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

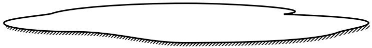
Dagstuhl Seminar 16221:
Algorithms for Optimization Problems
in Planar Graphs

Schloss Dagstuhl, Germany
June 1, 2016

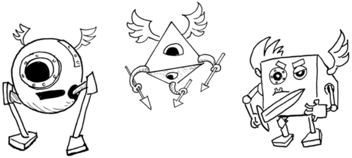
Main message

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:

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

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

Overview

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 $O(\sqrt{n})$.

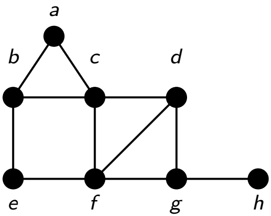
Treewidth — a measure of "tree-likeness"

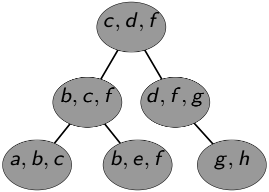
Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

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.





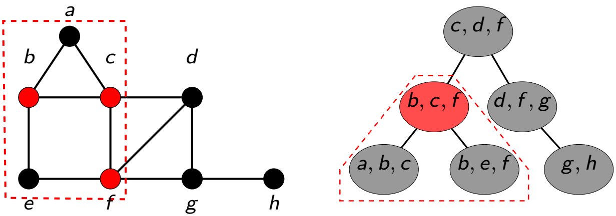
Treewidth — a measure of "tree-likeness"

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.

Finding tree decompositions

Various algorithms for finding optimal or approximate tree decompositions if treewidth is w:

- optimal decomposition in time $2^{O(w^3)} \cdot n$ [Bodlaender 1996].
- 4-approximate decomposition in time $2^{O(w)} \cdot n^2$ [Robertson and Seymour].
- 5-approximate decomposition in time $2^{O(w)} \cdot n$ [Bodlaender et al. 2013].
- $O(\sqrt{\log w})$-approximation in polynomial time [Feige, Hajiaghayi, Lee 2008].

As we are mostly interested in algorithms with running time $2^{O(w)} \cdot n^{O(1)}$, we may assume that we have a decomposition.

3-COLORING and tree decompositions

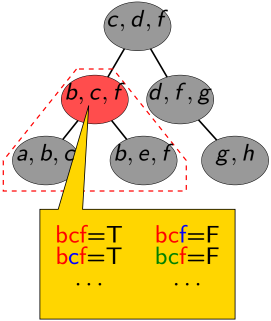
Theorem

Given a tree decomposition of width w, 3-COLORING can be solved in time $3^w \cdot w^{O(1)} \cdot n$.

B_x: vertices appearing in node x.
V_x: vertices appearing in the subtree rooted at x.

For every node x and coloring $c : B_x \to \{1, 2, 3\}$, we compute the Boolean value $E[x, c]$, which is true if and only if c can be extended to a proper 3-coloring of V_x.

Claim:
We can determine $E[x, c]$ if all the values are known for the children of x.



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

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?

P $\neq$ NP is not a sufficiently strong hypothesis: it is compatible with 3SAT being solvable in time $2^{O(n^{1/1000})}$ or even in time $n^{O(\log n)}$.

We need a stronger hypothesis!

Exponential Time Hypothesis (ETH)

Hypothesis introduced by Impagliazzo, Paturi, and Zane:

Exponential Time Hypothesis (ETH) [consequence of]

There is no $2^{o(n)}$-time algorithm for n-variable $\mathrm{3SAT}$.

Note: current best algorithm is 1.30704^n [Hertli 2011].

Note: an n-variable $\mathrm{3SAT}$ formula can have $m = \Omega(n^3)$ clauses.

Exponential Time Hypothesis (ETH)

Hypothesis introduced by Impagliazzo, Paturi, and Zane:

Exponential Time Hypothesis (ETH) [consequence of]

There is no $2^{o(n)}$-time algorithm for n-variable $\mathrm{3SAT}$.

Note: current best algorithm is 1.30704^n [Hertli 2011].

Note: an n-variable $\mathrm{3SAT}$ formula can have $m = \Omega(n^3)$ clauses.

Are there algorithms that are subexponential in the size $n + m$ of the $\mathrm{3SAT}$ formula?

Sparsification Lemma [Impagliazzo, Paturi, Zane 2001]

There is a $2^{o(n)}$-time algorithm for n-variable $\mathrm{3SAT}$.

$$\Updownarrow$$

There is a $2^{o(n+m)}$-time algorithm for n-variable m-clause $\mathrm{3SAT}$.

Lower bounds based on ETH

ETH + Sparsification Lemma

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

The textbook reduction from 3SAT to 3-Coloring:

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

(we can assume $n = O(m)$)

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.

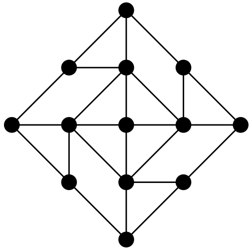
Consequence: Assuming ETH, there is no $2^{o(n)}$ time algorithm on n-vertex graphs for

- 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:

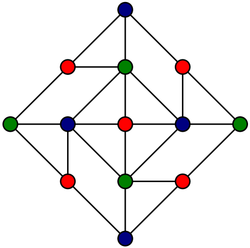


- In every 3-coloring of the gadget, opposite external connectors have the same color.
- Every coloring of the external connectors where the opposite vertices have the same color can be extended to all the gadget.
- If two edges cross, replace them with a crossover gadget.

