CMSC423: Bioinformatic Algorithms, Databases and Tools Lecture 10

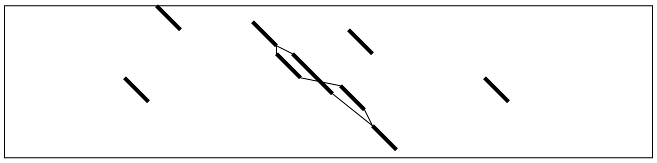
Sequence alignment: inexact alignment, multiple sequence alignment

Inexact alignment recap

- Affine gaps – need 4 matrices: global score, score of alignments ending in a match, score of alignment ending in a gap in seq1, score of alignment ending in a gap in seq2.
- In the "real" world, inexact alignment is performed only where necessary – heuristics pre-compute where an alignment is possible.
- Also, inexact alignment is easier if we bound the allowed error – only need to explore the neighborhood of the main diagonal in the DP matrix

Chaining approach

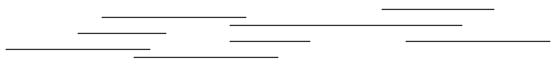
- Extends the FASTA idea
- Search for exact matches
- Find the longest consistent chain of exact matches
- Fill in the gaps in the chain using Smith-Waterman



- This is the approach used by MUMmer (Delcher et al.)

Chaining in 1-D

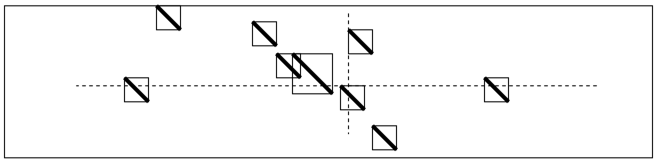
- Input: multiple overlapping intervals on a line
- Output: highest weight set of non-overlapping intervals
- Weight could be length of interval, or Smith-Waterman score, etc.



- Sort the endpoints (starts, ends) of the intervals
- For every interval j, store V[j] – best score of a chain ending in j
- MAX – store highest V[j] seen sofar
- Process endpoints in increasing order of x coordinate
- If we encounter left end (start) of interval j
  - V[j] = weight(j) + MAX
- If we encounter right end (end) of interval j
  - MAX = max{V[j], MAX}
- Running time?

Chaining in 2-D

- Easy to do in $O(n^2)$ (n - # of intervals)
- View alignments as "boxes"
- All boxes in a chain must follow each other in a "diagonal" order, i.e. the range of the x coordinates and y coordinates of any two boxes in a chain cannot overlap
- Similar to 1-D approach except at each step we must check if current box can extend any of the previously built chains
- $V[j] = \max_{\text{all previous boxes } k} \{V[k] + \text{weight}(j)\}$
- More complex algorithm leads to O(n log n) running time



Multiple sequence alignment

- Simultaneously identify relationship between multiple sequences

```
HBB_HUMAN     FFESFGDLSTPDAVMGNPKVKAHGKKVL-----GAFSDGLAHLDNLKGTF
HBB_HORSE     FFDSFGDLSNPGAVMGNPKVKAHGKKVL-----HSFGEGVHHLDNLKGTF
HBA_HUMAN     YFPHF-DLS-----HGSAQVKGHGKKVA-----DALTNAVAHVDDMPNAL
HBA_HORSE     YFPHF-DLS-----HGSAQVKAHGKKVG-----DALTLAVGHLDDLPGAL
MYG_PHYCA     KFDRFKHLKTEAEMKASEDLKKHGVTVL-----TALGAILKKKGHHEAEL
GLB5_PETMA    FFPKFKGLTTADQLKKSADVRWHAERII-----NAVNDAVASMDDTEKMS
LGB2_LUPLU    LFSFLKGTSEVP--QNNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATL
               *  :  .      . .:: *.  :      :.   :
```

- Note: multiple alignment implies (not necessarily optimal) pairwise alignment between the individual sequences

```
HBA_HUMAN     YFPHF-DLS-----HGSAQVKGHGKKVA-----DALTNAVAHVDDMPNAL
HBA_HORSE     YFPHF-DLS-----HGSAQVKAHGKKVG-----DALTLAVGHLDDLPGAL
```

Multiple alignment – formal definition

- M – multiple sequence alignment for $s_1,...,s_k$
- $D(s_i,s_j)$ – optimal score of alignment between s_i, s_j
- $d(s_i,s_j)$ – score of alignment btwn s_i, s_j induced by M
- score of M $d(M) = \mathrm{sum}_{\text{all pairs si, sj}}\, d(s_i, s_j)$
- also called sum-of-pairs

- Optimal multiple alignment minimizes d(M)

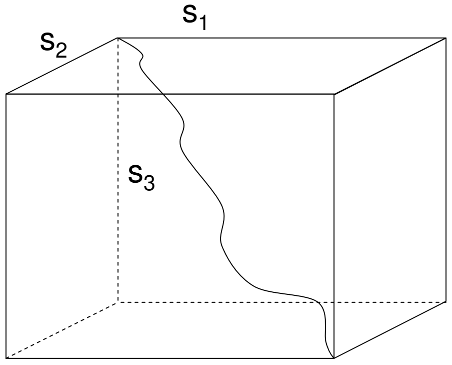
- Computing optimal d(M) is NP hard
- Note: in multiple alignment we think of "distance" rather than "similarity"

But....here's a solution

- Dynamic programming solution.   e.g. 3 sequences

- Score(i, j, k) – optimal alignment between s1[1..i], s2[1..j], s3[1..k] – do DP as usual

- s(i,j,k) = max {
        s(i-1, j-1, k-1) +
  match(s1[i], s2[j], s3[k]),
  ...



