# LATERAL GENE TRANSFER DETECTION USING MULTIPLE AGREEMENT FORESTS

by

Kartik Kakadiya

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
July 2023

*To my family, the constant source of love and support that has fueled my academic pursuit. This thesis is dedicated to you, for your unwavering belief in my abilities and the sacrifices you have made. Your presence in my life has been my greatest motivation.*

*Thank you.*

# Table of Contents

# List of Tables

## List of Figures

## Abstract

Phylogenetic trees are used to illustrate evolutionary relationships between and among species. However, lateral gene transfers (LGTs) can cause different evolutionary histories for genes compared to the species. One method to identify possible LGT scenarios uses mathematical models called maximum agreement forests (MAFs). Previous MAF-based models require vast sequence data and cannot identify specific transfers, such as antibiotic resistance origins. This study extends single MAF analysis to identify particular LGT using several MAFs, focusing on transfers found in all MAFs of two phylogenetic trees, called obligate transfers. We present a method for enumerating all MAFs using modified branching rules and cluster reduction along with a method for identifying obligate transfers without enumerating all MAFs. Our findings through experiments suggest listing all MAFs is feasible for identifying obligate transfers. Furthermore, we propose methods for tracing the LGT endpoints for non-binary reference trees to improve running time by performing non-binary LGT analysis.

The MAF-based method for identifying obligate transfers can also be used to find $x\%$ common LGTs, in other words, transfers present in $x\%$ of total MAFs. We present the results we achieved by applying these methods to a dataset constructed for 244 bacterial taxas and another dataset comprising 144 bacterial and archaeal genomes. Our findings imply that listing all MAFs for identifying obligate transfers is a practical approach. However, we should reduce the time required to enumerate all potential MAFs in order to make the method plausible for large datasets. We employed non-binary LGT tracing techniques on a dataset containing a set of 20 genomes each of five species within the family *Enterobacteriaceae*, and the *Enterococcaceae* dataset of 102 genomes. We show that since our techniques were successful in locating the sources and destinations, performing non-binary LGT analysis to reduce execution time is feasible.

## Acknowledgements

I would like to express my gratitude to Dr. Chris Whidden, my supervisor, for his guidance and knowledge throughout my research process. His invaluable knowledge regarding the rSPR project and persistent support have had a significant impact on this work. I sincerely appreciate his patience, enlightening feedback, and dedication to my academic development. His guidance has greatly influenced this thesis's conclusion and helped me further develop my research methodology.

I would like to thank Dr. Rob Beiko for the knowledge-sharing related to the bioinformatics part of this work. The discussions and meeting sessions among the group members have been invaluable in shaping my research ideas and expanding my horizons. Furthermore, I would like to thank Rob and Dr. Norbert Zeh for being readers of this thesis, and Dr. Travis Gagie for agreeing to chair the defense.

Finally, I would like to express my gratitude to my family for their love, understanding, and encouragement throughout this journey. My perseverance and determination have been motivated by their unwavering faith in me and their constant encouragement. Thank you, everyone!

# Chapter 1

# Introduction

In evolutionary biology, phylogenetic trees play an important role because they serve as the standard model for investigating and visualizing the evolutionary relationships among a group of species such as bacteria based on different physical and genetic traits such as DNA sequences [23]. Continuity of genetic traits throughout lineages of phylogenetic trees occurs through vertical inheritance, which involves the transmission of genetic information from parent species to their children. The evolution and diversification of species are the result of changes in genetic information through gene transmission or other external factors, called mutation. These mutations lead to the emergence of new species over an extended period of time.

Additionally, due to reticulate evolutionary events such as lateral gene transfer, recombination, hybridization, and incomplete lineage sorting, the evolutionary histories of individual genes may vary from the overall history of species [17]. In this thesis, we focus on lateral gene transfer (LGT), which is a process in which genes get transferred between distantly related organisms. Although primarily impacting prokaryotes, eukaryotes including plants and animals have been shown to be impacted by lateral gene transfers [27]. Therefore, it is important to detect these LGT events because these events make it difficult to study the vertical evolution of species and because they can contribute to the emergence of disease-causing pathogens [33] and antibiotic resistance traits [37].

To study and infer LGT events, a wide range of computational methods are used in combination with genomic patterns. Genes obtained through LGTs might have a different proportion of guanine (G) and cytosine (C) nucleotides in DNA or RNA sequences than the recipient genome. Therefore, the GC content method evaluates the proportion of guanine and cytosine across the genome to determine the presence of laterally transferred genes. Genomic G+C concentration varies significantly among early-branching prokaryotic and eukaryotic species, suggesting that a variety

of evolutionary processes were followed [16]. As per the sequence similarity method, an uncharacteristically high degree of similarity between DNA sequences of species located far apart in the tree can be used to identify LGTs.

Phylogenetic trees are often visualized as graphs and some graph theory algorithms are employed to recognize the LGTs. Rooted trees represent the evolution of taxa from an implied common ancestor, whereas unrooted trees are commonly represented as graphs with every non-leaf node having degree more than or equal to three [42]. Comparing individual gene trees to a reference "species tree" and reconciling the trees using a computation model called subtree prune and regraft (SPR) is one of the methods for recognizing lateral gene transfers [21]. Finding maximum agreement forests (MAFs) is one of the methods used for computing SPR distance for rooted trees [1]. A collection of disjoint subtrees obtained by removing some set of edges from a phylogenetic tree is called its forest. An agreement forest (AF) of two phylogenies is a forest that can be obtained from both of them by removing a specific set of edges. An agreement forest with the minimum number of subtrees is referred to as a maximum agreement forest. While MAF-based algorithms can compute the SPR distance for rooted trees efficiently, no MAF formulation exists for the unrooted trees [42]. However, MAFs can be used to calculate TBR distance for unrooted phylogenetic trees [1]. In this work, we employ methods for computing SPR distance that primarily rely on finding MAFs for rooted phylogenetic trees. Therefore, the aforementioned SPR technique can be referred to as rooted subtree prune and regraft (rSPR).

Another graph-based method used for LGT analysis depends on constructing supertrees from a collection of gene trees. Supertrees are generated by reconciling numerous smaller, overlapping phylogenetic trees into a single structure, and the first phylogeny of nearly all extant mammals, among other large-scale phylogenies, have been represented using them [3]. Supertrees are a powerful technique for creating comprehensive phylogenies of groups having hundreds of species because they may indirectly combine multiple forms of phylogenetic information in a single tree [4]. Supertrees were used to perform phylogenetic analysis on 220,240 proteins from 144 sequenced prokaryotic genomes in [2]. The first supertree construction algorithm based on SPR distance was presented and used to create a supertree from more than 40 thousand gene trees in [44].

Often, it is difficult to generate strictly bifurcating phylogenetic trees in order to use the aforementioned methods. Multifurcation refers to non-leaf vertices of phylogenetic trees having more than two children. A hard multifurcation is when more than two offspring species simultaneously diverge from an ancestral species; in contrast, a soft multifurcation is when there is ambiguity about the evolutionary relationships between the distinct species due to a lack of information [28]. We typically assume that all multifurcations in a phylogenetic tree are soft because simultaneous divergence events are extremely rare [15, 19]. In phylogenetic trees, the support value of an internal node indicates the degree of confidence or evidence supporting the bipartition at that particular node. It describes how strong or well-supported the inferred relationship is based on the information at hand and the methods used to build the tree. The support value can be defined in several ways with different interpretations. For example, the bootstrap support [14] of a given bifurcation is the proportion of trees, constructed by resampling the original sequence alignment dataset multiple times, that contains the bifurcation. The value of support ranges between 0 and 1. The value 1 indicates that we are fully assured about the bifurcation at that node. On the other hand, a 0 support value indicates that we do not have any evidence to support the bifurcation at that node. While executing our proposed algorithms, we use the required support threshold to help us avoid the identification of non-existent gene transfers.

When multifurcating trees are forced to resolve into binary trees, evolutionary relationships that are not supported by the original data are inferred, and it is possible that meaningless reticulation events are also inferred [41]. This motivates developing LGT analysis algorithms capable of operating on multifurcating trees. Whidden et al. [41] presented an $\mathcal{O}(2.42^k \cdot n)$ algorithm capable of handling multifurcating trees and was implemented by Lee [25]. Considering soft multifurcation, when dealing with non-binary reference trees, different binary resolutions of the reference tree might be possible for different gene trees. Consequently, it becomes challenging to pinpoint the origins and destinations of LGT events in the reference tree, especially when an LGT event involves a few children of a multifurcating node in the reference tree or when a destination is only a small number of children of a multifurcating node in the reference tree. To address the aforementioned problem, we present techniques

for mapping lateral gene transfers involving multifurcating nodes in section 3.1. We also mention the advantages and drawbacks of each method to help in selecting the appropriate method based on requirements. We also employ these methods on the *Enterobacteriaceae* dataset [18, 24] and the *Enterococcaceae* dataset [5, 24] and present the results we obtained in section 4.1. Table 1.1 presents information about both the datasets used for experiments.

Table 1.1: Non-binary LGT tracing datasets

| **Name** | **Taxa (count)** | **Description** |
|---|---|---|
| *Enterobacteriaceae* dataset | *Citrobacter freundii* (20)<br>*Enterobacter cloacae* (20)<br>*Escherichia coli* (20)<br>*Klebsiella oxytoca* (20)<br>*Salmonella enterica* (20) | A 100-taxon non-binary reference tree dataset of Gram-negative bacteria. |
| *Enterococcaceae* dataset | *Enterococcus faecium* (100)<br>*Enterococcus faecalis* (1)<br>*Enterococcus hirae* (1) | A 102-taxon non-binary reference tree dataset of Gram-positive bacteria. |

The emphasis of these MAF-based techniques for LGT tracing, however, has been on tracing all possible LGT scenarios based on a single MAF of a tree pair for a set of phylogenetic trees [40, 44, 35]. As a result, focusing on particular LGT occurrences, such as the transmission of antibiotic resistance between two phylogenetic trees, has not received much attention. Dempsey [12] formulated the core MAF problem that is related and centered on identifying the edge set present in all MAFs. In contrast, in this work, we concentrate on the edge set not present in any MAFs or, equivalently, on the transfers present in all MAFs to focus on transfers involving a specific gene of our interest, particularly antibiotic resistance genes. We define it as an obligate lateral gene transfer (OLGT) if a lateral gene transfer event occurs in all potential maximum agreement forests between a pair of trees. If the transfer event occurs in $x\%$ of all MAFs, it can be referred to as $x\%$ common transfer. This is possible because there can be multiple minimum LGT reconciliation scenarios based on each MAF. Therefore, it is essential to aggregate information from all of them as a method to study LGT.

Jordan Dempsey (personal communication, January 2023) suggested applying an optimization technique called edge protection [39, 12], to quickly identify OLGTs. The algorithms described in [43, 40] traverse through the edges of the phylogenetic

trees while attempting to cut them to obtain an MAF. Edge protection safeguards previously visited edges to prevent getting the same solution through different paths of tree traversal. However, this technique can only be used to identify obligate transfers and is unable to recognize 90% or 95% common transfers. Therefore, it is necessary to identify all potential maximum agreement forests in order to identify the $x\%$ common transfer between a reference tree and a gene tree. We first describe a basic approach to list all possible MAFs between a pair of trees in section 3.2.1.

Computing rSPR distance or obtaining an MAF for a pair of phylogenetic trees are NP-hard problems [8, 22, 11]. For example, our naive approach in section 3.2.1 has $\mathcal{O}(3^k \cdot n)$ running time complexity. As a result, several improvements have been made to improve the running time by refraining from enumerating all possible MAFs [40, 39]. Contrarily, as we focused on listing every possible MAF, we needed to figure out which optimizations were still applicable. Therefore, an improved approach with applicable optimizations is presented in section 3.2.2 to identify all possible MAFs. We used this algorithm to extract all MAFs from the Aquificae dataset [44], and the results are shown in section 4.2.1. We also present the implementation of the approach

Table 1.2: Obligate LGT tracing datasets

| **Name** | **Taxa (count)** | **Description** |
|---|---|---|
| *Aquificae* dataset | *Alphaproteobacteria* (15)<br>*Betaproteobacteria* (15)<br>*Deltaproteobacteria* (15)<br>*Epsilonproteobacteria* (14)<br>*Gammaproteobacteria* (13)<br>*Bacilli* (40)<br>*Clostridia* (34)<br>*Actinobacteria* (74)<br>*Deferribacteres* (2)<br>*Thermotogae* (11)<br>*Aquificae* (7)<br>*Nitrospira* (2)<br>*Synergistetes* (2) | A 244-taxon binary reference tree dataset for tracing obligate LGT via enumerating all MAFs. |
| Tanglegrams dataset | - | All possible small 4-7 leaf tree structures dataset for tracing obligate LGT via edge protection. |

based on the edge protection technique and the subsequent improvements in section 3.2.3 along with the results from applying it to all pairs of small 4-7 leaf trees that are different when relabeled, called tanglegrams [29], in section 4.2.2. Information on the datasets used for the experiments involving obligate LGT tracing is provided in Table 1.2.

The processing of phylogenetic trees using clusters is one of the improvements because it breaks them down into manageable smaller subtrees and reduces the experimental running time [26]. The identification of MAFs [39] and the construction of supertrees [44] both used the cluster reduction technique. Therefore, cluster reduction is also necessary to reduce the time it takes to enumerate all MAFs between two trees. Dempsey [12] mentioned that we can merge the MAFs from each individual cluster to obtain an overall MAF. For the purpose of obtaining all global MAFs, we present an algorithm for gluing together MAFs of separate clusters in section 3.3. We applied the clustering and gluing technique for enumerating MAFs to the 144 prokaryotic genomes dataset [2] and the outcomes are presented in section 4.2.3. Table 1.3 contains information about various taxas of the 144 prokaryotic genomes dataset.

