2017

Synthesizing species trees from gene trees using the parameterized and graph-theoretic approaches

Ju Cheol Moon
Iowa State University

Synthesizing species trees from gene trees using the parameterized and graph-theoretic approaches

by

Ju Cheol Moon

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:

Oliver Eulenstein, Major Professor

Pavan Aduri

Xiaoqiu Huang

Gurpur Prabhu

Giora Slutzki

Iowa State University

Ames, Iowa

2017

DEDICATION

I would like to dedicate this dissertation to my wife Soo Hyun Cho and to our daughter Chloe Moon without whose support I would not have been able to complete this work. I would also like to thank my friends and family for their loving guidance during the writing of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this dissertation. First and foremost, Dr. Oliver Eulenstein for his guidance, patience and support throughout this research and the writing of this dissertation. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee members for their efforts and contributions to this work: Dr. Pavan Aduri, Dr. Xiaoqiu Huang, Dr. Gurpur Prabhu, and Dr. Giora Slutzki.

ABSTRACT

Gene trees describe how parts of the species have evolved over time, and it is assumed that gene trees have evolved along the branches of the species tree. However, some of gene trees are often discordant with the corresponding species tree due to the complicated evolution history of genes. To overcome this obstacle, median problems have emerged as a major tool for synthesizing species trees by reconciling discordance in a given collection of gene trees. Given a collection of gene trees and a cost function, the median problem seeks a tree, called median tree, that minimizes the overall cost to the gene trees. Median tree problems are typically NP-hard, and there is an increased interest in making such median tree problems available for large-scale species tree construction.

In this thesis work, we first show that the gene duplication median tree problem satisfied the weaker version of the Pareto property and propose a parameterized algorithm to solve the gene duplication median tree problem. Second, we design two efficient methods to handle the issues of applying the parameterized algorithm to unrooted gene trees which are sampled from the different species. Third, we introduce the graph-theoretic formulation of the Robinson-Foulds median tree problem and a new tree edit operation. Fourth, we propose a new metric between two phylogenetic trees and examine the statistical properties of the metric. Finally, we propose a new clustering criteria in a bipartite network and propose a new NP-hard problem and its ILP formulation.

CHAPTER 1. INTRODUCTION

Phylogenetic tree inference of increasingly larger and more credible tree estimates is continuously enriching our fundamental knowledge about the evolutionary relationships of how evolutionary entities (e.g., molecular sequences, genomes, and species) have evolved over time. Through these relationships, we are able to understand the general evolutionary principles of how evolutionary entities have evolved the way they are today. While these principles are already greatly benefitting our society and our economy (e.g., in the areas of epidemiology, vaccine development, and conservation biology) [Nik-Zainal et al. (2012); Hufbauer et al. (2003); Roux et al. (2006); Harris et al. (2013); Forster and Renfrew (2006)], they also allow us to predict how evolutionary entities may change in the future. Thus, phylogenetic trees can also be used as powerful predictive tools that are not only of fundamental importance to biology [Dobzhansky (2013)], but are also benefiting numerous applications in other research areas [Jackson (2004); Nik-Zainal et al. (2012)].

Traditional phylogenetic tree inference approaches sample a single gene (gene family) for a set of species and construct the evolutionary history, or *gene tree*, of this gene. It is assumed that the gene tree is mimicking the evolution of the species, and, therefore, is identified with the species tree. However, gene trees for two distinct genes may differ due to complex evolutionary processes that affect the genomic locations differently (e.g., gene duplications, lateral transfer, or deep coalescence) [Page (1998); Cotton and Page (2005)]. Therefore, identifying a gene tree with its species tree can result in largely misleading phylogenetic analyses. Furthermore, a gene might only have been sampled for a small number of species, and a small number of species in the resulting species tree can largely limit its benefit for phylogenetic analyzes [Pamilo and Nei (1988)].

In contrast, today's genomic data sets provide us with an unprecedented wealth of novel evolutionary information for credible phylogenetic studies. Such data sets provide us with thousands of genes (e.g., the human genome has about $21,000$ genes [Ezkurdia et al. (2014); Consortium et al. (2004); Pennisi (2012)]) sampled from across the species of interest, and make it possible to infer large-scale and credible phylogenetic species trees. However, inferring such phylogenetic tree estimates from genomic-sized data sets is one of the most complex and challenging problems in computational phylogenetics.

Median tree problems provide a powerful tool to synthesize large-scale species tree estimates from collections of discordant gene trees [Bininda-Emonds (2004)]. Given a collection of gene trees, *median tree problems* (also referred to as supertree problems [Bininda-Emonds (2004)]) seek a tree, called *median tree*, that is minimizing the overall distance to the input trees using a problem-specific measure. Median tree problems that are typically used in practice are NP-hard, and thus have been addressed by standard local search heuristics, which have produced some credible estimates of species trees [Maddison and Knowles (2006); Than and Nakhleh (2009a)]. However, such heuristics are challenged to find a globally optimal species tree in a highly complex solution landscape whose size is double factorial in the size of the searched tree. In addition, this landscape has typically numerous local optima that can trap heuristic approaches [Bansal and Eulenstein (2013)].

The *gene duplication problem* is a median tree problem that seeks to compute a species tree that is the median tree for a given collection of gene trees under the gene duplication score. The *gene duplication score* is defined to be the minimum number of gene duplications necessary to explain the discordance between a gene tree and a species tree. While the gene duplication score can be computed in linear time [Zhang (1997)], the gene duplication problem is NP-hard [Ma et al. (2000)], $W[2]$-hard when parameterized by the gene duplication score [Bansal and Shamir (2011)], and hard to approximate to better than a logarithmic factor [Bansal and Shamir (2011)]. Therefore, effective local search heuristics for this problem have been proposed [Bansal and Eulenstein (2013); Wehe et al. (2013)], carefully analyzed, and applied to compute credible species trees [Cotton and Page (2002); Martin and Burg (2002); McGowen et al. (2008); Page (2000)]. As a consequence exact solutions using integer linear programming [Chang et al. (2011)]

and dynamic programming [Chang et al. (2013)] have been developed. Such exact algorithms are coming into reach of computing smaller phylogenetic studies, and they have helped resolving an evolutionary conjecture about an empirical data set posed in the literature [Page and Charleston (1997)]. However, exact solutions for large-scale studies remain still out of reach.

Although heuristics for the gene duplication problem have provided promising results, the steady increase of median tree problems has spurred the need to seek justification for any preference. While simulation studies and empirical studies are typically used to provide such preference, they mostly evaluate heuristic results of NP-hard median tree problems rather than the problems themselves [Bininda-Emonds (2004)]. In contrast, showing whether median tree problems satisfy certain theoretical properties allows for an exact evaluation [Wilkinson et al. (2007); Steel et al. (2000)]. Furthermore, such properties can also contribute to the design of more efficient and effective algorithms for median tree problems [Lin et al. (2012a)]. Pareto axioms, which have their origins in social choice theory [Arrow (1952)], offer a basic approach to study theoretical properties of median tree problems. Here, we focus on the desirable Pareto property for clusters. This problem has been extensively studied for several median tree problems [Wilkinson et al. (2007); Lin et al. (2012a)], and, Wilkinson et al. (2007) posed the conjecture that the gene duplication problem is Pareto for clusters.

In Chapter 4, we show that the Wilkinson et al. (2007) conjecture does not generally hold. Despite this negative result, we prove that a slightly modified version of the Pareto property for clusters is satisfied for the gene duplication problem. Besides the usefulness of this property for evolutionary studies, this property also allows us to design an exact polynomial-time algorithm for a fixed parameterization of the gene duplication problem that can be highly beneficial in practice. In an empirical study we demonstrate the ability of our exact algorithm to handle large-scale instances. Using large-scale instances, we analyze the performance of DupTree [Wehe et al. (2008)], a standard heuristic for the gene duplication problem, by comparing the heuristic results against the exact ones computed by our new algorithm.

As the gene duplication problem, median tree problems can be much more tractable and are better understood when their instances are *restricted* to gene trees whose leaf-genes are sampled from the same set of species [Bryant (2003)]. While most median tree problems remain

NP-hard in their restricted version, a class of such restricted problems has been effectively addressed using a parametrized approach, called *Strict Consensus Approach* [Lin et al. (2012a); Moon et al. (2016)]. Most notably, this class includes the classic gene duplication [Moon et al. (2016)], deep coalescence [Lin et al. (2012a)], and Robinson-Foulds [Bryant (2003)] problems. It has been demonstrated that this approach can significantly improve on the scalability of such problems [Lin et al. (2012a); Moon et al. (2016)].

However, the applicability of the Strict Consensus Approach has been severely limited due to its restriction to apply only to rooted gene trees that are sampled from the same set of species, while at the same time most available gene trees are sampled from various distinct sets of species [Bininda-Emonds (2004)] and are unrooted [Górecki and Eulenstein (2012)]. Note that most standard phylogenetic inference methods, like maximum likelihood [Felsenstein (1981)], maximum parsimony [Fitch (1971)], or neighbor joining [Saitou and Nei (1987)], only infer unrooted gene trees, and it is often difficult, if not impossible, to identify credible rootings [Boykin et al. (2010); Burleigh et al. (2011)]. For instance, outgroup rooting can result in incorrect rootings when evolutionary events cause heterogeneity in the gene trees, and molecular clock assumption or midpoint rooting can result in error when there is a molecular rate variation throughout the tree [Holland et al. (2003); Huelsenbeck et al. (2002)].

In Chapter 5 and 6, to overcome this limitation, we devise two efficient methods that extends the Strict Consensus Approach to handle unrestricted and unrooted median tree problems. Using empirical and simulation studies, we demonstrate that our new methods applied to the unrestricted and unrooted gene duplication problem improves significantly on scalability and accuracy when compared to standard heuristic approaches for this problem.

The Robinson-Foulds (RF) distance (or also referred to as symmetric difference) [Robinson and Foulds (1981)] is among the most widely used in comparative phylogenetics. The classic *RF median tree problem* is the median tree problem for the RF distance. This problem is NP-hard [McMorris and Steel (1993)] like most other standard median tree problems [Bininda-Emonds et al. (2002)]. In practice, NP-hard median tree problems have been addressed using standard local search heuristics that search the space of all candidate median trees [Maddison and Knowles (2006); Than and Nakhleh (2009b); Bansal et al. (2010a); Chaudhary et al. (2010);

Lin et al. (2012a)]. Given an initial starting candidate tree, such heuristics search for an optimal candidate tree within the *local neighborhood* of the initial tree that is the set of all trees into which the starting tree can be transformed by at most one tree edit operation. This procedure constitutes a *local search step*. The locally optimal tree found becomes the starting tree for the next local search step, and so on, until a local minima is found.

In Chapter 7, we introduce the first clique-based formulation of the RF median tree problem. This graph-theoretic formulation allowed us to develop a novel clique-based heuristic for the RF median tree problem that operates on a different type of search space than standard median tree heuristics that are searching the space of candidate median trees. Using large-scale published empirical data sets and simulated data, we demonstrate that our clique-based heuristic is able to improve on the best-known RF species tree estimates for the given data sets, when initialized with these trees. Finally, investigating how our clique-based heuristic relates to standard median tree heuristics reveals that it can be interpreted as a local search heuristic for candidate median trees that uses a novel tree edit distance, which we call *m-rooted multiple subtrees prune and regraft (m-rMSPR)*, to define the neighborhood for the local search step. We also show how the m-rMSPR neighborhood relates to the classic local neighborhoods used by standard median tree heuristics.

To compare phylogenetic trees, many pairwise distance measures have been proposed (e.g., Maximum Agreement Subtree (MAST) distance [Finden and Gordon (1985)], nearest-neighbor interchanging (NNI) distance [Waterman and Smith (1978)], subtree pruning and regrafting (SPR) distance [Allen and Steel (2001)], Robinson-Foulds distance [Robinson and Foulds (1981)], and bipartition matching distance [Lin et al. (2012b)]). However, they each have a variety of shortcomings, MAST distance is too stringent [Lin et al. (2012b)]; distance measures based on edit distances under tree edit operations (i.e., NNI or SPR) are NP-Hard [Allen and Steel (2001); DasGupta et al. (1997); Hickey et al. (2008)]; the Robinson-Foulds (RF) distance has a vary skewed distribution [Bryant and Steel (2009); Steel and Penny (1993)], in which most values are close to the maximum; the bipartition matching distance is restricted to unrooted phylogenetic trees [Lin et al. (2012b)].

In Chapter 8, we introduce a new pairwise distance measure for rooted phylogenetic trees and show the new distance measure induces a metric on the space of trees. We also propose how the distance measure can be computed in polynomial time and demonstrate statistical properties using the simulated data sets.

The median tree approaches are susceptible to the collection of gene trees. While the gene trees may agree strongly on the structure relating a large subset of the species, the remaining few species can effectively prevent this underlying structure [Pattengale et al. (2011)]. A species genome has multiple genes, intuitively, this information forms a bipartite network where nodes in one partition represent species, nodes in the other represent genes, and an edge represents the containment between a species and a gene. By finding dense subgraph in the network, closely related spices and genes can be identified.

The *highly connected subgraph* algorithm has been used to identify dense components in biological networks [Hartuv et al. (2000)]. An undirected graph with n vertices is highly connected if it can only be disconnected by removing more than $\frac{n}{2}$ of its edges. While this approach has been utilized to a normal graph [Sharan et al. (2007)], it cannot be applied to the large class of biological networks that is represented by using bipartite graphs. This is due to the fact that highly connected subgraphs do not exist in bipartite graphs.

In Chapter 9, we overcome this stringent limitation by proposing a natural adaptation of the definition for highly connected subgraphs to bipartite graphs. A bipartite graph $G = (U, V, E)$ is *highly bi-connected* if more than $\frac{1}{2}\min(|U|, |V|)$ of its edges are required to disconnect it. To identify useful highly connected subgraphs in bipartite biological networks, we analyze the *highly bi-connected (HBC)* problem that given a bipartite graph and a natural number k, decides whether this graph contains a highly bi-connected subgraph with k vertices. We show that this problem, like its related problem for identifying highly connected subgraphs [Hüffner et al. (2015)], is NP-Hard. Consequently, to address the HBC problem, we describe an *integer linear programming (ILP)* formulation and a heuristic algorithm that can handle large-scale instances. We also demonstrate the performance of our heuristic through a comparative study using exact ILP solutions and an applicability study for to protein function annotation.

CHAPTER 2. REVIEW OF THE LITERATURE

The pioneering work of Goodman et al. (1979) introduced the gene duplication problem under the assumption that all trees involved are rooted and full binary. This problem is a median tree problem under the gene duplication score that is defined for a gene tree and a species tree. The leaves of the species tree are uniquely labeled by species and we identify the leaves with their labels. The leaves of the gene tree are mapped to the leaves of the species tree through a one-to-one function called *leaf-labeling*. While such a function always exists, it is assumed that this function maps the genes to the species from which they were sampled. The leaf-labeling can be extended to a function called *least common ancestor (LCA) mapping*. This mapping relates each gene in the gene tree to the most recent species in the species tree that could have contained this gene. A gene in the gene tree is a called a *gene duplication* if it has a child with the same LCA mapping. The *gene duplication score* between a gene tree and a species tree for a given leaf labeling is the number of gene duplications.

A basic approach for evaluating median tree problems are their Pareto properties [Arrow (1952); Wilkinson et al. (2007); Lin et al. (2012a)]. A median tree problem is *Pareto* when the elementary evolutionary information that is commonly described by every input tree of each of its problem instances is also described by the corresponding median tree. Pareto properties are distinguished by the type of elementary information that is used, such as triplets, nestings, or clusters [Wilkinson et al. (2007)]. In this thesis, we are interested in the Pareto property for clusters. A *cluster* for a vertex in a tree is the set of all labels of the leaves of the subtree rooted at this vertex. The *cluster representation* of a tree is the set containing a cluster for each of the vertices in the tree. A tree and its cluster representation are equivalent representations of each other [Semple and Steel (2003a)]. The use of clusters as elementary evolutionary information that is common to the input trees requires that problem instances consist only of trees with

the same taxon set, which are called *consensus instances*. A median tree problem is *Pareto for clusters* if the consensus clusters for each of its consensus instances is contained in every corresponding solution, i.e. a median tree. While the Pareto for clusters property is restricted to make a statement only about the consensus instances, it still provides valuable information about median tree problems [Wilkinson et al. (2007); Lin et al. (2012a)]. This holds in particular true for applications of medium tree problems that only use consensus instances, such as the maximum parsimony method that typically produces several parsimony trees over the same taxon set which are represented by a median tree for evolutionary studies [Hedges et al. (1992)]. While it is known for various median tree problems whether they are Pareto for clusters, this is still unknown for the classical gene duplication problem.

The Strict Consensus Approach has been introduced in [Lin et al. (2012a)], which is applicable to a class of restricted median tree problems (i.e., problems where the input trees are restricted to a same leaf set) that satisfy the Pareto (for clusters) property [Wilkinson et al. (2007)] and a substructure property [Lin et al. (2012a)]. The Pareto property has its roots in social choice theory [Arrow (1952)], and various restricted median tree problems have been identified to satisfy this property [Wilkinson et al. (2007)].

The set of clusters common to a collection $\mathcal{T}$ of restricted trees define the *strict consensus clusters* of $\mathcal{T}$ [Semple and Steel (2003b)]. Strict consensus clusters form a rooted tree under the transitive reduction of the set-containment relationship between clusters, called the *strict consensus tree* of $\mathcal{T}$ [Semple and Steel (2003b)]. When a collection of restricted trees that are rooted have clusters in common (i.e., the *strict consensus clusters* of these trees), then one would expect that an optimal species tree for these trees under any "reasonable" median tree problem would also contain these clusters. The Pareto property is formalizing this idea.

For a restricted median tree problem, when the strict consensus clusters of each instance are contained in the every corresponding solution, then this problem satisfies the *Pareto* property [Wilkinson et al. (2007)]. Several restricted median tree problems are known to satisfy the Pareto property, including the deep coalescence problem [Lin et al. (2012a)] and the Robinson-Foulds problem [Bryant (2003)]. The Pareto property is not only valuable to practitioners for deciding which median tree problem might be most suitable for their analyses, but also, can lead

to formal characterizations that reduce the complexity of such problems [Lin et al. (2012a)]. Fig. 2.1 depicts an example, where the strict consensus tree of the input trees $T_1, \ldots, T_k$ is the tree G, which includes the multifurcation consisting of the *parent cluster* C_p and its *child clusters* $C_1, \ldots, C_l$. The optimal species tree, the tree S in the figure, refines this multifurcation based on the objective function of the applied median tree problem.

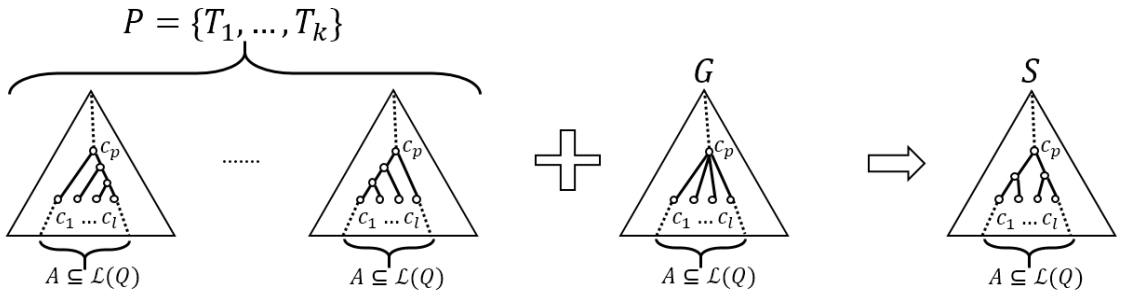

Figure 2.1: An illustration of the Strict Consensus Approach.
$P = \{T_1, \cdots, T_k\}$: input trees, G: the strict consensus tree of the input trees.
S: the optimal species tree (a refinement of G).
$C_p, C_1, \cdots, C_l$: the common clusters of the input trees.

The Strict Consensus Approach refines all of the multifurcations of the strict consensus tree to reach an optimal solution. Each multifurcation is refined by solving a sub-instance of the original problem instance. Given a multifurcation, expressed as a parent and its child clusters, the corresponding *sub-instance* is computed by *trimming* the trees of the original instance to contain only clusters that are on a path between the parent cluster and its child clusters. Fig. 2.1 illustrates such a trimming for the parent cluster C_p and its child clusters $C_1, \ldots, C_l$, where the resulting sub-instance is depicted as a collection of subtrees of the input trees $T_1, \ldots, T_k$. Lastly, the refinements for each multifurcation of the strict consensus tree have to result in a solution of the corresponding restricted median tree problem, which is warranted if this problem is satisfying the substructure property.

The pioneering work from Robinson and Foulds [Robinson and Foulds (1981)] has introduced the widely-used Robinson-Foulds (RF) distance (which is also called symmetric difference) for two trees. For the purpose of this work the trees involved are rooted and full binary. The *RF distance* between two trees over the same taxon set is the normalized count of the cardinality between the symmetric difference of the cluster representations of these trees. A *cluster* for a

vertex in a tree is the set of all labels of the leaves of the subtree rooted at this vertex. The *cluster representation* of a tree is the set containing a cluster for each of the vertices in the tree. A tree and its cluster representation are equivalent representations of each other [Semple and Steel (2003b)].

To evaluate the overall RF distance of a candidate median tree to its input trees, it might be necessary to compute this distance between two trees where the taxa set of one tree (i.e., an input tree) is a proper subset of the taxa set of the other tree (i.e., the candidate median tree). While the RF distance is not defined for such trees, this is a common problem that occurs for various other distances for median tree problems. A standard approach to address this problem is using the *minus method* [Wilkinson et al. (2007)] that restricts the larger of the two trees to only the taxa set of the smaller tree before computing the RF distance. Computing the RF distance between two trees over the same taxa has only a linear time complexity in the number of nodes in the trees involved [Day (1985)]. A randomized algorithm that runs in sublinear time has been shown to approximate the RF distance with a bounded error [Pattengale et al. (2007)].

Clique-based formulations have been proposed and studied for three median tree problems, which are the deep coalescence, gene duplication, and gene duplication-loss problems. However, these clique-based formulations have not been used to develop heuristics for the corresponding problems applicable to large-scale instances.

Than and Nakhleh [Than and Nakhleh (2009b)] showed that the deep coalescence problem for n species could be solved by exploring minimum weight $(n-1)$-cliques (i.e., cliques with $n-1$ vertices) in the vertex-weighted *compatibility graph* of all possible clusters over n species. A *cluster* over a set of n species is any non-empty subset of this set. The clusters with more than one species represent the vertices of the compatibility graph, and an edge is drawn between two vertices if they are compatible. Two clusters are *compatible* if and only if they are disjoint or one cluster contains the other one. It is known that a species tree over n taxa exists if $n-1$ clusters with more than one species are pairwise compatible [Semple and Steel (2003b)], and therefore form a clique in the compatibility graph. As a result, the deep coalescence problem can be solved by finding a minimum weight $(n-1)$-clique in the compatibility graph where

weights are assigned to the vertices based on their contribution of deep coalescence events to reconcile the input gene trees. The clique-based formulation of the deep coalescence problem was used to describe an ILP formulation of this problem, which then allowed to solve small instances of up to eight species exactly.

Later, Bayzid et al. (2013) showed that the gene duplication and gene duplication-loss problem were also solvable by using a clique-based approach that is similar to the one introduced by Than and Nakleh, using a different type of compatibility graph and weighting functions. Due to the different properties of the weighting functions, the gene duplication problem and the duplication-loss problem were solved by finding a maximum weight $(n-1)$ clique and a minimum weight $(n-1)$ clique in the compatibility graph respectively.

The original work from Lin et al. (2012b) has proposed the (bipartition) matching distance. Similar to the clusters in a rooted tree, every internal edge e in an unrooted tree T defines a nontrivial bipartition σ_e on the leaves, and the tree T is uniquely represented by the set of bipartitions $\Sigma(T) = \{\sigma_e | e \in E(T)\}$ where $E(T)$ is the set of internal edges in T. A bipartition can be represented by the binary vector, i.e., if $\sigma = (1, \cdots, k|k+1, \cdots n)$, then the corresponding binary representation is $[1, \cdots, 1, 0, \cdots, 0]$ where the number of 1's is equal to k. Given two trees, T_1 and T_2 on the same set of leaves, a complete weighted bipartite graph $G(X, Y, E)$ with $X = \Sigma(T_1)$ and $Y = \Sigma(T_2)$ is denoted by $B(T_1, T_2)$. The weight of each edge $e = \{u, v\}$ in $B(T_1, T_2)$ is set to $W(u, v) = \min\{HD(V_u, V_v), HD(V_u, \overline{V_v})\}$ where V_u and V_v are the two binary vector representations of the bipartition u and v, $\overline{V}$ is the complement vector representation of V, and HD is the Hamming distance. The bipartition matching distance $BM(T_1, T_2)$ between trees T_1 and T_2 is the weight of the minimum-weight perfect matching in $B(T_1, T_2)$ with the weighting scheme W. The bipartition matching distance can be computed between rooted trees by suppressing the roots of the trees, however, this becomes ignoring the imperative feature of rooted trees.

A key idea of *graph clustering* is to identify densely connected subgraphs as clusters that have many interactions within themselves and few interactions outside of themselves in the graph [Hüffner et al. (2014)]. A highly connected subgraph is defined as a subgraph with n vertices such that more than $\frac{n}{2}$ of its edges must be removed in order to disconnect the subgraph.

The concept of a highly connected graph is very similar to that of a *quasi-clique* (i.e., a graph where every vertex has a degree at least $\frac{n-1}{2}$ [Hüffner et al. (2014)]). Hartuv and Shamir [Hartuv and Shamir (2000)] proved that the HCS algorithm, which is based on the $\frac{n}{2}$ connectivity requirement, produces clusters with good homogeneity and separation properties [Pržulj et al. (2004)]. However, the concept of highly connected subgraphs is not applicable to bipartite graphs, since they do not contain such subgraphs.

CHAPTER 3. PRELIMINARIES

A *graph* G is an ordered pair (V, E) consisting of a non-empty set V of *nodes* and a set E of *edges*. We denote the set of nodes and edges of G by $V(G)$ and $E(G)$, respectively.

If $e = \{x, y\}$ is an edge of a graph G, then e is said to be *incident* with x and y. Two nodes x, y of G are *adjacent*, or *neighbor*, if $\{x, y\}$ is an edge of G. The set of neighbors of a node x in G is denoted by $N_G(x)$, or briefly $N(x)$. More generally for $U \subseteq V$, the neighbors in $V \setminus U$ of nodes in U are called *neighbors* of U; their set is denoted by $N(U)$.

If x is a node of a graph G, then the *degree* of x in G is the number of edges in G that are incident with x. The degree of a node x is denoted by $d_G(x)$, or briefly $d(x)$. The *degree of x in $U \subseteq V$* is the number of nodes in U that are adjacent to x, denoted by $d(x, U)$. The number $\delta(G) \coloneqq \min\{d(x) | x \in V\}$ is the *minimum degree* of G.

A *path* is a sequence of edges which connect a sequence of nodes that are all distinct from one another. The *path length* $pl_G(x, y)$ of two nodes x, y is the number of edges in a shortest path from x to y in G; if no such path exists, we set $pl_G(x, y) \coloneqq \infty$. If $|V| > 1$ and $G' = (V, E \setminus F)$ is connected for every set $F \subseteq E$ of fewer than l edges, then G is called *l-edge-connected*. The greatest integer l such that G is the l-edge-connected is *edge-connectivity* $\lambda(G)$ of G. For every non-trivial graph G, we have $\delta(G) \geq \lambda(G)$.

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. If $V' \subseteq V$, $E' \subseteq E$, and G' contains all the edges $\{x, y\} \in E$ with $x, y \in V'$, then G' is an induced subgraph of G and denoted $G' \coloneqq G[V']$. A graph $G = (X \cup Y, E)$ is called *bipartite* if $V = X \cup Y$ admits a partition into two disjoint subsets X and Y such that every edge connects a node in X to one in Y.

A *unrooted tree* T is an acyclic, connected, and undirected graph. The degree one nodes are called *leaves* and the remaining nodes are called *internal* nodes. The set of leaves in T is denoted by $\mathcal{L}(T)$, and the set of all internal nodes in T is denoted by $V_{int}(T)$. A unrooted

tree T is *binary* if every node has degree one or three. A *rooted tree* T is defined similar to an unrooted tree except that it has one distinguished node that leads a parent-child relationship on the nodes, called the *root* of T, denoted by $r(T)$. A rooted tree T is *binary* if every node has degree one, two or three.

Let $X \subseteq \mathcal{L}(T)$, we write $\overline{X}$ to denote the *leaf complement* of X where $\overline{X} = \mathcal{L}(T) \setminus X$. The *subtree* of T induced by X, denoted by $T(X)$, is the minimal connected subtree of T that contains X. The *restricted subtree* of T induced by X, denoted by $T|X$ is the tree obtained from $T(X)$ by suppressing all nodes of degree two with the exception of the root for a rooted tree.

Let T be an unrooted tree. A *split* is an unordered bipartition of sets. The split in which parts are A and B is denoted by $A|B$. T *displays* a split $A|B$ if there is an edge in T whose removal gives trees T_1 and T_2 such that $A \subseteq \mathcal{L}(T_1)$ and $B \subseteq \mathcal{L}(T_2)$. A split $A|B$ is *full* if $A \cup B = \mathcal{L}(T)$. The set of all full splits displayed by T is denoted $\Sigma(T)$. Let $X \subseteq \mathcal{L}(T)$. The restriction of a split $\sigma \cap X$ is defined as $A \cap X|B \cap X$, $\Sigma(T|X) \coloneqq \{x \cap X : \forall \sigma \in \Sigma(T) \wedge A \cap X \neq \emptyset \neq B \cap X\}$. A split $A|B$ is nontrivial if each of A and B has at least two elements; otherwise it is trivial.

Let T be a rooted tree. We define $\leq_T$ to be the partial order on $V(T)$, where $x \leq_T y$ if y is a node on the path between $r(T)$ and x. If $x \leq_T y$, we call x a *descendant* of y, and y an *ancestor* of x. We also define $x <_T y$ if $x \leq_T y$ and $x \neq y$, in this case we call x a *proper descendant* of y. If $\{x, y\} \in E(T)$ and $x \leq_T y$, then we call y the *parent* of x, denoted by $Pa_T(x)$, and x a *child* of y. The set of all children of y is denoted by $Ch_T(y)$. Let T be a rooted tree. The *subtree* of T rooted at $x \in V(T)$, denoted by $T(x)$, is the restricted subtree induced by $\{y \in V(T) \colon y \leq_T x\}$.

The *cluster* of x is defined by $\mathcal{C}_T(x) \coloneqq \mathcal{L}(T(x))$, and the set of all clusters of T is defined by $\mathcal{H}(T) = \bigcup_{y \in V(T)} \mathcal{C}_T(y)$. $X \in \mathcal{H}(T)$ is called a *trivial* cluster if $X = \mathcal{L}(T)$ or $|X| = 1$, it is called *non-trivial* otherwise. The *least common ancestor (LCA)* of $X \subseteq \mathcal{L}(T)$, denoted by $lca_T(X)$, is the unique smallest upper bound of X under $\leq_T$. Let T' be a rooted tree where $\mathcal{L}(T') \subseteq \mathcal{L}(T)$, we define *LCA mapping* $\mathcal{M} : V(T') \to V(T)$ by $\mathcal{M}_{T',T}(x) \coloneqq lca_T(\mathcal{C}_{T'}(x))$, or briefly $\mathcal{M}$ when it is clear.

Let T and T' be trees with the same leaves. We define $\leq$ to be the partial order on trees where $T \leq T'$ if $\Sigma(T) \subseteq \Sigma(T')$ for unrooted trees and $\mathcal{H}(T) \subseteq \mathcal{H}(T')$ for rooted trees. We say T' *refines* T if $T \leq T'$. Let $\mathcal{L}(t) \subseteq \mathcal{L}(T)$. A tree T *displays* tree t if $t \leq T|\mathcal{L}(t)$.

Let $\pi_i = (A_i, B_i)$ and $\pi_j = (A_j, B_j)$ be unordered pairs. We say that π_i *contains* π_j if $A_j \cup B_j \subseteq A_i$ or $A_j \cup B_j \subseteq B_i$ and that π_i and π_j are *disjoint* if $(A_i \cup B_i) \cap (A_j \cup B_j) = \emptyset$. We also say that p_j is *includes* p_i if $A_i \cup B_i \nsubseteq A_j$, $A_i \cup B_i \nsubseteq B_j$, and $(A_i \cup B_i) \subsetneq (A_j \cup B_j)$. Let T be a rooted binary tree. An unordered pair $\pi = (A, B)$ is called a *rooted split* on $\mathcal{L}(T)$ if $A \cup B \subseteq \mathcal{L}(T)$. An internal node determines a rooted split. A rooted split of a node $v \in V_{int}(T)$, denoted by $\pi(v)$, is the unordered pair $(\mathcal{C}_T(l), \mathcal{C}_T(r))$ where l and r are two children of v. Let $\Gamma(T)$ be a set of rooted splits of the internal nodes in T, then $\Gamma(T) = \bigcup_{v \in V_{int}(T)} \pi(v)$.

A *profile* is a tuple of trees. Let $P = (t_1, \ldots, t_k)$ be a profile, then its *leaves* is $\mathcal{L}(P) \coloneqq \bigcup_i^k \mathcal{L}(t_i)$, and its *nodes cardinality* is $|P| \coloneqq \sum_{i=1}^k |V(t_i)|$. A tree T is a *supertree* for P if $\mathcal{L}(T) = \mathcal{L}(P)$.

A tree T has the *non-contradiction* property if $T|\mathcal{L}(t_i) \leq t_i$ $(\forall i \in \{1, \cdots, k\})$. A tree T is called *guidance tree* for P if T is a supertree for P and T has the non-contradiction property. A profile is called rooted (unrooted) if all trees in the profile are rooted (unrooted).

