Simple Reconstruction of Binary Near-Perfect Phylogenetic Trees*

Srinath Sridhar[1], Kedar Dhamdhere[2], Guy E. Blelloch[1], Eran Halperin[3], R. Ravi[4], and Russell Schwartz[5]

[1] Computer Science Dept, CMU
srinath@cs.cmu.edu
[2] Google Inc, Mountain View, CA
kedar.dhamdhere@gmail.com
[3] ICSI, University of California, Berkeley
heran@icsi.berkeley.edu
[4] Tepper School of Business, CMU
ravi@cmu.edu
[5] Department of Biological Sciences, CMU
russells@andrew.cmu.edu

Abstract. We consider the problem of reconstructing near-perfect phylogenetic trees using binary character states (referred to as BNPP). A perfect phylogeny assumes that every character mutates at most once in the evolutionary tree, yielding an algorithm for binary character states that is computationally efficient but not robust to imperfections in real data. A near-perfect phylogeny relaxes the perfect phylogeny assumption by allowing at most a constant number q of additional mutations. In this paper, we develop an algorithm for constructing optimal phylogenies and provide empirical evidence of its performance. The algorithm runs in time $O((72\kappa)^q nm + nm^2)$ where n is the number of taxa, m is the number of characters and κ is the number of characters that share four gametes with some other character. This is fixed parameter tractable when q and κ are constants and significantly improves on the previous asymptotic bounds by reducing the exponent to q. Furthermore, the complexity of the previous work makes it impractical and in fact no known implementation of it exists. We implement our algorithm and demonstrate it on a selection of real data sets, showing that it substantially outperforms its worst-case bounds and yields far superior results to a commonly used heuristic method in at least one case. Our results therefore describe the first practical phylogenetic tree reconstruction algorithm that finds guaranteed optimal solutions while being easily implemented and computationally feasible for data sets of biologically meaningful size and complexity.

1 Introduction

Reconstruction of evolutionary trees is a classical computational biology problem [9]. In the maximum parsimony (MP) model of this problem one seeks the

* Supported in part by NSF grant CCR-0105548 and ITR grant CCR-0122581(The ALADDIN project)

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part II, LNCS 3992, pp. 799–806, 2006.

smallest tree to explain a set of observed organisms. Parsimony is a particularly appropriate metric for trees representing short time scales, which makes it a good choice for inferring evolutionary relationships among individuals within a single species or a few closely related species. The intraspecific phylogeny problem has become especially important in studies of human genetics now that large-scale genotyping and the availability of complete human genome sequences have made it possible to identify millions of single nucleotide polymorphisms (SNPs) [18], sites at which a single DNA base takes on two common variants.

Minimizing the length of a phylogeny is the problem of finding the most parsimonious tree, a well known NP-complete problem [7]. Researchers have thus focused on either sophisticated heuristics or solving optimally for special cases (e.g. fixed parameter variants [1, 3, 13]). Previous attempts at such solutions for the general parsimony problem have only produced theoretical results, yielding algorithms too complicated for practical implementation. A large number of related work has been published but it is impossible to mention all of them here.

Fernandez-Baca and Lagergren recently considered the problem of reconstructing optimal near-perfect phylogenies [6], which assume that the size of the optimal phylogeny is at most q larger than that of a perfect phylogeny for the same input size. They developed an algorithm to find the most parsimonious tree in time $nm^{O(q)}2^{O(q^2s^2)}$, where s is the number of states per character, n is the number of taxa and m is the number of characters. This bound may be impractical for sizes of m to be expected from SNP data, even for moderate q. Given the importance of SNP data, it would therefore be valuable to develop methods able to handle large m for the special case of $s = 2$, a problem we call Binary Near Perfect Phylogenetic tree reconstruction (BNPP).

