Pangenome Graphs: An Overview

Justin Mau

In the field of computational biology, genome sequencing and analysis has always been an important factor. This is used in many instances, including studying genetic similarities between species, predicting protein structures and functions, and tracking genetic mutations in viruses. However, as time has gone on, the standard center-genome focused approach for multiple sequence alignment and analysis has shown that biases can easily be introduced when just one genome is treated as representative of the population. Any rare alleles present in that sequence can bias future reads and analysis, and it does not always allow researchers to have a holistic view of the variations in the whole population. For this reason, pangenome graphs have developed that allow one to represent many sequences as a whole, while still maintaining each sequence's importance. In this paper, I aim to discuss pangenome graphs, provide an overview of various methods of building these graphs, as well as present my example graphs generated using simplified versions of these graph building algorithms.

In order to discuss these pangenomic graphs, it is useful to understand their structure, how they can be generated from multiple sequences, and how to visualize them. As stated earlier, instead of using one representative genome and aligning everything else to it, these pangenome graphs use multiple sequences. The two pangenome graphs that I implemented are the variation graph and the k-block graph.

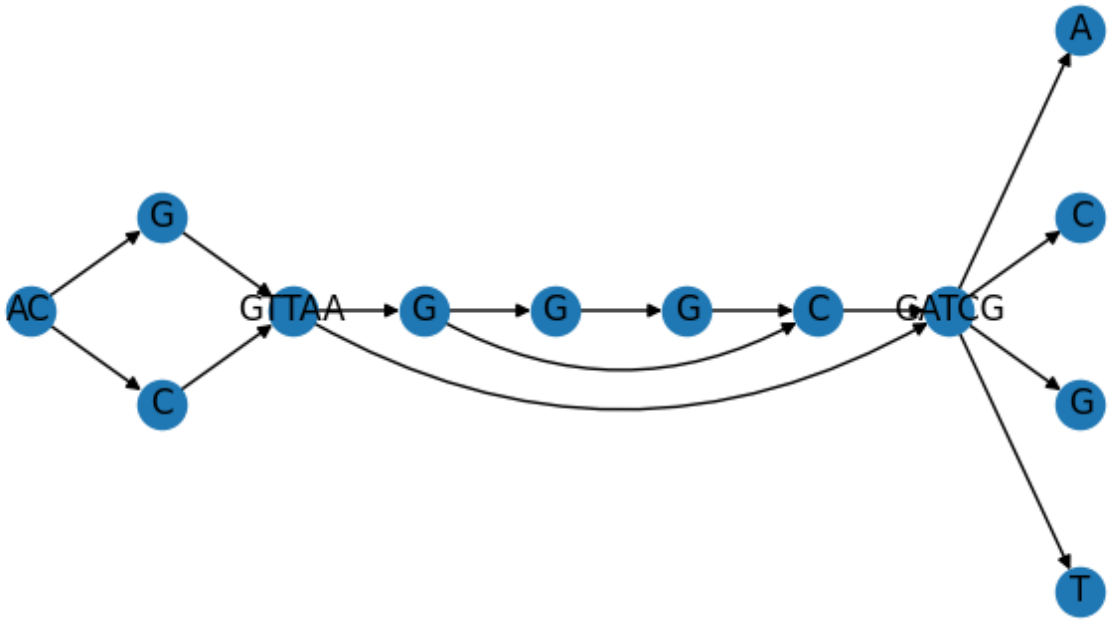

Figure 1: Variation Graph built from the following sequences:

ACGGTTAAGGGCGATCGA
ACGGTTAAG--CGATCGC
ACCGTTAA----GATCGG
ACCGTTAAGGGCGATCGT

The variation graph maps every difference in the sequences, with the graph converging at common subsequences. These common subsequences are valuable for analysis, as they can represent alleles that each genome in the population has in common, and the splits in the graph can show the various mutations and changes in each genome. Logically, the fewer branches in a variation graph, the more closely related the genomes are to one another. The graph also allows for jumps between distant nodes, which is quite useful for mapping reads to the graph. For my implementation of this graph, I iterate through each position *i* in the strings, and get the set of unique characters at *i* for all strings. From there, I compress these characters down, where any consecutive singleton sets combine into one single strings. For example, [‘A’] [‘C’] [‘G’, ‘C’] [‘G’] [‘T’] [‘T’] [‘A’] compresses down to [‘AC’] [‘G’, ‘C’] [‘GTTA’]. This makes sense because for any singleton at position *i*, all strings share that character. To account for skips, any

'-' is ignored when generating nodes. Later, edges are added by using a sliding window that keeps track of substring_A and substring_B, and then generates an edge between those nodes A and B. If substring_B is '-', then the window expands until it finds a non-'-' substring. Thus, an edge can be created that jumps one or many nodes, accounting for those '-' skip characters.

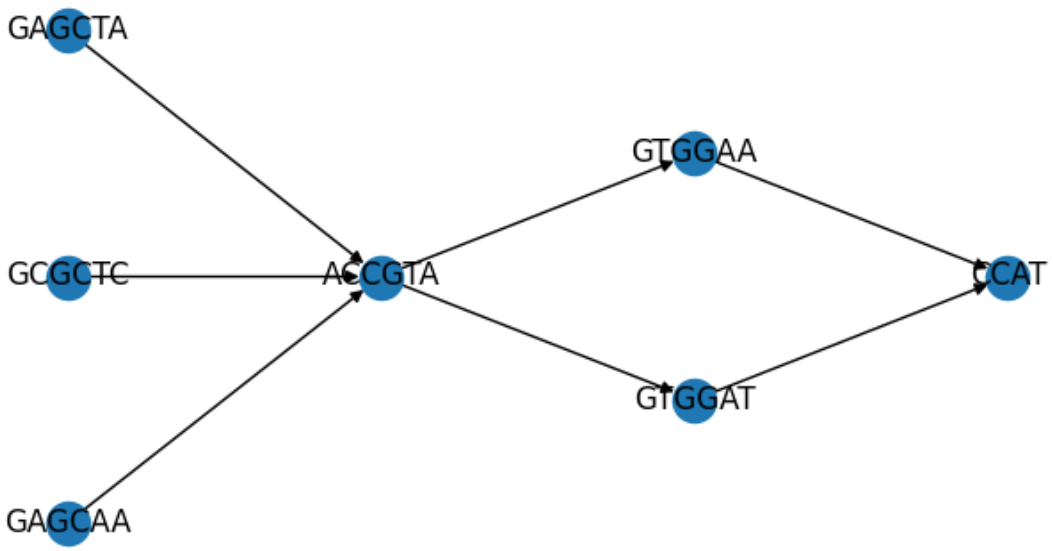

Figure 2: k-Block Graph built from the following sequences:

GAGCTAACCGTAGTGGAACCAT
GCGCTCACCGTAGTGGAACCAT
GAGCAAACCGTAGTGGATCCAT
GAGCTAACCGTAGTGGATCCAT
GCGCTCACCGTAGTGGATCCAT

The other graph, the k-block graph, instead splits each sequence into blocks of size k, and each unique subsequence in each block forms a node. The blocks being a set size can improve the processing time and can be adjusted depending on what sort of subsequences one is analyzing. My implementation of the k-Block graph was much simpler than the variation graph. It simply splits each sequence into k-blocks, generates nodes from the unique labels in each

block (a layer id is used to keep them lined up, as well as allow for nodes to have unique names even with the same labels), and finally adds the edges according to each sequence.

