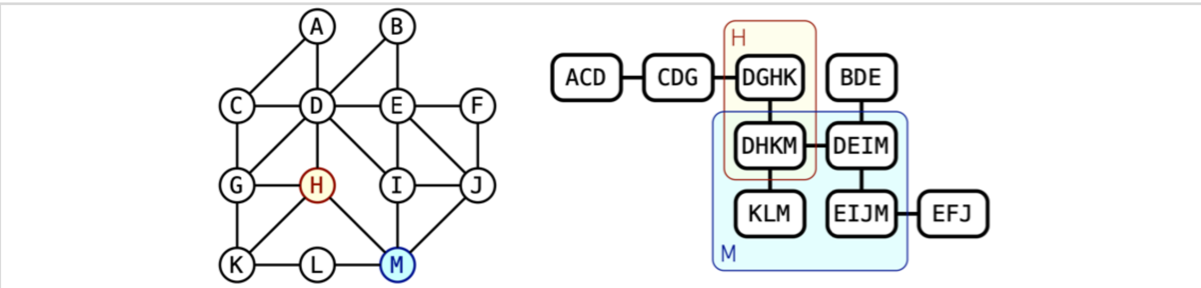
CS 7301.003.20F Lecture 19–October 21, 2020

Main topics are #graph_minors.

Tree Decompositions

- Last time we discussed minor-closed families. One theme that came up is the close relationship between a graph parameter called "treewidth" and various minor-closed families.
- Today, we're going to look at treewidth again from a different angle to see how it has some direct applications in algorithm design.
- So first, we need an alternative way to define it.
- A *tree decomposition* (T, X) of a graph G = (V, E) consists of a tree T = (I, F) and a function $X : I \to 2^V$ satisfying three constraints. I'll use vertex for members of V and node for members of I. A vertex v is *associated with* a node i or vice verse whenever v in X(i).
  1. Every vertex of G is associated with at least one node in T. (Or Cup_{i in I} X(i) = V).
  2. For every edge uv in G, at least one node in T is associated with both u and v.
  3. The nodes in T associated with any vertex of G define a connected subgraph of T.
- The *width* of a tree decomposition if max_i |X(i)| - 1.
- Here's a tree decomposition of width 3.



- On Monday, we defined the treewidth of G as the minimum k such that G is a partial k-tree. We can also define treewidth as the minimum width of any tree decomposition of G. The two definitions are equivalent.
- The big reason treewidth is useful in algorithm design is because small treewidth implies small separators.
- Say an n-vertex graph G is s-separable if either n = 1 or G has a 2 / 3 balanced separator S of size s(n) such that the components of G \ S are also s-separable.
- Lemma: Any graph of treewidth k is O(k)-separable. Conversely, any s-separable n-vertex graph has treewidth O(s(n) log n) or treewidth O(s(n) if s(n) = Omega(n^c) for some constant c > 0.
  - Suppose G has a tree decomposition (T, X) of width k.
    - WLOG, every node in T has degree 3.
    - So there is a node i that separates T into subtrees with at most 2 / 3 of the nodes

of T each.

        - The corresponding subset X(i) is a 2 / 3 separator for G of size k + 1, and the components of T \ X(i) has treewidth at most k.
    - On the other hand, suppose G is s-separable.
        - Let S be a 2 / 3 separator of size s(n).
        - We can recursively find path decompositions (T_1, X_1) and (T_2, X_2) for both components.
        - Let T be the concatenation of both paths, and for any node i in T_j, let X(i) = X_j(i) cup S.
        - (T, X) is a path decomposition with width k(n) ≤ k(2n / 3) + s(n).
- Corollary: Every planar graph has treewidth O(sqrt{n}). Every graph embeddable on a surface of genus g has treewidth O(sqrt{gn}).

