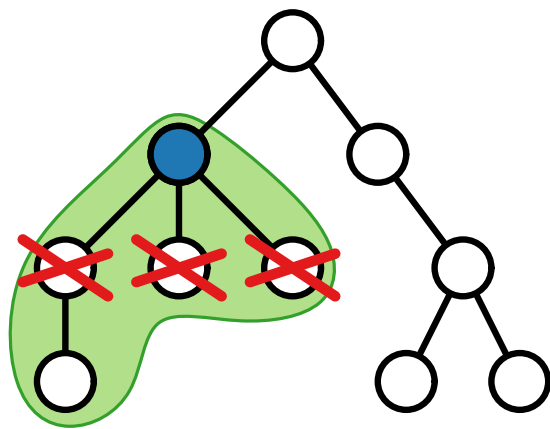


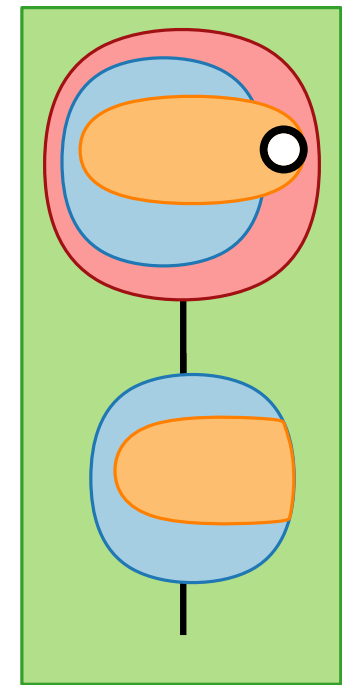
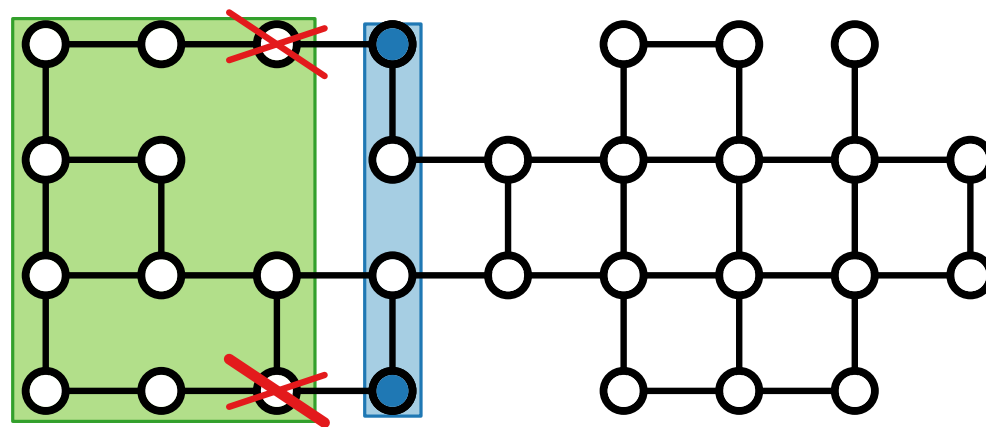
Advanced Algorithms

Parameterized Algorithms

Structural Parametrization



Alexander Wolff

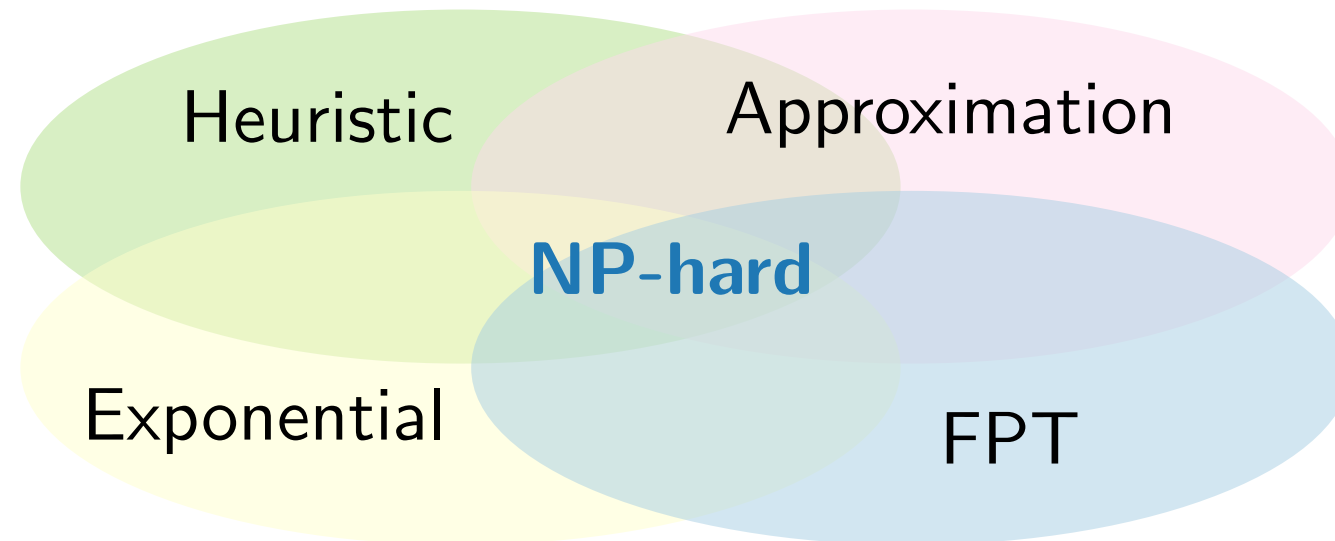


Summer term 2026

Dealing with NP-Hard Problems

What should we do?

- Sacrifice optimality for speed
 - Heuristics
 - Approximation Algorithms
- Optimal Solutions
 - Exact exponential-time algorithms
 - Fine-grained analysis – parameterized algorithms ← this lecture



Parameterized Algorithms

Classical complexity theory:

Running time is expressed as a function in the input size.

Parameterized algorithmics:

Running time is expressed as a function in the input size, as well as one or more additional parameter(s).

Example: (recall from AGT)

k -VERTEX COVER

Input: Graph G , $k \in \mathbb{N}$

Question: Is there a set $C \subseteq V(G)$ with $|C| \leq k$
s.t. $\forall \{u, v\} \in E(G): \{u, v\} \cap C \neq \emptyset$?

- NP-complete,
- but \exists algorithm that runs in $\mathcal{O}(2^k k \cdot (|V(G)| + |E(G)|))$ time.

Idea: If $k \in \mathcal{O}(1)$, then $\mathcal{O}(2^k k \cdot (|V(G)| + |E(G)|)) \subseteq \mathcal{O}(\text{poly}(|V(G)| + |E(G)|))$, i.e., if we assume the **parameter** k to be **fixed**, k -VERTEX COVER becomes **tractable**.

