1 Lecture 4, November 11, 1999

We shall touch upon an advanced topic – the theory of "Graph Minors" by Robertson and Seymour. This theory gives a very general condition under which a graph problem has a polynomial time algorithm (though the algorithms that come out of the theory are often not practical). We shall review a small part of this theory, and illustrate how dynamic programming can be used to solve some NP-hard problems on restricted classes of graphs (those of *bounded tree width*).

1.1 Dynamic programming on trees

As a motivating example to the forthcoming notion of *tree-width*, consider the maximum weight independent set problem on trees. The input is a tree $G(V, E)$ and a weight function $w : V \longrightarrow R$. (For simplicity in performing the arithmetic operations, think of the weights as nonnegative integers.) We wish to find an independent set (a set of vertices no two of which are adjacent) of maximum weight. This can be solved in polynomial time using dynamic programming as follows.

Fix an arbitrary vertex r as the root of the tree, and orient all edges away from r. The subtree rooted at v, denoted by $T(v)$, includes v and all vertices reachable from v under this orientation of edges. (Hence $T(r) = G$.) The *children* of v, denoted by $C(v)$ are all those vertices u with a directed edge (v, u). Leaves of the tree have no children (with a possible exception if we chose r to be one of the leaves of G – then it does have children).

Let $W(T(v))$ denote the maximum weight of an independent set of $T(v)$. We want to compute $W(T(r))$. Let $W^+(T(v))$ denote the maximum weight of an independent set of $T(v)$ that contains v, and let $W^-(T(v))$ denote the maximum weight of an independent set of $T(v)$ that does not contain v. Then we have

$$W^+(T(v)) = w(v) + \sum_{u \in C(v)} W^-(T(u))$$

and

$$W^-(T(v)) = \sum_{u \in C(v)} \max[W^-(T(u)), W^+(T(u))]$$

The dynamic program now works by repeating the following procedure:

1. Find a vertex v such that for all of its children u we have already computed $W^-(T(u))$ and $W^+(T(u))$.
2. Compute $W^-(T(v))$ and $W^+(T(v))$ as described above.

Eventually, output $\max[W^-(T(r)), W^+(T(r))]$. Note that as long as there are unvisited vertices, these vertices form a tree and hence have a leaf, and this leaf can be chosen in step 1 of the above procedure. Hence the algorithm never gets stuck.

Many NP-hard graph problems are polynomially solvable on trees. (There are exceptions. For example, bandwidth is NP-hard on trees.) We shall show how the class of graphs for which the above dynamic programming approach works (namely, trees) can be significantly generalize.

1.2 Tree decomposition

We now introduce the notion of a tree decomposition of a graph.

Definition 1 *A* tree decomposition *of a graph $G(V, E)$ is a tree T, where*

1. *each vertex i of T is labeled by a subset $H_i \subset V$ of vertices of G,*
2. *each edge of G is in a subgraph induced by at least one of the H_i,*
3. *for every three vertices $i, j, k \in T$ with j lying on the path from i to k in T, $H_i \bigcap H_k \subset H_j$.*

Remarks:

1. The condition $H_i \bigcap H_k \subset H_j$ is equivalent to the condition that for every $v \in V$, the set of vertices in T that have v as part of their label is connected in T.
2. We shall assume that no two vertices $i, j \in T$ have $H_i \subset H_j$. (Otherwise, merge i and j if they are adjacent, or i and its nearest neighbor on the path to j if they are not.)
3. A graph may have several different tree decompositions. Similarly, the same tree decomposition (as we defined it) can correspond to several different graphs. We can define H_i to be not just a set of vertices but also the set of induced edges, and then a tree decomposition uniquely determines a graph.