Our Work: Here we present theoretical and practical results on the optimal solution of the BNPP problem. We completely describe and analyze an intuitive algorithm for the BNPP problem that has running time $O((72\kappa)^q nm + nm^2)$, where κ is the number of characters that violate the *four gamete* condition, a test of perfectness of a data set explained below. Since $\kappa \leq m$ this result significantly improves the prior running time by removing the big-oh from the exponent. Furthermore, the complexity of the previous work would make practical implementation daunting; to our knowledge no implementation of it has ever been attempted. Our results thus describe the first practical phylogenetic tree reconstruction algorithm that finds guaranteed optimal solutions while being easily implemented and computationally feasible for data sets of biologically meaningful size and complexity. We implement our algorithm and demonstrate it on a selection of real mitochondrial, Y-chromosome and bacterial data sets, showing that it substantially outperforms its worst-case bounds and yields far superior results to a commonly used heuristic method in at least one case.

2 Preliminaries

A phylogenetic tree T is called *perfect* if for all states s and characters c, all taxa having state s at character c lie in a connected component of the phylogeny.

Since the problem of reconstructing a perfect phylogeny is NP-complete [2, 17], Gusfield considered an important special case when the number of states is bounded by 2, called the binary perfect phylogeny problem (BPP). He showed that the BPP problem can be solved in linear time [8]. The problem we consider is an extension called the binary *near* perfect phylogeny reconstruction (BNPP).

In defining formal models for parsimony-based phylogeny construction, we borrow definitions and notations from Fernandez-Baca and Lagergren [6]. The input to the BNPP problem is an $n \times m$ matrix I where rows R represent *taxa* and are strings over states. The columns C are referred to as *characters*. Thus, every taxon $r \in \{0,1\}^m$. In a *phylogenetic tree*, or *phylogeny*, each vertex v corresponds to a taxon and has an associated label $l(v) \in \{0,1\}^m$.

Definition 1. *A* phylogeny *for a set of n taxa R is a tree $T(V, E)$ with the following properties:*

1. *if a taxon $r \in R$ then $r \in l(V(T))$*
2. *for all $(u, v) \in E(T)$, $H(l(u), l(v)) = 1$ where H is the Hamming distance*

Definition 2. *For a phylogeny T:*

- *length(T)* $= |E(T)|$
- *penalty*$(T) =$ *length*$(T) - m$
- *vertex v of T is* terminal *if $l(v) \in R$ and* Steiner *otherwise.*

The BNPP problem: Given an integer q and an $n \times m$ input matrix I, where each row(taxon) $r \in \{0,1\}^m$, find a phylogeny T such that *length(T)* is minimized or declare `NIL` if all phylogenies have penalty larger than q. The problem is equivalent to finding the minimum Steiner tree on a hyper-cube if the optimal tree is at most q larger than the number of dimensions or declaring `NIL` otherwise. The problem is fundamental and therefore expected to have diverse applications besides phylogenies.

Definition 3. *We define the following additional* notations*:*

- *$r[i] \in \{0,1\}$: the state in character i of taxa r*
- *$\mu(e) : E(T) \to C$: the character corresponding to edge $e = (u, v)$ with the property $l(u)[\mu(e)] \neq l(v)[\mu(e)]$*

We say that an edge e *mutates* character c' if $\mu(e) = c'$. We will use the following well known definition and lemma on phylogenies:

Definition 4. *The* set of gametes $G_{i,j}$ *for characters i, j is defined as: $G_{i,j} = \{(k,l) | \exists r \in R, r[i] = k, r[j] = l\}$. Two characters $i, j \in C$ contain (all) four gametes when $|G_{i,j}| = 4$.*

Lemma 1. *[8] The most parsimonious phylogeny for input I is not perfect if and only if I contains the four-gamete property.*

Input Assumptions: If no pair of characters in input I contains the four-gamete property, we can use Gusfield's elegant algorithm [8] to reconstruct a

perfect phylogeny. We assume that the all zeros taxa is present in the input. If not, using our freedom of labeling, we convert the data so that it contains the same information with the all zeros taxa (see section 2.2 of Eskin et al [4] for details). We now remove any character that contains only one state. Such characters do not mutate in the whole phylogeny and are therefore useless in any phylogeny reconstruction. We now repeat the following preprocessing step. For every pair of characters c', c'' if $|G_{c',c''}| = 2$, we (arbitrarily) remove character c''. After preprocessing, we have the following lemma:

Lemma 2. *For every pair of characters* c', c'', $|G_{c',c''}| \geq 3$.

