Fixed Parameter Tractability of Binary Near-Perfect Phylogenetic Tree Reconstruction[⋆]

Guy E. Blelloch[⋆⋆], Kedar Dhamdhere[⋆⋆⋆], Eran Halperin[†], R. Ravi[‡], Russell Schwartz[§] and Srinath Sridhar[⋆⋆]

Abstract. We consider the problem of finding a Steiner minimum tree in a hypercube. Specifically, given n terminal vertices in an m dimensional cube and a parameter q, we compute the Steiner minimum tree in time $O(72^q + 8^q nm^2)$, under the assumption that the length of the minimum Steiner tree is at most $m + q$.

This problem has extensive applications in taxonomy and biology. The Steiner tree problem in hypercubes is equivalent to the phylogeny (evolutionary tree) reconstruction problem under the maximum parsimony criterion, when each taxon is defined over binary states. The taxa, character set and mutation of a phylogeny correspond to terminal vertices, dimensions and traversal of a dimension in a Steiner tree. Phylogenetic trees that mutate each character exactly once are called *perfect phylogenies* and their size is bounded by the number of characters. When a perfect phylogeny consistent with the data set exists it can be constructed in linear time. However, real data sets often do not admit perfect phylogenies. In this paper, we consider the problem of reconstructing near-perfect phylogenetic trees (referred to as BNPP). A near-perfect phylogeny relaxes the perfect phylogeny assumption by allowing at most q additional mutations. We show for the first time that the BNPP problem is fixed parameter tractable (FPT) and significantly improve the previous asymptotic bounds.

1 Introduction

One of the core areas of computational biology is phylogenetics, the reconstruction of evolutionary trees [13, 21]. This problem is often phrased in terms of a parsimony objective, in which one seeks the simplest possible tree to explain a set of observed taxa. Parsimony is a particularly appropriate objective for trees representing short time scales, such as those for inferring evolutionary relationships among individuals within a single species or a few closely related species. Such phylogeny problems have become especially important since we now have

[⋆] Supported in part by NSF grant CCF-043075 and ITR grant CCR-0122581(The ALADDIN project)

[⋆⋆] Computer Science Department, CMU. Email: {guyb, srinath}@cs.cmu.edu

[⋆⋆⋆] Google Inc, Mountain View, CA. Email: kedar.dhamdhere@gmail.com

[†] ICSI, 1947 Center St, Berkeley, CA. Email: heran@icsi.berkeley.edu

[‡] Tepper School of Business, CMU. Email: ravi@cmu.edu

[§] Dept Biological Sciences, CMU. Email: russells@andrew.cmu.edu

identified millions of single nucleotide polymorphisms (SNPs) [16, 17], sites at which a single DNA base takes on two common variants. Simply stated, if we examine any specific SNP site on the human genome, then all the individuals in the data sets can be classified into two classes. Therefore an individual's `A, C, G, T` string can be represented as a binary string with no loss of information.

Consider a $n \times m$ input matrix I, where each row represents an input taxon and is a string over states Σ. The columns of I are called *characters*. A phylogeny is a tree where vertices represent taxa and edges mutations. A phylogeny T for I is a tree that contains all the taxa in I and its length is the sum of the Hamming distances of adjacent vertices. Minimizing the length of a phylogeny is the problem of finding the most parsimonious tree, a well known NP-complete problem, even when $|\Sigma| = 2$ [10]. Researchers have thus focused on either sophisticated heuristics (e.g. [4, 11]) or solving optimally for special cases (e.g. [1, 18]).

In this work, we focus on the case when the set of states is binary, $|\Sigma| = 2$. The taxa can therefore be viewed as vertices of an m-cube, and the problem is equivalent to finding the Steiner minimum tree in an m-cube. In this setting, a phylogeny for I is called *perfect* if its length equals m. Gusfield showed that such phylogenies can be reconstructed in linear time [12]. If there exists no perfect phylogeny for input I, then one option is to slightly modify I so that a perfect phylogeny can be constructed for the resulting input. Upper bounds and negative results have been established for such problems. For instance, Day and Sankoff [6], showed that finding the maximum subset of characters containing a perfect phylogeny is NP-complete while Damaschke [7] showed fixed parameter tractability for the same problem. The problem of reconstructing the most parsimonious tree without modifying the input I seems significantly harder.

