Bioinformatics
(Matching and Alignment)

Marc Hellmuth

Aim:
Compare strings to score/evaluate the (dis)similarity between them.

Sequence alignment arises in many fields:

- Molecular biology
- Inexact text matching (e.g. spell checkers; web page search)
- Speech recognition

Biology:
In biomolecular sequences (DNA,RNA,Proteins) high sequence similarity implies significant functional or structural similarity.

Important:
similar function $\not\Rightarrow$ similar structure $\not\Rightarrow$ similar sequences

Sketch: Pattern Matching

Problem: Given text T and pattern P
Aim: Find all occurrences of P in T.

```
Brute-Force:
For $i = 1$ to $|T| - |P| + 1$
/*cmp $T_i \ldots T_{i+|P|-1}$ with $P_1 \ldots P_{|P|}$ */
   For j=1 to |P|
     If $P_j \neq T_{i+j-1}$ then GoTo mismatch
   EndFor
  write i
  mismatch
EndFor
```

Sketch: Improved Pattern Matching

Idea: For string S generate Datastructure $Z_i(S)$

$Z_i(S)$ for $i = 2, \ldots, |S|$ is length of longest substring of S, that starts on position i and is a prefix of S.

If $Z_i(S) > 0$ the interval $[i, i + Z_i(S) - 1]$ is called *Z-box*

Assume we could efficiently compute the $Z_i(S)$ - Does this help us?

Sketch: Improved Pattern Matching

Given text T and pattern P.

Let \$ be a character neither included in T nor in P.

Let $S = P\$T$

Compute $Z_i = Z_i(S)$ for $i = 2, \ldots, |S|$.
(This can be done in $O(|S|)$ time $\rightarrow$ "Datenstrukturen und effiziente Algorithmen")

```
For $i = 1$ to $|T| - |P| + 1$
   If $Z_{i+|P|-1} = |P|$ then write $i$
EndFor
```

(This can be done in $O(|T|)$ time.

$\Rightarrow$ "approximate" pattern matching $\Rightarrow$ alignment

Edit Distance

Edit Operations:

- `I`nsertion of character
- `D`eletion of character
- `R`eplacement of one character by some other one

Edit Distance = Min. Nr. of Edit Operations to transform string *u* to string *v* (equivalent transform string *v* to string *u*)

$$\begin{array}{ccccccc} \mathtt{D} & \mathtt{M} & \mathtt{M} & \mathtt{R} & \mathtt{M} & \mathtt{M} & \mathtt{I} \\ \hline \mathtt{w} & \mathtt{r} & \mathtt{i} & \mathtt{t} & \mathtt{e} & \mathtt{r} & - \\ - & \mathtt{r} & \mathtt{i} & \mathtt{d} & \mathtt{e} & \mathtt{r} & \mathtt{s} \end{array} \quad (\mathtt{M} = \text{Match})$$

Edit Script = string over alphabet $\{I, D, R, M\}$ that describes transformation from *u* to *v*.

Edit Distance Problem: For two strings compute edit distance and optimal edit script.

Example

$u = \texttt{TGCATAT}$ $v = \texttt{ATCCGAT}$

$$u = \texttt{TGCATAT} \xrightarrow{\text{del. last } \texttt{T}} \texttt{TGCATA} \xrightarrow{\text{del. last } \texttt{A}} \texttt{TGCAT} \xrightarrow{\text{add } \texttt{A} \text{ 1.pos}} \texttt{ATGCAT} \xrightarrow{\text{repl. } \texttt{G} \text{ by } \texttt{C} \text{ 3.pos}} \texttt{ATCCAT} \xrightarrow{\text{insert } \texttt{G} \text{ 5.pos}} \texttt{ATCCGAT} = v$$

Edit Distance ≤ 5

$$u = \texttt{TGCATAT} \xrightarrow{\text{ins. } \texttt{A} \text{ 1.pos}} \texttt{ATGCATAT} \xrightarrow{\text{del. } \texttt{T} \text{ 6.pos}} \texttt{ATGCATAT} \xrightarrow{\text{repl. } \texttt{A} \text{ by } \texttt{G} \text{ 5.pos}} \texttt{ATGCGTAT} \xrightarrow{\text{repl. } \texttt{G} \text{ by } \texttt{C} \text{ 3.pos}} \texttt{ATCCGAT} = v$$

Edit Distance ≤ 4 (OPTIMAL?)

