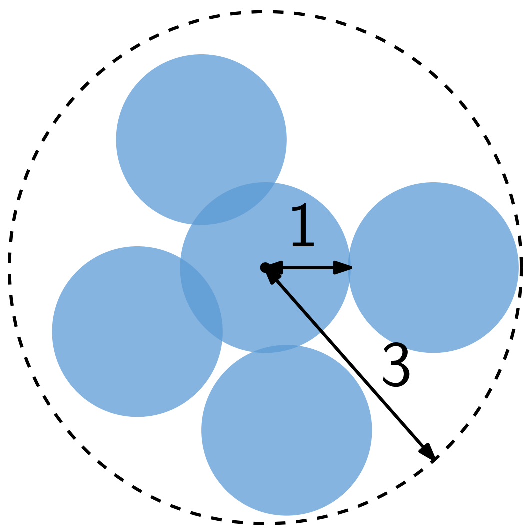


all disks intersecting the given one are contained in a disk of radius 3

k-Independent Set in unit disk graphs

- is there an independent set of size at least k?
- are there k disjoint disks?

- a solution should take some space:
  if total area is $< k \cdot \pi$, then NO
- large area implies that a greedy algorithm works:
  if total area is $\geq k \cdot 9 \cdot \pi$, then YES



all disks intersecting the given one are contained in a disk of radius 3

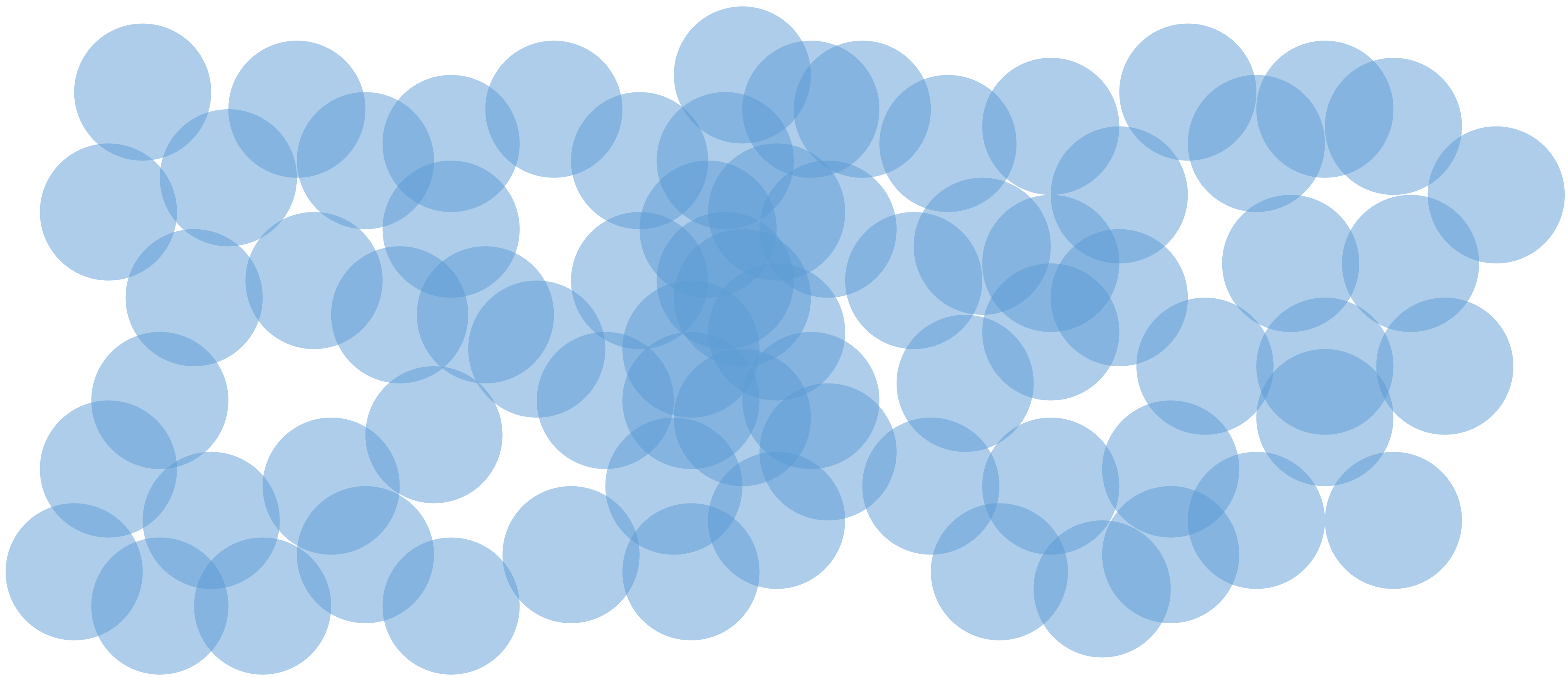
- assume that $\pi \cdot k \leq$ total area $\leq 9\pi \cdot k$

Geometric separator theorem for unit disks

Geometric separator theorem [Alber, Fiala, '04].

Given a collection of unit disks with total area A, there exists a set S of disks, such that:

- total area of disks in S is $\mathcal{O}(\sqrt{A})$,
- removing S gives connected parts of roughly equal area.

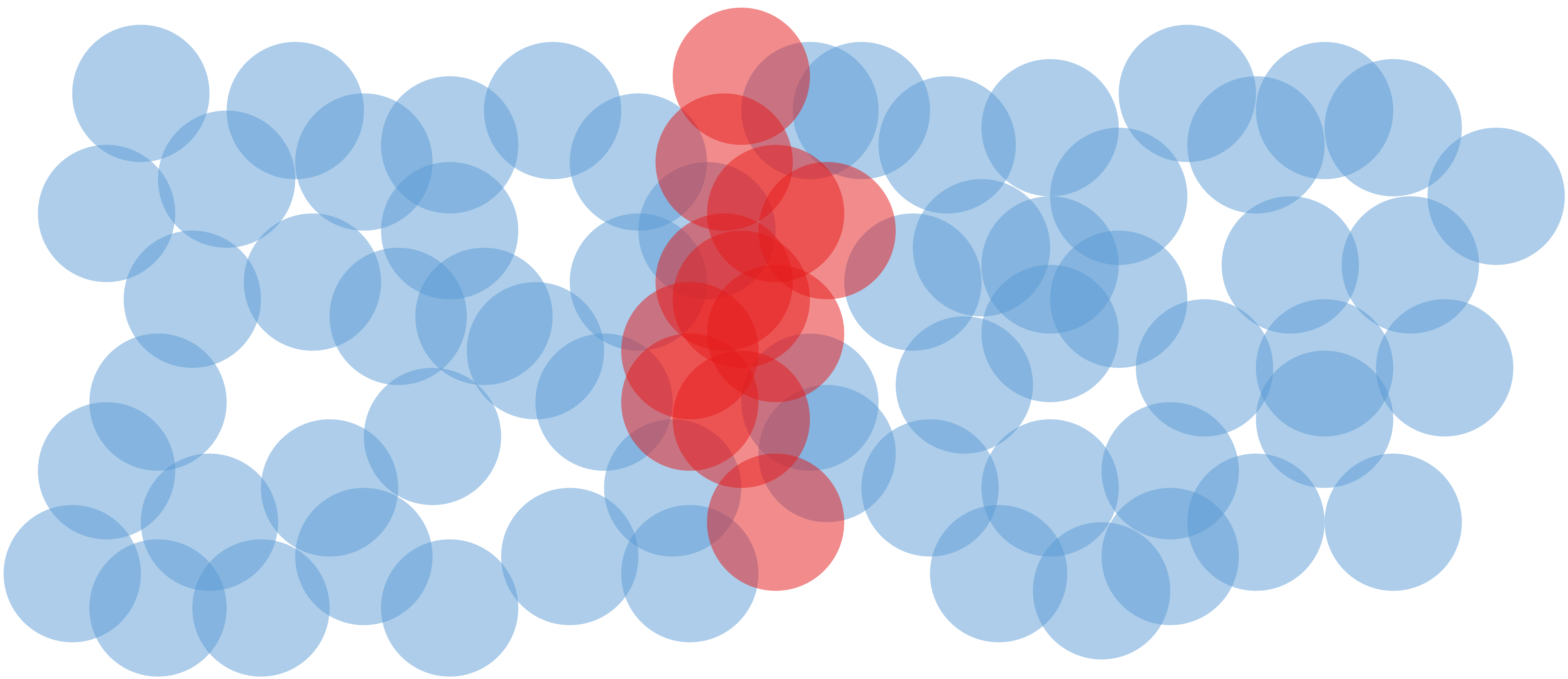


Geometric separator theorem for unit disks

Geometric separator theorem [Alber, Fiala, '04].

Given a collection of unit disks with total area A, there exists a set S of disks, such that:

- total area of disks in S is $\mathcal{O}(\sqrt{A})$,
- removing S gives connected parts of roughly equal area.

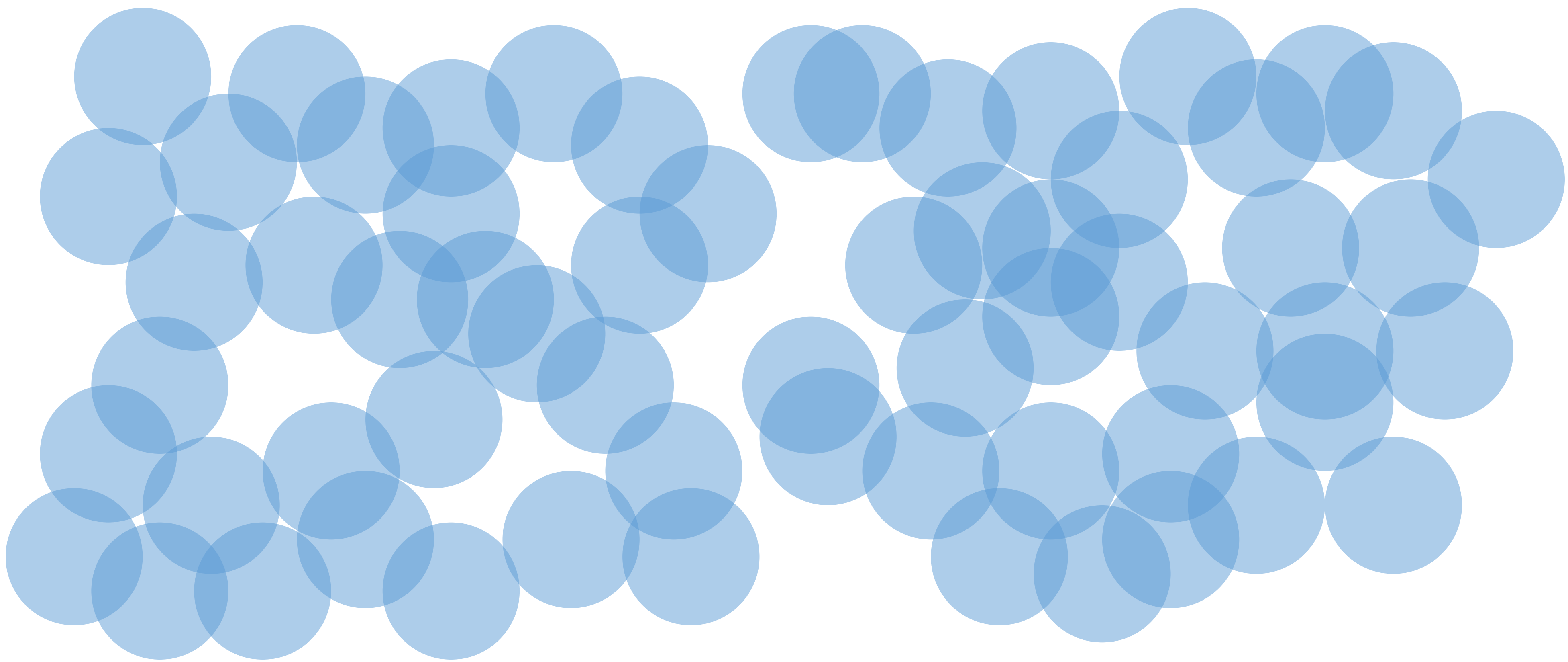


Geometric separator theorem for unit disks

Geometric separator theorem [Alber, Fiala, '04].

Given a collection of unit disks with total area A, there exists a set S of disks, such that:

- total area of disks in S is $\mathcal{O}(\sqrt{A})$,
- removing S gives connected parts of roughly equal area.



Divide & conquer using geometric separators

Algorithm [Alber, Fiala, '04].

1. $A =$ total area
2. if $A < \pi \cdot k$, return NO
3. if $A > 9\pi \cdot k$, return YES
4. find the geometric separator S of area $\mathcal{O}(\sqrt{A})$
5. guess the solution on S
6. remove S and recurse

Divide & conquer using geometric separators

Algorithm [Alber, Fiala, '04].

1. $A =$ total area
2. if $A < \pi \cdot k$, return NO
3. if $A > 9\pi \cdot k$, return YES
4. find the geometric separator S of area $\mathcal{O}(\sqrt{A})$
5. guess the solution on S
6. remove S and recurse

Divide & conquer using geometric separators

Algorithm [Alber, Fiala, '04].

1. $A =$ total area
2. if $A < \pi \cdot k$, return NO
3. if $A > 9\pi \cdot k$, return YES
4. find the geometric separator S of area $\mathcal{O}(\sqrt{A})$
5. guess the solution on S
6. remove S and recurse

- what is the largest possible independent set in S?

$$\text{area}(S)/\pi = \mathcal{O}(\sqrt{k})$$

Divide & conquer using geometric separators

Algorithm [Alber, Fiala, '04].

1. $A =$ total area
2. if $A < \pi \cdot k$, return NO
3. if $A > 9\pi \cdot k$, return YES
4. find the geometric separator S of area $\mathcal{O}(\sqrt{A})$
5. guess the solution on S
6. remove S and recurse

- what is the largest possible independent set in S?

$$\text{area}(S)/\pi = \mathcal{O}(\sqrt{k})$$

- what is the maximum number of independent sets in S?

$$\sum_{i=0}^{\mathcal{O}(\sqrt{k})} \binom{n}{i} = n^{\mathcal{O}(\sqrt{k})}$$

Divide & conquer using geometric separators

Algorithm [Alber, Fiala, '04].

1. $A =$ total area
2. if $A < \pi \cdot k$, return NO
3. if $A > 9\pi \cdot k$, return YES
4. find the geometric separator S of area $\mathcal{O}(\sqrt{A})$
5. guess the solution on S
6. remove S and recurse

- what is the largest possible independent set in S?

$$\text{area}(S)/\pi = \mathcal{O}(\sqrt{k})$$

- what is the maximum number of independent sets in S?

$$\sum_{i=0}^{\mathcal{O}(\sqrt{k})} \binom{n}{i} = n^{\mathcal{O}(\sqrt{k})}$$

- overall complexity is $n^{\mathcal{O}(\sqrt{k})}$

Evaluation

Strengths

- simple
- parameterized
- faster than what we had in the classical setting: $\sum_{k=1}^{n} n^{\mathcal{O}(\sqrt{k})} = 2^{\widetilde{\mathcal{O}}(\sqrt{n})}$, compared to $2^{\widetilde{\mathcal{O}}(n^{2/3})}$
- optimal (under ETH)
- works also for disks and other shapes with bounded area

Weaknesses

- doesn't work for general disk graphs, not to say about segment/string graphs
- necessarily requires a representation given

Evaluation

Strengths

- simple
- parameterized
- faster than what we had in the classical setting: $\sum_{k=1}^{n} n^{\mathcal{O}(\sqrt{k})} = 2^{\widetilde{\mathcal{O}}(\sqrt{n})}$, compared to $2^{\widetilde{\mathcal{O}}(n^{2/3})}$
- optimal (under ETH)
- works also for disks and other shapes with bounded area

Weaknesses

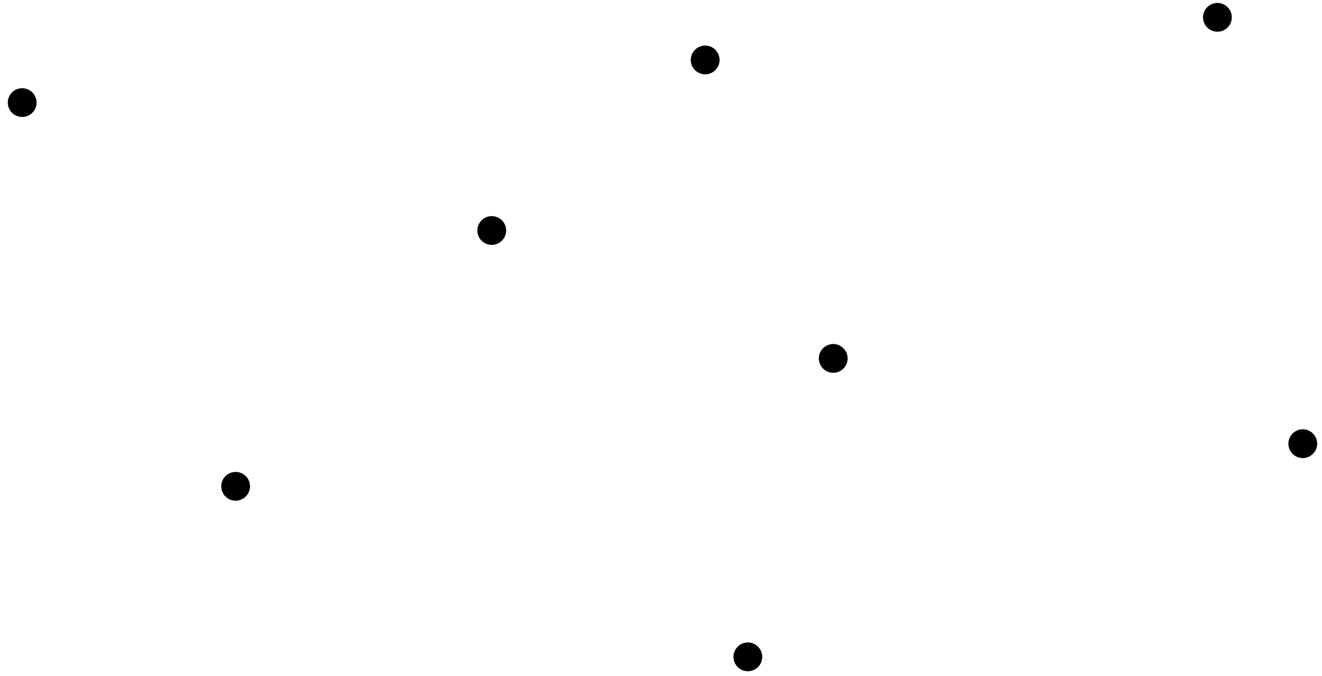
- doesn't work for general disk graphs, not to say about segment/string graphs
- necessarily requires a representation given

- in the remainder of this part we will learn how to address the first weakness, using a different approach

Voronoi-diagram approach

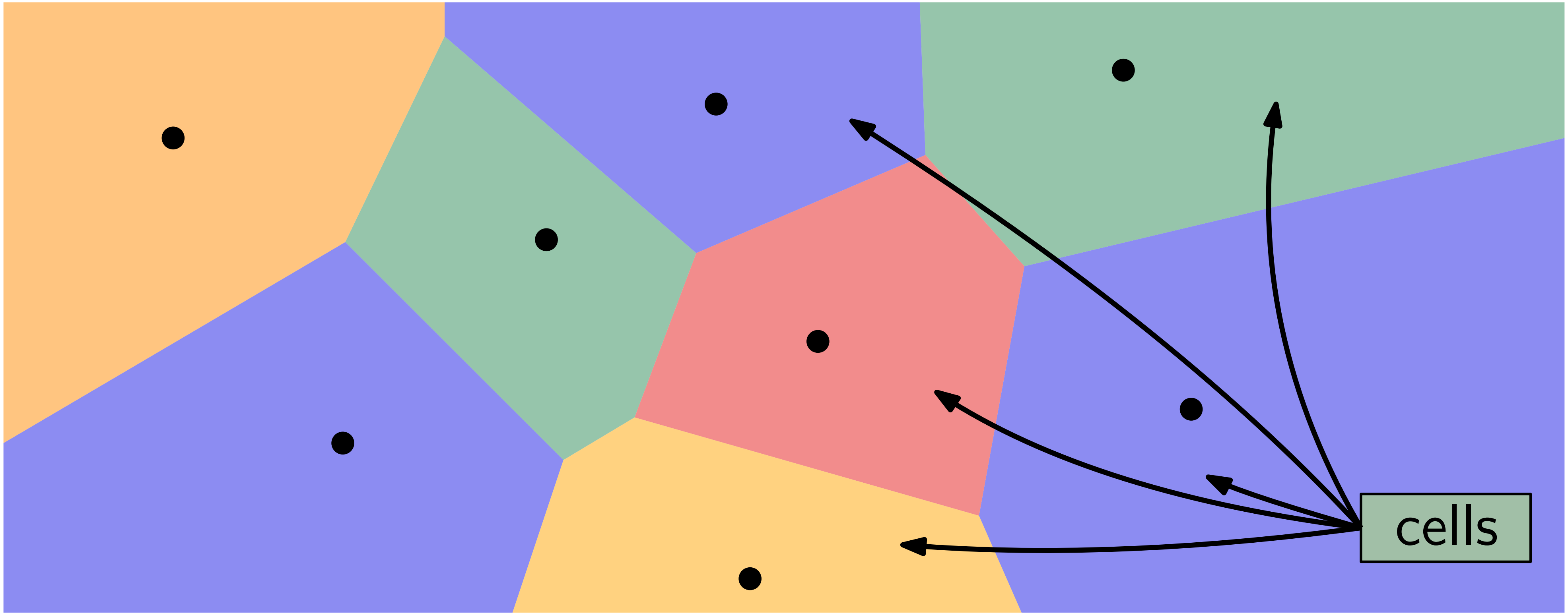
Voronoi diagrams

- we are given n points in the plane (objects)
- each point of the plane is assigned to the closest object



