Reconciling and Comparing Variation Graphs Using Homology Relations

Anna Lisiecka
Institute of Informatics, University of Warsaw, Poland

Adam Cicherski
Institute of Informatics, University of Warsaw, Poland

Norbert Dojer
Institute of Informatics, University of Warsaw, Poland

Abstract

In variation graphs genomic sequences are represented as paths sharing their nodes in homologous regions. Shared nodes must represent identical sequence fragments, but the definition does not impose strict criteria on when the paths should be merged and when not. Consequently, most variation graph building tools heuristically infer the graph structure from pairwise genome alignments or the results of seeding procedures of alignment algorithms.

In the current paper we introduce the concept of homology relation induced by a variation graph on the nucleotides of represented genomic sequences. This notion can be used to specify in a mathematically rigorous way the criteria the structure of a variation graph should meet. We investigate the relationships between the structures of variation graphs and their homology relations, as well as between operations on relations and on graphs. In particular, we show how to use the result of combining relations induced by different graphs to combine these graphs. Then, we propose homology-based methods of reconciling and comparing variation graphs representing the same genome collection. Moreover, we provide an implementation of homology-based tools for variation graph analysis.

2012 ACM Subject Classification Applied computing → Computational genomics

Keywords and phrases Pangenome, Variation Graph, Genome Homology

Digital Object Identifier 10.4230/LIPIcs.WABI.2026.29

Supplementary Material *Software*: `https://github.com/anialisiecka/vgrecon/` archived at `swh:1:dir:5c6c9a6e0bf9b74c40eb93f2b32b8beb452adfc6`

Funding This work was supported by the National Science Centre, Poland, under grant number 2022/47/B/ST6/03154.

1 Introduction

Variation graphs combine reference genomes and their variants into coherent data structures, representing genetic variation within a population. They intuitively represent both single nucleotide polymorphisms and large-scale structural variants (e.g. insertions, deletions and translocations). Efficient indexing methods [11, 13] make variation graphs an alternative to linear reference genomes that can reduce reference bias in mapping sequencing reads [7, 12].

Each variation graph node is labeled with a fragment of one or more individual genomes. Whole genomic sequences are represented as graph paths, and the concatenated labels of successive path nodes form the sequences. The framework is highly flexible in the sense that it does not impose constraints on when identical fragments of genomic sequences should be represented by a common node. Mathematically rigorous criteria the structure of a graph should meet were proposed in [2] and implemented in the variation graph building tool AlfaPang [3]. However, the authors note that AlfaPang may produce graphs with complex

local structures, and thus propose pipeline AlfaPang+, consisting of AlfaPang followed by the graph refinement tools smoothxg and gfafix [6]. Other variation graph building tools heuristically infer the graph structure from pairwise genome alignments or the results of seeding procedures of alignment algorithms.

It has been previously established [1] that variation graph construction tools exhibit structural instability, where identical input haplotypes may yield different graph representations across multiple runs. This instability may stem from the inherent non-determinism of the construction algorithms or the permutation of input sequences. For example, Minigraph-Cactus (MC) [9] requires the user to choose a reference genome that serves as the backbone of constructed graph. Other genomes are aligned to this backbone, and thus the similarity between non-reference sequence fragments may be overlooked if not accompanied by the similarity to the reference. As it was shown in [5], the choice of the reference genome has a significant impact on the topology of the resulting graph. PGGB [6] avoids the reference bias by inferring the graph structure from all-to-all pairwise genome alignments. However, its postprocessing procedures (used also in AlfaPang+) are non-deterministic, because the result depends on the order in which parallel computation threads complete their execution. Ensuring stability in graph construction is therefore essential to guarantee the reproducibility of various results and downstream analyses.

This raises the need to develop methods and tools for comparing and evaluating different graphs representing the same set of genomes. Usually such comparison is based on simple graph characteristics (size, node coverage by genomic paths, lengths of node labels, etc.), ignoring the way genomic sequences are represented within the graph. A better approximation of the similarity of the represented homology is given by the *segmentation distance* [5], derived from the sequence breakpoints induced by the edges of genomic paths. However, the segmentations are evaluated separately for each sequence, so this metric also indirectly compares the graph topology-encoded relationships between sequences.

In this article, we introduce the notion of a *homology relation*, which describes in a mathematically rigorous way how a variation graph models the sequence homology at the level of single nucleotides. The relation is representation-dependent, and the degree to which it reflects evolutionary homology in the classical sense depends on the particular graph construction method. The idea of introducing the concept of homology relations is inspired by the way of defining requirements for variation graphs in [2, 3], but its potential applications go beyond the formulation of graph properties. Here we investigate the relationships between the structures of variation graphs and their homology relations, as well as between operations on relations and on graphs. In particular, we show how to use the result of intersecting relations induced by different graphs to combine them into a single graph. Then, we propose homology-based methods of comparing variation graphs representing the same genome collection, specifically for evaluating a given graph against a gold standard, and for assessing the stability of graph construction algorithms. Moreover, we provide an implementation of homology-based tools for variation graph analysis, available at `https://github.com/anialisiecka/vgrecon/`.

2 Methods

2.1 Preliminaries

In bidirected graphs, every node has two sides (denoted here ± 1), and undirected edges connect adjacent nodes on specific sides. More formally, the edge connecting side o_1 of node v_1 with side o_2 of node v_2 is an unordered pair $\{\langle v_1, o_1\rangle, \langle v_2, o_2\rangle\}$. A node v can be

oriented in two possible ways, denoted $\langle v, +1\rangle$ and $\langle v, -1\rangle$. A *path* in a bidirected graph is a sequence of oriented nodes $\langle\langle v_1, o_1\rangle, \ldots, \langle v_m, o_m\rangle\rangle$ such that for all respective j, side o_{j-1} of node v_{j-1} is connected by an edge to side $-o_j$ of node v_j (so a path enters and exits the node on opposite sides).