In the general case when $|\Sigma| = s$, a phylogeny for I is called perfect if the length is $m(s-1)$. In this setting, Bodlaender et al. [3] proved a number of crucial negative results, among them that finding the perfect phylogeny when the number of characters is a parameter is $W[t]$-hard for all t. A problem is fixed parameter tractable on parameter k if there exists an algorithm that runs in time $O(f(k)\text{poly}(|I|))$ where $|I|$ is the input size. Since $FPT \subseteq W[1]$, this shows in particular that the problem is not fixed parameter tractable (unless the complexity classes collapse).

Fernandez-Baca and Lagergren considered the problem of reconstructing optimum near-perfect phylogenies [9]. A phylogeny is q-near-perfect if its length is $m(s-1)+q$. They find the optimum phylogeny in time $nm^{O(q)}2^{O(q^2s^2)}$, assuming a q-near-perfect phylogeny exists. This bound may be impractical for sizes of m to be expected from SNP data (binary states), 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 $s = 2$, when all taxa are represented by binary strings. This problem is called Binary Near-Perfect Phylogenetic tree reconstruction (BNPP).

In a prior work, Sridhar et al. [20] solve the BNPP problem in time $O(\binom{m}{q}72^q nm + nm^2)$. The main contribution of the prior work is two-fold: they simplify the previous algorithm [9] which results in the reduction of the exponent in the run-time to q and demonstrate the first empirical results on near-perfect phylogenies. For real data sets that were solved, the range of values for n, m and q were: 15-150, 49-1510 and 1-7 respectively. However, many instances were unsolvable because of the high running time.

Our Work: Here, we present a new algorithm for the BNPP problem that runs in time $O(72^q + 8^q nm^2)$. This result significantly improves the prior running time. Fernandez-Baca and Lagergren [9] in concluding remarks state that the most important open problem in the area is to develop a parameterized algorithm or prove $W[t]$ hardness for the near-perfect phylogeny problem. We make progress on this open problem by showing for the first time that BNPP is fixed parameter tractable (FPT). To achieve this, we use a divide and conquer algorithm. Each divide step involves performing a 'guess' (or enumeration) with cost exponential in q. Finding the Steiner minimum tree on a q-cube dominates the run-time when the algorithm bottoms out.

2 Preliminaries

In defining formal models for parsimony-based phylogeny construction, we borrow definitions and notations from a couple of previous works [9, 21]. The input to a phylogeny problem is an $n \times m$ binary matrix I where rows $R(I)$ represent *input taxa* and are binary strings. The column numbers $C = \{1, \cdots, m\}$ are referred to as *characters*. In a *phylogenetic tree*, or *phylogeny*, each vertex v corresponds to a taxon (not necessarily in the input) and has an associated label $l(v) \in \{0, 1\}^m$.

Definition 1. *A* phylogeny *for matrix I is a tree $T(V, E)$ with the following properties: $R(I) \subseteq l(V(T))$ and $l(\{v \in V(T) | degree(v) \leq 2\}) \subseteq R(I)$. That is, every input taxon appears in T and every leaf or degree-2 vertex is an input taxon.*

Definition 2. *A vertex v of phylogeny T is* terminal *if $l(v) \in R(I)$ and* Steiner *otherwise.*

Definition 3. *For a phylogeny T,* $\texttt{length}(T) = \sum_{(u,v) \in E(T)} d(l(u), l(v))$*, where d is the Hamming distance.*

A phylogeny is called an optimum phylogeny if its length is minimized. We will assume that both states $0, 1$ are present in all characters. Therefore the length of an optimum phylogeny is at least m. This leads to the following definition.

Definition 4. *For a phylogeny T on input I,* $\texttt{penalty}(T) = \texttt{length}(T) - m$; $\texttt{penalty}(I) = \texttt{penalty}(T^{opt})$*, where T^{opt} is any optimum phylogeny on I.*

Definition 5. *A phylogeny T is called* q-near-perfect *if* $\texttt{penalty}(T) = q$ *and* perfect *if* $\texttt{penalty}(T) = 0$.

Note that in an optimum phylogeny, no two vertices share the same label. Therefore, we can equivalently define an edge of a phylogeny as (t_1, t_2) where $t_i \in \{0,1\}^m$. Since we will always be dealing with optimum phylogenies, we will drop the label function $l(v)$ and use v to refer to both a vertex and the taxon it represents in a phylogeny.

