TREE-DECOMPOSITIONS OF GRAPHS

lecture notes

Robin Thomas*
School of Mathematics
Georgia Institute of Technology
Atlanta, Georgia 30332, USA

ABSTRACT

Tree-decompositions of graphs play an important role in graph structure theory, in the theory of algorithms and in practical computation. These notes survey the first two aspects.

Revised 3 June 1996

* Supported in part by NSF under Grant No. DMS-9303761, and by ONR under Contract number N00014-93-1-0325. This is essentially one chapter of my Structural Graph Theory class notes.

A. PRELIMINARIES

A1. Definitions. Graphs are undirected, and may have loops and multiple edges. More precisely, a *graph* G consists of a set of *vertices* $V(G)$, a set of *edges* $E(G)$ and a set of *incidences* between vertices and edges. Every edge is incident with one or two vertices. If it is incident with one vertex it is called a *loop*, if it is incident with two vertices it is called a *link*. Two edges are *parallel* if they have the same endpoints. A graph is *simple* if it has no loops and no parallel edges. *Paths* and *circuits* are subgraphs with the expected properties; they both have no "repeated" vertices and no "repeated" edges. (What we call a circuit is sometimes called a cycle, but we prefer this term because of matroid analogy.) If G is a graph and X is a vertex, a set of vertices, an edge, or a set of edges, then $G\backslash X$ denotes the graph that results when X is deleted from G.

A2. Definition. Let G be a graph, and let $e \in E(G)$. As explained earlier, $G\backslash e$ denotes the graph obtained from G by *deleting* e. By G/e we denote the graph obtained from G by *contracting* e, where contraction is defined as follows. If e is a loop then G/e is defined to be $G\backslash e$; otherwise G/e is obtained by deleting e and identifying the endpoints of e. Thus $|E(G/e)| = |E(G\backslash e)| = |E(G)| - 1$, and if e is link then $|V(G/e)| = |V(G)| - 1$. A graph H is a *minor* of a graph G if H can be obtained from a subgraph of G by repeatedly contracting edges. We say that G has an *H–minor* if G has a minor isomorphic to H.

A3. Exercise. A graph G has an H–minor if and only if a graph isomorphic to H can be obtained from G by applying the following operations:

(i) Deleting an edge,

(ii) contracting an edge,

(iii) deleting an isolated vertex.

The order in which these operations are carried out does not matter. Moreover, if G is connected, then (iii) is not needed.

A4. Exercise. A graph G has an H–minor if and only if for every $v \in V(H)$ there exists a connected non-null subgraph G_v of G and a mapping $\psi : E(H) \to E(G)$ such that

(i) for distinct $u, v \in V(H)$, G_u and G_v are vertex–disjoint,

(ii) ψ is 1–1 and $\psi(e) \notin E(G_v)$ for every $e \in E(H)$ and $v \in V(H)$, and

(iii) if $e \in E(H)$ has endpoints u, v, then $\psi(e)$ has one endpoint in G_u and the other in G_v.

A5. Definition. Consider the following cops–and–robbers game. The robber stands on a vertex of the graph and can at any time run at great speed to any other vertex along a path of the graph. He is not permitted to run through a cop, however. There are k cops, each of whom at any time either stands on a vertex or is in a helicopter (that is, is temporarily removed from the game). The objective of the player controlling the movement of the cops is to land a cop via helicopter on the vertex occupied by the robber, and the robber's objective is to elude capture. (The point of the helicopters is that cops are not constrained to move along paths of the graph — they move from vertex to vertex arbitrarily.) The robber can see the helicopter approaching its landing spot and may run to a new vertex before the helicopter actually lands. Thus, for the cops to capture the robber they will need to first occupy all vertices adjacent to the vertex where the capture is to take place, because otherwise the robber will be able to run to a different vertex and not be captured.

There are two forms of this game of interest. In the first, the robber is invisible, and so to capture him the cops must methodically search the whole graph. (An equivalent problem is that of clearing the vertices of some plague which infects along edges.) In the second form of the game the cops can see the robber at all times — the difficulty is just to corner him somewhere.

A6. Exercise. The result of the game does not change if we delete loops and multiple edges.

A7. Exercise. Two cops can capture a visible robber in a simple graph G if and only if G is a forest.

A8. Exercise. Two cops can capture an invisible robber in a simple graph G if and only if every component of G is a caterpillar, that is, a path with pendant edges attached to some of its vertices.