In variation graph the nodes are labeled with DNA sequences. The two possible node orientations correspond to the two strands of DNA molecules. The labeling function l extends to oriented nodes: $l(\langle v, +1\rangle) = l(v)$ and $l(\langle v, -1\rangle)$ is the reverse complement of $l(v)$, and further to paths: $l(\langle\langle v_1, o_1\rangle, \ldots, \langle v_m, o_m\rangle\rangle) = l(\langle v_1, o_1\rangle) \ldots l(\langle v_m, o_m\rangle)$.

Consider a set of homologous sequences $S = \{s_1, \ldots, s_n\}$. A *variation graph representing* S is defined as $G = \langle V, E, l, \pi\rangle$, where $\langle V, E, l\rangle$ is a labeled bidirected graph and π is a mapping from S to the set of G-paths satisfying the following conditions:

- $l(\pi(s_i)) = s_i$ for $i = 1, \ldots, n$,
- all the paths $\pi(s_i)$ jointly cover the whole graph G.

The paths $\pi(s_i)$ are called *genomic paths*.

Two representations $G = \langle V, E, l, \pi\rangle$ and $G' = \langle V', E', l', \pi'\rangle$ of S are *isomorphic* if there exist a bijection b between the sets of oriented nodes in G and G' satisfying the following conditions:

1. $b(\langle v, o\rangle) = \langle v', o'\rangle \iff b(\langle v, -o\rangle) = \langle v', -o'\rangle$,
2. $l(\langle v, o\rangle) = l(b(\langle v, o\rangle))$,
3. $\{\langle v_1, o_1\rangle, \langle v_2, o_2\rangle\} \in E \iff \{b(\langle v_1, o_1\rangle), b(\langle v_2, o_2\rangle)\} \in E'$,
4. $\pi(s_i) = \langle\langle v_1, o_1\rangle, \ldots, \langle v_m, o_m\rangle\rangle \iff \pi'(s_i) = \langle b(\langle v_1, o_1\rangle), \ldots, b(\langle v_m, o_m\rangle)\rangle$.

2.2 Homology relations

Below, we introduce the notion of the homology relation associated with a representation G of S. Intuitively, this relation identifies the nucleotides in S-sequences that are merged along the paths representing them in the graph. The definition accounts for the fact that variation graphs can represent homology between both parallel and antiparallel DNA strands.

The set of *positions* in $S = \{s_1, \ldots, s_n\}$ is defined as $Pos(S) = \{\langle i, j, o\rangle \mid 1 \leq i \leq n \wedge 1 \leq j \leq |s_i| \wedge o = \pm 1\}$, where $|s_i|$ denotes the length of sequence s_i. The representation $G = \langle V, E, l, \pi\rangle$ decomposes S-sequences into the concatenation of the labels of their oriented nodes. In this decomposition, each nucleotide $s_i[j]$ corresponds to an occurrence of a specific nucleotide in $l(\langle v, o\rangle)$ for some G-node v that occurs in $\pi(s_i)$ with orientation o. In this case, we say that the nucleotide from $l(\langle v, o\rangle)$ occurs in S at position $\langle i, j, o\rangle$, and its complementary nucleotide from $l(\langle v, -o\rangle)$ occurs in S at position $\langle i, j, -o\rangle$.

Two positions $\langle i, j, o\rangle, \langle i', j', o'\rangle \in Pos(S)$ are *homologous* with respect to G if they correspond to occurrences of the same nucleotide in the label of an oriented node shared by the paths $\pi(s_i)$ and $\pi(s_{i'})$. The set of all pairs of positions homologous with respect to G is called homology relation *associated* with G, and denoted by h_G (see Figure 1).

It follows directly from the definition that h_G is an equivalence relation on $Pos(S)$. Moreover, the equivalence classes of h_G that group occurrences of complementary nucleotides in $l(\langle v, o\rangle)$ and $l(\langle v, -o\rangle)$, contain positions with inverse orientations, i.e.

$$[\langle i, j, -o\rangle]_{h_G} = \{\langle i', j', -o'\rangle \mid \langle i', j', o'\rangle \in [\langle i, j, o\rangle]_{h_G}\}. \tag{1}$$

It should be noted that the structure of the homology relation is strictly determined by the underlying variation graph representation. The degree to which it reflects homology in the evolutionary sense depends on the specific graph construction method.

