Master Thesis

Towards a Better Approximation Algorithm for MaxRTC (not really though)

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 12, 2018

Abstract

insert abstract

Contents

Chapter 1

Introduction

define taxa somehwere
this is a placeholder, dont bother reading:

Phylogenetic trees are a graph-based representation of the evolutionary relationships between biological species (more abstractly called taxa). 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, instead of trying to preserve all the relationships found in the input set of trees, a supertree can be constructed by selecting the maximum number of relationships that do not lead to inconsistencies. This requires a prior deconstruction of the input trees into atomic relationships called rooted triplets. Rooted triplets are rooted phylogenetic trees with exactly three leaves x, y, z. A rooted triplet x,y—z is consistent with a tree if the lowest common ancestor of x and y is a proper descendent of the lowest common ancestor of x and z.

Given a set of rooted triplets, finding the supertree that maximizes the the number of consistent rooted triplets is called the "maximum rooted triplets consistency problem" (or MaxRTC). Solving MaxRTC exactly is an NP-hard problem (Jansson, 2001) , but a 3-approximation algorithm exists (Byrka et al., 2010). Although there is no evidence that the approximation ratio cannot be improved, an algorithm that does better than a 3-approximation has not been found yet.

An LP-relaxation of a natural Integer Linear Programming (ILP) formulation to MaxRTC yields, when applied to well-behaved inputs derived from real data, suprisingly 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.

Investigating whether alternative ILP formulations or more advanced polyhedral techniques such as lift-and-project (whereby relaxations of ILPs are structurally and systematically strengthened) offer any benefits may lead to further insights as well.

The overall goal is to design a polynomial-time approximation algorithm with ratio c ¡ 3 (i.e. in all cases it produces a solution to MaxRTC that is strictly less than three times as large as the optimum solution)

maybe mention phylogenetic networks and quartet methods in the introduction

Chapter 2

Preliminaries

2.1 Phylogenetic Trees and Rooted Triplets

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* 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. 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. 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 $L(t) = L^*$.

2.2 (Maximum) Rooted Triplet Consistency

Given a set of triplets, we would like build a tree that represent 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 exists. If not, output null.

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:

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \leq b \\ \text{and} & x \geq 0 \end{array}$$

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 and third lines, the constraints, describe the set of solutions that are permissible, or feasible. The solution space defined by the constraints is a convex polyhedron. The second line is problem specific, while the third line, the *non-negativity constraints* are generally a part of every program.

Linear programs can be solved by the well-known simplex method, originally proposed by Dantzig [1]. Eventhough the simplex method can take exponential time to terminate in the worst case, and interior-point methods [2], 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 polytope, 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. 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 [3].

Due to the NP-hardness of the problem, finding optima probably requires enumerating solutions in some way. While the amount 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 [4], 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-relaxation 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 [5].

LP-relaxation 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 usually 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, the approximation ratio usually cannot be better than the integrality gap [6].

Chapter 3

Related Work

RTC can be solved exactly in polynomial time as evidenced by Aho's algorithm [7], which runs in $O(kn)$ time.

MaxRTC on the other hand has been shown to be NP-hard [8]. An exact algorithm proposed by Wu [8] takes exponential time

A polynomial time algorithm for MaxRTC proposed in [9] achieves a 3-approximation by building a caterpillar tree that is always consistent with at least a third of the input triplets (explain how this works). 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 [10] (based on a derandomization strategy for labelling a given tree topology) and in [11] (based on a bottom up approach). [10] 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 [12] which applies min cut on a graph that represents the triplets inconsistency, or [13], as well as methods which do not work with triplets at all, such as [14].

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 [15]. An ILP similar to the one examined in the following chapters has been considered for quartet methods in [16].

It may be of importance what assumptions can be made about the given set of triplets. For instance, [17] points out that a polynomial-time approximation scheme is possible if the set of triplets is dense.

