The Posterior Probability Distribution of Alignments and its Application to Parameter Estimation of Evolutionary Trees and to Optimisation of Multiple Alignments.

L. Allison* & C. S. Wallace

TECHNICAL REPORT NO: 93/188

July 1993

email: lloyd@cs.monash.edu.au
Partly supported by Australian Research Council grant A49030439
* to whom correspondence should be sent.

Abstract. It is shown how to sample alignments from their posterior probability distribution given two strings. This can be extended to alignments of alignments. The result is firstly applied to the estimation of the edges of a given evolutionary tree over several strings. Secondly, when used in conjunction with simulated annealing, it gives a stochastic search method for an optimal multiple alignment.

Keywords: alignment, estimation, evolutionary tree, inductive inference, multiple alignment, sampling, stochastic.

1. Introduction.

Estimating the evolutionary "distance" between some given strings, for example DNA sequences, and finding a multiple alignment for them, are examples of inductive inference problems. We rarely know the "right" answer and must be satisfied with a "good" hypothesis. In previous papers (Allison et al 1992a, Yee & Allison 1993), minimum message length encoding was used to infer the relation, if any, between 2 strings and to compare models of evolution or relation. It is not possible to extend the algorithms given there directly to K>2 strings in any simple way due to rapidly increasing complexity. A stochastic process is one possible way around this difficulty. Such a process is described for two problems given a set of K>=2 strings and given an evolutionary-tree over them: firstly estimating the evolutionary "distances" on the edges or arcs of the tree and secondly finding a good multiple alignment. The method relies on sampling from the posterior probability distribution of alignments.

In an inductive inference problem we are given some data D and wish to form a "good" hypothesis H about the data. The following standard definitions are useful:

```
P(H&D)   =  P(H).P(D|H)   =  P(D).P(H|D)  -- joint probability

Definitions:
   P(H|D)  = posterior probability of H given D
   P(H)    = prior probability of H
   P(D|H)  = likelihood of H
   ML(P(E))= -log2(P(E))                  -- message length of event E

ML(H&D)  = ML(H)+ML(D|H) = ML(D)+ML(H|D) -- joint message length

ML(H|D)-ML(H'|D) = ML(H)-ML(H')+ML(D|H)-ML(D|H')
                 = -log posterior odds ratio of H & H'
```

In the present context, the data consist of some given strings. Alignments and evolutionary distances are hypotheses about those strings. P(D|H) is called the *likelihood* of the hypothesis H; it is a function of the data given the hypothesis and it is not the probability of the hypothesis. P(H) is the prior probability of the hypothesis H. P(H|D) is the posterior probability of H.

The message length (ML) of an event E is the minimal length, in bits, of a message to transmit E using an optimal code. A long message length indicates a low probability and a short message length indicates a high probability. It is convenient to work with message lengths, rather than probabilities, for a number of reasons: Typical probability values can be very small and the message lengths are of a more convenient magnitude for handling by a computer. The message paradigm reinforces the view that there should be no hidden parameter values in the statement of the hypothesis itself, ML(H). All real-valued parameters must be stated to some optimum but finite accuracy. In sequence alignment there is a natural *null-theory* which has a message length being that required to state the given strings individually; this takes approximately 2 bits per character in the case of DNA. It provides a method of hypothesis testing. The null-theory assumes that there is no pattern or structure in the data. This includes the assumption that an individual string is random. Wallace and Freeman (1987) give the statistical foundations of minimum message length encoding.

We often wish to find the "best" hypothesis, from a specified class of hypotheses. (Depending on the application, this might be the best evolutionary tree, a good estimate of the evolutionary "distance" between strings or the best multiple alignment.) It is generally possible to calculate, or at least to give a good approximation to, ML(H) and ML(D|H) under reasonable assumptions. ML(H) can even be ignored if it is a constant for all members of a class of hypotheses. It is usually not possible to calculate ML(D) which is unfortunate for it would yield ML(H|D). However by subtracting ML(H'&D) from ML(H&D) it is possible to get a $-\log_2$ posterior odds-ratio for two competing hypotheses H and H'; the shorter message length indicates the more likely hypothesis. If one hypothesis is the null-theory this also gives the hypothesis test.

An evolutionary tree which is a good hypothesis makes good predictions about the sorts of columns or *tuples* of characters that occur in good alignments. For example, given four strings s1 to s4 and assuming that all edges are similar, the tree ((s1 s2)(s3 s4)) predicts that tuples of the form xxyy appear more often than xyxy, whereas ((s1 s3)(s2 s4)) favours xyxy. Similarly, a good alignment contains many highly probable tuples which leads to a short message length for the data.

Yee and Allison(1993) showed that in order to obtain an unbiased estimate of the evolutionary "distance" between 2 strings it is necessary to use a weighted average of estimates from *all* alignments whereas the use of a single optimal alignment gives a biased estimate. That average can be computed in a few steps, each step taking time proportional to the product of the string lengths. This is feasible for 2 strings but not more unless the strings are short. In this paper, a stochastic process is used to average over not all but many alignments so as to get the estimates of the distances between the strings in an acceptable time. This is an example of Gibbs sampling or a Monte-Carlo method (Hastings 1970). Random alignments are generated from the posterior probability distribution of alignments. When used in conjunction with simulated annealing this also gives a stochastic search process for a good alignment.

