Better Methods for Solving Parsimony and Compatibility

Maria Bonet*, Mike Steel†, Tandy Warnow‡, and Shibu Yooseph§

Abstract

Evolutionary tree reconstruction is a challenging problem with important applications in Biology and Linguistics. In Biology, one of the most promising approaches to tree reconstruction is to find the "maximum parsimony" tree, while in Linguistics, the use of the "maximum compatibility" method has been very useful. However, these problems are NP-hard, and current approaches to solving these problems amount to heuristic searches through the space of possible tree topologies (a search which can, on large trees, take months to complete). In this paper, we present a new technique, *Optimal Tree Refinement*, for reconstructing very large trees. Our technique is motivated by recent experimental studies which have shown that certain polynomial time methods reliably return *contractions* of the true tree. We study the use of this technique in solving maximum parsimony and maximum compatibility, and present both hardness results and polynomial time algorithms.

1 Introduction

The task of the systematic biologist is to recover the order of speciation events that gave rise to the observed species under investigation. This order of speciation events is represented in the *topology* of the evolutionary tree for the species. For scientific as well as information-theoretic reasons, locating the root of the evolutionary tree is quite difficult (and even impossible under certain models of evolution), so that the primary objective of a phylogenetic method is the recovery of the *unrooted* version of the tree that gave rise to the taxa.

Many of the popular methods in phylogenetic tree reconstruction attempt to solve NP-hard optimization problems, and are usually implemented as heuristic searches through the space of different trees, *Maximum parsimony*, *maximum compatibility*, and *maximum likelihood* are three such optimization problems which have been used in biology, but polynomial time methods, such as *neighbor-joining* [31], also exist. By assuming a model of how biomolecular sequences evolve, it is possible to explore the performance of these methods. A recent result by Steel and Tuffley [**?**] showed that under a rather general model of biomolecular sequence evolution (which is closer to the biological assumptions than other proposed models), maximum likelihood and maximum parsimony are equivalent. On the other hand, there are conditions under this general model in which these two methods (and indeed, any method at all) are *inconsistent*, which means that even given infinite length sequences it can be impossible to determine the correct tree topology [**?**]. There are conditions under which it is possible to guarantee that the topology of the tree can be recovered (with arbitrarily high probability, given long enough sequences), but these conditions are unreasonable with respect to biomolecular sequences, since they assume that the sites within the sequences evolve identically and independently (the *iid* assumption). Under the *iid* assumption, however, it is known that all "reasonable" distance-based methods are *consistent*, that is, each will recover the correct topology given long enough sequences. For this reason, some statisticians and biologists prefer distance-based reconstruction methods over parsimony-based methods. However, both experimental [21] and analytical studies [11, 5, 3, 12] suggest that the actual sequence length that suffices for the various promising distance based methods to obtain (with high probability) an accurate topology estimation may be exceedingly large if the true tree contains significant divergence, even under the *iid* assumption. Consequently, although distance-based methods tend to be fast and are provably consistent under some models of biomolecular sequence evolution, they may not obtain even *approximately* correct reconstructions of the topology of the evolutionary tree, if that evolutionary tree is very large and contains significant divergence. On

*Universidad Politecnica de Catalunya, Dept. Lenguages y Sist. Informaticos, Jordi Girona Salgado 1-3, 08034 Barcelona. **Email: bonet@lsi.upc.es**. This work was done with support from ESPRIT 20244 ALCOM-IT.

†Biomathematics Research Centre, University of Canterbury, Christchurch, New Zealand. **Email: mas@math.canterbury.ac.nz**. The author would like to thank the Marsden Fund for supporting his research.

‡SAME

§DIMACS, Rutgers University, 96 Frelinghuysen Road, Piscataway, NJ 08854. **Email: yooseph@dimacs.rutgers.edu**. This work was partly supported by a Graduate Fellowship from the Institute for Research in Cognitive Science at the University of Pennsylvania and also by a Graduate Fellowship from the Program in Mathematics and Molecular Biology at the University of California at Berkeley, which is supported by the NSF under grant no. DMS-9406348. Part of this work was also supported by a DIMACS postdoctoral fellowship.

the other hand, heuristic approaches for the maximum parsimony problem have performed very well in experimental studies, and often perform significantly better than the best polynomial time distance-based methods, such as neighbor-joining [31] in reconstructing very large trees [17, 21, 22]. For this reason, many important data sets are analyzed in biology using heuristic parsimony searches. Unfortunately these searches can last for months or longer (see [?]) without necessarily finding the most parsimonious tree(s). This is partly due to the fact that parsimony is NP-hard to solve exactly [14, 10], and also perhaps due to the techniques (primarily branch-swapping) that have been used to search the tree space. Obtaining faster and more accurate methods for solving parsimony is an important objective in phylogenetics.

Evolutionary studies in Historical Linguistics similarly have the primary objective of recovering the topology of the evolutionary tree, and here too the most effective optimization problem (the *Maximum Compatibility* problem) is NP-hard to solve [32, 9, 23]. However, computationally expensive but exact algorithms exist for the Maximum Compatibility problem [1, 2, 19, 18], and the application of these algorithms to data for the Indo-European family of languages has potentially resolved conflicts that have troubled the historical linguistics community for centuries. Further studies by the researchers in this project have indicated that these current algorithms are not fast enough to handle the more complex data sets they are encountering in analyzing other langauge families, such as *Dravidian*, and that faster algorithms for the Maximum Compatibility problem (and the related problems that apply to *polymorphic data* [7]) need to be developed (Tandy Warnow, personal communication).

In this paper, we describe new methods for solving parsimony and compatibility, and at the same time we present a new general technique for tree reconstruction. Our methods and general technique are motivated by a recent experimental study [22], which demonstrated that some fast tree construction methods return with high frequency a contraction of the true tree (that is, the reconstructed tree can be obtained from the true tree by contracting some set of edges). Consequently, the true tree is a *refinement* of the reconstructed tree. We pose and study the *Optimal Tree Refinement* Problem, and consider this problem specifically for the maximum parsimony and maximum compatibility optimization criteria. Although some of our results are negative, showing that generally the optimal tree refinement problem is NP-hard to solve, we also obtain algorithms which will be efficient for many of the cases that we may expect to find in practice. We conclude with open problems.

