A Structural EM Algorithm for Phylogenetic Inference

Nir Friedman[*] **Matan Ninio**
School of Computer Science & Engineering
Hebrew University
Jerusalem, 91904, Israel
{nir,ninio}@cs.huji.ac.il

Itsik Pe'er
School of Computer Science
Tel-Aviv University
Tel-Aviv, 69978, Israel
izik@math.tau.ac.il

Tal Pupko
Institute of Statistical Mathematics
Tokyo, Japan
tal@ismspc5.ism.ac.jp

ABSTRACT

A central task in the study of evolution is the reconstruction of a phylogenetic tree from sequences of current-day taxa. A well supported approach to tree reconstruction performs maximum likelihood (ML) analysis. Unfortunately, searching for the maximum likelihood phylogenetic tree is computationally expensive. In this paper, we describe a new algorithm that uses Structural-EM for learning maximum likelihood trees. This algorithm is similar to the standard EM method for estimating branch lengths, except that during iterations of this algorithms the topology is improved as well as the branch length. The algorithm performs iterations of two steps. In the *E-Step*, we use the current tree topology and branch lengths to compute expected sufficient statistics, which summarize the data. In the *M-Step*, we search for a topology that maximizes the likelihood with respect to these expected sufficient statistics. As we show, searching for better topologies inside the M-step can be done efficiently, as opposed to standard search over topologies. We prove that each iteration of this procedure increases the likelihood of the topology, and thus the procedure must converge. We evaluate our new algorithm on both synthetic and real sequence data, and show that it is both dramatically faster and finds more plausible trees than standard search for maximum likelihood phylogenies.

1. INTRODUCTION

The understanding that many biological sequences share a single common origin is fundamental to biology. Such a set of contemporary sequences diverged from their ancestral sequence in a tree-like fashion. Inferring this *phylogenetic tree* has been a major research problem since the dawn of computational molecular biology, more than 30 years ago [3, 28]. The input data available is usually a set of sequences, one per species. The goal is to find the tree that describes the true evolutionary history of these sequences.

[*]Contact author.

Many attempts have been made to formalize the distinction of the true tree into a mathematically tractable criterion, giving rise to a variety of reconstruction algorithms. We mention a few of these (see [11] for more details). One such criterion is accordance of the tree with observed distances between pairs of sequences. The prominent method which uses this criterion is *Neighbor Joining* (NJ) [27], in which partial trees are iteratively combined to form a larger tree, in a bottom-up manner. A second important criterion is *Maximum Parsimony*. It states that substitutions are rare, and thus calls for finding the tree topology which implies as few substitutions as possible. Although it is NP-hard to find the most parsimonious tree [6, 14], effective heuristics [15, 16, 17, 29] exhibit reasonable performance in affordable time.

The criterion which we study is probabilistic. It builds on the view of evolution as a stochastic process, in which characters change over time according to some predetermined probabilities. Along each tree branch, such probabilities depend on the duration of the period that this branch represents, i.e., the branch *length*. These probabilities were estimated in many studies [1, 7, 18, 19, 20, 32]. This description of evolution in stochastic terms allows computing the likelihood of a specific phylogeny, and gives rise to Maximum Likelihood (ML) methods [9]. Indeed, finding the ML phylogenetic tree proves superior to other methods in terms of accuracy [10]. However, speed is the major obstacle, as we now explain.

ML reconstruction consists of two tasks. The first task involves branch length estimation: Given a topology, find branch lengths to maximize the likelihood. This task is accomplished by iterative methods such as *Expectation Maximization* (EM) [8, 9], or using Newton-Raphson optimization [24]. Each iteration of these methods requires computations that take on the order of the number of taxa times the number of training positions. In addition, these methods are only guaranteed to find local maxima, although in practice they often recover the global maximum [4].

The second, more challenging, ML reconstruction task is to find a tree *topology* that maximizes the likelihood. Naive, exhaustive search of the tree space is infeasible, and also the effectivity of exploring this space by heuristic paradigms, like simulated annealing [22] or genetic algorithms [23], is hampered by the costly procedure of re-estimating branch lengths afresh for different trees. Indeed, this task is usually tackled by iterative procedures that greedily construct the desired tree. The leading ML application for protein sequences is the MOLPHY software package [2]. One of its versions has been incorporated into the PHYLIP library as the ProtML application. MOLPHY uses the *Star Decomposition* top-down heuristic, in which an initial, star-like tree with a single internal node is iteratively refined [2]. In this method, the scoring of each intermediate topology considered requires finding its best branch lengths. The main cost of the algorithm is due to these repeated invocations of branch length optimization.

Our approach builds on Structural EM, the extension of the EM algorithm for learning combinatorial constructs [12]. As all EM-type algorithms, we use an expected value of the likelihood, computed using sufficient statistics, which are collected from the data. The basic EM-theorem states that improving this expected likelihood implies an increase in the likelihood itself [8]. In contrast to standard EM or Structural EM algorithms, we do not just iterate this improvement procedure over and over. After each iteration, which improves the expected likelihood, we employ a modification step. This novel step, which is necessary due to the nature of our problem, is guaranteed not to change the likelihood.

The paper is organized as follows: In Section 2 we review the framework of maximum likelihood phylogenetic reconstruction. In Section 3 we present theoretic basis to our algorithm, which we present in Section 4. Section 5 reports application to simulated and real data. We conclude, in Section 6, with a discussion of related and future work.

2. MAXIMUM LIKELIHOOD PHYLOGENETIC INFERENCE

We view evolution as a process involving the change (*substitution*) of characters into other characters. These characters are assumed to be elements of a fixed, finite, alphabet Σ, which is usually the set of 4 DNA nucleotides, 20 amino-acids, or 64 codon triplets.

A *model of evolution* is the distribution of substitutions along time. Such a model defines the probability $p_{a\to b}(t)$ of the character a transforming into the character b in the duration t. Such models have been devised, for instance, by [19, 20, 32] for nucleotides, [1, 7, 18] for amino acids, and [13] for codons. Different models imply different biological assumptions. However, there are some properties shared by all standard models [7]:

1. Lack of Memory -
$$p_{a\to b}(t+t') = \sum_{c\in\Sigma} p_{a\to c}(t)p_{c\to b}(t') \qquad (1)$$
This assumption implies that the model can be fully described by a single, $|\Sigma|\times|\Sigma|$ matrix. The (a,b) entry in this matrix is $p_{a\to b}(1)$. To obtain $p_{a\to b}(t)$ for $t\neq 1$, we need to exponentiate this matrix to the power of t, and take the (a,b) entry of the resulting matrix.

