# Sequence Alignment By Rare Event Simulation

MASCOS Symposium
University of Queensland, 26 November 2004

**Jonathan M. Keith and Dirk P. Kroese**

Department of Mathematics
The University of Queensland

Email: jonathan@maths.uq.edu.au
http://www.maths.uq.edu.au/~kroese/

# Outline

1. Sequence Alignment

2. Sequence Alignment via Rubinstein's Cross-Entropy Method

3. The Algorithm

4. Directions for Future Research

5. (Optional) Consensus Sequences

# 1 Sequence Alignment

Consider two sequences of numbers $1, \ldots, n_1$ and $1, \ldots, n_2$. An **alignment** is an arrangement of the two sequences into two stacked rows, possibly including "spaces" (two opposite spaces not allowed).

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | - | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | - | 2 | 3 | 4 | - | - | 5 | 6 | - | - |

Table 1: An example of an alignment

The two sequences of numbers could be associated with the positions of characters in a DNA or protein sequence.

**Example**

1,2, . . . ,10 $\rightsquigarrow$ `AGTGCAGATA`

1,2, . . . ,6 $\rightsquigarrow$ `ACTGGA`

What is the "best" alignment?

## Edit distance

To assess which alignment is better, we need to **score** the alignments (assign numbers to them).

A simple way is to measure the **edit distance** between the aligned strings:

- Each **mismatch** increases the score by 1.
- Each insertion (space) increases the score by 1.

## Example

```
AGT-GCAGATA          AG-TGCAGATA
ACTG--G-A--          A-CTG--GA--
Score: 8             Score: 6
```

Best possible score:5, e.g.,

```
AGTGCAGATA          1 2 3 4 5 6 7 8 9 10
ACTG--GA--          1 2 3 4 - - 7 8 - -
```

## Alignment graph, alignment vector

Each alignment can be characterised as a path through a directed graph.

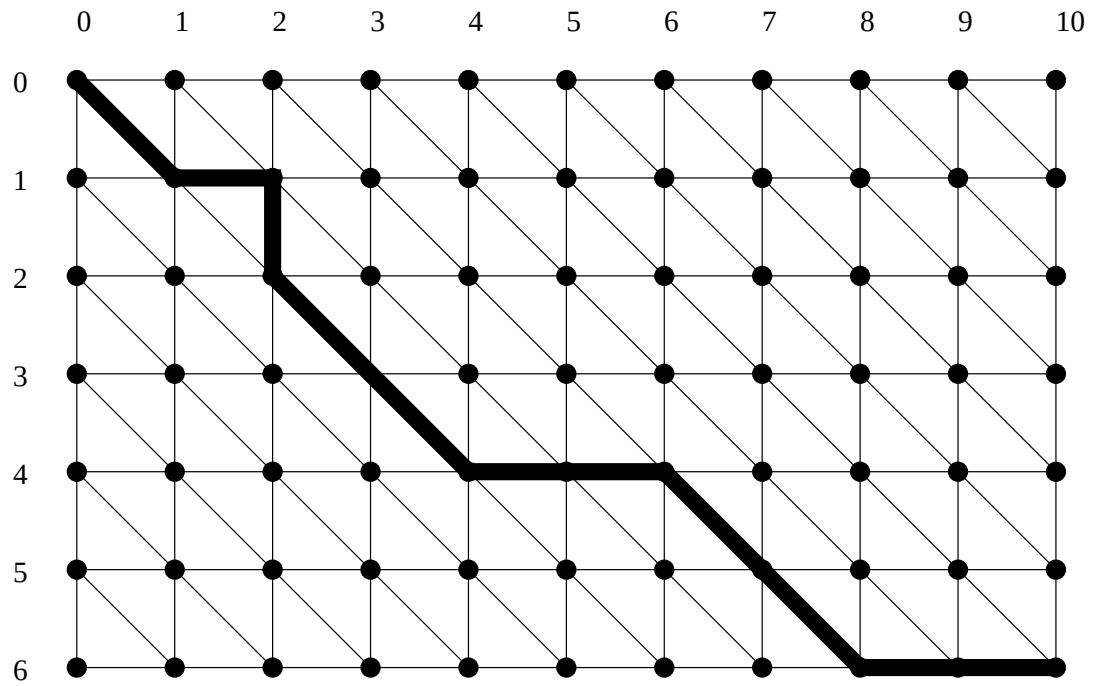


Figure 1: Alignment graph. Directed edges from $(i, j)$ to $(i + 1, j)$, $(i + 1, j + 1)$ and $(i, j + 1)$. Each path starting at $(0, 0)$ and ending at $(n_1, n_2)$ corresponds to an alignment.

