Bioinformatics: Tasks, techniques, tools

Rahul Siddharthan
Institute of Mathematical Sciences,
CIT Campus, Taramani, Chennai 600113

School on Parallel Computation: January 13, 2005

1 Introduction: Molecular Biology 101

We start with a very basic review of biology, necessary for any further work, but largely sufficient for getting started in computational biology. One can (and must) learn more "on the job".

Biomolecules are sequences of monomers (DNA, RNA=nucleotide sequences, proteins=amino acid sequences). DNA is the molecule that contains the entire blueprint for an organism. It contains genes that encode the sequences for every protein in the organism, as well as non-coding regions that, among other things, contain regulatory mechanisms for when and in what order different genes get turned on, and may have other functions as well.

Most genes code for proteins; some genes code for RNA molecules that play various roles in the cell. Both DNA and RNA are polymers of "nucleotides" which are bases of four kinds [adenine=A, cytosine=C, guanine=G, thymine=T (DNA only), uracil=U (RNA only)] attached to sugar-phosphate backbones. Apart from the one difference in bases, RNA and DNA are very similar except that DNA usually exists in double-stranded "base-paired" form and RNA is in single-stranded form.

The backbone of DNA (or RNA) is not symmetrical: each monomer has a 5'-phosphate group at one end and a 3'-hydroxyl group at the other. Each strand is usually read from the 5' to the 3' end. The two strands go in opposite directions. The nucleic acids are base-paired A to T, G to C. A-T bonds are weaker (double-bonds), G-C bonds are stronger (triple-bonds).

Proteins are the "building blocks" of life, responsible for a vast number of cellular processes. They regulate genes, catalyse various biochemical reactions, form machinery for synthesis of other molecules (including other proteins) and are important parts of organelles and tissues. They are polymers of amino acids (carboxylic acids with an amide group and a side chain). There are twenty naturally occurring amino acids, differing in their side chains.

Proteins tend to "fold" into complex three-dimensional conformations; usually the fold is unique and misfolding is rare. The details of the fold are biochemically important. Usually a few active "domains" (for example, binding to DNA, interaction with other proteins) help the protein play important roles in gene regulation, catalysis, etc; these domains tend to be well conserved across species, while the rest of the protein sequence can mutate a lot. Much computational effort goes into studying protein structure and function, but we will not discuss this vast subject here.

