

Monday April 8, 2002

Last time we introduced the fundamental alignment problem for aligning two strings: Find an alignment of the two strings to Maximize (over all possible alignments) the number of matches, minus the number of mismatches, minus the number of spaces. An alignment that achieves that maximum is called an optimal alignment. We showed that the number of alignments is too large to list them all, so that some other method will have to be used to find an optimal alignment (I say "an" optimal rather than "the" optimal because there may be ties, i.e. more than one alignment which achieves the maximum value. These are called co-optimal alignments. Soon we will discuss how to efficiently find an optimal alignment without listing all the possible alignments, since there are too many to do that.

Also, in the lecture notes for lecture 3, we introduce the idea of a substitution matrix, where the values for specific matches and mismatches are held. We talked about those in class today, but without much notation. The notes for lecture 3 use a formal notation to talk about the substitution matrix. Let me know if those notes are not clear enough.

But now we enrich the model in order to get alignments that are more biologically relevant. In the above model, each match has a positive value of one, and each mismatch and space each have a negative value of one. Perhaps spaces and mismatches should have different values. A space is associated with a insertion or deletion, while a mismatch is a point mutation. Maybe point mutations are much more likely than an indel. In DNA that seems to be the case, while in protein it depends on the region of the protein. In exterior loops, deletions seem to be more common, while in the hydrophobic core of a globular protein, indels are rare. At any rate, depending on what kinds of strings are being aligned, it may be better to have different values and costs for matches, mismatches and spaces. This leads to the more general objective function:

Find an alignment to maximize (over all alignments) $V_m \times (\#matches) - C_m \times (\#mismatches) - C_s \times (\#spaces)$,

where V_m is a number indicatig the value of a match, C_m is a number indicating the cost of a match, and C_s is a number indicating the cost of a space. Whenever those parameters, V_m, C_m, C_s , are specified, some alignment is optimal. We will see how to efficiently find an optimal alignment soon. And in lab you will see how changing those parameters affects the optimal alignment, using XPARAL. If you want to explore this on your home PC, you can download the Linux version of XPARAL (if you have Linux) or the (toy) windows version called WPARAL for windows.

By picking the right parameters, one can get biologically informative alignments in many instances. However, the model is still not rich enough for all purposes. The next step in enriching the model is the introduction of a term for *gaps*.

```

c  t  t  t  a  a  c  -  -  a  -  a  c
c  -  -  -  c  a  c  c  c  a  t  -  c

```

Figure 1: An alignment with seven spaces distributed into four gaps.

1 Gaps

1.1 Introduction to Gaps

Until now the central constructs used to measure the value of an alignment (and to define similarity), have been *matches*, *mismatches* and *spaces*. Now we introduce another important construct, *gaps*. Gaps help create alignments that better conform to underlying biological models and more closely fit patterns that one expects to find in meaningful alignments.

Definition A gap is any *maximal, consecutive run* of spaces in a *single* string of a given alignment ¹.

A gap may begin before the start of S , in which case it is bordered on the right by the first character of S , or it may begin after the end of S , in which case it is bordered on the left by the last character of S . Otherwise a gap must be bordered on both sides by characters of S . A gap may be as small as a single space. As an example of gaps, the alignment in Figure 1 has four gaps containing a total of seven spaces. That alignment would be described as having five matches, one mismatch, four gaps and seven spaces. Notice that the last space in the first string is followed by a space in the second string, but those two spaces are in two gaps and do not form a single gap.

By including a term in the objective function that reflects the gaps in the alignment, one has some influence on the *distribution* of spaces in an alignment, and hence the overall shape of the alignment. In the simplest objective function that includes gaps, each gap contributes a *constant* weight W_g , independent of how long the gap is.

Changing the value of W_g relative to the other weights in the objective function can change how spaces are distributed in the optimal alignment. A large W_g encourages the alignment to have few gaps, and the aligned portions of the two strings will fall into a few substrings. A smaller W_g allows more fragmented alignments.

