Exact ILP Solutions for Phylogenetic Minimum Flip Problems

Markus Chimani*
Algorithm Engineering group,
Friedrich-Schiller-Universität
07743 Jena, Germany
markus.chimani@uni-jena.de

Sven Rahmann
Bioinformatics, Computer Science XI, TU Dortmund
44221 Dortmund, Germany
sven.rahmann@tu-dortmunde.de

Sebastian Böcker
Chair for Bioinformatics,
Friedrich-Schiller-Universität
07743 Jena, Germany
sebastian.boecker@uni-jena.de

ABSTRACT

In computational phylogenetics, the problem of constructing a consensus tree or supertree of a given set of rooted input trees can be formalized in different ways. We consider the Minimum Flip Consensus Tree and Minimum Flip Supertree problem, where input trees are transformed into a 0/1/?-matrix, such that each row represents a taxon, and each column represents a subtree membership. For the consensus tree problem, all input trees contain the same set of taxa, and no ?-entries occur. For the supertree problem, the input trees may contain different subsets of the taxa, and unrepresented taxa are coded with ?-entries. In both cases, the goal is to find a perfect phylogeny for the input matrix requiring a minimum number of 0/1-flips, i.e., matrix entry corrections. Both optimization problems are NP-hard.

We present the first efficient Integer Linear Programming (ILP) formulations for both problems, using three distinct characterizations of a perfect phylogeny. Although these three formulations seem to differ considerably at first glance, we show that they are in fact polytope-wise equivalent. Introducing a novel column generation scheme, it turns out that the simplest, purely combinatorial formulation is the most efficient one in practice. Using our framework, it is possible to find exact solutions for instances with ∼100 taxa.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*Integer Programming*; J.3 [**Computer Applications**]: Life and Medical Sciences—*Biology and Genetics*

General Terms

Algorithms, Experimentation, Theory

Keywords

Perfect phylogeny, Consensus Tree, Supertree, Integer Linear Programming, Error correction

*Funding of MC's juniorprofessorship by the Carl-Zeiss-Foundation is gratefully acknowledged.

ACM-BCB 2010, Niagara Falls, NY, USA

1. INTRODUCTION

Relations about the ancestry of organisms can be represented as phylogenetic trees, that is, rooted trees where each leaf corresponds to a group of organisms, called *taxon*, whereas inner vertices represent hypothetical last common ancestors of the taxa in the corresponding subtree.

For supertree problems, the input is a set of phylogenetic trees with overlapping taxon sets, and the task is to assemble these input trees into a larger supertree, describing the evolutionary relationship between all taxa. If the input trees do not contain any contradictory information, then constructing a supertree is easy [13]. The major challenge of supertree methods is to deal with incompatible data in a reasonable way [15]. If all input trees share the same set of taxa, the supertree is called a consensus tree [10].

We can encode the input trees using a matrix: Each (non-trivial) subtree of each input tree defines a binary *character* with states $\{0, 1\}$, namely the property of being part of that subtree. If an input tree does not contain information about a particular taxon i, the character states j of that input tree are marked as unknown (?). If the input trees do not contain contradictory information, then we can resolve all ?-entries such that the resulting matrix is a perfect phylogeny. In case of incompatible input trees, we can formalize the supertree construction problem as an optimization problem, penalizing for deviations from the perfect phylogeny. The most prominent example for such an optimization formulation is Matrix Representation with Parsimony (MRP) [14]. Unfortunately, the underlying Maximum Parsimony problem is NP-hard [6].

Minimum flip models, which we consider in this article, take the view that the input is an erroneous version of the true matrix. Therefore, we seek the minimum number of "flips" (corrections) necessary to transform the input matrix into a matrix consistent with a perfect phylogeny. This problem is known to be NP-hard even if all input trees share the same set of taxa [5]. A straightforward fixed-parameter algorithm for the consensus tree problem, as well as an (exponential) branch-and-bound algorithm for the supertree problem were proposed in [2], but were shown to be practically infeasible for more than 20 taxa. Chen *et al.* [3] intro-

duced a branch-swapping heuristic for the supertree problem, and evaluations indicate that this approach is on par with the "gold standard" MRP, and superior to other approaches for supertree construction. Recently, more efficient fixed-parameter algorithms for the consensus tree problem have been developed [1, 11], but it is unclear how to extend them for the supertree problem.

In this paper, we show that reasonably large instances of both the minimum flip consensus tree and supertree problem can often be solved exactly within minutes, using carefully engineered (mixed) integer linear programming (ILP) formulations. The paper is also meant to show the key algorithm engineering steps to obtain a practically feasible exact solving strategy. Our method is a huge improvement over the currently best exact methods for the problem [2], increasing the number of taxa we can process by an order of magnitude. For our evaluations, we use existing simulated data [3]; refer to that article for an evaluation of the minimum flip model against other supertree models. Using simulated data has the advantage that the "true" phylogenetic tree is known and can be used to evaluate results, different from the usual situation in phylogenetic analysis.

