Single-exponential parameterized algorithms: good and bad news

Ignasi Sau

CNRS, LIRMM, Montpellier, France

Journées AGAPE

<ロ> <四> <四> <四> <三</p>

Good news

- Parameterizing by branchwidth
 - Graphs on surfaces
 - Main ideas of our approach
 - Extensions
- Parameterizing by the size of the solution



• Courcelle's theorem (1988):

Graph problems expressible in Monadic Second Order Logic (MSOL) can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ in graphs *G* such that $\mathbf{tw}(G) \leq k$.

• **Problem**: f(k) can be huge!!! (for instance, $f(k) = 2^{3^{4^{5^{6^{k}}}}}$)

A single-exponential parameterized algorithm is a FPT algo s.t.

$$f(k)=2^{\mathcal{O}(k)}.$$

• Courcelle's theorem (1988):

Graph problems expressible in Monadic Second Order Logic (MSOL) can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ in graphs *G* such that $\mathbf{tw}(G) \leq k$.

• **Problem**: f(k) can be huge!!! (for instance, $f(k) = 2^{3^{4^{5^{6^{n}}}}}$)

• A single-exponential parameterized algorithm is a FPT algo s.t.

$$f(k)=2^{\mathcal{O}(k)}.$$

• Courcelle's theorem (1988):

Graph problems expressible in Monadic Second Order Logic (MSOL) can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ in graphs *G* such that $\mathbf{tw}(G) \leq k$.

• **Problem**: f(k) can be huge!!! (for instance, $f(k) = 2^{3^{4^{5^{6^{k}}}}}$)

A single-exponential parameterized algorithm is a FPT algo s.t.

$$f(\mathbf{k}) = 2^{\mathcal{O}(\mathbf{k})}.$$

Good news

- Parameterizing by branchwidth
 - Graphs on surfaces
 - Main ideas of our approach
 - Extensions
- Parameterizing by the size of the solution



Next subsection is...

Good news

• Parameterizing by branchwidth

- Graphs on surfaces
- Main ideas of our approach
- Extensions
- Parameterizing by the size of the solution



Next subsubsection is...

Good news

Parameterizing by branchwidth

Graphs on surfaces

- Main ideas of our approach
- Extensions

• Parameterizing by the size of the solution

2 Bad news

Objective:

framework to obtain **single-exponential parameterized algorithms** for a class of NP-hard problems in **graphs embedded on surfaces**.

[Rué, S., Thilikos. ICALP'10]

Branch decompositions and branchwidth

- A branch decomposition of a graph G = (V, E) is tuple (T, μ) where:
 - *T* is a tree where all the internal nodes have degree 3.
 - μ is a bijection between the leaves of *T* and *E*(*G*).
- Each edge $e \in T$ partitions E(G) into two sets A_e and B_e .
- For each $e \in E(T)$, we define $\operatorname{mid}(e) = V(A_e) \cap V(B_e)$.
- The width of a branch decomposition is $\max_{e \in E(T)} |\mathbf{mid}(e)|$.
- The branchwidth of a graph G (denoted bw(G)) is the minimum width over all branch decompositions of G:

 $\mathbf{bw}(G) = \min_{(T,\mu)} \max_{e \in E(T)} |\mathbf{mid}(e)|$

ふりん 叫 ふぼとんぼう (四)

Branch decompositions and branchwidth

- A branch decomposition of a graph G = (V, E) is tuple (T, μ) where:
 - *T* is a tree where all the internal nodes have degree 3.
 - μ is a bijection between the leaves of *T* and *E*(*G*).
- Each edge $e \in T$ partitions E(G) into two sets A_e and B_e .
- For each $e \in E(T)$, we define $\operatorname{mid}(e) = V(A_e) \cap V(B_e)$.
- The width of a branch decomposition is $\max_{e \in E(T)} |\mathbf{mid}(e)|$.
- The branchwidth of a graph *G* (denoted **bw**(*G*)) is the minimum width over all branch decompositions of *G*:

$$\mathbf{bw}(G) = \min_{(T,\mu)} \max_{e \in E(T)} |\mathbf{mid}(e)|$$

・ロト・(四)・(日)・(日)・(日)・(日)

• **SURFACE** = TOPOLOGICAL SPACE, LOCALLY "FLAT"





9





• Surface Classification Theorem:

any compact, connected and without boundary surface can be obtained from the sphere S^2 by adding handles and cross-caps.

• Orientable surfaces:

obtained by adding $g \ge 0$ handles to the sphere \mathbb{S}^2 , obtaining the g-torus \mathbb{T}_g with Euler genus $\mathbf{eg}(\mathbb{T}_g) = 2g$.

• Non-orientable surfaces:

obtained by adding h > 0 cross-caps to the sphere \mathbb{S}^2 , obtaining a non-orientable surface \mathbb{P}_h with Euler genus $\mathbf{eg}(\mathbb{P}_h) = h$.

• Surface Classification Theorem:

any compact, connected and without boundary surface can be obtained from the sphere S^2 by adding handles and cross-caps.

• Orientable surfaces:

obtained by adding $g \ge 0$ handles to the sphere \mathbb{S}^2 , obtaining the *g*-torus \mathbb{T}_g with Euler genus $eg(\mathbb{T}_g) = 2g$.

• Non-orientable surfaces:

obtained by adding h > 0 cross-caps to the sphere \mathbb{S}^2 , obtaining a non-orientable surface \mathbb{P}_h with Euler genus $eg(\mathbb{P}_h) = h$.

• Surface Classification Theorem:

any compact, connected and without boundary surface can be obtained from the sphere S^2 by adding handles and cross-caps.

Orientable surfaces:

obtained by adding $g \ge 0$ handles to the sphere \mathbb{S}^2 , obtaining the *g*-torus \mathbb{T}_g with Euler genus $eg(\mathbb{T}_g) = 2g$.

• Non-orientable surfaces:

obtained by adding h > 0 cross-caps to the sphere \mathbb{S}^2 , obtaining a non-orientable surface \mathbb{P}_h with Euler genus $eg(\mathbb{P}_h) = h$.

EMBEDDED GRAPH: GRAPH DRAWN ON A SURFACE, NO CROSSINGS



• The Euler genus of a graph G, eg(G), is the least Euler genus of the surfaces in which G can be embedded.

EMBEDDED GRAPH: GRAPH DRAWN ON A SURFACE, NO CROSSINGS



• The Euler genus of a graph G, **eg**(G), is the least Euler genus of the surfaces in which G can be embedded.

- Applied in a bottom-up fashion on a rooted branch decomposition of the input graph *G*.
- For each graph problem, DP requires the suitable definition of tables encoding how potential (global) solutions are restricted to a middle set mid(e).
- The size of the tables reflects the dependence on $k = |\mathbf{mid}(e)|$ in the running time of the DP.
- The precise definition of the tables of the DP depends on each particular problem.

- Applied in a bottom-up fashion on a rooted branch decomposition of the input graph *G*.
- For each graph problem, DP requires the suitable definition of tables encoding how potential (global) solutions are restricted to a middle set mid(e).
- The size of the tables reflects the dependence on k = |mid(e)| in the running time of the DP.
- The precise definition of the tables of the DP depends on each particular problem.

How can we certificate a solution in a middle set mid(e)?

