The Isomorphism Problem for *k*-Trees is Complete for Logspace

Johannes Köbler   Sebastian Kuhnert

Nový Smokovec, August 24th, 2009

Outline

Graph isomorphisms

Definition

Let G, H be graphs

- A bijection $\varphi : V_G \to V_H$ is an *isomorphism*, if $\forall u, v \in V_G : \{u, v\} \in E(G) \Leftrightarrow \{\varphi(u), \varphi(v)\} \in E(H)$
- If such a φ exists, G and H are *isomorphic* ($G \cong H$).

Graph isomorphisms

Definition

Let G, H be graphs

- A bijection $\varphi\colon V_G \to V_H$ is an *isomorphism*, if $\forall u, v \in V_G : \{u, v\} \in E(G) \Leftrightarrow \{\varphi(u), \varphi(v)\} \in E(H)$
- If such a φ exists, G and H are *isomorphic* ($G \cong H$).
- For a graph G (w. l. o. g. $V_G = \{1, \ldots, n\}$) and $\varphi \in S_n$ let $\varphi(G)$ be the graph given by

$$V_{\varphi(G)} := \{\varphi(v) \mid v \in V_G\}$$
$$E_{\varphi(G)} := \Big\{\{\varphi(u), \varphi(v)\} \,\Big|\, \{u, v\} \in E_G\Big\}$$

Graph isomorphisms

Definition

Let G, H be graphs (colored with c_G, c_H)

- A bijection $\varphi\colon V_G \to V_H$ is an *isomorphism*, if $\forall u, v \in V_G : \{u, v\} \in E(G) \Leftrightarrow \{\varphi(u), \varphi(v)\} \in E(H)$ (and $\forall v \in V_G : c_G(v) = c_H(\varphi(v))$).
- If such a φ exists, G and H are *isomorphic* ($G \cong H$).
- For a graph G (w. l. o. g. $V_G = \{1, \ldots, n\}$) and $\varphi \in S_n$ let $\varphi(G)$ be the graph given by

$$V_{\varphi(G)} := \{\varphi(v) \mid v \in V_G\}$$

$$E_{\varphi(G)} := \Big\{\{\varphi(u), \varphi(v)\} \,\Big|\, \{u, v\} \in E_G\Big\}$$

$$c_{\varphi(G)}(v) := c_G(\varphi^{-1}(v))$$

Invariants and canonical labelings

Definition

Let $\mathcal{G}$ be a graph class and f a function defined on $\mathcal{G}$.

- f is an *invariant* for $\mathcal{G}$, if

$$\forall G, H \in \mathcal{G} : G \cong H \Rightarrow f(G) = f(H) .$$

Invariants and canonical labelings

Definition

Let $\mathcal{G}$ be a graph class and f a function defined on $\mathcal{G}$.

- f is an *invariant* for $\mathcal{G}$, if

$$\forall G, H \in \mathcal{G} : G \cong H \Rightarrow f(G) = f(H)\,.$$

- f is a *complete invariant* for $\mathcal{G}$, if

$$\forall G, H \in \mathcal{G} : G \cong H \Leftrightarrow f(G) = f(H).$$

Invariants and canonical labelings

Definition

Let $\mathcal{G}$ be a graph class and f a function defined on $\mathcal{G}$.

- f is an *invariant* for $\mathcal{G}$, if
$$\forall G, H \in \mathcal{G} : G \cong H \Rightarrow f(G) = f(H)\,.$$
- f is a *complete invariant* for $\mathcal{G}$, if
$$\forall G, H \in \mathcal{G} : G \cong H \Leftrightarrow f(G) = f(H).$$
- f is a *canonization* for $\mathcal{G}$,
  if f is a complete invariant with $\forall G \in \mathcal{G} : f(G) \cong G$.
  $f(G)$ is called *canonical form* of G.

Invariants and canonical labelings

Definition

Let $\mathcal{G}$ be a graph class and f a function defined on $\mathcal{G}$.

- f is an *invariant* for $\mathcal{G}$, if
$$\forall G, H \in \mathcal{G} : G \cong H \Rightarrow f(G) = f(H)\,.$$
- f is a *complete invariant* for $\mathcal{G}$, if
$$\forall G, H \in \mathcal{G} : G \cong H \Leftrightarrow f(G) = f(H).$$
- f is a *canonization* for $\mathcal{G}$,
if f is a complete invariant with $\forall G \in \mathcal{G} : f(G) \cong G$.
$f(G)$ is called *canonical form* of G.

Assume w.l.o.g. that $V_G = \{1, \ldots, n\}$.