There is important prior work in the treatment of alignment and evolutionary trees as inductive inference problems. Bishop and Thompson (1986) first cast pairwise-alignment as a maximum-likelihood problem. Thorne et al(1991, 1992) extended the maximum likelihood method to more general models of evolution. Felsenstein(1981, 1983) treated the inference of an evolutionary tree as a maximum likelihood problem, given a fixed multiple alignment. These are part of an important trend to make models of evolution explicit for better scrutiny and to place subsequent inferences on a sound statistical footing.

```
string A = TAATACTCGGC
string B = TATAACTGCCG

mutation instructions:
   copy
   change(ch) NB. ch differs from the corr' char in string A.
   del
   ins(ch)

a mutation sequence:
   copy; copy; delete; copy; copy; ins(A); copy; copy;
   del; copy; change(C); copy; ins(G)

generation instructions:
   match(ch)
   mismatch(ch_A,ch_B) NB. ch_A≠ch_B
   ins_A(ch_A)
   ins_B(ch_B)

a generation sequence:
   match(T); match(A); ins_A(A); match(T); match(A); ins_B(A); match(C);
   match(T); ins_A(C); match(G); mismatch(G,C); match(C); ins_BG

equivalent alignment:
   TAATA-CTCGGC-
   || || || | |
   TA-TAACT-GCCG
```

Figure 1: Basic Models.

2. Models of Evolution.

We model the evolution of a parent string A into a child string B by a finite-state *mutation-machine*. Such a machine can *copy* a character, *change* a character, *insert* a character or *delete* a character. Inserts and deletes are collectively called *indels*. The machine reads A and a sequence of mutation instructions and produces B.

If we wish to consider A and B to be of equal standing, we model their relation under a finite-state *generation-machine*. Such a machine can generate the same character in A and in B (match), generate different characters in A and in B (mismatch), generate a character in A only (ins_A) or generate a character in B only (ins_B). The machine reads a sequence of generation instructions and produces A and B.

Traditionally, an *alignment* is used to show a hypothetical relationship between 2 strings. One or both strings are padded out with *null* characters '-' until they have the same lengths and are then written out one above another. The two characters in a column are called a *pair* or a *(2-)tuple*. The all-null tuple is not allowed. It is more convenient to write a pair as <x,y> or just 'xy' in text.

There is an obvious correspondence between alignments, sequences of generation instructions and sequences of mutation instructions (figure 1) and they are used interchangeably as convenient. The notions can also be extended to more than 2 strings in an obvious way. Given strings A, B and a finite-state

```
              P          -- hypothetical, unknown parent
            .   .
         mA.       .mB   -- mutation machines
         .           .
       .               .
      A                 B -- given descendants

          m = mA.mB      -- equivalent generation machine

m's instns                Explanation &
pair (a,b)                   Rates.
-------------  --------------------------------------------------
match(x)       x<---x--->x      x<---y--->x
x x            pA'(c).pB'(c)    pA'(ch).pB'(ch)/3

mismatch(x,y)  x<---x--->y      x<---y--->y      x<---z--->y
x y            pA'(c).pB'(ch)   pA'(ch).pB'(c)   pA'(ch).pB'(ch)2/3

ins_A(x)       x<---x--->_      x<---y--->_      x<---_--->_
x _            pA'(c).pB'(d)    pA'(ch).pB'(d)   pA'(i)

ins_B(x)       _<---x--->x      _<---y--->x      _<---_--->x
_ x            pA'(d).pB'(c)    pA'(d).pB'(ch)   pB'(i)

               _<---x--->_      "invisible",
_ _            pA'(d).pB'(d)    not seen

c-copy, ch-change, i-insert, d-delete
Define instn rates per character of P:
   pA'(instn) = pA(instn)/(1-pA(i)); pB'(instn) = pB(instn)/(1-pB(i))
Note pA'(c)+pA'(ch)+pA'(d) = pB'(c)+pB'(ch)+pB'(d) = 1
Sum rates = (1+pA'(i)+pB'(i)) = (1-pA(i).pB(i))/((1-pA(i)).(1-pB(i)))
```

Figure 2: Composition of Two Machines, for DNA.

generation-machine (say), the probability of a sequence of instructions that produces A and B is the product of their individual probabilities and the message length of the sequence of instructions is the sum of their individual message lengths. This depends on P(match), P(mismatch), P(ins_A) and P(ins_B) which are called the *parameters* of the machine. The parameters might be given apriori, but generally they must be estimated from the strings. They correspond to the evolutionary "distance" of the strings. Note that an estimate can be got by counting instructions in one or more instruction sequences. The parameters consist of several values; a machine has more than one degree of freedom. If it is necessary to reduce them to a single number representing, say, the number of years since divergence, then a *rates-model* could be fitted to them in the style of Thorne et al(1991, 1992). Strictly, the probability of an instruction sequence should include a factor for the probability of the sequence's length but the probabilities of all plausible lengths are similar and, as we are only interested in the *relative* probabilities of instruction sequences in what follows, the length can be ignored.

The advantage of using finite-state machines is that there are standard techniques for manipulating them. The composition of two or more such machines is another one (figure 2).

An instruction sequence having the shortest message length can be found by the well-known dynamic programming algorithm (DPA). In general there are many optimal and near-optimal instruction sequences (or alignments). An estimate, E, of the machine parameters is a hypothesis. The best estimate minimises ML(E)+ML(A&B|E). The parameters are stated to some finite accuracy. ML(A&B|E) is not the message length of any one instruction sequence, it is the $-\log_2$ of the sum of the probabilities of *all* possible sequences or alignments for A and B. The hypothesis 'A&B' is that A and B are related by the inferred machine in some unspecified way and it is called the *r-theory*, the 'r' standing for related. Its message length can be calculated by a modified DPA which uses 'logplus' instead of 'min' in its general step (figure 3). Note that two different alignments represent exclusive hypotheses. Full details are given in Allison et al (1992a).

