Programmation dynamique dans les graphes peu denses

Ignasi Sau

Equipe AlGCo, LIRMM, CNRS

Journée Scientifique du LIRMM

19 juin 2014

Outline

Next section is...

1. Introduction
2. Treewidth and dynamic programming
3. Dynamic programming on sparse graphs
4. Generalizations and some recent results

Basic idea of dynamic programming

- According to WIKIPEDIA:

  "Dynamic programming is a method for solving complex problems by breaking them down into simpler subproblems."

- Roughly speaking, it is a clever brute force search.

- The idea is to recursively combine previously computed partial solutions of smaller instances.

- In this talk, we will focus exclusively on graphs.

Basic idea of dynamic programming

- According to WIKIPEDIA:

  "Dynamic programming is a method for solving complex problems by breaking them down into simpler subproblems."

- Roughly speaking, it is a clever brute force search.

- The idea is to recursively combine previously computed partial solutions of smaller instances.

- In this talk, we will focus exclusively on graphs.

Basic idea of dynamic programming

- According to WIKIPEDIA:

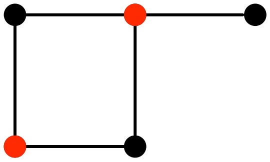
  “Dynamic programming is a method for solving complex problems by breaking them down into simpler subproblems.”

- Roughly speaking, it is a clever brute force search.

- The idea is to recursively combine previously computed partial solutions of smaller instances.

- In this talk, we will focus exclusively on $\boxed{\text{graphs}}$.

Independent set in a graph: set of vertices pairwise non-adjacent.

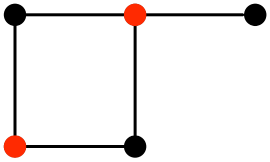


WEIGHTED INDEPENDENT SET

Input: A graph $G = (V, E)$ and a weight function $w : V \to \mathbb{N}$.
Output: An independent set of G of maximum weight.

Example: MAXIMUM INDEPENDENT SET

Independent set in a graph: set of vertices pairwise non-adjacent.

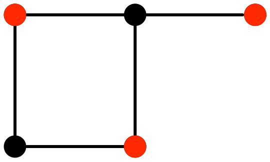


WEIGHTED INDEPENDENT SET

Input: A graph $G = (V, E)$ and a weight function $w : V \to \mathbb{N}$.
Output: An independent set of G of maximum weight.

Example: Maximum Independent Set

Independent set in a graph: set of vertices pairwise non-adjacent.

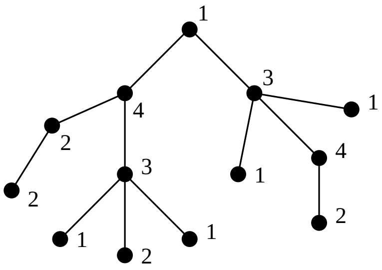


Weighted Independent Set

Input: A graph $G = (V, E)$ and a weight function $w : V \to \mathbb{N}$.
Output: An independent set of G of maximum weight.

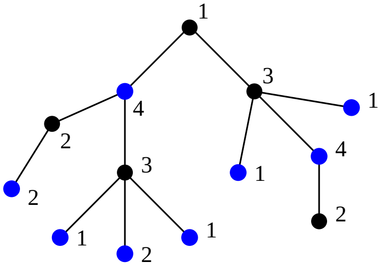
WEIGHTED INDEPENDENT SET in trees via DP

Merci Christophe !

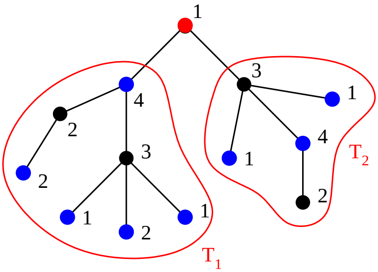


WEIGHTED INDEPENDENT SET in trees via DP

Merci Christophe !



Merci Christophe !

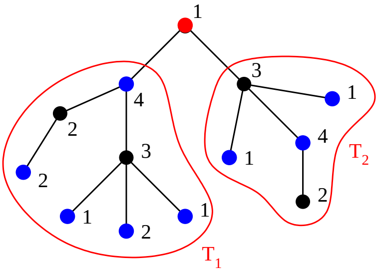


Remarks:

1. Every vertex in a tree is a separator.
2. The union of independent sets in distinct connected components is an independent set.

WEIGHTED INDEPENDENT SET in trees via DP

Merci Christophe !

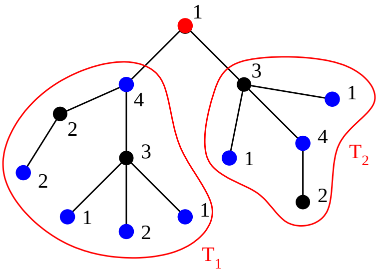


Let x be the root of T and $x_1, \ldots, x_\ell$ their children:

- $wIS(T, x) \rightarrow$ max. independent set in T containing x.
- $wIS(T, \overline{x}) \rightarrow$ max. independent set in T not containing x.

