Advanced Algorithms (II)

Yijia Chen
Fudan University

Review

Basics

1. Big-O notation
2. Divide and conquer
3. DFS and BFS
4. Independent set in trees

Treewidth

Tree decompositions of graphs

Definition

Let $\mathcal{G}$ be a graph. A tree decomposition of $\mathcal{G}$ is a tuple $\left(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})}\right)$, where $\mathcal{T}$ is a tree and B_t the bag at t such that the following conditions are satisfied:

(T1) For every $v \in V(\mathcal{G})$ the set

$$T_v := \left\{ t \in V(\mathcal{T}) \;\middle|\; v \in B_t \right\}$$

is nonempty and connected in $\mathcal{T}$, i.e., $\mathcal{T}[T_v]$ is a subtree of $\mathcal{T}$.

(T2) For every $e \in E(G)$ there exists a $t \in V(\mathcal{T})$ such that $e \subseteq B_t$.

Tree decomposition of graphs, examples

1. The complete graphs $\mathcal{K}_n$ for $n \in \mathbb{N}$.
2. The trees.
3. The grids $\mathcal{G}_{n \times n}$ for $n \in \mathbb{N}$.

Treewidth

The width of a tree decomposition $\big(\mathcal{T}, (B_t)_{t\in V(\mathcal{T})}\big)$ is

$$\mathrm{width}\big(\mathcal{T}, (B_t)_{t\in V(\mathcal{T})}\big) := \max\big\{|B_t| - 1 \;\big|\; t \in V(\mathcal{T})\big\}.$$

The treewidth of $\mathcal{G}$ is

$$\mathrm{tw}(\mathcal{G}) := \min\Big\{\mathrm{width}\big(\mathcal{T},(B_t)_{t\in V(\mathcal{T})}\big) \;\Big|\; \big(\mathcal{T}, (B_t)_{t\in V(\mathcal{T})}\big) \text{ is a tree decomposition of } \mathcal{G}\Big\}.$$

Treewidth, examples

1. $\mathrm{tw}(\mathcal{K}_n) = n - 1$ for the complete graphs $\mathcal{K}_n$.
2. $\mathrm{tw}(\mathcal{T}) = 1$ for every tree $\mathcal{T}$ of size at least 2.
3. $\mathrm{tw}(\mathcal{G}_{n \times n}) = n$ for every grid $\mathcal{G}_{n \times n}$.

Smooth tree decomposition

Definition

A tree decomposition $\left(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})}\right)$ is smooth if for every $\{t, t'\} \in E(T)$ we have

$$|B_t \setminus B_{t'}| = |B_{t'} \setminus B_t| = 1.$$

Theorem

Every tree decomposition can be efficiently transferred to a smooth one of the same width.

Theorem

Every graph $\mathcal{G}$ has a smooth tree decomposition of width $\mathrm{tw}(\mathcal{G})$.

Make tree decomposition smooth

Let $\left(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})}\right)$ be a tree decomposition of width w.

1. Make bags equal size: We choose a node $r \in V(\mathcal{T})$ with $|B_r| = w + 1$ as the root. Let t be a child of r with $|B_t| \leq w$. Clearly

$$|B_r \setminus B_t| + |B_t| \geq w + 1.$$

   We add $w + 1 - |B_t|$ vertices in $B_r \setminus B_t$ to B_t. After repeating this procedure recursively from the root to leaves, every bag has size $w + 1$.

2. Remove repetition: If there is an edge $\{t, t'\} \in E(\mathcal{T})$ with $B_t = B_{t'}$, then we merge t' with t.

3. Interpolation: Let $\{t, t'\} \in E(\mathcal{T})$ with $|B_t \cap B_{t'}| < w$, i.e.,

$$B_t \setminus B_{t'} = \{u_1, \ldots, u_\ell\} \quad \text{and} \quad B_{t'} \setminus B_t = \{v_1, \ldots, v_\ell\}$$

   for some $\ell > 2$ and pairwise distinct $u_1, \ldots, u_\ell, v_1, \ldots, v_\ell$. We insert new nodes $t_1, \ldots, t_{\ell-1}$ between t and t' with

$$B_{t_i} := \left(B_t \cap B_{t'}\right) \cup \{v_1, \ldots, v_i, u_{i+1}, \ldots, u_\ell\}$$

   for every $i \in [\ell - 1]$.

