

# Multiple Sequence Alignment Sum-of-Pairs and Clustal-W

Ulf Leser

# This Lecture

---

- Multiple Sequence Alignment
  - The problem
  - Theoretical approach: Sum-of-Pairs scores
  - Practical approach: Clustal-W

# Multiple Sequence Alignment

---

- We now align multiple ( $k > 2$ ) sequences
  - Note: Also BLAST aligns only two sequences
- Why?
  - Imagine  $k$  sequences of the promoter region of genes, all regulated by the same transcription factor  $f$ . Which subsequence within the  $k$  sequences is recognized by  $f$ ?
  - Imagine  $k$  sequences of proteins that bind to DNA. Which subsequence of the  $k$  sequences code for the part of the proteins that performs the binding?
- General
  - We want to know the common part(s) in  $k$  sequences
  - “common” does not mean identical
  - These parts can be anywhere within the sequences

# Definition

---

- Definition

- A *multiple sequence alignment (MSA)* of  $k$  Strings  $s_i$ ,  $1 \leq i \leq k$ , is a table of  $k$  rows and  $l$  columns ( $\text{sum}(|s_i|) \geq l \geq \max(|s_i|)$ ) such that
  - Row  $i$  contains the sequence of  $s_i$ , with an arbitrary number of blanks being inserted at arbitrary positions
  - Every symbol of every  $s_i$  stands in exactly one column
  - No column contains only blanks

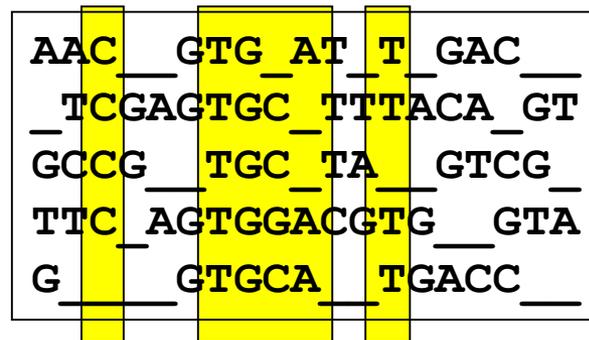
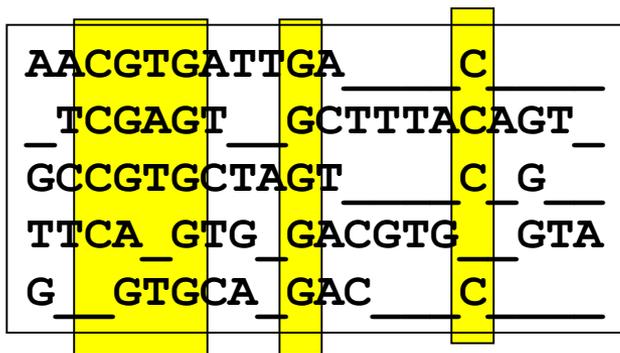
AACGTGATTGAC
TCGAGTGCTTTACAGT
GCCGTGCTAGTCG
TTCAGTGGACGTGGTA
GGTGCAGACC

AACGTGATTGA	_____	C	_____				
_____	TCGAGT	_____	GCTTTACAGT	_____			
GCCGTGCTAGT	_____	C	G	_____			
TTC	_____	GTG	_____	GACGTG	_____	GTA	
G	_____	GTGCA	_____	GAC	_____	C	_____

# Good MSA

---

- We are searching for **good (optimal) MSAs**
- Defining „optimal“ here is not as simple as in the  $k=2$  case
- Intuition
  - All sequences had a common ancestor and evolved by evolution
  - We want to assume as **few evolutionary events** as possible
  - Thus, we want **few columns** ( $\sim$  few INSDELS)
  - Thus, we want **homogeneous columns** ( $\sim$  few replacements)