Moreover, we can characterise this path by an **alignment vector** $\boldsymbol{x} = (x_1, \ldots, x_\ell)$ of 0s, 1s and 2s, where $x_i$ denotes the "direction" the path takes at the $i$th traversed node: 0 = horizontal, 1 = diagonal and 2 = vertical.

## Example

For example, the alignment in Table 1 corresponds to the path in Figure 1 and the alignment vector of length $\ell = 11$ in Table 2.

| $\boldsymbol{x}$ | 1 | 0 | 2 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | - | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| | 1 | - | 2 | 3 | 4 | - | - | 5 | 6 | - | - |

Table 2: Each alignment vector corresponds to an alignment

Let $S(\boldsymbol{x})$ denote the *score* of the alignment corresponding to alignment vector $\boldsymbol{x}$. Let $\mathcal{X}$ be the space of all possible alignment vectors.

We thus have translated the alignment problem into the Combinatorial Optimisation Problem:

$$\boxed{\min_{\boldsymbol{x}\in\mathcal{X}} S(\boldsymbol{x})\ .} \tag{1}$$

**Remark 1**

- When using the edit distance, the score depends in an "additive" way on the path. The optimal score can efficiently be determined by Dynamic Programming.

- In practice more complicated scoring functions are used. Still, DP works in many cases.

- When the scoring function depends on the *whole* path, then the COP is NP-complete. In particular, this holds for *structure* alignments, where the alignment corresponds to the spatial position of a protein residue (character).

- Deterministic algorithms only give one possible alignment and say nothing about the *distribution* of optimal alignments.

By using *randomised* optimisation technique, we can address the last two issues.

# 2 Sequence alignment via CE

We wish to "solve" (1) using a **Model-based Optimisation** approach:

1. Simulate a sample in $\mathcal{X}$ in accordance with a probabilistic model,
2. Update the model in light of the sample to produce a better scoring sample next time.

By iterating this procedure we expect to generate alignment vectors with a minimal or close to minimal score.

We need to specify

1. how we generate the samples ( i.e., alignment vectors),
2. how we update the model.

In our case:

1. Run a Markov chain on the alignment graph, starting at $(0, 0)$ and ending at $(n_1, n_2)$.
2. Adjust the one-step transition probabilities of the Markov chain via Cross-Entropy maximisation.

**Parameters:**
The chain has one-step transition probabilities:

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

(Note that here r stand for *right* and d for *down*.)

update the parameters of the Markov chain, i.e., the $r(i, j)$ and $d(i, j)$, at the end of each iteration.

## Cross-Entropy Method

Gather the parameters in a vector $\boldsymbol{v}$. Let $f(\cdot;\boldsymbol{v})$ denote the density (pmf) of the alignment vector $\boldsymbol{X}$ under $\boldsymbol{v}$. Let $H(\boldsymbol{X};\gamma) = I_{\{S(\boldsymbol{X})\leq\gamma\}}$.

Generate sequences $\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots$ and $\gamma_0, \gamma_1, \ldots$ as follows: Start with some $\boldsymbol{v}_0$. Let $n = 0$, and repeat the following until convergence is reached:

- Draw a random sample $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(N)}$ from $f(\cdot;\boldsymbol{v}_n)$.
- Calculate the scores for each of these vectors, and order them from smallest to biggest, $s_1 \leq \ldots \leq s_N$. Let $\xi$ be the integer part of $\rho N$. Define $\gamma_n = s_\xi$.
- Let $\boldsymbol{v}_{n+1}$ be that value of $\tilde{\boldsymbol{v}}$ which maximises

$$\sum_{k=1}^{N} H(\boldsymbol{X}^{(k)};\gamma_n) \log f(\boldsymbol{X}^{(k)};\tilde{\boldsymbol{v}}) \; . \qquad (2)$$

  Increase $n$ by 1.