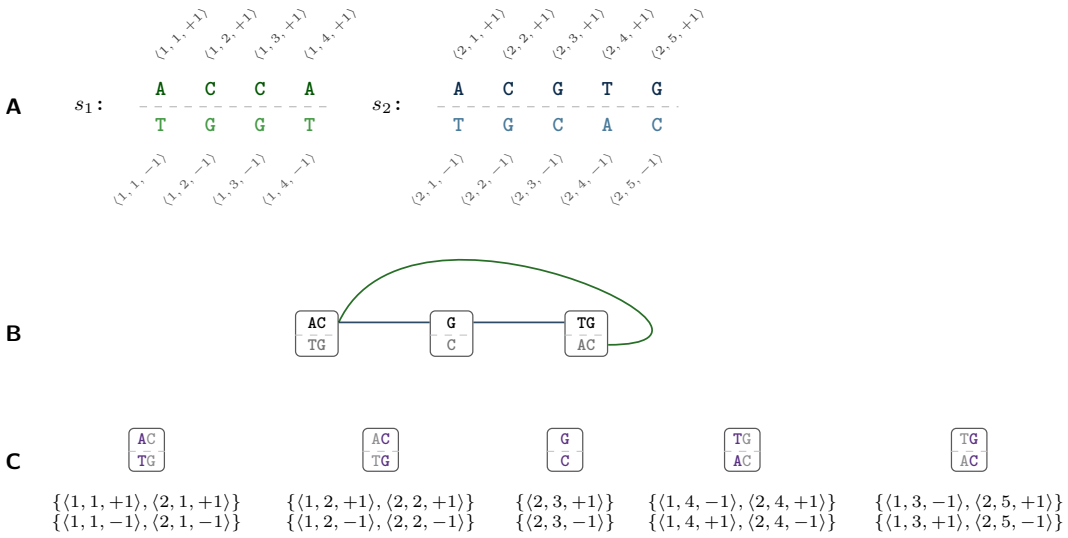

Figure 1 Variation graph representation of two DNA sequences and the associated homology relation. (A) Sequence set $S = \{s_1, s_2\}$ presented alongside the elements of $Pos(S)$. Residues and positions on the $'+'$ strand are shown over the dashed lines, and those on the $'-'$ strand are shown under these lines. (B) Variation graph representing S. Consecutive nodes on $\pi(s_1)$ and on $\pi(s_2)$ are connected with green and blue edges, respectively. Sequences s_1 and s_2 are spelled by the labels of the nodes that are traversed by the respective paths from left to right (thus oriented $'+1'$), and the reverse complements of those traversed from right to left (thus oriented $'-1'$). (C) Equivalence classes of the homology relation induced by the graph shown in (B). Each pair of complementary nucleotides in each node label (highlighted in purple) induces two classes, grouping the positions, on which these nucleotides occur in S-sequences.

2.3 Equivalent and canonical graphs

We say that two representations G and G', of a sequence set S are *equivalent* if they induce the same homology relation on $Pos(S)$. Isomorphic representations must be equivalent, but the opposite implication is not true. Below, we define structural modifications to the representation that do not alter the induced homology relations.

Given a path $\langle\langle v_1, o_1\rangle, \ldots, \langle v_m, o_m\rangle\rangle$, we refer to side $-o_1$ of node v_1 and side o_m of node v_m as *outer sides* of this path, while all other sides of its nodes are *inner* in this path. When all nodes in the path are different and each its inner side is incident with only one edge (connecting consecutive path nodes), the path is a *unitig*. A unitig having only one node is called *trivial*. *Splitting* a node v consists of replacing it with a unitig having $|l(v)|$ nodes labeled with subsequent $l(v)$ characters. Splitting operation can be reversed by *compressing* a unitig into a single node, labeled with the concatenation of the labels of the unitig nodes. Unitig is called *safe*, if none of its inner sides is an outer side of a genomic path (in particular, unitigs created with the splitting operation are safe). It is easily seen that the operations of splitting a node and compressing a safe unitig result in an equivalent representation.

A representation is *singular* if all its nodes are labeled with single characters (and thus cannot be further split). A representation is *compact* if all its safe unitigs are trivial (and thus cannot be further compressed). Below, we prove that every equivalence class of variation graph representations contains both a singular and a compact representation, each possessing a uniquely determined structure.

▶ **Theorem 1.** *Equivalent singular variation graph representations of a sequence set S are isomorphic.*

Proof. Assume G and G' are singular representations of S, inducing the same homology relation h on $Pos(S)$. Due to singularity, each oriented G-node $\langle v, o\rangle$ is labeled with a single nucleotide, the S-occurrences of which form an equivalence class of h. This class must also group the positions of the S-occurrences of some oriented G'-node $\langle v', o'\rangle$. The above correspondence establishes the bijection b defined as $b(\langle v, o\rangle) = \langle v', o'\rangle$ for all such pairs of oriented nodes in G and G'.

We show that b satisfies conditions 1-4 of the isomorphism definition. Conditions 1-2 follow directly from property (1) of homology relations. To show condition 4 let us denote oriented nodes representing position $\langle i, j, +1\rangle$ in G and G' by $\langle v_{i,j}, o_{i,j}\rangle$ and $\langle v'_{i,j}, o'_{i,j}\rangle$, respectively. The definition of b implies that $b(\langle v_{i,j}, o_{i,j}\rangle) = \langle v_{i,j}, o_{i,j}\rangle$ for all respective i and j. Moreover, each sequence $s_i \in S$ is represented by path $\pi(s_i) = \langle\langle v_{i,1}, o_{i,1}\rangle, \ldots, \langle v_{i,|s_i|}, o_{i,|s_i|}\rangle\rangle$ in G and by path $\pi'(s_i) = \langle\langle v'_{i,1}, o'_{i,1}\rangle, \ldots, \langle v'_{i,|s_i|}, o'_{i,|s_i|}\rangle\rangle$ in G'. Combining the above observations yields condition 4. Finally, condition 3 is the consequence of the fact that the adjacency structure of a variation graph is induced by its genomic paths. Namely, for every G-edge $\{\langle v_1, o_1\rangle, \langle v_2, o_2\rangle\}$, either $\langle\langle v_1, o_1\rangle, \langle v_2, -o_2\rangle\rangle$ or $\langle\langle v_2, o_2\rangle, \langle v_1, -o_1\rangle\rangle$ is a pair of two consecutive oriented nodes in path $\pi(s_i)$ for some $i \in \{1, \ldots, n\}$. Consequently, either $\langle b(\langle v_1, o_1\rangle), b(\langle v_2, -o_2\rangle)\rangle$ or $\langle b(\langle v_2, o_2\rangle), b(\langle v_1, -o_1\rangle)\rangle$ is a pair of two consecutive oriented nodes in $\pi'(s_i)$, so G' must contain edge $\{b(\langle v_1, o_1\rangle), b(\langle v_2, o_2\rangle)\}$. Similar argument applies to the opposite implications, which completes the proof. ◀

