Bioinformatics I

Lecture 07:

Multiple Sequence Alignment II

Dr. Sebastian Will

Bioinformatics Group, University Freiburg

Summer 2012

Multiple Alignment

Example: *Sequences*

$$a^{(1)} = \texttt{ACCCGAG}$$
$$a^{(2)} = \texttt{ACTACC}$$
$$a^{(3)} = \texttt{TCCTACGG}$$

$$\Rightarrow_{\text{align}} \quad A = \begin{matrix}\texttt{ACCCGA-G-}\\ \texttt{AC--TAC-C}\\ \texttt{TCC-TACGG}\end{matrix}$$

Alignment

Definition

A *multiple alignment A of K sequences* $a^{(1)}...a^{(K)}$ is a $K \times N$-matrix $(A_{i,j})_{\substack{1\le i\le K\\ 1\le j\le N}}$ (N is the *number of columns of A*)

where

1. each entry $A_{i,j} \in (\Sigma \cup \{-\})$
2. for each row i: deleting all gaps from $(A_{i,1}...A_{i,N})$ yields $a^{(i)}$
3. no column j contains only gap symbols

How to Score Multiple Alignments

As for pairwise alignment:

- Assume columns are scored independently
- Score is sum over alignment columns

$$S(A) = \sum_{j=1}^{N} s(A_{1j}, \ldots, A_{Kj})$$

Example

$$S\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{AC--TAC-C} \\ \texttt{TCC-TACGG} \end{pmatrix} = s\begin{pmatrix} A \\ A \\ T \end{pmatrix} + s\begin{pmatrix} C \\ C \\ C \end{pmatrix} + s\begin{pmatrix} C \\ - \\ C \end{pmatrix} + s\begin{pmatrix} C \\ - \\ - \end{pmatrix} + \cdots + s\begin{pmatrix} - \\ C \\ G \end{pmatrix}$$

How can we know similarities?

See later in this class how to learn pairwise similarities. *Not enough data to learn similarities for triples, 4-tuples, . . .*

How to Score Multiple Alignments

As for pairwise alignment:

- Assume columns are scored independently
- Score is sum over alignment columns

$$S(A) = \sum_{j=1}^{N} s(A_{1j}, \ldots, A_{Kj})$$

Example

$$S\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{AC--TAC-C} \\ \texttt{TCC-TACGG} \end{pmatrix} = s\begin{pmatrix} A \\ A \\ T \end{pmatrix} + s\begin{pmatrix} C \\ C \\ C \end{pmatrix} + s\begin{pmatrix} C \\ - \\ C \end{pmatrix} + s\begin{pmatrix} C \\ - \\ - \end{pmatrix} + \cdots + s\begin{pmatrix} - \\ C \\ G \end{pmatrix}$$

How can we know similarities?

See later in this class how to learn pairwise similarities. *Not enough data to learn similarities for triples, 4-tuples, . . .*

How to Score Multiple Alignments

As for pairwise alignment:

- Assume columns are scored independently
- Score is sum over alignment columns

$$S(A) = \sum_{j=1}^{N} s(A_{1j}, \ldots, A_{Kj})$$

Example

$$S\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{AC--TAC-C} \\ \texttt{TCC-TACGG} \end{pmatrix} = s\begin{pmatrix} A \\ A \\ T \end{pmatrix} + s\begin{pmatrix} C \\ C \\ C \end{pmatrix} + s\begin{pmatrix} C \\ - \\ C \end{pmatrix} + s\begin{pmatrix} C \\ - \\ - \end{pmatrix} + \cdots + s\begin{pmatrix} - \\ C \\ G \end{pmatrix}$$

How can we know similarities?

See later in this class how to learn pairwise similarities. *Not enough data to learn similarities for triples, 4-tuples, . . .*

Sum-Of-Pairs Score

Idea: approximate column scores by pairwise scores

$$s\begin{pmatrix} x_1 \\ \dots \\ x_K \end{pmatrix} = \sum_{1 \le k < l \le K} s(x_k, x_l)$$

Sum-of-pairs is the most commonly used scoring scheme for multiple alignments.

Definition: The sum-of-pairs score of a multiple alignment $\mathcal{A}$ is the sum of the scores of all pairwise alignments induced by $\mathcal{A}$.

Example:

$$\text{sum-of-pairs-score}\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{AC--TAC-C} \\ \texttt{TCC-TACGG} \end{pmatrix}$$

$$= \text{sc}\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{AC--TAC-C} \end{pmatrix} + \text{sc}\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{TCC-TACGG} \end{pmatrix} + \text{sc}\begin{pmatrix} \texttt{AC-TAC-C} \\ \texttt{TCCTACGG} \end{pmatrix}$$

Drawbacks: No theoretical justification as log-odd scores. Overcounting.

Sum-Of-Pairs Score

Idea: approximate column scores by pairwise scores

$$s\begin{pmatrix} x_1 \\ \dots \\ x_K \end{pmatrix} = \sum_{1 \le k < l \le K} s(x_k, x_l)$$

Sum-of-pairs is the most commonly used scoring scheme for multiple alignments.

Definition: The sum-of-pairs score of a multiple alignment $\mathcal{A}$ is the sum of the scores of all pairwise alignments induced by $\mathcal{A}$.

Example:

$$\text{sum-of-pairs-score}\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{AC--TAC-C} \\ \texttt{TCC-TACGG} \end{pmatrix}$$

$$= \text{sc}\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{AC--TAC-C} \end{pmatrix} + \text{sc}\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{TCC-TACGG} \end{pmatrix} + \text{sc}\begin{pmatrix} \texttt{AC-TAC-C} \\ \texttt{TCCTACGG} \end{pmatrix}$$

Drawbacks: No theoretical justification as log-odd scores. Overcounting.

Sum-Of-Pairs Score

Idea: approximate column scores by pairwise scores

$$s\begin{pmatrix} x_1 \\ \dots \\ x_K \end{pmatrix} = \sum_{1 \le k < l \le K} s(x_k, x_l)$$

Sum-of-pairs is the most commonly used scoring scheme for multiple alignments.

Definition: The sum-of-pairs score of a multiple alignment $\mathcal{A}$ is the sum of the scores of all pairwise alignments induced by $\mathcal{A}$.

Example:

$$\text{sum-of-pairs-score}\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{AC--TAC-C} \\ \texttt{TCC-TACGG} \end{pmatrix}$$

$$= \text{sc}\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{AC--TAC-C} \end{pmatrix} + \text{sc}\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{TCC-TACGG} \end{pmatrix} + \text{sc}\begin{pmatrix} \texttt{AC-TAC-C} \\ \texttt{TCCTACGG} \end{pmatrix}$$

Drawbacks: No theoretical justification as log-odd scores. Overcounting.

Sum-Of-Pairs Score

Idea: approximate column scores by pairwise scores

$$s\begin{pmatrix} x_1 \\ \dots \\ x_K \end{pmatrix} = \sum_{1 \le k < l \le K} s(x_k, x_l)$$

Sum-of-pairs is the most commonly used scoring scheme for multiple alignments.

Definition: The sum-of-pairs score of a multiple alignment $\mathcal{A}$ is the sum of the scores of all pairwise alignments induced by $\mathcal{A}$.

