

Algorithms in Genome Analysis, Spring 2023

Veli Mäkinen

Week 3

Multiple Sequence Alignment (MSA) and graph alignments

Multiple sequence alignment (MSA)

Small parsimony problem for scoring MSAs, NP-hard to compute optimal MSAs, exponential dynamic programming, heuristics, applications

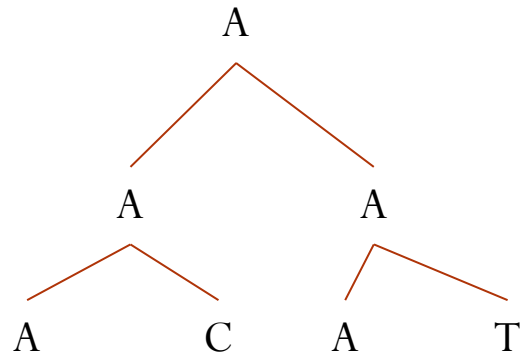
Multiple sequence alignment (MSA)

```
ACGATCGAGCGATC-ACGAT-GAGCAGCTAGC-ACTAGCGAGCATCGAC
ACGATC-AGCGATCGACGATCGTGCAGCTAGCGACTAGCGAGCATCGAC
ACGATC-AGCGATCGACGAT-GAGCAGCTAGC-ACTAGCGAGCATCGAC
ACGATC-AGCGATCGACGATCGTGCAGCTAGC-ACTAGCGAGCATCGAC
ACGATG-AGCGATCGACGATCGAGCAGCTAGC-ACTAGCGAGCATCGAC
ACGATCGAGCGATC-ACGATCGAGCAGCTAGC-ACTAG-GAGCATCGAC
ACGATGGAGCGACCGACGATCGTGCAGCTAGC--CTAG-GAGCATCGAC
ACGATCGAGCGACCGACGATCGAGCAGCTAGCGCCTAG-GAGCATCGAC
ACGATC-AGCGACCGACGATCGAGCAGCTAGCGCCTAG-GAGCATCGAC
ACGATC-AGCGACCGACGATCGTGCAGCTAGCGCCTAGCGAGCATCGAC
ACGATG-AGCGATCGACGATCGAGCAGCTAGCGCCTAGCGAGC-----C
ACGATCGAGCGATCGACGATCGAGCAGCTAGCGACTAG-GAGCATCGAC
ACGATCGAGCGATC-ACGA----GCAGCTAGCGA-----GAGCATCGAC
ACGATCGAGCGATCGACGATCGAGCAGCTAGCGACTAG-GAGCATCGAC
```

MSA problem

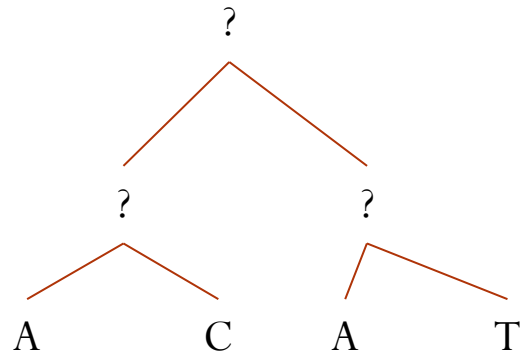
- Add gaps to m sequences to make them equal length
- Let $M[1..m, 1..n]$ be the resulting matrix
- $S(M[1..m, 1..n]) = \text{sum of column scores}$
- Finding M with max score $S(M)$ is NP-hard (see the course book)
- One can extend the dynamic programming pair-wise global alignment algorithm to solve the problem in $O(2^m N^m f(m))$ time, where N is the length of the longest input sequence and $f(m)$ is the time needed to compute the score of a column
- How to define the score of a column?
 - Sum of pair-wise alignment scores (SP)
 - Entropy
 - Small parsimony score
 - Fix an evolutionary tree whose leaves are the rows
 - Label the internal nodes so that the substitution scores between a child and its parent are maximized
 - Find a tree that maximizes the score