- A function $\psi : \mathcal{G} \to S_n$ that maps $G \mapsto \psi_G$
is a *canonical labeling* for $\mathcal{G}$,
if $G \mapsto \psi_G(G)$ is a canonization for $\mathcal{G}$.

k-trees

Definition

The class of *trees* can be defined inductively:

- A single vertex is a tree.
- If G is a tree, the following construction yields a tree G':
  - choose a vertex u in G and
  - connect u with a new vertex v:

    $V_{G'} := V_G \cup \{v\}$

    $E_{G'} := E_G \cup \{\{u, v\}\}$

k-trees

Definition

The class of *trees* can be defined inductively:

- A 1-Clique is a tree.
- If G is a tree, the following construction yields a tree G':
  - choose a 1-Clique in G and
  - connect C with a new vertex v:

$$V_{G'} := V_G \cup \{v\}$$
$$E_{G'} := E_G \cup \{\{c, v\} \mid c \in C\}$$

k-trees

Definition

The class of *k-trees* can be defined inductively:

- A k-Clique is a k-tree.
- If G is a k-tree, the following construction yields a k-tree G':
  - choose a k-Clique C in G and
  - connect C with a new vertex v:

    $V_{G'} := V_G \cup \{v\}$

    $E_{G'} := E_G \cup \{\{c, v\} \mid c \in C\}$

k-trees

Definition

The class of *k-trees* can be defined inductively:

- A k-Clique is a k-tree.
- If G is a k-tree, the following construction yields a k-tree G':
  - choose a k-Clique C in G and
  - connect C with a new vertex v:

    $$V_{G'} := V_G \cup \{v\}$$
    $$E_{G'} := E_G \cup \left\{ \{c, v\} \,\middle|\, c \in C \right\}$$

Partial k-trees are subgraphs of k-trees.

k-trees

Definition

The class of *k-trees* can be defined inductively:

- A k-Clique is a k-tree.
- If G is a k-tree, the following construction yields a k-tree G':
  - choose a k-Clique C in G and
  - connect C with a new vertex v:

    $$V_{G'} := V_G \cup \{v\}$$
    $$E_{G'} := E_G \cup \big\{\{c, v\} \,\big|\, c \in C\big\}$$

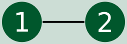
Partial k-trees are subgraphs of k-trees.

Example

- Partial 1-trees are forests.
- G is a partial k-tree iff its tree width is $\leq k$.

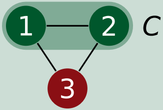
A 2-tree

Example



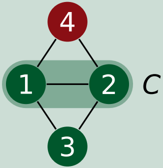
A 2-tree

Example



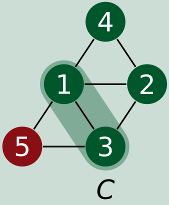
A 2-tree

Example



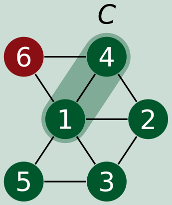
A 2-tree

Example



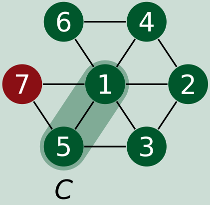
A 2-tree

Example



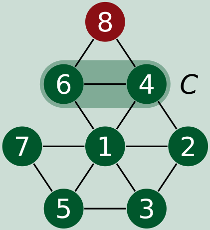
A 2-tree

Example



A 2-tree

Example



Known Results

Tree isomorphism and canonization

- In time $\mathcal{O}(n)$ [Aho, Hopcroft, Ullman 74]
- In NC [Miller, Reif 91]
- In L [Lindell 92]
- Complete for L [Jenner et al. 03]

Known Results

Tree isomorphism and canonization

- In time $\mathcal{O}(n)$ [Aho, Hopcroft, Ullman 74]
- In NC [Miller, Reif 91]
- In L [Lindell 92]
- Complete for L [Jenner et al. 03]

Partial k-tree isomorphism

- In time $O(n^{k+4.5})$ [Bodlaender 90]
- For $k = 2$ and 3 in time $\mathcal{O}(n \log n)$ [Arnborg, Proskurowski 92]
- In TC^1 [Grohe, Verbitsky 06]
- Canonizing in TC^2 [Köbler, Verbitsky 08]
- For $k = 2$ complete for L [Arvind, Das, Köbler 08]

Known Results

Tree isomorphism and canonization

- In time $\mathcal{O}(n)$ [Aho, Hopcroft, Ullman 74]
- In NC [Miller, Reif 91]
- In L [Lindell 92]
- Complete for L [Jenner et al. 03]

Known Results

Tree isomorphism and canonization

- In time $\mathcal{O}(n)$ [Aho, Hopcroft, Ullman 74]
- In NC [Miller, Reif 91]
- In L [Lindell 92]
- Complete for L [Jenner et al. 03]

k-tree isomorphism

- In AC^2 [Greco, Sekharan, Sridhar 02]
- In StUL, hard for L [Arvind, Das, Köbler 07]
  - (For k-paths: Complete for L)

