The square root phenomenon in planar graphs

Dániel Marx[1]

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

CS Theory Seminar
The Hebrew University of Jerusalem
May 22, 2013
Jerusalem, Israel

Main message

Are NP-hard problems easier on planar graphs?

Yes, usually.

By how much?

Often by exactly a square root factor.

Overview

Chapter 1:
Subexponential algorithms using treewidth.

Chapter 2:
Grid minors and bidimensionality.

Chapter 3:
Finding bounded-treewidth solutions.

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.

Chapter 1: Subexponential algorithms using treewidth

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

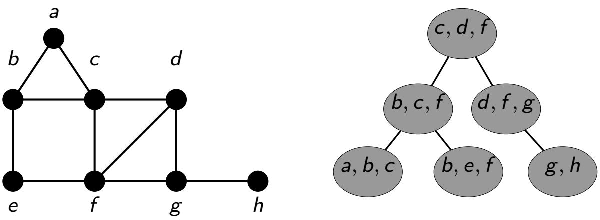
We need only the following basic facts:

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 $O(\sqrt{n})$.
3. Excluded Grid Theorem: a planar graph of treewidth k contains a $\Omega(k) \times \Omega(k)$ grid minor.

Treewidth — a measure of "tree-likeness"

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

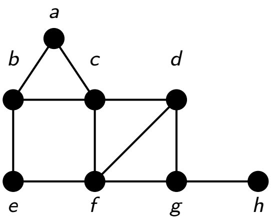
1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

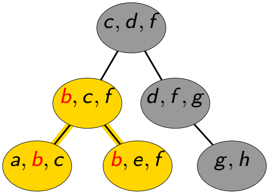


Treewidth — a measure of "tree-likeness"

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.

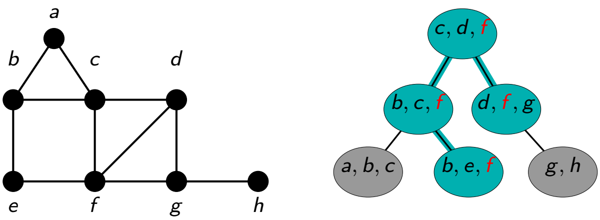




Treewidth — a measure of "tree-likeness"

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.



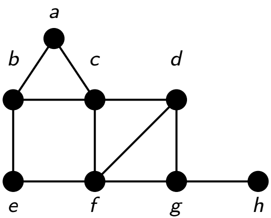
Treewidth — a measure of "tree-likeness"

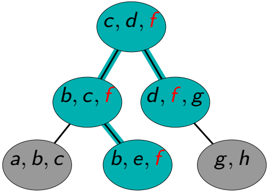
Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

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.





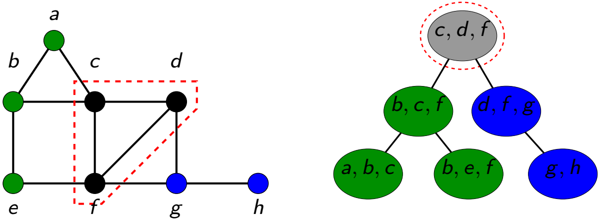
Treewidth — a measure of "tree-likeness"

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.



Each bag is a separator.

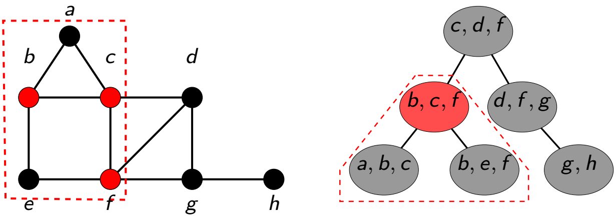
Treewidth — a measure of "tree-likeness"

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

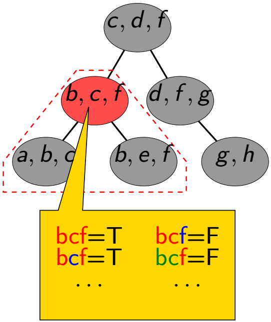
Theorem

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

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

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 $O(\sqrt{n})$.

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

$\textsc{3-Coloring}$ 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 $\mathrm{3SAT}$ 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)

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 $\Omega(n^3)$ clauses.

Exponential Time Hypothesis (ETH)

Hypothesis introduced by Impagliazzo, Paturi, and Zane:

Exponential Time Hypothesis (ETH)

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 $\Omega(n^3)$ clauses.

Sparsification Lemma [Impagliazzo, Paturi, Zane 2001]

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

$$\Updownarrow$$

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

Lower bounds based on ETH

Exponential Time Hypothesis (ETH)

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

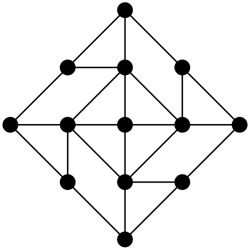
Corollary

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

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:

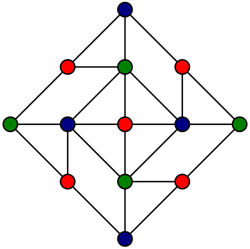


- In every 3-coloring of the gadget, opposite external connectors have the same color.
- Every coloring of the external connectors where the opposite vertices have the same color can be extended to the whole gadgets.
- If two edges cross, replace them with a crossover gadget.

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:

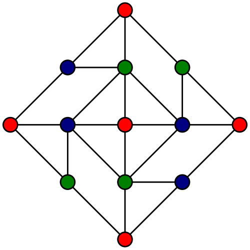


- In every 3-coloring of the gadget, opposite external connectors have the same color.
- Every coloring of the external connectors where the opposite vertices have the same color can be extended to the whole gadgets.
- If two edges cross, replace them with a crossover gadget.

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 gadgets.
- 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 edge/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 a 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.

