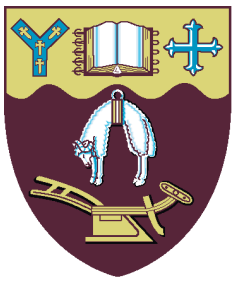
University of Canterbury
Department of Mathematics and Statistics



Evolution of Tandemly Repeated Sequences

A thesis submitted in
partial fulfilment
of the requirements of
the Degree for
Master of Science in Mathematics
at the
University of Canterbury
by

Michael Snook

Supervisor: Assoc. Prof. Charles Semple

2009

Abstract

Despite being found in all presently sequenced genomes, the evolution of tandemly repeated sequences has only just begun to be understood. We can represent the duplication history of tandemly repeated sequences with duplication trees. Most phylogenetic techniques need to be modified to be used on duplication trees.

Due to gene loss, it is not always possible to reconstruct the duplication history of a tandemly repeated sequence. This thesis addresses this problem by providing a polynomial-time locally optimal algorithm to reconstruct the duplication history of a tandemly repeated sequence in the presence of gene loss.

Supertree methods cannot be directly applied to duplication trees. A polynomial-time algorithm that takes a forest of ordered phylogenies and looks for a super duplication tree is presented. If such a super duplication tree is found then the algorithm constructs the super duplication tree. However, the algorithm does not always find a super duplication tree when one exists.

The SPR topological rearrangement in its current form cannot be used on duplication trees. The necessary modifications are made to an agreement forest so that the SPR operation can be used on duplication trees. This operation is called the duplication rooted subtree prune and regraft operation (DrSPR). The size of the DrSPR neighbourhood is calculated for simple duplication trees and the tree shapes that maximize and minimize this are given.

Acknowledgements

Firstly I would like to thank my supervisor Assoc. Prof. Charles Semple for pointing me in the right direction and stopping me from going off on wild tangents. I would also like to thank The Allan Wilson Centre for Molecular Ecology and Evolution and The Marsden Fund for their generous funding and thank Charles once again for finding me this funding.

I would also like the thank John Hawes and Peter Humphries for their help with proof reading various parts of this thesis.

Finally I would like to thank the other postgraduates from the Department of Mathematics and Statistics for keeping me sane during the researching and writing of this thesis. Especially those from the "Masters room".

Contents

Chapter 1

Introduction

Tandem repeats make up a large fraction of most DNA sequences. In particular, it is estimated that more than half of the human genome is made from such repeats [12]. Tandem repeats occur when sections of the genome create copies of themselves and place these copies adjacent to themselves in the genome. This act of creating copies is referred to as a duplication event. Once the copies have been made, they are treated individually; they are free to make copies and undergo mutation. Such repeated sequences are recognized as major generators of novelty in the genome [28]. More importantly, gene families such as immunity related genes occur as tandemly repeated sequences. Understanding tandemly repeated sequences will provide valuable knowledge about these important families of genes.

Despite their importance, tandemly repeated sequences have only just begun to be studied. The duplication model where genomes are created by a series of duplication events was introduced by Fitch in 1977 [13]. This model was based on unequal recombination during meiosis, which is considered to be the mechanism responsible for creating such sequences of genes. However, little more was done until the late 1990s.

The duplication process places a natural linear order on the genome. This order makes the duplication history of the genome more restrictive and gives duplication histories additional structure. Any phylogeny that represents a duplication history must also have this additional structure. Due to this, not every phylogeny can represent the history of a tandemly repeated sequence. Because of this, it is important to have efficient and accurate algorithms to reconstruct the duplication history of tandemly repeated sequences. Several such algorithms have been created over the past ten years [9, 10, 11, 22, 29]. The produced duplication histories are displayed as either rooted or unrooted duplication trees. All duplication trees studied in this thesis will be rooted duplication trees. Both rooted

and unrooted duplication trees have a great deal of structure. This structure gives rise to many interesting biological and mathematical problems.

The number of possible duplication trees on a given number of leaves was enumerated in [15] and a recursive equation was given. This showed that the duplication model is very restrictive. The probability of getting a random phylogeny that is a duplication tree decays quickly as the number of leaves in the phylogeny increases. Because of this, it is unlikely that algorithms and methods designed for phylogenetic trees will output duplication trees. Unless we develop new mathematical tools and algorithms, working with and understanding tandemly repeated sequences will be incredibly difficult. This thesis focuses on developing such new tools and algorithms for several different areas in phylogenetics. Some details of the topics covered in the remaining chapters are discussed below. Unless stated otherwise, the results and algorithms presented in this thesis are original work and to the best of my knowledge new.

Chapter 2 covers the necessary background material for this thesis. Sections 2.1 and 2.2 cover the basics of graphs and trees. Readers familiar with graph theory and phylogenetics will not find anything new in these sections. Section 2.3 introduces duplication trees and their basic properties. Along with these properties, there are a few definitions that the reader will not have come across before.

It is not always possible to reconstruct an accurate duplication history when confined to the duplication model alone. There are other factors that affect the creation of a genome. An example of this is gene inversion. This is a common occurrence in tandemly repeated sequences. Gene inversion occurs when the orientation of the gene in a sequence is changed or inverted. This can make it difficult to reconstruct the duplication history, as orientation of all genes when they are created via a duplication event are the same. A method for inferring the minimum number of such inversions needed to reconstruct the duplication history of sequences created by simple duplication events was presented in [20]. Another factor in the creation of tandemly repeated sequences is gene deletion or gene loss. This is when over time certain segments of a genome can erode away and be lost. Gene loss has been studied as a way of explaining differences in gene and species trees [24].

However, gene loss in the duplication model has received little attention and is the focus of Chapter 3. Gene loss is relatively rare and thus an optimal solution to reconstructing the duplication history of a tandemly repeated sequence allowing for gene loss would have the fewest occurrences of gene loss. It is conjectured that given a phylogeny, finding the optimal duplication history accounting for gene loss is NP-hard and therefore no efficient polynomial time algorithm exists unless $P = NP$. We present a polynomial-time locally optimal algorithm for this problem. This algorithm has been implemented in Java and some preliminary results are given.

In Chapter 4 we consider the problem of where if given a forest of duplication trees whether or not there is a supertree that displays this forest and is itself a duplication tree. Constructing supertrees for forests of trees can often reveal important information that can not be deduced from the forest itself. For ordinary phylogenies this problem has been studied extensively. For example see [1, 5, 6] and the references therein. However, the algorithms presented to solve this problem do not directly apply to duplication trees. Due to the fact that there may be exponentially many supertrees for a given forest, we cannot just run these algorithms and test whether the produced supertree is a duplication tree. To get around this, we present a polynomial-time algorithm that looks for a super duplication tree and, if it finds one, constructs the super duplication tree.

Given a phylogeny, often we want to improve it under some criteria. The first such operation created to do this is called nearest neighbour interchange (NNI) and was presented by Robinson in 1971 [25]. There are numerous other operations for doing this, most of which are modifications of Robinson's original operation. These will be discussed in Chapter 5. However, like the supertree algorithms, these operations can not guarantee that the final tree will be a duplication tree. Some, but not all of these operations can be modified to work with duplication trees. One operation that can be suitably modified is the subtree prune and regraft (SPR) operation. In Chapter 5 we provide the necessary modifications and prove that this modified operation referred to as DrSPR can be applied to all duplication trees. We then go on to calculate the size of the DrSPR neighbourhood for simple duplication trees. Searching the neighbourhood of a tree is a common way of

improving phylogenetic trees.

In Chapter 6 we calculate the simple duplication trees that minimize and maximize the DrSPR neighbourhood. This follows on from Chapter 5. Calculating the extremal cases gives us upper and lower bounds on the size of the neighbourhood.

Chapter 2

Preliminaries

This thesis will be self contained and no prior knowledge of phylogenetics will be assumed.

2.1 Graphs

We begin with a few basic graph theory definitions. These may already be familiar to the reader but an understanding of them is necessary to understand the material presented in this thesis. After this chapter, basic graph theory knowledge will be assumed and used freely throughout the thesis. A more in depth discussion on graphs and trees and how they are used in phylogenetics can be found in [27].

A graph $G = (V, E)$ is a non-empty set V of points called *vertices* and a multiset E of *edges* such that $E = \{\{u, v\} : u, v \in V\}$.

Graphs have a simple visual representation where the vertices are represented with dots and the edges are lines connecting the vertices. As an example, the graph $G = (V, E)$ where $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and

$$E = \{\{1,2\},\{2,3\},\{3,4\},\{4,5\},\{5,6\},\{6,7\},\{7,8\},$$
$$\{8,9\},\{9,10\},\{10,6\},\{1,8\},\{2,10\},\{3,7\},\{4,9\},\{5,1\}\}$$

can be represented with the picture in Figure 2.1. A graph $G = (U \cup V, E)$ is *bipartite* if its vertex set is made from two disjoint sets U and V and there is no edge $e \in E$ such that $e = \{u, v\}$ and $u, v \in V$ or $u, v \in U$. A *loop* in a graph is an edge $e = \{u, v\}$ such that $u = v$. All graphs studied in this thesis will have no loops.

For any edge e, we say that e is *incident* with u and v if $e = \{u, v\}$. Furthermore, we say that u and v are *adjacent*. A *path* is a sequence of distinct vertices $v_1, v_2, \ldots, v_n$ such that v_i is adjacent to v_{i+1} for all $i \in \{1, 2, \ldots, n-1\}$. In the graph in Figure 2.1, $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$ is a path. A *cycle* is a path such that the $v_1 = v_n$. If a graph G

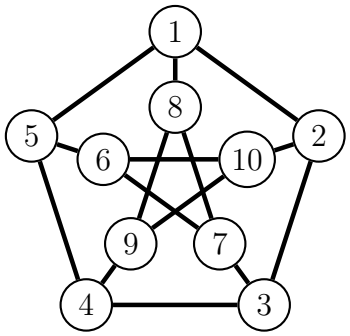


Figure 2.1: Example of a graph.

has no cycles, then we call G *acyclic*. We call a graph G *connected* if for every distinct pair of vertices u and v in V, there is path that includes both of them.

The *degree* of a vertex v, written $\deg(v)$ is the number of edges that v is incident with. For example, in the graph in Figure 2.1 every vertex has degree three. If a vertex v has degree one, then we call v a *leaf*, and the edge incident with v a *pendant edge*. Any edge that is not a pendant edge is an *internal edge*.

A *digraph* is a graph where the edges have a direction associated with them. Pictorially, we draw an arrow from vertex u to v if and only if $(u, v) \in E$. The *indegree* of a vertex v is the number of directed edges $e \in E$ incident with v such that $e = (u, v)$ for $u \in V$. Conversely, the *outdegree* of v is the number of edges $e \in E$ incident with v such that $e = (v, u)$ for $u \in V$. If a digraph is connected and has no loops, then it is a *network*.

A *clique* is a graph in which every vertex is joined with an edge to every other vertex. A graph $H = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. For example, the graph in Figure 2.2 $H = (V', E')$ where $v' = \{1, 2, 3, 4, 5\}$ and $E' = \{\{1, 2\}\{2, 3\}\{3, 4\}\{4, 5\}\{5, 1\}\}$ is a subgraph of the graph G shown above. This is written as $H \subseteq G$. A *tree* is a connected graph with no cycles. Furthermore, given any graph $G = (V, E)$, the following three statements are equivalent. See [27], page 7.

1. G is a tree;

2. for any two vertices u and $v \in V$ there exists a unique path from u to v;

3. G is connected and $|V| = |E| - 1$.

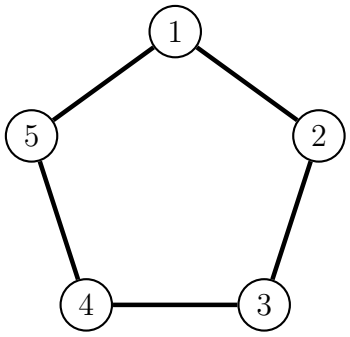


Figure 2.2: Example of a subgraph of the graph in Figure 2.1.

A *forest* is a collection of trees. An example of a forest of two trees can be seen in Figure 2.3. In Figure 2.3, the vertex v is a leaf while the vertex u is an internal vertex.

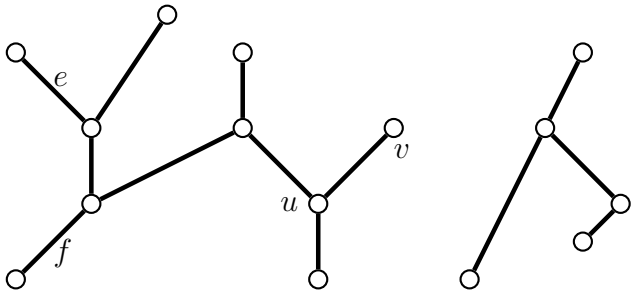


Figure 2.3: Example of a forest containing two trees.

The two edges e and f are both pendant edges.

2.2 Rooted Phylogenetic Trees

A tree is a rooted tree if it has a single vertex distinguished as the root. The root is usually denoted by ρ. A rooted tree is a rooted binary tree if the root has degree two and every other vertex has either degree one or three. A tree can be labeled by giving its vertices labels. For a tree $\mathcal{T}$, the label set denoted $\mathcal{L}(\mathcal{T})$ is the set of labels assigned to vertices of the tree.

A tree $\mathcal{T}$ is a *rooted binary phylogenetic tree* if $\mathcal{T}$ is a rooted binary tree, a vertex is labeled if and only if it is a leaf, every leaf has only one label and no two distinct

vertices have the same label. For ease of reading, in this thesis we will refer to a rooted binary phylogenetic tree with label set X as a rooted phylogenetic X-tree. Rooted binary phylogenetic X-trees will play a central part in this thesis. Unless otherwise stated, all trees will be rooted binary phylogenetic X-trees. Figure 2.4 shows an example of a rooted phylogenetic X-tree. Rooting a tree provides a natural sense of direction of the tree. This

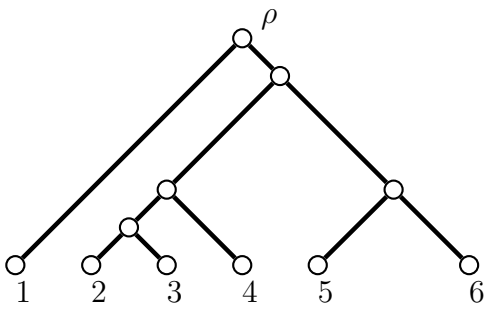


Figure 2.4: An example of a rooted binary phylogenetic X-tree.

allows us to define *ancestors* and *descendants* of vertices. Given a vertex v in a tree, the ancestors of v are the vertices on the unique path from v to the root ρ. Conversely, a vertex u, its *descendants* are all vertices u such that v is an ancestor of u. A vertex is both an ancestor and a descendant of itself. The *most recent common ancestor* of two vertices u and v denoted $mrca(u, v)$, is the vertex w that is an ancestor of both u and v such that no descendant of w is an ancestor of both u and v. These notations are illustrated in Figure 2.5. Given a vertex $v \in X\backslash\{\rho\}$ in a rooted phylogenetic X-tree, define

$$C(v) = \{u : u \text{ is a label of a descendant of } v\}.$$

That is, for a vertex v, $C(v)$ is the set of labels of all vertices that are descendants of v. The set $C(v)$ is called a *cluster*. For a rooted phylogenetic X-tree $\mathcal{T}$, the set of clusters $\{C(v) : v \in V\backslash\{\rho\}\}$ is called the set of clusters induced by $\mathcal{T}$. The tree $\mathcal{T}$ on the left in Figure 2.6 induces the following set of clusters.

$$\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}, \{2, 3\}, \{1, 2, 3\}, \{4, 5, 6\}, \{4, 5, 6, 7\}, \{4, 5, 6, 7, 8\}\}.$$

These clusters are displayed next to their corresponding vertices in the tree on the right in Figure 2.6. Take any phylogenetic X-tree $\mathcal{T}$ and some set $\mathcal{S} \subset X$. The *restriction* of

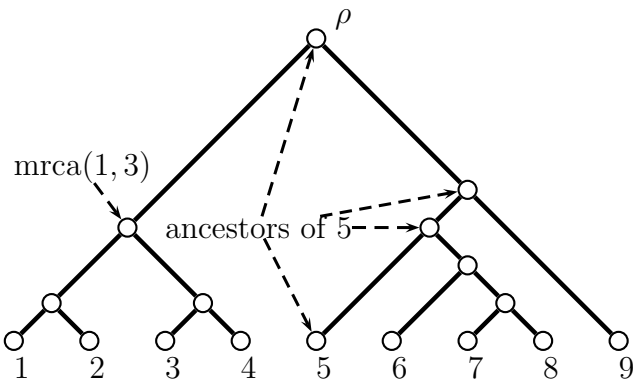


Figure 2.5: A rooted binary phylogenetic X-tree with selected ancestors and descendants marked.

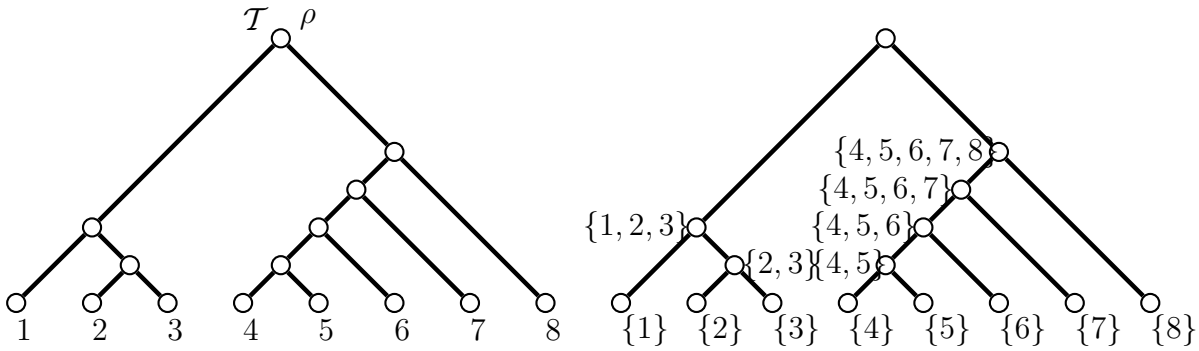


Figure 2.6: Two pictures of a rooted binary phylogenetic X-tree $\mathcal{T}$. The one on the left shows the labeling of the vertices of $\mathcal{T}$, the one on the right showing the clusters induced by the vertices of $\mathcal{T}$.

$\mathcal{T}$ to $\mathcal{S}$, written $\mathcal{T}|\mathcal{S}$ is the smallest subtree of $\mathcal{T}$ connecting all vertices with labels in $\mathcal{S}$, with all degree two vertices that are not the root suppressed. An example of this can be seen in Figure 2.7. Let $\mathcal{T}$ be a binary phylogenetic X-tree. We say a binary phylogenetic

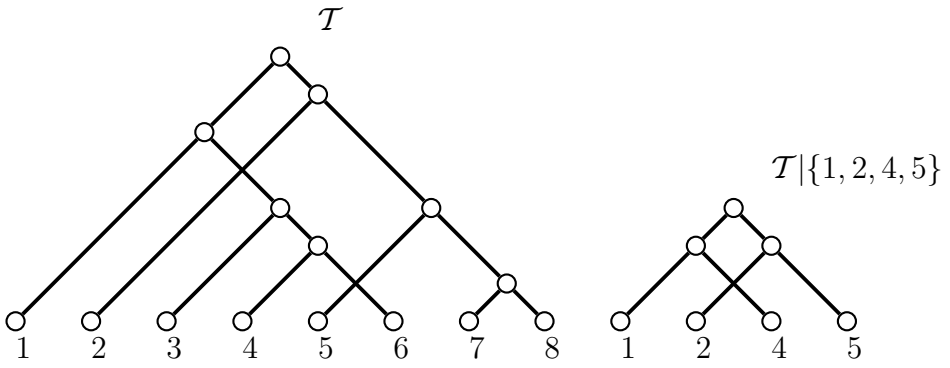


Figure 2.7: A phylogenetic X-tree $\mathcal{T}$ and the restriction $\mathcal{T}|\{1,2,4,5\}$.

X'-tree $\mathcal{T}'$ *displays* $\mathcal{T}$ if $X \subset X'$ and up to degree two vertices, the minimum subtree of $\mathcal{T}'$ that connects the elements of X is isomorphic to $\mathcal{T}$. That is, $\mathcal{T}'$ displays $\mathcal{T}$ if and only if $\mathcal{T} \cong \mathcal{T}'|X$.

Suppose we have two rooted binary phylogenetic X-trees $\mathcal{T}_1$ and $\mathcal{T}_2$. The tree $\mathcal{T}_1$ is a *pendant subtree* of $\mathcal{T}_2$ if $\mathcal{T}_2$ displays $\mathcal{T}_1$ and the set of clusters induced by the root of $\mathcal{T}_1$ in $\mathcal{T}_2$ is the same as the label set of $\mathcal{T}_1$. An example of this is shown in Figure 2.8. The tree $\mathcal{T}_2$ is on the left, while the pendant subtree $\mathcal{T}_1$ is shown on the right.

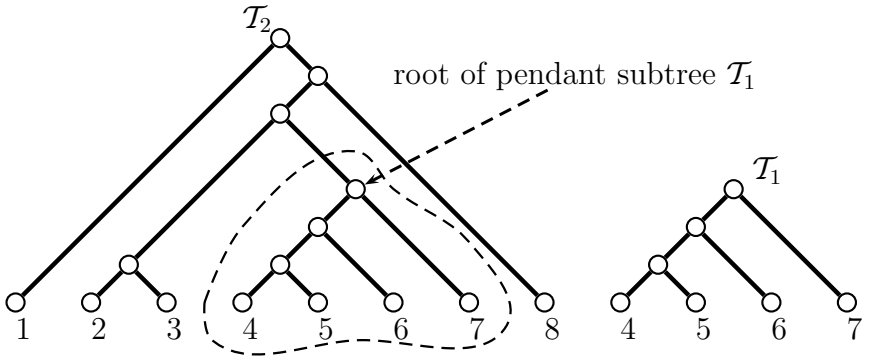


Figure 2.8: Example of a pendant subtree.

Phylogenetic trees are commonly used to represent the evolution of present day species. The leaves of the phylogenetic trees represent the present day species and each internal

vertex represents a common ancestor of all the vertex's descendants. Describing the evolution of species this way makes it easy to see which species are closely related to each other. Figure 2.9 gives an example of such a tree.

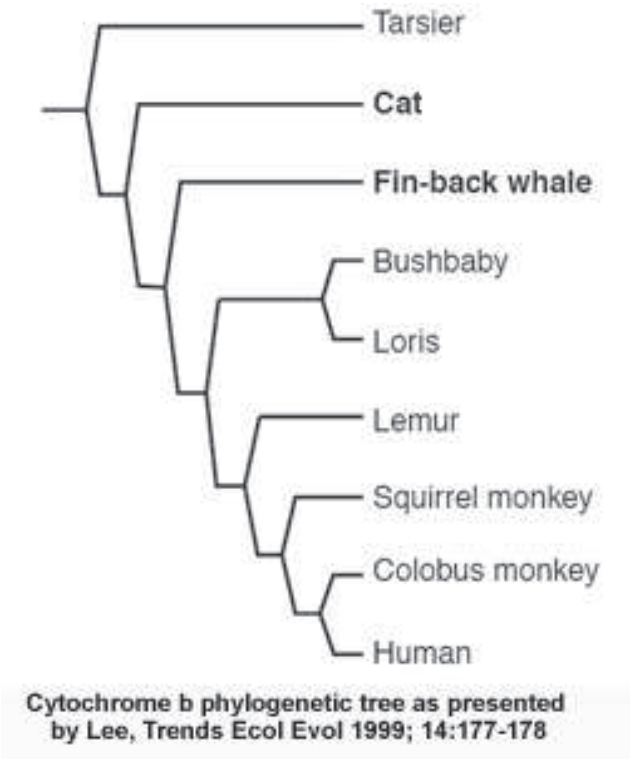


Figure 2.9: Example of a pyhlogenetic tree representing the evolution of present day species.

2.3 Duplication Trees

Duplication trees are the focus of this thesis. Duplication trees are a good way of representing the history of a sequence that was generated by subsequences making copies of themselves. An example of where this happens is in the evolution of the human genome, where sequences of genes will make copies of themselves. These copies are then aligned next to the originals.

Before we define a duplication tree, we will run through an example of how a duplication tree can represent the history of a sequence generated by subsequences making copies of themselves. This can be done as follows: suppose we start with a single element. The first subsequence that makes a copy of itself must be the only element. After the copy is made, it is placed next to the original and is then treated as a completely separate element in the sequence. A example of a duplication tree is shown in Figure 2.10.

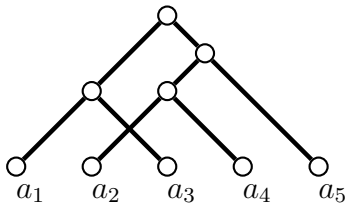
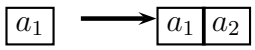
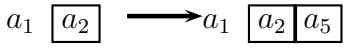


Figure 2.10: A duplication tree generated from a sequence making copies of subsequences.

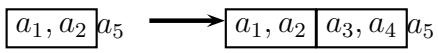
This tree represents the following sequence of copies, where the first box is the element that is copied and the second is the copy. First a_1 makes a copy of itself. This is shown below.



This will give the sequence a_1, a_2. Now suppose that a_2 makes a copy of itself. This will give the following sequence.



Now finally suppose that the subsequence a_1, a_2 makes a copy of itself, to give the sequence below.



The duplication tree that represents this sequence is shown in Figure 2.10. Each internal vertex represents one of the subsequences a_i being duplicated in one of the duplication events.

Given a linear ordering O on a set X, we say $x <_O y$ if $x, y \in X$ and x comes before y in the ordering O. Furthermore, for a subset $X' = \{x_1, x_2, \ldots, x_n\} \subseteq X$, if $x_1 <_O x_2 <_O \ldots <_O x_n$ and there is no element $y \in X, y \notin X'$ such that $x_1 <_O y <_O x_n$, then the sequence $(x_1, x_2, \ldots, x_n)$ is contained in the ordering O. This can be written as $(x_1, x_2, \ldots, x_n) \subseteq O$. For example, given the ordering $O = (1, 2, 3, 4, 5, 6)$, we have $(1, 2, 3, 4) \subseteq O$ and $(1, 2, 4, 5) \not\subseteq O$.

A *cherry* is two leaves of a tree that are adjacent to a common vertex. Cherries will be mentioned frequently throughout this thesis. We will define the cherry $c_i = (l_i, u_i, r_i)$ to be the cherry with leaves l_i and r_i that are incident with the vertex u_i. We may sometimes leave u_i out when it is not important. If we have a phylogenetic X-tree with a linear ordering O on X and two cherries $c_1 = (l_1, u_1, r_1)$ and $c_2 = (l_2, u_2, r_2)$ such that $l_1 <_O l_2 <_O r_1 <_O r_2$, then we say the two cherries *overlap*. Furthermore, we say that c_2 *left overlaps* c_1 or c_1 *right overlaps* c_2. However, if we have two cherries $c_1 = (l_1, u_1, r_1)$ and $c_2 = (l_2, u_2, r_2)$ such that $l_1 <_O l_2 <_O r_2 <_O r_1$ then c_1 *contains* c_2. Lastly, given a cherry $c_i = (l_i, u_i, r_i)$ we define the *width* of c_i, denoted $width(c_i)$, to be $|W|$ where $W = \{x : x \in X, l_i <_O x <_O r_i\}$. That is, the number of leaves between l_i and r_i.

The reverse of a duplication event is a duplication reduction. Suppose we have a rooted binary phylogenetic X-tree $\mathcal{T}$ and a linear ordering O on the label set X. Furthermore, assume that $\mathcal{T}$ has a sequence of cherries

$$(l_i, u_i, r_i), (l_{i+1}, u_{i+1}, r_{i+1}), \ldots, (l_j, u_j, r_j)$$

such that $i \leq j$ and

$$(l_i, l_{i+1}, \ldots, l_j, r_i, r_{i+1}, \ldots r_j) \subseteq O.$$

Let $\mathcal{T}$ be the tree obtained by deleting all leaves $l_i, l_{i+1}, \ldots, l_j, r_i, r_{i+1}, \ldots r_j$ and O' be the ordering obtained by replacing $(l_i, l_{i+1}, \ldots, l_j, r_i, r_{i+1}, \ldots r_j)$ with $(u_i, u_{i+1}, \ldots, u_j)$. We call a reduction of this type a *duplication reduction* and say $\mathcal{T}'$ was obtained from $\mathcal{T}$ via a duplication reduction. When we perform a duplication reduction on a cherry, we say that the cherry has been *eliminated*. Often we refer to this process as reducing the sequence of cherries.

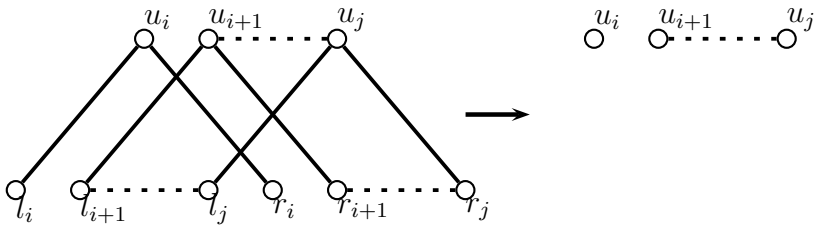


Figure 2.11: A visual representation of a duplication event.

Visually, a duplication reduction will look like Figure 2.11. When we represent the evolution of a sequence with a duplication tree, every non-leaf vertex represents an element that was copied in a duplication event. The event represented by the internal vertex v is the left descendant of v making a copy of itself which becomes the right descendant of v. An example of such an event could be a gene segment making a copy of itself. These duplication events are what we eliminate when we perform a duplication reduction. A duplication event is *visible* if both of its immediate descendants are leaves.

A rooted binary phylogenetic X-tree $\mathcal{T}$ with an ordering O on the set X, denoted by $(\mathcal{T}, O)$, is a *duplication tree* if and only if

1. $(\mathcal{T}, O)$ only contains the root ρ, or

2. there exists a duplication tree $(\mathcal{T}', O')$ such that $\mathcal{T}'$ can be obtained from $\mathcal{T}$ via a duplication reduction.

Note that this definition is only for rooted phylogenetic X-trees. This is because we will only consider rooted trees in this thesis. Every duplication tree considered in this thesis will be rooted. For ease of reading, we will refer to a rooted binary phylogenetic X-tree $\mathcal{T}$ with an ordering O on X as an *ordered phylogeny*. Furthermore, when talking about the ordered phylogeny $(\mathcal{T}, O)$, we will often denote this by just $\mathcal{T}$. That is, the ordered phylogeny $\mathcal{T}$ is the same as the ordered phylogeny $(\mathcal{T}, O)$.

If the tree $(\mathcal{T}, O)$ is a duplication tree, then it does not matter what duplication reduction we perform. The tree obtained after the reduction will always be a duplication tree. Finding such a duplication event to reduce, if one exists, is also very easy. Furthermore, if

$(\mathcal{T}, O)$ is not a duplication tree, then no matter what duplication reductions we perform on $(\mathcal{T}, O)$, we will not be able to reduce $(\mathcal{T}, O)$ to its root.

This definition provides us with a natural algorithm to test whether a given ordered phylogeny is a duplication tree. Simply start performing duplication reductions. If we get to the root ρ, then $(\mathcal{T}, O)$ is a duplication tree. However, if we get to a point where there is no duplication reduction that can be performed, then $(\mathcal{T}, O)$ is not a duplication tree.

Consider the two ordered phylogenies with ordering $O = (1, 2, 3, 4, 5, 6)$ presented in Figure 2.12. The tree $\mathcal{T}_1$ is a duplication tree, while $\mathcal{T}_2$ is not a duplication tree. To see

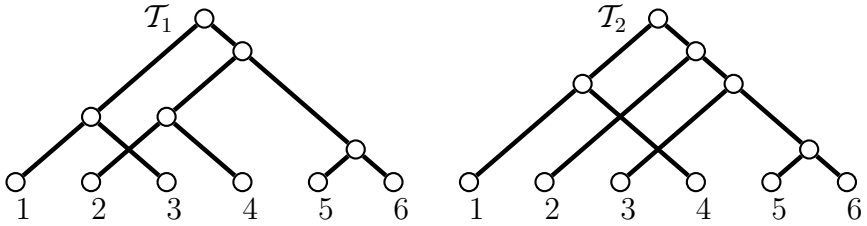


Figure 2.12: Two binary phylogenetic X-trees. The tree $\mathcal{T}_1$ is a duplication tree, while the tree $\mathcal{T}_2$ is not.

this, consider Figure 2.13. To reduce the tree $\mathcal{T}_1$ we can first reduce the cherry $(5, 6)$ then the cherries $(1, 3), (2, 4)$, then the cherry $(2, 5)$ and finally the cherry $(1, 2)$. We have now, through a series of duplication reductions, reduced $\mathcal{T}_1$ to its root and so $\mathcal{T}_1$ is a duplication tree. This reduction of $\mathcal{T}_1$ is shown in Figure 2.13. However, after we reduce $(5, 6)$ in $\mathcal{T}_2$, there is no set of cherries that correspond to a duplication event. Because we cannot reduce $\mathcal{T}_2$ to its root via duplication reductions, $\mathcal{T}_2$ is not a duplication tree. A duplication tree is *simple* if it contains no *multiple duplication event*. That is, no duplication event that involves more then one cherry. A simple duplication event corresponds to a single element in a sequence making a copy of itself, while a multiple event corresponds to a sequence of two or more consecutive elements making a copy of itself. Consider the two duplication trees in Figure 2.14. $\mathcal{T}_1$ is a simple duplication tree, while $\mathcal{T}_2$ is not. The second duplication event in $\mathcal{T}_2$ corresponds to the sequence $1, 2$ making a copy of itself. This is a multiple duplication event and therefore $\mathcal{T}_2$ is not a simple duplication tree. Let

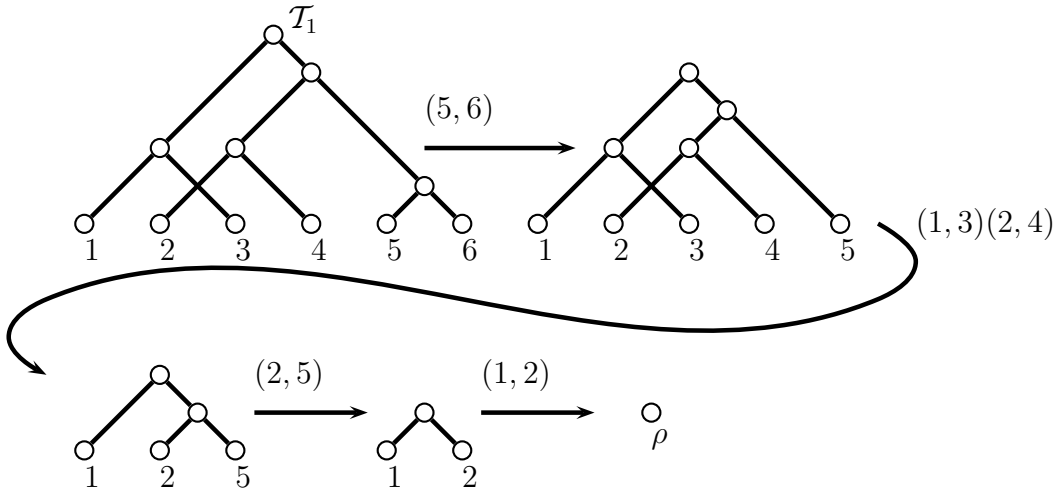


Figure 2.13: The duplication reductions that reduces $\mathcal{T}_1$ from Figure 2.12 to its root.