For each tree t in P we define its *span* $\langle t \rangle$, to be the set of all trees on $\mathcal{L}(P)$ displaying t. A *restricted span* $\langle t \rangle_G$ is the set of all trees on $\mathcal{L}(P)$ that display t and refine G. The *span* (*restricted span*) of P, denoted by $\langle P \rangle$ ($\langle P \rangle_G$), is the set of all profiles $Q = (T_1, \ldots, T_k)$ where $T_i \in \langle t_i \rangle$ ($T_i \in \langle t_i \rangle_G$), $\forall i \in \{1, \cdots, k\}$. The *strict consensus splits (clusters)* of Q are the splits (clusters) common to the trees in Q, and the *strict consensus tree* of Q, denoted by $\mathcal{SC}(Q)$, is the tree that is defined exactly by these common splits (clusters).

CHAPTER 4. STRICT CONSENSUS APPROACH

We show that the gene duplication problem is not Pareto for clusters by counterexample. However, we introduce a slightly weaker version of the Pareto for clusters property, which we call weak Pareto for clusters that, as we prove, is satisfied by the gene duplication problem. A median tree problem is *weak Pareto for clusters* if the consensus clusters for each of its instances over the same taxon set are contained in at least one corresponding solution (rather than all solutions). In contrast to the standard Pareto property, the weak Pareto property allows to give a preference to non-unique solutions of median tree problems that contain the strict consensus clusters. Furthermore, using the weak Pareto property we devise an efficient fixed parameterized algorithm (Strict Consensus Approach) for the gene duplication problem when reduced to consensus instances, where the parameter is the maximum number of children of the strict consensus tree of the input trees. Note that this reduced variant of the gene duplication problem is still NP-hard [Zhang (1997)].

In an empirical study we demonstrate that our exact algorithm can compute solutions for large-scale instances where standard heuristics, implemented in DupTree, fail to terminate within reasonable time. Our exact parameterized algorithm allows, for the first time, to analyze the effectiveness and efficiency of DupTree on large-scale instances. We provide such an analysis using a comparative simulated study.

4.1 Pareto Properties of the Gene Duplication Problem

Let $f\colon \mathcal{T}_X \times \mathcal{T}_X \to \mathbb{R}$ be a score function where X is a leaf set and $\mathcal{T}_X$ is the set of all trees over X. A *median tree problem* based on f is defined as follows.

Definition 4.1.1 (Median Tree Problem)**.**

Instance: *A rooted profile* $P = \{t_1, \cdots, t_n\}$ *such that* $\mathcal{L}(t_i) \subseteq \mathcal{L}(P)$.

Find: *The set of all trees that have the minimum aggregated score with respect to* f. *Formally,* $\operatorname{argmin}_{S \in \mathcal{T}_{\mathcal{L}(P)}} \sum_{i=1}^{n} f\big(T_i, S|\mathcal{L}(t_i)\big)$.

If $\mathcal{L}(t_i) = \mathcal{L}(P)$ $(\forall i \in \{1, \cdots, n\})$, then a median tree problem is also called a *consensus tree problem*. Throughout this chapter, we focus on a consensus tree problem and use a notation T instead of t, in general, as denoting a tree such that $\mathcal{L}(T) = \mathcal{L}(P)$.

Definition 4.1.2 (Pareto for Clusters)**.** *We say that a consensus tree problem satisfies* Pareto for clusters *[Lin et al. (2012a)] if for all profiles* $P = (T_1, \ldots, T_n)$ *and for all solutions* S *of* P, *we have* $\bigcap_{i=1}^{n} \mathcal{H}(T_i) \subseteq \mathcal{H}(S)$. *In addition, we say that a consensus tree problem satisfies* weak Pareto for clusters *if for all instances* $P = (T_1, \ldots, T_n)$, *there exists solution(s)* S *of* P *such that* $\bigcap_{i=1}^{n} \mathcal{H}(T_i) \subseteq \mathcal{H}(S)$.

Definition 4.1.3 (Duplication and Duplication Score)**.** *Let* T *be a gene tree and* S *be a species tree over the same leaf set,* $v \in V(T)$, *and* $\mathcal{M}\colon V(T) \to V(S)$ *be a LCA mapping. We say that* v *is a* gene duplication *if* $\mathcal{M}(v) \in \{\mathcal{M}(c) : c \in Ch_T(v)\}$. *The* gene duplication score *from* T *to* S, *denoted* $GD(T, S)$, *is the cardinality of the set* $\{v \in V(T) : v \text{ is a duplication}\}$. *See example Figure 4.1. Further, if* $P = (T_1, \ldots, T_n)$ *is a profile of gene trees, then we define* $GD(P, S) = \sum_{i=1}^{n} GD(T_i, S)$.

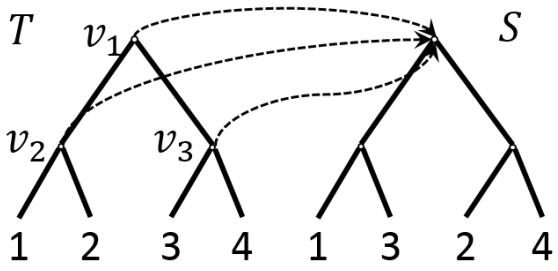

Figure 4.1: The LCA mapping from T to S is shown by dashed arrows. v_1 has the same LCA mapping with its children v_2 and v_3. v_1 is the only duplication in $V(T)$, thus $GD(T, S) = 1$.

Problem 4.1.1 (Gene Duplication Problem)**.** *We define the* gene duplication problem *to be the consensus tree problem based on the gene duplication score function.*

Theorem 4.1.1. *The gene duplication problem is not Pareto for clusters.*

Proof. Consider the profile of two gene trees $P = (T_1, T_2)$ and a candidate species tree S as shown in Figure. 4.2. We will show that S is a solution for P. We observe that $GD(T_1, T_2) = GD(T_2, T_1) = 3$, while $GD(T_1, S) = GD(T_2, S) = 1$. Therefore, neither T_1 nor T_2 is a solution. Since the gene duplication score from a tree to a different tree must be greater than 0, we know that S achieves the minimum score of $GD(P, S) = 2$. However, we observe that the two consensus clusters of P, $\{1, 2\}$ and $\{5, 6\}$, are not contained in S. □

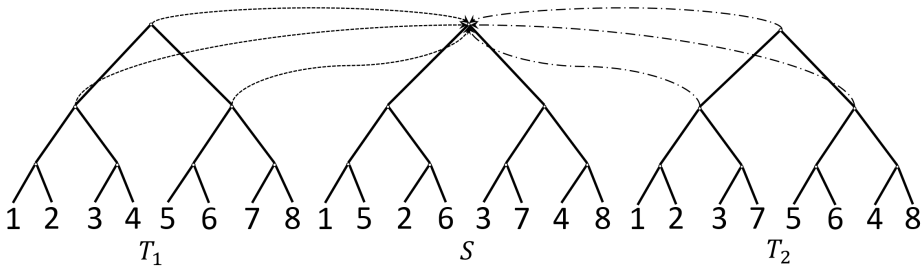

Figure 4.2: A counter example showing that the gene duplication problem is not Pareto for clusters (Theorem 4.1.1). The species tree S is a solution for the input profile (T_1, T_2) having a total gene duplication score of 2, but it lacks the consensus clusters $\{1, 2\}$ and $\{5, 6\}$.

Corollary 4.1.1. *Let X, Y be disjoint non-empty leaf sets. If T be a tree over $X \cup Y$ and $T' = T|X$, then $lca_T(A) \neq lca_T(B) \iff lca_{T'}(A) \neq lca_{T'}(B)$ $(\forall\ A, B\ s.t.\ \emptyset \subset A, B \subseteq X)$.*

Theorem 4.1.2. *The gene duplication problem is weak Pareto for clusters.*

Proof. Given a profile $P = (T_1, \ldots, T_n)$, let S be a solution for P, and a cluster $X \subseteq \mathcal{L}(P)$ where $X \in \bigcap_{i=1}^{n} \mathcal{H}(T_i)$. If $X \in \mathcal{H}(S)$, then since X is arbitrary we have P as a witness instance that is weak Pareto for clusters. Otherwise, we have $X \notin \mathcal{H}(S)$. In this case the proof proceeds by directly transforming S into a new tree R that contains X by not increasing the total number of gene duplications.

Let $v = lca_S(X)$. Since $X \notin \mathcal{H}(S)$, $S(v)$ must contain some leaves other than X. Therefore, let $Y = \mathcal{L}(S(v)) \backslash X$. We construct a new tree R from S by replacing the children of v by $S|X$ and $S|Y$. R clearly contains X because $\mathcal{L}(S|X) = X$. We now show that $GD(T, R) \leq GD(T, S)$ for every $T \in P$. However, we will show a stronger result where for all $T \in P$ and for all

edges $\{c, p\} \in E(T)$, we have $\mathcal{M}_{T,S}(c) \neq \mathcal{M}_{T,S}(p) \Rightarrow \mathcal{M}_{T,R}(c) \neq \mathcal{M}_{T,R}(p)$. Let $T \in P$ and $u = lca_T(X)$. Take any edge $\{c, p\}$ in T, and suppose $\mathcal{M}_{T,S}(c) \neq \mathcal{M}_{T,S}(p)$, we show that $\mathcal{M}_{T,R}(c) \neq \mathcal{M}_{T,R}(p)$. We consider different cases as follows:

1. $p \leq u$: Let $A = \mathcal{L}(T(c))$ and $B = \mathcal{L}(T(p))$, then we know $A \subset B \subseteq X$ because $u = lca_T(X)$. Since $\mathcal{M}_{T,S}(c) \neq \mathcal{M}_{T,S}(p)$, by definition we have $lca_S(A) \neq lca_S(B)$. Let $G = S(v)$ and $G' = G|X$, since $S(v)$ is a subtree of S we have $lca_G(A) \neq lca_G(B)$. Now we apply Corollary 4.1.1 and obtain $lca_{G'}(A) \neq lca_{G'}(B)$. Since G' is a subtree of R we have $lca_R(A) \neq lca_R(B)$, hence $\mathcal{M}_{T,R}(c) \neq \mathcal{M}_{T,R}(p)$ as required.

2. $u \leq c$: Then $v = lca_S(X) \leq \mathcal{M}_{T,S}(c)$ because $u = lca_T(X)$. Since S and R only differs by the subtree $S(v)$, we know that $\mathcal{M}_{T,S}(c) = \mathcal{M}_{T,R}(c)$ and $\mathcal{M}_{T,S}(p) = \mathcal{M}_{T,R}(p)$. Hence $\mathcal{M}_{T,R}(c) \neq \mathcal{M}_{T,R}(p)$.

3. Otherwise ($c \not\leq u$ and $u \not\leq p$)

   (a) $\mathcal{L}(T(c)) \subseteq Y$: Let $A = \mathcal{L}(T(c))$ and $B = \mathcal{L}(T(p))$. If $\mathcal{L}(T(p)) \subseteq Y$ then $A \subset B \subseteq Y$, similar to case 1, we apply Corollary 4.1.1 and obtain $lca_R(A) \neq lca_R(B)$. Otherwise, $\mathcal{L}(S(v)) = X \cup Y$, we know that $\mathcal{M}_{T,R}(c) \leq v$ but $\mathcal{M}_{T,R}(p) \not\leq v$. In both cases, we conclude that $\mathcal{M}_{T,R}(c) \neq \mathcal{M}T, R(p)$ as required.

   (b) $\mathcal{L}(T(c)) \not\subseteq Y$: Then $\mathcal{M}_{T,S}(c)$ must not be part of $S(v)$ since $\mathcal{L}(S(v)) = X \cup Y$. Therefore, similar to case 2, we have $\mathcal{M}_{T,S}(c) = \mathcal{M}_{T,R}(c)$ and $\mathcal{M}_{T,S}(p) = \mathcal{M}_{T,R}(p)$, hence $\mathcal{M}_{T,R}(c) \neq \mathcal{M}_{T,R}(p)$.

□

We introduce Algorithm 1 that converts a given solution that is not Pareto for clusters to the one that is.

Proposition 4.1.1. *Algorithm 1 is correct and runs in $\mathcal{O}(k^2)$, where $k = |\mathcal{L}(S)|$.*

Proof. The correctness follows directly from Theorem 4.1.2. For the runtime, the while loop performs in $\mathcal{O}(k)$ time, since $|\boldsymbol{X}| \leq |\mathcal{L}(S)|$. The time required by the loop body is asymptotically

Algorithm 1 Pareto Solution(P, S)

Input: A profile $P = (T_1, \ldots, T_n)$ and a species tree S such that $\bigcap_{i=1}^{n} \mathcal{H}(T_i) \nsubseteq \mathcal{H}(S)$.
Output: A species tree R such that $\bigcap_{i=1}^{n} \mathcal{H}(T_i) \subseteq \mathcal{H}(R)$.

```
R = S, X = ⋂_{i=1}^{n} H(T_i) \ H(R)
while |X| > 0 do
    X ∈ X where |X| is minimum
    v = lca_R(X), Y = L(R(v)) \ X
    Replace the children of v by R|X and R|Y
    X = ⋂_{i=1}^{n} H(T_i) \ H(R)
end while
return R
```

bound from above by the runtime of the *set minus* operation, which is $\mathcal{O}(k)$. This results in a runtime of $\mathcal{O}(k^2)$ as desired. □

4.2 Strict Consensus Approach

Definition 4.2.1 (Cut on trees)**.** *Let H and T be two trees over the same leaf set such that T refines H. Given an internal node h in H, a cut on T via H and h, denoted $Cut_{H,h}(T)$, is the minimal connected subtree of T that contains $\{\mathcal{M}_{H,T}(c) : c \in Ch_H(h)\}$, and we rename each leaf x by $\mathcal{L}(T(x))$. We further extend this to a profile of trees $P = (T_1, \ldots, T_n)$ by $Cut_{H,h}(P) \triangleq (Cut_{H,h}(T_1), \cdots, Cut_{H,h}(T_n))$.*

Theorem 4.2.1. *Let $P = (T_1, \cdots, T_n)$ be a profile of the gene duplication problem, H be the strict consensus tree of P, and h be an internal node in $V(H)$. If S is a solution for the profile P such that S refines H, then $Cut_{H,h}(S)$ is a solution for the instance $Cut_{H,h}(P)$.*

Proof. Let $S_h = Cut_{H,h}(S)$ and $P_h = Cut_{H,h}(P) = (Cut_{H,h}(T_1), \cdots, Cut_{H,h}(T_n))$. For the purpose of a contradiction we assume that S_h is not a solution for P_h, and R_h is a new solution for P_h. This implies that $GD(P_h, S_h) > GD(P_h, R_h)$. We modify S by substituting a subtree S_h with R_h. Let the resulting new tree be R. Since $GD(P_h, S_h) > GD(P_h, R_h)$, we know that $GD(P, S) > GD(P, R)$. This is a contradiction to that S is a solution for P. □

We propose Algorithm 2 that solves the gene duplication problem by dividing a given instance into an equivalent set of its subinstances. Then, each of these subinstances is solved

by another algorithm, called *GD-SOLVER*, which is provided as an additional input of Algorithm 2. For the GD-SOLVER we are utilizing the Chang et al. algorithm [Chang et al. (2013)] that is currently one of the most efficient exact solutions for the gene duplication problem.

Algorithm 2 Strict Consensus Approach Solution(P)

Input: A profile $P = (T_1, \ldots, T_n)$ and $GD - SOLVER$.
Output: A candidate solution for P.

$H = \mathcal{SC}(P)$
for all internal node h of H **do**
  $I_h = Cut_{H,h}(P)$
  $S_h = GD - SOLVER(P_h)$
  Refine children of h by the S_h
end for
return H

Theorem 4.2.2. *The time complexity of Algorithm 2 is $\mathcal{O}(3^r nmr/b)$ for an instance of input gene trees that have an overall number of n taxa, a strict consensus tree whose maximum out-degree is r, and m rooted splits. The parameter b refers to the size of the bit-vector that is used by the algorithm for encoding the rooted splits.*

Proof. Algorithm 2 executes the Chang et al. algorithm $\mathcal{O}(n)$ times, where each instance has an overall number of r taxa and $\mathcal{O}(m)$ unique rooted splits. The runtime of the Chang et al. algorithm for each of these instances is $\mathcal{O}(3^r mr/b)$ [Chang et al. (2013)]. Consequently, the time complexity of Algorithm 2 is $\mathcal{O}(3^r nmr/b)$. □

In practice, Algorithm 2 is limited by the maximum out-degree of the strict consensus tree of the input trees H, and the GD-SOLVER that is utilized. The out-degree of an internal node h of the strict consensus tree H of the trees in the original problem instance is equivalent to the number of taxa of the subinstance $P_h = Cut_{H,h}(P)$, which is an input for the GD-SOLVER. Since the gene duplication problem is NP-hard, exact algorithms that are used as GD-SOLVER can only compute smaller instances with up to k_{max} taxa in reasonable time (e.g., $k_{max} = 22$ as shown by Chang et al. (2013)). Consequently, Algorithm 2 can solve the gene duplication problem optimally in reasonable time, if each of the subproblems that it solves has at most k_{max} taxa. However, subinstances with more than k_{max} taxa can be effectively addressed

by using standard heuristics for the gene duplication problem, such as DupTree [Wehe et al. (2008)]. Utilizing heuristics to solve subinstances makes Algorithm 2 a heuristic too. However, Algorithm 2 greatly extends on the input sizes that can typically be handled by the heuristic that is used as GD-SOLVER. For instance, Figure 4.3 depicts a strict consensus tree that corresponds to subproblems that should be either addressed using an exact or heuristic GD-SOLVER. Now, we call the described method as the *Strict Consensus Approach*.

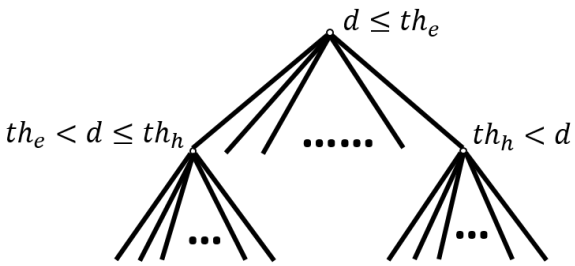

Figure 4.3: d: out degree, th_e: threshold for exact algorithm, th_h: threshold for heuristic algorithm. Different GD-SOLVERs are served by out degree of internal nodes (the number of taxa) in the substructure approach.

Our new software Gene Duplication Solver by Strict Consensus Approach (GDSS) implements Algorithm 2. GDSS adapts the exact Chang et al. algorithm and the classical local search heuristic DupTree as external GD-solvers. The inputs for GDSS is a list of gene trees and k_{max}. The output is the species tree that is found by utilizing the Strict Consensus Approach. GDSS is freely available at http://genome.cs.iastate.edu/ComBio/software.htm.

4.3 Experiments

We demonstrate that our parameterized algorithm can handle large-scale instances of the gene duplication problem using empirical data. A standard approach to compute such instances is to compute heuristic estimates using the program DupTree, which has been used to perform several compelling large-scale phylogenetic studies [Cotton and Page (2002); Martin and Burg (2002); McGowen et al. (2008); Page (2000)]. Now, for the first time, we are able to analyse the accuracy of DupTree on large-scale instances. Section 4.3.1 describes the performance study of our algorithm, and Section 4.3.2 analyzes the accuracy of DupTree.

4.3.1 Empirical study

Methods for inferring phylogenetic trees from molecular data, such as maximum parsimony and maximum likelihood, often produce a collection of several phylogenetic trees over the same taxon set for the given data [Stamatakis (2014)]. The phylogenetic information that is common in these trees is typically described by some type of consensus tree, such as the strict consensus tree. For a meaningful study it is generally expected that this consensus tree displays some structure, rather than being a star tree. This motivates our algorithm that refines the strict consensus tree by solving the gene duplication problem, which is fixed parameterized by the maximum degree of this consensus tree. However, it is still necessary to validate whether the consensus trees displayed by large-scale empirical data are of a manageable structure that is in practice solvable by our approach. Hence, we conducted a large-scale study using empirical data that measures the distribution of the node degrees in strict consensus trees.

For our study we use the primary alignment of $1,400$ 5S Ribosomal RNA bacterial sequences from the Comparative RNA Web Site Project [Cannone et al. (2002)]. Their taxonomic coverage of Phyla is Acidobacteria, Actinobacteria, Aquificae, Bacteroidetes, Chlamydiae, Chlorobi, Chloroflexi, Cyanobacteria, Deinococcus-Thermus, Firmicutes, Fusobacteria, Planctomycetes, Proteobacteria, and Verrucomicrobia. We inferred $40,000$ gene trees from this alignment using the maximum parsimony approach that is implemented in the program Parsimonator [Stamatakis (2014)]. Our objective is to represent these trees through their gene duplication median tree, i.e., the solution of the corresponding instance of the gene duplication problem. A standard approach to handle large-scale instances of this size is to utilize the heuristic DupTree for estimating a gene duplication median tree. However, the runtimes of DupTree are prohibitive for our instance with gene trees that have $1,400$ leaves. The runtimes of DupTree are reported with $1,000$ leaves and $2,000$ leaves to be 3.3 hours and up to 3 days respectively [Wehe et al. (2008)]. Furthermore, DupTree is a heuristic and thus lacks any type of performance guarantee about the trees that it estimates. Investigating into the degree distribution of strict consensus trees for large-scale empirical studies will evaluate to what extent our subproblem approach is applicable for such studies where heuristics may fail to produce estimates in reasonable time.

Dataset and Experimental Setup. We executed the program Parsimonator 40,000 times and gathered the best 400 gene trees in terms of their parsimony score. We computed 1,000 strict consensus trees using the following general boot strap technique; that is, 1000 times we sampled 40 gene trees randomly from the best 400 gene trees. For each sample we constructed a strict consensus tree from the chosen 40 gene trees. Figure 4.4 depicts the resulting degree distribution of the nodes of these strict consensus trees.

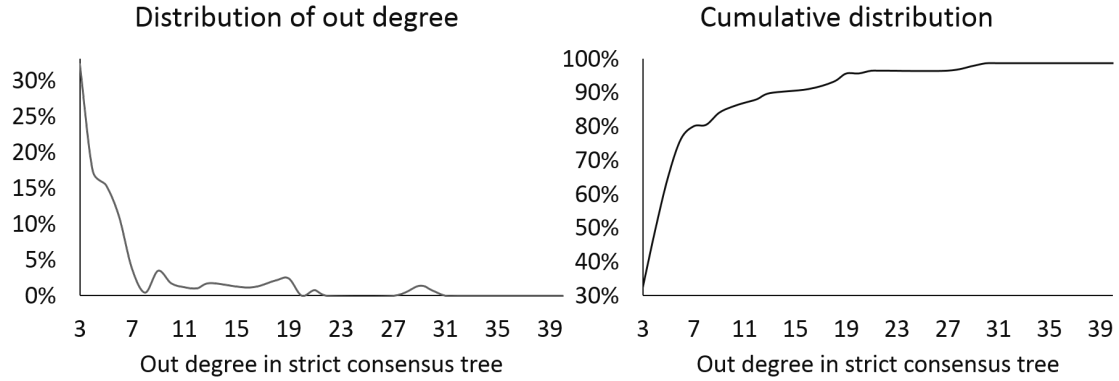

Figure 4.4: Distribution and the cumulative distribution of the node out degrees from 1,000 strict consensus trees.

Results. Figure 4.4 shows that 95% of the internal nodes have an out degree that is less than 20. The exact algorithm from Chang et al. (2013) solves the problem of 18 leaves within 10 minutes. Consequently, using our scalable approach, most of the subproblems resulting from the gene tree instances with 1,400 leaves can be solved optimally within reasonable time. However, all strict consensus trees from our simulation have at least one internal node whose out degree is around 800, and therefore, corresponding subproblems require a heuristic approach.

4.3.2 Simulated study

We analyze the performance of the heuristic DupTree for gene duplication problem using two types of simulated experiments. Experiment 1 uses small sized instances for which we do not apply our substructure approach. Experiment 2 uses large-scale ones that we solve using our subproblem approach. For both experiments to generate the instances we computed binary tree T randomly using the following method: i) given k leaves, ii) insert k leaves to a queue Q

and T, iii) randomly pops two nodes x, y from Q, iv) insert a new node z to Q and T such that $CH_T(z) = \{x, y\}$. v) repeat the process iii)-iv) until $|Q| = 1$.

Experiment 1. Analyzing DupTree's accuracy, Chang et al. (2013) reported that DupTree found optimal scores in only 8% of 1,000 runs using problem instances with 16 taxa. For our purposes we performed a more detailed analysis that provides the probabilities for DupTree to compute the optimal score under various conditions.

Dataset and Experiment Setup. For each $k \in \{5, 6, 7, 8, 9\}$, we generated 100 random instances, each consisting of 40 full binary gene trees with k leaves. For each of these instances we run DupTree t times for each $t \in \{1, 2, 3, \ldots, 20\}$. Above runs were performed 50 times.

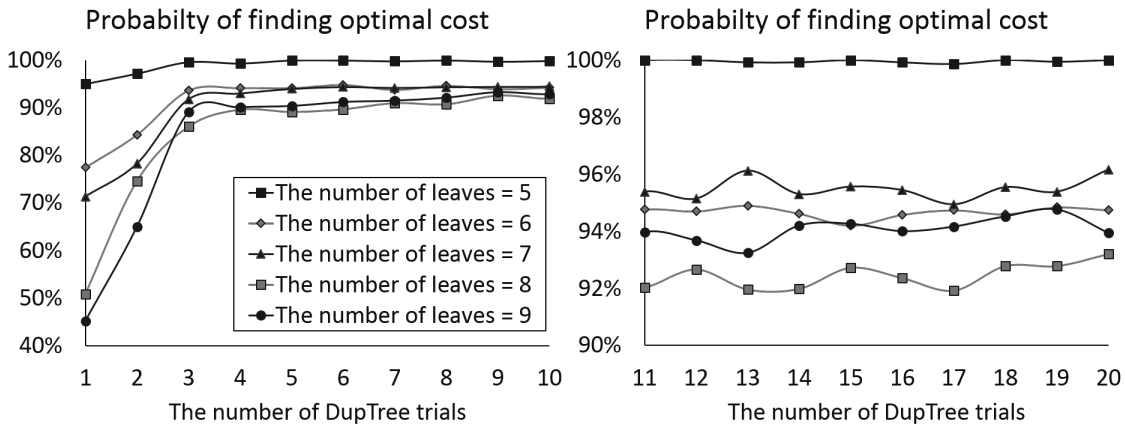

Figure 4.5: Probability of finding an optimal score by repeated DupTree runs. Each line in the chart stands for a different number of leaves.

Results. Figure 4.5 shows that DupTree identifies the optimal score in more than 90% of the runs when the trial number is larger than 4 for all numbers of leaves. When the number of leaves is 5 and the number of trials is larger than 2 DupTree identifies the optimal score in more than 99% the runs. For other numbers of leaves, the probability that DupTree reports the optimal one increases rapidly when the number of trials increase from 1 to 3. Increasing the number of trials further does not increase the probability that DupTree reports correct results.

Experiment 2. Using our subproblem approach we measure the error rate of estimated scores by DupTree and the running time ratio between the exact algorithm and DupTree for large-scale instances.

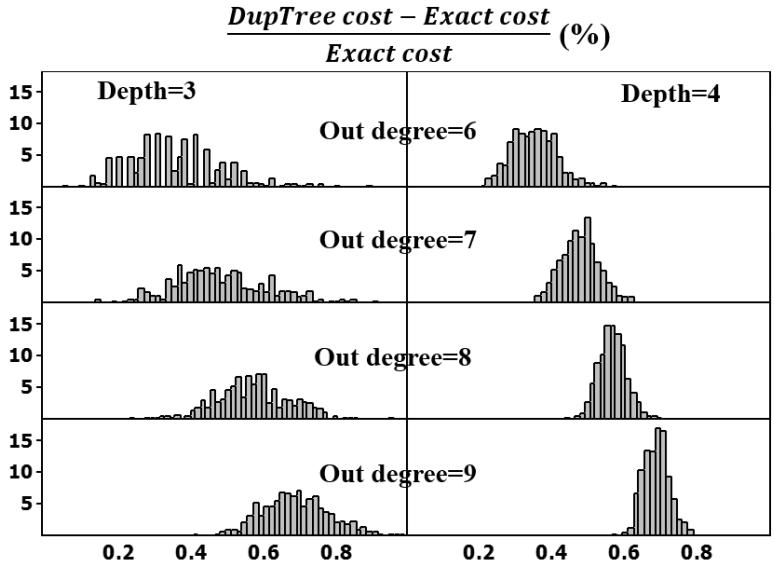

Figure 4.6: Error rates of optimal score by using DupTree under our subproblem approach. The left chart shows the result for templates of depth $d = 3$ and the right chart depicts this for depth $d = 4$. Each chart is divided into panels by out degree $b \in \{6, 7, 8, 9\}$. The number of leaves is b^d.

Dataset and Experimental Setup. We generated random instances of trees that conform to a given strict consensus tree. To generate an input, we used a given template tree for which we computed 40 randomly refined input trees. While the template tree can be equal to the strict consensus tree of the input trees as desired, the random refinement can also produce additional consensus clusters which would refine the template tree. Therefore, we omitted generated instances with additional consensus clusters. Template trees were generated as complete b-ary trees of depth d, denoted $C_{d,b}$, for each $d \in \{3, 4\}$ and $b \in \{6, 7, 8, 9\}$. For each such template tree we generated 10 random instances. Thus, we generated 80 random cases overall, where each one consists of 40 input trees. Next, we computed for each of these inputs an exact score and a heuristic estimate by solving the corresponding subproblems using our subproblem approach. To compute the exact score we used the algorithm from Chang et al. (2013). We computed a heuristic estimate for each instance of the gene duplication problem by taking the best score of 50 DupTree runs for this instance. The scores for each of these runs was computed by running DupTree 20 times on each subproblem of the original problem instance, and summing up the best scores found for each subproblem. To address how the species trees with optimal

scores differ from their heuristic estimates in terms of their topologies, rather than their gene duplication scores, we computed their Robinson Foulds (RF) distance [Robinson and Foulds (1981)].

Table 4.1: Average running time ratios between using the exact algorithm and DupTree under our subproblem approach. The runtime reported for DupTree is the aggregated time of repeating DupTree 20 times. The measuring scales are minutes as units.

Depth	Out degree	Taxa	Exact time	DupTree time	$\frac{\text{Exact time}}{\text{DupTree time}}$
3	6	216	0.03	0.09	0.31
3	7	343	0.20	0.14	1.44
3	8	512	1.44	0.21	7.01
3	9	729	14.39	0.29	58.88
4	6	1296	0.15	0.53	0.29
4	7	2401	1.21	0.97	1.36
4	8	4096	13.40	1.67	8.04
4	9	6561	159.56	2.71	58.84

Results. Figure 4.6 shows that the error rate of the duplication score computed by DupTree increases monotonically with the out degree for both depths. While the error rates of the estimated DupTree scores are within 1% of the corresponding optimal duplication scores, the RF distance between these trees ranges between 10% to 24% of the maximum possible RF score (i.e., $2(n-2)$ for two binary n-taxon trees). Table 4.1 shows that running times of DupTree are much faster for large node degrees b. In summary, we demonstrated that our approach can compute exact scores for large-scale instances when their strict consensus tree has a bounded degree. Moreover, by replacing exact solvers with heuristic solvers for each subproblem, we can obtain further speedups.

4.4 Conclusion

While we show that the gene duplication problem is not Pareto for clusters, we prove that this problem satisfies a slightly weaker version of this property, called weak Pareto. The weak Pareto property allows us to design an efficient parameterized algorithm for the gene duplication problem for input trees with the same taxon set. The parameter is the maximum degree of the strict consensus tree of the input trees. This parameterization can be, as we show in

our empirical studies, highly beneficial in practice. Thus our algorithm provides a substantial improvement in runtime and scalability when compared to previous solutions, which enables large-scale analyzes using the gene duplication problem. Further, our parameterized algorithm allows the inference of exact large-scale studies for the gene duplication problem that, so far, were only possible to be estimated by using heuristics.

CHAPTER 5. PHYLOGENETIC TREE COMPLETION I

We describe an efficient approach that adopts the Strict Consensus Approach to also handle unrestricted supertree problems. This is achieved by transforming an unrestricted instance of a supertree problem into a corresponding restricted instance of this problem, which then can be processed by the Strict Consensus Approach.

A key part of this transformation is a tree that is guiding this procedure, called guidance tree. *Guidance trees* for an unrestricted instance are unrooted trees that satisfies the *non-contradiction* property [Ranwez et al. (2007)]. Existing methods, such as the Strict Consensus Merger(SCM) [Huson et al. (1999); Fleischauer and Böcker (2016)], are available for the efficient construction of complex guidance trees that in practice often allow to break down supertree instances into much smaller sub-instances [Moon and Eulenstein (2016)].

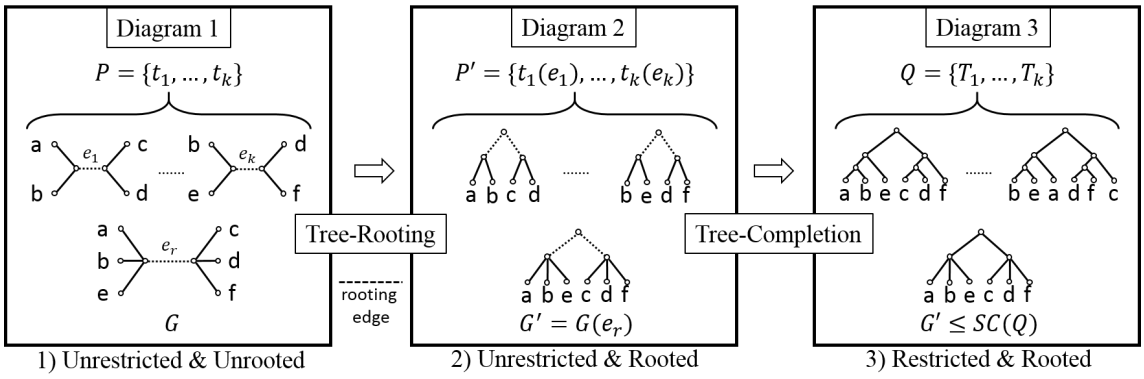

Figure 5.1: An illustration of Unrestricted and Unrooted Tree Rooting problem and Unrestricted and Rooted Tree Fill-in problem.