2. PRELIMINARIES

We assume that there are m taxa and n characters (or *features*), and each taxon i may possess or not possess each feature j, which is represented by a binary $m \times n$ matrix W. In the perfect phylogeny model studied extensively by Gusfield [9] and others [7], we assume that there exists an ancestral species that possesses none of the features (represented by a row of zeros). Speciation occurs as a tree-like process such that each state turns from 0 to 1 only once in the tree: We may say that the feature is "invented" along the corresponding edge. In the perfect phylogeny model, an invented feature never disappears and is never invented twice. We precisely define a perfect phylogeny as follows.

DEFINITION 1 (PERFECT PHYLOGENY). *An $m \times n$ perfect phylogeny is a binary rooted tree with m leaves whose vertices (root, inner nodes, and leaves) and edges are labeled with subsets of $\{1, \ldots, n\}$ such that*

- *the root is labeled with the empty set,*
- *the labels of different edges are disjoint,*
- *for each edge $u \xrightarrow{e} v$, where u is labeled U, e is labeled E, and v is labeled V, we have $V = U \dot\cup E$.*

The labels of the set of leaves of a perfect phylogeny P can then be expressed as a binary $m \times n$ matrix $W(P)$, where each row corresponds to a leaf label. We say that a binary matrix X admits *a perfect phylogeny if there is a perfect phylogeny P with $W(P) = X$. In this case, X is also called* perfect phylogeny matrix.

Given a binary matrix X, we may ask whether X admits a perfect phylogeny.

LEMMA 1 (SEE [4]). *Let $X \in \{0,1\}^{m \times n}$ and let $I(j) := \{i : X_{ij} = 1\}$ be the set of 1-indices in column j. Let $G(X)$ be the bipartite graph on the vertices $t_1, \ldots, t_m$ and $c_1, \ldots, c_n$, representing rows and columns, respectively, with undirected edge $\{t_i, c_j\}$ if and only if $X_{ij} = 1$. The following statements are equivalent:*

1. *X is a perfect phylogeny matrix.*
2. *For each pair of columns j and j', we have $I(j) \subset I(j')$ or $I(j') \subset I(j)$ or $I(j) \cap I(j') = \emptyset$.*
3. *$G(X)$ contains no induced path of length 4 with taxa nodes as its end points.*

For mnemonic reasons, we may call the forbidden path in $G(X)$ an "M" (or "W" or "Σ"). This becomes clear when drawing the bipartite graph with the partition sets on two distinct horizontal layers, taxa at the bottom and characters at the top. Let us call a bipartite graph without "M"s *M-free*. Then X is a perfect phylogeny matrix if and only if $G(X)$ is M-free.

We can test whether a binary $m \times n$ matrix is a perfect phylogeny matrix in $\Theta(mn)$ time [9] and even construct the corresponding phylogenetic tree in the same time if it exists.

3. MINIMUM FLIP PROBLEMS

We consider two variants of minimum flip problems: *Minimum Flip Consensus Tree* (MFCT) and *Minimum Flip Supertree* (MFST). Consider s rooted trees on the same set of m taxa. In each tree, each inner node v different from the root defines a binary character: the property of being in the subtree below v. If all input trees are binary, then each tree has $m - 2$ inner nodes, and we obtain $n = s(m - 2)$ characters, which do not admit a perfect phylogeny unless all trees are equal. MFST arises in the same context when not all taxa are present in all of the trees; thus, the state (0 or 1) of some taxa is not known and represented by a question mark (?). If the input trees are compatible, we can resolve the ?-entries such that the matrix becomes a perfect phylogeny. In both cases, a straightforward generalization is to ask for a perfect phylogeny that agrees as closely as possible with the input data. Back in 1992, Ragan [14] proposed this encoding of the input trees, and suggested to use Maximum Parsimony as an optimization version of the perfect phylogeny model to deal with incompatible input trees. In 2002, Chen *et al.* [2] introduced the alternative model of flipping to "correct" the input matrix. For the minimum flip model, we can measure "closeness" using the *flip distance*:

DEFINITION 2 (FLIP DISTANCE). *The* flip distance *$d(W, W')$ between matrices $W, W' \in \{0, 1, ?\}^{m \times n}$ is the number of elements that are equal to 0 in one matrix and equal to 1 in the other matrix, or vice versa.*

Note that the flip distance is neither a metric nor a pseudometric. We can now formally define the MFCT and MFST problems.

PROBLEM 1 (MFCT). *Given $W \in \{0,1\}^{m\times n}$, find a perfect phylogeny matrix X minimizing $d(W,X)$.*

PROBLEM 2 (MFST). *Given $W \in \{0,1,?\}^{m\times n}$, find a perfect phylogeny matrix X minimizing $d(W,X)$.*

Note that a perfect phylogeny matrix must not contain ?-entries and that in applications, we always have $m \le n+1$.

4. ILP FORMULATIONS

We describe three alternative (mixed) ILP formulations of the MFCT and MFST problems; their strengths and sizes are compared in the next section.