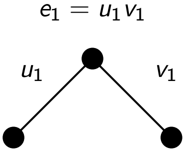
Bounded search tree method

Algorithm for VERTEX COVER:

$$e_1 = u_1 v_1$$

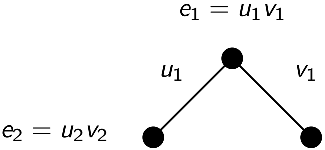
Bounded search tree method

Algorithm for VERTEX COVER:



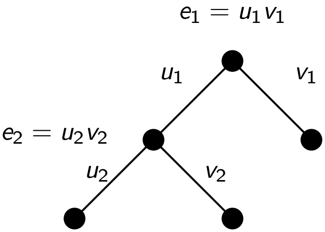
Bounded search tree method

Algorithm for VERTEX COVER:



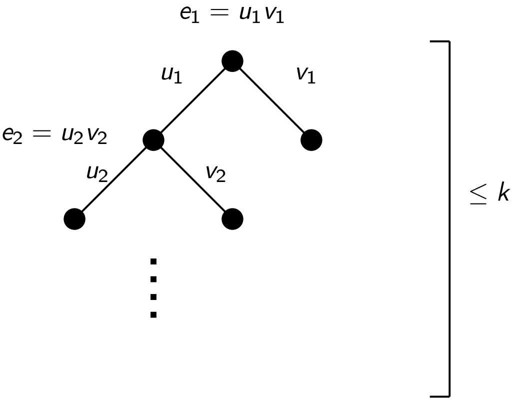
Bounded search tree method

Algorithm for Vertex Cover:



Bounded search tree method

Algorithm for VERTEX COVER:



Height of the search tree $\leq k$ $\Rightarrow$ at most 2^k leaves $\Rightarrow$ $2^k \cdot n^{O(1)}$ time algorithm.

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.
- $\ldots$

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

Subexponential parameterized algorithms

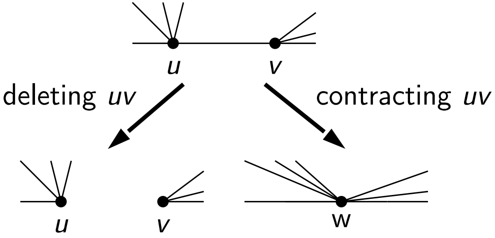
What kind of upper/lower bounds we have for $f(k)$?

- 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).
- 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]

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.

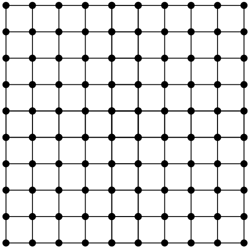


Note: minimum vertex cover size of H is at most the minimum vertex cover size of G.

Planar Excluded Grid Theorem

Theorem [Robertson, Seymour, Thomas 1994]

Every planar graph with treewidth at least $4k$ has a $k \times k$ grid minor.



Note: for general graphs, we need treewidth at least $k^{4k^4(k+2)}$ for a $k \times k$ grid minor [Diestel et al. 1999].

Bidimensionality for Vertex Cover

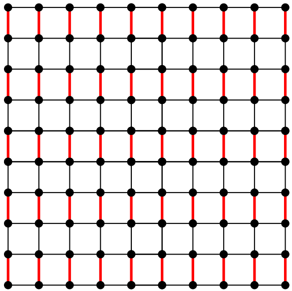
Observation: If the treewidth of a planar graph G is at least $4\sqrt{2k}$

$\Rightarrow$ It has a $\sqrt{2k} \times \sqrt{2k}$ grid minor (Planar Excluded Grid Theorem)

$\Rightarrow$ The grid has a matching of size k

$\Rightarrow$ The minimum vertex cover size of the grid is at least k

$\Rightarrow$ The minimum vertex cover size of G is at least k.

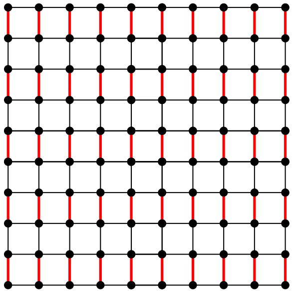


Bidimensionality for Vertex Cover

Observation: If the treewidth of a planar graph G is at least $4\sqrt{2k}$
$\Rightarrow$ It has a $\sqrt{2k} \times \sqrt{2k}$ grid minor (Planar Excluded Grid Theorem)
$\Rightarrow$ The grid has a matching of size k
$\Rightarrow$ The minimum vertex cover size of the grid is at least k
$\Rightarrow$ The minimum vertex cover size of G is at least k.

We use this observation to solve Vertex Cover on planar graphs:

- Set $w := 4\sqrt{2k}$.
- Find a 4-approximate tree decomposition.
  - If treewidth is at least w: we answer "vertex cover is $\geq k$."
  - If we get a tree decomposition of width $4w$, then we can solve the problem in time $2^{O(w)} \cdot n^{O(1)} = 2^{O(\sqrt{k})} \cdot n^{O(1)}$.

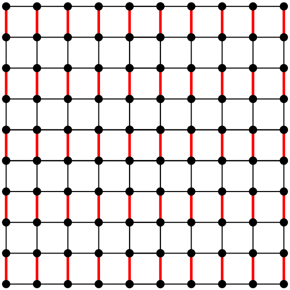


Bidimensionality

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 $c > 0$).



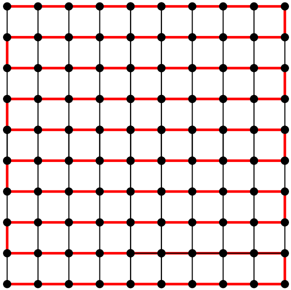
Examples: minimum vertex cover, length of the longest path, feedback vertex set are minor-bidimensional.

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 $c > 0$).



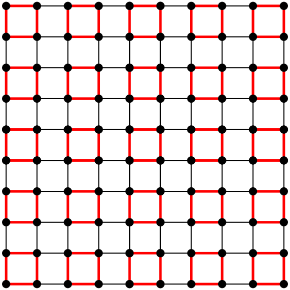
Examples: minimum vertex cover, length of the longest path, feedback vertex set are minor-bidimensional.