2. Reversibility - we assume that there is a *prior distribution* over characters, $\{p_a\}$ such that
$$p_a p_{a\to b}(t) = p_b p_{b\to a}(t)$$
for all $a,b\in\Sigma$ and $t\geq 0$. This assumption essentially states that the events "a evolved into b" and "b evolved into a" are equiprobable. Note, however, that the transition matrix entries are conditional probabilities, hence, this matrix needs not be symmetric.

So far, we have described the evolution of a single character over time. However, phylogeny deals with a plurality of species. Consider a set of N current day species. They are assumed to be the descendants of a single ancestral species, their lineages having diverged during history. The pattern, or *topology* of this divergence process is usually unknown, and its inference is the main goal of this study. A hypothetical topology is represented by an undirected tree T with N leaves, corresponding to the contemporary species. Internal nodes correspond to events of divergence. *Branches* represent periods in the history of some past-time species between those events. We label leaves by the indices $1,\ldots,N$, and internal nodes by $N+1,\ldots,N_T$. Formally, the topology T is described by the set of its branches. We use the notation $(i,j)\in T$ to indicate T having a branch between nodes i and j. We reserve the term *branch* for pairs of nodes in T, while using the term *edge* for arbitrary pairs of nodes, not necessarily in T.

In this paper, we pay special attention to *bifurcating* topologies, in which each internal node is adjacent to exactly three other nodes. These are the undirected analogues of binary trees. Thus, in a bifurcating topology there are $N-2$ internal nodes, indexed $N+1,\ldots,2N-2$.

We would like to introduce the model of evolution to our phylogenetic hypothesis, T. To this end, we also consider the duration of time that separates adjacent nodes. A *parameterization* of a topology T is a vector $\mathbf{t}$, comprising of a non-negative duration or *branch length* $t_{i,j}$ for each branch $(i,j)\in T$. The pair $(T,\mathbf{t})$ constitutes a *phylogenetic tree.*

Based on the model of evolution, we can assign probabilistic semantics to the phylogenetic tree. Formally, we associate with each node i a random variable X_i that describes the character at this node. The distribution of such a variable X_i, i.e., the set $\{P(X_i=a)\}_{a\in\Sigma}$ of probabilities, is denoted by $P(X_i)$, for short. The joint distribution $P(X_{[1\ldots N_T]})$ is defined as follows. Pick some node r as a root. For each node $i\neq r$ define its *parent*, $\pi(i)$ to be i's neighbor which is closer to the root. (Since T is a tree, this neighbor is uniquely defined.) Then the distribution $P(X_i)$ depends only on $P(X_{\pi(i)})$ and $t_{i,\pi(i)}$. Hence, the joint distribution $P(X_{[1\ldots N_T]})$ is:

$$P(X_{[1\ldots N_T]}\mid T,\mathbf{t}) = P(X_r)\prod_{i\neq r} P(X_i\mid X_{\pi(i)},t_{i,\pi(i)}) \qquad (2)$$

where $P(X_r=a)=p_a$, and $P(X_i=b\mid X_{\pi(i)}=a,t_{i,\pi(i)}) = p_{a\to b}(t_{i,\pi(i)})$. It is fairly straightforward to show, using reversibility of substitution probabilities, that the joint distribution $P(X_{[1\ldots N_T]})$ is invariant to the choice of r:

$$P(X_{[1\ldots N_T]}\mid T,\mathbf{t}) = \prod_i P(X_i)\prod_{(i,j)\in T}\frac{P(X_i\mid X_j,t_{i,j})}{P(X_i)} \qquad (3)$$

Usually, we observe only the characters in the leaf nodes $1,\ldots,N$. The likelihood of a single observation $x_{[1\ldots N]}$, given the phylogenetic hypothesis, is therefore the *marginal* distribution over these variables:

$$P(x_{[1\ldots N]} \mid T, \mathbf{t}) = \sum_{x_{N+1}} \ldots \sum_{x_{N_T}} P(x_{[1\ldots N_T]} \mid T, \mathbf{t}). \quad (4)$$

From a computational standpoint, we do not want to compute the marginal probability by summing over the $|\Sigma|^{N_T - N}$ possible assignments to $X_{[N+1\ldots N_T]}$. Instead, we can exploit the structure of the distribution, as specified by Eq. (2), for efficient computation. This is done by dynamic programming over the set of nodes [33]. In addition, this dynamic program can also answer marginal queries about variables and branches. These compute the induced, aposteriori probabilities $P(X_i = a \mid x_{[1\ldots N]}, T, \mathbf{t})$ and $P(X_i = a, X_j = b \mid x_{[1\ldots N]}, T, \mathbf{t})$.

For the sake of completion, we now describe the computation of these probabilities. See [33] for more details. Consider a single observed assignment $\{X_1 = x_1, \ldots, X_N = x_N\}$ to the leaves of a bifurcating tree $(T, \mathbf{t})$. Each branch $e = (i,j) \in T$, partitions the tree into two subtrees. Define the set $S(i,j)$ to include the leaves of the subtree which includes i. We inquire for the probability of the observed characters in $S(i,j)$ conditioned on possible assignments to X_i or X_j. More formally, for each character $a \in \Sigma$, we define *upward-messages* as follows:

$$\begin{aligned} U_{i\to j}(a) &\equiv P(\{X_k = x_k\}_{k\in S(i,j)} \mid X_i = a, T, \mathbf{t}) \\ u_{i\to j}(a) &\equiv P(\{X_k = x_k\}_{k\in S(i,j)} \mid X_j = a, T, \mathbf{t}) \end{aligned}$$

We can recursively compute upward-messages, according to the following formulae:

$$\begin{aligned} U_{i\to j}(a) &= \begin{cases} 1\{x_i = a\} & i \text{ is a leaf} \\ \prod_{k\neq j:(k,i)\in T} u_{k\to i}(a) & i \text{ is an internal node} \end{cases} \\ u_{i\to j}(a) &= \sum_b p_{a\to b}(t_{i,j}) U_{i\to j}(b) \end{aligned}$$

The computation of these messages across the whole tree can be completed in linear time.