- A subset of vertices of mid(e) (not restricted by some global condition).
 Examples: VERTEX COVER, DOMINATING SET.
 The size of the tables is bounded by 2^{O(k)}.
- A connected pairing of vertices of mid(e).
 Examples: LONGEST PATH, CYCLE PACKING, HAMILTONIAN CYCLE. The # of pairings in a set of k elements is k^{O(k)} = 2^{O(k log k)}... OK for planar graphs [Dom, Penninkx, Bodlaender, Fomin. ESA'05]; OK for graphs on surfaces [Dom, Fomin, Thilikos. SWAT'06].
 - Connected packing of vertices of mid(e) into subsets of arbitrary size. Examples: CONNECTED VERTEX COVER, MAX LEAF SPANNING TREE. Again, # of packings in a set of k elements is 2^{O(k log k)}.

How can we certificate a solution in a middle set mid(e)?

- A subset of vertices of mid(e) (not restricted by some global condition).
 Examples: VERTEX COVER, DOMINATING SET.
 The size of the tables is bounded by 2^{O(k)}.
- A connected pairing of vertices of mid(e).
 Examples: LONGEST PATH, CYCLE PACKING, HAMILTONIAN CYCLE. The # of pairings in a set of k elements is k^{O(k)} = 2^{O(k log k)}... OK for planar graphs [Dom, Penninkx, Bodlaender, Fomin. ESA 05]; OK for graphs on surfaces [Dom, Fomin, Thilikos. SWAT06].
 - Connected packing of vertices of mid(e) into subsets of arbitrary size. Examples: CONNECTED VERTEX COVER, MAX LEAF SPANNING TREE. Again, # of packings in a set of k elements is 2^{O(k log k)}.

How can we certificate a solution in a middle set mid(e)?

- A subset of vertices of mid(e) (not restricted by some global condition).
 Examples: VERTEX COVER, DOMINATING SET.
 The size of the tables is bounded by 2^{O(k)}.
- A connected pairing of vertices of mid(e).
 Examples: LONGEST PATH, CYCLE PACKING, HAMILTONIAN CYCLE. The # of pairings in a set of k elements is k^{O(k)} = 2^{O(k log k)}... OK for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. ESA'05]; OK for graphs on surfaces [Dorn, Fomin, Thilikos. SWAT'06].
 - Connected packing of vertices of **mid**(*e*) into subsets of arbitrary size. **Examples**: CONNECTED VERTEX COVER, MAX LEAF SPANNING TREE. Again, # of packings in a set of *k* elements is $2^{\Theta(k \log k)}$.

How can we certificate a solution in a middle set mid(e)?

- A subset of vertices of mid(e) (not restricted by some global condition).
 Examples: VERTEX COVER, DOMINATING SET.
 The size of the tables is bounded by 2^{O(k)}.
- A connected pairing of vertices of mid(e).
 Examples: LONGEST PATH, CYCLE PACKING, HAMILTONIAN CYCLE.
 The # of pairings in a set of k elements is k^{O(k)} = 2^{O(k log k)}...
 OK for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. ESA 05];
 OK for graphs on surfaces [Dorn, Fomin, Thilikos. SWAT06].
 - Connected packing of vertices of mid(e) into subsets of arbitrary size. **Examples**: CONNECTED VERTEX COVER, MAX LEAF SPANNING TREE. Again, # of packings in a set of k elements is $2^{\Theta(k \log k)}$.

How can we certificate a solution in a middle set mid(e)?

- A subset of vertices of mid(e) (not restricted by some global condition).
 Examples: VERTEX COVER, DOMINATING SET.
 The size of the tables is bounded by 2^{O(k)}.
- A connected pairing of vertices of mid(e).
 Examples: LONGEST PATH, CYCLE PACKING, HAMILTONIAN CYCLE.
 The # of pairings in a set of k elements is $k^{\Theta(k)} = 2^{\Theta(k \log k)}$...

OK for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. *ESA'05*]; OK for graphs on surfaces [Dorn, Fomin, Thilikos. *SWAT'06*].

Connected packing of vertices of mid(e) into subsets of arbitrary size. **Examples**: CONNECTED VERTEX COVER, MAX LEAF SPANNING TREE. Again, # of packings in a set of k elements is $2^{\Theta(k \log k)}$.

How can we certificate a solution in a middle set mid(e)?

- A subset of vertices of mid(e) (not restricted by some global condition).
 Examples: VERTEX COVER, DOMINATING SET.
 The size of the tables is bounded by 2^{O(k)}.
- A connected pairing of vertices of mid(e).
 Examples: LONGEST PATH, CYCLE PACKING, HAMILTONIAN CYCLE. The # of pairings in a set of k elements is k^{O(k)} = 2^{O(k log k)}... OK for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. ESA'05]; OK for graphs on surfaces [Dorn, Fomin, Thilikos. SWAT'06].
- Connected packing of vertices of mid(e) into subsets of arbitrary size. Examples: CONNECTED VERTEX COVER, MAX LEAF SPANNING TREE. Again, # of packings in a set of k elements is 2^{Θ(k log k)}.

How can we certificate a solution in a middle set mid(e)?

- A subset of vertices of mid(e) (not restricted by some global condition).
 Examples: VERTEX COVER, DOMINATING SET.
 The size of the tables is bounded by 2^{O(k)}.
- A connected pairing of vertices of mid(e).
 Examples: LONGEST PATH, CYCLE PACKING, HAMILTONIAN CYCLE. The # of pairings in a set of k elements is k^{O(k)} = 2^{O(k log k)}... OK for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. ESA'05]; OK for graphs on surfaces [Dorn, Fomin, Thilikos. SWAT'06].
- Connected packing of vertices of mid(e) into subsets of arbitrary size. Examples: CONNECTED VERTEX COVER, MAX LEAF SPANNING TREE. Again, # of packings in a set of k elements is 2^{O(k log k)}.

How can we certificate a solution in a middle set mid(e)?

- A subset of vertices of mid(e) (not restricted by some global condition).
 Examples: VERTEX COVER, DOMINATING SET.
 The size of the tables is bounded by 2^{O(k)}.
- A connected pairing of vertices of mid(e).
 Examples: LONGEST PATH, CYCLE PACKING, HAMILTONIAN CYCLE. The # of pairings in a set of k elements is k^{O(k)} = 2^{O(k log k)}... OK for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. ESA'05]; OK for graphs on surfaces [Dorn, Fomin, Thilikos. SWAT'06].
- Solution Connected packing of vertices of mid(e) into subsets of arbitrary size. **Examples**: CONNECTED VERTEX COVER, MAX LEAF SPANNING TREE. Again, # of packings in a set of k elements is $2^{\Theta(k \log k)}$.

How can we certificate a solution in a middle set mid(e)?

- A subset of vertices of mid(e) (not restricted by some global condition).
 Examples: VERTEX COVER, DOMINATING SET.
 The size of the tables is bounded by 2^{O(k)}.
- A connected pairing of vertices of mid(e).
 Examples: LONGEST PATH, CYCLE PACKING, HAMILTONIAN CYCLE. The # of pairings in a set of k elements is k^{O(k)} = 2^{O(k log k)}... OK for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. ESA'05]; OK for graphs on surfaces [Dorn, Fomin, Thilikos. SWAT'06].
- Solution Connected packing of vertices of mid(e) into subsets of arbitrary size. **Examples**: CONNECTED VERTEX COVER, MAX LEAF SPANNING TREE. Again, # of packings in a set of k elements is $2^{\Theta(k \log k)}$.

Next subsubsection is...

Good news

Parameterizing by branchwidth

- Graphs on surfaces
- Main ideas of our approach
- Extensions
- Parameterizing by the size of the solution










Nooses

Let *G* be a graph embedded in a surface Σ . A noose is a subset of Σ homeomorphic to \mathbb{S}^1 that meets *G* only at vertices.