Known Results

Tree isomorphism and canonization

- In time $\mathcal{O}(n)$ [Aho, Hopcroft, Ullman 74]
- In NC [Miller, Reif 91]
- In L [Lindell 92]
- Complete for L [Jenner et al. 03]

k-tree isomorphism

- In AC^2 [Greco, Sekharan, Sridhar 02]
- In StUL, hard for L [Arvind, Das, Köbler 07]
  - (For k-paths: Complete for L)
- Complete for L

Outline

1. Introduction
   - Isomorphisms and Canonization
   - k-trees
   - Known Results

2. Canonizing k-trees
   - Tree representation
   - The FL algorithm

Outline of our algorithm

k-tree G

$\downarrow$

canon of G

Outline of our algorithm

$$\begin{tikzcd}
k\text{-tree } G \arrow[r] & \text{tree representation } T(G) \arrow[d, "\text{variant of Lindell's algorithm}"] \\
\text{canon of } G & \text{canon of } T(G) \arrow[l]
\end{tikzcd}$$

Outline of our algorithm

$$\begin{tikzcd}
k\text{-tree } G \arrow[r] & \text{tree representation } T(G) \arrow[d, "\text{variant of Lindell's algorithm}"] \\
\text{canon of } G & \text{canon of } T(G) \arrow[l]
\end{tikzcd}$$

Requirements for this approach:

- Isomorphic k-trees must have isomorphic tree representations
- $T(G)$ must contain enough information to reconstruct an isomorphic copy of G
- Both construction and reconstruction must be possible in logspace

Tree representation of k-trees

Definition

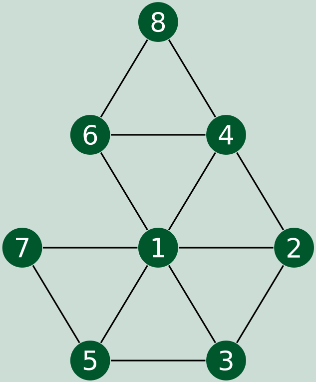
Let G be a k-tree.
The *tree representation* $T(G)$ is defined by

$$V_{T(G)} := \left\{ M \subseteq V_G \,\middle|\, M \text{ is a } k\text{- or } (k+1)\text{-clique in } G \right\}$$

$$E_{T(G)} := \left\{ \{M_1, M_2\} \subseteq V_{T(G)} \,\middle|\, M_1 \subsetneq M_2 \right\}$$

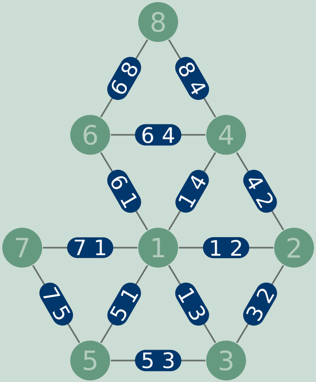
Tree representation of a 2-tree

Example



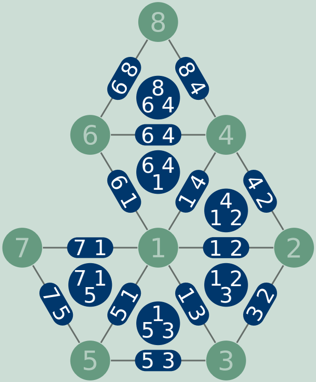
Tree representation of a 2-tree

Example



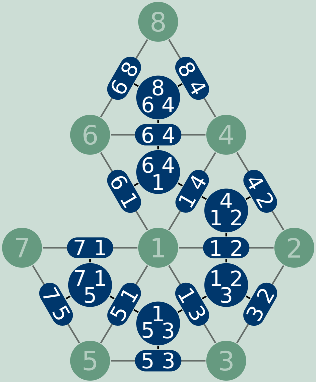
Tree representation of a 2-tree

Example



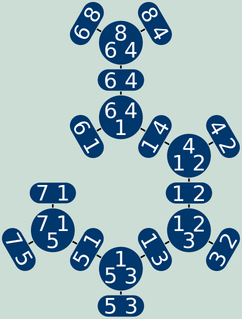
Tree representation of a 2-tree

Example



Tree representation of a 2-tree

Example



Tree representation of a 2-tree

Example

G:

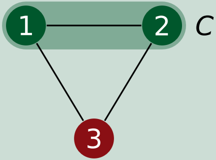
$T(G)$:

- $T(G)$ is a tree.

