2012

Phylogeny reconciliation under gene tree parsimony

Wen-Chieh Chang
Iowa State University

Phylogeny reconciliation under gene tree parsimony

by

Wen-Chieh Chang

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

Julie Dickerson

David Fernández-Baca

Dennis Lavrov

Leslie Miller

Iowa State University

Ames, Iowa

2012

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The growing genomic and phylogenetic data sets represent a unique opportunity to analytically and computationally study the relationship among diversifying species. Unfortunately, such data often result in contradictory gene phylogenies due to common yet unobserved evolutionary events, e.g., gene duplication or deep coalescence. Gene tree parsimony (GTP) methods address such issue by reconciling gene phylogenies into one consistent species evolutionary history as well as identifying the underlying events. In this study, we solve not only the GTP problem but also propose a new method to select gene trees in order to assist biologists in gaining insight from phylogenetic analysis.

First, we introduce exact solutions for the intrinsically complex GTP problem. Exact solutions for NP-hard problems, like GTP, have a long and extensive history of improvements for classic problems such as traveling salesman and knapsack. Our solutions presented here are designed via integer linear programming (ILP) and dynamic programming (DP), which are techniques widely used in solving problems of similar complexity. We also demonstrate the effectiveness of our solutions through simulation analysis and empirical datasets.

To ensure input data coherence for GTP analysis, as a method to strengthen species represented in a gene tree, we introduce the quasi-biclique (QBC) approach to analyze and condense input datasets. In order to take advantage of emerging techniques that further describe the sequence-host and gene-taxon relations, quasi-bicliques are optimized via weighted edge connectivities and distribution of missing information. Our study showed these QBC mining problems are NP-hard. We describe an ILP formulation that is capable of finding optimal QBCs in an effort to support GTP analysis. We also investigate the applicability of QBC to other applications such as mining genetic interaction networks to encouraging results.

CHAPTER 1. INTRODUCTION

1.1 Motivation

Modern DNA sequencing technologies provide an unprecedented wealth of gene and genomic data from which to infer phylogenetic trees. These snapshots of evolutionary histories are often affected by unapparent events like gene duplication and loss, deep coalescence, and horizontal gene transfers which then incur discrepancies between gene phylogenies and a species phylogenies. Based on these biological models, we can reconcile the differences by observing their topological information. The objective of reconciling such differences between a gene phylogeny and a species phylogeny is generally motivated by the parsimony principle. In this study, we focus the driving force of these evolution events to gene duplication and loss (GD), which can be seen as a fitness measurement of gene trees [28, 53, 30] to the species tree.

Further utilizing this reconciliation process, we can infer the unknown species phylogeny from a collection of gene phylogenies through different criteria under the parsimony principle. This gene tree parsimony (GTP) approach is a well-studied method [54, 13], and it serves as an important tool in phylogeny and supertree research.

In this study, we improve the solutions to GTP problem by solving critical steps involved in the analytical process. Before we present an overview of this study, we first briefly yet formally describe the GD model and the GTP problem.

1.2 Gene Duplication Model

As mentioned, gene trees undergone gene duplication and loss processes may result in a different topology from the species topology, which supposedly describes a species level of divergence. Although intuitively, we should observe the same divergence at the level of genomic

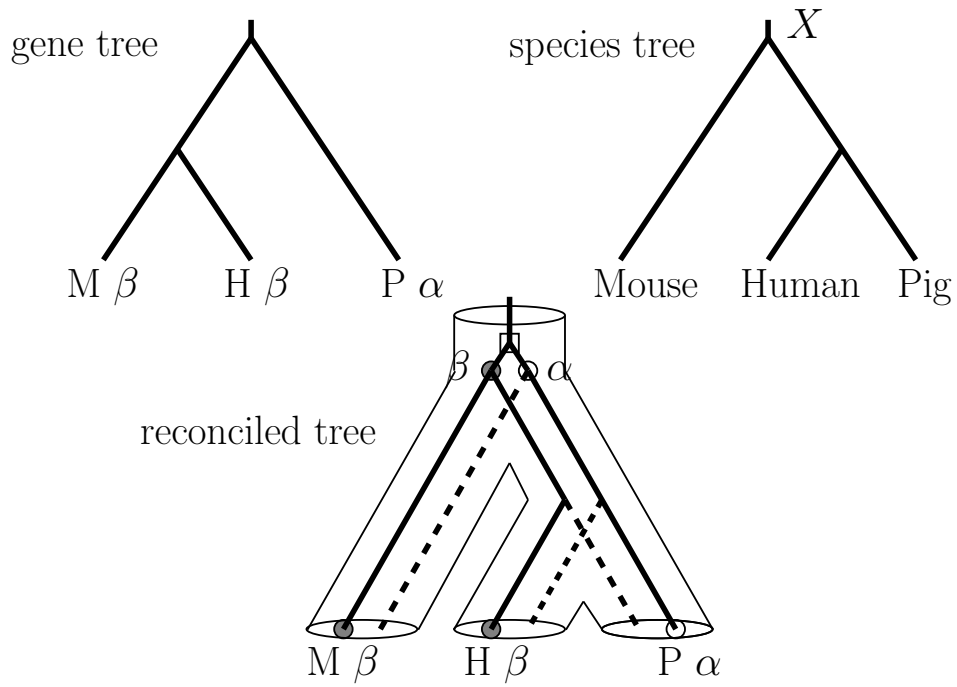

Figure 1.1: An example showing how the gene duplication model explains the differences between a gene tree and a species tree. While topologically different, the gene tree can be embedded into the species tree as expressed by the reconciled tree showing in solid lines.

evolution, it is not always the case, and a reconciliation process is required to explain the difference. Figure 1.1 depicts an example, which reconciles the inconsistencies between a gene tree and a species tree using gene duplications and subsequent losses. A gene duplication copies gene from the host X into the copies α and β that takes place prior to the speciation following X in the species tree. Then each of these copies undergoes the speciation events described by the species tree, which observe as species Mouse, Human and Pig. During this process three losses let genes M-α, H-α and P-β become extinct. Only genes M-β, H-β and P-α survive, which are the leaves of the gene tree. In summary, the inconsistency between the trees is reconciled by invoking one gene duplication and three losses. This is the minimum number of either gene duplications alone or gene duplications plus losses that are needed to reconcile the given gene tree with the given species tree. Therefore, the reconciliation distance from the gene tree to the species tree is one duplication (and four for considering both duplication and loss events). Such cost from a pair of gene tree and a species tree can be calculated efficiently based on established understandings of embedding the gene tree to the species tree [24, 8] and

efficient algorithm [3].

We will define the computational problem to find an optimal reconciliation scenario in the next chapter.

1.3 Gene Tree Parsimony Problem

The GTP problem seeks a species tree that implies the minimum reconciliation cost for a given collection of gene trees. The reconciliation cost for a species tree and a given collection of gene trees is the minimum number of evolutionary events that are invoked by the species tree to explain the conflicts between the gene trees. Although a variety of GTP problems have been introduced that use different types of evolutionary events (e.g., deep coalesce, horizontal gene transfer) to define the reconciliation cost, we define the cost measure by the duplication events. Again, we defer a formal definition of the GTP problem to the next chapter.

It has been shown that the GTP problem is NP-hard [45]. Furthermore, the GTP problem has no polynomial time solution while fixing the duplication cost as the parameter, and unless the long standing computational complexity debate results in $P = NP$, there is no approximation solutions within a logarithmic factor to the input size [1].

Previously, the GTP problem has been solved using branch-and-bound [21], local search heuristic [21], and a fixed-parameter tractable solution parameterized by the width of input trees [34]. In this work, we invistage the problem by finding its exact solutions.

1.4 Exact Solutions to NP-hard Problems

Exact solutions to NP-hard problems have a long and rich history that is comparable to alternative solutions focusing on efficiencies over qualities. In recent years, driven by a combination of improved computing hardware and analytic techniques, exact solutions of well studied classic problems (such as travelling salesman and independent set) are capable of solving inputs of moderate sizes [65, 26].

The advantage of studying exact solutions includes providing the optimal solutions and gaining fundamental knowledge to the solutions. We may further design alternative solutions

with good efficiencies, or evaluate other approaches against the optimal solution as a benchmark.

Here, we present exact solutions to the computational problems involved in this study. Particularly, we utilize integer linear programming and dynamic programming techniques which are commonly used in solving combinatorial optimization problems [56].

1.5 Our Proposed Approach

As previously introduced, the center of this study is the GTP problem. In addition to solving the GTP problem, it is also brought to our attention that in a typical GTP analysis, the first critical step is to assemble a biologically meaningful input data set. Although the number of gene phylogenies are increasing, there is inevitably missing information such that each gene phylogeny may cover different sets of species taxa.

This gene-taxon (or sequence-taxon) relation provides an invaluable opportunity where we investigate different approaches to properly select a dense collection of gene phylogenies that are highly related to each other. In this study, we introduce weighted quasi-biclique (WQBC) where missing information is allowed in an evenly distributed manner.

In the following sections, we first describe the solutions of GTP, including simulations and emprical tests for their effectiveness. Next, we describe the QBC optimization problem and its solution.

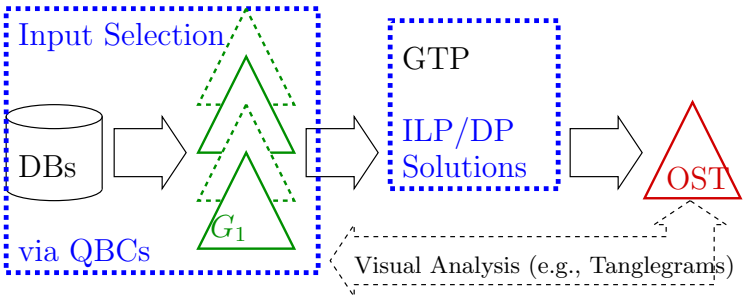

Figure 1.2: Our main components (highlighted in blue-dash areas) in solving the GTP reconciliation problem interactively.

CHAPTER 2. NOTATIONS AND DEFINITIONS

This chapter describes commonly used notations across different chapters. Further notations that is only relevent to a given chapter will be described in the corresponding chapter only. Notations of graphs and trees are adopted from Cormen *et al.* [10], and Semple *et al.* [60].

2.1 Basic Notations

In order to describe a phylogenetic tree, we rely on well accepted notions to represent given hierarchies.

A *rooted tree* T is a connected and acyclic graph consisting of a vertex set $V(T)$, an edge set $E(T)$, and that has exactly one distinguished vertex called *root*, which is denoted by $\mathsf{Rt}(T)$. Let T be a rooted tree. Define $\leq_T$ to be the partial order on $V(T)$, where $u \leq_T v$ if v is a vertex on the path between $\mathsf{Rt}(T)$ and u. Moreover, we write $u <>_T v$ if neither $u \leq_T v$ nor $v \leq_T u$ is true. The set of minima under $\leq_T$ is denoted by $L(T)$ and its elements are called *leaves*. We call u a *child* of v if $u \leq v$ and $\{u, v\} \in V(E)$. The set of all children of v is denoted by $\mathsf{Ch}_T(v)$. For a vertex $v \in V(T)$ we denote by $T(v)$ the subtree of T that consists of all vertices $u \leq_T v$. The *least common ancestor* of a non-empty subset $X \subseteq V(T)$, denoted as $\mathsf{LCA}_T(X)$, is the unique smallest upper bound of X under $\leq_T$. Tree T is called *full binary* if every vertex has either two or zero children. Throughout this work, all trees are rooted and full binary unless stated otherwise.

Recall that the objective of the afreomentioned, although informally, GTP problem, is to seek an optimal species phylogeny from a set of given gene phylogenies. In addition to represent each phylogeny as a tree, we also represent the binary relation between phylogenies and their taxa using a *bipartite graph*.

A *bipartite graph*, denoted by $(U+V, E)$, is a graph whose vertex set can be partitioned into the sets U and V such that its edge set E consists only of edges $\{u, v\}$ where $u \in U$ and $v \in V$ (U and V are independent sets). Let $G := (U + V, E)$ be a bipartite graph. The graph G is called *complete* if for any two vertices $u \in U$ and $v \in V$ there is an edge $\{u, v\} \in E$. A *biclique* in G is a pair (U', V') that induces a complete bipartite subgraph in G, where $U' \subseteq U$ and $V' \subseteq V$. Since any subgraph induced by a biclique is a complete bipartite graph, we use the two terms interchangeably. A pair (U, V) *includes* another pair (U', V') if $U' \subseteq U$ and $V' \subseteq V$. In such case, we also say that the pair (U', V') is *included* in (U, V). A pair (U, V) is non-empty if both U and V are non-empty. A *weighted bipartite graph*, denoted by $(U + V, E, \omega)$, is a complete bipartite graph $(U + V, E)$ with a weight function $\omega\colon E \to [0, 1]$.

2.2 Definitions

We now formally define the Gene Duplication (GD) problem.

2.2.1 Gene Duplication (GD) Problem

The terms *species tree* and *gene tree* refer to trees that represent the evolutionary history of a gene family or species respectively. In this section, let S be a species tree and G a gene tree.

To compare a gene tree with a species tree, a mapping from each gene in the gene tree to the most recent species in the species tree that could have contained the gene is required. A formal definition of such a mapping function follows.