WEIGHTED INDEPENDENT SET in trees via DP

Merci Christophe !



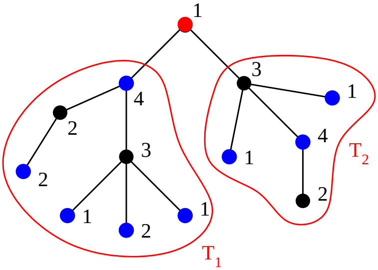
Let x be the root of T and $x_1, \ldots, x_\ell$ their children:

- $wIS(T, x) \rightarrow$ max. independent set in T containing x.
- $wIS(T, \overline{x}) \rightarrow$ max. independent set in T not containing x.

$$\begin{cases} wIS(T, x) = w(x) + \sum_{i \in \{1,\ldots,\ell\}} wIS(T, \overline{x_i}) \\ \end{cases}$$

WEIGHTED INDEPENDENT SET in trees via DP

Merci Christophe !



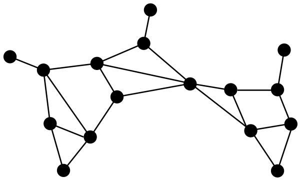
Let x be the root of T and $x_1, \ldots, x_\ell$ their children:

- $wIS(T,x) \rightarrow$ max. independent set in T containing x.
- $wIS(T,\overline{x}) \rightarrow$ max. independent set in T not containing x.

$$\begin{cases} wIS(T,x) &= w(x) + \sum_{i \in \{1,\ldots,\ell\}} wIS(T,\overline{x_i}) \\ wIS(T,\overline{x}) &= \sum_{i \in \{1,\ldots,\ell\}} \max\{wIS(T,x_i), wIS(T,\overline{x_i})\} \end{cases}$$

Next section is...

$\boxed{\text{Idea}}$ To measure the topological resemblance of a graph to a tree.

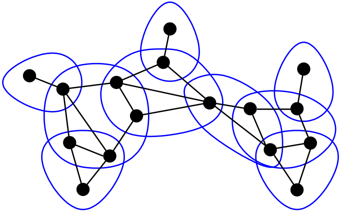


A tree decomposition of a graph $G = (V, E)$ is a pair $(T, \{X_t : t \in V(T)\})$, with T a tree and $\forall t \in T$, $X_t \subseteq V$ such that

- $\boxed{\text{covering of vertices}}$ $\forall x \in V, \exists t \in V(T)$ such that $x \in X_t$.
- $\boxed{\text{covering of edges}}$ $\forall \{x, y\} \in E, \exists t \in T$ such that $x, y \in X_t$.
- $\boxed{\text{consistence}}$ $\forall x \in V$, the set of "bags" containing x defines a connected subtree of T.

Tree decompositions

$\boxed{\text{Idea}}$ To measure the topological resemblance of a graph to a tree.

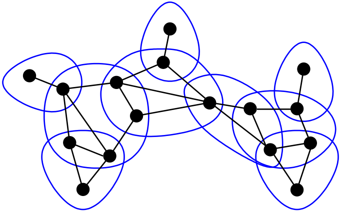


A tree decomposition of a graph $G = (V, E)$ is a pair $(T, \{X_t : t \in V(T)\})$, with T a tree and $\forall t \in T$, $X_t \subseteq V$ such that

- $\boxed{\text{covering of vertices}}$ $\forall x \in V, \exists t \in V(T)$ such that $x \in X_t$.
- $\boxed{\text{covering of edges}}$ $\forall \{x, y\} \in E, \exists t \in T$ such that $x, y \in X_t$.
- $\boxed{\text{consistence}}$ $\forall x \in V$, the set of "bags" containing x defines a connected subtree of T.

Tree decompositions

$\boxed{\text{Idea}}$ To measure the topological resemblance of a graph to a tree.

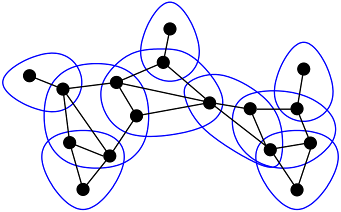


A tree decomposition of a graph $G = (V, E)$ is a pair $(T, \{X_t : t \in V(T)\})$, with T a tree and $\forall t \in T$, $X_t \subseteq V$ such that

- $\boxed{\text{covering of vertices}}$ $\forall x \in V, \exists t \in V(T)$ such that $x \in X_t$.
- $\boxed{\text{covering of edges}}$ $\forall \{x, y\} \in E, \exists t \in T$ such that $x, y \in X_t$.
- $\boxed{\text{consistence}}$ $\forall x \in V$, the set of "bags" containing x defines a connected subtree of T.

Tree decompositions

Idea To measure the topological resemblance of a graph to a tree.