The upward messages allow us to compute the marginal probability from the messages that reached an arbitrary branch (i,j):

$$P(x_{[1\ldots N]} \mid T, \mathbf{t}) = \sum_a p_a U_{i\to j}(a) u_{j\to i}(a).$$

Another task of interest is computing conditional probabilities of the form $P(X_i \mid x_{[1\ldots N]}, T, \mathbf{t})$ and $P(X_i, X_j \mid x_{[1\ldots N]}, T, \mathbf{t})$, for a branch $(i,j) \in T$. Fortunately, the upward messages allow computing these as well:

$$P(X_i = a \mid x_{[1\ldots N]}, T, \mathbf{t}) = \frac{P(a) U_{i\to j}(a) u_{j\to i}(a)}{P(x_{[1\ldots N]} \mid T, \mathbf{t})}$$

and

$$P(X_i = a, X_j = b \mid x_1, \ldots, x_N, T, \mathbf{t}) = \frac{P(a) U_{i\to j}(a) p_{a\to b}(t_{i,j}) U_{j\to i}(b)}{P(x_{[1\ldots N]} \mid T, \mathbf{t})}$$

Our main task is recovering phylogeny from observed data. An input data set D consists of M observations, drawn from the marginal distribution. That is, we have a sequence $x_i[1], \ldots, x_i[M]$ of characters for each leaf X_i, and these sequences are assumed to be *aligned* in the following sense: For each m, the characters in the m-th position across all species evolved from a single ancestral character, and thus comprise a single observation drawn from the marginal distribution. We further assume the model of evolution to be known.

We assume that different positions evolve independently. This allows computing the likelihood of the whole data set D given the phylogeny $(T, \mathbf{t})$, as follows:

$$L(T, \mathbf{t}) = P(D \mid T, \mathbf{t}) = \prod_{m=1}^{M} P(x_{[1\ldots N]}[m] \mid T, \mathbf{t}) \quad (5)$$

The Maximum Likelihood reconstruction task is to find a topology T and associated parameters $\mathbf{t}$ that maximize this likelihood. The phylogenetic tree $(T, \mathbf{t})$ is, in some sense, the most plausible candidate to having generated the data.

3. EXPECTED LIKELIHOOD

The likelihood of a tree is a complicated function to compute, let alone to maximize. However, as we detail below, it can be conveniently presented in terms of *frequency counts* registering the co-occurrences of characters along branches, across both the observed and ancestral sequences. These counts are therefore sufficient statistics. Unfortunately, we can not know the ancestral sequences, nor their counts. However, given some phylogenetic assumption $(T^0, \mathbf{t}^0)$, we can induce probabilities on these sequences. Assuming the phylogeny $(T^0, \mathbf{t}^0)$ we can thus compute the *expected* counts, which allow computing the expected log likelihood.

A key observation we make is that this expected likelihood can be decomposed into a sum of local terms, one per branch, each of which being a function of only some of the expected counts. Moreover, each term can be optimized independently of the other ones. We can thus perform an iteration of improving the current tree $(T^0, \mathbf{t}^0)$. In this section, we detail the different steps of these ideas.

3.1 Complete Data

We first deal with a somewhat unreasonable situation, where we get to observe the ancestral sequences. The study of this case constitutes the basis of later analysis.

In the complete-data scenario, our input is not only the set $D = \{x_i[m] : 1 \le i \le N, 1 \le m \le M\}$ of contemporary sequences, but also the set $H = \{x_i[m] : N+1 \le i \le 2N-2, 1 \le m \le M\}$ of ancestral sequences. When we observe the values of all $2N-2$ nodes for each position, the likelihood function is

$$\begin{aligned} L_{\text{complete}}(T, \mathbf{t}) &= P(D, H \mid T, \mathbf{t}) \\ &= \prod_m P(x_{[1\ldots 2N-2]}[m] \mid T, \mathbf{t}) \end{aligned}$$

Note that since in this case we do not marginalize over unobserved nodes, each $P(x_{[1\ldots 2N-2]}[m] \mid T, \mathbf{t})$ term is a product of conditional probabilities. Thus, by rearranging the order of multiplications as in Eq. (3), we can reformulate the likelihood in a more manageable form.

To do so, we need to count how many times each conditional probability appears in the product of terms that constitutes the likelihood function. A *frequency count* of an event is the number of times it was observed. In this paper, we focus on frequency counts, which register occurrences and co-occurrences of letters, and are collected from the complete data D, H. Formally, for an event Y, denote by $1\{Y\}$ its corresponding indicator variable. Let $\mathcal{S}_{i,j}(a,b) = \sum_m 1\{X_i[m] = b, X_j[m] = a\}$, and $\mathcal{S}_i(a) = \sum_m 1\{X_i[m] = a\}$.

PROPOSITION 3.1. *The likelihood can be rewritten as:*

$$\log L_{complete}(T, \mathbf{t}) = \sum_{(i,j)\in T} L_{local}(\mathcal{S}_{i,j}, t_{i,j}) + \sum_i \sum_a \mathcal{S}_i(a) \log p_a \qquad (6)$$

where

$$L_{local}(\mathcal{S}_{i,j}, t) = \sum_{a,b} \mathcal{S}_{i,j}(a,b)(\log p_{a\to b}(t) - \log p_b)$$

This formulation is motivated by the approach of Chow and Liu [5] for learning tree models. It is important for several reasons. First, only the term which involves the weight function L_{local} depends on the topology and branch lengths. Thus, when maximizing the likelihood we can ignore the righthand term of Eq. 6. Second, the log-likelihood is a *linear* function of the counts. Finally, the term $L_{\text{local}}(\mathcal{S}_{i,j}, t_{i,j})$ can be optimized independently of other such terms. We define a matrix W, with entries $w_{i,j} = \max_t L_{\text{local}}(\mathcal{S}_{i,j}, t)$. Then the (log-likelihood) score of a tree is simply

$$\max_{\mathbf{t}} \log L_{\text{complete}}(T, \mathbf{t}) = W(T) + \text{constant}$$

where $W(T) = \sum_{(i,j)\in T} w_{i,j}$. This reduces the problem of finding the highest scoring tree, in the case of complete data, to a combinatorial optimization problem in terms of the edge weights. (We address this problem in Section 4.)

3.2 Expected Likelihood Score

We now return to the case where we only observe the values of the N leaves. In this case our objective function is the likelihood according to Eq. (5). We use the notion of complete data to help us devise an approximate likelihood function that will guide us in finding high likelihood trees.

