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

Ulf Leser

This Lecture

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

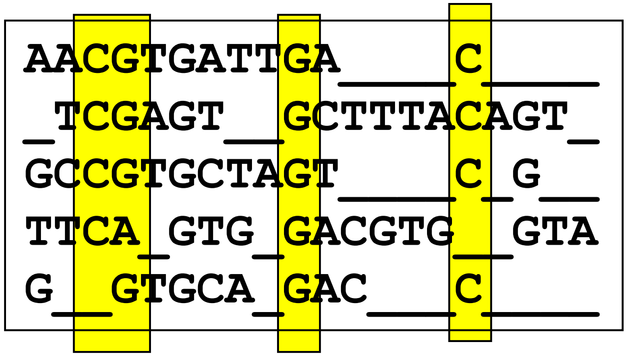
Multiple Sequence Alignment

- 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 different 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, $1 \le i \le k$, is a table of k rows and l columns ($\text{sum}(|s_i|) \ge l \ge \max(|s_i|)$) 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*

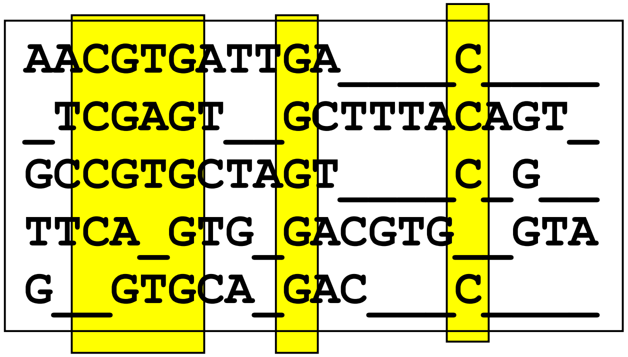
```
AACGTGATTGAC
TCGAGTGCTTTACAGT
GCCGTGCTAGTCG
TTCAGTGGACGTGGTA
GGTGCAGACC
```



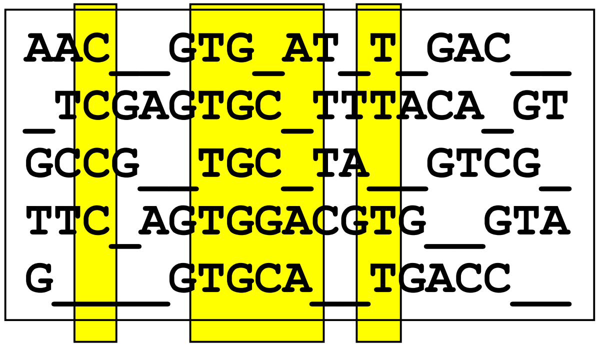
```
AACGTGATTGA____C____
_TCGAGT__GCTTTACAGT_
GCCGTGCTAGT____C_G__
TTCA_GTG_GACGTG__GTA
G__GTGCA_GAC___C____
```

Good MSA

- We are searching for good (optimal) MSAs
- Defining „optimal" 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 (~ few INSDELs)
  - Thus, we want homogeneous columns (~ few replacements)



```
AACGTGATTGA____C____
_TCGAGT__GCTTTACAGT_
GCCGTGCTAGT____C_G__
TTCA_GTG_GACGTG__GTA
G__GTGCA_GAC___C____
```



```
AAC__GTG_AT_T_GAC__
_TCGAGTGC_TTTACA_GT
GCCG__TGC_TA__GTCG_
TTC_AGTGGACGTG__GTA
G____GTGCA__TGACC__
```

This Lecture

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

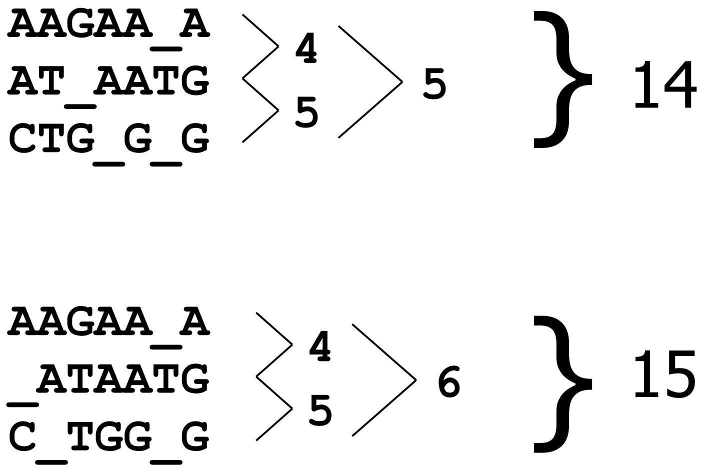
What Should we Count?