Table 1.3: Clustering dataset

| **Name** | **Taxa (count)** | **Description** |
|---|---|---|
| 144 prokaryotic genomes dataset | *Crenarchaeota* (4)<br>*Euryarchaeota* (12)<br>*Nanoarchaeota* (1)<br>*Aquificales* (1)<br>*Bacteroidetes* (2)<br>*Chlamydiales* (7)<br>*Chlorobi* (1)<br>*Cyanobacteria* (8)<br>High G+C *Firmicutes* (12)<br>Low G+C *Firmicutes* (34)<br>*Planctomycetes* (1)<br>*Proteobacteria* (56)<br>*Spirochaetales* (3)<br>*Thermotogales* (1)<br>*Thermus/Deinococcus* (1) | A 144 genomes binary reference tree and binary gene trees dataset for enumerating all MAFs with clustering and gluing. |

This thesis presents the first work on summarizing information contained in multiple MAFs. For example, evaluating a reference tree compared to multiple gene trees

or evaluating multiple MAFs of a pair of trees. We do not propose these methods for actually converting one tree into the other, but rather for aggregating information for objectives like evaluating highways of gene transfer [2]. To accomplish that, we list every possible MAF between two phylogenetic trees. To reduce overall time complexity, we apply some optimizations and improve branching scenarios. Additionally, we used clustering and gluing techniques to reduce the experiment's running time. Clustering and gluing work with binary phylogenetic trees, and we intend to extend them to non-binary trees in the future. The results we obtained on the 144 prokaryotic genomes dataset and the Aquificae dataset indicate that we were able to significantly reduce the running time compared to the naive approach. The first study on tracing the sources and destinations of LGTs involving non-binary nodes is also presented in this thesis. We can apply multifurcating LGT algorithms with a higher required support threshold in addition to our techniques in order to reduce the running time significantly.

# Chapter 2

# Preliminaries and Related Work

## 2.1 Phylogenetic Trees and Agreement Forests

Throughout this paper, we mostly use the definitions and notation from [26, 39, 41, 42, 12]. A *phylogenetic X-tree* $T(X)$ can be described as a tree whose leaves are bijectively labeled with members of the label set $X$. The leaf set of the phylogenetic tree $T(X)$ can be denoted as $\mathcal{L}(T)$. The phylogenetic tree $T(X)$ is referred to as a *rooted phylogenetic tree* if it has a certain root node $\rho$; otherwise, it is known as an *unrooted phylogenetic tree*. A *rooted binary phylogenetic tree* is a rooted phylogenetic tree $T$ in which each internal node has exactly two children. If some internal nodes contain more than two children, it is called *rooted multifurcating phylogenetic tree*, which can be denoted as $T^m$. As described in the Introduction, we assume that

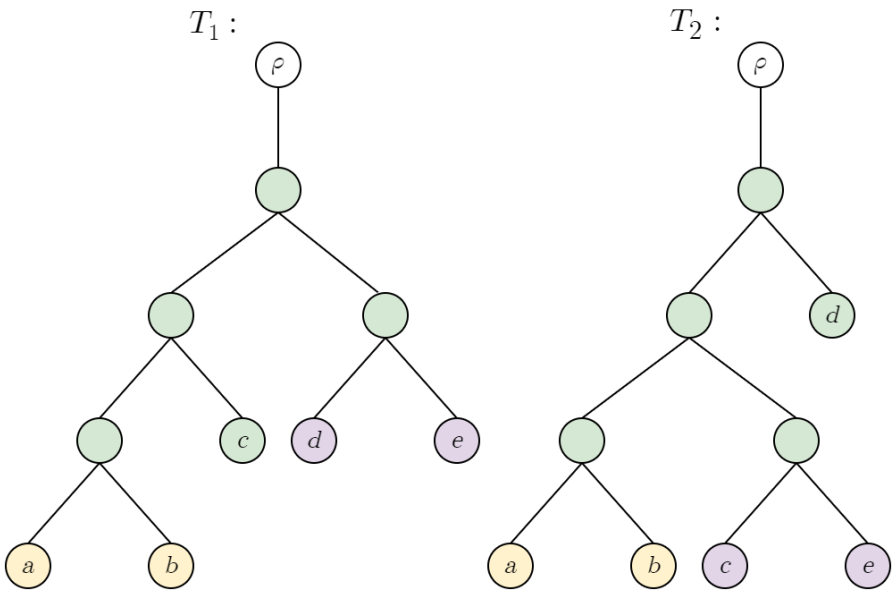



Figure 2.1: For phylogenetic trees $T_1$ and $T_2$, $(a, b)$ is a trivial sibling pair since it is present in both trees; $(d, e)$ is a sibling pair in $T_1$ and $(c, e)$ is a sibling pair in $T_2$.

multifurcations are soft and we can resolve them to construct fully binary trees. The term "*non-binary trees*" can also be used to describe multifurcating trees. The root node of these trees $\rho$ is considered to be part of the label set $X$. These phylogenetic trees can be represented as $T = (V, E)$ where $V$ is the *vertex set* and $E$ is the *edge set*.

For any node $u$, we can say that $u$ is a *descendant* of some node $v$ if and only if $v$ is on the path from $u$ to the root of the tree. In this case, we can say $v$ is an *ancestor* node of $u$. A node's *depth* is the number of its ancestors. The total number of nodes in a tree is referred to as its *tree size*. For a phylogenetic tree $T$, if there exists a pair of leaves $(a, c)$ such that $a$ and $c$ have a common parent node, then $(a, c)$ is called a *sibling pair* or a *cherry*. If that sibling pair exists for a pair of trees $(T_1, T_2)$, it is called a *trivial sibling pair* or a *trivial cherry* of $(T_1, T_2)$. Figure 2.1 shows a trivial sibling pair and a couple of non-trivial sibling pairs for $(T_1, T_2)$.

To define and find agreement forests we use an operation that cuts some edges of the phylogenetic trees in order to achieve maximum consensus. For a phylogenetic tree $T$, we can cut any edge $e = (u, v) \in E$, where $u$ and $v$ are the vertices, and this operation can be denoted as $T - \{e\}$. Performing edge cuts may result in some

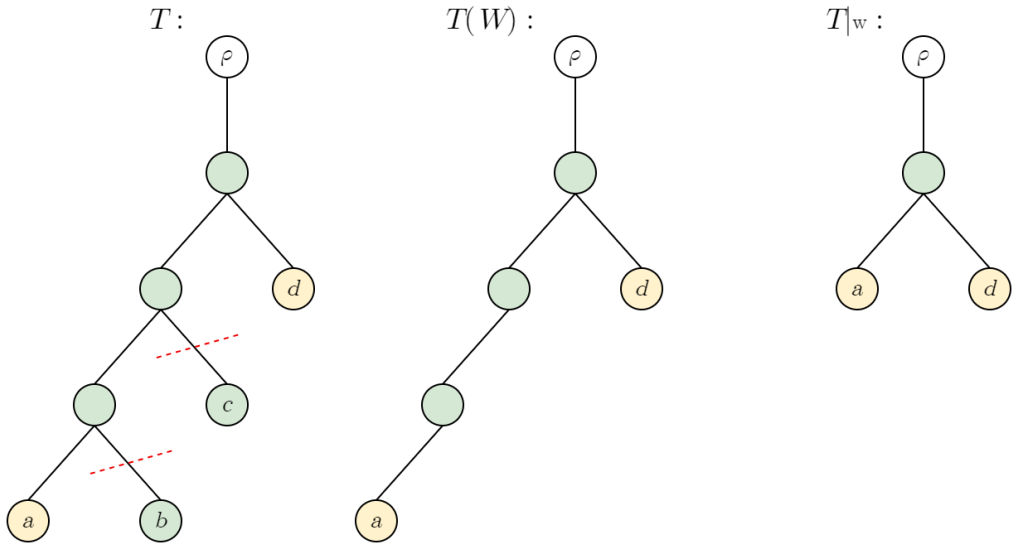



Figure 2.2: For phylogenetic tree $T$ and leaf subset $W = \{a, d\}$, $T(W)$ is minimal subtree of $T$ with leaf set $W$ and $T|_W$ is tree generated after applying forced contraction on $T(W)$.

internal nodes being left with only one child. If there exists an internal node $v$ that has only one child $u$, then we can remove $v$ from the tree along with its incident edges to its parent $w$ and $u$, and connect the nodes $u$ and $w$ with a new edge $(u, w)$. This operation is called *forced contraction* of node $v$. For a set $W \subseteq \mathcal{L}(T)$, the minimal subtree of $T$ that contains all elements of $W$ can be denoted $T(W)$. The tree generated after applying all forced contractions to $T(W)$ can be denoted as $T|_W$. Figure 2.2 shows a tree generated after applying forced contraction.

For a phylogenetic tree $T = (V, E)$ and an edge set $E' \subseteq E$, a collection of subtrees, also known as *components*, obtained by $T - E'$ is called a *forest* $\mathcal{F}$ of $T$. Suppose, we have a pair of rooted binary phylogenetic trees $T_1$ and $T_2$ with $\mathcal{L}(T_1) = \mathcal{L}(T_2) = \mathcal{L}(T)$ as their leaf set. An *agreement forest (AF)* of these two trees can be described as a set of binary trees $\mathcal{F} = \{t_1, t_2, ..., t_k\}$, where $\mathcal{L}_j := \mathcal{L}(T_j)$ for $j \in \{1, 2, ..., k\}$, such that the following conditions are satisfied:

1. $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_k$ partitions $\mathcal{L}(T)$,
2. $t_j = T_{1|\mathcal{L}_j} = T_{2|\mathcal{L}_j}$ for all $j \in \{1, 2, ..., k\}$,
3. For both $i = 1$ and $i = 2$ we have that $\{T_i(\mathcal{L}_j) : j = 1, 2, ..., k\}$ are edge-disjoint subtrees of $T_i$ [1].

Let $T_1^m$ and $T_2^m$ be two rooted multifurcating phylogenetic trees and let $T_1$ and $T_2$ be one of the binary refinements of $T_1^m$ and $T_2^m$, then an agreement forest $\mathcal{F}$ of $T_1$ and $T_2$ can also be regarded as an agreement forest of $T_1^m$ and $T_2^m$. An agreement forest with the minimum number of components possible is called *maximum agreement forest (MAF)*. There can be multiple possible maximum agreement forests for a pair of phylogenetic trees. All maximum agreement forests possible for two rooted phylogenetic trees $T_1$ and $T_2$ are shown in Figure 2.3.

If there is a component $t_i$ with no incident edges in a forest $\mathcal{F}$, it is referred to be a *singleton*. We remove the parent edge of a leaf node to make it a singleton in the forest $\mathcal{F}$. For example, nodes $a$ and $e$ of the MAF $\mathcal{F}_1$ in Figure 2.3 are considered as singletons. To *contract a sibling pair* $(a, c)$ in forest $\mathcal{F}$, we remove $a$ and $c$ from the forest and relabel the parent node of the sibling pair with $(a, c)$. If the parent node of a subtree $B$ in forest $\mathcal{F}$ lies on the path from node $a$ to $c$, it is called a *pendant subtree* of the pair $(a, c)$.

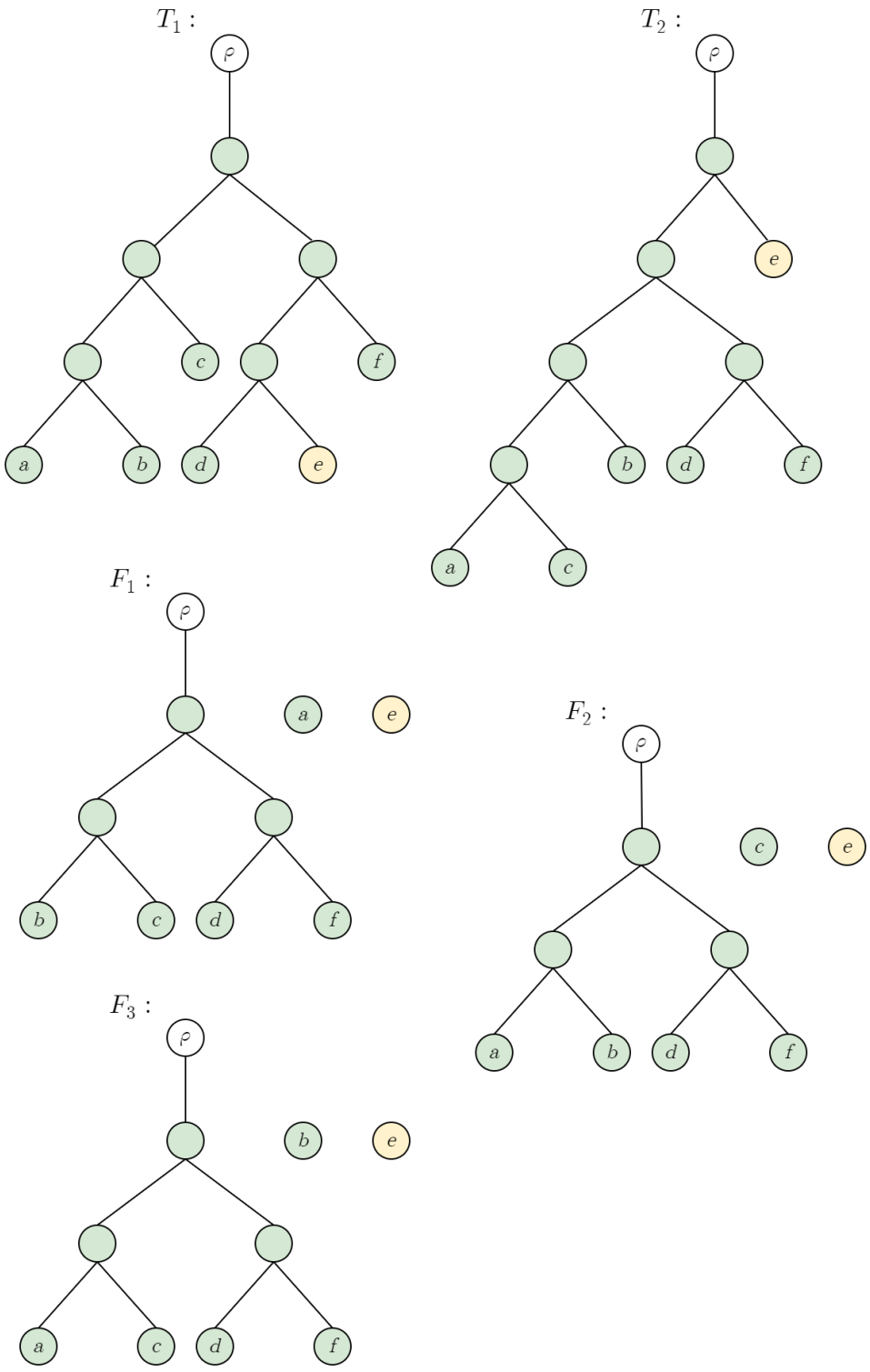



Figure 2.3: All MAFs of two phylogenetic trees $(T_1, T_2)$. Component with node $e$ is present in all the MAFs, therefore there is an obligate lateral gene transfer.

