Algorithms in the Real World
Computational Biology III:
Multiple Alignment and the Human Genome Project

Scribe notes by Marc Fasnacht
Based on a lecture by Guy Blelloch

December 6, 2001

1 Multiple Alignment

1.1 Motivation

In previous lectures we have seen algorithms that can find the optimal alignment of two strings given a scoring matrix. These methods are often used in biological applications to discover sequences with high sequence similarity in biological databases. Since sequence similarity implies structural and functional similarity in proteins, this allows to find functionally related proteins.

Often however, biologists are faced with the reverse problem: Given a set of biologically related proteins (e.g. proteins that perform the same or similar functions in different organisms) what are the conserved patterns in the corresponding amino acid sequence or DNA? This in not an obvious task: Evolutionary or functionally related proteins often differ significantly in the underlying amino acid sequences all while preserving the three-dimensional structure. For this reason pairwise alignment gives very little information about conserved parts. Aligning a large number of sequences however makes it easier to discover such patters. Specific applications for multiple alignment methods include

- assembling noisy "reads" of fragments of sequences
- characterizing families of genes. This can be done by finding a 'canonical member' of a family and by studying how the other members of the family differ from that gene.
- deducing the evolutionary history from DNA or protein sequences.
- improve the accuracy of structure prediction.

A	C	T	_	G
A	C	A	C	G
A	G	T	G	_
C	C	_	G	C

Figure 1: Example of multiple alignment

1.2 Problem statement

The multiple alignment problem can be defined as follows:

Definition 1 Given a set of k strings $S = \{S_1, S_2, \ldots, S_k\}$ find an alignment such that the number of aligned symbols is optimal.

As in the case of pairwise alignment, we can also look for local multiple alignment. In this case we select substrings S_i' from each string $S_i \in S$ and then globally align those substrings. Multiple alignment problems are typically NP-Hard, so that approximate methods are often used.

1.3 Scoring of Multiple Alignment

There is no easy generalization of the "goodness" or score of an alignment from pairwise to multiple alignment. Some of the commonly used scoring functions d include:

- **Distance from a consensus**
  In this method, we first find a consensus string S_c (which need not be part of set of strings S). We then define the scoring function as

  $$d = \sum_{S_i \in S} D(S_i, S_c) \tag{1}$$

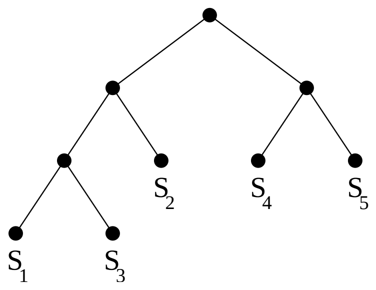
  where $D(S_i, S_j)$ is the weighted edit distance of strings S_i and S_j.

- **Pairwise distances**
  This uses

  $$d = \sum_{S_i \in S} \sum_{S_j \in S \setminus S_i} D(S_i, S_j) \tag{2}$$

  to score an alignment

- **Evolutionary Tree Alignment**
  Here the strings are arranged in a tree. The tree represents the evolutionary relationship between the sequences: strings that have a common parent node represent proteins/genes that have a common ancestor from which they evolved through different mutations. The score is then obtained by summing over the leaf and internal nodes of the tree. The figure below gives an example:



$$d = D(S_1, S_3) + D(S_4, S_5) + D(S_{13}, S_2) + D(S_{123}, S_{45}) \tag{3}$$

To calculate this sum, one needs to define distances between leaf nodes and internal nodes. For internal nodes no explicit string is available. The calculation can be done in different ways. This topic will be discussed in more detail in the next lecture.

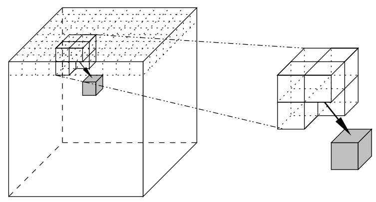
The main point concerning scoring methods is that there are different plausible methods. This means it is not essential to calculate an exact score, and it might be possible to use simplifying approximations. This is important since the scoring function affects the complexity of the algorithms used.

