Supplementary Material

All functional features included in GFAKaleidos

Table 1 List of all functional features included in GFAKaleidos

Catagory	Statistic	Description
GFA file	File size	The size of GFA files, measuring the compression efficiency of the graph.
	# of segments	The number of segments.
	# of links	The number of links.
	# of paths	The number of paths within graphs, representing the number of haplotypes.
	# of single direction segment	This means that segment A only exists as A- or A+.
	# of bidirectional direction segment	This means that segment A only exists as A- and A+.
	Growth	Estimates how the pangenome expands as additional genome assemblies are added [1].
Vertices	# of vertices	The number of vertices in the graph.
	Vertex size	The distribution of the number of bases in vertex labels across the graph.
	Total length	Total number of bases in vertex labels across the graph.
	N50	A larger N50 indicates that the main path in the graph is longer, suggesting a higher quality of pangenome construction.
	L50	The minimum number of vertices in the pangenome graph whose cumulative length accounts for at least half of the total pangenome size.
	Degree distribution	The degree distribution of vertices, with high-degree vertices often serving as mutation hotspots.
	Dead ends	The number of vertices that end without connecting to other vertices.
	Start ends	The number of vertices with in-degree is 0.
	Coverage	The number of bases/vertices/edges covered by different amounts of paths.
Edges	# of edges	The number of edges in the graph.
	# of loops	The number of loops in the graph, representing unmutated tandem repeats.
	Loop length	The Length distribution of loops in the graph.
	# of cycles	The number of cycles in the graph. Cycles represent tandem repeats but also increase the graph's path complexity.
	Minimum weight cycle	The minimum total length (sum of base counts on vertex labels) of cycles .
	Cycle distribution	The length distribution of cycles in the graph.
Subgraphs	Cuts	The number of cut points or bridges used to assess the connectivity.
	(Weak) Connected components	The number of connected components in the graph, representing co-occurring and co-localized gene families.
	Strongly connected components	The number of strongly connected components in the graph, representing genome regions of high complexity.
	Superbubbles	The number of superbubbles in the graph, representing polymorphisms [2].
	Simple bubbles	The number of simplebubbles in the graph.
	Nested bubbles	The distribution of bubble nesting depths in the graph [3].
	Bubble chains distribution	The length distribution of bubble chains.

Comparing different graph-building algorithms

The raw data for the two example GFA files provided on the webpage originates from the *Drosophila melanogaster* genome assembly on NCBI. The GitHub page offers a tutorial on constructing GFA files using various algorithms, along with download links.