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

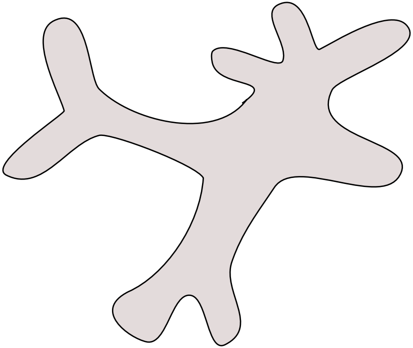
Tight bounds for minor-bidimensional planar problems.

- **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.

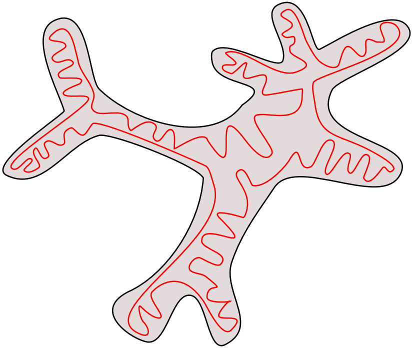
Chapter 3: Finding bounded treewidth solutions

So far the way we have used treewidth is to find something (e.g., Hamiltonian cycle) in a large bounded-treewidth graph:



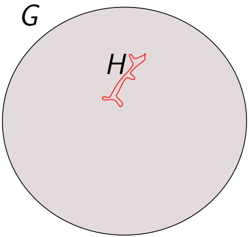
Chapter 3: Finding bounded treewidth solutions

So far the way we have used treewidth is to find something (e.g., Hamiltonian cycle) in a large bounded-treewidth graph:



Chapter 3: Finding bounded treewidth solutions

But we can also find small bounded-treewidth graphs in an arbitrary large graph.

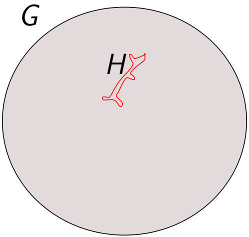


Theorem [Alon, Yuster, Zwick 1994]

Given a graph H and weighted graph G, we can find a minimum weight subgraph of G isomorphic to H in time $2^{O(|V(H)|)} \cdot n^{O(\mathrm{tw}(H))}$.

Chapter 3: Finding bounded treewidth solutions

But we can also find small bounded-treewidth graphs in an arbitrary large graph.



Theorem [Alon, Yuster, Zwick 1994]

Given a graph H and weighted graph G, we can find a minimum weight subgraph of G isomorphic to H in time $2^{O(|V(H)|)} \cdot n^{O(\mathrm{tw}(H))}$.

If the problem can be formulated as finding a graph of treewidth $O(\sqrt{k})$, then we get an $n^{O(\sqrt{k})}$ time algorithm.

Examples

Three examples:

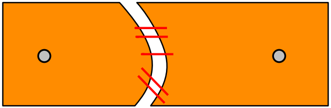
- PLANAR k-TERMINAL CUT
  Improvement from $n^{O(k)}$ to $2^{O(k)} \cdot n^{O(\sqrt{k})}$.
- PLANAR STRONGLY CONNECTED SUBGRAPH
  Improvement from $n^{O(k)}$ to $2^{O(k \log k)} \cdot n^{O(\sqrt{k})}$.
- TSP with shortest path metric of a planar graph
  Improvement from $2^{O(k)} \cdot n^{O(1)}$ to $2^{O(\sqrt{k} \log k)} \cdot n^{O(1)}$.

A classical problem

$s - t$ Cut

Input: A graph G, an integer p, vertices s and t

Output: A set S of at most p edges such that removing S separates s and t.



Theorem [Ford and Fulkerson 1956]

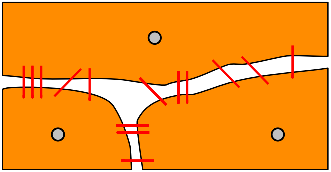
A minimum $s - t$ cut can be found in polynomial time.

What about separating more than two terminals?

More than two terminals

Multiway Cut (aka k-Terminal Cut)

Input: A graph G, an integer p, and a set T of k terminals
Output: A set S of at most p edges such that removing S separates any two vertices of T



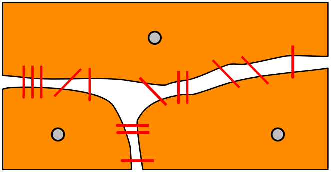
Theorem [Dalhaus et al. 1994]

NP-hard already for $k = 3$.

More than two terminals

Multiway Cut (aka k-Terminal Cut)

Input: A graph G, an integer p, and a set T of k terminals
Output: A set S of at most p edges such that removing S separates any two vertices of T



Theorem [Dalhaus et al. 1994] [Hartvigsen 1998] [Bentz 2012]

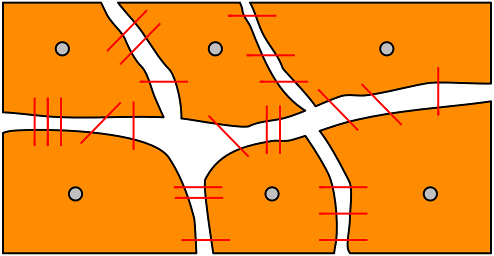
Planar k-Terminal Cut can be solved in time $n^{O(k)}$.

Theorem [Klein and M. 2012]

Planar k-Terminal Cut can be solved in time $2^{O(k)} \cdot n^{O(\sqrt{k})}$.