Parameterized Complexity Classes

Definition.

Let Π be a decision problem. If there is

- an algorithm \mathcal{A} and
- a computable function f

such that, given an instance I of Π and a parameter $k \in \mathbb{N}$, the algorithm \mathcal{A} provides the correct answer to I in time $f(k) \cdot |I|^{\mathcal{O}(1)}$, then \mathcal{A} and Π are called **fixed-parameter tractable (FPT)** with respect to k .

If \mathcal{A} provides the correct answer to I in time $|I|^{f(k)}$, then \mathcal{A} and Π are called **slice-wise polynomial (XP)** with respect to k . (Note that $\text{FPT} \subsetneq \text{XP}$.)

Example. k -VERTEX COVER can be solved in time $\mathcal{O}\left(\underbrace{2^k \cdot k}_{f(k)} \cdot \underbrace{(|V| + |E|)}_{|I|^{\mathcal{O}(1)}}\right)$.

\Rightarrow k -VERTEX COVER is FPT (and therefore also XP) with respect to k .

Examples and Counterexamples

k -VERTEX COVER

- NP-complete
- but FPT with respect to k

In all these examples, k is the *natural* parameter that comes with the decision problem.

k -CLIQUE

- NP-complete
- but XP with respect to k
- Under common assumptions, k -CLIQUE is not FPT with respect to k (namely, k -CLIQUE is $W[1]$ -complete with respect to k ; \rightarrow Section 13 in [1])
- There is an $\mathcal{O}(2^\Delta \cdot \Delta^2 \cdot (|V| + |E|))$ time algorithm for k -CLIQUE, where Δ is the maximum degree of the input graph $\Rightarrow k$ -CLIQUE is FPT with respect to Δ .

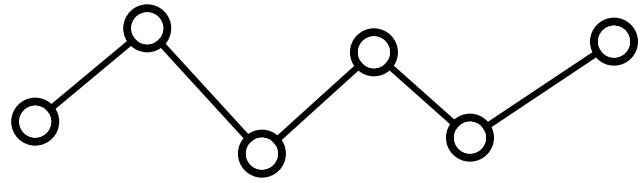
We can also study other types of parameters!

VERTEX k -COLORING

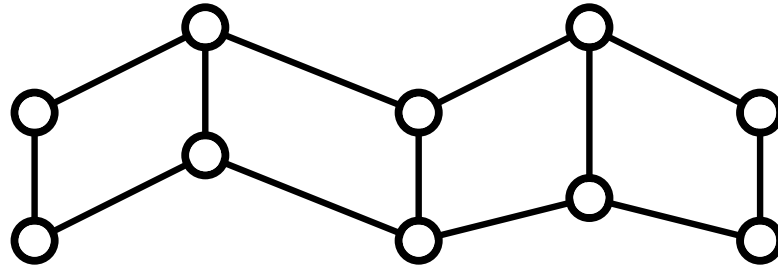
- NP-complete for every $k \geq 3$
- \Rightarrow neither FPT nor XP with respect to k , unless $P = NP$

Pathwidth and Treewidth (Intuition)

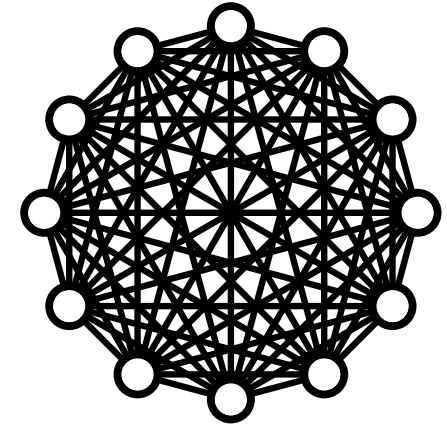
Pathwidth describes how *path-like* a graph is.



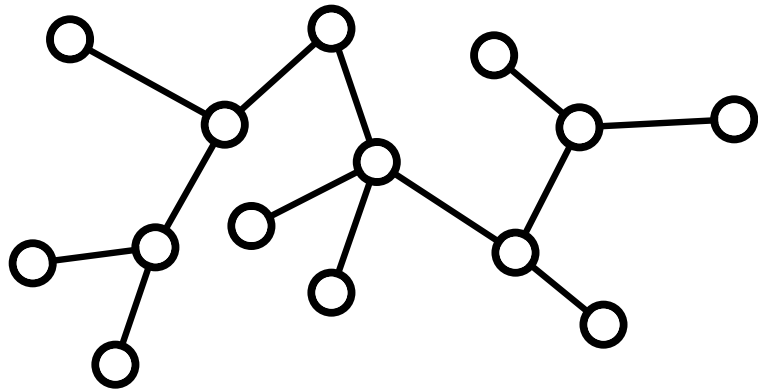
1



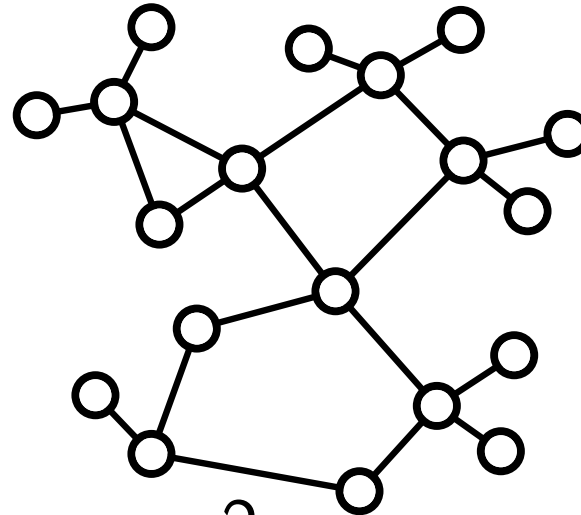
2

 $n - 1$

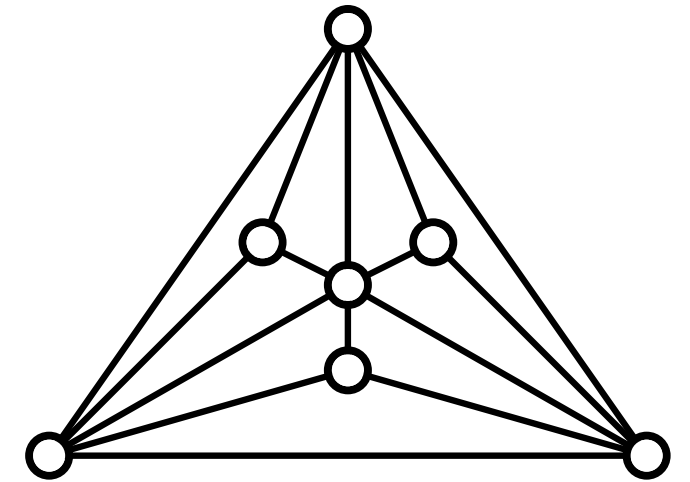
Treewidth describes how *tree-like* a graph is.



