Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal

Dániel Marx

Humboldt-Universität zu Berlin

Joint work with

Daniel Lokshtanov   Saket Saurabh

ACM-SIAM Symposium on Discrete Algorithms (SODA 2011) – Jan 24, 2011

Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal

Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal

Treewidth

Treewidth: A measure of how "tree-like" the graph is.
(Introduced by Robertson and Seymour in the Graph Minors project.)

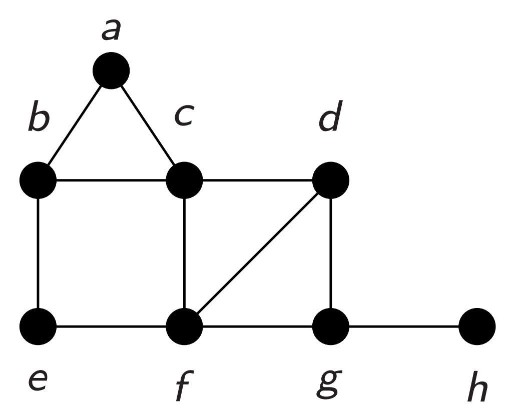
Significance:

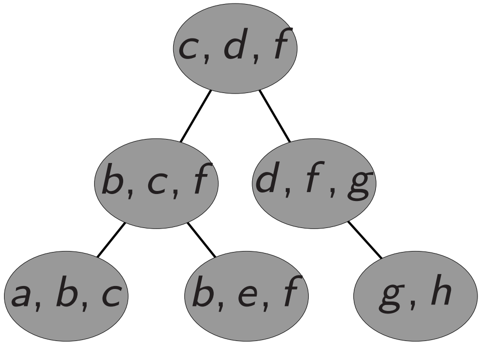
- Appears naturally in graph structure theory.
- Polynomial or even linear algorithms for NP-hard problems on bounded treewidth graphs.
- Crucial tool for planar approximation schemes.
- Useful for fixed-parameter tractability results.

Treewidth

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every vertex v, the bags containing v form a connected subtree.

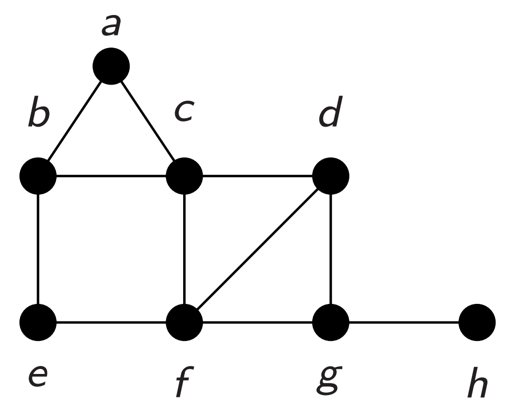


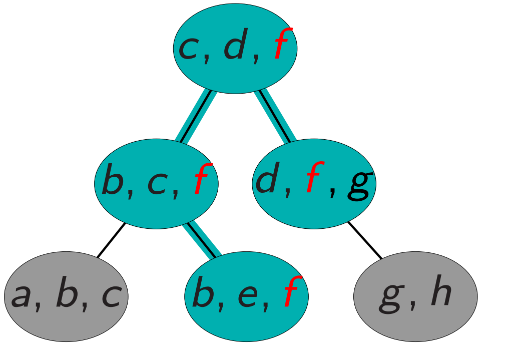


Treewidth

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every vertex v, the bags containing v form a connected subtree.





Treewidth

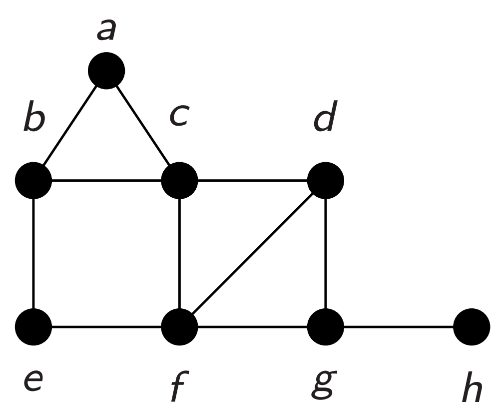
Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

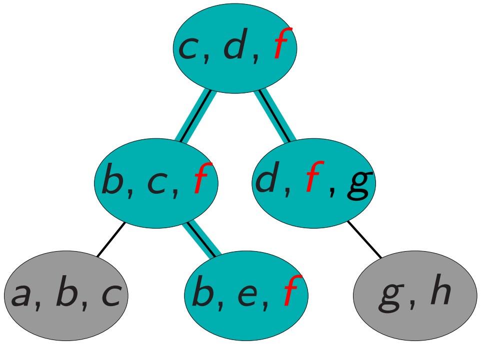
1. If u and v are neighbors, then there is a bag containing both of them.
2. For every vertex v, the bags containing v form a connected subtree.

Width of decomposition: largest bag size -1.

treewidth: width of the best decomposition.

