-

Minimum Message Length Encoding, Evolutionary Trees and Multiple-Alignment.

L. Allison, C. S. Wallace and C. N. Yee,
Department of Computer Science,
Monash University,
Australia 3168.

xxx@cs.monash.edu.au where xxx = lloyd, csw, cyee
Tel: (03) 565 5205
Fax: +61 3 565 5146

Technical Report 91/155
September 1991

A version of this paper now appears in HICSS-25: Hawaii International Conference on System Sciences, V1, p663-674, Jan 1992.

Abstract. A method of Bayesian inference known as minimum message length (MML) encoding is applied to problems over k≥2 mutating strings: inference of evolutionary-trees and multiple-alignment. The method allows the posterior odds-ratio of two competing hypotheses, for example two trees, to be calculated. A tree that is a good hypothesis for the strings forms the basis of a short message describing them. The work is a natural extension of applying MML encoding to the alignment of two strings as previously reported. It is seen that tree inference and multiple-alignment are intimately connected.

The mutation of one string into another is modelled by a finite-state machine. The machine can copy, change, insert and delete characters. Each machine has one or more parameters. Varying the machine architecture varies the mutation cost-functions.

The tree inference problem is broken down into inter-linked sub-problems: what is the best topology, what are the optimum parameter values for each edge of the tree, what is the implied cost function on character-tuples in a multiple-alignment, what is the probability of each multiple-alignment and finally what is the probability of an optimal alignment or of all alignments taken together? Each edge of the tree relates two, possibly hypothetical, strings and is described by one machine. Methods of estimating the parameter values for each edge (machine) are described.

The length of a message describing the strings and based on a tree includes $-\log_2$ of the sum of the probabilities of all multiple-alignments as implied by the topology and parameters of the tree. Computing this over all alignments is not feasible; heuristics and approximate methods to circumvent this problem are described.

Q: What is the difference between a hypothesis and a theory?
A: Think of a hypothesis as a card. A theory is a house made of hypotheses.
(From rec.humor.funny, attributed to Marilyn vos Savant.)

Keywords: inductive inference, minimum message/ description length, multiple-alignment, evolutionary-tree, phylogenetic tree.

1. Introduction.

In previous papers[Alli1, Alli2], minimum message length (MML) encoding was used to infer the relation, if any, between two strings and to compare models of mutation or relation. Here we apply MML encoding to problems over **k**≥2 strings of average length **n**: multiple-alignment and inference of evolutionary-trees. First we introduce the MML method.

MML encoding is a method of Bayesian inference. In it, we consider the message length of an *explanation* of given data D. An explanation consists of a message which first states an hypothesis H, then states the data using a code which would be optimal were H true. It is a basic fact of coding theory that the message length (ML) under optimal encoding of an event of probability p is $-\log_2(p)$ bits. From Bayes' theorem in the discrete case, given a hypothesis H and data D, P(H&D) = P(H).P(D|H) = P(D).P(H|D) or equivalently ML(H&D) = ML(H)+ML(D|H) = ML(D)+ML(H|D). P(D|H) is called the *likelihood* of the hypothesis although it is a function of the data given the hypothesis; it is not the probability of the hypothesis. P(H) is the prior probability of the hypothesis. P(H|D) is the posterior probability of the hypothesis. If ML(H) and ML(D|H) can be calculated for given H and D then the posterior odds-ratio of two hypotheses can be calculated. Analogous results hold in the continuous case and imply an optimum accuracy for the estimation of real-valued parameters, although the calculations can be difficult. MML encoding can be used to compare models, test hypotheses and estimate parameters. Wallace and Freeman[Wall2] give the statistical foundations.

Maximum-likelihood methods and MML encoding are related. A significant difference is that the former do not cost the model or hypothesis complexity, ML(H), and so tend to favour overly complex or detailed models. In sequence analysis the issue covers: the estimation of parameters (such as time) to optimal accuracy, the complexity of models of mutation and of evolution and the fact that a single alignment is too precise a theory. In those cases where the model complexity is constant it can be ignored and then maximum-likelihood and MML encoding give the same results.

Two being a special case of k≥2, we next link inference over two strings to the more general problems. Given strings A and B, the following questions can be raised:

Q1: Are they related?
Q2: Are they related by a particular model of relation and parameter values (˜ time since divergence)?
Q3: Are they related by a particular alignment?
Q4: Are they related by descent from a particular common ancestor?

An answer to any one of these questions can "only" be a hypothesis. An explanation of the strings can be based on such an answer. Explanations based on successive questions carry increasing amounts of information and must have increasing message lengths. A and B have infinitely many possible common ancestors. It is possible to dismiss most as wildly improbable but it is generally impossible to identify a common ancestor with any certainty unless extra information is available. A single alignment implies a subset of possible ancestors in a natural way. Conventional alignment programs return a single alignment which is optimal under some explicit or implicit assumptions. A given model can relate A and B by many different alignments and we called the answer to Q2 the r-theory (for relation). Its statement embodies the inferred parameters for the evolution of A and B followed by the strings A and B in a code based on all possible alignments. One of the by-products is a probability density plot of all alignments under those parameter values. The answer to Q1 would include the sum over all models of the integral over all parameter values. This calculation is impractical and the result would be of little use; Q2 is the question usually asked.

There is a natural null-theory in sequence analysis. If the strings are unrelated, it is not possible to do better than transmit them one after the other, at about two bits per character for DNA. Note that the lengths of the strings must also be transmitted. Any explanation that has a message length greater than that of the null-theory is not acceptable.

The approach to problems over two strings, as summarised above, generalises to problems over several strings: Questions analogous to Q1-Q4 arise when given k≥2 strings which might be related by an evolutionary-tree:

R1 (a): Are the strings related or, slightly more detailed,
R1 (b): are they related by a particular tree-topology? Consideration is usually restricted to binary trees and we follow this convention here. There is no difficulty in allowing arbitrary branching factors. Forests should be allowed so as to permit independent families but the message length of a forest includes the sum of the message lengths of its trees and the generalisation is obvious.
R2: Are they related by a particular tree (topology + parameter values for each edge). Each edge of the tree represents a simple two-string relation; the inferred parameters for each edge (~ time) are required. This is the generalised r-theory.
R3: Are they related by a particular multiple alignment?
R4: Are they related by descent through particular ancestors?