Define:
$\mathcal{X}(i,j)$: the set of all $\boldsymbol{x}$ going through $(i,j)$.
$\mathcal{X}_r(i,j)$: the set of $\boldsymbol{x}$ going $(i,j) \to (i+1,j)$.
$\mathcal{X}_d(i,j)$ the set of $\boldsymbol{x}$ going $(i,j) \to (i,j+1)$.
$\mathcal{X}'(i,j) = \mathcal{X}(i,j) - \mathcal{X}_r(i,j) - \mathcal{X}_d(i,j)$.

Then, we can write

$$\begin{aligned} f(\boldsymbol{x};\boldsymbol{v}) &= \prod_{i=0}^{n_1-1}\prod_{j=0}^{n_2-1}\Big( r(i,j)\, I_{\mathcal{X}_r(i,j)}(\boldsymbol{x}) \\ &+ d(i,j)\, I_{\mathcal{X}_d(i,j)}(\boldsymbol{x}) \\ &+ (1-r(i,j)-d(i,j))\, I_{\mathcal{X}'(i,j)}(\boldsymbol{x})\Big)\,. \end{aligned}$$

It follows that the optimal updating parameters are found by optimising

$$\begin{aligned} &\sum_{k=1}^{N}\sum_{i=0}^{n_1-1}\sum_{j=0}^{n_2-1} H(\boldsymbol{X};\gamma)\Big( \log(\tilde{r}(i,j))\, I_{\mathcal{X}_r(i,j)}(\boldsymbol{X}) \\ &+\log(\tilde{d}(i,j))\, I_{\mathcal{X}_d(i,j)}(\boldsymbol{X}) \\ &+\log(1-\tilde{r}(i,j)-\tilde{d}(i,j))\, I_{\mathcal{X}'(i,j)}(\boldsymbol{X})\Big) \end{aligned}$$

with respect to $\tilde{r}(i,j)$ and $\tilde{d}(i,j)$ for all $i$ and $j$.

We can do this by differentiating the previous expression with respect to $\tilde{r}(i,j)$ and $\tilde{d}(i,j)$ and equating it to zero. This gives:

$$\tilde{r}(i,j) = \frac{\sum_{k=1}^{N} H(\boldsymbol{X}^{(k)};\gamma)\, I_{\{\boldsymbol{X}^{(k)} \in \mathcal{X}_r(i,j)\}}}{\sum_{k=1}^{N} H(\boldsymbol{X}^{(k)};\gamma)\, I_{\{\boldsymbol{X} \in \mathcal{X}(i,j)\}}} \;, \tag{3}$$

and

$$\tilde{d}(i,j)\frac{\sum_{k=1}^{N} H(\boldsymbol{X}^{(k)};\gamma)\, I_{\{\boldsymbol{X}^{(k)} \in \mathcal{X}_d(i,j)\}}}{\sum_{k=1}^{N} H(\boldsymbol{X}^{(k)};\gamma)\, I_{\{\boldsymbol{X} \in \mathcal{X}(i,j)\}}} \;. \tag{4}$$

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 $\gamma$, and
- divide this number by the total number of paths passing through $(i,j)$ that have a score less than or equal to $\gamma$.

# 3 Algorithm

**1. Initialise** all $r(i,j) = d(i,j) = 1/3$ on the interior of the graph.

**2. Generate** $N$ paths via the Markov process.

**3. Calculate the scores** for each of these paths. Let $\gamma$ be the smallest score of the best $\rho \times 100\%$ alignments.

**4. Update the parameters.**

Update $r(i,j)$ as

$$\frac{\#\text{ paths from } (i,j) \text{ to } (i+1,j) \text{ with a score} \leq \gamma}{\#\text{ paths passing through } (i,j)}\,.$$

Update $d(i,j)$ as

$$\frac{\#\text{ paths from } (i,j) \text{ to } (i,j+1) \text{ with a score} \leq \gamma}{\#\text{ paths passing through } (i,j)}\,.$$

**5. Repeat** steps 2–5, until convergence has been reached.

**Remark 2**

- Note that the initial parameter vector, the stopping criterion, the sample size $N$ and the proportion $\rho$ have to be specified in advance.