¹Sometimes in the biological literature the term “space” (as we use it) is not used. Rather, the term “gap” is used both for “space” and for “gap” (as we have defined it here). This can cause much confusion, and in this book the terms “gap” and “space” have distinct meanings.

1.2 Why gaps?

Just as a space in an alignment corresponds to an insertion or deletion of a single character in the edit transcript, a gap in string S_1 opposite substring α in string S_2 corresponds to either a deletion of α from S_1 or to an insertion of α into S_2 . The concept of a gap in an alignment is therefore important in many biological applications because the insertion or deletion of an *entire* substring (particularly in DNA) often occurs as *single* mutational event. Moreover, many of these single mutational events can create gaps of quite varying sizes with almost equal likelihood (within a wide, but bounded, range of sizes). Much of the repetitive DNA discussed in Section ?? is caused by single mutational events that copy and insert long pieces of DNA. Other mutational mechanisms that make long insertions or deletions in DNA include: *unequal crossing-over* in meiosis (causing an insertion in one string and a reciprocal deletion in the other); DNA *slippage* during replication (a portion of the DNA is repeated on the replicated copy because the replication machinery loses its place on the template, slipping backwards and repeating a section); insertion of *transposable* elements (jumping genes) into a DNA string; insertions of DNA by *retro-viruses*; and *translocations* of DNA between chromosomes [6, 7]. In fact, the duplication of an entire gene (which shows up as a gap if the DNA strings before and after the duplication are aligned) is considered to be a more common event than the insertion or deletion of a single nucleotide or a single amino acid [4]. See Figure 2 for an example of gaps in real genomic data.

When computing alignments for the purpose of deducing evolutionary history over a long period of time, it is often the gaps that are the most informative part of the alignments. In DNA strings, single character substitutions due to point mutations occur continuously and usually at a much faster rate than (non-fatal) mutational events causing gaps. The analogous gene (specifying the “same” protein) in two species can thus be very different at the DNA sequence level, making it difficult to sort out evolutionary relationships on the basis of string similarity (without gaps). But large insertions and deletions in molecules that show up as gaps in alignments occur less frequently than substitutions. Therefore common gaps in pairs of aligned strings can sometimes be the key features used to deduce the overall evolutionary history of a set of strings [1, 8].

At the protein level, recall that many proteins are “built of different combinations of protein domains that have been selected from a relatively small repertoire”[2]. Hence two protein strings might be relatively similar over in several intervals, but differ in intervals where one contains a protein domain that the other does not. Such an interval most naturally shows up as a gap when two proteins are aligned. In some contexts, many biologists consider the proper identification of the major (long) gaps as *the* essential problem of protein alignment. If the long (major) gaps have been selected correctly, the rest of the alignment – reflecting point mutations – is then relatively easy to obtain.

An alignment of two strings is intended to reflect the cost (or likelihood) of muta-

