Appendix A

In this appendix we expand on the description of the FRESCO algorithm and some of its speedups. We adopt the *recend*, *recbegin*, *diagend* and *diagbegin* definitions developed in section 3, as well as the *{A, B, C, D}* and matrix *M* notations.

A.1 First Values

Initially, we assign a value of 0 to *M[0,0]*, as well as all squares within the first row or column. In this way, when considering a square whose A point is *M[0, 0]* itself, the score up to this square will be automatically 0, with no additional processing necessary in the recursion.

A.2 Note about Recursion Implementation

The recursion procedure is computed for every cell C in M[0,0] and is implemented via three nested loops, iterating over all possibilities (in order, from innermost to outermost loop) of *recend* (D), *diagstart* (B) and *recstart* (A) points. This order allows us to implement the speedups in the manner described.

A.3 SMAWK / Ranges speedup

In section 3.2 we describe a speedup in the context of the SMAWK algorithm. We have implemented this feature using a 'ranges' construct, whereby instead of iterating for all points D for each {A, B, C} set, we create ranges of points D for which the *same* optimal {A, B} choice exists. To accomplish this, we first note that if for two nonconsecutive points D_1 and D_2, we find the same optimal {A, B} pair (remembering that C is fixed) (i.e. the same *recstart* and *diagstart* maximize the path though {C, D_1} and {C, D_2}), and if the scoring scheme does not change concavity, then all points between D_1 and D_2 are maximized by that same {A, B}.

To see why this is true, we consider the contradicting case where a point D' between D_1 and D_2 could be optimized by a different {A, B} pair, say {A', B'}. We let *f(x)* be our scoring scheme, for the fixed {A, B, C}, with x indicating the change in size of rectangle as we vary D. Let *g(x)* be similarly defined for {A', B', C}, and note that *g* is simply *f* shifted in the x - y plane (i.e. $g = f(a + x) + b$). Further, observe that we need $f(D_1) > g(D_1)$, since otherwise {A', B'} would optimize D_1. Similarly, $f(D_2) > g(D_2)$ and $f(D') < g(D')$. Thus, we need *g* and *f* to meet at least twice. But by the proof below, this would imply that *f* changes concavity at some point (*). Hence, assuming the scoring scheme does not change concavity in the size of the rectangle, we conclude that if {A, B} maximizes the alignment score through D_1 and D_2, all points between D_1 and D_2 are maximized by {A, B} as well. *Note*: If $f(D_1) = g(D_1)$ and $f(D_2) = g(D_2)$ then either $f(x) = g(x)$ for all x, or we can move along D until this is not true. This case is rare in practice.

Since, given same concavity, all points between D_1 and D_2 are maximized by the same {A, B}, we implement the *ranges* speedup via binary search on the set of points *D* possible for the given point C.

Proof of (*):
Note: Rolle's Theorem (R): $\exists a,b \mid f(a)=0 \wedge f(b)=0 \quad \Rightarrow \quad \exists c \in (a,b) \mid f'(c)=0$

Definitions: We let our scoring function be $f(x)$, Then by the above introduction,
$g(x)=f(x+\alpha)+\beta \quad \Rightarrow \quad g'(x)=f'(x+\alpha)$, α, β nonzero.
Also, let $h(x)=f(x)-g(x)$.

Now assume the functions intersect in two points. I.e.
$\exists a,b \mid f(a)=g(a) \wedge f(b)=g(b)$
$\Rightarrow \quad h(a)=0 \wedge h(b)=0$
$\Rightarrow \quad \exists c \mid h'(c)=0 \quad (R)$
$\Rightarrow \quad f'(c)=g'(c)$
$\Rightarrow \quad f'(c)=f'(c+\alpha)$

For simplicity, let $d=c+\alpha$, $\xi=f'(c)=f'(c+\alpha)$ and $j(x)=f'(x)-\xi$

We have
$j(c)=0 \& j(d)=0$
$\Rightarrow \quad \exists \varepsilon \mid j'(\varepsilon)=0 \quad (R)$
$\Rightarrow \quad f''(\varepsilon)=0$

Hence there exists a point of inflection at ε

Hence $\exists a,b \mid f(a)=g(a) \wedge f(b)=g(b) \quad \Rightarrow \quad \exists c \mid f''(c)=0$
i.e. If the function intersects in at least two points, $\exists x \mid f$ has an inflection point.
$\Leftrightarrow$ If concavity same $\forall x$ => function does not intersect at more than 1 point, which is what we set out to prove.

(This requires well-behaved functions at ε (no cusps, etc))

Appendix B

In this appendix we detail the gap frequencies analysis. The alignment accuracy is calculated via the procedure introduced by Pollard et al, 2004. Please refer to that paper for the details.

The gap accuracy is calculated using the following the formula:

$$1 - \frac{1}{m} \left| \frac{\sum_{0}^{n} \frac{t-a}{\frac{t+a}{2}}}{n} \right|$$

Where t is the number of gaps in the evolved ('true') alignment, a represents the number of gaps in the generated alignment, n is the number of alignments and m is a normalization factor. The normalization and inversion (via subtraction from one) are computed for easy visualization of the resulting gap frequencies.