Lower bounds for hard problems

k-clique (and k-independent set) cannot be solved in $f(k)n^{o(k)}$ steps for any computable f.

Proof idea: Assume otherwise and let, e.g., $f(k) = 2^k$.

Then choose $k = \log n$ (or, in general $k = f^{-1}(n)$).

Split a graph G with n vertices into k groups of almost same size.

Build a new graph H whose vertices are valid 3-colorings of the groups. There is an edge between two vertices if their colorings are compatible.

The size of H is at most $N = k3^{n/k}$. H has a k-clique iff G is 3-colorable.

Find such a clique in time $N^{o(k)} = (k3^{n/k})^{o(k)} = 2^{o(n)}$.

We can therefore find out whether G is 3-colorable in time $2^{o(n)}$. Contradiction.

The Strong Exponential Time Hypothesis (SETH)

Let δ_r be the infimum of all δ'_r for which an algorithm exists that solves r-SAT in time $O(2^{\delta'_r n})$.

ETH: $\delta_3 > 0$

SETH: $\lim_{r \to \infty} \delta_r = 1$

SETH implies ETH. (why?)

ETH implies $\mathrm{W[1]} \neq \mathrm{FPT}$. (why?)

The faith into SETH is smaller than into ETH.

There are fewer results for SETH than for ETH.

Lower bound for algorithms on tree decompositions

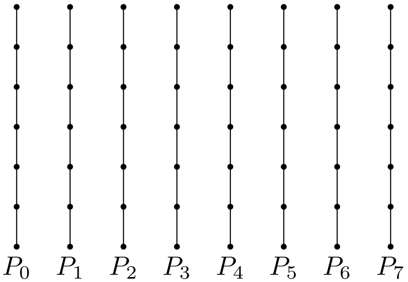
We have seen that Independent Set can be solved in time $2^k n^{O(1)}$ if the input is a tree decomposition of width k.

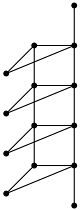
Under SETH we cannot solve this problem in time $(2-\epsilon)^k n^{O(1)}$ for any $\epsilon > 0$.

Proof idea: For a given CNF-SAT formula ϕ with n variables and m clauses construct a graph G with path-width $n+3$ and size $O(n^3 m)$.

G has an independent set of a given size iff ϕ is satifiable.

If we can find a maximal independent set in time $(2-\epsilon)^{n+3}|G|^{O(1)}$, then we solve CNF-SAT in time $(2-\epsilon)^n|\phi|^{O(1)}$ and SETH fails.





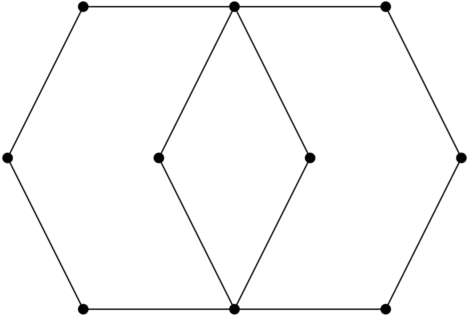
Connect gadget for every clause.

One connection has to go to an empty vertex.

Repeat $n+1$ times to avoid cheating.

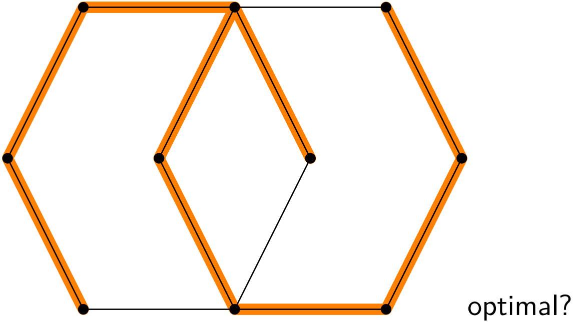
Overview

Spanning trees



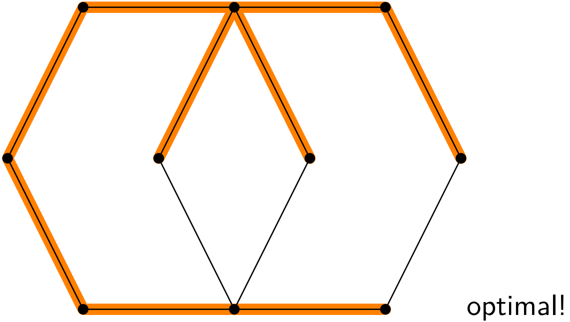
1. Minimum weight spanning tree $\rightarrow$ polynomial time
2. Maximum leaf spanning tree $\rightarrow$ NP-complete

Spanning trees



1. Minimum weight spanning tree $\rightarrow$ polynomial time
2. Maximum leaf spanning tree $\rightarrow$ NP-complete

Spanning trees



1. Minimum weight spanning tree $\rightarrow$ polynomial time
2. Maximum leaf spanning tree $\rightarrow$ NP-complete

Maximum Leaf Spanning Trees

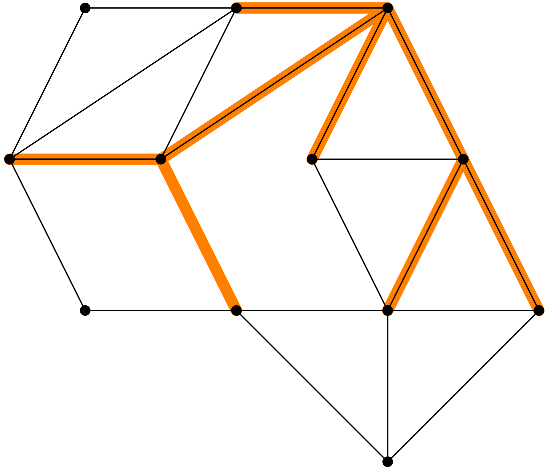
We consider this problem:

- Input: An undirected graph G and a number k
- Question: Does G contain a spanning tree with at least k leaves?

Applications: Operations research, network design

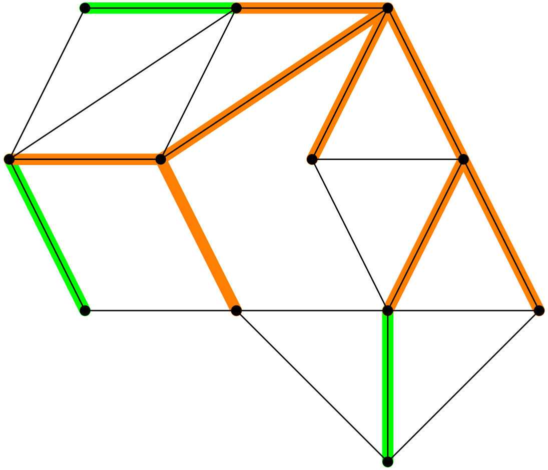
A simpler problem

How to turn a k-leaf tree into a k-leaf spanning tree:



A simpler problem

How to turn a k-leaf tree into a k-leaf spanning tree:



Known Results

APX-hard	Galbiati, Maffioli, Morzenti, 1994
2-approximation	Solis-Oba, 1998
3-approximation	Lu & Ravi, 1998
1.5-approximation (cubic)	Bonsma & Zickfeld, 2008
$O((17k)!\,(n+m))$	Bodlaender, 1993
$(2k)^{4k} n^{O(1)}$	Downey & Fellows, 1995
$O(14.23^k + n + m)$	Fellows, McCartin, Rosamond, Steege, 2000
$O(9.49^k k^3 + n^3)$	Bonsma, Brueggemann, Woeginger, 2003
$O(8.12^k k^3 + n^3)$	Estivill-Castro, Fellows, Langston, Rosamond, 2005
$O^*(1.94^n)$	Fomin, Grandoni, Kratsch, 2006
$6.75^k k^{O(1)} + n^{O(1)}$	Bonsma & Zickfeld, 2008
$4^k k^2 + n^{O(1)}$	Much simpler algorithm!

Known Results

APX-hard	Galbiati, Maffioli, Morzenti, 1994
2-approximation	Solis-Oba, 1998
3-approximation	Lu & Ravi, 1998
1.5-approximation (cubic)	Bonsma & Zickfeld, 2008
$O((17k)!\,(n+m))$	Bodlaender, 1993
$(2k)^{4k} n^{O(1)}$	Downey & Fellows, 1995
$O(14.23^k + n + m)$	Fellows, McCartin, Rosamond, Steege, 2000
$O(9.49^k k^3 + n^3)$	Bonsma, Brueggemann, Woeginger, 2003
$O(8.12^k k^3 + n^3)$	Estivill-Castro, Fellows, Langston, Rosamond, 2005
$O^*(1.94^n)$	Fomin, Grandoni, Kratsch, 2006
$6.75^k k^{O(1)} + n^{O(1)}$	Bonsma & Zickfeld, 2008
$4^k k^2 + n^{O(1)}$	Much simpler algorithm!

Known Results

APX-hard	Galbiati, Maffioli, Morzenti, 1994
2-approximation	Solis-Oba, 1998
3-approximation	Lu & Ravi, 1998
1.5-approximation (cubic)	Bonsma & Zickfeld, 2008
$O((17k)!\,(n+m))$	Bodlaender, 1993
$(2k)^{4k} n^{O(1)}$	Downey & Fellows, 1995
$O(14.23^k + n + m)$	Fellows, McCartin, Rosamond, Steege, 2000
$O(9.49^k k^3 + n^3)$	Bonsma, Brueggemann, Woeginger, 2003
$O(8.12^k k^3 + n^3)$	Estivill-Castro, Fellows, Langston, Rosamond, 2005
$O^*(1.94^n)$	Fomin, Grandoni, Kratsch, 2006
$6.75^k k^{O(1)} + n^{O(1)}$	Bonsma & Zickfeld, 2008
$4^k k^2 + n^{O(1)}$	Much simpler algorithm!