Example:

$$\text{sum-of-pairs-score}\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{AC--TAC-C} \\ \texttt{TCC-TACGG} \end{pmatrix}$$

$$= \text{sc}\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{AC--TAC-C} \end{pmatrix} + \text{sc}\begin{pmatrix} \texttt{ACCCGA-G-} \\ \texttt{TCC-TACGG} \end{pmatrix} + \text{sc}\begin{pmatrix} \texttt{AC-TAC-C} \\ \texttt{TCCTACGG} \end{pmatrix}$$

Drawbacks: No theoretical justification as log-odd scores. Overcounting.

Optimal Multiple Alignment

For 3 sequences a, b, c, use 3-dimensional matrix

$D_{i,j,k}$:= minimal sum of pairwise distances in MSA of prefix sequences $a[1..i]$, $b[1..j]$, and $c[1..k]$

(after initialization)

$$D_{i,j,k} = \min \begin{cases} D_{i-1,j-1,k-1} & +d(a_i, b_j) + d(a_i, c_k) + d(b_j, c_k) \\ D_{i-1,j-1,k} & +d(a_i, b_j) + 2\gamma \\ D_{i-1,j,k-1} & +d(a_i, c_k) + 2\gamma \\ D_{i,j-1,k-1} & +d(b_j, c_k) + 2\gamma \\ D_{i-1,j,k} & +2\gamma \\ D_{i,j-1,k} & +2\gamma \\ D_{i,j,k-1} & +2\gamma \end{cases}$$

For K sequences use K-dimensional matrix. (complexity $O(n^K)$)

One can show: MSA is NP-complete.

Optimal Multiple Alignment

For 3 sequences a, b, c, use 3-dimensional matrix

$D_{i,j,k}$:= minimal sum of pairwise distances in MSA of prefix sequences $a[1..i]$, $b[1..j]$, and $c[1..k]$

(after initialization)

$$D_{i,j,k} = \min \begin{cases} D_{i-1,j-1,k-1} & +d(a_i, b_j) + d(a_i, c_k) + d(b_j, c_k) \\ D_{i-1,j-1,k} & +d(a_i, b_j) + 2\gamma \\ D_{i-1,j,k-1} & +d(a_i, c_k) + 2\gamma \\ D_{i,j-1,k-1} & +d(b_j, c_k) + 2\gamma \\ D_{i-1,j,k} & +2\gamma \\ D_{i,j-1,k} & +2\gamma \\ D_{i,j,k-1} & +2\gamma \end{cases}$$

For K sequences use K-dimensional matrix. (complexity $O(n^K)$)

One can show: MSA is NP-complete.

Optimal Multiple Alignment

For 3 sequences a, b, c, use 3-dimensional matrix

$D_{i,j,k}$:= minimal sum of pairwise distances in MSA of prefix sequences $a[1..i]$, $b[1..j]$, and $c[1..k]$

(after initialization)

$$D_{i,j,k} = \min \begin{cases} D_{i-1,j-1,k-1} & +d(a_i, b_j) + d(a_i, c_k) + d(b_j, c_k) \\ D_{i-1,j-1,k} & +d(a_i, b_j) + 2\gamma \\ D_{i-1,j,k-1} & +d(a_i, c_k) + 2\gamma \\ D_{i,j-1,k-1} & +d(b_j, c_k) + 2\gamma \\ D_{i-1,j,k} & +2\gamma \\ D_{i,j-1,k} & +2\gamma \\ D_{i,j,k-1} & +2\gamma \end{cases}$$

For K sequences use K-dimensional matrix. (complexity $O(n^K)$)

One can show: MSA is NP-complete.

Speedup for the Exact Solution: Carillo-Lipman

Idea: Don't fill entries in K-dimensional matrix that are provably not on optimal trace. (Here only $K = 3$).

Definition

Let $d_{a,b}(i, j)$ be edit distance for suffixes $a[i + 1..|a|]$ and $b[j + 1..|b|]$. Analogously define $d_{a,c}(i, k)$ and $d_{b,c}(j, k)$.

Speedup for the Exact Solution: Carillo-Lipman

Idea: Don't fill entries in K-dimensional matrix that are provably not on optimal trace. (Here only $K = 3$).

Definition

Let $d_{a,b}(i, j)$ be edit distance for suffixes $a[i + 1..|a|]$ and $b[j + 1..|b|]$. Analogously define $d_{a,c}(i, k)$ and $d_{b,c}(j, k)$.

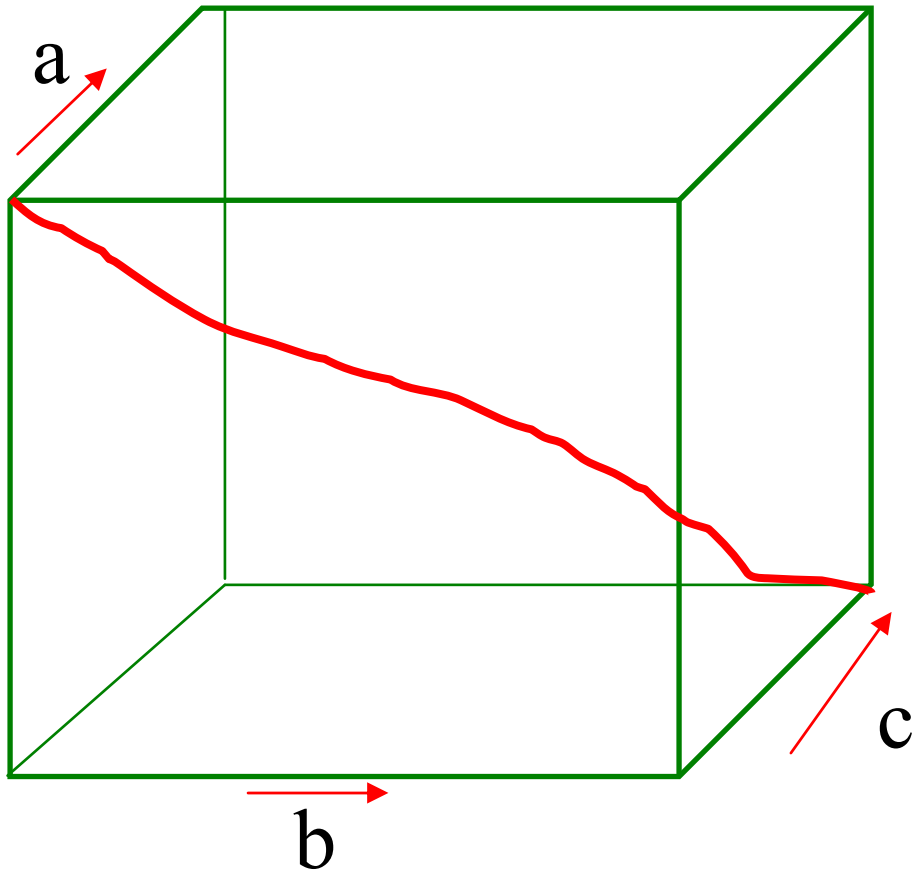
Remark: These *suffix distances* can be computed like prefix distances after reversing the sequences (cf. Hirschberg-algorithm).

Speedup for the Exact Solution: Carillo-Lipman

Idea: Don't fill entries in K-dimensional matrix that are provably not on optimal trace. (Here only $K = 3$).

