18.417 Introduction to Computational Molecular Biology

Lecture 9: October 7, 2004 — Scribe: Rob Beverly
Lecturer: Ross Lippert — Editor: Chris Peikert

Graphs: Peptide Identification

Introduction

Protein identification is another area of bioinformatics where computational methods supplement experimental techniques to great effect. One useful experimental tool often used is mass spectrometry. Mass spectrometry is a general technique for determining atomic composition. We will study its use in identifying and analyzing the composition of proteins. In particular, we will see that reconstructing the protein sequence from an experimental mass spectrometry result reduces to finding paths in graphs and dynamic programming.

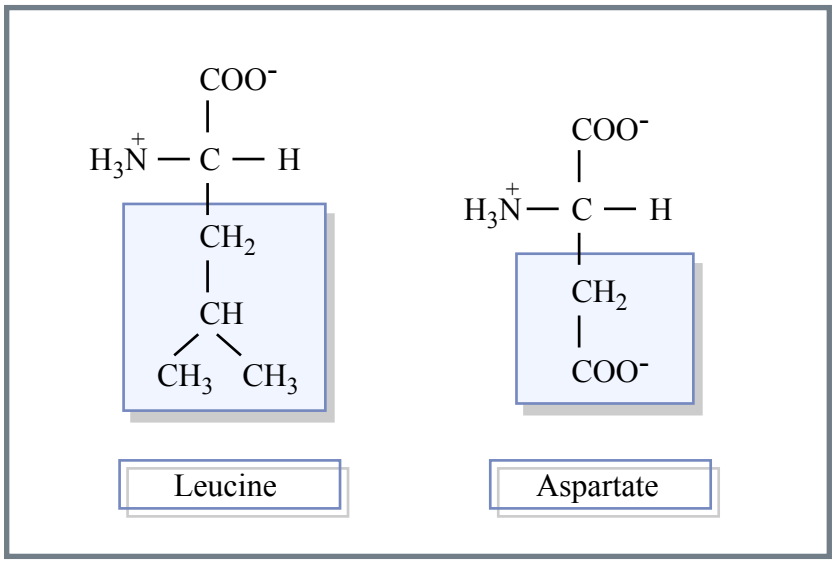
First, we give a high-level overview of proteins, peptides and mass spectrometry.

Peptides and Mass Spectrometry

Proteins are large molecules made up of sequences of amino acids. There are approximately 20 different amino acids with well known masses. Amino acids share a common piece of atomic composition and are differentiated by their "side-chains." Figure 9.1 shows Leucine and Aspartate with the side-chains highlighted. Note that the non-highlighted portions are identical.

Every protein is a linear chain of amino acids connected by a peptide bond. The weakest bond along the backbone is the peptide bond. In order to analyze proteins, there are various experimental methods that break them into many smaller charged pieces, typically at the peptide bond. The process of breaking proteins is known as *collision-induced separation* (CID).

The mass spectrometry method uses a tube with an electric field that causes charged fragments to fly from one end to the other. At the far end of the tube is a detector. The rate at which fragments propagate through the tube is (roughly) proportional



Adapted from Figure 9.1: Amino Acids Differentiated by their Side-Chains

to their mass. Therefore, by looking at when fragments arrive at the detector, one can see how many of each type were present in the original sample. There is also a technique called MS/MS which breaks the fragments again to obtain secondary fragments.

The fragments are classified into two categories: N-terminal and C-terminal. For a given peptide $P = p_1p_2...p_n$ where p_i is an amino acid, the N-terminal is the prefix of P when P is broken at some bond k and the C-terminal is the suffix of P broken at k. The mass of an N-terminal is thus $\sum_{i=1}^{k} m(p_i)$ and the mass of a C-terminal is $\sum_{i=k}^{n} m(p_i)$. Collision-induced separation is highly likely to produce all prefixes and suffixes of the peptide. Thus, for peptide `RSEA` the CID process will produce the following partial peptide terminals:

N-Terminals	C-Terminals
`R`	`A`
`RS`	`EA`
`RSE`	`SEA`

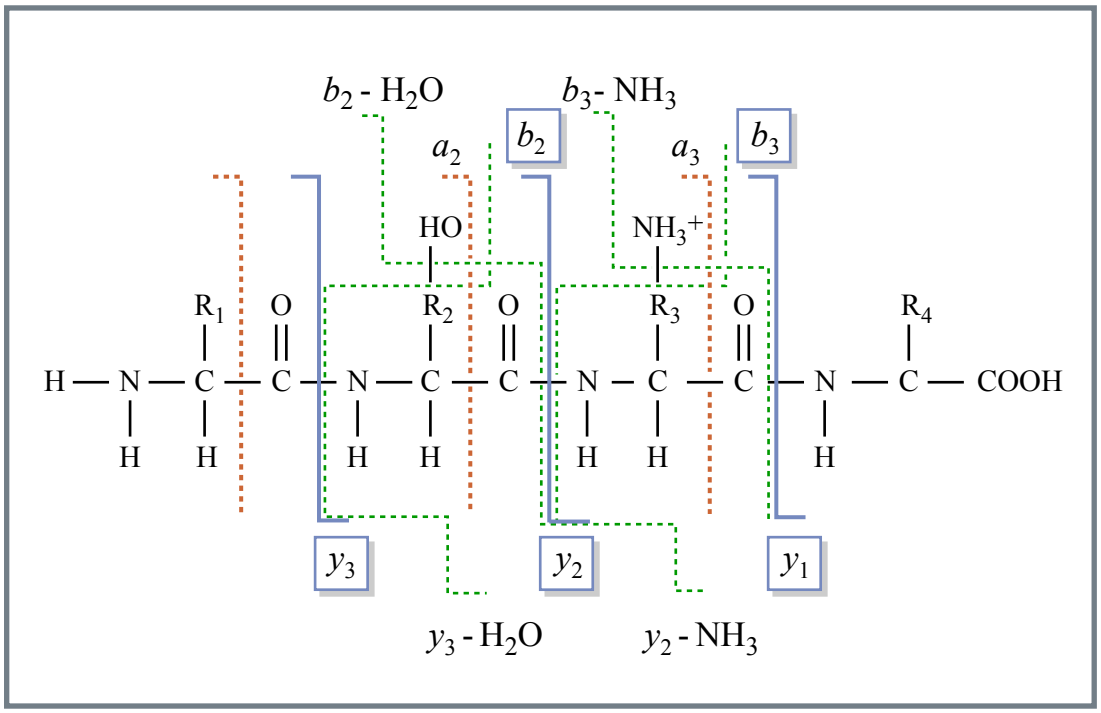
The mass spectrum is a graph of the intensity of different masses. Peaks in the spectrum represent different observed masses and therefore correspond to different fragments.

Mass Spectrum and Ions

The identification problem is then one of inferring the protein from the detected masses. One factor that is in our favor is that only a few types of bonds are broken in the process of making fragments. Most often it is the peptide bond (amine carboxyl chain). Further, double breaks (when a fragment itself breaks such that one piece is neither a suffix nor prefix) are rare enough to effectively be ignored.

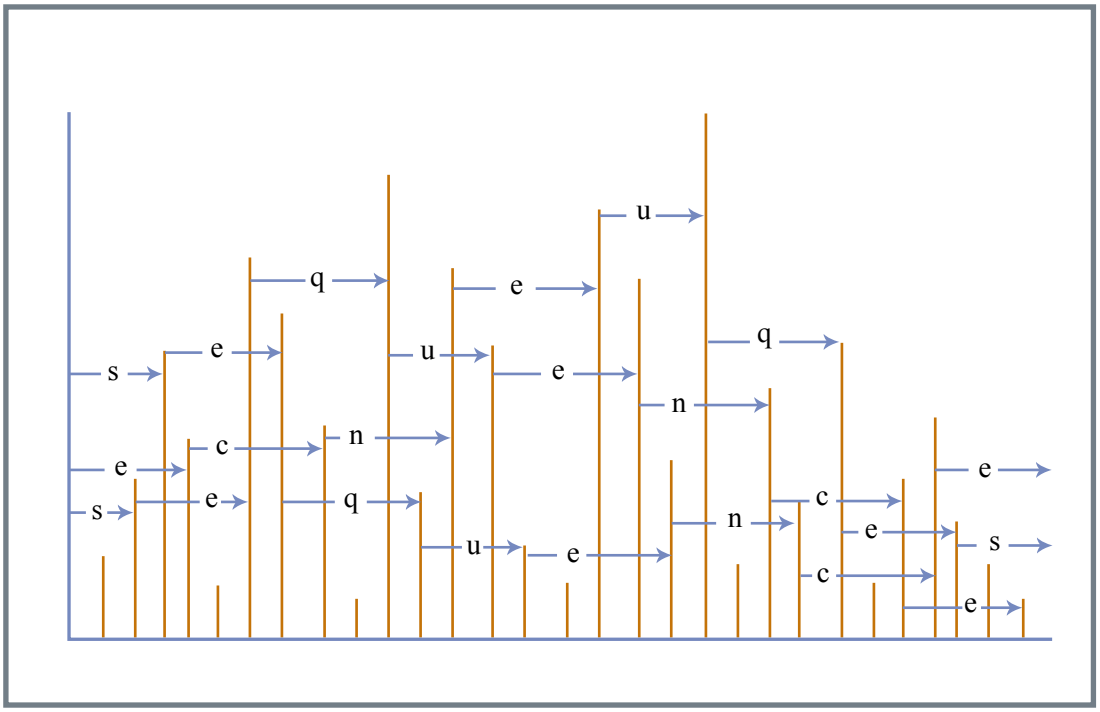
Unfortunately, CID and mass spectrometry are not ideal, and there is some noise in the data. For example, in the process of creating fragments, one of the terminal peptides may lose parts of its fragment resulting in a fragment with lower mass than expected. Thus, water (H_2O) may break off from an N-terminal and the resulting molecules will include an N-terminal with a mass 18 units less than expected. In addition, the water itself will show up on the mass spectrum. Fragments where pieces are missing are called as *ion types*.

