Seeing Through Hairballs: Simplifying Pangenome Graphs

Martin Senderák, Tomáš Vinař

Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava, Mlynská dolina, 842 48 Bratislava, Slovakia

Abstract

When building a pangenome graph of tens of individual bacterial genomes, one often encounters a problem that the structure of the resulting graph is generally unwieldy. In particular, some vertices are likely to connect sequences that appear in different contexts in different genomes. When visualizing such pangenomes, these vertices pull together different unrelated parts of individual genomes, creating a difficult to analyze hairball. In this paper, we explore this issue and propose a greedy heuristic algorithm to simplify pangenome graphs by eliminating certain vertices and edges. This means that certain portions of the original genomes will no longer be represented in the pangenome, and therefore it is important to minimize such graph modifications. We demonstrate effectiveness of our approach on a set of 50 *Escherichia coli* isolates and we show that visualization of such a simplified pangenome graph can unmask interesting features of the pangenome organization.

Our software and data are available at https://github.com/SendyM/Pangenomic_Graph_Filtering_and_Visualization

Keywords

pangenome graphs, visualization, heuristics

1. Introduction

Representing multiple individual genomes of the same species as a pangenome graph has become increasingly popular [1, 2]. Pangenome graphs are not only a compact data structure that can efficiently represent genetic variation in tens or even hundreds of individuals, but they can also serve as an important visualization tool for studying structural properties of the pangenome as a whole.

In this paper, we consider *block-based pangenome graphs* that are built by tools such as minigraph [3], PanGraph [4], or by our own pipeline based on GEESE software [5]. Similar sequences (both from within and between individuals) are clustered together and form multiple alignment blocks that will serve as vertices of the graph. Each individual can now be represented as a walk through these vertices and the edges of the graph are simply a union of edges from these walks.

Block-based pangenome graphs do not emphasize local differences between individuals, such as single nucleotide polymorphisms or short indels, as this information is hidden within multiple alignments corresponding to individual vertices. Instead they highlight larger-scale differences between individuals related to rearrangements, segmental duplications, or structural variation.

However, even when using only tens of individual bacterial genomes, the structure of the resulting graph is generally unwieldy. First, some vertices represent segments (such as transposable elements, phages and other mobile elements) that can occur in multiple locations within the same genome or appear in different sequence contexts in different individuals (Figure 1A). Consequently, when visualizing the pangenome, these vertices bring their otherwise unrelated neighbors to close proximity. Second, even though homologous sequences between individuals should be represented by the same vertex, sometimes they can be split into two or more vertices, generally fragmenting walks and creating a visual clutter (Figure 1B). In extreme cases, complex bubbles are formed, potentially compounding both of the previous issues (Figure 1C). Consequently, in visualization, many unrelated parts of the pangenome graph are typically brought together into a small space, creating an unwieldy *hairball* that effectively hides the underlying structure of the pangenome, as shown in Figure 2A.

A common practice is to anticipate these problems and attempt to identify the segments of sequences that may give rise to problematic vertices, and excising them from the genomes before building the pangenome. However, this step requires advance biological knowledge which is often not available or

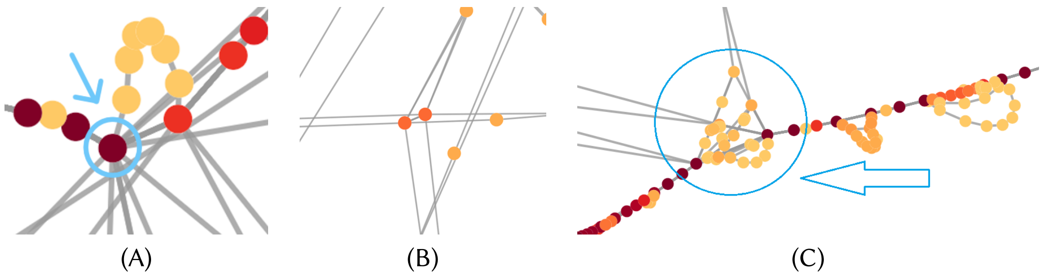


Figure 1: Common issues observed in pangenome graph visualizations. (A) Vertices occuring in many contexts. (B) Fragmented vertices (or small bubbles). (C) Dense vertex communities (or complex bubbles).

