UROP DNA Sequence Alignment (Densmore Lab) with CIDAR ICE and Benchling Technologies using the Needleman-Wusnch Heuristic

Jason Lu

June 18th - 22nd 2017

BOSTON UNIVERSITY

1 References / Sources (for the research):

2 Collaboration:

BU CIDAR WETLAB TEAM

Professor Douglas Densmore (dougd@bu.edu)
Marliene Pavan Rodrigues (mapavan@bu.edu)
Jingxuan (Eric) Wu (jxwu@bu.edu)
Blade Olson (bladeols@bu.edu)
Luis Ortiz (lortiz15@bu.edu)

3 Introduction / Abstract:

This report (that is based on the majority of bio-informatic field) is used to show how to apply a computer science dynamic programming algorithm (Needleman-Wunsch) onto a real-world problem, like solving the similarities, and scoring the "differences" between two very similarly built strains of Escherichia coli. The algorithm uses the concept of dynamic programming to build up top-down a cubic-time solution ($O(n^3)$), and one would use strong induction on the length of the sequences to determine the "score" between two or more similar strains of a biological specimen like E. coli. The algorithm can also be credited as a

biological way to calculate sequence alignment and edit-distance (mutations) between the two strains as well.

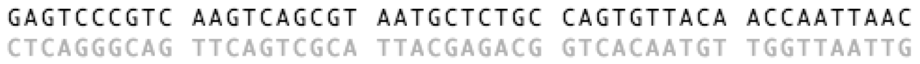
4 Motivation / Why Use This Algorithm:

Motivation:

Suppose the CIDAR LAB requires that we determine two different strains of bacteria that have the same genetic makeup. Using genomics, we are able to isolate the difference between the two strains onto two distinct parts:

Note: the following A,C,T,G genetic sequences are from Mary Pavan's (mapavan@bu.edu) CIDAR ICE workbook, from the strain with CIDAR 000990 **part id:**

Strain 1 on the top, Strain 2 on the bottom:



```
GAGTCCCGTC AAGTCAGCGT AATGCTCTGC CAGTGTTACA ACCAATTAAC
CTCAGGGCAG TTCAGTCGCA TTACGAGACG GTCACAATGT TGGTTAATTG
```

At first glance, it would be extremely difficult to determine whether the two strains can be considered compatible or not. However, there is a heuristic from Computer Science Algorithm Theory that would be able to solve this type of problem.

The algorithm in question is called the Needleman-Wunsch sequencing heuristic.

It uses the concept of dynamic programming to compare biological sequences, when then in turn can be used by micro-biologists (like Mary) to infer how similar the two strains of DNA are. A real-life example used in this very CIDAR LAB would be the comparison of two different strains of Escherichia coli, which is a deadly disease that has affected many people, and is very difficult to distinguish solely on the naked eye. One would need to use such an advanced computer heuristic to determine which strain is the more prohibitively dangerous one.

5 Needleman-Wunsch Algorithm: At the High Level

Notation (to understand what the pseudo-code of the Needleman-Wusnch Algorithm (in Python format) is trying to do / clarify any confusing symbols in the pseudo-code:

F(i,j) = the F-matrix on the i^{th} row and the j^{th} column

A_i = the i^{th} index (starting from 0) on the Sequence A alignment, similar to that of counting over the index of a string

B_j = the j^{th} index (starting from 0) on the Sequence B alignment, similar to that of counting over the index of a secondary, similar string

Match: the sequence alignment from A matches *exactly* with the sequence alignment from B, in both lexicographic and topological ordering

Penalty for Match: +1 per character in sequence (positive penalty, adds onto the score instead of subtract)

Mismatch: the sequence alignment from A does not match with the sequence alignment from B

Penalty for Mismatch: -1 per character in sequence

Gap / Indel: a missing piece of DNA (denoted as a " $-$ ") that makes it indeterminate whether sequence A or sequence B can be compared or not. Usually, move onto the next available index for comparison, unless it is the final index. In that case, we look back at the previous available index for comparison

Penalty for Gap / Indel: -2 per character in sequence

Insert: a required function to mitigate the impact of a gap / mismatch between the two alignments

Delete: a required function to mitigate the impact of a gap / mismatch between the two alignments

S(A_i,B_j): the coordinate in the pointers matrix of index i and index j on both sequences A and B

d: edit-distance (Levenshtein distance) between sequence Alignment A and sequence Alignment B

Now, we present the algorithm, using Python's pseudo-code approach with if/else statements, for loops and function calls within the code:

Algorithm:

Computing the F-Matrix Algorithm:

```
def computeFMatrix(Alignment A, Alignment B, F(i,j), d):
     for i=0 to len(A):
          F(i,0) = d * i
     for j = 0 to len(B):
          F(0,j) = d * j
     for i=1 to len(A):
          for j=1 to len(B):
               Match = F(i-1,j-1) + S($A_i$,$B_j$)
               Delete = F(i-1,j) + d
               Insert = F(i,j-1) + d
               F(i,j) = max(Match,Insert,Delete)
```

