Sequence Alignment

Gap Penalties, Gotoh's Algorithm and Smith/Waterman's Local Alignment

Rolf Backofen

Lehrstuhl für Bioinformatik
Institut für Informatik

Course Bioinformatics I — SS 2010

Gaps

- **problem:** gaps are different in nature - given a fixed number of gaps, a "small number of long gaps" is biologically likelier than a "big number of small gaps"
- **solution:** initiation of gaps is more expensive than extension of an existing gap
- **example:**

$$\begin{array}{ccccccccc} \multicolumn{2}{c}{\overbrace{- \quad -}^{\text{length 2}}} & A & \multicolumn{3}{c}{\overbrace{- \quad - \quad -}^{\text{length 3}}} & G & T & A \\ A & A & A & T & T & T & G & \multicolumn{2}{c}{\underbrace{T \quad -}_{\text{length 1}}} \end{array}$$

$\Rightarrow$ gap costs: $g(2) + g(3) + g(1)$

Gap Penalties

Definition

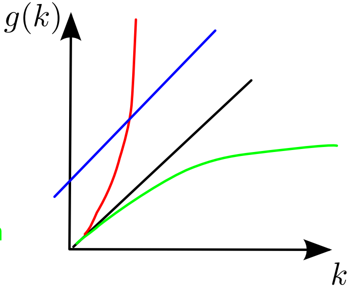
A *gap penalty* is a function $g(k) : \mathbb{N} \to \mathbb{R}$ that is subadditive, i.e.,

$$\forall k, l : g(k+l) \leq g(k) + g(l).$$

A gap penalty is called *affine* if there are $\alpha, \beta \in \mathbb{R}$ such that

$$g(k) = \alpha + \beta k$$

- examples for subadditive:
  $g(k) = \alpha + \beta k \Rightarrow$ affine, very common
  $g(k) = \alpha + \beta k^2$ ⚡
  $g(k) = \alpha + \beta \ln(k) b$ ✓
  $\Rightarrow$ biologically, the best approximation



Gap Penalty and Alignment

- **now:** we want to calculate optimal alignments where gaps are scored with gap penalties.
- **problem:** split like Needleman/Wunsch does not work

$$w\left(\begin{array}{ccc|c} A & A & - & - \\ - & A & G & T \end{array}\right) \qquad w\begin{pmatrix} A & A & - \\ - & A & G \end{pmatrix} + w\begin{pmatrix} - \\ T \end{pmatrix}$$

$$g(1) + w(A, A) + g(2) \qquad \neq \qquad \big(g(1) + w(a, a) + g(1)\big) + g(1)$$

Helps: More Distinction

1. substitution $\surd$

$$\begin{array}{rcc|c} u^\diamond & = & \dots & a_i \\ v^\diamond & = & \dots & b_j \end{array} \quad \Rightarrow \quad D_{i,j} = D_{i-1,j-1} + w(a_i, b_j)$$

2. insertion 1:

$$\begin{array}{rccc|c} u^\diamond & = & \dots & ? & a_i \\ v^\diamond & = & \dots & b_j & - \end{array} \quad \Rightarrow \quad D_{i,j} = D_{i,j} = D_{\underline{i-1},j} + g(1)$$

3. insertion 2:

$$\begin{array}{rccc|cc} u^\diamond & = & \dots & ? & a_{i-1} & a_i \\ v^\diamond & = & \dots & b_j & - & - \end{array} \quad \Rightarrow \quad D_{i,j} = D_{i,j} = D_{\underline{i-2},j} + g(2)$$

$\Rightarrow$ Algorithm of Smith-Waterman-Beyer

Smith-Waterman-Beyer

Theorem (Waterman, Smith and Beyer)

Let $g : \mathbb{N} \to \mathbb{R}$ be a gap penalty and w be cost function on $\Sigma \times \Sigma$. Let $a = a_1 \ldots a_n$ and $b = b_1 \ldots b_m$ be two words in Σ^*. We define $(D_{i,j})$ with $1 \le i \le n$ and $1 \le j \le m$ by

$$\begin{array}{rcl} D_{0,0} & = & 0, \\ D_{0,j} & = & g(j), \\ D_{i,0} & = & g(i), \\ D_{i,j} & = & \min \left\{ \begin{array}{l} \min\limits_{1 \le k \le j} \{D_{i,j-k} + g(k)\}, \\ D_{i-1,j-1} + w(a_i, b_j), \\ \min\limits_{1 \le k \le i} \{D_{i-k,j} + g(k)\}\} \end{array} \right\}. \end{array}$$

Then $D_{i,j} = D(a_1 \ldots a_i, b_1 \ldots b_j)$.