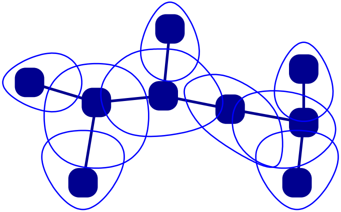


A tree decomposition of a graph $G = (V, E)$ is a pair $(T, \{X_t : t \in V(T)\})$, with T a tree and $\forall t \in T$, $X_t \subseteq V$ such that

- covering of vertices $\forall x \in V, \exists t \in V(T)$ such that $x \in X_t$.
- covering of edges $\forall \{x, y\} \in E, \exists t \in T$ such that $x, y \in X_t$.
- consistence $\forall x \in V$, the set of "bags" containing x defines a connected subtree of T.

Tree decompositions

$\boxed{\text{Idea}}$ To measure the topological resemblance of a graph to a tree.



A tree decomposition of a graph $G = (V, E)$ is a pair $(T, \{X_t : t \in V(T)\})$, with T a tree and $\forall t \in T$, $X_t \subseteq V$ such that

- $\boxed{\text{covering of vertices}}$ $\forall x \in V, \exists t \in V(T)$ such that $x \in X_t$.
- $\boxed{\text{covering of edges}}$ $\forall \{x, y\} \in E, \exists t \in T$ such that $x, y \in X_t$.
- $\boxed{\text{consistence}}$ $\forall x \in V$, the set of "bags" containing x defines a connected subtree of T.

Treewidth

- The width of a tree decomposition $\mathcal{T}_G = (T, \{X_t\})$ of G is

$$width(\mathcal{T}_G) = \max_{t \in T} |X_t| - 1$$

- The treewidth of a graph G is

$$tw(G) = \min_{\mathcal{T}_G} width(\mathcal{T}_G)$$

Idea

The smaller the treewidth of a graph, the more it resembles to a tree (if G is a tree, then $tw(G) = 1$).

Observation

Let $\mathcal{T}_G = (T, \{X_t : t \in V(T)\})$ be a tree decomposition of G.

- Every bag X_t is a separator of G.
- This makes tree decompositions a very suitable object for dynamic programming.

Treewidth

- The width of a tree decomposition $\mathcal{T}_G = (T, \{X_t\})$ of G is

$$width(\mathcal{T}_G) = \max_{t \in T} |X_t| - 1$$

- The treewidth of a graph G is

$$tw(G) = \min_{\mathcal{T}_G} width(\mathcal{T}_G)$$

Idea

The smaller the treewidth of a graph, the more it resembles to a tree (if G is a tree, then $tw(G) = 1$).

Observation

Let $\mathcal{T}_G = (T, \{X_t : t \in V(T)\})$ be a tree decomposition of G.

- Every bag X_t is a separator of G.
- This makes tree decompositions a very suitable object for dynamic programming.

Treewidth

- The width of a tree decomposition $\mathcal{T}_G = (T, \{X_t\})$ of G is

$$width(\mathcal{T}_G) = \max_{t \in T} |X_t| - 1$$

- The treewidth of a graph G is

$$tw(G) = \min_{\mathcal{T}_G} width(\mathcal{T}_G)$$

Idea

The smaller the treewidth of a graph, the more it resembles to a tree (if G is a tree, then $tw(G) = 1$).

Observation

Let $\mathcal{T}_G = (T, \{X_t : t \in V(T)\})$ be a tree decomposition of G.

- Every bag X_t is a separator of G.
- This makes tree decompositions a very suitable object for dynamic programming.

Treewidth

- The width of a tree decomposition $\mathcal{T}_G = (T, \{X_t\})$ of G is

$$width(\mathcal{T}_G) = \max_{t \in T} |X_t| - 1$$

- The treewidth of a graph G is

$$tw(G) = \min_{\mathcal{T}_G} width(\mathcal{T}_G)$$

Idea

The smaller the treewidth of a graph, the more it resembles to a tree (if G is a tree, then $tw(G) = 1$).

Observation

Let $\mathcal{T}_G = (T, \{X_t : t \in V(T)\})$ be a tree decomposition of G.

- Every bag X_t is a separator of G.
- This makes tree decompositions a very suitable object for dynamic programming.

Treewidth

- The width of a tree decomposition $\mathcal{T}_G = (T, \{X_t\})$ of G is

$$width(\mathcal{T}_G) = \max_{t \in T} |X_t| - 1$$

- The treewidth of a graph G is

$$tw(G) = \min_{\mathcal{T}_G} width(\mathcal{T}_G)$$

Idea

The smaller the treewidth of a graph, the more it resembles to a tree (if G is a tree, then $tw(G) = 1$).

Observation

Let $\mathcal{T}_G = (T, \{X_t : t \in V(T)\})$ be a tree decomposition of G.

- Every bag X_t is a separator of G.
- This makes tree decompositions a very suitable object for dynamic programming.

INDEPENDENT SET on graphs of bounded treewidth

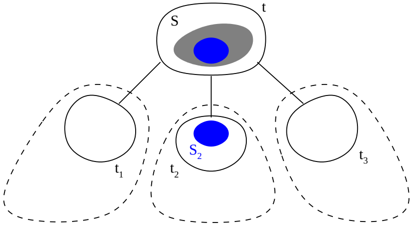
As in the case of trees, the problem can be solved via DP.