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:

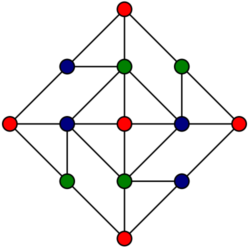


- In every 3-coloring of the gadget, opposite external connectors have the same color.
- Every coloring of the external connectors where the opposite vertices have the same color can be extended to all the gadget.
- If two edges cross, replace them with a crossover gadget.

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 all the 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 n variables m clauses	$\Rightarrow$	Graph G $O(m)$ vertices $O(m)$ edges	$\Rightarrow$	Planar graph G' $O(m^2)$ vertices $O(m^2)$ 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])

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: Grid minors and bidimensionality

More refined analysis of the running time: we express the running time as a function of input size n and a parameter k.

Definition

A problem is **fixed-parameter tractable (FPT)** parameterized by k if it can be solved in time $f(k) \cdot n^{O(1)}$ for some computable function f.

Examples of FPT problems:

- Finding a vertex cover of size k.
- Finding a feedback vertex set of size k.
- Finding a path of length k.
- Finding k vertex-disjoint triangles.
- ...

Note: these four problems have $2^{O(k)} \cdot n^{O(1)}$ time algorithms, which is best possible on general graphs.

W[1]-hardness

Negative evidence similar to NP-completeness. If a problem is **W[1]-hard,** then the problem is not FPT unless FPT=W[1].

Some W[1]-hard problems:

- Finding a clique/independent set of size k.
- Finding a dominating set of size k.
- Finding k pairwise disjoint sets.
- ...

For these problems, the exponent of n has to depend on k (the running time is typically $n^{O(k)}$).

Subexponential parameterized algorithms

What kind of upper/lower bounds we have for $f(k)$?

- For most problems, we cannot expect a $2^{o(k)} \cdot n^{O(1)}$ time algorithm on **general graphs.**
  (As this would imply a $2^{o(n)}$ algorithm.)
- For most problems, we cannot expect a $2^{o(\sqrt{k})} \cdot n^{O(1)}$ time algorithm on **planar graphs.**
  (As this would imply a $2^{o(\sqrt{n})}$ algorithm.)

Subexponential parameterized algorithms

What kind of upper/lower bounds we have for $f(k)$?

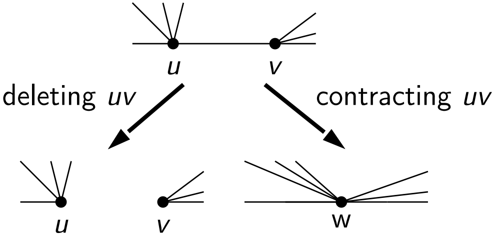
- For most problems, we cannot expect a $2^{o(k)} \cdot n^{O(1)}$ time algorithm on **general graphs.**
  (As this would imply a $2^{o(n)}$ algorithm.)
- For most problems, we cannot expect a $2^{o(\sqrt{k})} \cdot n^{O(1)}$ time algorithm on **planar graphs.**
  (As this would imply a $2^{o(\sqrt{n})}$ algorithm.)
- However, $2^{O(\sqrt{k})} \cdot n^{O(1)}$ algorithms do exist for several problems on planar graphs, even for some W[1]-hard problems.
- Quick proofs via grid minors and bidimensionality.
  [Demaine, Fomin, Hajiaghayi, Thilikos 2004]

Next: subexponential parameterized algorithm for k-PATH.

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.

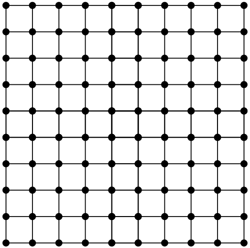


Note: length of the longest path in H is at most the length of the longest path in G.

Planar Excluded Grid Theorem

Theorem [Robertson, Seymour, Thomas 1994]

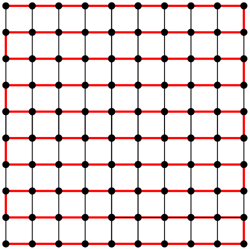
Every planar graph with treewidth at least $5k$ has a $k \times k$ grid minor.



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.

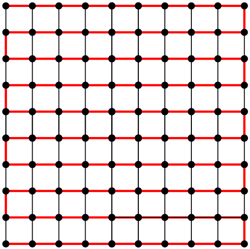


Bidimensionality for k-PATH

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:

- If treewidth w of G is at least $5\sqrt{k}$: we answer "there is a path of length at least k."
- If treewidth w of G is less than $5\sqrt{k}$, then we can solve the problem in time $2^{O(w)} \cdot n^{O(1)} = 2^{O(\sqrt{k})} \cdot n^{O(1)}$.

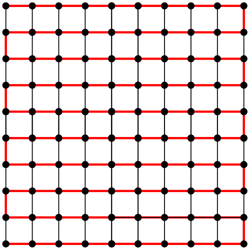


Bidimensionality for k-Path

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 can 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 $2^{O(w)} \cdot n^{O(1)} = 2^{O(\sqrt{k})} \cdot n^{O(1)}$.

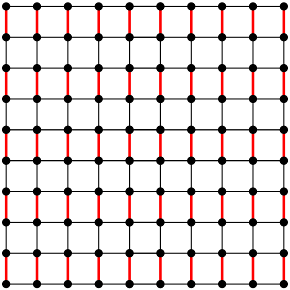


Bidimensionality

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 $c > 0$).



