Mixed Integer Linear Programming for Maximum Parsimony Phylogeny Inference

Srinath Sridhar, Fumei Lam, Guy E. Blelloch, R. Ravi and Russell Schwartz

Abstract

Reconstruction of phylogenetic trees is a fundamental problem in computational biology. While excellent heuristic methods are available for many variants of this problem, new advances in phylogeny inference will be required if we are to be able to continue to make effective use of the rapidly growing stores of variation data now being gathered. In this paper, we present two integer linear programming (ILP) formulations to find the most parsimonious phylogenetic tree from a set of binary variation data. One method uses a flow-based formulation that can produce exponential numbers of variables and constraints in the worst case. The method has, however, proven extremely efficient in practice on datasets that are well beyond the reach of the available provably efficient methods, solving several large mtDNA and Y-chromosome instances within a few seconds and giving provably optimal results in times competitive with fast heuristics than cannot guarantee optimality. An alternative formulation establishes that the problem can be solved with a polynomial-sized ILP. We further present a web server developed based on the exponential-sized ILP that performs fast maximum parsimony inferences and serves as a front end to a database of precomputed phylogenies spanning the human genome.

Index Terms

computational biology, algorithms, integer linear programming, steiner tree problem, phylogenetic tree reconstruction, maximum parsimony

S. Sridhar and G. E. Blelloch are with the Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave., Pitsburgh, PA 15213 U.S.A. Email: {srinath,blelloch}@cs.cmu.edu.

F. Lam is with the Computer Science Department, Brown University. Email:lam@cs.brown.edu

R. Ravi is with the Tepper School of Business, Carnegie Mellon University. Email: ravi@cmu.edu

R. Schwartz is with the Department of Biological Sciences, Carnegie Mellon University. Email: russells@andrew.cmu.edu

I. INTRODUCTION

Phylogeny construction, or the inference of evolutionary trees from some form of population variation data, is one of the oldest and most intensively studied problems in computational biology. Yet it remains far from solved. The problem has become particularly acute for the special case of intraspecies phylogenetics, or tokogenetics, in which we wish to build evolutionary trees among individuals in a single species. In part, the persistence of the problem reflects its basic computational difficulty. The problem in most reasonable variants is formally NP hard [1] and thus has no known efficient solution. The continuing relevance of phylogeny inference algorithms also stems from the fact that the data sets to be solved have been getting increasingly large in both population sizes and numbers of variations examined. The genomic era has led to the identification of vast numbers of variant sites for human populations [2], [3], as well as various other complex eukaryotic organisms [4], [5], [6]. Large-scale resequencing efforts are now under way to use such sites to study population histories with precision never previously possible [7]. Even more vast data sets are available for microbial and viral genomes. As a result, methods that were adequate even a few years ago may no longer be suitable today.

In this work, we focus on the inference of intraspecies phylogenies on binary genetic variation data, which is of particular practical importance because of the large amount of binary SNP data now available. The binary intraspecies phylogeny problem has traditionally been modeled by the minimum Steiner tree problem on binary sequences, a classic NP hard problem [1]. Some special cases of the problem are efficiently solvable, most notably the case of *perfect phylogenies*, in which each variant site mutates only once within the optimal tree [8], [9], [10]. However, real data will not, in general, conform to the perfect phylogeny assumption. The standard in practice is the use of sophisticated heuristics that will always produce a tree but cannot guarantee optimality (e.g., [11], [12], [13]). Some theoretical advances have recently been made in the efficient solution of *near-perfect phylogenies*, those that deviate only by a fixed amount from the assumption of perfection [14], [15], [16], [17]. These methods can provide provably efficient solutions in many instances, but still struggle with some moderate-size data sets in practice. As a result, some recent attention has turned to integer linear programming (ILP) methods [18]. ILPs provide provably optimal solutions and while they do not provide guaranteed run-time bounds, they may have practical run times far better than those of the provably efficient methods.

In the present work, we develop two ILP formulations to solve the most parsimonious phylogenetic tree problem on binary sequences. These methods find provably optimal trees from real binary sequence data, much like the prior theoretical methods and unlike the prevailing heuristic methods. Practical run time is, however, substantially lower than that of the existing provably efficient theoretical methods, allowing us to tackle larger and more difficult datasets. Below, we formalize the problem solved, present our methods, and establish their practical value on a selection of real variation data sets. We further document a web server providing open access to this ILP method and serving as a front-end to a database of local phylogenies inferred throughout the autosomal human genome. This work provides a platform for more extensive empirical studies of variation patterns on genomic scales than were previously possible and may also help lay the groundwork for more sophisticated optimization methods that are likely to be needed in the future.

II. DEFINITIONS

We will assume that the input to the problem is a haplotype matrix H where each row corresponds to a haploid sequence of a taxon and each column corresponds to a binary marker such as a Single Nucleotide Polymorphism (SNP). The input H can therefore be viewed as an $n \times m$ binary matrix.

Definition 1: A *phylogeny* T for input I is a tree where each vertex represents a binary string in $\{0,1\}^m$ and all the input sequences are represented in T. The *length* of T is the sum of the Hamming distances between all the adjacent vertices. The problem of constructing the most parsimonious (optimal) phylogeny is to find the phylogeny T^* such that $\texttt{length}(T^*)$ is minimized.

Definition 2: A phylogeny T for input I with m varying sites is q-near-perfect (or q-imperfect) if $\texttt{length}(T) = m + q$.

The problem of reconstructing phylogenies is closely related to the *Steiner Tree Problem*, a well studied problem in combinatorial optimization (for a survey and applications, see [19], [20]). Given a graph $G = (V, E)$ and a set of *terminals* in V, the problem is to find the smallest subgraph of G such that there is a path between any pair of terminals.

The problem can be related to the phylogeny construction problem as follows. Let graph G be the m-cube defined on vertices $V = \{0,1\}^m$ and edges $E = \{(u,v) \in V \times V : \sum_i |u_i - v_i| = 1\}$.