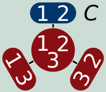
Tree representation of a 2-tree

Example

G:



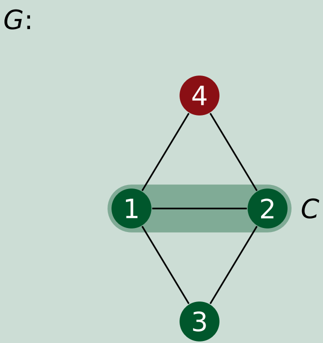
$T(G)$:

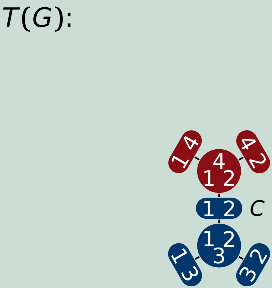


- $T(G)$ is a tree.

Tree representation of a 2-tree

Example



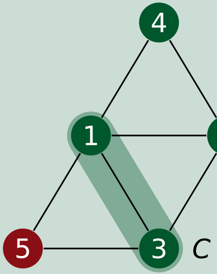


- $T(G)$ is a tree.

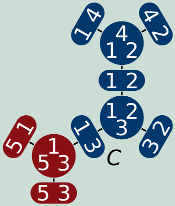
Tree representation of a 2-tree

Example

G:



$T(G)$:

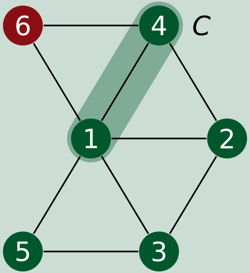


- $T(G)$ is a tree.

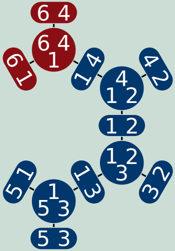
Tree representation of a 2-tree

Example

G:



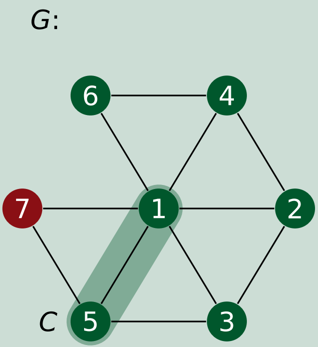
$T(G)$:

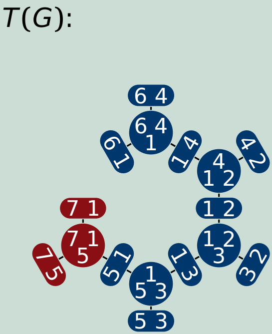


- $T(G)$ is a tree.

Tree representation of a 2-tree

Example

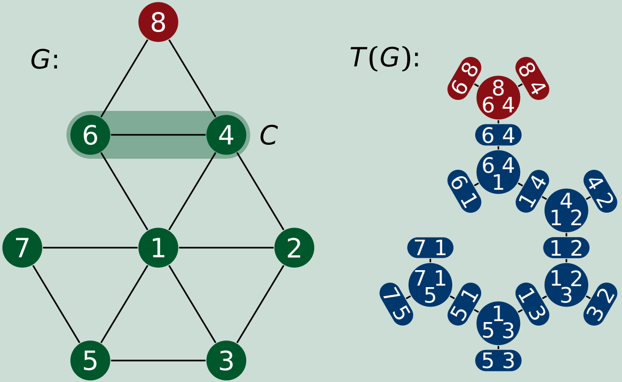




- $T(G)$ is a tree.

Tree representation of a 2-tree

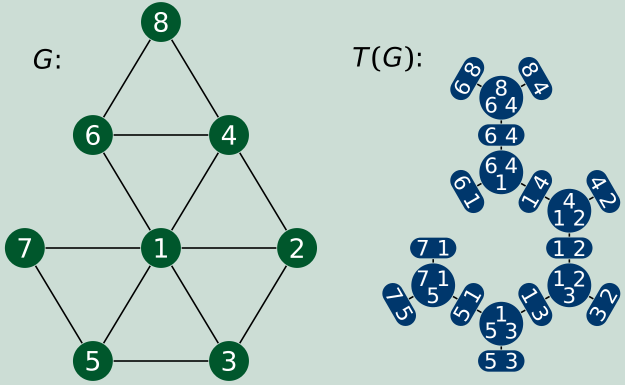
Example



- $T(G)$ is a tree.