Let $\mathcal{T} = \{1,\dots,m\}$ and $\mathcal{C} = \{1,\dots,n\}$ be the index sets corresponding to the rows (taxa) and columns (characters) in the input $W \in \{0,1,?\}^{m\times n}$. We define auxiliary constants

$$w_{tc} := \begin{cases} 1 & \text{if } W_{t,c} = 0, \\ 0 & \text{if } W_{t,c} = ?, \\ -1 & \text{if } W_{t,c} = 1, \end{cases} \qquad \text{for } t \in \mathcal{T}, c \in \mathcal{C}, \tag{1}$$

and denote the number of 1s in W by $\mathrm{Ones}(W)$.

The natural variables of the problem are given by an $m \times n$ matrix

$$X = (x_{tc})_{t\in\mathcal{T},c\in\mathcal{C}} \quad \in \{0,1\}^{m\times n} \tag{2}$$

that will represent a solution with $x_{tc} = 1$ if and only if taxon t has character c.

4.1 M-free Graph (MFG)

The simplest formulation uses only the natural variables x_{tc}. Indeed, the number of flips to move from W to X is

$$d(W,X) = \mathrm{Ones}(W) + \sum_{t\in\mathcal{T},c\in\mathcal{C}} w_{tc}\cdot x_{tc}. \tag{3}$$

To describe feasible solutions, recall that X admits a perfect phylogeny iff $G(X)$ is M-free (Lemma 1). Let t_1,t_2,t_3 and c_1,c_2 be distinct vertices of taxa and characters in $G(X)$, respectively, and consider the potential "M" formed by the path (t_1,c_1,t_2,c_2,t_3). To avoid this "M", either at most three of the four edges $(t_1,c_1),(t_2,c_1),(t_2,c_2),(t_3,c_2)$ may be present, or additional edges between those nodes must exist. This gives rise to the "M-constraints"

$$x_{t_1,c_1} + x_{t_2,c_1} + x_{t_2,c_2} + x_{t_3,c_2} \le 3 + x_{t_1,c_2} + x_{t_3,c_1} \tag{4}$$
$$\forall\, t_1,t_2,t_3 \in \mathcal{T} \text{ (pairwise disjoint)}, \quad \forall\, c_1,c_2 \in \mathcal{C},\ c_1 < c_2.$$

THEOREM 1. *Every feasible solution to the ILP*

$$\{\min(3),\ \textit{subject to } (2),(4)\}$$

gives a feasible solution to the MFST and MFCT problem with identical objective value. An optimal solution of the ILP is an optimal solution for these problems.

The proof follows directly from the above discussion and the M-free characterization of Lemma 1. While the number of variables in MFG is minimal, the number $O(m^3n^2)$ of constraints, although polynomial, is too high to be practical. Therefore we look for alternative formulations.

4.2 Comparing Character Pairs (CCP)

Observe that $G(X)$ contains an "M" involving the character pair (c,d) with $c < d$ if and only if there exist taxa t_1, t_2, t_3 such that (a) edge $\{t_1,c\}$ exists, but $\{t_1,d\}$ does not, (b) both edges $\{t_2,c\}$ and $\{t_2,d\}$ exist, and (c) edge $\{t_3,c\}$ does not exist, but $\{t_3,d\}$ exists.

Thus we want (conceptually binary) variables $y'_{tcd} = 1$ iff $x_{tc} = 1$ and $x_{td} = 0$ for all taxa t and characters $c \ne d$, and $z'_{tcd} = 1$ iff $x_{tc} = x_{td} = 1$ for all taxa t and characters $c < d$. To detect the existence of appropriate taxa, we also need the maxima $y''_{cd} = \max_t y'_{tcd}$ for $c \ne d$ and $z''_{cd} = \max_t z'_{tcd}$ for $c < d$. This allows us to forbid "M"s by ensuring that not all three existence conditions are met:

$$y''_{cd} + y''_{dc} + z''_{cd} \le 2 \qquad \forall c < d. \tag{5}$$

To ensure that the y'_{tcd}, y''_{cd}, z'_{tcd}, z''_{cd} variables take the desired values, we first impose the constraints that each of these is bounded between 0 and 1. Then, translating the logic definitions in an obvious way, we demand

$$z'_{tcd} \le x_{tc}\,, \qquad z'_{tcd} \le x_{td}\,, \qquad z'_{tcd} \ge x_{tc} + x_{td} - 1 \quad \forall \text{ defined } z'_{tcd}, \tag{6}$$

$$y'_{tcd} \le x_{tc}\,, \qquad y'_{tcd} \le 1 - x_{td}\,, \qquad y'_{tcd} \ge x_{tc} - x_{td} \quad \forall \text{ defined } y'_{tcd}, \tag{7}$$

$$y''_{cd} \ge y'_{tcd} \quad \forall \text{ defined } y'_{tcd}, \tag{8}$$

$$z''_{cd} \ge z'_{tcd} \quad \forall \text{ defined } z'_{tcd}. \tag{9}$$

We do not need to impose integer constraints explicitly, since any integer solution x will ensure integrality of the auxiliary variables as well.