- 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 the k sequences
  - We aggregate over all O(k^2) pairs to score the MSA
  - We still 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* $S=\{s_1,\ldots,s_k\}$
  - *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



```
AAGAA_A
AT_AATG
CTG_G_G
```

```
AAGAA_A
_ATAATG
C_TGG_G
```

```
d/i = 1
r   = 1
m   = 0
```

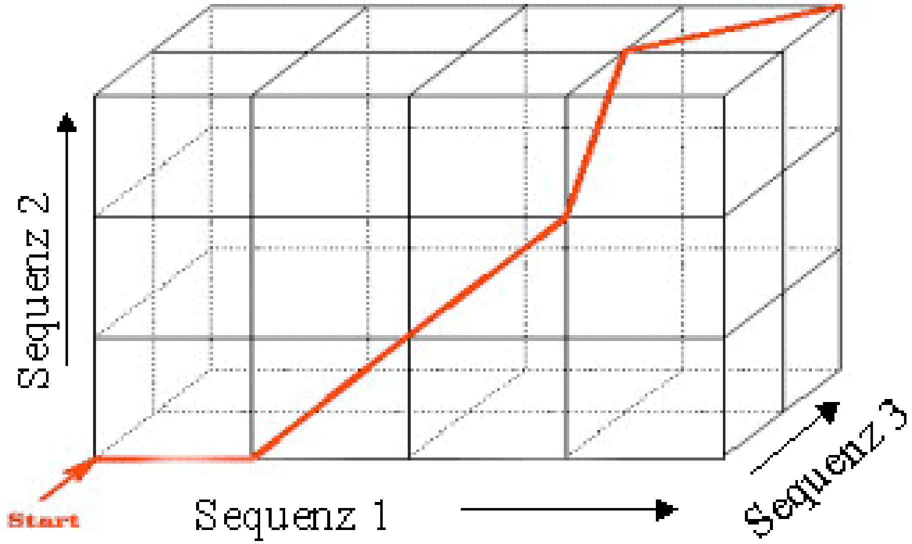
- Given a MSA over k sequences of length l – how complex is it to compute its sop-score?
- How do we find the best MSA?

Analogy

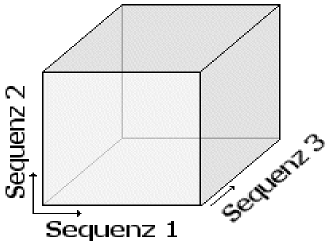
		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	1	2	3	4	5	6	7
1	v	1	1	2	3	4	5	6	7
2	i	2	2	2	2	3	4	5	7
3	n	3	3	3	3	3	4	5	6
4	t	4	4	4	4	3	4	5	6
5	n	5	5	5	5	4	4	5	6
6	e	6	6	6	6	5	4	5	6
7	r	7	7	6	7	6	5	4	5

- 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

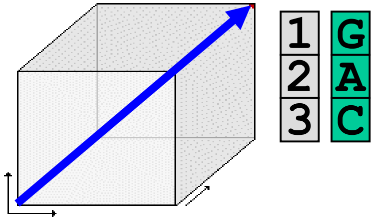


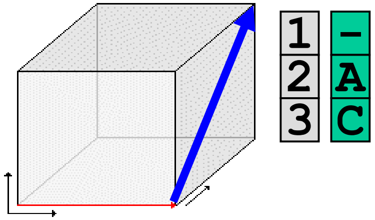
- Assume k=3
- Think of a 3-dimensional cube with the three sequences giving the values in each dimension
- Now, we have paths aligning growing prefixes of three sequences
- Every column has seven possible constellations (XXX, XX_, X_X, _XX, X__, _X_, __X)

