ABSTRACT

Motivation: We review proposed syntheses of probabilistic sequence alignment, profiling and phylogeny. We develop a multiple alignment algorithm for Bayesian inference in the links model proposed by Thorne et al. (1991, J. Mol. Evol., 33, 114–124). The algorithm, described in detail in Section 3, samples from and/or maximizes the posterior distribution over multiple alignments for any number of DNA or protein sequences, conditioned on a phylogenetic tree. The individual sampling and maximization steps of the algorithm require no more computational resources than pairwise alignment.

Methods: We present a software implementation (Handel) of our algorithm and report test results on (i) simulated data sets and (ii) the structurally informed protein alignments of BAiiBASE (Thompson et al., 1999, Nucleic Acids Res., 27, 2682–2690).

Results: We find that the mean sum-of-pairs score (a measure of residue-pair correspondence) for the BAiiBASE alignments is only 13% lower for Handel than for CLUSTALW (Thompson et al., 1994, Nucleic Acids Res., 22, 4673–4680), despite the relative simplicity of the links model (CLUSTALW uses affine gap scores and increased penalties for indels in hydrophobic regions). With reference to these benchmarks, we discuss potential improvements to the links model and implications for Bayesian multiple alignment and phylogenetic profiling.


Contact: ihh@fruitfly.org

1 INTRODUCTION

Most sequence profiling tools—including SAM (Karplus et al., 1998), HMMER (Eddy, 1996) and PSI-BLAST (Altschul et al., 1997), to name a few—use sequence weighting to correct for phylogenetic bias in the training set. Although this is a quick and effective correction for over-representation, it is a shortcut compared to a full phylogenetic model and, being a shortcut, it may miss potentially important clues about the selection pressures acting on the sequence family. The phylogenetic context of mutation events is significant in molecular evolution, as mutations on short branches may indicate variations in selection pressure, while an absence of mutation on short branches conveys virtually no information (Figure 1).

As an improvement on weighted training, one may consider evolutionary models of biological sequences (Thorne et al., 1991, 1992; Bishop and Thompson, 1986; Mitchison and Durbin, 1995; Durbin et al., 1998; Mitchison, 1999a; Hein et al., 2000; Hein, 2001). In contrast to static models such as profile hidden Markov models (HMMs), which give an instantaneous probability distribution over sequences (and thus effectively treat every sequence in the family as an independent realization of the HMM (Krogh et al., 1994)), evolutionary models give a joint distribution for all the sequences in a family at once, conditioned on the tree that relates them. Thus correlations between related sequences are built into the model (Durbin et al., 1998).
The simplest evolutionary model is a pairwise likelihood for the relationship between two sequences. Call these two sequences $A$ (for ancestor) and $D$ (for descendant). Then the pairwise likelihood could be written as

$$
Pr \left[ A \xrightarrow{\pi} D | t, \Theta \right] \tag{1}
$$

where $\pi$ is a description of the evolutionary relationship (or alignment) between the sequences, $t$ is the evolutionary time separating the sequences and $\Theta$ covers any additional parameters of the model.

To be useful as a phylogenetic likelihood, equation (1) should satisfy reversibility and additivity constraints.

The reversibility constraint may be expressed as

$$
Pr \left[ A \xrightarrow{\bar{\pi}} D | t, \Theta \right] = Pr \left[ D \xrightarrow{\bar{\pi}} A | t, \Theta \right] \tag{2}
$$

where $\bar{\pi}$ is the inversion of the evolutionary relationship $\pi$ (the meaning of this inversion may be model-dependent, but should essentially leave one-to-one residue alignments unchanged). Intuitively, this means that there is no inherent directionality to the branches of the tree; practically, it means that the placement of the root node is irrelevant.

The additivity constraint may be expressed as

$$
\sum_{X, \pi_1, \pi_2 : \pi_1 \circ \pi_2 = \pi} Pr \left[ A \xrightarrow{\pi_1} X_1 | t_1, \Theta \right] Pr \left[ X_1 \xrightarrow{\pi_2} D | t_2, \Theta \right] = Pr \left[ A \xrightarrow{\pi} D | t_1 + t_2, \Theta \right] \tag{3}
$$

where $X$ is a sequence and ‘$\circ$’ is the operator for conjoining two alignments (so that $\pi_1$ and $\pi_2$ are chosen to be compatible subalignments of $\pi$). Equation (3) is a probabilistic statement that the sequence evolves through time through a series of intermediate sequences.

Both additivity and reversibility are intuitive properties of an evolutionary model. In addition, for the purposes of doing multiple alignment (and training and database searching), we would like to be able to infer the alignment $\pi$ given the sequences $A$ and $D$ alone. This can be done without a combinatorial explosion if the Markov condition applies, which is to say that the likelihood equation (1) can be factored by splitting $\pi$ into stepwise segments along the sequence, where the probability of each segment depends only on the alignment of the segments that come immediately before. If this condition applies then standard Dynamic Programming (DP) algorithms can be used to infer $\pi$ (or sum over it, or sample it) with complexity linear in each sequence (see e.g. Durbin et al., 1998).

Given a likelihood of the form equation (1) that satisfies equations (2) and (3), we can write down an expression for the joint likelihood of a multiple alignment of a set of sequences $S$ related by a tree $T$. (The set $S$ includes all sequences at internal nodes of the tree, as well as leaf sequences. In practise we usually observe leaf sequences only; this amounts to a Bayesian ‘missing data’ problem and will be discussed below.) The joint likelihood is found by decomposing the multiple alignment into a set $\{\pi\}$ of pairwise branch alignments

$$
Pr \left[ \{\pi\}, S | T, \Theta \right] = Pr \left[ \text{root} | \Theta \right] \prod_b Pr \left[ A_b \xrightarrow{\pi_b} D_b | t_b, \Theta \right] \tag{4}
$$

where the tree $T$ designates one node as the root and assigns to every other (descendant) node $D_b$ an ancestral node $A_b$, separated from $D_b$ by evolutionary time $t_b$ (the length of branch $b$). The expression $Pr \left[ \text{root} | \Theta \right]$ is the prior probability of a single sequence (in this case, the root node) according with the evolutionary model.

A number of time-dependent probabilistic alignment models have been proposed in the molecular evolutionary literature. It is instructive to review and compare these models. At the time of writing, none of these models have been implemented as a general-purpose multiple alignment or profiling package.

### 1.1 Substitution models

Most of the proposed models deal with substitution only and thus rely on the alignment being pre-specified. Gaps in the alignment are either ignored, or are modeled as an extra kind of residue (a twenty-first amino acid or fifth nucleotide). This is the kind of model used by PSI-BLAST (Altschul et al., 1997).

The most common substitution model is a classical stochastic process: a continuous-time, finite-state Markov chain with one state for each possible residue (a twenty-first amino acid or fifth nucleotide). This is the kind of model used by PSI-BLAST (Altschul et al., 1997).