Complexity

$\Rightarrow$ **example:** line 1 $\Rightarrow$ always 1 diagonal
+ 1 up
+ some left

column 1:	2	+	1 (left)
column 2:	2	+	2 (left)

$$D_{1,2} = \min \begin{cases} D_{0,1} + w(a_1, b_2) \\ D_{0,2} + g(1) \\ \min \begin{cases} D_{1,1} + g(1) \\ D_{1,0} + g(2) \end{cases} \end{cases}$$

column n:	2	+	n (left)
$\sum$	$2n$	+	$\frac{n(n+1)}{2}$

cost per cell: $$\frac{2n + \frac{n(n+1)}{2}}{n} \quad \boxed{= 2 + \frac{n+1}{2} = O(n)}$$

$\Rightarrow$ on average a cell cost $O(n)$ for filling
$\Rightarrow$ total: $O(n^3)$ time and $O(n^2)$ space

- example:
  - 2 RNA sequences with $n = 30\,000 = 3 \cdot 10^4$
  - **assume:** computer with 1 Ghz
    $+$ 1 operation per unit

$$\Rightarrow \quad \frac{27 \cdot 10^{12}}{10^9} \begin{aligned}[t] &= 27 \cdot 10^3 \text{ s} \\ &= 27\,000 \text{ s} \\ &\approx 7.5 \text{ h} \end{aligned}$$

- (exercise - how much time would a quadratic algorithm have taken?)

Gotoh's Algorithm for Affine Gap-Penalties

- problem in S–W–B: gaps of any lengths have to be tested in each step
- therefore: using affine gap penalties $g(k) = \alpha + \beta k$

analyzing $D_{i,j}$

1. $\begin{array}{lcl|l} u^\diamond & = & \dots & a_i \\ v^\diamond & = & \dots & b_j \end{array} \Rightarrow$ cost $D_{i,j} = D_{i-1,j-1} + w(a_i, b_j)$
2. $\begin{array}{lcl|l} u^\diamond & = & \dots & a_i \\ v^\diamond & = & \dots & - \end{array} \Rightarrow$ **look at subcases**
   - a. $\begin{array}{lccl|l} u^\diamond & = & \dots & ? & a_i \\ v^\diamond & = & \dots & b_j & - \end{array} \Rightarrow$ cost $D_{i,j} = D_{i-1,j} + g(1)$
   - b. $\begin{array}{lccl|l} u^\diamond & = & \dots & a_j & a_i \\ v^\diamond & = & \dots & - & - \end{array} \Rightarrow$ cost $D_{i,j} = \star - g(k-1) + g(k)$

     $\underbrace{\qquad}_{k \text{ gaps}}$

     $$\begin{aligned} &= \star - \alpha - (k-1)\beta + \alpha + k\beta \\ &= \star + \beta \end{aligned}$$
3. analogous for gaps in a.

- $\star =$ cost for best alignment of $a_1 \dots a_i$ and $b_1 \dots b_j$ ending with a gap in b.

$\Rightarrow$ the length of the gap doesn't matter, since each elongation costs β

$\Rightarrow$ we have the following cases:
a. no gap
b. starting a new gap
c. elongate an existing gap

$\Rightarrow$ saving time because:

- S-W-B: test with all possible gap lengths
- Gotoh: just add β if a gap is elongated

- **comment:** if gap penalty is not affine (e.g. $g(k) = \alpha + \beta \cdot \ln(k)$) then

$$\begin{aligned} D_{i,j} &= \star - g(k-1) + g(k) \\ &= \star - \alpha - \ln(k-1) + \alpha + \ln(k) \\ &= \star + \ln(k) - \ln(k-1) \\ &= \star + \ln(\tfrac{k}{k-1}) \end{aligned}$$

$\Rightarrow$ depends on $k \Rightarrow$ Gotoh's idea doesn't work

Gotoh Matrices

$\Rightarrow$ further matrices needed

- $(D_{i,j})$ cost for alignment of prefixes $(a_1 \dots a_i, b_1 \dots b_j)$
- $(P_{i,j})$ cost for alignment of prefixes $(a_1 \dots a_i, b_1 \dots b_j)$ that ends with a gap in b (i.e., last column is $\begin{pmatrix} a_i \\ - \end{pmatrix}$)
- $(Q_{i,j})$ cost for alignment of prefixes $(a_1 \dots a_i, b_1 \dots b_j)$ that ends with a gap in a (i.e., last column is $\begin{pmatrix} - \\ b_j \end{pmatrix}$)

