A Structural EM Algorithm for Phylogenetic Inference*

Nir Friedman†
School of Computer Science & Engineering
Hebrew University
Jerusalem, 91904, Israel
nir@cs.huji.ac.il

Matan Ninio
School of Computer Science & Engineering
Hebrew University
Jerusalem, 91904, Israel
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
The Institute of Statistical Mathematics
4-6-7 Minami-Azabu, Minatu-ku
Tokyo, Japan 106-8569
tal@ism.ac.jp

Abstract

A central task in the study of molecular evolution is the reconstruction of a phylogenetic tree from sequences of current-day taxa. The most established approach to tree reconstruction is maximum likelihood (ML) analysis. Unfortunately, searching for the maximum likelihood phylogenetic tree is computationally prohibitive for large data sets. In this paper, we describe a new algorithm that uses *Structural EM* for learning maximum likelihood phylogenetic trees. This algorithm is similar to the standard EM method for edge-length estimation, except that during iterations of the Structural EM algorithm the topology is improved as well as the edge length. Our algorithm performs iterations of two steps. In the *E-Step*, we use the current tree topology and edge 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. We show that searching for better topologies inside the M-step can be done efficiently, as opposed to standard methods for topology search. We prove that each iteration of this procedure increases the likelihood of the topology, and thus the procedure must converge. This convergence point, however, can be a sub-optimal one. To escape from such "local optima", we further enhance our basic EM procedure by incorporating moves in the flavor of simulated annealing. We evaluate these new algorithms on both synthetic and real sequence data, and show that for protein sequences even our basic algorithm finds more plausible trees than existing methods for searching maximum likelihood phylogenies. Furthermore, our algorithms are dramatically faster than such methods, enabling, for the first time, phylogenetic analysis of large protein data sets in the maximum likelihood framework.

*A preliminary version of this paper appeared in *Fifth Annual International Conference on Computational Molecular Biology*, 2001. 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. T. Pupko was supported by a JSPS fellowship.

†Contact author.

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 (Camin & Sokal 1965, Sokal & Sneath 1963). 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.

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 Felsenstein (2001) for more details).

One such criterion considers distances between each pair of sequences, and supports the tree that best fits these observed distances. The prominent method which uses this criterion is *Neighbor-Joining* (NJ) (Saitou & Nei 1987), 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 (Day 1983, Graham & Foulds 1982), several effective heuristics exhibit reasonable performance in affordable time (Hendy & Penny 1982, Huson, Nettles & Warnow 1999*a*, Huson, Vawter & Warnow 1999*b*, Swofford 1998, Nixon 1999).

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 edge, such probabilities depend on the duration of the period that this edge represents, i.e., the edge *length*. These probabilities were estimated in many studies (Adachi 1995, Dayhoff 1978, Jones, Taylor & Thornton 1992, Jukes & Cantor 1969, Kimura 1980, Yang 1994). This description of evolution in stochastic terms allows computing the likelihood of a specific phylogeny, and gives rise to Maximum Likelihood (ML) methods (Felsenstein 1981). Indeed, finding the ML phylogenetic tree proves superior to other methods in terms of accuracy (Felsenstein 1988). However, speed is the major obstacle, as we now explain.

ML reconstruction consists of two tasks. The first task involves edge length estimation: Given a topology, find edge lengths to maximize the likelihood. This task is accomplished by iterative methods such as *Expectation Maximization* (EM) (Dempster, Laird & Rubin 1977, Felsenstein 1981), or using Newton-Raphson optimization (Olsen, Matsuda, Hagstrom & Overbeek 1994). Each iteration of these methods requires computations that take on the order of the number of taxa times the number of sequence positions. While these methods are only guaranteed to find local maxima, in practice they often recover the global maximum (Rogers & Swofford 1999).

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 effectiveness of exploring this space by heuristic paradigms, like simulated annealing (Dress & Kruger 1987), genetic algorithms (Lewis 1998), or other local search methods (Swofford 1998), is hampered by the costly procedure of re-estimating edge lengths afresh for different trees. Currently, this task is usually tackled by iterative procedures that greedily construct the desired tree. For protein sequences, the leading ML application is the MOLPHY software package (Adachi & Hasegawa 1996). 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 (Adachi & Hasegawa 1996). In this method, the scoring of each intermediate topology considered requires finding its best edge lengths. The main cost of the algorithm is due to these repeated invocations of edge length optimization. For DNA sequences, we compared performance against a leading application that is included in the PHYLIP package is FastDNAML (Olsen et al. 1994). This program iteratively adds sequences to the tree, while locally rearranging it at each addition step.

Our approach builds on Structural EM, the extension of the EM algorithm for learning combinatorial constructs (Friedman 1997). 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 log likelihood implies an increase in the likelihood itself (Dempster et al. 1977). 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 log 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.

2 Maximum Likelihood Phylogenetic Inference

We are concerned with evolution of biological sequences. Formally, our problem is probabilistic in nature, that is, we assume these sequences are the result of some stochastic process, which we explain in this section. We first describe the evolution of a single character along a single evolutionary lineage. We then introduce the multi-lineage process and its formulae, eventually presenting the problem of multi-character sequence evolution.

2.1 Evolution of a Single Character

We view evolution as a process involving the change (*substitution*) of character from one state into another. These character states 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 a character to transform from state a into state b in the duration t. Such models have been devised, for instance, by (Jukes & Cantor 1969, Kimura 1980, Yang 1994) for nucleotides, (Adachi 1995, Dayhoff 1978, Jones et al. 1992) for amino acids, and (Goldman & Yang 1994) for codons. Different models imply different biological assumptions. However, there are some properties shared by all standard models (Dayhoff 1978):

1. Lack of Memory -
$$p_{a\to b}(t+t') = \sum_{c\in\Sigma} p_{a\to c}(t)p_{c\to b}(t') \tag{1}$$
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 $t \neq 1$, this matrix is raised to the power of t, taking the (a, b) entry of the resulting matrix.

2. Reversibility - we assume that there is a *prior distribution* over character states, $\{p_a\}$ such that
$$p_a p_{a\to b}(t) = p_b p_{b\to a}(t) \tag{2}$$
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.

2.2 Phylogenetic Trees

So far, we have described the evolution of a single character along a single lineage. 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 divergence 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 edges. We use the notation $(i,j) \in T$ to indicate T having a edge between nodes i and j. We reserve the term *edge* for pairs of nodes in T, while using the term *link* 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 *edge length* $t_{i,j}$ for each edge $(i,j) \in T$. The pair $(T, \mathbf{t})$ constitutes a *phylogenetic tree*.

2.3 Joint and Marginal Probabilities

Based on the model of evolution, we can assign probabilistic semantics to the phylogenetic tree. Formally, we associate with each node i in T a random variable X_i that describes the character state 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. We now describe how the evolutionary process assigns random values to all the X_i-s, thus defining the *joint distribution* $P(X_{[1 \ldots N_T]})$:[1]

- Some node r is picked as a root.
- X_r is assigned a random value x_r, according to the prior distribution: $P(X_r = a) = p_a$.
- For each node $i \neq r$ define its *parent*, $\pi(i)$ to be i's neighbor which is closest to the root. (Since T is a tree, this neighbor is uniquely defined.)
- The nodes of T are visited in a preorder traversal. Upon visiting a node $j \neq r$, a random value x_j is assigned to X_j. This value depends solely upon the value assigned to $X_{\pi(j)}$: $P(X_j = b \mid X_{\pi(j)} = a, t_{j,\pi(j)}) = p_{a \to b}(t_{j,\pi(j)})$.