The *transformation* consists of two steps, a first step that is rooting the input trees, which is followed by a second step that is filling in these trees such that they are sampled from a common set of species. For the first step, we introduce the *Unrestricted and Unrooted Tree Rooting problem* that given an unrooted guidance tree and an unrestricted instance of input

trees, seeks a rooting of these trees such that they are displayed by some rooted version of the guidance tree. Figure 5.1 depicts an example instance of this problem in Diagram 1, where the trees $t_1, \ldots, t_k$ represent an unrestricted supertree instance, and tree G is a corresponding guidance tree. Diagram 2 illustrates a solution to this instance, which is a rooting of the input trees, i.e., the trees $t'_1, \ldots, t'_k$, and a rooting of the guidance tree, i.e., G', displaying the rooted input trees. For the second step, we introduce the *Unrestricted and Rooted Tree Fill-in problem* that given a collection of rooted gene trees sampled from different species and a corresponding guidance tree, seeks a collection of rooted trees sampled from the same species with an overall minimum number of nodes such that (i) each tree displays exactly one of the input gene trees, and (ii) their strict consensus tree is a refinement of the guidance tree (note the the guidance tree is a refinement of itself). Figure 5.1 is illustrating an example, where Diagram 2 and Diagram 3 represent an instance for the Unrestricted and Rooted Tree Fill-in problem and its solution respectively. We have devised efficient algorithms for the Unrestricted and Unrooted Tree Rooting problem and the Unrestricted and Rooted Tree Fill-in problem that effectively adopt the Strict Consensus Approach to handle unrestricted supertree problems.

Finally, we demonstrate the performance of our new approach for the unrestricted gene duplication problem when compared with standard heuristics for this problem using simulated studies and published empirical data sets.

5.1 Unrestricted and Unrooted Tree Rooting

Let $P = (t_1, \ldots, t_k)$ be a profile of unrestricted and unrooted trees and G be a guidance tree for P. A rooting in G is defined by selecting an edge $e_r \in E(G)$ on which the root is to be placed. Such a rooted tree is denoted by $G(e_r)$, and e_r is called *rooting edge*. As a preliminary step, we adapt the transformed digraph from Górecki and Eulenstein (2012) in order to solve the Unrestricted and Unrooted Tree Rooting problem that finds a rooted version of unrooted input trees.

Let $G' = G(e_r)$ be the rooted guidance. For $i \in \{1, \cdots, k\}$, the restricted subtree G'_i is defined as $G'|\mathcal{L}(t_i)$. We establish the digraph, denoted by D_i, as $V(D_i) = V(t_i)$ and $E(D_i) = \{\langle v, w\rangle, \langle w, v\rangle | \{v, w\} \in E(t_i)\}$, and the edges of D_i are labeled by G'_i as follows. If $v \in \mathcal{L}(G'_i)$,

$\langle v, w\rangle \in E(D_i)$ is labeled by v. Otherwise, $\langle v, w_l\rangle \in E(D_i)$ is labeled by $lca_{G'_i}\left(\bigcup_{j,j\neq l} m_j\right)$ where m_j is the label of $\langle w_j, v\rangle$. $(\forall j,\ \langle w_j, v\rangle \in E(D_i))$. Figure 5.2 depicts an example of this transformation, where the node set $V(D_i)$ is identical with $V(t_i)$ and the edges in $E(D_i)$ are created and labeled by t_i and G'_i.

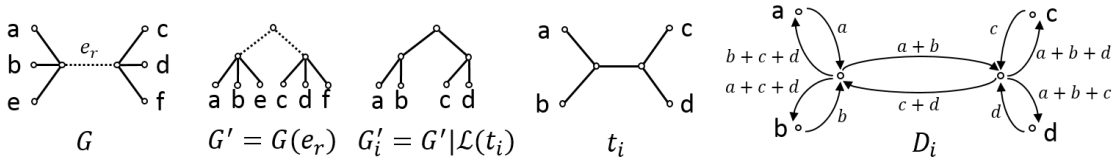

Figure 5.2: An example of a transformed digraph D_i from a given guidance tree G, $e_r \in E(G)$ and an input tree t_i. For the brevity, $lca_{G'_i}(\{a, b\})$ is denoted by $a + b$ here.

The edge $\langle w_j, v\rangle \in E(D_i)$ is called *incoming*, and the edge $\langle v, w_j\rangle \in E(D_i)$ is called *outgoing*. A pair of incoming and outgoing edges of $\{v, w_j\}$ is categorized as 1) **empty**: none of edges is labeled as $r(G'_i)$, 2) **in-single**: only incoming edge is labeled as $r(G'_i)$, 3) **out-single**: only outgoing edge is labeled as $r(G'_i)$, and 4) **double**: both edges are labeled as $r(G'_i)$. If $v \in V_{int}(t_i)$, the node $v \in D_i$ is classified based on pairs of incoming and outgoing edges; **M1**: one pair is in-single, and other pairs are out-singles, **M2**: one pair is empty, and other pairs are out-singles, **M3**: one pair is double, and other pairs are out-singles, **M4**: at least two pairs are doubles, and at least one pair is out-single, **M5**: all pairs is double, and **M6**: all pairs is out-single. Figure 5.3 illustrates the categories of edge pairs and node types. The original transformed digraph and Lemma 5.1.1 have been proposed by Górecki and Eulenstein (2012).

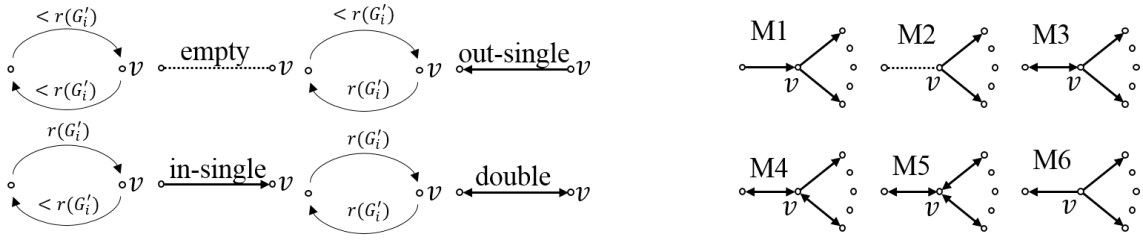

Figure 5.3: The categories of edge pairs and node types in a transformed graph.

Lemma 5.1.1 (Górecki and Eulenstein (2012))**.** *The following four cases are mutually exclusive.*

1. *D_i has one M2 node and all others are M1.*

2. *D_i has two M2 nodes that share the empty pair, and all others are M1.*

3. *D_i has one M6 node, and all others are M1.*

4. *D_i has one of M3, M4, or M5 nodes, and all others are M1, M3, M4, or M5.*

To apply the Strict Consensus Approach to unrestricted and unrooted median tree problem instances, as the first step of our method, we transform unrestricted and unrooted input trees into corresponding unrestricted and rooted trees.

Problem 5.1.1 (Unrestricted and Unrooted Tree Rooting Problem)**.**

Input: *An unrestricted and unrooted profile $P = \{t_1, \cdots, t_k\}$ and a rooted tree $G' = G(e_r)$ where G is a guidance tree for P and $e_r \in E(G)$.*

Output: *An unrestricted and rooted profile $P' = \{t_1(e_1), \cdots, t_k(e_k)\}$ where $e_i \in E(t_i)$ $(\forall i \in \{1, \cdots, k\})$ such that G' is a guidance tree for P'. (e_i is called a* non-contradict rooting edge.*)*

Lemma 5.1.2. *Let t_i be a tree in an unrooted profile P, G be a guidance tree for P, $G' = G(e_r)$ be a rooted tree where $e_r \in E(G)$, and D_i be the transformed digraph.*

1. *If D_i has one or two M2 node(s), then the empty pair of M2 corresponds to the non-contradict rooting edge in $E(t_i)$.*

2. *If D_i has one M6 node, then all out-single pairs of M6 correspond to the non-contradict rooting edges in $E(t_i)$.*

3. *If D_i has one of M3, M4, or M5 nodes, then all pairs of M3, M4, and M5 correspond to the non-contradict rooting edges in $E(t_i)$.*

Proof. Let $G_i = G|\mathcal{L}(t_i)$ and $G'_i = G'|\mathcal{L}(t_i)$.

1. Let $e_i = (v, w) \in E(t_i)$ be corresponded to the empty pair of M2. The full split $V|W \in \Sigma(t_i)$ is defined by removing of the edge e_i. Let $v' = lca_{G'_i}(V), w' = lca_{G'_i}(W)$, then $v', w' < r(G')$ because the empty pair is labeled as $< r(G')$. If v' or w' is not a child of $r(G')$, then the empty pair is supposed to be a single pair, hence, $v', w' \in Ch_{G'}(r(G'))$.

Let $t'_i = t_i(e_i)$, then we observe that $\mathcal{H}(G'_i) = \mathcal{H}(G'_i(v')) \cup \mathcal{H}(G'_i(w')) \cup \mathcal{L}(G'_i)$ and $\mathcal{H}(t'_i) = \mathcal{H}(t'_i(v)) \cup \mathcal{H}(t'_i(w)) \cup \mathcal{L}(t'_i)$. Because of $\Sigma(G_i) \subseteq \Sigma(t_i)$, $\mathcal{H}(G'_i(v')) \subseteq \mathcal{H}(t'_i(v))$ and $\mathcal{H}(G'_i(w')) \subseteq \mathcal{H}(t'_i(w))$, hence, $\mathcal{H}(G'_i) \subseteq \mathcal{H}(t'_i)$ and e_i is the non-contradict rooting edge.

2. Let $v \in V(t_i)$ be corresponded to the M6 node and $e_j = \{v, w_j\}$ be incident edges to v. The full split $V|W_j$ is defined by removing the edge e_j. Let $w'_j = lca_{G'_i}(W_j)$, then w'_j is a child of $r(G'_i)$ because $lca_{G'_i}(V) = r(G'_i)$ by the property of M6. We observe that $\mathcal{H}(G'_i) = \big(\bigcup_j \mathcal{H}(G'_i(w'_j))\big) \cup \mathcal{L}(G'_i)$. Let's pick an arbitrary e_{j^*} among e_j's and $t'_i = t_i(e_{j^*})$, then $\mathcal{H}(t'_i) = \big(\bigcup_j \mathcal{H}(t'_i(w_j))\big) \cup \big(\bigcup_{j \neq j^*} W_j\big) \cup \mathcal{L}(t'_i)$. Because of $\Sigma(G_i) \subseteq \Sigma(t_i)$, $\mathcal{H}(G'_i(w'_j)) \subseteq \mathcal{H}(t'_i(w_j))$. Hence, $\mathcal{H}(G'_i) \subseteq \mathcal{H}(t'_i)$ and all e_j's are the non-contradict rooting edges.

3. Suppose we choose a single pair $v \to w \in E(D_i)$ (among all pairs of M3, M4, and M5) and the corresponding edge $e_i = (v, w) \in E(t_i)$ as the rooting edge. The full split $V|W$ is defined by removing of the edge e_i. Let $v' = lca_{G'_i}(V)$ and $w' = lca_{G'_i}(W)$, then $v' = r(G'_i)$ and $w' < r(G'_i)$. Let $t'_i = t_i(e_i)$, then, we observe that $\mathcal{H}(G'_i) = \big(\mathcal{H}(G'_i|V) \setminus V\big) \cup \mathcal{H}(G'_i(w')) \cup \mathcal{L}(G'_i)$ and $\mathcal{H}(t'_i) = \mathcal{H}(t'_i(v)) \cup \mathcal{H}(t'_i(w)) \cup \mathcal{L}(t'_i)$. Because of $\Sigma(G_i) \subseteq \Sigma(t_i)$, $\mathcal{H}(G'_i|V) \setminus V \subseteq \mathcal{H}(t'_i(v))$ and $\mathcal{H}(G'_i(w')) \subseteq \mathcal{H}(t'_i(w))$, hence, $\mathcal{H}(G'_i) \subseteq \mathcal{H}(t'_i)$ and e_i is one of the non-contradict rooting edges. A similar proof works for a double pair $v \leftrightarrow w$.

□

Proposition 5.1.1. *The time complexity of Algorithm 3 is $\mathcal{O}(kn)$, where $n = |\mathcal{L}(P)|$ and k is the number of trees in P.*

Proof. The proof of correctness follows from Lemma 5.1.2. An lca operation can be computed in constant time after an initial preprocessing step requiring $\mathcal{O}(n)$ time [Górecki and Eulenstein (2012)], hence, the labeling procedure (lines 5 – 9) can be computed in $\mathcal{O}(n)$ time. The other two inner-for loops (lines 3 – 11) require $\mathcal{O}(n)$ iterations. The time required by the while loop (lines 12 – 19) is $\mathcal{O}(n)$. The outer for-loop (line 1 – 21) is executed k times, the entire algorithm requires $\mathcal{O}(kn)$ time. □

Algorithm 3 Unrestricted and Unrooted Tree Rooting(P, $G(e_r)$)

Input: An unrestricted/unrooted profile $P = \{t_1, \cdots, t_k\}$ and a rooted tree $G' = G(e_r)$ where G is a guidance tree for P and $e_r \in E(G)$.
Output: An unrestricted/rooted profile $P' = \{t_1(e_1), \cdots, t_k(e_k)\}$ where $e_i \in E(t_i)$ ($\forall i \in \{1, \cdots, k\}$) such that G' is a guidance tree for P'.

for all $t_i \in P$ **do**
  $G'_i = G'|\mathcal{L}(t_i)$, $V(D_i) = V(t_i)$
  for all $\{v, w\} \in E(t_i)$ **do** Add $\langle v, w\rangle$ and $\langle w, v\rangle$ to $E(D_i)$
  end for
  for all $\langle v, w_l\rangle \in E(D_i)$ **do** ▷ Suppose $\langle w_j, v\rangle$ is labeled by m_j
    if $v \in \mathcal{L}(t_i)$ **then** $\langle v, w_l\rangle$ is labeled by v
    else $\langle v, w_l\rangle$ is labeled by $lca_{G'_i}\big(\bigcup_{j, j\neq l} m_j\big)$ ($\forall j$ if $\langle w_j, v\rangle \in E(D_i)$)
    end if
  end for
  for all $v \in V(D_i)$ **do** Set the type of v among M1~M6
  end for
  while e_i is $NILL$ **do** ▷ Suppose e_i is initialized as $NILL$
    $v = POP(V(D_i))$
    if v is the type M2 **then**
      Find the incident empty pair and set the corresponding edge as e_i
    else if v is not M1 **then**
      Select an incident pair arbitrarily and set the corresponding edge as e_i
    end if
  end while
  Add $t_i(e_i)$ to P'
end for

5.2 Unrestricted and Rooted Tree Fill-in

Now, as the second step of our method, we transform unrestricted and rooted trees into corresponding restricted and rooted ones to apply the Strict Consensus Approach to unrestricted and unrooted median tree problem instances.

Problem 5.2.1 (Unrestricted and Rooted Tree Fill-in Problem)**.**

Input: *An unrestricted and rooted profile $P = \{t_1, \cdots, t_k\}$ and a guidance tree G for P.*

Output: *A restricted and rooted profile $Q \in \langle P\rangle_G$ such that $|Q| = \min_{Q' \in \langle P\rangle_G} |Q'|$.*

Lemma 5.2.1. *Let t be a tree in a rooted profile P and G be a guidance tree for P. If $T \in \langle t\rangle_G$, then $\mathcal{H}(G) \cap \big(\mathcal{H}(t) \setminus \mathcal{H}(G|\mathcal{L}(t))\big) = \emptyset$ and $|\mathcal{H}(T)| \geq |\mathcal{H}(G)| + |\mathcal{H}(t) \setminus \mathcal{H}(G|\mathcal{L}(t))|$.*

Proof. For the first part, $\mathcal{H}(G) \cap \big(\mathcal{H}(t) \setminus \mathcal{H}(G|\mathcal{L}(t))\big) = (\mathcal{H}(G) \cap \mathcal{H}(t)) \setminus \mathcal{H}(G|\mathcal{L}(t))$. For any cluster $c \in \mathcal{H}(G) \cap \mathcal{H}(t)$, we observe that $c \cap \mathcal{L}(t) = c$ and $c \subseteq \mathcal{L}(t)$. Hence, if $c \in \mathcal{H}(S) \cap \mathcal{H}(t)$, then $c \in \mathcal{H}(S|\mathcal{L}(t))$. It implies $(\mathcal{H}(S) \cap \mathcal{H}(t)) \setminus \mathcal{H}(S|\mathcal{L}(t)) = \emptyset$. For the second part, let $f\colon \mathcal{H}(\mathcal{T}) \times \mathcal{L} \to \mathcal{H}(\mathcal{T}_{\mathcal{L}})$ be surjective such that $\forall c' \in \mathcal{H}(\mathcal{T}_{\mathcal{L}}), \exists c \in \mathcal{H}(\mathcal{T})$, $c' = c \cap \mathcal{L}$ where $\mathcal{T}$ is a tree, $\mathcal{L} \subseteq \mathcal{L}(\mathcal{T})$

is a set of leaves, and $\mathcal{T}_{\mathcal{L}}$ is a tree on $\mathcal{L}$. From $T \in \langle t \rangle_G$, we observe that $\mathcal{H}(G) \subseteq \mathcal{H}(T)$ and $f(\mathcal{H}(T), \mathcal{L}(t)) = \mathcal{H}(T|\mathcal{L}(t)) = \mathcal{H}(t)$. Let $\mathcal{C}$ be a set of clusters such that $\mathcal{H}(T) = \mathcal{H}(G) \cup \mathcal{C}$ and $\mathcal{H}(G) \cap \mathcal{C} = \emptyset$. It follows $|\mathcal{H}(T)| = |\mathcal{H}(G)| + |\mathcal{C}|$. From $\mathcal{H}(t) = f(\mathcal{H}(T), \mathcal{L}(t)) = f(\mathcal{H}(G) \cup \mathcal{C}, \mathcal{L}(t)) = \mathcal{H}(G|\mathcal{L}(t)) \cup f(\mathcal{C}, \mathcal{L}(t))$, we observe that $\mathcal{H}(t) \setminus \mathcal{H}(G|\mathcal{L}(t)) \subseteq f(\mathcal{C}, \mathcal{L}(t))$. Hence, $|\mathcal{C}| \geq |f(\mathcal{C}, \mathcal{L}(t))| \geq |\mathcal{H}(t) \setminus \mathcal{H}(G|\mathcal{L}(t))|$ and $|\mathcal{H}(T)| \geq |\mathcal{H}(G)| + |\mathcal{H}(t) \setminus \mathcal{H}(G|\mathcal{L}(t))|$. □

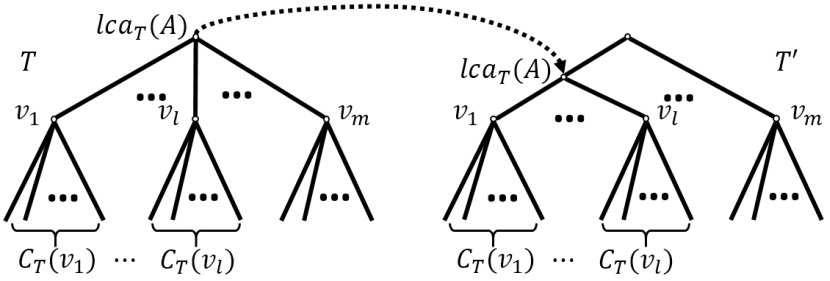

Figure 5.4: An illustration of the iLCA insertion.

Let T be a rooted tree, $A \subseteq \mathcal{L}(T)$ be a set of leaves, $v_{lca} = lca_T(A)$, and $Ch_T(v_{lca}) = \{v_1, ..., v_m\}$. If there exists set of nodes $B = \{v_1, ..., v_l\} \subset Ch_T(v_{lca})$ $(l < m)$ such that $A \subseteq \bigcup_{v \in B} \mathcal{C}_T(v)$ and $A \cap \mathcal{C}_T(v) \neq \emptyset$ $(\forall v \in B)$, then we can insert the *intrinsic least common ancestor (iLCA)* of A to be the parent of all nodes in B and the child of v_{lca}. Figure 5.4 depicts an example of the iLCA insertion.

Algorithm 4 Unrestricted and Rooted Tree Fill-in(P, G)

Input: An unrestricted/rooted profile $P = \{t_1, \cdots, t_k\}$ and a guidance tree G for P.
Output: A restricted/rooted profile $Q \in \langle P \rangle_G$ such that $|Q| = \min_{Q' \in \langle P \rangle_G} |Q'|$.

for all $t \in P$ **do**
  $T = G$
  for all $A \in \mathcal{H}(t) \setminus \mathcal{H}(G|\mathcal{L}(t))$ **do** ▷ let $v_{lca} = lca_T(A)$
    for all $v_a \in A$ **do**
      Find the path between v_{lca} and v_a, which is $v_{lca}, v_{ch}, \ldots, v_a$
      $B = B \cup v_{ch}$ ▷ $B \subset Ch_T(v_{lca})$ because $A \in \mathcal{H}(t) \setminus \mathcal{H}(G|\mathcal{L}(t))$
    end for
    $V(T) = V(T) \cup \{v_{ilca}\}$
    Set $Pa_T(v_{ilca}) = v_{lca}$ and $Ch_T(v_{ilca}) = B$
  end for
  Add T to Q ▷ from the definition, if $Q \in \langle P \rangle_G$, then $G \leq \mathcal{SC}(Q)$.
end for

Proposition 5.2.1. *Algorithm 4 runs in $\mathcal{O}(kn^2)$ time, where $n = |\mathcal{L}(P)|$ and k is the number of trees in P.*

Proof. Suppose that the cluster A' is produced by inserting the ilca of $A \in \mathcal{H}(t) \setminus \mathcal{H}(G|\mathcal{L}(t))$ through lines 3 – 10. Lemma 5.2.1 guarantees that $A' \cap \mathcal{L}(t) = A$ and the number of the produced clusters is $|\mathcal{H}(t) \setminus \mathcal{H}(S|\mathcal{L}(t))|$, which is the minimum. The for-loop (lines 3 – 10) requires $\mathcal{O}(n)$ executions, since $|\mathcal{H}(t) \setminus \mathcal{H}(G|\mathcal{L}(t))| < |n|$. An lca operation can be computed in constant time after an initial preprocessing step requiring $\mathcal{O}(n)$ time [Górecki and Eulenstein (2012)]. The inner for-loop (lines 4 – 7) are computable in $\mathcal{O}(n)$ since $|A|, |B| < |n|$. Thus, the for-loop (lines 3 – 10) runs in time $\mathcal{O}(n^2)$. The outer for-loop (lines 1 – 12) is executed k time, which the desired runtime follows. □

5.3 Experiments

We demonstrate the scalability and accuracy of our new method when applied to the *gene duplication problem* [Eulenstein et al. (2010); Goodman et al. (1979)]. Given a collection of gene trees, the gene duplication problem seeks a species tree that is a median tree for the input trees under the *gene duplication score*. This problem is NP-hard [Ma et al. (2000)], $W[2]$-hard when parameterized by the gene duplication score [Bansal and Shamir (2011)] and hard to approximate better than a logarithmic factor [Bansal and Shamir (2011)]. While exact solutions for the gene duplication problem are coming into the reach of smaller phylogenetic studies (e.g., 22 taxa [Chang et al. (2013)]), for large-scale studies such solutions remain out of reach. Therefore, local search heuristics have been proposed [Bansal and Eulenstein (2013); Wehe et al. (2013)], analyzed, and applied to compute supertrees from gene trees [Cotton and Page (2002); Martin and Burg (2002); McGowen et al. (2008); Page (2000)]. However, they warrant no performance guarantee on the computed species trees, and consequently, their accuracy cannot be analyzed. Recently, using the strict consensus approach, it has been demonstrated that the restricted gene duplication problem can be solved for instances of complete gene trees with up to $6,561$ taxa [Moon et al. (2016)]. Given these promising results, we analyze our new method when applied to the unrestricted gene duplication problem.

5.3.1 Empirical study

We compared the gene duplication scores computed by our new approach and the heuristic DupTree [Wehe et al. (2008)] using published empirical data sets.

Data sets. We used the seabird data set that consists of 7 incomplete rooted trees representing the evolutionary histories of proper subsets of an overall set of 121 seabird species [Kennedy et al. (2002)]. The rooted trees in the data set have been computed by standard phylogenetic inference methods such as maximum parsimony or maximum likelihood with outgroup rootings. However, outgroup rooting can result in incorrect rootings when evolutionary events cause heterogeneity in the gene trees [Holland et al. (2003); Huelsenbeck et al. (2002)]. To avoid inaccurate rootings, we interpreted the seabird data set as unrooted trees.

Experimental Setting. We generated the profile P of 7 *incomplete unrooted trees* from the seabird data set and computed the *unrooted guidance tree* G by using the SCM method [Huson et al. (1999)]. The unrooted guidance tree G has 188 edges, and hence the same number of *rooted guidance trees* are computed. For each rooted guidance tree G', we executed our proposed algorithms through the following two steps. Step 1: Algorithm 3 takes the rooted guidance tree G' and the profile P as input and produces the profile P' of 7 *incomplete rooted* trees as output. Step 2: Algorithm 4 takes the rooted guidance tree G' and the profile P' as input and produces the profile Q of 7 *complete rooted* trees as output. In summary, we produced 188 instances of the *rooted guidance tree* and profiles of 7 *incomplete* and *complete rooted* trees.

We evaluated the performance of the strict consensus approach for computing species trees of the gene duplication problem by using the generated instances. The approach takes the *rooted guidance tree* and the profile of *complete rooted* trees as input and also requires a value k_{max}, which determines that sub-instances of size not larger than k_{max} are solved exactly, and otherwise, are addressed heuristically by running DupTree. Due to the performance limitations of our standard working station, we set $k_{max} = 9$. Every rooted guidance tree in the generated 188 instances has 10 multifucations, and 2 of these multifurcations has the out-degree of greater than k_{max}. Therefore, 8 subinstnaces were solved exactly and 2 sub-instances are addressed heuristically. For the baseline method, DupTree was executed with the profile of 7 *incomplete*

rooted trees in the instances. For each instance, we reiterated the computing of species trees 1,000 times separately by the strict consensus approach and DupTree. In the following section, we present one of the results that contain species trees of the lowest gene duplication score.

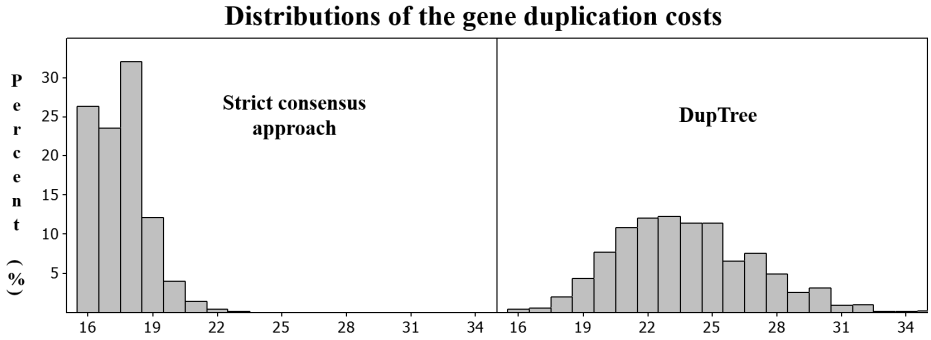

Figure 5.5: Distributions of the gene duplication scores of species trees that are computed by the strict consensus approach [left] and the DupTree heuristic [right].

Results and Discussion. Figure 5.5 depicts the distributions of the gene duplication scores of species trees that are computed by the strict consensus approach and the DupTree heuristic. The distribution of the strict consensus approach is right-skewed. In detail, 26.3% of the species trees that were computed by the strict consensus approach have the minimum gene duplication score of 16, but only 0.4% of those computed by DupTree have the minimum score of 16. These distributions demonstrate that the strict consensus approach improves significantly on the accuracy when compared to the DupTree results. This improvement can be explained by observing that the strict consensus approach solves all of the 10 sub-instances of the instance exactly, with the exception of 2 sub-instances that are addressed heuristically. Despite of this astonishing improvement, the strict consensus approach spent 30% more time than DupTree. In detail, the average computing times of the strict consensus approach is 8.5 seconds, and the times of the DupTree is 6.5 seconds.

5.3.2 Simulated study

We analyzed simulated data sets with a maximum of 2744 taxa, where each instance contains 20 incomplete unrooted trees.

Data sets. *Unrooted guidance trees* were created as b-ary trees for $b \in \{6, 7, 8\}$. Every b-ary

tree has the unique center node with an edge length of 4 between the center node and any leaf, and hence the number of leaves of the b-ary trees are 750, 1512, and 2744. The values for b and the length of 4 were determined by the performance limitations of our standard working station. For each unrooted guidance tree G, an intermediate profile consisting of 20 *complete unrooted trees* was computed, where each such tree is a random refinement of the guidance tree G. We then converted the intermediate profile into a profile P of *incomplete unrooted* trees by replacing each tree T in the intermediate profile by the tree $T|A$ for an independent random set $A \subseteq \mathcal{L}(T)$ (i.e., the set A is determined independently for each tree T). The probability that a leaf in $\mathcal{L}(T)$ is selected for the subset A is called the *leaf selection probability*, and we produced two profiles $P_{0.4}$, $P_{0.8}$ separately from the intermediate profile by setting the probability as 0.4 and 0.8.

Experiment Setting. We computed the *rooted guidance tree* G' by rooting an arbitrary edge of the unrooted guidance tree G. For each rooted guidance tree G', we executed our proposed algorithms through the following two steps. Step 1: Algorithm 3 takes the rooted guidance tree G' and the profile $P_{0.4}$ ($P_{0.8}$) as input and produces the profile $P'_{0.4}$ ($P'_{0.8}$) of 20 *incomplete rooted* trees as output. Step 2: Algorithm 4 takes the rooted guidance tree G' and the profile $P'_{0.4}$ ($P'_{0.8}$) as input and produces the profile $Q_{0.4}$ ($Q_{0.8}$) of 20 *complete rooted* trees as output. By repeating the procedure $1,000$ times, we produced a total number of $6,000$ instances of the *rooted guidance tree* and profiles of 20 *incomplete* and *complete rooted* trees.

We evaluated the performance of the strict consensus approach for computing species trees of the gene duplication problem by using the generated instances. We set $k_{max} = 9$, which is same as the previous empirical study. Therefore, all of subinstnaces were solved exactly because all internal nodes of rooted guidance trees that are generated from b-ary trees ($b \in \{6, 7, 8\}$), have out-degree less than k_{max}. For the baseline method, we executed DupTree with the profile of 20 *incomplete rooted* trees in each generated instance.

Results and Discussion. The strict consensus approach computed exact solutions while DupTree is heuristically estimating these solutions. Figure 5.6 shows that the error rate of DupTree increases as the number of leaves increases, and overall error rates are dramatically increased on the leaf selection probability of 40% than one of 80%. Moreover, Figure 5.7 shows that

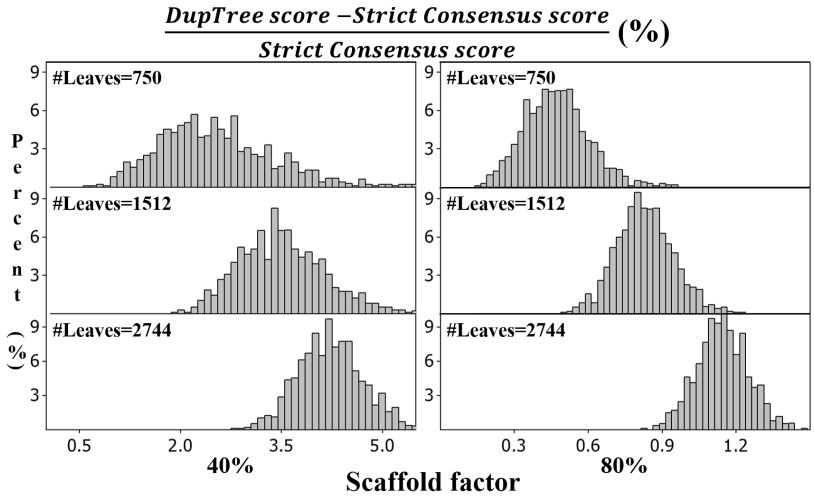

Figure 5.6: Distributions of the normalized gene duplication score differences of species trees that are computed by the strict consensus approach and the DupTree heuristic.

the gene duplication problem can be solved optimally in a reasonable time by using the strict consensus approach.

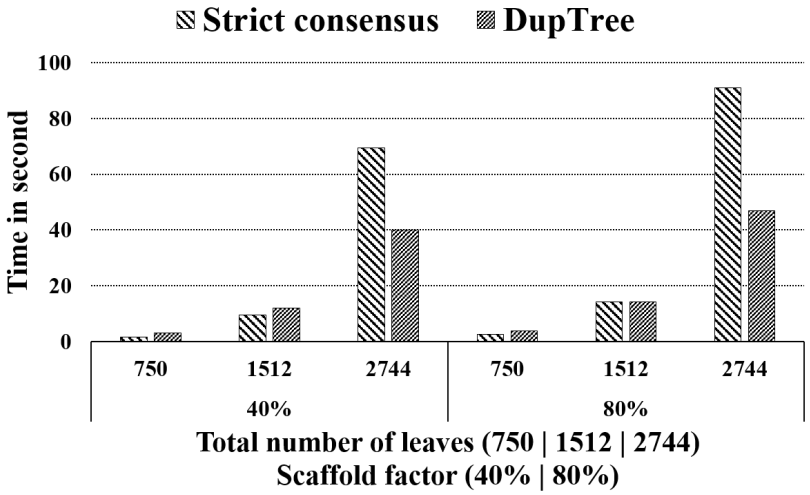

Figure 5.7: Average running times in second of computing species of the gene duplication problem by the strict consensus approach and the DupTree heuristic.

