SUTTA

Scoring-and-Unfolding Trimmed Tree Assembler

Bud Mishra and Giuseppe Narzisi
Courant Institute of Mathematical Sciences, New York University

March 2009

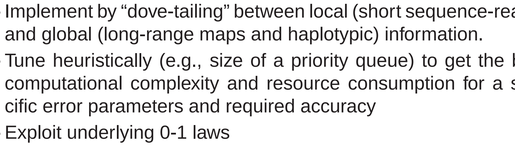
NYU Bioinformatics Lab
10th Floor, 715 Broadway
New York, NY 10012

Courant Institute of Mathematical Sciences
New York University
251 Mercer St., New York, NY 10012

Sequence Assembly Problem

Definition [Genome Sequence Assembly Problem] *Given a set of fragments/reads* $F = \{f_1, f_2, \ldots, f_n\}$ *find a reconstruction* R *and a valid layout of the reads* L *such that the following set of properties (oracles) are satisfied :*

- *The observed distribution of fragment reads start point,* D_{obs}*, has the minimum deviation from the source distribution* D_{src}
- *The distance between mated reads must be consistent with the size of the fragments generated.*
- *The observed distribution of restriction enzyme sites,* C_{obs} *is consistent with the distribution of experimental optical map data* C_{src}*.*



History and Motivations

1. Single Molecule Methods such as Optical Mapping.
2. Next Generation Sequencers.
3. SMASH (Single Molecule Approach to Sequencing by Hybridization) combining:
   (a) Optical Mapping
   (b) Binary Maps (with PNA probes)
   (c) Positional Sequencing by Hybridization (using a Beam Search Algorithm)

SUTTA provides a more general approach for haplotypic whole genome sequencing.

SUTTA assembler

Main Ingredients

- Exhaustive
  - Not a greedy algorithm.
  - Avoid getting stuck with a locally best solution

Implementation

- Use Branch-and-Bound (or Beam-Search) algorithm to improve algorithmic complexity.
  - Provide bounds and allow pruning of unpromising regions/directions.
  - Implement by "dove-tailing" between local (short sequence-reads) and global (long-range maps and haplotypic) information.
  - Tune heuristically (e.g., size of a priority queue) to get the best computational complexity and resource consumption for a specific error parameters and required accuracy
  - Exploit underlying 0-1 laws
  - Parallelize in a straight-forward way

Scoring

- Use a "score" function to choose the best global solution.
  - Achieve high accuracy
  - Model the "error processes" in the score, consisting of Bayesian likelihood and penalty functions
  - Use side-information (e.g., optical maps, mated pairs, base-content, homologous reference sequences, diluted sequencing, low complexity representation, etc.) to sharpen the score function
  - Use empirical-Bayes method to decide the statistics (null-model, threshold, p-values, base- or sequence-quality)
  - Agnostic to the underlying short- and long-range technologies, while being able to mix-and-match technologies

Pseudo-Code

Algorithm 1: SUTTA - pseudo code
Input: Set of N reads
Output: Set of contigs

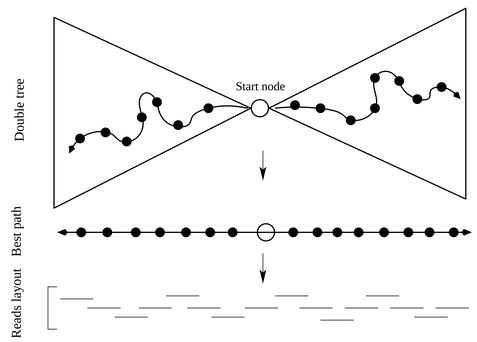
```
1  F := ∅;                                   /* Forest of trees */
2  C := ∅;                                   /* Set of contigs */
3  B := ⋃_i^N R_i;                           /* All the available reads */
4  while (B ≠ ∅) do
5      R := B.getNextRead();
6      if ( !R.isUsed() && !R.isContained() ) then
7          DT := create_double_tree(R);
8          F.add(DT);
9          Contig CTG := create_contig(DT);
10         C.add(CTG);
11         CTG.layout(); /* Compute layout of the contig */
12         B := B \ {CTG.reads};             /* Remove used reads */
13     else
                                             /* jump to next available read */
14     end
15 end
16 return C
```

Node expansion

1. Start with a random read (It will be the root of a tree; Use only the read that has not been "used" in a contig yet, or that is not "contained")
2. Create RIGHT Tree: Start with an unexplored leaf node (a read) with the best score-value; Choose all its non-contained "right"-overlapping reads and expand the node by making them its children; Compute their scores. (Add the "contained" nodes along the way, while including them in the computed scores; Check that no read occurs repeatedly along any path of the tree). STOP when the tree cannot be expanded any further.
3. Create LEFT Tree: Symmetric to previous step.

