Gregory: The "Tree" of Knowledge

Saket Saurabh

The Institute of Mathematical Sciences, India
and University of Bergen, Norway.

Gregory Gutin's 60th Birthday Conference, 2017,
London, January 8, 2017

Birthday Quote

You leave everyone in awe. At 60, you have more going on in your life than most people have in a whole lifetime. You're an inspiration to me and everyone who knows you! You leave us all wondering...what's next for this fascinating person!?! You are awesome!!

Birthday Quote

You leave everyone in awe. At 60, you have more going on in your life than most people have in a whole lifetime. You're an inspiration to me and everyone who knows you! You leave us all wondering...what's next for this fascinating person!?! You are awesome!!

Happy 60th Birthday Gregory!

My Articles with Gregory

- Parameterized Algorithms for Directed Maximum Leaf Problems.
- Spanning Directed Trees with Many Leaves.
- Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem.
- Parameterized Study of the Test Cover Problem.
- Parameterized complexity of MaxSat Above Average.
- Fixed-Parameter Tractability of Satisfying Beyond the Number of Variables

- Noga Alon, Fedor V. Fomin, Gregory Gutin, Michael Krivelevich, Saket Saurabh: Parameterized Algorithms for Directed Maximum Leaf Problems. ICALP 2007: 352-362
- Noga Alon, Fedor V. Fomin, Gregory Gutin, Michael Krivelevich, Saket Saurabh: Spanning Directed Trees with Many Leaves. SIAM J. Discrete Math. 23(1): 466-476 (2009)
- Nathann Cohen, Fedor V. Fomin, Gregory Gutin, Eun Jung Kim, Saket Saurabh, Anders Yeo: Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem. J. Comput. Syst. Sci. 76(7): 650-662 (2010)
- Robert Crowston, Gregory Gutin, Mark Jones, Saket Saurabh, Anders Yeo: Parameterized Study of the Test Cover Problem. MFCS 2012: 283-295
- Robert Crowston, Gregory Gutin, Mark Jones, Venkatesh Raman, Saket Saurabh: Parameterized complexity of MaxSat Above Average. Theor. Comput. Sci. 511: 77-84 (2013)
- Robert Crowston, Gregory Gutin, Mark Jones, Venkatesh Raman, Saket Saurabh, Anders Yeo: Fixed-Parameter Tractability of Satisfying Beyond the Number of Variables. Algorithmica 68(3): 739-757 (2014)

My Articles with Gregory and Trees

- Parameterized Algorithms for Directed Maximum Leaf Problems – finding directed out-tree/branching with at least k leaves

My Articles with Gregory and Trees

- Parameterized Algorithms for Directed Maximum Leaf Problems – finding directed out-tree/branching with at least k leaves
- Spanning Directed Trees with Many Leaves – finding directed out-tree/branching with at least k leaves

My Articles with Gregory and Trees

- Parameterized Algorithms for Directed Maximum Leaf Problems – finding directed out-tree/branching with at least k leaves
- Spanning Directed Trees with Many Leaves – finding directed out-tree/branching with at least k leaves
- Algorithm for finding *k-vertex out-trees* and its application to *k-internal out-branching* problem. Do I need to say anything here :)

My Articles with Gregory and Trees

- Parameterized Algorithms for Directed Maximum Leaf Problems – finding directed out-tree/branching with at least k leaves
- Spanning Directed Trees with Many Leaves – finding directed out-tree/branching with at least k leaves
- Algorithm for finding *k-vertex out-trees* and its application to *k-internal out-branching* problem. Do I need to say anything here :)
- Parameterized Study of the Test Cover Problem. We used trees here also :)

My Articles with Gregory and Trees

- Parameterized Algorithms for Directed Maximum Leaf Problems – finding directed out-tree/branching with at least k leaves
- Spanning Directed Trees with Many Leaves – finding directed out-tree/branching with at least k leaves
- Algorithm for finding *k-vertex out-trees* and its application to *k-internal out-branching* problem. Do I need to say anything here :)
- Parameterized Study of the Test Cover Problem. We used trees here also :)
- Parameterized complexity of MaxSat Above Average. Oh no trees here!

My Articles with Gregory and Trees

- Parameterized Algorithms for Directed Maximum Leaf Problems – finding directed out-tree/branching with at least k leaves
- Spanning Directed Trees with Many Leaves – finding directed out-tree/branching with at least k leaves
- Algorithm for finding *k-vertex out-trees* and its application to *k-internal out-branching* problem. Do I need to say anything here :)
- Parameterized Study of the Test Cover Problem. We used trees here also :)
- Parameterized complexity of MaxSat Above Average. Oh no trees here!
- Fixed-Parameter Tractability of Satisfying Beyond the Number of Variables Used the algorithm for Tree Subgraph Isomorphism

So I hope my connection with Gregory and Trees is more clearer now!

Problems

- Finding spanning trees in a graph is polynomial time.

Problems

- Finding spanning trees in a graph is polynomial time.
- However, the moment we put certain restrictions such as
  - finding trees with at least k leaves
  - finding trees with at most k leaves
  - finding trees with at least k internal vertices
  - given a tree T on k vertices and a graph G, check whether there exists a tree isomorphic to T.

  the problems become NP-complete.

Problems

- Finding spanning trees in a graph is polynomial time.
- However, the moment we put certain restrictions such as
  - finding trees with at least k leaves
  - finding trees with at most k leaves
  - finding trees with at least k internal vertices
  - given a tree T on k vertices and a graph G, check whether there exists a tree isomorphic to T.

  the problems become NP-complete.
- They remain NP-complete for both undirected and directed graphs (digraphs).

Problems

- Finding spanning trees in a graph is polynomial time.
- However, the moment we put certain restrictions such as
  - finding trees with at least k leaves
  - finding trees with at most k leaves
  - finding trees with at least k internal vertices
  - given a tree T on k vertices and a graph G, check whether there exists a tree isomorphic to T.

