Dynamic programming for graphs on surfaces

Ignasi Sau
Postdoc at Department of Computer Science of the Technion

Joint work with:

Juanjo Rué
Postdoc at École Polytechnique, Paris, France

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

[short version to appear in **ICALP'10**]

Outline

1. Some basic definitions
2. Motivation and previous work
3. Graphs on surfaces
4. Main ideas of our approach
5. Conclusions and further research

Outline

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:

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

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 $\mathbf{mid}(e) = V(A_e) \cap V(B_e)$.
- 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 $\mathbf{mid}(e) = V(A_e) \cap V(B_e)$.
- 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:

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

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.

- Barometer of intractability:

$$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq XP$$

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.

- Barometer of intractability:

$$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq XP$$

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.

- Barometer of intractability:

$$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq XP$$

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.

- Barometer of intractability:

$$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq XP$$

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.

- Barometer of intractability:

$$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq XP$$

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.

- Barometer of intractability:

$$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq XP$$

Outline

FPT and single-exponential algorithms

- Courcelle's theorem (1988):

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

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

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

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

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

FPT and single-exponential algorithms

- Courcelle's theorem (1988):

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

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

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

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

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

FPT and single-exponential algorithms

- Courcelle's theorem (1988):

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

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

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

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

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

FPT and single-exponential algorithms

- Courcelle's theorem (1988):

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

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

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

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

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

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.

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 $\textbf{mid}(e)$.
- The size of the tables reflects the dependence on $k = |\textbf{mid}(e)|$ 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.

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, 3-COLORING.
   The size of the tables is trivially 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 $k^{\Theta(k)} = 2^{\Theta(k \log k)}$...

   Done for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. *ESA'05*];
   Done 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, STEINER TREE.
   Again, # of packings in a set of k elements is $2^{\Theta(k \log k)}$.

   None of the current techniques seems to fit this class of **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, 3-COLORING.
   The size of the tables is trivially 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 $k^{\Theta(k)} = 2^{\Theta(k \log k)}$...

   Done for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. *ESA'05*];
   Done 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, STEINER TREE.
   Again, # of packings in a set of k elements is $2^{\Theta(k \log k)}$.

   None of the current techniques seems to fit this class of **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, 3-COLORING.
   The size of the tables is trivially 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 $k^{\Theta(k)} = 2^{\Theta(k \log k)}$...

   Done for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. *ESA'05*];
   Done 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, STEINER TREE.
   Again, $\#$ of packings in a set of k elements is $2^{\Theta(k \log k)}$.

   None of the current techniques seems to fit this class of **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, 3-COLORING.
   The size of the tables is trivially 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 $k^{\Theta(k)} = 2^{\Theta(k \log k)}$...
   Done for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. *ESA'05*];
   Done 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, STEINER TREE.
   Again, # of packings in a set of k elements is $2^{\Theta(k \log k)}$.

None of the current techniques seems to fit this class of **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, 3-COLORING.
   The size of the tables is trivially 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 $k^{\Theta(k)} = 2^{\Theta(k \log k)}$...

   Done for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. *ESA'05*];
   Done 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, STEINER TREE.
   Again, $\#$ of packings in a set of k elements is $2^{\Theta(k \log k)}$.

None of the current techniques seems to fit this class of **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, 3-COLORING.
   The size of the tables is trivially 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 $k^{\Theta(k)} = 2^{\Theta(k \log k)}$...

   Done for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. *ESA'05*];
   Done 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, STEINER TREE.
   Again, # of packings in a set of k elements is $2^{\Theta(k \log k)}$.

None of the current techniques seems to fit this class of **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, 3-COLORING.
   The size of the tables is trivially 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 $k^{\Theta(k)} = 2^{\Theta(k \log k)}$...

   Done for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. *ESA'05*];
   Done 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, STEINER TREE.
   Again, # of packings in a set of k elements is $2^{\Theta(k \log k)}$.

   None of the current techniques seems to fit this class of **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, 3-COLORING.
   The size of the tables is trivially 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 $k^{\Theta(k)} = 2^{\Theta(k \log k)}$...