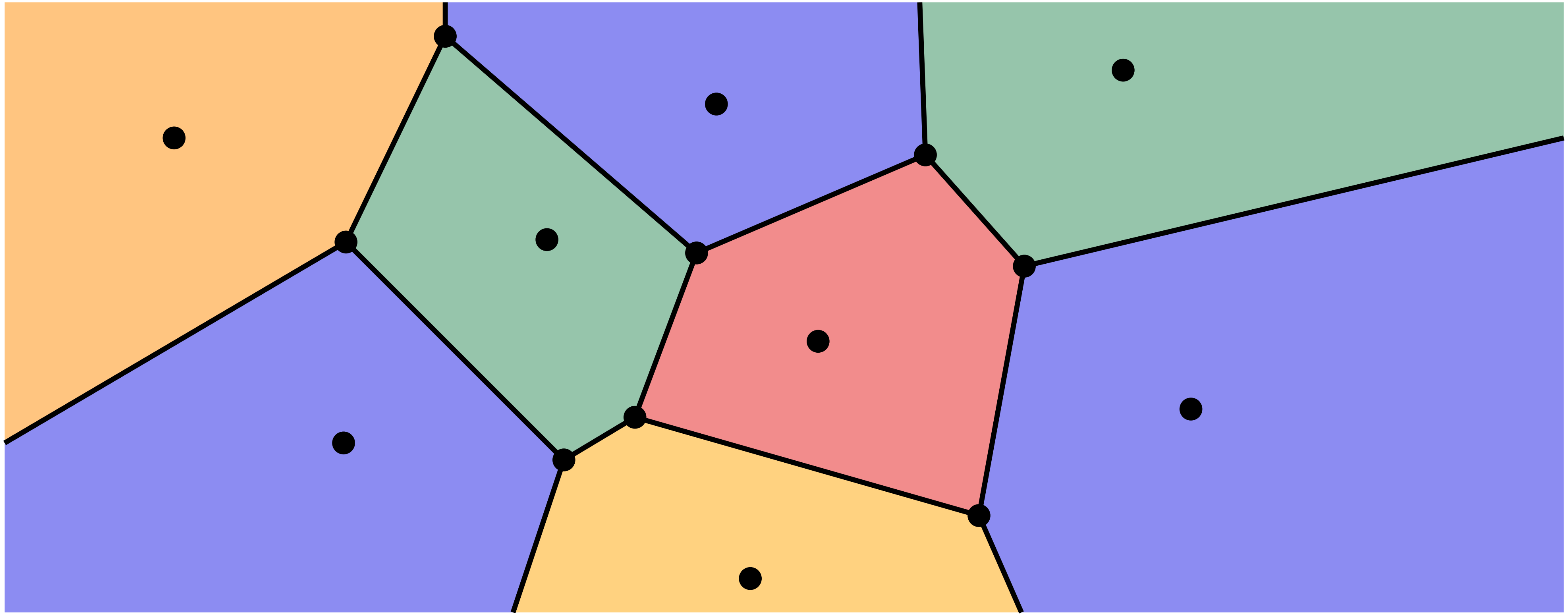
Voronoi diagrams

- we are given n points in the plane (objects)
- each point of the plane is assigned to the closest object



Voronoi diagrams

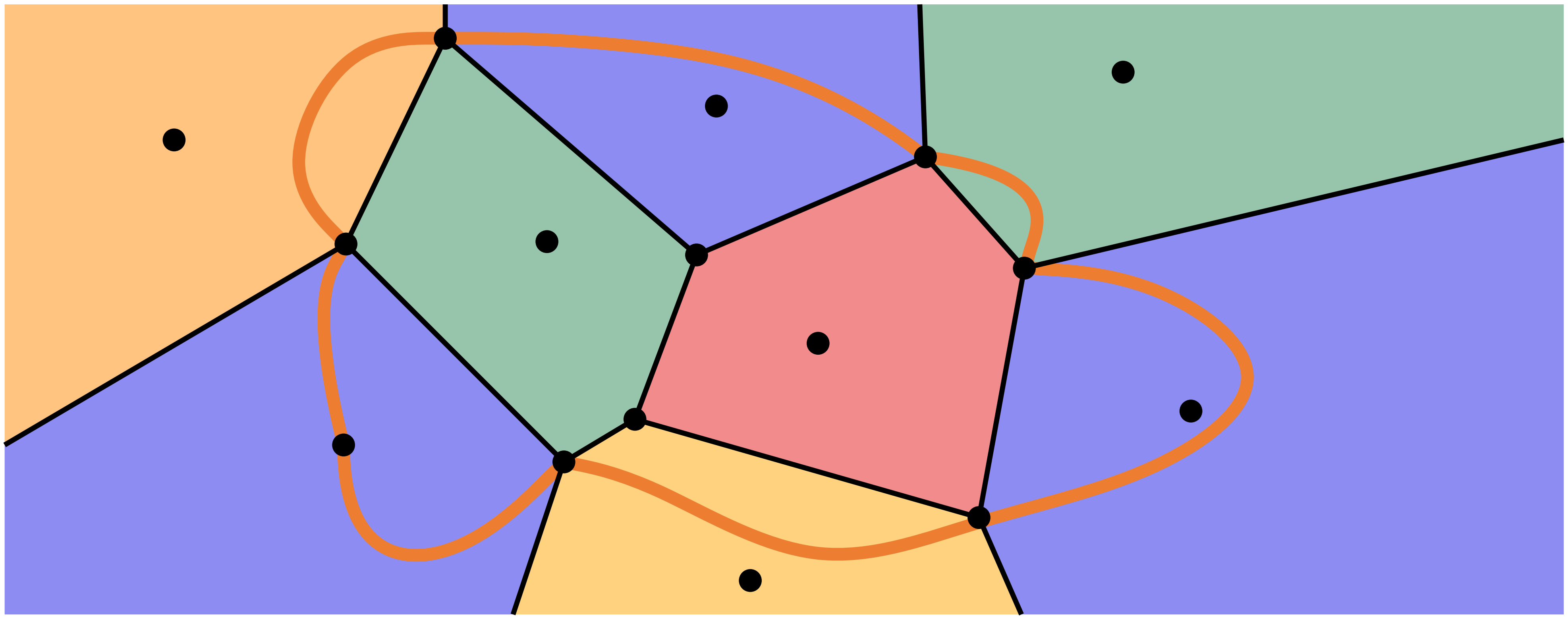
- we are given n points in the plane (objects)
- each point of the plane is assigned to the closest object



- it is (almost) a 3-regular 2-connected planar graph

Voronoi diagrams

- we are given n points in the plane (objects)
- each point of the plane is assigned to the closest object

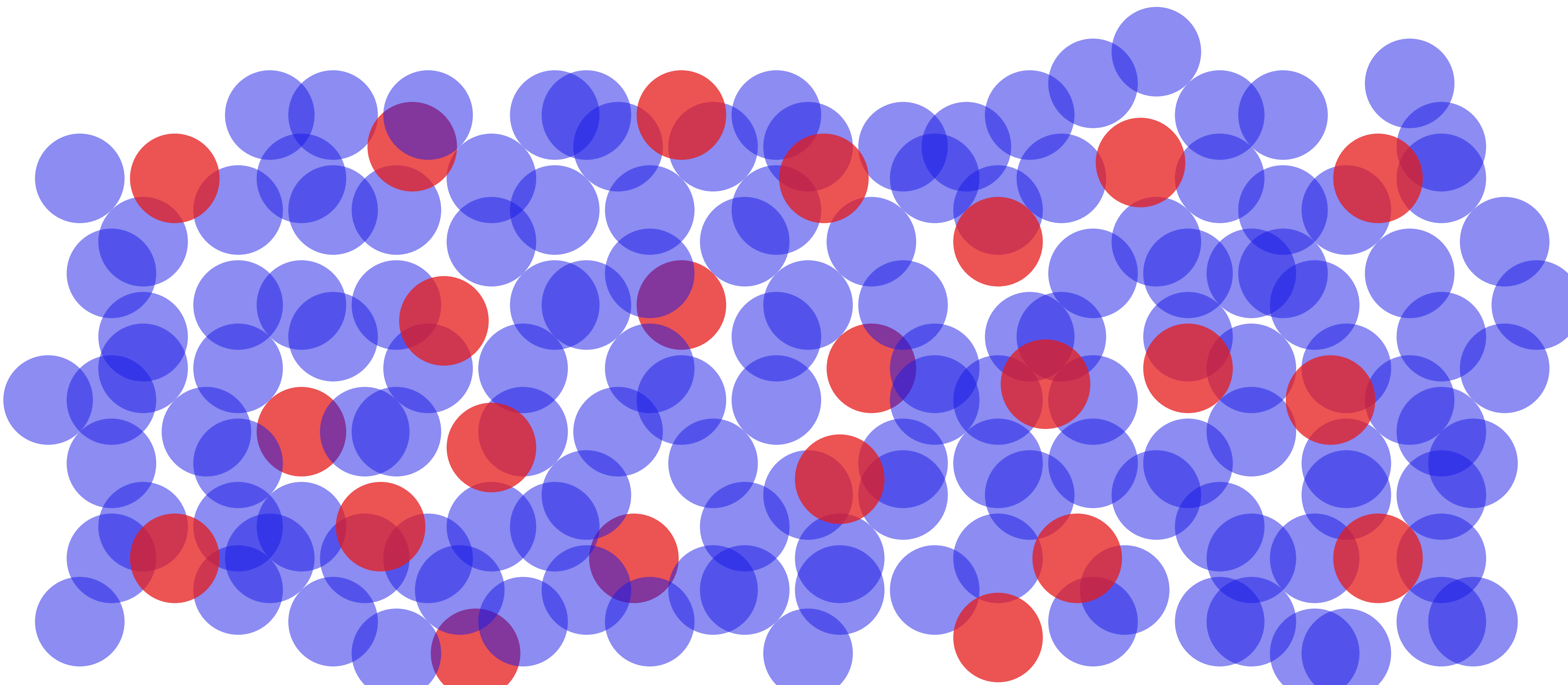


- it is (almost) a 3-regular 2-connected planar graph

Theorem [Marx, Pilipczuk '15]. Each graph like this has a balanced noose separator of size $\mathcal{O}(\sqrt{n})$.

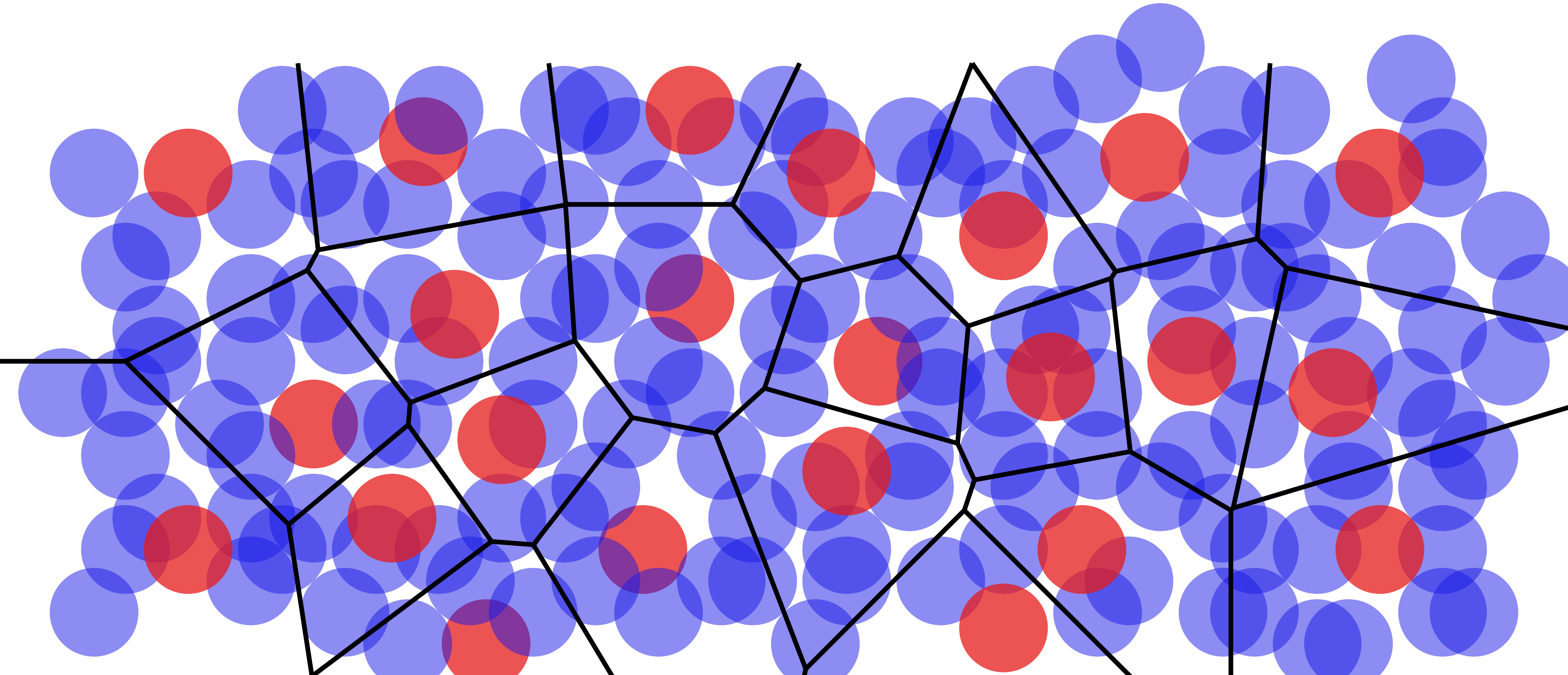
Solution Voronoi diagram

- consider a solution to the problem – k disjoint disks



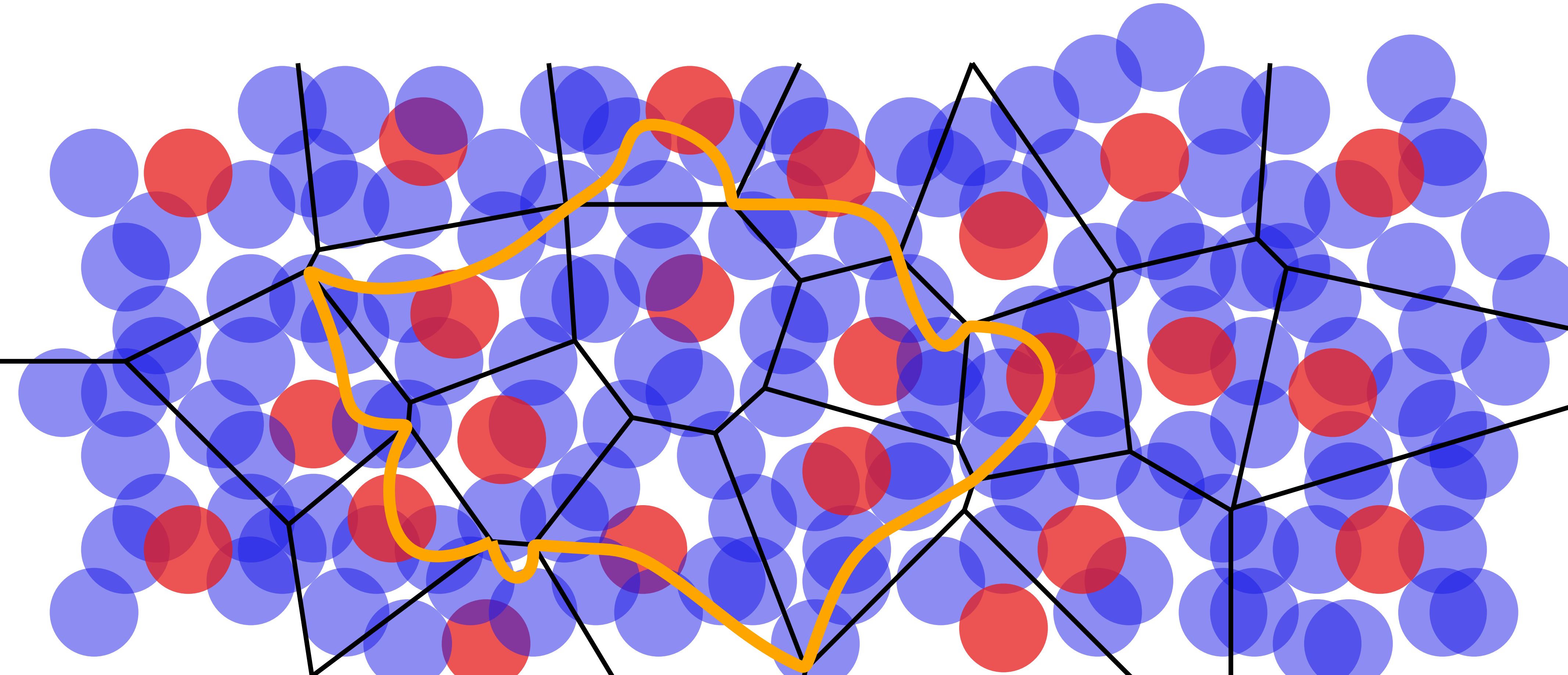
Solution Voronoi diagram

- consider a solution to the problem – k disjoint disks
- build the solution Voronoi diagram, where objects are centers of the disks in the solution



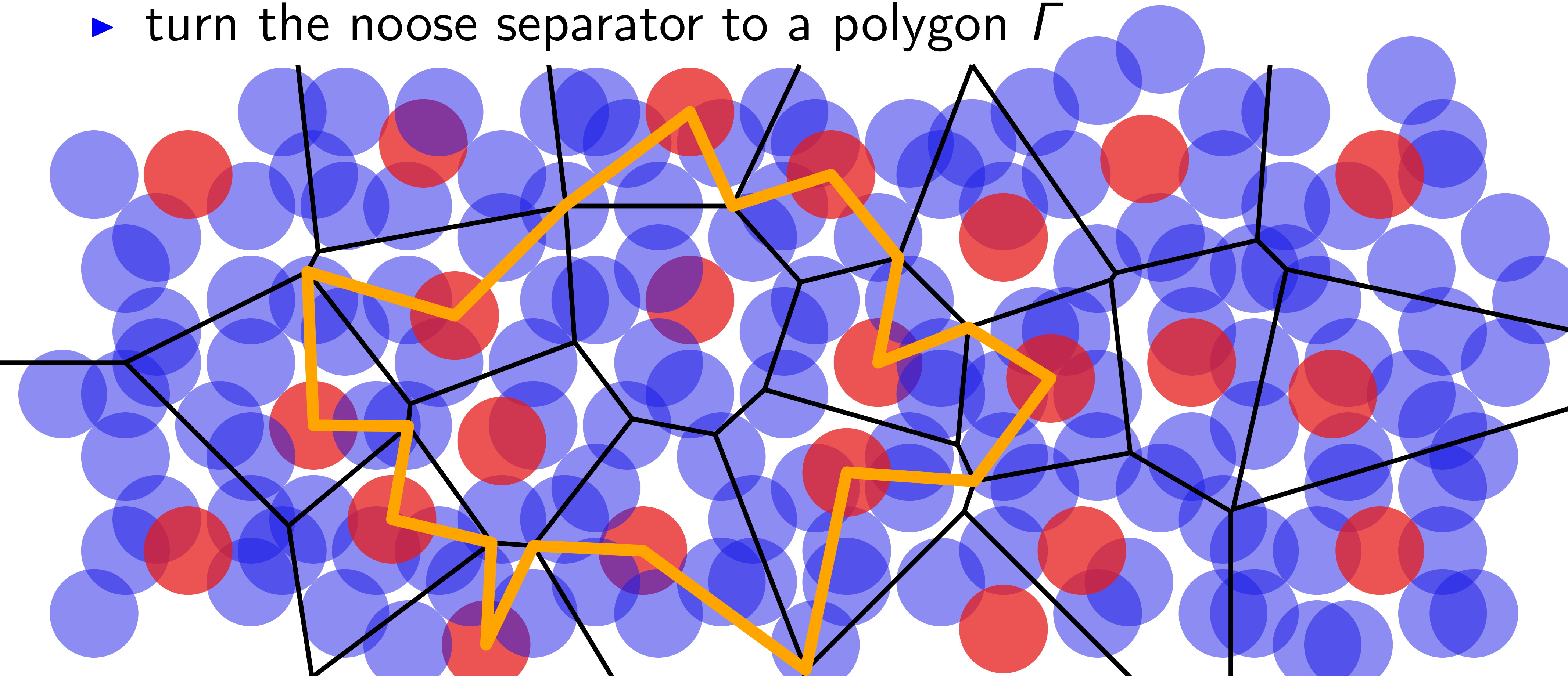
Solution Voronoi diagram

- consider a solution to the problem – k disjoint disks
- build the solution Voronoi diagram, where objects are centers of the disks in the solution
- there is a balanced noose separator, alternatingly visiting $\mathcal{O}(\sqrt{k})$ vertices and faces of the diagram



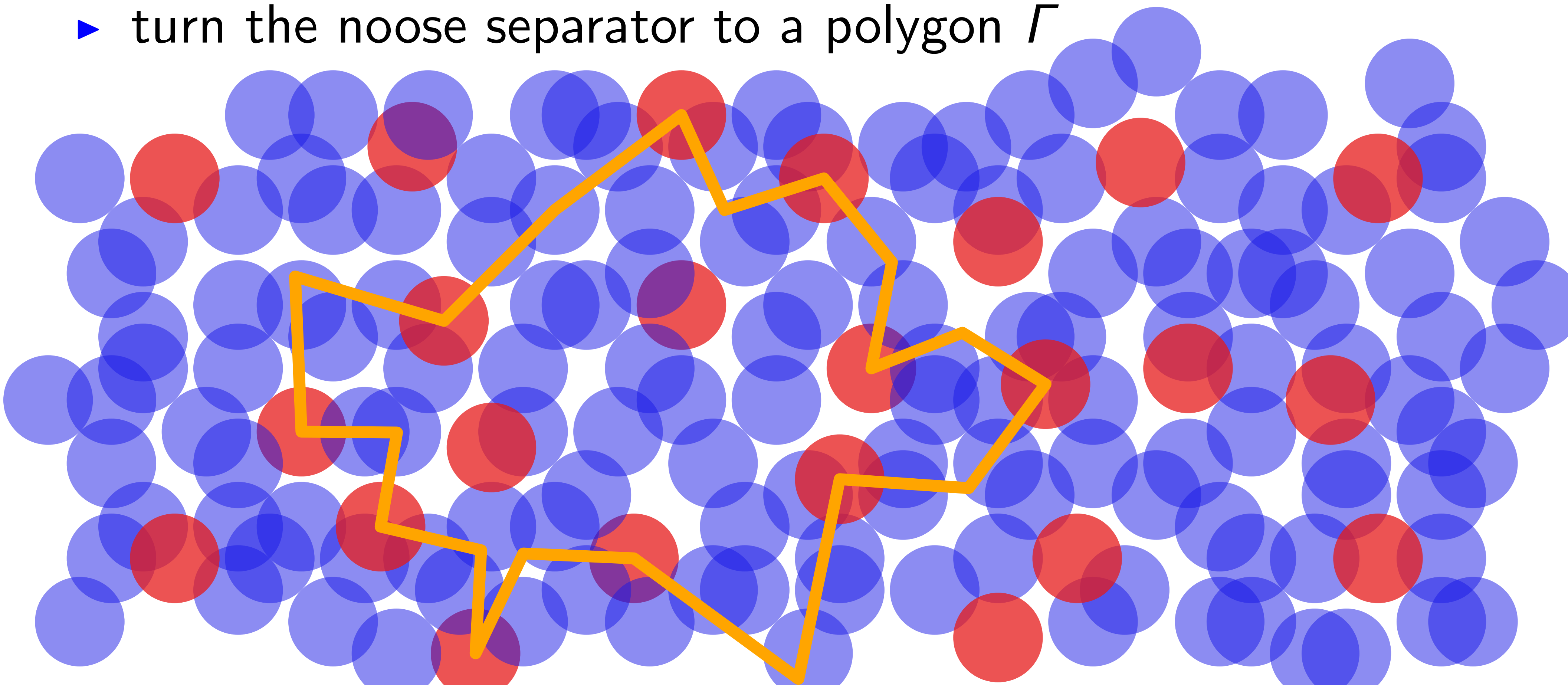
Solution Voronoi diagram

- consider a solution to the problem – k disjoint disks
- build the solution Voronoi diagram, where objects are centers of the disks in the solution
- there is a balanced noose separator, alternatingly visiting $\mathcal{O}(\sqrt{k})$ vertices and faces of the diagram
- turn the noose separator to a polygon Γ



