Lexicographic Cotree Factorization

Contents

1 Theoretical Concept

1.1 Trees

Trees are graphs (V, E) which have to be connected, acyclic and undirected. Rooted trees are just trees with one vertice specified as root. We may consider a inductive definition, detached from pure graph theory:

Definition 1.1. Let $\mathbb{T}_M$ be the set of all trees with labels in M:

$T = (r, \mathcal{C}) \in \mathbb{T}_M$, with $\mathcal{C}$ a multiset, if:

- T is called leaf $\iff \mathcal{C} = \varnothing$, $r \in M$
- $\mathcal{C} \subseteq \mathbb{T}_M$, $|\mathcal{C}| \in \mathbb{N}$, $r \in M$

1.1.1 Definitions on Trees

Let $T = (r, \mathcal{C}) \in \mathbb{T}_M$ be a Tree.

Definition 1.2. Nodes of T, named $\mathcal{N}(T) \subseteq \mathbb{T}_M$ are inductively defined by:

- Every child $C = (r_C, \mathcal{C}_C) \in \mathcal{C}$ is called node of T
- All nodes of the children $\mathcal{N}(C)$ are nodes of T as well

$$\implies \mathcal{N}(T) = \{T\} \cup \left(\bigcup_{C \in \mathcal{C}} \mathcal{N}(C) \right)$$

Nodes will usually denote the corresponding graph node, but with this notation the child nodes are acquired simultaneously.
The experienced reader will verify, that this matches the later defined complete subtrees. Which should be synonymous from here on.

Definition 1.3. Children of Node N of T: For $N = (r_N, \mathcal{C}_N)$

- Children of N as defined the set $\mathcal{C}_N$

Definition 1.4. Parent of Node N of T:

- Parent of N, denoted by $\mathcal{P}(N) = P \in \mathcal{N}(T)$ with $N \in \mathcal{C}_P$.
  While $\mathcal{P}(T) = \mathit{undef}$

Definition 1.5. Leafes of T:

- $\mathcal{L}(T) = \mathcal{N}(T) \cap \{(r, \varnothing) \mid r \in M\}$ are called leafs as above.

1.1.2 Level

The classical definition for a nodes level is by the distance of the node to the root of the tree. With the above definition we get: Let $T \in \mathbb{T}_M$ be a tree, N one of its nodes.

Definition 1.6 (Level)**.** The level of N will be:

$$l_T(N) = n \in \mathbb{N} : \qquad \text{with } \mathcal{P}^n(N) = \underbrace{\mathcal{P}(\mathcal{P}(\cdots \mathcal{P}}_{n \text{ times}}(N))) = T$$

1.1.3 Depth

The depth of some node N of T is like turning the definition of level upside down. With that the depth will be referencing to the shortest path to a leaf of the corresponding complete subtree.

Definition 1.7 (Depth)**.** We will define the depth of node $N = (r_N, \mathcal{C}_N)$ by:

$$d(N) = max\left(\{d(C) + 1 \mid C \in \mathcal{C}_N\} \cup \{0\}\right)$$

From this definition it follows that: $d(N) = 0 \Leftrightarrow N$ is leaf.

The depth said to be unambiguous if

$$d(N) = d(C) + 1 \qquad \forall C \in \mathcal{C}_N$$

and the depth of every child of N is unambiguous.

Definition 1.8 (Subtree)**.** $S = (r_S, \mathcal{C}_S)$ is subtree of a given Tree $T = (r_T, \mathcal{C}_T)$, denoted by $S \subseteq T$, if there exists a node $N = (r_N, \mathcal{C}_N) \in \mathcal{N}(T)$ with

$$r_N = r_S \text{ and } \forall K \in \mathcal{C}_S \, \exists C \in \mathcal{C}_N : \quad K \subseteq C$$

- If $N = T$, S should be called rooted subtree of T, denoted by $S \Subset T$.
- If $N \cong S$, S should be called complete subtree of T, denoted by $S \subseteq\!\!\!= T$.

Definition 1.9. For $S \subseteq T$ both Trees we denote $\mathcal{N}(S) \ni N = (r_N, \mathcal{C}_N) \hat{\in} \mathcal{N}(T)$ if there exists $M = (r_M, \mathcal{C}_M) \in \mathcal{N}(T)$ with $r_M = r_N$ and $\mathcal{C}_M \cap \mathcal{C}_N \neq \varnothing$
In other Words $\exists M \in \mathcal{N}(T) : \quad N \Subset M$

1.2 Cotree

With the definition above cotrees will be denoted $\mathbb{T}_{\mathcal{B}}$
With $\mathcal{B} = \{0, 1\}$ where "B" stands for boolean. Every cotree represents a cograph, where every leaf represents a vertice of the cograph. Two of them are connected if their lowest common parent's label, that's the one with lowest depth, is 1.

Definition 1.10 (Minimal Cotree)**.** Let $T = (r, \mathcal{C})$ be a cotree. T said to be minimal if for every path in T the sequence of label is always an alternating one.

