Dynamic programming in sparse graphs

Ignasi Sau

CNRS, LIRMM, Montpellier, France

Seminari de Teoria de Grafs, Combinatòria i Applicacions

Joint work with:

Juanjo Rué
Instituto de Ciencias Matemáticas, Madrid, Spain

Dimitrios M. Thilikos
Department of Mathematics, NKU of Athens, Greece

Outline

1. Motivation
2. Graphs on surfaces
   - Preliminaries
   - Main ideas of our approach
3. Extension to minor-free graphs
4. Conclusions
   - Further research
   - Some recent results

Next section is...

Some words on parameterized complexity

- Idea: given an NP-hard problem, fix one parameter of the input to see if the problem gets more "tractable".

  Example: the size of a VERTEX COVER.

- Given a (NP-hard) problem with input of size n and a parameter k, a fixed-parameter tractable (FPT) algorithm runs in

  $$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

  Examples: k-VERTEX COVER, k-LONGEST PATH.

Some words on parameterized complexity

- Idea: given an NP-hard problem, fix one parameter of the input to see if the problem gets more "tractable".

  Example: the size of a VERTEX COVER.

- Given a (NP-hard) problem with input of size n and a parameter k, a fixed-parameter tractable (FPT) algorithm runs in

  $$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

  Examples: k-VERTEX COVER, k-LONGEST PATH.

FPT and single-exponential algorithms

- Courcelle's theorem (1988):

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

- **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.

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

> Objective:
> build a framework to obtain **single-exponential algorithms** for a class of NP-hard problems in **sparse graphs**.

FPT and single-exponential algorithms

- Courcelle's theorem (1988):

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

- **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.

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

> Objective:
> build a framework to obtain **single-exponential algorithms** for a class of NP-hard problems in **sparse graphs**.

FPT and single-exponential algorithms

- Courcelle's theorem (1988):

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

- **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.

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

Objective:
build a framework to obtain **single-exponential algorithms** for a class of NP-hard problems in **sparse graphs**.

FPT and single-exponential algorithms

- Courcelle's theorem (1988):

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

- **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.

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

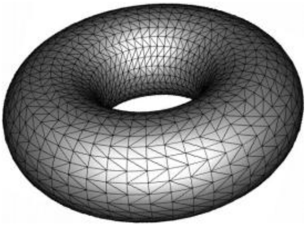
Objective:
build a framework to obtain **single-exponential algorithms** for a class of NP-hard problems in **sparse graphs**.

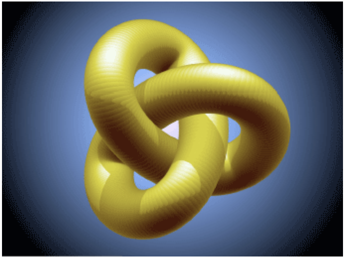
Next section is...

Next subsection is...

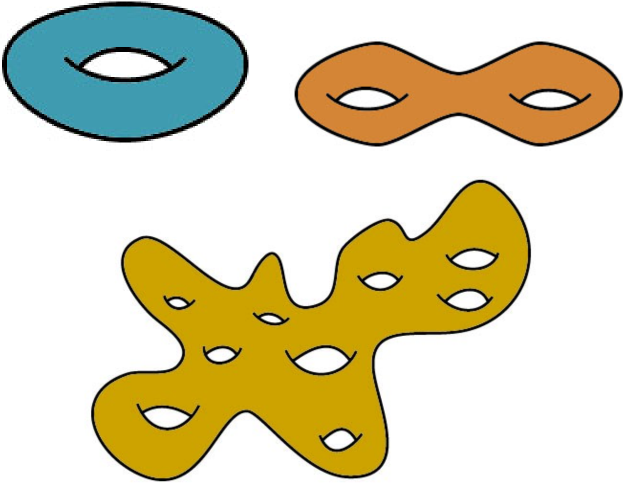
Surfaces

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

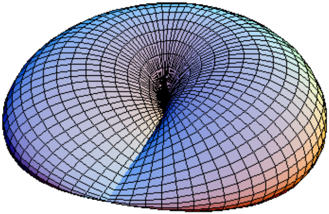


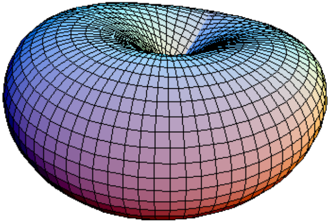


Handles



Cross-caps





Surface Classification Theorem

- **Surface Classification Theorem**:

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

- Orientable surfaces:

  obtained by adding $g \geq 0$ *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

- **Surface Classification Theorem**:

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

- Orientable surfaces:

  obtained by adding $g \geq 0$ *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 $\mathbb{S}^2$ by adding handles and cross-caps.

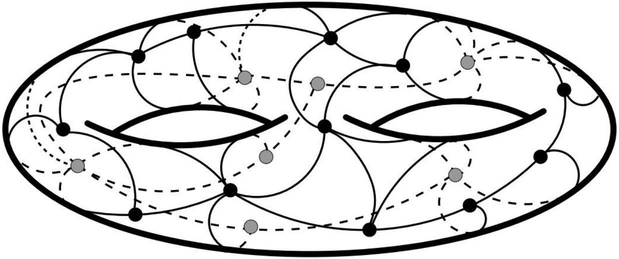
- Orientable surfaces:

  obtained by adding $g \geq 0$ *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 $\mathbf{eg}(\mathbb{P}_h) = h$.

