TD 10: Chordal Graphs

1 Clique Cover

A **clique cover** of a graph $G = (V, E)$ is a partition of V into disjoint sets $V_1, V_2, \dots, V_k$ such that for all $i \in \{1, \dots, k\}$, $G[V_i]$ is a clique. The clique-cover number of G is the minimum number k such that G has a clique cover with k sets.

1. Observe that the clique-cover number of G is equal to $\chi(\overline{G})$.
2. Give a polynomial-time algorithm for computing the clique-cover number of a chordal graph.

Solution:

The first part is easy to see because a coloring of $\overline{G}$ is a partition of V into independent sets. These sets are cliques in G (and vice-versa, cliques in G are independent sets in $\overline{G}$).

For the second part, note that we **cannot** use the previous observation to simply claim that we compute the chromatic number of $\overline{G}$, because $\overline{G}$ is not necessarily chordal, so it is not clear if COLORING can be solved in polynomial time in this graph. (We will see later that this argument can be made to work, but we need further tools.)

We therefore formulate a direct recursive algorithm. For the base case, if $G = (V, E)$ is a clique, we output a clique cover consisting of a single set (V). Otherwise, let v be a simplicial vertex, which must exist since G is chordal. We recursively compute a clique cover of $G - N[v]$ and add to this cover the set $N[v]$. The algorithm runs in polynomial time, as finding a simplicial vertex can be done in polynomial time and each time we recurse we have a smaller instance.

Let us argue for correctness. First, the collection of sets we output is indeed a collection of cliques that covers the whole graph because (i) this is true in the base case (ii) $N[v]$ is a clique, as v is simplicial. Suppose our algorithm outputs a cover with k sets, but for the sake of contradiction G can be covered with $\leq k - 1$ sets. This clearly cannot happen in the base case, so suppose that it happens for a non-clique G and among all such counter-examples, pick G with the smallest number of vertices. Suppose that the optimal solution places v in V_1 with $V_1 \neq N[v]$. Since V_1 is a clique, $V_1 \subseteq N[v]$, so the $(k - 2)$ sets $V_2, \dots, V_{k-1}$ of the supposed optimal solution partition $V \setminus V_1 \supseteq V \setminus N[v]$. However, our algorithm used $k - 1$ sets for $G - N[v]$ and this is optimal, as G is a minimum counter-example, contradiction.

2 A Tree Communication Problem

Consider the following problem: we are given a tree $T = (V, E)$ and a collection of pairs of vertices $(x, y) \in V^2$. Informally, T represents a communication network and the pair (x, y) encodes the fact that x wants to communicate with y. Recall that between any two $x, y \in V(T)$ there is a unique path. We want to assign colors to the pairs so that any two pairs that **interfere** with each other receive distinct colors, while using as few colors as possible.

1. Suppose that we say that (x_1, y_1) and (x_2, y_2) interfere when the unique path from $x_1 \to y_1$ and the unique path $x_2 \to y_2$ share a vertex. Show that in this case the minimum number of colors can be computed in polynomial time by constructing a chordal graph with one vertex for each communicating pair, an edge for each interference, and coloring that graph.

2. Show that if we instead define that two paths interfere when they share an **edge** (but sharing a vertex does not count as interference), then the intersection graph we obtain is **not** chordal.

Solution:

For the first part, we want to define an intersection graph as hinted and show that it is chordal. To do so, we will show that the intersection graph must necessarily contain a simplicial vertex. Deleting this vertex (that is, the corresponding communication pair) and repeating this argument will produce a PEO proving that the graph is chordal.

Pick an arbitrary vertex r of T and call it the root. For the communication pair (x, y) let P_{xy} be the unique path connecting x, y in T and we define the rank of P_{xy} as the minimum distance of any vertex of P_{xy} from r. Consider then a path P_{xy} of maximum rank. We claim that the vertex representing (x, y) in the intersection graph is simplicial.

Let z be the vertex of P_{xy} that is closest to r, so we can think of P_{xy} as the union of a path $x \to z$ with a path $z \to y$. We claim that for all other pairs (a, b) such that P_{ab} intersects P_{xy}, it must be the case that $z \in P_{ab}$. If this is true, then the neighbors of (x, y) in the intersection graph form a clique, so (x, y) is simplicial as desired. However, the claim that P_{ab} contains z follows from the selection of (x, y): if for the sake of contradiction P_{ab} contains a vertex of $P_{xy} \setminus \{z\}$ (because the two paths intersect), P_{ab} must also contain a vertex c that is as close to r as z (because the distance to z is maximal in our selection of x, y), so the unique path from c to the common vertex of P_{ab}, P_{xy} must contain z.

