Counting cherry-picking sequences in phylogenetic trees

by

Marit Anna Dee

to obtain the degree of Bachelor of Science
at the Delft University of Technology,

Student number: 5403626
Project duration: March 5, 2024 – June 20, 2024
Thesis committee: Dr. Y. Murakami, TU Delft, supervisor
Dr. W. Groenevelt, TU Delft

TU Delft

Abstract

Phylogenetic trees are used to represent evolutionary history of, among others, species, genes or languages. Phylogenetic trees can be reduced by so called cherry-picking sequences. In this thesis we take a closer look at cherry-picking sequences to obtain a more fundamental understanding on the structure and behaviour of these sequences on phylogenetic trees. For binary rooted trees, that is trees with a root, outdegree-2 tree nodes, and a set of leaves, we count the number of sequences that can reduce such trees. Furthermore, we try to find the number of sequences of a given tree that is needed to reduce all subtrees of a given tree. Thus, an enumeration problem and an optimization problem are considered in this thesis. For both problems, we first look for results on simply structured trees, such as caterpillars and double caterpillars. Lastly, we try to expand these results to general binary trees in order to find the answers to the two problems.

Contents

1 Introduction

Phylogenetic networks are used to represent the evolutionary history of biological species, genes or languages. These directed graphs depict evolutionary relationships among taxa, which are represented by leaves. Traditionally, these graphs are phylogenetic trees, which are rooted directed graphs where every node has exactly one incoming edge (except for the root).

However, the evolutionary history of organisms or large DNA sequences is often more complex than a tree-like structure. Evolutionary processes like hybridization, recombination or horizontal gene transfers lead to reticulations, which phylogenetic networks can better represent rather than phylogenetic trees [1].

A specific class of phylogenetic networks, called *cherry-picking networks*, or *orchard networks* was introduced by Janssen and Murakami [2] and independently by Erdös et al. [3]. These networks can be reduced to a single leaf with *cherry-picking sequences*, which will be detailed in Chapter 2.

Janssen and Murakami have shown that within a particular reconstructible class, cherry-picking networks are uniquely determined by their smallest cherry-picking sequence [2]. This result is used to address the problem of network containment, that is whether or not a given network is contained in another network [2, 4].

In this thesis, we will focus on binary phylogenetic trees, which is a subclass of orchard networks. First, we try to obtain fundamental insights in the structure and behaviour of cherry-picking sequences on binary phylogenetic trees by addressing an enumeration problem. Therefore, our first research question that we try to answer in this thesis is the following.

Input: A binary phylogenetic tree T on n leaves.
Question: How many different cherry-picking sequences can reduce T?

Furthermore, we take a closer look at the relation between subtrees and cherry-picking sequences via an optimization problem. A way to differentiate between cherry-picking sequences is to look at the number of subtrees that a sequence reduces. We say that two sequences are *equivalent* if they reduce the same subtrees of a given tree. Looking at the number of sequences that we need to reduce all subtrees of a tree would be equivalent to looking at how many equivalence classes of sequences we need to *cover* all subtrees. This brings us to our second research question that we want to answer in this thesis.

Input: A binary phylogenetic tree T on n leaves.
Question: How many cherry-picking sequences do we need to reduce all subtrees of T?

The structure of the thesis is as follows. In Chapter 2, we state necessary definitions and introduce cherry-picking sequences. In Chapter 3, we answer the two main research questions for caterpillars. In Chapter 4, we use the results from Chapter 3 to answer the two research questions for double caterpillars. In Chapter 5, we combine the results from the previous chapters in order to find the answers to the two main research questions for general binary trees. Finally, in Chapter 6, we give a summary of the results that are found in Chapters 3, 4 and 5 and state some suggestions for future research.

2

Preliminaries

First, we state a few definitions. Most of the definitions are based on the definitions stated in the article "On cherry-picking and network containment" [2].

Definition 2.1. A *phylogenetic tree* T is a directed acyclic graph, with vertices, or nodes V and edges E on a non-empty taxa set X. T has one *root* (indegree-0, outdegree-2), a set $L(T)$ of *leaves* (indegree-1, outdegree-0) bijectively labeled with X and all other nodes are *tree nodes* (indegree-1, outdegree at least 2). A phylogenetic tree is called *binary* if each tree node has outdegree-2.

Since all trees considered in this thesis are binary phylogenetic trees, the terms 'phylogenetic' and 'binary' will be dropped in the rest of this thesis. We primarily look at trees on the set of leaves $L = \{1, 2, \dots, n\}$.

For any edge uv in T, u is called a *parent* of v and v is called a *child* of u. If there exists a directed path from node u to v, then u is *above* v and v is *below* u. We call u the *tail* and v the *head* of an edge uv. The *depth* of a node u is given by the length of a shortest directed path from the root to u. A node with the highest depth is called the *deepest node* of T. There can be multiple deepest nodes in a tree.

For any set of vertices V, we call u the *lowest common ancestor* (LCA) of V if $u \notin V$, u is above all vertices $v \in V$ and no other vertices below u have this property. We denote u by $u = lca(V)$. Note that the LCA of any two vertices in a tree is unique.

Definition 2.2. Let T be a tree on n leaves. Let u and v be the two children of the root. We can relabel the leaves of T, such that we have $T_1 = \{1, \dots, j\}$ and $T_2 = \{j+1, \dots, n\}$ as a partition of the leaves $\{1, \dots, n\}$, where T_1 contains all leaves below u and T_2 all leaves below v. We call T_1 and T_2 the two *branches* of T.

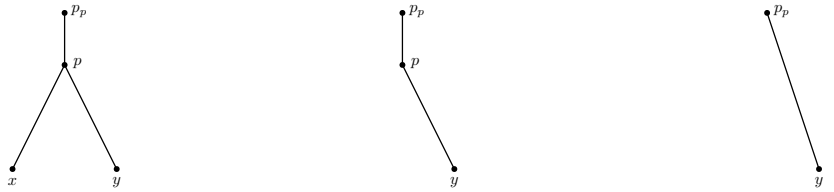
2.1. Cherry-picking sequences

In this section we introduce cherry-picking sequences. Therefore, we first need to define what a cherry is, and how one can pick a cherry.

Definition 2.3. Let (x, y) be an ordered pair of leaves in a tree T. We call (x, y) a *cherry* if x and y share a common parent, that is, if $p_x = p_y$, where p_x and p_y denote the parents of x and y, respectively. We say that (x, y) is a *reducible pair* if (x, y) is a cherry in T.

We can *reduce* a reducible pair from a tree to obtain a tree of a smaller size. Let T and T' be trees. We call T' a *subtree* of T if T' is can be obtained from T by deleting nodes and contracting edges. We denote the set of all subtrees of a tree T by $\mathcal{D}(T)$. With a subtree *rooted* at a node u of T, we refer to the subtree obtained by removing all nodes and their adjacent edges that are not below u. The (new) root of the obtained subtree is u.

Definition 2.4. Let T be a tree and let (x, y) be an ordered pair of leaves in T. If (x, y) is a cherry, let p denote the common parent of leaves x and y and let p_p denote the parent of p. We may *pick cherry* or *reduce the pair* (x, y) in T by executing the following steps



(a) Cherry (x, y). (b) Remove leaf x and edge px. (c) Suppress node p.

Figure 2.1: Picking cherry (x, y) following the steps from Definition 2.4

1. Remove leaf x in T together with the edge px;
2. Suppress node p by removing node p and replacing the edges py and $p_p p$ with one edge $p_p y$.

If (x, y) is not a cherry in T, we may reduce or pick the pair (x, y) in T by doing nothing.
The resulting tree after reducing pair (x, y) in T is denoted by $T(x, y)$. We say that (x, y) *affects* T if $T(x, y) \neq T$.

In Figure 2.1 the two steps of picking a cherry (x, y) are shown.

We can repeat the action of cherry-picking in a tree to eventually reduce a tree to a tree on a single leaf. Let T be a tree and let $S = S_1 S_2 \ldots S_{|S|} = (x_1, y_1) \ldots (x_{|S|}, y_{|S|})$ be a sequence of ordered pairs. The tree obtained by repeatedly reducing T with every element of S in order, is denoted by TS. We say that S reduces T if TS is a *single-leaf tree*, that is if TS is a tree with a single leaf, a root and no other vertices.

Definition 2.5. Let X be a non-empty taxa set. Let S be a sequence of ordered pairs of distinct elements of X. S is called a *cherry-picking sequence* (CPS) if the second coordinate of each ordered pair occurs as a first coordinate in some ordered pair later in the sequence or as the second coordinate of the last pair.

Definition 2.6. A CPS is a *tree-child sequence* (TCS) if every leaf appearing as the first coordinate does not appear as a second coordinate in the rest of the sequence.

Every tree has a TCS, since every CPS of a tree is a TCS.

Definition 2.7. A *subsequence* of a sequence S is a sequence of ordered pairs that are in S and appear in the same order as in S. A subsequence of a sequence can be obtained by deleting some pairs from the sequence. We say that the empty sequence is not a subsequence of any sequence.

We say that a TCS S is a *minimal* TCS for a tree T, if S reduces T and if every ordered pair in S affects the intermediate trees.
In Figure 2.2 an example of a TCS S that reduces a tree T is shown. Note that S is a minimal TCS for T, since every pair of S affects the tree. $S' = (5, 4)(1, 4)$ is an example of a subsequence of S. Note that S' is also a TCS. However, a subsequence of a TCS does not need to be a TCS in general.

Remark. A minimal TCS that reduces a tree T on n leaves, always has length $n - 1$, since with every pair a leaf is removed. T is reduced when it only has one leaf left, so with $n - 1$ pairs we can reduce T.

In this thesis we often refer to minimal TCSs for a tree as a *TCS of/for a tree* T.

Lemma 2.1. Let T be a tree on n leaves and let S be a TCS that reduces T. Let T' be a subtree of T that is reduced by S. Then there exists a unique subsequence of S that is a minimal TCS for T'.