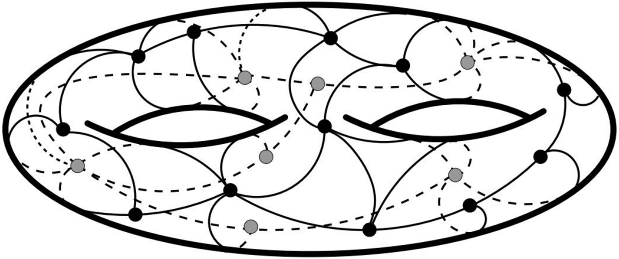
EMBEDDED GRAPH: GRAPH DRAWN ON A SURFACE, NO CROSSINGS



- The Euler genus of a graph G, $\mathbf{eg}(G)$, is the least Euler genus of the surfaces in which G can be embedded.

Graphs on surfaces

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.

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 $\mathbf{mid}(e) = V(A_e) \cap V(B_e)$.
- The width of a branch decomposition is $\max_{e \in E(T)} |\mathbf{mid}(e)|$.
- The branchwidth of a graph G (denoted $\mathbf{bw}(G)$) is the minimum width over all branch decompositions of G:

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

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 $\textbf{mid}(e) = V(A_e) \cap V(B_e)$.
- The width of a branch decomposition is $\max_{e \in E(T)} |\textbf{mid}(e)|$.
- The branchwidth of a graph G (denoted $\textbf{bw}(G)$) is the minimum width over all branch decompositions of G:

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

Dynamic programming (DP)

- 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 $\mathbf{mid}(e)$.
- 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.

Dynamic programming (DP)

- 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 $\mathbf{mid}(e)$.
- 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.

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

1. 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 $2^{\mathcal{O}(k)}$.

2. 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*].

3. *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 $2^{\Theta(k \log k)}$.

   None of the current techniques seemed to fit in this class of **connected packing-encodable** problems...

A classification of graph optimization problems

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

1. 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 $2^{\mathcal{O}(k)}$.

2. 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*].

3. *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 $2^{\Theta(k \log k)}$.

   None of the current techniques seemed to fit in this class of **connected packing-encodable** problems...

A classification of graph optimization problems

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

1. 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 $2^{\mathcal{O}(k)}$.

2. 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*].

3. *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 $2^{\Theta(k \log k)}$.

   None of the current techniques seemed to fit in this class of **connected packing-encodable** problems...

A classification of graph optimization problems

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

1. 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 $2^{\mathcal{O}(k)}$.

2. 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*].

3. *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 $2^{\Theta(k \log k)}$.

   None of the current techniques seemed to fit in this class of **connected packing-encodable** problems...

A classification of graph optimization problems

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

1. 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 $2^{\mathcal{O}(k)}$.

2. 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*].

3. *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 $2^{\Theta(k \log k)}$.

   None of the current techniques seemed to fit in this class of **connected packing-encodable** problems...

A classification of graph optimization problems

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

1. 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 $2^{\mathcal{O}(k)}$.

2. 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*].

3. *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 $2^{\Theta(k \log k)}$.

None of the current techniques seemed to fit in this class of **connected packing-encodable** problems...

A classification of graph optimization problems

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

1. 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 $2^{\mathcal{O}(k)}$.

2. 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*].

3. *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 $2^{\Theta(k \log k)}$.

None of the current techniques seemed to fit in this class of **connected packing-encodable** problems...

A classification of graph optimization problems

How can we certificate a solution in a middle set $\mathbf{mid}(e)$?

1. A subset of vertices of $\mathbf{mid}(e)$ (not restricted by some global condition).
   Examples: VERTEX COVER, DOMINATING SET.
   The size of the tables is bounded by $2^{\mathcal{O}(k)}$.

2. A *connected pairing* of vertices of $\mathbf{mid}(e)$.
   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*].

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

None of the current techniques seemed to fit in this class of **connected packing-encodable** problems...

A classification of graph optimization problems

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

1. 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 $2^{\mathcal{O}(k)}$.

2. 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*].

3. *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 $2^{\Theta(k \log k)}$.

   None of the current techniques seemed to fit in this class of **connected packing-encodable** problems...

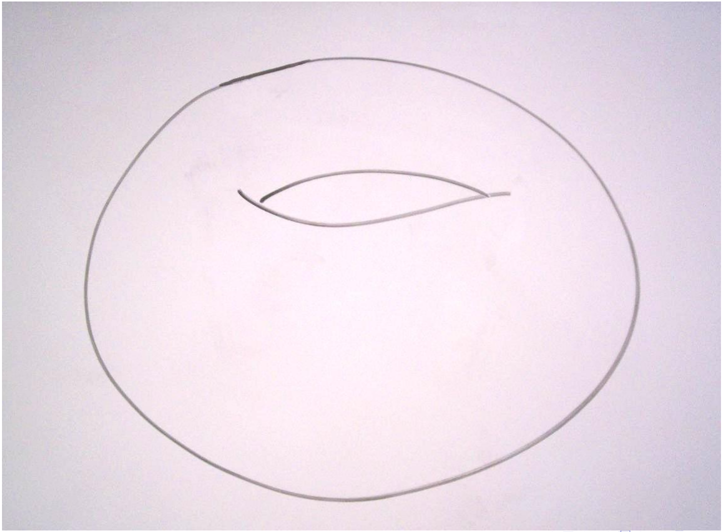
Next subsection is...

