SARS-CoV-2 Assembled Genome

Wiley Bui - Guogeng Li - Yunong Xia - Weijie Zhang

Abstract

Coronavirus disease 2019 (COVID-19) has been rapidly spreading across the globe since its first identification. This unprecedented pandemic has led to a tragic loss of human life and has presented a colossal challenge to public health research. The disease is caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), whose genome is of great importance to the elucidation of viral dynamics and the search for a proper treatment. Assembly of short reads from sequencing techniques into correct order is a fundamental task of bioinformatics research. A number of methods have been developed over the past few decades, among which Euler path based techniques have proved efficient and reliable. In this project we construct De Bruijn graphs from SAS-Cov-2 whole genome sequencing reads and implement an algorithm to establish an Euler path in each graph to explore the factors such as the length of K-mers that might affect genome assembly. In addition, we explore methods to visualize the graph and validate our results with a reference genome. Here, we report the results of our tentative approaches to compare the impact of different lengths of K-mers and the results of De Bruijn graph construction and visualization.

Summary of Previous Findings

Investigations of the SARS-CoV-2 genome have added valuable information to the understanding of the biology and epidemiology of COVID-19. Since the first genome sequencing study was published in January 2020, a rapidly growing number of related studies have been conducted regarding different research questions. Harilal, Divinlal et al. (2020) compared the genome resulting from qRT-PCR techniques or whole genome sequencing approaches (WGS). They conducted shotgun transcriptome sequencing using RNA extracted from COVID-19 patients and compared to the targeted whole genome sequencing using criteria such as viral detection, scalability and cost. Their results showed the effectiveness of WGS with enrichment methods. Furthermore, they demonstrated that the genome sequences of SARS-CoV-2 can be used to conduct phylogenetic analysis which will provide further information about viral origin, variants, and spreading.

In a recent review article, Chiara, Matteo et al. (2020) compared various next generation sequencing techniques of SARS-CoV-2 genome including genome assemblies. The viral genome is compact sized and does not contain large repetitive regions, which makes it relatively straightforward to make use of most existing state-of-the-art approaches. In this project we

were provided with raw SARS-CoV-19 genome data. In principle, however, data obtained from sequencing of patient samples should be enriched for the viral genome. This process is much more complex since it aims for the assembly of metagenomic reads. Thus, filtering and enrichment prior to assembly could serve as an efficient way to circumvent this issue.

Results

We used trimmomatic to clean raw data. Each read was treated as single-end sequencing and the cleaned data was stored separately. After cleaning the data, we found all kmers in each read and then cleaned the kmers list by throwing away kmer with frequency less than the threshold(usually 1 or 2). These kmers were then used to find their corresponding k-1 mers. We used kmers to represent edges in the de bruijn graph and the two nodes connected by the edges are the corresponding k-1 mers. The visualization examples are shown in figure 1, 2, and 3. There are some disconnected subgroups inside.

To report the assembled genome/contigs, we used the Eulerian path to traverse through the De Bruijn graph by using the Hierholzer's algorithm that utilized backtracking and depth-first traversal. First, the starting node must be picked. If a node has an extra outgoing edge then that node represents a starting node; otherwise, the first k-mer from the k-mer list gets to be the starting node. Second, we visit (traverse) through the graph by the depth-first traversal from following the starting node’s edges. Then we use an edge from the node visited to visit through its other edges by recursion until there are no edges left. However, this method doesn't mean all edges are discovered, so we use backtracking to discover the remaining unvisited edges until all edges are found (even for extra edges). When there is no outgoing edge left in a node, we prepend that node to our path, resulting in the assembled genome/contigs. To solve for duplicate edges, as long as a node in the graph has at most 1 (outgoing edges - incoming edges) as well as 1 (incoming edges - outgoing edges) while all other nodes have equal incoming and outgoing edges, the Hierholzer’s algorithm works. However, when the condition does not satisfy, the algorithm returns the path from the starting node until this current node.