The chain Q1:Q2:Q3:Q4 matches the chain R1:R2:R3:R4 and in particular Q2:Q3 matches R2:R3, although this requires some discussion. A particular tree (topology & edge parameters) implies a probability distribution on character-tuples and thus on multiple-alignments. The probability of a tree and the strings (R2) is the product of the prior probability of the tree and the sum of the probabilities of all such multiple-alignments given the tree. For k strings, where $k>2$, the number of alignments is very large. Therefore heuristic and approximate methods are required to make the computations feasible. R3 can be taken in two ways. We take the word 'related' to indicate 'related by an evolutionary-tree'. A tree has $2k-3$ edges and thus has $O(k)$ parameters but there are 5^k-1 possible tuples in a multiple alignment. There is a smaller but still exponential number of possible 'patterns'. Many of the tuples' probabilities are locked together under an evolutionary-tree hypothesis. If 'related' were taken to mean any arbitrary relation, there would be more degrees of freedom in the form of hypotheses which would enable the data to be fitted better but at the higher overhead, in general, of stating the increased number of parameters.

Reichert et al[Reic] were the first to apply compact encoding to the alignment of two strings. They used one particular model of relation and sought a single optimal alignment. This was difficult to do efficiently for the model that they chose. Bishop and Thompson[Bish] defined maximum-likelihood alignment for what is a 1-state model although they recognised the possibility of other models. Allison et al[Alli2] objectively compared 1, 3 and 5-state models of mutation using MML encoding. Felsenstein[Fels1, Fels2] developed a maximum-likelihood method for infering evolutionary-trees. To date, it is the method with the soundest statistical credentials. His program does not do alignment itself and requires a multiple-alignment as input. Friday[Frid] gives a general discussion of the problem of assigning figures of merit to inferred evolutionary-trees. Wallace and Boulton[Wall1, Boul2] developed an MML method for classification of arbitrary things. (It is not directly applicable to the alignment problem.) The things can have continuous and/or discrete-valued attributes. A further development[Boul3] inferred hierarchical classifications. Cheeseman and Kanefsky[Chee] applied MML encoding to evolutionary-trees. Milosavljevic[Milo] has examined the classification of macro-molecule sequences. (In classification, a class usually contains many things.)

The methods described in this paper relate to other methods as follows. They generalise Felsenstein's approach by allowing inserts and deletes to be inferred and by including the cost of the model. In contrast to Cheeseman and Kanefsky, ancestral strings are not inferred, for to do so involves arbitrary choices. However probabilities can be assigned to possible ancestors and to possible alignments given the parameters that are inferred.

2. Mutation-Models.

We consider two broad models to relate two strings A and B (**Figure 1**). In the *mutation-model* a simple machine reads string A and a sequence of mutation instructions and outputs string B. In the *generation-model* a simple machine reads a sequence of generation instructions and outputs strings A and B on a single tape two characters wide. There an obvious correspondence between a mutation sequence, a generation sequence and an alignment. Results based on these notions are equivalent and the notions can be used interchangeably as convenient.

The machines that we consider are probabilistic finite-state machines as described in [Alli2]; they are equivalent to hidden Markov models. Normally we consider symmetric machines where p(ins) = p(del) = $p(ins_A) = p(ins_B)$. The message length of an instruction is $-\log_2$ of its probability plus either two bits for one character or $\log_2(12)$ bits for two differing characters; for simplicity only, it is assumed that all characters are equally probable. In highly conserved sequences, the probability of a mutation is low and its message length is high. The probability of an instruction also depends on the current state, S, of the machine:

$$ML(copy|S)=-\log_2(P(copy|S))+2$$
$$ML(change|S)=-\log_2(P(change|S))+\log_2(12)$$
$$ML(ins|S)=ML(del|S)=-\log_2(P(ins|S))+2$$

A 1-state machine corresponds to Sellers' edit distance[Sell] with adjustable weights. A 3-state machine corresponds to linear indel costs or weights. Machines with more than three states can correspond to piece-wise linear indel costs. Note that multiple states allow an instruction to have different probabilities in different contexts. In particular, for a 3-state machine a run of indels can have a higher probability of continuing than of beginning, giving a linear message length (cost) as a function of length.

A message based on one generation sequence (alignment) states (i) the parameter values for the machine and (ii) the sequence of instructions. The parameters are estimated by an iterative process: Initial values are chosen and used to find an optimal sequence. It is an elementary result of coding theory that in an optimal code for the sequence, the probabilities of instructions should come from the observed frequencies of instructions. (To see this for an alphabet of two letters consider a sequence of length n and its probability $p^f.(1-p)^{n-f}$, take $-\log_2$, differentiate with respect to p and set to zero.) The change in the parameter values cannot increase the message length of the given alignment but may cause another alignment to become optimal. Iteration is therefore necessary and is guaranteed to converge. A few iterations, typically four to eight, are sufficient for convergence. The process could find only a local minimum in theory but this is not a problem in practice. The parameters must be stated to an optimum accuracy; the instructions are chosen from a multistate distribution and Boulton and Wallace[Boul1] give the necessary calculations.

The r-theory (Q2) is based on *all* instruction sequences (alignments). The algorithms[Alli2] that compute the r-theory for the various machines are dynamic programming algorithms [DPA] that perform <+,logplus> on message lengths in the inner step or <*,+> on probabilities, instead of the usual <+,min> on edit distances or <+,max> on lengths of common subsequences. (logplus(log(A),log(B)) = log(A+B)) The iterative process is used to estimate the parameter values based on *weighted* frequencies.