Main Needleman-Wunsch Algorithm:

```
def NW(Alignment A, Alignment B, F(i,j)):
     Alignment A = " "
     Alignment B = " "
     i = len(A)
     j = len(B)
     while (i > 0 or j > 0):
     # Case 1: the index i on Alignment A matches the index j on
     # Alignment B (match)
     if (i > 0 and j > 0 and F(i,j) = F(i-1,j-1) + S(A_i,B_j)):
          Alignment A = A_i + Alignment A
          Alignment B = B_j + Alignment B
          i = i-1
          j = j-1
     # Case 2: the index i on Alignment A does not match
     # the index j on Alignment B (mismatch)
     elif (i > 0 and F(i,j) = F(i-1,j)+d):
          Alignment A = A_i + Alignment A
          Alignment B = "-" + Alignment B
          i = i-1
     # Case 3: the index i or j on Alignments A or B
     # is missing (gap/ indel)
     else:
          Alignment A = "-"| + Alignment A
          Alignment B = B_j + Alignment B
          j = j-1
```

6 Proof of Correctness (Strong Induction on the length of the sequence):

The easiest (and most feasible) way to prove that the above algorithm indeed is the most optimal for comparing biological sequences is through strong induction. We first introduce the basis and recursive cases by dynamic programming, and then use the formulas from the dynamic programming step to build the results required for the induction. So, essentially the proof of correctness is a two-step process: first build base cases / recursive cases via dynamic programming, and then use strong induction on the length of the sequences

Dynamic Programming Steps to Building the Base Case and Recursive Case for Needleman-Wunsch:

First, we need to introduce the notion of edit-distance, as a way to determine *how much* to change between the sequences A and B before they are exactly matched alike. The fewer the changes, the closer the strings are in terms of genetic sequencing, and with respect to the Needleman-Wunsch Algorithm, the higher the score is in terms of similarity

Then, based on the above definition and general implementation of the edit-distance algorithm (as a subroutine to NW), we can finally build the base cases for the strong induction proof of correctness:

Proof (by strong induction on the length on max(len(sequence A, sequence B)) [the longer of the two sequences]:

Base Case (only 1 letter / character in each sequence A and B:

7 Running Time and a $O(n^2)$ Memoized Solution:

8 Conclusions / Remarks / Acknowledgements:

9 Extra Application: Using the NW Heuristic to determine sequence accuracy in the CIDAR C0040_CD and C0080_CD Benchling genetic sequences (C0040_CD has 2765 sequence characters, while C0080_CD has 3020 sequence characters)

Note: due to the sequence length being in the 1000s for both C0040_CD and C0080_CD projects, the algorithm may result in a slight inaccuracy in terms of the match : mismatch ratio percentage (floating point precision error due to Python integer division)

Image 1: Replicated Plasmid Sequence of C0080_CD sequence from CIDAR-LAB Benchling site (credit: Luis Ortiz (lortiz15@bu.edu)

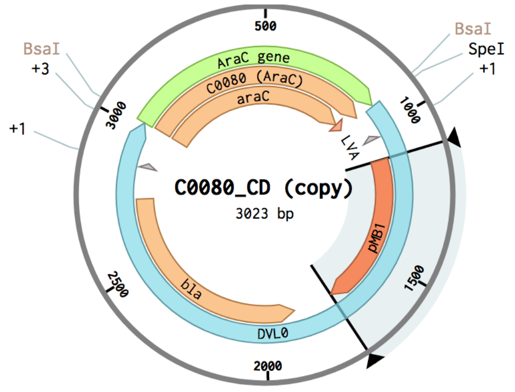


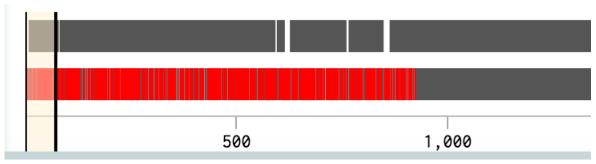
Image 2: Replicated Plasmid Sequence of C0040_CD sequence from CIDAR-LAB Benchling site (credit: Luis Ortiz (lortiz15@bu.edu)

Additional personal annotations: (just for reference for non-biological majors):

AraC gene: a gene / protein that is commonly used as an operon (a gene promoter) in the E. Coli bacteria cultures

We then look at a screenshot of the alignment between C0080_CD and C0040_CD copies, mostly from the first 100 base pairs. Using the algorithm, we add one point for each part of match, take away one point from each part of mismatch, and take away 2 points for each part of gap / missing sequence character. We then put the accuracy of the alignment as a ratio between $\frac{match}{mismatches}$ (since percentages can't be negative in this context) * $\frac{1}{total\ sequence\ characters}$, which will result in a ratio either greater to, equal to, or less than 1 (but not less than 0)

Naively, one would try and just match up the sequences one to one, which results in an ungainly situation where more than 1/4 of the C0040_CD sequence does not match up well to the C0080_CD sequence:



So, we would have to move the C0040_CD sequence (shift right) by at least 3020-2765 = 255 more base pairs in order to get a better approximation or a more accurate way to being able to see how well the two sequences match up to each other. In that case, we would have to start from approximately (256 * 2) + 50 = 562nd base pair on the second C0040_CD sequence (precisely, the 583rd base pair), and for the other one (256 * 3) + 50 = 818 base pair (842 precisely) to begin see how well the two sequences match up, or have a feasible estimate for the NW score.

Range of sampling:

C0080_CD: [583,641] (58 total base pairs)

C0040_CD: [842,900] (58 total base pairs)

For simplification, and higher chance of accuracy with smaller sample size (inferential statistics), we will segment each part of the C0040_CD and C0080_CD base pairs into 10 sequence characters at a time, and compute the Needleman-Wunsch score separately. Then, we apply the summation together, including adding additional penalites (+1 for match, -1 for mismatch and -2 for gap) before determining the entire Needleman-Wunsch score.