Dynamic programming in sparse graphs

Ignasi Sau

CNRS, LIRMM, Montpellier, France

Université Libre de Bruxelles, March 29 2012

Joint work with:

Juanjo Rué
Instituto de Ciencias Matemáticas, Madrid, Spain

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

Outline

Next section is...

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

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** $\leq k$.

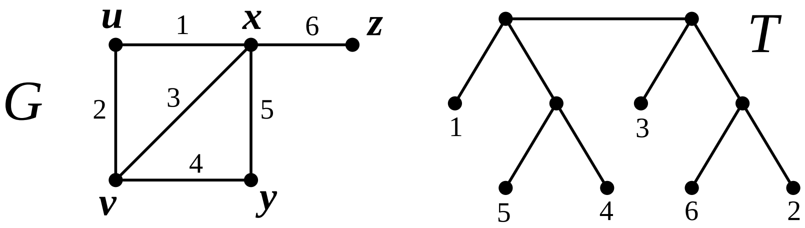
- **Problem**: $f(k)$ can be huge!!! (for instance, $f(k) = 2^{3^{4^{5^{6^{k}}}}}$)

- 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**.

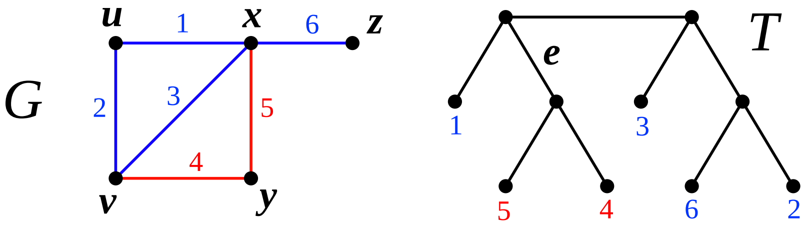
Branch decompositions and branchwidth



A branch decomposition of a graph G is a pair (T, μ):

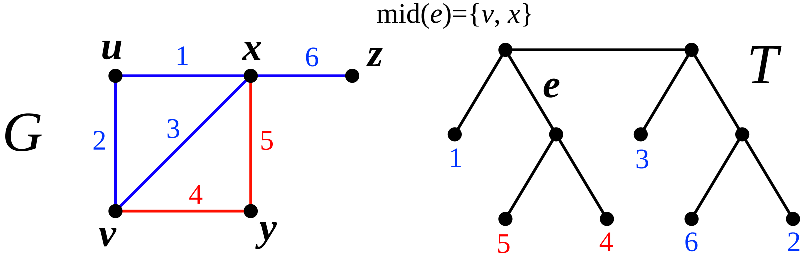
- T is a tree where all internal vertices have degree 3.
- μ is a bijection between the leaves of T and $E(G)$.

Branch decompositions and branchwidth



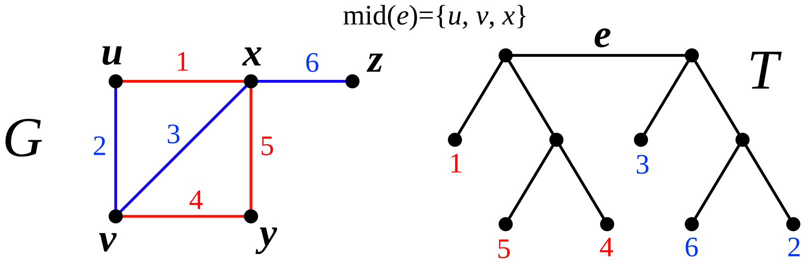
Each edge $e \in T$ partitions $E(G)$ into two sets A_e and B_e.

Branch decompositions and branchwidth



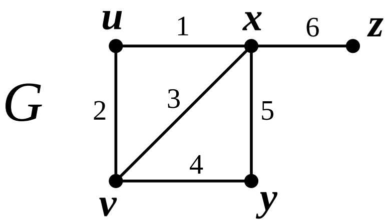
For each $e \in E(T)$, we define **mid**$(e) = V(A_e) \cap V(B_e)$.

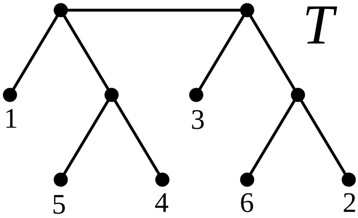
Branch decompositions and branchwidth



The width of (T, μ) is $\max_{e \in E(T)} |\mathbf{mid}(e)|$.

Branch decompositions and branchwidth

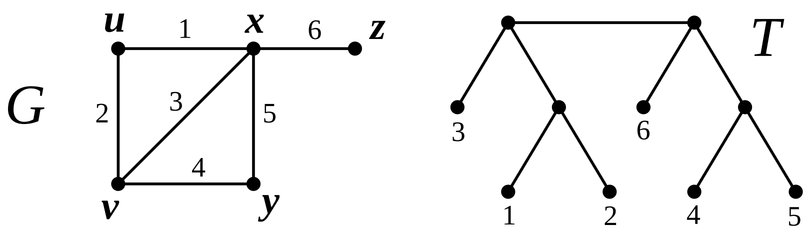




The branchwidth of a graph G, $\mathbf{bw}(G)$, is the minimum width over all branch decompositions of G:

