Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth

Michał Pilipczuk

Institute of Informatics, University of Warsaw, Poland

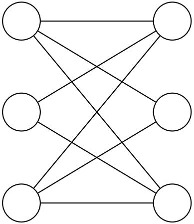
Workshop on Exact Algorithms and Lower Bounds, IIT Delhi,
December 14th, 2014

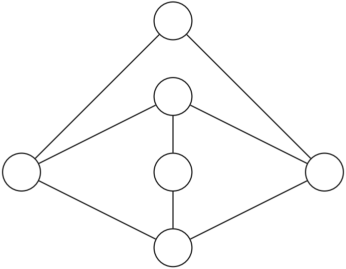
Source

- Joint work with:
  - Daniel Lokshtanov,
  - Marcin Pilipczuk, and
  - Saket Saurabh.
- Presented at FOCS 2014.
- Check out arxiv.org/abs/1404.0818

Graph Isomorphism

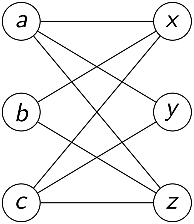
In the GRAPH ISOMORPHISM problem, given two graphs G_1 and G_2, we are to check if they are isomorphic.

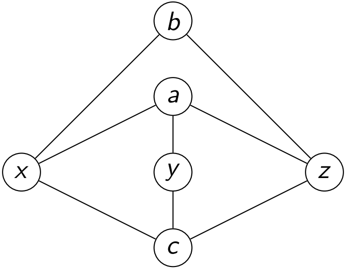




Graph Isomorphism

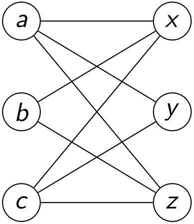
In the Graph Isomorphism problem, given two graphs G_1 and G_2, we are to check if they are isomorphic.

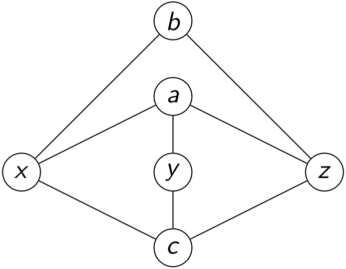




Graph Isomorphism

In the Graph Isomorphism problem, given two graphs G_1 and G_2, we are to check if they are isomorphic.





Graph Isomorphism

Input: Graphs G_1 and G_2
Question: Is there a bijection $\phi \colon V(G_1) \to V(G_2)$ s.t. $uv \in E(G_1)$ iff $\phi(u)\phi(v) \in E(G_2)$?

Graph Isomorphism

- Belongs **NP**, not known to be in **P**, but not **NP**-complete unless the PH collapses [Schöning'88].

Graph Isomorphism

- Belongs **NP**, not known to be in **P**, but not **NP**-complete unless the PH collapses [Schöning'88].
- Some believe it's an intermediate problem.

Graph Isomorphism

- Belongs **NP**, not known to be in **P**, but not **NP**-complete unless the PH collapses [Schöning'88].
- Some believe it's an intermediate problem.
- Many polynomial-time algorithms for special graph classes:
  - trees [Kelly'57]
  - planar graphs [Hopcroft-Wong'74]
  - interval graphs [Booth-Lueker'79]
  - permutation graphs [Colbourn'81]
  - bounded genus graphs [Miller'80], [Filotti-Mayer'80]
  - bounded degree graphs [Luks'82]
  - graphs with bounded eigenvalue multiplicity [Babai-Grigoryev-Mount'82]
  - bounded treewidth graphs [Bodlaender'90]
  - graphs excluding a fixed minor [Ponomarenko'91]
  - graphs excluding a fixed topological minor [Grohe-Marx'12]

Graph Isomorphism

- Belongs **NP**, not known to be in **P**, but not **NP**-complete unless the PH collapses [Schöning'88].
- Some believe it's an intermediate problem.
- Many polynomial-time algorithms for special graph classes:
  - trees [Kelly'57]
  - planar graphs [Hopcroft-Wong'74]
  - interval graphs [Booth-Lueker'79]
  - permutation graphs [Colbourn'81]
  - bounded genus graphs [Miller'80], [Filotti-Mayer'80]
  - bounded degree graphs [Luks'82]
  - graphs with bounded eigenvalue multiplicity [Babai-Grigoryev-Mount'82]
  - bounded treewidth graphs [Bodlaender'90]
  - graphs excluding a fixed minor [Ponomarenko'91]
  - graphs excluding a fixed topological minor [Grohe-Marx'12]
- In almost all the relevant cases above, these are **XP** algorithms.

- Root G_1 in r_1 and guess its image r_2.

Graph Isomorphism of trees

- Root G_1 in r_1 and guess its image r_2.
- Run a bottom-up dynamic programming with the following table:

  For $u \in V(G_1)$ and $v \in V(G_2)$, $T[u, v]$ is the answer to the question:
  Is the subtree rooted at u isomorphic to the subtree rooted at v?

Graph Isomorphism of trees

- Root G_1 in r_1 and guess its image r_2.
- Run a bottom-up dynamic programming with the following table:

  For $u \in V(G_1)$ and $v \in V(G_2)$, $T[u, v]$ is the answer to the question:
  Is the subtree rooted at u isomorphic to the subtree rooted at v?
- Fill $T[\cdot, \cdot]$ in a bottom-up manner. $T[r_1, r_2]$ is the answer.

Graph Isomorphism of trees

- Root G_1 in r_1 and guess its image r_2.
- Run a bottom-up dynamic programming with the following table:

  For $u \in V(G_1)$ and $v \in V(G_2)$, $T[u, v]$ is the answer to the question:
  Is the subtree rooted at u isomorphic to the subtree rooted at v?
- Fill $T[\cdot, \cdot]$ in a bottom-up manner. $T[r_1, r_2]$ is the answer.
- Computation at one step boils down to a matching problem.

