# Modelling-Alignment for Non-Random Sequences

David R. Powell*[1,2], Lloyd Allison[1] and Trevor I. Dix[1,2]

[1] School of Computer Science and Software Engineering,
Monash University, Australia 3800
[2] Victorian Bioinformatics Consortium
{powell, lloyd, trevor}@csse.monash.edu.au

**Abstract.** Populations of biased, non-random sequences may cause standard alignment algorithms to yield false-positive matches and false-negative misses. A standard significance test based on the *shuffling* of sequences is a partial solution, applicable to populations that can be described by *simple* models. Masking-out low information content intervals throws information away. We describe a new and general method, *modelling-alignment*: Population models are incorporated into the alignment process, which can (and should) lead to changes in the rank-order of matches between a query sequence and a collection of sequences, compared to results from standard algorithms. The new method is general and places very few conditions on the nature of the models that can be used with it. We apply modelling-alignment to local alignment, global alignment, optimal alignment, and the relatedness problem.

**Results:** As expected, modelling-alignment and the standard `prss` program from the FASTA package have similar accuracy on sequence populations that can be described by simple models, e.g. 0-order Markov models. However, modelling-alignment has higher accuracy on populations that are *mixed* or that are described by higher-order models: It gives fewer false positives and false negatives as shown by ROC curves and other results from tests on real and artificial data.
An implementation of the software is available via the Web [3].

## 1 Introduction

Sequence alignment is an important method for discovering patterns and relationships between sequences. The alignment of biological sequences can give clues to common ancestry or to common biological function. Alignment algorithms are used to answer two quite different kinds of questions, (i) Are two (or more) sequences *related* or not? (ii) *How* are two (or more) sequences related, *given* that they are related? There is a "fundamental difference between the searching [for related sequences] and [optimal] alignment operations." – [14]. It is also well known that populations of biased, non-random, low information content sequences may cause algorithms to return false-positive matches and false-negative misses for the first kind of problem and may also cause poor alignments, for the second kind of problem. There are partial solutions, with disadvantages: Masking-out low information content intervals [8] can reduce false-positives but it throws away some information and raises the question of how low is

* Partially funded by Australian Research Council Grant A49800558.
[3] http://www.csse.monash.edu.au/∼powell/m-align.tar.gz

low? The significance of a possible match is often assessed by comparing its score (or cost) with those obtained when "one member of the protein [say] pair [is] randomized" (shuffled, permuted) – [21]. We call this general kind of significance testing *shuffling*; note that an alignment is calculated under the assumption of random sequences and *then* the shuffling test is applied, possibly to a bad alignment if the sequences are non-random.

We describe a new method, *modelling-alignment* (M-alignment) which incorporates explicit models of sequence populations. It is based on the premise that two sequences are related if one tells something *new* and *useful* about the other, i.e. something not otherwise known. It has a natural null-theory and hypothesis test. Instead of a low information interval being masked-out, it is given a low, but non-zero, weight thus not discarding information.

A high information interval, i.e. a *feature*, is given a high weight; there is a big scoring advantage if it is matched in another sequence. In general this can, and should, change the rank-order of alignments between two sequences and the rank-order of matches between multiple sequences. There is no hard (and arbitrary) threshold for what is a *background* low information interval or for what is a high information feature; there is a continuous spectrum of information content from low to high.

M-alignment is general and places few restrictions on the kind of population model that can be used with it. If the "correct" model is not known in advance there are some sensible candidates to try and a good model can be detected on the basis of information content. Algorithm time-complexity remains $O(n^2)$ for reasonable models. This paper extends work on the 1-state mutation model [2] by applying M-alignment to the relatedness and optimal alignment problems, for both global [21] and local [27] alignment, under the 3-state, affine gap-costs model [13] of mutation.

M-alignment results are compared to those of the `prss` program – from the FASTA package [22] – which is based on `rdf2` [23] with a shuffling significance test. Receiver operating characteristics (ROC) curves show that, as expected, both methods give similar results on simple data, i.e. uniform and 0-order Markov model populations. M-alignment is more accurate, i.e. gives fewer false positives and fewer false negatives, on mixed populations, on populations of mixed sequences, and on populations described by higher-order models.