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

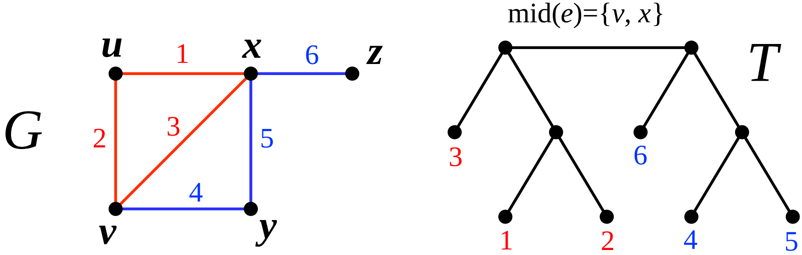
Branch decompositions and branchwidth



The branchwidth of a graph G, **bw**(G), is the minimum width over all branch decompositions of G:

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

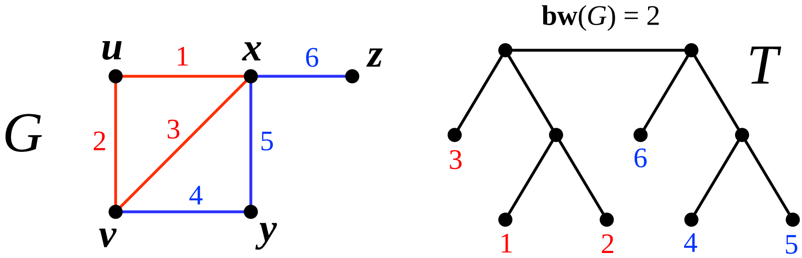
Branch decompositions and branchwidth



The branchwidth of a graph G, **bw**(G), is the minimum width over all branch decompositions of G:

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

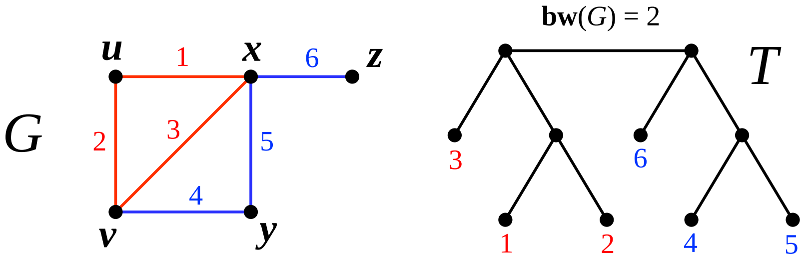
Branch decompositions and branchwidth



The branchwidth of a graph G, **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



We have the following relationship for graphs G such that $|E(G)| \geq 3$:

$$\mathbf{bw}(G) \;\leq\; \mathbf{tw}(G) + 1 \;\leq\; \frac{3}{2}\mathbf{bw}(G).$$

[Robertson i Seymour. *JCTSB'91*]

Next section is...

Next subsection is...

1. Motivation

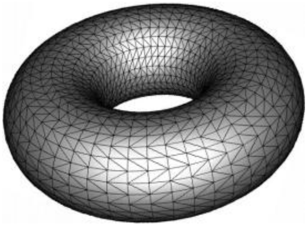
2. Graphs on surfaces
   - Preliminaries
   - Main ideas of our approach

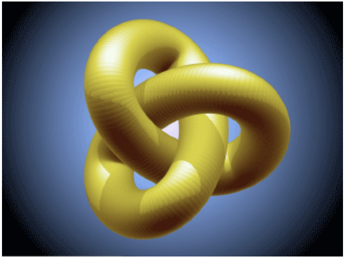
3. Extension to H-minor-free graphs

4. Some recent results

Surfaces

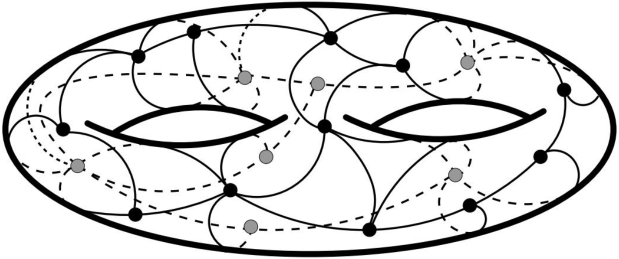
- **SURFACE** = TOPOLOGICAL SPACE, LOCALLY “FLAT”





Graphs on surfaces

Embedded graph: graph drawn on a surface, with no edge-crossings.



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

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 $|\mathbf{mid}(e)| \leq k$ 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 $|\mathbf{mid}(e)| \leq k$ in the running time of the DP.
- The precise definition of the tables of the DP depends on each particular problem.

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 **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...

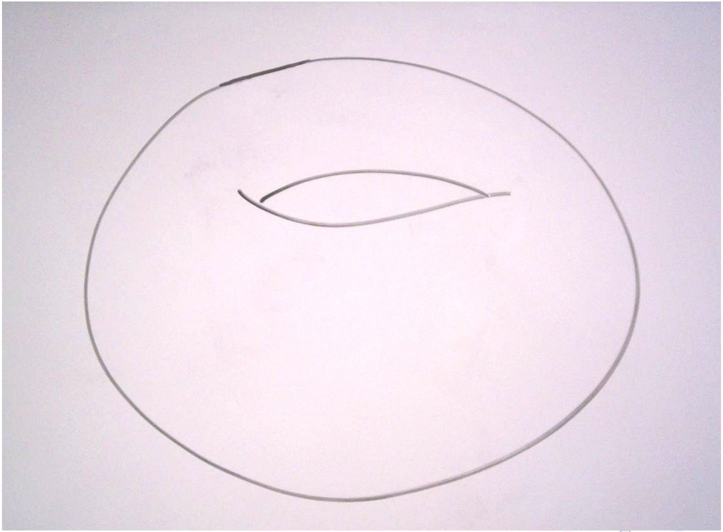
Next subsection is...

