Computing the Rooted Triplet Distance for Phylogenetic Trees and Caterpillars

L. van Eeuwijk

TU Delft

Computing the Rooted Triplet Distance for Phylogenetic Trees and Caterpillars

by

L. van Eeuwijk

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Thursday July 6, 2023 at 02:00 PM.

Student number:	5391342	
Project duration:	May 1, 2023 – July 6, 2023	
Thesis committee:	Y. Murakami,	TU Delft, supervisor
	Dr. B. J. Meulenbroek,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

In phylogenetics, it is important to figure out what method to obtain a phylogenetic tree, a tree showing how species evolve from one another, is more reliable. A phylogenetic tree is a mathematical tree, where every internal vertex has at least 2 children, and the leaves are labeled bijectively with a set X. One useful tool to help determine what method is more reliable is to find the rooted triplet distance between two trees. If the distance from a newly developed tree is far from what we know to be reliable trees, then this method is probably unreliable. For this distance, one should look at how many triplets the trees do not have in common. In this report, we will look at a specific algorithm to compute the rooted triplet distance between two special rooted phylogenetic trees, caterpillars, by checking how many triplets these two caterpillars have in common. In this report, we will map the leaves into $\mathbb{N}^3$ and $\mathbb{N}^d$, to determine how many triplets 3 or d trees have in common. Especially for $\mathbb{N}^d$ it becomes complicated to decide for what regions of $\mathbb{N}^d$ the leaves need to be counted, and for which ones not. After finding the closed-form notation for this, we will also discuss how many terms there are in the sum of this equation. Lastly, we will talk about the time complexity of this algorithm. The idea is that this algorithm will work faster than the naive approach, which runs in $O(n^3)$ time. Due to complications in the third and higher dimensions, this faster running time may or may not be accomplished, depending on whether another algorithm can be found to resolve these complications, which will not be done in this report. A closed-form notation can be found to express the number of triplets two caterpillars do not have in common, and the number of terms this equation has will also be discussed in Chapter 5.

Contents

1

Introduction

Phylogenetics is a study focusing on the evolutionary relationship among species. It helps us understand the evolutionary history by establishing connections between species. This allows us to classify and name species so that we can understand their similarities and differences. A very common way to display these connections is in the form of a tree, but how do we know what this tree should look like? There are many methods to determine what species should be connected to other species within the tree, and all of these methods will result in different trees. That begs the question; How valid are these methods? We can compare the looks of the trees and if one tree differs very much from most of the other trees one could conclude that the method that created this distant tree might not be that favorable.

But simply looking at the tree is not very exact, in 1975 A. J. Dobson published a paper [2] in which he came up with a method to determine a numerical distance between two phylogenetic trees, which is a useful tool to determine how closely related two trees are. The way we compute this distance is by looking at every combination of three leaves in a tree and whether these triplets have the same form in these two trees or not. We count how often they are not the same, and that is defined as the distance. However, computing this distance naively between two rather large trees can cost a lot of time, which is why many researchers have tried to find algorithms to compute this distance faster. The latest improvement in theoretical time complexity was found in 2013 by Brodal et al [4]. They managed to compute the rooted triplet distance in $O(nlog(n))$ time. Later in 2017 Jansson and Rajaby [5] found an algorithm that in practice ran faster, but theoretically was slower, namely $O(nlog^3(n))$.

Another way to find faster algorithms would be to restrict what the tree can look like. For caterpillars, trees where all vertices have a neighbor on a central path, an algorithm was found by J. Jansson and W. L. Lee [1], which is the algorithm that will be examined and expanded in this paper. While Jansson and Lee used the algorithm to compute the number of triplets two caterpillars have in common, this paper will focus more on computing the number of triplets more than two caterpillars have in common, by expanding the 2-dimensional algorithm from Jansson and Lee to 3- and eventually d-dimensional. The 2-dimensional algorithm maps the leaf of a caterpillar into $\mathbb{N}^2$, after which the number of leaves in specific regions of $\mathbb{N}^2$ get counted and summed up, which gives the rooted triplet distance. To the knowledge of the author, no research has been done on computing the amount of triplets d caterpillars have in common.

After giving some definitions in Chapter 2, Chapter 3 will give a fast algorithm to compute this distance between two caterpillars, which was found in [1]. This algorithm works by mapping all leaves of the caterpillars to $\mathbb{N}^2$, then all the triplets that these two caterpillars have in common will be counted using this mapping into $\mathbb{N}^2$, which gives us the rooted triplet distance by subtracting this number from the total amount of triplets. Chapter 4 will expand this algorithm to the third dimension, where we simply look for the number of triplets 3 caterpillars do not have in common, and Chapter 5 will expand this to the dth dimension.

2

Preliminaries

2.1. Definitions

To further understand the problem, some terms need to be defined;

A ***graph*** is a pair (V, E), where V is a set of vertices and E is a set of edges. A ***tree*** is a connected acyclic graph. A ***leaf*** is a vertex that has only one edge connected to it. An ***internal vertex*** is a vertex that has two or more edges connected to it. A ***rooted tree*** is a tree in which one vertex has been designated the root, and all vertices are directed away from this root. A ***rooted phylogenetic tree*** is a rooted tree, where every internal vertex has at least 2 children, and the leaves are labeled bijectively with a set X. A ***caterpillar*** is a tree where all vertices are on a central path or have a neighbor on the central path. In this paper, only rooted phylogenetic caterpillars will be looked at. A ***rooted triplet*** or ***triplet*** is a rooted phylogenetic tree with precisely three leaves, which has one of the three ***forms*** in Figure 2.1. When talking about a triplet in a tree with more than 3 leaves, the triplet is the tree obtained as a result of removing all other leaves, as well as removing all internal nodes that have no leaf as a neighbor, such that only 3 leaves and at most 2 internal vertices are left. A vertex u is a ***parent*** of another vertex v if a directed edge goes from u to the ***child*** v. An internal vertex x is ***lower*** than an internal vertex y if there is a path from y to x in the caterpillar, here y is ***higher*** than x. A leaf x is lower than a leaf y if there is a path from the parent of y to the parent of x in the caterpillar, here y is higher than x. A ***resolved triplet*** is a triplet where two leaves share a parent, which is not a parent of the third leaf.

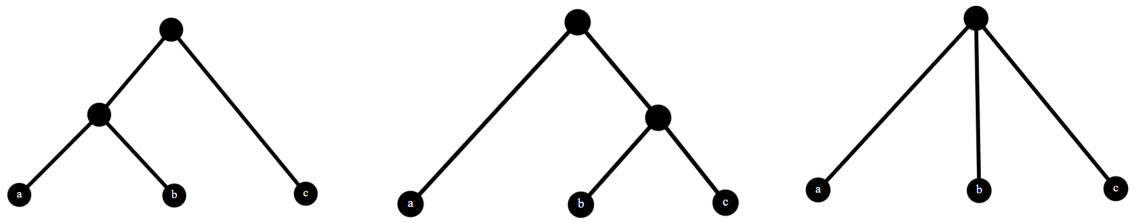


Figure 2.1: All possible triplets on a leaf-set $\{a, b, c\}$, the first two are resolved, the third is unresolved

An ***unresolved triplet*** is a triplet where all three leaves have the same parent. A ***good triplet*** is a triplet that has the same form in two trees. A ***bad triplet*** is a triplet that has a different form in two trees. The ***rooted triplet distance*** is the number of all bad triplets. There are $\binom{leaves}{3}$ triplets and all have to be checked to see if they are of the same form in the two trees, or of a different form.

