

SEQUENCE ALIGNMENT BY RARE EVENT SIMULATION

Jonathan Keith
Dirk P. Kroese

Department of Mathematics
University of Queensland
Brisbane 4072, AUSTRALIA

ABSTRACT

We present a new stochastic method for finding the optimal alignment of DNA sequences. The method works by generating random paths through a graph (the edit graph) according to a Markov chain. Each path is assigned a score, and these scores are used to modify the transition probabilities of the Markov chain. This procedure converges to a fixed path through the graph, corresponding to the optimal (or near-optimal) sequence alignment. The rules with which to update the transition probabilities are based on Rubinstein's *Cross-Entropy Method*, a new technique for stochastic optimization. This leads to very simple and natural updating formulas. Due to its versatility, mathematical tractability and simplicity, the method has great potential for a large class of combinatorial optimization problems, in particular in biological sciences.

1 INTRODUCTION

Sequence alignment is a frequently encountered theme in computational biology. Many biologically important molecules are linear arrangements of subunits and can therefore be characterised as sequences. For example, a protein consists of amino acid residues linked by peptide bonds in a specific order known as its *primary structure*. A protein can alternatively be characterised as a sequence of larger subunits called *secondary structures*. Yet another characterisation of a protein is its *tertiary structure*: the sequence of spatial positions and orientations taken by each of its amino acid residues. In order to study the structural, functional and evolutionary relationships amongst biologically similar molecules, it is often useful to first align the corresponding sequences. Sequence alignment is also an aspect of searching biological databases to detect homologies and is a key step in shotgun sequence assembly.

There are various forms of sequence alignment. Alignments can be made between sequences of the same type (for example, between the primary structures of proteins) or

between sequences of different type (for example, alignment of a DNA sequence to a protein sequence, or of a protein to a three-dimensional structure). *Pairwise alignment* involves only two sequences, whereas *multiple sequence alignment* involves more than two sequences (although the term sometimes encompasses pairwise alignment also). *Global* alignment aligns whole sequences, whereas *local* alignment aligns only parts of sequences.

Algorithms for sequence alignment have been extensively studied. The inaugural paper on the subject is that of Needleman and Wunsch (1970) and a useful reference is Gusfield (1997). The many algorithms used for sequence alignment are here classified as *deterministic*, *stochastic* or *heuristic*. *Deterministic* algorithms formulate sequence alignment as an optimisation problem and search deterministically for a globally optimal alignment. Two examples are the dynamic programming approach initiated by Needleman and Wunsch (1970), and the polyhedral approach initiated by Kececioğlu, Lenhof, Mehlhorn, Mutzel, Reinert, and Vingron (2000). *Stochastic* algorithms also formulate sequence alignment as an optimisation problem, but use stochastic optimisation techniques to search for a global optimum. Stochastic algorithms are often faster than deterministic ones, but have the disadvantage that they may return a sub-optimal alignment. Two examples are the HMM approach (Krogh, Brown, Mian, Sjolander, and Haussler 1994) and the Gibbs sampler approach (Lawrence, Altschul, Boguski, Liu, Neuwald, and Wootton 1993). *Heuristic* algorithms differ from stochastic algorithms in that they use a problem-specific search method, rather than standard stochastic optimisation techniques. Heuristic algorithms also may return a sub-optimal alignment. A list of examples is given by Pevzner (1992) (this paper also mentions several key references on the dynamic programming approach). The distinction between stochastic and heuristic algorithms is admittedly somewhat arbitrary.

The method presented here is a stochastic algorithm for pairwise global alignment. It uses an exciting new technique for stochastic optimisation known as the cross-entropy

method (Rubinstein 1999). This is the first application of the cross-entropy method to a problem in computational biology, but we anticipate that this versatile technique will find many other uses in this field.

The paper is structured as follows. The remainder of this introduction provides definitions of some key terms and notation. Section 2 describes global pairwise alignment in more detail, and in particular describes how alignments may be characterised as paths in a graph. Section 3 discusses cross-entropy and combinatorial optimisation via rare events simulation. Section 4 describes our main algorithm for sequence alignment by rare event simulation. In Section 5 we present some examples and in Section 6 we discuss the merits of the approach and potential directions for further research.

1.1 Definitions and Notation

The definitions and notation introduced here closely follow Gusfield (1997).

A *character* is an element of a set Σ called the *alphabet*. A *token* is a character or a space. The set of tokens is denoted by Σ' . A *string* S is an ordered list of characters written contiguously from left to right. (We use the terms *string* and *sequence* interchangeably.) For any string S , $S[i..j]$ is the (contiguous) *substring* of S that starts at position i and ends at position j of S . In particular, $S[1..j]$ is the *prefix* of S that ends at position j . For any string S , $S(i)$ denotes the i th character of S .

A (global) *alignment* of two strings S_1 and S_2 is obtained by first inserting spaces, either into or at the ends of S_1 and S_2 , and then placing the two resulting strings one above the other so that every character or space in either string is opposite a unique character or a unique space in the other string.

When comparing two characters, we say that the characters *match* if they are *identical*; otherwise we say they *mismatch*. An *edit operation* on a string s is one of three operations: a *substitution* of one character for another, an *insertion* of a character into or at the end of the string, or a *deletion* of a character. The *edit distance* between two strings S_1 and S_2 is the minimum number of edit operations needed to transform the first string into the second.

