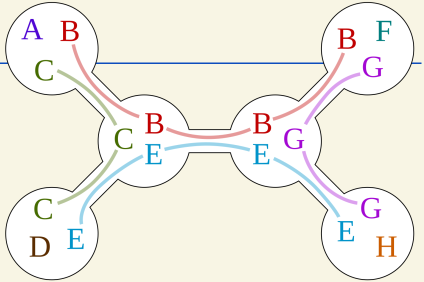
Treewidth II

DM898: Parameterized Algorithms
Lars Rohwedder



Today's lecture

Generalization of previous techniques from paths to trees:

- Tree decomposition
- Maximum Weight Independent Set over tree decomposition

Dynamic programming over trees

Maximum Weight Independent Set

The previous dynamic program for Maximum Weight Independent Set on paths easily generalizes to trees.

Weighted Independent Set

- Input: Graph $G = (V, E)$, weights $w : V \to \mathbb{Z}_{\geq 0}$
- Output: Vertex set $I \subseteq V$ with $(u, v) \notin E$ for each $u, v \in I$ where $\sum_{v \in I} w(v)$ is maximized

Maximum Weight Independent Set

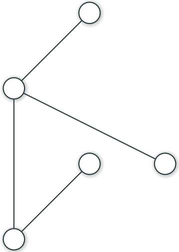
The previous dynamic program for Maximum Weight Independent Set on paths easily generalizes to trees.

Weighted Independent Set

- Input: Graph $G = (V, E)$, weights $w : V \to \mathbb{Z}_{\geq 0}$
- Output: Vertex set $I \subseteq V$ with $(u, v) \notin E$ for each $u, v \in I$ where $\sum_{v \in I} w(v)$ is maximized

Dynamic program if G is a tree

- Root G in an arbitrary vertex, creating set of children $\mathrm{child}(v) \subseteq V$, $v \in V$
- Let T_v be the subtree of v and descendants



Maximum Weight Independent Set

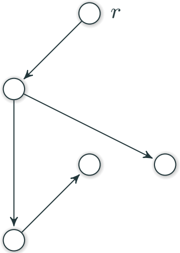
The previous dynamic program for Maximum Weight Independent Set on paths easily generalizes to trees.

Weighted Independent Set

- Input: Graph $G = (V, E)$, weights $w : V \to \mathbb{Z}_{\geq 0}$
- Output: Vertex set $I \subseteq V$ with $(u, v) \notin E$ for each $u, v \in I$ where $\sum_{v \in I} w(v)$ is maximized

Dynamic program if G is a tree

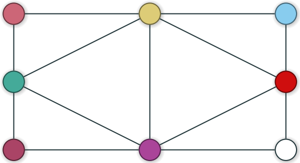
- Root G in an arbitrary vertex, creating set of children $\mathrm{child}(v) \subseteq V$, $v \in V$
- Let T_v be the subtree of v and descendants
- Dynamic table: $D[v]$, $v \in V$, which should contain maximum weight of independent set in T_v
- Recurrence (if v is chosen, none of the direct children can be): $D[v] = \max \left\{ w(v) + \sum_{u \in \mathrm{child}(v)} \sum_{u' \in \mathrm{child}(u)} D[u'] \, , \, \sum_{u \in \mathrm{child}(v)} D[u] \right\}$
- Proving correctness by induction is straight-forward
- Compute entries in order where children appear before parents. Then $D[r]$ contains optimal weight, solution can be output by easy modification



Treewidth

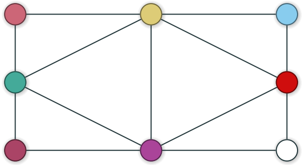
Treewidth and tree decomposition

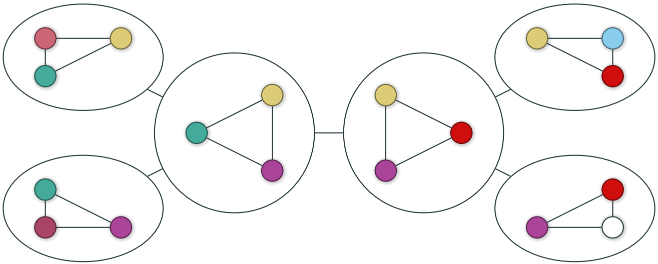
When is a graph **almost** a tree?



Treewidth and tree decomposition

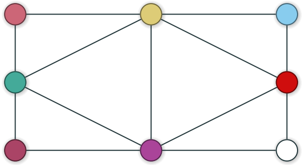
When is a graph **almost** a tree?

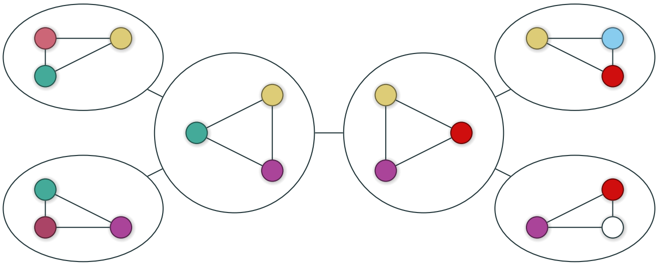




Treewidth and tree decomposition

When is a graph **almost** a tree?