Graph Isomorphism of trees

- Root G_1 in r_1 and guess its image r_2.
- Run a bottom-up dynamic programming with the following table:

  For $u \in V(G_1)$ and $v \in V(G_2)$, $T[u, v]$ is the answer to the question:
  Is the subtree rooted at u isomorphic to the subtree rooted at v?
- Fill $T[\cdot, \cdot]$ in a bottom-up manner. $T[r_1, r_2]$ is the answer.
- Computation at one step boils down to a matching problem.
- **Hard exercise**: Do it in linear time.

Planar Graph Isomorphism

Theorem (Hopcroft-Tarjan'73)

Given a graph G, one can in linear time compute its decomposition into 3-connected components. Moreover, the decomposition is isomorphism-invariant.

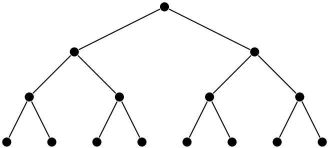
+

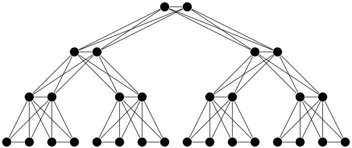
Theorem (Whitney)

A 3-connected planar graph has unique planar embedding.

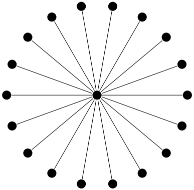
It is easy to compare *embedded* graphs.

Bounded treewidth graphs

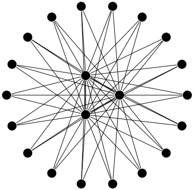

tree


treelike

Bounded treewidth graphs

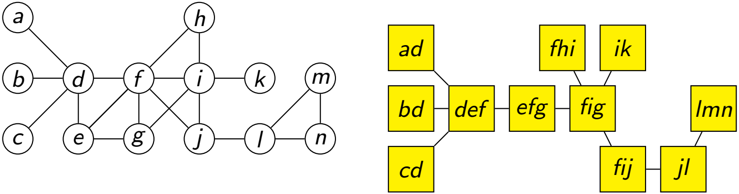


tree



treelike

Bounded treewidth graphs



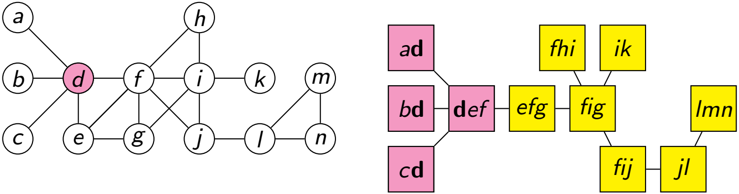
Definition

A *tree decomposition* of a graph G is a pair (T, β) where T is a tree and $\beta : V(T) \to 2^{V(G)}$ satisfying:

1. $\{t : v \in \beta(t)\}$ is nonempty and connected for every $v \in V(G)$;
2. for every $uv \in E(G)$ there exists $t \in V(T)$ such that $u, v \in \beta(t)$.

Width of the decomposition is $\max_{t \in V(T)} |\beta(t)| - 1$.
Treewidth of G is minimum possible width of tree decomposition of G.

Bounded treewidth graphs



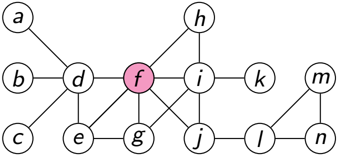
Definition

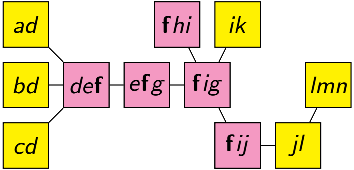
A *tree decomposition* of a graph G is a pair (T, β) where T is a tree and $\beta : V(T) \to 2^{V(G)}$ satisfying:

1. $\{t : v \in \beta(t)\}$ is nonempty and connected for every $v \in V(G)$;
2. for every $uv \in E(G)$ there exists $t \in V(T)$ such that $u, v \in \beta(t)$.

Width of the decomposition is $\max_{t \in V(T)} |\beta(t)| - 1$.
Treewidth of G is minimum possible width of tree decomposition of G.

Bounded treewidth graphs





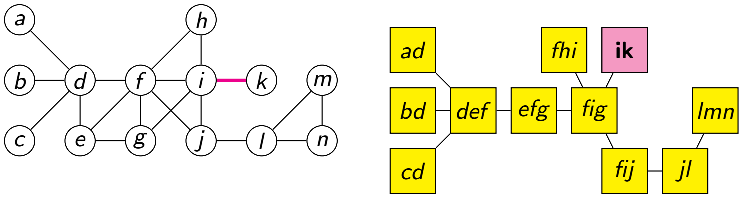
Definition

A *tree decomposition* of a graph G is a pair (T, β) where T is a tree and $\beta : V(T) \to 2^{V(G)}$ satisfying:

1. $\{t : v \in \beta(t)\}$ is nonempty and connected for every $v \in V(G)$;
2. for every $uv \in E(G)$ there exists $t \in V(T)$ such that $u, v \in \beta(t)$.

Width of the decomposition is $\max_{t \in V(T)} |\beta(t)| - 1$.
Treewidth of G is minimum possible width of tree decomposition of G.

Bounded treewidth graphs



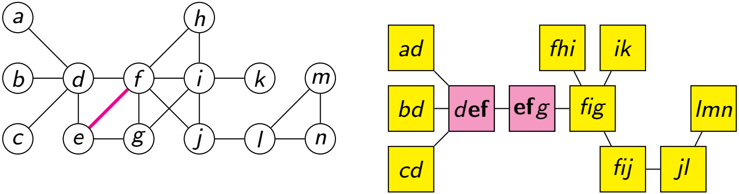
Definition

A *tree decomposition* of a graph G is a pair (T, β) where T is a tree and $\beta : V(T) \to 2^{V(G)}$ satisfying:

1. $\{t : v \in \beta(t)\}$ is nonempty and connected for every $v \in V(G)$;
2. for every $uv \in E(G)$ there exists $t \in V(T)$ such that $u, v \in \beta(t)$.

Width of the decomposition is $\max_{t \in V(T)} |\beta(t)| - 1$.
Treewidth of G is minimum possible width of tree decomposition of G.