2.2. Dividing $\mathbb{N}^d$

In chapter 3 of this report, an algorithm gets used, that splits $\mathbb{N}^d$ into sections. Let us first define this division for $\mathbb{N}^2$;

Take a point (p_1, p_2) in $\mathbb{N}^2$, we call this region $A_{0,0}$. Now we define the region $[1, p_1 - 1] \times \{p_2\}$ to be region $A_{1,0}$ and define the region $\{p_1\} \times [1, p_2 - 1]$ to be region $A_{0,1}$. Finally, we call the region $[1, p_1 - 1] \times [1, p_2 - 1]$ $A_{1,1}$. So for every point in $\mathbb{N}^2$, We create four regions, which the algorithm can then use for certain points.

We can expand this definition of the regions to $\mathbb{N}^d$. Let $(p_1,...,p_d)$ be a point in $\mathbb{N}^d$. If the ith coordinate of a point is in $[1,p_i-1]$, it will be in a region where the ith subscript of A is 1, whereas if the ith coordinate is p_i, it will be in a region where the ith subscript of A is equal to 0. To properly define the regions A, we first need to define a function f that maps leaves into $\mathbb{N}^d$ and we also define a function f' that maps triplets into $\mathbb{N}^d$, which will get used later for better efficiency.

Definition 2.2.1. *Let $T_1,...,T_d$ be d caterpillars with n leaves from a set X having $\binom{n}{3}$ triplets. Label all internal vertices of all caterpillars, such that the lowest internal vertex has label 1 and the vertex above it has label 2, etc. We define a leaf map f that maps leaves into $\mathbb{N}^d$.*

$$f: X \to \mathbb{N}^d; x \to f(x)$$
$$f(x) = (\textit{Label of parent of } x \textit{ in caterpillar } T_1,..., \textit{Label of parent of } x \textit{ in caterpillar } T_d)$$

We also define a triplet map f' that maps triplets into $\mathbb{N}^d$. This function will be used by the algorithm.

$$f': X \times X \times X \to \mathbb{N}^d; \tau \to f'(\tau)$$
$$f'(\tau) = (\textit{Label of root of } \tau \textit{ in } T_1,..., \textit{Label of root of } \tau \textit{ in } T_d)$$

Example 2.2.2. *Let T_1 and T_2 be as in Figure 2.2. Here we see that $f(a)=(3,1)$, $f(b)=(3,1)$, $f(c)=(2,1)$, $f(d)=(3,2)$, $f(e)=(1,3)$ and $f(f)=(3,3)$.*

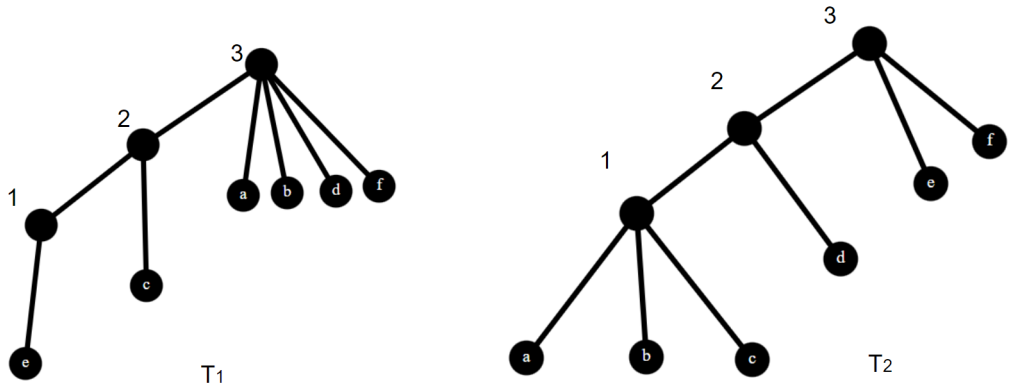


Figure 2.2: An example of how the map f works

Definition 2.2.3. *Let $(p_1,...,p_d)$ be a point in $\mathbb{N}^d$. For this point we define 2^d regions:*

$$A_{b_1,b_2,...,b_d} = \prod_{i=1}^{d} C_i, \textit{ where } C_i = \begin{cases} \{p_i\}, & \textit{if } b_i = 0 \\ [1, p_i - 1], & \textit{if } b_i \neq 0 \end{cases}$$

Definition 2.2.4. *Let $(p_1,...,p_d)$ be a point in $\mathbb{N}^d$. For this point we define the following numbers, which tell us how many leaves are in each region, which the algorithm can use:*

$$a_{b_1,b_2,...,b_d} = |\{x \in X : f(x) \in A_{b_1,b_2,...,b_d}\}|$$

3

The Algorithm

3.1. How Does the Algorithm Work?

The goal is to find the rooted triplet distance between two caterpillars. There are many ways to calculate the distance between caterpillars, one way is to look at all triplets in both caterpillars and count how many bad triplets there are. The time complexity for this naively is $O(n^3)$, as we consider every leaf 3 times, which is rather slow. For this report, the second algorithm from the paper Fast Algorithms for the Rooted Triplet Distance Between Caterpillars by Jesper Jansson and Wing Lik Lee [1] will be looked further into, which can compute the rooted triplet distance faster.

For a proper proof of the algorithm, one should consult the paper by Jansson and Lik Lee. In this section, we discuss how the algorithm works.

The idea is that we count the number of good triplets, and then we calculate the number of bad triplets, the rooted triplet distance, by subtracting the number of good triplets from the total amount of triplets. We count the number of good triplets by iterating over every point in the image of $f'(x)$. First divide $\mathbb{N}^2$ into four regions, as done in Section 2.2, which are different for each point in the image of $f'(x)$; $a_{0,0}$ is the number of leaves in region $\{p_1\} \times \{p_2\}$, $a_{0,1}$ is the number of leaves in region $\{p_1\} \times [1, p_2 - 1]$, $a_{1,0}$ is the number of leaves in region $[1, p_1 - 1] \times \{p_2\}$ and $a_{1,1}$ is the number of leaves in region $[1, p_1 - 1] \times [1, p_2 - 1]$. This is also illustrated in Figure 3.1.

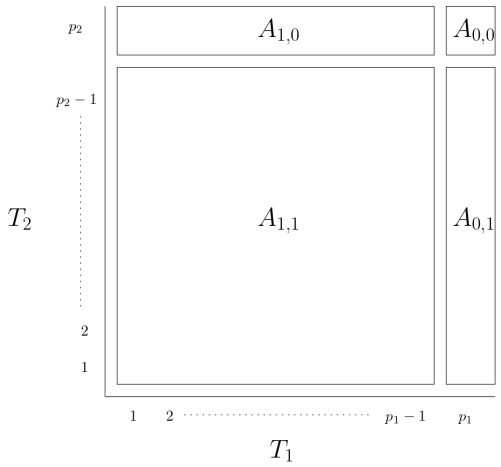


Figure 3.1: Division of $[1, p_1] \times [1, p_2]$ into four regions.

