RECOVERING EVOLUTIONARY TREES THROUGH HARMONIC GREEDY TRIPLETS*

MIKLÓS CSŰRÖS AND MING-YANG KAO

(extended abstract for SODA '99)

Abstract. We give a greedy learning algorithm for reconstructing an evolutionary tree based on a harmonic average on triplets of taxa. This algorithm runs in polynomial time in the input size. Using the Jukes-Cantor model of evolution, our algorithm is mathematically proven to require sample sequences of only polynomial lengths in the number of taxa in order to recover the correct tree topology with high probability. In addition to recovering the topology, the algorithm also estimates the tree edge lengths with high accuracy. Our theoretical analysis is supported by simulated experiments, in which the algorithm has demonstrated high success rates in reconstructing a large tree from short sequences.

1. Introduction. Algorithms for reconstructing evolutionary trees are principal tools in biology [11]. These algorithms usually compare aligned character sequences for the taxa in question to infer their evolutionary relationships [19]. In the past, such characters were often categorical variables of morphological features [12]. Newer studies have taken advantage of available biomolecular sequences. As a result, tree construction algorithms have not only enabled the exploration of evolutionary relationships among species but also led to the discovery of new proteins and to the inference of transmission chains of viral diseases such as AIDS [16].

In contrast to the benefits of using evolutionary tree algorithms, almost all the existing algorithms suffer from theoretical or practical drawbacks of requiring exponentially long sample sequences in order to recover the correct tree topology. Among the known algorithms, the class of parsimony algorithms [9] are the most popular. They attempt to compute a tree that minimizes the number of mutations leading to the observed sequences. Unfortunately, the problem of optimizing parsimony is NP-hard [5]. In addition, it is not a consistent method [3, 8] in that increasing the length of sample sequences generated by an evolutionary tree ad infinitum may still not result in the correct topology being inferred. The class of distance-based algorithms are also widely used [19]. These algorithms first calculate pairwise evolutionary distances between taxa from the observed sequences and then build a tree from the resulting distance matrix. Many of these algorithms strive to find an evolutionary tree among all possible trees to fit the observed distances the best according to some metric. Such an optimization task is provably NP-hard in known cases; see [4] for L_1 and L_2 metric norms, and see [1] for L_∞.

This paper presents a new learning algorithm, called *Harmonic Greedy Triplets* (HGT), which is based on a greedy selection of triplets of terminal taxa. This algorithm runs in polynomial time in the input size. In the Jukes-Cantor model of sequence evolution [14], this algorithm is mathematically proven to build the correct evolutionary tree with high probability from sequences of only polynomial lengths in the number of taxa. Thus, the HGT algorithm enables one to construct large trees in a reasonable amount of time from relatively short sequences. The latter feature is immediately useful because the lengths of currently available sequences range from a few hundred to a few thousand only. This feature is even more useful when the number of taxa is as large as in the study of green plant evolution [10]. In addition to returning the correct topology, the algorithm also estimates the edge lengths with high accuracy. Our theoretical analysis is supported by simulated experiments, in which the HGT algorithm has demonstrated high success rates on short sample sequences.

---

*Department of Computer Science, Yale University, New Haven, CT 06520; {csuros-miklos, kao-ming-yang}@cs.yale.edu. Research supported in part by NSF Grant 9531028.

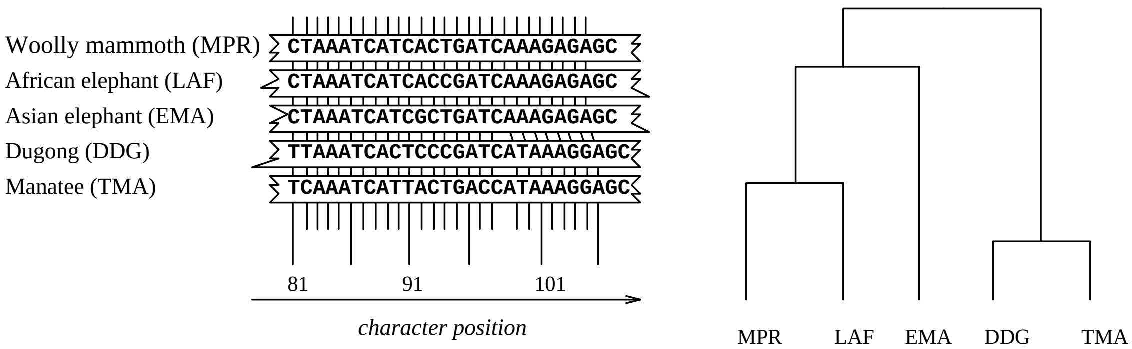

FIG. 2.1. *Taken from* [15]*, this example uses the sequences of 12S ribosomal RNA and cytochrome* b *in mitochondrial DNA to establish the evolutionary relationship among the woolly mammoth and its extant relatives. The sequences shown are for 12S rRNA, base positions 81–110, omitting deletions that are common in the five taxa above.*

At present, the Short Quartet Method (SQM) by Erdős *et al.* [6] is the only other known algorithm with comparable theoretical and experimental performance. The SQM algorithm does not always return a tree while the HGT algorithm does. At a basic level, this is due to the fact that the quartets used by the SQM algorithm contain unnecessary redundancy; when the quartets are inconsistent, it is NP-hard to build a tree that optimally fits these quartets [18]. In contrast, the HGT algorithm uses a small non-redundant set of taxon triplets to build the target tree. The SQM algorithm was originally analyzed for the Cavender-Farris model [3], which is the special binary-character case of the Jukes-Cantor model. The analysis techniques presented in this paper can be used to extend the theoretical analysis of the SQM algorithm to the Jukes-Cantor model. The theoretical and experimental performance of both HGT and SQM algorithms have demonstrated to be superior to such other distance-based algorithms as that of Farach and Kannan [7] and the widely used Neighbor Joining [17] algorithm.

The remainder of the paper is organized as follows. Section 2 reviews the Jukes-Cantor model of sequence evolution and proposes new techniques for distance-based algorithms. Section 3 details the HGT algorithm and proves its sample size requirements and running time. Section 4 discusses the simulated experiments. Section 5 concludes the paper with a summary.

2. Model and techniques.

2.1. Model of sequence evolution.. From a sample of aligned biological sequences, one can build an evolutionary tree (see Figure 2.1) that provides a stochastic model of the succession of pointwise mutations leading to the observed sequences. Without loss of generality, it is supposed that the elements of the sequences are taken from the same finite alphabet such as amino or nucleic acids, codons, etc. The evolutionary tree's nodes represent taxa, each corresponding to a sequence. In particular, leaves represent terminal taxa, each corresponding to an observable sequence. The non-leaf nodes represent the ancestors of these taxa. The mutations occur along the edges.