The BNPP problem: Given an integer q and an $n \times m$ binary input matrix I, if $\texttt{penalty}(I) \leq q$, then return an optimum phylogeny T, else declare `NIL`. The problem is equivalent to finding the minimum Steiner tree on an m-cube if the optimum tree is at most q larger than the number of dimensions m or declaring `NIL` otherwise. An optimum Steiner tree can easily be converted to an optimum phylogeny by removing degree-two Steiner vertices. The problem is fundamental and therefore expected to have diverse applications besides phylogenies.

Definition 6. *We define the following* notations*.*

- $r[i] \in \{0,1\}$*: the state in character i of taxon r*
- $\mu(e) : E(T) \rightarrow 2^C$*: the set of all characters corresponding to edge $e = (u,v)$ with the property for any $i \in \mu(e)$, $u[i] \neq v[i]$*
- *for a set of taxa M, we use T^*_M to denote an optimum phylogeny on M*

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

Definition 7. *Given matrix I, the* set of gametes $G_{i,j}$ *for characters i,j is defined as: $G_{i,j} = \{(r[i], r[j]) | r \in R(I)\}$. Two characters i,j* share t *gametes in I i.f.f. $|G_{i,j}| = t$.*

In other words, the set of gametes $G_{i,j}$ is a projection on the i,j dimensions.

Lemma 1. [12] *An optimum phylogeny for input I is not perfect i.f.f. there exists two characters i,j that share (all) four gametes in I.*

Definition 8. (Conflict Graph [15])*: A* conflict graph *G for matrix I with character set C is defined as follows. Every vertex v of G corresponds to unique character $c(v) \in C$. An edge (u,v) is added to G i.f.f. $c(u), c(v)$ share all four gametes in I. Such a pair of characters are defined to be in* conflict*. Notice that if G contains no edges, then a perfect phylogeny can be constructed for I.*

Simplifications: We assume that the all zeros taxon is present in the input. If not, using our freedom of labeling, we convert the data into an equivalent input containing the all zeros taxon (see section 2.2 of Eskin et al [8] 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. The BNPP problem asks for the reconstruction of an unrooted tree. For the sake of analysis, we will however assume that all the phylogenies are rooted at the all zeros taxon.

3 Algorithm

This section deals with the complete description and analysis of our algorithm for the BNPP problem. For ease of exposition, we first describe a randomized algorithm for the BNPP problem that runs in time $O(18^q + qnm^2)$ and returns an optimum phylogeny with probability at least 8^{-q}. We later show how to derandomize it. In sub-section 3.1, we first provide the complete pseudo-code and describe it. In sub-section 3.2 we prove the correctness of the algorithm. Finally, in sub-section 3.3 we upper bound the running time for the randomized and derandomized algorithms and the probability that the randomized algorithm returns an optimum phylogeny.

3.1 Description

We begin with a high level description of our randomized algorithm. The algorithm iteratively finds a set of edges E that decomposes an optimum phylogeny T^*_I into at most q components. An optimum phylogeny for each component is then constructed using a simple method and returned along with edges E as an optimum phylogeny for I.

We can alternatively think of the algorithm as a recursive, divide and conquer procedure. Each recursive call to the algorithm attempts to reconstruct an optimum phylogeny for an input matrix M. The algorithm identifies a character c s.t. there exists an optimum phylogeny T^*_M in which c mutates exactly once. Therefore, there is exactly one edge $e \in T^*_M$ for which $c \in \mu(e)$. The algorithm, then guesses the vertices that are adjacent to e as r, p. The matrix M can now be partitioned into matrices $M0$ and $M1$ based on the state at character c. Clearly all the taxa in $M1$ reside on one side of e and all the taxa in $M0$ reside on the other side. The algorithm adds r to $M1$, p to $M0$ and recursively computes the optimum phylogeny for $M0$ and $M1$. An optimum phylogeny for M can be reconstructed as the union of *any* optimum phylogeny for $M0$ and $M1$ along with the edge (r, p). We require at most q recursive calls. When the recursion bottoms out, we use a simple method to solve for the optimum phylogeny.

We describe and analyze the iterative method which flattens the above recursion. This makes the analysis easier. For the sake of simplicity we define the following notations.

– For the set of taxa M, $M(i, s)$ refers to the subset of taxa that contains state s at character i.
– For a phylogeny T and character i that mutates exactly once in T, $T(i, s)$ refers to the maximal subtree of T that contains state s on character i.