By applying the chain rule, the joint distribution $P(X_{[1 \ldots N_T]})$ is:

$$P(X_{[1 \ldots N_T]} \mid T, \mathbf{t}) = P(X_r) \prod_{j \neq r} P(X_j \mid X_{\pi(j)}, t_{j,\pi(j)}) \tag{3}$$

Recall, that by Eq. 2, $\frac{P(X_i|X_j,t_{i,j})}{P(X_i)} = \frac{P(X_j|X_i,t_{i,j})}{P(X_j)}$. Therefore, manipulating Eq. 3 shows that the joint distribution $P(X_{[1 \ldots N_T]})$ is invariant to the choice of r:

$$P(X_{[1 \ldots N_T]} \mid T, \mathbf{t}) = \left(\prod_i P(X_i) \right) \left(\prod_{(i,j) \in T} \frac{P(X_i \mid X_j, t_{i,j})}{P(X_i)} \right) \tag{4}$$

Usually, we observe only the character states 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

[1] We use the notation $X_{[1 \ldots k]}$ as a shorthand for $X_1, \ldots, X_k$.

these variables:

$$P(x_{[1\dots N]} \mid T, \mathbf{t}) = \sum_{x_{N+1}} \dots \sum_{x_{N_T}} P(x_{[1\dots N_T]} \mid T, \mathbf{t}). \tag{5}$$

The naive approach to computing the marginal probability would require summing over the $|\Sigma|^{N_T-N}$ possible assignments to $X_{[N+1\dots N_T]}$. Instead, we can exploit the structure of the distribution, as specified by Eq. 3, for efficient computation. This is done by dynamic programming over the set of nodes (Felsenstein 1981). We now review this procedure, as we use it in our developments.

Consider a single observed assignment $\{X_1 = x_1, \dots, X_N = x_N\}$, to the leaves of a bifurcating tree $(T, \mathbf{t})$. Each edge $e = (i, j) \in T$, partitions the tree into two subtrees. Define the set $S(i, j)$ to include the leaves of the subtree which includes i. We seek to compute the probability of the observed character states in $S(i, j)$ conditioned on possible assignments to X_i or X_j. More formally, for each character state $a \in \Sigma$, 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 $O(|\Sigma| N)$ time for a single observed assignment to $X_1, \dots, X_N$.

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

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

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

$$P(X_i = a \mid x_{[1\dots N]}, T, \mathbf{t}) = \frac{P(a) U_{i\to j}(a) u_{j\to i}(a)}{P(x_{[1\dots N]} \mid T, \mathbf{t})} \tag{6}$$

and

$$P(X_i = a, X_j = b \mid x_1, \dots, x_N, T, \mathbf{t}) = \frac{P(a) U_{i\to j}(a) p_{a\to b}(t_{i,j}) U_{j\to i}(b)}{P(x_{[1\dots N]} \mid T, \mathbf{t})} \tag{7}$$

2.4 Recovering the Maximum Likelihood Phylogeny

Our main task is recovering the phylogeny from observed data. As discussed above, there are many approaches for addressing this problem. We now describe the simplest variant of the *Maximum Likelihood* method, and the assumptions that justify it.

The input for the process is an *input data set* D that consists of M observations. These are assumed to be drawn from the (unknown) marginal distribution. That is, we have a sequence $x_i[1], \dots, x_i[M]$ of character states for each leaf X_i, and these sequences are assumed to be *aligned* in the following sense: For each m, the character states 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 (i.e., p_a and $p_{a\to b}(\cdot)$) to be known.

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

$$L(T, \mathbf{t}) = P(D \mid T, \mathbf{t}) = \prod_{m=1}^{M} P(x_{[1\ldots N]}[m] \mid T, \mathbf{t}) \tag{8}$$

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 Overview of Our Approach

In the following sections we develop the components needed for the Structural EM procedure. We start, however, with a high-level overview of the procedure and how the following developments fit.

As explained above, the general problem in maximum likelihood reconstruction are the properties of the likelihood function. This function is complex and unwieldy for optimization. For example, a local changes in the topology of the tree can have impact on edge lengths throughout the tree. Thus, the standard approaches for maximum likelihood reconstruction require branch-length optimization after each attempt to modify the tree.

The strategy of Structural EM is to avoid some of these problems by using an alternative scoring function. For efficiency, we want this scoring function to be *decomposable* — the score of a tree is the sum of independent scores for edges of the tree. Optimization of such a scoring function is a combinatorial optimization problem that can be addressed by optimization of each edge separately, and then combining appropriate edges by simple algorithms (such as maximum spanning tree). The main difficulty is how to find such an alternative scoring function, and to understand the relation between maxima of this alternative scoring function and our actual objective, the likelihood function.

In Section 4 we develop the foundation for the *expected log likelihood* score and how to evaluate it. We start by noting that had we observed not only the leaves, but also the ancestral sequences, the likelihood computations would have been much simpler. In fact, it would have been conveniently presented in terms of certain frequency counts, that summarize the input data. In such a case, the likelihood would have been decomposable, and hence, easily optimized. However, ancestral sequences are unobserved. Fortunately enough, assuming a given guess of the tree, we are able to induce probabilities on these sequences and obtain information on their expected content. We can use this information to evaluate the expected log likelihood of other trees.

Thus by using the expected log likelihood score, we can make use of the current guess of the tree topology in order to efficiently evaluate other trees. In Section 5, we show how we use this score to iteratively progress in our search for the maximum likelihood tree. In each iteration we use the tree found by the previous one to define the expected log likelihood score. We then find the tree that optimizes this score. As we show, each iteration finds a tree with higher likelihood, until the process converges.

4 Expected Log Likelihood and Counts

4.1 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 $D = \{x_i[m] : 1 \leq i \leq N, 1 \leq m \leq M\}$ of contemporary sequences, but also the set $H = \{x_i[m] : N+1 \leq i \leq 2N-2, 1 \leq m \leq M\}$ of ancestral sequences. When we observe the values of all $2N-2$ nodes of T for each position, the likelihood function is:

$$L_{\text{complete}}(T, \mathbf{t}) = P(D, H \mid T, \mathbf{t}) = \prod_m P(x_{[1\ldots 2N-2]}[m] \mid T, \mathbf{t}) \tag{9}$$

Note that since in this case we do not marginalize over unobserved nodes, each one of the $P(x_{[1\ldots 2N-2]}[m] \mid T, \mathbf{t})$ terms is a product of conditional probabilities. We rearrange the order of multiplications in Eq. 4 into a more manageable form, as follows:

$$\begin{aligned}
\log L_{\text{complete}}(T, \mathbf{t}) &= \sum_{m=1}^{M} \log P(x_{[1\ldots 2N-2]}[m] \mid T^0, \mathbf{t}^0) \qquad (10)\\
&= \sum_{m=1}^{M} \left[\left(\sum_{i=1}^{2N-2} \log P(x_i[m]) \right) + \left(\sum_{(i,j)\in T} \log \frac{P(x_i[m] \mid x_j[m], t_{i,j})}{P(x_i[m])} \right) \right] \\
&= \left(\sum_{i=1}^{2N-2} \sum_{m \mid x_i[m]=a} \log p_a \right) + \left(\sum_{(i,j)\in T} \sum_{(a,b)\in\Sigma^2} \sum_{m \mid (x_i[m], x_j[m])=(a,b)} \log \frac{p_{a\to b}(t)}{p_b} \right)
\end{aligned}$$

We therefore need to count how many times each conditional probability $\frac{p_{a\to b}(t)}{p_b}$ appears in this expression. 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 $\mathcal{S}_{i,j}(a,b) = \sum_m 1\{X_i[m] = a, X_j[m] = b\}$, and $\mathcal{S}_i(a) = \sum_m 1\{X_i[m] = a\}$.

Proposition 4.1 *The likelihood can be rewritten as:*

$$\log L_{\text{complete}}(T, \mathbf{t}) = \sum_{(i,j)\in T} L_{\text{local}}(\mathcal{S}_{i,j}, t_{i,j}) + \sum_i \sum_a \mathcal{S}_i(a) \log p_a \tag{11}$$

where

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

This formulation is motivated by the approach of Chow & Liu (1968) 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 edge lengths. Thus, when maximizing the likelihood we can ignore the righthand term of Eq. 11. 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

$$\max_{\mathbf{t}} \log L_{\text{complete}}(T, \mathbf{t}) = W(T) + \text{constant} \tag{12}$$

where

$$W(T) \equiv \sum_{(i,j)\in T} w_{i,j}$$

and the constant depends neither on the topology, nor on the edge lengths. 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 link weights. This problem is addressed in Section 5.