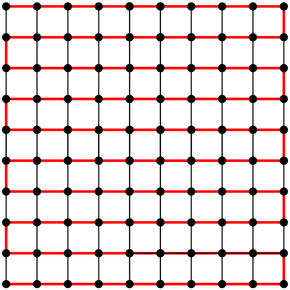
Examples: minimum vertex cover, length of the longest path, feedback vertex set are minor-bidimensional.

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 $c > 0$).



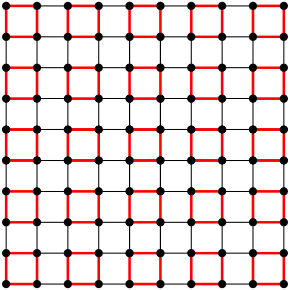
Examples: minimum vertex cover, length of the longest path, feedback vertex set are minor-bidimensional.

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.

Limits of bidimensionality?

Bidimensionality works nice for some problems, but fails completely even for embarrassingly simple generalizations.

- Works for k-Path, but not for $s - t$ paths.
- Works for cycles of length at least k, but not for cycles of length exactly k.
- Weighted versions, colored versions, counting versions, etc.

Bidimensionality on its own does not give subexponential parameterized algorithms for these problems!

Limits of bidimensionality?

Perhaps the most basic problem:

> **Subgraph Isomorphism**
> Given a graphs H and G, decide if G has a subgraph isomorphic to H.

Limits of bidimensionality?

Perhaps the most basic problem:

> **SUBGRAPH ISOMORPHISM**
> Given a graphs H and G, decide if G has a subgraph isomorphic to H.

Theorem [Eppstein 1999]

SUBGRAPH ISOMORPHISM for planar graphs can be solved in time $2^{O(k \log k)} \cdot n$ for $k := |V(H)|$.

Limits of bidimensionality?

Perhaps the most basic problem:

> SUBGRAPH ISOMORPHISM
> Given a graphs H and G, decide if G has a subgraph isomorphic to H.

Question already asked in the last seminar:

> Does the square root phenomenon appear for SUBGRAPH ISOMORPHISM on planar graphs?

Limits of bidimensionality?

Perhaps the most basic problem:

SUBGRAPH ISOMORPHISM Given a graphs H and G, decide if G has a subgraph isomorphic to H.

Question already asked in the last seminar:

Does the square root phenomenon appear for SUBGRAPH ISOMORPHISM on planar graphs?

- Assuming ETH, there is no $2^{o(k/\log k)}n^{O(1)}$ time algorithm for general patterns.

  [Hans Bodlaender's talk Thu 9:30]

- There is a $2^{O(\sqrt{k}\text{polylog}k)}n^{O(1)}$ time (randomized) algorithm for connected, bounded degree patterns.

  [Marcin Pilipczuk's talk Thu 9:00]

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

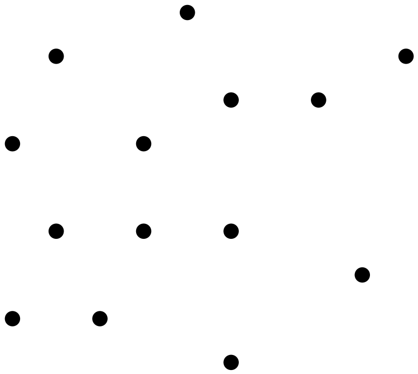
So far, we have exploited that the **input** has bounded treewidth and used standard algorithms.

Change of viewpoint:

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

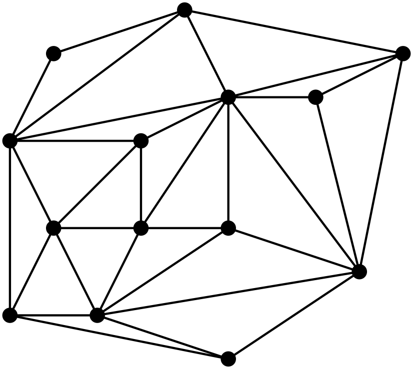
Minimum Weight Triangulation

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



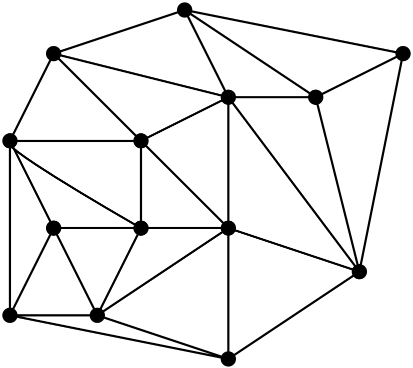
Minimum Weight Triangulation

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



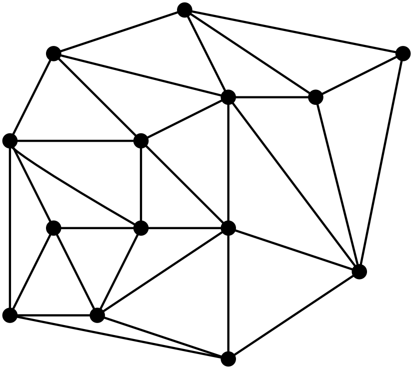
Minimum Weight Triangulation

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



Minimum Weight Triangulation

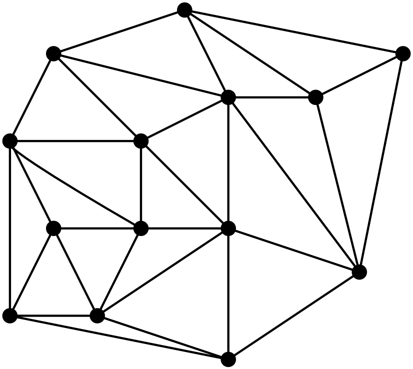
Given a set of n points in the plane, find a triangulation of minimum length.