(global pairwise) Alignment

Alternative way to edit script: Alignment

For two strings $u = u_1 \dots u_m$ and $v = v_1 \dots v_n$ an *alignment* $\mathscr{A}$ is a matrix with two rows and entries $\mathscr{A}[i, j]$ that are characters from Alphabet Σ (e.g.$\Sigma = \{A, C, G, T\}$) or a *gap* “-” s.t.

- 1st row $=$ u after deleting all gaps
- 2st row $=$ v after deleting all gaps
- in no column are two gaps

w	r	i	t	e	r	-
-	r	i	d	e	r	s

-	T	G	C	A	T	A	T
A	T	C	C	G	-	A	T

Cost-Function	$\delta : \Sigma \cup \{-\} \times \Sigma \cup \{-\} \rightarrow \mathbb{R}_{\geq 0}$
Unit-Cost-Function	$\delta(a, b) = 1$ if $a \neq b$
	$\delta(a, b) = 0$ if $a = b$
Alignment Costs	$w(\mathscr{A}) = \sum_{i=1} \delta(a_i, b_i)$

(global pairwise) Alignment

Alternative way to edit script: Alignment

For two strings $u = u_1 \dots u_m$ and $v = v_1 \dots v_n$ an *alignment* $\mathscr{A}$ is a matrix with two rows and entries $\mathscr{A}[i,j]$ that are characters from Alphabet Σ (e.g.$\Sigma = \{A, C, G, T\}$) or a *gap* “-” s.t.

- 1st row $=$ u after deleting all gaps
- 2st row $=$ v after deleting all gaps
- in no column are two gaps

```
w  r  i  t  e  r  -      -  T  G  C  A  T  A  T
-  r  i  d  e  r  s      A  T  C  C  G  -  A  T
```

Cost-Function	$\delta : \Sigma \cup \{-\} \times \Sigma \cup \{-\} \to \mathbb{R}_{\geq 0}$
Unit-Cost-Function	$\delta(a,b) = 1$ if $a \neq b$
	$\delta(a,b) = 0$ if $a = b$
Alignment Costs	$w(\mathscr{A}) = \sum_{i=1} \delta(a_i, b_i)$

Lemma

Edit Distance of two strings u, v equals the min. alignments costs $w(\mathscr{A})$ between u and v with unit-cost function.

How to compute Edit Distance? Dynamic Programming!

Recurrence Function D (Needleman-Wunsch Algorithm)

Given the strings $u = u_1 \dots u_m$ and $v = v_1 \dots v_n$

Assume $D[i,j]$ are the costs for an optimal alignment of substrings $u_1 \dots u_i$ and $v_1 \dots v_j$, $1 \leq i \leq m$, $1 \leq j \leq n$

$i = 0$: alignment empty string ε and $v_1 \dots v_j$
$j = 0$: alignment $u_1 \dots u_i$ and empty string ε

Init: $D[i,0] = i$; $D[0,j] = j$, $i,j \geq 0$;

Compute

$$D[i,j] = \min \begin{cases} D[i-1,j] & + \quad \delta(u_i, -) \\ D[i-1,j-1] & + \quad \delta(u_i, v_j) \\ D[i,j-1] & + \quad \delta(-, v_j) \end{cases}$$

(δ = unit-cost-function)

Lemma

$D[m,n]$=cost of optimal alignment between u and v.

Backtracing

Given the strings $u = u_1 \dots u_m$ and $v = v_1 \dots v_n$

Tracematrix is an $m \times n$ matrix with $T[i,j] \subseteq \{\leftarrow, \nwarrow, \uparrow\}$.