2 SEQUENCE ALIGNMENT

To be useful in applications, an alignment of two sequences should reflect in some way the commonalities of the sequences. Some alignments are therefore better than others. This concept is formalised using a *scoring function* to assign a value to an alignment. Let S_1 and S_2 be two sequences of length n_1 and n_2 , respectively, and let T_1 and T_2 be sequences of tokens obtained by inserting spaces into or at the ends of S_1 and S_2 such that T_1 and T_2 are of equal

length l . Let $\mathbf{x} = (T_1, T_2)$ represent an alignment of S_1 and S_2 . Conceptionally we can view (T_1, T_2) as a matrix with 2 rows and l columns where the first row contains the tokens of T_1 in order and the second row contains the tokens of T_2 in order. Let $V(\mathbf{x})$ be a scoring function on the space of all possible alignments. An optimal global sequence alignment is then an alignment \mathbf{x} which solves

$$\min_{\mathbf{x}} V(\mathbf{x}) \quad (1)$$

(or $\max_{\mathbf{x}} V(\mathbf{x})$, depending on the nature of the scoring function.)

Many of the scoring functions encountered in practice are of the following form. Each column i of the alignment is assigned a score $v(T_1(i), T_2(i))$, where $T_1(i)$ and $T_2(i)$ are the i th tokens of T_1 and T_2 respectively and v is the so-called *scoring matrix* defined over pairs of tokens (elements of Σ'). The score of the alignment is the sum of the column scores:

$$V(\mathbf{x}) = \sum_i v(T_1(i), T_2(i)). \quad (2)$$

An example of a scoring matrix is $v(x, y) = 0$ if $x = y$, otherwise $v(x, y) = 1$, where $x, y \in \Sigma'$. In this case, the minimum score is equal to the edit distance between S_1 and S_2 .

The classic approach to computing optimal alignments is via dynamic programming (Needleman and Wunsch 1970, Smith and Waterman 1981). Using this approach, the edit distance can be computed in $O(n_1 n_2)$ time (Gusfield 1997). The algorithm is outlined below.

Let $D(i, j)$ be the edit distance between the prefixes $S_1[1..i]$ and $S_2[1..j]$. Note that we allow null prefixes, in which case either i or j is zero (or both). The edit distance between S_1 and S_2 can then be calculated recursively using the following relation:

$$D(i, j) = \min[D(i-1, j) + 1, D(i, j-1) + 1, D(i-1, j-1) + t(i, j)],$$

where $t(i, j)$ is defined to have value 1 if $S_1(i) \neq S_2(j)$, and $t(i, j)$ has value 0 if $S_1(i) = S_2(j)$. The initial conditions on the recurrence are

$$D(i, 0) = i \quad \text{and} \quad D(0, j) = j.$$

The set of alignments realising the minimum score can be obtained at the same time by recursively computing the set of optimal alignments for each pair of prefixes. With some modifications, the algorithm can be implemented in $O(n_1 n_2)$ complexity (time) and $O(n_1)$ space, where $n_1 \leq n_2$ (Hirschberg 1977).

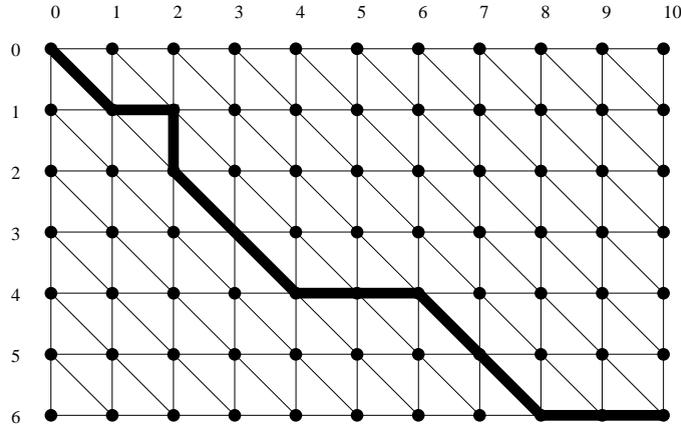


Figure 1: Each Alignment Corresponds to a Path

An alignment can alternatively be characterised as a path through a graph. Given two strings S_1 and S_2 of length n_1 and n_2 respectively, the *edit graph* for the strings is the array of $(n_1 + 1)(n_2 + 1)$ nodes, each labelled with a distinct pair (i, j) , $0 \leq i \leq n_1, 0 \leq j \leq n_2$ and a set of *horizontal*, *vertical* and *diagonal* directed edges joining node pairs of the form $((i, j), (i, j + 1))$, $((i, j), (i + 1, j))$ and $((i, j), (i + 1, j + 1))$ respectively. An edit graph is illustrated in Figure 1. Define an *alignment path* through the edit graph to be a path from node $(0, 0)$ to node (n_1, n_2) , that is, a sequence of edge-joined nodes $(0, 0), \dots, (n_1, n_2)$. Let \mathcal{X} be the space of all alignment paths. There is a one-to-one correspondence between alignments of S_1 and S_2 and alignment paths through the edit graph. This correspondence may be seen by defining the following isomorphism, mapping an alignment path to an alignment. First number the edges of the alignment path in order $k = 1, \dots, l$. Then the k th column of the alignment is determined from the k th edge of the alignment path in the following manner: for a horizontal edge $((i, j), (i, j + 1))$, let the column be $(-, S(j+1))'$ (“-” means space and “'” indicates transposition); for a vertical edge $((i, j), (i + 1, j))$ let the column be $(S(i+1), -)'$; and for a diagonal edge $((i, j), (i + 1, j + 1))$ let the column be $(S(i + 1), S(j + 1))'$.