The most common substitution model is a classical stochastic process: a continuous-time, finite-state Markov chain with one state for each possible residue (a twenty-first amino acid or fifth nucleotide). This is the kind of model used by PSI-BLAST (Altschul et al., 1997).

### 1.2 The links model

A more gap-conscious model was proposed by Thorne et al. (1991, 1992) and has recently been further analysed (Hein et al., 2000; Hein, 2001). Their ‘links’ model is a birth-death process with immigration, another canonical
Evolutionary HMMs

A model providing heterogeneity, long-term conservation and large indels was proposed by Mitchison and Durbin (1995) and has subsequently been developed (Durbin et al., 1998; Mitchison, 1999a). This model is called the ‘tree HMM’. It has certain similarities to profile HMMs, the main one being that it associates hidden information with a sequence.

A tree HMM may be pictured as a structured array of arrows similar to signals (or points) along two parallel railway tracks (Figure 3). The train runs from left to right; its exact route is determined by the positions of the signals. Each signal has two possible settings: ‘go straight ahead’ or ‘cross over to the other track’. Suppose that these railway signals are subject to thermal fluctuations: at any instant, each may flip to its alternate possible state, so that the statistical behavior of an individual signal is similar to a residue in the substitution model described above. In other words, each signal is modeled by a continuous-time two-state Markov chain with a separate transition matrix for each signal.

Now suppose that the signals are frozen in place and a train moves along the track. The path of the train will be determined by the instantaneous positions of the signals. Every time the train passes a signal site on the top row, the model emits a residue from a site-dependent probability distribution. The bottom row of signals is silent. Thus the top row contains ‘match’ states and the bottom row ‘delete’ states, in HMM terminology. However, in contrast to the match states of standard HMMs, which model a single-residue probability distribution, residues aligned to match states of tree HMMs may be related by a full substitution model.

As with the links model, inference of the alignment $\pi$
is a dynamic programming problem. The underlying Pair HMM has a chained structure.

The tree HMM is an attractive model. One drawback is that it does not attempt to model insertions. Some sort of fusion between the tree HMM and the links model might remedy this, since the equilibrium distribution of the links model mirrors that of an insert state in the profile HMM, in the same way that the path through the match and delete sites of the tree HMM converges on the distribution of paths through the match and delete states of the profile HMM.

Another drawback of the tree HMM is that, technically, it does not yield likelihoods of the form equation (1), since in that equation, \( A \) and \( D \) are sequences, whereas the evolving entity here is ‘the flip-state of all the signals, plus the residues at each match state’ which is more information than is carried by the sequence alone. For example, after a deletion, there may still be memory of part of the deleted sequence being deleted previously, and if reinsertion occurs, the old deletion will tend to be restored as well. Any individual path only goes through half of all the signals, and anyone observing the path (or the sequence) alone cannot determine the flip-state of the signals that are off the path. Thus, to obtain something like equation (1), the flip-states of these signals must be summed out. This leads to problems with the additivity condition equation (3), particularly for short times \( t \) (Mitchison, 1999a). However, these problems may not present a serious drawback for practical use.

1.4 An implementation of the links model

As a step towards developing evolutionary models for sequence alignment, we have implemented a general-purpose multiple alignment algorithm based on the links model. We consider this to be the simplest of the above-described models capable of doing multiple alignment, and one that may be useful in the design of more sophisticated systems in future.

We first present the links model in the terminology of Pair HMMs, and show how the likelihood for a multiple alignment may be factored into pairwise branch likelihoods. We describe a set of Markov Chain Monte Carlo (MCMC) sampling moves that are sufficient to achieve ergodicity in the space of all alignments while preserving the posterior distribution over sequence alignments. We note that the actual sequences at internal nodes can be summed out during alignment sampling, by replacing residues with ‘wildcards’. We present our program implementing these ideas, Handel, the source code to which may be modified and redistributed free of charge by all users under the terms of the GNU Public License (GPL, 2000). The program includes experimental features such as simulated annealing and a variant of over-relaxation (a method of exploring the alignment space at a rate faster than predicted by Brownian drift while leaving the posterior distribution unchanged). We describe the behavior of the program on both simulated test cases and benchmark structural alignments. Finally we discuss certain computer science and mathematical considerations that we hope may be helpful in designing novel evolutionary models.

2 MODEL

Let us reprise the links model as set forth by Thorne et al. (1991). Consider the legacy of an individual residue. Let \( p_n(t) \) be the probability that, at time \( t \), it has survived, spawning \( n \) descendants (including itself, its children, its grandchildren, its great-grandchildren and so on). Since the insertion rate is \( \lambda \) and the deletion rate is \( \mu \), the time-evolution of \( p_n(t) \) is described by

\[
\frac{dp_n}{dt} = \lambda(n-1)p_{n-1} + \mu np_{n+1} - (\lambda + \mu)n p_n
\]

with \( p_1(t) = 1 \), \( p_0(t) = 0 \) for \( n > 1 \), and taking \( p_n(t) = 0 \) for \( n \leq 0 \) at all \( t \).

The other eventuality is that, by time \( t \), the residue has died leaving \( n \) descendants. Call the probability of this event, \( q_n(t) \). It evolves as follows

\[
\frac{dq_n}{dt} = \begin{cases} 
\lambda(n-1)qn_{n-1} + \mu(n+1)qn_{n+1} & \text{for } n > 0 \\
\mu(q_1 + p_1) & \text{for } n = 0
\end{cases}
\]

with \( q_n(t) = 0 \) for all \( n \).

We must also consider descendants of the immortal link at the left-hand end of the sequence. Let \( r_n(t) \) be the probability that there are \( n \) such residues at time \( t \); then

\[
\frac{dr_n}{dt} = \begin{cases} 
\lambda nr_{n-1} + \mu(n+1)r_{n+1} - \lambda(n+1)r_n + \mu r_n & \text{for } n > 0 \\
\mu r_1 - \lambda r_0 & \text{for } n = 0
\end{cases}
\]

The solutions to the above equations are

\[
\begin{align*}
p_n &= \alpha \beta^{n-1} (1 - \beta) \\
q_n &= (1 - \alpha)(1 - \gamma) \quad \text{for } n = 0 \\
&\quad = (1 - \alpha)\gamma \beta^{n-1} (1 - \beta) \quad \text{for } n > 0 \\
r_n &= \beta^n (1 - \beta)
\end{align*}
\]

where

\[
\begin{align*}
\alpha(t) &= e^{-\mu t} \\
\beta(t) &= \frac{\lambda(1 - e^{(\lambda - \mu)t})}{\mu - \lambda e^{(\lambda - \mu)t}} \\
\gamma(t) &= 1 - \frac{\mu(1 - e^{(\lambda - \mu)t})}{(1 - e^{-\mu t})(\mu - \lambda e^{(\lambda - \mu)t})}
\end{align*}
\]

Conceptually, \( \alpha \) is the probability of ancestral residue survival, \( \beta \) is the probability of more insertions given one
or more extant descendants and \( \gamma \) is the probability of insertions given that the ancestral residue did not survive.

