Master Thesis

On the Integrality of the LP-relaxation for MaxRTC

Marco Kemmerling

Master Thesis DKE18-14

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science of Data Science for Decision Making
at the Department of Data Science and Knowledge Engineering
of the Maastricht University

Thesis Committee:

Dr. S. Kelk
Dr. M. Mihalák

Maastricht University
Faculty of Science and Engineering
Department of Data Science and Knowledge Engineering

June 20, 2018

Abstract

Supertree methods aim to merge multiple phylogenetic trees while resolving any conflicts, by some criterion, gracefully. One way to construct such supertrees is to solve the *maximum rooted triplets consistency problem* (MaxRTC), in which, given a set of rooted triplets, a maximum cardinality subset is selected that can be displayed by a single tree. MaxRTC is NP-hard and no known polynomial-time approximation algorithm better than a 3-approximation exists. When formulated as an ILP, it turns out that the problem can be solved surprisingly fast under certain conditions and that the corresponding LP-relaxation is surprisingly integral under those conditions. Here, we investigate these conditions and determine a certain level of inconsistency in the input above which solutions to the LP-relaxation *cannot* be integral. Based on our results, it is not clear how an approximation algorithm based on a rounding scheme of the LP-relaxation can be designed, although we cannot rule out that such a rounding scheme exists. Our results show that constraining only small subsets of decision variables to be integer often results in completely integral solutions, although there is no obvious way to transform this insight into a useful algorithm.

Contents

Chapter 1

Introduction

Phylogenetic trees are a graph-based representation of the evolutionary relationships between biological species (more abstractly called taxa). At each branch in a phylogenetic tree, two (or more) species diverge from a common ancestor. Rooted trees show the direction of evolution in terms of time, such that the root is the earliest common ancestor of all species in the tee. Unrooted trees lack this feature, but are still often used in practice. The challenge in building these trees is that the evolutionary history of the species in question is usually not known and has to be inferred from the characteristics of species found today. While early phylogenetic trees were constructed based purely on morphological features of species, a lot more information, e.g. in the form of molecular data such as protein and DNA sequences, is taken into consideration today [1]. Phylogenetic trees find use in the prediction of the function of genes and proteins [2], in epidemiology [3], and other many areas.

Supertree methods refer to methods that combine an input set of overlapping phylogenetic trees into one supertree, potentially leading to the inference of relationships between taxa that do not occur together in any single one of the input trees. Such methods are attractive because accurately constructing large phylogenetic trees is challenging, while constructing plausible smaller trees is much more tractable.

Since phylogenetic trees are inferred based on biological data, which is inherently "noisy", the trees themselves may not be totally accurate. As such, different phylogenetic trees may not be perfectly consistent with each other, which poses a problem for supertree methods. In this case, because preserving all the relationships found in the input set of trees is not possible, the problem becomes an optimisation problem. The exact optimisation criterion differs between different methods. One approach is to deconstruct the input trees into atomic relationships called rooted triplets and to then construct a supertree that includes as many of these rooted triplets as possible. The specific optimisation problem examined in this thesis is called the *maximum rooted triplets consis-*

tency problem.

An LP-relaxation of a natural integer linear programming formulation to this problem yields, when applied to well-behaved inputs derived from real data, surprisingly integral optimal results. That is, although the relaxed decision variables can potentially take any value between 0 or 1, for many instances the optimal solutions have the property that many or even all of the relaxed decision variables have value 0 or 1. On the other hand, on random instances and instances that have been corrupted with quite a lot of noise, the decision variables in the LP-relaxation tend to be non-integral. Exploring the mathematical conditions of data that produces integral decision variables may lead to insights that could potentially be leveraged into a better approximation algorithm.

The remainder of this thesis is structured into several chapters, with the next chapter giving some preliminaries, among which several useful mathematical definitions and brief introductions to the techniques employed in the thesis. Next, we give an overview of related work concerning exact and approximation algorithms of MaxRTC, as well as related results on quartet methods. Chapter 4 introduces the ILP formulation which is the central object of the thesis and states the objectives of this thesis. The following two chapters explore the conditions behind the integrality/fractionality of solutions to the LP-relaxation and examine whether variations on the formulation of the ILP can be beneficial. The final chapter gives a discussion and conclusion of the results.

Chapter 2

Preliminaries

2.1 Phylogenetic Trees and Rooted Triplets

some general references would be useful here. *Phylogenetic trees* are graph-based representations of the evolutionary relationships between biological species (more abstractly called taxa).

Here, a phylogenetic tree is defined as a rooted, distinctly leaf-labelled, binary tree (i.e. every internal node has exactly two children).

Given a phylogenetic tree P, A *proper descendant* of a node x is a descendant of x which is not x itself. The *lowest common ancestor* (lca) of two nodes x and y is the lowest node in P that has both x and y as descendants.

A *rooted triplet*, or simply *triplet* for brevity, is a phylogenetic tree with exactly three leaves. We denote a rooted triplet on leaf set $\{a, b, c\}$ by $ab|c$ if the lowest common ancestor (lca) of a and b is a proper descendent of the lowest common ancestor of a and c. Note that, $ab|c = ba|c$. The set of labels that label the leaves of a triplet t is denoted by $L(t)$. A rooted triplet is the smallest possible unit that conveys evolutionary information, a structure based on only two taxa cannot contain any information.

Given a rooted triplet $ab|c$, a tree P is said to be *consistent* with $ab|c$ if the lca of a and b in P is a proper descendent of the lca of a and c in P (see figure 2.1). A set of triplets T is said to be consistent if there exists a tree P such that P is consistent with every triplet in T.

A set of triplets T on a label set L is called *dense*, if for every subset $L^* \subset L$ with $|L^*| = 3$, there is at least one triplet $t \in T$ with $L(t) = L^*$ and *minimally dense* if there is exactly one triplet $t \in T$ with $L(t) = L^*$.

Here, the taxa set N is used interchangeably with the label set L. For

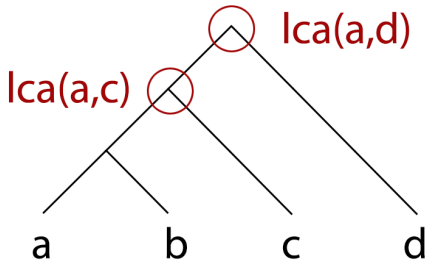

Figure 2.1: The tree above is consistent with triplet $ac|d$ since the lca of a and c is a proper descendant of the lca of a and d. It is *not* consistent with $ad|c$, since the lca of a and d is *not* a descendant of the lca of a and c.

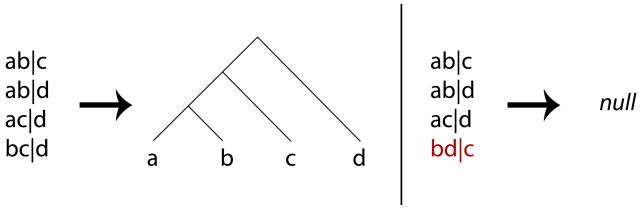

Figure 2.2: An illustration of the output of RTC given different inputs. Inputs are displayed on the left-hand side of each arrow, while the corresponding outputs are displayed on the right-hand side.

convenience, we denote $|N| = n$ and $|T| = k$.

2.2 (Maximum) Rooted Triplet Consistency

Given a set of triplets, we would like build a tree that represents this set of triplets well. This gives rise to two problems: (1) Rooted Triplet Consistency (RTC), a decision problem, and (2) Maximum Rooted Triplet Consistency (MaxRTC), a related optimisation problem.

(1) RTC: Given a set T of rooted triplets on leaf set L, output a phylogenetic tree on leaf set L which is consistent with every rooted triplet in T, if such a tree exists. If not, output null (see figure 2.2).

There may not always be a tree that is consistent with every triplet in R, in which case the following problem becomes relevant:

(2) MaxRTC: Given a set of rooted triplets T on leaf set L, find a maximum cardinality subset $T^* \subset T$ that is consistent.

2.3 (Integer) Linear Programming

A linear program (LP) is a problem that can be expressed in the following form:

$$\begin{aligned} &\text{maximize} && c^T x \\ &\text{subject to} && Ax \leq b \end{aligned}$$

where x is a vector of decision variables (the values of which are to be determined), c and b are vectors of coefficients and A is a coefficient matrix.

The first line in the above linear program is called the objective function, which is some expression to be maximised (or minimised). The second line, the constraints, describe the set of solutions that are permissible, or feasible (see figure 2.3). Often, we further require the variables to be non-negative, although this is not strictly necessary.

Linear programs can be solved by the well-known simplex method, originally proposed by Dantzig [4]. Even though the simplex method can take exponential time to terminate in the worst case, and interior-point methods [5], which are guaranteed to terminate in polynomial time, exist as an alternative, the simplex method is often preferred due to the fact that it is lightweight and typically very quick in practice. It relies on two observations: (1) the optimal solution will lie on one of the corner points of the polyhedron, where a corner point is defined as a point that does not lie on the line segment of two distinct feasible points, (2) due to the convexity of the polyhedron, any corner point x that is locally optimal must also be globally optimal. Here, local optimality means that there is no corner point y adjacent (i.e. having a common edge) to x that is associated with a better objective function value.

The simplex algorithm exploits these two observations in a hillclimbing fashion by starting at some corner point and then repeatedly moving to the adjacent corner point that most improves the value of the objective function (although a variety of different pivoting rules are possible). If none of the adjacent corner points improve the current solution, the global optimum has been found and the algorithm terminates.

An integer linear program (ILP) is identical to a linear program with the exception that the decision variables are required to be integer. Programs that only require some (proper) subset of the variables to be integer are called mixed integer linear programs (MILP). Sometimes integer variables are further restricted to only take binary values.

While, intuitively, the elimination of parts of the solution space (the fractional solutions) should make the problem easier, this is not actually the case. Due to the restriction to integer values, the feasible region ceases to be a convex polyhedron, meaning that the simplex method is not applicable. Not only can the simplex algorithm not be applied, but solving ILPs is NP-hard [6].

