The Square Root Phenomenon in Planar Graphs

Survey and New Results

Dániel Marx

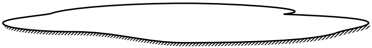
Institute for Computer Science and Control,
Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary

Satisfiability Lower Bounds and Tight Results for Parameterized and Exponential-Time Algorithms
Simons Institute, Berkeley, CA
November 6, 2015

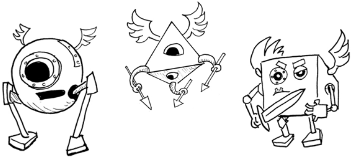
Main message

NP-hard problems become easier on planar graphs and geometric objects, and usually exactly by a square root factor.

Planar graphs



Geometric objects



Better exponential algorithms

Most NP-hard problems (e.g., 3-Coloring, Independent Set, Hamiltonian Cycle, Steiner Tree, etc.) remain NP-hard on planar graphs,[1] so what do we mean by "easier"?

[1]Notable exception: Max Cut is in P for planar graphs.

Better exponential algorithms

Most NP-hard problems (e.g., 3-Coloring, Independent Set, Hamiltonian Cycle, Steiner Tree, etc.) remain NP-hard on planar graphs,[1] so what do we mean by "easier"?

The running time is still exponential, but significantly smaller:

$$
\begin{aligned}
2^{O(n)} &\quad\Rightarrow\quad 2^{O(\sqrt{n})} \\
n^{O(k)} &\quad\Rightarrow\quad n^{O(\sqrt{k})} \\
2^{O(k)} \cdot n^{O(1)} &\quad\Rightarrow\quad 2^{O(\sqrt{k})} \cdot n^{O(1)}
\end{aligned}
$$

[1] Notable exception: Max Cut is in P for planar graphs.

Overview

We repeat some of the material from the boot camp, but will also see new results.

Chapter 1:
Subexponential algorithms using treewidth.

Chapter 2:
Grid minors and bidimensionality.

Chapter 3:
Beyond bidimensionality:
Finding bounded-treewidth solutions.

Chapter 1: Subexponential algorithms using treewidth

Treewidth is a measure of "how treelike the graph is."

We need only the following basic facts:

Treewidth

1. If a graph G has treewidth k, then many classical NP-hard problems can be solved in time $2^{O(k)} \cdot n^{O(1)}$ or $2^{O(k \log k)} \cdot n^{O(1)}$ on G.
2. A planar graph on n vertices has treewidth $O(\sqrt{n})$.

Treewidth — a measure of “tree-likeness”

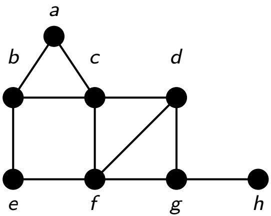
Treewidth — a measure of "tree-likeness"

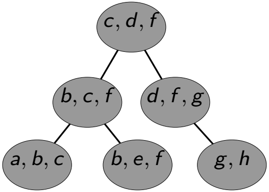
Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.





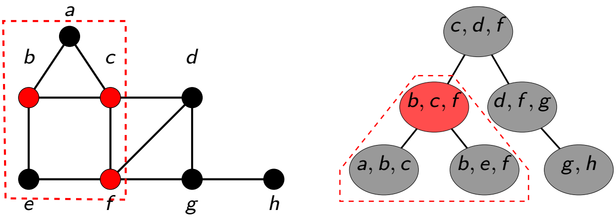
Treewidth — a measure of "tree-likeness"

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.



A subtree communicates with the outside world only via the root of the subtree.

Subexponential algorithm for 3-Coloring

Theorem [textbook dynamic programming]

3-Coloring can be solved in time $2^{O(w)} \cdot n^{O(1)}$ on graphs of treewidth w.

$+$

Theorem [Robertson and Seymour]

A planar graph on n vertices has treewidth $O(\sqrt{n})$.

Subexponential algorithm for 3-Coloring

Theorem [textbook dynamic programming]

3-Coloring can be solved in time $2^{O(w)} \cdot n^{O(1)}$ on graphs of treewidth w.

$$+$$

Theorem [Robertson and Seymour]

A planar graph on n vertices has treewidth $O(\sqrt{n})$.

$$\Downarrow$$

Corollary

3-Coloring can be solved in time $2^{O(\sqrt{n})}$ on planar graphs.

textbook algorithm + combinatorial bound

$\Downarrow$

subexponential algorithm

Lower bounds

Corollary

3-Coloring can be solved in time $2^{O(\sqrt{n})}$ on planar graphs.

Two natural questions:

- Can we achieve this running time on general graphs?
- Can we achieve even better running time (e.g., $2^{O(\sqrt[3]{n})}$) on planar graphs?

Lower bounds based on ETH

ETH + Sparsification Lemma

There is no $2^{o(m)}$-time algorithm for m-clause 3SAT.

The textbook reduction from 3SAT to 3-COLORING:

3SAT formula ϕ n variables m clauses	$\Rightarrow$	Graph G $O(n+m)$ vertices $O(n+m)$ edges

Corollary

Assuming ETH, there is no $2^{o(n)}$ algorithm for 3-COLORING on an n-vertex graph G.

Lower bounds based on ETH

ETH + Sparsification Lemma

There is no $2^{o(m)}$-time algorithm for m-clause 3SAT.

The textbook reduction from 3SAT to 3-Coloring:

3SAT formula ϕ		Graph G
n variables	$\Rightarrow$	$O(m)$ vertices
m clauses		$O(m)$ edges

Corollary

Assuming ETH, there is no $2^{o(n)}$ algorithm for 3-Coloring on an n-vertex graph G.

Transfering bounds

There are polynomial-time reductions from, say, 3-Coloring to many other problems such that the reduction increases the number of vertices by at most a constant factor.

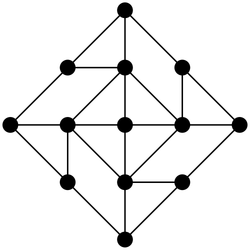
Consequence: Assuming ETH, there is no $2^{o(n)}$ time algorithm on n-vertex graphs for

- Independent Set
- Clique
- Dominating Set
- Vertex Cover
- Hamiltonian Path
- Feedback Vertex Set
- ...

Lower bounds based on ETH

What about 3-Coloring on planar graphs?

The textbook reduction from 3-Coloring to Planar 3-Coloring uses a "crossover gadget" with 4 external connectors:

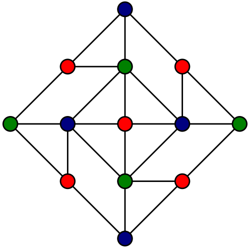


- In every 3-coloring of the gadget, opposite external connectors have the same color.
- Every coloring of the external connectors where the opposite vertices have the same color can be extended to the whole gadget.
- If two edges cross, replace them with a crossover gadget.

Lower bounds based on ETH

What about 3-Coloring on planar graphs?

The textbook reduction from 3-Coloring to Planar 3-Coloring uses a "crossover gadget" with 4 external connectors:

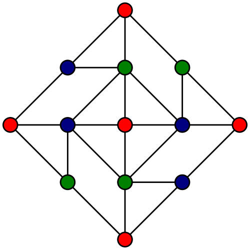


- In every 3-coloring of the gadget, opposite external connectors have the same color.
- Every coloring of the external connectors where the opposite vertices have the same color can be extended to the whole gadget.
- If two edges cross, replace them with a crossover gadget.

Lower bounds based on ETH

What about 3-Coloring on planar graphs?

The textbook reduction from 3-Coloring to Planar 3-Coloring uses a "crossover gadget" with 4 external connectors:



- In every 3-coloring of the gadget, opposite external connectors have the same color.
- Every coloring of the external connectors where the opposite vertices have the same color can be extended to the whole gadget.
- If two edges cross, replace them with a crossover gadget.

Lower bounds based on ETH

- The reduction from 3-COLORING to PLANAR 3-COLORING introduces $O(1)$ new edges/vertices for each crossing.
- A graph with m edges can be drawn with $O(m^2)$ crossings.