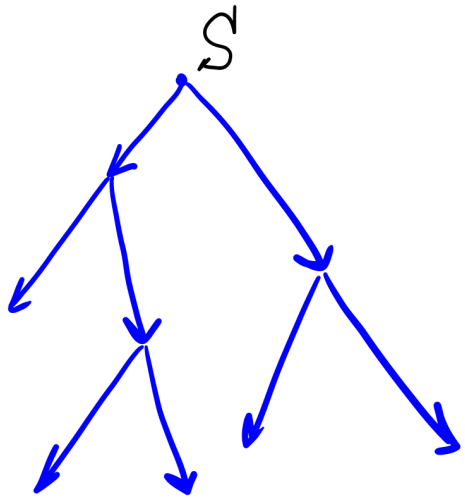
  the problems become NP-complete.
- They remain NP-complete for both undirected and directed graphs (digraphs).
- They remain NP-complete even when restricted to planar graphs or planar digraphs

Problems

- Finding spanning trees in a graph is polynomial time.
- However, the moment we put certain restrictions such as
  - finding trees with at least k leaves
  - finding trees with at most k leaves
  - finding trees with at least k internal vertices
  - given a tree T on k vertices and a graph G, check whether there exists a tree isomorphic to T.

  the problems become NP-complete.
- They remain NP-complete for both undirected and directed graphs (digraphs).
- They remain NP-complete even when restricted to planar graphs or planar digraphs even tournaments some time.

Out-Trees and Out-Branchings

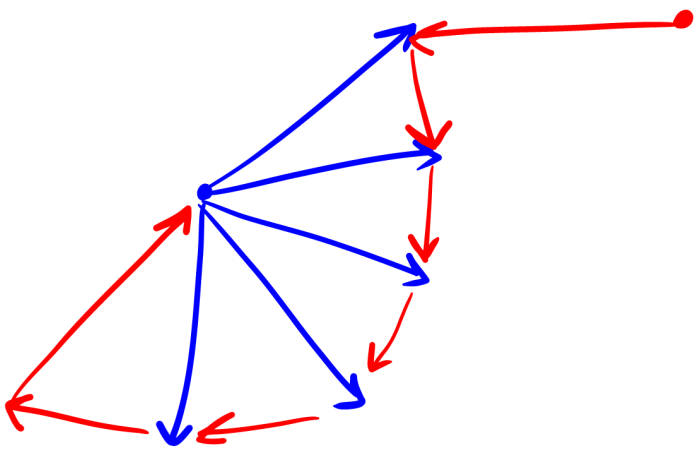


We say that a subdigraph T of a digraph D is an out-tree if T is an oriented tree with only one vertex s of in-degree zero (its root).
The vertices of T of out-degree zero are leaves.

Out-Trees and Out-Branchings

If T is a spanning out-tree, i.e. $V(T) = V(D)$, then T is an out-branching of D.

Out-tree with 5 leaves



Out-branching with one leaf

Fixed Parameter Tractable (FPT) Algorithms

For decision problems with input size n, and a parameter k, (which typically is the solution size), the goal here is to design an algorithm with running time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a function of k alone.

Problems that have such an algorithm are said to be fixed parameter tractable (FPT).

Directed Maximum Leaf Out-Branching

Directed Maximum Leaf Out-Branching (DMLOB)
Input: A digraph D and a positive integer k.
Parameter: k
Question: Does there exist a an out-branching (i.e., a rooted oriented spanning tree) in a given digraph with at least k leaves?

Our Results (2006–2007)

Spanning Directed Trees with Many Leaves

- The problem was open for sometime then. In fact very few parameterized algorithms were known for problems on digraphs.

Our Results (2006–2007)

Spanning Directed Trees with Many Leaves

- The problem was open for sometime then. In fact very few parameterized algorithms were known for problems on digraphs.

Structural Theorem/Win Win Theorem

Let D be a strongly connected digraph. Then in polynomial time we can either (a) obtain an out-branching of D with at least k leaves; or (b) obtain a path decomposition of width $\mathbf{pw}(G) = \mathcal{O}(k \log k)$ for the underlying undirected graph of D.

Our Results (2006–2007)

Spanning Directed Trees with Many Leaves

- The problem was open for sometime then. In fact very few parameterized algorithms were known for problems on digraphs.

Structural Theorem/Win Win Theorem

Let D be a strongly connected digraph. Then in polynomial time we can either (a) obtain an out-branching of D with at least k leaves; or (b) obtain a path decomposition of width $\mathbf{pw}(G) = \mathcal{O}(k \log k)$ for the underlying undirected graph of D.

- In the later case one can apply the algorithm running in time $2^{\mathcal{O}(\mathbf{pw}(G) \log \mathbf{pw}(G))} n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k(\log k)^2)} n^{\mathcal{O}(1)}$.

Follow up work and the current best

- Bonsma and Dorn used our ideas and used tree-width and showed that DMLOB can be solved in time $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$.

Follow up work and the current best

- Bonsma and Dorn used our ideas and used tree-width and showed that DMLOB can be solved in time $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$.
- Using another approach, Kneis, Langer, and Rossmanith obtained $4^k n^{\mathcal{O}(1)}$ time algorithm for DMLOB.

Follow up work and the current best

- Bonsma and Dorn used our ideas and used tree-width and showed that DMLOB can be solved in time $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$.
- Using another approach, Kneis, Langer, and Rossmanith obtained $4^k n^{\mathcal{O}(1)}$ time algorithm for DMLOB.
- Daligault, Gutin, Kim, and Yeo improved this to $3.72^k n^{\mathcal{O}(1)}$ time algorithm for DMLOB. This is the current best algorithm on the problem.

Kernelization

INFORMALLY: A kernelization algorithm is a polynomial-time transformation that transforms any given parameterized instance to an equivalent instance of the same problem, with size and parameter bounded by a function of the parameter.

Kernel: Formally

FORMALLY: A kernelization algorithm, or in short, a kernel for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that given $(x, k) \in \Sigma^* \times \mathbb{N}$, outputs in $p(|x| + k)$ time a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that

Kernel: Formally

FORMALLY: A kernelization algorithm, or in short, a kernel for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that given $(x, k) \in \Sigma^* \times \mathbb{N}$, outputs in $p(|x| + k)$ time a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that

- $(x, k) \in L \iff (x', k') \in L$,
- $|x'|, k' \leqslant f(k)$,

where f is an arbitrary computable function, and p a polynomial. Any function f as above is referred to as the size of the kernel.