4.2 Expected Log Likelihood

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. 8. We use the notion of complete data to help us devise an alternative likelihood function that will guide us in finding high likelihood trees. Improvement of this alternative function guarantees stepping uphill with respect to the true likelihood.

Assume that we are given a data set $D = \{x_i[m] : 1 \leq i \leq N, 1 \leq m \leq M\}$. 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 $(T, \mathbf{t})$:

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

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 4.2 *(based on Friedman (1997)) For any* $T, \mathbf{t}$

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

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:

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

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

Theorem 4.2 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

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

Thus, once 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 4.1, we can define the link 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. In Section 5 we solve this problem, by finding a maximum spanning tree according to these weights.

4.3 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 links, not just for edges of the current topology T^0. Recall that

$$
\begin{aligned}
E[\mathcal{S}_{i,j}(a,b) \mid D, T^0, \mathbf{t}^0] &= E\left[\sum_m 1\{X_i[m]=a, X_j[m]=b\} \mid D, T^0, \mathbf{t}^0\right] && (13)\\
&= \sum_m P(X_i[m]=a, X_j[m]=b \mid x_{[1\ldots N]}[m], T^0, \mathbf{t}^0) && (14)
\end{aligned}
$$

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

Proposition 4.3 *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*

$$P(x_i, x_k \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0) = \sum_{x_j} \frac{P(x_i, x_j \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0) P(x_j, x_k \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0)}{P(x_j \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0)} \qquad (15)$$

Based on this corollary we design a simple procedure of dynamic programming. We traverse the tree T^0 depth first, starting at the root. Upon visiting node i, we first compute $P(x_i, \pi(i) \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0)$ using the upward-message method described in Section 2.3. We then consider each node j visited before i, and compute $P(x_i, x_j \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0)$ from $P(x_i, \pi(i) \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0)$ and $P(\pi(i), x_j \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0)$, by applying Eq. 15. This procedure assures that after each such visit all conditional probabilities for edges that connect already-visited nodes are computed. Thus, the term $P(\pi(i), x_j \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0)$, above is always known prior to the first time is it needed. We proceed in this manner, and compute all the conditional probabilities of interest in a quadratic number of steps.

4.4 Approximated Expected Counts

Computation of these expected counts takes $O\left(M \cdot N^2|\Sigma|^3\right)$ time, which constitutes a major bottleneck in running time. Furthermore, storing these counts requires $O\left(|\Sigma|^2 N^2\right)$ space, while storing the upward messages require only $O\left(|\Sigma| MN\right)$ space. For practical problem size, e.g. a single protein of length 100-300 amino-acids over 50-100 taxa, the first term is the limiting factor. We now detail a method to avoid these complexity problems.

The main idea is to use approximate counts instead of exact ones. We suggest to approximate the counts $E[\mathcal{S}_{i,j}(a,b)] = \sum_m \Pr(X_i[m]=a, X_j[m]=b | X_{1\ldots N}[m])$, by

$$\widetilde{\mathcal{S}}_{i,j}(a,b) = \sum_m P(X_i = a \mid X_{[1\ldots N]}[m]) P(X_j = b \mid X_{[1\ldots N]}[m]) \qquad (16)$$

The reader is referred to Boyen, Friedman & Koller (1999) for justification and further details of this approximation. This latter computation is faster by $O\left(|\Sigma|\right)$ since we do not have to perform the $|\Sigma|$ summation operations of Eq. 15. Note that although this approximation treats X_i and X_j as independent at each position, the overall estimate of the count can capture dependencies between the two positions, because of the summation over positions. We note, that for actual tree edges $(i,j) \in T$, the exact expected count is already known, by Eq. 7, and therefore we do not approximate counts for these edges.

Observe, that the sole purpose of the expected frequency counts is the computation of link weights. Furthermore, the count $\widetilde{\mathcal{S}}_{i,j}(\cdot,\cdot)$ is required only for computing a single weight: w_{ij}. Hence, we can compute these weights one by one, each time constructing only a single count, reusing space. The space complexity is thus $O\left(|\Sigma|^2 N\right)$.

5 Structural EM Phylogenetic Inference

We now have all the components to describe the main steps of the Structural EM algorithm for phylogenetic inference. It is based on the development of the *Structural EM* method in learning Bayesian networks (Friedman 1997). The general outline is similar to the standard EM procedure (Felsenstein 1981), with the important exception that we optimize not only edge lengths, but also the topology during each EM iteration.

Our 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 measure $Q(T, \mathbf{t} : T^l, \mathbf{t}^l)$ of expected log likelihood of trees, and then to find a bifurcating tree that maximizes this expected log likelihood.

5.1 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 log 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 links (i,j), and for all character states $a, b \in \Sigma$, as discussed in Section 4.3.

We then maximize the expected log likelihood in two phases as follows:

> **M-Step I:** Optimize link lengths by computing, for each link (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 4.1 and 4.2.

Now we have the link lengths that maximize the expected log likelihood for each tree. This is similar to standard EM for computing edge lengths, except that we compute link 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 4.2 we have that for any tree T

$$\text{if } W^{l+1}(T) \geq W^{l+1}(T^l) \text{ then } L(T, \mathbf{t}^{l+1}) \geq L(T^l, \mathbf{t}^l) \tag{17}$$

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 5.1 *Let $W = (w_{i,j})$ be a $2N-2$ by $2N-2$ matrix of link weights. Finding a bifurcating topology T, such that $W(T)$ is maximized is an NP-hard problem.*

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

Fortunately, we can bypass this problem. Instead of finding the maximum weight bifurcating topology, we can efficiently find the maximum weighted topology employing maximum spanning tree algorithms. This topology is not necessarily bifurcating, however (17) 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 (17). Once we have such a topology we can transform it into a bifurcating topology T^{l+1} without changing the likelihood of the tree.

Proposition 5.2 *Let* $(T, \mathbf{t})$ *be a phylogenetic tree. Then there is a tree* $(T', \mathbf{t}')$ *such that* T *is a bifurcation with nodes* $1, \ldots, N$ *as leaves, and* $L(T, \mathbf{t}) = L(T', \mathbf{t}')$.

We prove this proposition in Section 5.2.

Using this result we can use the following maximization step.

M-Step II:

(**a**) Construct a topology T_*^{l+1} that maximizes $W^{l+1}(T)$, by finding a maximum spanning tree.

(**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). The 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., (Kruskal 1956). Step (b) is less trivial, and we discuss it in the next section.

To summarize, we describe one Structural EM iteration. We started with a tree $(T^l, \mathbf{t}^l)$ and constructed a tree $(T^{l+1}, \mathbf{t}^{l+1})$ such that $L(T^{l+1}, \mathbf{t}^{l+1}) \geq L(T^l, \mathbf{t}^l)$. We repeat such iterations until convergence to a local maximum.

5.2 Transforming a Tree to an Equivalent Bifurcating Tree

We now prove Proposition 5.2 by describing a procedure that takes an arbitrary tree and returns an equivalently scoring bifurcation.

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 apply a series of modifications to the maximum spanning tree. We take advantage of the lack of memory in our model of evolution to show that these modifications preserve the likelihood of the tree. We introduce these transformations in the following propositions.