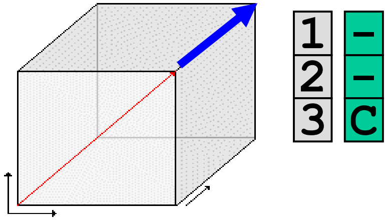


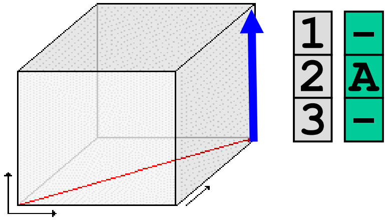
All Possible Steps

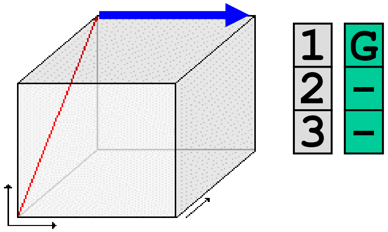
- d(i-1,j-1,k-1)
- d(i,j-1,k-1)
- d(i,j,k-1)
- d(i,j-1,k)
- d(i-1,j,k)
- d(i-1,j-1,k)
- d(i-1,j,k-1)

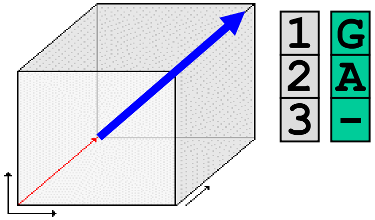


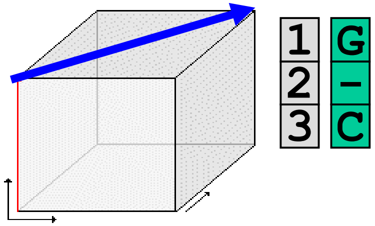






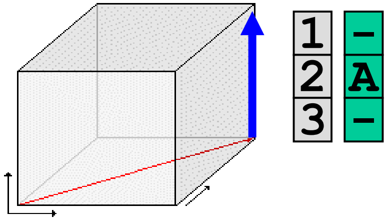






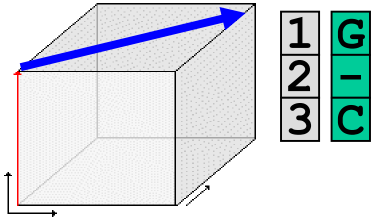
Concrete Examples

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



- 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)
- $d(i,j,k) = d(i,j-1,k) + 2$

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



- 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 $s_3[k-1]$
- $d(i,j,k) = d(i-1,j,k-1) + 2 + c_{ik}$

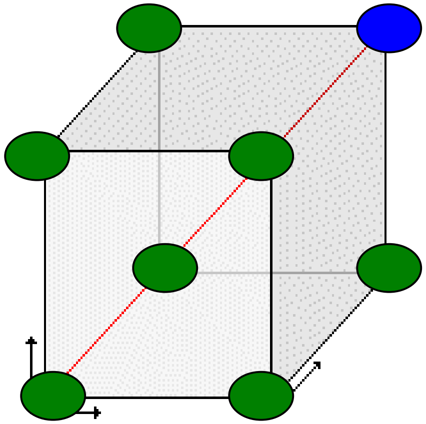
Dynamic Programming in three Dimensions

- 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

$$d(i,j,k) = \min \begin{cases} d(i-1,j-1,k-1) + c_{ij} + c_{ik} + c_{jk} \\ d(i-1,j-1,k) + c_{ij} + 2 \\ d(i-1,j,k-1) + c_{ik} + 2 \\ d(i,j-1,k-1) + c_{jk} + 2 \\ d(i-1,j,k) + 2 \\ d(i,j-1,k) + 2 \\ d(i,j,k-1) + 2 \end{cases}$$