At this point it is convenient to clarify some of the terminology we use in this paper. We use the term *rank-order* to mean order obtained when items are ranked according to some score. When a group or population of sequences is well explained by a particular model, we say that model is a *population model* for those sequences. An alignment can be seen as a hypothesis of how two (or more) sequences are related. Using information-theoretic techniques we can quantify how likely a hypothesis is, and thus discuss the *probability of an alignment*.

## 2 Systems and Methods

### 2.1 Models

There is considerable interest in the compression of biological sequences [15, 19, 24, 3], not so much to save disc space but as a criterion for comparing models of populations of sequences. The terms *biased*, *non-random*, *low information content* and *compressible* are equivalent for our purposes: If DNA bases, say, are generated uniformly at random, an optimal code assigns each one a two-bit code. This follows from Shannon's mathematical theory of communication [26]. If the DNA comes from a biased 0-order model say, shorter codes can be allocated *on average*. For example, if the respective probabilities of an A, T, G or C are [1/2, 1/4, 1/8, 1/8] this leads to optimal codes of [1, 2, 3, 3]-bits respectively, that is 1 3/4-bits on average. For higher-order models, code-words depend on the *context* of previous characters, on the 'k' previous characters for a Markov model of order k. Probabilistic finite-state automata (PFSAs), also known as hidden Markov models, have been used as models [12] of populations of sequences. Each of these models, and many others, can deliver a probabilistic prediction for the next character in a sequence given a context. From Shannon, the length of a code-word for the next character, its *message length*, is the -log of this probability and is a measure of the information content of the character in the context. M-alignment uses this quantity.

### 2.2 Relatedness and Alignment

The generic dynamic programming algorithm (DPA) for sequence comparison can be used to find various kinds of global alignments depending on how it is instantiated: E.g. Given *scores* of one for a match and zero for mismatch, insert and delete, the DPA finds the longest common subsequence (LCS). Given *costs* of zero for a match and one for mismatch, insert and delete, it finds an optimal alignment under the Levenshtein [18] metric also known as the simple edit-distance [25]. *Affine* gap-costs, for runs of inserts or deletes, are more plausible biologically and Gotoh [13] gave such an alignment algorithm; the key is to have three *states* for each cell in the DPA's matrix, for costs (or scores) *conditional* on diagonal, vertical and horizontal moves.

All of the above DPA variations find a global optimal alignment for some given criterion, or at least under the assumption that, two sequences are related. Shuffling tries to solve the relatedness problem by comparing the score (or cost) of an optimal alignment with the scores (costs) of alignments of the randomized sequences. If the former is not significantly better than the latter the sequences are deemed to be unrelated. The idea is that randomized sequences are *like* their originals in general statistical terms but are otherwise unrelated. Shuffling preserves 0-order statistics of sequences. It can be arranged to preserve 1st-order statistics [11] and even codon usage [5] but it is hard to imagine how to carry it out while preserving the statistics of an arbitrary model particularly if that is a mixture or is high-order.

A different approach to the relatedness problem considers a probabilistic *mutation model*. The costs (scores) of the DPA are replaced by the -log probabilities of match, mismatch, insert and delete. In fact costs and scores can be *normalized* [1] to show the underlying probabilities. An alignment may be seen as just a hypothesis about how sequences are related, and alignment algorithms as encoding the sequences and an alignment in an efficient manner. The alignment that leads to the shortest encoding is the best inference in a statistical sense [28]. The DPA can be used to find a *most probable* alignment, but it can also be modified to calculate the *joint* probability of two sequences:

An alignment is just a hypothesis. The set of all alignments is exclusive and exhaustive, given that two sequences are related. Rather than selecting the largest probability, their probabilities can therefore be *added*.