Definition 1 (Mapping)**.** *A* leaf-mapping *from G to S is a function $\mathcal{L}_{G,S}\colon L(G) \to L(S)$. The* extension $\mathcal{M}_{G,S}\colon V(G) \to V(S)$ *of the leaf-mapping $\mathcal{L}_{G,S}$ is the mapping defined by $\mathcal{M}_{G,S}(u) := \mathsf{LCA}_S(\mathcal{L}_{G,S}(G(u)))$.*

To simplify the exposition we shall assume that leaf-mappings are injections, and w.l.o.g. we identify the genes with the species from which they were sampled. After describing our ILP solution for identity leaf-mappings, we extend this formulation to cover non-injective leaf-mappings.

Definition 2 (Comparable)**.** *A gene tree G is* comparable *to S, denoted by $G \vdash S$, if $\mathcal{L}_{G,S}$ exists. A set of gene trees is* comparable *to S, denoted by $\mathcal{G} \vdash S$, if $G \vdash S$ for each gene tree $G \in \mathcal{G}$.*

We shall adopt the following notation: we use S for a species tree, $\mathcal{G}$ for a set of gene trees that is comparable to S, and G for an gene tree in $\mathcal{G}$.

Definition 3 (Duplication)**.** *A node $g \in V(G)$ is a* duplication *(w.r.t. S) if $\mathcal{M}_{G,S}(g) \in \mathcal{M}_{G,S}(\mathsf{Ch}_G(g))$.*

For consistency we follow the common practice to call what is stated above a definition, even though it is actually a theorem [24] that follows from the gene duplication model [28].

Definition 4 (Duplication cost)**.** *We define the following duplication costs:*

1. $\mathsf{Dup}(G,S) := |\{g \in V(G) \colon g \text{ is a duplication}\}|$ *is the* duplication cost *from G to S.*
2. $\mathsf{Dup}(\mathcal{G},S) := \Sigma_{G\in\mathcal{G}}\, \mathsf{Dup}(G,S)$ *is the* duplication cost *from $\mathcal{G}$ to S.*
3. $\mathsf{Dup}(\mathcal{G}) := \min_{\mathcal{G}\vdash T}\, \mathsf{Dup}(\mathcal{G},T)$ *is the* duplication cost *of $\mathcal{G}$.*

Problem 1 (Gene-Duplication (GD))**.**

Instance: A set of gene trees $\mathcal{G}$.

Find: The duplication cost $\mathsf{Dup}(\mathcal{G})$, and a species tree S^* such that $\mathsf{Dup}(\mathcal{G},S^*) = \mathsf{Dup}(\mathcal{G})$.

2.2.2 Weighted Quasi-biclique Problems

Although the concept of a quasi-biclique is not new (refs), we generalize its scope to incoporate edge weights. Our exploration to this new territory results in a family of related quasi-biclique optimization problems.

Definition 5 (α,β-WQB$_P$)**.** *Let $G := (U+V, E, \omega)$ and $\alpha, \beta \in [0,1]$. A percentage version α,β-*weighted quasi-biclique*, denoted as α,β-WQB$_P$, in G is a non-empty pair (U',V') that is included in (U,V) and satisfies the two properties:*

1. $\forall u \in U'$: $\sum_{v \in V'} \omega(u,v) \geq \alpha|V'|$, *and*

2. $\forall v \in V'$: $\sum_{u \in U'} \omega(u,v) \geq \beta|U'|$.

Definition 6 (α,β-WQB$_C$)**.** *Let* $G := (U+V, E, \omega)$ *and* $\alpha, \beta \in [0, \infty)$. *A constant version* α, β-weighted quasi-biclique*, denoted as* α,β-WQB$_C$ *, in* G *is a non-empty pair* (U', V') *that is included in* (U, V) *and satisfies the two properties:*

1. $\forall u \in U'$: $\sum_{v \in V'} \omega(u,v) \geq |V'| - \alpha$, *and*

2. $\forall v \in V'$: $\sum_{u \in U'} \omega(u,v) \geq |U'| - \beta$.

Definition 7 (Maximum α,β-WQB$_{P(C)}$)**.** *A* α,β-$WQB_{P(C)}$*, is a* maximum α,β-$WQB_{P(C)}$ *of a weighted bipartite graph* $G := (U+V, E, \omega)$*, if its weight is at least as much as the weight of any other* α,β-$WQB_{P(C)}$ *in* G *for given values of* α *and* β

Naturally, the optimization problems under different α,β-WQB criteria seeks a solution that yields the most total edge weight.

Problem 2 (α,β-WQB$_{P(C)}$)**.**

Instance: A weighted bipartite graph $G := (U+V, E, \omega)$*, and values* $\alpha, \beta \in [0,1]([0,\infty))$.

Find: A maximum weighted α,β-$WQB_{P(C)}$ *in* G.

To help further refine the resulting α,β-WQB to be relevent to targets of interest, we define another level of customization in finding an optimal α,β-WQB.

Problem 3 ($Query_{P(C)}$)**.**

Instance: A weighted bipartite graph $G := (U+V, E, \omega)$*, values* $\alpha, \beta \in [0,1]$ $([0,\infty))$*, and a pair* (P, Q) *included in* (U, V).

Find: The α,β-$WQB_{P(C)}$ *which includes* (P, Q) *and has a weight greater than or equal to the weight of any* α,β-$WQB_{P(C)}$ *which includes* (P, Q).

In the up-coming chapters, we discuss how these problems can be solved.

CHAPTER 3. SOLVING GTP USING ILP

To solve GTP under gene duplication model, the first approach is a formulation in integer linear programming (ILP). While solving an ILP is still computationally difficult, there are commercial solvers (e.g., CPLEX, Gurobi [1]) as well as open-source packages (e.g., GLPK, lp_solve) available that have been well optimized. The following ILP formulation and the consequent results produced from a collaboration with Dr. J. Gordon Burleigh, Dr. David Fernández-Baca, and Dr. Oliver Eulenstein.

3.1 An Integer Linear Programming Solution

The gene duplication problem seeks a species tree that implies the fewest gene duplication events across a collection of input gene trees. Solving this problem makes it possible to use large gene families with complex histories of duplication and loss in phylogenetic inference. However, the gene duplication problem is intrinsically hard, and therefore, most analysis must use heuristics that lack any performance guarantee. Here we introduce the first integer linear programming (ILP) formulation to solve instances of the gene duplication problem exactly. With simulations, we demonstrate that the ILP formulation can solve problem instances with up to 14 taxa. Furthermore, we apply the new ILP formulation to solve the gene duplication problem for the seed plant phylogeny based on a 12-taxon, 6,084-gene data set. The resulting optimal species tree is depicted in Figure 3.1. The unique, optimal solution, which places Gnetales sister to the conifers, represents a new, large-scale genomic perspective on one of the most puzzling questions in plant systematics.

[1] CPLEX and Gurobi are trademarks of IBM and Gurobi Inc. respectively.

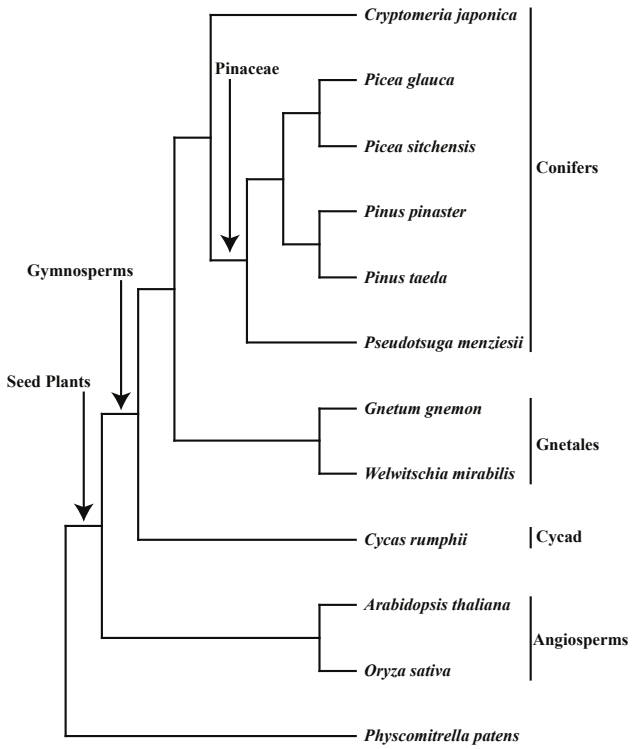

Figure 3.1: The unique optimal seed plant phylogeny based on 12 taxa and 6084 gene trees under the gene duplication model.

3.1.1 The Triple-Inconsistency problem and its equivalence to the GD problem

A *rooted triple* is a tree with three leaves. The rooted triple with leaves x, y, and z is denoted $xy|z$ if the path between x and y does not intersect with the path between z and the root. A rooted triple is *displayed* by a tree T if $\mathsf{LCA}_T(x,y) \leq_T \mathsf{LCA}_T(x,z)\,(= \mathsf{LCA}_T(y,z))$. The set of rooted triples $xy|z$ displayed by tree T that are rooted at vertex $u \in V(T)$, (i.e., $u = \mathsf{LCA}_T(\{x,y,z\})$) is denoted by $\mathsf{Trip}_T(v)$, and the set of all triples displayed by T is denoted by $\mathsf{Trip}(T)$.

Definition 8 (T(riple)-inconsistency)**.** *A rooted triple $t \in \mathsf{Trip}(G)$ is said to be* inconsistent *with S if $t \notin \mathsf{Trip}(S)$. A vertex $v \in V(G)$ is called* t(riple)-inconsistent *with S if there is a rooted triple in $\mathsf{Trip}_G(v)$ that is inconsistent with S.*

Definition 9 (T-inconsistency cost)**.** *We define the following t-inconsistency costs:*

1. $\mathsf{Tin}(G,S) := |\{v \in V(G)\colon v \text{ is t-inconsistent with } S\}|$ *is the* t-inconsistency cost *from G to S.*

2. $\mathsf{Tin}(\mathcal{G}, S) := \Sigma_{G \in \mathcal{G}} \mathsf{Tin}(G, S)$ *is the* t-inconsistency cost *from* $\mathcal{G}$ *to* S.

3. $\mathsf{Tin}(\mathcal{G}) := \min_{\mathcal{G} \vdash T} \mathsf{Tin}(\mathcal{G}, T)$ *is the* t-inconsistency cost *of* $\mathcal{G}$.

Problem 4 (T(riple)-Inconsistency)**.**

Instance: A set of gene trees $\mathcal{G}$.

Find: The t-inconsistency cost $\mathsf{Tin}(\mathcal{G})$*, and species tree* S^* *such that* $\mathsf{Tin}(\mathcal{G}, S^*) = \mathsf{Tin}(\mathcal{G})$.

Theorem 1 (Equivalence between duplication and t-inconsistency)**.** *Let* $u \in V(G)$*. Then* u *is a duplication w.r.t.* S *if and only if* u *is t-inconsistent with* S.

Proof. Let $x := \mathcal{M}_{G,S}(u)$.

Suppose u is a not a duplication. Let $ab|c \in \mathsf{Trip}_G(u)$. We will show that $ab|c \in \mathsf{Trip}(S)$. By the definition of $ab|c \in \mathsf{Trip}_G(u)$ we know that $\mathsf{LCA}_G(\{a, b, c\}) = u$, and together with our assumption that G is fully binary it follows that u has two children v and w, where w.l.o.g. $a, b \in L(G(v))$ and $c \in L(G(w))$. Let $v' := \mathcal{M}_{G,S}(v)$ and $w' := \mathcal{M}_{G,S}(w)$. From $a, b \in L(G(v))$ and $c \in L(G(w))$ follows that $a, b \in L(S(v'))$ and $c \in L(S(w'))$ respectively. Now, since u is not a duplication we have $v' <>_S w'$. Otherwise, we would have $w' \leq_S v'$ or $v' \leq_S w'$ from which $x = v'$ or $x = w'$ would follow respectively; contradicting that v is not a duplication. Hence, from $v' <>_S w'$ and $a, b \in L(S(v'))$ and $c \in L(S(w'))$ follows that $ab|c \in \mathsf{Trip}(S)$.

Suppose u is a duplication, and thus we have $x = \mathcal{M}_{G,S}(v)$ for a child $v \in \mathsf{Ch}(u)$. So u is not a leaf in G, and since G is fully binary it follows that there are two distinct vertices $a, b \in L(G(u))$ such that $\mathsf{LCA}_S(\{a, b\}) = x$. Therefore, x has two children y and z such that $a \leq_S y$ and $b \leq_S z$. Now we distinguish different cases for the vertices a and b based on their possible order relation to the children of u. Since G is fully binary and v is a child of u, there exists a child $w \in \mathsf{Ch}(u)$ where $w \neq v$. Now, we have the following cases.

1. Case 1: $a \leq_G v$, $b \leq_G v$: Let $c \leq_G w$. Then $ab|c \in \mathsf{Trip}_G(u)$. Further $c \leq_S y$ or $c \leq_S z$ and with $a \leq_S y$ and $b \leq_S z$, it follows that either $ac|b \in \mathsf{Trip}_S(x)$ or $bc|a \in \mathsf{Trip}_S(x)$. Hence, u is t-inconsistent as desired.