3SAT formula ϕ		Graph G		Planar graph G'
n variables	$\Rightarrow$	$O(m)$ vertices	$\Rightarrow$	$O(m^2)$ vertices
m clauses		$O(m)$ edges		$O(m^2)$ edges

Corollary

Assuming ETH, there is no $2^{o(\sqrt{n})}$ algorithm for 3-COLORING on an n-vertex planar graph G.

(Essentially observed by [Cai and Juedes 2001])

Lower bounds for planar problems

Consequence: Assuming ETH, there is no $2^{o(\sqrt{n})}$ time algorithm on n-vertex **planar graphs** for

- Independent Set
- Dominating Set
- Vertex Cover
- Hamiltonian Path
- Feedback Vertex Set
- ...

Lower bounds for planar problems

Consequence: Assuming ETH, there is no $2^{o(\sqrt{k})} \cdot n^{O(1)}$ time algorithm on **planar graphs** for

- k-Independent Set
- k-Dominating Set
- k-Vertex Cover
- k-Path
- k-Feedback Vertex Set
- ...

Summary of Chapter 1

Streamlined way of obtaining tight upper and lower bounds for planar problems.

- **Upper bound:**
  Standard bounded-treewidth algorithm + treewidth bound on planar graphs give $2^{O(\sqrt{n})}$ time subexponential algorithms.
- **Lower bound:**
  Textbook NP-hardness proof with quadratic blow up + ETH rule out $2^{o(\sqrt{n})}$ algorithms.

Works for Hamiltonian Cycle, Vertex Cover, Independent Set, Feedback Vertex Set, Dominating Set, Steiner Tree, ...

Chapter 2: Bidimensionality

Bidimensionality theory [Demaine, Fomin, Hajiaghayi, Thilikos 2005] gives very elegant subexponential algorithms on planar graphs for parameterized problems such as

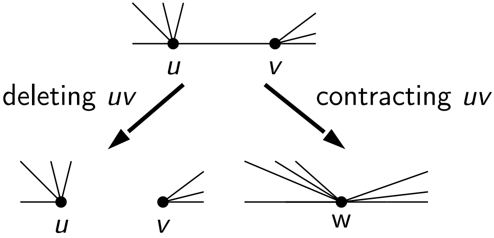
- k-Path
- Vertex Cover
- Feedback Vertex Set
- Independent Set
- Dominating Set

We already know that (assuming ETH), there are no $2^{o(\sqrt{k})} \cdot n^{O(1)}$ time algorithms for these problems.

Minors

Definition

Graph H is a **minor** of G ($H \leq G$) if H can be obtained from G by deleting edges, deleting vertices, and contracting edges.

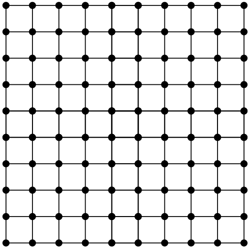


Note: length of the longest path in H is at most the length of the longest path in G.

Planar Excluded Grid Theorem

Theorem [Robertson, Seymour, Thomas 1994]

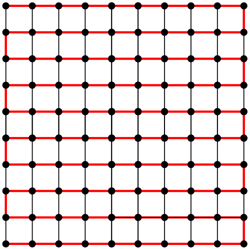
Every planar graph with treewidth at least $5k$ has a $k \times k$ grid minor.



Note: for general graphs, treewidth at least k^{100} or so guarantees a $k \times k$ grid minor [Chekuri and Chuzhoy 2013]!

Bidimensionality for k-Path

Observation: If the treewidth of a planar graph G is at least $5\sqrt{k}$
$\Rightarrow$ It has a $\sqrt{k} \times \sqrt{k}$ grid minor (Planar Excluded Grid Theorem)
$\Rightarrow$ The grid has a path of length at least k.
$\Rightarrow$ G has a path of length at least k.

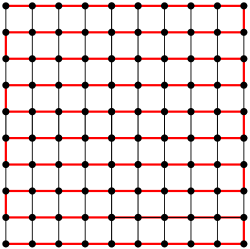


Bidimensionality for k-PATH

Observation: If the treewidth of a planar graph G is at least $5\sqrt{k}$
$\Rightarrow$ It has a $\sqrt{k} \times \sqrt{k}$ grid minor (Planar Excluded Grid Theorem)
$\Rightarrow$ The grid has a path of length at least k.
$\Rightarrow$ G has a path of length at least k.

We use this observation to find a path of length at least k on planar graphs:

- Set $w := 5\sqrt{k}$.
- Find an $O(1)$-approximate tree decomposition.
  - If treewidth is at least w: we answer "there is a path of length at least k."
  - If we get a tree decomposition of width $O(w)$, then we can solve the problem in time $2^{O(w \log w)} \cdot n^{O(1)} = 2^{O(\sqrt{k} \log k)} \cdot n^{O(1)}$.

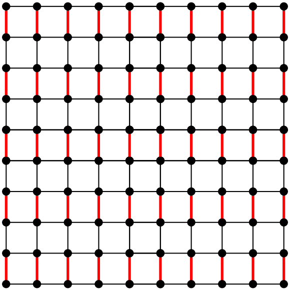


Bidimensionality

Definition

A graph invariant $x(G)$ is **minor-bidimensional** if

- $x(G') \leq x(G)$ for every minor G' of G, and
- If G_k is the $k \times k$ grid, then $x(G_k) \geq ck^2$ (for some constant $c > 0$).



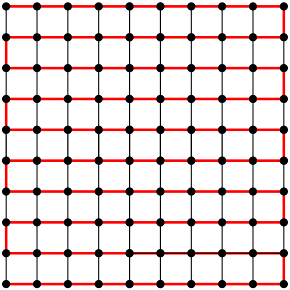
Examples: minimum vertex cover, length of the longest path, feedback vertex set are minor-bidimensional.

Bidimensionality

Definition

A graph invariant $x(G)$ is **minor-bidimensional** if

- $x(G') \leq x(G)$ for every minor G' of G, and
- If G_k is the $k \times k$ grid, then $x(G_k) \geq ck^2$ (for some constant $c > 0$).



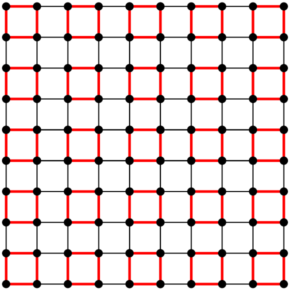
Examples: minimum vertex cover, length of the longest path, feedback vertex set are minor-bidimensional.

Bidimensionality

Definition

A graph invariant $x(G)$ is **minor-bidimensional** if

- $x(G') \leq x(G)$ for every minor G' of G, and
- If G_k is the $k \times k$ grid, then $x(G_k) \geq ck^2$ (for some constant $c > 0$).



Examples: minimum vertex cover, length of the longest path, feedback vertex set are minor-bidimensional.

Summary of Chapter 2

Tight bounds for minor-bidimensional planar problems.

- **Upper bound:**
  Standard bounded-treewidth algorithm + planar excluded grid theorem give $2^{O(\sqrt{k})} \cdot n^{O(1)}$ time FPT algorithms.
- **Lower bound:**
  Textbook NP-hardness proof with quadratic blow up + ETH rule out $2^{o(\sqrt{n})}$ time algorithms $\Rightarrow$ no $2^{o(\sqrt{k})} \cdot n^{O(1)}$ time algorithm.

Variant of theory works for **contraction-bidimensional** problems, e.g., Independent Set, Dominating Set.

Chapter 3: Finding bounded-treewidth solutions

So far, we have exploited that the **input** has bounded treewidth and used standard algorithms.

Chapter 3: Finding bounded-treewidth solutions

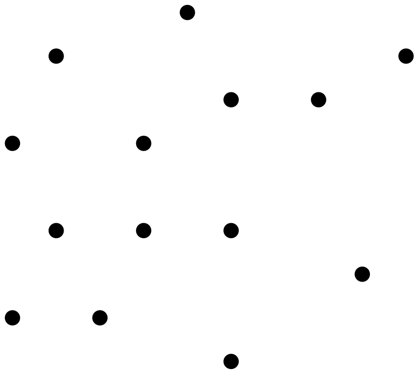
So far, we have exploited that the **input** has bounded treewidth and used standard algorithms.