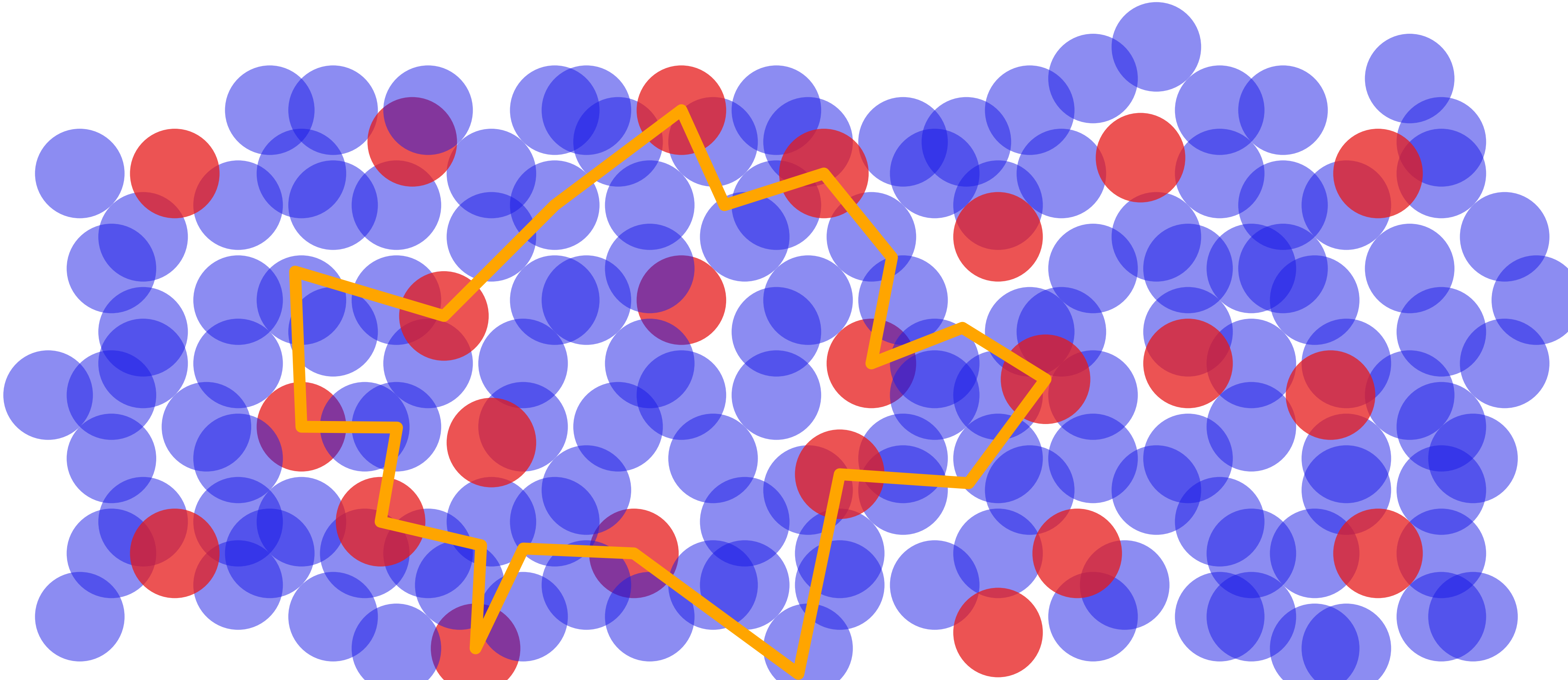
Solution Voronoi diagram

- consider a solution to the problem – k disjoint disks
- build the solution Voronoi diagram, where objects are centers of the disks in the solution
- there is a balanced noose separator, alternatingly visiting $\mathcal{O}(\sqrt{k})$ vertices and faces of the diagram
- turn the noose separator to a polygon Γ



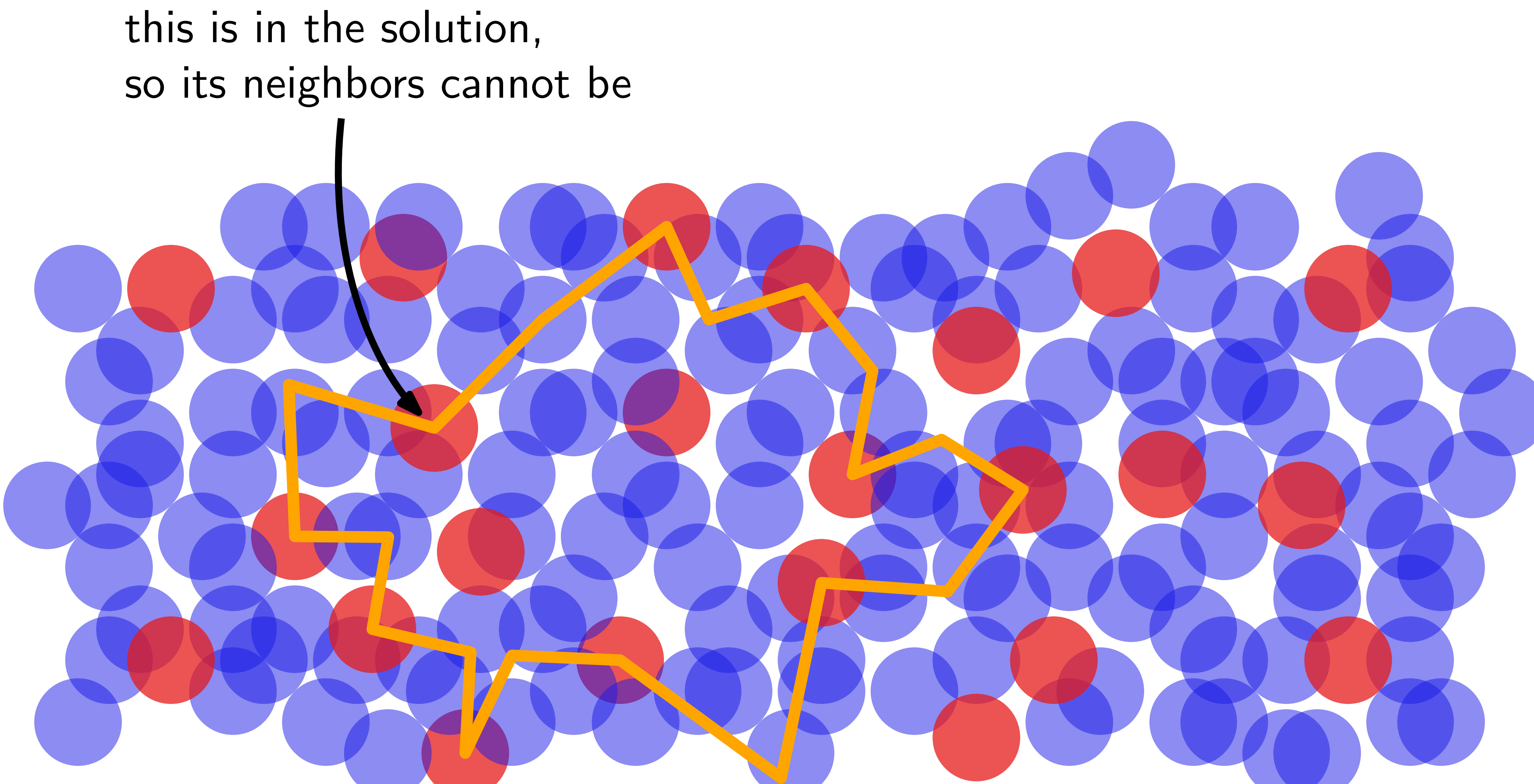
Separators in a solution Voronoi diagram

- every disk touching the outline of the polygon or any of the disks on its vertices can be discarded



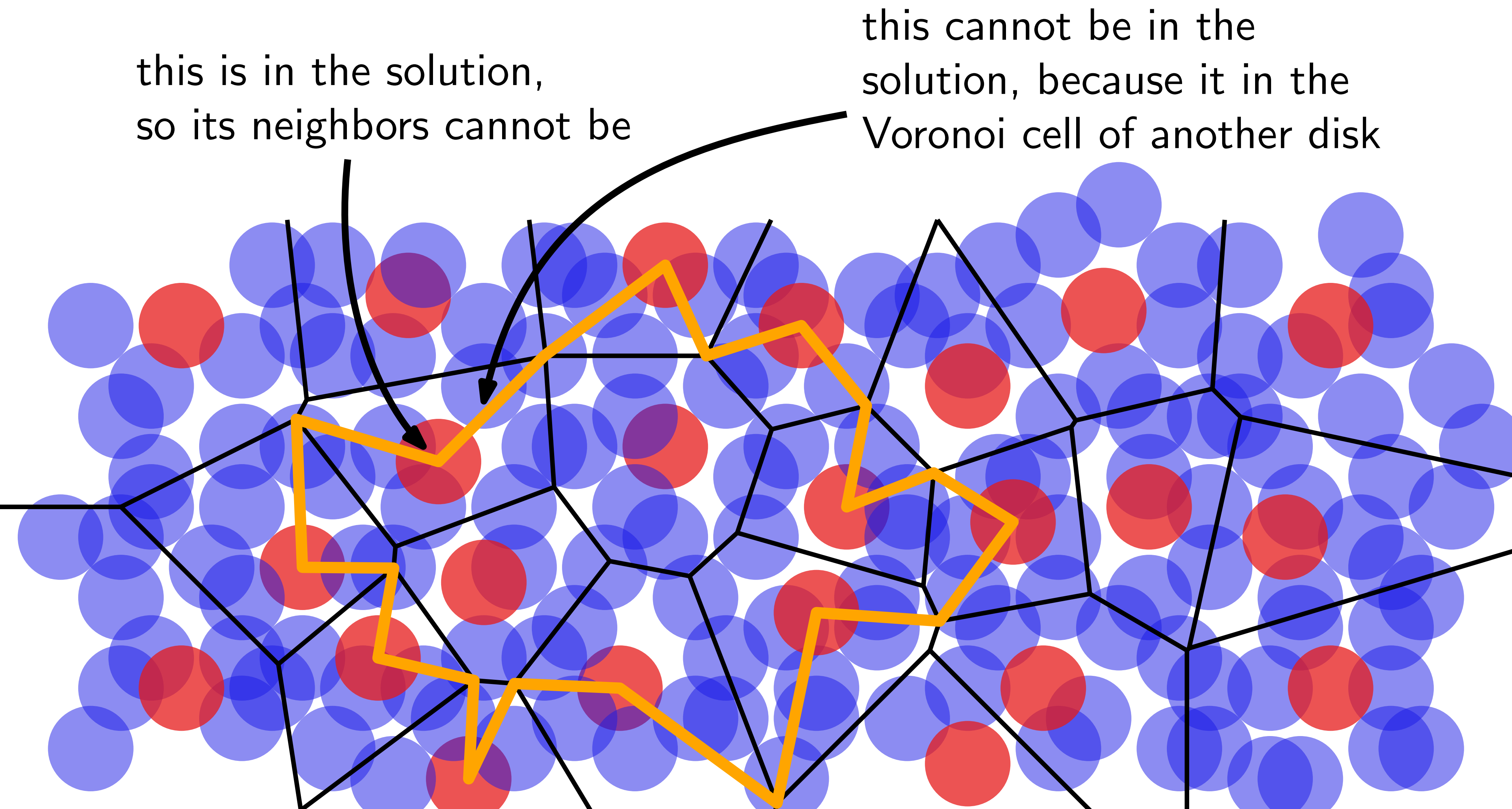
Separators in a solution Voronoi diagram

- every disk touching the outline of the polygon or any of the disks on its vertices can be discarded



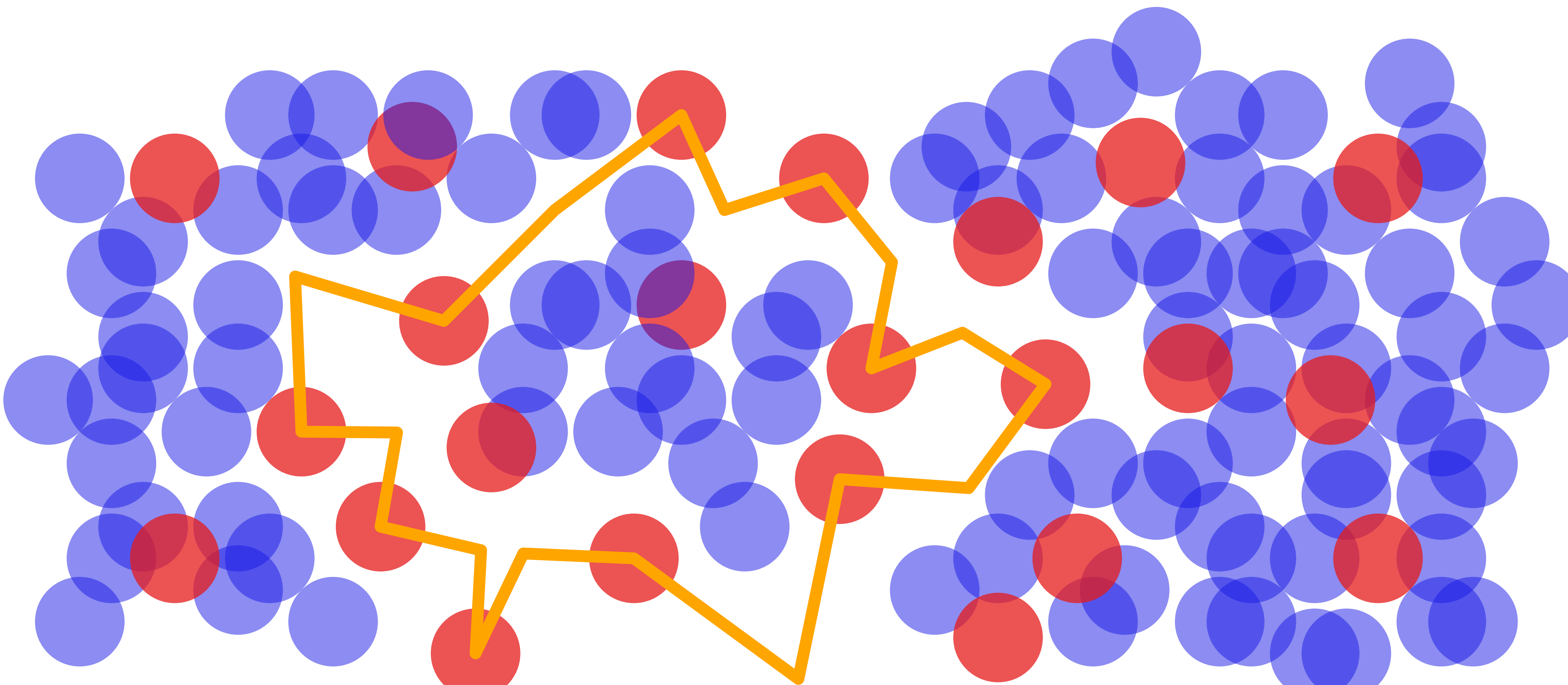
Separators in a solution Voronoi diagram

- every disk touching the outline of the polygon or any of the disks on its vertices can be discarded



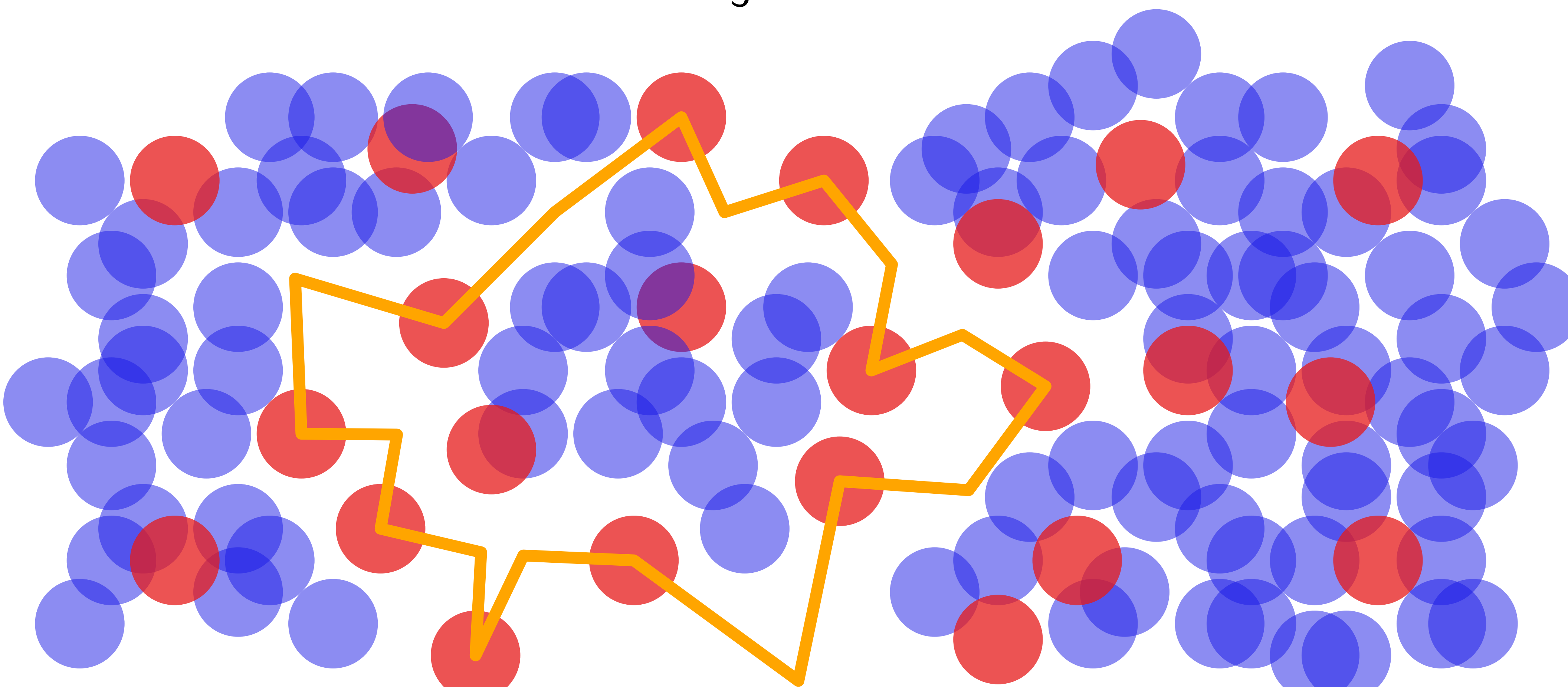
Separators in a solution Voronoi diagram

- every disk touching the outline of the polygon or any of the disks on its vertices can be discarded



Separators in a solution Voronoi diagram

- every disk touching the outline of the polygon or any of the disks on its vertices can be discarded
- apply recursion to disks inside and outside the polygon, we look for a solutions of size k_1, k_2, where $k_1 + k_2 = k$ and $k_1, k_2 \leq \frac{2}{3}k$



How to get a solution Voronoi diagram?

- but how can we know the solution Voronoi diagram?

How to get a solution Voronoi diagram?

- but how can we know the solution Voronoi diagram?
- we can't, but we can still guess the polygon separator Γ

How to get a solution Voronoi diagram?

- but how can we know the solution Voronoi diagram?
- we can't, but we can still guess the polygon separator Γ

- vertices of Γ are:
  - $\mathcal{O}(\sqrt{k})$ centers of disks
  - $\mathcal{O}(\sqrt{k})$ vertices the Voronoi diagram
    $\rightarrow$ each of them is uniquely defined by 3 centers

How to get a solution Voronoi diagram?

- but how can we know the solution Voronoi diagram?
- we can't, but we can still guess the polygon separator Γ

- vertices of Γ are:
  - $\mathcal{O}(\sqrt{k})$ centers of disks
  - $\mathcal{O}(\sqrt{k})$ vertices the Voronoi diagram
    $\rightarrow$ each of them is uniquely defined by 3 centers
- so in order to guess Γ we need to guess $\mathcal{O}(\sqrt{k})$ disks

this requires time $n^{\mathcal{O}(\sqrt{k})}$

How to get a solution Voronoi diagram?