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.

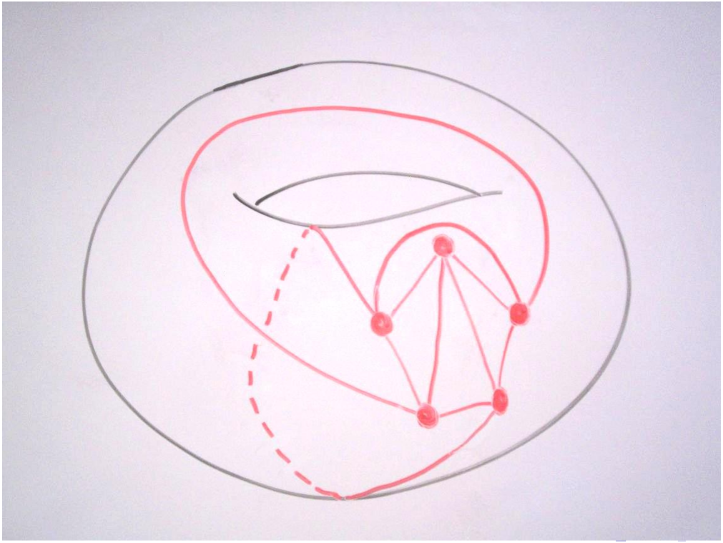
Nooses

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.



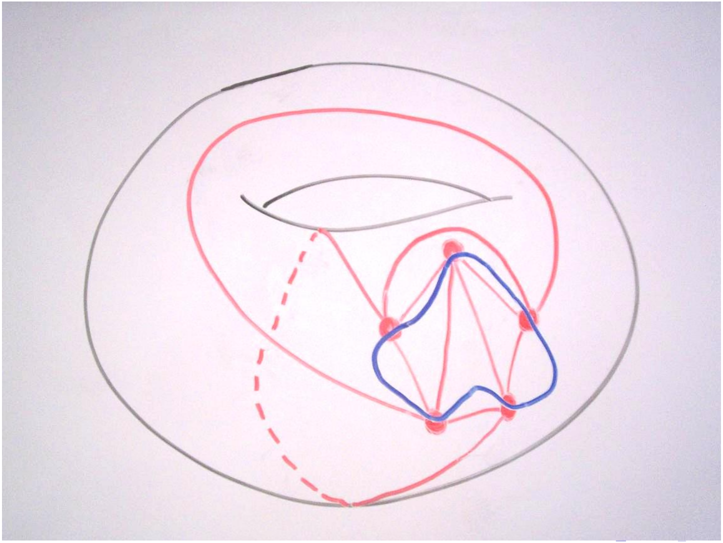
Nooses

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.



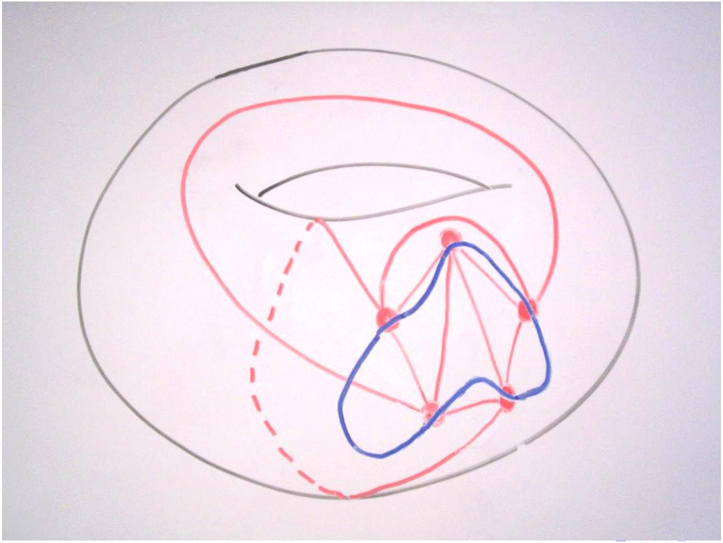
Nooses

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.



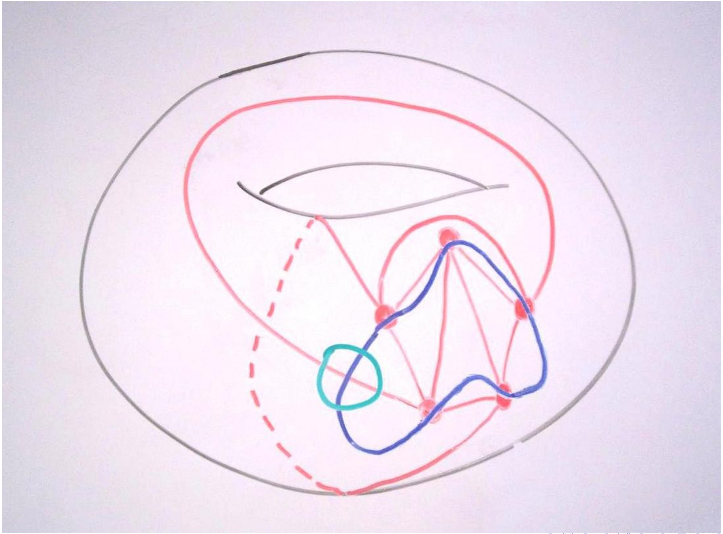
Nooses

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.