For the second part, consider a tree $T = K_{1,4}$ and call the leaves a, b, c, d. If we have the communication requests $(a, b), (b, c), (c, d), (a, d)$, the intersection graph formed when we only care about edge-disjointness is a C_4, which is not chordal. We note that because of this example, we can actually see that a polynomial-time algorithm is not possible for this problem, unless P=NP. Indeed, a tree $T = K_{1,n}$ is sufficient to encode (using appropriate communication pairs) the edges of an arbitrary graph G, so coloring the corresponding intersection graph is equivalent to edge-coloring G, which is NP-complete.

3 Coloring on Chordal Graphs again

We saw in class a polynomial-time algorithm for computing the chromatic number of a chordal graph. We consider here an alternative version which does not rely on perfect elimination orderings but rather uses the fact that minimal separators are cliques.

Consider the following recursive algorithm which takes as input a graph G and an integer k and decides if $\chi(G) \leq k$. If G has at most k vertices, the algorithm replies Yes. Otherwise, if G is a clique, the algorithm replies No. Otherwise, the algorithm finds two vertices x, y which are non-adjacent and computes a minimal xy-separator S. Let $C_1, C_2, \ldots, C_t$ be the connected components of $G - S$. We recursively execute the same algorithm on each $G[S \cup C_i]$ for $i \in \{1, \ldots, t\}$. If the answer is No for any such sub-instance we reply No, otherwise we reply Yes.

Prove that the algorithm sketched above is correct and runs in polynomial time.

Solution:

Let us first argue for correctness. The two base cases are clearly correct: if G has at most k vertices, it can be colored with at most k colors (assign a distinct color to each vertex); if this is not the case and G is a clique, then we need $|V| > k$ colors.

For the general case, observe that two non-adjacent x, y always exist, as G is not a clique. Furthermore, if G is chordal, then any minimal xy-separator S must be a clique. If $G[S \cup C_i]$ is not k-colorable, for some i, then G is also not k-colorable, as k-colorability is preserved by taking subgraphs. The interesting part is then to show that if all $G[S \cup C_i]$ are k-colorable, then G is k-colorable. In order to show this, suppose that for some $j \geq 1$ we have a k-coloring of $G[S \cup (C_1 \cup C_2 \cup \ldots \cup C_j)]$. (This is true for $j = 1$, as we have assumed that a k-coloring of $G[S \cup C_1]$ exists.) We show that from this k-coloring and a k-coloring of $G[S \cup C_{j+1}]$ we can obtain a k-coloring of $G[S \cup (C_1 \cup C_2 \cup \ldots \cup C_{j+1})]$. Repeating this will produce a coloring of the whole graph.

The key idea now is to make the coloring of $G[S \cup (C_1 \cup C_2 \cup \ldots \cup C_j)]$ and of $G[S \cup C_{j+1}]$ agree on S. Without loss of generality, both colorings assign colors $1, 2, \ldots, |S|$ to the vertices of S (because S is a clique), so by permuting colors in one coloring we can make sure that the two colorings assign the same color to each vertex of S. Taking the union of the two colorings is now a coloring of the larger graph, as C_{j+1} has not edge to $C_1 \cup \ldots \cup C_J$, because C_{j+1} is a connected component of $G - S$.

What remains is to argue that the algorithm runs in polynomial time, which is not obvious. First, let us point out that if we ignore the recursive calls, the rest of the algorithm runs in polynomial time: checking the size of G or whether G is a clique is easy; finding two non-adjacent x, y is easy; and a minimal xy-separator can be constructed by starting with $S := V \setminus \{x, y\}$ (which is clearly an xy-separator) and arbitrarily removing redundant vertices from S until S becomes minimal.

Second, the instances on which we recurse are smaller, but this is not obviously sufficient to ensure a polynomial running time. Indeed, we could have $|S| = n - 2$ and $|C_1| = |C_2| = 1$, in which case from an instance on n vertices we produce two instances on $n - 1$ vertices. Naively solving this recurrence gives $T(n) \leq 2T(n-1) \Rightarrow T(n) \leq 2^n$.

