Introduction

In this section:
(≈ Sections 14.1–14.3 in Gusfield)

1. Definition of multiple alignment
2. Biological motivation
3. Aligning strings with a profile

Definition

A (global) multiple alignment (globaali monirinnastus) of \( k > 2 \) strings is an obvious generalization of two-string alignments:
Insert spaces into the strings to make them equally long (say, \( l \) chars), and arrange them in \( k \) rows and \( l \) columns, each character or space in a unique column

Example: A multiple alignment of strings
\[
\begin{align*}
\text{a} & \quad \text{b} & \quad 
\text{c} & \quad \text{a} \\
\text{a} & \quad \text{b} & \quad \text{a} & \quad \text{b} \\
\text{a} & \quad \text{c} & \quad \text{b} & \quad \\
\text{c} & \quad \text{b} & \quad \text{c} & \\
\end{align*}
\]

Local Multiple Alignments

There are also local multiple alignments
A local multiple alignment of strings \( S_1, \ldots, S_k \) consists of selecting a substring \( S_i^0 \) of each \( S_i \), and aligning these \( k \) substrings globally

We restrict now to global multiple alignments

Motivation

Multiple alignment is a specific formalization of multiple string comparison, which is an important methodology and active research area in bio-sequence analysis; It is used for
- extracting and representing biologically important commonalities from a set of strings
  - which might go unnoticed if only two strings were compared
- inferring evolutionary history from DNA or protein sequences

Family Representations

Found commonalities are used to characterize (and to understand) families of proteins
- family: set of proteins related by structure, function, or evolutionary history, e.g., globins and immunoglobulins

“This classification is central to our understanding of how life has evolved, and makes elucidation and definition of such families one of the principal concerns of molecular biology”
(Cyrus Chothia, Nature, 1992)

Usefulness of Compact Representations

Family representations are useful: It’s been estimated that the \( \sim 100,000 \) human proteins could be organized in about thousand families (or even a few hundred only)

A new sequence can be tested for potential membership in a family by comparing it with a family representation

Commonly used forms: profiles, consensus sequences, and signatures
- all derived from multiple string comparison

Profiles as Family Representations

A profile of a multiple alignment \( \mathcal{M} \) with row-length \( l \) is a \([|\Sigma| \times l] \) matrix \( p \), where \( p(y, j) \) is the occurrence frequency that char \( y \) occurs in column \( j \) of \( \mathcal{M} \). \( \Sigma = \Sigma \cup \{\_\} \)

Example: The profile of the previous multiple alignment
\[
\begin{array}{c|ccccc}
p & 1 & 2 & 3 & 4 & 5 \\
\hline
\text{a} & .75 & .00 & .25 & .00 & .50 \\
\text{b} & .00 & .75 & .00 & .75 & .00 \\
\text{c} & .25 & .25 & .50 & .00 & .25 \\
\_ & .00 & .00 & .25 & .25 & .25 \\
\end{array}
\]

How to compare a string and a profile?
Aligning a String to a Profile

A profile \( p \) is a sequence of columns \( \cdots \) we can align a string \( S \) with \( p \), by inserting spaces in them:

**Example:**

<table>
<thead>
<tr>
<th>( y' )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>0.75</td>
<td>0.00</td>
<td>0.25</td>
<td>0.00</td>
<td>0.50</td>
</tr>
<tr>
<td>( b )</td>
<td>0.00</td>
<td>0.75</td>
<td>0.00</td>
<td>0.75</td>
<td>0.00</td>
</tr>
<tr>
<td>( c )</td>
<td>0.25</td>
<td>0.25</td>
<td>0.50</td>
<td>0.00</td>
<td>0.25</td>
</tr>
<tr>
<td>( _ )</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>( S' )</td>
<td>( a )</td>
<td>( a )</td>
<td>( b )</td>
<td>( _ )</td>
<td>( b )</td>
</tr>
</tbody>
</table>

How to score a string/profile alignment?