For the CCP-ILP $\{\min(3)$, subj. to (2), $\mathbf{0} \le y', y'', z', z'' \le \mathbf{1}$, (5)–(9)$\}$ we thus obtain the analogon of Theorem 1 (details omitted).

4.3 Set Inclusion and Disjointness (SID)

We derive another alternative formulation by using condition 2 in Lemma 1: X admits a perfect phylogeny iff at least one of the following conditions holds for each pair of characters or columns $X_{\cdot,c}$ and $X_{\cdot,d}$: (a) $X_{\cdot,c} \le X_{\cdot,d}$; (a') $X_{\cdot,d} \le X_{\cdot,d}$; (b) $X_{\cdot,c} + X_{\cdot,d} \le \mathbf{1}$. The inequalities are to be understood component-wise, and $\mathbf{1}$ is the m-dimensional column vector of ones.

Hence define for all character pairs $c \ne d$ (conceptually binary) variables $y_{cd} = 1$ iff $X_{\cdot,c} \le X_{\cdot,d}$. Furthermore, for all $c < d$ define similarly $z_{cd} = 1$ iff $X_{\cdot,c} + X_{\cdot,d} \le \mathbf{1}$. These properties are ensured by imposing bounds $0 \le y_{cd}, z_{cd} \le 1$ (again, we do not need to impose integer constraints on y and z) and inequalities

$$x_{tc} \le x_{td} + (1 - y_{cd}) \quad \forall\, t \in \mathcal{T}, \forall \text{ defined } y_{cd}, \tag{10}$$

$$x_{tc} + x_{td} \le 2 - z_{cd} \quad \forall\, t \in \mathcal{T}, \forall \text{ defined } z_{cd}, \tag{11}$$

$$1 \le y_{cd} + y_{dc} + z_{cd} \quad \forall \text{ defined } z_{cd}. \tag{12}$$

Here (10) forces y_{cd} to be zero if $X_{\cdot,c} \le X_{\cdot,d}$ is not satisfied, and (11) requires z_{cd} to be zero if the 1-sets of columns c

and d are not disjoint. Finally, (12) ensures that at least one of the three conditions (a),(a'),(b) above is met. Note the systematic difference of this constraint compared to (5).

For the SID-ILP {min (3), subj. to (2), $\mathbf{0} \leq y, z \leq \mathbf{1}$, (10)–(12)} we obtain a theorem analogous to Theorem 1 (details omitted).

5. COMPARISON OF ILP FORMULATIONS

We compare the three above formulations in terms of relative strength (feasible polytope of the natural variables defined by the LP relaxation) and in terms of the size of the LP matrix (number of variables times number of constraints).

5.1 Strength comparison

Let $\mathcal{P}_{\text{MFG}}, \mathcal{P}_{\text{CCP}}, \mathcal{P}_{\text{SID}}$ be the polytopes defined by the feasible fractional solutions to the LP relaxations of MFG, CCP, and SID, respectively, in their respective variable space. To compare them, we consider the natural variables used in the respective objective functions. In other words, we consider the projections $\mathcal{P}^x_{\text{CCP}}$ and $\mathcal{P}^x_{\text{SID}}$ of $\mathcal{P}_{\text{CCP}}$ and $\mathcal{P}_{\text{SID}}$ onto the x-variable space.

THEOREM 2. *The formulations* MFG, CCP, *and* SID *are polytope-wise equally strong, i.e.,* $\mathcal{P}_{\text{MFG}} = \mathcal{P}^x_{\text{CCP}} = \mathcal{P}^x_{\text{SID}}$.

PROOF SKETCH. $\mathcal{P}^x_{\text{SID}} \subset \mathcal{P}_{\text{MFG}}$: Consider any character pair $c, d \in \mathcal{C}$. Combining constraints (10) and (11) into (12) results in all M-constraints (4) for c, d.

$\mathcal{P}_{\text{MFG}} \subset \mathcal{P}^x_{\text{SID}}$: Let $\bar{x}$ be fractional feasible for MFG, i.e., $\bar{x} \in \mathcal{P}_{\text{MFG}}$. We construct assignments $\bar{y}, \bar{z}$ by choosing the component-wise maximal possible values (≤ 1) to satisfy all constraints (10) and (11). It is clear that such values have to exist. Assuming that $(\bar{x}, \bar{y}, \bar{z}) \notin \mathcal{P}_{\text{SID}}$, a constraint (12) for some character pair c, d would have to be violated. But based on our maximality assignments we know that there exist taxa t_1, t_2, t_3 with equality at (10) (w.r.t. $y_{c,d}$ and $y_{d,c}$) and (11) (w.r.t. $z_{c,d}$). Combining these into (12) gives a contradiction.