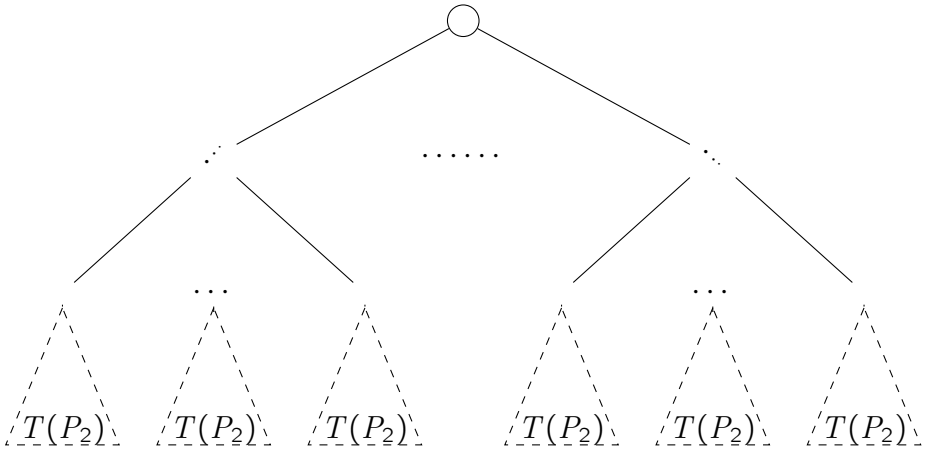
$$\text{cotree } T = (r, \mathcal{C}) \text{ is minimal } \iff \forall (r_C, \mathcal{C}_C) = C \in \mathcal{C} : \quad C \text{ is minimal, } r_C \neq r \text{ and } |\mathcal{C}| \neq 1$$

I will denote the minimal cotree of T with T_{min}

Definition 1.11 (Cotree isomorphy)**.** Two cotrees T, T' are called cotree isomorphi written $T \cong_C T'$ if their minimal cotrees $T_{min} = (r_T, \mathcal{C}_T)$ and $T'_{min} = (r_{T'}, \mathcal{C}_{T'})$ are rooted tree isomorphic: $T_{min} \cong T'_{min}$

1.2.1 Lexicographic Product

Denote the Cotree of a Cograph C as $T(C)$
Let C_1 be a Cograph. C_1 is Product of two other Cographs P_1,P_2 if it is cotree isomorphic to $T(P_1) \lhd T(P_2)$.
In other Words there exists a cotree, representing the identical cograph of the form:



Where if one would remove every subtree $T(P_2)$ one would get $T(P_1)$, as well as you would remove every former leaf from the tree.
$T(P_1) \lhd T(P_2)$ just means attach $T(P_2)$ where a leaf in $T(P_1)$ is.

Definition 1.12 (Factor of some Cotree)**.** Particularly a cotree F is factor of some cotree T denoted by $T \lhd F$ if

$$\forall L \in \mathcal{L}(T_{min}) \, \exists S \subseteq T_{min} : \quad L \in \mathcal{L}(S) \text{ and } S \cong_C F \qquad (1)$$

let those subtrees be $S_1, ..., S_n \quad n \in \mathbb{N}$ with minimal height.

$$\forall i, j \in \{1, ..., n\} : \quad i \neq j \implies \mathcal{N}(S_i) \backslash \{S_i\} \cap \mathcal{N}(S_j) \backslash \{S_j\} = \varnothing \qquad (2)$$

e.g. their nodes, except the root, are pairwise disjoint. We call them F-subtrees of T.

If F and T are minimal we may refer in (1) to the classical rooted tree isomorphy.

From the definition above it schould be clear that $n \Big| |\mathcal{L}(T)|$ from which follows that $|\mathcal{L}(F)| \Big| |\mathcal{L}(T)|$.

Theorem 1.1. *$S_1, ..., S_n$ are minimal or ismorph to the trivial factor.*

Proof. Let $F_{min} = (r_F, \mathcal{C}_F) \cong_C S_i \quad \forall i \in \{1, ..., n\}$ and $|\mathcal{C}_F| > 0$ e.g. F is not the trivial factor
Let $S_i = (r_{S_i}, \mathcal{C}_{S_i})$ then $|\mathcal{C}_{S_i}| \leq 2$ because it isn't the trivial factor and (2). For the reason that T is minimal the children of S_i are as well, and because it there at least two $C \in \mathcal{C}_{S_i}$ itself has to be minimal.
If the other hand F is the trivial factor because of (1), S_i will be as well. ■

Theorem 1.2. *Another interesting fact arising, for a given cotree T, factor cotree F and subtrees $S_1, ..., S_n \quad n \in \mathbb{N}$:*

$$(1) \qquad \forall N \in \mathcal{N}(T_{min})\, \exists i \in \{1, ..., n\} : \quad d(N) < d(F_{min}) \implies N \in \mathcal{N}(S_i)$$

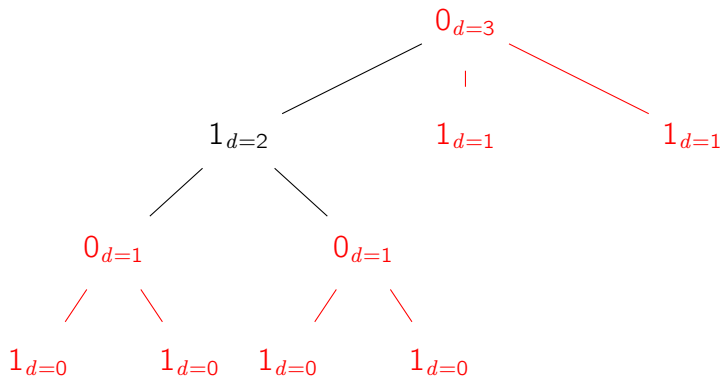
and

$$(2) \qquad \forall N \in \mathcal{N}(T_{min})\, \exists i \in \{1, ..., n\} : \quad d(N) = d(F_{min}) \implies N \ni S_i \cong F_{min}$$

Proof. (1) Let $N \in \mathcal{N}(T_{min})$ with $d(N) = k < d(F_{min})$
This means for every leaf $L \in \mathcal{L}(N)$ ther exists $\mathbb{N} \ni t \leq k$ with N being the $t'th$ parent of L. Because every leaf of T: $L \in \mathcal{L}(T_{min})$ is in some $F-$subtree of T, namely S_i, with $k < d(S_i) = d(F_{min})$ it follows that N is a node of some of those subtrees S_i.

