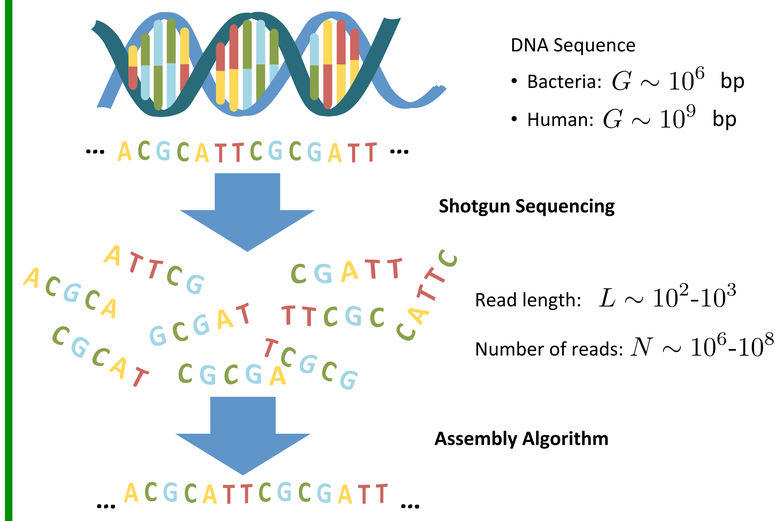
‘Not-So-Greedy’ Algorithm for DNA Shotgun Sequencing

Sam Kim, Ilan Shomorony, Thomas Courtade
EECS, University of California, Berkeley

Objective

DNA shotgun sequencing technologies generate short reads from the underlying sequence. Given these reads, an assembly algorithm attempts to reconstruct the original DNA sequence. We present a linear-time assembly algorithm that is near-optimal from an information-theory standpoint.

DNA Shotgun Sequencing



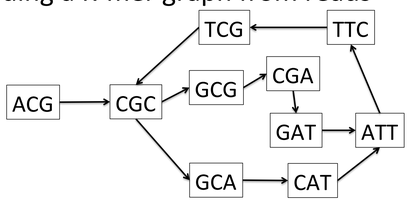
DNA Sequence
- Bacteria: $G \sim 10^6$ bp
- Human: $G \sim 10^9$ bp

Read length: $L \sim 10^2$-10^3

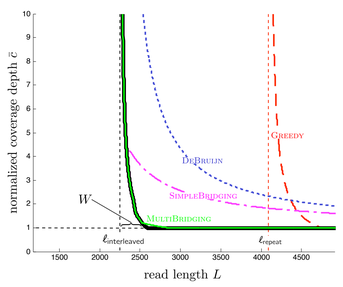
Number of reads: $N \sim 10^6$-10^8

Existing Algorithms

- Most algorithms (EULER, Allpaths, Velvet, ABySS) based on:
  - Building a K-mer graph from reads

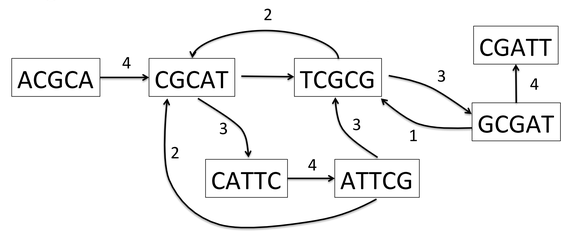


  - Finding an Eulerian path
- MULTIBRIDGING [1]:
  - Picks K dynamically
  - Uses reads to resolve repeats
  - Nearly optimal from information-theoretic perspective
  - Time and memory complexity are $O\left((L-K)NK\right)$



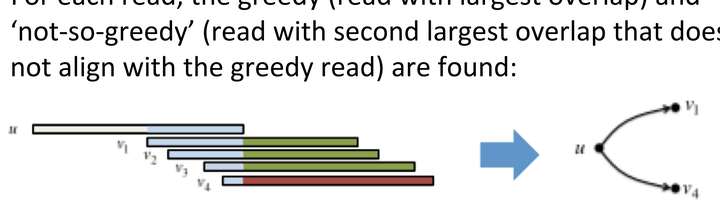
A “Not-so-greedy” Algorithm based on Read-Overlap Graphs

- Read-overlap graph contains one node per read
- Edge weights indicate size of overlap



Graph Construction

- In general, read-overlap graphs have $O(N^2)$ edges
- We generate a sparse subgraph, where each node has in-degree and out-degree at most 2:
- For each read, the greedy (read with largest overlap) and ‘not-so-greedy’ (read with second largest overlap that does not align with the greedy read) are found:



- Read-overlaps are efficiently found using incremental (decremental) Rabin-Karp-type hash to compute all reads’ prefixes and suffixes in order for $O(1)$ look up time on average:

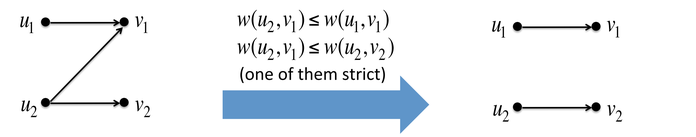
$$H(r) = \sum_{i=1}^{l} \phi(r[i])\alpha^{l-i} \mod p$$

where ϕ is a one-to-one function that assigns different numeric value to each base, *p* = 1000000007, α= 13.

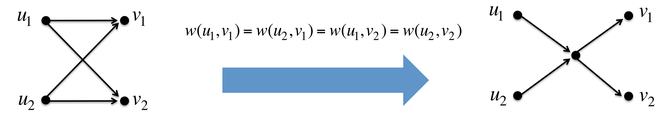
- Time and space complexity to construct graph: $O(LN)$
- Potentially, worst case runtime can be $O(N^2)$ if we end up hashing all reads’ prefixes and suffixes to the same bucket in our hash table, which can occur when finding overlaps at the noisy level.
- To account for this, we limit the number of read prefixes or suffixes in each bucket of the hash table to 2 assuming necessary conditions are satisfied.
- *Why “not-so-greedy”?* It is known that greedy assembly can fail, but this algorithm provably succeeds.

Identifying X and Z Structures

- Algorithm iteratively performs two operations on the graph in order to remove unnecessary transitive edges that have in-degree and out-degree equal to 2:
  - Delete middle edge from a Z-structure:



  - Transform X-structures:



- When these operations are no longer possible, we condense unambiguous edges and look for an Eulerian path
- Same sufficient conditions for success as MULTIBRIDGING:

> **Theorem:** Proposed algorithm correctly reconstructs the sequence, provided:
>
> - Interleaved repeats are bridged
> - Triple repeats are all-bridged

- Assembly is only possible only if [1]:

$$L > \ell_{\text{critical}} = \max(\ell_{\text{triple}}, \ell_{\text{interleaved}})$$

- The sufficient conditions in the theorem are a modest strengthening of the necessary conditions.
- Current C++ implementation needs only ~100ns/character.

Future Goals

- Handle the case when reads are noisy
- Build the “smallest” overlap graph that is guaranteed to contain the true sequence provided *t*-ary repeats are all-bridged

[1] G. Bresler, M. Bresler and D. Tse “Optimal Assembly for High-Throughput Shotgun Sequencing”
[2] J. T. Simpson and R. Durbin “Efficient construction of an assembly string graph using the FM-index”
[3] I. Ben-Bassat and B. Chor “String graph construction using incremental hashing