RWTH Aachen **SS 2013**
Lehrgebiet Theoretische Informatik Problem Set 3
Reidl—Rossmanith—Sánchez—Sikdar 14.05.2013

Parameterized Algorithms Tutorial

Tutorial Exercise T6

Recall that a smooth tree decomposition $\mathscr{T} = (T = (I, F), \{X_i \mid i \in I\})$ of a graph G of width k is one which satisfies the conditions:

1. $|X_i| = k + 1$ for all $i \in I$;
2. $|X_i \cap X_j| = k$ for all $\{i, j\} \in F$.

Show that if $\mathscr{T} = (T = (I, F), \{X_i \mid i \in I\})$ is a smooth tree decomposition of $G = (V, E)$ of width k then $|I| = |V| - k$.

Solution

We use induction on $|V|$. To show that there is a basis for induction, note that when $|V| = k + 1$ with a smooth tree decomposition of width k, then $|I| = 1$. Therefore assume that if G is any graph on $k + i + 1$ vertices, where $i \geq 0$, then a smooth tree decomposition $\mathscr{T}$ of G of width k has $i + 1$ nodes. Let G be a graph on $k + i + 2$ nodes and let $\mathscr{T} = (T = (I, F), \{X_i \mid i \in I\})$ be a smooth tree decomposition of G of width k. Consider a leaf node $l \in I$ of this tree decomposition. By the properties of a smooth tree decomposition, there exists $x \in X_l$ such that $x \notin X_i$ for all $i \in I \setminus \{l\}$ and that $X_l \setminus \{x\} \subseteq X_p$, where $p \in I$ is the parent of l. Deleting l from the tree decomposition yields a smooth tree decomposition for the graph $G - x$, which has $k + 1 + i$ vertices. By induction hypothesis, the decomposition obtained by deleting l has $k+i$ nodes. Therefore $\mathscr{T}$ has $k + i + 1$ nodes and this completes the proof.

Tutorial Exercise T7

Show that a graph $G = (V, E)$ of treewidth at most k has at most $k \cdot |V| - \binom{k+1}{2}$ edges.

Solution

Again we use induction on $|V|$. To show basis, note that when $|V| = k + 1$ then the number of edges in a graph of width at most k is at most

$$\binom{k+1}{2} = k(k+1) - \binom{k+1}{2}.$$

Assume that when G is any graph on $k + i + 1$ vertices with width at most k, then it has at most $k(k + i + 1) - \binom{k+1}{2}$ edges. Consider a graph G on $k + i + 2$ vertices and let $\mathscr{T} = (T = (I, F), \{X_i \mid i \in I\})$ be a smooth tree decomposition of width k. As in the last exercise, let $l \in I$ be a leaf node and let $x \in X_l$ be such that x occurs in none of the other bags of the decomposition. Since $\mathscr{T}$ is smooth, $X_l \setminus \{x\} \subseteq X_p$, where $p \in I$ is the parent of l. Deleting l from T yields a smooth tree decomposition of width k for the

graph $G-x$, which by induction hypothesis has at most $k(k+i+1)-\binom{k+1}{2}$ edges. Since the vertex x has at most k neighbors in G, the number of edges in G is at most

$$k(k+i+1)-\binom{k+1}{2}+k = k(k+i+2)-\binom{k+1}{2}.$$

Therefore, by induction, the claim is true.

Tutorial Exercise T8

Show that given a tree decomposition $\mathscr{T} = (T = (I, F), \{X_i \mid i \in I\})$ of width k, one can in $O(|I|)$ time obtain a tree decomposition of width k satisfying the condition $X_i \nsubseteq X_j$ for all $\{i, j\} \in F$.

Suppose $\mathscr{T} = (T = (I, F), \{X_i \mid i \in I\})$ is such a tree decomposition. Then show that

1. for all $\{i, j\} \in F$, $X_i \cap X_j$ is a cut in G;
2. if X_i is not a leaf, then X_i is a cut in G.