   Done for planar graphs [Dorn, Penninkx, Bodlaender, Fomin. *ESA'05*];
   Done 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, STEINER TREE.
   Again, $\#$ of packings in a set of k elements is $2^{\Theta(k \log k)}$.

None of the current techniques seems to fit this class of **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, 3-COLORING.
   The size of the tables is trivially 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 $k^{\Theta(k)} = 2^{\Theta(k \log k)}$...

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

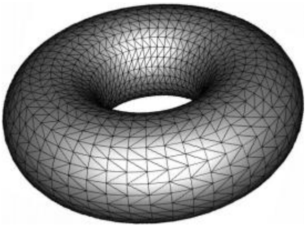
3. *Connected packing* of vertices of **mid**(e) into subsets of arbitrary size.
   Examples: CONNECTED VERTEX COVER, STEINER TREE.
   Again, $\#$ of packings in a set of k elements is $2^{\Theta(k \log k)}$.

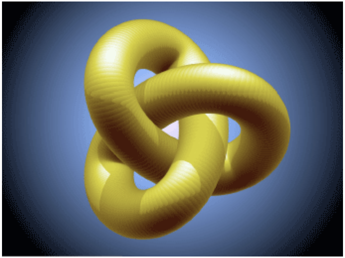
   None of the current techniques seems to fit this class of **packing-encodable** problems...

Outline

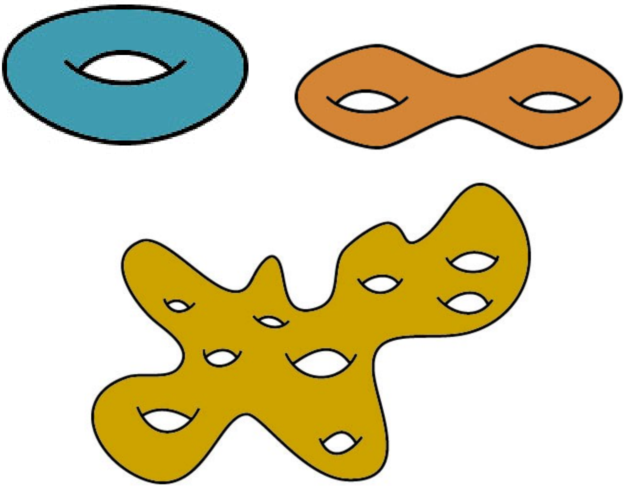
Surfaces

- Surface: connected compact 2-manifold.

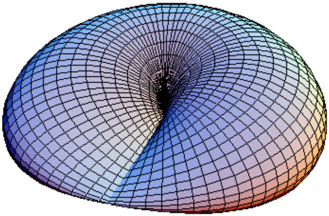


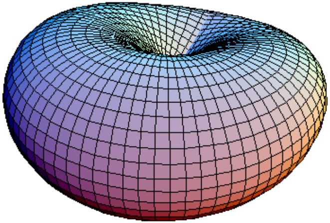


Handles



Cross-caps





Genus of a surface

- The 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 $\mathbf{eg}(\mathbb{P}_h) = h$.

Genus of a surface

- The 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 $\textbf{eg}(\mathbb{T}_g) = 2g$.

- 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 $\textbf{eg}(\mathbb{P}_h) = h$.

Genus of a surface

- The 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 $\mathbf{eg}(\mathbb{P}_h) = h$.

Graphs on surfaces

- An embedding of a graph G on a surface Σ is a drawing of G on Σ without edge crossings.
- An embedding defines vertices, edges, and faces.
- 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

- An embedding of a graph G on a surface Σ is a drawing of G on Σ without edge crossings.
- An embedding defines vertices, edges, and faces.
- The Euler genus of a graph G, $\mathbf{eg}(G)$, is the least Euler genus of the surfaces in which G can be embedded.