incomplete. Instead, here we propose a heuristic to analyze the structure of the pangenome, identifying vertices that are responsible for the hairball formation. These vertices can be removed from the graph, reconnecting the corresponding walks by bypassing the vertex, effectively simulating excision of the corresponding sequence from the original genomes. Alternately, in case of complex bubbles, the bubble internal structure can be simplified by removing rarely used edges.

We demonstrate the technique on a pangenome of 50 *Escherichia coli* isolates. We also compare our results to PanGraph's graph simplification routine and show that we can get superior results which moreover represent approximately 6× more of the length of the original input sequences than the PanGraph results.

2. Results

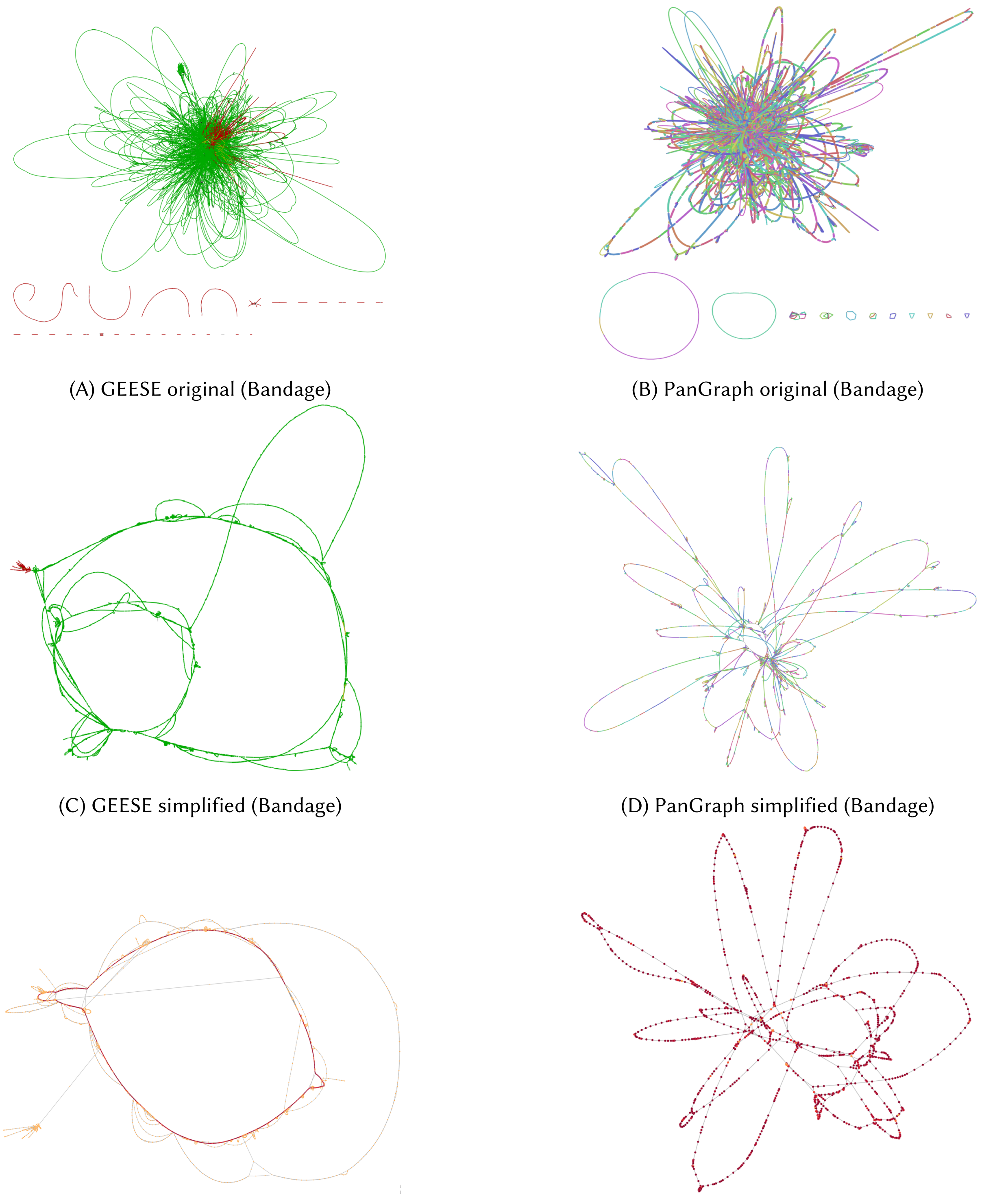
We have decided to explore the possibilities for graph simplification on a pangenome constructed from 50 *E. coli* isolates associated with wastewater and livestock [6]. We assembled the genomes from a combination of short and long-read data (see Methods for details) and used our own pipeline based on GEESE software [5] to assemble a pangenome graph. The graph building procedure collapses similar sequences at 95% identity to a single vertex (see Methods) which allows us to identify shared segments between individuals, but at the same time also captures homologies within individual genomes.