To work around this difficulty we will use a measure of progress that is slightly more interesting than the size of the graph. Define the "interesting size" of a graph G, denoted $s(G)$, to be equal to the number of edges of $\overline{G}$, that is the number of non-edges of G. We will prove that the number of recursive calls made by the algorithm is polynomially bounded by $s(G) < n^2$, and since in each recursive call we make an amount of work that is polynomial in n, the total running time will be polynomial.

Now consider the general case, where we recurse from G into $G[S \cup C_i]$, for $i \in [t]$, S a minimal clique separator. We claim $s(G[S \cup C_i]) < s(G)$ for all $i \in [t]$; and $s(G) \geq \sum_{i \in [t]} s(G[S \cup C_i])$. Before we prove these inequalities, let us explain why they imply that the number of recursive calls is polynomial. We can imagine the recursion tree of the execution of the algorithm where all leaves correspond to base case instances. By bounding the number of leaves we bound the size of the whole tree. We will do a proof by induction on $s(G)$, so suppose that the statement is true for all graphs of smaller "interesting size" than G. By the first inequality, all the instances we recurse on have strictly smaller interesting size, so the inductive hypothesis applies. In particular, suppose that for $G[S \cup C_i]$ we produce $s(G[S \cup C_i])^c$ base instances, for some constant $c \geq 1$. Then, the total number of base instances for G is at most $\sum_{i \in [t]} (s(G[S \cup C_i]))^c \leq (\sum_{i \in [t]} s(G[S \cup C_i]))^c \leq s(G)^c$, where in the first step we used standard facts about the function x^c (namely, $x^c + y^c \leq (x + y)^c$ when $c \geq 1$); and in the second step we used the second inequality.

Let us now prove the two inequalities. We have $s(G[S \cup C_i]) < s(G)$ because each edge that is missing is $G[S \cup C_i]$ is also missing in G and furthermore there exists a missing edge from C_i to C_j for $j \neq i$ in G (because S is a separator).

For the second inequality, it is key to observe that the sum $\sum_{i \in [t]} s(G[S \cup C_i])$ counts every non-edge at most once, because the only pairs of vertices which appear in two distinct graphs in the sum are pairs where both vertices are in S, and S is a clique. Therefore, since every non-edge is counted once and every non-edge of one of these graphs is also a non-edge of G we get the inequality.

4 Tree Intersection Models

Recall that in Exercise 2 we defined a class of intersection graphs and proved that they are chordal. In this exercise we define a richer class of intersection graphs on a tree and prove that this class is actually **exactly** the class of all chordal graphs.

Consider a tree T and let $T_1, T_2, \ldots, T_k$ be a collection of sub-trees, that is, a collection of connected induced subgraphs of T. We define the intersection graph G of this collection as follows: (i) we have a vertex in G for each sub-tree T_i (ii) T_i and T_j are adjacent in G if and only if T_i and T_j have a vertex in common.

(Observe that if all the trees in our collection are paths, the intersection graphs we obtain are the same as those of Exercise 2.)

1. Prove that the intersection graphs that can be formed in this way are chordal.

2. Prove that all chordal graphs can be formed as intersection graphs in this way. (Hint: you will need to use Exercise 5 of TD2 on this regarding the Helly property of trees.)

Solution:

For the first question, the proof is essentially identical to that of Exercise 2. In particular, we select a root r of T, and for each T_i find the vertex that is closest to the root, call this the sub-root of T_i. Select a T_i whose sub-root r_i is as far from r as possible. We claim that r_i must be contained in all T_j that intersect T_i and the argument is the same as in Exercise 2.

For the second question, we recall that if we have a collection $T_1, \ldots, T_k$ of sub-trees of a tree T such that any two T_i, T_j share a common vertex, there actually exists a vertex that is common to all k sub-trees. We use this fact to build a tree model of an arbitrary chordal graph G by induction. If G has at most 2 vertices, this is easy. Suppose then that G contains a simplicial vertex v and we can form by induction a tree intersection model of $G - v$. Let $N(v) = \{v_1, \ldots, v_k\}$ and $T_1, \ldots, T_k$ be the sub-trees representing the vertices of $N(v)$ in this model. Because $N(v)$ is a clique, any pair T_i, T_j has a non-empty intersection. Therefore, by the Helly property, there exists a node x of the tree such that $x \in T_i$ for all $i \in [k]$. We add to the tree a new leaf x' connected to x and represent v by the sub-tree $\{x, x'\}$.