The vertices are binary strings of length m and an edge connects two vertices if and only if their Hamming distance is 1. Let $V_T \subseteq V$ be the set of species corresponding to the rows of input matrix H. The maximum parsimony problem is then equivalent to the minimum Steiner tree problem on underlying graph G with terminal vertices V_T. Even in this restricted setting, the Steiner tree problem has been shown to be NP-complete [21]. However, the phylogeny reconstruction problem when the optimal phylogeny is q-near-perfect can be solved in time polynomial in n and m when $q = O(\log(\text{poly}(n, m)))$ [17]. If q is very large, though, such algorithms do not perform well. Moreover, these algorithms use a sub-routine that solves the Steiner tree problem on m-cubes when the dimensions are small. Therefore, improving the existing solutions for the general problem will also improve the running time for the restricted cases.

III. PREPROCESSING

We now describe a set of preprocessing steps that can substantially reduce the size of the input data without affecting the final output.

A. Reducing the set of possible Steiner vertices

The complexity of solving the Steiner tree problem in general graphs is a consequence of the exponentially many possible subsets that can be chosen as the final set of Steiner vertices in the most parsimonious phylogeny. Therefore, an important component of any computational solution to the Steiner tree problem is to eliminate vertices that cannot be present in any optimal tree. We describe an approach that has been used to eliminate such vertices when the underlying graph is the m-cube.

For input graph H and column c of H, the *split* $c(0)|c(1)$ defined by c is a partition of the taxa into two sets, where $c(0)$ is the set of taxa with value 0 in column c and $c(1)$ is the set of taxa with value 1 in column c. This forms a partition of the taxa since $c(0) \cup c(1)$ is the set of all taxa and $c(0) \cap c(1)$ is empty. Each of $c(0)$ and $c(1)$ is called a *block* of c. Buneman used the blocks of binary taxa to introduce a graph, now called the *Buneman graph* $\mathcal{B}(H)$, which captures structural properties of the optimal phylogeny [22]. We will explain the generalization of this graph due to Barthélemy [23]. Each vertex of the Buneman graph is an m-tuple of blocks $[c_1(i_1), c_2(i_2), \ldots c_m(i_m)]$ ($i_j = 0$ or 1 for each $1 \leq j \leq m$), with one block for each column and

```
function findBuneman($V_T$)
    1) let $\lambda \leftarrow V_T$; let $v \in \lambda$
    2) bunemanNeighbor($\lambda, v$)
function bunemanNeighbor($\lambda, v$)
    1) for all $j \in \{1, \ldots, m\}$
        a) let $v' \leftarrow v; v'_j \leftarrow c_j(1 - i_j)$
        b) if $v'$ is Buneman and $v' \notin \lambda$ then
            i) $\lambda \leftarrow \lambda \cup \{v'\}$
            ii) bunemanNeighbor($\lambda, v'$)
```

Fig. 1. Finding the Buneman graph in polynomial time

such that each pair of blocks has nonempty intersection ($c_j(i_j) \cap c_k(i_k) \neq \emptyset$ for all $1 \leq j, k \leq m$). There is an edge between two vertices in $\mathcal{B}(H)$ if and only if they differ in exactly one block. Buneman graphs are very useful because of the following theorem.

Theorem 3.1: [11], [24] For input matrix H, let T_H^* denote the optimal phylogeny on H and let $\mathcal{B}(H)$ denote the Buneman graph on H. If matrix H has binary values, then every optimal phylogeny T_H^* is a subgraph of $\mathcal{B}(H)$.

Using the above theorem, our problem is now reduced to constructing the Buneman graph on input H and solving our problem on underlying graph $\mathcal{B}(H)$. Ideally we would like to find the Buneman graph in time $O(\text{poly}(k))$ where k is the number of vertices in the Buneman graph. Note that this is output-sensitive. We first state the following theorem, which we will use to show the Buneman graph can be generated efficiently.

Theorem 3.2: [24] The Buneman graph $\mathcal{B}(H)$ is connected for any input matrix H in which all columns contain both states $0, 1$ and all pairs of columns are distinct.

To generate the graph $\mathcal{B}(H)$, let $i_1, i_2, \ldots i_m$ be the first taxon in H. Then $v = [c_1(i_1), c_2(i_2), \ldots c_m(i_m)]$ is a vertex of $\mathcal{B}(H)$. Now, there are several ways to generate the graph $\mathcal{B}(H)$. The pseudo-code in Figure 1 begins with V_T the set of vertices of the $\mathcal{B}(H)$ corresponding to H. The algorithm then iteratively selects a vertex v and enumerates all the neighbors. For each vertex, the algorithm checks if it obeys the conditions of the Buneman graph, if so it is added to λ and we recurse.

Lemma 3.3: The algorithm in Figure 1 finds the Buneman graph $\mathcal{B}(H)$ for the given input in time $O(km)$ where k is the number of vertices in $\mathcal{B}(H)$.

Proof: The algorithm begins with a vertex $v \in \mathcal{B}(H)$ and determines $\mathcal{B}(H)$ in the depth-first search order. By Theorem 3.2, the algorithm will visit all vertices in $\mathcal{B}(H)$. Step 1a iterates over all m possible neighbors of vertex v in the m-cube which takes time $O(m)$. For each vertex $v \in \mathcal{B}(H)$ function `bunemanNeighbor` is called using v exactly once. Therefore if there are k vertices in $\mathcal{B}(H)$, then the time spent to discover all of $\mathcal{B}(H)$ is $O(km)$. Note that instead of using depth-first search, we could use breadth-first search or any other traversal order. ■

B. Decomposition into smaller problems

In addition to allowing us to reduce the set of possible Steiner vertices, we show how Theorem 3.1 also allows us to decompose the problem into independent subproblems.

Definition 3: [25] A pair of columns i, j *conflict* if the matrix H restricted to i, j contains all four gametes $(0,0), (0,1), (1,0)$ and $(1,1)$. Equivalently, the columns *conflict* if the projection of H onto dimensions i, j contains all four points of the square.

For input I, the structure of the conflicts of I provides important information for building optimal phylogenies for I. For example, it is well known that a perfect phylogeny exists if and only if no pair of columns conflict [9], [24]. In order to represent the conflicts of H, we construct the *conflict graph* $\mathcal{G}$, where the vertices of $\mathcal{G}$ are columns of H and the edges of $\mathcal{G}$ correspond to pairs of conflicting columns [26]. The following theorem has been stated previously without proof [26]. For the sake of completeness, we provide an explicit proof using Theorem 3.1 and ideas from Gusfield and Bansal [26]. We denote the matrix H restricted to set of columns C as $C(H)$.