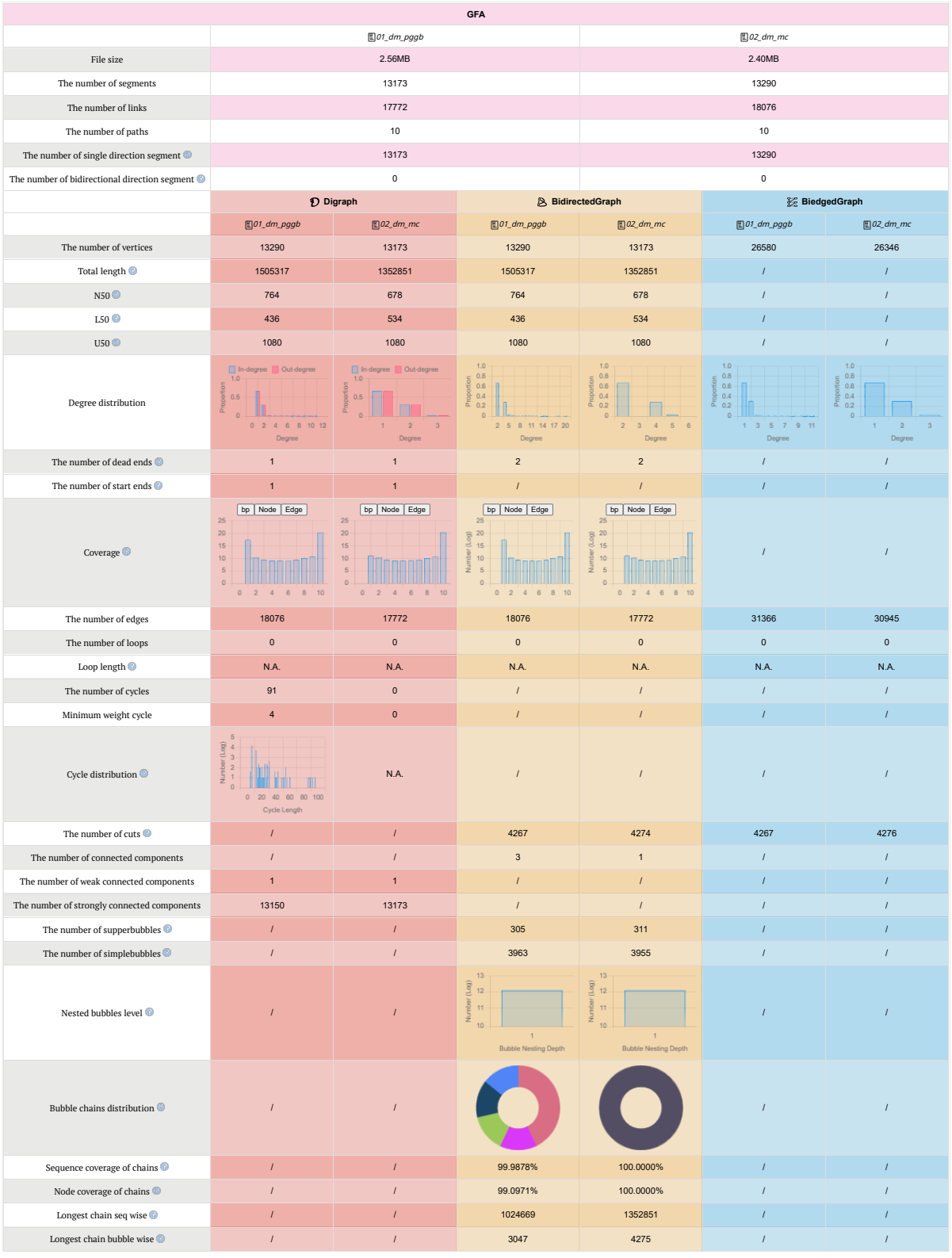

Fig. 1 Beyond numerical differences, the two example GFA files also vary in graphical statistics. Degree distribution, which reflects the complexity of variations at vertices, was normalized for clearer comparison. Results indicate that the MC-bidirected graph represents variation more succinctly, with a maximum vertex degree of 5, compared to 20 in the PGGB-bidirected graph. This is further supported by the bubble chain distribution: the MC-bidirected graph contains a single 1,352,851-base bubble chain, whereas PGGB fragments it into multiple slightly shorter chains.

Functional Capability Comparison of GFAKaleidos and State-of-the-Art Tools

Table 2 summarizes the functionalities of commonly used pangenome graph statistical tools. Compared to GFAKaleidos, these tools treat graph statistics as a secondary feature, with fewer selected metrics.

Table 2 Summary of functional capabilities in state-of-the-art tools

Name	Function	
Bandage [4]	Computing statistics	Node/Edge count
		Total length
		(Percentage) Dead ends
		Connected/Largest components
		N50
		Shortest/Longest/Median node
		Lower/Upper quartile node
	Rendering and interaction	
gfatools	Computing statistics	# of segments/links/arcs
		Max rank
		Total/Average segment length
		Sum of rank-0 segment lengths
		Max/Average degree
	Extracting subgraphs	gfatools view [options] <in.gfa>
	convert GFA to FASTA	gfatools gfa2fa [options] <in.gfa>
	print bubble-like regions	gfatools bubble [-g] <in.gfa> out.gfa
odgi [5]	Computing statistics	# of nucleotides/nodes/edges/paths/steps
		# of weak-connected-components/self-loops/nondeterministic edges
		# of A/C/G/T
		Sorting Goodness Eval Options
	Detecting Complex Regions	
	Extracting Selected Loci	
	Sorting and Layouting	
	Navigating and Annotating Graphs	
	Removing Artifacts and Complexing Regions	
PANCAT	Computing statistics	# of segments/edges
		# and percentage of 0-50/50-51/51-inf bp
		Size and percentage of 0-50/50-51/51-inf bp
		Total size of segments
	Comparing/Visualizing/Compressing graphs	
	Isolating subgraphs	
	Adding path offsets	
	Asserting completeness	
	Comparing graph alignments	
	Reconstructing sequences	
	Unfolding cycles	
Panacus [6]	Computing statistics	Coverage histogram
		Growth table

Backend Algorithm for Computing Statistical Metrics

- Graph models

Although bidirected graphs and biedged graphs differ in definition, they can be implemented using a single class, *BiedgedGraph*, to optimize storage. When computing statistics for bidirected graphs, vertices in *BiedgedGraph* should be treated as edges, and edges as vertices.