This was originally done for a 1-state mutation model [6]. Later, it was extended [4] to 3-state (affine gap-costs) and 5-state (piece-wise linear gap-costs) mutation models, and included the cost (complexity) of models so that simple and complex models can be compared fairly. Such considerations also give a natural *null-theory* that the sequences are not related, i.e. that one tells nothing new or useful about the other. Two sequences are unrelated if their joint probability is less than or equal to the product of their individual probabilities, given by the null-theory.

The next section describes how these methods are extended to local alignment and how the information content of compressible sequences is taken into account in M-alignment

# 3 Algorithm

The M-alignment algorithms (i) make use of probabilistic alignment methods for global and local alignment, and (ii) incorporate sequence population models to calculate the information content of characters in context. This is done in combination with the 3-state, affine gap-cost mutation model.

Firstly, probabilistic methods are used to estimate the probability of relatedness of sequences S1 and S2 under *local alignment*, i.e. that there is some configuration S1=A+L+B and S2=C+L'+D where intervals L and L' are related (globally) and intervals A, B, C, and D, which are possibly empty, are not related. A, B, C and D are compressed with the population model. L and L' are compressed with both the population and mutation models, in a way to be described. For the relatedness problem we *sum* over all such configurations, that is over all A, L, B, C, L' and D and all alignments of L with L', still in $O(n^2)$-time. For the optimal local alignment problem we choose the best such configuration.

Secondly, the DPA operates cell by cell on a two-dimensional array. For affine gap-costs, each cell contains three values for the -log probability of S1[1..i] and S2[1..j] conditional on the last operation being an insert, delete or match/mismatch. Each increment involves the -log probabilities of a mutation and of a character from one or both strings as appropriate. The key idea of M-alignment is to obtain the latter from a population model.

Then contribution of an insertion or deletion is calculated as the product or the mutation probability (more on this later), and probability of the character being inserted or deleted. This *character probability* is calculated by using the *population model*. A mismatch is calculated in a similar fashion, except using two probabilities, one for each character. The calculation of a match is slightly different since we have two different probabilities for the same character. For convenience, the probability of a match is multiplied by the *average* of the two character probabilities.

There is the problem of what mutation probabilities to use. This could be addressed by using fixed probabilities as is effectively done with scores in the Smith-Waterman algorithm. Alternatively, and as we do here, the mutation probabilities can be estimated by the algorithm. We do this by starting the probabilities at sensible values, then running the algorithm a number of times re-estimating the mutation probabilities each time. This is simple expectation maximisation which is guaranteed to converge in this case, possibly to a local maximum. The initial mutation probabilities used are calculated by transforming Smith-Waterman costs into probabilities [1]. Note that a side-effect of this is that an explicit probability is given to terminal gaps rather than the somewhat arbitrary zero used in the Smith-Waterman algorithm.

This explanation of the M-alignment algorithm is for the particular case of DNA sequences where the mutations are insertion, deletion, mismatch or match. Instead of simply considering matches and mismatches, it is sometimes useful to use a 4x4 grid of scores to assign a score for aligning each DNA base against each other DNA base. This is very similar to the use substitution matrices when aligning proteins (BLOSUM, PAM, etc.). The modification of M-alignment for this is given in Section 6.

## 4 Implementation

A version of the DPA was implemented to carry out M-alignment for a 3-state, affine gap-costs mutation model, local alignment. It accepts a population model as a parameter. It is able to return either the -log probability of two sequences being related (by summing alignment probabilities) or an optimal local alignment. The results from testing this program on both real and artificial data are given in the next section.

## 5 Tests

On the face of it M-alignment appears to address a similar problem as Hidden Markov Model alignment algorithms such as HMMER [10]. Indeed, HMMER and M-alignment have a number of similarities, both use probabilistic modelling, both sum over all possible alignments. *However*, they are for different problems and are therefore difficult to compare fairly. HMMER builds a profile HMM, ideally from a multiple sequence alignment. It is then possible to use this HMM to align against a sequence or search a database. This differs from M-alignment because M-alignment aligns two actual sequences in the presence of a model of the population. However, since M-alignment may use almost any type of model, it would be possible for it to use a HMM profile model, although it is not clear whether this would be useful.