For graphs on n vertices and $tw \leq k$, this algorithm solves INDEPENDENT SET in time

$$\boxed{O(4^k \cdot k^2 \cdot n)}$$

Independent Set on graphs of bounded treewidth

As in the case of trees, the problem can be solved via DP.



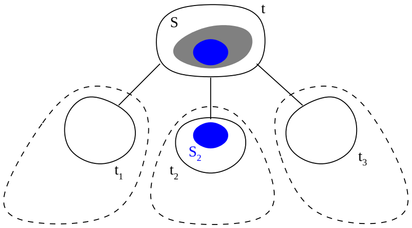
$$IS(S,t) = \left\{ \sum_{i \in [\ell]} \max \begin{array}{l} |S| + \\ \{IS(S_j^i, t_j) - |S_j| \ : \\ S_j^i \cap X_t = S_j \ \& \ S_j \subseteq S_j^i \text{ independent}\} \end{array} \right.$$

For graphs on n vertices and $tw \leq k$, this algorithm solves Independent Set in time

$$\boxed{O(4^k \cdot k^2 \cdot n)}$$

INDEPENDENT SET on graphs of bounded treewidth

As in the case of trees, the problem can be solved via DP.



$$IS(S,t) = \left\{ \sum_{i \in [\ell]} \max \quad \begin{array}{l} |S| + \\ \{IS(S_j^i, t_j) - |S_j| \ : \\ S_j^i \cap X_t = S_j \ \& \ S_j \subseteq S_j^i \text{ independent}\} \end{array} \right.$$

For graphs on n vertices and $tw \leq k$, this algorithm solves INDEPENDENT SET in time

$$\boxed{O(4^k \cdot k^2 \cdot n)}$$

Some words on parameterized complexity

$\boxed{\text{Idea}}$ given an NP-hard problem, fix a parameter k of the input to see if the problem gets more “tractable”.

Example: the size of a VERTEX COVER, or the TREEWIDTH.

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

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

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

Some words on parameterized complexity

$\boxed{\text{Idea}}$ given an NP-hard problem, fix a parameter k of the input to see if the problem gets more “tractable”.

Example: the size of a VERTEX COVER, or the TREEWIDTH.

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

$\boxed{f(k) \cdot n^{O(1)}}$ for some (computable) function f.

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

Algorithmic importance of treewidth

$\boxed{\text{Courcelle's theorem (1988)}}$

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

(In other words, problems expressible in MSOL are FPT when parameterized by the treewidth of the input graph.)

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

In fact, $f(k)$ **must** be an exponential tower whose height equals the number of alternate quantifiers in the MSOL formula that expresses the problem.

Algorithmic importance of treewidth

$\boxed{\text{Courcelle's theorem (1988)}}$

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

(In other words, problems expressible in MSOL are FPT when parameterized by the treewidth of the input graph.)

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

In fact, $f(k)$ **must** be an exponential tower whose height equals the number of alternate quantifiers in the MSOL formula that expresses the problem.

Algorithmic importance of treewidth

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 with **tw** $\leq k$.

(In other words, problems expressible in MSOL are FPT when parameterized by the treewidth of the input graph.)

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

In fact, $f(k)$ **must** be an exponential tower whose height equals the number of alternate quantifiers in the MSOL formula that expresses the problem.

Algorithmic importance of treewidth

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 with **tw** $\leq k$.

(In other words, problems expressible in MSOL are FPT when parameterized by the treewidth of the input graph.)

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

In fact, $f(k)$ **must** be an exponential tower whose height equals the number of alternate quantifiers in the MSOL formula that expresses the problem.

FPT single-exponential algorithms

- We would like to find functions $f(k)$ as small as possible that apply to as many problems as possible.

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

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

For many problems, such function $f(k)$ is best possible (under the ETH).

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

FPT single-exponential algorithms

- We would like to find functions $f(k)$ as small as possible that apply to as many problems as possible.
- A single-exponential parameterized algorithm is a FPT algo s.t.

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

For many problems, such function $f(k)$ is best possible (under the ETH).

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

FPT single-exponential algorithms

- We would like to find functions $f(k)$ as small as possible that apply to as many problems as possible.
- A single-exponential parameterized algorithm is a FPT algo s.t.

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

For many problems, such function $f(k)$ is best possible (under the ETH).

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

FPT single-exponential algorithms

- We would like to find functions $f(k)$ as small as possible that apply to as many problems as possible.
- A single-exponential parameterized algorithm is a FPT algo s.t.

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

For many problems, such function $f(k)$ is best possible (under the ETH).

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

Dynamic programming (DP) on tree decompositions

- Applied in a bottom-up fashion on a rooted tree 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 bag X_t.
- The size of the tables reflects the dependence on $|X_t| \leq k$ in the running time of the DP.
- The precise definition of the tables of the DP depends on each particular problem.