1



2



3

Path-/tree-like structure can be useful for designing dynamic programming algorithms.

(WEIGHTED) INDEPENDENT SET

Input. A graph G and a weight function $w: V(G) \rightarrow \mathbb{N}$.

Output. A set $I \subseteq V(G)$ that is **independent**, i.e., $\forall u, v \in I: \{u, v\} \notin E(G)$, and has **maximum weight**, i.e., $w(I) := \sum_{v \in I} w(v)$ is maximized.

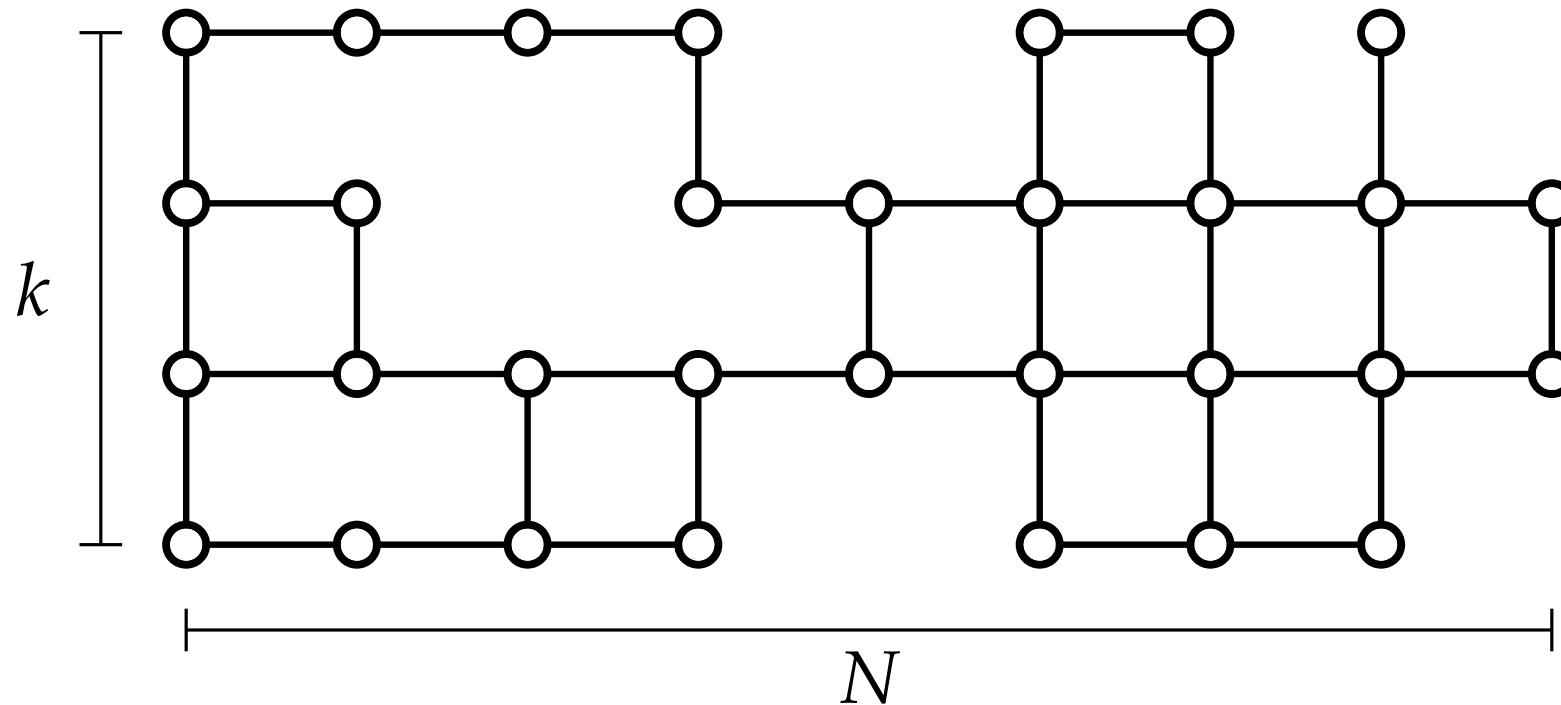


- (Already unweighted) INDEPENDENT SET is NP-complete,
- but can be solved efficiently on tree-like graphs (also when weighted).
- On trees, (WEIGHTED) INDEPENDENT SET can be solved in linear time.

Grid Graphs

In a $k \times N$ **grid graph**

- the vertex set consist of all pairs (i, j) where $1 \leq i \leq k$ and $1 \leq j \leq N$, and
- two vertices (i_1, j_1) and (i_2, j_2) are adjacent if and only if $|i_1 - i_2| + |j_1 - j_2| = 1$.



We will study INDEPENDENT SET in subgraphs of $k \times N$ grid graphs.

Goal: An FPT algorithm with respect to the parameter k , the *height* of the grid.