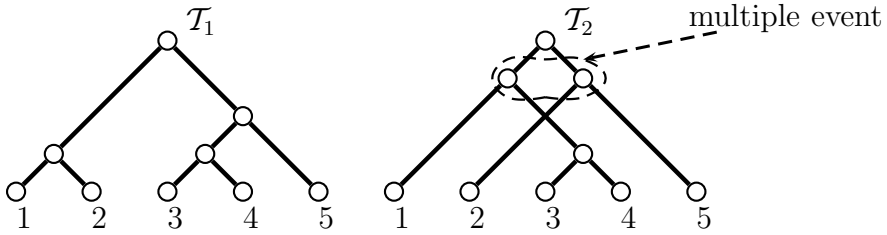


Figure 2.14: Two duplication trees. The tree $\mathcal{T}_1$ is a simple duplication tree and the tree $\mathcal{T}_2$ is not.

O be an ordering on a set X and O' an ordering on a set X' such that $X \subset X'$. We say that O' extends O if for all $x, y \in X$, $x <_O y$ if and only if $x <_{O'} y$. Furthermore, we say the ordered phylogeny $(\mathcal{T}', O')$ extends $(\mathcal{T}, O)$ if $\mathcal{T}'$ displays $\mathcal{T}$ and O' extends O.

Let $(\mathcal{T}, O)$ be a duplication tree. A *floor* i is the ordering of O after i duplication reductions. For example, consider the tree below in Figure 2.15. Note that the floor is

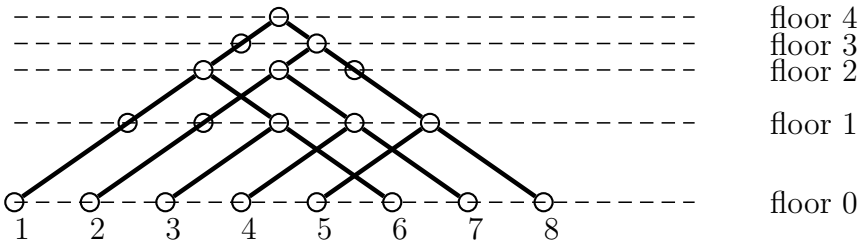


Figure 2.15: A duplication tree with all its floors displayed.

not always unique. Given a duplication tree, there may be two visible duplication events. In this case, there are two possible options for floor 1. This shows that there is not always a unique sequence of duplication reductions that reduces a duplication tree to its root. However, there is always a unique set of duplication reductions that reduces a duplication tree to its root. It is just the order in which these reductions happen that can vary. This can be easily shown by induction. Take a duplication tree with two leaves. Clearly there is only one set of duplication reductions that can reduce this tree. Now assume this is true for all duplication trees that have equal to or less than k leaves. Consider a duplication tree $\mathcal{T}$ with $k+1$ leaves. Assume we have two different sets of duplication reductions, denoted S_1 and S_2 that reduces $\mathcal{T}$ to its root. Without loss of generality, we can assume that S_1 contains every visible duplication event in the tree. If S_2 does not contain a visible duplication event then we cannot reduce $\mathcal{T}$ with the duplication reductions described by S_2 and hence have a contradiction. Therefore the first duplication reduction in S_2 can also be chosen to be the first duplication reduction in S_1. After we perform this duplication reduction, we will have a tree $\mathcal{T}'$ with no more than k leaves and hence, will only have one set of duplication reduction that reduces it to its root by our induction hypothesis.

Chapter 3

Deletion in the Evolution of Duplication Trees

Duplication trees arise from representing the evolution of tandemly repeated sequences. The main mechanism for generating such sequences is by subsequences repeatedly making copies of themselves. However, this is not the only way a DNA sequence can evolve over time. Gene loss plays an important part in the evolution of tandemly repeated sequences. Over the course of time, it is possible for subsequences of the genome to be lost. Once these gene segments are lost, the genome can continue to create tandem repeats like the lost segment was never there. This causes a problem when one tries to reconstruct the duplication history of a gene sequence. If the gene sequence has suffered gene loss, then there may not be a duplication tree that accurately displays the duplication history of the sequence. The DNA sequence has still been generated by tandem repeats, but due to the loss of gene segments, there is no duplication history consistent with the DNA sequence. For example, suppose we have a gene whose history is represented by the duplication tree in Figure 3.1 If, over time the gene segment represented by 4 erodes away, then we would

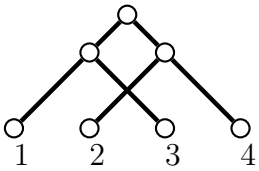


Figure 3.1: A duplication tree representing the duplication history of a gene.

be left with the gene segment $1, 2, 3$ whose history is represented by the tree in Figure 3.2. There is no duplication history consistent with this gene segment. However, gene loss is unlikely and occurs rarely. Given an arbitrary ordered phylogeny, we want to find the minimum number of deletions required to reconstruct a duplication history consistent with the given tree. This should give us a good estimate of the number of deletions that have taken place. We will never be able to know the exact number, but as deletions are

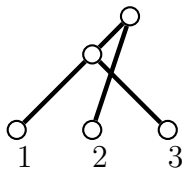


Figure 3.2: The duplication history of the gene in Figure 3.1 after the gene segment 4 has been lost.

rare, the minimum number of insertions needed to explain the duplication history should be very close to the number of deletions that have taken place. If we are given a rooted binary phylogenetic X-tree $\mathcal{T}$ that represents the evolution of a genome, then finding the minimum number of deletions to explain the duplication history corresponds to finding the minimum number of insertions required to reduce $\mathcal{T}$ to its root using duplication reductions.

It should be noted that this is not the same problem as finding a duplication tree $(\mathcal{T}', O')$ that extends $(\mathcal{T}, O)$ such that $|X'| - |X|$ is minimal. While both are interesting problems, this chapter is only concerned with the former. For example, consider the rooted binary phylogenetic X-tree in figure 3.3. The minimum number of insertions

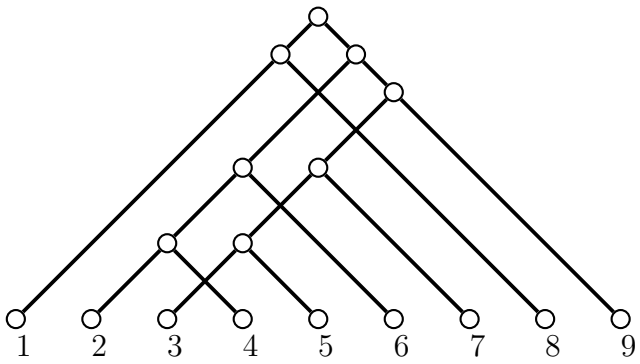


Figure 3.3: A binary phylogenetic X-tree.

required to reduce $\mathcal{T}$ to its root via duplication reductions is one while the minimum number of insertions needed to make $\mathcal{T}$ a duplication tree is two. To see this, consider Figure 3.4. If we add one insertion to $\mathcal{T}$ in the place shown in Figure 3.4, after performing the duplication reductions that reduce the cherries $(2, 4)(3, 5)$ and then $(2, 6)(3, 7)$, we can

reduce the remaining tree to its root. However, $\mathcal{T}$ requires two insertions to make it a duplication tree, if we add an insertion to the same place as we did when reducing $\mathcal{T}$ to its root, the resulting tree will not be a duplication tree. It can be easily checked that adding an insertion to any other edge of $\mathcal{T}$ will not make $\mathcal{T}$ a duplication tree. We need two insertions to make $\mathcal{T}$ a duplication tree. This can be seen in Figure 3.5. It can be easily checked that these two solutions are optimal.

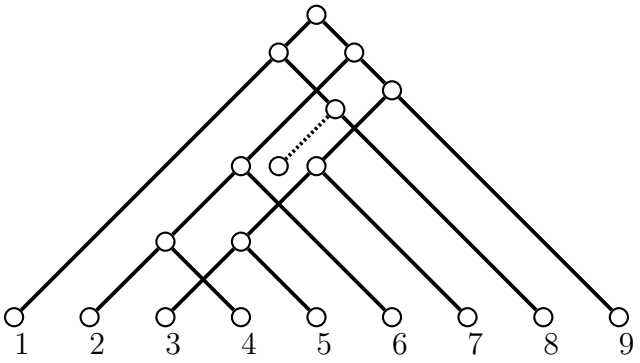


Figure 3.4: The minimum number of insertions required to reduce the rooted binary phylogenetic X-tree in Figure 3.3 to its root via duplication reductions.

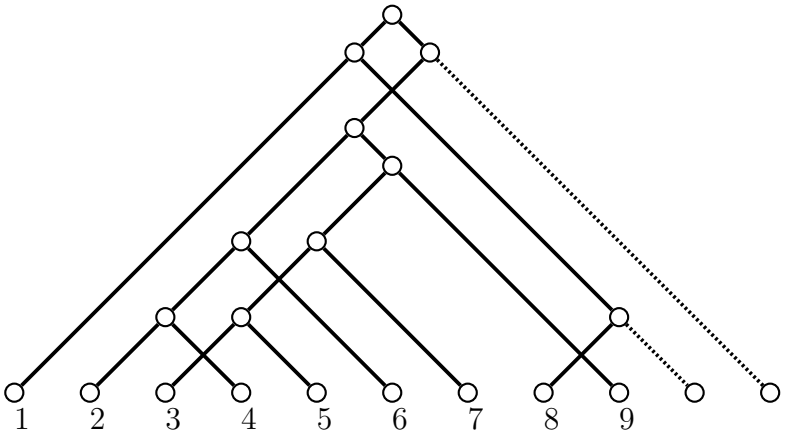


Figure 3.5: The minimum number of insertions required to turn the rooted binary phylogenetic X-tree from Figure 3.3 into a duplication tree.

For clarity, the problem we are interested in can be stated as follows.

Problem: Minimum Insertions and Duplication Reductions.

INSTANCE: An ordered phylogeny $\mathcal{T}$.
QUESTION: What is the minimum number of insertions required to reduce $\mathcal{T}$ to its root via duplication reductions?

The problem of finding the minimum number of insertions needed to explain the duplication history of a sequence can be considered in two ways. Firstly, given an ordered phylogeny, we want to add insertions and perform duplication reductions that reduces the tree to its root with the minimum number of insertions. Secondly, start with the root and perform duplication events and deletions until we have the given ordered phylogeny. We will show that both give the same result in Lemma 3.2 in the next section. It may seem more natural to start with a isolated vertex and perform duplication events and deletions until we reach the desired ordered phylogeny because this is going forward in time. However, when given an ordered phylogeny, it is often easier to add insertions and perform duplication reductions until we have completely reduced the ordered phylogeny.

This chapter is split into three sections. Section 3.1 introduces a number of lemmas that help simplify the problem. In Section 3.2 we introduce a locally optimal algorithm to solve this problem. This algorithm has been implemented in Java. The results of this are covered in Section 3.3.

3.1 Properties of Deletions and Insertions on Ordered Phylogenies

Presented in this section are a number of results that either simplify the problem or provide insight into MINIMUM INSERTIONS AND DUPLICATION REDUCTIONS.

Before we try to solve MINIMUM INSERTIONS AND DUPLICATION REDUCTIONS, we need to answer one important question: given any ordered phylogeny, can we always reduce it to its root with finite number of insertions and duplication reductions? This is answered in Lemma 3.1

Lemma 3.1. *Let $(\mathcal{T}, O)$ be any ordered phylogeny. It is always possible to reduce $(\mathcal{T}, O)$ to its root by inserting a finite number of leaves and performing duplication reductions.*

Proof. To prove this, all we need to show is that given any ordered phylogeny, we can reduce the size of X by at least one, using a finite number of insertions and duplication reductions.

Let $c = (l, u, r)$ be any cherry of $\mathcal{T}$. Any binary X-tree has at least one cherry, so such a cherry always exists. Let n be the width of the cherry c. Let $a_1, a_2, \ldots, a_n$ be the leaves between l and r. For every leaf a_i, insert a new vertex v_i on this leaf and a new leaf w_i incident with this vertex immediately after r in the ordering. This is shown in Figure 3.6, where the dotted lines represent the insertions. This gives the sequence of

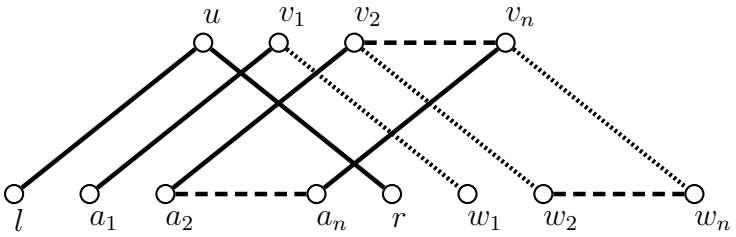


Figure 3.6: An example of reducing a cherry with insertions.

cherries $(l, u, r), (a_1, v_1, w_1), \ldots, (a_n, v_n, w_n)$ such that

$$(l, a_1, \ldots, a_n, r, w_1, \ldots, w_n) \subseteq O',$$

where O' extends O. We can now perform a duplication reduction and replace

$$(l, a_1, \ldots, a_n, r, w_1, \ldots, w_n)$$

with $(u, v_1, \ldots, v_n)$. Let $(\mathcal{T}'', O'')$ be the tree and ordering obtained by this reduction. The size of X has been reduced by one, using a finite number of insertions and a duplication reduction. Furthermore, $(\mathcal{T}, O)$ extends $(\mathcal{T}'', O'')$.

This shows that given any ordered phylogeny, we can reduce the number of leaves by one by a finite number of insertions and a duplication reduction. Therefore any ordered phylogeny can be reduced to its root by a finite number of insertions and duplication reductions.

□

Now we will prove the fact that Minimum Insertions and Duplication Reductions can be thought about and solved in two ways. We will denote the number of insertions in an optimal sequence of insertions and duplication reductions by $\text{opt}(\mathcal{T} \to \rho)$ and the number of deletions in an optimal sequence of duplication events and deletions by $\text{opt}(\rho \to \mathcal{T})$.

Lemma 3.2. *Let $\mathcal{T}$ be an ordered phylogeny. Then*

$$\text{opt}(\rho \to \mathcal{T}) = \text{opt}(\mathcal{T} \to \rho).$$

Proof. Suppose we have an optimal sequence of insertions and duplication reductions that reduces $\mathcal{T}$ to its root with k duplication reductions. If we are at the root ρ, then we can perform the duplication event that takes us to floor $k-1$ and then delete the elements corresponding to insertions in our optimal sequence of insertions and duplication reductions. This will give a sequence of duplication events and deletions that takes us from ρ to floor $k-1$ with the same number of deletions as insertions in the optimal sequence of insertions and duplication reductions that take us from floor $k-1$ to ρ. This will be our base case for induction.

Now suppose we have a sequence of duplication events and deletions that takes us from ρ to floor i with equal to or less deletions as insertions in an optimal sequence of insertions and duplication reductions that takes us from floor i to ρ. Perform the duplication event with the elements of floor i that were roots of cherries reduced by the duplication reduction that takes us from floor $i-1$ to floor i. We can then delete the elements that are not in floor i. This will give us a sequence of duplication events and deletions from ρ to floor $i-1$ with no more deletions than insertions in an optimal sequence of insertions and duplication reductions that takes us from floor $i-1$ to ρ. Therefore by induction,

$$\text{opt}(\rho \to \mathcal{T}) \leq \text{opt}(\mathcal{T} \to \rho).$$

Now assume that we have an optimal sequence of duplication events and deletions that produces the binary phylogenetic X-tree $\mathcal{T}$ and that our sequence has k duplication events.

Consider the duplication event that takes us to floor $k-1$ (this will be the first duplication event). This will consist of no deletions. We can therefore reduce floor $k-1$ to ρ by a single duplication reduction and no insertions. Therefore we can go from floor $k-1$ to ρ with no more insertions than deletions in our optimal sequence of duplication events and deletions that takes us from ρ to floor $k-1$. This proves the base case. Now assume that we have a sequence of insertions and duplication reductions that takes us from floor i to ρ with no more insertions than deletions in our optimal sequence of duplication events and deletions that takes us from ρ to floor i. Take floor $i-1$ and insert the elements that were deleted in our sequence of duplication events an deletions that takes us from floor i to floor $i-1$. Once these elements have be inserted we can perform the duplication reduction of the duplication event that takes us from floor i to floor $i-1$. When this is done, we will have a sequence of insertions and duplication reductions that takes us from floor $i-1$ to ρ with no more insertions than deletions in our optimal sequence of duplication events and deletions that takes us from ρ to floor $i-1$. This shows by induction that

$$\text{opt}(\rho \to \mathcal{T}) \geq \text{opt}(\mathcal{T} \to \rho)$$

and therefore

$$\text{opt}(\rho \to \mathcal{T}) = \text{opt}(\mathcal{T} \to \rho).$$

□

We have now seen that both approaches to solving Minimum Insertions and Duplication Reductions gives us the same solution. Let $\text{opt}(\mathcal{T})$ denote the minimum number of insertions required to reduce an ordered phylogeny $\mathcal{T}$ to its root by duplication reductions.

Lemma 3.3 may seem innocent and relatively straightforward, but it allows us to prove a great deal more interesting results. All results in the remainder of this section rely on Lemma 3.3 in some way.

Lemma 3.3. *Take any duplication event with cherries*

$$(l_1, u_1, r_1), \ldots, (l_n, u_n, r_n).$$

If we restrict the event to any nonempty subset of these cherries, then the cherries still form a duplication event in the restricted tree.

Proof. If a duplication event is made from the cherries

$$(l_1, u_1, r_1), \ldots, (l_n, u_n, r_n),$$

then by definition

$$(l_1, l_2, \ldots, l_n, r_1, r_2, \ldots, r_n) \in O.$$

If we choose a nonempty subset of cherries

$$Q = \{(l_i, u_i, r_i) \ldots, (l_k, u_k, r_k)\},$$

and restrict O to the respective leaves of these cherries and leaves not involved in this duplication event then

$$(l_i, \ldots, l_k, r_i, \ldots r_k) \in O|(O - \overline{Q}),$$

where $O|(O - \overline{Q})$ is the ordering O restricted to the elements in Q and the elements not involved in the duplication event. Therefore the cherries

$$(l_i, u_i, r_i) \ldots, (l_k, u_k, r_k)$$

correspond to a duplication event. □

Given an ordered phylogeny $\mathcal{T}$, it is very easy to reduce it as much as possible with duplication reductions. When this is done, the tree may have a considerably smaller leaf set. If reducing an ordered phylogeny $\mathcal{T}$ with duplication reductions affects the optimal solution to Minimum Insertions and Duplication Reductions, then this makes the problem much more complicated. Fortunately, we can reduce $\mathcal{T}$ as much as possible without affecting the optimal solution to Minimum Insertions and Duplication Reductions on $\mathcal{T}$. This is shown in Proposition 3.6, which relies on Lemmas 3.4 and 3.5. However, before we present these lemmas, we will need some new definitions.

Let $(\mathcal{T}, O)$ be a binary phylogenetic X-tree with an ordering O on the set X. A *partial duplication bipartite graph* is a bipartite graph $G_i = (U \cup V, E)$ such that:

- The vertex set U consists of the roots of all the visible cherries of $(\mathcal{T}, O)$ at floor i, plus all elements of X that are not part of a visible cherry of $(\mathcal{T}, O)$ at floor i.
- The vertex set V consists of the set X at floor i.
- The edge set E joins any two vertices labeled the same in U and V and the root of a visible cherry (in the set U) with its two descendants in V.

Note that because the floor of a duplication tree is not always unique, there may be multiple partial duplication bipartite graphs for a given duplication tree. A *partial duplication graph* is a number of partial duplication bipartite graphs joined together. We denote a partial duplication graph $G_{i,j}$ to be the graph made from joining partial duplication bipartite graphs constructed from floor i to floor j of the duplication tree. Figures 3.7, 3.8 and 3.9 illustrate how we join multiple partial duplication bipartite graphs together.

Consider the duplication tree shown in Figure 3.7. The partial duplication bipartite graphs G_0 and G_1 are shown in Figure 3.8 and the partial duplication graph $G_{0,1}$ is shown in Figure 3.9.

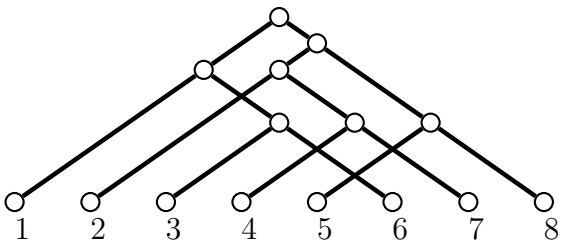


Figure 3.7: A duplication tree.

Note that even though the floor is defined for duplication trees, it still makes sense to talk about the partial duplication bipartite graph G_0 of a phylogenetic X-tree with an ordering on its leaves. This is because we don't need to perform any duplication reductions on the tree to obtain G_0. Furthermore, for some rooted binary phylogenetic X-trees, we can have floors $1, 2, 3, \ldots$ if we can perform enough duplication reductions on the tree

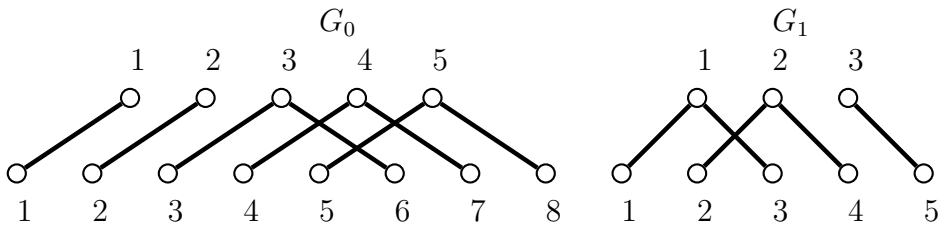


Figure 3.8: The partial duplication bipartite graphs G_0 and G_1 of the tree in Figure 3.7.

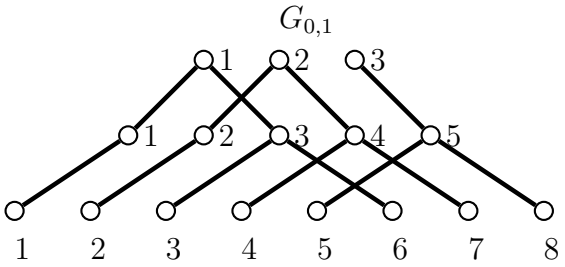


Figure 3.9: The partial duplication graph $G_{0,1}$ of the tree in Figure 3.7.

before we get stuck and can do no more reductions. Let G be a partial duplication bipartite graph that can be completely reduced by a single set on of insertions and a duplication reduction. We will define $\text{opt}(G)$ to be the minimum number of insertions required to reduce G.

Lemma 3.4. *Take any partial duplication bipartite graph G_0 and a set of insertions that allows us to reduce G_0 with a duplication reduction. Let G' be a new partial duplication bipartite graph made by deleting n edges from G that were involved in the duplication reduction of G, but were not part of cherries that were reduced in the reduction of G. Then* $\text{opt}(G') \leq \text{opt}(G_0) - n$.

Proof. Take the duplication bipartite graph G_0 after its insertions but before its reduction. This will be a sequence of cherries. Removing leaves from cherries that are not reduced corresponds to removing cherries from this sequence of cherries in G_0. By Lemma 3.3, this is still a sequence of cherries that corresponds to a duplication reduction. Therefore if we add the same insertions to G' we can still reduce it using a duplication reduction.

Since we removed n leaves that had insertions, there are n less insertions in G'. Therefore $\text{opt}(G') \leq \text{opt}(G_0) - n$. □

Suppose we have a cherry in a rooted binary phylogenetic X-tree. If we add insertions and perform a duplication reduction such that after the event, both leaves of the cherry have swapped sides of a leaf in the ordering, then we say the leaf and the cherry have been *swapped*. Furthermore, if the leaf is replaced with a cherry then we say that the two cherries have been swapped. For example, the insertions and duplication reduction in Figure 3.10 will swap the cherries (l_1, r_1) and (l_2, r_2).

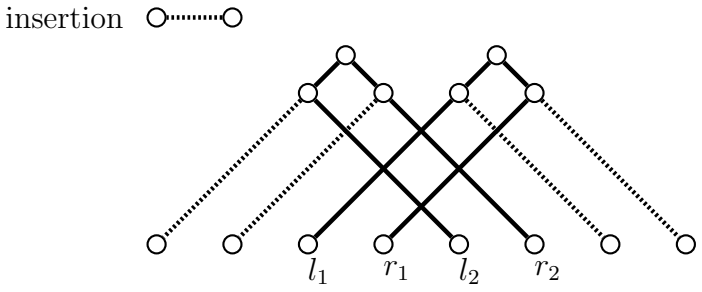


Figure 3.10: Swapping two cherries using insertions.

Recall that two cherries, $c_1 = (l_1, r_1)$ and $c_2 = (l_2, r_2)$, of an ordered phylogeny are said to overlap if $l_1 <_O l_2 <_O r_1 <_O r_2$. If we have two overlapping cherries, then we say these have been swapped if we add a number of insertions and perform a number of duplication reduction so that we end up with $l_2 <_{O'} r_2 <_{O'} l_1 <_{O'} r_1$ in the new ordering O'. However, Lemma 3.5 will show that we cannot do this with a single duplication reduction.

Lemma 3.5. *Suppose we have two cherries $c_1 = (l_1, r_1)$ and $c_2 = (l_2, r_2)$ that overlap each other. It is not possible to swap them with a single set of insertions and a duplication reduction.*

Proof. Because the cherries overlap, we know that if we restrict the ordering to just the leaves of the cherries we will have l_1, l_2, r_1, r_2. If we want to swap the cherries then we need to add insertions so that r_1 is the left leaf of a cherry and r_2 is the right leaf of a cherry. Because we are using a duplication reduction, every leaf that comes before

a left leaf must also be a left leaf in the duplication event. But then l_1 and l_2 would both be left leaves and therefore not be able to swap places. This would give us the ordering r_2, l_1, l_2, r_1 and hence the two cherries have not swapped because l_1 and l_2 have not swapped places. Therefore if c_1 and c_2 overlap, we cannot swap them using a set of insertions and a duplication reduction. □

Proposition 3.6. *Suppose* $(\mathcal{T}, O)$ *and* $(\mathcal{T}', O')$ *are two ordered binary phylogenetic* X*-trees such that* $(\mathcal{T}', O')$ *was obtained from* $(\mathcal{T}, O)$ *via a duplication event. Then* $\text{opt}(\mathcal{T}, O) = \text{opt}(\mathcal{T}', O')$.

Proof. Take an optimal sequence of duplication events and deletions that produces $(\mathcal{T}, O)$. Now do the final duplication event that gives $(\mathcal{T}', O')$. The final step had no deletions, therefore $opt(\mathcal{T}, O) \geq opt(\mathcal{T}', O')$.

We will refer to the cherries made by the duplication event that takes $\mathcal{T}$ to $\mathcal{T}'$ as the extra cherries. From Lemma 3.4 we know that any duplication reduction can be done in $\mathcal{T}'$ for equal to or less insertions than in $\mathcal{T}$ unless an extra cherry is reduced. We will show that if any duplication reduction can be done in $\mathcal{T}'$ with less insertions (by reducing an extra cherry instead of a leaf), then to get the extra cherry into a position to do this we would have needed extra insertions not used in the sequence of insertions and reductions on $\mathcal{T}$ which can now be used to make the leaf in $\mathcal{T}$ a cherry and therefore use no more insertions.

If any extra cherry is reduced in a duplication reduction involving leaves that are not extra cherries then the cherry would have required a number of insertions to get itself into the required position. This is because the extra cherries start out as a duplication event. These extra cherries will need insertions and duplication reductions so that they overlap the leaves not part of the extra cherries. If the extra cherries overlap but do not swap places then we do not have to do anything in $\mathcal{T}$ until the cherry is reduced. This is because if the extra cherries do not swap places then, as both their leaves started on one side of the leaves that are not an extra cherry, after the insertions and reductions there will still be a leaf on the starting side. It will require at least two insertions to make the

extra cherry overlap the leaf. In the tree $\mathcal{T}$ if we only reduce any non extra cherries in the series of insertions and reductions, when it comes time to perform the duplication reduction that reduces the extra cherry, we will have spare insertions. If we take one of these insertions and insert it in the correct place to make the extra cherry, then we can perform the same reduction with no more than $k + e$ insertions where k is the number of insertions in the event in $\mathcal{T}'$ and e is the number of extra cherries reduced in the event. Every extra cherry reduced in the reduction will have required insertions to get it into a position to be reduced which would not have been needed in $\mathcal{T}$. So when we reduce in $\mathcal{T}$ we will have used no more insertions in total then used in $\mathcal{T}'$. This covers the cases where the extra cherries are not swapped with anything.

If any extra cherry $c_e = (l_e, r_e)$ swaps its position with a non extra cherry $c_n = (l_n, r_n)$ then it will need at least $4 + d$ ($3 + d$ if an extra cherry is swapped with a leaf) insertions where if $c_e < c_n$, then d is the number of leaves not part of the swapped cherries between l_e and r_n, else d is the number of leaves not part of the swapped cherries between l_n and r_e. However if we do the same swap in $\mathcal{T}$ (with the root of c_e) and and c_n then we will only require $3 + d'$ ($2 + d'$ if we are swapping two leaves) insertions where $d' \leq d$.

If extra cherries swap places and no extra cherries are reduced in the swap then, they will need at least 2 insertions so they are no longer overlapping (because of Lemma 3.5) and then $4 + d$ insertions where d is as above. In $\mathcal{T}$, we can swap the roots of these with only $2 + d'$ insertions.

If an extra cherry is reduced in a reduction that swaps extra cherries then, either it required an insertion to get it into the appropriate place and we would therefore have an insertion spare to use in $\mathcal{T}$ to create the same cherry and hence use no more insertions, or it overlaps both cherries that are being reduced. There are two possible cases where this can happen. Consider Figure 3.11, where u_0 represents any number of extra cherries that are reduced and did not need insertions to get into position. The edge with b can either have an insertion between 4 and 5 or have no insertion. However, whether or not b has an insertion makes no difference and hence we can treat these two cases as the same case. After the reduction we would have the the cherries shown in Figure 3.12. Because u_0 had

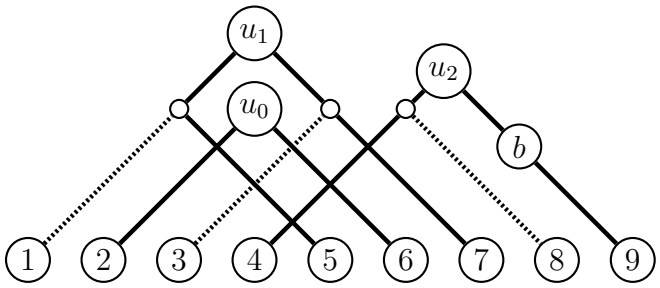


Figure 3.11: Reducing extra cherries while swapping extra cherries.

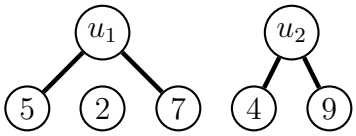


Figure 3.12: The cherries after the reduction of the event in Figure 3.11.

no insertions to get into position, we know that $u_0 <_O u_1$. If $u_1 <_O u_2$, then we do not need to do anything in $\mathcal{T}$ to get the order $2, u_1, u_2$. We used at least five insertions (two to get the cherries into position and three in the reduction) in $\mathcal{T}'$ and will therefore have more than enough free insertions to recreate the cherries shown in Figure 3.12 (if needed) when it comes time to reduce them.

If $u_2 <_O u_1$, then we need the order $2, u_2, u_1$ in O. This can be achieved with two insertions leaving three free insertions that can be used to create the cherries $(5, 7)$ and $(4, 9)$ if needed when we reduce them in $\mathcal{T}'$.

Any event that involves only extra cherries and does not swap any cherry can be ignored in $\mathcal{T}$. We have shown that any duplication reduction that reduces extra cherries and involves leaves not part of the extra cherries can be done in $\mathcal{T}$ with equal to or less insertions and any event that swaps cherries can also be done in $\mathcal{T}$ with equal to or less insertions. This shows that $\text{opt}(\mathcal{T}) \leq \text{opt}(\mathcal{T}')$ and therefore

$$\text{opt}(\mathcal{T}) = \text{opt}(\mathcal{T}').$$

□

Proposition 3.6 shows that if we are looking for an optimal sequence of insertions and duplication reductions that takes an ordered phylogeny to its root, it is always best to reduce any visible duplication events whenever possible. This is good news because it is easy to reduce a tree as much as possible and there is always a unique set of reductions to do this. When this is done, we will be working with a smaller tree making computation easier.

Intuitively, one would expect that if we have an ordered phylogeny that is made up from two non overlapping pendant subtrees, then we could treat them as separate trees and work on them individually. Theorem 3.7 shows that our intuition is correct.