The limiting behavior for very long branches is

\[
\begin{align*}
\alpha(t) & \rightarrow 0 \\
\beta(t) & \rightarrow \lambda/\mu \\
\gamma(t) & \rightarrow 0
\end{align*}
\]

as \( t \rightarrow \infty \). (7)

Since only \( \beta \) stays finite at large \( t \), both \( p_n \) and \( q_n \) tend to zero for \( n > 0 \), leaving only \( r_n \) finite for nonzero \( n \). In other words, all ancestral residues and their descendants tend inevitably to die away, leaving only residues descended from the immortal link. The limiting behavior of the distribution \( r_n \) as \( t \rightarrow \infty \) thus gives the equilibrium length distribution of the sequence, which is geometric with mean \( \lambda/\mu \).

Note that \( \lambda < \mu \) is a necessary condition to prevent the sequence from tending to infinite length. The immortal link is then required to prevent the sequence from tending to zero length.

2.1 The links model as a Pair HMM

Equations (5) and (6) may be framed succinctly using a type of hidden Markov model called a ‘Pair HMM’ (Durbin et al., 1998). A Pair HMM is similar to a standard sequence-emitting HMM (a ‘Single HMM’), but instead emits residues in two separate sequences. An alignment of two sequences to a Pair HMM implicitly aligns the sequences to each other as well. Pair HMMs have been used to obtain posterior probabilities for residue correspondence (Miyazawa, 1994), to recast standard pairwise sequence comparison algorithms in a probabilistic form (Bucher and Hofmann, 1996), and to study the accuracy of such algorithms analytically (Holmes and Durbin, 1998).

For our purposes, emitting states in a Pair HMM come in three flavours: either a state can emit residues in one or other of the two sequences, or it can emit residues in both sequences at once. These three kinds of state correspond to the three directions of score propagation in a pairwise DP matrix (e.g. as used by the Smith–Waterman algorithm) and may be thought of as match, insert and delete states. We also introduce ‘null’ (non-emitting) states to simplify the model.

The Pair HMM for the links model is shown in Figure 8 (the tree on which this model is based is shown in Figure 4). The central recurrent loop of this model uses all three types of state and describes the fate of an individual ancestral residue. Either the residue lives (match state) or dies (delete state). In each case it spawns a geometrically-distributed number of ancestor residues (insert state) although the geometric distribution is subtly altered if the ancestor dies (the match→insert and delete→insert transitions have different probabilities).

Note that a cursory inspection of Figure 8 reveals no direct delete→delete transition. This is because a deleted ancestral link may have given birth to orphaned descendant links. If transitions via null states are considered, however, there is a direct self-transition from the delete state with probability \( \frac{\lambda}{\mu} (1 - \gamma)(1 - \alpha) \).

The path likelihood for the Pair HMM of Figure 8 is as in equation (1), i.e. \( \Pr[A \rightarrow D] \ldots \). Here the evolutionary relationship \( \pi \) is taken to be the path through the Markov model. This Pair HMM can be used to derive a Single HMM modeling \( \Pr[A \ldots] \), the equilibrium probability of any sequence, by summing out descendant emissions in the Pair HMM. Furthermore, a Single HMM modeling \( \Pr[A \rightarrow D]A \ldots \), the conditional probability of a descendant sequence given its ancestor, can also be derived. Such an HMM can be represented as a profile HMM as implemented by SAM or HMMER.

Inference of the alignment of the two sequences, \( \pi \), employs dynamic programming. (Recall that \( \pi \) describes the evolutionary relationship between the sequences.) The optimal value of \( \pi \) may be obtained using the Viterbi algorithm; alternatively, the Forward algorithm can be used to calculate the sum-over-alignments likelihood \( \Pr[A \rightarrow D] = \sum_{\pi} \Pr[A \rightarrow D] \) or to sample an alignment from the posterior distribution \( \Pr[\pi|A \rightarrow D] \). These algorithms (Forward/Viterbi) and other standard HMM algorithms have been described with application to biological sequence analysis in the published literature (Durbin et al., 1998; Holmes, 2000). We assume some familiarity with these algorithms; the novice reader is referred to the textbook by Durbin et al. (1998).

There is one last point to note about the Pair HMM for the links model. Recall that our reversibility criterion for an evolutionary model, equation (2), required that an inversion relationship \( \bar{\pi} \) be defined for all alignments \( \pi \). For the links model, the inversion requires swapping insert and delete states in the path, except for those regions of the path where there are even numbers of alternating insert and delete states (see Figure 7 for an example). That this path inversion satisfies reversibility in general may be seen by considering the effect of the transformation on the Pair HMM transition matrix.

2.2 From Pair HMMs to Multiple HMMs

We now introduce the concept of a ‘Multiple HMM’. Just as a Pair HMM emits residues in two sequences using
three types of emit state (plus null states), a Multiple HMM emits residues in \( N \) sequences using up to \( 2^{N} - 1 \) types of emit state (one for every non-empty subset of the \( N \) sequences). The Viterbi and Forward algorithms for alignments of \( N \) sequences to a Multiple HMM use \( N \)-dimensional dynamic programming matrices.

Given an evolutionary tree relating \( N \) sequences (including the sequences at internal tree nodes), one can construct a composite Multiple HMM analogous to the Pair HMM shown in Figure 8 by considering the ancestor–descendant relationship of every branch. For example, the three-node tree in Figure 5 has the HMM shown in Figure 9, and the four-node tree in Figure 6 has the HMM shown in Figure 10. Note how the HMM structure of Figure 8 is embedded in Figure 9; likewise, Figure 9 is embedded in Figure 10.

It is useful to visualize composite Multiple HMMs and their relationship to the factorization of equation (4). There is an algorithm to construct a composite Multiple HMM for any evolutionary Pair HMM and tree \( T \), such that the likelihood function for this Multiple HMM is equal to \( \Pr([\pi], S[T, \Theta]) \) of equation (4) (unpublished). When this algorithm is applied to the links model, it produces an HMM whose Forward recursion is identical to a dynamic programming recursion described by Hein (2001).

Not all dynamic programming to Multiple HMMs has to be \( N \)-dimensional, and we have developed ways to sample alignments without recourse to such impractical measures.

By constraining the pairwise alignments along subsets of all tree branches (and the inferred sequences at subsets of all tree nodes), we provide a Gibbs sampler for the likelihood function described in equation (4). Progressive alignment and refinement algorithms are obtained by replacing the construct ‘sample an alignment \( \pi \) using the Forward algorithm’ with ‘find the optimal alignment \( \pi \) using the Viterbi algorithm’ in the Gibbs sampler code.