# This Lecture

---

- Multiple Sequence Alignment
  - The problem
  - Theoretical approach: Sum-of-Pairs scores
  - Practical approach: Clustal-W

# What Should we Count?

---

- For two sequences
  - We scored each column using a scoring matrix
  - Find the alignment such that the total score is maximal
- But – how do we **score a column** with 5\*T, 3\*A, 1\*\_?
- We would need an exponentially large scoring matrix
- Alternative: **Sum-of-Pairs Score**
  - We only score the alignment of each **pair of sequences**
  - We aggregate **over all pairs** to score the MSA
  - We need a clever algorithm to find the MSA with the best score

# Formally

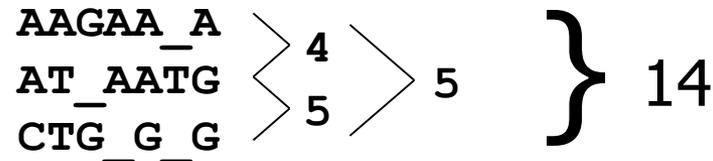
---

- Definition

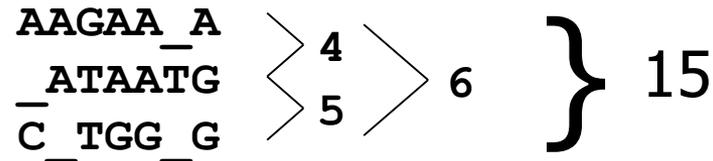
- Let  $M$  be a MSA for the set  $S$  of sequences  $S = \{s_1, \dots, s_k\}$
- The alignment of  $s_i$  with  $s_j$  *induced by  $M$*  is generated as follows
  - Remove from  $M$  all rows except  $i$  and  $j$
  - Remove all columns that contain only blanks
- The *sum-of-pairs score (sop)* of  $M$  is the sum of all pair-wise induced alignment scores
- The *optimal MSA for  $S$*  wrt. *sop* is the MSA with the lowest *sop*-score over all possible MSAs for  $S$

# Example

---



$$\begin{aligned}d/i &= 1 \\ r &= 1 \\ m &= 0\end{aligned}$$



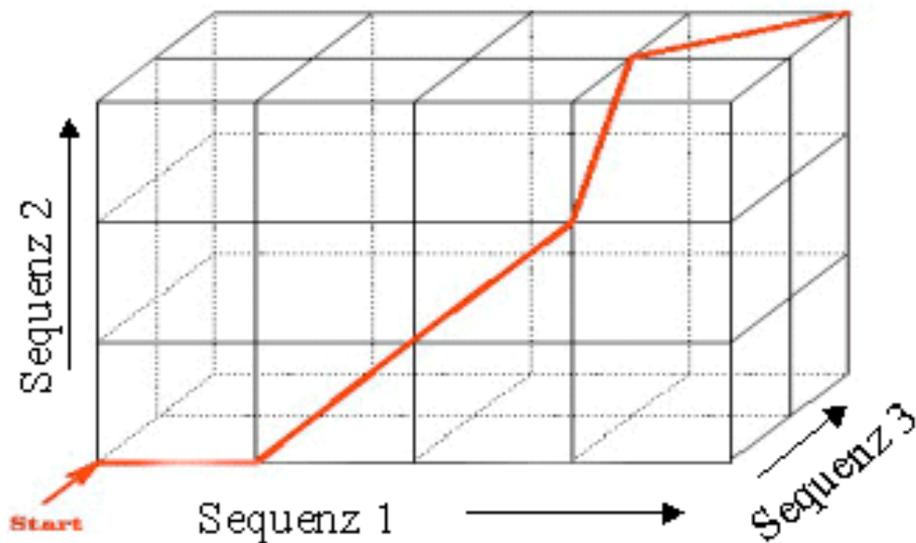
- Given a MSA over  $k$  sequences of length  $l$  – how complex is it to compute its sop-score?
- How do we find the **best MSA**?