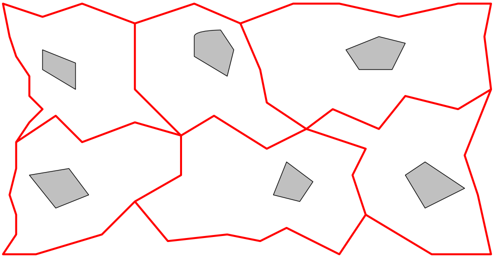
Dual graph

The first step of the algorithms is to look at the solution in the dual graph:



Dual graph

The first step of the algorithms is to look at the solution in the dual graph:

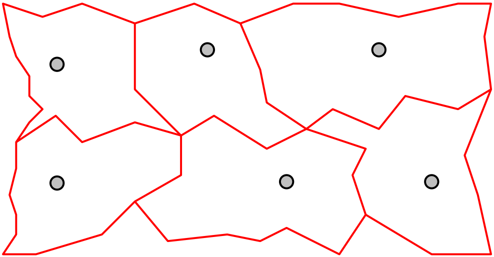


Recall:

Primal graph		Dual graph
vertices	$\Leftrightarrow$	faces
faces	$\Leftrightarrow$	vertices
edges	$\Leftrightarrow$	edges

Dual graph

The first step of the algorithms is to look at the solution in the dual graph:

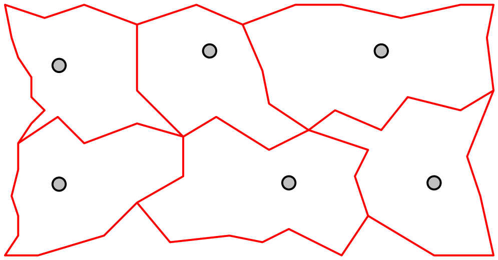


Recall:

Primal graph		Dual graph
vertices	$\Leftrightarrow$	faces
faces	$\Leftrightarrow$	vertices
edges	$\Leftrightarrow$	edges

We slightly transform the problem in such a way that the terminals are represented by **vertices** in the dual graph (instead of faces).

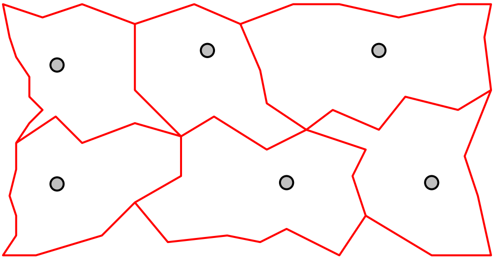
Finding the dual solution



Main ideas of [Dalhaus et al. 1994] [Hartvigsen 1998] [Bentz 2012]**:**

1. The dual solution has $O(k)$ branch vertices.
2. Guess the location of branch vertices ($n^{O(k)}$ guesses).
3. Deep magic to find the paths connecting the branch vertices (shortest paths are not necessarily good!)

Finding the dual solution



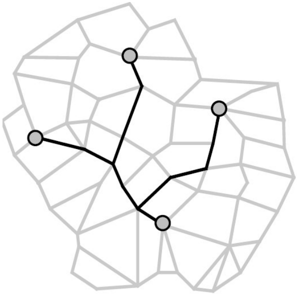
Idea for $n^{O(\sqrt{k})}$ time algorithm:

- Guess the graph H representing the branch vertices.
- Build a weighted complete graph G representing the distances in the planar graph.
- Find in time $n^{O(\mathrm{tw}(H))} = n^{O(\sqrt{k})}$ a minimum weight copy of H in G.

Problem: How to ensure that the solution separates the terminals?

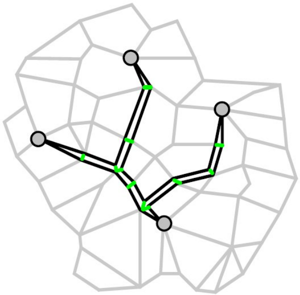
The Steiner tree

We find a minimum cost Steiner tree T of the terminals in the **dual** and cut open the graph along the tree.
(Steiner tree: $3^k \cdot n^{O(1)}$ time by [Dreyfus-Wagner 1972] or $2^k \cdot n^{O(1)}$ time by [Björklund 2007])



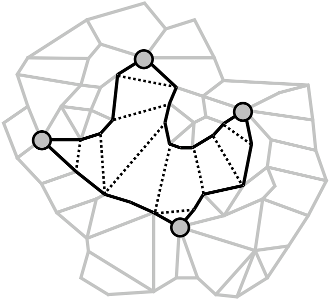
The Steiner tree

We find a minimum cost Steiner tree T of the terminals in the **dual** and cut open the graph along the tree.
(Steiner tree: $3^k \cdot n^{O(1)}$ time by [Dreyfus-Wagner 1972] or $2^k \cdot n^{O(1)}$ time by [Björklund 2007])



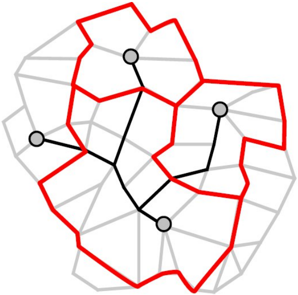
The Steiner tree

We find a minimum cost Steiner tree T of the terminals in the **dual** and cut open the graph along the tree.
(Steiner tree: $3^k \cdot n^{O(1)}$ time by [Dreyfus-Wagner 1972] or $2^k \cdot n^{O(1)}$ time by [Björklund 2007])



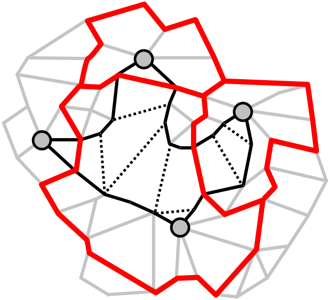
The Steiner tree

We find a minimum cost Steiner tree T of the terminals in the **dual** and cut open the graph along the tree.
(Steiner tree: $3^k \cdot n^{O(1)}$ time by [Dreyfus-Wagner 1972] or $2^k \cdot n^{O(1)}$ time by [Björklund 2007])



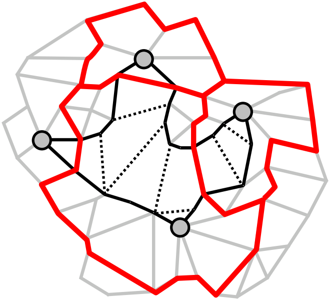
The Steiner tree

We find a minimum cost Steiner tree T of the terminals in the **dual** and cut open the graph along the tree.
(Steiner tree: $3^k \cdot n^{O(1)}$ time by [Dreyfus-Wagner 1972] or $2^k \cdot n^{O(1)}$ time by [Björklund 2007])