Key idea for planar graphs [Dorn et al. ESA'05]:

- Sphere cut decomposition: Branch decomposition where the vertices in each mid(e) are situated around a noose.
 [Seymour and Thomas. Combinatorica'94]
- Recall that the size of the tables of a DP algorithm depends on how many ways a partial solution can intersect **mid**(*e*).
- In how many ways can we draw polygons inside a circle such that they touch the circle only on its k vertices and they do not intersect?

$$ON(k) = \frac{1}{k+1} \binom{2k}{k} \sim \frac{4^k}{\sqrt{\pi}k^{3/2}} \approx 4^k.$$

$$(\Box \to 4^{\mathbb{R}} \to 4$$

Key idea for planar graphs [Dorn et al. ESA'05]:

- Sphere cut decomposition: Branch decomposition where the vertices in each mid(e) are situated around a noose.
 [Seymour and Thomas. Combinatorica'94]
- Recall that the size of the tables of a DP algorithm depends on how many ways a partial solution can intersect **mid**(*e*).
- In how many ways can we draw polygons inside a circle such that they touch the circle only on its k vertices and they do not intersect?

$$CN(k) = \frac{1}{k+1} \binom{2k}{k} \sim \frac{4^k}{\sqrt{\pi}k^{3/2}} \approx 4^k.$$

Key idea for planar graphs [Dorn et al. ESA'05]:

- Sphere cut decomposition: Branch decomposition where the vertices in each mid(e) are situated around a noose.
 [Seymour and Thomas. Combinatorica'94]
- Recall that the size of the tables of a DP algorithm depends on how many ways a partial solution can intersect **mid**(*e*).
- In how many ways can we draw polygons inside a circle such that they touch the circle only on its k vertices and they do not intersect?

$$CN(k) = \frac{1}{k+1} \binom{2k}{k} \sim \frac{4^k}{\sqrt{\pi}k^{3/2}} \approx 4^k.$$

Key idea for planar graphs [Dorn et al. ESA'05]:

- Sphere cut decomposition: Branch decomposition where the vertices in each mid(e) are situated around a noose.
 [Seymour and Thomas. Combinatorica'94]
- Recall that the size of the tables of a DP algorithm depends on how many ways a partial solution can intersect **mid**(*e*).
- In how many ways can we draw polygons inside a circle such that they touch the circle only on its k vertices and they do not intersect?



$$CN(k) = \frac{1}{k+1} \binom{2k}{k} \sim \frac{4^k}{\sqrt{\pi} k^{3/2}} \approx 4^k.$$

Key idea for planar graphs [Dorn et al. ESA'05]:

- Sphere cut decomposition: Branch decomposition where the vertices in each mid(e) are situated around a noose.
 [Seymour and Thomas. Combinatorica'94]
- Recall that the size of the tables of a DP algorithm depends on how many ways a partial solution can intersect **mid**(*e*).
- In how many ways can we draw polygons inside a circle such that they touch the circle only on its k vertices and they do not intersect?



$$CN(k) = \frac{1}{k+1} \binom{2k}{k} \sim \frac{4^k}{\sqrt{\pi}k^{3/2}} \approx 4^k$$

- Perform a planarization of the input graph by splitting the potential solutions into a number of pieces depending on the surface.
- Then, apply the sphere cut decomposition technique to a more complicated version of the problem where the number of pairings is still bounded by some Catalan number.
- Drawbacks of this technique:
 - ★ It depends on each **particular** problem.
 - ★ Cannot (a priori) be applied to the class of connected packing-encodable problems.

イロン イボン イモン イモン 三日

- Perform a planarization of the input graph by splitting the potential solutions into a number of pieces depending on the surface.
- Then, apply the sphere cut decomposition technique to a more complicated version of the problem where the number of pairings is still bounded by some Catalan number.
- Drawbacks of this technique:
 - ★ It depends on each **particular** problem.
 - ★ Cannot (a priori) be applied to the class of connected packing-encodable problems.

イロン イボン イモン イモン 一日

- Perform a planarization of the input graph by splitting the potential solutions into a number of pieces depending on the surface.
- Then, apply the sphere cut decomposition technique to a more complicated version of the problem where the number of pairings is still bounded by some Catalan number.
- Drawbacks of this technique:
 - ★ It depends on each **particular** problem.
 - Cannot (a priori) be applied to the class of connected packing-encodable problems.

(□) (四) (E) (E) (E)

- Perform a planarization of the input graph by splitting the potential solutions into a number of pieces depending on the surface.
- Then, apply the sphere cut decomposition technique to a more complicated version of the problem where the number of pairings is still bounded by some Catalan number.
- Drawbacks of this technique:
 - ★ It depends on each **particular** problem.
 - ★ Cannot (a priori) be applied to the class of connected packing-encodable problems.

Our approach is based on a new type of branch decomposition, called surface cut decomposition.

- Surface cut decompositions for graphs on surfaces generalize sphere cut decompositions for planar graphs.
 [Seymour and Thomas. *Combinatorica'94*]
- That is, we exploit directly the combinatorial structure of the potential solutions in the surface (**without planarization**).
- Using surface cut decompositions, we provide in a unified way single-exponential algorithms for connected packing-encodable problems, and with better genus dependence.

Our approach is based on a new type of branch decomposition, called surface cut decomposition.

- Surface cut decompositions for graphs on surfaces generalize sphere cut decompositions for planar graphs. [Seymour and Thomas. *Combinatorica'94*]
- That is, we exploit directly the combinatorial structure of the potential solutions in the surface (**without planarization**).
- Using surface cut decompositions, we provide in a unified way single-exponential algorithms for connected packing-encodable problems, and with better genus dependence.

Our approach is based on a new type of branch decomposition, called surface cut decomposition.

- Surface cut decompositions for graphs on surfaces generalize sphere cut decompositions for planar graphs.
 [Seymour and Thomas. Combinatorica'94]
- That is, we exploit directly the combinatorial structure of the potential solutions in the surface (without planarization).
- Using surface cut decompositions, we provide in a unified way single-exponential algorithms for connected packing-encodable problems, and with better genus dependence.

A surface cut decomposition of *G* is a branch decomposition (T, μ) of *G* and a subset $A \subseteq V(G)$, with |A| = O(g), s.t. for all $e \in E(T)$

• either $|\mathbf{mid}(e) \setminus A| \leq 2$,

or

- \star the vertices in **mid**(*e*) \ *A* are contained in a set \mathcal{N} of $\mathcal{O}(\mathbf{g})$ nooses;
- \star these nooses intersect in $\mathcal{O}(\mathbf{g})$ vertices;
- \leftarrow Σ \ $\bigcup_{N \in N}$ N contains exactly two connected components.

A surface cut decomposition of *G* is a branch decomposition (T, μ) of *G* and a subset $A \subseteq V(G)$, with |A| = O(g), s.t. for all $e \in E(T)$

either |mid(e) \ A| ≤ 2,

or

- \star the vertices in **mid**(*e*) \ *A* are contained in a set \mathcal{N} of $\mathcal{O}(g)$ nooses;
- \star these nooses intersect in $\mathcal{O}(\mathbf{g})$ vertices;
- \star Σ \ $\bigcup_{N \in N}$ *N* contains exactly two connected components.

A surface cut decomposition of *G* is a branch decomposition (T, μ) of *G* and a subset $A \subseteq V(G)$, with $|A| = O(\mathbf{g})$, s.t. for all $e \in E(T)$

- either |mid(e) \ A| ≤ 2,
- or
 - * the vertices in $mid(e) \setminus A$ are contained in a set \mathcal{N} of $\mathcal{O}(g)$ nooses;
 - \star these nooses intersect in $\mathcal{O}(\mathbf{g})$ vertices;
 - ★ Σ \ $\bigcup_{N \in N}$ *N* contains exactly two connected components.