# Analogy

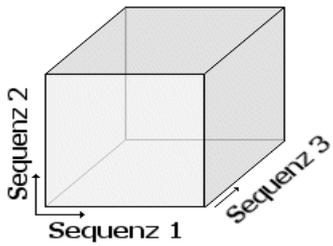
		0	1	2	3	4	5	6	7	
			w	r	i	t	e	r	s	
0		0	1	2	3	4	5	6	7	
1	v	1	1	2	3	4	5	6	7	
2	i	2	2	2	2	3	4	5	7	
3	n	3	3	3	3	3	4	5	6	
4	t	4	4	4	4	3	4	5	6	
5	n	5	5	5	5	4	4	5	6	
6	e	6	6	6	6	5	4	5	6	
7	r	7	7	6	7	6	5	4	5	

- Think of the  $k=2$  case
- Every alignment is a path through the matrix
- The **three possible directions** (down, right, down-right) conform to the **three possible constellations** in a column (XX, X\_, \_X)
- With growing paths, we align **growing prefixes** of both sequences

# Analogy

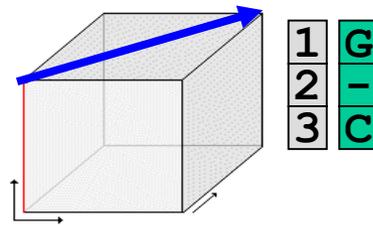
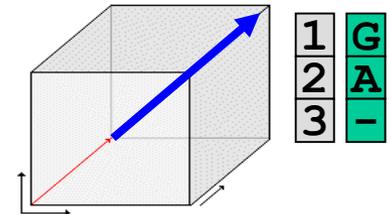
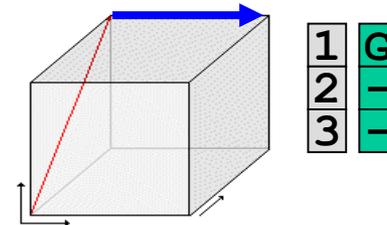
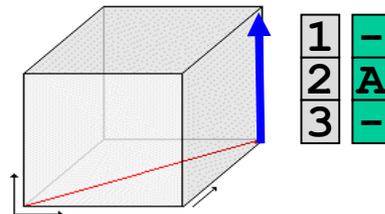
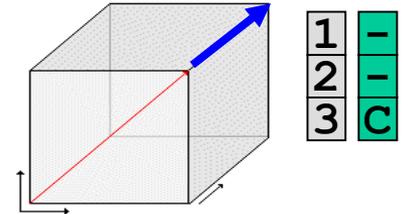
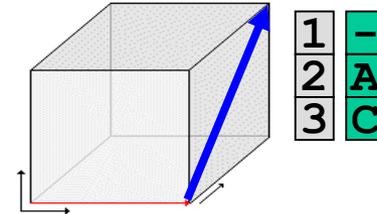
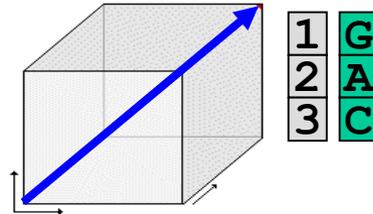


- Assume  $k=3$
- Think of a **3-dimensional cube** with the three sequences giving the values in each dimension
- Now, we have paths aligning growing **prefixes of three sequences**
- Every column has **seven possible constellations**  
(XXX, XX\_, X\_X, \_XX, X\_\_\_, \_X\_, \_\_X)



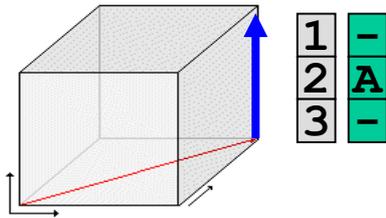
# All Possible Steps

- $d(i-1, j-1, k-1)$
- $d(i, j-1, k-1)$
- $d(i, j, k-1)$
- $d(i, j-1, k)$
- $d(i-1, j, k)$
- $d(i-1, j-1, k)$
- $d(i-1, j, k-1)$