2 Preliminary material

2.1 Definitions

Let T be a fixed leaf-labelled tree, in which the leaves are labelled by a set $S \subseteq Z^k$. Let T be equipped with labels in Z^k for each of its internal nodes as well. Then the *cost* of this *internally* labelled version of T (denoted by (T, L)) is the total number of times each site changes state in the tree. This may also be calculated as

$$cost(T, L) = \sum_{e \in E(T)} H(e),$$

where $H(e)$ is the Hamming distance between the endpoints of e, i.e. $H(e) = |\{i : x_i \neq y_i\}|$, where $e = (x, y)$. The *parsimony score* of T is then the minimum of $cost(T, L)$, taken over all internal labellings L of the nodes of T. The problem of finding the tree with the lowest cost for a given set of sequences which will label the leaves of the tree is the *maximum parsimony problem.*

We now define the *maximum compatibility problem.* As before, let T be a tree leaf-labelled by S, and let L be an internal labelling of the nodes of T, where all labels and elements in S are drawn from Z^k. Now consider a particular site (index i), and let r_i be the number of distinct values at the leaves for this site. It should be clear that no matter how the internal nodes of the tree are labelled in L, that the tree T must contain at least $r_i - 1$ edges on which the i^{th} site changes state between the endpoints. Any site i which has the property that on (T, L) it changes exactly $r_i - 1$ times is said to be *compatible* or *convex* on (T, L), and the number of such sites that are compatible on (T, L) is denoted by $compat(T, L)$. The *compatibility score* of T is then the maximum of $compat(T, L)$, as L ranges over all possible ways of labelling the internal nodes of T.

If the tree T is known, it is straightforward to solve both of these optimization problems: that is, to find the best labelling of the internal nodes of T. However, if T is not known, then the problem is to find the best tree which is optimal with respect to parsimony or with respect to compatibility; unfortunately, each of these problems is NP-hard. Worse, each is NP-hard even for binary (two-state) characters [9, 10].

A tree T_1 is said to be a *refinement* of tree T_2 if T_2 can be obtained from T_1 by contracting some of the edges. As discussed earlier, experimental studies have suggested that some methods frequently return contractions of the true tree. This motivates the following problems.

Optimal Tree Refinement with respect to Parsimony: (OTR-parsimony)
Input: A tree T which is leaf labelled by S (described by a set C of characters).
Output: A refinement T' of T, such that the cost of T' is the least among all refinements of T.

Optimal Tree Refinement with respect to Compatibility: (OTR-compatibility)
Input : A tree T which is leaf labelled by species from S (described by a set C of characters).
Output : A refinement T' of T, such that the compatibility score of T' is the largest among all refinements of T.

Throughout this paper, we will use the following notation for the parameters to the problem. We will let S denote the set of taxa (whether languages, biological species, or genes), and we will let C denote the set of characters. Each character $c \in C$ is a function $c : S \to Z$, where Z denotes the set of integers; each character $c \in C$ defines a *character state* on each $s \in S$ and this value is given by $c(s)$. We will let $k = |C|$ and $n = |S|$, and we will use r to denote the maximum number of states per character. We will use d to denote the maximum degree of any node in T. Some of these parameters may in general be expected to be small on some of the data sets we examine. For example, in Biology, biomolecular data (such as DNA or RNA sequences) are typically used to reconstruct trees, and for such data r is typically 4. When the tree to be refined is significantly resolved, then d may be quite small, and in general we may expect d to be not particularly large. On the other hand, k is the sequence length, and when using biomolecular sequences k can be on the order of hundreds or thousands, and n may take any number in a range between, say, $10-1,000$.

A special case of the parsimony and compatibility optimization problems is the *perfect phylogeny* problem, which asks whether a tree exists which has parsimony score $\sum_i(r_i-1)$. Such a tree would have every character compatible, and hence have optimal parsimony and compatibility scores. Thus, parsimony and compatibility are related in their extreme case. A tree which satisfies this constraint is said to be a *perfect phylogeny*. The perfect phylogeny problem is NP-hard in general, but solvable in P for the various fixed-parameter versions of the problem (i.e. when r, k, or n is fixed). (See [18, 1, 2, 32, 23, 28].)

2.2 Preliminary results on OTR-compatibility and OTR-parsimony

Theorem 1
- *OTR-parsimony and OTR-compatibility are solvable in polynomial time when n is bounded, and both are NP-hard when $r=2$ if k, d, and n are unbounded.*
- *OTR-parsimony is also NP-hard for $k=1$ if r, n and d are unbounded.*
- *The OTR-compatibility problem is NP-hard when both $r \geq 4$ and $d \geq 5$ are fixed.*

Proof. We provide a brief sketch of the easy results in this theorem, but omit the others due to space constraints. The Maximum Compatibility and Maximum Parsimony problems are both NP-hard for binary characters (i.e. when $r=2$), so that OTR-parsimony and OTR-compatibility are both NP-hard for $r=2$ (we let T be the n-star). If n is fixed, then both OTR-parsimony and OTR-compatibility become tractable, since an exhaustive search through all the refinements of the tree T combined with the computation of the optimal labellings of the tree (using [16]) is a polynomial time algorithm. ∎

One of the results in Theorem 1 is the obvious simple exhaustive search algorithm through all topologies. Since we will use that algorithm in a later section, we give it here:

Algorithm 1: Given T, simply examine each refinement and compute its optimal labelling using Fitch-Hartigan [16]. This uses $O((2d-3)!!)^{d^h/(d-1)}nkr)$ time, for a tree of height h, degree d, and having $n \leq d^h$ leaves, where $(2d-3)!! = (2d-3)(2d-5)(2d-7)\ldots 3$.