2. Case 2: $a \leq_G v$, $b \leq_G w$: We know that x has two children y and z and that $\mathcal{M}_{G,S}(v) = x$. Therefore there exist $c \leq_S y$ and $d \leq_S z$ such that $\mathsf{LCA}_S(c,d) = \mathcal{M}(v)$ where $c, d \in L(G(v))$. From the order relations $a \leq_S y, d \leq_S z$ and $d \leq_G v, b \leq_G w$ and $a \leq_G v, b \leq_G w$, it follows that a, b and d are pairwise different. Therefore the rooted triples $ad|b \in \mathsf{Trip}_G(u)$ and $bd|a \in \mathsf{Trip}_S(x)$ are well defined, from which follows that the vertex u is t-inconsistent.

3. Case 3: $a \leq_G w$, $b \leq_G w$ or $b \leq_G v$, $a \leq_G w$: Similarly to the previous cases it follows that u is t-inconsistent. □

From Theorem 1, the next corollary follows.

Corollary 1 (Equivalence between the GD problem and the T-Inconsistency problem)**.** *The t-inconsistency problem is a mathematical equivalent formulation of the duplication problem (i.e.* $\mathsf{Dup}(G,S) = \mathsf{Tin}(G,S)$*).*

An ILP solution for the T-Inconsistency problem

Table 3.1 lists the variables used, and their meaning. To explain our ILP solution, we first formulate all possible candidate trees in the solution space of the t-inconsistency problem. Next we formulate the t-inconsistency objective to identify an optimal t-inconsistency cost and an optimal candidate tree.

Let $X := \bigcup_{G \in \mathcal{G}} L(G)$ be the taxon set, $n := |X|$, $m := |\bigcup_{G \in \mathcal{G}} \mathsf{Trip}(G)|$, and $k := |\mathcal{G}|$. It follows that $\Sigma_{G \in \mathcal{G}} |G| = O(kn)$.

3.1.1.1 Formulating candidate species trees in terms of cluster hierarchies.

Here we formulate constraints that describe all species trees that are possible candidates for solving the t-inconsistency problem, that is, all trees to which the given gene tree set $\mathcal{G}$ is compatible. Based on our assumption that the leaf label function is the identity function, these are all trees with the leaf set X. Our ILP formulation is based on an alternative way of describing trees by specifying their clusters through a hierarchy of subsets of X.

Notation	Definition
$M(i,j)$	Taxon-cluster representation of (the) species tree: $M(i,j) = 1$ iff taxon i is in the cluster j. Additional constraints on M require the cluster set to form a binary hierarchy (tree).
$C(p,q,xy)$	Compatibility: $C(p,q,xy) = 1$ exactly if the cluster pair (p,q) has the gamete $xy \in \{01,10,11\}$.
$T(a,b,c,xyz)$	Rooted triple: $T(a,b,c,xyz) = 1$ exactly if the rooted triple with leaf set $\{a,b,c\}$ and topology xyz is displayed in M. Topologies for xyz are 011, 101, and 110 and refer to the rooted triples $bc\|a$, $ac\|b$ and $ab\|c$ respectively.
$D(g)$	t-inconsistency: $D(g) = 1$ if the gene vertex g is t-inconsistent w.r.t. a tree represented by matrix M.

Table 3.1: Notation used in our ILP solution.

Definition 10 (Clusters)**.** *Let T be a tree. For each vertex $v \in V(T)$ we define the* cluster *at v as $\{u \in L(T) : u \leq_T v\}$, i.e., $L(T(u))$. We shall denote the set of all clusters of T by $\mathcal{H}(T)$.*

Definition 11 ((Full) Binary hierarchy)**.** *Let F be a finite set. We call a set $\mathcal{H}$ of non-empty subsets of F a* (full) binary hierarchy *on F if the following properties are satisfied:*

1. *Trivial set property: $F \in \mathcal{H}$ and $\{v\} \in \mathcal{H}$ for each $v \in F$*
2. *Compatibility property: every pair of sets A and B in $\mathcal{H}$ is* compatible*; that is $A \cap B \in \{A, B, \emptyset\}$.*
3. *Cardinality property: $|\mathcal{H}| = 2|F| - 1$*

A *hierarchy* is defined as a binary hierarchy without requiring the cardinality property. There is a well-known and fundamental equivalence between hierarchies and trees that are not necessarily binary (e.g. [60]). The next result follows from this equivalence and the fact that a binary tree over l leaves has exactly $2l - 1$ cluster.

Theorem 2 (Equivalence between binary hierarchies and binary trees)**.** *Let $\mathcal{H}$ be a set of non-empty subsets of a set F. Then there is a binary tree T such that $\mathcal{H} = \mathcal{H}(F)$ if and only if $\mathcal{H}$ is a binary hierarchy on F.*

Since trees and binary hierarchies are equivalent, we use these terms interchangeably from now on. Now we formulate constraints that describe the hierarchies on X using the binary matrix presentation.

3.1.1.2 Binary matrix.

We describe $2n-1$ subsets of a hierarchy on X using a binary matrix M with a row for each species in X and $2n-1$ columns, where each column p represents the set $\{a \in X \colon M(a,p) = 1\}$.

3.1.1.3 Excluding sets satisfying the trivial set property.

We consider only the $n-2$ non-trivial sets that can be part of a binary hierarchy on X. To do this, we add the following constraints that allow only non-trivial sets. For each column p of M, we require

$$2 \leq \textstyle\sum_{a \in X} M(a,p) \leq (n-1) \ .$$

3.1.1.4 Uniqueness.

To ensure that a set of subsets is uniquely represented by the columns of M, we enforce a linear order of a binary interpretation of these columns. Suppose that $X = \{a_1, \ldots, a_n\}$ are the rows of M, then this order is achieved by adding the following $(n-3)$ constraints that apply to all pairs of adjacent columns p and q in M.

$$\textstyle\sum_{a_i \in X} 2^{i-1} M(a_i,p) \geq \sum_{a_j \in X} 2^{j-1} M(a_j,q) + 1 \ .$$

3.1.1.5 Compatibility.

Incompatibility can be tested directly by using the three-gamete condition (e.g., [33]). An incompatibility occurs for two columns p and q in M if and only if there exist three rows a, b and c in M that contain the *gametes* $(0,1)$, $(1,0)$, and $(1,1)$ in p and q respectively (i.e. $(M(a,p), M(a,q)) = (0,1)$, $(M(b,p), M(b,q)) = (1,0)$, and $(M(c,p), M(c,q)) = (1,1)$). To identify if a certain gamete $(x,y) \in \{(0,1),(1,0),(1,1)\}$ exists for p and q, we define a set of binary variables $C(p,q,xy)$ under the following constraints over all rows a in M.

$$\begin{aligned} C(p,q,01) &\geq -M(a,p) + M(a,q) \,, \\ C(p,q,10) &\geq M(a,p) - M(a,q) \,, \\ C(p,q,11) &\geq M(a,p) + M(a,q) - 1 \,. \end{aligned}$$

These constraints capture that $C(p, q, xy) = 1$ as long as $M(a, p) = x$ and $M(a, q) = y$ is satisfied for a gamete (x, y) in a certain row a in M. However, the reverse condition does not necessarily hold true without adding further constraints. To guarantee that clusters p, q are compatible, we require the following constraints

$$C(p, q, 01) + C(p, q, 11) + C(p, q, 10) = 2 \ .$$

3.1.2 Properties of the Formulation

3.1.2.1 Number of variables and constraints.

There are $O(n^2)$ variables for the matrix M, and $O(n^2)$ variables of the type $C(p, q, xy)$. $O(n)$ constraints are needed to exclude trivial sets and to guarantee uniqueness, and $O(n^3)$ constraints guarantee compatibility. In summary, there are $O(n^2)$ variables and $O(n^3)$ constraints to describe all candidates for the species tree.

3.1.2.2 Formulating the T-Inconsistency problem.

To formulate the t-inconsistency problem, we first describe variables $T(a, b, c, xyz)$ that detect whether a rooted triple is displayed by the tree presented by M. Then we describe variables $D(g)$ that detect if g is t-inconsistent by using the variables $T(a, b, c, xyz)$. Finally, we formulate the objective of the t-inconsistency problem based on the variables $D(g)$.

3.1.2.3 Variables $T(a, b, c, xyz)$.

We describe the binary variables $T(a, b, c, xyz)$ that are 1 exactly if a rooted triple over the leaf set $\{a, b, c\}$ with topology $(x, y, z) \in \{(0, 1, 1), (1, 0, 1), (1, 1, 0)\}$ is displayed by the tree that is presented by M. The parameters a, b, c are rows in M, and the settings 011, 101, and 110 of (x, y, z) refer to the rooted triples $bc|a$, $ac|b$ and $ab|c$ respectively. For each column p in

M, we introduce the following constraints.

$$T(a,b,c,011) \geq -M(a,p) + M(b,p) + M(c,p) - 1 \ ;$$
$$T(a,b,c,101) \geq M(a,p) - M(b,p) + M(c,p) - 1 \ ;$$
$$T(a,b,c,110) \geq M(a,p) + M(b,p) - M(c,p) - 1 \ ;$$
$$T(a,b,c,011) + T(a,b,c,101) + T(a,b,c,110) = 1 \ ,$$

since a rooted triple is uniquely resolved in a tree.

Variables $T(a,b,c,011)$, $T(a,b,c,101)$, and $T(a,b,c,110)$ are constructed for every triple $\{a,b,c\}$ for which a rooted triple is displayed by a gene tree in $\mathcal{G}$. Thus, there are $O(m)$ variables of this type. For each variable we have $O(n)$ constraints, which results in $O(nm)$ constraints overall.

Variables $D(g)$. We express the t-inconsistency of each vertex $g \in V(G)$ where $G \in \mathcal{G}$ by the binary variable $D(g)$. The variable is 1 if g is t-inconsistent with the tree described by matrix M, given the following constraints

$$D(g) \geq 1 - T(a,b,c,xyz) \ ,$$

where the rooted triple over the leaf set $\{a,b,c\}$ and topology xyz is an element in $\mathsf{Trip}_G(g)$.

Variables $D(g)$ are constructed for each internal vertex of a gene tree in $\mathcal{G}$, which results in $O(kn)$ variables. Intuitively, a constraint is constructed for each rooted triple that is displayed by a gene tree in $\mathcal{G}$, which yields $O(km)$ constraints. However, the following observation reduces the number of such constraints to $O(kn^2)$.

Let $u \in V(G)$ such that $\mathsf{Trip}_G(u) \neq \emptyset$, $\{v,w\} = \mathsf{Ch}(u)$, $a \in L(G(u))$ and $b \in L(G(v))$. A rooted triple $xy|z$ is in $\mathsf{Trip}_G(u)$ if and only if all $ax|b$, $ay|b$, and $bz|a$ are in $\mathsf{Trip}_G(u)$. Therefore, instead of enumerating all rooted triples in $\mathsf{Trip}_G(u)$ (which sums up to $O(m)$ in each gene tree G), we only need to enumerate a number of $O(n)$ rooted triples to represent $\mathsf{Trip}_G(u)$ while detecting if u is t-inconsistent (hence $O(kn^2)$ constraints over all).

3.1.2.4 T-Inconsistency objective.

This objective is expressed by the following expression.

$$\min \sum_{g \in V(\mathcal{G})} D(g) \ .$$

Once the optimal objective cost is found, a unique tree corresponding to the cost can be constructed from M. It is worth noting that an instance of unique optimal tree does not ensure an unique optimal solution to the corresponding ILP due to relaxed constraints for variables C. Although this can be addressed by adding additional constraints, the correctness of the objective value and the resulting tree is not affected.

3.1.2.5 Number of variables and constraints.

In summary, there are $O(n^2 + m + kn)$ variables, and the number of constraints is $O(n^3 + mn + kn^2)$.

Handling non-injective leaf mappings. A leaf mapping $\mathcal{L}_{G,S}$ is non-injective if and only if there is a vertex $u \in V(G)$ with distinct children v and w such that $\mathcal{L}_{G,S}(L(G(v))) \cap \mathcal{L}_{G,S}(L(G(w))) \neq \emptyset$; and if the latter holds true, it follows that u is a duplication. Therefore, it can be determined if u is a gene-duplication regardless of the topology of S. By pre-processing all such determined duplication vertices, the leaf-mapping over the remaining internal vertices of G can be made injective. Hence, the existing ILP formulation solves input gene trees with non-injective leaf mappings. Since the input gene tree size can be arbitrary, under the non-injective leaf mapping assumption, the ILP formulation has $O(n^2+m+l)$ variables and $O(n^3+mn+ln)$ constraints where $\Sigma_{G \in \mathcal{G}}|G| = l$.

Generating optimal species trees. The species tree corresponding to a feasible solution of an ILP instance can be constructed in $O(n^2)$ time [32]. Furthermore, a gene node g is identified as a duplication if and only if $D(g) = 1$.

3.2 Experiments and Results

Implementation. We implemented an ILP generator in Python that, given a set of gene trees, outputs the ILP described in the preceding section. We tested our formulation with both simulated and empirical gene tree data sets (described below). All analyses were on a GNU/Linux based PC with an Intel Core2 Quad 2.4 GHz CPU. We choose Gurobi 2.0 [31] to solve the ILP directly and CPLEX 12.1 [39] to enumerate optimal solutions when necessary.

Simulation experiments. We first evaluated the performance of our ILP solution with simulated gene tree data sets. Our simulation protocol included the following steps: (1) a species tree S of n taxa was randomly generated as the template of a gene tree; (2) a depth-first-search walk starting from $\mathsf{Rt}(S)$ simulated at most one evolutionary event at each vertex based on given probabilities for each event. These events could be a duplication (duplicating the whole current subtree) or a loss (cutting the current subtree). If there is neither a duplication nor a loss, the process proceeds to the next vertex. We used the same species tree to generate k gene trees.