3.1.1. Resolved Triplets

For resolved triplets, one can observe that the triplet must look the same in both caterpillars, as in Figure 3.2. We must have one leaf with a higher vertex as parent in both caterpillars, and the other two leaves must have a lower vertex as parent, thus the higher leaf will be in region $A_{0,0}$, and the lower two leaves will be in region $A_{1,1}$. Therefore we have $a_{0,0} \cdot \binom{a_{1,1}}{2}$ good triplets of this form.

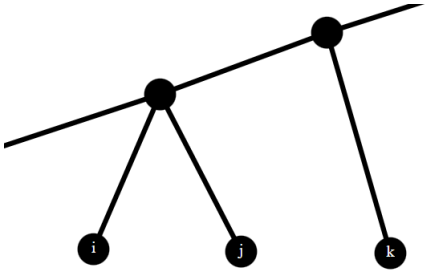


Figure 3.2: A good unresolved triplet $\{i, j, k\}$

3.1.2. Unresolved Triplets

One can count the number of unresolved triplets similarly, but it would have to be split up into 4 cases, as was done by Janssen and Lik Lee, which is also visualized in Figure 3.3.

Define a ***short*** unresolved triplet as a triplet where all three leaves have the same parent and define a ***long*** unresolved triplet as a triplet where two leaves share a parent, and the other leaf has a lower vertex as a parent. An unresolved triplet is a good triplet if it is unresolved in both caterpillars, which can happen in all combinations of long and short triplets. However in the different combinations of long and short, the leaves will be in other regions of $\mathbb{N}^2$, thus we need to count all combinations separately;
Case 1 (short, short): all leaves have the same parent in both caterpillars, so all leaves are in region $A_{0,0}$.
count: $\binom{a_{0,0}}{3}$
Case 2 (short, long): all leaves have the same parent the first caterpillar, but one has a different parent in the second caterpillar, so two leaves are in region $A_{0,0}$, and the other is in region $A_{0,1}$.
count: $\binom{a_{0,0}}{2} \cdot a_{0,1}$
Case 3 (long, short): all leaves have the same parent in the second caterpillar, but one has a different parent in the first caterpillar, so two leaves are in region $A_{0,0}$, and the other is in region $A_{1,0}$.
count: $\binom{a_{0,0}}{2} \cdot a_{1,0}$
Case 4 (long, long): one leaf has a different parent from the other two in both caterpillars, but this can happen in two ways;
Case 4a: the same leaf is separate in both caterpillars, thus this leaf is in $A_{1,1}$, while the other two are in $A_{0,0}$.
count: $\binom{a_{0,0}}{2} \cdot a_{1,1}$
Case 4b: in the two caterpillars a different leaf is separated, thus one leaf is in $A_{0,0}$, one leaf is in $A_{0,1}$ and one leaf is in $A_{1,0}$.
count: $a_{0,0} \cdot a_{0,1} \cdot a_{1,0}$
Now all good unresolved and resolved triplets have been counted, thus the rooted triplet distance can be computed:

$$d_{rt}(T_1, T_2) = \binom{n}{3} - \sum_{p \in Im(f')} (a_{0,0} \cdot \binom{a_{1,1}}{2} + \binom{a_{0,0}}{3} + \binom{a_{0,0}}{2}(a_{0,1} + a_{1,0} + a_{1,1}) + a_{0,0} \cdot a_{0,1} \cdot a_{1,0}).$$

3.2. Time Complexity

In the paper it was proven that this algorithm has a time complexity of $O(n\sqrt{\log n})$, which is theoretically faster than was previously found, and also definitely faster than $O(n^3)$. In the rest of the report, this algorithm will be expanded to the third and dth dimension, which will allow us to compute of the number of bad triplets in three or d caterpillars.

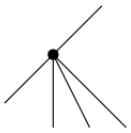
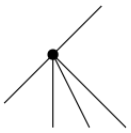
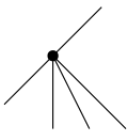
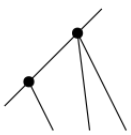
	T_1	T_2	position	count
Case 1 (short, short)			$i : A_{0,0}$ $j : A_{0,0}$ $k : A_{0,0}$	$\binom{a_{0,0}}{3}$
Case 2 (short, long)			$i : A_{0,1}$ $j : A_{0,0}$ $k : A_{0,0}$	$\binom{a_{0,0}}{2} \cdot a_{0,1}$
Case 3 (long, short)			$i : A_{1,0}$ $j : A_{0,0}$ $k : A_{0,0}$	$\binom{a_{0,0}}{2} \cdot a_{1,0}$
Case 4a (long, long)			$i : A_{1,1}$ $j : A_{0,0}$ $k : A_{0,0}$	$\binom{a_{0,0}}{2} \cdot a_{1,1}$
Case 4b (long, long)			$i : A_{1,0}$ $j : A_{0,1}$ $k : A_{0,0}$	$a_{0,0} \cdot a_{0,1} \cdot a_{1,0}$

Figure 3.3: Counting unresolved good triplets. The first two columns indicate what triplets look like in caterpillar 1 and caterpillar 2, the third column indicates what region the leaves are in, and the fourth column indicates how many good triplets there are for this case.

4

The Third Dimension

Now that the distance between two caterpillars has been discussed, let us look at how many bad triplets there are in T_1, T_2 and T_3. Define a bad triplet as a triplet that is not consistent in all 3 caterpillars, meaning that the triplet must have a different form in at least one of the three caterpillars. For this chapter we will expand the idea of Chapter 3 to have more inputs, and thus a higher dimension. Similarly to Chapter 3, we will first count the good triplets and then subtract that number from all possible triplets.

4.1. Dividing $\mathbb{N}^3$

To reiterate we divide $\mathbb{N}^3$ in 8 parts per point in the image of $f'(x)$, just like in section 2.2:

- $a_{0,0,0}$ is the number of leaves in region $\{p_1\} \times \{p_2\} \times \{p_3\}$.
- $a_{0,0,1}$ is the number of leaves in region $\{p_1\} \times \{p_2\} \times [1, p_3 - 1]$.
- $a_{0,1,0}$ is the number of leaves in region $\{p_1\} \times [1, p_2 - 1] \times \{p_3\}$.
- $a_{1,0,0}$ is the number of leaves in region $[1, p_1 - 1] \times \{p_2\} \times \{p_3\}$.
- $a_{0,1,1}$ is the number of leaves in region $\{p_1\} \times [1, p_2 - 1] \times [1, p_3 - 1]$.
- $a_{1,0,1}$ is the number of leaves in region $[1, p_1 - 1] \times \{p_2\} \times [1, p_3 - 1]$.
- $a_{1,1,0}$ is the number of leaves in region $[1, p_1 - 1] \times [1, p_2 - 1] \times \{p_3\}$.
- $a_{1,1,1}$ is the number of leaves in region $[1, p_1 - 1] \times [1, p_2 - 1] \times [1, p_3 - 1]$.

This is visualized in Figure 4.1, where $A_{1,1,1}$ is the largest region, behind the surfaces.

