Tree-width and algorithms

Zdeněk Dvořák

September 14, 2015

1 Algorithmic applications of tree-width

Many problems that are hard in general become easy on trees. For example, consider the problem of finding the size of the largest independent in a graph G. This problem is NP-complete for general graphs, but it can be solved in linear time for trees.

Consider a rooted tree T, and let r denote the root of T. For a vertex n of T, let T_n denote the subtree of T rooted in n. For every n we compute two numbers:

- $c(n)$ is the size of the largest independent set in T_n that contains n
- $d(n)$ is the size of the largest independent set in T_n that does not contain n

We proceed recursively, so that when processing n, we already computed these numbers for all sons of n. If n is a leaf, then $c(n) = 1$ and $d(n) = 0$. Otherwise,

$$c(n) = 1 + \sum_{n' \text{ son of } n} d(n')$$
$$d(n) = \sum_{n' \text{ son of } n} \max(c(n'), d(n'))$$

The size of the largest independent set in T is $\min(c(r), d(r))$.

Similar algorithms usually work even for graphs with bounded tree-width. It is useful to first simplify the decomposition. A tree decomposition (T, β) is *canonical* if

- T is rooted, and the root r satisfies $\beta(r) = \emptyset$.
- Each leaf n satisfies $\beta(n) = \emptyset$.

- Each non-leaf vertex n satisfies one of the following conditions:
  - n has exactly one son n', and $\beta(n) = \beta(n') \cup \{v\}$ for some vertex v.
  - n has exactly one son n', and $\beta(n) = \beta(n') \setminus \{v\}$ for some vertex v.
  - n has exactly two sons n_1 and n_2, and $\beta(n) = \beta(n_1) = \beta(n_2)$.

Lemma 1. *Every graph G of tree-width at most k has a canonical tree decomposition of width at most k, of polynomial size.*

Proof. Insert new vertices and split the original vertices of a tree decomposition of G as necessary. □

Suppose that (T, β) is a canonical tree decomposition of a graph G. For $n \in V(T)$, let $G_n = G\left[\bigcup_{n' \in V(T_n)} \beta(n')\right]$. For any $C \subseteq \beta(n)$, we compute

- $s(n, C)$ = size of the larges independent set $S \subseteq V(G_n)$ such that $S \cap \beta(n) = C$.

When the tree decomposition has width at most k, we compute at most 2^{k+1} numbers for each vertex of T. We proceed recursively from leaves, so that when a vertex is processed, we already computed these numbers for its sons.

- If n is a leaf, then $s(n, \emptyset) = 0$.
- If n has exactly one son n' and $\beta(n) = \beta(n') \cup \{v\}$, then
  - $s(n, C) = 1 + s(n', C \setminus \{v\})$ when $v \in C$ and no neighbor of v belongs to C,
  - $s(n, C) = -\infty$ when $v \in C$ and a neighbor of v belongs to C,
  - $s(n, C) = s(n', C)$ when $v \notin C$.
- If n has exactly one son n' and $\beta(n) = \beta(n') \setminus \{v\}$, then $s(n, C) = \max(s(n', C), s(n', C \cup \{v\}))$.
- If n has exactly two sons n_1 and n_2, and $\beta(n) = \beta(n_1) = \beta(n_2)$, then $s(n, C) = s(n_1, C) + s(n_2, C) - |C|$.

The size of the largest independent set in G is $s(r, \emptyset)$. The time complexity is $O(k2^k|V(T)|)$.

As a slightly more involved example, consider the computation of the size of the smallest dominating set, i.e., the smallest set $S \subseteq V(G)$ such that every vertex of G either belongs to S or has a neighbor in S. Again, let (T, β) be is a canonical tree decomposition of a graph G, and for $n \in V(T)$, let $G_n = G\left[\bigcup_{n' \in V(T_n)} \beta(n')\right]$.

