Steiner Tree Parameterized by Treewidth

Kousshik Raj M (17CS30022)

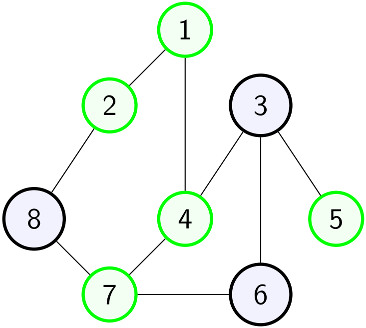
October 16, 2020

Problem

Given a graph G, a set of terminal vertices K, its tree decomposition of width at most k, find a connected subgraph of minimum possible size that contains all the terminals.

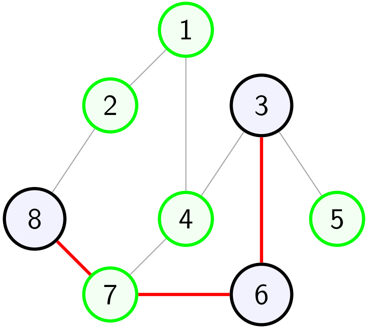
Problem

Given a graph G, a set of terminal vertices K, its tree decomposition of width at most k, find a connected subgraph of minimum possible size that contains all the terminals.



Problem

Given a graph G, a set of terminal vertices K, its tree decomposition of width at most k, find a connected subgraph of minimum possible size that contains all the terminals.



Tree Decomposition (Recap)

A tree decomposition of graph G is $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$, where T is a tree, $X_t \subseteq V(G)$, such that

Tree Decomposition (Recap)

A tree decomposition of graph G is $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$, where T is a tree, $X_t \subseteq V(G)$, such that

- $\displaystyle\bigcup_{t \in V(T)} X_t = V(G)$

Tree Decomposition (Recap)

A tree decomposition of graph G is $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$, where T is a tree, $X_t \subseteq V(G)$, such that

- $\bigcup\limits_{t \in V(T)} X_t = V(G)$
- $\forall (u, v) \in E(G), \exists t \in V(T)$ s.t $\{u, v\} \subseteq X_t$

Tree Decomposition (Recap)

A tree decomposition of graph G is $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$, where T is a tree, $X_t \subseteq V(G)$, such that

- $$\bigcup_{t \in V(T)} X_t = V(G)$$
- $\forall (u, v) \in E(G), \exists t \in V(T)$ s.t $\{u, v\} \subseteq X_t$
- $T_u = \{t \in V(T) : u \in X_t\}$ induces a connected a subtree, $\forall u \in V(G)$

Tree Decomposition (Recap)

Lemma 1

For a tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$, let $(a, b) \in E(T)$, and T_a, T_b be the two connected components in $T - \{(a, b)\}$, containing a and b, respectively. Furthermore, let $A = \bigcup_{t \in V(T_a)} X_t$ and $B = \bigcup_{t \in V(T_b)} X_t$.

- $A \cap B = X_a \cap X_b$
- There is no edge between $A - B$ and $B - A$

Nice Tree Decomposition (Recap)

A tree decomposition $(T, \{X_t\}_{t \in V(T)})$, where T is a tree rooted at r, is called nice if

Nice Tree Decomposition (Recap)

A tree decomposition $(T, \{X_t\}_{t \in V(T)})$, where T is a tree rooted at r, is called nice if

- $X_r = X_l = \phi$, for every leaf l of T.

Nice Tree Decomposition (Recap)

A tree decomposition $(T, \{X_t\}_{t \in V(T)})$, where T is a tree rooted at r, is called nice if

- $X_r = X_l = \phi$, for every leaf l of T.
- Every non-leaf node is one of the following types:
  Introduce Node:- A node t with only one child t', such that $X_t = X_{t'} \cup \{v\}, v \notin X_{t'}$
  Forget Node:- A node t with only one child t', such that $X_t = X_{t'} - \{v\}, v \in X_{t'}$
  Join Node:- A node t with exactly two children t_1, t_2 such that $X_t = X_{t_1} = X_{t_2}$

Nice Tree Decomposition (Variant)

We have graph G, a set of terminals K, a nice tree decomposition (of G) $(T, \{X_t\}_{t \in V(T)})$. We now make the following changes to the nice tree decomposition:-

Nice Tree Decomposition (Variant)

We have graph G, a set of terminals K, a nice tree decomposition (of G) $(T, \{X_t\}_{t \in V(T)})$. We now make the following changes to the nice tree decomposition:-

- Create a new type of node called Introduce Edge Node.
  Introduce Edge Node:- A node t, labelled with edge $(u, v) \in E(G)$ such that $u, v \in X_t$, and with only one child t' such that $X_t = X_{t'}$. Here, (u, v) is introduced at t.