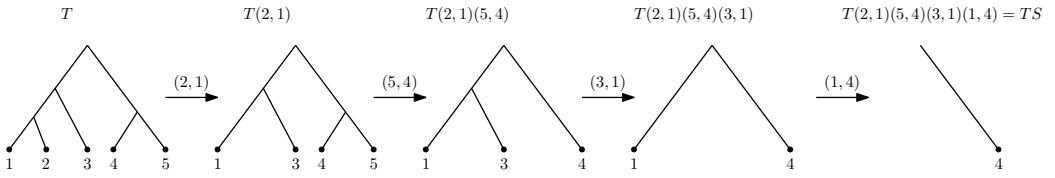


Figure 2.2: A tree T is reduced to leaf 4 by TCS $S = (2, 1)(5, 4)(3, 1)(1, 4)$.

Proof. Let T be a tree on n leaves and let S be a TCS that reduces T. Let T' be a subtree of T on k leaves, with $2 \leq k \leq n$. Note that S has $n-1$ pairs, of which exactly $k-1$ pairs affect T'. These $k-1$ pairs form a minimal TCS S' that reduces T', since every pair of S' affects T'. Note that all minimal TCSs that reduces T' have $k-1$ pairs. Therefore S' is the only subsequence of S that reduces T', that is a minimal TCS for T'.

□

A *partial TCS* S' of length m is a sequence of ordered pairs, where there exists a TCS S, such that S' is equal to the first m elements of S. If T is a tree and S and S' are partial TCSs, then applying S to T and then S' to TS is the same as appending S' to S, denoted SS', and applying the whole sequence to T. In notation, we write

$$(TS)S' = T(SS'),$$

hence we can write this tree without parentheses as TSS'.

3

Caterpillars

In this chapter we will look at simply structured trees called caterpillars. We try to count how many TCSs can reduce a caterpillar and we try to find the minimal number of TCSs that are needed to reduce all subtrees of such caterpillar.

3.1. Reducing subtrees using TCSs

In this section we try to find the minimal m and maximal M number of subtrees that can be reduced by a TCS of a caterpillar T. Doing so, we may obtain a better understanding of the possibilities for a single TCS, and see how we can combine multiple TCSs to eventually reduce all subtrees of a caterpillar.

Definition 3.1. Let T be a tree on n leaves $\{x_1, \dots, x_n\}$. Let p_i denote the parent of leaf x_i, for all i. T is called an n-*caterpillar* if $p_1 = p_2$ and (p_{i+1}, p_i) is an edge in T for all $i \in \{2, \dots, n-1\}$. The main path from the root, p_n, to leaf x_1 that connects all p_i is called the *spine* of the caterpillar.

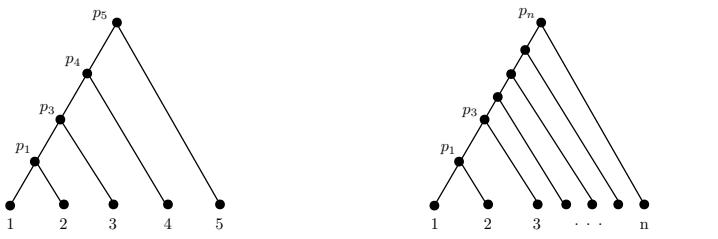
Note that the depth of every leaf in a caterpillar is unique, except for the leaves x_1 and x_2. Leaves x_1 and x_2 are the deepest nodes with depth $n-1$. We assume that a caterpillar on a taxa set $\{1, \dots, n\}$ displays its leaves in ascending order from the deepest node to the least deep node. We call a caterpillar on 3 leaves a *triplet*. In Figure 3.1 an example of a 5-caterpillar and an n-caterpillar is shown. Observe that every subtree of a caterpillar is again a caterpillar.

First, we observe how many TCSs can reduce a caterpillar on n leaves.

Lemma 3.1. Let T be an n-caterpillar, with $n \geq 2$. Then there exist 2^{n-1} different TCSs that reduce it.

Proof. We prove the lemma by induction on the number of leaves n. Let T be an n-caterpillar. Note that every TCS that reduces T has length $n-1$. For the base case, we consider a 2-caterpillar, that is a cherry (x_1, x_2). There are $2 = 2^{2-1}$ TCSs that reduce this 2-caterpillar, namely $S_1 = (x_1, x_2)$ and $S_2 = (x_2, x_1)$.

For the induction hypothesis, we assume that the lemma is true for all caterpillars on $m < n$ leaves. Let T be an n-caterpillar on the leaves $\{x_1, \dots, x_n\}$, with $n \geq 3$. T has only one cherry (x_1, x_2), so the first pair of a TCS that reduces T needs to be either (x_1, x_2) or (x_2, x_1). Upon reducing the first pair



(a) 5-caterpillar on leaves $\{1, 2, 3, 4, 5\}$ (b) n-caterpillar on leaves $\{1, \dots, n\}$

Figure 3.1: A caterpillar (a) on 5 leaves and (b) on n leaves.

(x_1, x_2), we obtain a caterpillar T' on $n-1$ leaves $\{x_2, \dots, x_n\}$. By the induction hypothesis there exist 2^{n-2} TCSs, $S_1, \dots, S_{2^{n-2}}$ that reduce T'. Adding (x_1, x_2) as a prefix to these sequences gives us 2^{n-2} different TCSs, $(x_1, x_2)S_1, \dots, (x_1, x_2)S_{2^{n-2}}$ that reduce T. With the same argument, swapping x_1 and x_2, we can find another 2^{n-2} different TCSs, $(x_2, x_1)R_1, \dots, (x_2, x_1)R_{2^{n-2}}$ that reduce T, where R_i is the sequence S_i with x_1 and x_2 swapped. So in total there are $2 \cdot 2^{n-2} = 2^{n-1}$ different TCSs that reduce T.

Thus by induction we have shown that there exist 2^{n-1} different TCSs that reduce an n-caterpillar, for all $n \geq 2$.

□

Let T be a caterpillar on n leaves and let $\mathcal{D}(T)$ be the set of all subtrees of T. The cardinality of the set of subtrees is $|\mathcal{D}(T)| = \sum_{k=2}^{n} \binom{n}{k} = 2^n - n - 1$, not counting the single-leaf trees as subtrees. We want to know how many subtrees can be reduced by a single TCS for T, by looking at the subsequences of the given TCS.

For each (minimal) TCS S that reduces T, there exist $\sum_{k=1}^{n-1} \binom{n-1}{k} = 2^{n-1} - 1$ subsequences, by removing some pairs from the sequence S. The empty sequence is not taken into account, which explains the -1 term. We do count S itself as a subsequence of S. Now not all of these subsequences are necessarily a TCS. For example, let T be a 4-caterpillar. The TCS $S = (1,2)(2,3)(3,4)$ reduces T and has subsequence $S' = (1,2)(3,4)$. However, S' is not a TCS, since 2 appears as the second coordinate of the first pair, but does not appear as a first coordinate in the rest of the sequence or in the last coordinate.

By Lemma 2.1 we know that every subtree of a caterpillar T that is reduced by a TCS S, corresponds with a unique subsequence of T that is a TCS. Therefore, the amount of subtrees that can be reduced by a TCS S is equal to the amount of subsequences of S that are still a TCS. Thus our problem reduces to finding the minimum and maximum amount of subsequences of a TCS that are still a TCS.

First, we observe the structure of a TCS that reduces an n-caterpillar. Let T be an n-caterpillar on the leaves $\{1, \dots, n\}$ and let $S = S_1 S_2 \dots S_{n-1}$ be a TCS that reduces T, where $S_k = (x_k, y_k)$ denotes the k-th element of S. Then pair S_i is of the form $(j, i+1)$ or $(i+1, j)$, where $j \leq i$ and $j \neq x_k$ for all $k < i$. The last requirement ensures the property of a TCS that every first coordinate does not appear as a second coordinate in the rest of the sequence. This happens in general, since the first coordinate is always removed from trees.

Note that the following subsequences are always a TCS.

- The original TCS S that reduces T is a subsequence and is a TCS. This gives 1 TCS;
- All single pairs are TCSs. This gives $(n-1)$ TCSs;
- All subsequences where the first $0 < k < n-2$ pairs are removed from the original TCS are TCSs. The properties of a TCS remain in such subsequences, since for the pairs $k+1, \dots, n-1$ the second coordinate of each ordered pair occurs as a first coordinate in some ordered pair later in the sequence, or as the second coordinate of the last pair. And just as in S, every leaf appearing as the first coordinate in such subsequence does not appear as a second coordinate in the rest of the sequence. This gives $(n-2)$ TCSs;
- All subsequences where the last $0 < k < n-2$ pairs are removed from the original TCS. Removing the last k pairs of a TCS remains the properties of a TCS, since every leaf appearing as the first coordinate still does not appear as a second coordinate in the rest of the sequence. Furthermore, the second coordinate of each ordered pair appears either as a first coordinate in the next ordered pair, or as the second coordinate of the last pair. This gives $(n-2)$ TCSs.

Summing all of these sequences gives $1 + (n-1) + (n-2) + (n-2)$ sequences. Since the first single pair is in the single pair category and the category where the last $n-2$ pairs are reduced, it is counted twice. The same holds for the last pair, when you remove the first $n-2$ pairs. So there are at least $1 + (n-1) + (n-2) + (n-2) - 2 = 3n - 6$ subsequences that are a TCS. So we have found a first

lower bound on the minimal number of subsequences that are a TCS. So $m \geq 3n - 6$.

To see if this is the highest lower bound on the minimum m and to find the maximum M, we will look at what happens if an ordered pair is removed from S in any other combination than the four combinations described above.

We first look at what happens if one removes an i-th pair of the form $(j, i+1)$. We assume that $i > 1$, because for removing the first pair, we have already shown that a TCS will be preserved. Here, j appears for the first time as a first coordinate in the sequence, because when a leaf appears as first coordinate in a pair, it is removed and cannot appear later in the sequence. Since $j \leq i$, j has appeared earlier in the sequence, in fact as a second coordinate of the $i-1$-th pair. By removing the pair $(j, i+1)$, j is a second coordinate in pair S_{i-1}, but no longer appears as a first coordinate later in the sequence. So the property of being a TCS is no longer met. So removing a pair (not the first) of the form $(j, i+1)$ will give a subsequence which is no longer a TCS. Note that removing all pairs after the removed pair will again give a TCS, because then j appears as a second coordinate in the last pair, namely pair S_{i-1}.