Results reported in [Alli2] support the widely held view that the 3 and 5-state machines are better models for DNA than the 1-state machine. The 3 and the 5-state machines have increasing numbers of parameters to be stated, which tends to increase the message length, but this is more than made up for by a saving in stating the strings. There are practical difficulties in going beyond about 5-states. We do not yet have enough data to reliably judge between the 3 and 5-state machines. We do not yet know how best to incorporate a mutation model with more than 1-state into multiple alignment; this is discussed later. Traditionally, string alignment programs have been based on Sellers' metric[Sell], linear and piece-wise linear cost functions for indels[Goto1, Goto2], or arbitrary concave cost functions[Mill]. Some of these programs have several parameters requiring ad-hoc adjustments. MML encoding provides an objective way to compare models of differing complexities and to infer their parameter values.

We now examine the questions Q1-Q4 informally on the example strings A=TAATACTCGGC and B=TATAACTGCCG. There are two important cautionary notes: The strings are necessarily short and results on short strings must be taken with a pinch of salt. We use the 1-state model only because it is the simplest to discuss, not because it is strongly supported.

Q1: Are strings A and B related? Intuitively it seems that they probably are related but it is hard to see exactly how. They have at the least diverged considerably.

Q2: How far and in what ways have A and B diverged? The front sections, TATAA:TAATA can be related in several ways involving either two mismatches or two indels. The central CTs probably match. The final

sections, GCCG:CGGC, can be related with two indels and a mismatch, or four indels or four mismatches. Overall and on average we expect about seven matches, one or two mismatches and a few indels.
Q3: What is the best alignment? It is not possible to say more than it probably involves T[A:AA]T[AA:A]CT[GCCG:CGGC] (**Figure 2**). As noted above, the 1-state model is used in this example for simplicity. The 3 and 5-state models would penalise isolated indels more heavily.
Q4: What was the most recent common ancestor of A and B? It is not possible to say more than it was probably one of the strings described by the pattern TA[A]TA[A]CTα where α involved two to four Cs and/or Gs.

When the r-theory algorithm is run on strings A and B the results (**Figure 3**) quantify and support the informal arguments made above. The optimal alignments stand out in the density plot but there are several of them and many other near-optimal alignments. Any one alignment has a message length greater than the null-theory and is not acceptable but taking them all together the r-theory is acceptable.

3. Trees over Two Strings.

Given two related "present-day" strings A and B, we think of them as having evolved from a most recent common ancestor P. This involves two unknown steps or mutation machines, M_A and M_B (**Figure 4**). The usual models are reversible and mutating P into A by M_A is equivalent to mutating A back into P by M_A'. However the composition of two s-state machines, M_A' and M_B, is not in general equivalent to an s-state machine for s>1. Fortunately, for typical overall indel rates a single s-state machine is a good approximation to the composition of two or more s-state machines. It is thus acceptable to relate A and B directly by a single s-state machine. **Figure 5** gives the -$\log_2$ probability distribution for indel lengths for compositions of 1, 2 and 512 machines. Figure 5(a) is for machines that give linear indel costs. Figure 5(b) is for machines that can cause short (s) or long (L) indels and that give asymptotically piece-wise linear costs. The calculations are based on machines that can copy or insert only. It is the shape of the curves, not their absolute values, that is important. This also suggests that when considering any two strings out of k, and the path of machines (edges) between them in a tree, in order to get partial information on a multiple-alignment, it is acceptable to combine two or more adjacent s-state machines (edges) into one such machine.

Note that there are two forests over A and B. The r-theory tree (Figure 4) forms one forest. The forest of two singleton trees corresponds to the null-theory.

If the ancestor P and the mutation instruction sequences that M_A and M_B executed were known, it would be possible to form an equivalent generation sequence (**Figure 6**) for a single generation machine G relating A and B. However neither P nor the mutation sequences are known. The idea to circumvent this problem is to synchronise M_A and M_B as they read P and to remove dependency on the unknown P by summing over all possibilities. M_A and M_B are forced to execute copy, change and delete instructions in parallel; note that each of these instructions reads one character from P. Equivalently, G executes a match, mismatch, ins_A or ins_B instruction as appropriate except that if both M_A and M_B execute a delete, G does nothing. Note that an insert reads no character from P. If M_A is about to execute an insert instruction and M_B is about to execute some other kind of instruction then M_B is delayed and G executes an ins_A. Similarly, if M_B (only) is about to execute an insert, G executes an ins_B. If both M_A and M_B are about to execute one or more inserts, they and G can be made to do one of several things; double-counting can be avoided if this is done carefully (Figure 6). The inserts can be interleaved in several different ways. For example, G can execute an ins_A and then an ins_B or v.v. Each of these options corresponds to a part of an alignment. Some, such as an ins_A immediately followed by an ins_B are of low, but non-zero, probability and are not usually seen in alignments. For example, a single optimal alignment always explains a pair of characters <A_i,B_j> as 'change(A_i,B_j)' and never as '$ins_A(A_i)$;$ins_B(B_j)$' and thus gives biased estimates.

We expect P, A and B to have the same number of characters on average provided that pA(i)=pA(d) and pB(i)=pB(d). However we expect the alignment to be longer than P, for a tuple is "gained" whenever

M_A or M_B does an insert but a character is only "lost" when both M_A and M_B delete it. For this reason, rates of instructions *per* character of P, ie. pA' and pB', have a denominator of (1-pA(i)) or (1-pB(i)). The expected number of pairs in the alignment *per* character of P is 1+pA'(i)+pB'(i).

The above considerations are generalised to k≥2 strings in the next section. Note that null is a non-character, not a proper character. Similarly an edge with a null at each end represents a non-operation for a machine, not a proper operation. It is introduced only to synchronise M_A and M_B.

The parameters of G could be calculated from the parameters of M_A and M_B if the latter were known. However neither P, M_A nor M_B are known. In this case parameter values for G can still be inferred from A and B under the r-theory. The values embody the weighted average over all possible P and instruction sequences for M_A and M_B. If a constant rate of mutation is assumed and M_A=M_B then it is possible to infer the parameters of M_A. For three or more strings it is possible to get a three-way fix on all hypothetical strings and estimate all machines without assuming a constant rate of mutation.