Every graph has a trivial tree decomposition for which T has one vertex including all of V. This partly motivates the following definition:

Definition 2 *The* tree-width *of G is the minimum integer p such that there exists a tree decomposition G with all subsets of cardinality at most $p + 1$.*

Observe that the tree-width of a graph is equal to the maximum of the tree-widths of its connected components. Another simple observation is the following:

Lemma 1 *Every graph with tree-width p has a vertex of degree at most p.*

Proof: For graph G, consider a tree decomposition T with width p. Consider a leaf l of T. Its label includes a vertex v that is not in the label of its neighbor in T, and hence not in any other label either. As $|H_l| \leq p+1$, it follows that the degree of l in G is at most p. $\square$

We can exactly characterize those graphs having tree-width $n - 1$.

Proposition 2 *An n-vertex graph has tree-width $n - 1$ iff it is a clique.*

Proof: If the graph is not a clique, then some edge (u, v) is missing. In this case, we can take $H_1 = V \setminus \{u\}$ and $H_2 = V \setminus \{v\}$ as a two node tree decomposition of tree-width $n - 2$.

Conversely, if G is connected and has a tree decomposition with tree-width below $n-1$, then it has a vertex of degree at most $n - 2$, and cannot be a clique. $\square$

We shall be interested in graphs with small tree-width. Clearly, the only graphs with tree-width 0 are independent sets.

Proposition 3 *G has tree-width at most 1 iff it is a forest.*

Proof: Consider an arbitrary tree G. We build a tree decomposition for it with tree-width 1 as follows. Pick a root r in G, and orient the edges away from r. Each oriented edge (u, v) in G makes one node labeled $\{u, v\}$ in T. (The label $\{u, v\}$ is equivalent to the label $\{v, u\}$, but we shall keep the orientation of the edge for ease of description.) Two nodes $\{u_1, v_1\}$ and $\{u_2, v_2\}$ are adjacent in T iff $v_1 = u_2$. It can be seen that this gives a tree decomposition of G.

Conversely, assume that G has tree-width 1. Then it has a vertex of degree at most 1. Removing this vertex from G, the tree-width cannot increase. Hence we can recursively remove vertices of degree at most 1, until we exhaust all vertices in G. This implies that G is a forest. □

Definition 3 *A* series-parallel *graph is a graph obtained from an independent set using the following operations:*

1. *Add a new node and connect it to an existing node by an edge.*
2. *Add a self loop.*
3. *Add an edge in parallel to an existing edge.*
4. *Subdivide an edge by creating a node in the middle.*

Proposition 4 *G has tree-width at most 2 iff it is a series-parallel graph.*

Proof: Let G be a series-parallel graph. We build a tree decomposition for it inductively, without ever exceeding tree-width 2. We do this by following the inductive construction of G. The most involved step is step 4 in the construction of series parallel graphs. Suppose edge (u, v) was subdivided, introducing a vertex w. Prior to the subdivision, the tree decomposition included a vertex j such that $u, v \in H_j$. Create a new node labeled by $\{u, v, w\}$ and connect it to j.

Conversely, assume a tree decomposition T with tree-width at most 2. We show that this corresponds to a series-parallel graph. We do this by removing a node from the tree decomposition, show that this corresponds to removing a vertex from the graph G, and then show that G can be obtained by adding back the vertex using one of the rules of Definition 3. Again we illustrate only one case. Take a leaf l of T. Assume that it is labeled by three vertices u, v, w (it cannot be labeled by more than three vertices). Then w.l.o.g., the neighbor of l in T does not contain w. Hence the only possible neighbors of w in G are u and v. If both are indeed neighbors, then w could have been obtained by subdividing an edge (u, v), and the graph prior to subdividing also had tree width at most 2 (by removing w from l). □

The notion of tree-width connects us to the notion of graph minors.

Definition 4 *H is a* minor *of G if it can be obtained from G by a sequence of operations of edge deletions, vertex deletions, and edge* contractions *(merging endponts together).*

A planar graph is a graph that can be drawn in the plane with no two edges crossing. Kuratowski showed that a graph is non-planar iff it contains either K_5 (a clique on five vertices) or $K_{3,3}$ (a complete bipartite graph on six vertices) as a minor. We observe that a graph is not a forest iff it contains K_3 as a minor. It is known (we shall not prove it) that a graph is not series-parallel iff it contains K_4 as a minor. (In particular, K_4 is a planar graph that is not series-parallel.) As forests have tree-width at most 1 and series-parallel graphs have tree-width at most 2, one may hope that graphs not containing K_5 as minors have tree-width at most 3, especially as K_5 itself has tree-width 4. However, this is false.