The proof with respect to CCP is analogous. □

We stress that the values of the objective functions of all three formulations, in particular even when only considering their LP relaxations, are identical. We may further add *transitivity constraints* to SID. Note that $y_{cd} = y_{de} = 1$ implies that $y_{ce} = 1$ as well. Furthermore, if $X_{\cdot,c'} \leq X_{\cdot,c}$ and $X_{\cdot,c} + X_{\cdot,d} \leq \mathbf{1}$, then $X_{\cdot,c'} + X_{\cdot,d} \leq \mathbf{1}$. Expressed linearly,

$$y_{ce} \geq y_{cd} + y_{de} - 1 \quad \forall\, c, d, e \in \mathcal{C} \text{ (pairwise disjoint)},$$
$$z_{ce} \geq y_{cd} + z_{de} - 1 \quad \forall\, c, d, e \in \mathcal{C} \text{ (pairwise disjoint)}.$$

Yet these constraints can be directly generated by the above SID-constraints and, hence, strengthen neither the ILP nor its LP-relaxation.

5.2 Size comparison

Since the strengths of the ILP formulations do not differ with respect to their LP relaxation, we focus on the size of the ILP. It must be understood that size of an ILP, and speed of solving it are not necessarily correlated. But in view of Theorem 2, we believe that size is a reasonable indicator for quality in this case. Table 1 summarizes the key quantities. While both CCP and SID require fewer constraints than MFG, they do so at the expense of additional variables. SID dominates (is smaller than) CCP in terms of size.

In a typical application, we want to compute the consensus tree or supertree of several (say, s) trees with $\Theta(m)$ taxa each, so $n = \Theta(m \cdot s)$. When s is small (and/or constant), it appears that SID is preferable over MFG, as it requires only slightly more variables than MFG, but much fewer constraints.

However, this is a theoretical point of view. In practice, constraints can be added on demand ("Branch-and-Cut", see next section). Also, for desirable input sizes (thousands of taxa, tens or hundreds of thousands of characters), even the number mn of natural variables in the basic MFG formulation exceeds standard computational capabilities. Generating $O(n^2)$ additional variables is out of the question. Therefore it appears more promising to carefully engineer the simple MFG formulation with a branch-and-cut approach and introduce variables only when necessary ("column generation") than to turn to a different formulation. We come back to these points in the concluding discussion.

6. ILP ENGINEERING

We engineer the MFG ILP formulation using simultaneously a branch-and-cut and a column generation technique.

6.1 Branch-and-Cut

To avoid the $O(m^3n^2)$ constraints where possible, we start with an ILP initially containing no constraints other than the variable bounds. This *current model* is augmented in each iteration if necessary.

Iteratively, (a) we obtain a fractional solution to the current model's LP-relaxation (replacing (2) by $0 \leq x_{tc} \leq 1$ for all $t \in \mathcal{T}$, $c \in \mathcal{C}$), (b) we identify violated constraints not yet in the model, and (c) add them to the current model. If we cannot find any violated constraints, we branch by fixing a fractional variable in both possible ways. While in the end we may have added all constraints, in practice a solution can often be found and proven optimal with only a small subset of the constraints. Step (b) is called the *separation step*; see [12], among others, for details on the general Branch-and-Cut framework.

Separation. The separation step can be done in $O(m^3n^2)$ time by checking each of the M-constraints. However, since we may need to carry out many such steps, efficient separation is important. Here we show that an $O(mn^2)$ method exists: We enumerate all character pairs, i.e., matrix columns χ, χ' (with indices $c_1\, c_2$, respectively) of the fractional solution. For each such pair, we compute the following taxon indices, within a single sweep through all taxa in $O(m)$ time:

$$t_1 := \operatorname*{argmax}_{i \in \mathcal{T}} \chi_i - \chi'_i, \qquad t_2 := \operatorname*{argmax}_{i \in \mathcal{T}} \chi_i + \chi'_i,$$
$$t_3 := \operatorname*{argmax}_{i \in \mathcal{T}} \chi'_i - \chi_i.$$

Table 1: Size of ILP formulations of Section 4, assuming $m \leq n$. Note that "# constraints" denotes the number of constraints without the variable bounds.

ILP	# variables	# constraints	size of LP matrix
MFG	$mn = O(mn)$	$3\binom{m}{3}\binom{n}{2} = O(m^3n^2)$	$O(m^4n^3)$
CCP	$mn + (m+1)\cdot 3\binom{n}{2} = O(mn^2)$	$(12m+1)\binom{n}{2} = O(mn^2)$	$O(m^2n^4)$
SID	$mn + 3\binom{n}{2} = O(n^2)$	$(3m+1)\binom{n}{2} = O(mn^2)$	$O(mn^4)$

We construct the M-constraint

$$(x_{t_1,c_1} - x_{t_1,c_2}) + (x_{t_2,c_1} + x_{t_2,c_2}) + (-x_{t_3,c_1} + x_{t_3,c_2}) \leq 3.$$

If it is satisfied, so are all M-constraints for the character pair c_1, c_2; otherwise it is a most violated constraint for this pair because of the choice of the taxon indices, i.e., the left hand side of the constraint maximally exceeds 3 for this character pair. For each character pair, we add the identified constraint (if any) to the current model.

