AN ALGORITHM FOR STATISTICAL ALIGNMENT OF SEQUENCES RELATED BY A BINARY TREE

JOTUN HEIN
Department of Ecology and Genetics,
University of Aarhus, Ny Munkegade, Aarhus, Denmark
jotun.hein@biology.au.dk

Keywords– Sequence evolution, insertion-deletion model, dynamic programming

Abstract

An algorithm is presented that allows the calculation of the probability of a set of sequences related by a binary tree that has evolved according to the Thorne-Kishino-Felsenstein model (1991) for a fixed set of parameters. There are two ideas underlying this algorithm. Firstly, a markov chain is defined that generates ancestral sequences and their alignment at two neighboring nodes in a tree. Secondly, a stochastic walk on the binary tree, that defines a markov chain generating ancestral sequences and their alignment at the internal nodes in the tree is described. The running time of this algorithm is $O(l^{2k})$, where l is the geometric average of the sequence lengths and k the number of sequences - leaves at the binary tree. This could be improved to $O(l^k)$.

Introduction

Proteins and DNA sequences evolve predominantly by substitutions, insertions and deletions of single characters (amino acids/nucleotides) or strings of these elements. During the last two decades, the analysis of the substitution process has improved considerably, and has increasingly been based on stochastic models. The process of insertions and deletions has not received the same attention and is presently being analysed by optimisation techniques (minimizing distance (parsimony) or maximising a similarity score).

A pioneering paper by Thorne, Kishino and Felsenstein[1] proposed a well defined time reversible markov model for insertion and deletions (denoted more briefly as the $\mathcal{TKF}$-model) that allowed a proper statistical analysis for two sequences. Such an analysis can be used to provide maximum likelihood (pairwise) sequence alignments, or to estimate the evolutionary distance between two sequences. Recently, an algorithm was presented by Steel and Hein[2] that allowed statistical alignment of sequences related by a star shaped tree - a tree with one internal node. Here a recursion is presented that leads to a polynomial-time algorithm for calculating the probability of k sequences that evolve on a binary tree according to the $\mathcal{TKF}$-model.

An alternative approach to statistical alignment has been taken by Allison and Wallace[3], Zhu, Liu and Lawrence,C.E.[4], and Mitchison[5]. These are

Bayesian or HMM approaches and also differ from the $\mathcal{TKF}$-model in not being based on an evolutionary process, but on a probability measure on alignments directly.

The $\mathcal{T}KF$-model used three classes of variables[1]. Firstly, to each character was associated a reversible substitution process (identical to the usual site substitution models that did not allow insertions or deletions). Secondly, to each character is associated a deletion stochastic variable, D_i, that is exponentially distributed with parameter μ. If this D_i fires, character i is removed. Thirdly, to the right of every character an insertion stochastic variable, called a mortal link, I_i, was associated. It is exponentially distributed with parameter, λ. If I_i fires a character is chosen in the stationary distribution of the substitution process. If the i–th character dies, I_i dies with it. To the left of the complete sequence there is an immortal link that can give birth to characters (with associated mortal links), at the same rate as mortal links, but cannot die. This prevents the empty sequence from becoming a sink.

We will let t_e denote the (scaled) time parameter that the markov process operates on the edge of the tree incident with edge e. One possible scaling is to measure time, so one unit corresponds to one expected event in the substitution process per position. Since each edge will have different time lengths, it will also have different time parameters and they are denoted by $\lambda_e = t_e\lambda$ and $\mu_e = t_e\mu$, where e refers to the e–th edge. If it is clear from the context which edge is in question, the subscript e is suppressed in the following.

The probability that a sequence at the equilibrium distribution is l characters long is $(1 - \frac{\lambda}{\mu})(\frac{\lambda}{\mu})^l$. The probability of a specific sequence of length l is $(1 - \frac{\lambda}{\mu})(\frac{\lambda}{\mu})^l \prod_{i=1,l} \pi_{s[i]}$ where $s[i]$ is the i–th character of the sequence and $\pi_{s[i]}$ the probability of this character in the equilibrium distribution of the substitutional process.