We will assume that the above lemma holds on the input matrix for the rest of the paper. Note that such characters c', c'' are identical (after possibly relabeling one character) and are usually referred to as non-informative. It is not hard to show that this preprocessing step does not change the correctness or running time of our algorithm.

Conflict Graph G**:** The *conflict graph* G, introduced by Gusfield et al. [11], is used to represent the imperfectness of the input in a graph. Each vertex $v \in V(G)$ of the graph represents a character $c(v) \in C$. An edge (u, v) is added if and only if all the four gametes are present in $c(u)$ and $c(v)$. Let VC be any minimum vertex cover of G. Damaschke [3] showed that the minimum number of characters that needs to be removed to support a perfect phylogeny is the minimum vertex cover of the conflict graph. Therefore $|VC|$ is a lower bound on $\texttt{penalty}(T_{opt})$ and this is often useful in practice. We now introduce new definitions that will be used to decompose a phylogeny:

Definition 5. *For any phylogeny* T *and set of characters* $C' \subseteq C$*:*

- *a* super node *is a maximal connected subtree* T' *of* T *s.t. for all edges* $e \in T', \mu(e) \notin C'$
- *the* skeleton *of* T, $s(T, C')$*, is the tree that results when all super nodes are contracted to a vertex. The vertex set of* $s(T, C')$ *is the set of super nodes. For all edges* $e \in s(T, C')$, $\mu(e) \in C'$.

Definition 6. *A* tag $t(u) \in \{0, 1\}^m$ *of super node* u *in* $s(T, C')$ *has the property that* $t(u)[c'] = l(v)[c']$ *for all* $c' \in C'$*, vertices* $v \in u$*;* $t[u][i] = 0$ *for all* $i \notin C'$.

Throughout this paper, w.l.o.g. we will deal with phylogenies and skeletons that are rooted at the all zeros taxa and tag respectively. Furthermore, the skeletons used in this work themselves form a perfect phylogeny in the sense that no character mutates more than once in the skeleton. Note that in such skeletons, tag $t(u)[i] = 1$ i.f.f. character i mutates exactly once in the path from the root to u. Figure 3(a) shows an example of a skeleton of a phylogeny. We will use the term *sub-phylogeny* to refer to a subtree of a phylogeny.

3 Algorithm Description

Throughout the analysis, we fix an optimal phylogeny T_{opt} and show that our algorithm finds it. We assume that both T_{opt} and its skeleton is rooted at the

function buildNPP (binary matrix I, integer q)

1. let $G(V, E)$ be the conflict graph of I
2. let $V_{nis} \subseteq V$ be the set of non-isolated vertices
3. for all $M \in 2^{c(V_{nis})}$, $|M| \leq q$
   (a) construct rooted perfect phylogeny $\texttt{PP}(V_{PP}, E_{PP})$ on characters $C \setminus M$
   (b) define $\lambda : R \mapsto V_{PP}$ s.t. $\lambda(r) = u$ i.f.f. for all $i \in C \setminus M$, $r[i] = t(u)[i]$
   (c) $T_f :=$ **linkTrees** (PP)
   (d) if $\texttt{penalty}(T_f) \leq q$ then return T_f
4. return NIL

Fig. 1. Pseudo-code to find the skeleton

function linkTrees (skeleton $Sk(V_s, E_s)$)