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.



Sphere cut decompositions

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?

- Exactly the number of *non-crossing partitions* over k elements, which is given by the k-th Catalan number:

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

Sphere cut decompositions

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?
- Exactly the number of *non-crossing partitions* over k elements, which is given by the k-th Catalan number:

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

Sphere cut decompositions

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?

- Exactly the number of *non-crossing partitions* over k elements, which is given by the k-th Catalan number:

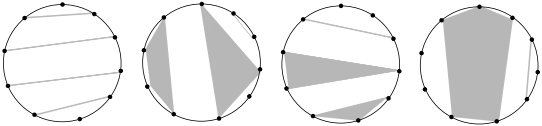
$$\mathrm{CN}(k) = \frac{1}{k+1}\binom{2k}{k} \sim \frac{4^k}{\sqrt{\pi}k^{3/2}} \approx 4^k.$$

Sphere cut decompositions

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?



- Exactly the number of *non-crossing partitions* over k elements, which is given by the k-th Catalan number:

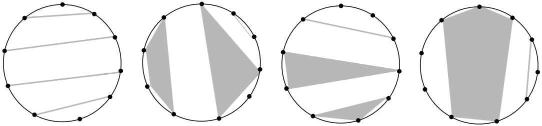
$$\mathrm{CN}(k) = \frac{1}{k+1}\binom{2k}{k} \sim \frac{4^k}{\sqrt{\pi}k^{3/2}} \approx 4^k.$$

Sphere cut decompositions

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?



- Exactly the number of *non-crossing partitions* over k elements, which is given by the k-th Catalan number:

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

“Old” idea for graphs on surfaces

Key idea for graphs on surfaces [Dorn *et al.* *SWAT’06*]:

- 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.

“Old” idea for graphs on surfaces

Key idea for graphs on surfaces [Dorn *et al.* *SWAT’06*]:

- 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.

“Old” idea for graphs on surfaces

Key idea for graphs on surfaces [Dorn *et al.* *SWAT’06*]:

- 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.

"Old" idea for graphs on surfaces

Key idea for graphs on surfaces [Dorn *et al.* *SWAT'06*]:

- 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.

From sphere to surface cut decompositions

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.

From sphere to surface cut decompositions

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.

From sphere to surface cut decompositions

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.

Surface cut decompositions (simplified version)

Let G be a graph embedded in a surface Σ, with $\mathbf{eg}(\Sigma) = \mathbf{g}$.

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

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

Surface cut decompositions (simplified version)

Let G be a graph embedded in a surface Σ, with $\mathbf{eg}(\Sigma) = \mathbf{g}$.

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

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

Surface cut decompositions (simplified version)

Let G be a graph embedded in a surface Σ, with $\mathbf{eg}(\Sigma) = \mathbf{g}$.

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

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

Surface cut decompositions (simplified version)

Let G be a graph embedded in a surface Σ, with $\mathbf{eg}(\Sigma) = \mathbf{g}$.

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

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

Surface cut decompositions (simplified version)

Let G be a graph embedded in a surface Σ, with $\mathbf{eg}(\Sigma) = \mathbf{g}$.

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

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

Main results (I)

Surface cut decompositions can be efficiently computed:

Theorem (Rué, Thilikos, and S.)

Given a G on n vertices embedded in a surface of Euler genus $\mathbf{g}$*, with* $\mathbf{bw}(G) \leq k$*, 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 $|A| = \mathcal{O}(\mathbf{g})$.
- 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.

Main results (I)

Surface cut decompositions can be efficiently computed:

Theorem (Rué, Thilikos, and S.)

Given a G on n vertices embedded in a surface of Euler genus $\mathbf{g}$*, with* $\mathbf{bw}(G) \leq k$*, 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 $|A| = \mathcal{O}(\mathbf{g})$.
- 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.

Main results (I)

Surface cut decompositions can be efficiently computed:

Theorem (Rué, Thilikos, and S.)

Given a G on n vertices embedded in a surface of Euler genus $\mathbf{g}$*, with* $\mathbf{bw}(G) \leq k$*, 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 $|A| = \mathcal{O}(\mathbf{g})$.
- 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.

Main results (II)

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 $\mathbf{g}$*, 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* $2^{\mathcal{O}(\log \mathbf{g} \cdot k + \log k \cdot \mathbf{g})}$.

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

Main results (II)

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 $\mathbf{g}$*, 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* $2^{\mathcal{O}(\log \mathbf{g} \cdot k + \log k \cdot \mathbf{g})}$.

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

Main results (II)

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 $\mathbf{g}$*, 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* $2^{\mathcal{O}(\log \mathbf{g} \cdot k + \log k \cdot \mathbf{g})}$.

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

Main enumerative result

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

Theorem (Rué, Thilikos, S.)

Let Σ be a surface whose boundary has $\beta(\Sigma)$ connected components. Then the number of non-crossing partitions on Σ with k vertices is asymptotically bounded by

$$f(\Sigma) \cdot k^{-3/2\chi(\Sigma)+\beta(\Sigma)-1} \cdot 4^k,$$

where

- $f(\Sigma)$ *is a function depending only on* Σ.
- $\chi(\Sigma)$ *is the Euler characteristic of* Σ: $\chi(\Sigma) = 2 - \mathbf{eg}(\Sigma) - \beta(\Sigma)$.