Nooses

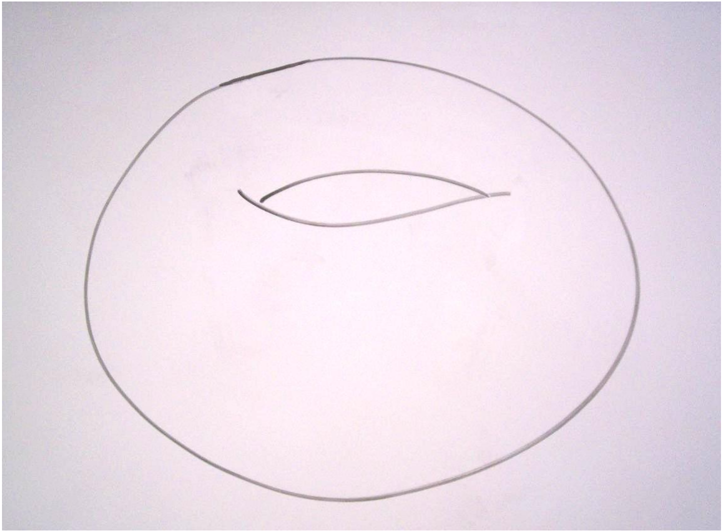
Let G be a graph embedded in a surface Σ. A noose is a subset of Σ homeomorphic to $\mathbb{S}^1$ that meets G only at vertices.

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.

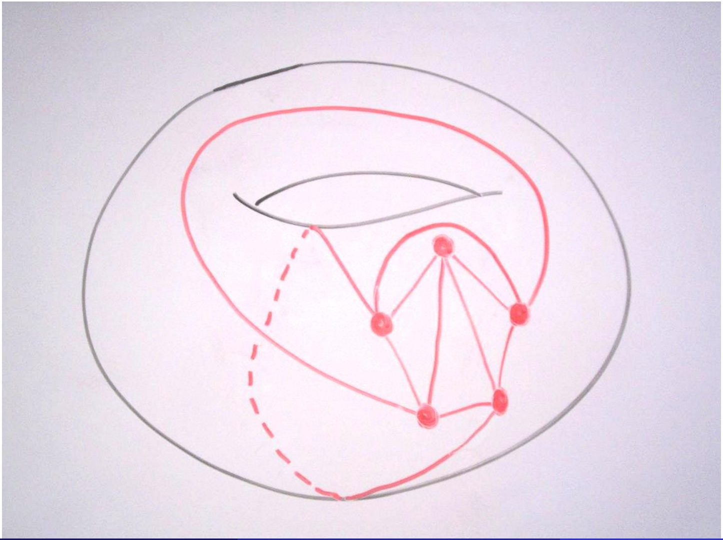
Nooses

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.



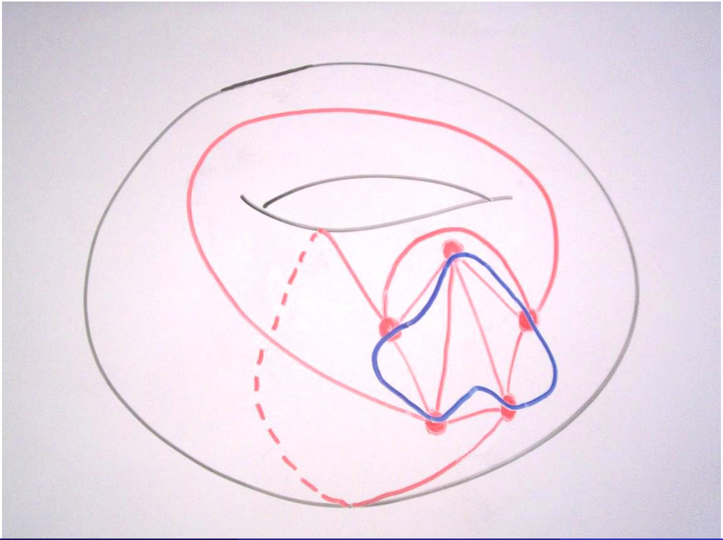
Nooses

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.