Since there is a one to one correspondence between alignments and alignment paths it should not cause confusion if the same symbol \mathbf{x} is used to represent both objects, and the same symbol \mathcal{X} is used to represent the corresponding spaces. Moreover, any scoring function for alignments may be regarded as a scoring function for alignment paths, thus conferring a *length* to each path. The optimal alignment therefore corresponds to the shortest (or longest) alignment path. For scoring functions of the form (2) one can associate weights to each edge in the edit graph: for a horizontal edge $((i, j), (i, j + 1))$ the corresponding weight is $v(-, s(j + 1))$; for a vertical edge $((i, j), (i + 1, j))$ the weight is $v(S(i + 1), -)$; and for a diagonal edge $((i, j), (i + 1, j + 1))$ the

weight is $v(S(i + 1), S(j + 1))$. The score of a given path is then the sum of the weights on its edges.

3 COMBINATORIAL OPTIMISATION VIA RARE EVENT SIMULATION

Consider the following minimisation problem. Let \mathcal{X} be a finite set of *states*, and let V be a real function on \mathcal{X} . We wish to find a state $\mathbf{x}^* \in \operatorname{argmin}_{\mathbf{x}} V(\mathbf{x})$. In other words, we wish to find \mathbf{x}^* such that

$$V(\mathbf{x}^*) \leq V(\mathbf{x}), \quad \text{for all } \mathbf{x} \in \mathcal{X}. \quad (3)$$

When the number of states in \mathcal{X} is large, simulation becomes a viable approach to the above optimisation problem. A possible simulation procedure is described next.

Let f be some probability mass function (pmf) on \mathcal{X} such that $f(\mathbf{x}) > 0$ for all \mathbf{x} . For each $\mathbf{x} \in \mathcal{X}$ and *threshold* $\gamma \in \mathbb{R}$ define

$$H(\mathbf{x}; \gamma) = \begin{cases} 1 & \text{if } V(\mathbf{x}) \leq \gamma, \\ 0 & \text{if } V(\mathbf{x}) > \gamma. \end{cases}$$

Suppose we wish to estimate

$$\ell_f(\gamma) := \sum_{\mathbf{x}} H(\mathbf{x}; \gamma) f(\mathbf{x}) = \mathbb{E}_f H(\mathbf{X}; \gamma), \quad (4)$$

where \mathbf{X} is a random vector and \mathbb{E}_f denotes the expectation operator under pmf f . Equation (4) indicates how we may estimate $\ell_f(\gamma)$ by simulation: If $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(N)}$ is a random sample from the pmf f then

$$\frac{1}{N} \sum_{k=1}^N H(\mathbf{X}^{(k)}; \gamma) \quad (5)$$

is an unbiased estimator of $\ell_f(\gamma)$.

Now let g be another pmf on \mathcal{X} . Observe that

$$\ell_f(\gamma) = \sum_{\mathbf{x}} H(\mathbf{x}; \gamma) \frac{f(\mathbf{x})}{g(\mathbf{x})} g(\mathbf{x}) = \mathbb{E}_g H(\mathbf{X}; \gamma) \frac{f(\mathbf{X})}{g(\mathbf{X})}. \quad (6)$$

Hence, an alternative way to estimate $\ell_f(\gamma)$ is by taking a random sample $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(N)}$ from g and evaluating the (unbiased) estimator

$$\frac{1}{N} \sum_{k=1}^N H(\mathbf{X}^{(k)}) \frac{f(\mathbf{X}^{(k)})}{g(\mathbf{X}^{(k)})}. \quad (7)$$

In simulation jargon, we have used *Importance Sampling* with a *change of measure* g . The optimal choice for g is determined by the variance of the random variable $H(\mathbf{X}) \frac{f(\mathbf{X})}{g(\mathbf{X})}$ under pmf g . The smaller the variance, the more accurate our estimate will be. It is not difficult to see that the change of measure \tilde{g} that yields the smallest variance is given by

$$\tilde{g}(\mathbf{x}) := \frac{H(\mathbf{x}; \gamma) f(\mathbf{x})}{\ell_f(\gamma)}. \quad (8)$$

Under this change of measure the random variable $H(\mathbf{X}) \frac{f(\mathbf{X})}{g(\mathbf{X})}$ is constant and equal to $\ell_f(\gamma)$.

We can now see the connection with our optimisation problem (3). Namely, if we choose $\gamma = \gamma^*$, where γ^* is the minimum of V , then the best way to simulate $\ell_f(\gamma^*)$ is to generate a random sample from pmf \tilde{g} , which in this case has only positive mass on $\text{argmin}_{\mathbf{x}} V(\mathbf{x})$. Hence, if we know \tilde{g} then we can solve (3). Moreover, if γ is close to γ^* then generating samples from the corresponding \tilde{g} will yield vectors \mathbf{X} for which $V(\mathbf{X})$ is close to the optimal value (namely $V(\mathbf{X}) \leq \gamma$).

The obvious problem with (8) is, of course, that we do not know $\ell_f(\gamma)$. We will shortly discuss a way around this problem.