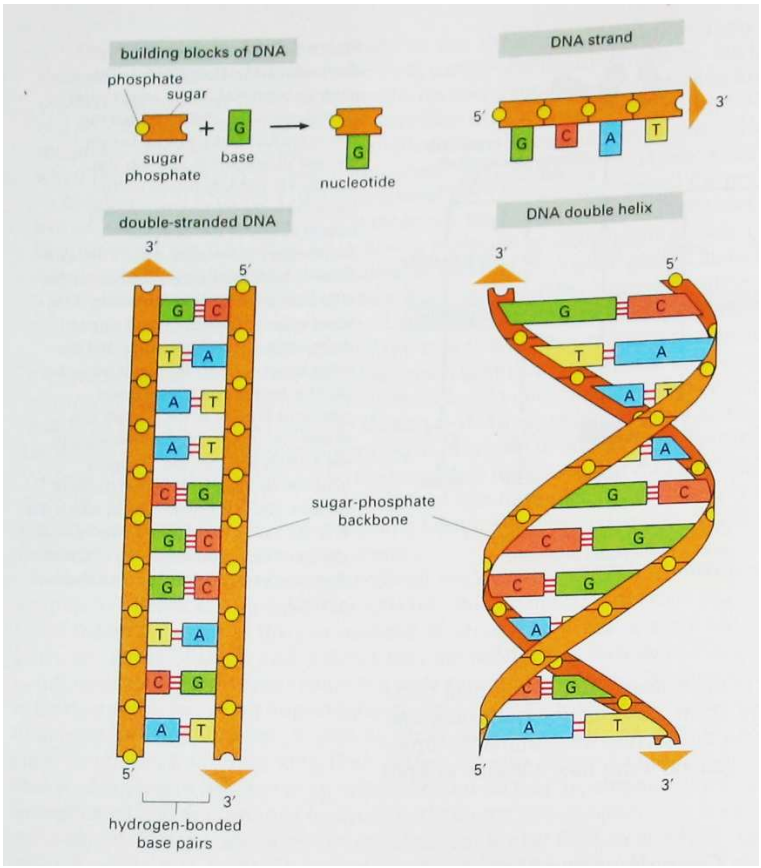


Figure 1: DNA (from Alberts et al., "Molecular Biology of the Cell")

Genes that code for proteins are first "transcribed" to "messenger RNA" (mRNA) molecules, and then the RNA is "translated" to proteins. Each "codon" of three nucleotides corresponds to a unique amino acid. Since there are 4 nucleotides, there are 64 possible codons; three of these are "stop codons" (TAA, TAG, TGA) (sometimes called "nonsense codons") and don't code for amino acids, instead indicating a stop to transcription. The remaining 61 code for 20 amino acids. Several codons (up to six) thus can code for the same amino acid. The "start codon" is ATG, which codes for the amino acid methionine.

2 What are the biological problems?

There are of course a huge number of problems in biology that can benefit from a quantitative treatment, ranging from single molecule behaviour to population biology and ecology. From the title, we are already restricting ourselves to bioinformatics, but we will mainly focus on DNA sequence analysis, with only occasional mention of proteins.

The following are a few issues of interest to biologists (and often of medical importance) that could benefit from analysis of DNA sequence:

- Cellular processes: how the cell carries out its normal tasks; how it responds to external events like heat shock and starvation; how it carries out complex cascades of events such as the process of cell division (mitosis).
- Development: How a complex organism (eg a worm, a fly, a human) develops from a single fertilised egg. As this embryonic cell divides, the daughter cells also slowly differentiate into functions. This happens as a result of "gradients" of various factors (some of them maternal) that change gene regulation in different parts of the embryo and ultimately cause different cells to develop in highly specialised ways.
- Evolution: How different species evolve, how new functionality develops.

All cellular and developmental processes are controlled by genes that get turned on in response to some external condition (stress, starvation, embryonic gradients) or cyclically (cell cycle). Computational study of how these genes are regulated and how they function is very useful. This is done by analysing the gene sequence and regulatory DNA sequence of the organism itself, and by comparison of this sequence with already-annotated sequence from other organisms. Highly similar (homologous) genes exist among widely different organisms; such genes are called "orthologues". Many subsystems in widely different organisms are very similar and are regulated by orthologous proteins; some proteins exist largely unchanged from primitive archaebacteria all the way to humans.

Moreover, many genes with heavy sequence identity often exist in the same organism, arising from ancestral "gene duplication" events; their function is often slightly differentiated, and in fact this is a major driving factor in evolution.

There are now "high-throughput" microarray experiments that can essentially give the response of every gene in the genome; analysing, clustering and interpreting this data, and combining it with other computational tasks in gene regulation, is of great interest.

Finally, the study of phylogeny (evolutionary history of organisms) and the classification (taxonomy) of organisms has been revolutionised by DNA sequencing.

3 Some tasks in DNA sequence analysis

Large quantities of sequence data are being published, for organisms from bacteria to higher mammals. It will take decades to analyse the data.