Fact: treewidth $= 1 \iff$ graph is a forest





Treewidth

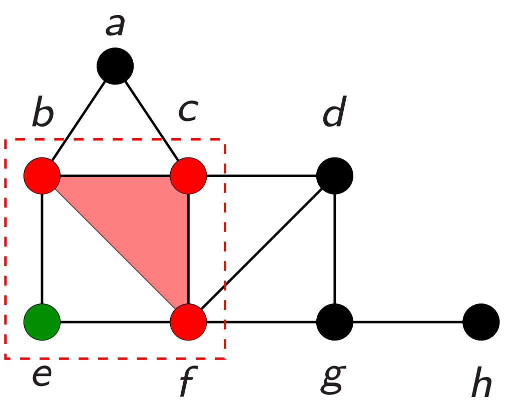
Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

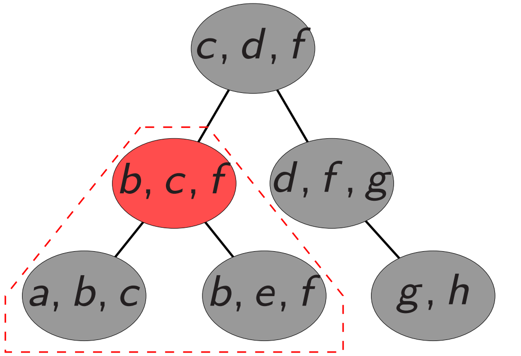
1. If u and v are neighbors, then there is a bag containing both of them.
2. For every vertex v, the bags containing v form a connected subtree.

Width of decomposition: largest bag size -1.

treewidth: width of the best decomposition.

Fact: treewidth $= 1 \iff$ graph is a forest





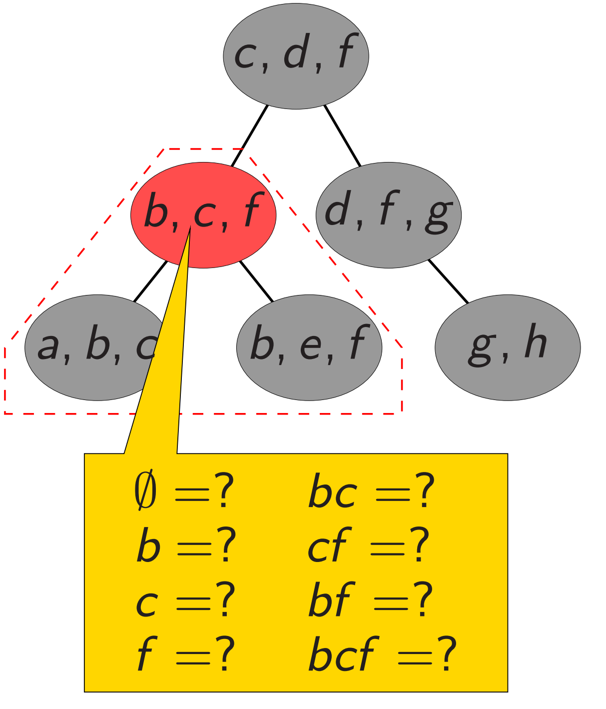
Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal

MAX INDEPENDENT SET and tree decompositions

Fact: Given a tree decomposition of width w, MAX INDEPENDENT SET can be solved in time $O(2^w \cdot n)$.

B_x: vertices appearing in node x.
V_x: vertices appearing in the subtree rooted at x.

- Define table $M[x, S]$: the maximum weight of an independent set $I \subseteq V_x$ with $I \cap B_x = S$.
- Compute the tables in bottom-up order.
- Size of each table is 2^{w+1}.



Algorithms

Given a tree decomposition of width w, dynamic programming gives:

INDEPENDENT SET	$O(2^w \cdot n)$
DOMINATING SET	$O(3^w \cdot n)$
MAX CUT	$O(2^w \cdot n)$
ODD CYCLE TRANSVERSAL	$O(3^w \cdot n)$
q-COLORING ($q \geq 3$)	$O(q^w \cdot n)$
PARTITION INTO TRIANGLES	$O(2^w \cdot n)$
	[various authors]

Algorithms

Given a tree decomposition of width w, dynamic programming gives:

INDEPENDENT SET	$O(2^w \cdot n)$
DOMINATING SET	$O(3^w \cdot n)$
MAX CUT	$O(2^w \cdot n)$
ODD CYCLE TRANSVERSAL	$O(3^w \cdot n)$
q-COLORING ($q \geq 3$)	$O(q^w \cdot n)$
PARTITION INTO TRIANGLES	$O(2^w \cdot n)$
	[various authors]

Question: Can we improve the base in any of these algorithms?

Supporting evidence: Running time matches the obvious DP table size. But...

Some history

DOMINATING SET

- Obvious approach: 9^w [Telle and Proskurowski ’93]
- More clever algorithm: 4^w [Alber et al. ’02]
- Even more clever algorithm: 3^w [Rooij et al. ’09] using fast subset convolution of [Björklund et al. ’07]

Some history

DOMINATING SET

- Obvious approach: 9^w [Telle and Proskurowski ’93]
- More clever algorithm: 4^w [Alber et al. ’02]
- Even more clever algorithm: 3^w [Rooij et al. ’09] using fast subset convolution of [Björklund et al. ’07]

HAMILTONIAN CYCLE

- 2^n time [Held and Karp ’62]
- 1.657^n (randomized) time [Björklund ’10]

Some history

DOMINATING SET

- Obvious approach: 9^w [Telle and Proskurowski '93]
- More clever algorithm: 4^w [Alber et al. '02]
- Even more clever algorithm: 3^w [Rooij et al. '09] using fast subset convolution of [Björklund et al. '07]

HAMILTONIAN CYCLE

- 2^n time [Held and Karp '62]
- 1.657^n (randomized) time [Björklund '10]

DIRECTED FEEDBACK VERTEX SET

- Trivial 2^n algorithm.
- Nontrivial 1.9977^n algorithm [Razgon '07]

Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal

SETH

Obviously, we need a hardness assumption.

P $\neq$ NP is not sufficiently strong: even a $O(2^{\sqrt{w}} \cdot n)$ algorithm seems to be compatible with it.

SETH

Obviously, we need a hardness assumption.