Tree decomposition

A tree decomposition of a graph $G = (V, E)$ is a tree $T = (V_T, E_T)$ and a set of bags X_t, $t \in V_T$, such that

- $X_t \subseteq V$ for all t and $\bigcup_{t \in V_T} X_t = V$
- For each $(u, v) \in E$ there is some $t \in V_T$ with $\{u, v\} \subseteq X_t$
- For every $v \in V$, the set of bags containing v, i.e., $\{t \in V_T : v \in X_t\}$, is a connected subtree of T
- The **width** of the decomposition is $\max_{t \in V_T} |X_t| - 1$
- The **treewidth** of the graph, $\text{tw}(G)$ is the smallest width over any decomposition. If G is a tree itself, $\text{tw}(G) = 1$

Nice tree decomposition

Nice tree decomposition

A tree decomposition $T = (V_T, E_T), (X_t)_{t \in V_T}$ with a root $r \in V_T$ is **nice** $X_r = \emptyset$ and $X_\ell = \emptyset$ for each leaf $\ell \in V_T$ and for every non-leaf $t \in V_T$ either

- t has exactly one child t' and $X_t = X_{t'} \cup \{v\}$ for some $v \in X_t \setminus X_{t'}$ (**introduce node**),
- t has exactly one child t' and $X_t = X_{t'} \setminus \{v\}$ for some $v \in X_{t'} \setminus X_t$ (**forget node**), or
- t has exactly two children t', t'' with $X_t = X_{t'} = X_{t''}$ **(join node)**

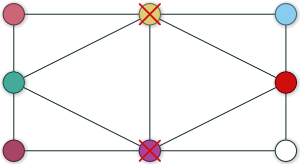
- We can in polynomial time transform a tree decomposition of width w to a nice path decomposition of the same width
- A nice tree decomposition is easier to work with in dynamic programming
- When devising FPT algorithms in $\mathrm{tw}(G)$ we assume that a tree decomposition is given

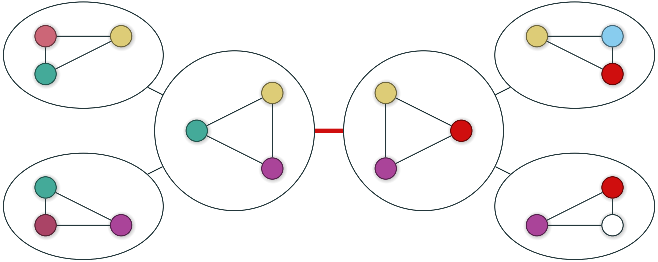
Separation

Lemma

Let $T = (V_T, E_T), (X_t)_{t \in V_T}$ be a tree decomposition of graph G. Let $(a, b) \in E_T$ be an edge of the decomposition and let $V_T^{(a)} \subseteq V_T$ ($V_T^{(b)} \subseteq V_T$) be the nodes of T on the side of a (resp., of b) of (a, b). Then there is no edge between $\bigcup_{t \in V_T^{(a)}} X_t \setminus (X_a \cap X_b)$ and $\bigcup_{t \in V_T^{(b)}} X_t \setminus (X_a \cap X_b)$. We say $X_a \cap X_b$ **separates** $\bigcup_{t \in V_T^{(a)}} X_t$ and $\bigcup_{t \in V_T^{(b)}} X_t$.

The proof is almost the same as for path decomposition. We omit it here



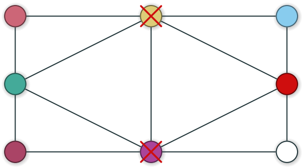


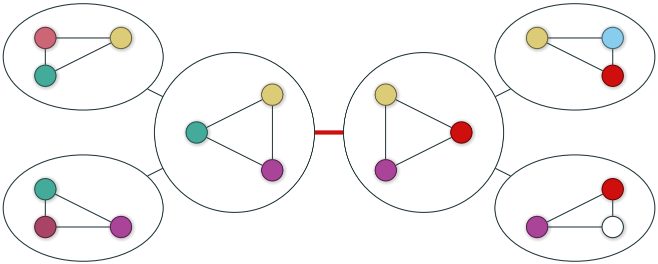
Separation

Lemma

Let $T = (V_T, E_T), (X_t)_{t \in V_T}$ be a tree decomposition of graph G. Let $(a, b) \in E_T$ be an edge of the decomposition and let $V_T^{(a)} \subseteq V_T$ ($V_T^{(b)} \subseteq V_T$) be the nodes of T on the side of a (resp., of b) of (a, b). Then there is no edge between $\bigcup_{t \in V_T^{(a)}} X_t \setminus (X_a \cap X_b)$ and $\bigcup_{t \in V_T^{(b)}} X_t \setminus (X_a \cap X_b)$. We say $X_a \cap X_b$ **separates** $\bigcup_{t \in V_T^{(a)}} X_t$ and $\bigcup_{t \in V_T^{(b)}} X_t$.

The proof is almost the same as for path decomposition. We omit it here





By fixing the choices in $X_i \cap X_{i+1}$ a problem usually splits into two independent subproblems

Uses of tree decompositions

- Similar to path decomposition, we can design dynamic programs over tree decompositions. The algorithm for Maximum Weight Independent Set generalizes in a straight-forward way, see exercises
- There is a large class of problems solvable in FPT time in $\mathrm{tw}(G)$, characterized by Courcelle's Theorem, see next lecture

Results for treewidth also imply to some other (easier to state) results. Some examples:

- Consider a planar graph G. Then $\mathrm{tw}(G) \leq O(\sqrt{n})$, see e.g. Corollary 7.24 from textbook. Many problems, e.g., Maximum Weight Independent Set, can be solved in time $2^{O(\mathrm{tw}(G))}n^{O(1)}$. Thus, on planar graphs such problems admit subexponential time algorithms with running time $2^{O(\sqrt{n})}$, even though these problems usually remain NP-hard also on planar graphs
- The treewidth of a graph is at most the size of the smallest vertex cover. Hence, an FPT algorithm for Vertex Cover parameterized by treewidth implies an FPT algorithm parameterized by solution size