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

Main enumerative result

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

Theorem (Rué, Thilikos, S.)

Let Σ be a surface whose boundary has $\beta(\Sigma)$ connected components. Then the number of non-crossing partitions on Σ with k vertices is asymptotically bounded by

$$f(\Sigma) \cdot k^{-3/2\chi(\Sigma)+\beta(\Sigma)-1} \cdot 4^k,$$

where

- $f(\Sigma)$ *is a function depending only on* Σ.
- $\chi(\Sigma)$ *is the Euler characteristic of* Σ: $\chi(\Sigma) = 2 - \mathbf{eg}(\Sigma) - \beta(\Sigma)$.

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

How to use this framework?

- We presented a framework for the design of DP algorithms on **surface-embedded** graphs running in time $2^{\mathcal{O}(k)} \cdot n$.

- How to use this framework?

  1. Let **P** be a **connected packing-encodable** problem on a surface-embedded graph G.
  2. As a **preprocessing** step, build a **surface cut decomposition** of G, using the 1st Theorem.
  3. Run a "natural" **DP algorithm** to solve **P** over the obtained surface cut decomposition.
  4. The **single-exponential running time** of the algorithm is a consequence of the 2nd Theorem.

How to use this framework?

- We presented a framework for the design of DP algorithms on **surface-embedded** graphs running in time $2^{\mathcal{O}(k)} \cdot n$.
- How to use this framework?

1. Let **P** be a **connected packing-encodable** problem on a surface-embedded graph G.
2. As a **preprocessing** step, build a **surface cut decomposition** of G, using the 1st Theorem.
3. Run a “natural” **DP algorithm** to solve **P** over the obtained surface cut decomposition.
4. The **single-exponential running time** of the algorithm is a consequence of the 2nd Theorem.

How to use this framework?

- We presented a framework for the design of DP algorithms on **surface-embedded** graphs running in time $2^{\mathcal{O}(k)} \cdot n$.

- How to use this framework?

  1. Let **P** be a **connected packing-encodable** problem on a surface-embedded graph G.
  2. As a **preprocessing** step, build a **surface cut decomposition** of G, using the 1st Theorem.
  3. Run a "natural" **DP algorithm** to solve **P** over the obtained surface cut decomposition.
  4. The **single-exponential running time** of the algorithm is a consequence of the 2nd Theorem.

How to use this framework?

- We presented a framework for the design of DP algorithms on **surface-embedded** graphs running in time $2^{\mathcal{O}(k)} \cdot n$.

- How to use this framework?

  1. Let **P** be a **connected packing-encodable** problem on a surface-embedded graph G.

  2. As a **preprocessing** step, build a **surface cut decomposition** of G, using the 1st Theorem.

  3. Run a “natural” **DP algorithm** to solve **P** over the obtained surface cut decomposition.

  4. The **single-exponential running time** of the algorithm is a consequence of the 2nd Theorem.

How to use this framework?

- We presented a framework for the design of DP algorithms on **surface-embedded** graphs running in time $2^{\mathcal{O}(k)} \cdot n$.

- How to use this framework?

  1. Let **P** be a **connected packing-encodable** problem on a surface-embedded graph G.
  2. As a **preprocessing** step, build a **surface cut decomposition** of G, using the 1st Theorem.
  3. Run a “natural” **DP algorithm** to solve **P** over the obtained surface cut decomposition.
  4. The **single-exponential running time** of the algorithm is a consequence of the 2nd Theorem.

Next section is...

Structure of minor-free graphs

- Idea: use the structure of minor-free graphs.
- Some (simplified) preliminaries:
  - h-clique-sum of two graphs G_1 and G_2:
    choose cliques $K_1 \subseteq G_1$ and $K_2 \subseteq G_2$ with $|V(K_1)| = |V(K_2)| = h$, identify them, and possibly remove some edges of that clique.
  - Apex in an embedded graph:
    add a vertex with any neighbors in the embedded graph.
  - Vortex of depth h in an embedded graph:
    paste a graph of pathwidth at most h in a face of the embedding.
- Structure theorem [Robertson and Seymour]:
  Fix a graph H. There exists a constant $h = f(|V(H)|)$ such that any H-minor-free graph G can be decomposed (in a tree-like way) into h-clique-sums from h-almost-embeddable graphs: obtained from graphs of genus at most h by adding at most h apices and at most h vortices of depth at most h.

Structure of minor-free graphs

- Idea: use the structure of minor-free graphs.
- Some (simplified) preliminaries:
  - h-clique-sum of two graphs G_1 and G_2:
    choose cliques $K_1 \subseteq G_1$ and $K_2 \subseteq G_2$ with $|V(K_1)| = |V(K_2)| = h$, identify them, and possibly remove some edges of that clique.
  - Apex in an embedded graph:
    add a vertex with any neighbors in the embedded graph.
  - Vortex of depth h in an embedded graph:
    paste a graph of pathwidth at most h in a face of the embedding.
- Structure theorem [Robertson and Seymour]:
  Fix a graph H. There exists a constant $h = f(|V(H)|)$ such that any H-minor-free graph G can be decomposed (in a tree-like way) into h-clique-sums from h-almost-embeddable graphs: obtained from graphs of genus at most h by adding at most h apices and at most h vortices of depth at most h.