- but how can we know the solution Voronoi diagram?
- we can't, but we can still guess the polygon separator Γ

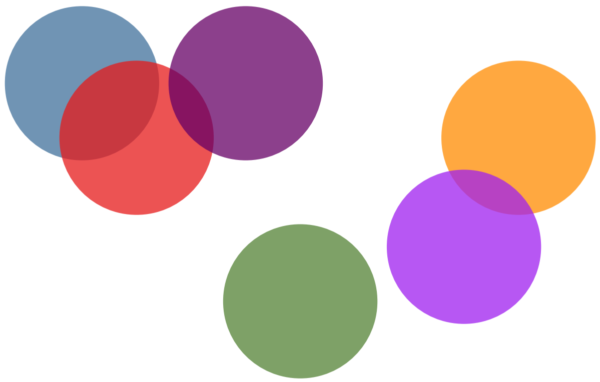
- vertices of Γ are:
  - $\mathcal{O}(\sqrt{k})$ centers of disks
  - $\mathcal{O}(\sqrt{k})$ vertices the Voronoi diagram
    $\rightarrow$ each of them is uniquely defined by 3 centers
- so in order to guess Γ we need to guess $\mathcal{O}(\sqrt{k})$ disks

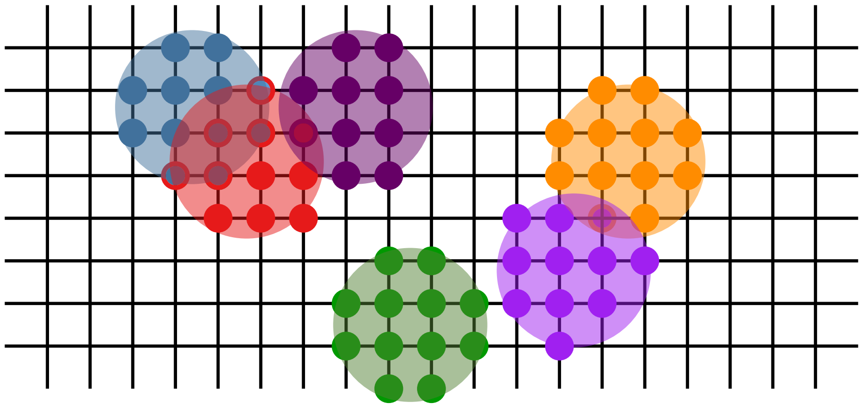
this requires time $n^{\mathcal{O}(\sqrt{k})}$

$$T(n,k) \leq n^{\mathcal{O}(\sqrt{k})} \cdot k^2 \cdot 2T(n, \tfrac{2}{3}k) = n^{\mathcal{O}(\sqrt{k})}$$

From disks to other geometric objects

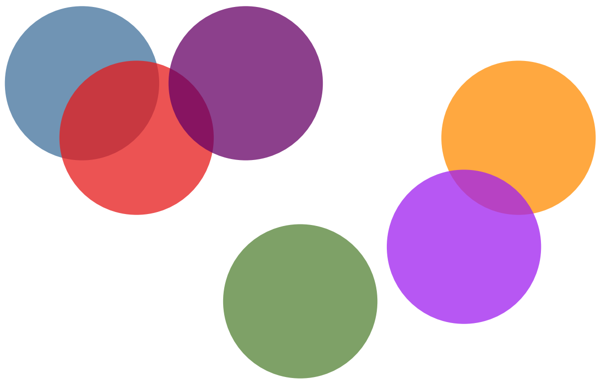
- disks can be seen as connected subgraphs of a fine grid

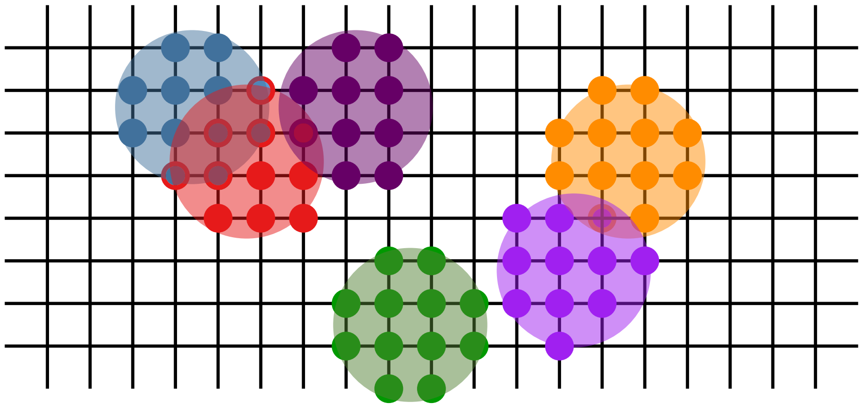




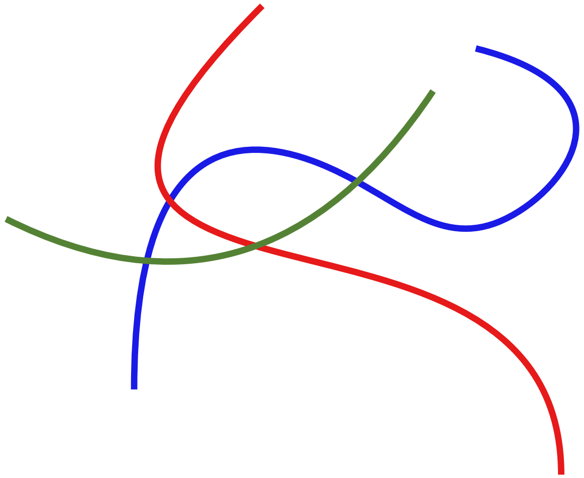
From disks to other geometric objects

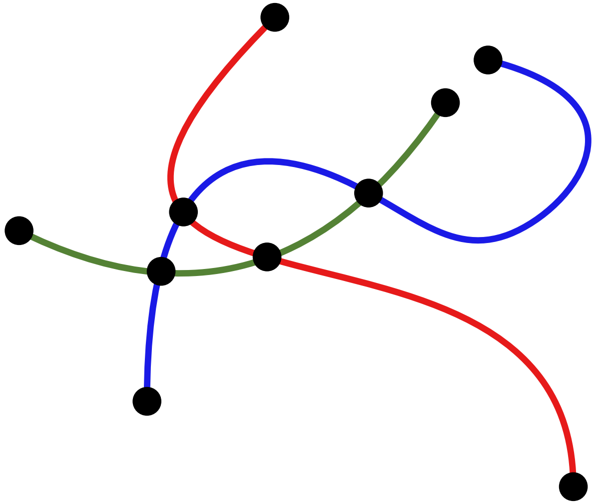
- disks can be seen as connected subgraphs of a fine grid





- string graphs = intersection graphs of connected subgraphs of planar graphs





General statement

- the whole approach can be re-interpreted in terms of packing disjoint subgraphs of planar graphs

Theorem [Marx, Pilipczuk '15].
Given a planar graph G with r vertices and n connected subgraphs of G, in time $n^{\mathcal{O}(\sqrt{k})} \cdot \mathrm{poly}(r)$ we can decide if there is a collection of k disjoint subgraphs.

General statement

- the whole approach can be re-interpreted in terms of packing disjoint subgraphs of planar graphs

Theorem [Marx, Pilipczuk '15].
Given a planar graph G with r vertices and n connected subgraphs of G, in time $n^{\mathcal{O}(\sqrt{k})} \cdot \mathrm{poly}(r)$ we can decide if there is a collection of k disjoint subgraphs.

- no assumptions on area
- works for weighted variants
- to some extent works also for covering variant (domination)

- necessarily requires geometric represention
- r is the number of geometric vertices: for string graphs it might be exponential in n

- for disks and segments $r = \mathrm{poly}(n)$

General statement

- the whole approach can be re-interpreted in terms of packing disjoint subgraphs of planar graphs

Theorem [Marx, Pilipczuk '15].
Given a planar graph G with r vertices and n connected subgraphs of G, in time $n^{\mathcal{O}(\sqrt{k})} \cdot \text{poly}(r)$ we can decide if there is a collection of k disjoint subgraphs.

- no assumptions on area
- works for weighted variants
- to some extent works also for covering variant (domination)

- necessarily requires geometric represention
- r is the number of geometric vertices: for string graphs it might be exponential in n

- for disks and segments $r = \text{poly}(n)$
- Open question: For disk graphs, is there a robust algorithm for Independent Set with complexity $2^{o(k)}$ or $2^{\widetilde{\mathcal{O}}(\sqrt{n})}$?

Lower bounds
for parameterized algorithms

Parameterized lower bounds

- we know that k-Independent Set can be solved in time $n^{\mathcal{O}(\sqrt{k})}$ in disk graphs
- we aim to show that this is asymptotically optimal

Parameterized lower bounds

- we know that k-Independent Set can be solved in time $n^{\mathcal{O}(\sqrt{k})}$ in disk graphs
- we aim to show that this is asymptotically optimal

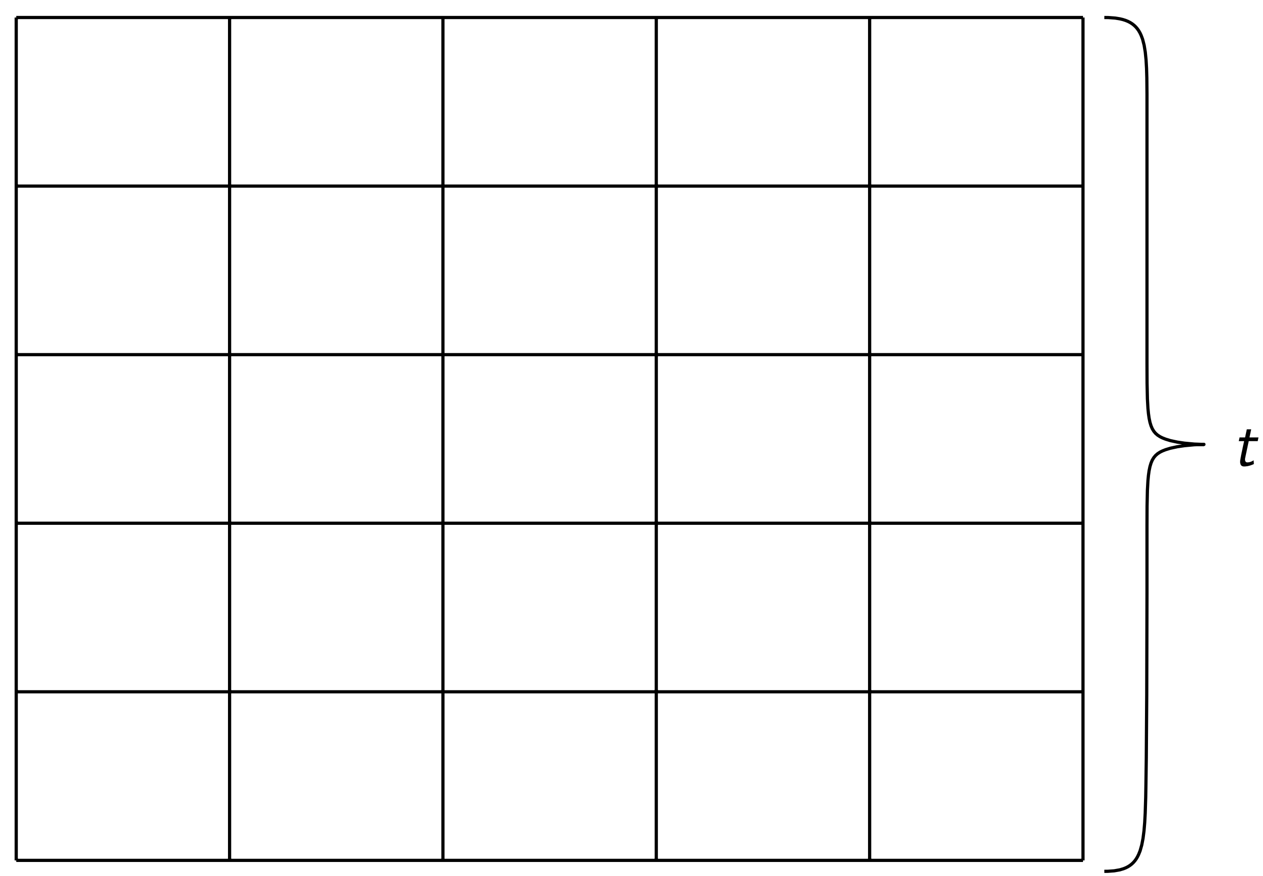
- we will need the following

Theorem.
Assuming the ETH, k-Clique cannot be solved in time $n^{o(k)}$.

- proof by a textbook reduction from 3-Sat

Grid Tiling

- we are given a square $t \times t$ grid



Grid Tiling

- we are given a square $t \times t$ grid
- in each cell (i, j) we have $S_{i,j} \subseteq [n] \times [n]$

(1,1)(1,2) (2,2)(2,3)	(1,1)(1,3) (1,4)(2,4) (3,1)	(1,4)(2,3) (2,4)(4,1)	(1,1)(1,4) (2,2)(2,3)	(1,1)(1,2) (2,2)(2,3)
(1,2)(1,3) (3,2)(4,1)	(2,1)(2,2) (3,3)(3,5)	(2,1)(2,3) (3,4)(3,5)	(2,5)(3,4) (4,1)(4,2)	(1,1)(1,2) (3,2)
(1,1)(1,2) (1,3)(1,4)	(1,1)(1,3) (2,4)(3,4)	(1,4)(2,1) (2,2)(2,3)	(1,2)(1,4) (3,1)(3,3)	(1,1)(1,2) (1,3)(2,2)
(1,2)(1,3) (2,2)(2,3)	(1,3)(2,1) (2,3)(2,4)	(2,1)(2,4) (3,1)(3,2)	(1,3)(2,3) (2,4)(4,1)	(1,4)(2,1) (2,2)(3,1)
(2,1)(3,1) (3,3)(4,2)	(2,2)(2,4) (4,3)(4,4)	(2,3)(3,2) (4,4)(4,5)	(1,3)(3,2) (3,4)(4,4)	(1,3)(3,3) (4,2)(4,3)

t

Grid Tiling

- we are given a square $t \times t$ grid
- in each cell (i,j) we have $S_{i,j} \subseteq [n] \times [n]$
- for each cell choose one pair, such that:
  - the first coordinates in each row are equal
  - the second coordinates in each column are equal

(1,1)**(1,2)** (2,2)(2,3)	(1,1)**(1,3)** (1,4)(2,4) (3,1)	**(1,4)**(2,3) (2,4)(4,1)	(1,1)**(1,4)** (2,2)(2,3)	(1,1)**(1,2)** (2,2)(2,3)
(1,2)(1,3) **(3,2)**(4,1)	(2,1)(2,2) **(3,3)**(3,5)	(2,1)(2,3) **(3,4)**(3,5)	(2,5)**(3,4)** (4,1)(4,2)	(1,1)(1,2) **(3,2)**
(1,1)**(1,2)** (1,3)(1,4)	(1,1)**(1,3)** (2,4)(3,4)	**(1,4)**(2,1) (2,2)(2,3)	(1,2)**(1,4)** (3,1)(3,3)	(1,1)**(1,2)** (1,3)(2,2)
(1,2)(1,3) **(2,2)**(2,3)	(1,3)(2,1) **(2,3)**(2,4)	(2,1)**(2,4)** (3,1)(3,2)	(1,3)(2,3) **(2,4)**(4,1)	(1,4)(2,1) **(2,2)**(3,1)
(2,1)(3,1) (3,3)**(4,2)**	(2,2)(2,4) **(4,3)**(4,4)	(2,3)(3,2) **(4,4)**(4,5)	(1,3)(3,2) (3,4)**(4,4)**	(1,3)(3,3) **(4,2)**(4,3)

t

Grid Tiling

- we are given a square $t \times t$ grid
- in each cell (i, j) we have $S_{i,j} \subseteq [n] \times [n]$
- for each cell choose one pair, such that:
  - the first coordinates in each row are equal
  - the second coordinates in each column are equal
- how fast can we solve it?

(1,1)**(1,2)** (2,2)(2,3)	(1,1)**(1,3)** (1,4)(2,4) (3,1)	**(1,4)**(2,3) (2,4)(4,1)	(1,1)**(1,4)** (2,2)(2,3)	(1,1)**(1,2)** (2,2)(2,3)
(1,2)(1,3) **(3,2)**(4,1)	(2,1)(2,2) **(3,3)**(3,5)	(2,1)(2,3) **(3,4)**(3,5)	(2,5)**(3,4)** (4,1)(4,2)	(1,1)(1,2) **(3,2)**
(1,1)**(1,2)** (1,3)(1,4)	(1,1)**(1,3)** (2,4)(3,4)	**(1,4)**(2,1) (2,2)(2,3)	(1,2)**(1,4)** (3,1)(3,3)	(1,1)**(1,2)** (1,3)(2,2)
(1,2)(1,3) **(2,2)**(2,3)	(1,3)(2,1) **(2,3)**(2,4)	(2,1)**(2,4)** (3,1)(3,2)	(1,3)(2,3) **(2,4)**(4,1)	(1,4)(2,1) **(2,2)**(3,1)
(2,1)(3,1) (3,3)**(4,2)**	(2,2)(2,4) **(4,3)**(4,4)	(2,3)(3,2) **(4,4)**(4,5)	(1,3)(3,2) (3,4)**(4,4)**	(1,3)(3,3) **(4,2)**(4,3)

t

Grid Tiling

- we are given a square $t \times t$ grid
- in each cell (i, j) we have $S_{i,j} \subseteq [n] \times [n]$
- for each cell choose one pair, such that:
  - the first coordinates in each row are equal
  - the second coordinates in each column are equal
- how fast can we solve it?
- guess everything: $(n^2)^{t^2} = n^{\mathcal{O}(t^2)}$

(1,1)**(1,2)** (2,2)(2,3)	(1,1)**(1,3)** (1,4)(2,4) (3,1)	**(1,4)**(2,3) (2,4)(4,1)	(1,1)**(1,4)** (2,2)(2,3)	(1,1)**(1,2)** (2,2)(2,3)
(1,2)(1,3) **(3,2)**(4,1)	(2,1)(2,2) **(3,3)**(3,5)	(2,1)(2,3) **(3,4)**(3,5)	(2,5)**(3,4)** (4,1)(4,2)	(1,1)(1,2) **(3,2)**
(1,1)**(1,2)** (1,3)(1,4)	(1,1)**(1,3)** (2,4)(3,4)	**(1,4)**(2,1) (2,2)(2,3)	(1,2)**(1,4)** (3,1)(3,3)	(1,1)**(1,2)** (1,3)(2,2)
(1,2)(1,3) **(2,2)**(2,3)	(1,3)(2,1) **(2,3)**(2,4)	(2,1)**(2,4)** (3,1)(3,2)	(1,3)(2,3) **(2,4)**(4,1)	(1,4)(2,1) **(2,2)**(3,1)
(2,1)(3,1) (3,3)**(4,2)**	(2,2)(2,4) **(4,3)**(4,4)	(2,3)(3,2) **(4,4)**(4,5)	(1,3)(3,2) (3,4)**(4,4)**	(1,3)(3,3) **(4,2)**(4,3)

t

Grid Tiling

- we are given a square $t \times t$ grid
- in each cell (i, j) we have $S_{i,j} \subseteq [n] \times [n]$
- for each cell choose one pair, such that:
  - the first coordinates in each row are equal
  - the second coordinates in each column are equal

- how fast can we solve it?
- guess everything: $(n^2)^{t^2} = n^{\mathcal{O}(t^2)}$
- guess the diagonal: $(n^2)^t = n^{\mathcal{O}(t)}$

(1,1)**(1,2)** (2,2)(2,3)	(1,1)**(1,3)** (1,4)(2,4) (3,1)	**(1,4)**(2,3) (2,4)(4,1)	(1,1)**(1,4)** (2,2)(2,3)	(1,1)**(1,2)** (2,2)(2,3)
(1,2)(1,3) **(3,2)**(4,1)	(2,1)(2,2) **(3,3)**(3,5)	(2,1)(2,3) **(3,4)**(3,5)	(2,5)**(3,4)** (4,1)(4,2)	(1,1)(1,2) **(3,2)**
(1,1)**(1,2)** (1,3)(1,4)	(1,1)**(1,3)** (2,4)(3,4)	**(1,4)**(2,1) (2,2)(2,3)	(1,2)**(1,4)** (3,1)(3,3)	(1,1)**(1,2)** (1,3)(2,2)
(1,2)(1,3) **(2,2)**(2,3)	(1,3)(2,1) **(2,3)**(2,4)	(2,1)**(2,4)** (3,1)(3,2)	(1,3)(2,3) **(2,4)**(4,1)	(1,4)(2,1) **(2,2)**(3,1)
(2,1)(3,1) (3,3)**(4,2)**	(2,2)(2,4) **(4,3)**(4,4)	(2,3)(3,2) **(4,4)**(4,5)	(1,3)(3,2) (3,4)**(4,4)**	(1,3)(3,3) **(4,2)**(4,3)

t

Grid Tiling

- we are given a square $t \times t$ grid
- in each cell (i,j) we have $S_{i,j} \subseteq [n] \times [n]$
- for each cell choose one pair, such that:
  - the first coordinates in each row are equal
  - the second coordinates in each column are equal

- how fast can we solve it?
- guess everything: $(n^2)^{t^2} = n^{\mathcal{O}(t^2)}$
- guess the diagonal: $(n^2)^t = n^{\mathcal{O}(t)}$
- we will show that this is optimal

(1,1)**(1,2)** (2,2)(2,3)	(1,1)**(1,3)** (1,4)(2,4) (3,1)	**(1,4)**(2,3) (2,4)(4,1)	(1,1)**(1,4)** (2,2)(2,3)	(1,1)**(1,2)** (2,2)(2,3)
(1,2)(1,3) **(3,2)**(4,1)	(2,1)(2,2) **(3,3)**(3,5)	(2,1)(2,3) **(3,4)**(3,5)	(2,5)**(3,4)** (4,1)(4,2)	(1,1)(1,2) **(3,2)**
(1,1)**(1,2)** (1,3)(1,4)	(1,1)**(1,3)** (2,4)(3,4)	**(1,4)**(2,1) (2,2)(2,3)	(1,2)**(1,4)** (3,1)(3,3)	(1,1)**(1,2)** (1,3)(2,2)
(1,2)(1,3) **(2,2)**(2,3)	(1,3)(2,1) **(2,3)**(2,4)	(2,1)**(2,4)** (3,1)(3,2)	(1,3)(2,3) **(2,4)**(4,1)	(1,4)(2,1) **(2,2)**(3,1)
(2,1)(3,1) (3,3)**(4,2)**	(2,2)(2,4) **(4,3)**(4,4)	(2,3)(3,2) **(4,4)**(4,5)	(1,3)(3,2) (3,4)**(4,4)**	(1,3)(3,3) **(4,2)**(4,3)

t

Hardness of GRID TILING

- $t \times t$ grid, each cell with some pairs from $[n] \times [n]$

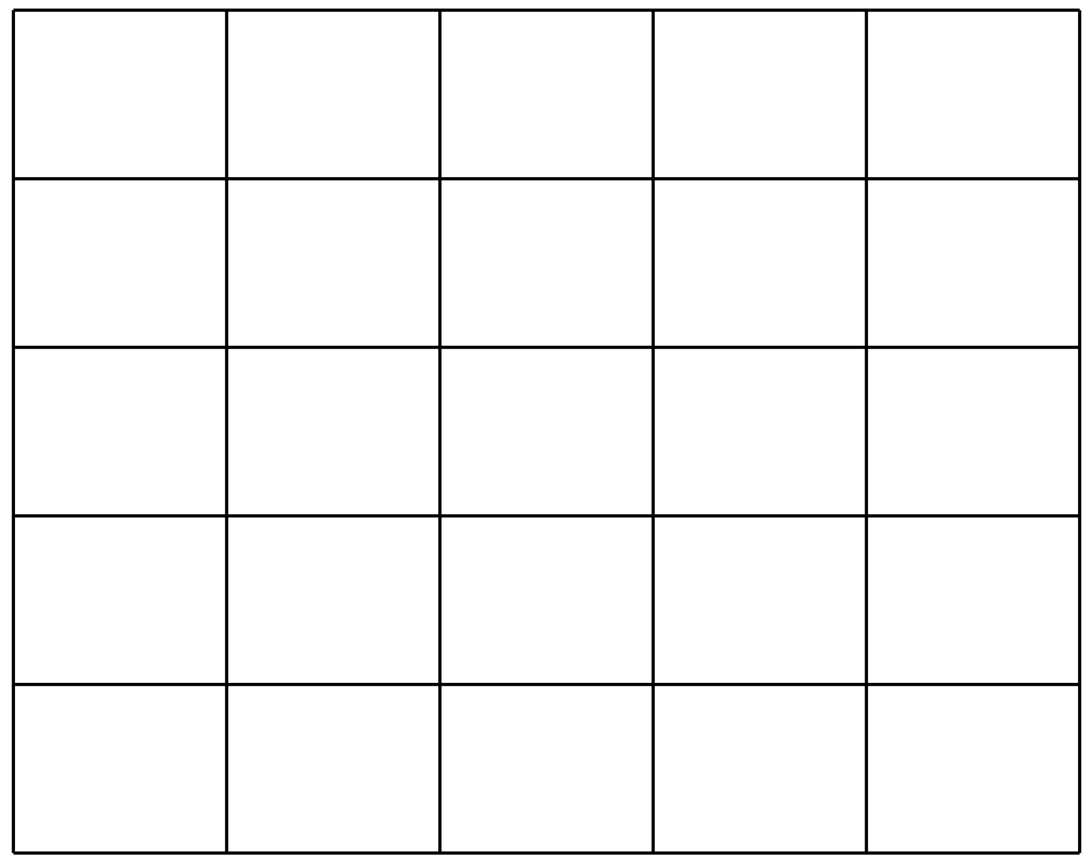
Theorem. GRID TILING cannot be solved in time $n^{o(t)}$, unless the ETH fails.