6.2 Column generation

The main bottleneck now is the number mn of variables. Since they directly and only describe the solution matrix, there can be no 0/1-ILP with fewer (conceptual) variables. However, in those cases where few flips are necessary, most entries of X will be identical to the input matrix W. In the case of ?-entries, we may be able to guess their corresponding X-value correctly beforehand. Our goal is to introduce an explicit variable only when we need to flip an entry from W or have incorrectly guessed the value of a ?-entry, or if we need a variable to introduce an M-constraint. This idea is formalized below.

The process of introducing variables as needed (together with their corresponding coefficients in the linear programming matrix) is called *column generation*. Here we detect the necessity of new variables during the separation step and add them accordingly (combinatorial column generation). Under the assumption that the input quality is good, i.e., there is a biologically relevant common supertree in the input data, we can hope that only few variables are required at all, substantially reducing the size of the problem instance and the required running time.

Variable Transformation. Variables not yet in the ILP are assumed to be 0. To use this to our advantage, let $\mathcal{TC}^+ := \{(t,c) \in \mathcal{T} \times \mathcal{C} \mid W_{t,c} = 0\}$, and $\mathcal{TC}^-$ analogously for $W_{t,c} = 1$. For each pair (t,c) in $\mathcal{TC}^-$ we then substitute the variable x_{tc} with its *negated* binary variable $\bar{x}_{tc}$, replacing each occurrence of x_{tc} with $1 - \bar{x}_{tc}$. Hence $\bar{x}_{tc}$ becomes 1 if the matrix entry X_{tc} is flipped from 1 to 0, and our ILP variables no longer encode the solution itself, but the entries that need to be flipped, relative to the input matrix (ignoring the special roles of ?-entries). We can then rewrite the objective function (3) as

$$d(W,X) = \sum_{(t,c)\in\mathcal{TC}^+} x_{tc} + \sum_{(t,c)\in\mathcal{TC}^-} \bar{x}_{tc}.$$

Hence the (presumably infeasible) solution of all 0s corresponds to the input matrix, and can be encoded without any active variable in the model.

Adding Variables. We start the algorithm without any constraints and without any variables. Column generation takes place within the separation step. For any pair $(t,c) \in \mathcal{T} \times \mathcal{C}$, we say that it is *dark-blue* if we already added the variable x_{tc} and *dark-red* if we added its negated variable $\bar{x}_{tc}$. It is *medium-blue* or *medium-red* if we fixed the type of its corresponding variable correspondingly, but did not yet add it to the ILP model as an active variable. Finally, such a pair is *light-blue* or *light-red*, if we implicitly assume that the pair has a variable of the respective type, but we still may change this assumption. Hence, a blue entry without active variable is corresponding to a 0 in the solution matrix X, and a red entry corresponds to 0 for the implicit negated variable, and therefore to a 1 in X.

Initially, all pairs in $\mathcal{TC}^+$ and $\mathcal{TC}^-$ are medium-blue and medium-red, respectively. For the ?-entries, we use a simple heuristic to obtain initial guesses: For each pair of taxa t, t', we count how many character states (disregarding ?-entries) they have in common and call this number their similarity. To choose a color for each ?-entry in taxon t, say at character index c, we find the two most similar taxa t', t'' to t that have a value 0 or 1 at character c (if they exist). If these values agree and equal 0, we color (t,c) light-blue; if they agree and equal 1, we color it light-red. In all other cases, we make a random choice between light-blue and light-red.

During the separation routine, we try to generate variables corresponding to non-?-entries first. Hence we run the separation algorithm twice: In the first pass, we only consider dark or medium colored entries. Whenever we find a violated constraint, we simultaneously instantiate the involved variables corresponding to medium colored entries (if any) and add the identified constraint. If this first pass fails to generate a new constraint, we know that the solution is feasible w.r.t. the already added variables and w.r.t all 0/1 entries in the input matrix. Thus, the objective function might be optimal and it remains to check if the remaining ?-entries can be set in a feasible way.

Therefore we run a second separation pass, also considering the light-colored entries. Yet, instead of adding an identified M-constraint and corresponding variables directly to the ILP, we check if we can circumvent the violation by changing a light-colored entry to its medium opposing color (e.g., from light-blue to medium-red). Also, we set each light-colored entry which is critical not to change in the following (i.e., its inverse color would result in a violated constraint) to its medium color-variant. If the violation cannot be thus circumvented, we add the constraint and the necessary variables, using their current color to decide on whether to use the negated variable. In any case, after separation, we reset all ?-entries to the light variant of their current color.