For some rooted tree $T$, a *rooted subtree prune and regraft (rSPR)* operation involves cutting some edge $(u, v)$, where $v$ is the parent of $u$. It would result in two subtrees of $T$, namely $T_u$ and $T_v$. It then attaches the subtree $T_u$ to a newly created internal node $v^{'}$ between edge $(x, y)$, where $y$ is the parent of $x$. This would result in adding two new edges $(v^{'}, y)$ and $(x, v^{'})$ along with removing edge $(x, y)$ from the edge set $E$. Finally, we contract the node $v$ since it would have been left with only one child. This operation is illustrated in Figure 2.4. When dealing with multifurcating trees, if node $v$ still contains more than one child, it should not be contracted. For two phylogenetic trees $T_1$ and $T_2$ with leaf set $\mathcal{L}(T_1) = \mathcal{L}(T_2)$, the minimum number of rSPR operations required to transform $T_1$ into $T_2$ is called their *rSPR distance*, $d_{rSPR}(T_1, T_2)$. The rSPR distance of the rooted phylogenetic trees $(T_1, T_2)$ is one less than the number of components in an MAF. Therefore, for an MAF $\mathcal{F}$ of $(T_1, T_2)$, $d_{rSPR}(T_1, T_2) = |\mathcal{F}| - 1$ [8, 43].

Finally, we can formalize the concept of obligate lateral gene transfers. For a pair of phylogenetic trees $(T_1, T_2)$ with their leaf set $\mathcal{L}(T_1) = \mathcal{L}(T_2)$, let there exist an edge $e_i = (x_i, y_i) \in E_i$, where $y_i$ is parent of $x_i$, with the leaf set below $e_i$ be $\mathcal{L}^l(T_i)$ and the leaf set above $e_i$ be $\mathcal{L}^u(T_i)$ and let node $x_i$ have $k_i$ child nodes with leaf set below them be $\{\mathcal{L}^l_0(T_i), \mathcal{L}^l_1(T_i), ..., \mathcal{L}^l_{k_i-1}(T_i)\}$ for $i \in \{1, 2\}$. The edge $e_i$ is a *transfer edge* in an MAF $\mathcal{F}$ of $(T_1, T_2)$, if and only if the following criteria are satisfied:

1. There does not exist a component of $\mathcal{F}$ that has some leaves of both $\mathcal{L}^l(T_i)$ and $\mathcal{L}^u(T_i)$,

2. There exists a component of $\mathcal{F}$ that has some leaves of $\mathcal{L}^l_m(T_i)$ and $\mathcal{L}^l_n(T_i)$, where $m \neq n$ and $0 \leq m, n < k_i$.

If the edge $e_i$ is a transfer edge in all possible MAFs of $(T_1, T_2)$, then it can be referred to as an *obligate lateral gene transfer edge* of $(T_1, T_2)$. Similarly, $x\%$ *common transfers* can be defined with a slight change in the definition above, where the transfer edge must be present in at least $x\%$ of all MAFs. In Figure 2.3, we have shown all possible MAFs for two rooted phylogenetic trees $T_1$ and $T_2$. In that, we can see that each MAF has node $e$ as a separate component. Therefore, it can be concluded that it is an obligate lateral gene transfer, and the parent edge of node $e$ can be considered as an OLGT edge. Similarly, MAFs $\mathcal{F}_1$, $\mathcal{F}_2$, and $\mathcal{F}_3$ each have an additional separate

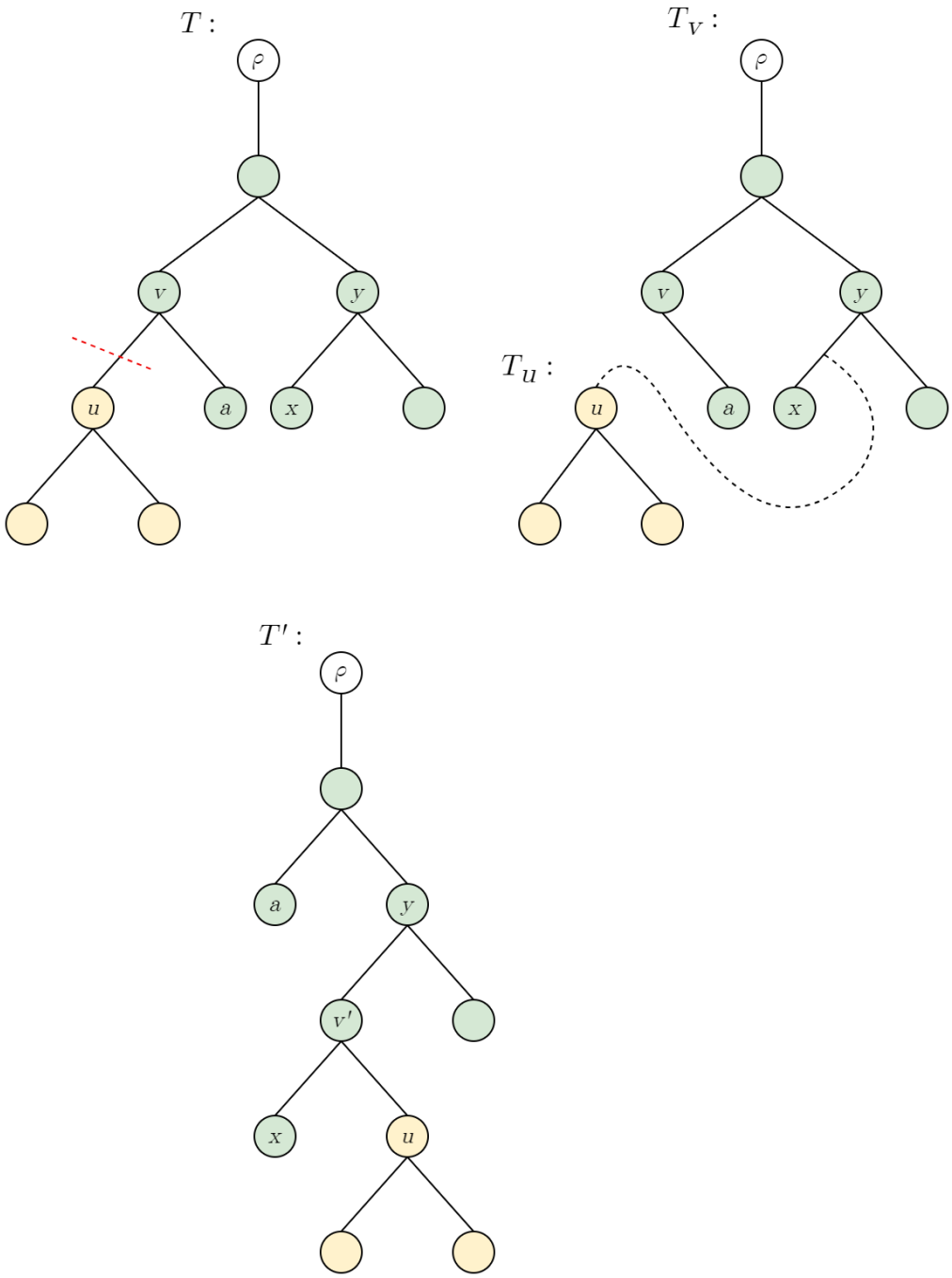



Figure 2.4: A *rSPR* operation on the tree, $T$. Prune operation creates trees $T_u$ and $T_v$. Regraft operation generates the resulting tree $T'$.

component $a$, $c$, and $b$, respectively. Due to their presence in only one of the three MAFs, they are all 33.33% common transfers.

In phylogenetics, the *Sackin index* is a metric used to assess the balance of a

phylogenetic tree. Using the Sackin index, we analyzed the impact of tree structures on the number of obligate transfers between tree pairs. The depths of all the leaves in the phylogenetic tree are added together in this metric. In comparison to a left- or right-aligned tree, a more balanced tree has a lower Sackin index. For trees $T_1$ and $T_2$ presented in Figure 2.3, their Sackin indexes are 16 and 18 respectively. Therefore, we can assume that the tree $T_1$ is more balanced than $T_2$. The *combined average Sackin index* can be calculated by dividing the total Sackin index of a set of phylogenetic trees by their average leaf count.

Understanding the algorithmic complexity in this work would benefit greatly from the understanding of a couple more notions of algorithmic complexity. A given problem is *fixed-parameter tractable (FPT)* if it can be solved in $\mathcal{O}(f(k) \cdot n^c)$, where $n$ is the input size, $c$ is a constant, and $k$ is a fixed parameter value passed to the algorithm along with input [13]. On the other hand, a *k-approximation* algorithm solves a problem by producing a solution that is within a factor $k$ of the optimal solution [38].

## 2.2 Related Work

In recent years, graph theory-based techniques have received a lot of attention for calculating rSPR distance and tracing lateral gene transfers. According to Bordewich and Semple [8], calculating the rSPR distance between two rooted binary phylogenetic trees is an NP-hard problem and it is fixed-parameter tractable in their rSPR distance. Hickey et al. [22] identified that calculating SPR distance is NP-hard for unrooted trees too. Collins [11] extended the FPT algorithm of the minimum rSPR distance via MAFs for non-binary trees.

There exist a few approximation algorithms, such as the 3-approximation algorithm put forth by Hein et al. [21] along with its NP-hardness proof. He claimed that maximum agreement forests (MAFs) are useful for calculating SPR distances for rooted trees. Allen and Steel [1] drew attention to the fact that the proof of the aforementioned claim of [21] is incorrect. Bordewich and Semple [8] corrected the claim by including the root of the trees $\rho$ in the label set $X$. Bonet et al. [6] presented the first correct 5-approximation algorithm based on the flaws of [21] and [31]. He proved that the claims of 3-approximation algorithms of both [21] and [31] are

incorrect. Bordewich et al. [7] were able to reduce the approximation ratio of 5 to 3 at the cost of an increased running time of $\mathcal{O}(n^5)$. The running time was significantly improved by Rodrigues et al. [32] to $\mathcal{O}(n^2)$. Later, a linear time 3-approximation algorithm was presented by Whidden and Zeh [43]. The approximation ratio of three was enhanced to two in subsequent work [34, 10, 9].

The earliest fixed-parameter algorithm for computing rSPR distance between two trees in $\mathcal{O}(4^k \cdot k^5 + n^3)$ running time was presented by Bordewich et al. [7]. A parameterized algorithm with an $\mathcal{O}(4^k \cdot k^5 + n^{\mathcal{O}(1)})$ running time was presented by Hallett and McCartin [20]. Later, Whidden and Zeh [43] used straightforward branching rules and a "shifting lemma" proven by Bordewich et al. [7] to reach a substantially better running time of $\mathcal{O}(3^k \cdot n)$. In [40], this approach was enhanced to achieve an even better constraint of $\mathcal{O}(2.42^k \cdot n)$. This improvement was made possible by recognizing three different branching scenarios and applying a depth-bounded search technique. In order to achieve $\mathcal{O}(2^k \cdot n)$ running time for calculating the rSPR distance between two binary rooted trees, the algorithm presented in [40] was also modified employing edge protection to ignore duplicate search branches [39]. In the same work, Whidden also presented an $\mathcal{O}(2.42^k \cdot n)$ running time algorithm for handling two multifurcating trees. For the MAF problem on two unrooted multifurcating trees, Shi et al. [36] presented a $\mathcal{O}(4^k \cdot n^5)$ running time algorithm.

The computation of an MAF can be broken down into a number of smaller problems through a cluster reduction technique proposed by Linz and Semple [26], which has proven to be very effective for lowering running times on biological datasets in practice. The complex weighting technique of [26] was later modified by Whidden et al. [44], and they reduced the time needed to compute such a cluster reduction from the cubic scaling stated by [26] to linear in the size of the trees. The problem of finding a core MAF was addressed by Dempsey [12] using the aforementioned branching and clustering strategies along with enhancing certain branching conditions.

# Chapter 3

# Multifurcating LGT and Obligate LGT

## 3.1 LGT mapping

In this section, we turn our attention to mapping lateral gene transfers (LGTs) between a rooted multifurcating phylogenetic reference tree $T^m$ and a set of rooted non-binary gene trees. For non-binary trees, their MAFs are agreement forests of their binary refinements which contain recently added internal nodes that are not present in the original trees. As previously mentioned, the possibility of different binary resolutions of the reference tree for different gene trees makes it challenging to map sources and destinations of LGT events for these trees. Our techniques are necessary to address the aforementioned problem of tracing LGTs for non-binary trees. LGT tracing was handled for binary reference trees and non-binary gene trees previously, by resolving multifurcations to create an internal node that contains the transferred children and/or the sibling children of the transfer destination [25]. However, this method may not work with a non-binary reference tree, because a multifurcation may need to be resolved in different conflicting ways for two different gene trees. Therefore, we offer three methods for mapping these types of transfers.

An example of a reference tree $T^m$ is shown in Figure 3.1, which is used to trace the LGT of multiple children of a multifurcating node. Our source node in this case is a hypothetical parent node of $f$ and $g$. We look at two scenarios for the destination node. Node $d$ is the first destination, and some hypothetical parent node of nodes $b$ and $c$ is the second. Now, we go over the capabilities of our three approaches to handle these types of LGTs, as well as discuss their benefits and drawbacks.

### 3.1.1 Move Parent Node (MP)

Move parent node method involves assigning the transfer to the parent node of the multiple source child nodes that are involved in the transfer. As a result, we can

map the transfer of several child nodes in one LGT move. Similar to this, if the destination involves multiple nodes, we map it to the parent of those nodes. Instead of transferring some of its children, this method is based on the hypothetical scenario of moving the parent node. However, moving the parent node would result in moving all children of the internal node. This comes with the disadvantage that we assign the transfer to nodes that are not involved in the transfer. Compared to some of our other methods discussed later, this technique is especially useful when we want to record the number of transfers equal to the minimum as it does not overcount LGTs. For the example mentioned in Figure 3.1, we show sources and destinations identified via this method in Table 3.1.