Theorem 3.4: Let χ denote the set of non-trivial connected components of conflict graph $\mathcal{G}$ and let V_{isol} denote the set of isolated vertices of $\mathcal{G}$. Then any optimal Steiner tree on H is a union of optimal Steiner trees on the separate components of $\mathcal{G}$ and $\texttt{length}(T^*_H) = |V_{isol}| + \sum_{C \in \chi} \texttt{length}(T^*_{C(H)})$.

Proof:

We use the fact that the optimal phylogeny is contained in the Buneman graph and show that the connected components impose restrictions on the set of possible edges in the Buneman graph. For two columns c and c', the block $c(i)$ is the *dominated block* of c with respect to the pair (c, c') if block $c(i)$ is contained in some block of c' (i.e., $c(i) \subset c'(0)$ or $c(i) \subset c'(1)$).

Similarly, block $c(i)$ is the *dominating block* of c with respect to the pair (c, c') if $c(i)$ contains some block of c'.

Let C be a component in $\chi \cup V_{isol}$. If C is the only component in $\mathcal{G}$, the theorem follows immediately. Otherwise, we can reorder the columns so that C consists of the first k columns, i.e., $c_1, c_2, \ldots c_k \in C$ and $c_{k+1}, \ldots c_m \notin C$. Recall that for any edge in the Buneman graph $\mathcal{B}(H)$, its endpoints correspond to two m-tuples of blocks which differ in exactly one column; label this edge by the column for which its endpoints differ. For any collection of columns $\alpha_1, \alpha_2, \ldots \alpha_l$, let $T^*_H[\alpha_1, \alpha_2, \ldots \alpha_l]$ denote the subgraph of T^*_H induced by the set of edges labeled by $\alpha_1, \alpha_2, \ldots \alpha_l$. We will characterize all edges in the Buneman graph labeled by columns in C using the following lemma from Gusfield and Bansal [26].

Lemma 3.5: [26] For a column c_i with $i > k$, c_i does not conflict with any column in connected component C, and therefore, exactly one of $c_i(0)$ or $c_i(1)$ is the dominating block in c_i with respect to *every* column in C.

Let $c_i(l_i)$ ($i > k$) denote the set of dominating blocks of c_i with respect to C. (It follows that $c_i(1 - l_i)$ is the dominated block in c_i with respect to every column in C).

Any vertex in the Buneman graph is an m-tuple of blocks which have pairwise nonempty intersection. Therefore, an edge e labeled by a column in C, say c_1, must have endpoints in which the blocks of column $c_{k+1}, c_{k+2}, \ldots c_m$, intersect both $c_1(0)$ and $c_1(1)$. This implies the blocks of $c_{k+1}, c_{k+2}, \ldots c_m$ are forced to be the dominating blocks with respect to component C, i.e., the last $m - k$ coordinates of the endpoints of e must be $c_{k+1}(l_{k+1}), c_{k+2}(l_{k+2}) \ldots c_m(l_m)$. Let $\mathcal{B}(C)$ be the subgraph of $\mathcal{B}(H)$ generated by the vertices whose last $m - k$ columns have this form. Then any edge labeled by a column in C has both endpoints in $\mathcal{B}(C)$.

Lemma 3.6: $T^*_H[C] = T^*_H[c_1, c_2, \ldots c_k]$ is an optimal Steiner tree on $\mathcal{B}(C)$.

Proof: We say that vertex $v \in \mathcal{B}(C)$ is a C-projected terminal vertex if there exists $h \in H$ with the same states as v in columns of C. We first show that any two terminals in $\mathcal{B}(C)$ that are C-projected vertices are connected by a path in $T^*_H[c_1, c_2, \ldots c_k]$. Suppose otherwise and let v_1 and v_2 be two distinct vertices in $\mathcal{B}(C)$ which are not connected by such a path. By definition of T^*_H, there is a path P in T^*_H connecting v_1 to v_2; we can assume that v_1 and v_2 are chosen so that the length of path P is minimized. Let $d_1, d_2, \ldots d_l$ denote the edge labels of P (by assumption, at least one of $d_1, d_2, \ldots d_l$ is not in $\{c_1, c_2, \ldots c_k\}$). If for some i, we have $d_i \in \{c_1, c_2, \ldots c_k\}$, then the endpoints u and w of d_i are in $\mathcal{B}(C)$, and either v_1, u or w, v_2 is a pair that is not connected in $T_H^*[c_1, c_2, \dots c_k]$, a contradiction to the choice of vertices v_1, v_2.

Therefore, all edge labels d_i are in the set $\{c_{k+1}, c_{k+2}, \dots c_m\}$. However, since v_1 and v_2 are in $\mathcal{B}(C)$, the final $m-k$ components of these two vertices are $c_{k+1}(l_{k+1}), c_{k+2}(l_{k+2}) \dots c_m(l_m)$ by definition. Finally, since there are no edges in P labeled by $c_1, c_2, \dots c_k$, it follows that v_1 and v_2 are equal in all components, a contradiction.

Therefore, $T_H^*[c_1, c_2, \dots c_k]$ is a Steiner tree on $\mathcal{B}(C)$ where the set of terminal vertices are the C-projected terminal vertices. Therefore if T_H^* is not optimal, then by removing $T_H^*[c_1, c_2, \dots c_k]$ from T_H^* and replacing it by a tree of smaller cost, we obtain a Steiner tree for H with smaller cost than T_H^*, a contradiction. ■

The terminal vertices of $C(H)$ correspond to C-projected terminal vertices of $\mathcal{B}(H)$. Therefore, the above shows that for every connected component C, $T_{C(H)}^*$ is a subgraph of T_H^*. Therefore,

$$\texttt{length}(T_H^*) = \sum_{C \in \chi \cup V_{isol}} \texttt{length}(T_{C(H)}^*) = |V_{isol}| + \sum_{C \in \chi} \texttt{length}(T_{C(H)}^*)$$

This completes the proof of Theorem 3.4. ■

Our decomposition preprocessing step proceeds as follows. We first construct the conflict graph $\mathcal{G}$ for input matrix H and identify the set of connected components of $\mathcal{G}$. We ignore the columns corresponding to the isolated vertices V_{isol} since they each contribute exactly one edge to the final phylogeny. Then the columns corresponding to each connected component c of χ can be used independently to solve for the most parsimonious phylogeny. Our problem is now reduced to input matrices H consisting of a single non-trivial connected component.