2.3 Composing a multiple alignment from pairwise branch alignments

From the above treatment (in particular equation 4) it should be clear that, for the present work (as with previous work by others) a multiple alignment is a composition of pairwise branch alignments. The multiple alignment represents the complete evolutionary history of all the sequences, whereas the pairwise alignments represent the individual historical accounts of each branch.

For our alignment algorithm, we need a well-defined and unambiguous way of decomposing a multiple alignment into a set of pairwise branch alignments for neighboring nodes. More generally, it is useful to obtain the pairwise alignment of any two nodes of the tree (not just neighboring nodes). Conversely, we need a way of composing a multiple alignment from a complete set of pairwise alignments. (A useful generalization of this task is to find the optimal multiple alignment given an incomplete pairwise alignment set.)
Fig. 8. The Pair HMM for the links model on the single-branch tree shown in Figure 4. Null states are shown as small circles and emit states as large circles. The parameters $\alpha$, $\beta$ and $\gamma$ are related to the branch length $t$ as described in equation (6).

Fig. 9. The Multiple HMM for the binary tree of Figure 5. Note that $\alpha_1 = \alpha(t_1)$, $\alpha_2 = \alpha(t_2)$, etc., according with equation (6).

The algorithms to do this will not be described in full, but the essential rule is as follows: residues $X_i$ and $Y_j$ in a multiple alignment containing sequences $X$ and $Y$ are considered to be aligned if and only if both of the following conditions hold:

- the residues $X_i$ and $Y_j$ are in the same column;

...
the column contains no gap characters for any of the sequences intermediate to X and Y on the tree.

The ‘intermediate’ sequences of the second clause include any sequences in the lineages \( A \rightarrow X \) or \( A \rightarrow Y \), where \( A \) is the most recent common ancestor of \( X \) and \( Y \) (according to the tree). This clause stipulates that residue deletion followed by re-insertion at the same position does NOT constitute a direct evolutionary relationship under this model (in agreement with Thorne et al.) and there should be no correlation between such residues. (To affirm this, note that Figure 10 contains no states that emit both \( W \)s and \( Y \)s without emitting \( X \)s; this is because \( X \) is intermediate to \( W \) and \( Y \) in Figure 6.) Thus alignment columns may be subdivided into ungapped cliques.

These rules, together with the stipulation that empty (all-gap) columns be ignored, provide an unambiguous way of converting multiple alignment into a set of pairwise branch alignments (and vice versa) given a guide tree.

### 2.4 Eliminating internal nodes

Often, the actual inference of ancestral sequences (at internal nodes of the tree) is unnecessary. In Bayesian language, these sequences are ‘missing data’ and the correct thing to do would be to sum them out of the likelihood function.

Unfortunately, summing over the indel histories of these sequences means relinquishing the branch-to-branch independence that allows us to conveniently factorize the likelihood function as in equation (4). In other words, if we consider all indel histories then we find ourselves again with a high-dimensional dynamic programming problem.

Despite the indel histories remaining intractable, we can sum over all substitution histories for a given multiple alignment using the post-order traversal algorithm of Felsenstein (1981) In computing a column (clique) likelihood, this algorithm stores conditional likelihoods for all the residue possibilities at missing nodes. (Note that a post-order traversal of a tree visits children before parents.)

Conditional distributions for any missing residue in the alignment can be calculated in this way. The conditional distributions for entire missing sequences can be modeled by profile HMs.

Whenever internal sequences are summed out by the Handel software in calculating an alignment score, the corresponding multiple alignment is printed with asterisk characters (‘*’) for the residues of internal sequences. Note that these differ in meaning from the standard IUPAC ambiguous characters (‘N’ for DNA, ‘X’ for protein). Where the score for a standard IUPAC character
is typically taken to be a (possibly weighted) average of all possible residue scores, our wildcard asterisk is scored as a probabilistic sum over all residues. Thus the score for matching a random protein residue to an ‘X’ is typically negative, whereas the score for matching the same residue to a ‘*’ is always zero. Since this summing out of internal residues essentially reprises the Felsenstein algorithm (Felsenstein, 1981), we refer to the asterisks as Felsenstein wildcards.

The intractability of summing out the indel histories at internal nodes remains a problem: without a way of eliminating these artefacts from the analysis, we are stuck with a maximum likelihood solution and our Bayesian expressiveness is restricted. In MCMC analysis, computational intractabilities like this are circumvented by sampling extensively from the posterior distribution. This is the approach taken by Håndel. We proceed by describing the Gibbs sampling moves that we use to navigate alignment space.

3 ALGORITHM

The most popular kind of multiple alignment algorithm is progressive alignment, whereby profiles for missing parents are estimated by aligning sibling sequences on a post-order traversal of the underlying binary tree (Feng and Doolittle, 1987; Barton and Sternberg, 1987; Taylor, 1987; Thompson et al., 1994). Recent algorithms have employed iterative refinement, revisiting branches following the initial alignment phase (Morgenstern et al., 1996). We henceforth refer to progressive alignment as ‘impatient’ and iterative refinement as ‘greedy’ strategies. A third (patient, non-greedy) strategy is to sample from a population of alignments, exploring suboptimal alignments in anticipation that short-term sacrifices will yield long-term improvements (Eddy, 1995; Notredame and Higgins, 1996).

All of the above types of strategy are used by Håndel. We begin by describing the sampling strategy, being the most principled approach (from a Bayesian viewpoint) from which the former two strategies can be derived.

Conceptually, a Håndel multiple alignment consists of (i) a binary tree; (ii) a set of observed (leaf-node) sequences; (iii) a set of probability profiles for missing (internal-node) sequences (typically containing Felsenstein wildcards) and (iv) a pairwise alignment associated with each branch. Probability profiles over internal sequences behave like Plan 7 profile HMMs (hmmer.wustl.edu).

Three types of ‘move’ are used to explore alignment space. The goal of these moves is to provide ergodicity, i.e. to provide a set of edit operations that, by composition, allow any one alignment to be transformed into any other alignment. (We do not consider edit operations on the underlying phylogenetic tree; for a treatment of this see Mau et al., 1996). We first discuss the motivation for each move, then describe the move slightly more formally.

The first move mirrors the sibling alignment step of impatient-progressive alignment. Given two sibling sequences, their alignment is sampled. The length of the parent sequence is implicitly sampled at this stage as well.

- **Move #1: parent sampling**
  - The goal is to align sibling nodes Y and Z and simultaneously infer their parent node X (see Figure 5).
  - Construct the Pair HMM for X, Y and Z (see Figure 9. Technically, this is a Multiple HMM and not a Pair HMM, but since X is missing data, the dynamic programming matrix is still only two-dimensional).
  - Sample the alignment of Y and Z using the Forward algorithm.
  - Deduce the implicit alignments XY and XZ and the sequence X.