Assume that we are given a data set $D = \{x_i[m] : 1 \le i \le N, 1 \le m \le M\}$. Suppose we have some candidate phylogeny $(T^0, \mathbf{t}^0)$. We aim at computing a function on arbitrary trees $(T, \mathbf{t})$, which is the expected value of the log-probability of the complete data. We can use expected counts based on $(T^0, \mathbf{t}^0)$ to compute this expected log-likelihood of $(T, \mathbf{t})$:

$$\begin{aligned} & Q(T, \mathbf{t} : T^0, \mathbf{t}^0) \\ &= E[\log L_{\text{complete}}(T, \mathbf{t}) \mid D, T^0, \mathbf{t}^0] \\ &= \sum_m \sum_{x_{[N+1\ldots 2N-2]}[m]} q(x_{[1\ldots 2N-2]}[m], T, \mathbf{t}, T^0, \mathbf{t}^0) \end{aligned}$$

where

$$q(x_{[1\ldots 2N-2]}, T, \mathbf{t}, T^0, \mathbf{t}^0) = \log P(x_{[1\ldots 2N-2]} \mid T, \mathbf{t}) P(x_{[N+1\ldots 2N-2]} \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0)$$

This term is a sum over an exponential number of assignments of values to the ancestors in each position. Before we analyze $Q(T, \mathbf{t} : T^0, \mathbf{t}^0)$ further, we examine its theoretical properties.

THEOREM 3.2. *(based on [12]) For any $T, \mathbf{t}$*

$$Q(T, \mathbf{t} : T^0, \mathbf{t}^0) - Q(T^0, \mathbf{t}^0 : T^0, \mathbf{t}^0) \le \log L(T, \mathbf{t}) - \log L(T^0, \mathbf{t}^0)$$

Proof: By definition:

$$Q(T, \mathbf{t} : T^0, \mathbf{t}^0) = \sum_H P(H \mid D, T^0, \mathbf{t}^0) \log P(H, D \mid T, \mathbf{t})$$

Therefore:

$$\begin{aligned} & Q(T, \mathbf{t} : T^0, \mathbf{t}^0) - Q(T^0, \mathbf{t}^0 : T^0, \mathbf{t}^0) \\ &= \sum_H P(H \mid D, T^0, \mathbf{t}^0) \log \frac{P(H, D \mid T, \mathbf{t})}{P(H, D \mid T^0, \mathbf{t}^0)} \\ &\le \log \sum_H P(H \mid D, T^0, \mathbf{t}^0) \frac{P(H, D \mid T, \mathbf{t})}{P(H, D \mid T^0, \mathbf{t}^0)} \\ &= \log \frac{\sum_H P(H, D \mid T, \mathbf{t})}{P(D \mid T^0, \mathbf{t}^0)} = \log L(T, \mathbf{t}) - \log L(T^0, \mathbf{t}^0) \end{aligned}$$

where we apply Jensen's inequality in the second step. ▮

Theorem 3.2 implies that improving the Q score forces an improvement of the objective likelihood. Fortunately, maximizing $Q(T, \mathbf{t} : T^0, \mathbf{t}^0)$ is feasible. Recall that L_{local} is a linear function of the counts $\mathcal{S}_{i,j}$. Thus, by linearity of expectation, we have that

$$Q(T, \mathbf{t} : T^0, \mathbf{t}^0) = \sum_{(i,j)} L_{\text{local}}(E[\mathcal{S}_{i,j} \mid D, T^0, \mathbf{t}^0], t_{i,j}) + \text{const}$$

Thus, if we compute the expected counts $E[\mathcal{S}_{i,j} \mid D, T^0, \mathbf{t}^0]$, we can optimize each $L_{\text{local}}(E[\mathcal{S}_{i,j} \mid D, T^0, \mathbf{t}^0], t_{i,j})$ term separately, by choosing an appropriate $t_{i,j}$. As in Section 3.2, we can define the edge weights $w_{i,j}$ to be these optimized local terms, thus allowing efficient evaluation of the expected score for different topologies. These weights define a combinatorial optimization problem, equivalent to finding the tree with highest (expected log-likelihood) score.

3.3 Computing Expected Counts

Before we consider how to use the expected score in our algorithm, we address the issue of computing expected counts. At each iteration we compute these counts for all edges, not just for branches of the current topology T^0. Recall that

$$\begin{aligned} & E[\mathcal{S}_{i,j}(a,b) \mid D, T^0, \mathbf{t}^0] \\ &= E[\sum_m 1\{X_i[m] = b, X_j[m] = a\} \mid D, T^0, \mathbf{t}^0] \\ &= \sum_m P(X_i[m] = b, X_j[m] = a \mid x_{[1\ldots N]}[m], T^0, \mathbf{t}^0) \end{aligned}$$

Thus, the problem of computing expected counts reduces to the problem of computing conditional probabilities over edges. We solve that using the following observation.

PROPOSITION 3.3. *Let $(T^0, \mathbf{t}^0)$ be a phylogenetic tree. Assume that internal nodes i, j and k are such that j is on the path from i to k in T^0. Then*

$$P(x_i, x_k \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0) = \sum_{x_j} \frac{P(x_i, x_j \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0) P(x_j, x_k \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0)}{P(x_j \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0)} \quad (7)$$

Based on this corollary we design a simple procedure of dynamic programming. We start by computing induced probabilities $P(x_i, x_j \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0)$, for each branch $(i,j) \in T^0$, using the upward-message method described in Section 2. Then for $l = 2, \ldots, N$ we compute $P(x_i, x_k \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0)$ for pairs i, k of nodes that are l branches apart, by choosing x_j on the path between them, and applying Eq. (7). We proceed in this manner, and compute all the conditional probabilities of interest in a quadratic number of steps.

4. STRUCTURAL EM

We now have all the components to describe the heart of the Structural EM algorithm. This algorithm proceeds in iterations. We start by choosing a tree $(T^1, \mathbf{t}^1)$ using, say, Neighbor-Joining. Then we improve the tree in successive iterations. In the l-th iteration, we start with the bifurcating tree $(T^l, \mathbf{t}^l)$ and construct a new bifurcating tree $(T^{l+1}, \mathbf{t}^{l+1})$. The high level idea is to use $(T^l, \mathbf{t}^l)$ to define the expected likelihood measure $Q(T, \mathbf{t} : T^l, \mathbf{t}^l)$ on trees, and then to find a bifurcating tree that maximizes this expected likelihood.

4.1 Structural EM iterations

A Structural-EM iteration consists of two steps, the E-step and the M-step. We now describe these in detail.

To define the expected likelihood we need to compute expected counts:

E-Step: Compute $E[\mathcal{S}_{i,j}(a,b) \mid D, T^l, \mathbf{t}^l]$ for all edges (i,j), as discussed in Section 3.3.