Known Results

APX-hard	Galbiati, Maffioli, Morzenti, 1994
2-approximation	Solis-Oba, 1998
3-approximation	Lu & Ravi, 1998
1.5-approximation (cubic)	Bonsma & Zickfeld, 2008
$O((17k)!\,(n+m))$	Bodlaender, 1993
$(2k)^{4k} n^{O(1)}$	Downey & Fellows, 1995
$O(14.23^k + n + m)$	Fellows, McCartin, Rosamond, Steege, 2000
$O(9.49^k k^3 + n^3)$	Bonsma, Brueggemann, Woeginger, 2003
$O(8.12^k k^3 + n^3)$	Estivill-Castro, Fellows, Langston, Rosamond, 2005
$O^*(1.94^n)$	Fomin, Grandoni, Kratsch, 2006
$6.75^k k^{O(1)} + n^{O(1)}$	Bonsma & Zickfeld, 2008
$4^k k^2 + n^{O(1)}$	Much simpler algorithm!

Known Results

APX-hard	Galbiati, Maffioli, Morzenti, 1994
2-approximation	Solis-Oba, 1998
3-approximation	Lu & Ravi, 1998
1.5-approximation (cubic)	Bonsma & Zickfeld, 2008
$O((17k)!\,(n+m))$	Bodlaender, 1993
$(2k)^{4k} n^{O(1)}$	Downey & Fellows, 1995
$O(14.23^k + n + m)$	Fellows, McCartin, Rosamond, Steege, 2000
$O(9.49^k k^3 + n^3)$	Bonsma, Brueggemann, Woeginger, 2003
$O(8.12^k k^3 + n^3)$	Estivill-Castro, Fellows, Langston, Rosamond, 2005
$O^*(1.94^n)$	Fomin, Grandoni, Kratsch, 2006
$6.75^k k^{O(1)} + n^{O(1)}$	Bonsma & Zickfeld, 2008
$4^k k^2 + n^{O(1)}$	Much simpler algorithm!

Known Results

APX-hard	Galbiati, Maffioli, Morzenti, 1994
2-approximation	Solis-Oba, 1998
3-approximation	Lu & Ravi, 1998
1.5-approximation (cubic)	Bonsma & Zickfeld, 2008
$O((17k)!\,(n+m))$	Bodlaender, 1993
$(2k)^{4k} n^{O(1)}$	Downey & Fellows, 1995
$O(14.23^k + n + m)$	Fellows, McCartin, Rosamond, Steege, 2000
$O(9.49^k k^3 + n^3)$	Bonsma, Brueggemann, Woeginger, 2003
$O(8.12^k k^3 + n^3)$	Estivill-Castro, Fellows, Langston, Rosamond, 2005
$O^*(1.94^n)$	Fomin, Grandoni, Kratsch, 2006
$6.75^k k^{O(1)} + n^{O(1)}$	Bonsma & Zickfeld, 2008
$4^k k^2 + n^{O(1)}$	Much simpler algorithm!

Known Results

APX-hard	Galbiati, Maffioli, Morzenti, 1994
2-approximation	Solis-Oba, 1998
3-approximation	Lu & Ravi, 1998
1.5-approximation (cubic)	Bonsma & Zickfeld, 2008
$O((17k)!\,(n+m))$	Bodlaender, 1993
$(2k)^{4k} n^{O(1)}$	Downey & Fellows, 1995
$O(14.23^k + n + m)$	Fellows, McCartin, Rosamond, Steege, 2000
$O(9.49^k k^3 + n^3)$	Bonsma, Brueggemann, Woeginger, 2003
$O(8.12^k k^3 + n^3)$	Estivill-Castro, Fellows, Langston, Rosamond, 2005
$O^*(1.94^n)$	Fomin, Grandoni, Kratsch, 2006
$6.75^k k^{O(1)} + n^{O(1)}$	Bonsma & Zickfeld, 2008
$4^k k^2 + n^{O(1)}$	Much simpler algorithm!

Known Results

APX-hard	Galbiati, Maffioli, Morzenti, 1994
2-approximation	Solis-Oba, 1998
3-approximation	Lu & Ravi, 1998
1.5-approximation (cubic)	Bonsma & Zickfeld, 2008
$O((17k)!\,(n+m))$	Bodlaender, 1993
$(2k)^{4k} n^{O(1)}$	Downey & Fellows, 1995
$O(14.23^k + n + m)$	Fellows, McCartin, Rosamond, Steege, 2000
$O(9.49^k k^3 + n^3)$	Bonsma, Brueggemann, Woeginger, 2003
$O(8.12^k k^3 + n^3)$	Estivill-Castro, Fellows, Langston, Rosamond, 2005
$O^*(1.94^n)$	Fomin, Grandoni, Kratsch, 2006
$6.75^k k^{O(1)} + n^{O(1)}$	Bonsma & Zickfeld, 2008
$4^k k^2 + n^{O(1)}$	Much simpler algorithm!

Known Results

APX-hard	Galbiati, Maffioli, Morzenti, 1994
2-approximation	Solis-Oba, 1998
3-approximation	Lu & Ravi, 1998
1.5-approximation (cubic)	Bonsma & Zickfeld, 2008
$O((17k)!\,(n+m))$	Bodlaender, 1993
$(2k)^{4k} n^{O(1)}$	Downey & Fellows, 1995
$O(14.23^k + n + m)$	Fellows, McCartin, Rosamond, Steege, 2000
$O(9.49^k k^3 + n^3)$	Bonsma, Brueggemann, Woeginger, 2003
$O(8.12^k k^3 + n^3)$	Estivill-Castro, Fellows, Langston, Rosamond, 2005
$O^*(1.94^n)$	Fomin, Grandoni, Kratsch, 2006
$6.75^k k^{O(1)} + n^{O(1)}$	Bonsma & Zickfeld, 2008
$4^k k^2 + n^{O(1)}$	Much simpler algorithm!

Directed Graphs

Directed Maximum Leaf Out-Tree (DMLOT) problem:

- Input: A directed graph G and a number k
- Question: Does G contain an out-tree with at least k leaves?

Directed Maximum Leaf Spanning Tree (DMLST) problem:

- Input: A directed graph G and a number k
- Question: Does G contain an out-tree with at least k leaves that spans all nodes?

Directed Graphs

Directed Maximum Leaf Out-Tree (DMLOT) problem:

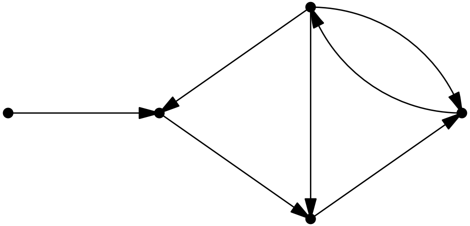
- Input: A directed graph G and a number k
- Question: Does G contain an out-tree with at least k leaves?

Directed Maximum Leaf Spanning Tree (DMLST) problem:

- Input: A directed graph G and a number k
- Question: Does G contain an out-tree with at least k leaves that spans all nodes?

Example

Open for a long time: DMLOT and DMLST in FPT?



Best directed out-tree: 3 leaves

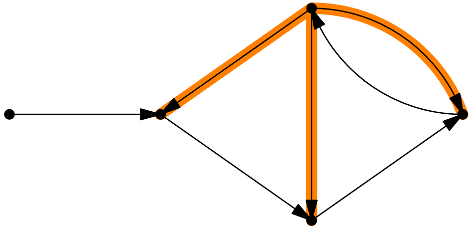
Best directed spanning tree: 1 leaf

$$\text{DMLOT} \neq \text{DMLST}$$

We cannot extend an out-tree into a spanning tree!

Example

Open for a long time: DMLOT and DMLST in FPT?



Best directed out-tree: 3 leaves

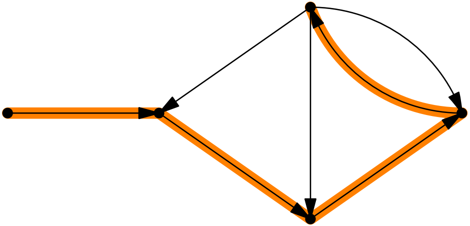
Best directed spanning tree: 1 leaf

$$\text{DMLOT} \neq \text{DMLST}$$

We cannot extend an out-tree into a spanning tree!

Example

Open for a long time: DMLOT and DMLST in FPT?



Best directed out-tree: 3 leaves

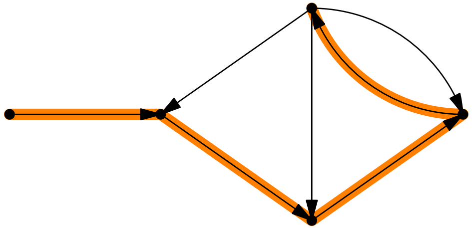
Best directed spanning tree: 1 leaf

DMLOT $\neq$ DMLST

We cannot extend an out-tree into a spanning tree!

Example

Open for a long time: DMLOT and DMLST in FPT?



Best directed out-tree: 3 leaves

Best directed spanning tree: 1 leaf

$$\text{DMLOT} \neq \text{DMLST}$$

We cannot extend an out-tree into a spanning tree!

Known results — directed graphs

Alon, Fomin, Gutin, Krivelevich, Saurabh, ICALP 2007

Theorem
G has either a k-leaf out-tree or its pathwidth is bounded by $2k^2$.

We call this a win-win scenario.

$\rightarrow$ solve DMLOT in time $c^{k^3 \log k} n^{O(1)}$.

Known results — directed graphs

Alon, Fomin, Gutin, Krivelevich, Saurabh, ICALP 2007

Theorem

G has either a k-leaf out-tree or its pathwidth is bounded by $2k^2$.

We call this a win-win scenario.

$\rightarrow$ solve DMLOT in time $c^{k^3 \log k} n^{O(1)}$.