Let[1] $p_k(t)$ (resp. $p'_k(t)$) denote the probability that after duration t, a mortal link has exactly k descendants, and the first of these is (resp. is not) the original mortal link. Let $p''_k(t)$ denote the probability that after duration t an immortal link has exactly k descendants (including itself).

Let the immortal link be labelled with a * and the mortal link and associated character by a # . We have[1]: For $k > 0$

$$p_k(t) = p\left(\begin{smallmatrix} \# & - & - & - & - \\ \# & \# & \# & \# & \# \\ 1 & 2 & & & k \end{smallmatrix}\right)(t) = e^{-\mu t}[1 - \lambda\beta(t)][\lambda\beta(t)]^{k-1};$$

$$p'_k(t) = p\left(\begin{smallmatrix} \# & - & - & - & - & - \\ - & \# & \# & \# & \# & \# \\ & 1 & 2 & & & k \end{smallmatrix}\right)(t) = [1 - e^{-\mu t} - \mu\beta(t)][1 - \lambda\beta(t)][\lambda\beta(t)]^{k-1};$$

$$p'_0(t) = p\left(\begin{smallmatrix} \# \\ - \end{smallmatrix}\right)(t) = \mu\beta(t),$$

and

$$p''_k(t) = p\left(\begin{smallmatrix} * & - & - & - & - & - \\ * & \# & \# & \# & \# & \# \\ & 1 & 2 & & & k \end{smallmatrix}\right)(t) = [1-\lambda\beta(t)][\lambda\beta(t)]^k;$$

where $\beta(t) = \frac{1-e^{(\lambda-\mu)t}}{\mu-\lambda e^{(\lambda-\mu)t}}$. In the following β is short for $\beta(t)$ and β_e is short for $\beta(t_e)$.

To exemplify these probabilities we have that

$$p\left(\begin{smallmatrix} C & - & - & - & - & - & - \\ P & T & D & I & G & S & L \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{smallmatrix}\right)(t) = \pi_C \cdot p_7(t) \cdot P_{C\to P} \cdot \pi_T \cdot .. \cdot \pi_L,$$

since the expression on the right hand side is the probability of selecting a C, times the probability that a character survives itself and has 6 additional children, times the probability of the substitutional evolution that a C mutates to a P, and lastly the probabilities that the new children are $T, D, \ldots, L$.

Any pairwise alignment can be represented by a sequence of the above three alignment building blocks having probabilities $p_k(t)$, $p'_k(t)$ and $p''_k(t)$, respectively. Each alignment building block describes fate of each character in the upper (parent) sequence into the lower (child) sequence. The decomposition of alignments into these building blocks and their easily obtainable probabilities were essential in the formulation of a dynamical programming algorithm that calculates the probability of observing two complete sequences. The probability of an alignment of two complete sequences is the product of the probability of the length of the upper sequence, l_1, $((1-\frac{\lambda}{\mu})(\frac{\lambda}{\mu})^{l_1})$, multiplied by the probabilities of all its alignment blocks and the probabilities of the observed substitutional molecular evolution (matched characters).

An alignment block starting with two aligned immortals (those with probability $p''_k(t)$) can be generated by a markov chain that starts in $\begin{smallmatrix} * \\ * \end{smallmatrix}$ $\left(\begin{smallmatrix} parent \\ child \end{smallmatrix}\right)$ (stops in E) and has the following transition probabilities:

	$\begin{smallmatrix} parent & - \\ child & \# \end{smallmatrix}$	E
$\begin{smallmatrix} parent & * \\ child & * \end{smallmatrix}$	$\lambda\beta$	1 - $\lambda\beta$
$\begin{smallmatrix} parent & - \\ child & \# \end{smallmatrix}$	$\lambda\beta$	1 - $\lambda\beta$

Blocks tracing the fate of a mortal link in the parent sequence down in the child sequence (those with probabilities $p_k(t)$ and $p'_k(t)$) can be generated by a chain that starts in $\begin{smallmatrix} \# \\ - \end{smallmatrix}$ with probability $(1-e^{-\mu})$ and in $\begin{smallmatrix} \# \\ \# \end{smallmatrix}$ with probability $e^{-\mu}$, and then proceeds according to the following transition probabilities:

	$\begin{matrix} parent & - \\ child & \# \end{matrix}$	E
$\begin{matrix} parent & \# \\ child & \# \end{matrix}$	$\lambda\beta$	1 - $\lambda\beta$
$\begin{matrix} parent & - \\ child & \# \end{matrix}$	$\lambda\beta$	1 - $\lambda\beta$
$\begin{matrix} parent & \# \\ child & - \end{matrix}$	$(1-\mu\beta-e^{-\mu})/(1-e^{-\mu})$	$(\mu\beta)/(1-e^{-\mu})$

Generating Ancestral Sequences and Their Alignments.

The basic idea in constructing recursions allowing the calculation of the likelihood of the observed sequences on the leaves is to construct a markov chain that generates ancestral sequences (without their actual characters) and their alignment with a probability that corresponds to the $\mathcal{TKF}$ process. Conditioned on the realisation of this process it is easy to calculate the probability of the observed sequences.

Thorne (pers. comm.) has together with Gary Churchill in another context used a similar markov chain in unpublished work generating sequence pairs.

The markov chain is constructed by combining the alignment block markov chains with a parent sequence generating markov chain. A markov chain generating one ancestral sequence can be made using three states - start (*) and stop (E) state plus a (#) state that jumps to itself with probability (λ/μ) giving the correct geometric distribution. The procedure is most easily described by an example, and we consider a binary tree with 4 leaves. Due to the time reversibility of the $\mathcal{TKF}$ process, likelihood value is independent of the placement of the root and the binary tree with four leaves can be arbitrarily rooted. We choose to root it at the ancestral sequence, a1 (see figure 1).

In the $\mathcal{TKF}$ process, two ancestral sequences will have two matched immortal links and these matched immortal links will be the starting state of the markov chain. A markov chain can be constructed that combines the probability of an additional character in the ancestral sequence and if such a character is chosen uses the markov chain generating alignment blocks to create such a block, before choosing yet another character in the ancestral sequence (see Table 1). There are five states in the markov chain. Two special states that starts $\left(\begin{smallmatrix} * \\ * \end{smallmatrix}\right)$ and thus ends $\left(\begin{smallmatrix} E \\ E \end{smallmatrix}\right)$ the markov chain and three states that corresponds to the possible configurations in a column in the alignment of two

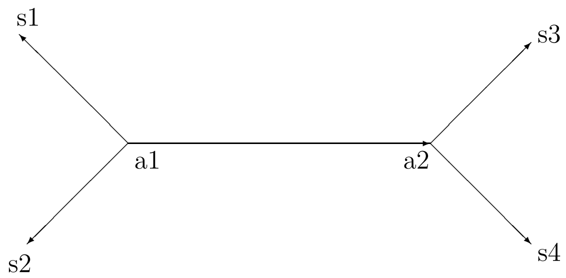


Figure 1: Tree arbitrarily rooted at a1 with 2 internal nodes and 4 leaves

	a1: - a2: #	a1: # a2: #	a1: # a2: -	a1: E a2: E
$a1$ $*$ $a2$ $*$	$\lambda\beta$	$\lambda/\mu(1-\lambda\beta)e^{-\mu}$	$\lambda/\mu(1-\lambda\beta)(1-e^{-\mu})$	$(1-\lambda/\mu)(1-\lambda\beta)$
$a1$ $-$ $a2$ $\#$	$\lambda\beta$	$\lambda/\mu(1-\lambda\beta)e^{-\mu}$	$\lambda/\mu(1-\lambda\beta)(1-e^{-\mu})$	$(1-\lambda/\mu)(1-\lambda\beta)$
$a1$ $\#$ $a2$ $\#$	$\lambda\beta$	$\lambda/\mu(1-\lambda\beta)e^{-\mu}$	$\lambda/\mu(1-\lambda\beta)(1-e^{-\mu})$	$(1-\lambda/\mu)(1-\lambda\beta)$
$a1$ $\#$ $a2$ $-$	$\frac{(1-e^{-\mu}-\mu\beta)}{1-e^{-\mu}}$	$\frac{\lambda\beta e^{-\mu}}{1-e^{-\mu}}$	$\lambda\beta$	$\frac{\beta(\mu-\lambda)}{1-e^{-\mu}}$

Table 1: Table of transition probabilities among states.

sequences $\begin{pmatrix} \# & \# & - \\ \# & - & \# \end{pmatrix}$. Assume the ancestral sequences and their alignment in Table 2 has been observed. What is the probability of for instance the first transition?