5.4 Conclusion

There is an increased interest in making NP-hard supertree problems available for large-scale species tree construction. While it has been demonstrated that the strict consensus tree

approach can successfully synthesize such trees, it is only able to handle restricted supertree problems, which have largely limited applicability in practice. In this work, we presented a novel approach that is overcoming this stringent limitation by adopting the strict consensus approach to also handle unrestricted supertree problems. Our studies on simulated and empirical studies established that our approach, implementing our efficient algorithms, makes the strict consensus tree approach a powerful tool for addressing large-scale supertree problems in their unrestricted version. While the approach is limited by supertree problems that satisfy (in their restricted version) the Pareto for clusters property and the substructure property, many such problems have not been analyzed whether they satisfy these properties. Future work will focus on such analyzes, and furthermore, investigate if other desirable Pareto properties can be used to address NP-hard supertree problems more effectively.

CHAPTER 6. PHYLOGENETIC TREE COMPLETION II

We introduce a novel approach to adopt the Strict Consensus Approach to handle unrestricted and unrooted median tree problem instances of large size. This approach consists of new computational problems and efficient algorithms to solve them.

In detail, our approach transforms unrestricted and unrooted input trees into corresponding restricted and rooted ones. This transformation is guided by a special tree, called the *guidance tree*, that satisfies the *non-contradiction* property [Ranwez et al. (2007)] to make this transformation possible. Existing methods, such as the Strict Consensus Merger(SCM) [Huson et al. (1999); Fleischauer and Böcker (2016)], are available for the efficient construction of complex guidance trees that in practice often allow to break down instances into much smaller sub-instances [Moon and Eulenstein (2016)].

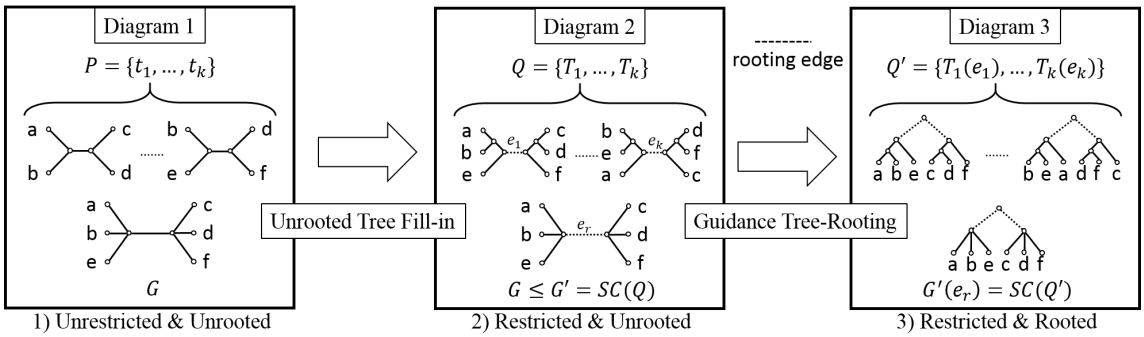

Figure 6.1: An illustration of the transformation from unrestricted and unrooted trees to the restricted and rooted trees.
Diagram 1: Unrestricted → Diagram 2: Restricted & Unrooted → Diagram 3: Rooted

The transformation of a collection of unrestricted and unrooted trees consists of two steps. First the trees are transformed into restricted trees by solving the "Guidance Tree based Fill-in Problem", which are then rooted by solving the "Guidance Tree based Rooting Problem". The *Guidance Tree based Fill-in Problem* seeks, for a given collection of unrestricted and unrooted

trees $t_1, \ldots, t_k$ and a corresponding guidance tree G, a restricted version of the input trees $T_1, \ldots, T_k$ with an overall minimum number of nodes, such that (i) each restricted tree T_i displays one unrestricted tree t_i, and (ii) their strict consensus tree $\mathcal{SC}(Q)$ is a refinement of, or identical to, the guidance tree G. The *Guidance Tree based Rooting Problem* seeks, for a given collection of restricted and unrooted trees $T_1, \ldots, T_k$ and a rooted guidance tree G', a rooted version of the input trees $T_1(e_1), \ldots, T_k(e_k)$ such that (i) each rooted tree $T_i(e_i)$ is the rooted version of one unrooted tree T_i and (ii) their strict consensus tree $\mathcal{SC}(Q')$ is identical to the *rooted* guidance tree $G'(e_r)$.

To solve the here introduced problems, the Guidance Tree based Fill-in Problem and the Guidance Tree based Rooting Problem, we describe efficient algorithms. Finally, we demonstrate the performance of our algorithms for the Strict Consensus Approach when applied to unrestricted and unrooted instances of the classical gene duplication problem, in comparison with standard heuristic approaches for this problem using simulated and published empirical data sets.

6.1 Unrestricted and Unrooted Tree Fill-in

We define the Unrestricted and Unrooted Tree Fill-in problem and propose the efficient algorithm for solving this problem. For a given profile of unrestricted and unrooted trees $t_1, \ldots, t_k$ and a corresponding guidance tree G, a restricted version of the input trees $T_1, \ldots, T_k$ with an overall minimum number of nodes such that (i) each restricted tree T_i displays one unrestricted tree t_i and (ii) their strict consensus tree $\mathcal{SC}(Q)$ is a refinement of or identical to the guidance tree G.

Problem 6.1.1. *(Unrestricted and Unrooted Tree Fill-in Problem)*

Input: *An unrooted profile $P = \{t_1, \cdots, t_k\}$ such that $\mathcal{L}(t_i) \subseteq \mathcal{L}(P)$ and a guidance tree G for P such that $\mathcal{L}(G) = \mathcal{L}(P)$ and $G|\mathcal{L}(t_i) \leq t_i$ $(\forall i \in \{1, \cdots, k\})$.*

Output: *An unrooted profile $Q \in \langle P \rangle_G$ such that $|Q| = \min_{Q' \in \langle P \rangle_G} |Q'|$.*

Lemma 6.1.1. *Let P be an unrooted profile and G be a guidance tree for P. If $Q \in \langle P \rangle_G$, then $G \leq \mathcal{SC}(Q)$.*

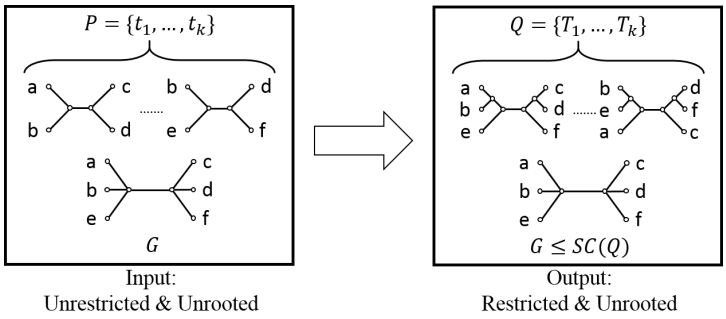

Figure 6.2: Schematic representation of the Unrestricted and Unrooted Tree Fill-in problem. Unrestricted input trees are transformed into the corresponding restricted trees.

Proof. By the definition of the restricted span $\langle P \rangle_G$, $\forall T \in Q$ is a refined tree of G, hence, $G \leq \mathcal{SC}(Q)$. □

Lemma 6.1.2. *Let t be a tree in an unrooted profile P, and G be a guidance tree for P. If $T \in \langle t \rangle_G$, then $\Sigma(G) \cap (\Sigma(t) \setminus \Sigma(G|\mathcal{L}(t))) = \emptyset$ and $|\Sigma(T)| \geq |\Sigma(G)| + |\Sigma(t) \setminus \Sigma(G|\mathcal{L}(t))|$.*

Proof. For the first part, $\Sigma(G) \cap (\Sigma(t) \setminus \Sigma(G|\mathcal{L}(t))) = (\Sigma(G) \cap \Sigma(t)) \setminus \Sigma(G|\mathcal{L}(t)) = \Sigma(t) \cap (\Sigma(G) \setminus \Sigma(G|\mathcal{L}(t)))$. If $\mathcal{L}(t) = \mathcal{L}(G)$, then $\Sigma(G) \setminus \Sigma(G|\mathcal{L}(t)) = \emptyset$. Otherwise, $\Sigma(G) \cap \Sigma(t) = \emptyset$.

For the second part, let $f \colon \Sigma(\mathcal{T}) \times \mathcal{L} \to \Sigma(\mathcal{T}_{\mathcal{L}})$ be surjective function such that $\forall \sigma' \in \Sigma(\mathcal{T}_{\mathcal{L}}), \exists \sigma \in \Sigma(\mathcal{T})$, $\sigma' = \sigma \cap \mathcal{L}$ where $\mathcal{T}$ is a tree, $\mathcal{L} \subseteq \mathcal{L}(\mathcal{T})$ is a leaves, and $\mathcal{T}_{\mathcal{L}}$ is a tree on $\mathcal{L}$. From $T \in \langle t \rangle_G$, follows that $\Sigma(G) \subseteq \Sigma(T)$ and $f(\Sigma(T), \mathcal{L}(t)) = \Sigma(T|\mathcal{L}(t)) = \Sigma(t)$. Let $\Sigma = \Sigma(T) \setminus \Sigma(G)$, then $|\Sigma(T)| = |\Sigma(G)| + |\Sigma|$. From $\Sigma(t) = f(\Sigma(T), \mathcal{L}(t)) = f(\Sigma(G) \cup \Sigma, \mathcal{L}(t)) = \Sigma(G|\mathcal{L}(t)) \cup f(\Sigma, \mathcal{L}(t))$, it follows $\Sigma(t) \setminus \Sigma(G|\mathcal{L}(t)) \subseteq f(\Sigma, \mathcal{L}(t))$. Therefore, $|\Sigma| \geq |f(\Sigma, \mathcal{L}(t))| \geq |\Sigma(t) \setminus \Sigma(G|\mathcal{L}(t))|$ and $|\Sigma(T)| \geq |\Sigma(G)| + |\Sigma(t) \setminus \Sigma(G|\mathcal{L}(t))|$. □

A rooting in a tree T is defined by selecting an edge $e \in E(T)$ on which the root is to be placed. Such a rooted tree is denoted by $T(e)$, and e is the *rooting edge*. There exist a bijective mapping, denoted by $\mathcal{M} : V(T) \to V(T_e)$, between $V(T)$ and $V(T_e) \setminus \{r(T_e)\}$.

Lemma 6.1.3. *Let t be a tree in an unrooted profile P, and G be a guidance tree for P. If $\sigma = A|B \in \Sigma(t) \setminus \Sigma(G|\mathcal{L}(t))$, then $G(A)$ and $G(B)$ share a single node $v \in V(G)$.*

Proof. Suppose that $G(A)$ and $G(B)$ does not share a single node. If $G(A)$ and $G(B)$ are not connected, then there exists at least one split $\sigma' \in \Sigma(G)$ such that $\sigma' \cap \mathcal{L}(t) = \sigma$. This is a

Algorithm 5 Unrestricted and Unrooted Tree Fill-in(P, G)

Input: An unrooted profile $P = \{t_1, \cdots, t_k\}$ such that $\mathcal{L}(t_i) \subset \mathcal{L}(P)$ and a guidance tree G for P such that $\mathcal{L}(G) = \mathcal{L}(P)$ and $G|\mathcal{L}(t_i) \leq t_i$ $(\forall i \in \{1, \cdots, k\})$.
Output: An unrooted profile $Q \in \langle P \rangle_G$ such that $|Q| = \min_{Q' \in \langle P \rangle_G} |Q'|$.

for all $t \in P$ **do**
  $T = G$
  $\Sigma = \Sigma(t) \setminus \Sigma(G|\mathcal{L}(t))$
  for all $\sigma \in \Sigma$ **do** ▷ let $\sigma = A|B$
    Pick arbitrarily $v_b \in B$
    $e_b = (v_b, v_b') \in E(T)$ ▷ $v_b' \in V_{int}(T)$
    $v_{lca} = LCA_{T(e_b)}(A)$
    for all $v_a \in A$ **do**
      $v = v_a$
      $Z = \emptyset$
      while $v \notin Z$ & $Pa_{T(e_b)}(v) \neq v_{lca}$ **do**
        Add v to Z
        $v = Pa_{T(e_b)}(v)$
      end while
      if $Pa_{T(e_b)}(v) = v_{lca}$ **then**
        Add v to V_{ch}
      end if
    end for
    $v_c \in V(T)$: mapped node of v_{lca}
    $V_{det} \subset V(T)$: mapped nodes of V_{ch}
    Add v_{new} to $V(T)$
    Add (v_c, v_{new}) to $E(T)$
    for all $v_{det} \in V_{det}$ **do**
      Remove $(v_c, v_{det}) \in V(T)$
      Add (v_{new}, v_{det}) to $E(T)$
    end for
  end for
  Add T to Q
end for
return Q

contradiction since $\sigma \in \Sigma(G|\mathcal{L}(t))$.

If $G(A)$ and $G(B)$ share more than one node, then $G(A)$ and $G(B)$ share at least one edge. Let $e \in E(G)$ be a shared edge and $\sigma_e = A_e|B_e \in \Sigma(G)$ be a split from removal of e, then $A \cap A_e \neq \emptyset$, $A \cap B_e \neq \emptyset$, $B \cap A_e \neq \emptyset$, and $B \cap B_e \neq \emptyset$. It follows $\sigma_e \cap \mathcal{L}(t) \in \Sigma(G|\mathcal{L}(t))$ and $\sigma_e \cap \mathcal{L}(t) \notin \Sigma(t)$. This is a contradiction since $\Sigma(G|\mathcal{L}(t)) \subseteq \Sigma(t)$. □

Lemma 6.1.4. *Let T be an unrooted tree, $A, B \subset \mathcal{L}(T)$, and $A \cap B = \emptyset$. Suppose that $T(A)$ and $T(B)$ share a single node $v_c \in V(T)$. If $v_b \in B$ and $e_b = (v_b, v_{b'})$ where $e_b \in E(T)$, then $\mathcal{M}(v_c) = LCA_{T'}(A)$ where $T' = T(e_b)$.*

Proof. Let $V_A = \{\mathcal{M}(v) \mid v \in V(T(A))\}$ and $V_B = \{\mathcal{M}(v) \mid v \in V(T(B))\}$, then $V_A \cap V_B =$

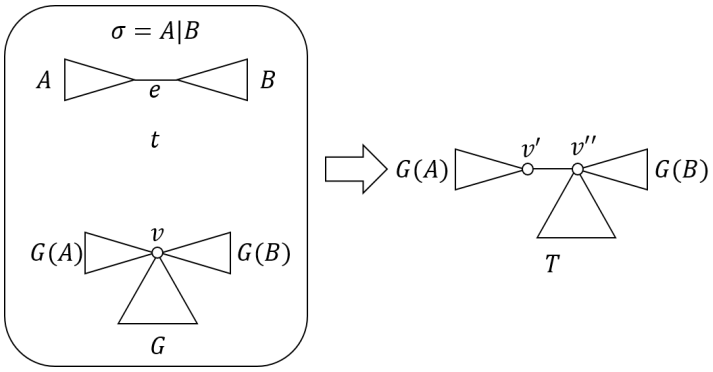

Figure 6.3: Schematic representation of the main idea of Algorithm 5.
t: an input tree, $\sigma = A|B$: defined by removal of e, G: the Guidance tree, T: the output tree
G does not have an edge that defines $\sigma = A|B$, but T has the edge.

$\{\mathcal{M}(v_c)\}$. By the definition of the rooting, we observe that $v \leq_{T'} \mathcal{M}(v_c)$ for all node $v \in V_A$ and $\mathcal{M}(v_c) \leq_{T'} v$ for all node $v \in V_B \setminus \{\mathcal{M}(v_b)\}$. $\mathcal{M}(v_c)$ is the smallest upper bound of A under $\leq_{T'}$, hence, $\mathcal{M}(v_c) = LCA_{T'}(A)$. □

Proposition 6.1.1. *Algorithm 5 runs in $\mathcal{O}(kn^2)$ time where k is the number of trees in a profile P and $n = |\mathcal{L}(P)|$.*

Proof. By Lemma 6.1.2, the number of splits in T (line 28), $|\Sigma(T)| = |\Sigma(G)| + |\Sigma(t) \setminus \Sigma(G|\mathcal{L}(t))|$, is the minimum among trees in $\langle t \rangle_G$. Lemma 6.1.3 guarantees that v_c (line 19) is the shared node between $G(A)$ and $G(B)$. The algorithm uses the temporary rooted tree $T(e_b)$ (line 7) to identify the v_c, which is guaranteed by Lemma 6.1.4.

By using the rooted tree $T(e_b)$, v_c is computable in $\mathcal{O}(n)$ time since an lca operation (line 7) can be computed in constant time after an initial preprocessing step requiring $\mathcal{O}(n)$ time [Bender and Farach-Colton (2000); Górecki and Eulenstein (2012)]. Every node (except the root) in $T(e_b)$ has a memory reference to its unique parent node. By using the memory reference to store the parent nodes of every node (except the root) and the data structure Q to determine which nodes have already been visited, the for-loops (lines 8 – 18) can be computable in $\mathcal{O}(n)$. The for-loops (23 – 26) are also computable in $\mathcal{O}(n)$ since $|V_{det}| < |n|$. Thus, the for-loop (lines 4 – 27) runs in time $\mathcal{O}(n^2)$. The outer for-loop (lines 1 – 29) is executed k times, from which the desired runtime follows. □

6.2 Restricted and Unrooted Tree Rooting

We define the Restricted and Unrooted Tree Rooting problem and propose the efficient algorithm for solving this problem. For a given profile of restricted and unrooted trees $T_1, \ldots, T_k$ and a strict consensus tree G of the profile, a rooted version of the input trees $T_1(e_1), \ldots, T_k(e_k)$ such that (i) each rooted tree $T_i(e_i)$ is the rooting tree of one unrooted tree T_i and (ii) their strict consensus tree $\mathcal{SC}(Q')$ is identical to the *rooted* guidance tree $G(e_r)$.

Problem 6.2.1. *(Restricted and Unrooted Tree Rooting Problem)*

Input: *An unrooted profile* $Q = \{T_1, \cdots, T_k\}$ *such that* $\mathcal{L}(T_i) = \mathcal{L}(Q)$ *and a rooting edge* $e_r \in E(G)$ *where* $G = \mathcal{SC}(Q)$.

Output: *A rooted profile* $Q' = \{T_1(e_1), \cdots, T_k(e_k)\}$ *such that* $G(e_r) = \mathcal{SC}(Q')$ *where* $e_i \in E(T_i)$.

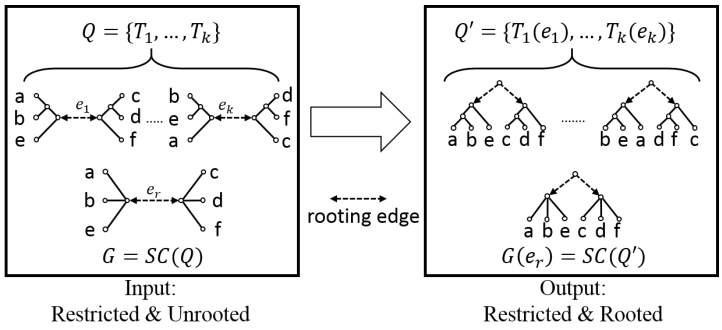

Figure 6.4: Schematic representation of theRestricted and Unrooted Tree Rooting problem. Unrooted input trees are transformed into the corresponding rooted trees.

Lemma 6.2.1. *Let* $Q = \{T_1, \cdots, T_k\}$ *be an unrooted profile,* $G = \mathcal{SC}(Q)$*, and* $e_r \in E(G)$ *be a rooting edge (defines* σ_r*). There exists an edge* $e_i \in E(T_i)$ *defines* σ_r *and* $G(e_r) = \mathcal{SC}(Q')$ *for the rooted profile* $Q' = \{T_1(e_1), \cdots, T_k(e_k)\}$.

Proof. Since $\Sigma(S) \subseteq \Sigma(T_i)$ $(\forall i\{1, \cdots, k\})$, there is an edge $e_i \in E(T_i)$ that defines σ_r. If T be an unrooted tree and $e_r \in E(T)$ be a rooting edge (defines $\sigma_r = A_r|B_r$), then there exists a bijective function $f : \Sigma(T) \setminus \{\sigma_r\} \to \mathcal{H}\big(T(e_r)\big) \setminus \{\mathcal{L}(T), A_r, B_r\}$ such that $f(\sigma) = A \vee B$ where $\sigma = A|B$. $\mathcal{H}\big(\mathcal{SC}(Q')\big) \setminus \{\mathcal{L}(Q'), A_r, B_r\} = \cap_i \mathcal{H}\big(T_i(e_i)\big) \setminus \{\mathcal{L}(Q'), A_r, B_r\} = \cap_i \big(\mathcal{H}\big(T_i(e_i)\big) \setminus \{\mathcal{L}(T_i), A_r, B_r\}\big)$

$= \cap_i\{f(\sigma_j) : \forall\sigma_j \in \Sigma(T_i) \setminus \{\sigma_r\}\} = \{f(\sigma_j) : \forall\sigma_j \in \cap_i\Sigma(T_i) \setminus \{\sigma_r\}\} = \{f(\sigma_j) : \forall\sigma_j \in \Sigma(G) \setminus \{\sigma_r\}\}$. Hence, there exists $e_i \in E(T_i)$ defines σ_r and $G(e_r) = \mathcal{SC}(Q')$. □

Algorithm 6 Restricted and Unrooted Tree Rooting(Q, G)

Input: An unrooted profile $Q = \{T_1, \cdots, T_k\}$ such that $\mathcal{L}(T_i) = \mathcal{L}(G)$ and a rooting edge $e_r \in E(G)$ where $G = \mathcal{SC}(Q)$.
Output: A rooted profile $Q' = \{T_1(e_1), \cdots, T_k(e_k)\}$ such that $G(e_r) = \mathcal{SC}(Q')$ where $e_i \in E(T_i)$.

1: $G' = G(e_r)$ ▷ σ_r is defined by removal of e_r
2: $\sigma_r = A_r|B_r$
3: **for all** $T \in Q$ **do**
4: **for all** $e \in E(T)$ **do**
5: $T' = T(e)$ ▷ σ is defined by removal of e
6: $\sigma = A|B$
7: **if** $(A_r = A \;\&\; B_r = B) \;||\; (A_r = B \;\&\; B_r = A)$ **then**
8: Add T' to Q'
9: Break
10: **end if**
11: **end for**
12: **end for**
13: **return** Q'

Proposition 6.2.1. *Algorithm 6 runs in $\mathcal{O}(kn)$ time where k is the number of trees in a profile P and $n = |\mathcal{L}(P)|$.*

Proof. Lemma 6.2.1 guarantees that there exists an split σ such that $\sigma = \sigma_r$ at line 7 and $\mathcal{SC}(Q') = G(e_r)$ at line 13. Assume σ is pre-computed when running Algorithm 5 and stored in a map data structure in the pair of edge key and split value. The σ which is defined by removal of e can be referenced in $\mathcal{O}(c)$, hence, the for-loop (lines 4 – 11) runs in time $\mathcal{O}(n)$. The outer for-loop (lines 1 – 29) is executed k times, from which the desired runtime follows. □

6.3 Experiments

We demonstrate the applicability of our new approach that adopts the Strict Consensus Approach to handle unrooted and unrestricted instances. While the Strict Consensus Approach is applicable to various median tree problems, we are choosing here the classic gene duplication problem [Eulenstein et al. (2010); Goodman et al. (1979)]. This problem is NP-hard [Ma et al. (2000)], $W[2]$-hard when parameterized by the gene duplication score [Bansal and Shamir (2011)] and hard to approximate better than a logarithmic factor [Bansal and Shamir (2011)].

While exact solutions for the gene duplication problem are coming into the reach of smaller phylogenetic studies (e.g., 22 taxa [Chang et al. (2013)]), for large-scale studies such solutions remain out of reach. Therefore, effective local search heuristics have been proposed [Bansal and Eulenstein (2013); Wehe and Burleigh (2010)], analyzed, and applied to compute large-scale median trees from gene trees [Cotton and Page (2002); Martin and Burg (2002); McGowen et al. (2008); Page (2000)]. Duptree [Wehe and Burleigh (2010)] is a standard implementation of these heuristics which we will use here for our comparative studies. All experiments were performed on a workstation with an Intel® Xeon® CPU E7-8837 @2.66GHz with 128GB RAM.

6.3.1 Empirical study

We evaluate the accuracy our new approach to solve the gene duplication problem in comparison with the program DupTree, using empirical data.

6.3.1.1 Data sets.

Multiple sequence alignments for 12 genes, each sampled from subsets of 2849 amphibian species, were obtained from the data base Dryad repository (published in [Pyron and Wiens (2011a,b)]). These 12 genes are C-X-C chemokine receptor type 4 (CXCR4), histone 3a (H3A), sodium-calcium exchanger (NCX1),
pro-opiomelanocortin (POMC), recombination-activating gene 1 (RAG1), rhodopsin (RHOD), seventh-in-absentia (SIA), solute-carrier family 8 (SLC8A3), tyrosinase (TYR), cytochrome b (cyt-b), and the subunits of the mitochondrial ribosome genes (12S/16S). We estimated unrooted gene trees for each of these alignments using the software RAxML version 8.2.9 [Stamatakis (2014)]. The command we used was: raxmlHPC -s ⟨phylip alignment file⟩ -n ⟨output suffix⟩ -m GTRGAMMA. These estimates were completed in 27 hours using 10 threads of our workstation. Finally, we combined the 12 trees, which are unrestricted and unrooted, into a profile P.

6.3.1.2 Experimental Setting.

For the DupTree approach we follow the procedure described in [Burleigh et al. (2011)] to root the trees in profile P. In this procedure two rooting methods were used, midpoint rooting [Boykin et al. (2010)] and optimal re-rooting [Burleigh et al. (2011)]. Then we run DupTree generating 500 estimates for each of the rooted profiles. The gene duplication scores for these estimates are summarized in the section A (midpoint rooting) and B (optimal re-rooting) of Fig. 6.5.

For our approach we computed the *unrooted guidance* tree G by using the SCM method [Huson et al. (1999)] for the profile P. By applying our Algorithm 4 to P and G, we computed a restricted version of P. Then we computed a rooted profile of P for each edge of G using Algorithm 6.

Now we applied the Strict Consensus Approach to the restricted and rooted profiles generated from P by using G. Again, we were running the Strict Consensus Approach on each of these profiles 500 times. The Strict Consensus Approach requires a value k_{max} that determines the maximum size (in the number of taxa) of the sub-problems that are solved exactly, while instances larger than k_{max} are addressed using the DupTree heuristic. According to the ability of our workstation we set k_{max} to 9. To summarize the results we considered only the best and the worst gene duplication scores under the 500 runs performed for each edge in G, which are depicted in sections C (worst) and D (best) of Fig. 6.5.

6.3.1.3 Results and Discussion.

We observe that the distributions of the best and the worst guidance tree rootings, shown in Fig. 6.5, are narrow and bell-shaped, reflecting that most sub-instances were solved exactly. Note that under the assumption that every sub-instance is solved exactly, there would be only one score in each of the sections C and D. An increased number of heuristically solved instances would spread more widely in terms of scores. In detail, these phenomena can be explained by the distributions of the node-degrees of the guidance tree G, which has 251 multifurcations, where 9 of them have a node-degree larger than k_{max}. Therefore, 96.4% of sub-instances were

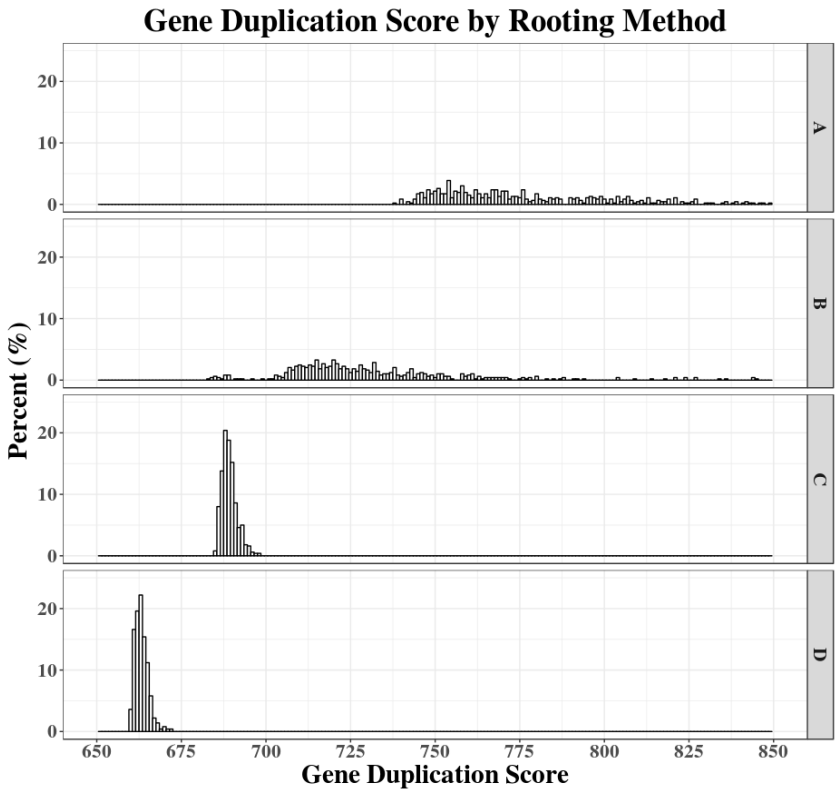

Figure 6.5: Gene duplication score comparison by the different rooting methods.
A: Gene trees midpoint rooting + DupTree heuristic.
B: Gene trees optimal re-rooting + DupTree heuristic.
C: Guidance tree rooting + Strict Consensus Approach (worst).
D: Guidance tree rooting + Strict Consensus Approach (best).

solved exactly, and only 3.6% of them were addressed heuristically. The overall distributions demonstrate that the guidance tree rooting together with the Strict Consensus Approach improves significantly on the accuracy and the variance when compared with the DupTree results. The descriptive statistics of the gene duplication scores are summarized in Table 6.1. The average time to compute the scores for A is 30 seconds, and for B is 80 minutes, and for C and D are each 50 seconds. Note, the much longer computing time for B is caused by the frequent re-rooting procedure of DupTree that when terminating in a local minima will re-root the input trees to proceed [Wehe et al. (2008)].

Table 6.1: Descriptive statistics of the gene duplication scores.
A: Gene trees midpoint rooting + DupTree heuristic.
B: Gene trees optimal re-rooting + DupTree heuristic.
C: Guidance tree rooting + Strict Consensus Approach (worst).
D: Guidance tree rooting + Strict Consensus Approach (best).

Group	Average	SD	Median	Min	Max	Range
A	785	42.8	771	738	1025	287
B	734	36.3	726	683	975	292
C	689	2.2	689	685	698	13
D	663	2.0	663	660	672	12

6.3.2 Simulated study

We evaluated the scalability of our approach when compared with DupTree for computing species trees by using simulated data sets with a maximum of 2744 taxa, where each instance contains 20 unrestricted and unrooted trees.

6.3.2.1 Data sets.

For the simulation *back-bone trees* were created that are unrooted b-ary trees for $b \in \{5, 6, 7\}$, such that each of these trees has a special *center node* where the path lengths between the center node and every leaf are always 4. Consequently, the numbers of leaves of the back-bone trees are 750, 1512, and 2744. For each back-bone tree, an *intermediate profile* consisting of 20 restricted and unrooted trees was computed, where each tree in the profile is a random refinement of the back-bone tree. We then converted the intermediate profiles into profiles P_b for $b \in \{5, 6, 7\}$ of unrestricted and unrooted trees by replacing each tree T in each of intermediate profiles by the tree $T|A$ for an independent random set $A \subseteq \mathcal{L}(T)$. The probability that a leaf is selected for the subset A is called the *scaffold factor*. We produced profiles $P_{b,25}$, $P_{b,50}$, and $P_{b,75}$ for each scaffold factor 25%, 50%, and 75%, and we adopted the initial back-bone trees as the guidance trees G_b.

6.3.2.2 Experiment Setting.

Algorithm 4 took $P_{b,s}$ and G_b as an input and produced the profile $Q_{b,s}$ of restricted and unrooted trees for each $b \in \{5, 6, 7\}$ and $s \in \{25, 50, 75\}$. Then we computed the *rooted guidance*

tree G'_b by using the midpoint edge of G_b. Algorithm 6 took $Q_{b,s}$ and G'_b as an input and produced the profile $Q'_{b,s}$ of restricted and rooted trees. By repeating the described procedure 500 times, we produced a total number of 4500 instances of restricted and rooted trees and its rooted guidance tree. We set $k_{max} = 9$, and therefore, all sub-instances were solved exactly, since the node-degrees of the guidance trees are less than k_{max}. For the baseline method, we generated the unrestricted and rooted profile from $P_{b,s}$ by using the midpoint rooting, and then run DupTree to generate 500 estimates.

