S1 Derivation of the difference recurrence relations and proof of the bounding formulae

The semi-global alignment problem (Equation 1 in the main text) is defined for \( i \geq 0 \) and \( j \geq 0 \). In the following subsections, we use simplified formulae, where \( i \geq 1 \) and \( j \geq 1 \), to describe the process of derivation of the difference recurrence relations and proof of the bounding formulae.

\[
S[i, j] = \max \begin{cases} 
S[i - 1, j - 1] + s(a_{i-1}, b_{j-1}) \\
E[i - 1, j] - G_{eH} \\
F[i, j - 1] - G_{eV}
\end{cases} \quad (4)
\]

\[
E[i, j] = \max \begin{cases} 
S[i, j] - G_{oH} \\
E[i - 1, j] - G_{eH}
\end{cases} \quad (5)
\]

\[
F[i, j] = \max \begin{cases} 
S[i, j] - G_{oV} \\
F[i, j - 1] - G_{eV}
\end{cases} \quad (6)
\]

The initial conditions (where \( i = 0 \) or \( j = 0 \)) are as follows:

\[
S[i, j] = \begin{cases} 
0 & (i = 0, j = 0) \\
-G_{oV} - j \cdot G_{eV} & (i = 0, j \neq 0) \\
-G_{oH} - i \cdot G_{eH} & (i \neq 0, j = 0)
\end{cases} \quad (7)
\]

\[
E[i, j] = \begin{cases} 
-G_{oH} - i \cdot G_{eH} & (j = 0) \\
-\inf & (i = 0)
\end{cases} \quad (8)
\]

\[
F[i, j] = \begin{cases} 
-G_{oV} - j \cdot G_{eV} & (i = 0) \\
-\inf & (j = 0)
\end{cases} \quad (9)
\]

S1.1 Derivation of the difference recurrence relations

Difference values \( \Delta H[i, j] \) are defined for \( i \geq 1 \), and \( \Delta V[i, j] \) for \( j \geq 1 \). \( \Delta E[i, j] \) and \( \Delta F[i, j] \) are defined across the cells in the whole DP matrices (\( i \geq 0 \) and \( j \geq 0 \)).

\[
\Delta H[i, j] = S[i, j] - S[i - 1, j] \quad (i \geq 1)
\]

\[
\Delta V[i, j] = S[i, j] - S[i, j - 1] \quad (j \geq 1)
\]

\[
\Delta E[i, j] = E[i, j] - S[i, j] \quad (i \geq 0, j \geq 0)
\]

\[
\Delta F[i, j] = F[i, j] - S[i, j] \quad (j \geq 0)
\]
S1.1.1 \( \Delta H \) and \( \Delta V \)

First, we derive the update formulae of \( \Delta H \) by substituting the right-hand sides of the update formula of \( S \) (Eq 4) into \( S[i, j] \) in the definition of \( \Delta H \) (Eq 10).

\[
\Delta H[i, j] = S[i, j] - S[i - 1, j]
\]

\[
= \max \left\{ \begin{array}{l} S[i - 1, j - 1] + s(a_{i-1}, b_{j-1}) \\ E[i - 1, j] - G_{eH} \\ F[i, j - 1] - G_{eV} \end{array} \right\} - S[i - 1, j]
\]

\[
\Delta V[i - 1, j] = \Delta H[i, j] - \Delta V[i - 1, j - 1] - \Delta E[i - 1, j]
\]

\[
= \max \left\{ \begin{array}{l} S[i - 1, j - 1] + s(a_{i-1}, b_{j-1}) \\ E[i - 1, j] - S[i - 1, j] - G_{eH} \\ F[i, j - 1] - S[i - 1, j] - G_{eV} \end{array} \right\} - \Delta V[i - 1, j]
\]

\[
A[i, j] = \max \left\{ \begin{array}{l} s(a_{i-1}, b_{j-1}) \\ \Delta E[i - 1, j] + \Delta V[i - 1, j] - G_{eH} \\ \Delta F[i, j - 1] + \Delta H[i, j - 1] - G_{eV} \end{array} \right\}
\]

The update formula for \( \Delta V \) is derived similarly:

\[
\Delta V[i, j] = A[i, j] - \Delta H[i, j - 1]
\]

S1.1.2 \( \Delta E \) and \( \Delta F \)

\[
\Delta E[i, j] = E[i, j] - S[i, j]
\]

\[
= \max \left\{ \begin{array}{l} S[i, j] - G_{oH} \\ E[i - 1, j] - S[i, j] - G_{eH} \end{array} \right\} - S[i, j]
\]

\[
\Delta F[i, j] = \Delta E[i, j] - \Delta V[i, j] - \Delta H[i, j]
\]

\[
= \max \left\{ \begin{array}{l} -G_{oH} \\ E[i - 1, j] - S[i, j] - G_{eH} \end{array} \right\} - \Delta V[i, j] - \Delta H[i, j]
\]