The resulting graph in Figure 2A visualized with Bandage [7] has 4262 vertices and is clearly a hairball. Note that input data contain *E. coli* chromosomes (shown in green) as well as plasmids (in red). Many plasmid sequences are clearly separated from the chromosomes, but some of the plasmid sequences share some vertices (such as insertion elements) with chromosomes.

For comparison, we have also constructed a similar pangenome graph with PanGraph [4], and it is shown in Figure 2B. The PanGraph pangenome has a similar number of vertices (4337), but it is slightly more collapsed compared to the GEESE graph, as the total length of the representative sequences of PanGraph vertices is approx. 60% of the representative sequences in the GEESE graph. Regardless, the PanGraph pangenome is also a hairball.

From the same data, we have also attempted to construct the pangenome with minigraph [3]. However, minigraph algorithm starts from a single reference and then iteratively augments the resulting graph with new sequences. In this process, it ignores large rearrangements (spanning regions over 100 Kbp in length) and also does not consider any within-genome similarities. Consequently, the resulting graph has a simpler structure, mostly consisting of a linear backbone with bubbles representing mid-size structural variation and does not represent the full complexity of the pangenome.

We have developed a greedy heuristic which identifies and removes some of the vertices and edges in the pangenome graph to address problems that were outlined in Figure 1 (see Methods). We have applied this heuristic to the pangenome graph produced by GEESE and obtained the structure shown in Figure 2C. The resulting simplified graph contains approx. 75% of the original vertices covering about 90% of the representative sequences from the original graph (Table 1). The graph shows a clear circular structure (corresponding to the fact that *E. coli* has a single circular chromosome) and allows for quick visual identification of common rearrangements and other structural variations. The hairball structure



(A) GEESE original (Bandage)

(B) PanGraph original (Bandage)

(C) GEESE simplified (Bandage)

(D) PanGraph simplified (Bandage)

(E) GEESE simplified (D3.js)

(F) PanGraph simplified (D3.js)

Figure 2: Comparison of original and simplified pangenomic graphs built from 50 *E. coli* genome assemblies. (A), (C), (E): Graphs constructed with GEESE. (B), (D), (F): Graphs constructed with PanGraph.

no longer obscures major pangenome features.

Bandage visualizes vertices of the pangenome as curved lines or boxes whose length is proportional to the length of the sequences which are connected by very short edges. This approach has two limitations. First, the visualization is mostly dominated by long sequences, hiding potentially interesting structural variation that involves small and medium-sized segments. Second, the short edges between vertices may sometimes not allow for enough freedom to construct a proper layout of some parts of the graph.