▶ **Theorem 2.** *Equivalent compact variation graph representations of a sequence collection are isomorphic.*

Proof. We first show that two different maximal safe unitigs cannot share a node. To this aim assume that two such unitigs share one. By definition, this node can be neither followed nor preceded by two different nodes within these unitigs. Since the unitigs are different, the terminal node of one unitig must be an internal node of the other one. Thus, the path forming the first unitig can be extended at this end with the next node from the other unitig. This contradicts the maximality of the first unitig, unless the node extending the path already belongs to it. However, in this case the unitig forms a maximal non-cyclic path on a cycle, in which every node is incident only to the edges of that cycle. This means that the cycle forms a separate connected component of the graph. Every node in this component must be covered by a genomic path, which therefore must terminate within it. Consequently, the cycle contains a terminal node of a genomic path, and thus an outer side of this path must be an inner side of at least one of the considered unitigs. This contradicts the assumption that both unitigs are safe.

The consequence of what we have shown is that two maximal safe unitigs that share a node are either identical or reversed paths. Therefore, the maximal safe unitigs partition the set of graph nodes into subsets that can be compressed independently. Thus every compact representation is the result of compressing all maximal safe unitigs in an equivalent singular representation. When two compact representations are equivalent, the singular representations are also equivalent, and hence isomorphic. As a result, equivalent compact representations are isomorphic too. ◀

2.4 Graph comparison

Different variation graphs representing the same sequence set are usually compared based on graph structure characteristics, e.g. the number of nodes, the length of node labels, or the number of genomic paths covering the node. In general, such measures poorly reflect the similarity between the homologies of represented sequences; graphs with very similar

structural properties can have completely different homology relations, while equivalent singular and compact representations can significantly differ in the size of their graphs and the length of node labels. A better approximation of the similarity between homologies is given by the *segmentation distance* [5]. Each edge on a genomic path establishes a *breakpoint* between two adjacent positions in the represented sequence. The set of all breakpoints on a genomic sequence is its *segmentation*. The segmentation distance is the size of symmetric difference between the segmentations induced by the representations. The representations are assumed to be compact, so the edges separate regions of homogeneous homology relations, and the segmentation distance quantifies the differences between the projections of these regions on the genomic sequences. Although projections capture homology more accurately than structural graph characteristics, the segmentation distance may still distort the representation of similarity between homologies. This effect is illustrated in Figure 2, where the distance between Graph B and Graph C is larger than between Graph A and Graph B, despite the latter pair differing more substantially in their encoded homology relations. This indicates that segmentation distance remains more sensitive to changes in segmentation and graph topology than to the underlying homology structure itself.

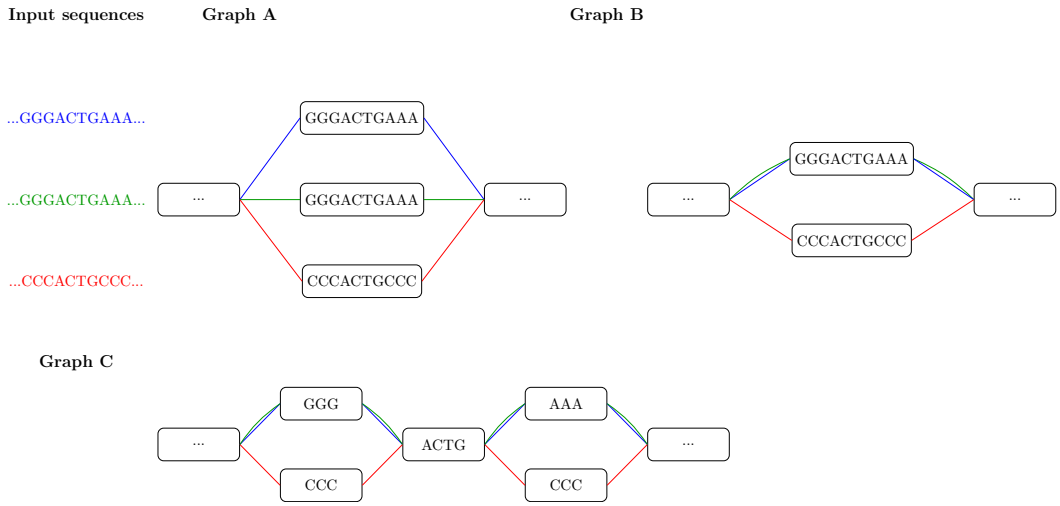

Figure 2 An example illustrating limitations of segmentation distance. The figure shows different subgraphs representing fragments of three DNA sequences. Dots indicate that the displayed subgraphs continue beyond the illustrated region. Multi-edges are used only for visualization purposes, and edge colors correspond to the respective genomic paths. In Graph A, each sequence fragment is represented by a separate node. In Graph B, the identical blue and green sequences are represented by a common node. In Graph C, the shared segment `ACTG`, present in all three sequences, is merged into single. The blue and green sequences are represented by the identical path. Graphs A and B have segmentation distance equal to 0, despite representing substantially different homology relationships. In contrast, the segmentation distance between Graphs B and C equals 6, even though Graph C preserves most of the homology structure represented in Graph B. In both graphs, the two identical sequence fragments remain recognized as homologous, while Graph C additionally merges the shared segment `ACTG` of the third sequence.

The limitations of the above approaches to evaluating variation graph similarity can be overcome by directly comparing the underlying homology relations. This comparison can be performed using any standard set similarity measure, e.g. the Jaccard distance. We apply this distance to the relations $\widehat{h_G} = h_G \setminus Id_{Pos(S)}$, where $Id_{Pos(S)}$ is the identity relation on

the set of all positions in S. The reason for this is that pairs of identical positions belong to every homology relation on $Pos(S)$. Thus, the *homology distance* between two homology relations is defined by the following formula:

$$d(h_1, h_2) = 1 - \frac{|\widehat{h_1} \cap \widehat{h_2}|}{|\widehat{h_1} \cup \widehat{h_2}|}$$

and between two variation graphs by $d_h(G_1, G_2) = d(h_{G_1}, h_{G_2})$.

