Managing Complexity

Independent Sets in Small Tree-Width Graphs

In solving the max-weight independent set problem on trees, we leveraged the fact that trees have small separating sets (single vertices) that could be used to create independent sub-problems (as long as we produced multiple solutions for each sub-problem).

Let’s push that further. The reasons for including the following definitions are to demonstrate that the intuitive idea of “drawing a graph” has a rigorous and precise foundation and to introduce some vocabulary that will be useful for the ensuing presentation.

Definitions

Drawing A drawing (or embedding) of a graph \( G = (V, E) \) is a pair of functions \( f, g \) such that

- \( f : V \rightarrow \mathbb{R}^2 \) and \( f \) is 1-1
- For each \( e = \{u, v\} \in E \), \( g(e) \) is a non-self-intersecting curve from \( f(u) \) to \( f(v) \): That is, \( g(e) \) is a continuous, 1-1 function \( g_e : [0, 1] \rightarrow \mathbb{R}^2 \) such that \( g_e(0) = f(u) \) and \( g_e(1) = f(v) \)
  - A curve \( g_e \) intersects a vertex \( w \) only at an endpoint of the curve (implying \( w \) is an end of \( e \)).

Planar A graph is planar if it has a drawing \( \{f, g\} \) such that no two curves intersect except possibly at their endpoints.

Plane Such a drawing is called a planar drawing (embedding); or, for short, a plane graph.

Faces A plane graph \( G \) divides the plane into connected regions bounded by closed walks in the graph called faces, exactly one of which is infinite (the outer face) of \( G \).

Nice Faces If \( G \) is 2-connected (no cut vertex), then each faces is bounded by a cycle.

Outerplanar A graph \( G = (V, E) \) is outerplanar if it has a planar drawing such that every vertex is on the outer face of \( G \).

Maximal Outerplanar An outer planar graph is maximal outer planar if adding any edge would produce a graph that is no longer outer planar.

Triangular Faces Every interior face of a maximal outerplanar graph is a triangle.

Theorem 1.

- Given an \( n \)-vertex graph \( G \), one can decide planarity in \( O(n) \) time.
- In fact, a planar drawing can be produced in \( O(n) \) time.
- In fact, we can produce a data structure that represents the vertices, edges, and faces of \( G \).
- So, given an outer planar graph \( G \), we can assume we have a planar drawing of \( G \) and an efficient data structure for traversing the drawing.

Our first goal for today is to extend the maximum independent set algorithm from the class of trees to the class of maximal outerplanar graphs.
Tree Decomposition of Maximal Outerplanar Graphs

There is a tree-like structure to outerplanar graphs that we would like to make precise.

**Definition 1. Weak Dual Graph** Given a plane graph \( G = (V, E, F) \), the weak dual graph \( G^D = (V^D, E^D) \) of \( G \) is given by \( V^D = F - \text{outer face} \) (the faces of \( G \)) and \( E^D = \{(f, g) : f, g \text{ share a boundary edge}\} \).

**Dual is Planar** Theorem: \( G^D \) is also planar

**Dual is a Tree** Theorem: If \( G \) is outerplanar then \( G^D \) is a tree

Our assumptions
- We have a planar drawing of a maximal outerplanar graph \( G \). We denote the drawing of \( G \) also by \( G \).
- We have constructed the dual tree \( T \) of \( G \).
- We can efficiently move between \( G \) and \( T \).
- Since \( G \) is maximal outerplanar, \( G \) is 2-connected: that is, \( G \) has no single cut vertex.

\( T \) defines a natural decomposition of the vertices of \( G \) into (overlapping) sets
- Each vertex \( v_f \in V^D \) corresponds to a face \( f \) of \( G \). Let \( V_f = \{v \in V : v \text{ is on } f\} \).
- Note that the sets \( V_f \subseteq V \) have the following features
  - **Vertex Coverage** Every vertex of \( G \) is in some \( V_f \).
  - **Edge Coverage** Every edge of \( G \) is in some \( V_f \).
  - **Small Size (and Intersection)** Each \( V_f \) is a small set, and so each intersection \( V_f \cap V_g \) is small (in this case, empty, consisting of a single vertex, or consisting of two vertices forming the single edge that lies on both \( f \) and \( g \)).
  - **Coherence** If \( f_2 \) is on the path between \( f_1 \) and \( f_3 \) in \( T \) and \( v \in V_{f_1} \cap V_{f_3} \) then \( v \in V_{f_2} \).