The size of smooth tree decompositions

Theorem

For every smooth tree decomposition $\left(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})}\right)$ *of* $\mathcal{G}$ *we have*

$$|V(\mathcal{T})| \leq |V(\mathcal{G})|.$$

Theorem

$$|E(\mathcal{G})| \leq \mathsf{tw}(\mathcal{G}) \cdot |V(\mathcal{G})|.$$

$\mathrm{tw}(\mathcal{K}_n) = n - 1$

$\mathrm{tw}(\mathcal{K}_n) \leq n - 1$: Take a tree decomposition with a singleton tree.

$\mathrm{tw}(\mathcal{K}_n) \geq n - 1$: Let $\left(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})}\right)$ be a smooth tree decomposition of $\mathcal{K}_n$ of width $\mathrm{tw}(\mathcal{K}_n)$. We show that there exists a B_t with $|B_t| = n$.

Trivial if $|V(\mathcal{T})| = 1$. Otherwise choose a leaf t and let t' be its parent in $\mathcal{T}$. By the smoothness

$$B_t \setminus B_{t'} = \{v\} \text{ for some } v \in V(\mathcal{K}_n).$$

Since v is adjacent to every other vertex in $\mathcal{K}_n$, we see that

$$B_t = V(\mathcal{K}_n).$$

Helly property for trees

Theorem

Let $\mathcal{T}$ be a tree and $\mathcal{T}_1, \ldots, \mathcal{T}_n$ subtrees of $\mathcal{T}$ such that

$$V(\mathcal{T}_i) \cap V(\mathcal{T}_j) \neq \emptyset$$

for every $i, j \in [n]$. Then

$$\bigcap_{i \in [n]} V(\mathcal{T}_i) \neq \emptyset.$$

Proof (Yaokun Wu, 2006)

We prove by induction on the size of $\mathcal{T}$.

Trivial for $|V(\mathcal{T})| = 1$.

Otherwise, let t be a leaf of $\mathcal{T}$. If $t \in V(\mathcal{T}_i)$ for every $i \in [n]$, then we are done.

Now assume $t \notin V(\mathcal{T}_i)$ for some $i \in [n]$. Consider

$$\mathcal{T} \setminus \{t\}, \mathcal{T}_1 \setminus \{t,\}, \ldots, \mathcal{T}_n \setminus \{t\}.$$

Then

- every $\mathcal{T}_i \setminus \{t\}$ is a (nonempty) subtree of $\mathcal{T} \setminus \{t\}$;
- $V(\mathcal{T}_i \setminus \{t\}) \cap V(\mathcal{T}_j \setminus \{t\}) \neq \emptyset$ for every $i, j \in [n]$.

The result follows from the induction hypothesis. $\square$

$\mathrm{tw}(\mathcal{K}_n) = n - 1$, again

Theorem

Let $\mathcal{G} = (V, E)$ be a graph and $S \subseteq V$ a clique. Then for every tree decomposition $\left(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})}\right)$ there is a node $t \in V(\mathcal{T})$ with $S \subseteq B_t$.

Proof.

For every $v \in V$ recall

$$T_v := \{t \in V(\mathcal{T}) \mid v \in B_t\}$$

induces a subtree $\mathcal{T}_v := \mathcal{T}[T_v]$ of $\mathcal{T}$.

Clearly for every $u, v \in S$ we have

$$V(\mathcal{T}_u) \cap V(\mathcal{T}_v) = T_u \cap T_v \neq \emptyset,$$

since there is an edge $\{u, v\}$ in $\mathcal{G}$.

The result follows from Helly property. □

Computing the treewidth

Theorem (Bodlaender, 1996)

The problem

TREEWIDTH	
Input:	A graph $\mathcal{G}$ and a number $k \in \mathbb{N}$.
Problem:	Decide whether $\mathrm{tw}(\mathcal{G}) \leq k$ and if so output a tree decomposition of $\mathcal{G}$ with width $\leq k$.

can be computed in time

$$2^{k^{O(1)}} \cdot \|\mathcal{G}\|.$$

Corollary

For every $k \in \mathbb{N}$ there is a linear time algorithm which on every graph $\mathcal{G}$ either outputs a tree decomposition of $\mathcal{G}$ of width $\leq k$ or reports that $\mathrm{tw}(\mathcal{G}) > k$.