Three (mis)matches
One (mis)match, two ins
…

$$\text{Let } c_{ij} = 0, \text{ if } S_1(i) = S_2(j), \text{ else } 1$$
$$\text{Let } c_{ik} = 0, \text{ if } S_1(i) = S_3(k), \text{ else } 1$$
$$\text{Let } c_{jk} = 0, \text{ if } S_2(j) = S_3(k), \text{ else } 1$$



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: $d(i,j,0)=$ …
  - 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

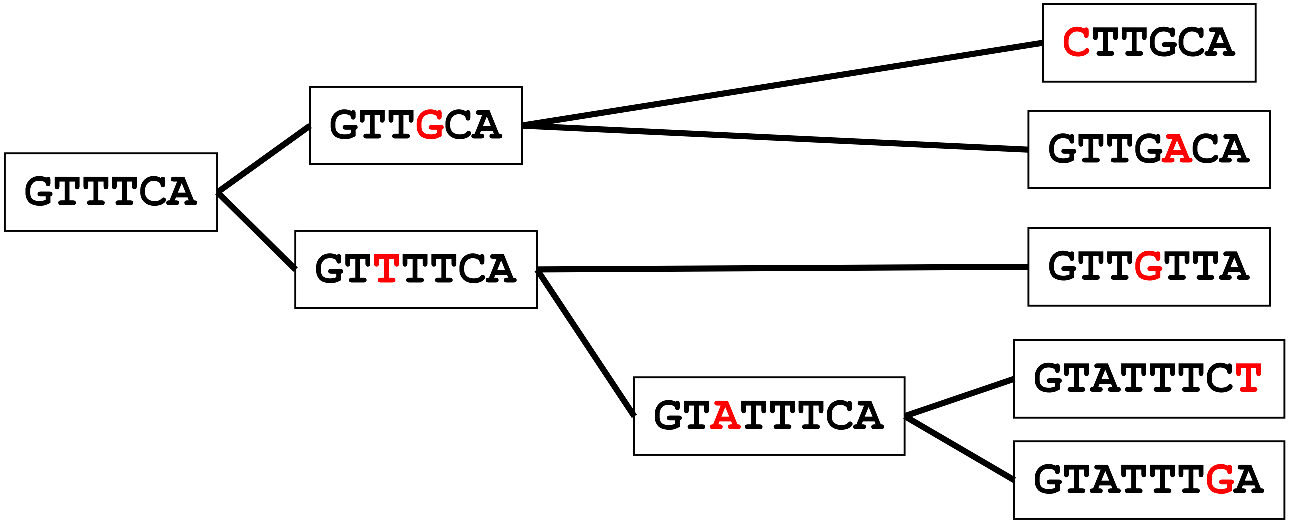
```
initialize matrix d;
for i := 1 to |$S_1$|
    for j := 1 to |$S_2$|
        for k := 1 to |$S_3$|
            if ($S_1(i) = S_2(j)$) then $c_{ij}$ := 0; else $c_{ij}$ := 1;
            if ($S_1(i) = S_3(k)$) then $c_{ik}$ := 0; else $c_{ik}$ := 1;
            if ($S_2(j) = S_3(k)$) then $c_{jk}$ := 0; else $c_{jk}$ := 1;
            $d_1$ := d[i - 1,j - 1,k - 1] + $c_{ij}$ + $c_{ik}$ + $c_{jk}$;
            $d_2$ := d[i - 1,j - 1,k] + $c_{ij}$ + 2;
            $d_3$ := d[i - 1,j,k - 1] + $c_{ik}$ + 2;
            $d_4$ := d[i,j - 1,k - 1] + $c_{jk}$ + 2;
            $d_5$ := d[i - 1,j,k) + 2;
            $d_6$ := d[i,j - 1,k) + 2;
            $d_7$ := d[i,j,k - 1) + 2;
            d[i,j,k] := min($d_1$, $d_2$, $d_3$, $d_4$, $d_5$, $d_6$, $d_7$);
        end for;
    end for;
end for;
```

Bad News 1: 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 2: Biological Meaningfulness

- Recall our motivation: Measure evolution



- Real number of events: 8
- sop-score: 2+3+6+6+2+…
  - 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 true 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: 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.