In what follows we consider only the simplest model of evolution, the 1-state model, where the probabilities of instructions are independent of context. In particular, linear gap (indel) costs require at least a 3-state model. It is also assumed below that P(insert)=P(delete) and that all characters are equally probable, all inserts are equally probable and all changes are equally probable. In that case, we have for

```
Boundary Conditions, i=1..|A|, j=1..|B|:
   D[0,0] = 0
   D[i,0] = D[i-1,0]+f(A[i], '-')
   D[0,j] = D[0,j-1]+f('-', B[j])

General Step, i=1..|A|, j=1..|B|:
   D[i,j] = g( D[i-1,j-1]+f(A[i],B[j]),    -- match or mismatch
               D[i-1,j  ]+f(A[i], '-'),    -- insA A[i]
               D[i,  j-1]+f('-', B[j]) )   -- insB B[j]

For Edit Distance:
   f(x,x) = 0; f(x,y) = f(x,'-') = f('-',y) = 1
   g(a,b,c) = min(a,b,c)

For r-theory:
   f(x,x)  =-log2(P(match(x))); f(x,y)  =-log2(P(mismatch(x,y)));
   f(x,'-')=-log2(P(del(x)));   f('-',y)=-log2(P(ins(y)))
   g(a,b,c)= logplus(a,logplus(b,c))
      where logplus(-log2(p),-log2(q)) = -log2(p+q)
```

Figure 3: The Dynamic Programming Algorithm (DPA).

DNA and the generation machine:

$$\mathrm{ML(match(x))} = -\log_2(\mathrm{P(match)})+\log_2(4) = -\log_2(\mathrm{P(match)})+2$$
$$\mathrm{ML(mismatch(x,y))} = -\log_2(\mathrm{P(mismatch)})+\log_2(12)$$
$$\mathrm{ML(ins_A(x))} = \mathrm{ML(ins_B(x))} = -\log_2(\mathrm{P(ins_A)})+2 = -\log_2(\mathrm{P(ins_B)})+2$$

These assumptions are made in the interests of simplicity as we are primarily interested in the underlying algorithms and methods. The model is simple but is not unacceptable for DNA. It is in any case the indels that are the main source of algorithmic complexity in multiple alignment.

3. The Posterior Distribution of Alignments.

The stochastic processes to be described require a method of sampling alignments (or machine instruction sequences), given 2 strings A and B, with the correct frequency as determined by the posterior probability distribution of alignments. A sampling procedure which achieves this aim is described below. It strongly resembles Hirschberg's (1975) linear-space version of the DPA and uses a recursive divide-and-conquer technique. Informally, it first chooses a point in B to correspond to the half-way point of A in the random alignment being generated. Second-level recursive calls then choose points in B to correspond to the quarter-points of A; these are conditional on the halfway point previously chosen. Third-level calls choose the one eighth points and so on.

Suppose we are given sections A[k..m] of A where m>k and B[L..n] of B. Let i be the middle position (k+m)div 2 of A[k..m]. Consider an arbitrary alignment of A[k..m] and B[L..n]. A[i] appears somewhere in the alignment. It either occurs alone as <A[i],-> which is equivalent to delete(A[i]) or as <A[i],B[j]> for some j which is equivalent to either copy(A[i]) or change(A[i],B[j]). In either case let j be the index of the last character of B[L..n] that occurs with or before A[i] and let j be L-1 if there is no such character. There is a certain probability distribution over the possible values for j. The probability of a particular value 'j' is the sum of the probabilities of all events that involve its choice ie. events (i) and (ii) below:

```
(i)  A[k..i-1]&B[L..j];    <A[i],  -  >; A[i+1..m]&B[j+1..n]
                       ... del A[i] ...
     for some j, L-1<=j<=n
(ii) A[k..i-1]&B[L..j-1];  <A[i], B[j]>; A[i+1..m]&B[j+1..n]
                             copy/
                     ... chng A[i]toB[j] ...
     for some j, L<=j<=n
```

Divide-and-Conquer Cases of A[k..m]&B[L..n], m>k, n>=L.

Recall that A[p..q]&B[r..s] is the hypothesis that those strings are related in some unspecified way and also that P(A[p..q]&B[r..s]) can be calculated for all 's' by the modified (logplus) DPA given previously in figure 3. Therefore the probabilities of each possible value of j can be calculated with two calls to the DPA: one on the forward strings A[k..i]&B[L..n] and one on the reversed strings A[i+1..m]&B[L..n]. This allows j to be sampled from its correct probability distribution. The sampling procedure is then called recursively on A[k..i]&B[L..j] and on A[i+1..m]&B[j+1..n].

The coordinates (i,j) are called an *internal terminal*. The coordinates (|A|,|B|) are called the *external terminal*. Since the first half of the overall alignment was forced to end with either <A[i],-> or <A[i],B[j]> and not <-,B[j]>, any subsequent reversed DPAs are forced to begin from an internal terminal with one of these alternatives by simple modification of the boundary conditions.

The lengths of the sections of A and B shrink with each recursive call to the sampling procedure. Eventually a single character of A remains and this leads to the *base cases* of the procedure. If the call is for an internal terminal, A[i] must occur at the end of the mini-alignment. There are two possibilities:

```
(i)    -    ...   -   A[i]
     B[j] ... B[k]  -
     ins  ... ins  del

(ii)  -    ...   -     A[i]
     B[j] ... B[k-1] B[k]
                     copy/
     ins  ... ins    change
```

Internal-Terminal Base-Cases of A[i]&B[j..k], k>=j.