A surface cut decomposition of *G* is a branch decomposition (T, μ) of *G* and a subset $A \subseteq V(G)$, with $|A| = O(\mathbf{g})$, s.t. for all $e \in E(T)$

• either $|\mathbf{mid}(e) \setminus A| \leq 2$,

or

- * the vertices in $\operatorname{mid}(e) \setminus A$ are contained in a set \mathcal{N} of $\mathcal{O}(g)$ nooses;
- ★ these nooses intersect in O(g) vertices;
- \star Σ \ $\bigcup_{N \in \mathcal{N}}$ *N* contains exactly two connected components.

A surface cut decomposition of *G* is a branch decomposition (T, μ) of *G* and a subset $A \subseteq V(G)$, with $|A| = O(\mathbf{g})$, s.t. for all $e \in E(T)$

either |mid(e) \ A| ≤ 2,

or

- * the vertices in $\operatorname{mid}(e) \setminus A$ are contained in a set \mathcal{N} of $\mathcal{O}(g)$ nooses;
- * these nooses intersect in $\mathcal{O}(\mathbf{g})$ vertices;
- * $\Sigma \setminus \bigcup_{N \in \mathcal{N}} N$ contains exactly two connected components.

Theorem (Rué, Thilikos, and S.)

Given a G on n vertices embedded in a surface of Euler genus **g**, with **bw**(G) $\leq k$, one can construct in $2^{3k+\mathcal{O}(\log k)} \cdot n^3$ time a surface cut decomposition (T, μ) of G of width at most $27k + \mathcal{O}(\mathbf{g})$.

- Partition G into polyhedral pieces, plus a set of A vertices, with |A| = O(g).
- For each piece H, compute a branch decomposition, using Amir's algorithm.
- Transform this branch decomposition to a **carving** decomposition of the **medial** graph of *H*.
- Make the carving decomposition bond, using Seymour and Thomas' algorithm.
- Transform it to a bond branch decomposition of H.
- Construct a branch decomposition of G by merging the branch decompositions of all the pieces.

Theorem (Rué, Thilikos, and S.)

Given a G on n vertices embedded in a surface of Euler genus **g**, with **bw**(G) $\leq k$, one can construct in $2^{3k+\mathcal{O}(\log k)} \cdot n^3$ time a surface cut decomposition (T, μ) of G of width at most $27k + \mathcal{O}(\mathbf{g})$.

- Partition G into **polyhedral** pieces, plus a set of A vertices, with |A| = O(g).
- For each piece H, compute a branch decomposition, using Amir's algorithm.
- Transform this branch decomposition to a **carving** decomposition of the **medial** graph of *H*.
- Make the carving decomposition bond, using Seymour and Thomas' algorithm.
- Transform it to a bond branch decomposition of *H*.
- Construct a branch decomposition of G by merging the branch decompositions of all the pieces.

Theorem (Rué, Thilikos, and S.)

Given a G on n vertices embedded in a surface of Euler genus **g**, with **bw**(G) $\leq k$, one can construct in $2^{3k+\mathcal{O}(\log k)} \cdot n^3$ time a surface cut decomposition (T, μ) of G of width at most $27k + \mathcal{O}(\mathbf{g})$.

- Partition G into **polyhedral** pieces, plus a set of A vertices, with |A| = O(g).
- For each piece *H*, compute a branch decomposition, using Amir's algorithm.
- Transform this branch decomposition to a **carving** decomposition of the **medial** graph of *H*.
- Make the carving decomposition bond, using Seymour and Thomas' algorithm.
- Transform it to a bond branch decomposition of *H*.
- Construct a branch decomposition of G by merging the branch decompositions of all the pieces.

Theorem (Rué, Thilikos, and S.)

Given a G on n vertices embedded in a surface of Euler genus **g**, with **bw**(G) $\leq k$, one can construct in $2^{3k+\mathcal{O}(\log k)} \cdot n^3$ time a surface cut decomposition (T, μ) of G of width at most $27k + \mathcal{O}(\mathbf{g})$.

Sketch of the construction of surface cut decompositions:

- Partition G into **polyhedral** pieces, plus a set of A vertices, with |A| = O(g).
- For each piece *H*, compute a branch decomposition, using Amir's algorithm.
- Transform this branch decomposition to a **carving** decomposition of the **medial** graph of *H*.
- Make the carving decomposition **bond**, using Seymour and Thomas' algorithm.
- Transform it to a bond branch decomposition of *H*.
- Construct a branch decomposition of *G* by **merging** the branch decompositions of all the pieces.

イロン イボン イモン イモン 一日

Theorem (Rué, Thilikos, and S.)

Given a G on n vertices embedded in a surface of Euler genus **g**, with **bw**(G) $\leq k$, one can construct in $2^{3k+\mathcal{O}(\log k)} \cdot n^3$ time a surface cut decomposition (T, μ) of G of width at most $27k + \mathcal{O}(\mathbf{g})$.

- Partition G into **polyhedral** pieces, plus a set of A vertices, with |A| = O(g).
- For each piece *H*, compute a branch decomposition, using Amir's algorithm.
- Transform this branch decomposition to a **carving** decomposition of the **medial** graph of *H*.
- Make the carving decomposition **bond**, using Seymour and Thomas' algorithm.
- Transform it to a bond branch decomposition of *H*.
- Construct a branch decomposition of G by merging the branch decompositions of all the pieces.

The main result is that if DP is applied on surface cut decompositions, then the time dependence on branchwidth is single-exponential:

Theorem (Rué, Thilikos, and S.)

Given a connected packing-encodable problem P in a graph G embedded in a surface of Euler genus **g**, with $\mathbf{bw}(G) \leq k$, the size of the tables of a dynamic programming algorithm to solve P on a surface cut decomposition of G is bounded above by $2^{\mathcal{O}(\log \mathbf{g} \cdot \mathbf{k} + \log k \cdot \mathbf{g})}$.

- This fact is proved using **analytic combinatorics**, generalizing Catalan structures to arbitrary surfaces.
- Upper bound of [Dorn, Fomin, Thilikos. SWAT'06]: 2^{O(g·k+log k·g²)}.

The main result is that if DP is applied on surface cut decompositions, then the time dependence on branchwidth is single-exponential:

Theorem (Rué, Thilikos, and S.)

Given a connected packing-encodable problem P in a graph G embedded in a surface of Euler genus **g**, with $\mathbf{bw}(G) \leq k$, the size of the tables of a dynamic programming algorithm to solve P on a surface cut decomposition of G is bounded above by $2^{\mathcal{O}(\log \mathbf{g} \cdot \mathbf{k} + \log k \cdot \mathbf{g})}$.

- This fact is proved using **analytic combinatorics**, generalizing Catalan structures to arbitrary surfaces.
- Upper bound of [Dorn, Fomin, Thilikos. SWAT'06]: 2^O(g·k+log k·g²).

The main result is that if DP is applied on surface cut decompositions, then the time dependence on branchwidth is single-exponential:

Theorem (Rué, Thilikos, and S.)

Given a connected packing-encodable problem P in a graph G embedded in a surface of Euler genus **g**, with $\mathbf{bw}(G) \leq k$, the size of the tables of a dynamic programming algorithm to solve P on a surface cut decomposition of G is bounded above by $2^{\mathcal{O}(\log \mathbf{g} \cdot \mathbf{k} + \log k \cdot \mathbf{g})}$.

- This fact is proved using **analytic combinatorics**, generalizing Catalan structures to arbitrary surfaces.
- Upper bound of [Dorn, Fomin, Thilikos. *SWAT'06*]: 2^{O(g·k+log k·g²)}.

