

Parameterized Algorithms Tutorial

Tutorial Exercise T9

Recall that a tree-decomposition of a graph $G = (V, E)$ is a pair $\langle T, \mathcal{X} = \{X_i \mid i \in V(T)\} \rangle$, where T is a tree whose vertices are called *nodes* and \mathcal{X} is a collection of subsets of $V(G)$ called *bags* such that the following hold.

1. $\bigcup_{i \in V(T)} X_i = V(G)$.
2. For each edge $\{x, y\} \in E(G)$, there exists $i \in V(T)$ such that $x, y \in X_i$.
3. For all $i, j, k \in V(T)$, if X_j is in the path between X_i and X_k in the tree T , then $X_j \supseteq X_i \cap X_k$.

Show that the last condition can be replaced by the equivalent condition: For each vertex $u \in V(G)$, the set of bags that contain u is a subtree of T .

Proposed Solution

We will show the two directions of “Bag condition \Leftrightarrow Subtree condition”

\Rightarrow Assume for some $v \in V(G)$, the bags containing v would *not* form a subtree in $\langle T, \mathcal{X} \rangle$. Then, because of Condition 1, we would could pick two bags X_i, X_j containing v such that some bag X_l inbetween X_i and X_j does *not* contain v . This already implies $X_l \not\supseteq X_i \cap X_j$.

\Leftarrow Assume there exist three bags X_i, X_j, X_l such that l lies on the path between i and j in T but $X_l \not\supseteq X_i \cap X_j$. Pick some vertex v from $(X_i \cap X_j) \setminus X_l$. But then the set of bags containing v cannot be connected and thus not form a tree.

Tutorial Exercise T10

A tree-decomposition $\langle T, \mathcal{X} = \{X_i \mid i \in V(T)\} \rangle$ of a graph $G = (V, E)$ is *nice* if it is rooted at some node and has only four types of nodes.

1. *Leaf nodes* i , the leaves of the decomposition, with $|X_i| = 1$.
2. *Introduce nodes* i that have exactly one child j such that $X_i = X_j \cup x$, for some vertex $x \in V(G)$.
3. *Forget nodes* i that have exactly one child j such that $X_i = X_j \setminus x$, for some vertex $x \in V(G)$.
4. *Join nodes* i that have exactly two children j and k such that $X_i = X_j = X_k$.

Given a tree decomposition $\langle T', \mathcal{X}' \rangle$ of G of width w , construct a nice tree-decomposition $\langle T, \mathcal{X} \rangle$ of G in polynomial time of width w such that $|V(T)| = O(w \cdot |V(T')|)$.

Proposed Solution

If a bag X_i is a leaf with $|X_i| > 1$ we can add a bag $X_i \setminus \{v\}$ for some $v \in X_i$ to it without changing the width of the decomposition. If we do so repeatedly, at some point each leaf has size 1. The number of bags added through this process is polyomial in the size of the treedecomposition.

Consider a bag X_i with child X_j such that $|X_i \cap X_j| > 1$. We add a new bag X_l between i and j with $X_l = X_i \cap X_j$. This neither invalidates the tree decomposition nor does it increase its width. After this process, we know that each parent-child pair X_i, X_j either satisfiyes $X_i \subseteq X_j$ or $X_j \subseteq X_i$. Wlog, assume $X_i \subseteq X_j$. If now $|X_i \cap X_j| > 1$, we introduce a bag X_l inbetween i and j with $X_l = (X_i \cap X_j) + v$ for some $v \in X_j \setminus X_i$. Repeating this process exhaustively will leave us with a tree decomposition where the intersection of two adjacent bags has size exactly one, which either constitutes a Forget- or an Introduce.

Finally, we need to take care of the join bags. Consider a join bag X_i with children X_{j_1}, \dots, X_{j_r} . We replace X_i by a binary tree of copies of X_i with r leafs and connect each leaf to one of the children X_{j_1}, \dots, X_{j_r} . Again, this neither invalidates the tree decomposition nor does it increase the width.

The polynomial running time of this procedure should be obvious.

Tutorial Exercise T11

Let $\langle T, \mathcal{X} \rangle$ be a tree-decomposition of a graph G . Suppose that the subtrees obtained by deleting a node t from T are T_1, \dots, T_r and for $1 \leq i \leq r$, let G_i be the graph induced by the vertices of G that are in the bags of T_i . Then the subgraphs $G_1 - X_t, \dots, G_r - X_t$ have no vertices in common and there are no edges between them.