The probability of each possibility is easily calculated and they are sampled at random. If the base case is for the external terminal there are more possibilities but each is no more complex than before and they are easily sampled:

```
(i)  A[i]  -    ...  -      or   -    A[i]  -      ...  -     etc.
      -   B[j] ... B[k]         B[j]  -    B[j+1] ... B[k]
     del  ins  ... ins          ins  del   ins     ... ins

(ii) A[i]  -      ...  -    or    -   A[i]    -      ...  -     etc.
     B[j] B[j+1] ... B[k]       B[j] B[j+1] B[j+2] ... B[k]
     copy/                           copy/
     chng ins    ... ins        ins  chng   ins    ... ins
```

External-Terminal Base-Cases of A[i]&B[j..k].

The alignment sampling procedure is just twice as slow as the DPA that it uses, ie. it takes O(|A|×|B|) time. This is because it solves one full-size (|A|×|B|) DPA problem, two quarter-size problems, four sixteenth-size problems etc. In Hirschberg's algorithm the divide-and-conquer technique was used to reduce the space used to O(|B|) - as it still does here - because only two rows of a matrix of length |B| are required to calculate the values needed.

4. K>=2 Strings.

It is infeasible to extend the DPA directly to K>2 strings. However it can be extended so as to align two alignments. A string can be considered to be a trivial case of an alignment. Given an alignment AS over a set of strings S and an alignment AT over another set of strings T, AS and AT can be aligned to give an alignment of S∪T. The result may not be optimal overall but this process can be used as an iterative step to improve a multiple alignment. This kind of *deterministic heuristic* is quite common and an example has been described in Allison et al(1992b): Given K>2 strings and an evolutionary tree over them an initial multiple alignment is found by some suboptimal process. The tree is then "broken" on some edge which partitions the K strings into two disjoint sets, S and T. The complete multiple alignment implies an alignment over S and another over T in a natural way and it is *projected* onto these two sets of strings to give two sub-alignments which are re-aligned against each other by the DPA to give a new complete multiple alignment. The process is iterated and terminates when there is no further improvement in the multiple alignment. The results are usually good although it is not guaranteed to give an optimal multiple alignment as it may get stuck in local optima and results may depend on the order of breaks in the tree. In a following section the alignment sampling process provides a way around these and other problems.

We model each edge of the evolutionary tree by a separate mutation-machine. The machines can be combined so as to calculate the probability (actually message length) of each K-tuple as it is met during the DPA (cf. figure 3). Felsenstein(1981, 1983) gives the necessary calculations. They involve calculating probabilities for each possible character value at internal nodes of the tree (figure 4). The probabilities of

```
Actual machine parameters:
     copy change indel
m1:  0.9  0.05   0.05
m2:  0.9  0.08   0.02
m3:  0.7  0.2    0.1
m4:  0.8  0.1    0.1
m5:  0.75 0.1    0.15

A                                A
 .                              .
m1.                            .m3
   . A:0.43          A:0.40 .
    .C:0.56          C:0.59.
     +---------------------+
    .         m5          .
   .                       .
m2.                            .m4
 .                              .
C                                C

Probable hypothetical characters.
Tuple ACAC's probability = 0.00027

Estimated operations carried out:
     P(copy) P(change) P(indel)
m1:  0.43    0.57      <0.01
m2:  0.57    0.43      <0.01
m3:  0.40    0.60      <0.01
m4:  0.59    0.41      <0.01
m5:  0.94    0.06      <0.01
```

Figure 4: Example, explanations for tuple ACAC.

the various instructions for a tuple can also be calculated and used to estimate instruction probabilities for each machine from a multiple alignment. All this can be done in an O(K) time traversal of the tree for each tuple but it is a significant computation so results are stored in a lookup-table for fast access if the tuple is met again, as it probably will be. The lookup-table speeds up the algorithm considerably. Two alignments of 4 strings of length 500 can be aligned, to give an 8-way alignment, in 15 seconds on a sparc-station.

5. Sampling Multiple Alignments.

It is infeasible to extend the alignment sampling procedure directly to K>2 strings and a given evolutionary tree for the same algorithmic reasons that the DPA cannot be directly extended. However, it is sufficient to hold many of the degrees of freedom of "the system", which is here a multiple alignment, fixed while sampling the remainder from the conditional posterior distribution so that is what is done: Given a multiple alignment, the tree is "broken" on a random edge which partitions the strings into two disjoint sets. The multiple alignment is projected onto the two sets to give two subalignments. A *random* realignment is sampled from the posterior distribution of alignments (of the subalignments) as described for just 2 strings. The process is iterated many times. The machine parameters are estimated for each multiple alignment sampled. Results from all samples are averaged to give the final estimate of the machine parameters. Standard deviations are also calculated and give an indication of the accuracy of estimates. A working-estimate of the machine parameters is needed to calculate the distributions in the algorithm and a weighted average from "recent" alignments is used for this purpose. To begin the sampling, an initial multiple alignment is found by the deterministic heuristic described previously.