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.



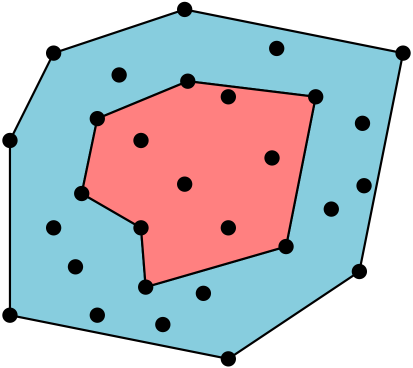
Theorem [Lingas 1998], [Knauer 2006]

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

Minimum Weight Triangulation

Theorem [Lingas 1998], [Knauer 2006]

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

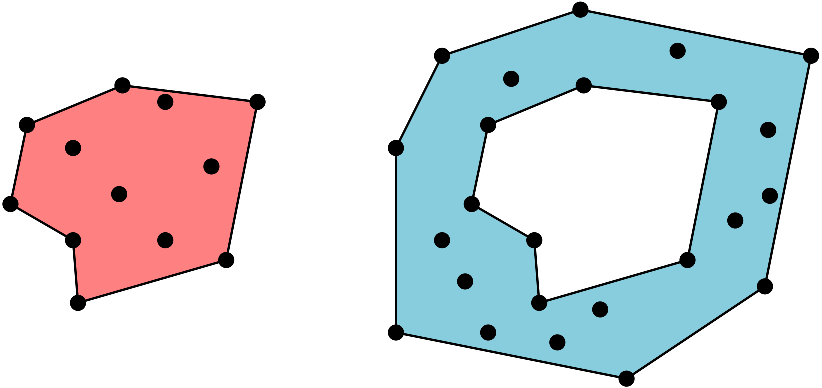


Main idea: guess a separator of size $O(\sqrt{n})$ of the solution and recurse.

Minimum Weight Triangulation

Theorem [Lingas 1998], [Knauer 2006]

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

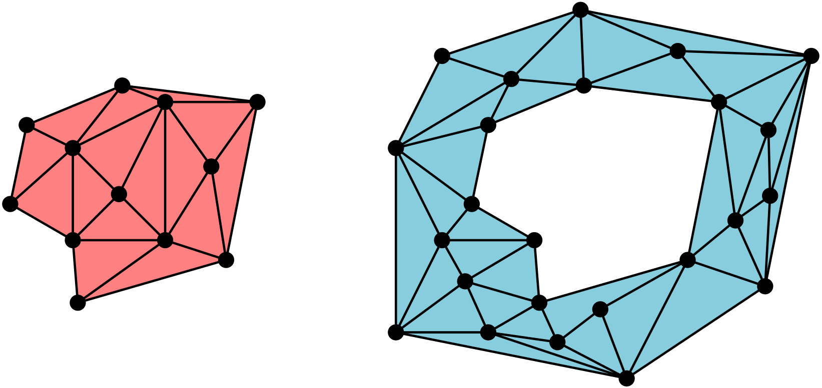


Main idea: guess a separator of size $O(\sqrt{n})$ of the solution and recurse.

Minimum Weight Triangulation

Theorem [Lingas 1998], [Knauer 2006]

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



Main idea: guess a separator of size $O(\sqrt{n})$ of the solution and recurse.

Lower bound

Theorem [Mulzer and Rote 2006]

Minimum Weight Triangulation is NP-hard.

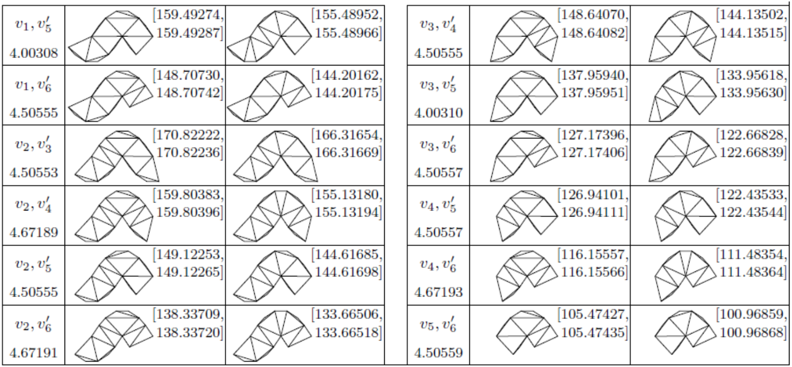
(solving a long-standing open problem of [Garey and Johnson 1979])

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.

Bidimensionality and counting

Does not work for counting k-paths in a planar graph:

- If treewidth w is $O(\sqrt{k})$: can be solved in time $2^{O(w)}n^{O(1)} = 2^{O(\sqrt{k})}n^{O(1)}$ using dynamic programming.
- If treewidth w is larger than $c\sqrt{k}$: answer is positive, but how much exactly?