Maximum Independent Set

- And using these small separators, one can solve some normally hard problems very quickly.
- For example, maximum independent set where you want a maximum size subset of vertices that are pairwise non-adjacent.
- Suppose G = (V, E) has a tree decomposition (T, X) of width k.
- We may assume T has at most n nodes. Let r be an arbitrarily chosen root of T.
- For each node i of T let D(i) denote the union of subsets X(j) for all descendants j of i (including j = i) and G_i be the induced subgraph over D(i).
- For each node i of T and any A subseteq X(i), let MIS(i, X(i)) denote the size of the largest independent set I of G_i such that I cap X(i) = A.
- We have MIS(i, A) =
    - |A| + sum_{child j of i} [
        - max_{J subseteq X(j) s.t. J cap X(i) = A cap X(j) and J is independent} MIS(j, J)
        - - |A cap X(j)|
- The best independent set is the maximum value of MIS(r, A) for all choices of A subseteq X(r).
- For each node i, there are 2^{k +1} subsets A and thus subproblems MIS(i, A) to consider.
- For each subproblem MIS(i, A) and each child j of i, there are 2^{k + 1} subsets J to consider, and we can determine in O(k^2) if that subset if an independent set.
- So the total time to evaluate MIS(i, A) given solutions to the recursive subproblems is O(2^k k^2 * deg(i)).
- If we solve all the subproblems in postorder (from leaves up). that leads to a total time of O(4^k k^2 n).
- That’s a linear time algorithm if k is a constant, granted the dependency on k is somewhat

large.
- And this general strategy works with A LOT of different problems that are NP-hard in general graphs.

PTAS for Planar Graphs

- Now, graphs of constant treewidth are perhaps not the most natural family of graphs to consider, but the previous result can be used as a subroutine for approximation algorithms in planar and surface graphs.
- First off, we need a way to bound the treewidth of certain planar graphs.
- Lemma [Eppstein '99]: A planar graph with diameter D has treewidth at most $3D + 1$.
  - We'll assume G is a triangulation as we can always add edges without increasing diameter.
  - Let (T, C) be a tree-cotree decomposition where T is a BFS tree rooted at some vertex r.
  - For each face f, let $X(f)$ be the set of at most $3D + 2$ vertices on shortest paths from f to r.
  - I claim (C^*, X) is a tree decomposition of width at most $3D + 1$. In short, for any vertex v, you can root C^* at a face incident to v. Any subtree of a face not associated with v will be bounded by edges on the opposite side of T as v, meaning none of the faces in that subtree are associated with v.
- Now, let G be a planar graph. We're going to find a *polynomial time approximation scheme* for maximum independent set. The polynomial time approximation scheme will find an independent set of size $(1 - \varepsilon)$ times optimal for any constant $\varepsilon > 0$ in polynomial time. This algorithm is due to Baker ['94].
- Choose a root vertex r of G, and let $d(v)$ denote the depth of v in a BFS tree from r.
- Let $G[i, j]$ be the induced subgraph of vertices v where $i \leq d(v) \leq j$, and let $G<i, j] := G[0, j] / G[0, i - 1]$. In other words, we contract the first $i - 1$ levels but keep the rest of the vertices through level j.
- Observe $G<i, j]$ has diameter at most $2j - 2i + 1$: There's a path between any pair that goes to the contracted root vertex and back.
- Now, consider what happens if we delete some strips of vertices from G. Specifically, for any $i \geq 0$ and $k \geq 2$ such that $i < k$, let $G_{i, k}$ denote the subgraph of G you get by deleting vertices of depth $i \bmod k$.
- Lemma: For any planar graph G and integers $i \geq 0$ and $k \geq 2$ such that $i < k$, the subgraph $G_{i, k}$ has treewidth $O(k)$.
  - $G_{i, k}$ is the disjoint union of several graphs $G[ak + i + 1, (a + 1)k + i - 1]$. Each graph $G<ak + i + 1, (a + 1)k + i - 1]$ has diameter at most $2k - 3$ and therefore treewidth at most $3(2k - 3) + 1 = 6k - 8$. $G_{i, k}$ has the same treewidth.

- So here's our algorithm:
  - Set $k := \lceil 1 / \epsilon \rceil$.
  - For each $0 \leq i < k$, compute the largest independent set in $G_{i, k}$ in $2^{O(k)} n$ time.
  - Return the largest set found.
- We compute only k independent sets, so the running time is $2^{O(k)} n = 2^{O(1 / \epsilon)} n$.
- But also note that there is a choice of i such that only a $1 / k \leq \epsilon$ fraction of the max independent set vertices have depth i mod k. The independent set we computed for that choice of i had size at least $(1 - \epsilon) *$ optimal.
- The same algorithm works on graphs of low genus. You can show a diameter D graph of genus g has treewidth $O(gD)$, implying a PTAS for max independent set that runs in time $2^{O(g / \epsilon)} n$.