6. Simulated Annealing.

The alignment sampling procedure is trivially modified for use in a simulated annealing approach to optimal multiple alignment. At the heart of the sampling procedure, a point ‘j’ of string B is found to correspond to the midpoint ‘i’ of string A. The point j is sampled from the probability distribution of possible values. If the values of the probability distribution are raised to some power p>=0, and the distribution is renormalised, the sampling of j and of alignments is modified in a useful way. When p=0 j is chosen with a uniform distribution. When p=1 j is chosen from the posterior distribution. When p>1 j is biased towards the more probable values. When p is very large j is chosen so as to lead to an optimal (pairwise) alignment. Increasing p amounts to lowering the temperature in simulated annealing. Since the algorithm actually uses message lengths, the message lengths are multiplied by p which is equivalent to raising the probabilities to the power p.

```
               s8                            s15
               .                             .
s9             .                             .             s14
   .           . e6                     e13  .           .
     .         .                             .         .
   e7  .       .                             .       . e12
         .     .                             .     .
           .   .                             .   .
              s4                             s7
               .                             .
                .                           .
              e2 .                         . e5
                  .                       .
                   .         e1          .
                    s2.............s1=s3
                   .                     .
                  .                       .
                 . e3                      . e4
                .                           .
               .                             .
              s5                             s6
           .   .                             .   .
         .     .                             .     .
   e8  .       .                             .       . e11
     .         .                             .         .
   .           . e9                          . e10       .
s10            .                             .             s13
               .                             .
               s11                           s12
```

Ancestor, s1≡s3, length(s1)=500
1st generation: s2 and s3
2nd generation: s4 to s7
3rd generation: s8 to s15
Probabilities for each edge during evolution:
P(copy)=.9; P(change)=.05; P(insert)=P(delete)=.025

Figure 5: Full Tree.

This strategy is very different in action from the simulated annealing strategy of Ishikawa et al(1992), which makes small perturbations to a multiple alignment. The present strategy can make large perturbations, especially when p is small.

7. Results.

A program was written containing the alignment sampling procedure and the simulated annealing method described above. It was first tested on artificially generated DNA data. Figure 5 shows the artificial evolution of three generations of strings. The original ancestor, s1, is a random string of 500 characters. The first-generation descendants are s2 and s3; s3 is identical to s1 because we are dealing with unrooted trees and wish to make all edges similar in this instance. Every edge in the tree corresponds to a mutation-machine with the following parameters: P(copy)=0.9, P(change)=0.05; P(insert)=P(delete)=0.025. One expects something like 10% mutation from parent to child but it could be more or less as the evolution process is random. One expects of the order of 20% mutation between s8 and s9 say, 30% mutation between s4 and s6, 40% mutation between s8 and s10 and 50% mutation between s8 and s12. Note that alignments can be found for s8 and s12 with more than 50% matches (their edit-distance is 189 not 250) and that an alignment with 60% to 70% matches can often be found even for two random, unrelated DNA sequences of similar length.

In three separate trials, first, second and third-generation strings were used as data sets - each with the correct evolutionary tree. The tree for trial 3 includes that for trial 2 which includes the trivial tree for trial 1. Three analyses were performed in each trial. First, the deterministic heuristic was used to find a good alignment and parameters were estimated from this alone. The alignment was used as a starting point for the next two analyses. Second, parameters were estimated from 1000 stochastically sampled alignments. Third, simulated annealing was used over 1000 trials with the message length multiplier increasing linearly from 1.0 to 5.0.

The results of these trials are summarised in Table 1. The lines marked *actual* give information from the evolution of the strings which is unknown to the analysis program. During evolution the machine on each edge had parameters 0.9, 0.05, 0.025, 0.025. The mutations for an edge would therefore be expected to consist of about 25 changes, 12.5 inserts, 12.5 deletes, but there is variation and so the actual numbers are given. The edit-distance (ED) of the 2 strings at the ends of an edge is also recorded and is generally less than the sum of the mutations that took place as a better alignment than the "real" one can usually be found.

Each line marked *det* gives the parameter estimates from the putative optimal alignment found by the deterministic heuristic. (There is of course no feasible algorithm guaranteed to produce an optimal alignment.) It tends to overestimate P(copy) and, with the exception of e7 and e10, to underestimate P(indel). This is consistent with results on two strings (Yee & Allison 1993).

Each line marked *Gibbs* gives the estimates from the 1000 stochastically sampled alignments. (The standard deviation of the estimate of indels is the largest and is the only standard deviation reproduced.) The actual proportion of indels on each edge lies within 1.5 standard deviations of the estimate of P(indel). Note that the standard deviation of the estimates for e1, which is common to each trial, increases as the data gets farther from it, as seems reasonable, roughly doubling with each extra generation.

The deterministic heuristic proved hard to beat at this level of mutation. It was not beaten by simulated annealing in any of the above trials although simulated annealing found many alignments with a message length just 2 bits more than that found by the heuristic. From other trials it seems that 4-way alignment is rather easy in that the heuristic was not beaten in several trials. On 8-way alignment with 10% mutation per edge, the heuristic was sometimes beaten, but never by more than a few bits on artificial data. It seems likely that there is a very large number indeed of near optimal alignments and that the search-space is not informative close to them. Extremely time consuming tests would need to be done to map out the alignment landscape. The search for the marginally "best" alignment may be rather pointless in any case.