The pseudo-code for the above described algorithm is provided in Figure 1. The algorithm performs 'guesses' at Steps 2a and 2c. If all the guesses performed by the algorithm are 'correct' then it returns an optimum phylogeny. Guess at Step 2a is correct i.f.f. there exists $T^*_{M_j}$ where $c(v)$ mutates exactly once. Guess at Step 2c is correct i.f.f. there exists $T^*_{M_j}$ where $c(v)$ mutates exactly once and edge

```
buildNPP(input matrix $I$)

 1. let $L := \{I\}, E := \emptyset$
 2. while $|\cup_{M_i \in L} N(M_i)| > q$
    (a) guess vertex $v$ from $\cup_{M_i \in L} N(M_i)$, let $v \in N(M_j)$
    (b) let $M0 := M_j(c(v), 0)$ and $M1 := M_j(c(v), 1)$
    (c) guess taxa $r$ and $p$
    (d) add $r$ to $M1$, $p$ to $M0$ and $(r, p)$ to $E$
    (e) remove $M_j$ from $L$, add $M0$ and $M1$ to $L$
 3. for each $M_i \in L$ compute an optimum phylogeny $T_i$
 4. return $E \cup (\cup_i T_i)$
```

Fig. 1. Pseudo-code to solve the BNPP problem. For all $M_i \in L$, $N(M_i)$ is the set of non-isolated vertices in the conflict graph of M_i. Guess at Step 2a is correct i.f.f. there exists $T^*_{M_j}$ where $c(v)$ mutates exactly once. Guess at Step 2c is correct i.f.f. there exists $T^*_{M_j}$ where $c(v)$ mutates exactly once and edge $(r, p) \in T^*_{M_j}$ with $r[c(v)] = 1, p[c(v)] = 0$. Implementation details for Steps 2a, 2c and 3 are provided in Section 3.3.

$(r, p) \in T^*_{M_j}$ with $r[c(v)] = 1, p[c(v)] = 0$. Implementation details for Steps 2a, 2c and 3 are provided in Section 3.3. An example illustrating the reconstruction is provided in Figure 2.

3.2 Correctness

We will now prove the correctness of the pseudo-code under the assumption that all the guesses performed by our algorithm are correct. Specifically, we will show that if penalty(I) $\leq q$ then function buildNPP returns an optimum phylogeny. The following lemma proves the correctness of our algorithm.

Lemma 2. *At any point in execution of the algorithm, an optimum phylogeny for I can be constructed as $E \cup (\cup_i T_i)$, where T_i is* any *optimum phylogeny for $M_i \in L$.*

Proof. We prove the lemma using induction. The lemma is clearly true at the beginning of the routine when $L = \{I\}, E = \emptyset$. As inductive hypothesis, assume that the above property is true right before an execution of Step 2e. Consider any optimum phylogeny $T^*_{M_j}$ where $c(v)$ mutates exactly once and on the edge (r, p). Phylogeny $T^*_{M_j}$ can be decomposed into $T^*_{M_j}(c(v), 0) \cup T^*_{M_j}(c(v), 1) \cup (r, p)$ with length $l = \texttt{length}(T^*_{M_j}(c(v), 0)) + \texttt{length}(T^*_{M_j}(c(v), 1)) + d(r, p)$. Again, since $c(v)$ mutates exactly once in $T^*_{M_j}$, all the taxa in $M0$ and $M1$ are also in $T^*_{M_j}(c(v), 0)$ and $T^*_{M_j}(c(v), 1)$ respectively. Let T', T'' be *arbitrary* optimum phylogenies for $M0$ and $M1$ respectively. Since $p \in M0$ and $r \in M1$ we know that $T' \cup T'' \cup (r, p)$ is a phylogeny for M_j with cost $\texttt{length}(T') + \texttt{length}(T'') + d(r, p) \leq l$. By the inductive hypothesis we know that an optimum phylogeny for I can be constructed using any optimum phylogeny for M_j. We have now shown that using any optimum phylogeny for $M0$ and $M1$ and adding edge (r, p) we can construct an optimum phylogeny for M_j. Therefore the proof follows by induction. □