Theorem 3.7. *Suppose we have the ordered phylogeny* $(\mathcal{T}, O)$ *shown in Figure 3.13, where* $O = O_1, O_2$*. Then*

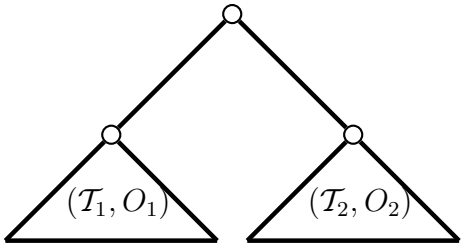


Figure 3.13: A duplication tree partitioned into two subtrees.

$$\text{opt}(\mathcal{T}) = \text{opt}(\mathcal{T}_1) + \text{opt}(\mathcal{T}_2).$$

Proof. Suppose we have a tree $\mathcal{T}$ with the property stated above and that we have optimal sequences of insertions and duplication reductions that completely reduce the subtrees $(\mathcal{T}_1, O_1)$ and $(\mathcal{T}_2, O_2)$. If we take our optimal sequence for the subtree $(\mathcal{T}_1, O_1)$ and completely reduce it followed by completely reducing the subtree $(\mathcal{T}_2, O_2)$ with its optimal sequence, then we have reduced the tree $\mathcal{T}$ to a single cherry. Once this is done, we can reduce the cherry with a final duplication reduction. This last reduction will not have required an insertion and therefore we have completely reduced $\mathcal{T}$ with no more insertions than in our optimal sequence for $(\mathcal{T}_1, O_1)$ plus the insertions in our optimal sequence for

$(\mathcal{T}_2, O_2)$. This shows that

$$\text{opt}(\mathcal{T}) \leq \text{opt}(\mathcal{T}_1) + \text{opt}(\mathcal{T}_2).$$

To prove the other direction we will show by induction that, given an optimal sequence of duplication events and deletions that produce the tree $\mathcal{T}$, we can separate the events into duplication events that only involve descendants of either the root of $\mathcal{T}_1$ or the root of $\mathcal{T}_2$ without increasing the number of necessary deletions.

Suppose we have the tree $\mathcal{T}$ and an optimal sequence of deletions and duplication events that gives the tree $\mathcal{T}$ after k duplication events. For clarification, when we talk about floor j in this proof, we will be referring to floor j of this optimal sequence. Note that because of the way the floor is defined, the first duplication event and set of deletions will take us from floor k to floor $k-1$.

Take the cherry where the left leaf is the root of the tree $(\mathcal{T}_1, O_1)$ and the right leaf is the root of the tree $(\mathcal{T}_2, O_2)$. The first duplication event in our optimal sequence will create this cherry. Take the first duplication event and set of deletions after this cherry has been formed in our optimal sequence of duplication events and reductions. If this event involves both roots, then split this up into two separate duplication events and deletions. One involving only the root of $(\mathcal{T}_1, O_1)$ and the other only involving the root of $(\mathcal{T}_2, O_2)$. From Lemma 3.3, we know that restricting the tree to descendants of only one of $(\mathcal{T}_1, O_1)$ or $(\mathcal{T}_2, O_2)$ we still have a duplication event. Because of this, we know that splitting it up in this way gives two duplication events. If the duplication event was only a simple duplication event then, the leaves of $\mathcal{T}_1$ and $\mathcal{T}_2$ will not be overlapping after the duplication event and before the deletions. If the duplication event involved the roots of both trees then, when we replace it with two separate events and sets of deletions, before we do the deletions, because each root is now involved in different duplication events, none of the descendants of $\mathcal{T}_1$ will overlap any descendant of $\mathcal{T}_2$.

Either way, we have a series of duplication events and deletions that take us to floor $k-2$ such that the total number of deletions is equal to the number of deletions in our optimal sequence of duplication reductions and deletions on $\mathcal{T}$ and at no point does any descendant of $\mathcal{T}_1$ overlap any descendant of $\mathcal{T}_2$. This will be the base case for our

induction.

Now suppose that any optimal sequence of duplication events and deletions that takes us from floor k to floor $k-i$, where $i \geq 2$ can be done such that no duplication event involves descendants of the roots of both $\mathcal{T}_1$ and $\mathcal{T}_2$ and that the total number of deletions in this sequence is the same as our optimal sequence on $\mathcal{T}$ from floor k to $k-i$. Consider the next duplication event in our optimal sequence. Using the same method as our base case, we can separate this duplication event out into no more than two duplication events such that only descendants of the root of $\mathcal{T}_1$ are involved in one and only descendants of the root of $\mathcal{T}_2$ are involved in the other with the total number of deletions for the two events remaining the same as the number of deletions in the duplication event and set of deletions in our optimal sequence. Thus we can no go from floor k to floor $k-i-1$ with no duplication events involving descendants of the roots of both $\mathcal{T}_1$ and $\mathcal{T}_2$ with the total number of deletions equal to the number of deletions in our optimal sequence up to floor $k-i-1$.

This proves by induction that given an optimal sequence of duplication events and deletions that produces the tree $\mathcal{T}$, we can split the duplication events up into two sets of duplication events. One acting only on the descendants of the root of $\mathcal{T}_1$ and the other acting only on the descendants of the root of $\mathcal{T}_2$. This gives us a sequence of duplication events and deletions that produces $\mathcal{T}_1$ and a sequence of duplication events and deletions that produces $\mathcal{T}_2$ such that the combined number of deletions is equal to or less then the number of deletions in an optimal sequence of duplication events and deletions that produces the tree $\mathcal{T}$. That is we have sequences of duplication events and deletions that produce $\mathcal{T}_1$ and $\mathcal{T}_2$ such that

$$\text{opt}(\rho \to \mathcal{T}) \geq \text{opt}(\mathcal{T}_1) + \text{opt}(\mathcal{T}_2).$$

Lemma 3.2 states that $\text{opt}(\rho \to \mathcal{T}) = \text{opt}(\mathcal{T})$ and as we have previously shown that $\text{opt}(\mathcal{T}) \leq \text{opt}(\mathcal{T}_1) + \text{opt}(\mathcal{T}_2)$ we conclude that

$$\text{opt}(\mathcal{T}) = \text{opt}(\mathcal{T}_1) + \text{opt}(\mathcal{T}_2).$$

□

Theorem 3.7 can be generalized to any number of non-overlapping subtrees. Using this, we can break the problem down into smaller problems that are easier to solve.

Problems like Minimum Insertions and Duplication Reductions are categorized into *complexity classes* depending on how complex the problem is to solve. The most well known classes are P and NP where the class P is the set of decision problems that are solvable in polynomial time by a deterministic algorithm. When we say solvable in polynomial-time we mean that the number of steps required to solve the problem is bounded by some polynomial in the size the input. For example the size of the input for a graph will probably be the number of vertices. The class NP is the set of decision problems that can be solved in polynomial-time by a nondeterministic algorithm. Whether or not P=NP is not yet known. Currently, there are many problems in NP that have no known deterministic polynomial-time algorithm to solve them. An optimization problem like Minimum Insertions and Duplication Reductions is referred to as NP-hard if it is provably at least as hard to solve as any problem in NP. For detailed information on complexity classes see [3, 14].

Conjecture 3.8. *The problem* Minimum Insertions and Duplication Reductions *is NP-hard.*

It seems likely that this problem is NP-hard due to the fact that often we have many choices of insertions to make and it is not possible to know the effect of these insertions until possibly many iterations later. Furthermore, for a certain class of ordered phylogenies, it is possible to construct graphs which have a vertex covering of size k if and only if $opt(\mathcal{T}) = k$ for the ordered phylogeny $\mathcal{T}$. However, whether finding a minimum vertex covering on this class of graphs is NP-hard is unknown.

3.2 Locally Optimal Algorithm

If Conjecture 3.8 is true, then there probably does not exist any efficient algorithm to find an optimal solution to this problem. Because of this, we are interested in finding other methods of solving this problem that are as close to optimal as possible. One such

method is to optimally solve the problem locally. This does not guarantee an overall optimal solution but can give solutions close to optimal. We present a polynomial-time algorithm that at each step reduces the set of cherries that optimizes the ratio of number of insertions required to the number of cherries eliminated. The method for this algorithm is based on the problem of finding a maximum set of a permuted ordering that is still in correct order. Finding such a set can be achieved by creating a permutation induced digraph and finding a maximum independent set of this graph. Section 3.2.1 will cover the permutation induced digraph and how it can help obtain locally optimal solutions. We will then present a locally-optimal algorithm in Section 3.2.2.

3.2.1 The Permutation Induced Digraph

Suppose we have an ordered sequence $(1, \ldots, n)$. Then a *permutation* of this sequence is a rearrangement of the elements in the sequence. An example of a permutation of $(1 \ldots, 10)$ is $(1, 2, 3, 6, 4, 5, 8, 10, 9, 7)$. Take any sequence S. We call S' a *subsequence* of S, written $S' \subseteq S$, if S' can be obtained from S by deleting a number of elements from S without changing the order of any element.

Let $\pi(O)$ be a permutation of the ordered sequence $O = (1, \ldots, n)$. A *permutation induced digraph* $D(\pi)$ for the permutation π is the digraph with vertices $(1, \ldots, n)$ and directed edges (u, v) between all elements u and v such that $u >_{\pi(O)} v$ and $u <_O v$.

As an example, the graph in Figure 3.14 is the permutation induced digraph for the permutation $(1, 2, 3, 6, 4, 5, 8, 10, 9, 7)$. For any graph $G = (V, E)$, a *vertex cover* is a set of vertices C such that for all $e = \{u, v\} \in E$, either $v \in C$ or $u \in C$. Conversely, an *independent set* of a graph is a set of vertices W such that for any edge $e = \{u, v\} \in E$, the vertices u and v are not both in w. These two are closely related. A minimum vertex cover is a vertex cover such that $|C|$ is the smallest possible on G. The complement of a minimum vertex cover is a maximum independent set, where a maximum independent set is an independent set such that $|W|$ is maximum.

We can use a vertex cover to find a set of elements from the permutation that are still in correct order. Lemma 3.9 will show that a minimum vertex cover on a permutation

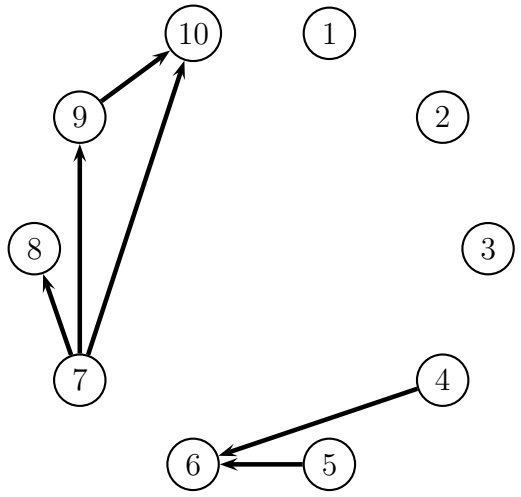


Figure 3.14: The permutation induced digraph for the permutation $(1, 2, 3, 6, 4, 5, 8, 10, 9, 7)$.

induced digraph will give us the minimum number of elements such that when removed, the resulting permuted ordering is a subsequence of the original sequence represented by the permutation induced digraph.

Lemma 3.9. *Let $\pi(O)$ be a permutation of the sequence $O = (1, \ldots, n)$. The minimum number of elements whose removal gives a subsequence $\pi'(O)$ of the permuted sequence $\pi(O)$ that is a subsequence of O is the same as a minimum vertex covering on the graph $D(\pi)$*

Proof. Suppose we have a minimum vertex covering on this graph. If we delete all vertices in the covering, the graph will be a collection of isolated vertices and will therefore have no edges. There is an edge between two vertices if their corresponding elements in the sequence are out of order with each other in the permuted sequence. If there are no edges then no elements in the sequence are out of order with each other. This would mean that the sequence is a subsequence of O. If we remove every vertex in a vertex covering, then the remaining graph is a collection of isolated vertices and therefore the remaining

vertices correspond to a subsequence of the permuted sequence that is also a subsequence of O. Thus, the number of elements in a minimum vertex covering of the graph is equal to or greater than the number of elements you need to remove to give a subsequence of $\pi(O)$ that is also a subsequence of O.

Now, suppose we have our set of elements whose removal gives a subsequence of O. If we remove these elements from our digraph then the digraph will have no edges. (If it did, it would mean that the resulting sequence was not a subsequence of O). Because the removal of these vertices removed all edges, all the edges in the digraph must have been incident with at least one of the vertices removed. Therefore the set we removed gave us a vertex covering.

Hence, the minimum number of elements whose removal gives a subsequence of O is equal to or greater then a minimum vertex covering. Therefore a minimum vertex covering on our constructed digraph corresponds to the elements whose removal puts our gives a subsequence of O.

□

We will show how the maximum number of elements from a permuted ordering that are still a subsequence of the original ordering helps us find a locally optimal solution later in this section. Before we do that, we will show how we can find such a maximum subsequence.We do this by finding a maximum independent set of the permutation induced digraph. Because the complement of a minimum vertex covering is a maximum independent set Lemma 3.9 can be restated as: the maximum number of elements in a permuted ordering $\pi(O)$ that are also a subsequence of O is the same as a maximum independent set in the permutation induced digraph $D(\pi)$.

Take any permutation induced digraph. The first thing to note is that this graph is *transitive*. That is, if there is an edge (u, v) and an edge (v, w), then there is an edge (u, w). Another thing to note is that there are no directed cycles. If there was a directed cycle, then we would have $u > v > u$. This is clearly impossible. These two facts make finding a maximum independent set easy. We can do this by finding the minimum number of *clique paths* required to cover the permutation induced digraph. In transitive graphs,

cliques are often called *clique paths*. If a vertex v is in a clique path, then we say that the clique path *covers* v. For example, the graph in Figure 3.14 has the following maximal clique paths.

$$\{1\}, \{2\}, \{3\}, \{4, 6\}, \{5, 6\}, \{7, 8\}, \{7, 9, 10\}.$$

It was shown in [16] that the minimum number of clique paths required to cover every vertex in the graph is the same as the size of a maximum independent set. A method for finding the maximum-weighted independent set for a transitive graph was presented in [19] based on methods for solving the minimum-cost flow problem. It uses the minimum-cost flow solution to find the minimum number of clique paths to cover the graph. The minimum-cost flow problem is: given a weighted network where each vertex v has a weight $w(v)$ and each edge e has a cost per unit of flow $c(e)$ through it and a maximum amount of flow allowed through it called the capacity, find the flow that minimizes the total cost such that each vertex has at least $w(v)$ units of flow through it and no edge has more units of flow through it than its capacity. This area has been studied extensively and numerous polynomial-time algorithms exist. For a good introduction to network flow problems, see [21].

Given any permutation induced digraph, we can turn it into a minimum-cost flow problem by adding two vertices: a sink node and a source node. To do this, we remove all vertices of degree zero, join all vertices with indegree zero and outdegree greater than zero with a directed edge coming from the source node and all vertices that have outdegree zero and indegree greater than zero with a directed edge to the sink node. For example, the permutation induced digraph from Figure 3.14 would give the network in Figure 3.15, where the flow requirement of each vertex is 1 and each edge has infinite capacity. An optimal solution to the minimum flow problem on this network is shown below in Figure 3.16, where the dotted lines represent an edge with two units of flow and the filled in lines represent edges with one unit of flow. We see that an optimal solution requires four units of flow. Indeed, a maximum independent set for the graph in Figure 3.14 minus the vertices of degree 0 has four vertices. One such solution is the set of vertices $\{4, 5, 8, 9\}$.

If we were worried about the size of the network, we could take any elements that

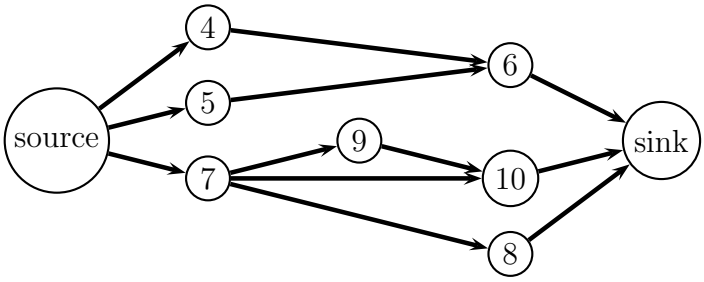


Figure 3.15: The permutation induced digraph from Figure 3.14 turned into a network.

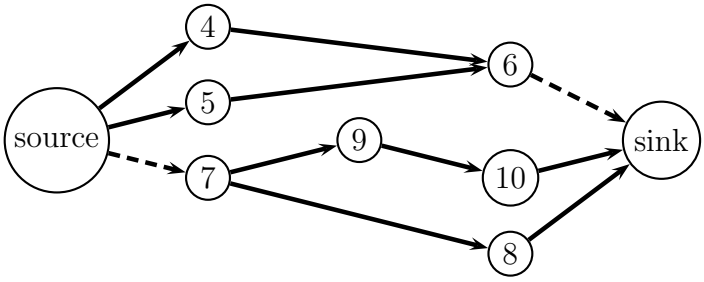


Figure 3.16: The min flow cost solution to the network in Figure 3.15.

are consecutive in both the ordered sequence and the permuted sequence and combine them into one vertex with a weighting of n where the vertex represents n elements. For example, the elements $4, 5$ are consecutive in both the ordered sequence and the permutation represented by the permutation induced digraph in Figure 3.14. We could combine them into one vertex in the network and give that vertex a weight of 2. This would not change the optimal solution, but would make the network smaller.

Now that we can find a maximum-sized set of elements in a permuted ordering that are a subsequence of the ordering, we want to apply this to finding the minimum number of insertions required to reduce a set of cherries in a phylogenetic X-tree with a duplication reduction. This is done in Lemma 3.11 which relies on Lemma 3.10.

Lemma 3.10. *Suppose we have the sequence of cherries shown in Figure 3.17, where* $\pi(2, n-1)$ *is a permutation of the sequence* $(r_2, \ldots, r_{n-1})$*. Then the minimum number of insertions required to reduce the cherries* $c_1 = (l_1, r_1)$ *and* $c_n = (l_n, r_n)$ *is twice the number of vertices in a minimum vertex covering of* $D(\pi(2, n-1))$*.*

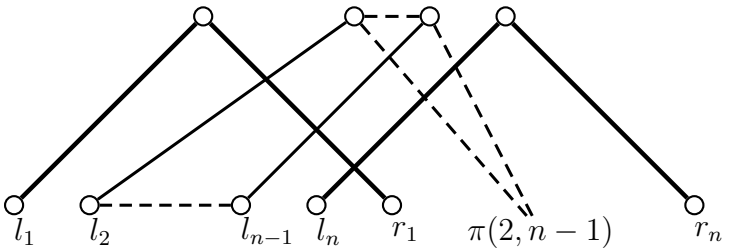


Figure 3.17: A set of overlapping cherries.

Proof. First note that any cherry $c_i = (l_i, u_i, r_i)$ with $i \in \{2, \ldots, n-1\}$ has its left leaf l_i between l_1 and l_n and its right leaf r_i between r_1 and r_n in the ordering O. Therefore by Lemma 3.3, for each cherry c_i, $i \in \{2, \ldots, n-1\}$, if we restrict the sequence to c_1, c_i and c_n then, we can reduce it with a duplication reduction. Furthermore, if we restrict the sequence to c_1, c_n and a set of cherries such that their right leaves are a subsequence of the ordering $(2, \ldots, n-1)$ then we can reduce this set of cherries with a single duplication reduction. Using this fact, it is clear that the maximum number of cherries that can be reduced along with c_1 and c_n by a single duplication reduction is the same as the maximal set of elements of $\pi(2, n-1)$ that are a subsequence of $(2, \ldots, n-1)$. The complement of the maximum number of elements in $\pi(2, n-1)$ that are a subsequence of $(2, \ldots, n-1)$ is the minimum number of elements in $\pi(2, n-1)$ such that removing them will produce a permutation that is a subsequence of $(2, \ldots, n-1)$. By Lemma 3.9, this corresponds to a minimum vertex covering on $D(\pi(2, n-1))$.

Take any two elements of a permutation that are not in a maximum subsequence of the permutation that is also a subsequence of the original ordering, deleting one of these will not put the other one in a maximum subsequence of the original ordering. Applying this to two cherries that overlap c_1 and c_n but whose right leaves are not in the maximum subsequence of $\pi(2, n-1)$ that is also a subsequence of $(2, \ldots, n-1)$ yields a similar result. Adding insertions to one of the cherries will not alter the other cherry so that it is now in a maximum set of cherries that can be reduced with c_1 and c_n by a single duplication reduction. Because of this we can consider cherries separately.

The removal of a cherry from this sequence is equivalent to two insertions, one on each

edge of the cherry. This is because both leaves from a cherry removed from the sequence are between l_1 and r_n. If the left or right leaf of a cherry has an insertion, then the cherry will no longer be a cherry after the insertions and so the other leaf will also need an insertion. Otherwise, we will not be able to reduce c_1 and c_n. Therefore the minimum number of insertions required to reduce the sequence of cherries with a single duplication reduction is the same as twice the number of elements whose removal from $\pi(2, n-1)$ gives a maximal subsequence of $(2, \ldots, n-1)$. This is twice the number of vertices in a minimum vertex cover of the graph $D(\pi(2, n-1))$.

□

Lemma 3.11. *Suppose we have two cherries c_1 and c_n in the duplication bipartite graph shown in Figure 3.18. Take a maximum set of elements from a permutation made from*

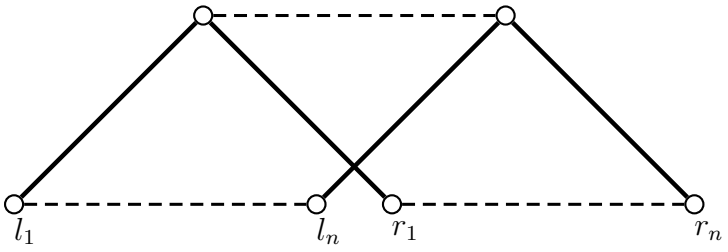


Figure 3.18: A set of cherries.

the right leaves of the cherries that correctly overlap both c_1 and c_n and form a subsequence of $(2, \ldots, n-1)$. Let P be the number of cherries that correctly overlap both c_1 and c_n who do not have elements in this maximum set. If we want to add insertions so that c_1 and c_n can be reduced, then we need to add at least $L + C + 2P$ insertions, where

1. *L is the number of leaves between l_1 and r_n that are not part of cherries,*
2. *C is the number of leaves between l_1 and r_n that are in cherries that do not correctly overlap both c_1 and c_n.*

Proof. Any leaf that is not part of a cherry in a duplication bipartite graph is not part of a cherry in any corresponding duplication tree. Therefore, any leaf between l_1 and r_n that

is not part of a cherry will require an insertion to allow c_1 and c_n to be reduced together. The same applies to any leaves that are part of cherries that do not correctly overlap.

The only leaves that are left are part of cherries that correctly overlap c_1 and c_n. From Lemma 3.10, we know that we will need a minimum of $2P$ insertions to reduce c_1 and c_n. The cherries whose leaves are part of the maximum set of elements from the permutation that are also a subsequence of $(2, \ldots, n-1)$will not require any insertions to allow c_1 and c_n to be reduced together. Therefore the minimum number of insertions required to reduce c_1 and c_n together is $L + C + 2P$. □

3.2.2 The LocallyInsert Algorithm

We now have a way of calculating the minimum number of insertions required to reduce a set of cherries using a duplication reduction. We can use these results to construct a locally optimal algorithm to solve Minimum Insertions and Duplication Reductions. Recall that it is conjectured that this problem is NP-hard, in which case, it is probable that we can not construct a polynomial-time optimal algorithm. The algorithm presented works by restricting the tree $\mathcal{T}$ to the partial duplication bipartite graph G_0. It then adds insertions and reduces the cherries that optimize the ratio of the number of insertions to the number of cherries that are eliminated. Lemma 3.11 is the basis for this calculation. We will define $width(c_i)$ to be the width of the cherry $c_i = (l_i, r_i)$.

Algorithm: LocallyInsert

Input: A partial duplication bipartite graph G with no visible duplication events and an ordering O on the leaves (vertices in V) of G.

Output: A set of cherries that maximizes the ratio of cherries eliminated to the number of insertions required to eliminate these cherries. There may be many such sets, in which case, the algorithm outputs the first such set it comes across.

1. Label all vertices that are left leaves of a cherry with $l_1, l_2, l_3, \ldots$ so that for any two leaves labeled l_i and l_j, $l_i <_O l_j$ if and only if $i < j$. Next label all vertices that are

right leaves of a cherry with r_i, where r_i is in a cherry with left leaf l_i. Set $i = 1$, Max to 0 and $Cherries$ to $\emptyset$.

2. If $1/width(c_i) > Max$, then replace Max with $1/width(c_i)$ and replace $Cherries$ with $\{c_i\}$. Otherwise, do not change Max or $Cherries$.

3. Let A be the set of vertices between l_i and r_i that are labeled l_j such that r_j is after r_i in O. For each r_j that forms a cherry with an $l_j \in A$, let B be the set of all vertices between l_i and r_j that are not part of cherries that can be reduced with the cherries c_i and c_j.

Remove these from the sequence and let C be the minimum number of insertions needed to reduce this set of cherries (Lemma 3.10) and K be the set of cherries that can be reduced in this solution. If $K/(B + C) > Max$, then set Max to $K/(B + C)$ and $Cherries$ to $\{K\}$. Otherwise, do not change Max or $Cherries$.

4. Increment i by 1 and go back to step 2. Repeat until this has been done for all cherries in G.

5. Output $Cherries$ and Max.

We will prove that LocallyInsert gives the set of cherries with the optimal ratio of insertions to cherries eliminated at the end of this section. First we will present an example of how LocallyInsert can be used. As LocallyInsert produces the set of cherries whose ratio of cherries reduced to insertions required is maximum, we can repeatedly use it to reduce an ordered phylogeny to its root.

Presented below is an example of using LocallyInsert to reduce the tree $\mathcal{T}$ shown in Figure 3.19 to its root. Note that if two cherries do not overlap, then they can not be reduced together without adding some insertions to at least one of them. If we add insertions to one of them, then clearly we can not reduce both with a single duplication reduction. Therefore, we only need to consider sets of cherries that overlap each other. The first thing we do is construct the partial duplication bipartite graph G_0. This gives the bipartite graph shown in Figure 3.20. Starting from the left, we subsequently check

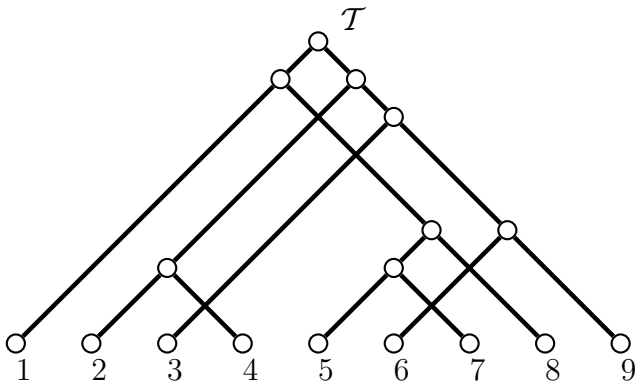


Figure 3.19: A duplication tree.

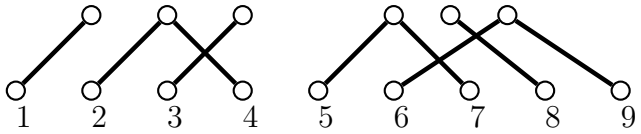


Figure 3.20: The partial duplication bipartite graph G_0 for the tree in Figure 3.19.

each cherry. We will refer to the number of cherries eliminated with a set of cherries divided by the number of insertions required as the *score* of the set of cherries. The first cherry we come across is the cherry $c_1 = (2, 4)$. First we count the leaves between 2 and 4. In this case there is only one leaf. This gives the cherry a score of $1/1 = 1$. That is, with one insertion, we can eliminate one cherry from the tree. This completes Step 2 for $i = 1$. In Step 3 we check all cherries that overlap the cherry $(2, 4)$ from the right. There are none, so we move on to the next cherry. We now increment i by one and return to step 2.

The next cherry is $c_2 = (5, 7)$. This cherry also has only one leaf between its leaves and thus has a score of 1. Now for step 3 with $i = 2$, we check all cherries that left overlap the cherry $(5, 7)$. This cherry is also overlapped by the cherry $(6, 9)$ on its right. We now take the leaves between 5 and 9 and count and remove the leaves that are between them and not part of a cherry or leaves from a cherry that only has one of its leaves between 5 and 9. There is only one, the leaf 8. We then remove all cherries that have

both their leaves between 5 and 9 that do not properly overlap both cherries. There are none for these two cherries. Finally, we take all the cherries that properly overlap both cherries and form a permutation out of their right leaves. We form the permutation by putting the place in the ordering of the left leaves of the cherries in the order of their corresponding right leaves. Once again, for this set of cherries there are no such cherries. We then add up all the scores (the number of leaves not part of cherries plus the number of leaves that are part of cherries that do not properly overlap both cherries plus twice the minimum number of elements whose removal gives a subsequence of O). This is a total of 1. We then divide the number of cherries eliminated by this (the two cherries plus the maximum number of elements from the permutation that are a subsequence of O) by the insertions required. This gives the two cherries $(5,7),(6,9)$ a score of 2. The last cherry to check is the cherry $(6,9)$. This has two leaves between its leaves and so has a score of 0.5. We choose the set of cherries with the highest score, in this case the cherries $(5,7)(6,9)$. We have now run out of cherries, so we choose the set with the highest score and add the corresponding insertion and perform all visible duplication reductions. This can be seen in Figure 3.21. We now repeat. The next partial duplication bipartite graph

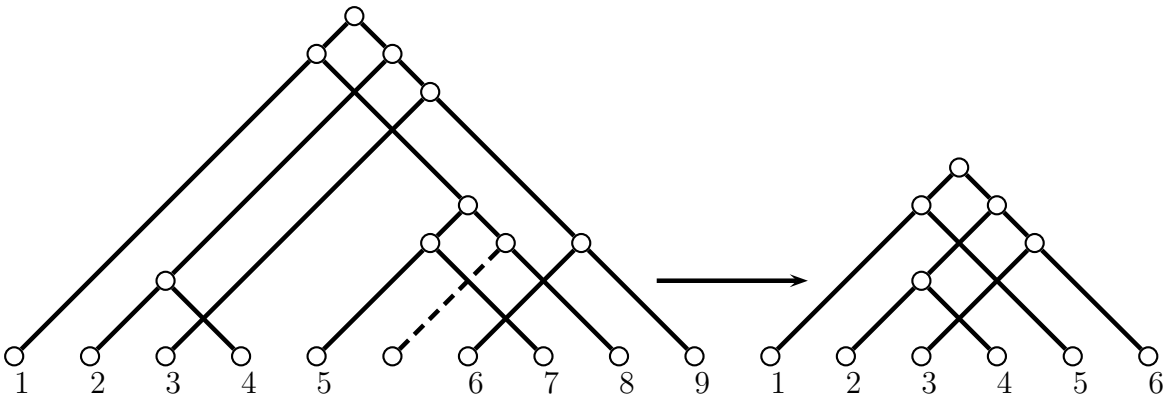


Figure 3.21: The set of insertions and the two duplication reductions initially used on the tree in Figure 3.19 by LocallyInsert.

is shown in Figure 3.22. When we calculate the scores again, we see that the cherry $(2,4)$ has score 1, the cherry $(3,6)$ has score 0.5 and the cherries $(2,4)(3,6)$ have score 2. The

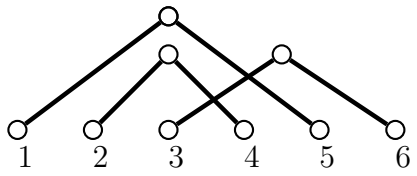


Figure 3.22: The partial duplication bipartite graph G_0 for the tree created in Figure 3.21.

cherries $(2,4)(3,6)$ have the highest score, so we add the required insertion to get the tree in Figure 3.23. This tree is now a duplication tree and can be completely reduced with

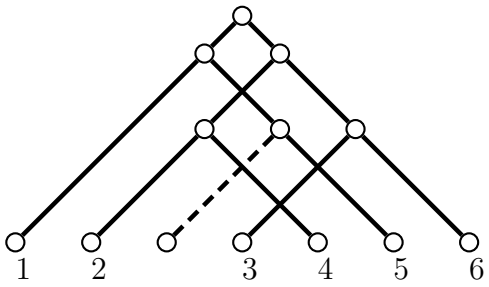


Figure 3.23: The insertion that LocallyInsert uses on the tree created in Figure 3.21.

no more insertions. This gives a total of two insertions required to reduce the tree $\mathcal{T}$. In this case this is an optimal sequence of insertions and deletions. This can be easily checked by adding a single insertion to any one of the 16 edges of the ordered phylogeny in Figure 3.19. None of the possible insertions allow us to completely reduce this tree. This will not always be the case with this algorithm. Note that in this example every insertion could only be inserted in one place in the ordering. This will not always be the case, some insertions might be able to be a left or a right leaf of the cherry that is formed by inserting them. There is no way of knowing which is better until you have done further reductions. You may even need to completely reduce the tree before you know which is better. Because of this, better results may be achieved by constructing some sort of two-step algorithm where we perform the two sets of insertions and duplication reductions whose total score is the greatest.

To conclude this section, we prove that LocallyInsert is locally optimal.

Theorem 3.12. *The set Cherries that is produced by* LocallyInsert *has the optimal ratio of cherries eliminated to insertions required.*

Proof. The algorithm works by starting with the left most cherry and tests all cherries that can be reduced with this cherry. From Lemma 3.10 we know that the ratio for pair of cherries is optimal. Because of this, if the algorithm checks all possible pairs of cherries, then the pair of cherries it outputs will give the optimal ratio. For the two cherries to be reduced together, they must overlap appropriately. That is for the cherries c_k and c_l to be reduced, the sequence l_k, l_l, r_k, r_l must be subsequence of O. If this is the case, then when $i = k$ this set of cherries will be considered. As mentioned before, if two cherries, c_1 and c_2, do not overlap, then they can not be reduced together in a single duplication reduction. The cherry c_1 would need at least one insertion in order to be involved in the duplication reduction that reduced c_2 and would therefore not be reduced in this reduction. So we only need to consider cherries that overlap each other.

Therefore every possible set of cherries is considered and the set that is chosen has the optimal ratio.

□

3.3 Java Implementation

The algorithm LocallyInsert has been implemented in Java. The results of which are presented in this section. Subsection 3.3.1 describes the biological model we use to generate the trees and their deletions and Subsection 3.3.2 discusses the results obtained by running LocallyInsert on the trees generated.

3.3.1 Biological Model

The ordered phylogenies that we are testing LocallyInsert on will all have 20 leaves. This is roughly the upper limit in the size of duplication trees seen in the literature. We will be running the algorithm on trees that have had between one and five deletions. To

do this, we will generate duplication trees with 21 to 25 leaves and then perform deletions on these trees.

It is believed that single duplication events are predominate over multiple events [30]. Because of this, the method used to construct the duplication trees favours single duplication events. To create a random duplication tree, simply start with a single cherry and perform random duplication events until you have obtained a tree with the required number of leaves. The following method was used to create the random duplication events. First we choose a random length for the event between one and the current length of the floor. We choose an event with length n with probability that decays proportional to $1/(n^{3/2})$. That is, an event with a length of one is $2^{3/2}$ more likely than an event of length of two, which is $3^{3/2}$ times as likely as an event of length three and so on.