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 if there is an absence of local conflicts, as demonstrated in [18].

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 $a, b|c$ and $b, c|d$ are in the set, then so are $a, b|d$ and $a, c|d$ [18].

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 all possible triplets on taxa set N. On a high level, the ILP selects a feasible tree such that this tree is consistent with a maximum

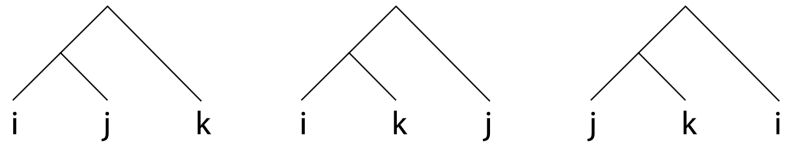

Figure 4.1: The three different topologies on a set of three taxa mentioned in constraint 4.2

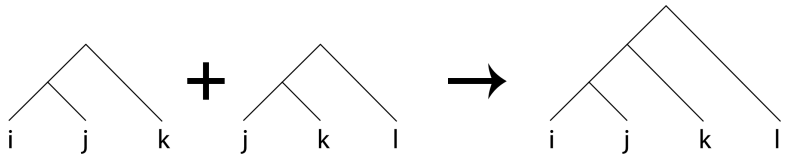

Figure 4.2: Intuition behind constraints 4.3 and 4.4

cardinality subset of triplets in T.

Three different triplet topologies are possible for every subset of three leaves. These topologies are mutually exclusive, meaning that at most one of them can be selected (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 to **??** model the logical implications described in [18], e.g. $t_{ij|k} \wedge t_{jk|l} \rightarrow t_{ik|l}$ is modelled by equation 4.3. 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 constraint will be referred to as the antecedent, while the third one will be referred to as the consequent.

todo: insert picture and explanations for logical implications

4.1 Thesis Objectives

Some work in [19] suggests that the ILP may be surprisingly fast and its relaxation surprisingly integral under certain conditions. The goal of the succeeding chapters is to replicate these results and examine the following questions:

1. Under which conditions will the LP-relaxation yield integral solutions?
2. Can these conditions be used to improve current approximation ratios of MaxRTC?
3. Are other ILP formulations possible and/or beneficial?

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 all triplets from every tree 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 a 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 topolgies have 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.

If there is no corruption, every triplet in the objective function can be given a value of 1 without violating any constraint. This solution cannot be further improved 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. The optimal integral solution is therefore always better than any fractional solution.

Unless otherwise specified, the experiments presented here are performed using the well-known commercial (I)LP solver CPLEX [20].

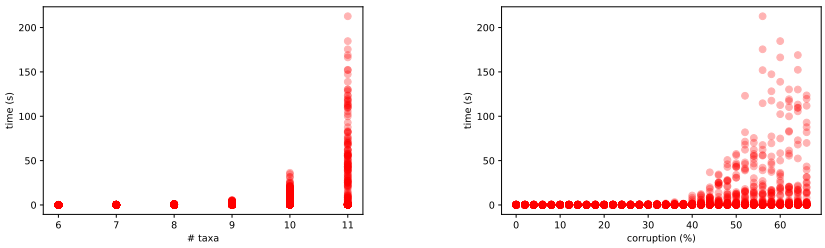

Figure 5.1: CPLEX solving time of the ILP measured in seconds as a function of taxa (left) and corruption (right).

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 instance 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. As can be seen in figure 5.1, 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?

As figure 5.2 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, this cannot be the case as the constraints depend only on the number of taxa and the only thing changed by corruption is the objective function. Clearly, different levels of corruption, and thus different objective functions, lead to differences in the integrality of instances.

