Finite-State Models in the Alignment of Macro-Molecules.

L. Allison, C. S. Wallace and C. N. Yee.

Department of Computer Science,
Monash University,
Australia 3168.

May 1991.
Revised Nov 1991.

uucp: xxx@cs.monash.edu.au xxx=[lloyd, csw, cyee]

Supported by Australian Research Council grant A48830856

Appears in: Journal of Molecular Evolution, **35**(1) 77-89 July 1992.

(An abstract of this paper was presented under the title 'Inductive Inference over Macro-Molecules' at the AAAI Symposium on Artificial Intelligence and Molecular Biology, Stanford, March 1990.)

Summary.

Minimum message length encoding is a technique of inductive inference with theoretical and practical advantages. It allows the posterior odds-ratio of two theories or hypotheses to be calculated. Here it is applied to problems of aligning or relating two strings, in particular two biological macro-molecules. We compare the *r-theory*, that the strings are *related*, with the null-theory, that they are not related. If they are related, the probabilities of the various alignments can be calculated. This is done for one, three and five-state models of relation or mutation. These correspond to linear and piece-wise linear cost functions on runs of insertions and deletions. We describe how to estimate parameters of a model. The validity of a model is itself an hypothesis and can be objectively tested. This is done on real DNA strings and on artificial data. The tests on artificial data indicate limits on what can be inferred in various situations. The tests on real DNA support either the three or five-state models over the one-state model. Finally, a fast, approximate, minimum message length string comparison algorithm is described.

Keywords: alignment, edit distance, homology, inductive inference, Minimum Message Length, MML, similarity, string.

1. Introduction.

Minimum message length (MML) encoding techniques are applied to inductive inference problems over strings derived from sequencing biological macro-molecules. The paper is framed in terms of DNA strings but the techniques apply to strings over other alphabets such as proteins. It extends work begun by Allison and Yee (1990).

Given two strings, we consider the questions (i) are they related and, if so, (ii) how are they related? We give algorithms to answer these questions for one, three and five-state models of relation. We consider the question of what is the right model for a family of strings and give some results for DNA. A fast approximate MML algorithm is also described.

To illustrate the MML method of string comparison, consider a *transmitter* who must send two strings to a *receiver*. To save on transmission costs, they may previously have agreed on any algorithms, codes and conventions to be used. Given data D, we consider a message in the form of an *explanation* which first states an hypothesis H and then states D using a code which would be optimal were H true. From Bayes' theorem, in the discrete case,

$$P(H\&D) = P(D|H).P(H) = P(H|D).P(D)$$

From coding theory, an optimal code for an event of probability p uses $-\log_2(p)$ bits. Therefore, the message length (ML) of the explanation is

$$ML(H\&D) = ML(D|H)+ML(H) = ML(H|D)+ML(D)$$

If ML(H) and ML(D|H) can be computed for an hypothesis H, and similarly for H', then the posterior odds-ratio, P(H|D)/P(H'|D) can be computed. (Similar results hold in the continuous case.) Note that we often shorten the phrase 'message length of an explanation based on the hypothesis' to 'message length of the hypothesis'.

The *null-theory* or hypothesis is that the strings are unrelated. In this case, the transmitter sends one string and then the other which requires approximately two bits per character for DNA. Conversely, the *r-theory* is that the strings are *related*. If the transmitter can find a "good" theory or hypothesis, perhaps a good alignment, relating the characters of the two strings then it may be possible to encode them more succinctly. The transmitter then sends a message which states (a) the inferred theory and (b) encodes the strings using a code based on the theory. The MML method selects the theory giving the shortest message. A very detailed theory may allow a very short encoding of the strings proper but the details of the theory add to the message length. MML encoding thus defines an optimal degree of detail in this trade-off.

It is assumed that any one string is random in the sense of Kolmogorov complexity (Kolmogorov, 1965; Chaitin, 1966; Li & Vitanyi, 1988). This is nearly the case for biological macro-molecules and it is difficult to encode a typical DNA string in much less than two bits per character. This assumption is not essential to the MML method in principle but makes its application easier. There is, for example, an echo of a presumed purine-any-pyrimidine (RNY) ancient genetic code (Shepherd, 1981) which could be accommodated by considering codons to be the characters within strings. Repetitive DNA is certainly not random and both the null-theory and the r-theory would have shorter message lengths for such DNA if this fact were taken into account.

We relate strings by simple generation and mutation machines. We consider finite-state machines, particularly one, three and five-state machines. Such machines are implicit in many existing alignment methods and there are efficient inference algorithms for them. Finite-state machines can approximate a wide range of models.

If accurate prior probabilities for strings and relations are known, an optimal code can be designed for them. The minimum message length is $-\log_2$ of the probability of the event described. Often such probabilities are not known in advance but a model having one or more parameters can be agreed on. In our case the parameters are the probabilities of machine instructions; intuitively, increasing time since divergence leads to increasing probabilities of mutations. A message must include the parameter values and the best values give the shortest message. Two hypotheses can be compared on the basis of message lengths and the difference in message lengths is $-\log_2$ of their posterior odds-ratio. This leads to hypothesis testing because a natural null-theory exists. No hypothesis having a message length longer than the null-theory is acceptable.