The realisation of the process starts in $\begin{pmatrix} * \\ * \end{pmatrix}$, then jumps to $\begin{pmatrix} \# \\ \# \end{pmatrix}$. This involves 3 independent events. First, that the immortal link in a1 had no children (probability $(1-\lambda\beta)$). Then that the ancestral sequence at a1 is elongated by one character (probability λ/μ) and lastly that this character has not been deleted in a2 (probability $e^{-\mu}$). This gives a combined probability of $\lambda/\mu(1-\lambda\beta)e^{-\mu}$. This process is continued until the end state $\begin{pmatrix} E \\ E \end{pmatrix}$ is reached.

a1	*	#	-	#	#	E
a2	*	#	#	-	#	E
		$\lambda/\mu(1-\lambda\beta)e^{-\mu}$	$\lambda\beta$	$\lambda/\mu(1-\lambda\beta)(1-e^{-\mu})$	$\frac{\lambda\beta e^{-\mu}}{1-e^{-\mu}}$	$(1-\lambda/\mu)(1-\lambda\beta)$

Table 2: Example of ancestral sequences and their alignment

The most complicated transition probabilities start at the state $\begin{pmatrix} \# \\ - \end{pmatrix}$. Going to $\begin{pmatrix} - \\ \# \end{pmatrix}$ will choose within the case with no survival of the character (probability $1 - e^{-\mu}$), if the character goes extinct (no children and not surviving - unconditional probability $\mu\beta$). Chosing among the possibility of going to [$\begin{pmatrix} \# \\ \# \end{pmatrix}$ and $\begin{pmatrix} \# \\ - \end{pmatrix}$] or E amounts to deciding between elongation of the ancestral sequence $(\lambda\beta)$ or not elongating $(1 - \lambda\beta)$. Choosing among $\begin{pmatrix} \# \\ \# \end{pmatrix}$ or $\begin{pmatrix} \# \\ - \end{pmatrix}$ again is to choose between survival and non-survival of a character.

Dynamical Programming Recursion for 4 sequences.

Let $P_\epsilon(\mathbf{s_i})$ be the probability of $s1[1,\ldots,i_1],\ldots,s4[1,\ldots,i_4]$ given that the ancestral sequences were generated by a markov chain (Table 1) ending in markov chain state ϵ. Let $\mathbf{H}$ be a 0,1 vector with four elements that for every leaf indicates if the character there has survived undeleted (1) from its parent character or if it has been inserted (0). Let l_j be the length of the j–th sequence. Let $^{\mathbf{i}}\mathbf{s}$ refer to $s1[i_1+1,\ldots,l_1],\ldots,s4[i_4+1,\ldots,l_4]$ and $^{\mathbf{i}}\mathbf{s_k}$ be $s1[i_1+1,\ldots,k_1],\ldots,s4[i_4+1,\ldots,k_4]$. Let $P(\alpha \to \epsilon)$ denote the probability of the transition from α to ϵ according to Table 1. Let $P(^{\mathbf{i}}\mathbf{s_k}, \mathbf{H} \mid \epsilon)$ be the probability that the subsequences $^{\mathbf{i}}\mathbf{s_k}$ have evolved from the configuration of characters at ancestral nodes corresponding to ϵ and that character $s_j[i_j+1]$ is a survived character if $H(j) = 1$ and otherwise it is an inserted character.