Definition

Let $d_{a,b}(i,j)$ be edit distance for suffixes $a[i+1..|a|]$ and $b[j+1..|b|]$. Analogously define $d_{a,c}(i,k)$ and $d_{b,c}(j,k)$.

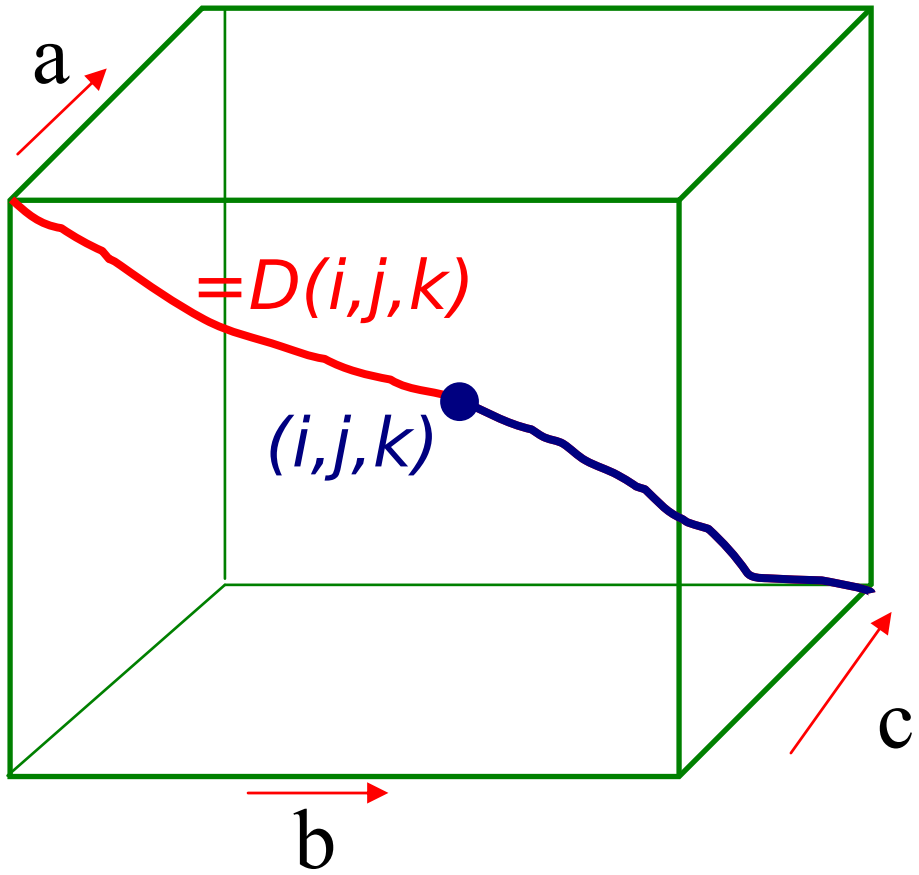


Speedup for the Exact Solution: Carillo-Lipman

Idea: Don't fill entries in K-dimensional matrix that are provably not on optimal trace. (Here only $K = 3$).

Definition

Let $d_{a,b}(i,j)$ be edit distance for suffixes $a[i+1..|a|]$ and $b[j+1..|b|]$. Analogously define $d_{a,c}(i,k)$ and $d_{b,c}(j,k)$.

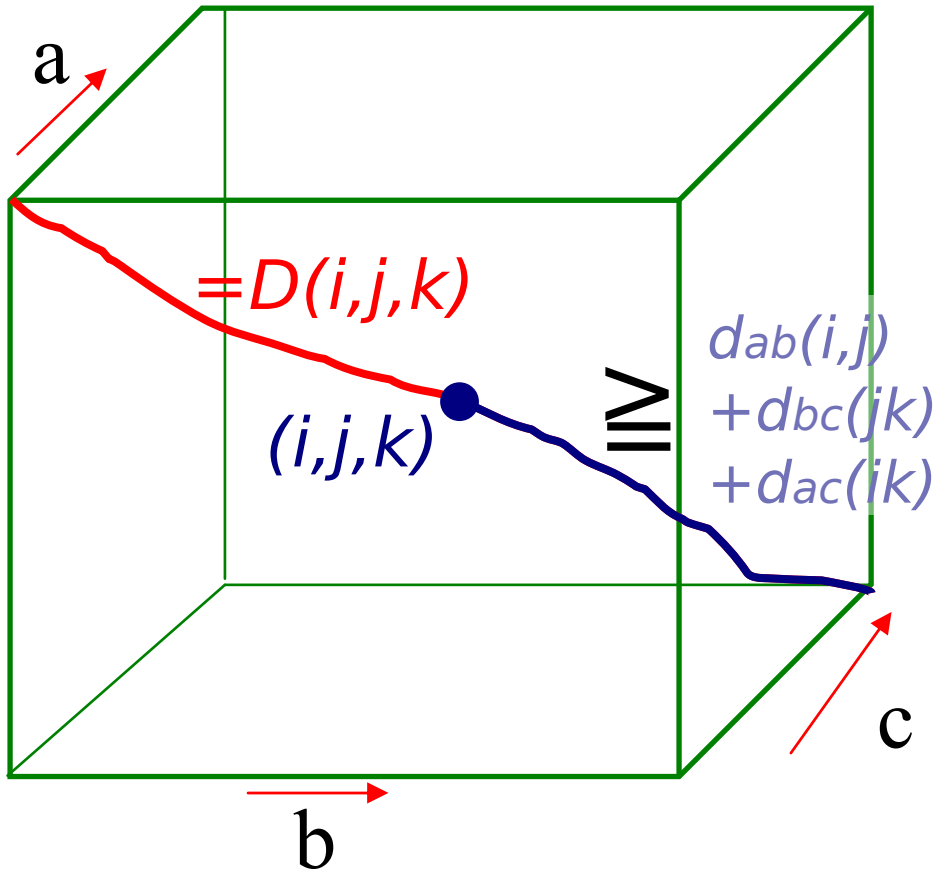


Speedup for the Exact Solution: Carillo-Lipman

Idea: Don't fill entries in K-dimensional matrix that are provably not on optimal trace. (Here only $K = 3$).

Definition

Let $d_{a,b}(i,j)$ be edit distance for suffixes $a[i+1..|a|]$ and $b[j+1..|b|]$. Analogously define $d_{a,c}(i,k)$ and $d_{b,c}(j,k)$.



Speedup for the Exact Solution: Carillo-Lipman

Key idea

Guess score z of some "good" alignment of a, b, and c.
If $D_{i,j,k} + d_{a,b}(i,j) + d_{a,c}(i,k) + d_{b,c}(j,k) \geq z$, then $D_{i,j,k}$ does not contribute to optimum.

This saves time in *forward dynamic programming* (vs. backward DP, which we used before). If many results are not sent forward, many entries are never computed ($\rightarrow$ speedup)

Implemented in tool MSA. Up to 6 sequences, $n = 200$.

Speedup for the Exact Solution: Carillo-Lipman

Key idea

Guess score z of some "good" alignment of a, b, and c.
If $D_{i,j,k} + d_{a,b}(i,j) + d_{a,c}(i,k) + d_{b,c}(j,k) \geq z$, then $D_{i,j,k}$ does not contribute to optimum.