Init: $T[0,0] = \emptyset$, $T[i,0] = \uparrow$, $T[0,j] = \leftarrow$ for $1 \leq i \leq m$, $1 \leq j \leq n$

$$\text{Set:} \quad \begin{array}{lll} \uparrow \in T[i,j] & \text{if} & D[i-1,j] + \delta(u_i, -) \\ \nwarrow \in T[i,j] & \text{if} & D[i-1,j-1] + \delta(u_i, v_j) \\ \leftarrow \in T[i,j] & \text{if} & D[i,j-1] + \delta(-, v_j) \end{array}$$

Runtime: $O(mn)$

Alignment with variable Gap-Costs

```
ACCGTCTGCT  ACCGTCTGCT
A-C--C-G-T  ACCGT-----
```

$w(\mathscr{A}) = 5$

This contradicts “biological intuition”:
Insertion of gap of length k is “evolutionary simpler to realize” then insertion of k gaps of length 1.

gap penalty function $g : \mathbb{N} \to \mathbb{R}$
$g(k)$ is penalty for inserting a gap of length k.
we need:

$$g(k+l) \leq g(k) + g(l),$$

as otherwise it might be better to insert 2 gaps of length k and l then one gap of length $k+l$.

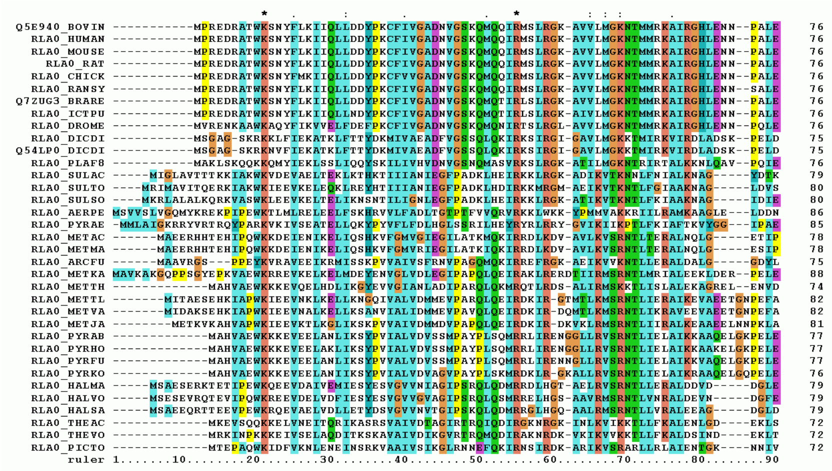
Alignment with variable Gap-Costs (Smith-Waterman-Alg.)

Init: $D[0,0] = 0$; $D[0,k] = D[k,0] = g(k)$, $k \geq 1$;

$$D[i,j] = \min \begin{cases} D[i-1,j-1] & + \quad \delta(u_i, v_j) \\ \min_{1 \leq k \leq i} D[i-k,j] & + \quad g(k) \\ \min_{1 \leq k \leq j} D[i,j-k] & + \quad g(k) \end{cases}$$

Tracematrix is an $m \times n$ matrix with $T[i,j] \subseteq \{\leftarrow_k, \nwarrow, \uparrow_k, k \in \mathbb{N}\}$