4. Evolutionary Trees and Multiple-Alignment.

Of the questions, R1-R4, over k≥2 strings, we consider R2 to be the most important one to answer with R3 a close second, to recap: **R2**: Are they related by a particular tree? **R3**: Are they related [under an evolutionary-tree] by a particular multiple-alignment?

The topology of the tree is important but of little use without the amount and form of mutation represented by each edge. Strictly, R3 is too detailed to judge a tree on although an optimal alignment is often useful. As for the two-string case, optimal multiple-alignments are not unique in general and even sub-optimal alignments may be plausible.

An MML encoding of the strings based on an answer to R2 or R3 consists of (i) the value of k, (ii) the tree topology, (iii) the parameter values for each edge and either (iv) the strings, based on an all alignments (R2) or (iv') the strings based on one optimal alignment (R3).
The tree that best explains the strings gives the shortest message length. The difference in message lengths of two trees gives $-\log_2$ of their posterior odds-ratio. Note, the most colourless assumption is that all topologies are equally likely so part (ii) is of constant length, given k, and is only needed for comparison with the null-theory.

A rooted binary tree can only be inferred if an assumption of a (near) constant rate of mutation is made. If this assumption is not made, an unrooted binary tree is inferred.

R2 and R3 can be broken down into sub-problems:
S1: What is the best tree-topology?
S2: What are the parameters for each edge?
S3: What is the cost function for tuples in a multiple alignment?
S4: What is the message length of an alignment?
S5: What is the message length of the strings given the tree :- considering all alignments taken together? (R2)
S5': What is the message length of the strings given the tree and based on one optimal alignment? (R3)

Note that answers to these sub-problems are not independent. The topology and parameter values define the tuple cost, which affects the alignment probabilities, which decide the plausibility of each tree. Two competing sets of answers to the sub-problems can be judged easily and objectively but finding the best answers is hard. On the other hand, the algorithms for the sub-problems are largely independent and can be studied separately. A prototype program has been written for R3 based on particular approaches to these problems. In addition, some other possibilities have been examined. These are discussed in the following section.

4.1. S1: Tree-Topology.

An unrooted tree over k≥2 strings or leaves has k-2 internal, hypothetical strings and 2k-3 edges. There are U_k unrooted tree topologies over k strings where

$U_2 = U_3 = 1$

$U_k = 1.3.5. ... .(2k-5), k\geq 4$

Each is considered equally probable apriori and $\log_2(U_k)$ bits are required to state its identity:

$$ML(topology)=\log_2(U_k)$$

The prototype program requires the user to specify one or more trees to be evaluated.

Standard distance-matrix methods could be used to propose plausible topologies. The ideal distance is proportional to time. It is thought that the parameters derived from the two-string r-theory give more accurate pair-wise distances than a simple edit-distance.

Felsenstein[Fels2] describes how local rearrangements can be used to find at least a locally optimal tree.

There is no great difficulty in allowing n-ary trees, except that there are more of them. Note that any multiple-alignment method that does not explicitly consider a tree nevertheless embodies an implicit model of relation of some kind, possibly a fixed tree.

4.2. S2: Edge Parameters.

Each edge of the tree relates two, possibly hypothetical, strings and is represented by a machine. Any realistic machine requires at least one parameter to specify instruction probabilities, being related to the amount of time or mutation the edge represents. The parameter(s) of each of the 2k-3 machines must be included in the message. The instructions are chosen from a multi-state distribution and Boulton and Wallace[Boul1] give the calculations to determine the precision with which to state the instruction probabilities.

The prototype program is based on 1-state machines. In that case each machine has three parameters - P(copy), P(change), P(indel) - and two degrees of freedom. MML encoding is invariant under smooth monotonic transformations of parameters so the probabilities are stated rather than being translated into a measure of time. No correlation between P(change) and P(indel) is assumed, although some saving in stating the parameters would result if one were to be assumed.

If a constant rate of mutation were assumed, it would only be necessary to describe k-1 machines. This would permit an objective test of this much discussed hypothesis: There is a saving of approximately 50% on the cost of stating parameters but presumably a rise in the cost of describing the strings. An intermediate theory, hypothesising a near constant rate of mutation is also possible, with different trade-offs.

An iterative scheme, similar to that used for two strings, is used to estimate the parameter values of the machines (edges). This is described later.

4.3. S3: Tuple Costs.

Here we consider the tree topology and parameters to be fixed. The probability of an alignment depends on the probabilities of the k-tuples of characters forming the alignment. Each tuple can be thought of as an instruction of a k-string generation machine that generates all of the k strings in parallel. This machine is implied by the 2-string machines on the edges of the tree. The probability of a tuple can be found by assigning its characters (possibly null) to the leaves of the tree and considering all explanations of the tuple. An explanation makes assignments of characters to the internal, hypothetical characters of the tree. This defines the operation on each edge and a probability can be calculated for the explanation. The probability of the tuple is the sum of the probabilities of all of its explanations. The message length of the tuple is $-\log_2$ of its probability.

We first consider the case where the mutation-model on two strings is a 1-state model. Felsenstein[Fels2] gave the necessary algorithm to calculate the probability of a tuple of characters given the probability of a change on each edge of the tree. It efficiently sums the probabilities of all explanations for a tuple over all possible assignments to internal, hypothetical characters. For each edge, all possible assignments to the characters at both ends are considered. The time complexity is $O(k \cdot |\text{alphabet}|^2)$ for each tuple. The algorithm can be used directly by allowing the possibility of indels. Care must be taken that only one insert is allowed *per* explanation of a tuple, as an insert begins a new tuple. The tuple's probability is computed in a post-order traversal of the tree. The probabilities of hypothetical characters and of machine operations, given the tuple, (**Figure 7**) are computed in a subsequent pre-order traversal.

A table of tuple probabilities can be precomputed for small k. If k is larger, probabilities can be computed as tuples are encountered in alignments, and stored in a hash-table for reuse. There is a considerable saving when $n >> |\text{alphabet}|$. This is the technique used in the prototype program.