3.1 Parametric Families

The pmfs discussed above often belong to the same family of distributions, e.g., $\{f(\cdot; \mathbf{p})\}$ where the parameter vector \mathbf{p} takes values in some subset of \mathbb{R}^k , for some fixed k . For such pmfs let us rewrite (4) as

$$\ell_p(\gamma) = \sum_{\mathbf{x}} H(\mathbf{x}; \gamma) f(\mathbf{x}; \mathbf{p}) = \mathbb{E}_{\mathbf{p}} H(\mathbf{X}; \gamma), \quad (9)$$

where we have used a simplified notation in which the subscript \mathbf{p} replaces $f(\cdot; \mathbf{p})$. Henceforth we will use this simplified notation when convenient, without further discussion.

The best way to estimate $\ell_p(\gamma)$ via (7) is to use the change of measure \tilde{g} defined as in (8). However, this pmf may not lie in the parametric family $f(\cdot; \mathbf{p})$. But we can still try to choose an ‘‘optimal’’ pmf $f(\cdot; \tilde{\mathbf{p}})$ in the sense that the *distance* between this pmf and \tilde{g} is minimal.

3.2 Cross Entropy

There are several ways to measure the distance between two distributions (or pmfs). A particular convenient ‘‘distance’’ is the *Cross Entropy* or the *Kullback-Leibler distance*. If u and v are two pmfs, the Cross Entropy is defined as

$$\mathcal{C}(u, v) = \mathbb{E}_u \log \frac{u(\mathbf{X})}{v(\mathbf{X})}.$$

For estimating (9) we choose the optimal parameter $\tilde{\mathbf{p}}$ such that the Cross Entropy between $H(\cdot; \gamma) f(\cdot; \mathbf{p}) / \ell_p(\gamma)$ and $f(\cdot; \tilde{\mathbf{p}})$ is minimal. Writing out this cross entropy, it is easy to see that $\tilde{\mathbf{p}}$ should be such that

$$\phi(\tilde{\mathbf{p}}; \mathbf{p}, \gamma) := \mathbb{E}_{\mathbf{p}} H(\mathbf{X}; \gamma) \log f(\mathbf{X}; \tilde{\mathbf{p}}) \quad (10)$$

is *maximal*. The power of the Cross Entropy approach is that the optimal parameter $\tilde{\mathbf{p}}$ can often be calculated *analytically*. We will see an example of this in the next section, where each component of $\tilde{\mathbf{p}}$ is found to be of the form

$$\frac{\mathbb{E}_{\mathbf{p}} H(\mathbf{X}; \gamma) I_{\{X \in A\}}}{\mathbb{E}_{\mathbf{p}} H(\mathbf{X}; \gamma) I_{\{X \in B\}}},$$

where $I_{\{X \in A\}}$ and $I_{\{X \in B\}}$ respectively denote the indicators of the events $\{X \in A\}$ and $\{X \in B\}$ for some $A \subset B \subset \mathcal{X}$. This number typically needs to be estimated. For this we can use the estimator

$$\frac{\sum_{k=1}^N H(\mathbf{X}^{(k)}) I_{\{\mathbf{X}^{(k)} \in A\}}}{\sum_{k=1}^N H(\mathbf{X}^{(k)}) I_{\{\mathbf{X}^{(k)} \in B\}}}, \quad (11)$$

where $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(N)}$ is a random sample from the pmf $f(\cdot; \mathbf{p})$.

It is important to note that the estimator above is only of practical use when N , \mathbf{p} and γ are such that the total number of samples for which the score is less than or equal to γ is not too small. For example when γ is close to γ^* and \mathbf{p} assigns almost no probability mass to vectors \mathbf{x} for which $V(\mathbf{x}) \leq \gamma$, most random samples would provide an estimator 0/0, unless N is exceedingly large. This poses a problem to the proposed minimisation procedure. On the one hand we would like to choose γ as close as possible to γ^* , and find (an estimate) of $\tilde{\mathbf{p}}$ via the procedure above, which assigns almost all mass to vectors close to the optimal vector(s). On the other hand, we would like to

keep γ relative large in order to obtain a viable estimator for $\tilde{\mathbf{p}}$.

3.3 Adaptive Estimation

To overcome this problem, one may consider a *sequence* of thresholds $\gamma_0, \gamma_1, \dots$ and a sequence of parameter vectors $\mathbf{p}_0, \mathbf{p}_1, \dots$, such that $\{\gamma_n\}$ converges to a value close to the optimal γ^* and $\{\mathbf{p}_n\}$ converges to a pmf that assigns high probability mass to vectors that give a small score. This strategy is embodied in the following procedure, see e.g., Rubinstein (1999):

Start with some \mathbf{p}_0 . Let $n = 0$.

Repeat the following until convergence is reached:

- Draw a random sample $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}$ from $f(\cdot, \mathbf{p}_n)$, where N is some fixed number.
- Calculate the scores for each of these vectors, and order them from smallest to biggest, $s_1 \leq \dots \leq s_N$. Let ξ be the integer part of ρN . Define $\gamma_n = s_\xi$.
- Define \mathbf{p}_{n+1} as the estimate of the optimal $\tilde{\mathbf{p}}$ in (10) with $\mathbf{p} = \mathbf{p}_n$. Thus, the components of \mathbf{p}_{n+1} are found from (11). Increase n by 1.

