Multiple alignment:

Approximation and heuristics

We will discuss:

- The concept of *approximation algorithms*
- A factor 2 approximation for MSA
- Progressive heuristics for MSA

References

- D. Gusfield. Algorithms on Strings, Trees, and Sequences. Cambridge University Press, chapter 14.6

- Thompson, Higgins, Gibson, (1994) `ClustalW`: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. Nucleic Acids Research, 22:4673-4680.

- Mount, Bioinformatics, part “Progressive Alignment” in chapter “Multiple sequence alignment”

Introduction

So, MSA is *NP*-complete. And now?

There are *three* possibilities:

- We use *exponential*-time algorithms, which solve small/medium-sized instances to provable optimality.
  E.g., high-dimensional DP (Ex. 7.1, 7.2, P3 choice 1), high-dimensional $\mathcal{A}^*$, mathematical programming techniques, ...

- We seek *near-optimal* instead of optimal solutions (in polynomial time).
  Approximation algorithms guarantee that they are not too far away from the optimal solution. E.g., P3, choice 2.

- We use *heuristics*.
  Constructive/improvement heuristics, local/tabu search, genetic algorithms, simulated annealing, and the like. E.g., progressive alignment.

Approximation algorithms

An *α-approximation algorithm* for a minimization* problem achieves an *approximation ratio* (or factor) $\alpha \geq 1$, that is,

- it runs in polynomial time, and,
- for every input x, computes a solution of cost at most

$$\alpha \cdot m^*(x) \ .$$

Example. 2-approximation for MIN-TSP (blackboard)

The class *APX* contains those optimization problems, which are constant factor-approximable. From the example, we know that MIN-TSP $\in$ *APX*.

*The definition for maximization problems is analogous. You will need it in Ex. 8.4

Approximation algorithms (2)

An algorithm is a *polynomial-time approximation scheme* (PTAS), if, in addition to the input x, it takes a parameter ε and is an $(1+\varepsilon)$-approximation algorithm.

That is, the algorithm can get as close to the optimum as desired.

The trade-off is the running time. Although it must be bounded by a polynomial in $|x|$, it may depend arbitrarily on ε, e.g., $O(|x|^{1/\varepsilon})$.

A *fully polynomial-time approximation scheme* (FPTAS) is a PTAS whose running time is polynomial, both in $|x|$ **and** $\frac{1}{\varepsilon}$.

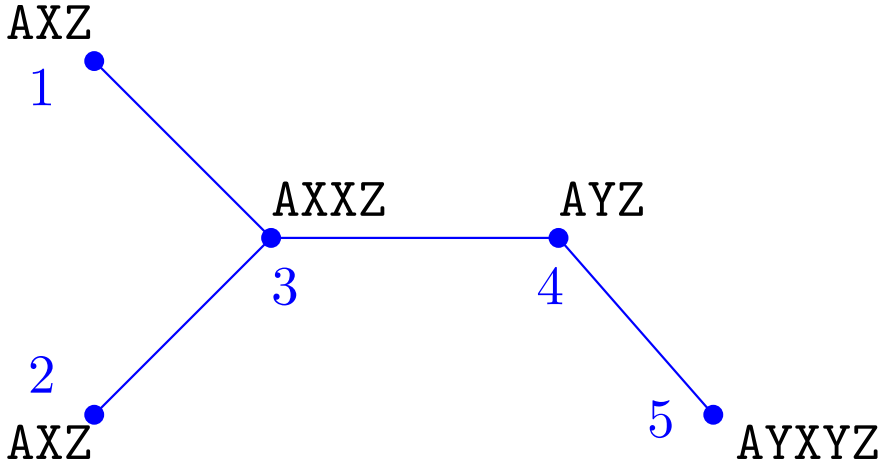
Some optimization problems have an FPTAS (e.g., MAX-KNAPSACK), some have a PTAS but no FPTAS (e.g, Euclidean MIN-TSP), others do not even have a PTAS (e.g., MAX-VERTEX-COVER). All provable assuming $P \neq NP$.

Approximating MIN-MSA

We will now develop a 2-approximation algorithm for MIN-MSA, which is due to Gusfield (1993).

We will refer by $D(s_i, s_j)$ to the optimal pairwise weighted edit distance between two sequences s_i and s_j and by $d(A)$ to the cost of an alignment A.