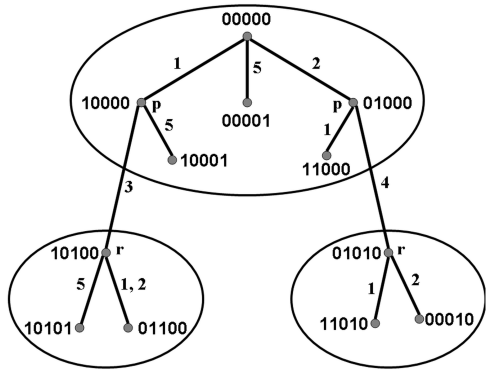

Fig. 2. Example illustrating the reconstruction. Underlying phylogeny is T_I^*; taxa r and p (both could be Steiner) are guessed to create $E = \{(10000, 10100), (01000, 01010)\}$; E induces three components in T_I^*. When all taxa in T_I^* are considered, character 3 conflicts with $1, 2$ and 5 and character 4 conflicts with 1 and 2; two components are perfect (penalty 0) and one has penalty 2; $\texttt{penalty}(I) =_{def} \texttt{penalty}(T_I^*) = 7$.

3.3 Bounds

In this sub-section we bound the probability of correct guesses, analyze the running time and show how to derandomize the algorithm. We perform two guesses at Steps 2a and 2c. Lemmas 3 and 5 bound the probability that all the guesses performed at these Steps are correct throughout the execution of the algorithm.

Lemma 3. *The probability that all guesses performed at Step 2a are correct is at least* 4^{-q}.

Proof. Implementation: The guess at Step 2a is implemented by selecting v uniformly at random from $\cup_i N(M_i)$.

To prove the lemma, we first show that the number of iterations of the `while` loop (step 2) is at most q. Consider any one iteration of the while loop. Since v is a non-isolated vertex of the conflict graph, $c(v)$ shares all four gametes with some other character c' in some M_j. Therefore, in every optimum phylogeny $T_{M_j}^*$ that mutates $c(v)$ exactly once, there exists a path P starting with edge e_1 and ending with e_3 both mutating c', and containing edge e_2 mutating $c(v)$. Furthermore, the path P contains no other mutations of $c(v)$ or c'. At the end of the current iteration, M_j is replaced with $M0$ and $M1$. Both subtrees of $T_{M_j}^*$ containing $M0$ and $M1$ contain (at least) one mutation of c' each. Therefore, $\texttt{penalty}(M0) + \texttt{penalty}(M1) < \texttt{penalty}(M_j)$. Since $\texttt{penalty}(I) \leq q$, there can be at most q iterations of the while loop.

We now bound the probability. Intuitively, if $|\cup_i N(M_i)|$ is very large, then the probability of a correct guess is large, since at most q out of $|\cup_i N(M_i)|$ characters can mutate multiple times in $T_{M_j}^*$. On the other hand if $|\cup_i N(M_i)| = q$ then we terminate the loop. Formally, at each iteration $|\cup_i N(M_i)|$ reduces by at least 1 (guessed vertex v is no longer in $\cup_i N(M_i)$). Therefore, in the worst case (to minimize the probability of correct guesses), we can have q iterations of the loop, with $q+1$ non-isolated vertices in the last iteration and $2q$ in the first iteration. The probability in such a case that all guesses are correct is at least

$$(\frac{q}{2q}) \times (\frac{q-1}{2q-1}) \times \ldots \times (\frac{1}{q+1}) = \frac{1}{\binom{2q}{q}} \geq 2^{-2q}$$

□

Buneman Graphs. We now show that r, p can be found efficiently. To prove this we need some tools from the theory of Buneman graphs [21].

Let M be a set of taxa defined by character set C of size m. A Buneman graph F for M is a vertex induced subgraph of the m-cube. Graph F contains vertices v i.f.f. for every pair of characters $i, j \in C$, $(v[i], v[j]) \in G_{i,j}$. Recall that $G_{i,j}$ is the set of gametes (or projection of M on dimensions i, j). Each edge of the Buneman graph is labeled with the character at which the adjacent vertices differ.

Buneman graphs have been defined in previous works on matrices M in which no two characters share exactly two gametes. The definition can be extended to allow such characters while preserving the following lemmas (see expanded version for details). We say that a subgraph F' of F is the same as an edge labeled tree T if F' is a tree and T can be obtained from F' by suppressing degree-two vertices. A phylogeny T is contained in a graph F if there exists an edge-labeled subgraph F' that is the same as the edge labeled (by function μ) phylogeny T. A Buneman graph F for input M has the property that every optimum phylogeny for M is contained in F [21]. From the definition of the Buneman graph F, we know that there exists no vertex $v \in F$ for which $(v[i], v[j]) \notin G_{i,j}$. Therefore, using the above property, we have:

Lemma 4. *In every optimum phylogeny T^*_M, the conflict graph on the set of taxa in T^*_M (Steiner vertices included) is the same as the conflict graph on M.*

Lemma 5. *The probability that all guesses performed at Step 2c are correct is at least 2^{-q}.*

Proof. Implementation: We first show how to perform the guess efficiently. For every character i, we perform the following steps in order.