Change of viewpoint:

In many cases, we have to exploit instead that the **solution** has bounded treewidth.

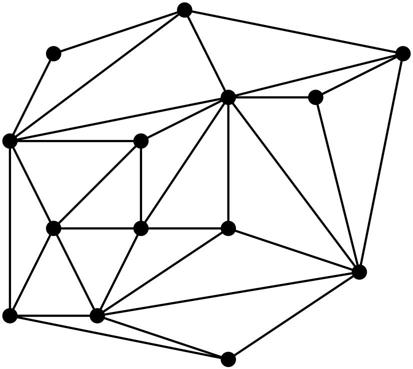
Minimum Weight Triangulation

Given a set of n points in the plane, find a triangulation of minimum length.



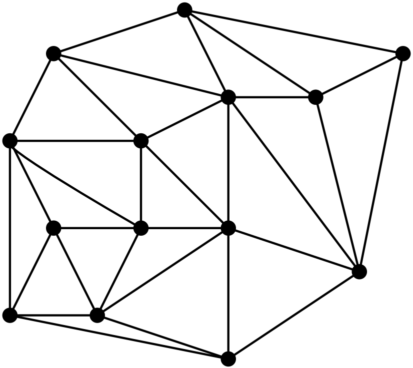
Minimum Weight Triangulation

Given a set of n points in the plane, find a triangulation of minimum length.



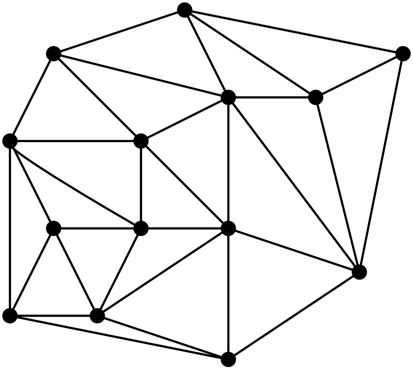
Minimum Weight Triangulation

Given a set of n points in the plane, find a triangulation of minimum length.



Minimum Weight Triangulation

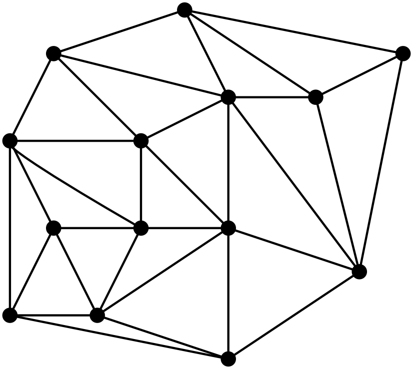
Given a set of n points in the plane, find a triangulation of minimum length.



Brute force solution: $2^{O(n)}$ time.

Minimum Weight Triangulation

Given a set of n points in the plane, find a triangulation of minimum length.



Theorem [Lingas 1998], [Knauer 2006]

Minimum Weight Triangulation can be solved in time $2^{O(\sqrt{n}\log n)}$.

Lower bound

Theorem [Mulzer and Rote 2006]

Minimum Weight Triangulation is NP-hard.

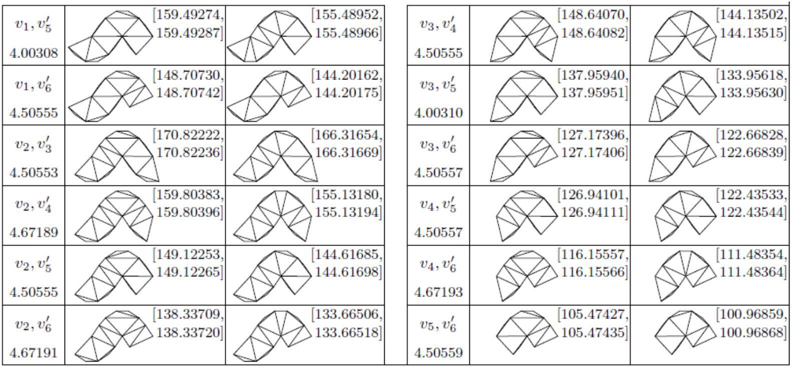
(solving a long-standing open problem of [Garey and Johnson 1979])

Lower bound

Theorem [Mulzer and Rote 2006]

Minimum Weight Triangulation is NP-hard.

(solving a long-standing open problem of [Garey and Johnson 1979])



v_1, v_5' 4.00308	[159.49274, 159.49287]	[155.48952, 155.48966]	v_3, v_4' 4.50555	[148.64070, 148.64082]	[144.13502, 144.13515]
v_1, v_6' 4.50555	[148.70730, 148.70742]	[144.20162, 144.20175]	v_3, v_5' 4.00310	[137.95940, 137.95951]	[133.95618, 133.95630]
v_2, v_3' 4.50553	[170.82222, 170.82236]	[166.31654, 166.31669]	v_3, v_6' 4.50557	[127.17396, 127.17406]	[122.66828, 122.66839]
v_2, v_4' 4.67189	[159.80383, 159.80396]	[155.13180, 155.13194]	v_4, v_5' 4.50557	[126.94101, 126.94111]	[122.43533, 122.43544]
v_2, v_5' 4.50555	[149.12253, 149.12265]	[144.61685, 144.61698]	v_4, v_6' 4.67193	[116.15557, 116.15566]	[111.48354, 111.48364]
v_2, v_6' 4.67191	[138.33709, 138.33720]	[133.66506, 133.66518]	v_5, v_6' 4.50559	[105.47427, 105.47435]	[100.96859, 100.96868]

Not for the fainthearted. . .

Lower bound

Theorem [Mulzer and Rote 2006]

Minimum Weight Triangulation is NP-hard.

(solving a long-standing open problem of [Garey and Johnson 1979])

It can be checked that the proof also implies:

Theorem [Mulzer and Rote 2006]

Assuming ETH, Minimum Weight Triangulation cannot be solved in time $2^{o(\sqrt{n})}$.

Main paradigm

Exploit that the **solution** has treewidth $O(\sqrt{n})$ and has separators of size $O(\sqrt{n})$.

Counting problems

Counting is harder than decision:

- Counting version of easy problems:
  not clear if they remain easy.
- Counting version of hard problems:
  not clear if we can keep the same running time.

Counting problems

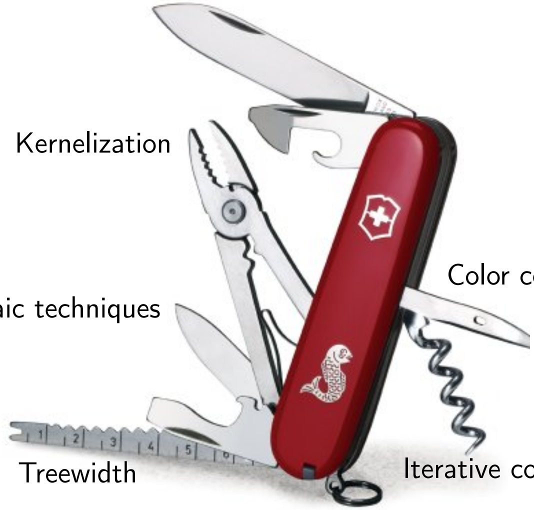
Counting is harder than decision:

- Counting version of easy problems:
  not clear if they remain easy.
- Counting version of hard problems:
  not clear if we can keep the same running time.

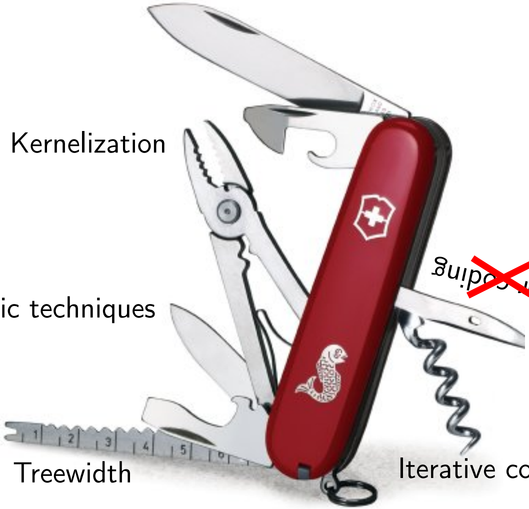
Working on counting problems is fun:

- You can revisit fundamental, "well-understood" problems.
- Requires a new set of lower bound techniques.
- Requires new algorithmic techniques.