### 5.1 Generation of Related Sequences

It is easy to generate a typical sequence given a population model. The difficulty is to generate *related pairs* that are typical of the population; uniform random mutations would cause descendants to drift towards the statistics of a uniform model. The solution is to propose a mutation at random and to either accept it or reject it.

We have a sequence, $A$, and a population model $M$. The entropy of $A$ under this model shall be written $M(A)$. The length of sequence $A$ will be written $|A|$. The sequence that results from making $x$ mutations to $A$ shall be referred to as $A_x$. We earlier described the motivation as wanting the mutated sequence to "fit" the population model. To make this precise, what is desired is that as $x$ tends to infinity, the entropy of the mutated sequence, $M(A_x)$, will tend to the entropy of the population model, $M$, itself.

The method used to mutate sequences was inspired by the Metropolis algorithm [20]. Consider every possible sequence as having a node, $S_i$, in a graph. An arc between two nodes, $S_i$ and $S_j$, implies that a single mutation relates the two corresponding sequences. As we make mutations to a sequence, we move between nodes in the graph. In the extreme case where many mutations are made, we want to visit each node with a frequency equal to probability $p$, where $p$ is probability of that node's sequence under the population model. A sufficient condition for this is to have the probability of taking the arc from $S_i$ to $S_j$ equal to the probability of taking the arc from $S_j$ to $S_i$.

We will assume a point mutation model for simplicity. That is, we consider making mutations of the form delete a character, insert a character or change a character. For an alphabet of 4 characters, sequence $A$ has $3\times|A|$ possible change mutations, $|A|$ possible delete mutations and $4 \times (|A| + 1)$ possible insert mutations. If we assume a uniform, or relatively flat, prior on the length of sequences, then we have the requirement when mutating a sequence that on average the number of deletions should be the same as the number of insertions. There is a degree of freedom in choosing the ratio of changes to insertions and deletions. We shall use a probability parameter $P_{change}$ for this. We define two other probabilities in terms of this.

$$P_{insert} = \frac{(1 - P_{change})(4 \times (|A| + 1))}{(5 \times |A| + 4)}$$

$$P_{delete} = \frac{(1 - P_{change})}{(5 + 4/|A|)}$$

When we mutate sequence $A$, a random choice is made whether to consider an insertion, a deletion or a change mutation using the three above probabilities (note they sum to unity). If a deletion is to be considered, then the position of the deletion is chosen uniform randomly over the length of the sequence. If a change, then the position and character are chosen randomly. If an insertion, then the position and the character to be inserted are randomly selected. The sequence that would result from this mutation is $A'$. Let $q$ be the probability given to sequence $A$ by the population model, and $q'$ the probability given to sequence $A'$. If $q'$ is greater than $q$ then the mutation is accepted and $A'$ becomes $A_1$. Otherwise, the mutation is accepted with probability $q'/q$. This process is repeated until the desired number of mutations has been made.

Since this method does not preclude a mutation "undoing" previous mutations, the resulting mutated sequence, $A_x$, may be more similar to $A$ than otherwise expected. This tends to be more likely if $M$ is a very low entropy model.

In the following tests we wish to examine the performance of M-alignment with local alignment. So, we generate related pairs of sequences, L and L', using the above method. Then, unrelated prefixes A and C and suffixes B and D are generated from the population model.

### 5.2 Results with Artificial Data

Artificial data was used to compare the new M-alignment algorithm with the common Smith-Waterman algorithm using shuffling as a significance test. The benefit of using artificial data is that the conditions can be controlled precisely to clearly illustrate the advantages and disadvantages of techniques. We also know what the answer is.

The implementation used for the Smith-Waterman algorithm was the `prss` program that is part of the FASTA 3.3 package. In all tests, the `prss` program was used with the standard parameters: +5 for a match, -4 for a mismatch, -16 and -4 for the first and subsequent characters in a sequence respectively. The default of 200 uniform shuffles was used to determine the significance of each alignment.