In each iteration, the new threshold value is selected to be greater than or equal to a proportion ρ of the current sample scores, thus ensuring (11) is a viable estimator for $\tilde{\mathbf{p}}$. Note that the stopping criterion, the initial vector \mathbf{p}_0 , the sample size N and the number ρ have to be specified in advance, but that for the rest the algorithm is “self-tuning”.

4 RARE EVENT SIMULATION IN SEQUENCE ALIGNMENT

In this section we derive our main algorithm by combining the main ideas of Sections 2 and 3.

In Section 2 the optimal sequence alignment problem was formulated in terms of a shortest path problem through a graph. Specifically, in that context, we need to find an alignment path \mathbf{x} through the edit graph for which the alignment score $V(\mathbf{x})$ is minimal. It is clear that this problem fits the combinatorial optimisation formulation of Section 3. In particular, the state space \mathcal{X} is given by the collection of all possible alignment paths.

On this space we define a class of probability mass functions $\{f(\cdot, \mathbf{p})\}$ in the following way: Let $M = \{M_0, M_1, \dots\}$ be a Markov chain on the edit graph, starting at the top left-hand corner $(0, 0)$ with the one-step transition probabilities shown in Table 1 (for all $0 \leq i \leq n_1 - 1$ and $0 \leq j \leq n_2 - 1$).

Note that here r stands for *right* and d for *down*. Moreover, for $j = 0, \dots, n_2 - 1$ the transition probability from (n_1, j) to $(n_1, j+1)$ is 1. Similarly, for $i = 0, \dots, n_1 -$

Table 1: The Transition Probabilities

from	to	with prob.
(i, j)	$(i + 1, j)$	$r(i, j)$
(i, j)	$(i, j + 1)$	$d(i, j)$
(i, j)	$(i + 1, j + 1)$	$1 - r(i, j) - d(i, j)$

1 the transition probability from (i, n_2) to $(i + 1, n_2)$ is 1. Finally, (n_1, n_2) is an absorbing state. Let τ be the time by which M has reached the absorbing state (note that $\tau \leq n_1 + n_2$), and let M^τ denote the path taken by M through the edit graph. We gather all parameters $\{r(i, j), d(i, j), 0 \leq i \leq n_1 - 1, 0 \leq j \leq n_2 - 1\}$ into a single parameter vector \mathbf{p} . For each such \mathbf{p} let

$$f(\mathbf{x}; \mathbf{p}) = \mathbb{P}_{\mathbf{p}}(M^\tau = \mathbf{x}), \quad \text{for all } \mathbf{x} \in \mathcal{X}.$$

This defines a proper probability distribution on \mathcal{X} with pmf $f(\cdot; \mathbf{p})$. Denote by $\mathcal{X}(i, j)$ the collection of all paths going through node (i, j) , and by $\bar{\mathcal{X}}(i, j)$ the collection of all paths *not* going through node (i, j) .

Now, consider the estimation of (9) via Importance Sampling. The optimal change of measure within the same parametric family is given by $f(\cdot, \tilde{\mathbf{p}})$, where $\tilde{\mathbf{p}}$ is such that (10) is maximal. To maximise (10) let us first look at $f(\mathbf{x}; \mathbf{p})$ for some fixed \mathbf{x} and \mathbf{p} . Let $\mathcal{X}_r(i, j)$ be the set of all paths \mathbf{x} making the transition from (i, j) to $(i + 1, j)$. Similarly define $\mathcal{X}_d(i, j)$ as the set of all paths \mathbf{x} making the transition from (i, j) to $(i, j + 1)$. Defining 1_A as the indicator function of a set A , we can write

$$\begin{aligned} f(\mathbf{x}; \mathbf{p}) &= \prod_{i=0}^{n_1-1} \prod_{j=0}^{n_2-1} \left(r(i, j) 1_{\mathcal{X}_r(i, j)}(\mathbf{x}) \right. \\ &\quad + d(i, j) 1_{\mathcal{X}_d(i, j)}(\mathbf{x}) \\ &\quad + (1 - r(i, j) - d(i, j)) 1_{\mathcal{X}'(i, j)}(\mathbf{x}) \\ &\quad \left. + 1_{\bar{\mathcal{X}}(i, j)}(\mathbf{x}) \right), \end{aligned} \quad (12)$$

where we have abbreviated the set $\mathcal{X}(i, j) - \mathcal{X}_r(i, j) - \mathcal{X}_d(i, j)$ to $\mathcal{X}'(i, j)$. It follows that

$$\begin{aligned} \phi(\tilde{\mathbf{p}}; \mathbf{p}, \gamma) &= \\ &\sum_{i=0}^{n_1-1} \sum_{j=0}^{n_2-1} \mathbb{E}_{\mathbf{p}} H(\mathbf{X}; \gamma) \left(\log(\tilde{r}(i, j)) 1_{\mathcal{X}_r(i, j)}(\mathbf{X}) \right. \\ &\quad + \log(\tilde{d}(i, j)) 1_{\mathcal{X}_d(i, j)}(\mathbf{X}) \\ &\quad \left. + \log(1 - \tilde{r}(i, j) - \tilde{d}(i, j)) 1_{\mathcal{X}'(i, j)}(\mathbf{X}) \right). \end{aligned}$$