Hardness of Grid Tiling

- $t \times t$ grid, each cell with some pairs from $[n] \times [n]$

Theorem. Grid Tiling cannot be solved in time $n^{o(t)}$, unless the ETH fails.

- reduction from k-Clique with vertices $1, 2, \ldots, n$, $t = k$



Hardness of Grid Tiling

- $t \times t$ grid, each cell with some pairs from $[n] \times [n]$

Theorem. Grid Tiling cannot be solved in time $n^{o(t)}$, unless the ETH fails.

- reduction from k-Clique with vertices $1, 2, \ldots, n$, $t = k$
- Sets for the cell (i, j):
  - $(x, y) \in S_{i,i}$ if $x = y$
  - $(x, y) \in S_{i,j}$ if $xy \in E$

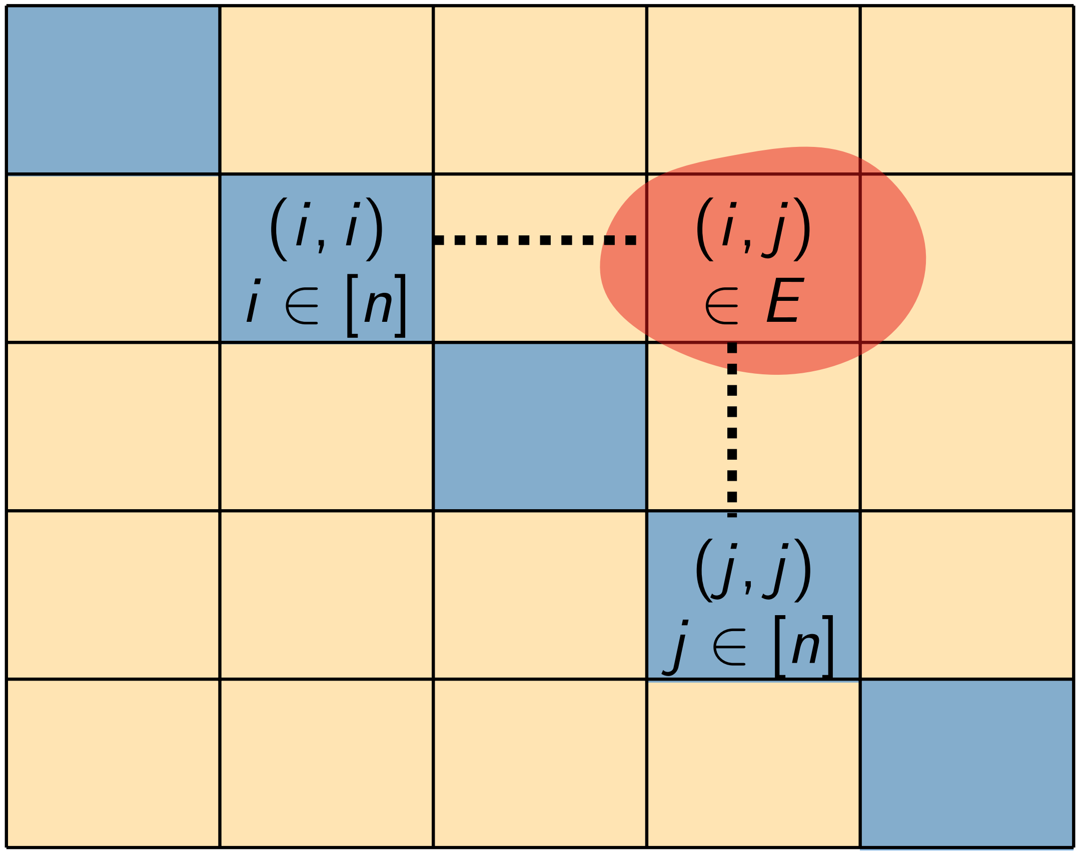
(i,i) $i \in [n]$	(i,j) $ij \in [n]$	(i,j) $ij \in [n]$	(i,j) $ij \in [n]$	(i,j) $ij \in [n]$
(i,j) $ij \in [n]$	(i,i) $i \in [n]$	(i,j) $ij \in [n]$	(i,j) $ij \in [n]$	(i,j) $ij \in [n]$
(i,j) $ij \in [n]$	(i,j) $ij \in [n]$	(i,i) $i \in [n]$	(i,j) $ij \in [n]$	(i,j) $ij \in [n]$
(i,j) $ij \in [n]$	(i,j) $ij \in [n]$	(i,j) $ij \in [n]$	(i,i) $i \in [n]$	(i,j) $ij \in [n]$
(i,j) $ij \in [n]$	(i,j) $ij \in [n]$	(i,j) $ij \in [n]$	(i,j) $ij \in [n]$	(i,i) $i \in [n]$

Hardness of Grid Tiling

- $t \times t$ grid, each cell with some pairs from $[n] \times [n]$

Theorem. Grid Tiling cannot be solved in time $n^{o(t)}$, unless the ETH fails.

- reduction from k-Clique with vertices $1, 2, \ldots, n$, $t = k$
- Sets for the cell (i, j):
  - $(x, y) \in S_{i,i}$ if $x = y$
  - $(x, y) \in S_{i,j}$ if $xy \in E$
- Selected pairs on the diagonal correspond to a clique

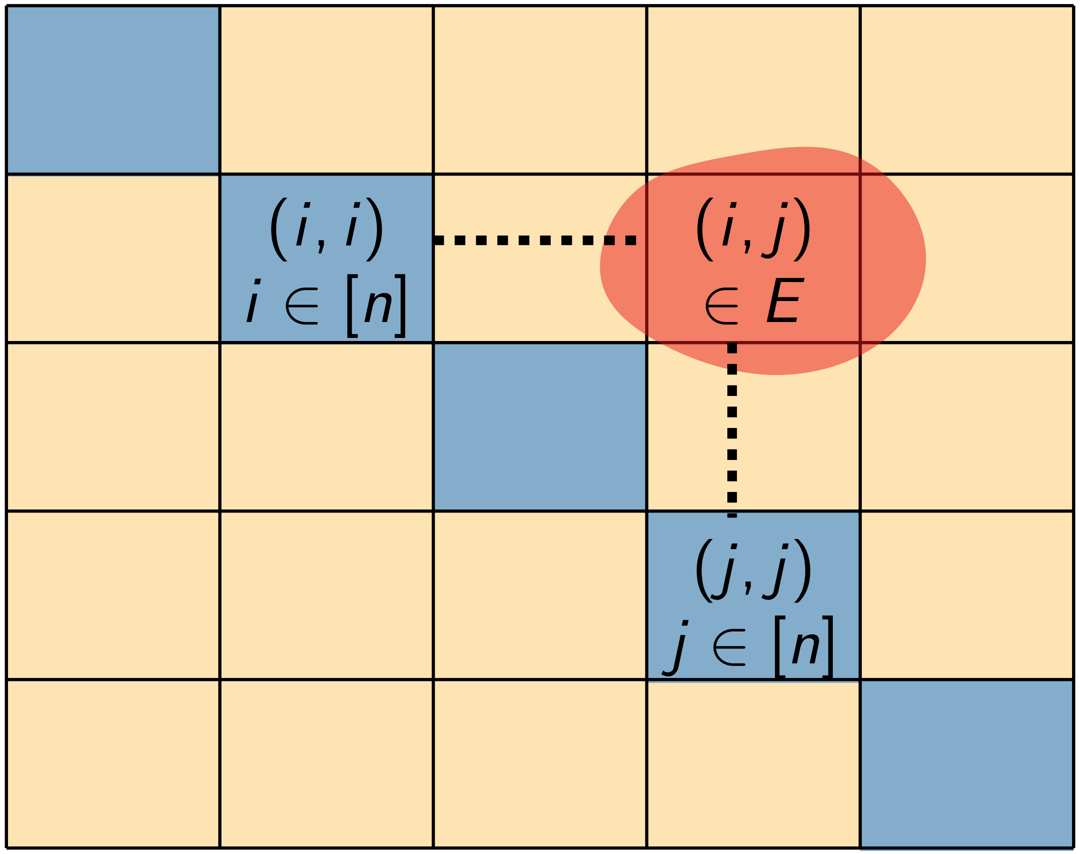


Hardness of GRID TILING

- $t \times t$ grid, each cell with some pairs from $[n] \times [n]$

Theorem. GRID TILING cannot be solved in time $n^{o(t)}$, unless the ETH fails.

- reduction from k-CLIQUE with vertices $1, 2, \ldots, n$, $t = k$
- Sets for the cell (i, j):
  - $(x, y) \in S_{i,i}$ if $x = y$
  - $(x, y) \in S_{i,j}$ if $xy \in E$
- Selected pairs on the diagonal correspond to a clique
- solving GRID TILING in time $n^{o(t)} \rightarrow$ solving k-CLIQUE in time $n^{o(k)}$



Grid Tiling

- we are given a square $t \times t$ grid
- in each cell (i, j) we have $S_{i,j} \subseteq [n] \times [n]$
- for each cell choose one pair, such that:
  - the first coordinates in each row are equal
  - the second coodrinates in each column are equal

Theorem. Assuming the ETH, there is no algorithm solving Grid Tiling in time $n^{o(t)}$.

Grid Tiling with $\leq$

- we are given a square $t \times t$ grid
- in each cell (i, j) we have $S_{i,j} \subseteq [n] \times [n]$
- for each cell choose one pair, such that:
  - the first coordinates in each row are non-decreasing
  - the second coodrinates in each column are non-decreasing

Theorem. Assuming the ETH, there is no algorithm solving Grid Tiling with $\leq$ in time $n^{o(t)}$.

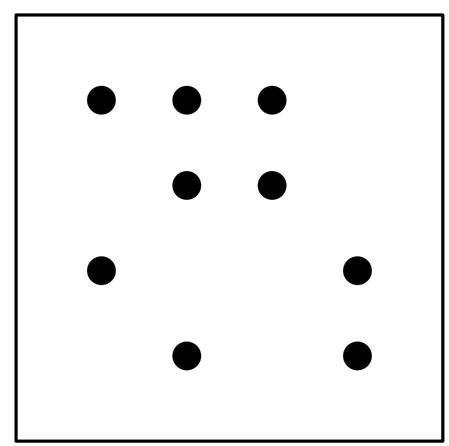
Grid Tiling with $\leq$

- we are given a square $t \times t$ grid
- in each cell (i, j) we have $S_{i,j} \subseteq [n] \times [n]$
- for each cell choose one pair, such that:
  - the first coordinates in each row are non-decreasing
  - the second coodrinates in each column are non-decreasing

Theorem. Assuming the ETH, there is no algorithm solving Grid Tiling with $\leq$ in time $n^{o(t)}$.

- each set $S_{i,j}$ can be seen as points of $n \times n$ grid

(1,1)(1,2)(1,3)
(2,2)(2,3)
(3,1)(3,4)
(4,2)(4,4)

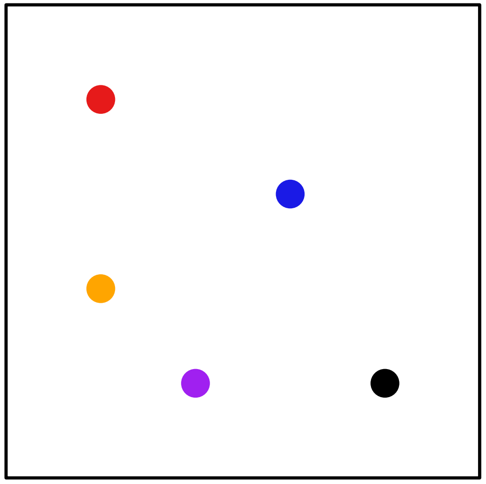


Hardness of Independent Set in UDGs

Theorem. Grid Tiling with $\leq$ cannot be solved in time $n^{o(t)}$, unless the ETH fails.

- $t \times t$ outer grid, $n \times n$ inner grids

a single cell:

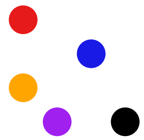


Hardness of Independent Set in UDGs

Theorem. Grid Tiling with $\leq$ cannot be solved in time $n^{o(t)}$, unless the ETH fails.

- $t \times t$ outer grid, $n \times n$ inner grids

a single cell:



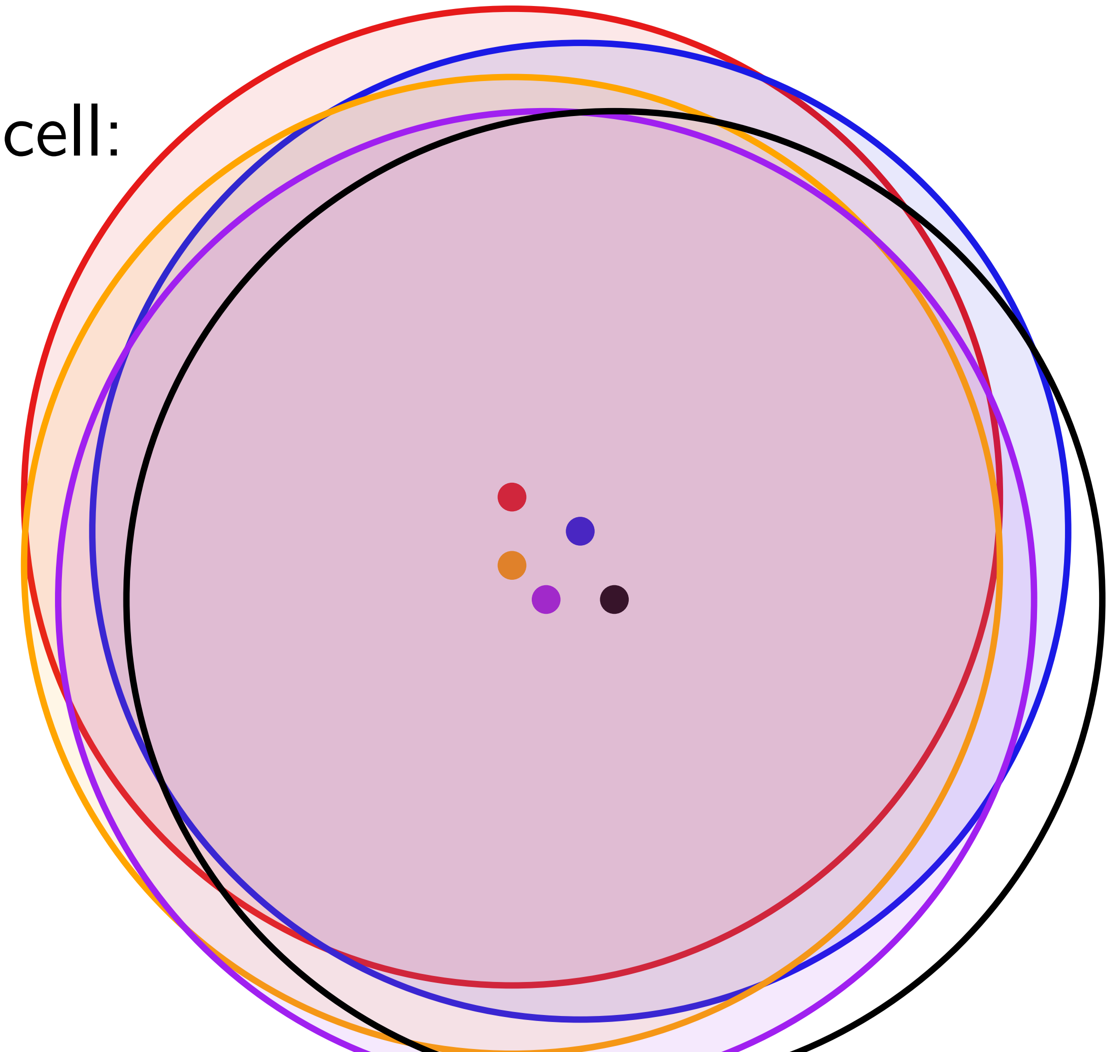
Hardness of Independent Set in UDGs

Theorem. Grid Tiling with $\leq$ cannot be solved in time $n^{o(t)}$, unless the ETH fails.

- $t \times t$ outer grid, $n \times n$ inner grids

a single cell:

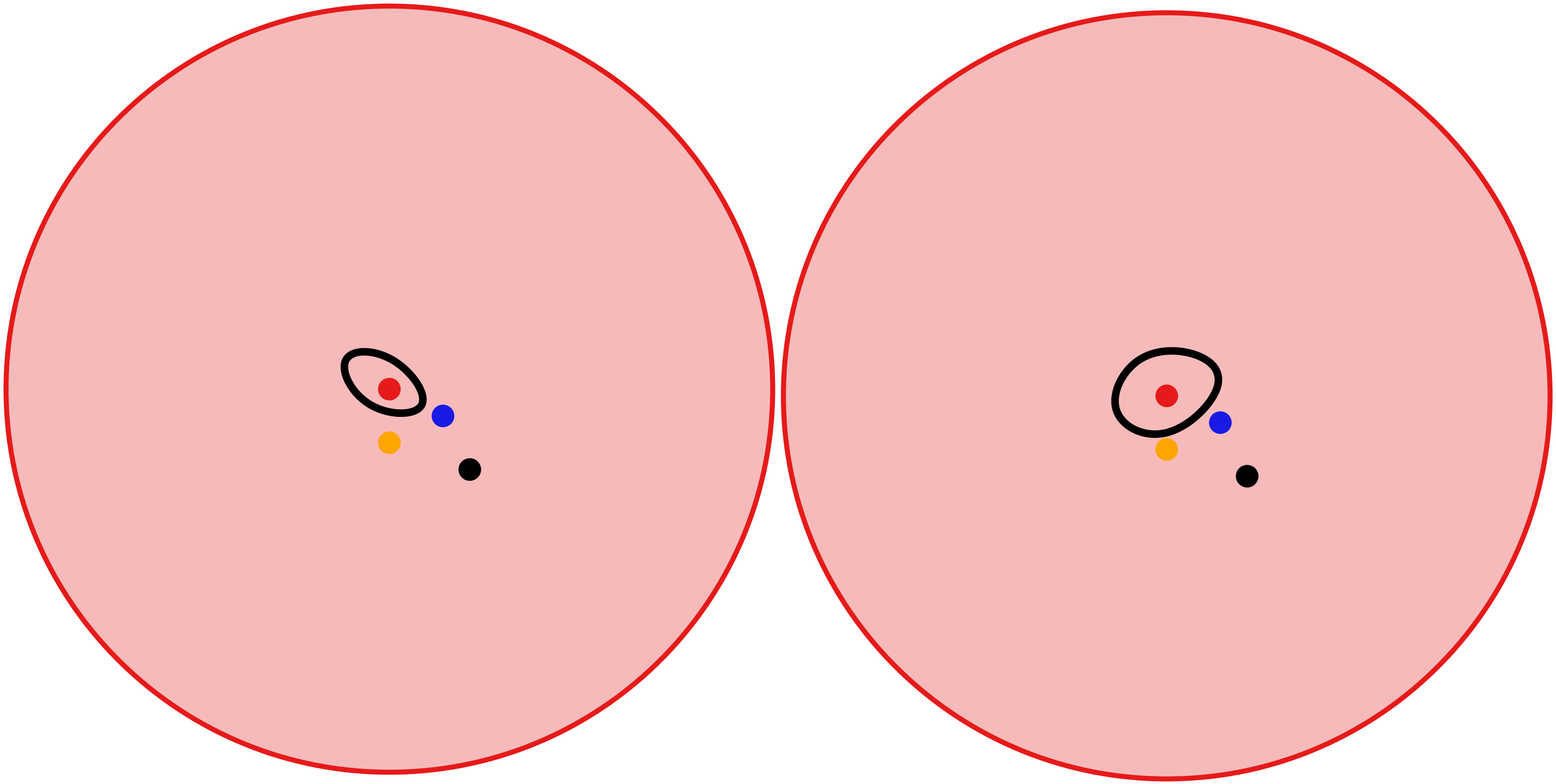
introduce unit disks centered at these points



Hardness of INDEPENDENT SET in UDGs

Theorem. GRID TILING WITH $\leq$ cannot be solved in time $n^{o(t)}$, unless the ETH fails.