The Steiner tree

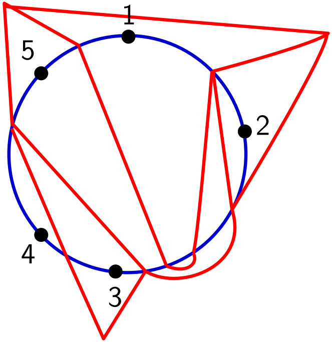
We find a minimum cost Steiner tree T of the terminals in the **dual** and cut open the graph along the tree.
(Steiner tree: $3^k \cdot n^{O(1)}$ time by [Dreyfus-Wagner 1972] or $2^k \cdot n^{O(1)}$ time by [Björklund 2007])



Key idea: the paths of the dual solution between the branch points/crossing points can be assumed to be shortest paths.

Topology

Key idea: the paths of the dual solution between the branch points/crossing points can be assumed to be shortest paths.



Thus a solution can be completely described by the location of these points and which of them are connected.
A "topology" just describes the connections without the locations.

Lower bounds

Theorem [Klein and M. 2012]

Planar k-Terminal Cut can be solved in time $2^{O(k)} \cdot n^{O(\sqrt{k})}$.

Natural questions:

- Is there an $f(k) \cdot n^{o(\sqrt{k})}$ time algorithm?
- Is there an $f(k) \cdot n^{O(1)}$ time algorithm (i.e., is it fixed-parameter tractable)?

Lower bounds

Theorem [Klein and M. 2012]

Planar k-Terminal Cut can be solved in time $2^{O(k)} \cdot n^{O(\sqrt{k})}$.

Natural questions:

- Is there an $f(k) \cdot n^{o(\sqrt{k})}$ time algorithm?
- Is there an $f(k) \cdot n^{O(1)}$ time algorithm (i.e., is it fixed-parameter tractable)?

The previous lower bound technology is of no help here: showing that there is no $2^{o(\sqrt{n})}$ time algorithm does not answer the question.

Lower bounds:

Theorem [M. 2012]

Planar k-Terminal Cut is W[1]-hard and has no $f(k) \cdot n^{o(\sqrt{k})}$ time algorithm (assuming ETH).

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

- $(x, k) \in A \iff (x', k') \in B$,
- $k' \le g(k)$ 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.

W[1]-hardness vs. NP-hardness

W[1]-hardness proofs are more delicate than NP-hardness proofs: we need to control the new parameter.

Example: k-Independent Set can be reduced to k'-Vertex Cover with $k' := n - k$. But this is **not** a parameterized reduction.

W[1]-hardness vs. NP-hardness

W[1]-hardness proofs are more delicate than NP-hardness proofs: we need to control the new parameter.

Example: k-Independent Set can be reduced to k'-Vertex Cover with $k' := n - k$. But this is **not** a parameterized reduction.

NP-hardness proof

Reduction from some graph problem. We build n vertex gadgets of constant size and m edge gadgets of constant size.

W[1]-hardness proof

Reduction from k-Clique. We build k large vertex gadgets, each having n states (and/or $\binom{k}{2}$ large edge gadgets with m states).

Planar problems

Another difference: Most problems remain NP-hard on planar graphs, but become FPT.

Algorithmic techniques for planar problems:

- Baker's shifting technique + treewidth
- Bidimensionality
- Protrusions

Very few W[1]-hardness results so far for planar problems.

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:
If there is a parameterized reduction from k-Clique to problem A mapping (x, k) to $(x', g(k))$, then an $f(k) \cdot n^{o(g^{-1}(k))}$ algorithm for problem A gives an $f(k) \cdot n^{o(k)}$ algorithm for k-Clique, contradicting ETH.

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.

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 $(x', g(k))$
$f(k) \cdot n^{o(k)}$ algorithm	$\Leftarrow$	$f(k) \cdot n^{o(g^{-1}(k))}$ 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

- Horizontal neighbors agree in the first component.
- Vertical neighbors agree in the second component.

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

$k = 3$, $D = 5$

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

- Horizontal neighbors agree in the first component.
- Vertical neighbors agree in the second component.

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

$k = 3$, $D = 5$

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

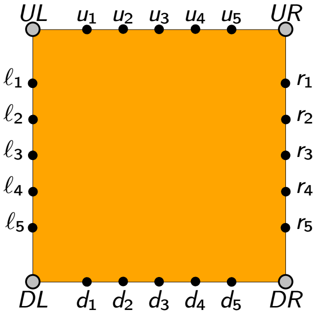
v_i and v_j are adjacent for every $1 \leq i < j \leq k$.

The gadget

For every set $S_{i,j}$, we construct a gadget such that

- for every $(x, y) \in S_{i,j}$, there is a minimum multiway cut that represents (x, y).
- every minimum multiway cut represents some $(x, y) \in S_{i,j}$.

Main part of the proof: constructing these gadgets.



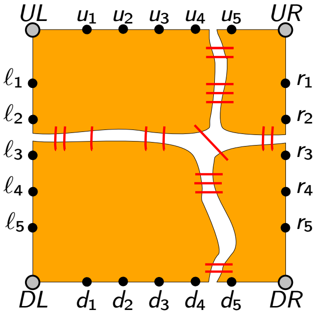
The gadget.

The gadget

For every set $S_{i,j}$, we construct a gadget such that

- for every $(x, y) \in S_{i,j}$, there is a minimum multiway cut that represents (x, y).
- every minimum multiway cut represents some $(x, y) \in S_{i,j}$.

Main part of the proof: constructing these gadgets.