Proposition 5.3 *Let* $(T, \mathbf{t})$ *be a phylogeny, and let* $N < j \leq 2N-2$. *Consider two cases:*

- *If* $d(j) = 1$, *i.e.* j *is a leaf, 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. 5. Re-order the summation indices in that equation so that x_j is innermost. By Eq. 3, each product in this sum has only one term $p_{a \to b}(t_{i,j})$ involving x_j. Then:

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

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. ∎

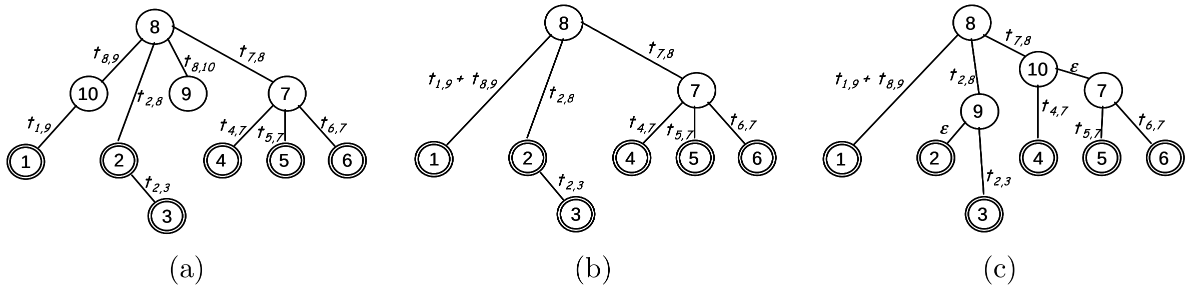


Figure 1: An illustration of the tree modification steps defined in Propositions 5.3 and 5.4. Observed taxa are numbered 1 to 6, and internal nodes are numbered 7 to 10. The transformation from (a) to (b) involves removing two nodes: 9 according to the first case in Proposition 5.3, and 10 according to the second case. The transformation from (b) to (c) involves reinserting these nodes according to Proposition 5.4. Node 9 is inserted since $D(2) = 1$ and $d(2) = 2$, and node 10 is inserted since $D(7) = 3$ and $d(7) = 4$.

Whenever T is not a bifurcating tree, we can use Proposition 5.3 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 5.4 *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 edges* $(i, i_{D(i)}), \ldots, (i, i_{d(i)})$ *are replaced by the edges* $(i', i_{D(i)}), \ldots, (i', i_{d(i)})$*, whose lengths are set as* $t'_{i',j} = t_{i,j}$ *for* $j = i_{D(i)}, \ldots, i_{d(i)}$*.*
- *The edge* (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. 3 remain unchanged when switching from $(T, \mathbf{t})$ to $(T', \mathbf{t}')$. Hence, $L(T, \mathbf{t}) = L(T', \mathbf{t}')$. ∎

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 edge 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. For example, 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 5.3) and the insertion step (Proposition 5.4), reusing the indices of the deleted nodes for re-insertion. Note that the order of deletion/insertion steps may be arbitrary, 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 5.3 and 5.4.

6 Avoiding Local Optima

EM is a greedy method. Indeed, the theory presented thus far can only guarantee that the algorithm described above would find a local maximum, rather than a global one. Unfortunately, local optima are a practical difficulty. Even when initiating Structural EM iterations from random points in our search space, the chances of convergence to the global optimum are very small, for practical problem size.

We propose two strategies to overcome this handicap. The first strategy is to use an educated initialization of the search procedure. Instead of starting the first Structural EM iteration from a random phylogenetic tree, we employ a fast heuristic to find some phylogeny that makes sense, and use that as the initial $(T^0, \mathbf{t}^0)$. In our implementation, we use the Neighbor-Joining heuristic, a renowned standard (Saitou & Nei 1987). This strategy combines both the speed of Neighbor-Joining, and the refined analysis capability of Structural EM. As we show in section 7, it produces impressive results.

A second, more systematic strategy to avoid local maxima incorporates randomness into the EM greedy search. This may enable detection of a global optimum even if they are not in the basin of attraction of the heuristic initial guess. We devote the remainder of this section to the development of a specific, novel method for random search of optimal phylogenies, based on synthesis between Structural EM and *Simulated Annealing*.

Simulated annealing (Kirkpatrick, Gelatt. & Vecchi 1983) is an established paradigm for introducing randomness into greedy optimization procedures, by performing a random tour of the search space. Simulated Annealing has been previously applied to phylogenetic analysis (Barker 1997), even in the maximum-likelihood framework (Dress & Kruger 1987). Topologies are usually considered neighbors if their edge sets differ by exactly one edge, i.e., they can be obtained from one another by a step of *Subtree Pruning and Regrafting*. Another commonly used, narrower definition for neighbors is *Nearest Neighbor Interchange*, that considers swapping of node pairs in internal quartets of the topology.

Such applications of Simulated Annealing tour the likelihood landscape in search for the global maximum. In contrast, we wish to maintain the Structural EM framework, and utilize the convenient function of expected log likelihood. Instead of a random tour that variates the tree topology itself, we perturb the input to the steps that compute the topology. We do rely on Simulated Annealing theory and intuition for incorporating randomness. Each of the next couple of sections presents a specific implementation of this paradigm of *Annealed Structural EM*.

6.1 Annealed Structural EM by Perturbed Edge Weights

Recall that our Structural EM algorithm iteratively computes a $(2N-2) \times (2N-2)$ weight matrix $W = [w_{i,j}]$ where $w_{i,j}$ is the contribution of substitutions along a putative edge (i, j) to the expected log-likelihood of the tree. We next construct a spanning tree T whose total weight $W[T]$ is maximum. This is the optimal construction, with respect to the expected score. Indeed, this score points to a useful "direction" for improvement. However, it is biased by the tree using which we computed the expected sufficient statistics. Thus, at later iterations of the procedure, the trees that maximizes this score will tend to be similar to the tree found in the previous iteration. Furthermore, this self-bias gives rise to stationary points of Structural EM iterations.

A possible solution to these difficulties is to choose trees randomly, in a manner that is guided by the expected score, but not necessarily choose the highest-scoring tree. This will use the expected sufficient statistics to point towards improvements, but can escape local maxima. A simple and well understood method for such weighted randomization is to sample a tree T according to the Metropolis distribution, i.e., with probability proportional to $e^{W[T]/\sigma}$, where σ is an *annealing* *temperature* parameter. Thus, if σ is small, we choose the maximum spanning tree with high probability. When σ is large, we select trees uniformly. The interesting behavior occurs in intermediate ranges of σ, where we select among the many spanning tree topologies whose (expected log-) likelihoods are reasonably close to the maximum. (Note that these topologies are not guaranteed to actually resemble the maximum spanning tree.) Unfortunately, sampling spanning trees from this exponential distribution is a complex task (Propp & Wilson 1998). Instead, we devise an efficient sampling procedure that generates a distribution of spanning trees with similar properties.

We now modify Structural EM by introduction of random noise to the set that constructs the maximum spanning tree. Given the weight matrix W, we construct a new matrix $\widetilde{W}$ with perturbed weights, by adding Gaussian noise to the matrix. Let $\widetilde{w}_{i,j} = w_{i,j} + \epsilon_{i,j}$, where $\epsilon_{i,j}$ are random variables, each normally distributed around zero with variance σ^2. To preserve symmetry, we fix $\epsilon_{j,i} = \epsilon_{i,j}$ for all i, j. Apart from this symmetry constraint, $\{\epsilon_{i,j}\}$ are independent. Instead of computing the maximum spanning tree according to the actual weight matrix W, we use its perturbed version $\widetilde{W}$.

We interleave this sampling procedure with Structural EM iterations. The standard deviation of the white noise assumes the role of the annealing temperature. As in standard annealing, we start with an initial temperature $\sigma = \sigma_0$, and gradually reduce it, e.g. by successive multiplications of the current temperature by a cooling factor ρ. We start with an arbitrary tree $(T^0, \mathbf{t}^0)$, and set the iteration counter l to zero. We then iterate as follows:

E-Step: Compute expected counts for all links (i, j). Same as in Section 5.1.

M-Step I: Optimize link lengths, and compute the weight matrix W^{l+1}, same as in Section 5.1.

Modified M-Step II:

- Perturb W^{l+1} by adding Gaussian noise with variance σ_l^2 to each matrix element. Denote the resulting matrix by $\widetilde{W}^{l+1}$.
- Construct a topology T_*^{l+1} that maximizes $\widetilde{W}^{l+1}(T)$, by finding a maximum spanning tree.
- Construct a bifurcating topology T^{l+1} such that $L(T_*^{l+1}, \mathbf{t}^{l+1}) = L(T^{l+1}, \mathbf{t}^{l+1})$, same as in Section 5.1.
- Set $\sigma_{l+1} \leftarrow \rho \cdot \sigma_l$.