(2) Let $N \in \mathcal{N}(T_{min})$ with $d(N) = k = d(F_{min})$
This means for every leaf $L \in \mathcal{L}(N)$ ther exists $\mathbb{N} \ni t \leq k$ with N being $L's$ $t'th$ parent.
Now suppose $N \not\ni S_i$ for all $F-$subtrees of T S_i. From (1) it follows that $F \in \mathcal{N}(S_i)$ for some $F-$subtrees of T S_i. And with that $d(S_i) > d(F_{min}) = k$ which contradicts $S_i \cong F_{min}$
■

With this counterexample it will be clear, why the opposite direction won't hold:



Our factor will be the minimal cotree for the cograph $(V, E) = (\{1, 2\}, \varnothing)$. The subtrees S_1, S_2, S_3 (read from left to right) are marked red. As you see the root of the whole tree is root of S_3 too, but with depth 3 it is indeed not smaller than $d(S_3)$.
The background of this happening is the ambiguous depth of this node, because without the left subtree, depth of this node would be 1 as well.
With this in our mind we should think about creating an unambiguous cotree from a normal one.

1.3 Balancing Cotrees

1.4 Labeling

A Labeling for Trees gives every node of the tree a equivalence class with an order between them. Two of them be synonymous if they have the same equivalence classes and there exists an order isomorphism between their equivalence classes.

1.5 Connection of depth and level

Depth and Level are two non synonymous methods for labeling nodes in trees.
But if the tree has unambiguous depth then the two labelings are indeed synonymous with inverse equivalence class ordering.

Proof. by Induction Let $T = (r, \mathcal{C})$ be a Tree with unambiguous depth and $d(T) = n$

- Base Case:
  For $n = 0$ T is a leaf $\implies$ both labelings are synonymous

  $$l_T^{(max)} - l_T(T) = 0 - 0 = 0 = d(T)$$

  with $l_T^{(max)}$ the number of levels of the tree T

- Induction Hypothesis:
  Let the labelings be synonymous for trees T with unambiguous depth of n. N node of T and with order isomorphism:

  $$l_T^{(max)} - l_T(N) = d(N)$$

  with $l_T^{(max)}$ the number of levels of the tree T

- Induction Step:
  Let $T = (r, \mathcal{C})$ be a tree of unambiguous depth
  $\implies$ all $C \in \mathcal{C}$ are of unambigous depth.
  For every node N of $C \in \mathcal{C}$:

  $$l_T(N) = l_C(N) + 1$$

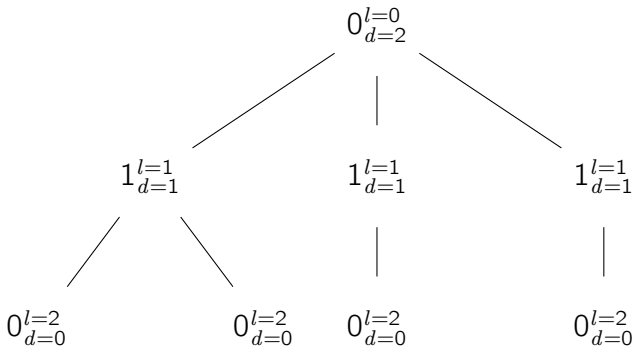
  because every child of T is on level 1 and itself is unambiguous.
  With that let N be node of child $C \in \mathcal{C}$:

  $$l_T^{(max)} - l_T(N) = l_T^{(max)} - (l_C(N) + 1) = l_C^{(max)} + 1 - (l_C(N) + 1) \quad (1)$$
  $$= l_C^{(max)} - l_C(N) \overset{IH}{=} d(N) \quad (2)$$

  And for T: $d(T) = l_T^{(max)} - l_T(T) = l_T^{(max)}$
  with $l_T^{(max)}$ the number of levels of the tree T
  $\implies$ the two labelings are synonymous, because the order isomorphy is continued (see (1)).

■

Example:



1.6 Balancing

Because we may alter Cotrees in certain ways to leave them Cotree-Isomorphic to itself, e.g. it represents the same Cograph, we could use that to simplify our problem.
One way to do so is balancing the tree, that its depth will be unambiguos.
The Process starts inductively from the leafs, in other words from every node of the tree with depth equal to 0.

Proof. Induction for $d(T) = n$

- Base Case:

  $$n = 0 \implies \text{depth is unambiguous because there are no children}$$

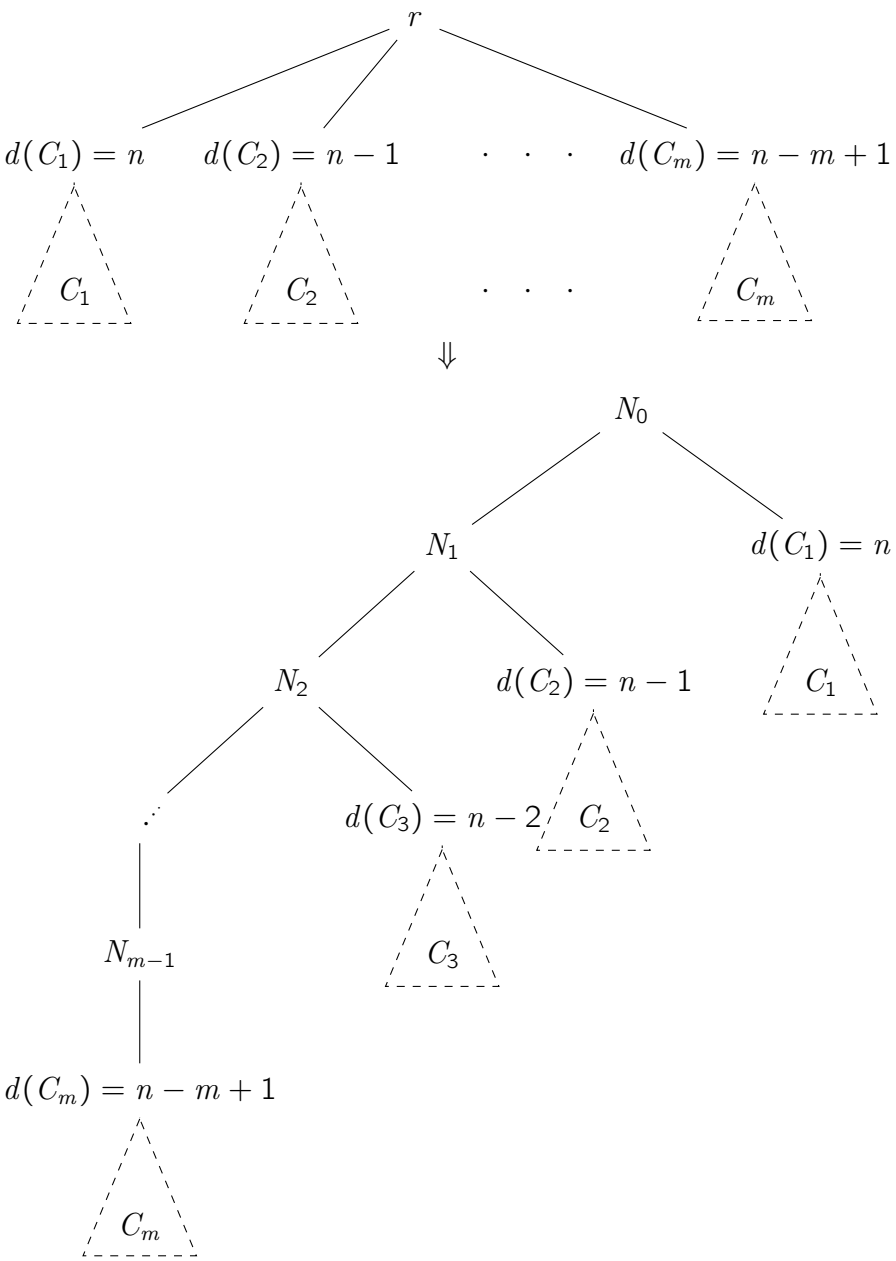
- Induction Hypothesis:
  For every child of T: $C_1, ..., C_m$ the depth is unambiguous.

- Induction Step:
  Let $T = (r, \mathcal{C})$ be a tree with $d(T) = n + 1$
  $\implies \forall C \in \mathcal{C} : \quad d(C) \leq n$ as well as C's depth is unambiguous
  let $d_{min} = min(\{d(C) \mid C \in \mathcal{C}\})$
  Now disconnect every child and attach a chain of nodes each with the same label as r. The length of the chain said to be $n - d_{min}$.
  Let $N_1, ..., N_\delta$ be its nodes, counted from bottom to top. And let N_0 be r
  Now for every disconnected child C attach it to the chain on node:

  $$N_{n-d(C)}$$

  Now $N_0, N_1, ..., N_{n-d_{min}}$ have unambiguous depth.
  Node $N_{n-d_{min}}$ has unambiguous depth because all its childs are of depth d_{min}.
  If $N_{n-d_{min}-k}$ has unambiguous depth so has $N_{n-d_{min}-k-1}$, because the depth of every attached child is $d_{min} + k$ as well as for the child node $N_{n-d_{min}-k}$. Every child of $N_{n-d_{min}-k-1}$ has unambiguous depth.

■



We denote a given tree as balanced if it arises from the minimal cotree of an arbitrary cotree put trough this algorithm. Written $\beta(T) = \beta(T_{min})$.
With this method we have got a way for ensuring that level-labeling and depth labeling are indeed synonymous, providing us an easy way to modify the AHU-Algorithm after our needs.

Theorem 1.3. *For two cotrees T_1, T_2 it holds that*

$$T_1 \cong_C T_2 \quad \Longleftrightarrow \quad \beta(T_1) \cong \beta(T_2)$$

Proof. $''\Rightarrow''$ From $T_1 \cong_C T_2$ it follows that $T_{1_{min}} \cong T_{2_{min}}$. Because the Algorithm depends just on the depth of the nodes of the tree, and the two minimal cotrees are isomorphic they get altered the same way which means $\beta(T_1) \cong \beta(T_2)$
$''\Leftarrow''$ From $\beta(T_1) \cong \beta(T_2)$ it follows that $T_{1_{min}} = \beta(T_1)_{min} \cong \beta(T_1)_{min} = T_{2_{min}}$ and from that $T_1 \cong_C T_2$ ■

Theorem 1.4. *For a minimal cotree T_{min} and a minimal subtree $S \subseteq T_{min}$ with $\mathcal{L}(T_{min}) \cap \mathcal{L}(S) \neq \varnothing$, e.g. they share at least one leaf:*
$\beta(S) \subseteq \beta(T_{min})$ and its root will be found on depth $d(\beta(S))$

Proof. Let be $N \in \mathcal{N}(T_{min})$ with $S \Subset N$. It follows that $d(S) \leq d(N)$ and $d(\beta(S)) \leq d(\beta(N))$, because they share at least one leaf. Now let $S = (r_S, \mathcal{C}_S)$, $N = (r_N, \mathcal{C}_N)$ and $D \in \mathcal{N}(\beta(N))$ with $\forall C_N \in \mathcal{C}_N : \quad D \not\cong_C C \quad \wedge \quad d(D) = d(\beta(S))$. In other words D is the node introduced by the algorithm with the depth of S.