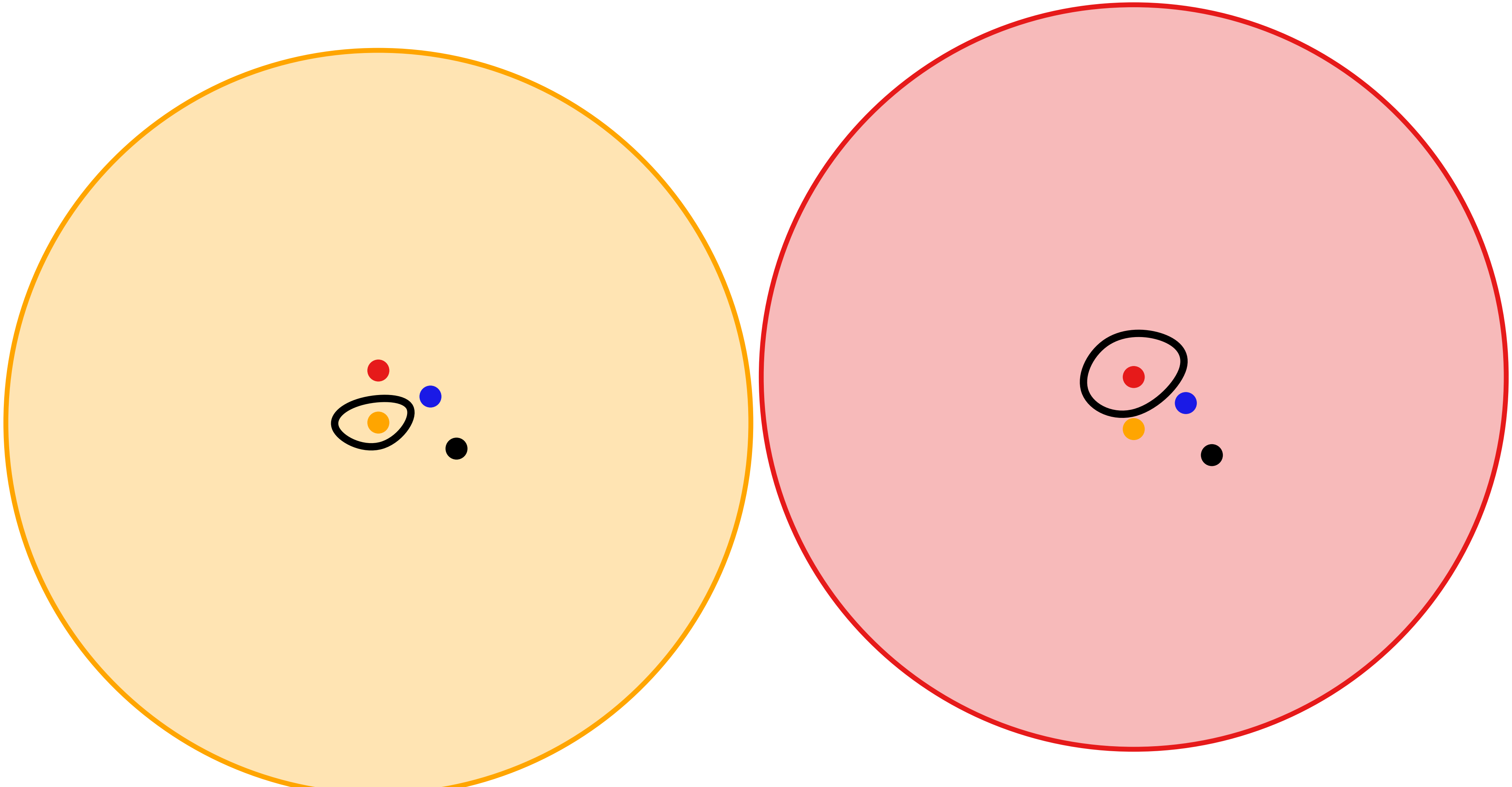
- $t \times t$ outer grid, $n \times n$ inner grids



Hardness of INDEPENDENT SET in UDGs

Theorem. GRID TILING WITH $\leq$ cannot be solved in time $n^{o(t)}$, unless the ETH fails.

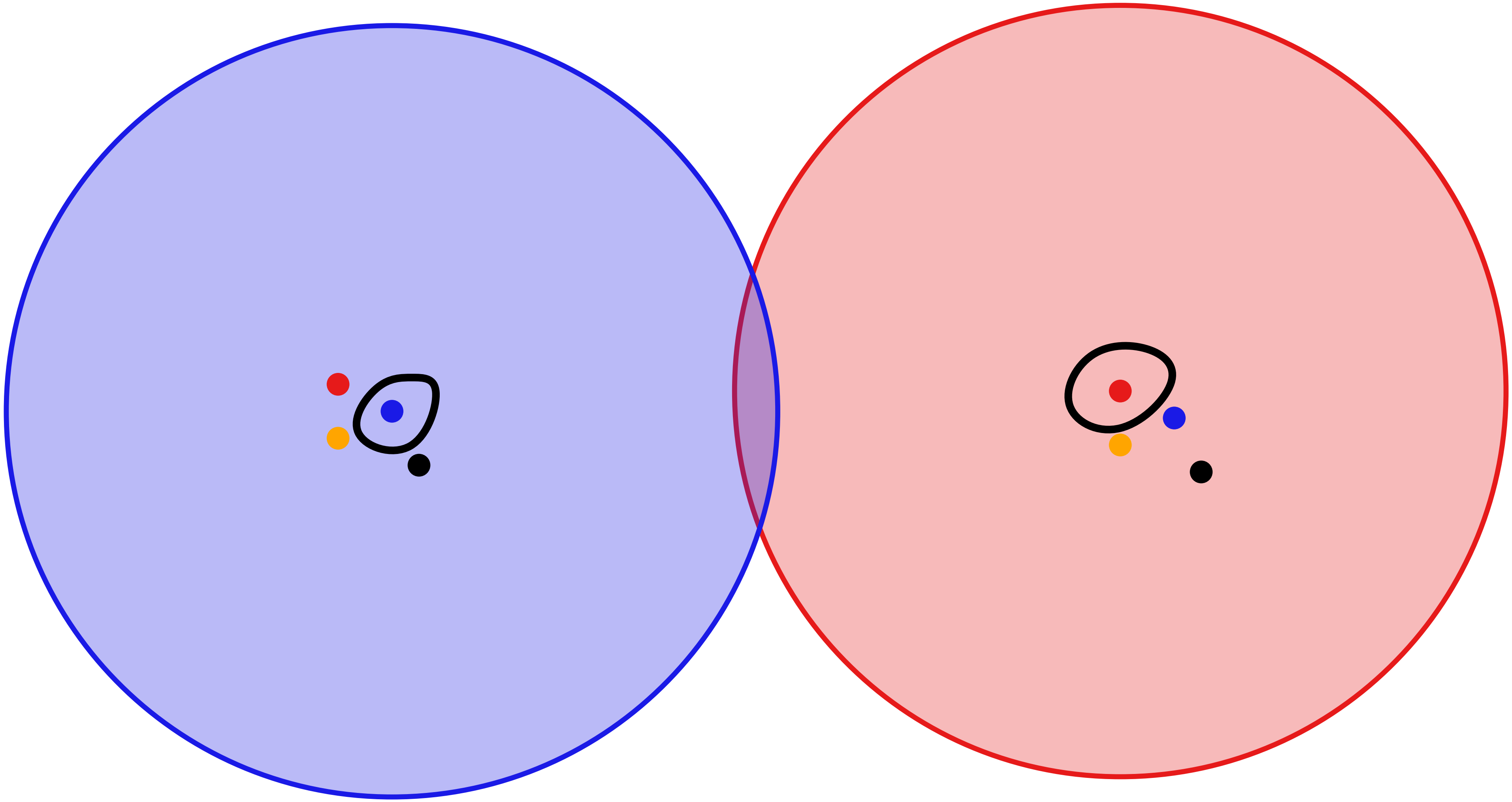
- $t \times t$ outer grid, $n \times n$ inner grids



Hardness of Independent Set in UDGs

Theorem. Grid Tiling with $\leq$ cannot be solved in time $n^{o(t)}$, unless the ETH fails.

- $t \times t$ outer grid, $n \times n$ inner grids



Hardness of INDEPENDENT SET in UDGs

Theorem. GRID TILING WITH $\leq$ cannot be solved in time $n^{o(t)}$, unless the ETH fails.

- $t \times t$ outer grid, $n \times n$ inner grids
- disks from one cell form a clique:
  we have t^2 cliques $\rightarrow$ size of max independent set is $\leq t^2$
- disks from consecutive cells can be chosen if coordinates are non-decreasing

Hardness of Independent Set in UDGs

Theorem. Grid Tiling with $\leq$ cannot be solved in time $n^{o(t)}$, unless the ETH fails.

- $t \times t$ outer grid, $n \times n$ inner grids
- disks from one cell form a clique:
  we have t^2 cliques $\rightarrow$ size of max independent set is $\leq t^2$
- disks from consecutive cells can be chosen if coordinates are non-decreasing
- so the solution of size $k = t^2$ exists if and only if there is a solution for Grid Tiling

Hardness of Independent Set in UDGs

Theorem. Grid Tiling with $\leq$ cannot be solved in time $n^{o(t)}$, unless the ETH fails.

- $t \times t$ outer grid, $n \times n$ inner grids
- disks from one cell form a clique:
  we have t^2 cliques $\to$ size of max independent set is $\leq t^2$
- disks from consecutive cells can be chosen if coordinates are non-decreasing
- so the solution of size $k = t^2$ exists if and only if there is a solution for Grid Tiling
- number of disks $N \leq t^2 \cdot n^2$
- solving Independent Set in time $N^{o(\sqrt{k})}$
  $\to$ solving Grid Tiling in time $n^{o(t)}$ $\to$ the ETH fails $\square$

Other faces of GRID TILING

- similar approach can be used to show lower bounds for (CONNECTED) DOMINATING SET [Marx + Kisfaludi-Bak]
- reductions are not specific to disks: in general they can be adjusted for any convex fat shapes

Other faces of Grid Tiling

- similar approach can be used to show lower bounds for (Connected) Dominating Set [Marx + Kisfaludi-Bak]
- reductions are not specific to disks: in general they can be adjusted for any convex fat shapes

- there is a variant for k-Coloring

Theorem [Biró, Bonnet, Marx, Miltzow, Rz., '16].
k-Coloring of intersection graphs of translates of any convex fat shape cannot be solved in time $2^{o(\sqrt{nk})}$.

here k is a function of n

Other faces of GRID TILING

- similar approach can be used to show lower bounds for (CONNECTED) DOMINATING SET [Marx + Kisfaludi-Bak]
- reductions are not specific to disks: in general they can be adjusted for any convex fat shapes

- there is a variant for k-COLORING

Theorem [Biró, Bonnet, Marx, Miltzow, Rz., '16].
k-COLORING of intersection graphs of translates of any convex fat shape cannot be solved in time $2^{o(\sqrt{nk})}$.

here k is a function of n

- there are also versions for any dimension d:

for INDEPENDENT SET: $2^{\mathcal{O}(k^{1-1/d})}$ [Marx, Sidiropoulos '15]

for k-COLORING: $2^{\widetilde{\mathcal{O}}(n^{1/d} \cdot k^{1-1/d})}$ [BBMMRz '16]

Other faces of GRID TILING

- similar approach can be used to show lower bounds for (CONNECTED) DOMINATING SET [Marx + Kisfaludi-Bak]
- reductions are not specific to disks: in general they can be adjusted for any convex fat shapes

- there is a variant for k-COLORING

Theorem [Biró, Bonnet, Marx, Miltzow, Rz., '16].
k-COLORING of intersection graphs of translates of any convex fat shape cannot be solved in time $2^{o(\sqrt{nk})}$.

here k is a function of n

- there are also versions for any dimension d:

for INDEPENDENT SET: $2^{\mathcal{O}(k^{1-1/d})}$ [Marx, Sidiropoulos '15]

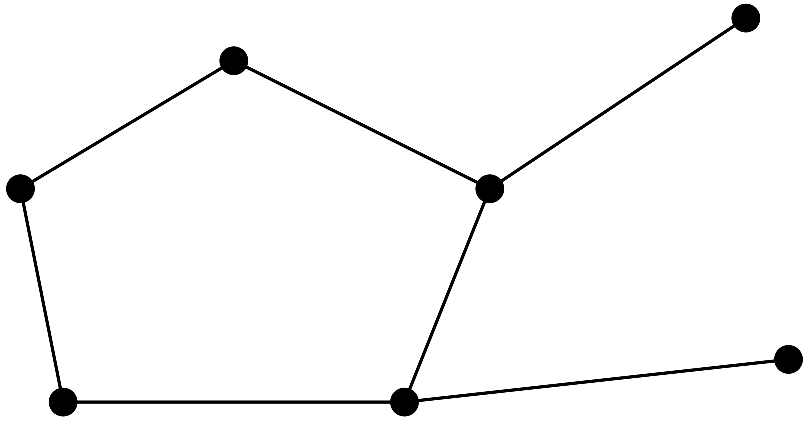
for k-COLORING: $2^{\widetilde{\mathcal{O}}(n^{1/d} \cdot k^{1-1/d})}$ [BBMMRz '16]

... but it's a different story

Bidimensionality in geometric graphs

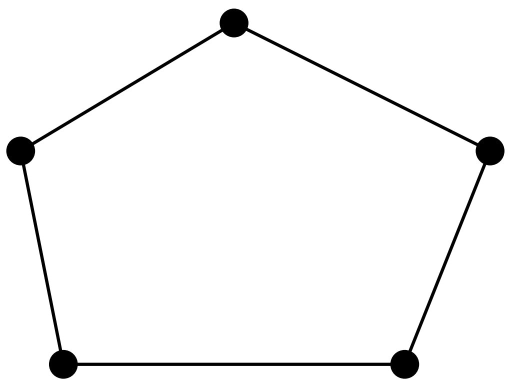
Minors

- minor = a graph obtained by deleting vertices/edges and contracting edges



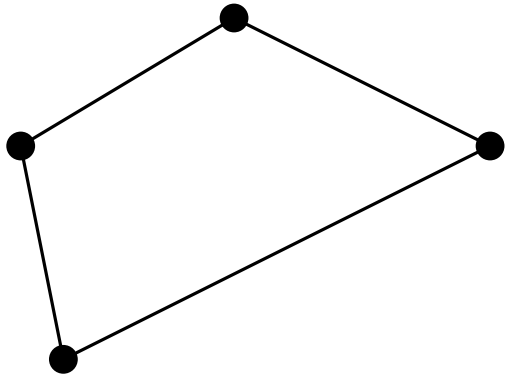
Minors

- minor $=$ a graph obtained by deleting vertices/edges and contracting edges



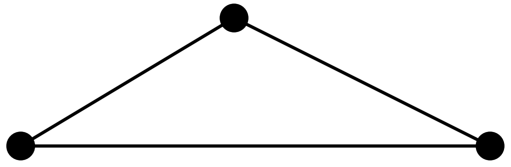
Minors

- minor $=$ a graph obtained by deleting vertices/edges and contracting edges



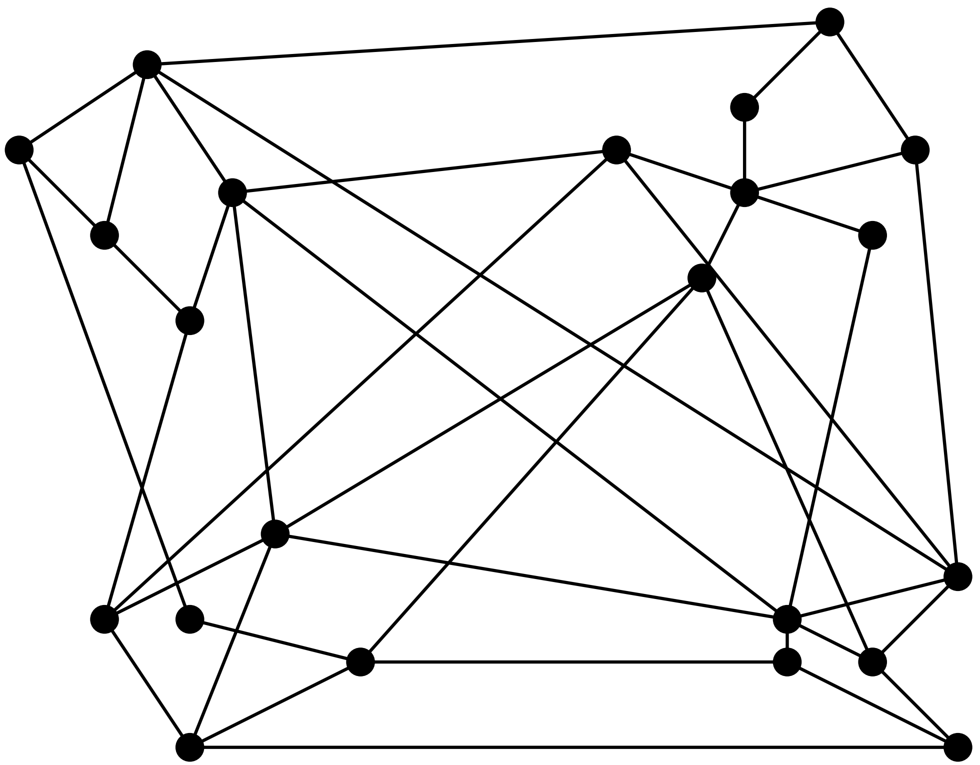
Minors

- minor $=$ a graph obtained by deleting vertices/edges and contracting edges



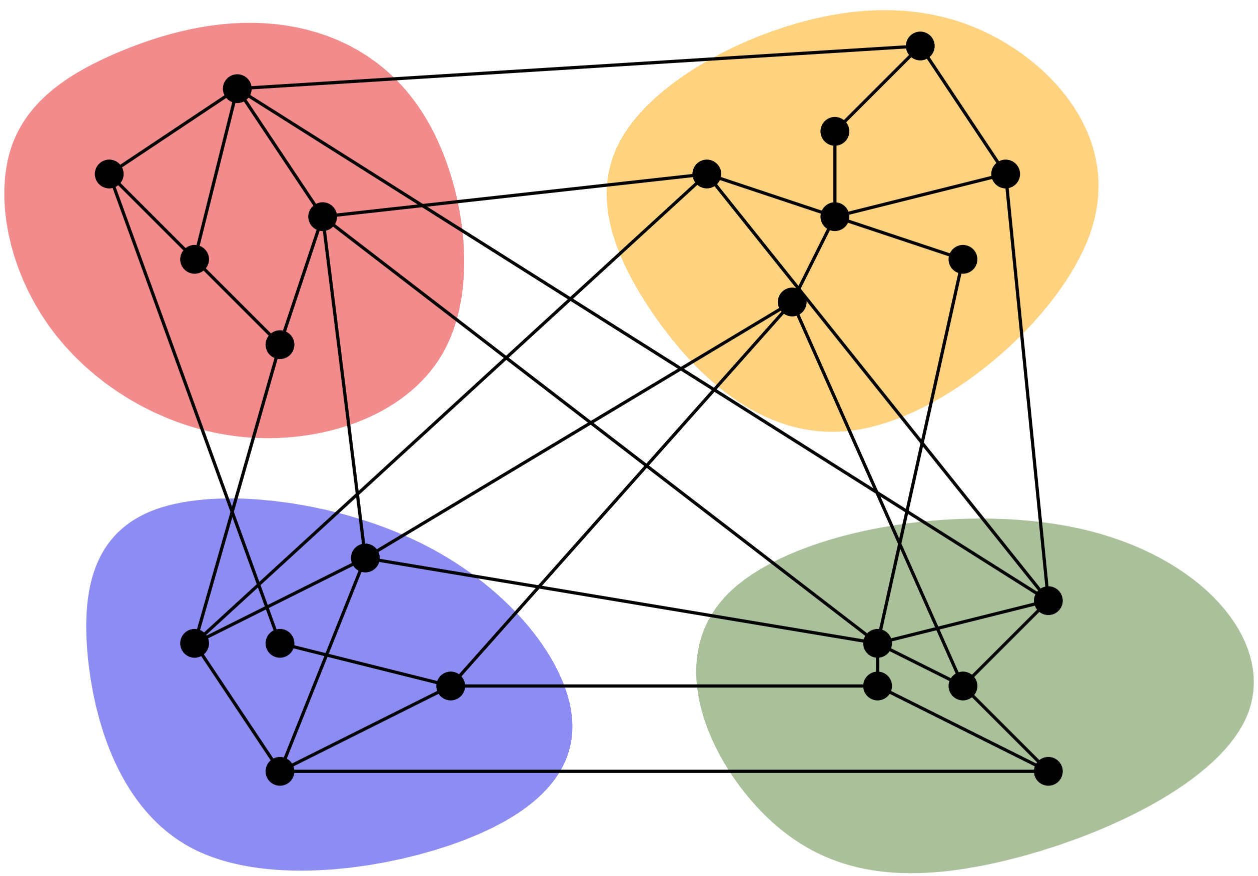
Minors

- minor = a graph obtained by deleting vertices/edges and contracting edges
- find some disjoint connected subgraphs and contract them to single vertices



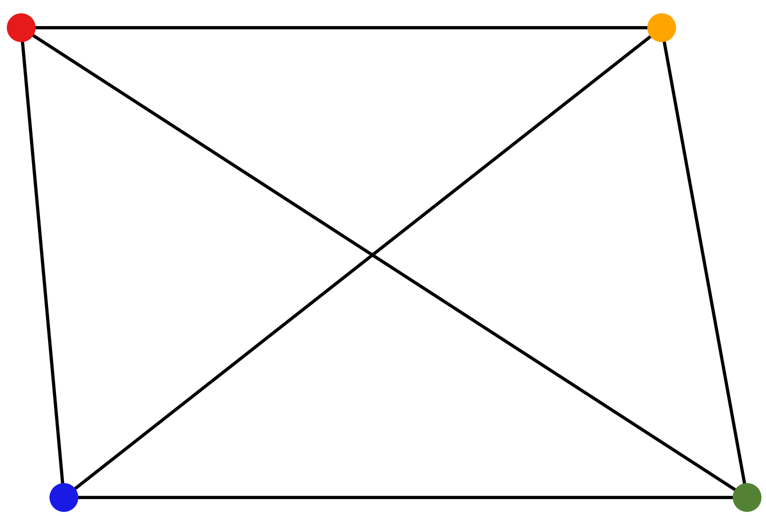
Minors

- minor = a graph obtained by deleting vertices/edges and contracting edges
- find some disjoint connected subgraphs and contract them to single vertices



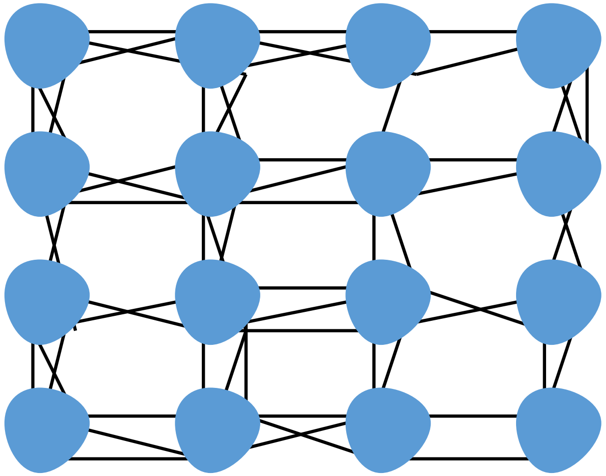
Minors

- minor $=$ a graph obtained by deleting vertices/edges and contracting edges
- find some disjoint connected subgraphs and contract them to single vertices



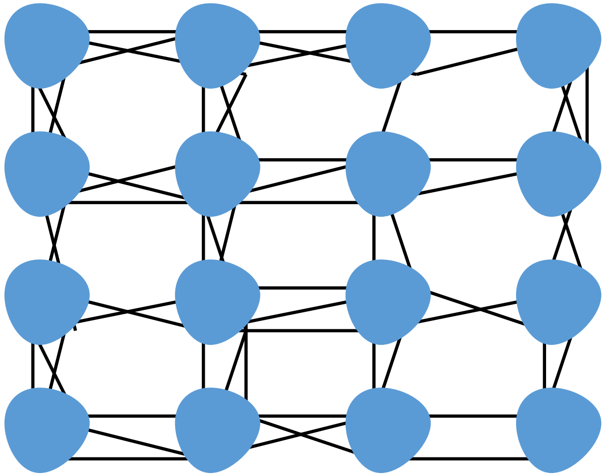
Grid minor theorem

- the presence of $t \times t$ grid minor forces treewidth $\geq t$



Grid minor theorem

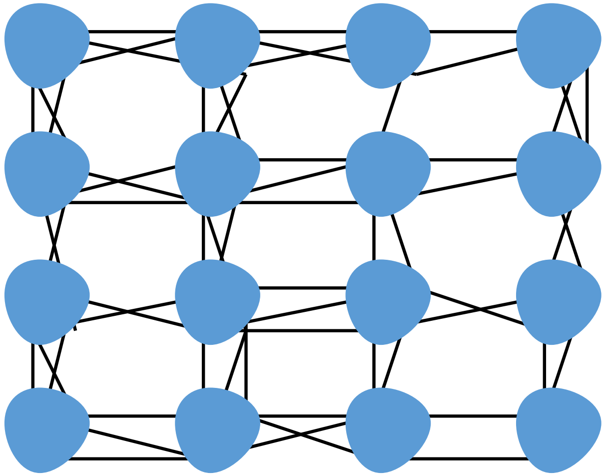
- the presence of $t \times t$ grid minor forces treewidth $\geq t$



Grid minor theorem [Robertson, Seymour '86].
Every graph with treewidth $\geq f(t)$ contains a $t \times t$ grid minor.