There are practical and theoretical advantages to MML encoding as a method of inductive inference. The message paradigm keeps us *honest* in that anything not agreed to *apriori* by the transmitter and the receiver must be included in the message. When real-valued parameters are involved, the method implies an optimal degree of precision for their estimates. It is invariant under continuous, monotonic transformation of parameters. It focuses attention on the language being encoded, on what it can express and on the probabilities of messages. When accurate prior probabilities are not known, all the techniques and heuristics of data-compression and of coding theory (Hamming, 1980) can still be used to get a good code. MML encoding is *safe* in that the use of a suboptimal code cannot make a theory appear more acceptable than it should be.

Note that in practice no encoding actually takes place; we simply calculate what the message length would be if encoded. Note also that arithmetic coding (Langdon, 1984; Witten *et al*, 1987) is capable of transmitting information in fractions of a bit.

MML string comparison is closely related to maximum-likelihood methods. An important difference is that MML methods fully cost the model complexity. Felsenstein (1981) gave a maximum-likelihood method to infer an evolutionary binary-tree for a family of aligned strings. It is the method with the best statistical credentials for that problem to date. Bishop *et al* (1987) gave a maximum-likelihood method to infer the time since two related strings diverged. It is similar to our MML method for a one-state model. Thorne *et al* (1991) also recognised the benefits of this kind of approach to alignment. Bishop and Friday (1986) note that "there is as yet no formally developed method for assessing the significance of the difference in the likelihoods (of two different [phylogenetic] trees)." The MML method does not suffer from this problem and has a formal interpretation of differences in message length and indeed has a rigorous test of statistical significance.

Reichert *et al* (1973), Wong *et al* (1974) and Cohen *et al* (1975) first considered the brief encoding of pairs of strings under a non-standard model of relation but did not compare models. Milosavljevic (1990) considered the classification of aligned macro-molecules by MML methods.

MML encoding has developed outside Molecular Biology. Solomonoff (1964) proposed MML encoding as a basis for inductive inference. Wallace and Boulton (1968) applied it in producing a practical classification algorithm and later an algorithm for hierarchical classification (Boulton & Wallace, 1973). Georgeff and Wallace (1984) give a good introduction to MML methods and Wallace and Freeman (1987) describe the statistical foundations.

2. Relation Models.

There are many ways of explaining or modelling two strings given that they are related. In the *mutation model*, a simple machine reads one given string and a sequence of *mutation instructions* while writing the second given string (figure 1). In the *generative model*, a different simple machine reads a

```
string A:      TATACGTTACAC
instructions: copy, copy, ins(A), copy, copy,
               change(G), change(C),
               copy, copy,
               del, copy, copy, copy, ins(A)
string B:      TAATAGCTTCACA
```

Figure 1. Mutation Model Example.

sequence of *generative instructions* while writing the two given strings in parallel (figure 2). Note that for traditional reasons we use the terms *change* when a different character appears in each string, *delete* (del) when a character appears in string A and not in string B and *insert* (ins) when a character appears in B and not in A under both models but with appropriate meanings. For both models, a given sequence of instructions is equivalent to a traditional alignment (figure 3). It is therefore clear that the mutation and generative models are equivalent and we usually prefer the generative model on grounds of symmetry. The terms ‘alignment’ and ‘instruction sequence’ are used interchangeably.

Note that alignments and sequences of generative or mutation instructions are not unique. Often we want an optimal explanation, alignment or instruction sequence, but even suboptimal explanations may be plausible.

Given two macro-molecules, A and B, it is often assumed that they evolved independently from a common ancestor P. We could model this by two mutation machines, MA and MB, which each read P and their own instruction sequence and produced A and B respectively. A and B have infinitely many possible ancestors P but most are wildly improbable. Summing over all possible P gives the probability of A and B being related under this model. If we knew P and the instruction sequences for MA and for MB, the sequences could be synchronised to derive an alignment or an instruction sequence for a single generation machine. An alignment corresponds to a set of possible ancestors in a natural way(figure 4). Therefore the r-theory should not be based on just one alignment but summed over all of them. We can decide if A and B are related but P must remain uncertain unless extra information is provided. Note that there may have been characters in P that were deleted in both A and B and for which no direct evidence remains. Note also that the assumption of a constant rate of evolutionary mutation implies that MA and MB are equal and it is difficult to see how one could make any other assumption given just two strings.

Traditional alignment programs (Needleman & Wunsch, 1970; Sankoff & Kruskal, 1983; Waterman, 1984; Bishop & Rawlings, 1987) select single alignments that have a minimum cost, or equivalently a maximum score, under some measure. The minimum message length principle implies that the cost of an alignment should be the length of a message under an optimal code for alignments. On the other hand, to answer the question, ‘are A and B related’, *all* alignments should contribute according to their probabilities. The corresponding message and its code should transmit A and B given that they are related; it should not transmit a single alignment which is a more detailed hypothesis, specifying a particular relation.

```
instructions: match(T), match(A), ins(A), match(T), match(A),
              change(C,G), change(G,C),
              match((T), match(T),
              del(A), match(C), match(A), match(C), ins(A)

string A: TATACGTTACAC
string B: TAATAGCTTCACA
```

Figure 2. Generative Model Example.

```
string A: TA-TACGTTACAC-
          || ||   || |||
string B: TAATAGCTT-CACA
```

Figure 3. An Alignment.

```
Alignment 1: TA-TACGTTACAC-
             || ||  || |||
             TAATAGCTT-CACA

Ancestors 1: TA[A-]TA[CG][CG]TT[A-]CAC[A-]


Alignment 2: T-ATACG-TT-ACAC
             | ||| | || |||
             TAATA-GCTTCACA-

Ancestors 2: T[A-]ATA[C-]G[C-]TT[C-]ACA[C-]
```

Figure 4: Alignments and Possible Ancestors.