**Scoring a String/Profile Alignment**

Common approach: (1) The score \( S(x, j) \) of a char \( x \) aligned with a column \( j \) is the average of the pair-wise character scores btw \( x \) and any character at col \( j \):

\[
S(x, j) = \sum_{y \in \Sigma} s(x, y) \times p(y, j)
\]

(2) score of the full alignment = sum of column scores

**Example:**
Assume character scores \( s(a, a) = 2 \), \( s(a, b) = s(a, c) = -1 \), and \( s(a, c) = -3 \)

The first column of the previous alignment adds \( S(a, 1) = 0.75 \times 2 + 0.25 \times (-3) \) to the total score, and the second \( S(a, _) = 1.0 \times (-1) \)

Computing an Optimal String/Profile Alignment

An optimal string/profile alignment can be computed as a straight-forward extension of string/string alignments

Let \( V(i, j) \) denote the value of an optimal alignment of prefix \( S[1 \ldots i] \) with columns \( 1 \ldots j \) of profile \( p \)

**Recurrences:**

**Base cases:**

\[
V(0, j) = \sum_{k=1}^{j} S(\_ , k) \quad (\text{\_} \text{ against } j \text{ first columns of } p)
\]

\[
V(i, 0) = \sum_{k=1}^{j} s(S[i], \_ ) \quad (S[1 \ldots i] \text{ against spaces})
\]

**NB** \( S \) (avg against column) vs. \( s \) (char-char score)

Computing Multiple String Alignments

Overview of this section:

1. Sum-of-pairs multiple alignment
   - Exact solution in exponential time
   - Bounded-error approximation in polynomial time
2. Practical approaches (heuristics)

Computing Multiple Alignments

Consider computing optimal multiple-string alignments

What is the score or goodness to be optimized?

There is no universally accepted way to score multiple alignments

- Many methods in practice compute multiple alignments without any formal guarantee of their quality!

The sum of pairs score (pari summa pairi steytys) is one simple and often used measure for the quality of multiple alignments

- Applied, e.g., in a subtask of the MACAW multiple alignment program

The Sum-of-Pairs Score

The sum of pairs score is defined as follows:

The **induced pairwise alignment** of two strings \( S_i \) and \( S_j \) participating in an alignment \( M \) is obtained by ignoring all other rows of \( M \) except those of \( S_i \) and \( S_j \)

The **score** of this induced alignment is the two-string alignment score btw the rows of \( S_i \) and \( S_j \)

- (the score of a space/space alignment is taken to be 0)

The sum of pairs (SP) score of a multiple alignment \( M \) is the sum of all pairwise alignment scores induced by \( M \)

Example of the SP Score

**Example:** Consider a multiple alignment \( M \) of strings \( S_1 = \text{AAGAAA}, S_2 = \text{ATAAATG}, \) and \( S_3 = \text{CTGGG} \):

\[
S_1: \quad A \quad A \quad G \quad A \quad _ \quad A
S_2: \quad A \quad T \quad _ \quad A \quad A \quad T \quad G
S_3: \quad C \quad T \quad G \quad G \quad G \quad G
\]

With standard edit distance the SP score of \( M \) is

\[
D(S'_1, S'_2) + D(S'_1, S'_3) + D(S'_2, S'_3) = 4 + 5 + 5
\]

- The results to be discussed apply (weighted) edit distance as the two-string alignment score, and thus minimize the score
Solving the SP Alignment Problem

Given a set of strings \( \{S_1, \ldots, S_k\} \), the SP alignment problem is to compute a global alignment that has a minimal sum-of-pairs score.

The SP alignment problem can be solved exactly with dynamic programming, in time \( O(3^w n^2) \) (if \( n = |S_i| = \cdots = |S_k| \))

\[ \cdots \]

\[ \text{-- impractical for more than 4 protein-length strings (of a few hundred characters)} \]

Exponential time seems inevitable, since the problem is known to be NP-hard.

Exact Solution for SP Alignment

Let’s sketch the main ideas of a dynamic programming algorithm for computing optimal SP alignment of strings \( A[1 \ldots n_1], B[1 \ldots n_2], \text{and } C[1 \ldots n_3] \)

Gusfield (Sec. 14.6.1) gives full pseudocode without much explanation.

To simplify notation, assume that the score of any char/space alignment is \( d \).

The method fills an array \( D(i, j, k) \) of size \( (n_1 + 1) \times (n_2 + 1) \times (n_3 + 1) \), where

\[ \text{If } D(i, j, k) \text{ is the optimal SP score of aligning the prefixes } A[1 \ldots i], B[1 \ldots j] \text{ and } C[1 \ldots k] \]

SP Alignment Recurrences

Base cases:

\[ D(0,0,0) = 0 \] is rather obvious

What about the initial boundaries of \( D(i,j,k) \) with \( i = 0, j = 0, \text{ or } k = 0? \)

Consider \( D(i,j,0) \): What is the SP score for an optimal alignment of \( A[1 \ldots i], B[1 \ldots j], \text{and } C[1 \ldots 0] \)?

\[ \text{A: } D_{A,B}(i,j) + (i + j) \times d, \text{where } D_{A,B}(i,j) \text{ is the edit distance of } A[1 \ldots i], B[1 \ldots j] \]

Formulas for \( D(0,0,k) \) and \( D(0,j,k) \) are similar.