1.4 Methods

There are a number of approaches to solving the multiple alignment problem. The following gives a brief overview over the most commonly used techniques

1.4.1 Extensions to dynamic programming

This method is a generalization of the dynamic programming solution to the pairwise alignment problem. It can be shown that for p strings of length n we can fill a p-dimensional dynamic programming table in $O(n^p)$ time and space. Given this table, we can find the exact solution. However, this approach is not practical for p more than 3 or 4. To see why this is so, consider the 3-dimensional case:



In order to fill in the next cell in the table, given the previous ones, we have to look at 7 neighboring cells, instead of 3 as in the 2-dimensional case. It is easy to see that the number of cells we need to consider increases rapidly with increasing p ($O(n^p)$). This is exponential in the number of strings.

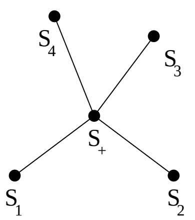
Center Star Method This heuristic method can be applied if the scoring scheme satisfies the triangle inequality, i.e. $D(S_i, S_j) \leq D(S_i, S_k) + D(S_k, S_j)$. The center star algorithm includes the following steps:

1. Find the center string $S_+ \in S$ which minimizes

$$\sum_{S_i \in S \setminus S_+} D(S_i, S_+) \tag{4}$$

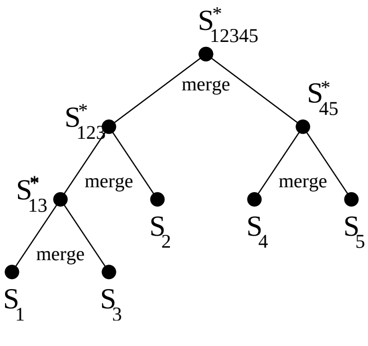
2. Add the remaining strings $(S \setminus S_+)$ one by one in such a way that the alignment of each is optimal with respect to S_+. Add spaces if needed

It can be shown that this method can find an alignment scoring at most a factor of two worse than the optimal alignment. The time complexity of the method is $O(k^2n^2)$.



Clustering Method The main idea behind this method is to iteratively merge multiple alignments of subsets of strings into a single multiple alignment of the union of the two subsets. There are different methods of choosing the subsets to be merged at each step. The clustering method described below uses agglomerative clustering:

1. Compute the pairwise distance $D(S_i, S_j)$ for all pairs in S.
2. Cluster using this distance matrix
   (a) Initially put all strings in their own cluster.
   (b) Repeat
      - Find the 2 "closest" clusters and merge them into a new cluster.
      - Find the best best alignment of the two strings from the subclusters and make it the string for the new cluster.



It is of course possible to apply other clustering methods.

2 Sequencing the Human Genome

2.1 The Human Genome Project

The Human Genome Project was formally begun by the academic community in 1990. The project was originally estimated to take 15 years, but advances in technological and significant contributions by the bio-tech industry have accelerated progress. A first draft of the human genome was announced jointly by the International Human Genome Project and Celera Genomics Corporation in June of 2000. The goals of the project include [2]

- determine the sequences of the 3 billion chemical base pairs that make up human DNA,
- identify all the approximate 30,000 genes in human DNA,
- store this information in databases,
- improve tools for data analysis.

Algorithmic methods have played and still play a very important part in this project. Below we give a brief overview of some of the methods used in the Human Genome Project.

2.2 Overview of basic approach

The basic approach in sequencing the genome of an organism is as follows

1. Divide the chromosomes into smaller fragments using "restriction enzymes".
2. Clone these DNA fragments using host cells (usually bacteria).
3. Find an exact map of a portion of the fragments (e.g. the first and last 600 base pairs (pb)).
4. Use sequence alignment to put the fragments back together.
5. Find the exact map of the required subset of the fragments.

In the following sections we will examine some of these steps in more detail.

2.3 Biological Tools

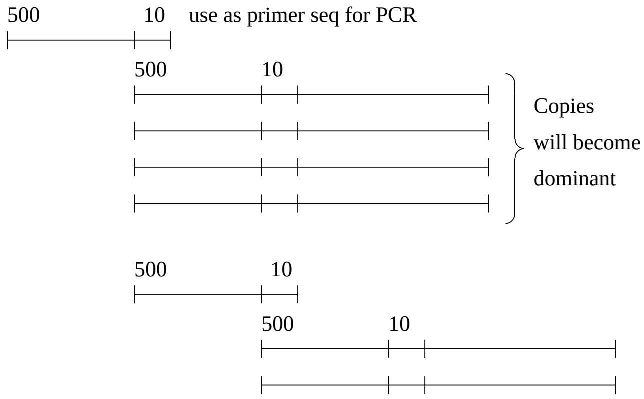
A number of biological tools are used while sequencing the genome of an organism:

- **Restriction enzymes** can cut chromosomes at particular sites. The sites are specified by a base-pair pattern, which can be of variable length (e.g. ACATTAG).
- **Radiation** is used to cut the chromosomes at random locations.
- **Cloning vectors** allow to copy DNA sequences of variable length. Examples of cloning vectors are
  - Yeast Artificial Chromosome (YAC), which can clone up to 1 million bp,
  - Cosmids, which can clone sequences of up to 40 kbp,
  - The BAC-PAC method, which allows to clone up to 200 kbp.
- **Polymerase Chain Reactions** (PCR) are used to clone an interval on a chromosome between two given strings, called *primer sequences*. The primer sequences can be chosen by the experimenter. This method is a cheap way to rapidly make essentially an unlimited number of copies of DNA between two "user"-defined points in the sequence.

A more detailed description of some of these techniques can be found in [4].

2.4 Direct Sequencing

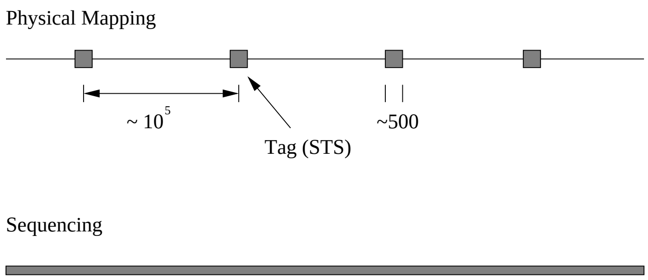
The most direct approach to determining DNA sequence is *direct sequencing*. Laboratory methods can sequence 300 to 500 bp from one end of a DNA string. In theory, this could be used to sequence the human genome: Start at one end of a chromosome, and sequence the first 500 bp. Use this sequence as a primer for PCR to make copies of DNA following the first 500 bp. The copies start after the first 500 bp, so in principle the process could be iterated until the sequence of the whole chromosome is determined.



Unfortunately, this does not work in practice: the experimental process is very slow. Each iteration takes days so that is impossible to sequence 3 billion bp. Another problem is that if the method gets stuck for some reason, the remainder of the string cannot be determined (unless a new starting point can be found). In practice the method does get stuck very often, particularly if there are repeated substrings which are quite common in DNA of higher organisms.

2.5 Mapping vs. Sequencing

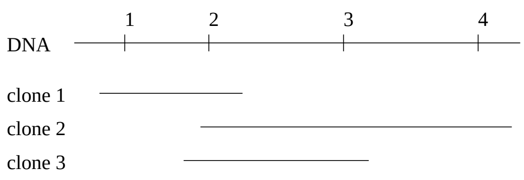
Physical mapping is the problem of finding the true physical location of certain "landmarks" or features of interest in the genome. This is different from sequencing, which aims at finding the entire sequence of all the base pairs in DNA. Physical mapping is an important step towards sequencing as we will see below.



The "landmarks" are called *sequence tagged-site* (STS). An STS is essentially a DNA sequence of length 200-300 bp whose ends (20-30 bp long) are unique in the entire genome. In the Human Genome project, a set of STS were chosen that are about $100,000$ bp apart. In this way any substring of length $100,000$ or more contains at least one STS. If we chop a chromosome into chunks of about that length, we will have a tag on each chunk. Given an STS map, this tag can then be used to identify where in the genome the chunks belong.

2.6 STS-content mapping and ordered clone libraries

To do STS content mapping, the DNA is first cloned and cut in random places. The cutting is done such that the fragments cover several STS. We also assume that we have significant overlap between fragments. PCR is then used to determine which fragments contain which STS. This information is organized in a binary table . Finally the STS are ordered, resulting in the map. The figure below gives a schematic outline for a small problem.



