CSL851: Algorithmic Graph Theory **Fall 2013**

Lecture 5: November 4

Lecturer: Amit Kumar *Scribes: Jatin Batra*

5.1 Recap of tree-width

$G = (V, E)$ has a **tree decomposition** (T,B) if T is a tree and B is a function from $V(T)$ to $2^{V(G)}$ (i.e. with every node x of T is associated a subset of vertices $B(x)$ of G) such that the following two properties hold

- $\forall\{\mathbf{u},\mathbf{v}\} \in \mathbf{E(G)}\ \exists \mathbf{x} \in \mathbf{V(T)}\ \mathbf{B(x)} \supseteq \{\mathbf{u},\mathbf{v}\}$ informally, every edge in the graph must be completely contained inside some node of T
- $\forall \mathbf{v} \in \mathbf{V(G)}\ \mathbf{B^{-1}(v)} = \{\mathbf{x}\ :\ \mathbf{v} \in \mathbf{B(x)}\}$ **forms a (connected) subtree of T** informally, the nodes in T containing any specific vertex of G must be all connected - we'll use this fact in the algorithms for graphs with bounded tree-width

then,

$$\textbf{tree-width of G} = \min_{\mathbf{(T,B)}} \max_{\mathbf{x} \in \mathbf{V(T)}} (|\mathbf{B(x)}| - \mathbf{1})$$

that is, for a given tree decomposition we consider the size of the largest $B(x) - 1$ (call this the tree-width of the tree decomposition) and pick the tree decomposition that minimizes this quantity; this makes sense because we want the tree decomposition to look as tree-like as possible.
In what follows, 'tree decomposition of a graph G' will mean the optimal tree decomposition in the sense just described. Also, the vertices of T shall be refered to as 'nodes' and the vertices of G shall be refered to as 'vertices' to avoid confusion. The phrase 'vertex v contained inside node x' shall mean $v \in B(x)$.

Exercise: Prove that G has tree-width 1 if and only if G is a tree.

5.2 Properties of tree-width

Lemma 5.1 *Suppose G has tree-width w. Then,*

1. *For any $e \in E(G)$, G' obtained by deleting e has tree-width atmost w.*
2. *For any $e \in E(G)$, G' obtained by contracting e has tree-width atmost w.*

Proof: Let (T, B) be a tree decomposition of G.

1. (T, B) is a valid tree decomposition for G' because any edge in G' is an edge in G and any vertex in G' is a vertex in G, and hence the two requirements for a tree decomposition are satisfied in G' as they're satisfied in G. Since the tree width of (T, B) is w, tree-width of $G' \leq w$.

2. Suppose we contract $e = \{u, v\}$ to a single vertex z. Keep T unchanged, and build B' as follows from B - whenever $u \in B(x)$ for any x, replace u with z; similarly for v. We claim that the new (T, B') is a tree decomposition for G'. Let us check the two conditions for tree decomposition -

   (a) Any edge $\{a, z\}$ in G' is contained inside the $B(x)$ for which $B'(x)$ contains the edge $\{a, u\}$ or $\{a, v\}$ in G'.

   (b) We must prove that $B'_{-1}(z)$ is connected. This can be seen from the following observations

      i. $B'^{-1}(z) = B^{-1}(u) \cup B^{-1}(v)$,

      ii. $B^{-1}(u)$ and $B^{-1}(v)$ are individually connected

      iii. $B^{-1}(u) \cap B^{-1}(v) \supseteq \{u, v\}$ (because u, v being an edge in G must be completely contained inside some node of T) and hence $B^{-1}(u) \cap B^{-1}(v)$ is nonempty.

■

Definition 5.2 *A graph G' is said to be a* ***minor*** *of a graph G if G' can be obtained from G by a sequence of edge deletions and edge contractions.*

By 5.1, if G' is a minor of G, then tree-width$(G) \geq$ tree-width(G'). This can be used to prove a lower bound on the tree-width of a graph.