Known results — directed graphs

Alon, Fomin, Gutin, Krivelevich, Saurabh, FST&TCS 2007

Some improvements by the same authors:

DMLOT in $c^{k^2 \log k} n^{O(1)}$ time

DMLOT in $c^{k \log k} n^{O(1)}$ time for acyclic graphs

(Improved bounds on the pathwidth.)

Results — directed graphs

Bonsma & Dorn, 2007:

DMLST in $c^{k^3 \log k} n^{O(1)}$ time

Bonsma & Dorn, 2008:

DMLST and DMLOT in $c^{k \log k} n^{O(1)}$

Now:

DMLST and DMLOT in $O(4^k nm)$

Results — directed graphs

Bonsma & Dorn, 2007:

DMLST in $c^{k^3 \log k} n^{O(1)}$ time

Bonsma & Dorn, 2008:

DMLST and DMLOT in $c^{k \log k} n^{O(1)}$

Now:

DMLST and DMLOT in $O(4^k nm)$

Results — directed graphs

Bonsma & Dorn, 2007:

DMLST in $c^{k^3 \log k} n^{O(1)}$ time

Bonsma & Dorn, 2008:

DMLST and DMLOT in $c^{k \log k} n^{O(1)}$

Now:

DMLST and DMLOT in $O(4^k nm)$

Results — directed graphs

Bonsma & Dorn, 2007:

DMLST in $c^{k^3 \log k} n^{O(1)}$ time

Bonsma & Dorn, 2008:

DMLST and DMLOT in $c^{k \log k} n^{O(1)}$

Now:

DMLST and DMLOT in $O(4^k nm)$

A simple algorithm to find k-leaf trees

Idea: Start at some node and grow a tree.

Run the following algorithms on all nodes v:

- mark v blue.
- Repeat:
  Choose a blue leaf u.

  (a) Mark it red
  OR
  (b) Connect u's outside neighbors to u and mark them blue

  if the tree has $\geq k$ leaves, then answer **YES**
  if there is no blue leaf, answer **NO**

(Outside neighbor: Neighbor that is not yet in the tree)

A simple algorithm to find k-leaf trees

Idea: Start at some node and grow a tree.

Run the following algorithms on all nodes v:

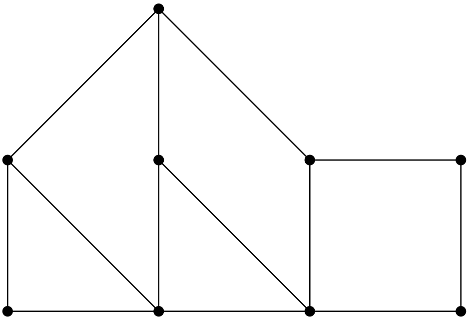
- mark v blue.
- Repeat:
  Choose a blue leaf u.

  (a) Mark it red
  OR
  (b) Connect u's outside neighbors to u and mark them blue
  (if $\deg(u) = 1$ follow the path)

  if the tree has $\geq k$ leaves, then answer **YES**
  if there is no blue leaf, answer **NO**

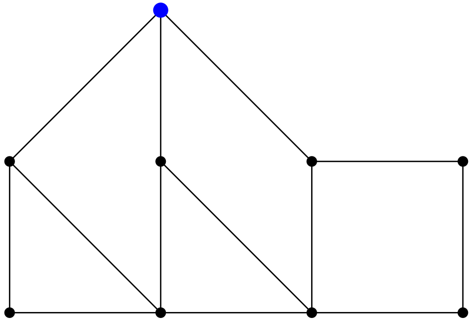
(Outside neighbor: Neighbor that is not yet in the tree)

Example



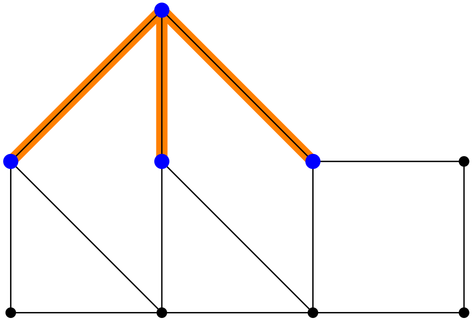
- We grow a tree.
- A blue leaf can be expanded.
- A red leaf remains a leaf.

Example



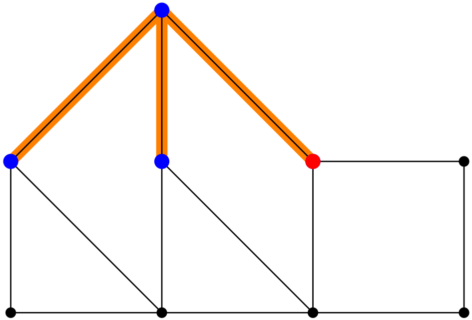
- We grow a tree.
- A blue leaf can be expanded.
- A red leaf remains a leaf.

Example



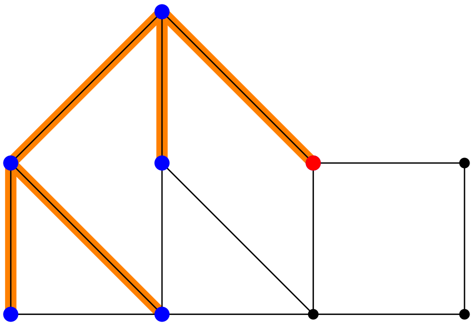
- We grow a tree.
- A blue leaf can be expanded.
- A red leaf remains a leaf.

Example



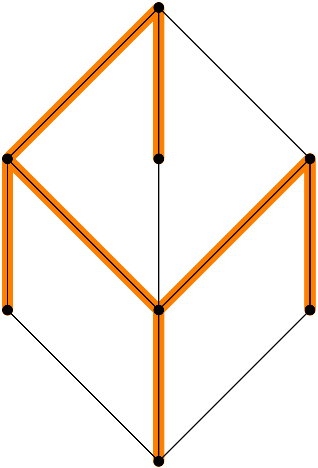
- We grow a tree.
- A blue leaf can be expanded.
- A red leaf remains a leaf.

Example



- We grow a tree.
- A blue leaf can be expanded.
- A red leaf remains a leaf.

We can't grow every tree



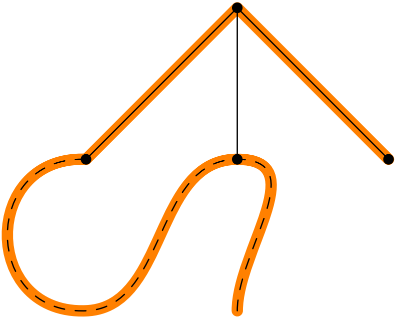
Is the algorithm correct?

Correctness

Theorem

If there is a k-leaf tree, the algorithm finds some k-leaf tree.

Proof: Modify a k-leaf spanning tree.



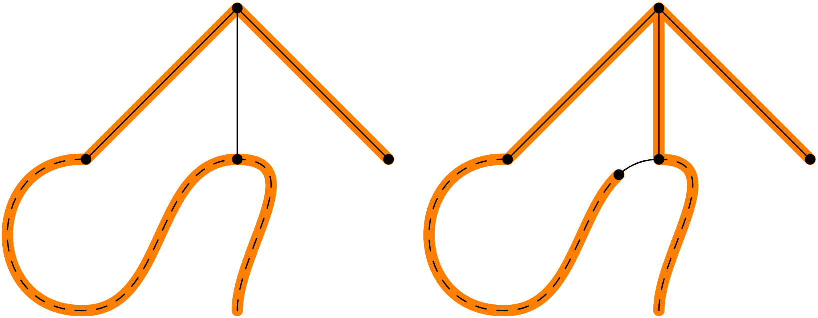
This theorem holds for directed graphs, too $\Rightarrow$ DMLOT.

Correctness

Theorem

If there is a k-leaf tree, the algorithm finds some k-leaf tree.

Proof: Modify a k-leaf spanning tree.



This theorem holds for directed graphs, too $\Rightarrow$ DMLOT.

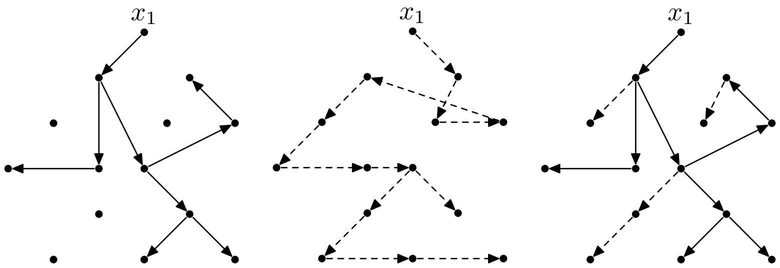
A very useful theorem

Theorem

Let G contain a directed spanning tree with root r.
Then every out-tree with root r can be extended into a spanning tree.

Proof

Use the spanning tree to extend the out-tree.



$\Rightarrow$ the algorithm solves DMLST, too.

Running time (FPT)

In every step one of the following happens:

- No recursive branch. Tree grows. Number of red and blue leaves does not change.
- A blue leaf becomes a red leaf.
- The number of blue leaves is increased.

Let r be the number of red leaves and b be the number of blue leaves.

Then the function $2r + b$ grows in each recursive call.

If $2r + b \geq 2k$, the algorithm terminates.

The recursion depth is at most $2k$ and there are at most 2^{2k} recursive calls.

Running time (FPT)

In every step one of the following happens:

- No recursive branch. Tree grows. Number of red and blue leaves does not change.
- A blue leaf becomes a red leaf.
- The number of blue leaves is increased.

Let r be the number of red leaves and b be the number of blue leaves.

Then the function $2r + b$ grows in each recursive call.

If $2r + b \geq 2k$, the algorithm terminates.

The recursion depth is at most $2k$ and there are at most 2^{2k} recursive calls.

Running time (FPT)

In every step one of the following happens:

- No recursive branch. Tree grows. Number of red and blue leaves does not change.
- A blue leaf becomes a red leaf.
- The number of blue leaves is increased.