Independent sets via tree decompositions

Independent sets via tree decompositions (1)

Let $\mathcal{G}$ be a graph and $\big(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})}\big)$ a smooth tree decomposition of $\mathcal{G}$. And let $w := \text{width}\big(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})}\big)$.

We fix an arbitrary node $r \in V(\mathcal{T})$ as the root of $\mathcal{T}$.

Let $t \in V(\mathcal{T})$. We define $\mathcal{G}_t$ as the induced subgraph of G on vertices in B_t.

Furthermore, $\mathcal{G}_{\leq t}$ is the induced subgraph of $\mathcal{G}$ on vertices in

$$B_t \cup \bigcup_{\text{descendants } t' \text{ of } t} B_{t'}.$$

Independent sets via tree decompositions (2)

By dynamic programming we compute for every $t \in V(\mathcal{T})$ and every $X \subseteq B_t$ independent in $\mathcal{G}_t$:

$$I(t, X) := \text{size of a largest independent set } I \text{ of } \mathcal{G}_{\leq t} \text{ with } I \cap B_t = X$$
$$= |X| + \sum_{\text{children } t' \text{ of } t} \max\left\{ I(t', X') - |X' \cap X| \;\middle|\; X' \cap B_t = X \cap B_{t'} \right\}.$$

Note there are at most

$$|V(\mathcal{T})| \cdot 2^{w+1} \leq |V(\mathcal{G})| \cdot 2^{w+1}$$

many $I(t, X)$.

Independent sets via tree decompositions (3)

Theorem

For every $k \in \mathbb{N}$ there is a linear time algorithm which on every graph $\mathcal{G}$ with $\mathsf{tw}(\mathcal{G}) \leq k$ outputs a largest independent set in $\mathcal{G}$.

Partial k-Trees

k-trees and partial k-trees.

Definition

Let $k \in \mathbb{N}$. Then the set of k-trees is defined as follows.

(K1) A complete graph $\mathcal{K}_{k+1}$ is a k-tree.

(K2) Let $\mathcal{G}$ be a graph and $v \in V$ such that

- $\mathcal{N}^{\mathcal{G}}[v]$ is isomorphic to $\mathcal{K}_{k+1}$, where $\mathcal{N}^{\mathcal{G}}[v]$ is the induced subgraph of $\mathcal{G}$ on

$$N^{\mathcal{G}}[v] := \{u \in V(\mathcal{G}) \mid \{u, v\} \in E(\mathcal{G})\} \cup \{v\}$$

- $\mathcal{G}[V(\mathcal{G}) \setminus \{v\}]$ is a k-tree.

Then $\mathcal{G}$ is a k-tree.

Definition

A graph is a partial k-tree if it is a subgraph of a k-tree.

Partial k-trees and bounded treewidth

Theorem

A graph $\mathcal{G}$ is a partial k-tree if and only if $\mathrm{tw}(\mathcal{G}) \leq k$.

Lemma

Let $\mathcal{G}$ be a subgraph of $\mathcal{H}$, i.e., $V(\mathcal{G}) \subseteq V(\mathcal{H})$ and $E(\mathcal{G}) \subseteq E(\mathcal{H})$. Then $\mathrm{tw}(\mathcal{G}) \leq \mathrm{tw}(\mathcal{H})$.

Theorem

1. *Every graph of treewidth $\leq k$ is a partial k-tree.*
2. *Every k-tree has a tree decomposition of width $\leq k$.*

Proof

Let $\mathcal{G}$ be a graph with $\mathrm{tw}(\mathcal{G}) \leq k$. Moreover, let $\mathscr{T} = \big(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})}\big)$ be a smooth tree decomposition of $\mathcal{G}$ of width k.