Hence, maximising (10) with respect $\tilde{r}(i, j)$ and $\tilde{d}(i, j)$ for all i and j amounts to differentiating the expression above with respect to $\tilde{r}(i, j)$ and $\tilde{d}(i, j)$ and equating it to zero. This gives the set of equations

$$\frac{1}{\tilde{r}(i, j)} \mathbb{E}_{\mathbf{p}} H(\mathbf{X}; \gamma) I_{\{X \in \mathcal{X}_r(i, j)\}} - \frac{1}{1 - \tilde{r}(i, j) - \tilde{d}(i, j)} \mathbb{E}_{\mathbf{p}} H(\mathbf{X}; \gamma) I_{\{X \in \mathcal{X}'(i, j)\}} = 0,$$

and

$$\frac{1}{\tilde{d}(i, j)} \mathbb{E}_{\mathbf{p}} H(\mathbf{X}; \gamma) I_{\{X \in \mathcal{X}_d(i, j)\}} - \frac{1}{1 - \tilde{r}(i, j) - \tilde{d}(i, j)} \mathbb{E}_{\mathbf{p}} H(\mathbf{X}; \gamma) I_{\{X \in \mathcal{X}'(i, j)\}} = 0,$$

from which it follows that

$$\tilde{r}(i, j) = \frac{\mathbb{E}_{\mathbf{p}} H(\mathbf{X}; \gamma) I_{\{X \in \mathcal{X}_r(i, j)\}}}{\mathbb{E}_{\mathbf{p}} H(\mathbf{X}; \gamma) I_{\{X \in \mathcal{X}(i, j)\}}}$$

and

$$\tilde{d}(i, j) = \frac{\mathbb{E}_{\mathbf{p}} H(\mathbf{X}; \gamma) I_{\{X \in \mathcal{X}_d(i, j)\}}}{\mathbb{E}_{\mathbf{p}} H(\mathbf{X}; \gamma) I_{\{X \in \mathcal{X}(i, j)\}}}.$$

As in (11), we can estimate $\tilde{r}(i, j)$ and $\tilde{d}(i, j)$ in the following way. We run N independent copies of the Markov process M using the one-step transition probabilities in \mathbf{p} . This leads to the random sample of paths $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(N)}$ from the pmf $f(\cdot; \mathbf{p})$. The estimators of $\tilde{r}(i, j)$ and $\tilde{d}(i, j)$ are respectively given by

$$\frac{\sum_{k=1}^N H(\mathbf{X}^{(k)}; \gamma) I_{\{X^{(k)} \in \mathcal{X}_r(i, j)\}}}{\sum_{k=1}^N H(\mathbf{X}^{(k)}; \gamma) I_{\{X^{(k)} \in \mathcal{X}(i, j)\}}}, \quad (13)$$

and

$$\frac{\sum_{k=1}^N H(\mathbf{X}^{(k)}; \gamma) I_{\{X^{(k)} \in \mathcal{X}_d(i, j)\}}}{\sum_{k=1}^N H(\mathbf{X}^{(k)}; \gamma) I_{\{X^{(k)} \in \mathcal{X}(i, j)\}}}. \quad (14)$$

These estimators have an easy interpretation. For example, to obtain $\tilde{r}(i, j)$ we count the number of paths (out of N) going from (i, j) to $(i, j+1)$ that have a score less than or equal to γ , and divide this number by the total number of paths passing through (i, j) that have a score less than or equal to γ . The estimator for $\tilde{d}(i, j)$ has a similar natural interpretation.

Using the algorithm outlined in Section 3 we now construct a sequence $\gamma_0, \gamma_1, \dots$ decreasing to γ and a sequence $\mathbf{p}_0, \mathbf{p}_1, \dots$ tending to some vector $\bar{\mathbf{p}}$ such that γ is close to γ^* and such that $f(\cdot; \bar{\mathbf{p}})$ assigns positive mass only to alignment paths \mathbf{x} for which $V(\mathbf{x}) \leq \gamma$.

4.1 Main Algorithm

1. Initialize as follows: $j := 0$ (iteration counter); Choose an initial vector of transition probabilities \mathbf{p}_0 , for example with $r(i, j) = d(i, j) = 1/3$.
2. Generate N paths of the Markov process M , using the transition probabilities specified in \mathbf{p}_j .
3. Calculate the scores for each of these paths, and order them from smallest to biggest, $s_1 \leq \dots \leq s_N$. Let ξ be the integer part of ρN . Define $\gamma_j = s_\xi$.
4. Find the next parameter vector \mathbf{p}_{j+1} from (13) and (14), for each (i, j) .
5. Increment j and repeat steps 2–5, until convergence has been reached.

Note that the stopping criterion, the initial vector \mathbf{p}_0 , the sample size N and the number ρ have to be specified in advance.

4.2 A Modified Algorithm

Paths generated by the Markov process described above are centred around the path consisting of diagonal edges leading down and to the right from $(0, 0)$. Paths that deviate far from this centre path are rare, and consequently some parts of the edit graph are unlikely to be explored. This poses a problem when the optimal path deviates substantially from the centre path. For example, to optimally align AAAAABBBBB with BBBBBB, we must prefix the second string with 5 spaces. This means that the corresponding path through the edit graph, $(0, 0), \dots, (5, 0), (6, 1), \dots, (10, 5)$, initially travels along the upper border. The algorithm does not converge properly in this case because it is most unlikely that the Markov chain M will follow such a path along the border unless \mathbf{p} is such that $r(0, 0), \dots, r(4, 0)$ are very close to 1.