1. let $S := \texttt{root}(Sk)$
2. let $R_S := \{s \in R | \lambda(s) = S\}$
3. for all children S_i of S
   (a) let Sk_i be subtree of Sk rooted at S_i
   (b) $(r_i, c_i) :=$ **linkTrees**(Sk_i)
4. let $\texttt{cost} := \sum_i c_i$
5. for all i, let $l_i := \mu(S, c_i)$
6. for all i, define $p_i \in \{0,1\}^m$ s.t. $p_i[l_i] \neq r_i[l_i]$ and for all $j \neq l_i$, $p_i[j] = r_i[j]$
7. let $\tau := R_S \cup (\cup_i \{p_i\})$
8. let $D \subseteq C$ be the set of characters where taxa in τ differ
9. guess root taxa of S, $r_S \in \{0,1\}^m$ s.t. $\forall i \in C \setminus D, \forall u \in \tau,\ r_S[i] = u[i]$
10. let c_S be the size of the optimal Steiner tree of $\tau \cup \{r_S\}$
11. return $(r_S, \texttt{cost} + c_S)$

Fig. 2. Pseudo-code to construct and link imperfect phylogenies

all zeros label and tag respectively. The high level idea of our algorithm is to first guess the characters that mutate more than once in T_{opt}. The algorithm then finds a perfect phylogeny on the remaining characters. Finally, it adds back the imperfect components by solving a Steiner tree problem. The algorithm is divided into two functions: `buildNPP` and `linkTrees` and the pseudo-code is provided in Figures 1 and 2.

Function `buildNPP` starts by determining the set of characters $c(V_{nis})$ that corresponds to the non-isolated vertices of the conflict graph in Step 2. From set $c(V_{nis})$, the algorithm then selects by brute-force the set of characters M that mutate more than once in T_{opt}. Only characters corresponding to non-isolated vertices can mutate more than once in any optimal phylogeny (a simple proof follows from Buneman graphs [16]). Since all characters of $C \setminus M$ mutate exactly once, the algorithm constructs a perfect phylogeny on this character set using Gusfield's linear time algorithm [8]. The perfect phylogeny is unique because of Lemma 2. Note that PP is the skeleton $s(T_{opt}, C \setminus M)$. Since the tags of the skeleton are unique, the algorithm can now determine the super node where

every taxon resides as defined by function λ in Step 3b. This rooted skeleton PP is then passed into function `linkTrees` to complete the phylogeny.

Function `linkTrees` takes a rooted skeleton Sk (sub-skeleton of PP) as argument and returns a tuple (r, c). The goal of function `linkTrees` is to convert skeleton Sk into a phylogeny for the taxa that reside in Sk by adding edges that mutate M. Notice that using function λ, we know the set of taxa that reside in skeleton Sk. The phylogeny for Sk is built bottom-up by first solving the phylogenies on the sub-skeleton rooted at children super nodes of Sk. Tuple (r, c) returned by function call to `linkTrees`(Sk) represents the cost c of the optimal phylogeny when the label of the root vertex in the root super node of Sk is r. Let $S =$ `root`(Sk) represent the root super node of skeleton Sk. R_S is the set of input taxa that map to super node S under function λ. Let its children super nodes be $S_1, S_2, \ldots$. Assume that recursive calls to `linkTrees`(S_i) return (r_i, c_i). Notice that the parents of the set of roots r_i all reside in super node S. The parents of r_i are denoted by p_i and are identical to r_i except in the character that mutates in the edge connecting S_i to S. Set τ is the union of p_i and R_S, and forms the set of vertices inferred to be in S. Set D is the set of characters on which the labels of τ differ i.e. for all $i \in D, \exists r_1, r_2 \in \tau, r_1[i] \neq r_2[i]$. In Step 9, we guess the root r_S of super node S. This guess is 'correct' if it is identical to the label of the root vertex of S in T_{opt}. Notice that we are only guessing $|D|$ bits of r_S. Corollary 1 of Lemma 3 along with optimality requires that the label of the root vertex of T_{opt} is identical to τ in all the characters $C \setminus D$:

Lemma 3. *There exists an optimal phylogeny T_{opt} that does not contain any degree 2 Steiner roots in any super node.*

Proof. Figure 3(b) shows how to transform a phylogeny that violates the property into one that doesn't. Root `10` is degree 2 Steiner and is moved into parent supernode as `01`. Since `10` was Steiner, the transformed tree contains all input.