Structure of minor-free graphs

- Idea: use the structure of minor-free graphs.
- Some (simplified) preliminaries:
  - h-clique-sum of two graphs G_1 and G_2: choose cliques $K_1 \subseteq G_1$ and $K_2 \subseteq G_2$ with $|V(K_1)| = |V(K_2)| = h$, identify them, and possibly remove some edges of that clique.
  - Apex in an embedded graph: add a vertex with any neighbors in the embedded graph.
  - Vortex of depth h in an embedded graph: paste a graph of pathwidth at most h in a face of the embedding.
- Structure theorem [Robertson and Seymour]: Fix a graph H. There exists a constant $h = f(|V(H)|)$ such that any H-minor-free graph G can be decomposed (in a tree-like way) into h-clique-sums from h-almost-embeddable graphs: obtained from graphs of genus at most h by adding at most h apices and at most h vortices of depth at most h.

Structure of minor-free graphs

- Idea: use the structure of minor-free graphs.
- Some (simplified) preliminaries:
  - h-clique-sum of two graphs G_1 and G_2:
    choose cliques $K_1 \subseteq G_1$ and $K_2 \subseteq G_2$ with $|V(K_1)| = |V(K_2)| = h$,
    identify them, and possibly remove some edges of that clique.
  - Apex in an embedded graph:
    add a vertex with any neighbors in the embedded graph.
  - Vortex of depth h in an embedded graph:
    paste a graph of pathwidth at most h in a face of the embedding.
- Structure theorem [Robertson and Seymour]:
  Fix a graph H. There exists a constant $h = f(|V(H)|)$ such that any H-minor-free graph G can be decomposed (in a tree-like way) into h-clique-sums from h-almost-embeddable graphs:
  obtained from graphs of genus at most h by adding at most h apices and at most h vortices of depth at most h.

Structure of minor-free graphs

- Idea: use the structure of minor-free graphs.
- Some (simplified) preliminaries:
  - h-clique-sum of two graphs G_1 and G_2:
    choose cliques $K_1 \subseteq G_1$ and $K_2 \subseteq G_2$ with $|V(K_1)| = |V(K_2)| = h$, identify them, and possibly remove some edges of that clique.
  - Apex in an embedded graph:
    add a vertex with any neighbors in the embedded graph.
  - Vortex of depth h in an embedded graph:
    paste a graph of pathwidth at most h in a face of the embedding.
- Structure theorem [Robertson and Seymour]:
  Fix a graph H. There exists a constant $h = f(|V(H)|)$ such that any H-minor-free graph G can be decomposed (in a tree-like way) into h-clique-sums from h-almost-embeddable graphs: obtained from graphs of genus at most h by adding at most h apices and at most h vortices of depth at most h.

Structure of minor-free graphs

- Idea: use the structure of minor-free graphs.
- Some (simplified) preliminaries:
  - h-clique-sum of two graphs G_1 and G_2: choose cliques $K_1 \subseteq G_1$ and $K_2 \subseteq G_2$ with $|V(K_1)| = |V(K_2)| = h$, identify them, and possibly remove some edges of that clique.
  - Apex in an embedded graph: add a vertex with any neighbors in the embedded graph.
  - Vortex of depth h in an embedded graph: paste a graph of pathwidth at most h in a face of the embedding.
- Structure theorem [Robertson and Seymour]: Fix a graph H. There exists a constant $h = f(|V(H)|)$ such that any H-minor-free graph G can be decomposed (in a tree-like way) into h-clique-sums from h-almost-embeddable graphs: obtained from graphs of genus at most h by adding at most h apices and at most h vortices of depth at most h.

Structure of minor-free graphs

- Idea: use the structure of minor-free graphs.
- Some (simplified) preliminaries:
  - h-clique-sum of two graphs G_1 and G_2:
    choose cliques $K_1 \subseteq G_1$ and $K_2 \subseteq G_2$ with $|V(K_1)| = |V(K_2)| = h$,
    identify them, and possibly remove some edges of that clique.
  - Apex in an embedded graph:
    add a vertex with any neighbors in the embedded graph.
  - Vortex of depth h in an embedded graph:
    paste a graph of pathwidth at most h in a face of the embedding.
- Structure theorem [Robertson and Seymour]:
  Fix a graph H. There exists a constant $h = f(|V(H)|)$ such that any H-minor-free graph G can be decomposed (in a tree-like way) into h-clique-sums from h-almost-embeddable graphs:
  obtained from graphs of genus at most h by adding at most h apices and at most h vortices of depth at most h.

Extension to minor-free graphs

- Strategy: use an extension of ***surface cut decomposition*** in each almost-embeddable graph, and then merge them.
- The clique-sums and the apices are "easy" to deal with, but the vortices are more complicated...
- We can capture their combinatorial behavior with h-triangulations: partition in the disk in which no subset of $h+1$ blocks pairwise intersect. *(A non-crossing partition is a* 1*-triangulation.)*
- It is known that the $\#$ of h-triangulations on k elements satisfies

$$T_h(k) \quad \leq_{k\to\infty} \quad \frac{h!}{\pi^{h/2}} \cdot k^{-3h/2} \cdot 4^{hk}$$