The probability of the complete sequences can then be calculated through following recursions:

$$P_\epsilon(s_k) = \sum_\alpha \sum_{i \in S_\epsilon} \sum_{H \in C_\epsilon} P_\alpha(\mathbf{s_i})\mathbf{P}(\alpha \to \epsilon)\mathbf{P}(^{\mathbf{i}}\mathbf{s_k}, \mathbf{H} \mid \epsilon). \tag{1}$$

S_ϵ are all prefixes, $\mathbf{s_i}$, of $\mathbf{s_k}$, where $\mathbf{i(j)} = \mathbf{k(j)}$ if $\epsilon(parent(j)) ='-'$. Parent refers to the parent node on the tree, which is the argument to ϵ. This condition prevents a substring from being nonempty, but having no parent character that has generated it. C_ϵ is the set of 4-dimensional vectors over $\{0,1\}$, obeying the restriction $H(j) = 0$ if $\epsilon(parent(j)) ='-'$. The restriction guarantees that H and ϵ define a connected set on T.

In recursion (1) two points should be noted. First, $P_{*,*}(s_k)$, can be calculated directly, serving as initialisation of the recursion. Second, $P_\epsilon(s_k)$, is present on both sides of the equation, but the coefficients in front of them on the left side can be calculated easily, allowing a proper recursion to be formulated.

The only quantity on the left side of equation (1) that is difficult to calculate is $P(\mathbf{{}^{i}s_k}, \mathbf{H} \mid \epsilon)$. The exact value of $P(\mathbf{{}^{i}s_k}, \mathbf{H} \mid \epsilon)$ is

$$F(H, \mathbf{{}^{i}s}) \left(\prod_{\mathbf{j:H(j)=0}} \mathbf{p'_{k(j)-i(j)+1}(t_j)} \pi_{\mathbf{s_j[i(j):k(j)]}} \right) \left(\prod_{\mathbf{j:H(j)=1}} \mathbf{p_{k(j)-i(j)+1}(t_j)} \pi_{\mathbf{s_j[i(j)+1:k(j)]}} \right)$$

$F(H, \mathbf{{}^{i}s})$ is a function that evaluates the probability of observing the characters at the leaves j, where $H(j) = 1$ using Felsenstein's algorithm[6]. F can also perform the same evaluation directly on a tuple of characters and gap signs. The above is then the product of

1. the probability of the observed substitutional evolution (the F-function).
2. the probability of the strings that have been created using alignment blocks like $\left(\begin{smallmatrix} \# & - & - & - & - \\ - & \# & \# & \# & \# \end{smallmatrix} \right)$. This probability also includes the probability of picking the inserted characters.
3. the probability of the strings that have been created using alignment blocks like $\left(\begin{smallmatrix} \# & - & - & - & - \\ \# & \# & \# & \# & \# \end{smallmatrix} \right)$. This probability also includes the probability of picking the inserted characters. This does not include the probability of the first character in the corresponding substring as this has evolved from its parent and has not been inserted.

Following a simple example, based on the four sequences: GCTAC, AATTAG, CGGAG and CCTG.

s1	G	C	T	A			-	C
s2	A	A	T	T			A	G
a1							#	-
					α	$\longrightarrow$	ϵ	
a2							#	-
s3	C	G	G				A	G
s4	C	C	T	G			-	-

In this case $H = (0, 1, 1, 0)$ and $\epsilon =$(#,#). Assume $\alpha =$(#,-). The first columns including α signifies the probability of those sequences given their ancestral sequences were generated by a markov chain ending in α and no assertion is otherwise made about their alignment. Then comes the transition from α to ϵ. The columns after ϵ signify the remaining part of the sequences given that they have been derived from two ancestral characters and that these ancestral characters survived in s2 and s3. The contribution of this specific configuration to the whole sum,

$$P_{\#,\#}(GCTAC, AATTAG, CGGAG, CCTG),$$

would be the product of following three factors:

$$P_{\#,-}(GCTA, AATT, CGG, CCTG)$$

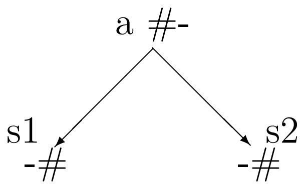
$$P(\#- \rightarrow \#\#) = \frac{e^{-\mu}\lambda\beta}{1-e^{-\mu}}$$

$$p_1'(t_1) \cdot \pi_C \cdot p_2(t_2) \cdot \pi_G \cdot p_2(t_3) \cdot \pi_G \cdot p_0'(t_4) \cdot F(-, A, A, -)$$