Since a graph is an Eulerian if a node from the De Bruijn graph has at most 1 outgoing edge and at most 1 incoming edge while all other nodes must have the equal amount of incoming and outgoing edges, our De Bruijn graph unfortunately does not meet the requirement. We tried different k values and different k-mers cleaning threshold, but none of them have Eulerian Walk. The possible reasons are: 1. Sequencing errors are not cleaned completely and the graphs are disconnected. 2. Our path finder algorithm does not try to connect disconnected subgroups. These may be solved by: 1. Comparing similar kmers to each other by multiple sequence alignments, and correcting residuals by most frequent one. 2. For nodes without entrance, try

to find its longest walk in the graph and then return the longest path among those of the non-entrance nodes.

The result incomplete contigs are then aligned with the reference genome by BLAST. The example result is shown in figure 6. The alignment results get better when k is from 1 to 20. When k is above 20, there is no obvious trend as k increases. We do find they align well when k = 30, as no obvious cut off shown in the alignment dot plot.

The future work includes improving path finding algorithms so that it can discover the longest path in the graph. We could also remove redundant edges in the graph to make it more balanced.

Conclusion

We assemble Coronavirus genome by using De Bruijn Graph. The effects on k impacts dramatically on the De Bruijn graph nodes and edges as k increases from 0 to 20, but anything above k = 20 grows sublinearly. The contigs are generated by finding Eularian paths from the De Bruijn graph by exploring the length of k-mers on how it might impact the genome assembly. We later compare the path against the reference genome. As a result, the genome assembly is mostly proportional to the reference genome for most k values (*Figure 6*).

Currently, we have some unit tests for smaller graphs that work successfully on the Eulerian path but not with larger ones from the De Bruijn graph due to multiple uneven numbers of outgoing and incoming edges. We do have an Eulerian circuit checker to detect if it can create a path or not, but not how to resolve if there can't be a path. For our future work, we suggest implementing a balancing algorithm to make every node have the same number of outgoing edges with the incoming edges. Therefore, the Eulerian circuit exists, which gives us the path by using the Eulerian graph / Hierholzer algorithm.

Methods

We clean the provided raw FASTQ file from the SAS-Cov-2 genome data[1] using a trimmomatic filtering tool. Next, we put the data sequences into a list of k-mers, where the list only accepts the k-mers if they show up more than two times. We then use the list of k-mers to create the De Bruijn graph; each edge represents a k-mer and the nodes connected are two k-1 mers.

After generating the De Bruijn graph, we begin to utilize the Hierholzer algorithm with a start node and traverse through the graph by following the edges from that starting node by recursion to visit until there are no other unvisited edges at the end node. Following this, we

use backtracking to discover the remaining unvisited edges until all edges are found by simply checking if all incoming and outgoing edges are both equal to 0. Once the condition satisfies, the path of the graph gets generated and saves to the Results/ folder. Furthermore, we take the path results and graph them out by using the Graph-Visualization module to analyze the outcome.

