CS 6968: Data Str Alg Scalable Comp Spring 2023

Lecture Genomic Assembly Graphs — Apr 3, 2023

Prof. Prashant Pandey *Scribe: Hamza Sheikh*

1 Overview

In the last two lectures we discussed graphs scalability issues. That encompasses graph representation and various computation on graphs like existance of node or nearest neighbor. We also discussed streaming graphs and incremental computations, as well as the API guarantees on such implementations.

In this lecture we discussed how we can assemble genome representations using graphs. Two genomic assembly graphs where introduced:

1. DBG: de Bruijn Graph [1]
2. OLC: Overlap-layout consensus [2]

2 Assembly in the real world

2.1 Assembly Pipeline

Sample $\Longrightarrow$ Instrument (transfers biological structure into readable data)

$\Longrightarrow$ Reads (short sequences of genome) $\Longrightarrow$ **Assembly** (comparison to reference genome)

$\Longrightarrow$ result: genome of sampled entity

Depending on the instrument used, reads can be:

1. Short reads: 100-200 bases
2. Long reads: 5,000-10,000 bases

2.2 Errors

The instrument creates errors while reading the genome, these are:

- **Insertion Error:** base does not exist in actual genome
- **Deletion Error:** base exists but was not read
- **Substitution Error:** Base different than the actual value (e.g. A instead of T)

The probability of these errors occurring varies between the types of reads, with a 0.5% to 1% for short reads, and 2% to 10% for long reads.

To have more accurate results, the instrument reads every location multiple times. This is called **sequencing depth**, and is typically between 30 to 50 times in practice.

2.3 Types of Assembly

- **Reference based:** We already have a reference genome of the species that's used to assemble the sample.
- **De Novo:** No reference genome for the species, so an algorithm needs to be built to measure the tendency of reads to be located next to each other. (e.g. reads overlap, algorithm matches prefixes to suffixes of different reads)

2.4 de Bruijn Graph

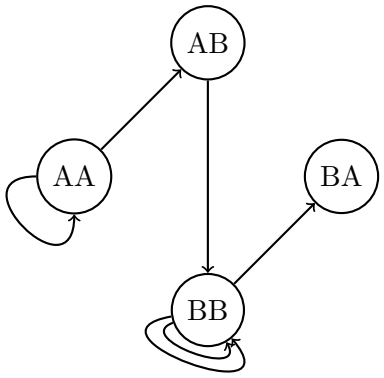
After reads are transformed into k-mers, a de Bruijn graph is constructed by creating a node for each distinct (k-1)-mer, where a node's edges represent the link to a node holding its suffix.

2.4.1 Example

Genome: AAABBBBA

3-mers: AAA, AAB, ABB, BBB, BBB, BBA

2-mers: AA, AA, AA, AB, AB, BB, BB, BB, ..., BA



The genome can be reconstructed by a **Eulerian walk**. which is a walk crossing each edge exactly once. This is a simple graph and the Eulerian walk can be seen easily, but in practice, the walk is not unique and one of the toughest problems to solve in assembly.

2.4.2 Eulerian Walk Definitions

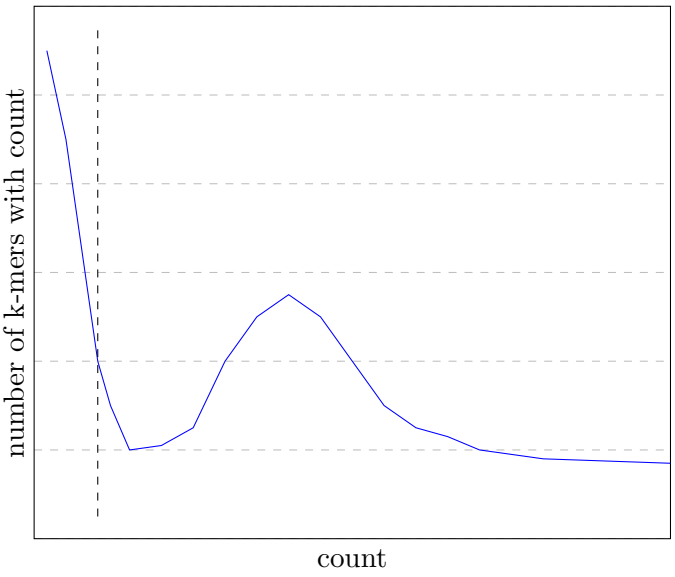
Balanced node: $indegree = outdegree$

Semi-balanced node: $|indegree - outdegree| = 1$

Connected Graph: each node can be reached by some other node

Eularian Graph:: A directed connected graph that has at most 2 semi-balanced nodes (start and end nodes) and all other nodes are balanced. (e.g. graph above).

2.5 Error Correction



The graph above shows a distribution of the number of distinct k-mers present. The graph shows that there are lots of k-mers that appear only a few times (before the dotted line). These represent the erroneous k-mers, as an error in a read will generate k-mers that'll appear only in that error read. The second spike is usually proportional to the sequencing depth and the k-mers close to that number or more likely to be true. Lastly, the tail represents the most frequent k-mers in the genome, which corresponds to the repeating regions in the genome.

2.6 Data Structures to represent dBG

- **Filters + hash table** (efficient way to find k-mers with low count)
- **dBG based FM-index** (compressed dBG)

2.6.1 Refining

In practice, coefficient of variance is used as a metric for refining constructed graphs to have more obvious configs.

Refining also includes removing island tips and bubbles.

2.7 Practical issues in de Bruijn Graphs

1. Repeats cause mis-assemblies, as the eularian walk would not be unique, creating multiple possibilities of the genome sequencing. Moreover, short k-mers lose the ability to resolve repetitive regions.
2. Errors in k-mer construction, steming from reads' errors, introduce false k-mers, and the constructed de Bruijn graph might be Eularian to start with

There is a tradeoff between increasing and decreasing the size of k-mers: When k is small, the error in each k-mer/node tend to be lower, but it will lose its locality information.

When k is large, the node preserve the locality information in the regions of the genome while providing more distinct k-mers, but the errors in each node tends to be higher, and it is harder to handle performance wise.

In practice, k is typically between 30 to 50.

2.8 Overlap-layout-consensus (OLC)

2.8.1 Overlap

Build the overlap graph on reads without chopping them into k-mers. No locality information loss.

create an edge between to reads/nodes when their suffix-prefix match.

To build an overlap graph, Calculating the distance between nodes (matching prefix-suffix) costs $O(n^2.m^2)$ where n is the number of reads and m is the size of each read.

In practice, LSH/minHash based techniques are used instead of brute force distance calculation with all pair of nodes. [3]

Then the expensive overlap calculation is only done on the reads that maps to the same bucket to create the overlap graph.

2.8.2 Layout

During the layout phase, the overlap graph is cleaned by removing errenous edges. Unresolved regions are then treated seperately with heuristics.

2.8.3 Consensus

Map contigs back to the actual reads to have higher confidence. When majority agrees then accept the contig. This is not ideal as reads still have the errors.

References

[1] Bruijn, de, N. G. A combinatorial problem *Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam*, 49(7):758–764., 1946.

[2] Peltola, Hannu and Soderlund, Hans and Ukkonen, Esko SEQAID: A DNA sequence assembling program based on a mathematical model *Oxford University Press*, 1946.

[3] Heng Li. Minimap and miniasm: fast mapping and de novo assembly for noisy long *Bioinform.*, 32(14):2103–2110, 2016.