Simulated annealing beat the deterministic heuristic by a significant 34 bits when the level of mutation was increased to 15% per edge. Table 2 gives results for a tree with the topology of figure 5

```
            1st generation        2nd generation        3rd generation
      data: s2 & s3               s4 to s7              s8 to s15
      ----- --------------------- --------------------- --------------------
e1  actual: 19 ch 16i 19d (35 indel)  ED=51
       det: .900 .040 .061        .894 .044 .063        .878 .060 .062
     Gibbs: .893 .038 .070(.005)  .878 .038 .085(.009)  .863 .043 .094(.016)

e2  actual: 27ch 15i 8d (23 indel)    ED=49
       det:                       .910 .054 .036        .928 .048 .025
     Gibbs:                       .899 .052 .049(.007)  .900 .054 .046(.007)
e3  actual: 21ch 12i 8d (20 indel)    ED=38
       det:                       .916 .057 .026        .915 .061 .025
     Gibbs:                       .917 .048 .035(.005)  .920 .048 .033(.008)
e4  actual: 28ch 9i 15d (24 indel)    ED=52
       det:                       .899 .059 .043        .920 .038 .043
     Gibbs:                       .891 .054 .056(.007)  .906 .036 .058(.009)
e5  actual: 21ch 13i 13d (26 indel)   ED=45
       det:                       .915 .055 .030        .913 .055 .032
     Gibbs:                       .908 .052 .041(.006)  .904 .061 .035(.008)

e6  actual: 30ch 10i 16d (26 indel)   ED=55
       det:                                             .896 .066 .038
     Gibbs:                                             .893 .062 .045(.006)
e7  actual: 18ch 11i 9d (20 indel)    ED=37
       det:                                             .924 .035 .042
     Gibbs:                                             .918 .031 .052(.007)
e8  actual: 15ch 13i 12d (25 indel)   ED=37
       det:                                             .937 .037 .026
     Gibbs:                                             .932 .026 .041(.006)
e9  actual: 18ch 17i 18d (35 indel)   ED=50
       det:                                             .909 .042 .050
     Gibbs:                                             .895 .032 .073(.009)
e10 actual: 30ch 10i 14d (24 indel)   ED=51
       det:                                             .898 .048 .055
     Gibbs:                                             .879 .052 .069(.009)
e11 actual: 25ch 16i 9d (25 indel)    ED=47
       det:                                             .907 .057 .036
     Gibbs:                                             .906 .053 .042(.007)
e12 actual: 25ch 11i 16d (27 indel)   ED=47
       det:                                             .909 .045 .046
     Gibbs:                                             .902 .040 .059(.008)
e13 actual: 25ch 11i 7d (18 indel)    ED=43
       det:                                             .912 .064 .024
     Gibbs:                                             .906 .064 .030(.007)

Message Lengths (bits):
       det: 1409                  2792                  5444
      Null: 2019                  4038                  8058
     Gibbs: 1432(+-12.3)          2994(+-32)            5872(+-75) mean(SD)

Evolution: Ancestor length = 500
   P(copy)=.9; P(change)=0.05; P(insert)=P(delete)=0.025 for each edge

key: actual #changes #inserts #deletes (#indel)   ED=edit distance
     det  =P(copy) P(change) P(indel) estimates from deterministic heuristic
     Gibbs=P(copy) P(change) P(indel)(Std Dev) est' from Gibbs sampling 1000x
```

Table 1: Estimated Edges from Evolution at 10% Mutation/Edge.

```
            det            SA             Gibbs
            -------------- -------------- --------------------
e1  actual: 29ch 21i 24d (45 indel) ED=68
            .854 .089 .057 .876 .073 .051 .850 .073 .078(.015)

e2  actual: 38ch 25i 16d (41 indel) ED=74
            .862 .093 .046 .869 .080 .051 .838 .067 .095(.017)
e3  actual: 35ch 14i 12d (26 indel) ED=58
            .920 .050 .030 .908 .069 .024 .887 .062 .052(.012)
e4  actual: 44ch 16i 23d (39 indel) ED=77
            .863 .103 .034 .864 .119 .018 .836 .107 .058(.014)
e5  actual: 41ch 21i 25d (46 indel) ED=82
            .845 .133 .022 .833 .160 .008 .836 .089 .075(.019)

e6  actual: 33ch 22i 18d (40 indel) ED=70
            .877 .059 .064 .875 .041 .085 .850 .047 .103(.011)
e7  actual: 36ch 19i 13d (32 indel) ED=66
            .880 .079 .041 .876 .093 .031 .877 .084 .040(.009)
e8  actual: 25ch 14i 20d (34 indel) ED=57
            .890 .062 .049 .896 .053 .051 .890 .053 .057(.010)
e9  actual: 43ch 22i 23d (45 indel) ED= 84
            .841 .098 .062 .838 .091 .072 .813 .079 .108(.012)
e10 actual: 34ch 19i 12d (31 indel) ED=64
            .895 .045 .060 .880 .047 .074 .877 .052 .071(.009)
e11 actual: 45ch 16i 22d (38 indel) ED=75
            .851 .090 .059 .860 .074 .065 .836 .075 .089(.013)
e12 actual: 40ch 18i 16d (34 indel) ED=72
            .876 .083 .041 .889 .078 .033 .861 .090 .049(.008)
e13 actual: 43ch 16i 21d (37 indel) ED=74
            .866 .083 .051 .854 .079 .066 .846 .075 .079(.011)

Message Lengths (bits):
            6735           6701           7556(+-109) mean(SD)
Null: 8045

Evolution: Ancestor length = 500
   P(copy)=.85; P(change)=.075; P(insert)=P(delete)=.0375 for each edge

key: actual #changes #insert #delete (#indel)  ED=edit distance
     P(copy) P(change) P(indel) from deterministic heuristic,
     simulated annealing & Gibbs sampling 1000x
```

Table 2: Estimated Edges from Evolution at 15% Mutation/Edge.

where the machine for each edge had the following parameters: P(copy)=0.85, P(change)=0.075, P(insert)= P(delete)=0.0375. Strings such as s8 and s12 are tenuously related here. Estimates are given from single alignments by the deterministic heuristic (det) and by simulated annealing (SA) and from stochastic sampling of 1000 alignments (Gibbs). There is an increased tendency for (near) optimal alignments to underestimate P(indel), and this effect is most marked on "internal" edges of the tree, see e5 for example. A similar effect has been noted with algorithms under development for the maximum-parsimony alignment problem. The standard deviations in the parameter estimates from the sampling process increase with the greater level of mutation, as is to be expected.