Table 1
Comparison of GEESE and PanGraph pangenome graphs before and after simplification. HI is the hairball index describing the complexity of the graph (see Methods).

graph	vertices		edges		total length		HI
GEESE original	4262		6979		24.2 Mbp		4.8
GEESE simplified	3197	(75%)	4143	(59%)	21.5 Mbp	(89%)	2.9
PanGraph original	4337		7195		14.3 Mbp		5.1
PanGraph simplified	1233	(28%)	1887	(26%)	3.6 Mbp	(25%)	4.7

Therefore we have also explored alternative visualization where vertices are represented as simple nodes connected by variable length edges. The layout of the graphs was produced by a force-directed approach implemented in D3.js library (see Methods). Figure 2E shows the same simplified GEESE pangenome graph. We used the color of the vertices and edges to represent the number of individual genomes in which each vertex or edge is included (darker color means more individuals). The visualization clearly shows a circular structure of the core *E. coli* pangenome in dark, with auxiliary parts of the pangenome shown in lighter colors as bubbles or other connections that may represent large but relatively rare rearrangements.

We have also used PanGraph "simplify" function on the PanGraph generated pangenome (Figures 2D,F). This resulted in a great reduction of visual complexity, however only 25% of the original sequences were represented in this simplified graph (less than a single genome). Moreover, the graph still resembles a hairball structure, without clear notion of a single circular chrosome which would be expected in a graph illustrating *E. coli* genome organization.

3. Methods

3.1. Building a pangenome graph with GEESE

Both short and long sequencing reads of 50 *E. coli* isolates [6] were downloaded from NCBI and assembled into near-complete genomes with unicycler [8]. For visualization purposes, all contigs longer than 1 Mbp were labeled as "chromosome" and shorter circular contigs were labeled as "plasmids". The remaining contigs were labeled as "plasmids" if they mapped to the PLSDB database [9] on at least 80% of their length and to the reference *E. coli* chromosome on less than 20% of their length.

The resulting genomes were used to build a pangenome graph as follows. First, last software [10] was used to create all-to-all genome alignments; only alignments of length at least 100 with identity at least 95% were kept. These alignments were used to decompose the original contigs into non-overlapping atomic segments by GEESE [5] so that no alignment boundary (a breakpoint) lies within the atoms. For practical reasons, we only consider atoms of length 1000 or more. Due to this requirement, and also because it is often impossible to determine precise alignment boundaries, certain parts of the original sequence are not covered by atomic segments and are excluded from further analysis [11].

GEESE splits atoms into equivalence classes based on sequence similarity so that if two atoms have sequence identity above 95%, they are in the same class. Each class forms a single vertex of the pangenome graph and is assigned one of the member sequences as a representative. Each input contig contains several non-overlapping atoms and thus can be seen as a sequence of class identifiers. Two nodes of the graph (atom classes) are connected by an edge if they are consecutive in at least one of the contigs.

This procedure resulted in a pangenome with 4262 vertices and 6979 sequences, with vertices representing 24.2 Mbp of genome sequences in total. For comparison, the length of a single *E. coli* genome in our set ranges between 4.5 and 5.3 Mbp, and the total length of the whole data set is 253.7 Mbp.

3.2. Simplifying the pangenome graph

The pangenome graph is represented by its vertices and a set of walks through these vertices corresponding to input contigs. The edges of the graph are simply a union of edges of all walks. Each vertex v is also characterized by the length of its representative sequence ℓ_v. The depth d_v of a vertex v is the number of walks that include the vertex. The context of a vertex on a walk is a triple that includes the vertex, its predecessor, and its successor on that walk. Thus each vertex can occur in multiple contexts. We define the context-width w_v of vertex v as the number of unique contexts in which it occurs on all walks. Note that typically $d_v > w_v$, since the vertex likely occurs in the context of the same predecessor and successor on multiple walks. Finally, we call a vertex duplicated, if it occurs in some walk at least twice.