The second move mirrors the branch alignment step of greedy-refined alignment. Given a branch, the pairwise alignment for that parent–child sequence pair is resampled, by applying the Forward algorithm to Figure 8. This move resamples alignments inferred during the impatient-progressive phase.

- **Move #2: branch sampling**
  - The goal is to realign the adjacent nodes X and Y (see Figure 4).
  - Fix all pairwise branch alignments except branch XY.
  - Construct the Pair HMM for X and Y (see Figure 8).
  - Resample the alignment of X and Y using the Forward algorithm.

The third move completes the ergodicity requirement. Given any internal node, the sequence at that node is resampled by inserting or deleting residues without disturbing the pairwise alignment of adjacent nodes. This move resamples parent sequence lengths inferred during the impatient-progressive phase.

- **Move #3: node sampling**
  - The goal is to resample the sequence at internal node X.
  - Let the parent of X be W. Let the children of X be Y and Z (see Figure 6).
  - Fix all pairwise branch alignments except branches WX, XY and XZ.
— Construct the Multiple HMM for $X$ and its neighbours (see Figure 10).

— Resample the sequence $X$, conditioned on the relative alignment of $W$, $Y$ and $Z$. (In other words, all variants of the original multiple alignment having either a residue or a gap character at each column of row $X$ are considered. Furthermore, each column may be followed by zero or more inserted columns containing a residue in row $X$ and a gap character in all other rows. This yields a dynamic programming problem whose complexity is linear in the original alignment length.)

Each of the three moves relies on the decomposition of the joint multiple alignment likelihood into a product of pairwise branch likelihoods, as in equation (4). Each involves fixing or partially constraining some or all of the pairwise branch alignments and sampling from the resulting subspace. Since the moves sample exactly from conditional distributions, they satisfy detailed balance (Gilks et al., 1996). Our algorithm thus generates an unbiased sample from the posterior distribution over alignments.

The Gibbs sampler proceeds as follows: firstly, a multiple alignment is constructed from unaligned sequences by parent sampling up the guide tree; secondly, each branch and node is visited once (but in a random order) for either branch sampling or node sampling respectively (the frequency of node sampling versus branch sampling moves is specified by the user). The second step is repeated to generate as many samples as required.

For each of the sampling moves, a maximum-likelihood equivalent may be obtained by replacing the sampling step (the Forward algorithm) with an optimization step (the Viterbi algorithm). Thus, impatient-progressive alignment (Thompson et al., 1994) corresponds to application of the maximum-likelihood version of parent sampling and greedy-refined alignment corresponds to application of the maximum-likelihood versions of branch sampling and node sampling.

As well as using Gibbs sampling in a Bayesian context to average over suboptimal alignments, we are interested in using it in a maximum-likelihood context to inject noise and escape locally optimal solutions. Thus, the maximum-likelihood version of the Gibbs sampler periodically saves the current alignment, applies the greedy-refinement algorithm, records the likelihood of the refined alignment then restores the saved alignment. At the end of the sampling run, the refined alignments are considered along with the sampled alignments, and the alignment with the highest likelihood is picked. The suggested period for greedy-refinement is the correlation time of the Gibbs sampler.

A related strategy for escaping local optima is simulated annealing (Kirkpatrick et al., 1983). This has been applied to multiple alignment (Lukashin et al., 1992; Kim et al., 1994). A simulated annealing version of our algorithm, with a variable temperature parameter $T$, may be obtained by raising all ‘probabilities’ to the power $1/T$. It is possible to sample from the (fixed-temperature) posterior distribution using annealing methods (Neal, 1998).

Commonly, a multiple alignment is produced using a program that either does not recognize the existence of sequences at internal nodes or does not output said sequences. A multiple alignment from such a program may be adapted for the links model by using node sampling to infer internal sequences consistent with the supplied alignment.

3.1 Ordered over-relaxation

A problem with Gibbs sampling and other MCMC methods is that a random walk on a Markov chain follows Brownian motion statistics, i.e. the root-mean-squared drift grows only as $n^{1/2}$ where $n$ is the number of steps taken.

This is a fairly slow way of exploring alignment space. It would be faster if we could make the random walk somewhat self-avoiding, i.e. build in some kind of heuristic to ‘boldly go where no alignment has gone before’ rather than retreading old turf. But can this be done without destroying detailed balance, so that the algorithm still emits alignments from the posterior distribution?

The answer is yes, it can, using a technique developed by Neal (1995). The key is to define a strict weak ordering on alignments, i.e. a consistent sort relation that places similar alignments together. (In our software, we sort by the centroid of all match states resolved transverse to the main diagonal of the dynamic programming matrix.)

Once the order statistic is defined, it may be used to deter self-repeating sampling behaviour by sampling $N$ new alignments at each Gibbs sampling move (instead of just one). The $N$ alignments, plus the original alignment, are then sorted. If the position of the original alignment in the sorted list is $k$, then the $(N - k)$th alignment in the list is chosen to be the new one. This procedure is known as ‘ordered over-relaxation’ and a symmetry argument demonstrates that it preserves detailed balance (Neal, 1995). Sampling $N$ alignments from a forward matrix is not significantly more computationally expensive than sampling one, since resampling can be carried out without recomputing the matrix.

4 IMPLEMENTATION

The algorithms described here are implemented in the handel package, which is available from http://www.biowiki.org/Handel. Handel is written in C++ for a Unix system and distributed under the GNU Public
License (GPL, 2000). The code compiles cleanly using the gcc compiler version egcs-2.91.66 or later. It was
developed under RedHat Linux version 6.0.

The package consists of three programs (prefixed tkf after Thorne, Kishino and Felsenstein, who first described
the links model):

- tkfemit, given a guide tree, generates an alignment
from the links model;

- tkfdistance, given a set of sequences (aligned or
unaligned), estimates a distance matrix under the links
model by using bracketed minimization (Press et al.,
1992) to find the ML time parameter for each sequence
pair;

- tkfalign, given a set of sequences and a guide tree,
samples alignments from the posterior distribution
and/or attempts to find the highest-scoring alignment.

The following are specifiable by the user:

- the birth and death rates λ and μ (or, equivalently,
the mean sequence length λ/μ and the total indel rate
λ + μ);

- the substitution rate matrix (specified as a sym-
metric matrix Q together with an equilibrium residue
probability vector p; the reversible (generally
non-symmetric) rate matrix derived from this is
\( P^{1/2} Q P^{-1/2} \) where \( P \) is a diagonal matrix whose
diagonal elements are the entries of p, i.e. \( p_{ij} = p_i \delta_{ij} \)
(Bruno and Arvestad, 1997)). The Dayhoff PAM series
is provided as a default option for proteins (Dayhoff et
al., 1978), as is the 6-parameter transition/transversion
model with nucleotide bias for DNA (Hasegawa et al.,
1985);

- various control parameters for the Gibbs sampler
including the number of sampling rounds, the ratio
of Branch Sampling to Node Sampling moves, the
annealing schedule and the number of trials for ordered
over-relaxation.