Now, we look what happens when removing a pair $(i+1, j)$.

Lemma 3.2. Let T be an n-caterpillar and let $S = S_1 S_2 \dots S_{n-1}$ be a TCS that reduces T, where $S_k = (x_k, y_k)$ denotes the k-th ordered pair of S. Then the subsequence obtained upon removing an ordered pair S_i of the form $(i+1, j)$, where $j \leq i$, is a TCS.

Proof. Let T be an n-caterpillar and let $S = S_1 S_2 \dots S_{n-1}$ be a TCS that reduces T, where $S_k = (x_k, y_k)$ denotes the k-th ordered pair of S. Let S' be the subsequence obtained by removing ordered pair $S_i = (i+1, j)$, where $j \leq i$. Note that $j \neq x_k$ for all $k < j$, since the first coordinate of an ordered pair is always removed. We assume that $i \leq n-2$, since we have already shown that upon removing the last pair of a TCS, a TCS is obtained. Note that the leaf $i+1$ is appearing for the first time in the sequence S in the pair S_i. So all pairs earlier in the sequence have their second coordinate appearing as a first coordinate in an ordered pair of S', or appearing in the last pair S_{n-1} of S'. Just as in the TCS S, we have that every leaf appearing as the first coordinate in S' does not appear as a second coordinate in the rest of the sequence. So upon removing the pair S_i, both requirements of a TCS are still met for all pairs in S'. Thus S' is a TCS. □

In particular, the TCS obtained in the setting of Lemma 3.2 corresponds to the tree where leaf $i+1$ is removed. Note that this lemma also holds when you remove multiple pairs of this form. Using these observations, we can find two TCSs which will give the minimum and maximum number of subtrees that can be reduced by one TCS.

First, take the TCS $S = (1,2)(2,3)\cdots(n-1,n)$. This TCS has only pairs of the form $(j, i+1)$, thus removing any of those pairs, apart from those at the start or the end, will no longer give a TCS. So the TCS subsequences obtained by one of the four actions defined above, are the only subsequences of S that are a TCS. So S has $3n-6$ subsequences that are TCSs. Thus the minimum number of subtrees that can be reduced by a TCS of an n-caterpillar is at most $3n-6$, and the lower bound obtained earlier is tight. Thus $m = 3n-6$.

To find the maximum M, we look at the TCS $S = (2,1)(3,1)\cdots(n,1)$. This TCS has all pairs of the form $(i+1, j)$, so by Lemma 3.2 we know that removing any of the pairs will still preserve a TCS. So all $2^{n-1}-1$ subsequences of S are a TCS, which gives a lower bound for maximum number of subtrees that can be reduced by a TCS, and thus we have $M \geq 2^{n-1}-1$. Note that the total number of subsequences of a TCS S gives an upper bound for the maximum $M \leq 2^{n-1}-1$, combined with the lower bound, we have $M = 2^{n-1}-1$. We combine these results in the following lemma, the proof follows from the reasoning above.

Lemma 3.3. Let T be an n-caterpillar. There exists a TCS for T which has exactly $3n-6$ subsequences that are TCSs. There exists a TCS for T which has $2^{n-1}-1$ subsequences that are TCSs. Every TCS for T has between $3n-6$ and $2^{n-1}-1$ subsequences that are TCSs.

Because this last sequence has the nice property of having only TCS subsequences, we give a general definition for this type of sequence.

Definition 3.2. A TCS is called a *1-fixed tree child sequence* of length $n-1$ if every second coordinate of each pair is leaf 1. We denote the 1-fixed TCS by $S_{\underline{1}}$.

Sequences can also be fixed on a higher leaf number, say leaf $i > 1$.

Definition 3.3. A TCS is called a *i-fixed tree child sequence* if pair $i+1$ and all later pairs in the sequence have leaf i as a second coordinate. We denote the i-fixed TCS by $S_{\underline{i}}$.

Remark. For $k > 1$, there exist 2^{k-2} different k-fixed TCSs, because for the first $k-1$ leaves there are no restrictions in the ordering of picking the cherries. Thus, by Lemma 3.1 we have 2^{k-2} partial TCSs that reduce the subcaterpillar rooted at $lca(\{1,\dots,k-1\})$. For the rest of the leaves, there is no choice in the ordering of the cherries, as leaf k has to be fixed.

Let T be a 6-caterpillar on taxa set $X = \{1,\dots,6\}$. An example of a 1-fixed TCS that reduces T is $(2,1)(3,1)(4,1)(5,1)(6,1)$. And an example of a 4-fixed TCS that reduces T is $(2,1)(3,1)(1,4)(5,4)(6,4)$. Note that $(1,2)(3,2)(2,4)(5,4)(6,4)$ is also a 4-fixed TCS that reduces T.

3.2. Minimal subtree covering number

Let T be an n-caterpillar. We know that with one TCS of length $n-1$, at most $M = 2^{n-1}-1$ subtrees can be reduced. So not all $|\mathcal{D}(T)| = 2^n - n - 1$ subtrees can be covered by one TCS. We want to know how many TCSs we at least need in order to reduce all subtrees in $\mathcal{D}(T)$. We call this number the *minimal subtree covering number* of T, denoted by $S(T)$.

Since the fixed TCSs reduce the maximal number of subtrees with one TCS, we first look at the subtrees that are covered by a fixed TCS.

Lemma 3.4. A 1-fixed TCS reduces all subtrees of an n-caterpillar that contain leaf 1.

Proof. Let T be an n-caterpillar. The 1-fixed TCS is $S = (2,1)(3,1)\cdots(n,1)$. Note that every subtree on two leaves with leaf 1 is reduced by a single pair of S. Moreover, S reduces all subtrees with leaf 1. All pairs $(i,1)$ where i is not in the subtree, do nothing. All other pairs $(j,1)$, where j is a leaf in the subtree, reduce the leaves j in ascending order.

□

This does not hold for an i-fixed TCS, because for example the 3-fixed TCS for an n-caterpillar

$$S_{\underline{3}} = (2,1)(3,1)(4,3)(5,3)\cdots(n,3)$$

does not reduce the cherry $(2,3)$, which is a subtree of the n-caterpillar. However, we can prove that it reduces all subtrees where 3 is the lowest leaf.

Lemma 3.5. Let T be an n-caterpillar and let T' be a subtree of T, with lowest leaf i, for some $1 \le i \le n$. Then the i-fixed TCS reduces T'.

Proof. Let T be an n-caterpillar and let T' be a subtree of T, with lowest leaf i, for some $1 \le i \le n$. Let $S_{\underline{i}}$ be an $i-$fixed TCS that reduces T. Note that T' is also a caterpillar, since every subtree of a caterpillar is a caterpillar. Because the leaves of a caterpillar are ordered from the least deep node to the deepest node, the lowest leaf i is the deepest node of T', and thus part of the only cherry in T'. Therefore, the first $i-1$ pairs of $S_{\underline{i}}$ do not affect T', since the first $i-1$ pairs of $S_{\underline{i}}$ contain leaves with labels fewer than i. All other pairs of $S_{\underline{i}}$ are of the form (j,i), where $j = i+1,\dots,n$. So leaf i never gets removed, and thus the sequence $S_{\underline{i}}$ reduces T' by removing all other leaves one by one, in ascending order. □

Using this result, we can give an upper bound on the minimal subtree covering number $S(T)$ for n-caterpillar T. We claim that with $n-1$ k-fixed TCSs, where $k = 1,\dots,n-1$, all subtrees of T can be reduced. This is true since every subtree has some lowest leaf i, which will be reduced by the i-fixed TCS. Thus that would give the upper bound $S(T) \le n-1$.

Theorem 3.1. Let T be a caterpillar on n leaves. The minimal subtree covering number is given by

$$S(T) = n-1.$$

Proof. Let T be a caterpillar on n leaves. Note that there exist $n-1$ cherries with leaf n that are all subtrees of T. Observe that no two of such cherries can be reduced by one TCS, since such cherry can only be reduced in the last pair of a TCS that reduces T. Therefore, we need at least $n-1$ different TCSs to reduce all subtrees in $\mathcal{D}(T)$, giving a lower bound $S(T) \geq n-1$. Together with the upper bound from above, we have that $S(T) = n-1$.

□

4

Double caterpillars

We can try to find similar results for different types of trees. In this chapter we study double caterpillars, or $(k, n-k)$-caterpillars.

Definition 4.1. Let T be a tree on n leaves, with root r. T is called a *double caterpillar* or the $(k, n-k)$*−caterpillar*, with $1 \leq k \leq n-1$, if the root has children p_k and p_{k+1} and if the subtrees rooted at p_k and p_{k-1} are caterpillars on k and $n-k$ leaves, respectively, where p_k and p_{k+1} denote the parents of leaves k and $k+1$, respectively.

Note that the $(n-1, 1)$-caterpillar, or equivalently the $(1, n-1)$-caterpillar, is the same as the 'regular' n-caterpillar. We order the leaves in a double caterpillar T from deepest node to the least deep node in the first branch $T_1 = \{1, \ldots, k\}$. And we order the leaves from the least deep node to the deepest node in the other branch T_2. In Figure 4.1 an example of a double caterpillar is shown, with the right leaf ordering.

4.1. How many TCSs reduce a double caterpillar?

With a recurrence relation, we can find the number of TCSs that reduce a double caterpillar.

Definition 4.2. Let T be a $(k, n-k)$-caterpillar. We let $N(T) = N(k; n-k)$ denote the number of minimal TCSs that reduce T.