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.

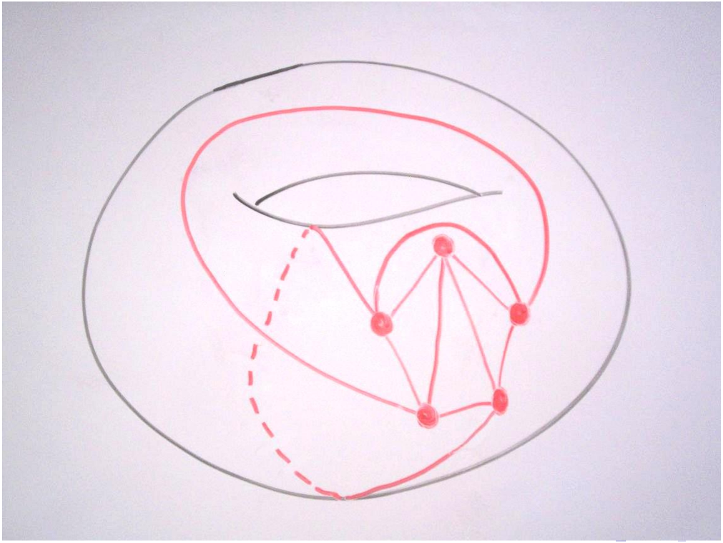
Nooses

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.



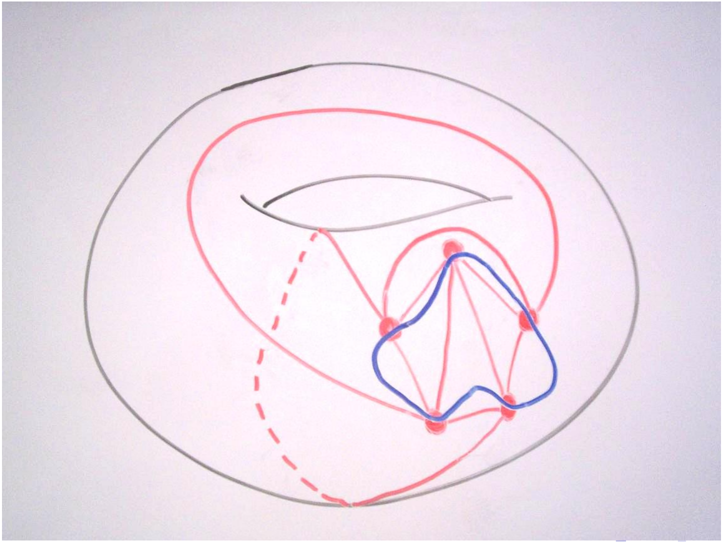
Nooses

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.



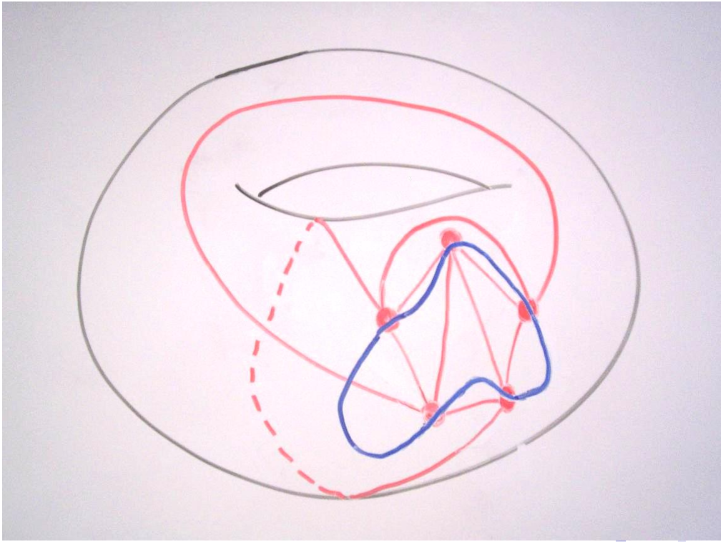
Nooses

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.



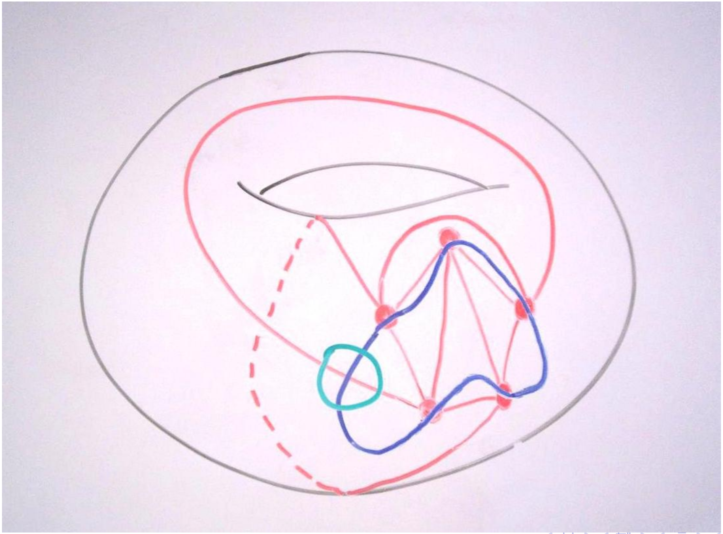
Nooses

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:

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

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:

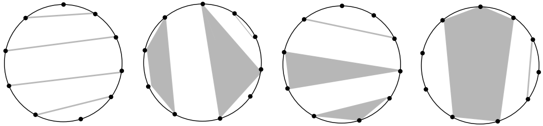
$$\text{CN}(k) = \frac{1}{k+1}\binom{2k}{k} \sim \frac{4^k}{\sqrt{\pi}k^{3/2}} \approx 4^k.$$

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:

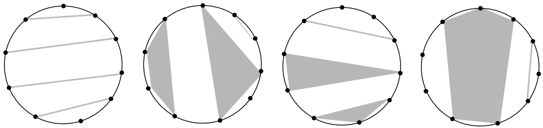
$$\mathrm{CN}(k) = \frac{1}{k+1}\binom{2k}{k} \sim \frac{4^k}{\sqrt{\pi}k^{3/2}} \approx 4^k.$$

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:

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

"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.

Sphere cut ⇝ surface cut

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.

Sphere cut $\rightsquigarrow$ surface cut

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.

Sphere cut $\rightsquigarrow$ surface cut

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
  - ⋆ 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.

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

1. 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* $27k + \mathcal{O}(\mathbf{g})$*.*

2. DP on surface cut decompositions 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* $2^{\mathcal{O}(\log \mathbf{g} \cdot k + \log k \cdot \mathbf{g})}$*.*

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

Main results

1. 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* $27k + \mathcal{O}(\mathbf{g})$*.*

2. DP on surface cut decompositions 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* $2^{\mathcal{O}(\log \mathbf{g} \cdot k + \log k \cdot \mathbf{g})}$*.*

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

Main results

1. 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* $27k + \mathcal{O}(\mathbf{g})$*.*

2. DP on surface cut decompositions 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* $2^{\mathcal{O}(\log \mathbf{g} \cdot k + \log k \cdot \mathbf{g})}$*.*

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

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 H-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 (1983-2012)]:
  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 H-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 (1983-2012)]:
  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 H-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 (1983-2012)]:
  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 H-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 (1983-2012)]:
  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 H-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 (1983-2012)]:
  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 H-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 (1983-2012)]:
  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 H-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 (1983-2012)]:
  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 H-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 H-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 H-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 H-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

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

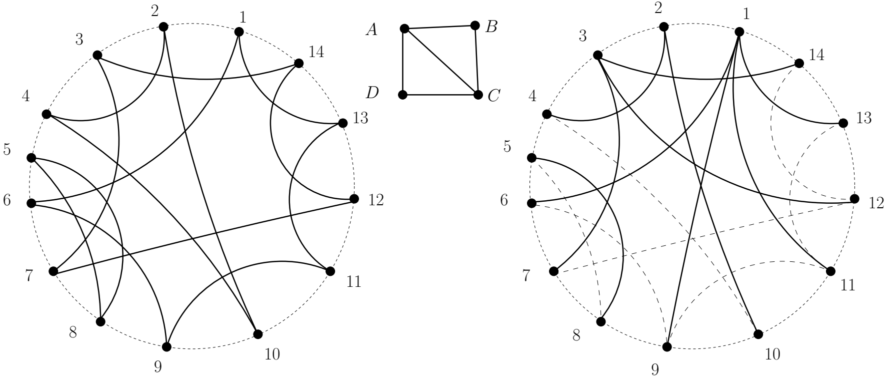
Extension to H-minor-free graphs

- 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

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

Example of a 3-triangulation

A 3-triangulation of the disc $\mathbb{D}_{14}$ with four blocks $A = \{1, 6, 9, 11, 13\}$, $B = \{2, 4, 10\}$, $C = \{3, 7, 12, 14\}$, and $D = \{5, 8\}$.



A partition is an h-triangulation iff its incidence graph has clique size $\leq h$.

H-minor-free cut decompositions

- In order to define *H*-minor-free cut decompositions, we first need a suitable version of the Robertson & Seymour Structure Theorem, in which every *h*-almost-embeddable piece is embedded in a polyhedral way: it is 3-vertex-connected, and the shortest non-contractible noose has length ≥ 3.
- Then, *H*-minor-free cut decompositions are defined in the "natural" way (quite technical)...
- We just give some intuition about how to deal with the vortices.
- Connected packing: collection of vertex-disjoint connected subgraphs of the input graph. We are interested in their intersection with the middle sets.

H-minor-free cut decompositions

- In order to define *H*-minor-free cut decompositions, we first need a suitable version of the Robertson & Seymour Structure Theorem, in which every *h*-almost-embeddable piece is embedded in a polyhedral way: it is 3-vertex-connected, and the shortest non-contractible noose has length ≥ 3.
- Then, *H*-minor-free cut decompositions are defined in the "natural" way (quite technical)...
- We just give some intuition about how to deal with the vortices.
- Connected packing: collection of vertex-disjoint connected subgraphs of the input graph. We are interested in their intersection with the middle sets.