Since CPLEX is a fairly sophisticated (I)LP solver, it might be that some

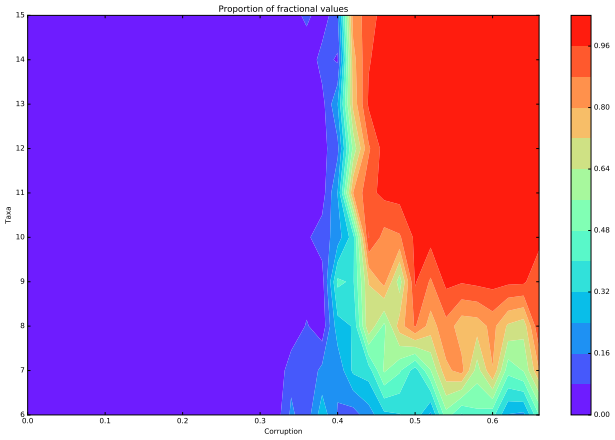

Figure 5.2: Average proportion of fractional values as a function of the amount of corruption and the number of taxa. The vertical axis represents the amount of corruption and the horizontal axis represents the number of taxa. The colour signifies the relative amount of fractional values, with blue meaning little or no fractional values and red meaning almost complete fractionality.

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 [21], which is generally more primitive than CPLEX. The results, given in figure 5.3, are not exactly the same as the ones obtained through CPLEX, but the sudden phase change at 40% corruption is present here as well.

The kind of fractional values that occur in LP-relaxation solutions are illustrated in figure 5.4. Most prominently, at 40% corruption, the dominant values change from 0 and 1 to 0.2 and 0.6.

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. As figure 5.5 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. (maybe look at how upper and lower bounds evolve)

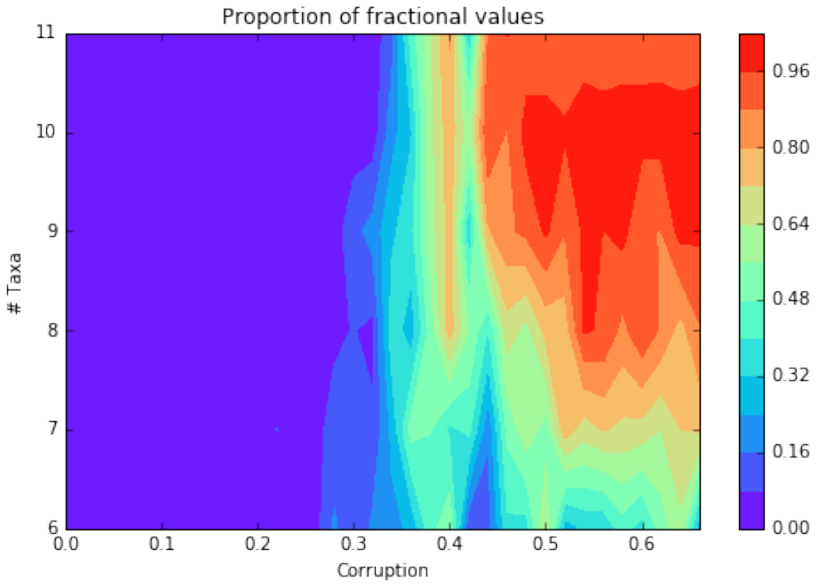

Figure 5.3: Heatmap as in figure 5.2, but here the data is generated using GLPK instead of CPLEX.

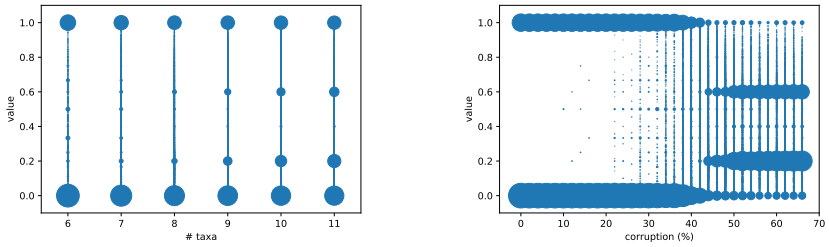

Figure 5.4: Each circle represents a unique fractional value, given on the vertical axis. The size of each circle is determined by the frequency of the respective value. The figure on the left displays these values as a function of the number of taxa, while the figure on the right shows these values as a function of the amount of corruption.