This paper employs the Jukes-Cantor model [14] of sequence evolution defined as follows. Let

$$\mathcal{A} = \{a_1, \ldots, a_m\}$$

be a a finite alphabet. An *evolutionary tree* T for $\mathcal{A}$ is a binary tree of n leaves with an *edge mutation probability* p_e for each tree edge e. The edge mutation probabilities are bounded away from 0 and $1 - 1/m$, i.e., there exist f and g such that for every edge e of T

$$0 < f \leq p_e \leq g < 1 - \frac{1}{m}.$$

Given a sequence $s_1 \cdots s_\ell \in \mathcal{A}^\ell$, a set of n *mutated sequences*, each of length ℓ, is generated by ℓ random labelings of the tree nodes. The labelings of leaf j give the symbols of the j-th sequence $s_1^{(j)} \cdots s_\ell^{(j)}$, and the i-th labeling of the tree gives the i-th symbols of the sequences $s_i^{(1)}, \ldots, s_i^{(n)}$. The i-th labeling is carried out in the following manner, starting from the root and proceeding towards the leaves. The root is labeled by s_i. On edge e, the child's label is the same as the parent's with probability $1 - p_e$, or is different with probability $p_e/(m-1)$, for each different symbol. More precisely, the label change along edge e is randomly determined by an $m \times m$ *edge transition matrix* $\mathbf{P}_e$ whose entries are given by

$$\mathbf{P}_e[i,j] \stackrel{\text{def}}{=} \Pr\{\, \text{child's label is } j \,|\, \text{parent's label is } i \,\} = \begin{cases} 1 - p_e & \text{if } i = j; \\ \frac{p_e}{m-1} & \text{if } i \neq j. \end{cases}$$

The *topology* $\Psi(T)$ of T is the unrooted tree obtained from T by omitting the edge mutation probabilities and replacing the two edges between the root and its children with a single edge. We further require that the leaves of $\Psi(T)$ are labeled with the same sequences as in T, but $\Psi(T)$ need not be labeled otherwise. Our task is to design a learning algorithm that takes n mutated sequences as input and recovers $\Psi(T)$ with high probability.

2.2. Probabilistic and algebraic techniques.. In this stochastic model of sequence evolution, the node labels can be viewed as random variables taking values on the alphabet $\mathcal{A}$, for which the joint probability distribution is defined by T. The HGT algorithm is based on a notion of closeness of the leaves. The closeness of two random variables X, Y, corresponding to random labels of two nodes in T is defined as

$$\sigma_{XY} \stackrel{\text{def}}{=} \Pr\{\, X = Y \,\} - \frac{1}{m-1} \Pr\{\, X \neq Y \,\}. \tag{2.1}$$

Note that $\sigma_{XY} = \sigma_{YX}$. If X and Y are the endpoints of edge e, then $\sigma_{XY} = 1 - p_e m/(m-1)$. Since the factor $m/(m-1)$ appears often in our analysis, we define

$$\alpha \stackrel{\text{def}}{=} \frac{m}{m-1}.$$

Notice that α equals two for $m = 2$ and tends to 1 as the alphabet size grows.

The following lemma shows that the closeness values can be calculated using the edge mutation probabilities in T.

Lemma 2.1. *Let X and Z be random variables corresponding to two random node labels in T.*

(i) If X and Z are the endpoints of the edge e, then $\sigma_{XY} = 1 - \alpha p_e = (\det \mathbf{P}_e)^{1/(m-1)}$.

(ii) If node Y lies on the path between X and Y, then $\sigma_{XZ} = \sigma_{XY}\sigma_{YZ}$.

Proof. (*Sketch.*) The proof of Part (i) is straightforward using the definitions of σ_{XY} and $\mathbf{P}_e$. Part (ii) is a consequence of Part (i) and the independent edge mutation model. □

The closeness of any two leaves can be estimated using the sample as

$$\hat{\sigma}_{XY} = \frac{1}{\ell} \sum_{i=1}^{\ell} I_{X_i Y_i}, \tag{2.2}$$

with

$$I_{xy} = \begin{cases} -\frac{1}{m-1} & \text{if } x \neq y; \\ 1 & \text{if } x = y, \end{cases}$$

where $X_1, \ldots, X_\ell$ and $Y_1, \ldots, Y_\ell$ are the observed symbols in positions $1, \ldots, \ell$ of the sequences for the two leaves X and Y.

The next lemma is useful for analyzing the deviations of the estimation given by Equation (2.2).

LEMMA 2.2. *Let $\hat{\sigma}_{XY}$ be the estimate of σ_{XY} given by Equation* (2.2). *Then the following inequalities hold when using sample sequences of length ℓ. For $\epsilon > 0$,*

$$\Pr\left\{ \frac{\hat{\sigma}_{XY}}{\sigma_{XY}} \leq 1 - \epsilon \right\} \leq \exp\left(-\frac{2}{\alpha^2} \ell \sigma_{XY}^2 \epsilon^2 \right); \tag{2.3}$$

$$\Pr\left\{ \frac{\hat{\sigma}_{XY}}{\sigma_{XY}} \geq 1 + \epsilon \right\} \leq \exp\left(-\frac{2}{\alpha^2} \ell \sigma_{XY}^2 \epsilon^2 \right). \tag{2.4}$$

Proof. Straightforward by using Hoeffding's inequality [13] on sums of bounded random variables. □

The *distance* of two nodes X and Y is defined as $\Delta_{XY} \stackrel{\text{def}}{=} -\ln \sigma_{XY}$. By Lemma 2.1, distance is an additive measure along any path in the tree. For every edge e with endpoints X and Y, the *edge length* is defined as $\Delta_e = \Delta_{XY}$. In most stochastic models of evolution, Δ_e is directly related to the divergence time between species X and Y. Therefore, an evolutionary tree building algorithm should not only recover the topology $\Psi(T)$, but estimate the edge lengths as accurately as possible. The HGT algorithm estimates the edge lengths from short sequences very accurately.