3 OTR-parsimony

3.1 Basic terminology

In this section we present three other algorithms for OTR-parsimony. Algorithm 2 is an exact algorithm, Algorithm 3 is a 2-approximation, and Algorithm 4 is a Polynomial Time Approximation Scheme (PTAS). All these algorithms share a common structure and are based upon similar ideas, although the PTAS draws from ideas in these algorithms and uses Algorithm 1 to solve small problems exactly. We will describe in some detail the method for Algorithm 2, but due to space constraints, we will let the reader infer the implementation details for the related algorithms and the PTAS by analogy. Without loss of generality, we will assume that the input to the OTR-parsiony problem is a rooted tree. We begin with some terminology.

Definition 1 *We will let $[r]^k$ denote the set of k-tuples over $\{1,2,\ldots,r\}$; thus each element of $[r]^k$ denotes a possible label for an internal node in a refinement T', and the leaves of T are labelled by elements of $[r]^k$. For $x \in V(T)$, we denote by T_x the subtree of T rooted at x. Since T is rooted, for T' a refinement of T we can define a mapping from $V(T)$ to $V(T')$ as follows. For each node $v \in V(T)$, let S_v be the set of leaves in T_v; we then associate to v the node $v' \in V(T')$ defined by $v' = lca_{T'}(S_v)$. We will call this mapping from $V(T)$ to $V(T')$ the* **canonical mapping**, *and denote by v' the image of v in T' under the canonical mapping. Note that this natural mapping is injective. Given a node $v \in V(T)$ with children $v_1, v_2, \ldots, v_p$ ($p \leq d$), let $F_{T'}(v)$ denote the minimal subtree of T' containing the node set $\{v', (v_1)', (v_2)', \ldots, (v_p)'\}$.*

3.2 Algorithm 2: Exact algorithm for OTR-parsimony

Algorithm 2: This algorithm uses dynamic programming to compute an optimal refinement (with respect to parsimony) for each subtree rooted at each node in T for each possible label at the node. For each $x \in V(T)$, and for each vector $s \in [r]^k$, we will compute a most parsimonious tree $T(x,s)$ and its parsimony score $cost(x,s) := cost(T(x,s))$ subject to the constraints that x' (the image of x in $T(x,s)$ under the canonical mapping) is labelled s and $T(x,s)$ is a refinement of T_x. This computation is made from the leaves up.

Suppose we have computed $T(x,s)$ and $cost(x,s)$ for each $s \in [r]^k$ and each node x below a specific node y, and we now wish to compute $T(y,s)$ and $cost(y,s)$ for each $s \in [r]^k$. Let $y_1, y_2, \ldots, y_p$ (where $p \leq d$) be the children of y. We seek an unknown rooted tree $T(y,s)$ refining T_y, and a labelling L of $T(y,s)$ by elements in $[r]^k$ so as to minimize the parsimony score. Given such a tree $T(y,s)$ and its optimal internal labelling L, we can define $F_{T(y,s)}(y)$ (see Definition 1) and the set $A(y,s)$ of labels assigned to the nodes in $F_{T(y,s)}(y)$ by L. It is clear that the parsimony score of $T(y,s)$ given L is the sum of the parsimony score of $F_{T(y,s)}(y)$ given the labels in $A(y,s)$, summed together with the parsimony costs of the p subtrees rooted at $(y_1)', (y_2)', \ldots, (y_p)'$. Two observations about this cost now follow. The parsimony score of $F_{T(y,s)}(y)$ given $A(y,s)$ is simply the cost of a minimum spanning tree (MST) on

$A(y,s)$, and the selection of labels for the $(y_i)'$ can be made arbitrarily within the set $A(y,s)$. Hence, the parsimony score of $T(y,s)$ given the label set $A(y,s)$ for $F_{T(y,s)}(y)$ equals $cost(MST(A(y,s)) + \sum_{i=1\ldots p} min\{cost(y_i,a) : a \in A(y,s)\}$.

Since the label set $A(y,s)$ can be any subset of size at most $2d-2$ within $[r]^k$, it follows that

$$cost(y,s) = min_A\{cost(MST(A)) + \sum_{i=1\ldots p} min\{cost(y_i,a) : a \in A\},$$

where A ranges over the set of subsets of $[r]^k$ of size at most $2d-2$. Call these sets A *local label sets*. Note that the values $cost(y_i,a)$ have already been computed, since we proceed from the bottom up. This suggests the following algorithm:

Stage 1: Initialize $cost(y,a) = \infty$ (or some large enough value) for each $a \in [r]^k$ and $y \in V(T)$. For all $\{x,y\} \subset [r]^k$, compute $H(x,y)$. For all local label sets A compute a minimum spanning tree $MST(A)$ and its cost.

Stage 2: Proceeding from the leaves upward towards the root, process each node v as follows:

If v is a leaf THEN

> $cost(v,s) = 0$ if v is labelled by s, else $cost(v,s) = \infty$ (or some large enough value)
>
> $T(v,s) = v$

ELSE

> For each local label set A and each $y \in V(T)$, define $Cost(y,A)$ to be the cost of the most parsimonious refinement of T_y such that $A(y,s) = A$. We use the discussion above to compute $Cost(y,A)$. The MST of A and the selection of labels so as to minimize the cost defines a refinement of (the unrooted form of) T_y, which we denote by $T(y,A)$. Note that the choice of label for the root of $T(y,A)$ can be any element in A, without increasing the parsimony score. For each $a \in A$, if $Cost(y,A) < cost(y,a)$ then $cost(y,a) \leftarrow Cost(y,A)$.

Stage 3: To compute the cost of the optimal refinement, find the minimum of $cost(root,x)$ where $x \in [r]^k$. At the end, given the optimal cost achievable for the entire tree T, we backtrack to reconstruct one of the most parsimonious trees.