Sankoff and Cedergren[Sank] give a similar algorithm that finds the single most probable explanation for each tuple; this is too detailed a hypothesis in general and is another case where a max over probabilities should be replaced with a '+'. However the complexity of their algorithm is only O(k) per tuple. It gives a good approximation when all mutation probabilities are small and all edges are similar.

The situation is more complex when the model relating two strings has more than one state. The 3-state model, giving linear indel costs, is of particular interest. There are 2^k-2 patterns of indels possible in a tuple (**Figure 8**). The k-string generation machine therefore has 2^k-1 states. (2^k-1 is also the number of neighbours that each cell depends on in a k-dimensional DPA matrix.) Each tuple can therefore occur in 2^k-1 contexts with a different probability in each one. This is a serious problem unless a simplification can be found. There is some hope because the probabilities of instructions in these states are not independent as there are only O(k) free parameters in the tree.

4.4. S4: Alignment Message Length.

The message for the multiple-alignment itself states (i) the length of the alignment and (ii) the sequence of tuples in the alignment. The former is encoded under the $\log^*$ prior. The message length of a tuple was given in the previous section. It is therefore easy to compare two multiple-alignments and conclude which is best on the basis of message length. The difficulty is to find a good alignment.

4.5. S5': Optimal Multiple-Alignment.

The dynamic-programming algorithm [DPA] can be naively extended to k strings to find an optimal multiple-alignment. The k-dimensional DPA matrix has $O(n^k)$ entries. Each entry depends on 2^k-1 neighbours. If the mutation-model has one state, the extension is straightforward and the time complexity is $O(n^k.2^k)$. "Only" $O(n^{k-1})$ space is needed to calculate the value of the DPA. The n^k factor becomes painful, in both time and space, for $n \geq 200$ and k as small as three. It is necessary to reduce the volume of the DPA matrix that is searched to make the algorithms practical and there are a number of possible approaches.

Altschul and Lipman[Alts] devised a *safe* method to restrict the volume investigated for an edit-distance based on fixed tree-costs (they also give a method for star-costs). (The *tree-cost* of a tuple is the minimum number of mutations required to give the tuple; summing over all tuples gives the tree-cost of an alignment.) Pair-wise alignments are done first and used to delimit a volume that is guaranteed to contain an optimal multiple-alignment. However, no results are available to indicate how practical the method is for tree-costs. The method relies on the triangle inequality and it might be applicable to the MML calculations.

A related heuristic approach is to use alignment-regions as *fuzzy strings*. Pair-wise alignments are performed on sibling leaf strings. A threshold (in edit-distance or in message length) is applied to limit the area of each alignment. Fuzzy strings are aligned in the obvious way until a complete multiple-alignment is obtained. The method is fast if the threshold is tight but is not guaranteed to find an optimal alignment. We have done some experiments with such an algorithm using fixed tree-costs. With a threshold of 1, eight strings of 200 characters can be aligned on a DEC-5000. (With this kind of complexity the model of computer makes only a small difference to the practical limits.)

Hein[Hein] gave a multiple alignment algorithm which uses directed acyclic graphs (DAGs) to represent fuzzy ancestral strings. The algorithm is fast but an optimal alignment is not guaranteed. Fixed tree-costs are used but this is by no means necessary. The value of DAGs is that in sections where there is consensus between the strings, the effective number of neighbours of a cell in the DPA "matrix" is much reduced.

Long *exact* matches between strings are fast to find using hash-tables or suffix trees. Heuristics for multiple-alignment based on splitting the strings on a long exact match, ideally near their centres, are clearly possible.

The prototype program operates in two stages. During an initial post-order traversal, strings and then alignments are aligned recursively until a complete multiple-alignment is obtained:

```
procedure RA(var Result:Alignment; T:Tree);
   var Left, Right :Alignment;
begin if leaf(T) then
           StringToAlignment( T^.s, Result )
      else RA( Left,  T^.left );
           RA( Right, T^.right );
           Align( Left, Right, Result )
      end if
end
```

The machine parameters are held constant, at some reasonable but otherwise arbitrary values, during the initial stage. An improvement stage follows. During a second post-order traversal the complete alignment is projected onto subalignments which are realigned:

```
procedure Improve( var Algn :Alignment );
   var A, B :Alignment; ss :StringSet;

   procedure Im(T:Tree; var ss:StringSet);
      var ss1, ss2 :StringSet;
   begin if not leaf(T) then
            Im( T^.left, ss1 );
            Im( T^.right, ss2 );
            Project( Algn, ss1, A, B );
            Align( A, B, Algn );
            if ss1+ss2<>AllStrings then
               Project(Algn, ss2, A, B);
               Align( A, B, Algn )
            end if
         else {leaf(T)} ss := [T^.s]
         end if
   end;

begin Im( T, ss)
end
```

The machine parameters are continually reestimated during this second stage. The overall process takes $O(kn^2)$ time and global optimality is not guaranteed. The improvement stage can be iterated. More research needs to be done on strategies for optimal multiple-alignment.

Note that if the mutation-model on two strings has three states, to model linear indel costs, the k-string generation machine has 2^k-1 states and each entry in the DPA matrix has 2^k-1 entries. This also adds a multiplicative factor of 2^k-1 to the time complexity.

4.6. S5: Sum over all Alignments.

The part of the message that describes the strings proper is ideally based on all alignments. The number of bits required to state the strings is -$\log_2$ of the sum of the probabilities of all alignments. This can be computed by a <+,logplus> dynamic programming algorithm. It suffers from the same combinatorial explosion as finding an optimal alignment. Fortunately most of the probability is concentrated in a narrow region which raises hopes for a fast approximate algorithm. (This has been used to implement a fast, approximate r-theory algorithm for two strings[Alli2].) Feng[Alli3] has performed experiments on three strings. First, probability density plots are calculated for pair-wise comparisons. These are "thresholded" to give a small alignment region. The regions are projected into the DPA cube. The r-theory is calculated over this volume.

A further approach that we intend to implement is to to sum over all alignments in a region within a distance ‘w’ of a multiple-alignment found as described in the previous section.