P $\neq$ NP is not sufficiently strong: even a $O(2^{\sqrt{w}} \cdot n)$ algorithm seems to be compatible with it.

> Strong Exponential Time Hypothesis (SETH):
>
> $s_k = \inf\{\delta \mid n\text{-variable } k\text{-SAT } \textit{can be solved in } 2^{\delta n}\}$
>
> **Conjecture:** [Impagliazzo-Paturi '01] $s_k \to 1$

We can use a somewhat weaker assumption:

> No faster SAT:
>
> **Conjecture:** n-variable m-clause SAT (with arbitrary clause length) cannot be solved in time $(2-\epsilon)^n \cdot \text{poly}(m)$ for any $\epsilon > 0$.

Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal

Results

Main result: If the Strong Exponential Time Hypothesis (SETH) is true, then given a tree decomposition of width w,

INDEPENDENT SET		$(2-\epsilon)^w \cdot n^{O(1)}$
DOMINATING SET		$(3-\epsilon)^w \cdot n^{O(1)}$
MAX CUT	cannot be	$(2-\epsilon)^w \cdot n^{O(1)}$
ODD CYCLE TRANSVERSAL	solved in time	$(3-\epsilon)^w \cdot n^{O(1)}$
q-COLORING ($q \geq 3$)		$(q-\epsilon)^w \cdot n^{O(1)}$
PARTITION INTO TRIANGLES		$(2-\epsilon)^w \cdot n^{O(1)}$

The lower bounds match the known algorithms (up to the ϵ in the base).

Note: For some problems, we can obtain stronger results by proving the same lower bound with respect to pathwidth or feedback vertex number.

Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal

Reductions

Suppose we have a reduction:

n-variable SAT instance	$\Rightarrow$	INDEPENDENT SET instance of treewidth $w \leq c \cdot n$.

Then:

$(2-\epsilon)^{c \cdot n}$ algorithm for SAT	$\Leftarrow$	$(2-\epsilon)^{w} \cdot n^{O(1)}$ algorithm for INDEPENDENT SET

- To get a $(2-\epsilon)^{w}$ lower bound, we need $c \leq 1$.

Reductions

Suppose we have a reduction:

n-variable SAT instance	$\Rightarrow$	INDEPENDENT SET instance of treewidth $w \leq c \cdot n$.

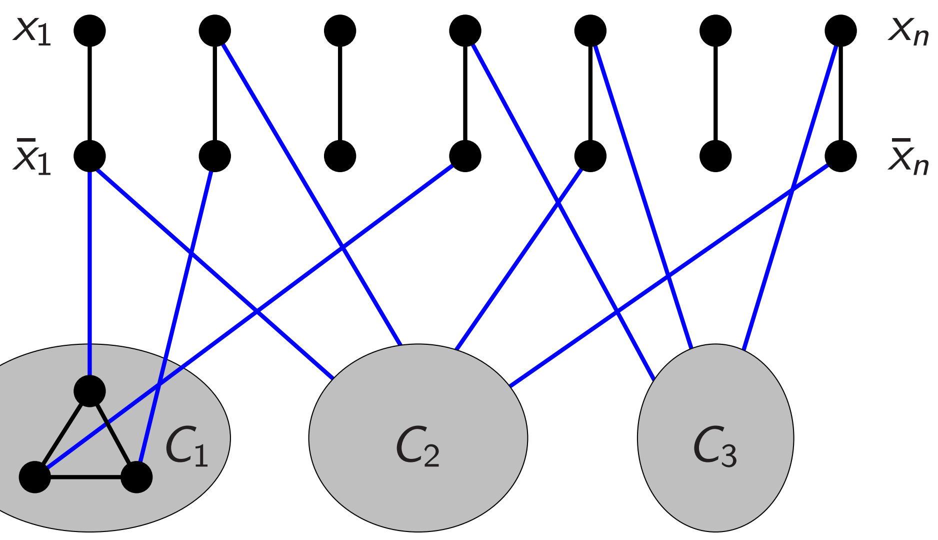
Then:

$(2-\epsilon)^{c \cdot n}$ algorithm for SAT	$\Leftarrow$	$(2-\epsilon)^{w} \cdot n^{O(1)}$ algorithm for INDEPENDENT SET

- To get a $(2-\epsilon)^w$ lower bound, we need $c \leq 1$.
- **More generally:** For any c, we get a $(2^{1/c}-\epsilon)^w$ lower bound
  $\Rightarrow$ To get a $(3-\epsilon)^w$ lower bound (e.g., for DOMINATING SET), we need $c \leq \log_3 2 \approx 0.631$.