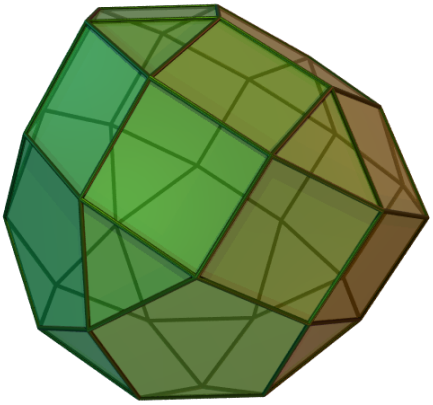

Figure 2.3: Example of a feasible region in three dimensions. Source: https://commons.wikimedia.org/wiki/Polyhedra

Due to the NP-hardness of the problem, finding optima probably requires enumerating solutions in some way. While the number of solutions is finite, and enumeration is thus possible, the search space can be arbitrarily large in practice, necessitating the use of more sophisticated (implicit) enumeration procedures. One common such procedure is based on a branch-and-bound approach [7], where the branching consists of fixing the values of subsets of variables. The solutions of the resulting problems are then computed and used as a bound. This bound is then compared to the (at this point) best-known integer solution, called the incumbent, to determine whether the subtree is worth exploring. Specifically, in the case of binary decision variables, at every level of the tree we branch on some variable x_i by setting $x_i = 0$ in one subtree and $x_i = 1$ in the other subtree. Each of these subtrees can potentially be eliminated by computing the solution to the *LP-relaxation* at the root of the subtree. The LP-relaxation of an ILP is the LP that arises when the integrality constraints of all variables are removed. Since any solution valid for the ILP is also valid for the LP-relaxation, optimal solutions to the LP-relaxation will be as least as good as that of the ILP, and can thus be used to bound the quality of ILP solutions. If the solution to the LP-relaxation at the root of the subtree is integral and better than the current incumbent, it becomes the new incumbent and the subtree can be eliminated, since a better solution cannot possibly be found by fixing additional variables. If the value of the solution to the LP-relaxation is worse than that of the incumbent solution, then the subtree can be eliminated because the solution to the LP-relaxation is an upper bound to all integer solutions in the subtree. If no further subtrees can be eliminated, a new node to branch on is selected and the process is repeated.

In some cases relaxations of ILPs have completely integral solutions. A way

to guarantee the integrality of solutions to LP-relaxations is to have a *totally unimodular* constraint matrix. A matrix is totally unimodular if the determinant of every square submatrix of A is 0, -1, or 1. If A is totally unimodular, then every corner point of $Ax \leq b$ is integral, and thus the solution to the LP-relaxation will be integral [8].

LP-relaxations are also commonly used to obtain approximation algorithms for ILP problems, i.e. algorithms that are guaranteed to find solutions within at least a certain factor of the optimum solution value. This often involves some sort of rounding strategy, where variables that are fractional in the solution to the LP-relaxation are rounded up or down to integral values in such a way that the resulting solution is feasible.

A given LP-relaxation is associated with a so-called *integrality gap* given by:

$$IG = sup_I \frac{OPT(I)}{OPT_{relax}(I)} \tag{2.1}$$

where I denotes a specific instance, and $OPT(I)$ and $OPT_{relax}(I)$ the optimal solution of the ILP and LP-relaxation respectively, i.e. the integrality is the supremum of the ratio between the ratio of the optimal solutions to the ILP and the LP-relaxation.

For approximation algorithms based on rounding the solution of some LP-relaxation, and using the LP-relaxation as the lower bound, the approximation ratio cannot be better than the integrality gap [9].

Chapter 3

Related Work

RTC can be solved exactly in polynomial time as evidenced by Aho's algorithm [10], which runs in $O(kn)$ time.

MaxRTC on the other hand has been shown to be NP-hard [11]. An exact dynamic-programming algorithm proposed by Wu [11] takes $O((k+n^2)3^n)$ time. In practice, an instance on 20 taxa takes approximately 14 hours to solve using this algorithm [11].

A polynomial time algorithm for MaxRTC proposed in [12] achieves a 3-approximation by building a caterpillar tree that is always consistent with at least a third of the input triplets. Since it finds a tree consistent with at least a third of the input triplets, it is guaranteed that the solution is at least a third of the optimal one. Other 3-approximation algorithms have been proposed in [13] (based on a derandomization strategy for labelling a given tree topology) and in [14] (based on a bottom up approach). [13] notes that achieving a 2-approximation or better is unlikely. find reference for the unique games conjecture thing

There are heuristic approaches to solve the problem in practice, such as [15] which applies min cut on a graph that represents the triplets inconsistency, or [16], which is based on an algorithm proposed in [17] which applies a max cut on a variant of the same graph. The method in [16] was tested on artificial data with up to 750 taxa and several real datasets with up to 267 taxa. While no formal guarantees exist, the approach has been demonstrated to achieve good results in practice and can solve instances of the aforementioned sizes within minutes.

Methods based on (unrooted) quartets instead of triplets have received considerable attention in the literature as well and are potentially interesting since there are many commonalities between rooted and unrooted methods. Unrooted trees do not explicitly state in which direction the evolution of the represented

species occurred. When constructing unrooted trees, no useful information can be inferred from unrooted triplets, similar to the fact that relationships between two taxa are not helpful in constructing rooted trees. Unfortunately, even the quartet consistency decision problem (the quartet analogue of RTC) is NP-hard [18]. An ILP similar to the one examined in the following chapters has been considered for quartet methods in [19].

It may be of importance what assumptions can be made about the given set of triplets. For instance, [20] points out that, a polynomial-time approximation scheme (PTAS) is possible if the set of triplets is dense. A PTAS is a polynomial-time algorithm that, given an instance and a parameter $\varepsilon > 0$ produces a solution that is within $1 - \varepsilon$ of the optimum [9]. While polynomial, the running time increases with the requested accuracy, such that arbitrarily increasing the accuracy quickly becomes prohibitive. Since it is known that a PTAS is possible for dense triplet sets, results on non-dense sets would be more interesting in terms of improving upon a 3-approximation.

Chapter 4

An ILP Approach to MaxRTC

The following presents an ILP to solve MaxRTC that is based on the observation that a set of triplets forms a valid tree (i.e. a tree exists such that every triplet in the set is consistent with that tree), if there is an absence of local conflicts, as demonstrated in [21].

More specifically, for minimally dense triplet sets, a set of triplets forms a valid tree if and only if, for every subset of four leaves $\{a, b, c, d\}$, the subset of triplets whose leaves are in $\{a, b, c, d\}$ are conflict-free. This is equivalent to saying that, for every subset of four leaves $\{a, b, c, d\}$, if $ab|c$ and $bc|d$ are in the set, then so are $ab|d$ and $ac|d$ [21].

Given a triplet set T and a taxa set N, an ILP can thus be defined as follows:

$$Max \sum_{ij|k \in T} t_{ij|k} \tag{4.1}$$

$$t_{ij|k} + t_{ik|j} + t_{jk|i} = 1 \qquad \forall i, j, k \in N \tag{4.2}$$

$$t_{ij|k} + t_{jk|l} - t_{ik|l} \leq 1 \qquad \forall i, j, k, l \in N \tag{4.3}$$

$$t_{ij|k} + t_{jk|l} - t_{ij|l} \leq 1 \qquad \forall i, j, k, l \in N \tag{4.4}$$

$$t_{ij|k} \in \{0, 1\} \qquad \forall i, j, k \in N \tag{4.5}$$

where

$$i \neq j \neq k \neq l \tag{4.6}$$

A feasible solution to this ILP is a tree, or more specifically a consistent subset of size $\binom{n}{3}$ of the $3\binom{n}{3}$ triplets possible on taxa set N. On a high level,

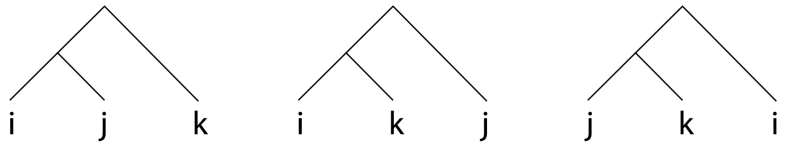

Figure 4.1: The three different topologies on a set of three taxa mentioned in constraint 4.2.

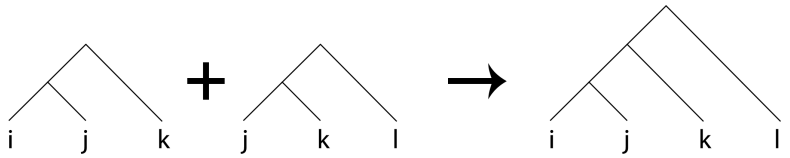

Figure 4.2: Intuition behind constraints 4.3 and 4.4: any tree which displays $ij|k$ and $jk|i$ must also display $ik|l$ and $ij|l$.

the ILP selects a feasible tree such that this tree is consistent with a maximum cardinality subset of triplets in T.

Three different triplet topologies are possible for every subset of three leaves. These topologies are mutually exclusive, since a single tree represents exactly one topology. (see figure 4.1). Further, if none of them are selected, then the given leaves must be siblings, which contradicts the assumption of binary trees. Hence, we require that exactly one of the triplets must be selected, which is enforced by equation 4.2.

Equations 4.3 and 4.4 model the logical implications described in [21], i.e. if a tree contains triplets $ij|k$ and $jk|l$, then it must also contain triplets $ik|l$ and $ij|l$. This is split up into the two implications $t_{ij|k} \wedge t_{jk|l} \rightarrow t_{ik|l}$ (constraint 4.3) and $t_{ij|k} \wedge t_{jk|l} \rightarrow t_{ij|l}$ (constraint 4.4). These implications ensure that the triplets selected by the ILP solver form a valid tree, e.g. there is no tree which contains $t_{ij|k}$ and $t_{jk|l}$ but not $t_{ik|l}$. The first two triplets in each of these constraints will be referred to as the antecedent, while the third one will be referred to as the consequent.

Note that the ILP has $O(n^3)$ variables and $O(n^4)$ constraints and thus the resulting programs quickly become very large, as is illustrated in table 4.1.

n	# variables	# constraints
10	10^3	10^4
20	10^3	10^5
30	10^4	10^5
40	10^4	10^6
50	10^5	10^6
100	10^6	10^8
200	10^6	10^9
10^6	10^{18}	10^{24}

Table 4.1: Number of decision variables and constraints in the ILP with growing numbers of taxa. The last entry in the table corresponds to the number of taxa in the "comprehensive tree of life" presented in [22].