1. if all taxa in $M0$ contain the same state s in i, then fix $r[i] = s$
2. if all taxa in $M1$ contain the same state s in i, then fix $r[i] = s$
3. if $r[i]$ is unfixed then guess $r[i]$ uniformly at random from $\{0, 1\}$

Assuming that the guess at Step 2a (Figure 1) is correct, we know that there exists an optimum phylogeny $T^*_{M_j}$ on M_j where $c(v)$ mutates exactly once. Let $e \in T^*_{M_j}$ s.t. $c(v) \in \mu(e)$. Let r' be an end point of e s.t. $r'[c(v)] = 1$ and p'

be the other end point. If the first two conditions hold with the same state s, then character i does not mutate in M_j. In such a case we know that $r'[i] = s$, since $T^*_{M_j}$ is optimal and the above method ensures that $r[i] = s$. Notice that if both conditions are satisfied simultaneously with different values of s then i and $c(v)$ share exactly two gametes in M_j and therefore $i, c(v) \in \mu(e)$. Hence, $r'[i] = r[i]$. We now consider the remaining cases when exactly one of the above conditions hold. We show that if $r[i]$ is fixed to s then $r'[i] = s$. Note that in such a case at least one of $M0, M1$ contain both the states on i and $i, c(v)$ share at least 3 gametes in M_j. The proof can be split into two symmetric cases based on whether r is fixed on condition 1 or 2. One case is presented below:

Taxon $r[i]$ is fixed based on condition 1: In this case, all the taxa in $M0$ contain the same state s on i. Therefore, the taxa in $M1$ should contain both states on i. Hence i mutates in $T^*_{M_j}(c(v), 1)$. For the sake of contradiction, assume that $r'[i] \neq s$. If $i \notin \mu(e)$ then $p'[i] \neq s$. However all the taxa in $M0$ contain state s. This implies that i mutates in $T^*_{M_j}(c(v), 0)$ as well. Therefore i and $c(v)$ share all four gametes on $T^*_{M_j}$. However i and $c(v)$ share at most 3 gametes in M_j - one in $M0$ and at most two in $M1$. This leads to a contradiction to Lemma 4. Once r is guessed correctly, p can be computed since it is is identical to r in all characters except $c(v)$ and those that share two gametes with $c(v)$ in M_j. We make a note here that we are assuming that e does not mutate any character that does not share two gametes with $c(v)$ in M_j. This creates a small problem that although the length of the tree constructed is optimal, r and p could be degree-two Steiner vertices. If after constructing the optimum phylogenies for $M0$ and $M1$, we realize that this is the case, then we simply add the mutation adjacent to r and p to the edge (r, p) and return the resulting phylogeny where both r and p are not degree-two Steiner vertices.

The above implementation therefore requires only guessing states corresponding to the remaining unfixed characters of r. If a character i violates the first two conditions, then i mutates once in $T^*_{M_j}(i, 0)$ and once in $T^*_{M_j}(i, 1)$. If $r[i]$ has not been fixed, then we can associate a pair of mutations of the same character i with it. At the end of the current iteration M_j is replaced with $M0$ and $M1$ and each contains exactly one of the two associated mutations. Therefore if q' characters are unfixed then $\texttt{penalty}(M0) + \texttt{penalty}(M1) \leq \texttt{penalty}(M_j) - q'$. Since $\texttt{penalty}(I) \leq q$, throughout the execution of the algorithm there are q unfixed states. Therefore the probability of all the guesses being correct is 2^{-q}. □

This completes our analysis for upper bounding the probability that the algorithm returns an optimum phylogeny. We now analyze the running time. We use the following lemma to show that we can efficiently construct optimum phylogenies at Step 3 in the pseudo-code.