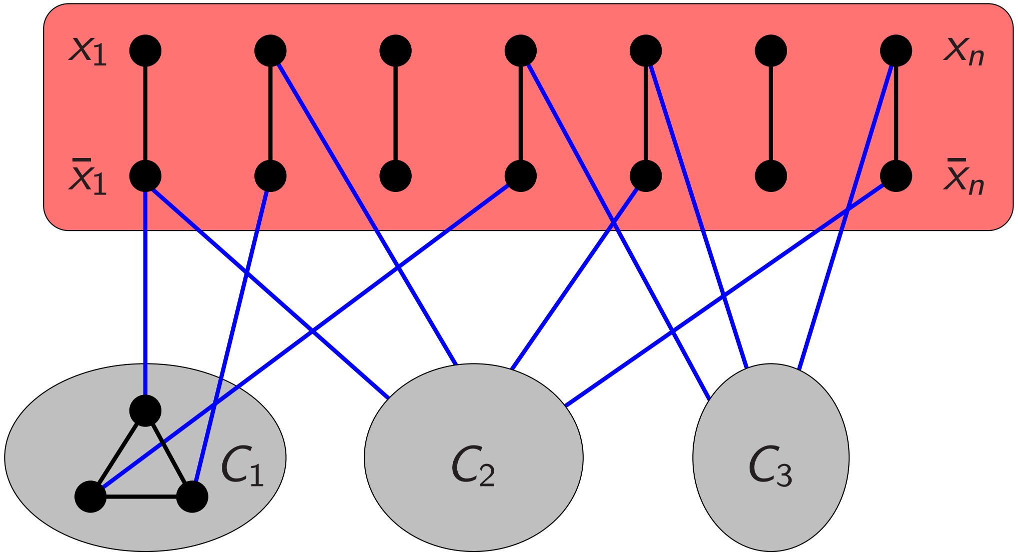
Textbook reduction

How large is the treewidth in the textbook reduction from SAT to INDEPENDENT SET?



Textbook reduction

How large is the treewidth in the textbook reduction from SAT to INDEPENDENT SET?

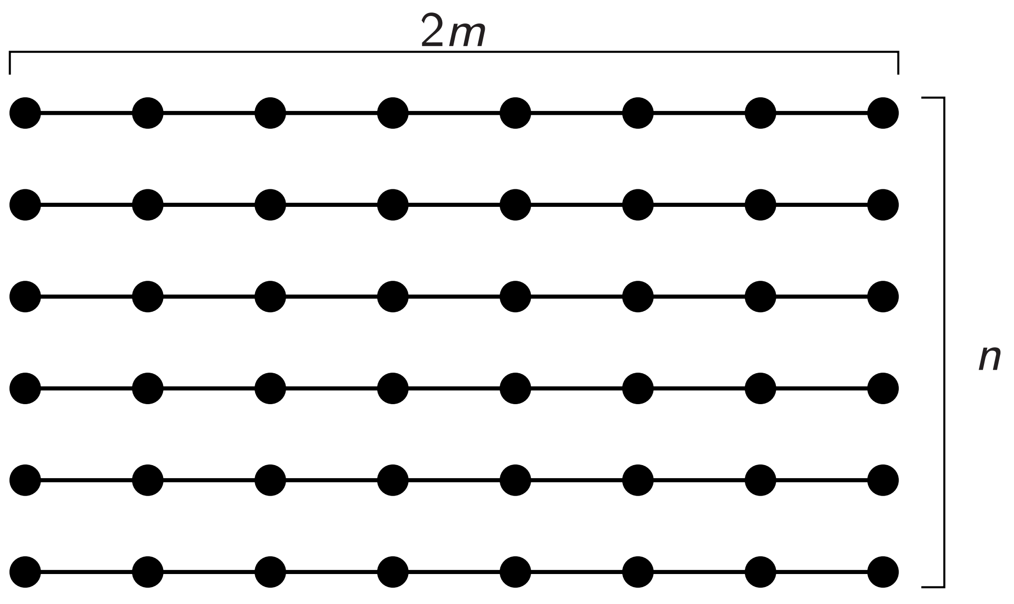


Treewidth is about $2n$, which gives a $(2^{\frac{1}{2}} - \epsilon)^w \approx 1.41^w$ lower bound.

We need treewidth $\leq n$ for the $(2 - \epsilon)^w$ lower bound.

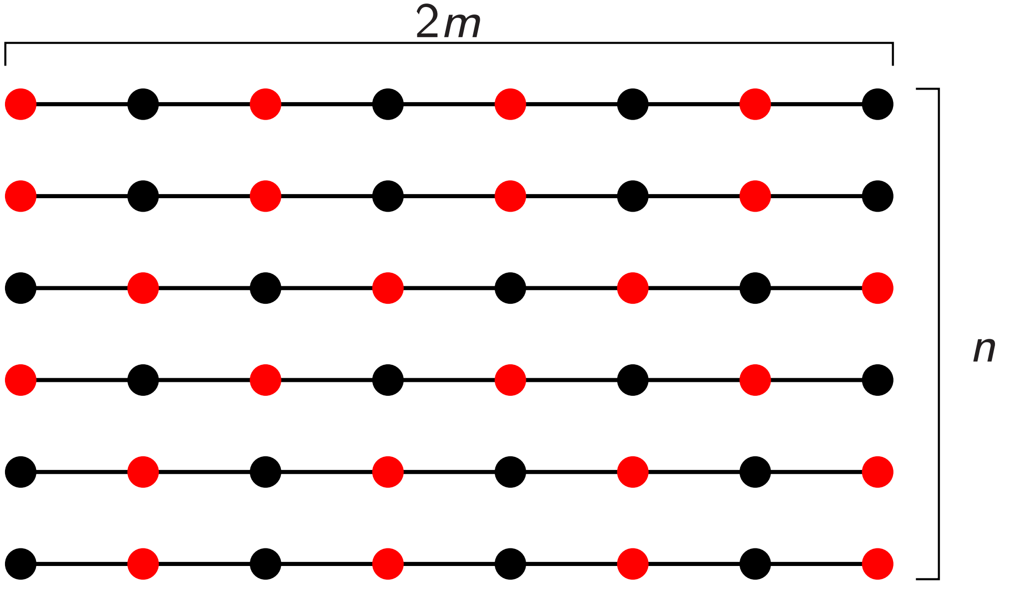
New reduction for INDEPENDENT SET

n variables, m clauses	$\Rightarrow$	n paths of $2m$ vertices each
2 states per each variable	$\Rightarrow$	2 possible states for each path