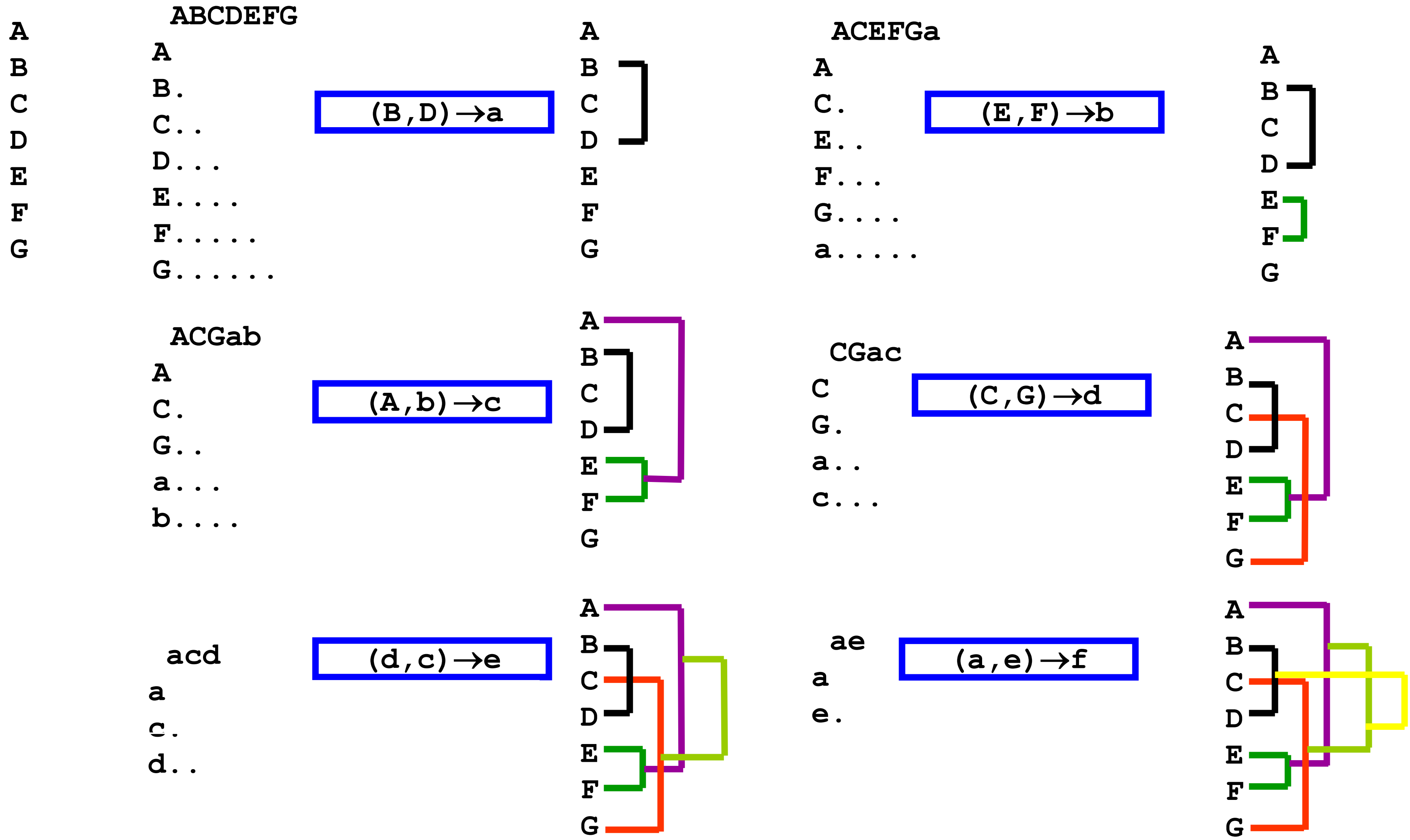
Clustal-W

- 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 turn smaller MSA into 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] = 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



```
A
B
C
D
E
F
G
```

```
 ABCDEFG
A
B.
C..
D...
E....
F.....
G......
```