In our simulation experiments, we used a duplication rate of 0.25 duplications per gene at each speciation vertex and a loss rate of 0.3. These events produced a similar tree size distribution and optimal duplication cost to the gene trees used by Sanderson and McMahon [59]. We varied the number of taxa in the species tree from 6 to 14 and the number of input gene trees from 10 to 1000. We performed 10 simulation replicates for each different combination of species and gene tree number. For each simulated data set, we also compared the ILP score to results from DupTree [64], a fast hill-climbing heuristic implementation for the problem, to determine if the heuristic finds the optimal solution.

Seed plant analysis. Next, we tested the ability of the ILP formulation to solve the seed plant phylogeny problem using a large-scale genomic data set. First, to build the gene trees, amino acid alignments for gene familieswere selected from Phytome v. 2, an online comparative genomics database based on publicly available sequence data from 136 plant species [35]. To ensure positional homology throughout the alignments, columns and sequences of questionable

certainty were masked using default settings of the program REAP [35, 36].We sampled sequences from the nine gymnosperm taxa represented in Phytome with the most data, including cycad taxon *Cycas rumphii*, Gnetales taxa *Gnetum gnemon* and *Welwitschia mirabilis*, and, from the conifers, *Cryptomeria japonica* from Cupressaceae, and *Pseudotsuga menziesii*, *Picea glauca*, *Picea sitchensis*, *Pinus pinaster*, and *Pinus taeda* from Pinaceae. We also sampled sequences from two representative angiosperm taxa, *Arabidopsis thaliana* and *Oryza sativa*, and the non-seed plant, *Physcomitrella patens*.

We selected all the 6,084 masked amino acid alignments from gene families in Phytome that had at least 4 sequences and had sequences from at least 3 of the selected taxa. All species were found in at least 376 gene families. To build the gene trees, we performed ML phylogenetic analyses on each of the gene alignments using RAxML-VI-HPC version 2.2.3 [62]. The ML analyses used the JTT amino acid substitution model [40] with rate variation among sites (the "PROTMIX" model; see [62]). The trees were then rooting using mid-point rooting, as implemented in the Phylip program retree [25]. We applied the ILP formulation to solve the GD problem using all 6,084 gene trees.

	n = 6		n = 8		n = 10		n = 12		n = 14	
k	time	Dup	time	Dup	time	Dup	time	Dup	time	Dup
10	0.06	34.80	0.34	49.70	22.98	60.10	200.53	68.80	12597.21	78.40
50	0.03	189.50	1.26	265.00	8.74	280.00	159.26	346.40	2953.62	393.10
100	0.06	382.80	0.63	523.30	9.64	598.50	117.38	701.60	2191.65	825.70
200	0.05	788.20	0.54	994.90	11.03	1217.30	168.85	1372.50	2709.91	1627.70
500	0.25	1910.30	0.79	2458.60	13.92	2987.00	220.17	3678.80	4270.05	4001.70
1000	0.57	3842.60	0.96	5283.10	23.54	6140.90	330.34	7026.40	5014.61	8258.80

Table 3.2: Our ILP running time and the optimal duplication cost solved in Gurobi using simulated k gene trees of n taxa as inputs. The simulated gene trees are generated under 0.25 duplication rate and 0.3 loss rate. At each configuration, the result is the average of 10 trials. The running time is measured in seconds.

3.3 A Casestudy in Seed Plant

The relationships among the major lineages of seed plants has long been a major question in plant systematics, especially with regard to the position of Gnetales, a clade of three genera

(Gnetum, Ephedra, and Welwitschia) that lack obvious morphological links to other extant seed plants (e.g.,, [6, 18, 48, 50, 61]). Cladistic analyses of morphological characters generally have placed Gnetales sister to the angiosperms, or flowering plants [15, 19, 18, 20, 37, 52]; however, early analyses of molecular characters rarely supported this placement [6, 48, 58, 61]. Most recently, maximum likelihood (ML) and maximum parsimony (MP) analysis of 15-17 plastid loci placed Gnetales sister to the other seed plants [57]. However, a loss of plastid *ndh* genes appears to link Gnetales with Pinaceae [4]. An MP analysis of EST sequences from 43 nuclear genes similarly linked Gnetales with the conifers [16]. Yet later MP and ML analyses of EST sequences from over $1,200$ nuclear loci placed Gnetales sister to the other gymnosperms [17]. All of these molecular analyses of the seed plant phylogeny have been limited to putatively orthologous genes. However, the GD problem provides a way to incorporate large gene families into the phylogenetic inference of seed plants.

Our implementation of the ILP formulation finished running the data set in approximately two minutes. We identified a unique optimal solution with $47,658$ duplications (Figure 3.1).In the optimal species tree, the seed plants are split into angiosperm and gymnosperm clades (Figure 3.1). In the gymnosperm clade, Gnetales are sister to a conifer clade. With $6,084$ genes, this GTP analysis of seed plants includes by far the most genes ever used to infer the seed plant phylogeny. Our GTP analysis of this large, previously underutilized, data source provides a novel line of evidence that angiosperms and all extant gymnosperms are sister clades. Like most ML analyses of multi-locus data sets, our results show a close affinity between Gnetales and conifers (e.g., [6, 7, 48, 61, 66]); however, unlike many of these analyses, GTP does not place Gnetales sister to Pinaceae. Due to the necessarily limited taxon sampling, especially among non-Pinaceae conifers, our results regarding the placement of Gnetales are neither precise nor definitive. Still, the placement of Gnetales sister to the conifers, is an intriguing result that is consistent with some morphological characters, such as ovulate cone scales and resin canals, which appear to support conifer monophyly [18]. However, in contrast to our result, the deletion of the *ndh* genes in Gnetales and Pinaceae suggests that these clades are sister. Although the GTP results are intriguing, they should be interpreted with caution. For example, the results

do not provide any measures of confidence or suggest the degree to which alternate phylogenetic hypotheses are suboptimal. Furthermore, the gene trees were rooted using mid-point rooting, which may produce incorrect rootings when the sequences do not evolve at a constant rate of evolution [38]. Also, the taxon sampling in this analysis is limited, and the seed plant phylogeny problem can be sensitive to taxon sampling [58]. Thus, although our result provides a novel large-scale genomic perspective on the seed plant phylogeny, it is not a definitive.

3.4 Conclusions

Our ILP formulation provides exact solutions to the largest instances of the GD problem analyzed to date. Thus, it can provide a large-scale genomic perspective on important phylogenetic questions that previously could only be addressed by heuristics. Furthermore, our simulation experiments demonstrate that these heuristic estimates can be misled with as few as 10 taxa. Even when heuristics identify an optimal solution they cannot, unlike ILP, determine if the solution is unique. In future research the ILP implementation will be useful, not only for solving empirical data sets, but for assessing the performance of different heuristics by comparing their estimates to the exact ILP solution. Ultimately, it also will be useful to expand the scale of solvable instances beyond 14 taxa. While this challenge may be addressed by improved ILP formulations, investigations into other algorithm concepts might also be effective (e.g., [21, 63]).

CHAPTER 4. SOLVING GTP USING DYNAMIC PROGRAMMING

The second approach is a recursive solution in dynamic programming (DP). One attractive property of this solution is the fundamental structure discovered. The design, implementation and analysis of this DP solution is the result of a collaboration with André Wehe, Dr. Paweł Górecki, and Dr. Oliver Eulenstein.

4.1 Our Contribution

We introduce dynamic programming algorithms that use bit-vector data structure and Gray encoding to solve the GTP problems exactly. Using bit-vector and Gray encoding allows us to efficiently utilize large registers of modern processors. As a result our algorithms have complexities in $O(3^n mn/b)$ time and $O(2^n + m)$ space, where n is the overall number of taxa presented in input trees, m is the number of unique rooted splits in the input gene trees, and b is the size of the bit-vector used. To the best of our knowledge the largest species trees that were exactly computed for non-trivial GTP instances had 8 taxa for the deep coalescence cost function and 14 taxa for the gene duplication events. We demonstrate in a simulation study that our algorithm can compute species trees with up to 22 taxa for each of the GTP problems in less then 17 hours on a standard workstation. These are by far the largest exact solutions that have been computed for GTP problems. Using our exact algorithms we showed that species trees inferred by the GTP heuristic implemented in DupTree [64] are largely misled when using instances with at least 16 taxa. Our empirical study with an instance consisting of a collection of gene family trees from 16 taxa showed that DupTree reported optimal species trees in only 8% of one thousand runs. This makes our exact algorithms a viable alternative to DupTree for instances with smaller taxa numbers. Furthermore, our algorithms also report

the number of all optimal solutions, which can be beneficial in phylogenetic analyzes. As an example, our experiments confirmed that biologists have already identified all optimal solutions for the collection of gene trees from our empirical study.

4.2 Preliminaries

A *gene tree* is a rooted binary tree (DAG) whose leaves are labeled by species names. Let T be a gene tree. By $L(T)$ we denote the set of all leaf labels present in T. A node n is called *internal* if it has two children, which are denoted by $\mathsf{Ch}(n)$. A cluster of a node g is the set of leaf labels visible from g. A *species tree* is a gene tree whose leaves are uniquely labeled.

Let $S = \langle V_S, E_S \rangle$ be a species tree. For nodes $a, b \in V_S$, let $a + b$ be the least common ancestor of a and b in S. We also use the binary order relation $a \leq b$ if b is a node on the path between a and the root of S. Two nodes a and b are called *siblings* if they are children of $a + b$. Let $A \subseteq L(S)$. Let $S|_A$ be the species tree inferred from S by the following operations: (1) remove leaves from S that are labeled by species from $L(S) \setminus A$, and (2) contract all nodes of degree 2 except for the root.

For a gene tree G and a species tree S such that $L(G) \subseteq L(S)$, a *least common ancestor mapping*, or lca-mapping, is a function from the nodes of G to the nodes of S such that $M(g) = s$ if g and s are leaves with the same label, or $M(g) = M(a) + M(b)$ if g is an internal node of G such that a and b are the children of g.

Now, we introduce several cost functions used when reconciling a gene tree G and a species tree S. An internal node $g \in V_G$ is a *gene duplication* if $M(g) = M(g')$ for g' a child of g. The total number of gene duplications is called *duplication cost* and denoted by $D(G, S)$. We define the *deep coalescence* cost function as follows: $DC(G, S) = \sum_{a,b \text{ siblings in } G} |\pi(M(a), M(b)) - 1|$, where $\pi(x, y)$ is the set of all nodes on the shortest path connecting x and y in S. Note, the standard definition of the *deep coalescence* cost function [2] differs by $1 - |V_G|$ from ours, which is a constant value dependent on G. Next, we define *the loss cost [68]:* $L(G, S) = DC(G, S) + 2 \times D(G, S) - |V_G| + 1$ and *the duplication-loss cost*: $DL(G, S) = D(G, S) + L(G, S)$. For a given gene tree G and a given species tree S all costs mentioned above can be computed in linear time and space [44].

In this paper, we need a more general construction that will precisely assign partial costs to a node of a species tree. Let us assume that we are given *a contribution function* $\xi\colon V_G \times V_S \to R$, which for a node $g \in V_G$ and $s \in V_S$ assigns a real $\xi(g,s)$ representing a cost contribution of a node g to a node s when reconciling G and S. Then, the cost of reconciling a gene tree G and a species S will be represented in a general form $\sigma(G,S) = \sum_{g\in G, s\in S} \xi(g,s)$. Let us define the cost contribution functions for our standard cost functions. Let $\mathbf{1}$ be the indicator function, that is, $\mathbf{1}[P] = 1$ if P is satisfied, and $\mathbf{1}[P] = 0$ otherwise. Then,

$$\xi^D(g,s) = \mathbf{1}[\exists g' \in \mathsf{Ch}(g) \text{ and } M(g) = s = M(g')], \tag{4.1}$$

$$\xi^L(g,s) = \mathbf{1}[\{g',g''\} = \mathsf{Ch}(g) \text{ and } s \in \mathrm{In}(M(g'),M(g''))], \tag{4.2}$$

$$\xi^{DC}(g,s) = \mathbf{1}[\{g',g''\} = \mathsf{Ch}(g) \text{ and } M(g) \neq s \in \pi(M(g'),M(g''))], \tag{4.3}$$

$$\xi^{DL}(g,s) = \xi^D(g,s) + \xi^L(g,s), \tag{4.4}$$

where $\mathrm{In}(v,w)$ is the set of loss nodes between v and w defined as follows. If $v \leq w$, then $\mathrm{In}(v,w) = \pi(v,w) \setminus \{v\}$. If $w \leq v$, then $\mathrm{In}(v,w) = \pi(v,w) \setminus \{w\}$. Finally, $\mathrm{In}(v,w) = \pi(v,w) \setminus \{v,w,v+w\}$ if v and w are incomparable. In this general setting, one can prove that: $D(G,S) = \sigma^D(G,S)$, $L(G,S) = \sigma^L(G,S)$ and so on [29].