If X and Y are leaves, their distance can be estimated from the sample sequences as

$$\hat{\Delta}_{XY} = \begin{cases} -\ln \hat{\sigma}_{XY} & \text{if } \hat{\sigma}_{XY} > 0; \\ \infty & \text{otherwise.} \end{cases} \tag{2.5}$$

The pairwise distances can be arranged in a matrix format, which is the input to many distance-based algorithms. Our analysis exploits the fact that the distances are estimated from an input sample, enabling us to derive probabilistic assertions about the problem. We note in passing that most algorithms, which are not based on a similar analysis, need an accurate estimation of distances between nodes that are far apart. Such an estimation may require exponentially long sequences. Moreover, in lack of a sufficiently long sample, the algorithms are likely to output a hypothesis tree with an incorrect topology. This problem is further discussed below in §2.3.

2.3. Graph-theoretic techniques.. The *path length* d_{XY} is the number of edges on the tree path between X and Y. Since for every edge e in T, $p_e \leq g$, we have $\sigma_{XY} \leq (1-\alpha g)^{d_{XY}}$. Therefore, the estimation of σ_{XY} requires sequences that are exponentially long in d_{XY}, and the farther X and Y are, the harder it is to estimate σ_{XY} and Δ_{XY}.

In light of this observation, one naturally attempts to build T using only pairs of leaves that are close to each other. The efficiency limits of such methods are well captured by the notion of depth, which we define as follows, similarly to Erdős *et al.* [6].

Root depth of a tree is the shortest path's length leading from the root to a leaf. For an arbitrary edge e, the *edge depth* of e is defined as the maximum of the root depths of the two subtrees obtained after removing e by rooting them at the endpoints of e. The *depth of the tree* is the maximum edge depth.

A binary tree with n leaves has always $\mathcal{O}(\log n)$ depth. The depth can be as small as $\Theta(1)$ for unbalanced trees such as the "caterpillar" tree, in which at least one child of every internal node is a leaf. A learning method using only distances between close nodes requires the estimation of values at least as large as $(1-\alpha g)^{2 \cdot \text{depth} + \Theta(1)}$, as opposed to $(1-\alpha g)^{\text{diameter}}$, where the diameter

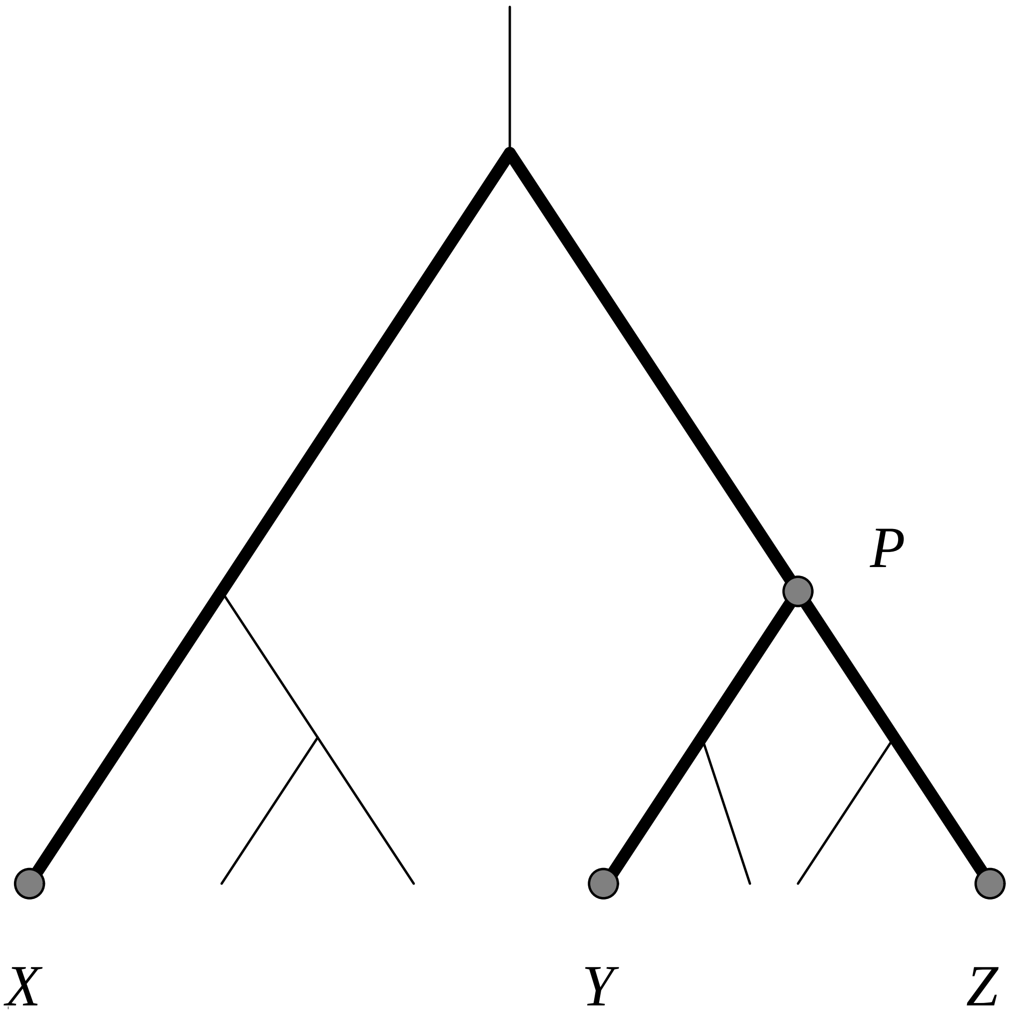

FIG. 3.1. *Node P is the center of a triplet formed by the leaves X, Y, and Z.*

is defined as the longest path length in the tree. The diameter of a binary tree with n leaves is larger than twice the depth, is always $\Omega(\log n)$, and can be as large as $\Theta(n)$ for unbalanced trees. The importance of the depth vs. diameter difference was pointed out by Erdős *et al.* [6]. The SQM algorithm uses distances of close nodes, which results in the fact that a sample of size $\mathcal{O}\left(\frac{\log n}{f^2(1-2g)^{4\cdot\mathrm{depth}}}\right)$ is enough to recover the correct topology with probability $1 - o(1)$. The HGT algorithm, like the SQM algorithm, only uses distances of close nodes and has the same asymptotic sample size requirements. An example of different approaches is Farach and Kannan's method [7]. Their algorithm first "fixes" the distances $\hat{\Delta}$ derived from the sample in order to make the distance matrix additive and thus possible to be generated by a tree. Their algorithm needs a sample of size $\mathcal{O}\left(\frac{\log n}{f^2(1-2g)^{2\cdot\mathrm{diameter}}}\right)$ to recover the correct topology with probability $1 - o(1)$.

