Today’s Lecture

• Alignment algorithms
  – Smith-Waterman, Needleman-Wunsch

• Local vs global

• Computational complexity of pairwise alignment

• Multiple sequence alignment

• Improved scoring of pairwise alignments
  – Affine gap penalties
  – Profiles

• Smith-Waterman special cases
Above path corresponds to following alignment (w/ lower case letters considered unaligned):

```
aCGTTGAATGAccca
```
```
gCAT-GAC-GA
```
Alignment algorithms

- **Smith-Waterman** algorithm to find highest scoring alignment
  = dynamic programming algorithm to find highest-weight path
  - Is a *local* alignment algorithm:
    - finds alignment of subsequences rather than the full sequences.
- Can process nodes in any order in which parents precede children. Commonly used alternatives are
  - depth order
  - row order
  - column order
• If constrain path to
  – start at upper-left corner node and
  – extend to lower-right corner node,
get a **global** alignment instead

• This sometimes called *Needleman-Wunsch algorithm*
  – (altho original N-W alg treated gaps differently)

• ∃ variants which constrain path to
  – start on the left or top boundary,
  – extend to the right or bottom boundary.
Local vs. Global Alignments: Biological Considerations

- Many proteins consist of multiple ‘domains’ (modules), some of which may be present
  - with similar, but not identical sequence
  in many other proteins
  - e.g. ATP binding domains, DNA binding domains, protein-protein interaction domains ...

  Need *local alignment* to detect presence of similar regions in otherwise dissimilar proteins.

- Other proteins consist of single domain evolving as a unit
  - e.g. many enzymes, globins.

  Global alignment sometimes best in such cases
  - ... but even here, some regions are more highly conserved (more slowly evolving) than others, and most sensitive similarity detection may be local alignment.
3-D structures of rat Rab Geranylgeranyl Transferase complexed with REP-1, + paralogs.

adapted from Rasteiro and Pereira-Leal *BMC Evolutionary Biology* 2007 7:140
Multidomain architecture of representative members from all subfamilies of the mammalian RGS protein superfamily.

from www.unc.edu/~dsiderov/page2.htm
Similar considerations apply to aligning DNA sequences:

• (semi-)global alignment may be preferred for aligning
  – cDNA to genome
  – recently diverged genomic sequences (e.g. human / chimp)
  \textit{but} local alignment often gives same result!

• between more highly diverged sequences, have
  – rearrangements (or large indels) in one sequence vs the other,
  – variable distribution of sequence conservation,

& these usually make local alignments preferable.
Complexity

• For two sequences of lengths $M$ and $N$, edit graph has
  - $(M+1)(N+1)$ nodes,
  - $3MN+M+N$ edges,
• time complexity: $O(MN)$
• space complexity to find
  highest score and beginning & end of alignment
  is $O(\min(M,N))$
  (since only need store node’s values until children processed)
• space complexity to reconstruct highest-scoring alignment: $O(MN)$
• For genomic comparisons may have
  – $M, N \approx 10^6$ (if comparing two large genomic segments), or
  – $M \approx 10^3, N \approx 10^9$ (if searching gene sequence against entire genome);
  in either case $MN \approx 10^{12}$.
• Time complexity $10^{12}$ is (marginally) acceptable.
• $\exists$ speedups which reduce constant by
  – reducing calculations per matrix cell, using fact that score often 0
    • (our program $swat$).
    • still guaranteed to find highest-scoring alignment.
  – reducing # cells considered, using nucleating word matches
    • ($BLAST$, or $cross\_match$).
    • Lose guarantee to find highest-scoring alignment.
The *Edit Graph* for a Pair of Sequences
Multiple Alignment via Dynamic Programming

- **Higher dimension edit graph**
  - each **dimension** corresponds to a **sequence**; co-ordinates labelled by residues
  - Each **edge** corresponds to a **aligned column** of residues (with gaps).
  - Can put arbitrary weights on edges; in particular,
    - can make these correspond to probabilities under an evolutionary model (Sankoff 1975).
  - implicitly assumes independence of columns

- **Highest weight path through graph again gives optimal alignment**
Generalization to Higher Dimension

Each “cell” in 3-dimensional case looks like this:

Each edge projects onto a gap or residue in each dimension, defining an alignment column; e.g. red edge defines

V

—

M
The *Edit Graph* for a Pair of Sequences

The diagram illustrates an edit graph for a pair of DNA sequences. Each node represents a nucleotide (A, C, G, T), and the arrows indicate the transitions between nucleotides. The graph visually represents the possible edit operations (insertions, deletions, and substitutions) required to transform one sequence into another.
• # edges & # vertices are proportional to **product** of sequence lengths.
  – For $k$ sequences of size $N$, is of order $O(N^k)$
    • impractical even for proteins ($N \sim 300$ to 500 residues) if $k > 5$: $300^5 = 2.4 \times 10^{12}$
Multiple alignments: paths in huge WDAGs

- To find high-scoring paths, need to
  - reduce size of graph
  - restrict allowed weighting schemes, and/or
  - sacrifice optimality guarantees

- Durbin et al. discuss methods implementing these ideas:
  - Hein
  - Carillo-Lipman
  - progressive alignment (e.g. Clustal)

- HMMs provide nice (but not guaranteed optimal) approach for constructing multiple alignments
The *Edit Graph* for a Pair of Sequences
Better Scoring Models

- Optimal alignment scoring depends on probabilistic modelling (to be discussed later).
- Inherent limitation of dynamic programming: each alignment column (edge in WDAG) scored independently
  - biologically unrealistic, but
  - required for dynamic programming to work!
Two strategies to allow partial non-independence while preserving dynamic programming framework:

- Enhance graph
- Allow scores to depend on position within the sequence (i.e. not just on a BLOSUM-type score matrix)
  - so some substitutions (of same residues) or gaps penalized more heavily than others
• Usual scoring scheme assigns same penalty $g$ to each gap edge, so
  – weights on extended gaps of size $s$ are linear in $s$, i.e.
  – total gap penalty $\text{gap}(s) = s \times g$.
  – e.g. in above example, if each $g = -6$, total penalty on gap would be

\[
\text{gap}(5) = 5 \times -6 = -30
\]
Gap Penalties

- Would like more flexible gap penalties:
- In proteins, insertions & deletions are rare;
  - but when occur, often consist of several residues, because
    - they are in regions (loops) tolerant of length changes
  - at DNA level, indels in protein coding sequence usually a multiple of 3 nucleotides
    - otherwise, would change reading frame
- In noncoding sequence,
  - the most common indel size is 1
  - but larger indels occur much more frequently than multiple independent single-base indels
• Can allow arbitrary **convex** gap penalties
  
  – \( \text{gap}(s+t) \geq \text{gap}(s) + \text{gap}(t) \), where \( s \) and \( t \) are (integer) gap sizes

by extending edit graph:
  
  – add edges corresponding to **arbitrary length** gaps from each vertex to each horizontally or vertically downstream vertex
  
  – (convexity condition prevents favoring two adjacent short gaps over a single long gap).

**Time complexity now** \( O(MN(M+N)) \)
  
  – often unacceptable for moderate \( M, N \).
  
  – Also: how to choose appropriate weights? (need data to estimate!)
Affine Gap Penalties

• **Affine** gap penalties:
  – less general than arbitrary convex penalties, but
  – more general than linear penalties.

• Two parameters:
  – *gap opening* penalty $g_o$
  – *gap extension* penalty $g_e$

• $\text{gap}(n)$ (penalty for size $n$ gap) is then
  
  \[ g_o + n \cdot g_e = g_i + (n - 1) \cdot g_e \]

  where the gap *initiating* penalty $g_i = g_o + g_e$
• Example: for BLOSUM62, good penalties are
  – \( g_i = -12 \),
  – \( g_e = -2 \)
These perform *much* better than linear penalty
  – (e.g. \( g = -6 \))
• N.B. Durbin *et al.* reverse \( g_i \) and \( g_o \)
  – \( g_i \) is called the ‘gap opening’ penalty
• Can obtain affine penalties using extension of edit graph, retaining complexity \( O(MN) \):
Edit Graph for Affine Gap Penalties

Double # vertices, creating left-right pair in place of each original vertex. Each cell looks like this:

- gap-opening edges from left vertex to right vertex of each pair: weight $g_o$
- gap extension edges going horizontally or vertically between right vertices: weight $g_e$
- diagonal edges originate from either left or right vertex, but always go to a left vertex.

Each left vertex has out-degree and in-degree = 2
Each right vertex has out-degree and in-degree = 3
• Paths in the augmented graph still correspond to alignments
  – can ∃ more than one path for same alignment
  – but highest scoring paths still give best alignments

• Score assigned to size $n$ gap is $g_o + n g_e$
  – *i.e.* affine penalty

• Smith-Waterman-Gotoh algorithm
Profiles (position-specific scoring)
The *Edit Graph* for a Pair of Sequences
Profiles: Position-specific scoring scheme specifying score of each possible substitution at each position of a sequence.

From R. Luthy, I. Xenarios and P. Bucher, Improving the sensitivity of the sequence profile method Protein Sci. 3: 139-146 (1994)
• This is an important improvement!
  – reflects fact that different parts of sequence may evolve at different rates
• e.g. in proteins,
  – internal core region of tightly packed residues, or active sites of enzyme, are more highly conserved;
  – surface residues, particularly in loops, often less conserved.
  – so scores tend to be correlated (high scores in core, lower on surface)
Rates of amino acid exchange in mammalian proteins by burial status


H: hydrophobic
P: polar
• PSIBLAST approach:
  – initially compare query sequence to database sequences (using BLOSUM-type scoring matrix),
  – build profile using initial matches
  – rescan database using profile

• Optimal choice of
  – substitution matrix,
  – gap penalties, or
  – profiles

  depends on probabilistic modelling (to be discussed later!)
Smith-Waterman special cases

- Various special cases are optimal path problems for *subgraphs* of edit graph:

- *Gap-free* alignments correspond to paths confined to a diagonal of edit graph
  - (i.e. subgraph without horizontal & vertical edges).

- Find *perfectly* matching segments using weights
  +1 for identical residue pair,
  -∞ (or large negative penalty) for mismatches or gaps.

Less efficient than “sorting pointers” method from lecture 1 / HW1.
The *Edit Graph* for a Pair of Sequences
• Find *imperfect internal repeats* by searching edit graph of sequence against itself
  – i.e. the same sequence labels columns and rows
  *above (\& not including) the main diagonal*:
  – if include main diagonal, best path will be identity match to self
  – complexity = $O(N^2)$ where $N =$ sequence length.

Graph for finding imperfect internal repeats:
• Find *short tandem repeats* (e.g. microsatellites, minisatellites):
  – scan a *band* just above main diagonal.
  – Complexity = $O(kN)$ where $k$ is width of the band.
  – Manageable even for large $N$, if $k$ small.

Graph for finding short tandem repeats:
• Other alignment tasks:
  – EST, or cDNA, to genomic sequence (exons)
  – protein to genomic.
• Can solve by variants of Smith-Waterman:
  – e.g. cDNA vs genomic:
    • set moderately large negative penalty for mismatch and for gap opening,
    • 0 for gap extension.
    • issue of proper placement of splice sites ...