Example Multiple Alignment



```
                                 *   .      :       .          .   *         : : :      .
Q5E940_BOVIN -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE     76
  RLA0_HUMAN -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE     76
  RLA0_MOUSE -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE     76
    RLA0_RAT -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE     76
  RLA0_CHICK -----------MPREDRATWKSNYFMKIIQLLDDYPKCFVVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE     76
  RLA0_RANSY -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--SALE     76
Q7ZUG3_BRARE -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQTIRLSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE     76
  RLA0_ICTPU -----------MPREDRATWKSNYFLKIIQLLNDYPKCFIVGADNVGSKQMQTIRLSLRGK-AIVLMGKNTMMRKAIRGHLENN--PALE     76
  RLA0_DROME -----------MVRENKAAWKAQYFIKVVELFDEFPKCFIVGADNVGSKQMQNIRTSLRGL-AVVLMGKNTMMRKAIRGHLENN--PQLE     76
  RLA0_DICDI -----------MSGAG-SKRKKLFIEKATKLFTTYDKMIVAEADFVGSSQLQKIRKSIRGI-GAVLMGKKTMIRKVIRDLADSK--PELD     75
Q54LP0_DICDI -----------MSGAG-SKRKNVFIEKATKLFTTYDKMIVAEADFVGSSQLQKIRKSIRGI-GAVLMGKKTMIRKVIRDLADSK--PELD     75
  RLA0_PLAF8 -----------MAKLSKQQKKQMYIEKLSSLIQQYSKILIVHVDNVGSNQMASVRKSLRGK-ATILMGKNTRIRTALKKNLQAV--PQIE     76
  RLA0_SULAC -----MIGLAVTTTKKIAKWKVDEVAELTEKLKTHKTIIIANIEGFPADKLHEIRKKLRGK-ADIKVTKNNLFNIALKNAG-----YDTK     79
  RLA0_SULTO ----MRIMAVITQERKIAKWKIEEVKELEQKLREYHTIIIANIEGFPADKLHDIRKKMRGM-AEIKVTKNTLFGIAAKNAG-----LDVS     80
  RLA0_SULSO ----MKRLALALKQRKVASWKLEEVKELTELIKNSNTILIGNLEGFPADKLHEIRKKLRGK-ATIKVTKNTLFKIAAKNAG-----IDIE     80
  RLA0_AERPE MSVVSLVGQMYKREKPIPEWKTLMLRELEELFSKHRVVLFADLTGTPTFVVQRVRKKLWKK-YPMMVAKKRIILRAMKAAGLE---LDDN     86
  RLA0_PYRAE -MMLAIGKRRYVRTRQYPARKVKIVSEATELLQKYPYVFLFDLHGLSSRILHEYRYRLRRY-GVIKIIKPTLFKIAFTKVYGG---IPAE     85
  RLA0_METAC ------MAEERHHTEHIPQWKKDEIENIKELIQSHKVFGMVGIEGILATKMQKIRRDLKDV-AVLKVSRNTLTERALNQLG-----ETIP     78
  RLA0_METMA ------MAEERHHTEHIPQWKKDEIENIKELIQSHKVFGMVRIEGILATKIQKIRRDLKDV-AVLKVSRNTLTERALNQLG-----ESIP     78
  RLA0_ARCFU ------MAAVRGS---PPEYKVRAVEEIKRMISSKPVVAIVSFRNVPAGQMQKIRREFRGK-AEIKVVKNTLLERALDALG-----GDYL     75
  RLA0_METKA MAVKAKGQPPSGYEPKVAEWKRREVKELKELMDEYENVGLVDLEGIPAPQLQEIRAKLRERDTIIRMSRNTLMRIALEEKLDER--PELE     88
  RLA0_METTH -------------MAHVAEWKKKEVQELHDLIKGYEVVGIANLADIPARQLQKMRQTLRDS-ALIRMSKKTLISLALEKAGREL--ENVD     74
  RLA0_METTL -------MITAESEHKIAPWKIEEVNKLKELLKNGQIVALVDMMEVPARQLQEIRDKIR-GTMTLKMSRNTLIERAIKEVAEETGNPEFA     82
  RLA0_METVA -------MIDAKSEHKIAPWKIEEVNALKELLKSANVIALIDMMEVPAVQLQEIRDKIR-DQMTLKMSRNTLIKRAVEEVAEETGNPEFA     82
  RLA0_METJA --------METKVKAHVAPWKIEEVKTLKGLIKSKPVVAIVDMMDVPAPQLQEIRDKIR-DKVKLRMSRNTLIIRALKEAAEELNNPKLA     81
  RLA0_PYRAB -------------MAHVAEWKKKEVEELANLIKSYPVIALVDVSSMPAYPLSQMRRLIRENGGLLRVSRNTLIELAIKKAAQELGKPELE     77
  RLA0_PYRHO -------------MAHVAEWKKKEVEELAKLIKSYPVIALVDVSSMPAYPLSQMRRLIRENGGLLRVSRNTLIELAIKKAAKELGKPELE     77
  RLA0_PYRFU -------------MAHVAEWKKKEVEELANLIKSYPVVALVDVSSMPAYPLSQMRRLIRENNGLLRVSRNTLIELAIKKVAQELGKPELE     77
  RLA0_PYRKO -------------MAHVAEWKKKEVEELANIIKSYPVIALVDVAGVPAYPLSKMRDKLR-GKALLRVSRNTLIELAIKRAAQELGQPELE     76
  RLA0_HALMA -----MSAESERKTETIPEWKQEEVDAIVEMIESYESVGVVNIAGIPSRQLQDMRRDLHGT-AELRVSRNTLLERALDDVD-----DGLE     79
  RLA0_HALVO -----MSESEVRQTEVIPQWKREEVDELVDFIESYESVGVVGVAGIPSRQLQSMRRELHGS-AAVRMSRNTLVNRALDEVN-----DGFE     79
  RLA0_HALSA -----MSAEEQRTTEEVPEWKRQEVAELVDLLETYDSVGVVNVTGIPSKQLQDMRRGLHGQ-AALRMSRNTLLVRALEEAG-----DGLD     79
  RLA0_THEAC -------------MKEVSQQKKELVNEITQRIKASRSVAIVDTAGIRTRQIQDIRGKNRGK-INLKVIKKTLLFKALENLGD----EKLS     72
  RLA0_THEVO -------------MRKINPKKKEIVSELAQDITKSKAVAIVDIKGVRTRQMQDIRAKNRDK-VKIKVVKKTLLFKALDSIND----EKLT     72
  RLA0_PICTO -------------MTEPAQWKIDFVKNLENEINSRKVAAIVSIKGLRNNEFQKIRNSIRDK-ARIKVSRARLLRLAIENTGK----NNIV     72
       ruler 1.......10........20........30........40........50........60........70........80........90
```

