Treewidth: Vol. 2

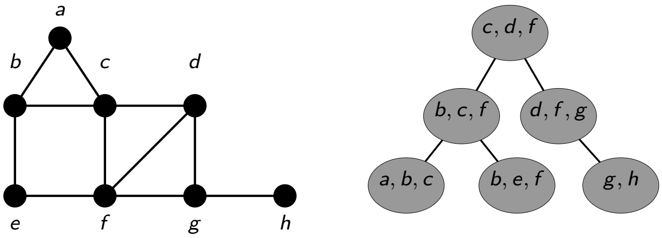
Dániel Marx

Lecture #12
January 25, 2022

Treewidth — a measure of "tree-likeness"

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 v, the bags containing v form a connected subtree.

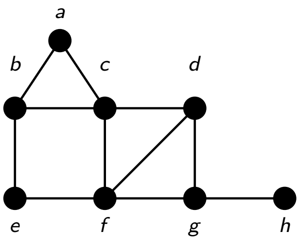


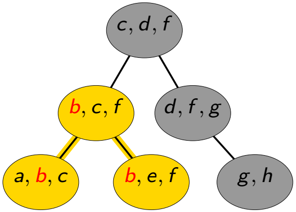
A subtree communicates with the outside world only via the root of the subtree.

Treewidth — a measure of "tree-likeness"

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 v, the bags containing v form a connected subtree.



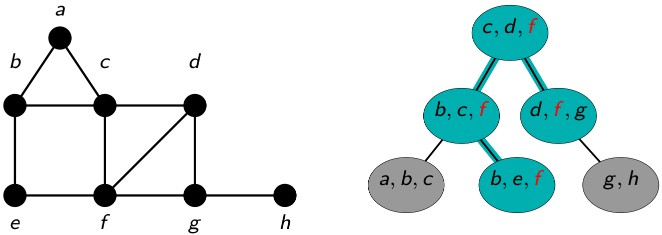


A subtree communicates with the outside world only via the root of the subtree.

Treewidth — a measure of "tree-likeness"

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 v, the bags containing v form a connected subtree.



A subtree communicates with the outside world
only via the root of the subtree.

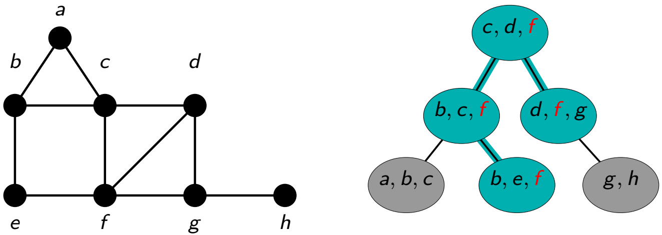
Treewidth — a measure of "tree-likeness"

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 v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.



A subtree communicates with the outside world
only via the root of the subtree.

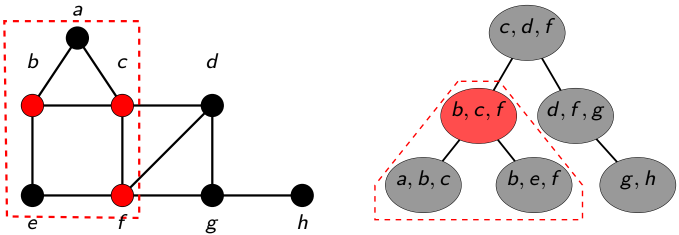
Treewidth — a measure of "tree-likeness"

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 v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.



A subtree communicates with the outside world only via the root of the subtree.

WEIGHTED MAX INDEPENDENT SET and treewidth

Theorem

Given a tree decomposition of width w, WEIGHTED MAX INDEPENDENT SET can be solved in time $O(2^w \cdot w^{O(1)} \cdot n)$.

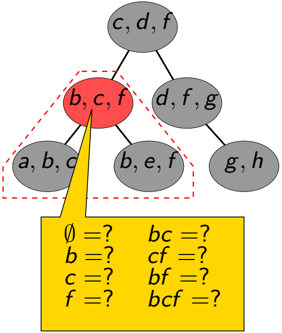
B_x: vertices appearing in node x.

V_x: vertices appearing in the subtree rooted at x.

Generalizing our solution for trees:

Instead of computing 2 values $A[v]$, $B[v]$ for each **vertex** of the graph, we compute $2^{|B_x|} \leq 2^{w+1}$ values for each bag B_x.

$M[x, S]$:
the max. weight of an independent set $I \subseteq V_x$ with $I \cap B_x = S$.

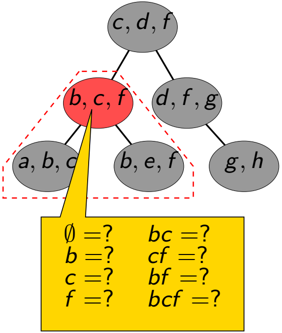


Weighted Max Independent Set and treewidth

Theorem

Given a tree decomposition of width w, Weighted Max Independent Set can be solved in time $O(2^w \cdot w^{O(1)} \cdot n)$.

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



Generalizing our solution for trees:

Instead of computing 2 values $A[v]$, $B[v]$ for each **vertex** of the graph, we compute $2^{|B_x|} \le 2^{w+1}$ values for each bag B_x.

$M[x,S]$:
the max. weight of an independent set $I \subseteq V_x$ with $I \cap B_x = S$.

How to determine $M[x,S]$ if all the values are known for the children of x?

Monadic Second Order Logic

Extended Monadic Second Order Logic (EMSO)

A logical language on graphs consisting of the following:

- Logical connectives $\wedge$, $\vee$, $\rightarrow$, $\neg$, $=$, $\neq$
- quantifiers $\forall$, $\exists$ over vertex/edge variables
- predicate $\mathrm{adj}(u, v)$: vertices u and v are adjacent
- predicate $\mathrm{inc}(e, v)$: edge e is incident to vertex v
- quantifiers $\forall$, $\exists$ over vertex/edge set variables
- $\in$, $\subseteq$ for vertex/edge sets

Example:
The formula

$$\exists C \subseteq V \forall v \in C \; \exists u_1, u_2 \in C (u_1 \neq u_2 \wedge \mathrm{adj}(u_1, v) \wedge \mathrm{adj}(u_2, v))$$

is true on graph G if and only if G has a cycle.