This saves time in *forward dynamic programming* (vs. backward DP, which we used before). If many results are not sent forward, many entries are never computed ($\rightarrow$ speedup)

Implemented in tool MSA. Up to 6 sequences, $n = 200$.

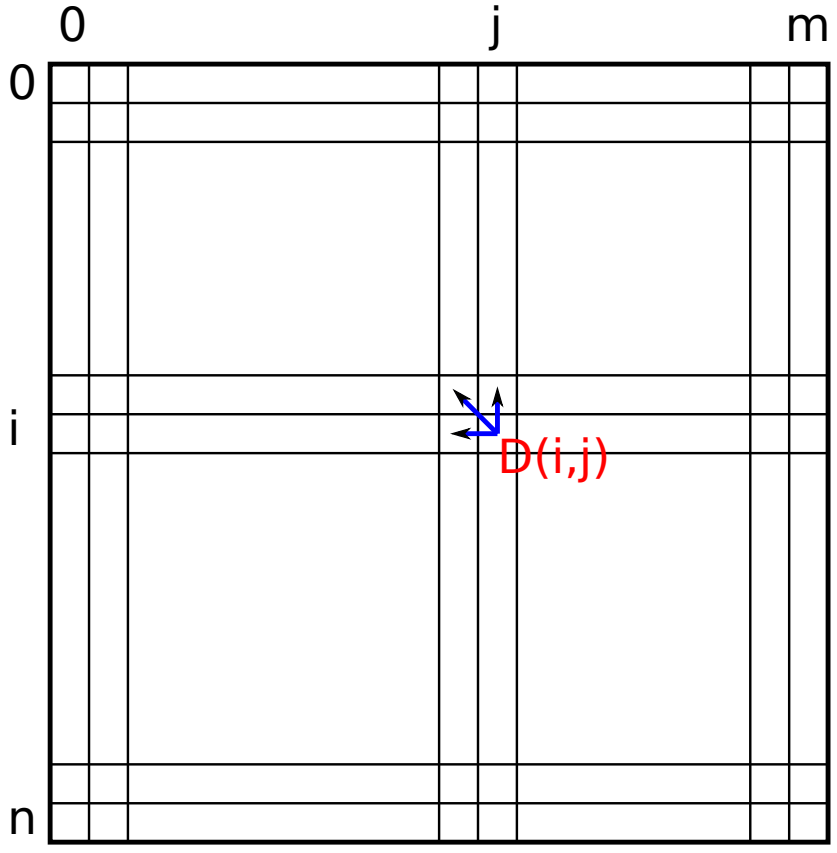
Forward Dynamic Programming

$$\text{Case K=2: } D_{i,j} = \min \begin{cases} D_{i-1,j-1} & + & d(a_i, b_j) \\ D_{i-1,j} & + & \gamma \\ D_{i,j-1} & + & \gamma \end{cases}$$

Forward Dynamic Programming

Case K=2: $D_{i,j} = \min \begin{cases} D_{i-1,j-1} & + & d(a_i, b_j) \\ D_{i-1,j} & + & \gamma \\ D_{i,j-1} & + & \gamma \end{cases}$

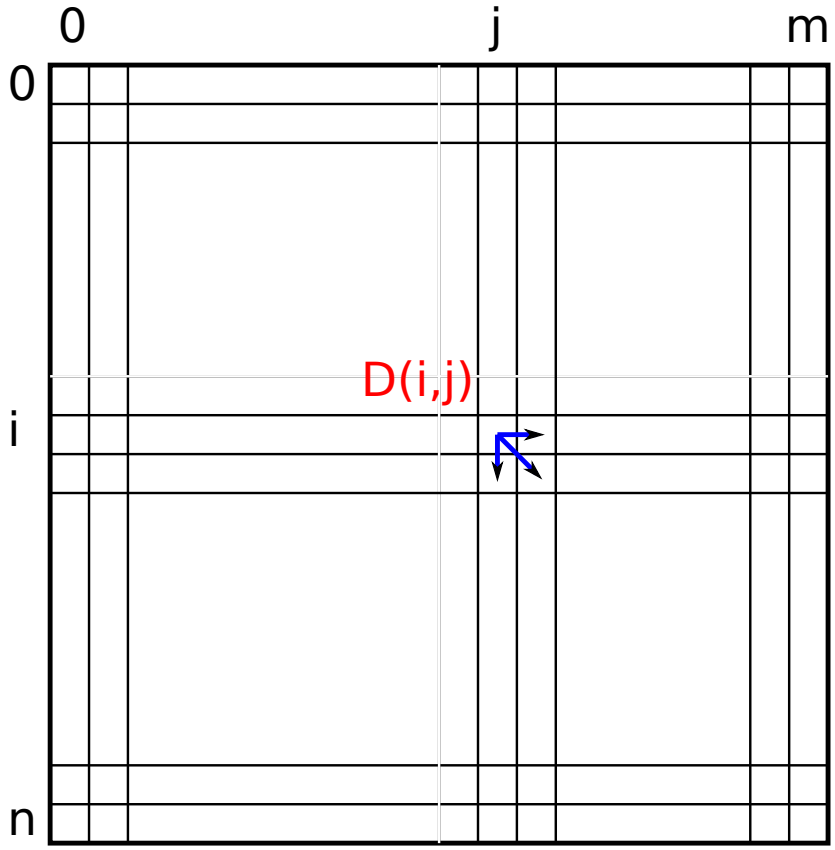
"Backward" DP: (classical evaluation)



Forward Dynamic Programming

Case K=2: $D_{i,j} = \min \begin{cases} D_{i-1,j-1} & + & d(a_i, b_j) \\ D_{i-1,j} & + & \gamma \\ D_{i,j-1} & + & \gamma \end{cases}$

"Forward DP: Send scores in DP entries forward



Forward Dynamic Programming: Pseudocode

Pseudocode $K = 3$

Init queue Q :=empty
Push $Q, (0,0,0) \mapsto 0$
While Q not empty
  $(i,j,k) \mapsto s$:= top(Q)
  send $s + 2\gamma$ to $(i,j,k+1)$
  send $s + 2\gamma$ to $(i,j+1,k)$
  send $s + 2\gamma$ to $(i+1,j,k)$
  $\vdots$
  send $s + d(a_{i+1}, b_{j+1}, c_{k+1})$ to $(i+1,j+1,k+1)$

Procedure send s to (i,j,k):
If $(i,j,k) \mapsto s'$ in Q:
  update entry to $(i,j,k) \mapsto min(s,s')$
Otherwise:
  Push $Q, (i,j,k) \mapsto s$

Forward Dynamic Programming: Pseudocode

Pseudocode $K = 3$

```
Init queue Q :=empty
Push Q, (0,0,0) ↦ 0
While Q not empty
  (i,j,k) ↦ s := top(Q)
  send s + 2γ to (i,j,k+1)
  send s + 2γ to (i,j+1,k)
  send s + 2γ to (i+1,j,k)
  ⋮
  send s + d(a_{i+1}, b_{j+1}, c_{k+1}) to (i+1,j+1,k+1)

Procedure send s to (i,j,k):
If (i,j,k) ↦ s' in Q:
  update entry to (i,j,k) ↦ min(s,s')
Otherwise:
  Push Q, (i,j,k) ↦ s
```

Forward DP Allows Lazy Evaluation

Recall: in Carillo-Lipman, we send forward only from entries that pass the criterion

$$D_{i,j,k} + d_{a,b}(i,j) + d_{a,c}(i,k) + d_{b,c}(j,k) < z$$

Only entries that received any input are processed!