Lemma 6. *For a set of taxa M, if the number of non-isolated vertices of the associated conflict graph is t, then an optimum phylogeny T^*_M can be constructed in time $O(3^s 6^t + nm^2)$, where $s = \texttt{penalty}(M)$.*

Proof. We use the approach described by Gusfield and Bansal (see Section 7 of [14]) that relies on the Decomposition Optimality Theorem for recurrent mutations. We first construct the conflict graph and identify the non-trivial connected components of it in time $O(nm^2)$. Let κ_i be the set of characters associated with component i. We compute the Steiner minimum tree T_i for character set κ_i. The remaining conflict-free characters in $C \setminus \cup_i \kappa_i$ can be added by contracting each T_i to vertices and solving the perfect phylogeny problem using Gusfield's linear time algorithm [12].

Since $\texttt{penalty}(M) = s$, there are at most $s+t+1$ distinct bit strings defined over character set $\cup_i \kappa_i$. The Steiner space is bounded by 2^t, since $| \cup_i \kappa_i| = t$. Using the Dreyfus-Wagner recursion [19] the total run-time for solving all Steiner tree instances is $O(3^{s+t}2^t)$. □

Lemma 7. *The algorithm described solves the BNPP problem in time $O(18^q + qnm^2)$ with probability at least 8^{-q}.*

Proof. For a set of taxa $M_i \in L$ (Step 3, Figure 1), using Lemma 6 an optimum phylogeny can be constructed in time $O(3^{s_i}6^{t_i} + nm^2)$ where $s_i = \texttt{penalty}(M_i)$ and t_i is the number of non-isolated vertices in the conflict graph of M_i. We know that $\sum_i s_i \leq q$ (since $\texttt{penalty}(I) \leq q$) and $\sum_i t_i \leq q$ (stopping condition of the `while` loop). Therefore, the total time to reconstruct optimum phylogenies for all $M_i \in L$ is bounded by $O(18^q + qnm^2)$. The running time for the while loop is bounded by $O(qnm^2)$. Therefore the total running time of the algorithm is $O(18^q + qnm^2)$. Combining Lemmas 3 and 5, the total probability that all guesses performed by the algorithm is correct is at least 8^{-q}. □

Lemma 8. *The algorithm described above can be derandomized to run in time $O(72^q + 8^q nm^2)$.*

Proof. It is easy to see that Step 2c can be derandomized by exploring all possible states for the unfixed characters. Since there are at most q unfixed characters throughout the execution, there are 2^q possibilities for the states.

However, Step 2a cannot be derandomized naively. We use the technique of bounded search tree [5] to derandomize it efficiently. We select an arbitrary vertex v from $\cup_i N(M_i)$. We explore both the possibilities on whether v mutates once or multiple times. We can associate a search (binary) tree with the execution of the algorithm, where each node of the tree represents a selection v from $\cup_i N(M_i)$. One child edge represents the execution of the algorithm assuming v mutates once and the other assuming v mutates multiple times. In the execution where v mutates multiple times, we select a different vertex from $\cup_i N(M_i)$ and again explore both paths. The height of this search tree can be bounded by $2q$ because at most q characters can mutate multiple times. The path of height $2q$ in the search tree is an interleaving of q characters that mutate once and q characters that mutate multiple times. Therefore, the size of the search tree is bounded by 4^q.

Combining the two results, the algorithm can be derandomized by solving at most 8^q different instances of Step 3 while traversing the while loop 8^q times for a total running time of $O(144^q + 8^q nm^2)$. This is, however, an over-estimate. Consider any iteration of the while loop when M_j is replaced with $M0$ and $M1$. If a state in character c is unfixed and therefore guessed, we know that there are two associated mutations of character c in both $M0$ and $M1$. Therefore at iteration i, if q'_i states are unfixed, then $\texttt{penalty}(M0) + \texttt{penalty}(M1) \leq \texttt{penalty}(M_j) - q'_i$. At the end of the iteration we can reduce the value of q used in Step 2 by q'_i, since the penalty has reduced by q'_i. Intuitively this implies that if we perform a total of q' guesses (or enumerations) at Step 2c, then at Step 3 we only need to solve Steiner trees on $q - q'$ characters. The additional cost $2^{q'}$ that we incur results in reducing the running time of Step 3 to $O(18^{q-q'} + qnm^2)$. Therefore the total running time is $O(72^q + 8^q nm^2)$. □

4 Discussion and Conclusions

Discussion: If all Steiner tree problem instances on the q-cube are solved in a pre-processing step, then our running time just depends on the number of iterations of the while loop, which is $O(8^q nm^2)$. Such pre-processing would be impossible to perform with previous methods. Alternate algorithms for solving Steiner trees may be faster in practice as well.