▶ **Proposition 3.** *Distance function d is a metric on the set of homology relations on $Pos(S)$.*

Proof. Since d is defined via the Jaccard distance between finite sets of non-trivial homology pairs, it directly inherits the metric properties of the Jaccard distance over finite sets. ◀

The distance function d_h satisfies the metric properties except the identity axiom, since $d_h(G_1, G_2) = 0$ whenever G_1 and G_2 are equivalent. However, both d and d_h satisfy the triangle inequality, unlike most heuristic similarity scores.

2.5 Graph reconciliation

Consider two representations G_1 and G_2 of a sequence set S. Representation G of S is the *reconciliation* of G_1 and G_2 if $h_G = h_{G_1} \cap h_{G_2}$.

▶ **Theorem 4.** *The singular (respectively compact) reconciliation of representations G_1 and G_2 of the same sequence set S always exists and is unique up to isomorphism.*

Proof. First we will show the construction of a singular S-representation with homology relation $h = h_{G_1} \cap h_{G_2}$. Since both h_{G_1} and h_{G_2} are equivalence relations, h is an equivalence relation, too. A similar argument leads to the observation that the equivalence classes $[\langle i, j, +1\rangle]_h$ and $[\langle i, j, -1\rangle]_h$ group positions with the same coordinates i, j and inverse orientations.

A singular representation G of S is constructed as follows:

- each node of G represents a pair of h-classes grouping positions with inverse orientations,
- the two classes within the pair correspond to the two possible node orientations,
- consequently, every h-class c is represented by an oriented node, denoted below by $vo(c)$,
- $l(vo([\langle i, j, +1\rangle]_h)) = s_i[j]$ and $l(vo([\langle i, j, -1\rangle]_h))$ is the complement of $s_i[j]$,
- $\pi(s_i) = \langle vo([\langle i, 1, +1\rangle]_h), \ldots, vo([\langle i, |s_i|, +1\rangle]_h)\rangle$ for $i = 1, \ldots, n$,
- the edges of G connect nodes that are adjacent along the paths $\pi(s_i)$, respecting their orientations.

Resulting singular representation can be transformed into a compact one by compressing its maximal safe unitigs into single nodes, as in the proof of Theorem 2.

Finally, the uniqueness of singular and compact representations is a direct consequence of their existence and Theorems 1 and 2. ◀

Reconciliation can be applied to variation graph evaluation in several ways. First, it simplifies the problem of homology distance calculation – once we compute the sizes of homology relations induced by G_1, G_2 and their reconciliation G, their homology distance is given by formula

$$d_h(G_1, G_2) = 1 - \frac{|\widehat{h_G}|}{|\widehat{h_{G_1}}| + |\widehat{h_{G_2}}| - |\widehat{h_G}|}.$$

Second, the ratios $|\widehat{h_G}|/|\widehat{h_{G_1}}|$ and $|\widehat{h_G}|/|\widehat{h_{G_2}}|$ serve as asymmetric similarity measures, describing the proportion of position pairs in one relation that also belong to the other. We will call each of these ratios the *homology reconciliation index (HRI)*, and denote by $HRI_1(G_1, G_2)$ and $HRI_2(G_1, G_2)$, respectively. When one of the representations is evaluated with respect to the other (serving as a *gold standard*), the ratios establish the statistics *precision* and *recall*.

Finally, we generalize the concept of reconciliation to any number of input representations: G is a reconciliation of representations $G_1, \ldots, G_n$ if $h_G = h_{G_1} \cap \ldots \cap h_{G_n}$. It can be computed by iteratively applying the above construction, and the result does not depend on the order of input representations. The notion of HRI is generalized accordingly:

$$HRI_i(G_1, \ldots, G_n) = |\widehat{h_G}|/|\widehat{h_{G_i}}|.$$

When applied to representations constructed in different runs of the same graph building tool, HRIs can be used to quantify the stability of its construction algorithm. Their values range from 0 to 1, where HRI of exactly 1 indicates that all homology pairs from the individual run are perfectly preserved within the reconciled graph. Conversely, lower values indicate that the individual graph instance induces a broader homology relation, containing pairs that are not universally shared across the other runs.

2.6 Computing reconciliation graph and similarity measures

All we need to calculate the above mentioned measures is to compute the size of relation $\widehat{h_G}$ for a given graph $G = \langle V, E, l, \pi \rangle$ and to construct the reconciliation of two given graphs G_1 and G_2.

The first task is straightforward. It is easily seen that a node $v \in V$ that has k_v occurrences in genomic paths of graph G, induces $2|l(v)|$ equivalence classes of relation h_G, and each of these classes consists of k_v positions. Moreover, an h_G-class with k elements contributes k^2 pairs to relation h_G, and thus $k(k-1)$ pairs to relation $\widehat{h_G}$. Consequently, the size of $\widehat{h_G}$ is given by formula

$$|\widehat{h_G}| = \sum_{v \in V} 2|l(v)|k_v(k_v - 1).$$

Therefore, $|\widehat{h_G}|$ can be computed in linear time with respect to the size of graph G.

A naive approach to the second task (by enumerating all pairs in $\widehat{h_{G_1}}$ and $\widehat{h_{G_2}}$, intersecting the results, and building the singular graph as in the proof of Theorem 4) leads to an algorithm with $O(k_{max}^2 N)$ time and space complexity, where $k_{max} = \max_{v \in V} k_v$ and N is the total length of genomic sequences. This is too much when it comes to processing large genomic data sets.