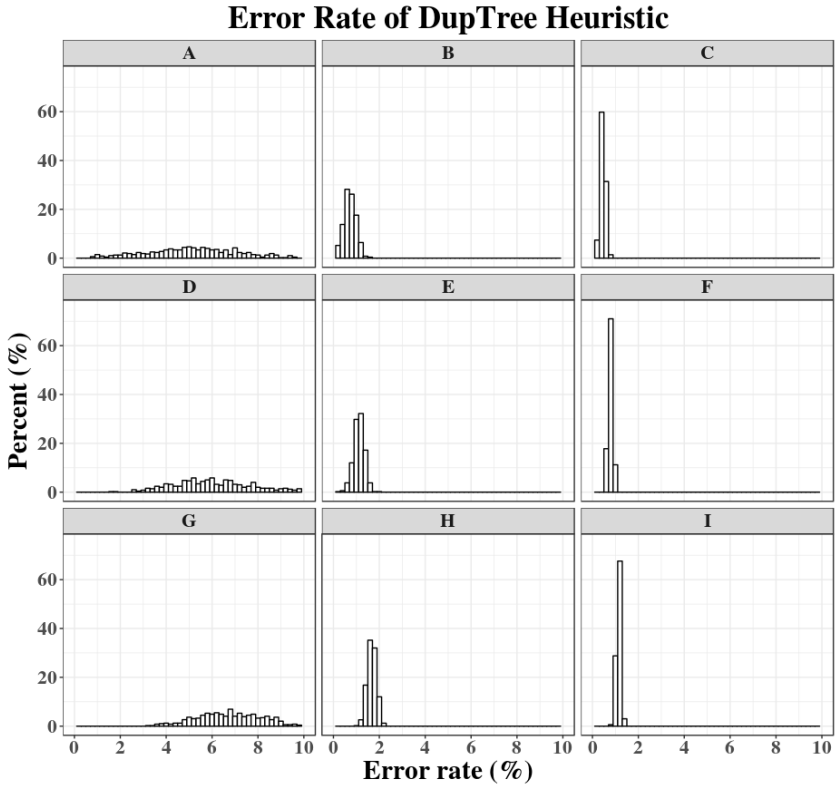

Figure 6.6: Error rate of DupTree heuristic by different taxa and scaffold factor.
Taxa=750 → A: Scaffold=25%, B: Scaffold=50%, C: Scaffold=75%
Taxa=1512 → D: Scaffold=25%, E: Scaffold=50%, F: Scaffold=75%
Taxa=2744 → G: Scaffold=25%, H: Scaffold=50%, I: Scaffold=75%

6.3.2.3 Results and Discussion.

Our approach computed optimal species trees, while DupTree estimated these trees heuristically. Fig. 6.6 shows that the error rates of DupTree are increasing with an increasing number of taxa. Observe that the error rates are significantly higher scored for the 25% (A,D,G) scaffold

factor than for the 50% (B,E,H) and 75% (C,F,I) scaffold factors. This suggests that the error rate of DupTree is greatly impacted by small scaffold factors. Fig. 6.7 shows that the gene duplication problem can be solved optimally in a reasonable time by using our approach.

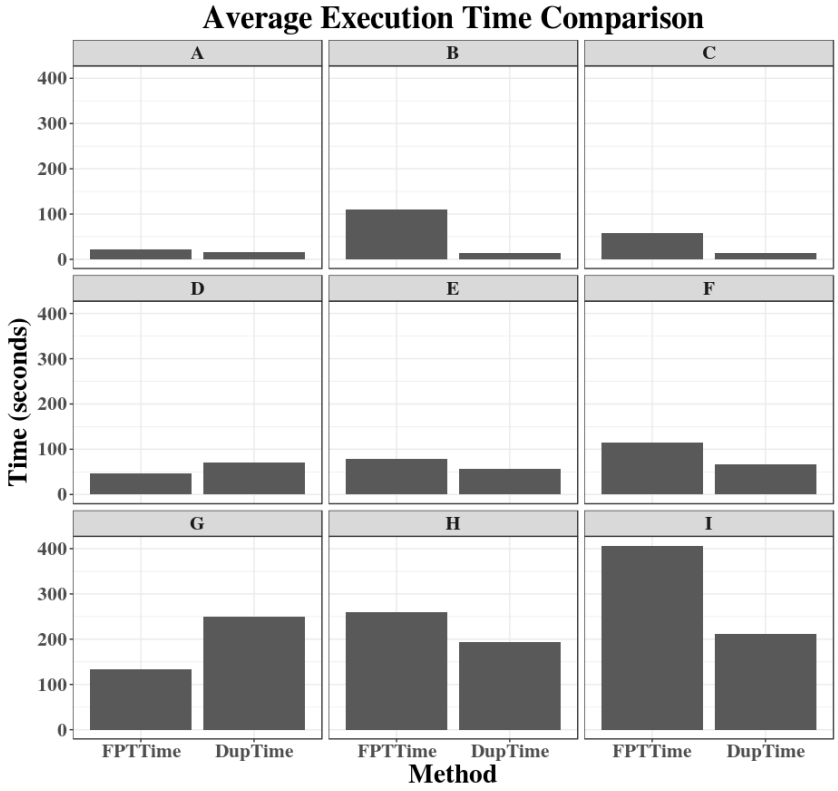

Figure 6.7: Average execution time of our approach (FTPTime) and DupTree heuristic (DupTime) by different taxa and scaffold factor.
Taxa=750 → A: Scaffold=25%, B: Scaffold=50%, C: Scaffold=75%
Taxa=1512 → D: Scaffold=25%, E: Scaffold=50%, F: Scaffold=75%
Taxa=2744 → G: Scaffold=25%, H: Scaffold=50%, I: Scaffold=75%

6.4 Conclusion

There is an increased interest in making NP-hard median tree problems available for large-scale species tree construction. While it has been demonstrated that the Strict Consensus Approach can successfully synthesize such trees, it is only able to handle restricted median problems, which have largely limited applicability in practice. In this work, we introduced a novel approach that is overcoming these limitations by adopting the Strict Consensus Approach to handle unrestricted median tree problems. Our studies on empirical and simulated exper-

iments demonstrate that our approach makes the Strict Consensus Approach a powerful tool for addressing large-scale supertree problems in their unrestricted version. While the approach is limited to median tree problems that are weak Pareto and satisfy the substructure property, many median tree problems have not been analyzed whether the satisfy these properties.

CHAPTER 7. GRAPH-THEORETIC APPROACH

We propose the first clique-based formulation of the RF median tree problem. To address the clique-based problem that is finding a minimum vertex weight clique in the compatibility graph, we devise a graph-theoretic heuristic that is different from standard median tree heuristics. As a preliminary step, an initial median tree is computed by a standard RF median tree heuristic. Our clique-based heuristic takes the initial median tree as the input, and transforms it to an equivalent starting clique in the compatibility graph. The heuristic then searches for an optimal candidate clique within the *local neighborhood* of the starting clique that is the set of all cliques into which this clique can be transform by a single *m-rooted split interchange (m-RSI)* operation. This procedure defines a *local search step*. The optimal clique found becomes the starting clique for the next local search step, and so on, until a local minima is found. Typically, the local search step is executed thousands of times for large-scale inputs, and hence, an efficient local search is imperative for the heuristic to terminate in a reasonable time. Therefore, we have devised an algorithm for the local search that runs in linear time in the size of the starting clique.

We demonstrate that our new approach utilized for the RF median tree problem improves significantly on accuracy when compared to the initial median tree by using empirical and simulated studies. In addition, the clique-based heuristic can also be applied to the deep coalescence, gene duplication, and gene duplication-loss problems by adapting this heuristic to the weighting functions corresponding to these problems.

Finally, we show that our clique-based heuristic can be interpreted as a standard local search heuristic for median trees that uses a novel tree edit operation, which we call m-rMSPR. This edit operation is fundamentally different from any of the classic edit operations used in standard local search heuristics, which are the rooted nearest neighbor interchange (rNNI) [Robinson

(1971); Moore et al. (1973)], rooted subtree prune and regraft (rSPR) [Swofford and Olsen (1990); Allen and Steel (2001); Bordewich and Semple (2005)], and rooted tree bisection and regraft (rTBR) [Swofford and Olsen (1990); Allen and Steel (2001); Chen et al. (2006)]. We proved that while the m-RSI neighborhood does not relate to the rSPR and rTBR neighborhoods by set containment, it properly contains the rNNI neighborhood.

7.1 Compatibility Graph

We defined a compatibility graph of rooted splits. The vertices of a compatibility graph correspond to the rooted splits, and an edge is drawn between two vertices if they are compatible. An example of a compatibility graph is depicted in Figure 7.1.

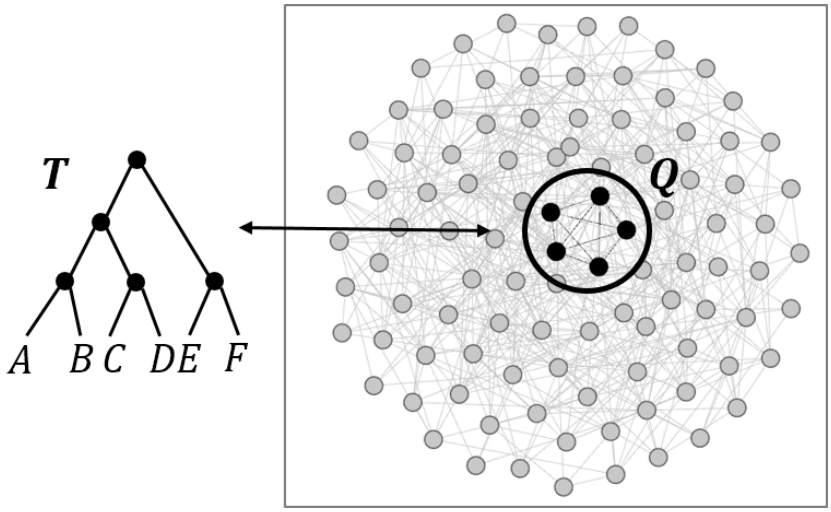

Figure 7.1: An example of the compatibility graph. The rooted binary tree T of 6 taxa is defined by the clique $\mathcal{Q}$ of 5 vertices in the compatibility graph, and conversely.

A pair of rooted splits are *compatible* if one contains the other, or they are disjoint.

Definition 7.1.1. *Let $\mathcal{X}$ be a set of n taxa. The* compatibility graph $CG(\mathcal{X})$ *on $\mathcal{X}$ has a vertex for the rooted split on $\mathcal{X}$, and there is an edge between two vertices if and only if the associated rooted splits are compatible.*

A rooted binary tree of n taxa is defined by $n-1$ pairwise compatible clusters of more than 1 taxa Semple and Steel (2003b). This fact acts as a lemma for Theorem 7.1.1.

Theorem 7.1.1. *Let $\mathcal{Q}$ be a set of $n-1$ rooted splits on $\mathcal{X}$ of n taxa. All pairs of rooted splits in $\mathcal{Q}$ are compatible if and only if there exists a binary rooted tree on $\mathcal{X}$ whose set of rooted splits is $\mathcal{Q}$.*

Proof. If two rooted splits $p_i = (A_i, B_i)$, $p_j = (A_j, B_j)$ in $\mathcal{Q}$ are compatible, then

$$(A_i \cup B_i) \cap (A_j \cup B_j) \in \{\emptyset, (A_i \cup B_i), (A_j \cup B_j)\}$$

Therefore, $\mathcal{Q}$ defines a set of $n-1$ pairwise compatible clusters, which is a hierarchy on $\mathcal{X}$. □

Corollary 7.1.1. *There exists a $(n-1)$-clique in the compatibility graph on $\mathcal{X}$ of n taxa if and only if there exists a binary rooted tree on $\mathcal{X}$ whose set of rooted splits is the vertex set of the $(n-1)$-clique.*

For rooted splits p_i and p_j, we say that p_i is *included* in p_j if $A_i \cup B_i \nsubseteq A_j$, $A_i \cup B_i \nsubseteq B_j$, and $(A_i \cup B_i) \subsetneq (A_j \cup B_j)$.

Definition 7.1.2. *Let $p_i = (A_i, B_i)$ be a rooted split on $\mathcal{X}$, $p_j = (A_j, B_j)$ be a rooted split on $\mathcal{Y}$, and $\mathcal{X} \subseteq \mathcal{Y}$. We define a function:*

$$inc(p_i, p_j) = \mathcal{I}[p_i \text{ is included in } p_j|\mathcal{X}]$$

where $p_j|\mathcal{X} = (A_j \cap \mathcal{X}, B_j \cap \mathcal{X})$ and $\mathcal{I}$ is the indicator function:

$$\mathcal{I}[condition] = \begin{cases} 1 & condition \text{ is true} \\ 0 & otherwise \end{cases}$$

Lemma 7.1.1. *Let g and S be rooted binary trees, $u \in V_{int}(g)$, $v \in V_{int}(S)$, and $\mathcal{L}(g) \subseteq \mathcal{L}(S)$. If $inc(\pi_g(u), \pi_S(v)) = 1$, then $\mathcal{M}(u) = v$.*

Proof. Let $w = \mathcal{M}(u)$. Since $inc\big(\pi_g(u), \pi_S(v)\big) = 1$, it follows that $\mathcal{C}_g(u) \subset (\mathcal{C}_S(v) \cap \mathcal{L}(g))$ and $w \leq v$. Suppose that $v \neq w$, then $\mathcal{C}_S(w)$ is a subset of the cluster of one of v's children. Hence, $inc\big(\pi_g(u), \pi_S(v)\big) = 0$ contradicts the assumption. □

Theorem 7.1.2. *Let g and S be rooted binary trees where $\mathcal{L}(g) \subseteq \mathcal{L}(S)$. For $u \in V_{int}(g)$,*

$$\sum_{v \in V_{int}(S)} inc(\pi_g(u), \pi_S(v)) \leq 1$$

Proof. If $inc(\pi_g(u), \pi_S(v)) = 1$, then there is no $w \in V_{int}(S)$ such that $w = \mathcal{M}(u)$ and $v \neq w$ by Lemma 7.1.1. Therefore, there exists at most one vertex v $(\forall v \in V_{int}(S))$ such that $inc(\pi_g(u), \pi_S(v)) = 1$. □

7.2 Robinson-Foulds Median Tree

The RF distance between two rooted trees is defined as the number of symmetric differences between the two sets of clusters of the two trees.

Definition 7.2.1. *(RF distance Chang et al. (2013)) Let g and S be rooted binary trees where $\mathcal{L}(g) \subseteq \mathcal{L}(S)$. The RF distance is defined:*

$$RF(g, S) = \sum_{u \in V_{int}(g)} \sum_{v \in V_{int}(S)} 2 \cdot \mathcal{I}[\mathcal{M}(u) = v \wedge \mathcal{C}_g(u) \neq \mathcal{C}_S(v) \cap \mathcal{L}(g)]$$

Proposition 7.2.1. *Let g and S be rooted binary trees where $\mathcal{L}(g) \subseteq \mathcal{L}(S)$. Then*

$$RF(g, S) = \sum_{u \in V_{int}(g)} \sum_{v \in V_{int}(S)} 2 \cdot inc\big(\pi_g(u), \pi_S(v)\big)$$

Proof. If $inc\big(\pi_g(u), \pi_S(v)\big) = 1$, then $\mathcal{M}(u) = v$ by Lemma 7.1.1 and $\mathcal{C}_g(u) \subsetneq \mathcal{C}_S(v) \cap \mathcal{L}(g)$ by the definition of the function *inc*. Hence, $inc\big(\pi_g(u), \pi_S(v)\big) = \mathcal{I}[\mathcal{M}(u) = v \wedge \mathcal{C}_g(u) \neq \mathcal{C}_S(v) \cap \mathcal{L}(g)]$. □

Let $\mathcal{G} = \{g_1, g_2, \cdots, g_k\}$ be a profile of rooted binary trees and $\mathcal{L}(\mathcal{G}) = \mathcal{X}$. Then,

$$RF(\mathcal{G}, S) = \sum_{i}^{k} RF(g_i, S)$$

We constructed the compatibility graph $CG(\mathcal{X})$ on $\mathcal{X}$ and calculated the weight of each vertex z, denoted by $W_{inc}(z)$, to be the total number of rooted splits in $\mathcal{G}$ that are included in z. Formally,

$$W_{inc}(z) = \sum_{i}^{k} \sum_{u \in V_{int}(g_i)} 2 \cdot inc(\pi_{g_i}(u), z)$$

If $\mathcal{Q}$ is a set of vertices in $CG(\mathcal{X})$, then total weight of $\mathcal{Q}$ is defined as $W_{inc}(\mathcal{Q}) = \sum_{z \in \mathcal{Q}} W_{inc}(z)$. The optimal RF median tree is found by computing a $(n-1)$-clique $\mathcal{Q}$ so as to minimize the $W_{inc}(\mathcal{Q})$.

Theorem 7.2.1. *Let $\mathcal{G} = \{g_1, g_2, \cdots, g_k\}$ be a profile of rooted binary trees, $\mathcal{L}(P) = \mathcal{X}$, and $|\mathcal{X}| = n$. If $\mathcal{Q}$ is the $(n-1)$-clique in $CG(\mathcal{X})$ minimizing $W_{inc}(\mathcal{Q})$, and S is the rooted binary tree that is defined by the clique $\mathcal{Q}$, then S minimizes RF distance $RF(\mathcal{G}, S)$.*

Proof. By Corollary 7.1.1, the $(n-1)$-clique in the compatibility graph defines the rooted binary tree S on $\mathcal{X}$ that has $n-1$ internal vertices. Then,

$$W_{inc}(\mathcal{Q}) = \sum_{z \in \mathcal{Q}} W_{inc}(z) = \sum_{z \in \mathcal{Q}} \sum_{i}^{k} \sum_{u \in V_{int}(g_i)} 2 \cdot inc(\pi_{g_i}(u), z)$$

$$= \sum_{i}^{k} \sum_{u \in V_{int}(g_i)} \sum_{z \in \mathcal{Q}} 2 \cdot inc(\pi_{g_i}(u), z) = \sum_{i}^{k} \sum_{u \in V_{int}(g_i)} \sum_{v \in V_{int}(S)} 2 \cdot inc\big(\pi_{g_i}(u), \pi_S(v)\big)$$

$$= \sum_{i}^{k} RF(g_i, S) = RF(\mathcal{G}, S)$$

Therefore, the minimum weight clique defines the rooted binary tree S that minimize $RF(\mathcal{G}, S)$.

□

7.3 Rooted Split Interchange

In this section, we introduced a clique edit operation, called m-RSI, and a tree edit operation, called m-rMSPR. Examples of the operations are depicted in Figure 7.2.

Definition 7.3.1. *(m-rooted splits interchange (m-RSI)) Let sets $\mathcal{Q}$ and $\mathcal{Q}'$ be $(n-1)$-cliques in the compatibility graph $CG(\mathcal{X})$ on $\mathcal{X}$ of n taxa. A m-RSI between $\mathcal{Q}$ and $\mathcal{Q}'$ occurs if $\mathcal{Q} \neq \mathcal{Q}'$ and $|\mathcal{Q} \setminus \mathcal{Q}'| = |\mathcal{Q}' \setminus \mathcal{Q}| \leq m$. In order words, $\mathcal{Q}'$ is obtained from $\mathcal{Q}$ by deleting at most m vertices, adding the same number of vertices in $CG(\mathcal{X})$.*

Definition 7.3.2. *(m-rooted multiple subtrees prune and regraft (m-rMSPR)) Let T, T' be the rooted binary trees that are defined by $(n-1)$ cliques $\mathcal{Q}$, $\mathcal{Q}'$ in the compatibility graph $CG(\mathcal{X})$ on $\mathcal{X}$ of n taxa. A m-rMSPR between T and T' occurs if $\mathcal{Q}$ can be transformed from $\mathcal{Q}'$ by a single m-RSI operation, and conversely.*

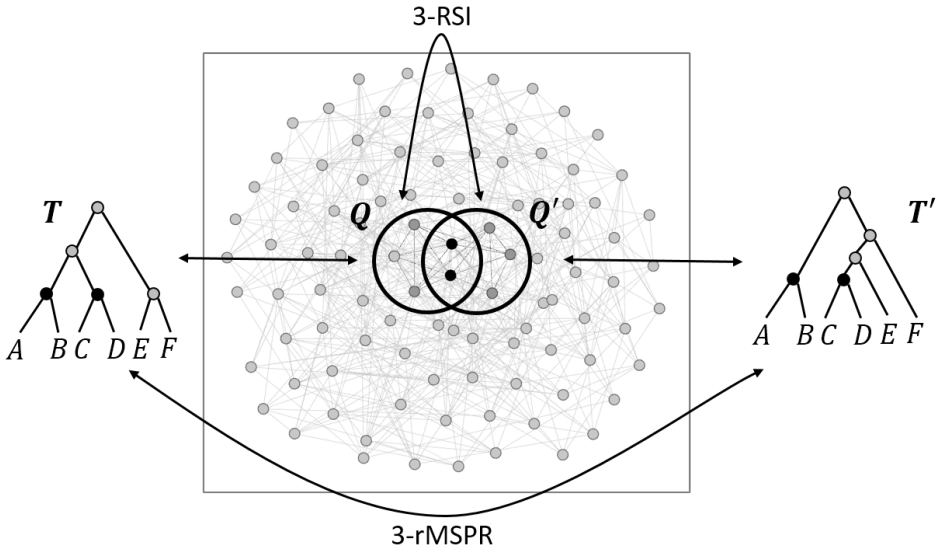

Figure 7.2: An example of 3-RSI clique edit operation and 3-rMSPR tree edit operation. Two rooted binary trees T, T' of 6 taxa are defined by the two cliques $\mathcal{Q}$, $\mathcal{Q}'$ (respectively) of 5 vertices in the compatibility graph. The 3-RSI operation between $\mathcal{Q}$ and $\mathcal{Q}'$ is correlated with the 3-rMSPR operation between T and T'.

Proposition 7.3.1. 1-*RSI (*1-*rMSPR) does not exist.*

Proof. Let S be the rooted binary tree that is defined by $\mathcal{Q}$ and $u \in V_{int}(S)$. Then, $\pi_S(u)$ is deterministic by rooted splits of u's children and parent. □

Proposition 7.3.1 acts as a lemma for Theorem 7.3.1.

Theorem 7.3.1. *Let set $\mathcal{Q}$ be $(n-1)$-clique in the compatibility graph $CG(\mathcal{X})$ on $\mathcal{X}$ of n taxa and S be the rooted binary trees defined by $\mathcal{Q}$. Suppose that a $(n-1)$-clique $\mathcal{Q}'$ is obtained from $\mathcal{Q}$ by the single m-RSI operation. If $\pi_S(u) \in \mathcal{Q}, (u \in V_{int}(S))$ is replaced, then there exists $\pi_S(v) \in \mathcal{Q}, (u \neq v \in V_{int}(S))$ which is also replaced and u-v has a parent-child relationship. (i.e., $u = Ch_S(v)$ or $v = Ch_S(u)$).*

Proof. Suppose that $\pi_S(u) = (A, B)$ is replaced by the different rooted split (A', B') without changing any of $\pi_S(l)$, $\pi_S(r)$, and $\pi_S(Pa_S(u))$ where l and r are two children of u. Then, (A', B') is not compatible with at least two rooted splits among $\pi_S(l)$, $\pi_S(r)$, and $\pi_S(Pa_S(u))$. Thus, $\mathcal{Q}'$ is not a $(n-1)$-clique, then this is contradiction. □

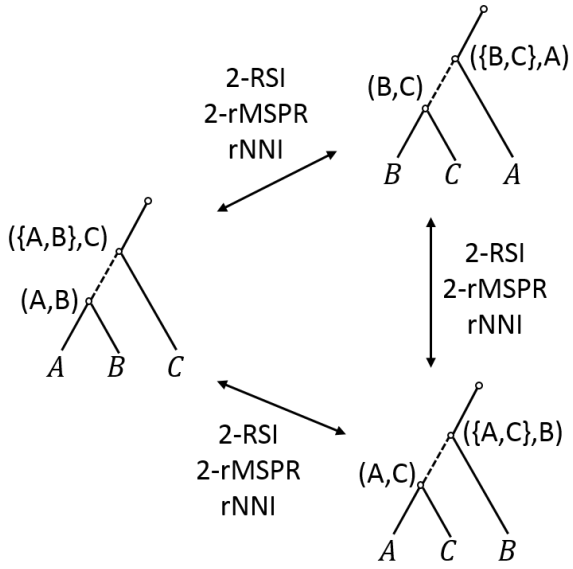

Figure 7.3: A cycle of 2-RSI (2-rMSPR) operations. Each operation can be considered as a rNNI operation. A dotted line denotes an *axis* of rNNI operation, and the two rooted splits of incident vertices of the axes are replaced in the correlated cliques.

An internal vertex u of a rooted binary tree T has two incident edges that connects its children l and r. A rooted binary tree T' is obtained from T by deleting $e = \{u, l\}$ (or $e' = \{u, r\}$), adding the edge between l (or r) and the vertex subdivides the edge that is incident with $Pa_T(u)$ and u's sibling, and then suppressing any degree-two vertices [Moore et al. (1973); Robinson (1971)]. This tree edit operation is called rooted nearest neighbor interchange (rNNI). An example is depicted in Figure 7.3.

Proposition 7.3.2. $rNNI = 2\text{-}RSI$ $(2\text{-}rMSPR)$

Proof. Let set $\mathcal{Q}$ be $(n-1)$-clique in a compatibility graph $CG(\mathcal{X})$ and S be the rooted binary tree that is defined by $\mathcal{Q}$. Suppose that a $(n-1)$-clique $\mathcal{Q}'$ is obtained from $\mathcal{Q}$ by 2-RSI with the two rooted splits $\pi_S(u), \pi_S(v)$ where $u, v \in V_{int}(S)$. Then, $u = Ch_S(v)$ or $v = Ch_S(u)$ by Theorem 7.3.1. That is, the edge $e = \{u, v\}$ is internal. Hence, the rNNI operation of an internal edge (axis) can be interpreted as the 2-RSI operation between the rooted splits of the incident vertices of the axis. Figure 7.3 shows that a cycle of 2-RSI (2-rMSPR) operations is equivalent with a cycle of rNNI operations. □

Let T be a rooted binary tree, $e = \{u, v\}$ and $u \leq_T v$. A rooted binary tree T' is obtained from T by deleting e, adding the edge between u and the vertex that subdivides the edge of $T \backslash e$, and then suppressing any degree-two vertices. This tree edit operation is called rooted subtree prune and regraft (rSPR) [Allen and Steel (2001); Bordewich and Semple (2005); Swofford and Olsen (1990)]. Analogously, T' is obtained from T by deleting e, adding an edge between vertices such that each of the vertices subdivides the edge of one and the other component of $T \setminus e$, and then suppressing any degree-two vertices. This tree edit operation is called rooted tree bisection and regraft (rTBR) [Allen and Steel (2001); Chen et al. (2006); Swofford and Olsen (1990)].

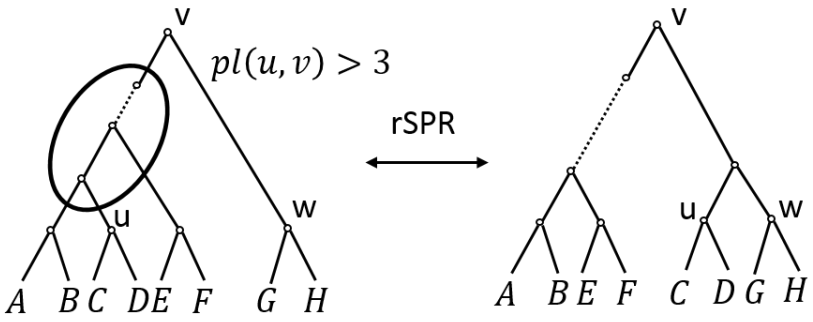

Figure 7.4: An example of a rSPR operation. The figure shows that $rSPR, rTBR \nsubseteq$ 3-RSI (3-$rMSPR$).

Proposition 7.3.3. 3-RSI (3-$rMSPR$) $\neq rSPR, rTBR$

Proof. The proof of $rSPR, rTBR \nsubseteq$ 3-RSI is illustrated at Figure 7.4. Suppose that a rooted binary tree S' is obtained from a rooted binary tree S by a rSPR operation that prunes the subtree rooted at u and regrafts the subtree to the edge incident with v and w. If $pl_S(u, v) > 3$ and $pl_S(u, w) > 3$, then the $(n-1)$-clique $\mathcal{Q}'$ that defines S' cannot be obtained from the $(n-1)$-clique $\mathcal{Q}$ that defined S by any 3-RSI operation. Hence, $rSPR, rTBR \nsubseteq$ 3-RSI. Figure 7.5 shows the proof of 3-$RSI \nsubseteq rSPR, rTBR$ by a counter example $((A, B), (C, D)) \leftrightarrow ((A, D), (B, C))$. □

Corollary 7.3.1.

$$rNNI \subseteq m\text{-}RSI\ (m\text{-}rMSPR) \neq rSPR, rTBR, (\forall m > 1)$$

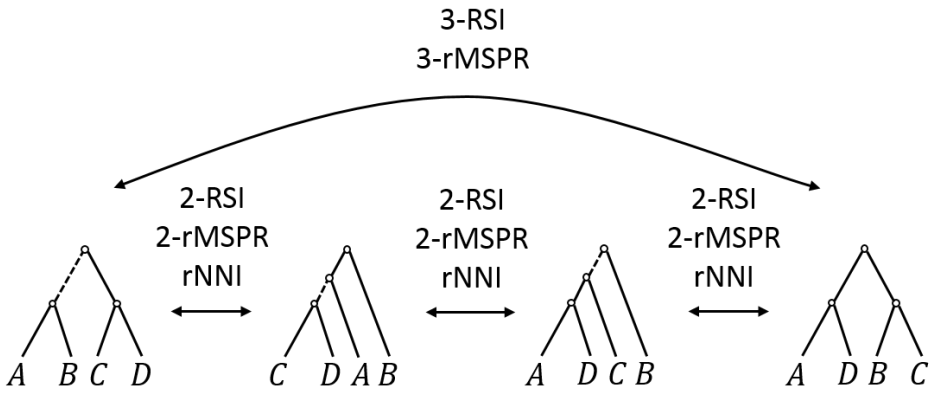

Figure 7.5: An example of the resulting trees by the 2 and 3-RSI (rMSPR) operations. A dotted line denotes an *axis* of the NNI operation. $((A,B),(C,D)) \leftrightarrow ((A,D),(B,C))$ shows that 3-RSI (3-$rMSPR$) $\not\subseteq rSPR, rTBR$.

7.4 Local Search Problem

We provided a definition for the 3-RSI operation and formulated the related local search problem in the compatibility graph of rooted splits.

Definition 7.4.1. *(3-RSI based local search) Given a rooted binary tree S and a vertex $u \in V_{int}(S)$, 3-$RSI_S(u)$ is a set of rooted binary trees obtained as follows ($\mathcal{X} = \mathcal{L}(S), n = |\mathcal{X}|$):*

1. *Construct the $(n-1)$-clique $\mathcal{Q}$ that defines S in the compatibility graph $CG(\mathcal{X})$.*
2. *Form the set of internal vertices R by adding u and u's adjacent vertices such that $|R| \geq 2$. (ignore adjacent leaf.)*
3. *Find all $(n-1)$-cliques $\mathcal{Q}'$ in $CG(\mathcal{X})$ that $|\mathcal{Q} \setminus \mathcal{Q}'| \leq 3$ and $\mathcal{Q} \setminus \mathcal{Q}' \subseteq \pi_S(R)$.*
4. *Return every S' that is defined by each $\mathcal{Q}'$ to 3-$RSI_S(u)$.*
5. *Repeat the above process for other possible R.*

In addition, we use the following notation

$$3\text{-}RSI_S = \cup_{u \in V_{int}} 3\text{-}RSI_S(u)$$

3-RSI_S is called the 3-RSI *neighborhood* of the rooted binary tree S, and $|3\text{-}RSI_S| = \sum_{n-1} \mathcal{O}(1) = \mathcal{O}(n)$.

Problem 7.4.1. *(RF local search)*

Instance: *A profile $\mathcal{G}$ and a median tree S for $\mathcal{G}$.*

Find: $S' = \underset{S' \in 3\text{-}RSI_S}{\operatorname{argmin}} RF(\mathcal{G}, S')$

We introduced Algorithm 7 that solves the RF local search problem in linear time to the size of the starting clique that is defined by the initial median tree.

Algorithm 7 3-RSI($\mathcal{G}$, S)

Instance: A profile $\mathcal{G}$ and a median tree S for $\mathcal{G}$.
Find: $S' = \underset{S' \in 3\text{-}RSI_S}{\operatorname{argmin}} RF(\mathcal{G}, S')$

1: $S' = S$
2: **for** all $u \in V_{int}(S)$ **do**
3: construct a clique $\mathcal{Q}$ that defines S' ($S' \to \mathcal{Q}$)
4: **for** 3 pairs of u' adjacent v, w **do**
5: $R = \{u, v, w\}$
6: remove leaf vertices from R
7: **if** $|R| \geq 2$ **then**
8: $L = \cup_{x \in R} \mathcal{C}_S(x)$
9: $D = \cup_{x \in R} Ch_S(x) \setminus R$ for internal $Ch_S(x)$
10: construct a compatibility graph CG'
with rooted splits (A, B) such that
$A \cup B \subseteq L$ and $\exists x \in D, \mathcal{C}_S(x) \subsetneq A \cup B$
11: set weights of rooted splits in CG',
that is $W_{inc}(z), \forall z \in V(CG')$
12: Find $|R|$-clique $\mathcal{Q}'$ as to minimize $W_{inc}(\mathcal{Q}')$
13: **if** $W_{inc}(\mathcal{Q}') < W_{inc}(\pi_S(R))$ **then**
14: replace $\pi_S(R)$ in $\mathcal{Q}$ with $\mathcal{Q}'$
15: build S' that is defined by $\mathcal{Q}$ ($\mathcal{Q} \to S'$)
16: **end if**
17: **end if**
18: **end for**
19: **end for**
20: **return** S'

Theorem 7.4.1. *Algorithm 7 is correct, and runs in $\mathcal{O}(kn)$ time where $n = |\mathcal{L}(\mathcal{G})|$, and k is the number of trees in $\mathcal{G}$.*

Proof. For the correctness, observe that the condition of rooted splits in line 10 ($A \cup B \subseteq L$ and $\exists x \in D, \mathcal{C}_S(x) \subsetneq A \cup B$). This condition grantees that any rooted splits in CG' is compatible with all rooted splits in $\mathcal{Q} \setminus \pi_S(R)$. There are two cases of vertex $y \in V_{int}(S) \setminus R$. That is (i)

$x \leq_S y$ or $y \leq_S x$ $\forall x \in R$, and (ii) others. Hence, (i) satisfies *contains*, and (ii) satisfies *disjoint*. Furthermore, the number of CG' is fixed by this condition. Therefore, the algorithm requires $\mathcal{O}(kn)$ time to repeat 3-RSI based local search for all internal vertices. □

Algorithm 7 can also be applied to the deep coalescence, gene duplication, and gene duplication-loss problems by adapting the different weighting functions corresponding to these problems. The weighting function of the deep coalescence problem was introduced by Than and Nakhleh [Than and Nakhleh (2009b)], and the functions of the gene duplication and gene duplication-loss problems were proposed by Bayzid et al. (2013).