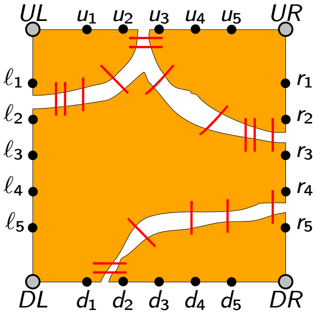
A cut representing (2,4).

The gadget

For every set $S_{i,j}$, we construct a gadget such that

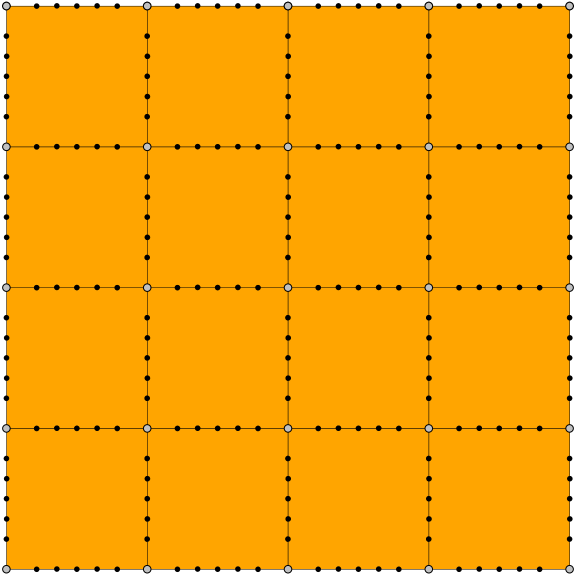
- for every $(x, y) \in S_{i,j}$, there is a minimum multiway cut that represents (x, y).
- every minimum multiway cut represents some $(x, y) \in S_{i,j}$.

Main part of the proof: constructing these gadgets.

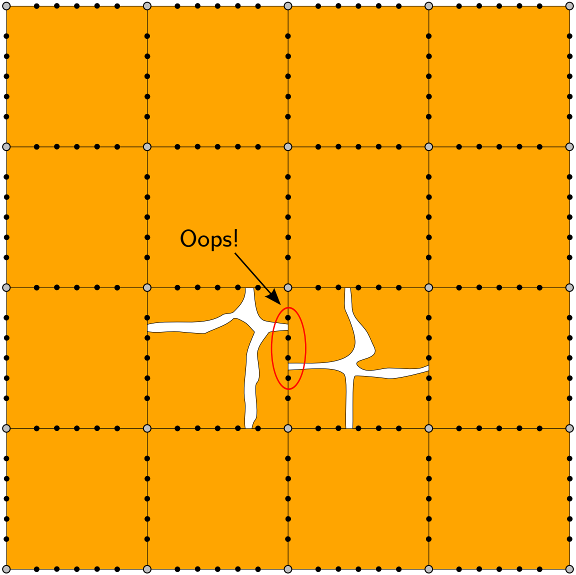


A cut not representing any pair.

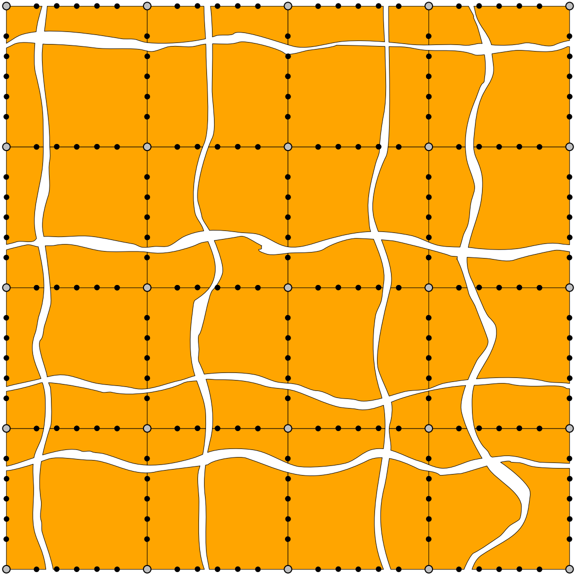
Putting together the gadgets



Putting together the gadgets



Putting together the gadgets



Planar Multiway Cut

- **Upper bound:**
  Looking at the dual + cutting open a Steiner tree + guessing a topology + finding a graph of treewidth $O(\sqrt{k})$.
- **Lower bound:**
  ETH + reduction from Grid Tiling + tricky gadget construction rule out $f(k) \cdot n^{o(\sqrt{k})}$ time algorithms.

Strongly Connected Subgraph

Undirected graphs:

Steiner Tree: Find a minimum weight connected subgraph that contains all k terminals.

Theorem [Dreyfus-Wagner 1972]

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

Strongly Connected Subgraph

Undirected graphs:

Steiner Tree: Find a minimum weight connected subgraph that contains all k terminals.

Theorem [Dreyfus-Wagner 1972]

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

Directed graphs:

Strongly Connected Subgraph: Find a minimum weight strongly connected subgraph that contains all k terminals.

Theorem [Guo, Niedermeier, Suchý 2011]

Strongly Connected Subgraph on general directed graphs is W[1]-hard parameterized by k.

Theorem [Feldman and Ruhl 2006]

Strongly Connected Subgraph can be solved in time $n^{O(k)}$ on general directed graphs.

STRONGLY CONNECTED SUBGRAPH on planar graphs

Theorem [Feldman and Ruhl 2006]

STRONGLY CONNECTED SUBGRAPH can be solved in time $n^{O(k)}$ on general directed graphs.

Natural questions:

- Is there an $f(k) \cdot n^{o(k)}$ time algorithm on planar graphs?
- Is there an $f(k) \cdot n^{O(1)}$ time algorithm (i.e., is it fixed-parameter tractable) on planar graphs?

Strongly Connected Subgraph on planar graphs

Theorem [Feldman and Ruhl 2006]

Strongly Connected Subgraph can be solved in time $n^{O(k)}$ on general directed graphs.

Natural questions:

- Is there an $f(k) \cdot n^{o(k)}$ time algorithm on planar graphs?
- Is there an $f(k) \cdot n^{O(1)}$ time algorithm (i.e., is it fixed-parameter tractable) on planar graphs?