Kernel: Formally

FORMALLY: A kernelization algorithm, or in short, a kernel for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that given $(x, k) \in \Sigma^* \times \mathbb{N}$, outputs in $p(|x| + k)$ time a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that

- $(x, k) \in L \iff (x', k') \in L$,
- $|x'|, k' \leqslant f(k)$,

where f is an arbitrary computable function, and p a polynomial. Any function f as above is referred to as the size of the kernel.

Polynomial kernel $\implies$ f is polynomial.

Our Results and what has happened after that

Kernel(s) for problems with no kernel: On out-trees with many leaves

- Showed that DMLOB does not have polynomial kernel

Our Results and what has happened after that

Kernel(s) for problems with no kernel: On out-trees with many leaves

- Showed that DMLOB does not have polynomial kernel but if we *fix the root* the problem has kernel.

Our Results and what has happened after that

Kernel(s) for problems with no kernel: On out-trees with many leaves

- Showed that DMLOB does not have polynomial kernel but if we *fix the root* the problem has kernel.
- Rooted DMLOB has n kernel of size $\mathcal{O}(k^3)$.

Our Results and what has happened after that

Kernel(s) for problems with no kernel: On out-trees with many leaves

- Showed that DMLOB does not have polynomial kernel but if we *fix the root* the problem has kernel.
- Rooted DMLOB has n kernel of size $\mathcal{O}(k^3)$.
- First such example in the literature. Currently, the size of the kernel is $\mathcal{O}(k^2)$ – not known whether this is optimal!

Our Results and what has happened after that

Kernel(s) for problems with no kernel: On out-trees with many leaves

- Showed that DMLOB does not have polynomial kernel but if we *fix the root* the problem has kernel.
- Rooted DMLOB has n kernel of size $\mathcal{O}(k^3)$.
- First such example in the literature. Currently, the size of the kernel is $\mathcal{O}(k^2)$ – not known whether this is optimal!
- Daligault, Gutin, Kim, and Yeo showed that Rooted DMLOB admits $\mathcal{O}(k)$ kernel on acyclic digraphs.

Our Results and what has happened after that

Kernel(s) for problems with no kernel: On out-trees with many leaves

- Showed that DMLOB does not have polynomial kernel but if we *fix the root* the problem has kernel.
- Rooted DMLOB has n kernel of size $\mathcal{O}(k^3)$.
- First such example in the literature. Currently, the size of the kernel is $\mathcal{O}(k^2)$ – not known whether this is optimal!
- Daligault, Gutin, Kim, and Yeo showed that Rooted DMLOB admits $\mathcal{O}(k)$ kernel on acyclic digraphs. Apart from planar digraphs this is the only class which is known to admit linear kernels.

Tree/Out-Tree Subgraph Isomorphism

Tree Subgraph Isomorphism
Input: A tree T and an undirected graph G.
Parameter: $k = |V(T)|$
Question: Does there exist a tree T^* in G that is isomorphic to T?

Tree/Out-Tree Subgraph Isomorphism

Tree Subgraph Isomorphism
Input: A tree T and an undirected graph G.
Parameter: $k = |V(T)|$
Question: Does there exist a tree T^* in G that is isomorphic to T?

Out-Tree Subgraph Isomorphism
Input: An out-tree T and a digraph D.
Parameter: $k = |V(T)|$
Question: Does there exist an out-tree T^* in D that is isomorphic to T?

Minimum Leaf Out-Branching

Minimum Leaf Out-Branching (k-MLOB)
Input: A digraph D and a positive integer k.
Parameter: k
Question: Does D has an out-branching with at least k internal vertices?

Minimum Leaf Out-Branching

MINIMUM LEAF OUT-BRANCHING (k-MLOB)
Input: A digraph D and a positive integer k.
Parameter: k
Question: Does D has an out-branching with at least k internal vertices?

- A graph admits an out-branching with at most one leaf if and only if it has a Hamiltonian Path.

k-MLOB

- Gutin, Razgon, and Kim designed an algorithm for k-MLOB running in time $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$ and an $\mathcal{O}(k^2)$ sized kernel.

k-MLOB

- Gutin, Razgon, and Kim designed an algorithm for k-MLOB running in time $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$ and an $\mathcal{O}(k^2)$ sized kernel.

Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem

- Designed an algorithm for Out-Tree Subgraph Isomorphism running in time 5.704^k and an algorithm for k-MLOB running in time $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$.

Our Method – An outline of the algorithms

We call a k-internal out-tree *minimal* if none of its proper *subtrees* is a k-internal out-tree, or minimal k-tree in short.

Our Method – An outline of the algorithms

We call a k-internal out-tree *minimal* if none of its proper *subtrees* is a k-internal out-tree, or minimal k-tree in short.

Structural Theorem

Let T be a k-internal out-tree. Then T is minimal if and only if number of internal nodes is exactly k and every leaf u is the only child of its parent.

Our Method – An outline of the algorithms

We call a k-internal out-tree *minimal* if none of its proper *subtrees* is a k-internal out-tree, or minimal k-tree in short.

Structural Theorem

Let T be a k-internal out-tree. Then T is minimal if and only if number of internal nodes is exactly k and every leaf u is the only child of its parent.

Forward Direction: Cannot have more than k internal vertices, else by removing any of its leaves, we obtain a subtree of T with at least k internal vertices (so internal nodes $= k$).

Our Method – An outline of the algorithms

We call a k-internal out-tree *minimal* if none of its proper *subtrees* is a k-internal out-tree, or minimal k-tree in short.

Structural Theorem

Let T be a k-internal out-tree. Then T is minimal if and only if number of internal nodes is exactly k and every leaf u is the only child of its parent.

Forward Direction: Cannot have more than k internal vertices, else by removing any of its leaves, we obtain a subtree of T with at least k internal vertices (so internal nodes $= k$). If there are sibling leaves u and w, then removing one of them provides a subtree of T with same number of internal nodes.

Our Method – An outline of the algorithms

We call a k-internal out-tree *minimal* if none of its proper *subtrees* is a k-internal out-tree, or minimal k-tree in short.

Structural Theorem

Let T be a k-internal out-tree. Then T is minimal if and only if number of internal nodes is exactly k and every leaf u is the only child of its parent.