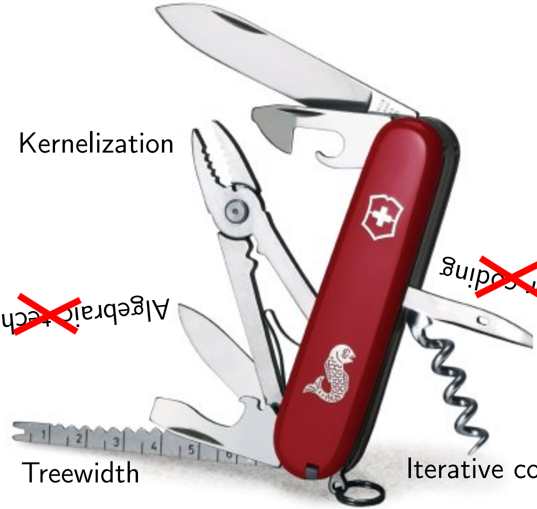
FPT techniques



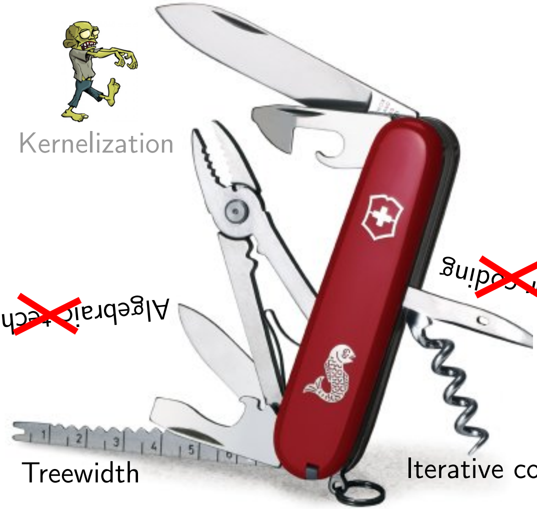
FPT techniques . . . for counting?



FPT techniques . . . for counting?



FPT techniques . . . for counting?



Counting Triangulations

Natural idea:
Guess size-$O(\sqrt{n})$ separator of the triangulation, solve the two subproblems, multiply the number of solutions in the two subproblems.

Counting Triangulations

Natural idea:
Guess size-$O(\sqrt{n})$ separator of the triangulation, solve the two subproblems, multiply the number of solutions in the two subproblems.

Does not work:
More than one separator could be valid for a triangulation
$\Rightarrow$ we can signifcantly overcount the number of triangulations.

Counting Triangulations

Natural idea:
Guess size-$O(\sqrt{n})$ separator of the triangulation, solve the two subproblems, multiply the number of solutions in the two subproblems.

Does not work:
More than one separator could be valid for a triangulation
$\Rightarrow$ we can signifcantly overcount the number of triangulations.

Theorem [M. and Miltzow 2015+]

The number of triangulations can be counted in time $2^{O(\sqrt{n}\log n)}$.

Use canonical separators and enforce that they are canonical in the triangulation.

What do we know about a matching lower bound?

Lower bounds, anyone?

Seems challenging: we need a *counting complexity* lower bound for a *delicate geometric problem.*

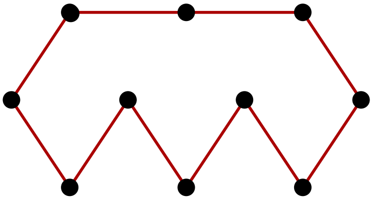
Related lower bounds:

- Finding a restricted triangulation (only a given list of pairs of points can be connected) is NP-hard, and there is no $2^{o(\sqrt{n})}$ time algorithm, assuming ETH.
  [Lloyd 1977], [Schulz 2006].
- Minimum Weight Triangulation is NP-hard.
  [Mulzer and Rote 2006]

TSP

TSP

Input: A set T of cities and a distance function d on T
Output: A tour on T with minimum total distance



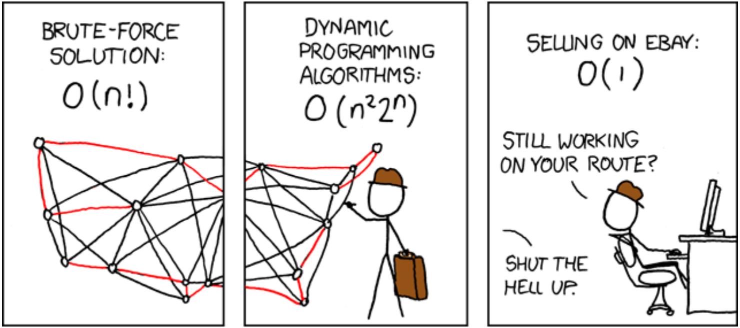
Theorem [Held and Karp 1962]

TSP with n cities can be solved in time $O(2^n \cdot n^2)$.

TSP

TSP

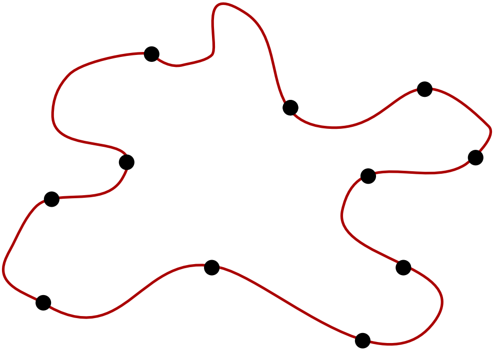
Input:	A set T of cities and a distance function d on T
Output:	A tour on T with minimum total distance



http://xkcd.com/399/

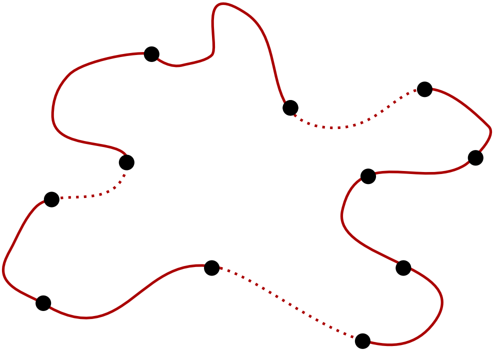
c-change TSP

- c-change operation: removing c steps of the tour and connecting the resulting c paths in some other way.
- A solution is c-change-OPT if no c-change can improve it.
- We can find a c-change-OPT solution in $n^{O(c)} \cdot D$ time, where D is the maximum (integer) distance.



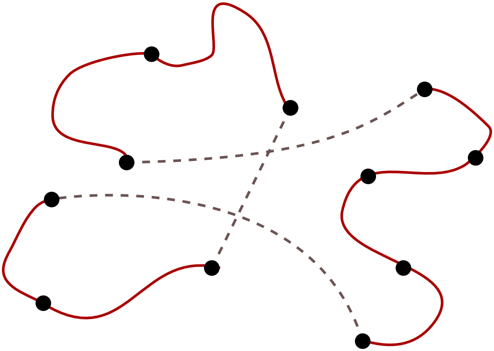
c-change TSP

- c-change operation: removing c steps of the tour and connecting the resulting c paths in some other way.
- A solution is c-change-OPT if no c-change can improve it.
- We can find a c-change-OPT solution in $n^{O(c)} \cdot D$ time, where D is the maximum (integer) distance.



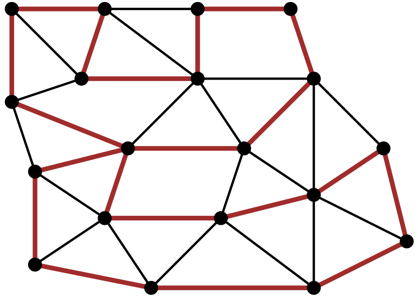
c-change TSP

- c-change operation: removing c steps of the tour and connecting the resulting c paths in some other way.
- A solution is c-change-OPT if no c-change can improve it.
- We can find a c-change-OPT solution in $n^{O(c)} \cdot D$ time, where D is the maximum (integer) distance.