Courcelle's Theorem

Courcelle's Theorem

There exists an algorithm that, given a width-w tree decomposition of an n-vertex graph G and an EMSO formula ϕ, decides whether G satisfies ϕ in time $f(w, |\phi|) \cdot n$.

If we can express a property in EMSO, then we immediately get that testing this property is FPT parameterized by the treewidth w of the input graph.

$\Rightarrow$ The following problem are FPT parameterized by treewidth:

- c-Coloring
- Hamiltonian Cycle
- Partition into Triangles
- ...

Running time

Lots of research on this topic:

Many of the hard algorithmic problems on graphs can be solved in time $f(w) \cdot n^{O(1)}$ if a tree decomposition of width w is given.

In other words: these problems are **fixed-parameter tractable (FPT)** parameterized by treewidth.

Running time

Lots of research on this topic:

Many of the hard algorithmic problems on graphs can be solved in time $f(w) \cdot n^{O(1)}$ if a tree decomposition of width w is given.

In other words: these problems are **fixed-parameter tractable (FPT)** parameterized by treewidth.

What does the $f(w)$ depend on?

1. The **number of subproblems** at each node.
   (often depends on the **number of states** of each vertex)
2. The time needed to handle a **join node**.

Running time

Lots of research on this topic:

Many of the hard algorithmic problems on graphs can be solved in time $f(w) \cdot n^{O(1)}$ if a tree decomposition of width w is given.

In other words: these problems are **fixed-parameter tractable (FPT)** parameterized by treewidth.

What does the $f(w)$ depend on?

1. The **number of subproblems** at each node.
   (often depends on the **number of states** of each vertex)
2. The time needed to handle a **join node**.

Can we prove lower bounds on the best possible $f(w)$ for a problem?

Exponential Time Hypothesis (ETH)

3CNF: ϕ is a conjuction of clauses, where each clause is a disjunction of at most 3 literals (= a variable or its negation), e.g., $(x_1 \vee x_3 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3) \vee (x_1 \vee x_2 \vee x_4)$.

3SAT: given a 3CNF formula ϕ with n variables and m clauses, decide whether ϕ is satisfiable.

- Current best algorithm is 1.30704^n [Hertli 2011].
- Can we do **significantly** better, e.g, $2^{O(n/\log n)}$?

Exponential Time Hypothesis (ETH)

3CNF: ϕ is a conjuction of clauses, where each clause is a disjunction of at most 3 literals (= a variable or its negation), e.g., $(x_1 \vee x_3 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3) \vee (x_1 \vee x_2 \vee x_4)$.

3SAT: given a 3CNF formula ϕ with n variables and m clauses, decide whether ϕ is satisfiable.

- Current best algorithm is 1.30704^n [Hertli 2011].
- Can we do **significantly** better, e.g, $2^{O(n/\log n)}$?

Hypothesis introduced by Impagliazzo, Paturi, and Zane in 2001:

Exponential Time Hypothesis (ETH) [consequence of]

There is no $2^{o(n)}$-time algorithm for n-variable 3SAT.

Exponential Time Hypothesis (ETH)

3CNF: ϕ is a conjuction of clauses, where each clause is a disjunction of at most 3 literals (= a variable or its negation), e.g., $(x_1 \vee x_3 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3) \vee (x_1 \vee x_2 \vee x_4)$.

3SAT: given a 3CNF formula ϕ with n variables and m clauses, decide whether ϕ is satisfiable.

- Current best algorithm is 1.30704^n [Hertli 2011].
- Can we do **significantly** better, e.g, $2^{O(n/\log n)}$?

Hypothesis introduced by Impagliazzo, Paturi, and Zane in 2001:

Exponential Time Hypothesis (ETH) [real statement]

There is a constant $\delta > 0$ such that there is no $O(2^{\delta n})$ time algorithm for 3SAT.

Sparsification

Exponential Time Hypothesis (ETH) [consequence of]

There is no $2^{o(n)}$-time algorithm for n-variable $\mathrm{3SAT}$.

Observe: an n-variable $\mathrm{3SAT}$ formula can have $m = \Omega(n^3)$ clauses.

Are there algorithms that are subexponential in the size $n + m$ of the $\mathrm{3SAT}$ formula?

Sparsification

Exponential Time Hypothesis (ETH) [consequence of]

There is no $2^{o(n)}$-time algorithm for n-variable 3SAT.

Observe: an n-variable 3SAT formula can have $m = \Omega(n^3)$ clauses.

Are there algorithms that are subexponential in the size $n + m$ of the 3SAT formula?

Sparsification Lemma

There is a $2^{o(n)}$-time algorithm for n-variable 3SAT.

$\Updownarrow$

There is a $2^{o(n+m)}$-time algorithm for n-variable m-clause 3SAT.

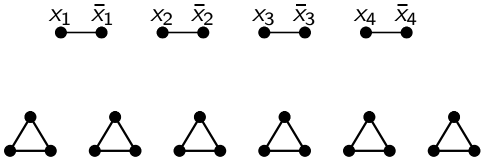
Intuitively: When considering a hard 3SAT instance, we can assume that it has $m = O(n)$ clauses.

Lower bounds based on ETH

Exponential Time Hypothesis (ETH) + Sparsification Lemma

There is no $2^{o(n+m)}$-time algorithm for n-variable m-clause 3SAT.

The textbook reduction from 3SAT to INDEPENDENT SET:

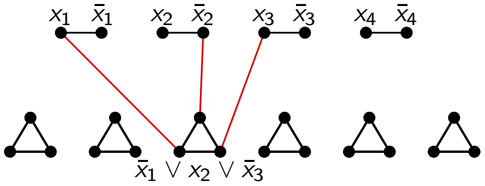


Lower bounds based on ETH

Exponential Time Hypothesis (ETH) + Sparsification Lemma

There is no $2^{o(n+m)}$-time algorithm for n-variable m-clause 3SAT.

The textbook reduction from 3SAT to INDEPENDENT SET:



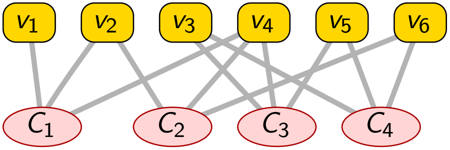
formula is satisfiable $\Leftrightarrow$ there is an independent set of size $n + 2m$

Lower bounds based on ETH

Exponential Time Hypothesis (ETH) + Sparsification Lemma

There is no $2^{o(n+m)}$-time algorithm for n-variable m-clause 3SAT.