- Connected components

The Tarjan algorithm [7] and Gabow algorithm [8] are used to enumerate connected components in graphs, as [9] demonstrates their superior speedup over Kosaraju algorithm on most graphs, despite all three having linear complexity. However, due to the graph's large size, the traditional recursive implementation may cause stack overflow errors. To address this, the code replaces recursion with an iterative approach.

- Cycles

Cycle enumeration is the most time-consuming step in the entire pipeline. The Szwarcfiter and Lauer algorithm [10] achieves a time complexity of $O(|V| + |E||C|)$, making it the most efficient algorithm currently available. Here, $|V|$, $|E|$, and $|C|$ denote the number of vertices, edges, and cycles in the graph respectively.

The Szwarcfiter and Lauer algorithm is used to identify cycles within the strongly connected components (SCCs) of the graph. Since SCCs are disjoint, parallel processing is an effective optimization strategy.

- Bubbles

Similar to BubbleGun [2], we present a fast and efficient method for detecting bubbles and superbubbles across various graph models. The algorithm maintains a dynamic set S for each candidate source vertex s, tracking the currently accessible vertices. During traversal, if a terminal vertex (tip) with no children or a cycle pointing back to s is encountered, path exploration is terminated early to prevent redundant computations. When the set S reduces to a single vertex t with no other visited vertices (i.e., vertices with at least one visited parent), t is identified as the sink vertex. If the subgraph between s and t satisfies the conditions of directed acyclicity and vertex reachability, it is classified as a bubble.

By leveraging the dynamic set S and an early termination strategy, the algorithm achieves an average complexity of $O(|V| + |E|)$, making it well-suited for large-scale graphs. GFAKaleidos demonstrates significant performance improvements over BubbleGun, processing large graphs in a fraction of the time. For instance, on a human pangenome graph comprising 17,565,936 segments and 24,328,848 links, GFAKaleidos completes execution of all statistics in just 352 seconds, compared to BubbleGun's runtime of approximately 2.5 hours.

Reference

1. Liao W W, Asri M, Ebler J, Doerr D, et al. A draft human pangenome reference. Nature, 2023, 617(7960): 312–324

2. Dabbaghie F, Ebler J, Marschall T. Bubblegun: enumerating bubbles and superbubbles in genome graphs. Bioinformatics, 2022, 38(17): 4217–4219
3. Eizenga J M, Novak A M, Sibbesen J A, Heumos S, Ghaffaari A, Hickey G, Chang X, Seaman J D, Rounthwaite R, Ebler J, Rautiainen M, Garg S, Paten B, Marschall T, Sirén J, Garrison E. Pangenome graphs. Annual review of genomics and human genetics, 2020, 21(1): 139–162
4. Wick R R, Schultz M B, Zobel J, Holt K E. Bandage: interactive visualization of de novo genome assemblies. Bioinformatics, 2015, 31(20): 3350–3352
5. Guarracino A, Heumos S, Nahnsen S, Prins P, Garrison E. Odgi: understanding pangenome graphs. Bioinformatics, 2022, 38(13): 3319–3326
6. Parmigiani L, Garrison E, Stoye J, Marschall T, Doerr D. Panacus: fast and exact pangenome growth and core size estimation. Bioinformatics, 2024, 40(12): btae720
7. Tarjan R. Depth-first search and linear graph algorithms. SIAM journal on computing, 1972, 1(2): 146–160
8. Gabow H N. Path-based depth-rst search for strong and biconnected components. Information Processing Letters, 2000, 74: 107–114
9. Hsu D F, Lan X, Miller G, Baird D. A comparative study of algorithm for computing strongly connected components. In: Proceedings of the 2017 IEEE 15th International Conference on Dependable, Autonomic and Secure Computing. 2017, 431–437
10. Szwarcfiter J L, Lauer P E. A search strategy for the elementary cycles of a directed graph. BIT Numerical Mathematics, 1976, 16(2): 192–204