Sketch of the enumerative part: Bipartite subdivisions

- Subdivision of the surface in vertices, edges and **2-dimensional regions** (not necessary contractible).
- All vertices lay in the boundary.
- 2 types of 2-dimensional regions: black and white.
- Each vertex is incident with exactly 1 black region (also called *block*).
- Each border is rooted.



Fixing the number of vertices on a given surface, we have an infinite number of bipartite subdivisions.

Non-crossing partitions in higher genus surfaces

- Each bipartite subdivision induces a non-crossing partition on the set of vertices.
- **Problem:** Different bipartite subdivisions can define the same non-crossing partition.



• **Objective:** finding "good" bounds for the number of non-crossing partitions on a given surface.

Non-crossing partitions in higher genus surfaces

- Each bipartite subdivision induces a non-crossing partition on the set of vertices.
- **Problem:** Different bipartite subdivisions can define the same non-crossing partition.



• **Objective:** finding "good" bounds for the number of non-crossing partitions on a given surface.

After some study of bicolored trees and its asymptotics...

Theorem (Rué, Thilikos, S.)

Let Σ be a surface with boundary. Then the number of non-crossing partitions on Σ with k vertices is asymptotically bounded by

$$\frac{\mathcal{C}(\Sigma)}{\Gamma\left(-3/2\chi(\Sigma)+\beta(\Sigma)\right)}\cdot k^{-3/2\chi(\Sigma)+\beta(\Sigma)-1}\cdot 4^{k}$$

where

- $C(\Sigma)$ is a function depending only on Σ (cubic maps in $\overline{\Sigma}$ with $\beta(\Sigma)$ faces).
- $\chi(\Sigma)$ is the Euler characteristic ($\chi(\Sigma) = 2 eg(\Sigma)$).
- β(Σ) is the number of components of the boundary (it depends linearly on the branchwidth of the input graph).

In the case of the **disk** (Catalan numbers): $\frac{1}{\sqrt{\pi}} \cdot k^{-3/2} \cdot 4^k$.

イロン 不留 とくほど 不良とう ほ

After some study of bicolored trees and its asymptotics...

Theorem (Rué, Thilikos, S.)

Let Σ be a surface with boundary. Then the number of non-crossing partitions on Σ with k vertices is asymptotically bounded by

$$\frac{\mathcal{C}(\Sigma)}{\Gamma\left(-3/2\chi(\Sigma)+\beta(\Sigma)\right)}\cdot k^{-3/2\chi(\Sigma)+\beta(\Sigma)-1}\cdot 4^{k}$$

where

- $C(\Sigma)$ is a function depending only on Σ (cubic maps in $\overline{\Sigma}$ with $\beta(\Sigma)$ faces).
- χ(Σ) is the Euler characteristic (χ(Σ) = 2 eg(Σ)).
- β(Σ) is the number of components of the boundary (it depends linearly on the branchwidth of the input graph).

In the case of the **disk** (Catalan numbers): $\frac{1}{\sqrt{\pi}} \cdot k^{-3/2} \cdot 4^k$.

After some study of bicolored trees and its asymptotics...

Theorem (Rué, Thilikos, S.)

Let Σ be a surface with boundary. Then the number of non-crossing partitions on Σ with k vertices is asymptotically bounded by

$$\frac{\mathcal{C}(\Sigma)}{\Gamma\left(-3/2\chi(\Sigma)+\beta(\Sigma)\right)}\cdot k^{-3/2\chi(\Sigma)+\beta(\Sigma)-1}\cdot 4^{k}$$

where

- $C(\Sigma)$ is a function depending only on Σ (cubic maps in $\overline{\Sigma}$ with $\beta(\Sigma)$ faces).
- χ(Σ) is the Euler characteristic (χ(Σ) = 2 eg(Σ)).
- β(Σ) is the number of components of the boundary (it depends linearly on the branchwidth of the input graph).

In the case of the **disk** (Catalan numbers): $\frac{1}{\sqrt{\pi}} \cdot k^{-3/2} \cdot 4^k$.

(□) (@) (E) (E) E (0)

How to use this framework?

- We presented a framework for the design of DP algorithms on surface-embedded graphs running in time 2^{O(k)} ⋅ n.
- How to use this framework?
 - Let P be a connected packing-encodable problem on a surface-embedded graph G.
 - As a preprocessing step, build a surface cut decomposition of G, using the 1st Theorem.
 - Run a "clever" DP algorithm to solve P over the obtained surface cut decomposition.
 - The single-exponential running time of the algorithm is a consequence of the 2nd Theorem.

- We presented a framework for the design of DP algorithms on surface-embedded graphs running in time 2^{O(k)} ⋅ n.
- How to use this framework?
 - Let P be a connected packing-encodable problem on a surface-embedded graph G.
 - As a preprocessing step, build a surface cut decomposition of G, using the 1st Theorem.
 - 3 Run a "clever" DP algorithm to solve P over the obtained surface cut decomposition.
 - The single-exponential running time of the algorithm is a consequence of the 2nd Theorem.

イロン イボン イモン トモ

- We presented a framework for the design of DP algorithms on surface-embedded graphs running in time 2^{O(k)} ⋅ n.
- How to use this framework?
 - Let P be a connected packing-encodable problem on a surface-embedded graph G.
 - As a preprocessing step, build a surface cut decomposition of G, using the 1st Theorem.
 - 3 Run a "clever" DP algorithm to solve P over the obtained surface cut decomposition.
 - The single-exponential running time of the algorithm is a consequence of the 2nd Theorem.

イロン イボン イモン トモ

- We presented a framework for the design of DP algorithms on surface-embedded graphs running in time 2^{O(k)} ⋅ n.
- How to use this framework?
 - Let P be a connected packing-encodable problem on a surface-embedded graph G.
 - As a preprocessing step, build a surface cut decomposition of G, using the 1st Theorem.
 - Run a "clever" DP algorithm to solve P over the obtained surface cut decomposition.
 - The single-exponential running time of the algorithm is a consequence of the 2nd Theorem.

<ロ> <四> <四> <三> <三> <三> <三> <三> <三
- We presented a framework for the design of DP algorithms on surface-embedded graphs running in time 2^{O(k)} ⋅ n.
- How to use this framework?
 - Let P be a connected packing-encodable problem on a surface-embedded graph G.
 - As a preprocessing step, build a surface cut decomposition of G, using the 1st Theorem.
 - Run a "clever" DP algorithm to solve P over the obtained surface cut decomposition.
 - The single-exponential running time of the algorithm is a consequence of the 2nd Theorem.

- Can this framework be applied to more complicated problems? Fundamental problem: H-MINOR CONTAINMENT
 - * Minor containment for host graphs *G* on surfaces. [Adler, Dorn, Fomin, S., Thilikos. *SWAT'10*] With running time $2^{O(k)} \cdot h^{2k} \cdot 2^{O(h)} \cdot n$. $(h = |V(H)|, k = \mathbf{bw}(G), n = |V(G)|)$
 - * Single-exponential algorithm for planar host graphs. [Adler, Dorn, Fomin, S., Thilikos. *ESA'10*]

Truly single-exponential: $2^{\mathcal{O}(h)} \cdot n$.

Can it be generalized to host graphs on arbitrary surfaces?

Rooted graph problems: DISJOINT PATHS

- Can this framework be applied to more complicated problems? Fundamental problem: H-MINOR CONTAINMENT
 - * Minor containment for host graphs *G* on surfaces. [Adler, Dorn, Fomin, S., Thilikos. *SWAT'10*] With running time $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$. $(h = |V(H)|, k = \mathbf{bw}(G), n = |V(G)|)$
 - * Single-exponential algorithm for planar host graphs. [Adler, Dorn, Fomin, S., Thilikos. *ESA'10*]

Truly single-exponential: $2^{\mathcal{O}(h)} \cdot n$.