Next, we turn to the maximizing the expected likelihood. This is done in two phases.

M-Step I: Optimize edge lengths by computing, for each edge (i,j) its best length $t^{l+1}_{i,j} = \arg\max_t L_{\text{local}}(E[\mathcal{S}_{i,j}(a,b) \mid D, T^l, \mathbf{t}^l], t)$, as discussed in Sections 3.1 and 3.2.

Now we have the edge lengths that maximize the expected likelihood for each tree. This is similar to standard EM for computing branch lengths, except that we compute edge lengths also for pairs (i,j) that are not adjacent in T^l.

Once we have $\mathbf{t}^{l+1}$, we can define W^{l+1} to be the $2N-2$ by $2N-2$ matrix $\{w^{l+1}_{i,j}\}$, where $w^{l+1}_{i,j} = L_{\text{local}}(E[\mathcal{S}_{i,j}(a,b) \mid D, T^l, \mathbf{t}^l], t^{l+1}_{i,j})$. At this stage, by Theorem 3.2 we have that for any tree T

$$\text{if } W^{l+1}(T) \geq W^{l+1}(T^l) \text{ then } L(T, \mathbf{t}^{l+1}) \geq L(T^l, \mathbf{t}^l) \quad (8)$$

Since we are learning *bifurcating* topologies, the appropriate maximization step is to construct a bifurcating topology T^{l+1} with leaves $1, \ldots, N$ such that $W^{l+1}(T^{l+1})$ is maximized. Unfortunately, finding such a topology is an intractable problem.

THEOREM 4.1. *Let $W = (w_{i,j})$ be a $2N-2$ by $2N-2$ matrix of edge weights. Finding a bifurcating topology T, whose leaves are the nodes $1, \ldots, N$, such that $W(T)$ is maximized is an NP-hard problem.*

Proof: Reduction from s-t-Hamiltonian path. ▮

Fortunately enough, though, we have an alternative solution to this problem. Indeed, finding the maximum weight bifurcating topology is intractable. But as we show, we can efficiently find the maximum weighted topology. This topology is not necessarily bifurcating, however (8) still applies to it and thus it improves the likelihood. In fact, this topology provides the best lower bound on the improvement in the likelihood, according to (8). Once we have such a topology we can transform it into a bifurcating topology T^{l+1}. As we show, this transformation does not change the likelihood of the tree.

Thus, we are using the following maximization step.

M-Step II:

(a) Construct a topology T^{l+1}_* in that maximizes $W^{l+1}(T)$.

(b) Construct a bifurcating topology T^{l+1} such that $L(T^{l+1}_*, \mathbf{t}^{l+1}) = L(T^{l+1}, \mathbf{t}^{l+1})$.

Note that we do not restrict the topology T^{l+1}_* we construct in step (a). Thus, nodes in $1, \ldots, N$ may be of degree greater than 1 and nodes in $N+1, \ldots, 2N-2$ may be of degree different than 3. Thus, we are searching for a maximum spanning tree, and we accomplish this stage using a standard algorithm, e.g., [21].

Step (b) is less trivial, and we discuss it in the next section.

4.2 Transforming a Tree to an Equivalent Bifurcating Tree

Suppose we have a phylogenetic tree $(T, \mathbf{t})$ with $2N-2$ nodes, in which every node i has degree $d(i)$. Our goal is to construct a bifurcating tree $(T', \mathbf{t}')$ that has the same likelihood. In such a tree, every node i would have degree $D(i)$ where:

$$D(i) = \begin{cases} 1 & 1 \leq i \leq N \\ 3 & N+1 \leq i \leq 2N-2 \end{cases}$$

We take advantage of the lack of memory in our models of evolution, and apply a series of likelihood-preserving modifications to the tree. We introduce these transformations in the following propositions.

PROPOSITION 4.2. *Let $(T, \mathbf{t})$ be a phylogeny, and let $N < j \leq 2N-2$. Consider two cases:*

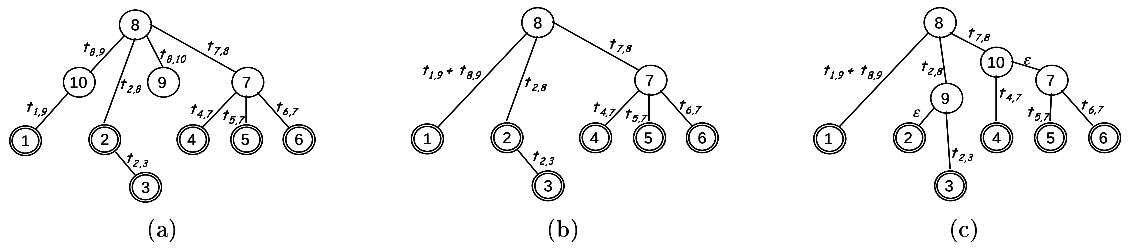

Figure 1: An illustration of the tree modification steps defined in Propositions 4.2 and 4.3. Observed taxa are numbered 1 to 6, and internal nodes are numbered 7 to 10. The transformation from (a) to (b) involves removing two nodes: 9 according to the first case in Proposition 4.2, and 10 according to the second case. The transformation from (b) to (c) involves reinserting these nodes according to Proposition 4.3. Node 9 is inserted since $D(2) = 1$ and $d(2) = 2$, and node 10 is inserted since $D(7) = 3$ and $d(7) = 4$.

- *If $d(j) = 1$, then let $(T', \mathbf{t}')$ be equal to $(T, \mathbf{t})$, except that j is removed.*
- *If $d(j) = 2$, and $(i,j), (j,k) \in T$, then let $(T', \mathbf{t}')$ be equal to $(T, \mathbf{t})$, where (i,k) replaces $(i,j), (j,k)$, $t'_{i,k} = t_{i,j} + t_{i,k}$, and j is removed.*

In either case, $L(T, \mathbf{t}) = L(T', \mathbf{t}')$.

Proof: For the first case, let i be the only neighbor of j. Consider the marginal distribution as a sum of products, as in Eq. (4). Re-order the summation indices in that equation so that x_j is innermost. By Eq. (2), each product in this sum has only one term $p_{a\to b}(t_{i,j})$ involving x_j. Then:

$$P(x_{[1\ldots N]} \mid T, \mathbf{t}) = \sum_{\{x_k \mid k \neq j, N<k\leq 2N-2\}} P(\{x_k \mid k \neq j\} \mid T', \mathbf{t}') \sum_{x_j = b} p_{a\to b}(t_{i,j})$$

But $\sum_{x_j=b} p_{a\to b}(t_{i,j}) = 1$, and the result follows.