From $\mathscr{T}$ we define a graph $\mathcal{H}_{\mathscr{T}}$ by induction on $V(\mathcal{T})$ such that

(H1) $\mathcal{H}_{\mathscr{T}}$ is a k-tree;

(H2) $\mathcal{H}_{\mathscr{T}}[B_t]$ is isomorphic to $\mathcal{K}_{k+1}$ for every $t \in V(\mathcal{T})$.

(H3) $\mathcal{G} \subseteq \mathcal{H}_{\mathscr{T}}$.

If $|V(\mathcal{T})| = 1$, then $\mathcal{H}_{\mathscr{T}}$ is $\mathcal{K}_{k+1}$, and we are done.

Otherwise, choose a leaf t and let t' be its parent in $\mathcal{T}$. Therefore, $B_t \setminus B_{t'} = \{v\}$ for some $v \in V(\mathcal{K}_n)$. Then $\mathscr{T}' := \big(\mathcal{T} \setminus \{t\}, (B_t)_{t \in V(\mathcal{T} \setminus \{t\})}\big)$ is a smooth tree decomposition of the graph $\mathcal{G} \setminus \{v\}$.

By (H2) of the induction hypothesis, $\mathcal{H}_{\mathscr{T}'}[B_t \cap B_{t'}]$ is isomorphic to $\mathcal{K}_k$. Then from $\mathcal{H}_{\mathscr{T}'}$, we obtain $\mathcal{H}_{\mathscr{T}}$ by adding the vertex v and the edges $\{v, u\}$ for every $u \in B_t \cap B_{t'}$.

Proof (cont'd)

Let $\mathcal{H}$ be a k-tree. We show that $\mathrm{tw}(\mathcal{H}) \leq k$ by induction on the construction of $\mathcal{H}$.

If $\mathcal{H}$ is isomorphic to $\mathcal{K}_{k+1}$, i.e., (K1), then we are done.

Otherwise by (K2) let $v \in V(\mathcal{H})$ satisfy that $\mathcal{H} \setminus \{v\}$ is a k-tree and $\mathcal{N}^{\mathcal{G}}[v]$ is isomorphic to $\mathcal{K}_{k+1}$.

By induction hypothesis, there is a tree decomposition $\left(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})}\right)$ of $\mathcal{H} \setminus \{v\}$ of width k. As

$$N^{\mathcal{H}}(v) := \left\{u \in V(\mathcal{H}) \;\middle|\; \{u, v\} \in E(\mathcal{H})\right\}$$

is a clique in $\mathcal{H} \setminus \{v\}$, by Helly property, there is a B_t with $N^{\mathcal{H}}(v) \subseteq B_t$.

We add a new leaf t_0 adjacent to t and set $B_{t_0} := N^{\mathcal{H}}[v]$.

Graph Isomorphism Problems

Graph isomorphism

Definition

Let $\mathcal{G}$ and $\mathcal{H}$ be two graphs. A function $f : V(\mathcal{G}) \to V(\mathcal{H})$ is an isomorphism if

(GI1) f is a bijection;

(GI2) for every $u, v \in V(\mathcal{G})$ we have $\{u, v\} \in E(\mathcal{G})$ if and only if $\{f(u), f(v)\} \in E(\mathcal{H})$.

If such an f exists, then $\mathcal{G}$ and $\mathcal{H}$ are isomorphic.

Graph Isomorphism (GI) problem

GI	
Input:	Two graphs $\mathcal{G}$ and $\mathcal{H}$.
Problem:	Decides whether $\mathcal{G}$ and $\mathcal{H}$ are isomorphic.

Remark

1. GI *is in* NP.
2. GI *is not* NP*-complete, unless Polynomial Hierarchy collapses (which most people do not believe).*
3. *We don't know whether* GI *is* P*-hard.*
4. *Some people believe* GI *is in* P*, but we don't even have a quantum polynomial time algorithm.*

Graph isomorphism problems and treewidth

GI on bounded treewidth graphs

Theorem (Bodlaender, 1990)

Let $k \in \mathbb{N}$. Then there is a polynomial time algorithm which decides GI *on graphs $\mathcal{G}$ with* $\mathsf{tw}(\mathcal{G}) \leq k$.