- So a *witness* that we have an out-tree with k internal nodes is an out-tree with at most $2k$ nodes.

Our Method – An outline of the algorithms

We call a k-internal out-tree *minimal* if none of its proper *subtrees* is a k-internal out-tree, or minimal k-tree in short.

Structural Theorem

Let T be a k-internal out-tree. Then T is minimal if and only if number of internal nodes is exactly k and every leaf u is the only child of its parent.

- So a *witness* that we have an out-tree with k internal nodes is an out-tree with at most $2k$ nodes.
- So we enumerate all non-isomorphic trees on $2k$ nodes and use Out-Tree Subgraph Isomorphism algorithm to test whether such a tree is present or not.

Follow up work and the current best

- The best deterministic algorithm uses our structural lemma and runs in time $5.1^k n^{\mathcal{O}(1)}$ [Zehavi, ESA 2016]
- Randomized runs in time $4^k n^{\mathcal{O}(1)}$ and uses reduction to multilinear monomial testsing. [Zehavi, IPEC 2013]

Kernel – An open question?

- Is there linear sized vertex kernel for k-MLOB?

Spend Rest of my time on Tree Isomorphism

Tree Isomorphism

- Alon, Yuster and Zwick in their seminal paper that introduced Color-Coding gave $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time algorithm (JACM, 1995).

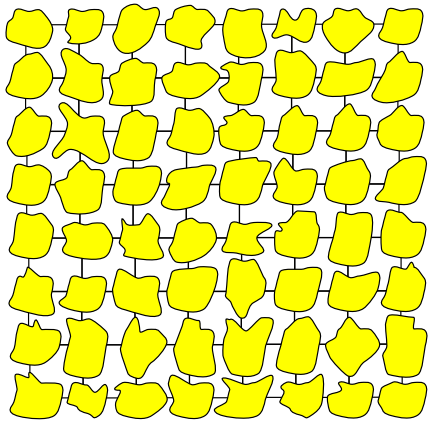
Tree Isomorphism

- Alon, Yuster and Zwick in their seminal paper that introduced Color-Coding gave $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time algorithm (JACM, 1995).
- Current best is $2^k n^{\mathcal{O}(1)}$ randomized [Koutis and Williams, ICALP 2009] and $2.618^k n^{\mathcal{O}(1)}$ deterministic (Fomin et al, JACM, 2016).

A Special Graph Class – Planar Graphs

- Tree Subgraph Isomorphism, even finding a path of length k is NP-complete on this graph class.

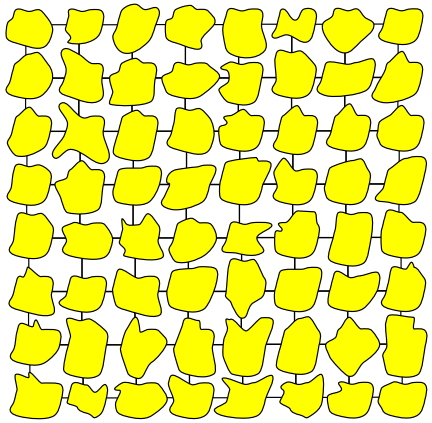
Grid minors and treewidth



Grid Minor Theorem Robertson, Seymour; GM V

There is a function f such that $\mathbf{tw}(G) \geqslant f(k)$ implies the existence of a $k \times k$ grid minor in G.

Grid minors and treewidth

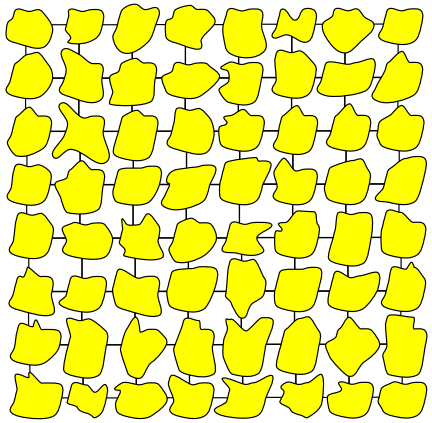


Grid Minor Theorem Robertson, Seymour; GM V

There is a function f such that $\mathbf{tw}(G) \geqslant f(k)$ implies the existence of a $k \times k$ grid minor in G.

- **Upper bound**: $f(k) \in \tilde{\mathcal{O}}(k^{19})$ (Chuzhoy '15)

Grid minors and treewidth

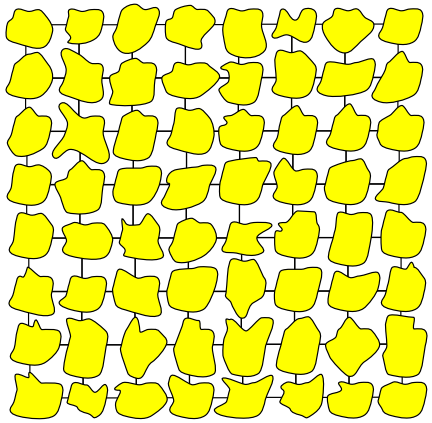


Grid Minor Theorem Robertson, Seymour; GM V

There is a function f such that $\mathbf{tw}(G) \geqslant f(k)$ implies the existence of a $k \times k$ grid minor in G.

- **Upper bound**: $f(k) \in \tilde{\mathcal{O}}(k^{19})$ (Chuzhoy '15)
- **Lower bound**: $f(k) \in \Omega(k^2 \log k)$

Grid minors and treewidth

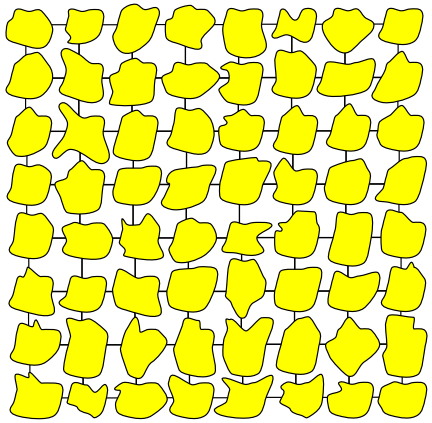


Grid Minor Theorem Robertson, Seymour; GM V

There is a function f such that $\mathbf{tw}(G) \geqslant f(k)$ implies the existence of a $k \times k$ grid minor in G.