Our models and codes may have parameters for the form of a generative machine and the probabilities of its instructions which must be included in the message. In this way simple and complex models may be compared on an equal footing. Methods of estimating parameter values are described later. In following sections we specify and compare one, three and five-state models. We consider finite-state machines because they can approximate other models and have efficient inference algorithms for the r-theory. They are also implicit in many current alignment methods.

3. The One-State Model.

The one-state generative machine has a single state and transitions or instructions of the following forms: match(ch), change(chA,chB), ins(ch), del(ch). The *match* instruction generates the same character 'ch' in both strings. The *change* instruction generates chA in string A and chB in string B, chA≠chB. The *insert* (ins) instruction generates a character in string B and the *delete* (del) instruction generates a character in string A. A single instruction having probability 'p' is optimally encoded in $-\log_2(p)$ bits plus two bits for each character, except that the two characters in a change are different and together require $\log_2(12)<4$ bits:

$$\mathrm{ML(match(ch))} = -\log_2(\mathrm{P(match)}) + 2$$
$$\mathrm{ML(change(chA,chB))} = -\log_2(\mathrm{P(change)}) + \log_2(12)$$
$$\mathrm{ML(ins(ch))} = -\log_2(\mathrm{P(ins)}) + 2$$
$$\mathrm{ML(del(ch))} = -\log_2(\mathrm{P(del)}) + 2$$

Often we require that P(ins)=P(del). The probability of a sequence of instructions of fixed length is the product of the probabilities of the individual instructions and the message length of the sequence is the sum of the message lengths of the instructions.

If the probabilities of instructions are known *apriori*, a fixed code can be based on them. The familiar dynamic programming algorithm (DPA) is used to compute the message length of an optimal instruction sequence (figure 5a). The cost of an instruction is its message length - see above. Note that a match does not have zero cost. D[i,j] is the message length of an optimal sequence to generate A[1..i] and B[1..j]. It has been shown (Allison *et al*, 1990) how to normalise Seller's (1974, 1980) and similar metrics so as to extract the probabilities implicit in them. Ignoring the matter of the coding of lengths and of parameter values for now, the algorithm finds an alignment or instruction sequence having the minimum message length.

```
match  = ML(match(ch))
change = ML(change(ch1,ch2))
insert = ML(ins(ch))
delete = ML(del(ch))

Boundary Conditions:
   D[0, 0] := 0
   D[i, 0] := delete × i,  i:=1..|A|
   D[0, j] := insert × j,  j:=1..|B|

General Step, i:=1..|A| and j:=1..|B|:
  D[i,j] := min( D[i-1,j  ] + delete,
                 D[i  ,j-1] + insert,
                 D[i-1,j-1] +
                    if A[i]=B[j] then match
                                 else change )
```

Figure 5a. Dynamic Programming Algorithm, 1-state.

When probabilities are not known in advance, they must be inferred from the strings. An iterative approach finds at least a local optimum: initial probabilities are guessed and an optimal alignment is found. New probabilities are calculated from the frequencies of instructions observed in the alignment. The new values are optimal for the alignment and cannot increase its message length but they may cause a new alignment to become optimal - with an even lower message length - so iteration is necessary. Convergence is guaranteed and is quick; three or four iterations are usually sufficient for convergence to within one bit which is more than enough for practical purposes. The inferred parameter values are included in the message with an optimum accuracy; Boulton and Wallace (1969) give the necessary calculations for a multi-state distribution from which instructions are drawn.

The length of a sequence is not known in advance and must be included in any message. In the absence of more information, we use Rissanen's (1983) universal $\log^*$ prior to transmit the length.

The overhead of transmitting the length of an instruction sequence and the parameter values varies only slowly with the parameter values and becomes a smaller fraction of the total message length as string length increases. Since the DPA does not consider the overhead when finding an alignment, the overall result is not guaranteed to be optimal. In practice it is, particularly for reasonably long strings.

Optimal alignments are not unique in general. Even suboptimal alignments make some contribution, often a considerable contribution, to the r-theory. The 'min' in the DPA (figure 5a) chooses a single alignment having the smallest message length, the largest probability. If the probabilities are instead *added*, for they correspond to exclusive events, then D[i,j] is the $-\log_2$ probability that A[1..i] and B[1..j] arose through being generatively related by the machine but in *some* unspecified way. This yields the probability of A and B given the r-theory. The addition of probabilities is a probabilistic argument but a corresponding code has been devised by Wallace (1990) for general inference where one of a number of hypotheses may be true. It reduces the message length of the r-theory below that of a single alignment. The saving is negligible, but non-zero, for identical strings but can easily be 5% when P(match)˜0.6 and can make the difference between an acceptable and an unacceptable hypothesis. (The r-theory algorithm computes with message lengths directly and these are combined by 'logplus' where logplus(log(P),log(Q)) = log(P+Q).)

Instruction probabilities must be inferred if they are not known: the previous iterative process is used. The weighted averages of instruction frequencies are computed when alternative alignments (instruction sequences) are combined in the modified DPA (figure 5b). It is clear that the modified algorithm takes

O(|A|×|B|) time and space.

The null-theory is that strings A and B arose independently and this is the complement of the r-theory. The null-theory encodes string A and then string B. It must include their lengths. Rather than transmit two lengths which would be a severe penalty to the null-theory, we transmit the sum of the string lengths using the universal prior and the difference of the lengths using a binomial prior. This is reasonable and reduces the penalty because random insertions and deletions perform a random walk on the difference of the lengths. If P(ins)=P(del)=1/2, the null-theory and the r-theory have the same prior over |A|–|B|. If P(ins)=P(del)<1/2, the r-theory has a tighter distribution on |A|–|B|.