Contig Construction

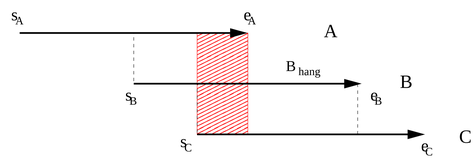
- The expand node routine is applied twice to generate LEFT and RIGHT trees for the start read.
- Next, the best LEFT path is concatenated with the root and the best RIGHT path to create a globally optimal contig.



Overlap Score

- "Weighted transitivity" score that formulates the following intuition: if read A overlaps read B, and read B overlaps read C, we will score those overlaps strongly if in addition A and C also overlap. This scoring approach implicitly assumes that the coverage is higher than 3.

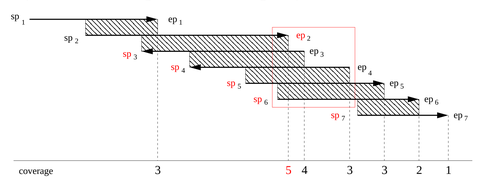
$$\texttt{if}(\pi_{(A,B)} \wedge \pi_{(B,C)})\texttt{then}\{S_{\pi_{(A,B,C)}} = S_{\pi_{(A,B)}} + S_{\pi_{(B,C)}} + (\pi_{(A,C)} ? S_{\pi_{(A,C)}} : 0)\} \quad (1)$$



- A simple generalization for higher coverage is obvious.
- This score cannot resolve repeats or haplotypic variations. Solution: augment the score with information for optical map alignment or mated-pair distances to put an appropriate reward/penalty term.

Dynamic Coverage Score

- **Observation**: compressed (expanded) regions are characterized by an increase (decrease) in the depth of coverage compared to the expected average coverage of the shotgun process.
- **Idea**: penalize solutions whose observed coverage deviates from the expected coverage of the shotgun process.



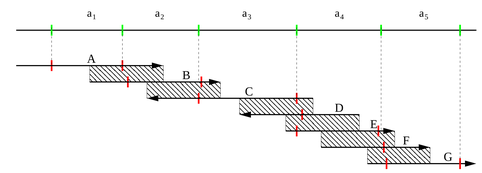
Optical Map Score

- **Observation**: restriction enzymes cut at precise locations in the genome. Let $< a_1, a_2, \ldots, a_n >$ be the *restriction map* obtained by a restriction enzyme digestion process.
- **Idea**: Organize the restriction sizes a_i into n-tuples. Build a hash table according to:

$$\mathcal{H}(a, b_1, \ldots, b_n) = < \left\lfloor \frac{b_1}{a} \times \alpha \right\rfloor, \left\lfloor \frac{b_2}{a} \times \alpha \right\rfloor, \ldots, \left\lfloor \frac{b_n}{a} \times \alpha \right\rfloor > \quad (2)$$

store a in the corresponding slot (with possible collisions).

- Create an in-silico map of the candidate solution and score it according to the number of *hits* that its n-tuples have in the hash table.



Streptococcus suis strain P1/7

Shotgun data from Sanger Institute (2,007,491 bp)

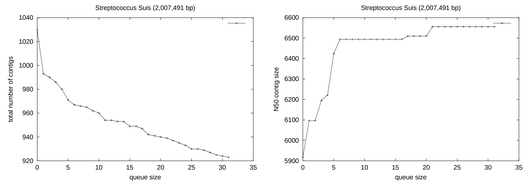


Figure 1: Queue size analysis

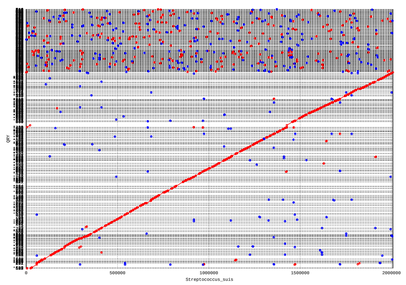


Figure 2: DotPlot

References

[1] Sun Kim, Haixu Tang and Elaine R. Mardis. Genome Sequencing Technology and Algorithms. *Artech House Publishers,*, 1 edition (October 31, 2007).

[2] Kececioglu and Myers. Combinatorial algorithms for DNA sequence assembly. *Algorithmica* (1995) vol. 13 (1-2) pp. 7-51

[3] Adam M Phillippy et al. Genome assembly forensics: finding the elusive mis-assembly. *Genome Biology* (2008) vol. 9 (3) pp. R55