As demonstrated in Section 7, this combination of randomness and Structural EM is very effective. It detects trees which are more likely than those constructed by other methods, and it is capable of escaping local optima.

Intuitively, the perturbation of W has an effect similar to the Metropolis distribution—the probability that T is the maximum spanning tree of $\widetilde{W}$ depends on $W[T]$ and the annealing parameter. The larger the annealing parameter, this distribution becomes more diffused.

To demonstrate this, consider pruning and regrafting of a single subtree of the topology T^* which is the maximum spanning tree with respect to the unperturbed W. Let $e^* = (i^*, j^*)$ be a edge of T^*, and let $e^+ = (i^+, j^+)$ be a candidate edge such that $T^+ = T^* \cup \{e^+\} \setminus \{e^*\}$ is another spanning tree. The difference between the perturbed weights of these trees is:

$$\widetilde{W}[T^+] - \widetilde{W}[T^*] = W[T^+] - W[T^*] + \epsilon_{i^+,j^+} - \epsilon_{i^*,j^*}$$

This quantity is normally distributed with a negative mean $W[T^+] - W[T^*]$ and variance $2\sigma^2$. It follows that the non-greedy move from T^* to T^+ is facilitated with probability

$$\Pr(\widetilde{W}[T^+] > \widetilde{W}[T^*]) = \Phi\left(\frac{W[T^+] - W[T^*]}{\sqrt{2}\sigma}\right)$$

where $\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} exp\left(\frac{-t^2}{2}\right) dt$ is the Gaussian cumulative error function. The decay of this probability is bounded by an exponent in $W[T^+] - W[T^*]$:

Lemma 6.1 (Variation of (Chernoff 1952)) $\Phi(x) < exp(x)$

Hence, the probability that our perturbed Structural EM iteration would prefer T^+ to T^* is smaller than $exp\left(\frac{W[T^+]-W[T^*]}{\sqrt{2}\sigma}\right)$. This bound on the probability of choosing a worse tree, elucidates two properties reminiscent of the Metropolis distribution:

- Decay which is exponential in the score difference, $W[T^+] - W[T^*]$.
- The role of σ as the annealing parameter, that determines the scale in which score differences affect that probability.

6.2 Annealed Structural EM by Perturbed Position Weights

In this section we describe another method to randomize local searches. This method is more general, and can actually be adapted to any optimization of a score which is a sum of independent contributions. In fact, it has already been suggested as a promising heuristic for maximum parsimony phylogenetic reconstruction (Nixon 1999), a non-probabilistic approach. More recently, this paradigm has been demonstrated to perform well on a general setting in computational learning (Elidan, Ninio, Lotner & Friedman 2001).

Instead of randomizing edge weights, we now randomize the summary of the input data. More specifically, we randomly *reweight* each of the positions in the training sequences. This perturbs our expected sufficient statistics, and thereby also leads to perturbed edge weights.

Formally, we replace Eq. 13 by the following:

$$\text{weighted-}E_\Omega[\mathcal{S}_{i,j}(a,b) \mid D, T, \mathbf{t}] = \sum_m \omega_m P(X_i[m] = a, X_j[m] = b \mid x_{[1\ldots N]}, T^0, \mathbf{t}^0) \qquad (18)$$

where $\Omega = \{\omega_m\}$ is the set of random independent positional weights. Below, we discuss the choice of random distributions for generating ω_m. However, we note that as in Section 6.1, a parameter σ, related to the randomization procedure, assumes the role of the temperature in the Metropolis process.

We initially fix the temperature σ_0 and tree $(T^0, \mathbf{t}^0)$, and set the iteration counter l to 0. We then iterate as follows:

Modified E-Step:

- Randomly choose weights $\omega_1, \ldots, \omega_M$, according to the distribution dictated by σ_l.
- Compute weighted-$E_\Omega[\mathcal{S}_{i,j}(a,b) \mid D, T^l, \mathbf{t}^l]$ for all links (i,j), according to Eq. 18.
- Set $\sigma_{l+1} \leftarrow \rho \cdot \sigma_l$.

M-Step I: Optimize link lengths, as in Section 5.1, only using weighted-$E_\Omega[\cdot]$ instead of $E[\cdot]$.
M-Step II: Construct a bifurcating topology T^{l+1} same as in Section 5.1.

We now explain the justification for the above procedure. We focus on a single E-Step, and discuss a problem more general than ours: We consider the abstract task of characterizing a probability distribution based on a data set of M observed samples. The standard approach is to summarize the data by frequency counts. The rationale is that counts reflect our best guess of the true probabilities for each possible sample. This guess, however, is an obvious overfit. It is predisposed towards the samples that were observed, which may not fairly represent the distribution from which they were drawn. We would like to randomly sample the space of possible such distributions, rather than deterministically choose the most likely one.

An intuitive way to achieve that is to resample, with replacement, observations from the input set. This resampling of the input is the principle guiding the bootstrap method, a standard procedure for evaluation of solution significance (Efron 1979). This resampling procedure is used here instead to improve the search for the optimal solution.

A naive resampling policy would repeatedly choose a random observation out of the input set. This is mimicked by initializing zero weights $\omega_1, \ldots, \omega_M$ corresponding to the input observations, and repeatedly incrementing a random member of $\Omega = \{\omega_m\}$. In this spirit, we do assign random weights to positions along our sequence data. However, to avoid discrete weights, we prefer to repeatedly increase each weight ω_m by a random number, rather than by a single unit. These random numbers are chosen according to the Gamma distribution with shape parameter K and scale parameter b:

$$Gamma_{K,b}(x) = \frac{x^{K-1} exp\left(\frac{-x}{b}\right)}{b^K \Gamma(K)}$$

The variance of this distribution equals the inverse of shape parameter: $\sigma^2 = \frac{1}{K}$.

This theory suggests using resampled, or perturbed weights as a randomization mechanism for Structural EM, together integrating into a Simulated Annealing-type algorithm. As in Section 6.1, we follow the guidelines of Simulated Annealing. Again, the standard deviation σ assumes the role of the annealing temperature, and the probability of a weight being Δ away from its mean value decays exponentially with $\frac{\Delta}{\sigma}$.

7 Empirical Evaluation

We have implemented our algorithms in a program called SEMPHY. The program is written in C++, and runs on several Unix platforms, as well as Microsoft Windows. For details regarding availability of SEMPHY, see `http://learning.cs.huji.ac.il/SEMPHY/`. We investigate the performance of our algorithms on both amino-acid and DNA data sets, for both real and simulated data. We first describe the results on the amino-acid sequences whose large alphabet necessitates a complex methodology for phylogenetic analysis.

7.1 Protein Sequences

In these tests we used MOLPHY (Adachi & Hasegawa 1996) as the main reference point against which we compared SEMPHY. (Other applications, like PAUP or FastDNAML, offer maximum-likelihood solutions only for DNA data, rather than protein sequences.)

We first consider the basic Structural EM algorithm, using the Neighbor-Joining tree as a starting point, and apply it to simulated data. We then perform additional simulation studies, to evaluate enhancements to the basic paradigm. Eventually, we apply our methods to real biological data sets.

We first evaluated the performance of the basic Structural EM algorithm. Our evaluation was performed on a comprehensive evaluation of synthetic data sets. These data sets were generated by constructing an *original* phylogeny, and then sampling from 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.