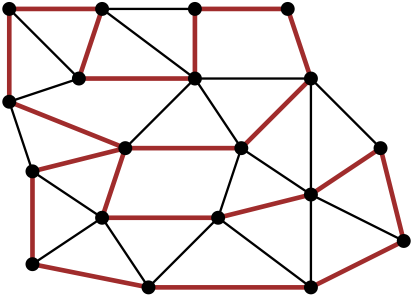
TSP on planar graphs

Assume that the cities correspond to the set of all vertices of a (weighted) planar graph and distance is measured in this (weighted) planar graph.



TSP on planar graphs

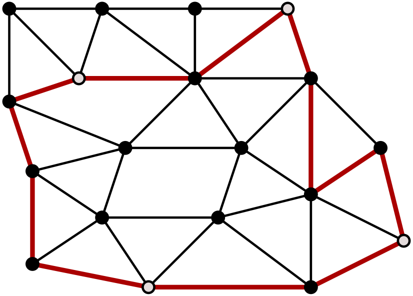
Assume that the cities correspond to the set of all vertices of a (weighted) planar graph and distance is measured in this (weighted) planar graph.



- Can be solved in time $n^{O(\sqrt{n})}$.
- Assuming ETH, no $2^{o(\sqrt{n})}$ time algorithm.

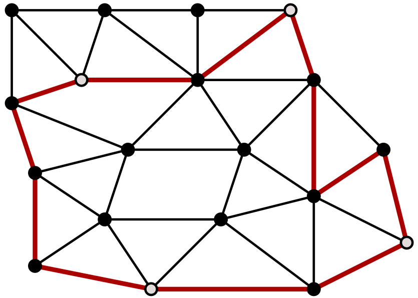
Subset TSP on planar graphs

Assume that the cities correspond to a subset T of vertices of a planar graph and distance is measured in this planar graph.



Subset TSP on planar graphs

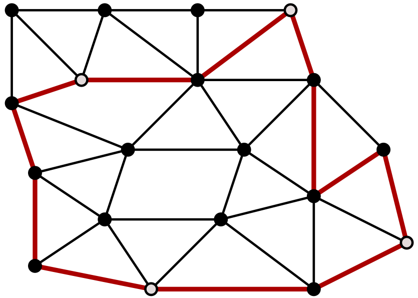
Assume that the cities correspond to a subset T of vertices of a planar graph and distance is measured in this planar graph.



- Can be solved in time $n^{O(\sqrt{n})}$.
- Can be solved in time $2^k \cdot n^{O(1)}$.
- **Question:** Can we restrict the exponential dependence to k **and** exploit planarity?

Subset TSP on planar graphs

Assume that the cities correspond to a subset T of vertices of a planar graph and distance is measured in this planar graph.

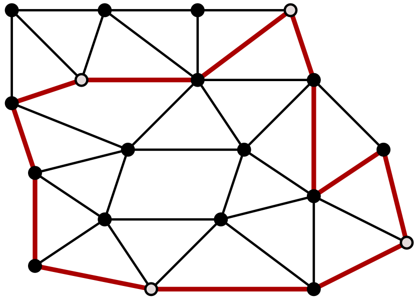


Theorem [Klein and M. 2014]

Subset TSP for k cities in a unit-weight planar graph can be solved in time $2^{O(\sqrt{k}\log k)} \cdot n^{O(1)}$.

Subset TSP on planar graphs

Assume that the cities correspond to a subset T of vertices of a planar graph and distance is measured in this planar graph.



Theorem [Klein and M. 2014]

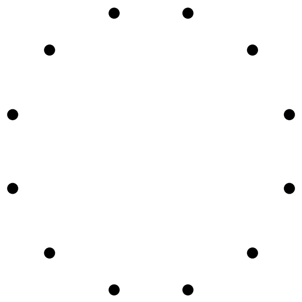
Subset TSP for k cities in a weighted planar graph can be solved in time $(2^{O(\sqrt{k}\log k)} + W) \cdot n^{O(1)}$ if the weights are integers not more than W.

Main paradigm

Exploit that the **solution** has treewidth $O(\sqrt{k})$ and has separators of size $O(\sqrt{k})$.

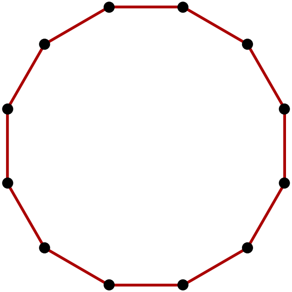
The treewidth bound

Can we bound the treewidth of the solution by $O(\sqrt{k})$?



The treewidth bound

Can we bound the treewidth of the solution by $O(\sqrt{k})$?

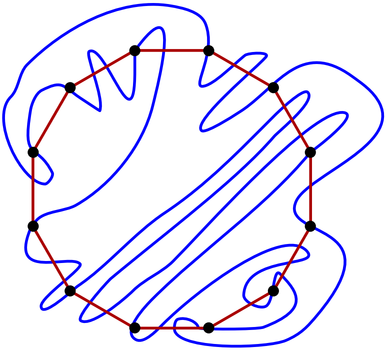


The treewidth of the solution is of course 2.

??? Does not seem to be very insightful.

The treewidth bound

Can we bound the treewidth of the solution by $O(\sqrt{k})$?

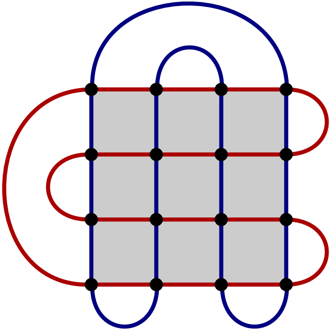


Lemma

For every 4-change-OPT solution, there is an optimum solution such that their union has treewidth $O(\sqrt{k})$.

Proof idea

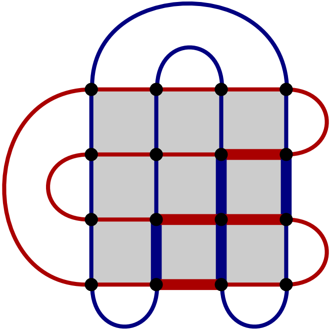
To prove that treewidth of the union is $O(\sqrt{k})$, we mostly need to show that the union has $O(\sqrt{k})$ faces.



Crucial point: there are not too many red-blue-red-blue faces of length 4, because such they cannot form 4×4 grids.

Proof idea

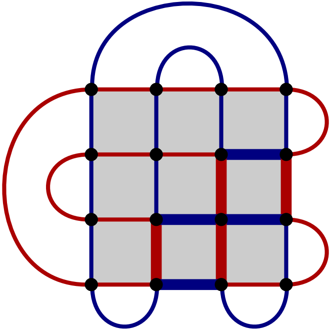
To prove that treewidth of the union is $O(\sqrt{k})$, we mostly need to show that the union has $O(\sqrt{k})$ faces.



- Let us exchange these two sets of edges between the two tours.

Proof idea

To prove that treewidth of the union is $O(\sqrt{k})$, we mostly need to show that the union has $O(\sqrt{k})$ faces.

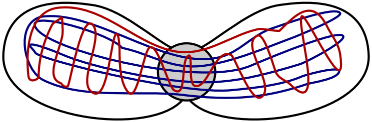


- Let us exchange these two sets of edges between the two tours.
- The 4-change-OPT tour cannot improve.
- The optimum tour cannot improve.
- We get another optimum tour that has fewer crossings with the 4-change-OPT tour.

Using the treewidth bound

Lemma

For every 4-change-OPT solution, there is an optimum solution such that their union has treewidth $O(\sqrt{k})$.



- The union has separators of size $O(\sqrt{k})$.
- In each component, the set of cities visited by the optimum solution is nice: it is the same as what $O(\sqrt{k})$ segments of the 4-change-OPT tour visited.
- Define subproblems based on visiting cities on the union of $O(\sqrt{k})$ segments 4-change-OPT tour.