- We can introduce a mixing factor in the updating procedure ($\alpha\times$ the old parameter value $+$ $(1-\alpha)\times$ the new value.

- Instead of starting always at $(0,0)$ we can let the MC start anywhere on the left-upper border, with probabilities $p(i,j)$. These are updated via the same argument as

$$\frac{\#\text{ paths starting from } (i,j) \text{ with a score} \leq \gamma}{\#\text{ paths starting from } (i,j)} \ .$$

# 4 Research Directions

1. The alignments were generated via a specific Markov process, but there are **many different ways** to do this. Which are the better ones?

2. For structure-to-structure alignment the alignment problem is NP-complete. How does the CE method compare to existing heuristics?

3. How easy is it implement the algorithm on a parallel computer. How does it perform on a parallel computing cluster?

4. What is the relationship with other MBO algorithms, and how can we use this to further increase the efficiency?

# 5 Consensus Sequences

Consider a set of $k$ sequences $\mathcal{S} = \{S_1, \ldots, S_k\}$ of lengths $L_1, \ldots, L_k$ respectively. A *consensus sequence* for the sequences of $\mathcal{S}$ is a sequence $X$ that minimises

$$\sum_{i=1}^{k} d(X, S_i)$$

where $d(X, S_i)$ is the edit distance between $X$ and $S_i$.

**Remark 3** A consensus sequence $X$ can be used to induce a multiple sequence alignment of $\mathcal{S}$ by first forming pairwise alignments between $X$ and each $S_i$.

## Consensus Sequences by CE

Now let $\mathcal{X}$ be the set of sequences formed from the letters A, C, G, T with lengths in the range $L_{min} \leq L \leq L_{max}$. We wish to find a consensus sequence by model-based optimisation:

1. Simulate a sample in $\mathcal{X}$ in accordance with a probabilistic model,
2. Update the model in light of the sample to produce a better scoring sample next time.

We need to specify

1. how we generate the samples ( i.e., candidate consensus sequences),
2. how we update the model.

```
TCCTTGTCGTATAAACTAATACACCAGTCTTGTAAACCGAAGATGAAAACCTTTTT
TCCTTGTAGTATAAACTAATACACCAGTCTTGTCGCCGGAGATGAAAACCTTTTTT
TCCTTGTCGTATAAACTAATACACCAGTCTTGTAAACCGAAGATGAAAACCTTTTT
TCCTTGTAGTATAAACTAATACACCCGTCTTGTACGCCGGAGATGAAAACCTTTTT
TCCTTGTAGTAGAAACTAATACACCAGTCGTGTAAACCGGAGATGAAAACCTTTTT
TCCTTGTAGTAGAAACTAATACACCAGTCTTGTAAACCGGAGATGAAAACCTTTTT
TCCTTGTCGTATAAACTAATACACCAGTCTTGTACGCCGGAGATGAAAACCTTTTT
TCCTTGTAGTATAAACTAATACACCAGTCTTGTAAGCCGGAGATGAAAACCTTTTT
TCCTTGTCGTATAAACTAATACACCAGTCTTGTAAACCGGAGATGCAAACCTTTTT
TCCTTGTAGTATAAACTAATACACCAGTCTTGTAAGCCGGAGATGAAAACCCTTTT
TCCTTGTCGTATAAACTAATACACCAGTCTTGTAAACCGAAGATGAAAACCTTTTT
TCCTTGTAGTATAAACTAATACACCAGTCTTGTAAGCCGGAGATGAAAACCTTTTT
TCCTTGTAGTATAAACTAATACACCAGTCTTGTAAGCCGGAGATGAAAACCTTTTT
TCCCCGCAGTATAGACTAACACACCAGTCTTGTAAGCCGGAGATGAAAACCCTTTT
TCCTTGTGGTATAGACTAATACACCAGTCTTGCAAACCGAAGATGAGAACCTTTTT
TCCTTGTAGTACAAACTAACACACCAGTCTTGTAAACCGGAGATGAAAACCTTTCT
TCCTTGTAGTATAAACTAATACACCAGTCTTGTAAACCGGAGATGAAAACCTTTTT
```

Table 3: An example of a consensus sequence