H-minor-free cut decompositions

- In order to define H-minor-free cut decompositions, we first need a suitable version of the Robertson & Seymour Structure Theorem, in which every h-almost-embeddable piece is embedded in a polyhedral way: it is 3-vertex-connected, and the shortest non-contractible noose has length ≥ 3.
- Then, H-minor-free cut decompositions are defined in the "natural" way (quite technical)...
- We just give some intuition about how to deal with the vortices.
- Connected packing: collection of vertex-disjoint connected subgraphs of the input graph. We are interested in their intersection with the middle sets.

H-minor-free cut decompositions

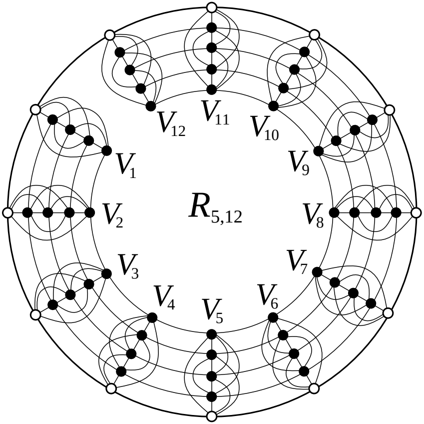
- In order to define H-minor-free cut decompositions, we first need a suitable version of the Robertson & Seymour Structure Theorem, in which every h-almost-embeddable piece is embedded in a polyhedral way: it is 3-vertex-connected, and the shortest non-contractible noose has length ≥ 3.
- Then, H-minor-free cut decompositions are defined in the "natural" way (quite technical)...
- We just give some intuition about how to deal with the vortices.
- Connected packing: collection of vertex-disjoint connected subgraphs of the input graph. We are interested in their intersection with the middle sets.

H-minor-free cut decompositions

- In order to define H-minor-free cut decompositions, we first need a suitable version of the Robertson & Seymour Structure Theorem, in which every h-almost-embeddable piece is embedded in a polyhedral way: it is 3-vertex-connected, and the shortest non-contractible noose has length ≥ 3.
- Then, H-minor-free cut decompositions are defined in the "natural" way (quite technical)...
- We just give some intuition about how to deal with the vortices.
- Connected packing: collection of vertex-disjoint connected subgraphs of the input graph. We are interested in their intersection with the middle sets.

Vortex patterns

Vortex of depth h in an embedded graph:
paste a graph of pathwidth at most h in a face of the embedding.

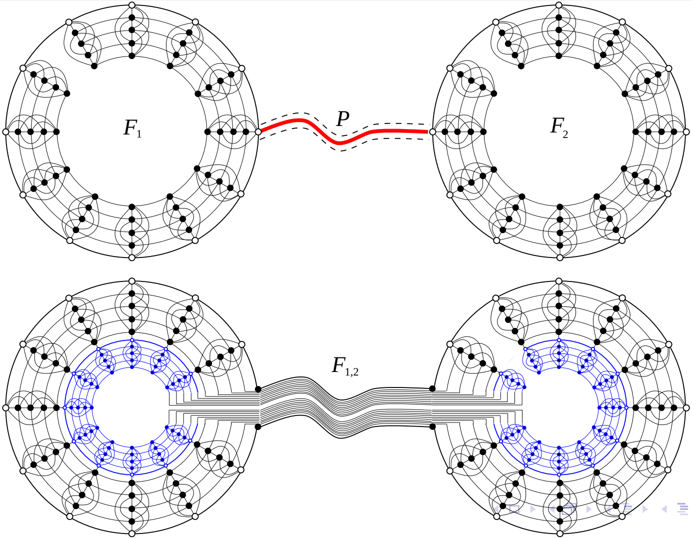


It can be easily seen that each vortex is a minor of a vortex pattern (preserving the vertices in the face of the embedding).

Merging vortices

Lemma

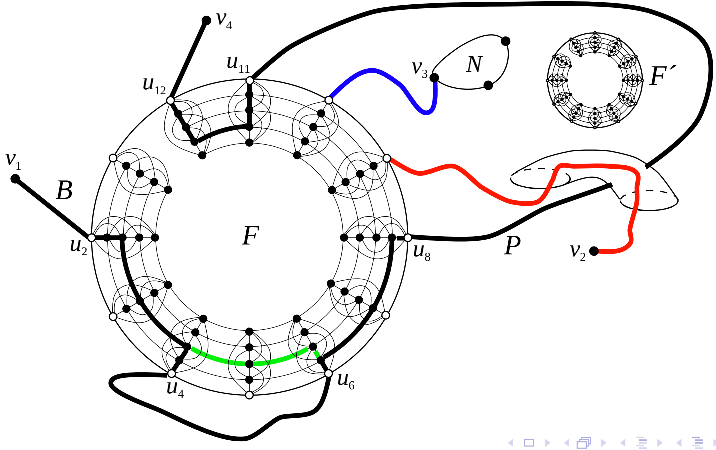
We may assume that each connected subgraph meets at most one vortex.