Below we propose a reconciliation construction algorithm, whose complexity is independent on both parameters. Given variation graph $G = \langle V, E, l, \pi \rangle$, we define $P_G = \sum_{v \in V} k_v$. Note that all components of G except l can be encoded in space $O(P_G)$. Our algorithm reconciles variation graphs G_1 and G_2 in time and space depending on parameters P_{G_1} and P_{G_2} only. To shorten the notation, we denote these parameters below as P_1 and P_2, respectively.

A node v in a reconciliation graph G is *trivial* if $k_v = 1$. Because trivial nodes induce only singleton equivalence classes of h_G, they contribute no pairs to $\widehat{h_G}$. Therefore, they are redundant for the computation of $|\widehat{h_G}|$, so our algorithm omits them from the output reconciliation graph. When a complete reconciliation graph is needed, the output can be postprocessed using the GFALace tool [8], which recovers trivial nodes and reconstructs genomic paths.

In our algorithm the segmentation induced by G_1 serves as a reference. Its breakpoints are sorted according to their order on the genomic sequences, and the overlaps between a query sequence fragment and the reference segments can be computed via binary search. The nodes of G_2 are processed consecutively and their occurrences are queried in this structure. Processing a node v yields non-trivial nodes representing those sequence positions in the reconciliation graph that were represented by v in G_2. This is done in the following steps:

1. **Breakpoint detection.** For each occurrence of v, all G_1-breakpoints located within this occurrence are identified, along with respective sides of the two G_1-nodes, whose occurrences the breakpoint separates. Moreover, for each breakpoint the corresponding position in $l(v)$ is calculated.
2. **Sorting and grouping.** Breakpoints identified within all occurrences are sorted together according to their $l(v)$-positions and split into groups sharing this position.
3. **Creating reconciliation nodes.** Each group of breakpoints is further divided into subgroups sharing the same G_1-node and its side. The division is performed twice, with respect to G_1-nodes representing fragments starting and ending at the $l(v)$-position. Subgroups assigned to opposite sides of the same G_1-node are called *matching*. Then three types of reconciliation nodes can be created:
   - v occurrences with no G_1-breakpoint inside form a single node,
   - each pair of matching subgroups yields a node, whose occurrences form a subset of the occurrences of respective G_1-node,
   - each non-matching subgroup yields a node, whose occurrences have one end inherited from the respective G_1-node and its side, and the other end inherited from one of sides of v.

An example illustrating the reconciliation algorithm is presented in Figure 3. Below we analyze its computational complexity.

▶ **Theorem 5.** *The time complexity of the reconciliation algorithm is* $O((P_2 + P_1) \log P_1)$.

Proof. The construction of the reference graph structure is dominated by sorting its breakpoints, which requires $O(P_1 \log P_1)$ time. Below we analyze separately the cost of each step of processing a G_2-node, summed up over all G_2-nodes.

1. The algorithm iterates through all P_2 genomic occurrences of G_2-nodes. For each occurrence, the algorithm performs the binary search over the sorted vector of G_1-breakpoints to locate the occurrence starting position. This requires $O(\log P_1)$ time, so the total time of all searches is $O(P_2 \log P_1)$. Each search is followed by enumerating G_1-breakpoints located within the occurrence. Since each G_1-breakpoint belongs to only one G_2-node occurrence, the total time of the enumeration phase is $O(P_1)$. Thus the total time of step 1 is $O(P_1 + P_2 \log P_1)$.
2. Denote the number of G_1-breakpoints that are located within the occurrences of a G_2-node v by P_v. Then sorting these breakpoints needs $O(P_v \log P_v)$ time, and sorting the breakpoints located within the occurrences of all G_2-nodes needs time
$$O(\sum_{v \in V_2} P_v \log P_v) = O(\sum_{v \in V_2} P_v \log P_1) = O(P_1 \log P_1).$$
3. Using the lookup tables, the whole step for a G_2-node v can be executed in a single iteration over the sorted set of the sides of G_1-breakpoints located within v-occurrences, enhanced with the ends of these occurrences. This needs $O(P_v + k_v)$ time, resulting in $O(\sum_{v \in V_2}(P_v + k_v)) = O(P_1 + P_2)$ time for all G_2-nodes.

Combining the above results we obtain the total time complexity $O((P_2 + P_1) \log P_1)$, which completes the proof. ◀