The odds-ratio of the r-theory and the null-theory is the binary exponential of the difference in their message lengths. Since they are complementary, their posterior probabilities can be calculated.

```
Boundary Conditions:
  D[0, 0].ML := 0
  D[0, 0].ins := 0               instruction
  D[0, 0].del := 0               frequency
  D[0, 0].change := 0            counters
  D[0, 0].match := 0

  D[i, 0].ML := delete × i      |
  D[i, 0].ins := 0              |
  D[i, 0].del := i              |
  D[i, 0].change := 0           |
  D[i, 0].match := 0            |  i:=1..|A|

  D[0, j].ML := insert × j      |
  D[0, j].ins := j              |
  D[0, j].del := 0              |
  D[0, j].change := 0           |
  D[0, j].match := 0            |  j:=1..|B|

General Step, i:=1..|A| and j:=1..|B|:
  down   := D[i-1,j  ] + delete; across := D[i,  j-1] + insert
  diag   := D[i-1,j-1] + if A[i[=B[j] then match else change

  Pdown  := exp2(down); Pacross := exp2(across)
  Pdiag  := exp2(diag); Psum := Pdown+Pacross+Pdiag

  D[i,j].ML := logplus( down, across, diag )

  D[i,j].del   := ( Pdown  *(D[i-1,j  ].del + 1)   | compute a
                  + Pacross* D[i,  j-1].del        | weighted
                  + Pdiag  * D[i-1,j-1].del )/Psum | frequency

  D[i,j].ins   := similar
  D[i,j].match := similar
  D[i,j].change:= similar
```

Figure 5b. R-theory Algorithm, 1-state.

Running the r-theory in *reverse*, D'[i+1,j+1] is $-\log_2$ the probability of A[i+1..|A|] and B[j+1..|B|] being generatively related. Thus, D[i,j]+D'[i+1,j+1] is the $-\log_2$ probability of A[1..i] and B[1..j] being generated and A[i+1..|A|] and B[j+1..|B|] being generated and is proportional to the probability of the "true" alignment partitioning the strings in this way. A density plot of this function is a good way of visualising possible alignments (figure 6). Note the areas of uncertainty where there are multiple reasonable explanations, CG:GC and ACAC:CACA. In this example the r-theory is certainly *acceptable* but any single alignment is at best marginally acceptable over the null-theory. The density plot is a "fuzzy" alignment. It is a good representation of the possible ancestors of A and B for use in multiple string alignment or in the reconstruction of evolutionary trees (Allison *et al*, 1992).

4. The Three-State Model.

We note that string alignment algorithms employing linear costs $w(L)=a+b\times L$ for indels of length L are in common use. The constant 'a' is a start-up cost to discourage the inference of many short runs of indels and is thought to be realistic for macro-molecules. This can be modelled by a three-state machine (figure 7). When P(ins|S1) < P(ins|S2) there is a low start-up probability, a high start-up cost, on block indels. A linear cost function corresponds to a geometric prior on indel lengths, for $-\log_2(p^L q) = -\log_2(q) - \log_2(p)\times L$.

Gotoh (1982) has given an algorithm to infer an optimal alignment for linear weights. Briefly, the DPA is modified to maintain three values in each cell of the D matrix. One value corresponds to each state. As before, we have further modified the algorithm to *add* probabilities of alignments. Again, instruction probabilities must be inferred from the strings if they are not known *apriori*. The iterative scheme described previously is used.

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

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

null-theory=  62.4    r-theory=  58.5
probability related=0.94
```

Figure 6. Alignment Densities.

```
states:
  S1: match/change state
  S2: insert state
  S3: delete state

transitions:
  S1--match--->S1  S1--change-->S1
  S1--insert-->S2  S1--delete-->S3
  S2--match--->S1  S2--change-->S1
  S2--insert-->S2  S2--delete-->S3
  S3--match--->S1  S3--change-->S1
  S3--insert-->S2  S3--delete-->S3

conservation:
  p(m|Sj)+p(c|Sj)+p(i|Sj)+p(d|Sj)=1, j=1,2,3
```

Figure 7. Three-State Machine.

The algorithm is run forwards and backwards and results added as before to produce alignment density plots. The three-state model has more parameters than the one-state model and these add to the length of the header part of the r-theory message.

5. The Five-State Model.

Gotoh (1982) also provided an algorithm to find a single optimal alignment under *piece-wise* linear indel costs. Bilinear indel costs can be modelled by a five-state machine (figure 8). We have modified the