3SAT formula ϕ n variables m clauses	$\Rightarrow$	Graph G $O(n+m)$ vertices $O(n+m)$ edges

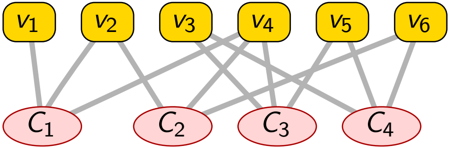


Lower bounds based on ETH

Exponential Time Hypothesis (ETH) + Sparsification Lemma

There is no $2^{o(n+m)}$-time algorithm for n-variable m-clause 3SAT.

3SAT formula ϕ		Graph G
n variables	$\Rightarrow$	$O(n+m)$ vertices
m clauses		$O(n+m)$ edges



Corollary

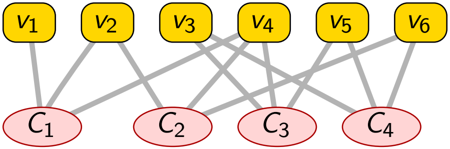
Assuming ETH, there is no $2^{o(n)}$ algorithm for Independent Set on an n-vertex graph.

Lower bounds based on ETH

Exponential Time Hypothesis (ETH) + Sparsification Lemma

There is no $2^{o(n+m)}$-time algorithm for n-variable m-clause 3SAT.

3SAT formula ϕ		Graph G
n variables	$\Rightarrow$	$O(n+m)$ vertices
m clauses		$O(n+m)$ edges



Corollary

Assuming ETH, there is no $2^{o(w)} \cdot n^{O(1)}$ algorithm for INDEPENDENT SET on graphs of treewidth w.

Lower bounds for treewidth

Similarly, assuming ETH, there is no $2^{o(w)} \cdot n^{O(1)}$ time algorithm for

- Independent Set
- Dominating Set
- Odd Cycle Transversal
- ...

Are there other problems where some other form of running time is optimal?

Hamiltonian cycle and treewidth

Theorem

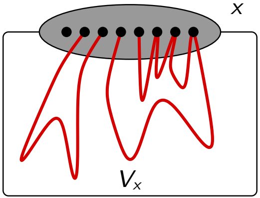
Given a tree decomposition of width w, HAMILTONIAN CYCLE can be solved in time $w^{O(w)} \cdot n$.

B_x: vertices appearing in node x.

V_x: vertices appearing in the subtree rooted at x.

If H is a Hamiltonian cycle, then the subgraph $H[V_x]$ is a set of paths with endpoints in B_x.

What are the important properties of $H[V_x]$ "seen from outside"?



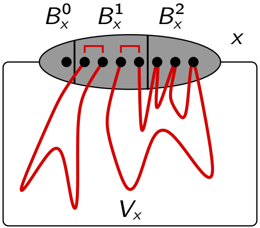
Hamiltonian cycle and treewidth

Theorem

Given a tree decomposition of width w, HAMILTONIAN CYCLE can be solved in time $w^{O(w)} \cdot n$.

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

If H is a Hamiltonian cycle, then the subgraph $H[V_x]$ is a set of paths with endpoints in B_x.



What are the important properties of $H[V_x]$ "seen from outside"?

- The subsets B_x^0, B_x^1, B_x^2 of B_x having degree 0, 1, and 2.
- The matching M of B_x^1.

No. of subproblems (B_x^0, B_x^1, B_x^2, M) for node x: at most $3^w \cdot w^w$.
For each subproblem, we have to determine if there is a set of paths with this pattern.

Cut and count

A very powerful technique for many problems on graphs of bounded-treewidth.

Classical result:

Theorem

Given a tree decomposition of width w, Hamiltonian Cycle can be solved in time $w^{O(w)} \cdot n^{O(1)} = 2^{O(w \log w)} \cdot n^{O(1)}$.

Seems like best possible, but...

Cut and count

A very powerful technique for many problems on graphs of bounded-treewidth.

Classical result:

Theorem

Given a tree decomposition of width w, HAMILTONIAN CYCLE can be solved in time $w^{O(w)} \cdot n^{O(1)} = 2^{O(w \log w)} \cdot n^{O(1)}$.

Seems like best possible, but...

Improved algorithms:

Given a tree decomposition of width w, HAMILTONIAN CYCLE can be solved in time $4^w \cdot n^{O(1)}$.

The first technique achieving this was *Cut & Count.*

Isolation Lemma

Isolation Lemma [Mulmuley, Vazirani, Vazirani 1987]

Let $\mathcal{F}$ be a nonempty family of subsets of U and assign a weight $w(u) \in [N]$ to each $u \in U$ uniformly and independently at random. The probability that there is a **unique** $S \in \mathcal{F}$ having minimum weight is at least

$$1 - \frac{|U|}{N}.$$

Proof:

- Def: $x \in U$ is singular in a weight assignment if there are minimum weight sets $A, B \in \mathcal{F}$ with $x \in A$ and $x \notin B$.
- **Claim 1:** $x \in U$ is singular in a random w with probability $\leq 1/N$.

  Set w randomly except $w(x) = 0$. If a (resp. b) is the min. weight of a set containing x (resp. not containing x), then x becomes singular *only* if we set $w(x) = b - a$.
- **Claim 2:** If there is no singular $x \in U$, then there is a unique minimum weight set.

Isolation Lemma

Isolation Lemma [Mulmuley, Vazirani, Vazirani 1987]

Let $\mathcal{F}$ be a nonempty family of subsets of U and assign a weight $w(u) \in [N]$ to each $u \in U$ uniformly and independently at random. The probability that there is a **unique** $S \in \mathcal{F}$ having minimum weight is at least

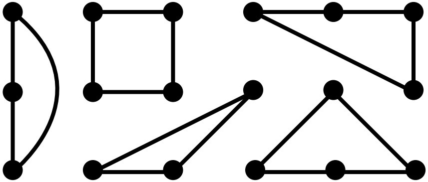
$$1 - \frac{|U|}{N}.$$

Let $U = E(G)$ and $\mathcal{F}$ be the set of all Hamiltonian cycles.

- By setting $N := |V(G)|^{O(1)}$, we can assume that there is a unique minimum weight Hamiltonian cycle.
- If N is polynomial in the input size, we can guess this minimum weight.
- So we are looking for a Hamiltonian cycle of weight **exactly** C, under the assumption that there is a **unique** such cycle.