In Lemma 8, we showed that the guesses performed at Step 2c do not affect the overall running time. We can also establish a trade-off along similar lines for Step 2a that can reduce the theoretical run-time bounds. Details of such trade-offs will be analyzed in the expanded version.

Conclusions: We have presented an algorithm to solve the BNPP problem that is theoretically superior to existing methods. In an empirical evaluation [20], the prior algorithm reconstructed optimum phylogenies for values of q up to 7. Our algorithm should solve for larger values of q since it is clearly expected to out-perform prior methods and its own worst case guarantees. The algorithm is intuitive and simple and hence is one of the few theoretically sound phylogenetic tree reconstruction algorithms that is also expected to be practical.

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 *International Colloquium on Automata, Languages and Programming* (1992).
3. H. Bodlaender, M. Fellows, M. Hallett, H. Wareham and T. Warnow. The Hardness of Perfect Phylogeny, Feasible Register Assignment and Other Problems on Thin Colored Graphs. In *Theoretical Computer Science* (2000).
4. M. Bonet, M. Steel, T. Warnow and S. Yooseph. Better Methods for Solving Parsimony and Compatibility. In *Journal of Computational Biology*, 5(3) (1992).

5. R.G. Downey and M. R. Fellows. Parameterized Complexity. *Monographs in Computer Science* (1999).
6. W. H. Day and D. Sankoff. Computational Complexity of Inferring Phylogenies by Compatibility. *Systematic Zoology* (1986).
7. P. Damaschke. Parameterized Enumeration, Transversals, and Imperfect Phylogeny Reconstruction. In proc *International Workshop on Parameterized and Exact Computation* (2004).
8. E. Eskin, E. Halperin and R. M. Karp. Efficient Reconstruction of Haplotype Structure via Perfect Phylogeny. In *Journal of Bioinformatics and Computational Biology* (2003).
9. D. Fernandez-Baca and J. Lagergren. A Polynomial-Time Algorithm for Near-Perfect Phylogeny. In *SIAM Journal on Computing*, 32 (2003).
10. L. R. Foulds and R. L. Graham. The Steiner problem in Phylogeny is NP-complete. In *Advances in Applied Mathematics* (3) (1982).
11. G. Ganapathy, V. Ramachandran and T. Warnow. Better Hill-Climbing Searches for Parsimony. In *Workshop on Algorithms in Bioinformatics* (2003).
12. D. Gusfield. Efficient Algorithms for Inferring Evolutionary Trees. In *Networks*, 21 (1991).
13. D. Gusfield. Algorithms on Strings, Trees and Sequences. *Cambridge University Press* (1999).
14. D. Gusfield and V. Bansal. A Fundamental Decomposition Theory for Phylogenetic Networks and Incompatible Characters. In proc *Research in Computational Molecular Biology* (2005).
15. D. Gusfield, S. Eddhu and C. Langley. Efficient Reconstruction of Phylogenetic Networks with Constrained Recombination. In Proc *IEEE Computer Society Bioinformatics Conference* (2003).
16. D. A. Hinds, L. L. Stuve, G. B. Nilsen, E. Halperin, E. Eskin, D. G. Ballinger, K. A. Frazer, D. R. Cox. Whole Genome Patterns of Common DNA Variation in Three Human Populations. `www.perlegen.com`. In *Science* (2005).
17. The International HapMap Consortium. The International HapMap Project. `www.hapmap.org`. *Nature* 426 (2003).
18. S. Kannan and T. Warnow. A Fast Algorithm for the Computation and Enumeration of Perfect Phylogenies. In *SIAM Journal on Computing*, 26 (1997).
19. H. J. Promel and A. Steger. The Steiner Tree Problem: A Tour Through Graphs Algorithms and Complexity. *Vieweg Verlag* (2002).
20. S. Sridhar, K. Dhamdhere, G. E. Blelloch, E. Halperin, R. Ravi and R. Schwartz. Simple Reconstruction of Binary Near-Perfect Phylogenetic Trees. In proc *International Workshop on Bioinformatics Research and Applications* (2006).
21. C. Semple and M. Steel. Phylogenetics. *Oxford University Press* (2003).
22. M. A. Steel. The Complexity of Reconstructing Trees from Qualitative Characters and Subtrees. In *J. Classification*, 9 (1992).