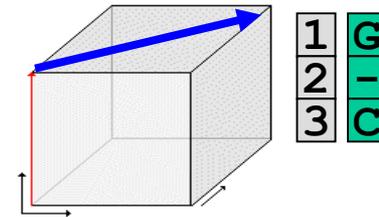


# Concrete Examples

$d(i, j-1, k)$



$d(i-1, j, k-1)$



- Best sop-score for  $d(i, j-1, k)$  is known
- We want to compute  $d(i, j, k)$
- This requires to align one symbol with two blanks (blank/blank does not count)
- $d(i, j, k) = d(i, j-1, k) + 2$

- Best sop-score for  $d(i-1, j, k-1)$  is known
- We want to compute  $d(i, j, k)$
- This requires aligning a blank with  $s_1[i-1]$  and with  $s_3[k-1]$  and to align  $s_1[i-1]$  and  $s_3[k-1]$
- $d(i, j, k) = d(i-1, j, k-1) + 2 + c_{ik}$

# Dynamic Programming in three Dimensions

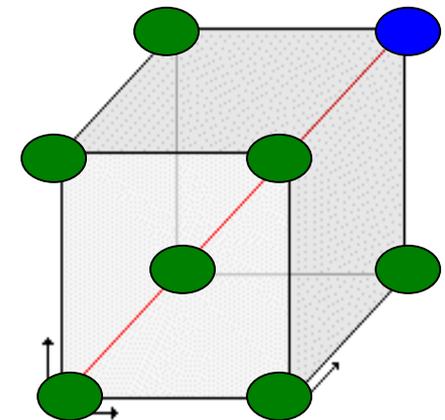
- We compute the best possible alignment  $d(i,j,k)$  for every **triple of prefixes** (lengths  $i,j,k$ ) using the following formula
  - We assume the usual edit costs:  $I/D/R=+1$ ,  $M=0$

$$d(i,j,k) = \min \left\{ \begin{array}{ll} d(i-1,j-1,k-1) + c_{ij} + c_{ik} + c_{jk} & \text{Three (mis)matches} \\ d(i-1,j-1,k) + c_{ij} + 2 & \text{One (mis)match, two ins} \\ d(i-1,j,k-1) + c_{ik} + 2 & \dots \\ d(i,j-1,k-1) + c_{jk} + 2 & \\ d(i-1,j,k) + 2 & \\ d(i,j-1,k) + 2 & \\ d(i,j,k-1) + 2 & \end{array} \right.$$

Let  $c_{ij} = 0$ , if  $S_1(i) = S_2(j)$ , else 1

Let  $c_{ik} = 0$ , if  $S_1(i) = S_3(k)$ , else 1

Let  $c_{jk} = 0$ , if  $S_2(j) = S_3(k)$ , else 1



# Initialization

---

- Of course, we have  $d(0,0,0)=0$
- Aligning in **one dimension**:  $d(i,0,0)=2*i$ 
  - Same for  $d(0,j,0)$ ,  $d(0,0,k)$
- Aligning in **two dimensions**:  $d(i,j,0)= \dots$ 
  - Let  $d_{a,b}(i,j)$  be the alignment score for  $S_a[1..i]$  with  $S_b[1..j]$
  - $d(i, j, 0) = d_{1,2}(i, j) + (i+j)$
  - $d(i, 0, k) = d_{1,3}(i, k) + (i+k)$
  - $d(0, j, k) = d_{2,3}(j, k) + (j+k)$