The artificial data was generated using a number of different population models. In all cases, 10 parent sequences were generated from the population model(s). From each parent, 25 child sequences were generated having a mutated subsequence in common with parent. The amount of mutation was varied so some children were similar to their parent and some were very dissimilar. The method of mutation was described in Section 5.1.

The child sequences were considered as the library to be searched, and each parent was used in turn as the query sequence. Thus for each query there were 25 related sequences of differing relatedness and 225 unrelated sequences. This ratio of related to unrelated sequence in the library is high compared to real sequence databases but will suffice for testing purposes. Each parent sequence was compared against every child sequence making 2500 pair-wise comparisons. Of these 2500 comparisons, 250 are between related sequences, and 2250 between unrelated sequences.

To present the results of the tests Receiver Operating Characteristics (ROC) [14, 7] plots are used. Each algorithm tested produces a number measuring the relatedness of the two sequences being compared. This may be the raw Smith-Waterman score, or the p-value found by the `prss` program, or the log odds ratio produce by M-alignment. For each test, the 2500 pair-wise comparisons are ranked in order of significance by this measure. For each possible cut-off value the number of true positives and the number of false positives are counted. Plotted on the y-axis is the ratio of false positives to the total number of unrelated pair-wise comparisons. And on the x-axis the ratio of true positives to the total number of related comparisons. The better an algorithm is at separating related from unrelated sequences the closer its curve will be to the bottom right corner of the ROC plots. Since we are mostly interested in the behaviour for a smallish number of false positives the y-axis is plotted with a log scale. Recall that M-alignment may take as a parameter the model for the sequences. Except where noted otherwise, the M-alignment algorithm is told to fit a 1st order Markov Model to the sequences.

In tests with a simple unbiased model, (i.e. each character occurs with probability 1/4) the raw Smith-Waterman score, the `prss` program and M-alignment perform similarly. For a population model that produces sequences with a simple composition bias, tests showed that the `prss` program and M-alignment perform in a similar manner, while raw Smith-Waterman score is inferior.

Figure 1 shows the results of a test that combines unbiased and biased sequences. Instead of all 10 parent sequences coming from the same population model, 5 come

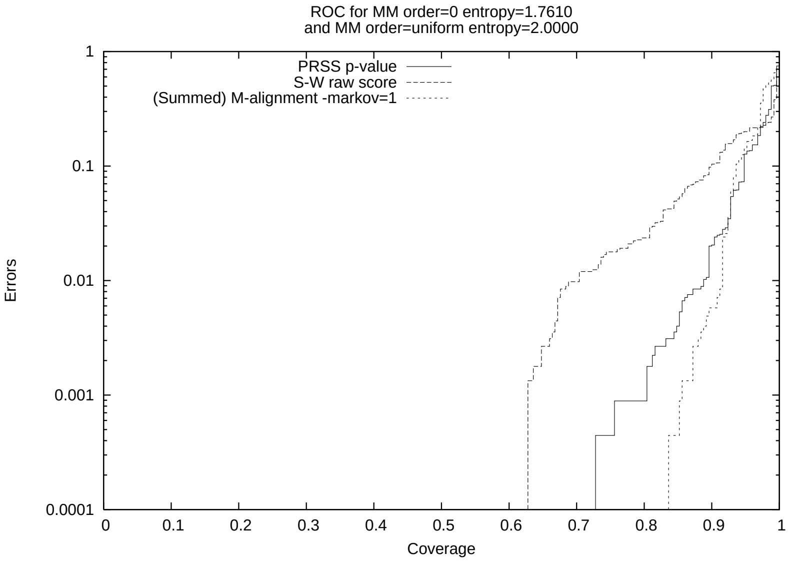


**Fig. 1.** ROC for a uniform sequence model and biased composition sequences.

from an unbiased model, and 5 from a 0th order Markov Model (simple composition bias). The shuffling of the `prss` program does account for sequences with a biased composition, but as seen in this test it does not perform as well as M-alignment when there are different models in the population. Similar behaviour has been seen when using other combinations of models.