MAYBE INDUCTION for subsettrees with same root? ■

1.7 Properties for the lexicographical Product of Cographs

1.7.1 Unambigous depth, Balanced Cotrees and Factorization

Theorem 1.5. *Let $T_{min}, F_{min} \in \mathbb{T}_{\mathcal{B}}$ be cotrees with $T_{min} \lhd F_{min}$ and unambiguous depths.*
$S_1, ..., S_n \subseteq T$ the F_{min}-subtrees of T_{min}. I state that:

$$(1) \qquad \forall N \in \mathcal{N}(T_{min})\, \exists i \in \{1, ..., n\} : \quad d(N) < d(F_{min}) \iff N \in \mathcal{N}(S_i)$$

and

$$(2) \qquad \forall N \in \mathcal{N}(T_{min})\, \exists i \in \{1, ..., n\} : \quad d(N) = d(F_{min}) \iff N \Supset S_i \cong F_{min}$$

Proof. (1) "⇒" proved above.
"⇐" Let $N \in \mathcal{N}(T_{min})$ with $N \in \mathcal{N}(S_i)$ for some $i \in \{1, ..., n\}$
$N's$ depth will be unambiguous because T_{min} has unambiguous depth. As well as S_i, for the reason that $\mathcal{L}(S_i) \subset \mathcal{L}(T_{min})$ and all $S_i's$ are pairwise disjoint except for their root $R \in \mathcal{N}(T_{min})$ which itself has unambiguous depth.
Because there $\exists D \in \mathcal{N}(T_{min})$ with $S_i \Subset D$ and both are unambiguous from which follws that $d(S_i) = d(D) = d(F_{min})$. And because $N \in \mathcal{N}(D)$ it follows that $d(N) < d(D) = d(F_{min})$

(2) "⇒" proved above.
"⇐" Let $N \in \mathcal{N}(T_{min})$ with $N \Supset S_i$ for some $i \in \{1, ..., n\}$. They have at least one common leaf with each other.
$N's$ depth will be unambiguous because T_{min} has unambiguous depth, as well as S_i.
Because both are unambiguous and have common leafs this implies $d(S_i) = d(N) = d(F_{min})$.
■

The theorem states that for those cotrees we may identify the nodes D of T with $S_i \Subset D$ just by their depth. With that the search for factors will be doable.
But because this are special forms of cotrees we need to get this statement for cotrees originated from some unambiguous but arbitrary minimal cotrees.

Theorem 1.6. *Let $T, F \in \mathbb{T}_{\mathcal{B}}$ be cotrees with $T \lhd F$.*
$S_1, ..., S_n \subseteq T$ the F-subtrees of T. As well as F is not the trivial factor. I state that:

$$(1) \qquad \forall N \in \mathcal{N}(\beta(T))\, \exists i \in \{1, ..., n\} : \quad d(N) < d(\beta(F)) \iff N \in \mathcal{N}(\beta(S_i))$$

and

$$(2) \qquad \forall N \in \mathcal{N}(\beta(T))\, \exists i \in \{1, ..., n\} : \quad d(N) = d(\beta(F)) \iff N \Supset \beta(S_i) \cong \beta(F)$$

Proof. (1) "⇒" Let $N \in \mathcal{N}(\beta(T))$ with $d(N) < d(\beta(F))$. It should be clear that $N = \beta(N)$. Because $\exists i \in \{1, ..., n\} \forall L \in \mathcal{L}(N)$ with $L \in \mathcal{L}(S_i)$, because from $S_{i_{min}} \cong F_{min}$ it follows that $\beta(S_i) \cong \beta(F)$ [Th. 1.3]. Therefore $N \in (\beta(S_i))$.
"⇐" Let $N \in \mathcal{N}(\beta(S_i))$. From Theorem 1.3 and $S_{i_{min}} \cong F_{min}$ it follows that $\beta(F) \cong \beta(S_i)$ which implies $d(\beta(S_i)) = d(\beta(F))$ and with that $N < d(\beta(F))$

(2) "⇒" Let $N \in \mathcal{N}(\beta(T)) : \quad d(N) = d(\beta(F))$, it is clear that $N = \beta(N)$. Now let $L \in \mathcal{L}(N)$ be a leaf of N. We know that there $\exists i \in \{1, ..., n\}$ with $L \in \mathcal{L}(\beta(S_i))$. Then $d(\beta(S_i)) = d(\beta(F)) = d(N)$. So if $N \not\ni \beta(S_i)$ this contradicts the assumption, because then N has to be above or below $\beta(S_i)$. With that $N \ni \beta(S_i) \cong \beta(F)$
"⇐" Let $\mathcal{N}(\beta(T)) \ni N \ni \beta(S_i) \cong \beta(F)$ because of the unambiguity of $N, \beta(S_i), \beta(F)$ and share at least one child they surely have the same depth.
$\implies d(N) = d(\beta(S_i)) = d(\beta(F))$

■

The great thing about this theorem is, we proved that we may find the Roots of the F-Subtrees of T on the same depth. This will give us a way to use the AHU Algorithm on the depth, where once there is over one depth just one id, suggesting that every subtree appending there will be isomorphic to the others. Of course it will be a little harder than that.