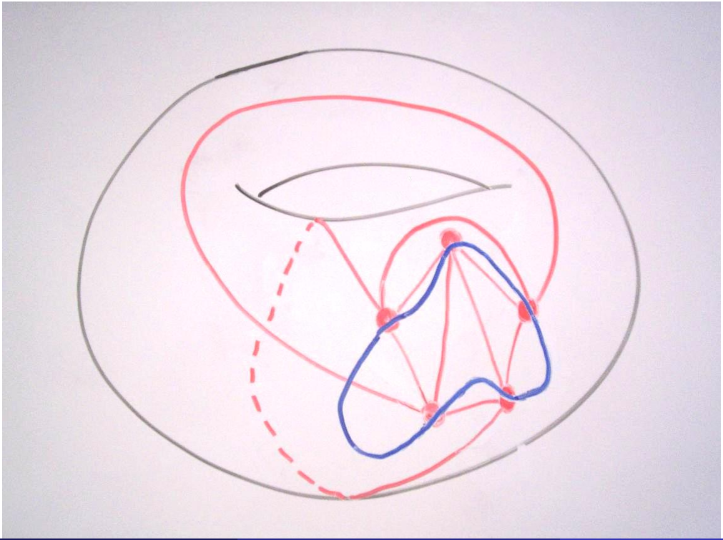
Nooses

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.



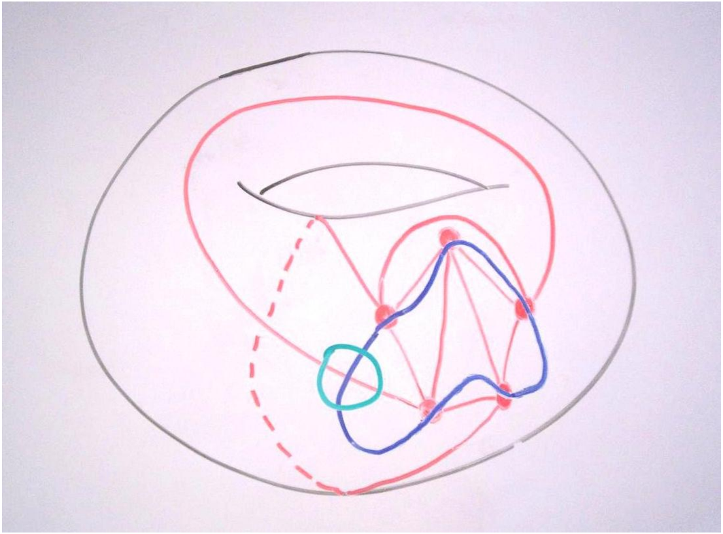
Nooses

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.



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.



Outline

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 we can 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 we can 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 we can 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:

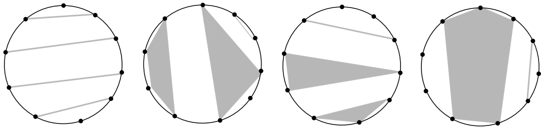
$$\text{CN}(k) = \frac{1}{k+1}\binom{2k}{k} \sim \frac{4^k}{\sqrt{\pi}k^{3/2}} \approx 4^k.$$

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 we can 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:

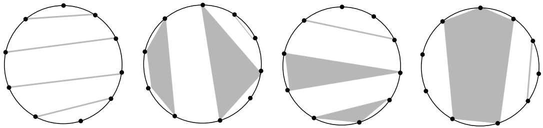
$$\text{CN}(k) = \frac{1}{k+1}\binom{2k}{k} \sim \frac{4^k}{\sqrt{\pi}k^{3/2}} \approx 4^k.$$

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 we can 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:

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

From sphere to surface cut decompositions

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 heavily on each **particular** problem.
  - ★ **Bad** dependence of the running time on the **genus** of the surface.
  - ★ Cannot be applied to **packing-encodable** problems.

From sphere to surface cut decompositions

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 heavily on each **particular** problem.
  - ★ **Bad** dependence of the running time on the **genus** of the surface.
  - ★ Cannot be applied to **packing-encodable** problems.

From sphere to surface cut decompositions

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 heavily on each **particular** problem.
  - ★ **Bad** dependence of the running time on the **genus** of the surface.
  - ★ Cannot be applied to **packing-encodable** problems.