3. The HGT algorithm..

3.1. Triplet centers.. A *triplet* XYZ refers to one consisting of three distinct leaves X, Y, and Z of T. Every triplet defines an internal node at which the three pairwise paths between the leaves intersect (see Figure 3.1), with the four nodes forming a star-shaped topology. This internal node is called the *center* of the triplet. Given any triplet of leaves X, Y and Z, their distances from the center P can be obtained by the straightforward equation $\Delta_{XP} = (\Delta_{XY} + \Delta_{XZ} - \Delta_{YZ})/2$,

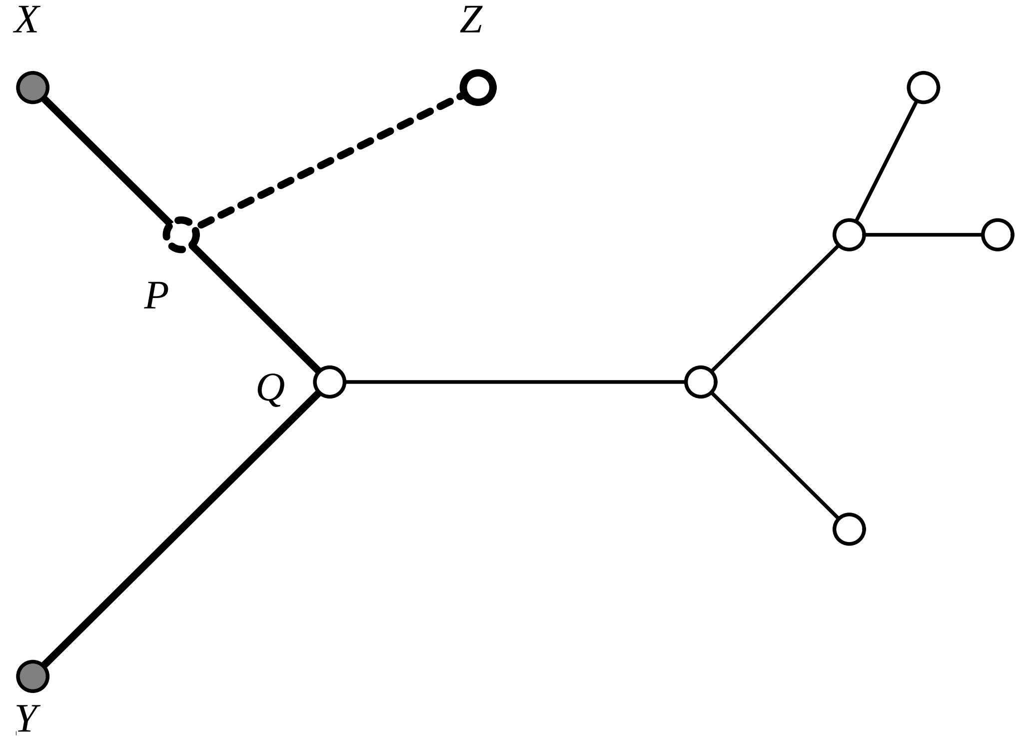

FIG. 3.2. *A new internal node P and a new leaf Z are added by using the leaves X and Y already in the tree. Here $\hat{\Delta}_{XP} < \hat{\Delta}_{XQ}$ supposedly, so P is added on the edge XQ.*

based on the additivity of the distances. The HGT algorithm selects triplets of leaves to find the internal nodes by using the distance estimation

$$\hat{\Delta}_{XP} = \frac{\hat{\Delta}_{XY} + \hat{\Delta}_{XZ} - \hat{\Delta}_{YZ}}{2}, \tag{3.1}$$

given that all three pairwise distance values are finite.

The *average closeness* of a triplet is the harmonic mean of the pairwise closeness values

$$\sigma_{XYZ} \stackrel{\text{def}}{=} \frac{3}{\frac{1}{\sigma_{XY}} + \frac{1}{\sigma_{XZ}} + \frac{1}{\sigma_{YZ}}}.$$

The next lemma shows that the accuracy of the estimation in Equation (3.1) depends on the triplet's average closeness.

LEMMA 3.1. *Let XYZ be a triplet with center P. Then for $0 < \epsilon < 1$,*

$$\Pr\left\{ \hat{\Delta}_{XP} - \Delta_{XP} \geq \frac{-\ln(1-\epsilon)}{2} \right\} \leq 3e^{-\frac{2}{9\alpha^2}\ell\sigma^2_{XYZ}\epsilon^2}.$$

Proof. (*Sketch.*) The proof consists of conditioning the probability on the left-hand side on the estimation accuracy in the three terms of Equation (3.1). The arising conditional probabilities are bounded from above using Equations (2.3) and (2.4). Balancing the terms leads to the Lemma. □

3.2. Building the tree.. To grow a partially built tree T^*, the HGT algorithm picks a leaf not yet in T^*, and adds a new internal node as the center of the triplet formed by the new leaf and two other leaves already in the tree (see Figure 3.2). This step is repeated until T^* has a leaf for every sample sequence. The only question is how to pick the triplets so that the true tree T and the hypothesis tree T^* have the same topology with high probability. Based on Lemma 3.1, the algorithm uses a greedy approach by selecting a triplet with the maximal average closeness among the possible choices. We show that this greedy selection can help recover the correct topology from a polynomial-sized sample with high probability.

The greedy selection alone is insufficient because different triplets may define the same internal node and we may end up with an erroneous topology by adding the same nodes more than once. It is essential to refine the selection by including a criterion to recognize separate inner nodes. Our criterion is based on the fact that the tree nodes cannot be arbitrarily close since the distances are bounded from below by $-\ln(1-\alpha f)$. The HGT algorithm adopts a threshold

$$\Delta_{\min} \leq \lambda \frac{-\ln(1-\alpha f)}{2},$$

with $0 < \lambda \leq 1$. The factor λ governs the algorithm's accuracy in the edge length estimation. The algorithm considers two nodes separate if they are at least $\Delta_{\min}$ apart when using distances estimated from a sample. More precisely, the center P of a triplet XYN and the center Q of triplet XUV are determined to be separate if $|\hat{\Delta}_{XP} - \hat{\Delta}_{XQ}| \geq \Delta_{\min}$, where $\hat{\Delta}_{XP} = (\hat{\Delta}_{XY} + \hat{\Delta}_{XN} - \hat{\Delta}_{YN})/2$ and $\hat{\Delta}_{XQ} = (\hat{\Delta}_{XU} + \hat{\Delta}_{XV} - \hat{\Delta}_{XV})/2$. Notice that two triplet centers can only be compared this way if the triplets share at least one leaf. As the HGT algorithm builds T^*, it keeps track of each internal node's defining triplet used for adding that node to T^*. The test whether a triplet XYN can be used to add an internal node P and the leaf $N \notin T^*$ is implemented in a subroutine called Split-Edge. This subroutine also computes the edge length estimates Δ^*_{PN}, Δ^*_{PQ}, $\Delta^*_{PQ'}$, when P is inserted on edge $QQ' \in T^*$. The details of the subroutine are discussed in §3.3.