In Carillo-Lipman, this saves the computation of many matrix entries.

Recall: MSA aligns 6 sequences, $n = 200$.
Otherwise very expensive: $200^6 = 6.4 \times 10^{13}$ entries

Most Common Heuristics: Iterative Alignment

IDEA: Iteratively merge alignments of subsets of sequences into multiple alignment
Methods differ in

- selection of subsets, i.e. order of merges
- merging procedure

Example: simplest iterative alignments strategy

1. align two sequences with minimum edit distance
2. merge in sequence with minimum edit distance among remaining sequences
3. repeat (until all sequences are aligned)

More sophisticated: *progressive alignment*,
e.g., Feng-Doolittle and CLUSTAL variants

Aligning Alignments in Progressive Alignment

Two (multiple) alignments A and B can be aligned by DP in the same way as two sequences.

Idea:

- An alignment is a sequence of alignment columns.

Example:

```
ACCCGA-G-
AC--TAC-C
TCC-TACGG
```

$$\equiv \binom{A}{A}\binom{C}{C}\binom{C}{-}\binom{C}{-}\dots\binom{-}{G}.$$

- Assign similarity to two columns from resp. A and B, e.g. $s(\binom{-}{C}, \binom{G}{C})$ by *sum-of-pairs*.

We can use dynamic programming, which recurses over alignment scores of prefixes of alignments.

Consequences for progressive alignment scheme:

- Optimization only *local*.
- Commits to local decisions. "Once a gap, always a gap"

Progressive Alignment — Example

IN: $a^{(1)} = ACCG, a^{(2)} = TTGG, a^{(3)} = TCG, a^{(4)} = CTGG$

$$w(x,y) = \begin{cases} 0 \text{ iff } x = y \\ 2 \text{ iff } x = - \text{ or } y = - \\ 3 \text{ otherwise (for mismatch)} \end{cases}$$

- Compute all against all edit distances and cluster

Align ACCG and TTGG

		T	T	G	G
	0	2	4	6	8
A	2	3	5	7	9
C	4	5	6	8	10
C	6	7	8	9	11
G	8	9	10	8	9

Align ACCG and TCG

		T	C	G
	0	2	4	6
A	2	3	5	7
C	4	5	3	6
C	6	7	5	6
G	8	9	8	5

Align ACCG and CTGG

		C	T	G	G
	0	2	4	6	8
A	2	3	5	7	9
C	4	2	5	8	10
C	6	4	5	8	11
G	8	7	7	5	8

Align TTGG and TCG

		T	C	G
	0	2	4	6
T	2	0	3	6
T	4	2	3	6
G	6	5	5	3
G	8	8	8	5

Align TTGG and CTGG

		C	T	G	G
	0	2	4	6	8
T	2	3	2	5	8
T	4	5	3	5	8
G	6	7	6	3	5
G	8	9	9	6	3

Align TCG and CTGG

		C	T	G	G
	0	2	4	6	8
T	2	3	2	5	8
C	4	2	5	5	8
G	6	5	5	5	5

Progressive Alignment — Example

IN: $a^{(1)} = ACCG, a^{(2)} = TTGG, a^{(3)} = TCG, a^{(4)} = CTGG$

$$w(x,y) = \begin{cases} 0 \text{ iff } x = y \\ 2 \text{ iff } x = - \text{ or } y = - \\ 3 \text{ otherwise (for mismatch)} \end{cases}$$

- Compute all against all edit distances and cluster

  $\Rightarrow$ distance matrix

	$a^{(1)}$	$a^{(2)}$	$a^{(3)}$	$a^{(4)}$
$a^{(1)}$	0	9	5	8
$a^{(2)}$		0	5	3
$a^{(3)}$			0	5
$a^{(4)}$				0

  $\Rightarrow$ Cluster (e.g. UPGMA)
  $a^{(2)}$ and $a^{(4)}$ are closest, Then: $a^{(1)}$ and $a^{(3)}$

Progressive Alignment — Example

IN: $a^{(1)} = ACCG, a^{(2)} = TTGG, a^{(3)} = TCG, a^{(4)} = CTGG$

$$w(x,y) = \begin{cases} 0 \text{ iff } x = y \\ 2 \text{ iff } x = - \text{ or } y = - \\ 3 \text{ otherwise (for mismatch)} \end{cases}$$

- Compute all against all edit distances and cluster
  $\Rightarrow$ guide tree $((a^{(2)}, a^{(4)}), (a^{(1)}, a^{(3)}))$
- Align $a^{(2)}$ and $a^{(4)}$: `TTGG` / `CTGG` , Align $a^{(1)}$ and $a^{(3)}$: `ACCG` / `-TCG`
- Align the alignments!

Align `TTGG` / `CTGG` and `ACCG` / `-TCG`

		A / -	C / T	C / C	G / G
	0	4	12	20	28
TC	8	10	...		
TT	16	⋮	⋱		
GG	24				
GG	32				

Progressive Alignment — Example

IN: $a^{(1)} = ACCG, a^{(2)} = TTGG, a^{(3)} = TCG, a^{(4)} = CTGG$

$$w(x,y) = \begin{cases} 0 \text{ iff } x = y \\ 2 \text{ iff } x = - \text{ or } y = - \\ 3 \text{ otherwise (for mismatch)} \end{cases}$$

- Compute all against all edit distances and cluster

  $\Rightarrow$ guide tree $((a^{(2)}, a^{(4)}), (a^{(1)}, a^{(3)}))$
- Align $a^{(2)}$ and $a^{(4)}$: `TTGG` / `CTGG` , Align $a^{(1)}$ and $a^{(3)}$: `ACCG` / `-TCG`
- Align the alignments!

Align `TTGG`/`CTGG` and `ACCG`/`-TCG`

		A / -	C / T	C / C	G / G
	0	4	12	20	28
TC	8	10	...		
TT	16	⋮	⋱		
GG	24				
GG	32				

- $w(TC, --) =$

  $w(T,-) + w(C,-) + w(T,-) + w(C,-) = 8$
- $w(--, A-) =$

  $w(-,A) + w(-,-) + w(-,A) + w(-,-) = 4$
- $w(TC, A-) =$

  $w(T,A) + w(C,A) + w(T,-) + w(C,-) = 10$
- $w(TC, CT) =$

  $w(T,C) + w(C,C) + w(T,T) + w(C,T) = 6$
- ...

Progressive Alignment — Example

IN: $a^{(1)} = ACCG, a^{(2)} = TTGG, a^{(3)} = TCG, a^{(4)} = CTGG$

$$w(x,y) = \begin{cases} 0 \text{ iff } x = y \\ 2 \text{ iff } x = - \text{ or } y = - \\ 3 \text{ otherwise (for mismatch)} \end{cases}$$

- Compute all against all edit distances and cluster
  $\Rightarrow$ guide tree $((a^{(2)}, a^{(4)}), (a^{(1)}, a^{(3)}))$
- Align $a^{(2)}$ and $a^{(4)}$: `TTGG` / `CTGG` , Align $a^{(1)}$ and $a^{(3)}$: `ACCG` / `-TCG`
- Align the alignments!