		STS			
		2	4	1	3
	1	1	0	1	0
clone	2	1	1	0	1
	3	1	0	0	1

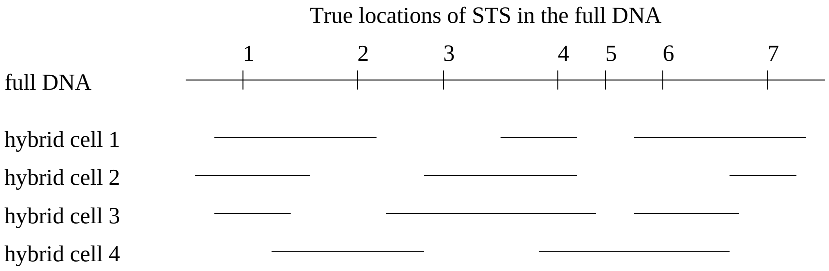
The ordering of the STS is an application of the *consecutive ones problem*: we want to reorder the columns in the binary matrix above so that we have a block of consecutive ones in each row:

		STS			
		1	2	3	4
	1	1	1	0	0
clone	2	0	1	1	1
	3	0	1	1	0

If there were no errors in the data, we could solve the problem in linear time using an algorithm due to Booth and Lueker. Unfortunately, the data is rarely error-free, so that other algorithms have to be used. Before discussing these algorithms, we look at another mapping method, which leads to the same problem.

2.7 Radiation Hybrid Mapping

In radiation hybrid mapping, the DNA is cut up and put into hamster cells. The hamster cells will replicate this DNA, along with their own. Over successive generations, the hamster cells replicates less and less of the human DNA. In the last generation, a single hamster cell contains about 10 fragments consisting of 15-20% human DNA. The fragments from each cell are disconnected and non-overlapping. However, fragments from different cells will cover other parts of the DNA and potentially overlap.



Again, we can determine which STS are contained in each of the fragments. We can then construct a binary matrix which summarizes this information:

		STS						
		2	4	5	3	1	7	6
	1	1	1	0	0	1	0	1
fragment	2	0	1	0	1	1	1	0
	3	0	1	0	1	1	0	1
	4	1	1	1	0	0	0	1

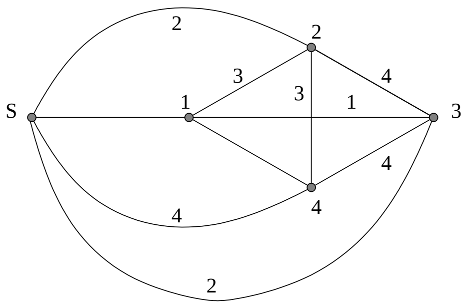
To establish the ordering of the fragments, we need to swap the columns of this matrix such that the number of blocks of ones in each row is minimized (or alternatively the number of transitions between 0 and 1 is minimal).

		STS						
		1	2	3	4	5	6	7
	1	1	1	0	1	0	1	0
fragment	2	1	0	1	1	0	0	1
	3	1	0	1	1	0	1	0
	4	0	1	0	1	1	1	0

We can see that the STS-content mapping problem with errors from before reduces to the same problem. Errors include false negatives, false positives and chimeric clones (joining of non-adjacent fragments) [1]. Errors of this kind result in a matrix with multiple blocks of ones on each row.

2.8 Traveling salesman formulation of STS ordering

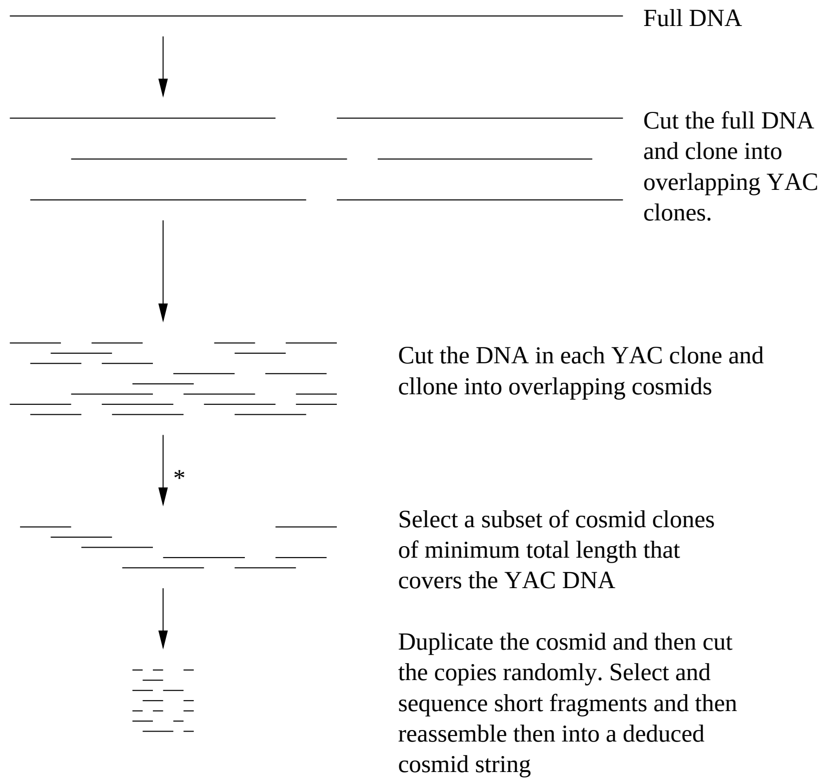
Both of the above STS ordering problems can be formulated as a traveling salesman problem: Each STS is assigned to a node in an undirected graph. In addition, one extra node labeled s is introduced and we add an additional column filled with 0 in the binary matrix. The edge between two nodes STS u and STS v has a weight corresponding to the hamming distance between the columns for STS u and STS v in the binary table. The edges between nodes s and u have a weight corresponding to the number of ones in the column for STS u.