The final test illustrates the benefits if one is able to better model the population. The population model in this test is a little more complicated. This model consists of two sub-models. One is a simple uniform model, and the other a very biased 1st order Markov Model that produces sequences of characters that are TATA-rich. Sequences are generated by choosing one of these two sub-models at random, then using that sub-model to produce a random number of characters. This is repeated a number of times to produce a sequence of sufficient length. This model is, in a sense, a blend between a uniform model, with an entropy of 2 bits per character, and a 1st order model, with an entropy of 1.1 bits per character. Figure 2 shows the results of the different algorithms on sequences of this type. Our algorithm is performing significantly better than the `prss` program. The best performer in this figure is the M-alignment algorithm using the "blendModel". This blendModel was designed with some knowledge of the population model. Specifically, the blendModel knows that the data is produced by a combination of a uniform model and also the exact 1st order model used. It does not know the probability with which these sub-models are chosen, nor the length of the sequences produced by the sub-models. As can be seen in the figure, having this extra knowledge about the population model allows for better separation of related and unrelated sequences. The M-alignment algorithm is superior to the Smith-Waterman algorithms because an arbitrary left-to-right sequence model can be used. Thus, more

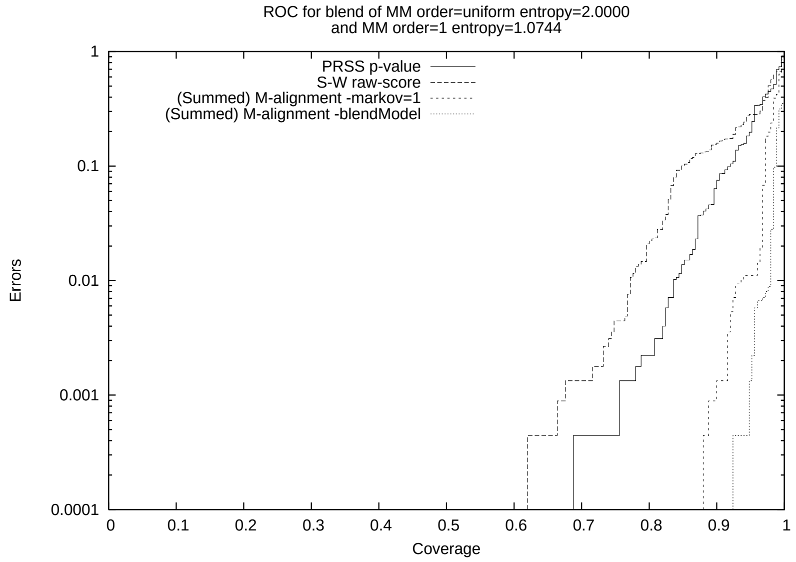


**Fig. 2.** ROC for a blended model

complex population models may be used with M-alignment to allow better differentiation between related and unrelated sequences.

It may be argued that this test is unfair to the `prss` program since it would be common practice to mask out low-complexity regions before doing the comparison with the `prss` program. However, these low-complexity regions do give *some* information about whether the sequences are related, just not as much information as high-complexity regions.

It is clear from the ROC plots that M-alignment is superior to the `prss` program when sequences are of low complexity, even when it is simply a mixture of uniform and biased composition sequences. However, the `prss` program performs well on these types of sequence if the population is confined to one type. This may be explained because the `prss` program does not have a natural cut-off p-value, and indeed the best cut-off depends on the type of the population. M-alignment has a natural cut-off at a log-odds ratio of zero. At zero, both the null model and the alignment model are equally good explanations of the data. In practice, a slightly larger cut-off value would be used; a cut-off of 3 bits indicates the alignment model is 8 times better than the null model, a cut-off of 10 bits gives a confidence of about 99.9% provided population model assumptions are appropriate.

### 5.3 Results with Real Data

In this section we show that M-alignment also performs well on real genomic DNA sequences. We chose a relatively low information sequence from *Plasmodium falciparum* chromosome 2. The sequence is from the end of the intron and start of the second exon of PFB0010w and of 157 characters in length. This sequence was chosen because PFB0010w is known to be related to a number of genes and pseudo-genes and because it has considerable repetition. Thus this is the type of sequence we expect traditional algorithms to have difficulty with.