Dynamic programming (DP) on tree decompositions

- Applied in a bottom-up fashion on a rooted tree 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 bag X_t.
- The size of the tables reflects the dependence on $|X_t| \leq k$ 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 bag X_t of a tree decomposition?

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

2. A *connected pairing* of vertices of X_t.
   Examples: HAMILTONIAN CYCLE, LONGEST PATH, CYCLE PACKING.
   The # of pairings in a set of k elements is $\boxed{k^{\Theta(k)} = 2^{\Theta(k \log k)}}$.

A classification of graph optimization problems

How can we certificate a solution in a bag X_t of a tree decomposition?

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

2. A *connected pairing* of vertices of X_t.
   Examples: HAMILTONIAN CYCLE, LONGEST PATH, CYCLE PACKING.
   The # of pairings in a set of k elements is $\boxed{k^{\Theta(k)} = 2^{\Theta(k \log k)}}$.

A classification of graph optimization problems

How can we certificate a solution in a bag X_t of a tree decomposition?

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

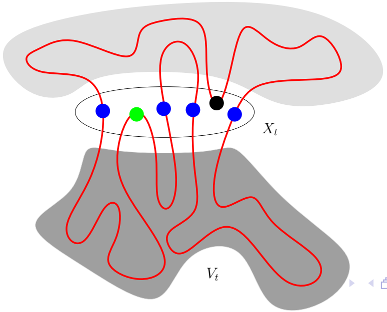
2. A *connected pairing* of vertices of X_t.
   Examples: HAMILTONIAN CYCLE, LONGEST PATH, CYCLE PACKING.
   The # of pairings in a set of k elements is $\boxed{k^{\Theta(k)} = 2^{\Theta(k \log k)}}$.

A classification of graph optimization problems

How can we certificate a solution in a bag X_t of a tree decomposition?

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

2. A *connected pairing* of vertices of X_t.
   Examples: HAMILTONIAN CYCLE, LONGEST PATH, CYCLE PACKING.
   The $\#$ of pairings in a set of k elements is $\boxed{k^{\Theta(k)} = 2^{\Theta(k \log k)}}$.

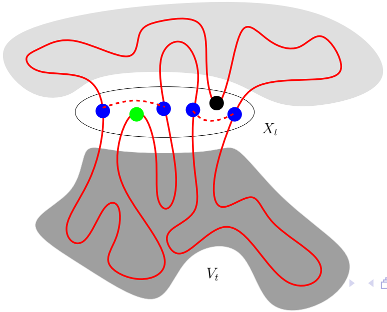


A classification of graph optimization problems

How can we certificate a solution in a bag X_t of a tree decomposition?

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

2. A *connected pairing* of vertices of X_t.
   Examples: HAMILTONIAN CYCLE, LONGEST PATH, CYCLE PACKING.
   The # of pairings in a set of k elements is $\boxed{k^{\Theta(k)} = 2^{\Theta(k \log k)}}$.



A classification of graph optimization problems

How can we certificate a solution in a bag X_t of a tree decomposition?

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

2. A *connected pairing* of vertices of X_t.
   Examples: HAMILTONIAN CYCLE, LONGEST PATH, CYCLE PACKING.
   The $\#$ of pairings in a set of k elements is $\boxed{k^{\Theta(k)} = 2^{\Theta(k \log k)}}$.

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

A classification of graph optimization problems

How can we certificate a solution in a bag X_t of a tree decomposition?

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

2. A *connected pairing* of vertices of X_t.
   Examples: HAMILTONIAN CYCLE, LONGEST PATH, CYCLE PACKING.
   The $\#$ of pairings in a set of k elements is $\boxed{k^{\Theta(k)} = 2^{\Theta(k \log k)}}$.

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

★ How can be improve the bound $\boxed{2^{O(k \log k)}}$ to $\boxed{2^{O(k)}}$?

Next section is...

Sparse graphs

- A family of graphs is sparse if it has a linear number of edges.

- **Archetypical example**: Planar graphs.
  By Euler's formula, if $G = (V, E)$ is a planar graph, then

$$|E| \leq 3 \cdot |V| - 6.$$

- Graphs that can be embedded on surfaces of bounded genus.

- Graphs that exclude a fixed graph H as a (topological) minor.

Sparse graphs

- A family of graphs is sparse if it has a linear number of edges.

- **Archetypical example**: Planar graphs.
  By Euler's formula, if $G = (V, E)$ is a planar graph, then

$$|E| \ \leq \ 3 \cdot |V| - 6.$$

- Graphs that can be embedded on surfaces of bounded genus.

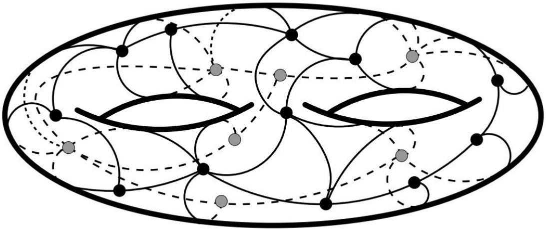
- Graphs that exclude a fixed graph H as a (topological) minor.