W[1]-hard problems

- W[1]-hard problems probably have no $f(k)n^{O(1)}$ algorithms.
- Many of them can be solved in $n^{O(k)}$ time.
- For many of them, there is no $f(k)n^{o(k)}$ time algorithm on **general** graphs (assuming ETH).
- For those problems that remain W[1]-hard on **planar** graphs, can we improve the running time to $n^{o(k)}$?

Grid Tiling

GRID TILING

Input: A $k \times k$ matrix and a set of pairs $S_{i,j} \subseteq [D] \times [D]$ for each cell.

Find: A pair $s_{i,j} \in S_{i,j}$ for each cell such that

- Vertical neighbors agree in the 1st coordinate.
- Horizontal neighbors agree in the 2nd coordinate.

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

$k = 3$, $D = 5$

Grid Tiling

Grid Tiling

Input: A $k \times k$ matrix and a set of pairs $S_{i,j} \subseteq [D] \times [D]$ for each cell.

Find: A pair $s_{i,j} \in S_{i,j}$ for each cell such that

- Vertical neighbors agree in the 1st coordinate.
- Horizontal neighbors agree in the 2nd coordinate.

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

$k = 3$, $D = 5$

Grid Tiling

GRID TILING

Input: A $k \times k$ matrix and a set of pairs $S_{i,j} \subseteq [D] \times [D]$ for each cell.

Find: A pair $s_{i,j} \in S_{i,j}$ for each cell such that

- Vertical neighbors agree in the 1st coordinate.
- Horizontal neighbors agree in the 2nd coordinate.

Simple proof:

Fact

There is a parameterized reduction from k-CLIQUE to $k \times k$ GRID TILING.

Grid Tiling and planar problems

Theorem

$k \times k$ Grid Tiling is W[1]-hard and, assuming ETH, cannot be solved in time $f(k)n^{o(k)}$ for any function f.

This lower bound is the key for proving hardness results for planar graphs.

Examples:

- Multiway Cut on planar graphs with k terminals
- Independent Set for unit disks
- Strongly Connected Steiner Subgraph on planar graphs
- Scattered Set on planar graphs

Grid Tiling with $\leq$

Grid Tiling with $\leq$

Input: A $k \times k$ matrix and a set of pairs $S_{i,j} \subseteq [D] \times [D]$ for each cell.

Find: A pair $s_{i,j} \in S_{i,j}$ for each cell such that

- 1st coordinate of $s_{i,j} \leq$ 1st coordinate of $s_{i+1,j}$.
- 2nd coordinate of $s_{i,j} \leq$ 2nd coordinate of $s_{i,j+1}$.

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

$k = 3,\ D = 5$

Grid Tiling with $\leq$

GRID TILING WITH $\leq$

Input: A $k \times k$ matrix and a set of pairs $S_{i,j} \subseteq [D] \times [D]$ for each cell.

Find: A pair $s_{i,j} \in S_{i,j}$ for each cell such that

- 1st coordinate of $s_{i,j} \leq$ 1st coordinate of $s_{i+1,j}$.
- 2nd coordinate of $s_{i,j} \leq$ 2nd coordinate of $s_{i,j+1}$.

Variant of the previous proof:

Theorem

There is a parameterized reduction from $k \times k$-GRID TILING to $O(k) \times O(k)$ GRID TILING WITH $\leq$.

Very useful starting point for geometric problems!

Independent Set for unit disks

Theorem [Alber and Fiala 2004]

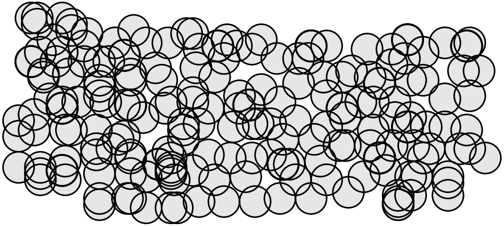
The INDEPENDENT SET problem for unit (diameter) disks can be solved in time $n^{O(\sqrt{k})}$.

Complicated proof using a geometric separator theorem, simple proof by shifting.

Independent Set for unit disks

Theorem [Alber and Fiala 2004]

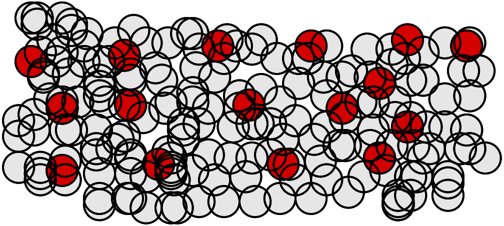
The Independent Set problem for unit (diameter) disks can be solved in time $n^{O(\sqrt{k})}$.



Independent Set for unit disks

Theorem [Alber and Fiala 2004]

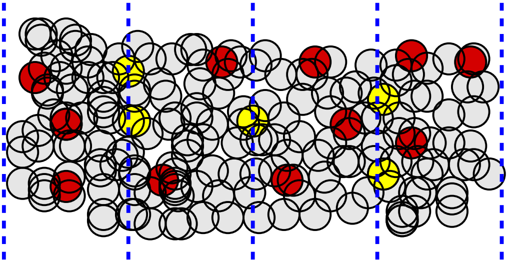
The INDEPENDENT SET problem for unit (diameter) disks can be solved in time $n^{O(\sqrt{k})}$.



Independent Set for unit disks

Theorem [Alber and Fiala 2004]

The Independent Set problem for unit (diameter) disks can be solved in time $n^{O(\sqrt{k})}$.



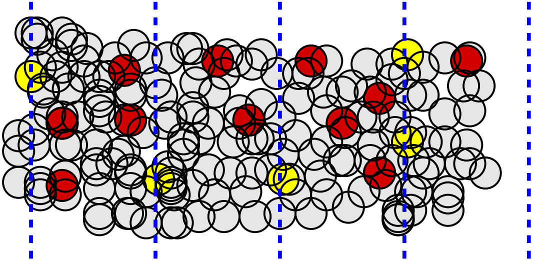
Consider a family of vertical lines at distance $\lfloor\sqrt{k}\rfloor$ from each other, going through $(i, 0)$ for some integer $0 \le i < \lfloor\sqrt{k}\rfloor$.

Claim: Exists i such that the lines hit $O(\sqrt{k})$ disks of the solution.

Independent Set for unit disks

Theorem [Alber and Fiala 2004]

The Independent Set problem for unit (diameter) disks can be solved in time $n^{O(\sqrt{k})}$.



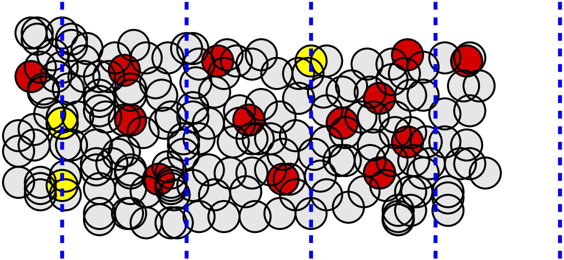
Consider a family of vertical lines at distance $\lfloor\sqrt{k}\rfloor$ from each other, going through $(i, 0)$ for some integer $0 \leq i < \lfloor\sqrt{k}\rfloor$.

Claim: Exists i such that the lines hit $O(\sqrt{k})$ disks of the solution.

Independent Set for unit disks

Theorem [Alber and Fiala 2004]

The INDEPENDENT SET problem for unit (diameter) disks can be solved in time $n^{O(\sqrt{k})}$.



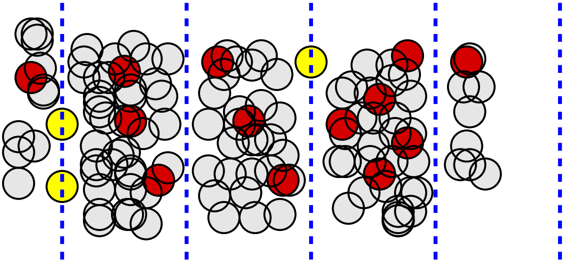
Consider a family of vertical lines at distance $\lfloor\sqrt{k}\rfloor$ from each other, going through $(i, 0)$ for some integer $0 \le i < \lfloor\sqrt{k}\rfloor$.

Claim: Exists i such that the lines hit $O(\sqrt{k})$ disks of the solution.