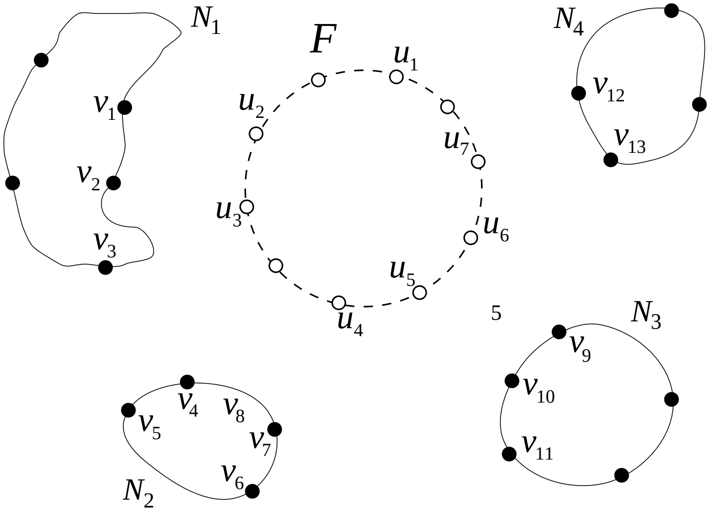
How connected subgraphs can cross a vortex

Lemma

We may assume that the total number of times that the subgraphs in a connected packing meet each vortex is $O_h(k)$.

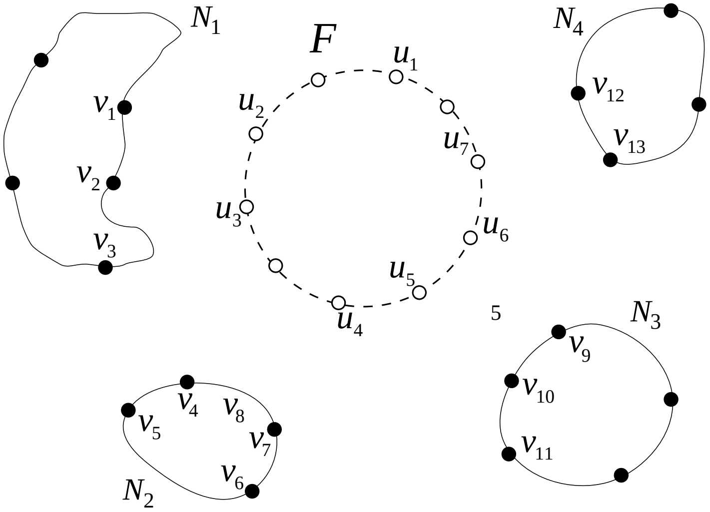


Simulating the behavior of a vortex



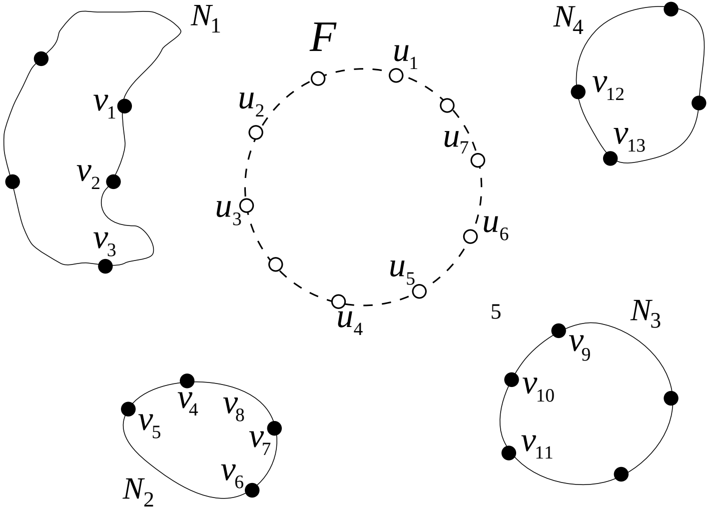
Example (in the plane) of our approach to simulate the behavior of the vortices.

Simulating the behavior of a vortex



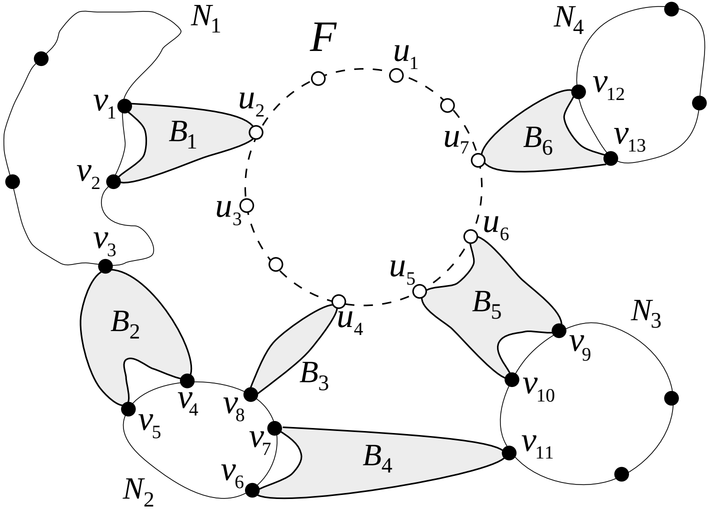
There are four nooses N_1, N_2, N_3, N_4 (drawn with full lines), and one vortex F of depth 2 (drawn with a dashed circle).