Bounded treewidth graphs



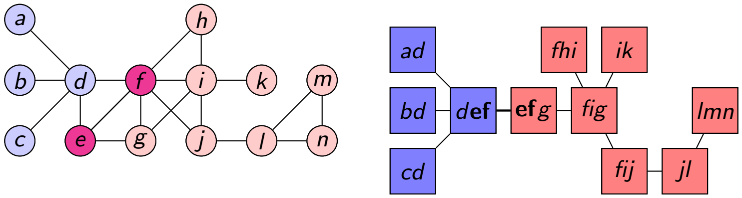
Definition

A *tree decomposition* of a graph G is a pair (T, β) where T is a tree and $\beta : V(T) \to 2^{V(G)}$ satisfying:

1. $\{t : v \in \beta(t)\}$ is nonempty and connected for every $v \in V(G)$;
2. for every $uv \in E(G)$ there exists $t \in V(T)$ such that $u, v \in \beta(t)$.

Width of the decomposition is $\max_{t \in V(T)} |\beta(t)| - 1$.
Treewidth of G is minimum possible width of tree decomposition of G.

Bounded treewidth graphs



Definition

A *tree decomposition* of a graph G is a pair (T, β) where T is a tree and $\beta : V(T) \to 2^{V(G)}$ satisfying:

1. $\{t : v \in \beta(t)\}$ is nonempty and connected for every $v \in V(G)$;
2. for every $uv \in E(G)$ there exists $t \in V(T)$ such that $u, v \in \beta(t)$.

Width of the decomposition is $\max_{t \in V(T)} |\beta(t)| - 1$.
Treewidth of G is minimum possible width of tree decomposition of G.

GI in bounded treewidth graphs

Theorem (Bodlaender'90)

Isomorphism of two n-vertex graphs of treewidth at most k can be tested in time $n^{\mathcal{O}(k)}$.

GI in bounded treewidth graphs

Theorem (Bodlaender'90)

Isomorphism of two n-vertex graphs of treewidth at most k can be tested in time $n^{\mathcal{O}(k)}$.

Theorem (Our result)

Isomorphism of two n-vertex graphs of treewidth at most k can be tested in time $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$.

GI in bounded treewidth graphs

Theorem (Bodlaender'90)

Isomorphism of two n-vertex graphs of treewidth at most k can be tested in time $n^{\mathcal{O}(k)}$.

Theorem (Our result)

Isomorphism of two n-vertex graphs of treewidth at most k can be tested in time $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$.

Theorem (Our result)

There is an algorithm, that given a graph G and integer k, runs in $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$ and either concludes that $\mathbf{tw}(G) > k$, or labels the vertices with numbers $1, 2, \ldots, n$ such that two isomorphic graphs receive labelings certifying the isomorphism.

Comparing tree decompositions

- In the planar case, it was easy to compare two graphs decomposed into 3-connected components.

Comparing tree decompositions

- In the planar case, it was easy to compare two graphs decomposed into 3-connected components.
- Similarly, it is not hard to compare pairs $(G_1, (T_1, \beta_1))$ and $(G_2, (T_2, \beta_3))$.

Comparing tree decompositions

- In the planar case, it was easy to compare two graphs decomposed into 3-connected components.
- Similarly, it is not hard to compare pairs $(G_1, (T_1, \beta_1))$ and $(G_2, (T_2, \beta_3))$.
  - Guess matching roots of tree decompositions.
  - Do bottom-up dynamic programming, computing isomorphic subtrees with labeled vertices in the top bags.
  - I.e., for every $v_1 \in V(T_1)$, $v_2 \in V(T_2)$, and every bijection $\pi : \beta_1(v_1) \to \beta_2(v_2)$, compute if G_i restricted to the subtree rooted in v_i are isomorphic consistently with π.

Comparing tree decompositions

- In the planar case, it was easy to compare two graphs decomposed into 3-connected components.
- Similarly, it is not hard to compare pairs $(G_1, (T_1, \beta_1))$ and $(G_2, (T_2, \beta_3))$.
  - Guess matching roots of tree decompositions.
  - Do bottom-up dynamic programming, computing isomorphic subtrees with labeled vertices in the top bags.
  - I.e., for every $v_1 \in V(T_1)$, $v_2 \in V(T_2)$, and every bijection $\pi : \beta_1(v_1) \to \beta_2(v_2)$, compute if G_i restricted to the subtree rooted in v_i are isomorphic consistently with π.
- Thus, in some sense, we look for an isomorphic-invariant way to compute a (near-)optimal tree decomposition.

Comparing tree decompositions

- In the planar case, it was easy to compare two graphs decomposed into 3-connected components.
- Similarly, it is not hard to compare pairs $(G_1, (T_1, \beta_1))$ and $(G_2, (T_2, \beta_3))$.
  - Guess matching roots of tree decompositions.
  - Do bottom-up dynamic programming, computing isomorphic subtrees with labeled vertices in the top bags.
  - I.e., for every $v_1 \in V(T_1)$, $v_2 \in V(T_2)$, and every bijection $\pi : \beta_1(v_1) \to \beta_2(v_2)$, compute if G_i restricted to the subtree rooted in v_i are isomorphic consistently with π.
- Thus, in some sense, we look for an isomorphic-invariant way to compute a (near-)optimal tree decomposition.
  - We can have some preliminary guessing, like guess one matched pairs of vertices etc.

Comparing tree decompositions

- In the planar case, it was easy to compare two graphs decomposed into 3-connected components.
- Similarly, it is not hard to compare pairs $(G_1, (T_1, \beta_1))$ and $(G_2, (T_2, \beta_3))$.
  - Guess matching roots of tree decompositions.
  - Do bottom-up dynamic programming, computing isomorphic subtrees with labeled vertices in the top bags.
  - I.e., for every $v_1 \in V(T_1)$, $v_2 \in V(T_2)$, and every bijection $\pi : \beta_1(v_1) \to \beta_2(v_2)$, compute if G_i restricted to the subtree rooted in v_i are isomorphic consistently with π.
- Thus, in some sense, we look for an isomorphic-invariant way to compute a (near-)optimal tree decomposition.
  - We can have some preliminary guessing, like guess one matched pairs of vertices etc.
  - More formally, we can generate $f(k)n^{\mathcal{O}(1)}$ candidate decompositions, and compare every pair.