4.2. Counting Good Triplets

4.2.1. Resolved Triplets

In Chapter 3 we saw that resolved triplets only occurred when one leaf was in $A_{0,0}$ and the other two were in $A_{1,1}$. In the third dimension we can observe that something similar happens, namely one leaf must be in $A_{0,0,0}$, while the other two are in $A_{1,1,1}$. We have that these triplets must look like Figure 3.2 in all three caterpillars, thus one leaf is on the higher node three times, and thus in region $A_{0,0,0}$, while the other two leaves are on lower nodes in all three caterpillars, thus these are in region $A_{1,1,1}$. This makes the count for resolved triplets $a_{0,0,0} \cdot \binom{a_{1,1,1}}{2}$. One might think that one leaf in $A_{0,0,0}$ and the other two in for example $A_{0,1,1}$ would also lead to a good resolved triplet, but that is not the case as that would mean the triplet is unresolved in T_1.

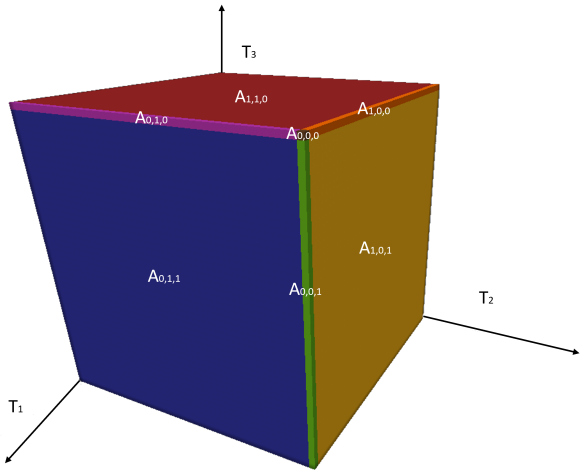


Figure 4.1: Division of $[1, p_1] \times [1, p_2] \times [1, p_3]$ into eight regions.

4.2.2. Unresolved Triplets

Just like in Fast Algorithms for the Rooted Triplet Distance Between Caterpillars [1], we will have to split up the unresolved section into cases, all combinations of long and short (each triplet can be short or long, as in Chapter 3), thus we have 8 cases, and cases with at least two longs will have subcases.
Case 1 (short, short, short): all leaves have the same parent in all caterpillars, so all leaves are in region $A_{0,0,0}$.
count: $\binom{a_{0,0,0}}{3}$
Case 2 (short, short, long): all leaves have the same parent in the first and second caterpillar, but one has a different parent in the third caterpillar, so two leaves are in region $A_{0,0,0}$, and the other is in region $A_{0,0,1}$.
count: $\binom{a_{0,0,0}}{2} \cdot a_{0,0,1}$
Case 3 (short, long, short): all leaves have the same parent in the first and third caterpillar, but one has a different parent in the second caterpillar, so two leaves are in region $A_{0,0,0}$, and the other is in region $A_{0,1,0}$.
count: $\binom{a_{0,0,0}}{2} \cdot a_{0,1,0}$
Case 4 (long, short, short): all leaves have the same parent in the second and third caterpillar, but one has a different parent in the first caterpillar, so two leaves are in region $A_{0,0,0}$, and the other is in region $A_{1,0,0}$.
count: $\binom{a_{0,0}}{2} \cdot a_{1,0,0}$
Case 5 (short, long, long): one leaf has a different parent from the other two in the second and third caterpillars, but this can happen in two ways;
Case 5a: the same leaf is separate in both caterpillars, thus this leaf is in $A_{0,1,1}$, while the other two are in $A_{0,0,0}$.
count: $\binom{a_{0,0,0}}{2} \cdot a_{0,1,1}$
Case 5b: in the two caterpillars a different leaf is separated, thus one leaf is in $A_{0,0,0}$, one leaf is in $A_{0,0,1}$ and one leaf is in $A_{0,1,0}$.
count: $a_{0,0,0} \cdot a_{0,0,1} \cdot a_{0,1,0}$
Case 6 (long, short, long): one leaf has a different parent from the other two in the first and third caterpillars, but this can happen in two ways;
Case 6a: the same leaf is separate in both caterpillars, thus this leaf is in $A_{1,0,1}$, while the other two are in $A_{0,0,0}$.
count: $\binom{a_{0,0,0}}{2} \cdot a_{1,0,1}$
Case 6b: in the two caterpillars a different leaf is separated, thus one leaf is in $A_{0,0,0}$, one leaf is in $A_{0,0,1}$ and one leaf is in $A_{1,0,0}$.
count: $a_{0,0,0} \cdot a_{0,0,1} \cdot a_{1,0,0}$
Case 7 (long, long, short): one leaf has a different parent from the other two in the first and second caterpillars, but this can happen in two ways;
Case 7a: the same leaf is separate in both caterpillars, thus this leaf is in $A_{1,1,0}$, while the other two are in $A_{0,0,0}$.
count: $\binom{a_{0,0,0}}{2} \cdot a_{1,1,0}$
Case 7b: in the two caterpillars a different leaf is separated, thus one leaf is in $A_{0,0,0}$, one leaf is in $A_{1,0,0}$ and

one leaf is in $A_{0,1,0}$.
count: $a_{0,0,0} \cdot a_{1,0,0} \cdot a_{0,1,0}$
Case 8 (long, long, long): one leaf has a different parent from the other two in all caterpillars, but this can happen in five ways;
Case 8a: the same leaf is separate in all caterpillars, thus this leaf is in $A_{1,1,1}$, while the other two are in $A_{0,0,0}$.
count: $\binom{a_{0,0,0}}{2} \cdot a_{1,1,1}$
Case 8b: in the first and second caterpillars the same leaf is separated, but in the third another one is, thus one leaf is in $A_{0,0,0}$, one leaf is in $A_{1,1,0}$ and one leaf is in $A_{0,0,1}$
count: $a_{0,0,0} \cdot a_{1,1,0} \cdot a_{0,0,1}$.
Case 8c: in the first and third caterpillars the same leaf is separated, but in the second another one is, thus one leaf is in $A_{0,0,0}$, one leaf is in $A_{1,0,1}$ and one leaf is in $A_{0,1,0}$
count: $a_{0,0,0} \cdot a_{1,0,1} \cdot a_{0,1,0}$.
Case 8d: in the second and third caterpillars the same leaf is separated, but in the first another one is, thus one leaf is in $A_{0,0,0}$, one leaf is in $A_{0,1,1}$ and one leaf is in $A_{1,0,0}$
count: $a_{0,0,0} \cdot a_{0,1,1} \cdot a_{1,0,0}$.
Case 8e: in all caterpillars a different leaf is separated, thus one leaf is in $A_{1,0,0}$, one leaf is in $A_{0,1,0}$ and one leaf is in $A_{0,0,1}$.
count: $a_{1,0,0} \cdot a_{0,1,0} \cdot a_{0,0,1}$
All of these cases are also represented in Figure 4.2 and Figure 4.3.
Now that all good triplets have been found, the number of bad triplets between the three caterpillars can be computed:

$$d_{rt}(T_1,T_2,T_3) = \binom{n}{3} - \sum_{p \in Im(f')} (a_{0,0,0} \cdot \binom{a_{1,1,1}}{2} + \binom{a_{0,0,0}}{3} + \binom{a_{0,0,0}}{2}(a_{0,0,1}+a_{0,1,0}+a_{1,0,0}+a_{0,1,1}+a_{1,0,1}+a_{1,1,0}+a_{1,1,1}) +$$
$$a_{0,0,0} \cdot (a_{0,0,1}a_{0,1,0} + a_{0,0,1}a_{1,0,0} + a_{0,1,0}a_{1,0,0} + a_{0,0,1}a_{1,1,0} + a_{0,1,0}a_{1,0,1} + a_{1,0,0}a_{0,1,1}) + a_{0,0,1}a_{0,1,0}a_{1,0,0}).$$