In practice, if a threshold $\Delta_{\min}$ is not readily available, i.e., if f is not known, then the HGT algorithm can be run iteratively, decreasing the threshold until we succeed in building a complete tree. The ending value of the iteration can be based on the sample sequence length ℓ, since the smallest observable probability is $1/\ell$. This iterative approach results in an extra $\mathcal{O}(\log \ell)$ factor in the running time.

3.3. Updating the tree.. The HGT algorithm returns an unrooted tree T^*, with edge weights Δ^*_e for every edge $e \in T^*$. The hypothesis tree is initialized by a triplet ABC with its center D. The edge lengths Δ^*_{AD}, Δ^*_{BD} and Δ^*_{CD} are set to $\hat{\Delta}_{AD}$, $\hat{\Delta}_{BD}$ and $\hat{\Delta}_{CD}$, respectively, calculated using Equation (3.1). In order to estimate the edge lengths in subsequent steps and to efficiently test if a triplet center is separate from the nodes already in T^*, the HGT algorithm keeps track of the defining triplets used for adding new internal nodes. For every internal node $Q \in T^*$, the set $\mathit{Def}(Q)$ is defined so that $\mathit{Def}(Q) = \{X, Y, N\}$ if Q was added to T^* using the triplet XYN. For sake of completeness and simplicity, $\mathit{Def}(X)$ is defined for every leaf also, by $\mathit{Def}(X) = \{X\}$. Using Def, the Split-Edge subroutine determines efficiently whether a triplet can be used to add new nodes.

Let NXY be a triplet with $X, Y \in T^*$ and $N \notin T^*$. Let QQ' be an edge in T^*. Then the center P of NXY can be compared to Q and Q' if $\{X, Y\} \cap \mathit{Def}(Q) \neq \emptyset$ and $\{X, Y\} \cap \mathit{Def}(Q') \neq \emptyset$. In this case the triplet NXY is said to *match* the edge QQ'.

The Split-Edge subroutine's input is an edge QQ' of T^* and a matching triplet NXY with $N \notin T^*$, $X, Y \in T^*$. The subroutine tests whether NXY defines a new internal node on QQ' and if so, it updates T^* by adding the internal node P and the leaf N and setting the new edge lengths. Without loss of generality, we may assume that $X \in \mathit{Def}(Q)$. Let $\hat{\Delta}_{XQ} = 0$ if Q is a leaf, i.e., if $X = Q$. Otherwise, calculate $\hat{\Delta}_{XQ}$ using Equation (3.1) with $\mathit{Def}(Q)$. Calculate $\hat{\Delta}_{XP}$ also using Equation (3.1) in conjunction with triplet NXY. If $|\hat{\Delta}_{XP} - \hat{\Delta}_{XQ}| < \Delta_{\min}$, then Split-Edge determines that P is not a new internal node, and returns the message "too close".

It is also possible that P lies outside of the edge QQ'. If Q' is on the tree path between X and Q in T^* and $\hat{\Delta}_{XP} > \hat{\Delta}_{XQ}$, then Split-Edge returns the message "outside this edge". This happens also when Q is on the path between X and Q' and $\hat{\Delta}_{XP} < \hat{\Delta}_{XQ}$.

The same calculations and tests are carried out to compare Q' and P, using a leaf $X' \in \{X, Y\} \cap \mathit{Def}(Q')$. Assume that P is neither "too close" nor "outside this edge" and consequently T^* gets updated. Let $\hat{\Delta}_1 = \hat{\Delta}_{XP} - \hat{\Delta}_{XQ}$ and $\hat{\Delta}_2 = \hat{\Delta}_{X'P} - \hat{\Delta}_{X'Q'}$. $|\hat{\Delta}_1|$ provides an estimate of Δ_{PQ}. Similarly, $|\hat{\Delta}_2|$ is an estimate of $\Delta_{PQ'}$. Using the edge length $\Delta^*_{QQ'}$ in T^*, the new edge lengths are calculated as

$$\Delta^*_{PQ} = \frac{\Delta^*_{QQ'} \pm \hat{\Delta}_1 \mp \hat{\Delta}_2}{2},$$

where the $\pm$ and $\mp$ signs depend on whether Q' lies on the path between Q and X and whether Q lies on the path between Q' and X', in a straightforward manner. The edge length estimate $\Delta^*_{PQ'}$ is calculated as $\Delta^*_{QQ'} - \Delta^*_{PQ}$. The edge length estimate Δ^*_{NP} is simply set to $\hat{\Delta}_{NP}$ using Equation (3.1) with triplet NXY. At this point, Split-Edge inserts the new nodes with the estimated edge lengths and returns the message "OK" . This mechanism avoids the accrual of edge length estimation errors. Consequently the subroutine can detect separate nodes efficiently and estimate edge lengths accurately.

3.4. Outline of the algorithm.. The HGT algorithm is outlined in Figure 3.3.

3.5. Running time and sample size.

THEOREM 3.2. *The running time of the* HGT *algorithm on a sample consisting of n sequences of length ℓ is $\mathcal{O}(n^5 + \ell n^2)$.*

Proof. The closeness values in line (4) can be computed in $\mathcal{O}(\ell n^2)$. There are at most $n-3$ iterations on the candidate set $\mathcal{S}$ with size at most $\mathcal{O}(n^3)$, and Split-Edge is called with $\mathcal{O}(n)$ edges for each triplet in line (16). One can implement $\mathcal{S}$ as a balanced binary search tree. Then, the insertions and deletions of triplets take $\mathcal{O}(n^3 \log n)$ time over all iterations of the repeat-loop. □ *Remark.* We conjecture that the term $\mathcal{O}(n^5)$ can be substantially reduced with high probability.