Grid minor theorem

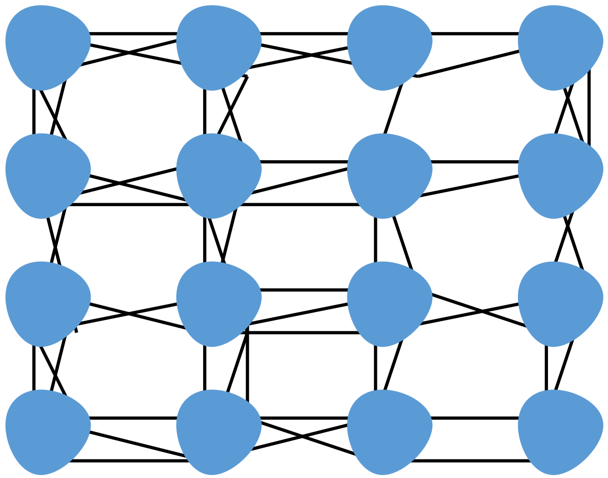
- the presence of $t \times t$ grid minor forces treewidth $\geq t$



Grid minor theorem [Chuzhoy, Tan '19].
Every graph with treewidth $\widetilde{\Omega}(t^9)$ contains a $t \times t$ grid minor.

Grid minor theorem

- the presence of $t \times t$ grid minor forces treewidth $\geq t$



Grid minor theorem [Chuzhoy, Tan '19].
Every graph with treewidth $\widetilde{\Omega}(t^9)$ contains a $t \times t$ grid minor.

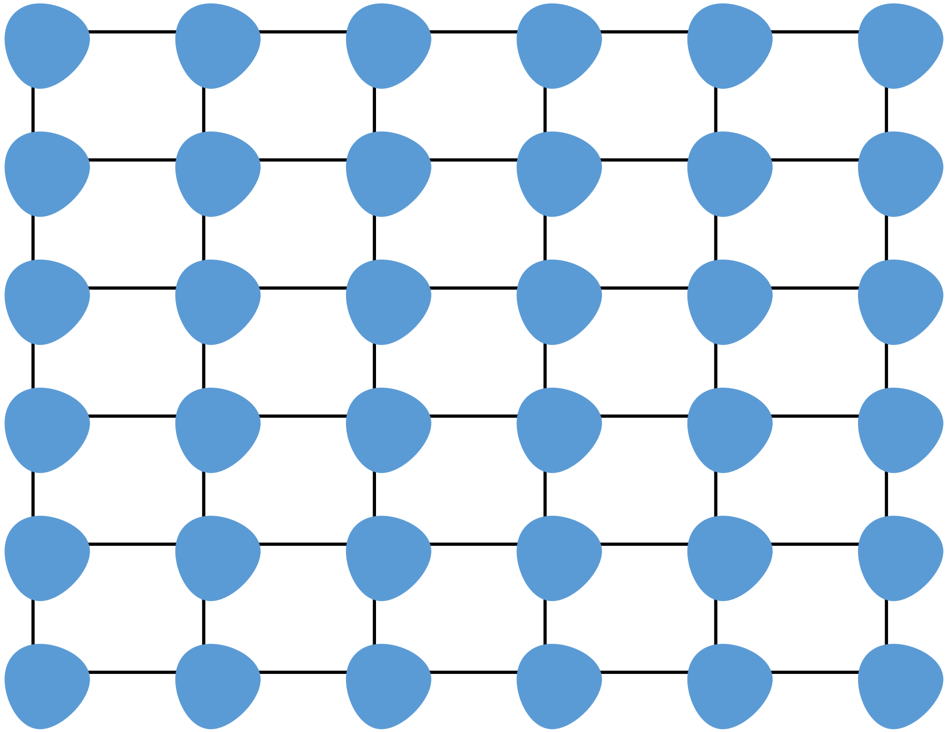
Planar grid minor theorem [Robertson, Seymour, Thomas '94, Gu, Tamaki '12].
Every planar graph with treewidth $\geq 9/2 \cdot t$ contains a $t \times t$ grid minor. There is a poly-time algorithm for finding a grid or a tree decomposition.

Bidimensionality for planar graphs

- if treewidth is $\mathcal{O}(\sqrt{k})$, then many problem can be solved in time $2^{\widetilde{\mathcal{O}}(\sqrt{k})} \cdot \mathrm{poly}(n)$

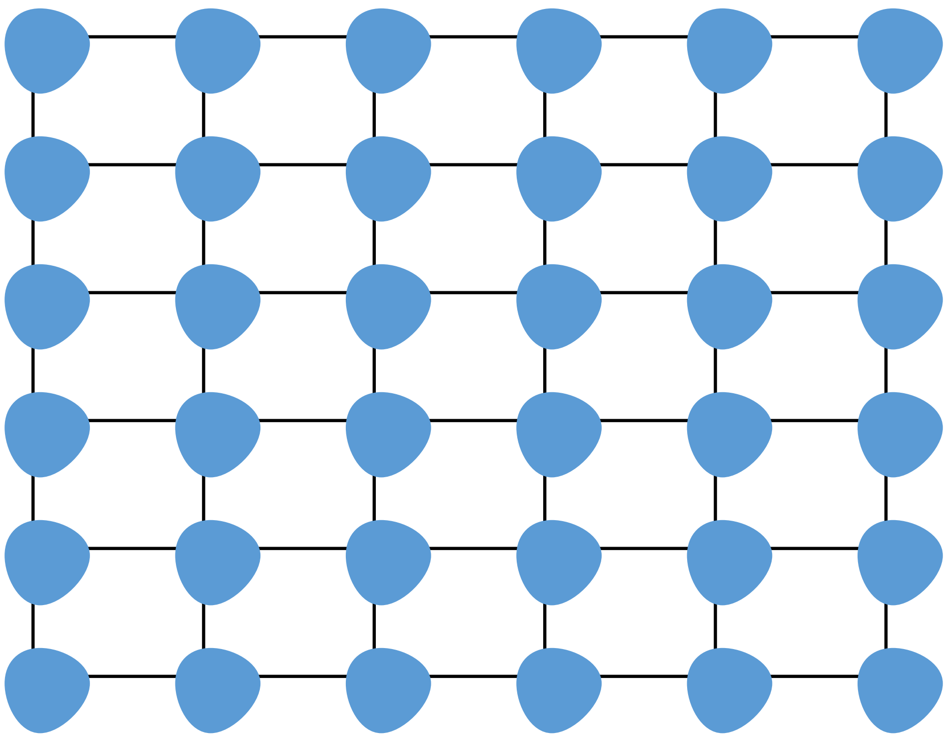
Bidimensionality for planar graphs

- if treewidth is $\mathcal{O}(\sqrt{k})$, then many problem can be solved in time $2^{\widetilde{\mathcal{O}}(\sqrt{k})} \cdot \mathrm{poly}(n)$
- if not, we have a $100\sqrt{k} \times 100\sqrt{k}$ grid minor



Bidimensionality for planar graphs

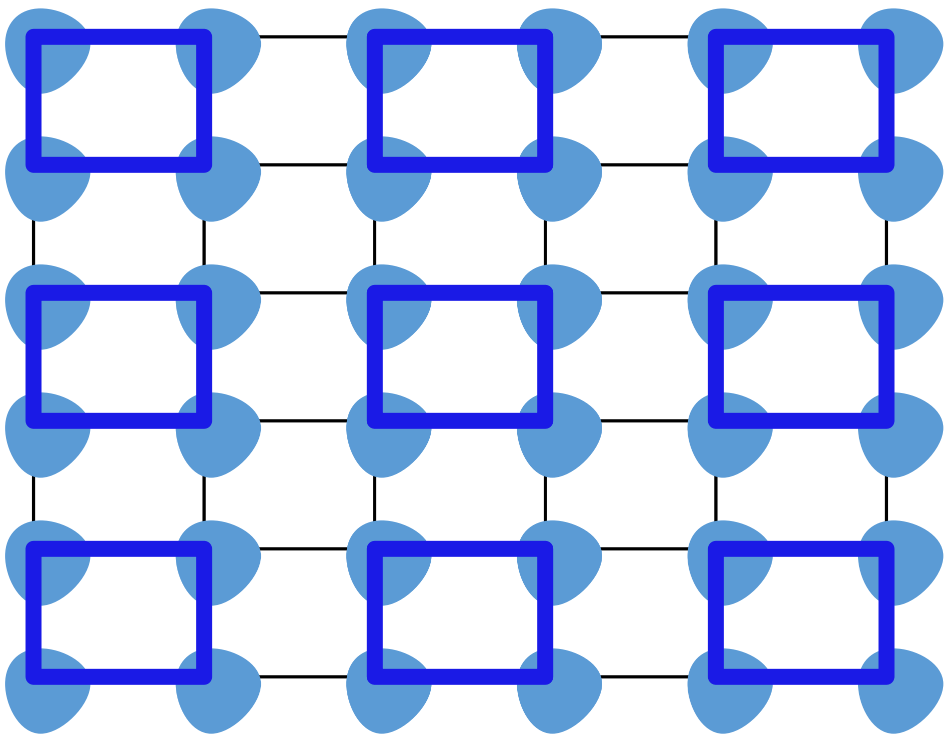
- if treewidth is $\mathcal{O}(\sqrt{k})$, then many problem can be solved in time $2^{\widetilde{\mathcal{O}}(\sqrt{k})} \cdot \mathrm{poly}(n)$
- if not, we have a $100\sqrt{k} \times 100\sqrt{k}$ grid minor



- k-Feedback Vertex Set:
  is there a feedback vertex set of size $\leq k$?

Bidimensionality for planar graphs

- if treewidth is $\mathcal{O}(\sqrt{k})$, then many problem can be solved in time $2^{\widetilde{\mathcal{O}}(\sqrt{k})} \cdot \mathrm{poly}(n)$
- if not, we have a $100\sqrt{k} \times 100\sqrt{k}$ grid minor

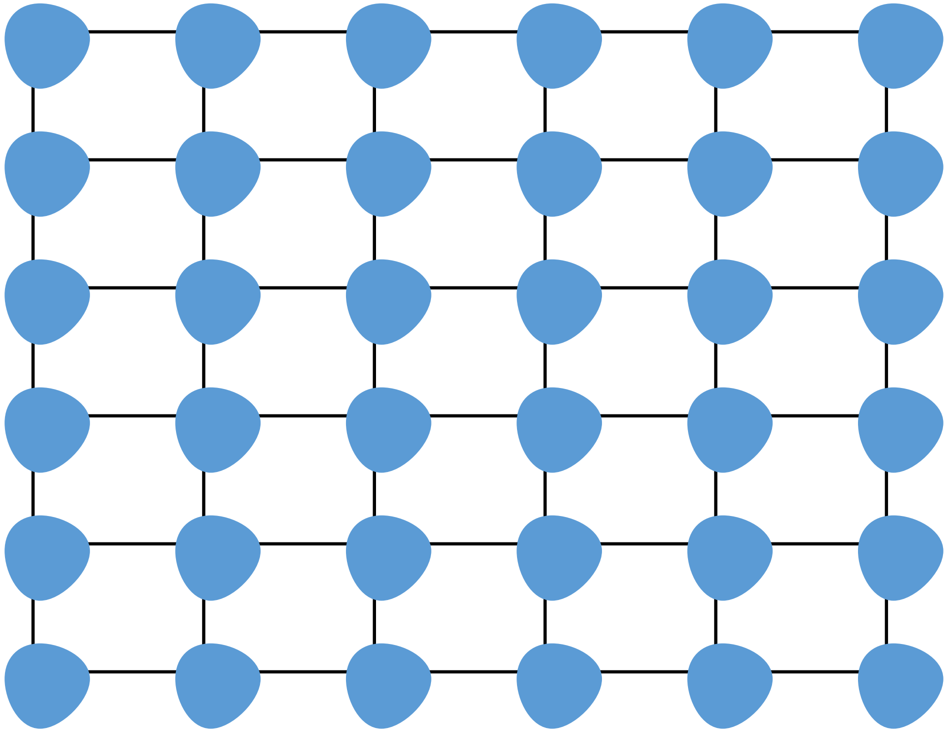


NO!

- k-Feedback Vertex Set:
  is there a feedback vertex set of size $\leq k$?

Bidimensionality for planar graphs

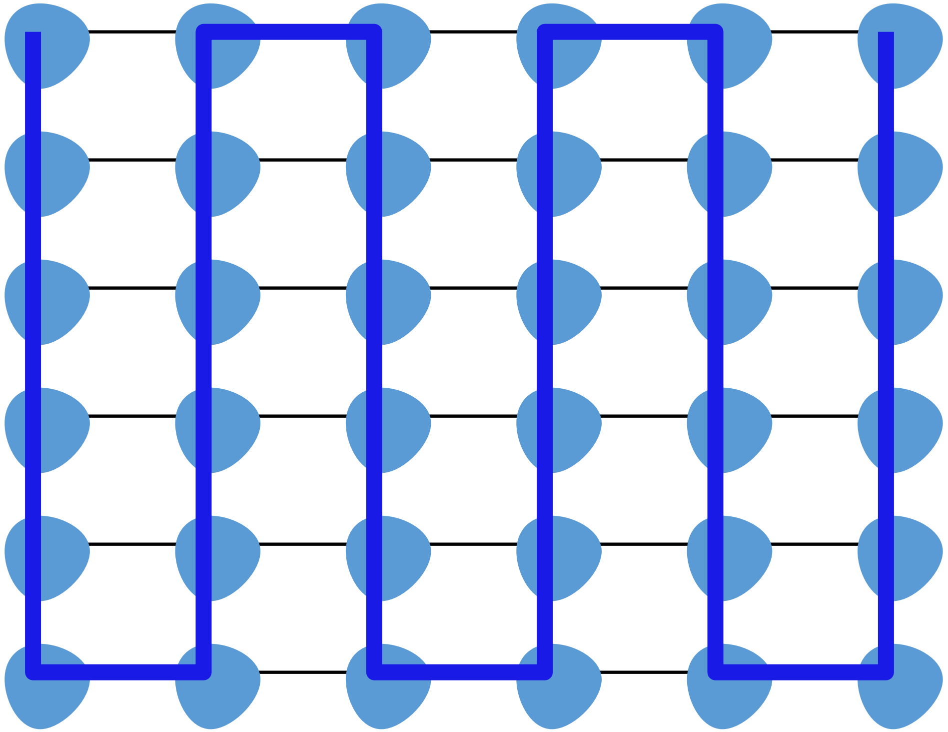
- if treewidth is $\mathcal{O}(\sqrt{k})$, then many problem can be solved in time $2^{\widetilde{\mathcal{O}}(\sqrt{k})} \cdot \mathrm{poly}(n)$
- if not, we have a $100\sqrt{k} \times 100\sqrt{k}$ grid minor



- k-Path:
  is there a path of length $\geq k$?

Bidimensionality for planar graphs

- if treewidth is $\mathcal{O}(\sqrt{k})$, then many problem can be solved in time $2^{\widetilde{\mathcal{O}}(\sqrt{k})} \cdot \mathrm{poly}(n)$
- if not, we have a $100\sqrt{k} \times 100\sqrt{k}$ grid minor

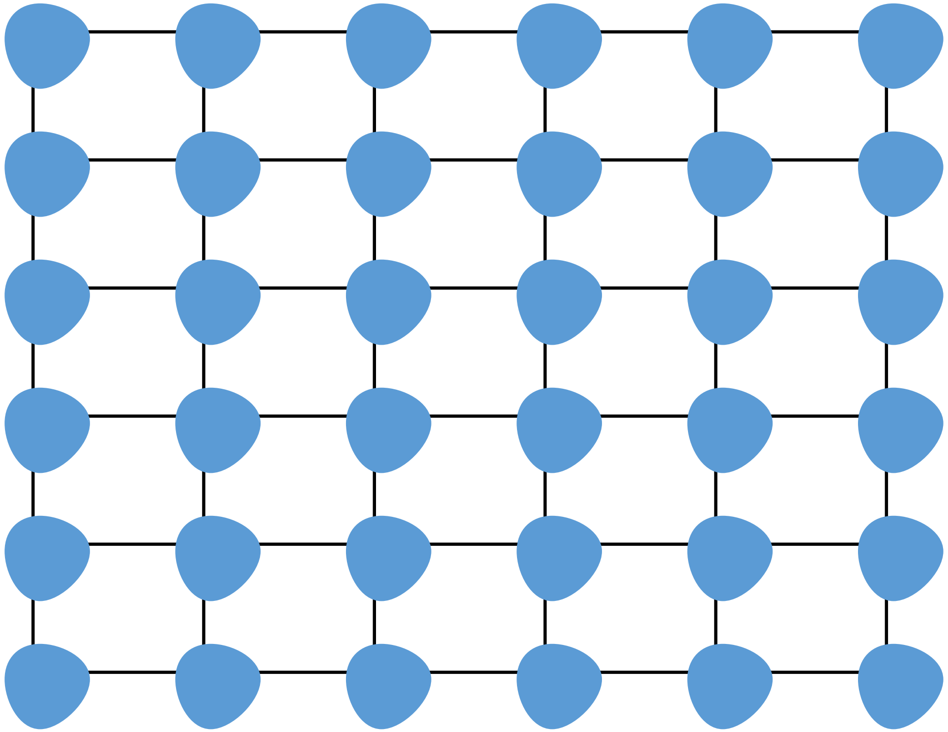


YES!

- k-Path:
  is there a path of length $\geq k$?

Bidimensionality for planar graphs

- if treewidth is $\mathcal{O}(\sqrt{k})$, then many problem can be solved in time $2^{\widetilde{\mathcal{O}}(\sqrt{k})} \cdot \text{poly}(n)$
- if not, we have a $100\sqrt{k} \times 100\sqrt{k}$ grid minor



- $2^{\widetilde{\mathcal{O}}(\sqrt{k})} \cdot \text{poly}(n)$-algorithms for many parameterized problems

Grid minors in unit disk graphs

- we aim to prove a grid minor theorem for unit disk graphs

Grid minors in unit disk graphs

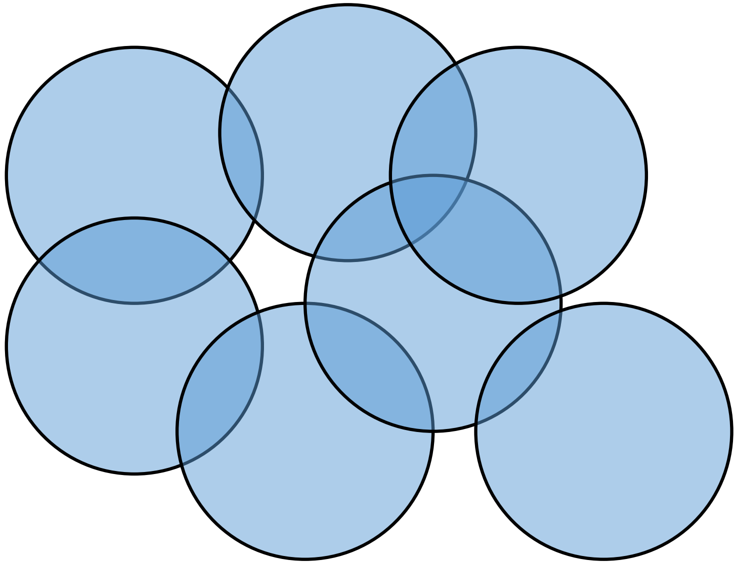
- we aim to prove a grid minor theorem for unit disk graphs

Lemma [Fomin, Lokshtanov, Saurabh '11].
Every unit disk graph G with bounded maximum degree and treewidth $\Omega(t)$ has a $t \times t$ grid minor.