For any disjoint sets $B, C \subseteq \beta(n)$, we compute

- $s(n, B, C) =$ size of the smallest set $S \subseteq V(G_n)$ such that $S \cap \beta(n) = C$, no vertex of B has a neighbor in S, and every vertex of $V(G_n) \setminus B$ either belongs to S, or has a neighbor in S.

When the tree decomposition has width at most k, we compute at most 3^{k+1} numbers for each vertex of T. We proceed recursively from leaves, so that when a vertex is processed, we already computed these numbers for its sons.

- If n is a leaf, then $s(n, \emptyset, \emptyset) = 0$.
- If n has exactly one son n' and $\beta(n) = \beta(n') \cup \{v\}$, then
  - $s(n, B, C) = s(n', B \setminus \{v\}, C)$ when $v \in B$ and no neighbor of v belongs to C,
  - $s(n, B, C) = \infty$ when $v \in B$ and a neighbor of v belongs to C,
  - $s(n, B, C) = 1 + \min\{s(n', B', C \setminus \{v\}) : B \subseteq B' \subseteq B \cup N\}$ when $v \in C$, N is the set of neighbors of v in $\beta(n') \setminus C$, and $N \cap B = \emptyset$,
  - $s(n, B, C) = \infty$ when $v \in C$ and a neighbor of v belongs to B,
  - $s(n, B, C) = s(n', B, C)$ when $v \notin B \cup C$ and a neighbor of v belongs to C, and
  - $s(n, B, C) = \infty$ when $v \notin B \cup C$ and no neighbor of v belongs to C.
- If n has exactly one son n' and $\beta(n) = \beta(n') \setminus \{v\}$, then $s(n, B, C) = \min(s(n', B, C), s(n', B, C \cup \{v\}))$.
- If n has exactly two sons n_1 and n_2, and $\beta(n) = \beta(n_1) = \beta(n_2)$, then $s(n, B, C) = \min\{s(n_1, B_1, C) + s(n_2, B_2, C) - |C| : B_1, B_2 \subseteq \beta(n) \setminus C, B_1 \cap B_2 = B\}$.

The size of the smallest dominating set in G is $s(r, \emptyset, \emptyset)$. The time complexity is $O(k5^k|V(T)|)$.

2 Finding a tree decomposition

We use a variant of a lemma from the last lecture.

Lemma 2. *Let G be a graph of tree-width at most k and let $f : V(G) \to \mathbf{R}^+$ be an arbitrary function. For a set $X \subseteq V(G)$, let $f(X) = \sum_{x \in X} f(x)$. Then G contains a set $S \subseteq V(G)$ of size at most $k+1$ such that every component C of $G - S$ satisfies $f(V(C)) \le f(V(G))/2$.*

We say that a graph G is *s-fragile* if for every $G' \subseteq G$ and $W \subseteq V(G')$, G' contains a set $S \subseteq V(G')$ of size at most s such that every component C of $G' - S$ contains at most $|W|/2$ vertices of W.

Lemma 3. *Every graph of tree-width at most k is $(k+1)$-fragile.*

Proof. As every subgraph of G has tree-width at most k, it suffices to prove the condition of $(k+1)$-fragility for $G = G'$. Let $f(v) = 1$ for $v \in W$ and $f(v) = 0$ otherwise, and apply Lemma 2. □

Lemma 3 has an approximate converse.

Lemma 4. *Every s-fragile graph has tree-width at most $2s$.*

Proof. We prove a stronger claim.