THEOREM 3.3. *Assume that the evolutionary tree T has n leaves and depth d. Let $0 < \delta < 1$. There exists an*

$$\ell = \mathcal{O}\left(\frac{\log \frac{1}{\delta} + \log n}{(1-\alpha g)^{4d+8} f^2 \lambda^2}\right) \tag{3.2}$$

such that with probability at least $1-\delta$, the HGT *algorithm outputs a T^* satisfying both statements below:*

1. *$T^* = \Psi(T)$.*
2. *For each edge PQ in T^*,*

$$|\Delta^*_{PQ} - \Delta_{PQ}| < \lambda(-\ln(1-\alpha f)).$$

Proof. (*Sketch.*) Define the closeness thresholds $\sigma_{\text{lg}} = (1-\alpha g)^{2d+4}/17$ and $\sigma_{\text{sm}} = \sigma_{\text{lg}}/\sqrt{2}$. Define the *large triplets* as those with average closeness at least σ_{lg}, and the *small triplets* as those with average closeness at most σ_{sm}. This definition is motivated by the fact that there is a defining triplet of leaves for every internal node such that the pairwise path lengths between the leaves are at most $2d+2$. Consequently, large triplets define the tree.

Let E_g denote the event that $\hat{\sigma}_{XYZ} > \hat{\sigma}_{X'Y'Z'}$ for every large triplet XYZ and every small triplet $X'Y'Z'$. Let E_c denote the event that for every triplet XYZ that is not small, $\left|\hat{\Delta}_{WP} - \Delta_{WP}\right| <$

(1) **Algorithm** Harmonic Greedy Triplets
(2) **Input:** $\Delta_{\min}$ and n mutated sequences of length ℓ generated by T.
(3) **Output:** Unrooted tree T^* with edge weights Δ^*.
(4) Compute $\hat{\sigma}$ and $\hat{\Delta}$ using Equations (2.2) and (2.5).
(5) For all leaves X, set $\mathit{Def}(X) \leftarrow \{X\}$.
(6) Find a triplet ABC with the maximum $\hat{\sigma}_{ABC}$.
(7) Let T^* be the star defined by ABC and its center D.
(8) Set the Δ^*_{AD}, Δ^*_{BD}, Δ^*_{CD} values using Equation (3.1).
(9) Set $\mathit{Def}(D) \leftarrow \{A, B, C\}$.
(10) Let $\mathcal{S}$ be the set of all triplets WAB, WAC, WBC with positive pairwise closenesses.
(11) **repeat**
(12)   **for** each $NXY \in \mathcal{S}$ in decreasing order of $\hat{\sigma}_{NXY}$ where $N \notin T^*$ **do begin**
(13)   **for** each edge QQ' on the path between X and Y in T^* **do begin**
(14)   **if** $\{X, Y\} \cap \mathit{Def}(Q) = \emptyset$ or $\{X, Y\} \cap \mathit{Def}(Q') = \emptyset$
(15)   **then** set *Split* $\leftarrow$ "mismatch".
(16)   **else** set *Split* $\leftarrow$ Split-Edge(Q, Q', NXY, T^*).
(17)   **if** *Split* = "too close"
(18)   **then** remove NXY from $\mathcal{S}$.
(19)   **if** *Split* = "OK"
(20)   **then**
(21)   Remove all triplets containing N from $\mathcal{S}$.
(22)   Add to $\mathcal{S}$ all $NN'Z$ with $N' \notin T^*$ and $Z \in T^*$.
(23)   **Exit** both for-loops.
(24)   **end**
(25)   **end**
(26)   **if** T^* was not updated in the for-loops
(27)   **then fail** and **stop**.
(28) Output T^*.

FIG. 3.3. *The* HGT *algorithm. The algorithm maintains a set $\mathcal{S}$ of candidate triplets, which is the set of those triplets that have exactly two nodes in the hypothesis tree T^*. Line (10) initializes the candidate set, while lines* (21) *and (22) update it properly every time a new node is added to T^*. The Split-Edge subroutine in line (16) checks whether NXY can be used to add a new node on edge $QQ' \in T^*$, and if so, updates T^* and returns "OK" in the variable Split. Otherwise, it returns the reason why a new node cannot be inserted. The subroutine is discussed in more details in §3.3.*

$\frac{\Delta_{\min}}{2}$, for $W = X, Y, Z$. We prove using Lemmas 3.1 and 2.2 that if

$$\ell \geq \ell_g = 324 \frac{3 \ln n + \ln \frac{7}{\delta}}{\sigma_{\lg}^2 f^2 \lambda^2},$$

then $\Pr\{E_g\} \geq 1 - \delta/2$. Furthermore, if

$$\ell \geq \ell_c = 210 \alpha^2 \frac{3 \ln n + \ln \frac{3}{\delta}}{\sigma_{\lg}^2},$$

then $\Pr\{E_c\} \geq 1 - \delta/2$. Therefore, if $\ell \geq \max\{\ell_c, \ell_g\}$, then $\Pr\{E_g \text{ and } E_c\} \geq 1 - \delta$. Finally we show that E_g and E_c imply that T^* is built without using small triplets, and that Split-Edge