Small parsimony problem

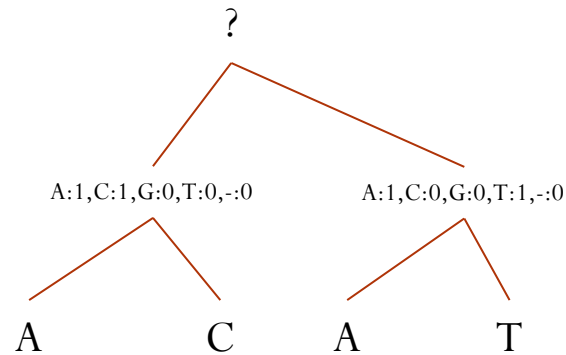


$$\text{Score} = 4s(A,A) + s(A,C) + s(A,T)$$

Small parsimony problem solved by dynamic programming

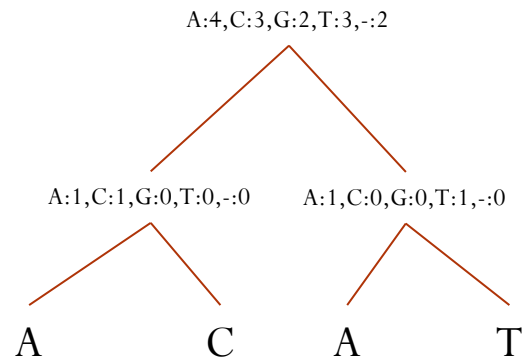


Small parsimony problem solved by dynamic programming



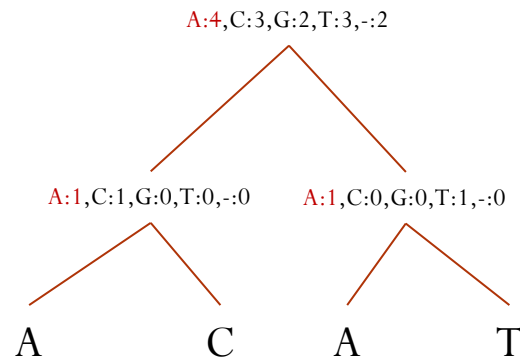
Assume $s(a,a)=1$, $s(a,b)=0$, for $a \neq b$

Small parsimony problem solved by dynamic programming



Assume $s(a,a)=1$, $s(a,b)=0$, for $a \neq b$

Small parsimony problem solved by dynamic programming



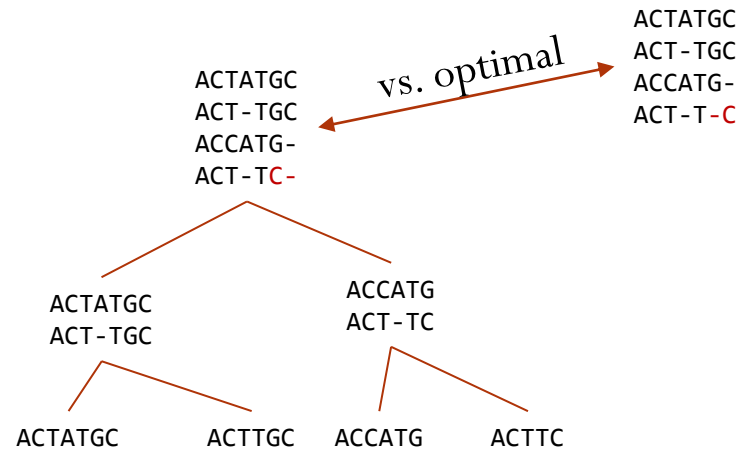
$O(|\Sigma|^2 m)$ time

Assume $s(a,a)=1$, $s(a,b)=0$, for $a \neq b$

Progressive alignment: heuristic for solving the MSA problem

- Compute alignment score $S(A,B)$ or edit distance $D(A,B)$ for all pairs of input sequences A and B
- Use these scores / distances to find an optimal evolutionary tree using distance-based phylogeny algorithms (see Elements of Bioinformatics)
- This tree is used as a *guide tree* to align the sequences from bottom to top
 - Two neighboring leaves are aligned optimally using global alignment
 - Alignments in internal nodes are interpreted as sequences of columns (profiles) and aligned optimally to each others using global alignment, with the modification of using MSA-type scoring