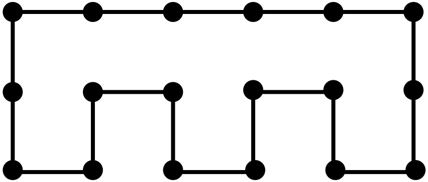
Cycle covers

- **Cycle cover:** A subgraph having degree exactly two at each vertex.



Cycle covers

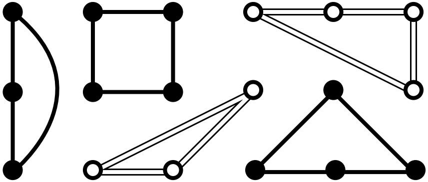
- **Cycle cover:** A subgraph having degree exactly two at each vertex.



- A Hamiltonian cycle is a cycle cover, but a cycle cover can have more than one component.

Cycle covers

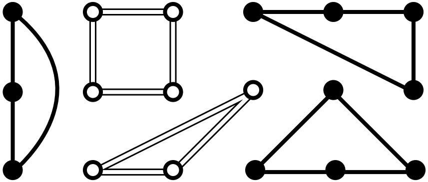
- **Cycle cover:** A subgraph having degree exactly two at each vertex.



- A Hamiltonian cycle is a cycle cover, but a cycle cover can have more than one component.
- **Colored cycle cover:** each component is colored black or white.

Cycle covers

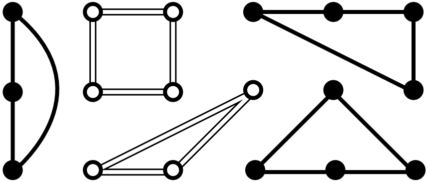
- **Cycle cover:** A subgraph having degree exactly two at each vertex.



- A Hamiltonian cycle is a cycle cover, but a cycle cover can have more than one component.
- **Colored cycle cover:** each component is colored black or white.

Cycle covers

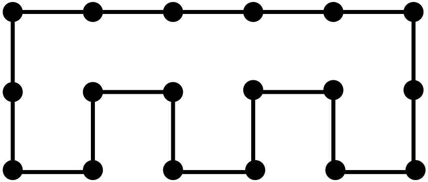
- **Cycle cover:** A subgraph having degree exactly two at each vertex.



- A Hamiltonian cycle is a cycle cover, but a cycle cover can have more than one component.
- **Colored cycle cover:** each component is colored black or white.
- A cycle cover with k components gives rise to 2^k colored cycle covers.
  - If there is no weight-C Hamiltonian cycle: the number of weight-C colored cycle covers is $0 \bmod 4$.
  - If there is a unique weight-C Hamiltonian cycle: the number of weight-C colored cycle covers is $2 \bmod 4$.

Cycle covers

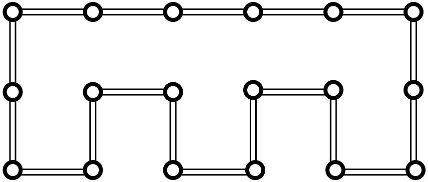
- **Cycle cover:** A subgraph having degree exactly two at each vertex.



- A Hamiltonian cycle is a cycle cover, but a cycle cover can have more than one component.
- **Colored cycle cover:** each component is colored black or white.
- A cycle cover with k components gives rise to 2^k colored cycle covers.
  - If there is no weight-C Hamiltonian cycle: the number of weight-C colored cycle covers is $0 \bmod 4$.
  - If there is a unique weight-C Hamiltonian cycle: the number of weight-C colored cycle covers is $2 \bmod 4$.

Cycle covers

- **Cycle cover:** A subgraph having degree exactly two at each vertex.



- A Hamiltonian cycle is a cycle cover, but a cycle cover can have more than one component.
- **Colored cycle cover:** each component is colored black or white.
- A cycle cover with k components gives rise to 2^k colored cycle covers.
  - If there is no weight-C Hamiltonian cycle: the number of weight-C colored cycle covers is $0 \bmod 4$.
  - If there is a unique weight-C Hamiltonian cycle: the number of weight-C colored cycle covers is $2 \bmod 4$.

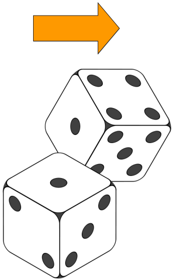
Cut and Count

- Assign random weights $\leq 2|E(G)|$ to the edges.
- If there is a Hamiltonian cycle, then with probability $1/2$, there is a C such that there is a **unique** weight-C Hamiltionian cycle.
- Try all possible C.
- Count the number of weight-C colored cycle covers: can be done in time $4^w \cdot n^{O(1)}$ if a tree decomposition of width w is given.
- Answer YES if this number is $2 \bmod 4$.

Cut and Count

HAMILTONIAN CYCLE

Random weights
success probability: 1/2

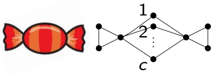


Counting weighted colored cycle covers

$4^w \cdot n^{O(1)}$ time

Optimal algorithms for tree decompositions

Assuming ETH, these running times are best possible:

Maximum Independent Set Dominating Set	$2^{O(w)}$
Hamiltonian Cycle *Cut & Count*	~~$2^{O(w \log w)}$~~ $2^{O(w)}$
Chromatic Number	$2^{O(w \log w)}$
Cycle Packing	$2^{O(w \log w)}$
Hitting Candy Graphs H_c : 	$2^{O(w^c)}$
3-Choosability	$2^{2^{O(w)}}$
3-Choosability Deletion	$2^{2^{2^{O(w)}}}$

Best possible bases

Algorithms given a tree decomposition of width w:

Independent Set	2^w
Dominating Set	3^w
c-Coloring	c^w
Odd Cycle Transversal	3^w
Partition into Triangles	2^w
Max Cut	2^w
#Perfect Matching	2^w

Are these constants best possible?

Can we improve 2 to 1.99?

Best possible bases

Algorithms given a tree decomposition of width w:

Independent Set	2^w
Dominating Set	3^w
c-Coloring	c^w
Odd Cycle Transversal	3^w
Partition into Triangles	2^w
Max Cut	2^w
#Perfect Matching	2^w

ETH seems to be too weak for this:
2^w vs. 4^w is just a polynomial difference!