Selecting the events uniformly at random (UAF) gives a disproportionate picture of what is believed to be reflected in nature (single or small events being the most common). For example, if the floor has nine leaves in it, then an event of length four to nine is roughly as likely as an event of length one to three if chosen UAF. This is the reason for the decaying probability.

Once we have constructed the trees, we delete the appropriate amount of leaves. In this model, we only consider deleting leaves because if we consider deleting any edge, it is possible to end up with a duplication tree on a very small number of leaves. If we only delete leaves, then we can be assured to always have the number of leaves required. While only deleting leaves may not be entirely realistic, it is a good starting point and can provide a good estimation of how the LocallyInsert performs.

3.3.2 Results

As stated above, the Java implementation of LocallyInsert was run on ordered phylogenies with twenty leaves with deletions varying from one to five. A random ordered phylogeny was constructed as described in Section 3.3.1 and the number of insertions required to reduce it to its root using LocallyOptimal was recorded. This was done $10,000$ times for each number of deletions. While $10,000$ may seem unnecessarily large,

note that the number of binary phylogenetic X-trees on twenty leaves is 2.22×10^{20}. While not all of these trees are possible from the construction in Section 3.3.1, a large enough number of them are. The results are displayed in the table below. The rows in the table correspond to the number of insertions needed for a tree with $1-5$ deletions and the columns correspond to the number of times $0-10+$ insertions were used by LocallyInsert. For example, the value 2083 in the second row and second column shows that out of the $10,000$ ordered phylogenies who had five deletions, 2083 of them needed zero insertions to be reduced to its root by LocallyInsert. The value 2026 in the third column and second row shows that 2026 ordered phylogenies with five deletions needed one insertion to be reduced by LocallyInsert.

No insertions	0	1	2	3	4	5	6	7	8	9	10	>10
5 deletions	2083	2026	1694	1262	907	605	443	307	194	158	105	216
4 deletions	2642	2254	1669	1132	778	505	343	232	158	100	66	121
3 deletions	3275	2584	1563	921	626	390	243	141	90	67	40	60
2 deletions	4573	2551	1261	658	376	259	136	68	53	21	21	23
1 deletion	6498	2183	626	332	181	100	39	20	10	7	2	2

Clearly LocallyInsert is not perfect. Using over ten insertions to reconstruct a duplication history that only had one deletion is not optimal. However, this is a rare occurrence. Roughly $60-80\%$ of the time when the current tree is not a duplication tree, the number of insertions used in LocallyInsert is equal to or less then the actual number of deletions that occurred. A number of steps can be taken to improve the performance of LocallyInsert. Often, there may be two or more sets of cherries with the same ratio of insertions and cherries reduced. If we try all optimal possibilities when this occurs, then this may greatly improve LocallyInsert. Alternatively, as stated in Section 3.2.2, if we take the two sets of cherries who together have the optimal ratio (by taking one set, reducing it and then taking the next set), then we could improve the results of LocallyInsert.

Note the number of trees that are still duplication trees is very large ($20 - 65\%$ depending on the number of deletions). This is bad news if duplication trees do evolve similarly to the trees generated in Section 3.3.1. If this is the case, then reconstructing the duplication history for a given phylogeny may be impossible as a large number of deletions could have occurred with the resulting tree still being a duplication tree. It may be possible that almost all genes with consistent duplication histories may have had deletions somewhere in their history. As a result, we may never be able to completely understand their duplication history.

Chapter 4

Supertree Methods for Duplication Trees

Supertree methods are being used increasingly frequently, particularly in evolutionary biology. These methods enable the user to construct a parent tree from a forest of smaller phylogenetic X-trees whose label sets have elements in common. This parent tree often displays all information present in each individual tree in the forest, plus some additional information that can not be gained by looking at the individual trees by themselves. There are two types of supertree algorithms. The first type determines whether a supertree exists for a given forest of trees and produces one if it exists. If there is no supertree, then the algorithm will produce no tree. The second type of algorithm will always output a tree. The tree may not be a supertree for the entire forest of trees, but will be as close as possible. Currently, there are many efficient algorithms that construct supertrees under varying conditions, for example [1, 6, 26]. For an overview of the area, see [5]. However, none of the algorithms at present can be directly used on duplication trees; they will not always produce a supertree that is also a duplication tree when applied to a forest of trees with the same ordering O. In this section we extend these results to duplication trees. Given a forest of trees, there may be an exponential number of different phylogenetic X-trees that displays the forest of trees. This can happen when there is little overlap between the label sets of the trees in the forest. This will give a great deal of choice when we create a supertree for the forest and lead to an exponential number of possible supertrees that display the forest. There are algorithms that generate all trees that display a forest [6, 8, 23]. Even with these algorithms, searching through all possible trees that display the forest to find a duplication tree is a very unattractive prospect.

Recall that a rooted binary phylogenetic X-tree $\mathcal{T}$ displays another rooted binary phylogenetic X-tree $\mathcal{T}'$ if $X' \subseteq X$ and $\mathcal{T}|X \cong \mathcal{T}'$, where $\mathcal{T}|X'$ is $\mathcal{T}$ restricted to the leaves of $\mathcal{T}'$. Suppose we have a forest $\mathcal{F}$. A supertree for this forest is a tree $\mathcal{T}_{\mathcal{F}}$ such

that for all $t \in \mathcal{F}$, t is displayed in $\mathcal{T}_{\mathcal{F}}$. For convenience we will call a supertree that is also a duplication tree a super duplication tree.

Consider the following example. Suppose we have the forest $\mathcal{F}$ shown in Figure 4.1. There are three trees that display the forest $\{t_1, t_3\}$. These are shown in Figure 4.2. We

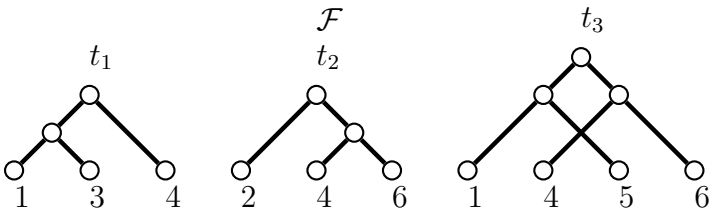


Figure 4.1: A forest.

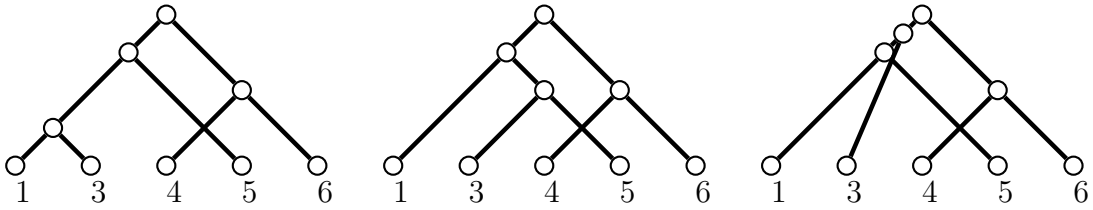


Figure 4.2: Supertrees for the trees t_1 and t_3 from Figure 4.1.

could add a leaf labeled 2 to any of the above trees in any position as long as it does not form a cherry with 4 or 6. As each tree has 6 edges not incident with 4 or 6, there are a total of 18 binary phylogenetic X-trees with label set $\{1, 2, 3, 4, 5, 6\}$ that displays the forest $\mathcal{F}$. This is a lot considering the label set has only six elements. We can not guarantee that the supertree method we use will output a super duplication tree. If this does not happen, then there is no way of knowing whether or not there is a super duplication tree for the forest $\mathcal{F}$. Clearly if we are studying a forest of duplication trees and are interested in the duplication history of the entire forest, a supertree that is not duplication tree is of little use.

Note that the input trees do not need to be duplication trees. A forest $\mathcal{F}$ of binary phylogenetic X-trees containing one or more trees that are not duplication trees may

still have a super duplication tree. For example in the forest below (Figure 4.3 with the ordering $O = (1, 2, 3, 4)$, t_1 is not a duplication tree. However, the forest is displayed in

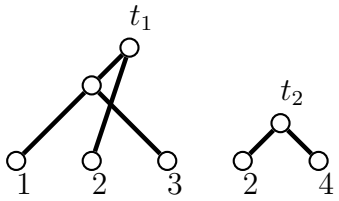


Figure 4.3: A forest containing a tree that is not a duplication tree.

the duplication tree $\mathcal{T}_\mathcal{F}$ in Figure 4.4. In this chapter we describe a polynomial-time al-

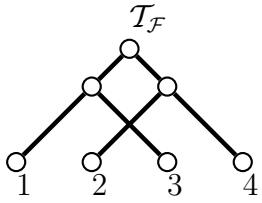


Figure 4.4: A super duplication tree for the forest in Figure 4.3.

gorithm that when given forest of binary phylogenetic X-trees it attempts to find a super duplication tree, then constructs such a tree if one is found. If one does not exist, then no tree is produced. This algorithm does not always produce a super duplication tree when one exists. It is provided here as a starting point for research into super duplication trees. This chapter is broken up into two sections. Section 4.1 presents the CherryFind algorithm. The CherryFind algorithm plays an important part in the SuperDuplicationTrees algorithm, which is presented in section 4.2.

4.1 The CherryFind Algorithm

We begin with Proposition 4.1 which gives a sufficient condition for the cases where a super duplication tree does not exist. It will be used in a polynomial-time algorithm presented at the end of this section. The proof that Proposition 4.1 can be checked in polynomial-time will follow as a direct consequence from Lemmas 4.2 and 4.3.

Proposition 4.1. *Let $\mathcal{F}$ be a forest of ordered phylogenies with an ordering O on the leaves of the forest. Suppose that there is no possible sequence of cherries*

$$\mathcal{S} = (l_i, u_i, r_i), (l_{i+1}, u_{i+1}, r_{i+1}), \ldots, (l_k, u_k, r_k),$$

such that:

1. *$l_i, l_{i+1}, \ldots, l_k, r_i, r_{i+1}, \ldots, r_k \subseteq O$ and*

2. *no cherry in $\mathcal{S}$ is displayed in any tree of $\mathcal{F}$ unless it is a cherry of that tree.*

Then there does not exist a super duplication tree that displays $\mathcal{F}$.

Proof. Let $\mathcal{F}$ be a forest with no sequence of cherries that satisfies the above two conditions. Assume that there exists a super duplication tree $\mathcal{T}_{\mathcal{F}}$ that displays the forest $\mathcal{F}$. Take any visible duplication event from the super duplication tree $\mathcal{T}_{\mathcal{F}}$. This can be represented by the sequence of cherries

$$\mathcal{S} = (l_i, u_i, r_i), (l_{i+1}, u_{i+1}, r_{i+1}), \ldots, (l_k, u_k, r_k).$$

We know that this sequence of cherries does not satisfy both of the above conditions. The first condition will hold for all sequences representing a duplication event. The sequence represents a duplication event and therefore satisfies condition one. This implies that condition two does not hold. Therefore there is at least one cherry $c_j \in \mathcal{S}$ that is displayed in a tree $t \in \mathcal{F}$ but is not a cherry of t. If this is the case, then $\mathcal{T}_{\mathcal{F}}$ will not display the tree t.
This is a contradiction, as we assumed that $\mathcal{T}_{\mathcal{F}}$ displayed the forest $\mathcal{F}$. Therefore if a forest $\mathcal{F}$ has no sequence of cherries that satisfy the above two conditions, then there is no super duplication tree $\mathcal{T}_{\mathcal{F}}$ that displays the forest $\mathcal{F}$.

□

Note that to satisfy Proposition 4.1 we do not actually require that one of the cherries in the sequence $\mathcal{S}$ is a cherry in one of the trees in the forest $\mathcal{F}$. If a forest has no super duplication tree, then either the forest has no sequence of cherries that satisfies

Proposition 4.1, or after every possible set of duplication reductions that can be done on the forest by reducing a set of cherries that satisfies Proposition 4.1 we end up at a subforest that has no sequence of cherries that satisfies Proposition 4.1. Now we can test whether a given forest of duplication trees has a super duplication tree. However, before we do this, we need to know how many possible duplication events we need to test. If there is an exponential number of them, then Proposition 4.1 is of no use. This is done in Lemma 4.2

For any real number let $\lceil n \rceil$ denote the smallest integer z such that $z \geq n$ and let $\lfloor n \rfloor$ denote the largest integer z such that $z \leq n$.

Lemma 4.2. *The set of all possible visible duplication events a duplication tree with $n \geq 4$ leaves can have has cardinality*

$$\sum_{w=1}^{\lceil n/2 \rceil} (2w - 1).$$

Proof. Take a duplication tree with $n \geq 4$ leaves. A duplication tree with n leaves cannot have a visible duplication event involving more than $\lfloor n/2 \rfloor$ cherries. There will be $n - 2w + 1$ possible duplication events involving w cherries. These will have the cherries (l_i, r_i) shown below.

$$\big((1, w+1), (2, w+2), \ldots (w, 2w)\big), \big((2, w+2), (3, w+3), \ldots (w+1, 2w+1)\big), \ldots$$
$$\ldots, \big((n-2w, n-w+1), (n-2w+1, n-w+1), \ldots (n-w, n)\big)$$

This gives

$$\sum_{w=1}^{\lfloor n/2 \rfloor} (n - 2w + 1) = \sum_{w=1}^{\lceil n/2 \rceil} (2w - 1)$$

possible duplication events.

□

Now that we know that there is only a polynomial number of duplication events to check, we can test whether a forest of binary phylogenetic X-trees has a super duplication tree. Note that even though there is only a polynomial number of checks required to find

out if a forest has a sequence of cherries that satisfies the conditions of Proposition 4.1, it does not mean that we can determine whether a forest has a super duplication tree in polynomial time. At each step we only need to do a polynomial number of checks, but we may need a non-polynomial number of steps. This will be explained more in the next section. Despite this, CherryFind can sometimes inform us that there does not exist a super duplication tree for a given forest. For example, consider the three duplication trees in Figure 4.5 with ordering $O = (1, 2, 3, 4, 5, 6)$. All three of these are duplication

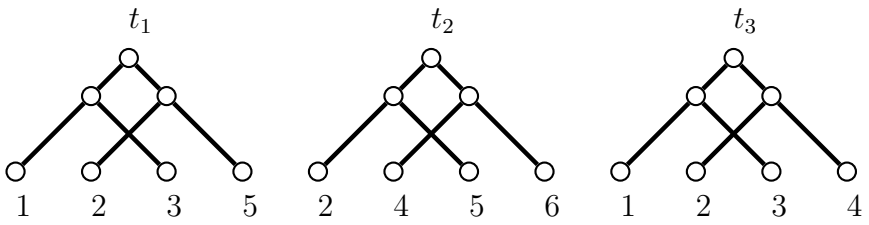


Figure 4.5: A forest of three duplication trees.

trees. However, the only supertree $\mathcal{T}$ that displays this forest (shown in Figure 4.6) is not a duplication tree. Using Proposition 4.1, we can see why there is no super duplication

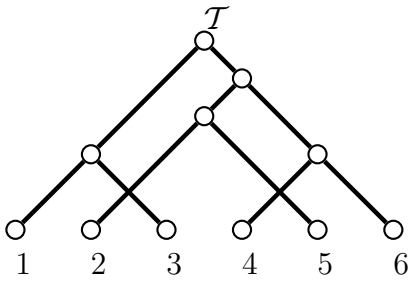


Figure 4.6: The only super duplication tree for the forest in Figure 4.5.

tree for our forest of three duplication trees. There are $\sum_{w=1}^{\lceil n/2 \rceil}(2w-1)$ possible visible duplication events in a duplication tree with $n \geq 4$ leaves. Our tree has six leaves and therefore there are nine possible visible duplication events in any super duplication tree, if one exists. These possible events reduce the cherries

$$(1,2), (2,3), (3,4), (4,5), (5,6), ((1,3)(2,4)), ((2,4)(3,5)), ((3,5)(4,6)), ((1,4)(2,5)(3,6)).$$

Checking the simple reductions, we see that $(1,2)$ and $(2,3)$ are displayed in t_1, but not as cherries, $(3,4)$ is displayed in t_3 not as a cherry and $(4,5)$ and $(5,6)$ are displayed in t_2 but not as cherries. The cherry $(2,4)$ is displayed in t_2 but not as a cherry, so this excludes the events that reduce the cherries $((1,3)(2,4))$ and $((2,4)(3,5))$. The cherry $(3,5)$ is displayed in t_1 not as a cherry and so this excludes the reduction that reduces the cherries $((3,5)(4,6))$. Finally, because the cherry $(1,4)$ is displayed in t_3 not as a cherry, the duplication event that reduces the cherries $((1,4)(2,5)(3,6))$ is not possible.
Therefore there is no sequence of cherries

$$\mathcal{S} = (l_i, u_i, r_i), (l_{i+1}, u_{i+1}, r_{i+1}), \ldots, (l_k, u_k, r_k),$$

such that:

1. $l_i, l_{i+1}, \ldots, l_k, r_i, r_{i+1}, \ldots, r_k \in O$ and
2. no cherry in $\mathcal{S}$ is displayed in any tree of $\mathcal{F}$ unless it is a cherry of that tree.

Therefore by Proposition 4.1, there does not exist a duplication tree that displays the three trees in our forest.

Because Proposition 4.1 characterizes the forests that have no super duplication tree, we would like a way to check for a sequence of cherries that satisfies conditions one and two of Proposition 4.1. The algorithm CHERRY FIND (presented below) outlines a method for finding a suitable sequence of cherries if one exists.

Algorithm: CHERRYFIND
Input: A forest of binary phylogenetic X-trees $\mathcal{F}$ with an ordering O on the leaves of $\mathcal{F}$
Output: A sequence of cherries

$$\mathcal{S} = (l_i, u_i, r_i), (l_{i+1}, u_{i+1}, r_{i+1}), \ldots, (l_k, u_k, r_k),$$

such that $l_i, l_{i+1}, \ldots, l_k, r_i, r_{i+1}, \ldots, r_k \in O$ and every cherry in $\mathcal{S}$ is a cherry in every tree of $\mathcal{F}$ that it is displayed in, or NO if no such sequence exists.

1. List all possible duplication events starting with the biggest possible for a tree with ordering O.

2. Successively run through each duplication event and check the trees in the forest $\mathcal{F}$. If you find a duplication event such that every cherry in it is a cherry in every tree in $\mathcal{F}$ that displays it, then stop and output the sequence of cherries.

3. If all duplication events have been checked, then output NO.

To calculate the complexity of CherryFind, we need to know how many duplication events we need to check and how many possible distinct cherries there are that need to be checked in order to check every possible duplication event. The number of events was covered in Lemma 4.2 and the number of distinct cherries will be covered in Lemma 4.3.

Lemma 4.3. *If a duplication tree $\mathcal{T}$ has n leaves, then there are*

$$\sum_{w=\lceil n/2 \rceil}^{n-1} (w)$$

distinct cherries that can be in a visible duplication event of $\mathcal{T}$.

Proof. Suppose we have a duplication tree with n leaves. As stated in Lemma 4.2, there can be no duplication event with more than $\lfloor n/2 \rfloor$ cherries in it and there will be $n-2w+1$ possible duplication events that involve w cherries. List all the possible duplication events involving w cherries in order corresponding to the ordering of the duplication tree O. The first duplication event will have w distinct cherries. The next duplication event will only have one cherry that has not appeared in the first duplication event. Each event after this will only have one new cherry. If there are m duplication events with w cherries, then we will have $m + w - 1$ distinct cherries. Since $m = n - 2w + 1$, this will give $n - 2w + 1 + w - 1 = n - w$ distinct cherries in a duplication tree with n leaves that are part of a duplication event involving w cherries. All these cherries will be distinct because they are distinct in the set of cherries with width $w - 1$ and all other sets of cherries will have a different width and therefore not include these cherries. The number of cherries involved in an event can range from 1 to $\lfloor n/2 \rfloor$. Therefore there will be

$$\sum_{w=1}^{\lfloor n/2 \rfloor} (n - w) = \sum_{w=\lceil n/2 \rceil}^{n-1} (w)$$

possible distinct cherries in a visible duplication event in $\mathcal{T}$.

□

The time complexity of CherryFind is in terms of $q = n \cdot r$ where $n = |O|$ and r is the number of trees in the forest. Clearly there can be no more unique duplication events than there are distinct cherries. Therefore there are

$$\sum_{w=\lceil n/2 \rceil}^{n-1} (w) < \sum_{w=1}^{n-1} (w) = (n^2 - n)/2$$

unique distinct to check. In the worst case, we will have to check every cherry in every tree in $\mathcal{F}$. This will require $O(n^2 \cdot r)$ operations. Hence Cherry Find has quadratic running time in terms of q.

4.2 The SuperDuplicationTrees Algorithm

Now that we have a way of establishing whether a forest has a super duplication tree, we can move on to constructing a super duplication tree. This can be done using the Super Duplication Tree algorithm which makes use of the CherryFind algorithm from the last section. It is based on Lemma 4.4.

Lemma 4.4. *Let $\mathcal{F}$ be a forest of binary phylogenetic X-trees and $\mathcal{T}$ a super duplication tree for $\mathcal{F}$. Let $\mathcal{T}'$ be the tree obtained by performing a single duplication reduction. Take the forest $\mathcal{F}$. For every tree in $\mathcal{F}$ that displays a cherry that was reduced in the duplication reduction on $\mathcal{T}$, replace it with its root. For every leaf that is from a cherry that was reduced in the duplication reduction, replace it with the root of the cherry. Call the forest created in this way $\mathcal{F}'$. Then $\mathcal{T}'$ is a super duplication tree for $\mathcal{F}'$.*

Proof. Assume that $\mathcal{T}'$ is not a super duplication tree for $\mathcal{F}'$. Because $\mathcal{T}$ is a duplication tree, $\mathcal{T}'$ is also a duplication tree. So, if $\mathcal{T}'$ is not a super duplication tree for $\mathcal{F}'$, then $\mathcal{T}'$ must not be a supertree for $\mathcal{F}'$. If this is true, then there must be a tree $t' \in \mathcal{F}'$ that is not displayed in $\mathcal{T}'$. Let t be the tree t' before any reduction was performed. We know that t is displayed in $\mathcal{T}$. Furthermore, because t' is a restriction of t, we know that t'

is also displayed in $\mathcal{T}$. If t has a cherry that was reduced in the duplication reduction, then this cherry will be replaced with its root in both t and $\mathcal{T}$ and hence still displayed in $\mathcal{T}'$. The tree t cannot have both leaves from a cherry that was reduced unless they are a cherry in t as well (because of Proposition 4.1). Therefore if any leaf from a cherry that was reduced is replaced by its root, in t then t' is still displayed in $\mathcal{T}'$ and hence a contradiction as we assumed that t was not displayed in $\mathcal{T}'$. Therefore, $\mathcal{T}'$ displays $\mathcal{F}'$.

□

This is the basic idea for the algorithm. If there exists a super duplication tree, then if we keep reducing this tree and the forest, then the reduced tree will display the reduced forest. This is how the algorithm runs. We keep reducing the forest and if we get to a forest of isolated vertices, then we can construct a super duplication tree with the duplication events used to reduce the forest. Note that this algorithm is not optimal. It has the possibility of missing a super duplication tree and outputting NO when a super duplication tree exists. However, if it outputs a tree, then this tree is a super duplication tree for the input forest $\mathcal{F}$.

Algorithm: SUPERDUPLICATIONTREES

Input: A forest of binary phylogenetic X-trees $\mathcal{F}$ with an ordering O on the leaves of $\mathcal{F}$

Output: A duplication tree that displays all trees in the forest or NO if no such tree was found.

1. Set $X = X_0$, $O = O_0$, $j = 0$ and $\mathcal{S}_0 = \emptyset$.

2. If $\mathcal{F}$ consists of only roots, then construct any duplication tree with leaf set consisting of all the roots of $\mathcal{F}$. Now perform the duplication events corresponding to the sequences of cherries represented by the $\mathcal{S}_j$'s on this tree. Output this tree and stop.

3. Otherwise, let C be the set of all possible visible cherries that a duplication tree with $|O_j|$ leaves could have. Run CHERRYFIND. If CHERRYFIND outputs NO, output NO and stop. Otherwise let $\mathcal{S}_j$ denote the sequence of cherries that CHERRY FIND outputs.

4. For every cherry $c_n = (l_n, u_n, r_n) \in \mathcal{S}_j$, remove r_n from X_j and O_j. Call this new set X_{j+1} and the new ordering O_{j+1}. Everywhere c_n is a visible cherry in $\mathcal{F}$, replace it

with a single vertex labeled l_n. Everywhere there is a leaf in a tree of $\mathcal{F}$ labeled r_n, relabel l_n. Increment j by 1. and return to step 2.

Given an input of r trees and an ordering of size n, the algorithm SuperDuplicationTrees calls CherryFind at most $n-1$ times and reduces at most $nr/2$ cherries each time CherryFind is called. The time complexity of CherryFind was shown to be $O(n^2 \cdot r)$ in Section 4.1. This gives a complexity of $(n-1)(O(n^2 \cdot r) + n \cdot r/2)$ for SuperDuplicationTrees. This gives it a running time $O(n^3 \cdot r)$.

Before we give a proof that a super duplication tree exists for a forest if one is produced by SuperDuplicationTrees, we will run through an example of the SuperDuplicationTrees algorithm.

Suppose we have the forest $\mathcal{F}$ in Figure 4.7 with the ordering $O = (1, 2, 3, 4, 5, 6)$. When we run CherryFind on this forest, it will output the sequence of cherries $(3, 5)(4, 6)$.

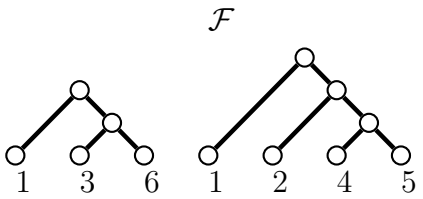


Figure 4.7: A forest.

So if we reduce all cherries $(3, 5)$ and $(4, 6)$, replacing all other leaves labeled 5 with 3 and leaves labeled 6 with 4 then we get the following updated forest shown in Figure 4.8. The ordering is now $O_1 = (1, 2, 3, 4)$. When we run CherryFind again it will output the cherry $(3, 4)$. Performing the duplication reduction on the forest will change the forest to the one in Figure 4.9. A further calling of CherryFind will output the cherry $(2, 3)$. A final run of CherryFind will output the cherry $(1, 2)$. If we then construct a duplication tree from the duplication events we will obtain the following tree in Figure 4.10. To conclude this chapter, we provide a proof that a forest of ordered phylogenies has a super duplication tree if SuperDuplicationTrees outputs a tree.

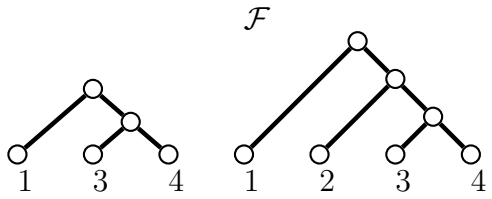


Figure 4.8: The forest from Figure 4.7 after the first reduction of SuperDuplicationTrees.

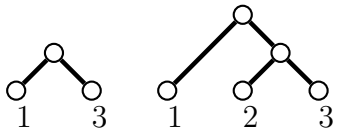


Figure 4.9: The forest from Figure 4.7 after the second reduction of SuperDuplicationTrees.

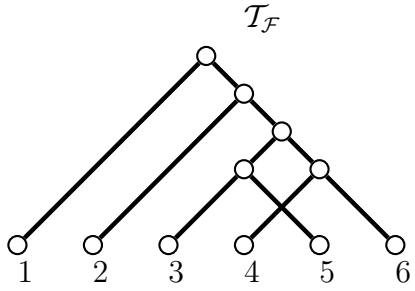


Figure 4.10: Supertree for the forest from Figure 4.7.

Theorem 4.5. *If* SuperDuplicationTrees *outputs a tree $\mathcal{T}$, then $\mathcal{T}$ is a duplication tree that displays the forest $\mathcal{F}$.*

Proof. Suppose that SuperDuplicationTrees outputs a tree $\mathcal{T}$. The tree $\mathcal{T}$ is made from a series of duplication events, so clearly it is a duplication tree.

Consider the partial duplication bipartite graph G_0 of $\mathcal{T}$. Every cherry in G_0 is either displayed in a tree of $\mathcal{F}$ or has at most one of its leaves displayed in any tree of $\mathcal{F}$. Every cherry that gets reduced/eliminated at iteration one will be displayed in G_0. The only other changes in the trees of $\mathcal{F}$ will be relabeling of leaves. The graph G_0 has a leaf with every label from X and therefore all relabeled leaves will also be displayed in G_0. This shows that every change in the trees of $\mathcal{F}$ at iteration one will be displayed in G_0.

Now suppose that every modification to the trees in $\mathcal{F}$ up to iteration i of SuperDuplicationTrees is displayed in the partial duplication graph $G_{0,i-1}$. Consider the partial duplication graph $G_{0,i}$ and the modifications made to the trees of $\mathcal{F}$ at iteration $i+1$ of SuperDuplicationTrees. Because $\mathcal{T}$ is constructed by doing the duplication events in the reverse order that the duplication reductions were done to the trees in $\mathcal{F}$, every cherry that was reduced in iteration $i+1$ will correspond to the cherries between floor $i-1$ and floor i of $\mathcal{T}$. By the definition of a partial duplication graph, these cherries will all be displayed in $G_{0,i}$, and these will be the only cherries between floors $i-1$ and i. The graph $G_{0,i}$ will have a vertex set corresponding to O_i and because $O_{i+1} \subset O_i$ will therefore display all leaves that were relabeled at iteration $i+1$.

This shows by induction that every reduction and relabeling done to the trees in $\mathcal{F}$ is displayed in $\mathcal{T}$. Once all trees in $\mathcal{F}$ have been reduced to their roots, any tree with a label set containing the labels of the roots will display $\mathcal{F}$. Therefore if SuperDuplicationTrees outputs a tree $\mathcal{T}$, then $\mathcal{T}$ is a duplication tree and displays the forest $\mathcal{F}$. □

The SuperDuplicationTrees algorithm is a start, but needs refinement. Further research into choosing suitable sequences of cherries is needed, so that we do not miss a super duplication tree if one exists. If we could characterize which sequences of cherries are definitely in a super duplication tree, then the algorithm would output a super duplication

tree if and only if one exists. If this is not possible, then adding in some sort of backtrack procedure, so that if we do not find a super duplication tree we go back a step and try again would be an improvement. Of course, unless we find some sort of criteria that tells us which cherries not to bother with to ensure that there is no super duplication tree, we may need to do a non-polynomial number of steps.

Chapter 5

SPR Moves on Simple Duplication Trees

In evolutionary biology, topological rearrangements are often used to improve a given phylogenetic X-tree under varying criteria. Some of the most common rearrangements are subtree prune and regraft (SPR), nearest neighbour interchange (NNI) and tree bisection and reconnection (TBR). All three of these rearrangements work by cutting an edge of the tree and then attaching the component that was just disconnected by the cut somewhere else on the tree. These rearrangements can be used to calculate a distance between two trees, where the distance is defined to be the minimum number of such rearrangements needed to transform one tree into the other.

If we are trying to improve a duplication tree through topological rearrangements, then a rearrangement that produces a tree that is not a duplication tree is clearly of little use. Consequently, it is of interest which topological rearrangements can be applied to duplication trees. It was shown in [4] that given two duplication trees $\mathcal{T}_1$ and $\mathcal{T}_2$ with the same ordering O, we can not always transform $\mathcal{T}_1$ into $\mathcal{T}_2$ by NNI when we are restricted by only being able to perform a rearrangement if the tree produced is also a duplication tree. However, it was also shown in [4] that under the same restriction, we can always transform $\mathcal{T}_1$ into $\mathcal{T}_2$ by a series of SPR moves. In the case of rooted duplication trees, this process is described as moving through RDT space and on rooted trees a SPR move is referred to as a rSPR (rooted SPR) move.

The rooted SPR distance can be calculated by the size of the maximum agreement forest (MAF) of the two trees, where a MAF is defined to be an agreement forest with the minimum number of components. Furthermore, the trees in a MAF for $\mathcal{T}_1$ and $\mathcal{T}_2$ describe a procedure that will transform $\mathcal{T}_1$ into $\mathcal{T}_2$ with the minimum number of operations. Because of this, finding a MAF for two phylogenetic X-trees is often used to calculate the distance between the two phylogenetic X-trees. We will define a MAF in Section 5.1. To

this end, we would like a notion of a maximum duplication agreement forest (MDAF). A MDAF will describe a method of transforming one duplication tree into another by SPR moves that preserve the duplication tree property. This will give us an accurate method for calculating a distance between two duplication trees or improving a duplication tree under some criteria. Defining a maximum duplication agreement forest will be covered in Section 5.1. This will be done in the context of rooted duplication trees, but can also be applied to unrooted duplication trees. When applied to a rooted duplication tree, we will refer to a SPR move as a DrSPR (duplication rooted subtree prune and regraft) move.

In Section 5.2 we will restrict ourselves to simple duplication trees and calculate the number of unique simple duplication trees that are one DrSPR move away from any given simple duplication tree. The set of trees that are one topological rearrangement away from a tree $\mathcal{T}$ is commonly referred to as the neighbourhood of $\mathcal{T}$. Searching the neighbourhood is used in hill climbing methods which are often used when looking for an optimal tree given some criteria.

5.1 The Maximum Duplication Agreement Forest

Having a duplication agreement forest will provide us with a way of performing rSPR moves on a duplication tree which will guarantee that we stay in RDT space. Being able to do this will allow us to use rSPR moves to improve a given duplication tree under various conditions. We will begin with a definition of an agreement forest and an example of how a MAF can be used to calculate the $rSPR$ distance between two rooted phylogenetic X-trees. After this, we will provide a definition of a duplication agreement forest.

Suppose we have two rooted phylogenetic X-trees $\mathcal{T}_1$ and $\mathcal{T}_2$ with the same label set X. An *agreement forest* for $\mathcal{T}_1$ and $\mathcal{T}_2$ is a forest of rooted trees $\mathcal{F} = \{t_\rho, t_1, \ldots, t_k\}$ with label sets $\mathcal{L}_\rho, \mathcal{L}_1, \ldots, \mathcal{L}_k \subseteq X \cup \rho$ such that the following is satisfied:

1. The label sets $\mathcal{L}_\rho, \mathcal{L}_1, \ldots, \mathcal{L}_k$ partition $X \cup \rho$ and $\rho \in \mathcal{L}_\rho$.