It can be shown that a traveling salesman tour on this graph with edge weight w corresponds to a permutation of columns in the binary matrix with exactly $w/2$ consecutive blocks of ones. So finding the tour with minimal edge weight corresponds to solving the STS ordering problem.

2.9 Top-down, bottom up sequencing: the picture using YACs

The problem when trying to sequence the entire genome of an organism is that very long DNA strings can not be sequenced in one piece. For this reason more hierarchical or recursive approaches are used. The figure below gives a high level overview of a possible approach:



There are several algorithmic problems that arise during this process. Here we will only discuss fragment selection for sequencing (third to fourth row in the picture), and shotgun sequencing (the last row in the picture). More details on the other steps can be found in [1].

Fragment selection for sequencing Given a set of overlapping cosmid clones, it would be too expensive to sequence all of then. We need to find a subset of fragment of minimal total length that still covers the entire YAC clone. This subset is called the *minimum tiling path.* One way to efficiently finding this subset is assigning each fragment to a node in a directed graph. The weight of the edges is the overlap of the fragments on the vertices they connect (i.e. how many STS do they have in common). Finding the shortest path in this DAG is equivalent to finding the minimum tiling path.

Shotgun Sequencing The shotgun approach proceeds as follows:

1. Fragment the DNA into strings of about 500 bp.

2. Overlap detection: each pair of fragments, find the best score for overlap (on average one needs an overlap of about 5-10 bp). This is a variant of the local alignment problem and can be solved by dynamic programming.

3. To find the substring layout, one can use a bottom up clustering technique.

4. Deciding consensus: not all the characters assigned to a position in the layout a necessarily the same, so the best character for this position needs to be found. Different choices for solving this problem include using multiple alignment over small regions, using the most common character or even letting a human decide.

In the picture, the shotgun approach is only applied on the last step for sequences on the order of 100K-300K bps. However, the idea can be applied across the whole genome, called whole-genome shotgun sequencing. This idea was pushed by Craig Venter, Eugene Myers, and Jim Weber and is the basis for how Celera sequenced the Human Genome. Before Celera successfully sequenced the fruit fly Drosophila and then the human genome, many in the biology community did not believe the whole-genome shotgun sequencing would work. In fact in the mid 90s the NIH turned down a grant proposal for pursuing the work. It is largely the power of fast computers and smart algorithms that allowed Celera to prove the rest of the community wrong.

A paper by Eugene Myers gives a nice overview of the method [3].

3 Summary

Multiple alignment is widely used in biology to study problems such as characterizing families of genes or educing the evolutionary history from DNA or protein sequences. Aligning multiple sequences is NP-hard, and a number of heuristic methods, such as the Center Star Method or the Clustering methods have been developed to solve it.

Sequencing the Human Genome has been one of the major milestones in recent biological research. The human genome database will be invaluable for future biological and medical research. String algorithms, such as multiple alignment, have played an important role at various steps of the project and have strongly contributed to the incredible speed with which the project near completion.

References

[1] Dan Gusfield, Algorithms on strings trees, and sequences, Cambridge University Press, Cambridge, 1997.

[2] http://www.ornl.gov/hgmis/

[3] Gene Myers, *Whole-Genome DNA Sequencing*, IEEE Computational Engineering and Science 3, 1 (1999), 33-43.

[4] L. Streyer, *Biochemistry*, Freeman (1995)