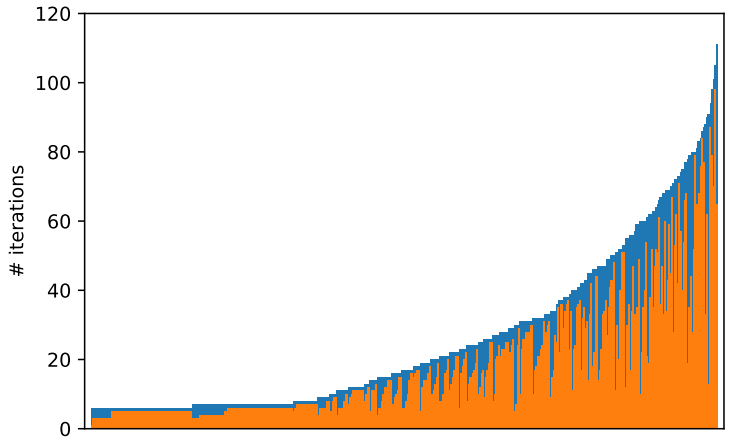

Figure 5.5: Comparison of the total number of iterations and and how many iterations are executed after the optimal solution is found. Each bar represents one instance, with the blue part denoting the number of iterations after the optimal solution has been found and the orange part denoting the number 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.

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 extracting the triplets from all of them.

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.

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

Most notably, with pairs of trees, the LP-relaxation produces integral solutions for all instances. Only small amounts of fractional variables can be observed for sets of three trees, and the amount of fractional variables rapidly increases for sets of more than three trees.

The observation that no integral solutions are present with triplets from pairs of trees can also be replicated on the well-known Poaceae dataset [22].

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 would then simply be another way of saying that τ is large. 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 [23] [24]. In this case the growth might be further restrained due to the input being minimally dense.

If we have triplets from two trees on exactly the same label set, it is always optimal to simply choose one of the trees since it is not possible to have more than one triplet for each set of three labels.
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 $\{a, b, c\}$, $a, b|c$ occurs in C and $c, b|a$ 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.

Why is fractionality so low on three trees?

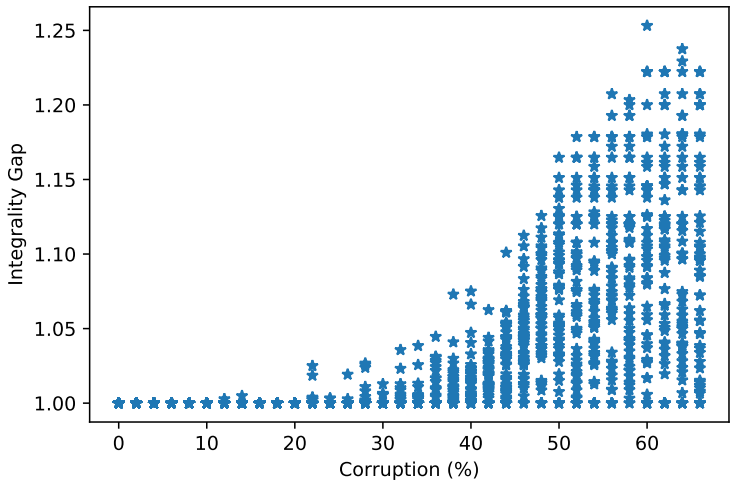

Figure 5.6: Is it fair to call this integrality gap? (since its not the maximum possible gap, but rather the actual gap for individual instances..)