2. For all $i \in \{\rho, 1, 2, \ldots, k\}$, $t_i \cong \mathcal{T}_1|\mathcal{L}_i \cong \mathcal{T}_2|\mathcal{L}_i$.

3. The trees in $\{\mathcal{T}_1(\mathcal{L}_i) : i \in \{\rho, 1, \ldots, k\}\}$ and $\{\mathcal{T}_2(\mathcal{L}_i) : i \in \{\rho, 1, \ldots, k\}\}$ are vertex-disjoint rooted subtrees of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively.

In order for the SPR characterization to work, we place ρ at the end of an extra pendant edge in both trees. Before we move onto a duplication agreement forest, we will give an example of how we can use an agreement forest for two trees to transform one tree into the other with rSPR moves.

Suppose we have the two rooted phylogenetic X-trees in Figure 5.1 and we want to calculate the rSPR distance between them. To do this we need to construct a maximum

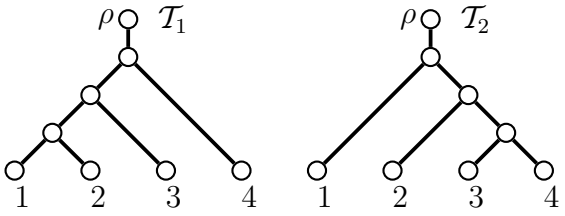


Figure 5.1: Two binary phylogenetic X trees.

agreement forest for them. Calculating the MAF for two phylogenetic X-trees is an NP-hard problem, see [7]. However, for such small trees this is no problem. A MAF $\mathcal{F}$ for the two trees is shown in Figure 5.2. It can be easily checked that this forest is optimal. If we want to transform $\mathcal{T}_2$ into $\mathcal{T}_1$, then we prune a tree in $\mathcal{F}$ from $\mathcal{T}_2$ and regraft it to

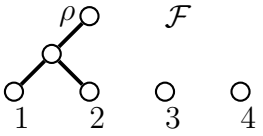


Figure 5.2: A MAF for the two trees in Figure 5.1.

the appropriate place in $\mathcal{T}_1$. We can begin by pruning and regrafting the tree with label set 4. This will give the following tree $\mathcal{T}_3$ (Figure 5.3). If we then prune the tree with label set 3 and regraft it we will obtain $\mathcal{T}_1$ in Figure 5.4 We have now transformed $\mathcal{T}_2$ into $\mathcal{T}_1$ with two operations. Therefore $\mathcal{T}_1$ and $\mathcal{T}_2$ have a rSPR distance of two. A MAF is not always unique. The forest in Figure 5.5 is also a MAF for the two trees in Figure 5.1

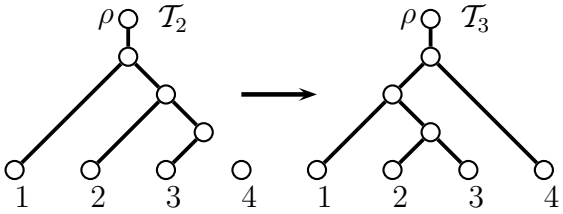


Figure 5.3: Performing one rSPR operation on the tree $\mathcal{T}_2$ from Figure 5.1.

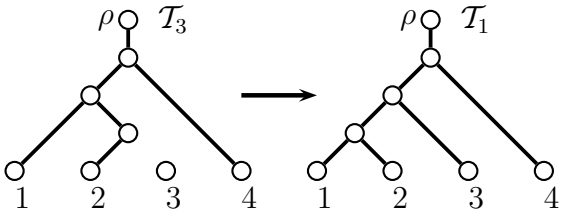


Figure 5.4: Performing one rSPR operation on the tree $\mathcal{T}_3$ from Figure 5.3.

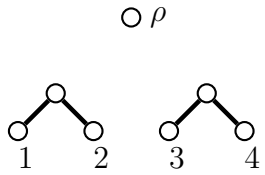


Figure 5.5: An alternative MAF for the two trees in Figure 5.1.

Not every sequence of rSPR moves described by a forest will travel through RDT space. For example, consider the following two duplication trees in Figure 5.6. The forest $\mathcal{F}$

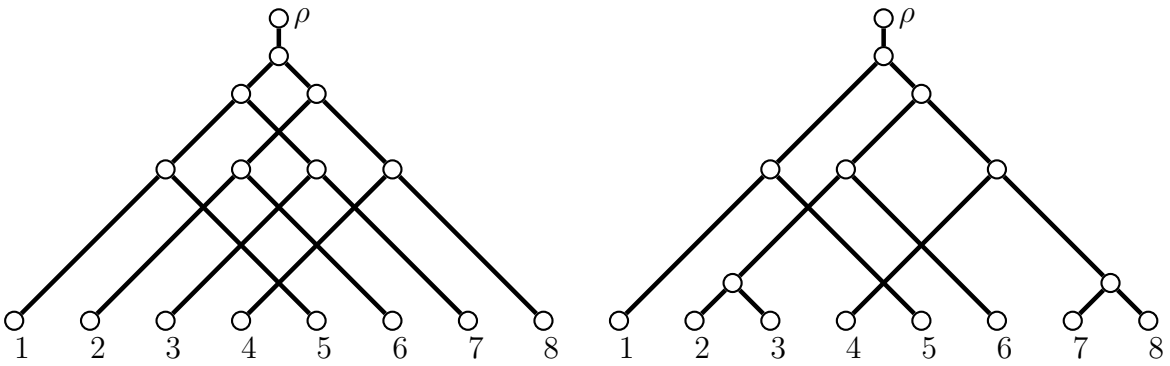


Figure 5.6: Two duplication trees.

in Figure 5.7 is a MAF for these two duplication trees. However there is no sequence of rSPR moves described by this forest that can transform one into the other traveling through RDT space.

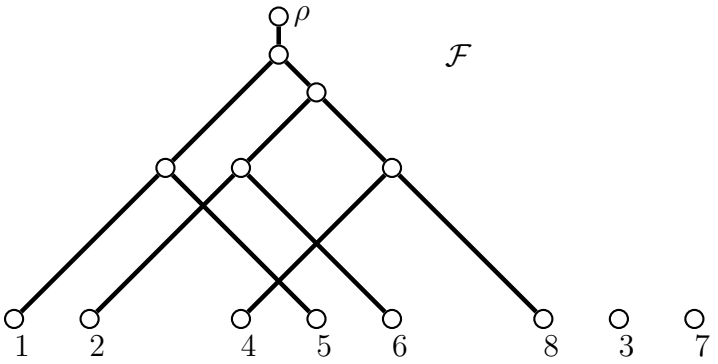


Figure 5.7: A MAF for the trees in Figure 5.6 that does not describe a series of rSPR operations that transforms one tree into the other traveling through RDT space.

In [7] it was shown that for two phylogenetic X-trees, their rSPR distance satisfies the equation $d_{rSPR}(\mathcal{T}_1, \mathcal{T}_2) = m(\mathcal{T}_1, \mathcal{T}_2)$ where $m(\mathcal{T}_1, \mathcal{T}_2)$ is the number of components in a MAF for the two trees minus one. We want to do a similar thing for duplication trees allowing us to calculate the DrSPR distance for two duplication trees. As not every sequence of SPR moves described by an agreement forest produces a duplication tree, we need a more restrictive definition for a duplication agreement forest (DAF). Such a DAF

will produce a sequence of DrSPR moves that will transform one duplication tree into the other.

Suppose we have two duplication trees $\mathcal{T}_1$ and $\mathcal{T}_2$ with the same label set X and ordering O on X. A *duplication agreement forest* (DAF) for $\mathcal{T}_1$ and $\mathcal{T}_2$ is a forest of rooted trees $\mathcal{F} = \{t_\rho, t_1, \ldots, t_k\}$ with label sets $\mathcal{L}_\rho, \mathcal{L}_1, \ldots, \mathcal{L}_k \subseteq X$ such that the following is satisfied:

1. $\mathcal{F}$ is an agreement forest for $\mathcal{T}_1$ and $\mathcal{T}_2$.

2. If there exists a tree $t_i \in \mathcal{F}$ such that $i \geq 1$ and the root of t_i is l_m or r_m in the duplication event with leaves $l_1, \ldots, l_m, \ldots, l_n, r_1, \ldots, r_m, \ldots, r_n$ in either of the trees, then either $l_m, \ldots, l_n$ are roots of trees in $\mathcal{F}$ or $r_1, \ldots, r_m$ are roots of trees in $\mathcal{F}$.

If we have a forest $\mathcal{F}$ such that condition (2) is satisfied for all trees whose root is l_j where $j > 1$ and r_j where $j < n$ and that satisfies condition (1), then we say $\mathcal{F}$ is a *DAF without equality*. This is a weaker result and cannot be guaranteed to describe a sequence of DrSPR moves that transforms one duplication tree into the other.

Furthermore, a *maximum duplication agreement forest* (MDAF) is a DAF that describes a sequence of DrSPR moves with the fewest possible moves. If there exists a tree $t \in \mathcal{F}$ that could be grafted to another tree in the forest such that the forest would still satisfy condition (1), then we say t is *forced*. Note that, unlike a MAF for regular rSPR, it may not be the case that a MDAF has the fewest components. This is because forced trees need to be pruned and regrafted twice. A MDAF might have one more component then another DAF, but the DAF may have two more forced trees in it meaning that the number of DrSPR moves it describes is actually more than the minimum number of DrSPR moves needed. It is unknown whether this occurs, but is something to watch out for. Note also that it is not the case that every tree in a duplication agreement forest must be a duplication tree. To see this, consider the two trees in Figure 5.8. The forest $\mathcal{F}$ in Figure 5.9 is a duplication agreement forest for $\mathcal{T}_1$ and $\mathcal{T}_2$, yet the tree in $\mathcal{F}$ containing the root is not a duplication tree.

This given definition for a DAF is a more restrictive definition than that of an agreement forest. Lemma 5.1 will justify the need for the extra restriction. After we have

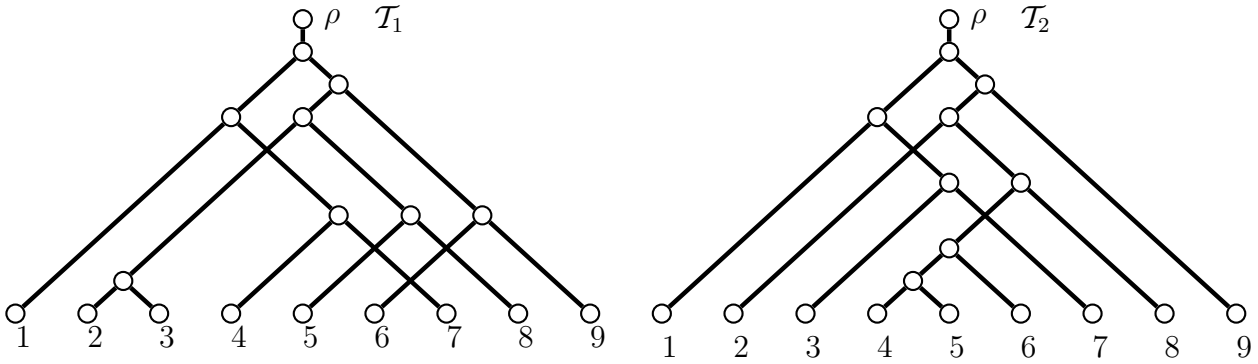


Figure 5.8: Two duplication trees.

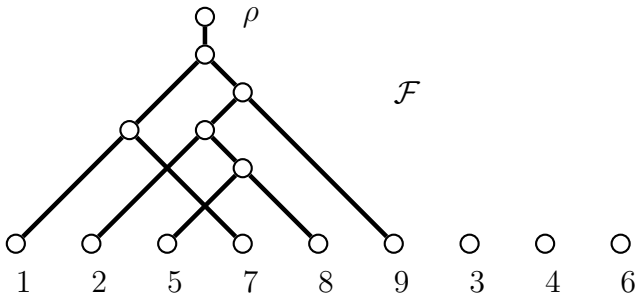


Figure 5.9: A MDAF for the two duplication trees in Figure 5.8 such that the tree that contains the root is not a duplication tree.

justified the extra condition, Theorem 5.2 will show that any DAF for two duplication trees $\mathcal{T}_1$ and $\mathcal{T}_2$ describes a sequence of DrSPR moves that transforms $\mathcal{T}_1$ into $\mathcal{T}_2$.

Lemma 5.1. *Suppose we have the duplication event in Figure 5.10 in some duplication tree $\mathcal{T}$ and a tree with root l_i or r_i in an agreement forest $\mathcal{F}$. If $\mathcal{F}$ is an agreement forest*

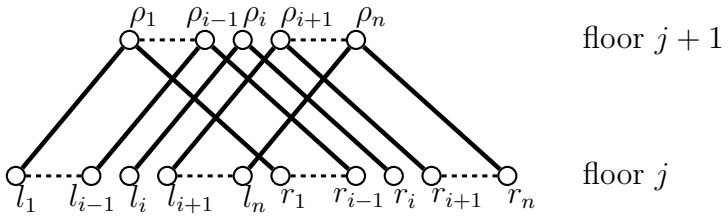


Figure 5.10: A sequence of cherries.

that describes a sequence of DrSPR moves, then either $l_{i+1}, \ldots, l_n$ must be roots of trees in $\mathcal{F}$ or $r_1, \ldots, r_{i-1}$ must be roots of trees in $\mathcal{F}$. That is, a forest describes a sequence of DrSPR moves only if it is a DAF without equality.

Proof. Condition (1) is necessary to describe a sequence of SPR moves that transforms $\mathcal{T}_1$ into $\mathcal{T}_2$ so it is clearly necessary to describe a sequence of DrSPR that transforms $\mathcal{T}_1$ into $\mathcal{T}_2$.

Now for condition (2). Assume that we prune the tree with root l_i and regraft it to any edge above floor j that gives a tree $\mathcal{T}' \ncong \mathcal{T}$ without pruning the trees with roots $l_{i+1} \ldots, l_n$ or $r_1, \ldots, r_{i-1}$. This will not change the unique set of duplication reductions that can be reduced to bring us to floor j. Once this is done, we can reduce all visible duplication events that do not have a leaf in $l_1, \ldots, l_n, r_1, \ldots, r_n$. When this is done, if $\mathcal{T}'$ is a duplication tree, then we should be able to reduce it to its root. However, because l_i has been pruned and regrafted, $l_1, \ldots, l_n, r_1, \ldots, r_n$ is no longer a duplication event and therefore $\mathcal{T}'$ is not a duplication tree. Hence, $\mathcal{F}$ does not describe a sequence of DrSPR moves.

Now assume that the tree with root l_i was regrafted to any edge below floor j, that gives a tree $\mathcal{T}' \ncong \mathcal{T}$ without pruning the trees with roots $l_{i+1} \ldots, l_n$ or $r_1, \ldots, r_{i-1}$. If we start with an isolated vertex and start performing duplication events corresponding to the events in $\mathcal{T}'$, then when we reach floor $j+1$ we will no longer be able to perform the necessary duplication event to get $l_1, \ldots, l_n, r_1, \ldots, r_n$. So once again $\mathcal{T}$ is not a duplication tree, and thus $\mathcal{F}$ does not describe a sequence of DrSPR moves.
Replacing l_i with r_i proves the case for pruning and regrafting the tree with root r_i. This shows that we cannot prune and regraft trees with roots l_i or r_i without first pruning and regrafting trees with roots $l_{i+1}, \ldots, l_n$ or $r_1 \ldots, r_{i-1}$. If there are no trees in the forest with roots $l_{i+1}, \ldots, l_n$ or $r_1 \ldots, r_{i-1}$, then we cannot prune and regraft them. This would mean that we could not prune and regraft a tree with root l_i or r_i and stay in RDT space. Therefore, it is necessary for a forest $\mathcal{F}$ to be a DAF without equality to describe a sequence of DrSPR moves that transforms $\mathcal{T}_1$ into $\mathcal{T}_2$.

□

Lemma 5.1 shows that if we want to prune and regraft a tree whose root is in a duplication event, then we have to work from the inside out, pruning and regrafting all trees whose roots are on the inside until we reach the tree we want to prune and regraft. In [4] an operation DELETE was described. This was a means of pulling apart a multiple duplication event while still maintaining the property that the tree was a duplication tree. Here we describe a slight modification on this operation called DELETE2. The operation DELETE2 will be used in Theorem 5.2. Suppose we have the multiple duplication event shown in Figure 5.11. We can prune the tree with root B and regraft it to either the edge incident with A of C while still maintaining the property that the tree is a duplication tree. The the two possibilities after this operation is shown in Figure 5.12. We could apply similar operations to the tree denoted by C in Figure 5.11. Just like the DELETE operation, DELETE2 will always preserve the duplication tree property.

Theorem 5.2. *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two duplication trees with the same ordering O and let $\mathcal{F}$ be a forest. The forest $\mathcal{F}$ describes a sequence of DrSPR moves that transforms $\mathcal{T}_1$ into $\mathcal{T}_2$ if $\mathcal{F}$ is a DAF*

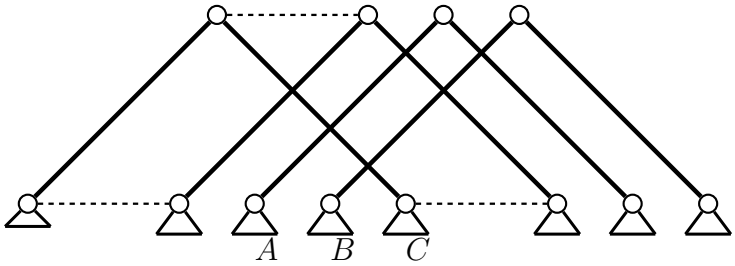


Figure 5.11: A duplication event.

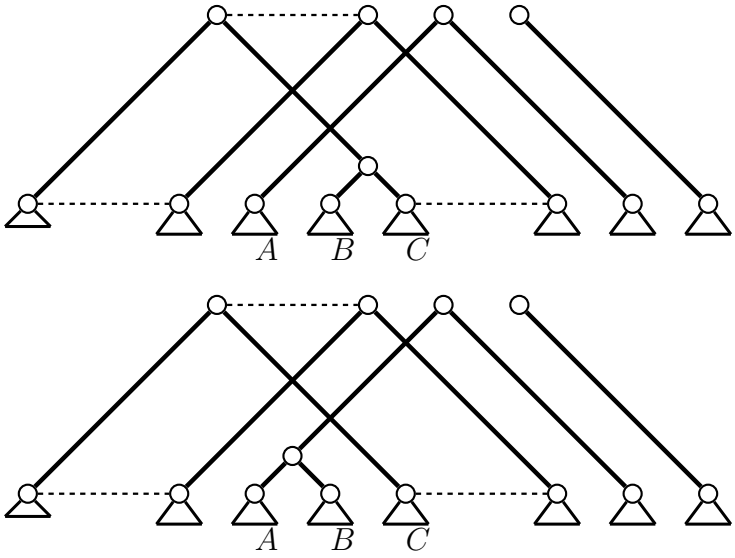


Figure 5.12: Two possible DELETE2 operations on the subtree labeled B from the duplication event in Figure 5.11

Proof. Assume that we have two duplication trees $\mathcal{T}_1$ and $\mathcal{T}_2$ and a forest of k components that is a DAF.

If $k = 2$ then we can prune and regraft one of the trees in $\mathcal{F}$ that does not contain the root to make the other tree. Because the trees $\mathcal{T}_1$ and $\mathcal{T}_2$ are duplication trees, this describes a sequence of DrSPR moves.

Now assume that any forest of size at most k that is a DAF describes a sequence of rSPR moves through RDT space. Suppose we have a forest of size $k + 1$. To begin with, we will assume that the root of every tree in $\mathcal{F}$ is part of a simple duplication event in both trees. If this is the case, then pick any tree $t_i \in \mathcal{F}$ such that no root of any other tree in $\mathcal{F}$ is an ancestor of the root of t_i in $\mathcal{T}_1$. Because t_i is in a simple duplication event in both trees, the root of t_i will be on the path from f to g in both trees, where f is the largest element in O that is less than all leaves of t_i and g is the smallest element of O that is greater than all leaves of t_i. Because of this we can prune t_i from its position in $\mathcal{T}_1$ and regraft it to its position in $\mathcal{T}_2$. This is depicted in Figure 5.13. The resulting tree

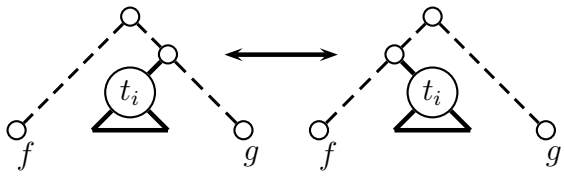


Figure 5.13: A DrSPR operation on a simple duplication tree.

$\mathcal{T}_3$ will be a duplication tree. Now the size of the forest for $\mathcal{T}_2$ and $\mathcal{T}_3$ is at most k and by the induction hypothesis describes a sequence of rSPR moves that transforms $\mathcal{T}_3$ into $\mathcal{T}_2$ traveling through RDT space. As we got to $\mathcal{T}_3$ from $\mathcal{T}_1$ by traveling through RDT space, we can transform $\mathcal{T}_1$ into $\mathcal{T}_2$ by DrSPR moves.

Now suppose that there exist some trees in $\mathcal{F}$ that are in a multiple duplication event in $\mathcal{T}_1$ or $\mathcal{T}_2$.

The DELETE2 rearrangement can be used to pull apart multiple duplication events by working from the inside out. Because the forest is a DAF, any multiple duplication event can be pulled apart from the inside until it no longer includes the roots of any trees

in $\mathcal{F}$. After we perform DELETE2 on all possible multiple duplication events, the two resulting trees will all have the same multiple events. If we do this rearrangement to all trees in $\mathcal{T}_1$ that are in a multiple duplication event, then we will obtain a tree $\mathcal{T}_3$ such that all trees in $\mathcal{F}$ are in simple duplication events in $\mathcal{T}_3$. We can obtain the tree $\mathcal{T}_4$ by doing the same rearrangement to $\mathcal{T}_2$. Now all trees in $\mathcal{F}$ are in simple duplication events in both $\mathcal{T}_3$ and $\mathcal{T}_4$ and we have just shown above that if this is the case, then we can transform $\mathcal{T}_3$ into $\mathcal{T}_4$ by a series of DrSPR moves described by $\mathcal{F}$, for all forests $\mathcal{F}$ with no more than $k+1$ components. As we can transform $\mathcal{T}_1$ into $\mathcal{T}_3$ and $\mathcal{T}_2$ into $\mathcal{T}_4$ by DrSPR moves described by $\mathcal{F}$, we can transform $\mathcal{T}_1$ into $\mathcal{T}_2$ by a sequence of moves described by $\mathcal{F}$. This shows by induction that if $\mathcal{F}$ is a DAF, then $\mathcal{F}$ describes a sequence of DrSPR moves that transforms $\mathcal{T}_1$ into $\mathcal{T}_2$ only traveling through RDT space.

□

Note that in proving Theorem 5.2 we have also shown that you can always transform one duplication tree into another on the same label set traveling only through RDT space. This is not the case for duplication agreement forests without equality. For example, consider the two duplication trees in Figure 5.14. The forest shown in Figure 5.15 is a DAF without equality but does not describe a sequence of DrSPR moves that transforms $\mathcal{T}_1$ into $\mathcal{T}_2$.

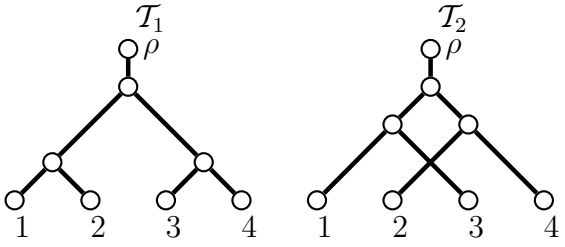


Figure 5.14: Two duplication trees.

Now that we have a duplication agreement forest, the next step is to calculate the distance between two duplication trees. Bounds on this distance are given in Corollary 5.3.

Corollary 5.3. *Let $(\mathcal{T}_1, O)$ and $(\mathcal{T}_2, O)$ be two duplication trees with the same ordering O and $\mathcal{F}$ be a MDAF on $\mathcal{T}_1$ and $\mathcal{T}_2$ with k_1 components and $\mathcal{F}'$ be a MDAF without equality*

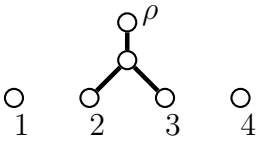


Figure 5.15: A DAF without equality for the two duplication trees in Figure 5.14 that does not describe a sequence of DrSPR moves that transforms one tree into the other.

with k_2 components. Then

$$(k_2 - 1) \leq d_{DrSPR}(\mathcal{T}_1, \mathcal{T}_2) \leq 3(k_1 - 1),$$

where d_{DrSPR} is the DrSPR distance between the two trees $\mathcal{T}_1$ and $\mathcal{T}_2$.

Proof. The lower bound follows from the fact that $d_{rSPR}(\mathcal{T}_1, \mathcal{T}_2) = m(\mathcal{T}_1, \mathcal{T}_2)$ [7] and that a forest describes a sequence of DrSPR moves only if it is a DAF without equality.
The upper bound follows from Theorem 5.2. We can perform DELETE2 on all trees in $\mathcal{F}$ on $\mathcal{T}_1$ and $\mathcal{T}_2$ to obtain trees $\mathcal{T}_3$ and $\mathcal{T}_4$ which only differ in simple duplication events. This is a total of at most $2(k-1)$ moves. Now it follows from the first part of the induction proof in theorem 5.2 that we can transform $\mathcal{T}_3$ into $\mathcal{T}_4$ with at most $(k_1 - 1)$ moves. Therefore

$$(k_2 - 1) \leq d_{DrSPR}(\mathcal{T}_1, \mathcal{T}_2) \leq 3(k_1 - 1).$$

□

Conjecture 5.4. *If $\mathcal{T}_1$ and $\mathcal{T}_2$ are two duplication trees with the same ordering O and $\mathcal{F}$ is a MDAF with k components, then $d_{DrSPR}(\mathcal{T}_1, \mathcal{T}_2) = 2q + r$, where $q + r = k - 1$ and q is the number of trees in $\mathcal{F}$ that are forced.*

If we want to stay in RDT space, we can only prune and regraft subtrees that are either l_n or r_1 of a duplication event. Because of this, we may need to prune and regraft a subtree once and then later in the sequence we may need to prune the same tree again and regraft it back to its starting position. Every subtree that this happens to will have been forced. It seems likely that we only need to prune and regraft each forced tree twice. Once to get it out of the way, then once again to place it where it needs to be. If a tree

is not forced, then it only needs to be moved once in the sequence of DrSPR moves. This gives an upper bound of $2q + r$ moves.

5.2 The DrSPR Neighbourhood on Simple Duplication Trees

For the remainder of this chapter, we will only be concerned with DrSPR moves on simple duplication trees.

Often we want to find an optimal tree $\mathcal{T}$ for some given criteria. A common method for this is hill climbing. In hill climbing, we start with an initial tree that is either random or chosen specifically. We search the neighbourhood of trees $\mathcal{T}'$ that are distance 1 from $\mathcal{T}$, looking for the tree $\mathcal{T}'$ in the neighbourhood with highest/lowest score under the given criteria. If there is no such $\mathcal{T}'$, then we output $\mathcal{T}$. Otherwise, we repeat this process for $\mathcal{T}'$. Hill climbing is guaranteed to always find local extrema but can get stuck at these and miss the global extrema. Due to the way that hill climbing works, it is useful to know the size of the neighbourhood for a given tree. If the size of the neighbourhood of $\mathcal{T}$ is almost as large as the number of trees with the same number of leaves as $\mathcal{T}$, then hill climbing will be no better than an exhaustive search. Robinson [25] showed that the size of the neighbourhood for NNI on a tree with n leaves was given by $2n - 6$. The size of the SPR neighbourhood was shown to be $2(n-3)(2n-7)$ by Allen and Steel in [2]. Also in [2] it was shown that the size of the TBR neighbourhood was dependant on the shape of the tree $\mathcal{T}$. The bound

$$cn^2 \log n + O(n^2) \leq |N_{TBR}(\mathcal{T})| \leq \frac{2}{3}n^3 - 4n^2 + \frac{16}{3}n + 2$$

for the size of the TBR neighbourhood denoted by $|N_{TBR}(\mathcal{T})|$ was given in [18]. This was later improved to

$$|N_{TBR}(\mathcal{T})| = 4\Gamma(\mathcal{T}) - (4n-2)(n-3)$$

in [17], where

$$\Gamma(\mathcal{T}) = \sum |A| \cdot |B|$$

and is taken over all all non-trivial splits $A|B$ of $\mathcal{T}$.
In this section we are interested in calculating the size of the DrSPR neighbourhood of simple duplication trees, which we will denote as $\eta(\mathcal{T})$ for the simple duplication tree $\mathcal{T}$. The size of the DrSPR neighbourhood of a simple duplication tree is dependant on the shape of the tree.

This section is broken up into three subsections. In Subsection 5.2.1 we will provide formulas for the number of simple duplication trees with n leaves. Subsection 5.2.2 will cover all possible DrSPR moves on simple duplication trees. Then finally in Subsection 5.2.3 we will provide a formula for calculating the size of the DrSPR neighbourhood of an arbitrary simple duplication tree.

5.2.1 Simple Duplication Trees

If we are calculating the size of the neighbourhood of a simple duplication tree in order to use hill climbing methods, then it would be useful to know the number of possible simple duplication trees with the same number of leaves. Two different formulas are given for this in Lemmas 5.5 and 5.6

Lemma 5.5. *Let $\Omega(n)$ be the number of simple duplication trees on $n \geq 2$ leaves. This is given by the following recursive equation.*

$$\Omega(n) = \sum_{i=1}^{n-1} \Omega(i) \cdot \Omega(n-i),$$

where $\Omega(1) = 1$.

Proof. Let $(\mathcal{T}, O)$ be a simple duplication tree with $n \geq 2$ leaves. The first duplication event coming down from the root in $\mathcal{T}$ will partition the label set X into two disjoint sets S_1 and S_2 such that $s_1 <_O s_2$ for all $s_1 \in S_1$, $s_2 \in S_2$ and $|S_1| > 0, |S_2| > 0$. There are $n-1$ different possible partitions that give the required sets S_1 and S_2.

Consider the partition that gives the set $|S_1| = i$. Then $|S_2| = n - i$. The number of different simple duplication trees that have i leaves is given by $\Omega(i)$, and there are $\Omega(n-i)$ different possible simple duplication trees on $n-i$ leaves. This gives $\Omega(i)\cdot\Omega(n-i)$ different possible simple duplication trees on n leaves where the first duplication event partitions the label set X into S_1 and S_2 where $|S_1| = i$ and $|S_2| = n - i$. The value of i can range from 1 to $n-1$. Therefore the number of unique simple duplication trees on $n \geq 2$ leaves is given by the recursive equation

$$\Omega(n) = \sum_{i=1}^{n-1} \Omega(i) \cdot \Omega(n-i).$$

□

While this recursive equation is simple and gives us a good intuition about what is happening, a closed form expression would still be preferable. Theorem 5.6 provides us with such a closed form.

Theorem 5.6. *The number of simple duplication trees on $n \geq 2$ leaves denoted by $\Omega(n)$ is given by*

$$\Omega(n) = \prod_{j=2}^{n} \frac{4j-6}{j}.$$

Proof. When $n = 2$ we have

$$\Omega(2) = \frac{4 \cdot 2 - 6}{2} = 1.$$

There is only one possible simple duplication tree with 2 leaves. This proves the base case.

Now assume that this is true for $2, \ldots, n-1$. From [27] (page 17) we obtain

$$|RB(n)| = (2n-3) \cdot |RB(n-1)| = (2n-3)!!,$$

where $|RB(n)|$ is the number of rooted binary trees on n leaves. However, not every rooted binary tree on n leaves is a simple duplication tree. For every edge e we add to a simple duplication tree $\mathcal{T}$ on $n-1$ leaves, there are n places we can insert it in the

ordering. Of the n places we can insert our edge e, only two will give a simple duplication tree $\mathcal{T}'$. This gives

$$\Omega(n) = \frac{2(2n-3)}{n} \cdot \Omega(n-1) = \frac{2(2n-3)}{n} \cdot \prod_{j=2}^{n-1} \frac{4j-6}{j} = \prod_{j=2}^{n} \frac{4j-6}{j}.$$

□

A curious result about $\Omega(n)$ is that the m greatest primes in the prime factorization seem to all have degree one and to be a list of those primes in order (for some m). For example, below is the prime factorizations for selected values of n between 58 and 70. We can see that the last m primes all have degree one and they do not skip a prime.

$$
\begin{aligned}
&\vdots \\
\Omega(58) &= (2)^3(3)(31)(37) \\
&\quad \underbrace{(59)(61)(67)(71)(73)(79)(83)(89)(97)(101)(103)(107)(109)(113)} \\
\Omega(59) &= (2)^4(3)(5)(23)(31)(37) \\
&\quad \underbrace{(61)(67)(71)(73)(79)(83)(89)(97)(101)(103)(107)(109)(113)} \\
&\vdots \\
\Omega(64) &= (3)(5)^3(11)^2(13)(17)(23)(37)(41) \\
&\quad \underbrace{(67)(71)(73)(79)(83)(89)(97)(101)(103)(107)(109)(113)} \\
\Omega(65) &= (2)(3)(5)^2(11)^2(17)(23)(37)(41) \\
&\quad \underbrace{(67)(71)(73)(79)(83)(89)(97)(101)(103)(107)(109)(113)(127)} \\
\Omega(66) &= (2)(3)(5)^2(11)(17)(23)(37)(41)(43) \\
&\quad \underbrace{(67)(71)(73)(79)(83)(89)(97)(101)(103)(107)(109)(113)(127)} \\
\Omega(67) &= (2)^2(3)(5)^2(11)(17)(23)(37)(41)(43) \\
&\quad \underbrace{(71)(73)(79)(83)(89)(97)(101)(103)(107)(109)(113)(127)(131)} \\
&\vdots \\
\Omega(69) &= (2)^2(3)^3(5)^3(7)(11)(19)(37)(41)(43) \\
&\quad \underbrace{(71)(73)(79)(83)(89)(97)(101)(103)(107)(109)(113)(127)(131)} \\
\Omega(70) &= (2)^2(3)^3(5)^2(11)(19)(37)(41)(43) \\
&\quad \underbrace{(71)(73)(79)(83)(89)(97)(101)(103)(107)(109)(113)(127)(131)(137)} \\
&\vdots
\end{aligned}
$$

5.2.2 Simple DrSPR moves

In this section we will calculate all possible DrSPR moves on simple duplication trees. This will provide us with a way of exploring the DrSPR neighbourhood of an arbitrary simple duplication tree. First we will show that the DrSPR distance of two simple duplication trees is the same as their rSPR distance. This will be done in Lemma 5.8. The proof of this will require Lemma 5.7. After this, we will calculate all possible DrSPR moves on a simple duplication tree such that the resulting tree is still a simple duplication tree. We will end this section by calculating what possible simple DrSPR moves produce unique simple duplication trees. We will refer to a DrSPR move that takes a simple duplication tree to another simple duplication tree as a *simple DrSPR move*

Lemma 5.7. *Every restriction of a simple duplication tree is a simple duplication tree.*

Proof. Let $\mathcal{T}$ be a simple duplication tree and $\mathcal{T}'$ be some restriction of $\mathcal{T}$. If $\mathcal{T}'$ is not a duplication tree, then after a number of duplication reductions, none of the visible cherries of $\mathcal{T}'$ will be able to be reduced. If this is the case, then every visible cherry must have width greater than zero. As $\mathcal{T}'$ is displayed in $\mathcal{T}$, if there is a cherry in $\mathcal{T}'$ with width greater than zero, then there must be a visible cherry in $\mathcal{T}$ with width greater than zero after some number of duplication reductions. But $\mathcal{T}$ is a simple duplication tree and so this is a contradiction as after any set of duplication reductions, every visible cherry of $\mathcal{T}$ has width zero. Therefore every restriction of a simple duplication tree is a simple duplication tree. □

If $\mathcal{T}$ is not a simple duplication tree, then it is possible that a restriction of $\mathcal{T}$ will not be a duplication tree. For example the tree $\mathcal{T}$ in Figure 5.16 is a duplication tree. However, as also shown in Figure 5.16, the restriction $\mathcal{T}|\{1,2,3\}$ is not a duplication tree.