The update formula for \( \Delta F \) is obtained in a similar fashion:

\[
\Delta F[i, j] = \max \left\{ \begin{array}{l} -G_{oV} \\ \Delta F[i, j - 1] - \Delta V[i, j] - \Delta H[i, j] \end{array} \right\}
\]

S1.1.3 Initial conditions

These are defined across the cells in the left edge column \( i = 0 \) and the top edge row \( j = 0 \). From Equation 4, the initial conditions for \( \Delta H \) are derived as follows:

\[
\Delta H[i, j] = S[i, j] - S[i - 1, j] = \begin{cases} G_{oH} + G_{eH} & (i = 1, j = 0) \\ G_{eH} & (i \geq 2, j = 0) \end{cases}
\]
For the initial conditions for $\Delta V$:

$$
\Delta V[i, j] = S[i, j] - S[i, j - 1] = \begin{cases} 
G_{ov} + G_{ev} & (i = 0, j = 1) \\
G_{ev} & (i = 0, j \geq 2)
\end{cases} 
$$  

(20)

The initial conditions for $\Delta E$ and $\Delta F$ are defined as shown below. The differences from the $-\inf$ value are clipped to $-G_{ov}$ and $-G_{ov}$, which are the minimum values of $\Delta E$ and $\Delta F$, as we prove in Section S1.2.3. This modification does not cause a gap penalization error at the edges as pointed out by Flouri et al. (2015) because this setting guarantees $E[0, j] - G_{ev} \leq S[0, j] - G_{ov} - G_{ev}$ and $F[i, 0] - G_{ev} \leq S[i, 0] - G_{ov} - G_{ev}$ for the first updates of $\Delta E$ (where $i = 1$) and $\Delta F$ ($j = 1$), respectively.

$$
\Delta E[i, j] = E[i, j] - S[i, j] = \begin{cases} 
0 & (j = 0) \\
-G_{ov} & (i = 0)
\end{cases} 
$$  

(21)

$$
\Delta F[i, j] = F[i, j] - S[i, j] = \begin{cases} 
0 & (i = 0) \\
-G_{ov} & (j = 0)
\end{cases} 
$$  

(22)

### S1.2 Proof of the bounding formulae

We use the following lemma in the proof:

**Lemma 1** If $p = \max\{q, r\}$, then $p \geq q$ and $p \geq r$ always hold.

#### S1.2.1 Lower bounds of $\Delta E$ and $\Delta F$

According to Equation 5 and Lemma 1, $E[i, j] \geq S[i, j] - G_{ov}$ always holds for any $i \geq 1$ and $j \geq 1$. Then, the following inequality is instantly derived:

$$
\Delta E[i, j] = E[i, j] - S[i, j] \geq -G_{ov} 
$$  

(23)

A similar process is applicable to $\Delta F$:

$$
\Delta F[i, j] = F[i, j] - S[i, j] \geq -G_{ov} 
$$  

(24)

#### S1.2.2 Upper bounds of $\Delta E$ and $\Delta F$

From Equation 4 and 5, the following transformation is obtained:

$$
\Delta E[i, j] = E[i, j] - S[i, j] \\
= \max \left\{ \frac{S[i, j] - G_{ov}}{E[i - 1, j] - G_{ev}} \right\} - S[i, j] \\
= \max \left\{ \frac{-G_{ov}}{E[i - 1, j] - S[i, j] - G_{ev}} \right\} \\
= \max \left\{ \frac{-G_{ov}}{E[i - 1, j] - \max \left\{ \frac{S[i - 1, j - 1] + s(a_{i-1, b_{j-1}})}{E[i - 1, j] - G_{ev}} \right\} - G_{ev}} \right\} - G_{ev} \\
= \max \left\{ \frac{-G_{ov}}{E[i - 1, j] - \min \left\{ \frac{-S[i - 1, j - 1] - s(a_{i-1, b_{j-1}})}{-E[i - 1, j] + G_{ev}} \right\} - G_{ev}} \right\} \\
= \max \left\{ \frac{-G_{ov}}{\min \left\{ \frac{-S[i - 1, j - 1] + E[i - 1, j] - s(a_{i-1, b_{j-1}}) - G_{ev}}{0} \right\} - G_{ev}} \right\} \\
= \max \left\{ \frac{-G_{ov}}{B_H[i, j]} \right\} 
$$  

(25)