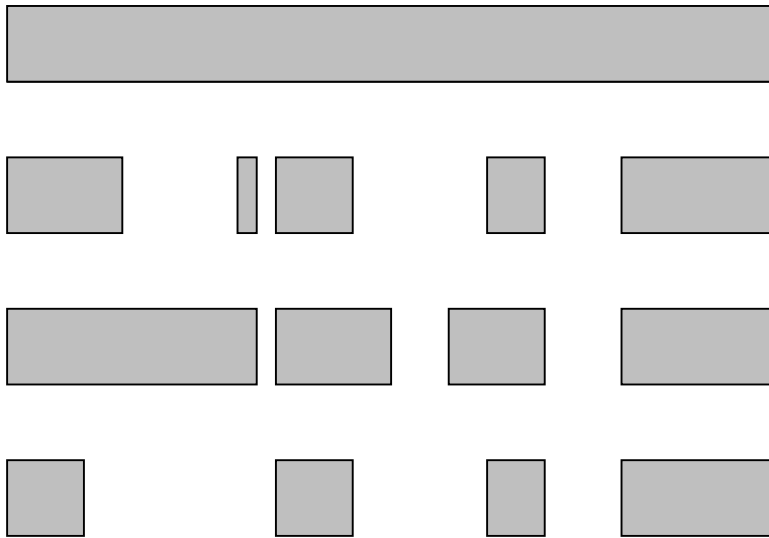


Figure 2: Each of the four rows depicts part of the RNA sequence of one strain of the HIV-1 virus. The HIV virus mutates rapidly, so that mutations can be observed and traced. The bottom three sequences are from virus strains that have each mutated from an ancestral strain represented in the top row. Each of the bottom sequences is shown aligned to the top sequence. A black box represents a substring that matches the corresponding substring in the top sequence, while each white space represents a gap resulting from a known sequence deletion. This figure is adapted from a one in [5].

tional events needed to transform one string to another. Since a gap of more than one space can be created by a single mutational event, the alignment model should reflect the true distribution of spaces into gaps, and not merely reflect the number of spaces in the alignment. It follows that the model must specify how to weight gaps so as to reflect their biological meaning. In this chapter we will discuss different proposed *schemes* for weighting gaps, and in later chapters we will discuss additional issues in scoring gaps. But first we consider a concrete example illustrating the utility of the gap concept.

1.3 cDNA matching: a concrete illustration

One concrete illustration of the use of gaps in the alignment model comes from the problem of *cDNA matching*. In this problem, one string is much longer than the other and the alignment best reflecting their relationship should consist of a *few* regions of very high similarity interspersed with “long” gaps in the shorter string (see Figure 3). Note that the matching regions can have mismatches and spaces, but they should be a small percentage of the region.

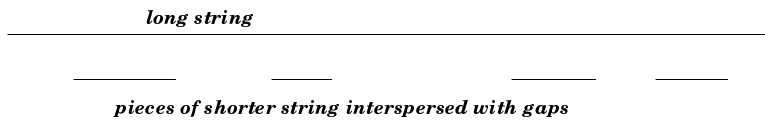


Figure 3: In cDNA matching, one expects the alignment of the smaller string with the longer string to consist of a few regions of very high similarity, interspersed with relatively long gaps.

Biological setting of the problem

In eukaryotes, a gene that codes for a protein is typically made up of alternating *exons* (expressed sequences) that contribute to the code for the protein, and *introns* (intervening sequences) which do not. The number of exons (and hence also introns) is generally modest (4 to 20 say) but the lengths of the introns can be huge compared to the lengths of the exons.

At a *very* coarse level, the protein specified by a eukaryotic gene is made in the following steps. First an RNA molecule is transcribed from the DNA of the gene. That RNA transcript is a complement of the DNA in the gene in that each *A* in the gene is replaced by *U* (uracil) in the RNA; each *T* is replaced by *A*; each *C* by *G* and each *G* by *C*. Moreover, the RNA transcript covers the entire gene, introns as well as exons. Then, in a process that is not completely understood, each intron-exon boundary in the transcript is located, the RNA corresponding to the introns is spliced out (or *snurped* out by a molecular complex called a *snrnp*[9]) and the RNA regions corresponding to exons are concatenated. Additional processing occurs that we will

not describe. The resulting RNA molecule is called the *messenger RNA (mRNA)*; it leaves the cell nucleus and is used to create the protein it encodes.

Each cell (usually) contains a copy of all the chromosomes and hence of all the genes of the entire individual, yet in each specialized cell (a liver cell for example) only a small fraction of the genes are expressed. That is, only a small fraction of the proteins that are encoded in the genome are actually produced in that specialized cell. A standard method to determine which proteins are expressed in the specialized cell-line, and to hunt for the location of the encoding genes, is to capture the mRNA in that cell after it leaves the cell nucleus. That mRNA is then used to create a DNA string complementary to it. This string is called *cDNA* (complementary DNA). So compared to the original gene, the cDNA string consists only of the concatenation of exons in the gene.