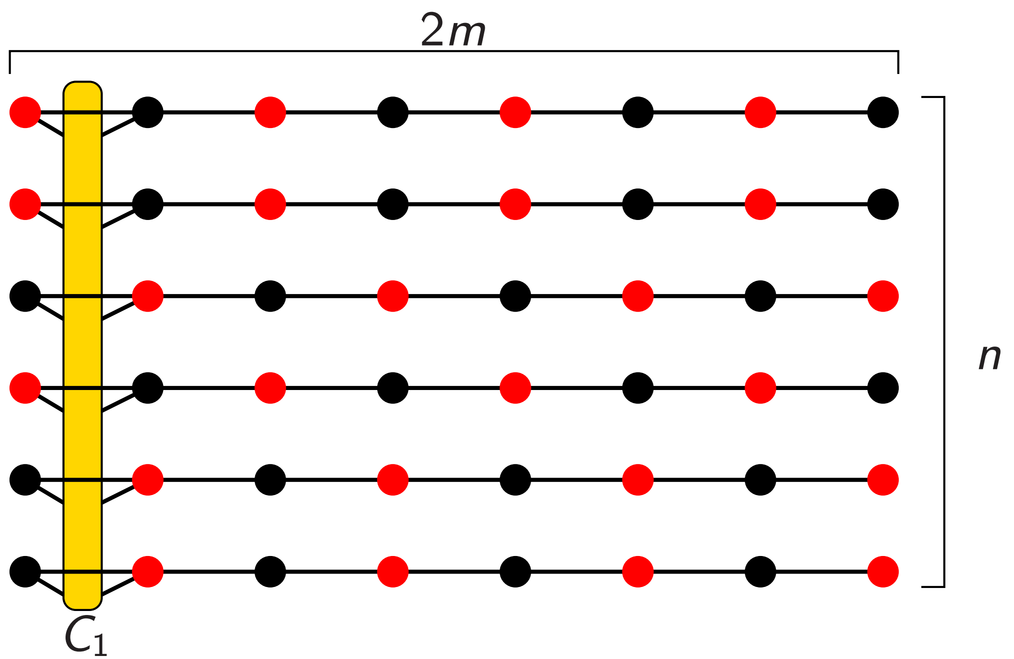
New reduction for INDEPENDENT SET

n variables, m clauses	$\Rightarrow$	n paths of $2m$ vertices each
2 states per each variable	$\Rightarrow$	2 possible states for each path



New reduction for INDEPENDENT SET

n variables, m clauses	$\Rightarrow$	n paths of $2m$ vertices each
2 states per each variable	$\Rightarrow$	2 possible states for each path

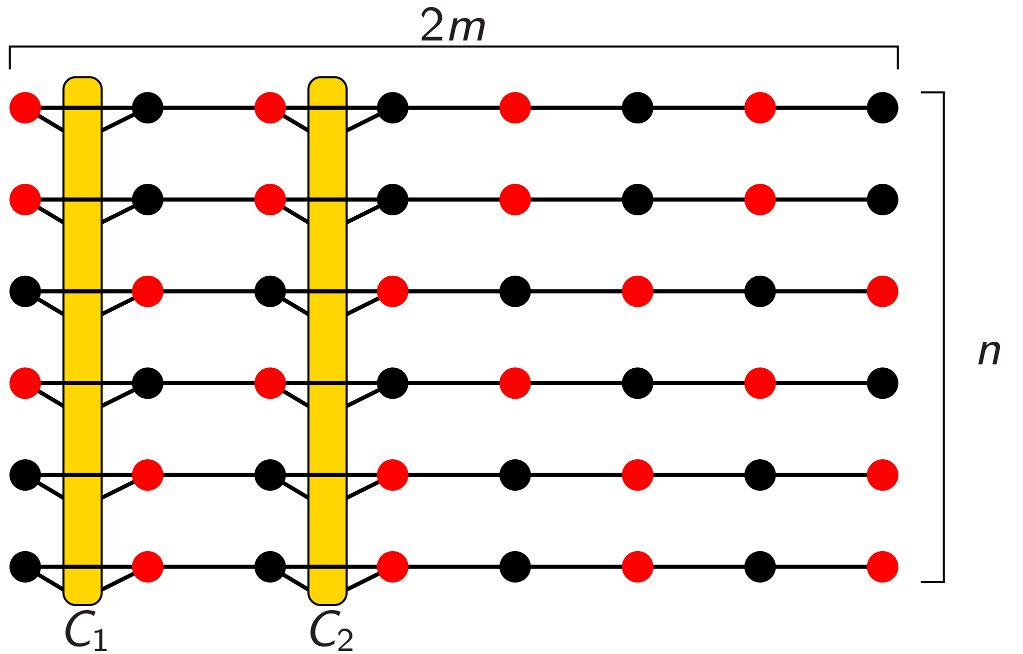


Clause gadgets check that every clause is satisfied.

Treewidth is only $n + O(1)$.

New reduction for INDEPENDENT SET

n variables, m clauses	$\Rightarrow$	n paths of $2m$ vertices each
2 states per each variable	$\Rightarrow$	2 possible states for each path