[25] gives several conflicting phylogenies of parasitic protists in the apicomplexan group. 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 Why 40% (working title)

The abrupt phase change at roughly 40% corruption warrants an explanation. A possible hypothesis may be that fractional solutions start to be beneficial, i.e. better than the optimum integer solution at 40% corruption.

As figure 5.6 shows, this is not strictly the case, i.e. there are instances with integrality gaps > 1 with less than 40% corruption, as well as instances with integrality gap $= 1$ with more than 40% corruption. Nevertheless, there is a clear tendency for larger integrality gaps as the amount of corruption increases.

The notion of corruption only makes sense in light of artificial data. Even in artificial data, it acts as a proxy for something else, namely a measure of inconsistency of the input triplets. Another way to measure this may be the number of triplets that are *recoverable* from the input, where the number of *recovered triplets* is defined as the optimal objective function value of the ILP divided by the number of input triplets.
Figure 5.7 (left) indicates that, for less than 50% corruption, the ILP solution

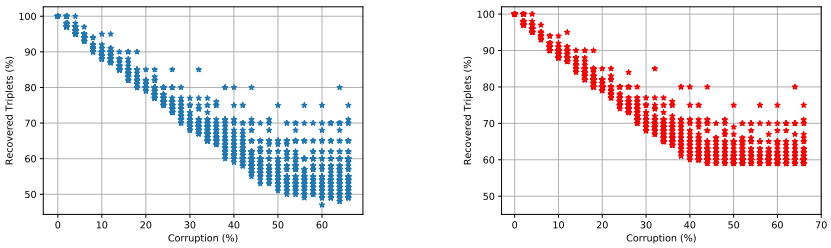

Figure 5.7: Corruption vs recovered triplets for the ILP (left) and the LP-relaxation (right).

always roughly consists of all uncorrupted triplets, 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 right graph of figure 5.7 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 LP-relaxation diverge at exactly 40%.

Generally, in figure 5.7 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. I have a feeling this might be difficult to formalise, because figure 5.7 suggests that the ILP solution is $\geq 1 - c$ and not $= 1 - c$ (which would be a lot more convenient for the argument..). Here is a sketch: I can show that, if we corrupt a single triplet on a tree with 4 leaves, then we can only recover $|T| - 1$ triplets. Bigger trees are just trees on 4 leaves with extra stuff attached, so this should generalise. Leap of faith: If corrupting 1 triplet means that we can recover $|T| - 1$ triplets, then corrupting $p\%$ of triplets means we can recover $(1 - p\%)|T|$ triplets. Problem: this isn't generally true, we can often corrupt certain sets of triplets in a certain way and do better than this. But it is true in a worst-case sense.

The LP-relaxation can always recover 60% of triplets because, 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 (this will be referred to as the (0.6/0.2/0.2)-strategy). All constraints will be satisfied this way: The first constraint will always $= 1$, 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 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 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". An approximation algorithm (with any guarantee of producing good results) based on a rounding scheme of the LP-relaxation therefore seems unlikely. 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

It should be emphasised that, while $\left(v/\frac{1-v}{2}/\frac{1-v}{2}\right)$-strategies can be applied to triplet sets that are not minimally dense, the resulting solution will have an objective function value $< v|T|$. Of course, it is not clear which variable should get which value, but this can be handled by random tie breaking.

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.