ETH and SETH

Exponential Time Hypothesis (ETH)

There is a constant $\delta > 0$ such that there is no $O(2^{\delta n})$ time algorithm for $\mathrm{3SAT}$.

$$\text{Let } s_d = \inf\{c : d\text{-}\mathrm{SAT} \text{ has a } 2^{cn} \text{ algorithm}\}$$

$$\text{Let } s_\infty = \lim_{d\to\infty} s_d.$$

ETH: $s_3 > 0$ SETH: $s_\infty = 1$.

ETH and SETH

Exponential Time Hypothesis (ETH)

There is a constant $\delta > 0$ such that there is no $O(2^{\delta n})$ time algorithm for $\mathrm{3SAT}$.

$$\text{Let } s_d = \inf\{c : d\text{-}\mathrm{SAT} \text{ has a } 2^{cn} \text{ algorithm}\}$$

$$\text{Let } s_\infty = \lim_{d \to \infty} s_d.$$

ETH: $s_3 > 0$ SETH: $s_\infty = 1$.

In other words:

Strong Exponential-Time Hypothesis (SETH)

There is no $\epsilon > 0$ such that d-SAT for every d can be solved in time $O((2 - \epsilon)^n)$.

ETH and SETH

Exponential Time Hypothesis (ETH)

There is a constant $\delta > 0$ such that there is no $O(2^{\delta n})$ time algorithm for $\mathrm{3SAT}$.

$$\text{Let } s_d = \inf\{c : d\text{-}\mathrm{SAT} \text{ has a } 2^{cn} \text{ algorithm}\}$$

$$\text{Let } s_\infty = \lim_{d\to\infty} s_d.$$

ETH: $s_3 > 0$ SETH: $s_\infty = 1$.

In other words:

Strong Exponential-Time Hypothesis (SETH)

There is no $\epsilon > 0$ such that d-SAT for every d can be solved in time $O((2-\epsilon)^n)$.

Consequence of SETH

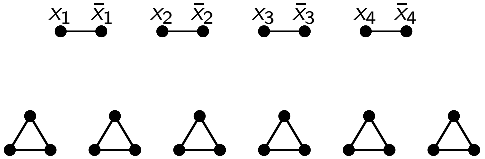
There is no $(2-\epsilon)^n \cdot m^{O(1)}$ time algorithm for SAT (with clauses of aribtrary length).

Lower bounds based on SETH

Strong Exponential-Time Hypothesis (SETH)

There is no $\epsilon > 0$ such that d-SAT for every d can be solved in time $O((2-\epsilon)^n)$.

The textbook reduction from 3SAT to Independent Set:

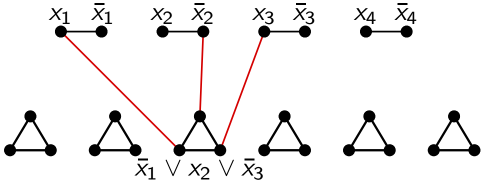


Lower bounds based on SETH

Strong Exponential-Time Hypothesis (SETH)

There is no $\epsilon > 0$ such that d-SAT for every d can be solved in time $O((2-\epsilon)^n)$.

The textbook reduction from 3SAT to Independent Set:



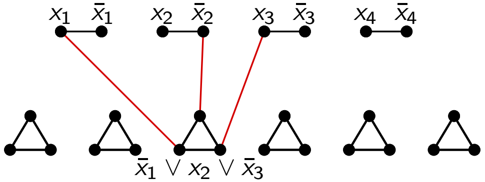
formula is satisfiable $\Leftrightarrow$ there is an independent set of size $n + m$

Lower bounds based on SETH

Strong Exponential-Time Hypothesis (SETH)

There is no $\epsilon > 0$ such that d-SAT for every d can be solved in time $O((2-\epsilon)^n)$.

The textbook reduction from 3SAT to Independent Set:



Treewidth of the constructed graph is at most $2n+2$.

Lower bounds based on SETH

Strong Exponential-Time Hypothesis (SETH)

There is no $\epsilon > 0$ such that d-SAT for every d can be solved in time $O((2-\epsilon)^n)$.

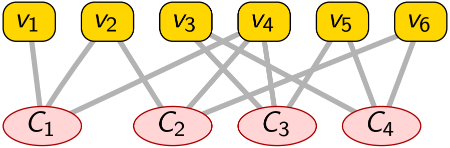
$\mathrm{3SAT}$ formula ϕ n variables m clauses	$\Rightarrow$	Graph G treewidth $2n+2$

Lower bounds based on SETH

Strong Exponential-Time Hypothesis (SETH)

There is no $\epsilon > 0$ such that d-SAT for every d can be solved in time $O((2-\epsilon)^n)$.

d-SAT formula ϕ n variables m clauses	$\Rightarrow$	Graph G treewidth $2n+d$

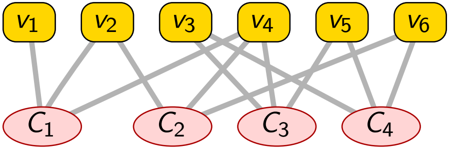


Lower bounds based on SETH

Strong Exponential-Time Hypothesis (SETH)

There is no $\epsilon > 0$ such that d-SAT for every d can be solved in time $O((2-\epsilon)^n)$.

d-SAT formula ϕ n variables m clauses	$\Rightarrow$	Graph G treewidth $2n + d$



Corollary

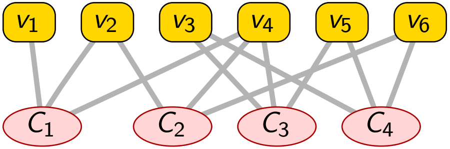
Assuming SETH, there is no $(2-\epsilon)^{w/2} \cdot n^{O(1)}$ algorithm for Independent Set for any $\epsilon > 0$.

Lower bounds based on SETH

Strong Exponential-Time Hypothesis (SETH)

There is no $\epsilon > 0$ such that d-SAT for every d can be solved in time $O((2-\epsilon)^n)$.

d-SAT formula ϕ n variables m clauses	$\Rightarrow$	Graph G treewidth $2n + d$



Corollary

Assuming SETH, there is no $(1.41-\epsilon)^w \cdot n^{O(1)}$ algorithm for Independent Set for any $\epsilon > 0$.