- **Upper bound**: $f(k) \in \tilde{\mathcal{O}}(k^{19})$ (Chuzhoy '15)
- **Lower bound**: $f(k) \in \Omega(k^2 \log k)$
- **Minor-free**: $f(k) \in \Theta(k)$ for graphs excluding a fixed minor.

Grid minors and treewidth

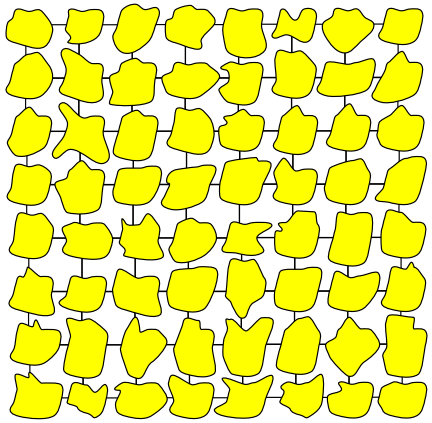


Grid Minor Theorem Robertson, Seymour; GM V

There is a function f such that $\mathbf{tw}(G) \geqslant f(k)$ implies the existence of a $k \times k$ grid minor in G.

- **Upper bound**: $f(k) \in \tilde{\mathcal{O}}(k^{19})$ (Chuzhoy '15)
- **Lower bound**: $f(k) \in \Omega(k^2 \log k)$
- **Minor-free**: $f(k) \in \Theta(k)$ for graphs excluding a fixed minor.
  - **Planar**: $\mathbf{tw}(G) > 4.5k + 1$ implies a $k \times k$ grid minor.

Grid minors and treewidth



Grid Minor Theorem

Robertson, Seymour; GM V

There is a function f such that $\mathbf{tw}(G) \geqslant f(k)$ implies the existence of a $k \times k$ grid minor in G.

- **Upper bound**: $f(k) \in \tilde{\mathcal{O}}(k^{19})$ (Chuzhoy '15)
- **Lower bound**: $f(k) \in \Omega(k^2 \log k)$
- **Minor-free**: $f(k) \in \Theta(k)$ for graphs excluding a fixed minor.
  - **Planar**: $\mathbf{tw}(G) > 4.5k + 1$ implies a $k \times k$ grid minor.
  - **H-minor-free**: $\mathbf{tw}(G) > c_H \cdot k$ implies a $k \times k$ grid minor.

Bidimensionality

- k-Path: Is there a simple path on k vertices in a graph?

Bidimensionality

- k-Path: Is there a simple path on k vertices in a graph?
- **Goal**: $2^{\mathcal{O}(\sqrt{k})} \cdot n$ algorithm for planar graphs.

Bidimensionality

- k-Path: Is there a simple path on k vertices in a graph?
- **Goal**: $2^{\mathcal{O}(\sqrt{k})} \cdot n$ algorithm for planar graphs.
- **Fact 1**: k-Path can be solved in time $2^{\mathcal{O}(w)} \cdot n$ on a tree decomposition of width w.

Bidimensionality

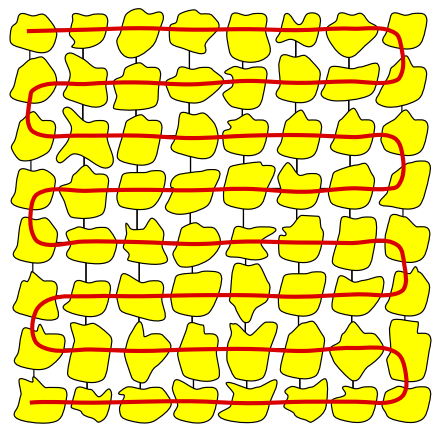
- k-Path: Is there a simple path on k vertices in a graph?
- **Goal**: $2^{\mathcal{O}(\sqrt{k})} \cdot n$ algorithm for planar graphs.
- **Fact 1**: k-Path can be solved in time $2^{\mathcal{O}(w)} \cdot n$ on a tree decomposition of width w.
- **Fact 2**: If there is $\sqrt{k} \times \sqrt{k}$ grid minor, then there is a k-path.

Algorithm

- Approximate treewidth.

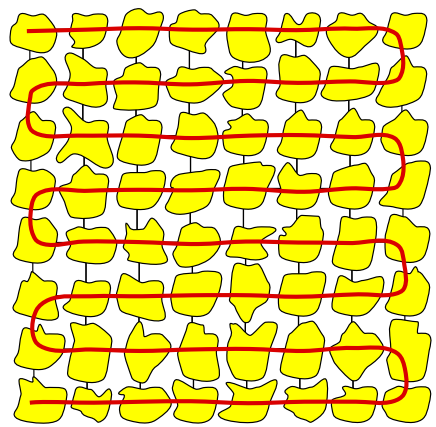
Algorithm

- Approximate treewidth.
- If $\mathbf{tw} \geqslant 100\sqrt{k}$ then there is a $\sqrt{k} \times \sqrt{k}$ grid minor, so also a k-path.



Algorithm

- Approximate treewidth.
- If $\mathbf{tw} \geqslant 100\sqrt{k}$ then there is a $\sqrt{k} \times \sqrt{k}$ grid minor, so also a k-path.
- Otherwise we have a tree decomposition of width $\mathcal{O}(\sqrt{k})$. Apply dynamic programming.



Bidimensionality: drawbacks

- The argument is elegant, but very delicate.

Bidimensionality: drawbacks

- The argument is elegant, but very delicate.
  - Assumption "large grid minor $\Rightarrow$ certain answer" is very strong.

Bidimensionality: drawbacks

- The argument is elegant, but very delicate.
  - Assumption "large grid minor $\Rightarrow$ certain answer" is very strong.
- DIRECTED k-PATH: simple path on k vertices in a **directed** graph.

Bidimensionality: drawbacks

- The argument is elegant, but very delicate.
  - Assumption "large grid minor $\Rightarrow$ certain answer" is very strong.
- DIRECTED k-PATH: simple path on k vertices in a **directed** graph.
  - A large grid minor in the underlying undirected graph tells little about directed k-paths.