C. Merging Rows and Columns

We now transform the input matrix H to possibly reduce its size. We can remove rows of H until all the rows are distinct since this does not change the phylogeny. Furthermore, we can remove all the columns of H that do not contain both states 0 and 1 since such columns will not affect the size or the topology of the phylogeny. Finally, we will assign weights w_i to column i; w_i is initialized to 1 for all i. We iteratively perform the following operation: identify columns i and j that are identical (up to relabeling 0, 1), set $w_i := w_i + w_j$ and remove column j from the matrix. Notice that in the final matrix H, all pair-wise rows are distinct, all pair-wise columns are distinct (even after relabeling 0, 1), every column contains both 0, 1 and all the columns have weights $w_i \geq 1$. From now, the input to the problem consists of the matrix H along with vector w containing the weights for the columns of H. We can now redefine the length of a phylogeny using a weighted Hamming distance as follows.

Definition 4: The `length` of phylogeny $T(V, E)$ is
$\texttt{length}(T) = \sum_{(u,v)\in E} \sum_{i\in D(u,v)} w_i$, where $D(u, v)$ is the set of indices where u, v differ.

It is straight-forward to prove the correctness of the pre-processing step.

Lemma 3.7: The length of the optimal phylogeny on the pre-processed input is the same as that of the original input.

IV. ILP FORMULATION

A common approach for studying the minimum Steiner tree problem is to use integer and linear programming methods. For convenience, we will consider the more general problem of finding a minimum Steiner tree for directed weighted graphs G (we represent an undirected graph as a directed graph by replacing each edge by two directed edges). The input to the minimum directed Steiner tree problem is a directed graph, a set of terminals T and a specified root vertex $r \in T$. The minimum Steiner tree is the minimum cost subgraph containing a directed path from r to every other terminal in T.

For a subgraph S of graph G, we associate a vector $x^S \in \mathbb{R}^E$, where edge variable x^S_e takes value 1 if e appears in the subgraph S and 0 otherwise. A subset of vertices $U \subset V$ is *proper* if it is nonempty and does not contain all vertices. For $U \subset V$, let $\delta^+(U)$ denote the set of edges (u, v) with $u \in U$, $v \notin U$ and for a subset of edges $F \subseteq E$, let $x(F) = \sum_{e\in F} x_e$. Finally, edge-weights are given by $w_e \in R^E$.

The problem of finding a minimum directed Steiner tree rooted at r has previously been examined with an ILP based on graph cuts [27], [28], [29]:

$$\min \quad \sum_{u,v} w_{u,v} x_{u,v} \quad \text{subject to} \tag{1}$$

$$x(\delta^+(U)) \geq 1 \quad \forall \text{ proper } U \subset V \text{ with } r \in U,\ T \cap \overline{U} \neq \emptyset \tag{2}$$

$$x_{u,v} \in \{0, 1\} \quad \text{for all } (u, v) \in E. \tag{3}$$

Constraints (2) impose that r has a directed path to all terminal vertices T. Note that in our phylogenetic tree reconstruction problem, the underlying graph for the problem is the Buneman graph and any input taxon can be chosen as the root vertex r. Since the Buneman graph may have an exponential number of vertices and edges with respect to the size of the input matrix H, the running time for solving this integer program may be doubly-exponential in m in the worst case.

We develop an alternative formulation based on multicommodity flows [29]. In this formulation, one unit of flow is sent from the root vertex to every terminal vertex. Every terminal vertex except the root acts as a sink for one unit of flow and the Steiner vertices have perfect flow conservation. We use two types of variables, $f^t_{u,v}$ and $s_{u,v}$, for each edge $(u, v) \in E$. The variables $f^t_{u,v}$ are real valued and represent the amount of flow along edge (u, v) whose destination is terminal t. Variables $s_{u,v}$ are binary variables denoting the presence or absence of edge (u, v). The program is then the following:

$$\min \quad \sum_{u,v} w_{u,v} s_{u,v} \quad \text{subject to} \tag{4}$$

$$\sum_v f^t_{u,v} = \sum_v f^t_{v,u} \quad \text{for all } u \notin T \tag{5}$$

$$\sum_v f^t_{v,t} = 1, \sum_v f^t_{t,v} = 0, \sum_v f^t_{r,v} = 1 \quad \text{for all } t \in T \tag{6}$$

$$0 \leq f^t_{u,v} \leq s_{u,v} \quad \text{for all } t \in T \tag{7}$$

$$s_{u,v} \in \{0, 1\} \quad \text{for all } e \in E. \tag{8}$$

Constraints (5) impose the condition of flow conservation on the Steiner vertices. Constraints (6) impose the inflow/outflow constraints on terminals in T. Finally, constraints (7) impose the condition that there is positive flow on an edge only if the edge is selected. By the max-flow min-cut theorem, the projection of the solution onto the variables s satisfy constraints (2) [28]. The results will thus satisfy the following theorem:

Theorem 4.1: All integer variables of the above linear program are binary and the solution to the ILP gives a most parsimonious phylogenetic tree.

V. ALTERNATIVE POLYNOMIAL-SIZED ILP FORMULATION

The preceding ILP requires in the worst case an exponentially large number of variables and constraints. It is, however, possible to formulate this problem with only a polynomial number (in n and m) of variables and constraints. The exponential-sized ILP ultimately proved more efficient in practice than the polynomial-sized ILP and we therefore used that one for our empirical validation. We nonetheless include this alternative formulation because it may prove more promising for future improvements and extensions to more general cases of the Steiner tree problem than will our exponential-sized ILP. Note that preprocessing operations B and C above for the exponential-sized ILP will also be relevant to the polynomial-sized ILP. We will therefore assume we have performed those proprocessing steps and in particular that we have eliminated all redundant rows and columns in the data set.

We will use $h_{i,j}, 1 \leq i \leq n$ to denote the state of the i^{th} taxon at site j of the input matrix H. Note that these are not variables of the linear program. We will use $h_{i,j}, n+1 \leq i \leq 2n$ to represent the state of the i^{th} Steiner vertex at site j. We will therefore use nm such variables in the ILP.

However, T^* might not use n Steiner vertices and therefore we associate binary variables p_i to denote the presence or absence of a Steiner vertex i.