We say that a species tree S is over a set of species I if $L(S) = I$. Let Q be a collection of gene trees $G_1, G_2, \ldots, G_n$. By $L(Q)$ we denote the set of all species present in trees of Q. Now, in a natural way we extend the notion of the cost function to collections of gene trees. For a given species tree S over $L(Q)$, by $c(Q,S)$ we denote the total cost $\sum_{G\in Q} c(G,S)$.

Finally, we can define the following problem.

Problem 5. *[OST - Optimal Species Tree] Given a collection of gene trees Q and a cost function c find a species tree S^{opt} that minimizes the total cost $c(Q,S)$ in the set of all species trees S over $L(Q)$.*

The species tree S^{opt} will be called optimal (for Q and c) and the optimal cost we denote by $c^{opt}(Q)$. We can also define a simpler variant of the previous problem.

Problem 6. *[OCC - Optimal Cost Computation] Given a collection of gene trees Q and a cost function c compute the optimal cost $c^{opt}(Q)$.*

The optimization variant of the OCC problem under the duplication cost is NP-hard [45].

4.3 Solving GTP using Dynamic Programming

In this section we show a dynamic programming solution for the OCC problem. Next, we provide an extension to the algorithm that solves the OST problem.

Let G be a gene tree. Any internal node g of G determines a multiset $\{A, B\}$, where A and B are the clusters of children of g. Such a multiset will be called *a rooted split* and denoted by $A|B$. For a collection Q of gene trees let $r(Q)$ be the multiset of all rooted splits present in G.

4.3.1 Dynamic Programming Formula

Let Q be a collection of gene trees, c be a cost function and $Z \subseteq L(Q)$ The dynamic programming formula Δ^c for the OCC problem is defined as follows (the superscript c is omitted in Δ, Γ and Λ):

$$\Delta(Q,Z) = \begin{cases} \Lambda(Q,s) & \text{if } Z = \{s\}, \\ \min\limits_{\substack{Z=X\cup Y \\ X\cap Y=\emptyset \\ X,Y\neq\emptyset}} (\Delta(Q,X) + \Delta(Q,Y) + \Gamma(Q,X,Y)) & \text{otherwise,} \end{cases} \tag{4.5}$$

where $\Lambda(Q,s)$ is the total cost contribution of the nodes from Q to a leaf of some species tree over $L(G)$ labeled by s, and $\Gamma(Q,X,Y)$ is the total cost contribution of the nodes from Q to an internal node v of some species tree over $L(G)$ such that the cluster of v is $X \cup Y$ and v has two children whose clusters are X and Y. Please observe, that under our standard cost functions these definitions do not depend on the choice of the species tree.

The total cost contribution functions for our standard costs are defined as follows

$$\begin{aligned} \Lambda(Q,s) &= \textstyle\sum_{q\in r(Q)} \lambda(q,s) \\ \Gamma(Q,X,Y) &= \textstyle\sum_{q\in r(Q)} \gamma(q,X,Y), \end{aligned}$$

where $\lambda^D(A|B,s) = \mathbf{1}[A \cup B = \{s\}]$, $\lambda^L(A|B,s) = 0$, $\lambda^{DL} = \lambda^D$, $\lambda^{DC}(A|B,s) = 0$ and

$$\gamma^D(A_1|A_2,X_1,X_2) = \mathbf{1}[A_1 \cup A_2 \subseteq X_1 \cup X_2 \wedge (\exists i\colon \neg A_i \subseteq X_1 \wedge \neg A_i \subseteq X_2)] \quad (4.6)$$

$$\gamma^L(A_1|A_2,X_1,X_2) = \mathbf{1}[\exists i,j\colon A_i \subseteq X_j \wedge (A_{i+1} \subseteq X_1 \cup X_2 \Rightarrow A_{i+1} \cap X_1 \neq \emptyset \neq A_{i+1} \cap X_2)] \quad (4.7)$$

$$\gamma^{DC}(A_1|A_2,X_1,X_2) = \mathbf{1}[\exists i,j\colon (\neg A_i \subseteq X_j \Rightarrow A_i \subseteq X_1 \cup X_2) \wedge \neg A_{i+1} \subseteq X_1 \cup X_2] \quad (4.8)$$

$$\gamma^{DL}(q,X_1,X_2) = \gamma^D(q,X_1,X_2) + \gamma^L(q,X_1,X_2). \quad (4.9)$$

For simplicity, in the above equations $i+1$ (and similarly $j+1$) is 1 if $i = 2$. The following lemma establishes relation between a contribution function and the total cost contribution function.

Lemma 1. *Let G be a gene tree, c be a cost function and S be a species tree over $L(G)$. Then,*

- for each leaf v in S labeled by a species s:

$$\Lambda^c(\{G\},s) = \textstyle\sum_{g \in V_G} \xi^c(g,v).$$

- for each internal node v in S whose children have clusters X and Y:

$$\Gamma^c(\{G\},X,Y) = \textstyle\sum_{g \in V_G} \xi^c(g,v).$$

Proof. The proof follows easily for each cost function: D, L, DC and DL and the definitions of λ and γ. □

Theorem 3. *Let Q be a collection of gene trees and c be a cost function. Then*

$$\Delta^c(Q, L(Q)) = c^{opt}(Q).$$

Proof. It is sufficient to prove that for each $Z \subseteq L(Q)$,

$$\Delta^c(Q,Z) = \min_{S \in Tr(Z)} \sum_{\substack{G \in Q, g \in G \\ v \in V_S, c_v \subseteq Z}} \xi^c(g,v),$$

where $Tr(Z)$ is the set of all species trees over $L(Q)$ having a node whose cluster is Z. The proof follows by induction from Lemma 1. We omit technical details. □

Solving the OST problem is straightforward from the values of Δ: it is sufficient to track which partitions of the cluster Z into X and Y in formula (4.5) yield the minimal value. Such partitions determine clusters in optimal species trees and they can be used to infer one or all optimal species trees.

Similar to solving the OST problem using Δ, our formula can be extended to count the number of all optimal species trees. In addition to track which partition of a cluster Z yields the minimal value, we keep track of how many partitions yield such value. Let $\Omega(Q, Z)$ denote the count. Within one partition (X, Y) of Z, $\Omega(Q, X) \times \Omega(Q, Y)$ is the count of all optimal trees under (X, Y), and $\Omega(Q, Z)$ is the sum of $\Omega(Q, X) \times \Omega(Q, Y)$ over all optimal partitions under Z. Due to the recursive nature of Δ, such count equals the number of different species trees whose set of leaves is Z.

4.3.2 Algorithm and its implementation

In this section we describe in more details the algorithm and its implementation. Let Q be a collection of input gene trees. First, the gene trees present in Q are converted into the multiset of rooted splits $r(Q)$, that is, each split consists of two clusters and its multiplicity. The algorithm that solves the OST problem is based on the formula (4.5). It can be efficiently implemented by using bit-vector representation of sets. In other words, each cluster, that is, a set of species from $L(Q)$, is represented as a bit-vector of the size n, where n is the size of $L(Q)$. Then, each bit-vector operation (see formulas (4.6)-(4.9)) requires $O(n/b)$ time, where b is the word size of the platform.

Our algorithm creates the array of 2^n values of Δ. To compute a single $\Delta(Q, Z)$ value, we need to generate all splits of Z. Each split of Z can be computed in constant time by using Gray code. Then, we have $\sum_{2 \leq k \leq n} \binom{n}{k} 2^k = O(3^n)$ splits of the elements from the powerset of $L(Q)$, therefore the time complexity of the optimal cost computation is $O(3^n mn/b)$, where m is the number of unique rooted splits in $r(Q)$. Clearly, the space complexity is $O(2^n + m)$.

As already mentioned in the previous section, to solve the OST problem, it is sufficient to store exactly one rooted split that yields the minimal score in 4.5. In such a case the algorithm requires additional $O(2^n)$ units of space (that is, one optimal split per each value

in the Δ array). Clearly, the time complexity is the same. To obtain the total number of optimal solutions, we need an additional counter in each element of the Δ array. Updating the counter can be completed in constant time when computing the minimal score in the dynamic programming formula. Hence under the same complexity, we also find out the total number of optimal solutions.

In practice, gene trees have less species than present in whole input collection. The standard solution is to apply *the minus method* [14], where the cost is computed for the species tree contracted to the set of species present in a given gene tree. Formally, for a given cost c, a species tree S and a gene tree G such that $L(G) \subseteq L(S)$, instead of computing $c(S, G)$ we compute $c(G, S|_{L(G)})$. If c is the duplication cost, then there is no difference in the dynamic programming algorithm. For the other costs, we need the following changes (easy proofs are omitted):

- the data structure $r(Q)$ is the multiset of triplets: $\{A|B|L(G)\colon\ G \in Q,\ A|B$ is a rooted split from $G\}$; in other words each rooted split is extended with the set of species present in its source gene tree.

- the formula (4.7) for gene losses is replaced by:

$$\begin{aligned}\gamma^L(A_1|A_2|C, X_1, X_2) \quad = \quad & \mathbf{1}[\exists i, j\colon A_i \subseteq X_j \wedge \\ & (A_{i+1} \subseteq X_1 \cup X_2 \Rightarrow A_{i+1} \cap X_1 \neq \emptyset \neq A_{i+1} \cap X_2) \wedge \\ & X_{j+1} \cap C \neq \emptyset],\end{aligned}$$

- the formula (4.8) for deep coalescence is replaced by:

$$\begin{aligned}\gamma^{DC}(A_1|A_2|C, X_1, X_2) \quad = \quad & \mathbf{1}[\exists i, j\colon (\neg A_i \subseteq X_j \Rightarrow A_i \subseteq X_1 \cup X_2) \wedge \\ & \neg A_{i+1} \subseteq X_1 \cup X_2 \wedge X_{j+1} \cap C \neq \emptyset].\end{aligned}$$

The time and space complexity remain the same, however, the size of $r(Q)$ (the parameter m) can increase when the minus method is applied. Therefore, the run-time is usually longer for

the DC and DL costs than for the D cost as observed in our simulation study. See Fig. 4.1 for comparison.

4.4 Experiments

We performed a total of 70 runs on different simulated data sets for 10, 12, 14, 16, 18, 20, and 22 taxa (10 runs each). Each instance was generated with 100 gene trees. The run-times of our software for gene duplications, duplications plus losses, and deep coalescence are depicted in Fig. 4.1, While instances were generated only based on a gene duplication plus loss model, run times deep coalescence as the cost function of GTP were included for comparison purposes. We observed that data sets with up to 16 taxa can be computed less than 1 minute. Our program found optimal species trees for data sets with up to 22 taxa in less than 17 hours for the GTP problems based on gene duplication, gene duplication plus loss. Further, we analyzed the number of optimal solutions, and found that each of these generated instances have a unique solution.

We generated instances for the GTP problem based on deep coalescence using Mesquite [47] as outlined in the simulation protocol from Maddison and Knowles [46]. Our effective population size was 10,000. Similarly to the previous simulation we simulated data sets for 10, 12, 14, 16, 18, 20, and 22 taxa. The runtime results of our software spent approximitely 3 times longer solving instances from the deep coalescence model than that of the duplication plus loss model. We found that the number of rooted splits differ by a factor of 2.8 to 4.1 between instances that were generated by the two different models of the same taxon size. The deep coalescence instances were also reported to have up to 4 optimal species trees.

4.4.1 Empirical Study

We re-evaluated the data set from Guigó et al. [30] of 53 gene families from the 16 eukaryotes. This data set has been well studied by Page and Charleston [55], who identified 12 optimal trees for the GTP problem based on gene duplication plus loss events. Using our software, we were able to verify, for the first time, that Page and Charleston identified all optimal trees. Our software finished the analysis of the data set from Page and Charleston in 42 seconds.

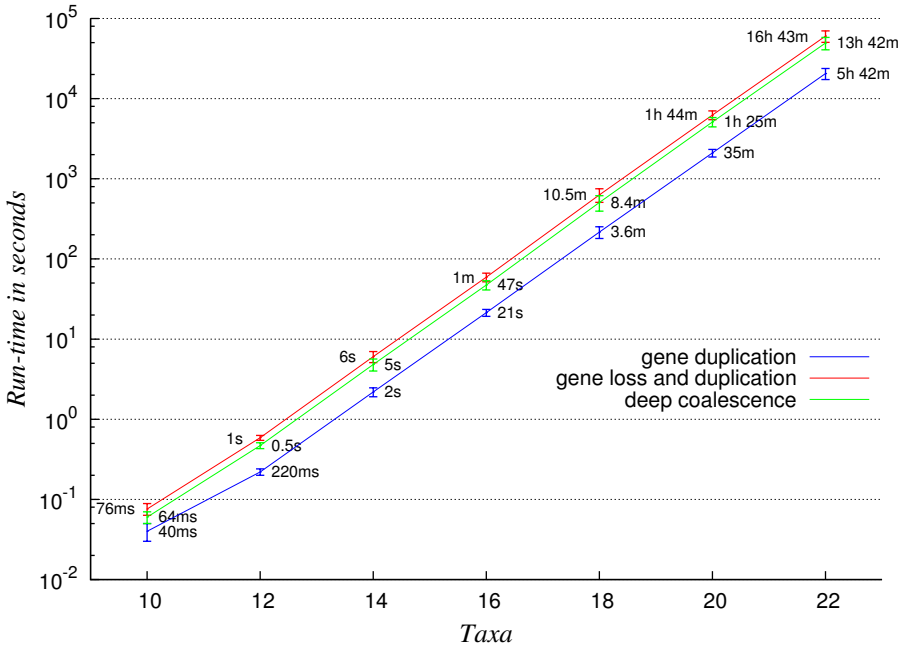

Figure 4.1: Summary of experiments performed on simulated data sets under standard evolutionary costs. The run-time means and standard deviations are summarized on the logarithmic scale for our standard cost functions: gene duplications (blue curve), gene duplications and losses (red curve) and deep coalescence (green curve). Observe that the data sets with less than 16 taxa were resolved in less than 1 minute, while the largest data sets with 22 taxa required up to 17 hours of computations.