	T_1	T_2	T_3	position	count
Case 1 (short, short, short)				$i : A_{0,0,0}$ $j : A_{0,0,0}$ $k : A_{0,0,0}$	$\binom{a_{0,0,0}}{3}$
Case 2 (short, short, long)				$i : A_{0,0,1}$ $j : A_{0,0,0}$ $k : A_{0,0,0}$	$\binom{a_{0,0,0}}{2} \cdot a_{0,0,1}$
Case 3 (short, long, short)				$i : A_{0,1,0}$ $j : A_{0,0,0}$ $k : A_{0,0,0}$	$\binom{a_{0,0,0}}{2} \cdot a_{0,1,0}$
Case 4 (long, short, short)				$i : A_{1,0,0}$ $j : A_{0,0,0}$ $k : A_{0,0,0}$	$\binom{a_{0,0,0}}{2} \cdot a_{1,0,0}$
Case 5a (short, long, long)				$i : A_{0,1,1}$ $j : A_{0,0,0}$ $k : A_{0,0,0}$	$\binom{a_{0,0,0}}{2} \cdot a_{0,1,1}$
Case 5b (short, long, long)				$i : A_{0,1,0}$ $j : A_{0,0.1}$ $k : A_{0,0,0}$	$a_{0,0,0} \cdot a_{0,0,1}$ $\cdot a_{0,1,0}$
Case 6a (long, short, long)				$i : A_{1,0,1}$ $j : A_{0,0,0}$ $k : A_{0,0,0}$	$\binom{a_{0,0,0}}{2} \cdot a_{1,0,1}$
Case 6b (long, short, long)				$i : A_{1,0,0}$ $j : A_{0,0,1}$ $k : A_{0,0,0}$	$a_{0,0,0} \cdot a_{0,0,1}$ $\cdot a_{1,0,0}$

Figure 4.2: Counting unresolved good triplets. The first three columns indicate what triplets look like in caterpillar 1 and caterpillar 2 and caterpillar 3, the fourth column indicates what region the leaves are in, and the fifth column indicates how many good triplets there are for this case.

Case 7a (long, long, short)				$i : A_{1,1,0}$ $j : A_{0,0,0}$ $k : A_{0,0,0}$	$\binom{a_{0,0,0}}{2} \cdot a_{1,1,0}$
Case 7b (long, long, short)				$i : A_{1,0,0}$ $j : A_{0,1,0}$ $k : A_{0,0,0}$	$a_{0,0,0} \cdot a_{0,1,0}$ $\cdot a_{1,0,0}$
Case 8a (long, long, long)				$i : A_{1,1,1}$ $j : A_{0,0,0}$ $k : A_{0,0,0}$	$\binom{a_{0,0,0}}{2} \cdot a_{1,1,1}$
Case 8b (long, long, long)				$i : A_{1,1,0}$ $j : A_{0,0,1}$ $k : A_{0,0,0}$	$a_{0,0,0} \cdot a_{1,1,0}$ $\cdot a_{0,0,1}$
Case 8c (long, long, long)				$i : A_{1,0,1}$ $j : A_{0,1,0}$ $k : A_{0,0,0}$	$a_{0,0,0} \cdot a_{1,0,1}$ $\cdot a_{0,1,0}$
Case 8d (long, long, long)				$i : A_{0,1,1}$ $j : A_{1,0,0}$ $k : A_{0,0,0}$	$a_{0,0,0} \cdot a_{0,1,1}$ $\cdot a_{1,0,0}$
Case 8e (long, long, long)				$i : A_{1,0,0}$ $j : A_{0,1,0}$ $k : A_{0,0,1}$	$a_{1,0,0} \cdot a_{0,1,0}$ $\cdot a_{0,0,1}$

Figure 4.3: Counting unresolved good triplets. The first three columns indicate what triplets look like in caterpillar 1 and caterpillar 2 and caterpillar 3, the fourth column indicates what region the leaves are in, and the fifth column indicates how many good triplets there are for this case.

5

The dth Dimension

Now that the second and third dimensions have been covered, we want to know how many bad triplets there are in d caterpillars. As with the third dimension, define a bad triplet as a triplet that is not the same in all d caterpillars.

5.1. d-Dimensional Sum

When one looks at the terms of the sum for the third dimension one can see that every term is the product of the cardinality of three regions, and in particular the indices of these regions follow a certain rule; every combination of three regions $a_{b_1,b_2,b_3}, a_{c_1,c_2,c_3}, a_{d_1,d_2,d_3}$ appears, such that $b_i + c_i + d_i$ is either equal to 1 or equal to 0 and anagrams get avoided (if $a_{0,0,1}a_{0,1,0}a_{1,0,0}$ is in the sum, $a_{0,1,0}a_{0,0,1}a_{1,0,0}$ will not be in the sum, as it would count those triplets twice). The claim is that this also works for d dimensions, now it just needs a proof. Recall that f' refers to the triplet mapping. Let (X,Y,Z) be an arbitrary triplet, with X,Y and Z in regions $A_{x_1,x_2,...,x_d}, A_{y_1,y_2,...,y_d}$ and $A_{z_1,z_2,...,z_d}$ respectively for the point $f'((X,Y,Z))$.

Lemma 5.1.1. *Let $T_1,...,T_d$ be caterpillars on the same set of taxa. Let X, Y and Z be leaves in the caterpillars. A triplet (X,Y,Z) is a good unresolved triplet if $x_i + y_i + z_i = 0$ or $x_i + y_i + z_i = 1, \forall i = 1,2,...,d$.*

Proof. For unresolved triplets, we need that the triplet is unresolved in every caterpillar, for which the order does not matter. All we need is that this triplet is unresolved in every caterpillar, so we look at x_i, y_i and z_i d times.
If $x_i + y_i + z_i = 0$ all three leaves are on the same internal vertex in caterpillar T_i, thus are unresolved in caterpillar T_i.
If $x_i + y_i + z_i = 1$ then two leaves are on the same internal vertex in caterpillar T_i, while one leaf is lower in the caterpillar, which still makes this triplet unresolved in caterpillar T_i. This means they are unresolved in every caterpillar, thus they are a good unresolved triplet □

Lemma 5.1.2. *Let $T_1,...,T_d$ be caterpillars on the same set of taxa. If a triplet $\tau = (X,Y,Z)$ is good and unresolved, then there exists exactly one point $p \in Im(f')$ such that at that point, $x_i + y_i + z_i = 0$ or $x_i + y_i + z_i = 1$ for all caterpillars T_i. This point is $f'(\tau)$.*

Proof. Let us look at the point $p = f'(\tau)$. By definition of f' we have for this triplet that $x_i + y_i + z_i = 0$, $x_i + y_i + z_i = 1$ or $x_i + y_i + z_i = 2$. If $x_i + y_i + z_i = 3$, it implies that none of the three leaves are mapped to the ith coordinate of $f'(\tau)$, which is a contradiction. If $x_i + y_i + z_i = 2$ for caterpillar T_i, then in this particular caterpillar we have that one leaf is on a higher internal vertex than the other two, meaning the triplet is resolved in this caterpillar, thus the triplet cannot be a good unresolved triplet. If $x_i + y_i + z_i = 0$ or $x_i + y_i + z_i = 1$ for all i we have that this is a good unresolved triplet by Lemma 5.1.1, thus for point $f'(\tau)$: $x_i + y_i + z_i = 0$ or $x_i + y_i + z_i = 1$.