B. TREE–WIDTH

B1. Definition. [8] Let G be a graph. A *tree–decomposition* of G is a pair (T, W), where T is a tree and $W = (W_t : t \in V(T))$ is such that
(i) $\bigcup_{t \in V(T)} W_t = V(G)$, and every edge of G has both endpoints in some W_t, and
(ii) if $t, t', t'' \in V(T)$ and t' lies on the path from t to t'' in T, then $W_t \cap W_{t''} \subseteq W_{t'}$.

We define the *width* of (T, W) to be

$$\max\{|W_t| - 1 : t \in V(T)\},$$

and the *order* of (T, W) to be 0 if $E(T) = \emptyset$ and

$$\max\{|W_t \cap W_{t'}| : t, t' \text{ adjacent in } T\}$$

otherwise.

B2. Exercise. [8] Let (T, W) be a tree–decomposition of G, let H be a connected subgraph of G, and let $t, t'' \in V(T)$ be such that $V(H) \cap W_t \neq \emptyset \neq V(H) \cap W_{t''}$. Prove that if $t' \in V(T)$ lies on the path of T between t and t'', then $V(H) \cap W_{t'} \neq \emptyset$.

B3. Exercise. [8] Let (T, W) be a tree–decomposition of G, and let $e \in E(T)$ with endpoints t_1, t_2. Then $T \backslash e$ has exactly two components T_1, T_2 say. Prove that there is no edge between $\bigcup_{t \in V(T_1)} W_t - (W_{t_1} \cap W_{t_2})$ and $\bigcup_{t \in V(T_2)} W_t - (W_{t_1} \cap W_{t_2})$. In other words, if both of these sets are non–empty, then $W_{t_1} \cap W_{t_2}$ is a cutset.

B4. Exercise. Let (T, W) be a tree–decomposition of a graph G, and let H be a complete subgraph of G. Prove that there exists $t \in V(T)$ such that $V(H) \subseteq W_t$.

B5. Definition. Let G be a graph. The *tree–width* of G is the least integer k such that G has a tree–decomposition of width k.

B6. Exercise. Prove that if G has an H–minor, and G has tree–width at most k, then H has tree–width at most k.

B7. Exercise. Prove that a simple graph G has tree–width at most 1 if and only if G is a forest.

B8. Exercise. Prove that a graph G has tree–width at most 2 if and only if G has no K_4–minor.

B9. Exercise. Let $n \geq 0$ be an integer. Prove that K_n has tree–width $n - 1$.

B10. Exercise. Prove that if G has tree–width at most k, then $k + 1$ cops can capture a visible robber in G.

Hint. Mimic the proof for trees.

B11. Exercise. Let $n \geq 2$ be an integer. Prove that the $n \times n$ grid (the adjacency graph of the $n \times n$ chess board) has tree–width n.

Hint. Use the previous exercise for a lower bound.

B12. Exercise. [6] A graph G is chordal if and only if G has a tree–decomposition (T, W) such that every W_t is a clique.

B13. Exercise. For a graph G and an integer k, the following are equivalent.
(i) G has tree–width at most k, and
(ii) G is a subgraph of a chordal graph with no subgraph isomorphic to K_{k+2}.

B14. Exercise. [11] For every planar graph H there exists an integer n such that the $n \times n$ grid has a minor isomorphic to H.

B15. Theorem. *For every planar graph H there exists an integer k such that if a graph G has tree–width at least k, then it has an H–minor.*

B16. Exercise. If H is non–planar then no such integer exists.

B17. Remark. The above theorem was first proved in [9]. A simpler proof with a better bound on k was given in [11]; the bound there is $k = 20^{2n^5}$, where $n = 2|V(H)| + 4|E(H)|$. It seems to be an interesting and difficult problem to decide if k is bounded by some polynomial in n.

B18. Exercise. Let H be the $n \times n$ grid, and let k be as in the above theorem. Prove that $k \geq \Omega(n^2 \log n)$.

Hint. Random graphs.

C. TREE-WIDTH AND ALGORITHMS

C1. Theorem. [2] *Computing tree–width is NP–hard.*

C2. Theorem. [4] *For every fixed integer k there is a linear-time algorithm to decide if an input graph has tree-width at most k.*

C3. Theorem. [12] *There exists a polynomial-time algorithm that, given a graph G and an integer k, either finds a tree-decomposition of G of width at most k^4, or (correctly) answers that G has tree-width at least k.*

C4. Remark. For every fixed integer k there is a dynamic programming algorithm with running time $O(|V(G)|^{k+2})$ to test if the input graph has tree–width $\leq k$, and if so, to find a tree–decomposition of width $\leq k$ [2]. This algorithm is relatively easy to implement.