Table 3.1: Move Parent Node method

| **Source** | **Destination** |
|---|---|
| $w$ | $d$ |
| $w$ | $u$ |

### 3.1.2 Move Parent with Maintain List (MPML)

To address the issue with the Move parent node method, this method handles transfer in the same manner as the Move parent node method described above with one minor addition. We are moving the parent node, therefore we are unable to identify the precise nodes involved in the transfer. To handle that, we keep a list of the nodes involved in transfers at the source and destination in this method. We maintain entries for the nodes involved in each transfer in this list, along with the transfer's source and destination. Although this method captures all the relevant information, it is somewhat complex to analyze and visualize. We display the sources, destinations, and node list in Table 3.2 for the example from Figure 3.1.

Table 3.2: Move Parent with Maintain List method

| **Source** | **Destination** | **List** |
|---|---|---|
| $w$ | $d$ | $(f, g)$,$(d)$ |
| $w$ | $u$ | $(f, g)$,$(b, c)$ |

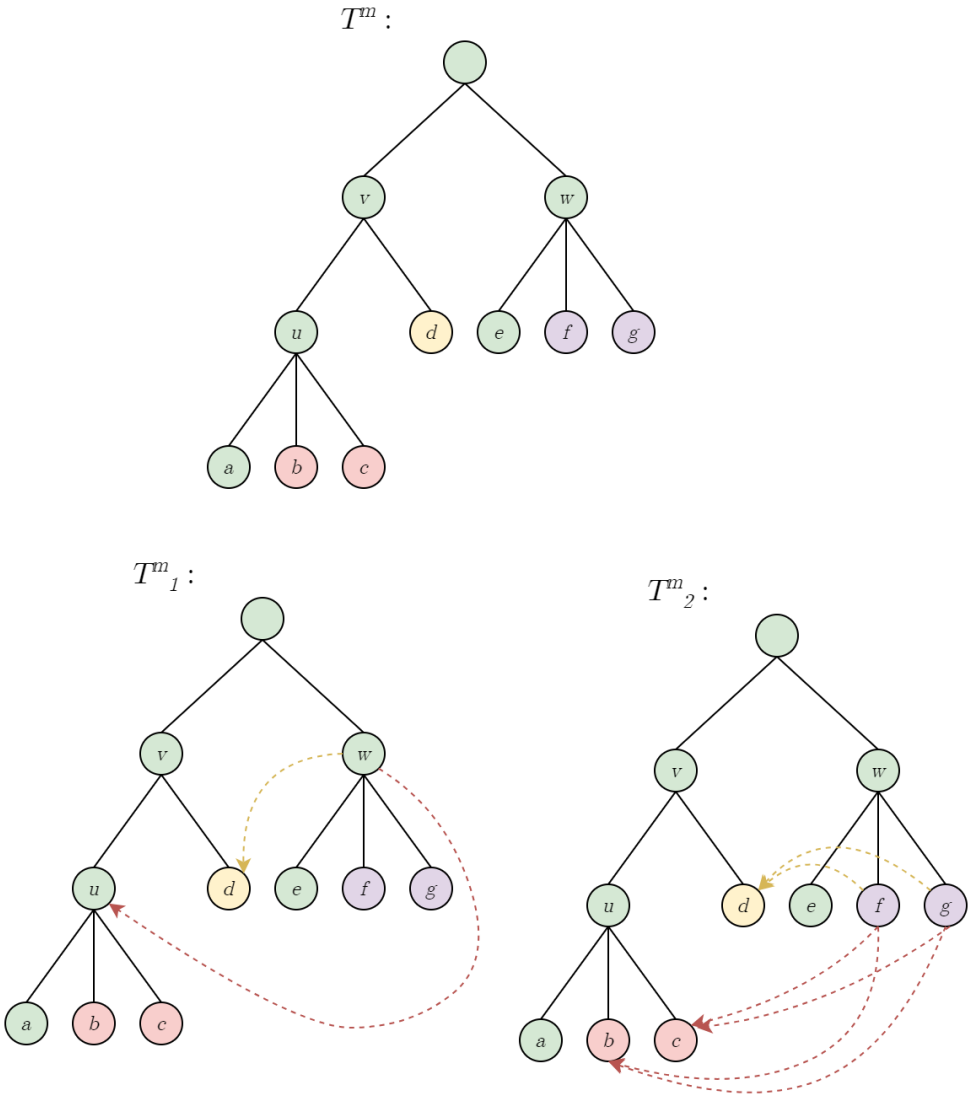



Figure 3.1: $T^m$ is reference tree involved in the lateral gene transfer. $T_1^m$ shows sources and destinations of transfers for the Move parent node, and Move parent with maintain list methods. $T_2^m$ shows sources and destinations of transfers for the Move individual node method.

### 3.1.3 Move Individual Node (MI)

To address the issue of moving nodes not involved in the transfer of the Move parent node method and the issue of challenging to analyze of the Move parent with maintain list method, the Move individual node method maps the transfer of each individual source child to each individual destination child. The drawback of this approach is that because we are treating each child separately, it might map a higher number of

Table 3.3: Move Individual Node method

| Case | Source | Destination |
|---|---|---|
| Case 1 | $f$ | $d$ |
| Case 1 | $g$ | $d$ |
| Case 2 | $f$ | $b$ |
| Case 2 | $f$ | $c$ |
| Case 2 | $g$ | $b$ |
| Case 2 | $g$ | $c$ |

transfers than the actual number of LGTs. For the example from Figure 3.1, we show the sources and destinations obtained via this method in Table 3.3.

In this section, we described three different approaches for mapping the LGTs involving multifurcating nodes of non-binary trees. Any of the aforementioned approaches can be chosen depending on the requirements as well as the benefits and drawbacks. In Section 4.1 we examine the difference between applying these methods on the *Enterobacteriaceae* dataset and the *Enterococcaceae* dataset.

## 3.2 Obligate LGT tracing

We now concentrate on the methods of identifying obligate lateral gene transfers (OLGTs) between a pair of rooted phylogenetic trees. In order to achieve that, we need to find all possible maximum agreement forests (MAFs) between that pair of trees. In order to enumerate all MAFs, we first present a naive approach that travels through all execution paths possible for a sibling pair. After that, we will introduce concepts of edge protection, edge replacement, and edge preference to further enhance the performance of the algorithm. The algorithms presented in this section are heavily inspired by the work presented in [40] and [12]. These algorithms are compatible with rooted binary phylogenetic trees, although the current implementation [24] can also handle non-binary gene trees. Lastly, we present a method based on edge protection for tracing OLGTs without enumerating all MAFs.

### 3.2.1 Naive approach

**Theorem 3.2.1.** *For two rooted X-trees $(T_1, T_2)$ and the parameter $k$, all possible MAFs can be computed in $\mathcal{O}(3^k \cdot n)$ time.*

*Proof.* Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be the forests of trees $T_1$ and $T_2$ respectively. For a singleton node in $a$ in $\mathcal{F}_2$, we must make it singleton in $\mathcal{F}_1$ as well in order to create an agreement forest. According to Lemma 4.2 in [12], a trivial sibling pair $(a, c)$ of $(\mathcal{F}_1, \mathcal{F}_2)$ can be contracted since they reside in the same connected component of every MAF.

**Algorithm 1:** Naive All MAFs Algorithm

```
Data: rooted X-trees (T1, T2), parameter k
Result: S_m, the set of all possible MAFs for (T1, T2)
Function MAF(F1, F2, k, S_m):
    if k < 0 then
        return;
    while F2 has singletons or F1 has sibling pairs do
        do
            choose valid singleton a;
            make a a singleton in F1;
        while F2 has singletons;
        do
            choose valid sibling pair (a, c);
            if (a, c) is trivial sibling pair then
                contract (a, c) in both F1 and F2;
            else
                if a and c are in separate components of F2 then
                    MAF(F1, F2//e_a, k − 1, S_m);
                    MAF(F1, F2//e_c, k − 1, S_m);
                else
                    let B be a pendant subtree along the path from a to c in F2;
                    MAF(F1, F2//e_a, k − 1, S_m);
                    MAF(F1, F2//e_B, k − 1, S_m);
                    MAF(F1, F2//e_c, k − 1, S_m);
        while F1 has sibling pairs;
    if F1 = F2 then
        S_m ← S_m ∪ (F1, F2);
S_m ← ∅;
MAF(T1, T2, k, S_m);
```

According to Lemma 3.6 in [40], for a non-trivial sibling pair $(a, c)$ of $\mathcal{F}_1$, if $a$ and $c$ are present in the separate components of $\mathcal{F}_2$ and they are not singletons (case **AC**) as well then an AF can be produced by cutting either $e_a$ or $e_c$. However, if $a$ and $c$ are present in the same connected component of $\mathcal{F}_2$ and $B$ is a pendant subtree of $a$ and $c$ (case **ABC**), an AF can be obtained by cutting either $e_a$, $e_B$ or $e_c$ in accordance with Lemma 3.7 in [40].

Based on the "shifting lemma" [7, 43], we can argue that an MAF can be obtained by cutting either of the three edges. Therefore, we are able to list every potential MAF in this situation because we are traversing through all three edges. For the same reason, the algorithm would take $\mathcal{O}(3^k \cdot n)$ as we make at most 3 recursive calls up to depth $k$ and each iteration would take $\mathcal{O}(n)$ time [40]. □

The Algorithm 1 displays the naive version of finding all maximum agreement forests between a pair of phylogenetic trees in accordance with Theorem 3.2.1. In the initial invocation of the MAF function, $\mathcal{F}_1 = T_1$, $\mathcal{F}_2 = T_2$, and $\mathcal{S}_m = \emptyset$. In the end, $\mathcal{S}_m$ will contain all possible MAFs between $T_1$ and $T_2$.

### 3.2.2 Better approach

In this approach, we make some improvements in the naive algorithm to avoid unnecessary or repetitive calls to the recursive function and present the Algorithm 2. The first of those improvements is the edge protection case first presented in section 6.3.2 of [39]. According to that, for a sibling pair $(a, c)$ in $\mathcal{F}_1$, if $a$ and $c$ are present in the same component then instead of performing cutting $e_a$, $e_B$ or $e_c$ to produce an AF, we should protect the $e_a$ after first two edge cut operations. The rationale behind this is that the MAFs discovered by cutting $e_c$ before cutting $e_a$ would have already been enumerated in the first recursive call that cuts $e_a$ first.

The second modification is the Depth rule presented in section 6.3.3 of [39]. According to it, if we have no protected edges, we select a sibling pair $(a, c)$ for which $a$ has more depth in $\mathcal{F}_1$. In the case of multiple sibling pairs with the greatest depth, we rely on the depth of $a$ in $\mathcal{F}_2$. If there exists some protected edge $e_a$ and let $z$ be the parent node of the smallest subtree of $\mathcal{F}_1$ that contains $e_a$, the sibling pair with the maximum depth in that subtree should be chosen. The depth rule is crucial in proving the improvement in running time through edge protection.

**Algorithm 2:** Better All MAFs Algorithm

```
Data: rooted X-trees (T1, T2), parameter k
Result: S_m, the set of all possible MAFs for (T1, T2)
Function MAF(F1, F2, k, S_m):
    if k < 0 then
        return;
    while F2 has singletons or F1 has sibling pairs do
        do
            choose valid singleton a;
            make a a singleton in F1;
        while F2 has singletons;
        do
            choose valid sibling pair (a, c) by depth rule;
            if (a, c) is trivial sibling pair then
                contract (a, c) in both F1 and F2;
            else
                if a and c are in separate components of F2 then
                    MAF(F1, F2//e_a, k − 1, S_m);
                    MAF(F1, F2//e_c, k − 1, S_m);
                else
                    if a and c has one pendant subtree B in F2 then
                        MAF(F1, F2//e_B, k − 1, S_m);
                        MAF(F1, F2//e_a, k − 1, S_m);
                        ReplaceAC operation;
                    else
                        MAF(F1, F2//e_a, k − 1, S_m);
                        MAF(F1, F2//{e_B1, ..., e_Bm}, k − m, S_m);
                        protect edge e_a;
                        MAF(F1, F2//e_c, k − 1, S_m);
        while F1 has sibling pairs;
    if F1 = F2 then
        S_m ← S_m ∪ (F1, F2);
S_m ← ∅;
MAF(T1, T2, k, S_m);
```

Lemma 4.6 in [12] states that for a sibling pair $(a, c)$ in $\mathcal{F}_1$, if $a$ and $c$ are present in the same component and have just one pendant subtree $B$ (case **AB**) on the path from $a$ to $c$ in $\mathcal{F}_2$, then instead of performing three edge cuts, there exists an MAF that cuts $e_a$ if and only if there exists another MAF that cuts $e_c$. Therefore, we can execute the recursive function for the edge $e_a$ and then swap node $c$ with node $a$ in the MAFs discovered by the edge $e_a$ recursive call to produce all the MAFs that would have been enumerated by calling the recursive function that cuts edge $e_c$. This operation is called the *ReplaceAC (RAC)* operation. The ReplaceAC operation also ensures that the recursive function that cuts edge $e_c$ gets called if node $a$ is protected. The running time of it depends on the number of MAFs enumerated in the first operation and it should be analyzed in future work.

Another modification that we made for the experiments is that we preferred cutting edge $e_B$ before cutting edge $e_a$ in the **AB** case. Then, we check whether $e_B$ recursive call was able to discover any MAFs or not. We perform cut $e_a$ recursive call and ReplceAC operation if and only if $e_B$ recursive call was able to find any MAFs. We refer to this modification as a *PreferB (PB)* case.

### 3.2.3 100% OLGT approach

**Algorithm 3:** 100% OLGT Algorithm

**Data:** rooted X-trees ($T_1$, $T_2$), parameter $k$
**Result:** $\mathcal{E}_o$, the set of obligate edges in $T_2$

```
1  Function TraceOLGT(T1, T2, s, Eo):
2      for edge e in E_T2 do
3          protect edge e;
4          find an MAF F_e of (T1, T2);
5          if |F_e| > s then
6              Eo ← Eo ∪ {e};
7  Eo ← ∅;
8  find an MAF F of (T1, T2);
9  s ← |F|;
10 TraceOLGT(T1, T2, s, Eo);
```

In this section, we discuss a slightly different approach that we used in experiments for tracing obligate transfers without enumerating all MAFs. This method was suggested by our fellow researcher Jordan Dempsey (personal communication, January 2023). This method can only be used to trace obligate transfers present in all MAFs, and it can not be used to trace the transfers present in 90% or 95% of the MAFs. We have presented the implementation of the method in Algorithm 3.