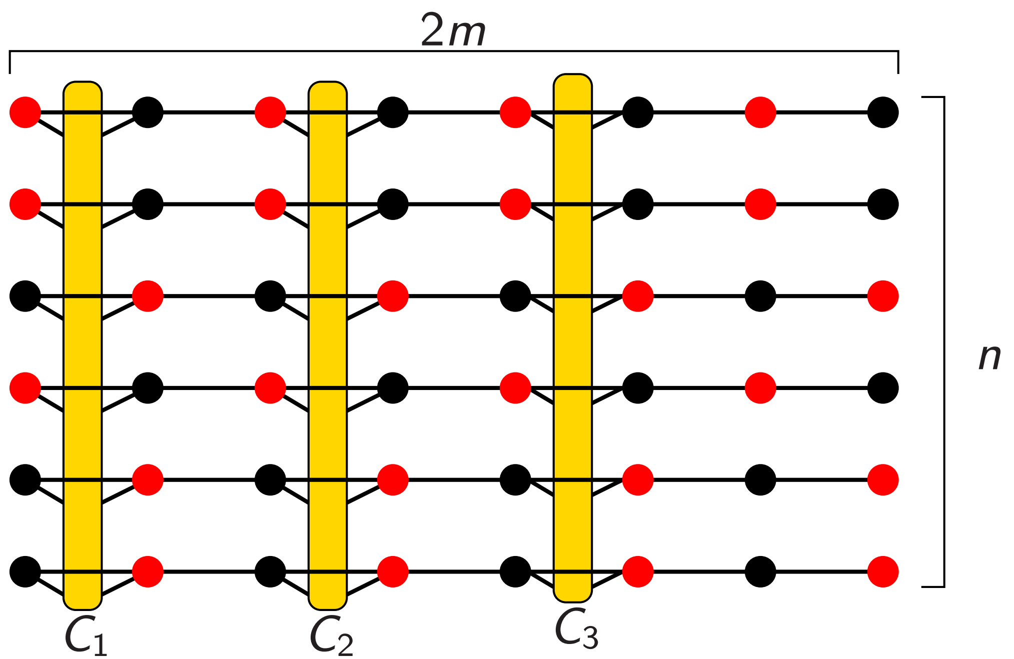


Clause gadgets check that every clause is satisfied.

Treewidth is only $n + O(1)$.

New reduction for INDEPENDENT SET

n variables, m clauses	$\Rightarrow$	n paths of $2m$ vertices each
2 states per each variable	$\Rightarrow$	2 possible states for each path

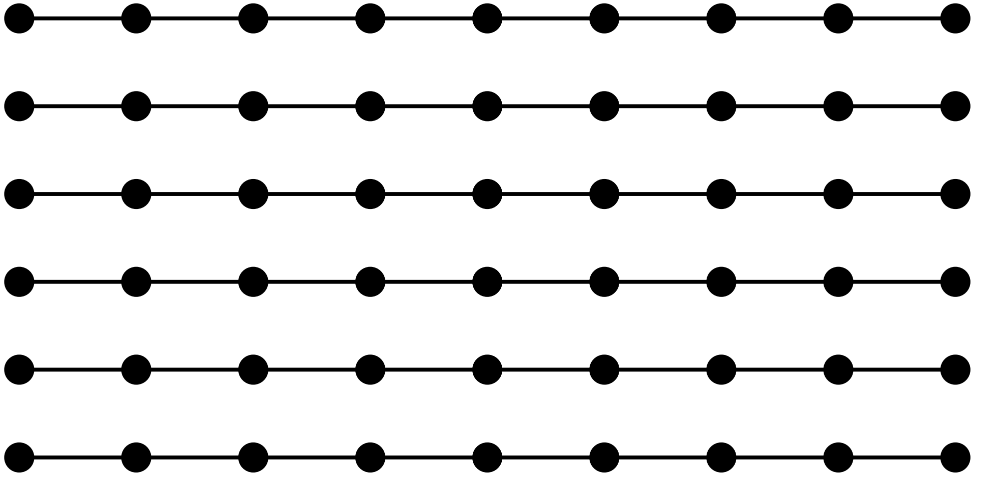


Clause gadgets check that every clause is satisfied.

Treewidth is only $n + O(1)$.

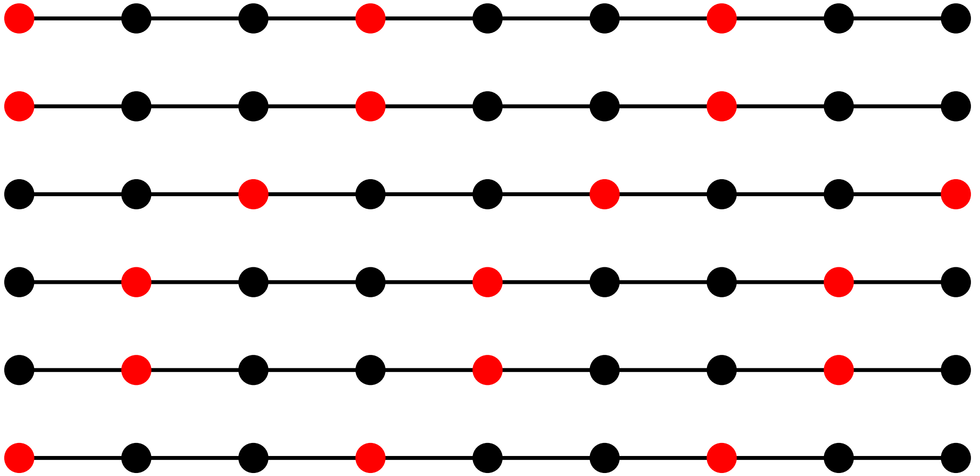
New reduction for DOMINATING SET

Now there are 3 possible optimal states for each path:



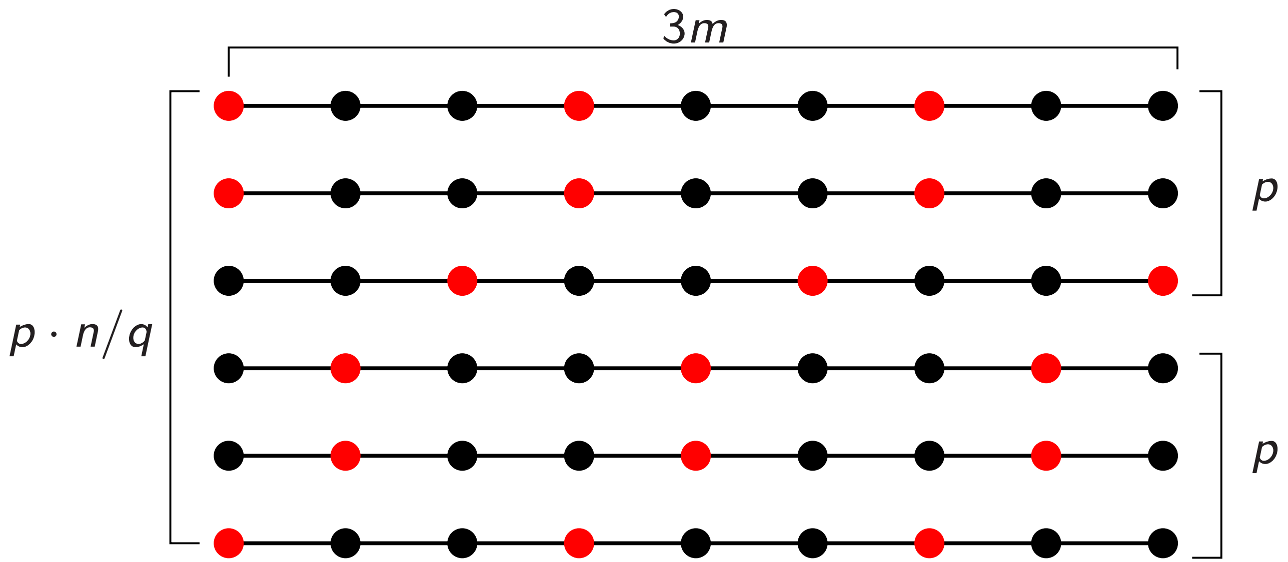
New reduction for DOMINATING SET

Now there are 3 possible optimal states for each path:



New reduction for **DOMINATING SET**

Now there are 3 possible optimal states for each path:

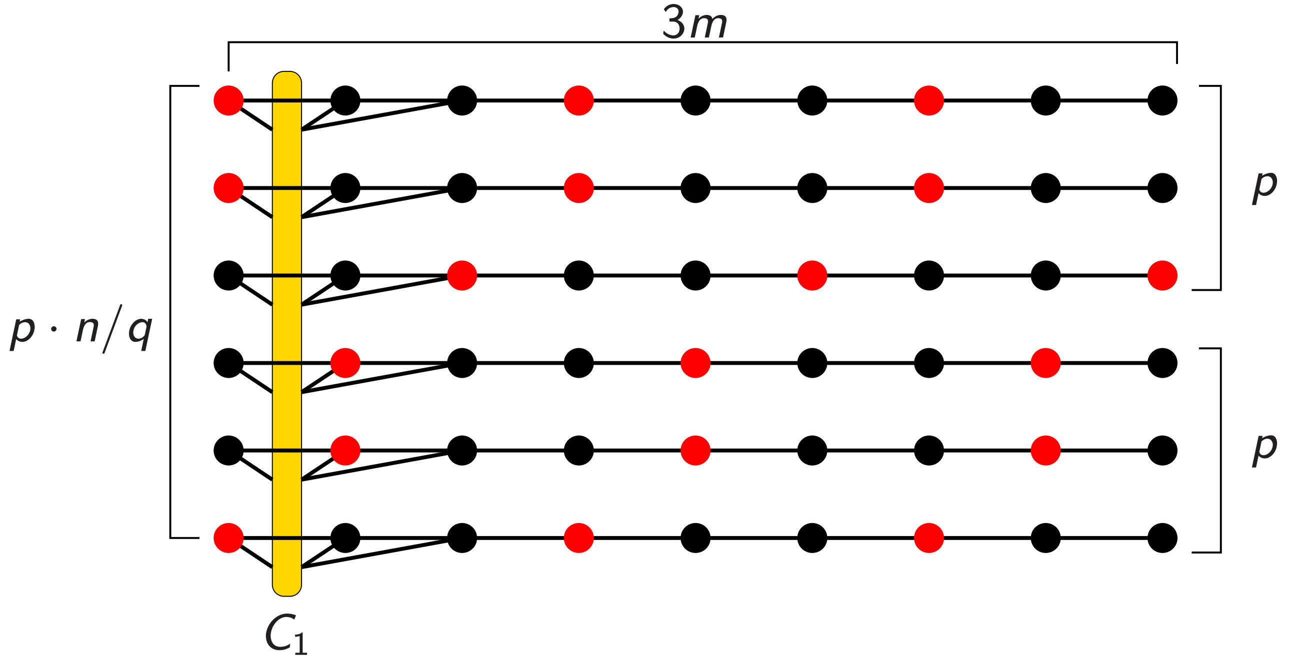


Partition variables into n/q groups of size $q = O(1)$. The 2^q possibilities for a group of variables are represented by a group of p paths, where $2^q \leq 3^p$, i.e., $p = \lceil \log_3 2^q \rceil \approx 0.631q$.

$\Rightarrow$ Treewidth is $n \cdot \log_3 2$ and the $(3-\epsilon)^w$ bound follows.