Independent Set for unit disks

Theorem [Alber and Fiala 2004]

The Independent Set problem for unit (diameter) disks can be solved in time $n^{O(\sqrt{k})}$.



Consider a family of vertical lines at distance $\lfloor\sqrt{k}\rfloor$ from each other, going through $(i,0)$ for some integer $0 \le i < \lfloor\sqrt{k}\rfloor$.

Algorithm: Guess i and the $O(\sqrt{k})$ disks hit by the lines $\Rightarrow$ Remove every disk intersected by the lines or disks $\Rightarrow$ Problem falls apart into strips of height $O(\sqrt{k})$; can be solved optimally in time $n^{O(\sqrt{k})}$.

Independent Set for unit disks

Theorem [Alber and Fiala 2004]

The Independent Set problem for unit (diameter) disks can be solved in time $n^{O(\sqrt{k})}$.

Matching lower bound:

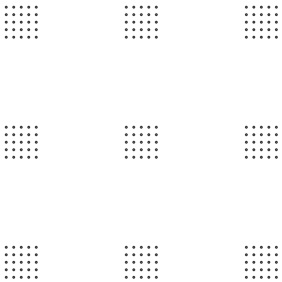
Theorem

There is a reduction from $k \times k$ Grid Tiling with $\leq$ to k^2-Independent Set for unit disks. Consequently, Independent Set for unit disks is

- is W[1]-hard, and
- cannot be solved in time $f(k)n^{o(\sqrt{k})}$ for any function f.

Reduction to unit disks

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

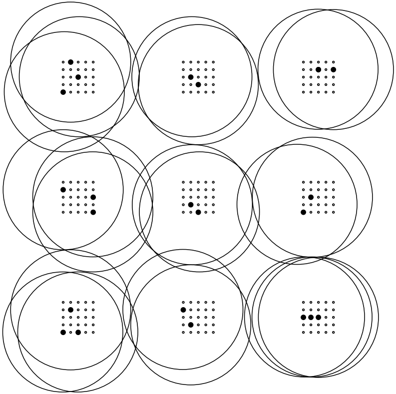


Every pair is represented by a unit disk in the plane.

$\leq$ relation between coordinates $\iff$ disks do not intersect.

Reduction to unit disks

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

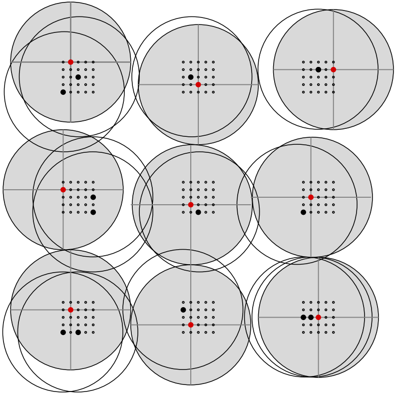


Every pair is represented by a unit disk in the plane.

$\leq$ relation between coordinates $\iff$ disks do not intersect.

Reduction to unit disks

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



Every pair is represented by a unit disk in the plane.

$\leq$ relation between coordinates $\iff$ disks do not intersect.

Challenges

Key idea

We were able to find a separator that hits $O(\sqrt{k})$ disks of the solution and breaks the instance in a nice way.

Two natural directions:

1. Can we solve Independent Set for disks with arbitrary radius in time $n^{O(\sqrt{k})}$?
2. Can we solve Scattered Set (find k vertices that are at distance at least d from each other) on planar graphs in time $n^{O(\sqrt{k})}$, if d is part of the input?

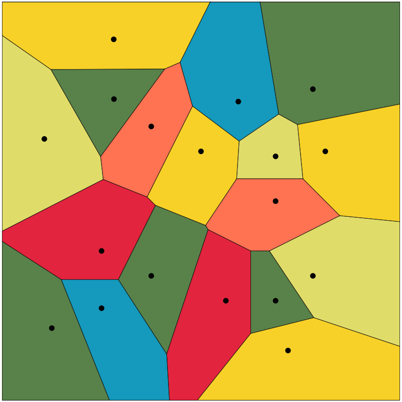
Problem:
The shifting algorithm for unit disks crucially uses the fact that the disks have similar area.

Main paradigm

Exploit that the **solution** has treewidth $O(\sqrt{k})$ and has separators of size $O(\sqrt{k})$.

Voronoi diagrams

Voronoi diagram: we partition the points of the plane according to the closest center.

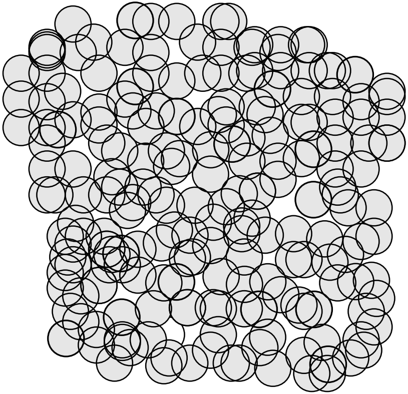


Observation: every cell is convex.

- Assume that the branch points of the diagram have degree 3.
- Ignore what happens at infinity.

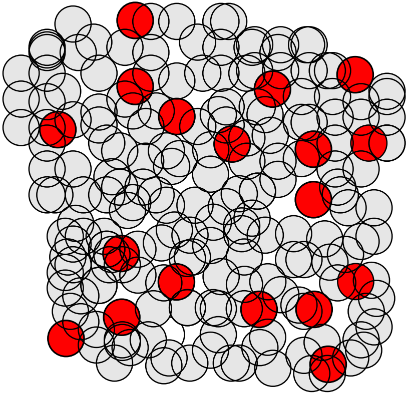
Voronoi separators

Consider the Voronoi diagram of the centers of the solution disks.



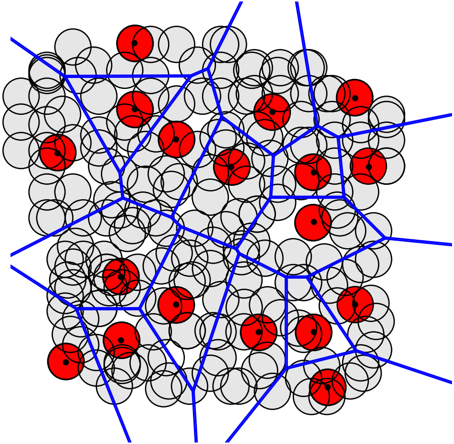
Voronoi separators

Consider the Voronoi diagram of the centers of the solution disks.



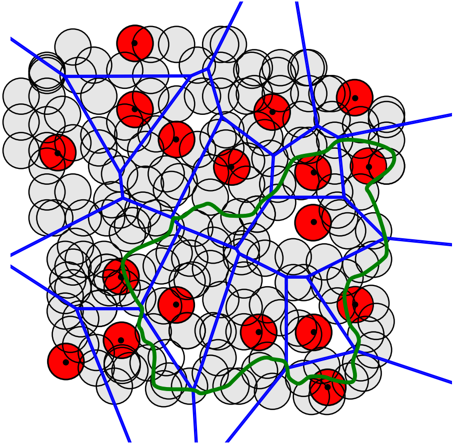
Voronoi separators

Consider the Voronoi diagram of the centers of the solution disks.



Voronoi separators

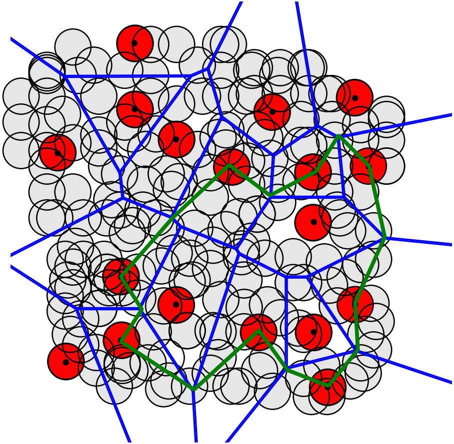
Consider the Voronoi diagram of the centers of the solution disks.



There is a "$\frac{2}{3}$-face-balanced noose" of length $O(\sqrt{k})$.

Voronoi separators

Consider the Voronoi diagram of the centers of the solution disks.