Tree representation of a 2-tree

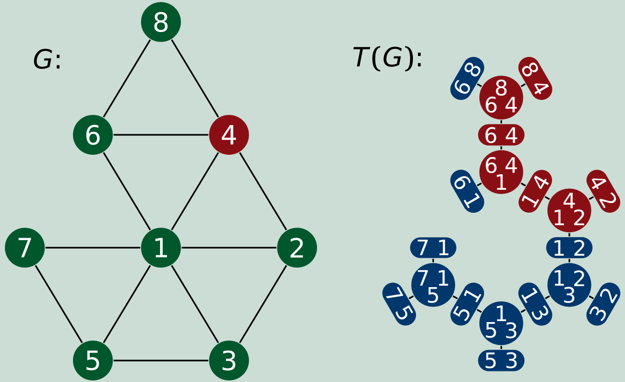
Example



- $T(G)$ is a tree.

Tree representation of a 2-tree

Example

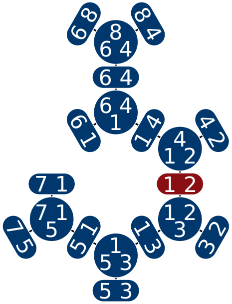


- $T(G)$ is a tree.
- For any $v \in V_G$, the nodes of $T(G)$ that contain v form a subtree of $T(G)$.

The kernel of a k-tree

Fact

For any k-tree G, the *center* *of $T(G)$ is a single node.*



Johannes Köbler, Sebastian Kuhnert

The kernel of a k-tree

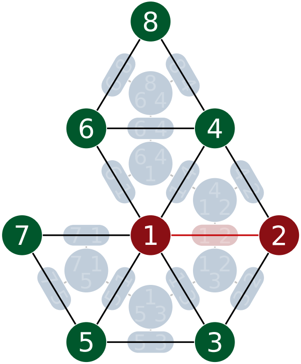
Fact

For any k-tree G, the center of $T(G)$ is a single node.

Definition

For a k-tree G, the *kernel* $\ker(G)$ is the clique corresponding to the center node of $T(G)$.

- The kernel of a k-tree was introduced before [Greco, Sekharan, Sridhar 02]



The kernel of a k-tree

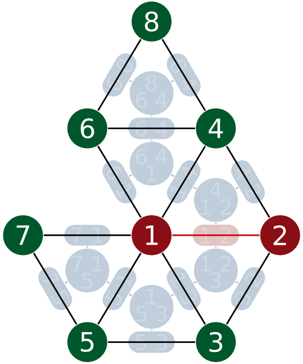
Fact

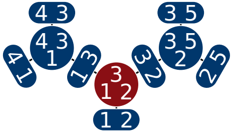
For any k-tree G, the center of $T(G)$ is a single node.

Definition

For a k-tree G, the *kernel* $\ker(G)$ is the clique corresponding to the center node of $T(G)$.

- The kernel of a k-tree was introduced before [Greco, Sekharan, Sridhar 02]
- Note that $\ker(G)$ is either a k- or a $(k+1)$-clique.





The kernel of a k-tree

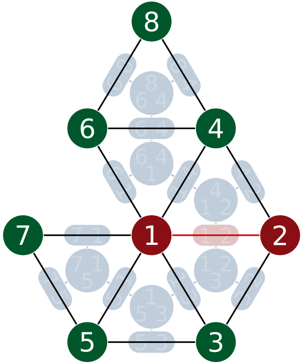
Fact

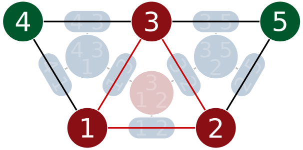
For any k-tree G, the center of $T(G)$ is a single node.

Definition

For a k-tree G, the *kernel* $\ker(G)$ is the clique corresponding to the center node of $T(G)$.

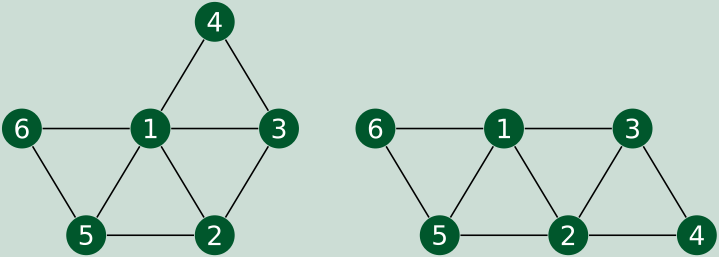
- The kernel of a k-tree was introduced before [Greco, Sekharan, Sridhar 02]
- Note that $\ker(G)$ is either a k- or a $(k+1)$-clique.
- We define $k' := \|\ker(G)\|$.





The need for colors

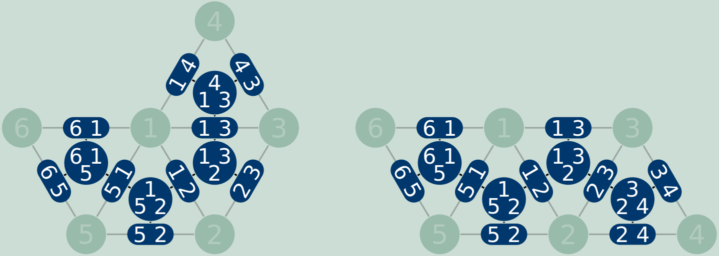
Example



- These 2 graphs are non-isomorphic . . .