Corollary 1. *In T_{opt}, the LCA of the set τ is the root of super node S.*

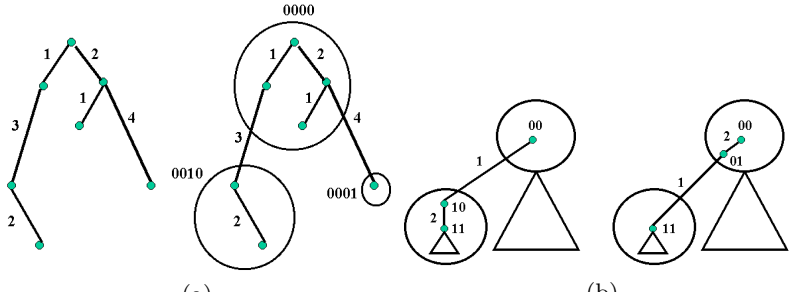

Fig. 3. (a) Phylogeny T and skeleton $s(T, C')$, $C' = \{3, 4\}$. Edges are labeled with characters that mutate μ and super nodes with tags t. (b) Transform to remove a degree 2 Steiner root from a super node. Note: the size of the phylogeny is unchanged.

Input size (Rows × Cols) and Desc	Penalty of opt(q)	Parsimony Score	Remarks and *total* run-time of our algorithm Intel P4 2.4Ghz, 1G RAM
24 × 1041 mtDNA genus Pan [19]	2	63	`pars` program of `phylip` [5](default parameters) gives parsimony score 252; 0.59 secs
15 × 98 chr Y, genus Pan [19]	1	99	identical to original paper which uses branch-and-bound; 0.33 secs
17 × 1510 Bacterial DNA sequence [14]	7	96	0.47 secs
150 × 49, HapMap chr Y, 4 ethnic groups [12]	1	16	0.3 secs

Fig. 4.

In step 10, the algorithm finds the cost of the optimum Steiner tree for the terminal set of taxa $\tau \cup \{r_S\}$. We use Dreyfus-Wagner recursion [15] to compute this minimum Steiner tree. The function now returns r_S along with the cost of the phylogeny rooted in S which is obtained by adding the cost of the optimum Steiner tree in S to the cost of the phylogenies rooted at c_i. The following Lemma bounds the running time of our algorithm and completes the analysis:

Lemma 4. *The algorithm described above runs in time* $O((18\kappa)^q nm + nm^2)$ *and solves the BNPP problem with probability at least* 2^{-2q}*. The algorithm can be easily derandomized to run in time* $O((72\kappa)^q nm + nm^2)$*.*

Proof. The probability of a correct guess at Step 9 in function `linkTrees` is exactly $2^{-|D|}$. Notice that the Steiner tree in super node S has at least $|D|$ edges. Since $\texttt{penalty}(T_{opt}) \leq q$, we know that there are at most $2q$ edges that can be added in all of the recursive calls to `linkTrees`. Therefore, the probability that all guesses at Step 9 are correct is at least 2^{-2q}. The time to construct the optimum Steiner tree in step 10 is $O(3^{|\tau|}2^{|D|})$. Assuming that all guesses are correct, the total time spent in Step 10 over all recursive calls is $O(3^{2q}2^q)$. Therefore, the overall running time of the randomized algorithm is $O((18\kappa)^q nm + nm^2)$. To implement the randomized algorithm, since we do not know if the guesses are correct, we can simply run the algorithm for the above time, and if we do not have a solution, then we restart. *Although presented as a randomized algorithm for ease of exposition, it is not hard to see that the algorithm can be derandomized by exploring all possible roots at Step 9.* The derandomized algorithm has total running time $O((72\kappa)^q nm + nm^2)$.