Better lower bound

We need a reduction of the following form for every d:

d-SAT formula ϕ n variables m clauses	$\Rightarrow$	Graph G treewidth $n + O_d(1)$

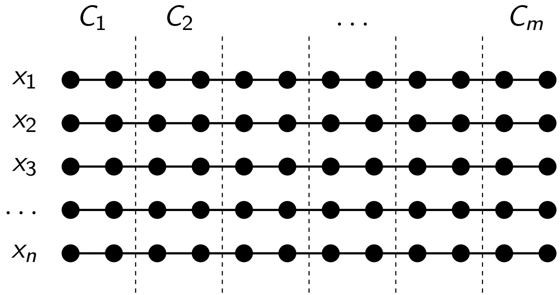
This would show:

Theorem

Assuming SETH, there is no $(2-\epsilon)^w \cdot n^{O(1)}$ algorithm for INDEPENDENT SET for any $\epsilon > 0$.

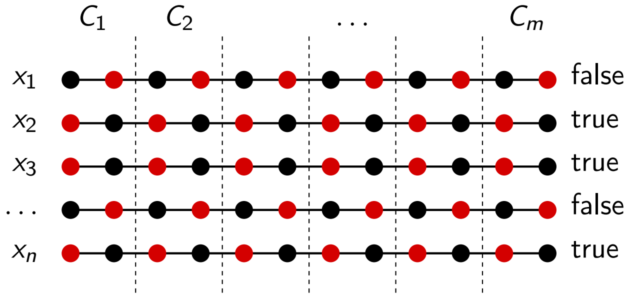
Better lower bound

d-SAT with n variables and m clauses $\Rightarrow$ n paths of $2m$ vertices.



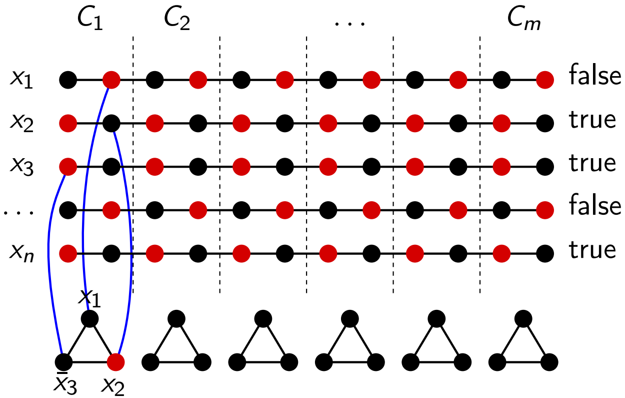
Better lower bound

d-SAT with n variables and m clauses $\Rightarrow$ n paths of $2m$ vertices.



Better lower bound

d-SAT with n variables and m clauses $\Rightarrow$ n paths of $2m$ vertices.

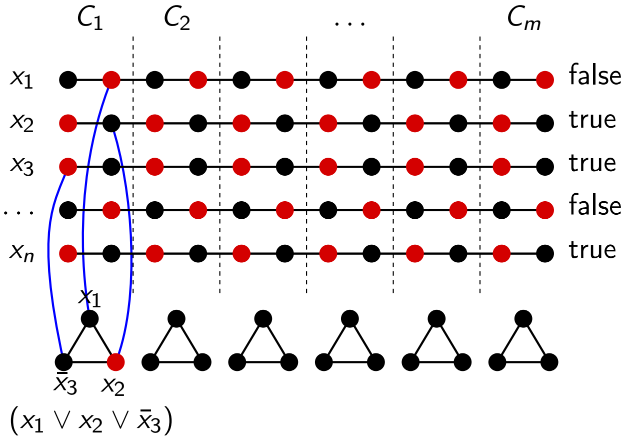


$(x_1 \vee x_2 \vee \bar{x}_3)$

Independent set of size $nm + m \iff$ formula is satisfiable

Better lower bound

d-SAT with n variables and m clauses $\Rightarrow$ n paths of $2m$ vertices.

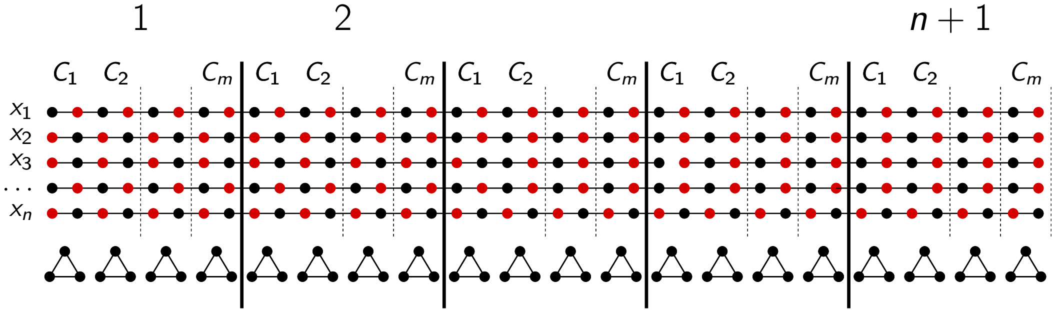


Not difficult to show: treewidth is at most $n + d$

A problem

A path may start as "true" and switch to "false".

Simple solution: repeat the instance $n+1$ times.

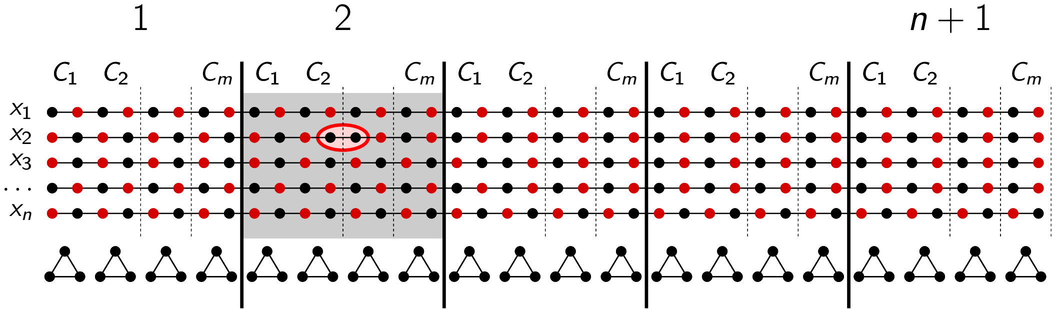


By the Pigeonhole Principle, there is a repetition where no switch occurs.