We use $2\binom{2n}{2}$ edge selection binary variables $e_{i,j}$ to denote the presence or absence of directed edge (i,j). We want $\sum_{i,j} e_{i,j}$ to be the number of edges in T^*.

To define the distance between a pair of vertices, we need some additional auxiliary variables. We use $\binom{n}{2}m$ variables $c_{i,j,k} = |h_{i,k} - h_{j,k}|$ to denote whether vertices i,j differ at site k. The absolute value for this constraint can be expressed as a linear equation. Now distance $r_{i,j} = \sum_{k=1}^{m} w_k c_{i,j,k}$.

To define the objective, however, we need $\sum_{i,j} e_{i,j} r_{i,j}$ which is quadratic. We can instead achieve the same result by defining the following linear constraint $s_{i,j} \geq r_{i,j} - m w_{max} + m w_{max} e_{i,j}$, where $w_{max} = \max_i w_i$. Now the objective function is simply to minimize $\sum_{i,j} s_{i,j}$.

We, however, need additional constraints to ensure that the output is a tree and it connects all the terminal vertices. First, we have $O(n^2)$ constraints: for all i,j, $\sum_k c_{i,j,k} \geq 1$. We also have $2n$ integer variables d_i representing the *depth* of a vertex i from the root (arbitrarily the first row of H). We ensure that vertex a can connect to another vertex of the phylogeny only if it is of depth one smaller with the constraints that for all i,j, $y_{i,j} - d_i + d_j \geq -1, y_{i,j} + d_i - d_j \geq 1, (2n+1)e_{i,j} + y_{i,j} \leq 2n+1$. Also, $\sum_j e_{i,j} + p_i = 1$ for all i to ensure that there exists only one parent for every vertex (except the root). Finally the constraint $\sum_{i,j} e_{i,j} = n-1$ ensures, that the set of edges selected forms a tree. We now have the following theorem. Putting these components together results in the following ILP:

$$\min \quad \sum_{i,j} s_{i,j} \quad \text{subject to} \tag{9}$$

$$c_{i,j,k} \geq h_{i,k} - h_{j,k} \quad \text{for all } 1 \leq i,j \leq n,\ i \neq j,\ 1 \leq k \leq m \tag{10}$$

$$c_{i,j,k} \geq h_{j,k} - h_{i,k} \quad \text{for all } 1 \leq i,j \leq n,\ i \neq j,\ 1 \leq k \leq m \tag{11}$$

$$r_{i,j} = \sum_{k=1}^{m} w_k c_{i,j,k} \quad \text{for all } 1 \leq i,j \leq n,\ i \neq j \tag{12}$$

$$s_{i,j} \geq r_{i,j} - mw_{max} + mw_{max}e_{i,j} \quad \text{for all } 1 \leq i,j \leq n,\ i \neq j \tag{13}$$

$$\sum_k c_{i,j,k} \geq 1 \quad \text{for all } 1 \leq i,j \leq n,\ i \neq j \tag{14}$$

$$y_{i,j} - d_i + d_j \geq -1 \quad \text{for all } 1 \leq i,j \leq n,\ i \neq j \tag{15}$$

$$y_{i,j} + d_i - d_j \geq 1 \quad \text{for all } 1 \leq i,j \leq n,\ i \neq j \tag{16}$$

$$(2n+1)e_{i,j} + y_{i,j} \leq 2n+1 \quad \text{for all } 1 \leq i,j \leq n,\ i \neq j \tag{17}$$

$$\sum_j e_{i,j} + p_i = 1 \quad \text{for all } 1 \leq i \leq n \tag{18}$$

$$\sum_{i,j} e_{i,j} = n-1 \tag{19}$$

We further constrain all variables to be non-negative and fix the depth of the root node to be zero.

Theorem 5.1: The above linear program uses polynomial number of variables and constraints and the solution of the ILP is a most parsimonious phylogenetic tree.

Proof: We have nm variables coding unknown allele values for the n Steiner nodes that might be present in the phylogeny, $(2n)(2n-1)$ edge selection variables identifying the edges in the phylogeny, $\frac{1}{2}(n)(n-1)m$ auxiliary variables used to measure Hamming distances, $\frac{1}{2}(n)(n-1)$ variables specifying the Hamming distances of selected edges only, $2n$ depth variables, $2n-1$ parent variables, and $(2n)(2n-1)$ auxiliary y_{ij} variables used in setting the depth constraints. The total variable set therefore has size $O(n^2m)$.

We have $2n(n-1)m$ constraints for computing absolute values (lines 10-11), $n(n-1)$ for determining edge costs between nodes (line 12), $n(n-1)$ for determining weights of selected edges (line 13), $n(n-1)$ enforcing that all nodes are connected to the phylogeny (line 14), $3n(n-1)$ for enforcing node depth constraints (lines 15-17), n for ensuring each node has a parent (line 18), and one forcing the phylogeny to have $n-1$ edges and thus to be a tree (line 19). The total number of constraints is therefore also $O(n^2m)$.

The correctness of the program is established in the text above explaining its derivation.

■

VI. EMPIRICAL VALIDATION

Experience with both ILPs showed the exponential-sized one to be generally the more efficient in practice. This seems to be the case in practice as the LP relaxation of the exponential sized ILP produces integer solutions or just requires a few rounding iterations. In contrast the polynomial sized ILP contains integer variables that remain fractional and therefore require many relaxations to be solved. We therefore used that variant for our empirical studies. We applied the ILP to several sets of variation data chosen to span a range of data characteristics and computational difficulties. We used only non-recombining data (Y chromosome, mitochondrial, and bacterial DNA) because imperfection in non-recombining data is most likely to be explained by recurrent mutations. We used two Y chromosome data sets: a set of all human Y chromosome data from the HapMap [2] and a set of predominantly chimpanzee primate data [30]. Several different samples of mitochondrial DNA (mtDNA) were also included [31], [32], [33], [34]. Finally, we analyzed a single bacterial sample [35].