If we look at another point $q \in Im(f')$ such that X,Y and Z are in some region $A_{b_1,...,b_d}$, it implies that at least one coordinate of q is greater than it was for p and none are less than they were for p. If q had a coordinate that was less than that coordinate in p X,Y and/or Z would not be in any region $A_{b_1,...,b_d}$, and so $x_i + y_i + z_i$ is not defined here. Assume that the jth coordinate of q got increased, then in caterpillar j we will have that the label of the parents of X,Y and Z is now lower than the jth coordinate of q, and therefore

$x_j + y_j + z_j = 3$. This means that there cannot be more than one point $p \in Im(f')$ such that at that point, $x_i + y_i + z_i = 0$ or $x_i + y_i + z_i = 1$ for all caterpillars T_i. □

Lemma 5.1.3. *Let $T_1, ..., T_d$ be caterpillars on the same set of taxa. Let X, Y and Z be leaves in the caterpillars. A triplet (X, Y, Z) is a good resolved triplet in T_i if $x_i + y_i + z_i = 2$, for all $i = 1, 2, ..., d$ and $x_i = x_j, y_i = y_j, z_i = z_j$, for all $i = 1, 2, ..., d$, for all $j = 1, 2, ..., d$*

Proof. We have that $x_i + y_i + z_i = 2$, for all $i = 1, 2, ..., d$ and $x_i = x_j, y_i = y_j, z_i = z_j$, for all $i = 1, 2, ..., d$, for all $j = 1, 2, ..., d$, which means there is one leaf, let's assume leaf Z without loss of generality, that is at the root of the triplet in every caterpillar, while leaves X and Y are lower than Z in all caterpillars. Because of this, the triplet is resolved in every caterpillar, with Z as the higher leaf, making this a good resolved triplet. □

Lemma 5.1.4. *Let $T_1, ..., T_d$ be caterpillars on the same set of taxa. If a triplet $\tau = (X, Y, Z)$ is good and resolved, then there exists exactly one point $p \in Im(f')$ such that at that point, $x_i + y_i + z_i = 2$ for all caterpillars T_i. This point is $f'(\tau)$.*

Proof. As the triplet is good and resolved, we have that one leaf, let's assume leaf Z without loss of generality, which is always a child of the root of the triplet, while X and Y are lower in the caterpillar. If we look at the regions for $f'(\tau) = f(Z)$, we see that $Z \in A_{0,...,0}$ and $X, Y \in A_{1,...,1}$. Therefore $x_i = y_i = 1$, for all $i = 1, 2, ..., d$ and $z_i = 0$, for all $i = 1, 2, ..., d$, making $x_i + y_i + z_i = 2$ for all caterpillars T_i at $f'(\tau)$

If we look at another point $q \in Im(f')$ such that X, Y and Z are in some region $A_{b_1,...,b_d}$, it implies that at least one coordinate of q is greater than it was for p and none are less than they were for p. If q had a coordinate that was less than that coordinate in p Z would not be in any region $A_{b_1,...,b_d}$, and so z_i is not defined here. Assume that the jth coordinate of q got increased, then in caterpillar j we will have that the label of the parents of Z is now lower than the jth coordinate of q, and therefore $z_j = 1 \Rightarrow x_i + y_i + z_i = 3$. This means that there cannot be more than one point $p \in Im(f')$ such that at that point, $x_i + y_i + z_i = 2$ for all caterpillars T_i. □

Theorem 5.1.5. *Let $T_1, ..., T_d$ be d caterpillars with n leaves from a set X. The number of bad triplets $d_{rt}(T_1, T_2, ..., T_d)$ can be computed in the following way:*

$$d_{rt}(T_1, T_2, ..., T_d) = \binom{n}{3} - \sum_{p \in Im(f')} (a_{0,0,...,0} \cdot \binom{a_{1,1,...,1}}{2} + \binom{a_{0,0,...,0}}{3} + \binom{a_{0,0,...,0}}{2} \cdot \sum_{\beta \in B} \beta + \binom{a_{0,0,...,0}}{1} \cdot \frac{1}{2} \sum_{\gamma \in C} \gamma + \frac{1}{6} \sum_{\delta \in D} \delta), \tag{5.1}$$

where

- $B = \{a_{b_1,b_2,...,b_d} | \sum_{i=1}^{d} b_i \geq 1\}$,
- $C = \{a_{b_1,b_2,...,b_d} \cdot a_{c_1,c_2,...,c_d} | \sum_{i=1}^{d} b_i \geq 1, \sum_{i=1}^{d} c_i \geq 1 \text{ and } \forall i = 1, 2, ..., d, b_i + c_i \leq 1\}$,
- $D = \{a_{b_1,b_2,...,b_d} \cdot a_{c_1,c_2,...,c_d} \cdot a_{d_1,d_2,...,d_d} | \sum_{i=1}^{d} b_i \geq 1, \sum_{i=1}^{d} c_i \geq 1, \sum_{i=1}^{d} d_i \geq 1 \text{ and } \forall i = 1, 2, ..., d, b_i + c_i + d_i \leq 1\}$.

Proof. This follows closely with how we have determined the distance for the $d = 2$ and $d = 3$ cases. We count all good triplets, then the number of bad triplets is the number of triplets minus the number of good triplets. Good triplets get counted per point p in $Im(f')$. To count the good triplets per point p, we first divide them into resolved and unresolved triplets.

Resolved triplets have been covered in Lemma 5.1.3 and Lemma 5.1.4; by Lemma 5.1.3 a triplet (X, Y, Z) is a good resolved triplet if $x_i + y_i + z_i = 2$ and $x_i = x_j, y_i = y_j, z_i = z_j$ for all i, j. And by Lemma 5.1.4 every good resolved triplet (X, Y, Z) will have exactly one point in $Im(f')$, $f'((X, Y, Z))$ such that $x_i + y_i + z_i = 2$, thus this triplet gets counted exactly once. We also see from Lemma 5.1.4 that, without loss of generality, $Z \in A_{0,...,0}$ and $X, Y \in A_{1,...,1}$ when looking at the point $f'((X, Y, Z))$, thus the number of good resolved triplets for $f'((X, Y, Z))$ is $a_{0,0,...,0} \cdot \binom{a_{1,1,...,1}}{2}$.