Lemma 4.1. Let T be a $(k, n-k)$-caterpillar, with $3 \leq k+1 \leq n-1$. There are $N(k; n-k)$ TCSs that reduce T, where

$$N(k; n-k) = 2 \cdot N(k-1; n-k) + 2 \cdot N(k; n-k-1).$$

Proof. Let T be a $(k, n-k)$-caterpillar, with $3 \leq k+1 \leq n-1$. Recall that the leaves are ordered from 1 to n, where 1 is the deepest node and k the least deep node from the first k leaves. From the last $n-k$ leaves, $k+1$ is the least deep node and n is the deepest node. Thus, there are two cherries that can be picked, cherry $(1, 2)$ and cherry $(n-1, n)$. By picking cherry $(1, 2)$, the resulting tree is a $(k-1, n-k)$-caterpillar. By picking cherry $(n-1, n)$, an $(k, n-k-1)$-caterpillar is obtained. Recall that there are always two ordered pairs that can pick a cherry. Thus for picking either cherry, there are

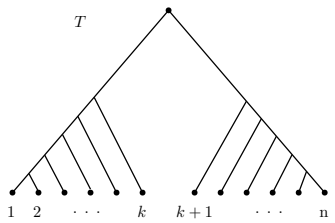


Figure 4.1: A $(k, n-k)$-caterpillar T, where $T_1 = \{1, \ldots, k\}$ and $T_2 = \{k+1, \ldots, n\}$.

two options. So in total there are $N(k;n-k) = 2 \cdot N(k-1;n-k) + 2 \cdot N(k;n-k-1)$ TCSs that reduce T.

□

Let T be a $(k,n-k)$-caterpillar. We discuss the base cases for Lemma 4.1. If $k = 1$ or if $k = n-1$, then T is a caterpillar on n leaves, thus $N(k;n-k) = 2^{n-1}$. With these corner cases, we get the following recurrence relation for $(2,n-2)$-caterpillars, which is equivalent to a $(n-2,2)$-caterpillar.

Remark. Let T be an $(2,n-2)$-caterpillar, for $n \geq 4$. The number of TCSs that reduce T is given by

$$N(2;n-2) = 2 \cdot N(2;n-3) + 2^{n-1}.$$

The recurrence relation from Lemma 4.1 can be extended to a closed form for the number of TCSs of a double caterpillar, making use of the binomial of Pascal's triangle.

Theorem 4.1. Let T be an $(k,n-k)$-caterpillar, with $1 \leq k \leq n-1$. The number of TCSs that reduce T is given by

$$N(k;n-k) = \binom{n-2}{k-1} \cdot 2^{n-1}.$$

Proof. We prove the theorem by induction on the number of leaves n. For the base case we consider a tree T on $n = 4$ leaves. This is a $(2,2)$-caterpillar. All TCSs that reduce T consist of 3 pairs. For the first pair there are 4 cherries to choose, $(1,2),(2,1),(3,4)$ or $(4,3)$. After picking the first cherry, there is only one cherry left, giving two choices for the second pair. For the third pair, there is again one cherry to pick, which gives 2 choices. After picking 3 cherries, the tree is reduced, so in total there are $4 \cdot 2 \cdot 2 = 16$ TCSs that reduce T. This is exactly $\binom{4-2}{2-1} \cdot 2^{4-1} = 2 \cdot 8 = 16$.

For the induction hypothesis, we assume that the theorem holds for all $(k,m-k)$-caterpillars on $m < n$ leaves. Let T be a $(k,n-k)$-caterpillar on n leaves. From Lemma 4.1, we know that $N(k;n-k) = 2 \cdot N(k-1;n-k) + 2 \cdot N(k;n-k-1)$. Since the $(k-1,n-k)$- and $(k,n-k-1)$-caterpillars have $n-1$ leaves, we can use the induction hypothesis and we find the following.

$$\begin{aligned} N(k;n-k) &= 2 \cdot [N(k-1;(n-1)-(k-1)) + N(k;(n-1)-k)] \\ &= 2 \cdot \left[\binom{(n-1)-2}{(k-1)-1} \cdot 2^{n-2} + \binom{(n-1)-2}{k-1} \cdot 2^{n-2} \right] \\ &= 2 \cdot \left[\binom{n-3}{k-2} \cdot 2^{n-2} + \binom{n-3}{k-1} \cdot 2^{n-2} \right] \\ &= \binom{n-2}{k-1} \cdot 2^{n-1} \end{aligned}$$

Thus, by induction we have shown that for all $n \geq 4, k \geq 2$ with $n-k \geq 2$, the number of TCSs that reduce a $(k,n-k)$-caterpillar is given by

$$N(k;n-k) = \binom{n-2}{k-1} \cdot 2^{n-1}.$$

□

4.2. Minimal subtree covering number

For n-caterpillars we found a minimum subtree covering number given by $S(T) = n-1$, according to Theorem 3.1. In this section we use this result and try to find a minimum subtree covering number for double caterpillars.

Let T be a $(k,n-k)$-caterpillar with branches $T_1 = \{1,\ldots,k\}$ and $T_2 = \{k+1,\ldots,n\}$. We try to find minimal number of TCSs that reduce all subtrees in $\mathcal{D}(T)$. Note that cherries (x,y), which form subtrees in $\mathcal{D}(T)$, where $x \in T_1$ and $y \in T_2$, need to be reduced at the end of a TCS, since they can only appear as a cherry at the end after picking all other leaves. There are $k(n-k)$ such cherries in $\mathcal{D}(T)$, giving a lower bound on the minimum covering number $S(T) \geq k(n-k)$. In the next theorem, we show that this bound is always attained.

Theorem 4.2. Let T be a $(k, n-k)$-caterpillar, with $1 \leq k \leq n-1$. The minimum subtree covering number of T is given by

$$S(T) = k(n-k).$$

Proof. We prove the theorem by induction on the number of leaves, n. For the base case, we have $n = 2$. This gives a $(1,1)$-caterpillar, which is just a cherry, with only itself as a subtree, so it can be reduced by only 1 TCS. Thus $S(T) = 1$.

For the induction hypothesis, we assume the theorem holds for all $(k, m-k)$-caterpillars on $m < n$ leaves. Let T be a $(k, n-k)$-caterpillar. If $k = 1$, we have an n-caterpillar, that has $n-1$ TCSs that reduce all subtrees of T, by Theorem 3.1.

Suppose $k \geq 2$. Upon removing leaf 1, which is equivalent to picking the cherry $(1,2)$, we obtain a $(k-1, n-k)$ caterpillar on $n-1$ leaves and we call this double caterpillar T'. By the induction hypothesis, we know that $S(T') = (k-1)(n-k)$. So there are $(k-1)(n-k)$ sequences, $S_1, \ldots, S_{(k-1)(n-k)}$, that reduce all subtrees in $\mathcal{D}(T')$. Note that these sequences with cherry $(1,2)$ as prefix, $(1,2)S_i$, for $i = 1, \ldots, (k-1)(n-k)$, are TCSs for T reducing all subtrees in $\mathcal{D}(T')$ and all subtrees in $\mathcal{D}(T) \backslash \mathcal{D}(T')$ containing leaves 1 and 2.

Thus, the only subtrees of T that need to be reduced are those in $\mathcal{D}(T) \backslash \mathcal{D}(T')$ which do not contain leaf 2. We claim that these subtrees can be reduced by $n-k$ extra TCSs. Note that the cherries $(1, k+1), (1, k+2), \ldots, (1, n)$ are subtrees in $\mathcal{D}(T)$ that still need to be reduced. The only way that these subtrees can be reduced by a TCS of T is by having the corresponding cherry as the last pair of the TCS. Therefore the $n-k$ TCSs need to start with $(2,1) \cdots (k,1)$, in order for leaf 1 to remain for the last pair. After applying partial TCS $P = (2,1) \cdots (k,1)$ to T, we obtain a caterpillar on $n-k+1$ leaves, on the leaves $1, k+1, \ldots, n$. By Theorem 3.1, we need $n-k$ sequences, $R_1, \ldots, R_{n-k}$, to reduce all subtrees of this caterpillar. By appending the partial TCS P to those $n-k$ partial TCSs, that is PR_i, for $i = 1, \ldots, n-k$, we obtain $n-k$ different TCSs for T. One can verify that these TCSs reduce all subtrees in $\mathcal{D}(T) \backslash \mathcal{D}(T')$ not containing leaf 2. So we found $(k-1)(n-k) + (n-k) = k(n-k)$ TCSs that reduce all subtrees of T. This gives us an upper bound for $S(T)$. Combined with the lower bound from above, we obtain $S(T) = k(n-k)$. Thus by induction, for all $n \geq 2$, a $(k, n-k)$-caterpillar has $k(n-k)$ TCSs that reduce all its subtrees.

□

5 Trees

With the results from the (double) caterpillars, we can try to answer our two research questions for general binary trees. In order to do this, we look at the structure of trees in terms of caterpillars, using partitions on the leaves.

5.1. Caterpillar decomposition

Trees can have varying structures. In order to use the results we found for (double) caterpillars we use an algorithm to give a tree a suitable structure to work with. We do this here by using the so-called *Heavy-Light-algorithm* introduced by Sleator and Tarjan [5] for the link/cut tree structure and by Harel and Tarjan [6] for their data structure for finding nearest common ancestors. The algorithm decomposes a tree into heavy-edge paths. For every non-leaf node, its two outgoing edges are labelled heavy or light, depending on the number of its descendants. The heavy edge is the edge that has the greatest number of descendants. When both edges have the same number of descendants, the heavy edge is selected arbitrarily. The selected heavy-edges form heavy-edge paths, decomposing the tree into paths. Note that every edge of the tree is either a heavy edge or a light edge. Using the heavy-edge paths as the spines of sub-caterpillars gives a decomposition of caterpillars for the given tree. This way we can redraw a tree and find *maximal sub-caterpillars* and relabel the leaves ascending from left to right. In Figure 5.1 the heavy-light algorithm is executed on a tree as an example.