**Theorem 3.2.2.** *For two rooted X-trees* $(T_1 = (V_{T1}, \mathcal{E}_{T1}), T_2 = (V_{T2}, \mathcal{E}_{T2}))$ *and the parameter* $k$*, all obligate LGTs can be traced in* $\mathcal{O}(2.42^k \cdot n^2)$ *time.*

*Proof.* According to [39], an MAF between two rooted X-trees can be computed in

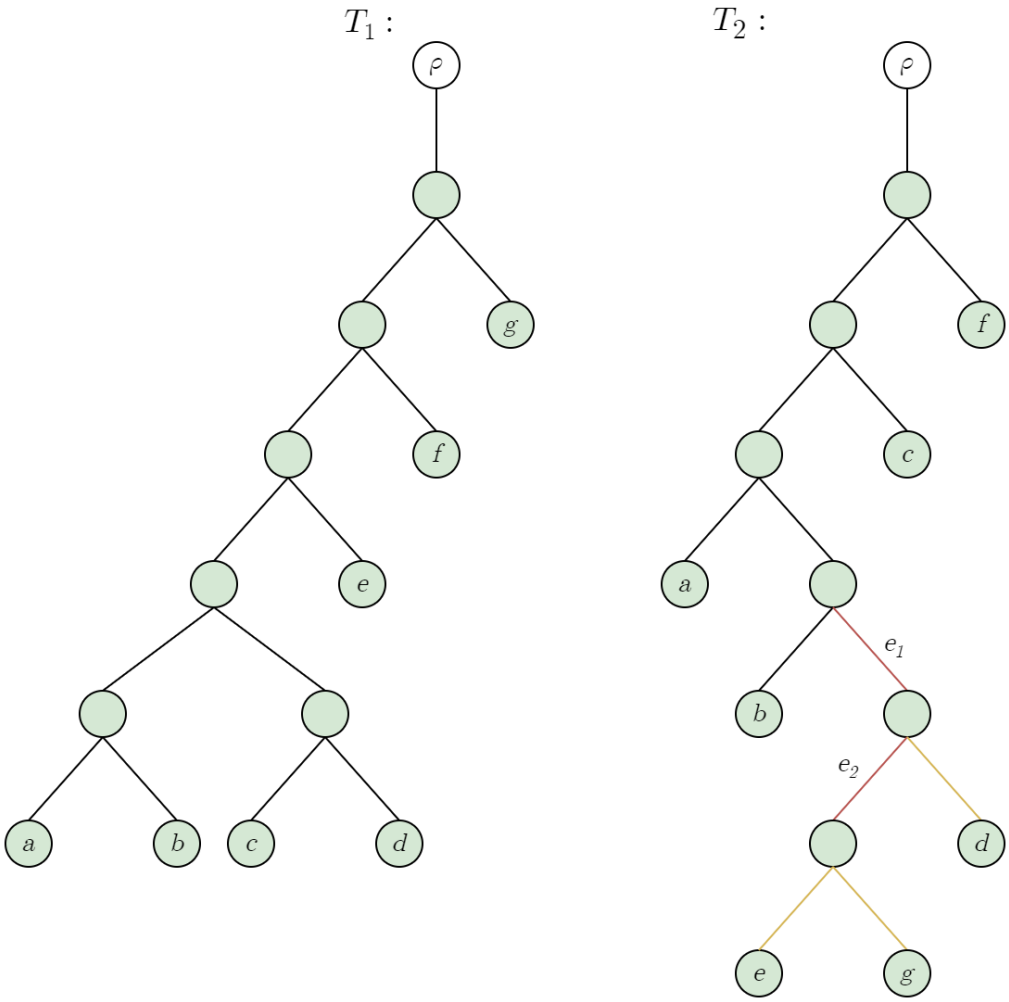



Figure 3.2: The tree pair has rSPR distance 3 however, the algorithm 3 suggested 5 OLGT edges which are colored in $T_2$.

$\mathcal{O}(2.42^k \cdot n)$ time and in $\mathcal{O}(2^k \cdot n)$ time if both are binary trees. A rooted X-tree with $n$ leaves has $\mathcal{O}(n)$ edges. Therefore, we can argue that our Algorithm 3 can be executed in $\mathcal{O}(2.42^k \cdot n^2)$ time and $\mathcal{O}(2^k \cdot n^2)$ respectively using algorithms presented in [39]. □

While employing algorithm 3, we came across some pairs of trees that had a higher OLGT count than their rSPR distance. The tree pairs are shown in Figure 3.2. This was caused by the fact that the algorithm 3 treated the edges $e_1$ and $e_2$ of $T_2$ as OLGT edges even though all of their descendant edges are OLGT edges. Therefore, we made some minor modifications to the previous algorithm to obtain the correct OLGT edge count. The new algorithm 4, keeps track of whether or not all descendant edges are OLGT and only performs an OLGT check on that edge if they are not. To support the OLGT check of ancestral edges, it still maintains an OLGT status for

**Algorithm 4:** Fixed 100% OLGT Algorithm

```
Data: rooted X-trees (T1, T2), parameter k
Result: E_o, the set of obligate edges in T2
Function TraceOLGT(T1, T2, s, E_o, n):
    is_obligate = true;
    for every child c of n do
        e = (c, n);
        if not TraceOLGT(T1, T2, s, E_o, c) then
            protect edge e;
            find an MAF F_e of (T1, T2);
            if |F_e| > s then
                E_o ← E_o ∪ {e};
            else
                is_obligate = false;
    return is_obligate;
E_o ← ∅;
find an MAF F of (T1, T2);
s ← |F|;
n ← root of T2;
TraceOLGT(T1, T2, s, E_o, n);
```

that edge. Therefore, by avoiding the OLGT check on unnecessary edges, we also reduce the algorithm's experimental running time.

### 3.3 Handling Clusters

Linz and Semple [26] presented the idea of partitioning the input trees into smaller subtrees and then solving the rSPR problem for individual clusters. This idea can be used to solve the problem of enumerating all possible MAFs between a pair of rooted binary phylogenetic trees. Lemmas 3.10 to 3.14 in [12] state that we can produce a global MAF by gluing the MAFs of separate clusters. Here, we present Algorithm 5 that we implemented based on these lemmas.

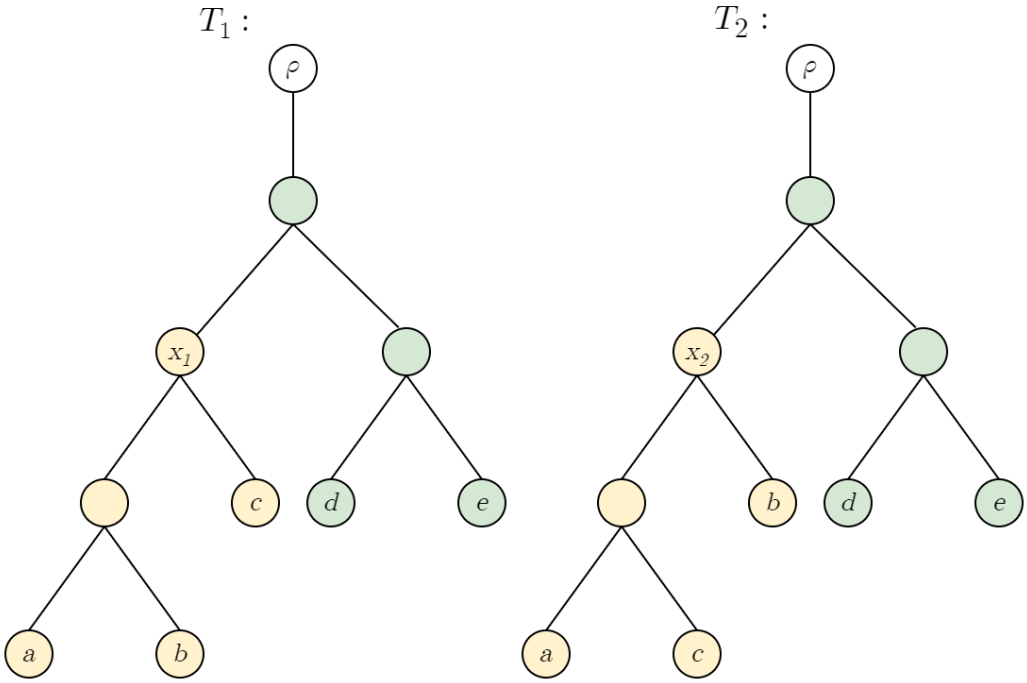



Figure 3.3: Two rooted binary phylogenetic trees $T_1$ and $T_2$ with cluster points $x_1$ and $x_2$ respectively. The clusters are shown with different colors.

We begin by outlining a few definitions and notations that will be helpful in understanding the methods discussed in this section. For two rooted binary phylogenetic trees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ with their leaf set $\mathcal{L}(T_1) = \mathcal{L}(T_2) = \mathcal{L}(T)$, assume there exists nodes $x_1 \in V_1$ and $x_2 \in V_2$ such that $\mathcal{L}(T_{x1}) = \mathcal{L}(T_{x2}) \neq \mathcal{L}(T)$, where $\mathcal{L}(T_{x1})$ and $\mathcal{L}(T_{x2})$ are the set of all leaves that has $x_1$ and $x_2$ as their common

**Algorithm 5:** Merge MAFs Algorithm

**Data:** Maximum agreement forests ($\mathcal{F}_u$, $\mathcal{F}_l$)
**Result:** $\mathcal{F}_m$, the merged MAF

```
1  Function MergeMAFs(F_u, F_l):
2      F_m ← F_u;
3      if lower cluster is a separate component c_l in F_m then
4          remove c_l from F_m;
5          add all components of F_l except separate ρ in F_m;
6      else
7          if F_u does not have cluster node then
8              if F_l has separate ρ component then
9                  add all components of F_l except separate ρ in F_m;
10             else
11                 find best node n_u for F_l in F_m;
12                 attach root component of F_l to n_u;
13                 add all the other components of F_l to F_m;
14         else
15             find the parent n_c of cluster point for F_l in F_m;
16             if F_l has separate ρ component then
17                 remove cluster child of n_c;
18                 contract node n_c in F_m;
19                 add all components of F_l except separate ρ in F_m;
20             else
21                 replace cluster child of n_c with root component of F_l;
22                 add all the other components of F_l to F_m;
23     return F_m
```

ancestor, respectively, as described in Figure 3.3. Let $T_{x1}$ and $T_{x2}$ be the subtree of each trees with $x_1$ and $x_2$ as the root node, respectively. The upper cluster $(T_{u1}, T_{u2})$ can be obtained by replacing $T_{xi}$ in $T_i$ with a new leaf node $\alpha$ for $i \in \{1, 2\}$. The lower cluster $(T_{l1}, T_{l2})$ can be obtained by attaching separate root node($\rho$) to $T_{x1}$ and $T_{x2}$. The nodes $x_1$ and $x_2$ are called the cluster points.

While enumerating MAFs using Algorithm 2, we start with the lowest cluster and move upwards. After enumerating all MAFs of the lower cluster, we select an MAF

$\mathcal{F}_l$ and attach the component with the root node to the upper cluster before listing its MAFs [39]. For an MAF $\mathcal{F}_u$ of the upper cluster, we can say that the *lower cluster is a separate component* if there exists a component $c_l \in \mathcal{F}_u$ with $\mathcal{L}(c_l) \subset \mathcal{L}(T_x)$, where $\mathcal{L}(T_x)$ is the leaf set of the lower cluster. For an MAF $\mathcal{F}_u$ of the upper cluster, we can say that $\mathcal{F}_u$ *does not have cluster node* if the parent of the cluster point $x$ got removed from $\mathcal{F}_u$ due to contraction.

The analysis of the merging algorithm for a pair of rooted binary phylogenetic trees is out of the scope of this thesis still we present our limited analysis. A single merge operation can be executed in $\mathcal{O}(\log n)$ time if implemented carefully. From Theorem 3.2.1, we can deduce that the upper bound of the enumerating MAFs is $\mathcal{O}(3^k \cdot n)$. With the improvement presented in section 3.2.2 suggested in [39, 12] and the RAB case in [12], the running time of $\mathcal{O}(2.42^k \cdot n + |S_m|)$ is achieved, where $|S_m|$ is the number of MAFs to glue together. It is an open question whether $|S_m|$ is $\mathcal{O}(2.42^k \cdot n)$ or not, which would determine the running time here.

# Chapter 4

# Experiments

We now proceed to the experimental phase of our research on tracing lateral gene transfers (LGTs), after outlining the theoretical framework and research methodology in the previous chapters. We implemented all of the non-binary LGT tracing methods and obligate LGT finding algorithms [24] discussed in Chapter 3 and employed them on various datasets for LGT analysis. All the experiments were conducted on the cloud resources provided by the Digital Research Alliance of Canada (the Alliance) Federation. This cluster has 2 x AMD Rome 7532 @ 2.40 GHz 256M cache L3 CPU specification. However, the running time of the experiments may vary depending on the resources available at the time of execution.

## 4.1 LGT Mapping

The goal of experimenting with LGT mapping techniques presented in section 3.1 was to utilize them to discover the real sources and destinations of LGTs involving non-binary nodes. We used these techniques on two different datasets, the *Enterobacteriaceae* dataset [18, 24], and the *Enterococcaceae* datasets [5, 24], for the experiments. Both of these datasets are multifurcating datasets, which means that in addition to the gene trees, the reference tree is also a non-binary tree. Therefore, these datasets were suitable for our non-binary LGT tracing methods. The primary distinction between the two families is that the *Enterococcaceae* family is Gram-positive while the *Enterobacteriaceae* family is Gram-negative. Gram-positive bacteria lack the outer membrane layer, which makes them less resistant to antibiotics than Gram-negative ones [30].

For the Move parent node and the Move parent with maintain list methods, we expected to achieve the same number of transfer counts, and for the Move individual node method, a slightly higher number of transfer counts. We also desired to analyze the amount of time that would be required to compare tree pairs with a

given rSPR distance, and we anticipated that this time would increase as the rSPR distance increased. Hereby, we present the results gathered by applying our proposed LGT mapping methods in section 3.1 on the *Enterobacteriaceae* dataset and the *Enterococcaceae* dataset.