A new variable, $B_H[i, j]$, is introduced to replace the ternary minimum block in the formula. The variable is evaluated by means of Lemma 1:

$$
B_H[i, j] = \min \left\{ \frac{-S[i - 1, j - 1] + E[i - 1, j] - s(a_{i-1, b_{j-1}}) - G_{ev}}{0} \right\} \leq 0 
$$  

(26)

Thus, variable $\Delta E[i, j]$ is bounded from above:

$$
\Delta E[i, j] \leq 0 
$$  

(27)

Similar derivation is applicable to $\Delta F$:

$$
\Delta F[i, j] \leq 0 
$$  

(28)
Lemma 2
We first postulate the following lemma:

\[ \Delta S1.2.4 \text{ Upper bounds of } V \]

Similarly reasoning applies to \( \Delta \logically\ follows:\)

Proof: The update formula for \( S \) is defined as in Equation 4 for \( i \geq 1 \) and \( j \geq 1 \). Then the following transformation logically follows:

\[
S[i, j] = \max \left\{ \begin{array}{l}
S[i - 1, j - 1] + s(a_{i-1}, b_{j-1}) \\
E[i - 1, j] - G_{e_H} \\
F[i, j - 1] - G_{e_V}
\end{array} \right. \\
\]

\[
= \max \left\{ \begin{array}{l}
S[i - 1, j - 1] + s(a_{i-1}, b_{j-1}) \\
\max \left\{ \begin{array}{l}
S[i - 1, j - 1] - G_oH - G_{e_H} \\
E[i - 2, j] - G_{e_H} \\
F[i, j - 1] - G_{e_V}
\end{array} \right) - G_{e_H}
\end{array} \right. \\
\]

\[
= \max \left\{ \begin{array}{l}
S[i - 1, j - 1] + s(a_{i-1}, b_{j-1}) \\
\max \left\{ \begin{array}{l}
S[i - 1, j - 1] - G_oH - G_{e_H} \\
S[i - 2, j] - G_oH - 2G_{e_H}
\end{array} \right)
\end{array} \right. \\
\]

\[
= \max \left\{ \begin{array}{l}
S[i, j - 1] - G_{e_V} \\
\max_{1 \leq k \leq i} \left\{ S[i - k, j] - G_oH - k \cdot G_{e_H} \right) - G_{e_H}
\end{array} \right. \\
\]

\[
\geq \max_{1 \leq k \leq i} S[i - k, j] - G_oH - k \cdot G_{e_H}
\]

In the last transformation, we used Lemma 1. Hence,

\[
S[i, j] - S[i - k, j] \geq -G_oH - k \cdot G_{e_H} \quad \text{where } 1 \leq k \leq i
\]

Similarly, for the vertical:

\[
S[i, j] - S[i, j - l] \geq -G_oV - l \cdot G_{e_V} \quad \text{where } 1 \leq l \leq j
\]
Using Lemma 2, we next prove the following lemma, which postulates that the diagonal difference is always bounded by the maximum value in the substitution matrix:

**Lemma 3** \( S[i, j] - S[i - 1, j - 1] \leq M \) for any \( i \geq 1 \) and \( j \geq 1 \)

**Proof:** Let us assume that the following inequality holds for any \( 1 \leq s \leq i \) and \( 1 \leq t \leq j \) except for the \( s = i \) and \( t = j \) pair:

\[
S[s, t] - S[s - 1, t - 1] \leq M
\]

Then, we evaluate the upper bound of the \( S[i, j] \) value using the following transformation and three resulting terms (i)–(iii):

\[
S[i, j] = \max_{1 \leq k \leq i} \left\{ \begin{array}{l}
S[i - 1, j - 1] + s(a_{i-1}, b_{j-1}) \\
E[i - 1, j] - G_{eh} \\
F[i, j - 1] - G_{ev}
\end{array} \right. \\
= \max_{1 \leq k \leq i} \left\{ \begin{array}{l}
S[i - 1, j - 1] + s(a_{i-1}, b_{j-1}) \\
S[i - 1, j] - G_{oh} - G_{eh} \\
S[i - 1, j] - G_{oh} - 2G_{eh}
\end{array} \right.
\]

**Case (i):** When the first term (i) reaches the maximum of the ternary maximum block, the equation below holds for \( S[i, j] \) and \( S[i - 1, j - 1] \):

\[
S[i, j] - S[i - 1, j - 1] \leq \max_{p, q} s(p, q) = M
\]

Thus, Equation 35 is also true at \((i, j)\) in this case.