Let r be the number of red leaves and b be the number of blue leaves.

Then the function $2r + b$ grows in each recursive call.

If $2r + b \geq 2k$, the algorithm terminates.

The recursion depth is at most $2k$ and there are at most 2^{2k} recursive calls.

Running time (FPT)

In every step one of the following happens:

- No recursive branch. Tree grows. Number of red and blue leaves does not change.
- A blue leaf becomes a red leaf.
- The number of blue leaves is increased.

Let r be the number of red leaves and b be the number of blue leaves.

Then the function $2r + b$ grows in each recursive call.

If $2r + b \geq 2k$, the algorithm terminates.

The recursion depth is at most $2k$ and there are at most 2^{2k} recursive calls.

Running time (FPT)

In every step one of the following happens:

- No recursive branch. Tree grows. Number of red and blue leaves does not change.
- A blue leaf becomes a red leaf.
- The number of blue leaves is increased.

Let r be the number of red leaves and b be the number of blue leaves.

Then the function $2r + b$ grows in each recursive call.

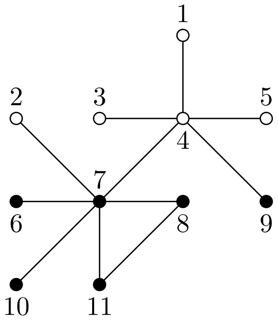
If $2r + b \geq 2k$, the algorithm terminates.

The recursion depth is at most $2k$ and there are at most 2^{2k} recursive calls.

Adding Recursion to Color-Coding

Idea:

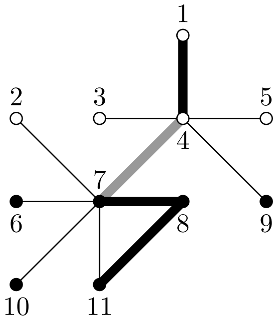
1. Randomly color G in black and white.
2. Recursively check for a black $\lceil k/2 \rceil$-node path and a white $\lfloor k/2 \rfloor$-node path that combine to form a k-node path in G.



Adding Recursion to Color-Coding

Idea:

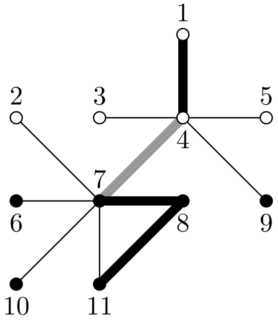
1. Randomly color G in black and white.
2. Recursively check for a black $\lceil k/2 \rceil$-node path and a white $\lfloor k/2 \rfloor$-node path that combine to form a k-node path in G.



The Algorithm for LONGEST PATH

Crucial details:

1. Try $3 \cdot 2^k$ colorings *in each call*.
2. Return *all* the $(u, v) \in V^2$ with $u \xrightarrow{k} v$ that were found.



Combine black (u, x) and white (y, v) into new (u, v) if $\{x, y\} \in E$

Error Probability

Sources of error:

1. Bad coloring: $= 1 - 2^{-k}$
2. Good coloring, error in recursion: $\leq 2^{-k} \cdot 2 \cdot p_{\lceil k/2 \rceil}$

p_k: Pr[algorithm misses needed (u, v) with $u \xrightarrow{k} v$]

Due to the $3 \cdot 2^k$ iterations, $p_k \leq \left(1 - 2^{-k} + 2^{-k+1} p_{\lceil k/2 \rceil}\right)^{3 \cdot 2^k}$.

Proof that $p_k \leq 1/4$: $p_1 = 0$, and by induction

$$\left(1 - 2^{-k} + 2^{-k+1} p_{\lceil k/2 \rceil}\right)^{3 \cdot 2^k} \leq \left(1 - 2^{-k-1}\right)^{\frac{3}{2} \cdot 2^{k+1}} \leq e^{-\frac{3}{2}} < \frac{1}{4}.$$

Error Probability

Sources of error:

1. Bad coloring: $= 1 - 2^{-k}$
2. Good coloring, error in recursion: $\leq 2^{-k} \cdot 2 \cdot p_{\lceil k/2 \rceil}$

p_k: Pr[algorithm misses needed (u, v) with $u \xrightarrow{k} v$]

Due to the $3 \cdot 2^k$ iterations, $p_k \leq \left(1 - 2^{-k} + 2^{-k+1} p_{\lceil k/2 \rceil}\right)^{3 \cdot 2^k}$.

Proof that $p_k \leq 1/4$: $p_1 = 0$, and by induction

$$\left(1 - 2^{-k} + 2^{-k+1} p_{\lceil k/2 \rceil}\right)^{3 \cdot 2^k} \leq \left(1 - 2^{-k-1}\right)^{\frac{3}{2} \cdot 2^{k+1}} \leq e^{-\frac{3}{2}} < \frac{1}{4}.$$

Error Probability

Sources of error:

1. Bad coloring: $= 1 - 2^{-k}$
2. Good coloring, error in recursion: $\leq 2^{-k} \cdot 2 \cdot p_{\lceil k/2 \rceil}$

p_k: $\Pr[$ algorithm misses needed (u, v) with $u \xrightarrow{k} v$ $]$

Due to the $3 \cdot 2^k$ iterations, $p_k \leq \left(1 - 2^{-k} + 2^{-k+1} p_{\lceil k/2 \rceil}\right)^{3 \cdot 2^k}$.

Proof that $p_k \leq 1/4$: $p_1 = 0$, and by induction

$$\left(1 - 2^{-k} + 2^{-k+1} p_{\lceil k/2 \rceil}\right)^{3 \cdot 2^k} \leq \left(1 - 2^{-k-1}\right)^{\frac{3}{2} \cdot 2^{k+1}} \leq e^{-\frac{3}{2}} < \frac{1}{4}.$$

Total Running Time

Number of recursive calls:

$$T_k \leq 3 \cdot 2^k (T_{\lceil k/2 \rceil} + T_{\lfloor k/2 \rfloor}) \leq 3 \cdot 2^{k+1} T_{\lceil k/2 \rceil}$$

Observe that

$$k + \lceil k/2 \rceil + \lceil \lceil k/2 \rceil /2 \rceil + \cdots + 1 \leq 2k + \log k.$$

Total running time:

$$O(3^{\log k} 2^{2k+2\log k}) = O(k^{\log 3} k^2 4^k) = O^*(4^k)$$

Total Running Time

Number of recursive calls:

$$T_k \leq 3 \cdot 2^k (T_{\lceil k/2 \rceil} + T_{\lfloor k/2 \rfloor}) \leq 3 \cdot 2^{k+1} T_{\lceil k/2 \rceil}$$

Observe that

$$k + \lceil k/2 \rceil + \lceil \lceil k/2 \rceil / 2 \rceil + \cdots + 1 \leq 2k + \log k.$$

Total running time:

$$O(3^{\log k} 2^{2k+2\log k}) = O(k^{\log 3} k^2 4^k) = O^*(4^k)$$

Total Running Time

Number of recursive calls:

$$T_k \leq 3 \cdot 2^k (T_{\lceil k/2 \rceil} + T_{\lfloor k/2 \rfloor}) \leq 3 \cdot 2^{k+1} T_{\lceil k/2 \rceil}$$

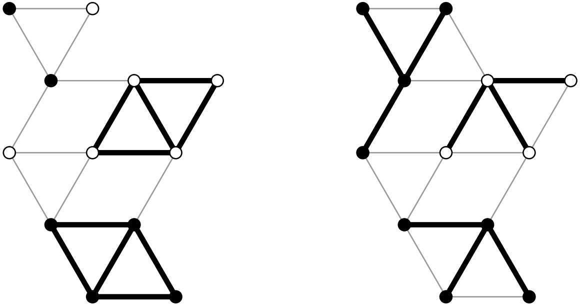
Observe that

$$k + \lceil k/2 \rceil + \lceil \lceil k/2 \rceil / 2 \rceil + \cdots + 1 \leq 2k + \log k.$$

Total running time:

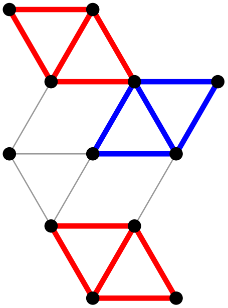
$$O(3^{\log k} 2^{2k+2\log k}) = O(k^{\log 3} k^2 4^k) = O^*(4^k)$$

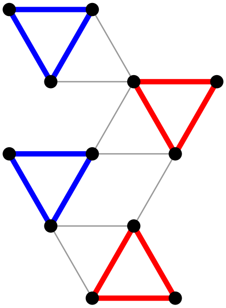
Divide-and-Color for Packing Problems



H-Graph Packing $\quad\quad$ $K_{1,3}$-Packing

Divide-and-Color for Packing Problems





H-GRAPH EDGE-PACKING   EDGE-DISJ. TRIANGLE PACKING

Summary: Randomized Divide-and-Color

Graph Problem	**Runtime Bound**
Longest Path	$O^*(4^k)$
H-Graph Packing	$O^*(2^{2(h-1)k})$, $h := \lvert V[H] \rvert$
H-Graph Edge-Packing	$O^*(2^{2(h-1)k})$, $h := \lvert E[H] \rvert$
Edge-Disjoint Triangle Packing	$O^*(2^{4k})$
$K_{1,s}$-*Packing*	$O^*(2^{2sk})$

... with exponentially small error probability.