Simplifications

- **Recall**: in the planar case, we could assume the graph is 3-connected.
  - Due to unique decomposition into 3-connected components.

Simplifications

- **Recall**: in the planar case, we could assume the graph is 3-connected.
  - Due to unique decomposition into 3-connected components.
- **Assumption 1**: no clique separators.
  - A decomposition by clique separators with unique set of bags. [Tarjan'85]
  - In particular, 2-connected.

Simplifications

- **Recall**: in the planar case, we could assume the graph is 3-connected.
  - Due to unique decomposition into 3-connected components.
- **Assumption 1**: no clique separators.
  - A decomposition by clique separators with unique set of bags. [Tarjan'85]
  - In particular, 2-connected.
- **Assumption 2**: $\forall uv \notin E(G)$, there is a u-v vertex cut of size $\leq k$.
  - If not true for some uv, add edge uv. (So-called *improved graph*.)
  - Isomorphism-invariant operation if done at once for all such uv.
  - Maintains assumption $\mathbf{tw}(G) \leq k$.

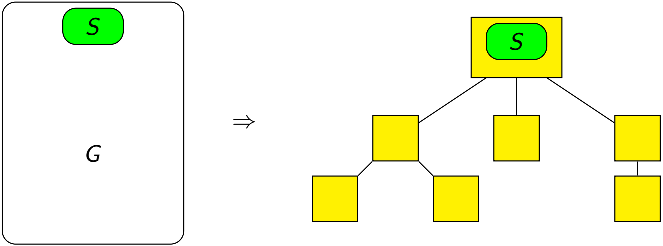
Task summary

Assumptions:

- 2-connected graph G, no clique separators.
- For every $uv \notin E(G)$, there is a u-v vertex cut of size at most k.

Task:
Compute isomorphism-invariant tree decomposition of G of width $\sim k$.
(Possibly after some small preliminary guessing.)

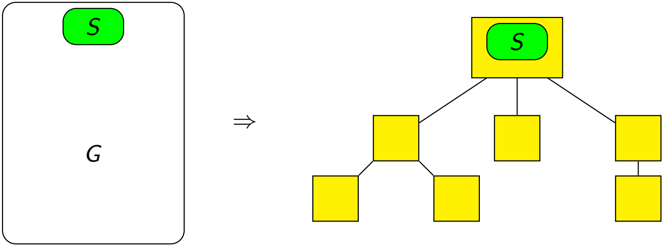
Robertson-Seymour approximation



Task in the recursion:

- given a graph G and a set $S \subseteq V(G)$, $|S| \leq 10k$,
- compute a tree decomposition of G of width $\mathcal{O}(k)$ with S in the top bag.

Robertson-Seymour approximation



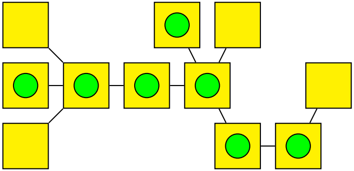
Task in the recursion:

- given a graph G and a set $S \subseteq V(G)$, $|S| \leq 10k$,
- compute a tree decomposition of G of width $\mathcal{O}(k)$ with S in the top bag.

Step 1: If $S = V(G)$, return single bag S.
Step 2: If $|S| < 10k$, then add an arbitrary vertex to S and recurse.

Robertson-Seymour approximation

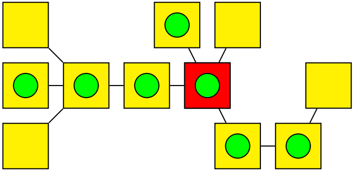
optimum decomposition of G:



Step 3: Assume then $|S| = 10k$ and $\mathbf{tw}(G) \leq k$.

Robertson-Seymour approximation

optimum decomposition of G:



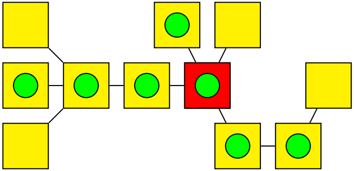
Step 3: Assume then $|S| = 10k$ and $\mathbf{tw}(G) \leq k$.

Lemma

There exists $Y \subseteq V(G)$, $|Y| \leq k+1$, such that every connected component of $G - Y$ contains at most $|S|/2$ vertices of S.

Robertson-Seymour approximation

optimum decomposition of G:

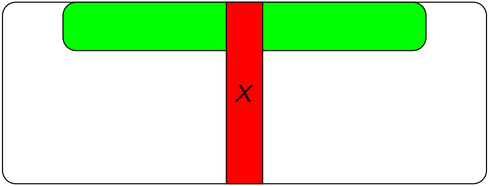


Step 3: Assume then $|S| = 10k$ and $\mathbf{tw}(G) \le k$.

Lemma

There exists $Y \subseteq V(G)$, $|Y| \le k+1$, such that every connected component of $G - Y$ contains at most $|S|/2$ vertices of S.

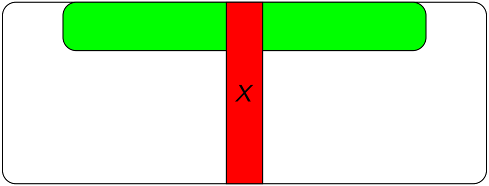
- There is a partition $S = S_1 \uplus S_2$ with $|S_1|, |S_2| \le 2|S|/3$ s.t. the minimum S_1-S_2 cut has size at most $k+1$.
- Iterate through all such partitions and let X be the found mincut.
- Pick $X \cup S$ as the root bag.
- Recurse on every connected component C of $G - (S \cup X)$ with graph $G[N[C]]$ and $S := N(C)$.