Furthermore, for the same data set we computed the number of optimal solutions for the GTP problem based on gene duplications, and found that there are 29 additional optimal trees. This computation took our software only 14 seconds to complete.

Having already computed the exact result for the Guigó et al. data set, we used this data set to evaluated the accuracy of the heuristic DupTree [64]. This type of heuristic aims to solve the GTP problem for gene duplications and appears to have produced several credible results [54, 12, 49, 51]. We run the heuristic thousand times on the Guigó et al. data set. DupTree found optimal species trees in only 8% of the runs we performed.

4.5 Conclusion

Our fast bit-vector and Gray encoding algorithms provided the by far largest GTP species trees from non-trivial instances to date. Since the exact algorithms can compute species trees

with up to 22 taxa in reasonable time, they can provide new perspectives on phylogenetic questions that previously could only be addressed by heuristics. This is of particular importance, since our empirical study showed that heuristic estimates can be misled in the vast majority of runs on the same data-set with as few as 16 taxa. In difference to heuristics our algorithms report the number of optimal solutions, which can benefit phylogenetic analyzes. Future research will focus on expanding the scale of solvable instances beyond 22 taxa, and to assess the performance of other GTP heuristics by comparing their estimates to the exact results provided by our algorithms.

CHAPTER 5. DATA MINING USING QUASI-BICLIQUE

5.1 Introduction

As introduced in the earlier section, we utilize quasi-bicliques (QBC) as an input data selection/mining technique to ensure the coherence of the input data. To improve upon previous and related quasi-biclique work, we introduce two parameters that control the lower-bound of missing information allowed on either side of the resulting bi-partition vertex sets.

As a newly proposed problem, our QBC study includes both complexity and a solution in ILP. We are currently investigating improvements to our solution and other applications of QBC besides assisting GTP analysis. Collaborators in the QBC problem include Sudheer Vakati, Dr. Roland Krause, and Dr. Oliver Eulenstein.

5.2 Motivation and Related Work

The binary relation between the gene trees (which are generally gene families) and the representing species taxa has been studied using biclique enumeration [22]. While extracting a dense component in the gene-species bi-partite graph ensures the overlapping coverage of taxa among the input gene trees, there is inevitably missing information among the inputs, limiting the sizes of bicliques. Thus, extracting QBCs which tolerate a few missing edges is the evolutionary step of the concept [67, 42]. In the previous studies of QBCs, it is typically assumed that the relations between gene trees and covering taxa are only binary, i.e., either it is included in the input or it is not. However, individual gene trees can have different weights among the input to represent the quality [5], and the significance of taxa can vary. It is our observation that incorporating such non-binary information can be helpful to the selection process. Our weighted quasi-biclique is therefore the latest evolution in QBC study that includes weights of

the gene-species relation.

5.3 Maximum QBC problem

As introduced earlier, we have different creteria to define an optimal α,β-WQB. Unfurtunately, these variations are all computationally difficult as we describe here.

Lemma 2. *The α,β-WQB$_{P(C)}$ problem is NP-hard.*

We now prove that checking for the existence of a percentage α,β-WQB in a bipartite graph is NP-complete. Note that, checking the existence of a constant version α,β-WQB in a bipartite graph can be done in polynomial time. For rest of the section we only refer to percentage version α,β-WQB's.

Problem 7 (Existence)**.**

Instance: A weighted bipartite graph $G := (U + V, E, \omega)$, values $\alpha, \beta \in [0, 1]$.

Find: If there exists a α,β-WQB$_P$ (U', V') in G.

To prove the hardness of existence problem we need some auxiliary definitions. A *modified weighted bipartite graph*, denoted by $(U + V, E, \Omega)$, is a complete bipartite graph $(U + V, E)$ with a weight function $\Omega : E \to [0, 1]$ where, for any two edges e and e', $|\Omega(e) - \Omega(e')| \leq 1$.

Definition 12 (Modified α,β-WQB (MO-WQB))**.**

Let $G := (U + V, E, \Omega)$ be a modified weighted bipartite graph. A non-empty pair (U', V') included in (U, V) is a MO-WQB of G, if it satisfies the three properties: (1) (U', V') includes $(\emptyset, V)$, (2) $\forall u \in U'$: $\sum_{v \in V'} w(u, v) \geq 0$, and (3) $\forall v \in V'$: $\sum_{u \in U'} w(u, v) \geq 0$.

Problem 8 (One sided existence)**.**

Instance: A weighted bipartite graph $G := (U + V, E, \omega)$, values $\alpha, \beta \in [0, 1]$.

Find: If there exists a α,β-WQB$_P$ (U', V') in G which includes the pair $(\emptyset, V)$.

Problem 9 (Modified Existence)**.**

Instance: A modified weighted bipartite graph $G := (U + V, E, \Omega)$.

Find: If there exists a MO-WQB *in G.*

The outline of the proof starts first by reducing the *partition* problem, which is NP-complete [27], to the *modified existence* problem. The *modified existence* problem is then reduced to the one sided existence problem. The one sided existence problem reduces to the existence problem. The detail of the proofs can be found here [9].

Lemma 3. *The modified existence problem is NP-complete.*

Lemma 4. *The* one sided existence *problem is NP-complete.*

Lemma 5. Existence *problem is NP-complete.*

5.4 IP Formulations for the α,β-WQB Problem

Although greedy approaches are often used in problems of a similar structure, e.g., multi-dimensional knapsack [41], δ-QB [43], in our experiments, both greedy and randomized approach did not identify solutions close enough to the exact solutions. In our experiments, simple greedy and randomized solutions yielded accuracies ranging from 60% to 95% depending on various parameters without performance guarantee. Hence we consider that it is rather important here to find exact solutions in order to demonstrate the usefulness of α,β-WQB's. Here we present integer programming (IP) formulations solving the α,β-WQB problem in exact solutions.

Due to the similarity in formulating constraints between α,β-WQB$_C$ and α,β-WQB$_P$, we start by formulating a solution to α,β-WQB$_P$. Our initial IP requires quadratic constraints, which are then replaced by linear constraints such that it can be solved by various optimization software packages. Our final formulation is further improved by adopting the implication rule to simplify variables involved. This improved formulation requires variables and constraints linear to the number of input edges, and thus, suits better for sparse graphs. Throughout the section, unless stated otherwise, $G := (U + V, E, \omega)$ represents a weighted bipartite graph, and $G' = (U', V')$ represents the maximum weighted α,β-WQB of G and E' represents the edges induced by G' in G.

5.4.1 Quadratic Programming

For each $u \in U$ ($v \in V$), a binary variable x_u (x_v) is introduced. The variable x_u (x_v) is 1 if and only if vertex u (v) is in U' (V'). The integer program to find the solution G' can be formulated as follows.

$$\begin{aligned}
\text{Binary variables:} \quad & x_u, \text{ s.t. } x_u = 1 \text{ iff } u \in U' && \text{for each } u \in U && (5.1)\\
& x_v, \text{ s.t. } x_v = 1 \text{ iff } v \in V' && \text{for each } v \in V && (5.2)\\
\text{Subject to:} \quad & \textstyle\sum_{v \in V} \omega(u,v) \cdot x_v x_u \geq \alpha \sum_{v \in V} x_v x_u && \text{for all } u \in U && (5.3)\\
& \textstyle\sum_{u \in U} \omega(u,v) \cdot x_u x_v \geq \beta \sum_{u \in U} x_u x_v && \text{for all } v \in V && (5.4)\\
\text{Maximize:} \quad & \textstyle\sum_{(u,v) \in U \times V} x_u x_v \cdot \omega(u,v) && && (5.5)
\end{aligned}$$

The quadratic terms in the constraints are necessary because, α and β thresholds apply only to vertices in U' and V'. This formulation uses variables and constraints linear to the size of input vertices, i.e., $O(|U| + |V|)$. Since solving a quadratic program usually requires a proprietary solver, we reformulate the program so that all expressions are linear.

5.4.2 Converted Linear Programming

A standard approach to convert a quadratic program to a linear one is introducing auxiliary variables to replace the quadratic terms. Here we introduce a binary variable y_{uv} for every edge (u,v) in G, such that, $y_{uv} = 1$ if and only if $x_u = x_v = 1$, i.e., the edge (u,v) is in G'. The linear program to find the solution G' is formulated as follows.

$$\begin{aligned}
\text{Binary variables:} \quad & \text{Same as in (5.1) and (5.2)} && && \\
& y_{uv}, \text{ s.t., } y_{uv} = 1 \text{ iff } x_u = x_v = 1 && \text{for all } (u,v) \in E && (5.6)\\
\text{Subject to:} \quad & y_{uv} \leq (x_u + x_v)/2 && \text{for all } (u,v) \in E && (5.7)\\
& y_{uv} \geq x_u + x_v - 1 && \text{for all } (u,v) \in E && (5.8)\\
& \textstyle\sum_{v \in V} \omega(u,v) \cdot y_{uv} \geq \alpha \sum_{v \in V} y_{uv} && \text{for all } u \in U && (5.9)\\
& \textstyle\sum_{u \in U} \omega(u,v) \cdot y_{uv} \geq \beta \sum_{u \in U} y_{uv} && \text{for all } v \in V && (5.10)\\
\text{Maximize:} \quad & \textstyle\sum_{(u,v) \in U \times V} \omega(u,v) \cdot y_{uv} && && (5.11)
\end{aligned}$$

Expressions (5.7) and (5.8) state the condition that $y_{uv} = 1$ if and only of $x_u = x_v = 1$. Expression (5.8) ensures that, for any edge whose end points (u, v) are chosen to be in G', y_{uv} is set to 1. Due to the use of y_{uv} variables, this formulation requires $O(|U||V|)$ variables and constraints.

5.4.3 Improved Linear Programming

Observe that constraint (5.7) becomes trivial if $y_{uv} = 0$. In other words, this constraint formulates implications, e.g., for binary variables p and q, the expression $p \leq q$ is equivalent to $p \rightarrow q$. Expanding on this idea, we eliminate the requirement of variables y_{uv} in constraints (5.9) and (5.10) in the next formulation while sharing the rest of the aforementioned linear program.

$$\text{Subject to:} \quad \sum_{v \in V} (\omega(u, v) - \alpha)\, x_v \geq |V|(x_u - 1) \qquad \text{for all } u \in U \tag{5.12}$$

$$\sum_{u \in U} (\omega(u, v) - \beta)\, x_u \geq |U|(x_v - 1) \qquad \text{for all } v \in V \tag{5.13}$$

There is a variable x_v for every vertex v in G. There is a variable y_{uv} for every edge (u, v) in G whose weight is not 0. The variable y_{uv} is set to 1 if and only if both x_u and x_v are set to 1. For any vertex $u \in U$ ($v \in V$), the variable x_u (x_v) is set to 1 if and only if vertex u (v) is in G'. Constraint (5.12) can also be explained as follows. If $x_u = 1$, the constraint transforms to the second constraint in the α,β-WQB definition. If $x_u = 0$, constraint (5.12) becomes trivial. Constraint (5.13) can be explained in a similar manner.

5.4.4 Generalized Formulation for α,β-WQB$_P$ and α,β-WQB$_C$

Recall that the difference between the two problems α,β-WQB$_P$ and α,β-WQB$_C$ is in the edge weight summation which we can combine as the following properties: (1) $\forall u \in U'$: $\sum_{v \in V'} \omega(u, v) \geq \alpha_P |V'| - \alpha_C$, and (2) $\forall v \in V'$: $\sum_{u \in U'} \omega(u, v) \geq \beta_P |U'| - \beta_C$, where (α_P, β_P) and (α_C, β_C) are the parameters given in α,β-WQB$_P$ and α,β-WQB$_C$ respectively. Following the same reasoning in the previous paragraphs, linear constraints (5.12) and (5.13)

are now updated as the following.

$$\text{Subject to:} \quad \sum_{v \in V} (\omega(u,v) - \alpha_P)\, x_v + \alpha_C \geq |V|(x_u - 1) \qquad \text{for all } u \in U \tag{5.14}$$

$$\sum_{u \in U} (\omega(u,v) - \beta_P)\, x_u + \beta_C \geq |U|(x_v - 1) \qquad \text{for all } v \in V \tag{5.15}$$

As a results, the problem instance is a α,β-WQB$_C$ problem if $(\alpha_P, \beta_P) = (1,1)$, and it is a α,β-WQB$_P$ problem if $(\alpha_C, \alpha_C) = (0,0)$. Note that the formulation does not require either condition to present; it essentially defines a generalization of α,β-WQB problems when all 4 parameters are valid and non-zero.

If there are n vertices in U and m vertices in V, there will be a total of $m + n + 2k$ constraints and $m+n+k$ variables where k is the number of edges whose weight is not equal to 0. The above formulations can be extended to solve the query problem by adding an additional constraint $x_v = 1$ to the formulation, for every vertex $v \in P \cup Q$. Similar constraints also help us explore sub optimal solutions, e.g., excluding known vertices in subsequent solutions, or provide a lower-bound of required query items in the optimal solution.