Kernelization on sparse graph classes

- Framework for planar graphs

  Guo and Niedermeier: *Linear problem kernels for NP-hard problems on planar graphs*
- Meta-result for graphs of bounded genus

  Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh and Thilikos: *(Meta) Kernelization*
- Meta-result for graphs excluding a fixed graph as a minor

  Fomin, Lokshtanov, Saurabh and Thilikos: *Bidimensionality and kernels*
- Here: Meta-result for graphs excluding a fixed graph as a topological minor

FPT algorithms for $\mathcal{F}$-*Deletion*

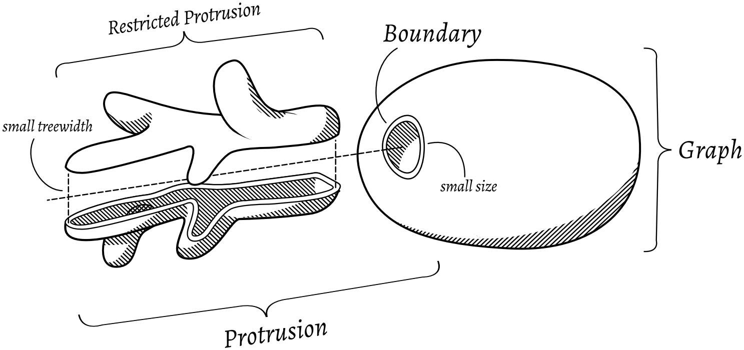
The $\mathcal{F}$-Deletion problem:

Input: A graph G, an integer k

Question: Is there a set $X \subseteq V(G)$ of size at most k such that $G - X$ contains no graph from $\mathcal{F}$ as a minor?

- Many special results, e.g. $\mathcal{F} = \{K_4\}$
- $\mathcal{F}$ contains a planar graph: FPT by Robertson-Seymour
  Fellows and Langston: Nonconstructive tools for proving polynomial-time decidability
- $2^{O(k \log k)} n^2$-Algorithm for *Planar-$\mathcal{F}$-Deletion*, later improved to $2^{O(k)} n^2$ if $\mathcal{F}$ contains only connected graphs
  Fomin, Lokshtanov, Misra and Saurabh: Nearly optimal FPT algorithms for Planar-$\mathcal{F}$-Deletion / Planar-$\mathcal{F}$-Deletion:

Protrusion

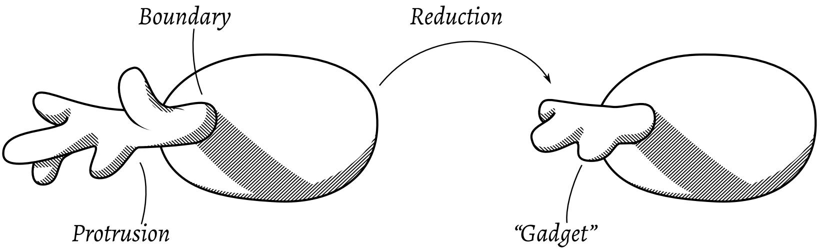


Definition

$X \subseteq V(G)$ is a *t-protrusion* if

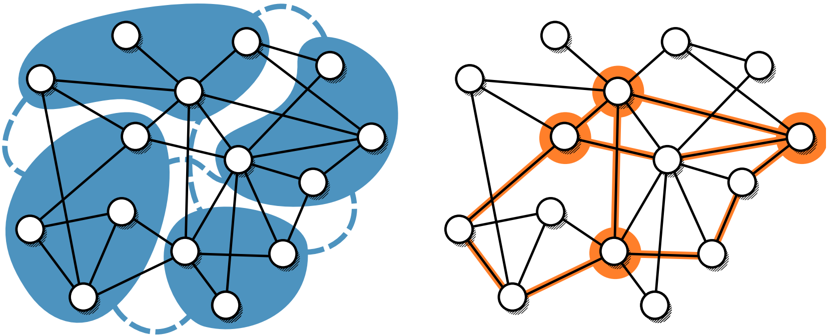
1. $|\partial(X)| = |N(X) \setminus X| \leq t$ (small boundary)
2. $\mathbf{tw}(G[X]) \leq t$ (small treewidth)

Protrusion replacement



- We want to replace a large protrusion by something smaller
- Possible if problem has *finite integer index*
- Recursive structure of graphs of small treewidth (i.e. protrusion) helps
- Lots of technicalities omitted...

Minors, top-minors

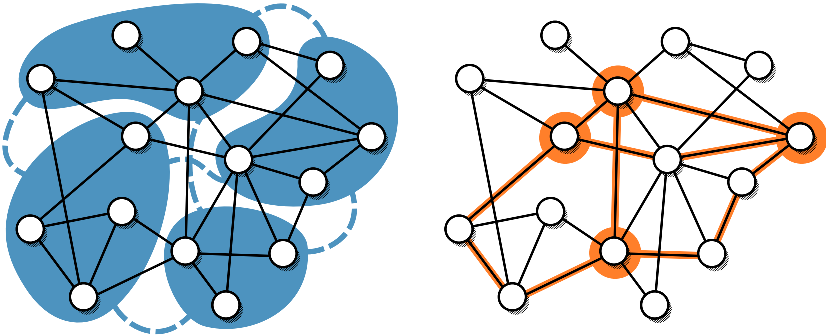


Graphs excluding a fixed Minor/Top-Minor:

- d-degenerate (d depends on the excluded graph)
- closed under taking minors/top-minors

$\Rightarrow$ every minor/top-minor *also* d-degenerate

Minors, top-minors

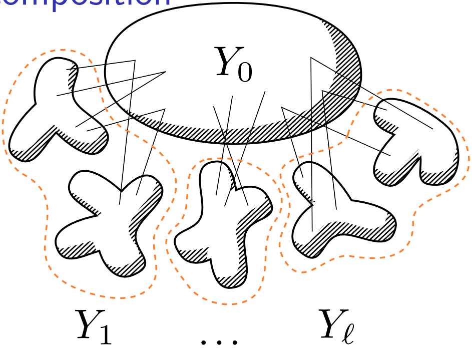


Graphs excluding a fixed Minor/Top-Minor:

- d-degenerate (d depends on the excluded graph)
- closed under taking minors/top-minors

$\Rightarrow$ every minor/top-minor *also* d-degenerate

Protrusion decomposition



(α, t)-*Protrusion decomposition* is a partition

$V = Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ such that:

1. for $1 \leq i \leq \ell$, $N(Y_i) \subseteq Y_0$
2. $\ell \leq \alpha$ and $|Y_0| \leq \alpha$
3. for $1 \leq i \leq \ell$, $Y_i \cup N(Y_i)$ is a t-protrusion

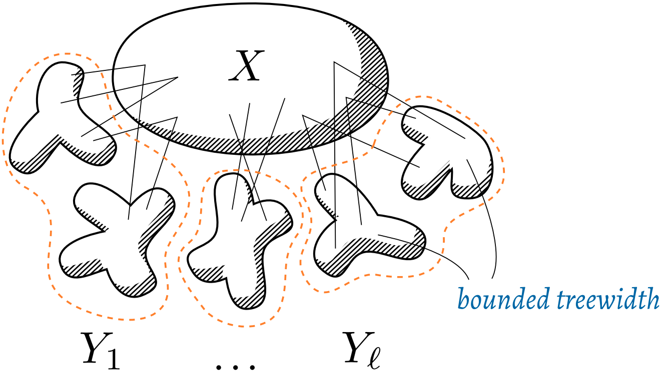
...in H-topological minor free graphs

Lemma

Let G exclude H as a topological minor and let $X \subseteq V(G)$ be such that $\mathbf{tw}(G - X) \leq t$. Then G has a $(O(|X|), 2t + |H|)$-protrusion decomposition.

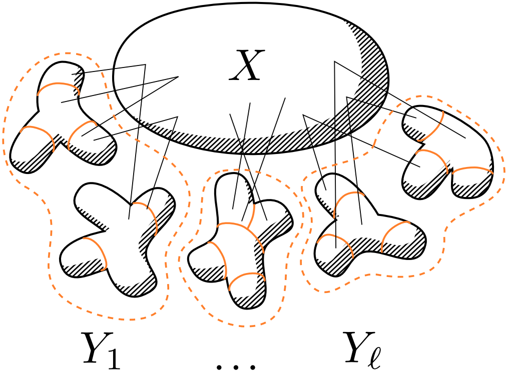
- Can be computed in linear time if X is given
- X is called a *treewidth-t modulator*

Proof sketch



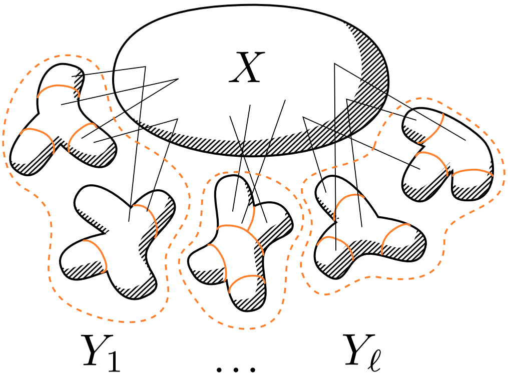
- Given X such that $\mathbf{tw}(G - X) \leq t$
- Group components of $G - X$ by respective neighbourhood in X and obtain $Y_1, \ldots, Y_\ell$

Proof sketch



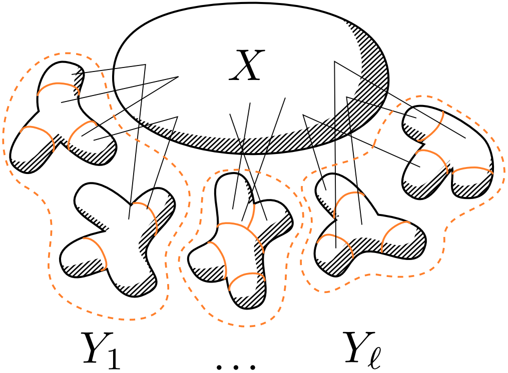
- From bottom up, mark bags whose subtree induces component with more than $|H|$ neighbours in X
- Number of such bags at most linear $|X|$: otherwise we can construct $K_{|H|}$ and thus H as a top. minor