4.1 Thesis Objectives

Earlier work in [23] suggests that the ILP may run surprisingly quickly under certain conditions, and that this might be related to a high degree of integrality in the LP-relaxation under those conditions. The goal of the succeeding chapters is to verify and further explore this claim by examining the following questions:

1. Under which conditions will the LP-relaxation yield solutions in which many decision variables take the value 0 or 1 (as opposed to fractional values)?

2. Can these conditions be used to improve current approximation ratios of MaxRTC?

3. Are variations of the ILP possible and, if yes, are they beneficial in terms of how quickly solutions to the ILP can be found?

Question 1 and 2 will be addressed in chapter 5, while question 3 will be addressed in chapter 6.

Chapter 5

Exploration of Integrality Conditions

5.1 Data

In the following, the performance of the (I)LP is examined on both artificial and real data. Two approaches are taken to generate artificial data: (1) multiple random trees are generated and the union of triplets obtained from those trees are used as the input to the (I)LP, (2) a single random tree is generated and all its triplets are extracted, resulting in a minimally dense triplet set. In order to convert this triplet set into a set with some level of inconsistency, a certain fraction c of triplets is corrupted, where corruption means that the selected triplet is changed from its original topology A to one of its alternate topologies B or C (with equal probability). The probability to keep the original topology of any given triplet is thus $P(A) = 1 - c$, while the probabilities of each of the other topologies is $P(B) = P(C) = \frac{c}{2}$. At $c = \frac{2}{3}$, each of the topologies has an equal chance ($= \frac{1}{3}$) of appearing. The resulting set has "maximum entropy", i.e. the data is devoid of any signal. This way of generating data by corrupting the triplets from a single tree allows relatively precise control over the degree of "inconsistency" in the input, but might not result in very realistic data.

Random trees are generated by starting with a leaf set of size 1 and repeatedly picking a random leaf from this set to branch on until the desired number of taxa is reached. When branching, the original leaf is removed from the set and replaced with two new ones. On average, the resulting trees are balanced. pseudocode?

Unless otherwise specified, the experiments presented here are performed using the well-known commercial (I)LP solver CPLEX [24] with default parameters. CPLEX uses an internal algorithm to determine which solution method to apply. By default, it uses the (dual) simplex method, but may in some in-

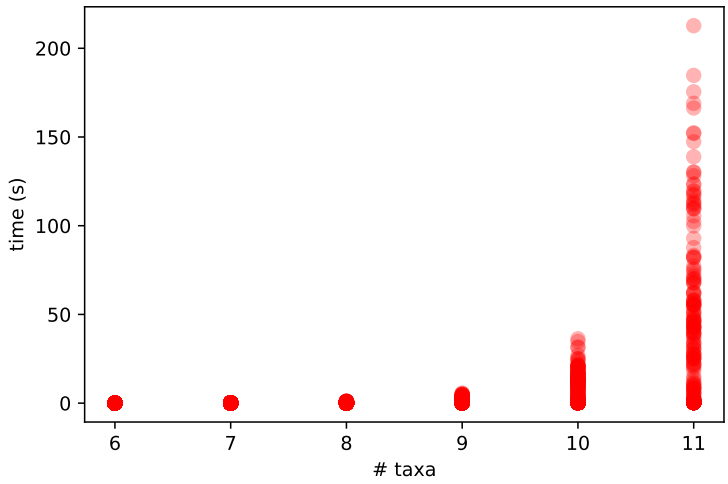

Figure 5.1: CPLEX solving time of the ILP measured in seconds as a function of taxa.

stances conclude that applying interior-point methods is more advantageous.

5.2 (I)LP Performance on Corruption Data

The experiments presented in this section are based on sets of (I)LP instances with corruption values ranging from 0% to 66% in 2% increments and between 6 to 11 taxa. For each combination of corruption and taxa values, 10 instances are randomly generated, which amounts to 2040 instances in total.

The amount of time the ILP needs to solve instances is taken as the starting point of the investigation into the ILPs behaviour. Unsurprisingly, the higher the number of taxa, the longer the time required to solve an instance (see figure 5.1). As can be seen in figure 5.2, instances with less than 40% corruption can always be solved quickly, while instances with more than 40% corruption can be much slower to solve.

Since the ILP relies on the underlying LP-relaxation, examining the relaxation might yield insights into this phenomenon. Especially of interest are the questions: (1) Is there a relationship between the amount of corruption and the occurrence of integral/fractional values in the solution of the LP-relaxation, (2) when fractional solutions occur, what values do the fractional variables take?

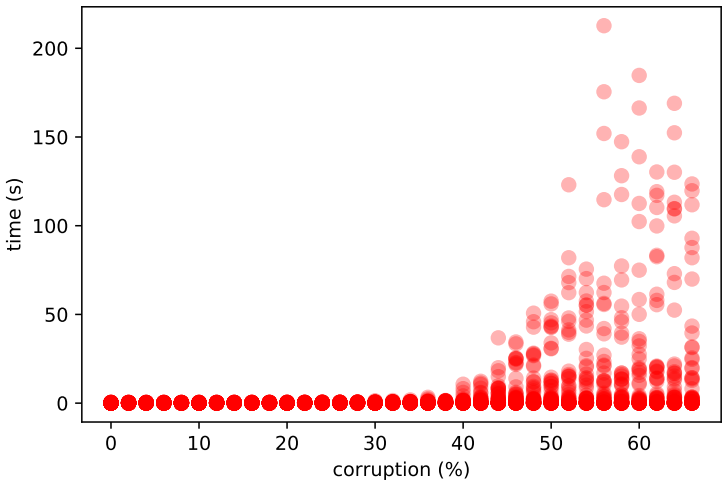

Figure 5.2: CPLEX solving time of the ILP measured in seconds as a function of the amount of corruption.

In order to answer question (1), we examine the average proportion of fractional values, i.e. the proportion of decision variables that are fractional for instances with the same taxa-corruption configuration. As figure 5.3 shows, there is a fairly abrupt phase change around 40% corruption, with *almost* no fractional solutions on smaller amounts of corruption, and a dramatic increase in the proportion of fractional values with higher amounts of corruption.

One might suspect that in the case of integral solutions, the constraint matrix is totally unimodular. However, the constraints depend only on the number of taxa, not on the exact set of input triplets, and the only thing changed by corruption is the objective function. In our experiments, for every given number of taxa we observed instances with fractional optima, so the underlying constraint matrix is not totally unimodular. While instances might exist whose integrality is due to a totally unimodular constraint matrix, this is not the case in any of the instances observed in these experiments, and totally unimodular constraint matrices can therefore not serve as a general explanation for the observed phenomenon.

The preceding chain of reasoning hinges on the assumption that (a variant of) the simplex method rather than an interior-point method is used to solve instances. Since the simplex method operates on the corner points, a fractional solution shows that at least one fractional corner point exists. It appears that, for instances with ≥ 10 taxa and $\geq 40\%$ corruption, CPLEX does often switch to interior point methods. However, when the experiments are replicated using the simplex method, the results remain consistent, i.e. fractional solutions can

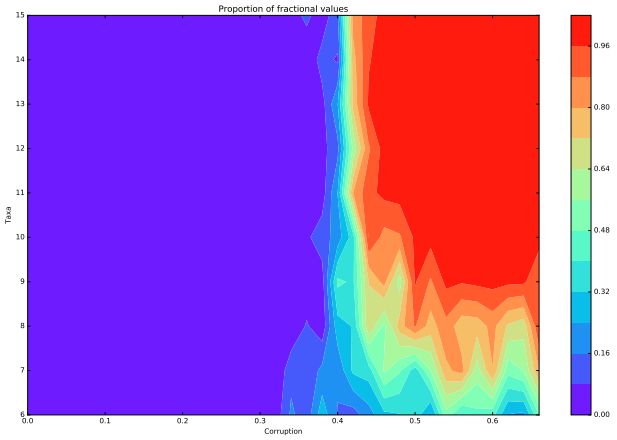

Figure 5.3: Average proportion of fractional values as a function of the amount of corruption and the number of taxa. The vertical axis represents the amount of corruption and the horizontal axis represents the number of taxa. The colour signifies the relative amount of fractional values, with blue meaning little or no fractional values and red meaning almost complete fractionality.

still be observed for the same instances as before and none of the figures presented in this section change in any noticeable way.

The answer to question (2), i.e. what kind of fractional values occur in solutions to the LP-relaxation is illustrated in figure 5.6. Most prominently, at 40% corruption, the dominant values change from 0 and 1 to 0.2 and 0.6. So again, in this figure as in the others, there is a clear change in behaviour at 40% corruption.

In trying to understand the phenomena presented here, we can ask two similar questions: (1) Why do instances with low amounts of corruption tend to be integral? (2) Why do instances with high amounts of corruption tend to be fractional? The latter question is explored in section 5.4. A first step towards the former question, namely the reason behind integrality with 0% corruption, is provided below.

Lemma 5.2.1. *If there is no corruption, giving every triplet in the objective function a value of* 1 *is the only optimal solution for both the ILP and the LP-relaxation.*

Proof. Clearly, the resulting solution is feasible since the triplets are extracted from a single tree and not corrupted. This solution cannot be further improved

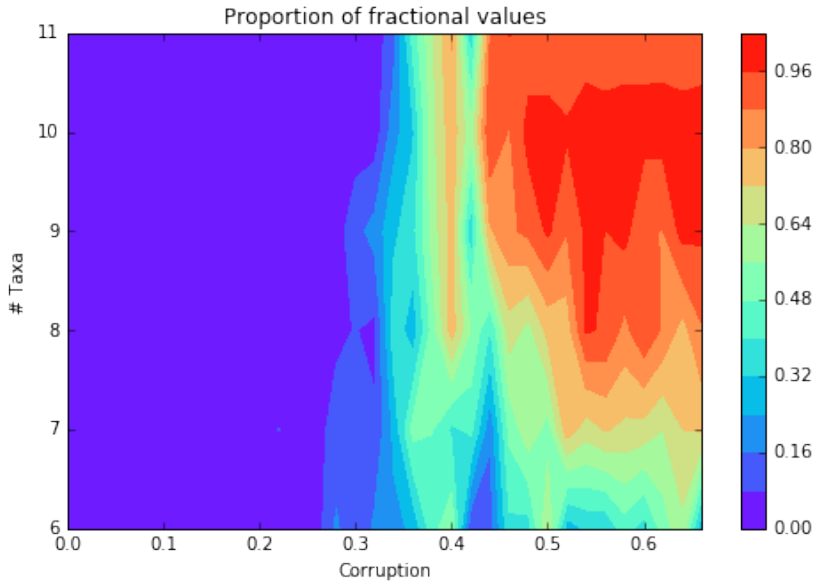

Figure 5.4: Heatmap as in figure 5.3, but here the data is generated using GLPK instead of CPLEX.