Note that if the generation machine has ‘s’ states, pairs of states must be considered at each transition. So if linear indel costs are modelled, a multiplicative factor of $(2^k-1)^2$, or $O(2^{2k})$, is included. This is a severe problem unless a simplification or an approximation can be found.

4.7. Estimating Edge Parameters.

The probability of a tuple can be calculated, as described previously - if all the edge parameters are known. An iterative scheme is used to estimate unknown parameters; it is a generalisation of the scheme used for two strings: Initial values are assumed and used to find and to compute the message length of a multiple alignment (say) and hence of the tree hypothesis. For each tuple, the frequencies of operations for each machine are calculated. These frequencies are accumulated and used to calculate the instruction probabilities for the next iteration. The new values cannot increase the message length of the multiple-alignment but they may cause another alignment to be become optimal but this must reduce the message length further so the process converges.

A test program has been written to investigate the estimation process alone for the 1-state model. It assumes that a long and "perfect" alignment is known, its tuples occurring with their predicted frequencies. It then attempts to estimate the true machine parameters from arbitrary starting values. Convergence is rapid. The message length usually converges to within 1% of the optimum value within three iterations. All parameters are reestimated simultaneously. Note that the perfect alignment would be sub-optimal! Its statistical properties could only derive from a weighted average over all alignments.

This test program is also used to calculate limits on acceptable hypotheses for asymptotically long strings. The overhead of stating parameter costs and string lengths becomes a vanishing fraction of the total message length for very long strings and can be ignored. Briefly, the message length of the null-theory is calculated and the critical point is where this equals the message length of the perfect alignment. For example, when the machines in a tree are all identical and P(change)=P(indel) then the following results are obtained for different values of k:

```
                 k:  2    3    4    5
   critical P(copy): 0.68 0.79 0.82 0.83
```

The critical values of P(copy) would be higher for short strings. They would be lower if the probabilities of all alignments were added. [Alli1] gives empirical results for two strings.

The estimation scheme described above has been built into the prototype alignment program. It will also be used in any r-theory algorithm in which case weighted frequencies will be used for estimation.

The programs do not correct for the "invisible" tuple which has a probability of approximately $(k-2).p(del)^3$. This results in an underestimate of indels by a similar amount but this is considerably less than the accuracy with which parameters can be estimated.

4.8. Null-Theory.

The null-theory for the k strings is that they are not related. Its statement consists of (i) the value of k, (ii) the lengths of the strings and (iii) the characters in the strings. Part (iii) requires two bits per character for DNA if all characters are equally probable. Part (ii) is potentially a large penalty over a tree hypothesis because the latter states only a single length - of an instruction sequence. A reasonable approach, which mitigates this penalty, is to transmit the total length of the strings, encoded under the $\log^*$ prior, followed by the set of lengths according to a k-nomial probability distribution. The second part is short if the lengths are similar. This generalises the binomial distribution used for two strings in[Alli2]. The total message length of the null-theory is therefore:

$$
\begin{aligned}
ML_{null} = & -\log_2(P(k)) \\
& -\log_2(P(total)) \\
& -\log_2(P(length_1, ..., length_k \mid total)) \\
& +total*2 \\
\text{where } & total = length_1 + ... + length_k
\end{aligned}
$$

Any hypothesis that has a message length longer than that of the null-theory is not acceptable. The previous section included predictions of critical values of P(copy) for asymptotically long strings and some special trees. Extensive tests have not been done but when the alignment program is run on artificially generated data the results of the significance test are consistent with the predictions. In particular, random strings are always detected as being unrelated.

4.9. Results.

Figure 9 shows the results of the prototype program on a small example. All three trees are acceptable hypotheses but tree 1 is the most plausible by a significant margin. All trees are un-rooted. Note that little benefit is gained from the hash-table for such short strings.

Figure 10 gives results on transthyretin DNA and RNA sequences[Duan]. The grouping ((human sheep) (mouse rat)) is favoured on those four sequences. Joining chicken to the edge between the two pairs is narrowly the best option.

There is some debate over the branching pattern of human, chimpanzee and gorilla in evolutionary trees for the primates. **Figure 11** shows results on globin pseudo-gene sequences from primates. They favour the grouping (gorilla (human chimp)) and are consistent with the conclusions of Holmes[Holm, Ch 12] on these sequences. First, the three trees over gorilla, human, chimpanzee and owl-monkey sequences were evaluated. Owl-monkey was a consistent outsider. The orangutan sequence was then added and three trees were evaluated.

5. Conclusions.

We have shown how to use MML encoding to infer evolutionary-trees and multiple-alignments. We have argued that these problems are intimately connected. The posterior probability of a tree depends on the sum of the implied probabilities of all multiple-alignments. In this we agree with the sentiment, but not the detailed methods, of Hogeweg and Hesper[Hoge] that "the alignment of sets of sequences and the construction of phyletic trees cannot be treated separately." The use of a single multiple-alignment leads to an underestimate of the posterior probability of the tree hypothesis and a bias in the parameter estimates. Nevertheless it is a good approximation when the strings are similar. A prototype program of this kind has been written for the 1-state model.

The MML method can compute the posterior odds-ratios of competing hypotheses such as different trees. The evolutionary-tree hypothesis is explicitly built into the message. The existence of a natural null-theory leads to an in-built significance test. These remarks are qualified with ‘subject to the validity of the underlying model.’ The MML method requires the model to be defined explicitly. In principle, two competing models can be compared objectively; this was done for two strings in [Alli2].

The most significant doubts about the model used are as follows. We have not, as yet, modelled different transition/transversion rates but this would be straightforward. What seems to be more difficult is to bring a multi-state mutation model into multiple-alignment. Complex mutation events, such as transpositions, duplications and reversals, could be modelled in principle although finding an optimal "alignment" would become much harder. It must also be pointed out that what is completely lacking from our method, and from most (all?) sequence analysis programs, as yet, is any model of the pressure of selection.

The program discussed above is written in Pascal and is available, free for non-profit, non-classified teaching and research purposes, by email (only) from: lloyd@bruce.cs.monash.edu.au. The alignment programs for two strings, based on 1, 3 and 5-state models of relation and discussed in [Alli2] are written in C and are also available. At least a good work-station with floating point hardware is required for their use.