```
states:
  S1: match/change state
  S2: (short-)insert state
  S3: (short-)delete state
  S4: long-insert state
  S5: long-delete state

transitions:
  S1, S2, S3 as before plus
  S1--long-ins-->S4   S1--long-del-->S5
  S4--long-ins-->S4   S4--end-long-->S1
  S5--long-del-->S5   S5--end-long-->S1

conservation:
  p(m|S1)+p(c|S1)+p(i|S1)+p(d|S1)+
  p(long-ins|S1)+P(long-del|S1)=1    etc.
```

Figure 8: Five-State Model

DPA to maintain five values, one for each state, in the D matrix. This kind of model is appropriate if short indels are thought to be relatively frequent and long indels rather infrequent which might account for point mutations and larger events, perhaps introns/exons. A fully-connected five-state machine would have a very large number of transitions and parameters. To reduce this we simply augment the three-state machine with long-insert and long-delete states connected only to S1.

As before, probabilities of alternative sequences are *added*. Note that because of this a medium length indel is considered to be partly long and partly short. Resulting indel costs are smooth and asymptotically linear; figure 9 plots the cost of stating the length of a run of indels.

The search for optimal five-state parameter values is more difficult. Because long-indels are expected to be rare they are initially given a low probability, a large start-up cost, but a high probability of continuation. A useful strategy is to run the three-state model first and base initial probabilities for S1, S2 and S3 on the results with minor adjustments to conserve probability. The iterative process is used from this starting point.

Note that a special case of five-state data is when the long indels occur at the ends of one or both strings, when string A=A1;A2;A3 and B=B1;B2;B3 where A2 and B2 are related by a three-state-model and A1, A3, B1 and B3 are unrelated, possibly null. This amounts to looking for a common *pattern* A2:B2 in A and B. This situation is rather common because interesting pieces of DNA from a chromosome may have flanking regions attached. Thus the five-state model generalises attempts to deal with end-segments in a special way as described by Allison and Yee (1990).

6. Model Comparisons.

The one, three and five-state algorithms were run on both artificial data and real DNA strings.

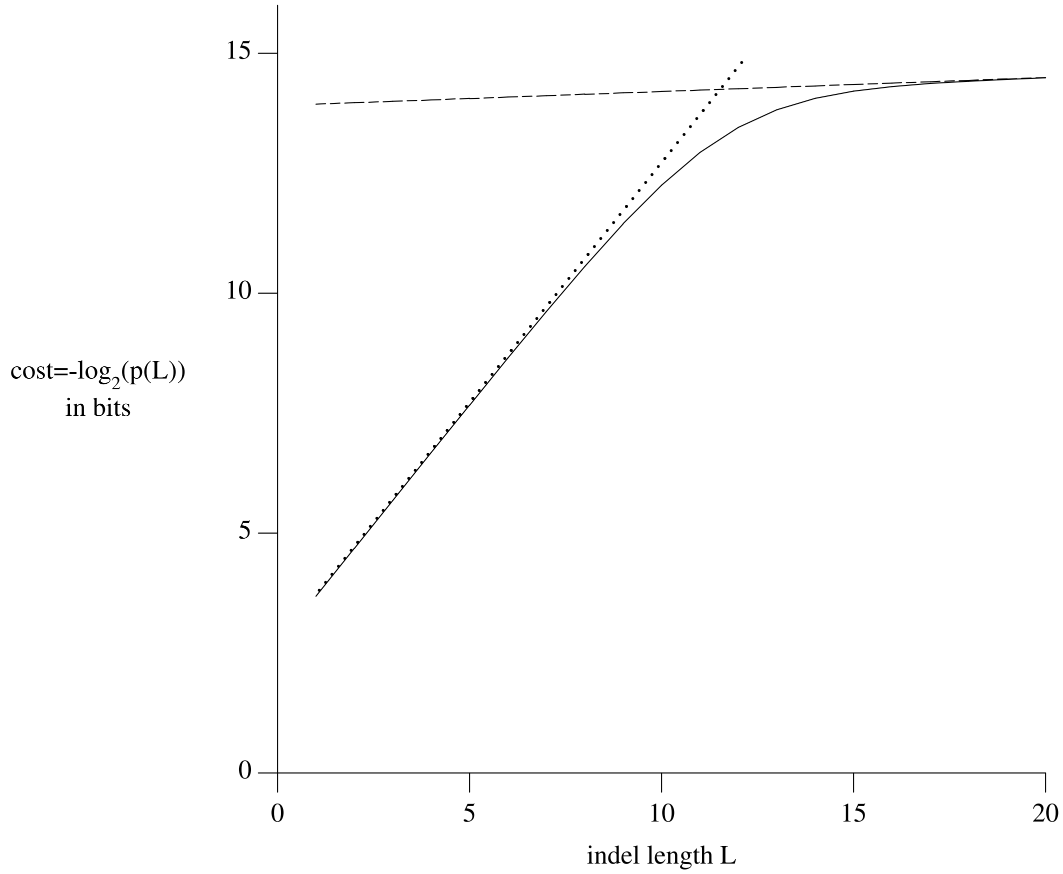
The cost of stating the parameters of a model is a decreasing fraction of the total message length as string lengths increase. For short strings, this may cause a simpler model to be favoured. To study this effect, pairs of strings of various lengths were *generated* by a five-state model with various known parameter values. For each setting, a collection of 15 to 30 pairs was generated. For each model and collection, the r-theory algorithm was run and the message lengths averaged. This was done for the one, three and five-state models.

Figure 10(a) gives plots of the total message lengths per symbol versus string length for strings generated by a five-state model having parameters indicated in table 1. In this case the probability of changes was held very low to isolate the interaction between short and long indels. There is a cross-over from favouring the three-state to favouring the five-state model at a string length of 500 to 1000 characters. A single pair of strings from an unknown source and shorter than this cannot be identified as coming from a five-state rather than a three-state model. The inference for long strings is stronger than might be immediately apparent because the cost per symbol is plotted.

A pair of strings longer than 500 to 1000 characters can be identified as coming from a five-state model. Many pairs of shorter strings, assumed to be generated by the same model and parameter values, can also be so identified. It is sufficient for the parameter values of the pairs to be related rather than identical in this latter situation so that they need not be stated in full for each pair. To indicate what would happen in an extreme case of this situation, figure 10(b) gives message lengths under the three models when parameter costs are excluded. The data is the same as for figure 10(a). In this case, the cross-over from the three-state to the five-state inference is at about 150 characters for the given parameter values. At this point a pair is reasonably likely to have at least one long insert or delete for the given parameter values.

Note that the DNA coding regions for many proteins have lengths in the difficult range of 500 to 1000 characters. Also the cross-overs would occur at shorter lengths for an alphabet size larger than 4.

(The bumps in figures 10(a) and 10(b) are of the order of 0.03 bits/char or about 2%; they are the inevitable result of using random strings. It is reassuring that the plots for the three algorithms closely parallel each other over the bumps.)



short: dotted, $p_s(L)=0.098\times0.50^{L-1}\times0.50$

long: dashed, $p_l(L)=0.002\times0.98^{L-1}\times0.02$

combined: solid, $p_c(L)=p_s(L)+p_l(L)$

Figure 9: Combining Length Costs of Long and Short Indels.