Intuitively it seems likely that if the two trees are simple duplication trees, then any MAF is also a MDAF. Indeed this is true as shown in Lemma 5.8. Knowing this gives us an upperbound on the number of rearrangements required to transform $\mathcal{T}_1$ into $\mathcal{T}_2$ traveling through RDT space.

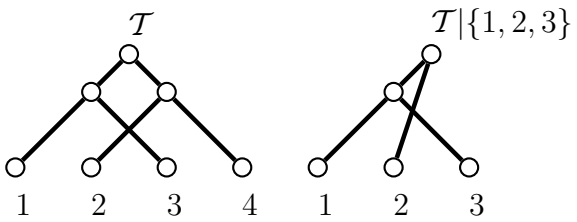


Figure 5.16: A restriction of a duplication tree that is not a duplication tree.

Lemma 5.8. *Let $(\mathcal{T}_1, O)$ and $(\mathcal{T}_2, O)$ be simple duplication trees with the same ordering O. If $\mathcal{F}$ is a MDAF for $\mathcal{T}_1$ and $\mathcal{T}_2$ and $\mathcal{F}'$ is a MAF for $\mathcal{T}_1$ and $\mathcal{T}_2$, then $|\mathcal{F}| = |\mathcal{F}'|$. Furthermore, $d_{DrSPR}(\mathcal{T}_1, \mathcal{T}_2) = d_{rSPR}(\mathcal{T}_1, \mathcal{T}_2)$, where d_{rSPR} is the rSPR distance and d_{DrSPR} is the DrSPR distance of $\mathcal{T}_1$ and $\mathcal{T}_2$.*

Proof. If $\mathcal{F}$ is a MDAF, then by definition, $\mathcal{F}$ is an agreement forest. Therefore $|\mathcal{F}| \geq |\mathcal{F}'|$. Because $\mathcal{T}_1$ and $\mathcal{T}_2$ are simple duplication trees, no tree $t_i \in \mathcal{F}'$ is forced. Because of this, condition (3) is satisfied (with equality). Any $t_i \in \mathcal{F}'$ is a restriction of $\mathcal{T}_1$ and $\mathcal{T}_2$. Any restriction of a simple duplication tree is itself a duplication tree (Lemma 5.7), therefore t_i is a duplication tree for all $t_i \in \mathcal{F}'$. This shows that $\mathcal{F}'$ satisfies conditions (1) to (3). Therefore $\mathcal{F}'$ is a duplication agreement forest. This gives $|\mathcal{F}| \leq |\mathcal{F}'|$ and therefore $|\mathcal{F}| = |\mathcal{F}'|$.

Now, suppose $|\mathcal{F}| = |\mathcal{F}'| = k$. If $d_{DrSPR}(\mathcal{T}_1, \mathcal{T}_2) < k-1$ then there would be a sequence of $DrSPR$ moves that transforms $\mathcal{T}_1$ to $\mathcal{T}_2$ with less than $k-1$ moves. This sequence of moves is a restricted sequence of $rSPR$ moves, and so $d_{rSPR}(\mathcal{T}_1, \mathcal{T}_2) < k-1$. This is a contradiction as Bordewich and Semple showed in [7] that $d_{rSPR}(\mathcal{T}_1, \mathcal{T}_2) = k-1$. Hence, $d_{DrSPR}(\mathcal{T}_1, \mathcal{T}_2) \geq k-1$

Let $m_D(\mathcal{T}_1, \mathcal{T}_2)$ be the number of components in a maximum duplication agreement forest for the trees $\mathcal{T}_1$ and $\mathcal{T}_2$.

If $k = 2$, then there is only one subtree t_i that needs to be moved to transform $\mathcal{T}_1$ into $\mathcal{T}_2$ and vice versa. We can do this in one move. Therefore if $k = 2$, then $d_{DrSPR}(\mathcal{T}_1, \mathcal{T}_2) = k - 1 = 1$.

Now assume that $d_{DrSPR}(\mathcal{T}_1, \mathcal{T}_2) \leq k-1$ for all forests of size at most k. Take a pair of simple duplication trees $\mathcal{T}_1$ and $\mathcal{T}_2$ such that their MDAF has size at most $k+1$. Without loss of generality there exists at least one tree $t_i \in \mathcal{F}$ that needs to be moved to its right in $\mathcal{T}_1$ to place it in the same position as in $\mathcal{T}_2$. That is, the tree is moved so that the cluster of its root's immediate ancestor has a greater largest element than before the move. Take the right most tree that needs to be moved in this fashion in $\mathcal{T}_1$. If we move this to the appropriate place on the right we will get a tree $\mathcal{T}_3$ such that $m_D(\mathcal{T}_1, \mathcal{T}_3) = 2$ and $m_D(\mathcal{T}_2, \mathcal{T}_3) = k-1$. Therefore by the induction hypothesis we have $d_{DrSPR}(\mathcal{T}_1, \mathcal{T}_3) \leq 1$ and $d_{DrSPR}(\mathcal{T}_2, \mathcal{T}_3) \leq k-2$. This gives

$$d_{DrSPR}(\mathcal{T}_1, \mathcal{T}_2) \leq 1 + (k-2) = k-1.$$

Therefore

$$d_{DrSPR}(\mathcal{T}_1, \mathcal{T}_2) = k-1 = d_{rSPR}(\mathcal{T}_1, \mathcal{T}_2).$$

□

Presented below are a few definitions that will be used throughout the remainder of this chapter. Let $(\mathcal{T}, O)$ be a duplication tree with edge set E and vertex set V. We will define for u and $v \in V$, $\delta(u, v) : V \times V \to \mathbb{Z}$ by $\delta(u, v) = |H|$, where H is the set of edges on the path connecting the vertices u and v.

It can be shown that $\delta(u, v)$ is a *metric*, that is for all $u, v, w \in V$

1. $\delta(u, v) = 0$ if and only if $e = f$,
2. $\delta(u, v) = \delta(v, u)$ and
3. $\delta(u, w) \leq \delta(u, v) + \delta(v, w)$.

Every edge is incident with two vertices. One of these will be the descendant of the other. We will denote $\underline{e}$ to be the descendant vertex incident with e and $\overline{e}$ to be the ancestor vertex incident with e. If a tree is pruned and regrafted by cutting the edge e, then we will denote the tree pruned by t_e.

Finally, for any simple duplication tree $(\mathcal{T}, O)$ with the ordering $O = (1, 2, \ldots, n)$ we will define the circumference of $(\mathcal{T}, O)$ to be the path from 1 to n. Furthermore, we will

define the path from 1 to ρ as the left circumference of $(\mathcal{T}, O)$ and the path from n to ρ as the right circumference of $(\mathcal{T}, O)$.

To calculate the neighbourhood of a simple duplication tree, we need to know what SPR moves produce simple duplication trees and of these moves, which ones produce distinct trees. The possible moves will be covered in Lemma 5.9 and of these, the ones that do not produce distinct simple duplication trees will be characterized in Lemma 5.10. We will then prove that all remaining possible DrSPR moves produce distinct simple duplication trees.

Lemma 5.9. *Let $(\mathcal{T}, O)$ be a simple duplication tree. Suppose we obtain the tree $(\mathcal{T}', O)$ by making one SPR move. The tree $\mathcal{T}'$ is a simple duplication tree if and only if the subtree t that was pruned and regrafted was grafted to an edge whose smallest descendant is immediately to the right of the largest descendant of t_e, or whose largest descendant is immediately to the left of the smallest descendant of t_e in the ordering O.*

Proof. For this proof, we will only consider the case where t moves to its left. The case where t moves to its right can be proven in a similar way.
Suppose we prune the subtree t_e and regraft it to an edge whose largest descendant is immediately to the left of the smallest descendant of t_e. The subtree $\mathcal{T}_{sub}$ of $\mathcal{T}$ that this occurs in is shown below in Figure 5.17.

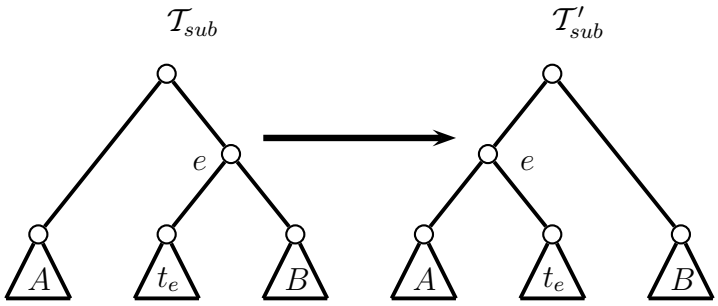


Figure 5.17: A rSPR operation on a simple duplication tree that stays in RDT space.

Because the tree $\mathcal{T}_{sub}$ is a simple duplication tree, the subtrees A, B and t_e must also be simple duplication trees. This implies that the tree $\mathcal{T}'_{sub}$ is also a simple duplication tree and as the rest of $\mathcal{T}$ is not affected, $\mathcal{T}'$ is a simple duplication tree.

Now suppose we pruned the tree t_e and regrafted it to an edge whose largest descendant is not immediately to the left of t_e's smallest descendant. There are two possible ways this can happen. These are shown in Figure 5.18. Because the trees A, B, C and t_e do

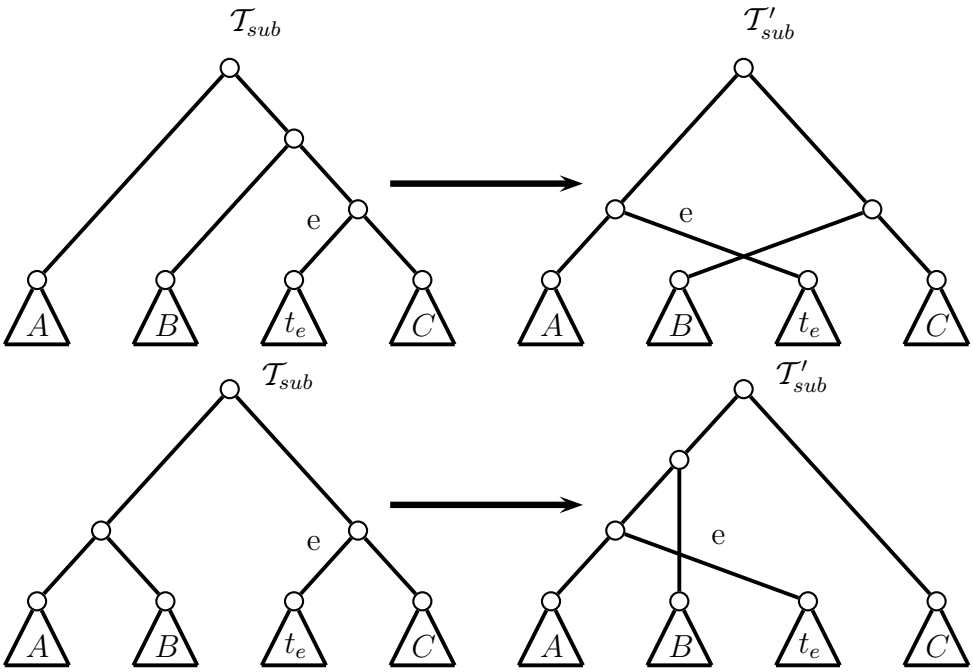


Figure 5.18: The two possible rSPR operations on a simple duplication tree that does not stay in RDT space.

not overlap, the tree $\mathcal{T}'_{sub}$ is not a simple duplication tree.

This shows that the tree $\mathcal{T}'$ is a simple duplication tree if and only if the sub tree t_e that was pruned and regrafted was grafted to an edge whose smallest descendant is immediately to the right of the largest descendant of t_e or whose largest descendant is immediately to the left of t_e's smallest descendant in the ordering O. Note that this will also hold for any edge on the left (right) circumference that is pruned and regrafted to another edge on the left (right) circumference.

□

We will now classify which of the possible simple DrSPR moves do not give distinct simple duplication trees.

Lemma 5.10. *Let $(\mathcal{T}, O)$ be a simple duplication tree with the pendant subtree $\mathcal{T}_{sub}$ shown in Figure 5.19. Let $\mathcal{T}'$ be the tree obtained by pruning the tree t_g and regrafting it to the*

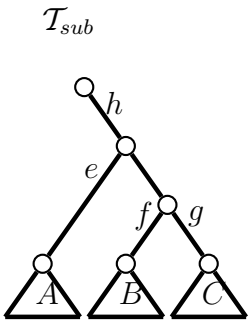


Figure 5.19: Pendent subtree of a duplication tree.

edge h, $\mathcal{T}''$ be the tree obtained by pruning t_e and regrafting it to the edge f and let $\mathcal{T}'''$ be the tree obtained by pruning the tree t_f and regrafting it to the edge e. Then

$$\mathcal{T}' \cong \mathcal{T}'' \cong \mathcal{T}'''.$$

Proof. The three subtrees after their respective DrSPR moves are shown in Figure 5.20. It is clear that these three subtrees are isomorphic. Because the only change in the tree

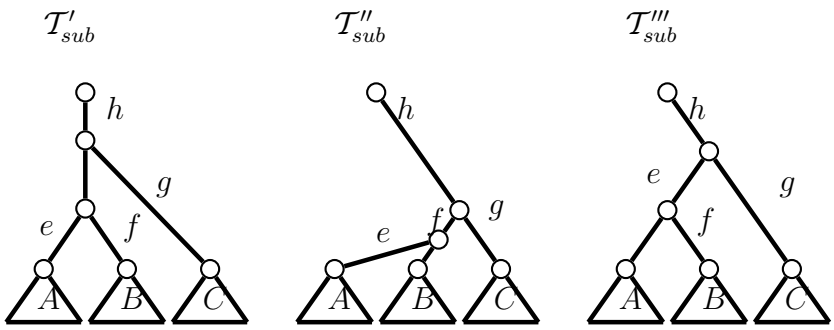


Figure 5.20: Three subtrees made by performing three different rSPR operations on the subtree from Figure 5.19.

$\mathcal{T}$ is in the subtrees shown and the subtrees are isomorphic, we conclude that

$$\mathcal{T}' \cong \mathcal{T}'' \cong \mathcal{T}'''.$$

□

For convenience, for the remainder of this chapter, when given the option from Lemma 5.10, we will always prune t_f and regraft it to the edge e. That is, pruning a tree t_f and regrafting it to an edge e such that $\delta(\underline{e}, \underline{f}) = 3$, e is not a descendant of f, f is not a descendant of e and e is incident with $\text{mrca}(e, f)$. When we talk about the number of distinct DrSPR moves a simple duplication tree has, we will count the move of pruning t_f and regrafting it to the edge e as distinct.

Lemmas 5.11 and 5.12 show that all remaining DrSPR moves on a tree $\mathcal{T}$ that give a simple duplication tree $\mathcal{T}' \ncong \mathcal{T}$ are distinct.

Lemma 5.11. *Let $(\mathcal{T}, O)$ be a simple duplication tree and e and f be distinct edges of $\mathcal{T}$. Let j be an edge such that e is a descendant of j and $\delta(\underline{e}, \underline{j}) \geq 3$. If we prune t_e and regraft it to j to create $\mathcal{T}'$, there is no simple duplication tree $\mathcal{T}''$ such that $\mathcal{T}' \cong \mathcal{T}''$ and $\mathcal{T}''$ was obtained by pruning regrafting f.*

Proof. Let E be the cluster induced by the vertex $\underline{e}$ in $\mathcal{T}$ and J be the cluster induced by the vertex $\underline{j}$ in $\mathcal{T}$. Consider the set of clusters induced by $\mathcal{T}'$. As well as the cluster E, there will be a cluster $\{J \cup E\}$ and a cluster $\{J \backslash \{E\}\}$. This is because we now have a duplication event where one immediate descendant is $\underline{j}$ and the other is $\underline{e}$. Because e is a descendant of j, the set of clusters induced by $\mathcal{T}''$ will induce no cluster $\{J \backslash \{E\}\}$. This is because $\delta(\underline{e}, \underline{j}) \geq 3$, which implies that there are two duplication events on the path between $\underline{j}$ and $\underline{e}$ that do not include e. We can only move one edge so after we prune and regraft f, there will still be at least one duplication event on this path that does not include e. This can be seen in Figure 5.21 below. If we remove f, there will still be a

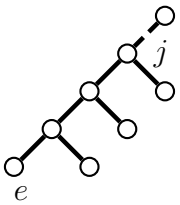


Figure 5.21: Subtree of the tree $\mathcal{T}$ that contains the edges e and j.

duplication event between j and e so there can be no cluster $\{J \backslash \{E\}\}$ where $E \subset J$.

Hence, if e is a descendant of j then there exists no such $\mathcal{T}''$.

□

Lemma 5.12. *Let $(\mathcal{T}, O)$ be a simple duplication tree and e and f be two distinct edges of $\mathcal{T}$. Let $\mathcal{T}_e$ be the simple duplication tree obtained by pruning and regrafting the tree t_e such that $\mathcal{T} \ncong \mathcal{T}_e$. Also let $\mathcal{T}_f$ be the simple duplication tree created by pruning and regrafting the tree t_f such that $\mathcal{T} \ncong \mathcal{T}_f$. If $\delta(\underline{e}, \underline{f}) > 3$ or $\underline{e}$ is a descendant of $\underline{f}$ or $\underline{f}$ is a descendant of $\underline{e}$, then $\mathcal{T}_e \ncong \mathcal{T}_f$.*

Proof. Let j be the edge that t_f is grafted to to create $\mathcal{T}_f$ and i the edge that t_e is grafted to to create $\mathcal{T}_e$. To begin with, assume that $\underline{e}$ is a descendant of $\underline{f}$. The set of clusters induced by the tree $\mathcal{T}_f$ will have a cluster $\{J, F\}$ where the sets J and F are the clusters induced by $\underline{j}$ and $\underline{f}$ respectively in $\mathcal{T}$and $\{J\} \cap \{F\} = \emptyset$. If $\mathcal{T}_e \cong \mathcal{T}_f$ then the set of clusters induced by $\mathcal{T}_e$ must also have a cluster $\{J, F\}$. As only a subtree of the subtree with root $\underline{f}$ has been pruned and regrafted, for $\mathcal{T}_e$ to have the cluster $\{J, F\}$, the tree $\mathcal{T}$ must have also induced this cluster. But $\mathcal{T}_f$ was made from pruning the subtree with root $\underline{f}$ and regrafting it to j. This implies that $\mathcal{T} \cong \mathcal{T}_f$. This is a contradiction and therefore if $\underline{f}$ is a descendant of $\underline{e}$ or $\underline{e}$ is a descendant of $\underline{f}$, then $\mathcal{T}_e \ncong \mathcal{T}_f$.

So we assume that e is not a descendant of f and f is not a descendant of e and that $\delta(e, f) > 3$.

If $j = e$ then the tree $\mathcal{T}_f$ induces the cluster $\{E, F\}$ where $\{E\} \cap \{F\} = \emptyset$. The tree $\mathcal{T}$ can not induce this cluster. If it did, then $\mathcal{T} \cong \mathcal{T}_f$ and we would have a contradiction. If the tree $\mathcal{T}_e$ also induces this cluster then i is descendant of f. If i is a descendant of f but $i \neq f$, then in the set of clusters induced by $\mathcal{T}_e$, every cluster that contains F will also contain E. Now consider the set of clusters induced by $\mathcal{T}_f$. Because there is a duplication event such that one descendant of this event is $\underline{e}$ and the other is $\underline{f}$, there exists a cluster that contains F but does not contain E. Namely the cluster $\{F\}$. Therefore if i is a descendant of f then $\mathcal{T}_e \ncong \mathcal{T}_f$.
Now consider the case where $i = f$ (and $j = e$). Because $\delta(e, f) > 3$, there exists a

duplication event on the path between $\underline{e}$ and $\underline{f}$ that only involves one of e or f. Without loss of generality, we can assume that this event involves f. Let f' be the other edge involved in this duplication event. Subtrees of the trees $\mathcal{T}, \mathcal{T}_f$ and $\mathcal{T}_e$ can be seen in the Figure 5.22. The tree $\mathcal{T}_e$ will induce the cluster $\{E, F, F'\}$. If the $\mathcal{T}_e \cong \mathcal{T}_f$, then the tree

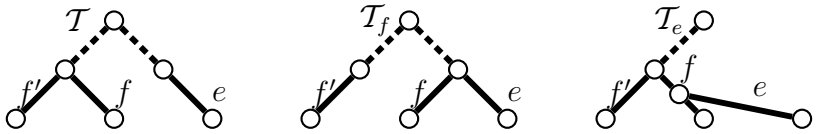


Figure 5.22: Subtrees of the trees $\mathcal{T}$, $\mathcal{T}_f$ and $\mathcal{T}_e$.

$\mathcal{T}_f$ must also induce this cluster. The tree $\mathcal{T}_f$ was made by pruning the tree with root $\underline{f}$ and regrafting it to e. So if $\mathcal{T}_f$ induces $\{E, F, F'\}$, so does $\mathcal{T}$. This can only happen if $\delta(\underline{e}, \underline{f}) = 3$. But we assumed that $\delta(\underline{e}, \underline{f}) > 3$. Hence, if $j = e$, $\mathcal{T}_e \ncong \mathcal{T}_f$.

Finally if $j \neq e$ (and $i \neq f$), then the tree $\mathcal{T}_f$ induces the cluster $\{J, F\}$ where $\{J\} \cap \{F\} = \emptyset$. If $\mathcal{T}_e$ also induces this cluster, then f is a descendant of j in $\mathcal{T}$. Because $\mathcal{T} \ncong \mathcal{T}_f$, $\delta(j, f) > 1$. The tree $\mathcal{T}_f$ also induces the clusters $\{J\}$ and $\{F\}$ (remember that $\{J\} \cap \{F\} = \emptyset$). If the tree $\mathcal{T}_e$ induces this cluster, then there is a duplication event in $\mathcal{T}_e$ such that one descendant is j and the other is f. To get this event without moving j or f we would need to prune and regraft all trees with roots incident with $\underline{w}$ where w is an edge on the path between $\underline{j}$ and $\underline{f}$. If $\delta(j, f) = n$ then there are $n - 1$ such trees. We can only prune and regraft one edge (the edge e). So this is only possible if $\delta(e, f) < 3$ and the edge e is incident with $\underline{w}$ for some edge w on the path between $\underline{j}$ and $\underline{f}$.

Note that if $\delta(j, f) = 1$, then $\mathcal{T}_f \cong \mathcal{T}$. Therefore this is only possible if $\delta(j, f) = 2$. But, if $\delta(j, f) = 2$ and e is incident with some vertex w on the path between $\underline{j}$ and $\underline{f}$, then $\delta(e, v) \leq 3$. This is a contradiction as we assumed that $\delta(e, f) > 3$. This implies that $\mathcal{T}_e \ncong \mathcal{T}_f$.

Therefore if $\delta(e, f) > 3$ or $\underline{e}$ is a descendant of $\underline{f}$ or $\underline{f}$ is a descendant of $\underline{e}$, then $\mathcal{T}_e \ncong \mathcal{T}_f$.

□

We have now covered the cases for all distinct pairs of edges e and f such that $\delta(\underline{e}, \underline{f}) > 3$, $\underline{e}$ is a descendant or ancestor of $\underline{f}$ and the edges are incident with a common vertex and $\underline{e}$ and $\underline{f}$ are part of the same duplication event. The only possible pairs of edges that do not fall into one of these cases are the pairs of edges considered in Lemma 5.10.

5.2.3 Simple DrSPR Neighbourhood Size

The next step is to count how many DrSPR moves each individual simple duplication tree has. Because we are working with simple duplication trees, we are able to break the tree up into smaller trees and work locally. This is because no subtree overlaps any other subtree unless they share common vertices. We will break up our simple duplication trees as follows.

Let $(\mathcal{T}, O)$ be a simple duplication tree, i and j be two consecutive elements in O and $\rho_{i,j} = \text{mrca}(i, j)$. Let t be the tree obtained by deleting all vertices and edges from $\mathcal{T}$ that are not on the path from i and j. We call t a *shrub* of $\mathcal{T}$ of size $(\delta(i, \rho_{i,j}), \delta(j, \rho_{i,j}))$.

As an example, suppose we have the simple duplication tree shown in Figure 5.23. The shrub made from the leaves labeled 1 and 2 is shown in Figure 5.24 and has size

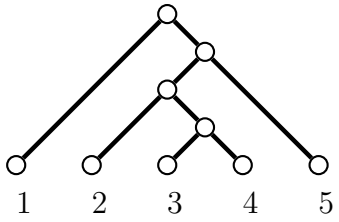


Figure 5.23: A duplication tree.

$(1, 3)$. As a convention, a shrub made from the leaves i and j will have size (I, J) and vice versa. Lemma 5.10 showed when DrSPR moves were not distinct on simple duplication trees. There are three possible moves that give the same simple duplication tree. Two of these occur in the same shrub, while the third occurs in a separate shrub. The following

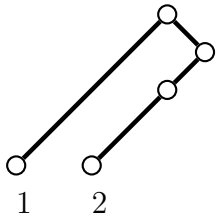


Figure 5.24: The shrub with leaves 1 and 2 from the tree in Figure 5.23.

lemma will be used to count the number of times this happens in a tree and will be used to identify the chosen distinct DrSPR move in this situation and which shrub it belongs to.

We will define an *interior edge* to be an edge that is not incident with a leaf.

Lemma 5.13. *Let $(\mathcal{T}, O)$ be a simple duplication tree and $|O| = n$. There are $n-2$ pairs of edges e and f such that $\delta(\underline{e}, \underline{f}) = 3$ and $\underline{f}$ is not a descendant or ancestor of $\underline{e}$ and that when t_e is pruned and regrafted to f to make the tree $\mathcal{T}'$, $\mathcal{T}'$ is still a simple duplication tree. Moreover, every interior edge is incident with exactly one such pair of edges*

Proof. Let g be an interior edge of the tree $\mathcal{T}$. Let g_1 and g_2 be the two vertices incident with g. Without loss of generality, we can assume that g_2 is a descendant of g_1. Because g is an interior vertex, the vertices g_1 and g_2 are not leaves. This implies that g_1 and g_2 are roots of duplication events. Suppose that the vertex g_2 is right descendant of g_1. Let e be the left descendant of g_1 and f be the left descendant of g_2. This is shown in Figure 5.25 below. Now, $\delta(\underline{e}, \underline{f}) = 3$ and e and f are not descendants of each other. If we

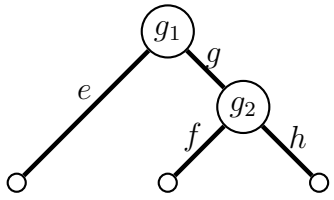


Figure 5.25: Subtree of a duplication tree with an internal edge g.

were to prune e or f and regraft it to f or e respectively, then the resulting tree would

be the same simple duplication tree. This shows that every interior edge is incident with a pair of edges e and f such that $\delta(\underline{e}, \underline{f}) = 3$, e and f are not descendants of each other and if we prune e or f and regraft it to f or e respectively, then the resulting tree is still a simple duplication tree.

The edge g is incident with up to four other edges. However, there are only three edges that are incident with vertices that are not descendants of each other. The only vertex distinct from g_1 incident with the edge k such that $\underline{k} = g_1$ is an ancestor of all vertices incident with g and hence does not satisfy the required conditions. Let e be the other edge in the duplication event that has root g_1 and f and h be the left and right edges respectively of the duplication event with root g_2. Again this is shown in Figure 5.25. If g was a right descendant of the duplication event with root g_1 then all the descendants of f will be between all the descendants of e and h. This implies that we can not get a simple duplication tree by pruning the tree with root h and regrafting it to e. Consider the edges f and h. We see that $\delta(f, h) = 2 \neq 3$. So if g is a right descendant of the duplication event then there is only one pair of edges e and f incident with g such that $\delta(e, f) = 3$. Similarly, if g_2 is a left descendant of the duplication event with root g_1 there is only one pair of edges e and f such that $\delta(\underline{e}, \underline{f}) = 3$, and pruning the tree t_e and regrafting it to f gives the same simple duplication tree as pruning t_f and regrafting it to e. Therefore every interior edge spans exactly one pair of such edges.

Every simple duplication tree with n leaves has $2(n-1)$ edges. Therefore there are $2(n-1) - n = 2n - 2 - n = n - 2$ interior edges. Every interior edge has exactly one pair of edges e and f such that $\underline{e}$ is not a descendant or ancestor of $\underline{f}$ and that $\delta(e, f) = 3$. Therefore there are $n - 2$ such pairs of edges in any simple duplication tree and every interior edge is incident with exactly one such pair of edges.

□

To use shrubs to calculate the size the simple DrSPR neighbourhood for a simple duplication tree, we will need to assign distinct DrSPR moves to each shrub. We will assign a distinct DrSPR move to a shrub if the move involves pruning an edge and regrafting it to another edge in the same shrub.

Lemma 5.14. *Suppose we have a shrub t, of size (I, J) induced by a simple duplication tree $(\mathcal{T}, O)$. Define $\xi(i, j)$ to be the number of distinct DrSPR moves this shrub adds to $\eta(\mathcal{T})$. Then*

$$\xi(i,j) = 2IJ + \sum_{k=1}^{I-3}(k) + \sum_{h=1}^{J-3}(h) - 2 - \psi(\rho_{i,j}),$$

where $\psi(\rho_{i,j})$ is the number of edges incident with $\rho_{i,j}$ in t that are interior edges in $\mathcal{T}$. Note that this will always be $0, 1$ or 2.

Proof. Let h be an edge on the path from i to $\rho_{i,j}$ that is not incident with $\rho_{i,j}$. The number of edges that h can be moved to that creates a simple duplication tree $\mathcal{T}' \ncong \mathcal{T}$ is all the edges that are ancestors of h in the shrub except the edge that is incident with $\overline{h}$ plus all edges on the path from j to $\rho_{i,j}$. This gives $\sum_{k=1}^{I-2}(k) + (I-1)J$ moves for the edges on the path from i to $\rho_{i,j}$ not incident with $\rho_{i,j}$. The edge on the path from i to $\rho_{i,j}$ incident with $\rho_{i,j}$ can not be moved to the edge on the path from j to $\rho_{i,j}$ incident with $\rho_{i,j}$ because this would not change $\mathcal{T}$. However, this edge can be moved to all other edges on the path from j to $\rho_{i,j}$. So for the edges on the path from i to $\rho_{i,j}$ we have a total of

$$\sum_{k=1}^{I-2}(k) + (I-1)J + (J-1) = \sum_{k=1}^{I-2}(k) + IJ - 1$$

moves. Doing the same for the edges on the path from j to $\rho_{i,j}$ gives a total of

$$2IJ + \sum_{k=1}^{I-2}(k) + \sum_{h=1}^{J-2}(h) - 2$$

moves that can be performed within the shrub.

From Lemma 5.10 we know that if we prune a tree by cutting an edge e and regraft it to another edge f such that $\underline{f}$ is an ancestor of $\underline{e}$ and $\delta(\underline{e}, \underline{f}) = 2$, then we do not get a distinct simple duplication tree. We can obtain the same tree by moving an edge from one side of an adjacent shrub to the other. There will be $I-2$ edges e with this property on the path from i to $\rho_{i,j}$ and $J-2$ on the path from j to $\rho_{i,j}$. Subtracting these from the above equation gives

$$2IJ + \sum_{k=1}^{I-2}(k) + \sum_{h=1}^{J-2}(h) - 2 - (I-2) - (J-2) = 2IJ + \sum_{k=1}^{I-3}(k) + \sum_{h=1}^{J-3}(h) - 2.$$

Using Lemma 5.13 we see that every interior edge uniquely determines a pair of edges that satisfy Lemma 5.10 and these are the only possible pairs of edges that satisfy Lemma 5.10. If we assign all interior edges to the shrub whose root is incident with them, then every interior edge is assigned to exactly one shrub. Every interior edge corresponds to two moves in the same shrub that give the same result. There are $\psi(\rho_{i,j})$ such edges in each shrub. Thus we need to remove $\psi(\rho_{i,j})$ from the number of possible DrSPR moves in each shrub. From Lemmas 5.11 and 5.12 we know that all other moves we can make in this shrub are distinct. Therefore there are

$$\xi(i,j) = 2IJ + \sum_{k=1}^{I-3}(k) + \sum_{h=1}^{J-3}(h) - 2 - \psi(\rho_{i,j})$$

distinct moves that can be assigned to a shrub of size (I, J). □

We are now in a position to calculate the size of the DrSPR neighbourhood for an arbitrary simple duplication tree. That is, given a simple duplication tree, find the number os simple duplication trees that are one DrSPR move away. This is done using Theorem 5.15

Theorem 5.15. *Let $(\mathcal{T}, O)$ be a simple duplication tree with label set $\{1, \ldots, n\}$.*

$$\eta(\mathcal{T}) = \sum_{i=1}^{n-1}(\xi(i,i+1)) + \sum_{j=1}^{\delta(1,\rho)-3}(j) + \sum_{k=1}^{\delta(n,\rho)-3}(k).$$

Proof. Successively calculating $\xi(i, i+1)$ gives the number of distinct moves contributed by all the shrubs of $\mathcal{T}$. Using Lemma 5.10, we know that if we take any edge e on the left circumference and prune and regraft it to another edge on the left circumference such that $\delta(\underline{e}, \underline{f}) = 2$, then we do not get a distinct simple duplication tree. However if $\delta(\underline{e}, \underline{f}) > 2$, then we do get a distinct simple duplication tree, an equivalent move will be counted in another shrub. The edge incident with 1 has $\delta(1, \rho) - 3$ edges that it can be pruned and regrafted to that create a distinct simple duplication tree. The next edge on the left circumference has $\delta(1, \rho) - 3 - 1$ such edges. Continuing all the way up the path from 1 to ρ gives $\sum_{j=1}^{\delta(1,\rho)-3}(j)$ such moves. Similarly, there are $\sum_{k=1}^{\delta(n,\rho)-3}(k)$ moves for the edges on the right circumference. This gives

$$\eta(\mathcal{T}) = \sum_{i=1}^{n-1}(\xi(i,i+1)) + \sum_{j=1}^{\delta(1,\rho)-3}(j) + \sum_{k=1}^{\delta(n,\rho)-3}(k).$$

□

Because a shrub with leaves i and j has size (I, J), we can interchange i and j with I and J in $\xi(i, j)$ whenever it simplifies things. That is $\xi(i, j) = \xi(I, J)$.