Proposed Solution

Let $\mathscr{T} = (T = (I, F), \{X_i \mid i \in I\})$ be a tree decomposition of a graph G of width k. Root the tree T at an arbitrary node. Do a depth-first search of the tree T postorder. When the traversal has finished visiting all the children of a node i with parent j, check whether $X_i \subseteq X_j$. If this is the case, delete i and add an edge from each child of i to j. At the end of this process, there is no parent-child pair such that $X_{\text{child}} \subseteq X_{\text{parent}}$. The time taken is $O(|I| \cdot k)$.

Assume that $\mathscr{T} = (T = (I, F), \{X_i \mid i \in I\})$ be a tree decomposition of a graph G of width k. Since for all $\{i, j\} \in F$ neither $X_i \subseteq X_j$ nor $X_j \subseteq X_i$, there exists $u \in X_i \setminus X_j$ and $v \in X_j \setminus X_i$. We claim that in $G - (X_i \cap X_j)$ there is no path from u to v. Let $u, x_1, \ldots, x_r, v$ denote a path from u to v in G with r internal vertices. One of these internal vertices must be in $X_i \cap X_j$. For, if not, then for at least one of the edges $\{u, x_1\}$ or $\{x_r, v\}$, no bag of the decomposition contains both end points. But if at least one internal vertex is in $X_i \cap X_j$, this path cannot exist in $G - (X_i \cap X_j)$.

Tutorial Exercise T9

Design a dynamic programming algorithm for the Independent Set problem on graphs of treewidth at most k. Assume that you are given as input, a graph, and a smooth tree decomposition of the graph. What is the running time of your algorithm?

Solution

Let $G = (V, E)$ be the input graph and let $\mathscr{T} = (T = (I, F), \{X_i \mid i \in I\})$ be a smooth tree decomposition of G of width k. Root the tree T at an arbitrary node r and for $i \in I$ let $V_i := \bigcup_{j \in T_i} X_j$, where T_i is the subtree of T rooted at the node i. For $S \subseteq X_i$, define $f_i(S)$ to be the size of a maximum independent set in $G[V_i]$ whose interesection with X_i is precisely S. The table associated with node $i \in I$ consists of at most 2^{k+1} rows, with each row corresponding to a distinct subset of the bag X_i. The entry for a row corresponding to $S \subseteq X_i$ is the number $f_i(S)$.

Here is how the tables are updated.

Leaf Nodes. If $i \in I$ is a leaf of T, then for $S \subseteq X_i$

$$f_i(S) = \begin{cases} |S|, & \text{if } S \text{ is an independent set in } G; \\ -\infty, & \text{otherwise.} \end{cases}$$

Non-leaf Nodes. Let i be an internal node with children $j_1, \ldots, j_p$. Then for $S \subseteq X_i$, set $f_i(S) = -\infty$ if S is not an independent set in G. Otherwise,

$$f_i(S) = |S| + \sum_{l=1}^{p} \max\{(f_{j_l}(W) - |S \cap W|)\},$$

where the maximum is taken over all $W \subseteq X_{j_l}$ such that the following conditions hold:

1. $S \cap X_{j_l} = W \cap X_i$;
2. $S \cup W$ is independent.

The first condition on W states that the vertices of X_i that are in W are precisely those vertices of S that are part of X_{j_l}. The intuition behind this is that $f_i(S)$ gives the size of a maximum independent set of $G[V_i]$ that intersects X_i in precisely S. Consider a child, say, X_{j_1} of X. How does such an independent set intersect X_{j_1}? If this independent set intersects X_{j_1} in a set W, then W must contain $S \cap X_{j_1}$. Additionally, W must not contain any other vertex in $X_i \setminus S$ because this would violate the condition that the independent set intersects X_i in precisely S. These two conditions may be stated as $S \cap X_{j_1} = W \cap X_i$, which is the first condition above. Moreover, we want $S \cup W$ to be independent, which is the second condition. Finally, we want to compute the maximum value of $f_{j_1}(W)$ over all such sets W. For each maximum value, we subtract the contribution of S in the independent set in T_{j_l} so as not to count the vertices of S multiple times. But this means that we need to add back $|S|$, which is exactly what we do.