1.7.2 Splitting

The Hard Part about finding Factors will be the cotrees T in which at least one node N has at least two S_i, S_j as its rooted subtrees.
The key idea for solving this is, after converting T to $\beta(T)$ with unambiguous depth, we may split up nodes if they are not prime and can be splitted. With that we always assure that children will indeed always be prime.

Definition 1.13. Let T be a cotree.. $N = (r_N, \mathcal{C}_N) \in \mathcal{N}(T)$ with $\mathcal{P}(N) = P = (r_P, \mathcal{C}_P)$ and $r_P = r_N$. $\mathcal{C}_N = \{(C_1, m_1), ..., (C_n, m_n)\}$ (in thought of a multiset). Let $g = gcd(m_1, ..., m_n) \neq 1$. Then: To split N (written $\zeta(N)$) means:

- $\mathcal{C}_{P'} := C_P \backslash N \cup \{(\hat{N}, g)\}$ with

$$\hat{N} = (r_{\hat{N}}, \mathcal{C}_{\hat{N}}) \qquad r_{\hat{N}} = r_N, \quad \mathcal{C}_{\hat{N}} = \{(\zeta(C_1), m_1/g), ..., (\zeta(C_n), m_n/g)\}$$

And then we set:

$$P' := (r_P, \mathcal{C}_{P'})$$

This is well defined, because $g \,\big|\, m_1, m_2, ..., m_n$. And for the reason that $g = gcd(m_1, ..., m_n)$ there $\nexists d \in \mathbb{N}\backslash\{1\}$ with $d \,\big|\, \frac{m_1}{g}, ..., \frac{m_n}{g}$, e.g. the multiset $\mathcal{C}_{\hat{N}}$ is prime.
The cotree resulting from this T' is cotree isomorphic to T because in T_{min} N will be collapsed into P because they share the label. For T'_{min} all $\hat{N}$ will be collapsed into P' as well. With that and because nothing else changed in the splitting of N: $T'_{min} \cong T_{min}$ impling T and T' are cotree isomorphic. ■

Theorem 1.7. *It should be clear that for two cotrees $T, T' \in \mathbb{T}_{\mathcal{B}}$ with*

$$T \cong_C T' \iff \zeta(\beta(T)) \cong \zeta(\beta(T'))$$

Proof. Because $T_1 \cong_C T_2 \iff \beta(T_1) \cong \beta(T_2)$ and ζ depending completly on tree structure, both Trees will be altered the same way. ■

Theorem 1.8. *Now let $T, F \in \mathbb{T}_{\mathcal{B}}$ be cotrees with $T \lhd F$. And assume we made both unambiguous: $T := \beta(T)$, $F := \beta(F)$*
$S_1, ..., S_n \subseteq T$ the F-subtrees of T. And F being the first factor found, e.g. $S_1, ..., S_n$ having lowest depth of all Factors. Assume that $i_0 = 0, i_1, ..., i_k, i_{k+1} = n \in [0, n] \cap \mathbb{N}$ with $i_0 < i_1 < ... < i_{k+1}$ where $S_j \Subset P_t = (p_t, \mathcal{C}_{P_t})$ with $i_t < j < i_{t+1}$. That says just that the Nodes $S_1, ..., S_n$ have the same root iff they lay inside the same interval. We also have to assume that we already did $\zeta(S_1), ..., \zeta(S_n)$. Then

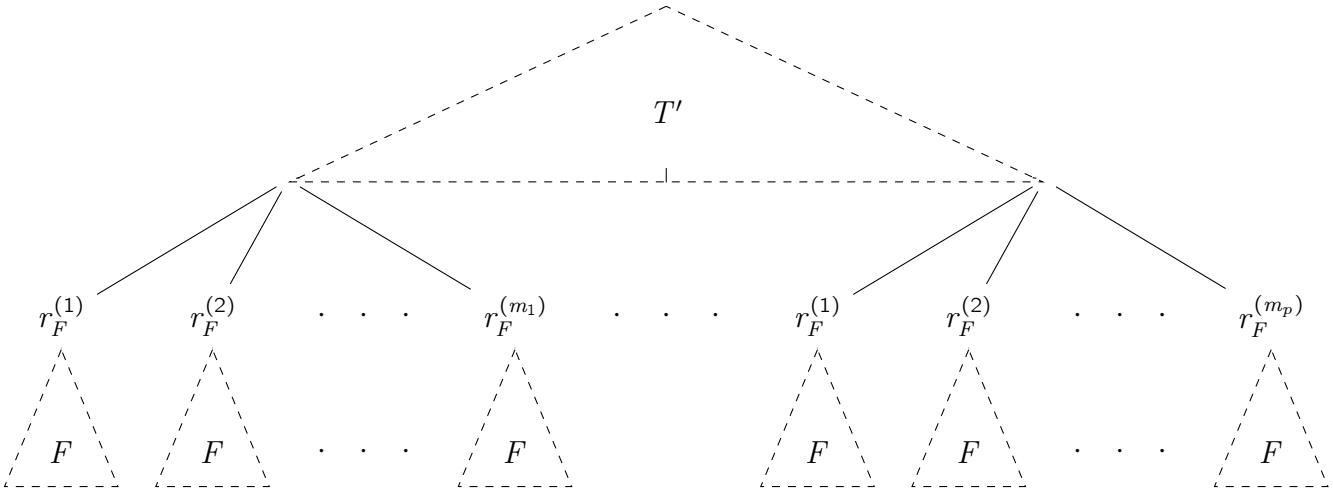
$$\forall \mathcal{P}(P_j) = R_j = (r_j, \mathcal{C}_{R_j}),\ j \in \{0, ..., k\} : \quad r_j = p_j \quad \Longrightarrow \quad R_j \cong_C (p_j, \{(S_j, i_{j+1} - i_j)\} \cup \mathcal{C}_{R_j} \backslash P_j)$$