Figures

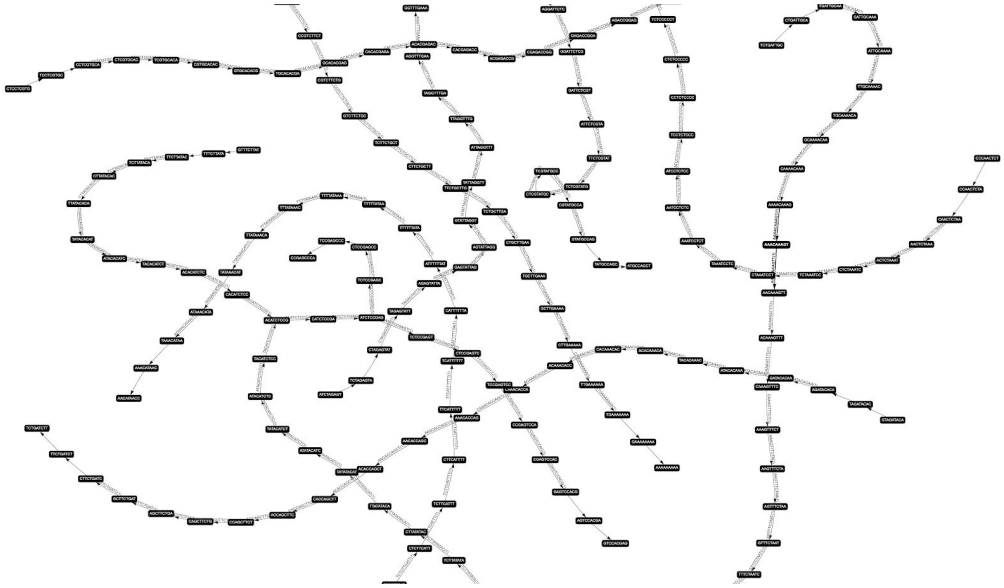

Figure 1 De Bruijn Graph with k = 10

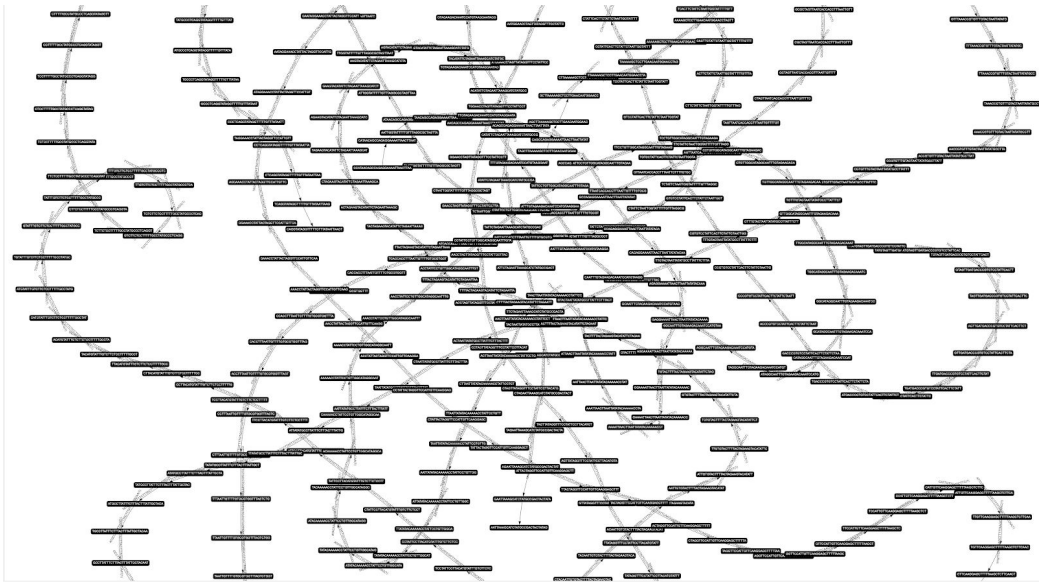

Figure 2 De Bruijn Graph with k = 20

Figure 3 De Bruijn Graph with k = 50

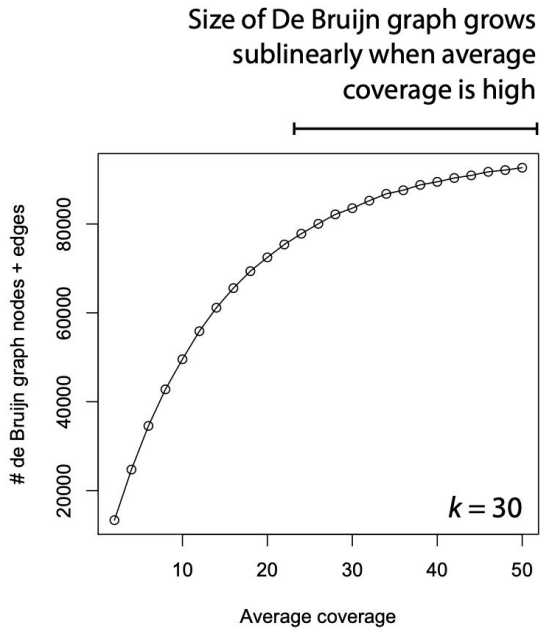

Figure 4 The effects of k

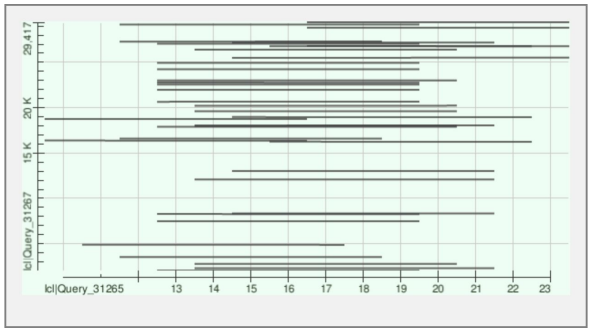

Figure 5 Extreme case (K = 2)