Bidimensionality and counting

Does not work for counting k-paths in a planar graph:

- If treewidth w is $O(\sqrt{k})$: can be solved in time $2^{O(w)}n^{O(1)} = 2^{O(\sqrt{k})}n^{O(1)}$ using dynamic programming.
- If treewidth w is larger than $c\sqrt{k}$: answer is positive, but how much exactly?

Works for counting vertex covers of size k in a planar graph:

- If treewidth w is $O(\sqrt{k})$: can be solved in time $2^{O(w)}n^{O(1)} = 2^{O(\sqrt{k})}n^{O(1)}$ using dynamic programming.
- If treewidth w is larger than $c\sqrt{k}$: answer is 0.

Counting k-matching

Counting matchings can be significantly harder than **finding** a matching!

- Counting perfect matchings is #P-hard [Valiant 1979].
- Counting matchings of size k is #W[1]-hard [Curticapean 2013], [Curticapean and M. 2014].
- Counting matchings of size k is FPT in planar graphs. [Frick 2004]

Open question: Is there a $2^{O(\sqrt{k})} \cdot n^{O(1)}$ algorithm for counting k matchings in planar graphs?

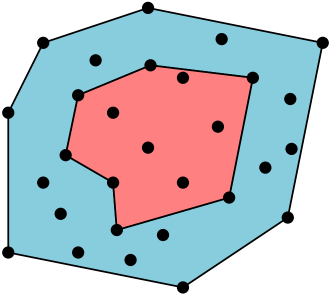
Counting k-matching

Counting matchings can be significantly harder than **finding** a matching!

- Counting perfect matchings in planar graphs is polynomial-time solvable.
  [Kasteleyn 1961], [Temperley and Fischer 1961].
- **Corollary:** we can count matchings covering $n-k$ vertices in time $n^{O(k)}$
- ... but (assuming ETH) there is no $f(k)n^{o(k/\log k)}$ time algorithm [Curticapean and Xia 2015].

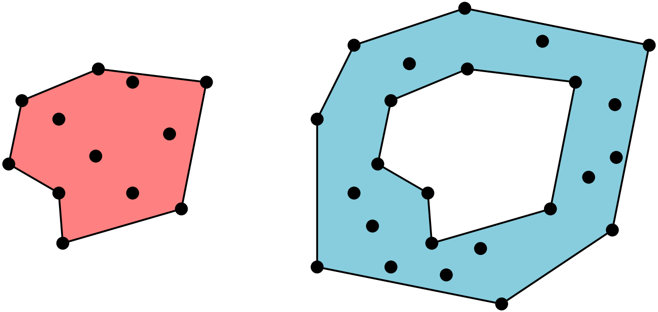
Counting Triangulations

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



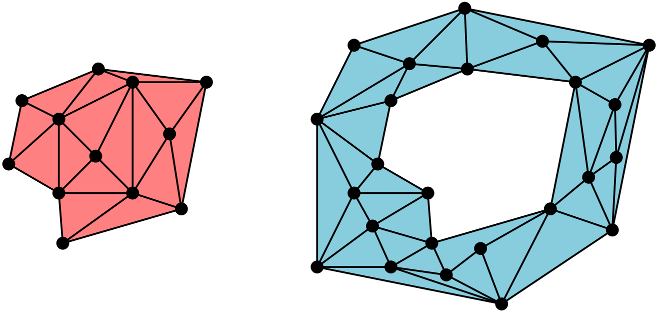
Counting Triangulations

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



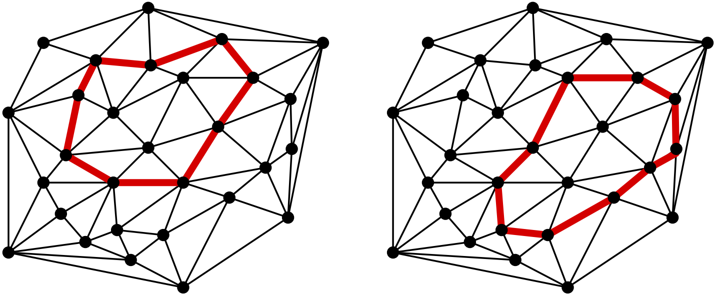
Counting Triangulations

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 significantly overcount the number of triangulations.

Counting Triangulations

Theorem [M. and Miltzow 2016]

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

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

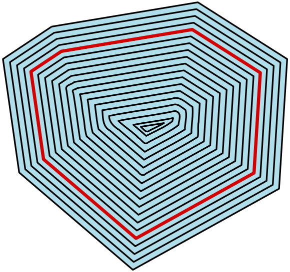
Counting Triangulations

Theorem [M. and Miltzow 2016]

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

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

More than $\sqrt{n}$ layers:



Use the first layer of size $\leq \sqrt{n}$.

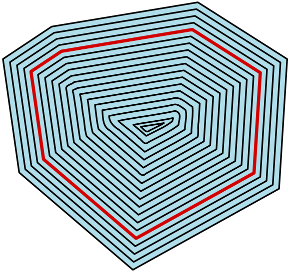
Counting Triangulations

Theorem [M. and Miltzow 2016]

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

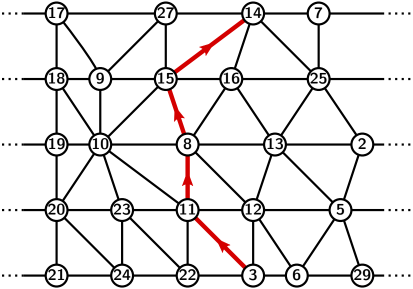
Main idea: Use canonical separators and enforce that they are canonical in the triangulation.