Proof sketch



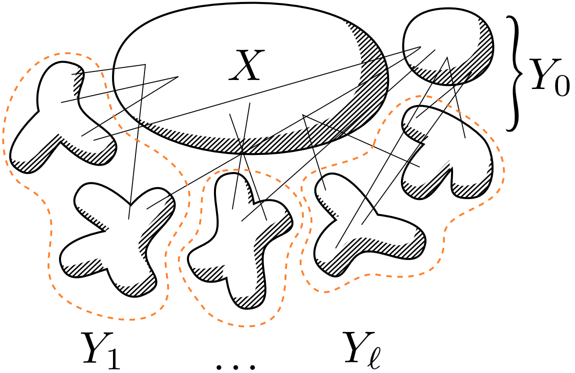
- From bottom up, mark bags whose subtree induces component with more than $|H|$ neighbours in X
- Number of such bags at most linear $|X|$: otherwise we can construct $K_{|H|}$ and thus H as a top. minor

Proof sketch



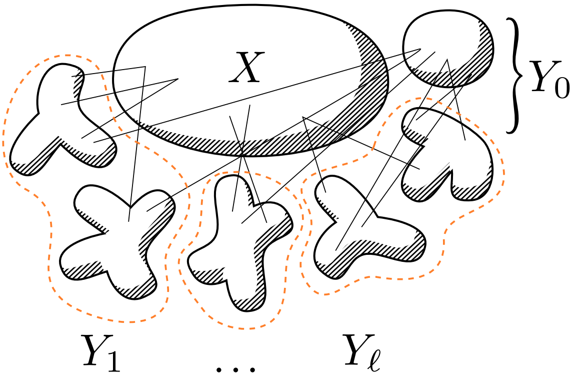
- From bottom up, mark bags whose subtree induces component with more than $|H|$ neighbours in X
- Number of such bags at most linear $|X|$: otherwise we can construct $K_{|H|}$ and thus H as a top. minor

Proof sketch



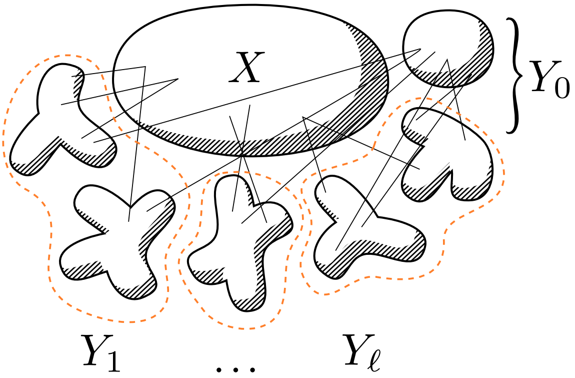
- Add content of bags to X to obtain Y_0, by previous observations $|Y_0| = O(|X|)$
- LCA marking ensures that now $|N(Y_i)| \leq 2t + |H|$ for $1 \leq i \leq \ell$

Proof sketch



- Add content of bags to X to obtain Y_0, by previous observations $|Y_0| = O(|X|)$
- LCA marking ensures that now $|N(Y_i)| \leq 2t + |H|$ for $1 \leq i \leq \ell$

Proof sketch



- Add content of bags to X to obtain Y_0, by previous observations $|Y_0| = O(|X|)$
- LCA marking ensures that now $|N(Y_i)| \leq 2t + |H|$ for $1 \leq i \leq \ell$

The theorem

Theorem

Fix a graph H. Let Π be a parameterized graph problem on the class of H-topological-minor-free graphs that is treewidth-bounding[a] and has finite integer index[b]. Then Π admits a linear kernel.

a) A parameterized graph problem is *treewidth-bounding* if every yes-instance contains a $O(k)$-sized treewidth-t-modulator for some fixed t.

b) Also required by all previous results

- Holds for e.g. *Feedback Vertex Set, Chordal Vertex Deletion, (Proper) Interval Vertex Deletion, Cograph Vertex Deletion, Edge Dominating Set, Connected Vertex Cover*

The theorem

Theorem

Fix a graph H. Let Π be a parameterized graph problem on the class of H-topological-minor-free graphs that is treewidth-bounding[a] and has finite integer index[b]. Then Π admits a linear kernel.

a) A parameterized graph problem is *treewidth-bounding* if every yes-instance contains a $O(k)$-sized treewidth-t-modulator for some fixed t

b) Also required by all previous results

- Holds for e.g. *Feedback Vertex Set*, *Chordal Vertex Deletion*, *(Proper) Interval Vertex Deletion*, *Cograph Vertex Deletion*, *Edge Dominating Set*, *Connected Vertex Cover*

The theorem

Theorem

Fix a graph H. Let Π be a parameterized graph problem on the class of H-topological-minor-free graphs that is treewidth-bounding[a] and has finite integer index[b]. Then Π admits a linear kernel.

a) A parameterized graph problem is *treewidth-bounding* if every yes-instance contains a $O(k)$-sized treewidth-t-modulator for some fixed t

b) Also required by all previous results

- Holds for e.g. *Feedback Vertex Set*, *Chordal Vertex Deletion*, *(Proper) Interval Vertex Deletion*, *Cograph Vertex Deletion*, *Edge Dominating Set*, *Connected Vertex Cover*

The theorem

Theorem

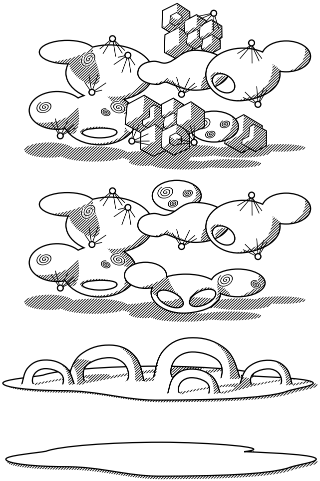
Fix a graph H. Let Π be a parameterized graph problem on the class of H-topological-minor-free graphs that is treewidth-bounding[a] and has finite integer index[b]. Then Π admits a linear kernel.

a) A parameterized graph problem is *treewidth-bounding* if every yes-instance contains a $O(k)$-sized treewidth-t-modulator for some fixed t

b) Also required by all previous results

- Holds for e.g. *Feedback Vertex Set*, *Chordal Vertex Deletion*, *(Proper) Interval Vertex Deletion*, *Cograph Vertex Deletion*, *Edge Dominating Set*, *Connected Vertex Cover*

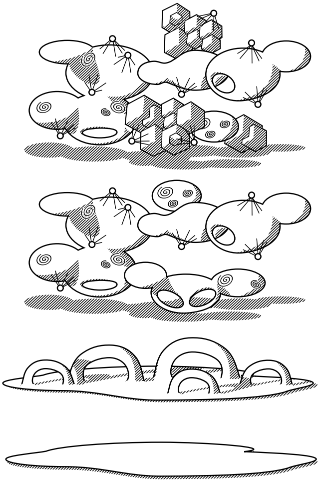
Treewidth-bounding?



H-Topological-Minor-Free	*Treewidth-bounding*
$\cup$	
H-Minor-Free	*Bidimensional* *+ separation property*
$\cup$	
Bounded Genus	*Quasi-compact*
$\cup$	
Planar	*“Distance-property”*

Ealier properties imply treewidth-bounding!

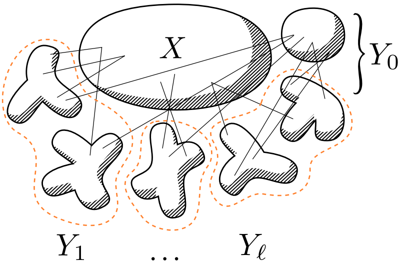
Treewidth-bounding?



H-Topological-Minor-Free	*Treewidth-bounding*
$\cup$	
H-Minor-Free	*Bidimensional* *+ separation property*
$\cup$	
Bounded Genus	*Quasi-compact*
$\cup$	
Planar	*“Distance-property”*

Ealier properties imply treewidth-bounding!

Proof idea



- Problem is treewidth-bounding: there exists a treewidth-t-modulator (if it is a yes-instance)
- Exhaustively reduce all $(2t + |H|)$-protrusions in polynomial time

$\Rightarrow$ Every such protrusion has now constant size

- There exists a $(O(|X|), 2t + |H|)$-protusion-decomposition:

The theorem

Planar-$\mathcal{F}$-Deletion:

Input: A graph G, an integer k

Problem: Is there a set $X \subseteq V(G)$ of size at most k such that $G - X$ contains no graph from $\mathcal{F}$ as a minor?

Theorem

Let $\mathcal{F}$ be a fixed finite family of graphs containing at least one planar graph. There exists an algorithm to solve Planar-$\mathcal{F}$-Deletion in time $2^{O(k)} \cdot n^2$.

Considerations

- No finite state property, because $\mathcal{F}$ can contain disconnected graphs

$\Rightarrow$ Protrusion reduction not possible!

- As $\mathcal{F}$ contains a planar graph, a solution X will fullfill $\mathbf{tw}(G - X) \leq t$ for some constant t

$\Rightarrow$ Use iterative compression to have solution X' that works as a treewdith modulator

- But: We are working on general graphs! Bounds for H-(topological)-minor-free graphs do not apply!

Algorithm outline

From iterative compression: got solution X, $|X| \leq k+1$ and want disjoint solution $\tilde{X}$, $|\tilde{X}| \leq k$.