# Algorithm

---

```
initialize matrix d;
for i := 1 to |S1|
  for j := 1 to |S2|
    for k := 1 to |S3|
      if (S1(i) = S2(j)) then cij := 0; else cij := 1;
      if (S1(i) = S3(k)) then cik := 0; else cik := 1;
      if (S2(j) = S3(k)) then cjk := 0; else cjk := 1;
      d1 := d[i - 1, j - 1, k - 1] + cij + cik + cjk;
      d2 := d[i - 1, j - 1, k] + cij + 2;
      d3 := d[i - 1, j, k - 1] + cik + 2;
      d4 := d[i, j - 1, k - 1] + cjk + 2;
      d5 := d[i - 1, j, k) + 2;
      d6 := d[i, j - 1, k) + 2;
      d7 := d[i, j, k - 1) + 2;
      d[i, j, k] := min(d1, d2, d3, d4, d5, d6, d7);
    end for;
  end for;
end for;
```

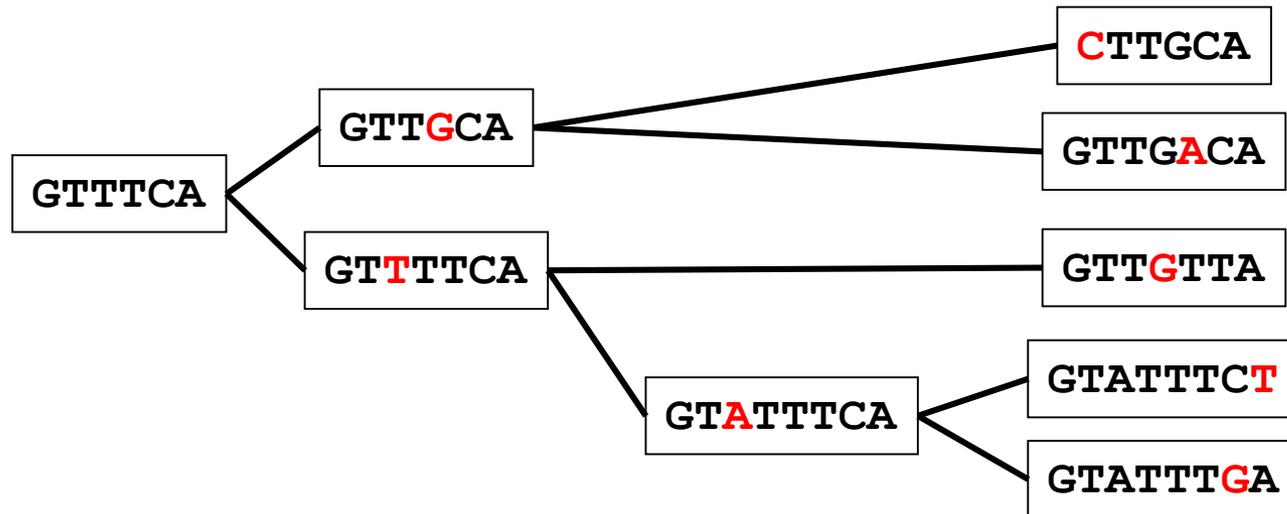
# Bad News: Complexity

---

- For 3 sequences of length  $n$ 
  - There are  $n^3$  cells in the cube
  - For each cell (top-left-front corner), we need to look at 7 corners
  - Together:  $O(7 * n^3)$  operations
- For  $k$  sequences of length  $n$ 
  - There are  $n^k$  cell corners in the cube
  - For each corner, we need to look at  $2^k - 1$  other corners
  - Together:  $O(2^k * n^k)$  operations

# Bad News: Biological Meaningfulness

- Recall our motivation: **Measure evolution**



- Real number of events: 8
- sop-score:  $2+3+6+6+2+\dots$ 
  - Single mutations are **counted multiple times**

CT\_TGC\_A  
GT\_TGACA  
GT\_TGTTA  
GTATTTCT  
GTATTTGA

# This Lecture

---

- Multiple Sequence Alignment
  - The problem
  - Theoretical approach: Sum-of-Pairs scores
  - Practical approach: Clustal-W