I will present an algorithm deciding the problem

Input:	Two graphs $\mathcal{G}$ and $\mathcal{H}$ and a smooth tree decomposition of $\mathcal{G}$ of width k.
Problem:	Decides whether $\mathcal{G}$ and $\mathcal{H}$ are isomorphic.

in time

$$(|V(\mathcal{G})| + |V(\mathcal{H})|)^{O(k)}.$$

Isomorphism via connected components

Let $\mathscr{C}_{\mathcal{G}}$ be the set of connected components of $\mathcal{G}$ and $\mathscr{C}_{\mathcal{H}}$ the set of connected components of $\mathcal{H}$.

Then $\mathcal{G}$ and $\mathcal{H}$ are isomorphic if and only if there is a bijection $h : \mathscr{C}_{\mathcal{G}} \to \mathscr{C}_{\mathcal{H}}$ such that $\mathcal{G}[C]$ and $\mathcal{H}[h(C)]$ are isomorphic for every $C \in \mathscr{C}_{\mathcal{G}}$.

This is equivalent to that there is a perfect matching in the following bipartite graph.

1. The left part is $\mathscr{C}_{\mathcal{G}}$ and the right part $\mathscr{C}_{\mathcal{H}}$.
2. There is an edge between a $C \in \mathscr{C}_{\mathcal{G}}$ and a $C' \in \mathscr{C}_{\mathcal{H}}$ if $\mathcal{G}[C]$ and $\mathcal{H}[C']$ are isomorphic.

Isomorphism via separators

Let $S \subseteq V(\mathcal{G})$ and

$$\mathscr{C}_{\mathcal{G}\setminus S} := \big\{ C \;\big|\; C \text{ a connected component of } \mathcal{G} \setminus S \big\}.$$

Then $\mathcal{G}$ and $\mathcal{H}$ are isomorphic if and only if there is a set $S' \subseteq V(\mathcal{H})$, a function $h : \mathscr{C}_{\mathcal{G}\setminus S} \to \mathscr{C}_{\mathcal{H}\setminus S'}$ and functions $f_C : S \cup C \to S' \cup h(C)$ for all $C \in \mathscr{C}_{\mathcal{G}\setminus S}$ such that

1. $|S| = |S'|$;
2. h is a bijection;
3. f_C is an isomorphism between $\mathcal{G}[S \cup C]$ and $\mathcal{H}[S' \cup h(C)]$ for every $C \in \mathscr{C}_{\mathcal{G}\setminus S}$, and $f_C(S) = S'$;
4. $f_{C_1} \restriction S = f_{C_2} \restriction S$ for every $C_1, C_2 \in \mathscr{C}_{\mathcal{G}\setminus S}$.

The sets $\mathscr{C}_t$

Let $\big(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})}\big)$ be a smooth tree decomposition of width k for the graph $\mathcal{G}$. Again we choose an arbitrary root r in $\mathcal{T}$.

For every $t \in V(\mathcal{T})$ we define

$$\mathscr{C}_t := \big\{ C \;\big|\; C = \emptyset \text{ or } C \text{ a connected component of } \mathcal{G}_{\leq t} \setminus B_t \big\}.$$

Connected components via tree decompositions (1)

Lemma

Every nonempty $C \in \mathscr{C}_t$, i.e., a connect component in $\mathcal{G}_{\leq t} \setminus B_t$, is a connected component of $\mathcal{G} \setminus B_t$.

Proof.

Clearly there is a connected component C' in $\mathcal{G} \setminus B_t$ with $C \subseteq C'$.

Assume that $C' \setminus C \neq \emptyset$. Then there is an edge $\{u, v\} \in E(\mathcal{G})$ with $u \in V(\mathcal{G}_{\leq t}) \setminus B_t$ and $v \in V(\mathcal{G}) \setminus V(\mathcal{G}_{\leq t})$.

But then, $\{u, v\}$ is not contained in any bag of the tree decomposition. $\square$

Connected components via tree decompositions (2)

Lemma

Let t_1 be a child of t. Then for every nonempty $C_1 \in \mathscr{C}_{t_1}$ there is a unique $C \in \mathscr{C}_t$ with $C_1 \subseteq C$, and $C_1 \cap C' = \emptyset$ for all other $C' \in \mathscr{C}_t$.