Figure 9.2 shows a peptide and the various points at which it may break to form N-terminal and C-terminals. Also shown are b-ions (prefixes where bits have fallen off) and y-ions (suffixes missing pieces).



Adapted from Figure 9.2: Possible Fragments of a Peptide

Thus, the mass spectrum contains not only the N-terminal and C-terminal fragments, but also all of the masses corresponding to the different possible ion types. An example providing the intuition behind the mass spectrum is given in Figure 9.3.



Adapted from Figure 9.3: Peaks in a Spectrum Graph

Spectrum Graphs and Identification

Let $S = \{s_1, s_2, ..., s_n\}$ be an experimental spectrum where each s_i corresponds to a mass peak in the mass spectrum. S will include masses of fragment ions as well as some noise. We will assume that there are a set of k different ion types, which adjust the fragment weights by masses $(-\delta_1, \ldots, -\delta_k)$, respectively. We would like to find a peptide whose theoretical spectrum best matches the experimental spectrum.

This problem reduces to finding the longest path in a DAG: for each mass s_i in the experimental spectrum S, we create k vertices corresponding to masses $s_i+\delta_1, ...s_i+\delta_k$. One of these vertices represents the correct mass of the fragment. In addition, we add a vertex corresponding to zero mass, and the parent mass m (since the parent mass is also represented in the spectrum). We then add edges (i, j) for any i, j such that $j - i$ is the weight of some single amino acid are connected with a directed edge.

In this DAG, finding the longest path from 0 to m corresponds to the correct path to reconstruct the sequence. Unfortunately, experimental spectra often induce some or all of the following problems in our graphs:

- There is no path, due to an incomplete spectrum (i.e., missing prefix or suffix);
- There are multiple paths of the same maximal length (because many ion types increase the probability of such coincidences);
- C-terminal ions and N-terminal ions cannot be differentiated solely by mass;
- Noise in the spectrum, due to imprecision in the mass spectrometry process.

One solution to the no-path problem is to add edges to the graph between vertices whose masses differ by the sum of any two amino acid weights. In case an isolated prefix or suffix is missing, these extra edges compensate to complete the path. Unfortunately, this technique further increases the chance of multiple paths and therefore pushes the problem elsewhere.

The multiple-paths problem can be addressed probabilistically. Suppose we know the probability of each ion type occurring: call it $P(\delta_i)$. We would then have an idea of the probability that a peak $x + \delta_i$ is present for fragment of mass x. Assume that the events of different ion types occurring are independent. Then we will assign appropriate probabilities to the vertices as follows: we will want to "reward" every ion type that explains a particular mass in the spectrum, and also reward ions that are *missing* (these further explain the mass, since they occur with low or zero probability). Thus, the vertex score will be:

$$\left(\prod_{l \in spec} p(\delta_i)\right)\left(\prod_{l \notin spec} 1 - p(\delta_i)\right)$$

The task, as before, is to find the maximal weighted path, where the weight of a path is the product of all probabilities along that path. In practice, we would take logs of all the vertex probabilities, and maximize the sum of these logs.

Protein Identification via Database Search

The idea behind database search to identify proteins is that we assume knowledge of all proteins from a genome in a database. Instead of having to search all 20^ℓ possible peptide sequences (of length ℓ, over 20 amino-acids) to find a weight match, the search could be limited to only those peptides present in the database. Searches then essentially amount to a linear search through the database. This often involves identifying candidates that are within a certain tolerance, because processes such as glycosylation and phosphorylation modify the masses of real-life peptides. Thus, the database search must find best matches between an experimental spectrum and theoretical spectrum allowing up to k modifications. This can be done via a process called *spectral alignment*.