### 4.1.1 *Enterobacteriaceae* dataset results

The *Enterobacteriaceae* dataset [18, 24] is comprised of 7321 gene trees and a reference tree with 100 leaves. This dataset includes 20 genomes each of 5 different *Enterobacteriaceae* family species, including *Citrobacter freundii*, *Enterobacter cloacae*, *Escherichia coli*, *Klebsiella oxytoca*, and *Salmonella enterica*. This is a multifurcating dataset, which suggests that the reference tree and many of the gene trees contain multifurcating internal nodes. We experimented with 5 different required support values that include 0, 0.5, 0.7, 0.9, and 1.0. In total, we have tested 36605 (7321 gene trees with 5 different required support values) tree pairs. We assigned a maximum time limit of 72 hours for the comparison of each individual tree pair. Out of 36605, about 135 tree pairs failed to complete the comparison before the assigned time limit.

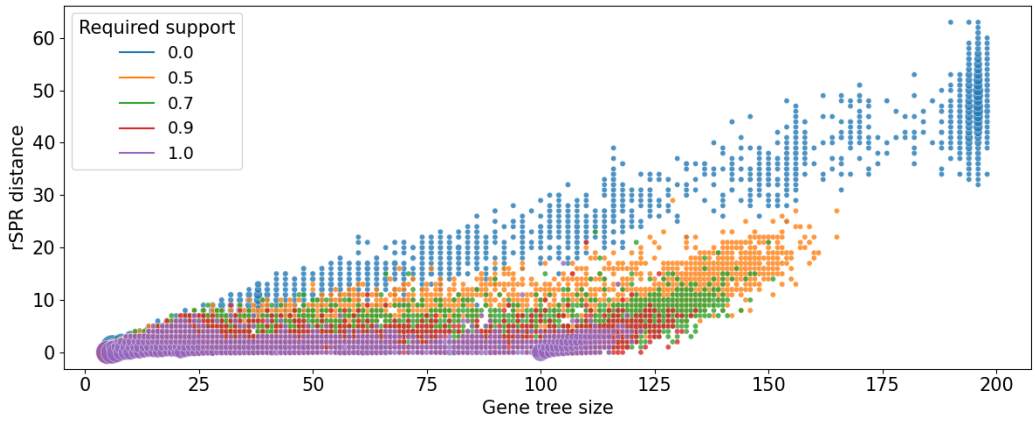
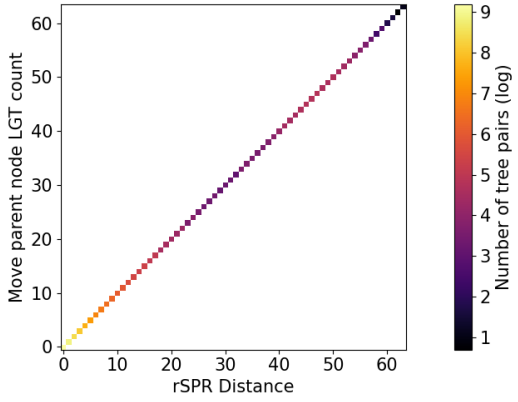


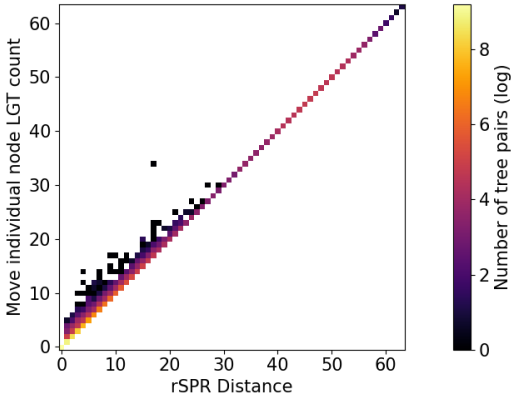

Figure 4.1: Relationship between the gene tree size and the rSPR distance to the reference tree for 5 distinct required support values. The size of each circle indicates the number of tree pairs that are there for a given gene tree size and rSPR distance.

The total number of nodes in the tree defines its *tree size*. For binary trees, the number of leaves is often considered as the tree size, but for multifurcating trees, we also take the number of internal nodes into account. The relationship between the size of the gene trees and their rSPR distance to the reference tree for a given required

support value is shown in Figure 4.1. For the required support values ranging between 0.0 and 1.0, the size of the reference tree varied between 170 (required support 1.0) - 199 (required support 0.0) and the gene tree size varied between 6 and 199. For the *Enterobacteriaceae* dataset, the rSPR distance ranged from 0 to 63 for a required support value of 0.0 and from 0 to 17 for a value of 1.0. We can infer from the plot in Figure 4.1 that as the gene tree size increases or the required support value decreases, there is an increase in the rSPR distance between the reference tree and the gene trees. Another noticeable outcome is the reduction in the overall gene tree size as the required support value increases. This is because we contract nodes with lower support values than the threshold. When using a higher required support threshold, the rSPR distance decreased by more than what could be accounted for the decreasing tree size i.e. the distances for required support higher than 0 are below the distances for required support 0 in the figure. These outcomes emphasize the importance of algorithms that can analyze multifurcating input trees because non-binary LGT analysis would identify many more transfers based on unsupported relationships. Furthermore, the considerable reduction in rSPR distance suggests that multifurcating algorithms with a reasonable support threshold may be much faster than binary algorithms even though the multifurcating algorithms are more complex.



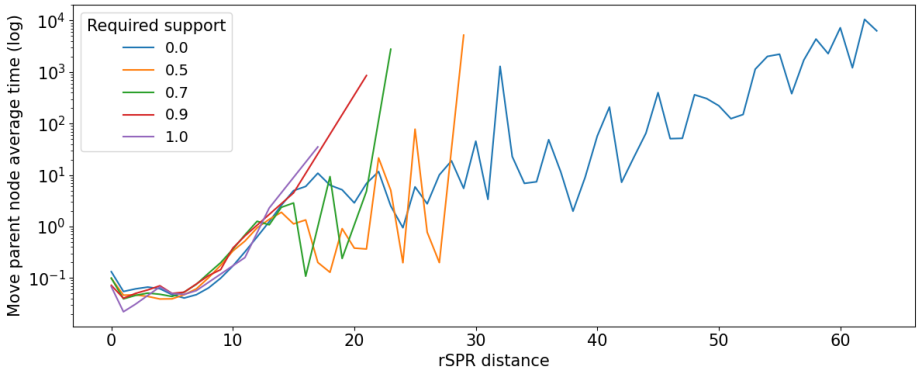

(a) rSPR distance vs. Move parent node LGT count



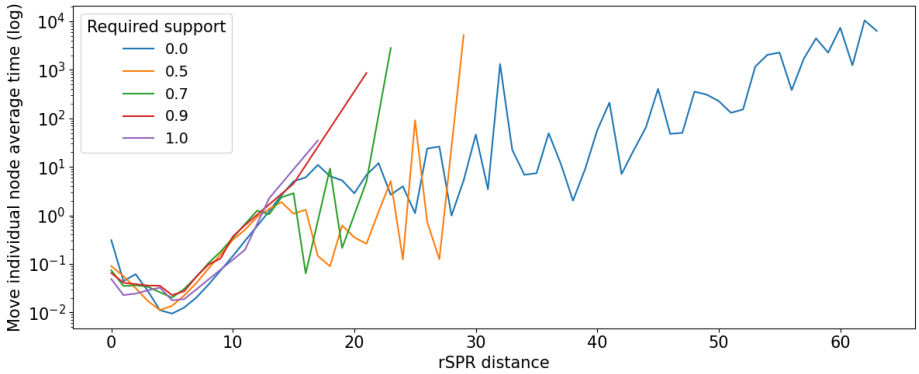

(b) rSPR distance vs. Move individual node LGT count

Figure 4.2: Heatmaps showing the Move parent node LGT count and the Move individual node LGT count compared to actual rSPR distance between the reference tree and a set of gene trees. These are combined results of tracing LGTs for 5 distinct required support values (total 36470 tree pairs)

In Figure 4.2, we present the transfer count results obtained after employing our LGT tracing methods presented in section 3.1 on the *Enterobacteriaceae* dataset. For the required support value of 0.0, the Move parent node transfer count ranged from 0 to 63, and for the required support value of 1.0, it ranged from 0 to 17. We can infer from Figure 4.2a that the rSPR distance between tree pairs was identical to the Move parent node LGT count as mentioned in 3.1.1. Nevertheless, as we trace each child separately, Figure 4.2b shows that the Move individual node LGT count was greater than the rSPR distance between tree pairs. The highest difference between the Move individual node LGT count and the rSPR distance was 17. Since we are not tracing any additional transfers, the Move parent node with maintain list produced




(a) Average move parent node method running time (log) for tree pairs with different rSPR distance and required support values




(b) Average move individual node method running time (log) for tree pairs with different rSPR distance and required support values

Figure 4.3: Average running time (log) required to trace LGT using the Move parent node and the Move individual node methods with 5 distinct required support values.

the same transfer count as the Move parent node method. These findings suggest that moving individual nodes is a reasonable trade-off for simplifying processing, though maintaining a list may be preferable if you require information about both the number of transfers and their precise sources and destinations.

Figure 4.3 displays the average running time required to perform the LGT tracing operation between tree pairs when applying our techniques. We expect all of our methods to utilize roughly the same running time since they are only utilized for mapping. Since the running time is exponential, we have kept a log of the actual running time. For the Move parent node method, the maximum running time for a single tree pair was 10 hours 56 minutes. When using the Move individual node method, the same tree pair took 11 hours 06 minutes. For all of our methods, there was very little difference in the amount of time required between the support threshold of 0.9 and 1.0. The overall trend of the running time was that it increased as the rSPR distance increased and the required support value decreased, as shown in both sub-graphs of Figure 4.3. The fluctuations were caused by the number of tree pairs that were available at that rSPR distance and required support level. According to the results shown in Figure 4.3, non-binary LGT analysis is much quicker than binary analysis because they have reduced distances. However, it is also possible that high thresholds may miss some transfers.

### 4.1.2 *Enterococcaceae* dataset results

The *Enterococcaceae* dataset [5, 24] consists of 4516 gene trees and a reference tree with 102 leaves. The dataset includes 100 genomes of the *Enterococcus faecium* species along with one genome of each of the *Enterococcus faecalis* and the *Enterococcus hirae* species of the *Enterococcaceae* family of bacterial organisms. The *Enterococcaceae* dataset is also a multifurcating dataset, where both the reference tree and gene trees are non-binary trees. In pre-processing, a total of 389 gene trees had to be removed from the dataset since they contained the same genome more than once. We have experimented with 4127 valid gene trees and 5 different required support values, including 0, 0.5, 0.7, 0.9, and 1.0. We have examined 20635 tree pairs in total. Same as the *Enterobacteriaceae* dataset, we set a maximum execution time of 72 hours for each individual tree pair. About 2329 tree pairs out of 20635 were unable to finish

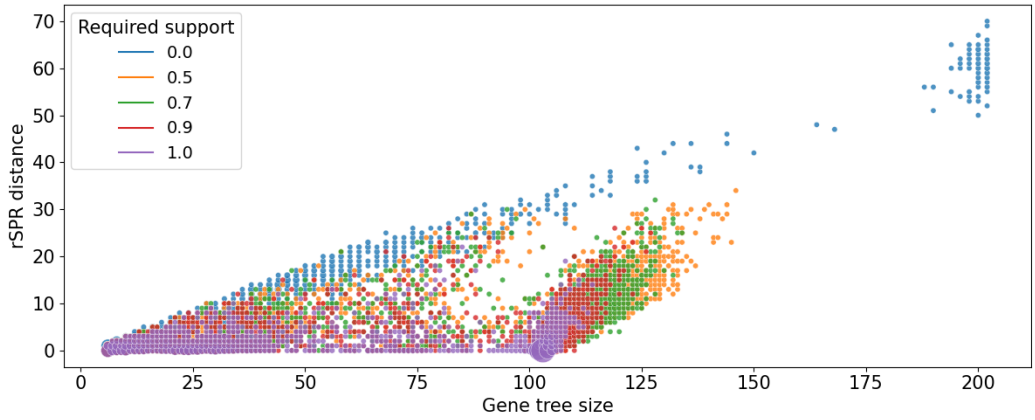
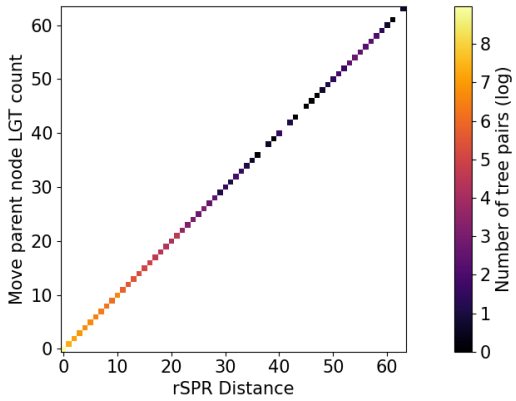



Figure 4.4: Relationship between the gene tree size and the rSPR distance to the reference tree for 5 distinct required support values. The size of each circle indicates the number of tree pairs that are there for a given gene tree size and rSPR distance.

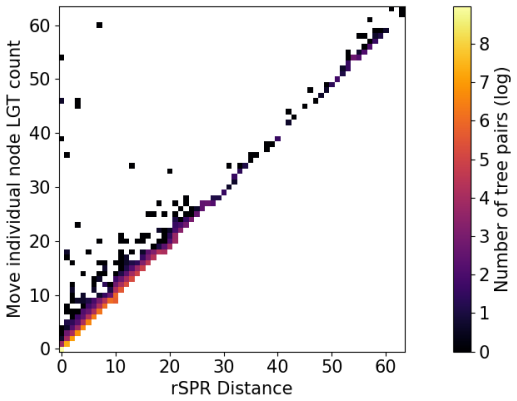
their execution within the allotted time frame.