The second case follows from Eq. (1), using similar arguments. ▮

Whenever T is not a bifurcating tree, we can use Proposition 4.2 to simplify T by removing a single node. Let T' be the resulting topology. It follows that there must be a node $i > N$ of T' whose degree is larger than 3, or a node $i \leq N$ which is not a leaf. This exactly means that there exists i such that $d(i) > D(i)$.

PROPOSITION 4.3. *Let $(T, \mathbf{t})$ be a phylogenetic tree, let i be a node with $d(i) > D(i)$, and let $i_1, \ldots, i_{d(i)}$ be i's neighbors in T. Let $(T', \mathbf{t}')$ be a tree with a new node i' such that $(T', \mathbf{t}')$ is equal to $(T, \mathbf{t})$ except that:*

- *The branches $(i, i_{D(i)}), \ldots, (i, i_{d(i)})$ are replaced by the branches $(i', i_{D(i)}), \ldots, (i', i_{d(i)})$, whose lengths are set as $t'_{i',j} = t_{i,j}$ for $j = i_{D(i)}, \ldots, i_{d(i)}$.*
- *The branch (i, i') is added and its length is fixed to be $t'_{i,i'} = 0$.*

Then $L(T, \mathbf{t}) = L(T', \mathbf{t}')$.

Proof: For $a \neq b$, $p_{a\to b}(t_{i,i'}) = 0$, and $p_{a\to a}(t_{i,i'}) = 1$. All remaining terms in Eq. (2) remain unchanged when switching from $(T, \mathbf{t})$ to $(T', \mathbf{t}')$. The result follows. ▮

We note that in practice we slightly modify the insertion procedure in two respects. First, we use a small positive duration instead of a zero branch length. This allows later rounds to differentiate the two nodes i, i'. Second, when $D(i) = 3$, we choose the neighbors i_1, i_2 carefully, rather than arbitrarily. In the spirit of the Neighbor-Joining heuristic, we choose, among the neighbors of i, to group the nodes i_1, i_2 which are closest to each other.

As long as our tree is not bifurcating, we can apply the deletion step (Proposition 4.2) and the insertion step (Proposition 4.3), reusing the index of the deleted nodes for reinsertion. Note that the order of deletion/insertion steps is not crucial, thus we can perform all deletions, and then all insertions, or interleave them. Each such operation increases the fraction of nodes in T for which $D(i) = d(i)$, and thus we eventually end up with a bifurcating tree. Figure 1 illustrates the modifications steps of Propositions 4.2 and 4.3.

5. EMPIRICAL EVALUATION

We have implemented our algorithm in a program called SEMPHY. The program is written in C++, and runs on several Unix platforms, as well as Microsoft Windows.

To evaluate the performance of SEMPHY we performed comprehensive evaluation on synthetic data sets. These data sets were generated by constructing an *original* phylogeny, and then sampling its marginal distribution. Each synthetic data set comprised of two sets of sequences: a *training* set and a *test* set. Both sets were simulated assuming the same phylogeny. That is, both consisted of observations drawn from the same marginal distribution, which we wish to characterize. Only the training set has been used for inferring a phylogeny. We graded the inferred phylogenetic tree by two figures of merit, which are the log-likelihood values of each such set (training/test). While the likelihood of the training set is exactly the target of ML optimization, the second figure of merit aims at detecting undesired effects of overfitting the inferred phylogeny to the data. Log-likelihoods were normalized as follows: A baseline, which is the log-