7.5 Experiments

We evaluate the performance of our clique-based heuristic, called 3-RSI (see Algorithm 7), for the RF median tree problem. As a preliminary step, a median tree is computed by the RF heuristic RF-SPR from Bansal et al. (2010b) that is then used as the initial input for our 3-RSI heuristic. Throughout the experiments, RF-SPR median tree denotes the initial median tree, and 3-RSI median tree denotes the output tree of our 3-RSI local search heuristic. We compared the RF distances between 3-RSI median trees and RF-SPR median trees by using published empirical data sets and simulated data sets. For statistical analyses, the Wilcoxon signed-rank test (non-parametric test) was used to compare the median value differences of RF distances between matched pairs of RF-SPR and 3-RSI median trees to alleviate the normal distribution assumption. All of our experiments were performed on an Intel® Core™ i7 860 2.80GHz workstation with 8GB RAM.

7.5.1 Empirical study

Published chloroplast 5S ribosomal RNA data set of 118 total taxa from the comparative RNA web site [Cannone et al. (2002)] is used for the empirical study. 100 input trees are inferred from the data set using the maximum parsimony approach that is implemented in Parsimonator [Stamatakis (2014)]. We compared the performance between RF-SPR and 3-RSI median trees by executing both heuristics $1,000$ times for the data set.

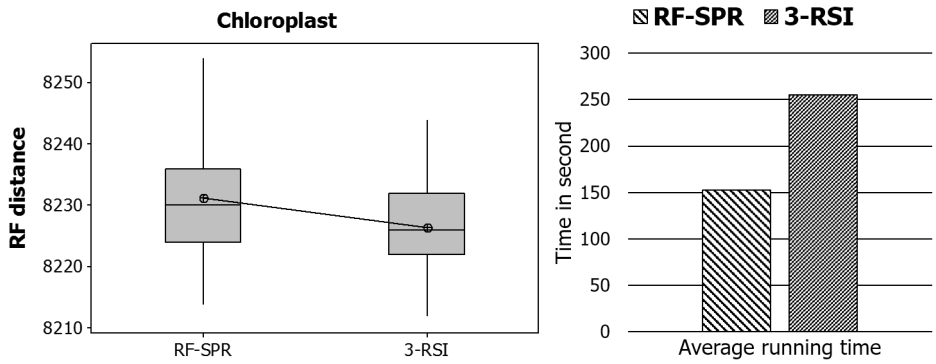

Figure 7.6: The box plot of $RF(\mathcal{G}, \text{RF-SPR})$ and $RF(\mathcal{G}, \text{3-RSI})$, and the bar chart of average running time. The mean value is denoted by the symbol $\oplus$, the median value is represented by the band inside the box, and the first and third quartiles are represented by the bottom and top of a box respectively.

Figure 7.6 depicts RF distances of 3-RSI and RF-SPR median trees. RF distances of 3-RSI median trees for both data sets were significantly lower than the ones of RF-SPR median trees (p-value < 0.05). The average running time of 3-RSI is 259 seconds and that of RS-SPR is 158 seconds. In addition, 95% confidence intervals (CI) for median value differences of the Wilcoxon signed-rank test are summarized in Table 7.1.

7.5.2 Simulated study

We used the exact algorithm [Chang et al. (2013)] for the RF median tree problem to analyze optimality of our 3-RSI heuristic. Due to the fact that the Pareto property for clusters is satisfied for the RF median tree problem [Bryant (2003)], this problem can be solved exactly by using the strict consensus approach for profiles of complete rooted binary gene trees (i.e., trees with the same label set) analogous to the deep coalescence [Lin et al. (2011)] and the gene duplication problem [Moon et al. (2016)].

The strict consensus approach is described as follows. A given profile is divided into an equivalent set of its subprofiles based on the strict consensus tree of the given profile. Then, subsolutions are computed by solving the subpropiles and the solution is obtained by refining the strict consensus tree by the subsolutions [Moon et al. (2016)]. This approach is limited by the maximum out-degree (k_{max}) of the strict consensus tree of the given profile. We restricted the maximum out-degree ($k_{max} = 10$) to compute the exact solution in a reasonable time.

For the strict consensus approach, the guidance-trees were created as b-ary trees with depth d, for the number of taxa $b^d = \{8^2, 9^2, 10^2, 5^3\}$. For each guidance-tree, a profile $\mathcal{G}$ consisting of 100 rooted binary trees was computed, where each tree in the profile is a random refinement of the guidance tree. With this profile $\mathcal{G}$, we compared the RF distances of RF-SPR and 3-RSI median trees. The described procedure is repeated $1,000$ times for each number of taxa.

Figure 7.7 depicts RF distances of 3-RSI, RF-SRP, and the exact solution median trees. RF distances of 3-RSI median trees for all numbers of taxa were significantly lower than the ones of RF-SPR median trees (p-value < 0.05). In detail, 95% confidence intervals (CI) for median value differences of the Wilcoxon signed-rank test are summarized in Table 7.1. Figure 7.7 also shows that 3-RSI has a longer average running time compared to RF-SPR (about 2-3 times) for all number of taxa.

Table 7.1: Median value of RF scores and 95% confidence intervals (CI) for median value difference of the Wilcoxon signed-rank test.

	Median of RF		95% CI
Taxa	RF-SPR	3-RSI	
Chloroplast	8230	8226	(5.0, 6.0)
$64 (= 8^2)$	9744	9740	(2.0, 3.0)
$81 (= 9^2)$	12828	12826	(1.0, 2.0)
$100 (= 10^2)$	16292	16288	(5.0, 5.0)
$125 (= 5^3)$	15126	15124	(3.0, 3.0)

Figure 7.8 depicts distributions of improvement ratios of 3-RSI median trees compared to RF-SPR median trees. An improvement ratio ($0 \sim 100\%$) is calculated by following formula:

$$\frac{RF(\mathcal{G}, \text{RF-SPR}) - RF(\mathcal{G}, \text{3-RSI})}{RF(\mathcal{G}, \text{RF-SPR}) - RF(\mathcal{G}, \text{Exact Solution})}(\%)$$

In conclusion, 100% of the ratio means that 3-RSI heuristic improves RF-SPR median tree optimally.

7.6 Conclusion

Inferring large-scale phylogenetic trees accurately is one of the grand challenges in computational biology. Addressing this challenge, standard local search heuristics for the NP-hard RF

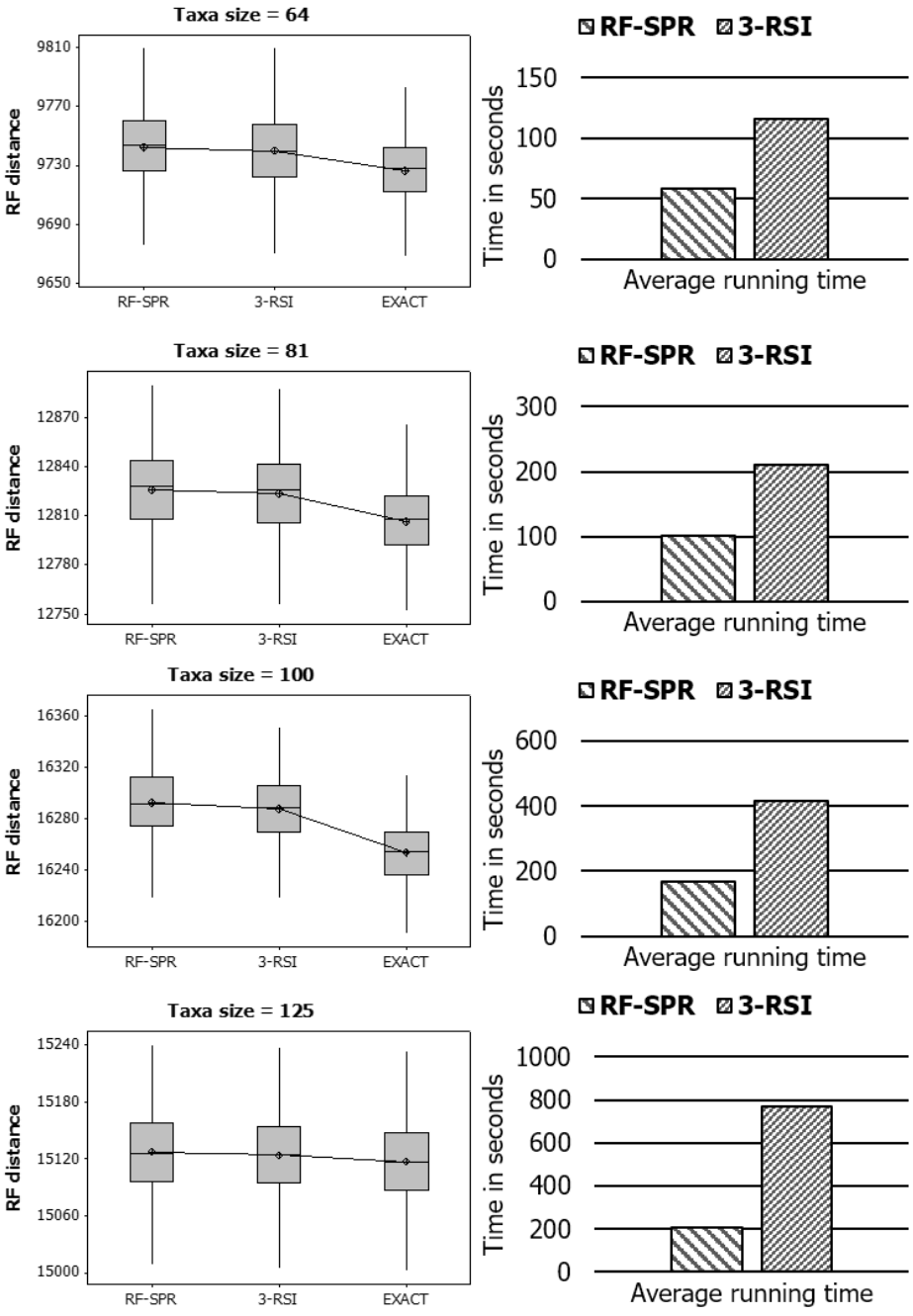

Figure 7.7: Box plots of $RF(\mathcal{G}, \text{RF-SPR})$, $RF(\mathcal{G}, \text{3-RSI})$, and $RF(\mathcal{G}, \text{Exact Solution})$. The mean, median, first quartile, and third quartile are represented as in Figure 7.6.

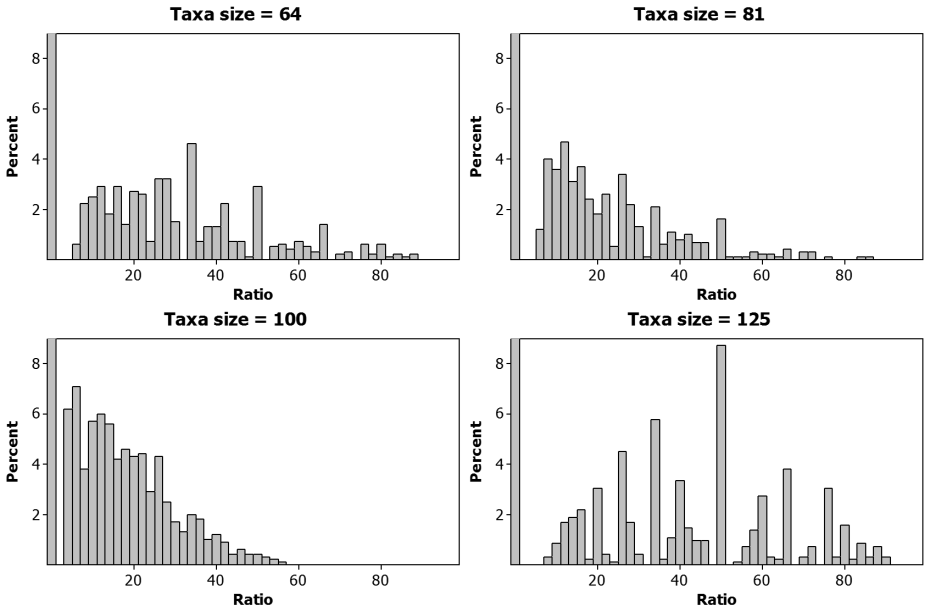

Figure 7.8: Distributions of improvement ratios of 3-RSI median tree compared to RF-SPR median tree. An improvement ratio ($0 \sim 100\%$) is calculated by dividing a difference between $RF(\mathcal{G}, \text{RF-SPR})$ and $RF(\mathcal{G}, \text{3-RSI})$ by a difference between $RF(\mathcal{G}, \text{RF-SPR})$ and $RF(\mathcal{G}, \text{Exact Solution})$. Therefore, 100% of the ratio means that 3-RSI heuristic improves RF-SPR median tree optimally.

median tree problem have provided credible results. However, our studies using exact solutions for large-scale trees have shown that these heuristics can have a significant inaccuracy (see Figure 7.8). Consequently, this work describes a new RF heuristic that is fundamentally different from current standard heuristics for the RF median tree problem.

To develop this new RF heuristic, we introduce a graph-theoretic formulation of the RF median tree problem where optimal trees relate to minimum vertex weight cliques in a compatibility graph. Therefore, our clique-based heuristic is searching for optimal cliques in this graph, in contrast to standard local search heuristics that search the space of all candidate median trees. As we demonstrate, while the clique-based heuristic does not provide exact results, it improves significantly on the RF median tree estimates resulting from the standard RF heuristics. Clearly, our clique-based heuristic can only improve on the standard heuristics, as it is initialized with the RF median tree estimates provided by the standard heuristics.

Investigating how the clique-based heuristic and standard local search heuristics may relate, we found that the clique-based heuristic can be equivalently expressed as a standard local search

heuristic albeit using a distinct tree edit operation that we introduce. This result also shows that our clique-based heuristic has the ability to improve on RF median tree estimated of standard local searches.

CHAPTER 8. CLUSTER MATCHING DISTANCE

We propose the first cluster matching distance between two rooted phylogenetic trees. We define a complete weighted bipartite graph $G = (X, Y, E)$ from two rooted trees T_1 and T_2 where every cluster in T_1 and T_2 is represented by a node in X and Y. The weight of an each edge $e = \{u, v\}$ is defined as the cardinality of the symmetry difference between the clusters u and v. Our cluster matching distance between trees T_1 and T_2 is the weight of the minumum-weight perfect matching in the complete weight bipartite graph G.

We demonstrate that our new distance measure induces a metric on the spaces of trees. In addition, we show the bound on the diameter with respect to the distance measure and the bound on the change in the distance measure caused by a single tree edit operations such as rNNI, rSPR, and rTBR.

8.1 Cluster Matching Distance

Definition 8.1.1. *(Robinson-Foulds Distance) Given two trees, T_1 and T_2 on the same set of leaves,*

$$RF(T_1, T_2) = \frac{1}{2}\big((\|\mathcal{H}(T_1) \setminus \mathcal{H}(T_2)|) + (|\mathcal{H}(T_2) \setminus \mathcal{H}(T_1)|)\big).$$

Given two trees, T_1 and T_2 on the same set of leaves, we define a complete weighted bipartite graph $G = (V_{int}(T_1) \cup V_{int}(T_2), E)$. We denote this complete bipartite graph by $B(T_1, T_2)$.

Definition 8.1.2. *(Cluster Matching Distance) We set the weight of each edge $e = \{u, v\}$ in $B(T_1, T_2)$ to*

$$W(u, v) = |\mathcal{C}_{T_1}(u) \ominus \mathcal{C}_{T_2}(v)|$$

$$= |\big(\mathcal{C}_{T_1}(u) \setminus \mathcal{C}_{T_2}(v)\big) \cup \big(\mathcal{C}_{T_2}(v) \setminus \mathcal{C}_{T_1}(u)\big)|$$

The cluster matching distance $CM(T_1, T_2)$ *between* T_1 *and* T_2 *is the weight of the minimum weight perfect matching in* $B(T_1, T_2)$.

Let n be the number of nodes in $B(T_1, T_2)$, i.e., $n = |V\big(B(T_1, T_2)\big)|$. Computing the weights of edges in $B(T_1, T_2)$ requires $\mathcal{O}(n^3)$ time by using vector representation of clusters, and the minimum-weight perfect matching problem can be solved by using Kuhn-Munkres algorithm, which is also asymptotically bounded by $\mathcal{O}(n^3)$ Kuhn (1955); Munkres (1957).

8.1.1 A new metric space

Lemma 8.1.1. *The cluster matching distance is a metric. For any rooted trees* T_i, T_j, *and* T_k *on same leaves,*

1. $CM(T_i, T_j) \geq 0$
2. $CM(T_i, T_j) = 0$ *if and only if* $T_i = T_j$
3. $CM(T_i, T_j) = CM(T_j, T_i)$
4. $CM(T_i, T_k) \leq CM(T_i, T_j) + CM(T_j, T_k)$

Proof. Properties 1, 2, and 3 follow directly from Definition 8.1.2. Suppose that M_{ij} and M_{jk} are the minimum weight perfect matching in $B(T_i, T_j)$ and $B(T_j, T_k)$. Now, construct a matching M_{ik} in $B(T_i, T_k)$ such that $M_{ik} = \{(u, w) | (u, v) \in M_{ij} \wedge (v, w) \in M_{jk}\}$. For u, v, and w, $W(u, w) \leq W(u, v) + W(v, w)$. We have

$$
\begin{aligned}
CM(T_i, T_k) &\leq \sum_{(u,w)\in M_{ik}} W(u, w) \\
&\leq \sum_{(u,v)\in M_{ij}, (v,w)\in M_{jk}} (W(u, v) + W(v, w)) \\
&= \sum_{(u,v)\in M_{ij}} W(u, v) + \sum_{(v,w)\in M_{jk}} W(v, w) \\
&= CM(T_i, T_j) + CM(T_j, T_k)
\end{aligned}
$$

□

Definition 8.1.3. *Let $T(n)$ be the space of all rooted binary trees on n leaves. The* diameter *Δ of $T(n)$ with respect to a distance metric D on $T(n)$ is defined as*

$$\Delta(T(n), D) = \max\{D(T_1, T_2) | T_1, T_2 \in T(n)\}$$

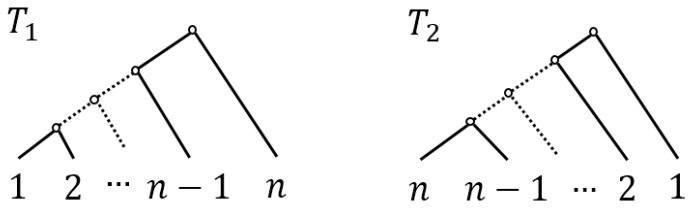

Figure 8.1: An example for two trees T_1 and T_2 on n leaves such that $RF(T_1, T_2) = n - 2$.

Theorem 8.1.1.

$$\Delta(T(n), RF) = n - 2$$

$$\Delta(T(n), CM) = \Theta(n^2)$$

Proof. Consider two trees T_1 and T_2 as shown in Fig. 8.1. The leaves in T_1 are ordered as $(1, 2, \cdots, n)$ and the leaves in T_2 are ordered as $(n, n-1, \cdots, 1)$. For the RF distance, $\mathcal{H}(T_1) = \big\{\{1,2\}, \{1,2,3\}, \cdots, \{1,2,\cdots,n-1,n\}\big\}$ and $\mathcal{H}(T_2) = \big\{\{n, n-1\}, \{n, n-1, n-2\}, \cdots, \{n, n-1, \cdots, 2, 1\}\big\}$. Hence, $RF(T_1, T_2) = \frac{1}{2}\big(|\mathcal{H}(T_1)| + |\mathcal{H}(T_2)|\big) = n - 2$.

8.1.2 Gradients to the tree edit operations

For the CM distance, the cluster of the root nodes are identical to $\{1, \cdots, n\}$, thus the weight between the root nodes is always 0. Let the node of the cluster $\{n, n-1, \cdots, \frac{3}{4}n\}$ in T_2 be v_l, then $W(u, v_l) > \frac{n}{4}$ for all $u \in V(T_1) \setminus \{r(T_1)\}$. Similarly, let the node of the cluster $\{n, n-1, \cdots, \frac{n}{4}\}$ in T_2 be v_u, then $W(u, v_u) \geq \frac{n}{4}$ for all $u \in V(T_1) \setminus \{r(T_1)\}$. There are $\frac{n}{2}$ nodes in $V(T_2)$ whose weight are greater than $\frac{n}{4}$ with any non-root nodes in $V(T_1)$ between u_l and v_u, therefore any matching in $B(T_1, T_2)$ have a weight at least $\frac{n}{4} \times \frac{n}{2} = \Omega(n^2)$. For the upper bound, consider two arbitrary trees T_1 and T_2 on n leaves. For $u \in V(T_1) \setminus \{r(T_1)\}$, in $B(T_1, T_2)$, $W(u, v) \leq n - 3$ for all $v \in V(T_2) \setminus \{r(T_2)\}$. Hence a matching in $B(T_1, T_2)$ can have a weight at most $\mathcal{O}(n)$. Therefore, $\Delta(T(n), CM) = \Theta(n^2)$. □

Let T_1 be a rooted binary tree and $\phi(T_1)$ be the set of trees derived by applying operation ϕ to a tree T_1, where ϕ be one of rooted nearest neighbor interchange (rNNI), rooted subtree prune and regraft (rSPR), or rooted tree bisection and regraft (rTBR).

- rNNI: Let $T_2 \in rNNI(T_1)$. An internal vertex u of a rooted binary tree T_1 has two incident edges that connects its children l and r. A rooted binary tree T_2 is obtained from T_1 by deleting $e = \{u, l\}$ (or $e' = \{u, r\}$), adding the edge between l (or r) and the vertex subdivides the edge that is incident with $Pa_{T_1}(u)$ and u's sibling, and then suppressing any degree-two vertices Moore et al. (1973); Robinson (1971).

- rSPR: Let $T_2 \in rSPR(T_1)$. $e = \{u, v\}$ and $u \leq_{T_1} v$. A rooted binary tree T_2 is obtained from T_1 by deleting e, adding the edge between u and the vertex that subdivides the edge of $T_1 \setminus e$, and then suppressing any degree-two vertices Allen and Steel (2001); Bordewich and Semple (2005); Swofford and Olsen (1990).

- rTBR: Let $T_2 \in rTBR(T_1)$. Analogous to rSPR, a rooted binary tree T_2 is obtained from T_1 by deleting e, adding an edge between vertices such that each of the vertices subdivides the edge of one and the other component of $T_1 \setminus e$, and then suppressing any degree-two vertices Allen and Steel (2001); Chen et al. (2006); Swofford and Olsen (1990).

Definition 8.1.4. *The* gradient *of a tree edit operation ϕ with respect to a distance metric D on $T(n)$ is defined as*

$$\mathcal{G}(T(n), D, \phi) = \max\{D(T_1, T_2) | T_1, T_2 \in T(n) \wedge T_2 \in \phi(T_1)\}$$

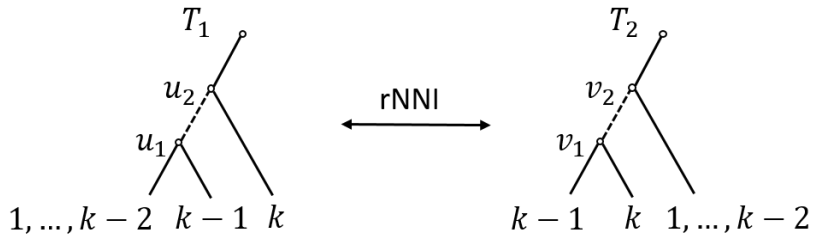

Figure 8.2: An example of rNNI operation. T_1 and T_2 are rooted binary trees, and $T_1 \in rNNI(T_2)$ and $T_2 \in rNNI(T_1)$. $\mathcal{C}_{T_1}(u_1) = \{1, \cdots, k-2, k-1\}$, $\mathcal{C}_{T_1}(u_2) = \{1, \cdots, k-2, k-1, k\}$, $\mathcal{C}_{T_2}(v_1) = \{k-1, k\}$, and $\mathcal{C}_{T_2}(v_2) = \{1, \cdots, k-2\}$ where $3 \leq k \leq n$.

Theorem 8.1.2.

$$\mathcal{G}(T(n), RF, rNNI) = 1$$

$$\mathcal{G}(T(n), CM, rNNI) = \Theta(n)$$

Proof. Consider two trees T_1 and T_2 as shown in Fig. 8.2. Suppose that T_1 in Fig. 8.2 is a caterpillar tree, $\mathcal{C}_{T_1}(u_1) = \{1, \cdots, k-2, k-1\}$, $\mathcal{C}_{T_1}(u_2) = \{1, \cdots, k-2, k-1, k\}$, $\mathcal{C}_{T_2}(v_1) = \{k-1, k\}$, and $\mathcal{C}_{T_2}(v_2) = \{1, \cdots, k-2, k-1, k\}$ where $3 \leq k \leq n$. The rNNI operation replaces the cluster $\mathcal{C}_{T_1}(u_1) = \{1, \cdots, k-2, k-1\}$ in $\mathcal{H}(T_1)$ with the cluster $\mathcal{C}_{T_2}(v_1) = \{k-1, k\}$ in $\mathcal{H}(T_2)$. Hence, $RF(T_1, T_2) = \mathcal{G}(T(n), RF, NNI) = 1$.

For the CM distance, in $B(T_1, T_2)$, $W(u_2, v_2) = 0$ because $\mathcal{C}_{T_1}(u_2) = \mathcal{C}_{T_2}(v_2) = \{1, \cdots, k-2, k-1, k\}$. The edge weight between u_1 and v_1 in $B(T_1, T_2)$, $W(u_1, v_1)$ is $k-1$ because $|\mathcal{C}_{T_1}(u_1) \ominus \mathcal{C}_{T_2}(v_1)| = |\{1, \cdots, k-2, k-1\} \ominus \{k-1, k\}| = |\{1, \cdots, k-2\} \cup \{k\}| = k-1$. Therefore, $\mathcal{G}(T(n), CM, rNNI) = \Theta(n)$ because $3 \leq k \leq n$. □

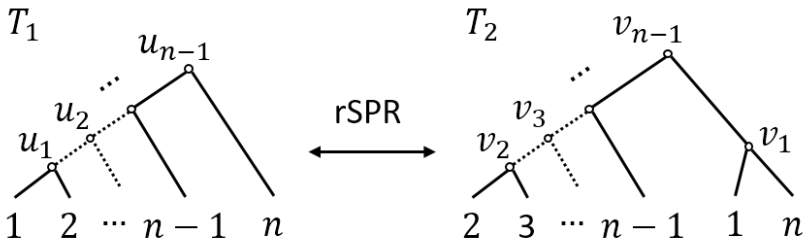

Figure 8.3: An example of rSPR operation such that $RF(T_1, T_2) = n-2$. T_1 and T_2 are rooted binary trees, and $T_1 \in rSPR(T_2)$ and $T_2 \in rSPR(T_1)$.

Theorem 8.1.3.

$$\mathcal{G}(T(n), RF, rSPR) = n - 2$$

$$\mathcal{G}(T(n), CM, rSPR) = \Theta(n^2)$$

Proof. Consider two trees T_1 and T_2 as shown in Fig. 8.3. The bound for the RF distance is derived by prune one leaf (1) at one end of T_1 and regraft it to the other end (n) of the tree. Hence, $\mathcal{G}(T(n), RF, rSPR) = n - 2$.

For the CM distance, consider two trees T_1 and T_2 as shown in Fig. 8.4. By the rSPR operation from T_1 to T_2, the edge $\{u_k, u_{k+1}\}$ is deleted, and the subtree $T_1(u_k)$ is grafted between u_k and u_{k+1} where $1 < l < k < n$. Note that $\mathcal{C}_{T_2}(v_m) = \{l+1, \cdots, m\}$ for $v_{l+1} <_{T_2} v_m \leq_{T_2} v_k$. Suppose that $\frac{n}{4} \leq l$ and $\frac{3}{4}n \leq k$, then in $B(T_1, T_2)$,

$$W(u, v_m) = \begin{cases} m & u \leq_{T_1} u_l \\ m-1 & u = u_{l+1} \\ l+\delta & \text{otherwise } (\delta > 0) \end{cases}$$

$W(u, v_m) > \frac{n}{4}$ for $v_{l+1} <_{T_2} v_m \leq_{T_2} v_k$ because $m > l+1$ and there are at least $\frac{n}{2}$ such a node v_m. Hence, any matching in $B(T_1, T_2)$ have a weight at least $\frac{n}{4} \times \frac{n}{2} = \Omega(n^2)$. The upper bound is trivial by Theorem 8.1.1. □

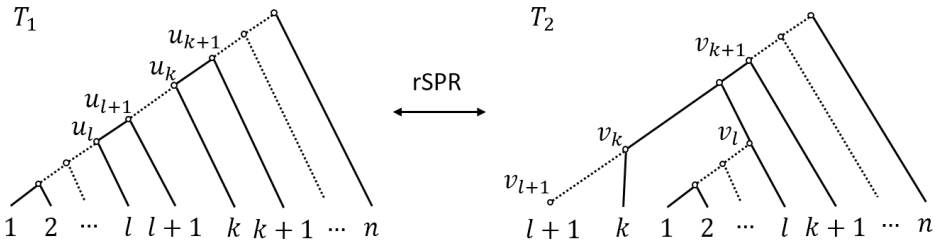

Figure 8.4: An example of rSPR operation such that $CM(T_1, T_2) = \Theta(n^2)$. T_1 and T_2 are rooted binary trees, and $T_1 \in rSPR(T_2)$ and $T_2 \in rSPR(T_1)$.

Theorem 8.1.4.

$$\mathcal{G}(T(n), RS, rTBR) = n - 2$$

$$\mathcal{G}(T(n), RSM, rTBR) = \Theta(n^2)$$

Proof. Since rSPR is a special case of rTBR, the results follow from Theorem 8.1.3. □

8.2 Experiments

We demonstrate the characteristics of the RF distance and the CM distance with the simulated data sets. First, we show the distributions of the RF distance and the CM distance

between a pair of randomly generated binary trees based on the two different models. Second, we compare how the RF distance and the CM distance are correlated with the number of consecutive tree edit operations that are rNNI, rSPR, and rTBR edit operations.

All experiments were performed on a workstation with an Intel® Xeon® CPU E7-8837 @2.66GHz with 128GB RAM.

8.2.1 Distribution of the tree distance metrics

We demonstrate the distributions of the RF distance and the CM distance between a pair of random binary trees. To generate random binary trees, we use the Yule-Harding Harding (1971) model and the birth-death process Arvestad et al. (2004) model.

8.2.1.1 Yule-Harding Model

Data set. We produced rooted profiles P_n and Q_n where $n \in \{100, 1000\}$. The profile P_n (and also Q_n) consists of $100,000$ random binary trees, and each random tree was generated by the following procedure: i) we started from a list of n single-node trees representing leaves. ii) we removed two randomly chosen trees from the list and made their roots as the children of the root of a new tree. iii) we added the new tree to the list. iv) we repeated this process until the list contains only one tree Górecki and Eulenstein (2015). It can be shown that such a process is equivalent to the classical Yule-Harding model for rooted tree shapes Betkier et al. (2015).

Experiment Setting. For the profiles $P_n = \{p_1, \cdots, p_{100000}\}$ and $Q_n = \{q_1, \cdots, q_{100000}\}$, we computed the RF distance and the CM distance between p_i and q_j for all $i = j$ and $i, j \in \{1, \cdots, 100000\}$.

Table 8.1: Descriptive statistics of the RF distance and the CM distance between a pair of randomly generated binary trees (Yule-Harding model) on 100 and $1,000$ leaves.

Distance	Leaves	Mean	SD	Min	Max	Range
RF	100	97.77	0.48	93	98	5
CM	100	891.6	38.28	760	1123	363
RF	1000	997.78	0.47	994	998	4
CM	1000	17659.27	423.3	16253	20031	3778

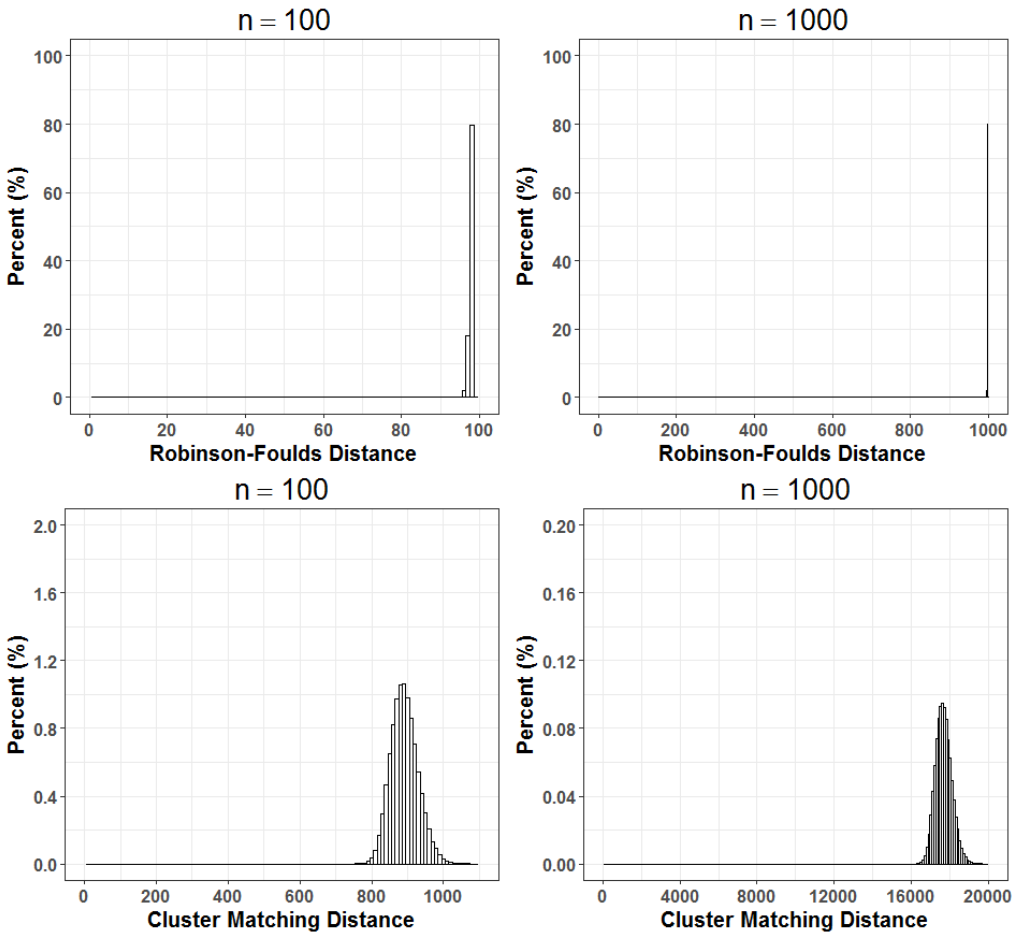

Figure 8.5: Distribution of the RF distance and the CM distance between a pair of randomly generated binary trees (Yule-Harding model) on 100 and 1,000 leaves.