Figure 4.4 illustrates the correlation between the gene tree size and its rSPR distance to the reference tree for a specific required support value. The rSPR distance for the *Enterococcaceae* dataset ranged from 0 to 70 for the required support value of 0.0 and from 0 to 22 for the required support value of 1.0. Therefore, we can deduce from Figure 4.4 that there is an increase in the rSPR distance between the reference tree and the gene trees as gene tree size increases or required support values decrease. When compared to the results from the *Enterobacteriaceae* dataset, we notice that the tree pairs are not completely segregated for various support thresholds, which demonstrates the wide variation in rSPR distance between the tree pairs in this dataset. Additionally, even with a higher support threshold, we see that there are many more tree pairs with higher rSPR distances, which indicates that we might not miss a lot of transfers when applying the higher value of the support threshold. Therefore, it is necessary to trace these transfers by applying multifurcating algorithms. Given that the rSPR distance between the tree pairs decreases by more than can be explained just by the reduced tree size as the support threshold increases, we expect a decrease in the running time of multifurcating algorithms.

In Figure 4.5, we present the transfer count results obtained after employing our LGT tracing methods presented in section 3.1 on the *Enterococcaceae* dataset. For the required support value of 0.0, the Move parent node transfer count ranged from




(a) rSPR distance vs. Move parent node LGT count



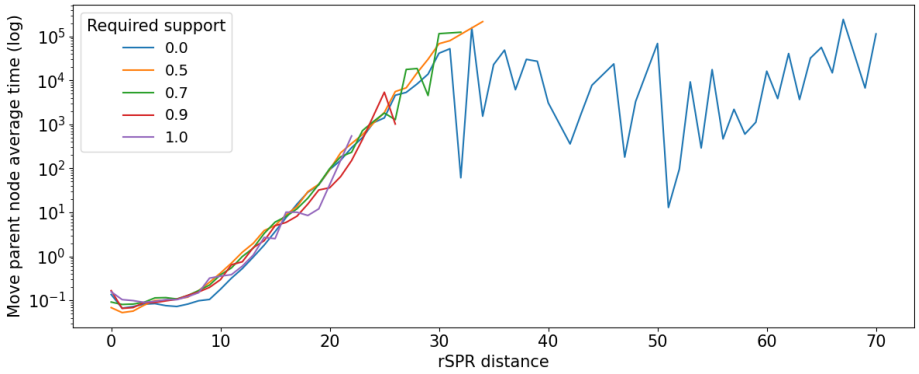

(b) rSPR distance vs. Move individual node LGT count

Figure 4.5: Heatmaps showing the Move parent node LGT count and Move individual node LGT count compared to actual rSPR distance between the reference tree and a set of gene trees. These are combined results of tracing LGT for 5 distinct required support values (total 18306 tree pairs)

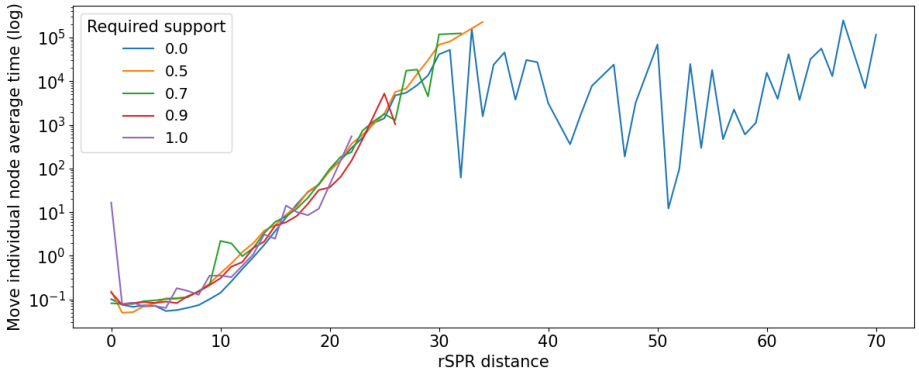
0 to 70, and for the required support value of 1.0, it ranged from 0 to 22. The Move individual node transfer count ranged from 0 to 71 for the required support value of 0.0, and from 0 to 67 for the required support value of 1.0. The rSPR distance between tree pairs, as shown in Figure 4.5a, was equal to the Move parent node LGT count. However, Figure 4.5b demonstrates that the Move individual node LGT count was much higher than the rSPR distance between tree pairs. The highest difference between the Move individual node LGT count and the rSPR distance was 59. The Move parent node method and the Move parent node with maintain list both resulted in the same number of transfers. The number of trees with a specific rSPR distance from the reference tree is indicated by the color chart in the heatmaps. The heatmaps show that there were fewer trees overall with higher rSPR distance, but the number was still statistically significant. From Figure 4.5, we can infer that applying the Move individual node method on the *Enterococcaceae* dataset might not be an appropriate choice considering the substantial variations in the number of transfers. However, the Move parent with maintain list method would be more suitable as it provides the precise number of transfers in addition to the list of sources and destinations.

Figure 4.6 shows the average amount of time taken by our techniques to perform LGT tracing between tree pairs. Same as previously described, we have taken a log

of the actual running time. The maximum running time for a single tree pair was approximately 71 hours for all our methods. The total running time for the entire dataset using the Move parent node method was 72 days 5 hours, and 22 minutes, and the total running time using the Move individual node method was 72 days 16 hours, and 35 minutes. As shown in both sub-graphs of Figure 4.6, when the rSPR distance increased and the required support value decreased, the running time generally increased. The fluctuations were caused by the number of tree pairs that were available at that rSPR distance and required support level. We have fewer fluctuations compared to the *Enterobacteriaceae* dataset, though, because the trees were evenly distributed across the various rSPR distance levels. When using the Move




(a) Average move parent node method running time (log) for tree pairs with different rSPR distance and required support values




(b) Average move individual node method running time (log) for tree pairs with different rSPR distance and required support values

Figure 4.6: Average running time (log) required to trace LGT using the Move parent node and the Move individual node methods with 5 distinct required support values.

individual node method, we were not expecting the spike at rSPR distance 0, and we believe the reason might be the availability of resources at the time of execution. Similar to the *Enterobacteriaceae* dataset, non-binary LGT analysis is much faster than binary LGT analysis. Therefore, it is beneficial to find a suitable threshold support that does not miss a lot of transfers and accelerates the execution.

## 4.2 Obligate LGT tracing results

We now change our attention to evaluating the obligate lateral gene transfer (OLGT) methods we described in section 3.2. The goal of these experiments was to identify OLGTs by enumerating every possible MAF between a pair of trees. Along with the impact of the PB case, which is mentioned in section 3.2.2, we also wanted to determine the effects of the optimizations presented there on our results. When compared to a naive approach, we expected that our optimizations would take much less time to list all MAFs between a pair of rooted trees. We expected the algorithm with PB case to enumerate a lesser number of MAFs than there actually are given that we are not traversing some useful edges. However, the objective of it was to determine if we could significantly reduce the running time or not. To determine the total number of OLGT edges, the 100% OLGT approach was used as a benchmark.

Our primary goal with the merging cluster MAFs method was to determine whether we could produce all global MAFs after performing the gluing operation. Along with that, we were also interested in comparing the running times for clustering and gluing to the running times required when clustering is not used.

### 4.2.1 Better approach OLGT results

In this section, we present the results gathered by applying our proposed better approach OLGT algorithm in section 3.2.2 on the Aquificae dataset [39]. The dataset contained gene trees constructed over a set of 1251 different taxas of bacteria. The Aquificae dataset contains 40631 gene trees and a reference tree with 244 leaves and has been used previously to analyze "highways" of lateral gene transfer across a diverse set of bacteria [39]. Our current implementation of the OLGT algorithm requires a binary reference tree so we could not use the *Enterobacteriaceae* dataset and the *Enterococcaceae* dataset. In order to evaluate the actual results, we applied

various approaches to the dataset without using any clustering. We set a maximum permitted time for the execution to 72 hours for individual tree pairs. In 40334 of the 40631 tree pairs, the execution could be finished in the allotted time.

Table 4.1: Better approach OLGT results

| **Optimizations** | **Number of MAFs (average)** | **Running time** |
|---|---|---|
| Naive approach | 9.251971 | 56h47m |
| Better approach | 9.251971 | 44h08m |
| Better approach (PB case) | 9.058754 | 42h35m |
| All optimizations approach | 1.101899 | 1h16m |

Table 4.1 shows the results that we were able to obtain using 4 different approaches for enumerating MAFs. Both the naive approach presented in section 3.2.1 and the better approach presented in section 3.2.2 enumerated every possible MAF as expected, though the better approach with the PB case enumerated 3% fewer MAFs, as we might miss traversing some necessary branches. Additionally, we used the all optimizations algorithm described in [39, 41] for calculating the rSPR distance via effectively generating MAFs. The average number of MAFs produced by this algorithm was slightly more than one. For a single tree pair, there were anywhere between 1 and 3240 MAFs. For our better approach, the running time has significantly improved because it took us about 22% less time to list every MAF. However, we were unable to make significant progress with the PB case. As expected, all optimizations algorithm was much faster than other approaches. The table suggests that enumerating all MAFs would require significantly more time than computing the rSPR distance between them. However, these are two different problems, and listing all MAFs requires traversing through a lot more branches. Therefore, the branching scenarios should be enhanced to enumerate all MAFs more efficiently in the future. In addition, we would like to extend future experiments with this dataset for our clustering-based approach.

### 4.2.2 100% OLGT results

In this section, we present the results gathered after applying our 100% OLGT approach presented in section 3.2.3 on the binary tree tanglegrams dataset [29]. We intended to thoroughly compare all pairs of small trees to examine the effect of tree

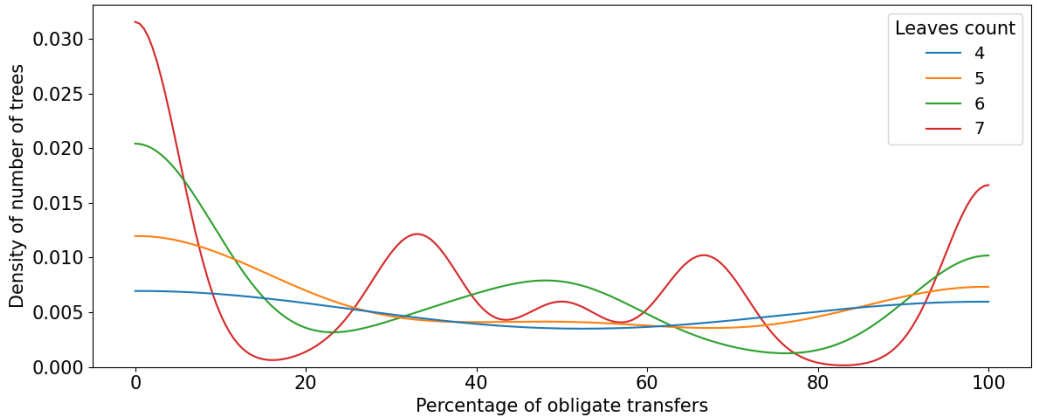



Figure 4.7: Distribution of trees, showing what percentage of the total transfers are obligate for all possible binary tree pairs with 4 to 7 leaves.

structures on the number of obligate LGTs between tree pairs. Some tree pairs have the same AFs because the trees are identical except for relabeling the leaves in both trees in the same way, such as by swapping the labels of two leaves. Tanglegrams generalize this concept to include only one tree pair for each such pair of trees that can be relabeled. This greatly reduces the number of tree pairs that must be evaluated, enabling us to examine all pairs of rooted tanglegrams with 4-7 leaves. This dataset is comparatively small when compared to others that we used in this study, but allows for exhaustively comparing all pairs of small trees to examine the impact of tree structure on LGT analysis. A total of 27231 tree pairs were available for our experiments. The rSPR distance between tree pairs ranged from 0 to 5.

Figure 4.7 shows the result that we obtained regarding the proportion of obligate LGTs among all LGTs between tree pairs. The number of tree density distributions for different leaf counts is shown in the figure. The maximum number of obligate transfers we found for a tree pair was 4, and the rSPR distance between those tree pairs was between 4 and 5. Out of 27231 tree pairs, 5900 tree pairs had all of their transfers OLGT, while 11214 tree pairs had none. We can infer from the graph that a majority of the trees either had no obligate transfers or all of their transfers were obligate. Several tree pairs had multiple obligate transfers, indicating that small tree pairs of random trees frequently contain such transfers. This does not necessarily mean that biological datasets will have obligate transfers, but it does imply that it is something that should be examined.

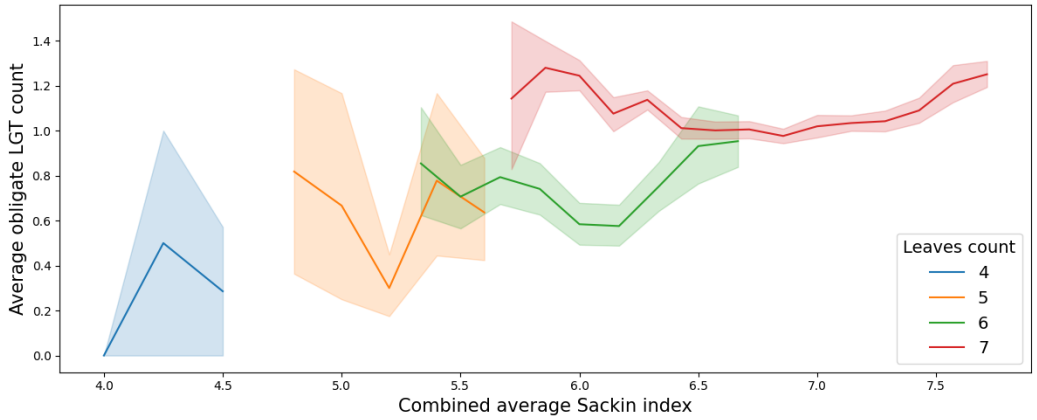



Figure 4.8: Distribution of trees showing how tree structure affects the number of OLGT between tree pairs.

We used the combined average Sackin index method to analyze the relationship between the structure of the tree pair and the number of OLGT. The Sackin index value for the seven-leaf trees ranged from 20 to 24. A more balanced tree is indicated by a lower value, and left- or right-aligned trees are indicated by higher values of the Sackin index. From Figure 4.8, we observe that the average obligate LGT count is higher in trees with a combined average Sackin index that is either too low or too high. Therefore, we can say that generally more balanced or more one-sidedly aligned trees tend to have a higher number of obligate transfers. Although we were unable to determine the exact reason, their tree structure's symmetry may be the reason for this behavior.