From sphere to surface cut decompositions (II)

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 **packing-encodable** problems, and with **better genus** dependence.

From sphere to surface cut decompositions (II)

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 **packing-encodable** problems, and with **better genus** dependence.

From sphere to surface cut decompositions (II)

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 **packing-encodable** problems, and with **better genus** dependence.

From sphere to surface cut decompositions (II)

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 **packing-encodable** problems, and with **better genus** dependence.

Let G be a graph embedded in a surface Σ, with **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.

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.

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

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

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 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 $2^{\mathcal{O}(k)} \cdot k^{\mathcal{O}(\mathbf{g})} \cdot \mathbf{g}^{\mathcal{O}(\mathbf{g})}$.

This fact is proved using topological graph theory and **analytic combinatorics**, generalizing Catalan structures to arbitrary surfaces.

Main results

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

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

This fact is proved using topological graph theory and **analytic combinatorics**, generalizing Catalan structures to arbitrary surfaces.

Main results

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

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

This fact is proved using topological graph theory and **analytic combinatorics**, generalizing Catalan structures to arbitrary surfaces.

Main results

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

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

This fact is proved using topological graph theory and **analytic combinatorics**, generalizing Catalan structures to arbitrary surfaces.

Main results

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

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

This fact is proved using topological graph theory and **analytic combinatorics**, generalizing Catalan structures to arbitrary surfaces.

Outline

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

Further research

1. 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*]

     Not *really* single-exponential: $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$.
     ($h = |V(H)|$, $k = \mathbf{bw}(G)$, $n = |V(G)|$)

   - Single-exponential algorithm for a planar host graph.
     [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?

2. Can this framework be extended to **more general graphs**?

   Ongoing work: minor-free graphs...

Further research

1. 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*]

     Not *really* single-exponential: $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$.
     ($h = |V(H)|$, $k = \mathbf{bw}(G)$, $n = |V(G)|$)

   - Single-exponential algorithm for a planar host graph.
     [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?

2. Can this framework be extended to **more general graphs**?

   Ongoing work: minor-free graphs...

Further research

1. 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*]

     Not *really* single-exponential: $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$.
     ($h = |V(H)|$, $k = \mathbf{bw}(G)$, $n = |V(G)|$)

   - Single-exponential algorithm for a planar host graph.
     [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?

2. Can this framework be extended to **more general graphs**?

   Ongoing work: minor-free graphs...

Further research

1. 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*]

     Not *really* single-exponential: $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$.
     ($h = |V(H)|$, $k = \mathbf{bw}(G)$, $n = |V(G)|$)

   - Single-exponential algorithm for a planar host graph.
     [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?

2. Can this framework be extended to **more general graphs**?

   Ongoing work: minor-free graphs...

Further research

1. 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*]

     Not *really* single-exponential: $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$.
     ($h = |V(H)|$, $k = \mathbf{bw}(G)$, $n = |V(G)|$)

   - Single-exponential algorithm for a planar host graph.
     [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?

2. Can this framework be extended to **more general graphs**?

   Ongoing work: minor-free graphs...

Further research

1. 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*]

     Not *really* single-exponential: $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$.
     $(h = |V(H)|, k = \mathbf{bw}(G), n = |V(G)|)$

   - Single-exponential algorithm for a planar host graph.
     [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?

2. Can this framework be extended to **more general graphs**?

   Ongoing work: minor-free graphs...

Further research

1. 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*]

     Not *really* single-exponential: $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$.
     ($h = |V(H)|$, $k = \mathbf{bw}(G)$, $n = |V(G)|$)

   - Single-exponential algorithm for a planar host graph.
     [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?

2. Can this framework be extended to **more general graphs**?

   Ongoing work: minor-free graphs...

Further research

1. 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*]

     Not *really* single-exponential: $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$.
     ($h = |V(H)|$, $k = \mathbf{bw}(G)$, $n = |V(G)|$)

   - Single-exponential algorithm for a planar host graph.
     [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?

2. Can this framework be extended to **more general graphs**?

   Ongoing work: minor-free graphs...

Gràcies!