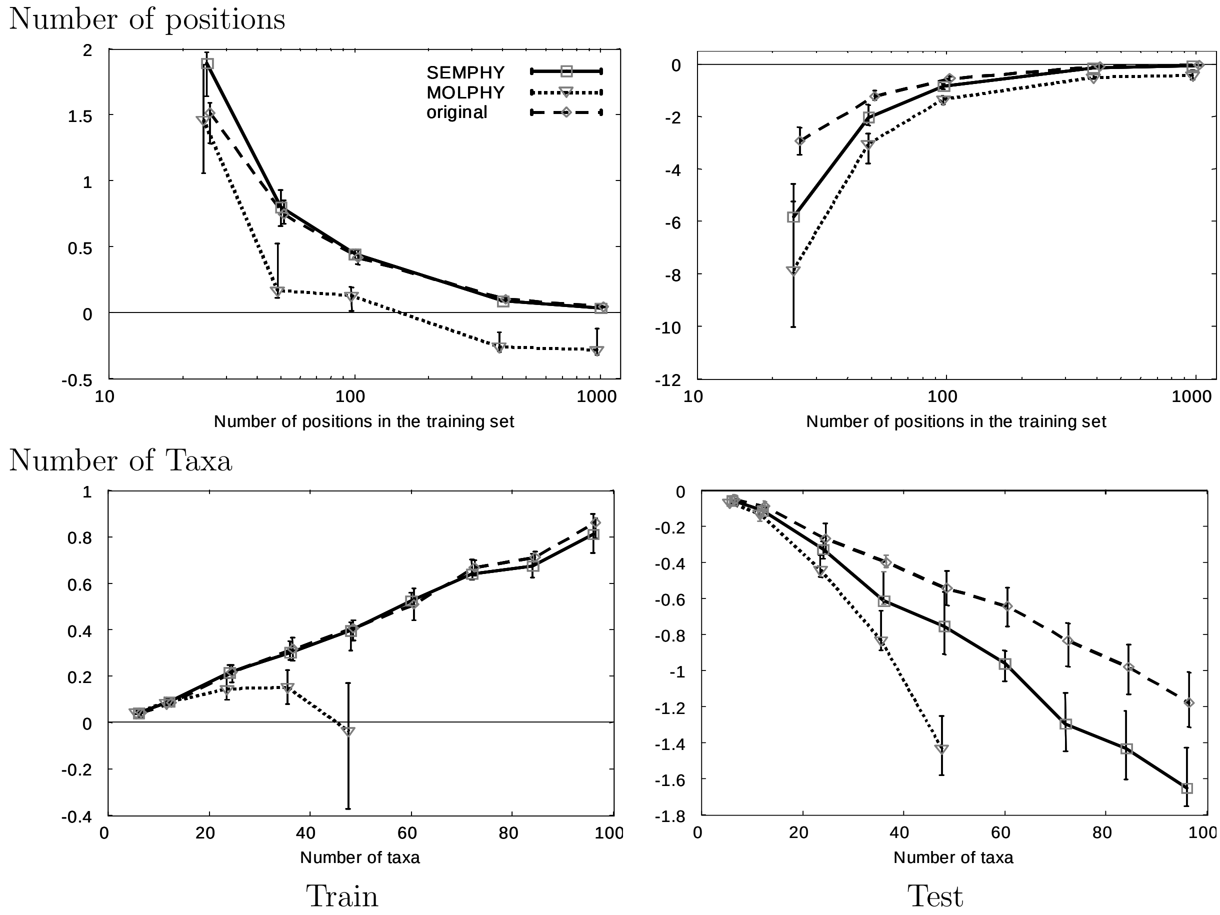


Figure 2: Summary of results from synthetic data. Top row shows the likelihood of reconstructed tree as a function of training sequence length. Bottom row shows the likelihood of reconstructed tree as a function of number of taxa. The left column shows likelihood of trees on the sequences from which 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 edge 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.

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-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 these 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 (12–96), with training sequences of length 100 positions.

The size of the test-set was 1000 positions for all benchmarks. To assess the reproduceability of our results, we repeated these experiments 10 times. That is, we generated 10 "original" trees and from each we sampled training and test data. These phylogenies had uniform topology (whose directed analogue is a fully binary tree), and edge lengths that are sampled from the Gamma distribution to simulate a mix of short and potentially very long edges. The distribution parameters were chosen to fit the data of (Pupko 2000). Sequences for these trees were simulated using the JTT amino-acid replacement model of (Jones et al. 1992).

This gives us a sense about the amount of information that the training data contains about the original model. Since the training data are not perfect, we expect that edge length optimization with the original topology will learn parameters that are somewhat different than these original parameters. By definition, these learned parameters perform better on the training data (since we optimize the training data likelihood). However, they usually perform worse on the test data. (In fact, for test data comprised of sufficiently long sequences, no model can perform better than the original model.) This phenomenon is often referred to as *over-fitting*. The performance of the original tree with parameters estimated from the training data is an upper bound on the performance we can expect from any training method on the particular training data. This performance is therefore a milestone for errors of different methods.

Figure 2 summarizes the results for these tests. These 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 re-estimated edge lengths, which indicates that in some sense in these situations we have less information in the training sequences.

Second, we see that in terms of training data likelihood, the performance of SEMPHY closely tracks the performance of the original topology with re-estimated edge 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 vanishes.

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, which is only quadratic in the number of taxa. (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.

Next, we turned to evaluate enhancements and modifications to the basic approach. Figure 4 demonstrates the validity of approximate counts, discussed in 4.4. Observe, that performance

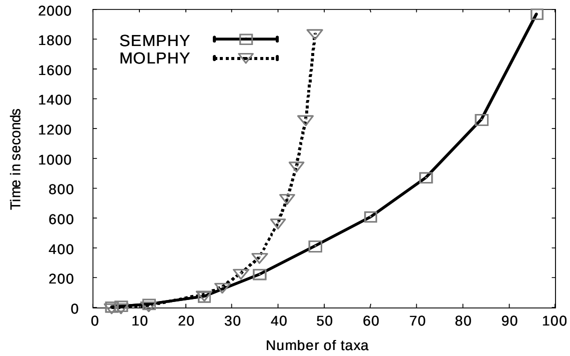


Figure 3: Average running times (on a 600 MHZ Pentium machine) for SEMPHY runs in the bottom chart of Figure 2.

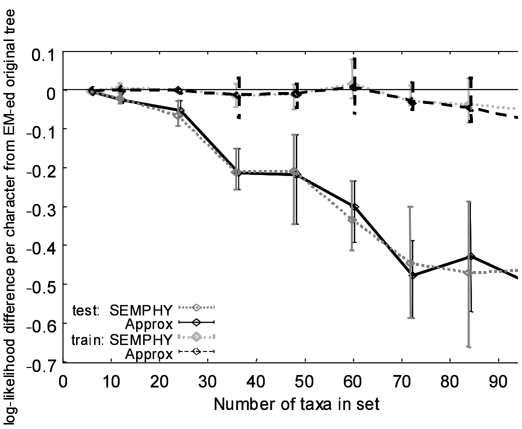


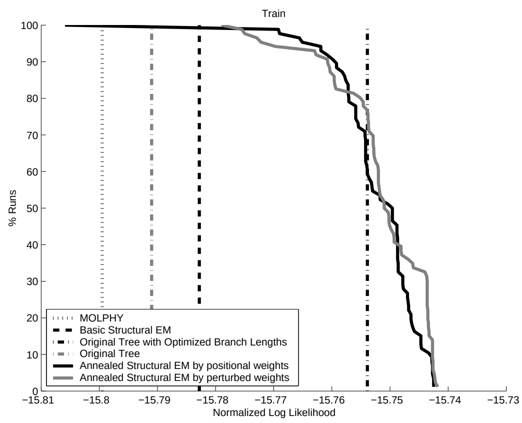
Figure 4: Results of basic Structural EM using approximated counts vs. exact counts. The data sets used were simulated as in the bottom chart of Figure 2. The x-axis represents the number of taxa. The y-axis represents the likelihood of the output tree, measured in units of average log-likelihood per position. The y-axis baseline is the score of the original tree with optimized edge lengths.

is not noticeably degraded by the use of this approximation. Recall, that for protein sequences, approximate counts are 20-times faster than exact ones. Therefore, counts are no longer the limiting factor in iteration running time, which shows a close to 5-fold improvement. This efficiency makes it possible to perform significantly more Structural EM iterations, and thus enables the enhancements discussed in Section 6.