Proposed Solution

First, note that if $\langle T, \mathcal{X} \rangle$ is a valid tree decomposition for a graph G then $\langle T', \mathcal{X}' \rangle$ with $T' \leq T$ and $\mathcal{X}' = \{X_i \in \mathcal{X} \mid i \in T'\}$ is a valid tree decomposition for $G' := G[\bigcup_{X' \in \mathcal{X}'} X']$. Otherwise, whichever condition is violated in $\langle T', \mathcal{X}' \rangle$ will also be violated in $\langle T, \mathcal{X} \rangle$.

Furthermore, if $\langle T, \mathcal{X} \rangle$ is a valid tree decomposition for G then it is also a valid tree decomposition for $G - v$ for some $v \in G$ (we can simply remove v from every bag).

Now, suppose the above was false, i.e. there are at least two subgraph $H_i = G_i - X_t, H_j = G_j, 1 \leq i, j \leq r$ with $H_1 \neq H_2$ that either share a vertex or an edge.

- *Vertex case:* Let $v \in H_i, H_j$ be a common vertex. By construction of H_1, H_2 it follows that $v \notin X_t$. As $\langle T_i, \mathcal{X}_i \rangle$ with $\mathcal{X}_i := \{X_l \setminus X_t \mid l \in T_i\}$ is valid tree decomposition for H_i , it must contain v in some bag. But the same holds for H_j with a similar defined tree decomposition $\langle T_j, \mathcal{X}_j \rangle$ and it follows that the then there exist two bags in subtrees of X_t that contain a vertex that is *not* in X_t . This contradicts our assumption of a valid tree decomposition $\langle T, \mathcal{X} \rangle$.
- *Edge case:* Analogous.

Tutorial Exercise T12

The notion of treewidth can be defined in several ways. One way to frame the definition of treewidth is by using the following game called the *cops-and-robber* game. The game consists of a set of cops trying to catch a robber. The robber lives in the the graph and

can move with infinite speed along the edges of the graph. He cannot, however, move through a vertex should a cop be guarding it. The cops move about in helicopters, the point being that they are not constrained to move along the edges of the graph, but they have finite speed. The game proceeds as follows. Initially, the robber occupies some vertex of the graph. The cops announce their positions (a set of vertices) and move towards them with finite speed. Seeing their positions, the robber announces his position (a vertex) and moves to that vertex instantaneously. Not all cops need land on vertices at once and not all cops need change positions, that is, if a cop occupies a vertex, it may continue occupying that vertex in the next move of the game. The cops *catch* the robber when one of them lands on a vertex occupied by him.

Show that if a graph has treewidth k then $k + 1$ cops can always catch the robber in a cops-and-robbber game on the graph. The converse also holds and this is not so easy to show. In the exercises, we will assume this equivalent formulation of treewidth.

Proposed Solution

Assume a nice tree decomposition. Then the $k + 1$ cops can catch the robber as follows: first k of them occupy the vertices of the root bag. As shown in T11, this separates the subgraphs induced by the non-occupied bags. When the robber announces his position, the cops can close in on him by moving in the direction of that subgraph. This is clearly possible if the current bag is a non-join bag by moving a single cop (Introduce: move a cop to the new vertex, Forget: remove a cop). If the bag is a join bag, then, in a nice tree decomposition, all children contain the exact same vertices—thus this reduced to the previous case.

As in each step the component in which the robber resides shrinks by one vertex, the cops always win in a finite number of steps.

Homework H7

Use the properties of a tree-decomposition to show that if a graph G contains a clique C , then every tree-decomposition $\langle T, \mathcal{X} \rangle$ of G has a bag X such that $V(C) \subseteq X$.

Proposed Solution

For $|C| = 1$ and $|C| = 2$ this holds per definition. Let us prove the rest by induction. Assume the above holds for cliques of size r . Consider a clique $C \subseteq V(G)$ with $|C| = r + 1$. If the above does not hold, then there exists a tree decomposition $\langle T, \mathcal{X} \rangle$ of G such that $C \not\subseteq X$ for each $X \in \mathcal{X}$. Pick three distinct vertices $u, v, w \in C$. By assumption, the cliques $C - u, C - v$ and $C - w$ must occur in at least one bag. Let us denote these bags by X_u, X_v and X_w , i.e. $C - a \subseteq X_a$ for $a \in \{u, v, w\}$.

Now, there must exist a bag $Y \in \mathcal{X}$ from which every of the three bags can be reached via a path that does not touch the other bag (this includes the case that Y is one of the three bags). But then we have the following:

- $C \setminus \{u, v, w\} \subseteq Y$ as these vertices are in the common intersection of all three bags
- $u \in Y$ because Y lies between X_v and X_w which contain u
- $v \in Y$ because Y lies between X_u and X_w which contain v
- $w \in Y$ because Y lies between X_u and X_v which contain w

It immediately follows that therefore, $C \subseteq Y$ contrary to our assumption.