Nice Tree Decomposition (Variant)

We have graph G, a set of terminals K, a nice tree decomposition (of G) $(T, \{X_t\}_{t \in V(T)})$. We now make the following changes to the nice tree decomposition:-

- Create a new type of node called Introduce Edge Node.
  Introduce Edge Node:- A node t, labelled with edge $(u, v) \in E(G)$ such that $u, v \in X_t$, and with only one child t' such that $X_t = X_{t'}$. Here, (u, v) is introduced at t.
- There is exactly one Introduce Edge node corresponding to each edge of $E(G)$

Nice Tree Decomposition (Variant)

We have graph G, a set of terminals K, a nice tree decomposition (of G) $(T, \{X_t\}_{t \in V(T)})$. We now make the following changes to the nice tree decomposition:-

- Create a new type of node called Introduce Edge Node.
  Introduce Edge Node:- A node t, labelled with edge $(u, v) \in E(G)$ such that $u, v \in X_t$, and with only one child t' such that $X_t = X_{t'}$. Here, (u, v) is introduced at t.
- There is exactly one Introduce Edge node corresponding to each edge of $E(G)$
- Choose arbitrary $v_0 \in K$, and add it to all the bags in the tree decomposition. So, we have, $v_0 \in X_t, \forall t \in V(T)$

Algorithm (Definitions)

- For $t \in V(T)$, $G_t = \{V_t = \bigcup_{t \in V(T_t)} X_t, E_t = \{(u,v) : E_{uv} \in V_t\}\}$, where T_t is the subtree rooted at t, and E_{uv} is the node that introduces edge (u, v)

Algorithm (Definitions)

- For $t \in V(T)$, $G_t = \{V_t = \bigcup_{t \in V(T_t)} X_t, E_t = \{(u, v) : E_{uv} \in V_t\}\}$, where T_t is the subtree rooted at t, and E_{uv} is the node that introduces edge (u, v)
- H is minimum Steiner tree for graph G and terminals K. H_t is the part of H in G_t, with connected components $C_1, C_2, ..., C_q$.

Algorithm (Definitions)

- For $t \in V(T)$, $G_t = \{V_t = \bigcup_{t \in V(T_t)} X_t, E_t = \{(u, v) : E_{uv} \in V_t\}\}$, where T_t is the subtree rooted at t, and E_{uv} is the node that introduces edge (u, v)
- H is minimum Steiner tree for graph G and terminals K. H_t is the part of H in G_t, with connected components $C_1, C_2, ..., C_q$.
- $H_t \neq \phi$, and $X_t \cap C_i \neq \phi, \forall 1 \leq i \leq q$

Algorithm (Definitions)

- For $t \in V(T)$, $G_t = \{V_t = \bigcup_{t \in V(T_t)} X_t, E_t = \{(u, v) : E_{uv} \in V_t\}\}$, where T_t is the subtree rooted at t, and E_{uv} is the node that introduces edge (u, v)
- H is minimum Steiner tree for graph G and terminals K. H_t is the part of H in G_t, with connected components $C_1, C_2, ..., C_q$.
- $H_t \neq \phi$, and $X_t \cap C_i \neq \phi, \forall 1 \leq i \leq q$
- $\forall t \in V(T)$, $\forall X \subseteq X_t$, $\forall P$ is a partition of X, $dp[t, X, P]$ denotes the size of the smallest (edgewise) subgraph H_t such that
  - $K \cap V_t \subseteq V(H_t)$
  - $V(H_t) \cap X_t = X$
  - $C_i \cap X = P_i$, where $P_i \in P$, $\forall 1 \leq i \leq q$