The need for colors

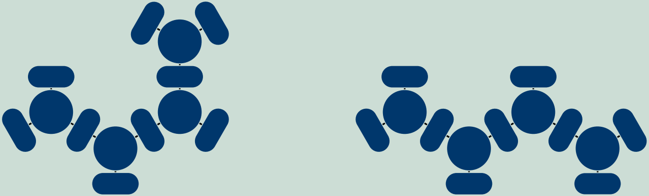
Example



- These 2 graphs are non-isomorphic ...

The need for colors

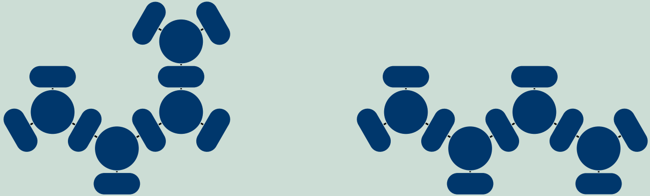
Example



- These 2 graphs are non-isomorphic . . .
  . . . but their tree representations are

The need for colors

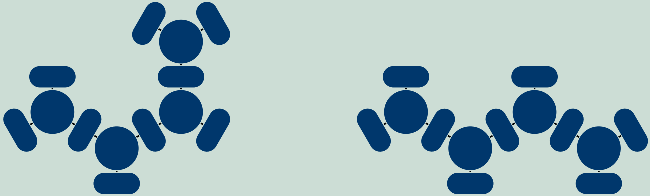
Example



- These 2 graphs are non-isomorphic . . .
  . . . but their tree representations are
- Thus it is impossible to reconstruct an isomorphic copy of G from an isomorphic copy of $T(G)$

The need for colors

Example



- These 2 graphs are non-isomorphic . . .
  . . . but their tree representations are
- Thus it is impossible to reconstruct an isomorphic copy of G from an isomorphic copy of $T(G)$
- Solution: Color the nodes of $T(G)$
  to fully encode the structure of G

Coloring the tree representation

Definition

Let G be a k-tree with $V_G = \{1, \ldots, n\}$ and $K := \ker(G) = \{1, \ldots, k'\}$ and let $v \in V_G$.

- The *level* of v is

$$l(v) := \min \left\{ d_{T(G)}(K, M) \,\middle|\, M \in V_{T(G)}, v \in M \right\}$$

Now let $\pi \in S_{k'}$ be a permutation on K.

- The *color* of v is

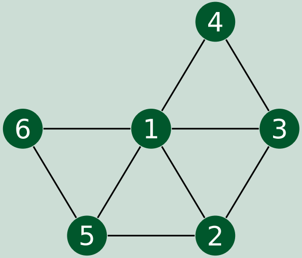
$$c_\pi(v) := \begin{cases} \pi(v) & \text{if } v \in \ker(G) \\ l(v) + k' & \text{otherwise} \end{cases}$$

- The *colored tree representation $T(G, \pi)$* of G is $T(G)$ with K as root and each $M \in V_{T(G)}$ colored by

$$c_\pi(M) := \left\{ c_\pi(v) \,\middle|\, v \in M \right\}$$

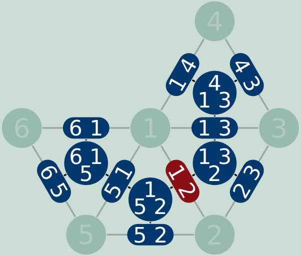
Colored tree representations

Example



Colored tree representations

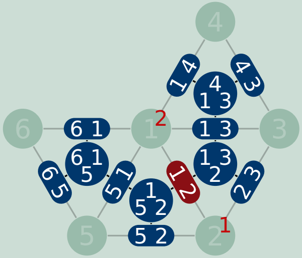
Example



- $\ker(G) = \{1, 2\}$, $k' = 2$
- $l(1) = l(2) = 0$
  $l(3) = l(5) = 1$
  $l(4) = l(6) = 3$

Colored tree representations

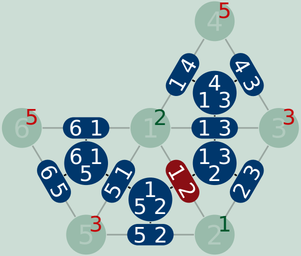
Example



- $\ker(G) = \{1, 2\}$, $k' = 2$
- $l(1) = l(2) = 0$
  $l(3) = l(5) = 1$
  $l(4) = l(6) = 3$
- Use $\pi = (12)$
- For $v \in \ker(G)$:
  $c_\pi(v) = \pi(v)$
  $c_\pi(1) = 2$
  $c_\pi(2) = 1$