The above examples might be easy because the tree is uniform. To investigate this, s13 and s15 were deleted from the last data set. This left s12 connected to s3 by a "long" edge denoted ‘e4+e10’ and s14 connected to s3 by an edge denoted ‘e5+e12’. Using figure 2 to combine the two machines that correspond to e4+e10, say, during evolution we predict P(copy)=0.726, P(change)=0.131, P(insert)= P(delete)=0.071. The estimate from Gibbs sampling is close to these figures; see table 3. The near optimal alignments from the deterministic heuristic and from simulated annealing significantly overestimate P(copy) and

```
        det             SA              Gibbs
        --------------  --------------  --------------------
e1    : .877 .075 .048  .868 .103 .030  .840 .050 .110(.024)

e2    : .864 .082 .054  .865 .076 .060  .844 .075 .081(.015)
e3    : .917 .049 .034  .930 .030 .040  .889 .038 .073(.014)

e4+e10: .789 .140 .071  .796 .108 .095  .729 .120 .152(.022)
e5+e12: .744 .192 .064  .731 .230 .039  .693 .149 .158(.031)
     (predicted .726 .131 .142)

e6    : .867 .062 .072  .872 .051 .078  .848 .050 .103(.010)
e7    : .891 .076 .033  .882 .083 .035  .877 .081 .041(.009)
e8    : .896 .059 .045  .889 .052 .058  .882 .053 .065(.009)
e9    : .835 .099 .065  .840 .101 .060  .817 .086 .098(.010)

Message Lengths (bits):
        5391            5375            6278(144) mean(SD)
        Null: 6076

Ancestor Length = 500

key: P(copy) P(change) P(indel) from deterministic heuristic,
     simulated annealing and Gibbs sampling 1000x
```

Table 3: Edge Estimates for s8...s12&s14, 15% Mutation/Edge.

underestimate P(indel) for these edges. Counting the actual numbers of mutations during evolution, as given in table 2, confirms this conclusion. The standard deviation for the estimates from sampling is large but a single alignment gives no hint of this uncertainty. Note that simulated annealing bettered the heuristic by 16 bits in message length. S12 and s14 are only distantly related to the other strings and an "average" alignment for the six strings has a message length worse than the null theory.

```
(a) from 10% mutation/edge tree:
        det              SA                Gibbs
        --------------   --------------    --------------------
   est: .619 .288 .093   .619 .288 .093    .547 .188 .266(.035)
   (predicted: .604 .179 .217;  ED=189)

   Message lengths (bits):
        2002             2002              2277(49) mean(SD)
        Null: 2001

(b) from 15% mutation/edge tree:
        det              SA                Gibbs
        --------------   --------------    --------------------
   est: .538 .370 .091   .534 .379 .088    .409 .141 .450(.098)
   (predicted: .468 .230 .302;  ED=222)

   Message Lengths (bits):
        2181             2144              2662(107) mean(SD)
        Null: 2045

Key: P(copy) P(change) P(indel) for deterministic heuristic,
     simulated annealing and Gibbs sampling 1000x.
```

Table 4: Estimates for s8 & s12 from Full-Trees at 10% & 15% Mutation/Edge.

S8 and s12 were compared as an example of a pair of distant relatives, both from the tree at 10% mutation per edge and from the tree at 15% mutation per edge, and the results are given in table 4. The deterministic heuristic and simulated annealing give alignments with message lengths greater than the null theory: They are not acceptable hypotheses and the parameter estimates, particularly for indels, are poor. Note that at the higher level of mutation the deterministic heuristic became stuck in a local optimum during the cycle of estimating from an alignment and then finding a new alignment, even though there were only two strings. Gibbs sampling gives the best estimates of the proportions of mismatches and indels although there is considerable uncertainty in those estimates.

Transthyretin is a protein expressed in the brain and, amongst other things, is of interest because of its relevance to the evolution of birds, reptiles and mammals (Schreiber et al 1992). The cds sequences for transthyretin (Duan et al 1991) from human, sheep, mouse, rat, chicken and reptile (T.rugosa) were obtained from Genbank. There is little doubt about the correct topology of the evolutionary tree, which is shown in figure 6 annotated with the estimates of the parameters for each edge obtained by averaging over 1000 sampled alignments.

Templeton et al(1992) have used mitochondrial DNA sequences to infer evolutionary relationships amongst human races as part of the debate about the origin of our species. They inferred trees from a given multiple alignment. The algorithms described here cannot handle the dozens of sequences used in that study but figure 7 shows edge estimates obtained by Gibbs sampling for one tree over four sequences.

```
                sheep             human
                   .               .
                     .           .
      (.911,.081,.008) .       .(.913,.086,.001)
                         .   .
                           +
                           .
                           .(.973,.026,.001)
                           +
                         .   .
       (.945,.054,.001).       .
                     .           .
                   .               .
mouse...........+                    .(.838,.136,.026(+-.003))
(.951,.041,.008) .                     .
                 .                       .
                 .(.945,.055,.001)         .
                 .                           .
                 .                             +.....................reptile
                rat                            .    (.853,.146,.001)
                                               .
                                               .(.923,.076,.001)
                                               .
                                               .
                                            chicken

key: (P(match),P(change),P(indel))
Genbank ids: HUMPALA(27-470),  OATTHYRE(12-452), MMALBR(27-467),
             RATPALTA(10-453), GDTRTHY(26-478),  TRTRANST(16-468).
```

Figure 6: Edge Estimates by Sampling for Transthyretin cds.