Results and Discussion. Tab. 8.1 summarizes the descriptive statistics and Fig. 8.5 shows the distributions of the RF distance and the CM distance between a pair of randomly generated trees. The distribution of the RF distance is extremely skewed left, and hence their range is very narrow and also their minimum value and mean value are very close to their theoretical maximum value. In addition, the standard deviation and the range of the RF distance are similar between 100 and 1,000 leaves. It tells us that they are not proportional to the number of leaves. On the other hand, the CM distances are more broadly distributed in the form of a bell-shape and shows a wider ranges for both 100 and 1,000 leaves.

8.2.1.2 Birth-Death Process Model

Data set. Similar to the Yule-Harding model, we produced rooted profiles P_n and Q_n where $n \in \{100, 1000\}$. The profile P_n (and also Q_n) consists of $100,000$ random binary trees, and we used the software DendroPy version 3.10 Sukumaran and Holder (2010) for generating a birth-deadth process simulated tree (birth rate=0.1, death rate=0).

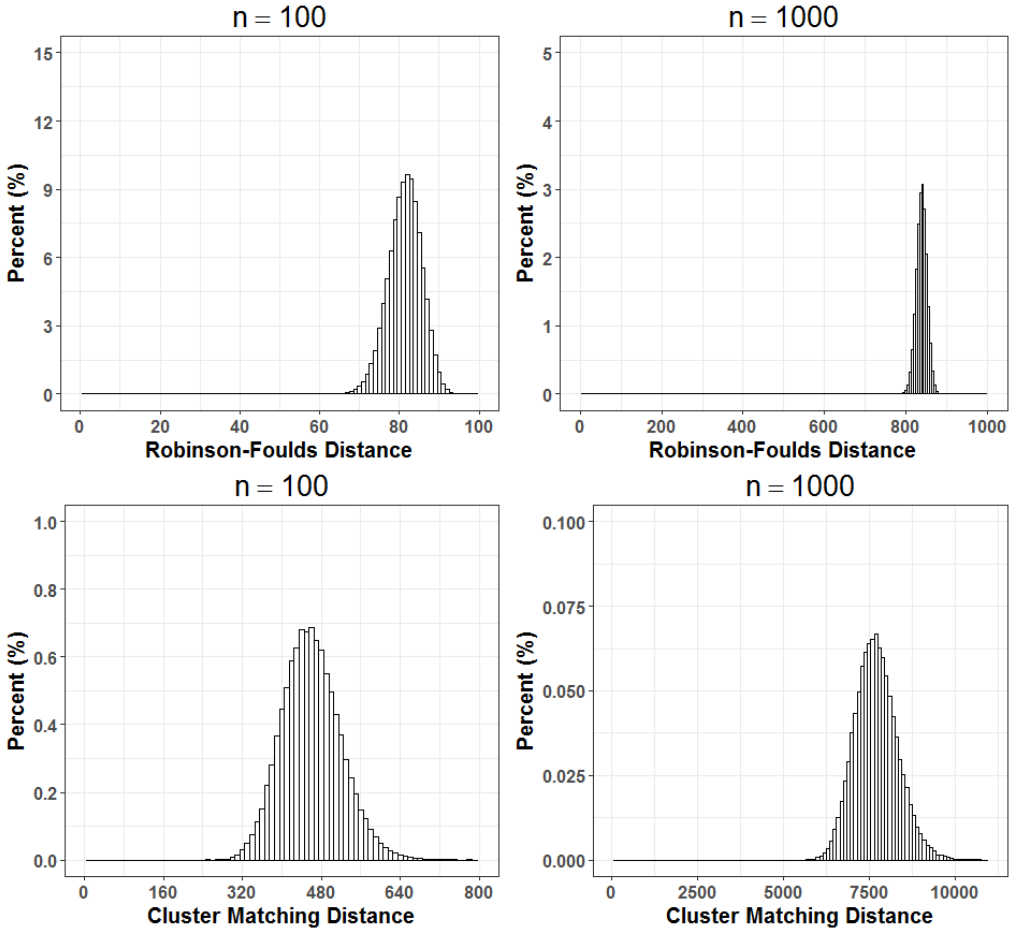

Figure 8.6: Distribution of the RF distance and the CM distance between a pair of randomly generated binary trees (birth-death process model) on 100 and $1,000$ leaves.

Experiment Setting. For the profiles $P_n = \{p_1, \cdots, p_{100000}\}$ and $Q_n = \{q_1, \cdots, q_{100000}\}$, we computed the RF distance and the CM distance between p_i and q_j for all $i = j$ and $i, j \in \{1, \cdots, 100000\}$.

Results and Discussion. Tab. 8.2 summarizes the descriptive statistics and Fig. 8.6 shows the distributions of the RF distance and the CM distance between a pair of randomly generated

Table 8.2: Descriptive statistics of the RF distance and the CM distance between a pair of randomly generated binary trees (birth-death process model) on 100 and 1,000 leaves.

Distance	Leaves	Mean	SD	Min	Max	Range
RF	100	81.37	4.16	61	96	35
CM	100	460.52	59.41	249	851	602
RF	1000	837.8	12.89	788	883	95
CM	1000	7705.55	623.8	5257	11327	6070

trees. Unlike the Yule-Harding model, the distributions of the RF distance and the CM distance are both in the form of a bell-shape. However, the distribution of the CM distance shows a wider ranges than the one of the RF distance for both 100 and 1,000 leaves. To show the relative standard deviation, we computed the coefficient of variation $c_v = \frac{\sigma}{\mu}$ where σ is the standard deviation and μ is the mean value. It follows that c_v of the CM distance is greater than the c_v of RF distance for both 100 and 1,000 leaves.

Table 8.3: Coefficient of variation of the RF distance and the CM distance between a pair of randomly generated binary trees (birth-death process model) on 100 and 1,000 leaves.

Distance	Leaves	Coefficient of variation, $c_v = \frac{\sigma}{\mu}$
RF	100	$\frac{4.16}{81.37} = 0.051$
CM	100	$\frac{59.41}{460.52} = 0.129$
RF	1000	$\frac{12.89}{837.8} = 0.015$
CM	1000	$\frac{623.8}{7705.55} = 0.080$

8.2.2 Tree distance metrics under tree editing operations

We demonstrate how the RF distance and the CM distance correlate with the number of consecutive rNNI, rSPR, and rTBR edit operations. From the result of the section 8.2.1, the RF distance is expected to be saturated faster than the CM distance by repeating the number of tree edit operations.

Data set. We created a rooted profile P consisting of 1,000 random binary trees on 500 leaves, and each tree in P was simulated by Yule-Harding model that was explained in the section 8.2.1.1.

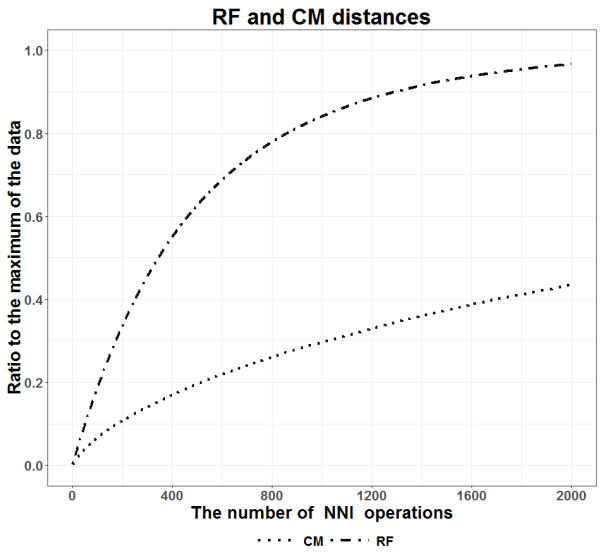

Figure 8.7: The average RF distance and CM distance of $1,000$ trees on 500 leaves as a function of the number of consecutive rNNI operations.

Experiment Setting. We repeated the tree edit operations and measured the pairwise distances by the following procedure:

1. We produced the profile Q_{nni}, Q_{spr}, and Q_{tbr} for each rNNI, rSPR, and rTBR tree edit operation. (Let $d \in \{nni, spr, tbr\}$.)

2. The initial $Q_d = \{q_1, \cdots, q_{1000}\}$ is identical to $P = \{p_1, \cdots, p_{1000}\}$.

3. We applied the randomized tree edit operation d to all trees in Q_d, and the edited trees are stored in Q_d.

4. We measured the RF distance and the CM distance between p_i and q_j for all $i = j$ and $i \in \{1, \cdots, 1000\}$.

5. We averaged the measured RF distances and CM distances.

6. We repeated the process from the step 2. to the step 5. for $2,000$ times for the rNNI operation and 500 times for the other operations.

For a given tree T, we applied the randomized tree edit operations by the following method:

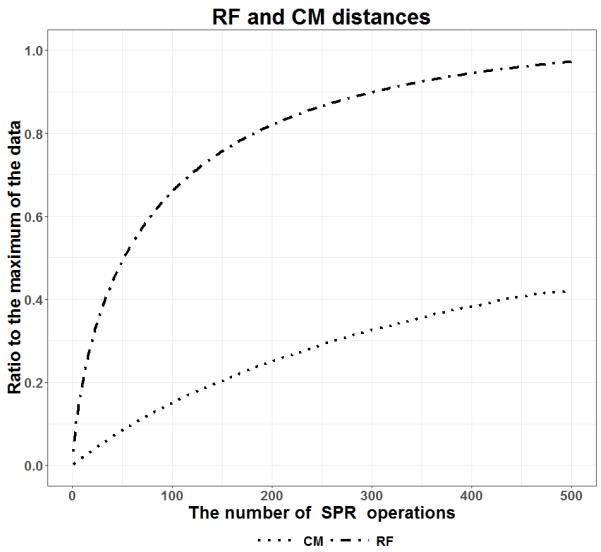

Figure 8.8: The average RF distance and CM distance of $1,000$ trees on 500 leaves as a function of the number of consecutive rSPR operations.

- rNNI: Choose an internal edge $e = \{u, v\} \in E(T)$ $(u \leq_{T_1} v)$ arbitrary and delete e. Add the edge between u and the vertex subdivides the edge that is incident with $Pa_T(v)$ and v's sibling.

- rSPR: Choose an edge $e = \{u, v\} \in E(T)$ $(u \leq_{T_1} v)$ arbitrary and delete e. Add the edge between u and the vertex that subdivides the randomly chosen edge of $T \setminus e$.

- rTBR: Choose an edge e arbitrary and delete e. Add an edge between vertices such that each of the vertices subdivides the randomly chosen edge of one and the other component of $T \setminus e$.

Results and Discussion. Fig. 8.7 shows the average RF distance and CM distance between the initial tree and rNNI operation applied trees. The gradient of the RF distance curve is very steep between $0 \sim 1,200$ operations and the inclination of the curve is gradual after $1,600$ operations. However, the CM distance after $1,600$ operations is still in an increasing trend. Fig. 8.8 shows the average RF distance and CM distance between the initial tree and rSPR operation applied trees. While the gradient of the RF distance curve is gradual after 400 operations, but the gradient of the CM distance is in an increasing trend. Fig. 8.9 shows the average RF distance and CM distance between the initial tree and rTBR operation applied trees. Unlike the rSPR

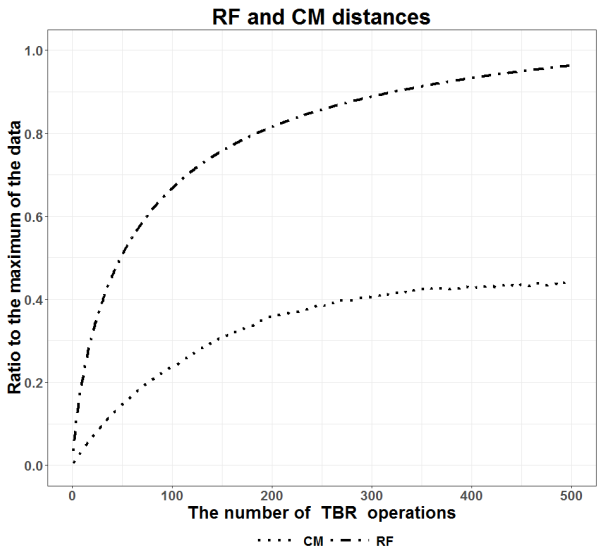

Figure 8.9: The average RF distance and CM distance of $1,000$ trees on 500 leaves as a function of the number of consecutive rTBR operations.

operation, the gradients of the RF distance and the CM distance curve are both steep between $0 \sim 200$ operations, and they are gradual after 350 operations.

8.3 Conclusion

While the Robinson-Foulds distance measure has been used widely to compare phylogenetics trees in computational biology, the distance is poorly distributed, shows insufficient discrimination, and is too sensitive to tree edit operations. To overcome these shortcomings, we introduced a new tree metric.

Our new distance measure is a metric on the space of rooted trees, and it can be computed in polynomial time in contrast to edit distances under tree edit operations. Throughout the simulated experiments, we demonstrated that this new metric is distributed much more broadly and less biased, and also the metric is less sensitive to a tree edit operation than the Robinson-Fould metric.

CHAPTER 9. HIGHLY BI-CONNECTED SUBGRAPH

A key idea of *graph clustering* is to identify densely connected subgraphs as clusters that have many interactions within themselves and few interactions outside of themselves in the graph [Hüffner et al. (2014)]. A study by Pržulj et al. (2004) determines clusters, which could indicate protein functions, by using the HCS algorithm in PPI networks. A highly connected subgraph is defined as a subgraph with n vertices such that more than $\frac{n}{2}$ of its edges must be removed in order to disconnect the subgraph. The concept of a highly connected graph is very similar to that of a *quasi-clique* (i.e., a graph where every vertex has a degree at least $\frac{n-1}{2}$ [Hüffner et al. (2014)]). Hartuv and Shamir [Hartuv and Shamir (2000)] proved that the HCS algorithm, which is based on the $\frac{n}{2}$ connectivity requirement, produces clusters with good homogeneity and separation properties [Pržulj et al. (2004)]. However, the concept of highly connected subgraphs is not applicable to bipartite graphs, since they do not contain such subgraphs.

Bipartite graphs are frequently used to represent biological networks. Bicliques in bipartite PPI networks play an important role in identifying functional protein groups [Wang (2013)]. A study by Andreopoulos et al. (2007) identifies locally significant proteins, that mediate the function of proteins, by exploring bicliques in PPI networks. However, a biclique is too stringent for identifying the functional groups [Geva and Sharan (2011)]. A *quasi-biclique* allows a specified number of missing edges in a biclique [Wang (2013)]. Bu et al. (2003) show that quasi-bicliques consist of relevant protein functions and also propose a method to predict protein functions based on the classification of known proteins within the quasi-bicliques. Although a quasi-biclique is less stringent, it allows for the inclusion of proteins that interact with few other proteins in the quasi-biclique [Chang et al. (2012)].

We propose the HBC problem to identify highly bi-connected subgraphs in bipartite networks. Essential for this work is Theorem 9.1.3 that describes a highly bi-connected graph $G = (U, V, E)$ equivalently as a graph where the minimum degree is larger than $\frac{1}{2}$ of the minimum cardinality of the vertex sets U and V. Using this theorem we show the NP-hardness of the HBC problem by a polynomial time reduction from the exact 3-sets cover problem. Further, Theorem 9.1.3 is also used to describe an initial IQP formulation for the HBC problem that contains quadratic constraints. This initial IP is then transformed into an ILP by replacing the quadratic constraints with linear ones using simplified variables by adapting implication rules. Our heuristic follows a seed based approach, where seeds are expanded to highly bi-connected subgraphs with the maximum number of vertices, and resulting subgraphs with the largest number of vertices are returned. Finally, we demonstrate the performance of our heuristic algorithm by comparing its results with exact ILP solutions for small-scale instances of the HBC problem, and through an experimental study that annotates protein function by analyzing a bipartite protein-function network built from data provided by the *UniProt-GOA* human database [Consortium et al. (2014)].

9.1 Highly Bi-Connected Subgraph Problem

Definition 9.1.1 (Highly Connected Subgraph)**.** *A graph G is called* highly connected *if $\lambda(G) > \frac{|V|}{2}$. An induced subgraph $G[V']$ (where $V' \subseteq V$) that is highly connected is called a* highly connected subgraph *(HCS).*

Theorem 9.1.1. *There is no highly connected bipartite graph.*

Proof. Let $G = (X \cup Y, E)$ be a bipartite graph and highly connected. Then $\frac{|X|+|Y|}{2} \geq \min(|X|, |Y|) \geq \delta(G) \geq \lambda(G) > \frac{|X|+|Y|}{2}$, which is a contradiction. □

Definition 9.1.2 (Highly Bi-Connected Subgraph)**.** *Let $G = (X \cup Y, E)$ be a bipartite graph. A bipartite graph G is called* highly bi-connected *if $\lambda(G) > \frac{1}{2}\min(|X|, |Y|)$. An induced bipartite subgraph $G[X' \cup Y'], (X' \subseteq Y, Y' \subseteq Y)$ is a* highly bi-connected subgraph *(HBCS) if $G[X' \cup Y']$ is highly bi-connected.*

Theorem 9.1.2. *Let $G = (X \cup Y, E)$ be a bipartite graph. If $\lambda(G) > \frac{1}{2}\min(|X|, |Y|)$ and $|X| \geq |Y|$, then $dst(u, v) = 2$ for all distinct $u, v \in X$.*

Proof. $d(u), d(v) \geq \delta(G) \geq \lambda(G) > \frac{1}{2}\min(|X|, |Y|) = \frac{|Y|}{2}$. There exists at least one vertex $z \in Y$ such that $P = uzv$ is a path because $|N_G(u) \cap N_G(v)| \geq 1$. □

Corollary 9.1.1. *(Dankelmann and Volkmann, 1995, page 273) Let $G = (X \cup Y, E)$ be a bipartite graph. If $dst(u, v) = 2$ for all distinct $u, v \in X$, then $\lambda(G) = \delta(G)$.*

Theorem 9.1.3. *Let $G = (X \cup Y, E)$ be a bipartite graph. If $\delta(G) > \frac{1}{2}\min(|X|, |Y|)$, then $\delta(G) = \lambda(G)$.*

Proof. Supposed that $|X| \geq |Y|$, and $u, v \in X$. $d(u), d(v) \geq \delta(G) > \frac{|Y|}{2}$. There exists at least one vertex $z \in Y$ such that $P = uzv$ is a path because $|N_G(u) \cap N_G(v)| \geq 1$. Hence, $d(u, v) = 2$ and $\delta(G) = \lambda(G)$ by Theorem 9.1.1. □

Corollary 9.1.2. *A bipartite graph G is highly bi-connected if $\delta(G) > \frac{1}{2}\min(|X|, |Y|)$, $\delta(G) \geq \lceil \frac{1}{2}\min(|X|, |Y|) \rceil$, or $2\delta(G) \geq \min(|X|, |Y|) + 1$.*

Problem 9.1.1 (HBCS Problem)**.**

Instance: A undirected bipartite graph $G = (X \cup Y, E)$ and positive integer k.

Question: Is there a vertex set $X' \cup Y'$ such that $|X'| + |Y'| = k$ and $G' = G[X' \cup Y']$ is highly bi-connected?

Theorem 9.1.4. *HBCS problem is NP-hard.*

Proof. The *exact cover by 3-sets (X3C) problem* is known to be NP-hard [Karp (1972)]. The reduction algorithm takes an instance $\langle S, T \rangle$ of the X3C problem where S is a finite set of $3k$ elements and T is a collection of l triples (three-element subsets of S). Without loss of generality, we assume that $k < l < 2k$.

Step 1. A bipartite graph $G_A = (X_A \cup Y_A, E_A)$ is created by linking an element $s_i \in S$ with $x_i \in X_A$ and $t_j \in T$ with $y_j \in Y_A$. An edge $(x_i, y_j) \in E_A$ is established iff $s_i \notin t_j$. Note that $|X_A| = 3k$, $|Y_A| = l$, and $d(y_j, X_A) = 3k - 3$.

Step 2. A bipartite graph $G_B = (X_B \cup Y_B, E_B)$ is constructed as $X_B = X_a \cup X_b \cup X_c$ where $|X_a| = |X_b| = 3k$, $|X_c| = 6$ and $Y_B = Y_a \cup Y_b \cup Y_c$ where $|Y_a| = |Y_b| = l$, $|Y_c| = 3$. We set edges to make that $G[X_a \cup Y_a]$ and $G[X_b \cup Y_b]$ are equivalent with G_A. Each vertex in X_c is adjacent to all vertices in $Y_a \cup Y_b$, and similarly, each vertex in Y_c is adjacent to all vertices in $X_a \cup X_b$. Note that $|X_B| = 6k + 6$ and $|Y_B| = 2l + 3$.

Step 3. A bipartite graph $G = (X \cup Y, E)$ is built as $X = X_1 \cup X_2 \cup \cdots \cup X_{3k}$ and $Y = Y_B$ where $|X_1| = |X_2| = \cdots = |X_{3k}| = 6k + 6$. We connect edges to achieve that $G[X_i \cup Y]$ $(1 \leq i \leq 3k)$ is identical to G_B. Note that $|X| = 3k(6k + 6)$ and $|Y| = 2l + 3$.

These steps can be done in polynomial time. The output of the reduction algorithm is an instance $\langle G, 18k^2 + 20k + 3\rangle$ of the HBCS problem. Suppose that $\langle S, T\rangle$ has a perfect cover $T' \subseteq T$ where $|T'| = k$. We claim that G has a HBCS $G' = G[X' \cup Y']$ such that $X' = X$, $Y' = Y'_a \cup Y'_b \cup Y_c$ where Y'_a and Y'_b contain k vertices associated with k triples in T'. In the induced bipartite G', $d(x, Y') \geq (k - 1) + 3 = k + 2 > \frac{1}{2}\min(|X'|, |Y'|) = \frac{1}{2}|Y'| = \frac{1}{2}(2k + 3) = k + \frac{3}{2}$ $(x \in X')$ and $d(y, X') \geq 3k(3k + 3) > k + \frac{3}{2}$ $(y \in Y')$. Thus, $\delta(G') > \frac{1}{2}\min(|X'|, |Y'|)$ and $|X'| + |Y'| = 3k(6k + 6) + (2k + 3) = 18k^2 + 20k + 3$.

Conversely, Suppose that G has a HBCS $G' = G[X' \cup Y']$ where $|X'| + |Y'| = 18k^2 + 20k + 3$. We claim that $\langle S, T\rangle$ has a perfect cover $T' \subseteq T$ such that $X' = X$, $Y' = Y'_a \cup Y'_b \cup Y_c$ where Y'_a and Y'_b contain k vertices associated with k triples in T'.

First, we prove that $|Y'_a| + |Y'_b| = 2k$. If $|Y'_a| + |Y'_b| < 2k$, then $|X| > 3k(6k + 6)$. This is a contradiction. If $|Y'_a| + |Y'_b| > 2k$, then there is a positive integer p such that $2k + (2p - 1) \leq |Y'_a| + |Y'_b| \leq 2k + 2p$. We assume that $|Y'_a| \leq |Y'_b|$, hence $|Y'_a| \leq k + p$. Now, we consider $X'_{i,a} = X_{i,a} \cap X'$ and prove $X'_{i,a} \subsetneq X_{i,a}$. Suppose that $X'_{i,a} = X_{i,a}$. For a vertex $x \in X'_{i,a}$, $d(x, Y') > \frac{1}{2}|Y'| \geq \frac{1}{2}(2k + 2p - 1 + 3) = k + p + 1$. By the construction, $d(x, Y'_a) > k + p - 2$ because $d(x, Y'_b) = 0$ and $d(x, Y'_c) = 3$. The inequality can be written $d(x, Y'_a) \geq k + p - 1$ since a degree is an integer. The number of edges between all $X'_{i,a}$ and Y'_a is at least $3k(k + p - 1) = 3k^2 + 3pk - 3k$. For a vertex $y \in Y_a$, $d(y, X'_{i,a}) = 3k - 3$. The number of edges between all $X'_{i,a}$ and Y'_a is at most $(k + p)(3k - 3) = 3k^2 + 3pk - 3k - 3p$. There is no integer e such that $3k^2 + 3pk - 3k \leq e \leq 3k^2 + 3pk - 3k - 3p$ for a positive integer p. Thus, $X'_{i,a} \subset X_{i,a}$ and there are at least $3k$ vertices in X not in X' since there is at least one vertex in $X_{i,a}$ not in $X'_{i,a}$.

$|X'| \leq 3k(6k+6) - 3k = 18k^2 + 15k$, $|Y'| < 4k+3$ ($\because l < 2k$), and $|X'| + |Y'| < 18k^2 + 19k + 3$. This is a contradiction, hence $|Y'_a| + |Y'_b| = 2k$.

Second, we prove that $|Y'_a| = |Y'_b| = k$. We assume that $|Y'_a| < k$ and $X'_{i,a} = X_{i,a}$. For a vertex $x \in X'_{i,a}$, $d(x, Y') > \frac{1}{2}|Y'| = \frac{1}{2}(2k+3) = k + \frac{3}{2}$. Hence, $d(x, Y'_a) \geq k - 1$ ($\because d(x, Y'_a) > k + \frac{3}{2} - 3$) and the number of edges between all $X'_{i,a}$ and Y'_a is at least $3k(k-1) = 3k^2 - 3k$. For a vertex $y \in Y_a$, $d(y, X'_{i,a}) = 3k - 3$. The number of edges between all $X'_{i,a}$ and Y'_a is less than $k(3k-3) = 3k^2 - 3k$. This is a contradiction. Therefore, $X'_{i,a} \subset X_{i,a}$ and there are at least $3k$ vertices in X not in X'. $|X'| \leq 3k(6k+6) - 3k = 18k^2 + 15k$, $|Y'| = 2k + 3$, and $|X'| + |Y'| \leq 18k^2 + 17k + 3 < 18k^2 + 20k + 3$. Consequently, $|Y'_a| = |Y'_b| = k$.

Finally, we prove the original claim. For each vertex $x \in X'_{i,a}$, $d(x, Y'_a) \geq k - 1$ because $d(x, Y') > k + \frac{3}{2}$, $d(x, Y'_b) = 0$, and $d(x, Y'_c) = 3$. Suppose that there exists a vertex x such that $d(x, Y'_a) > k - 1$. The number of edges between all $X'_{i,a}$ and Y'_a is greater than $3k(k-1)$. For each vertex $y \in Y'_a$, $d(y, X'_{i,a}) = 3k - 3$. The number of edges between all $X'_{i,a}$ and Y'_a is $k(3k-3) = 3k(k-1)$. This is a contradiction, and hence $d(x, Y'_a) = k-1$ and $d(y, X'_{i,a}) = 3k-3$ ($\forall x, y \in X'_{i,a}, Y'_a$). This means that the corresponding subset $T'$is an exact cover of S. □

9.2 Integer Linear Programming

The first IQP formulation requires quadratic constraints, which are then replaced by linear constraints such that it can be solved by various optimization software packages [Chang et al. (2012)]. Furthermore, the second ILP formulation is improved by using the implication rule to simplify variables involved.

9.2.1 Quadratic programming for maximum HBCS

Let $G = (X \cup Y, E)$ be a bipartite graph. For each $x \in X$ ($y \in Y$), a binary variable v_x (v_y) is introduced. The variable v_x (v_y) is 1 if and only if the vertex v_x (v_y) is in X' (Y'). The

integer programing is formulated as follows.

$$\text{maximize} \qquad \sum_{x \in X} v_x + \sum_{y \in Y} v_y$$

$$\text{subject to} \qquad 2 \sum_{y \in Y} e_{xy} v_y v_x \geq v_x (W+1) \qquad \forall x \in X \tag{9.1}$$

$$2 \sum_{x \in X} e_{xy} v_x v_y \geq v_y (W+1) \qquad \forall y \in Y \tag{9.2}$$

$$\sum_{x \in X} v_x \geq \sum_{y \in Y} v_y \text{ or } \sum_{x \in X} v_x \leq \sum_{y \in Y} v_y \tag{9.3}$$

$$W = \sum_{y \in Y} v_y \text{ or } W = \sum_{x \in X} v_x \tag{9.4}$$

$$v_x \in \{0, 1\} \qquad \forall x \in X \cup Y$$

$$\text{where} \qquad e_{xy} = \begin{cases} 0 & xy \notin E \\ 1 & xy \in E \end{cases}$$

The quadratic terms in constraints are necessary because the constraints apply only to vertices in $X' \cup Y'$.

9.2.2 Linear programming for maximum HBCS

v_x (v_y) has two possible values such as 0 or 1. The constraint (9.1) is turned into $\sum_{y \in Y} e_{uv} v_y \geq W + 1$ in case of $v_x = 1$ and it becomes trivial when $v_x = 0$. Thus, the constraints (9.1) and (9.2) are reestablished as follow.

$$2 \sum_{y \in Y} e_{xy} v_y - W - 1 \geq (|X| + |Y|)(v_x - 1) \qquad \forall x \in X$$

$$2 \sum_{x \in X} e_{xy} v_x - W - 1 \geq (|X| + |Y|)(v_y - 1) \qquad \forall y \in Y$$

In order to obtain an optimal solution, we solve the ILP problem twice by setting the constraints (9.3) and (9.4) separately each time (i.e., $W = \sum_{x \in X} v_x$ if $\sum_{x \in X} v_x \leq \sum_{y \in Y} v_y$ or $W = \sum_{y \in Y} v_y$ if $\sum_{x \in X} v_x \geq \sum_{y \in Y} v_y$). In summary, this formulation uses variables and constraints linear to the size of input vertices. i.e., $\mathcal{O}(|X| + |Y|)$.

9.3 Heuristic Algorithms

For a given bipartite graph and a subset of a vertex partition of this graph, Algorithm 8 identifies a subgraph that satisfies the following four conditions: i) the subgraph is highly bi-connected; ii) one vertex partition of the subgraph is identical with the given subset; iii) the number of vertices in the other vertex partition is greater than or equal to the number of vertices in the given subset; and iv) the number of vertices in the subgraph is maximized. Let n be the number of vertices in the given bipartite graph. The time complexity of Algorithm 8 is $\mathcal{O}(n^2)$. This follows directly from $\mathcal{O}(n)$ executions of the for-loop (Steps 1-5), where the time complexity of executing the body of this loop is asymptotically bound by Step 2 requiring $\mathcal{O}(n)$ time.

Algorithm 8 MaxVertex-HBCS(G, Y')

Input: A bipartite graph $G = (X \cup Y, E)$ and a vertex set $Y' \subseteq Y$.
Output: A maximum vertex HBCS $G' = (X' \cup Y', E')$ such that $|X'| \geq |Y'|$.

1: **for** all $v \in N(Y')$ **do**
2: **if** $|N(v) \cap Y'| > \frac{1}{2}|Y'|$ **then**
3: $X' = X' \cup \{v\}$
4: **end if**
5: **end for**
6: **if** $|X'| \geq |Y'|$ AND $G[X' \cup Y']$ is HBCS **then**
7: return $G[X' \cup Y']$
8: **end if**

Algorithm 9 enumerates maximum vertex HBCSs for a given bipartite graph that uses a greedy approach to identify seed vertex sets. The while loop (Steps 3-11) identifies maximum vertex HBCS until no more maximum vertex HBCS can be found from the seed vertices. Algorithm 9 maintains the list of seed vertex sets to avoid repeating the process on the seed vertex sets that are already examined. Let n be the number of vertices in the given bipartite. The time complexity of Step 5 is $\mathcal{O}(n^2)$, and this step is repeated $\mathcal{O}(n^2)$ times through nested for and while loop. Hence, the overall time complexity is $\mathcal{O}(n^4)$.

Algorithm 9 GreedyEnum-MaxVertexHBCS(G)

Input: A bipartite graph $G = (X \cup Y, E)$.
Output: A set of maximum vertex HBCS $G' = (X' \cup Y', E')$ such that $|X'| \geq |Y'|$.

```
for u ∈ Y do
    Y' = {u}
    while Y' ≠ Y AND Q does not contain Y' do
        Q = Q ∪ {Y'}
        G' =MaxVertex-HBCS(G, Y')
        if G' ≠ NULL then
            OUTPUT G'
            Find a vertex v ∈ N_G(X') \ Y' that maximize d_G(v, X').
            Y' = Y' ∪ {v}
        end if
    end while
end for
```

9.4 Experiments

We analyze the performance of the heuristic algorithm by comparing its results with exact ILP solutions for small-scale instances of the HBC problem, and through an experimental study.

9.4.1 Comparative study

We compare heuristic estimates with the exact ILP results for $1,000$ random graphs as input. The random graphs were selected with equal probability from graphs with the following: an overall number of vertices ranging between 10 and 26 vertices and edge densities ranging between 0.6 and 0.8. The resulting differences between exact solutions and heuristic estimates are depicted in Fig. 9.1.