Definition 5.1. Let T be a tree on n leaves. A set of leaves $C = \{x_1, \dots, x_p\}$ is called a *sub-caterpillar* of T if the subtree of T rooted at the $lca(C)$ is a caterpillar on the set of leaves C. We denote the number of leaves in the sub-caterpillar by $|C|$.

Definition 5.2. Let T be a tree on n leaves. A sub-caterpillar C is called a *maximal sub-caterpillar* of T if C is no subset of any other sub-caterpillar of T.

Remark. Note that the leaves of a tree can be uniquely partitioned by its maximal sub-caterpillars, $C_1, \dots, C_k$, since every leaf appears in exactly one maximal sub-caterpillar. We call the set of all maximal sub-caterpillars of a tree T the *maximal sub-caterpillar decomposition* of T.

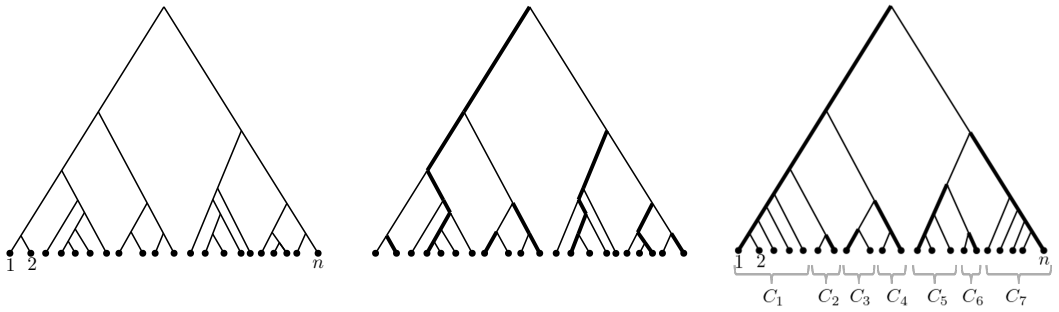
5.2. Minimal subtree covering number

For caterpillars we found a minimal subtree covering number. With the decomposition of trees into maximal sub-caterpillars, we can also attempt to find the minimal subtree covering number for more general trees on n leaves.

Lemma 5.1. Let T be a tree on n leaves. Then $S(T) \geq n - 1$.

Proof. Let T be a tree on n leaves. Then there are $n-1$ different cherries (x, n) that need to be reduced, where $x = 1, \dots, n-1$. Just as in Theorem 3.1, we see that no two of these cherries can be picked in one TCS. Therefore, for all $n-1$ cherries (x, n), we need a new TCS. This gives the lower bound $S(T) \geq n - 1$.

□



(a) Tree T before applying the HL-algorithm.

(b) The heavy edges of T, according to the HL-algorithm are fattened.

(c) Maximal sub-caterpillar decomposition of T after applying the HL-algorithm and relabelling the leaves.

Figure 5.1: Heavy-Light algorithm applied to a tree T on n leaves in three steps.

Lemma 5.2. Let T be a tree on n leaves, with maximal sub-caterpillar decomposition $\{C_1, \dots, C_k\}$. Then

$$S(T) \geq \prod_{i=1}^{k} |C_i|.$$

Proof. Let T be a tree on n leaves, with maximal sub-caterpillar decomposition $\{C_1, \dots, C_k\}$. There are $m = \prod_{i=1}^{k} |C_i|$ different subtrees of T on k leaves, where each leaf is from a different maximal caterpillar C_i. Call these subtrees $T_1, \dots, T_m$. Note that no two subtrees T_i and T_j $(i \neq j)$, can be reduced by the same TCS for T, since within every maximal sub-caterpillar, all leaves can only be picked by a pair with leaves in that sub-caterpillar until the sub-caterpillar is reduced to a single leaf, from which point that leaf can form a cherry with a leaf from another maximal sub-caterpillar. So only one leaf per maximal sub-caterpillar C_i can form a pair in a TCS with a leaf from another maximal sub-caterpillar C_j, for $i \neq j$. Therefore, we need at least $m = \prod_{i=1}^{k} |C_i|$ different TCSs in order to reduce all subtrees of T. This gives us the lower bound $S(T) \geq \prod_{i=1}^{k} |C_i|$. □

These lower bounds for the minimal covering number might be tight for certain types of trees.

Conjecture 5.1. Let T be a tree on n leaves with maximal sub-caterpillar decomposition $\{C_1, \dots, C_k\}$, where $|C_j| = 1$ for some $1 \leq j \leq k$ and $|C_i| \geq 2$ for all $i \in \{1, \dots, j-1, j+1, \dots, k\}$. Then

$$S(T) = \prod_{i=1}^{k} |C_i|.$$

Conjecture 5.1 can perhaps be proved by induction on the number of maximal sub-caterpillars k.

Conjecture 5.2. Let T be a tree on n leaves, where $\{C_1, \dots, C_k\}$ is the maximal sub-caterpillar decomposition of T. If the second branch is $T_2 = C_k = \{n\}$ and $|C_1|, \dots, |C_{m-1}| \geq 2$ and $|C_m| = \dots = |C_k| = 1$ for some $1 \leq m \leq k$, then

$$S(T) = \max\left\{\prod_{i=1}^{k} |C_i|, n-1\right\}.$$

We state a proof for this conjecture, assuming Conjecture 5.1 is true, in Appendix A.

Remark. For integers $z_i \geq 2$ we have

$$\prod_{i=1}^{k} z_i \geq \sum_{i=1}^{k} z_i.$$

With this result we come to the following conjecture for trees with only maximal sub-caterpillars on more than 2 leaves.

Conjecture 5.3. Let T be a non-caterpillar tree on n leaves, with maximal sub-caterpillar decomposition $\{C_1, \dots, C_k\}$, with $|C_i| \geq 2$ for all $i \in \{1, \dots, k\}$. Then

$$S(T) = \prod_{i=1}^{k} |C_i|.$$

We state a proof for this conjecture, assuming Conjecture 5.1 is true, in Appendix A.

Conjecture 5.4. Let T be a tree on n leaves, with $\{C_1, \dots, C_k\}$ as its maximal sub-caterpillar decomposition. Let the two branches of T be $T_1 = \cup_{i=1}^{j} C_i$ and $T_2 = \cup_{i=j+1}^{k} C_i$, where $|C_1|, \dots, |C_{m-1}|, |C_{l+1}|, \dots |C_k| \geq 2$ and $|C_m| = \cdots = |C_l| = 1$ for some m, l with $1 \leq m \leq l \leq k$. Then

$$S(T) = \max\left\{\prod_{i=1}^{k-1} |C_i|, |T_1||T_2|\right\}.$$

Perhaps this could be proved similarly as in the (example) proof of Conjecture 5.2.

5.3. How many TCSs reduce a tree?

In Chapters 3 and 4 we found the number of TCSs that reduce a (double) caterpillar. In this section we try to expand these results to more general trees.

Let T be a tree on n leaves, with maximal sub-caterpillar decomposition $\{C_1, \dots, C_k\}$. Let $T_1 = \cup_{i=1}^{m} C_i$ and $T_2 = \cup_{i=m+1}^{k} C_i$ be the two branches of T, for some $1 \leq m \leq k$. We denote the number of TCSs that reduce T by $N(T) = N(|C_1|, \dots, |C_m|; |C_{m+1}|, \dots, |C_k|)$.

In Lemma 3.1 we proved that when $k = n$, that is T is an n-caterpillar, there exist $N(n-1; 1) = 2^{n-1}$ different TCSs that reduce T. In Theorem 4.1 we proved that for $k = 2$, that is, T is a double caterpillar with $|C_1| = p, |C_2| = n - p$, for some $1 \leq p \leq n-1$, there exist $N(p; n-p) = \binom{n-2}{p-1} \cdot 2^{n-1}$ TCSs that reduce T.

5.3.1. How many TCSs reduce a tree on 3 maximal sub-caterpillars?

The next step is to consider the case when $k = 3$. Without loss of generality, we can assume that $T_1 = C_1 \cup C_2$ and $T_2 = C_3$. Say that $|C_1| = p, |C_2| = q$ and $|C_3| = r$, where $p \geq q \geq 2$ and $r \geq 1$. In order to reduce T we can start by picking a cherry from either C_1, C_2 or C_3. For each cherry there are two ordered pairs to chose from, giving the following recurrence relation.

$$N(p, q; r) = 2 \cdot N(p-1, q; r) + 2 \cdot N(p, q-1; r) + 2 \cdot N(p, q; r-1).$$

There are three base cases to consider for $N(p, q; r)$.

- If $p = 1$, C_1 and C_2 are no longer maximal sub-caterpillars, since the single leaf from C_1 forms a sub-caterpillar with the leaves from C_2. Therefore we have a new maximal sub-caterpillar decomposition $\{C_2', C_3\}$, where $C_2' = C_1 \cup C_2$, with $|C_2'| = q + 1$. Thus we have a $(q+1, r)-$caterpillar and therefore $N(p, q; r) = N(q+1; r) = 2^{q+r}\binom{q+r-1}{q}$.
- If $q = 1$, C_1 and C_2 are no longer maximal sub-caterpillars, since the single leaf from C_2 forms a sub-caterpillar with the leaves from C_1. Therefore we have a new maximal sub-caterpillar decomposition $\{C_1', C_3\}$, where $C_1' = C_1 \cup C_2$, with $|C_1'| = p + 1$. Thus we have a $(p+1, r)-$caterpillar and therefore $N(p, q; r) = N(p+1; r) = 2^{p+r}\binom{p+r-1}{p}$.
- If $r = 1$, then C_3 does not contain a reducible pair. There are $N(p; q)$ different choices for reducing all leaves from C_1 and C_2 to a single leaf. After doing so, that leaf forms a cherry with the single leaf from C_3, for which there are 2 choices to pick it. So in total, we have $N(p, q; r) = 2N(p; q) = 2^{p+q}\binom{p+q-2}{p-1}$ TCSs that reduce the tree.

5.3.2. How many TCSs reduce a tree on k maximal sub-caterpillars?

We can extend the results for the trees on three maximal sub-caterpillars to a general recurrence relation for trees on k maximal sub-caterpillars.