Definition. Let S be a set of sequences, and let T be a tree whose nodes are labeled by the strings in S. A multiple alignment A of S is *consistent* with T if the score $d(A_{i,j})$ of the projection $A_{i,j}$ is equal to $D(s_i, s_j)$ for each pair s_i and s_j that is linked by an edge in T.



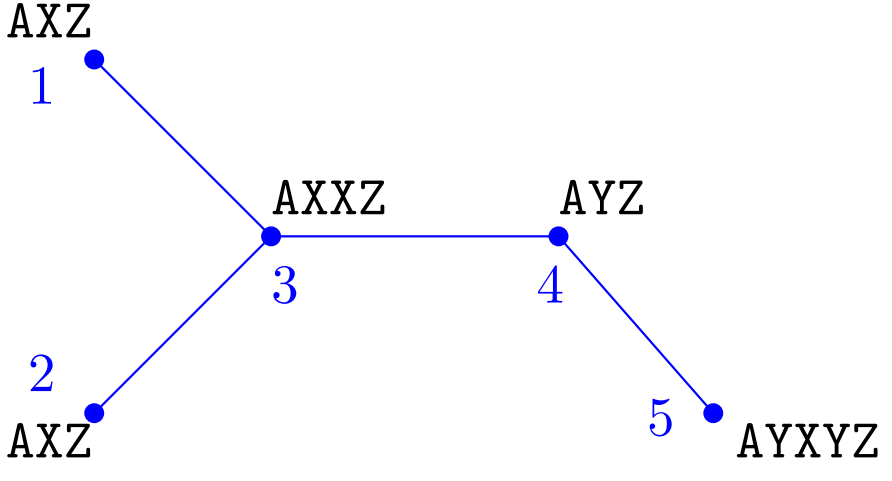
3	A X X - Z
1	A X - - Z
2	A - X - Z
4	A Y - - Z
5	A Y X Y Z

Approximating MIN-MSA (2)

The following statement is quite useful.

Theorem. For any set of sequences S and any tree T whose nodes are labeled with $s \in S$, we can efficiently find a multiple alignment of S that is consistent with T.

Constructive (!) proof. Blackboard.



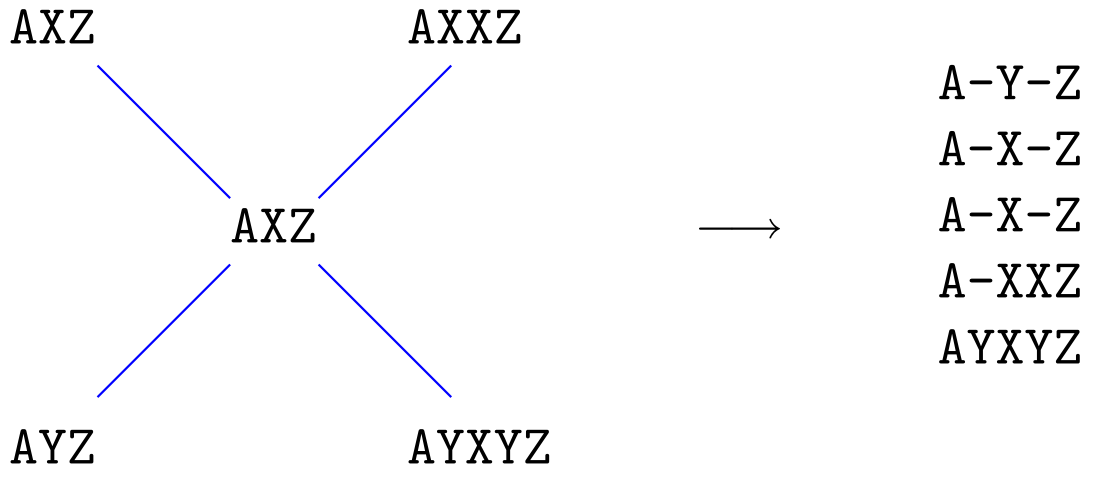
3	A X X - Z
1	A X - - Z
2	A - X - Z
4	A Y - - Z
5	A Y X Y Z

Approximating MIN-MSA (3)

The *center* of the sequences in S is

$$s_c := \arg\min_{s \in S} \sum_{t \in S} D(s,t) \ , \quad \text{let} \quad M := \sum_{t \in S} D(s_c, t) \ .$$

The *center star* is a star tree of $|S|$ nodes with s_c in the middle:



According to the theorem on the previous slide we now build a multiple alignment A^c that is consistent with the center star of the input sequences.