9.4.2 Empirical study

In this experimental study, we present the results of the protein function prediction by using our heuristic algorithm. The *Gene Ontology (GO)* [Ashburner et al. (2000)] is currently the dominant approach for machine-legible protein function annotations [Friedberg (2006)]. GO is a controlled vocabulary that describes three aspects of protein functions: molecular function, biological process, and cellular location. Each aspect is described by a directed acyclic graph of terms and relationships that captures functional information in a standardized fashion that

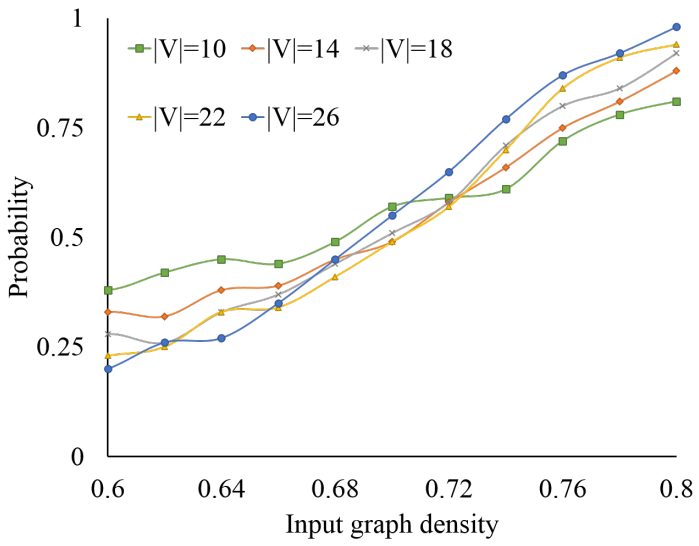

Figure 9.1: The performance of the maximum vertex heuristic algorithm is evaluated by comparing its results with exact ILP solutions for small-scale instances.

is both computationally amenable and interpretable by humans. We use the *Biological Process* classification scheme in this study. The main task of the experimental study is to predict sets of GO terms for the target proteins with confidence scores.

Target Proteins. We obtained annotated proteins from the January versions of the 2012, 2013, 2014, 2015, and 2016 UniProt-GOA human database [Consortium et al. (2014)]. Proteins are considered to be experimentally annotated if they are associated with GO terms having EXP, IDA, IPI, IMP, IGI, IEP, TAS, or IC evidence codes. The set of target proteins is selected by using the following scheme with two distinct time frames t_0 and t ($t_0 < t$)

$$\begin{aligned}\text{Targets}(t) = &\text{ Set of proteins at least one experimental annotation exist at } (t)\\ &\cap \text{ Set of proteins only non-experimental annotation exist at } (t_0)\end{aligned}$$

The predictive model is trained with non-experimental annotations of target proteins and experimental annotations of non-target proteins at $t_0 = 2012$. The performance of the model is evaluated by comparing the predicted annotations made by us for 2012 to existing experimental annotations in 2013-2016.

Experimental Design. Our predictive model uses the maximum vertex HBCS. For a given set of annotations between proteins and GO terms, we create a bipartite graph that has one

vertex partition representing the set of proteins, the other vertex partition representing the set of GO terms, and edges representing the set of annotations. After that, Algorithm 9 finds a list of maximum vertex HBCS from the created bipartite graph. Every pair of a protein and a GO term in each HBCS of the found list is considered as a predictive annotation. The confidence score of the predictive annotation, which indicates the strength of the prediction, is the maximum sequence identity between the target protein and any neighboring non-target proteins of the GO term in the found HBCS. Other sequence identity measures, such as 3D sequence structure, genomic context, or interaction based, will be evaluated in future research work.

Evaluation Metric. For a given target protein i and some decision threshold $t \in [0, 1]$, the precision and recall are calculated as

$$pr_i(t) = \frac{\sum_f I(f \in P_i(t) \wedge f \in T_i)}{\sum_f I(f \in P_i(t))}, \; rc_i(t) = \frac{\sum_f I(f \in P_i(t) \wedge f \in T_i)}{\sum_f I(f \in T_i)}$$

where f is a protein function in the biological process GO terms, T_i is a set of experimentally determined GO terms for protein i, and $P_i(t)$ is a set of predicted protein functions of i with score greater than or equal to t. f ranges over all protein functions and $I(\cdot)$ stands for the indicator function. For a fixed decision threshold t, a point in the precision-recall space is created by averaging precision and recall across targets. Precision and recall at threshold t is calculated as

$$pr(t) = \frac{1}{m(t)} \cdot \sum_{i=1}^{m(t)} pr_i(t), \; rc(t) = \frac{1}{n} \cdot \sum_{i=1}^{n} rc_i(t)$$

where $m(t)$ is the number of proteins on which at least one prediction is made on threshold t and n is the number of proteins in a target set. It should be noted that unlike [Radivojac et al. (2013)], we did not consider the GO DAG topology, but simply ran our assessment on GO terms as a "flat" vocabulary.

Results and Discussion. The quality of protein function prediction can be measured in different ways that reflect differing motivations for understanding protein functions. For this study, we show the precision-recall curves with all proteins having non-experimental annotations 2012 as the basis for predictions. We used the proteins that gained experimental annotations in 2013-2015 to test our method. The results are shown in Fig 9.2.

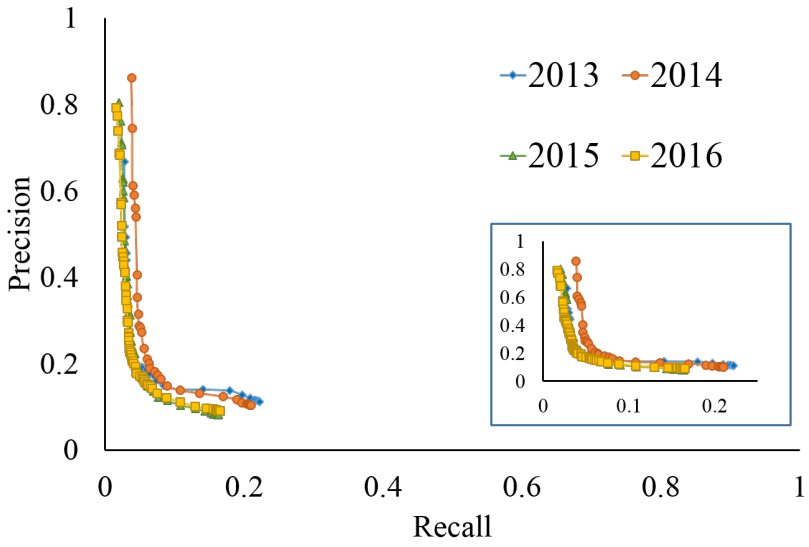

Figure 9.2: Precision-recall curves for our method. The model is trained with non-experimental annotations of target proteins and experimental annotations of non-target proteins at $t_0 = 2012$. The performance of the model is evaluated by comparing the predicted annotations made by us for 2012 to existing experimental annotations in $2013 \sim 2016$.

While our method has an overall low recall rate, it does have a high precision rate at low recall values. For some niche biological applications, such a method may be useful, as biomedical researchers may prefer generating protein function predictions with a high precision rate while trading off recall to minimize false positives for the results they do use. To estimate performance at different recall values, we used the $F_{max(\beta)}$ for the different years defined as

$$F_{max(\beta)} = \max_t \left\{ (1+\beta^2) \frac{pr(t) \cdot rc(t)}{\beta^2 \cdot pr(t) + rc(t)} \right\}$$

where values for β are 0.1, 0.2, 0.5 and 1.0. F_β is a weighted harmonic mean of the precision and recall. We find the maximal value for each year using different values of β as weight. The lower β, the more weight is given to precision over recall. The results are shown in Table 9.1.

9.5 Conclusion

Our proposed HBC approach sets a way for the functional annotation of proteins based on identifying highly bi-connected subgraphs in bipartite protein-function networks. While we show that the HBC problem is NP-hard, and we describe an ILP formulation and an effective heuristic. The comparative study displays accuracy of our heuristic by comparing its results

Table 9.1: Results of $F_{max(\beta)}$ analysis. See text for details on how $F_{max(beta)}$ is calculated. The lower the value of β, the more precision is weighted over recall. Our method performs best overall with $\beta = 0.1$.

	$F_{max(\beta)}$			
Year	0.1	0.2	0.5	1.0
2013	0.54	0.36	0.14	0.16
2014	0.71	0.47	0.17	0.14
2015	0.58	0.34	0.12	0.11
2016	0.54	0.31	0.12	0.12

with exact ILP solutions. Furthermore, the experimental study demonstrates the applicability of the heuristic for functionally annotating proteins. Future research will investigate other maximization objectives for identifying highly bi-connected subgraphs and partitioning problems of bipartite graphs based on highly bi-connected subgraphs.

CHAPTER 10. SOFTWARE IMPLEMENTATION

The software is developed to address the median tree problem that are the Gene Duplication, Robinson-Foulds, and Deep Coalescence. The software implemented the methods in the Chapter 4, 5, and 6.

- Software: http://genome.cs.iastate.edu/ComBio/software/genie.jar
- Operating system(s): Platform independent
- Requirements: : Java Runtime Environment version 8 or higher

The following figures are the example of gene trees and their estimated median trees in terms of the gene duplication, the Robinson-Foulds, and the deep coalescence scores.

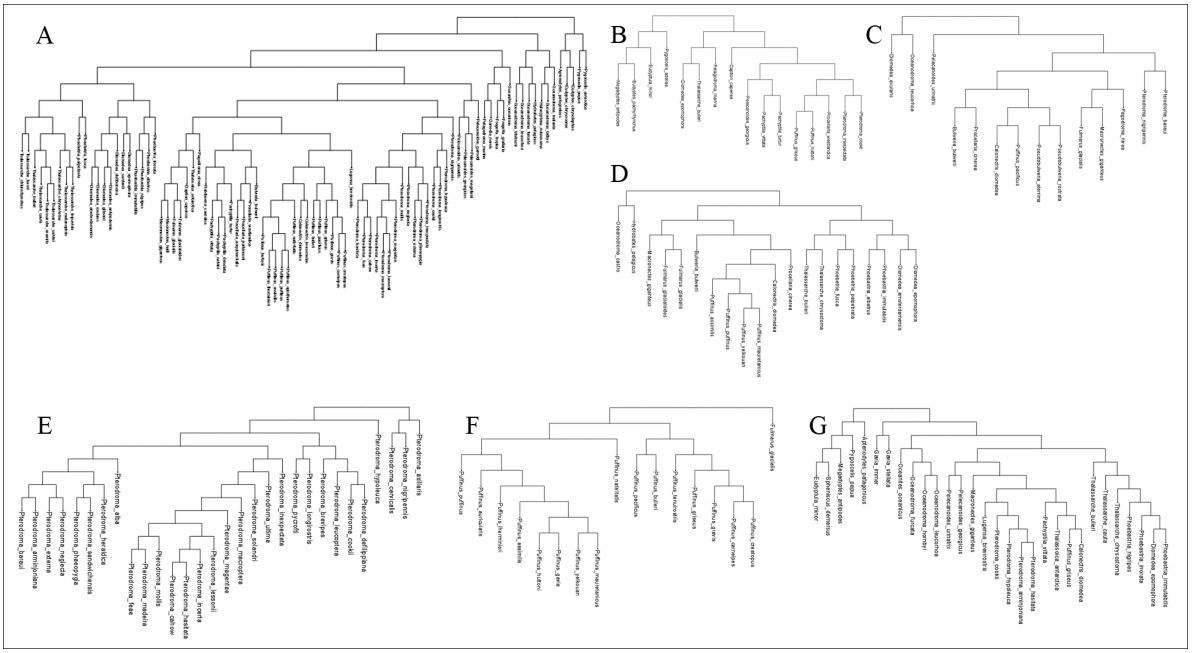

Figure 10.1: The gene trees of 121 Seabirds.
A: 90 Taxa, B: 16 Taxa, C: 14 Taxa, D: 20 Taxa, E: 30 Taxa, F: 17 Taxa, G: 30 Taxa.

Figure 10.2: The synthesized species trees computed by the Strict Consensus Approach.
A: The gene duplication median tree with parsimony score 22.
B: The deep coalescence median tree with parsimony score 28.
C: The Robinson-Foulds median tree with parsimony score 58.

CHAPTER 11. CONCLUSIONS

There is an increased interest in making NP-hard median tree problems available for large-scale species tree construction. We showed that the gene duplication satisfies the weak Pareto for clusters, which allows us to design an efficient parameterized algorithm for the gene duplication problem for input trees with the same taxon set. The parameter is the maximum degree of the strict consensus tree of the input trees.

While it has been demonstrated that the parametrized algorithm can successfully synthesize large-scale species trees, it is only able to handle restricted median tree problems (i.e., rooted binary gene trees sampled from identical species), which have largely limited applicability in practice. We introduced two novel methods that overcome these limitations by adopting the parameterized algorithm to handle unrestricted median tree problems (i.e., unrooted binary gene trees sampled from different species).

We introduced a graph-theoretic formulation of the Robinson-Foulds (RF) median tree problem where optimal trees relate to minimum node weight cliques in a compatibility graph. The clique-based heuristic searches for optimal cliques in this graph, in contrast to standard local search heuristics that search the space of all candidate median trees. Investigating how the clique-based heuristic and standard local search heuristics may relate, we found that the clique-based heuristic can be equivalently expressed as a standard local search heuristic albeit using a distinct tree edit operation. This result shows that the clique-based heuristic has the ability to improve on RF median tree estimated of standard local searches.

While the RF distance measure has been used widely to compare phylogenetics trees in computational biology, the distance is poorly distributed, shows insufficient discrimination, and is too sensitive to tree edit operations. To overcome these shortcomings, we introduced a new tree distance measure which is a metric on the space of rooted trees. The distance can be

computed in polynomial time in contrast to edit distances under tree edit operations.

Finally, we proved that the Highly Bi-Connected Subgraph (HBCS) problem is NP-hard, and we describe an ILP formulation and an effective heuristic. The comparative study displays accuracy of our heuristic by comparing its results with exact ILP solutions.

BIBLIOGRAPHY

Allen, B. L. and Steel, M. (2001). Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of combinatorics*, 5:1–15.

Andreopoulos, B., An, A., Wang, X., Faloutsos, M., and Schroeder, M. (2007). Clustering by common friends finds locally significant proteins mediating modules. *Bioinformatics*, 23(9):1124–1131.

Arrow, K. J. (1952). *Social Choice and Individual Values*. Yale University Press, New Haven, Connecticut, United States.

Arvestad, L., Berglund, A.-C., Lagergren, J., and Sennblad, B. (2004). Gene tree reconstruction and orthology analysis based on an integrated model for duplications and sequence evolution. In *Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, pages 326–335. ACM.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., et al. (2000). Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29.

Bansal, M. S., Burleigh, J. G., and Eulenstein, O. (2010a). Efficient genome-scale phylogenetic analysis under the duplication-loss and deep coalescence cost models. *BMC bioinformatics*, 11 Suppl 1:S42.

Bansal, M. S., Burleigh, J. G., Eulenstein, O., and Fernández-Baca, D. (2010b). Robinson-Foulds supertrees. *Algorithms for molecular biology*, 5:18.

Bansal, M. S. and Eulenstein, O. (2013). Algorithms for genome-scale phylogenetics using gene tree parsimony. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(4):939–56.

Bansal, M. S. and Shamir, R. (2011). A Note on the Fixed Parameter Tractability of the Gene-Duplication Problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(3):848–50.

Bayzid, M., Mirarab, S., and Warnow, T. (2013). Inferring optimal species trees under gene duplication and loss. In *Proc. Pacific symposium on biocomputing*, volume 18, pages 250–261.

Bender, M. A. and Farach-Colton, M. (2000). The LCA Problem Revisited. In *Proceedings of the Latin American Symposium on Theoretical Informatics*, pages 88–94, New York, New York, United States. Springer.

Betkier, A., Szczęsny, P., and Górecki, P. (2015). Fast algorithms for inferring gene-species associations. In *International Symposium on Bioinformatics Research and Applications*, pages 36–47. Springer.

Bininda-Emonds, O. R. (2004). *Phylogenetic Supertrees. Combining Information to Reveal the Tree of Life.* Springer Science & Business Media, New York, New York, United States.

Bininda-Emonds, O. R., Gittleman, J. L., and Steel, M. A. (2002). The (Super)tree of life: Procedures, problems, and prospects. *The (Super)tree of life: Procedures, problems, and prospects*, 33:265–289.

Bordewich, M. and Semple, C. (2005). On the computational complexity of the rooted subtree prune and regraft distance. *Annals of combinatorics*, 8(4):409–423.

Boykin, L. M., Kubatko, L. S., and Lowrey, T. K. (2010). Comparison of methods for rooting phylogenetic trees: a case study using Orcuttieae (Poaceae: Chloridoideae). *Molecular Phylogenetics and Evolution*, 54(3):687–700.

Bryant, D. (2003). A Classification of Consensus Methods for Phylogenetics. In *Discrete Mathematics and Theoretical Computer Science*, volume 61, pages 163–185. American Mathematical Society, Providence, Rhode Island, United States.

Bryant, D. and Steel, M. (2009). Computing the distribution of a tree metric. *IEEE/ACM transactions on computational biology and bioinformatics*, 6(3):420–426.

Bu, D., Zhao, Y., Cai, L., Xue, H., Zhu, X., Lu, H., Zhang, J., et al. (2003). Topological structure analysis of the protein–protein interaction network in budding yeast. *Nucleic Acids Research*, 31(9):2443–2450.

Burleigh, J. G., Bansal, M. S., Eulenstein, O., Hartmann, S., Wehe, A., and Vision, T. J. (2011). Genome-Scale Phylogenetics: Inferring the Plant Tree of Life from 18,896 Gene Trees. *Systematic Biology*, 60(2):117–125.

Cannone, J. J., Subramanian, S., Schnare, M. N., Collett, J. R., D'Souza, L. M., Du, Y., Feng, B., Lin, N., Madabusi, L. V., Müller, K. M., et al. (2002). The comparative rna web (crw) site: an online database of comparative sequence and structure information for ribosomal, intron, and other rnas. *BMC bioinformatics*, 3(1):2.

Chang, W.-C., Burleigh, G. J., Fernández-Baca, D., and Eulenstein, O. (2011). An ILP solution for the gene duplication problem. *BMC bioinformatics*, 12(Suppl 1):S14.

Chang, W.-C., Górecki, P., and Eulenstein, O. (2013). Exact Solutions for Species Tree Inference from discordant Gene Trees. *Journal of Bioinformatics and Computational Biology*, 11(05):1342005.

Chang, W.-C., Vakati, S., Krause, R., and Eulenstein, O. (2012). Exploring biological interaction networks with tailored weighted quasi-bicliques. *BMC Bioinformatics*, 13(Suppl 10):S16.

Chaudhary, R., Bansal, M. S., Wehe, A., Fernández-Baca, D., and Eulenstein, O. (2010). iGTP: a software package for large-scale gene tree parsimony analysis. *BMC bioinformatics*, 11:574.

Chen, D., Eulenstein, O., Fernández-Baca, D., and Burleigh, J. G. (2006). Improved heuristics for minimum-flip supertree construction. *Evolutionary bioinformatics online*, 2:347–56.

Consortium, I. H. G. S. et al. (2004). Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–945.

Consortium, U. et al. (2014). Uniprot: a hub for protein information. *Nucleic Acids Research*, 43:989.

Cotton, J. A. and Page, R. D. (2002). Going nuclear: gene family evolution and vertebrate phylogeny reconciled. *Proceedings of the Royal Society of London B: Biological Sciences*, 269(1500):1555–1561.

Cotton, J. A. and Page, R. D. (2005). Rates and Patterns of Gene Duplication and Loss in the Human Genome. *Proceedings of the Royal Society of London B: Biological Sciences*, 272(1560):277–83.

Dankelmann, P. and Volkmann, L. (1995). New sufficient conditions for equality of minimum degree and edge-connectivity. *Ars Combinatoria*, 40:270–278.

DasGupta, B., He, X., Jiang, T., Li, M., Tromp, J., and Zhang, L. (1997). On distances between phylogenetic trees. In *SODA*, volume 97, pages 427–436.

Day, W. H. (1985). Optimal algorithms for comparing trees with labeled leaves. *Journal of Classification*, 2(1):7–28.

Dobzhansky, T. (2013). Nothing in biology makes sense except in the light of evolution. *The american biology teacher*, 75(2):87–91.

Eulenstein, O., Huzurbazar, S., and Liberles, D. A. (2010). *Evolution after Gene Duplication*, chapter Reconciling Phylogenetic Trees, pages 185–206. Wiley, Hoboken, New Jersey, United States.

Ezkurdia, I., Juan, D., Rodriguez, J. M., Frankish, A., Diekhans, M., Harrow, J., Vazquez, J., Valencia, A., and Tress, M. L. (2014). Multiple evidence strands suggest that there may be as few as 19 000 human protein-coding genes. *Human molecular genetics*, 23(22):5866–5878.

Felsenstein, J. (1981). Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17(6):368–376.

Finden, C. and Gordon, A. (1985). Obtaining common pruned trees. *Journal of Classification*, 2(1):255–276.

Fitch, W. M. (1971). Toward Defining the Course of Evolution: minimum change for a specific tree topology. *Systematic Biology*, 20(4):406–416.

Fleischauer, M. and Böcker, S. (2016). Collecting reliable clades using the Greedy Strict Consensus Merger. *PeerJ Computer Science*, 4:e2172.

Forster, P. and Renfrew, C. (2006). *Phylogenetic methods and the prehistory of languages*. McDonald Inst of Archeological, Cambridge, England, United Kingdom.

Friedberg, I. (2006). Automated protein function predictionâĂŤthe genomic challenge. *Briefings in Bioinformatics*, 7(3):225–242.

Geva, G. and Sharan, R. (2011). Identification of protein complexes from co-immunoprecipitation data. *Bioinformatics*, 27(1):111–117.

Goodman, M., Czelusniak, J., Moore, G. W., Romero-Herrera, A., and Matsuda, G. (1979). Fitting the Gene Lineage into its Species Lineage, a Parsimony Strategy Illustrated by Cladograms Constructed from Globin Sequences. *Systematic Biology*, 28(2):132–163.

Górecki, P. and Eulenstein, O. (2012). A Robinson-Foulds Measure to Compare Unrooted Trees with Rooted Trees. In *Proceedings of the International Symposium on Bioinformatics Research and Applications*, pages 115–126, New York, New York, United States. Springer.

Górecki, P. and Eulenstein, O. (2015). Gene tree diameter for deep coalescence. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 12(1):155–165.

Harding, E. (1971). The probabilities of rooted tree-shapes generated by random bifurcation. *Advances in Applied Probability*, 3(1):44–77.

Harris, S. R., Cartwright, E. J., Török, M. E., Holden, M. T., Brown, N. M., Ogilvy-Stuart, A. L., Ellington, M. J., Quail, M. A., Bentley, S. D., Parkhill, J., and Peacock, S. J. (2013). Whole-genome sequencing for analysis of an outbreak of meticillin-resistant staphylococcus aureus: a descriptive study. *The Lancet Infectious diseases*, 13(2):130–136.

Hartuv, E., Schmitt, A. O., Lange, J., Meier-Ewert, S., Lehrach, H., and Shamir, R. (2000). An algorithm for clustering cDNA fingerprints. *Genomics*, 66(3):249–256.

Hartuv, E. and Shamir, R. (2000). A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(4):175–181.

Hedges, S. B., Kumar, S., Tamura, K., and Stoneking, M. (1992). Human origins and analysis of mitochondrial dna sequences. *Science*, 255(7):737–739.

Hickey, G., Dehne, F., Rau-Chaplin, A., and Blouin, C. (2008). Spr distance computation for unrooted trees. *Evolutionary bioinformatics online*, 4:17.

Holland, B., Penny, D., and Hendy, M. (2003). Outgroup misplacement and phylogenetic inaccuracy under a molecular clockâĂŤa simulation study. *Systematic biology*, 52(2):229–238.

Huelsenbeck, J. P., Bollback, J. P., Levine, A. M., and Olmstead, R. (2002). Inferring the root of a phylogenetic tree. *Systematic biology*, 51(1):32–43.

Hufbauer, R. A., Marrs, R. A., Jackson, A. K., Sforza, R., Bais, H. P., Vivanco, J. M., and Carney, S. E. (2003). Population structure, ploidy levels and allelopathy of Centaurea maculosa (spotted knapweed) and C. diffusa (diffuse knapweed) in North America and Eurasia. In *Proceedings of the International Symposium on Biological Control of Weeds*, pages 121–126, Morgantown, WV. USDA Forest Service. Forest Health Technology Enterprise Team.

Hüffner, F., Komusiewicz, C., Liebtrau, A., and Niedermeier, R. (2014). Partitioning biological networks into highly connected clusters with maximum edge coverage. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 11(3):455–467.

Hüffner, F., Komusiewicz, C., and Sorge, M. (2015). Finding highly connected subgraphs. In *SOFSEM 2015: Theory and Practice of Computer Science*, pages 254–265. Springer.

Huson, D. H., Nettles, S. M., and Warnow, T. J. (1999). Disk-Covering, a Fast-Converging Method for Phylogenetic Tree Reconstruction. *Journal of Computational Biology*, 6(3–4):369–386.

Jackson, A. P. (2004). A reconciliation analysis of host switching in plant-fungal symbioses. *Evolution*, 58(9):1909–23.

Karp, R. M. (1972). *Reducibility among combinatorial problems*. Springer.

Kennedy, M., Page, R. D., and Prum, R. (2002). Seabird supertrees: combining partial estimates of procellariiform phylogeny. *The Auk*, 119(1):88–108.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97.

Lin, H. T., Burleigh, J. G., and Eulenstein, O. (2011). The deep coalescence consensus tree problem is pareto on clusters. In *International Symposium on Bioinformatics Research and Applications*, pages 172–183. Springer.

Lin, H. T., Burleigh, J. G., and Eulenstein, O. (2012a). Consensus Properties for the Deep Coalescence Problem and their Application for Scalable Tree Search. *BMC Bioinformatics*, 13(10):S12.

Lin, Y., Rajan, V., and Moret, B. M. (2012b). A metric for phylogenetic trees based on matching. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(4):1014–1022.

Ma, B., Li, M., and Zhang, L. (2000). From Gene Trees to Species Trees. *SIAM Journal on Computing*, 30(3):729–752.

Maddison, W. P. and Knowles, L. L. (2006). Inferring Phylogeny Despite Incomplete Lineage Sorting. *Systematic Biology*, 55(1):21–30.

Martin, A. P. and Burg, T. M. (2002). Perils of paralogy: using hsp70 genes for inferring organismal phylogenies. *Systematic Biology*, 51(4):570–587.

McGowen, M. R., Clark, C., and Gatesy, J. (2008). The vestigial olfactory receptor subgenome of odontocete whales: phylogenetic congruence between gene-tree reconciliation and supermatrix methods. *Systematic biology*, 57(4):574–590.

McMorris, F. and Steel, M. A. (1993). The complexity of the median procedure for binary trees. *Proceedings of the international federation of classification societies.*

Moon, J. and Eulenstein, O. (2016). Synthesizing Large-Scale Species Trees using Guidance Trees. In *Proceedings of the International Conference on Bioinformatics and Computational Biology*, pages 103–108, Red Hook, New York, United States. Curran Associates, Inc.

Moon, J., Lin, H. T., and Eulenstein, O. (2016). Consensus Properties and their Large-Scale Applications for the Gene Duplication Problem. *Journal of Bioinformatics and Computational Biology*, 14(03):1642005.

Moore, G. W., Goodman, M., and Barnabas, J. (1973). An iterative approach from the standpoint of the additive hypothesis to the dendrogram problem posed by molecular data sets. *Journal of theoretical biology*, 38(3):423–457.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38.

Nik-Zainal, S., Loo, P. V., Wedge, D. C., Alexandrov, L. B., Greenman, C. D., Lau, K. W., Raine, K., Jones, D., Marshall, J., Ramakrishna, M., Shlien, A., Cooke, S. L., Hinton, J., Menzies, A., Stebbings, L. A., Leroy, C., Jia, M., Rance, R., Mudie, L. J., Gamble, S. J., Stephens, P. J., McLaren, S., Tarpey, P. S., Papaemmanuil, E., Davies, H. R., Varela, I., McBride, D. J., Bignell, G. R., Leung, K., Butler, A. P., Teague, J. W., Martin, S., Jönsson, G., Mariani, O., Boyault, S., Miron, P., Fatima, A., Langerød, A., Aparicio, S. A., Tutt, A., Sieuwerts, A. M., Borg, A., Thomas, G., Salomon, A. V., Richardson, A. L., Børresen-Dale, A.-L., Futreal, P. A., Stratton, M. R., and Campbell, P. J. (2012). The Life History of 21 Breast Cancers. *Cell*, 149(5):994–1007.

Page, R. D. (1998). GeneTree: comparing gene and species phylogenies using reconciled trees. *Bioinformatics*, 14(9):819–820.

Page, R. D. (2000). Extracting species trees from complex gene trees: reconciled trees and vertebrate phylogeny. *Molecular Phylogenetics and Evolution*, 14:89–106.

Page, R. D. and Charleston, M. A. (1997). Reconciled trees and incongruent gene and species trees. *Math Hierach Biol*, 37:57–70.

Pamilo, P. and Nei, M. (1988). Relationships between gene trees and species trees. *Molecular biology and evolution*, 5(5):568–583.

Pattengale, N., Aberer, A., Swenson, K., Stamatakis, A., and Moret, B. (2011). Uncovering hidden phylogenetic consensus in large data sets. *IEEE/ACM transactions on computational biology and bioinformatics*, 8(4):902–911.

Pattengale, N. D., Gottlieb, E. J., and Moret, B. M. (2007). Efficiently computing the Robinson-Foulds metric. *Journal of computational biology*, 14(6):724–35.

Pennisi, E. (2012). Encode project writes eulogy for junk dna. *Science*, 337(6099):1159–1161.

Pržulj, N., Wigle, D. A., and Jurisica, I. (2004). Functional topology in a network of protein interactions. *Bioinformatics*, 20(3):340–348.

Pyron, R. A. and Wiens, J. J. (2011a). A large-scale phylogeny of Amphibia including over 2800 species, and a revised classification of extant frogs, salamanders, and caecilians. *Molecular Phylogenetics and Evolution*, 61(2):543–583.

Pyron, R. A. and Wiens, J. J. (2011b). Data from: A large-scale phylogeny of Amphibia including over 2800 species, and a revised classification of extant frogs, salamanders, and caecilians. http://datadryad.com/resource/doi:10.5061/dryad.vd0m7.

Radivojac, P., Clark, W. T., Oron, T. R., Schnoes, A. M., Wittkop, T., Sokolov, A., Graim, K., Funk, C., Verspoor, K., Ben-Hur, A., et al. (2013). A large-scale evaluation of computational protein function prediction. *Nature Methods*, 10(3):221–227.

Ranwez, V., Berry, V., Criscuolo, A., Fabre, P.-H., Guillemot, S., Scornavacca, C., and Douzery, E. J. (2007). PhySIC: A Veto Supertree Method with Desirable Properties. *Systematic Biology*, 56(5):798–817.

Robinson, D. F. (1971). Comparison of labeled trees with valency three. *Journal of combinatorial theory, Series B*, 11(2):105–119.

Robinson, D. F. and Foulds, L. R. (1981). Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1):131–147.

Roux, J. J. L., Wieczorek, A. M., Ramadan, M. M., and Tran, C. T. (2006). Resolving the native provenance of invasive fireweed (Senecio madagascariensis Poir.) in the Hawaiian Islands as inferred Poir.) in the Hawaiian Islands as inferred from phylogenetic analysis. *Diversity and Distributions*, 12:694–702.

Saitou, N. and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425.

Semple, C. and Steel, M. A. (2003a). *Phylogenetics*, volume 24. Oxford University Press, Oxford.

Semple, C. and Steel, M. A. (2003b). *Phylogenetics*. Oxford University Press, Oxford, England, United Kingdom.

Sharan, R., Ulitsky, I., and Shamir, R. (2007). Network-based prediction of protein function. *Molecular Systems Biology*, 3(1):88.

Stamatakis, A. (2014). RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30(9):1312–1313.

Steel, M., Dress, A. W., and Böcker, S. (2000). Simple but fundamental limitations on supertree and consensus tree methods. *Syst Biol*, 49(2):363–368.

Steel, M. A. and Penny, D. (1993). Distributions of tree comparison metricsâĂŤsome new results. *Systematic biology*, 42(2):126–141.

Sukumaran, J. and Holder, M. T. (2010). Dendropy: a python library for phylogenetic computing. *Bioinformatics*, 26(12):1569–1571.

Swofford, D. and Olsen, G. (1990). *Phylogeny reconstruction. In 'Molecular Systematics'.(Eds DM Hillis and C. Moritz.) pp. 411–501.* Sinauer Associates: Sunderland, Massachusetts.

Than, C. and Nakhleh, L. (2009a). Species Tree Inference by Minimizing Deep Coalescences. *PLOS Computational Biology*, 5(9):e1000501.

Than, C. and Nakhleh, L. (2009b). Species tree inference by minimizing deep coalescences. *PLoS computational biology*, 5(9):e1000501.

Wang, L. (2013). Near optimal solutions for maximum quasi-bicliques. *Journal of Combinatorial Optimization*, 25(3):481–497.

Waterman, M. S. and Smith, T. F. (1978). On the similarity of dendrograms. *Journal of Theoretical Biology*, 73(4):789–800.

Wehe, A., Bansal, M. S., Burleigh, J. G., and Eulenstein, O. (2008). DupTree: A program for large-scale phylogenetic analyses using gene tree parsimony. *Bioinformatics*, 24(13):1540–1541.

Wehe, A. and Burleigh, J. G. (2010). Scaling the gene duplication problem towards the Tree of Life: Accelerating the rSPR heuristic search. In *Proceedings of the International Conference on Bioinformatics and Computational Biology*, pages 133–138, Red Hook, New York, United States. Curran Associates, Inc.

Wehe, A., Burleigh, J. G., and Eulenstein, O. (2013). Efficient algorithms for knowledge-enhanced supertree and supermatrix phylogenetic problems. *IEEE/ACM transactions on computational biology and bioinformatics*, 10(6):1432–1441.

Wilkinson, M., Cotton, J. A., Lapointe, F.-J., and Pisani, D. (2007). Properties of Supertree Methods in the Consensus Setting. *Systematic Biology*, 56(2):330–337.

Zhang, L. (1997). On a mirkin-muchnik-smith conjecture for comparing molecular phylogenies. *Journal of Computational Biology*, 4(2):177–187.