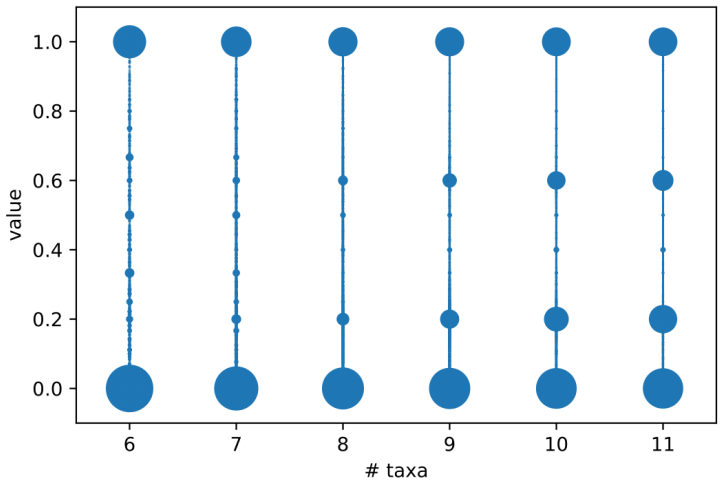

Figure 5.5: Each circle represents a unique fractional value taken by a decision variable, given on the vertical axis. The size of each circle is determined by the frequency of the respective value. Values from all variables (not just the one in the objective function) are visualised as a function of the number of taxa. All levels of corruption are considered for each given number of taxa.

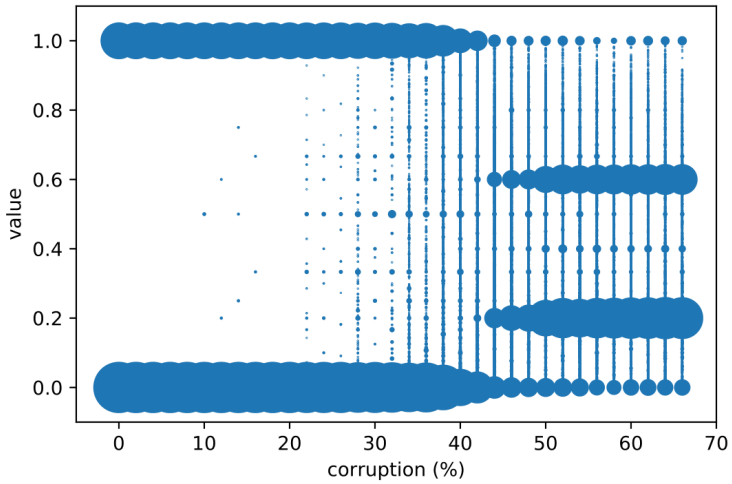

Figure 5.6: Each circle represents a unique fractional value, given on the vertical axis. The size of each circle is determined by the frequency of the respective value.These values are displayed as a function of the amount of corruption.

since every variable in the objective function already has the highest value it can possibly take. To arrive at a fractional solution, at least one of the values would have to be decreased, without any chance of compensating for it by increasing some other value. □

5.2.1 GLPK

Since CPLEX is a fairly sophisticated (I)LP solver, it might be that some CPLEX-specific optimisation is responsible for the phenomena observed here. To verify that these phenomena are indeed solver-independent, we repeat the experiments using GLPK [25], an open-source (I)LP solver, which is generally more primitive than CPLEX. The results, given in figure 5.4, are not exactly the same as the ones obtained through CPLEX, but the sudden phase change at 40% corruption is present here as well.

5.2.2 Early Termination

this feels orphaned, consider moving somewhere else It may be the case that the ILP-solver always finds the optimal solution quickly but cannot prove that this solution really is the optimum right away. In that case it may be possible to argue that, empirically, the ILP-solver can be stopped after relatively few iterations without sacrificing the quality of the solution.

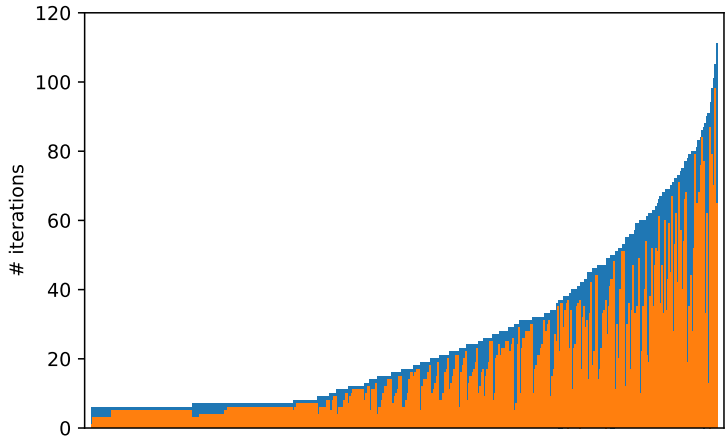

Figure 5.7: Comparison of the total number of iterations and how many iterations are executed after the optimal solution is found. Each bar represents one instance, with the blue part denoting the number of iterations after the optimal solution has been found and the orange part denoting the number of iterations before this point. Instances with ≤ 5 total iterations are excluded. The remaining instances are displayed in order of the number of total iterations.

As figure 5.7 shows, this is the case for some instances, but not consistently for all instances. It is thus not advisable to terminate the solving process prematurely if optimal solutions are desired.

5.3 (I)LP Performance on Multiple Tree Data

As described in section 5.1, triplet sets can also be obtained by generating multiple trees and producing a triplet set by taking the union of all triplets represented by at least one of the trees.

In the following, we do this for random trees, while examining how many fractional solutions there are with triplets extracted from sets of two to six trees. The results are shown in table 5.1 below.

Most notably, with pairs of trees, the LP-relaxation produces integral solutions for all instances.

Lemma 5.3.1. *Given triplets extracted from two trees, both on the same set of taxa, it is always optimal to simply choose the triplets belonging to a single tree.*

# trees	fractional solutions (%)
2	0
3	9
4	37
5	73
6	95

Table 5.1: The amount of fractional solutions in instances with triplets extracted from multiple trees. The number of trees is shown in the left column and the average number (each based on > 1000 individual instances) of fractional solutions on the right. make sure table is not between lemma and proof in the final version. will hopefully resolve itself

Proof. Selecting a single tree achieves $\binom{n}{3}$ triplets and this cannot be improved upon since a single tree cannot display more than $\binom{n}{3}$ triplets. This holds for the LP-relaxation as well due to constraint 4.2:s for every subset of decision variables on three taxa, the sum of their values is exactly one and consequently the objective function value can be at most a third of the number of decision variables: $\frac{1}{3}3\binom{n}{3} = \binom{n}{3}$. □

Note that, although fractional optima cannot be beneficial in this case, we can still find fractional optima that are as good as the integral optimum. Assume a caterpillar tree C and a companion caterpillar tree C^* such that for every subset of three leaves $\{a, b, c\}$, $ab|c$ occurs in C and $cb|a$ occurs in C^*. If an integer solution is required, then it is always optimal to simply choose one of the trees, set all its triplet variables to 1 and set the triplet variables of the other tree to 0. But we can also create a fractional equally good solution by setting all triplets from both trees to 0.5. Of course, it is not clear whether this fractional optimum is a corner point solution or a convex combination of corner points. In the latter case, the simplex method will not consider the fractional solution.

To further verify the observation that no fractional solutions are present with triplets from pairs of trees, the LP-relaxation was applied to pairs of trees from the well-known Poaceae dataset [26]. Here again, all solutions were completely integral.

However, the results on triplets from more than two trees are of interest as well, as only small amounts of fractional variables can be observed for sets of three trees, and the amount of fractional variables increases rapidly for sets of more than three trees (see table 5.1).

It might be that the number of underlying trees τ, i.e. the minimum number of trees such that every triplet in the input is consistent with at least one tree, is a good predictor for the amount of fractional values in the solution. A

high amount of corruption might then be correlated with large τ. However, when examining how many underlying trees there are within the given range of taxa and corruption, not a single instance requires more than $\tau = 3$. It is known that τ grows quite slowly from [27]. In this case the growth might be further restrained due to the input being minimally dense. To summarise, the behaviour observed in table 5.1 does not generalise to other data and thus τ only has limited use in trying to understand the behaviour of the (I)LP.

probably need some kind of glue paragraph here. and a couple more examples of real data

In [28], several conflicting phylogenies of parasitic protists in the apicomplexan group are given (see figure 5.8). Specifically, there are six trees, each on the same eight taxa. 92 unique triplets can be extracted from the trees, while only $\binom{8}{3} = 56$ can be selected. When the LP-relaxation is applied to the 92 triplets, the resulting solution is completely integral.

5.4 Examination of the 40% Phase Change

The abrupt phase change at roughly 40% corruption warrants an explanation. A simple explanation might be that corruption is correlated with the gap between the ILP optimum and LP-relaxation optimum. To investigate this, we define the *instance integrality gap* as the ratio of $\frac{OPT_{LP-relax}}{OPT_{ILP}}$ for any given instance.

As figure 5.9 shows, it is not strictly the case that fractional solutions start to be beneficial at 40% corruption, as there are instances with instance integrality gaps > 1 with less than 40% corruption, as well as instances with instance integrality gaps $= 1$ with more than 40% corruption. Nevertheless, there is a clear tendency for larger integrality gaps as the amount of corruption increases.