**Case (ii):** When the second term (ii) reaches the maximum, the case is further subdivided into the following two subcases (a) and (b):

\[
S[i, j] = \max_{1 \leq k \leq i} \left\{ S[i - k, j] - G_{oh} - k \cdot G_{eh} \right. \\
= \max_{1 \leq k \leq i} \left\{ S[0, j] - G_{oh} - i \cdot G_{eh} \right. \}
\]

The first term (a), where \( k = i \), is evaluated as follows by means of Equation 35, Lemma 2(horizontal), and the maximum of the initial conditions for \( \Delta V \) (Eq 20):

\[
S[0, j] - G_{oh} - i \cdot G_{eh} = S[0, j] + (S[0, j - 1] - S[0, j - 1]) + (S[i - 1, j - 1] - S[i - 1, j - 1] - G_{oh} - i \cdot G_{eh})
\]

The second term (b), where \( 1 \leq k \leq i - 1 \), is evaluated as follows:

\[
\max_{1 \leq k \leq i - 1} S[i - k, j] - G_{oh} - k \cdot G_{eh} = \max_{1 \leq k \leq i - 1} \left\{ (S[i - k, j] - S[i - k - 1, j] - G_{oh} - k \cdot G_{eh}) \right. \\
= \max_{1 \leq k \leq i - 1} \left\{ (S[i - k, j] - S[i - k - 1, j] - G_{oh} - k \cdot G_{eh}) \right. \}
\]
Thus, Equation 35 also holds at \((i, j)\) in this case:

\[
S[i, j] - S[i - 1, j - 1] \leq \max \left\{ \begin{array}{l}
S[i - 1, j - 1] - G_{eH} - G_{eV} \\
\max_{1 \leq k \leq i} M + S[i - 1, j - 1]
\end{array} \right\} - S[i - 1, j - 1]
\]

\[
= \max \left\{ \begin{array}{l}
-G_{eH} - G_{eV} \\
\max_{1 \leq k \leq i} M
\end{array} \right\} = M
\]

\[(41)\]

**Case (iii):** Similar to Case (ii), with Lemma 2 (vertical).

**S1.2.5**

Next, we evaluate another antidiagonal interlayer difference (i.e., the difference between a pair of cells at the same coordinates in the DP matrices) to state another lemma:

**Lemma 4**

\[
E[i - 1, j] - S[i, j] \leq M + 2G_{eH} \text{ for any } i \geq 1, j \geq 1.
\]

\[
F[i, j - 1] - S[i - 1, j] \leq M + 2G_{eV} \text{ for any } i \geq 1, j \geq 1.
\]

**Proof:**

\[
E[i - 1, j] = \max \left\{ \begin{array}{l}
S[i - 1, j] - G_{oH} \\
E[i - 2, j] - G_{eH}
\end{array} \right\}
\]

\[
= \max \left\{ \begin{array}{l}
S[i - 1, j] - G_{oH} \\
S[i - 2, j] - G_{oH} - G_{eH}
\end{array} \right\}
\]

\[
= \max_{0 \leq k \leq i - 1} S[i - k - 1, j] - G_{oH} - k \cdot G_{eH}
\]

\[
= \max \left\{ \begin{array}{l}
S[i, j] - G_{eH} - (i - 1) \cdot G_{eH} \\
\max_{0 \leq k \leq i - 2} S[i - k - 1, j] - G_{oH} - k \cdot G_{eH} \quad (k = i - 1) \\
\max_{0 \leq k \leq i - 2} S[i - k - 1, j] - G_{oH} - k \cdot G_{eH} \quad (0 \leq k \leq i - 2)
\end{array} \right\}
\]

\[(42)\]

**Case (i):** The first term (i) is evaluated from above by means of the maximum value of the initial conditions for \(\Delta V\) and Lemma 2 as shown below:

\[
S[0, j] - G_{oH} - (i - 1) \cdot G_{eH} = S[0, j] + (S[0, j - 1] - S[0, j - 1]) + (S[i, j - 1] - S[i, j - 1]) - G_{oH} - (i - 1) \cdot G_{eH}
\]

\[
= (S[0, j] - S[0, j - 1]) + (S[0, j - 1] - S[i, j - 1]) + S[i, j - 1] - G_{oH} - (i - 1) \cdot G_{eH}
\]

\[
\leq -G_{eV} + G_{oH} + i \cdot G_{eH} + S[i, j - 1] - G_{oH} - (i - 1) \cdot G_{eH}
\]

\[
= -G_{eV} + G_{eH} + S[i, j - 1]
\]