Lemma

There exists a partition $S = S_1 \uplus S_2$ *with* $|S_1|, |S_2| \leq 2|S|/3$ *such that there is* $X \subseteq V(G)$ *with* $|X| \leq k+1$ *that separates* S_1 *from* S_2.

Robertson-Seymour approximation

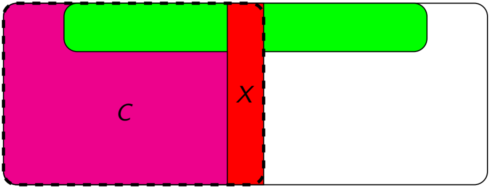


Lemma

There exists a partition $S = S_1 \uplus S_2$ with $|S_1|, |S_2| \leq 2|S|/3$ such that there is $X \subseteq V(G)$ with $|X| \leq k+1$ that separates S_1 from S_2.

- Find X by checking mincut for every balanced partition $S = S_1 \uplus S_2$.
- Pick $X \cup S$ as a root bag.
  - Size $\leq 10k + k + 1$.
- Recurse on every connected component C of $G - (S \cup X)$ with graph $G[N[C]]$ and $S := N(C)$.
  - $|N(C)| \leq 2|S|/3 + |X| < 10k$.

Robertson-Seymour approximation

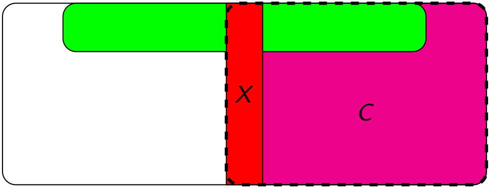


Lemma

There exists a partition $S = S_1 \uplus S_2$ with $|S_1|, |S_2| \leq 2|S|/3$ such that there is $X \subseteq V(G)$ with $|X| \leq k+1$ that separates S_1 from S_2.

- Find X by checking mincut for every balanced partition $S = S_1 \uplus S_2$.
- Pick $X \cup S$ as a root bag.
  - Size $\leq 10k + k + 1$.
- Recurse on every connected component C of $G - (S \cup X)$ with graph $G[N[C]]$ and $S := N(C)$.
  - $|N(C)| \leq 2|S|/3 + |X| < 10k$.

Robertson-Seymour approximation



Lemma

There exists a partition $S = S_1 \uplus S_2$ with $|S_1|, |S_2| \leq 2|S|/3$ such that there is $X \subseteq V(G)$ with $|X| \leq k+1$ that separates S_1 from S_2.

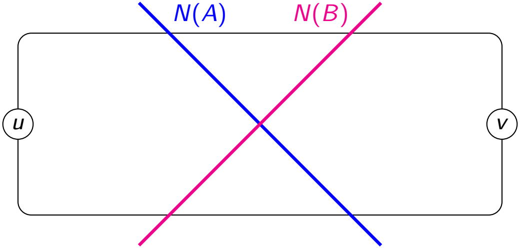
- Find X by checking mincut for every balanced partition $S = S_1 \uplus S_2$.
- Pick $X \cup S$ as a root bag.
  - Size $\leq 10k + k + 1$.
- Recurse on every connected component C of $G - (S \cup X)$ with graph $G[N[C]]$ and $S := N(C)$.
  - $|N(C)| \leq 2|S|/3 + |X| < 10k$.

Robertson-Seymour approximation

Two arbitrary decisions:

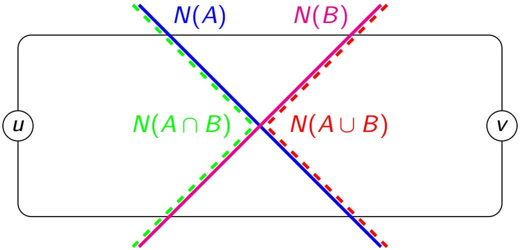
- **Step 2**: If $|S| < 10k$, then add an arbitrary vertex to S and recurse.
  - Which vertex to choose?
- **Step 3**: Pick any separator X that splits S well.
  - Which separator to choose?

Submodularity of cuts



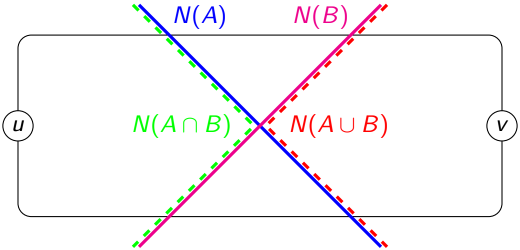
- Let $N(A)$ and $N(B)$ be two minimum uv separators.

Submodularity of cuts



- Let $N(A)$ and $N(B)$ be two minimum uv separators.
- Then $N(A \cap B)$ and $N(A \cup B)$ are also minimum uv separators.

Submodularity of cuts

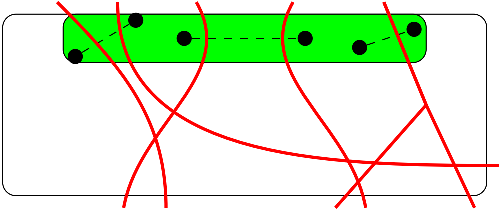


- Let $N(A)$ and $N(B)$ be two minimum uv separators.
- Then $N(A \cap B)$ and $N(A \cup B)$ are also minimum uv separators.
- Therefore there is a notion of minimum uv separator *closest to u* and *closest to v*.
  - Unique minimum separators that leaves inclusion-wise minimal and maximal set of vertices reachable from u.

Solution to Step 2

- **Step 2**: If $|S| < 10k$, then add an arbitrary vertex to S and recurse.