The sets \( \{V_f : f \in V^D\} \) are an example of a tree decomposition of \( G \).

Note The dual of \( G \) was only used to help us identify subsets of vertices (the \( V_f \)) having the features we desire.

Maximum Independent Sets in Maximal Outerplanar Graphs

What does an independent set \( I \) of \( G \) look like?
- Each set \( I_f = I \cap V_f \) is an independent set of \( V_f \).
- The sets \( I_f \cap (V_f \cap V_g) = I_g \cap (V_f \cap V_g) \) for every pair \( f, g \in V^D \).
- That is, \( I_f \) and \( I_g \) are identical on \( V_f \cap V_g \).
- \( I_f \cap (V_f \cap V_g) \) is of size at most 1.

This suggests that we might be able to devise a dynamic programming algorithm to find such an \( I \)
- Root \( T \) at some vertex \( v_f \), and attempt to build \( I \) from the leaves of \( T \) to the root.
- For a leaf \( v_f \), identify the parent \( v_g \) and the vertices \( \{u, v\} \) in \( (V_f \cap V_g) \).
- Determine independent sets \( I^f_\emptyset, I^f_u, I^f_v \) such that
  - \( I^f_\emptyset \) is an independent set of maximum size in \( V_f \) that contains neither of \( u, v \).
– $I_u^f$ is an independent set of maximum size in $V_f$ that contains $u$ but not $v$
– $I_v^f$ is an independent set of maximum size in $V_f$ that contains $v$ but not $u$

Suppose now that for some vertex $v_g$ of $T$ we have been able to compute, for each subtree $T_i$ of $v_g$, the sets $I_{\emptyset}^i, I_u^i, I_v^i$. Note that $v_g$ has at most two children, unless $v_g$ is the root of $T$, in which case it can have 3 children.

Then we can compute $I_{\emptyset}^g, I_u^g, I_v^g$, where $u, v$ connect $v_g$ to its parent in $T$ as follows:

• For every independent set $U$ of $V_g$ (don’t worry, there are only 3 such $U$!)
  – determine the largest independent set $I_{\emptyset}^g$ of $T_g$ (the subtree rooted at $g$) that contains neither $u$ nor $v$ by using the $I_{\emptyset}^i, I_u^i, I_v^i$ for each subtree $T_i$
  – Similarly compute the largest independent sets $I_u^g$ and $I_v^g$

The entire algorithm takes time proportional to the size of $T$: $O(n)$ time assuming that we start with the tree $T$ of faces of $G$ and the sets $V_f$.

Tree-Width

What made this work so well?

• The subproblems associated with subtrees of $T$ depended only on the subtree, and a small number of possibilities for merging solutions to children’s subtrees to obtain solutions to parent subtrees
• The subgraph of $G$ associated with a node of $T$ was also small, so considering all possible solutions for that subgraph was not too time-consuming.

Tree Decompositions can be defined for arbitrary graphs and they introduce a new parameter, called the tree-width of the graph, which is, roughly speaking, the size of the largest subgraph of $G$ associated with a vertex of the decomposition tree $T$ (in fact, it’s one less than this quantity).

A number of normally NP-hard problems can be solved feasibly on graphs that have bounded tree-width. These include

• Independent Set (also with weights)
• Graph Coloring
• Vertex Cover
• Graph Isomorphism

Impediments to using this technique can include

• Determining minimum tree-width of a graph is NP-hard
• Finding a $k$-tree decomposition takes some effort but can be done in time linear in the size of the graph but exponential in $k$ [Bodlaender, 1996].

---

1This information can all be computed in $O(n)$ time given an appropriate description of an outerplanar drawing of $G$. 

Fall 2018 3 Bill Lenhart