Informally, the vertices of a minor H of a graph G corresponds to a partition $\mathcal{P}$ of the vertices of G such that there is a one-to-one correspondence between the vertices of H and the parts in $\mathcal{P}$, and the each part in $\mathcal{P}$ induces a connected subgraph of G. The edges in H are a subset of the edges between the corresponding parts in G. Here the part corresponding to a vertex v_H of H is the set of vertices of G which were collapsed (by contracting edges) to the vertex v_H, this part is connected because we can only collapse two vertices of G into a single vertex if they were connected by an edge. Here's a picture illustrating this

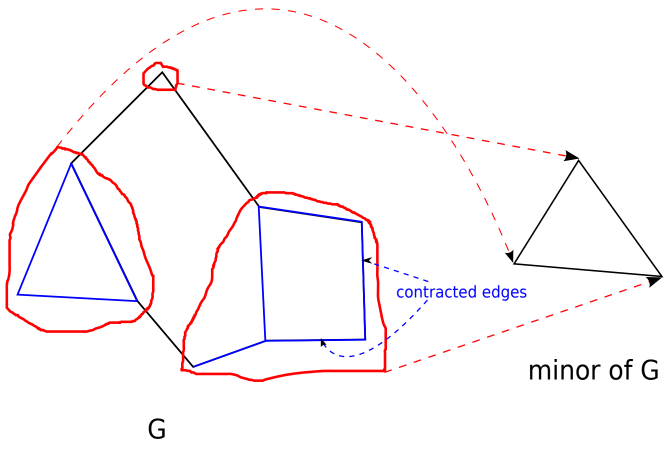

Figure 5.1: Intuition of minor

A graph property is closed under taking minors if every minor of a graph having the property also has the property. An example is planarity.

5.2.1 Planarity and minors

Planarity is closed under taking minors. We can verify that $K_{3,3}$ and K_5 are not planar, this means that any graph that contains K_5 or $K_{3,3}$ as a minor is not planar.

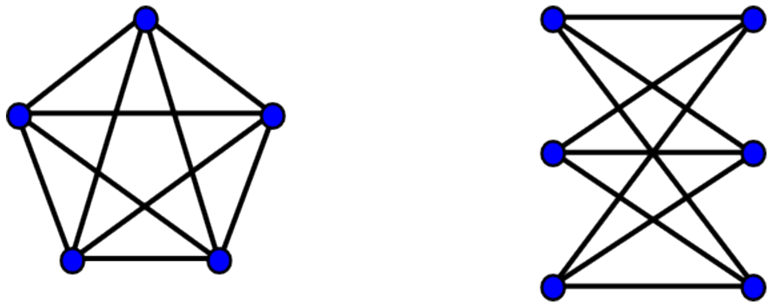

Figure 5.2: Obstructions to planarity

Kuratowski's theorem says that the converse is also true, that is, any non-planar graph contains $K_{3,3}$ or K_5 as a minor. A more general version of this theorem says that this idea is true for graphs drawn on any surface. This means that for any given surface (such a sphere or donut), there is finite set of forbidden minors. Hence, a given graph can be drawn on a given surface if and only if it does not contain as a minor any of the (finite number of) forbidden graphs for that surface.

The following much more general theorem of Robertson and Seymour says that this idea is true for any property (not just planarity) that is closed under taking minors

Theorem 5.3 *Let P be a property that is closed under taking minors. Then, $\exists$ a finite set of graphs $\mathcal{F}$ such that if a graph G does not have the property P, then $\exists H \in \mathcal{F}$ such that H is a minor of G.*

5.2.2 Series-Parallel graphs

Informally, a series-parallel graph is a graph that can be obtained by connecting electrical resistors in series and parallel. A formal definition is

Definition 5.4 *A graph G with a specified pair of vertices (s, t) is said to be a series-parallel graph if it can be obtained as follows*

1. ***base case*** *A graph consisting of a single edge is a series-parallel graph (with s, t being the two vertices).*
2. ***combining in series*** *If G_1 with s_1, t_1 and G_2 with s_2, t_2 (G_1 and G_2 are on disjoint sets of vertices) are series-parallel graphs, then so is the graph obtained by joining t_1 and s_2 with an edge.*
3. ***combining in parallel*** *If G_1 with s_1, t_1 and G_2 with s_2, t_2 (G_1 and G_2 are on disjoint sets of vertices) are series-parallel graphs, then so is the graph obtained by joining s_1, s_2 with an edge and t_1, t_2 with an edge.*

We will prove in this subsection that any series-parallel graph has tree-width atmost 2. First, we give an equivalent definition of series-parallel graphs that is easy to work with

Definition 5.5 *A graph G is a series-parallel graph if it can be obtained by applying a sequence of the following operations starting from a single edge*

1. *Add a self loop.*
2. *Add parallel edges.*
3. *Subdivide an edge.*

Here's a picture illustrating how any series-parallel graph (according to the first definition) can be obtained by these operations - we apply the reverse of the operations of the second definition on a series-parallel graph (which is clearly series-parallel by the first definition) and arrive at a single edge (and so the graph could have been constructed by applying the operations in the reverse of the order shown starting from a single edge).