Homework H8

Let $\langle T, \mathcal{X} \rangle$ be a tree-decomposition of a graph G . Suppose that the subtrees obtained by deleting an edge $\{i, j\} \in E(T)$ are T_i, T_j and let G_i, G_j be the subgraphs induced by the vertices in the bags of T_i and T_j , respectively. Show that deleting the set $X_i \cap X_j$ from $V(G)$ disconnects G into two subgraphs $G_i - (X_i \cap X_j)$ and $G_j - (X_i \cap X_j)$; that is, they do not share vertices and there is no edge with one end in each of them.

Proposed Solution

Analogous to T11.

Homework H9

Use the cops-and-robber formulation to show that the $k \times k$ grid has treewidth k . That is, show that $k + 1$ cops are necessary and sufficient for them to be able to catch the robber on the $k \times k$ grid.

Proposed Solution

The strategy for $k + 1$ cops is simple: occupy the leftmost column of the grid and then move forward through the grid.

For the lower bound, i.e. that k cops do *not* suffice (for $k \geq 1$), we need to provide a winning strategy for the *robber*. Let G be $k \times k$ -grid and let further $C_i \subseteq V(G)$ denote the set of vertices occupied by cops in round i . Similarly, let $R_i \subseteq V(G)$ be the vertices of the component in which the robber resides.

Definition 1. Let C_i be the current positions of the cops and let the robber be positioned in some component R_i of $G - C_i$. Let further C_{i+1} be the next positioning of the cops. If the robber can choose a component R_{i+1} of $G - C_{i+1}$ with $|R_{i+1}| \geq |R_i|$, we call this an escape move.

If we can show that the robber can always make an escape move, it follows that he has a winning strategy: the component in which he resides will always be as large as the last one, therefore he is never caught.

Lemma 1. If the cops are positioned in one row (or column) of G , then the robber has an escape move.

Proof. Let C_i be the current positioning of the cops and let this be either a row or a column of vertices. Consider C_{i+1} :

1. If $C_i = C_{i+1}$ then just holding his position is an escape move for the robber
2. If $C_i \neq C_{i+1}$ then $C_i \cap C_{i+1}$ is not separator of the grid (i.e. $G - (C_i \cup C_{i+1})$ has one component) and therefore will automatically reside in a component of size $k^2 - k$, which cannot possibly be smaller as before

It follows that in both cases the robber has an escape move. □ □

Lemma 2. Let $S \subseteq V(G)$ be a vertex set with $|S| = k - 1$ such that $G - S$ contains more than one connected component. Then one component of $G - S$ has at least size $\frac{k(k+1)}{2}$.

Proof. It is easy to see that in order to minimize the size of all components, one would divide the grid into exactly two components. The best one can do, then, is separate a portion of size $\frac{(k-1)(k-2)}{2}$ from the rest of the graph by taking a diagonal of size $k-1$. The remaining component then has a size of $\frac{k(k+1)}{2}$. \square

Theorem 1. *On a $k \times k$ grid with k cops, the robber has a winning strategy by choosing always the largest possible component.*

Proof. We will show that $|R_i| \geq \frac{k(k+1)}{2} - 1$ for all $i \geq 0$. For the initial placement C_0, R_0 , this clearly holds. Now, assume the above during the i state of the game. Let us first handle the following easy cases:

- The cops do not move, i.e. $C_i = C_{i+1}$: the robber just stays in his component
- The cops occupy a row or column: then, by Lemma 1, the robber can find a component of size at least as large as $|R_i|$. It follows that $|R_{i+1}| \geq |R_i| \geq \frac{k(k+1)}{2} - 1$
- If $G - C_i$ has only one connected component, the robber can choose any connected component of $G - C_{i+1}$, therefore $|R_{i+1}| \geq \frac{k(k+1)}{2} - 1$ is easily achievable by just taking the largest one.

Otherwise, $G - C_i$ has at least two components and the cops do not occupy a row or column. If $G - (C_i \cap C_{i+1})$ has one connected component, the robber can choose freely until the cops arrive at C_{i+1} , thus he can find a sufficiently large component. But if $G - (C_i \cap C_{i+1})$ consists of multiple components *and* the cops change their position, then $|C_i \cap C_{i+1}| \leq k-1$ and by Lemma 2 one of these components has at least size $\frac{k(k+1)}{2}$. As then only one cop can move, we have that $|R_{i+1}| \geq \frac{k(k+1)}{2} - 1$. \square