Extension to minor-free graphs

- Strategy: use an extension of ***surface cut decomposition*** in each almost-embeddable graph, and then merge them.
- The clique-sums and the apices are "easy" to deal with, but the vortices are more complicated...
- We can capture their combinatorial behavior with h-triangulations: partition in the disk in which no subset of $h+1$ blocks pairwise intersect. *(A non-crossing partition is a* 1*-triangulation.)*
- It is known that the $\#$ of h-triangulations on k elements satisfies

$$T_h(k) \quad \leq_{k\to\infty} \quad \frac{h!}{\pi^{h/2}} \cdot k^{-3h/2} \cdot 4^{hk}$$

Extension to minor-free graphs

- Strategy: use an extension of ***surface cut decomposition*** in each almost-embeddable graph, and then merge them.
- The clique-sums and the apices are "easy" to deal with, but the vortices are more complicated...
- We can capture their combinatorial behavior with h-triangulations: partition in the disk in which no subset of $h+1$ blocks pairwise intersect. *(A non-crossing partition is a 1-triangulation.)*
- It is known that the $\#$ of h-triangulations on k elements satisfies

$$T_h(k) \leq_{k\to\infty} \frac{h!}{\pi^{h/2}} \cdot k^{-3h/2} \cdot 4^{hk}$$

Extension to minor-free graphs

- Strategy: use an extension of ***surface cut decomposition*** in each almost-embeddable graph, and then merge them.
- The clique-sums and the apices are "easy" to deal with, but the vortices are more complicated...
- We can capture their combinatorial behavior with h-triangulations: partition in the disk in which no subset of $h+1$ blocks pairwise intersect. *(A non-crossing partition is a* 1*-triangulation.)*
- It is known that the $\#$ of h-triangulations on k elements satisfies

$$T_h(k) \leq_{k\to\infty} \frac{h!}{\pi^{h/2}} \cdot k^{-3h/2} \cdot 4^{hk}$$

Extension to minor-free graphs

- Strategy: use an extension of ***surface cut decomposition*** in each almost-embeddable graph, and then merge them.
- The clique-sums and the apices are "easy" to deal with, but the vortices are more complicated...
- We can capture their combinatorial behavior with h-triangulations: partition in the disk in which no subset of $h+1$ blocks pairwise intersect. *(A non-crossing partition is a* 1*-triangulation.)*
- It is known that the $\#$ of h-triangulations on k elements satisfies

$$T_h(k) \quad \leq_{k\to\infty} \quad \frac{h!}{\pi^{h/2}} \cdot k^{-3h/2} \cdot 4^{hk}$$

Next section is...

1. Motivation
2. Graphs on surfaces
   - Preliminaries
   - Main ideas of our approach
3. Extension to minor-free graphs
4. Conclusions
   - Further research
   - Some recent results

Next subsection is...

Further research

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?

Further research

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?

Further research

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?

Further research

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?

Further research

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?

Further research

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

Next subsection is...

1. Motivation
2. Graphs on surfaces
   - Preliminaries
   - Main ideas of our approach
3. Extension to minor-free graphs
4. Conclusions
   - Further research
   - Some recent results

Lower bounds (I)

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

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

- Disjoint Paths cannot be solved in $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ time.
- d-Distortion cannot be solved in $2^{o(d \log d)} \cdot n^{\mathcal{O}(1)}$ 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 $2^{\mathcal{O}(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ time optimal?
- Parameterizing by solution size: Directed FVS, Interval Completion... Is $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ time optimal?

Lower bounds (I)

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

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

- DISJOINT PATHS cannot be solved in $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ time.
- d-DISTORTION cannot be solved in $2^{o(d \log d)} \cdot n^{\mathcal{O}(1)}$ 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 $2^{\mathcal{O}(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ time optimal?
- Parameterizing by solution size: DIRECTED FVS, INTERVAL COMPLETION... Is $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ time optimal?

Lower bounds (I)

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

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

- DISJOINT PATHS cannot be solved in $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ time.
- d-DISTORTION cannot be solved in $2^{o(d \log d)} \cdot n^{\mathcal{O}(1)}$ 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 $2^{\mathcal{O}(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ time optimal?
- Parameterizing by solution size: DIRECTED FVS, INTERVAL COMPLETION... Is $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ time optimal?

Lower bounds (I)

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

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

- DISJOINT PATHS cannot be solved in $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ time.
- d-DISTORTION cannot be solved in $2^{o(d \log d)} \cdot n^{\mathcal{O}(1)}$ 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 $2^{\mathcal{O}(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ time optimal?
- Parameterizing by solution size: DIRECTED FVS, INTERVAL COMPLETION... Is $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ time optimal?

Lower bounds (I)

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

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

- Disjoint Paths cannot be solved in $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ time.
- d-Distortion cannot be solved in $2^{o(d \log d)} \cdot n^{\mathcal{O}(1)}$ 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 $2^{\mathcal{O}(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ time optimal?
- Parameterizing by solution size: Directed FVS, Interval Completion... Is $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ time optimal?

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 $\mathcal{O}(2-\varepsilon)^n$ time (SETH), then for any $\varepsilon > 0$:

- Independent Set cannot be solved in $(2-\varepsilon)^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- Dominating Set cannot be solved in $(3-\varepsilon)^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- Max Cut cannot be solved in $(2-\varepsilon)^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- For any $q \geq 3$, q-Coloring cannot be solved in $(q-\varepsilon)^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$.