INDEPENDENT SET in $k \times N$ Grid Graphs _{N}

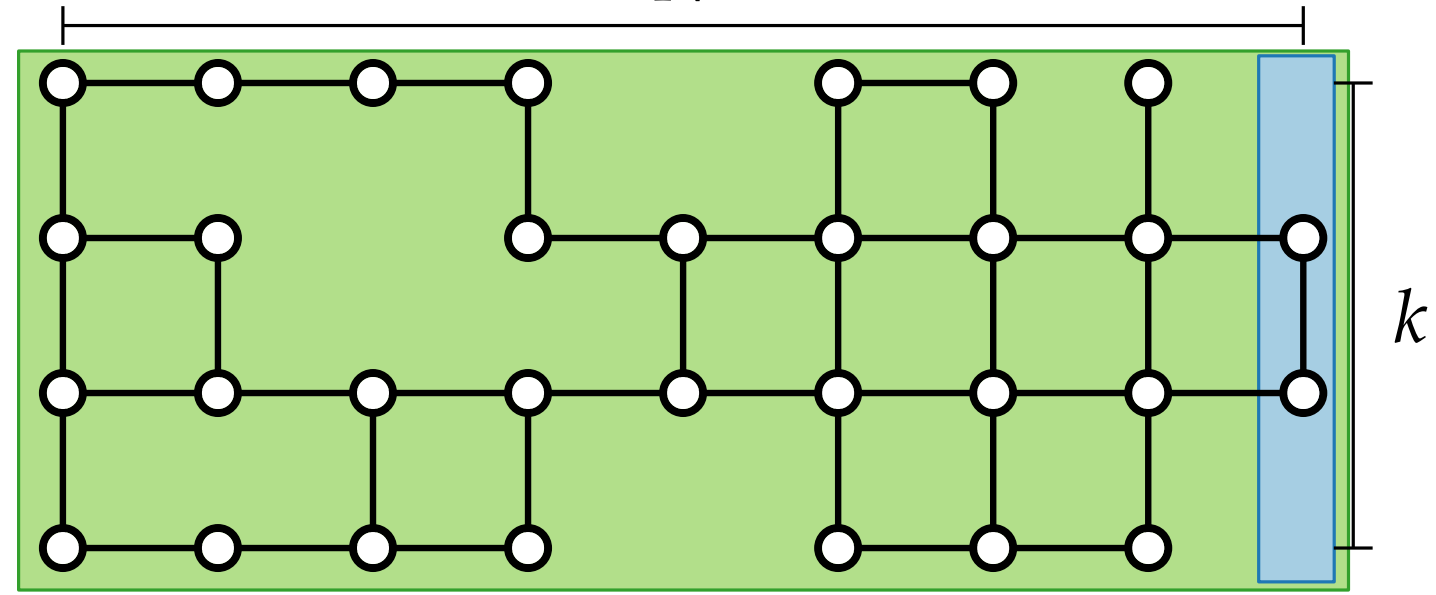
Let X_j be the j -th column, that is,
 $X_j = V(G) \cap \{(i, j) \mid 1 \leq i \leq k\}$.

Let G_j be the graph induced by the first j columns $X_1 \cup X_2 \cup \dots \cup X_j$.

Let $1 \leq j \leq N$. For each $Y \subseteq X_j$ let

$C[j, Y] :=$ maximum weight of an independent set I in G_j such that $I \cap Y = \emptyset$

$C[N, \emptyset] =$ solution



INDEPENDENT SET in $k \times N$ Grid Graphs _{N}

Let X_j be the j -th column, that is,
 $X_j = V(G) \cap \{(i, j) \mid 1 \leq i \leq k\}$.

Let G_j be the graph induced by the first j columns $X_1 \cup X_2 \cup \dots \cup X_j$.

Let $1 \leq j \leq N$. For each $Y \subseteq X_j$ let

$C[j, Y] :=$ maximum weight of an independent set I in G_j such that $I \cap Y = \emptyset$

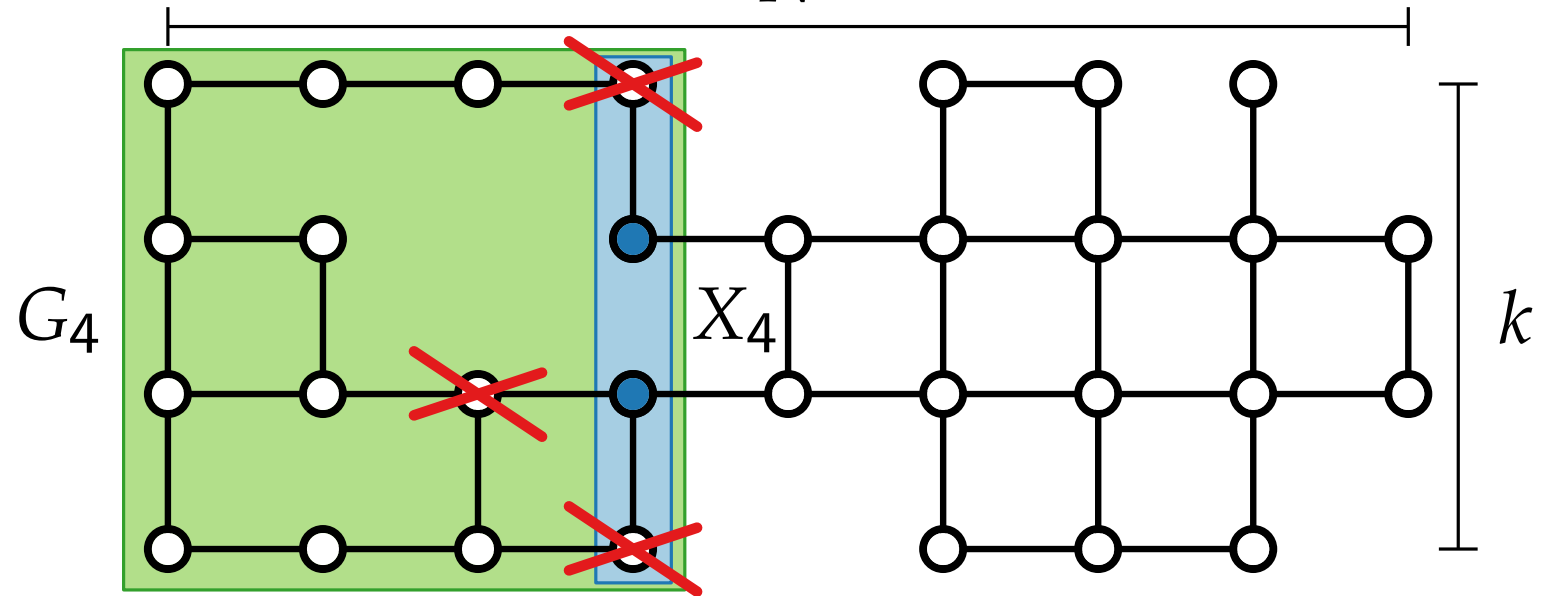
$C[1, Y] = \max_{I \subseteq X_1 \setminus Y \text{ where } I \text{ independent}} \{w(I)\}$

$C[j, Y] = \max_{I \subseteq X_j \setminus Y \text{ where } I \text{ independent}} \{w(I) + C[j-1, X_{j-1} \cap N(I)]\}$