Theorem [Chitnis, Hajiaghayi, M.]

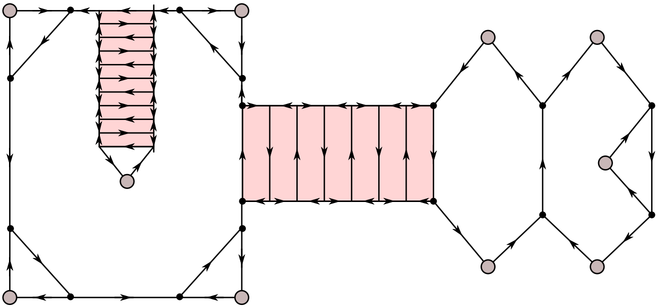
Strongly Connected Subgraph can be solved in time $2^{O(k \log k)} \cdot n^{O(\sqrt{k})}$ on planar directed graphs.

Theorem [Chitnis, Hajiaghayi, M.]

Strongly Connected Subgraph has no $f(k) \cdot n^{o(\sqrt{k})}$ time algorithm on planar directed graphs (assuming ETH).

Optimum solutions

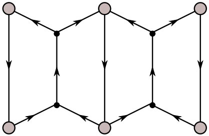
Closely looking at the $n^{O(k)}$ algorithm of [Feldman and Ruhl 2006] shows that an optimum solution consists of directed paths and "bidirectional strips":

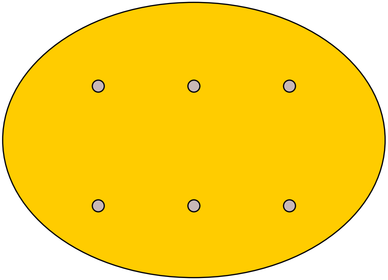


With some work, we can bound the number paths/strips by $O(k)$.

Algorithm

[Ignore the bidirectional strips for simplicity]

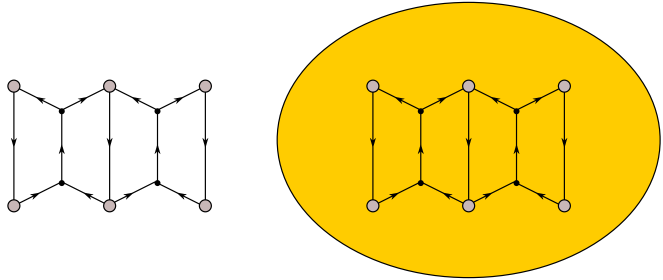




- We guess the topology of the solution ($2^{O(k \log k)}$ possibilities).
- Treewidth of the topology is $O(\sqrt{k})$.
- We can find the best realization of this topology (matching the location of the terminals) in time $n^{O(\sqrt{k})}$.

Algorithm

[Ignore the bidirectional strips for simplicity]



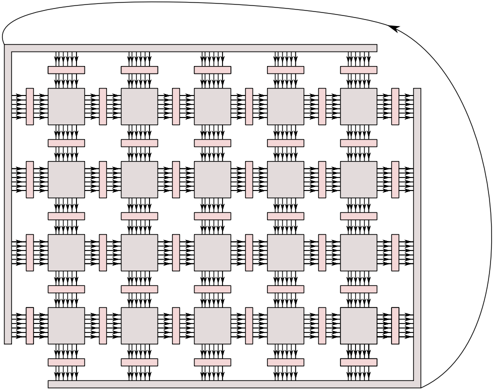
- We guess the topology of the solution ($2^{O(k \log k)}$ possibilities).
- Treewidth of the topology is $O(\sqrt{k})$.
- We can find the best realization of this topology (matching the location of the terminals) in time $n^{O(\sqrt{k})}$.

Lower bound

Theorem [Chitnis, Hajiaghayi, M.]

STRONGLY CONNECTED SUBGRAPH has no $f(k) \cdot n^{o(\sqrt{k})}$ time algorithm on planar directed graphs (assuming ETH).

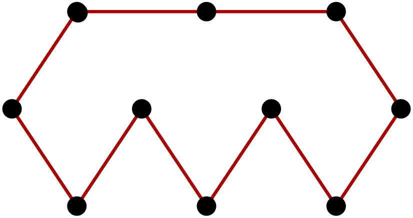
The proof is by reduction from GRID TILING and complicated construction of gadgets.



TSP

TSP

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



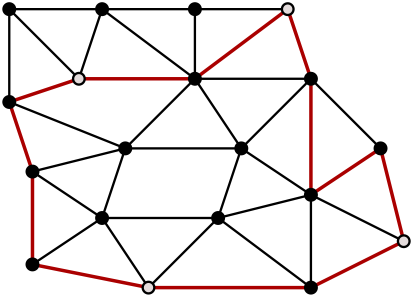
Theorem [Held and Karp]

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

Dynamic programming:
Let $x(v, T')$ be the minimum length of path from v_{start} to v visiting all the cities $T' \subseteq T$.

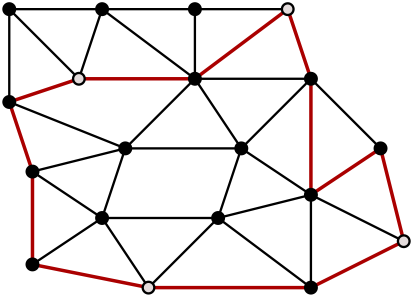
TSP on planar graphs

Assume that the distance function d is generated by a (weighted) planar graph and T is a subset of vertices.



TSP on planar graphs

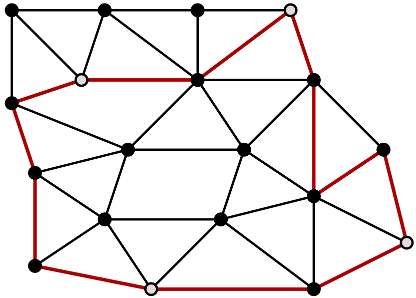
Assume that the distance function d is generated by a (weighted) planar graph and T is a subset of vertices.