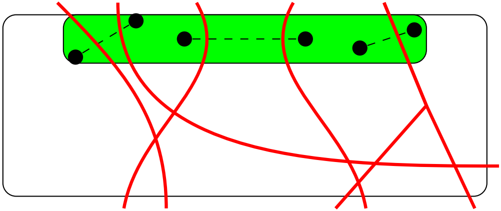
Solution to Step 2



- **Step 2**: If $|S| < 10k$, then add an arbitrary vertex to S and recurse.
- Suppose $|S| < 10k$. For every $u, v \in S$, $uv \notin E(G)$, $u \neq v$, let $X_{u,v}$ be the minimum uv separator closest to u.
  Pick root bag

$$B := S \cup \bigcup_{u,v \in S, uv \notin E(G), u \neq v} X_{u,v}.$$

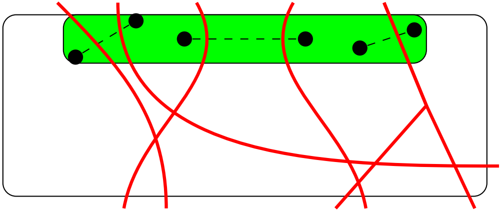
Solution to Step 2



$$B := S \cup \bigcup_{u,v \in S, uv \notin E(G), u \neq v} X_{u,v}.$$

- Again, recurse on $(G[N[C]], N(C))$ for C being connected components of $G - B$.
  - Definition is isomorphism invariant, and $|B| = \mathcal{O}(k|S|^2)$.
  - $N(C)$ can be as big as $\mathcal{O}(k|S|^2)$.

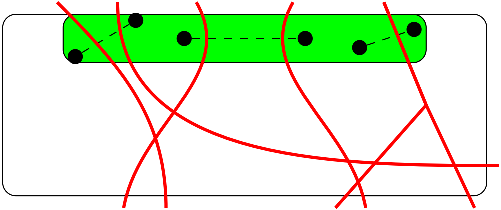
Solution to Step 2



$$B := S \cup \bigcup_{u,v \in S, uv \notin E(G), u \neq v} X_{u,v}.$$

- Again, recurse on $(G[N[C]], N(C))$ for C being connected components of $G - B$.
    - Definition is isomorphism invariant, and $|B| = \mathcal{O}(k|S|^2)$.
    - $N(C)$ can be as big as $\mathcal{O}(k|S|^2)$.
- Issue: do we always make progress?

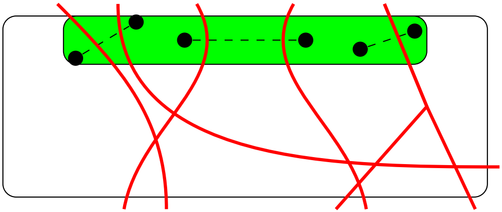
Solution to Step 2



$$B := S \cup \bigcup_{u,v \in S, uv \notin E(G), u \neq v} X_{u,v}.$$

- We will always recurse on instances of the form $(G[N[C]], S := N(C))$ for some connected C.

Solution to Step 2

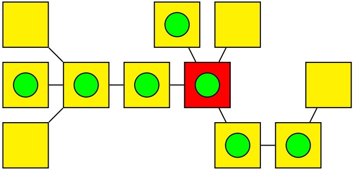


$$B := S \cup \bigcup_{u,v \in S, uv \notin E(G), u \neq v} X_{u,v}.$$

- We will always recurse on instances of the form $(G[N[C]], S := N(C))$ for some connected C.
- Hence $S = N(C)$ is always a separator, and thus never a clique.
  - We need to hack it at the begining of the recursion, but we can use preliminary guessing for that, e.g., guess a mapping on one non-edge.

Solution to Step 3

optimum
decomposition of G:

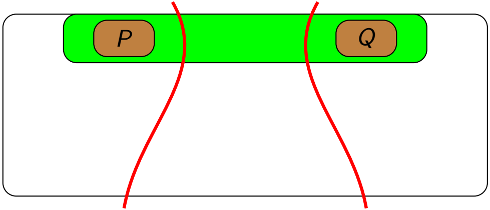


Step 3: We have $|S| \geq 10k$ and $\mathbf{tw}(G) \leq k$.

Lemma

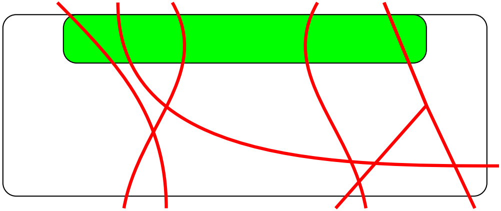
There exists a partition $S = S_1 \uplus S_2$ with $|S_1|, |S_2| \leq 2|S|/3$ such that there is $X \subseteq V(G)$ with $|X| \leq k+1$ that separates S_1 from S_2.

Solution to Step 3



- For every $P, Q \subseteq S$, $P \cap Q = \emptyset$, $|P| = |Q| = k + 2$, if there exists a PQ separator of size at most $k + 1$, let $X_{P,Q}$ be the minimum one closest to P.

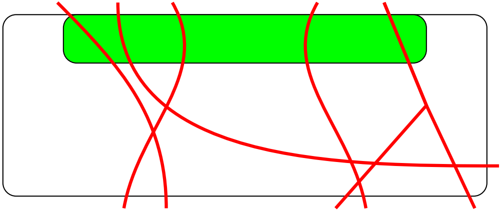
Solution to Step 3



- For every $P, Q \subseteq S$, $P \cap Q = \emptyset$, $|P| = |Q| = k + 2$, if there exists a PQ separator of size at most $k + 1$, let $X_{P,Q}$ be the minimum one closest to P.
- Pick root bag

$$B := S \cup \bigcup_{P,Q \text{ as above}} X_{P,Q}.$$

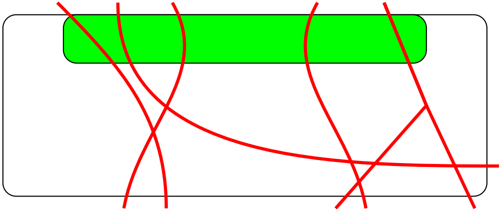
Solution to Step 3



- For every $P, Q \subseteq S$, $P \cap Q = \emptyset$, $|P| = |Q| = k + 2$, if there exists a PQ separator of size at most $k + 1$, let $X_{P,Q}$ be the minimum one closest to P.
- Pick root bag

$$B := S \cup \bigcup_{P,Q \text{ as above}} X_{P,Q}.$$

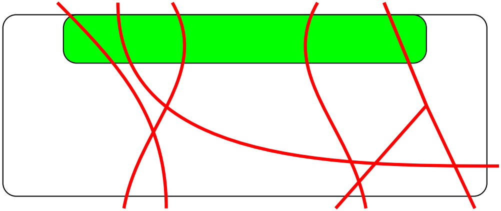
- Recurse as previously on all $(G[N[C]], N(C))$ for C being connected components of $G - B$.



$$B := S \cup \bigcup_{P,Q \text{ as before}} X_{P,Q}.$$

- We have a bound $|B| = \mathcal{O}(k|S|^{2k+4})$.