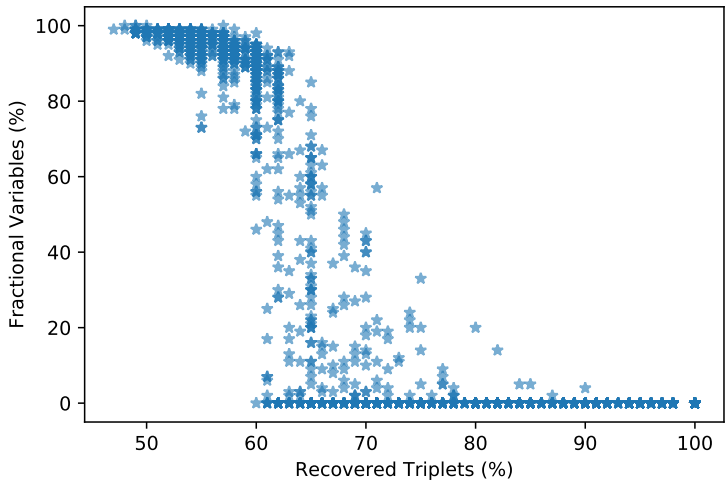

Figure 5.8: The horizontal axis denotes the ratio of the optimum ILP solution value and the number of input triplets, while the vertical axis represents the proportion of fractional values in the solution of the LP-relaxation. Each dot represents a single instance.

The $(0.6/0.2/0.2)$-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 and can be mixed with an optimal $(\frac{1}{3}/\frac{1}{3}/\frac{1}{3})$-strategy on T_3 without risking infeasibility. It cannot be mixed with a $(0.6/0.2/0.2)$ strategy on T_1 since constraints of the form $0.6 + 0.6 - 0.0 \leq 1$ may cause issues. 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 5.8 shows, fractional solutions do occur even when more than 60% of triplets can be recovered. The above simply explains the systematic fractionality above 40% corruption, but there may be other (possibly more individualistic) reasons for the occurrence of fractional solutions.

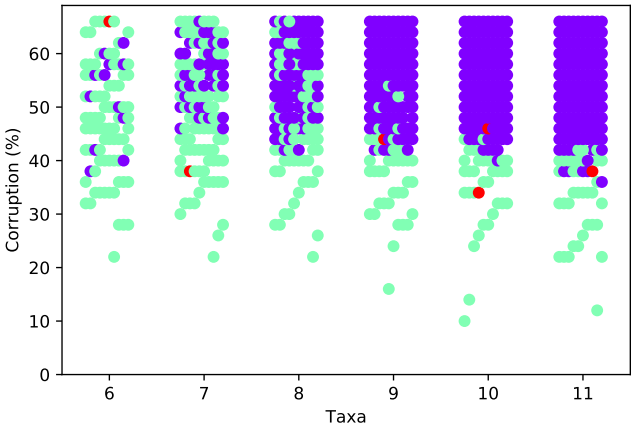

Figure 5.9: Results from setting single triplets = 1. If there is no triplet that, when set = 1, results in an integral solution, the instance is marked in violet. If there is at least one triplet that, when set = 1, results in an integral *and* optimal solution, the instance is marked in green. If there 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.

5.5 Fixing Subsets of Variables

Considering the low levels of fractionality of instances with low to moderate corruption, it may be that the fractionality of a solution is driven by some relatively small subset of all triplets. If this is the case, then simply constraining this subset to be integer (thus arriving at a mixed integer linear programming problem) may be enough to find overall integral solutions. In the following, we check if such subsets exists.

Preliminary results suggest 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. To take advantage of this, for the remainder of this section, instead of constraining subsets of triplets to be integer, they are constrained to be 1 instead, which means the resulting problems are ordinary linear programs instead of mixed integer linear programs.

For 41% (out of 847) of instances with fractional solutions, there was at least one triplet that, if set = 1, resulted in a completely integral solution. For instances with fractional solutions on less than 40% corruption, 93% can be turned integral by setting some (single) triplet = 1.