Consider the measurement of *recovered triplets*, defined as the optimal objective function value of the (I)LP divided by the number of input triplets. We will show in due course that this has important explanatory power.

Figure 5.10 (top) indicates that, for less than 50% corruption, the ILP can always recover roughly all uncorrupted triplets (i.e. those triplets that were extracted from the original tree and *not* subsequently corrupted), while the corrupted triplets are (mostly) excluded. This trend does not continue for higher amounts of corruption, as around 50% of triplets are recovered even for 60% corruption.

The bottom graph of figure 5.10 holds the key to understanding the 40% phenomenon, as it shows the (possibly fractional) amount of triplets the LP-relaxation can recover. In contrast to the ILP, the LP-relaxation can recover 60% of all input triplets for any amount of corruption. The graphs of ILP and

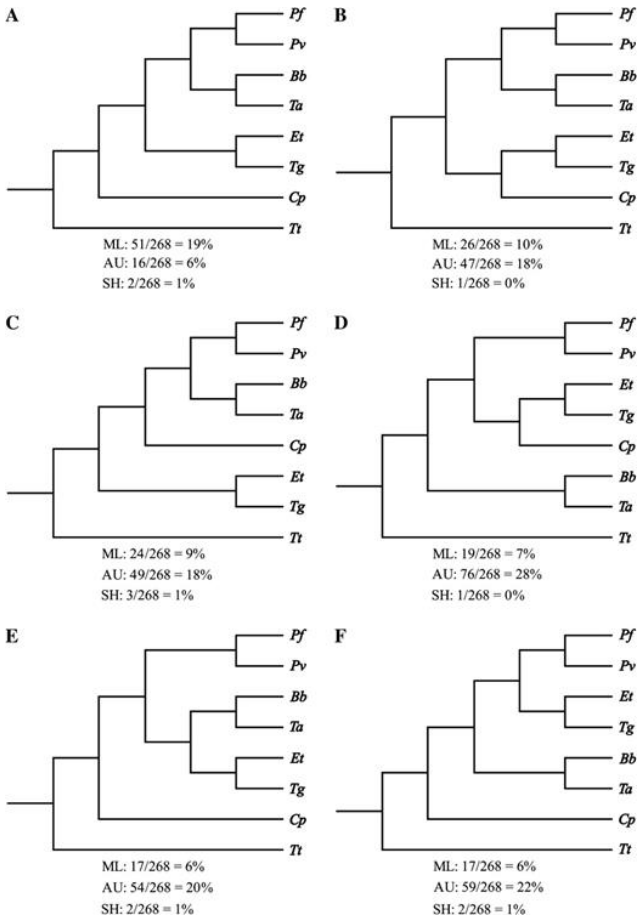

Figure 5.8: Conflicting phylogenies of parasitic protists in the apicomplexan group. Source: [28].

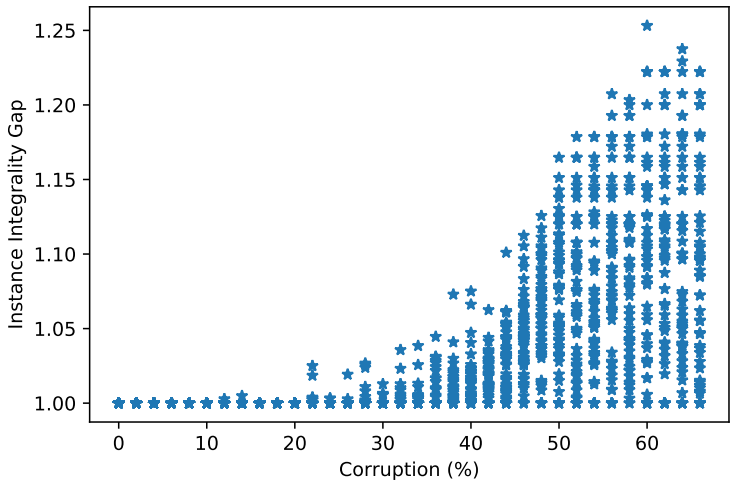

Figure 5.9: The instance integrality gap with varying amounts of corruption. Each datapoint represents one individual instance. The shown data features instances with between 6 and 11 taxa.

LP-relaxation diverge at exactly 40%.

Generally, in figure 5.10 it appears that often the best thing the ILP can do is to simply include every remaining triplet from the original tree but none of the corrupted triplets, meaning that it will recover roughly a fraction of $1 - c$ triplets. The LP-relaxation on the other hand always seems to be able to recover at least 60% of triplets, even with more than 40% corruption. Re: "if you have 40% corruption, you would expect that 60% survive, but of course it can differ due to variance". But the variance is very one-sided (and its the wrong side..). To be clear: 40% corruption means I corrupt a subset of size $0.4|T|$. The instances below the lines in figure 5.10 are probably due to rounding (we can't corrupt a fractional number of triplets), but for the instances above the line it's more complex than that. For example, I can construct a tree on 4 taxa, extract the 4 triplets from it, corrupt 2 of them in a specific way and get out a different valid tree, i.e. 50% corruption but 100% recovered triplets.

Theorem 5.4.1. *The LP-relaxation can always recover at least* 60% *of triplets.*

Proof. If fractional values are permitted, we can always simply set all triplet variables in the objective function to 0.6 and each of their alternate topologies to 0.2. All constraints will be satisfied this way: The first constraint will always equal 1. The other constraints have two possibilities:
(1) $0.6 + 0.6 - x \leq 1$, where x can only be 0.6 or 0.2, in either the case the

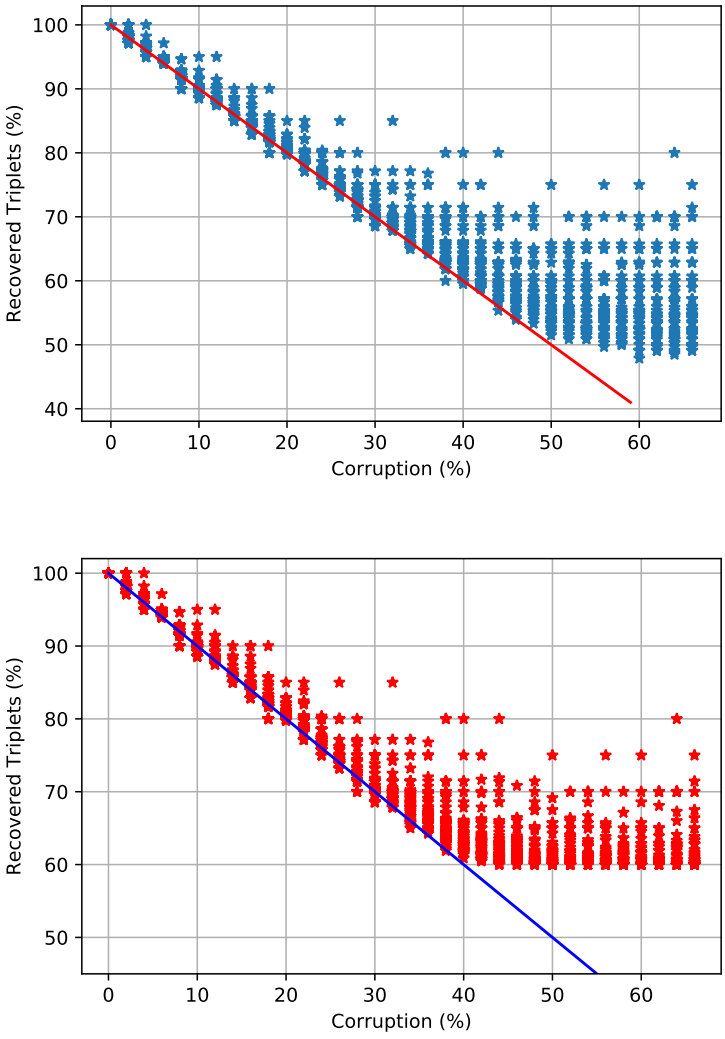

Figure 5.10: Corruption vs recovered triplets for the ILP (top) and the LP-relaxation (bottom). The red (top) and blue (bottom) lines show the function $f(x) = 100 - x$.

constraint is satisfied,
(2) $0.6 + 0.2 - x \leq 1$, where the antecedent is ≤ 1, so it does not matter what the consequent is. □

The above solution strategy will henceforth be referred to as the (0.6/0.2/0.2)-strategy.

The following shows that it is not possible to set the variables in the objective function > 0.6 with this kind of strategy. The (0.6/0.2/0.2)-strategy can be generalised to a $\left(v/\frac{1-v}{2}/\frac{1-v}{2}\right)$-strategy, i.e. all triplets in the objective function are set to some value v and each of their alternate topologies is set to $\frac{1-v}{2}$. This strategy is subject to the constraint $2v - \frac{1-v}{2} \leq 1$ and $2v - \frac{1-v}{2} \leq 1 \iff 4v - (1-v) \leq 2 \iff 5v \leq 3 \iff v \leq \frac{3}{5}$, i.e. setting $v > 0.6$ results in infeasible solutions.

If all triplets in the objective function are set to 0.6, the resulting solution does not contain any "signal", i.e. we have not gained any new information in regards to which triplets to choose. However, we know that the ILP optimum is *at most* $0.6|T|$, which means that, in this specific situation, the 3-approximation from [12] becomes a $\frac{0.6}{\frac{1}{3}} = 1.8$-approximation. An approximation algorithm based on a rounding scheme of the LP-relaxation might therefore still be possible, i.e. the rounding scheme could be applied when the LP-relaxation solution does contain a "signal", and when it does not, the 3-approximation from [12] could be applied. Since the 3-approximation becomes 1.8-approximation in this case, the overall algorithm could be better than a 3-approximation if the rounding scheme can achieve a solution that is within less than a third of the optimum. Of course, it is not clear how to design a rounding scheme that leads to even feasible solutions in practice.