More than $\sqrt{n}$ layers:



Use the first layer of size $\leq \sqrt{n}$.

Less than $\sqrt{n}$ layers:



Build separators from "canonical outgoing paths."

What do we know about a 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]

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 $n^{o(k)}$?

Scattered Set

Scattered Set
Given a graph G and integers k and d, find a set of S of k vertices that are at distance at least d from each other.

- For $d = 2$, we get Independent Set.
- For fixed $d > 2$, bidimensionality gives $2^{O(\sqrt{k})} \cdot n^{O(1)}$ algorithms.
- What happens if d is part of the input?

Scattered Set

Scattered Set
Given a graph G and integers k and d, find a set of S of k vertices that are at distance at least d from each other.

- For $d = 2$, we get Independent Set.
- For fixed $d > 2$, bidimensionality gives $2^{O(\sqrt{k})} \cdot n^{O(1)}$ algorithms.
- What happens if d is part of the input?

Theorem [M. and Pilipczuk 2015]

Scattered Set on planar graphs (with d in the input)

- can be solved in time $n^{O(\sqrt{k})}$,

  [Michał Pilipczuk's talk Wed 11:00]
- cannot be solved in time $f(k)n^{o(\sqrt{k})}$ (assuming ETH).

  [following slides]

W[1]-hardness

Definition

A **parameterized reduction** from problem A to B maps an instance (x, k) of A to instance (x', k') of B such that

- $(x, k) \in A \iff (x', k') \in B$,
- $k' \leq g(k)$ for some computable function g.
- (x', k') can be computed in time $f(k) \cdot |x|^{O(1)}$.

Easy: If there is a parameterized reduction from problem A to problem B and B is FPT, then A is FPT as well.

Definition

A problem P is **W[1]-hard** if there is a parameterized reduction from k-CLIQUE to P.

W[1]-hardness

Definition

A **parameterized reduction** from problem A to B maps an instance (x, k) of A to instance (x', k') of B such that

- $(x, k) \in A \iff (x', k') \in B$,
- $k' \leq g(k)$ **for some computable function** g.
- (x', k') can be computed in time $f(k) \cdot |x|^{O(1)}$.

Easy: If there is a parameterized reduction from problem A to problem B and B is FPT, then A is FPT as well.

Definition

A problem P is **W[1]-hard** if there is a parameterized reduction from k-Clique to P.

Tight bounds

Theorem [Chen et al. 2004]

Assuming ETH, there is no $f(k) \cdot n^{o(k)}$ algorithm for k-Clique for any computable function f.

Transfering to other problems:

k-Clique (x, k)	$\Rightarrow$	Problem A (x', k^2)
$f(k) \cdot n^{o(k)}$ algorithm	$\Leftarrow$	$f(k) \cdot n^{o(\sqrt{k})}$ algorithm

Bottom line:

To rule out $f(k) \cdot n^{o(\sqrt{k})}$ algorithms, we need a parameterized reduction that blows up the parameter at most quadratically.

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 is W[1]-hard

Reduction from k-Clique

Definition of the sets:

- For $i = j$: $(x, y) \in S_{i,j} \iff x = y$
- For $i \neq j$: $(x, y) \in S_{i,j} \iff x$ and y are adjacent.

	(v_i, v_i)			

Each diagonal cell defines a value v_i . . .

Grid Tiling is W[1]-hard

Reduction from k-Clique

Definition of the sets:

- For $i = j$: $(x, y) \in S_{i,j} \iff x = y$
- For $i \neq j$: $(x, y) \in S_{i,j} \iff x$ and y are adjacent.

	$(v_i, .)$			
$(., v_i)$	(v_i, v_i)	$(., v_i)$	$(., v_i)$	$(., v_i)$
	$(v_i, .)$			
	$(v_i .,)$			
	$(v_i, .)$			

... which appears on a "cross"

Grid Tiling is W[1]-hard

Reduction from k-Clique

Definition of the sets:

- For $i = j$: $(x, y) \in S_{i,j} \iff x = y$
- For $i \neq j$: $(x, y) \in S_{i,j} \iff x$ and y are adjacent.

	$(v_i, .)$			
$(., v_i)$	(v_i, v_i)	$(., v_i)$	$(., v_i)$	$(., v_i)$
	$(v_i, .)$			
	$(v_i, .)$		(v_j, v_j)	
	$(v_i, .)$			

v_i and v_j are adjacent for every $1 \leq i < j \leq k$.

Grid Tiling is W[1]-hard

Reduction from k-Clique

Definition of the sets:

- For $i = j$: $(x, y) \in S_{i,j} \iff x = y$
- For $i \neq j$: $(x, y) \in S_{i,j} \iff x$ and y are adjacent.

	$(v_i, .)$		$(v_j, .)$	
$(., v_i)$	(v_i, v_i)	$(., v_i)$	(v_j, v_i)	$(., v_i)$
	$(v_i, .)$		$(v_j, .)$	
$(., v_j)$	(v_i, v_j)	$(., v_j)$	(v_j, v_j)	$(., v_j)$
	$(v_i, .)$		$(v_j, .)$	

v_i and v_j are adjacent for every $1 \leq i < j \leq k$.