Theorem 2 *The optimal refinement of a tree with n leaves labelled by elements of $[r]^k$ and degree bound d can be computed in $O(nr^{k(2d-2)} + kr^{2k})$ time.*

3.3 Algorithm 3: 2-Approximating OTR-Parsimony

We present a 2-approximation algorithm for OTR with running time $O(n^{d+1}+n^2k)$. The theorem which shows that this approximation algorithm achieves a 2-approximation ratio is based upon results in [24], which we now summarize.

Assume T is rooted and semi-labelled by elements of $S \subseteq Z^k$. We now use some techniques and ideas, with appropriate extensions to this problem, from [24]. We define a *lifted labelling* of a tree T to be an assignment of labels to the internal nodes of T such that for each node v in T, the label for v is drawn (i.e. *lifted*) from the set of labels of its children. (Our terminology differs from that in [24], in which a lifted labelling is called a *lifted tree*.) Since in the OTR problem, we simultaneously seek a refinement as well as a labelling, we now extend the definition of a lifted labelling as follows. We say that T' is a **lifted labelled-refinement** of T if T' refines T, and for each internal node v in T', the label for v is drawn from the labels of its children. We now return to the results in [24]:

Theorem 3 *[From [24]]: For any tree T and any scoring function on pairwise alignments which satisfies the triangle inequality, there is a lifted labelling of cost no more than twice the cost of an optimal tree alignment.*

Theorem 4 *Let T be an arbitrary tree leaf-labelled by elements of $[r]^k$. Then there exists a lifted labelled refinement of T of cost at most twice the cost of the optimal refinement.*

Proof. Note that the Hamming distance between two sequences is a scoring function on pairs of sequences, in which deletions or insertions have a very large cost, and any substitution has the same cost as any other. Consequently, we can apply Theorem 3. Let T' be the optimal refinement for T; hence, each node in T' is labelled by elements of $[r]^k$, and T' refines T. If we consider T' as a semi-labelled tree, then by Theorem 3, there exists a lifted labelling of T' yielding a 2-approximation for the optimal tree alignment problem on T'. T' with this lifted labelling of T' is a lifted labelled refinement of T with parsimony score at most twice that of the optimal. ∎

Thus, to 2-approximate the optimal refinement of T, we find an optimal lifted labelled refinement of T. A straightforward modification to Algorithm 2 gives an algorithm for finding an optimal refinement of T which will consider only labels from the original set S, and hence obtain a refinement which is at least as good as the optimal lifted labelling refinement of T. The only changes that need be considered are the possible *local label sets*, since in this case $A \subset S$. A straightforward analysis of the running time of this approach shows the following:

Theorem 5 *We can obtain a 2-approximation to the OTR-parsimony problem in $O(n^{d+1} + n^2k)$ time.*

3.4 A PTAS:

It is also possible to modify the approximation scheme given in [24] (and reanalyzed in [25]) to obtain an approximation scheme for the OTR-parsimony. Due to space constraints, we provide the following brief description.

Definition 2 *Let T be a rooted tree with leaves labelled by finite sequences, and let X be a subset of $V(T)$ containing the leaves and the root of T. For $v \in X$, a **X-child of** v is a node $w \in X$ in the subtree of T rooted at v such that the internal nodes on the path from v to w in T are not also in X. A **X-lifted labelling of** T is a labelling of the* *internal nodes of T such that the label for each node in X is drawn from its X-children. A* **b-family for a tree** *T is a set $\mathcal{V} = \{V_0, V_2, \ldots, V_{b-1}\}$ of subsets of $V(T)$ such that $V_i \cap V_j = \{root(T)\} \cup \mathcal{L}(T)\ \forall i, j,\ i \neq j$, where $\mathcal{L}(T)$ indicates the leaves of T.*

Theorem 6 *Let $\mathcal{V} = \{V_0, V_1, \ldots, V_{b-1}\}$ be a b-family for a tree T, and let T_i be an optimal V_i-lifted labelling of T. Then $\sum_i cost(T_i) \leq (b + 2 - \frac{2}{2^b})\ cost(T^*)$ where T^* is T equipped with an optimal internal labelling.*

In [25], this theorem is proved for a particular b-family, but the proof goes through for arbitrary b-familes as well. We continue.

Definition 3 *T' is a X-***lifted labelled refinement** *of T if T' refines T and if the labelling of the internal nodes of T' is a X'-lifted labelling, where X' is the image of X under the canonical mapping.*

Moreover,

Theorem 7 *Let $\mathcal{V} = \{V_0, V_2, \ldots, V_{b-1}\}$ be a b-family for T, and let T_i be the optimal V_i-lifted refinement of T. Then*

$$\frac{\sum_i cost(T_i) \leq (b + 2 - 2}{2^b) cost(T^*)}$$

where T^* is an optimal labelled refinement of T. Hence for some i,

$$\frac{cost(T_i) \leq (1 + 2}{b - \frac{2}{b 2^b}) cost(T^*).}$$

The proof is similar to that of Theorem 4 and uses Theorem 6.

This suggests a natural approach for achieving a $(1+ \frac{2}{b} - \frac{2}{b2^b})$-approximation ratio, in which for fixed b, the optimal V_i-lifted labelled refinements are computed for each $i = 0, 1, 2, \ldots, b-1$, and the best one selected. As in the 2-approximation algorithm, instead of computing an optimal V_i-lifted labelled refinement of T, we will compute a labelled refinement of T which is guaranteed to be at least as good as a V_i-lifted labelled refinement. We omit the details of how the algorithm is performed but note that it follows along the same lines as our 2-approximation algorithm in modifying the approximation scheme given in [24]. We obtain then:

Theorem 8 *We can find a labelled refinement T' of T such that $cost(T') \leq (1+ \frac{2}{b} - \frac{2}{b2^b})\ cost(T^*)$ in*

$$O((2d-3)!!)^{d^b/(d-1)} n^{d^b+2} kr)$$

time, where T^* is an optimal refinement of T.