### 4.2.3 Clustering results

In this section, we present the results gathered after applying our proposed merging cluster MAFs method in section 3.3 on the 144 prokaryotic genomes dataset [2] containing binary rooted tree pairs. This dataset was chosen for the experiments because our algorithm does not currently support gluing operations for multifurcating trees. The protein tree data set consists of 22,437 trees constructed from a set of proteins from 4 to 144 microbial genomes [2]. We employed our clustering with merging algorithm along with the better approach OLGT algorithm presented in section 3.2.2. To compare the results, we also employed the better OLGT approach without clustering.

Table 4.2 displays the average number of MAFs we enumerated for the dataset

Table 4.2: Clustering OLGT results

| **Optimizations** | **Number of MAFs (average)** |
|---|---|
| Better approach | 14.963104 |
| Better approach (PB case) | 14.919852 |
| Clustering approach | 14.963104 |

while employing the better OLGT approach (with or without the PB case), and while using clustering and gluing with the same approach. As expected, we were able to produce all global MAFs after gluing together the MAFs of individual clusters. The OLGT approach (with the PB case) enumerated slightly fewer MAFs. The number of MAFs varied between 1 to as high as 3645 for a single tree pair. The average number of MAFs between tree pairs may vary in different datasets. As a result, gluing MAFs while using clustering might require more time. However, we observed that using the clustering and gluing approach to enumerate all MAFs is beneficial for this dataset.

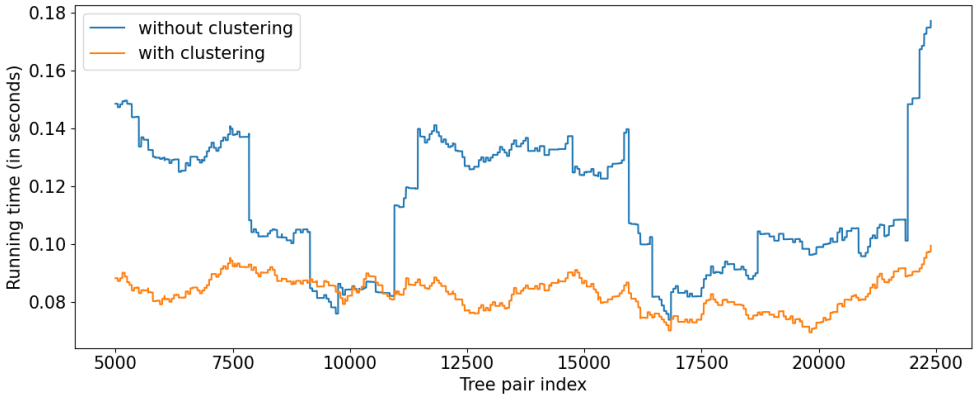



Figure 4.9: Running time comparison for tree pairs of enumerating MAFs with clustering (also gluing) and without clustering.

Our primary objective for this experiment was to analyze the improvements in the running time when using clustering and gluing to find MAFs. For almost all tree pairs, the clustering and gluing approach utilized less time than one without clustering, according to Figure 4.9. We can therefore infer that using clustering and gluing is a better method for enumerating all potential MAFs for the 144 prokaryotic genome dataset. Different datasets may require more time for clustering and gluing depending on the number of clusters and MAFs between each tree pair. When tree pairs have fewer MAFs, we expect to get better timing results compared to the method that does not use clustering.

# Chapter 5

# Conclusion

In this thesis, our primary objective was to develop some techniques for the lateral gene transfer (LGT) analysis of phylogenetic trees and apply them to real-world bacterial datasets to analyze LGT events involving antibiotic resistance. First, we developed a few methods for tracing LGT events involving some children of non-binary nodes of a reference tree. To help others choose methods based on their requirements, we also listed the advantages and disadvantages of each method. These techniques were applied to the *Enterobacteriaceae* dataset and the *Enterococcaceae* dataset for LGT tracing and we were able to achieve the desired outcomes. The Move parent node method resulted in achieving the precise number of transfers. The Move individual node method, however, performed better for the *Enterobacteriaceae* dataset when compared to the *Enterococcaceae* dataset because the number of transfers varied less. Therefore, we suggested employing the Move parent node with maintain list method to obtain an accurate count of transfers with their sources and destinations for the *Enterococcaceae* dataset. In future work, we should apply these methods to synthetic datasets using evolution simulators to analyze how well they recognize transfers.

After that, we shifted our attention to tracing the obligate lateral gene transfers (OLGT) between a pair of phylogenetic trees. Two different strategies were developed for that, one based on creating all MAFs and the other based on edge protection. We used the tangle-grams dataset with 4 to 7 leaves to analyze the edge protection-based approach for tracing OLGTs. We observed that obligate transfers are more common in usually more balanced or one-sidedly aligned trees and overall are common in random tree pairs with a small number of leaves. The MAF-based approach was made possible by advancements in [39, 12] to prevent traversing unnecessary edges of phylogenetic trees in order to enumerate MAFs. The method was employed on the Aquificae dataset and was able to enumerate all MAFs in less amount of time than the naive approach. As the problem of enumerating all MAFs differs from the

work in [12] of identifying the edge set present in all MAFs, we cannot apply all of their optimizations. The running time of our approach depends on the number of enumerated MAFs. Therefore, a part of our future work will be to analyze the running time of this algorithm. Additionally, we like to reduce the running time in the future by enhancing branching scenarios. In addition to that, we intend to apply these methods to bacterial datasets to study LGTs involving antibiotic resistance.

To make improvements in the experimental running time of the algorithm used to enumerate all potential MAFs, we combined it with the clustering strategy mentioned in [26, 12]. We developed a technique to combine MAFs from different clusters to produce global MAFs. We applied this approach to the 144 prokaryotic genomes dataset and noticed a substantial decrease in overall running time. The time needed depends on the rSPR algorithm and the number of MAFs present for a pair of trees, and we need to determine in the future whether gluing that depends on the number of MAFs will eventually require a longer time than the rSPR method or not. As our current method of gluing MAFs only works with binary trees, future work for this will include extending this method to glue MAFs of non-binary trees.

## Bibliography


[1] Benjamin Lang Allen and Mike A. Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of Combinatorics*, 5:1–15, 2001.

[2] Robert G. Beiko, Timothy J. Harlow, and Mark A. Ragan. Highways of gene sharing in prokaryotes. *Proceedings of the National Academy of Sciences*, 102(40):14332–14337, 2005.

[3] Olaf R. P. Bininda-Emonds. The evolution of supertrees. *Trends in Ecology & Evolution*, 19:315–322, 06 2004.

[4] Olaf R. P. Bininda-Emonds. Supertree construction in the genomic age. *Methods in enzymology*, 395:745–57, 2005.

[5] Grace A. Blackwell, Martin Hunt, Kerri M. Malone, Leandro Lima, Gal Horesh, Blaise T. F. Alako, Nicholas R. Thomson, and Zamin Iqbal. Exploring bacterial diversity via a curated and searchable snapshot of archived dna sequences. *PLOS Biology*, 19(11):1–16, 11 2021.

[6] Maria Luisa Bonet, Katherine St. John, Ruchi Mahindru, and Nina Amenta. Approximating subtree distances between phylogenies. *Journal of computational biology: a journal of computational molecular cell biology*, 13 8:1419–34, 2006.

[7] Magnus Bordewich, Catherine McCartin, and Charles Semple. A 3-approximation algorithm for the subtree distance between phylogenies. *J. Discrete Algorithms*, 6:458–471, 2008.

[8] Magnus Bordewich and Charles Semple. On the computational complexity of the rooted subtree prune and regraft distance. *Annals of Combinatorics*, 8:409–423, 2005.

[9] Z. Chen, Y. Harada, Y. Nakamura, and L. Wang. Faster exact computation of rspr distance via better approximation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 17(03):916–929, may 2020.

[10] Zhi-Zhong Chen, Youta Harada, and Lusheng Wang. A new 2-approximation algorithm for rspr distance. In *International Symposium on Bioinformatics Research and Applications*, 2017.

[11] Joshua Stewart Collins. *Rekernelisation algorithms in hybrid phylogenies*. Master's thesis, University of Canterbury, New Zealand, 2009.

[12] Jordan Dempsey. *Common structure among maximum agreement forests of phylogenetic trees*. Master's thesis, Dalhousie University, Halifax, Canada, 08 2021.

[13] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer Publishing Company, Incorporated, 01 2013.

[14] Joseph Felsenstein. Confidence limits on phylogenies: An approach using the bootstrap. *Evolution*, 39, 1985.

[15] Joseph Felsenstein. Phylogenies and the comparative method. *The American Naturalist*, 125(1):1–15, 1985.

[16] N Galtier and M Gouy. Inferring phylogenies from dna sequences of unequal base compositions. *Proceedings of the National Academy of Sciences*, 92(24):11317–11321, 1995.

[17] Nicolas Galtier and Vincent Daubin. Dealing with incongruence in phylogenomic analyses. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 363:4023–9, 10 2008.

[18] Joseph Gillespie, Rebecca Wattam, Stephen Cammer, Joseph Gabbard, Maulik Shukla, Oral Dalay, Timothy Driscoll, Deborah Hix, Shrinivasrao Mane, Chunhong Mao, Eric Nordberg, Mark Scott, Julie Schulman, Eric Snyder, Dan Sullivan, Chunxia Wang, Andrew Warren, Kelly Williams, Tian Xue, and Bruno Sobral. Patric: the comprehensive bacterial bioinformatics resource with a focus on human pathogenic species. *Infection and immunity*, 79:4286–98, 09 2011.

[19] Alan Grafen and William Donald Hamilton. The phylogenetic regression. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 326(1233):119–157, 1989.

[20] Michael Hallett and Catherine Mccartin. A faster fpt algorithm for the maximum agreement forest problem. *Theory of Computing Systems*, 41:539–550, 10 2007.

[21] Jotun Hein, Tao Jiang, Lusheng Wang, and Kaizhong Zhang. On the complexity of comparing evolutionary trees. *Discret. Appl. Math.*, 71:153–169, 1996.

[22] Glenn Hickey, Frank Dehne, Andrew Rau-Chaplin, and Christian Blouin. Spr distance computation for unrooted trees. *Evolutionary Bioinformatics Online*, 4:17 – 27, 2008.

[23] David Hillis, Craig Moritz, and Barbara Mable. Molecular systematics / edited by david m. hillis, craig moritz. *SERBIULA (sistema Librum 2.0)*, 01 1996.

[24] Kartik Kakadiya. rspr fpt software. https://github.com/ktkakadiya/rspr, 2023.

[25] Ben Lee and Chris Whidden. Implementation and optimisations for computing maximum agreement forests for rooted multifurcating trees. *Dalhousie Computer Science In-House*, 2021.

[26] Simone Linz and Charles Semple. A cluster reduction for computing the subtree distance between phylogenies. *Annals of Combinatorics*, 15:465–484, 2011.

[27] Philippe Lopez and Eric Bapteste. Molecular phylogeny: reconstructing the forest. *Comptes Rendus Biologies*, 332(2):171–182, 2009.

[28] Wayne P. Maddison. Reconstructing character evolution on polytomous cladograms. *Cladistics*, 5, 1989.

[29] Erick Matsen. tangle: Version used in rSPR Ricci curvature paper., March 2015.

[30] Samuel I. Miller. Antibiotic resistance and regulation of the gram-negative bacterial outer membrane barrier by host innate immune molecules. *mBio*, 7(5):10.1128/mbio.01541–16, 2016.

[31] Estela Rodrigues, Marie-France Sagot, and Yoshiko Wakabayashi. Some approximation results for the maximum agreement forest problem. In *Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques*, volume 2129, pages 159–169, 01 2001.

[32] Estela M. Rodrigues, Marie-France Sagot, and Yoshiko Wakabayashi. The maximum agreement forest problem: Approximation algorithms and computational experiments. *Theoretical Computer Science*, 374(1):91–110, 2007.

[33] Vania Rosas-Magallanes, Patrick Deschavanne, Lluis Quintana-Murci, Roland Brosch, Brigitte Gicquel, and Olivier Neyrolles. Horizontal Transfer of a Virulence Operon to the Ancestor of Mycobacterium tuberculosis. *Molecular Biology and Evolution*, 23(6):1129–1135, 03 2006.

[34] Frans Schalekamp, Anke van Zuylen, and Suzanne van der Ster. A duality based 2-approximation algorithm for maximum agreement forest. *CoRR*, abs/1511.06000, 2015.

[35] Feng Shi, Jianer Chen, Qilong Feng, and Jianxin Wang. Parameterized algorithms for the maximum agreement forest problem on multiple rooted multifurcating trees. *J. Comput. Syst. Sci.*, 97:28–44, 2016.

[36] Feng Shi, Jianxin Wang, Yufei Yang, Qilong Feng, Weilong Li, and Jianer Chen. A fixed-parameter algorithm for the maximum agreement forest problem on multifurcating trees. *Science China Information Sciences*, 59:1 – 14, 2015.

[37] M. Syvanen. Horizontal gene transfer: Evidence and possible consequences. *Annual Review of Genetics*, 28(1):237–261, 1994.

[38] Vijay V. Vazirani. Approximation algorithms. *Approximation Algorithms*, 2001.

[39] Chris Whidden. *Efficient Computation and Application of Maximum Agreement Forests*. PhD thesis, Dalhousie University, Halifax, Canada, 07 2013.

[40] Chris Whidden, Robert G. Beiko, and Norbert Zeh. Fixed-parameter algorithms for maximum agreement forests. *SIAM J. Comput.*, 42:1431–1466, 2013.

[41] Chris Whidden, Robert G. Beiko, and Norbert Zeh. Fixed-parameter and approximation algorithms for maximum agreement forests of multifurcating trees. *Algorithmica*, 74:1019 – 1054, 2015.

[42] Chris Whidden and Frederick A. Matsen. Calculating the unrooted subtree prune-and-regraft distance. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 16(3):898–911, may 2019.

[43] Chris Whidden and Norbert Zeh. A unifying view on approximation and fpt of agreement forests. In *Workshop on Algorithms in Bioinformatics*, 2009.

[44] Chris Whidden, Norbert Zeh, and Robert G. Beiko. Supertrees based on the subtree prune-and-regraft distance. *Systematic Biology*, 63:566 – 581, 2014.