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

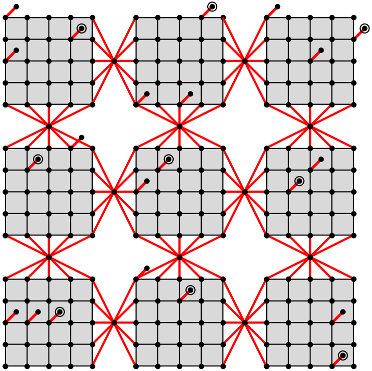
There is a parameterized reduction from $k \times k$-Grid Tiling to $O(k) \times O(k)$ Grid Tiling with $\leq$.

Very useful starting point for geometric (and also some planar) problems!

Grid Tiling with $\leq$ $\Rightarrow$ Scattered Set

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

$\Rightarrow$



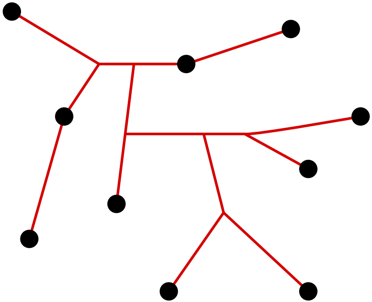
required distance: at least n black edges + 4 red edges

Solution to $k \times k$ grid tiling $\quad\Rightarrow\quad$ scattered set of size k^2

Steiner Tree

Steiner Tree

Given an edge-weighted graph G and set $T \subseteq V(G)$ of terminals, find a minimum weight tree in G containing every vertex of T.



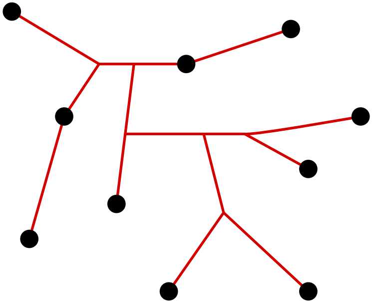
Theorem [Dreyfus and Wagner 1971]

Steiner Tree with k terminals can be solved in time $3^k \cdot n^{O(1)}$.

Steiner Tree

Steiner Tree

Given an edge-weighted graph G and set $T \subseteq V(G)$ of terminals, find a minimum weight tree in G containing every vertex of T.

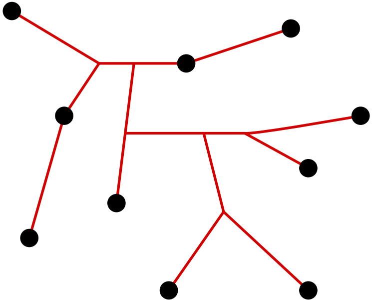


Theorem [Björklund et al. 2007]

Steiner Tree with k terminals can be solved in time $2^k \cdot n^{O(1)}$.

Steiner Tree

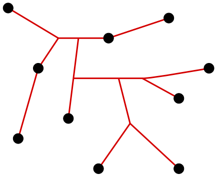
Steiner Tree
Given an edge-weighted graph G and set $T \subseteq V(G)$ of terminals, find a minimum weight tree in G containing every vertex of T.



Open question: Can we solve Steiner Tree on planar graphs with k terminals in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$?

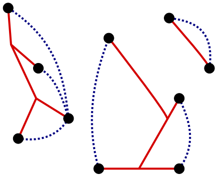
Variants of Steiner Tree

Steiner Tree



Connect all the terminals

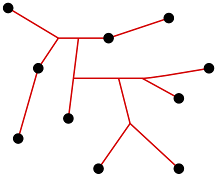
Steiner Forest



Create connections satisying every request

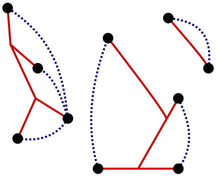
Variants of STEINER TREE

STEINER TREE



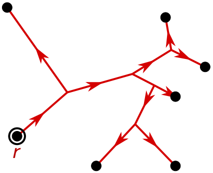
Connect all the terminals

STEINER FOREST



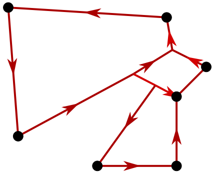
Create connections satisying every request

STEINER TREE



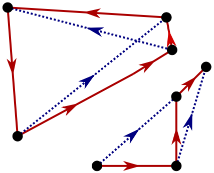
Make every terminal reachable from the root

STRONGLY CONNECTED STEINER SUBGRAPH (SCSS)



Make all the terminals reachable from each other

DIRECTED STEINER NETWORK (DSN)



Create connections satisying every request

Directed Steiner Network

Theorem [Feldman and Ruhl 2006]

Directed Steiner Network with k requests can be solved in time $n^{O(k)}$.

Corollary: Strongly Connected Steiner Subgraph with k terminals can be solved in time $n^{O(k)}$.

Proof is based on a “pebble game”: $O(k)$ pebbles need to reach their destinations using certain allowed moves, tracing the solution.

Directed Steiner Network

A new combinatorial result:

Theorem [Feldmann and M. 2016]

Every minimum cost solution of Directed Steiner Network with k requests has cutwidth and treewidth $O(k)$.

Directed Steiner Network

A new combinatorial result:

Theorem [Feldmann and M. 2016]

Every minimum cost solution of Directed Steiner Network with k requests has cutwidth and treewidth $O(k)$.

A new algorithmic result:

Theorem [Feldmann and M. 2016]

If a Directed Steiner Network instance with k requests has a minimum cost solution with treewidth w, then it can be solved in time $f(k, w) \cdot n^{O(w)}$.

Corollary: A new proof that DSN and SCSS can be solved in time $f(k)n^{O(k)}$.