4 OTR-Compatibility

The maximum compatibility method, like parsimony, is not a consistent estimator [13] of trees when the the *iid* assumption is made. However, when the data are such that most of the characters should be compatible on the true tree, then the maximum compatibility method may be a consistent estimator. While such data sets may be rare, they do exist in biology (see, for example, Hillis' 228 taxon data set in [17], as discussed in [20, 27]), and approaches that are based upon solving compatibility may in those (unusual) cases be appropriate. However, in Historical Linguistics, it may be more typical to have such data, as has been indicated in [26]. In fact, the maximum compatibility method and OTR-compatibility have been used to resolve the evolutionary history of the Indo-European family of languages [7]. Thus, in some types of biological data, and in linguistic data, we may expect that most of the characters will be compatible on the true tree, so that the number t of characters which must be deleted in order for all the remaining characters to be compatible on some tree will in general be small. We will call t the *imperfection* of the data (since what remains fits a "perfect phylogeny"; see Section 2), and we will study the maximum compatibility problem and the OTR-Compatibility problem in the context where t is small as well as where d (the maximum degree in the tree) is small.

We have already shown that OTR-compatibility is NP-hard for unbounded degree trees with binary characters and also for bounded degree trees with $r \geq 4$. In the following sections we will show:

1. OTR-compatibility is hard to approximate, even when d is bounded.
2. OTR-compatibility on binary characters can be solved in polynomial time for every fixed degree bound d. We can also 2-approximate the minimum number of characters which are not compatible on the optimal refinement of the tree T in $O(k^2 n)$ time, if all the characters are binary.
3. The imperfection of a set of binary characters can be 2-approximated in $O(k^2 n)$ time, and can be solved exactly in $O(k^2 n + k^2 3^t)$ time.
4. Maximum compatibility of r-state characters can be solved in $O(2^{2r} k^{t+2} n)$

4.1 Preliminary material about binary character compatibility

In the earlier discussion of compatibility, we stated that a character $c : S \to Z$ is compatible on a tree T leaf labelled by S if the internal nodes of the tree can each be labelled by states for c so that c changes state exactly $r_c - 1$ times (where r_c is the number of states c attains on S). There is a closely related concept, which we now precisely define.

Definition 4 *A set C of characters is said to be* compatible *if there is a tree T on which every character in C is compatible.*

There are particular statements we can make when the characters are binary, because then the space of trees on which the characters are compatible has a unique minimal element (with respect to refinement).

First, we define the *character encoding* of the tree T as follows. Each edge e in the tree T defines a bipartition on the leaves of T and hence can be described by a binary character

c_e. The collection $\{c_e : e \in E(T)\}$ is the character encoding (or set of splits) of T, and is denoted by $C(T)$. The following lemma is part of the standard literature (see [15]):

Lemma 1 *Let C be a collection of binary characters defined on a set S. Then there is a tree T on which every character in C is compatible if and only if every pair of characters in C is compatible. Furthermore, given a set C of compatible binary characters, there is a unique tree T satisfying $C(T) = C$.*

Now, consider the OTR-compatibility problem, in which we are given a tree T and a set C of binary characters defined on the leaves of T, and we wish to find a refinement of T on which the maximum number of characters in C are compatible. We will show we can solve OTR-compatibility on a tree T by solving Max Compatibility on a derived set of characters.

Note the following:

Observation 1 *Suppose that T' is some refinement of T, and that $c \in C$ is compatible on T' (for some labelling L). Then c is compatible with every character in $C(T)$.*

Consequently, we can *preprocess* the set C by deleting from C any character which is incompatible with some character in $C(T)$. (Note that this is not the same as deleting characters which are incompatible *on* T.) This preprocessing can be done in $O(nk)$ time using an algorithm in [15] for "tree-compatibility". We will call the reduced set of characters C^*.

We will now prove the following:

Theorem 9 *Let T be a tree with a set C of binary characters defined on the leaves of T. Let T' be the optimal refinement of T with respect to compatibility, and let C' be the characters in C which are compatible on T'. Then C' is a maximum compatible subset of C^*, where $C^* = \{c \in C : c$ is compatible with every character in $C(T)\}$. Conversely, if C' is a maximum compatible subset of C^*, then the tree T' defined by $C(T') = C(T) \cup C'$ is an optimal refinement of T with respect to compatibility.*

Proof. We first prove that the size of a maximum compatible subset of C^* is a lower bound for the number of characters in C compatible on an optimal refinement of T. If $C' \subseteq C^*$ is a maximum cardinality subset of compatible characters, then the set $C' \cup C(T)$ is a set of pairwise compatible characters, and hence it is *setwise* compatible by Lemma 1. Furthermore, there is a tree T' defined by $C(T') = C(T) \cup C'$. T' is a refinement of T and every character in $C(T) \cup C'$ is compatible on T'. Consequently, the size of a maximum compatible set of characters in C^* is a lower bound on the number of characters compatible on an optimal refinement of T.

Now, suppose T' is an optimal refinement of T, and that $C' \subset C$ is the set of characters compatible on T'. By Observation 1, it follows that $C' \subseteq C^*$, and that C' is a set of compatible characters. Hence, the number of characters from C compatible on an optimal refinement of T is a lower bound on the size of a maximum compatible subset of C^*. Since both quantities are lower bounds of each other, they are identical. ∎

To summarize, when the set C only consists of binary characters, then in $O(kn)$ preprocessing, we can reduce the OTR-compatibility problem to the Max Compatibility problem on a derived set of binary characters.

4.2 Maximum compatibility and OTR-compatibility for binary characters

The result in Theorem 14 indicates that the OTR-compatibility problem is hard, even to approximate, for bounded degree trees when the set of characters is allowed to have an unbounded number of character states. The exact algorithm for OTR-compatibility for $r \geq 3$ state characters was not potentially fast enough to be of general use except under unusual circumstances. However, when $r = 2$, then many problems related to maximum compatibility become tractable. In this section we describe several of these results.

We now describe how to approximate the *imperfection* of an optimal solution to the Max Compatibility problem for binary characters.

4.2.1 Approximating the imperfection of binary characters

Let C be a set of characters defined on a set S of taxa. We begin with some definitions.