Sparse graphs

- A family of graphs is sparse if it has a linear number of edges.
- **Archetypical example**: Planar graphs.
  By Euler's formula, if $G = (V, E)$ is a planar graph, then

$$|E| \leq 3 \cdot |V| - 6.$$

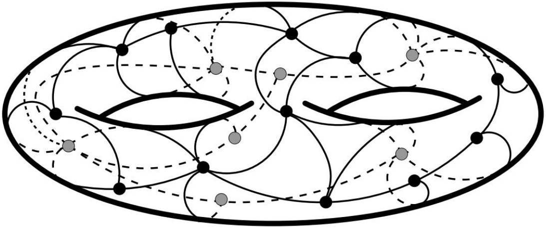
- Graphs that can be embedded on surfaces of bounded genus.



- Graphs that exclude a fixed graph H as a (topological) minor.

Sparse graphs

- A family of graphs is sparse if it has a linear number of edges.
- **Archetypical example**: Planar graphs.
  By Euler's formula, if $G = (V, E)$ is a planar graph, then
  $$|E| \leq 3 \cdot |V| - 6.$$
- Graphs that can be embedded on surfaces of bounded genus.



- Graphs that exclude a fixed graph H as a (topological) minor.

How sparsity helps for dynamic programming?

- We will consider a tree-decomposition of a sparse graph, and exploit the structure of the subgraph induced by the bags.

- More precisely, we will use the existence of tree decompositions of small width and with nice topological properties.

- These nice properties will $\boxed{\text{not}}$ change the DP algorithms, but the analysis of their running time.

How sparsity helps for dynamic programming?

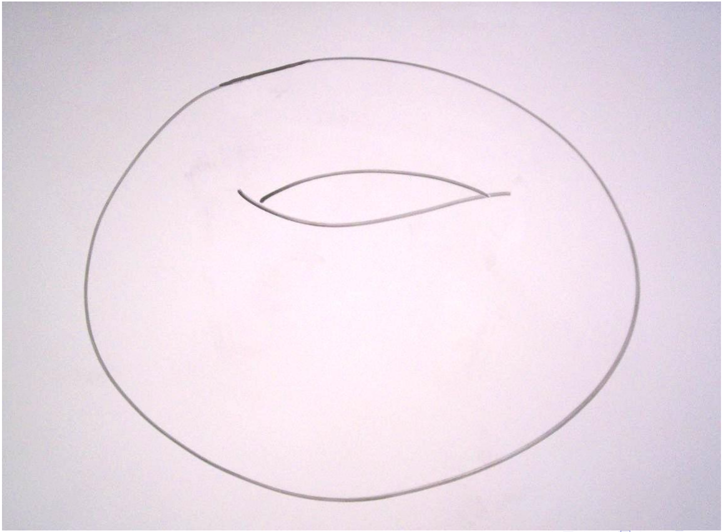
- We will consider a tree-decomposition of a sparse graph, and exploit the structure of the subgraph induced by the bags.
- More precisely, we will use the existence of tree decompositions of small width and with nice topological properties.
- These nice properties will $\boxed{\text{not}}$ change the DP algorithms, but the analysis of their running time.

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.

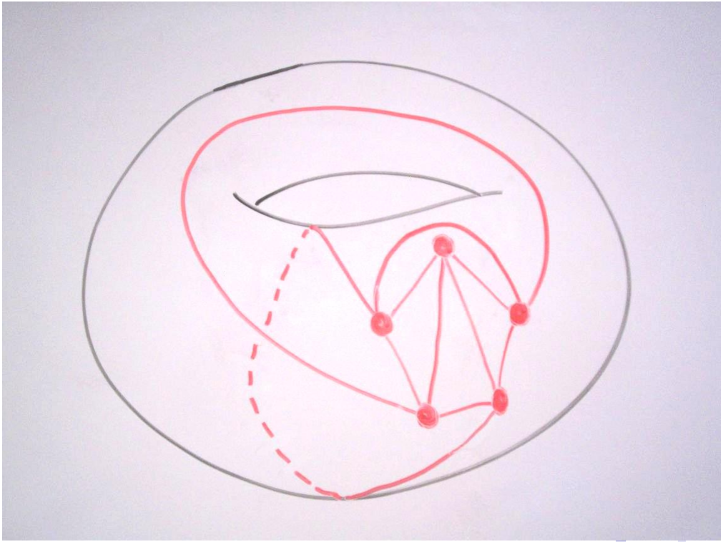
Nooses

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.