Spectral Alignment

For a spectrum $S = \{s_1, s_2, ..., s_n\}$, consider an *order-preserving shift* Δ_i that transform S into $\{s_1, s_2, \ldots, s_{i-1}, s_i + \Delta_i, s_{i+1} + \Delta_i, \ldots, s_n + \Delta_i\}$.

Define the k-similarity $D(k)$ between sets A and B to be the maximum number of elements in common between these sets after any k shifts. This is actually equivalent to the edit distance, which is solved by dynamic programming. Here's how:

Represent the sets A and B by boolean "indicator arrays," i.e. there is a 1 at index i if and only if mass i is present in the set. Thus, if A is a set of size n, there will be n ones in the array and $a_n - n$ zeros. A shift then corresponds to the addition or deletion of a block of zeros in the array. We can then take the spectral product to create an $n \times m$ matrix with nm ones, at indices $(a_i, b_j) \forall i, j$. The number of 1s along the main diagonal represents the shared peak count with no shifts.

As before in the edit graphs, we want to find a path that maximizes the number of ones along some diagonal path from $(0, 0)$ to (n, m) that takes at most k straight shifts either to the right or downward. We can compute such a path via dynamic programming. Let $D_{ij}(k)$ be the k-similarity between prefix A_i and prefix B_j such that the last elements of A_i and B_j are equal. If we stay along the diagonal and the next elements of A and B are equal, the number of matches increases by one. If not, we subtract one from k (decrementing the number of shifts left available) and add one to begin the new shifted match. The recurrence relation is:

$$D_{i,j}(k) = max_{(i',j')<(i,j)} \begin{cases} D_{i',j'}(k) + 1 \text{ if } i, j \text{ codiagonal} \\ D_{i',j'}(k-1) + 1 \text{ otherwise} \end{cases}$$

The k-similarity is then $D(k) = max_{ij} D_{ij}(k)$. An example of a spectral alignment matrix, including several shifts, is given in Figure 9.4.

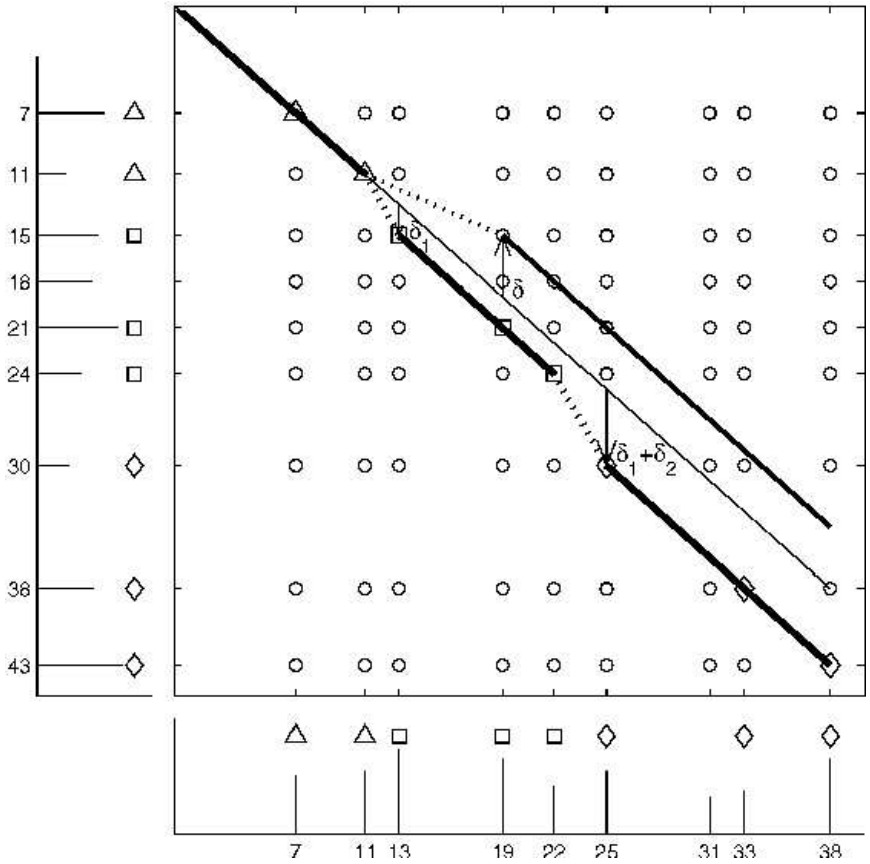


Figure 9.4: Example Spectral Alignment Matrix