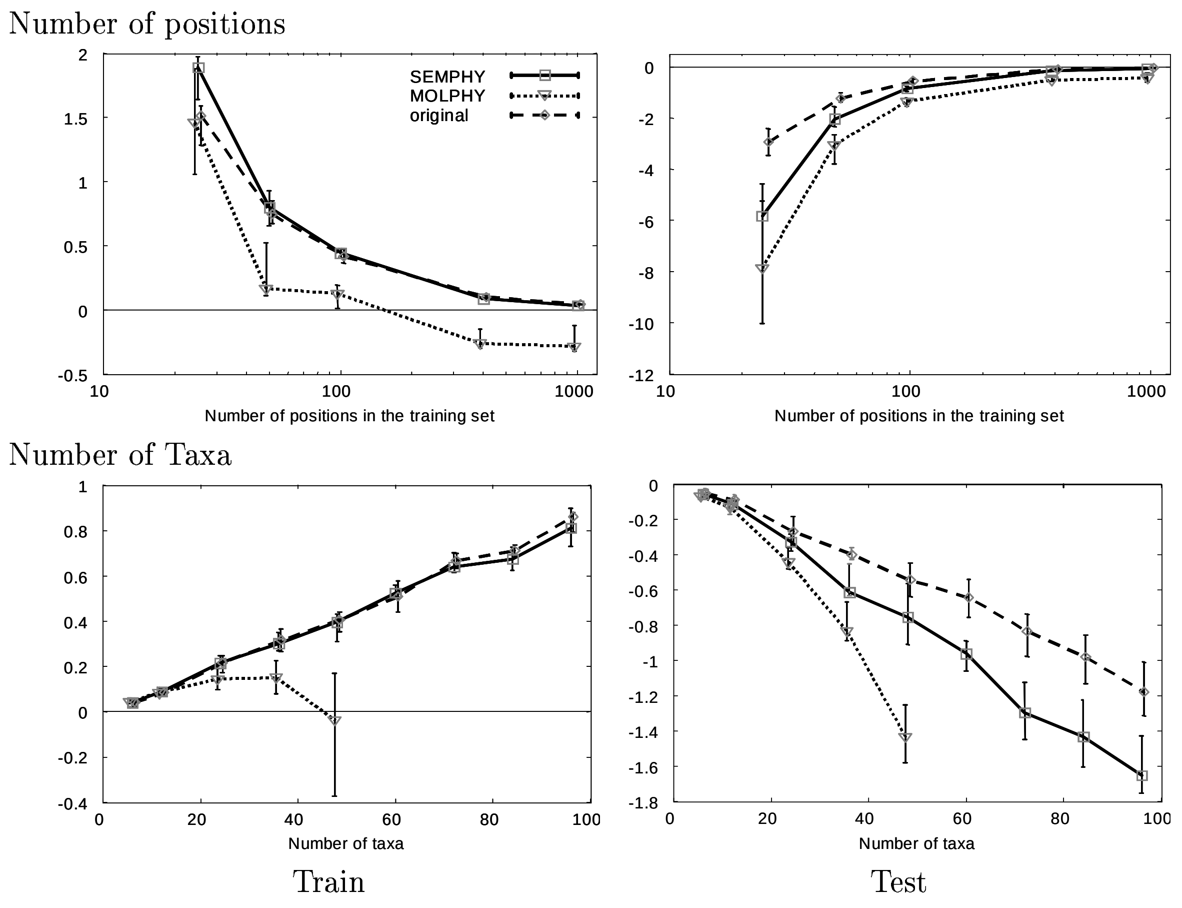

Figure 2: Summary of results from synthetic data. Top row shows the likelihood of reconstructed tree as a function of training sequence length. Bottom row shows the likelihood of reconstructed tree as a function of number of taxa. The left column shows likelihood of trees on the sequences from which they were learned (training data), and the right column shows likelihood on independent sequences sampled from the original tree. Each graph presents three curves: a solid curve for SEMPHY, a dotted curve for MOLPHY, and a dashed curve for the likelihood obtained by optimizing branch lengths for the original topology, according to the training data. The y-axis unit is average log-likelihood per position. The y-axis baseline is the score of the original tree from which the data was sampled. The curves represent 10 independent runs from different trees. The graphs plot the average performance of each method, and the vertical error-bars represent the interval containing 60% of the runs.

likelihood of the original phylogeny, was subtracted from the log-likelihood of the inferred tree. The result was divided by the number of positions.

In our tests, we examined the effect of the number of training positions and the number of taxa on the quality of the learned phylogenies. For this purpose we examined two sets of phylogenies:

- The first consisted of 48 taxa, and different lengths of the training sequence (from 25 to 1000 positions).
- The second consisted of different numbers of taxa (10–100), with training sequences of length 100 positions.

The size of the test-set was 1000 positions for all benchmarks. A series of 10 data sets was generated for each such case. These phylogenies had uniform topology (whose directed analogue is a fully binary tree), and branch lengths that are sampled from the Gamma distribution to simulate a mix of short and potentially very long branches. The distribution parameters, $\alpha = 0.01, \beta = 0.067$, were chosen to fit the data of [25]. Sequences for these trees were simulated using the model of amino-acid evolution of [18].

In these tests we used MOLPHY [2] as the main reference point against which we compared SEMPHY. (Other application, like PAUP or FastDNAML, offer maximum-likelihood solutions only for DNA data.) For comparison, we also considered the likelihood of a tree obtained by optimizing branch lengths for the original topology. The branch lengths for this tree are overfit, by definition, and hence, this comparison provides a clue regarding the effects of overfitting the topology by SEMPHY or MOLPHY.

Figure 2 summarizes the results for these tests. These

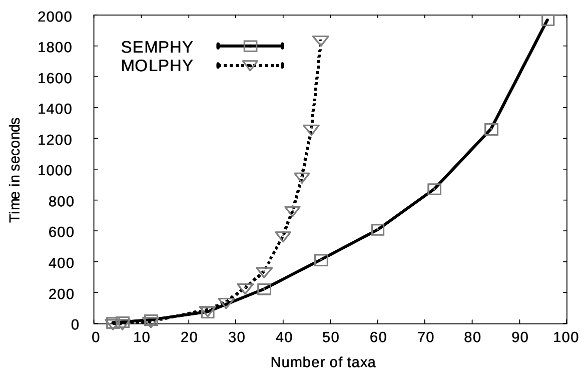

Figure 3: Average running times (on a 600 MHZ Pentium machine) for SEMPHY runs in the bottom chart of Figure 2.

results show several trends.

First, as one can expect, the quality of the learned methods deteriorates when the number of taxa increases and when the number of training positions decreases. In these situations all methods performed worse on the test data. This deterioration includes the original topology with reestimated branch lengths, which indicates that in some sense in these situations we do not have enough information in the training sequence to recover the original phylogeny.

Second, we see that in terms of training data likelihood, the performance of SEMPHY closely tracks the performance of the original topology with reestimated branch lengths. This indicates that SEMPHY finds topologies that, based on the training data, are almost as good as the original ones. (In some situations, they score better than the original topology.) The difference between performances of SEMPHY and the original topology on the test data indicates that the additional knowledge of the original topology helps in finding better approximations of the true phylogeny, which should be expected. As we see, however, for large number of training positions, this difference vanish.

Finally, we see that SEMPHY clearly outperforms MOLPHY in terms of quality of solutions (on both the training set and test set likelihoods). While SEMPHY is close to the performance of the original topology on the training set, we see that MOLPHY is worse, even when the number of training positions grow. We also see that for large number of taxa, MOLPHY performance deteriorates, even according to the training set likelihood. In addition, Figure 3 shows the running time as a function of the number of taxa. We see that MOLPHY's running time grows much faster than SEMPHY's running time. (Note that the running time of both programs grows roughly linearly in the number of training positions.) These results show that SEMPHY's speed allows, for the first time, ML phylogenetic inference on a large scale.

We also applied both SEMPHY and MOLPHY to real data sets, one of nuclear lysozyme proteins [25], and another of concatenated mitochondrial proteins [26]. Both data sets focused on mammalian genes, with the exception of several outgroup species. The proteins in each data set were aligned using CLUSTALW [30], and columns with deleted amino-acids were excluded from the analysis. Naturally, we did not have any test set in these cases.

The lysozyme data set consists of 43 protein sequences, of length 122. SEMPHY has found a tree for this data set, whose overall likelihood is -2892.11. MOLPHY found a tree whose likelihood is -2916.67.

The mitochondrial data set consists of 34 sequences, each being a concatenation of the 13 proteins coded by mammalian mitochondria. The total length of each sequence is 3578. The log-likelihood of the tree we obtain using SEMPHY is -70533.5, compared to -74227.9 attained by MOLPHY. The improvement is of 1.03 on average, per position. The tree we found is identical to the tree available in the biological literature [26].

6. DISCUSSION

This paper presents a new approach for maximum likelihood phylogenetic reconstruction. On the theoretic aspect, we build on existing theory of learning and inference, while on the applicative aspect we show promising results, both on real and synthetic data. This raises many research questions, both theoretic and practical.

Our algorithm assumes a constant rate of evolution. Recently, it was shown that the assumption of rate heterogeneity across sites is statistically superior to the constant-rate assumption [31]. An important extension to this work would be to incorporate this model of variable rates into our development.