A problem

A path may start as “true” and switch to “false”.

Simple solution: repeat the instance $n + 1$ times.

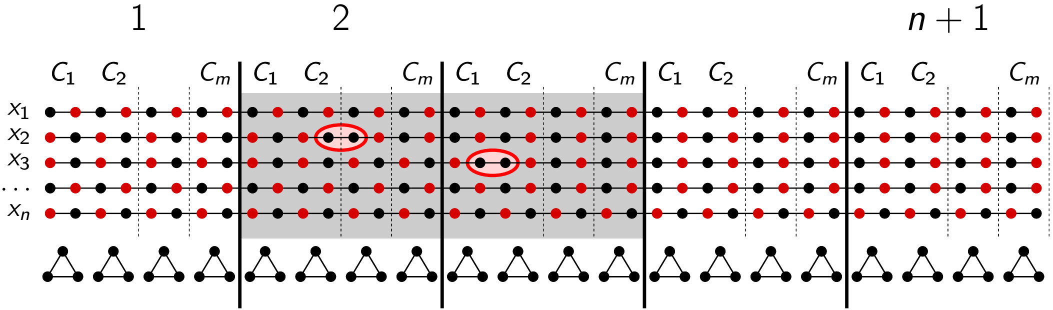


By the Pigeonhole Principle, there is a repetition where no switch occurs.

A problem

A path may start as "true" and switch to "false".

Simple solution: repeat the instance $n+1$ times.



By the Pigeonhole Principle, there is a repetition where no switch occurs.

Lower bound for Independent Set

We have shown: Reduction from n-variable d-SAT to Independent Set in a graph with treewidth $w = n + d$.

$$(2-\epsilon)^w \cdot n^{O(1)} \text{ algorithm for Independent Set}$$
$$\Downarrow$$
$$(2-\epsilon)^n \cdot n^{O(1)} \text{ algorithm for } d\text{-SAT}$$

As this is true for any d, having such an algorithm for Independent Set would violate SETH.

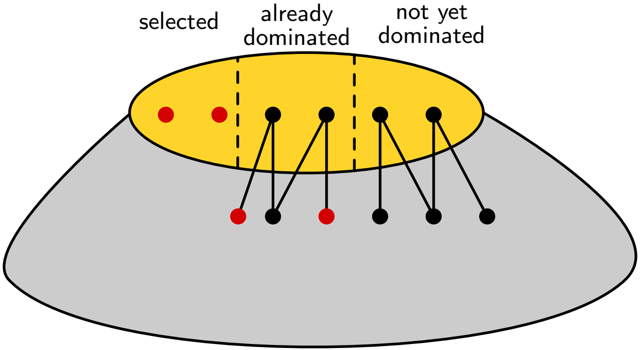
Theorem

Assuming SETH, there is no $(2-\epsilon)^w \cdot n^{O(1)}$ algorithm for Independent Set for any $\epsilon > 0$.

Dominating Set and treewidth

Dominating Set: Given G and k, find a set S of k vertices such that every vertex of G is in S or has a neighbor in S.

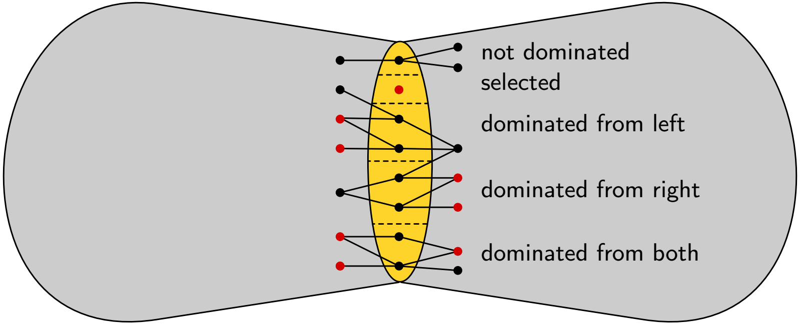
Each vertex has three possible states in a bag:



$\Rightarrow 3^{w+1}$ different subproblems at each node.

DOMINATING SET and treewidth

How to solve a subproblem at a join node?



- Natural approach: $9^w \cdot n^{O(1)}$ [Telle and Proskurowski 1993]
- Considering the 5 possibilities: $5^w \cdot n^{O(1)}$
- More efficiently: $4^w \cdot n^{O(1)}$ [Alber et al. 2002]
- Fast subset convolution: $3^w \cdot n^{O(1)}$ [Björkund et al. 2007], [Rooij et al. 2009]

Lower bound for Dominating Set

Theorem

Assuming SETH, there is no $(3-\epsilon)^w \cdot n^{O(1)}$ algorithm for Dominating Set for any $\epsilon > 0$.

We need a reduction of the following form for every d:

d-SAT formula ϕ n variables m clauses	$\Rightarrow$	Graph G treewidth $n/\log_2 3 + O_d(1)$

Then

$$(3-\epsilon)^w \leq (3-\epsilon)^{n/\log_2 3} \cdot 3^{O_d(1)} = O_d((2-\epsilon')^n)$$

Lower bound for Dominating Set

Theorem

Assuming SETH, there is no $(3-\epsilon)^w \cdot n^{O(1)}$ algorithm for Dominating Set for any $\epsilon > 0$.

We need a reduction of the following form for every d:

d-SAT formula ϕ n variables m clauses	$\Rightarrow$	Graph G treewidth $n/\log_2 3 + O_d(1)$

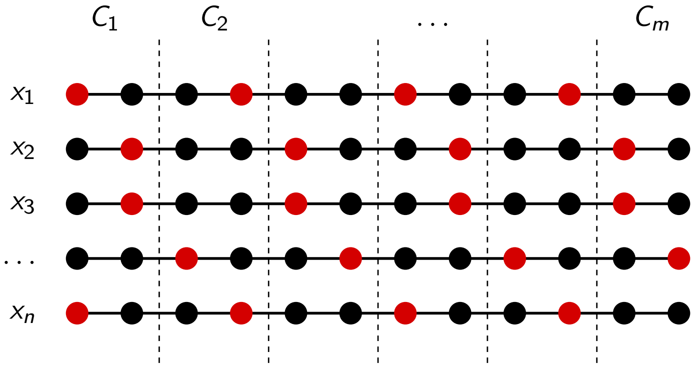
Then

$$(3-\epsilon)^w \leq (3-\epsilon)^{n/\log_2 3} \cdot 3^{O_d(1)} = O_d((2-\epsilon')^n)$$

How to increase treewidth by only $1/\log_2 3 = 0.6309$ for every variable?