Computing Inner Cells of \( D(i,j,k) \)

The values of inner cells \( D(i,j,k) \) with \( i, j, k > 0 \) depend on seven adjacent cells \( D(i',j',k') \) \( \text{where } i' = i - 1, j' = j - 1 \) or \( k' = k - 1 \)

The optimum for \( D(i,j,k) \) is the minimum of the seven cases sketched below:

First, an optimal alignment of \( A[1 \ldots i], B[1 \ldots j], \text{and } C[1 \ldots k] \) can align the last chars of each, giving score

\[ D(i-1,j-1,k-1) + s(A[i], B[j]) + s(A[i], C[k]) + s(B[j], C[k]) \]

Speedup Heuristics

Computation of value \( D(n_1, n_2, n_3) \) can be considered also as finding a shortest path from cell \((0,0,0)\) to \((n_1, n_2, n_3)\)

Standard path finding heuristics (Branch-and-bound, \( A^* \)) can be applied to prune the search space (that is, cells evaluated)

Such optimizations are applied in the multiple sequence alignment program called MSA.

\[ \text{reported to align six strings of } \sim 200 \text{ characters in a “practical” amount of time} \]

A Bounded-Error Approximation

Next, a polynomial-time method for approximating an optimal SP-alignment (Gusfield, 1993)

Gusfield’s method (like also some heuristics of practice) is based on extending an alignment by one string at a time

Key idea: When aligning a set of strings \( S = \{S_1, \ldots, S_k\} \), concentrate to optimizing \( k - 1 \) pairwise distances given in the form of a tree.

The tree used in Gusfield’s method is a center star (to be explained).

Alignments Consistent with a Tree

Let \( T \) be a tree (connected acyclic graph) that has strings \( S_1, \ldots, S_k \) as its nodes

A multiple alignment \( M \) of \( S = \{S_1, \ldots, S_k\} \) is consistent with \( T \) if the score of the induced pairwise alignment of \( S_i \) and \( S_j \) equals \( D(S_i, S_j) \) whenever \( (S_i, S_j) \) is an edge of \( T \)

\[ \text{That is, strings that are adjacent in the tree are aligned in a pairwise optimal way (while the induced score of other pairs does not matter)} \]
Denote the SP-score for alignment $S$ and for the optimal alignment $\mathcal{M}$. Inequality and $\mathcal{M}$.

**Main result:**

Let $\mathcal{M}^*$ be an optimal multiple alignment of $S$. Denote by $d^*(S, S_i)$ the score of the pairwise alignment of $S_i$ and $S_j$ induced by $\mathcal{M}^*$. Denote the SP-score for alignment $\mathcal{M}$, by $d(\mathcal{M}) = \sum_{i \neq j} d(S_i, S_j)$ and for the optimal alignment $\mathcal{M}^*$ by $d(\mathcal{M}^*) = \sum_{i \neq j} d^*(S_i, S_j)$.

Main result: **Theorem 14.6.2:** $d(\mathcal{M}_c) < 2d(\mathcal{M}^*)$.

**Proof.** Consider the ratio $d(\mathcal{M}_c)/d(\mathcal{M}^*)$ in terms of expressions that count each induced distance twice: $v(\mathcal{M}_c) = \sum_{i \neq j} d(S_i, S_j)$ and $v(\mathcal{M}^*) = \sum_{i \neq j} d^*(S_i, S_j)$. For $\mathcal{M}_c$ and $\mathcal{M}^*$, different zero weight spaces inserted in $S_i$ against spaces in $S_i$ -- this alignment has score $D(S_i, S_i')$.

**Theorem 14.6.1** Given a set of strings $S = \{S_1, \ldots, S_n\}$ and a tree $T$ made of strings of $S$, we can efficiently compute a multiple alignment $\mathcal{M}$ of $S$ that is consistent with $T$.