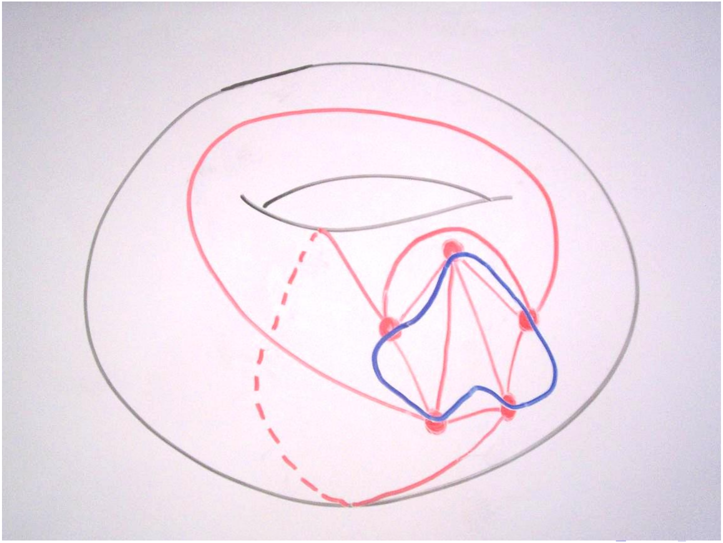
Nooses

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.



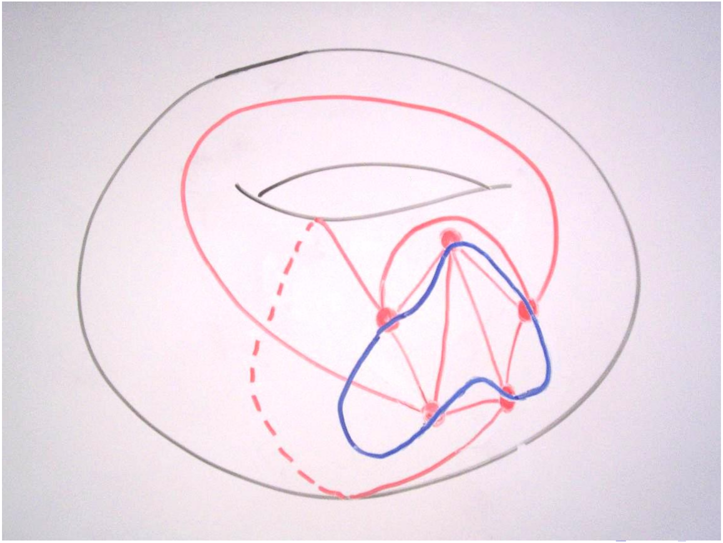
Nooses

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.



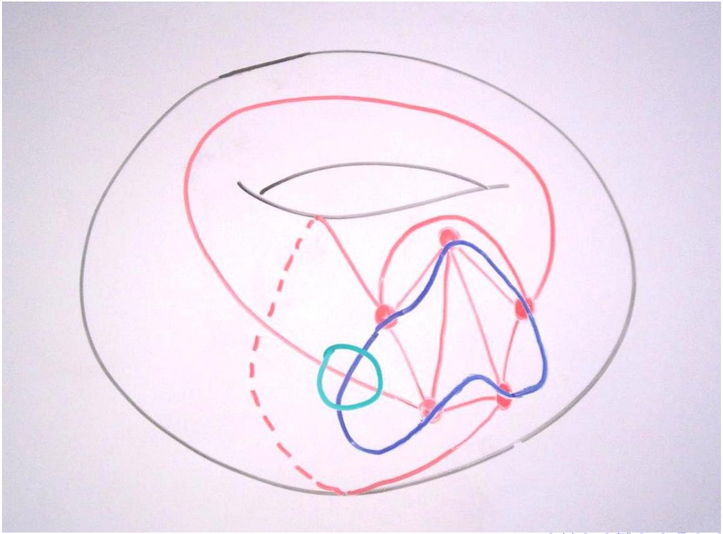
Nooses

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

- Let G be a planar graph. A sphere cut decomposition of G is a tree decomposition $(T, \{X_t : t \in V(T)\})$ of G such that the vertices in each bag X_t are situated around a noose in the plane.

(NB: several details are missing in this definition)

Theorem (Seymour and Thomas '94)

Every planar graph G has a sphere cut decomposition whose width equals $tw(G)$, and that can be computed in polynomial time.

- The size of the tables of a DP algorithm depends on how many ways a partial solution can intersect the vertices in a bag X_t.

Sphere cut decompositions

- Let G be a planar graph. A sphere cut decomposition of G is a tree decomposition $(T, \{X_t : t \in V(T)\})$ of G such that the vertices in each bag X_t are situated around a noose in the plane.

(NB: several details are missing in this definition)

Theorem (Seymour and Thomas '94)

Every planar graph G has a sphere cut decomposition whose width equals $\mathrm{tw}(G)$, and that can be computed in polynomial time.

- The size of the tables of a DP algorithm depends on how many ways a partial solution can intersect the vertices in a bag X_t.

Sphere cut decompositions

- Let G be a planar graph. A sphere cut decomposition of G is a tree decomposition $(T, \{X_t : t \in V(T)\})$ of G such that the vertices in each bag X_t are situated around a noose in the plane.

(NB: several details are missing in this definition)

Theorem (Seymour and Thomas '94)

Every planar graph G has a sphere cut decomposition whose width equals $tw(G)$, and that can be computed in polynomial time.

- The size of the tables of a DP algorithm depends on how many ways a partial solution can intersect the vertices in a bag X_t.