Colored tree representations

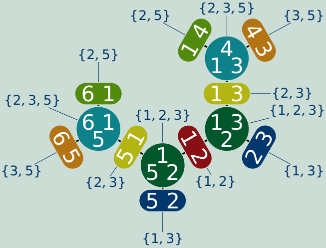
Example



- $\ker(G) = \{1, 2\}$, $k' = 2$
- $l(1) = l(2) = 0$
  $l(3) = l(5) = 1$
  $l(4) = l(6) = 3$
- Use $\pi = (12)$
- For $v \in \ker(G)$:
  $c_\pi(v) = \pi(v)$
  $c_\pi(1) = 2$
  $c_\pi(2) = 1$
- For $v \notin \ker(G)$:
  $c_\pi(v) = k' + l(v)$
  $c_\pi(3) = c_\pi(5) = 3$
  $c_\pi(4) = c_\pi(6) = 5$

Colored tree representations

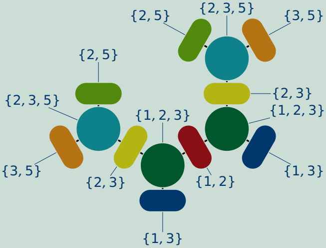
Example



- $\ker(G) = \{1, 2\}$, $k' = 2$
- $l(1) = l(2) = 0$
  $l(3) = l(5) = 1$
  $l(4) = l(6) = 3$
- Use $\pi = (12)$
- For $v \in \ker(G)$:
  $c_\pi(v) = \pi(v)$
  $c_\pi(1) = 2$
  $c_\pi(2) = 1$
- For $v \notin \ker(G)$:
  $c_\pi(v) = k' + l(v)$
  $c_\pi(3) = c_\pi(5) = 3$
  $c_\pi(4) = c_\pi(6) = 5$

Colored tree representations

Example

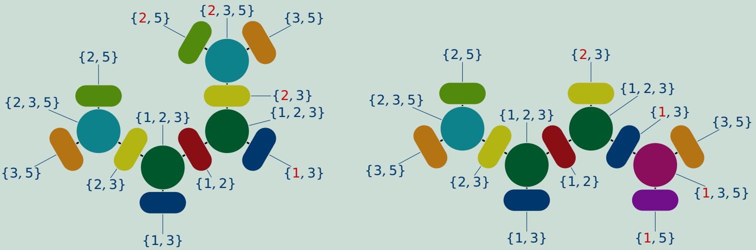


- $\ker(G) = \{1, 2\}$, $k' = 2$
- $l(1) = l(2) = 0$
  $l(3) = l(5) = 1$
  $l(4) = l(6) = 3$
- Use $\pi = (12)$
- For $v \in \ker(G)$:
  $c_\pi(v) = \pi(v)$
  $c_\pi(1) = 2$
  $c_\pi(2) = 1$
- For $v \notin \ker(G)$:
  $c_\pi(v) = k' + l(v)$
  $c_\pi(3) = c_\pi(5) = 3$
  $c_\pi(4) = c_\pi(6) = 5$

- From an isomorphic copy of $T(G, \pi)$ an isomorphic copy of G can be reconstructed

Colored tree representations

Example



- From an isomorphic copy of $T(G, \pi)$ an isomorphic copy of G can be reconstructed

Facts on $T(G, \pi)$

Lemma

For a k-tree G and a permutation π on $\ker(G)$, $T(G, \pi)$ can be computed in FL.

Facts on $T(G, \pi)$

Lemma

For a k-tree G and a permutation π on $\ker(G)$, $T(G, \pi)$ can be computed in FL.

Lemma

Let G, H be k-trees such that $G \cong H$ via φ, $V(G) = V(H) = \{1, \ldots, n\}$ and $\ker(G) = \ker(H) = K$. Then also $T(G, \pi_1) \cong T(H, \pi_2)$ via φ, provided that $\pi_1(u) = \pi_2(\varphi(u))$ for all $u \in K$.

Facts on $T(G, \pi)$

Lemma

For a k-tree G and a permutation π on $\ker(G)$, $T(G, \pi)$ can be computed in FL.

Lemma

Let G, H be k-trees such that $G \cong H$ via φ, $V(G) = V(H) = \{1, \ldots, n\}$ and $\ker(G) = \ker(H) = K$. Then also $T(G, \pi_1) \cong T(H, \pi_2)$ via φ, provided that $\pi_1(u) = \pi_2(\varphi(u))$ for all $u \in K$.

Lemma

Let G be a k-tree and let π be a permutation on the kernel K of G.