Here are a few of the tasks involved in the analysis:

- Sequence assembly: Sequencing involves a "shotgun" approach where DNA fragments 1000bp long are sequenced with significant overcoverage. These then have to be "assembled".
- Annotation: The assembled genomes have to be "annotated": genes identified and marked out, their functions identified, and so on.
- Motif finding: Non-coding DNA contains "regulatory" regions where proteins called "transcription factors" bind to "turn on" genes. Identifying such regions, and binding sites for individual TFs, is of great importance. TFs typically bind to small "motifs", so the task is to find overrepresented short "motifs" in larger quantities of sequence.
- Sequence alignment: In the last two tasks, it is very useful to compare genomes of previously sequenced species. "Comparative genomics" is becoming a very important subfield. Detection and alignment of homologous sequence is an important task here.
- Phylogenetic trees: Given sequence data from different species, it is useful to reconstruct their phylogenetic relationship.

Algorithms exist for all these tasks, but all are evolving with increasing understanding of the function of non-coding DNA, increasing mathematical and algorithmic sophistication in the methods, and increasing raw computational power available to tackle these tasks.

There is not much explicit mention of parallel programming in what follows. But most problems are intrinsically parallelisable, and many tasks require several independent runs that can be done trivially in parallel.

4 Assembly of genomes

The basic problem is that the entire genome is sequenced in millions of small fragments (at most a few hundred basepairs long, a so-called "shotgun" method) with large (at least 8x-10x) overcoverage. Overlapping fragments are then pieced together like a jigsaw puzzle. It can be a hard problem because DNA sequence is far from random, and has many repeats and closely similar segments that can cause incorrect assembly. Moreover, the sequencing can contain errors, and a small part of the DNA may be impossible to sequence. Therefore, often, complete assembly of the genome is impossible, and what's published is a set of "contigs" representing fragments of the genome.

Figure 2: DNA contains genes and non-coding regions; genes themselves commonly contain non-coding regions called introns

This is approximately a "smallest common superstring" problem: Given strings $s_1, s_2, \ldots$, find the smallest string T such that every s_i is a substring of T. This is an NP-hard problem but there are fast approximate algorithms.

There are lots of missing details above that we won't discuss here.

5 Gene annotation

Identifying a gene in DNA sequence from scratch can be extremely hard. In higher organisms, the problem is further complicated by the fact that genes aren't continuous, but fragmented into "exons" (actual coding regions) and "introns" (non-coding regions interspersed in the genes).

Recall that the genetic code is in "codons" of three nucleotides. In searching for genes, one has six possible "reading frames" to search for: two strands of DNA, and three possible frames (choices of start of codon) on each site.

All genes start with the codon ATG, which codes for the amino acid methionine, but not every occurrence of ATG indicates a start site. All genes stop with a "stop codon" (one of TAA, TAG, TGA), and no stop codon occurs in the same reading frame before an actual stop site. There are 64 possible codons so there's a 1 in 20 chance of a codon being a stop codon. If a given reading frame has a stretch of sequence much longer than 20 codons (60 nucleotides), that suggests it is likely to be coding sequence; this is often a first step in searching for genes.

Many genes are discovered with additional help: by comparison with known genes from other organisms, or searching for a match to a known protein sequence. The most popular tool to align genetic or protein sequences is BLAST. This searches unannotated sequence for matches to a "query sequence". A drawback is that sequencing generally doesn't assemble an entire genome but only a number of "contigs" which it is hard to further assemble with confidence. If these contigs are not very big, it's likely that genes are split across contigs, in which case BLAST will give poor matches.

Another complication is that, even within the same organism, many genes have duplicate pairs ("paralogs") that arose from some ancestral gene-duplication events. These all have good BLAST hits with one another, and therefore will all have good BLAST hits with any orthologous gene(s) in another species. In that case, making the mapping of genes from one organism to another can be tricky. One can consider "synteny", that is, the ordering of genes. Eg, if genes A, B, C are contiguous in one species and have multiple matches A', B', C' in the other species, but in only one case is the order A'-B'-C' the same, that is the likely assignment. One can also use "reciprocal best match" as a criterion: one matches A to A' if A' is the best BLAST match of A in species 2 and A is the best BLAST match of A' in species 1. Like other problems in this field, the best approach is not agreed on.