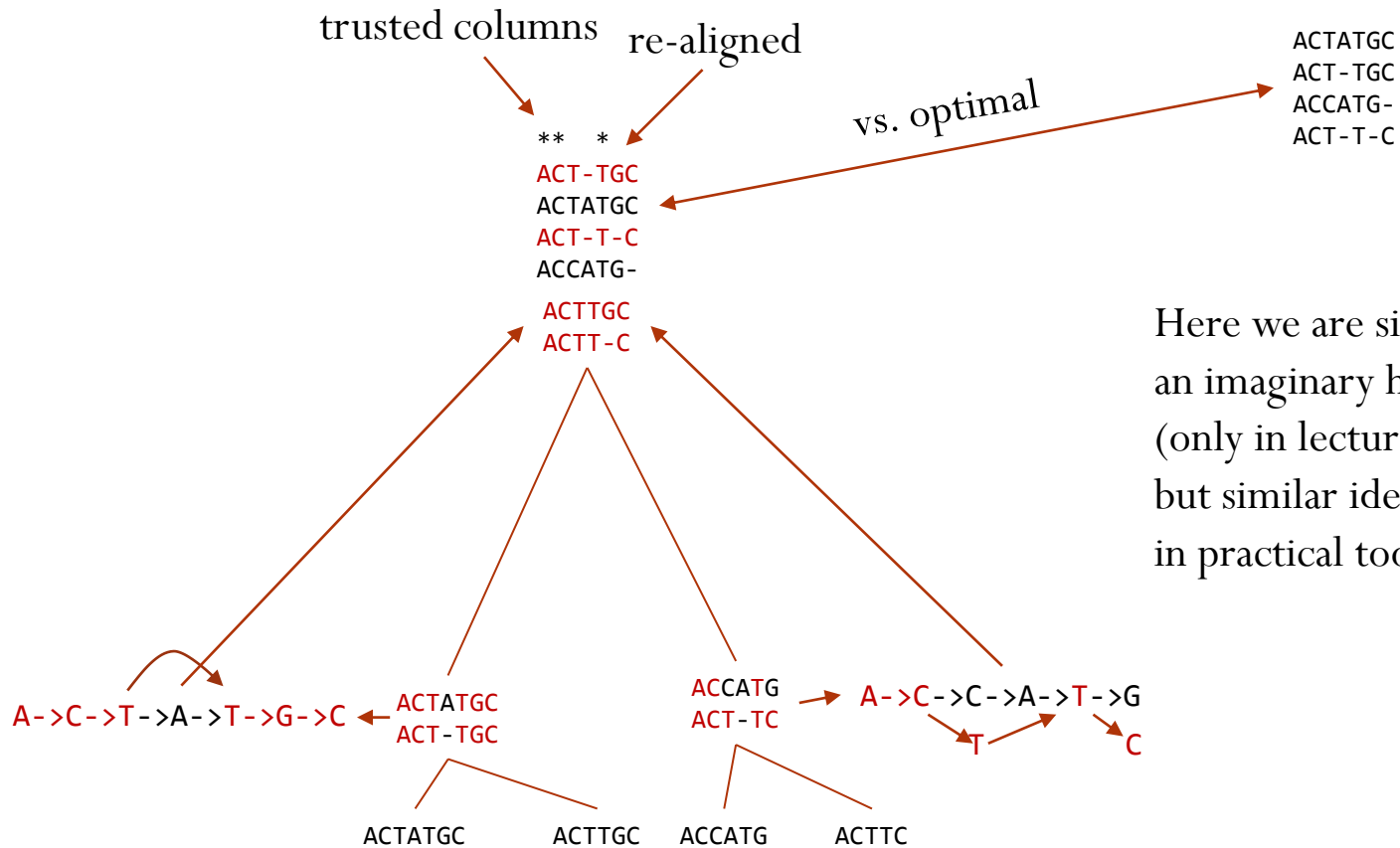
Progressive alignment



DAG-path alignment

- To circumvent the "once a gap, always a gap" problem of progressive alignment, one can replace the sequence of columns representation with a labeled directed acyclic graph (labeled DAG)
- Then the core alignment problem becomes that of finding a path P^A in DAG A and a path P^B in DAG B with maximum alignment score $S(P^A, P^B)$
- This problem is easy to solve by a slight modification of the global alignment dynamic programming:
 - Let $\ell(v)$ give the single-character label of node v and let $S(v,w)$ give an optimal alignment score among paths ending at nodes v and w of two DAGs, respectively.
 - If an optimal alignment ends with a substitution $\ell(v) \rightarrow \ell(w)$, it is sufficient to look for the pair (v', w') of nodes, where v' is an in-neighbor of v and w' is an in-neighbor of w , such that score $S(v', w')$ is maximum:
 $S(v,w) = S(v', w') + s(\ell(v), \ell(w))$
 - Alignments ending with gaps can be handled analogously
 - This yields an $O(|E^A||E^B|)$ time algorithm on two DAGs A and B with the sets of edges E^A and E^B , respectively

Progressive alignment with DAG-paths



Here we are simulating an imaginary heuristic (only in lecture's head), but similar ideas are used in practical tools

Jumping alignment

- Given sequence A, MSA M, and threshold k for jumps.
- Find a path through the columns of M spelling sequence B so that you can jump at most k times from row to row and $S(A,B)$ is maximized.