- Can be solved in time $2^{O(\sqrt{n})}$.
- Can be solved in time $2^k \cdot n^{O(1)}$.
- Can we solve it in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$?

TSP on planar graphs

Assume that the distance function d is generated by a (weighted) planar graph and T is a subset of vertices.



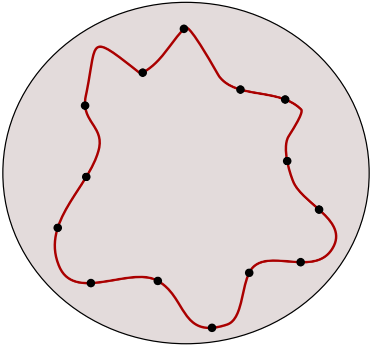
Theorem [Klein and M.]

TSP with a distance function d generated by a planar graph can be solved in time $2^{O(\sqrt{k})} \cdot W^{O(1)}$, where W is the maximum distance in d.

Note: We do not have to know the graph, only the function d.

TSP and treewidth

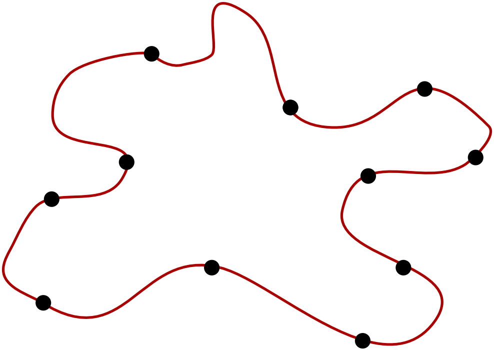
- We wanted to formulate the problem as finding a low treewidth subgraph.
- A cycle has treewidth 2, is this of any help?



Problem:
We have to remember the subset of cities visited by the partial tour (2^k possibilities).

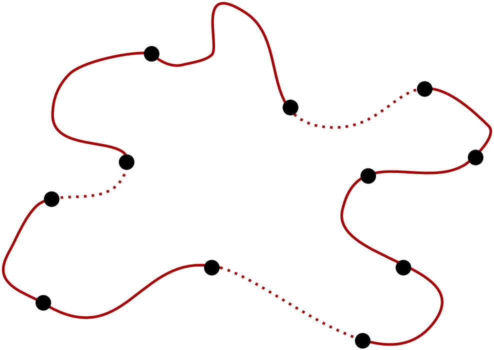
c-change TSP

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



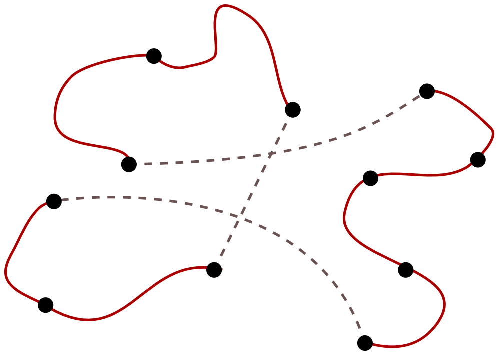
c-change TSP

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



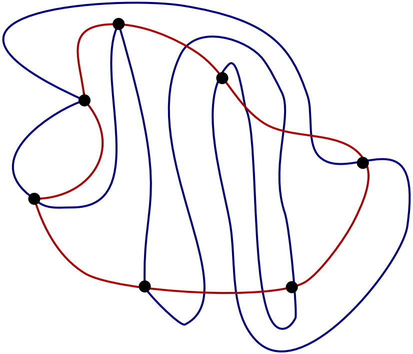
c-change TSP

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



The crossing graph

Consider a optimum solution and a 4-OPT solution:
[assume that the two tours do not share edges, etc.]



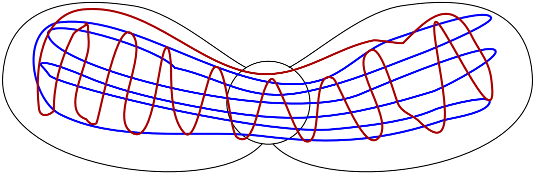
Lemma

The crossing graph of an optimum solution and a 4-OPT solution has $O(k)$ vertices and has treewidth $O(\sqrt{k})$.

The crossing graph

Lemma

The crossing graph of an optimum solution and a 4-OPT solution has $O(k)$ vertices and has treewidth $O(\sqrt{k})$.



- The crossing graph 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-OPT tour visited ($k^{O(\sqrt{k})}$ possibilities).

Summary of Chapter 3

Parameterized problems where bidimensionality does not work.

- **Upper bounds:**
  Algorithms based on finding a bounded-treewidth subgraph.
  Treewidth bound is problem-specific:
  - k-Terminal Cut: dual solution has $O(k)$ branch vertices.
  - Planar Strongly Connected Subgraph: solution consists of $O(k)$ paths/strips.
  - TSP with a planar graph metric: the crossing graph of an optimum solution and a 4-OPT solution has size $O(k)$.
- **Lower bounds:**
  To rule out $f(k) \cdot n^{o(\sqrt{k})}$ time algorithms, we have to prove W[1]-hardness by reduction from Grid Tiling.

Conclusions

- **Chapter 1:** Subexponential algorithms using treewidth.
  - Algorithms: standard treewidth algorithms.
  - Lower bounds: textbook NP-completeness proofs + ETH.
- **Chapter 2:** Grid minors and bidimensionality.
  - Algorithms: standard treewidth algorithms + excluded grid theorem.
  - Lower bounds: textbook NP-completeness proofs + ETH.
- **Chapter 3:** Finding bounded treewidth solutions.
  - Algorithms: the solution can be represented by a graph of treewidth $O(\sqrt{k})$.
  - Lower bounds: grid-like W[1]-hardness proofs to rule out $f(k) \cdot n^{o(\sqrt{k})}$ algorithms.

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?
  - ...