Given C and S, we will define the **incompatibility graph**, G_C, as follows. The nodes of G_C are the elements of C, and the edge (i, j) is in $E(G_C)$ if and only if the characters i and j are incompatible. An *independent set* in a graph $G = (V, E)$ is a subset $V_0 \subset V$ such that for all nodes v, w in V, $(v, w) \notin E$. A *vertex cover* in G is a subset $V_1 \subset V$ such that for all edges $(a, b) \subset E$, at least one of a or b is in V_1. Consequently, the complement of an independent set is a vertex cover, and vice-versa. Since pairwise compatibility of binary characters suffices for setwise compatibility by Lemma 1, the maximum independent set of G_C corresponds to the maximum compatible subset of C. Consequently, if we can find a maximum independent set in G_C we will have solved the Max Compatibility problem.

The maximum independent set problem is however very hard to solve [6], but in the cases that we are interested in, the imperfection is generally quite low (see the discussion in the introduction). Consequently, the size of the maximum independent set will in general be quite large, and consequently the size of the minimum vertex cover (the complement to the maximum independent set) will be quite small. There is also a very simple and fast 2-approximation algorithm for finding the minimum vertex cover (see [8]) which we will use to obtain a 2-approximation algorithm for the imperfection t. We now briefly describe the 2-approximation algorithm.

A greedy 2-approximation algorithm for the minimum vertex cover A *matching* in a graph $G = (V, E)$ is a subset $E_0 \subseteq E$ such that no two edges in E_0 share an endpoint. It follows that if V_1 is a vertex cover E_0 is a matching, then V_1

contains at least one endpoint from each of the edges in E_0. Consequently, the size of a minimum vertex cover is bounded from below by the size of *any* matching. On the other hand, if E_0 is a *maximal matching* (meaning there does not exist any matching E_1 which properly contains E_0), then the set V^* which contains *both* endpoints of every edge is a vertex cover (since its complement is an independent set), and its cardinality is bounded from above by twice the size of a minimum vertex cover. Obtaining a maximal matching in a graph is a trivially easy problem – simply remove an edge and both its endpoints from the graph, until there are no remaining edges. The edges that have been removed constitute a maximal matching. This is an $O(|V|+|E|)$ algorithm, and it produces a 2-approximate solution to the minimum vertex cover.

Consequently, the following holds:

Theorem 10 *The imperfection of a set of binary characters can be 2-approximated in $O(k^2n)$ time.*

Proof. We begin by constructing the incompatibility graph G_C; this takes $O(k^2n)$ time since pairwise compatibility of two characters takes $O(n)$ time. Then we use the 2-approximation algorithm for the minimum vertex cover [8], which takes $O(k^2)$ time. The complement of the vertex cover is then an independent set in G_C and hence denotes a set C' of pairwise (and hence setwise) compatible characters, with $|C'| \geq k - 2t$. ∎

4.2.2 Solving Max Compatibility of binary characters

We now show how to solve the maximum compatibility problem for binary characters.

We begin by constructing G_C, the incompatibility graph. We then compute a 2-approximation to the imperfection using the above algorithm. This explicitly gives us a vertex cover V_0 of size at most $2t$, and furthermore V_0 consists of both endpoints of $p \leq t$ edges (this follows from the greedy algorithm used to approximate the vertex cover). The complement of V_0 is an independent set, V_1. Now, consider a maximum independent set I in G_C. Let $A_I = I \cap V_0$. Note that although many different I define the same A_I, for each set $A \subseteq V_0$ there is a unique maximum such independent set $I(A)$ such that $I(A) \cap V_0 = A$. This $I(A)$ is defined by $I(A) = A \cup (V_1 - \Gamma(A))$, where $\Gamma(A)$ denotes the neighbors of A. Consequently, to find a maximum independent set, we examine all possible sets $A \subseteq V_0$ and then in $O(k^2)$ time per set A, we compute $I(A)$. At the end, we return a largest such set $I(A)$. All we need to do is to compute how many different sets A there are, and show we can list them efficiently. There are only 3^p sets A, since each A contains at most one of the endpoints of each of the p edges used to construct V_0. We do not have to keep track of more than one set A at a time, so that this does not incur a cost in terms of space.

Consequently, we have proven the following:

Theorem 11 *The maximum compatibility problem can be solved in $O(k^2n+3^tk^2)$ time, where there are k binary characters defined on n taxa, and having imperfection t.*

We now address the final result regarding binary characters, which is useful when t (the imperfection) is not small enough to make the exact algorithm feasible, but d (the maximum degree) is small enough to use another technique. Once again, we note that no character which is incompatible with T will be compatible on any refinement of T, and hence we preprocess the data by deleting all such characters.

Let V be the set of internal nodes in T. Consider the set E' of edges of T where each $e_i = (u, v) \in E'$ is such that neither u nor v is a leaf. Subdivide each edge $e_i = (u, v) \in E'$, by adding a new node w_i on e_i (i.e. add edges $(u, w_i), (w_i, v)$ and remove (u, v)). Let T' be the resulting labelled tree. Recall that each leaf in T (and hence in T') is a species from S and hence each leaf is assigned a state for each character. We now show how to *extend* the characters to take values on the added nodes, w_i. The deletion of the node w_i from T' creates two subtrees. For each character c, if there are leaves in the two subtrees having the same state for c, then we assign $c(w_i)$ to that shared state. If there are two different states which are shared between the two subtrees, then c is not compatible on any refinement of T, and we would have deleted c from the set. The final possibility is that no state is shared between the two subtrees, in which case we assign a new state to that node. In this way, we have set states for each of the characters on each of the newly introduced nodes, w_i. We now show how this permits us to solve $O(n)$ independent Max Compatibility instances.