New reduction for DOMINATING SET

Now there are 3 possible optimal states for each path:

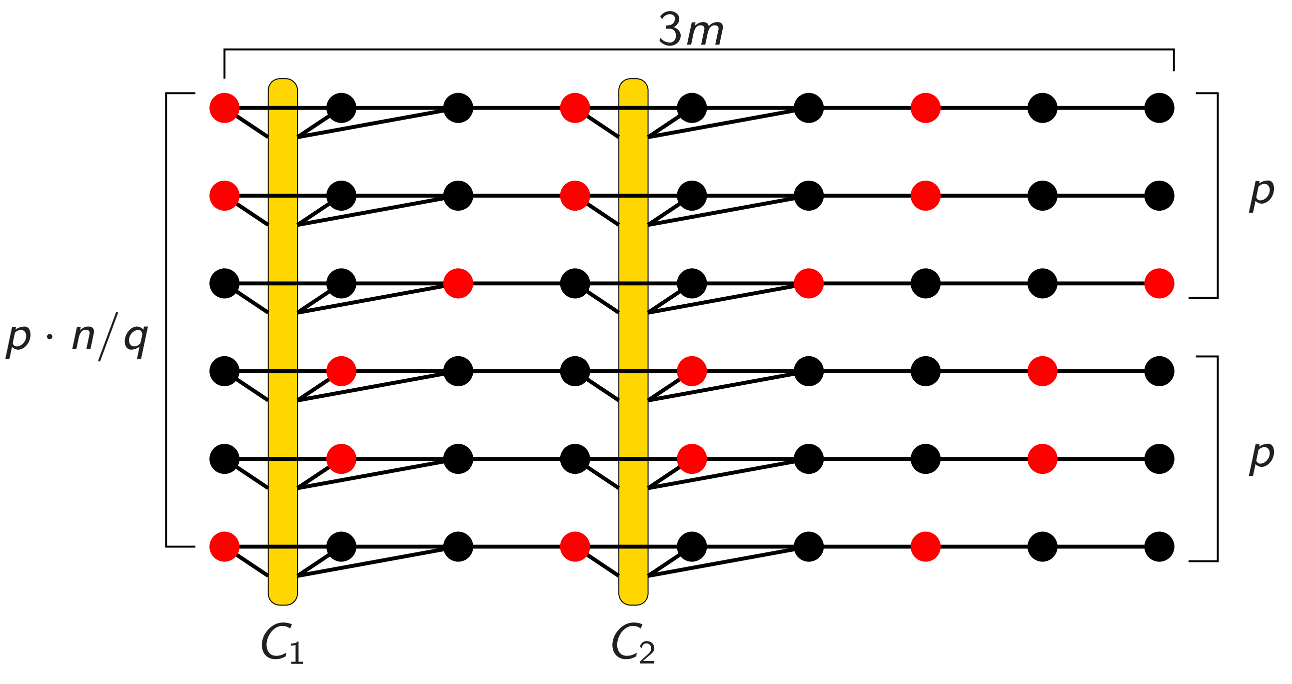


Partition variables into n/q groups of size $q = O(1)$. The 2^q possibilities for a group of variables are represented by a group of p paths, where $2^q \leq 3^p$, i.e., $p = \lceil \log_3 2^q \rceil \approx 0.631q$.

$\Rightarrow$ Treewidth is $n \cdot \log_3 2$ and the $(3 - \epsilon)^w$ bound follows.

New reduction for DOMINATING SET

Now there are 3 possible optimal states for each path:



Partition variables into n/q groups of size $q = O(1)$. The 2^q possibilities for a group of variables are represented by a group of p paths, where $2^q \leq 3^p$, i.e., $p = \lceil \log_3 2^q \rceil \approx 0.631q$.

$\Rightarrow$ Treewidth is $n \cdot \log_3 2$ and the $(3-\epsilon)^w$ bound follows.

Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal

Decompositions?

We know that INDEPENDENT SET

- Can be solved in time $2^w \cdot n$ if a tree decomposition of width w is given in the input.
- Cannot be solved in time $(2-\epsilon)^w \cdot n^{O(1)}$ for any $\epsilon > 0$ even if a tree decomposition of width w is given input.

Decompositions?

We know that INDEPENDENT SET

- Can be solved in time $2^w \cdot n$ if a tree decomposition of width w is given in the input.
- Cannot be solved in time $(2-\epsilon)^w \cdot n^{O(1)}$ for any $\epsilon > 0$ even if a tree decomposition of width w is given input.

What if the graph has treewidth w, but no tree decomposition is given in the input?

Theorem: [Bodlaender '96] Width w decomposition in time $2^{O(w^3)} \cdot n$.
Theorem: [Robertson and Seymour '95] 4-approximation in time $3^{3w} \cdot \mathrm{poly} n$.
Theorem: [Feige et al. '05] $\sqrt{\log w}$ approximation in polynomial time.

To have a $2^{(1+o(1))w}$ algorithm, we would need a $(1+o(1))$ approximation in time $2^{(1+o(1))w}$.

Conclusions

- Tight lower bounds for several basic problems on tree decompositions.
- Are there other problems where we can show that there is no $(c-\epsilon)^k \cdot n^{O(1)}$ time algorithm (where k is something else than treewidth)?
  Example: Can we solve STEINER TREE with k terminals in time $(2-\epsilon)^k \cdot n^{O(1)}$?

Conclusions

- Tight lower bounds for several basic problems on tree decompositions.
- Are there other problems where we can show that there is no $(c-\epsilon)^k \cdot n^{O(1)}$ time algorithm (where k is something else than treewidth)?
  Example: Can we solve STEINER TREE with k terminals in time $(2-\epsilon)^k \cdot n^{O(1)}$?
- Results are conditional on SETH.
  - If you believe SETH: our results are strong lower bounds.
  - If you don't believe SETH: our results show that improving the algorithms requires an improved general SAT algorithm, and hence not a graph theory/treewidth related question.