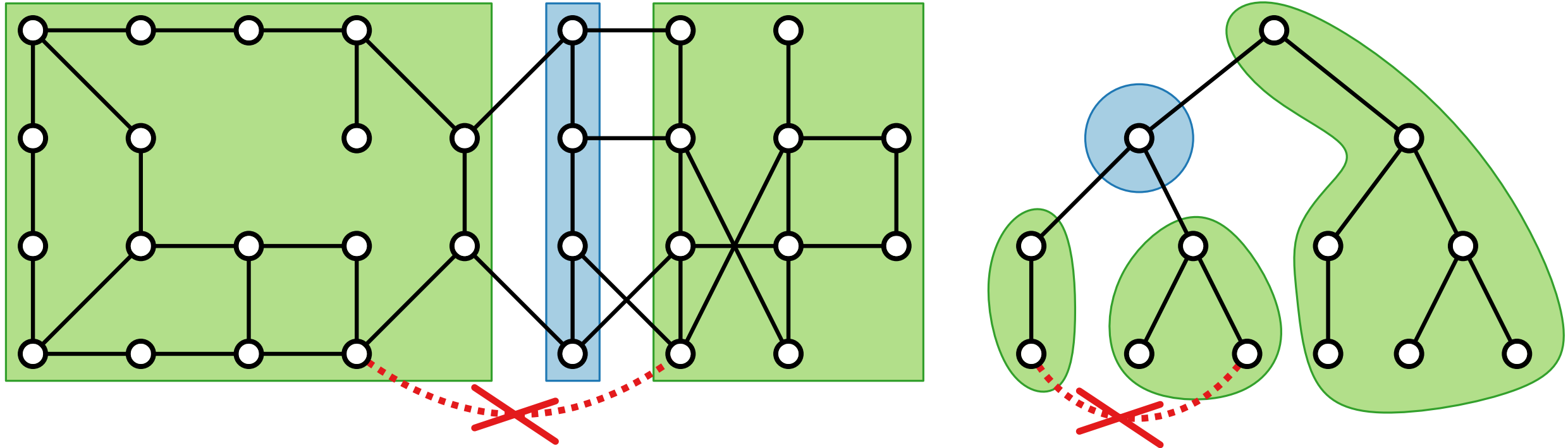
For each j there are $\leq 2^k$ choices of Y , and for each Y there are $2^{|X_j \setminus Y|}$ choices of I .

For each of these $\leq N3^k$ choices of I , we need to test whether I is independent.

\rightarrow total running time $\leq 3^k k^{\mathcal{O}(1)} N$.



Can We Apply This Approach to Other Graphs?



Yes!

We mainly used the fact that the graph consists of a **sequence of small separators**.

A similar fact was used in the algorithm for trees.

Goal: Define a more general graph class featuring a structure that is suited for this kind of dynamic programming approach.

Path Decompositions

Let G be a graph.

A **path decomposition** of G is a sequence $P = (X_1, X_2, \dots, X_r)$ of **bags**, where $X_i \subseteq V(G)$, such that

(P1) $\bigcup_{i=1}^r X_i = V(G)$

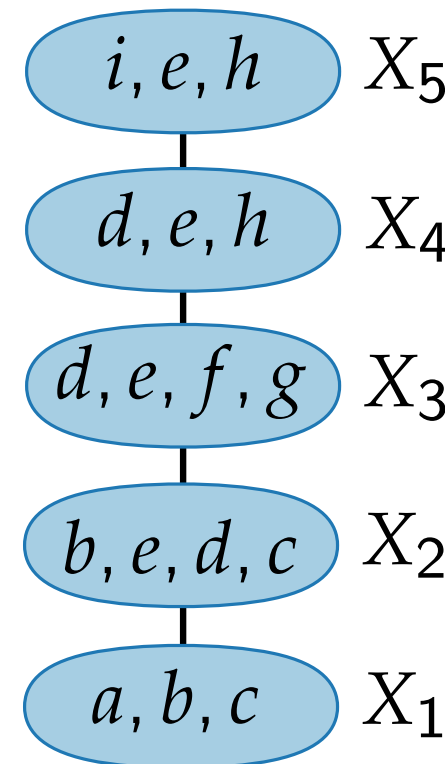
(P2) $\forall \{u, v\} \in E(G) \exists i \in \{1, 2, \dots, r\} : \{u, v\} \subseteq X_i$

(P3) $\forall v \in V(G)$, if $v \in X_i \cap X_j$ with $i \leq j$, then $v \in X_i \cap X_{i+1} \cap \dots \cap X_j$

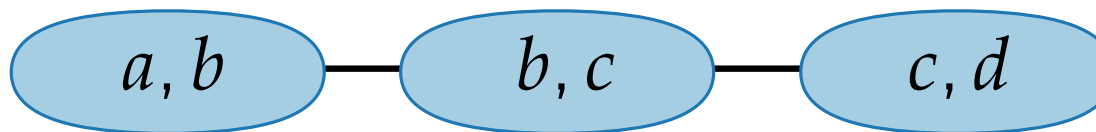
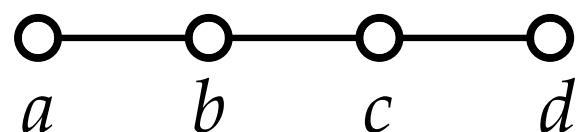
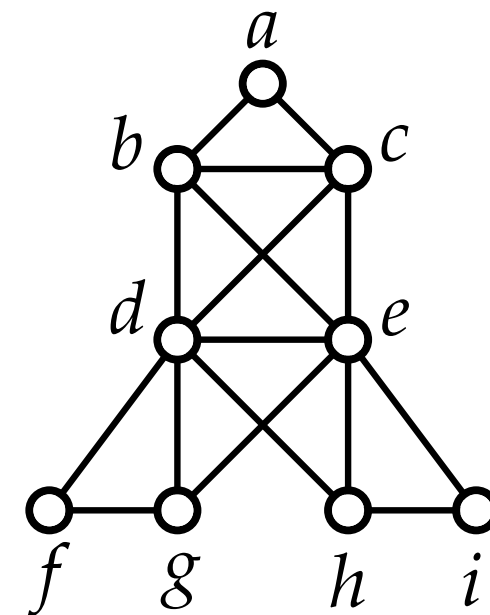
The **width** of P is $w(P) = \max_{1 \leq i \leq r} |X_i| - 1$.

The **pathwidth** $\text{pw}(G)$ of G is the minimum width of a path decomposition of G .

$w(P) = 3$



$\text{pw}(G) \leq 3$



$\text{pw}(G) = 1$

Okay – But Where Are the Separators?

Lemma 1. Let $i < r$. Then there is no edge between $A = (X_1 \cup X_2 \cup \dots \cup X_i) \setminus (X_i \cap X_{i+1})$ and $B = (X_{i+1} \cup X_{i+2} \cup \dots \cup X_r) \setminus (X_i \cap X_{i+1})$.

Proof. Assume there are $a \in A$ and $b \in B$ s.t. $\{a, b\} \in E(G)$.

Let $j \leq i$ s.t. $a \in X_j$ and let $k \geq i + 1$ s.t. $b \in X_k$.