Let v be one of the internal nodes in T and let S_v be $\{v\} \cup \Gamma(v)$ where $\Gamma(v)$ is the neighbors of v in the new tree, T'. It is then easy to see that for each character $c \in C$, there is at most one set S_v on which c is *not constant* (this is due to the fact that the characters are all binary, and all characters that are not compatible on any refinement of T have already been eliminated). For this reason, the imperfection of the set C (i.e. the number of characters which are incompatible on some fixed optimal refinement of T) is precisely the sum, over all internal nodes $v \in V(T)$, of the *imperfection* of C on S_v. This quantity can be estimated (either exactly or approximately) by using the appropriate Max Compatibility algorithm for binary characters.

We summarize our results on binary characters as follows:

Theorem 12 *The imperfection of the optimal refinement can be 2-approximated in $O(nk^2)$ time. The OTR-compatibility problem can be solved exactly in $O(3^qnk^2)$ time, where q is the maximum imperfection on any subproblem S_v, and in $O(g(d)nk)$ time, where $g(d)$ is the number of rooted binary trees on d nodes.*

4.3 Max Compatibility and OTR-compatibility for $r \geq 3$

We can show that approximating OTR-compatibility for a *bounded* degree tree is as hard as approximating MAX Independent set on graphs [4]. We begin with a result from [4] on maximum independent set.

Theorem 13 *[4] There is an $\epsilon > 0$ such that approximating MAX Independent set within a factor n^ϵ is NP-hard, where n is the number of vertices in the input graph.*

Theorem 14 *Let T be a bounded degree tree with leaf set S (described by a set C of k characters). Then there is an $\epsilon > 0$ such that approximating OTR-compatibility within a factor k^ϵ is NP-hard.*

Proof. We will give a simple reduction from the Independent Set problem.

Let $G = (V, E)$ be the graph, with n nodes and m edges, in the instance I of the Independent Set problem. We will construct the instance I' to the OTR-compatibility problem as follows. The tree T is given in Figure 1. Note that it has degree bounded by 5.

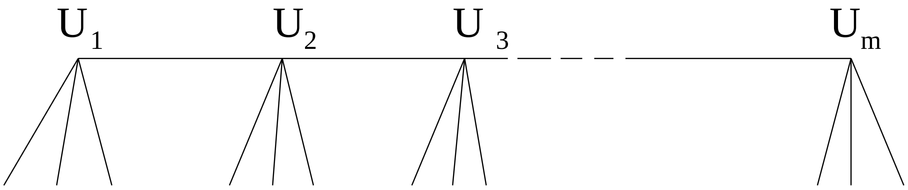

Figure 1: Topology of tree T

T has m internal nodes, $u_1, u_2, ..., u_m$, where node u_i corresponds to edge $e_i \in E$. We will use s_i^1, s_i^2, s_i^3 to denote the three species adjacent to the internal node u_i in T. Let A denote the set of species. We will define n characters, $c_1, c_2, ..., c_n$, where, the idea is that, character c_j corresponds to node $v_j \in V$. We now describe how the character states are set for the species in A.

Let $e_i = (v_j, v_k)$ be an edge in E. Then, for all $l \notin \{j, k\}$, set $c_l(s_i^1) = c_l(s_i^2) = c_l(s_i^3) = l$. Also, set
$c_j(s_i^1) = c_j(s_i^2) = k, c_j(s_i^3) = j$,
$c_k(s_i^1) = c_k(s_i^3) = j, c_k(s_i^2) = k$.

We make the following easily proven claim.

Claim: The graph G has an independent set $\{v_{i_1}, v_{i_2}, ..., v_{i_p}\}$ if and only if the character set $\{c_{i_1}, c_{i_2}, ..., c_{i_p}\}$ can be made compatible on some refinement of T.

Because of the nature of the reduction (i.e. an independent set of size p corresponds to a refinement of T with p compatible characters and vice-versa), it can be seen that approximating MAX OTR-compatibility for bounded degree trees is at least as hard as approximating MAX Independent set. ∎

We describe an algorithm to solve the OTR-compatibility problem that runs in time exponential in the incompatibility score t and r. We make use of the $O(2^{2r}nk^2)$ time algorithm of [18] for finding perfect phylogenies.

Examine all subsets of characters, from largest to smallest, and find the first such subset C_0 such that C_0 is compatible with T (that is, every character in C_0 is compatible on some refinement of T). This is equivalent to testing whether $C(T) \cup C_0$ has a perfect phylogeny, where $C(T)$ is the character encoding of T. By the above, this can be tested in $O(2^{2r}k^2n)$ time for each set C_0, for a total of $O(2^{2r}k^{t+2}n)$ time. Thus, we have:

Theorem 15 *OTR-compatibility can be solved in $O(2^{2r}k^{t+2}n)$ time.*

Algorithm :
Begin
For each $C_0 \subseteq C$, where $|C_0| = k - t$, do

- For each internal $v \in T$ which has degree greater than 3, do:
  1. Let $\Gamma(v) = \Gamma_1(v) \cup \Gamma_2(v)$ where $\Gamma_1(v)$ consists of all the neighbours of v which are leaves and $\Gamma_2(v)$ consists of all the non-leaf neighbours of v. For each $u_j \in \Gamma_2(v)$ add a new node w_j on the edge (v, u_j). Compute the labelling of w_j so as to make every character in C_0 convex (each character must contain every state that appears on both sides of w_j).
  2. If some new node has a character that requires more than one character state for that character to be convex, then RETURN(No). Let $S_v = \Gamma_1(v) \cup \{w_j | w_j$ is a new node and w_j is a neighbor of $v\}$. Use the algorithm of [18] to determine if a perfect phylogeny exists for (S_v, C_0). If any (S_v, C_0) fails to have a perfect phylogeny then RETURN(No), else RETURN(yes).

Return (Yes) if and only if some C_0 returns (Yes).
End

The run time of the above algorithm is $O(2^{2r}k^{t+2}n)$ since there are $O(k^t)$ subsets of size $k - t$, and for each subset C_0 of this size, we apply the perfect phylogeny algorithm to each (S_v, C_0).