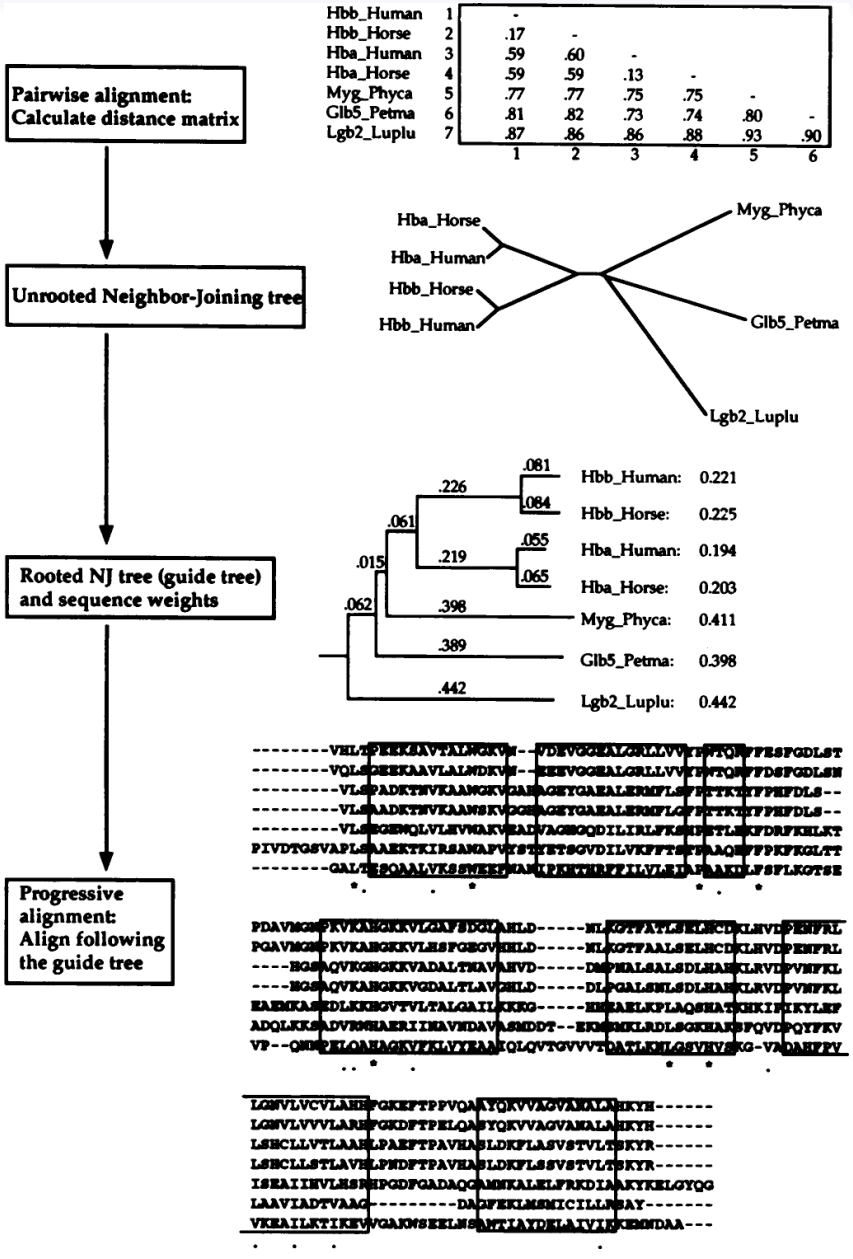
Align `TTGG` / `CTGG` and `ACCG` / `-TCG`

		A / -	C / T	C / C	G / G
	0	4	12	20	28
TC	8	10	...		
TT	16	⋮	⋱		
GG	24				
GG	32				

$\Longrightarrow$ after filling and traceback

```
TTGG
CTGG
ACCG
-TCG
```

A Classical Approach: CLUSTAL W



- prototypical progressive alignment
- similarity score with affine gap cost
- neighbor joining for tree construction
- sequence weighting
- 'tricks' for gap handling

Details: see last week's slides

Improving Progressive Alignment

- Limitations of progressive alignment:
  once a gap – always a gap: decisions are *local*
  (cf. *consistency*, Bioinformatics II)

- Iterative refinement:
  iteratively select one sequence or alignment of some sequences and realign to rest

- Performance bottleneck: guide tree construction requires $O(K^2)$ all-2-all comparisons
  (vs. $O(K)$ comparisons construct MSA)

  How to align hundreds or thousands of sequences?
  - use faster distance measure
  - construct draft alignment and reestimate distances

Improving Progressive Alignment

- Limitations of progressive alignment:
  once a gap – always a gap: decisions are *local*
  (cf. *consistency*, Bioinformatics II)

- Iterative refinement:
  iteratively select one sequence or alignment of some sequences and realign to rest

- Performance bottleneck: guide tree construction requires $O(K^2)$ all-2-all comparisons
  (vs. $O(K)$ comparisons construct MSA)

  How to align hundreds or thousands of sequences?

  - use faster distance measure
  - construct draft alignment and reestimate distances

Improving Progressive Alignment

- Limitations of progressive alignment:
  once a gap – always a gap: decisions are *local*
  (cf. *consistency*, Bioinformatics II)

- Iterative refinement:
  iteratively select one sequence or alignment of some sequences and realign to rest

- Performance bottleneck: guide tree construction requires $O(K^2)$ all-2-all comparisons
  (vs. $O(K)$ comparisons construct MSA)

  How to align hundreds or thousands of sequences?

  - use faster distance measure
  - construct draft alignment and reestimate distances

Improving Progressive Alignment

- Limitations of progressive alignment:
  once a gap – always a gap: decisions are *local*
  (cf. *consistency*, Bioinformatics II)

- Iterative refinement:
  iteratively select one sequence or alignment of some sequences and realign to rest

- Performance bottleneck: guide tree construction requires $O(K^2)$ all-2-all comparisons
  (vs. $O(K)$ comparisons construct MSA)

  How to align hundreds or thousands of sequences?