Proof.

Let C_1 be a connected component of $\mathcal{G}_{\leq t_1} \setminus B_{t_1}$.

Observe that

$$\mathcal{G}_{\leq t_1} \setminus B_{t_1} \subseteq \mathcal{G}_{\leq t} \setminus B_t,$$

so C_1 is connected in $\mathcal{G}_{\leq t} \setminus B_t$, and the result follows. □

Connected components via tree decompositions (3)

Lemma

Let t be a node in $\mathcal{T}$ with children $t_1, \ldots, t_n$. And let $C \in \mathscr{C}_t$ be nonempty. Then, there is a unique $i \in [m]$ such that

$$C \subseteq \bigcup \mathscr{C}_{t_i} \cup \{v\} \qquad \text{where } \{v\} = B_{t_i} \setminus B_t.$$

Intuitively, C is shattered, i.e., broken into several smaller connected components, by the bag of exactly one child of t.

Connected components via tree decompositions (4)

Lemma

Let t_1, t_2 be two distinct children of t. For every $i \in [2]$, let v_i be the vertex in $\mathcal{G}$ with $\{v_i\} = B_{t_i} \setminus B_t$; and $C_i \in \mathscr{C}_{t_i}$. Then for every $C \in \mathscr{C}_t$

$$\big(C_1 \cup \{v_1\}\big) \cap C = \emptyset \quad \textit{or} \quad \big(C_2 \cup \{v_2\}\big) \cap C = \emptyset.$$

Connected components via tree decompositions (5)

Proof.

It is easy to see

$$\big(C_1 \cup \{v_1\}\big) \cap \big(C_2 \cup \{v_2\}\big) = \emptyset.$$

Assume $(C_1 \cup \{v_1\}) \cap C \neq \emptyset \neq (C_2 \cup \{v_2\}) \cap C$. Then there is a path P from $C_1 \cup \{v_1\}$ to $C_2 \cup \{v_2\}$ in C. Without loss of generality, we can assume that all vertices on P are in

$$\big(C_1 \cup \{v_1\}\big) \cup \big(C_2 \cup \{v_2\}\big).$$

Then there is an edge between $C_1 \cup \{v_1\}$ and $C_2 \cup \{v_2\}$, which cannot be contained in any bag of the tree decomposition. □

Decompose $\mathcal{H}$

Let $\mathcal{H}$ be a second graph for which we want to decide whether $\mathcal{G}$ and $\mathcal{H}$ are isomorphic.

We define (the set of pairs of separators and connected components)

$$\mathscr{SC}(\mathcal{H}) := \big\{(S, C) \mid S \subseteq V(\mathcal{H}) \text{ with } |S| = k+1 \text{ and } (C = \emptyset \text{ or } C \text{ a connected component of } \mathcal{H} \setminus S)\big\}$$

Partial isomorphisms

Definition

Let $t \in V(\mathcal{T})$, $S_1 := B_t$, and $C_1 \in \mathscr{C}_t$. Moreover, let $(S_2, C_2) \in \mathscr{SC}(\mathcal{H})$. We say (S_1, C_1) and (S_2, C_2) are f-isomorphic for a function $f : S_1 \to S_2$, denoted by $(S_1, C_1) \equiv^f (S_2, C_2)$, if there is a function $F : S_1 \cup C_1 \to S_2 \cup C_2$ such that

(F1) $F \restriction S_1 = f$;

(F2) for every $u, v \in S_1 \cup C_1$ we have $\{u, v\} \in E(\mathcal{G})$ if and only if $\{F(u), F(v)\} \in E(\mathcal{H})$.

That is, F is an isomorphism between $\mathcal{G}[S_1 \cup C_1]$ and $\mathcal{H}[S_2 \cup C_2]$ extending f.

Extending partial isomorphisms

Our goal is to compute for each $t \in V(\mathcal{T})$ the set

$$\mathscr{F}_t := \Big\{ (f, B_t, C_1, S_2, C_2) \;\Big|\; (B_t, C_1) \equiv^f (S_2, C_2) \text{ where } C_1 \in \mathscr{C}_t \text{ and } (S_2, C_2) \in \mathscr{SC}(\mathcal{H}) \Big\}.$$

using dynamic programming.