The absence of a signal in the LP-relaxation solution also has consequences for the performance of the ILP, which likely cannot work effectively if the bounds resulting from the solved LPs are all dominated by the (0.6/0.2/0.2)-strategy and thus very similar. although I'm not sure about this, since setting some variables to 0 or 1 already disrupts the (0.6/0.2/0.2)-scheme. It depends if this remains local or not

$\left(v/\frac{1-v}{2}/\frac{1-v}{2}\right)$-strategies can be applied to triplet sets that are not minimally dense, since this minimal density only affects the objective function, i.e. optimality. The feasibility is not affected since minimal density does not affect the set of taxa, which is the only thing determining the constraints, so the proof of theorem 5.4.1 still holds. It should be emphasised that the resulting solution will have an objective function value $< v|T|$, since at least one objective function variable out of some subset $\{t_{ij|k}, t_{jk|i}, t_{ik|j}\}$ will have to take a value of $\frac{1-v}{2}$ (due to the fact that the instance is not minimally dense).

To characterise the solution quality of triplet sets that are not minimally dense, T can be divided into three non-intersecting subsets T_1, T_2, and T_3. T_1 is the subset of T that *is* minimally dense, i.e. contains exactly one triplet for every subset of three taxa. T_2 (T_3) is the subset of T that contains exactly two (three) triplets for every subset of three taxa. Then, the solution qualities of T_1, T_2, and T_3 are given by $v|T_1|$, $v\frac{1}{2}|T_2| + \frac{1-v}{2}\frac{1}{2}|T_2|$, and $v\frac{1}{3}|T_3| + \frac{1-v}{2}\frac{2}{3}|T_3|$ respectively. The solution quality of T is thus given by

$$v|T_1| + v\frac{1}{2}|T_2| + \frac{1-v}{2}\frac{1}{2}|T_2| + v\frac{1}{3}|T_3| + \frac{1-v}{2}\frac{2}{3}|T_3| \tag{5.1}$$

or

$$\sum_{i=1}^{3} v\frac{1}{i}|T_i| + \frac{1-v}{2}\left(1-\frac{1}{i}\right)|T_i| \tag{5.2}$$

Note that in the worst case, $T = T_3$, and the quality of the solution is $0.6\frac{1}{3}|T| + \frac{1-v}{2}\frac{2}{3}|T| = \frac{1}{3}|T|$. In fact, *any* feasible solution will have an objective function value of $\frac{1}{3}|T|$ due to constraint 4.2, i.e. in this case the solution qualities of the ILP and LP-relaxation are always equal.

The (0.6/0.2/0.2)-strategy is not optimal for T_2, since it will only achieve a solution quality of $(0.6 + 0.2)|T_2| = 0.8|T_2|$, while $1.0|T_2|$ can be achieved by setting each of the two triplets (out of three on each set of three taxa) that are in the objective function to 0.5. This (0.5/0.5/0.0)-strategy is always optimal since $2\binom{n}{3}$ triplets are in the objective function and by setting each of them to 0.5 we achieve a solution value of $\binom{n}{3}$, which we know is the best we can do.

Such a (0.5/0.5/0.0)-strategy can be mixed with an optimal $(\frac{1}{3}/\frac{1}{3}/\frac{1}{3})$-strategy on T_3 without risking infeasibility because the largest value is 0.5 and thus the largest value of any antecedent is $0.5 + 0.5 \leq 1$. Constraint 4.2 does not cause infeasibilities when mixing strategies in this way because, by definition, T_1, T_2, and T_3 are non-intersecting and thus constraints of type 4.2 that concern T_2 never intersect with constraints that concern T_3. The (0.5/0.5/0.0)-strategy cannot be mixed with a (0.6/0.2/0.2) strategy on T_1 since antecedents of the form $0.6 + 0.6$ may cause issues, as the lowest value the consequent can take is 0 and $0.6 + 0.6 - 0 > 1$. To summarise, an optimal solution can be achieved on $T_2 \cup T_3$ with a combination of straightforward strategies, on $T_1 \cup T_3$ a straightforward strategy is optimal under certain conditions, and it is not clear how an optimal solution can be achieved on $T_1 \cup T_2$.

The reasoning behind the occurrence of fractional solutions when the data is minimally dense and at least 60% of triplets can be recovered has been elaborated on above, but as figure 5.11 shows, fractional solutions do occur even when more than 60% of triplets can be recovered. The above simply explains

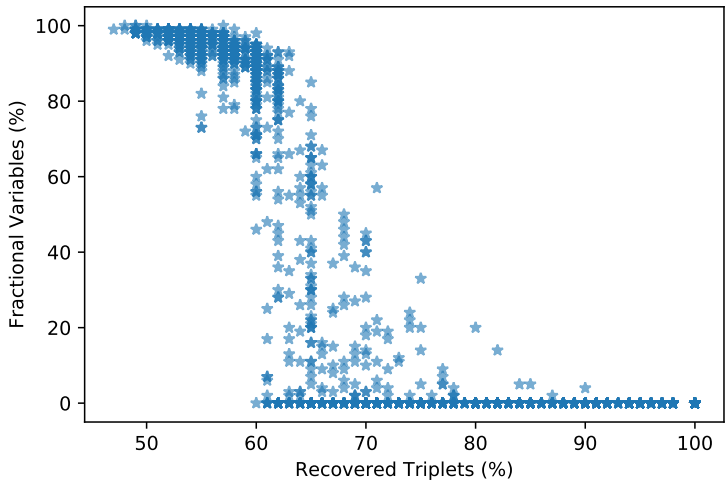

Figure 5.11: The horizontal axis denotes the ratio of the optimum ILP solution value and the number of input triplets, while the vertical axis represents the proportion of fractional values in the solution of the LP-relaxation. Each dot represents a single instance.

the systematic fractionality above 40% corruption, but there may be other (possibly more instance-specific) reasons for the occurrence of fractional solutions ideally give specific example here, but not too important.

5.5 Fixing Subsets of Variables

It might of interest to examine whether constraining only some subset of variables to be integer is enough to find overall integral solutions. This would be especially promising if such subsets were small and frequent. In the following, we check if such subsets exist.

Preliminary tests show that, if constraining a subset to be integral leads to an overall integral solution, the variables in the constrained subset always turn out to be 1. Due to this observation, for the remainder of this section, instead of constraining subsets of triplets to be integer, they are constrained, or *fixed*, to be 1. This is associated with a noticeable speedup since the resulting program is an ordinary linear program instead of a mixed integer linear program.

For 41% (out of 847) of instances with fractional solutions, there was at least one triplet that, if set to 1, resulted in a completely integral solution. For

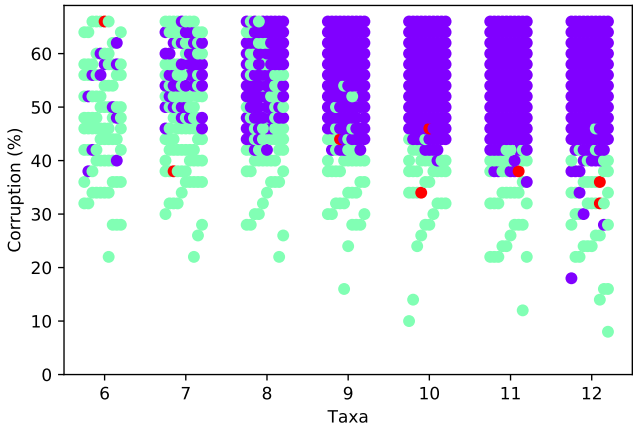

Figure 5.12: Results from setting single triplets to 1. If there is no triplet that, when set to 1, results in an integral solution, the instance is marked in violet. If there is at least one triplet that, when set to 1, results in an integral *and* optimal solution, the instance is marked in green. If there is no triplet that results in an optimal solution, but there is one that results in an integral one, the instance is marked in red. The instances are roughly centered on their respective number of taxa on the horizontal axis, but slightly offset to avoid complete overlap with other instances.

instances with fractional solutions on less than 40% corruption, 93% can be turned integral by setting some (single) triplet to 1. Figure 5.12 shows that this strategy of fixing single triplets becomes less effective with increasing numbers of taxa. This is especially the case for instances with $> 40\%$ corruption.

Naturally, if fixing a single variable does not yield an integral solution, fixing a size 2 subset of variables might accomplish this. Figure 5.13 shows that this is often the case, although the strategy again becomes less effective with increasing numbers of taxa.

Performing a thorough test for size 3 subsets could not be done in reasonable time, and even for size 2 subsets the search had to be restricted to ≤ 10 taxa.

5.5.1 Enumeration & Sampling

Naturally, the question whether we can take advantage of this algorithmically arises. While it is feasible to simply go through all sets of size 1, it is not guaranteed that one of them will produce an integral solution. Enumerating all sets of size 2 or 3, especially with higher numbers of taxa, is less reasonable since, in practice, this will usually take more time than simply solving the ILP in the first

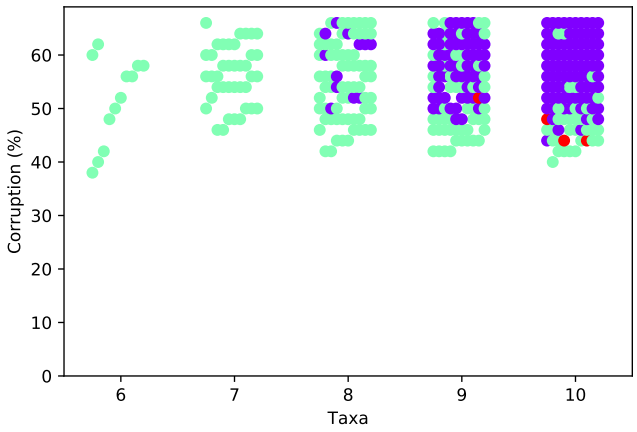

Figure 5.13: Results from setting pairs of triplets to 1, where a single triplet was not enough. Colours have the same meaning as in figure 5.12.

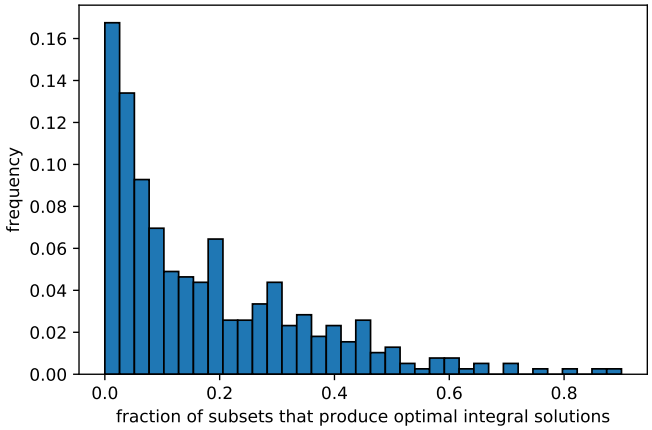

Figure 5.14: Fraction of size 1 subsets that, when fixed, produce integral solutions (horizontal axis) and in how many instances each fraction occurs (vertical axis).