Can it be generalized to host graphs on arbitrary surfaces?

Rooted graph problems: DISJOINT PATHS

- Can this framework be applied to more complicated problems? Fundamental problem: H-MINOR CONTAINMENT
 - * Minor containment for host graphs *G* on surfaces. [Adler, Dorn, Fomin, S., Thilikos. *SWAT'10*] With running time $2^{O(k)} \cdot h^{2k} \cdot 2^{O(h)} \cdot n$. $(h = |V(H)|, k = \mathbf{bw}(G), n = |V(G)|)$
 - * Single-exponential algorithm for planar host graphs. [Adler, Dorn, Fomin, S., Thilikos. *ESA'10*]

Truly single-exponential: $2^{\mathcal{O}(h)} \cdot n$.

Can it be generalized to host graphs on arbitrary surfaces?

Rooted graph problems: DISJOINT PATHS

- Can this framework be applied to more complicated problems? Fundamental problem: H-MINOR CONTAINMENT
 - * Minor containment for host graphs *G* on surfaces. [Adler, Dorn, Fomin, S., Thilikos. *SWAT'10*] With running time $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$. $(h = |V(H)|, k = \mathbf{bw}(G), n = |V(G)|)$
 - * Single-exponential algorithm for planar host graphs. [Adler, Dorn, Fomin, S., Thilikos. *ESA'10*]

Truly single-exponential: $2^{\mathcal{O}(h)} \cdot n$.

Can it be generalized to host graphs on arbitrary surfaces?

Rooted graph problems: DISJOINT PATHS

◆□ → ◆□ → ◆豆 → ◆豆 → □ 亘

- Can this framework be applied to more complicated problems? Fundamental problem: H-MINOR CONTAINMENT
 - * Minor containment for host graphs *G* on surfaces. [Adler, Dorn, Fomin, S., Thilikos. *SWAT'10*] With running time $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$. $(h = |V(H)|, k = \mathbf{bw}(G), n = |V(G)|)$
 - * Single-exponential algorithm for planar host graphs. [Adler, Dorn, Fomin, S., Thilikos. *ESA'10*]

Truly single-exponential: $2^{\mathcal{O}(h)} \cdot n$.

Can it be generalized to host graphs on arbitrary surfaces?

Rooted graph problems: **DISJOINT PATHS**

2 Can this framework be extended to more general graphs? Ongoing work: minor-free graphs...

- Can this framework be applied to more complicated problems? Fundamental problem: H-MINOR CONTAINMENT
 - * Minor containment for host graphs *G* on surfaces. [Adler, Dorn, Fomin, S., Thilikos. *SWAT'10*] With running time $2^{O(k)} \cdot h^{2k} \cdot 2^{O(h)} \cdot n$. $(h = |V(H)|, k = \mathbf{bw}(G), n = |V(G)|)$
 - * Single-exponential algorithm for planar host graphs. [Adler, Dorn, Fomin, S., Thilikos. *ESA'10*]

Truly single-exponential: $2^{\mathcal{O}(h)} \cdot n$.

Can it be generalized to host graphs on arbitrary surfaces?

Rooted graph problems: **DISJOINT PATHS**

Can this framework be extended to more general graphs?
Ongoing work: minor-free graphs...

Next subsubsection is...

Good news

Parameterizing by branchwidth

- Graphs on surfaces
- Main ideas of our approach
- Extensions
- Parameterizing by the size of the solution



I planned to say something about it on the board, but finally I decided to skip it, sorry!

- Idea: use the structure of minor-free graphs.
- The apices are not a real problem, but the vortices are a different story...
- We can capture their behavior with the help of *h*-triangulations.
- Maaaaaany details missing, hopefully written down soon.

Good news

- Parameterizing by branchwidth
 - Graphs on surfaces
 - Main ideas of our approach
 - Extensions
- Parameterizing by the size of the solution



Parameterizing by the size of the solution

- Another natural parameter: size of the solution.
- Example: *k*-VERTEX COVER.
- Let Θ_c be the graph with two vertices and $c \ge 1$ parallel edges.
- [Joret, Paul, S., Saurabh, Thomassé. 2011]:

The $k - \Theta_c$ -DELETION problem

Instance: A graph G = (V, E) and a non-negative integer *k*. **Parameter**: *k*.

Question: Does there exist $S \subseteq V$, $|S| \le k$, such that $G[V \setminus S]$ does not contain Θ_c as a minor?

Parameterizing by the size of the solution

- Another natural parameter: size of the solution.
- Example: *k*-VERTEX COVER.
- Let Θ_c be the graph with two vertices and $c \ge 1$ parallel edges.
- [Joret, Paul, S., Saurabh, Thomassé. 2011]:

The $k - \Theta_c$ -DELETION problem

Instance: A graph G = (V, E) and a non-negative integer *k*. **Parameter**: *k*.

Question: Does there exist $S \subseteq V$, $|S| \le k$, such that $G[V \setminus S]$ does not contain Θ_c as a minor?

Parameterizing by the size of the solution

- Another natural parameter: size of the solution.
- Example: *k*-VERTEX COVER.
- Let Θ_c be the graph with two vertices and $c \ge 1$ parallel edges.
- [Joret, Paul, S., Saurabh, Thomassé. 2011]:

The $k - \Theta_c$ -DELETION problem

Instance: A graph G = (V, E) and a non-negative integer *k*. **Parameter**: *k*.

Question: Does there exist $S \subseteq V$, $|S| \le k$, such that $G[V \setminus S]$ does not contain Θ_c as a minor?

- For c = 1: k-VERTEX COVER. Well-known algorithms in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. [Balasubramanian, Fellows, Raman. *IPL'98*]
- For c = 2: k-FEEDBACK VERTEX SET.
 Algorithms in time 2^{O(k)} n^{O(1)} were more difficult to obtain:
 [Guo, Gramm, Hüffner, Niedermeier, Wernicke. J. Comp. Syst. Sci.'06]
 [Dehne, Fellows, Langston, Rosamond, Stevens. Th. Comp. Syst.'06]
- For c = 3: k-Diamond Hitting Set.

• For *c* = 1: *k*-VERTEX COVER.

Well-known algorithms in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. [Balasubramanian, Fellows, Raman. *IPL'98*]

For c = 2: k-FEEDBACK VERTEX SET.
 Algorithms in time 2^{O(k)} · n^{O(1)} were more difficult to obtain:
 [Guo, Gramm, Hüffner, Niedermeier, Wernicke. J. Comp. Syst. Sci.'06]
 [Dehne, Fellows, Langston, Rosamond, Stevens. Th. Comp. Syst.'06]

• For *c* = 3: *k*-DIAMOND HITTING SET.

- For c = 1: k-VERTEX COVER. Well-known algorithms in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. [Balasubramanian, Fellows, Raman. *IPL'98*]
- For c = 2: *k*-FEEDBACK VERTEX SET. Algorithms in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ were more difficult to obtain: [Guo, Gramm, Hüffner, Niedermeier, Wernicke. *J. Comp. Syst. Sci.*'06] [Dehne, Fellows, Langston, Rosamond, Stevens. *Th. Comp. Syst.*'06]
- For *c* = 3: *k*-DIAMOND HITTING SET.