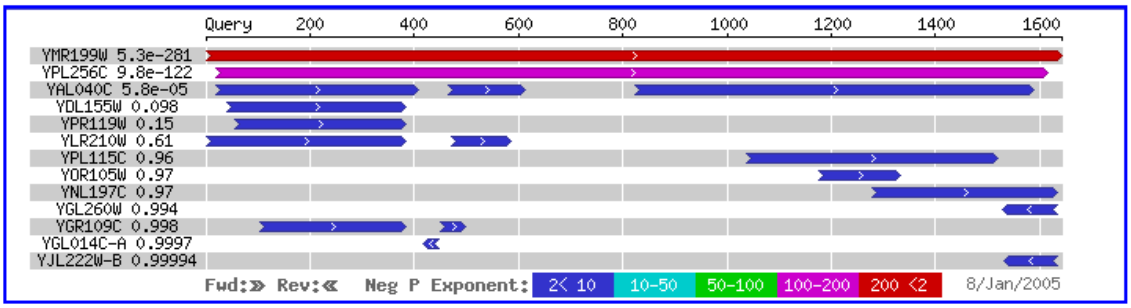


Figure 3: Result of running BLAST on yeast genome with yeast gene CLN1 (YMR199W) coding sequence as query: graphical view, using web interface at `http://www.yeastgenome.org`. CLN2 (YPL256C) is a close homologue of CLN1, arising from an ancestral whole-genome-duplication event, while CLN3 (YAL040C) is also closely related; several other genes exhibit significantly homologous domains.

6 Regulatory sites

Genes are transcribed to mRNA which is translated to protein, but this does not happen automatically. A molecule called "RNA polymerase" must be recruited to near the start site of the gene. To recruit the polymerase, other proteins called "transcription factors" are usually needed. These generally bind to short, specific "motifs" upstream of the start site of the gene.

Thus, it is of interest to find overrepresented short sequences that could represent putative binding sites for TFs.

Motif-finding is also of interest to protein biologists who could use it to predict structural or functional properties of a protein.

Two common techniques for motif finding are: Monte-Carlo sampling (the Gibbs sampler) and entropy maximisation (MEME). The Gibbs sampler involves starting from a random configuration and making "moves"; unlike Metropolis "try/accept/reject" moves that physicists are familiar with, Gibbs moves calculate the result of every possible move in a category, and sample from that distribution. This is the timeconsuming part of the algorithm but, fortunately, is the part that can be easily parallelised (though I'm not aware of parallel implementations). For example, if one expects three binding sites, place them randomly, then pick one randomly and sample every possible new location for it; and repeat.

7 Phylogeny

Pairwise distances may be calculated between two sequences given an evolutionary model. This can be done at several levels: at the lowest, single-gene level it could be homology (sequence identity) of the gene, while at the highest, whole-genome level it could be the global order of the genes and the number of elementary "shuffling" operations needed to take one ordering to the other.

Given a set of species with estimated pairwise evolutionary distances, the task is to construct a tree showing their combined ancestry. Here we give one of many algorithms used for this task, the UPGMA algorithm ("unweighted pair group method using arithmetic averages"):

We grow the tree from "clusters" of sequences. We define the distance d_{ij} of two clusters as the average pairwise distance of all pairs of sequences between those clusters. Then the algorithm is:

1. Initially put each sequence in a cluster by itself, and define one leaf for each cluster, at height zero.
2. Find the two clusters C_i, C_j for which d_{ij} is minimal. (If no unique answer, pick one pair randomly.)
3. Merge these clusters into $C_k = C_i \cup C_j$.
4. Define a new node k at height $d_{ij}/2$, whose daughter nodes are i, j.
5. Repeat, at each stage merging two clusters as above, until only one cluster remains.

