1 Lecture 5, November 18, 1999

We continue with the study of tree-width and graph minors.

1.1 Coloring

Recall that a legal k-coloring of a graph colors the vertices with k colors such that adjacent vertices receive different colors. The *chromatic number* of a graph is the minimum k that allows a legal k-coloring. Computing the chromatic number is NP-hard.

Proposition 1 *Every graph with tree-width p has chromatic number at most $p+1$.*

Proof: The proof is via the method of *inductive coloring*. We have seen that a graph $G(V,E)$ of tree-width p has a vertex v with degree at most p. If the rest of the graph is legally colored by $p+1$ colors then we can also $(p+1)$-color G by assigning to v one of the colors not assigned to its neighbors. As the subgraph induced on $V \setminus \{v\}$ also has tree-width at most p, the proof follows by induction. $\Box$

Though planar graphs have unbounded tree-width, the method of inductive coloring does apply to them. As every planar graph has a vertex of degree at most 5, planar graphs are 6-colorable. By the fact that planar graphs do not contain K_5, it follows that they are 5-colorable. Consider the five neighbors of a vertex v of degree five. Two of them are not adjacent, and can be contracted to v while preserving planarity, giving a graph G'. A legal 5-coloring of G' can be extended to a 5-coloring of G because now two of the neighbors of v have the same color.

The famous *4-color theorem* says that every planar graph is 4-colorable. Current proofs of this theorem are complicated and use extensive computer assisted case analysis.

Hadwiger conjectured that every graph that does not contain K_{p+1} as a minor is legally p-colorable. This conjecture is still open. If true, it would generalize the 4-color theorem. A weaker form of the conjecture is known to hold. (Namely, every graph that does not contain K_p as a minor is legally $f(p)$-colorable, where $f(p)$ is a slowly growing function of p. In particular, $f(p) \leq O(p\sqrt{\log p})$. The proof is based on the fact that inductive coloring of G fails only if there is a subgraph of high minimal degree, but then this subgraph must have a large clique as a minor.)

Deciding whether a planar graph is 3-colorable is NP-hard. On the other hand, for graphs of bounded tree-width, their chromatic number can be determined in polynomial time. This can be done via dynamic programming on the tree decomposition of the graph. We use the following theorem, whose proof is beyond the scope of the course:

Theorem 2 *For graphs that have bounded tree-width, a corresponding tree decomposition can be found in polynomial (in fact, linear) time.*

Given a tree decomposition T of tree-width p for k-colorable graph G, we legally color G with k colors as follows. Pick an arbitrary root r for T and orient edges away from r. For vertex j in T, let $T(j)$ denote the subtree of T rooted at j, and let $V(j)$ denote the set of vertices of G that label nodes in $T(j)$. We use the fact that $|H_j| \leq p+1$. For each vertex $j \in T$, we maintain an indicator vector c_j of length k^{p+1}. Each entry corresponds to one possible (not necessarily legal) k-coloring of the vertices of H_j. This entry is 1 if the

k-coloring can be completed to a legal k-coloring of the subgraph induced on $V(j)$, and 0 otherwise. Note that the vector c_j can be completely determined from the vectors c_u of all the children u of j, and from H_j, because for a coloring of $V(j)$ to be legal, it suffices that it is legal on H_j and on $V(u)$ for each child u of j *separately* – there is no interaction between the children of j except through edges present in H_j. Hence using dynamic programming we can eventually compute c_r, and if c_r has a nonzero entry then G is k-colorable.

The running time of the algorithm can be estimated as follows. Every tree decomposition T of graph G has at most n vertices. (A leaf l of T has in its label a vertex v of G not appearing anywhere else in T. Removing v from G, the tree T is updated by removing at most the leaf l. Continue recursively.) Hence T has at most n edges. At each edge (i, j) we look at most k^{p+1} times (once for each coloring of its parent vertex i). Each time we look at an edge we at most scan the whole indicator vector c_j of length k^{p+1} of the child vertex j. As $k \leq p + 1$, the running time is proportional to $np^{O(p)}$ (assuming that a tree decomposition is given). Note that for fixed p, this running time is linear (assuming fixed-time random access to the adjacency matrix of G).