5 Discussion and conclusion

In this paper we have discussed the computational complexity of solving the Optimal Tree Refinement problem with respect to two optimization criteria, *parsimony* and *compatibility*. This approach to tree reconstruction is motivated by the empirical observation (see [22]) that certain methods are likely to produce contractions of the true tree. This suggests that that a two step process of evolutionary tree reconstruction (first obtain a contraction of the tree, and then refine the tree optimally) may produce more accurate topologies than existing methods. This approach is especially appropriate when the optimization problem is hard to solve. Although the resultant problems are likely to also be hard to solve, narrowing the search space may in some cases lead to easier subproblems, and hence better estimates of the topology.

We have not addressed the problem of optimally refining a tree with respect to maximum likelihood estimation, which is potentially a very powerful tool, and we have also not addressed the use of heuristics for solving OTR problems in general. We leave these approaches for tree reconstruction to later work.

References

[1] R. Agarwala and D. Fernández-Baca, *A Polynomial-time Algorithm for the Perfect Phylogeny Problem when the*

Number of Character States is Fixed, SIAM J. on Computing, Vol. 23, No. 6, pp. 1216-1224, 1996.

[2] R. Agarwala and D. Fernández-Baca, *Fast and Simple Algorithms for Perfect Phylogeny and Triangulating Colored Graphs*, DIMACS Tech Report 94-51, 1994.

[3] Ambainis, A., Desper, R., Farach, M., and S. Kannan, *Nearly tight bounds on the learnability of evolution,* to appear, FOCS 1997.

[4] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, *Proof verification and intractability of approximation problems*, Proc. 33rd IEEE Symp. on Foundations of Computer Science, pp. 13-22, 1992.

[5] Atteson, K. *On the performance of neighbor-joining tree reconstruction methods,* Proceedings, COCOON 1997.

[6] Bellare, M. and M. Sudan, *Improved non-approximability results*, Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, (Montreal), ACM, pp. 184-193.

[7] Bonet, M., Phillips, C.A., Warnow, T., and S. Yooseph. *Constructing evolutionary trees in the presence of polymorphic characters,* ACM Symposium on the Theory of Computing, 1996. Accepted for SIAM J. Computing.

[8] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, MIT Press, 1990.

[9] W. H. E. Day and D. Sankoff, *Computational complexity of inferring phylogenies by compatibility*, Syst. Zool., Vol. 35, No. 2 (1986), pp. 224–229.

[10] Day, W.H.E. 1983: *Computationally difficult parsimony problems in phylogenetic systematics,* Journal of Theoretical Biology, 103: 429-438.

[11] Erdös, P.L., Steel, M., Székeley, L. and T. Warnow. *Inferring big trees from short sequences.* Proceedings of International Congress on Automata, Languages, and Programming 1997.

[12] Farach, M. and S. Kannan (1996). *Efficient algorithms for inverting evolution, Proceedings of the ACM Symposium on the Foundations of Computer Science*, 230–236.

[13] Felsenstein, J. *Cases in which parsimony or compatibility methods will be positively misleading*, Syst. Zoology, 27:401-410, 1978.

[14] Foulds, L.R., and R.L. Graham. 1982. *The Steiner Problem in Phylogeny is NP-Complete*, Adv. Appl. Math. 3, 43-49.

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

[16] J. A. Hartigan, *Minimum mutation fits to a given tree,* Biometrics 29, 1973, pp. 53-65.

[17] Hillis, D. (1997). *Inferring complex phylogenies*, Nature, Vol. 383, pp. 130-131.

[18] S. Kannan and T. Warnow, *Finding and enumerating all perfect phylogenies when the number of states is fixed,* SIAM J. on Computing (to appear). A preliminary version appeared in SODA '95.

[19] Kannan, S. and T. Warnow. 1994. *Inferring Evolutionary History from DNA Sequences,* SIAM J. on Computing, Vol. 23, No. 4, pp. 713-737. (A preliminary version of this paper appeared at FOCS 1990.)

[20] *Reply from A. Purvis and D.L.J. Quicke*, TREE 12(9):357-358, 1997.

[21] Rice, K.A. and T. Warnow 1997. *Parsimony is hard to beat!* Proceedings COCOON 1997.

[22] Rice, K.A. M. Steel, T. Warnow, and S. Yooseph (1997) *Hybrid Tree Construction Methods,* manuscript.

[23] M. A. Steel, *The complexity of reconstructing trees from qualitative characters and subtrees*, Journal of Classification, 9 (1992), pp. 91–116.

[24] Wang, L., Jiang, T., and E. Lawler, *Approximation algorithms for tree alignment with a given phylogeny,* Algorithmica 16, 1996, pp. 302-315.

[25] Wang, L. and D. Gusfield, *Improved approximation algorithms for tree alignment,* Journal of Algorithms, to appear.

[26] Warnow, T. *Mathematical approaches to comparative linguistics.* Proceedings of the National Academy of Sciences, 1997.

[27] Z. Yang and N. Goldman. *Are big trees indeed easy?* TREE 12(9):357, 1997.

[28] McMorris, F.R., Warnow, T. and Wimer, T. 1993: *Triangulating vertex colored graphs, SIAM journal of discrete mathematics*, Vol. 7, No. 2, pp. 296-306.

[29] Fitch, Wm. 1971: Toward defining the course of evolution: minimum change for a specified tree topology, *Syst. Zool.*, 20:406-416.

[30] Warnow, T., Ringe, D. and A. Taylor. 1996. Reconstructing the evolutionary history of natural languages, Association for Computing Machinery and the Society of Industrial and Applied Mathematics, Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA), 1996, pp. 314-322.

[31] Saitou, N., and M. Nei. 1987. *The neighbor-joining method: A new method for reconstructing phylogenetic trees.* Mol. Biol. Evol. 4:406-425.

[32] H. Bodlaender, M. Fellows, and T. Warnow, *Two strikes against perfect phylogeny*, In Proceedings of the 19th International Colloquium on Automata, Languages, and Programming, Springer Verlag, Lecture Notes in Computer Science (1992), pp. 273–283.