(B,D)→a

```
 ACEFGa
A
C.
E..
F...
G....
a.....
```

(E,F)→b

```
 ACGab
A
C.
G..
a...
b....
```

(A,b)→c

```
 CGac
C
G.
a..
c...
```

(C,G)→d

```
 acd
a
c.
d..
```

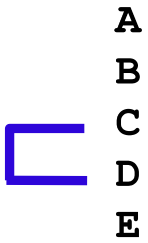
(d,c)→e

```
 ae
a
e.
```

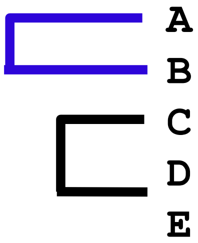
(a,e)→f

Example

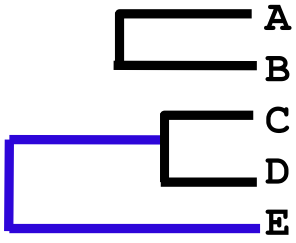
	A	B	C	D	E
A		17	59	59	77
B			37	61	53
C				13	41
D					21

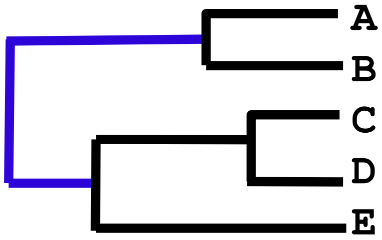


	A	B	E	CD
A		17	77	59
B			53	49
E				31

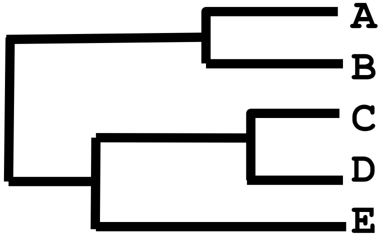


	E	CD	AB
E		31	65
CD			54





Example



```
C   PADKTNVKAAWGKVGAHAGEYGA
D   AADKTNVKAAWSKVGGHAGEYGA

A   PEEKSAVTALWGKVNVDEYGG
B   GEEKAAVLALWDKVNEEEYGG

C   PADKTNVKAAWG_KVGAHAGEYGA
D   AADKTNVKAAWS_KVGGHAGEYGA
E   AA__TNVKTAWSSKVGGHAPA__A

A   PEEKSAV_TALWG_KVN__VDEYGG
B   GEEKAAV_LALWD_KVN__EEEYGG
C   PADKTNVKAA_WG_KVGAHAGEYGA
D   AADKTNVKAA_WS_KVGGHAGEYGA
E   AA__TNVKTA_WSSKVGGHAPA__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 each column
- Example

```
A   …P…
B   …G…
C   …P…

D   …A…
E   …A…
F   …Y…
```

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
```

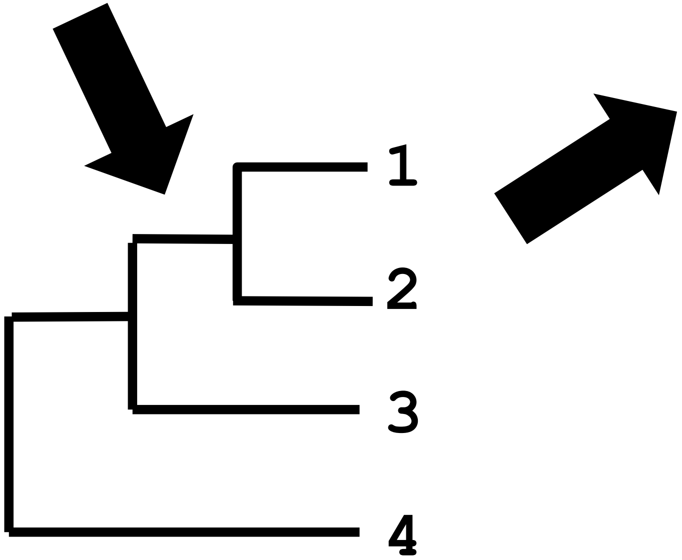
Issues

- 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 of length n
- 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