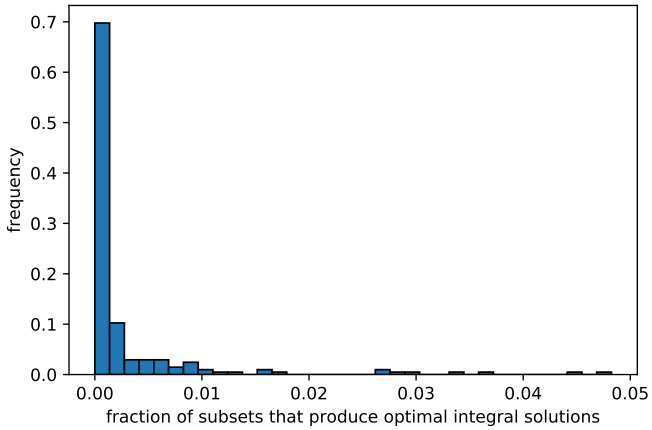

Figure 5.15: Fraction of size 2 subsets that, when fixed, produce integral solutions (horizontal axis) and in how many instances each fraction occurs (vertical axis).

place. Note that, such an enumeration procedure might still be of theoretical interest if it could be guaranteed that a better than 3-approximation can be achieved by enumerating all subsets of some fixed size c, since enumerating all subsets of size c takes only polynomial time ($O(n^{3c})$).

One possible recourse to derive a practical algorithm from the above might be to restrict the subsets to be explored to variables that drop out fractional in the LP-relaxation. But this is problematic for two reasons: (1) In instances with high corruption, which tend to be the more problematic ones, almost every variable drops out fractional (see figure 5.3), (2) there are cases where setting a variable to 1 which had value of 0 in the solution to the LP-relaxation results in a completely integral solution, but 0 is an *integral* value, i.e. we would not consider these cases if we restricted our focus solely to fractional variables.

Since enumeration is not always doable in reasonable time, the next best alternative might lie in a sampling based approach. More specifically, we could sample subsets of triplets (with certain sizes) and set these triplets to 1 with the hope that some small number of samples is enough to find an optimal integral solution.

However, there are instances where there is only a single variable (or a single pair of variables in the case of size 2 subsets), that, if set to 1, results in an integral solution. In other words, we need to try every single variable to guarantee finding a solution. From a worst-case analysis perspective this is not very promising, and even in an average case analysis this is problematic, since it

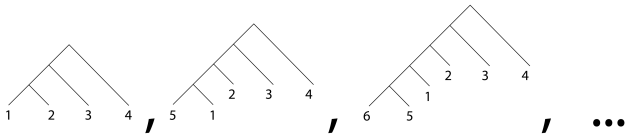

Figure 5.16

is often the case that only a small number of subsets can be utilised successfully (see figures 5.14 and 5.15).

5.5.2 Quality of Integral Solutions

Once we have found an integral solution, we have no sensible way of evaluating whether it is an optimal solution or not. This may be acceptable if we can show that the resulting solution is never too far away from the optimum. So the question is: if we set a variable to 1 that would have been 0 in every optimal solution, how far away from the optimal solution will we at most be?

It is clear that the resulting solution quality can be at most $OPT - 1$. In the following, we show that the resulting solution can be worse than $OPT - 1$. Assume we have a tree on 4 leaves labelled $\{1, 2, 3, 4\}$. We can extract 4 triplets from this tree: $T = \{12|3,\ 12|4,\ 13|4,\ 23|4\}$. Now suppose that we corrupt $12|4$ into $24|1$. The remaining triplets $\{12|3,\ 13|4,\ 23|4\}$ still form a consistent subset of size 3, or $|T|-1$. The maximum size of any consistent set that contains $24|1$ is 2. That is, if we were to set the $24|1$ variable to 1, we could at best select a consistent set that with a quality of $OPT - 1$.

However, the difference between optimal and actual solution can be > 1, if further leaves are added to the tree. The first tree in figure 5.16 shows our original tree. The second tree is the original one with an additional taxon. If we set $t_{24|1} = 0$, we can still get a consistent subset of size $|T| - 1 = 9$. But if we set $t_{24|1} = 1$, we can now at most get a solution quality of $OPT - 3$ because we have additional triplets $51|2$ and $51|4$ which are not consistent with $24|1$.

We can keep adding taxa in this way so that the quality of the $t_{24|1} = 0$-solution stays at $|T| - 1$, while the quality of the $t_{24|1} = 1$-solution keeps (in comparison) going down, i.e. if we add m additional taxa, the quality of the $t_{24|1} = 1$-solution will be $OPT - (2m + 1)$. Since OPT grows a lot more quickly than $2m + 1$, $\lim_{m\to\infty} \frac{OPT}{OPT-(2m+1)} = 1$, i.e. we cannot construct arbitrarily worse instances this way, but we have shown that the quality of such a solution can be worse than $OPT - 1$. Of course, it would be desirable to show what the worst case is.

Chapter 6

Variations on the ILP

6.1 Redundant Constraints

The ILP formulation in [23], whose results prompted the research presented here, is not exactly the same as the one given in chapter 4:

$$Max \sum_{ij|k \in T} t_{ij|k} \tag{6.1}$$

$$t_{ij|k} + t_{ik|j} + t_{jk|i} = 1 \qquad \forall i, j, k \in N \tag{6.2}$$

$$t_{ij|k} + t_{jk|l} - t_{ik|l} \leq 1 \qquad \forall i, j, k, l \in N \tag{6.3}$$

$$t_{ij|k} + t_{jk|l} - t_{ij|l} \leq 1 \qquad \forall i, j, k, l \in N \tag{6.4}$$

$$t_{ij|l} + t_{jk|l} - t_{ik|l} \leq 1 \qquad \forall i, j, k, l \in N \tag{6.5}$$

$$t_{ij|l} + t_{ik|l} - t_{jk|l} \leq 1 \qquad \forall i, j, k, l \in N \tag{6.6}$$

$$t_{ij|k} \in \{0, 1\} \qquad \forall i, j, k \in N \tag{6.7}$$

where

$$i \neq j \neq k \neq l \tag{6.8}$$

Specifically, it has two additional constraints in equations 6.5 and 6.6. The rest of the formulation is identical.

These additional two constraints are not strictly necessary. To start, equations 6.5 and 6.6 produce identical constraints, as we can convert between the two equations by simply switching i and j, which is permissible since all permutations of a given set $\{i, j, k, l\}$ are considered when generating the constraints.

Further, equation 6.5 is implied by equations 6.2 to 6.4. Suppose we have some constraint $t_{a,b|d} + t_{b,c|d} - t_{a,c|d} \leq 1$ generated by equation 6.5, i.e. if $t_{a,b|d}$

and $t_{b,c|d}$ are true, then $t_{a,c|d}$ must also be true. For $i = a, j = b, k = c, l = d$, equation 6.3 generates $t_{a,b|c} + t_{b,c|d} - t_{a,c|d} \leq 1$. For $i = c, j = b, k = a, l = d$, equation 6.3 generates $t_{b,c|a} + t_{b,a|d} - t_{a,c|d} \leq 1$. For $i = a, j = c, k = b, l = d$, equation 6.4 generates $t_{a,c|b} + t_{c,b|d} - t_{a,c|d} \leq 1$. Note that the second part of the antecedent in each of the three constraints above ($t_{b,c|d}$, $t_{b,a|d}$, $t_{c,b|d}$) is assumed to be true. Each of the three constraints contains a different triplet in the first part of the antecedent, but all on the same three leaves. Thus, one of these three triplets must be true due to 6.2, meaning that the (full) antecedent will be true in exactly one of these constraints and the consequent ($t_{a,c|d}$) must therefore also be true.

Since the additional two constraints are already implied by the original set of constraints, they cannot change the feasible region of the ILP. It may, however, be the case that they somehow restrict the the feasible region of the LP-relaxation without affecting the feasible region of the ILP. In this case, it may advantageous in terms of solving speed to include the additional constraints.

It is well-known that an ILP formulation whose LP-relaxation closely approximates the convex hull of the integral solutions will run more quickly than a (mathematically equivalent) ILP formulation with a weaker LP-relaxation. This is why it can sometimes be useful to include constraints in the ILP formulation that are already implied by others. However, in this case, there is no noticeable performance increase, as is shown in figure 6.1.

6.2 Preventing the Spread of Fractionality

In the LP-relaxation, constraint 4.2 is likely to cause unnecessary fractional values, because a fractional value for any one of $\{t_{ij|k}, t_{ik|j}, t_{jk|i}\}$ means that at least one other variable in the set must take a fractional value. This constraint could therefore potentially facilitate chain reactions of fractionality. One strategy to mitigate this might be to relax the strict equality $=$ to $\leq$, i.e. $t_{ij|k} + t_{ik|j} + t_{jk|i} \leq 1$. While this stops the propagation of fractional values, it allows for all three variables to have a value of 0, which implies that i, j, k are siblings and binary trees cannot be guaranteed anymore. However, the decision to make i, j, k siblings is not enforced in any of the other constraints. For example, the ILP solver might decide to set $t_{1,2|3} = t_{2,3|1} = t_{1,3|2} = 0$, but also to set some other variable $t_{1,*|2} = 1$, which is a contradiction since the lca(1,*) would have to be a proper descendent of the lca(1,2).