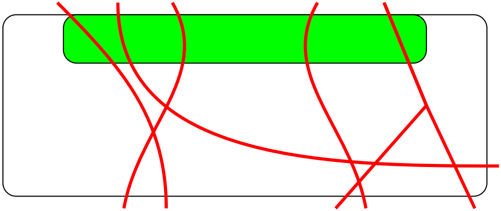
Solution to Step 3



$$B := S \cup \bigcup_{P,Q \text{ as before}} X_{P,Q}.$$

- We have a bound $|B| = \mathcal{O}(k|S|^{2k+4})$.
- But the main question is: how big can be $N(C)$ for C being a connected component of $G - B$.

Solution to Step 3



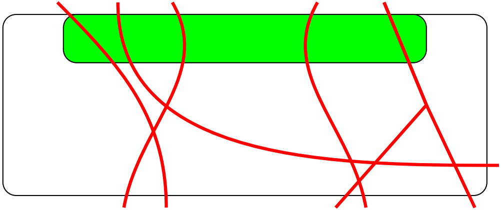
$$B := S \cup \bigcup_{P,Q \text{ as before}} X_{P,Q}.$$

- We have a bound $|B| = \mathcal{O}(k|S|^{2k+4})$.
- But the main question is: how big can be $N(C)$ for C being a connected component of $G - B$.

Lemma (The crux)

For every connected component C of $G - B$ we have $|N(C)| \leq |S|$.

Solution to Step 3



Lemma (The crux)

For every connected component C of $G - B$ we have $|N(C)| \leq |S|$.

- We analyze adding sets $X_{P,Q}$ to B one-by-one, and analyze sizes of $N(C)$ for intermediate connected components of $G - B$.

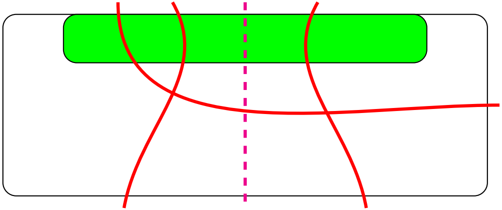
Solution to Step 3

Lemma (The crux)

For every connected component C of $G - B$ we have $|N(C)| \leq |S|$.

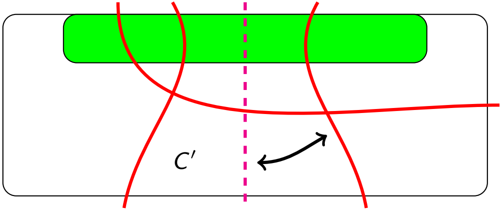
- We analyze adding sets $X_{P,Q}$ to B one-by-one, and analyze sizes of $N(C)$ for intermediate connected components of $G - B$.
- Initially, every component of $G - S$ has neighborhood contained in S.

Solution to Step 3



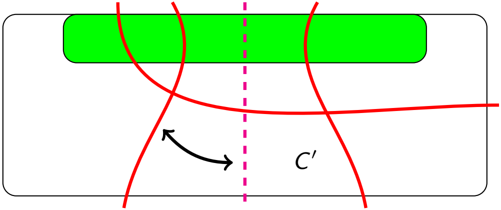
- Now consider adding one new pink cut $X_{P,Q}$.

Solution to Step 3



- Now consider adding one new pink cut $X_{P,Q}$.
- A border replacement argument says that the borders of components do not increase.
  - Formally, we use the notion of a separation and submodularity.

Solution to Step 3



- Now consider adding one new pink cut $X_{P,Q}$.
- A border replacement argument says that the borders of components do not increase.
  - Formally, we use the notion of a separation and submodularity.

Summary

Step 2: If $|S| < 10k$, add a minimum cut for every nonedge.

- Bag of size $\mathcal{O}(k^3)$.
- Blow up to $|S| = \mathcal{O}(k^3)$ in the subcalls.

Summary

Step 2: If $|S| < 10k$, add a minimum cut for every nonedge.

- Bag of size $\mathcal{O}(k^3)$.
- Blow up to $|S| = \mathcal{O}(k^3)$ in the subcalls.

Step 3: If $|S| \geq 10k$, add a minimum cut of size $\leq k+1$ for every pair (P, Q) of sets of size $k+2$.

- Bag of size $\mathcal{O}(k|S|^{2k+4})$.
- The crux: does not blow up $|S|$ in the subcalls.
- Thus, $|S| = \mathcal{O}(k^3)$ all the time and bags size is bounded by $2^{\mathcal{O}(k \log k)}$.

Summary

Step 2: If $|S| < 10k$, add a minimum cut for every nonedge.

- Bag of size $\mathcal{O}(k^3)$.
- Blow up to $|S| = \mathcal{O}(k^3)$ in the subcalls.

Step 3: If $|S| \geq 10k$, add a minimum cut of size $\leq k+1$ for every pair (P, Q) of sets of size $k+2$.

- Bag of size $\mathcal{O}(k|S|^{2k+4})$.
- The crux: does not blow up $|S|$ in the subcalls.
- Thus, $|S| = \mathcal{O}(k^3)$ all the time and bags size is bounded by $2^{\mathcal{O}(k \log k)}$.

Theorem

For a graph G of treewidth $\leq k$, we can obtain an isomorphism-invariant family of at most n^2 tree decompositions with bags of size $2^{\mathcal{O}(k \log k)}$ and adhesions of size $\mathcal{O}(k^3)$.