The pre-processing and ILP formulation was performed in C++ and solved using the Concert callable library of CPLEX 10.0. In each case, the ILP was able to find an optimal tree on the data after preprocessing. We also used the `pars` program of `phylip`, which attempts to heuristically find the most parsimonious phylogeny. `pars` was run with default parameters. Empirical tests were conducted on a 2.4 GHz Pentium 4 computer with 1G RAM, running Linux. We attempted to use the `penny` program of `phylip`, which finds provably optimal solutions by branch-and-bound, but it terminated in under 20 minutes only for the smallest mitochondrial data set and was aborted by us after 20 minutes for all other tests.

We first used the mitochondrial data as a basic validation of the correctness of the methods and the reasonableness of the maximum parsimony criterion on these data. The HVS-I and HVS-II segments of the mitochondrial D-loop region have exceptionally high mutation rates [31], providing a good test case of the ability of our algorithm to distinguish regions we would expect to have perfect or near-perfect phylogenies from those expected to have highly imperfect phylogenies. Figure 2 shows a scan of 201-site long windows across the complete 16569-site mtDNA genome. Since the mtDNA is circular, the windows wrap around over the ends in

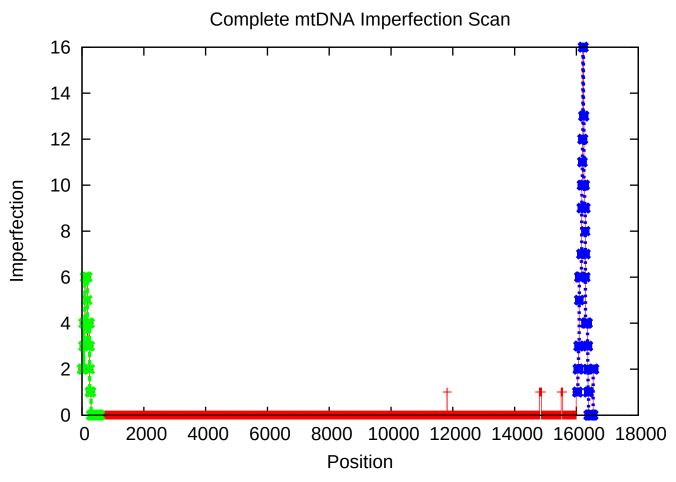
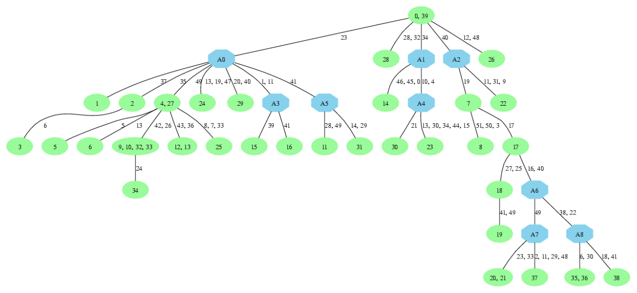


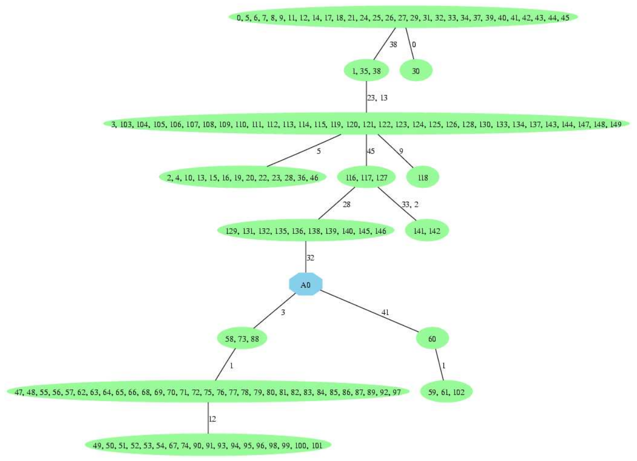
Fig. 2. Imperfection of the most parsimonious phylogeny for overlapping windows across the complete mitochondrial genome. The x-axis shows the sites in their order along the genomic axis. The y-axis shows the imperfection for the window centered on the corresponding site. The hyper variable D-loop region ($1 \ldots 577$ and $16028 \ldots 16569$) shows significantly larger imperfection.

the genome order. The y-axis corresponds to *imperfection*, which is the number of recurrent mutations in the most parsimonious phylogeny. The figure does indeed show substantially larger phylogenies within the high mutation rate D-loop region ($1 \ldots 577$ and $16028 \ldots 16569$) than in the low mutation rate coding regions, confirming the relevance of a parsimony metric for such data sets.

We then ran the ILP on a collection of data sets to assess its efficiency. Figure 3 provides two examples of most parsimonious phylogenies for data sets at opposite extremes of difficulty: an mtDNA sample [31] with imperfection 21 (Fig. 3(a)) and the human Y chromosome sample, with imperfection 1 (Fig. 3(b)). Table I presents the empirical run-time data for all of the datasets. The columns 'input before' and 'input after' correspond to the size of the original input and that after preprocessing (rows $\times$ columns). The table also provides the ILP size for each data set (variables, constraints). Run times vary over several orders of magnitude and appear largely insensitive to the actual sizes of the data sets. Rather, the major determinant of run time appears to be a dataset's imperfection, i.e., the difference between the optimal length and the number of variant sites. It has recently been shown that the phylogeny problem under various assumptions is fixed parameter tractable in imperfection [14], [15], [16], [17] possibly suggesting why it is a critical factor in run time determination. The `pars` program of `phylip`, despite providing no guarantees of optimality, does indeed find optimal phylogenies in all of the above instances. It



(a)



(b)

Fig. 3. Examples of trees of varying levels of difficulty. Nodes labeled with numbers correspond to the numbered input haplotypes, while those labeled A# correspond to inferred Steiner nodes. Edges are labeled with the site variations to which they correspond. (a) Human mitochondrial data from Wirth et al. [31] (b) Human Y chromosome from HapMap [2]

is, however, slower than the ILP in most of these cases.