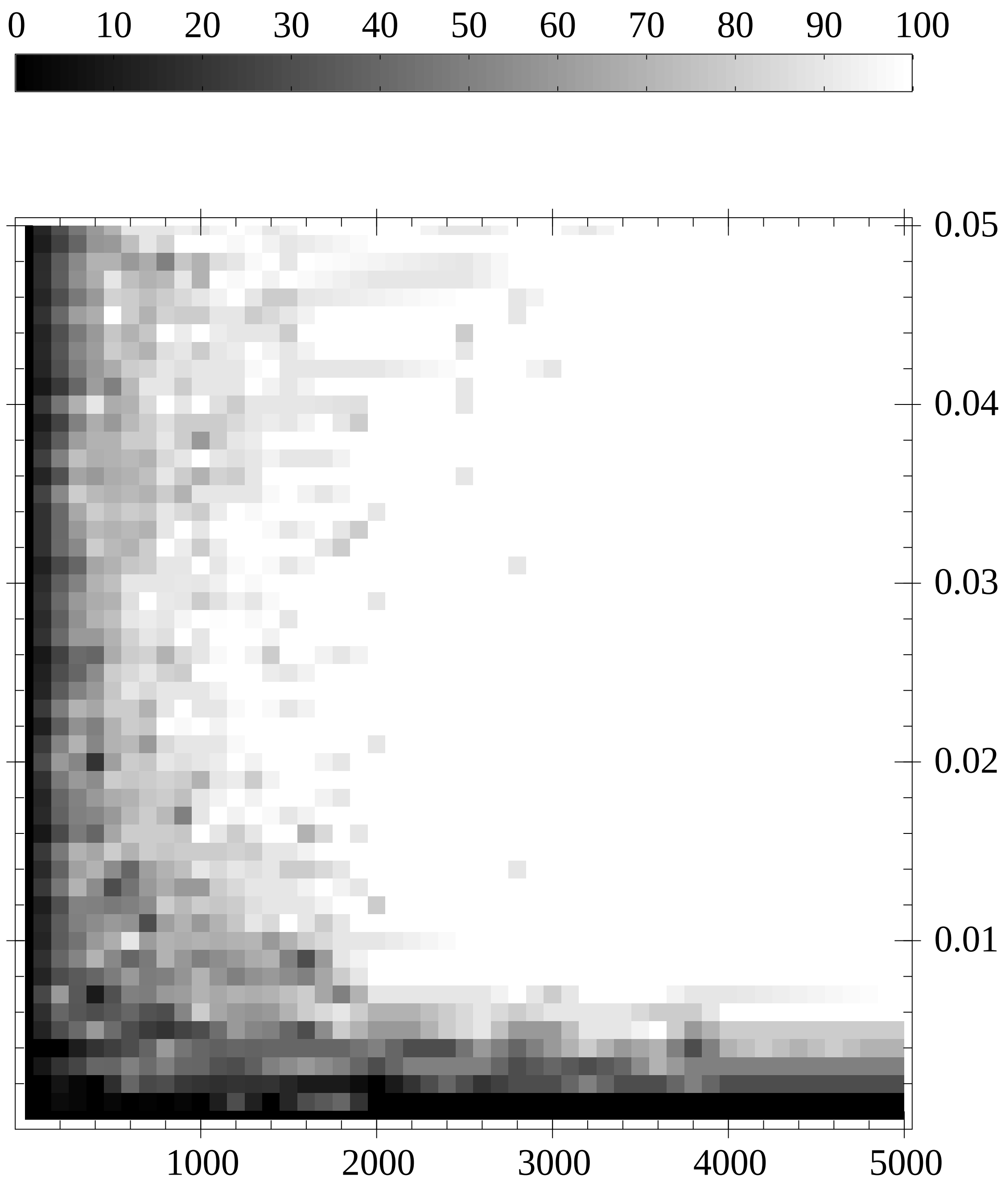

FIG. 4.1. *This picture gives the success rates on caterpillar trees with fifty leaves. The horizontal axis shows the sample sequence lengths with the vertical axis showing the mutation probabilities. Shading shows the relative success rate from among ten experiments.*

recognizes identical nodes while estimating edge lengths within $\pm 2\Delta_{\min}$ error. □

4. Experimental results. Erdős *et al.* [6] conducted experiments to evaluate the SQM algorithm by simulating binary sequence evolutions on caterpillar trees with fifty leaves. The caterpillar trees are best suited for demonstrating that the topology of a large evolutionary tree can be recovered from short sequences even if some terminal taxa are far away from each other. Almost all the existing distance-based algorithms take no advantage of the topology of the tree and require long sequences in order to estimate the distances between nodes that are far apart.

The caterpillar trees in both their and our studies have the same mutations probabilities p on all edges except on the two farthest edges from the root, which have a mutation probability $5p$. The study measures the success rate of the algorithm in question at various edge probabilities and sequence lengths. The success rate for each edge weight and sample size is computed as the fraction of runs out of ten replicates, in which the topology is recovered correctly. The experimental comparison of SQM to the Neighbor Joining algorithm and others clearly shows the performance advantage of the SQM algorithm. We have conducted the same experiment for the HGT algorithm; the results are summarized in Figure 4.1. The HGT algorithm has everywhere achieved a success rate at least comparable to that of SQM. The SQM algorithm does not estimate edge weights. The HGT algorithm estimates the edge weights with high accuracy as shown in Figure 4.2, which depicts the distribution of the estimation error

$$\max_{XY \in \{\text{edges of } T^*\}} |\Delta^*_{XY} - \Delta_{XY}|$$

at various mutation probabilities and sample lengths for the caterpillar trees described above. The graphs show that the error quickly decreases with the sample size, and for $\ell \geq 2500$, it is less than about $-0.25 \ln(1-2p)$.

5. Discussion and future research. Intuitively, triplets are the smallest structures that can be used to reconstruct a tree. Nonetheless, triplets alone do not guarantee good theoretical performance as evidenced by the recent proof that the Neighbor Joining algorithm [2] requires sample sequences of exponential lengths. The HGT algorithm achieves its theoretical performance by employing a greedy strategy directly based on our probabilistic analysis of the Jukes-Cantor model of sequence evolution. We have extended the analysis to the more general model of sequence evolution where the mutation along an edge obeys a general transition matrix. We intend to present the results in a different study. It is also of great significance to conduct further experiments on a wide range of biologically more realistic trees.

Acknowledgments. We would like to thank Dana Angluin and Stanley Eisenstat for very useful discussions.

REFERENCES

[1] R. Agarwala, V. Bafna, M. Farach, B. Narayanan, M. Paterson, and M. Thorup, *On the approximability of numerical taxonomy (fitting distances by tree metrics)*, in Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, Atlanta, Georgia, 28–30 Jan. 1996, pp. 365–372.

[2] K. Atteson, *The performance of neighbor-joining algorithms of phylogeny reconstruction*, in Computing and Combinatorics, Third Annual International Conference, Shanghai, China, T. Jiang and D. T. Lee, eds., vol. 1276 of Lecture Notes in Computer Science, Berlin, 1997, Springer-Verlag, pp. 101–110.

[3] J. Cavender, *Taxonomy with confidence*, Mathematical Biosciences, 40 (1978), pp. 271–280.

[4] W. H. E. Day, *Computational complexity of inferring phylogenies from dissimilarity matrices*, Bulletin of Mathematical Biology, 49 (1987), pp. 461–467.

[5] W. H. E. Day, D. S. Johnson, and D. Sankoff, *The computational complexity of inferring rooted phylogenies by parsimony*, Mathematical Biosciences, 81 (1986), pp. 33–42.

[6] P. Erdős, K. Rice, M. A. Steel, L. A. Székely, and T. Warnow, *The Short Quartet Method*, Mathematical Modeling and Scientific Computing, (1998). to appear.

[7] M. Farach and S. Kannan, *Efficient algorithms for inverting evolution*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, 22–24 May 1996, pp. 230–236.

[8] J. Felsenstein, *Cases in which parsimony or compatibility methods will be positively misleading*, Systematic Zoology, 22 (1978), pp. 240–249.

[9] ———, *Numerical methods for inferring evolutionary trees*, The Quarterly Review of Biology, 57 (1982), pp. 379–404.

[10] Green Plant Phylogeny Research Coordination Group. `http://ucjeps.berkeley.edu/bryolab/greenplantpage.html`.