Graph Isomorphism

Theorem

For a graph G of treewidth $\leq k$, we can obtain an isomorphism-invariant family of at most n^2 tree decompositions with bags of size $2^{\mathcal{O}(k \log k)}$ and adhesions of size $\mathcal{O}(k^3)$.

Graph Isomorphism

Theorem

For a graph G of treewidth $\leq k$, we can obtain an isomorphism-invariant family of at most n^2 tree decompositions with bags of size $2^{\mathcal{O}(k \log k)}$ and adhesions of size $\mathcal{O}(k^3)$.

Theorem

We can check if two graphs of treewidth $\leq k$ are isomorphic in time $2^{2^{\mathcal{O}(k \log k)}} \cdot n^{\mathcal{O}(1)}$.

Graph Isomorphism

Theorem

For a graph G of treewidth $\leq k$, we can obtain an isomorphism-invariant family of at most n^2 tree decompositions with bags of size $2^{\mathcal{O}(k \log k)}$ and adhesions of size $\mathcal{O}(k^3)$.

Theorem

We can check if two graphs of treewidth $\leq k$ are isomorphic in time $2^{2^{\mathcal{O}(k \log k)}} \cdot n^{\mathcal{O}(1)}$.

Now: a quick sketch how to reduce dependency on k to $2^{k^{\mathcal{O}(1)}}$.

Getting single-exponential running time

- Instead of isomorphism-invariant tree decomposition, we want only *isomorphism-invariant family of candidate bags*.
  - Formally, we require to capture at least one full decomposition in the family of bags.
  - Later, we can use DP on tuples (bag B, a connected component of $G - B$, labeling of B) and compare them.
  - Alternatively, can use recent framework of [Otachi-Schweitzer'14].

Getting single-exponential running time

- Instead of isomorphism-invariant tree decomposition, we want only *isomorphism-invariant family of candidate bags*.
  - Formally, we require to capture at least one full decomposition in the family of bags.
  - Later, we can use DP on tuples (bag B, a connected component of $G - B$, labeling of B) and compare them.
  - Alternatively, can use recent framework of [Otachi-Schweitzer'14].
- We have tree decomposition with bags of size $2^{\mathcal{O}(k \log k)}$ and adhesions of size $\mathcal{O}(k^3)$, but the graph is of treewidth $\leq k$.

Getting single-exponential running time

- Instead of isomorphism-invariant tree decomposition, we want only *isomorphism-invariant family of candidate bags*.
  - Formally, we require to capture at least one full decomposition in the family of bags.
  - Later, we can use DP on tuples (bag B, a connected component of $G - B$, labeling of B) and compare them.
  - Alternatively, can use recent framework of [Otachi-Schweitzer'14].
- We have tree decomposition with bags of size $2^{\mathcal{O}(k \log k)}$ and adhesions of size $\mathcal{O}(k^3)$, but the graph is of treewidth $\leq k$.
- Every our bag B can be further decomposed with width $\leq k$.

Getting single-exponential running time

- Instead of isomorphism-invariant tree decomposition, we want only *isomorphism-invariant family of candidate bags*.
  - Formally, we require to capture at least one full decomposition in the family of bags.
  - Later, we can use DP on tuples (bag B, a connected component of $G - B$, labeling of B) and compare them.
  - Alternatively, can use recent framework of [Otachi-Schweitzer'14].
- We have tree decomposition with bags of size $2^{\mathcal{O}(k \log k)}$ and adhesions of size $\mathcal{O}(k^3)$, but the graph is of treewidth $\leq k$.
- Every our bag B can be further decomposed with width $\leq k$.
- We output every subset of size $\mathcal{O}(k^4)$ of every bag in our decompositions, and this is guaranteed to capture some decomposition of width $\mathcal{O}(k^4)$.

Getting single-exponential running time

Theorem

For a graph G of treewidth $\leq k$, we can output an isomorphism-invariant family $\mathcal{B}$ of size $2^{\mathcal{O}(k^5 \log k)} \cdot n^2$, where every element of $\mathcal{B}$ is a subset of $V(G)$ of size $\mathcal{O}(k^4)$ and $\mathcal{B}$ contains all bags of some tree decomposition of G.

Getting single-exponential running time

Theorem

For a graph G of treewidth $\leq k$, we can output an isomorphism-invariant family $\mathcal{B}$ of size $2^{\mathcal{O}(k^5 \log k)} \cdot n^2$, where every element of $\mathcal{B}$ is a subset of $V(G)$ of size $\mathcal{O}(k^4)$ and $\mathcal{B}$ contains all bags of some tree decomposition of G.

Theorem

Isomorphism of two n-vertex graphs of treewidth at most k can be tested in time $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$.

Getting single-exponential running time

Theorem

For a graph G of treewidth $\leq k$, we can output an isomorphism-invariant family $\mathcal{B}$ of size $2^{\mathcal{O}(k^5 \log k)} \cdot n^2$, where every element of $\mathcal{B}$ is a subset of $V(G)$ of size $\mathcal{O}(k^4)$ and $\mathcal{B}$ contains all bags of some tree decomposition of G.

Theorem

Isomorphism of two n-vertex graphs of treewidth at most k can be tested in time $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$.

Theorem

There is an algorithm, that given a graph G and integer k, runs in $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$ and either concludes that $\mathbf{tw}(G) > k$, or labels the vertices with numbers $1, 2, \ldots, n$ such that two isomorphic graphs receive labelings certifying the isomorphism.

Conclusions

Theorem

Isomorphism of two n-vertex graphs of treewidth at most k can be tested in time $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$.

Conclusions

Theorem

Isomorphism of two n-vertex graphs of treewidth at most k can be tested in time $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$.

Open problems:

- What about FPT algorithm for graph isomorphism parameterized by the maximum degree?
  - Luks' algorithm has running time $\mathcal{O}(n^{f(\Delta)})$.
- What about FPT algorithm for graph isomorphism parameterized by the size of an excluded minor?
  - Ponomarenko's algorithm has running time $\mathcal{O}(n^{f(|H|)})$.