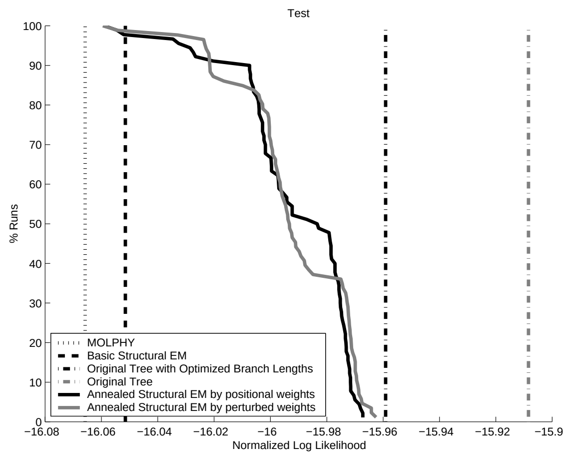
We now analyze the performance of Annealed Structural EM algorithms. Since these are randomized algorithms, we investigate this performance by their repeated applications. We characterize the distribution of both test and train scores of trees that these algorithms find. Figure 5 presents this cumulative distribution for Annealed Structural EM by perturbed weights, and by positional weights.

The most obvious conclusion from Figure 5 is the ability of Annealed Structural EM to find remarkably high-scoring trees.

On both train and test data, MOLPHY and basic Structural EM are no match for the annealed algorithms. On the train data, more than 50% of the runs find trees that more likely than the original tree. This means that by taking the best solution out of few independent runs, will learn



Train



Test

Figure 5: Performance of Annealed Structural EM on train data (top) and test data (bottom). Two data sets (train+test) each of 24 sequences of length 400 were simulated. Each of our two variants of Annealed Structural EM was rerun 80 times against the train data set plotting the cumulative curves of the obtained solution score for each data set. The x-axis denotes the log likelihood per sequence position. The y-axis denotes the fraction of algorithm reruns that produced solutions of this likelihood or better. Several vertical baselines were drawn for comparison to other methods (basic Structural EMMOLPHY) and to the original tree. Annealing initial temperature was set to $\sigma_0 = 0.1$, with the cooling factor $\rho = 0.95$. The stopping criterion is the final temperature $\sigma_l = 0.005$, requiring 60 iterations of the algorithm.

such a high-scoring tree with very high probability. On the test data, our score approaches that of the edge-length-optimized original tree. Lastly, there is no clear winner between the two variants considered. This promotes the claim that randomization helps Structural EM more than the method by which it is incorporated into the Structural EM iterations.

While the basic Structural EM algorithm usually converges in few iterations (often as few as 3), its annealed versions perform around 60 iterations. Still, the complexity is only quadratic. Annealed Structural EM thus enjoys reasonable running times even for large data sets. For example, analysis of 60 protein sequences of length 100 would take an hour, 3-fold faster than MOLPHY.

We also applied both SEMPHY and MOLPHY to real data sets, one of nuclear lysozyme proteins (Pupko 2000), and another of concatenated mitochondrial proteins (Reyes, Gissi, Pesole, Catzeflis & Saccone 2000). Both data sets focused on mammalian genes, with the exception of several outgroup species. The proteins in each data set were aligned using CLUSTALW (Thompson, Higgins & Gibson 1994), 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. Compared to the tree found by MOLPHY with likelihood is -2916.67, we get an improvement of 0.57 per position on average.

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, which corresponds to the best result in the biological literature (Reyes et al. 2000). Compared to the tree likelihood of -74227.9 attained by

Number of Positions	Max Pairwise Distance	Model Tree					
		A	B	AB	C	D	CD
300	0.1	19 (23)	18 (20)	21 (21)	19 (16)	19 (18)	19 (18)
300	0.3	40 (58)	36 (53)	45 (54)	65 (70)	64 (68)	69 (69)
300	1	14 (44)	11 (36)	13 (37)	57 (82)	56 (83)	69 (81)
300	2	0.2 (8)	0.4 (4)	0.6 (5)	20 (59)	21 (62)	18 (59)
600	0.1	69 (69)	55 (63)	58 (65)	66 (66)	60 (61)	61 (62)
600	0.3	80 (93)	74 (87)	76 (91)	95 (94)	93 (97)	91 (96)
600	1	47 (85)	47 (77)	49 (82)	86 (99)	85 (99)	83 (98)
600	2	6 (35)	3 (26)	4 (28)	38 (93)	44 (91)	43 (94)

Table 1: Comparison of SEMPHY and FastDNAML on simulated DNA data. Each table column corresponds a model tree topology, used to simulate the sequences. Each row corresponds to a simulation setup, i.e., the number of positions and combination of branch lengths. Each entry is the percent chance, measured across 1000 simulated data sets, that the original model topology would be identically reconstructed by SEMPHY (FastDNAML).

MOLPHY, the improvement is of 1.03 on average, per position.

7.2 DNA Sequences

We also evaluated the performance of our method on DNA data. Here the performance was compared to the FastDNAML program (Olsen et al. 1994). We followed the experimental design of Ranwez & Gascuel (2001). We thus modeled evolution by Kimura's two parameter model (Kimura 1980), with a ratio of two between transitions and transversions. Six model trees, each of 12 sequences were evaluated. For each tree, 4 combinations of branch length were used, as in (Ranwez & Gascuel 2001). Two sequence lengths (300bp and 600bp) were examined, with 1000 such data sets simulated for each. SEMPHY was run once for each such dataset. FastDNAML was run for comparison.

The results are summarized in Table 1. For evaluation of success, this table uses the criterion of Ranwez & Gascuel (2001): identical reconstruction of the topology of the original model tree. This criterion has a serious flaw, since often the program finds topologies which are more likely than the original tree. We thus further consider the likelihood of the reconstructed tree, compared to the original tree likelihood. Results are summarized in Table 2.

We also compared the performance of both programs on trees which are not artificial. To construct such trees we used a cytochrome-b DNA dataset. This dataset was constructed as follows: One cytochrome-b sequence was chosen for each rodent genus available in GenBank. The aligned dataset contained 133 sequences, and 1138 nucleotide positions. Trees were constructed (by SEMPHY) for subsets of these sequences. For each such tree, and for each prescribed number of positions, we simulated a series of 10 random datasets. We compared average performance of SEMPHY vs. FastDNAML on each such series, by considering the difference in the log-likelihood of the trees these applications reconstruct. This difference is normalized per sequence position.

Results are summarized in Table 3. It appears that the taxa in this dataset are rather close together, and from our evaluation FastDNAML performed very well on this task, almost always exceeding the performance of SEMPHY. We note, that FastDNAML employs a very different, incremental approach for searching the ML tree, locally rearranging the tree upon addition of sequences (Olsen et al. 1994). In principle, it is possible to replace the local rearrangement step by

Number of Positions	Max Pairwise Distance	Model Tree					
		A	B	AB	C	D	CD
300	0.1	93	89	91	99	99	99
300	0.3	68	69	65	93	92	91
300	1	45	46	47	70	68	69
300	2	44	47	48	44	39	39
600	0.1	87	86	86	97	98	97
600	0.3	80	82	81	93	96	94
600	1	58	64	63	84	86	85
600	2	38	37	39	42	49	47

Table 2: Performance of SEMPHY on simulated DNA data. Table rows and columns are as in Table 1. Each entry is the percent chance, measured across 1000 simulated data sets, that the likelihood of the SEMPHY reconstructed tree is higher, or equivalent to the likelihood of the original model topology.

Number of Sequences	100bp	300bp	1000bp	2000bp
10	-0.0028	0.0008	0.0000	-0.0000
15	0.0014	0.0000	0.0000	-0.0000
20	-0.0065	0.0018	-0.0000	-0.0000
30	0.0125	0.0041	0.0010	0.0000
40	0.0136	0.0047	0.0000	-0.0000
80	0.0693	0.0101	0.0033	0.0020
100	0.1255	0.0223	0.0014	0.0004

Table 3: Comparison of SEMPHY and FastDNAML on DNA data synthesized using natural trees. Each table entry registers the difference per position between the log-likelihood of the tree found by FastDNAML and the one found by SEMPHY. These figures are averaged over 10 subsets of the available data.

a Structural EM step. We believe that such a combination of ideas from both FastDNAML and SEMPHY may be superior to both applications.