6. References.

[Alli1] Allison L. & C. N. Yee. Minimum message length encoding and the comparison of macro-molecules.
Bull. Math. Biol. 52(3) 431-453 1990.

[Alli2] Allison L., C. S. Wallace & C. N. Yee. Inductive inference over macro-molecules.
AAAI Spring Symposium on A.I. & Mol. Biol. Stanford March 1990 (extended abstract), and TR 90/148 Dept. Comp. Sci. Monash University 1990 (in full).

[Alli3] Allison L. & Du Xiaofeng. Relating three strings by minimum message length encoding (abstract).
Genes, Proteins and Computers, Chester, April 1990.

[Alts] Altschul S. F. & D. J. Lipman. Trees, stars and multiple biological sequence alignment.
SIAM J. Appl. Math. 49(1) 197-209 1989.

[Bish] Bishop M. J. & E. A. Thompson. Maximum likelihood alignment of DNA sequences.
J. Mol. Biol. 90 159-165 1986.

[Boul1] Boulton D. M. & C. S. Wallace. The information content of a multistate distribution.
J. Theor. Biol. 23 269-278 1969.

[Boul2] Boulton D. M. & C. S. Wallace. A program for numerical classification.
Comp. J. 13 63-69 1970.

[Boul3] Boulton D. M. & C. S. Wallace. An information measure for hierarchic classification.
Comp. J. 16 254-261 1973.

[Chee] Cheeseman P. & B. Kanefsky. Evolutionary tree reconstruction (extended abstract).
AAAI Spring Symposium on the Theory and Application of Minimal-Length Encoding Stanford 35-39 March 1990.

[Duan] Duan W., M. G. Achen, S. J. Richardson, M. C. Lawrence, R. E. H. Wettenhall, A. Jaworowski & G. Schreiber. Isolation, characterization, cDNA cloning and gene expression of an avian transthyretin: implications for the evolution of the structure and function of transthyretin in vertebrates.
Euro. J. Biochem. to appear.

[Fels1] Felsenstein J. Evolutionary trees from DNA sequences: a maximum likelihood approach.
J. Mol. Evol. 17 368-376 1981.

[Fels2] Felsenstein J. Inferring evolutionary trees from DNA sequences.
Statistical Analysis of DNA Sequence Data. B. S. Weir (ed) Marcel Dekker 133-150 1983.

[Frid] Friday A. Quantitative aspects of the estimation of evolutionary trees.
Folia Primatol. 53 221-234 1989.

[Goto1] Gotoh O. An improved algorithm for matching biological sequences.
J. Mol. Biol. 162 705-708 1982.

[Goto2] Gotoh O. Optimal sequence alignment allowing for long gaps.
Bull. Math. Biol. 52(3) 359-373 1990.

[Hein] Hein J. A new method that simultaneously reconstructs ancestral sequences for any number of homologous sequences when the phylogeny is given.
Mol. Biol. Evol. 6 649-668 1989.

[Hoge] Hogeweg P. & B. Hesper. The alignment of sets of sequences and the construction of phyletic trees: an integrated method.
J. Mol. Evol. 20 175-186 1984.

[Holm] Holmes E. C. Pattern and Process in the Molecular Evolution of the Order Primates.
PhD Thesis, Cambridge University, 1989.

[Mill] Miller W. & E. W. Myers. Sequence comparison with concave weighting functions.
Bull. Math. Biol. 50(2) 97-120 1988.

[Milo] Milosavljevic A. D. Categorization of macro-molecular sequences by minimum length encoding.
University of California at Santa Cruz, UCSC CRL 90-41 1990.

[Reic] Reichert T.A., D. N. Cohen & K. C. Wong. An application of information theory to genetic mutations and the matching of polypeptide sequences.
J. Theor. Biol. 42 245-261 1973.

[Sank] Sankoff D. & R. J. Cedergren. Simultaneous comparison of three or more sequences related by a tree.
Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison. D. Sankoff & J. B. Kruskal (eds) Addison Wesley 253-263 1983.

[Sell] Sellers P. H. On the theory and computation of evolutionary distances.
SIAM J. Appl. Math. 26(4) 787-793 1974.

[Wall1] Wallace C. S. & D. M. Boulton. An information measure for classification.
Comp. J. 11 185-195 1968.

[Wall2] Wallace C. S. & P. R. Freeman. Estimation and inference by compact coding.
J. Royal Stat. Soc. B 49(3) 240-265 1987.

```
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. $ch_A \neq ch_B$
 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.

```
TATAA    TA-TAA    T-ATAA
||  |    || ||     | || |
TAATA 2  TAATA- 2  TAAT-A 2 etc.

-CGGC    CGGC-    CGGC
 | |      | |
GCCG- 3  -GCCG 3  GCCG 4

--CGGC    CGGC--    CG-GC-
  ||        ||       |  |
GCCG-- 4  --GCCG 4  -GC-CG 4

-CGGC-
 |  |
GC--CG 4 etc
```

Figure 2: Possible Sub-alignments.

```
     T A T A A C T G C C G
    • - - .
T:  - • - - . .
A:  - # • + - . .
A:  . - • + + - - . .
T:    . - • • - - - . .
A:    . . - # • - - - .
C:      . . - + • + - - .
T:        . - + + • # + . .
C:        . . . + # + # + .
G:          . . - + # # # +
G:              . . + # + #
C:                . . + # •

1-state machine

key:        •  #  +  -  .
MML plus: 0..1..2..4..8..16 bits

  1 avg algn = 64.9 bits
 null-theory = 56.0 bits
    r-theory = 53.8 bits
= : ≠ : indel = 7 : 1 : 5.9
probability related = 0.8
```

Figure 3: Probability Density Plot.

```
      P
     . .
 M_A.   .M_B
   .     .
 A        B
```

Figure 4: Common Ancestor.