4 Experiments and Conclusion

We tested the derandomized algorithm using non-recombining DNA sequences. In such sequences, the most likely explanation for a pair of characters exhibiting all four gametes is recurrent mutations. The results are summarized in Figure 4.

Conclusion: We have presented an algorithm for inferring optimal near-perfect binary phylogenies that improves the running time of the previous method. This

problem is of considerable practical interest for phylogeny reconstruction from SNP data. In practice, we find that the algorithm significantly outperforms its worst case running time. Our algorithm is easily implemented unlike previous theoretical algorithms. At the same time, the algorithm returns guaranteed optimal solution unlike popular fast heuristics such as `pars`.

References

1. R. Agarwala and D. Fernandez-Baca. A Polynomial-Time Algorithm for the Perfect Phylogeny Problem when the Number of Character States is Fixed. In: *SIAM Journal on Computing*, 23 (1994).
2. H. Bodlaender, M. Fellows and T. Warnow. Two Strikes Against Perfect Phylogeny. In proc *ICALP*, (1992).
3. P. Damaschke. Parameterized Enumeration, Transversals, and Imperfect Phylogeny Reconstruction. In proc *IWPEC*, (2004).
4. E. Eskin, E. Halperin and R. M. Karp. Efficient Reconstruction of Haplotype Structure via Perfect Phylogeny. In *JBCB* 2003.
5. J. Felsenstein. PHYLIP version 3.6. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle. (2005).
6. D. Fernandez-Baca and J. Lagergren. A Polynomial-Time Algorithm for Near-Perfect Phylogeny. In: *SIAM Journal on Computing*, 32 (2003).
7. L. R. Foulds and R. L. Graham. The Steiner problem in Phylogeny is NP-complete. In: *Advances in Applied Mathematics* (3), (1982).
8. D. Gusfield. Efficient Algorithms for Inferring Evolutionary Trees. In: *Networks*, 21 (1991).
9. D. Gusfield. Algorithms on Strings, Trees and Sequences. *Cambridge University Press*, (1999).
10. D. Gusfield and V. Bansal. A Fundamental Decomposition Theory for Phylogenetic Networks and Incompatible Characters. In proc: *RECOMB* (2005).
11. D. Gusfield, S. Eddhu and C. Langley. Efficient Reconstruction of Phylogenetic Networks with Constrained Recombination. In Proc *IEEE CSB* (2003).
12. The International HapMap Consortium. The International HapMap Project. *Nature* 426 (2003).
13. S. Kannan and T. Warnow. A Fast Algorithm for the Computation and Enumeration of Perfect Phylogenies. In *SIAM Journal on Computing*, 26 (1997).
14. M. Merimaa, M. Liivak, E. Heinaru, J. Truu and A. Heinaru. Functional coadaption of phenol hydroxylase and catechol 2,3-dioxygenase genes in bacteria possessing different phenol and p-cresol degradation pathways. *unpublished*
15. H. J. Promel and A. Steger. The Steiner Tree Problem: A Tour Through Graphs Algorithms and Complexity. *Vieweg Verlag* (2002).
16. C. Semple and M. Steel. Phylogenetics. *Oxford University Press* (2003).
17. M. A. Steel. The Complexity of Reconstructing Trees from Qualitative Characters and Subtrees. In *J. Classification*, 9 (1992).
18. S. T. Sherry, M. H. Ward, M. Kholodov, J. Baker, L. Pham, E. Smigielski, and K. Sirotkin. dbSNP: The NCBI Database of Genetic Variation. In *Nucleic Acids Research*, 29 (2001).
19. A. C. Stone, R. C. Griffiths, S. L. Zegura, and M. F. Hammer. High levels of Y-chromosome nucleotide diversity in the genus Pan. In *Proceedings of the National Academy of Sciences* (2002).