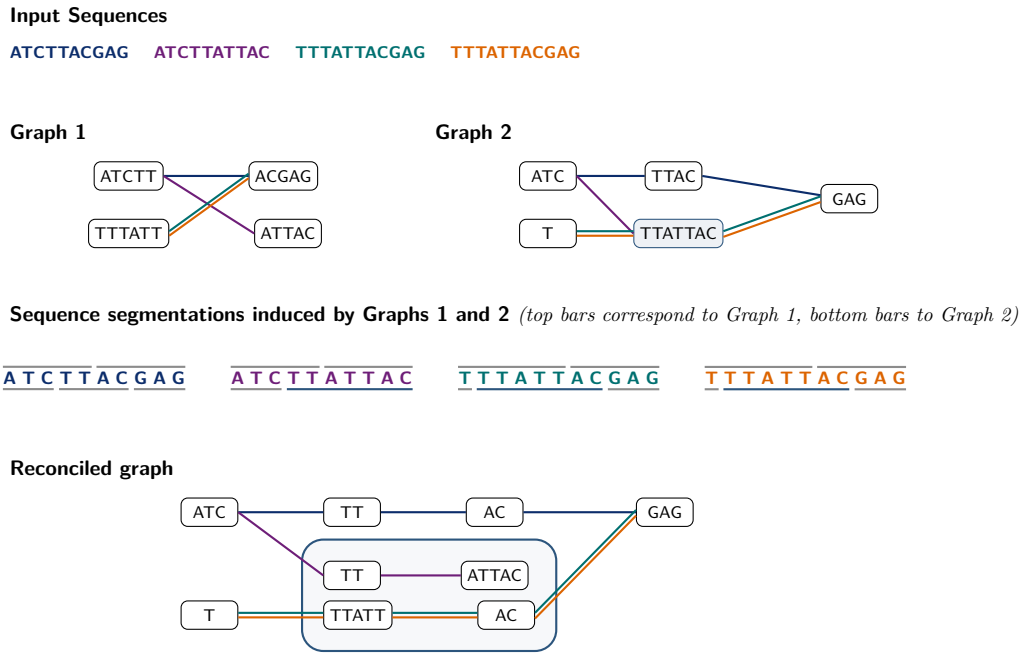

Figure 3 Workflow of the variation graph reconciliation algorithm. The figure illustrates the input sequences, their individual graph structures (Graph 1 and Graph 2), input sequence segmentations induced by both graphs (top bars correspond to segmentation induced by Graph 1, bottom bars to Graph 2, with the segments corresponding to the highlighted node from Graph 2 marked in blue), and the final output as a reconciled graph, where the blue rectangle indicates how the shaded node from Graph 2 has been partitioned in the reconciled graph.

▶ **Theorem 6.** *The space complexity of the reconciliation algorithm is* $O(P_1 + P_2)$.

Proof. First, the segmentation data for the reference graph G_1 is stored using parallel vectors – specifically dedicated to the starting positions, corresponding node IDs, and orientations of each interval – requiring $O(P_1)$ space. Similarly, the grouped interval structures for the query graph G_2 require $O(P_2)$ space.

Second, processing a G_2-node v requires local structures for storing ends of v-occurrences and G_1-breakpoints located within these occurrences. The size of these structures is $O(P_v+k_v)$ (using the notation from the proof of Theorem 5). Thus all the structures used in the algorithm have their size bounded by $O(P_1 + P_2)$, which completes the proof. ◀

The C++ implementations of the reconciliation algorithm and of the $|\widehat{h_G}|$ calculator are available at `https://github.com/anialisiecka/vgrecon/`.

3 Experiments

We conducted several series of experiments. First, we investigated the structural stability of variation graph construction tools across multiple runs. Second, we compared homology and segmentation distances calculated for variation graphs built using different tools. Third, we evaluated the scalability of our graph reconciliation algorithm by varying the number of genomes in the input dataset.

To evaluate the stability of variation graph construction tools, we generated 10 graph instances for Minigraph-Cactus, PGGB, AlfaPang and AlfaPang with graph simplification steps of PGGB (namely `smoothxg`), which we denote AlfaPang+ and measured their homology reconciliation index. The datasets were constructed using 10 human haplotypes

of chromosome 6. For Minigraph-Cactus, we varied the reference sequence across the 10 runs to analyze how the choice of reference affects the final graph structure. For PGGB, we executed the tool 10 times with identical parameters to evaluate the impact of its inherent non-determinism. For AlfaPang and AlfaPang+, identical configurations were used across all 10 runs to evaluate structural consistency under alignment-free and post-processed workflows. For AlfaPang+, the analysis evaluated two distinct graph refinement configurations using `smoothxg`: the first variant used its native default parameters, while the second variant applied the default parameters native to PGGB.

The analysis of the homology reconciliation index (see Table 1) reveals significant differences in the structural stability of the graphs generated by these tools. For Minigraph-Cactus, a relatively low degree of similarity between instances was observed, with a mean similarity of 0.1 and values spanning from 0.06 to 0.19. This is a direct consequence of the algorithm's sensitivity to the choice of the reference sequence – changing the reference during construction significantly alters the final graph topology and node segmentation, resulting in divergent homology representations.

In contrast, PGGB exhibits a markedly different characteristic. Despite its inherent non-determinism, the generated graphs show an extremely high level of structural consistency relative to the reconciled graph, maintaining HRI between 0.9961 and 0.9964. Such stable values indicate that the graphs produced by PGGB remain nearly identical across all trials, with minor fluctuations having only a marginal impact on the final genomic representation.

The next tool, AlfaPang, has a fully deterministic algorithm, assuring perfect reproducibility across different runs, which is reflected by HRI constantly equal to 1.

In contrast, the inclusion of a graph refinement phase in the AlfaPang+ workflow introduces notable structural changes, the extent of which depends heavily on the chosen post-processing parameters. When using the native default settings of `smoothxg`, the mean HRI decreases to 0.65, with individual values ranging from 0.62 to 0.66. However, when AlfaPang+ is configured with the PGGB-native parameters for `smoothxg`, the structural consistency improves drastically. This variant achieves a very high mean HRI of 0.9915, closely matching the performance of PGGB itself, with values tightly bounded between 0.9914 and 0.9917. This comparison highlights that the choice of `smoothxg` parameters affects not only the final graph structure, but also the reproducibility of the graphs across independent runs.

Table 1 Summary of the homology reconciliation index between input variation graphs and their respective reconciliations. The analysis covers the mean, standard deviation, minimum, and maximum values across 10 runs of each tool. For Minigraph-Cactus, the variation originates from using different reference sequences during graph construction. For PGGB, the differences are a result of the algorithm's inherent non-determinism. For AlfaPang, the zero variance reflects its deterministic and alignment-free nature, while for AlfaPang+, the fluctuations arise from the subsequent graph refinement phase.