Leaves

Let t be a leaf of $\mathcal{T}$.

Then $\mathscr{C}_t = \{\emptyset\}$. Hence,

$$\mathscr{F}_t := \Big\{(f, B_t, \emptyset, S_2, \emptyset) \;\Big|\; (B_t, \emptyset) \equiv^f (S, \emptyset) \text{ where } S_2 \subseteq V(\mathcal{H}) \text{ with } |S_2| = k+1\Big\}.$$

This can be computed in time

$$(k+1)! \cdot |V(\mathcal{H})|^{O(k)}.$$

Non-leaves (1)

Let t be a node in $\mathcal{T}$ with children $t_1, \ldots, t_m$ for some $m \geq 1$.

Now let $C_1 \in \mathscr{C}_t$ be nonempty. By Lemma 3, there is a unique $i \in [m]$ such that

$$C_1 \subseteq \bigcup \mathscr{C}_{t_i} \cup \{v\} \qquad \text{where } \{v\} = B_{t_i} \setminus B_t.$$

For every $(S_2, C_2) \in \mathscr{SC}(\mathcal{H})$ and every $f : B_t \to S_2$ with $(B_t, \emptyset) \equiv^f (S_2, \emptyset)$ we want to check whether $(B_t, C_1) \equiv^f (S_2, C_2)$.

Non-leaves (2)

$(B_t, C_1) \equiv^f (S_2, C_2)$ if and only if some for $v' \in V(\mathcal{H}) \setminus S_2$, $u \in S_2$, and

- $S_2' := S_2 \cup \{v'\} \setminus \{u\}$,
- $\mathscr{C}_1^* := \left\{ C^* \;\middle|\; C^* \text{ a connected component of } \mathcal{G} \setminus B_{t_i} \text{ with } C^* \subseteq C_1 \right\}$ and
  $\mathscr{C}_2^* := \left\{ C^* \;\middle|\; C^* \text{ a connected component of } \mathcal{H} \setminus S_2' \text{ with } C^* \subseteq C_2 \right\}$,
- $f' : B_{t_i} \to S_2'$ defined by

$$f'(w) = \begin{cases} v' & \text{if } w = v \\ f(w) & \text{otherwise,} \end{cases}$$

we have

(N1) every connected component of $\mathcal{H} \setminus S_2'$ is either contained in or disjoint with C_2;

(N2) $C_2 \subseteq \bigcup \mathscr{C}_2^* \cup \{v'\}$;

(N3) there is a bijection $h : \mathscr{C}_1^* \to \mathscr{C}_2^*$ such that for every $C^* \in \mathscr{C}_1^*$

$$\left(B_{t_i}, C^*\right) \equiv^{f'} \left(S_2', h(C^*)\right).$$

Non-leaves (3)

(N1) and (N2) can be checked in polynomial time.

To verify (N3) we create a bipartite graph $\mathcal{B}$:

1. the left part is $\mathscr{C}_1^*$ and the right part $\mathscr{C}_2^*$;
2. there is an edge between $C_1^* \in \mathscr{C}_1^*$ and $C_2^* \in \mathscr{C}_2^*$ if $(B_{t_i}, C_1^*) \equiv^{f'} (S_2', C_2^*)$.

Then (N3) holds if and only if there is a perfect matching in $\mathcal{B}$, which can be decided in polynomial time.

The final step

$\mathcal{G}$ and $\mathcal{H}$ are isomorphic if and only if for some $S_2 \subseteq V(\mathcal{H})$ with $|S_2| = k+1$ and $f : B_r \to S_2$ there is a perfect matching in the following bipartite graph.

1. The left part is $\mathscr{C}_r$ and the right part $\mathscr{C}^* := \left\{ C_2 \;\middle|\; (S_2, C_2) \in \mathscr{SC}(\mathcal{H}) \right\}$.
2. There is an edge between a $C_1 \in \mathscr{C}_r$ and a $C_2 \in \mathscr{C}^*$ if $(B_r, C_1) \equiv^f (S_2, C^*)$.