# Different Scoring Function

---

- If we knew the **phylogenetic tree** of the k sequences
  - Align every parent node pairwise with its children
  - Aggregate all alignment scores
  - This gives the “real” number of evolutionary operations
    - But not yet the best MSA
- But: Finding the true phylogenetic **tree requires a MSA**
  - Not covered in this lecture
- Use a heuristic: **ClustalW**
  - Thompson, J. D., Higgins, D. G. and Gibson, T. J. (1994). "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice." *Nucleic Acids Res* **22**(22): 4673-80.

# Clustal-W

---

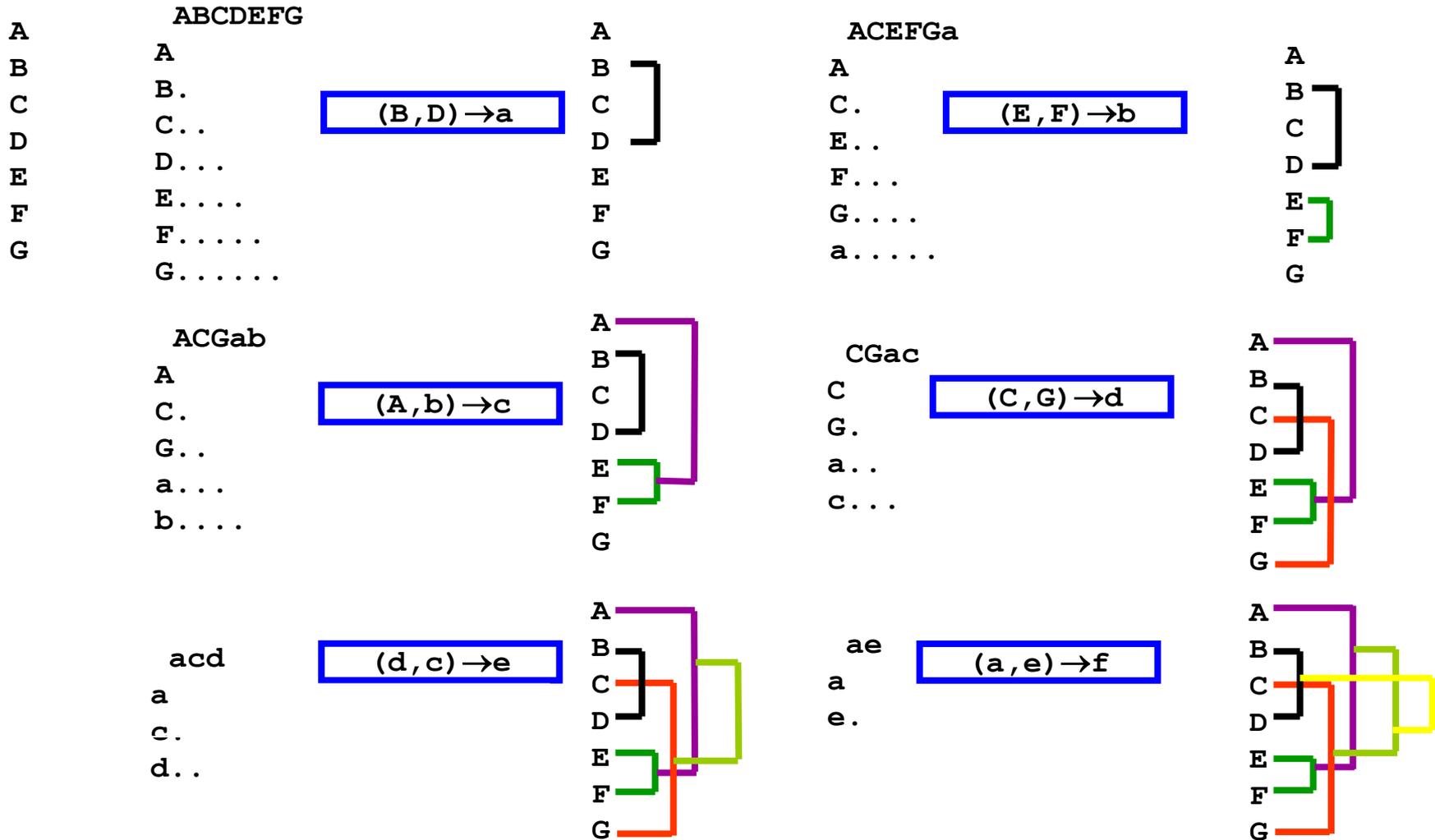
- Main idea
  - Compute a “good enough” phylogeny – the **guide tree**
  - Use the guide tree to iteratively align **small MSA to larger MSA**
    - “Progressive” MSA
    - Starting from single sequences
    - Add more and more sequences and smaller MSA to ever bigger MSA
    - Needs a fast method to **align two MSAs**
- Standard method for a long time
- Many newer (better) proposals
  - DAlign, T-Coffee, HMMT, PRRT, MULTALIGN, ...