```
S1 (match/change):
    P(match)=0.8  P(change)=0.000001
    P(short insert)=P(short delete)=0.098
    P(long  insert)=P(long  delete)=0.002
S2 (short insert):
    P(match)=0.49  P(change)=0.000001  (return to S1)
    P(short indel)=0.5  P(short delete)=0.01
S3 (short delete): similar to S2
S4 (long insert):
    P(long insert)=0.98
    P(end long)=0.02   (return to S1)
S5 (long delete): similar to S4

average short indel length = 2
average long indel length = 50
```

Table 1: 5-State Parameters for Generating Test Data.

When the similarity within each pair of strings is increased, the general picture is as described above but the cross-overs occur at greater string lengths. Increasing similarity favours simpler models. In the extreme case, two identical strings are best described by the one-state model.

It is clear that to draw any general conclusions about the "true" model for *real* DNA, a pair of very long related strings or a large number of pairs of related strings is required. Such data are hard to come by.

Tests were done using chicken hemoglobin alpha and beta chains, Genbank identifiers CHKHBAM and CHKHBBM. The purpose was to compare the models and to illustrate the differences between conventional alignments and the r-theory. Waterman(1984, p484) recommends the alpha-chain subsequence A = TTTCCCCA ... TCAAATC (39 bases) and the beta-chain subsequence B = TTTGCGTC ... CATGGTC (57 bases) as a good pair for this kind of exercise. Table 2 summarises the results of running the one, three and five-state r-theory algorithms on the complete sequences CHKHBAM and CHKHBBM. The three-state model is best by a small margin and the five-state model is next best. Figure 11(a) shows the section of the probability density plot of all alignments, under the three-state model, covering the subsequences A and B. It is clear that there are many plausible alignments in the grey area and that it is unwise to support any single alignment strongly - unless there is compelling extra evidence. For comparison purposes, figure 11(b) shows an optimal alignment of the same section under Sellers' metric (no gap penalty) and figure 11(c) shows an optimal alignment under linear indel costs w(k)=3+k; it can be seen that the former infers many small gaps.

```
CHKHBAM: chicken alpha-globin mrna & 5' flank.  526 bp ss-mRNA
CHKHBBM: chicken hemoglobin beta chain mrna.  601 bp ss-mRNA

1-state   3-state   5-state
--------------------------> model
2263      2189      2200  (bits)

Null-theory: 2281 (bits)
```

Table 2: Message Lengths (in bits) for CHKHBAM and CHKHBBM.

```
CCAAGACCTACTTTCCC-CACTTCG--ATC-TGTC-A-C---ACGGC--TCC--GCTCAAATCAA-GG---G-GC
  | ||  | |||| |  | ||| |  | | | || | |   || ||  |||  |  |||  | | ||   | ||
--A-GAGGTTCTTTGCGTC-CTTTGGGAACCTCTCCAGCCCCACTGCCATCCTTGG-CAACCCCATGGTCCGCGC

--ACGGCAAGAAGGTAGTGGC
  ||||||||||   ||||
CCACGGCAAGAA---AGTG--

CHKHBAM[120..195] (top) : CHKHBBM[170-255]
```

Figure 11(b): An Optimal Alignment under Sellers' Metric.

```
CCAAGACCTACTTTCCCCACTTCGATCTGTCACACGGCTCCGCT--------CAAATCAAGGGGCACGGCAAGAA
  | |  ||      ||       | || ||   |  | | ||         |||  | | || |   ||
AGAGGTTCTTTGCGTCCTTTGGGAACCTCTCCAGCCCCACTGCCATCCTTGGCAACCCCATGGTCCGCGCCCACG

GGTAGT--GGC
|  ||   |
GCAAGAAAGTG

CHKHBAM[120-195] (top) : CHKHBBM[170-255]