First 90 positions of a protein multiple sequence alignment of instances of the acidic ribosomal protein P0 (L10E) from several organisms. (wikipedia)

Distance VS Scoring Function

Note: Instead of using a distance matrix D we can use a Similarity/Scoring Matrix S and maximize.

Init: $S[i,0] = -i * gap - cost$; $S[0,j] = -j * gap - cost$; for $i,j \geq 0$;

Compute

$$S[i,j] = \max \begin{cases} S[i-1,j] & + & \delta(u_i,-) \\ S[i-1,j-1] & + & \delta(u_i,v_j) \\ S[i,j-1] & + & \delta(-,v_j) \end{cases}$$

with e.g.

$$\delta(a,b) = \begin{cases} 1 & \text{if } a = b \\ -1 & \text{if } a \neq b \text{ and } a,b \neq - \\ -3 & \text{else (gap-costs)} \end{cases}$$

Local vs Global Alignment

Needleman-Wunsch computes a *global optimal Alignment*

NW reasonable if sequences have almost same length

If sequences have quite different length, then the sequences are "shredded":

```
R--------LCPMNLCGCSQ-----------------------KY
RCGEQGSNMECPNNLC-CSQYGYCGMGGDYCGKGCQNGACWTSKR
```

Reason: gaps are penalized equally on each position
Reasonable: less penalization of gaps at end and beginning

Local Alignment: find best alignment of two substrings of two sequences (Smith-Waterman-Algorithm)

Smith-Waterman-Algorithm

Need scoring function that penalizes insertion/deletions with a negative value

Compute

$$S[i,j] = \max \begin{cases} 0 & & \\ S[i-1,j] & + & \delta(u_i,-) \\ S[i-1,j-1] & + & \delta(u_i,v_j) \\ S[i,j-1] & + & \delta(-,v_j) \end{cases}$$

with init: $S[i,0] = S[0,j] = 0$; for $i,j \geq 0$;

First row states: we can start on each point a new alignment, if the score of the alignment computed so-far has a negative weight.

here maximize score. either minimize distance or maximize score

Standart Tool BLAST

BLAST = **B**asic **L**ocal **A**lignment **S**earch **T**ool

- quick heuristic alignment algorithm
- divides query sequences into short strings and initially only looks for (exact) matches of those strings in database strings. This is afterwards extended to get the entire alignment.
- much faster, but no optimality guarantee

Databases e.g. for nucleotide sequences (Genbank of NCBI, EMBL, ...) or protein databases (SwissProt, RefSeq, Pfam, ...).

BLAST homepage: `blast.ncbi.nlm.nih.gov/`
Tutorial: `digitalworldbiology.com/BLAST`

BLAST“Types”

type	query	target
blastn	nucleotide	nucleotide
blastp	protein	protein
blastx	nucleotide (transl)	protein
tblastn	protein	nucleotide (transl)
tblastx	nucleotide (transl)	nucleotide (transl)