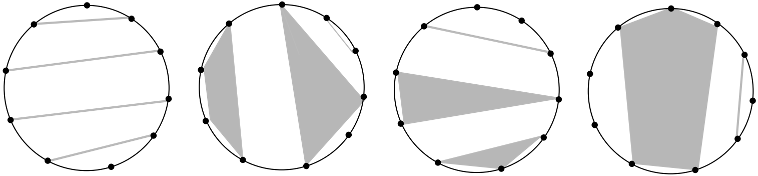
Sphere cut decompositions (2)

- Suppose we do DP on a sphere cut decomposition of width $\leq k$.
- 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 (2)

- Suppose we do DP on a sphere cut decomposition of width $\leq k$.
- 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?

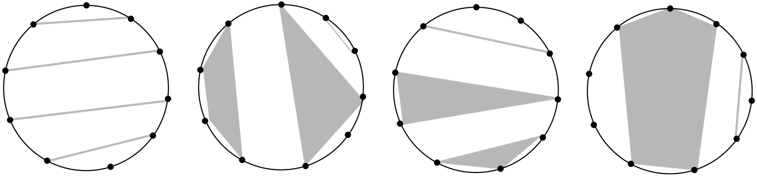


- Exactly the number of *non-crossing partitions* over k elements, which is given by the k-th Catalan number:

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

Sphere cut decompositions (2)

- Suppose we do DP on a sphere cut decomposition of width $\leq k$.
- 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:

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

How to use this framework?

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

This idea was first used in [Dorn, Penninkx, Bodlaender, Fomin '05]

How to use this framework?

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

This idea was first used in [Dorn, Penninkx, Bodlaender, Fomin '05]

How to use this framework?

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

This idea was first used in [Dorn, Penninkx, Bodlaender, Fomin '05]

How to use this framework?

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

This idea was first used in [Dorn, Penninkx, Bodlaender, Fomin '05]

How to use this framework?

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

This idea was first used in [Dorn, Penninkx, Bodlaender, Fomin '05]

Next section is...

1. Introduction
2. Treewidth and dynamic programming
3. Dynamic programming on sparse graphs
4. Generalizations and some recent results

Generalizations to other sparse graph classes

This idea has been generalized to other graph classes and problems:

- Graphs on surfaces:

  [Dorn, Fomin, Thilikos '06]
  [Rué, S., Thilikos '10]

- *H*-minor-free graphs:

  [Dorn, Fomin, Thilikos '08]
  [Rué, S., Thilikos '12]

Some recent results on general graphs

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

If 3-SAT cannot be solved in time $2^{o(n)}$, then DISJOINT PATHS cannot be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ on general graphs.

[Lokshtanov, Marx, Saurabh '11]

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

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

[Cygan, Nederlof, (Pilipczuk)^2, van Rooij, Wojtaszczyk '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 on general graphs

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

If 3-SAT cannot be solved in time $2^{o(n)}$, then DISJOINT PATHS cannot be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ on general graphs.

[Lokshtanov, Marx, Saurabh '11]

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

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

[Cygan, Nederlof, (Pilipczuk)^2, van Rooij, Wojtaszczyk '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 on general graphs

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

If 3-SAT cannot be solved in time $2^{o(n)}$, then DISJOINT PATHS cannot be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ on general graphs.

[Lokshtanov, Marx, Saurabh '11]

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

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

[Cygan, Nederlof, (Pilipczuk)^2, van Rooij, Wojtaszczyk '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 on general graphs

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

If 3-SAT cannot be solved in time $2^{o(n)}$, then DISJOINT PATHS cannot be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ on general graphs.

[Lokshtanov, Marx, Saurabh '11]

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

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

[Cygan, Nederlof, (Pilipczuk)^2, van Rooij, Wojtaszczyk '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 on general graphs

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

If 3-SAT cannot be solved in time $2^{o(n)}$, then DISJOINT PATHS cannot be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ on general graphs.

[Lokshtanov, Marx, Saurabh '11]

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

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

[Cygan, Nederlof, (Pilipczuk)^2, van Rooij, Wojtaszczyk '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 on general graphs

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

If 3-SAT cannot be solved in time $2^{o(n)}$, then DISJOINT PATHS cannot be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{\mathcal{O}(1)}$ on general graphs.

[Lokshtanov, Marx, Saurabh '11]

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

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

[Cygan, Nederlof, (Pilipczuk)^2, van Rooij, Wojtaszczyk '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 on general graphs (2)

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

[Bodlaender, Cygan, Kratsch, Nederlof '14]

- The approach is based on linear algebra.

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

[Fomin, Lokshtanov, Saurabh '14]

- The approach is based on matroids.

Some recent results on general graphs (2)

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

[Bodlaender, Cygan, Kratsch, Nederlof '14]

- The approach is based on linear algebra.

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

[Fomin, Lokshtanov, Saurabh '14]

- The approach is based on matroids.

Gràcies!

CATALONIA, THE NEXT STATE IN EUROPE