- Given X, obtain $(|X|, t)$-protrusion-decomposition $Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$, where t depends on $\mathcal{F}$
- Guess intersections I of $\tilde{X}$ with Y_0 (in time $2^{O(k)}$)
- New solution $\tilde{X}$ can intersect at most $\leq k$ clusters

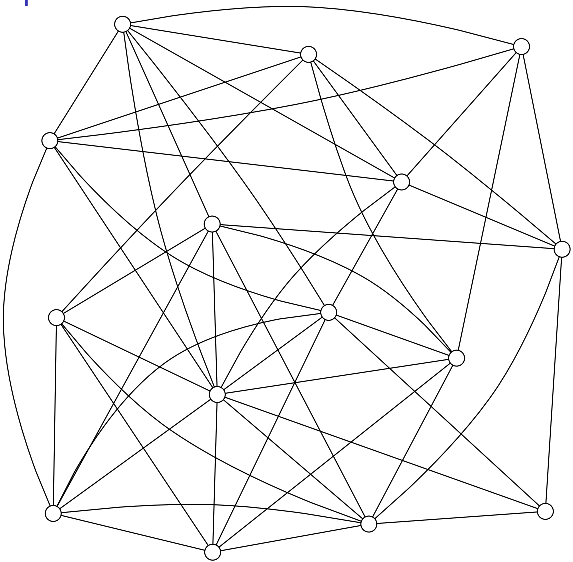
$\Rightarrow$ Disregarding those $\leq k$ clusters, $G - I$ is H-minor-free!

$\Rightarrow$ $\ell = O(k)$ or we have a no-instance

Using a the finite state property of solutions sets inside the protrusions we can enumerate all necessary vertex sets in $2^{O(k)}$ time (quite technical)

Overview

Random Graphs



Erdős–Rényi graph $G(n, 1/2)$: Every edge has probability $1/2$

Random Graphs

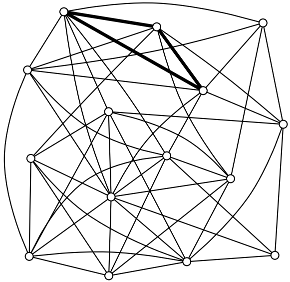
Random graphs have interesting properties

- many hard problems become easy
- zero-one laws
- hard to prove that hard problems are hard

Overview of this section

- Dominating Set as hard as FO-model checking on $G(n, 1/2)$
- Complexity does not change for $G(n, p)$ for rational p
- Finding k-rows whose AND is the zero vector also hard
- Finding k-rows whose XOR is the zero vector is easy (the Even Set problem)

Finding Triangles is Easy



The best algorithm in the worst-case:

$O(n^\omega)$ to test for a triangle

What is the running time for $G(n, 1/2)$?

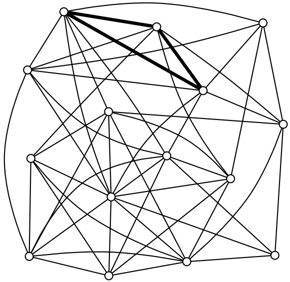
To be more precise: The average running time?

Answer:

The average running time is $O(1)$.

Reason: Abundance of witnesses.

Finding Triangles is Easy



The best algorithm in the worst-case:

$O(n^{\omega})$ to test for a triangle

What is the running time for $G(n, 1/2)$?

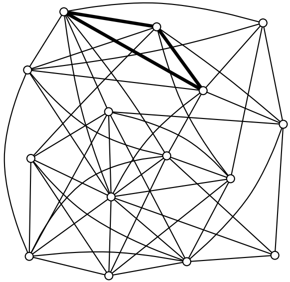
To be more precise: The average running time?

Answer:

The average running time is $O(1)$.

Reason: Abundance of witnesses.

Finding Triangles is Easy



The best algorithm in the worst-case:

$O(n^\omega)$ to test for a triangle

What is the running time for $G(n, 1/2)$?

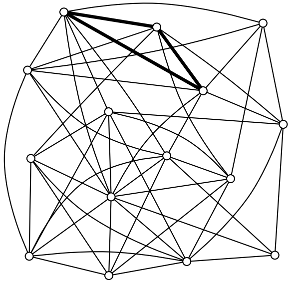
To be more precise: The average running time?

Answer:

The average running time is $O(1)$.

Reason: Abundance of witnesses.

Finding Cliques is Easy



Finding a k-clique:

NP-complete in the worst case.

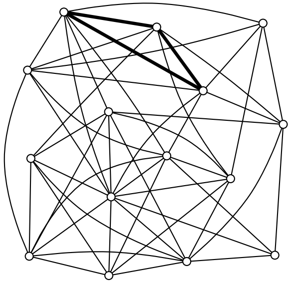
Quasipolynomial time on average.

Parameterized complexity:

W[1]-complete, cannot be solved in $n^{o(k)}$ under ETH.

FPT on average [Founoulakis, Friedrich, Hermelin].

Finding Cliques is Easy



Finding a k-clique:

NP-complete in the worst case.

Quasipolynomial time on average.

Parameterized complexity:

W[1]-complete, cannot be solved in $n^{o(k)}$ under ETH.

FPT on average [Founoulakis, Friedrich, Hermelin].

First-Order Logic on Graphs

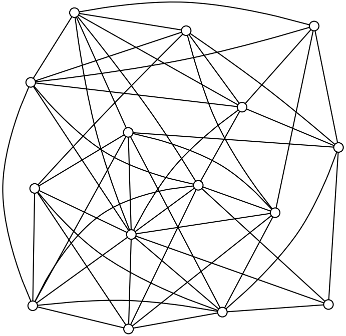
Many problems can be expressed using logic.

Quantification over vertices $\exists$, $\forall$, adjacency $\sim$,

equality $=$, and $\wedge$, or $\vee$, not $\neg$.

A graph G contains a k-clique iff

$$G \models \exists x_1 \exists x_2 \ldots \exists x_k \bigwedge_{i \neq j} x_i \sim x_j,$$



a dominating set of size k iff

$$G \models \exists x_1 \exists x_2 \ldots \exists x_k \forall y \bigvee_i (x_i \sim y \vee x_i = y).$$

First-Order Logic on Graphs

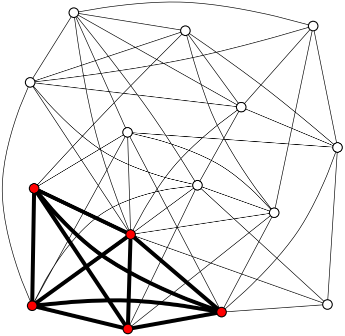
Many problems can be expressed using logic.

Quantification over vertices $\exists$, $\forall$, adjacency $\sim$,

equality $=$, and $\wedge$, or $\vee$, not $\neg$.

A graph G contains a k-clique iff

$$G \models \exists x_1 \exists x_2 \dots \exists x_k \bigwedge_{i \neq j} x_i \sim x_j,$$



a dominating set of size k iff

$$G \models \exists x_1 \exists x_2 \dots \exists x_k \forall y \bigvee (x_i \sim y \vee x_i = y).$$

First-Order Logic on Graphs

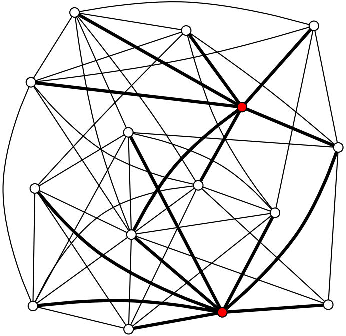
Many problems can be expressed using logic.

Quantification over vertices $\exists$, $\forall$, adjacency $\sim$,

equality $=$, and $\wedge$, or $\vee$, not $\neg$.

A graph G contains a k-clique iff

$$G \models \exists x_1 \exists x_2 \ldots \exists x_k \bigwedge_{i \neq j} x_i \sim x_j,$$



a dominating set of size k iff

$$G \models \exists x_1 \exists x_2 \ldots \exists x_k \forall y \bigvee_i \left(x_i \sim y \vee x_i = y \right).$$

Main result: Problems that are equivalent on average

p-DOMINATING SET

Input:	A graph G and $k \in \mathbf{N}$.
Parameter:	k
Problem:	Is there a dominating set of size $\leq k$ for G?

p-MATRIX($\wedge$)

Input:	A boolean matrix $M \in \{0,1\}^{n \times n}$ and $k \in \mathbf{N}$.
Parameter:	k
Problem:	Are there k rows in M whose logical AND is the zero vector?

p-MC(FO)

Proof outline

$$\boxed{\text{Solving } p\text{-}\textsc{Dominating Set} \text{ on } G(n, 1/2)}$$

$\downarrow$ Lemma **??**

$$\boxed{\text{Solving } p\text{-}\textsc{Matrix}(\wedge) \text{ on uniformly distributed square matrices}}$$

$\downarrow$ Lemma **??**

$$\boxed{\text{Solving } (G, \chi) \models \phi'' \text{ on } G(n, 1/2)}$$

$\downarrow$ Lemma **??**

$$\boxed{\text{Solving } (G, \chi) \models \phi' \text{ on } G(n, 1/2)}$$

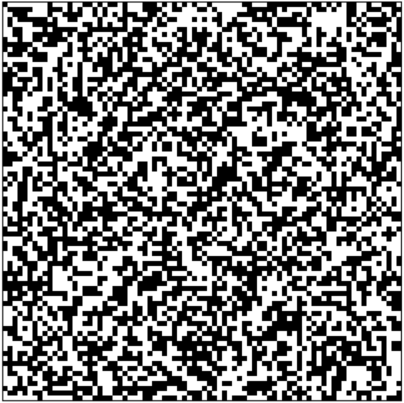
$\downarrow$ Lemma **??**

$$\boxed{\text{Solving } G \models \phi \text{ on } G(n, 1/2)}$$

$\downarrow$ Lemma **??**

p-Dominating Set$\rightarrow$ p-Matrix$(\wedge)$

"Are there k rows in M whose logical AND is the zero vector?"

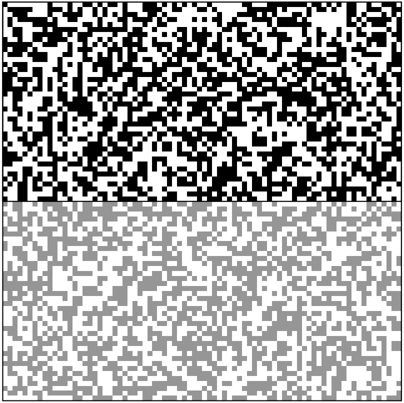


"Are there k rows in $\bar{M}$ whose logical OR is the one vector?"