Lemma 5.3. Let T be a tree on n leaves with maximal sub-caterpillar decomposition $\{C_1, \dots, C_k\}$. Let $T_1 = \bigcup_{i=1}^{m} C_i$ and $T_2 = \bigcup_{i=m+1}^{k} C_i$ be the two branches of T, for some $1 \leq m \leq k$. The number of TCSs that reduce T is given by the following recurrence relation:

$$\begin{aligned} N(T) &= N(|C_1|, \dots, |C_m|; |C_{m+1}|, \dots, |C_k|) \\ &= 2 \cdot [N(|C_1| - 1, \dots, |C_m|; |C_{m+1}|, \dots, |C_k|) + N(|C_1|, |C_2| - 1, \dots, |C_m|; |C_{m+1}|, \dots, |C_k|) \\ &\quad + \dots + N(|C_1|, \dots, |C_m|; |C_{m+1}|, \dots, |C_k| - 1)]. \end{aligned}$$

There are a few base cases that we need to consider for $N(|C_1|, \dots, |C_m|; |C_{m+1}|, \dots, |C_k|)$.

- $|C_2| = \dots = |C_j| = 1$ for some $j \in \{2, \dots, m\}$. $\bigcup_{i=1}^{j} C_i$ is a new maximal caterpillar on $|C_1| + j - 1$ leaves. Thus $N(|C_1|, \dots, |C_m|; |C_{m+1}|, \dots, |C_k|) = N(|C_1| + j - 1, |C_{j+1}|, \dots, |C_m|; |C_{m+1}|, \dots, |C_k|)$
- $|C_j| = \dots = |C_{k-1}| = 1$ for some $j \in \{m+1, \dots, k-1\}$. $\bigcup_{i=j}^{k} C_i$ is a new maximal caterpillar on $|C_k| + k - i$ leaves. Thus $N(|C_1|, \dots, |C_m|; |C_{m+1}|, \dots, |C_k|) = N(|C_1|, \dots, |C_m|; |C_{m+1}|, \dots, |C_{j-1}|, |C_k| + k - j)$.
- $|C_i| = 1$ for some $i \in \{1, \dots, n\}$. The maximal sub-caterpillar C_i has only one leaf, thus it does not contain a cherry that can be picked. Therefore, we define $N(n_1, \dots, n_m; n_{m+1}, \dots, n_k) = 0$ if $n_j = 1$ for any $j \in \{1, \dots, n\}$.

5.3.3. Can we find a closed form for $N(T)$?

In Chapter 4, we found a closed form for the recurrence relation on the number of TCSs that reduce a double caterpillar. We want to find a closed form for the recurrence relation of the number of TCSs that reduce general trees. In the previous subsection we saw that we have many base cases, since the maximal sub-caterpillar decomposition of a tree may change after removing a leaf. Therefore it is hard to find a general closed form for the number of TCSs that reduce a tree. However, in this subsection we try to give a feeling on what the closed form would look like by giving a few examples. In order to give these examples, we create an auxiliary graph for a tree, by using *cherry covers*, which was introduced by van Iersel et al. in 2021 [7].

Definition 5.3. Let T be a tree on n leaves. We call a pair of edges $K = \{p_x x, p_x y\}$ a *cherry shape* if x and y are vertices in T, with $p_x = p_y$.

Definition 5.4. Let T be a tree on n leaves, with edges E. We call $K_1, \dots, K_m$ a *cherry cover* of T if K_i are cherry shapes in T and form a partition on the edges E of T.

The following examples show how we can use the results of a double caterpillar, as shown in Example 5.1, for a tree that contains that double caterpillar, as shown in Example 5.2.

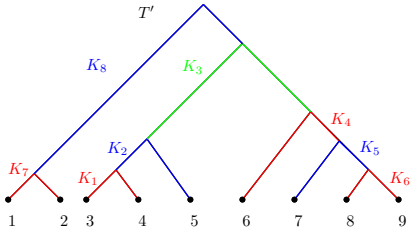
Example 5.1. Let T be a $(3, 4)$-caterpillar. In Figure 5.2a, a cherry cover for T is shown. In Figure 5.2b, an auxiliary tree A is made for T using the cherry shapes from Figure 5.2a, as the vertices of the tree.

Recall that a TCS S that reduces T has 6 pairs, as T has 7 leaves. Every pair in S corresponds with a vertex from tree A. A cherry shape K_i can only occur as a pair in S after all the vertices K_j that are below K_i in A, have been picked earlier in the sequence.

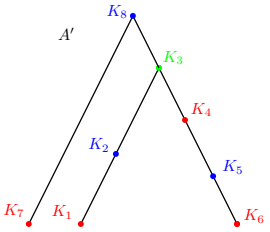
(a) Cherry cover for $(3, 4)$-caterpillar T.

(b) Auxiliary tree A for cherry cover of T.

Figure 5.2: Cherry cover for $(3, 4)$-caterpillar with its auxiliary tree.



(a) Cherry cover for tree T'.



(b) Auxiliary tree A' for cherry cover of T'.

Figure 5.3: Cherry cover for tree T' with its auxiliary tree A'.

An example of a TCS that reduces T is $S_1 = (2,1)(7,6)(3,1)(6,5)(4,5)(5,1)$. In terms of cherry shapes, we have the ordering $K_1, K_6, K_2, K_5, K_4, K_3$ that corresponds with S_1. Note that K_3 always has to be in the last position, as all other cherry shapes are below K_3 in A. Furthermore, K_1 always has to be before K_2 and K_6 has to appear before K_5, which has to be before K_4. These restrictions give the following results. For the sixth element of a TCS S that reduces T, we have no other choice than the pair that corresponds with K_3. That leaves us with 5 positions to divide the other cherry shapes in S. Note that when we choose a cherry shape for the second and the fourth position, the other positions will automatically be divided by the order in which the cherry shapes have to be placed. Thus we have $\binom{5}{2}$ choices for placing the cherry shapes. This corresponds with the binomial coefficient in the number of TCSs that reduce T, which is given by $N(T) = 2^6\binom{5}{2}$, according to Theorem 4.1. The term 2^6 comes from the fact that we have 2 choices for picking each cherry, and we have 6 cherries that we pick in the sequence, resulting in 2^6 choices in total after picking the order of the cherry shapes.

Example 5.2. Let T' be a tree on 9 leaves, with maximal sub-caterpillar decomposition $\{\{1,2\},\{3,4,5\},\{6,7,8,9\}\}$, as illustrated in Figure 5.3a. Note that tree T from Example 5.1 is a subtree of T'. T' is not a double caterpillar, so we do not know a formula for the number of TCSs that reduce T'. However, using the cherry cover and auxiliary tree A', as illustrated in Figure 5.3, we can give an educated guess on the number of TCSs that reduce T'.

Let S be a TCS that reduces T'. We know that S has length 8, as T' has 9 leaves. Furthermore, from A' we know that the last pair has to correspond with cherry shape K_8. Note that for cherry shape K_7, there are no other restrictions on the position than that it has to be before K_8. Thus, K_7 can be placed in either 7 positions, thus we have $\binom{7}{1}$ choices for placing K_7. Once K_7 is placed somewhere, then we have 6 positions left to divide in S. Note that for these 6 positions, we have the same restrictions as in Example 5.1, thus we have $\binom{5}{2}$ choices for placing the cherry shapes $K_1, \ldots, K_6$. So in total we have $\binom{7}{1}\binom{5}{2}$ choices for ordering all the cherry shapes with respect to A'. Thus, using the same logic as in Example 5.1, we would have $N(T') = 2^8\binom{7}{1}\binom{5}{2}$.

The problem of placing the cherry shapes in the right order as illustrated in these examples can be viewed as a special case of the Job Scheduling Problem as described by Lawler et al. in Chapter 9 of Logistics of Production and Inventory [8]. The jobs that need to be performed in the Optimal Job Scheduling problem in this case would be the vertices in the auxiliary trees, which correspond with the cherry shapes. There is one ”machine” that can perform the jobs, and they have to be done in particular order, such that each job is done after all jobs below it are covered. The jobs have to be done in exactly $n-1$ steps. Therefore, an enumeration of the sequences can lead to an advancement in a seemingly different field.

6

Conclusion and discussion

In this thesis we explored an enumeration problem and an optimization problem on the number of TCSs for phylogenetic binary trees. With results from pretty structured trees such as caterpillars and double caterpillars we tried to generate a formula for the number of TCSs, $N(T)$, that reduce a tree T on n leaves and to find the minimum number of TCSs, $S(T)$, that we need to reduce all subtrees of a given tree T on n leaves.

First we summarize our results for the enumeration problem.

Input: A binary phylogenetic tree T on n leaves. **Question:** How many different TCSs can reduce T?

In Chapter 3 we have found the number of TCSs that reduce a caterpillar with Lemma 3.1. For a caterpillar T on n leaves, there exist $N(T) = 2^{n-1}$ different TCSs that reduce T.

In Chapter 4 we found a recurrence relation on the number of TCSs that reduce a double caterpillar with Lemma 4.1. The base case for this recurrence relation was the number of TCSs for an $n-$caterpillar $N(n-1,1) = N(1,n-1) = 2^{n-1}$. With these results, we found a closed form for the number of TCSs $N(T)$ that reduce a double caterpillar in Lemma 4.1. For a $(k,n-k)$-caterpillar T, there exist $N(T) = N(k,n-k) = 2^{n-1}\binom{n-2}{k-1}$ different TCSs that reduce T.

In Chapter 5 we tried to generalize the results from Chapters 3 and 4 for general binary trees. In Lemma 5.3 we found a similar recurrence relation as in Lemma 4.1 for the number of TCSs that reduce a tree. We have not yet found a closed form for the number of TCSs, $N(T)$, that reduce a general binary tree. Future research, with programming and enhancing the thoughts based on cherry covers described in Examples 5.1 and 5.2 together with the Job Scheduling Problem, could perhaps lead to a closed form for the number of TCSs that reduce a tree.