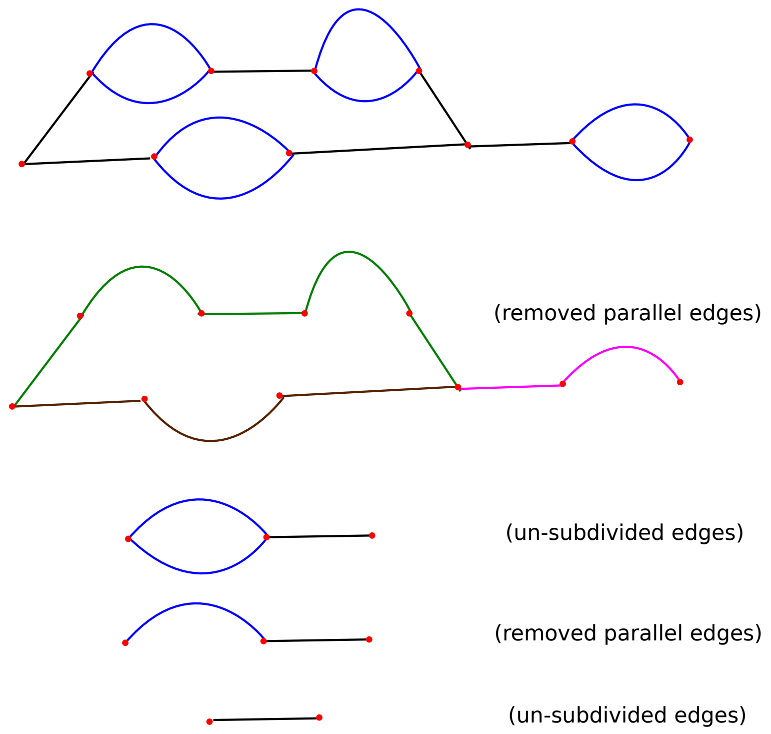

Figure 5.3: Construction of series parallel graphs

We will prove that a series-parallel graph has tree-width atmost 2 by induction on the number of operations used to construct it starting from a single vertex. Initially, the tree-width is clearly atmost 2. Adding parallel edges or self loops does does not increase tree-width (the argument is essentially the same as the one for 5.1). Hence, it suffices to prove the following lemma

Lemma 5.6 *If G' is obtained by subdividing an edge in G, $tree-width(G') \leq max\{2, tree-width(G)\}$.*

Proof:

Suppose we subdivide u, v to get a new vertex z and edges u, z and v, z. Start with the tree decomposition (T, B) of G. Since u, v is an edge in G, there is a node x in T with $u, v \subset B(x)$. Let T' be the same as T but with a new leaf y connected by an edge to x, and B' extended from B with $B'(y) = \{u, v, z\}$. This is a valid tree decomposition with tree width $max\{2, tree - width(G)\}$ since $|B'(y)| = 3$. ■

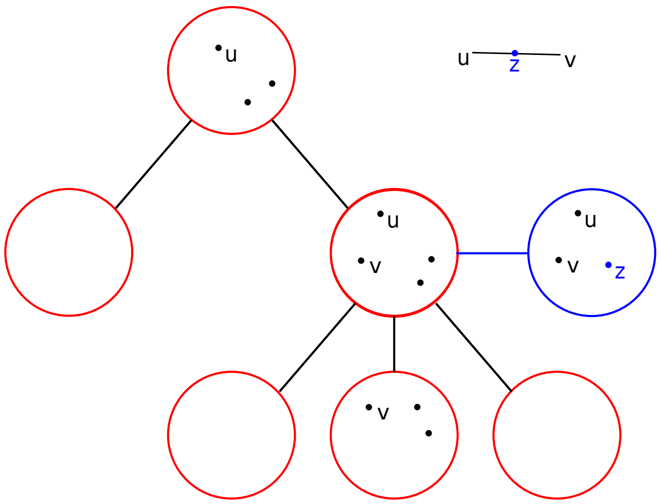

Figure 5.4: Proof of 5.6

Hence we have the following theorem

Theorem 5.7 *Tree-width of a series-parallel graph is atmost 2.*

5.2.3 Number of nodes in a tree decomposition

We prove that given a graph containing n vertices, any tree decomposition without redundancies contains atmost n nodes, where 'redundancy' is defined as follows