  - use faster distance measure
  - construct draft alignment and reestimate distances

Improving Progressive Alignment

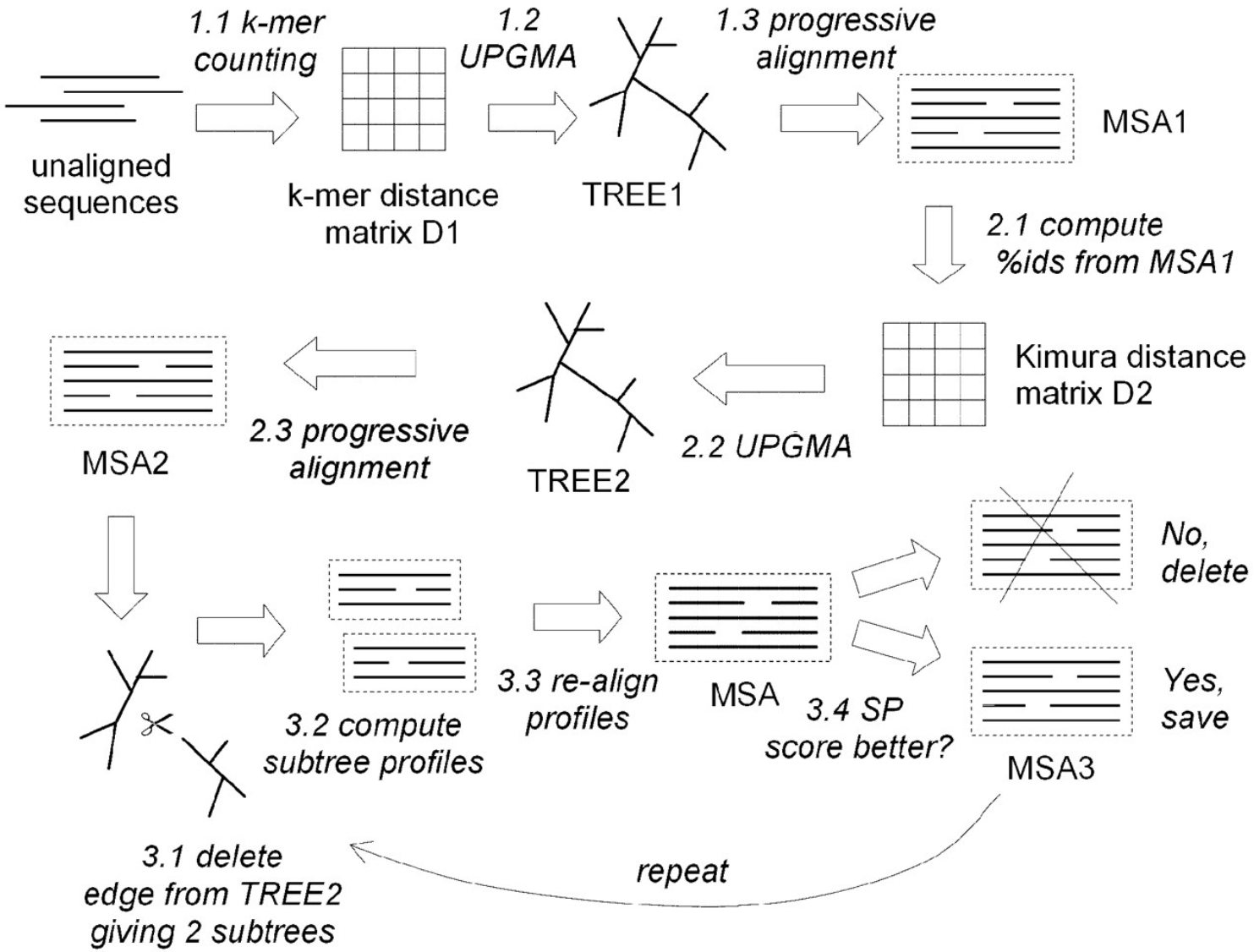
- Limitations of progressive alignment:
  once a gap – always a gap: decisions are *local*
  (cf. *consistency*, Bioinformatics II)

- Iterative refinement:
  iteratively select one sequence or alignment of some sequences and realign to rest

- Performance bottleneck: guide tree construction requires $O(K^2)$ all-2-all comparisons
  (vs. $O(K)$ comparisons construct MSA)

  How to align hundreds or thousands of sequences?
  - use faster distance measure
  - construct draft alignment and reestimate distances

Advanced Progressive Alignment in MUSCLE



1.) alignment draft and 2.) reestimation 3.) iterative refinement

How to compare MSA tools?

- Criteria
  - speed
  - degree of optimization (of SP-score) by heuristics
  - quality of computed alignments (in terms of true alignments)
- Benchmark
  - measuring alignment quality requires comparison to reference
  - benchmark instances: sets of protein sequences with hand-curated alignments
  - measure average alignment quality across instances
- Example: BAliBASE [Thompson et al., Bioinformatics. 1999]
  - reference alignments: 3D structural superpositions

How to compare MSA tools?

- Criteria
  - speed
  - degree of optimization (of SP-score) by heuristics
  - quality of computed alignments (in terms of true alignments)
- Benchmark
  - measuring alignment quality requires comparison to reference
  - benchmark instances: sets of protein sequences with hand-curated alignments
  - measure average alignment quality across instances
- Example: BAliBASE [Thompson et al., Bioinformatics. 1999]
  - reference alignments: 3D structural superpositions

How to compare MSA tools?

- Criteria
  - speed
  - degree of optimization (of SP-score) by heuristics
  - quality of computed alignments (in terms of true alignments)
- Benchmark
  - measuring alignment quality requires comparison to reference
  - benchmark instances: sets of protein sequences with hand-curated alignments
  - measure average alignment quality across instances
- Example: BAliBASE [Thompson et al., Bioinformatics. 1999]
  - reference alignments: 3D structural superpositions

Comparison of MSA tools

BAliBASE 3.0: 6255 sequences

Defines core regions of reliably aligned residues

Measure accuracy of computed (K-way) alignment A by comparison to reference alignment R (of same sequences):

$SPS(A|R)$ = average over pairwise alignments of rows k and l:

$$\frac{\text{valid columns in rows } k \text{ and } l \text{ of } A \text{ that identically occur in } R}{\text{total valid columns in rows } k \text{ and } l \text{ of } A}$$

(*valid*: ≥ 1 non-gap; *identical:* same sequence position)

For example:

$$A = \begin{pmatrix} \texttt{UVW-XZ} \\ \texttt{UWX--Z} \\ \texttt{UV-XYZ} \end{pmatrix} \text{ vs. } R = \begin{pmatrix} \texttt{UVWX-Z} \\ \texttt{U-W-XZ} \\ \texttt{UV-XYZ} \end{pmatrix}$$

Consider $k = 1$, $l = 2$: total=5, identical=3, ratio = 0.6

Comparison of MSA tools

BAliBASE 3.0: 6255 sequences

Defines core regions of reliably aligned residues

Measure accuracy of computed (K-way) alignment A by comparison to reference alignment R (of same sequences):

$SPS(A|R)$ = average over pairwise alignments of rows k and l:

$$\frac{\text{valid columns in rows } k \text{ and } l \text{ of } A \text{ that identically occur in } R}{\text{total valid columns in rows } k \text{ and } l \text{ of } A}$$

(*valid*: ≥ 1 non-gap; *identical:* same sequence position)

For example:

$$A = \begin{pmatrix} \texttt{UVW-XZ} \\ \texttt{UWX--Z} \\ \texttt{UV-XYZ} \end{pmatrix} \text{ vs. } R = \begin{pmatrix} \texttt{UVWX-Z} \\ \texttt{U-W-XZ} \\ \texttt{UV-XYZ} \end{pmatrix}$$

Consider $k = 1$, $l = 2$: total=5, identical=3, ratio = 0.6

Comparison of MSA tools

BAliBASE 3.0: 6255 sequences

Defines core regions of reliably aligned residues

Measure accuracy of computed (K-way) alignment A by comparison to reference alignment R (of same sequences):

$SPS(A|R)$ = average over pairwise alignments of rows k and l:

$$\frac{\text{valid columns in rows } k \text{ and } l \text{ of } A \text{ that identically occur in } R}{\text{total valid columns in rows } k \text{ and } l \text{ of } A}$$