On the other hand, the optimization problem consisted of finding the minimal number of TCSs that are needed to reduce all subtrees of a binary tree on n leaves. We called this number the minimal covering number for a tree T, denoted by $S(T)$.

Input: A binary phylogenetic tree T on n leaves. **Question:** How many TCSs do we need to reduce all subtrees of T?

Just like with the enumeration problem, we first found results for caterpillars in Chapter 3. In Lemma 3.1 we found that with $S(T) = n-1$ TCSs we can reduce all subtrees of a caterpillar T on n leaves. This result was found by introducing fixed TCSs, that have the nice property of reducing all subtrees that have the leaf which is fixed on as its lowest leaf, as found in Lemma 3.5.

In Chapter 4, we found the minimal covering number for double caterpillars. In Theorem 4.2, we showed that the minimal covering number for a $(k,n-k)$-caterpillar is given by $S(T) = k(n-k)$. In the proof of this theorem, we again used fixed TCSs.

In Chapter 5, we used the Heavy-Light algorithm to decompose general trees into maximal sub-caterpillars, in order to use the results from double caterpillars that we found in Chapter 5. For trees with

particular assumptions on the maximal sub-caterpillar decomposition, we found multiple conjectures on the minimal covering number in Conjectures 5.2 and 5.3. If Conjecture 5.1 can be proven, the proofs that are stated in Appendix A, can be used to confirm that

$$S(T) = \prod_{i=1}^{k} |C_i|,$$

for trees with only maximal sub-caterpillars on at least 2 leaves.

For general trees, without restrictions on the maximal sub-caterpillar decomposition, no concrete formula has been found.

With the results of this thesis we have proposed first steps in solving the enumeration and optimization problems associated to cherry-picking sequences for trees. With future research, similar results may be expanded from binary trees to non-binary trees and eventually even to general phylogenetic networks.

A

Appendix

Here, we state a proof of Conjectures 5.2 and 5.3, assuming that Conjecture 5.1 is true.

Conjecture (**5.2**)**.** Let T be a tree on n leaves, where $\{C_1, \dots, C_k\}$ is the maximal sub-caterpillar decomposition of T. If the second branch is $T_2 = C_k = \{n\}$ and $|C_1|, \dots, |C_{m-1}| \geq 2$ and $|C_m| = \dots = |C_k| = 1$ for some $1 \leq m \leq k$. Then

$$S(T) = \max\left\{\prod_{i=1}^{k} |C_i|, n-1\right\}.$$

Proof. We state a proof of the conjecture, assuming Conjecture 5.1 is true. We prove the conjecture by induction on the number of single-leaf maximal sub-caterpillars $k - m + 1$.

For the base case we consider the case where $m = k$. Let T be a tree where $\{C_1, \dots, C_k\}$ is the maximal sub-caterpillar decomposition of T, then we have $|C_i| \geq 2$ for $i = 1, \dots, k-1$ and $C_k = \{n\} = T_2$. By Conjecture 5.1 we know that $S(T) = \prod_{i=1}^{k} |C_i|$. Since $|C_i| \geq 2$ for all $i \neq j$, we have $\prod_{i=1}^{k} |C_i| = \prod_{i=1, i\neq j}^{k} |C_i| \geq \sum_{i=1, i\neq j}^{k} |C_i| = n - 1$. Thus $S(T) = \max\left\{\prod_{i=1}^{k} |C_i|, n-1\right\}$.

For the induction hypothesis, we assume that the conjecture holds for all trees with less than $k-m+1$ single-leaf maximal sub-caterpillars. Let T be a tree on n leaves with $\{C_1, \dots, C_k\}$ as its maximal sub-caterpillar decomposition, with $|C_1|, \dots, |C_{m-1}| \geq 2$, $|C_m| = \dots = |C_k| = 1$ and $T_2 = C_k = \{n\}$. Upon removing leaf n, we obtain a tree T' on $n-1$ leaves, where $T'_2 = C_{k-1} = \{n-1\}$, with $\{C_1, \dots, C_{k-1}\}$ as its maximal sub-caterpillar decomposition. Note that these are the same maximal caterpillars as for T. T' has $k - m$ single-leaf maximal sub-caterpillars, thus by the induction hypothesis, we have that $S(T') = \max\left\{\prod_{i=1}^{k-1} |C_i|, n-2\right\}$. We consider two cases:

1) $n - 2 \geq \prod_{i=1}^{k-1} |C_i|$, hence $S(T') = n - 2$;

2) $\prod_{i=1}^{k-1} |C_i| > n - 2$, hence $S(T') = \prod_{i=1}^{k-1} |C_i|$.

If case 1) is true, we have $n-2$ sequences $S_1, \dots, S_{n-2}$ that reduce all subtrees in $\mathcal{D}(T')$. The only subtrees of T that are not reduced by those sequences are the subtrees that contain leaf n. Note that leaf n is always the least deep leaf of such subtree. In order for all $n-1$ cherries containing leaf n to be reduced, we need at least $n-1$ different TCSs, since no two such cherries can appear as a pair in one TCS. Since $C_{k-1} = \{n-1\}$, leaf $n-1$ is the least deep leaf of T' and is therefore always reduced in the last pair of a TCS. Thus, all $n-2$ cherries containing leaf $n-1$ appear as the last pair of exactly one sequence S_i. By adjusting the sequences S_i such that the last pair is of the form $(n-1, x_i)$, where $x_i = 1, \dots, n-2$, we obtain $n-2$ sequences $\bar{S}_1, \dots, \bar{S}_{n-2}$. By adding (n, x_i) as a suffix to the partial TCSs $\bar{S}_i$, we obtain $n-2$ TCSs $\bar{S}_1(n, x_1), \dots, \bar{S}_{n-2}(n, x_{n-2})$ that reduce all subtrees in $\mathcal{D}(T)$, except the cherry $(n-1, n)$. In order to reduce this cherry, we need one more TCS that is fixed on leaf $n-1$ and $(n, n-1)$ is the last pair. One can verify that with those $n-1$ TCSs all subtrees in $\mathcal{D}(T)$ are reduced, giving us

the upper bound

$$S(T) \leq n-1 = \max\left\{\prod_{i=1}^{k-1} |C_i|, n-1\right\}.$$

If case 2) is true, we have $a = \prod_{i=1}^{k-1} |C_i| \geq n-1$ TCSs $R_1, \dots, R_a$, that reduce all subtrees in $\mathcal{D}(T')$. Just like in case 1), we adjust the last pair of the sequences R_i, such that every leaf $1, \dots, n-1$ appears at least once as the last element of the last pair in a sequence $\bar{R}_i$. Since we have $a \geq n-1$ sequences, all $n-1$ leaves can appear as the last element at least once. Thus we are able to remove all cherries involving n. Note that n is always the least deep node in a subtree of T, it can be reduced at the end of a TCS. Upon adding pairs (n, x_i) as a suffix to all $\bar{R}_i$, we obtain a sequences $\bar{R}_1(n, x_1), \dots, \bar{R}_a(n, x_a)$ that reduce all subtrees in $\mathcal{D}(T)$, including all cherries containing leaf n. This gives us an upper bound

$$S(T) \leq a = \prod_{i=1}^{k-1} |C_i| = \max\left\{\prod_{i=1}^{k-1} |C_i|, n-1\right\}.$$

So in both cases we found the upper bound

$$S(T) \leq \max\left\{\prod_{i=1}^{k-1} |C_i|, n-1\right\}.$$

Combined with the lower bounds from Lemmas 5.2 and 5.1, we have

$$S(T) = \max\left\{\prod_{i=1}^{k-1} |C_i|, n-1\right\}.$$

So by induction we have shown that for all trees T with $k - m + 1 \geq 1$ single-leaf maximal sub-caterpillars, such that one child of the root is a leaf, we have

$$S(T) = \max\left\{\prod_{i=1}^{k-1} |C_i|, n-1\right\}.$$

□

Conjecture (**5.3**)**.** Let T be a non-caterpillar tree on n leaves, with maximal sub-caterpillar decomposition $\{C_1, \dots, C_k\}$, with $|C_i| \geq 2$ for all $i \in \{1, \dots, k\}$. Then

$$S(T) = \prod_{i=1}^{k} |C_i|.$$

Proof. We state an example of a proof for the conjecture, assuming that Conjecture 5.1 is true. We prove the conjecture by induction on the number of leaves n.

For the base case we consider the balanced tree T on 4 leaves. The maximal sub-caterpillar decomposition of T is $\{C_1, C_2\}$, where $C_1 = \{1, 2\}$ and $C_2 = \{3, 4\}$. By Lemma 5.2 we know that we need at least $2 \cdot 2 = 4$ TCSs to reduce all subtrees of T. One can verify that with the following four TCSs we can reduce all subtrees of T.

$$S_1 = (1,2)(3,4)(2,4)$$
$$S_2 = (1,2)(4,3)(2,3)$$
$$S_3 = (2,1)(3,4)(1,4)$$
$$S_4 = (2,1)(4,3)(1,3)$$

Thus we have

$$S(T) = 4 = \prod_{i=1}^{2} |C_i|.$$

For the induction hypothesis, we assume that the conjecture holds for all trees on less than n leaves. Let T be a tree on n leaves, with maximal sub-caterpillar decomposition $\{C_1, \dots, C_k\}$ with $|C_i| \geq 2$ for all i.

We consider three cases, where for the third case, we consider two subcases 3a) and 3b).

1) $|C_1| \geq 3$ or we can relabel the leaves of T such that $|C_1| \geq 3$;
2) $|C_i| = 2$ for all $i \in \{1, \dots, n\}$ and branch $T_1 = \{C_1\}$, or we can relabel such that $T_1 = \{C_1\}$.
3) $|C_i| = 2$ for all $i \in \{1, \dots, n\}$ and $|T_1| \geq 4$
   a) We can relabel the leaves of T, such that removing a leaf from $C_1 = \{1, 2\}$ does not change the maximal sub-caterpillar decomposition.
   b) Upon removing any leaf of T, say leaf $1 \in C_1$ the maximal sub-caterpillar decomposition changes to $\{C_2', C_3, \dots, C_k\}$, where C_2' is a triplet on the leaves $\{2, 3, 4\}$.