An example of how we can use shrubs to determine the size of the DrSPR neighbourhood of a simple duplication tree is shown below.

Suppose we want to find the simple DrSPR neighbourhood of the tree $\mathcal{T}$ (left in Figure 5.26). We can split it up into the following shrubs (Figure 5.26 right). When this

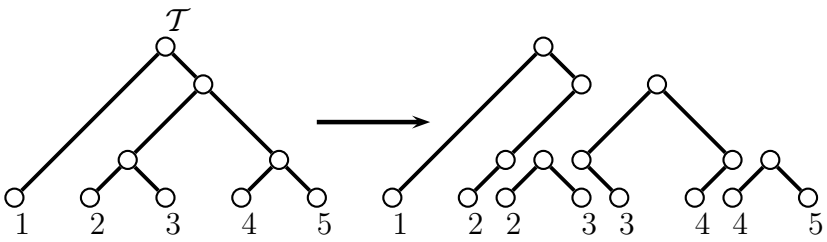


Figure 5.26: Splitting a simple duplication tree up into shrubs.

is done we can calculate the score for each shrub and add them together. This is shown with the scores of each shrub in the circles in Figure 5.27. When we add the totals up we see that $\eta(\mathcal{T}) = 0 + 0 + 3 + 4 + 0 + 0 = 7$.

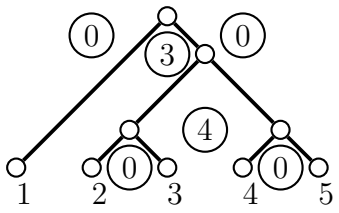


Figure 5.27: The number of distinct DrSPR moves of each shrub displayed on the tree from Figure 5.26.

Chapter 6

Extremal Cases

The size of the DrSPR neighbourhood depends on the shape of the tree. Therefore it is of interest what shapes give the largest and smallest sized neighbourhood. Finding these extremal cases will give us tight upper and lower bounds on $\eta(\mathcal{T})$. This will be useful as we can compare the upper bound with the number of simple duplication trees on n leaves. If the size of the neighbourhood of a tree on n leaves is too large, then hill climbing methods will be no better than an exhaustive search of all simple duplication trees on n leaves. This chapter will be concerned with finding these extremal cases and developing upper and lower bounds. We will only consider simple duplication trees on $n \geq 4$ leaves because all simple duplication trees on 2 or 3 leaves have the same size neighbourhoods.

This chapter is broken up into four sections. In the first section we will derive formulas that calculate what happens to the size of the DrSPR neighbourhood of a simple duplication tree if we perturb the tree slightly. Section 6.2 will give the size of the DrSPR neighbourhood of a few simple duplication tree shapes. These simple duplication tree shapes will be used in Section 6.3 to find the simple duplication tree shape that maximizes the size of the DrSPR neighbourhood and in Section 6.4 to find the simple duplication tree shapes that minimize the size of the DrSPR neighbourhood. This will provide us with upper and lower bounds for the size of the DrSPR neighbourhood of a simple duplication tree.

6.1 Perturbing the DrSPR Neighbourhood

The lemmas presented in this section will be used to prove the extremal cases in Sections 6.3 and 6.4. They all show what happens to $\xi(i, j)$ if we perturb a shrub slightly. These will be used to prove the extremal cases.

Lemma 6.1. *Suppose we have a shrub of size (I, J). If we increase the length of J by one (by inserting a vertex on the path from j to $\rho_{i,j}$) then, $\xi(i,j)$ increases by $2I + \max(J-2,-1)$. Furthermore, if we reduce the length of J by one then $\xi(i,j)$ is decreased by $2I + \max(J-3,-1)$.*

Proof. To begin with consider the equation $\xi(i,j)' - \xi(i,j)$, where $\xi(i,j)'$ is obtained by increasing the length of j by one. This is equal to

$$\begin{aligned}
\xi(i,j)' - \xi(i,j) &= 2I(J+1) + \sum_{k=1}^{I-3}(k) + \sum_{h=1}^{J-2}(h) - 2 - \psi(\rho_{i,j})' \\
&\quad - (2IJ + \sum_{k=1}^{I-3}(k) + \sum_{h=1}^{J-3}(h) - 2 - \psi(\rho_{i,j})) \\
&= 2I(J+1) - 2IJ + \sum_{k=1}^{I-3}(k) - \sum_{k=1}^{I-3}(k) + \sum_{h=1}^{J-2}(h) \\
&\quad - \sum_{h=1}^{J-3}(h) - 2 + 2 - \psi(\rho_{i,j})' + \psi(\rho_{i,j}) \\
&= 2I + \sum_{h=1}^{J-2}(h) - \sum_{h=1}^{J-3}(h) - \psi(\rho_{i,j})' + \psi(\rho_{i,j}).
\end{aligned}$$

If $J = 1$, then when we increase the length of J, it will no longer be a leaf. This will increase the number of interior edges incident with $\rho_{i,j}$ and therefore $-\psi(\rho_{i,j})' + \psi(\rho_{i,j}) = -1 = J - 2$ and

$$\sum_{h=1}^{J-2}(h) - \sum_{h=1}^{J-3}(h) = 0.$$

This gives $2I - 1 = 2I + \max(J-2,-1)$.

Now suppose $J > 1$. Because the number of interior edges incident with $\rho_{i,j}$ does not change, $-\psi(\rho_{i,j})' + \psi(\rho_{i,j}) = 0$ and

$$\sum_{h=1}^{J-2}(h) - \sum_{h=1}^{J-3}(h) = \max(J-2,0).$$

Because $J > 1$, we know that $J - 2 > -1$ and therefore $2I + \max(J-2,0) = 2I + \max(J-2,-1)$. This shows that $\xi(i,j)' - \xi(i,j) = 2I + \max(J-2,-1)$ and therefore if

we increase the length of J by one, $\xi(I, J)$ increases by $2I + \max(J-2, -1)$.

Now suppose $\xi(i,j)'$ is obtained by decreasing the length of J by one. Once again we consider the equation $\xi(i,j)' - \xi(i,j)$.

$$\begin{aligned}
\xi(i,j)' - \xi(i,j) &= 2I(J-1) + \sum_{k=1}^{I-3}(k) + \sum_{h=1}^{J-4}(h) - 2 - \psi(\rho_{i,j})' - \\
&\quad (2IJ + \sum_{k=1}^{I-3}(k) + \sum_{h=1}^{J-3}(h) - 2 - \psi(\rho_{i,j})) \\
&= 2I(J-1) - 2IJ + \sum_{k=1}^{I-3}(k) - \sum_{k=1}^{I-3}(k) + \sum_{h=1}^{J-4}(h) \\
&\quad - \sum_{h=1}^{J-3}(h) - 2 + 2 - \psi(\rho_{i,j})' + \psi(\rho_{i,j}) \\
&= -2I + \sum_{h=1}^{J-4}(h) - \sum_{h=1}^{J-3}(h) - \psi(\rho_{i,j})' + \psi(\rho_{i,j}).
\end{aligned}$$

Clearly J can not be one or else we would have a shrub with a side of length zero. If $J = 2$ then $-\psi(\rho_{i,j})' + \psi(\rho_{i,j}) = 1$ and $\sum_{h=1}^{J-4}(h) - \sum_{h=1}^{J-3}(h) = 0$. This would give

$$\xi(i,j)' - \xi(i,j) = -2I + 1 = -(2I + \max(J-3, -1))$$

and therefore $\xi(i,j)$ decreases by $2I + \max(J-3, -1)$.

Now, if $J > 2$, then $-\psi(\rho_{i,j})' + \psi(\rho_{i,j}) = 0$ and

$$\sum_{h=1}^{J-4}(h) - \sum_{h=1}^{J-3}(h) = -\max(J-3, 0).$$

Because $J > 2$, $J - 3 \geq 0$ and therefore $\max(J-3, 0) = \max(J-3, -1)$. Therefore, if we decrease J by one, then $\xi(i,j)$ decreases by $2I + \max(J-3, -1)$. □

Lemma 6.2. *Let $\mathcal{T}$ be a simple duplication tree on n leaves. Suppose that $\delta(n, \rho) = I$. If we increase this by one by adding another vertex/edge to this path, then the number of distinct DrSPR moves this path induces increases by* $\max(I-2, 0)$. *If we decrease this by one, then the number of distinct DrSPR moves this path induces is decreased by* $\max(I-3, 0)$.

Proof. The number of distinct DrSPR moves this path includes is $\sum_{k=1}^{I-3}(k)$. If we increase I by one then the sum increases by $I+1-3 = I-2$ if $I-2 > 0$ and by 0 if $I-2 \leq 0$. This is equal to $\max(I-2, 0)$.

Now if we decrease I by one then the sum decreases by either $I-3$ if $I-3 > 0$ and 0 if $I-3 \leq 0$. Therefore if we decrease I by one, the number of distinct DrSPR moves induced by this path is reduced by $\max(I-3, 0)$. □

Lemma 6.3. *Suppose that we have a shrub of size (I, J). If $I \geq J+1 \geq 2$ then $\xi(I-1, J+1) \geq \xi(I, J)$.*

Proof. Let $\xi(i,j)' = \xi(I-1, J+1)$. Consider the equation $\xi(i,j)' - \xi(i,j)$.

$$\begin{aligned}
\xi(i,j)' - \xi(i,j) &= 2(I-1)(J+1) + \sum_{k=1}^{I-4}(k) + \sum_{h=1}^{J-2}(h) - 2 - \psi(\rho_{i,j})' \\
&- (2IJ + \sum_{k=1}^{I-3}(k) + \sum_{h=1}^{J-3}(h) - 2 - \psi(\rho_{i,j})) \\
&= 2(I-1)(J+1) - 2IJ + \sum_{k=1}^{I-4}(k) - \sum_{k=1}^{I-3}(k) \\
&+ \sum_{h=1}^{J-2}(h) - \sum_{h=1}^{J-3}(h) - \psi(\rho_{i,j})' + \psi(\rho_{i,j}) \\
&= 2(I-J-1) - \max(I-3, 0) + \max(J-2, 0) - \psi(\rho_{i,j})' + \psi(\rho_{i,j}) \\
&= \underbrace{I - \max(I-3, 0) + \max(J-2, 0) - J}_{a} + \underbrace{I-J-2}_{b} - \underbrace{\psi(\rho_{i,j})' + \psi(\rho_{i,j})}_{c}
\end{aligned}$$

If $I = 2$, then $J = 1$, (because $I \geq J+1 \geq 2$). In this case, $\psi(\rho_{i,j}) - \psi(\rho_{i,j})' = 0$ and $a = 1$, $b = -1$ and $c = 0$. Therefore $a+b+c \geq 0$. Assume $I \geq 3$. If $\psi(\rho_{i,j})' > \psi(\rho_{i,j})$, then because $I-1 > 2$, $J = 1$. If this is the case, then $a \geq 2$, $b \geq 0$ and $c \geq -1$. From this we obtain $a+b+c \geq 1 > 0$. Finally, if $\psi(\rho_{i,j})' = \psi(\rho_{i,j})$ and $I \geq 3$ then $a \geq 1, b \geq -1$ and $c = 0$. This gives $a+b+c \geq 0$. Therefore $\xi(i,j)' \geq \xi(i,j)$ and therefore if $I \geq J+1 \geq 2$, then $\xi(I-1, J+1) \geq \xi(I, J)$. □

Lemma 6.4. *Suppose we have two shrubs with sizes (I, J) and (K, L) respectively, where $J - 1 \leq I \leq J$, $L - 1 \leq K \leq L$, and $2 \leq J \leq K$. Then*

$$\xi(I, J) + \xi(K, L) \leq \xi(I, J - 1) + \xi(K + 1, L).$$

Proof. Let $\xi(I, J)' = \xi(I, J - 1)$ and $\xi(K, L)' = \xi(K + 1, L)$. Consider the expression $\xi(I, J) + \xi(K, L) - \xi(I, J)' - \xi(K, L)'$. Using the formula from Lemma 5.14 yields

$$\begin{aligned}
&\xi(I, J) + \xi(K, L) - \xi(I, J - 1) - \xi(K + 1, L) \\
&\quad = 2IJ + \sum_{g=1}^{I-3}(g) + \sum_{h=1}^{J-3}(h) - 2 - \psi(\rho_{i,j}) \\
&\quad + 2KL + \sum_{g=1}^{K-3}(g) + \sum_{h=1}^{L-3}(h) - 2 - \psi(\rho_{k,l}) \\
&\quad - (2I(J-1) + \sum_{g=1}^{I-3}(g) + \sum_{h=1}^{J-4}(h) - 2 - \psi(\rho_{i,j})') \\
&\quad - (2(K+1)L + \sum_{g=1}^{K-2}(g) + \sum_{h=1}^{L-3}(h) - 2 - \psi(\rho_{k,l})') \\
&\quad \leq 2I - 2L + \max(J - 3, 0) - \max(K - 2, 0)
\end{aligned}$$

If $J = 2$, then $\psi(\rho_{i,j})' = \psi(\rho_{i,j}) - 1$. Otherwise $\psi(\rho_{i,j})' = \psi(\rho_{i,j})$. This is the reason for the inequality above. Note that because $K \geq 2$ there is already an interior edge incident with $\rho_{k,l}$ on the side with size K and so $\psi(\rho_{k,l})' = \psi(\rho_{k,l})$. As $L \geq I$ and $K - 2 \geq J - 3$ we have

$$\xi(I, J) + \xi(K, L) - \xi(I, J-1) - \xi(K+1, L) \leq 2I - 2L + \max(J-3, 0) - \max(K-2, 0) \leq 0$$

and therefore

$$\xi(I, J) + \xi(K, L) \leq \xi(I, J - 1) + \xi(K + 1, L).$$

□

Recall that for any real number $\lceil n \rceil$ denotes the smallest integer z such that $z \geq n$ and that $\lfloor n \rfloor$ denotes the largest integer z such that $z \leq n$.

Lemma 6.3 shows that a shrub has maximum score when balanced and the smallest when it has size $(1, n-1)$ where $I+J=n$. Furthermore, Lemma 6.4 shows that the total score of two balanced shrubs is maximized when one is made as large as possible. Using this, we see that if $I+J+K+L=m$, then $\xi(I,J)+\xi(K,L)$ is minimized when

$$I=1,\ J=\lfloor m/2 \rfloor - 1,\ K=1 \text{ and } L=\lceil m/2 \rceil - 1$$

and maximized when

$$I=J=1,\ K=\lfloor m/2 \rfloor - 1 \text{ and } L=\lceil m/2 \rceil - 1.$$

Given a simple duplication tree $\mathcal{T}$ and a shrub of this tree, it may be useful to know how many edges in the tree are not incident with this shrub. This will be used in the Section 6.3 when we find the simple duplication tree shape that maximizes $\eta(\mathcal{T})$. To do this, we will use Lemma 6.5.

Lemma 6.5. *Let $\mathcal{T}$ be a duplication tree with n leaves. Define the sum of a shrub with size (I,J) to be $I+J$. The sum of the sizes of all shrubs plus $\delta(1,\rho)+\delta(n,\rho)$ is $4(n-1)$. That is*

$$\sum_{i=1}^{n-1}(\delta(i,\rho_{i,i+1})+\delta(i+1,\rho_{i,i+1}))+\delta(1,\rho)+\delta(n,\rho)=4(n-1).$$

Proof. A tree $\mathcal{T}$ on n leaves has $2(n-1)$ edges. Every edge in $\mathcal{T}$ will either add one to the length of the side of two shrubs or add one to the length of the side of one shrub and add one to the length of the circumference. That is, every edge in $\mathcal{T}$ gets counted exactly twice in the above sum and therefore the sum equals $2 \cdot 2(n-1) = 4(n-1)$.

□

6.2 Simple Duplication Tree Shapes

Before we find the extremal cases, it will be beneficial to define a few simple duplication tree shapes and the size of their neighbourhoods. These tree shapes will be used in the next section when we are defining the extremal cases.

A simple duplication tree $\mathcal{T}$ with the form in Figure 6.1 is called a *caterpillar*. We

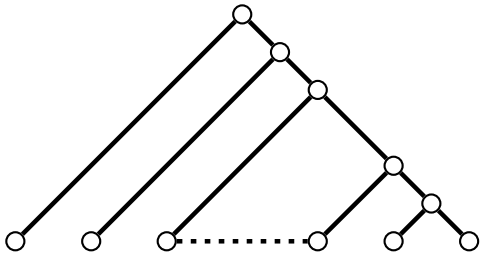


Figure 6.1: A caterpillar.

define a *double caterpillar* to be a simple duplication tree with the shape shown in Figure 6.2. Furthermore, we say that $\mathcal{T}$ is a *balanced double caterpillar* if $|m - n| \leq 1$.

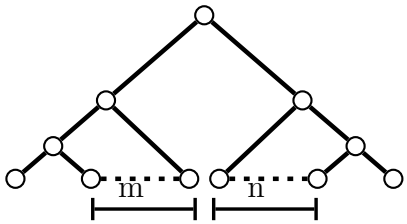


Figure 6.2: A double caterpillar.

Alternatively we call a simple duplication tree an *inverse (balanced) double caterpillar* if it has the shape shown in Figure 6.3.

It should be noted that without the ordering on the leaves that comes with duplication trees, a double caterpillar and an inverse double caterpillar are the same. All of these caterpillars have simple formulas for the size of their neighbourhoods. These formulas are given in Lemmas 6.6-6.8.

Lemma 6.6. *Let $\mathcal{T}$ be a caterpillar on n leaves. If $n > 2$ then*

$$\eta(\mathcal{T}) = \sum_{i=1}^{n-3}(i) + 1.$$

Proof. All simple duplication trees on n leaves contain $n - 1$ separate shrubs. Any caterpillar will have $n - 2$ shrubs of size $(1, 2)$ and one of size $(1, 1)$. Using the formula from

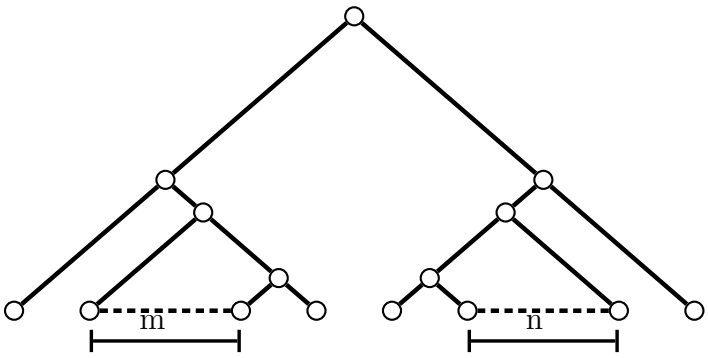


Figure 6.3: A inverse double caterpillar.

Lemma 5.14, we see that a shrub of size $(1, 1)$ has zero possible moves and a shrub of size $1, 2$ has one move. This gives $n - 2$ moves for the shrubs.

Without loss of generality we will assume that $\mathcal{T}$ is a right caterpillar, that is a caterpillar such that the left circumference has length 1 and the right circumference has length $n-1$. There is only one edge on the left circumference of $\mathcal{T}$, so there are no possible moves for this edge. Now consider the right circumference of $\mathcal{T}$. There will be $n - 1$ edges in this path. If we prune an edge e and regraft it to another edge f on this path, we get a distinct simple duplication tree only if $\delta(e, f) > 2$. Pruning t_e to an edge f on the same circumference such that $\delta(e, f) = 1$ will not change the tree and from Lemma 5.10, if $\delta(e, f) = 2$, then we will get the same tree by pruning and regrafting an edge in a shrub. This move is assigned to the shrub so pruning t_e to f will not give a distinct DrSPR move. However, if $\delta(e, f) > 2$ then this is a distinct DrSPR move from Lemma 5.11. For the first edge on the right circumference there will be $(n - 1) - 3 = n - 4$ edges f such that $\delta(e, f) > 2$. The next edge on the right circumference will have $n - 5$ such edges. Doing this for all edges on the right circumference gives $\sum_{i=1}^{n-4}(i)$ moves that can be made with edges on the right circumference. This gives

$$\eta(\mathcal{T}) = (n - 2) + \sum_{i=1}^{n-4}(i) = (n - 3) + 1 + \sum_{i=1}^{n-4}(i) = \sum_{i=1}^{n-3}(i) + 1$$

if $\mathcal{T}$ is a caterpillar on n leaves. □

Lemma 6.7. *Let $\mathcal{T}$ be a double caterpillar on $n \geq 4$ leaves. Then*

$$\eta(\mathcal{T}) = n + \sum_{i=1}^{r-3}(i) + \sum_{j=1}^{s-3}(j),$$

where $r = \delta(1, \rho)$ *and* $s = \delta(n, \rho)$*. Moreover, if* $\mathcal{T}$ *is a balanced double caterpillar then,*

$$\eta(\mathcal{T}) = n + \sum_{i=1}^{\lfloor n/2 \rfloor - 3}(i) + \sum_{j=1}^{\lceil n/2 \rceil - 3}(j).$$

Proof. Using a similar argument as Lemma 6.6, we deduce that the left circumference of a double caterpillar will induce

$$\sum_{i=1}^{r-3}(i)$$

distinct DrSPR moves, while the right circumference will induce

$$\sum_{j=1}^{s-3}(j)$$

distinct DrSPR moves. $\mathcal{T}$ will have $n-1$ shrubs, two will have size $(1,1)$, one will have size $(2,2)$ and the remaining $n-4$ will have size $(1,2)$. The two shrubs of size $(1,1)$ will add nothing to the size of the neighbourhood, the shrub of size $(2,2)$ will add 4 and the shrubs of size $(1,2)$ will each add one to the neighbourhood. Therefore if $\mathcal{T}$ is a double caterpillar, then

$$\eta(\mathcal{T}) = \ n - 4 + 4 + \sum_{i=1}^{r-3}(i) + \sum_{j=1}^{s-3}(j) = \ n + \sum_{i=1}^{r-3}(i) + \sum_{j=1}^{s-3}(j).$$

If $\mathcal{T}$ is a balanced double caterpillar then $r = \lfloor n/2 \rfloor$ and $s = \lceil n/2 \rceil$ or vice versa. Therefore if $\mathcal{T}$ is a balanced double caterpillar, then

$$\eta(\mathcal{T}) = n + \sum_{i=1}^{\lfloor n/2 \rfloor - 3}(i) + \sum_{j=1}^{\lceil n/2 \rceil - 3}(j).$$

□

Lemma 6.8. *Suppose that* $\mathcal{T}$ *is an inverse double caterpillar on* $n \geq 4$ *leaves. Then*

$$\eta(\mathcal{T}) = 2rs + \sum_{i=1}^{r-3}(i) + \sum_{j=1}^{s-3}(j) + n - 8,$$

where the shrub that contains the root of $\mathcal{T}$ has size (r, s).

Furthermore, if $\mathcal{T}$ is a balanced inverse double caterpillar then

$$\eta(\mathcal{T}) = 2\lfloor n/2 \rfloor \cdot \lceil n/2 \rceil + \sum_{i=1}^{\lfloor n/2 \rfloor - 3} (i) + \sum_{j=1}^{\lceil n/2 \rceil - 3} (j) + n - 8.$$

Proof. The shrub that contains the root of $\mathcal{T}$ adds

$$2rs + \sum_{i=1}^{r-3}(i) + \sum_{j=1}^{s-3}(j) - 4$$

distinct DrSPR moves to $\mathcal{T}$ (Lemma 5.14). There will be $n-2$ other shrubs in $\mathcal{T}$. Two of these will have size $(1,1)$ and will therefore add nothing to $\eta(\mathcal{T})$. The remaining $n-4$ shrubs will all have size $(1,2)$ or $(2,1)$. These will all add one to $\eta(\mathcal{T})$. Because $\delta(1,\rho) = 2 = \delta(n,\rho)$ in an inverse double caterpillar, the circumference of $\mathcal{T}$ adds no distinct DrSPR moves to $\mathcal{T}$. This gives

$$\eta(\mathcal{T}) = 2rs + \sum_{i=1}^{r-3}(i) + \sum_{j=1}^{s-3}(j) - 4 + n - 4 = 2rs + \sum_{i=1}^{r-3}(i) + \sum_{j=1}^{s-3}(j) + n - 8,$$

if $\mathcal{T}$ is an inverse double caterpillar. Furthermore, if $\mathcal{T}$ is a balanced inverse double caterpillar, then $r = \lfloor n/2 \rfloor$ and $s = \lceil n/2 \rceil$ or vice versa. In this case

$$\eta(\mathcal{T}) = 2\lfloor n/2 \rfloor \cdot \lceil n/2 \rceil + \sum_{i=1}^{\lfloor n/2 \rfloor - 3} (i) + \sum_{j=1}^{\lceil n/2 \rceil - 3} (j) + n - 8.$$

□

A *christmas tree* is a simple duplication tree with the shape shown in Figure 6.4.

Lemma 6.9. *Let $\mathcal{T}$ be a christmas tree on $n \geq 3$ leaves. Then, $\eta(\mathcal{T}) = 1 + 3(n-3)$.*

Proof. The circumference of a christmas tree induces no DrSPR moves. A tree $\mathcal{T}$ on n leaves has $n-1$ shrubs. If $\mathcal{T}$ is a christmas tree, then one of its shrubs has size $(1,1)$, another has size $(1,2)$ (or $(2,1)$) and all the other shrubs have size $(1,3)$. Calculating the score for each shrub we see that $\xi(1,1) = 0$, $\xi(1,2) = 1$ and $\xi(1,3) = 3$. This gives $0 + 1 + 3(n-3)$ and therefore $\eta(\mathcal{T}) = 1 + 3(n-3)$.

□

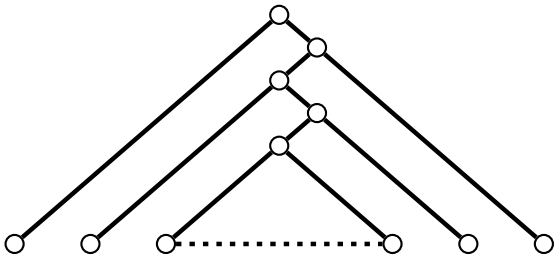


Figure 6.4: A christmas tree.

6.3 The Maximum Case

In this section we will find the simple duplication tree shape that maximizes $\eta(\mathcal{T})$. This case is easier that the minimum case because there is one distinct tree shape that maximizes $\eta(\mathcal{T})$. In Lemma 6.10, we will prove that the only simple duplication tree that has a shrub with maximum score is a balanced inverse double caterpillar. We will then show that if a tree has the maximum sized neighbourhood on n leaves, then that tree must have a shrub of maximum score.

Lemma 6.10. *The maximum score a shrub can have in a simple duplication tree on n leaves is when it has size $(\lceil n/2 \rceil, \lfloor n/2 \rfloor)$. Furthermore, the only duplication tree $\mathcal{T}$ on n leaves with a shrub of size $(\lceil n/2 \rceil, \lfloor n/2 \rfloor)$ is a balanced inverse double caterpillar.*

Proof. It follows from Lemma 6.3 that $\xi(i, j)$ of a shrub with size (I, J), where $I + J = m$ is maximum when $I = \lceil m/2 \rceil$ and $J = \lfloor m/2 \rfloor$.

Now suppose we have a shrub with size (I, J), where $I + J \geq n + 1$. Any shrub of a simple duplication tree shares at most one edge with any other shrub. Moreover, every edge of a shrub, except for an edge on the circumference is shared with another shrub. There is a maximum of one edge on the left circumference in each shrub and one edge on the right circumference in each shrub. Therefore if $I + J \geq n + 1$, then there must be at least $n + 1 - 2 = n - 1$ distinct shrubs in $\mathcal{T}$ that share an edge with any shrub of size (I, J) ,where $I + J \geq n + 1$. However a simple duplication tree on n leaves only has $n - 1$ shrubs, so any shrub can share an edge with no more than $n - 2$ other shrubs. This is a

contradiction. Therefore the maximum size a shrub can have is (I, J) where $I + J = n$ and the score of this shrub is maximized when $I = \lceil n/2 \rceil$ and $J = \lfloor n/2 \rfloor$

Any two shrubs share a maximum of one edge. The shrub of maximum size has a total of $\lceil n/2 \rceil + \lfloor n/2 \rfloor) = n$ edges. As any shrub can have no more than two edges on the circumference of $\mathcal{T}$, there must be $n-2$ shrubs that share an edge with the shrub of maximum size. As there are $n-2$ shrubs after the shrub of maximum size, every other shrub in $\mathcal{T}$ shares an edge with the shrub of maximum size.
From Lemma 6.5, we know that the total size of all shrubs in a tree plus the length of the circumference is $4(n-1)$. The edges in the shrub of maximum size are counted once for this shrub and once for the other shrub/circumference they are in. When we take these away from the total we get $4(n-1) - 2n = 2(n-2)$. Every edge is counted twice, so there are $n-2$ edges unaccounted for. If a side of a shrub has length $\lceil n/2 \rceil$, then it must be incident with $\lceil n/2 \rceil - 1$ edges not in the path. Similarly , if a path has length $\lfloor n/2 \rfloor$, it must be incident with $\lfloor n/2 \rfloor - 1$ edges not in the path. So there are $\lceil n/2 \rceil - 1 + \lfloor n/2 \rfloor - 1 = n - 2$ edges incident with the shrub. Therefore, all edges in the tree $\mathcal{T}$ are either in the shrub of size $(\lceil n/2 \rceil, \lfloor n/2 \rfloor)$ or incident with this shrub. The only tree with this property is an inverse double caterpillar. To be maximum, the shrub containing ρ must have size $(\lfloor n/2 \rfloor, \lceil n/2 \rceil)$ and is therefore a balanced inverse double caterpillar. □

Theorem 6.11. *Let $\mathcal{T}$ be a simple duplication tree on n leaves. If $\mathcal{T}$ is a balanced inverse double caterpillar, then $\eta(\mathcal{T}) \geq \eta(\mathcal{T}')$ for all trees $\mathcal{T}'$ on n leaves.*

Proof. First we will show that for a simple duplication tree $\mathcal{T}$ on n leaves, $\eta(\mathcal{T}) > \eta(\mathcal{T}')$ for all other simple duplication trees on n leaves if and only if $\mathcal{T}$ has a shrub with maximum possible score. Once this is done, Lemma 6.10 will imply that if $\mathcal{T}$ is a balanced inverse double caterpillar on n leaves, then $\eta(\mathcal{T}) > \eta(\mathcal{T}')$ for all $\mathcal{T}'$ with n leaves.

There are only 5 simple duplication trees on 4 leaves. It can be easily checked that if $\mathcal{T}$ is the tree with the shrub with greatest score, then $\eta(\mathcal{T}) > \eta(\mathcal{T}')$ for all other $\mathcal{T}'$ on 4 leaves. Now assume that this is true for trees on leaves $4, \ldots, n-1$. Let $\mathcal{T}$ and

$\mathcal{T}'$ be simple duplication trees on n leaves where $\mathcal{T}$ has a shrub of maximum score and $\mathcal{T}'$ does not and $\eta(\mathcal{T}) \leq \eta(\mathcal{T}')$. Note that this means $\mathcal{T}$ is a balanced inverse caterpillar (Lemma 6.10). So the shrub with maximum score in $\mathcal{T}$ shares an edge with two cherries, one on either side of the leaves of the shrub. If we delete the leaf from $\mathcal{T}$ that is on the side of the shrub with length $\lceil n/2 \rceil$ then $\eta(\mathcal{T})$ will decrease by $2\lfloor n/2 \rfloor + \max(\lceil n/2 \rceil - 3, -1) + 1$ with the $+1$ coming from a shrub of size $(1, 2)$ that became a cherry after the deletion and the rest being lost from the shrub with size $(\lceil n/2 \rceil, \lfloor n/2 \rfloor)$ using Lemma 6.1. Because $n > 4$, we know that $\lceil n/2 \rceil - 3 > -1$. Therefore we can remove the max() from the equation from Lemma 6.1 giving a reduction of

$$2\lfloor n/2 \rfloor + \lceil n/2 \rceil - 2.$$

We will call the tree created in this way $\mathcal{T}''$. By the induction hypothesis $\eta(\mathcal{T}'')$ is greater than all other trees on $n - 1$ leaves. Now consider the tree $\mathcal{T}'$. Let $\mathcal{T}'''$ be the tree obtained by deleting any leaf from $\mathcal{T}'$. If deleting any leaf reduces $\eta(\mathcal{T}')$ by less than $2\lfloor n/2 \rfloor + \lceil n/2 \rceil - 2$ then because $\eta(\mathcal{T}) \leq \eta(\mathcal{T}')$ we will have $\eta(\mathcal{T}'') \leq \eta(\mathcal{T}''')$. This is a contradiction as $\eta(\mathcal{T}'') \leq \eta(\mathcal{T}''')$ and both $\mathcal{T}''$ and $\mathcal{T}'''$ have $n-1$ leaves. So that means if we delete any edge of $\mathcal{T}'$, then $\eta(\mathcal{T}')$ needs to be reduced by more than $2\lfloor n/2 \rfloor + \lceil n/2 \rceil - 2$. We will show that this is not possible by showing that the if we delete the leaf labeled 1 or the leaf labeled n, then one of them will reduce $\eta(\mathcal{T}')$ by less than $2\lfloor n/2 \rfloor + \lceil n/2 \rceil - 2$. Note that if we add up the reductions from deleting the leaf labeled 1 and deleting the leaf labeled n and the total reduction is less than

$$2(2\lfloor n/2 \rfloor + \lceil n/2 \rceil - 2) = 2n + 2\lfloor n/2 \rfloor - 4,$$

then at least one of 1 and n will reduce $\eta(\mathcal{T}')$ by less than $2\lfloor n/2 \rfloor + \lceil n/2 \rceil - 2$. To begin with assume that both the leaves labeled 1 and n are in cherries. In this case, the three possible trees (up to symmetry which make no difference) are shown in Figure 6.5. It is possible for i to not exist if $n = 5$. If this is the case then I would be the length of the side of j in the shrub $(2, j)$ and J would be the length of the side of j in the shrub $(j, n-1)$.