Proposition 5 *The tree width of planar-graphs can be as high as $\Omega(\sqrt{n})$.*

We shall use the following important lemma in the proof of Proposition 5.

Lemma 6 *Let $G(V, E)$ be a graph of tree width p. Assume that the vertices $v \in V$ have nonnegative weights $w(v)$ with $\sum_{v \in V} w(v) = 1$. Then there is a set S of at most $p+1$* separating *vertices whose removal separates G into two parts S_1 and S_2 (one of which may be empty) with the following properties:*

1. *Each part has total weight no more than $2/3$.*
2. *All paths from S_1 to S_2 must go through S.*

Proof: Let T be a tree decomposition of G with tree-width p. Pick a root r in T and orient edges away from r. For a subtree of T rooted at node v, let $W(v)$ be the sum of the weights of the distinct vertices in the labels of the subtree rooted at v. Observe that $W(r) = 1$. Let j be a vertex with $W(j) \geq 1/2$ and $W(u) < 1/2$ for all children u of j. Let $S = H_j$. Then $|S| \leq p+1$. Now make j the root of T. For child u of j in T, define $S(u)$ to be the vertices of G appearing in the labeles of the subtree rooted at u but not in S. Then the $S(u)$ are disjoint, and any path connecting a vertex from $S(u)$ to a vertex in $S(u')$, where $u \neq u'$ and both u and u' are children of j, must go through S. Let $W'(u) = \sum_{v \in S(u)} w(v)$. Then $W'(u) < 1/2$. If for some u, $W'(u) \geq 1/3$, then take $S_1 = S(u)$ and $S_2 = V \setminus S(u) \setminus S$. If $W'(u) < 1/3$ for all u, then we can combine several $S(u)$ to get a set S_1 of weight between $1/3$ and $2/3$. $\square$

We now proof Proposition 5.

Proof: Consider a $\sqrt{n}$ by $\sqrt{n}$ mesh, and let its tree-width be p. Consider only the set B of $\sqrt{n}$ nodes of the bottom row of the mesh, give each one of them weight $1/\sqrt{n}$, and every other vertex weight 0. By Lemma 6 there is a set S of $p+1$ vertices whose removal breaks the mesh into two components S_1 and S_2, none of which contains more than $2\sqrt{n}/3$ vertices from B. If $p < \sqrt{n}/6$, then each component contains at least $\sqrt{n}/6$ vertices from B. On the bottom row of the mesh there must be a point b to the left of which there are at least $\sqrt{n}/12$ vertices $l_1, l_2 \ldots$ from S_1, and to the right of which there are $\sqrt{n}/12$ vertices $r_1, r_2, \ldots$ from S_2 (or vice versa), where vertices are numbered in order of increasing distance from b. Consider the $\sqrt{n}/12$ pairs (l_i, r_i). There are disjoint paths between them (for pair i, go straight up and connect through row i). As all these paths must go through S, $p \geq \sqrt{n}/12$. (The constants in this proposition can be improved.) $\square$

Tree-width is connected to planarity via the following theorem of Robertson and Seymour, whose proof is beyond the scope of the course.

Theorem 7 *Let H be a planar graph. If G has no H-minor, then the tree width of G is bounded by some function of H, independent of G.*

(Homework in Handout 3)

References: The work of Roberston and Seymour has been published in a long sequence of papers (can be found via a search on a web database, such as MathSciNet). The book "Parameterized Complexity" by R.G. Downey and M.R. Fellows, Springer series of Monographs in Computer Science, 1999, should include a good exposition of the subject, as well as the use of dynamic programming for fixed parameter optimization problems in general. Hopefully, our library will have a copy of this book soon.