Gotoh – 1982

- let $g(k) = \alpha + k\beta$ be an affine gap penalty, and let $w : \Sigma \times \Sigma \to \mathbb{R}$ be a cost function.
- recursive definition of matrices $(D_{i,j})$, $(P_{i,j})$, and $(Q_{i,j})$:

$$D_{i,j} \quad = \quad \min \left\{ \begin{array}{l} D_{i-1,j-1} + w(a_i, b_j) \\ P_{i,j} \\ Q_{i,j} \end{array} \right\},$$

with $i, j \geq 1$, where for $1 \leq i \leq |a|$ and $1 \leq j \leq |b|$,

$$P_{i,j} \quad = \quad \min \left\{ \begin{array}{l} D_{i-1,j} + g(1) \\ P_{i-1,j} + \beta \end{array} \right\}$$

$$Q_{i,j} \quad = \quad \min \left\{ \begin{array}{l} D_{i,j-1} + g(1) \\ Q_{i,j-1} + \beta \end{array} \right\}$$

Initialization

- **Initialization:** $D_{i,j}$ as usual: $D_{0,0} = 0$, $D_{0,j} = g(j)$ and $D_{i,0} = g(i)$
- for $P_{i,j}$:
  - recursion only on the first index $(P_{i,j} \to P_{i-1,j} \to \ldots \to P_{0,j})$
  - **hence:** only initialization for $P_{0,j}$.
  - **but:** $P_{0,j}$ is best alignment of ϵ and $b_1 \ldots b_j$ that ends with gap in b $\Rightarrow$ the only possible alignment would be:

$$\begin{array}{cccccc} - & - & \ldots & - & - & \color{red}{-} \\ b_1 & b_2 & \ldots & b_{j-1} & b_j & \color{red}{-} \end{array}$$

$$\color{red}{\underbrace{b_j \ -}_{\textbf{disallowed in alignments!}}}$$

- **Thus:**

$$\begin{aligned} P_{j,0} &= \text{not used} \\ P_{0,j} &= \infty \\ \\ Q_{j,0} &= \infty \\ Q_{0,j} &= \text{not used} \end{aligned}$$

- **order of calculation:**

```
initialization
for i=1 to n
    for j=1 to n
        calculate $P_{i,j}$
        calculate $Q_{i,j}$
        calculate $D_{i,j}$
    end
end
```

Traceback Matrices (tr^D), (tr^P) and (tr^Q)

- simple arrows are not enough (because of jumping between the matrices)
- $\mathrm{tr}^D \in \{{}^D\nwarrow, {}^Q\bullet, {}^P\bullet\}$.

$$\forall i,j>0: \quad \begin{array}{rcl} {}^D\nwarrow \in \mathrm{tr}^D_{i,j} & \Leftrightarrow & D_{i,j} = D_{i-1,j-1} + w(a_i, b_j), \\ {}^Q\bullet \in \mathrm{tr}^D_{i,j} & \Leftrightarrow & D_{i,j} = Q_{i,j}, \\ {}^P\bullet \in \mathrm{tr}^D_{i,j} & \Leftrightarrow & D_{i,j} = P_{i,j}; \end{array}$$

- $\mathrm{tr}^P \in \{{}^D\uparrow, {}^P\uparrow\}$.

$$\forall i,j>0: \quad \begin{array}{rcl} {}^P\uparrow \in \mathrm{tr}^P_{i,j} & \Leftrightarrow & P_{i,j} = P_{i-1,j} + \beta, \\ {}^D\uparrow \in \mathrm{tr}^P_{i,j} & \Leftrightarrow & P_{i,j} = D_{i-1,j} + g(1); \end{array}$$

- $\mathrm{tr}^Q \in \{{}^D\leftarrow, {}^Q\leftarrow\}$.

$$\forall i,j>0: \quad \begin{array}{rcl} {}^Q\leftarrow \in \mathrm{tr}^Q_{i,j} & \Leftrightarrow & Q_{i,j} = Q_{i,j-1} + \beta, \\ {}^D\leftarrow \in \mathrm{tr}^Q_{i,j} & \Leftrightarrow & Q_{i,j} = D_{i,j-1} + g(1). \end{array}$$

- finding alignments:
  - arrows (no matter which matrix) as before:

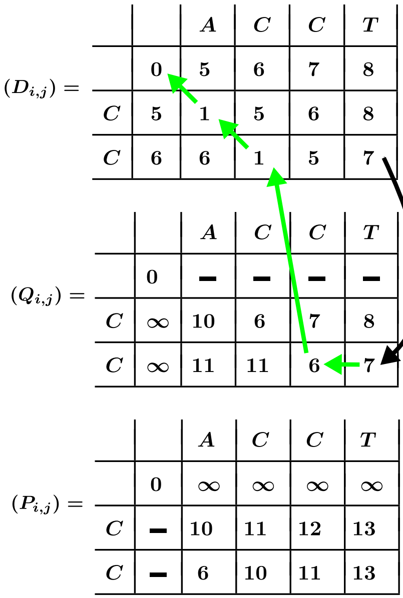
    a : $\nwarrow, \uparrow \Leftrightarrow$ symbol a_i $\quad \leftarrow \Leftrightarrow$ –.

    b : $\leftarrow, \nwarrow \Leftrightarrow$ symbol b_j $\quad \uparrow \Leftrightarrow$ –.
  - points = change of matrix, nothing more

Example

- given: $a = CC$ and $b = ACCCT$.
- cost functions:
  - substitutions: $w(x,y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{else} \end{cases}$
  - gap penalty: $g(k) = 4 + k$ $(\beta = 1)$.
- **wanted:** optimal alignment using Gotoh

Matrix and Traceback



$$
(D_{i,j}) = \begin{array}{c|c|c|c|c|c|}
 & & A & C & C & T \\ \hline
 & 0 & 5 & 6 & 7 & 8 \\ \hline
C & 5 & 1 & 5 & 6 & 8 \\ \hline
C & 6 & 6 & 1 & 5 & 7 \\ \hline
\end{array}
$$

$$
(Q_{i,j}) = \begin{array}{c|c|c|c|c|c|}
 & & A & C & C & T \\ \hline
 & 0 & - & - & - & - \\ \hline
C & \infty & 10 & 6 & 7 & 8 \\ \hline
C & \infty & 11 & 11 & 6 & 7 \\ \hline
\end{array}
$$

$$
(P_{i,j}) = \begin{array}{c|c|c|c|c|c|}
 & & A & C & C & T \\ \hline
 & 0 & \infty & \infty & \infty & \infty \\ \hline
C & - & 10 & 11 & 12 & 13 \\ \hline
C & - & 6 & 10 & 11 & 13 \\ \hline
\end{array}
$$

complete filled matrices

one of the two possible final traceback

- tracebacks:

  1. $D \nwarrow \quad D \nwarrow \quad D \leftarrow \quad Q \leftarrow \quad Q \bullet$

     $$\begin{array}{cccc} \mathrm{C} & \mathrm{C} & - & - \\ \mathrm{A} & \mathrm{C} & \mathrm{C} & \mathrm{T} \end{array}$$

  2. $D \leftarrow \quad D \leftarrow \quad D \nwarrow \quad D \nwarrow$

     $$\begin{array}{cccc} - & - & \mathrm{C} & \mathrm{C} \\ \mathrm{A} & \mathrm{C} & \mathrm{C} & \mathrm{T} \end{array}$$

Needleman-Wunsch with Similarity

- **up to now:**
  - minimal alignment distance wanted

    $w(x,x) = 0 \Rightarrow$ low costs for identical symbols
  - matrix $(D_{i,j})$, where $D_{i,j}$ lowest distance of $a_1..a_i$, $b_1..b_j$

- **now:**
  - maximal similarity wanted

    $s(x,x)$ high $\Rightarrow$ high similarity for identical symbols
  - matrix $(S_{i,j})$, where

    $S_{i,j}$ best similarity for prefixes $a_1 \dots a_i$ and $b_1 \dots b_j$

$\Rightarrow$ **recursion:**

$$S_{i,j} = \max \left\{ \begin{array}{ll} S_{i,j-1} & +s(-,b_j), \\ S_{i-1,j-1} & +s(a_i,b_j), \\ S_{i-1,j} & +s(a_i,-) \end{array} \right\}$$

- **main usage:**
  - local alignment $\Rightarrow$ search for motifs that are locally similar,

    e.g. $a =$ ACAVIAC$\boxed{\text{AIALAG}}$ACG

    $b =$ VVAIV$\boxed{\text{AIALAG}}$YY

Distance vs. Similarity

- why is distance not useful here?