Clearly,

$$d(A^c) = \sum_{i<j} d(A^c_{i,j}) \ , \quad d(A^c_{i,j}) \geq D(s_i, s_j) \ , \quad \text{and} \quad d(A^c_{i,c}) = D(s_i, s_c) \ .$$

Approximating MIN-MSA (4)

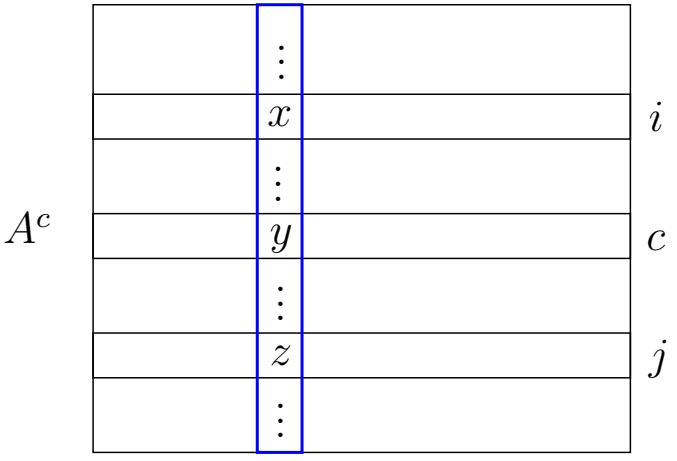
We say that a scoring scheme δ satisfies the *triangle inequality* if for any three characters $x, y, z \in \Sigma$ the following holds:

$$\delta(x, z) \leq \delta(x, y) + \delta(y, z) \ .$$

If this is the case, then for any pair of sequences s_i and s_j

$$d(A^c_{i,j}) \leq d(A^c_{i,c}) + d(A^c_{c,j}) = D(s_i, s_c) + D(s_c, s_j) \ .$$

Proof. The inequality holds because the triangle inequality holds for each column and the score is the sum over all columns; the equality holds due to the construction of A^c.



Approximating MIN-MSA (5)

The following main theorem states that the center star approximation yields an approximation ration of two, that is, alignments constructed this way are at most twice as bad as optimal alignments.

Theorem. Let A^* be an optimal alignment of S, and let $k := |S|$.

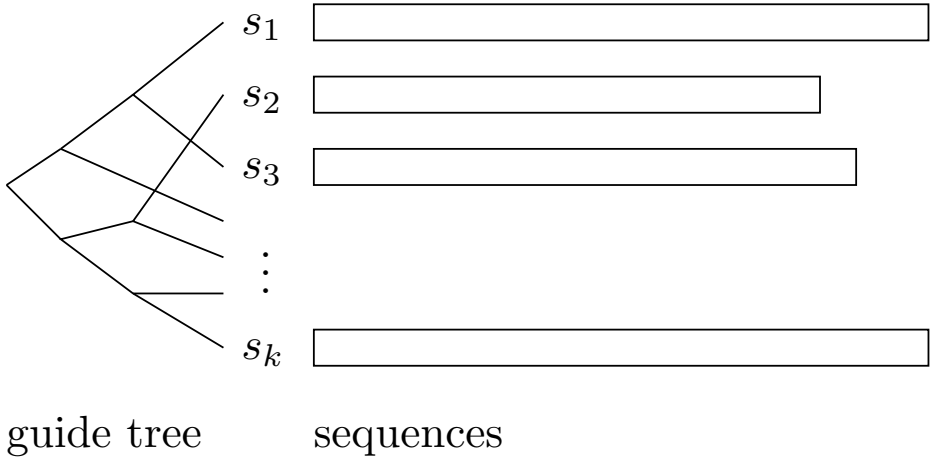
$$\frac{d(A^c)}{d(A^*)} \leq 2\frac{k-1}{k} < 2 \ ,$$

if the scoring scheme satisfies the triangle inequality.

Proof. Blackboard.

Progressive alignment

Probably the most commonly used approach to MSA is *progressive alignment*. In general, this works by constructing a series of pairwise alignments, first starting with pairs of sequences and then later also aligning sequences to existing alignments (profiles) and profiles to profiles.



Progressive alignment is a heuristic and does not directly optimize any known global scoring function of alignment correctness. However, it is fast and efficient, and often provides reasonable results.