- *From any colored tree $T \cong T(G, \pi)$, an isomorphic copy G' of G can be computed in FL.*
- *It is possible to compute an isomorphism between G and G' from any given isomorphism between $T(G, \pi)$ and T in FL.*

Canonical labeling for k-trees

Theorem

Given a k-tree G with $V_G = \{1, \ldots, n\}$ and kernel $K = \{1, \ldots, k'\}$, a canonical labeling $\psi_G \in S_n$ can be computed in FL.

The algorithm

```
foreach π ∈ S_k′ do
  compute T(G, π)
  compute a canonical labeling φ_T(G,π) of
      T(G, π)
choose π_1 ∈ S_k′ such that φ_T(G,π_1)(T(G, π_1)) is
    minimal
reconstruct a labeling ψ_G of G from φ_T(G,π)
return ψ_G as canonical labeling
```

Our results

Corollary

For any fixed k,

- *k-tree canonization is in FL and*
- *k-tree isomorphism is in L.*

Corollary

For any fixed k,

- *computing a generating set of $\mathrm{Aut}(G)$ for a given k-tree G is in FL,*
- *computing a canonical labeling coset for a given k-tree is in FL, and*
- *k-tree automorphism is L-complete.*

Summary and open problems

- Canonical labeling for k-trees can be reduced in logspace to canonical labeling for trees
- k-tree isomorphism is L-complete
- For k-trees, a canonical labeling coset can be computed in logspace

Summary and open problems

- Canonical labeling for k-trees can be reduced in logspace to canonical labeling for trees
- k-tree isomorphism is L-complete
- For k-trees, a canonical labeling coset can be computed in logspace

- What about *partial* k-tree isomorphism?
  - Can the upper bound of TC^1 be improved? (e. g. to NL, $\oplus\mathrm{L}$, L)
  - Is it hard for NL or $\oplus\mathrm{L}$?
- Can our approach be generalized?
  - To hookup classes that are not isomorphism complete, c. f. [Klawe, Corneil, Proskurowski 82]
  - To chordal graphs with small s-components, c. f. [Toda 06]

Thank You!

Literature I

Aho, A.V., J.E. Hopcroft, J.D. Ullman (1974).
The design and analysis of computer algorithms.
Addison-Wesley.

Arnborg, Stefan, Andrzej Proskurowski (1992).
'Canonical representations of partial 2- and 3-trees'.
In: *BIT Num. Math.* 32.2 (June 1992), pp. 197–214.

Arvind, Vikraman, Bireswar Das, Johannes Köbler (2007).
'The Space Complexity of k-Tree Isomorphism'.
In: *Algorithms and Computation. Proceedings of 18th ISAAC*. Springer, pp. 822–833.

— (2008). 'A Logspace Algorithm for Partial 2-Tree Canonization'.
In: *Proceedings of the 3rd International Computer Science Symposium in Russia (CSR)*. Springer, pp. 40–51.

Literature II

Bodlaender, Hans L. (1990).
'Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees'.
In: *J. Algor.* 11.4 (Dec. 1990), pp. 631–643.

Greco, J. G. Del, C. N. Sekharan, R. Sridhar (2002).
'Fast Parallel Reordering and Isomorphism Testing of k-Trees'. In: *Algorithmica* 32.1, pp. 61–72.

Grohe, Martin, Oleg Verbitsky (2006). 'Testing Graph Isomorphism in Parallel by Playing a Game'.
In: *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Proceedings, Part I*. Springer, pp. 3–14.

Jenner, B. et al. (2003).
'Completeness Results for Graph Isomorphism'.
In: *J. Comput. Syst. Sci.* 66, pp. 549–566.

Literature III

- Klawe, Maria M., Derek G. Corneil, Andrzej Proskurowski (1982). 'Isomorphism Testing in Hookup Classes'. In: *J. Algebraic Discrete Meth.* 3.2 (June 1982), pp. 260–274.
- Köbler, Johannes, Oleg Verbitsky (2008). 'From Invariants to Canonization in Parallel'. In: *Proceedings of the 3rd International Computer Science Symposium in Russia (CSR)*. Springer, pp. 216–227.
- Lindell, Steven (1992). 'A logspace algorithm for tree canonization. Extended abstract'. In: *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*. New York: ACM, pp. 400–404.
- Miller, G., J. Reif (1991). 'Parallel tree contraction part 2: further applications'. In: *SIAM J. Comput.* 20, pp. 1128–1147.

Literature IV

Toda, Seinosuke (2006). ‘Computing Automorphism Groups of Chordal Graphs Whose Simplicial Components Are of Small Size’. In: *IEICE Trans. Inform. Syst.* E89-D.8, pp. 2388–2401.