# Step 1: Compute the Guide Tree

---

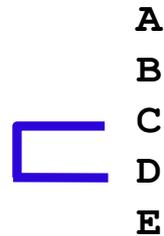
- Compute all pair-wise alignments and store in matrix M
  - $M[i,j] = \text{sim}(s_i, s_j)$
- Compute the guide tree using **hierarchical clustering**
  - Choose the smallest  $M[i,j]$
  - Let  $s_i$  and  $s_j$  form a **new (next) branch** of the tree
  - Compute the distance from the ancestor of  $s_i$  and  $s_j$  to all other sequences as the average of the distances to  $s_i$  and  $s_j$ 
    - Set  $M' = M$
    - Delete rows and columns  $i$  and  $j$
    - Add a new column and row ( $ij$ )
    - For all  $k \neq ij$ :  $M'[ij,k] = (M[i,k] + M[j,k]) / 2$
  - **Iterate until  $M'$  has only one column / row**

# Sketch

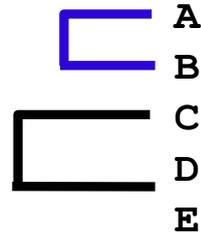


# Example

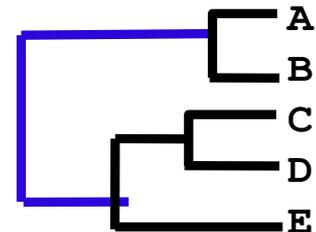
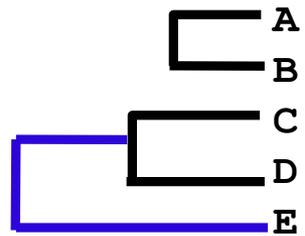
	A	B	C	D	E
A		17	59	59	77
B			37	61	53
C				13	41
D					21