Bidimensionality: drawbacks

- The argument is elegant, but very delicate.
  - Assumption "large grid minor $\Rightarrow$ certain answer" is very strong.
- DIRECTED k-PATH: simple path on k vertices in a **directed** graph.
  - A large grid minor in the underlying undirected graph tells little about directed k-paths.
  - Similar problem for searching for k-path of maximum weight, or a cycle of length exactly k, ...

Bidimensionality: drawbacks

- The argument is elegant, but very delicate.
  - Assumption "large grid minor $\Rightarrow$ certain answer" is very strong.
- DIRECTED k-PATH: simple path on k vertices in a **directed** graph.
  - A large grid minor in the underlying undirected graph tells little about directed k-paths.
  - Similar problem for searching for k-path of maximum weight, or a cycle of length exactly k, ...
- Can such problems be solved in time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ on planar graphs?

Bidimensionality: drawbacks

- The argument is elegant, but very delicate.
  - Assumption "large grid minor $\Rightarrow$ certain answer" is very strong.
- DIRECTED k-PATH: simple path on k vertices in a **directed** graph.
  - A large grid minor in the underlying undirected graph tells little about directed k-paths.
  - Similar problem for searching for k-path of maximum weight, or a cycle of length exactly k, ...
- Can such problems be solved in time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ on planar graphs?
  - Until Recently, no better algorithms were known than $2^{\mathcal{O}(k)} \cdot n^c$ inherited from the general setting.

Bidimensionality: drawbacks

- The argument is elegant, but very delicate.
  - Assumption "large grid minor $\Rightarrow$ certain answer" is very strong.
- DIRECTED k-PATH: simple path on k vertices in a **directed** graph.
  - A large grid minor in the underlying undirected graph tells little about directed k-paths.
  - Similar problem for searching for k-path of maximum weight, or a cycle of length exactly k, ...
- Can such problems be solved in time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ on planar graphs?
  - Until Recently, no better algorithms were known than $2^{\mathcal{O}(k)} \cdot n^c$ inherited from the general setting.
- **General**: Let a *k-pattern* be a vertex subset P such that $|P| \leqslant k$ and $G[P]$ is connected.

Bidimensionality: drawbacks

- The argument is elegant, but very delicate.
  - Assumption "large grid minor $\Rightarrow$ certain answer" is very strong.
- DIRECTED k-PATH: simple path on k vertices in a **directed** graph.
  - A large grid minor in the underlying undirected graph tells little about directed k-paths.
  - Similar problem for searching for k-path of maximum weight, or a cycle of length exactly k, ...
- Can such problems be solved in time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ on planar graphs?
  - Until Recently, no better algorithms were known than $2^{\mathcal{O}(k)} \cdot n^c$ inherited from the general setting.
- **General**: Let a *k-pattern* be a vertex subset P such that $|P| \leqslant k$ and $G[P]$ is connected.
  - We are looking with k-patterns with some prescribed property.

Bidimensionality: drawbacks

- The argument is elegant, but very delicate.
  - Assumption "large grid minor $\Rightarrow$ certain answer" is very strong.
- DIRECTED k-PATH: simple path on k vertices in a **directed** graph.
  - A large grid minor in the underlying undirected graph tells little about directed k-paths.
  - Similar problem for searching for k-path of maximum weight, or a cycle of length exactly k, ...
- Can such problems be solved in time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ on planar graphs?
  - Until Recently, no better algorithms were known than $2^{\mathcal{O}(k)} \cdot n^c$ inherited from the general setting.
- **General**: Let a *k-pattern* be a vertex subset P such that $|P| \leqslant k$ and $G[P]$ is connected.
  - We are looking with k-patterns with some prescribed property.
  - **Idea**: Find a small family of subgraphs of small treewidth that cover every pattern.

Pattern coverage

Main result

Suppose G is a planar graph on n vertices, and suppose k is a positive integer. Then there exists a family $\mathcal{F}$ of subsets of $V(G)$ such that:

(a) $|\mathcal{F}| \leqslant 2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ for some constant c;

(b) For each $A \in \mathcal{F}$, we have $\mathbf{tw}(G[A]) \leqslant \tilde{\mathcal{O}}(\sqrt{k})$;

(c) For every k-pattern P, there is some $A \in \mathcal{F}$ such that $P \subseteq A$.

Moreover, $\mathcal{F}$ can be constructed in randomized time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$.

Pattern coverage

Main result

Suppose G is a planar graph on n vertices, and suppose k is a positive integer. Then there exists a family $\mathcal{F}$ of subsets of $V(G)$ such that:

(a) $|\mathcal{F}| \leqslant 2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ for some constant c;

(b) For each $A \in \mathcal{F}$, we have $\mathbf{tw}(G[A]) \leqslant \tilde{\mathcal{O}}(\sqrt{k})$;

(c) For every k-pattern P, there is some $A \in \mathcal{F}$ such that $P \subseteq A$.

Moreover, $\mathcal{F}$ can be constructed in randomized time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$.

- Directed k-Path algorithm:

Pattern coverage

Main result

Suppose G is a planar graph on n vertices, and suppose k is a positive integer. Then there exists a family $\mathcal{F}$ of subsets of $V(G)$ such that:

(a) $|\mathcal{F}| \leqslant 2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ for some constant c;

(b) For each $A \in \mathcal{F}$, we have $\mathbf{tw}(G[A]) \leqslant \tilde{\mathcal{O}}(\sqrt{k})$;

(c) For every k-pattern P, there is some $A \in \mathcal{F}$ such that $P \subseteq A$.

Moreover, $\mathcal{F}$ can be constructed in randomized time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$.

- Directed k-Path algorithm:
  - Construct $\mathcal{F}$.

Pattern coverage

Main result

Suppose G is a planar graph on n vertices, and suppose k is a positive integer. Then there exists a family $\mathcal{F}$ of subsets of $V(G)$ such that:

(a) $|\mathcal{F}| \leqslant 2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ for some constant c;

(b) For each $A \in \mathcal{F}$, we have $\mathbf{tw}(G[A]) \leqslant \tilde{\mathcal{O}}(\sqrt{k})$;

(c) For every k-pattern P, there is some $A \in \mathcal{F}$ such that $P \subseteq A$.

Moreover, $\mathcal{F}$ can be constructed in randomized time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$.