- For c = 1: k-VERTEX COVER. Well-known algorithms in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. [Balasubramanian, Fellows, Raman. *IPL'98*]
- For c = 2: k-FEEDBACK VERTEX SET.
 Algorithms in time 2^{O(k)} · n^{O(1)} were more difficult to obtain:
 [Guo, Gramm, Hüffner, Niedermeier, Wernicke. J. Comp. Syst. Sci.'06]
 [Dehne, Fellows, Langston, Rosamond, Stevens. Th. Comp. Syst.'06]
- For *c* = 3: *k*-DIAMOND HITTING SET.

- For c = 1: k-VERTEX COVER. Well-known algorithms in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. [Balasubramanian, Fellows, Raman. *IPL'98*]
- For c = 2: *k*-FEEDBACK VERTEX SET. Algorithms in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ were more difficult to obtain: [Guo, Gramm, Hüffner, Niedermeier, Wernicke. *J. Comp. Syst. Sci.*'06] [Dehne, Fellows, Langston, Rosamond, Stevens. *Th. Comp. Syst.*'06]
- For *c* = 3: *k*-DIAMOND HITTING SET.

- For c = 1: k-VERTEX COVER. Well-known algorithms in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. [Balasubramanian, Fellows, Raman. *IPL'98*]
- For c = 2: *k*-FEEDBACK VERTEX SET. Algorithms in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ were more difficult to obtain: [Guo, Gramm, Hüffner, Niedermeier, Wernicke. *J. Comp. Syst. Sci.*'06] [Dehne, Fellows, Langston, Rosamond, Stevens. *Th. Comp. Syst.*'06]
- For c = 3: k-DIAMOND HITTING SET.

- For c = 1: k-VERTEX COVER. Well-known algorithms in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. [Balasubramanian, Fellows, Raman. *IPL'98*]
- For c = 2: *k*-FEEDBACK VERTEX SET. Algorithms in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ were more difficult to obtain: [Guo, Gramm, Hüffner, Niedermeier, Wernicke. *J. Comp. Syst. Sci.*'06] [Dehne, Fellows, Langston, Rosamond, Stevens. *Th. Comp. Syst.*'06]
- For c = 3: k-DIAMOND HITTING SET.

[Fomin, Lokshtanov, Misra, Philip, Saurabh. STACS'11]:

Let \mathcal{F} be a finite list of graphs.

The *k*-*F*-DELETION problem

Instance: A graph G = (V, E) and a non-negative integer k. **Parameter**: k.

Jestion: Does there exist $S \subseteq V$, $|S| \leq k$, such that $G[V \setminus S]$ does not contain any graph in \mathcal{F} as a minor?

If $\Theta_c \in \mathcal{F}$ for some $c \ge 1$, then k- \mathcal{F} -DELETION admits a kernel of size $\mathcal{O}(k^2 \log^{3/2} k)$.

[Fomin, Lokshtanov, Misra, Philip, Saurabh. STACS'11]:

Let \mathcal{F} be a finite list of graphs.

The *k*-*F*-DELETION problem

Instance: A graph G = (V, E) and a non-negative integer *k*. **Parameter**: *k*.

Question: Does there exist $S \subseteq V$, $|S| \leq k$, such that $G[V \setminus S]$ does not contain any graph in \mathcal{F} as a minor?

If $\Theta_c \in \mathcal{F}$ for some $c \ge 1$, then k- \mathcal{F} -DELETION admits a kernel of size $\mathcal{O}(k^2 \log^{3/2} k)$.

[Fomin, Lokshtanov, Misra, Philip, Saurabh. STACS'11]:

Let \mathcal{F} be a finite list of graphs.

The *k*-*F*-DELETION problem

Instance: A graph G = (V, E) and a non-negative integer *k*. **Parameter**: *k*.

Question: Does there exist $S \subseteq V$, $|S| \leq k$, such that $G[V \setminus S]$ does not contain any graph in \mathcal{F} as a minor?

If $\Theta_c \in \mathcal{F}$ for some $c \ge 1$, then k- \mathcal{F} -DELETION admits a kernel of size $\mathcal{O}(k^2 \log^{3/2} k)$.

- If Θ_c ∈ F for some c ≥ 1, then k-F-DELETION admits a kernel of size O(k² log^{3/2} k).
- This kernel yiels a simple FPT algorithm with running time $\binom{k^2 \log^{3/2} k}{k} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}.$
- Using iterative compression, we obtain an algorithm for $k \Theta_c$ -DELETION problem in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.

OPEN QUESTIONS:

- Can we do it faster in sparse graphs?
- Deleting at most k vertices so that the resulting graph has tw < c?</p>

- If Θ_c ∈ F for some c ≥ 1, then k-F-DELETION admits a kernel of size O(k² log^{3/2} k).
- This kernel yiels a simple FPT algorithm with running time $\binom{k^2 \log^{3/2} k}{k} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}.$
- Using iterative compression, we obtain an algorithm for $k \cdot \Theta_c$ -DELETION problem in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.

• OPEN QUESTIONS:

- Can we do it faster in sparse graphs?
- Deleting at most k vertices so that the resulting graph has tw ≤ c?

- If Θ_c ∈ F for some c ≥ 1, then k-F-DELETION admits a kernel of size O(k² log^{3/2} k).
- This kernel yiels a simple FPT algorithm with running time $\binom{k^2 \log^{3/2} k}{k} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}.$
- Using iterative compression, we obtain an algorithm for $k \cdot \Theta_c$ -DELETION problem in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.
- OPEN QUESTIONS:
 - Can we do it faster in sparse graphs?
 - Deleting at most k vertices so that the resulting graph has $tw \le c$?

- If Θ_c ∈ F for some c ≥ 1, then k-F-DELETION admits a kernel of size O(k² log^{3/2} k).
- This kernel yiels a simple FPT algorithm with running time $\binom{k^2 \log^{3/2} k}{k} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}.$
- Using iterative compression, we obtain an algorithm for $k \cdot \Theta_c$ -DELETION problem in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.
- OPEN QUESTIONS:
 - Can we do it faster in sparse graphs?
 - Deleting at most *k* vertices so that the resulting graph has **tw** $\leq c$?

Good news

- Parameterizing by branchwidth
 - Graphs on surfaces
 - Main ideas of our approach
 - Extensions
- Parameterizing by the size of the solution



For an FPT problem, is it always possible to obtain algorithms with running time $c^k \cdot n^{O(1)}$?

[Lokshtanov, Marx, Saurabh. SODA'11]:

Assuming that 3SAT cannot be solved in $2^{o(n)}$ time (ETH), then:

- DISJOINT PATHS cannot be solved in $2^{o(\mathsf{tw} \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ time.
- *d*-DISTORTION cannot be solved in $2^{o(d \log d)} \cdot n^{O(1)}$ time.

Here, $\mathbf{tw} = \mathbf{tw}(G)$ and n = |V(G)|. These bounds are tight.

OPEN QUESTIONS:

- Parameterizing by treewidth: HAMILTONIAN PATH, FVS, CONNECTED VERTEX COVER, CONNECTED DOMINATING SET... Is $2^{O(\text{tw}\log \text{tw})} \cdot n^{O(1)}$ time optimal?
- Parameterizing by solution size: DIRECTED FVS, INTERVAL COMPLETION... Is 2^{O(k log k)} · n^{O(1)} time optimal?

(日) (四) (日) (日) (日)

For an FPT problem, is it always possible to obtain algorithms with running time $c^k \cdot n^{O(1)}$?

[Lokshtanov, Marx, Saurabh. *SODA'11*]: Assuming that 3SAT cannot be solved in 2^{*o*(*n*)} time (ETH), then:

DISJOINT PATHS cannot be solved in 2^{o(tw log tw)} · n^{O(1)} time.
 d-DISTORTION cannot be solved in 2^{o(d log d)} · n^{O(1)} time.