Planar Steiner Problems

Square root phenomenon for SCSS:

Theorem [Chitnis, Hajiaghayi, M. 2014]

Strongly Connected Steiner Subgraph with k terminals can be solved in time $f(k)n^{O(\sqrt{k})}$ on planar graphs.

Proof by a complicated generalization of the Feldman-Ruhl pebble game.

Planar Steiner Problems

Square root phenomenon for SCSS:

Theorem [Chitnis, Hajiaghayi, M. 2014]

Strongly Connected Steiner Subgraph with k terminals can be solved in time $f(k)n^{O(\sqrt{k})}$ on planar graphs.

Proof by a complicated generalization of the Feldman-Ruhl pebble game.

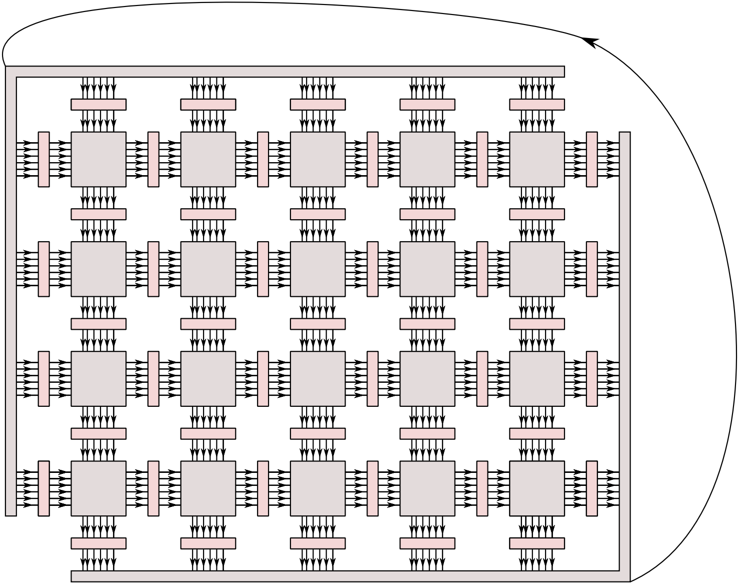
Lower bound:

Theorem [Chitnis, Hajiaghayi, M. 2014]

Assuming ETH, Strongly Connected Steiner Subgraph with k terminals cannot be solved in time $f(k)n^{o(\sqrt{k})}$ on planar graphs.

Proof by reduction from Grid Tiling.

Lower bound for planar SCSS

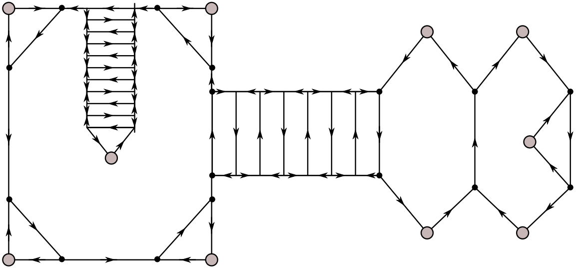


Planar Strongly Connected Steiner Subgraph

A new combinatorial result:

Theorem [Feldmann and M. 2016]

Every minimum cost solution of SCSS with k terminals has "distance $O(k)$ from treewidth 2."



Corollary

Every minimum cost solution of SCSS with k terminals has treewidth $O(\sqrt{k})$ on planar graphs.

Planar Strongly Connected Steiner Subgraph

Corollary

Every minimum cost solution of SCSS with k terminals has treewidth $O(\sqrt{k})$ on planar graphs.

We have seen:

Theorem [Feldmann and M. 2016]

If a Directed Steiner Network instance with k requests has a minimum cost solution with treewidth w, then it can be solved in time $f(k, w) \cdot n^{O(w)}$.

Corollary: A new proof that SCSS can be solved in time $f(k)n^{O(\sqrt{k})}$ on planar graphs.

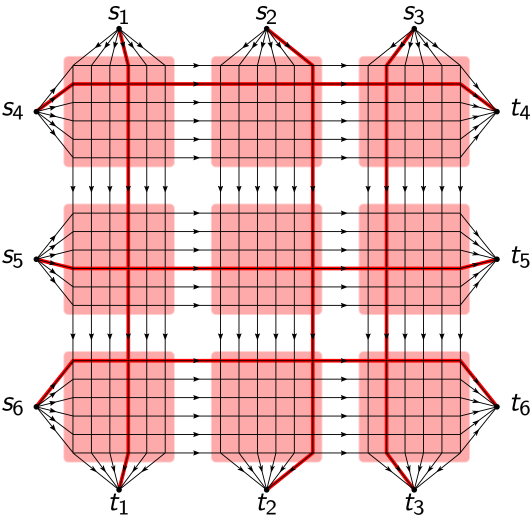
Planar Directed Steiner Network

No square root phenomenon for DSN:

Theorem [Chitnis, Hajiaghayi, M. 2014]

Directed Steiner Network with k requests is W[1]-hard on planar graphs and (assuming ETH) cannot be solved in time $f(k)n^{o(k)}$.

Planar Directed Steiner Network



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 $O(\sqrt{n})$.
  - Strongly Connected Steiner Subgraph on planar graphs: optimum solution can be made treewidth-2 with $O(k)$ deletions $\Rightarrow$ treewidth is $O(\sqrt{k})$.
- **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 algorithms for counting k-matchings in planar graphs?
    - $\dots$