- Directed k-Path algorithm:
  - Construct $\mathcal{F}$.
  - For each $A \in \mathcal{F}$, try to find a k-path in $G[A]$ by DP.

Pattern coverage

Main result

Suppose G is a planar graph on n vertices, and suppose k is a positive integer. Then there exists a family $\mathcal{F}$ of subsets of $V(G)$ such that:

(a) $|\mathcal{F}| \leqslant 2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ for some constant c;

(b) For each $A \in \mathcal{F}$, we have $\mathbf{tw}(G[A]) \leqslant \tilde{\mathcal{O}}(\sqrt{k})$;

(c) For every k-pattern P, there is some $A \in \mathcal{F}$ such that $P \subseteq A$.

Moreover, $\mathcal{F}$ can be constructed in randomized time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$.

- Directed k-Path algorithm:
  - Construct $\mathcal{F}$.
  - For each $A \in \mathcal{F}$, try to find a k-path in $G[A]$ by DP.
  - **Running time:** $(2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c) \cdot (2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n) = 2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^{c+1}$

Applications

- Applies to a range of pattern-searching problems:

Applications

- Applies to a range of pattern-searching problems:
  - (directed) k-path of largest weight;

Applications

- Applies to a range of pattern-searching problems:
  - (directed) k-path of largest weight;
  - cycle of length exactly k;

Applications

- Applies to a range of pattern-searching problems:
  - (directed) k-path of largest weight;
  - cycle of length exactly k;
  - k-local search for Planar Vertex Cover.

Applications

- Applies to a range of pattern-searching problems:
  - (directed) k-path of largest weight;
  - cycle of length exactly k;
  - k-local search for Planar Vertex Cover.
  - **Needed**: $2^{\tilde{\mathcal{O}}(w)} \cdot n^c$ algorithm on graphs of treewidth w.

Applications

- Applies to a range of pattern-searching problems:
  - (directed) k-path of largest weight;
  - cycle of length exactly k;
  - k-local search for Planar Vertex Cover.
  - **Needed**: $2^{\tilde{\mathcal{O}}(w)} \cdot n^c$ algorithm on graphs of treewidth w.
- Subgraph Isomorphism:
  Given G and H with $|V(H)| \leqslant k$, is H a subgraph of G?

Applications

- Applies to a range of pattern-searching problems:
  - (directed) k-path of largest weight;
  - cycle of length exactly k;
  - k-local search for Planar Vertex Cover.
  - **Needed**: $2^{\tilde{\mathcal{O}}(w)} \cdot n^c$ algorithm on graphs of treewidth w.
- Subgraph Isomorphism:
  Given G and H with $|V(H)| \leqslant k$, is H a subgraph of G?
  - When H is connected and has maximum degree bounded by a constant, we obtain an algorithm with running time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$.

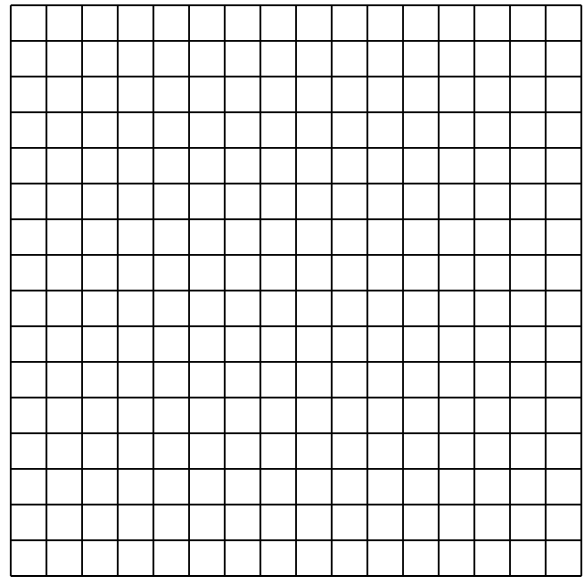
Applications

- Applies to a range of pattern-searching problems:
  - (directed) k-path of largest weight;
  - cycle of length exactly k;
  - k-local search for Planar Vertex Cover.
  - **Needed**: $2^{\tilde{\mathcal{O}}(w)} \cdot n^c$ algorithm on graphs of treewidth w.
- Subgraph Isomorphism:
  Given G and H with $|V(H)| \leqslant k$, is H a subgraph of G?
  - When H is connected and has maximum degree bounded by a constant, we obtain an algorithm with running time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$.
  - Without the maxdeg bound, we get running time $2^{\mathcal{O}(k/\log k)} \cdot n^c$ using (Bodlaender, Nederlof, van der Zanden; ICALP'16).

Applications

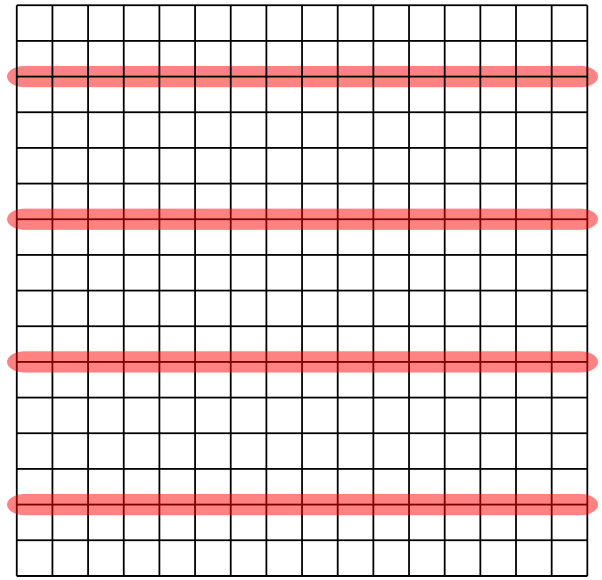
- Applies to a range of pattern-searching problems:
  - (directed) k-path of largest weight;
  - cycle of length exactly k;
  - k-local search for Planar Vertex Cover.
  - **Needed**: $2^{\tilde{\mathcal{O}}(w)} \cdot n^c$ algorithm on graphs of treewidth w.
- Subgraph Isomorphism:
  Given G and H with $|V(H)| \leqslant k$, is H a subgraph of G?
  - When H is connected and has maximum degree bounded by a constant, we obtain an algorithm with running time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$.
  - Without the maxdeg bound, we get running time $2^{\mathcal{O}(k/\log k)} \cdot n^c$ using (Bodlaender, Nederlof, van der Zanden; ICALP'16).
  - This running time is tight under ETH! (Bodlaender et al.)