The basic simplifying operation is *remove-and-reconnect*, where some vertex v is removed from the graph and from all of the walks. To keep each walk contiguous, the predecessor of vertex v on the walk is reconnected with the successor of vertex v on the walk.

The algorithm for simplifying the pangenome graph is a heuristic consisting of two steps.

1. **Basic iterative filtering.** The goal of the basic iterative filtering is to reduce the number of vertices that clearly contribute to hairball effects in the graph. We remove-and-reconnect vertices that have length $\ell_v < L$, depth $d_v < D$, degree $w_v > W$, and we also remove-and-reconnect duplicated vertices. After this step, we recompute the depth and degree of the remaining vertices and iterate at most T times. Here, L (minimum length), D (minimum depth), W (maximum context-width), and T (the number of iterations) are user-specified parameters.
2. **Complex bubble filtering.** We search for complex compact bubbles by considering pairs of vertices (v_s, v_t) which are on some walk at most X vertices apart. If the out-degree of vertex v_s and in-degree of vertex v_t are both greater than Δ, we consider the pair of vertices (v_s, v_t) a boundary of a complex compact bubble. Values X and Δ are user-specified parameters.
   Our goal is to simplify complex compact bubbles by removing less frequent contexts of vertices belonging to the bubble. In particular, for a pair of vertices (v_s, v_t) that form a boundary of a complex compact bubble, we collect all subwalks where v_s and v_t are at a distance less than X; vertices on these subwalks will be part of the bubble. Vertex v can occur on these subwalks in multiple contexts. For each vertex v that is part of the bubble, we keep only its most frequent context; for all walks where the vertex v occurs in other contexts, we remove it from the walk and reconnect its predecessor and successor.

3.3. Hairball index

The graph filtering heuristic introduced in the previous section contains a number of user-specified parameters that control its behaviour and affect the result. To avoid manual fine-tuning and visual comparisons of many graphs, we have created an indicator called *hairball index* (HI) that measures a structural complexity of a filtered graph with respect to the original graph. In our experiments, the HI was used to explore the grid of the parameters to find combinations that produce graphs that can be visualized well.

The HI is a linear combination of multiple components:

$$\begin{aligned}\mathrm{HI} = \quad & 0.5E/V + 0.25(1-Q) + 0.1EC/EC_0 + \\ & 0.1(V_0 - V)/V_0 + 0.1(E_0 - E)/E_0 + \\ & 0.05(L_0 - L)/L_0\end{aligned}$$

Here, E is the number of edges, V is the number of vertices, EC is the average length of the shortest path between pairs of vertices, and L is the total length of all sequences represented in the graph. The variables with subscript 0 represent the same indicators in the original graph. Indicator Q is a modularity coefficient with values between 0 and 1, measuring whether the graph can be split into densely connected communities that are themselves only loosely connected [12]. Low values indicate

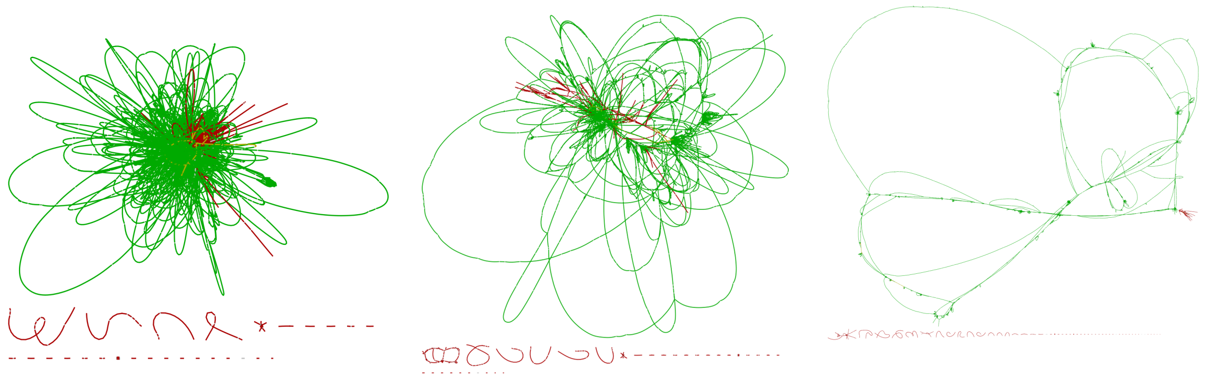

(A) original graph, HI=4.78 (B) HI=3.28 (C) HI=2.95

Figure 3: Connection between hairball index (HI) and visual complexity of a pangenome graph. Graphs (B) and (C) are different simplifications of graph (A).