Definition 5.8 *Let $parent(x)$ denote the parent of a node x in a rooted tree. A node x in a tree decomposition (T, B) is said to be* redundant *if $B(x) \subseteq B(parent(x))$.*

If x is redundant, then we can obtain another valid tree decomposition by deleting x and adding the children of x to the children of $parent(x)$. Hence, we can obtain a tree decomposition without redundancies starting from any tree decomposition, in a natural way.
Observe that any node x with must contain a vertex $U(x)$ not present in its parent in a tree decomposition without redundancies (else x would be redundant) - if there are many choices for $U(x)$, choose one arbitrarily. The proof of the following lemma is based on the idea that if $v \notin B(z)$ but v is present somewhere in the subtree rooted at z, then v cannot be present in nodes not in the subtree rooted at z (this idea follows from the connectivity requirement in the definition of tree decomposition) as shown in the picture

Lemma 5.9 *If x, y are distinct nodes in a tree decomposition without redundancies, then $U(x) \neq U(y)$.*

Proof: Suppose $U(x) = U(y) = v$. Since the nodes containing v must form a connected set in T, every node of the only path in T connecting x and y must contain v. Without loss of generality, we can assume that

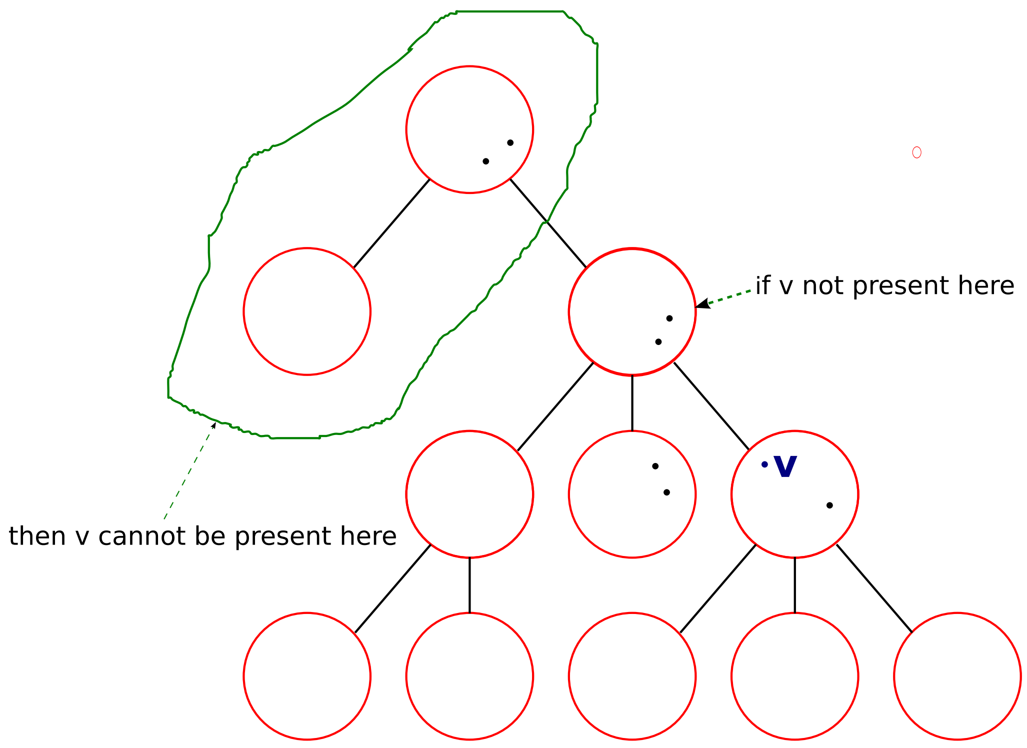

Figure 5.5: Proof of 5.9

x is not present in the subtree rooted at y (otherwise, we can switch the roles of x and y). But, then any path between x and y must pass through the parent of y, implying that $v = U(y) \in B(parent(y))$ which is a contradiction. ■

By pigeonhole principle, this implies that **there can be atmost n nodes in a tree decomposition without redundancies**. This fact is useful for algorithms on graphs that use tree decompositions.