$$\forall \mathcal{P}(P_j) = R_j = (r_j, \mathcal{C}_{R_j}),\ j \in \{0, ..., k\} : \quad r_j \neq p_j \quad \Longrightarrow \quad P_j \cong_C (p_j, \{(S_j, i_{j+1} - i_j)\})$$

That is if we may split up the nodes where the roots of the F-subtrees of T lay, the parent R_j will indeed have just a number of S_j as children. If we aren't allowed to split P_j we also know that it will have just a number of S_j as children. So for both the only children they got are the S_j from which it follows that the corresponding Knuth Tuple (in respect to multisets) for every Node with $d(P_j)$ will surely have a prime Tuple (in respect to multisets) dividing each of them and isn't divisible by any multisets. It's clear that this Knuth Tuple matches the of S_j.
As stated above, if we ignore the S_j there will be found K_n or $\bar{K}_n$ cographs as cotrees on P_j or R_j. This will come in handy later, when showing that there is no other Factor of T that wont be divided by F.

If we use the construction in 1.8, the next factor found will be of the form K_n or $\bar{K}_n$. The second one consisting just of n unconnected vertices. But how to choose n?

Theorem 1.9. *Let $m_1, ..., m_p \quad p \in \mathbb{N}$ be the multiplicities of the S_j from R_j or P_j. The new tree will be looking like:*



This construction suggesting that we have to choose n:

$$n := gcd(m_1, m_2, ..., m_p)$$

Because that is the K_n all of them have in common. Observe that if $n = 1$ the factor is indeed the trivial factor. We now know that the next factor will be K_n or $\bar{K}_n$ depending on the label of F. Because this factor gets taken out of the tree the remaining children of the nodes holding F m_i times, should now be leafs with number $\frac{m_i}{n}$. The factorization of K_n or $\bar{K}_n$ itself will be unambiguous and depends on the prime factorization of n, order not neglected.

Theorem 1.10. *The factorization of K_n with $n = p_1 \cdot p_2 \cdot ... \cdot p_k \quad p_1, ..., p_k$ prime will be:*

$$K_n = K_{p_{\tau(1)}} \circ K_{p_{\tau(2)}} \circ ... \circ K_{p_{\tau(k)}} \qquad \forall \tau \in S_k$$

The construction for $\bar{K}_n$ can be done similarly.

With this we now can extract the last prime factor, while afterwards constructing the different factorizations for the remaining K-parts. Now looking at every node with depth above the shortly used it is clear that F will be factor of it. One problem remaining are cotrees having multiple factorizations, the K_n not counted. First looking at it it seems we have to traverse the tree recursively where recursion is called if a factor is found, thus the recursive call ignoring this factor. However

Theorem 1.11. *Let $T, F_1, F_2 \in \mathbb{T}_{\mathcal{B}}$ with $T \triangleleft F_1$, $T \triangleleft F_2$ and $K_n \neq F_1 \neq F_2 \neq K_n$, F_1, F_2 prime. Then $\exists S_1, S_2 \subseteq T$ with $F_1 \cong_C F_1' \Subset S_1$ and $F_2 \cong_C F_2' \Subset S_2$*

$$S_1 = S_2$$

Proof. Now assume $S_1 \neq S_2$ then we may choose them with $S_1 \subset S_2$ or $S_2 \subset S_1$ but then $F_2 \triangleleft F_1$ or $F_1 \triangleleft F_2$ respectively contradicting the assumptions. ■

From the theorem above it should be clear that the recursion only has to search for another factor till S_1 is exceeded, especially the S_1 with lowest depth. For implementation purposes after finding factor on subtrees of T:

$$V_1, ..., V_n : \quad d' = min\{m \mid i \in \{1, ..., n\} m = max\{d(\beta(P)) \mid P \in \mathcal{P}(V_i)\}\}$$

d' the highest depth the recursion should search for another factor.

1.8 Find Isomorph Trees with AHU

A brilliant paper about this topic: [Tree Isomorphism Algorithms: Speed vs. Clarity, Author(s): Douglas M. Campbell and David Radford]
The Algorithm of Aho, Hopcroft and Ullman is some brilliant Algorithm for check if some unordered rooted trees are isomorphic. It will work bottom up level wise and in $\mathcal{O}(n)$

1.8.1 Knuth Tuples

An easy way to check for isomorphy are Knuth Tuples. They are constructed using $\Sigma = \{(,)\} \cup M$ inductively. We will approach this from the view of multisets, which helps us comparing and manipulating those tuples flawlessly. Construct Knuth-Tuple for Tree $T \in \mathbb{T}_M$ where children have isomorphy invariant id's:
$\mathcal{K} : \mathbb{T}_M \longrightarrow \mathcal{P}(\mathbb{N} \times \mathbb{N})$

- $T = (r, \varnothing) \Leftrightarrow T$ is leaf: $\mathcal{K}(T) = \varnothing$
- $T = (r, \mathcal{C}) : \mathcal{K}(T) = \{(i, n) \mid i \text{ id}, n \text{ number of children with this id}\}$

With this calculation for the Knuth-tuple we're able to traverse the unambiguous tree bottom up and compare the Knuth-Tuples per depth. While splitting every node we are allowed to assure the subtrees we are dealing with will be prime. This splitting is done by finding the prime multiset dividing the one we are looking at. The quotient in $\mathbb{N}$ the new multiplicity of the prime tuple above.

