Heuristic alignment algorithms

Using dynamic programming to compute similarity between two sequences will cost $O(n^m)$. With this cost, aligning a sequence against a database containing millions of sequences is not feasible. Because of that, there have been many attempts to produce faster algorithms. The goal of these algorithms is to search as small a fraction as possible of the cells in the dynamic programming matrix.

Two of the best-known algorithms are FASTA and BLAST. They work from the same basic idea, namely that – most sequences in the database don’t match, and because of that the algorithms apply some heuristic to exclude many of the unrelated sequences. If these heuristics are used there will be no guarantee that we will find the optimal scoring alignment.

General approach

Seeds: All (good ungapped matches) to subsequences of $X$ of a given length ($X$ is our query).

For each sequence $Y$ in the database

1. Search for seeds in $Y$
2. Extend alignment around seeds or partition the alignment problem
3. If a high scoring match in last step was found, then use dynamic programming around a good match

FASTA (Fast Alignment)

Seeds: All subsequences of length $ktup$. Typical values of $ktup$ are

- Proteins: 1-2
- DNA: 4-6

Store all seeds with their starting position in $X$ in a hashtable.

1. Using the hashtable we can now find all exact matches (hotspots) to seeds in $Y$. The running time will be linear in the length of $Y$: $O(|Y|)$
2. Chain hotspots into runs of hotspots on the same diagonal. This is done efficiency by sorting hotspots on (j-i). A run consists of one or more consecutive hotspots along a common diagonal (see fig. 1)

   a. Using a function that takes a set of hotspots and their distance to each other, we can find out how good the runs are. Score(run) = δ(hotspot, distance)

   b. Pick 10 high scoring runs:

   \[ R_1 = (α_1, β_1) \]
   \[ \ldots \]
   \[ R_{10} = (α_{10}, β_{10}) \]

   c. If \( \max\{S(α_i, β_i)\} \leq i \geq 10 \) is sufficient high we continue.

\[ \text{Fig. 1 – Dot plot} \]

   d. Now we can construct a weighted graph. If the longest path in this graph is sufficiently high we can continue to next step. The length of a path is the sum of the weights on the edges and the weights on the vertices of the path. (See fig 2)
3. Perform banded dynamic programming around the high scoring run (see fig. 3). The running time of this step is $O(|X| + |Y|)$, because ‘c’ is limited by a constant.

**Fig. 2 – Weighted graph**

**Fig. 3 – Banded dynamic programming**

**BLAST** (Basic Local Alignment Search Tool)

Word: subsequence of length d. Typical values of d are
- Protein: 3
- DNA: 11
Seeds in BLAST are longer than seeds in FASTA, but it isn’t necessary to find exact matches. Seeds: All words \( w' \) such that \( s(w, w') \geq t \) for some word \( w \) in \( X \).

1. Find exact matches to seeds in \( Y \)
2. Extend each match to maximal extension and report matches with score > \( c \)

To make this algorithm to a “gapped version” we need one more step.

3. Perform local dynamic programming around middle position \((x_i, y_i)\) of best match. Extend alignment in all directions stopping when score drops below a certain percentage of highest score so far (see fig 4).

*Fig. 4 – Local dynamic programming around middle position \((x_i, y_i)\) of best match*