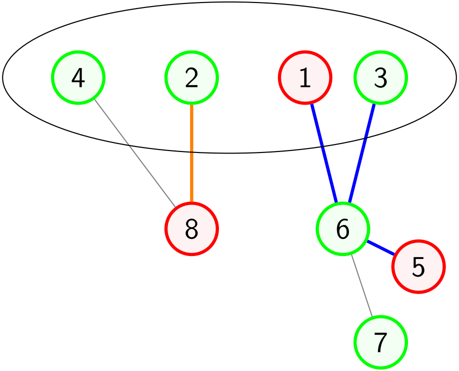
Algorithm (Definitions)

- For $t \in V(T)$, $G_t = \{V_t = \bigcup_{t \in V(T_t)} X_t, E_t = \{(u, v) : E_{uv} \in V_t\}\}$, where T_t is the subtree rooted at t, and E_{uv} is the node that introduces edge (u, v)
- H is minimum Steiner tree for graph G and terminals K. H_t is the part of H in G_t, with connected components $C_1, C_2, ..., C_q$.
- $H_t \neq \phi$, and $X_t \cap C_i \neq \phi, \forall 1 \leq i \leq q$
- $\forall t \in V(T)$, $\forall X \subseteq X_t$, $\forall P$ is a partition of X, $dp[t, X, P]$ denotes the size of the smallest (edgewise) subgraph H_t such that
  - $K \cap V_t \subseteq V(H_t)$
  - $V(H_t) \cap X_t = X$
  - $C_i \cap X = P_i$, where $P_i \in P$, $\forall 1 \leq i \leq q$
- $dp[t, X, P] = \infty$, if no such H_t is possible

Algorithm (Definitions)

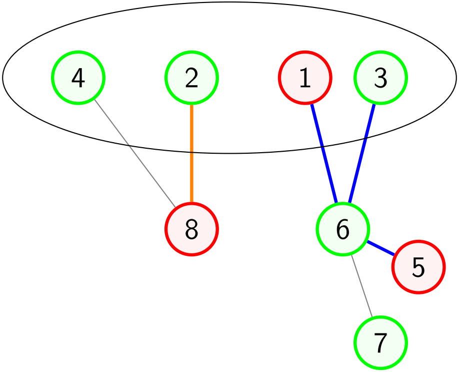
- For $t \in V(T)$, $G_t = \{V_t = \bigcup_{t \in V(T_t)} X_t, E_t = \{(u,v) : E_{uv} \in V_t\}\}$, where T_t is the subtree rooted at t, and E_{uv} is the node that introduces edge (u, v)
- H is minimum Steiner tree for graph G and terminals K. H_t is the part of H in G_t, with connected components $C_1, C_2, \ldots, C_q$.
- $H_t \neq \phi$, and $X_t \cap C_i \neq \phi, \forall 1 \leq i \leq q$
- $\forall t \in V(T)$, $\forall X \subseteq X_t$, $\forall P$ is a partition of X, $dp[t, X, P]$ denotes the size of the smallest (edgewise) subgraph H_t such that
  - $K \cap V_t \subseteq V(H_t)$
  - $V(H_t) \cap X_t = X$
  - $C_i \cap X = P_i$, where $P_i \in P$, $\forall 1 \leq i \leq q$
- $dp[t, X, P] = \infty$, if no such H_t is possible
- $dp[r, \{v_0\}, \{\{v_0\}\}]$ is the size of minimum Steiner tree for G

Algorithm (Definitions)



Algorithm (Definitions)

- $V_t = \{1, 2, ..., 8\},\ E_t = \{(2, 8), (1, 6), (3, 6), (5, 6), (4, 8), (6, 7)\}$
- $K \cap V_t = \{1, 5, 8\}, X_t = \{1, 2, 3, 4\}$
- $H_t = \{\{1, 2, 3, 5, 6, 8\}, \{(2, 8), (1, 6), (3, 6), (5, 6)\}\}$
- $X = \{1, 2, 3\},\ P = \{\{1, 3\}, \{2\}\}$
- $dp[t, X, P] = 4$



Algorithm (Transitions)

Leaf Node

$$dp[t, X, P] = \begin{cases} 0 & X = \{v_0\} \\ \infty & otherwise \end{cases}$$

t is a leaf node

- In a leaf node t, $X_t = \{v_0\}$. There are only two choices for X.
- If $X = \phi$, $P = \phi$, there is no valid H_t, as it doesn't include the terminal v_0
- If $X = \{v_0\}$, $P = \{\{v_0\}\}$, we can treat v_0 as H_t.