We avoid a full description of the Handel software
architecture here, but note that it employs a generic
dynamic programming engine (inspired by the Dynamite
project (Birney and Durbin, 1997)) for probabilistic work
with Single and Pair HMMs of arbitrary topology, as well
as custom DP code for the Multiple HMMs of Figures 9
and 10. It also uses a logging and exception-handling
framework that is customizable by the user at run-time.
For example, if the user specifies the ‘-log BREAKDOWN’
option on the command line, a full breakdown of the
alignment log-likelihood by branch and node is reported
at each sampling step. Likewise, DP matrices, tracebacks,
HMM transition probabilities, substitution matrices and
many other intermediate computation results can be
selectively output (to files or standard error) without
recompiling the code. The infrastructure supporting these
features is provided by the DART (DNA/Amino/RNA
tools) library (Holmes, 2000).

Handel also links to some external open source li-
braries, namely Numat09 for linear algebra (Davies,
1999) and randlib for random number generation
(Brown et al., 1997).

We tested Handel on simulated data. We report the
results of these tests as well as the behaviour of the
algorithm on real biological sequences. In view of the
performance of the algorithm on real data we suggest a
number of improvements to the links model to improve its
suitability for practical use.

4.1 Tests on simulated data

As a control test of the multiple alignment algorithm’s
ability to reconstruct evolutionary histories that are known
to fit the links model, we generated trees from a Yule
process (Durbin et al., 1998) and then generated multiple
DNA sequence alignments using Handel’s tkfemit
program. Various different settings were tried for the
number of leaf sequences, the mean sequence length, the
mean speciation time of the Yule process and the indel
rate.

We then tested several different variants of the align-
ment algorithm, with the aim of comparing impatient-
progressive (i.e. single-pass maximum likelihood
alignment), greedy-refined (i.e. multi-pass maximum
likelihood alignment) and a number of different sampling
strategies.

Firstly, we calculated a distance matrix from the un-
aligned leaf sequences using Handel’s tkfdistance
program and estimated a tree from this using neighbor
(Bruno et al., 2000). We then ran the following exper-
iments using Handel’s tkfalign program (driven by Perl scripts):

A. Starting from the full correct alignment, remove the
sequences corresponding to internal nodes. Keeping
the alignment of the leaf nodes fixed, re-estimate
the indel history of the internal sequences, without
using knowledge of which branches underwent
substitutions (i.e. replacing the internal sequences
with Felsenstein wildcards).

B. Starting with the unaligned leaf sequences and
the distance matrix, do an impatient-progressive
alignment, using wildcard residues at internal nodes.
(Recall that the impatient-progressive strategy does
a single pass over the tree from the leaves upwards,
estimating maximum-likelihood indel histories for
internal nodes.)
C. Starting with the alignment generated by experiment B, do a greedy-refined alignment. (Recall that the greedy-refined strategy does multiple passes over the tree, re-aligning until there are no more improvements to be found.)

D. Starting with the alignment generated by experiment C, do 100 sampling moves, followed by greedy-refinement. Felsenstein wildcards are used for the sequences at internal nodes. Each sampling move is chosen to be node sampling (at a random node) or branch sampling (at a random branch) with equal probability.

E. Starting with the alignment generated by experiment C, do 100 simulated annealing steps (from \( kT = 2 \) down to \( kT = 0 \)), followed by greedy-refinement. Felsenstein wildcards are used for the sequences at internal nodes. Each simulated annealing move is chosen to be node sampling (at a random node) or branch sampling (at a random branch) with equal probability.

F. Starting with the alignment generated by experiment C, do 100 ordered over-relaxed sampling moves, each with a trial set size of 11, followed by greedy-refinement. Felsenstein wildcards are used for the sequences at internal nodes. Each over-relaxed sampling move is chosen to be node sampling (at a random node) or branch sampling (at a random branch) with equal probability.

G. Starting with the alignment generated by experiment C, do 100 sampling moves without using Felsenstein wildcards for internal-node sequences, followed by iterative refinement. Each over-relaxed sampling move is chosen to be node sampling (at a random node) or branch sampling (at a random branch) with equal probability.

All experiments except A used the tree generated by \texttt{neighbor} (experiment A used the original simulated tree). For the sampling moves, the frequencies of branch sampling and node sampling were set to be equal. All algorithms summed over the residues of internal sequences (i.e. used wildcards rather than attempting to estimate the exact sequence) unless otherwise stated. The parameter settings to run \texttt{tkfalign} were set equal to those used with \texttt{tkfemit}.

The results of these simulation experiments are plotted in Figures 11 and 12. Figure 11 shows normalized log-likelihoods for the alignments generated by experiments A–G under various different conditions.

Let us consider the experiments in turn. Experiment A—replacing internal-node sequences by wildcards—is something of a control, to determine how the use of wildcards affects the log-likelihood. As expected, the log-likelihood always increases, by from 5 to 25% in these tests. However, impatient-progressive alignment (experiment B) yields, in all these test cases, a multiple alignment with a higher likelihood than the true alignment, indicating that (on average) the true alignment is not the highest-scoring one.

The likelihoods for experiment C (greedy-refinement) are higher than for B; furthermore, the variation in these likelihoods is smaller, indicating that greedy-refinement not only finds higher-scoring alignments but does so with better reproducibility. The same trend is evident, although less marked, when comparing experiment C to experiments D–F (sampling, annealing and over-relaxation). Little distinction is evident between D, E and F, however, suggesting that the method of sampling employed does not greatly affect the rate of convergence to the maximum likelihood alignment, at least for DNA sequences under these simulated conditions.

The final datapoint, experiment G, shows the likelihoods obtained in estimating residues at internal nodes, rather than using Felsenstein wildcards. It is interesting to note that this likelihood is in all cases higher than for experiment A. In other words, the maximum alignment likelihood is significantly greater than the sum over all alignments of leaf sequences consistent with the true alignment, indicating that the true signal is rivalled by noise.

The above observations hold valid for all the parameterizations tried in these experiments. In general, the gains in log-likelihood due to wildcards (A), impatient-progressive alignment (B), greedy-refinement (C) and sampling (D) are reduced as the ‘noise’ parameters are increased.

Figure 12 gives some idea of the accuracy of these procedures, measured by the symmetrized Sum-of-Pairs-Score (SPS), the probability that any two residues will be correctly aligned (Thompson et al., 1999). Little difference is discernible between experiments B–G, except perhaps a slight improvement following greedy-refinement. This may be because the sampling runs did not converge or because the signal-to-noise ratio is too low in these tests (Hwa and Lässig, 1998; Holmes and Durbin, 1998).