	A	B	E	CD
A		17	77	59
B			53	49
E				31

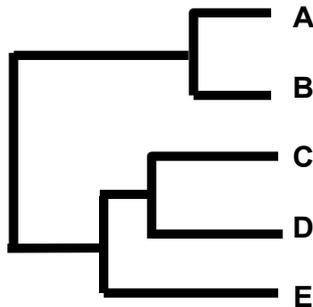


	E	CD	AB
E		31	65
CD			54



# Example

---



C PADKTNVKAAWGKVG**A**HAGEYGA

D AADKTNVKAAWSKVGGHAGEYGA

A PEEKSAV**T**ALWGKVN**V**DEYGG

B GEEKAAV**L**LALWDKVNEEEYGG

C PADKTNVKA**A**WG\_KVG**A**HAGEYGA

D AADKTNVKA**A**WS\_KVGGHAGEYGA

E AA\_\_TNV**K**TAWSSKVGGHAPA\_\_**A**

A PEEKSAV\_**T**ALWG\_**K**VN\_\_VDEYGG

B GEEKAAV\_**L**LALWD\_**K**VN\_\_EEYGG

C PADKTNV**K**AA\_**W**G\_**K**VGAHAGEYGA

D AADKTNV**K**AA\_**W**S\_**K**VGGHAGEYGA

E AA\_\_TNV**K**TA\_**W**SS**K**VGGHAPA\_\_**A**

Once a gap, always a gap

## Step 2: Progressive MSA

---

- **Pair-wise alignment of MSAs** in the order of the guide tree
- Aligning a MSA  $M_1$  with a MSA  $M_2$ 
  - Use the usual (global) alignment algorithm
  - To score a column, compute the **average score over all pairs of symbols** in these columns
- Example

<b>A</b>	...P...	Score of this column $(2 * s(P, A) + s(P, Y) +$ $2 * s(G, A) + s(G, Y) +$ $2 * s(P, A) + s(P, Y) ) / 9$
<b>B</b>	...G...	
<b>C</b>	...P...	
<b>D</b>	...A...	
<b>E</b>	...A...	
<b>F</b>	...Y...	

# Issues

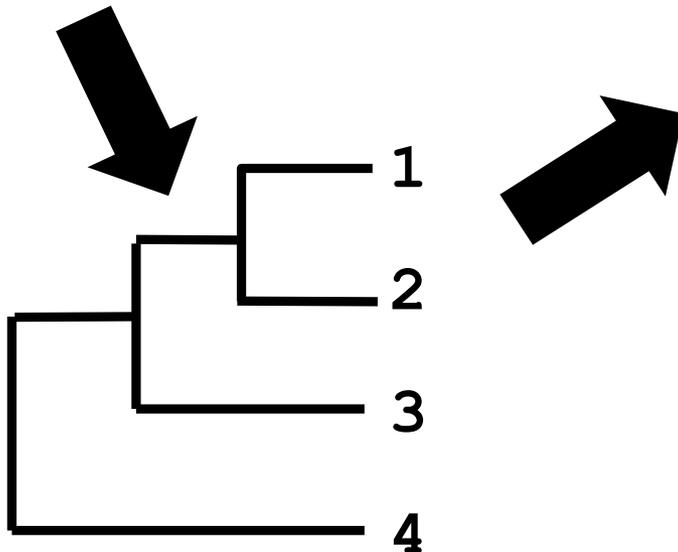
---

- There is a lot to say about whether **hierarchical clustering** actually computes the “correct” tree
- Clustal-W actually uses a different, more accurate phylogenetic algorithm called “neighbor-joining”
- Clustal-W is fast:  $O(k^2*n^2+k^2*\log(k))$ 
  - For k sequences; plus cost for computing pairwise alignments
- Idea behind **progressive alignment**
  - Find strong signals (highly conserved blocks) first
  - Outliers are added last
  - Increases the chances that conserved blocks survive
  - Several improvements to this scheme are known

# Problems of Progressive MSA

Angelehnt: Cedric Notredame, 2001

- 1: MAYFIELD THE LAST FAT RER
- 2: MAYFIELD THE FAST RAT
- 3: MAYLEENE IS A FAT RAT
- 4: MAYROONI THE LAST BIG RAT



```
MAYFIELD THE LAST FAT RER
MAYFIELD THE FAST RAT ____
MAYLEENE IS _ _ A _ FAT RAT
MAYROONI THE LAST BIG RAT
```

Besser:

```
MAYFIELD THE LAST FAT RER
MAYFIELD THE FAST ____ RAT
MAYLEENE IS _ _ A _ FAT RAT
MAYROONI THE LAST BIG RAT
```

# Further Reading

---

- Merkl & Waack, chapter 13
- Böckenhauer & Bongartz, chapter 5.3