7. EVALUATION

While the MFCT problem is fixed-parameter tractable in the optimal number of flips [1, 11], for the MFST there exists only a slow (feasible up to 20 taxa in practice) exact method [2] and several heuristics [8, 3], of which [3] empirically represents the state of the art. To investigate the practicality of the engineered MFG approach, we conduct a set of experiments on a regular PC (Intel Core-i7 2.67GHz). Each test instance is run on a single CPU core, in 32-bit mode (restricted to 2GB RAM), and given a maximum of 30 minutes computation time. We use SCIP (`http://scip.zib.de`) as a Branch-and-Cut framework, and ILOG CPLEX 11.2 as the LP solver back-end.

7.1 Test instances

We use instances generated by Eulenstein *et al.* [8] for the evaluation of their heuristics. Model trees were generated for 48 or 96 taxa using a Yule birth process. Sequences were generated along the model tree using Monte Carlo simulations under the Kimura 2-parameter model. Each set of sequences was replicated s times ($s \in \{2, 4, 6, \ldots, 20\}$), and from each replicate, either 25%, 50%, or 75% of the taxa were deleted. This was done in two ways: according to a *fixed* deletion scheme, or according to a random choice with above deletion *prob*ability for each taxon; these deletion strategies are henceforth referred to as *fixed* and *prob*, respectively. From the resulting multiple sequence alignments, Maximum Parsimony was used to construct phylogenetic trees, whose topology defines the input trees of the supertree method.

There are 100 instances for each of the 120 parameter combinations, resulting in 12,000 instances. We randomly selected 10 out of each set of 100, and therefore considered a set of 1,200 supertree problem instances uniformly covering all parameter settings. We use the best heuristic solutions from [3] as initial upper bounds for our ILP approach.

7.2 Can we solve instances to provable optimality despite NP-hardness?

Table 2(a) summarizes our results with respect to the constructed instance properties. The number of taxa is the most influential parameter: While we solve nearly 72% of the smaller instances, we can only prove optimal solutions for roughly 25% of the larger instances, within 30 minutes. The number s of subtrees plays a crucial role; this is not surprising as the number of characters is $n \approx ms$. In contrast, the difference between *prob* and *fixed* is of virtually no influence. Many deletions also compromise solvability. This seems to be due to the column generation scheme, as in this case we need to generate many more variables, effectively running out of time.

We also investigate the solvability in terms of the final lower bound (LB; either the optimum or best lower bound obtained by the ILP) for the number of necessary flips, which is of interest for the comparison with fixed parameter approaches. Table 2(b) shows the percentage of instances solved to provable optimality for different LB ranges. The number of flips is in fact the most influential parameter when considering the ILP's success ratio within a given time bound.

7.3 How good are existing heuristics?

Since we are now able to compute exact solutions for medium-sized MFST problems, we can investigate the quality of existing heuristics. Interestingly, the improved heuristic by Chen *et al.* [3] found the optimal solution in 96% of all solved instances. When our ILP could not find (or prove) an optimal solution, we obtained a better upper bound only in 5 of 619 instances. We conclude that the heuristic [3] is very strong (on the considered types of input).

8. DISCUSSION

We developed three ILP formulations to solve minimum flip problems to provable optimality in practice despite their NP-hardness. The MFST and MFCT problems allow identical ILP formulations (except that for MFCT, the auxiliary constants w_{tc} from (1) never take the value zero); this is particularly noteworthy as there exist engineered fixed-parameter algorithms [1] for MFCT but not for MFST. Furthermore, we have shown the three formulation to be polytope-wise equivalent, but offer a trade-off in the number of constraints and variables.

Among the three ILP formulations, we chose to further engineer MFG, which has only natural variables, for the following reasons. For problems of interesting size (m in the order of thousands, n in the order of several thousands), already the number mn of natural variables will become a bottleneck. While we could use a standard branch-and-cut strategy, as the separation problem is efficiently solvable, an efficient column generation scheme is critical for the practical applicability of the approach. Hence, we developed a combinatorial column generation scheme that does not require extra computational effort beyond separation: Variables are only introduced if required to describe a violated M-constraint. When trying to apply a similar (though much more involved) column generation scheme to the alternative formulations, say to SID, it turns out that the identification of additionally required variables would require time comparable to our MFG separation algorithm, diminishing its the potential benefit of fewer constraints.

Interestingly, our column generation scheme loosely connects the size of the active ILP model to the difficulty of the problem (optimal number of flips), and hence to parameterized algorithms: If there is a solution with few flips, we expect that it can be found quickly with few variables and M-constraints, hence with a small ILP. It may be interesting to explore such connections between parameterized algorithms and combinatorial column generation for ILPs further in more depth and in more generality.

We are now able to solve MFST instances with up to almost 100 taxa and 1000 characters in some cases within 30 minutes; beyond that, our current approach clearly takes too much time to be practical. A very insightful finding is that the heuristic from [3] found the optimal solution in most cases. Our study is the first evaluation formally allowing to conclude this property. Further research towards strong facet-defining inequalities, on a more sophisticated second pass in the separation routine (i.e., trying to find a feasible solution for inactive ?-entries), and on better heuristics for initially predicting ?-entries all seem to be worthwhile undertakings in order to optimally solve larger instances.