4.2 Tests on biological data

We tested \texttt{Handel} on five sets of reference alignments from BAliBASE, a database of structurally informed multiple alignments with annotations specifying the meaningful ‘core’ segments (Thompson et al., 1999). We tried three different algorithms on the BAliBASE sequence datasets:

K. Construct a distance matrix using the \texttt{tkfdistance} program in \texttt{Handel}. Estimate a tree using \texttt{neighbor} (Bruno et al., 2000). Do an impatient-
Evolutionary HMMs

Fig. 11. Rescaled log-likelihoods for the simulation experiments of Section 4.1. All simulations used the 6-parameter nucleotide substitution model (Hasegawa et al., 1985) with substitution rate 1 and transition/transversion ratio 4. Trees were generated from a Yule process with mean speciation time $t$. The birth–death parameters were determined by the total indel rate $i = \lambda + \mu$ and by the mean sequence length. The points and error bars show 95% confidence limits for the mean fractional change in log-likelihood (relative to the true alignment) over 1000 trials per experiment, assuming the central limit theorem applies. The log-likelihoods in this plot were normalized relative to the true alignments to allow for the wide variability induced by our procedure of generating a new tree for each trial using a Yule process. The individual experiment parameters were as follows. Upper left: 6 leaf sequences, mean length 50 nt. Upper right: 12 leaf sequences, mean length 50 nt. Lower left: 24 leaf sequences, mean length 100 nt. Lower right: 48 leaf sequences, mean length 100 nt.

progressive alignment using the tkfalign program.

L. Apply greedy-refinement to the alignment produced by K using Handel's tkfalign program.

M. Apply the Gibbs sampler with ordered over-relaxation and periodic refinement to the alignment produced by L, using Handel's tkfalign program. The period for refinement is $10(1 + N \log N)$ where $N$ is the number of sequences to be aligned.
Fig. 12. Symmetrized Sum-of-Pairs Score (SPS) for the simulations described in Section 4.1 and Figure 11. The SPS score is defined to be the probability that any two aligned residues from the reference alignment are aligned in the test alignment (Thompson et al., 1999). For these tests a symmetrized SP score (i.e. averaged with the reference and test alignments swapped) was used. The parameters are as defined in Figure 11. As with Figure 11, the error bars show 95% confidence limits for the mean SPS of each experiment.

The total number of samples is $1 + N \log N$ times the refinement period. (Note that $N \log N$ is proportional to the time taken to sort the tree.)

Release 2.01 of BAliBASE was used for these tests. The benchmark was conducted without any optimization of Handel’s parameters: the default settings (mean sequence length 50, total indel rate 0.1, Dayhoff substitution model) were used for all tests. Brief descriptions of the BAliBASE categories may be found in Table 1.

The results of the BAliBASE benchmark are summarized in Tables 2 and 3. Table 2 lists the mean (unsymmetrized) SPSs for each BAliBASE category, whereas
from greedy-realignment, with only 27% decreasing. An increase in going from impatient-progressive to greedy-realignment, overall, 62% of the test sets showed an increased SPS.

The improvements were most prevalent for datasets the test alignments improving and only 19% deteriorating.

As a footnote to the above discussion, it should in fairness be noted that the BAliBASE benchmark inherently favors global alignment algorithms (such as Handel’s) over local algorithms, since trimmed sequences are used for the alignments. The performance of Handel at local alignment has not been evaluated. However, it may be possible to garner some idea from the performance on BAliBASE.

Table 1. BAliBASE categories (Thompson et al., 1999)

<table>
<thead>
<tr>
<th>BAliBASE subdirectory</th>
<th>Description of aligned sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>ref1/test1</td>
<td>Equidistant, similar lengths; high identity (&gt;35%)</td>
</tr>
<tr>
<td>ref1/test2</td>
<td>Equidistant, similar lengths; medium identity (20–40%)</td>
</tr>
<tr>
<td>ref1/test3</td>
<td>Equidistant, similar lengths; low identity (&lt;25%)</td>
</tr>
<tr>
<td>ref2/test1</td>
<td>Highly-related family (&gt;25%); plus ‘orphan’ outliers (&lt;20%)</td>
</tr>
<tr>
<td>ref3/test</td>
<td>Equidistant divergent subfamilies (&lt;20% between subfamilies)</td>
</tr>
<tr>
<td>ref4/test</td>
<td>N/C terminal extensions</td>
</tr>
<tr>
<td>ref5/test</td>
<td>Insertions</td>
</tr>
</tbody>
</table>

The SPS is defined by Thompson et al. as the frequency with which any two residues from the same column of the reference alignment are correctly aligned in the test alignment (Thompson et al., 1999). As in Thompson et al. (but unlike Figure 12) the SPS for these tests was not symmetrized with respect to the reference and test alignments. The SPSs for the CLUSTALW program (Thompson et al., 1994) are displayed for comparison.

Table 2. Mean SPS for the reference alignments in BAliBASE, using the tests described in Section 4.2

<table>
<thead>
<tr>
<th>BAliBASE subdirectory</th>
<th>Handel (SPS)</th>
<th>CLUSTALW (SPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K</td>
<td>L</td>
</tr>
<tr>
<td>ref1/test1</td>
<td>0.775</td>
<td>0.784</td>
</tr>
<tr>
<td>ref1/test2</td>
<td>0.673</td>
<td>0.689</td>
</tr>
<tr>
<td>ref1/test3</td>
<td>0.654</td>
<td>0.658</td>
</tr>
<tr>
<td>ref2/test1</td>
<td>0.814</td>
<td>0.827</td>
</tr>
<tr>
<td>ref3/test</td>
<td>0.481</td>
<td>0.525</td>
</tr>
<tr>
<td>ref4/test</td>
<td>0.348</td>
<td>0.359</td>
</tr>
<tr>
<td>ref5/test</td>
<td>0.573</td>
<td>0.603</td>
</tr>
<tr>
<td>All</td>
<td>0.660</td>
<td>0.675</td>
</tr>
</tbody>
</table>

Table 3. Mean Total Column Score (TCS) for the reference alignments in BAliBASE, using the tests described in Section 4.2

<table>
<thead>
<tr>
<th>BAliBASE subdirectory</th>
<th>Handel (TCS)</th>
<th>CLUSTALW (TCS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K</td>
<td>L</td>
</tr>
<tr>
<td>ref1/test1</td>
<td>0.640</td>
<td>0.656</td>
</tr>
<tr>
<td>ref1/test2</td>
<td>0.535</td>
<td>0.553</td>
</tr>
<tr>
<td>ref1/test3</td>
<td>0.519</td>
<td>0.523</td>
</tr>
<tr>
<td>ref2/test1</td>
<td>0.266</td>
<td>0.292</td>
</tr>
<tr>
<td>ref3/test</td>
<td>0.083</td>
<td>0.118</td>
</tr>
<tr>
<td>ref4/test</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>ref5/test</td>
<td>0.277</td>
<td>0.330</td>
</tr>
<tr>
<td>All</td>
<td>0.401</td>
<td>0.420</td>
</tr>
</tbody>
</table>

The TCS is defined as the frequency with which entire columns from the reference alignment are exactly reproduced in the test alignment (Thompson et al., 1999). The TCSs for the CLUSTALW program (Thompson et al., 1994) are displayed for comparison.