> Let G be an s-fragile graph, and let $W \subseteq V(G)$ have size at most $2s+1$. Then G has a tree-decomposition (T, β) of width at most $2s$ such that $W \subseteq \beta(n)$ for some $n \in V(T)$. (1)

We prove (1) by induction on the number of vertices of G (i.e., we assume that (1) holds for all graphs with less than $|V(G)|$ vertices). If $|V(G)| \le 2s$, then we can let T be the tree with one vertex n and $\beta(n) = V(G)$. Hence, assume that $|V(G)| \ge 2s+1$. By adding vertices to W if necessary, we can assume that $|W| = 2s+1$. Let $S \subseteq V(G)$ be a set of size at most s such that every component of $G - S$ contains at most $(2s+1)/2$ vertices of W. Let $C_1, \ldots, C_m$ be the components of $G - S$, and for $1 \le i \le m$, let $G_i = G[V(C_i) \cup S]$. Note that G_i contains at most $(2s+1)/2 + s < 2s+1$ vertices of W, hence $W \not\subseteq V(G_i)$, and thus $|V(G_i)| < |V(G)|$. By the induction hypothesis, G_i has a tree decomposition (T_i, β_i) of width at most $2s$ such that $W \cap V(G_i) \subseteq \beta(n_i)$ for some $n_i \in V(T_i)$.

Let T be the tree obtained from the disjoint union of $T_1, \ldots, T_m$ by adding a new vertex n adjacent to $n_1, \ldots, n_m$. Let $\beta(n') = \beta_i(n')$ for $n' \in V(T) \setminus \{n\}$, where $i \in \{1, \ldots, m\}$ satisfies $n' \in V(T_i)$; and let $\beta(n) = W$. Then (T, β) is a tree decomposition of G of width at most $2s$ and $W \subseteq \beta(n)$. □

As a corollary, we have the following.

Theorem 5. *For every $s \geq 0$, there exists a polynomial-time algorithm that for a graph G either decides that G has tree-width at least s, or returns a tree decomposition of G of width at most $2s$.*

Proof. The proof of Lemma 3 gives an algorithm that either finds a tree decomposition of G of width $2s$, or finds a subgraph $G' \subseteq G$ and a set $W \subseteq V(G')$ showing that G is not s-fragile. In the latter case, Lemma 3 shows that G does not have tree-width at most $s - 1$.

To execute the algorithm, we need for a given set W and graph G' to decide whether there exists a set S of size at most s such that each component of $G' - S$ contains at most $|W|/2$ vertices of W. To do so, we can simply test all such sets $S \subseteq V(G')$. This results in an algorithm with time complexity $O(|V(G)|^{s+2})$. □

With a little work, this can be improved to $O(f(s)|V(G)|^2)$ for some function f. The current best approximation algorithms is by Bodlaender, Drange, Dregi, Fomin, Lokshtanov and Pilipczuk: decides that either the tree-width is at most $5k - 1$, or at least k, in time $O(c^k|V(G)|)$.

Let us remark that such an approximation is sufficient for the described algorithms. E.g., we can find the size of the largest independent set of a graph with tree-width at most k in time $O(c^k|V(G)|)$, even if the tree decomposition is not given in advance—we find a decomposition of width at most $5k - 1$ using the algorithm of Bodlaender et al., and then apply the algorithm for independent sets to this approximate decomposition, which has time complexity $O(k2^{5k}|V(G)|)$.

Furthermore, for fixed k, given a tree decomposition of G of width at most $5k - 1$, it is possible to find a tree decomposition of G of width at most k (when it exists) in linear time, using an algorithm of Bodlaender and Kloks. Hence, we have the following.

Theorem 6. *For every $k \geq 1$, there exists a linear-time algorithm which for given graph G either decides that $\mathrm{tw}(G) > k$, or finds a tree decomposition of G of width at most k.*

3 Exercises

1. ($\star$) Modify the algorithm that finds the size of the largest independent set in a graph G of bounded tree-width so that it also returns one such largest independent set $S \subseteq V(G)$.

2. ($\star\star$) Modify the algorithm that finds the size of the smallest dominating set in a graph G of bounded tree-width so that it returns the number of all (not necessarily smallest) dominating sets in G.

3. ($\star\star\star$) Design a polynomial-time algorithm that determines whether a graph of bounded tree-width (given with its tree decomposition) is 3-colorable.

4. ($\star$) Prove Lemma 2.