This can be posed as a missing data problem where the rate of each position is an additional unobserved variable. Our decomposition of expected log likelihood extends in a natural manner to this case, and thus the general procedure we described can be applied to the more expressive model. Another direction for future research, is to combine the advantages of gradual tree construction (either bottom-up, or top-down), with those of our method. A hybrid approach may prove even faster, and more accurate than existing ones.

This paper still does not explore the power of our method in depth. More thorough examination of its performance, and comparison to several existing methods, are in place. Furthermore, inferring phylogeny for dozens of species should not be the final goal. Rather, the challenge of analyzing hundreds of sequences, in a maximum likelihood framework, seems almost practical.

Finally, it still remains to exploit the new method for extensive biological research. Recent data sets in molecular evolution are becoming bigger and bigger, holding the promise to resolve classical questions about the divergence of life. Although these sets contain lots of potential information, the lack of a fast and accurate inference tool stands in the way of their utilization. We hope our maximum-likelihood based solution will support this analytic endeavor, and promote new insights.

Acknowledgments

We thank Amir Ben-Dor, Chain Linhart, Dana Pe'er, Stuart Russell, and the anonymous RECOMB referees for useful discussion and comments on previous drafts of this paper. This work as supported in part by ISF Grant 244/99, and Israel Ministry of Science Grant 2008-1-99. N. Friedman was supported in part by an Alon Fellowship. I. Pe'er has been supported by the Clore foundation.

7. REFERENCES

[1] J. Adachi. *Modeling of Molecular Evolution and Maximum Likelihood Inference of Molecular Phylogeny.* PhD thesis, The Graduate University for Advanced Studies, Hayama., 1995.

[2] J. Adachi and M. Hasegawa. Molphy version 2.3, programs for molecular phylogenetics based on maximum likelihood. Technical report, The institute of Statistical Mathematics, 1996.

[3] J. H. Camin and R. R. Sokal. A method for deducing branching sequences in phylogeny. *Evolution*, 19:311–326, 1965.

[4] B. Chor, M. D. Hendy, B. R. Holland, and D. Penny. Multiple maxima of likelihood in phylogenetic trees: an analytic approach. In *4th Ann. Int. Conf. Computational Molecular Biology*, pages 108–117, 2000.

[5] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Info. Theory*, 14:462–467, 1968.

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

[7] M. O. Dayhoff. *Atlas of Protein Sequence and Structure, Volume 5, Supplement 3.* National Biomedical Research Foundation, 1978.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc.*, 39:1–39, 1977.

[9] J. Felsenstein. Evolutionary trees from DNA sequences: a maximum likelihood approach. *J. Molecular Evolution*, 17:368–376, 1981.

[10] J. Felsenstein. Phylogenies from molecular sequences: Inference and reliability. *Annual Reviews in Genetics*, 22:521–565, 1988.

[11] J. Felsenstein. *Inferring Phylogenies.* Sinauer Associates, 2000. In press.

[12] N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Proc. 14th Int. Conf. Machine Learning*, pages 125–133. 1997.

[13] N. Goldman and Z. Yang. A codon-based model of nucleotide substitution for protein coding DNA sequences. *Molecular Biology and Evolution*, 11(5):725–736, 1994.

[14] R. L. Graham and L. R. Foulds. Unlikelihood that minimal phylogenies for a realistic biological study can be constructed in reasonable computational time. *Mathematical Biosciences*, 60:133–142, 1982.

[15] M. D. Hendy and D. Penny. Branch and bound algorithms to determine minimal evolutionary trees. *Mathematical Biosciences*, 59:277–290, 1982.

[16] D. H. Huson, S. M. Nettles, and T. J. Warnow. Disk-covering, a fast-converging method for phylogenetic tree reconstruction. *J. Computational Biology*, 6:369–386, 1999.

[17] D. H. Huson, L. Vawter, and T. J. Warnow. Solving large scale phylogenetic problems using DCM2. In *Proc. 7th Int. Conf. Intelligent Systems for Molecular Biology*, pages 118–129, 1999.

[18] D. T. Jones, W. R. Taylor, and J. M. Thornton. The rapid generation of mutation data matrices from protein sequences. *Comp. App. Biosciences*, 8:275–282, 1992.

[19] T. H. Jukes and C. R. Cantor. Evolution of protein molecules. In H. N. Munro, editor, *Mammalian protein metabolism*, pages 21–132. Academic Press, 1969.

[20] M. Kimura. A simple model for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Molecular Evolution*, 16:111–120, 1980.

[21] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Am. Math. Soc.*, 7:48–50, 1956.

[22] P. J. M. Van Laarhoven and E. H. L. Aarts. *Simulated Annealing.* Kluwer, 1988.

[23] P. O. Lewis. A genetic algorithm for maximum likelihood phylogeny inference using nucleotide sequence data. *Molecular Biology and Evolution*, 15(3):277–283, 1998.

[24] G. J. Olsen, H. Matsuda, R. Hagstrom, and R. Overbeek. fastDNAmL: a tool for construction of phylogenetic trees of dna sequences using maximum likelihood. *Comp. App. Biosciences*, 10(1):41–48, 1994.

[25] T. Pupko. *Positive selection and parallel substitution in mammalian evolution - a statistical approach.* PhD thesis, Tel Aviv University, Tel Aviv, 2000.

[26] A. Reyes, C. Gissi, G. Pesole, F. M. Catzeflis, and C. Saccone. where do rodents fit? evidence from the complete mitochondrial genome of Sciurus vulgaris. *Molecular Biology and Evolution*, 17:979–983, 2000.

[27] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[28] R. R. Sokal and P. H. A. Sneath. *Principles of numerical taxonomy.* W. H. Freeman, 1963.

[29] D. L. Swofford. PAUP: Phylogenetic analysis using parsimony. Technical report, Smithsonian Institution, 1993.

[30] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: Improving the sensitivity of progressie multiple sequence alignment through sequence weighting, position specific gep penalties and weight matrix choice. *Nuc. Acid Res.*, 22:4673–4680, 1994.

[31] Z. Yang. Maximum likelihood estimation of phylogeny from DNA sequences when substitution rates differ over sites. *Molecular Biology and Evolution*, 10:1396–1401, 1993.

[32] Z. Yang. Estimating the pattern of nucleotide substitution. *J. Molecular Evolution*, 39:105–111, 1994.

[33] Z. Yang, S. Kumar, , and M. Nei. A new method of inference of ancestral nucleotide and amino acid sequences. *Genetics*, 141:1641–1650, 1995.