VII. ONLINE TOOL

In order to provide more general access to our methods, we have implemented a web server based on our worst-case exponential-sized ILP. The server provides a front end to a an implementation of the ILP in C++ using the CPLEX 10 libraries. We call the server SCan for IMperfect Phylogenies (SCIMP). It can be accessed at `http://www.cs.cmu.edu/˜imperfect/index.html`. There are two ways to use the webserver, as explained below.

	input			time(secs)		
Data Set	before	after	length	our ILP	`pars`	ILP size
human chrY [2]	150×49	14×15	16	0.02	2.55	(510, 697)
bacterial [35]	17×1510	12×89	96	0.08	0.06	(780, 995)
chimp mtDNA [30]	24×1041	19×61	63	0.08	2.63	(1480, 1982)
chimp chrY [30]	15×98	15×98	99	0.02	0.03	(736, 12012)
human mtDNA [31]	40×52	32×52	73	13.39	11.24	(55308, 62467)
human mtDNA [34]	395×830	34×39	53	53.4	712.95	(63070, 70673)
human mtDNA [32]	13×390	13×42	48	0.02	0.41	(1288, 1604)
human mtDNA [33]	44×405	27×39	43	0.09	0.59	(5264, 6636)

TABLE I

EMPIRICAL RUN-TIME RESULTS ON A SELECTION OF NON-RECOMBINING DATASETS.

Firstly, the users can input a haplotype variation data-set. These are simply a set of n haplotype sequences typed over m SNPs. As stated in the previous sections, this has to be phased data. Therefore essentially the input is an $n \times m$ $\{0, 1\}$ matrix.

Alternatively, the users can select any region of the genome and provide the number of contiguous SNPs to examine in that region. The user also needs to specify the population group to use. The webserver currently has support for the Central European population (CEPH) and Yoruba African population (YRI). The entire HapMap (phase II) phasing data is present in the webserver's backend database and this makes it easy for users to quickly examine and construct phylogenies for any region of interest. Since the HapMap data for these two populations were sequenced in trios, the number of phasing errors should be very small.

The webserver can be used in two different modes. As has been described until now, the user can just request it to construct the most parsimonious phylogenetic tree and return the topology, the parsimony score (number of mutations) and the imperfection (number of recurrent mutations).

The webserver can also perform an imperfection scan. The user specifies the location and size and additionally for this mode provides a window length w in the number of SNPs. The webserver then slides this window across the genome and for each overlapping set of w consecutive SNPs constructs a maximum parsimony phylogeny. The server returns to the user a plot of the imperfection (number of recurrent mutations) of each of these windows across the entire region examined. It can further provide the maximum parsimony tree found within each window.

In addition to providing a general interface to the phylogeny inference code, the server also houses a precomputed database of maximum parsimony phylogenies that it constructed offline for more than 3.7 million instances using the HapMap SNPs. Therefore, when users request to see phylogenies that are present in this precomputed data-set (or while performing scans), the results are returned as soon as they are fetched with no online solution required. This precomputed databases currently has phylogenies for every continguous region of up to 10 SNPs in all of HapMap.

Figure 3 provides examples of the server output.

VIII. CONCLUSION

We have developed ILP formulations for optimally solving for the most parsimonious phylogeny using binary genome variation data. These methods fill an important practical need for fast methods for generating provably optimal trees from large SNP variation datasets. This need is not served well by the heuristic methods that are currently the standard for tree-building, which generally work well in practice but cannot provide guarantees of optimality. More recent theoretical methods that find provably optimal trees within defined run-time bounds are inefficient in practice without a fast sub-routine to solve the general problem on smaller instances. The ILP approach allows extremely fast solutions of the easy cases while still yielding run-times competitive with a widely used fast heuristic for hard instances. Such methods are likely to be increasingly important as data sets accumulate on larger population groups and larger numbers of variant sites.

ACKNOWLEDGMENT

We thank Daniel Gusfield for helpful discussions and for motivating our use of LP for problems in phylogenetics. This work was supported by U.S. National Science Foundation grants IIS-0612099, CCR-0105548, and CCR-0122581 (The ALADDIN project). A preliminary version of this paper appeared at the 2007 International Symposium on Bioinformatics Research and Applications [36].

References

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[2] T. International HapMap Consortium, “The International HapMap Project. www.hapmap.org,” *Nature*, vol. 426, pp. 789–796, 2005.

[3] E. M. Smigielski, K. Sirotkin, M. Ward, and S. T. Sherry, “dbSNP: a database of single nucleotide polymorphisms.” *Nucleic Acids Research*, vol. 28, no. 1, pp. 352–355, 2000.

[4] T. C. Sequencing and A. Consortium, “Initial sequence of the chimpanzee genome and comparison with the human genome,” *Nature*, vol. 437, no. 7055, pp. 69–87, 2005. [Online]. Available: http://dx.doi.org/10.1038/nature04072

[5] L.-T. K. et al., “Large-scale discovery and genotyping of single-nucleotide polymorphisms in the mouse,” *Nature Genetics*, pp. 381–386, 2000.

[6] L.-T. et al., “Genome sequence, comparative analysis and haplotype structure of the domestic dog,” *Nature*, vol. 438, no. 7069, pp. 803–819, 2005. [Online]. Available: http://dx.doi.org/10.1038/nature04338

[7] T. E. P. Consortium, “The ENCODE (ENCyclopedia Of DNA Elements) Project,” *Science*, vol. 306, no. 5696, pp. 636–640, 2004.

[8] R. Agarwala and D. Fernandez-Baca, “A polynomial-time algorithm for the perfect phylogeny problem when the number of character states is fixed,” *SIAM Journal on Computing*, vol. 23, pp. 1216–1224, 1994.

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

[10] S. Kannan and T. Warnow, “A fast algorithm for the computation and enumeration of perfect phylogenies,” *SIAM Journal on Computing*, vol. 26, pp. 1749–1763, 1997.

[11] H. J. Bandelt, P. Forster, B. C. Sykes, and M. B. Richards, “Mitochondrial portraits of human populations using median networks,” *Genetics*, vol. 141, pp. 743–753, 1989.

[12] J. Felsenstein, “PHYLIP (Phylogeny Inference Package) version 3.6. distributed by the author, Department of Genome Sciences, University of Washington, Seattle,” 2005.

[13] N. Saitou and M. Nei, “The neighbor-joining method: a new method for reconstructing phylogenetic trees.” *Molecular Biology and Evolution*, vol. 4, no. 4, pp. 406–425, 1987.

[14] G. E. Blelloch, K. Dhamdhere, E. Halperin, R. Ravi, R. Schwartz, and S. Sridhar, “Fixed parameter tractability of binary near-perfect phylogenetic tree reconstruction,” *International Colloquium on Automata, Languages and Programming*, 2006.