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 $\mathcal{O}(2-\varepsilon)^n$ time (SETH), then for any $\varepsilon > 0$:

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

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 $\mathcal{O}(2-\varepsilon)^n$ time (SETH), then for any $\varepsilon > 0$:

- Independent Set cannot be solved in $(2-\varepsilon)^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- Dominating Set cannot be solved in $(3-\varepsilon)^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- Max Cut cannot be solved in $(2-\varepsilon)^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- For any $q \geq 3$, q-Coloring cannot be solved in $(q-\varepsilon)^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$.

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 $\mathcal{O}(2-\varepsilon)^n$ time (SETH), then for any $\varepsilon > 0$:

- Independent Set cannot be solved in $(2-\varepsilon)^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- Dominating Set cannot be solved in $(3-\varepsilon)^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- Max Cut cannot be solved in $(2-\varepsilon)^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$ time.
- For any $q \geq 3$, q-Coloring cannot be solved in $(q-\varepsilon)^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$.

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?

Randomized single-exponential algorithms

[Cygan, Nederlof, Pilipczuk, Pilipczuk, van Rooij, Wojtaszczyk. *arXiv, March 2, 2011*]:

They present randomized algorithms for connected packing-encodable problems in general graphs with time $c^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$.

- They introduce a dynamic programming technique called Cut&Count.
  (It relies on a probabilistic result called the Isolation Lemma.)
- In addition, they provide tight lower bounds for many connected packing-encodable problems.

OPEN QUESTIONS:

- Can these algorithms be derandomized?
- Can the lower bounds be improved to $c^{\mathbf{tw}\log\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$?

Randomized single-exponential algorithms

[Cygan, Nederlof, Pilipczuk, Pilipczuk, van Rooij, Wojtaszczyk. *arXiv, March 2, 2011*]:

They present randomized algorithms for connected packing-encodable problems in general graphs with time $c^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$.

- They introduce a dynamic programming technique called Cut&Count.
  (It relies on a probabilistic result called the Isolation Lemma.)
- In addition, they provide tight lower bounds for many connected packing-encodable problems.

OPEN QUESTIONS:

- Can these algorithms be derandomized?
- Can the lower bounds be improved to $c^{\mathbf{tw}\log\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$?

Randomized single-exponential algorithms

[Cygan, Nederlof, Pilipczuk, Pilipczuk, van Rooij, Wojtaszczyk. *arXiv, March 2, 2011*]:

They present randomized algorithms for connected packing-encodable problems in general graphs with time $c^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$.

- They introduce a dynamic programming technique called Cut&Count.
  (It relies on a probabilistic result called the Isolation Lemma.)
- In addition, they provide tight lower bounds for many connected packing-encodable problems.

OPEN QUESTIONS:

- Can these algorithms be derandomized?
- Can the lower bounds be improved to $c^{\mathbf{tw} \log \mathbf{tw}} \cdot n^{\mathcal{O}(1)}$?

Randomized single-exponential algorithms

[Cygan, Nederlof, Pilipczuk, Pilipczuk, van Rooij, Wojtaszczyk. *arXiv, March 2, 2011*]:

They present randomized algorithms for connected packing-encodable problems in general graphs with time $c^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$.

- They introduce a dynamic programming technique called Cut&Count.
  (It relies on a probabilistic result called the Isolation Lemma.)
- In addition, they provide tight lower bounds for many connected packing-encodable problems.

OPEN QUESTIONS:

- Can these algorithms be derandomized?
- Can the lower bounds be improved to $c^{\mathbf{tw}\log\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$?

Randomized single-exponential algorithms

[Cygan, Nederlof, Pilipczuk, Pilipczuk, van Rooij, Wojtaszczyk. *arXiv, March 2, 2011*]:

They present randomized algorithms for connected packing-encodable problems in general graphs with time $c^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$.

- They introduce a dynamic programming technique called Cut&Count.
  (It relies on a probabilistic result called the Isolation Lemma.)
- In addition, they provide tight lower bounds for many connected packing-encodable problems.

OPEN QUESTIONS:

- Can these algorithms be derandomized?
- Can the lower bounds be improved to $c^{\mathbf{tw}\log\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$?

Randomized single-exponential algorithms

[Cygan, Nederlof, Pilipczuk, Pilipczuk, van Rooij, Wojtaszczyk. *arXiv, March 2, 2011*]:

They present randomized algorithms for connected packing-encodable problems in general graphs with time $c^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$.

- They introduce a dynamic programming technique called Cut&Count.
  (It relies on a probabilistic result called the Isolation Lemma.)
- In addition, they provide tight lower bounds for many connected packing-encodable problems.

OPEN QUESTIONS:

- Can these algorithms be derandomized?
- Can the lower bounds be improved to $c^{\mathbf{tw}\log \mathbf{tw}} \cdot n^{\mathcal{O}(1)}$?

Randomized single-exponential algorithms

[Cygan, Nederlof, Pilipczuk, Pilipczuk, van Rooij, Wojtaszczyk. *arXiv, March 2, 2011*]:

They present randomized algorithms for connected packing-encodable problems in general graphs with time $c^{\mathbf{tw}} \cdot n^{\mathcal{O}(1)}$.

- They introduce a dynamic programming technique called Cut&Count.
  (It relies on a probabilistic result called the Isolation Lemma.)
- In addition, they provide tight lower bounds for many connected packing-encodable problems.

OPEN QUESTIONS:

- Can these algorithms be derandomized?
- Can the lower bounds be improved to $c^{\mathbf{tw} \log \mathbf{tw}} \cdot n^{\mathcal{O}(1)}$?

Gràcies!