Algorithm (Transitions)

Introduce Vertex Node

$$dp[t, X, P] = \begin{cases} dp[t', X - \{v\}, P - \{\{v\}\}] & v \in X, \{v\} \in P \\ dp[t', X, P] & v \notin K, v \notin X \\ \infty & otherwise \end{cases}$$

t, whose child is t', introduces vertex v

- If v is a terminal, then v must be in X.
- v is isolated in G_t. So, if $v \in X$, then $\{v\}$ must be a block of P. We can ignore v from H_t and use previously calculated value from child
- If $v \notin X$, $H_t = H_{t'}$

Algorithm (Transitions)

Introduce Edge Node

$$dp[t, X, P] = \begin{cases} dp[t', X, P] & u \text{ or } v \notin X \\ dp[t', X, P] & P_u \neq P_v \\ min\{\min\limits_{\forall P'_{u+v} = P} dp[t', X, P'] + 1, dp[t', X, P]\} & otherwise \end{cases}$$

t, whose child is t', introduces edge (u, v)

P_c is the block in the partition where vertex c occurs

P_{u+v} is the resulting partition after merging the blocks of vertices u and v

- If either u or v is not in X, we cannot add the edge.
- If u and v occur in separate connected components, we cannot consider the edge.
- If they occur together, then we consider the case where the edge is included and the case where it is not.

Algorithm (Transitions)

Forget Node

$$dp[t, X, P] = min\{\min_{\forall P', P'_{|v} = P} dp[t', X \cup \{v\}, P^{'}], dp[t', X, P]\}$$

t, whose child is t', forgets vertex v

$P_{|v}$ is a new partition obtained from P after removing the occurrence of v

- There are two cases, one where v is considered and one where it is not
- If v is considered, then it is included in one of the existing partitions, and minimum of all such partitions is calculated
- If v is not considered, $H_t = H'_t$

Algorithm (Transitions)

Join Node

$$dp[t, X, P] = \min_{\forall P_1 \bigoplus P_2 = P} dp[t_1, X, P_1] + dp[t_2, X, P_2]$$

t, whose children are t_1 and t_2, is a join node

$\bigoplus$ denotes a merge of two partitions

- We need to merge two partial solutions H_{t_1} and H_{t_2} into H_t.
- We can treat partitions P_1 and P_2 as forests, with each connected component corresponding to exactly one of the blocks
- If we merge the two forests edgewise, and if the connected components of the resultant multigraph correspond to the blocks in partition P, then $P = P_1 \bigoplus P_2$

Algorithm (Time Complexity)

- $\forall t \in V(T), |X_t| \leq k + 2$. Number of states per node is $2^{(k+2)} \cdot (k+2)^{(k+2)} = O(k^{O(k)})$

Algorithm (Time Complexity)

- $\forall t \in V(T), |X_t| \leq k+2$. Number of states per node is $2^{(k+2)} \cdot (k+2)^{(k+2)} = O(k^{O(k)})$
- Worst case transition complexity is from join node, $(k+2)^{(k+2)} \cdot (k+2)^{(k+2)} = O(k^{O(k)})$

Algorithm (Time Complexity)

- $\forall t \in V(T), |X_t| \leq k + 2.$ Number of states per node is $2^{(k+2)} \cdot (k+2)^{(k+2)} = O(k^{O(k)})$
- Worst case transition complexity is from join node, $(k+2)^{(k+2)} \cdot (k+2)^{(k+2)} = O(k^{O(k)})$
- Total computations per node is $O(k^{O(k)})$. Total number of nodes is $O(kn)$

Algorithm (Time Complexity)

- $\forall t \in V(T), |X_t| \leq k+2$. Number of states per node is $2^{(k+2)} \cdot (k+2)^{(k+2)} = O(k^{O(k)})$
- Worst case transition complexity is from join node, $(k+2)^{(k+2)} \cdot (k+2)^{(k+2)} = O(k^{O(k)})$
- Total computations per node is $O(k^{O(k)})$. Total number of nodes is $O(kn)$
- The algorithm runs in $O(k^{O(k)}.n)$

THANK YOU!