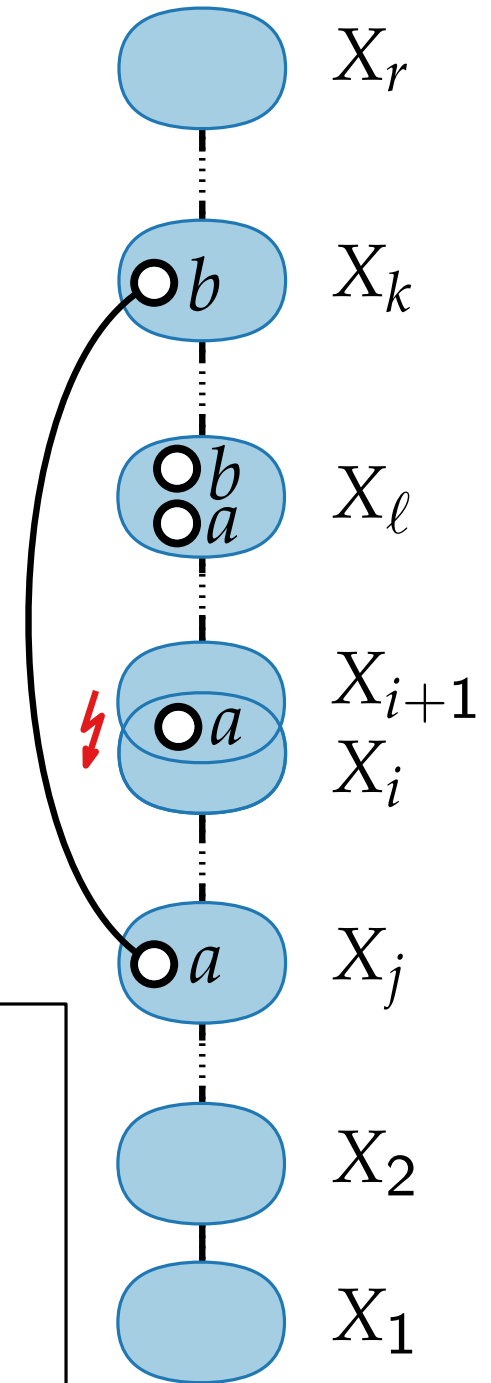
(P2) \Rightarrow there is a bag X_ℓ s.t. $a, b \in X_\ell$, w.l.o.g. let $\ell \geq i + 1$.

(P3) $\Rightarrow a \in X_i \cap X_{i+1}$; contradiction to $a \in A$. \square

$$\text{(P1)} \quad \bigcup_{i=1}^r X_i = V(G)$$

$$\text{(P2)} \quad \forall \{u, v\} \in E(G) \exists i \in \{1, 2, \dots, r\} : \{u, v\} \subseteq X_i$$

$$\text{(P3)} \quad \forall v \in V(G): \text{ if } v \in X_i \cap X_j \text{ with } i \leq j, \text{ then } v \in X_i \cap \dots \cap X_j$$



Computing Path Decompositions

k -PATHWIDTH

Input. Graph G , $k \in \mathbb{N}$

Question. Is the pathwidth of G at most k ?

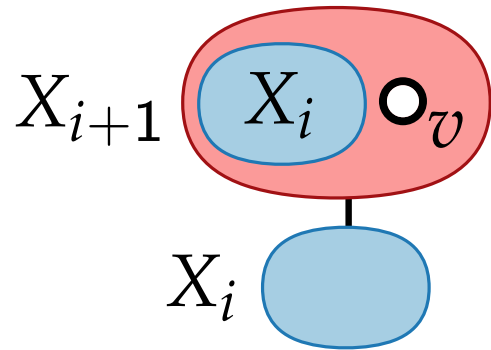
- NP-complete
- FPT in k
 - The algorithm constructs a path decomposition of width $\leq k$.
 - Its runtime depends linearly on $|V(G)| + |E(G)|$.

\Rightarrow When designing FPT algorithms with respect to the pathwidth, we may assume to be given a path decomposition!

Nice Path Decompositions

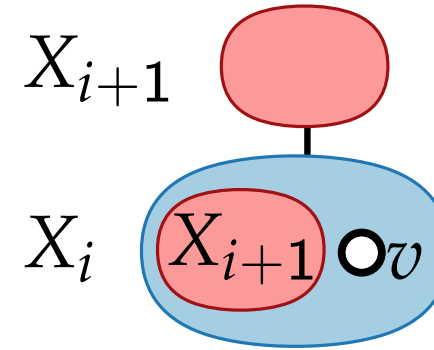
A path decomposition is **nice** if $|X_1| = 1$ and each other bag has one of two **types**:

X_{i+1} is of type **Introduce** if



$$X_{i+1} = X_i \cup \{v\} \text{ where } v \notin X_i$$

X_{i+1} is of type **Forget** if

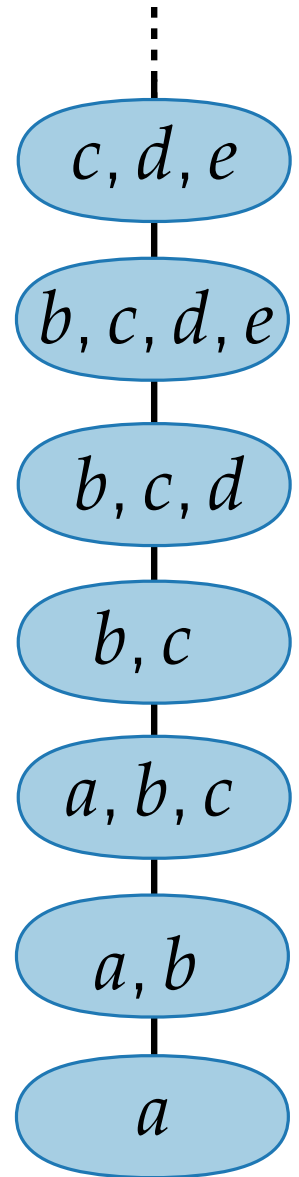
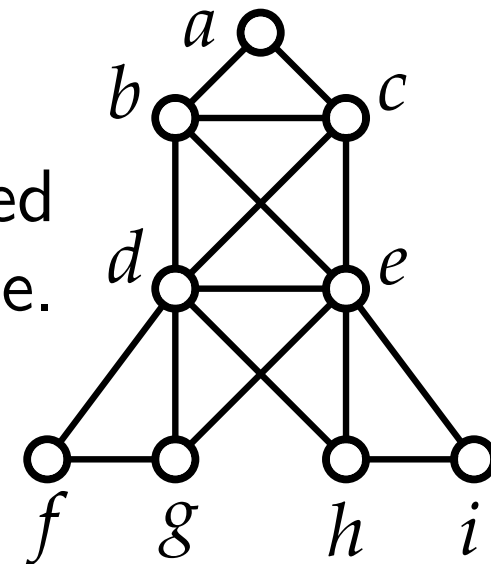


$$X_i = X_{i+1} \cup \{v\} \text{ where } v \notin X_{i+1}$$

Observation. The number of bags is $r \leq 2|V(G)| - 1$.