dense graphs, while higher values mean graphs that are more modular. We computed the modularity coefficient using implementation in the NetworkX library with greedy split to communities (function greedy_modularity_communities).

The first three terms measure the properties that lead to graphs that can be visualized well. High values indicate graphs that likely contain a global hairball. The remaining terms indicate how much of the information was removed from the graph and introduce balance between visual complexity of the graph and the amount of information that it represents. Figure 3 demonstrates the connection between values of HI and the visual complexity of graphs.

3.4. Visualization of pangenome graphs

The standard accepted tool for visualization of pangenome graphs is Bandage [7]. Bandage represents vertices of the graph as large curved boxes or thick lines whose lengths are proportional to the length of the sequence represented. The layout of the graph is created by a fast force-directed approach as implemented in the OGDF library [13, 14].

While the Bandage approach to displaying vertices leads to graphs where the amount of dedicated space is proportional to the length of the sequence it represents, it has a disadvantage that it visually diminishes importance of locally highly connected regions and structural variation. The drawing is typically dominated by long stretches of sequence where no interesting rearrangements happen, while regions such as recombination hotspots and other densely connected regions are collapsed into large hairballs that are difficult to see and analyze.

Thus we also experimented with other representations that use more traditional graph visualization approach, where graph vertices are represented as small nodes and are connected by (potentially long) edges. We used a force-directed layout implemented in D3.js [15]. The library allows the user to tune parameters interactively, possibly helping to reveal certain features of the graph, such as circular structure of the core genome as shown in Figure 2E.

3.5. Building and simplifying pangenome graphs with PanGraph

PanGraph [4] pangenome graphs were created from the same data set as in Section 3.1, by following the steps recommended in PanGraph documentation, keeping default parameters. PanGraph approach to graph construction is different from GEESE. While GESSE optimizes segmentation of sequences into atoms based on a global requirement of producing a set that is invariant to transitive closure, PanGraph employs a progressive alignment strategy with splitting nodes where possible rearrangements are detected. PanGraph also includes `simplify` command which we used to simplify its output graph as displayed in Figure 2D,F. The output GFA files were visualized both by Bandage and by D3.js.

4. Conclusion

This paper presents a work in progress on visualization of pangenome graphs. A typical pangenome contains vertices that represent sequences that occur in different individual genomes in different contexts. This leads to graph visualizations that contain dense hairballs and beacause of that, it is difficult to perceive any other high-level features of the pangenome organization. Our proposed method removes some of the vertices and the edges from the pangenome graph. On a set of 50 *Escherichia coli* isolates, we have demonstrated that our technique leads to a visualization showing distinct features of the pangenome organization, while preserving approx. 90% of the represented sequences.

Our work opens several interesting avenues to further research. First of all, we have only demonstrated our approach an a single pangenome graph constructed from 50 individual bacterial genomes. Natural extension would be to look at pangenomes of a variety of bacterial species, pangenomes constructed from hundreds or thousands of isolates, and at pangenomes of more complex organisms, such as humans or plants. When applying our approach to larger genomes, scalability issues need to be explored and resolved.

Our proposed method is a simple heuristic. It would be interesting to formulate the problem of simplifying pangenome graphs as a well-stated optimization problem and pursue exact algorithms for finding optimal pangenome simplifications. In this paper, we have proposed two operations: removal of a vertex with path reconnection, or bypassing a vertex on certain paths. Another interesting operation is to split a vertex into multiple vertices, each being used in different contexts. In this way, only information about homology of certain segments would be removed, but the sequences themselves would still be represented.