Let $L = \delta(\rho, 1)$ and $R = \delta(\rho, n)$. If we delete the leaf 1, then the number of distinct DrSPR moves the left circumference has is reduced by $\max(L - 3, 0)$ (Lemma 6.2). The

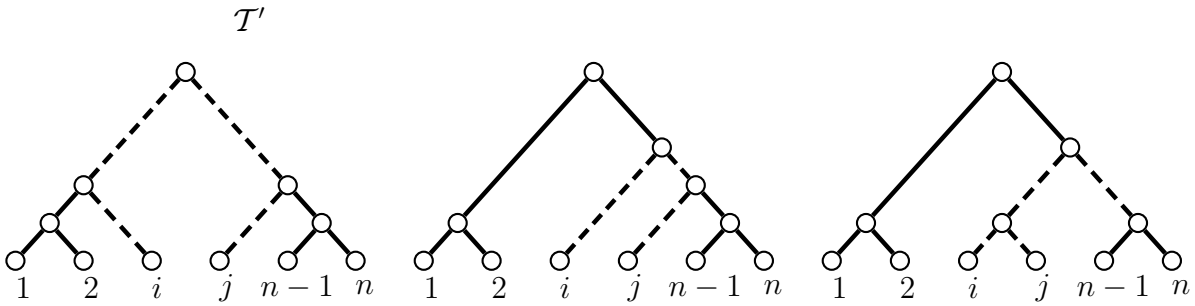


Figure 6.5: The three possible simple duplication trees (up to mirror images) that have both the leaf labeled 1 and n in cherries.

cherry $(1, 2)$ induces no DrSPR moves. The length of the leaf labeled 2 in the shrub $(2, i)$ is two. When we delete 1, the length of the leaf labeled 2 in the shrub will be reduced by one and so the shrub $(2, i)$ will lose

$$2I - \max(2 - 3, -1) = 2I - 1$$

moves (Lemma 6.1). Doing the same for the cherry $(n - 1, n)$ gives a total decrease of at most

$$\max(L - 3, 0) + \max(R - 3, 0) + 2I + 2J - 2.$$

Assume that

$$\max(L - 3, 0) = 0 = \max(R - 3, 0).$$

No path in a simple duplication tree on n leaves can be longer than $n - 1$. From this, we know that $I + J \leq n - 1$ because neither 1 nor n can add to this length and they can share a maximum of one edge. When we delete 1 and n we will decrease $\eta(\mathcal{T}')$ by

$$2I + 2J - 1 \leq 2n - 3 < 2n + 2\lfloor n/2 \rfloor - 4$$

and therefore we have obtained a contradiction. Now assume that $\max(L - 3, 0) > 0$. At most one of the paths I and J can share at most a single edge with either the path L or the path R. This can only happen if j is a left descendant of ρ or i is a right descendant of ρ. Clearly both of these can not happen because then they would cross over and we would not have a simple duplication tree. As $L + R \leq n$ and every edge that is in the path I or the path J but not in the path from ρ to 1 (with length (L) or the path from

ρ to n (with length (R) will reduce this, we know that

$$L + R \leq n - I - J + 4.$$

Because $n \geq 5$ we see that the total decrease is at most

$$\begin{aligned}
& \max(L-3,0) + \max(R-3,0) + 2I + 2J - 2 \\
\leq\ & L - 3 + R + 2I + 2J - 2 \text{ (because } \max(L-3,0) > 0) \\
=\ & L + R + 2I + 2J - 5 \\
\leq\ & n - I - J + 4 + 2I + 2J - 5 \text{ (because } L + R \leq n - I - J + 4) \\
=\ & n + I + J - 1 \\
\leq\ & 2n - 2 \text{ (because } I + J \leq n - 1) \\
<\ & 2n + 2\lfloor n/2 \rfloor - 4 \text{ (because } n \geq 5, \text{ and therefore } 2\lfloor n/2 \rfloor \geq 4)
\end{aligned}$$

and once again we have a contradiction.

Now assume that the shrub $(n-1, n)$ is not a cherry and first suppose that $R > 1$. The two possible options are shown in Figure 6.6. Note that in some cases the leaf labeled

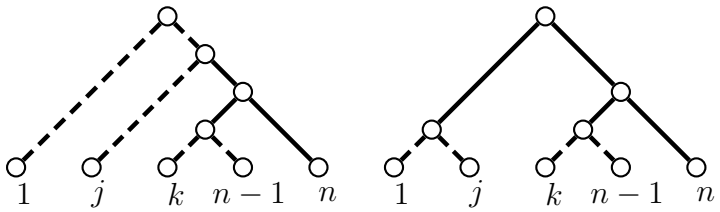


Figure 6.6: The two possible simple duplication trees such that the leaf labeled n is not in a cherry and $R > 1$.

j might also be 1. However, this makes no difference to the argument presented below. Let N' be the length of the side $n-1$ in the shrub $(n-1, n)$. When we delete the leaf n, in the shrub (j, k), K will decrease by one reducing $\eta(\mathcal{T}')$ by $2J + \max(K-3, -1)$. The shrub $(n-1, n)$ will no longer exist, reducing $\eta(\mathcal{T}')$ by

$$2N' + \sum_{n-1}^{N'-3}(n) - 2 - \psi(\rho_{n-1,n}) = 2N' + \sum_{n-1}^{N'-3}(n) - 3,$$

where N' is the length of the side $n-1$ in the shrub $(n-1,n)$. The length of R will increase by $N'-2$. This will increase $\eta(\mathcal{T}')$ by $\sum_{r=R-2}^{R+N'-5}(r)$. This gives a reduction of

$$2J + \max(K-3,-1) + 2N' + \sum_{n-1}^{N'-3}(n) - \sum_{r=R-2}^{R+N'-5}(r) - 3. \qquad (\star)$$

Note that because $R > 1$ we have

$$\sum_{r=R-2}^{R+N'-5}(r) \geq \sum_{n-1}^{N'-3}(n).$$

Therefore the equation $(\star)$ will be maximized when J is as large as possible. This will be when $N' = 2$ and $K = 3$. Let J_{max} be the maximum value of J when doing this. When this is done, $\eta(\mathcal{T}')$ will be reduced by

$$2J_{max} + 2N' - 3 = 2J_{max} + 1$$

when we delete n. Because the leaf k is not part of R or J, the maximum length of J is one less than if $(n-1,n)$ was a cherry. So if $(n-1,n)$ was a cherry, then $\eta(\mathcal{T}')$ could be reduced by

$$2(J_{max}+1) + \max(R-3,0).$$

Therefore $\eta(\mathcal{T}')$ is reduced less when we delete n if $(n-1,n)$ is not a cherry than if $(n-1,n)$ is a cherry. The same argument applies if $(1,2)$ is not a cherry and therefore at least one of 1 and n reduces $\eta(\mathcal{T}')$ by less than $2\lfloor n/2 \rfloor + \lceil n/2 \rceil - 2$.

Finally assume that $R = 1$. The two possible cases are shown in Figure 6.7 below. In

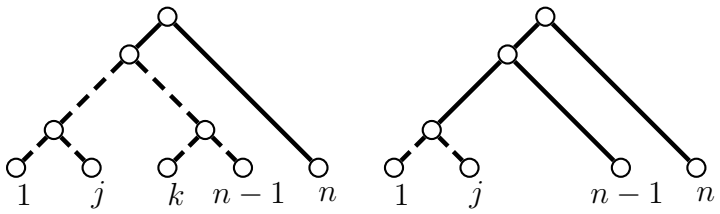


Figure 6.7: The two possible simple duplication trees such that the leaf labeled n is not in a cherry and $R = 1$.

the case on the right, when we delete n, $\eta(\mathcal{T})$ will only be reduced by one. This is clearly less than $2\lfloor n/2 \rfloor + \lceil n/2 \rceil - 2$ and hence a contradiction. So assume that the length of $n-1$ in the shrub $(n-1, n)$ denoted by N' is greater than one. This is shown on the left in Figure 6.7. We will handle this case slightly differently to the previous cases because if we delete 1 and n it is possible to reduce $\eta(\mathcal{T}')$ by more than $2\lfloor n/2 \rfloor + \lceil n/2 \rceil - 2$. When we delete n we will eliminate the shrub $(n-1, n)$. This will reduce $\eta(\mathcal{T})$ by

$$2N' + \sum_{n=1}^{N'-3}(n) - 2 - \psi(\rho) = 2N' + \sum_{n=1}^{N'-3}(n) - 3.$$

However after we delete n, the right circumference will have length $N'-1$ and therefore increase $\eta(\mathcal{T}')$ by $\sum_{n=1}^{N'-4}(n)$. This gives a total reduction of

$$2N' + \max(N'-3, 0) - 3.$$

If this is greater than $2\lfloor n/2 \rfloor + \lceil n/2 \rceil - 2$, then $N' \geq 4$ because $n \geq 5$ and therefore $2\lfloor n/2 \rfloor + \lceil n/2 \rceil - 2 \geq 5$. Therefore, because $N' \geq 4$, we have

$$2N' + N' - 3 - 3 \geq 2\lfloor n/2 \rfloor + \lceil n/2 \rceil - 2,$$

which implies that $3(N'-1) > 2\lfloor n/2 \rfloor + \lceil n/2 \rceil \geq 3\lfloor n/2 \rfloor$ and therefore $N' - 1 > \lfloor n/2 \rfloor$. This implies that $N' > \lceil n/2 \rceil$.

Now consider the shrub $(1, 2)$. Because $R = 1$, we know that $L > 1$. From above, we know that deleting 1 will reduce $\eta(\mathcal{T}')$ by the most if $(1, 2)$ is a cherry. If this is the case, then when we delete 1 we will reduce $\eta(\mathcal{T}')$ by $2I + \max(L-3, 0)$. If we fix the total length of $I + L$, then this reduction is maximum when I is as large as possible. Using a similar equation as in the first case when both 1 and n were in cherries we see that $R + L \leq n - I - N' + 4$. We know that $R = 1$ and therefore $L \leq n - I - N' + 3$. Every edge except for the leaf 2 can add length to either I or N' and they can share up to one edge. Therefore $I + N' \leq n$. We know that $N' > \lceil n/2 \rceil$, so $I \leq \lfloor n/2 \rfloor$. Because of this, when we delete 1, we will reduce $\eta(\mathcal{T}')$ by at most

$$2\lfloor n/2 \rfloor + \max(L-3, 0) - 1 = 2\lfloor n/2 \rfloor + \max(3-3, 0) - 1 = 2\lfloor n/2 \rfloor - 1 < 2\lfloor n/2 \rfloor + \lceil n/2 \rceil - 2.$$

Therefore we obtain a contradiction as we have just constructed a tree $\mathcal{T}'''$ such that $\eta(\mathcal{T}'') \leq \eta(\mathcal{T}''')$.

Now we have shown that if you take any tree $\mathcal{T}'$ with $n > 4$ leaves, then deleting one of the leaves 1 and n reduces $\eta(\mathcal{T}')$ by less than $2\lfloor n/2 \rfloor + \lceil n/2 \rceil - 2$. Therefore if there exists a tree $\mathcal{T}'$ such that $\eta(\mathcal{T}) \leq \eta(\mathcal{T}')$, then we can delete a leaf to gives us a tree $\mathcal{T}'''$ with $n-1$ leaves such that $\eta(\mathcal{T}'') \leq \eta(\mathcal{T}''')$. This is a contradiction as $\eta(\mathcal{T}'')$ is the maximum possible neighbourhood size on $n-1$ leaves. Therefore there can exist no tree $\mathcal{T}'$ with n leaves such that $\eta(\mathcal{T}) \geq \eta(\mathcal{T}')$. Therefore $\eta(\mathcal{T}) > \eta(\mathcal{T}')$ for all trees $\mathcal{T}'$ with n leaves if and only if $\mathcal{T}$ is a simple duplication tree on n leaves with a shrub of maximum possible score. From Lemma 6.10, we know that the only simple duplication tree with a shrub of maximum score is a balanced inverse double caterpillar. Therefore if $\mathcal{T}$ is a simple duplication tree on n leaves, then $\eta(\mathcal{T}) > \eta(\mathcal{T}')$ for all simple duplication trees on n leaves if and only if $\mathcal{T}$ is a balanced inverse double caterpillar.

□

6.4 The Minimum Case

Now that we have the shape that maximizes $\eta(\mathcal{T})$, we can move onto the tree shapes that give the minimum value for $\eta(\mathcal{T})$. Lemmas 6.12 and 6.14 give lower bounds on how $\eta(\mathcal{T})$ increases as the number of leaves of $\mathcal{T}$ increases. We will define $\eta(\mathcal{T}_n)$ to be the minimum value of $\eta(\mathcal{T})$ for all trees on n leaves.

Lemma 6.12. *If $\mathcal{T}$ is a simple duplication tree on $n \geq 3$ leaves such that $\eta(\mathcal{T})$ is minimum, then if we add another edge to this tree to create the tree $\mathcal{T}'$ on $n+1$ leaves, then $\eta(\mathcal{T}') > \eta(\mathcal{T})$.*

Proof. Assume that $\eta(\mathcal{T})$ is minimal and that $\eta(\mathcal{T}) \geq \eta(\mathcal{T}')$. Take the tree $\mathcal{T}'$. Every simple duplication tree has at least one shrub of size $(1,1)$. This shrub will correspond to the last duplication event. If we take this shrub and remove one of the edges to get $\mathcal{T}''$, we will remove one shrub and decrease the size of two other shrubs. This will decrease $\eta(\mathcal{T}')$, by Lemma 6.1 or 6.2 (depending on where in the tree the cherry is located). This would

give $\eta(\mathcal{T}'') < \eta(\mathcal{T}')$. But this implies $\eta(\mathcal{T}) \geq \eta(\mathcal{T}') > \eta(\mathcal{T}'')$. This is a contradiction as both $\mathcal{T}$ and $\mathcal{T}''$ have n leaves and therefore $\eta(\mathcal{T})$ can not be minimum. Therefore if $\eta(\mathcal{T})$ is minimum for all simple duplication trees of n leaves and $\mathcal{T}'$ has $n+1$ leaves, then $\eta(\mathcal{T}') > \eta(\mathcal{T})$.

□

Lemma 6.12 implies that if we have a tree with minimum sized neighbourhood on n leaves and add another leaf to it, then the size of its neighbourhood increases. This is not always the case when $\eta(\mathcal{T})$ is not minimal. For example, take the tree $\mathcal{T}$ shown in Figure 6.8 below and add the dashed leaf to make the tree $\mathcal{T}'$. Adding the extra leaf has

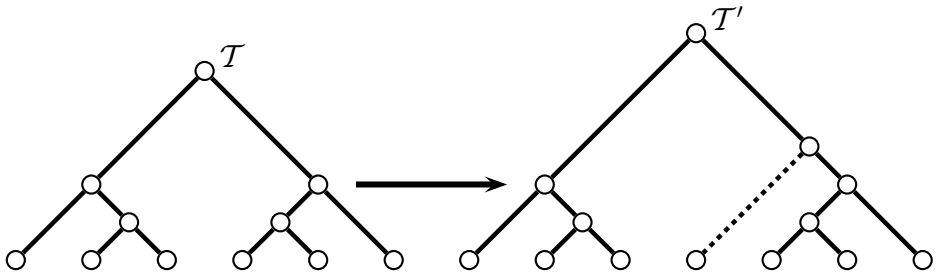


Figure 6.8: An example of when adding an extra leaf reduces the size of the DrSPR neighbourhood.

decreased the neighbourhood size, $\eta(\mathcal{T}) = 16$, while $\eta(\mathcal{T}') = 11$.

While Lemma 6.13 is intuitive, is needed for Lemma 6.14.

Lemma 6.13. *Let $(\mathcal{T}, O)$ be a duplication tree and $(\mathcal{T}_v, O_v)$ a pendant subtree of $(\mathcal{T}, O)$ with root v. Then $(\mathcal{T}_v, O_v)$ is a duplication tree.*

Proof. Assume that $(\mathcal{T}_v, O_v)$ is not a duplication tree. Then there exists a floor of $(\mathcal{T}_v, O_v)$ such that there can be no duplication reduction performed. We will call this floor floor i. Now consider $(\mathcal{T}, O)$. If we perform all possible duplication reductions, then we will reach a floor in $\mathcal{T}$ such that no visible duplication event can be reduced without reducing a cherry in floor i. When this happens, because no cherry in floor i of $(\mathcal{T}_v, O_v)$ can be reduced, there will be no visible duplication event. This would mean that no duplication

reduction can take place and hence $(\mathcal{T}, O)$ is not a duplication tree. This is a contradiction. Therefore every pendant subtree of a duplication tree is a duplication tree.

□

We will define an *interior edge* of a simple duplication tree $\mathcal{T}$ to be an edge that is not in the circumference of $\mathcal{T}$.

Lemma 6.14. *Let $\mathcal{T}$ be a simple duplication tree such that $\eta(\mathcal{T}) \leq \eta(\mathcal{T}')$ for all other simple duplication trees $\mathcal{T}'$ on n leaves. If we adjoin an edge to an interior edge of $\mathcal{T}$ to create the tree $\mathcal{T}''$, then $\eta(\mathcal{T}) + 3 \leq \eta(\mathcal{T}'')$.*

Proof. Take any pendant subtree of $\mathcal{T}''$ with root v. This subtree will be a simple duplication tree (Lemma 6.13). As every simple duplication tree contains at least one cherry, every vertex in a simple duplication tree has at least one cherry as a descendant.
Now, suppose $\eta(\mathcal{T}) + 3 > \eta(\mathcal{T}'')$. Consider the vertex created by adjoining an edge to create $\mathcal{T}''$. We will denote this vertex by u. Because the vertex u was created by adjoining an edge to an interior edge, u is not on the circumference of $\mathcal{T}''$. This implies that there is at least one cherry that is a descendant of u and is not on the circumference of $\mathcal{T}''$. Because of this, there are two shrubs that share the edge e that is incident with the root of this cherry. Let (I, J) be the size of the shrub to the left of e and (K, L) be the size of the shrub to the right of e that contains e. Without loss of generality, we can assume that $J = 2$ and $K \geq 3$. This is shown in Figure 6.9. When we delete j or k, $\eta(\mathcal{T}'')$ is

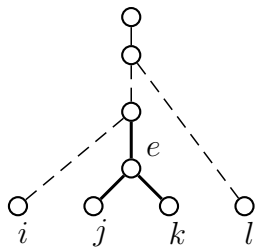


Figure 6.9: An interior cherry.

reduced by

$$(2I + \max(J-3,-1)) + (2L + \max(K-3,-1)) \geq (2-1) + (2+0) = 3,$$

by Lemma 6.1. If $\mathcal{T}'''$ is made by deleting j or k from $\mathcal{T}''$, then this gives a tree $\mathcal{T}'''$ on n leaves such that $\eta(\mathcal{T}''') < \eta(\mathcal{T})$. This is a contradiction as $\eta(\mathcal{T}) \leq (\mathcal{T}')$ for all simple duplication trees $\mathcal{T}'$ on n leaves. Therefore $\eta(\mathcal{T}) + 3 \leq \eta(\mathcal{T}'')$, if $\mathcal{T}''$ is made by adjoining an edge to an interior edge of $\mathcal{T}$. □

Theorem 6.15. *Let $\mathcal{T}$ be a simple duplication tree on n leaves such that $\eta(\mathcal{T}) \leq \eta(\mathcal{T}')$ for all $\mathcal{T}'$ on n leaves.*

- *If $\mathcal{T}$ is a caterpillar and $n = 4$ or 5, then $\eta(\mathcal{T}) \leq \eta(\mathcal{T}')$ for all $\mathcal{T}'$ on n leaves.*
- *If $n = 6, 7$ or 8 and $\mathcal{T}$ is a balanced double caterpillar then $\eta(\mathcal{T}) \leq \eta(\mathcal{T}')$ for all $\mathcal{T}'$ on n leaves.*
- *If $n > 8$ and $\mathcal{T}$ is a balanced double caterpillar with a christmas tree attached to the leaves 1 and/or n and such that $4 \leq \delta(\rho, 1) \leq 5$ and $4 \leq \delta(\rho, n) \leq 5$, then $\eta(\mathcal{T}) \leq \eta(\mathcal{T}')$ for all $\mathcal{T}'$ on n leaves.*

Proof. Unless $\delta(1,\rho) = 5$ and $\delta(n,\rho) = 5$ adjoining an edge to the circumference of any tree that does not have any edges not incident with the circumference adds at most 3 to $\eta(\mathcal{T})$. Therefore by Lemma 6.14, for $n = 2, \ldots, 10$, there exists a tree $\mathcal{T}$ such that $\eta(\mathcal{T})$ is minimized on n leaves if and only if $\mathcal{T}$ has no edge that is not incident with the circumference. Having edges on both sides of the circumference adds ≥ 3 to $\eta(\mathcal{T})$. Therefore if $n \leq 5$, $\eta(\mathcal{T})$ is minimized if and only if $\mathcal{T}$ is a caterpillar.

When $n = 6$, adding another edge to make a caterpillar on 6 leaves to a caterpillar on 5 leaves will increase $\eta(\mathcal{T})$ by 3. This will give us $\eta(\mathcal{T}) = 7$. However if we turn $\mathcal{T}$ into a balanced double caterpillar, then $\eta(\mathcal{T}) = 6$. Without loss of generality, there is only one possible unbalanced double caterpillar and this also has a neighbourhood of size 7. If there exists a tree $\mathcal{T}'$ on 6 leaves such that $\eta(\mathcal{T}') \leq \eta(\mathcal{T}) = 6$, then $\mathcal{T}'$ must have

an interior edge. However, using Lemma 6.14, we know that if this is so, we could delete an edge from a cherry that is a descendant of this edge to reduce $\eta(\mathcal{T}')$ by at least 3. However, $\eta(\mathcal{T}_5) = 4$ and $\eta(\mathcal{T}) = 6$. So if $\eta(\mathcal{T}') \leq 6$ then we can find a simple duplication tree $\mathcal{T}''$ on 5 leaves such that $\eta(\mathcal{T}'') \leq 6 - 3 = 3$. This is a contradiction as $\eta(\mathcal{T}_5) = 4$. Therefore if $\mathcal{T}$ is a simple duplication tree on 6 leaves, $\eta(\mathcal{T})$ is minimized if and only if $\mathcal{T}$ is a balanced double caterpillar.

A similar argument will show that if $n = 7$ or 8, then $\eta(\mathcal{T})$ is minimized if and only if $\mathcal{T}$ is a balanced double caterpillar.

Now suppose $n = 9$. Let $\mathcal{T}$ be the balanced double caterpillar shown below in Figure 6.10. Then $\eta(\mathcal{T}) = 13$. Assume that $\mathcal{T}'$ is a simple duplication tree on 9 leaves such

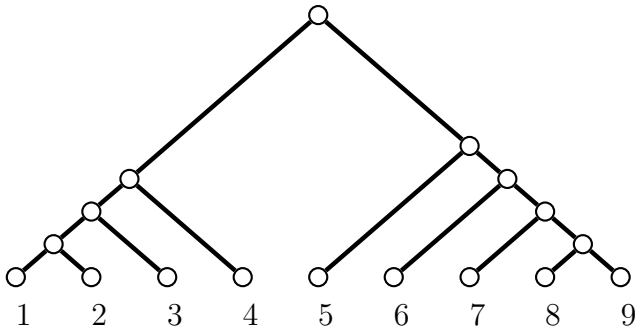


Figure 6.10: A balanced double caterpillar with nine leaves.

that $\eta(\mathcal{T}') \leq 12$. If $\delta(1, \rho) \geq 6$, then deleting the edge that is incident with 1 will reduce $\eta(\mathcal{T}')$ by at least 4 unless the shrub that contains 1 has size$(1, \geq 2)$. However, if the shrub that contains 1 has size $(1, x)$, where $x \geq 2$, then because every vertex is the ancestor of a cherry, there exists a cherry on the interior of $\mathcal{T}'$. When we delete a leaf from this cherry, $\eta(\mathcal{T}')$ will reduce by at least 3. In both cases, this will give a simple duplication tree $\mathcal{T}'''$ on 8 leaves such that $\eta(\mathcal{T}''') \leq 9$. However, from above we know that $\eta(\mathcal{T}_8) = 10$. So this is a contradiction. The same argument applies if $\delta(n, \rho) \geq 6$

So assume that $\delta(1, \rho) < 6$ and $\delta(n, \rho) < 6$. If $\mathcal{T}'$ has an interior edge, then there is an interior cherry. Deleting one of the leaves from this cherry will reduce $\eta(\mathcal{T}')$ by at least

3 and hence obtain a contradiction. Therefore if $\eta(\mathcal{T}') \leq 12$, $\mathcal{T}$ has no interior edges. If $\mathcal{T}$ is a caterpillar then $\eta(\mathcal{T}') = 22 > 13 = \eta(\mathcal{T})$. So the only possibility left for a tree $\mathcal{T}'$ such that $\eta(\mathcal{T}') \leq 12$ is a double caterpillar. The formula from Lemma 6.7 shows that for a double caterpillar $\mathcal{T}''$ on n leaves

$$\eta(\mathcal{T}'') = n + \sum_{i=1}^{r-3}(i) + \sum_{j=1}^{s-3}(j),$$

where $r + s = n$. This is minimized when $r = \lfloor n/2 \rfloor$ and $s = \lceil n/2 \rceil$, that is, when $\mathcal{T}''$ is a balanced double caterpillar. Therefore if $\mathcal{T}'$ is a tree on 9 leaves then $\eta(\mathcal{T}') \geq 13$ and $\eta(\mathcal{T}) \leq \eta(\mathcal{T}')$ for all $\mathcal{T}'$ on 9 leaves if and only if $\mathcal{T}$ is a balanced double caterpillar.

Finally, we will show that for $n > 8$, $\eta(\mathcal{T}_n) \geq 10 + 3(n-8)$. This is true for the base case $n = 9$.

So assume that this is true for $9, \ldots, n-1$. Take a duplication tree $\mathcal{T}$ on n leaves such that $\eta(\mathcal{T}) < 10 + 3(n-8)$. In $\mathcal{T}$, because $n > 9$, one of the following possibilities must occur

1. $\delta(1, \rho) > 5$ and the shrub that contains 1 has size $(1, 1)$, or
2. $\delta(n, \rho) > 5$ and the shrub that contains n has size $(1, 1)$, or
3. there exists an interior cherry.

In cases 1 or 2 we can delete 1 or n respectively to give a duplication tree $\mathcal{T}'$ on $n-1$ leaves such that $\eta(\mathcal{T}') < \eta(\mathcal{T}_{n-1})$. This is a contradiction. In case 3, once again we can obtain a simple duplication tree on $n-1$ leaves such that $\eta(\mathcal{T}') < \eta(\mathcal{T}_{n-1})$. Therefore, for $n > 8$, $\eta(\mathcal{T}_n) \geq 10 + 3(n-8)$. If we take the simple duplication tree $\mathcal{T}$ shown above and adjoin a christmas tree to the edge 8, then each time we add an edge, we increase $\eta(\mathcal{T})$ by 3.

Therefore, if $\mathcal{T}$ is a balanced double caterpillar with a christmas tree attached to the leaves 1 and/or n and such that $n > 8$, $4 \leq \delta(\rho, 1) \leq 5$ and $4 \leq \delta(\rho, n) \leq 5$, then $\eta(\mathcal{T}) \leq \eta(\mathcal{T}')$ for all $\mathcal{T}'$ with n leaves.

□

From Lemma 6.10 and Theorem 6.15 we get tight upper and lower bounds for $\eta(\mathcal{T})$ where $\mathcal{T}$ is a simple duplication tree on $n \geq 4$ leaves. The cases $n = 2$ and $n = 3$ has not been included because there is only one possible duplication tree $\mathcal{T}$ on 2 leaves and in this case $\eta(\mathcal{T}) = 0$ and $\eta(\mathcal{T}) = 1$ for all duplication trees $\mathcal{T}$ with 3 leaves.

An upper bound for $\eta(\mathcal{T})$ on $n \geq 4$ leaves is given by

$$\eta(\mathcal{T}) \leq 2\lfloor n/2 \rfloor \cdot \lceil n/2 \rceil + \sum_{i=1}^{\lfloor n/2 \rfloor - 3} (i) + \sum_{j=1}^{\lceil n/2 \rceil - 3} (j) + n - 8.$$

This is just the equation from Lemma 6.8 for $\eta(\mathcal{T})$ where $\mathcal{T}$ is an inverse balanced double caterpillar.

The lower bound is given by

$$\eta(\mathcal{T}) \geq \begin{cases} \sum_{i=1}^{n-3}(i) + 1 & \text{when } 4 \leq n \leq 5 \\ n + \sum_{i=1}^{\lfloor n/2 \rfloor - 3} (i) + \sum_{j=1}^{\lceil n/2 \rceil - 3} (j) & \text{when } 6 \leq n \leq 8 \\ 10 + 3(n-8) & \text{when } n > 8. \end{cases}$$

We can simplify this further to obtain

$$\eta(\mathcal{T}) \geq \begin{cases} 2n - 6 & \text{when } 4 \leq n \leq 8 \\ 10 + 3(n-8) & \text{when } n > 8. \end{cases}$$

When we compare the upper bound to the number of simple duplication trees on n leaves, we see that the upper bound on the size of the neighbourhood is much smaller then the number of possible trees. Because of this, we can effectively apply hill-climbing techniques to improve a simple duplication.

While these results were just for simple duplication trees, similar methods could be applied to trees with multiple duplication events.

List of Figures

Bibliography

[1] A. Aho, Y. Sagiv, T. Szymanski, and J. Ullman, *Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions*, SIAM **10** (1981), no. 3, 405–421.

[2] B. Allen and M. Steel, *Subtree transfer operations and their induced metrics on evolutionary trees*, Annals of Combinatorics **5** (2001), 2001.

[3] S. Arora and B. Barak, *Complexity theory: A modern approach*, Cambridge University Press, 2009.

[4] D. Bertrand and O. Gascuel, *Topological rearrangements and local search method for tandem duplication trees*, IEEE/ACM Trans. Comput. Biol. Bioinformatics **2** (2005), no. 1, 15–28.

[5] O. Bininda-Emonds (ed.), *Phylogenetic supertrees*, Computational Biology, vol. 4, Kluwer Academic Publishers, 2004.

[6] M. Bordewich, G. Evans, and C. Semple, *Extending the limits of supertree methods*, Annals of Combinatorics **10** (2006), 31–51.

[7] M. Bordewich and C. Semple, *On the computational complexity of the rooted subtree prune and regraft distance*, Annals of Combinatorics **8** (2004), no. 4, 409–423.

[8] M. Constantinescu and D. Sankoff, *An efficient algorithm for supertrees*, Journal of Classification **12** (1995), no. 1, 101–112.

[9] D.Jaitly, P.Kearney, G.Lin, and B.Ma, *Methods for reconstructing the history of tandem repeats and their application to the human genome*, J. Comput. Syst. Sci. **65** (2002), no. 3, 494–507.

[10] O. Elemento and O. Gascuel, *A fast and accurate distance algorithm to reconstruct tandem duplication trees*, Bioinformatics **18** (2002), 92–99.

[11] O. Elemento, O. Gascuel, and M. Lefranc, *Reconstructing the duplication history of tandemly repeated genes*, Molecular Biology and Evolution **19** (2002), 278–288.

[12] Lander E.S. et al, *Initial sequencing and analysis of the human genome*, Nature (2001), 860–921.

[13] W. Fitch, *Phylogenies constrained by cross-over process as illustrated by human hemoglobins in a thirteen-cycle, eleven amino-acid repeat in human apolipoprotein a-i*, Genetics **86** (1977), 623–644.

[14] M. Garey and D. Johnson, *Computers and intractability: A guid to the theory of np-completeness*, W.H. Freeman and Company, 1979.

[15] O. Gascuel, M. Hendy, A. Jean-Marie, and R. McLachlan, *The combinatorics of tandem duplication trees*, Systematic Biology **52** (2003), 110–118.

[16] M. Golumbic, *Algorithmic graph theory and perfect graphs*, Computer Science and Applied Mathematics, Academic Press, 1980.

[17] P. Humphries, *Combinatorial aspects of leaf-labelled trees*, Ph.D. thesis, University of Canterbury, 2008.

[18] ———, *Bounds on the size of the tbr unitneighbourhood*, Annals of Combinatorics, (accepted).

[19] D. Kagaris and S. Tragoudas, *Maximum weighted independent sets on transitive graphs and applications*, Integr. VLSI J. **27** (1999), no. 1, 77–86.

[20] M. Lajoie, D. Bertrand, N. El-Mabrouk, and O. Gascuel, *Duplication and inversion history of a tandemly repeated genes family*, Journal of Computational Biology **14** (2007), no. 4, 462–478.

[21] E. Lawler, *Combinatorial optimization networks and matroids*, Dover Publishers Inc, 1976.

[22] L.Zhang, B. Ma, L. Wang, and Y. Xu, *Greedy method for inferring tandem duplication history*, Bioinformatics **19** (2003), 1497–1504.

[23] M. Ng and N. Wormald, *Reconstruction of rooted trees from subtrees*, Discrete Appl. Math. **69** (1996), no. 1-2, 19–31.

[24] R. Page, *Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas*, Systematic biology **43** (1994), no. 1, 58–77.

[25] D. Robinson, *Comparison of labeled trees with valency three*, Journal of Combinatorial Theory **11** (1971), 105–119.

[26] C. Semple, P. Daniel, W. Hordijk, R. Page, and M. Steel, *Supertree algorithms for ancestral divergence dates and nested taxa*, Bioinformatics **20** (2004), no. 15, 2355–2360.

[27] C. Semple and M. Steel, *Phlyogenetics*, Oxford University Press, 2003.

[28] S.Ohno, *Evolution by gene duplication*, Springer-Verlag, 1970.

[29] M. Tang, M.Waterman, and S.Yooseph, *Zinc finger gene clusters and tandem gene duplication*, RECOMB '01: Proceedings of the fifth annual international conference on Computational biology, 2001, pp. 297–304.

[30] J. Zhang and N. Mei, *Evolution of antennapedia-class homeobox genes*, Genetics **142** (1996), 295–303.

Index