Lemma. A path decomposition of width k can be transformed into a nice path decomposition of width k in polynomial time.

\Rightarrow When designing FPT algorithms w.r.t. the pathwidth, we may assume to be given a *nice* path decomposition.



INDEPENDENT SET in Graphs of Bounded Pathwidth

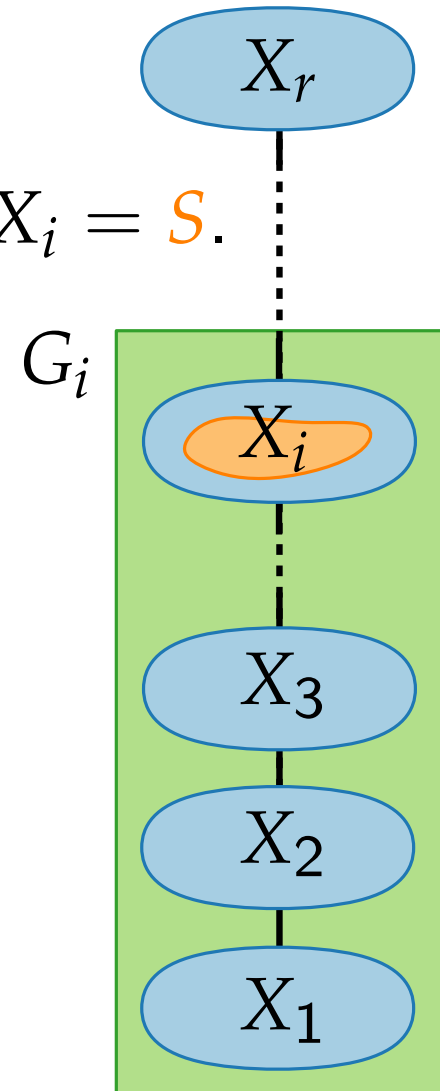
Assume we are given a nice path decomposition $P = (X_1, X_2, \dots, X_r)$ of width k .

For every $i \in \{1, 2, \dots, r\}$, let G_i be the graph induced by $X_1 \cup X_2 \cup \dots \cup X_i$.

For each $S \subseteq X_i$, let

$D[i, S] :=$ maximum weight of an independent set I in G_i such that $I \cap X_i = S$.

(P1) $\Rightarrow G_r = G \Rightarrow$ solution =



INDEPENDENT SET in Graphs of Bounded Pathwidth

Assume we are given a nice path decomposition $P = (X_1, X_2, \dots, X_r)$ of width k .

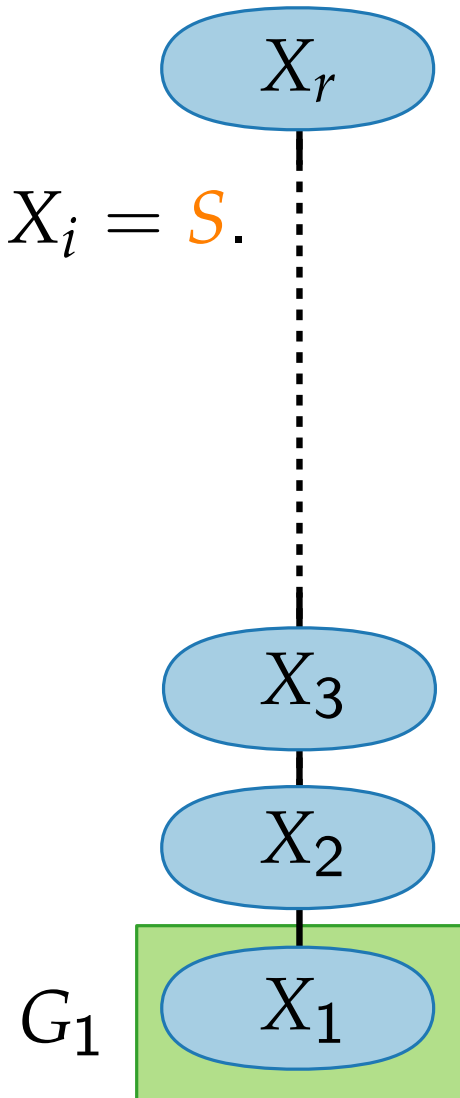
For every $i \in \{1, 2, \dots, r\}$, let G_i be the graph induced by $X_1 \cup X_2 \cup \dots \cup X_i$.

For each $S \subseteq X_i$, let

$D[i, S] :=$ maximum weight of an independent set I in G_i such that $I \cap X_i = S$.

(P1) $\Rightarrow G_r = G \Rightarrow$ solution $= \max_{S \subseteq X_r} D[r, S]$

$D[1, S] = \left\{ \right.$



INDEPENDENT SET in Graphs of Bounded Pathwidth

Assume we are given a nice path decomposition $P = (X_1, X_2, \dots, X_r)$ of width k .

For every $i \in \{1, 2, \dots, r\}$, let G_i be the graph induced by $X_1 \cup X_2 \cup \dots \cup X_i$.

For each $S \subseteq X_i$, let

$D[i, S] :=$ maximum weight of an independent set I in G_i such that $I \cap X_i = S$.

Assume that $i > 1$. If S is not independent, $D[i, S] = -\infty$.

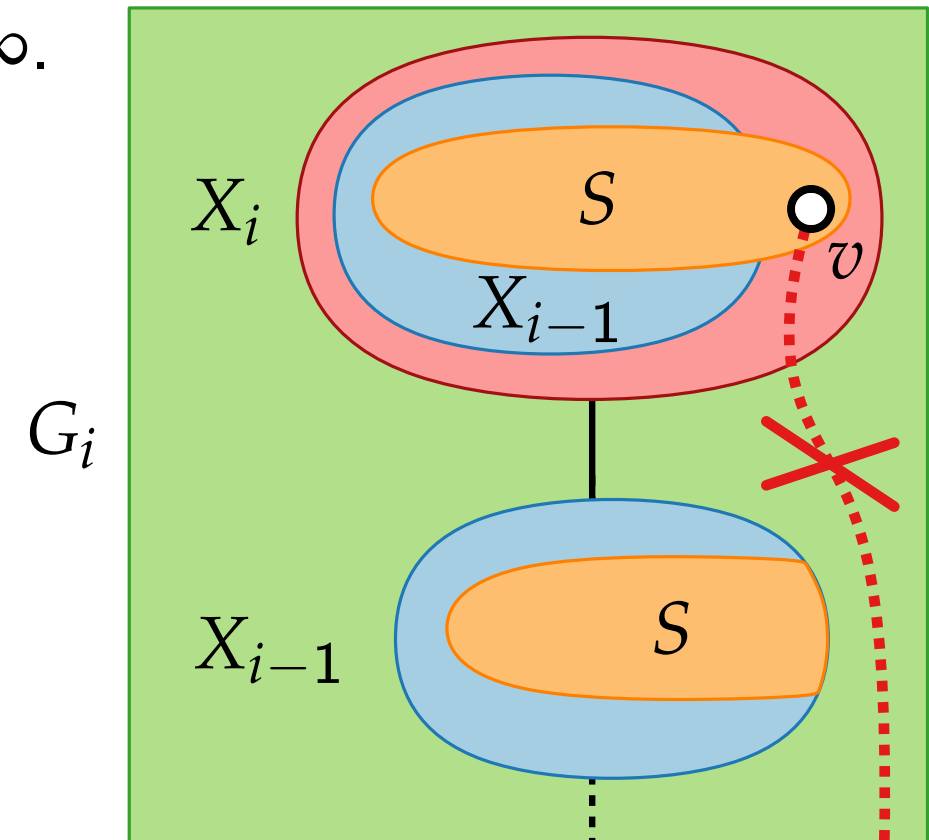
Otherwise, we distinguish between the two types of X_i .