We have demonstrated on examples that hairball index introduced in this paper characterizes the visual complexity of the pangenome graph. It may be of interest to determine whether any specific threshold values imply consistently improved visualization or whether the index can be used systematically to determine threshold values in heuristic steps of pangenome simplification algorithms.

Finally, we did not study the sequences that were removed from the pangenome. It is clear that transposable elements, phages, and other mobile elements will be represented, however more detailed analysis may point out other interesting functional classes, which may differ between different organisms.

Acknowledgments

Our research was supported by grants from the Slovak Research and Development Agency (APVV-22-0144), the Scientific Grant Agency (VEGA 1/0538/22), and European Union’s Horizon 2020 Research and Innovation Staff Exchange programme under the Marie Skłodowska-Curie grant agreement No. 872539 (PANGAIA). This research was enabled in part by support provided by the Digital Research Alliance of Canada.

Declaration on Generative AI

Authors did not use generative AI to write or correct the text of this paper.

References

[1] B. Paten, A. M. Novak, J. M. Eizenga, E. Garrison, Genome graphs and the evolution of genome inference, Genome Res 27 (2017) 665–676.
[2] W.-W. Liao, et al., A draft human pangenome reference, Nature 617 (2023) 312–324.
[3] H. Li, X. Feng, C. Chu, The design and construction of reference pangenome graphs with minigraph, Genome Biol 21 (2020) 265.

[4] N. Noll, M. Molari, L. P. Shaw, R. A. Neher, PanGraph: scalable bacterial pan-genome graph construction, Microb Genom 9 (2023).

[5] D. P. Rubert, F. V. Martinez, J. Stoye, D. Doerr, Analysis of local genome rearrangement improves resolution of ancestral genomic maps in plants, BMC Genomics 21 (2020) 273.

[6] L. P. Shaw, et al., Niche and local geography shape the pangenome of wastewater- and livestock-associated Enterobacteriaceae, Sci Adv 7 (2021) eabe3868.

[7] R. R. Wick, M. B. Schultz, J. Zobel, K. E. Holt, Bandage: interactive visualization of de novo genome assemblies, Bioinformatics 31 (2015) 3350–3352.

[8] R. R. Wick, L. M. Judd, C. L. Gorrie, K. E. Holt, Unicycler: Resolving bacterial genome assemblies from short and long sequencing reads, PLoS Comput Biol 13 (2017) e1005595.

[9] G. P. Schmartz, A. Hartung, P. Hirsch, F. Kern, T. Fehlmann, R. Muller, A. Keller, PLSDB: advancing a comprehensive database of bacterial plasmids, Nucleic Acids Res 50 (2022) D273–D278.

[10] S. M. Kielbasa, R. Wan, K. Sato, P. Horton, M. C. Frith, Adaptive seeds tame genomic sequence comparison, Genome Res 21 (2011) 487–493.

[11] M. Višňovská, T. Vinař, B. Brejová, DNA Sequence Segmentation Based on Local Similarity, in: T. Vinař (Ed.), Information Technologies - Applications and Theory (ITAT), volume 1003 of *CEUR-WS*, 2013, pp. 36–43.

[12] A. Clauset, M. E. Newman, C. Moore, Finding community structure in very large networks, Phys Rev E 70 (2004) 066111.

[13] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, P. Mutzel, et al., The Open Graph Drawing Framework (OGDF), in: R. Tamassia (Ed.), Handbook of graph drawing and visualization, 2014, pp. 543–569.

[14] S. Hachul, M. Jünger, Large-graph layout algorithms at work: An experimental study, Journal of Graph Algorithms and Applications 11 (2007) 345–369.

[15] M. Bostock, V. Ogievetsky, J. Heer, D^3 data-driven documents, IEEE Transactions on Visualization and Computer Graphics 17 (2011) 2301–2309.