But... it's expensive

- 3 sequences – need to fill in the cube $O(n^3)$
- k sequences – k-dimensional cube $O(n^k)$ time/space
- There are tricks that can help – similar to AI techniques for reducing the search space
- Basic idea – if we can estimate optimal score, we can prune the search space.
- Note – these are just heuristics – not guaranteed to work faster

Alternative – approximation algorithm

- Can we efficiently compute a multiple alignment with a score that's not too bad?
- The Star method:
  - build all k^2 pairwise alignments ($O(k^2n^2)$)
  - pick sequence sc that is closest to all other sequences: sum $_{si}$ D(sc, s_i) is minimal over all choices of sc
  - iteratively align each sequence to sc
- Theorem: sum-of-pairs score of star alignment is at most twice as big as optimal multiple alignment score

Iterative alignment

```
SC YFPHFDLSHGSAQVKAHGKKVGDALTLAVGHLDDLPGAL
```

- Take sequences si in order:
  - align s1 with sc - results in gaps being inserted in both sequences

```
SC YFPHFDLSHGSAQVKAHGKKVGDALTLAVGHLDDLPGAL
S1 YFPHFDLSHG-AQVKG--KKVADALTNAVAHVDDMPNAL
```

  - align s2 with sc - if gaps must be inserted – insert in previously aligned sequences

```
SC YFPHF-DLS-----HGSAQVKAHGKKVG-----DALTLAVAHLDDLPGAL
S1 YFPHF-DLS-----HG-AQVKG—GKKVA-----DALTNAVAHVDDMPNAL
S2 FFPKFKGLTTADQLKKSADVRWHAERII-----NAVNDAVASMDDTEKMS
```

  - and so on (note: if gaps coincide with previously introduced gaps no need to change previously aligned sequences)

```
SC YFPHF-DLS-----HGSAQVKAHGKKVG-----DALTLAVAHLDDLPGAL
S1 YFPHF-DLS-----HG-AQVKG—GKKVA-----DALTNAVAHVDDMPNAL
S2 FFPKFKGLTTADQLKKSADVRWHAERII-----NAVNDAVASMDDTEKMS
S3 LFSFLKGTSEVP--QNNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATL
```

Theorem proof

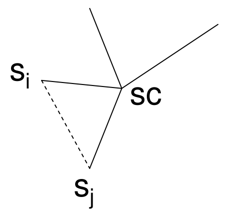
- Theorem: star alignment is 2-optimal
- Assumption: distances obey triangle inequality

OPT = $\sum_{si,sj} d^*(s_i,s_j) \geq \sum_{si,sj} D(s_i,s_j) \geq k \sum_{si} D(s_i, sc)$

STAR = $\sum_{si,sj} d(s_i,s_j) \leq \sum_{si} D(s_i, sc) + \sum_{sj} D(s_j, sc)$

$= 2k \sum_{si} D(s_i, sc)$

=> STAR/OPT ≤ 2 Q.E.D



Consensus sequence

- For every column j in the alignment, pick the amino-acid AA that minimizes $\sum_i d(AA, S_i[j])$ (usually becomes majority rule)
- Intuitively – this is the sequence of the ancestor of all the sequences in the multiple alignment
- We can define the multiple alignment problem as:
  - find the multiple alignment that minimizes $\sum_i D(CO, S_i)$
- Related to "Steiner" string problem:
  - find a string S* and a multiple alignment such that $\sum_i D(S^*, S_i)$ is minimal
- Both formulations are NP hard

```
CO YFPHFKDLS-----HGSAQVKAHGKKVG-----DALTLAVAHVDDTPGAL
S1 YFPHF-DLS-----HGSAQVKAHGKKVG-----DALTLAVAHLDDLPGAL
S2 YFPHF-DLS-----HG-AQVKG–GKKVA-----DALTNAVAHVDDMPNAL
S3 FFPKFKGLTTADQLKKSADVRWHAERII-----NAVNDAVASMDDTEKMS
S4 LFSFLKGTSEVP--QNNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATL
```

Iterative alignment revisited

- Pick a sequence (e.g. SC) as a starting point
- Align S1 to it & build consensus for the alignment
- Take S2 and align it to the consensus (instead of SC)
- repeat...
- Problem: consensus (or any single sequence) ignores the other sequences being aligned.
- Solution: keep track of % of each amino-acid aligned in each column
- score of alignment to profile – combination of scores to each AA.

100% F

50% S
25% N
25% -

75% A
25% Q

```
S1 YFPHF-DLS-----HGSAQVKAHGKKVG-----DALTLAVAHLDDLPGAL
S2 YFPHF-DLS-----HG-AQVKG–GKKVA-----DALTNAVAHVDDMPNAL
S3 FFPKFKGLTTADQLKKSADVRWHAERII-----NAVNDAVASMDDTEKMS
S4 LFSFLKGTSEVP--QNNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATL
```