[15] D. Fernandez-Baca and J. Lagergren, “A polynomial-time algorithm for near-perfect phylogeny,” *SIAM Journal on Computing*, vol. 32, pp. 1115–1127, 2003.

[16] S. Sridhar, G. E. Blelloch, R. Ravi, and R. Schwartz, “Optimal imperfect phylogeny reconstruction and haplotyping,” *Computational Systems Bioinformatics*, 2006.

[17] S. Sridhar, K. Dhamdhere, G. E. Blelloch, E. Halperin, R. Ravi, and R. Schwartz, “Simple reconstruction of binary near-perfect phylogenetic trees,” *International Workshop on Bioinformatics Research and Applications*, 2006.

[18] D. Gusfield, “Haplotyping by pure parsimony,” *Combinatorial Pattern Matching*, 2003.

[19] X. Cheng and D. D. (Eds.), *Steiner Trees in Industry*. Springer, 2002.

[20] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner Tree Problem*. Annals of Discrete Mathematics, 1992, vol. 53.

[21] L. R. Foulds and R. L. Graham, “The Steiner problem in phylogeny is NP-complete,” *Advances in Applied Mathematics*, vol. 3, 1982.

[22] P. Buneman, “The recovery of trees from measures of dissimilarity,” *Mathematics in the Archeological and Historical Sciences, F. Hodson et al., Eds.*, pp. 387–395, 1971.

[23] J. Barthélemy, “From copair hypergraphs to median graphs with latent vertices,” *Discrete Math*, vol. 76, pp. 9–28, 1989.

[24] C. Semple and M. Steel, *Phylogenetics*. Oxford University Press, 2003.

[25] V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph, “Haplotyping as perfect phylogeny: A direct approach,” *Journal of Computational Biology*, vol. 10, pp. 323–340, 2003.

[26] D. Gusfield and V. Bansal, “A fundamental decomposition theory for phylogenetic networks and incompatible characters,” *Research in Computational Molecular Biology*, 2005.

[27] J. Beasley, “An algorithm for the Steiner problem in graphs,” *Networks*, vol. 14, pp. 147–159, 1984.

[28] N. Maculan, “The Steiner problem in graphs,” *Annals of Discrete Mathematics*, vol. 31, pp. 185–212, 1987.

[29] R. Wong, “A dual ascent approach for Steiner tree problems on a directed graph,” *Mathematical Programming*, vol. 28, pp. 271–287, 1984.

[30] A. C. Stone, R. C. Griffiths, S. L. Zegura, and M. F. Hammer, “High levels of Y-chromosome nucleotide diversity in the genus Pan,” *Proceedings of the National Academy of Sciences USA*, pp. 43–48, 2002.

[31] T. Wirth, X. Wang, B. Linz, R. P. Novick, J. K. Lum, M. Blaser, G. Morelli, D. Falush, and M. Achtman, “Distinguishing human ethnic groups by means of sequences from Helicobacter pylori: Lessons from Ladakh,” *Proceedings of the National Academy of Sciences USA*, vol. 101, no. 14, pp. 4746–4751, 2004.

[32] S. Sharma, A. Saha, E. Rai, A. Bhat, and R. Bamezai, “Human mtDNA hypervariable regions, HVR I and II, hint at deep common maternal founder and subsequent maternal gene flow in Indian population groups,” *American Journal of Human Genetics*, vol. 50, pp. 497–506, 2005.

[33] C. J. Lewis, R. Tito, B. Lizarraga, and A. Stone, “Land, language, and loci: mtDNA in Native Americans and the genetic history of Peru,” *American Journal of Physical Anthropology*, vol. 127, pp. 351–360, 2005.

[34] A. Helgason, G. Palsson, H. S. Pedersen, E. Angulalik, E. D. Gunnarsdottir, B. Yngvadottir, and K. Stefansson, “mtDNA variation in Inuit populations of Greenland and Canada: migration history and population structure,” *American Journal of Physical Anthropology*, vol. 130, pp. 123–134, 2006.

[35] M. Merimaa, M. Liivak, E. Heinaru, J. Truu, and A. Heinaru, “Functional co-adaption of phenol hydroxylase and catechol 2,3-dioxygenase genes in bacteria possessing different phenol and p-cresol degradation pathways,” *Eighth Symposium on Bacterial Genetics and Ecology*, vol. 31, pp. 185–212, 2005.

[36] S. Sridhar, F. Lam, G. Blelloch, R. Ravi, and R. Schwartz, “Efficiently finding the most parsimonious phylogenetic tree via linear programming,” in *Proceedings of the 2007 International Symposium on Bioinformatics Research and Applications*, 2007.

Srinath Sridhar Srinath Sridhar received his BS in Computer Science from University of Texas at Austin in 2003. Since then, he has been a PhD student at the Department of Computer Science, Carnegie Mellon University. His primary area of research is computational biology.

Fumei Lam Fumei Lam received a BA from U.C. Berkeley in mathematics in 2000 and a PhD from the Massachusetts Institute of Technology in applied math in 2005. She spent one year as a postdoctoral researcher at Carnegie Mellon University and is currently a postdoc in the Center for Computational Molecular Biology at Brown University.

Guy Blelloch Guy Blelloch received a BA degree in Physics and a BS degree in Engineering in 1983 from Swarthmore College, and MS and PhD degrees in Computer Science from the Massachusetts Institute of Technology in 1986 and 1988, respectively. He is currently a Professor of Computer Science at Carnegie Mellon University and co-director of the ALADDIN center for the study of algorithms. His research interests are in programming languages and applied algorithms.

R. Ravi R. Ravi received his B. Tech. in Computer Science and Engineering from the Indian Institute of Technology, Madras in 1989 and a PhD in Computer Science from Brown University in 1993. After post-doctoral fellowships at UC Davis and DIMACS, Princeton University, he joined the Operations Research faculty at the Tepper School of Business at Carnegie Mellon University in 1995, where he is currently Carnegie Bosch Professor of Operations Research and Computer Science.

Russell Schwartz Russell Schwartz received his BS, MEng, and PhD degrees from the Department of Electrical Engineering and Computer Science at the Massachusetts of Technology, the last in 2000. He later worked in the Informatics Research group at Celera Genomics. He joined the faculty of Carnegie Mellon University in 2002, where he is currently an Associate Professor of Biological Sciences.