The Multiple Sequence Alignment Problem in Biology - *paper summary*

Algorithms for Bioinformatics, FCUP

António Almeida, up201505836

Introduction

- Character sequence comparison is useful in multiple fields
  - Speech Recognition
  - Computer Science
  - **Molecular Biology**
- In molecular biology, through **sequence alignment**, some use cases include:
  - Comparing generic protein sequences
  - Construction of evolutionary trees based on sequence data
  - Protein engineering - **multiple sequence alignment**
- How?
  - Dynamic programming methods
  - Despite their great rise in complexity with dimension, they've proven to be useful in solving optimization problems associated with sequence comparison

Goals

- Give an introduction to the problem of multiple sequence alignment
- Contextualize it on the field of biology
- Formally define key concepts related to the problem
    - Sequences
    - Alignment
    - Paths
    - Optimal Paths
- Make observations on these concepts with the purpose of reducing the computational costs of the dynamic programming procedures used on multiple sequence alignment

Sequences and Alignments

Assuming an alphabet of characters - $\alpha = \{\alpha_1, \ldots, \alpha_n\}$. For simplicity, α is the set of natural numbers.

A ***sequence*** of k characters, or a *k-sequence*, is a set of the form $S = (\alpha_n 1, \ldots, \alpha_{nk})$

Alongside α, let's consider β, such that $\beta = \{-\} \cup \{\alpha_1, \ldots, \alpha_n\}$, i.e., adding the blank character "-" to α

An ***alignment*** of the sequences $S_1, \ldots, S_n$ is another set of sequences $\overline{S_1}, \ldots, \overline{S_n}$ such that each sequence $\overline{S_i}$ is obtained from S_i by inserting blanks in positions where some of the other sequences have a nonblank character.

Each alignment of the sequences $S_1, \ldots, S_n$ **encodes a set of different insertions, deletions, and replacements that transform one sequence into another**.

Paths

To any given set of N sequences $S_1, \ldots, S_n$ we'll associate a *lattice* $L(S_1, \ldots, S_n)$ in N-dimensional space.

This lattice consists of the N-dimensional hypercubes resulting from the Cartesian Product of N strings.

Each string represents a particular sequence. *Number of Squares = Number of Characters in Sequence*

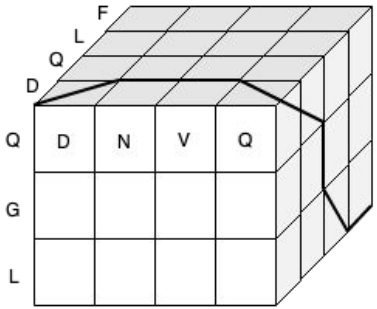
A *path* $\gamma(S_1, \ldots, S_n)$ between the sequences $S_1, \ldots, S_n$ is a "line" **joining the original corner to the end corner of the lattice**.

A projection of the path, $\mathrm{p}_{ij}(\gamma(S_i, S_j))$, represents an alignment of the sequences S_1 and S_2.

Example

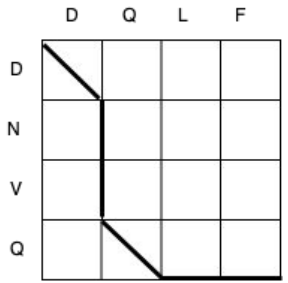
The following figure is an illustration of the notion of an ***alignment***, a ***lattice***, a ***path***, and a ***path projection***, considering the amino acid sequences:

$$S_1 = DQLF$$
$$S_2 = DNVQ$$
$$S_3 = QGL$$



- $\gamma(S_1, S_2, S_3)$

$p_{ij}(\gamma(S_1, S_2)) = D\_\_Q\_LF$
$p_{ij}(\gamma(S_2, S_3)) = DNVQ\_\_\_$
$p_{ij}(\gamma(S_1, S_3)) = \_\_\_QGL\_$



$p_{ij}(\gamma(S_1, S_2))$

Optimal Paths

If we define set of scoring rules that penalize each individual deletion, insertion, or replacement, then we can assign a **score** to a given alignment. There are multiple way to establish scoring rules.

The scoring rules allow us to assign to any given path $\gamma(S_1, \ldots, S_n)$ a *measure*, $M(\gamma)$. It measures (pun intended) the similarity among the sequences $S_1, \ldots, S_n$ according to the alignment associated with γ.

Corresponding to each $M(\gamma)$, there is at least one path, $\gamma^*(S_1, \ldots, S_n)$ such that M is minimum at γ^*.

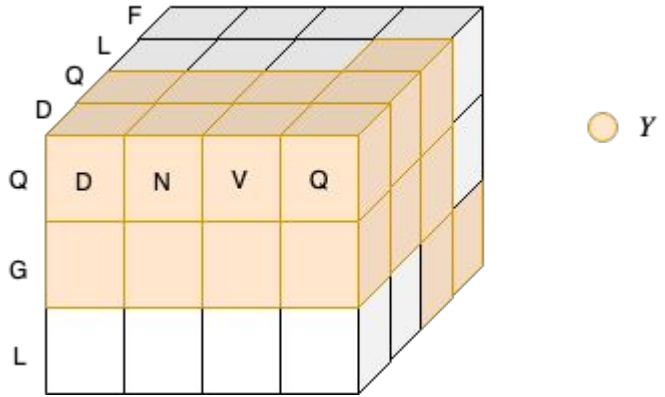
Such path is called an *optimal path*. Optimal paths in a lattice $L(S_1, \ldots, S_n)$ can be found by dynamic programming methods.

Results

The authors demonstrate we can define a set of paths X as the only possible candidates to be an optimal path, based on formally defined upper bound for $M(\gamma)$.

This means we need only to run the dynamic programming procedures on its equivalent subregion, Y

This can greatly reduce amount of computation necessary to find an optimal path, depending on the size of Y.



Results

Further computational improvements may be achieved by noting that not all points in the subregion Y must be stored in the computation of an optimal path.

The relationship between the alignment of N sequences and the problem of aligning subsets of these sequences can be exploited to develop multiple alignment algorithms with reduced computational requirements. Said arguments would be a function would be a function of the size of Y.

The authors compare their method to Ukkonen's, and conclude the latter is more efficient when the measure of distance is very simple, yet it becomes quite complicated and less efficient with more realistic measure of distance, while theirs not.

Conclusion

The paper is of great historic reference and does a great job at formally defining and exploring key concepts on the subject of multiple sequence alignment.

It’s observations are straightforward, despite the heavy reliance on mathematical expressions throughout their methods, given the very generic approach.