Unresolved triplets have been covered in Lemma 5.1.1 and Lemma 5.1.2; by Lemma 5.1.1 a triplet (X, Y, Z) is a good unresolved triplet if $x_i + y_i + z_i \leq 1$ for all i. And by Lemma 5.1.2 every good unresolved triplet (X, Y, Z) will have exactly one point in $Im(f')$, $f'((X, Y, Z))$ such that $x_i + y_i + z_i \leq 1$, thus this triplet gets counted exactly once. To write the number of unresolved triplets in closed-form notation they are split up into four parts; The number $\binom{a_{0,0,...,0}}{3}$ is the number of good unresolved triplets where all three leaves are in region $A_{0,0,...,0}$. The number $\binom{a_{0,0,...,0}}{2} \cdot \sum_{\beta \in B} \beta$ is the number of good unresolved triplets where two leaves are in

region $A_{0,0,\ldots,0}$, and the third leaf is in another region, where B is the set of numbers of leaves in other regions. The number $\binom{a_{0,0,\ldots,0}}{1} \cdot \frac{1}{2}\sum_{\gamma\in C}\gamma$ is the number of good unresolved triplets where one leaf is in region $A_{0,0,\ldots,0}$, and the other two leaves are in two other regions. C is a set of products, we multiply $a_{b_1,b_2,\ldots,b_d}$ and $a_{c_1,c_2,\ldots,c_d}$, where both have at least one index 1 (so they are not $a_{0,0,\ldots,0}$), and they do not have a 1 on the same index (so that they are a good triplet). The way C is defined all products of numbers of leaves will be counted twice, thus we need to divide this sum by 2. The number $\frac{1}{6}\sum_{\delta\in D}\delta$ is the number of good unresolved triplets where no leaf is in region $A_{0,0,\ldots,0}$. D is a set of products, we multiply $a_{b_1,b_2,\ldots,b_d}$, $a_{c_1,c_2,\ldots,c_d}$ and $a_{d_1,d_2,\ldots,d_d}$, where all three have at least one index 1 (so they are not $a_{0,0,\ldots,0}$), and they do not have a 1 on the same index (so that they are a good triplet). The way D is defined all products of numbers of leaves will be counted six times, thus we need to divide this sum by 6.

Now every good triplet has been found per point p, so if we sum over all $p \in Im(f')$ we will count all good triplets. Finally, subtract this number from all possible triplets, and out comes the number of bad triplets. □

5.2. Amount of Subcases when there are k Long Triplets

In previous chapters, we looked at combinations of long and short triplets to count the amount of unresolved good triplets. A case where k caterpillars had a long triplet had a certain amount of subcases, there was a different subcase for each possible combination of regions, such that there are k caterpillars with a long triplet. One might wonder how many subcases there are. The amount of subcases depends on the number of long triplets in that case, for two longs we saw 2 subcases in Chapter 3 and Chapter 4, and for three longs we saw 5 subcases in Chapter 4. In this section, we will first discuss how these cases are structured when a caterpillar with a long triplet gets added, then we will show how many subcases there exactly are.

5.2.1. Structure

Lemma 5.2.1. *If there are d caterpillars, and we look at the case of a long triplet in k caterpillars, where $k < d$, the counts of the subcases are as depicted below:*
The count of one subcase is $\binom{a_{0,0,\ldots,0}}{2} \cdot a_{b_1,\ldots,b_d}$, where $b_i = 1$ if the triplet is long for caterpillar T_i and 0 otherwise. The counts of all other subcases are a combination of three different regions, such that the triplet is still long and unresolved.

Proof. As we have a long triplet in k caterpillars, we need that the k indices (that represent these caterpillars) of the regions of $A_{b_1,b_2,\ldots,b_d}$, $A_{c_1,c_2,\ldots,c_d}$ and $A_{d_1,d_2,\ldots,d_d}$ are 1 exactly once, in other words: for all i where T_i has a long triplet, $b_i + c_i + d_i = 1$. There are two ways this can happen; either the same leaf is on a lower vertex than the root of the triplet in all k caterpillars, or at least two leaves are on a lower vertex in different caterpillars. If the same leaf is on a lower vertex than the root of the triplet in all k caterpillars, then the count for this subcase would be $\binom{a_{0,0,\ldots,0}}{2} \cdot a_{b_1,\ldots,b_d}$, where $b_i = 1$ if the triplet is long for caterpillar T_i and 0 otherwise. If at least two leaves are on a lower vertex in different caterpillars, we get that at most one leaf is in region $A_{0,0,\ldots,0}$, and the other two cannot be in the same region, as then they would both need to have a 1 in the same index, which is not possible by Lemma 5.1.1. However, we need all k indexes to be 1 exactly once, thus the three leaves would be in three different regions. □

Let us define $\#_k$ to be the number of subcases for a long triplet in k caterpillars, with d caterpillars as input.

Observation 5.2.2. *If we now look at the case of a long triplet in $k+1$ caterpillars by adding another caterpillar to the input, we can build that from the case of a long triplet in k caterpillars with d caterpillars as input. We append a 1 to the subscript of one of the three regions that were in the subcase and a 0 to the other two regions. Only in the subcase with $\binom{a_{0,0,\ldots,0}}{2} \cdot a_{1,1,\ldots,1}$ do we not get three new subcases, but only two instead, because adding a 1 to either of the $a_{0,0,\ldots,0}$ will give the same result, thus the number of subcases in the case of $k+1$ long triplets will be $\#_k \cdot 3 - 1$. This is further illustrated in Example 5.2.3.*

Example 5.2.3. *Let us look at case 4 from Figure 3.3. If we want to go from two long triplets to three long triplets, we add another triplet, where any of the three leaves can be on the lower vertex. The leaf that is on the lower vertex in the new triplet will have a 1 appended to its region, while the other two leaves will have a 0 appended. In subcase 4a, if the region of either j or k gets a 1 appended, it will result in the same count, while if a 1 gets appended to the region of i, it will result in a different count from the other two. Thus subcase 4a will result in only two new subcases if we add another long triplet. In subcase 4b it does not matter where we append the 1, because for all three options, we get a different count, so subcase 4b will result in three new subcases when we add another long triplet.*

5.2.2. Amount Of Subcases

Lemma 5.2.4. *If there are d caterpillars with a long triplet in k caterpillars, where $k < d$, the number of subcases is $\frac{3^{k-1}+1}{2}$ for $k \geq 1$.*

Proof. Let us prove this by induction: For 1 long triplet, we have 1 subcase. For 2 long triplets we have $2 = \frac{3^{1-1}+1}{2}$ subcases as seen in Chapter 3.

Assume for a long triplet in k caterpillars the number of subcases is equal to $\frac{3^{k-1}+1}{2}$. By our observation in section 5.2.1. we have that the number of subcases for $k+1$ is equal to $\#_k \cdot 3 - 1$.

$$\#_k \cdot 3 - 1 = \frac{3^{k-1}+1}{2} \cdot 3 - 1 = \frac{3^k+3}{2} - \frac{2}{2} = \frac{3^k+1}{2} = \frac{3^{(k+1)-1}+1}{2} = \#_{k+1}$$

□

5.3. Amount Of Subcases For d Caterpillars

Now that we know how many subcases one gets when one has a long triplet in k caterpillars, we can look at the total number of subcases one gets when there are d caterpillars. We have d positions where a triplet can be long, we want k to be long, so we have $\binom{d}{k}$ combinations of k long triplets. This means we have $\binom{d}{k}\frac{3^{k-1}+1}{2}$ subcases for k long triplets, now we just need to add up all k's.