**Case (ii):** The second term (ii) is evaluated using Lemma 2 and 3:

\[
\max_{0 \leq k \leq i - 2} S[i - k - 1, j] - G_{oH} - k \cdot G_{eH} = \max_{0 \leq k \leq i - 2} S[i - k - 1, j] + (S[i - k - 2, j - 1] - S[i - k - 2, j - 1])
\]

\[
+ (S[i, j - 1] - S[i, j - 1]) - G_{oH} - k \cdot G_{eH}
\]

\[
= \max_{0 \leq k \leq i - 2} S[i - k - 1, j] - G_{oH} - k \cdot G_{eH}
\]

\[
\leq \max_{0 \leq k \leq i - 2} M + G_{oH} + (k + 2) \cdot G_{eH} + S[i, j - 1] - G_{oH} - k \cdot G_{eH}
\]

\[
= M + 2G_{eH} + S[i, j - 1]
\]

\[(44)\]

Hence,

\[
E[i - 1, j] - S[i, j - 1] \leq \max \left\{ \begin{array}{l}
-G_{eV} + G_{eH} + S[i, j - 1] \\
M + 2G_{eH} + S[i, j - 1]
\end{array} \right\} - S[i, j - 1]
\]

\[
= \max \left\{ \begin{array}{l}
-G_{eV} + G_{eH} \\
M + 2G_{eH}
\end{array} \right\} = M + 2G_{eH}
\]

\[(45)\]

The proof is similar for \(F\):

\[
F[i, j - 1] - S[i - 1, j] \leq M + 2G_{eV}
\]

\[(46)\]
Finally, we derive the upper bound of $\Delta H$ using Equation 27 and Lemmas 2 and 4:

$$\Delta H[i, j] = S[i, j] - S[i-1, j]$$

$$= \max \left\{ \begin{array}{l}
S[i - 1, j - 1] + s(a_{\text{i} - 1}, b_{j - 1}) \\
E[i - 1, j] - G_H \\
F[i, j - 1] - G_E
\end{array} \right\} - S[i - 1, j]$$

$$= \max \left\{ \begin{array}{l}
-(S[i - 1, j] - S[i - 1, j - 1]) + s(a_{\text{i} - 1}, b_{j - 1}) \\
(E[i - 1, j] - S[i - 1, j]) - G_H \\
(F[i, j - 1] - S[i - 1, j]) - G_E
\end{array} \right\}$$

$$\leq \max \left\{ \begin{array}{l}
M + G_{ov} + G_E \\
-G_H \\
M + G_E
\end{array} \right\}$$

$$= M + G_{ov} + G_E$$

Hence,

$$\Delta H[i, j] \leq M + G_{ov} + G_E$$

Similarly, for $\Delta V[i, j]$:

$$\Delta V[i, j] \leq M + G_{oh} + G_{eh}$$
S1.3 Derivation of the offsetted difference recurrence relation

The definitions of the difference DP matrices with an offset and of the substitution matrix with an offset are the same as in the main text:

\[
\Delta H_G[i, j] = \Delta H[i, j] + G_{oh} + G_{eh}
\]

(50)

\[
\Delta V_G[i, j] = \Delta V[i, j] + G_{ov} + G_{ev}
\]

(51)

\[
\Delta E'_G[i, j] = \Delta E[i, j] + \Delta V[i, j] + G_{oh} + G_{ov} + G_{ev}
\]

(52)

\[
\Delta F'_G[i, j] = \Delta F[i, j] + \Delta H[i, j] + G_{ov} + G_{oh} + G_{eh}
\]

(53)

\[
s_G(x, y) = s(x, y) + G_{oh} + G_{eh} + G_{ov} + G_{ev}
\]

(54)

S1.3.1 \( \Delta H_G \) and \( \Delta V_G \)

\[
\Delta H_G[i, j] = \Delta H[i, j] + G_{oh} + G_{eh}
\]

\[
= A[i, j] - \Delta V[i - 1, j] + G_{oh} + G_{eh}
\]

\[
= \max \left\{ s(a_{i-1}, b_{j-1}) \right\}
\]

\[
\Delta E'[i - 1, j] + \Delta V[i - 1, j] - G_{eh} \right\}
\]

\[
- \Delta V[i - 1, j] + G_{oh} + G_{eh}
\]

\[
(55)
\]

\[
\Delta V_G[i, j] = A[i, j] - \Delta V[i - 1, j]
\]

\[
A_G[i, j] \text{ is defined as in the main text:}
\]

\[
\Delta V_G[i, j] = A_G[i, j] - \Delta H_G[i, j - 1]
\]

(57)

S1.3.2 \( \Delta E'_G \) and \( \Delta F'_G \)

\[
\Delta E'_G[i, j] = \Delta E[i, j] + \Delta V[i, j] + G_{oh} + G_{ov} + G_{ev}
\]

\[
= \max \left\{ \frac{s(a_{i-1}, b_{j-1})}{\Delta E'[i - 1, j]} \right\}
\]

\[
\Delta E'[i - 1, j] + \Delta V[i - 1, j] - A_{G}[i, j] - A_{G}[i, j] - \Delta H[i, j - 1] + G_{oh} + G_{ov} + G_{ev}
\]

\[
- \Delta H[i, j - 1] + G_{eh} - G_{ev}
\]

\[
\Delta H_G[i, j - 1]
\]

\[
(58)
\]