This sequence was used in a nucleotide-nucleotide BLAST search against the `nr` database which was run using the NCBI BLAST server. The default options were used with the exception that filtering was turned off. This search resulted in 60765 "matching" sequences — no matches were found when filtering was on. Each of these 60765 sequences were retrieved with an extra 50 characters at each end so the alignment programs could have some extra context information. The vast majority of these sequences are in fact unrelated to the query sequence. Most of these false positives are due to matching `TATA` regions. Using M-alignment with a sequence model that correctly apportions weight to such regions should give good results. The model we used, was a fifth-order Markov Model trained on all 60765 matching sequences. A fifth-order model was chosen because it achieved the best compression across the whole population taking into account the complexity of the model. While such a model is not ideal, we assume that most of the sequences are unrelated but do represent a certain population, and that a fifth-order model can capture much of the information about this population. If a sequence is to be said to be related to the query sequence then it must have more in common than simply coming from that population.

The `prss` and M-alignment programs were then used to do a pair-wise comparison between the query sequence and each of the 60765 matching sequences. Thus, a ranking of all the matching sequences was produced by both programs. M-alignment gives roughly 1300 sequences a significant score, while `prss` claims more than 10000 significant matches. It is difficult to assess objectively which ordering is better. Our subjective comparison of the two rankings suggests that the M-alignment ranking is superior.

A fragment was taken from pseudo-gene PFB0045c [17], which is known to be distantly related to the gene the query sequence was taken from, PFB0010w. This fragment comes from the region of PFB0045c that corresponds to the intron of PFB0010w. The fragment was not found by the BLAST search. If this fragment is ranked amongst the other sequences it would, ideally, rank fairly high since the vast majority of the 60765 sequences are unrelated to the query sequence. The `prss` program ranks this fragment at 25499 of 60765, while the M-alignment program ranks it at 241.

## 6 Proteins

Our use of the M-alignment algorithm has concentrated on DNA sequences thus far, and our implementation is solely for DNA sequences. Here we will briefly describe how it would be possible to extend M-alignment to protein sequences.

When aligning proteins it is not sufficient to simply have a mismatch score (or cost) unlike for DNA. Protein alignment commonly uses a scoring matrix such as PAM [9] or BLOSUM [16] which gives a score for aligning any pair of amino acids. The BLOSUM matrices use the BLOCKS database to calculate the joint probability of two amino-acids occurring together, this is then normalised to account for the bias in appearance of each amino-acid. These are not directly useful to M-alignment. For M-alignment it is necessary to have a *conditional probability* for each pair of amino-acids. So, if we have an alignment that aligns glycine with leucine we need $p(glycine|leucine)$ and $p(leucine|glycine)$, note that these probabilities may be different. It should be straightforward to derive these probabilities from the BLOCKS database. The bias in the appearance of amino acids is then taken into account by the population model.

There are two possible ways to encode each pair of amino acids. We define $Pr(x,y)$ as follows to be the average of these.

$$\begin{aligned} Pr(x,y|a[1..i-1],b[1..j-1]) = \frac{1}{2}[&p(y|x)Pr(x|a[1..i-1]) \\ &+p(x|y)Pr(y|b[1..j-1])] \end{aligned}$$

It is not clear what type of population model would work well for proteins. We have concentrated on DNA sequences thus far because DNA sequences typically have a larger variation in information content than protein sequences.

## 7 Conclusion

M-alignment in global- and local-, sum-of-all-alignment form performs well at predicting relatedness for various kinds of population. On uniform random populations, accuracy is equivalent to the standard Smith-Waterman program with shuffling; it is hard to see how a significance test based on shuffling could cope well with arbitrary models of populations. M-alignment performs better, as shown by ROC curves, for compressible (non-random) populations if the true population model, or a reasonable model, is known. It can perform badly if a bad model is used, as is to be expected, but it can detect the better of two or more population models on the basis of compression.