Lemma 5.3.1. *The total number of subcases when comparing d caterpillars is $\binom{d}{0} + \sum_{k=1}^{d}\binom{d}{k}\frac{3^{k-1}+1}{2}$ for $d \geq 2$.*

Proof. As discussed above, for k long triplets we get $\binom{d}{k}\frac{3^{k-1}+1}{2}$ subcases. Add all of these up for all k between 1 and d and all subcases with at least one long triplet have been counted, now just add the subcase for all shorts and all subcases have been counted. □

5.4. Time Complexity

The purpose of finding this algorithm was to be able to calculate the number of bad triplets d caterpillars have faster than naively checking every triplet. The part that takes the most amount of time is computing how many leaves are in each region $A_{b1,\ldots,b_d}$ for each point in $Im(f')$. An algorithm to find this faster can be found in [3], which tells us counting one region for n leaves takes $O(n\log^{d-2+1/d}(n))$ time, we count for 2^d regions, giving us a time complexity of $O(2^d n\log^{d-2+1/d}(n))$ (if $d = 2$ we get $O(n\log^{1/2}(n))$, which we had before). The biggest issue with this counting algorithm is that it counts for every triplet $\tau \in Im(f')$, which works out in the 2-dimensional case, as there we have $Im(f') \subset Im(f)$. However, In the third dimension and higher $Im(f') \not\subset Im(f)$. This is due to the fact that the third dimension has ghost points.

Definition 5.4.1. *A ghost point g is a point such that $g \in Im(f')$ and $g \notin Im(f)$.*

Example 5.4.2. *Let us look at the caterpillars and triplet $\tau = (i, j, k)$ from Figure 5.1. Here we see that $f(i) = (1,4,2)$, $f(j) = (3,3,2)$ and $f(k) = (3,4,1)$ and with that we can read that $f'(\tau) = (3,4,2)$. It is also observable that neither a, nor b, nor c maps to $(3,4,2)$. In this example, $(3,4,2) \in Im(f')$, but $(3,4,2) \notin Im(f)$, so $(3,4,2)$ is a ghost point. The 3-dimensional visualization of this is illustrated in Figure 5.2.*

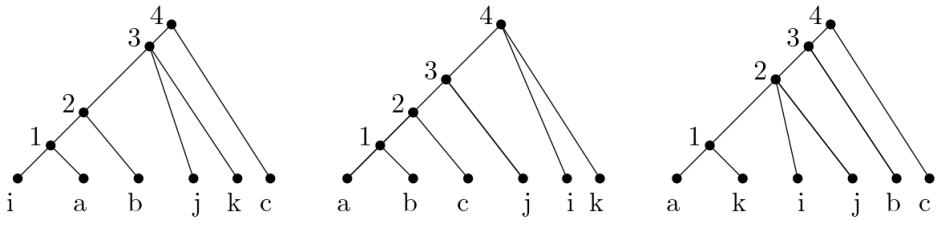


Figure 5.1: Example of three caterpillars with a ghost point

If we now run the algorithm from [3] we have that we will not cover these ghost points and thus we will not count all good triplets. One solution would be to not count for every leaf, but instead for every triplet, but if we count for every triplet we first need to determine $f'(\tau)$ for each triplet τ which already takes $O(dn^3)$

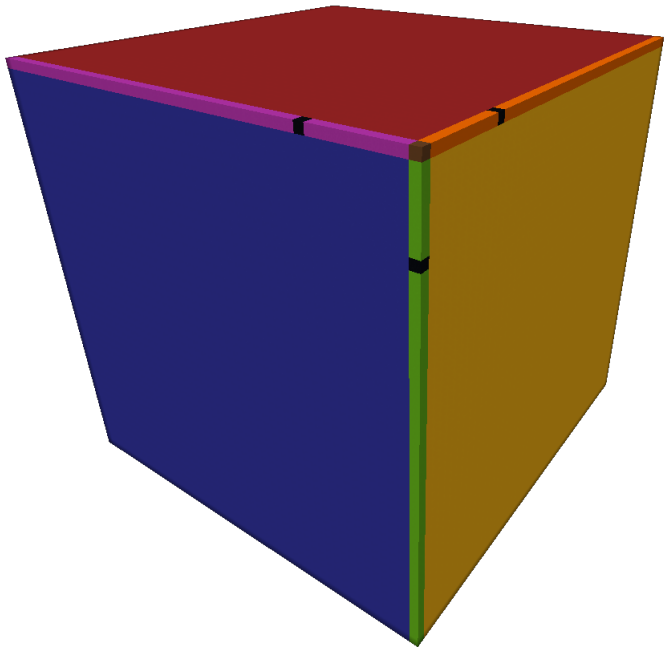


Figure 5.2: Illustration of a ghost point

time, which is the same time complexity as comparing all triplets naively. Another potential solution would be to use some other algorithm to determine all the ghost points, this way we can use the algorithm from [3]. If this ghost point finding algorithm finds χ ghost points in $O(\gamma)$ time, our new algorithm would have a time complexity of $O(\nu \log^{d-2+1/d}(\nu) + \gamma)$, where $\nu = n + \chi$. Such a ghost point finding algorithm that has a time complexity lower than $O(dn^3)$ could be formed in further research, but will not be discussed anymore in this report.

6

Conclusion

In this report, we tried to compute the number of bad triplets there are in a set of d caterpillars in a fast way. This was done by expanding an algorithm designed by Jansson and Lee to work for higher dimensions. First, we map the leaves to a point in the dth dimension, after which we iterate over every point in f' to compute the number of good triplets. This can be done using a closed-form equation, where the number of terms for which the number of leaves needs to be computed is $\binom{d}{0} + \sum_{k=1}^{d} \binom{d}{k} \frac{3^{k-1}+1}{2}$ for $d \geq 2$. This works fine in the second dimension and gives us $O(n\sqrt{\log n})$ as the time complexity, but in higher dimensions, we find an issue due to ghost points. Because we no longer have that $Im(f') \subset Im(f)$, we cannot use the algorithm from [3] directly, so unless we find a way to find all ghost points in less than $O(n^3)$ time, this algorithm will not have a better time complexity than computing the difference naively. Therefore one of the most obvious future steps for this research is to find or develop an algorithm that can find these ghost points as fast as possible.

Another step this research could take is to expand the algorithm to work for other types of trees. One could look at double caterpillars, a tree that can be split up into two caterpillars, or one could look at the distance between a caterpillar and a non-caterpillar, or maybe even the distance between two non-caterpillars. Caterpillars are of course a very specific type of tree, and if we can compute the distance between more generic trees in a faster way, that would be even more useful.

A third way to expand this research is to look at the practicality of the algorithms. If we look at the 2-dimensional algorithm we see that this is theoretically faster than checking all triplets naively, but we of course do not know whether this will be the case in practice. Someone could create a program that applies this algorithm and time it to see if the program computes the distance faster using the naive approach or using the created algorithm.

Bibliography

[1] Jansson, J. & Lee, W.L. (2021). *Fast algorithms for the rooted triplet distance between caterpillars.* Department of Computing, The Hong Kong Polytechnic University.

[2] Dobson, A. J. (1975). *Comparing the shapes of trees.* Springer, Berlin.

[3] Chain, T. M. & Pătraşcu, M. *Counting Inversions, Offline Orthogonal Range Counting,and Related Problems.*

[4] Brodal, G. S. & Fagerberg, R. & Mailun, T. & Pedersen, C. N. S. & Sand, A. (2013). *Efficient Algorithms for Computing the Triplet and Quartet DistanceBetween Trees of Arbitrary Degree.*

[5] Jansson, J. & Rajaby, R. (2017). *A more practical algorithm for the rooted triplet distance.*