The first factor is known because it has been calculated recursively. This algorithm allows an $\mathrm{O}(l^8)$ algorithm to be formulated. The recursions have to be applied to all prefixes and possible ending states, (ϵ, s_k). This accounts for a factor $constant \cdot l_1 \cdot, \ldots, \cdot l_4$. For each fixed (ϵ, s_k), the right side of the recursion has to be calculated. The summing over $(\alpha \rightarrow \epsilon)$ and H is a constant factor and $P_\epsilon({}^i s_k)$ is constant, since it is tabulated. The additional growth in complexity is due to the summation over all possible prefixes (of the prefixes) that have evolved from the markov chain ending in α and the remaining suffixes that have evolved from the ϵ assignment to internal nodes.

Recursion for arbitrary number of sequences.

Extending the algorithm for four sequences to an arbitrary number of sequences related by known tree requires some additional concepts. A markov chain is used to generate the ancestral sequences and their alignment in proportion to the $\mathcal{TKF}$ process. Two points should be noted about such alignments. Firstly, the columns with characters will constitute a connected set on the tree, relating the sequences and this will therefore itself be a tree. The leaves of this tree only relating the ancestral sequences will be called internal leaves. Secondly, the $\mathcal{TKF}$ process does not itself unambigously define the order of all characters in the alignment. The simplest illustration of this is



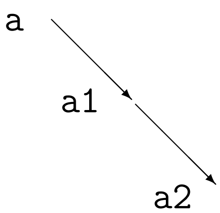
In this case the character at the root did not survive itself, but left one descendant. This happened in both the left branch and the right branch. How are they to be aligned? Obviously the first column must be (s1,a,s2) = (-,#,-), but which of the characters at the leaves should come first? The evolutionary history does not specify this, so the alignments

s1	-	#	-		-	-	#
a	#	-	-	or	#	-	-
s2	-	-	#		-	#	-

are evolutionary equivalent.

A further complication relative to the simple case is that the whole run of inserted characters at a1 cannot be created immediately. The children of the first character must be created before the children of the second character etc. all the way down the tree.

Here is an example with 3 ancestral sequences:




```
# - - - - - - - a

# # - - - # # # a1

- # # # # - - - a2
```

In this case the character (at a) has five descendants (at a1) and survived itself. The second character (of the five descendants at a1) has four descendants (at a2) including itself. These four grandchildren must be generated before characters 3-5 at a1. One way to handle this is to create the children of a character before creating its siblings to the right.

With these points in mind it is possible to generalise the markov chain generating ancestral sequences on neighboring nodes to the set of ancestral sequences at all internal nodes.

Fix one node as the root and then traverse the tree depth-first and for instance left-child before right-child. Such an order of traversal will pick out one alignment among the evolutionary equivalent. In the example above, the first alignment is chosen because (a, s1) will be visited before (a, s2).

This process can be described as a stochastic walk on the tree relating the ancestral sequences. Each time an edge is traversed a process analogous to the process relating the two ancestral sequences in the 4-sequences with 2 ancestral sequences case is performed.

Following procedure written as pseudocode will perform a random walk on the tree and ancestors and their alignments in accordance with the distribution dictated by the $\mathcal{TKF}$ process and then select among evolutionary equivalent alignments according to the depth-first, order of children traversal dictated by the traversal order of this tree.

Let ANCESTORS be a 2-dimensional array, where each entry has either *, - or # at each entry. It has dimensions $[ancestors][0;\infty]$. Prefill this with $-$. The pseudocode will now place "*" and "#" signs according to the $\mathcal{TKF}$ process.

In the pseudocode, the function *random* will choose a real number uniformly betweeen 0.0 and 1.0. The variable of *type* is necessary to handle the immortal links in the begining of the alignment - where the treewalk has to continue all the way to the internal leaves, since they cannot be deleted.