Grid minors in unit disk graphs

- we aim to prove a grid minor theorem for unit disk graphs

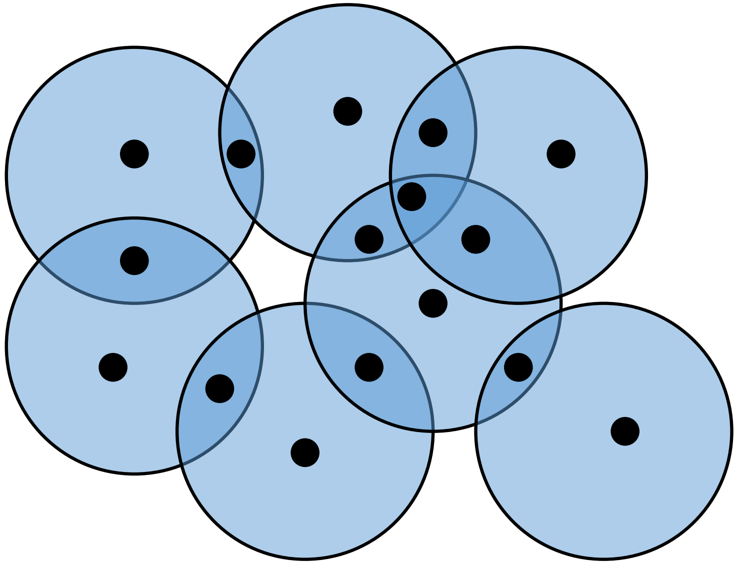
Lemma [Fomin, Lokshtanov, Saurabh '11].
Every unit disk graph G with bounded maximum degree and treewidth $\Omega(t)$ has a $t \times t$ grid minor.



Grid minors in unit disk graphs

- we aim to prove a grid minor theorem for unit disk graphs

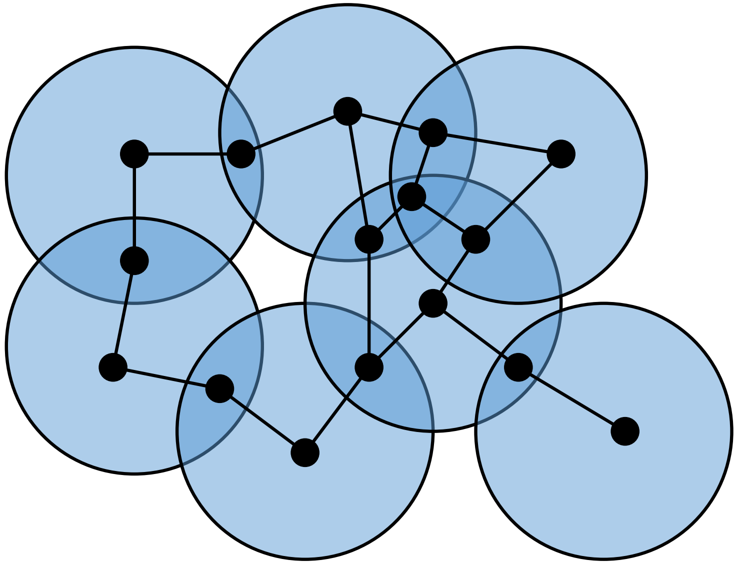
Lemma [Fomin, Lokshtanov, Saurabh '11].
Every unit disk graph G with bounded maximum degree and treewidth $\Omega(t)$ has a $t \times t$ grid minor.



Grid minors in unit disk graphs

- we aim to prove a grid minor theorem for unit disk graphs

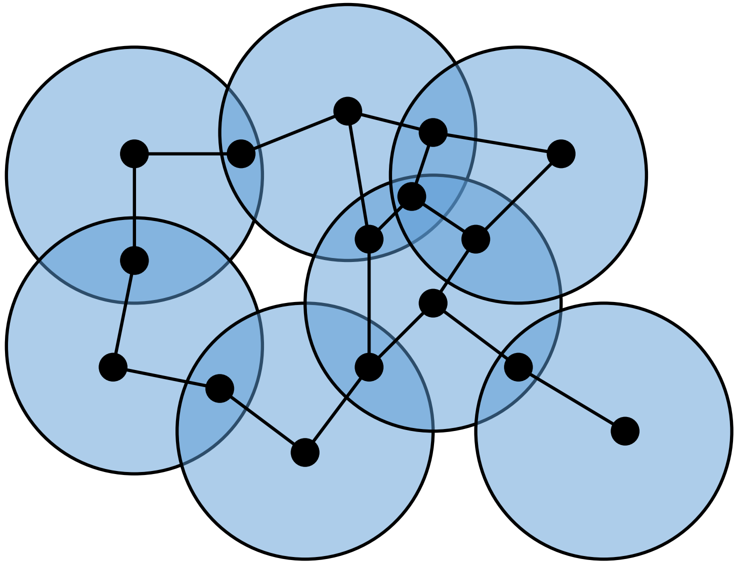
Lemma [Fomin, Lokshtanov, Saurabh '11].
Every unit disk graph G with bounded maximum degree and treewidth $\Omega(t)$ has a $t \times t$ grid minor.



- $R(G)$ – region graph, $R(G)$ is planar

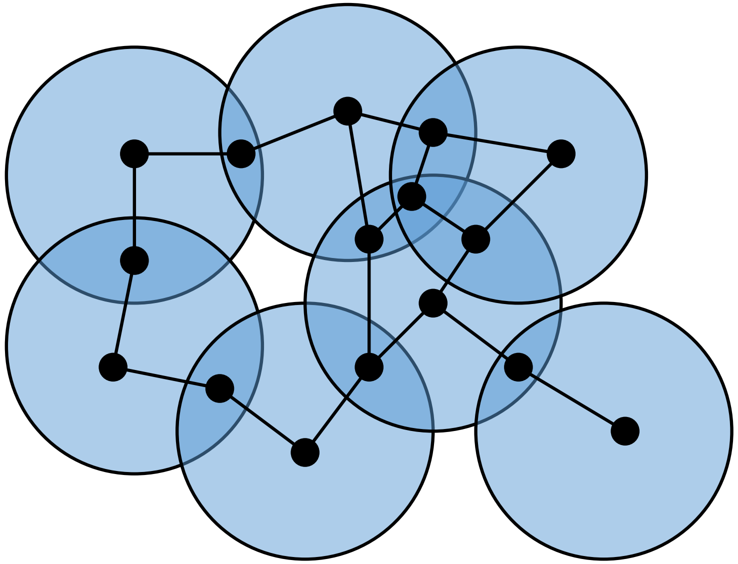
Grid minors in unit disk graphs, continued

- $R(G)$ – region graph,
  $R(G)$ is planar



Grid minors in unit disk graphs, continued

- $R(G)$ – region graph,
  $R(G)$ is planar

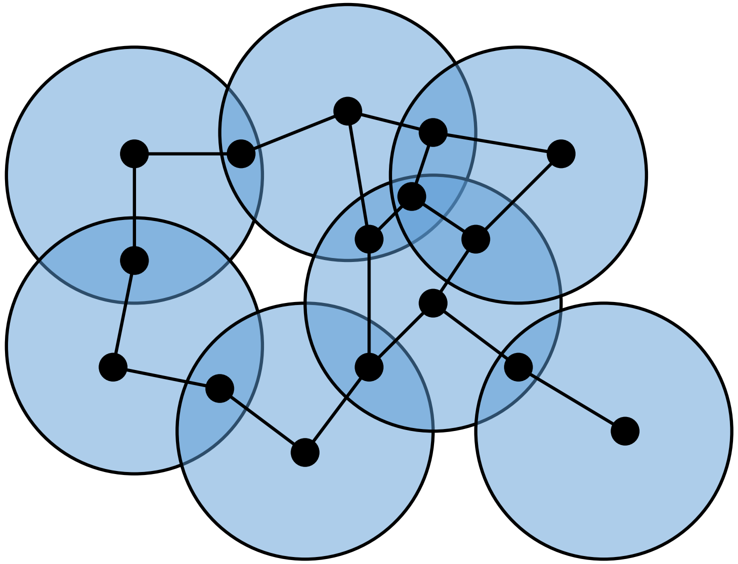


Lemma. $\mathrm{tw}(G) = \mathcal{O}(\mathrm{tw}(R(G)))$

- construct a tree decomposition of G based on a tree decomposition of $R(G)$

Grid minors in unit disk graphs, continued

- $R(G)$ – region graph,
  $R(G)$ is planar



Lemma. $\mathrm{tw}(G) = \mathcal{O}(\mathrm{tw}(R(G)))$

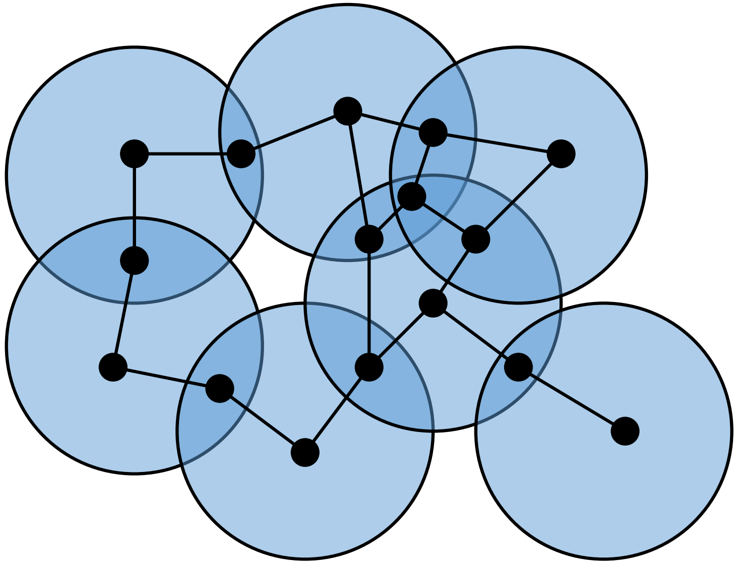
- construct a tree decomposition of G based on a tree decomposition of $R(G)$

How to use it?

- $R(G)$ contains $t \times t$ grid minor, where $t = \mathcal{O}(\mathrm{tw}(R(G)))$.

Grid minors in unit disk graphs, continued

- $R(G)$ – region graph,
  $R(G)$ is planar



Lemma. $\mathrm{tw}(G) = \mathcal{O}(\mathrm{tw}(R(G))$

- construct a tree decomposition of G based on a tree decomposition of $R(G)$

How to use it?

- $R(G)$ contains $t \times t$ grid minor, where $t = \mathcal{O}(\mathrm{tw}(R(G)))$.
- using this, we construct a $t' \times t'$ grid minor in G, where $t' = \mathcal{O}(t) = \mathcal{O}(\mathrm{tw}(G))$

Grid minor theorem for unit disk graphs

Lemma[Fomin, Lokshtanov, Saurabh '11].
Every unit disk graph G with bounded maximum degree and treewidth $\Omega(t)$ has a $t \times t$ grid minor.

Grid minor theorem for unit disk graphs

Lemma[Fomin, Lokshtanov, Saurabh '11].

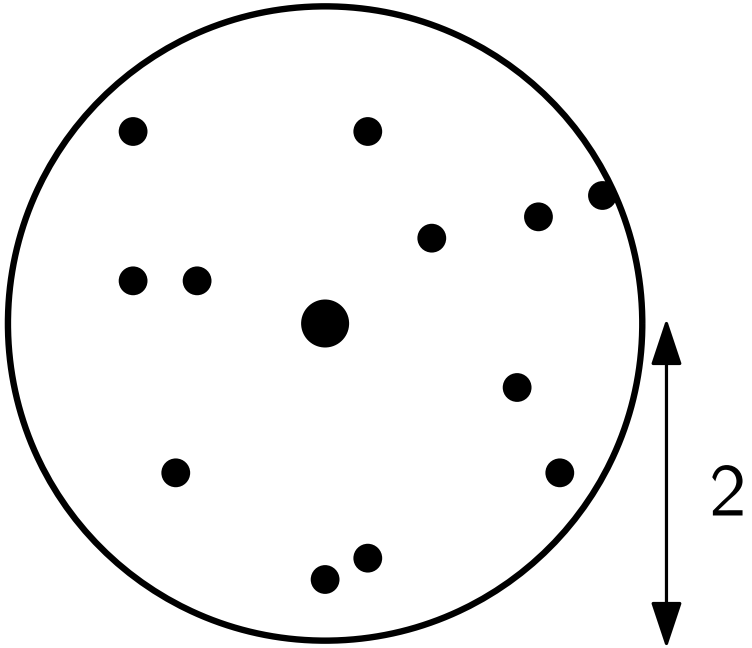
Every unit disk graph G with bounded maximum degree and treewidth $\Omega(t)$ has a $t \times t$ grid minor.

- if G has no clique of size p, then $\Delta \leq 6p$

Grid minor theorem for unit disk graphs

Lemma[Fomin, Lokshtanov, Saurabh '11].
Every unit disk graph G with bounded maximum degree and treewidth $\Omega(t)$ has a $t \times t$ grid minor.

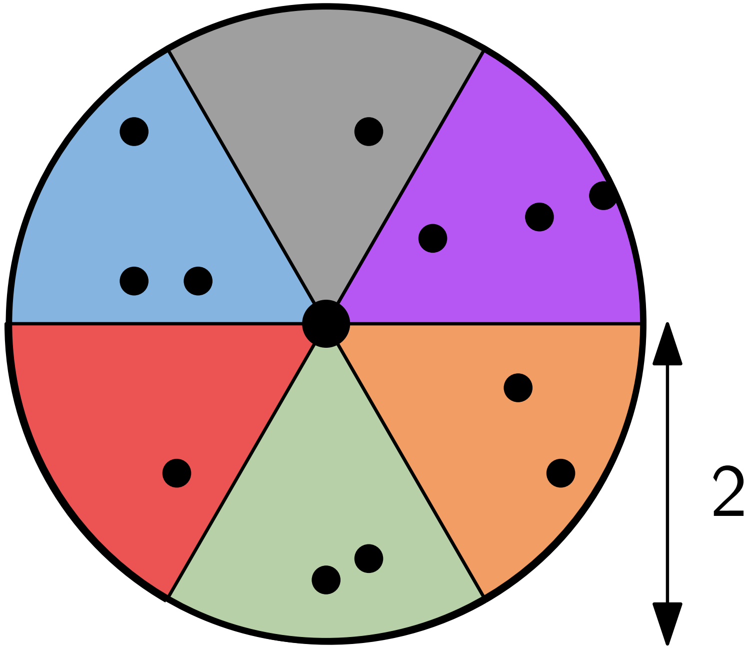
- if G has no clique of size p, then $\Delta \leq 6p$
- take a vertex of degree Δ
- centers or all neighbors are in the radius-2 disk



Grid minor theorem for unit disk graphs

Lemma[Fomin, Lokshtanov, Saurabh '11].
Every unit disk graph G with bounded maximum degree and treewidth $\Omega(t)$ has a $t \times t$ grid minor.

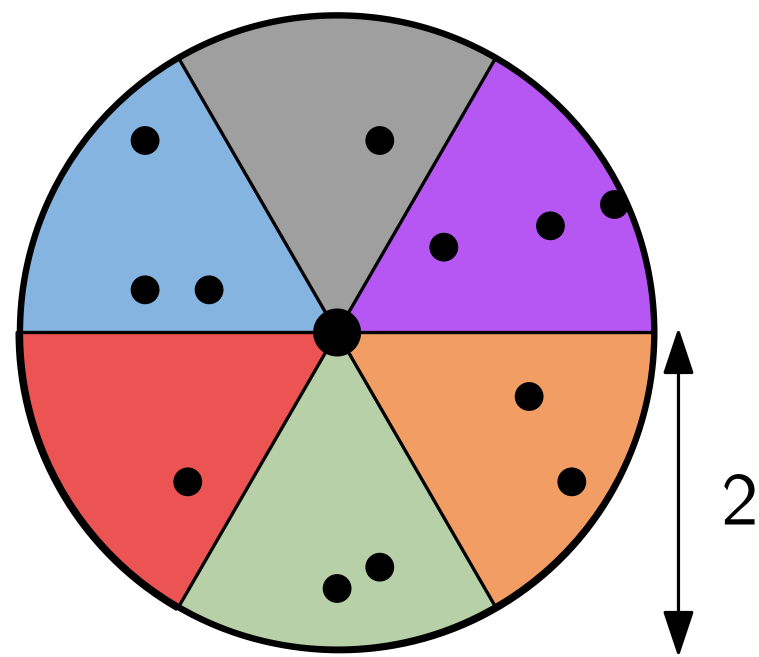
- if G has no clique of size p, then $\Delta \leq 6p$
- take a vertex of degree Δ
- centers or all neighbors are in the radius-2 disk
- centers in each region correspond to a clique



Grid minor theorem for unit disk graphs

Lemma[Fomin, Lokshtanov, Saurabh '11].
Every unit disk graph G with bounded maximum degree and treewidth $\Omega(t)$ has a $t \times t$ grid minor.

- if G has no clique of size p, then $\Delta \leq 6p$
- take a vertex of degree Δ
- centers or all neighbors are in the radius-2 disk
- centers in each region correspond to a clique
- add some technical magic



Theorem [FLS '11].
Every unit disk graph with no p-clique and treewidth $\Omega(p \cdot t)$ has a $t \times t$ grid minor.

Yet another win-win algorithm

- k-Feedback Vertex Set in unit disk graphs: is there a feedback vertex set of size $\leq k$?

Yet another win-win algorithm

- k-FEEDBACK VERTEX SET in unit disk graphs:
  is there a feedback vertex set of size $\leq k$?

Initialization.

$C \leftarrow$ a maximum clique in G (polynomial to find)
$t \leftarrow 100\sqrt{k}$
$\varepsilon \leftarrow 0.25$

Yet another win-win algorithm

- k-Feedback Vertex Set in unit disk graphs: is there a feedback vertex set of size $\leq k$?

Initialization.

$C \leftarrow$ a maximum clique in G (polynomial to find)
$t \leftarrow 100\sqrt{k}$
$\varepsilon \leftarrow 0.25$

1. If $|C| > k + 2$, return NO.

Yet another win-win algorithm

- k-Feedback Vertex Set in unit disk graphs: is there a feedback vertex set of size $\leq k$?

Initialization.

$C \leftarrow$ a maximum clique in G (polynomial to find)
$t \leftarrow 100\sqrt{k}$
$\varepsilon \leftarrow 0.25$

1. If $|C| > k+2$, return NO.
2. If $|C| > k^{\varepsilon}$, branch:

Yet another win-win algorithm

- k-Feedback Vertex Set in unit disk graphs:
  is there a feedback vertex set of size $\leq k$?

Initialization.

$C \leftarrow$ a maximum clique in G (polynomial to find)
$t \leftarrow 100\sqrt{k}$
$\varepsilon \leftarrow 0.25$

1. If $|C| > k+2$, return NO.
2. If $|C| > k^{\varepsilon}$, branch:

$$T(n,k) \leq k^{2\varepsilon} \cdot T(n, k-k^{\varepsilon}) \leq \exp\{k^{1-\epsilon} \log k\} \cdot \text{poly}(n)$$

Yet another win-win algorithm

- k-Feedback Vertex Set in unit disk graphs: is there a feedback vertex set of size $\leq k$?

Initialization.

$C \leftarrow$ a maximum clique in G (polynomial to find)
$t \leftarrow 100\sqrt{k}$
$\varepsilon \leftarrow 0.25$

1. If $|C| > k+2$, return NO.
2. If $|C| > k^{\varepsilon}$, branch: $\exp\{k^{1-\epsilon}\log k\} \cdot \text{poly}(n)$

Yet another win-win algorithm

- k-Feedback Vertex Set in unit disk graphs: is there a feedback vertex set of size $\leq k$?

Initialization.

$C \leftarrow$ a maximum clique in G (polynomial to find)
$t \leftarrow 100\sqrt{k}$
$\varepsilon \leftarrow 0.25$

1. If $|C| > k+2$, return NO.
2. If $|C| > k^{\varepsilon}$, branch: $\exp\{k^{1-\epsilon} \log k\} \cdot \text{poly}(n)$
3. If $|C| < k^{\varepsilon}$, then one of the following occurs:
   (a) treewidth $= \mathcal{O}(k^{\varepsilon} \cdot t) = k^{\mathcal{O}(1/2+\varepsilon)}$, divide & conquer
   $\exp\{k^{1+\epsilon}\} \cdot \text{poly}(n)$
   (b) grid minor of size $t \times t \rightarrow$ return NO

Yet another win-win algorithm

- k-Feedback Vertex Set in unit disk graphs:
  is there a feedback vertex set of size $\leq k$?

Initialization.

$C \leftarrow$ a maximum clique in G (polynomial to find)
$t \leftarrow 100\sqrt{k}$
$\varepsilon \leftarrow 0.25$

1. If $|C| > k+2$, return NO.
2. If $|C| > k^{\varepsilon}$, branch: $\exp\{k^{1-\epsilon} \log k\} \cdot \mathrm{poly}(n)$
3. If $|C| < k^{\varepsilon}$, then one of the following occurs:
   (a) treewidth $= \mathcal{O}(k^{\varepsilon} \cdot t) = k^{\mathcal{O}(1/2+\varepsilon)}$, divide & conquer
   $\exp\{k^{1+\epsilon}\} \cdot \mathrm{poly}(n)$
   (b) grid minor of size $t \times t \rightarrow$ return NO

Overall running time is $2^{\mathcal{O}(k^{0.75} \cdot \log k)} \cdot \mathrm{poly}(n)$.

Concluding comments

- this works for k-Cycle Packing, k-Cycle, k-Path, (Connected) k-Vertex Cover
- can be used to obtain EPTASes

Concluding comments

- this works for k-Cycle Packing, k-Cycle, k-Path, (Connected) k-Vertex Cover
- can be used to obtain EPTASes
- does not generalize to non-unit disk graphs

Concluding comments

- this works for k-Cycle Packing, k-Cycle, k-Path, (Connected) k-Vertex Cover
- can be used to obtain EPTASes
- does not generalize to non-unit disk graphs
- we know algorithms with running time $2^{\mathcal{O}(\sqrt{k})} \cdot \mathrm{poly}(n)$ e.g. [Fomin, Lokshtanov, Panolan, Saurabh, Zehavi '19]
- no $2^{o(\sqrt{k})} \cdot \mathrm{poly}(n)$-algorithms, unless the ETH fails

Concluding comments

- this works for k-Cycle Packing, k-Cycle, k-Path, (Connected) k-Vertex Cover
- can be used to obtain EPTASes
- does not generalize to non-unit disk graphs
- we know algorithms with running time $2^{\mathcal{O}(\sqrt{k})} \cdot \mathrm{poly}(n)$ e.g. [Fomin, Lokshtanov, Panolan, Saurabh, Zehavi '19]
- no $2^{o(\sqrt{k})} \cdot \mathrm{poly}(n)$-algorithms, unless the ETH fails