8 Sequence alignment

Now that we have given an overview of all these topics, let's look a little more closely at sequence alignment in particular. There are several possibilities of "scoring functions" for an alignment, but usually the basic parameters are "substitution matrices" and "gap penalties." For a scoring function, an algorithm may be "exact", that is, find the alignment with the best score; or it may be "heuristic", that is, use a fast but approximate technique to find an alignment with a good, but not necessarily the best, score.

An example of an alignment is

```
ACAATGCAGTGACCCAGCGT----ACGTTAAGATCATG
ACAGTG---TGTCCCAGCCTACACACGT-AAGTTCATG
```

We score each vertically aligned pair of bases with a number that depends on whether they match or mismatch: for example, the "log-odds" score which is the log ratio of the joint probability to their individual probabilities:

$$s(b_1, b_2) = \log \frac{p(b_1, b_2)}{p(b_1)p(b_2)}$$

For protein alignments there are standard substitution matrices used. For DNA alignment, we can probably get away with a simpler scoring rule:

$$s(b_1, b_2) = 1 \text{ if } b_1 = b_2, -p \text{ otherwise}$$

where p is a "mismatch penalty". Note that, for random sequence, p has to be greater than $1/3$ roughly (and in practice, much greater, since DNA sequence is not random), otherwise you can have arbitrarily long matches.

We score the gaps with a "gap penalty": for a gap of length ℓ, this is either a "linear penalty"

$$g = \ell d$$

that is, proportional to the gap, or an "affine penalty"

$$g = d + (\ell - 1)e$$

where the penalty for the first gap, d, is larger than the penalty for increasing the gap, e.

8.1 Global alignment: Needleman-Wunsch

This is an algorithm to find the best global alignment of two sequences, given the above scoring function. It is an example of "dynamic programming". The idea is to build up alignments of longer sequences from alignments of shorter sequences.

Suppose you have two sequences x, y of lengths ℓ_1 and ℓ_2, and you are trying to find the best global alignment. Construct a matrix F_{ij}, in which each element is calculated from an earlier element (with smaller i and/or j), and each element contains two components:

- The score of the best alignment of the first i bases in sequence 1 and the first j bases in sequence 2. This is

$$F_{ij} = \max \begin{cases} F_{i-1,j-1} + s(x_i, y_j) \\ F_{i-1,j} - d \\ F_{i,j-1} - d \end{cases}$$

  (for simplicity, we are not considering affine gap penalties, but the algorithm can be extended to include those.)

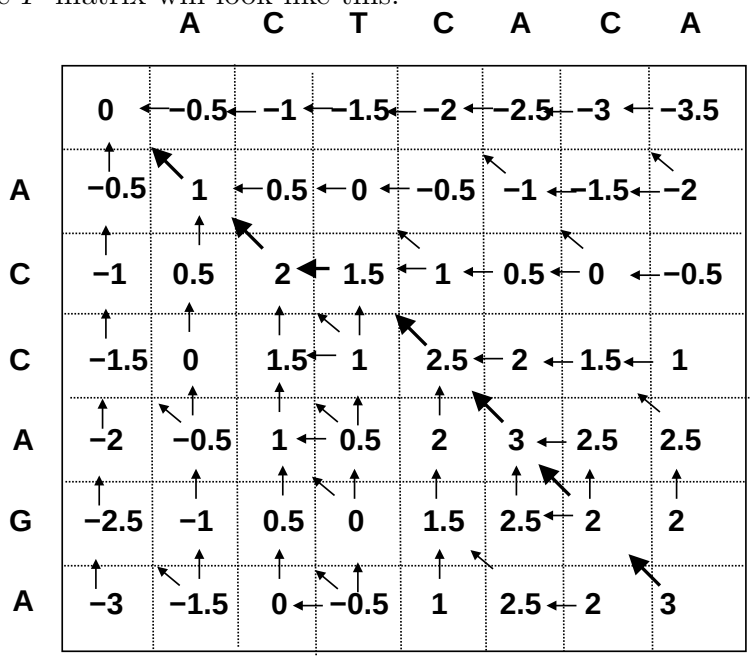
- A pointer to the previously calculated element of the F matrix from where this element was calculated.