Table 2: Percentage of instances solved to provable optimality within 30 minutes. A "*" denotes that the respective row/column is averaged over all possible settings for the respective parameter (darker rows aggregate over the lighter shaded rows below).

(a) Relative to instance properties.

			number of subtrees (s)										
m	del		2	4	6	8	10	12	14	16	18	20	*
48	*	*	100	98.3	95	88.3	71.7	68.3	58.3	51.7	46.7	38.3	**71.7**
	25%	*	100	100	100	95	100	80	85	85	55	55	85.5
		fixed	100	100	100	90	100	80	100	80	50	60	86
		prob	100	100	100	100	100	80	70	90	60	50	85
	50%	*	100	95	100	90	95	95	90	65	75	55	86
		fixed	100	90	100	80	90	100	90	60	80	60	85
		prob	100	100	100	100	100	90	90	70	70	50	87
	75%	*	100	100	85	80	20	30	0	5	10	5	43.5
		fixed	100	100	80	80	20	20	0	10	10	10	43
		prob	100	100	90	80	20	40	0	0	10	0	44
96	*	*	100	70	45	18.3	11.7	3.3	3.3	0	0	0	**25.2**
	25%	*	100	85	90	40	30	10	10	0	0	0	36.5
		fixed	100	70	90	30	40	10	0	0	0	0	34
		prob	100	100	90	50	20	10	20	0	0	0	39
	50%	*	100	60	40	15	5	0	0	0	0	0	22
		fixed	100	60	20	10	0	0	0	0	0	0	19
		prob	100	60	60	20	10	0	0	0	0	0	25
	75%	*	100	65	5	0	0	0	0	0	0	0	17
		fixed	100	30	0	0	0	0	0	0	0	0	13
		prob	100	100	10	0	0	0	0	0	0	0	21
*	*	*	100	84.2	70	53.3	41.7	35.8	30.8	25.8	23.3	19.2	**48.4**

(b) Rel. to LB

LB	solved
0	100%
1	100%
2	100%
3–4	100%
5–8	97%
9–16	87%
17–32	64%
33–64	55%
65–128	56%
129–256	19%
257–512	1%
513–1024	0%

9. REFERENCES

[1] S. Böcker, Q. B. A. Bui, and A. Truss. An improved fixed-parameter algorithm for minimum-flip consensus trees. In *Proc. of International Workshop on Parameterized and Exact Computation (IWPEC 2008)*, volume 5018 of *Lect. Notes Comput. Sc.*, pages 43–54. Springer, 2008.

[2] D. Chen, L. Diao, O. Eulenstein, D. Fernández-Baca, and M. J. Sanderson. Flipping: A supertree construction method. In *Bioconsensus*, volume 61 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 135–160. American Mathematical Society, 2003.

[3] D. Chen, O. Eulenstein, D. Fernández-Baca, and J. G. Burleigh. Improved heuristics for minimum-flip supertree construction. *Evol. Bioinform. Online*, 2:391–400, 2006.

[4] D. Chen, O. Eulenstein, D. Fernández-Baca, and M. Sanderson. Supertrees by flipping. In *Proc. of Conference on Computing and Combinatorics (COCOON 2002)*, volume 2387 of *Lect. Notes Comput. Sc.*, pages 391–400. Springer, 2002.

[5] D. Chen, O. Eulenstein, D. Fernández-Baca, and M. Sanderson. Minimum-flip supertrees: complexity and algorithms. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 3(2):165–173, 2006.

[6] W. Day, D. Johnson, and D. Sankoff. The computational complexity of inferring rooted phylogenies by parsimony. *Math. Biosci.*, 81:33–42, 1986.

[7] G. Estabrook, C. Johnson, and F. McMorris. An idealized concept of the true cladistic character. *Math. Bioscience*, 23:263–272, 1975.

[8] O. Eulenstein, D. Chen, J. G. Burleigh, D. Fernández-Baca, and M. J. Sanderson. Performance of flip supertree construction with a heuristic algorithm. *Syst. Biol.*, 53(2):299–308, Apr 2004.

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

[10] S. Kannan, T. Warnow, and S. Yooseph. Computing the local consensus of trees. *SIAM J. Comput.*, 27(6):1695–1724, 1998.

[11] C. Komusiewicz and J. Uhlmann. A cubic-vertex kernel for flip consensus tree. In *Proc. of Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2008)*, Leibniz International Proceedings in Informatics, pages 280–291, 2008.

[12] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Discrete Mathematics and Optimization. Wiley-Interscience, 1999.

[13] M. P. Ng and N. C. Wormald. Reconstruction of rooted trees from subtrees. *Discrete Appl. Math.*, 69(1-2):19–31, 1996.

[14] M. A. Ragan. Phylogenetic inference based on matrix representation of trees. *Mol. Phylogenet. Evol.*, 1(1):53–58, Mar 1992.

[15] C. Semple and M. Steel. A supertree method for rooted trees. *Discrete Appl. Math.*, 105(1-3):147–158, 2000.