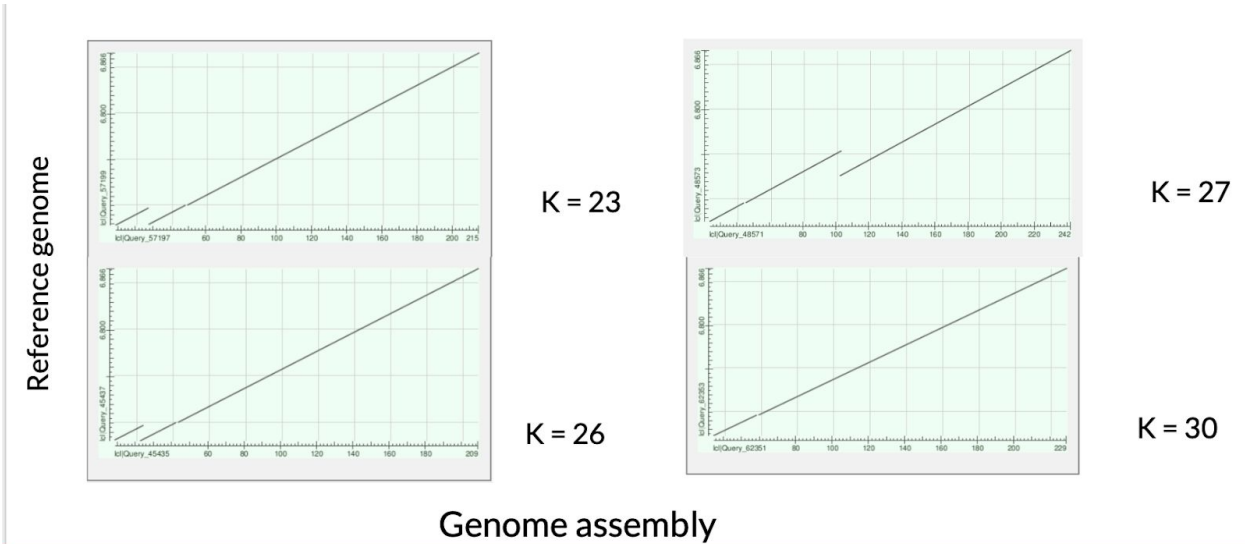

Figure 6 Alignment to reference genome

Acknowledgements

Wiley B	Worked on De Bruijn graph, but mainly on Eulerian path & its unit tests
Yunong X	Worked on De Bruijn graph construction and contig alignments.
Weijie Z	Worked on raw data tidying and contig alignment and visualization.
Guogeng Li	Visualize De Bruijn graph and implement Eulerian path by Hierholzer algorithm.

References

"Hierholzer's Algorithm for Directed Graph." GeeksforGeeks, 5 June 2020, www.geeksforgeeks.org/hierholzers-algorithm-directed-graph/.

"De-Bruijn-Graph-Genome-Visualization." Github, 10 Feb, 2019, https://github.com/schostac/De-Bruijn-Graph-Genome-Visualization.

"Igraph R version" https://igraph.org/r/doc/make_de_bruijn_graph.html.

Harilal, Divinlal et al. "SARS-CoV-2 Whole Genome Amplification and Sequencing for Effective Population-Based Surveillance and Control of Viral Transmission." *Clinical chemistry* vol. 66,11 (2020): 1450-1458. doi:10.1093/clinchem/hvaa187

Chiara, Matteo et al. "Next generation sequencing of SARS-CoV-2 genomes: challenges, applications and opportunities." *Briefings in bioinformatics*, bbaa297. 7 Dec. 2020, doi:10.1093/bib/bbaa297