F_{00} is defined to be zero, and One fills out the entire F matrix like this, and then uses a "backtrace" from the last element (following the arrows) to construct the global alignment. For example, consider aligning two short strings $x =$ "ACCAGA", $y =$ "ACTCATA" with the parameters

$$s(b_1, b_2) = 1 \text{ if } b_1 = b_2, -1 \text{ otherwise}$$

$$d = 0.5$$

The F-matrix will look like this:



Note that some elements can be obtained in more than one way; one can choose to store only one pointer (which will always give a best answer, but the answers may not be unique), or all pointers (which allows us to recover all answers).

Finally, one does a "traceback" from the bottom-right element, following the arrows. The bold arrows give the alignment below:

```
ACTCACA
AC-CAGA
```

However, the following alignments have the same score:

```
ACTCAG-A
AC-CA-CA

ACTCA-GA
ACTCAC-A
```

8.2 Local alignment: Smith-Waterman

Often we don't want to align the entire sequence, but want to find subsequences of both sequences that align well. This can be done by a simple modification of the above Needleman-Wunsch algorithm: create a F-matrix as before, but now, do not let any element of the F-matrix become negative; instead, when its best value calculated from previous elements becomes negative, mark that element zero (without any pointers). Also, keep track of the the element with the maximum score computed. After calculating the F matrix, do a traceback not from the bottom-right element but from the maximum-scoring element, until you hit a zero element (which could be the top-left element or some other element).

If one wants gapless alignments (infinite gap penalty), each element F_{ij} is just calculated from $F_{i-1,j-1}$. One does not need any pointers: the traceback from element F_j just goes to $F_{i-1,j-1}$. However, in this case one can think of more efficient algorithms.

8.3 Heuristic alignments

The above algorithms all give a "best answer" in terms of the scoring function used; however, they all have computation time of order $\ell_1\ell_2$ (product of lengths of strings). When aligning lots of large sequences, computation time can become an issue.

Therefore, "heuristic" algorithms have been developed that give a good, but not necessarily, best, answer. The best-known is probably BLAST, which now exists in several versions. The basic idea is that each alignment is likely to contain short stretches of exact matches; therefore, a dictionary of all possible short exact matches is built in a preprocessing step, and then the alignment is built from these matches.

8.4 Multiple alignment

So far we have discussed alignment of two sequences. Alignment of multiple sequences is somewhat trickier. Here are two approaches.

1. One can make a global alignment of the closest two sequences, then make successive alignments of the next-closest sequences consistent with existing alignments. This is the approach taken by ClustalW, one of the oldest multiple alignment programs. Either one needs to supply a phylogenetic tree, or the program needs to estimate one from quality of pairwise alignments.

2. One can make successive pairwise *local* alignments, in order of score, without prior phylogenetic ordering of species. Lower-scoring alignments are carried out only when consistent with high-scoring alignments. Enforcing consistency conditions across different species can be a bit tricky. This is the approach taken by Dialign (and also by a sequence alignment program I'm currently working on). This approach lends itself in an obvious way to parallelism (one can make a list of pairwise alignments in parallel), and in fact a parallel version of Dialign was recently published.

9 Conclusion

This is only a quick overview of things that are done in computational biology and bioinformatics. For more information, read the literature...

10 References

For computational techniques, a good reference is "Biological sequence analysis" by Durbin, Eddy, Krogh, Mitchison (Cambridge, 1998). It covers some areas in detail, and other areas (like motif-finding and genome assembly) hardly at all. Moreover, the field is fast moving and the book shows its age in places. Nevertheless it's a good start.

For molecular biology, a standard and very readable reference is "Molecular biology of the cell" by Alberts et al (4th edition; earlier editions are rather outdated now but the new edition is very up-to-date.) The book begins at a very basic level but covers a huge amount of ground eventually.