Here, $\mathbf{tw} = \mathbf{tw}(G)$ and n = |V(G)|. These bounds are tight.

OPEN QUESTIONS:

- Parameterizing by treewidth: HAMILTONIAN PATH, FVS, CONNECTED VERTEX COVER, CONNECTED DOMINATING SET... Is $2^{\mathcal{O}(\text{tw} \log \text{tw})} \cdot n^{\mathcal{O}(1)}$ time optimal?
- Parameterizing by solution size: DIRECTED FVS, INTERVAL COMPLETION... Is $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ time optimal?

・ロン ・四 と ・ ヨン ・ ヨ

For an FPT problem, is it always possible to obtain algorithms with running time $c^k \cdot n^{O(1)}$?

[Lokshtanov, Marx, Saurabh. SODA'11]:

Assuming that 3SAT cannot be solved in $2^{o(n)}$ time (ETH), then:

- DISJOINT PATHS cannot be solved in $2^{o(\mathsf{tw} \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ time.
- *d*-DISTORTION cannot be solved in $2^{o(d \log d)} \cdot n^{O(1)}$ time.

Here, $\mathbf{tw} = \mathbf{tw}(G)$ and n = |V(G)|. These bounds are tight.

OPEN QUESTIONS:

- Parameterizing by treewidth: HAMILTONIAN PATH, FVS, CONNECTED VERTEX COVER, CONNECTED DOMINATING SET... Is $2^{\mathcal{O}(\text{tw} \log \text{tw})} \cdot n^{\mathcal{O}(1)}$ time optimal?
- Parameterizing by solution size: DIRECTED FVS, INTERVAL COMPLETION... Is $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ time optimal?

・ コート 人口 トート モート・ トレート

For an FPT problem, is it always possible to obtain algorithms with running time $c^k \cdot n^{O(1)}$?

[Lokshtanov, Marx, Saurabh. SODA'11]:

Assuming that 3SAT cannot be solved in $2^{o(n)}$ time (ETH), then:

- DISJOINT PATHS cannot be solved in $2^{o(\mathsf{tw} \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ time.
- *d*-DISTORTION cannot be solved in $2^{o(d \log d)} \cdot n^{O(1)}$ time.

Here, $\mathbf{tw} = \mathbf{tw}(G)$ and n = |V(G)|. These bounds are tight.

OPEN QUESTIONS:

- Parameterizing by treewidth: HAMILTONIAN PATH, FVS, CONNECTED VERTEX COVER, CONNECTED DOMINATING SET... Is $2^{O(\text{tw} \log \text{tw})} \cdot n^{O(1)}$ time optimal?
- Parameterizing by solution size: DIRECTED FVS, INTERVAL COMPLETION... Is $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ time optimal?

For an FPT problem, is it always possible to obtain algorithms with running time $c^k \cdot n^{O(1)}$?

[Lokshtanov, Marx, Saurabh. SODA'11]:

Assuming that 3SAT cannot be solved in $2^{o(n)}$ time (ETH), then:

- DISJOINT PATHS cannot be solved in $2^{o(\mathsf{tw} \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ time.
- *d*-DISTORTION cannot be solved in $2^{o(d \log d)} \cdot n^{O(1)}$ time.

Here, $\mathbf{tw} = \mathbf{tw}(G)$ and n = |V(G)|. These bounds are tight.

OPEN QUESTIONS:

- Parameterizing by treewidth: HAMILTONIAN PATH, FVS, CONNECTED VERTEX COVER, CONNECTED DOMINATING SET... Is $2^{\mathcal{O}(\text{tw} \log \text{tw})} \cdot n^{\mathcal{O}(1)}$ time optimal?
- Parameterizing by solution size: DIRECTED FVS, INTERVAL COMPLETION... Is $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ time optimal?

For problems that can be solved in $c^k \cdot n^{\mathcal{O}(1)}$ for some constant c > 1, which is the best c?

[Lokshtanov, Marx, Saurabh. SODA'11]:

Assuming that SAT cannot be solved in $O(2 - \varepsilon)^n$ time (SETH), then for any $\varepsilon > 0$:

- INDEPENDENT SET cannot be solved in $(2 \varepsilon)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- DOMINATING SET cannot be solved in $(3 \varepsilon)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- MAX CUT cannot be solved in $(2 \varepsilon)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- For any $q \ge 3$, q-COLORING cannot be solved in $(q \varepsilon)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$.

Here, $\mathbf{tw} = \mathbf{tw}(G)$ and n = |V(G)|. These bounds are tight.

OPEN QUESTIONS:

- Can one prove similar bounds when *G* is planar? Or maybe better algorithms?
- Lower bounds for other parameters?

For problems that can be solved in $c^k \cdot n^{\mathcal{O}(1)}$ for some constant c > 1, which is the best c?

[Lokshtanov, Marx, Saurabh. SODA'11]:

Assuming that SAT cannot be solved in $O(2 - \varepsilon)^n$ time (SETH), then for any $\varepsilon > 0$:

- INDEPENDENT SET cannot be solved in $(2 \varepsilon)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- DOMINATING SET cannot be solved in $(3 \varepsilon)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- MAX CUT cannot be solved in $(2 \varepsilon)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- For any $q \ge 3$, q-COLORING cannot be solved in $(q \varepsilon)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$.

Here, $\mathbf{tw} = \mathbf{tw}(G)$ and n = |V(G)|. These bounds are tight.

OPEN QUESTIONS:

- Can one prove similar bounds when *G* is planar? Or maybe better algorithms?
- Lower bounds for other parameters?

イロン イロン イヨン イヨン 三日

For problems that can be solved in $c^k \cdot n^{\mathcal{O}(1)}$ for some constant c > 1, which is the best c?

[Lokshtanov, Marx, Saurabh. SODA'11]:

Assuming that SAT cannot be solved in $O(2 - \varepsilon)^n$ time (SETH), then for any $\varepsilon > 0$:

- INDEPENDENT SET cannot be solved in $(2 \varepsilon)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- DOMINATING SET cannot be solved in $(3 \varepsilon)^{\mathsf{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- MAX CUT cannot be solved in $(2 \varepsilon)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- For any $q \ge 3$, q-COLORING cannot be solved in $(q \varepsilon)^{\mathsf{tw}} \cdot n^{\mathcal{O}(1)}$.

Here, $\mathbf{tw} = \mathbf{tw}(G)$ and n = |V(G)|. These bounds are tight.

OPEN QUESTIONS:

- Can one prove similar bounds when *G* is planar? Or maybe better algorithms?
- Lower bounds for other parameters?
Lower bounds (II)

For problems that can be solved in $c^k \cdot n^{\mathcal{O}(1)}$ for some constant c > 1, which is the best c?

[Lokshtanov, Marx, Saurabh. SODA'11]:

Assuming that SAT cannot be solved in $O(2 - \varepsilon)^n$ time (SETH), then for any $\varepsilon > 0$:

- INDEPENDENT SET cannot be solved in $(2 \varepsilon)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- DOMINATING SET cannot be solved in $(3 \varepsilon)^{\mathsf{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- MAX CUT cannot be solved in $(2 \varepsilon)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- For any $q \ge 3$, q-COLORING cannot be solved in $(q \varepsilon)^{\mathsf{tw}} \cdot n^{\mathcal{O}(1)}$.

Here, $\mathbf{tw} = \mathbf{tw}(G)$ and n = |V(G)|. These bounds are tight.

OPEN QUESTIONS:

- Can one prove similar bounds when *G* is planar? Or maybe better algorithms?
- Lower bounds for other parameters?

(ロ) (部) (目) (日) (日) (の)

Gràcies!