To remedy this problem, we allow M to start at any position along the *upper border* $B_u := \{(0, j), j = 0, 1, \dots, n_2\}$ or the *left border* $B_l := \{(i, 0), i = 0, 1, \dots, n_1\}$. Denote the probability that M starts at (i, j) by $\alpha(i, j)$. The theory above can be carried through with only slight modifications. For example, \mathcal{X} is now the set of paths through the edit graph, starting on the upper or left boundary; and the parameter vector \mathbf{p} now includes the initial probabilities $\alpha(i, j)$. Moreover, letting $\mathcal{Y}(i, j)$ be the set of paths that start at (i, j) , the right-hand side of (12) should be multiplied by

$$\sum_{(i, j) \in B} \alpha(i, j) 1_{\mathcal{Y}(i, j)}(\mathbf{x}),$$

where $B = B_l \cup B_u$ is the set of possible starting states. In addition to updating the $r(i, j)$ and $d(i, j)$, we now also have to update the $\alpha(i, j)$. This leads in addition to (13)

and (14) to the following updating formula for $\alpha(i, j)$:

$$\frac{\sum_{k=1}^N H(\mathbf{X}^{(k)}; \gamma) I_{\{\mathbf{X}^{(k)} \in \mathcal{Y}(i, j)\}}}{\sum_{k=1}^N H(\mathbf{X}^{(k)}; \gamma)}, \quad (15)$$

where $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(N)}$ is a random sample of alignment paths from the pmf $f(\cdot; \mathbf{p})$.

5 SOME EXAMPLES

In this section we give the results of a number of tests to which the algorithm has been subjected. In each case we search for an optimal alignment of two sequences S_1 of length n_1 and S_2 of length n_2 , with respect to the scoring matrix $v(x, y) = 0$ if $x = y$, otherwise $v(x, y) = 1$. For all cases we chose the parameters $\rho = 0.1$ and $N = 100(n_1 + n_2)$. Moreover, the initial transition probabilities for the Markov chain M were $1/3$ for each of the three directions, and the starting state of M was chosen uniformly on the union of the upper border and the left border.

The first test was to see if the algorithm could correctly find alignments of the strings of the form $S_1 = S_2 = \text{AAAAAAAAAA}$, for various lengths $n_1 (= n_2)$. The algorithm was found to converge to correct ‘‘self-alignment’’ for $n_1 = 10, 100$ and 1000 . The number of iterations required seem to vary as $\sqrt{n_1}$, see the table below.

Table 2: The Number of Iterations Required for the Self-Alignment Test, for Various String Lengths

n_1	10	100	1000
iterations	6	15	41

The second series of tests involved *right-shifts* of the form $S_1 = \text{AAAAAAAAAA}$ and $S_2 = \text{BBBBBBBBBBAAAAAAAAAA}$. In other words, S_2 is obtained from S_1 by prepending n_1 B’s. Note that the size of S_2 is $2n_1$. The test was performed for different values of n_1 (10, 100 and 1000), and the algorithm was found to converge to the optimal alignment. Similar results were found for a third series of tests, this time involving *left-shifts*: $S_1 = \text{AAAAAAAAAA}$ and $S_2 = \text{AAAAAAAAAAABBBBBBBBBB}$.

Finally, we applied the algorithm to two protein sequences from Escherichia coli: Nitrogen Regulatory Protein P-II 1 (database: gi121386) and Nitrogen Regulatory Protein P-II 2 (database: gi1707971), cf. Carr, Cheah, Suffolk, Vasudevan, Dixon, and Ollis (1996) and Xu, Cheah, Carr, van Heeswijk, Westerhoff, Vasudevan, and Ollis (1998).

The two protein sequences are shown below:

```
MKKIDAI IKPFKLLDDVREALAEVGITGMTVTEVKGFGRQ
KGHTELYRGA EYMVDFLPKVKIERTAQTKIGDGKIFVF
```

```
DVARVIRIRTGEEDDAAI
```

```
MKLVTVI IKPFKLEDEVREALSSIGIQGLTVTEVKGFGRQ
KGHAELYRGA EYSVNFLPKVKIDVAIADDQLDEVIDIVS
KAAVTGKIGDGKIFVAELQRVIRIRTGEADEAAL
```

The algorithm found the following alignment:

```
MKKIDAI IKPFKLLDDVREALAEVGITGMTVTEVKGFGRQ
MKLVTVI IKPFKLEDEVREALSSIGIQGLTVTEVKGFGRQ
```

```
KGHTELYRGA EYMVDFLPKVKI _____ E _____ R _____
KGHAELYRGA EYSVNFLPKVKIDVAIADDQLDEVIDIVS
```

```
TAQ _____ TGKIGDGKIFVFDVARVIRIRTGEEDDAAI
KAAVTGKIGDGKIFVAELQRVIRIRTGEADEAAL
```

which gives an optimal edit score of 39. The algorithm took a few seconds to execute. Note that the edit distance does not impose gap penalties. A more sophisticated scoring function would favour a single big gap to multiple small gaps.

6 CONCLUSIONS

In this paper we have described a new randomized algorithm for sequence alignment. The algorithm generates random alignments via a random walk. The transition probabilities of the random walk are altered/updated dynamically by minimizing the Cross-Entropy distance between two distributions. This leads to simple and effective updating rules.