Likewise, for \( \Delta F'_G[i, j] \):

\[
\Delta F'_G[i, j] = \max \left\{ \frac{s(a_{i-1}, b_{j-1})}{\Delta F'[i - 1, j]} \right\}
\]

\[
\Delta F'[i - 1, j] + \Delta V[i - 1, j] + 2G_{oh} + G_{ov} + G_{ev} - \Delta H[i, j - 1] - G_{eh} - G_{ev}
\]

(59)
S1.4 Bounding formulae for difference DP matrices with an offset

S1.4.1 $\Delta H_G$ and $\Delta V_G$

The bounds of $\Delta H_G$ instantly follow from Inequality 30 and 48 after addition of the gap penalty offsets:

$$0 \leq \Delta H_G[i, j] \leq M + G_{oh} + G_{eh} + G_{ov} + G_{ev}$$

(60)

Similarly, for $\Delta V_G$, from Inequality 31 and 49, we get

$$0 \leq \Delta V_G[i, j] \leq M + G_{oh} + G_{eh} + G_{ov} + G_{ev}$$

(61)

S1.4.2 $\Delta E_G'$ and $\Delta F_G'$

From Lemmas 2 and 4, the upper bound of $\Delta E_G'$ is derived in the following way:

$$\Delta E_G'[i, j] = \Delta E[i, j] + \Delta V[i, j] + G_{oh} + G_{ov} + G_{ev}$$

$$= E[i, j] - S[i, j] - G_{eh} + G_{ov} + G_{ev}$$

$$= \max \left\{ \frac{S[i, j] - G_{eh}}{E[i - 1, j] - G_{eh}} \right\} - S[i, j] - G_{eh} + G_{ov} + G_{ev}$$

$$\leq \max \left\{ \frac{S[i, j] - S[i, j] - G_{eh}}{E[i - 1, j] - S[i, j] - G_{eh}} + G_{ev} \right\}$$

$$\leq \max \left\{ \frac{M + G_{oh} + G_{eh} + G_{ov} + G_{ev}}{M + G_{oh} + G_{eh} + G_{ov} + G_{ev}} \right\}$$

$$= M + G_{oh} + G_{eh} + G_{ov} + G_{ev}$$

(62)

The lower bound is instantly obtained from Inequality 23 via addition of the gap penalty offsets. The complete bounding formulae are shown below:

$$0 \leq \Delta E_G'[i, j] \leq M + G_{oh} + G_{eh} + G_{ov} + G_{ev}$$

(63)

In a similar fashion for $\Delta F_G'$:

$$0 \leq \Delta F_G'[i, j] \leq M + G_{oh} + G_{eh} + G_{ov} + G_{ev}$$

(64)

S2 Library design and implementation details

The library is implemented in the pure C programming language in an object-oriented manner. Given that the C language does not explicitly support classes or object-specialized functions (class methods), we designed APIs to take a pointer to an object instance as the first argument to treat a C function call as a class method call. The return object and other reference arguments are also handled by pointers, and thus we consider an object instance and the pointer to an instance equivalent in the context of argument passing in the description below.

S2.1 Target architectures

The library implies 64-bit little-endian architectures with 128- or 256-bit-wide SIMD instruction and unaligned load/store capability. The possible targets are x86_64 with SSE4.1 or AVX2 (Intel Corporation (2016), Advanced Micro Devices Inc. (2013)), AArch64 with NEON (ARM Ltd. (2017)), and 64-bit PowerPC with AltiVec instructions (in little-endian mode; OpenPOWER Foundation (2017)). The library currently supports only the x86_64 architecture, whereas the architecture-dependent operations are separated from the algorithm implementation into headers, which are specified to provide several abstract vector types (e.g., v32i8,t, v16i8,t, and v2i64,t), operations on them, and several bit manipulation operations like popcnt and trailing zero count. We provided two variants, SSE4.1 and AVX2, in the current implementation.