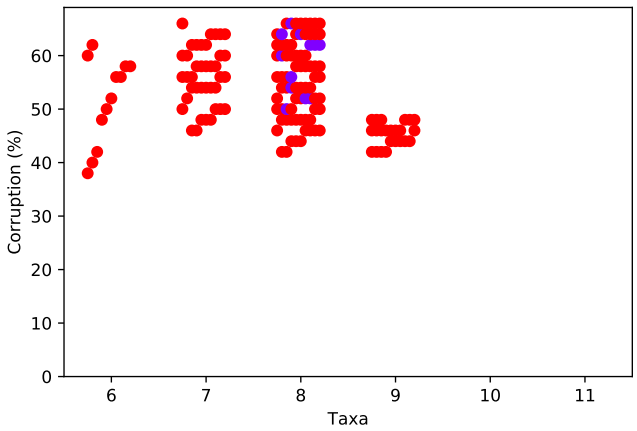

Figure 5.10: Results from setting pairs of triplets = 1, where a single triplet was not enough. If there is no pair that, when set = 1, results in an integral solution, the instance is marked in violet. If there is, then it is marked red. add higher taxa. Colours should be consistent with the first graph..

insert results

- sometimes setting a variable = 1 that dropped out as 0 in the relaxation results in integral solutions (though probably not optimal integral)
- sometimes there is only a single variable (or a single pair of variables in the case of size 2 subsets), that, if set = 1, results in an integral solution. (which doesn't sound great if we're thinking of some kind of sampling algorithm)

5.5.1 Enumeration & Sampling

Naturally, the question whether we can take advantage of this 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 the number of subsets is given by $\binom{\binom{n}{3}}{3}$. In practice, this will usually take more time than simply solving the ILP in the first place.

One possible recourse 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, (2) there are cases where setting a variable = 1 which was 0 in the solution to the LP-relaxation results in a completely integral solution.

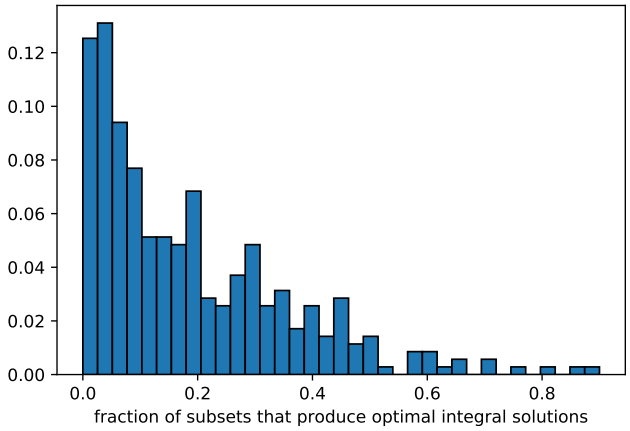

Figure 5.11

Since enumeration is not always doable in reasonable time, the next best alternative might lie in a sampling based approach. I.e. we sample subsets of triplets (with certain sizes) and set these triplets = 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 = 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 encouraging, although the average case analysis might still be promising.

5.5.2 Quality Guarantees on 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 the optimal solution, how far away from the optimal solution will we at most be?

Not sure whether to keep the rest of this section. I originally thought I could construct an arbitrarily bad solution, but that does not actually work
Assume we have a tree on 4 leaves labelled $\{1, 2, 3, 4\}$. We can extract 4 triplets from this tree: $T = \{1, 2|3,\ 1, 2|4,\ 1, 3|4,\ 2, 3|4\}$. Now suppose that we corrupt $1, 2|4$ into $2, 4|1$. The remaining triplets $\{1, 2|3,\ 1, 3|4,\ 2, 3|4\}$ still form a consistent subset of size 3, or $|T| - 1$. The maximum size of any consistent set that

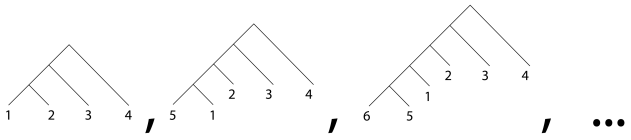

Figure 5.12

contains $2,4|1$ is 2. That is, if we were to set the $2,4|1$ 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 5.12 shows our original tree. The second tree is the original one with an additional taxon. If we set $t_{2,4|1} = 0$, we can still get a consistent subset of size $|T| - 1 = 9$. But if we set $t_{2,4|1} = 1$, we can now at most get a solution quality of $OPT - 3$ because we have additional triplets $5,1|2$ and $5,1|4$ which are not consistent with $2,4|1$.