indel cost w(k)=3+k
```

Figure 11(c): An Optimal Alignment under Linear Indel Costs.

The Blur sequences are members of the large Alu family (Bains, 1986; Jurka & Milosavljevic 1991). The typical Blur sequence is about 280 characters long. Table 3 gives some results of the one, three and five-state models on Blurs 8, 11 and 13. When whole strings are compared, the evidence strongly supports either the three or five-state model rather than the one-state model. (The null-theories are considerably poorer, of the order of 1200 bits.) The three and five-state models give message lengths within a few bits of each other which is insufficient to conclude in favour one rather than the other. Blur11' and Blur13' were derived by forming an alignment and truncating any terminal overhanging segment on either string. Results for this pair slightly favour the one-state model suggesting that the overhangs were the dominant factor in the whole strings. The difficulty is probably that the Blur's are too short.

The globin genes and pseudogenes are often used in phylogenetic studies of primates (Holmes, 1989). Table 4 gives the results of the r-theory algorithms of the three models when run on human beta

	1-state	3-state	5-state
	--------------------------> model		
Blur8 :Blur13	1039	946	943
	(1016)	(912)	(909)
Blur11 :Blur13	1179	963	957
	(1155)	(928)	(922)
Blur11':Blur13'	825	830	836
	(801)	(800)	(802)

Lengths in parentheses are exclusive of parameter costs.

Table 3: Message Lengths (in bits) for some Blur Sequences.

type globin pseudogene, orangutan beta type globin pseudogene and owl monkey psi beta 1 globin pseudogene. The three and five-state models give similar results and are much better than the one-state model. These three strings are very similar and the null-theory messages are much longer, in excess of 8,000 bits.

The above results strongly suggest that the one-state model is not the correct one to use with DNA. They do not distinguish between the three and five-state models. It is clear that a great many more tests are required to discover if one of these, or some other model, is the correct one to use.

7. A Fast Approximate Algorithm.

The basic DPA takes O(|A|×|B|) time and space. If a single optimal MML alignment is required, Hirschberg's (1975) divide and conquer technique can be used to reduce the space complexity to O(|B|). This is not applicable if the MML of the r-theory is required. Observations of alignment density plots (figure 6) indicate that most of the probability is concentrated in a narrow region. Fast approximate MML algorithms can take advantage of this fact.

	1-state	3-state	5-state	
HSBGLOP:ORAHBBPSE	5209	4998	5008	including parameters
	(5177)	(4956)	(4960)	excluding parameters
HSBGLOP:ATRB1GLOP	6414	6004	6007	
	(6383)	(5960)	(5958)	
ORAHBBPSE:ATRB1GLOP	6369	5976	5973	
	(6339)	(5932)	(5924)	

Genbank id HSBGLOP: human beta type globin pseudogene 2153bp.
Genbank id ORAHBBPSE: orangutan beta type pseudogene bases 1593 to 3704, 2111bp.
Genbank id ATRB1GLOP: owl monkey psi beta 1 globin pseudogene 2124bp.

Table 4: Message Lengths (in bits) for pairs of Globin Pseudogenes.

An approximate three-state algorithm has been programmed. Firstly, Ukkonen (1983) gave an O(|A|×D(A,B)) edit-distance algorithm for Seller's metric. This is fast for similar strings. The algorithm does not calculate the D matrix directly but rather the location of *contours* in it. The algorithm halts when a contour reaches D[|A|,|B|]. We applied Ukkonen's technique to linear indel costs. Secondly, as mentioned in section 3, arbitrary costs in a metric can be normalised to yield instruction probabilities. This process can be reversed to choose integer weights corresponding to "reasonable" probabilities. An optimal alignment is found for these weights by Ukkonen's technique. Finally, a window containing all D cells that lie within a diagonal distance 'w' of the optimal alignment is formed and the r-theory algorithm is run only within this window. All cells outside this window are considered to have probability zero and an infinite message length. The time and space taken for this step is O(max(|A|,|B|)×w). A graph of message lengths against w is given in figure 12. The random data was generated by a three-state model. The dominant influence on required window size is the probability of a match in state 1 which is indicated. It can be seen that a small window includes almost all the probability. This windowing technique could be applied to other models. It is not reliable when the strings are only marginally similar.

8. Conclusions.

We have solved the question of two strings being related (the r-theory) for one, three and five-state models of string relation. The biological validity of such models is open to debate but we note that one and three-state models are implicit in alignment algorithms in common use, often with various weights to adjust in an *ad-hoc* way. A major advantage of a formal model is that its assumptions are explicit and available for scrutiny and modification. We have given an objective method of comparing models of relation over a given set of strings and have applied it to the one, three and five-state models. Results on artificial data suggest that the "true" model of relation can be inferred, given enough data.

The tests performed are only a first step and the models can be elaborated to account for unequal probabilities of changes, effects of codon bias, the genetic code and so on. The assembly of a large body of real biological test data remains a problem. Pairs of strings should be selected from many families to avoid bias.

In general, a single alignment is excessively precise. All alignments are possible; some are more probable than others. The r-theory should include them all in a weighted fashion.

Modulo normalisation, an arbitrary cost function w(L) on blocks of matches, changes, inserts or deletes of length L implies a prior probability distribution $2^{-w(L)}$ on run lengths. Any knowledge specific to the problem can be used to design a good prior. Conversely if a cost function gives results judged (externally) to be good, it gives information about good priors and mechanisms relating strings. In this way an expert's knowledge and intuition can be made explicit.