S2.2 API design

The library is supposed to be a component of seed-and-extend–style alignment algorithms, that is, the library supports only the semi-global extension alignment (not local or global alignment). We also regard the library as thread-safe, with a global immutable configuration context and the thread-local DP matrix context. The global context is initialized with a set of substitution matrix, gap penalties, and several other parameters like the X-drop threshold. The local context is generated from the global object, inheriting the configuration and initializing its own DP matrix and memory arena. The memory management in the local context is not designed to be interthread-portable; thus, the users must not pass any derived object of a local context (an object that is returned from a function that takes a local context as the first argument) to another local context. The following code snippet shows the signature of the two context initialization functions: global and local.\(^1\)

The two boundary arguments, alim and blim, are added to inform the library at the tail address of the user space, which is utilized to index reverse-complemented sequences. A sequence pointer that leads to an address larger than the boundary is

\(^1\)Because this document is not a manual for the library, the detailed description of the parameters and behavior of the APIs is omitted.
treated as a “phantom sequence,” and the reverse-complemented one at the mirrored address is used. This behavior enables library users to save memory for sequences, where only forward ones are kept in memory and reverse-complemented ones are distinguished by mirrored pointers.

```c
/* global context initialization function */
gaba_t * gaba_init (gaba_params_t const * params);

/* local context initialization function */
gaba_dp_t * gaba_dp_init (gaba_t const * global_context,
                          uint8_t const *alim,
                          uint8_t const * blim);
```

The alignment function takes three arguments—“tail object” of the previous band and reference side and query side sequences—and tries to extend alignment after the tail object with two input sequences. The function returns a new tail object with at least one of the following three states: X-drop termination, reference-side sequence depletion, or query side sequence depletion. This behavior enables us to handle the input sequence as a linear concatenation (or list) of subsequences. This feature is introduced to make the API compatible with circular or graphically structured genomic sequences like string graphs. The fill

```
root function is provided to start the banded alignment, which internally creates an “empty” tail object and pass it to the normal matrix fill-in function. The following code snippet is a simple linear-to-linear alignment calculation with a 32-base-long “margin” sequence, which is generally an array of zeros, to ensure that the ends of the input sequences are covered by the band, where r and q are pointers to the reference side and the query side sequence segments, and m is a pointer to the margin, whereas rsp and qsp are respectively start positions in the reference and query. Note that each subsequence is distinguished by a “sequence ID,” which is a 32-bit unsigned number uniquely assigned to the subsequences.

```c
/* build section structs */
gaba_section_t rsec = gaba_build_section(0, r, 0x800000000000);
gaba_section_t qsec = gaba_build_section(0, q, 0x800000000000);
gaba_section_t msec = gaba_build_section(0, m, 0x800000000000);

/* keep current pointers on rp and qp */
gaba_section_t const *rp = &rsec, *qp = &qsec;

/* fill-in the body of the banded matrix */
gaba_fill_t *f = gaba_dp_fill_root(dp, rp, qsec->len,
                                   qp, qsec->len);

/* keep section with the maximum cell value on m */
gaba_fill_t *m = f;

/* fill-in the tail of the banded matrix */
```

```
uint32_t flag = GABA_STATUS_TERM;
do {
  if (f->status & GABA_STATUS_UPDATE_A) {
    rp = &rsec;
  } else {
    if (f->status & GABA_STATUS_UPDATE_B) {
      qp = &qsec;
    }
    flag |= f->status & GABA_STATUS_UPDATE_A | GABA_STATUS_UPDATE_B;
    f = gaba_dp_fill(dp, f, rp, qsp);
  }
  m = (f->max > m->max) ? f : m;
}while (!(flag & f->status));

/* fill-in stage is done, m holds block with the maximum */
```

The tail object also keeps information on the maximum-scoring cell in the band. The search_max function returns a set of reference side and query side sequence IDs and local positions within the subsequences.

```
gaba_pos_pair_t gaba_dp_search_max(gaba_dp_t * local_context,
                                    gaba_fill_t const *section);
```

The traceback function is designed to handle seed-and-extend–style alignment efficiently accepting two tail objects: forward and reverse. The resulting paths are concatenated at their root in opposite directions to generate the complete (full-length) alignment path. It is also possible to insert short matches between the two roots as the seed sequence of the alignment. Passing tail objects to the function results in an alignment object, which contains the score of the alignment, alignment path, and a list of the corresponding sections and breakpoint coordinates for them.