8. Conclusions.

A multiple alignment can give an estimate of the edges of an evolutionary tree. The use of a good or optimal alignment gives a biased estimate of the true values. Forming the weighted average of estimates from all alignments would give an unbiased estimate of the edges. This can be done efficiently for 2 strings but is infeasible for more than 2. However, averaging the estimates from many alignments sampled from their posterior probability distribution gives a good approximation and is feasible. In addition, sampling from the probability distribution raised to an increasing power (or from message lengths with an increasing multiplier) effects a simulated annealing search for an optimal alignment. A computer program embodying these ideas has been implemented for the simplest, 1-state, model of evolution. The implementation is practical for 8 to 10 strings of several hundred characters when used on a good work-station. With optimisation each limit could be increased somewhat. One temptation is to reduce the time-complexity of the underlying DPA from quadratic to near linear by a windowing technique under the assumption that most of the probability in alignment-space is concentrated near the current alignment. However this may be a trap because the assumption may be invalid, particularly if the strings contain repeated and transposed sections. The sampling method is certainly a good candidate for a parallel implementation.

It would be useful to be able to handle many more than 10 strings. To do this it will probably be necessary to use a method, related to the one described here, but which samples definite strings for the internal nodes of the tree in a stochastic manner. (The current method only makes implicit probabilistic assignments to internal nodes.) The problem can then be treated as an iterated 3-string problem. Each step will then be relatively simple but many more of them will probably be required to explore the search-space adequately. The resulting program would be a stochastic version of a method proposed by Sankoff et al(1976).

```
1                                       3
 .                                     .
  .                                   .
   .(.979,.020,.002)                 .(.989,.011,.001)
    .                               .
     .      (.983,.006,.011)       .
      +...........................+
     .                             .
    .                               .
   .(.912,.012,.077(+-.005))         .(.975,.012,.013)
  .                                   .
 .                                     .
2                                       4


key: (P(copy) P(change) P(indel))
Sequences:
 1 HUMMTDL006 717 bp Human (Eastern Pygmie) Mitochondrial DNA control region
 2 HUMMTDL016 668 bp Human (!Kung) Mitochondrial DNA control region
 3 HUMMTDL026 721 bp Human (Yoruban) Mitochondrial DNA control region
 4 HUMMTDL046 730 bp Human (Western Pygmy) Mitochondrial DNA control region
```

Figure 7: Edge Estimates by Sampling for 4 Human Mitochondrial Sequences.

9. References.

Allison L., C. S. Wallace & C. N. Yee. (1992a) Finite-state models in the alignment of macro-molecules. J. Mol. Evol. **35** 77-89.

Allison L., C. S. Wallace & C. N. Yee. (1992b) Minimum message length encoding, evolutionary trees and multiple alignment. 25th Hawaii Int. Conf. Sys. Sci. **1** 663-674.

Bishop M. J., A. E. Friday & E. A. Thompson. (1987) Inference of evolutionary relationships. In *Nucleic Acid and Protein Sequence Analysis*. M. J. Bishop & C. J. Rawlings (eds), IRL Press, Oxford 359-385.

Duan W., M. G. Achen, S. J. Richardson, M. C. Lawrence, R. E. H. Wettenhall, A. Jaworowski. & G. Schreiber. (1991) Isolation, characterisation, cDNA cloning and gene expression of an avian transthyretin: implications for the evolution of structure and function of transthyretin in vertebrates Eur. J. Biochem. **200** 679-687

Felsenstein J. (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Mol. Evol. **17** 368-376.

Felsenstein J. (1983) Inferring evolutionary trees from DNA sequences. In *Statistical Analysis of DNA Sequence Data* B. S. Weir (ed), Marcel Dekker 133-150.

Hastings W. K. (1970) Monte Carlo sampling methods using Markov chains and their applications. Biometrika **57** 97-109.

Hirschberg D. S. (1975) A linear space algorithm for computing maximal common subsequences. Comm. A.C.M. **18**(6) 341-343.

Ishikawa M., T. Toya, M. Hoshida, K. Nitta, A. Ogiwara & M. Kanehisa. (1992) Multiple sequence alignment by parallel simulated annealing. Institute for New Generation Computing (ICOT) TR-730

Sankoff D., R. J. Cedergren & G. Lapalme. (1976) Frequency of insertion-deletion, transversion, and transition in evolution of 5S ribosomal RNA. J. Mol. Evol. **7** 133-149.

Schreiber G., A. R. Aldred & W. Duan. (1992) Choroid plexus, brain protein-homeostasis and evolution. Today's Life Science Sept. 22-28.

Templeton A. R., S. Blair Hedges, S. Kumar, K. Tamura & M. Stoneking. (1992) Human origins and analysis of mitochondrial DNA sequences. Science **255** 737-739.

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

Thorne J. L., H. Kishino & J. Felsenstein. (1992) Inching towards reality: an improved likelihood model of sequence evolution. J. Mol. Evol. **34** 3-16.

Wallace C. S. & P. R. Freeman. (1987) Estimation and inference by compact encoding. J. Royal Stat. Soc. B **49** 240-265.

Yee C. N. & L. Allison. (1993) Reconstruction of strings past. Comp. Appl. BioSci. **9**(1) 1-7.

PS. A related paper now appears as:
L. Allison & C. S. Wallace, An Information Measure for the String to String Correction Problem with Applications, 17th Australian Computer Science Conference (ACSC-17), isbn-0-473-02313-X, University of Canterbury, Christchurch, New Zealand 19-21 January 1994; Australian Computer Science Communications **16** (!) pp659−668 1994.