5.5 Results and Discussion

Finding appropriate values for α and β is a critical part in an application. With no specific standard set to compare to, we use simulated data sets to explore the problem. We then use our IP model to explore α,β-WQB's in a real world application, a recent data set of functional groups formed in genetic interactions. The filtered data set, compared to the raw data, served to investigate the role of non-existing edges in the input bipartite graph. While mathematically equivalent in the modeling step, a non-edge in a experimentally generated network represents either a true non-interaction or a false-negative. Assuming the input consists of meaningful features, our preliminary results show that α,β-WQB's may recover missing edges with potentially higher weights better than δ-quasi-bicliques.

5.5.1 Simulations

As part of simulation studies, we try to retrieve a known maximum weighted quasi biclique from a weighted bipartite graph using both versions of α,β-WQB's. In each simulation ex-

periment we do the following. The pair (U, V) represents the vertices of a weighted bipartite graph G. We randomly choose $U' \subseteq U$ and $V' \subseteq V$ as vertices of the known quasi biclique in G. The sizes of both U' and V' are set the same and is picked randomly, but is limited to a specific percentage of the total vertices on each side. Random edges between the vertices of U' and V' in G are introduced according to a pre-determined edge density d. The edges between vertices of $U \backslash U'$ and $V \backslash V'$ of G are also generated randomly according to a pre-determined density d'. The edge weights of the known quasi-biclique (U', V') are determined by a Gaussian distribution with a mean mn and standard deviation dev. Weights of the edges of G not present in the quasi-biclique are also determined by a Gaussian distribution with a lower mean mn' and standard deviation dev'. We now retrieve maximum weighted α,β-WQB$_P$ and α,β-WQB$_C$ from G by using specific values α and β calculated as described below.

For retrieving α,β-WQB$_P$, the values α and β are chosen in two different ways. As part of the simulation we evaluate the performance of both methods. The first method sets both α and β to the mean of the weights of the edges of the quasi biclique. In the second method, α and β are calculated as given below:

$$\alpha = \min \{C_{u'} \mid C_{u'} = (\textstyle\sum_{v' \in V'} w(u', v'))/|V'| \text{ for all } u' \in U'\}$$
$$\beta = \min \{C_{v'} \mid C_{v'} = (\textstyle\sum_{u' \in U'} w(u', v'))/|U'| \text{ for all } v' \in V'\}$$

Similarly, for retrieving α,β-WQB$_C$, the values α and β are calculated as given below.

$$\alpha = |V| - \min \{C_{u'} \mid C_{u'} = (\textstyle\sum_{v' \in V'} w(u', v')) \text{ for all } u' \in U'\}$$
$$\beta = |U| - \min \{C_{v'} \mid C_{v'} = (\textstyle\sum_{u' \in U'} w(u', v')) \text{ for all } v' \in V'\}$$

The ILP models of the corresponding α,β-WQB problems are generated in Python, and solved in Gurobi 4 [31] on a PC with an Intel Core2 Quad 2.4 GHz CPU with 8 GB memory. For the evaluation, let (U'', V'') represent the maximum weighted α,β-WQB returned by the ILP model. The number of vertices of U' in U'' and V' in V'' is our evaluation criterion. The percentage of vertices of U' (V') in U'' (V'') is called the recall of U' (V'). For a specific graph sizes experiments were run by varying the values mn and mn'. The values dev and dev' were set 0.1. The densities d and d' are set to 0.8 and 0.2. The experiments were run for graphs of

size 16×16, 32×32 and 40×40. Each experiment is repeated thrice and the average number of recalled vertices is calculated. The recall of the experiments can be seen in Table 5.1. For each graph size the first two columns represent the recall values for percentage version α,β-WQB's and the third column represents the recall value for constant version α,β-WQB. As the difference between the means increases, so does the average recall. The second method of choosing α and β for percentage version α,β-WQB yields a consistently higher recall.

		16×16					
mn	mn'	Method 1		Method 2		Constant	
		A_U	A_V	A_U	A_V	A_U	A_V
0.5	0.5	33	77	55	100	44	22
0.55	0.45	27	38	100	100	83	88
0.6	0.4	66	77	83	88	100	88
0.65	0.35	72	77	88	88	88	88
0.7	0.3	72	100	100	100	100	100
0.75	0.25	66	83	100	100	100	100
0.8	0.2	66	100	100	100	100	100
		32×32					
mn	mn'	Method 1		Method 2		Constant	
		A_U	A_V	A_U	A_V	A_U	A_V
0.5	0.5	33	30	55	6	11	40
0.55	0.45	0	0	0	0	0	0
0.6	0.4	88	56	91	80	66	70
0.65	0.35	56	66	40	66	40	66
0.7	0.3	78	83	91	91	85	91
0.75	0.25	70	69	100	100	100	100
0.8	0.2	100	78	100	91	100	91
		40×40					
mn	mn'	Method 1		Method 2		Constant	
		A_U	A_V	A_U	A_V	A_U	A_V
0.5	0.5	75	37	83	79	61	62
0.55	0.45	0	25	0	0	11	0
0.6	0.4	49	53	61	66	61	66
0.65	0.35	66	91	91	100	91	100
0.7	0.3	88	77	100	83	100	83
0.75	0.25	64	91	93	100	93	100
0.8	0.2	70	76	100	100	100	100

Table 5.1: Recall of vertices in the simulation. For every experiment, the value in the $A_U(A_V)$ column represents the average recall of $U'(V')$.

5.5.2 Genetic Interaction Networks

A comprehensive set of genetic interaction and functional annotation published recently by Costanzo *et al.* [11] is amongst the best single data sources for weighted biological networks. The aim of our application is to identify the maximum weighted quasi-bicliques consisting of genes in different functional classes in the Costanzo dataset.

Pairwise comparisons of the total 18 functional classes provide 153 sets. For every distinct pair (A, B) of such classes, we build a weighted bipartite graph $(U_A + V_B, E, \omega)$ where genes from functional class A are represented as vertices in U_A and genes from functional class B are represented as vertices in V_B.

The absolute values of the interaction score ε, are used as the edge weights. Values greater than 1 are rounded off to 1. Any gene present in both the functional classes A and B is represented as different vertices in the partitions U_A and V_B and the edge between those vertices is given a weight of 1. We build LP models of both α,β-WQB versions for the bipartite graphs to identify the maximum weighted quasi-bicliques.

5.5.2.1 Biological Interpretation and Examples

Genes with high degree and strong links dominate the results. In several instances, the quasi-bicliques are trivial in the sense that only one gene is present in U', and it is linked to more than 20 genes in V'. Such quasi-bicliques are maximal by definition but provide limited insight. A minimum of $m = 2$ genes per subset was included as an additional constraint to the LP model. It might be sensible to implement such restrictions in the application in general.

We observed the following with the maximum weighted α,β-WQB$_P$'s in the data sets. we Given the low overall weight, the α,β-WQB$_P$'s generated with the parameters $\alpha = \beta = 0.1$ were the most revealing. A notable latent set that was obtained identified genes involved in amino acid biosynthesis (SER2, THR4, HOM6, URE2) and was found to form a 4×10 maximum weighted quasi-biclique with genes coding for proteins of the translation machinery, elongation factors in particular (ELP2, ELP3, ELP4, ELP6 , STP1, YPL102C, DEG1, RPL35A, IKI3, RPP1A). These connections, to our knowledge, are not described and one might speculate

that this is a way how translation is coupled to the amino-acid biosynthesis. In some cases the maximum weighted quasi-biclique is centered around the genes that are annotated in more than one functional class as they provide strong weights. These genes are involved in mitochondrial to nucleus-signaling and are examples where our approach recovers known facts. Using the query approach, it is possible to obtain quasi-bicliques around a gene set of interest quickly and extend the approach proteins of interest.

Maximum weighted α,β-WQB$_C$'s generated from the data sets with parameters $\alpha = \beta = 5$ reveal the following. Genetic interaction networks allow to study protein-coding genes as well as genes that might only code for RNAs. A noteworthy example was discovered in the comparison of genes involved in nuclear transport and those with an unknown bioprocess revealed proteins that are part of the nuclear pore transport (POM34, NUP60, NUP157, THP2 and POM152). They interact with a number of genes that are lined up on chromosome 15 (YML033W, YML034C (SRC1) and YML035C-A) as well as and YDR431W. Most of these genes they interact with are annotated as "dubious" in the current version of the Yeast Genome Database SGD [23]. SRC1 overlaps with another uncharacterized gene YML034C-A. It would be possible that locus codes of a long RNA are involved in nuclear transport.

5.5.2.2 Recovering Missing Edges

The published data sets have edges under different thresholds removed. To sample such missing edges, we calculate the average weight of all the edges removed in the 153 bipartite graphs (generated above), and the calculated average weight is 0.0522.

For each of the 153 maximum weighted quasi-bicliques of either version, the missing edges induced by the quasi-bicliques are then identified, and the average missing edge weight e of each is calculated. The average missing edge weight e is always greater than 0.0522. In other words, we observe that a missing edge in a maximum weighted quasi-biclique has a higher expected weight than the weight of a randomly selected missing edge. This happens when the α and β values are chosen to derive α,β-WQB's which are more dense in terms of weight.

We further compare e from our approach to e from the δ-quasi-bicliques (δ-QB) described by Liu *et al.* [43]. All quasi-bicliques (including exact δ-QB using our IP formulation) used to

induce average missing edge weight e are:

(1) D05/M1: δ-QB with $\delta = 0.5$ and minimum node size is 1, i.e., $m = 1$.

(2) D05/M2: δ-QB with $\delta = 0.5$; $m = 2$.

(3) AB/M2: α,β-WQB$_P$ using the minimum average edge weights found from D05/M2 as α and β; $m = 2$.

(4) AX/M2: α,β-WQB$_P$ where $X = \alpha = \beta \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$; $m = 2$.

(5) CX/M2: α,β-WQB$_C$ where $X = \alpha = \beta \in \{1, 2, 3, 4, 5\}$; $m = 2$.

Comparing the averages of e from A005/M2 to A05/M2 (please see Table 5.2), we see a steady increase. Since α and β can be seen as expected edge weights of the resulting QB, the changing in e shows that QB's identify sub-graphs of expected edge weights. However, we do not see a similar pattern in the constant versions. Overall, in this particular experiment data set, the removed edge weights are at most 0.16, hence e can never approach closely to the parameter α no matter how lenient the parameters are.

WQB$_P$	D05/M1	D05/M2	AB/M2	A005/M2	A01/M2	A02/M2	A03/M2	A04/M2	A05/M2
avg(e)	0.0855	0.0844	0.0850	0.0806	0.0830	0.0867	0.0905	0.0934	0.1169
WQB$_C$					C1/M2	C2/M2	C3/M2	C4/M2	C5/M2
avg(e)					0.1008	0.0805	0.0809	0.0823	0.0825

Table 5.2: A comparison of e under various QB parameters showing improvements of recovered edge weight expectation in α,β-WQB's.

5.6 Conclusions and Outlook

We address noise and incompleteness in biological networks by introducing graph-theoretical optimization problems that identify variations of novel weighted quasi-bicliques. These quasi-biclique problems incorporate biological interaction levels in different analytical settings and exhibit improvements over un-weighted quasi-bicliques. To meet demands of biologists we also provide a query version of (weighted) quasi-biclique problems. We prove that our problems are NP-hard, and describe IP formulations that can tackle moderate-sized problem instances. Simulations and empirical data solved by our IP formulation suggest that our weighted quasi-biclique problems are applicable to various other biological networks.

Future work will concentrate on the design of algorithms for solving large-scale instances of weighted quasi-biclique problems within guaranteed bounds. Greedy approaches may result in effective heuristics that can analyze ever-growing biological networks. A practical extension to the query problem is the development of an efficient enumeration of all maximal α,β-WQB's.

CHAPTER 6. SUMMARY AND OUTLOOK

6.1 Summary

In this study, we explored an ILP solution to the GTP problem under gene duplication model, a dynamic programming solution to the same problem with better performance, and weighted quasi-bicliques to enhance knowledge that can be extracted from a bipartite graph.

Our ILP formulation to the GTP problem under gene duplication model demonstrated that an exact solution to a NP-Complete problem can yield practical results. From the experience of ILP formulation, we further designed the dynamic programming (DP) solution that could solve the GTP problem under various cost function and outperformed the ILP formulation. While the ILP formulation leveraged on existing software packages, the dynamic programming solution further took advantage of computer hardware advancement and data representation to achieve its performance.

Our simulated experiments in the quasi-biclique study showed that we successfully recovered known maximum weighted quasi bicliques from weight bipartite graphs. An empirical experiment of Quasi-Biclique on a genetic interaction network discovered biological important sub-graphs.

6.2 Outlook

As demonstrated, the weighted quasi-biclique has applications broader than gene-species relations that the study focused on. We had also observed that weighted quasi-biclique problem shares similar structures to knapsack problems which should provide further expansions to solve weighted quasi-biclique problems effectively.

While both ILP and DP solutions were designed to solve the same GTP problem, the paths

of achieving the goal are quite different. In the case of ILP formulations, since a binary tree structure is encoded in the solution matrix, we may modify the constraints to solve minimum flipping problem as well as RF distances.