Another way to stop the propagation of fractional values, which is potentially better because it does not change the feasible region of the ILP, may be to replace constraint 4.2 by two new constraints:

$$t_{ij|k} + t_{ik|j} + t_{jk|i} \leq 1 \tag{6.9}$$

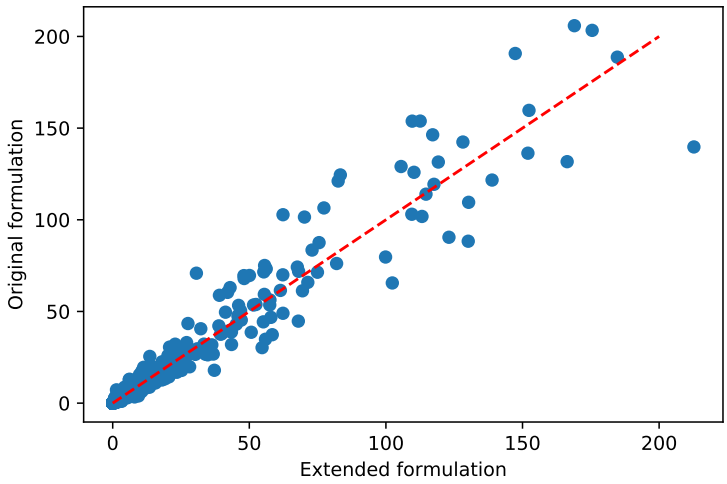

Figure 6.1: Running time in seconds for the original (presented in chapter 4) and the extended formulations. The dashed red line is the diagonal $f(x) = x$. If a point is below (above) the line, the instance was solved faster using the original formulation (extended formulation). The instances are the same as those used in section 5.2.

$$t_{ij|k} + t_{ik|j} + t_{jk|i} > 0 \tag{6.10}$$

The strict inequality above causes the feasible region to be an open set, which is not permissible. A workaround is the introduction of some variable ϵ close to 0, such that:

$$t_{ij|k} + t_{ik|j} + t_{jk|i} \geq \epsilon \tag{6.11}$$

However, when this change is implemented and compared against the results in earlier sections, no differences in fractionality can be perceived.

Chapter 7

Conclusion

We examined an ILP and its relaxation for MaxRTC in order to combine several phylogenetic trees into a supertree. The fact that this ILP is fast and the solutions of its relaxation fairly integral under certain conditions spawned the investigation presented in this thesis. We have verified these observations and determined the integrality/fractionality of the LP-relaxation to be related to the level of inconsistency, measured by the amount of corruption, in a given instance. Specifically, when less then 60% of triplets can be recovered from the input, the LP-relaxation will systematically yield fractional solutions. This is due to a fractional strategy that is always feasible and, on minimally dense triplet sets, always achieves a solution that recovers 60% of the input triplets. This fractional strategy assigns the same value to every decision variable in the objective function, i.e. it is not informative with respect to which triplets should be included in the final supertree. Still, we have shown that this does not rule out the possibility of a better than 3-approximation algorithm based on some rounding scheme of the LP-relaxation. Finding such a rounding scheme would thus be of great interest for future research.

Fractional solutions can also occur when more than 60% of triplets can be recovered, but this does not happen very frequently. In practice, it might be an option to simply run the LP-relaxation, and check whether the resulting solution is integral. If the results from the experiments presented here transfer to real data, and real data instances can be assumed to be the equivalent of $< 40\%$ corruption instances, this will usually be the case. In case the resulting solution is not integral, the instance could still be solved by means of one of the existing approximation algorithms. For any specific instance, the solution could then potentially be shown to be better than a 3-approximation through the bound on the quality of the optimum solution we obtained with the LP-relaxation. Note again that, of the instances based on real data examined in this thesis, the LP-relaxation produced integral solutions for all of them. An approach like this is still subject to the limitation that the (I)LP does not scale well in terms of the number of decision variables and constraints (see section 4).

A further avenue of deriving a practical algorithm from the results presented in this thesis is the idea of fixing the values of some limited subset of decision variables as described in section 5.5. Issues with this idea are two-fold: (1) To guarantee a certain level of solution quality for such an algorithm, it would have to be determined how far away from the optimum solutions that are integral but not optimal can be. While the proportion of subsets that, if fixed, result in integral but not optimal solutions is small, they are still relevant for a worst-case analysis of the solution quality. The worst case solution quality needs to be determined to be able to guarantee that the solution is at most some specific amount away from the optimum. (2) The speed with which any approach based on fixing subsets can yield solutions. While section 5.5.1 elaborated on the issues with sampling based approaches, it might be possible to come up with some (possibly heuristic) procedure to determine which subsets should be fixed. While speed is naturally important for practical algorithms, a theoretical algorithm that runs in polynomial-time, but is practically prohibitive, would still be of interest. Especially if such an algorithm could be guaranteed to yield better than 3-approximations, it would prove polynomial-time better than 3-approximations are theoretically possible.

A possibly fruitful future research direction which has not been touched upon in this thesis may lie in investigating whether more advanced polyhedral techniques such as lift-and-project (whereby relaxations of ILPs are structurally and systematically strengthened) can offer any benefits.

Bibliography

[1] T. Brown, *Genomes*. Garland Science, 2 ed., 2002.

[2] B. S. Chang and M. J. Donoghue, "Recreating ancestral proteins," *Trends in Ecology & Evolution*, vol. 15, no. 3, pp. 109–114, 2000.

[3] R. M. Bush, "Predicting the evolution of human influenza a," *Science*, vol. 286, no. 5446, pp. 1921–1925, 1999.

[4] G. B. Dantzig, "A history of scientific computing," ch. Origins of the Simplex Method, pp. 141–151, New York, NY, USA: ACM, 1990.

[5] F. A. Potra and S. J. Wright, "Interior-point methods," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 281–302, 2000.

[6] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, pp. 85–103, Springer US, 1972.

[7] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, p. 497, jul 1960.

[8] A. J. Hoffman and J. B. Kruskal, "Integral boundary points of convex polyhedra," in *50 Years of Integer Programming 1958-2008*, pp. 49–76, Springer Berlin Heidelberg, 2009.

[9] V. V. Vazirani, *Approximation Algorithms*. Berlin, Heidelberg: Springer-Verlag, 2001.

[10] A. V. Aho, Y. Sagiv, T. G. Szymanski, and J. D. Ullman, "Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions," *SIAM Journal on Computing*, vol. 10, pp. 405–421, aug 1981.

[11] B. Y. Wu, "Constructing the maximum consensus tree from rooted triples," *Journal of Combinatorial Optimization*, vol. 8, no. 1, pp. 29–39, 2004.

[12] L. Gasieniec, J. Jansson, A. Lingas, and A. Östlin, "On the complexity of constructing evolutionary trees," *Journal of Combinatorial Optimization*, vol. 3, no. 2/3, pp. 183–197, 1999.

[13] J. Byrka, P. Gawrychowski, K. T. Huber, and S. Kelk, "Worst-case optimal approximation algorithms for maximizing triplet consistency within phylogenetic networks," *Journal of Discrete Algorithms*, vol. 8, no. 1, pp. 65–75, 2010.

[14] J. Byrka, S. Guillemot, and J. Jansson, "New results on optimizing rooted triplets consistency," in *Algorithms and Computation*, pp. 484–495, Springer Berlin Heidelberg, 2008.

[15] C. Semple and M. Steel, "A supertree method for rooted trees," *Discrete Applied Mathematics*, vol. 105, no. 1-3, pp. 147–158, 2000.

[16] G. Sevillya, Z. Frenkel, and S. Snir, "Triplet MaxCut: a new toolkit for rooted supertree," *Methods in Ecology and Evolution*, vol. 7, no. 11, pp. 1359–1365, 2016.

[17] S. Snir and S. Rao, "Using max cut to enhance rooted trees consistency," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 3, no. 4, pp. 323–333, 2006.

[18] M. Steel, "The complexity of reconstructing trees from qualitative characters and subtrees," *Journal of Classification*, vol. 9, pp. 91–116, jan 1992.

[19] J. Weyer-Menkhoff, C. Devauchelle, A. Grossmann, and S. Grünewald, "Integer linear programming as a tool for constructing trees from quartet data," *Computational Biology and Chemistry*, vol. 29, no. 3, pp. 196–203, 2005.

[20] A. Chester, R. Dondi, and A. Wirth, "Resolving rooted triplet inconsistency by dissolving multigraphs," in *Theory and Applications of Models of Computation* (T.-H. H. Chan, L. C. Lau, and L. Trevisan, eds.), (Berlin, Heidelberg), pp. 260–271, Springer Berlin Heidelberg, 2013.

[21] S. Guillemot and M. Mnich, "Kernel and fast algorithm for dense triplet inconsistency," *Theoretical Computer Science*, vol. 494, pp. 134–143, 2013.

[22] C. E. Hinchliff, S. A. Smith, J. F. Allman, J. G. Burleigh, R. Chaudhary, L. M. Coghill, K. A. Crandall, J. Deng, B. T. Drew, R. Gazis, K. Gude, D. S. Hibbett, L. A. Katz, H. D. Laughinghouse, E. J. McTavish, P. E. Midford, C. L. Owen, R. H. Ree, J. A. Rees, D. E. Soltis, T. Williams, and K. A. Cranston, "Synthesis of phylogeny and taxonomy into a comprehensive tree of life," *Proceedings of the National Academy of Sciences*, vol. 112, no. 41, pp. 12764–12769, 2015.

[23] A. Derks, "Integer linear programming for constructing phylogenetic trees from rooted triplets (bachelors thesis)," 2012.

[24] "CPLEX (12.8)." `http://www.ibm.com/software/integration/optimization/cplex-optimizer/`.

[25] "GNU Linear Programming Kit (4.32)." `http://www.gnu.org/software/glpk/glpk.html`.

[26] Grass Phylogeny Working Group, N. P. Barker, L. G. Clark, J. I. Davis, M. R. Duvall, G. F. Guala, C. Hsiao, E. A. Kellogg, and H. P. Linder, "Phylogeny and subfamilial classification of the grasses (poaceae)," *Annals of the Missouri Botanical Garden*, vol. 88, no. 3, p. 373, 2001.

[27] L. van Iersel, S. Kelk, N. Lekić, and S. Linz, "Satisfying ternary permutation constraints by multiple linear orders or phylogenetic trees," *Theoretical Computer Science*, vol. 609, pp. 1–21, 2016.

[28] C.-H. Kuo, J. P. Wares, and J. C. Kissinger, "The apicomplexan whole-genome phylogeny: An analysis of incongruence among gene trees," *Molecular Biology and Evolution*, vol. 25, no. 12, pp. 2689–2698, 2008.