Thus we see that dynamic programming algorithms that work on trees often extend to graphs of bounded tree-width. To appreciate this extension, observe that graphs with bounded tree-width need not "look like" trees (e.g., series-parallel graphs).

1.2 A general polynomial time algorithm

Robertson and Seymour show a general condition under which a graph property has a polynomial time algorithm. We review here without proofs some of the main ingredients of their theory.

Define a partial order on graphs such that $H \leq G$ iff H is a minor of G (isomorphic graphs are considered identical). A family $\mathcal{F}$ of graphs is closed w.r.t. taking minors if $G \in \mathcal{F}$ implies $H \in \mathcal{F}$ for every $H < G$. There are several ways of expressing a minor closed family $\mathcal{F}$:

1. Via a graph property. For example, the family of planar graphs, the family of graphs of tree-width at most 2, etc. Sometimes such a representation is not known.

2. As an explicit list of graphs. This representation might be infinite.

3. By a list $\mathcal{O}$ of graphs not in $\mathcal{F}$. Again, this representation might be infinite.

4. Note that if $H \in \mathcal{O}$ then H cannot be a minor of a graph in $\mathcal{F}$. Hence we may equivalently think of $\mathcal{O}$ as a list of *forbidden minors* – those graphs that never appear as minors of graphs in $\mathcal{F}$.

5. The list $\mathcal{O}$ may be redundent in the sense that it might contain two graphs H_1 and H_2 with $H_1 < H_2$. Removing H_2 from $\mathcal{O}$, we can still define $\mathcal{F}$ as those graphs that do not have minors in $\mathcal{O}$. Removing graphs from $\mathcal{O}$ in this way, we remain with a list $\mathcal{O}$ of forbidden minors that is minimal – no forbidded minor is contained in another. For example, for planar graphs $\mathcal{O} = K_5, K_{3,3}$, and for graphs of tree-width at most 2, $\mathcal{O} = K_4$.

Let us study more carefully the forbidden minor representation. In what sense is it better than the explicit list of graphs representation? For planar graphs, we see an obvious advantage – the list of forbidden minors is finite (just two graphs). But is this a coincidence? This brings us to Wagner's conjecture.

A partial order is a *well-quasi-order* (wqo) if it does not contain an infinite *antichain*, where an antichain is a set of elements no two of which are comparable. Wagner conjectured that the minor-induced partial order for graphs is a wqo. Robertson and Seymour proved this conjecture. It follows that *every minor closed family* can be repesented by a finite list of forbidden minors (the *obstruction set*). We note that the theorem of Robertson and Seymour does not tell us how to find the obstruction set – it just proves its existence. For planar graphs we know the obstruction set. For many other minor-closed families we do not know it.

Robertson and Seymour go a step further. They show that the following problem has a polynomial time algorithm for every fixed k:

Disjoint paths: given an undirected graph G and k pairs of vertices $(s_1, t_1), \dots (s_k, t_k)$, are there vertex disjoint paths connecting s_i and t_i for every pair?

To appreciate this algorithm, note that the same problem is NP-hard on directed graphs, even when $k = 2$.

Using the algorithm for disjoint paths, there is an algorithm that for every fixed H checks whether H is a minor of $G(V, E)$. The running time of the algorithm is polynomial, $O(|V|^3)$, where the leading constant depends only on H (in a way that grows very rapidly with H). When H is part of the input, the problem is NP-hard.

This leads to the following general theorem:

Theorem 3 *Every property of graphs that is inherited by minors (such as being embeddable on a surface of genus g, having a linkless embedding in 3-dimensional space, etc.) can be* decided *in polynomial time.*

The algorithm implied by the theorem is the list of forbidden minors for the infinite family of graphs having the graph property, coupled with the algorithm for checking minor inclusion. As remarked earlier, the theory just shows the existence of a polynomial time algorithm, but does not necessarily produce such an algorithm. Moreover, the algorithm just answers *yes* or *no*, without necessarily supplying a witness in the usual sense (e.g., the algorithm may decide that the graph has a linkless embedding without actually finding one).

References:

- Ton Kloks. "Treewidth – Computations and Approximations". Lecture notes in Computer Science 842, Springer-Verlag, Berlin, 1994.