C5. Remark. An important feature of tree-width is that many NP–hard problems can be solved in linear time if the input graphs come with a tree–decomposition of width at most k, for some fixed integer k. Here is an outline of the general method.

C6. Exercise. Let k be a fixed integer, let G be a graph, and let $Z \subseteq V(G)$ with $|Z| \leq k+1$. Suppose we want to compute certain information $P(G, Z)$ about G and Z, and this information satisfies the following axioms:

(i) $P(G, Z)$ can be computed in constant time if $|V(G)| \leq k+1$,

(ii) if $Z' \subseteq Z$ then $P(G, Z')$ can be computed from the knowledge of $P(G, Z)$ in constant time, and

(iii) if (G_1, G_2) is a separation of G with $V(G_1) \cap V(G_2) \subseteq Z$, then $P(G, Z)$ can be computed from the knowledge of $P(G_1, Z \cap V(G_1))$ and $P(G_2, Z \cap V(G_2))$ in constant time.

Then, given a tree–decomposition of G of width at most k, $P(G, \emptyset)$ can be computed in linear time.

C7. Exercise. For every fixed integer k, the independence number of G can be computed in linear time if a tree–decomposition of G of width at most k is given.

Hint. For $A \subseteq Z$, let α_A be the maximum cardinality of an independent set $I \subseteq V(G)$ with $I \cap Z = A$ (or zero if no such set exists). Let $P(G, Z) = (\alpha_A : A \subseteq Z)$.

C8. Exercise. For every fixed integer k, the chromatic number of G can be computed in linear time if a tree–decomposition of G of width at most k is given.

C9. Exercise. For every fixed integer k and every fixed graph H, given an input graph G and a tree–decomposition of G of width at most k, one can determine whether G has an H–minor in linear time.

D. A MIN-MAX THEOREM FOR TREE-WIDTH

D1. Definition. [13] Let G be a graph and let $k \geq 0$ be an integer. If $X \subseteq V(G)$ then by an *X–flap* we mean the vertex-set of a component of $G\backslash X$. An *escape* in G of order k is a function σ which assigns, to every $X \subseteq V(G)$ with $|X| < k$, a set $\sigma(X) \subseteq V(G)$ such that for every $X, Y \subseteq V(G)$ with $|X|, |Y| < k$

(i) $\sigma(X)$ is a union of X–flaps of G,

(ii) $\sigma(X) \neq \emptyset$, and

(iii) if $X \subseteq Y$ then $\sigma(Y)$ is the union of precisely those Y–flaps that intersect $\sigma(X)$.

Escapes correspond to winning strategies for the robber.

D2. Exercise. A graph G has an escape of order k if and only if $k - 1$ cops cannot capture a visible robber.

D3. Definition. [13] Let G be a graph, and let $k \geq 0$ be an integer. A *haven* of order k in G is a function assigning to every set $X \subseteq V(G)$ with $|X| < k$ an X–flap $\beta(X)$ such that if $X \subseteq Y \subseteq V(G)$ and $|Y| < k$, then $\beta(Y) \subseteq \beta(X)$.

D4. Exercise. Every haven is an escape.

D5. Theorem. (Min-max theorem for tree-width [13]) *Let $k \geq 0$ be an integer. A graph has a haven of order k if and only if it has tree–width at least $k - 1$.*

D6. Exercise. Prove only if.

D7. Exercise. Let G be a graph, and let $k \geq 0$ be an integer. Then the following are equivalent.

(i) G has tree–width at least $k - 1$,

(ii) G has an escape of order k,

(iii) $k - 1$ cops cannot capture a visible robber in G, and

(iv) G has a haven of order k.

D8. Remark. The decision problem whether the tree–width of G is at most k is obviously in NP. The min-max theorem comes close to proving that it is in co–NP, but not quite. In fact, it is not even clear whether a haven can be specified in polynomial space. Since computing tree–width is NP–hard, it is unlikely that it is in co–NP.

E. BRANCH-WIDTH

Branch-width is a variant of tree–width that can be extended to matroids.