(*valid*: ≥ 1 non-gap; *identical:* same sequence position)

For example:

$$A = \begin{pmatrix} \texttt{UVW-XZ} \\ \texttt{UWX--Z} \\ \texttt{UV-XYZ} \end{pmatrix} \text{ vs. } R = \begin{pmatrix} \texttt{UVWX-Z} \\ \texttt{U-W-XZ} \\ \texttt{UV-XYZ} \end{pmatrix}$$

Consider $k = 1$, $l = 2$: total=5, identical=3, ratio = 0.6

Comparison of MSA tools

BAliBASE 3.0: 6255 sequences

Defines core regions of reliably aligned residues

Measure accuracy of computed (K-way) alignment A by comparison to reference alignment R (of same sequences):

$SPS(A|R)$ = average over pairwise alignments of rows k and l:

$$\frac{\text{valid columns in rows } k \text{ and } l \text{ of } A \text{ that identically occur in } R}{\text{total valid columns in rows } k \text{ and } l \text{ of } A}$$

(*valid*: ≥ 1 non-gap; *identical:* same sequence position)

For example:

$$A = \begin{pmatrix} \texttt{UVW-XZ} \\ \texttt{UWX--Z} \\ \texttt{UV-XYZ} \end{pmatrix} \text{ vs. } R = \begin{pmatrix} \texttt{UVWX-Z} \\ \texttt{U-W-XZ} \\ \texttt{UV-XYZ} \end{pmatrix}$$

Consider $k = 1$, $l = 2$: total=5, identical=3, ratio = 0.6

Comparison of MSA tools

BAliBASE 3.0: 6255 sequences

Defines core regions of reliably aligned residues

Measure accuracy of computed (K-way) alignment A by comparison to reference alignment R (of same sequences):

$SPS(A|R)$ = average over pairwise alignments of rows k and l:

$$\frac{\text{valid columns in rows } k \text{ and } l \text{ of } A \text{ that identically occur in } R}{\text{total valid columns in rows } k \text{ and } l \text{ of } A}$$

(*valid*: ≥ 1 non-gap; *identical:* same sequence position)

For example:

$$A = \begin{pmatrix} \texttt{UVW-XZ} \\ \texttt{UWX--Z} \\ \texttt{UV-XYZ} \end{pmatrix} \text{ vs. } R = \begin{pmatrix} \texttt{UVWX-Z} \\ \texttt{U-W-XZ} \\ \texttt{UV-XYZ} \end{pmatrix}$$

Consider $k = 1$, $l = 2$: total=5, identical=3, ratio = 0.6

ratio for $k = 2$,$l = 3$: total=6, identical= , ratio=

Comparison of MSA tools

some results from [Edgar, MUSCLE 2004]

	BAliBASE score	BAliBASE time (s)
MUSCLE	0.896	81
MAFFT	0.844	16
MUSCLE-fast	0.849	11
CLUSTALW	0.860	160

Conclusion: Multiple Sequence Alignment

- Uses of Multiple Sequence Alignments (MSA)
- Sum-Of-Pairs Scoring (SP)
- Methods for solving the MSA problem: exact/heuristic
- Heuristics: no guarantee, but effective in practice
- Iterative/Progressive Alignment
- ClustalW: classic progressive alignment
  - distance matrix
  - guide tree
  - optimized gap penalties
- MUSCLE: progressive alignment with various improvements
  - guide tree re-estimation
  - iterative refinement
- Benchmarks (BAliBASE)

Conclusion: Multiple Sequence Alignment

- Uses of Multiple Sequence Alignments (MSA)
- Sum-Of-Pairs Scoring (SP)
- Methods for solving the MSA problem: exact/heuristic
- Heuristics: no guarantee, but effective in practice
- Iterative/Progressive Alignment
- ClustalW: classic progressive alignment
  - distance matrix
  - guide tree
  - optimized gap penalties
- MUSCLE: progressive alignment with various improvements
  - guide tree re-estimation
  - iterative refinement
- Benchmarks (BAliBASE)

Conclusion: Multiple Sequence Alignment

- Uses of Multiple Sequence Alignments (MSA)
- Sum-Of-Pairs Scoring (SP)
- Methods for solving the MSA problem: exact/heuristic
- Heuristics: no guarantee, but effective in practice
- Iterative/Progressive Alignment
- ClustalW: classic progressive alignment
  - distance matrix
  - guide tree
  - optimized gap penalties
- MUSCLE: progressive alignment with various improvements
  - guide tree re-estimation
  - iterative refinement
- Benchmarks (BAliBASE)

Conclusion: Multiple Sequence Alignment

- Uses of Multiple Sequence Alignments (MSA)
- Sum-Of-Pairs Scoring (SP)
- Methods for solving the MSA problem: exact/heuristic
- Heuristics: no guarantee, but effective in practice
- Iterative/Progressive Alignment
- ClustalW: classic progressive alignment
  - distance matrix
  - guide tree
  - optimized gap penalties
- MUSCLE: progressive alignment with various improvements
  - guide tree re-estimation
  - iterative refinement
- Benchmarks (BAliBASE)

Conclusion: Multiple Sequence Alignment

- Uses of Multiple Sequence Alignments (MSA)
- Sum-Of-Pairs Scoring (SP)
- Methods for solving the MSA problem: exact/heuristic
- Heuristics: no guarantee, but effective in practice
- Iterative/Progressive Alignment
- ClustalW: classic progressive alignment
  - distance matrix
  - guide tree
  - optimized gap penalties
- MUSCLE: progressive alignment with various improvements
  - guide tree re-estimation
  - iterative refinement
- Benchmarks (BAliBASE)

Conclusion: Multiple Sequence Alignment

- Uses of Multiple Sequence Alignments (MSA)
- Sum-Of-Pairs Scoring (SP)
- Methods for solving the MSA problem: exact/heuristic
- Heuristics: no guarantee, but effective in practice
- Iterative/Progressive Alignment
- ClustalW: classic progressive alignment
  - distance matrix
  - guide tree
  - optimized gap penalties
- MUSCLE: progressive alignment with various improvements
  - guide tree re-estimation
  - iterative refinement
- Benchmarks (BAliBASE)

Conclusion: Multiple Sequence Alignment

- Uses of Multiple Sequence Alignments (MSA)
- Sum-Of-Pairs Scoring (SP)
- Methods for solving the MSA problem: exact/heuristic
- Heuristics: no guarantee, but effective in practice
- Iterative/Progressive Alignment
- ClustalW: classic progressive alignment
  - distance matrix
  - guide tree
  - optimized gap penalties
- MUSCLE: progressive alignment with various improvements
  - guide tree re-estimation
  - iterative refinement
- Benchmarks (BAliBASE)

Conclusion: Multiple Sequence Alignment

- Uses of Multiple Sequence Alignments (MSA)
- Sum-Of-Pairs Scoring (SP)
- Methods for solving the MSA problem: exact/heuristic
- Heuristics: no guarantee, but effective in practice
- Iterative/Progressive Alignment
- ClustalW: classic progressive alignment
  - distance matrix
  - guide tree
  - optimized gap penalties
- MUSCLE: progressive alignment with various improvements
  - guide tree re-estimation
  - iterative refinement
- Benchmarks (BAliBASE)