Simulating the behavior of a vortex



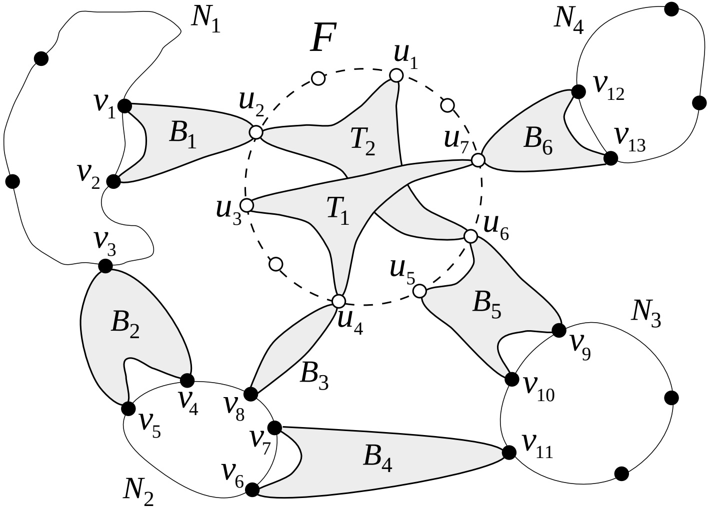
Black vertices correspond to vertices in the separator S (thus, in the nooses), while white vertices belong to the base set of the vortex.

Simulating the behavior of a vortex



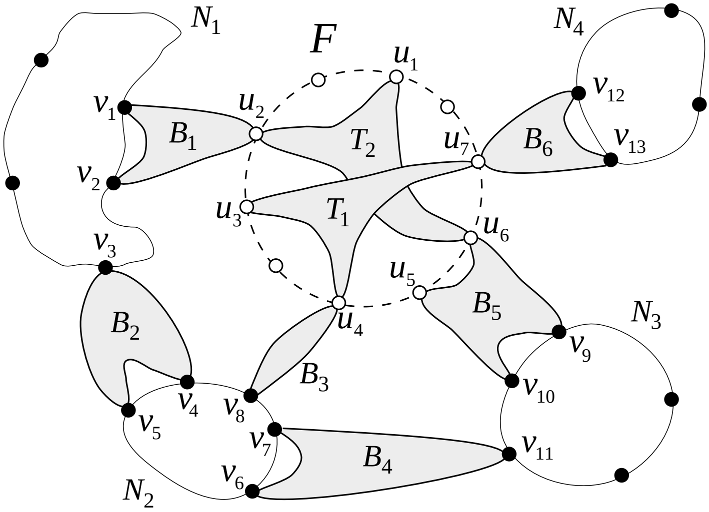
The non-crossing packing in Σ has six connected subgraphs B_1, B_2, B_3, B_4, B_5, and B_6.

Simulating the behavior of a vortex



The 2-triangulation of the vortex F has two connected subgraphs T_1 and T_2.

Simulating the behavior of a vortex



With the two subgraphs T_1 and T_2 corresponding to a 2-triangulation of the vortex, subgraphs B_1 and B_5 (resp. B_6 and B_6) get merged.

... the final result

Theorem

Every connected packing-encodable problem whose input is an n-vertex graph G that excludes an h-vertex graph H as a minor, and has branchwidth at most k, can be solved by a DP algorithm on an H-minor-free cut decomposition of G with tables of size $2^{O_h(k)} \cdot n^{O(1)}$.

We prove that, given an H-minor-free graph G, an H-minor-free cut decomposition of G of width $O_h(\mathbf{bw}(G))$ can be constructed in $O_h(n^3)$ time. Therefore, we conclude the following result.

Theorem

Every connected packing-encodable problem whose input is an n-vertex graph G that excludes an h-vertex graph H as a minor and has branchwidth at most k, can be solved in $2^{O_h(k)} \cdot n^{O(1)}$ steps.

Theorem

Every connected packing-encodable problem whose input is an n-vertex graph G that excludes an h-vertex graph H as a minor, and has branchwidth at most k, can be solved by a DP algorithm on an H-minor-free cut decomposition of G with tables of size $2^{O_h(k)} \cdot n^{O(1)}$.

We prove that, given an H-minor-free graph G, an H-minor-free cut decomposition of G of width $O_h(\mathbf{bw}(G))$ can be constructed in $O_h(n^3)$ time. Therefore, we conclude the following result.

Theorem

Every connected packing-encodable problem whose input is an n-vertex graph G that excludes an h-vertex graph H as a minor and has branchwidth at most k, can be solved in $2^{O_h(k)} \cdot n^{O(1)}$ steps.

... the final result

Theorem

Every connected packing-encodable problem whose input is an n-vertex graph G that excludes an h-vertex graph H as a minor, and has branchwidth at most k, can be solved by a DP algorithm on an H-minor-free cut decomposition of G with tables of size $2^{O_h(k)} \cdot n^{O(1)}$.

We prove that, given an H-minor-free graph G, an H-minor-free cut decomposition of G of width $O_h(\mathbf{bw}(G))$ can be constructed in $O_h(n^3)$ time. Therefore, we conclude the following result.

Theorem

Every connected packing-encodable problem whose input is an n-vertex graph G that excludes an h-vertex graph H as a minor and has branchwidth at most k, can be solved in $2^{O_h(k)} \cdot n^{O(1)}$ steps.

Next section is...

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

Some recent results

1. For an FPT problem, is it **always** possible to obtain algorithms with running time $2^{\mathcal{O}(\mathbf{tw})} \cdot n^{\mathcal{O}(1)}$?

   [Lokshtanov, Marx, Saurabh. *SODA'11*]
   If 3SAT cannot be solved in time $2^{o(n)}$, then DISJOINT PATHS cannot be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ in general graphs.

   - HAMILTONIAN PATH, FVS, CONNECTED VERTEX COVER, ... Is $2^{\mathcal{O}(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ best possible?

2. Randomized algorithms for connected packing-encodable problems in general graphs in time $2^{\mathcal{O}(\mathbf{tw})} \cdot n^{\mathcal{O}(1)}$.