We can keep adding taxa in this way so that the quality of the $t_{2,4|1} = 0$-solution stays at $|T| - 1$, while the quality of the $t_{2,4|1} = 1$-solution keeps (in comparison) going down, i.e. if we add m additional taxa, the quality of the $t_{2,4|1} = 1$-solution will be $OPT - (2m + 1)$. This means we have no way of guaranteeing that a given integral solution found by a subset-fixing procedure will be close to the optimal solution.

5.6 Results in search of a section

- mention somewhere that integrality has nothing to do with total unimodularity.... Note that the constraints depend only on the set of taxa, but not on the set of triplets. Corruption therefore only changes the objective function, while the constraints stay constant.

- correlation between fractionality magnitude and how far fractionality spreads

- caterpillar trees

- could look at how 3-approximation behaves, does it have trouble with the same instances the ILP does?

We have looked at the results of minimally dense triplet sets. Overdetermined, meaning the data is dense, but not minimally dense, i.e. there can be more than one triplet for each subset of three leaves. Underdetermined: starting with a minimally dense set, we delete some triplets so that the density property

is lost.

Overdetermined would expect to be harder. Underdetermined, sparsity, so conflicts are less likely.

Chapter 6

Variations on the ILP

6.1 Redundant Constraints

The ILP formulation in [19], 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 $t_{a,b|d}$

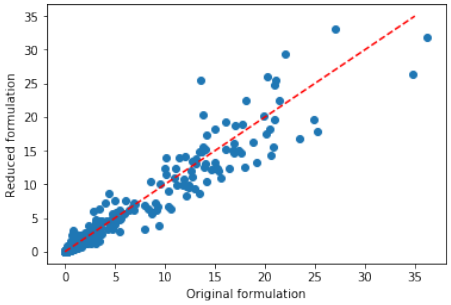

Figure 6.1: Running time in seconds for the original and the reduced formulations. The dashed red line is the diagonal $f(x) = x$. If a point is below (above) the line, the instance was solved faster using the reduced formulation (original formulation). The instances are the same as those used in section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

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.

Figure 6.1 compares the time the ILP takes to solve instances using both the original and the extended formulation. The additional constraints do not appear to improve solving times.

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.

A potentially better way to stop the propagation of fractional values may be to replace constraint 4.2 by two new constraints:

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

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

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 section, no differences in fractionality can be perceived.

6.3 A Lower Bound on The Solution Quality

A trivial lower bound on the solution quality can be obtained by using the observation that a tree with at least a third of the input triplets can always be found [**?**]. Solutions with lower quality could simply be made infeasible by adding the constraint:

$$\sum_{t_{ij|k} \in T} t_{ij|k} \geq \frac{|T|}{3} \tag{6.12}$$

which could potentially eliminate some subset of the solution space. This does not turn out to be helpful, since the initial feasible solution the solver finds is always better than $\frac{|T|}{3}$. not sure if this is even worth mentioning.

Chapter 7

Conclusion

presented a phenomenon where relaxation was surprisingly integral up to 40% corruption. it turns out that there is a fractional strategy that beats any integral strategy after 40% however, using this strategy every triplet in the objective function gets the same value, meaning that it is not helpful in deciding which triplets in the objective function should be selected (since we cannot select all of them)

Bibliography

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

[2] 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.

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

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

[5] 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.

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

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] M. A. Ragan, "Phylogenetic inference based on matrix representation of trees," *Molecular Phylogenetics and Evolution*, vol. 1, no. 1, pp. 53–58, 1992.

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

[16] 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.

[17] 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.

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

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

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

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

[22] G. P. W. 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.

[23] 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.

[24] R. Atkins, "Betweenness and nonbetweenness," 2016.

[25] 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.