E1. Definition. [10] A tree T is *ternary* if every vertex of T has valency three or one. The vertices of valency one are called the *leaves* of T. Let G be a graph. A *branch–decomposition* of G is a pair (T, τ), where T is a ternary tree, and τ is a bijection between the set of leaves of T and $E(G)$. Let $e \in E(T)$, and let T_1, T_2 be the two components of $T \backslash e$. For $i = 1, 2$, let E_i be the set of all $\tau(t)$ for all leaves t of T with $t \in V(T_i)$. Then $E(G) = E_1 \cup E_2$, and we define the *order* of e to be the number of vertices of G incident with both E_1 and E_2. We define the *width* of (T, τ) to be the maximum order of the edges of T (or zero if $|E(G)| \leq 1$), and we define the *branch–width* of G to be the least integer $k \geq 0$ such that G has a branch–decomposition of width k, or zero if $|E(G)| \leq 1$, in which case G has no branch-decomposition.

E2. Exercise. If G has an H–minor, then the branch–width of H is at most the branch–width of G.

E3. Exercise. A graph G has branch–width zero if and only if every component of G has at most one edge.

E4. Exercise. A graph G has branch–width at most one if and only if every component of G has at most one vertex of valency more than one.

E5. Exercise. A graph G has branch–width at most two if and only if G is series–parallel.

E6. Exercise. [10] For $n \geq 3$ the complete graph K_n has branch–width $\lceil \frac{2}{3} n \rceil$.

E7. Exercise. [10] Let G be a graph, let t be the tree–width of G, and let b be the branch–width of G. If $b \geq 2$ then

$$b \leq t + 1 \leq \left\lfloor \frac{3}{2} b \right\rfloor .$$

E8. Definition. Let M be a matroid. We define the *branch–decomposition* of M similarly as for graphs. We define the *order* of an edge $e \in E(T)$ to be $\kappa(E_1) = \kappa(E_2)$, where κ is the connectivity function of M and E_1, E_2 are as in E1, and then we define the branch–width in the same way as for graphs.

E9. Exercise. Let G be a graph. Prove that G has branch–width at most two if and only if $\mathcal{M}(G)$ has branch–width at most two. Notice that the two definitions use different definitions of an order of an edge.

E10. Open problem. Let G be a graph with branch–width at least two. Is it true that G and $\mathcal{M}(G)$ have the same branch–width?

F. PATH-WIDTH

F1. Definition. [7] A tree–decomposition (T, W) of a graph G is a *path–decomposition* of G if T is a path. The *path–width* of G is the least integer k such that G has a path–decomposition of width k.

F2. Exercise. Let T_k be the ternary tree of height k. Prove that the path–width of T_k tends to ∞ as $k \to \infty$.

F3. Exercise. If G has an H–minor, then the path–width of H is at most the path–width of G.

F4. Theorem. [3] *If F is a forest and G is a graph of path-width at least $|V(F)| - 1$, then G has an F–minor.*

F5. Exercise. If F is not a forest, then no such bound exists.

F6. Exercise. The bound $|V(F)| - 1$ is best possible.

F7. Remark. Theorem F4 was originally proved by Robertson and Seymour in [7] with a much larger bound. The proof in [3] is easier. Diestel [5] found a yet simpler proof.

F8. Theorem. [2] *Computing path-width is NP–hard.*

F9. Definition. Let G be a graph. We define the *linear–width* of G to be the least integer $k \geq 0$ such that the edges of G can be numbered $e_1, e_2, \ldots, e_m$ in such a way that for every $i = 1, 2, \ldots, m-1$, there are at most k vertices incident with both $\{e_1, e_2, \ldots, e_i\}$ and $\{e_{i+1}, \ldots, e_m\}$. This again can be extended to matroids.

F10. Exercise. What is the relation between path–width and linear–width?

F11. Remark. Linear-width can be extended to matroids, but what should the analogue of F4 be? The next result suggests a possible approach.

F12. Theorem. [1] *Let Q be an outerplanar graph, and let R be a graph such that the deletion of some vertex of R produces a forest. For every such pair Q and R there exists an integer k such that every 2-connected graph of path-width at least k has a Q-minor or an R-minor.*

F13. Definition. A graph G is an *interval graph* if for every $v \in V(G)$ there exists an interval $I_v \subseteq \mathbb{R}$ such that $u, v \in V(G)$ are adjacent in G if and only if $I_u \cap I_v \neq \emptyset$.

F14. Exercise. Every interval graph is chordal.

F15. Exercise. Find a chordal graph that is not an interval graph.