In the examples given, the new algorithm does not outperform the standard dynamic programming approach. The examples are trivial, and merely demonstrate that the algorithm behaves sensibly. The significance of the new algorithm is that it may be possible to use it in conjunction with scoring functions that are not easily handled via existing approaches.

The cross entropy method has recently proved to be useful for solving difficult combinatorial problems in telecommunications and management science, see for example de Boer (2000), Lieber, Rubinstein, and Elmakis (1997), and Rubinstein (2000). In this paper, it is applied to a problem in computational biology for the first time. One of the merits of the Cross-Entropy (CE) approach is that it opens up a whole range of possible applications in computational biology. For example, it could be of help in solving multiple DNA sequence alignment and protein sequence alignment that involve complicated scoring functions for which no polynomial time algorithms exist. From a mathematical point of view the CE method is attractive because, unlike many other randomized algorithms, it can be sub-

jected to a rigorous analysis. This is important for any further developments and improvements of the algorithm.

An obvious direction for future research is to investigate which problems in computational biology could benefit from the CE approach. Work is underway to apply the CE method to protein folding, in which (due to the three dimensional structure of a protein) the score is a complicated function of the positions of the amino acids.

Another direction for research is to study in more detail how the algorithm could be modified and how this would affect its efficiency. For example, there are many ways to randomly generate the alignments. In the current algorithm the alignments are generated through a Markov process, but this is not essential for the CE approach. Other modifications could include “Ant Colony” heuristics, see for example Dorigo and Gambardella (1997) and Gutjahr (2000).

Finally, in this introductory paper we have made no attempt to compare the algorithm with other (randomized) algorithms for sequence alignment. It would be interesting to carry out such investigations in the future.

ACKNOWLEDGMENT

We would like to thank Dr. Thomas Huber for his assistance in the preparation of this paper.

REFERENCES

- Carr, P., E. Cheah, P. Suffolk, S. Vasudevan, N. Dixon, and D. Ollis. 1996. X-ray structure of the signal transduction protein from *Escherichia coli* at 1.9 Å. *Acta Crystallogr. D* 52:93 – 104.
- de Boer, P.-T. 2000. *Analysis and efficient simulation of queueing models of telecommunications systems*. Ph. D. thesis, University of Twente.
- Dorigo, M., and L. Gambardella. 1997. Ant colony systems: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation* 1 (1): 53 – 66.
- Gusfield, D. 1997. *Algorithms on strings, trees and sequences*. Cambridge: Cambridge University Press.
- Gutjahr, W. 2000. A graph-based ant system and its convergence. *Future Generations Computing* 16:873 – 888.
- Hirschberg, D. 1977. Algorithms for the longest common subsequence problem. *J. ACM* 24:664 – 675.
- Kececioğlu, J., H.-P. Lenhof, K. Mehlhorn, P. Mutzel, K. Reinert, and M. Vingron. 2000. A polyhedral approach to sequence alignment problems. *Disc. Appl. Math.* 104:143 – 186.
- Krogh, A., M. Brown, I. Mian, K. Sjolander, and D. Haussler. 1994. Hidden markov models in computational biology: applications to protein modeling. *J. Mol. Biol.* 235:1501 – 1531.
- Lawrence, C., S. Altschul, M. Boguski, J. Liu, A. Neuwald, and J. Wootton. 1993. Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science* 262:208 – 214.
- Lieber, D., R. Rubinstein, and D. Elmakis. 1997. Quick estimation of rare events in stochastic networks. *IEEE Trans. Rel.* 46 (2): 254 – 265.
- Needleman, S., and C. 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.
- Pevzner, P. 1992. Multiple alignment, communication cost and graph matching. *SIAM J. Appl. Math.* 52:1763 – 1779.
- Rubinstein, R. 1999. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability* 2:127 – 190.
- Rubinstein, R. 2000. Combinatorial optimization via cross-entropy. In *Encyclopedia of Management Sciences*, ed. S. Gass and C. Harris: Kluwer.
- Smith, T., and M. Waterman. 1981. Identification of common molecular subsequences. *J. Mol. Biol.* 147:195 – 197.
- Xu, Y., E. Cheah, P. Carr, W. van Heeswijk, H. Westerhoff, S. Vasudevan, and D. Ollis. 1998. GlnK, a pihomologue: structure reveals ATP binding site and indicates how the t-loops may be involved in molecular recognition. *J. Mol. Biol.* 282 (1): 149 – 165.

AUTHOR BIOGRAPHIES

JONATHAN KEITH gained his Ph.D. in Mining and Mineral Processing in 2000 at the University of Queensland, Australia. He is currently a Research Officer in the Mathematics Department at the University of Queensland. His interests include computational biology, phylogenetic inference, DNA sequencing, Markov Chain Monte Carlo simulation, the Cross-Entropy method and stereology. His email address is <j.keith1@mailbox.uq.edu.au>.

DIRK P. KROESE holds a Ph.D. degree in Mathematical Sciences from the University of Twente, The Netherlands. He has held faculty and research staff positions at the University of Twente, Princeton University, U.S.A. and the University of Melbourne, Australia. From 1998 to 2000 he was a Senior Research Fellow at the Teletraffic Research Centre in Adelaide, Australia. Currently he is a Lecturer in Statistics at the University of Queensland, Australia. His interests include randomised algorithms, efficient simulation, the Cross-Entropy method, queueing theory, performance analysis and computational biology. His web address is <<http://www.maths.uq.edu.au/~kroese/>>.