A piece-wise linear cost function on runs of instructions implies a finite-state machine or model. All such models have an O(|A|×|B|) MML algorithm for the r-theory and for alignment probability densities. However, the number of transitions in the machine, and thus the complexity of the algorithm, rapidly increases with the number of linear segments making the cost function. We note that Gotoh(1990)(p366) feels that piece-wise linear function with as few as two segments, equivalent to the five-state machine, are sufficient for DNA. Given enough data, experiments on machines with up to seven-states could confirm or deny this. The r-theory algorithms perform floating-point operations in their inner loops and are typically two to ten times slower than DPA's that use only integer arithmetic, depending on the computer used. (Integer approximations are possible.) Miller and Myers (1988) have given an O(|A|×|B|) or O(|A|×|B|×log(max(|A|,|B|))) algorithm to find a single optimal alignment for arbitrary concave functions. There does not seem to be an r-theory algorithm or a density-plot algorithm with this complexity for such functions.

Given a single pair of strings from an unknown source, over an alphabet of size four and of length less than 500 to 1000 characters it is typically not possible to reliably identify them as being the products of a five-state process even if they are. Longer strings or many pairs of shorter strings generated by the same five-state process can be so identified.

The algorithms described are not suitable for fast data-base searches, except possibly using an array processor or other special processor. They can be used for ranking the matches found by a fast heuristic algorithm. They are intended for use on small numbers of strings, for inferring parameters of mutation or for investigating models of relation and mutation.

Programs for the one, three and five-state models have been written in C and are available, free of charge, for non-profit, non-classified research purposes, by email only, from lloyd@bruce.cs.monash.edu.au. Their use requires, at the least, a good work-station with floating-point hardware.

9. References.

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

Allison L., C. S. Wallace & C. N. Yee (1990). When is a string like a string?
Proc. Artificial Intelligence and Mathematics, Ft. Lauderdale,

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

Bains W. (1986). The multiple origins of the human Alu sequences.
J. Mol. Evol. **23**, 189-199.

Bishop M. J. & A. E. Friday (1986). Molecular sequences and hominoid phylogeny.
Major Topics in Primate and Human Evolution. B. Wood, L. Martin & P. Andrews (eds). CUP 150-156.

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

Bishop M. J. & C. J. Rawlings (eds) (1987). *Nucleic Acid and Protein Sequence Analysis, a Practical Approach.*
IRL Press.

Boulton D. M. & C. S. Wallace (1969). The information content of a multistate distribution.
J. Theo. Biol. **23**, 269-278.

Boulton D. M. & C. S. Wallace (1973). An information measure for hierarchic classification.
Comp. J. **16**, 254-261.

Chaitin G. J. (1966). On the length of programs for computing finite binary sequences.
J. Assoc. Comput. Mach. **13**(4), 547-569.

Cohen D. N., T. A. Reichert & A. K. C. Wong (1975). Matching code sequences utilizing context free quality measures.
Math. Biosci. **24**, 25-30.

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

Georgeff M. P. & C. S. Wallace (1984). A general selection criterion for inductive inference.
Proc. European Conference on Artificial Intelligence, 473-482.

Gotoh O. (1982). An improved algorithm for matching biological sequences.
J. Mol. Biol. **162**, 705-708.

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

Hamming R. W. (1980). *Coding and Information Theory.*
Prentice Hall.

Hirschberg D. S. (1975). A linear space algorithm for computing maximal common subsequences.
Comm. Assoc. Comp. Mach. **18**(6), 341-343.

Holmes E. C. (1989). *Pattern and Process in the Evolution of the Primates*.
PhD thesis Cambridge University.

Jurka J. & A. Milosavljevic (1991). Reconstruction and analysis of human Alu genes.
J. Mol. Evol. **32**, 105-121.

Kolmogorov A. N. (1965). Three approaches to the quantitative definition of information.
Problems of Inf. & Transmission **1**(1), 1-7.

Langdon G. G. (1984). An introduction to arithmetic coding.
IBM J. Res. & Dev. **28**(2), 135-149.

Li M. & P. M. B. Vitanyi (1988). Two decades of applied Kolmogorov Complexity.
Proc. of the Third Annual Conference on Structure in Complexity Theory. IEEE, 80-101.

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

Milosavljevic A. D. (1990). *Categorization of Macromolecular Sequences by Minimal Length Encoding*.
PhD thesis University of California at Santa Cruz, UCSC-CRL-90-41.

Needleman S. B. & C. D. Wunsch (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins.
J. Mol. Biol. **48**, 443-453.

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

Rissanen J. (1983). A universal prior for integers and estimation by minimum description length.
Annals Stats. **11**(2), 416-431.

Sankoff D. & J. B. Kruskall eds (1983). *Time Warps, String Edits and Macro-Molecules.* Addison Wesley.

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

Sellers P. H. (1980). The theory and computation of evolutionary distances: pattern recognition. J. Algorithms **1**, 359-373.

Shepherd J. C. W. (1981). Method to determine the reading frame of a protein from the purine/pyrimidine genome sequence and its possible evolutionary justification. Proc. Nat. Acad. Sci. **78**, 1596-1600.

Solomonoff R. (1964). A formal theory of inductive inference, I and II. Inf. & Control **7**, 1-22 and 224-254.

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

Ukkonen E. (1983). On approximate string matching. *Proc. Int. Conf on Foundations of Computation Theory.* M. Karpinski (ed), Springer Verlag LNCS **158**, 482-495.

Wallace C. S. & D. M. Boulton (1968). An information measure for classification. Comp. J. **11**(2), 185-194.

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

Wallace C. S. (1990). Classification by minimum message length inference. AAAI Spring Symposium on the Theory and Application of Minimum Length Encoding, Stanford, 5-9.

Waterman M. S. (1984). General methods of sequence comparison. Bull. Math. Biol. **46**(4), 473-500.

Witten I. H., R. M. Neal & J. G. Cleary (1987). Arithmetic coding for data compression. Comm. Assoc. Comp. Mach. **30**(6), 520-540.

Wong, A. K. C., T. A. Reichert, D. N. Cohen & B. O. Aygun (1974). A generalized method for matching informational macromolecular code sequences. Comp. in Biol. & Med. **4**, 43-57.