```
Generation       Explanations and Rates.
Instruction.

match(x)
x x      x<---x--->x     x<---y--->x
         pA'(c).pB'(c)   pA'(ch).pB'(ch)/3

mismatch(x,y)
x y      x<---x--->y     x<---y--->y
         pA'(c).pB'(ch)  pA'(ch).pB'(c)

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

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

         x<---_--->_
         pA'(i)

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

         _<---_--->x
         pB'(i)

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

c-copy, ch-change, i-insert, d-delete
define pA'(instn) = pA(instn)/(1-pA(i))
       pB'(instn) = pB(instn)/(1-pB(i))
       rates per character of P
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 6: Explanations for Generation Instructions.

```
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.

Prior tuple probability = 0.00027


Estimated operations carried out:

     copy change indel
m1:  0.43 0.57
m2:  0.57 0.43
m3:  0.40 0.6
m4:  0.59 0.41
m5:  0.94 0.06

(Probabilities less than 0.01 omitted.)
```

Figure 7: Example, explanations for tuple ACAC.

```
string A: -  A  -  A  -  A
string B: B  -  -  B  B  -
string C: C  C  C  -  -  -

          1  2  3  4  5  6
```

Figure 8: Indel Patterns, k=3.

```
Input:
   4                  - k
   acgtacgtacagt      - string 1
   actgtacgtacgt      -        2
   acgtactagct        -        3
   accgtactgagct      -        4
   ((1 2)(3 4))       - tree 1
   ((1 3)(2 4))       -      2
   ((1 4)(2 3))       -      3


Tree 1:   M.L. =   108 bits *

   estimate of parameters:
   mc1: 0.88 0.02 0.09
   mc2: 0.88 0.02 0.09
   mc3: 0.94 0.03 0.03
   mc4: 0.81 0.02 0.17
   mc5: 0.71 0.18 0.11

   s1                    s4
    .                    .
     .m1               .m4
      .      m5       .
       .--------------.
      .                .
     .m2               .m3
    .                    .
   s2                    s3

   s 1[   1.. ]: ac--gtacgt-acagt
   s 2[   1.. ]: ac-tgtacgt-ac-gt
   s 3[   1.. ]: ac--gtac-t-ag-ct
   s 4[   1.. ]: acc-gtac-tgag-ct

Tree 2:   M.L. =   113 bits
Tree 3:   M.L. =   122 bits
Null-Theory= 124.5 bits
NB. all trees are unrooted.

CPU time = 7 secs/tree (Pyramid)
```

Figure 9: Example Run.

```
s1: HUMPALA  Human prealbumin
    (transthyretin) mRNA, complete cds.
s2: RATPALTA Rat prealbumin
    (transthyretin) mRNA, complete
s3: MUSPALB  Mouse mRNA for prealbumin
s4: sheep    Transythyretin cDNA
s5: chicken  TTR cDNA transthyretin

s1-s4:
  Tree ((1 2)(3 4)): M.L. =3693 bits
  Tree ((1 3)(2 4)): M.L. =3661 bits
  Tree ((1 4)(2 3)): M.L. =3569 bits *

  Null-Theory = 4781 bits

s1-s5:
  Tree ((1 4) (2(5 3))): M.L. =4846 bits
  Tree ((1 4) ((2 5)3)): M.L. =4844 bits
  Tree (((1 5)4) (2 3)): M.L. =4807 bits
  Tree ((1(4 5)) (2 3)): M.L. =4758 bits
  Tree (((1 4)5) (2 3)): M.L. =4750 bits*
*incl: 16.6 length, 67.1 params,
       4655.9 alignment
   Null-Theory = 6134 bits

            s5
            |
            |
s1          |m5        s2
  .         |         .
   .m1      |        .m2
    .   m6  |   m7  .
     s6-----s8-----s7
    .               .
   .m4               .m3
  .                   .
s4                     s3

estimate of parameters:
     copy change indel
mc1: 0.89  0.09  0.02
mc2: 0.84  0.04  0.12
mc3: 0.95  0.05  0.01
mc4: 0.88  0.08  0.05
mc5: 0.74  0.18  0.08
mc6: 0.96  0.04  0.00
mc7: 0.88  0.10  0.03
```

CPU time approx’ 5 min/tree (sparc station).

Figure 10: Transthyretin DNA/RNA sequences.

```
1: GGBGLOP:Gorilla
   beta-type globin pseudogene
2: HSBGLOP:Human beta-type globin
   pseudogene standard; DNA; PRI; 2153 BP.
3: PTBGLOP:Chimpanzee beta-type globin
   pseudogene standard; DNA; PRI; 2149 BP.
4: ATRB1GLOP:Owl-monkey psi beta 1 globin
   pseudogene.  2124 bp ds-DNA
5: ORAHBBPSE: Orangutan bases 1593 to 3704

s1-s4:
   Tree ((1 2)(3 4)): M.L. =7241 bits
   Tree ((1 3)(2 4)): M.L. =7236 bits
   Tree ((1 4)(2 3)): M.L. =7197 bits*
   Null-Theory = 17193 bits
   Elapsed time ~50 min's (Sparc station)

s1-s5:
   Tree (((1 2) 3)(4 5)): M.L. =8023 bits
   Tree (((1 3) 2)(4 5)): M.L. =8031 bits
   Tree ((1 (2 3))(4 5)): M.L. =7980 bits*
*incl: 18.7 length, 98.8 params,
       7852 alignment
   Null-Theory = 21426 bits
   NB. all trees are un-rooted.
   Elapsed time ~50 min's (Sparc station)

estimate of parameters:
     copy change indel
mc1: 0.989 0.011 0.000
mc2: 0.994 0.004 0.003
mc3: 0.988 0.012 0.001
mc4: 0.868 0.087 0.045
mc5: 0.969 0.016 0.015
mc6: 0.996 0.001 0.003
mc7: 0.985 0.011 0.005

            s1:GGB
              |
              |m1
       m2  m6 |    m7         m4
s2:HSB---s6---s7-------s8----------s4:ATR
         |             |
         |m3           |m5
         |             |
       s3:PTB          |
                     s5:ORA
```

Figure 11: Primate Globin Pseudo-genes.