```
gaba_alignment_t * gaba_dp_trace(gaba_dp_t * local_context,
                                   gaba_fill_t const *sections);
```

**S3 Results for SSE 4.1 variants**

Our libgaba and the other SIMD implementations (non-diff and diff-raw) support both SSE4.1 128-bit-wide and AVX2 256-bit-wide vectorizations. We compared the performance of the two vectorization variants on several combinations of CPU microarchitectures and compilers. The detailed configurations of the four machines are listed in Table S1, and the results are shown in Table S2. We tested two additional variants in instruction encoding, REX- and VEX-prefix encoded, for the SSE4.1 ones to investigate the effect of the instruction encoding on performance.
The results showed that the libgaba implementation was generally as fast as or slightly slower than editdist in all the other tested environments when the AVX2 instruction was enabled, regardless of the compiler and its version, indicating that the design of the algorithm and data structures and tuning applied to the library were generally effective for x86_64 processors. It is also noteworthy that the acceleration ratio trends were roughly consistent with the results on the Skylake system, which are presented in the main text.

| CPU arch | Model     | Clock | DRAM speed | OS          | Compilers              | Description                                |
|----------|-----------|-------|------------|-------------|------------------------|--------------------------------------------|
| Ivy Bridge | Core i5-3230M | 2.6GHz | DDR3-1600  | Mac OS X 10.11.6 | Apple clang, clang-4.0, gcc-5.4.0 | MacBook Pro Retina 13” early 2013          |
| Haswell   | Xeon ES-2670 v3 | 2.30GHz | DDR3-1600  | Red Hat Enterprise Linux 6.8 | gcc-4.9.3, Intel C compiler 16.0.3 (gcc-16.0.3) | Shirokane3 at Human Genome Center         |
| Skylake   | Core i5-6260U | 2.80GHz @ boost | DDR4-2133 | Ubuntu 16.04.2 LTS | clang-3.8, gcc-5.4.1 | Intel NUC i5SYH                           |
| Zen       | Ryzen 7-1700 | 3.70GHz @ boost | DDR4-2400 | Ubuntu 17.04 | clang-3.8, gcc-6.3.0 | Personal Desktop                          |

The systems are distinguished by their CPU microarchitectures and the other components (DRAM, OS, and available compilers). The Haswell system is a single node of the Shirokane3 cluster at the Human Genome Center, the University of Tokyo. All the others are personal.

References

Advanced Micro Devices Inc. (2013). AMD64 architecture programmer’s manual volume 1-5. [http://developer.amd.com/resources/developer-guides-manuals/]

ARM Ltd. (2017). ARMv8-A reference manual (issue B.a). [http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0487b.a/index.html]

Flouri, T., Kobert, K., Rognes, T., and Stamatakis, A. (2015). Are all global alignment algorithms and implementations correct? bioRxiv. doi: 10.1101/031500.

Intel Corporation (2016). Intel® 64 and IA-32 architectures software developer manuals. [https://software.intel.com/en-us/articles/intel-sdm/]

OpenPOWER Foundation (2017). Power instruction set architecture (ISA) version 3.0B. [https://openpowerfoundation.org/?resource_lib=power-isa-version-3-0/].
| System   | Arch. | Speed | Compiler | Flags | Fill (ms) | Trace (ms) | Conv (ms) | Total (ms) |
|----------|-------|-------|----------|-------|-----------|-----------|-----------|------------|
| libgaba  | Ivy Bridge | Core i5-3230M | 2.6GHz DDR3-1600 | 64-bit GP reg | SSE4.1 -msse4.2 | 1.296 | 0.072 | 0.057 | 1.425 |
|          |       |       | Apple clang | -march=native | 1.296 | 0.072 | 0.057 | 1.425 |
| non-diff | Ivy Bridge | Core i5-3230M | 2.6GHz DDR3-1600 | 64-bit GP reg | SSE4.1 -msse4.2 | 0.976 | 0.229 | 0.133 | 1.338 |
|          |       |       | Apple clang | -march=native | 0.976 | 0.229 | 0.133 | 1.338 |
|          | Skylake | Core i5-6260U | 2.6GHz DDR4-2133 | 64-bit GP reg | SSE4.1 -msse4.2 | 0.374 | 0.075 | 0.022 | 0.471 |
|          |       |       | Apple clang | -march=native | 0.374 | 0.075 | 0.022 | 0.471 |
| Zen      | Ryzen 7-1700 | 3.7GHz DDR4-2400 | 64-bit GP reg | SSE4.1 -msse4.2 | 0.278 | 0.062 | 0.034 | 0.374 |
|          |       |       | Apple clang | -march=native | 0.278 | 0.062 | 0.034 | 0.374 |

The benchmark setting and the definitions of the Fill, Trace, Conv, and Total columns are the same as in the main text. The detailed specifications of the systems are listed in Table S1. The boldfaced numbers show the fastest variant for each pair system-stage (column). The REX-encoded binaries were generated using architecture flag -msse4.2 to enable both SSE4.1 instructions and the popcnt instruction (included in SSE 4.2). The VEX-encoded binaries were generated with the -mavx flag, where SSE 4.1 instructions were used in the explicit vectorizations and several AVX instructions (e.g., vmovaps) were employed in the automatic loop vectorization and some libc functions like memcpy and memcp. The -march=native flag was applied to enable all the available instructions and optimizations as the fastest baselines on each system. 12