Is there a directed dominating set in the graph with adjacency

p-Dominating Set$\rightarrow$ p-Matrix($\wedge$)

"Are there k rows whose logical OR is the one vector?"



Without loss of generality:

Search only in the upper part.

p-Dominating Set $\to$ p-Matrix($\wedge$)

"Are there k rows in $\bar{M}$ whose logical OR is the one vector?"

$$\bar{M} = \begin{pmatrix} A & B & C & D & E \end{pmatrix}$$

Construct an undirected graph $G = (V, E)$ with adjacency matrix

$$M' = \begin{pmatrix} A' & B & C^T \\ B^T & A'' & D \\ C & D^T & E' \end{pmatrix}$$

A', A'', E' are symmetric, build from A and E.

1. If M is random, then G is a random graph.
2. If M contains k rows whose logical OR is $\mathbf{1}$, then M' contains $3k$ such rows.
3. That means that G has dominating set of size $3k$.

Proof outline

$\boxed{\text{Solving } p\text{-}\textsc{Dominating Set} \text{ on } G(n,1/2)}$

$\downarrow$ Lemma **??**

$\boxed{\text{Solving } p\text{-}\textsc{Matrix}(\wedge) \text{ on uniformly distributed square matrices}}$

$\downarrow$ Lemma **??**

$\boxed{\text{Solving } (G,\chi) \models \phi'' \text{ on } G(n,1/2)}$

$\downarrow$ Lemma **??**

$\boxed{\text{Solving } (G,\chi) \models \phi' \text{ on } G(n,1/2)}$

$\downarrow$ Lemma **??**

$\boxed{\text{Solving } G \models \phi \text{ on } G(n,1/2)}$

$\downarrow$ Lemma **??**

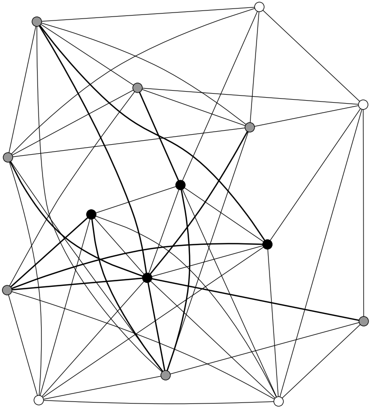
Three special FO-formulas

A vertex u can have a color $\chi(u)$.

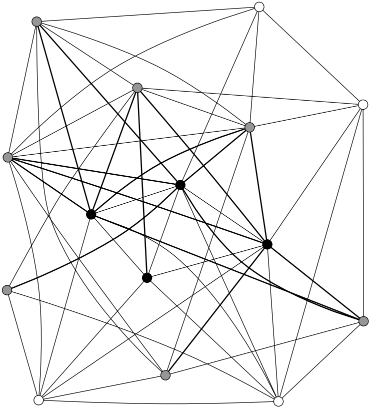
$$\phi \equiv \forall\bar{x}\forall\bar{y}\exists z\Big(\bigwedge_{i,j=1}^{k} x_i \neq y_j \to \bigwedge_{i=1}^{k}(x_i \sim z \wedge y_i \not\sim z)\Big)$$

$$\begin{aligned}\phi' \equiv \forall\bar{x}\forall\bar{y}\exists z\Bigg(\bigwedge_{i=1}^{k}\Big(\chi(x_i) = black \wedge \chi(y_i) = white\Big)\\ \to \chi(z) = gray \wedge \bigwedge_{i=1}^{k}(x_i \sim z \wedge y_i \not\sim z)\Bigg)\end{aligned}$$

$$\begin{aligned}\phi'' \equiv \forall\bar{x}\forall\bar{y}\exists z\Bigg(\bigwedge_{i=1}^{k}\Big(\chi(x_i) = black \wedge \chi(y_i) = white\Big)\\ \to \chi(z) = gray \wedge \bigwedge_{i=1}^{k}(x_i \not\sim z \wedge y_i \not\sim z)\Bigg)\end{aligned}$$



ϕ''

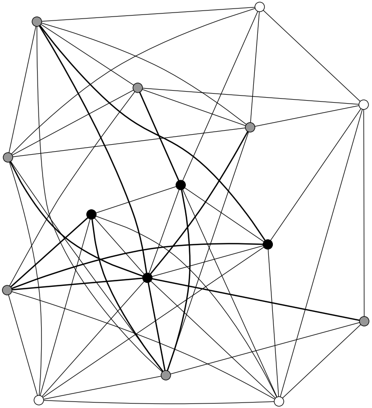


ϕ'

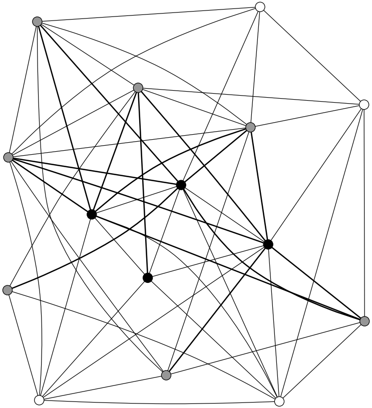
Flip edges between gray and black vertices.

$\neg\phi'$: Are there *k* black and *k* white vertices such that every gray vertex is adjacent to a black or non-adjacent to a white vertex.

$\neg\phi''$: Are there *k* black and *k* white vertices such that every gray



ϕ''



ϕ'

Flip edges between gray and black vertices.

$\neg\phi'$: Are there *k* black and *k* white vertices such that every gray vertex is adjacent to a black or non-adjacent to a white vertex.

ϕ: “Can I always find vertices that attach in every possible way to

Zero-One Laws

Famous result about random graphs:

Let ψ be an arbitrary FO-sentence. Then

$$\lim_{n\to\infty} \Pr[G(n,1/2) \models \psi] \in \{0,1\}.$$

One ingredient in the proof:

Let Φ be the extension axioms, i.e.,

$$\forall \bar{x} \forall \bar{y} \exists z \Big(\bigwedge_{i,j=1}^{k} x_i \neq y_j \to \bigwedge_{i=1}^{k} (x_i \sim z \wedge y_i \not\sim z) \Big).$$

Then $\Phi \models \psi$ or $\Phi \models \neg\psi$.

Zero-One Laws

Let ψ be an arbitrary FO-sentence. Then

$$\lim_{n \to \infty} \Pr[G(n, 1/2) \models \psi] \in \{0, 1\}.$$

Given ψ it is PSPACE-complete to decide what the limit is [Etienne Grandjean, 1983].

We can solve p-$\mathrm{MC}(\mathrm{FO})$ using p-Dominating Set as follows

1. Enumerate all possible proofs $\Phi' \vdash \psi$ and $\Phi' \vdash \neg\psi$ for growing $\Phi' \subset \Phi$ until you find a valid one.
2. Then find out if $G \models \Phi'$.
3. If yes, we know the answer.
4. If no, run a slow $n^{|\psi|}$ algorithm.

Proof outline

$$\boxed{\text{Solving } p\text{-}\textsc{Dominating Set} \text{ on } G(n, 1/2)}$$

$$\downarrow *$$

$$\boxed{\text{Solving } p\text{-}\textsc{Matrix}(\wedge) \text{ on uniformly distributed square matrices}}$$

$$\downarrow *$$

$$\boxed{\text{Solving } (G, \chi) \models \phi'' \text{ on } G(n, 1/2)}$$

$$\downarrow *$$

$$\boxed{\text{Solving } (G, \chi) \models \phi' \text{ on } G(n, 1/2)}$$

$$\downarrow *$$

$$\boxed{\text{Solving } G \models \phi \text{ on } G(n, 1/2)}$$

$$\downarrow *$$

Probabilities other than 1/2

The results do not depend on the probability $1/2$.

Theorem

Let $0 < p, q < 1$, $p, q \in \mathbf{Q}$.
p-$\mathrm{MC}(\mathrm{FO})$ *can be solved on* $G(n, p)$ *in expected FPT time iff* p-$\textsc{Dominating Set}$ *can be solved on* $G(n, q)$ *in expected FPT time.*

The proof uses some dirty tricks.

E.g.: We average the running time over all possible inputs.

We can use parts of the input as "random bits" to derandomize a randomized algorithm.

Average complexity of Even Set

One way to define Even Set is this:

p-EVEN SET	
Input:	A boolean matrix $M \in \{0,1\}^{n \times n}$ and $k \in \mathbf{N}$.
Parameter:	k
Problem:	Are there k rows in M whose logical XOR is the zero vector?

p-EVEN SET is famous because its complexity was unknown for a long time.

Bhattacharyya, Bonnet, Egri, Ghoshal, Karthik C. S., Lin, Manurangsi, and Marx showed recently that p-EVEN SET is W[1]-hard under randomized fpt-reductions.

Average complexity of Even Set

While p-$\textsc{Matrix}(\wedge)$ is as hard as p-$\mathrm{MC}(\mathrm{FO})$ on average, we get:

Theorem

p-$\textsc{Even Set}$ can be solved in FPT on average.

Proof:

- reduce to rectangular matrix
- compute the rank in $\mathbf{F}_2$
- if full rank, answer is no
- otherwise solve in $n^{O(k)}$

Conclusion

- Worst-case complexity:

  p-CLIQUE, p-DOMINATING SET, p-MC(FO) increasingly hard

- Average-case complexity:

  p-CLIQUE easy, p-DOMINATING SET, p-MC(FO) equally hard

- Worst-case complexity:

  p-EVEN SET and p-MATRIX($\wedge$) hard

- Average-case complexity:

  p-EVEN SET easy, p-MATRIX($\wedge$) as hard as p-MC(FO)

Big open question:

Can we relate hardness on average to hardness in the worst-case?