Assume we are in case 1). Remove leaf 1 from C_1 and call the resulting tree T'. The maximal sub-caterpillar decomposition of T' is $\{C_1', C_2, \dots, C_k\}$ where $C_1' = C_1 \backslash \{1\}$. By the induction hypothesis, we know that

$$S(T') = |C_1'| \cdot \prod_{i=2}^{k} |C_i| = |C_1 - 1| \cdot \prod_{i=2}^{k} |C_i| = \prod_{i=1}^{k} |C_i| - \prod_{i=2}^{k} |C_i| = a.$$

Observe that $\mathcal{D}(T)$ contains subtrees

i) with both leaves 1 and 2;
ii) without leaf 1, with leaf 2;
iii) with leaf 1, without leaf 2;
iv) without leaves 1 and 2.

Take the TCSs, $R_1, \dots, R_a$, that reduce all trees in $\mathcal{D}(T')$. Append $(1, 2)$ as a prefix in order to obtain sequences $(1, 2)R_i$, for $i = 1, \dots, a$. Note that those sequences are TCSs for T that reduce all subtrees in i), ii) and iv). Thus, the only subtrees of T that still need to be reduced, are those in iii).

We claim that with $b = \prod_{i=2}^{k} |C_i|$ extra TCSs, $Q_1, \dots, Q_b$, we can reduce the remaining subtrees. These sequences can be decomposed in the parts where each maximal caterpillar is reduced. Let $C_1, \dots, C_p$ be the maximal caterpillars that are in the first branch, T_1, of T and let $C_{p+1}, \dots, C_k$ be the maximal caterpillars in the second branch, T_2, of T. From every maximal caterpillar, we need every leaf to remain at least once, because we need all possible cherries (from different maximal sub-caterpillars) to be reduced with some TCS.

Each of the sequences Q_i start with $\underline{S_1} \underline{S_{x_2}} \cdots \underline{S_{x_p}} (x_2, 1) \cdots (x_p, 1)$, where $\underline{S_{x_j}}$ is the x_j-fixed sequence which reduces maximal caterpillar C_j, by fixing on leaf x_j of C_j for $2 \leq j \leq p$. $\underline{S_1}$ reduces the first maximal caterpillar, by fixing on leaf 1. Note that for every caterpillar C_j, there are $|C_j|$ choices for choosing the leaf x_j on which to fix the sequence. Thus, with $\prod_{i=2}^{p} |C_i|$ different combinations, we can obtain all possible combinations of the x_j's in the first p parts of the sequences Q_i. After applying this part of the sequences Q_i to T, we obtain a tree T'', with $\{C_1'', C_{p+1}, \dots, C_k\}$ as maximal sub-caterpillar decomposition, where $C_1'' = \{1\} = T_1''$. By Conjecture 5.2, we know that

$$S(T'') = \max\left\{|C_1''| + \sum_{i=p+1}^{k} |C_i| - 1, |C_1''| \cdot \prod_{i=p+1}^{k} |C_i|\right\} = \max\left\{\sum_{i=p+1}^{k} |C_i|, \prod_{i=p+1}^{k} |C_i|\right\} = \prod_{i=p+1}^{k} |C_i|,$$

since $|C_i| \geq 2$ for all $i \in \{p + 1, \dots, k\}$.

Thus with $\prod_{i=p+1}^{k} |C_i|$ TCSs we can reduce all subtrees of T''. So for the second part of the sequences Q_i we have $\prod_{i=p+1}^{k} |C_i|$ different options. Together with the $\prod_{i=2}^{p} |C_i|$ options for the first part, we obtain $\prod_{i=2}^{p} |C_i| \cdot \prod_{i=p+1}^{k} |C_i| = \prod_{i=2}^{k} |C_i| = b$ different sequences $Q_1, \dots, Q_b$, which reduce all subtrees in iii).

Thus with the $a = \prod_{i=1}^{k} |C_i| - \prod_{i=2}^{k} |C_i|$ sequences $(1,2)R_i$ that reduce all subtrees in i), ii) and iv), and the $b = \prod_{i=2}^{k} |C_i|$ sequences Q_i that reduce all subtrees in iii), we have $a + b = \prod_{i=1}^{k} |C_i|$ TCSs that reduce all subtrees of T.

Assume that we are in case 2) or case 3a). Upon removing leaf 1 from $C_1 = \{1,2\}$, we obtain a tree T' on $n-1$ leaves with maximal sub-caterpillar decomposition $\{C_1', C_2, \dots, C_k\}$, where $C_1' = C_1 \backslash \{1\} = \{2\}$. By Conjecture 5.2 we know that

$$S(T') = \max\left\{|C_1'| + \sum_{i=2}^{k} |C_i| - 1, |C_1'| \cdot \prod_{i=2}^{k} |C_i|\right\} = \max\{2(k-1), 2^{k-1}\} = 2^{k-1},$$

since $|C_i| = 2$ for all $i \in \{2, \dots, k\}$. So there exist 2^{k-1} sequences, $S_1, \dots, S_{2^{k-1}}$, that reduce all subtrees of T'.

Note that $\mathcal{D}(T)$ has subtrees

i) with both leaves 1 and 2;

ii) without leaf 1, with leaf 2;

iii) with leaf 1, without leaf 2;

iv) without leaves 1 and 2.

One can verify that upon appending $(1,2)$ as a prefix to sequences S_i, we obtain 2^{k-1} TCSs for T, $(1,2)S_1, \dots, (1,2)S_{2^{k-1}}$ that reduce all subtrees in i), ii) and iv). We can obtain another 2^{k-1} TCSs for T by swapping leaves 1 and 2 in all these sequences, obtaining the TCSs $(2,1)\bar{S}_1, \dots, (2,1)\bar{S}_{2^{k-1}}$, where $\bar{S}_i$ denotes the sequence S_i with 1 and 2 swapped. Note that these TCSs reduce all subtrees in i), iii) and iv). Thus with $2 \cdot 2^{k-1} = 2^k = \prod_{i=1}^{k} |C_i|$ TCSs we can reduce all subtrees in $\mathcal{D}(T)$.

Assume we are in case 3b). Without loss of generality, we can assume that $|T_1| \geq |T_2| \geq 4$. Let $T_1 = \cup_{i=1}^{m}$ and $T_2 = \cup_{i=m+1}^{k}$, for some m with $m \geq k - m \geq 2$. We replace all leaves from T_1 by a single leaf l and call the obtained tree T'. By Conjecture 5.1 we know that $S(T') = \prod_{i=m+1}^{k} |C_i| = 2^{k-m}$. Thus there exist 2^{k-m} sequences, $S_1, \dots, S_{2^{k-m}}$, that reduce all subtrees of T'.

Let T'' be the subtree rooted at the $lca(T_1)$. By the induction hypothesis we have $S(T'') = \prod_{i=1}^{m} |C_i| = 2^m$. Thus there exist 2^m sequences, $R_1, \dots, R_{2^m}$, that reduce all subtrees of T''. Note that $2^m > 2m$, since we have $m \geq 2$. Thus we can adjust the sequences R_i such that every leaf in $T_1 = \{1, \dots, 2m\}$ appears at least once as the last coordinate of the last pair of such sequence $\bar{R}_i$.

We can combine the sequences S_i and $\bar{R}_i$ in order to reduce all subtrees of T. We create $2^m \cdot 2^{k-m} = 2^k$ sequences $\bar{R}_1 S_{1,1}, \dots, \bar{R}_{2^m} S_{1,2^m}, \dots, \bar{R}_1 S_{2^{k-m},1}, \dots, \bar{R}_{2^m} S_{2^{k-m},2^m}$, where $S_{i.j}$ is denotes the sequence S_i where leaf l is replaced by the leaf that is the last element of sequence $\bar{R}_j$. One can verify that these $2^k = \prod_{i=1}^{k} |C_i|$ sequences are TCSs for T that reduce all subtrees of T.

In all three cases we have shown that with $\prod_{i=1}^{k} |C_i|$ TCSs we can reduce all subtrees of T. This gives an upper bound on the minimal covering number $S(T) \leq \prod_{i=1}^{k} |C_i|$. Combined with the lower bound from Lemma 5.2, we have $S(T) = \prod_{i=1}^{k} |C_i|$.

Thus by induction, for all $n \geq 4$, a tree T on n leaves, with maximal sub-caterpillar decomposition $\{C_1, \dots, C_k\}$, where $|C_i| \geq 2$ for all i, has $S(T) = \prod_{i=1}^{k} |C_i|$ TCSs that reduce all its subtrees.

□

Bibliography

1. Huson, D. H., Rupp, R. & Scornavacca, C. in *Phylogenetic Networks: Concepts, Algorithms and Applications* 68–84 (Cambridge University Press, 2010).
2. Janssen, R. & Murakami, Y. On cherry-picking and network containment. *Theoretical Computer Science* (2021).
3. Erdős, P. L., Semple, C. & Steel, M. A class of phylogenetic networks reconstructable from ancestral profiles. *Mathematical Biosciences* **313,** 33–40 (2019).
4. van Iersel, L., Semple, C. & Steel, M. Locating a tree in a phylogenetic network. *Information Processing Letters* **110,** 1037–1043 (2010).
5. Sleator, D. D. & Tarjan, R. E. A Data Structure for Dynamic Trees. *Journal of Computer and System Sciences 26, 362-391* (1983).
6. Harel, D. & Tarjan, R. E. Fast Algorithms for finding Nearest Common Ancestors. *SIAM Journal on Computing 13(2)* (1984).
7. van Iersel, L., Janssen, R., Jones, M., Murakami, Y. & Zeh, N. A unifying characterization of tree-based networks and orchard networks using cherry covers. *Advances in Applied Mathematics* **129,** 102222. ISSN: 0196-8858 (2021).
8. Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. & Shmoys, D. B. in *Logistics of Production and Inventory* 445–522 (Elsevier, 1993).