2 Practical Improvements

3 Algorithm

3.1 Cotree with unambiguous depth from minimal cotree

By creating new nodes:

for $\forall C = (r_c, \mathcal{C}_C) \in \mathcal{C}_T$ **do**
  calculateUnambiguousCotree$(r_C, \mathcal{C}_C)$ ▷ Assure Children will be unambiguous
end
$\mathcal{D}_{max\{d(C) \mid C \in \mathcal{C}_T\}-1} := \mathcal{C}_T$
for $i := max\{d(C) \mid C \in \mathcal{C}_T\} - 1$ *to* $i = min\{d(C) \mid C \in \mathcal{C}_T\}$ **do**
  $\mathcal{C}_i := \{C \mid d(C) = i \wedge C \in \mathcal{C}_T\}$
  $\mathcal{D}_{i+1} := \varnothing$
  $\mathcal{D}_i := \mathcal{D}_i \cup (r_T, \mathcal{D}_{i+1})$
  if $C_i \neq \varnothing$ **then**
    $\mathcal{C}_T := \mathcal{C}_T \backslash \mathcal{C}_i$
    $\mathcal{D}_{i+1} := \mathcal{D}_{i+1} \cup \mathcal{C}_i$
  end
end

Function calculateUnambiguousCotree$(r_T, \mathcal{C}_T)$ for $T_{min} = (r_T, \mathcal{C}_T) \in \mathbb{T}_{\mathcal{B}}$

By assigning multiple depths to ambiguous nodes:

$r := r_T$
$\mathcal{C} := \varnothing$
$d_{min} := \infty$
$d_{max} := -\infty$
for $\forall C = (r_c, \mathcal{C}_C) \in \mathcal{C}_T$ **do**
  $(r_N, \mathcal{C}_N, d^N_{min}, d^N_{max}) :=$ calculateUnambiguousCotree$(r_C, \mathcal{C}_C)$
  $\mathcal{C} := \mathcal{C} \cup \{(r_N, \mathcal{C}_N, d^N_{min}, d^N_{max})\}$
  if $d^N_{max} + 1 < d_{min}$ **then**
    $d_{min} := d^N_{max} + 1$
  end
  if $d^N_{max} + 1 > d_{max}$ **then**
    $d_{max} := d^N_{max} + 1$
  end
end
return $(r, \mathcal{C}, d_{min}, d_{max})$

Function calculateUnambiguousCotree$(r_T, \mathcal{C}_T)$ for $T_{min} = (r_T, \mathcal{C}_T) \in \mathbb{T}_{\mathcal{B}}$ returns $(r, \mathcal{C}, d_{min}, d_{max})$

$r := r_T$
$\mathcal{C} := \varnothing$
$d_{min} := \infty$
$d_{max} := -\infty$
for $\forall C = (r_c, \mathcal{C}_C) \in \mathcal{C}_T$ **do**
  $(r_N, \mathcal{C}_N, d^N_{min}, d^N_{max}) :=$ calculateUnambiguousCotree$(r_C, \mathcal{C}_C)$
  $\mathcal{C} := \mathcal{C} \cup \{(r_N, \mathcal{C}_N, d^N_{min}, d^N_{max})\}$
  if $d^N_{max} + 1 < d_{min}$ **then**
    $d_{min} := d^N_{max} + 1$
  end
  if $d^N_{max} + 1 > d_{max}$ **then**
    $d_{max} := d^N_{max} + 1$
  end
end
return $(r, \mathcal{C}, d_{min}, d_{max})$

Function calculateUnambiguousCotree$(r_T, \mathcal{C}_T)$ for $T_{min} = (r_T, \mathcal{C}_T) \in \mathbb{T}_{\mathcal{B}}$ returns $(r, \mathcal{C}, d_{min}, d_{max})$

```
(r, C, d_min, d_max) :=calculateUnambiguousCotree(r_T, C_T);
depthdict[][];
filldepthdict(depthdict);
factors[];
for d = 1; d ≤ d_max; d := d + 1 do
    primeTuples[];
    gcds[];
    ids[];
    for every node N = (r_N, C_N, d_Nmin, d_Nmax) in depthdict[d] do
        KT :=getKnuthTuple(N, d);
        if primeMultiset(KT) is not in primeTuples then
            primeTuples.append(primeMultiset(KT));
        end
        gcds[N] := k    with k·primeMultiset(KT) = KT;
        ids[N] := primeTuples.position(primeMultiset(KT));
    end
    if primeTuples.length() == 1 then
                                                          ▷ factor found
        write ids, multiplicities and kchilds
    end
    else
    end
end
return (r, C, d_min, d_max)
```

Function calculateFactors($r_T, \mathcal{C}_T$) for $T_{min} = (r_T, \mathcal{C}_T) \in \mathbb{T}_{\mathcal{B}}$ returns $S \subseteq \mathbb{T}_{\mathcal{B}}^n$