Figure 3: k-block graph on real world viral protein sequences

While these two implementations are helpful to view simple examples of pangenome graphs, they do not help much for real-world genome, where the genomes themselves may be massive. In Figure 3, I ran my k-block graph algorithm on a database of multiple virus protein sequences, which are all fairly sizeable. As one can see, the graph produced is incredibly dense and difficult to parse. However, there are a number of useful graph building algorithms in use today, and many more continue to be developed and improved. They all have different methods of building the graphs, and some are more tailored for specific analyses. Two early examples are Minigraph (Li et al., 2020) and Cactus (Armstrong et al., 2020). Minigraph focuses solely on structural variants, which are helpful for studying pathogen evolution. Cactus, on the other hand, uses a phylogenetic tree as a guide to build the multiple alignments, and then can add that to a

minigraph variant. While studies show that Cactus is very accurate, it does not scale well with the ever increasing size of studied genomes. Another option is PPanGGOLiN (Gautreau et al., 2020). PPanGGOLiN takes a different approach and partitions the genomes into families and neighborhoods. The families are used as the nodes, while neighborhoods are used for edges. This allows for a more compact graph that is a bit simpler to analyze and is effective for finding the persistent/shell genome. Finally, there is Pandora (Colquhoun et al., 2021), which takes individual MSAs from genetic coding sequences to build the graph. Pandora has proven effective for finding single-nucleotide polymorphisms (SNPs) and outperforms single-reference based algorithms.

Pangenome graphs are an emerging field of computational biology that continues to develop and improve to meet the ever rising need for a holistic multiple genome analysis. While there are some simple implementations of these graphs, for most research it is recommended to use the much more complex options recently developed. Many of these methods are still bogged down by slower runtimes, but the field is likely to only keep improving with time.

Sources:

Yang Z, Guarracino A, Biggs PJ, Black MA, Ismail N, Wold JR, Merriman TR, Prins P, Garrison E, de Ligt J. Pangenome graphs in infectious disease: a comprehensive genetic variation analysis of *Neisseria meningitidis* leveraging Oxford Nanopore long reads. Front Genet. 2023 Aug 10;14:1225248. doi: 10.3389/fgene.2023.1225248. PMID: 37636268; PMCID: PMC10448961.

Li H, Feng X, Chu C. The design and construction of reference pangenome graphs with minigraph. Genome Biol. 2020 Oct 16;21(1):265. doi: 10.1186/s13059-020-02168-z. PMID: 33066802; PMCID: PMC7568353.

Armstrong J, Hickey G, Diekhans M, Fiddes IT, Novak AM, Deran A, Fang Q, Xie D, Feng S, Stiller J, Genereux D, Johnson J, Marinescu VD, Alföldi J, Harris RS, Lindblad-Toh K, Haussler D, Karlsson E, Jarvis ED, Zhang G, Paten B. Progressive Cactus is a multiple-genome aligner for the thousand-genome era. Nature. 2020 Nov;587(7833):246-251. doi: 10.1038/s41586-020-2871-y. Epub 2020 Nov 11. PMID: 33177663; PMCID: PMC7673649.

Gautreau G, Bazin A, Gachet M, Planel R, Burlot L, Dubois M, Perrin A, Médigue C, Calteau A, Cruveiller S, Matias C, Ambroise C, Rocha EPC, Vallenet D. PPanGGOLiN: Depicting microbial diversity via a partitioned pangenome graph. PLoS Comput Biol. 2020 Mar 19;16(3):e1007732. doi: 10.1371/journal.pcbi.1007732. Erratum in: PLoS Comput Biol. 2021 Dec 10;17(12):e1009687. PMID: 32191703; PMCID: PMC7108747.

Colquhoun RM, Hall MB, Lima L, Roberts LW, Malone KM, Hunt M, Letcher B, Hawkey J, George S, Pankhurst L, Iqbal Z. Pandora: nucleotide-resolution bacterial pan-genomics with reference graphs. Genome Biol. 2021 Sep 14;22(1):267. doi: 10.1186/s13059-021-02473-1. PMID: 34521456; PMCID: PMC8442373.