Progressive alignment (2)

`ClustalW` is one of the most popular programs for computing a MSA. The weighted version, `ClustalW` is the successor of `Clustal`, which was introduced about ten years ago. It is similar to the older *Feng-Doolittle* method.

Algorithm `ClustalW` progressive alignment

1. Construct a distance matrix of all $\frac{k(k-1)}{2}$ pairs by pairwise dynamic programming alignment followed by approximate conversion of similarity scores to evolutionary distances.

2. Construct a *guide* tree*, which is a binary tree whose leaves represent the sequences and whose interior node represent alignments. The root node represents a complete multiple alignment.

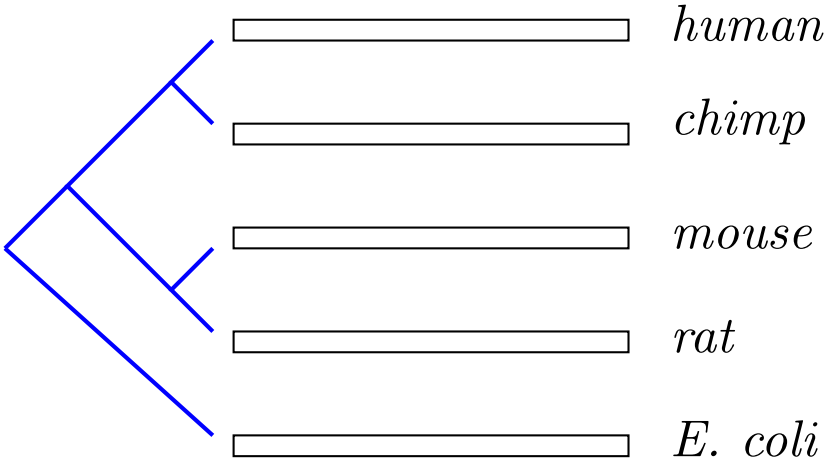
3. Progressively align sequences at nodes of tree, using sequence-sequence, sequence-profile and profile-profile alignment.

*using the neighbor joining tree-building method

Progressive alignment (3)

How to choose a good guide tree?

We try to align along an approximation* of the underlying phylogenetic tree. The more similar two sequences are, the more reliable is their pairwise alignment, i.e., we make fewer mistakes in the beginning.



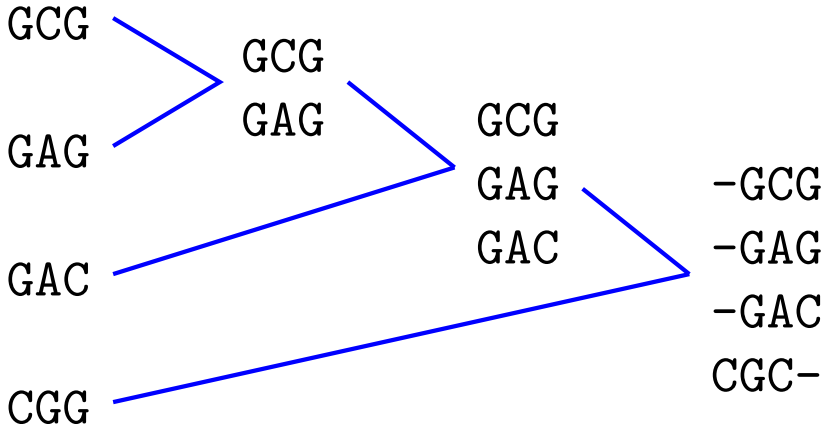
*Why only approximation?

Progressive alignment (4)

Example.

$S = \{\texttt{GCG}, \texttt{GAC}, \texttt{CGG}, \texttt{GAG}\}$

guide tree, e.g.,



Progressive alignment (5)

There are no provable performance guarantees associated with the program. However, it works well in practice and the following features contribute to its accuracy:

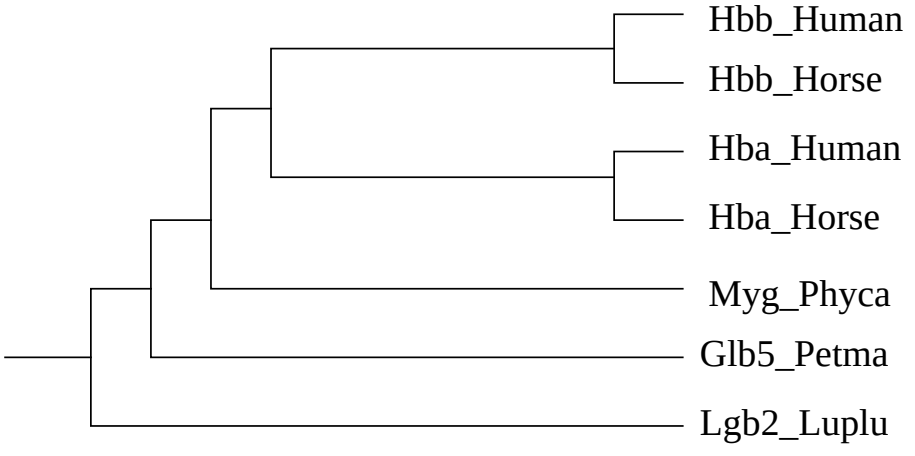
- Sequences are weighted to compensate for the defects of the SP score.
- The substitution matrix used is chosen based on the similarity expected of the alignment, e.g., BLOSUM80 for closely related sequences and BLOSUM50 for less related ones.
- Position-specific gap-open profile penalties are multiplied by a modifier that is a function of the residues observed at the position (hydrophobic residues give higher gap penalties than hydrophilic or flexible ones.)
- Gap-open penalties are also decreased if the position is spanned by a consecutive stretch of five or more hydrophilic residues.
- Gap-open and gap-extension penalties increase, if there are no gaps in the column, but gaps near by.

Progressive alignment (6)

Example. For a set of sequences A, we compute all pairwise distances:

Hbb_Human	1	0					
Hbb_Horse	2	0.17	0				
Hba_Human	3	0.59	0.60	0			
Hba_Horse	4	0.59	0.59	0.13	0		
Myg_Phyca	5	0.77	0.77	0.75	0.75	0	
Glb5_Petma	6	0.81	0.82	0.73	0.74	0.80	0
Lgb2_Luplu	7	0.87	0.86	0.86	0.88	0.93	0.90
		1	2	3	4	5	6

This gives rise to the following tree:



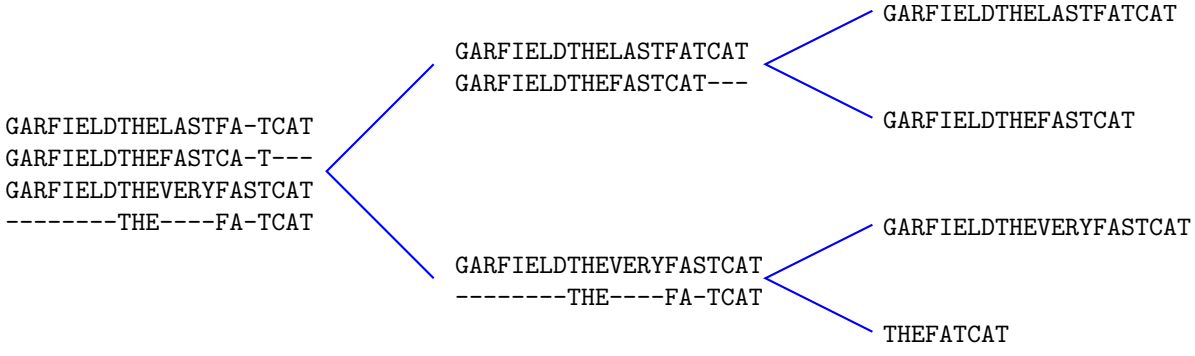
First we align Hbb_Human with Hbb_Horse and Hba_Human with Hba_Horse. Then we align the two resulting profiles etc.

Progressive alignment (7)

Disadvantages.

- Quality depends on initial choice. Once subalignments have been buit, they are “frozen” (*once a gap, alwyas a gap*). Iterative *improvement* heuristics (e.g., SAGA) may help.

- Greedy view at each step.




```
clustalW   (score=20)

SeqA      GARFIELDTHELASTFA-TCAT
SeqB      GARFIELDTHEFASTCA-T---
SeqC      GARFIELDTHEVERYFASTCAT
SeqD      --------THE----FA-TCAT


correct   (score=24)

SeqA      GARFIELDTHELASTFA-TCAT
SeqB      GARFIELDTHEFAST----CAT
SeqC      GARFIELDTHEVERYFASTCAT
SeqD      --------THE----FA-TCAT
```

- Most methods always return a multiple alignment, regardless of whether the sequences are related or not*. Biological thinking has to be done by the user.

*cf. the example sequences for the programming exercise P2, which are completely unrelated