12.1. Introduction

Sequence Assembly is the process of constructing the “best guess” clone (cosmid or BAC) sequence from a set of overlapping reads of the clone. The problem is complicated by the fact that there could be errors in read sequences and the fact that there could be many repeats in the clone sequence. In the last lecture, the broad outline of the sequence assembly process was described as below:

1. Compare reads pairwise to determine potentially overlapping reads.
2. Determine layout, i.e. the pattern of overlaps.
3. Determine the consensus sequence.

In this lecture, we examine each of the above steps in detail.

12.2. Finding Pairwise Overlaps

Because the reads are overall fairly accurate (about 99% accurate), finding matches is much simpler than finding simple subtle sequence similarities. We can assume that two overlapping reads will have perfectly matching words of significant lengths in common. The strategy for identifying overlapping reads is as follows:

- Find all pairs of reads with perfectly matching words of length greater than $n$.
- For each such pair, each exactly matching word defines a diagonal in Smith-Waterman (S-W) matrix. Enlarge each such diagonal to a band to allow for imperfect matches. Take the union of such bands and do a S-W search of each band. (??)
- If S-W score exceeds a threshold, record an alignment.
Finding Word Matches

Using a data structure called suffix trees, word matches could be found in $O(m)$ time, where $m$ is the number of matches [1]. But this data structure is very space-inefficient. A space efficient, but slower solution is known, which builds one tree for the entire set of reads [2].

A simpler method to find word matches, found more useful in practice is the following:

1. Construct a list $P$ of pointers to all positions in all reads.
2. Define an order relation $<$ on pointers. Say $p < q$ iff pointer $p$ points to a string lexicographically lesser than the substring $q$ points to.
3. Sort $P$ by above partial order $<$. The matching words are found by examining elements of $P$ that are near each other in the sorted list.

An example:
Read: A A C A G T G G A

Unsorted List:
P1 : A A C A G T G G A
P2 : A C A G T G G A
P3 : C A G T G G A
P4 : A G T G G A
P5 : G T G G A
P6 : T G G A
P7 : G G A

Sorted list :
P1 : A A C A G T G G A
P2 : A C A G T G G A
P4 : A G T G G A
P3 : C A G T G G A
P7 : G G A
P5 : G T G G A
P6 : T G G A

12.3. Constructing The Layout

Constructing the layout involves determining the pattern of overlaps of reads and determining how the reads align. This is the hardest step in assembly process. What makes it hard is a combination of two factors:
errors in the read data and the possibility of repeats in the sequence. For example, in figure 1, the tough part
is to decide if the discrepancies in the reads are “real”, or if they are due to base-calling errors.

Figure 12.1: Are discrepancies real, or due to data errors?

Phrap, a software developed by Phil Green, uses error probabilities to decide this. Phrap takes the fol-
lowing approach:

1. Re-score each pairwise alignment, taking error probabilities into account.

2. Compute the log likelihood ratio \( \text{LLR} \) for

\[
H_1 : \text{the probability of observed data assuming overlap is real,}
\]

\[\text{vs. } H_0 : \text{the probability observing the data assuming reads are from 95 \% similar repeats. (This figure of 95 but has been found to work well in practice.)}\]

3. Construct the layout using a greedy algorithm, “with revisions”.

Assumptions

This approach makes the following assumptions.

1. Alignment positions are independent of each other.

2. When a base-calling error occurs at a read position, the base calls in the two reads disagree at that position. (Correction required to avoid this assumption is very small).

3. Errors in reads are independent of each other.

Consider each position in alignment. There are two cases to consider: either the base cells match or they don’t.

**Case 1:** Base cells agree (or match).

\[
Prob(\text{agree}|H_1(\text{overlap})) = (1 - e)(1 - f).
\]

\[
Prob(\text{agree}|H_0(\text{repeat})) = (1 - e)(1 - f) \times 0.95
\]

\[LLR = \log(1/0.95)\]

**Case 2:** Discrepancy between base cells.
The LLR for the entire alignment is the sum of above terms over all alignment positions.

12.3.1. Greedy Algorithm For Layout Construction

The layout is maintained as a data structure called contig. Each contig consists of a set of reads and for each read, an orientation (complemented or uncomplemented) and an offset with respect to the start of the contig is determined. Say, the set of reads is represented by a set of vertices in a graph, and there is an edge between a pair of reads if an overlap is determined between the reads. Then, a contig corresponds to the connected component in the graph thus constructed. See figure 2.

Figure 12.2: Contigs in a layout

The greedy algorithm to construct the layout considers the reads in some order and seeks to grow each contig at each step. A read is added to a contig if it is likely to overlap with another read in the same contig. Following are the main steps of the greedy algorithm:

1. Initially, each read forms a separate contig.
2. Process read pair alignments in decreasing LLR score order.
3. For each pair, consider a potential merge between two contigs. The selected pair defines an offset. Consider all pairs of reads (one from each contig) consistent with this offset. Pairs are consistent if
   - There are no negative LLR scoring pairs.
   - The positive LLR scoring pairs “cover” the region of overlap. See figure 3.
4. If the pair is consistent, accept the merge of the two contigs.

In practice, the greedy algorithm usually gets the layout correct (because of the ability of LLR-scores to distinguish most repeats). Even when it fails, the layout is usually mostly correct, with a small number of false joins. In such cases, the layout is broken at all weak joins (places where reads have positive LLR scores to other locations in layout). Then, all possible ways of rejoining the pieces are considered and the “best” one (i.e. the highest scoring on by an “appropriate” criterion) is chosen.
12.4. Construction Of Contig Sequences

Final sequence is constructed as a mosaic of highest quality regions of the reads (it is not a “consensus” sequence). The main steps are as follows:

- Construct a weighted directed graph:
  - Nodes are (selected) read positions.
  - Edges are of two types:
    1. Bidirectional: These edges are between aligned positions in two overlapping reads. The weights of the edges are zero.
    2. Unidirectional: These are between positions in the same read in the 3” to 5” direction. Weight of these edges is the sum of phrap quality measures for bases. (see figure 4)

- Determine the contig sequence from the read segments that lie on the highest weight path through the graph. The highest weight path can be obtained using Tarjan’s algorithm (with modifications to deal with non-trivial cycles).

12.5. Dealing With Real Data Sets

The sequence assembly is complicated because of the following problems in the real data sets:

1. Chimeric reads.
2. Unremoved vector sequence.

3. Deletion clones.

The approach usually taken is to identify such problem reads upfront (from the results of pairwise matches). The reads are either removed altogether from the assembly process or treated specially.

References

[1] Gusfield Text on Strings