Lower bound for DOMINATING SET

Each path has now 3 different states.



A group of g variables can be described by $\approx \log_3 2^g = g/\log_2 3 = 0.6309g$ paths.

Best possible bases

Assuming SETH...

Independent Set	no $(2-\epsilon)^w$
Dominating Set	no $(3-\epsilon)^w$
c-Coloring	no $(c-\epsilon)^w$
Odd Cycle Transversal	no $(3-\epsilon)^w$
Partition into Triangles	no $(2-\epsilon)^w$
Max Cut	no $(2-\epsilon)^w$
#Perfect Matching	no $(2-\epsilon)^w$

Distance-d versions

d-Scattered Set: find a set S of k vertices with pairwise distance $\geq d$.

Theorem

If there is an $\epsilon > 0$ and an algorithm solving d-Scattered Set in time $(d-\epsilon)^w \cdot n^{O(1)}$ on a tree decomposition of width w, then the SETH fails.

d states: selected; unselected at distance $1, 2, \ldots, \geq d-1$ from a selected.

Distance-d versions

d-Scattered Set: find a set S of k vertices with pairwise distance $\geq d$.

Theorem

If there is an $\epsilon > 0$ and an algorithm solving d-Scattered Set in time $(d-\epsilon)^w \cdot n^{O(1)}$ on a tree decomposition of width w, then the SETH fails.

d states: selected; unselected at distance $1, 2, \ldots, \geq d-1$ from a selected.

(k,d)-center: find a set S of k vertices such that every vertex is at most distance d from S.

Theorem

If there is an $\epsilon > 0$ and an algorithm solving (k,d)-center in time $(2d+1-\epsilon)^w \cdot n^{O(1)}$ on a tree decomposition of width w, then the SETH fails.

$2d+1$ states: selected; at distance $1, 2, \ldots, d$ from a selected vertex going up/down.

Connected problems

CONNECTED (k, d)-CENTER: find a set S of k vertices such that every vertex is at most distance d from S and $G[S]$ **is connected**.

- Handling connectivity in a standard way gives a $2^{O(w \log w)} \cdot n^{O(1)}$ algorithm.
- $2^{O(w)} \cdot n^{O(1)}$ running time requires the *Cut & Count* technique.

Theorem

- Given a tree decomposition of width w, CONNECTED (k, d)-CENTER can be solved in time $(2d + 2)^w \cdot n^{O(1)}$.
- If there is an $\epsilon > 0$ and an algorithm solving CONNECTED (k, d)-CENTER in time $(2d + 2 - \epsilon)^w \cdot n^{O(1)}$ on a tree decomposition of width w, then the SETH fails.

List Coloring

List Coloring is a generalization of ordinary vertex coloring: given a

- graph G,
- a set of colors C, and
- a list $L(v) \subseteq C$ for each vertex v,

the task is to find a coloring c where $c(v) \in L(v)$ for every v.

Theorem

Vertex Coloring is FPT parameterized by treewidth.

However, list coloring is more difficult:

Theorem

List Coloring is W[1]-hard parameterized by treewidth.

Parameterized reductions

Definition

Parameterized reduction from problem A to problem B: a function ϕ with the following properties:

- $\phi(x)$ is a yes-instance of B $\iff$ x is a yes-instance of A,
- $\phi(x)$ can be computed in time $f(k) \cdot |x|^{O(1)}$, where k is the parameter of x,
- If k is the parameter of x and k' is the parameter of $\phi(x)$, then $k' \leq g(k)$ for some function g.

Parameterized reductions

Definition

Parameterized reduction from problem A to problem B: a function ϕ with the following properties:

- $\phi(x)$ is a yes-instance of $B \iff x$ is a yes-instance of A,
- $\phi(x)$ can be computed in time $f(k) \cdot |x|^{O(1)}$, where k is the parameter of x,
- If k is the parameter of x and k' is the parameter of $\phi(x)$, then $k' \leq g(k)$ for some function g.

Theorem

If there is a parameterized reduction from problem A to problem B and B is FPT, then A is also FPT.

Intuitively: Reduction $A \to B$ + algorithm for B gives an algorithm for A.

W[1]-hard: Clique can be reduced to it.

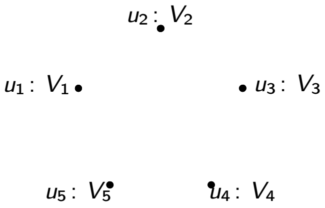
List Coloring

Theorem

List Coloring is W[1]-hard parameterized by treewidth.

Proof: By reduction from Multicolored Independent Set.

- Let G be a graph with color classes $V_1, \dots, V_k$.
- Set C of colors: the set of vertices of G.
- The colors appearing on vertices $u_1, \dots, u_k$ correspond to the k vertices of the clique, hence we set $L(u_i) = V_i$.



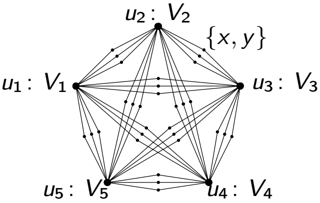
List Coloring

Theorem

List Coloring is W[1]-hard parameterized by treewidth.

Proof: By reduction from Multicolored Independent Set.

- Let G be a graph with color classes $V_1, \ldots, V_k$.
- Set C of colors: the set of vertices of G.
- The colors appearing on vertices $u_1, \ldots, u_k$ correspond to the k vertices of the clique, hence we set $L(u_i) = V_i$.
- If $x \in V_i$ and $y \in V_j$ are adjacent in G, then we need to ensure that $c(u_i) = x$ and $c(u_j) = y$ are not true at the same time $\Rightarrow$ we add a vertex adjacent to u_i and u_j whose list is $\{x, y\}$.



Treewidth and complexity

- Many natural problems are FPT parameterized by treewidth — but not all (e.g., List Coloring).
- The ETH can be used to prove tight lower bounds on the $f(k)$ in the running time $f(k)n^{O(1)}$.
- The SETH can be used to prove tight lower bounds on c in the running time $c^k \cdot n^{O(1)}$.

Treewidth — a measure of "tree-likeness"

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 v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.