If X_i is **Introduce**, then

$$D[i, S] = \begin{cases} D[i-1, S], & \text{if } v \notin S \\ w(v) + D[i-1, S \setminus \{v\}], & \text{if } v \in S \end{cases}$$

Let I' denote the independent set corresponding to

Why is $I' \cup \{v\}$ independent? Due to Lemma 1!



INDEPENDENT SET in Graphs of Bounded Pathwidth

Assume we are given a nice path decomposition $P = (X_1, X_2, \dots, X_r)$ of width k .

For every $i \in \{1, 2, \dots, r\}$, let G_i be the graph induced by $X_1 \cup X_2 \cup \dots \cup X_i$.

For each $S \subseteq X_i$, let

$D[i, S] :=$ maximum weight of an independent set I in G_i such that $I \cap X_i = S$.

Assume that $i > 1$. If S is not independent, $D[i, S] = -\infty$.

Otherwise, we distinguish between the two types of X_i .

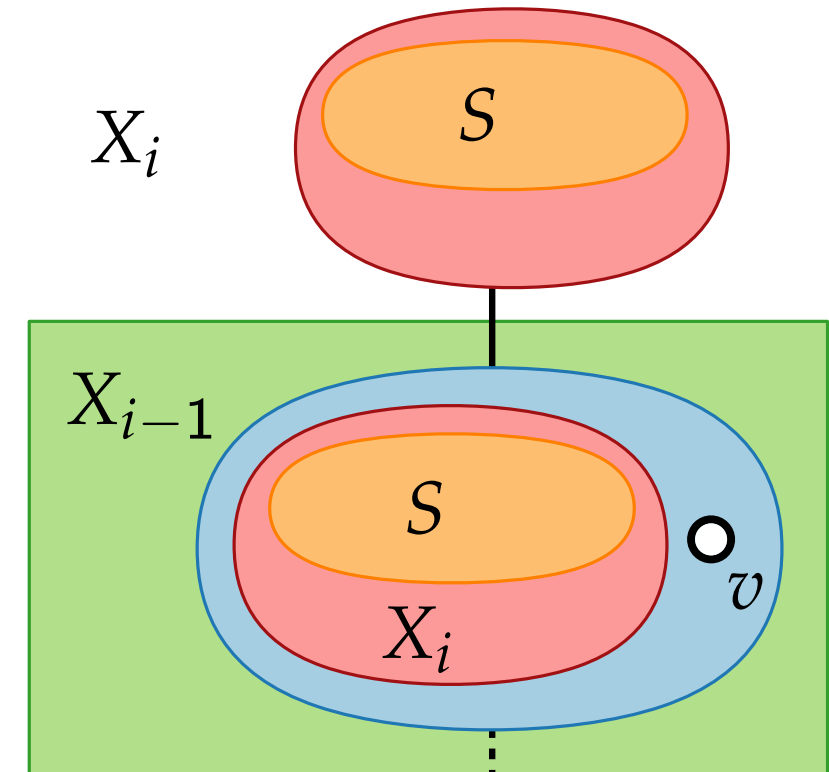
If X_i is **Forget**, then

$$D[i, S] = \max\{ D[i-1, S], D[i-1, S \cup \{v\}] \}$$

$$v \notin I \Rightarrow I \cap X_{i-1} = S$$

$$v \in I \Rightarrow I \cap X_{i-1} = S \cup \{v\}$$

$$G_i = G_{i-1}$$



INDEPENDENT SET in Graphs of Bounded Pathwidth

Assume we are given a nice path decomposition $P = (X_1, X_2, \dots, X_r)$ of width k .

For every $i \in \{1, 2, \dots, r\}$, let G_i be the graph induced by $X_1 \cup X_2 \cup \dots \cup X_i$.

For each $S \subseteq X_i$, let

$D[i, S] :=$ maximum weight of an independent set I in G_i such that $I \cap X_i = S$.

Assume that $i > 1$. If S is not independent, $D[i, S] = -\infty$.

Otherwise, we distinguish between the two types of X_i .

For each of the $\leq 2|V(G)| - 1$ many bags, there are $\leq 2^{k+1}$ choices for S .

For each of these choices, we need to test whether S is independent.

This can be done in $k^{\mathcal{O}(1)}$ time (\rightarrow Section 7.3.1 in [1]).

\Rightarrow Total running time $\in 2^{k+2} k^{\mathcal{O}(1)} \cdot |V(G)|$

Theorem. INDEPENDENT SET is FPT with respect to pathwidth.

Discussion

- The fixed-parameter tractability of a problem may be studied with respect to various structural parameters.
- The assumption that the chosen parameter is small should be plausible!
- **Treewidth** is among the most studied parameters.
 - It is defined like pathwidth, except that the bags form a tree instead of a path.
 - **Nice** tree decompositions have only one additional bag type ...
 - ... and can be constructed efficiently from a tree decomposition.
- Our $2^{\text{pw}(G)} \text{pw}(G)^{\mathcal{O}(1)} |V(G)|$ -time algorithm for INDEPENDENT SET can easily be turned into an algorithm that runs in $2^{\text{tw}(G)} \text{tw}(G)^{\mathcal{O}(1)} |V(G)|$ time.

Theorem. INDEPENDENT SET is FPT with respect to treewidth.

References and Literature

- [1] Parameterized Algorithms,
M. Cygan, F. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk,
M. Pilipczuk, S. Saurabh, Springer International Publishing 2015.

Sections 1, 7.1, 7.2, 7.3