**Proof.** (Sketch) First compute an optimal alignment (with distance $D(S_i, S_j)$) for some pair of strings $S_i$ and $S_j$ that are adjacent in the tree.

Until all strings have been aligned, select two strings, $S_i$ in $\mathcal{M}$ (possibly with inserted spaces) and $S'$ not yet aligned, s.t. $(S_i, S')$ is an edge of $T$. Align $S_i$ and $S'$, assigning zero weight for spaces inserted in $S'$ against spaces in $S_i$.

**Center Star Method**

Tree-based alignment is applied to approximating an optimal alignment, by using a tree where other strings are connected to a suitably selected center string.

Define the consensus error (konsensusvirhe) of a string $S$ relative to a set of strings $\mathcal{S}$ to be $E(S) = \sum_{S \in \mathcal{S}} E(S, S')$ for all $S \in \mathcal{S}$.

- A center string can easily be found in polynomial time.
- A center star is a tree that consists of an edge btw the center string and each other string of $S$.

**Approaching Optimal Alignment**

**Lemma 14.6.1**

If the char scores satisfy the triangle inequality, then

$$d(S, S_j) \leq d(S, S_i) + d(S_i, S_j) \quad (1)$$

and

$$D(S, S_j) = D(S, S_i) + D(S_i, S_j) \quad (2)$$

for any $S_i, S_j \in S$.

**Proof.** (1) follows because the triangle equality holds at each column of the alignment of $S_i$, $S_j$, and $S_j$.

(2) holds because the alignment $\mathcal{M}_c$ is consistent with the center star.
Proof of the error bound (2)

On the other hand, \( v(M^*) \) can be estimated downwards:

\[
v(M^*) = \sum_{i,j} d^*(S_i, S_j) \\
\geq \sum_{i,j} D(S_i, S_j) - \sum_{i,j} D(S_i, S_j) \\
- \sum_i E(S_i) \geq \sum_i E(S_i) - k E(S_i)
\]

Therefore,

\[
d(M_i)/d(M^*) = v(M_i)/v(M^*) \\
\leq 2(k-1)E(S_i)/kE(S_i) = 2 - 2/k
\]

Remarks on Center-Star Approximation

The SP score \( d(M_i) \) of a center-star alignment could be much less than twice the optimal score

- in limited tests \( d(M_i) \) deviated from the optimal SP score by 2–16% only

Also a polynomial time approximation scheme has been developed for the SP alignment problem

A method of (Bafna, Lawler & Pevzner, 1994) produces a multiple alignment of \( k \) strings with SP score \( \leq 2 - q/k \) times the optimum;

Accuracy of the approximation can be increased, but that also increases the running time (as a function of \( q \))

Commonly Applied Heuristics

Numerous multiple alignment methods and programs are used in practice, and results produced by them are reported in hundreds of papers

Bounded-error heuristics are not widely used in practice for multiple alignment

Most methods apply variants of two ideas: iterative pairwise alignments and finding motifs common to the strings

Methods that apply iterative pairwise alignments build an alignment by iteratively merging alignments of two subsets of the strings together

Iterative Pairwise Alignments

A simple example: Compute the edit distances btw all strings, and then by aligning the most similar pair of strings

Then repeatedly align the closest pair of strings \((S_i, S_j)\) s.t. one of them is in the alignment and the other is not (as in the proof of Th. 14.6.1)

--- The method computes an alignment that is consistent with a minimum spanning tree (wrt the pairwise edit distances)

Iterative Alignments and Clustering

Other variants merge larger sub-alignments together

- merging could happen by aligning some representation (profile, consensus sequence) of the sub-alignments

Methods can be seen as applications of ideas from clustering (klusterointi, ryvästys)

Which of the variants are superior is a difficult question to answer

Repeated-Motif Methods

The second major category of heuristics first find a motif common to many strings of \( S \)

- a substring or a small similar subsequence

aka anchor, core, block, region, q-gram etc

When a “good” motif (wide and common to many strings) is found, strings containing it are shifted s.t. occurrences of motifs are aligned against each other

Then substrings on each side of the motifs are aligned recursively; When no good motifs are found, remaining subproblems are solved by iterative alignment

Repeated-Motif Methods (2)

Strings not containing motifs are aligned separately, and the two sub-alignments are finally merged together

For example, the MACAW program locates motifs at the top level, but aligns strings between motifs applying SP score

Again, there are numerous ways to realize these ideas, and it is hard to justify which are the best