M-alignment can be used practically to determine relatedness in small or moderate collections of sequences. It could also be used on a client-computer to post-process, and re-rank, putative matches returned from a large collection of sequences by a fast search algorithm running on a server. This use could reduce false-positives but could not bring back any false negatives. The superior performance of M-alignment was illustrated for this task by using a sequence from *Plasmodium falciparum* chromosome 2.

## References


1. L. Allison. Normalization of affine gap costs used in optimal sequence alignment. *Journal of Theoretical Biology*, 161:263–269, 1993.
2. L. Allison, D. R. Powell, and T. I. Dix. Compression and approximate matching. *The Computer Journal*, 42(1):1–10, 1999.
3. L. Allison, D. R. Powell, and T. I. Dix. Modelling is more versatile than shuffling. Technical report, Monash University, School of Computer Science and Software Engineering., 2000.
4. L. Allison, C. S. Wallace, and C. N. Yee. Finite-state models in the alignment of macromolecules. *Journal of Molecular Evolution*, 35:77–89, 1992.

5. S. F. Altschul and B. W. Erickson. Significance of nucleotide sequence alignments: A method for random sequence permutation that preserves dinucleotide and codon usage. *Mol. Biol. Evol.*, 2(6):526–538, 1985.
6. M. J. Bishop and E. A. Thompson. Maximum likelihood alignment of DNA sequences. *J. Mol. Biol.*, 190:159–165, 1986.
7. S. E. Brenner, C. Chothia, and T. J. P. Hubbard. Assessing sequence comparison methods with reliable structurally identifed distant evolutionary relationships. *Proc. Natl. Acad. Sci.*, 95:6073–6078, May 1998.
8. J.-M. Claverie and D. J. States. Information enhancement methods for large scale sequence analysis. *Comp. Chem*, 17(2):191–201, 1993.
9. M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5:345–352, 1978.
10. S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14:755–763, 1998.
11. W. M. Fitch. Random sequences. *Journal of Molecular Biology*, 163:171–176, 1983.
12. M. P. Georgeff and C. S. Wallace. A general selection criterion for inductive inference. In *European Conf. on Artificial Intelligence*, pages 473–482, 1984.
13. O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705–708, 1982.
14. M. Gribskov and N. L. Robinson. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers and Chemistry*, 20(1):25–33, 1996.
15. S. Grumbach and F. Tahi. A new challenge for compression algorithms: genetic sequences. *Inf. Proc. and Management*, 30(6):875–886, 1994.
16. S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Academy Science*, 89(10):915–919, 1992.
17. R. Huestis and K. Fischer. Prediction of many new exons and introns in *Plasmodium falciparum* chromosome 2. *Molecular and Biochemical Parasitology*, 118:187–199, 2001.
18. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady.*, 10(8):707–710, 1966.
19. D. M. Loewenstern and P. N. Yianilos. Significantly lower entropy estimates for natural DNA sequences. Technical Report 96-51, DIMACS, December 1996.
20. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, june 1953.
21. S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.
22. W. R. Pearson. Effective protein sequence comparison. *Meth. Enzymol.*, 266:227–258, 1996.
23. W. R. Pearson and D. J. Lipman. Improved tools for biological comparison. *Proc. Natl. Acad. Sci. USA*, 85:2444–2448, April 1988.
24. E. Rivals, O. Delgrange, J.-P. Delahaye, M. Dauchet, M.-O. Delorme, A. Hénaut, and E. Ollivier. Detection of significant patterns by compression algorithms: the case of approximate tandem repeats in DNA sequences. *CABIOS*, 13(2):131–136, 1997.
25. P. H. Sellers. On the theory and computation of evolutionary distances. *SIAM J. Appl. Math.*, 26(4):787–793, 1974.
26. C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication.* U. of Illinois Press, 1949.
27. T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.
28. C. S. Wallace and P. R. Freeman. Estimation and inference by compact coding. *Journal of the Royal Statistical Society series B.*, 49(3):240–265, 1987.