   [Cygan, Nederlof, (Pilipczuk)^2, van Rooij, Wojtaszczyk. *FOCS'11*]

   - They introduce a DP technique called Cut&Count. *(It relies on a probabilistic result called the Isolation Lemma.)*
   - Can these algorithms be derandomized?

Some recent results

1. For an FPT problem, is it always possible to obtain algorithms with running time $2^{\mathcal{O}(\mathbf{tw})} \cdot n^{\mathcal{O}(1)}$?

   [Lokshtanov, Marx, Saurabh. *SODA'11*]
   If 3SAT cannot be solved in time $2^{o(n)}$, then DISJOINT PATHS cannot be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ in general graphs.

   - HAMILTONIAN PATH, FVS, CONNECTED VERTEX COVER, ...
     Is $2^{\mathcal{O}(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ best possible?

2. Randomized algorithms for connected packing-encodable problems in general graphs in time $2^{\mathcal{O}(\mathbf{tw})} \cdot n^{\mathcal{O}(1)}$.

   [Cygan, Nederlof, (Pilipczuk)^2, van Rooij, Wojtaszczyk. *FOCS'11*]

   - They introduce a DP technique called Cut&Count.
     (It relies on a probabilistic result called the Isolation Lemma.)
   - Can these algorithms be derandomized?

Some recent results

1. For an FPT problem, is it always possible to obtain algorithms with running time $2^{\mathcal{O}(\mathbf{tw})} \cdot n^{\mathcal{O}(1)}$?

   [Lokshtanov, Marx, Saurabh. *SODA'11*]
   If 3SAT cannot be solved in time $2^{o(n)}$, then DISJOINT PATHS cannot be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ in general graphs.

   - HAMILTONIAN PATH, FVS, CONNECTED VERTEX COVER, ...
     Is $2^{\mathcal{O}(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ best possible?

2. Randomized algorithms for connected packing-encodable problems in general graphs in time $2^{\mathcal{O}(\mathbf{tw})} \cdot n^{\mathcal{O}(1)}$.

   [Cygan, Nederlof, (Pilipczuk)^2, van Rooij, Wojtaszczyk. *FOCS'11*]

   - They introduce a DP technique called Cut&Count.
     (It relies on a probabilistic result called the Isolation Lemma.)
   - Can these algorithms be derandomized?

Some recent results

1. For an FPT problem, is it always possible to obtain algorithms with running time $2^{\mathcal{O}(\mathbf{tw})} \cdot n^{\mathcal{O}(1)}$?

   [Lokshtanov, Marx, Saurabh. *SODA'11*]
   If 3SAT cannot be solved in time $2^{o(n)}$, then DISJOINT PATHS cannot be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ in general graphs.

   - HAMILTONIAN PATH, FVS, CONNECTED VERTEX COVER, ...
     Is $2^{\mathcal{O}(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ best possible?

2. Randomized algorithms for connected packing-encodable problems in general graphs in time $2^{\mathcal{O}(\mathbf{tw})} \cdot n^{\mathcal{O}(1)}$.

   [Cygan, Nederlof, (Pilipczuk)^2, van Rooij, Wojtaszczyk. *FOCS'11*]

   - They introduce a DP technique called Cut&Count.
     (It relies on a probabilistic result called the Isolation Lemma.)
   - Can these algorithms be derandomized?

Some recent results

1. For an FPT problem, is it always possible to obtain algorithms with running time $2^{\mathcal{O}(\mathbf{tw})} \cdot n^{\mathcal{O}(1)}$?

   [Lokshtanov, Marx, Saurabh. *SODA'11*]
   If 3SAT cannot be solved in time $2^{o(n)}$, then DISJOINT PATHS cannot be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ in general graphs.

   - HAMILTONIAN PATH, FVS, CONNECTED VERTEX COVER, ...
     Is $2^{\mathcal{O}(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ best possible?

2. Randomized algorithms for connected packing-encodable problems in general graphs in time $2^{\mathcal{O}(\mathbf{tw})} \cdot n^{\mathcal{O}(1)}$.

   [Cygan, Nederlof, (Pilipczuk)^2, van Rooij, Wojtaszczyk. *FOCS'11*]

   - They introduce a DP technique called Cut&Count.
     (It relies on a probabilistic result called the Isolation Lemma.)
   - Can these algorithms be derandomized?

Some recent results

1. For an FPT problem, is it always possible to obtain algorithms with running time $2^{\mathcal{O}(\mathbf{tw})} \cdot n^{\mathcal{O}(1)}$?

   [Lokshtanov, Marx, Saurabh. *SODA'11*]
   If 3SAT cannot be solved in time $2^{o(n)}$, then DISJOINT PATHS cannot be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ in general graphs.

   - HAMILTONIAN PATH, FVS, CONNECTED VERTEX COVER, ...
     Is $2^{\mathcal{O}(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ best possible?

2. Randomized algorithms for connected packing-encodable problems in general graphs in time $2^{\mathcal{O}(\mathbf{tw})} \cdot n^{\mathcal{O}(1)}$.

   [Cygan, Nederlof, (Pilipczuk)^2, van Rooij, Wojtaszczyk. *FOCS'11*]

   - They introduce a DP technique called Cut&Count.
     (It relies on a probabilistic result called the Isolation Lemma.)
   - Can these algorithms be derandomized?

Gràcies!

CATALONIA, THE NEXT STATE IN EUROPE