8 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, to further develop such ideas in phylogenetic context. On the applicative aspect we show good results, both on real and synthetic data. For amino-acid sequences, we find significantly better trees then possible so far. Moreover, our algorithm is computationally efficient. We therefore enable, for the first time, maximum likelihood phylogenetic inference on a large scale for protein sequences. On DNA data-sets SEMPHY's performance is not as good as FastDNAML. Further improvements to the algorithm presented here, such as incorporating step-wise addition of sequences when building the tree, flanked by SEMPHY rearrangements, might prove to be even more effective.

Being a first attempt at applying a completely new strategy for phylogenetic reconstruction, our success to compare with state-of-the-art standard applications, and to systematically outperform them in many cases, proves the potential of our approach, and encourages further study.

This work raises many research questions, both theoretic and practical. Our algorithm assumes a uniform rate of evolution. It was shown that the assumption of rate heterogeneity across sites is statistically superior to the constant-rate assumption (Yang 1993). 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.

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.

Validating Structural EM for more models for DNA evolution (e.g., codon models) is an obvious such examination. One other obvious extension would be to try replacing the local optimization step in FastDNAML with a Structural EM step. 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, Gal Elidan, Joe Felsenstein, Scott Kirkpatrick, Chaim Linhart, Dana Pe'er, Stuart Russell, Arend Sidow, David Swofford and the anonymous referees for useful discussions and comments on previous drafts of this paper.

References

Adachi, J. (1995), Modeling of Molecular Evolution and Maximum Likelihood Inference of Molecular Phylogeny, PhD thesis, The Graduate University for Advanced Studies, Hayama.

Adachi, J. & Hasegawa, M. (1996), Molphy version 2.3, programs for molecular phylogenetics based on maximum likelihood, Technical report, The Institute of Statistical Mathematics, Tokyo, Japan.

Barker, D. (1997), *LVB 1.0: reconstructing evolution with parsimony and simulated annealing*, University of Edinburgh, Edinburgh.

Boyen, X., Friedman, N. & Koller, D. (1999), Discovering the structure of complex dynamic systems, *in* K. Laskey & H. Prade, eds, 'Proc. Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI '99)', Morgan Kaufman, San Francisco.

Camin, J. H. & Sokal, R. R. (1965), 'A method for deducing branching sequences in phylogeny', *Evolution* **19**, 311–326.

Chernoff, H. (1952), 'A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations', *Ann. Math. Statist.* **23**, 493–507.

Chow, C. K. & Liu, C. N. (1968), 'Approximating discrete probability distributions with dependence trees', *IEEE Trans. on Info. Theory* **14**, 462–467.

Day, W. H. E. (1983), 'Computationally difficult parsimony problems in phylogenetic systematics', *Journal of Theoretical Biology* **103**, 429–438.

Dayhoff, M. O. (1978), *Atlas of Protein Sequence and Structure, Volume 5, Supplement 3*, National Biomedical Research Foundation, Washington, D.C.

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977), 'Maximum likelihood from incomplete data via the EM algorithm', *Journal of the Royal Statistical Society* **39**, 1–39.

Dress, A. & Kruger, M. (1987), 'Parsimonious phylogenetic trees in metric spaces and simulated annealing', *Advances in Applied Mathematics* **8**, 8–37.

Efron, B. (1979), 'Bootstrap methods: Another look at the jackknife', *Annals of Statistics* **7(1)**, 1–26.

Elidan, G., Ninio, M., Lotner, N. & Friedman, N. (2001), Weight perturbations for escaping local maxima in learning graphical models, Submitted for publication.

Felsenstein, J. (1981), 'Evolutionary trees from DNA sequences: a maximum likelihood approach', *Journal of Molecular Evolution* **17**, 368–376.

Felsenstein, J. (1988), 'Phylogenies from molecular sequences: Inference and reliability', *Annual Reviews in Genetics* **22**, 521–565.

Felsenstein, J. (2001), *Inferring Phylogenies*, Sinauer Associates, Sunderland, Massachusetts. In press.

Friedman, N. (1997), Learning belief networks in the presence of missing values and hidden variables, *in* D. Fisher, ed., 'Proceedings of the Fourteenth International Conference on Machine Learning', Morgan Kaufman, San Francisco, pp. 125–133.

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

Graham, R. L. & Foulds, L. R. (1982), 'Unlikelihood that minimal phylogenies for a realistic biological study can be constructed in reasonable computational time', *Mathematical Biosciences* **60**, 133–142.

Hendy, M. D. & Penny, D. (1982), 'Branch and bound algorithms to determine minimal evolutionary trees', *Mathematical Biosciences* **59**, 277–290.

Huson, D. H., Nettles, S. M. & Warnow, T. J. (1999*a*), 'Disk-covering, a fast-converging method for phylogenetic tree reconstruction', *Journal of Computational Biology* **6**(3–4), 369–386.

Huson, D. H., Vawter, L. & Warnow, T. J. (1999*b*), Solving large scale phylogenetic problems using DCM2, *in* T. Lengauer et al., eds, 'Proceedings of the Seventh International Conference on SIntelligent Systems for Molecular Biology (ISMB '99)', AAAI press, pp. 118–129.

Jones, D. T., Taylor, W. R. & Thornton, J. M. (1992), 'The rapid generation of mutation data matrices from protein sequences', *Computer Applications in the Biosciences* **8**, 275–282.

Jukes, T. H. & Cantor, C. R. (1969), Evolution of protein molecules, *in* H. N. Munro, ed., 'Mammalian protein metabolism', Academic Press, pp. 21–132.

Kimura, M. (1980), 'A simple model for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences', *Journal of Molecular Evolution* **16**, 111–120.

Kirkpatrick, S., Gelatt., C. & Vecchi, M. (1983), 'Optimization by simulated annealing', *Science* **220**, 671–680.

Kruskal, J. B. (1956), 'On the shortest spanning subtree of a graph and the traveling salesman problem', *Proceedings of the American Mathematical Society* **7**, 48–50.

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

Nixon, K. C. (1999), 'The parsimony ratchet, a new method for rapid parsimony analysis', *Cladistics* **15**, 407–414.

Olsen, G. J., Matsuda, H., Hagstrom, R. & Overbeek, R. (1994), 'fastDNAmL: a tool for construction of phylogenetic trees of dna sequences using maximum likelihood', *Computer Applications in the Biosciences* **10**(1), 41–48.

Propp, J. G. & Wilson, D. B. (1998), 'How to get a perfectly random sample from a generic markov chain and generate a random spanning tree of a directed graph', *Journal of Algorithms* **27(2)**, 170–217.

Pupko, T. (2000), Algorithmic improvements and biological applications of maximum likelihood methods of reconstruction of ancestral amino-acid sequences, PhD thesis, Tel Aviv University, Tel Aviv.

Ranwez, V. & Gascuel, O. (2001), 'Quartet-based phylogenetic inference: Improvements and limits', *Mol. Biol. Evol.* **18**, 1103–1116.

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

Rogers, J. & Swofford, D. (1999), 'Multiple local maxima for likelihoods of phylogenetic trees: A simulation study', *Molecular Biology and Evolution* **16**, 1079–1085.

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

Sokal, R. R. & Sneath, P. H. A. (1963), *Principles of numerical taxonomy*, W. H. Freeman, San Francisco.

Swofford, D. L. (1998), *PAUP*beta: Phylogenetic analysis Using Parsimony*, Sinauer, Sunderland, Massachusetts.

Thompson, J. D., Higgins, D. G. & Gibson, T. J. (1994), 'CLUSTAL W: Improving the sensitivity of progressie multiple sequence alignment through sequence weighting, position specific gep penalties and weight matrix choice', *Nucleic Acids Research* **22**, 4673–4680.

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

Yang, Z. (1994), 'Estimating the pattern of nucleotide substitution', *Journal of Molecular Evolution* **39**, 105–111.