The DP solution to the GTP problem had already shown promising flexibility by solving the problem under different cost functions. Other improvements to the solution may include limiting the candidate splits as well as leveraging parallel programming techniques.

From a computer science perspective, the problems and solutions presented here in this study are practical and applicable to the scientific community interested in phylogenetic analysis, as well as future research based on the progress presented here.

BIBLIOGRAPHY

[1] Bansal, M. and Shamir, R. (2011). A note on the fixed parameter tractability of the gene-duplication problem. *IEEE/ACM Trans. Comput. Biol. Bioinf.*, 8(3):848–50.

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

[3] Bender, M. A. and Farach-Colton, M. (2000). The LCA problem revisited. *LATIN 2000: Theoretical Informatics*, pages 88–94.

[4] Braukmann, T. W. A., Kuzmina, M., and Stefanović, S. (2009). Loss of all plastid ndh genes in Gnetales and conifers: extent and evolutionary significance for the seed plant phylogeny. *Current Genetics*, 55(3):323–337.

[5] Burleigh, J. G., Driskell, A. C., and Sanderson, M. J. (2006). Supertree bootstrapping methods for assessing phylogenetic variation among genes in genome-scale data sets. *Syst. Biol.*, 55(3):426–440.

[6] Burleigh, J. G. and Mathews, S. (2004). Phylogenetic signal in nucleotide data from seed plants: Implications for resolving the seed plant tree of life. *Am. J. Bot.*, 91(10):1599–1613.

[7] Burleigh, J. G. and Mathews, S. (2007). Assessing systematic error in the inference of seed plant phylogeny. *Int. J. Plant Sci.*, 168(2):125–135.

[8] Chang, W.-C. (2005). Gene tree reconcilation with soft multifurcations. Master's thesis, Iowa State University.

[9] 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.

[10] Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2009). *Introduction to algorithms*. The MIT Press, 3rd edition.

[11] Costanzo, M., Baryshnikova, A., Bellay, J., Kim, Y., Spear, E. D., Sevier, C. S., Ding, H., Koh, J. L. Y., Toufighi, K., Mostafavi, S., Prinz, J., St Onge, R. P., VanderSluis, B., Makhnevych, T., Vizeacoumar, F. J., Alizadeh, S., Bahr, S., Brost, R. L., Chen, Y., Cokol, M., Deshpande, R., Li, Z., Lin, Z.-Y., Liang, W., Marback, M., Paw, J., San Luis, B.-J., Shuteriqi, E., Tong, A. H. Y., van Dyk, N., Wallace, I. M., Whitney, J. A., Weirauch, M. T., Zhong, G., Zhu, H., Houry, W. A., Brudno, M., Ragibizadeh, S., Papp, B., Pál, C., Roth, F. P., Giaever, G., Nislow, C., Troyanskaya, O. G., Bussey, H., Bader, G. D., Gingras, A.-C., Morris, Q. D., Kim, P. M., Kaiser, C. A., Myers, C. L., Andrews, B. J., and Boone, C. (2010). The genetic landscape of a cell. *Science (New York, N.Y.)*, 327(5964):425–31.

[12] Cotton, J. A. and Page, R. D. M. (2002). Going nuclear: Gene family evolution and vertebrate phylogeny reconciled. In *Proc. R. Soc. London, Ser. B*, volume 269, pages 1555–1561.

[13] Cotton, J. A. and Page, R. D. M. (2004). *Phylogenetic Supertrees: combining information to reveal the tree of life*, chapter Tangled tr, pages 107–125. Kluwer Academic Publishers, Dordrecht, Netherlands.

[14] Cotton, J. A. and Wilkinson, M. (2007). Majority-rule supertrees. *Systematic biology*, 56(3):445–452.

[15] Crane, P. R. (1985). Phylogenetic analysis of seed plants and the origin of angiosperms. *Annals of the Missouri Botanical Garden*, 72:716–793.

[16] de La Torre-Bárcena, J. E., Egan, M., Katari, M. S., Brenner, E. D., Stevenson, D. W., Coruzzi, G. M., and DeSalle, R. (2006). ESTimating plant phylogeny: lessons from partitioning. *BMC Evol. Biol.*, 6(1):48.

[17] de La Torre-Bárcena, J. E., Kolokotronis, S.-O., Lee, E. K., Stevenson, D. W., Brenner, E. D., Katari, M. S., Coruzzi, G. M., and DeSalle, R. (2009). The impact of outgroup choice and missing data on major seed plant phylogenetics using genome-wide EST data. *PLoS ONE*, 4(6):e5764.

[18] Donoghue, M. J. and Doyle, J. A. (2000). Seed plant phylogeny: Demise of the anthophyte hypothesis? *Current Biology*, 10(3):R106–R109.

[19] Doyle, J. A. (2006). Seed ferns and the origin of angiosperms. *The Journal of the Torrey Botanical Society*, 133(1):169–209.

[20] Doyle, J. A. and Donoghue, M. J. (1986). Seed plant phylogeny and the origin of angiosperms: An experimental cladistic approach. *The Botanical Review*, 52(4):321–431.

[21] Doyon, J.-P. and Chauve, C. (2010). Branch-and-bound approach for parsimonious inference of a species tree from a set of gene family trees. Technical report, LIRMM.

[22] Driskell, A. C., Ané, C., Burleigh, J. G., McMahon, M. M., O'meara, B. C., and Sanderson, M. J. (2004). Prospects for building the tree of life from large sequence databases. *Science (New York, N.Y.)*, 306(5699):1172–4.

[23] Engel, S. R. and Balakrishnan, R. (2010). Saccharomyces genome database provides mutant phenotype data. *Nucleic acids res.*, 38(suppl 1):D433–D436.

[24] Eulenstein, O. (1998). *Vorhersage von Genduplikationen und deren Entwicklung in der Evolution*. PhD dissertation, University of Bonn.

[25] Felsenstein, J. (2005). PHYLIP (Phylogeny Inference Package) version 3.6. Distributed by the author.

[26] Fomin, F., Grandoni, F., and Kratsch, D. (2009). A measure & conquer approach for the analysis of exact algorithms. *Journal of the ACM (JACM)*, 56(5):1–32.

[27] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York.

[28] Goodman, M., Czelusniak, J., Moore, G. W., Romero-Herrera, A. E., and Matsuda, G. (1979). Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences. *Syst. Zool.*, 28:132–163.

[29] Górecki, P. and Tiuryn, J. (2007). Inferring phylogeny from whole genomes. *Bioinformatics*, 23(2):e116–122.

[30] Guigó, R., Muchnik, I., and Smith, T. F. (1996). Reconstruction of Ancient Molecular Phylogeny. *Mol. Phylogenet. Evol.*, 6(2):189–213.

[31] Gurobi Optimization, Inc. (2010). Gurobi Optimization 2.0.2. http://www.gurobi.com/.

[32] Gusfield, D. (1991). Efficient algorithms for inferring evolutionary trees. *Networks*, 21:19–28.

[33] Gusfield, D. (1997). *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press.

[34] Hallett, M. T. and Lagergren, J. (2000). New algorithms for the duplication-loss model. In *RECOMB*, pages 138–146.

[35] Hartmann, S., Lu, D., Phillips, J., and Vision, T. J. (2006). Phytome: a platform for plant comparative genomics. *Nucleic Acids Res.*, 34(Database issue):D724–D730.

[36] Hartmann, S. and Vision, T. J. (2008). Using ESTs for phylogenomics: Can one accurately infer a phylogenetic tree from a gappy alignment? *BMC Evol. Biol.*, 8(1):95.

[37] Hilton, J. and Bateman, R. M. (2006). Pteridosperms are the backbone of seed-plant phylogeny. *The Journal of the Torrey Botanical Society*, 133(1):119–168.

[38] Holland, B. R., Penny, D., and Hendy, M. D. (2003). Outgroup misplacement and phylogenetic inaccuracy under a molecular clock: A simulation study. *Syst. Biol.*, 52(2):229–238.

[39] IBM, Inc. (2009). IBM ILOG CPLEX 12.1. http://www.ibm.com/software/integration/optimization/cplex/.

[40] Jones, D. T., Taylor, W. R., and Thornton, J. M. (1992). The rapid generation of mutation data matrices from protein sequences. *Comput. Appl. Biosci.*, 8(3):275–282.

[41] Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer.

[42] Liu, X., Li, J., and Wang, L. (2008). Quasi-bicliques: Complexity and binding pairs. *Lecture Notes In Computer Science; Vol. 5092*.

[43] Liu, X., Li, J., and Wang, L. (2010). Modeling protein interacting groups by quasi-bicliques: Complexity, algorithm and application. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 99(1).

[44] Ma, B., Li, M., and Zhang, L. (1998). On reconstructing species trees from gene trees in term of duplications and losses. *RECOMB '98 Proceedings of the second annual international conference on Computational molecular biology*, pages 182–191.

[45] Ma, B., Li, M., and Zhang, L. (2000). From gene trees to species trees. *SIAM J. Comput.*, 30(3):729–752.

[46] Maddison, W. and Knowles, L. (2006). Inferring phylogeny despite incomplete lineage sorting. *Systematic Biology*, 55(1):21–30.

[47] Maddison, W. and Maddison, D. (2008). Mesquite: a modular system for evolutionary analysis.

[48] Magallón, S. and Sanderson, M. J. (2002). Relationships among seed plants inferred from highly conserved genes: Sorting conflicting phylogenetic signals among ancient lineages. *Am. J. Bot.*, 89(12):1991–2006.

[49] Martin, A. P. and Burg, T. M. (2002). Perils of paralogy: Using HSP70 genes for inferring organismal phylogenies. *Syst. Biol.*, 51(4):570–587.

[50] Mathews, S. (2009). Phylogenetic relationships among seed plants: Persistent questions and the limits of molecular data. *Am. J. Bot.*, 96:228–236.

[51] 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. *Syst. Biol.*, 57(4):574–590.

[52] Nixon, K. C., Crepet, W. L., Stevenson, D. W., and Friis, E. M. (1994). A reevaluation of seed plant phylogeny. *Annals of the Missouri Botanical Garden*, 81(3):484–533.

[53] Page, R. D. M. (1994). Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Systematic Biology*, 43(1):58.

[54] Page, R. D. M. (2000). Extracting species trees from complex gene trees: Reconciled trees and vertebrate phylogeny. *Mol. Phylogenet. Evol.*, 14(1):89–106.

[55] Page, R. D. M. and Charleston, M. A. (1997). Reconciled trees and incongruent gene and species trees. *Mathematical hierarchies in biology*, 37:57–70.

[56] Papadimitriou, C. and Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity*. Dover Publications.

[57] Rai, H. S., Reeves, P. A., Peakall, R., Olmstead, R. G., and Graham, S. W. (2008). Inference of higher-order conifer relationships from a multi-locus plastid data set. *Botany*, 86:658–669.

[58] Rydin, C., Kallersjo, M., and Friis, E. M. (2002). Seed plant relationships and the systematic position of Gnetales based on nuclear and chloroplast DNA: Conflicting data, rooting problems, and the monophyly of conifers. *Int. J. Plant Sci.*, 163(2):197–214.

[59] Sanderson, M. J. and McMahon, M. (2007). Inferring angiosperm phylogeny from EST data with widespread gene duplication. *BMC Evol. Biol.*, 7(Suppl 1):S3.

[60] Semple, C. and Steel, M. A. (2003). *Phylogenetics*. Oxford University Press.

[61] Soltis, D. E., Soltis, P. S., and Zanis, M. J. (2002). Phylogeny of seed plants based on evidence from eight genes. *Am. J. Bot.*, 89(10):1670–1681.

[62] Stamatakis, A. (2006). RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22(21):2688–2690.

[63] Than, C. and Nakhleh, L. (2009). Species tree inference by minimizing deep coalescences. *PLoS Comput. Biol.*, 5(9):e1000501.

[64] 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.

[65] Woeginger, G. J. (2003). Exact algorithms for NP-hard problems: A survey. *Combinatorial Optimization–Eureka, You Shrink!*, 2570/2003:185–207.

[66] Wu, C.-S., Wang, Y.-N., Liu, S.-M., and Chaw, S.-M. (2007). Chloroplast genome (cpDNA) of cycas taitungensis and 56 cp protein-coding genes of gnetum parvifolium: Insights into cpdna evolution and phylogeny of extant seed plants. *Mol. Biol. Evol.*, 24:1366–1379.

[67] Yan, C., Burleigh, J. G., and Eulenstein, O. (2005). Identifying optimal incomplete phylogenetic data sets from sequence databases. *Molecular phylogenetics and evolution*, 35(3):528–535.

[68] Zhang, L. (2011). From gene trees to species trees ii: Species tree inference by minimizing deep coalescence events. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(6):1685–1691.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my deep gratitude to those who helped me during the process of completing this thesis. First and foremost, Dr. Oliver Eulenstein for his guidance, patience and support throughout the years in researching and writing. Without his kind encouragements and inspirations, this thesis would no complete. I would also like to thank my committee members for their patience and feedback to this work: Dr. Julie Dickerson, Dr. David Fernández-Baca, Dr. Dennis Lavrov, and Dr. Leslie Miller. I would additionally like to thank fellow students in our research lab for all the discussions and wonderful times: Mukul Bansal, Duhong Chen, Ruchi Chaudhary, Akshay Deepak, Sudheer Vakati and André Wehe. Last but not least, I would like to thank my family for all the caring and support that made this work possible.