Project Description

Type of Input Data

Our input will be *structure-annotated sequence*, i.e., in practice each input item **a** is a pair of strings of the same length. The first string will be an amino-acid sequence, the seccond string its secondary structure annotation. Figure 1 in "Bi-Alignments with Affine Gaps Costs" shows and example

```
SeqA RAKLPLKEKKLTATANYHPGIRYIMTGYSAKYIYSSTYARFR
StrA CHHHHHHHHHHHHHCCCCTCEEEEEEECCTCEEEEEEEEECCC
```

We will use a similarity measure on the aminoacid sequences (e.g. the BLOSUM matrix, see e.g. wiki page, plus a (user defined) gap penalty) and a similarity measure of the secondary structure on the letters used in the secondary structure annotration, again with a gap penalty.

For the affine gap cost version we will of course need as a *gap open* and a *gap extend* parameter.

Finally, we need a single parameter to score the shift penalty, called Δ in the papers.

Suggestion. Read scoring models from a file in a simple format that can be edited easily.

Aims:

1. Implement a simple, linear-cost, non-optimized version for shift alignments.

   this is equ.(3) of "Bi-Alignments as Models of Incongruent Evolution ..." (2nd paper). Note that the variables in the recursion are 4-dimensional index vectors x and 4-dimensions (binary) patters.

   Don't waste time to optimize this, we only need it to check correctness.

   As a sanity check we also compute the pairwise alignment of our input objects (a) scored with only the sequence scoring function, (b) scored with only the structure scoring function, and (c) with the sum of the sequence and structure scores are scoring function. Note that algorithmically this is three times the very same algorithm, just with different scoring models. Of course the algorithm is the simple, linear-cost, Needlemann-Wunsch algorithm for pairwise sequence alignments.

   From these three pairwise alignments we can compute an upper bound on the number of shifts that we can possibly have: This bound is explained in "Incongruences Between Sequence and Secondary Structure Alignments of Nucleic Acids" Theorem 2/equ.(18). Use this to crosscheck the bialignment implementation!

   Note: we need a backtracing only for the bi-alignment algorithm, for the pairwise alignments it suffices to compute the scores to use equ.(18).

As an inspiration of for the output format you might want to use Fig.3 in "Bi-Alignments as Models of Incongruent Evolution ...", of course replacing the RNA dot-bracket notation by the protein secondary structure notation. The format is chosen such that the upper and lower blocks are the alignments with the first and second scoring model, and the two blocks are aligned according to the shifts. But feel free to experiment!

Note that the shifts can be directly read off the columns of the 4-way alignment, e.g. equ.(2.2) in "Bi-Alignments with Affine Gaps Costs". This is true in general.

2. Next we implement the affine gap cost version explained in "Bi-Alignments with Affine Gaps Costs", more precisely equ.(2.6).

   A few comments are in order:

   1. make sure you understand the notation: x and y are the 2-dimensional index that together form the index vector characterizing an alignment column. x refers to alignment w.r.t. to the first scoring function, y refers to the alignment w.r.t to the second scoring function. The p and q are the corresponding part of the end gap pattern. Note that they are also 2-D vectors.
   2. again we will need the three types of pairwise alignments (for each of the individual scoring models, and for the sum), only this time the alignments need to be computed with Gotoh's algorithm for affine gap costs.

3. Finally we want to optimize this code by avoiding the computation of the 4D dynamic programming matrix M. We known that in each alignment column the difference $|x_1 - y_1| + |x_2 - y_2|$ of the indices, i.e., the total "visible" shift at this column, cannot be larger than the maximal number of possible shits, for which we can compute an upper bound for given input data according to Theorem 2/equ.(18). The goal is therefore to restrict the memory requirements and the CPU requirements by only computing those entries in M that can be optimal, i.e., that have no more shifts than the pre-computed bound.

4. **Optional!** In principle one can get an even better bound by noticing that Theorem 2/equ.(18) also holds for the alignments of prefixes, i.e., for each index pair (i, j) in the DP matrices of the three pairwise alignments. This may then yield even better bounds. It is unclear whether this worth the effort or not.

5. Once we have a working version of point 3.:

   1. Test and benchmark for resource consumption. How does the performance depend on sequence and structure similarity?

2. Extract a text of Pfam protein families and from those say the coorresponding *E. coli* and *Xanthomonas* representative, compute their predicted secondary structures using web resources or downloadable tools (see webpage). Do we find evidence for incongruent evolution? This step will also need a bit of playing with the shift score Δ. Note that Theorem 2/equ.(18) will rule out shifts already without computing the bialignment itself. (how?) Obviously then there is not point in computing the bialignment.

   If we can expect shifts, compute the the corresponding bi-alignments.