Running Time. The work done at a node i is proportional to

$$4^k \cdot k^2 \cdot (\text{number of children of } i).$$

Thus the total work done is $O(4^k \cdot k^2 \cdot |F|)$ which is $O(4^k \cdot k^2 \cdot |V|)$.

Homework H5

Let $T = (V, E)$ be a tree and let $w \colon V \to \mathbf{R}^+ \cup \{0\}$ be a weighting of the vertices of the tree by nonnegative reals. Then show that there exists a set S of at most two vertices such that for every component $T[C]$ of $T[V \setminus S]$, $w(C) \leq w(V)/2$. Here for any $X \subseteq V$, $w(X) = \sum_{u \in X} w(u)$.

Generalize this result to show that if $\mathscr{T} = (T = (I, F), \{X_i \mid i \in I\})$ is a width k decomposition of $G = (V, E)$ and $w \colon I \to \mathbf{R}^+ \cup \{0\}$ is a weighting of the nodes of the tree, there exists $i \in I$ such that each component of $G[V \setminus X_i]$ has weight at most $w(V)/2$.

[10 points]

Homework H6

In this exercise, we want to design a dynamic programming algorithm for the DOMINATING SET problem on graphs of bounded treewidth. Assume that you have a nice tree decomposition $\mathscr{T} = (T = (I, F), \{X_i \mid i \in I\})$ to begin with. As usual, root this decomposition at an arbitrary node and for $i \in I$, let $V_i := \bigcup_{j \in T_i} X_j$, where T_i is the subtree of T rooted at X_i. We have to decide what information we wish to maintain in the tables associated with the nodes in the tree decomposition. In this dynamic program, every vertex u of the graph is in one of three states:

- State `Green` meaning that u is in the dominating set.
- State `Yellow` meaning that u is dominated but not in the dominating set.
- State `Red` meaning that u is not yet dominated.

A table entry for a node i (with bag $X_i = \{u_1, \ldots, u_p\}$) of the tree decomposition consists of a state configuration $(c_1, \ldots, c_p) \in \{\texttt{Green}, \texttt{Yellow}, \texttt{Red}\}^p$ of the vertices of the bag X_i and a number $s_i(c_1, \ldots, c_p)$. The number $s_i(c_1, \ldots, c_p)$ is the size of a minimum dominating set in the graph $G[V_i]$ such that vertex u_j is in state c_j. If no such dominating set exists then the entry is set to ∞.
If i is a leaf node with bag $X_i = \{u_1, \ldots, u_p\}$, then for any state configuration $\tilde{c} = (c_1, \ldots, c_p)$ of the vertices in X_i,

$$s_i(\tilde{c}) = \begin{cases} \infty, & \text{if some } c_j \text{ is } \texttt{Yellow} \text{ but } u_j \text{ is not dominated by any vertex in } X_i. \\ \#_{\texttt{Green}}(\tilde{c}). & \end{cases}$$

Let i be a forget node with child j and suppose that $X_i = \{u_1, \ldots, u_p\}$ and $X_j = \{u_1, \ldots, u_p, u\}$ (so that u is the vertex that is forgotten). Since u is being forgotten, to evaluate $s_i(c_1, \ldots, c_p)$ we need to make sure that u is either in the dominating set or is already dominated in T_j. One can verify that

$$s_i(c_1, \ldots, c_p) = \min_{c_{p+1} \in \{\texttt{Green}, \texttt{Yellow}\}} s_j(c_1, \ldots, c_p, c_{p+1}).$$

Show how one might update the introduce and join nodes

[10 points]