It is routine to capture mRNA and make cDNA libraries (complete collections of a cell's mRNA) for specific cell lines of interest. As more libraries are built up, one collects a *reflection* of all the genes in the genome and a taxonomy of the cells that the genes are expressed in. In fact, a major component of the human genome project [3] is to obtain cDNAs reflecting most of the genes in the human genome. This effort is also being conducted by several private companies, and has led to some interesting disputes over patenting cDNA sequences.

After cDNA is obtained, the problem is to determine where the gene associated with that cDNA resides. Presently, this problem is most often addressed with laboratory methods. But if the cDNA is sequenced or partially sequenced (and in the human genome project for example, the intent is to sequence parts of each of the obtained cDNAs), and if one has sequenced the part of the genome containing the gene associated with that cDNA (as for example one would have after sequencing the entire genome), then the problem of finding the gene site given a cDNA sequence becomes a string problem. It becomes one of aligning the cDNA string against the longer string of sequenced DNA in a way that reveals the exons. It becomes the cDNA matching problem discussed above.

Why gaps are needed in the objective function

If the objective function includes terms only for matches, mismatches and spaces, there seems no way to encourage the optimal alignment to be of the desired form. It's worth a moment's effort to explain why.

Certainly you don't want to set a large penalty for spaces, since that would align all the cDNA string close together, rather than allowing gaps in the alignment corresponding to the long introns. And you want a rather high penalty for mismatches – although there may be a few sequencing errors in the data, so that some mismatches will occur even when the cDNA is properly cut up to match the exons, there should not be a large percentage of mismatches. In summary, you want small penalties for spaces, relatively large penalties for mismatches, and positive values for matches.

Now what kind of alignment would likely result using an objective function that has low space penalty, high mismatch penalty, positive match value of course, and no term for gaps? Remember that the long string contains more than one gene, that the exons are separated by long introns, and that DNA has an alphabet of only four letters present in roughly equal amounts. Under these conditions, the optimal alignment would probably be the *longest common subsequence* between the short cDNA string and long anonymous DNA string. And because the introns are long and DNA has only four characters, that common subsequence would likely match *all* of the characters in the cDNA. Moreover, because of small but real sequencing errors, the true alignment of the cDNA to its exons would not match all the characters. Hence the longest common subsequence would likely have a higher score than the correct alignment of the cDNA to exons. But the longest common subsequence would fragment the cDNA string over the longer DNA and not give an alignment of the desired form – it would not pick out its exons.

Putting a term for gaps in the objective function rectifies the problem. By adding a constant gap weight W_g for each gap in the alignment, and setting W_g appropriately (by experimenting with different values of W_g), the optimal alignment can be induced to cut up the cDNA to match its exons in the longer string². As before, the space penalty is set to zero, the match value is positive and the mismatch penalty is set high.

References

- [1] S. Baldauf and J. Palmer. Animals and fungi are each other's closest relatives: Congruent evidence from multiple proteins. *Proc. of the Nat. Academy of Science*, 90:11558–11562, 1993.
- [2] C. Chothia. One thousand families for the molecular biologist. *Nature*, 357:543–544, 1992.
- [3] N. Cooper. *The human genome project*. University Scientific Press, 1994.
- [4] M. O. Dayhoff. Computer analysis of protein evolution. *Scientific American*, pages 87–94, July 1969.
- [5] N.J. Deacon, J. Mills, and et al. Genomic structure of an attenuated quasi species of hiv-1 from a blood transfusion donor and recipients. *Science*, 270:988–991, 1995.
- [6] W. H. Li and D. Graur. *Fundamentals of Molecular Evolution*. Sinauer, Sunderland, MA, 1991.

²This really works, and it is a very instructive exercise to try it out empirically.

- [7] E. McConkey. *Human genetics: The molecular revolution*. Jones and Bartlett, Boston, MA, 1993.
- [8] S.K. Shyue, D. Hewett-Emmett, H. Sperling, D. Hunt, J. Bowmaker, J. Mollon, and W.H. Li. Adaptive evolution of color vision of genes in higher primates. *Science*, 269:1265–1267, 1995.
- [9] J. Steitz. Snurps. *Scientific American*, pages 56–63, June 1988.