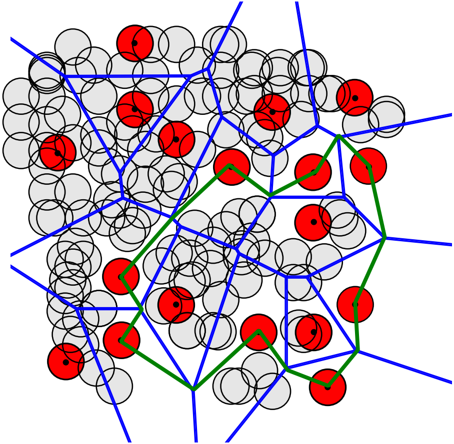


There is a "$\frac{2}{3}$-face-balanced noose" of length $O(\sqrt{k})$.

$\Rightarrow$ There is a corresponding polygon of length $O(\sqrt{k})$.

Voronoi separators

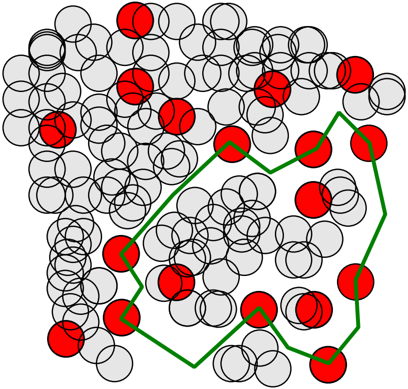
Consider the Voronoi diagram of the centers of the solution disks.



Algorithm: guess $O(\sqrt{k})$ disks and a polygon going through them, remove any disks intersecting the polygon or the guessed disks, recursion on the inside and the outside.

Voronoi separators

Consider the Voronoi diagram of the centers of the solution disks.



Algorithm: guess $O(\sqrt{k})$ disks and a polygon going through them, remove any disks intersecting the polygon or the guessed disks, recursion on the inside and the outside.

Running time

Number of candidate polygons

Number of centers: n.

Potential locations of Voronoi branch points: n^3.

$\Rightarrow$ Number of polygons of length $O(\sqrt{k})$: $n^{O(\sqrt{k})}$.

Recursion

$T(n,k)$: running time with n centers and solution size at most k.

$$T(n,k) = n^{O(\sqrt{k})} T(n, \frac{2}{3}k)$$

Running time

Number of candidate polygons

Number of centers: n.

Potential locations of Voronoi branch points: n^3.

$\Rightarrow$ Number of polygons of length $O(\sqrt{k})$: $n^{O(\sqrt{k})}$.

Recursion

$T(n,k)$: running time with n centers and solution size at most k.

$$\begin{aligned} T(n,k) &= n^{O(\sqrt{k})} T(n, \frac{2}{3}k) \\ &= n^{O(\sqrt{k})} \cdot n^{O(\sqrt{\frac{2}{3}k})} \cdot n^{O(\sqrt{(\frac{2}{3})^2 k})} \cdot n^{O(\sqrt{(\frac{2}{3})^3 k})} \cdots \\ &= n^{O((1+(\frac{2}{3})^{\frac{1}{2}}+(\frac{2}{3})^{\frac{2}{2}}+(\frac{2}{3})^{\frac{3}{2}}+\dots)\sqrt{k})} = n^{O(\sqrt{k})}. \end{aligned}$$

Running time

Number of candidate polygons

Number of centers: n.

Potential locations of Voronoi branch points: n^3.

$\Rightarrow$ Number of polygons of length $O(\sqrt{k})$: $n^{O(\sqrt{k})}$.

Recursion

$T(n,k)$: running time with n centers and solution size at most k.

$$T(n,k) = n^{O(\sqrt{k})}\, T(n, \frac{2}{3}k)$$

$$= n^{O(\sqrt{k})} \cdot n^{O(\sqrt{\frac{2}{3}k})} \cdot n^{O(\sqrt{(\frac{2}{3})^2 k})} \cdot n^{O(\sqrt{(\frac{2}{3})^3 k})} \dots$$

$$= n^{O((1+(\frac{2}{3})^{\frac{1}{2}}+(\frac{2}{3})^{\frac{2}{2}}+(\frac{2}{3})^{\frac{3}{2}}+\dots)\sqrt{k})} = n^{O(\sqrt{k})}.$$

This gives another $n^{O(\sqrt{k})}$ time algorithm for Independent Set for unit disks, **which can now be generalized to disks of arbitrary size and to planar graphs.**

Higher dimensions

Bidimensionalty for planar graphs:

- $2^{O(\sqrt{n})}$, $2^{O(\sqrt{k})} \cdot n^{O(1)}$, $n^{O(\sqrt{k})}$ time algorithms.
- There is no tridimensionalty!

Higher dimensions

Bidimensionality for 2-dimensional geometric problems:

- $2^{O(\sqrt{n})}$, $2^{O(\sqrt{k})} \cdot n^{O(1)}$, $n^{O(\sqrt{k})}$ time algorithms.
- What about higher dimensions?

Higher dimensions

Bidimensionality for 2-dimensional geometric problems:

- $2^{O(\sqrt{n})}$, $2^{O(\sqrt{k})} \cdot n^{O(1)}$, $n^{O(\sqrt{k})}$ time algorithms.
- What about higher dimensions?

"Limited blessing of low dimensionality:"

Theorem

Independent Set for unit spheres in d dimensions can be solved in time $n^{O(k^{1-1/d})}$.

Matching lower bound:

Theorem [M. and Sidiropoulos 2014]

Assuming ETH, Independent Set for unit spheres in d dimensions cannot be solved in time $n^{o(k^{1-1/d})}$.

Higher dimensions

Bidimensionality for 2-dimensional geometric problems:

- $2^{O(\sqrt{n})}$, $2^{O(\sqrt{k})} \cdot n^{O(1)}$, $n^{O(\sqrt{k})}$ time algorithms.
- What about higher dimensions?

"Limited blessing of low dimensionality:"

Theorem [Smith and Wormald 1998]

Euclidean TSP in d dimensions can be solved in time $2^{O(n^{1-1/d+\epsilon})}$.

Matching lower bound:

Theorem [M. and Sidiropoulos 2014]

Assuming ETH, Euclidean TSP in d dimension cannot be solved in time $2^{O(n^{1-1/d-\epsilon})}$ for any $\epsilon > 0$.

Summary of Chapter 3

Parameterized problems where bidimensionality does not work.

- **Upper bounds:**
  Algorithms exploiting that some representation of the solution has bounded treewidth. Treewidth bound is problem-specific:
  - Minimum Weight Triangulation/Counting triangulations: n-vertex triangulation has treewidth $O(\sqrt{n})$.
  - Subset TSP on planar graphs: the union of an optimum solution and a 4-change-OPT solution has treewidth $O(\sqrt{k})$.
  - Independent Set for unit disks: Voronoi diagram of the solution has treewidth $O(\sqrt{k})$.
- **Lower bounds:**
  To rule out $f(k) \cdot n^{o(\sqrt{k})}$ time algorithms for W[1]-hard problems, we have to prove hardness by reduction from Grid Tiling.

Conclusions

- A robust understanding of why certain problems can be solved in time $2^{O(\sqrt{n})}$ etc. on planar graphs and why the square root is best possible.

Conclusions

- A robust understanding of why certain problems can be solved in time $2^{O(\sqrt{n})}$ etc. on planar graphs and why the square root is best possible.
- Going beyond the basic toolbox requires new problem-specific algorithmic techniques and hardness proofs with tricky gadget constructions.

Conclusions

- A robust understanding of why certain problems can be solved in time $2^{O(\sqrt{n})}$ etc. on planar graphs and why the square root is best possible.
- Going beyond the basic toolbox requires new problem-specific algorithmic techniques and hardness proofs with tricky gadget constructions.
- The lower bound technology on planar graphs cannot give a lower bound without a square root factor. Does this mean that there are matching algorithms for other problems as well?
  - $2^{O(\sqrt{k})} \cdot n^{O(1)}$ time algorithm for Steiner Tree with k terminals in a planar graph?
  - $2^{O(\sqrt{k})} \cdot n^{O(1)}$ time algorithm for finding a cycle of length **exactly** k in a planar graph?
  - ...