Example: grid



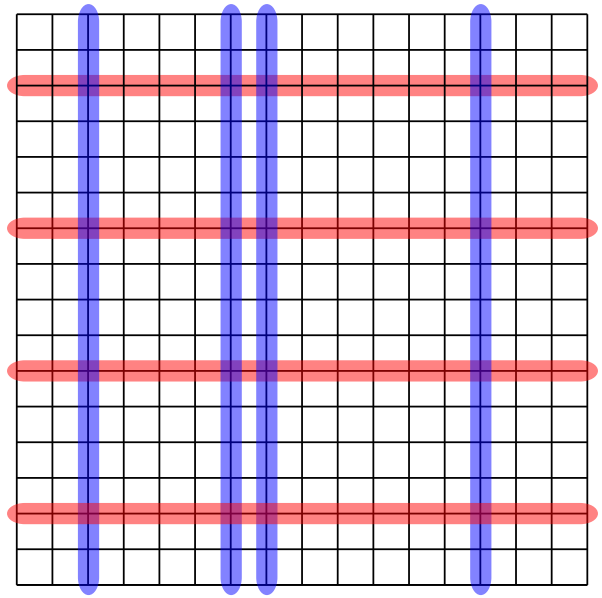
- Take a $k \times k$ grid.

Example: grid



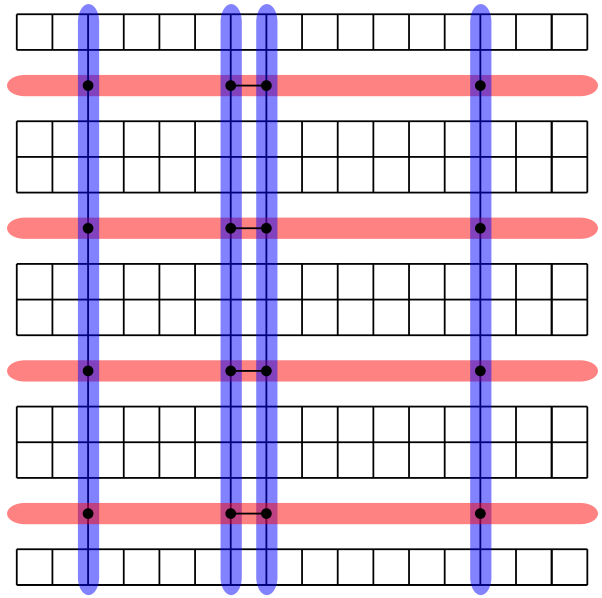
- Take a $k \times k$ grid.
- **Baker**: Guess an index modulo $\sqrt{k}$ s.t. there are $\leqslant \sqrt{k}$ vertices of the pattern in the corresponding rows.

Example: grid



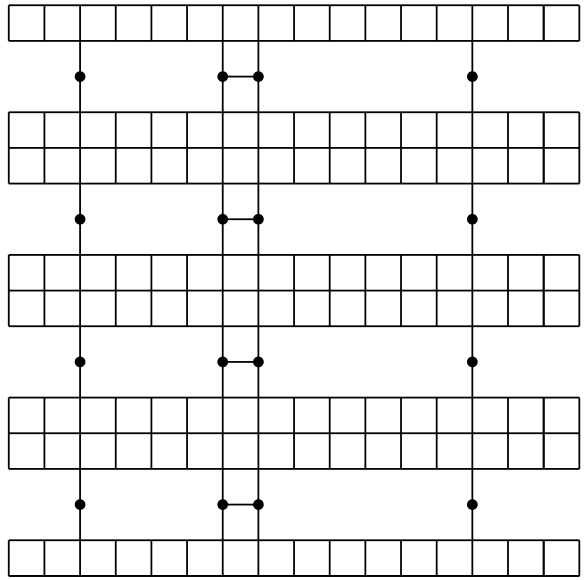
- Take a $k \times k$ grid.
- **Baker**: Guess an index modulo $\sqrt{k}$ s.t. there are $\leqslant \sqrt{k}$ vertices of the pattern in the corresponding rows.
- Guess columns in which these vertices lie, $\binom{k}{\sqrt{k}}$ options.

Example: grid



- Take a $k \times k$ grid.
- **Baker**: Guess an index modulo $\sqrt{k}$ s.t. there are $\leqslant \sqrt{k}$ vertices of the pattern in the corresponding rows.
- Guess columns in which these vertices lie, $\binom{k}{\sqrt{k}}$ options.
- Remove the rows apart from intersections with columns. What remains has treewidth $\mathcal{O}(\sqrt{k})$.

Example: grid



- Take a $k \times k$ grid.
- **Baker**: Guess an index modulo $\sqrt{k}$ s.t. there are $\leqslant \sqrt{k}$ vertices of the pattern in the corresponding rows.
- Guess columns in which these vertices lie, $\binom{k}{\sqrt{k}}$ options.
- Remove the rows apart from intersections with columns. What remains has treewidth $\mathcal{O}(\sqrt{k})$.
- In total, $\sqrt{k} \cdot \binom{k}{\sqrt{k}} = 2^{\mathcal{O}(\sqrt{k} \log k)}$ possible guesses.

Once More Trees

- k-Distinct In- and Out-Branchings (finding an out-branching and in-branching in a digraph D that do not share at least k arcs in common). Bang-Jensen, SS and Simonsen showed that the problem is FPT on strongly connected digraphs.

Once More Trees

- k-Distinct In- and Out-Branchings (finding an out-branching and in-branching in a digraph D that do not share at least k arcs in common). Bang-Jensen, SS and Simonsen showed that the problem is FPT on strongly connected digraphs.
- Gutin, Reidl and Wahlström showed that k-Distinct In- and Out-Branchings is FPT on digraphs.

Let me wish Gregory once again Happy Birthday!

Let me wish Gregory once again Happy Birthday!

Thank You!

Any Questions?