(a) $a =$ $\text{XX}\boxed{\text{AA}}\text{CIXX}$

$b =$ $\text{YY}\boxed{\text{AA}}\text{YYG}$

$D = 0$

$S = 10$

$$\text{Dist } w(x,y) = \begin{cases} 0 & \text{if } x = y \\ 5 & \text{else} \end{cases}$$

$$\text{Sim } s(x,y) = \begin{cases} 5 & \text{if } x = y \\ 0 & \text{else} \end{cases}$$

(b) $a =$ $\text{XX}\boxed{\text{AAAA}}\text{YY}$

$b =$ $\text{YY}\boxed{\text{AAAA}}\text{YY}$

$D = 0$

$S = 20$

$\Rightarrow$ using distances, (a) and (b) are equally good

$\Rightarrow$ but (b) is better local motif

$\Rightarrow$ is represented best by similarity

- Needleman-Wunsch with similarities instead of distances

  $w(x,y) \Rightarrow$ $s(x,y)$, which can be positive or negative
  - no metric
  - positive means similar

Local Alignment

- Smith-Waterman local alignment
  - recursion still working on alignments of prefixes

  $\Rightarrow$ matrix $H(i,j)$, which is the best local alignment of prefixes $a_1 \ldots a_i$ and $b_1 \ldots b_j$, where **initial** gaps (but not **final**) are free

- example: $a = \quad \text{AAC}|^{3}\text{GG}|^{5}\text{TAC}$

  $b = \quad \text{GGGAA}|^{5}\text{GG}|^{7}\text{TGG}$

- two types of gaps: $-$ $=$ gap which is scored

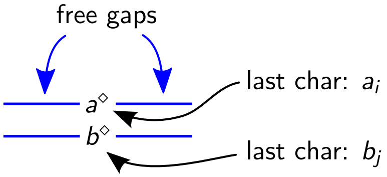
  $\circ$ $=$ initial or final gap (unscored)

- **Then:** $\begin{array}{cccccccc} \circ & \circ & \circ & A & A & C & G & G \\ G & G & G & A & A & - & G & G \end{array} \in H(5,7)$

- **But:** $\begin{array}{cccccc} \circ & \circ & \circ & A & A & C \\ G & G & G & A & A & \circ \end{array} \notin H(3,5)$ albeit $\begin{array}{cccccc} \circ & \circ & \circ & A & A & C \\ G & G & G & A & A & - \end{array} \in H(3,5)$

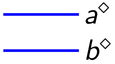
- **Remark 1:** in which cell does one find the final optimal, local alignment?

assume



is optimal, local alignment

- **recall:** H considers alignments of the form



$$\Rightarrow \in H(i,j)$$

- concrete example: $a = GGAAATT$ and $b = CCAAAGG$

$\Rightarrow$ optimum:

$$\underbrace{\begin{array}{ccccccc} G & G & \circ & \circ & A & A & A \\ \circ & \circ & C & C & A & A & A \end{array}}_{S = H(5,5) + 0} \Bigg| \begin{array}{cccc} \circ & \circ & T & T \\ G & G & \circ & \circ \end{array}$$

$\Rightarrow$ search all $H(i,j)$ for maximal value

Smith-Waterman Recursion

- recursion: $H_{i,j} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j) \\ H_{i-1,j} + s(a_i, -) \\ H_{i,j-1} + s(-, b_j) \\ \mathbf{0} \end{cases}$

when other entries < 0 then gap out prefixes $a_1 \dots a_i$ and $b_1 \dots b_j$ for free

- Initialitation: $\begin{aligned} H_{0,0} &= 0, \\ H_{0,j} &= 0, \\ H_{i,0} &= 0 \end{aligned}$

- how to do traceback:
  - start with $H(i,j)$ that is maximal
  - follow directions (maximal entries in DP)

    like Needleman-Wunsch
  - stop when a $H(i,j) = 0$ is reached.

Example

- scoring: $s(x,y) = \begin{cases} +2 & \text{if } x = y \\ -1 & \text{else} \end{cases} \Rightarrow s(-,x) = s(x,-) = -1$
- similarity can be extended to gap penalties (negative values !)
- matrix for $a = CCC$ and $b = ACACCTT$

		A	C	A	C	C	T	T
	0	0	0	0	0	0	0	0
C	0	0	2	1	2	2	1	0
C	0	0	2	1	3	4	3	2
C	0	0	2	1	3	**5**	4	3

- best value (= end of traceback): cell $(3,5)$ with $H_{3,5} = 5$
  $\Rightarrow$ traceback: $\nwarrow \leftarrow \nwarrow \nwarrow$
- associated alignment: $\begin{array}{cccc} C & \_ & C & C \\ C & A & C & C \end{array}$ $\Rightarrow$ Wert:$5 = 2 - 1 + 2 + 2$