[11] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, Cambridge, UK, 1997.

[12] D. M. Hillis, *Molecular versus morphological approaches to systematics*, Annual Review of Ecology and Systematics, 18 (1987), pp. 23–42.

[13] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, Journal of the American Statistical Association, 58 (1963), pp. 13–30.

[14] T. H. Jukes and C. R. Cantor, *Evolution of protein molecules*, in Mammalian Protein Metabolism, H. N. Munro, ed., vol. III, Academic Press, New York, 1969, ch. 24, pp. 21–132.

[15] M. Noro, R. Masuda, I. A. Dubrovo, M. C. Yoshida, and M. Kato, *Molecular phylogenetic inference of the Woolly Mammoth* mammuthus primigenius*, based on complete sequences of mitochondrial cytochrome* b *and 12S ribosomal RNA genes*, Journal of Molecular Evolution, 46 (1998), pp. 314–326.

[16] C.-Y. Ou, C. A. Cieselski, G. Myers, C. I. Bandea, C.-C. Luo, B. T. M. Korber, J. I. Mullins, G. Schochetman, R. L. Berkelman, A. N. Economou, J. J. Witte, L. J. Furman, G. A. Satten, K. A. MacInnes, J. W. Curran, and H. W. Jaffe, *Molecular epidemiology of HIV transmission in a dental practice*, Science, 256 (1992), pp. 1165–1171.

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

[18] M. Steel, *The complexity of reconstructing trees from qualitative characters and subtrees*, Journal of Classification, 9 (1992), pp. 91–116.

[19] D. L. Swofford, G. J. Olsen, P. J. Waddell, and D. M. Hillis, *Phylogenetic inference*, in Molecular Systematics, D. M. Hillis, C. Moritz, and B. K. Mable, eds., Sinauer Associates, Inc., Sunderland, Ma, 2nd ed., 1996, ch. 11, pp. 407–514.

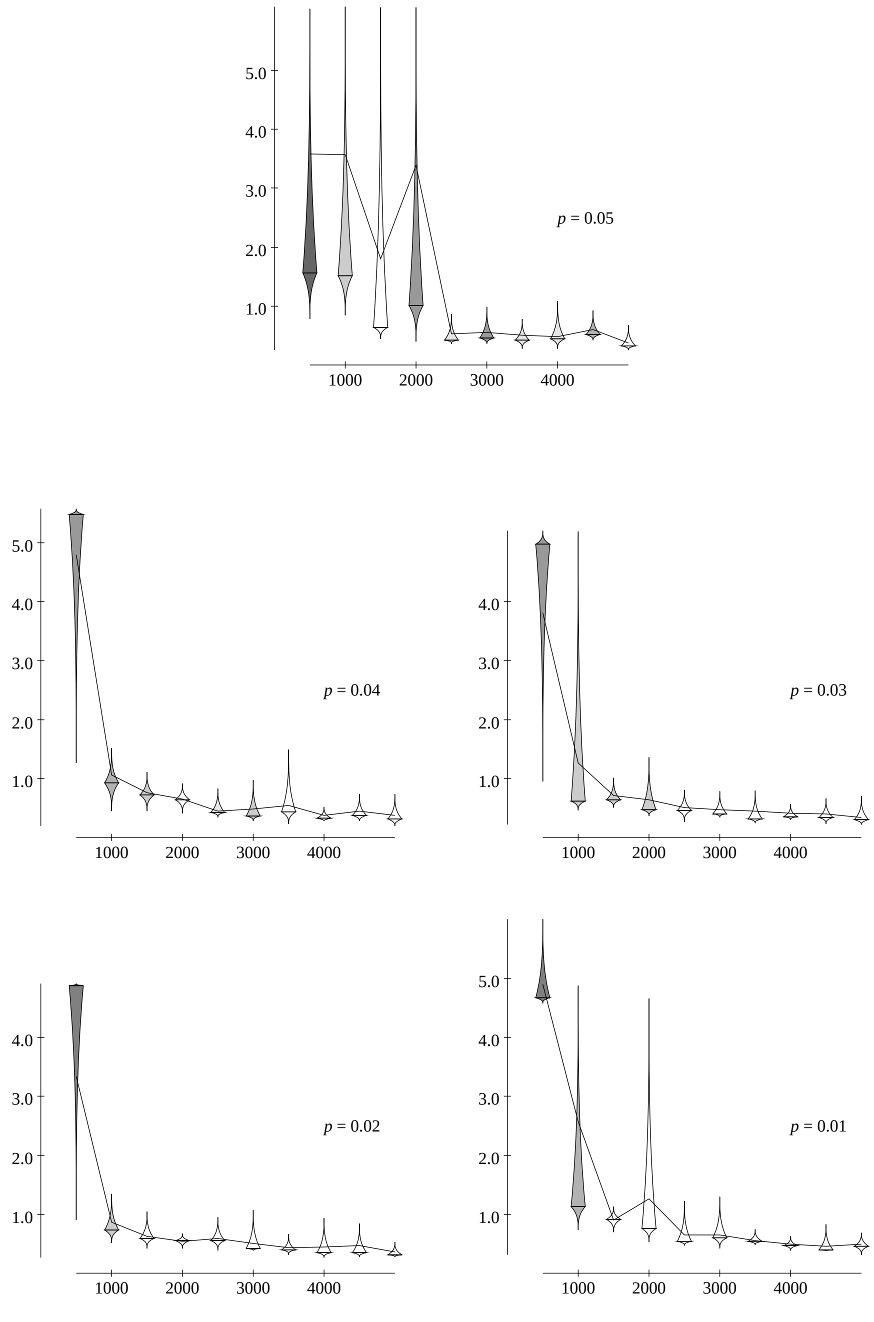

FIG. 4.2. *Edge length estimation errors. The five graphs depict the distribution of the edge length estimation errors for fifty-taxon caterpillar trees with different mutation probabilities* p. *In each graph, the horizontal axis shows the sample length, and the vertical axis shows the error defined as the maximum of* $|\Delta^*_{XY} - \Delta_{XY}|$ *over the edges* XY *of* T^*, *divided by* $-\ln(1-2p)$. *Each diamond-like plot mark displays the distribution information derived from ten runs of the simulation for the given sample length. The maximum error is shown by the top and the minimum error is shown by the bottom of the mark. The widest part is at the median. The shade indicates the success rate, white meaning that the correct topology was recovered ten out of ten times. Darker shades indicate lower success rates as in Figure 4.1. The line plots go through the average errors.*