Variables used:
  parent, child, enfant : node in tree;
  type: immortal (*), mortal (#)
  column: integer initialized to 0.

siblings(child,type) (processes a new character)
{ ANCESTORS[child][column]=type;
  for (enfant member of child's children) treewalk(child, enfant, type);
  column++;
}

treewalk(parent, child, type) (processes a character down an edge)
{ if ($type * random < e^{-\mu}$) {

```
    ANCESTORS[child][column]=type;
    for (enfant member of child's children) treewalk(child, enfant, type);
    while (random < λβ) siblings(child,type);
  }
  else if (random > μβ/(1 − e^{−μ})) {
    siblings(child,type);
    while (random < λβ) siblings(child,type);
    }
  }
}
```

The procedure must be called at the begining and each time the ancestral sequence at the root is chosen to be elongated by one character. This can be done as follows:

```
siblings(root, immortal)
while (random < λ/μ) siblings(root,mortal);
```

When no more elongations are chosen at the edges and the ancestral sequence at the root is completed, the whole process is terminated.

The steps of this stochastic process have to be grouped, so each group of consecutive steps creates one column in the alignment. This is done by defining a new group each time the last step finished tracking a character and its ramifications down the tree. The state of the tree walking process can at that point be summarized by labelling all the nodes where the last character didn't survive and so far had no additional children with a "-". Due to the nature of the process, this will define a subtree hanging from the root and terminating in internal laves or in nodes labelled "-". Nodes above these are labelled "#". The set of such node labellings is the state space of the markov chain. The transition probability from state to the next is the product of the probabilities of three sets of events.

First, a set of edges where it is chosen that the parent could not have additional children.

Second, a new character is created by either elongating the ancestral sequence or deciding that the parent at an edge should have an additional child.

Third, the fate of the new character is traced down the tree. The probability of such a transition will be the product of $\prod e^{-\mu_e}$ and $\prod (1 - e^{-\mu_e})$ for the edges with survival and no survival, respectively.

Using this markov chain the central recursion for 4 sequences (eqn. 1) can now be used for k sequences related by a binary tree. The size of the set of states in the markov chain is less than 2^k, due to the restriction that the positions in an alignment column with # must be a connnected set, so # and - cannot be chosen freely.

Discussion

The algorithm presented here is slow. However, an $O(l^k)$ algorithm is possible by using the geometric tails of the p-functions to make an algorithm, that only adds 0 or 1 character of the leaf sequences at a time. Doing this in the general case is slightly more technical and will not be pursued here.

The present method could possibly be made to work for four to seven sequences by different methods of corner cutting allowing the amount of computation to be reduced. For larger number of sequences it seems that markov chain monte carlo (MCMC) methods would be the obvious choice.

Acknowledgements. I am grateful to Anne-Mette Krabbe, Bjarne Knudsen, Carsten Wiuf, Jens Ledet Jensen, Christian Storm and Mike Steel for critical comments on the manuscript.

References

1. J.L. Thorne, H. Kishino and J. Felsenstein, An evolutionary model for maximum likelihood alignment of DNA sequences, *J. Mol. Evol.* **33**, 114-124 (1991).

2. Steel, M. and J.Hein Applying the Thorne-Kishino-Felsenstein model to sequence evolution of a star tree, *Appl. Math. Lett.* (submitted) (2000).

3. Allison,L. and Wallace, C.S., The posterior probability distribution of alignments and its application to parameter estimation of evolutionary trees and to optimisation of multiple alignments, *J. Mol. Evol.* **39**, 418-430 (1994).

4. Zhu,J.,Liu,J.S. and Lawrence,C.E. Baysian adaptive sequence alignment algorithms, *Bioinformatics* **14**, 25-39 (1998).

5. G. Mitchison, A probabilistic treatment of phylogeny and sequence alignment, *J. Mol. Evol.* **49(1)**, 11-22 (1999).

6. Felsenstein, J. Evolutionary trees from DNA sequences: a maximum likelihood approach, *J.Mol.Evol.* **17**, 368-76 (1981).