F16. Exercise. Prove that a graph G is an interval graph if and only if the maximal cliques of G can be linearly ordered $C_1, C_2, \ldots, C_k$ so that if $v \in C_i \cap C_{i''}$ and $1 \leq i \leq i' \leq i'' \leq k$, then $v \in C_{i'}$.

F17. Exercise. Let $k \geq 0$ be an integer, and let G be a graph. Prove that G has path–width at most k if and only if G is a subgraph of an interval graph that has no subgraph isomorphic to K_{k+2}.

F18. Definition. [3] Let $k \geq 0$ be an integer. A *stoppage* of order k is a set $\mathcal{S}$ of separations of order $< k$ such that

(i) if (A, B) is a separation of G of order $< k$, then $\mathcal{S}$ contains one of $(A, B), (B, A)$, and
(ii) if $(A_1, B_1), (A_2, B_2) \in \mathcal{S}$ then $A_1 \cup A_2 \neq G$.

If $(A, B) \in \mathcal{S}$ it helps to think of A as the "small" side and B as the "big" side of the separation.

F19. Theorem. (Min-max theorem for path-width [3]) *Let $k \geq 0$ be an integer. A graph has a stoppage of order k if and only if it has path–width at least $k - 1$.*

F20. Exercise. Let $k \geq 0$ be an integer, and let G be a graph. The following conditions are equivalent.

(i) G has a path–width $\geq k - 1$,
(ii) k cops can capture an invisible robber in G, and
(iii) G has a stoppage of order k.

F21. Definition. Let M be an $n \times m$ 0–1 matrix. We define the *track–number* of M to be the minimum integer $k \geq 0$ such that there exists a matrix M' obtained from M by permuting columns with the following property. In every row of M' we find the left–most one and the right–most one, and change all zeros in between to ones. We do this for every row, and the property we are describing is that after this operation every column contains at most k ones. The GATE MATRIX LAYOUT problem asks, given k and M, if the track–number of M is at most k. This is an important problem in VLSI design. The columns of M correspond to *gates*. If there is a row with ones in columns corresponding to gates g_1 and g_2 then g_1 and g_2 must be *connected*. The gates will be placed consecutively next to each other on a long thin rectangular board, and the connections will be realized by means of parallel *tracks*. Obviously, we want to minimize the number of tracks.

F22. Exercise. Prove that for every integer $k \geq 0$ and for every 0–1 matrix M, there is a graph G such that M has track–number at most k if and only if G has path–width at most $k-1$. Moreover, the graph G can be constructed in linear time.

REFERENCES

1. C. Anhalt and R. Thomas, Two-connected graphs of large path-width, manuscript.
2. S. Arnborg, D. G. Corneil and A. Proskurowski, Complexity of finding embeddings in a k-tree, *SIAM J. Alg. Disc. Meth.* **8** (1987), 277-284.
3. D. Bienstock, N. Robertson, P. D. Seymour and R. Thomas, Quickly excluding a forest, *J. Combin. Theory Ser. B* **52** (1991), 274–283.
4. H. L. Bodlaender, manuscript
5. R. Diestel, Graph Minors $\mathrm{I}_{1+\epsilon}$, *Combinatorics, Probability and Computing* **4** (1995), 27–30.
6. I. Kříž and R. Thomas, Clique-sums, tree-decompositions and compactness, *Discrete Math.* **81** (1990), 177–185.
7. N. Robertson and P. D. Seymour, Graph Minors I. Excluding a forest, *J. Combin. Theory Ser. B* **35** (1983), 39–61.
8. N. Robertson and P. D. Seymour, Graph Minors III. Planar tree-width, *J. Combin. Theory Ser. B* **36** (1984), 49–63.
9. N. Robertson and P. D. Seymour, Graph Minors V. Excluding a planar graph, *J. Combin. Theory Ser. B* **41** (1986), 92–114.
10. N. Robertson and P. D. Seymour, Graph Minors X. Obstructions to tree-decomposition, *J. Combin. Theory Ser. B* **52** (1991), 153–190.
11. N. Robertson, P. D. Seymour and R. Thomas, Quickly excluding a planar graph, *J. Combin. Theory Ser. B* **62** (1994), 323–348.
12. N. Robertson, P. D. Seymour and R. Thomas, unpublished.
13. P. D. Seymour and R. Thomas, Graph searching, and a min-max theorem for tree-width, *J. Combin. Theory Ser. B* **58** (1993), 22–33.