A ACGATCGAGCGATCACGATGAGCAGCTAGCACTAGCGAGCATCGAC

ACGATC-AGCGATCGACGATCGTGCAGCTAGCGACTAGCGAGCATCGAC
 ACGATC-AGCGATCG**ACGAT-GAGCAGCTAGC-ACTAGCGAGCATCGAC**

ACGATC-AGCGATCGACGATCGTGCAGCTAGC-ACTAGCGAGCATCGAC
 ACGATG-AGCGATCGACGATCGAGCAGCTAGC-ACTAGCGAGCATCGAC

k=1

B **ACGATCGAGCGATC**-ACGATCGAGCAGCTAGC-ACTAG-GAGCATCGAC

ACGATGGAGCGACCGACGATCGTGCAGCTAGC--CTAG-GAGCATCGAC
 ACGATCGAGCGACCGACGATCGAGCAGCTAGCGCCTAG-GAGCATCGAC

ACGATC-AGCGACCGACGATCGAGCAGCTAGCGCCTAG-GAGCATCGAC
 ACGATC-AGCGACCGACGATCGTGCAGCTAGCGCCTAGCGAGCATCGAC

ACGATG-AGCGATCGACGATCGAGCAGCTAGCGCCTAGCGAGC-----C
 ACGATCGAGCGATCGACGATCGAGCAGCTAGCGACTAG-GAGCATCGAC

ACGATCGAGCGATC-ACGA----GCAGCTAGCGA----GAGCATCGAC
 ACGATCGAGCGATCGACGATCGAGCAGCTAGCGACTAG-GAGCATCGAC

Jumping alignment

- Application: From which species an unknown sequence U is from?
- An MSA represents the *pangenome* of a species
- You can test U against several MSAs to decide the most likely source.

A ACGATCGAGCGATCACGATGAGCAGCTAGCACTAGCGAGCATCGAC

ACGATC-AGCGATCGACGATCGTGCAGCTAGCGACTAGCGAGCATCGAC
ACGATC-AGCGATCG**ACGAT-GAGCAGCTAGC-ACTAGCGAGCATCGAC**
ACGATC-AGCGATCGACGATCGTGCAGCTAGC-ACTAGCGAGCATCGAC
ACGATG-AGCGATCGACGATCGAGCAGCTAGC-ACTAGCGAGCATCGAC

k=1

B **ACGATCGAGCGATC**-ACGATCGAGCAGCTAGC-ACTAG-GAGCATCGAC
ACGATGGAGCGACCGACGATCGTGCAGCTAGC--CTAG-GAGCATCGAC
ACGATCGAGCGACCGACGATCGAGCAGCTAGCGCCTAG-GAGCATCGAC
ACGATC-AGCGACCGACGATCGAGCAGCTAGCGCCTAG-GAGCATCGAC
ACGATC-AGCGACCGACGATCGTGCAGCTAGCGCCTAGCGAGCATCGAC
ACGATG-AGCGATCGACGATCGAGCAGCTAGCGCCTAGCGAGC-----C
ACGATCGAGCGATCGACGATCGAGCAGCTAGCGACTAG-GAGCATCGAC
ACGATCGAGCGATC-ACGA----GCAGCTAGCGA----GAGCATCGAC
ACGATCGAGCGATCGACGATCGAGCAGCTAGCGACTAG-GAGCATCGAC

Sequence to graph alignment

We saw how to align paths of two DAGs, but what if our graphs have cycles?


Sequence to graph alignment

- Input: Sequence A and a labeled directed graph G
- Output: Min edit distance $D(A,P)$ over all paths P of G
- Trivial solution:
 - Enumerate all paths of length at most $2|A|$, compute the edit distance with A, and pick the minimum.
 - Longer paths cannot have better alignments
 - Exponential time
- How to avoid the enumeration?
 - Compute values $d(i,v)$ that give (in the end) the edit distance for aligning $A[1..i]$ to a path ending at node v
 - $$d(i,v) = \min \begin{cases} d(i-1,v') + \delta(A[i], \ell(v)), (v',v) \in E \\ d(i-1,v) + 1 \\ d(i,v') + 1, (v',v) \in E \end{cases}$$
 - At each row i, we compute values $d(i,v)$ ignoring insertions (last case, cyclic dependency)
 - One can see that the minimum $d(i,v)$ at row i cannot be improved by insertions, so these values are final
 - We propagate these final values to their out-neighbors
 - These neighbors can then be seen to have their final values, and we can proceed identically, until all values at the row are final
 - At each row we may need to visit each edge, so the running time is $O(|E||A|)$, if we are able to maintain the correct order of propagations efficiently
 - See course book (2nd edition) for correctness proof and details about the data structure needed for propagations (double-linked list of double-linked lists)
 - Let us now simulate this algorithm on some small input to gain some insights

Sequence to graph alignment

Black=before insertion propagation

Red = after insertion propagation



		A	C	C	G	T
	0	0	0	0	0	0
A	1	00	11	11	11	11
C	2	11	00	11	22	22
C	3	22	11	00	21	32
G	4	33	22	11	00	21
T	5	44	33	21	11	00
C	6	55	44	00	21	11
A	7	66	55	11	11	22