In general, Handel performs less well than CLUSTALW, correctly aligning 13% fewer residue pairs and 19% fewer exact columns. The differences are most marked for the ref4 category (N/C terminal extensions), followed by ref3 (divergent families) and ref5 (insertions). A possible explanation for the ref4 performance is that the links model is global and is thus poorly equipped to deal with N/C-terminal extensions; nor does it model affine gaps (i.e. single-event large indels). This latter fact may also explain the performance on the ref5 benchmark. As for ref3, we note that the links model does not make any attempt to profile the protein, whereas the CLUSTALW algorithm contains several profiling-like steps that implicitly introduce prior knowledge of both indel and substitution patterns in nature (see e.g. Durbin et al., 1998, for a discussion). Another difference between CLUSTALW and Handel is that CLUSTALW uses the BLOSUM series of substitution matrices (Henikoff and Henikoff, 1992) which may be better models for long-time conservation but do not satisfy the additivity constraint of equation (3). Thus CLUSTALW should do—and does—markedly better at reconciling the divergent families of ref3.

All of the above points indicate areas in which the links model might be improved. Suggestions as to how some of these improvements (elementary profiling, variable-size indels) might be achieved were advanced by Thorne et al. (1992) in their second publication on the links model.

As a footnote to the above discussion, it should in fairness be noted that the BAliBASE benchmark inherently favors global alignment algorithms (such as Handel’s) over local algorithms, since trimmed sequences are used for the alignments. The performance of Handel at local alignment has not been evaluated. However, it may be possible to garner some idea from the performance on BAli-
iBASE reference set ref4, as the N/C terminal extensions of these alignments are similar to the troublesome ‘ragged ends’ faced by local algorithms.

The comparison of a reference alignment with a single test alignment is fundamentally a maximum likelihood approach. It would be interesting to see how Hande1 fares in a Bayesian benchmark, e.g. finding the expectation of the SPS and TCS measures over the whole posterior distribution.

5 DISCUSSION

On the face of it, Bayesian multiple alignment has many attractions. Primarily, it is transparent: the underlying model is clearly defined and all the parameters appear in the likelihood function (and may, as a consequence, be ‘trained’ on data). Missing information can be handled systematically, by summing out (as with our Felsenstein wildcards) or by MCMC. Sampling is a natural solution to the problem of local maxima. Access to the posterior distribution (either directly or via MCMC) leads to meaningful confidence estimates for suboptimal alignments and offers a principled way to integrate other probabilistic methods such as sampling over trees or incorporating other data types. Marginal distributions at individual tree nodes can be used to generate profile HMMs customized for particular species or phyla.

In view of these enticements, we are encouraged by the performance of Hande1 on the BAliBASE benchmark. The mean residue pair fidelity (SPS) was only 13% lower than CLUSTALw, which seems remarkable considering the parametric simplicity of the links model: a substitution matrix, an indel rate and a mean sequence length are all it needs. To get this close to CLUSTALw’s accuracy without the use of affine gap penalties or hydrophobic core modeling is highly unexpected. We are of the opinion that these results conform the merit in pursuing probabilistic multiple alignment and we hope that our algorithms—and the freely available source code to Hande1—may be of assistance to researchers interested in developing more sophisticated evolutionary models.

Regarding such improved models, there are many ways forward. The BAliBASE benchmark results suggest that local alignment, affine gaps and some kind of profiling (or mutation rate heterogeneity) might be of benefit. Two of these areas were addressed by Thorne et al. (1992) who proposed adapting the links model to include: (i) ‘fast’ and ‘slow’ mutation zones and (ii) links spanning multiple residues, to allow for affine-type indels. Both of these modifications involve some demarcation of the sequence into regions as a necessary part of inference. The first modification (fast and slow zones) might be used to implement a local algorithm, by flanking a slow zone (the alignable region) with two very fast zones (the unalignable region). Another possibility is some kind of fusion between the links model and the tree HMM (Mitchison, 1999b). One concept that might be usefully imported from tree HMMs is the idea of having multiple classes of column for different sites (e.g. hydrophobic, hydrophilic, small residues, etc.) (Mitchison and Durbin, 1995). This is somewhat similar to the use of Dirichlet mixture priors in HMM profiling (Sjölander et al., 1996), albeit with less learning potential. For expanded learning potential, one might not wish to train entire substitution rate matrices on sparse datasets but instead to constrain the parameter space, so that (for example) one learns only the equilibrium residue distribution (Bruno, 1996). One can also imagine nested birth–death processes; for example, a birth–death process for protein domains, where each domain supported a birth–death process for residues (i.e. a singly-nested links model) (Thorne, 2000). One could then introduce multiple different classes of domain with different indel and substitution parameters. (Possibly ‘structural feature’ would be a more appropriate nomenclature than ‘domain’ here.) There may be many other interesting stochastic processes whose joint distributions converge (exactly or approximately) on hidden Markov models or indeed on stochastic context-free or higher power grammars.

These are just a few of the options available to model developers. As more genomes are sequenced and databases inexorably grow, well-defined models will be the only maintainable option for large-scale database clustering and organization. Probabilistic modeling may not be immortal. However, in the absence of a full solution to the protein folding problem, it may be safe to say that the death of this link is yet some way off.

Acknowledgements

Some of the code in Hande1 is taken directly from other open source projects. We gratefully acknowledge Robert Davies, author of the Newmat linear algebra library (Davies, 1999) and the University of Texas Biomathematics department for their randlib random number library (Brown et al., 1997). We are also grateful to Sean Eddy and Graeme Mitchison whose clear exposition of certain points of scoring and probability within the HMMER source code was of considerable help (Eddy, 1996).

This work has benefited from interactions with many people, particularly Graeme Mitchison and Jeff Thorne. We would also like to thank Joe Nemec, Julie Thompson-Maaloum, Kevin Karplus, Kristina Ecker, Guy Slater, Roger Sayle, Janet Newman, Jeff Koshi, David Pollock, Anders Krogh, Des Higgins, Eliza McKenna, Ewan Birney, Richard Durbin and Gerald Rubin.

W.J.B. and I.H. acknowledge support from DOE grant/contract W-7405-ENG-36 and Los Alamos National Laboratory. I.H. was supported by the 1998–1999 Fulbright Zeneca research fellowship for Bioinformatics.
REFERENCES