Statistic	Minigraph-Cactus	PGGB	AlfaPang	AlfaPang+[1]	AlfaPang+[2]
Mean	0.1007	0.9963	1	0.6474	0.9915
SD	0.0424	0.0001	0	0.0134	0.0001
Min	0.0567	0.9961	1	0.6197	0.9914
Max	0.1866	0.9964	1	0.6621	0.9917

1 Refinement step performed using `smoothxg` with its native default parameters.
2 Refinement step performed using `smoothxg` with default parameters from PGGB.

For the second experiment, we used simulated bacterial genomes previously employed for the evaluation of SibeliaZ [10]. The datasets were generated using ALF [4], a tool designed to simulate bacterial genome evolution. The evolutionary model of ALF incorporates both point mutations (insertions, deletions, and substitutions) and large-scale structural variations, including translocations and gene gain or loss events. We selected six distinct datasets simulated at varying phylogenetic distances from their common ancestor, with divergence levels ranging from 0.03 to 0.18 substitutions per site. Each dataset comprises 10 bacterial genomes, approximately 1.5 Mbp in size and containing 1500 genes. The raw genomic sequences from each dataset served as direct input to four distinct variation graph construction tools: PanGenome Graph Builder (PGGB) [6], Minigraph-Cactus [9], AlfaPang and AlfaPang+ [3].

All these tools were applied to genomic sequences generated by ALF and the output variation graphs were compared using two metrics: the homology distance and the normalized version of the segmentation distance. The original segmentation distance was defined in [5] as the number of sequence breakpoints induced by only one of the two graphs. We divide this number by the total number of breakpoints induced by both graphs, which ensures that the distance falls in the range [0,1], and thus it is directly comparable with the homology distance. Figure 4 summarizes the results. For each pair of variation graph-building tools, both metrics follow the same trend – the distances increase with increasing divergence between input sequences. Moreover, both metrics have very similar values for the comparison of graphs constructed using PGGB, Minigraph-Cactus and AlfaPang+. On the other hand, when comparing a graph created using AlfaPang with a graph from another tool, the segmentation distance is much larger than the homology distance, especially for datasets with low genomic divergence. This effect arises, because, without additional graph simplification, AlfaPang

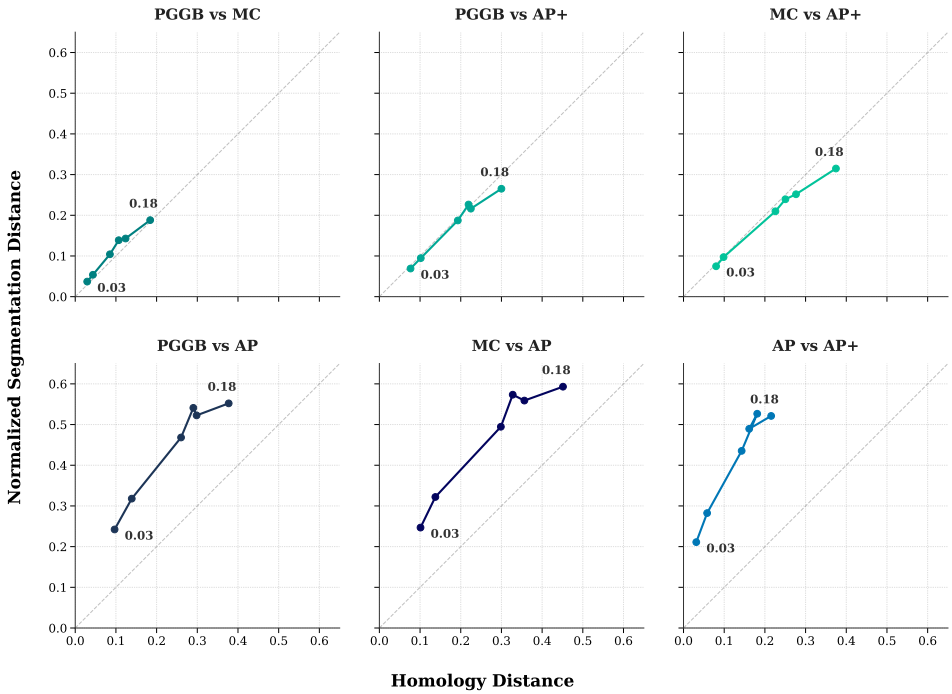

Figure 4 Distances between different variation graphs. Numerical labels correspond to the genomic divergence of the simulated dataset. MC: Minigraph-Cactus, AP: AlfaPang, AP+: AlfaPang+.

may merge relatively short sequence fragments into single nodes. As a consequence, longer homologous sequences can become fragmented into multiple vertices, similarly to the effect illustrated in the Figure 2. Furthermore, while this fragmentation creates numerous artificial breakpoints – which drastically increases the segmentation distance – these merged fragments are of limited length. Consequently, they have a negligible impact on the total homology relation, leaving the homology distance largely unchanged. This highlights a major advantage of our metric: while structural distances fluctuate drastically based on variations in sequence partitioning, the homology distance scales with the lengths of sequence fragments with modified homology.

To evaluate the scalability of our tool, we conducted experiments on datasets consisting of 50, 100, 200, and 400 *Escherichia coli* genomes, allowing us to assess performance across varying input sizes. For each dataset, two variation graphs were generated using PGGB and Minigraph-Cactus, respectively, and used as an input for the reconciliation construction algorithm. Performance results (see Figure 5) illustrate the resource requirements for the core reconciliation algorithm and the postprocessing step, performed with the GFALace tool. The evaluation shows that our algorithm scales efficiently with the number of genomes. This reflects the efficiency achieved by operating at the graph topology level rather than processing raw, redundant sequence lengths. Surprisingly, the postprocessing step introduces much larger computational footprint – it is $\sim 4\times$ slower and requires $\sim 2\times$ more RAM.

4 Conclusions and discussion

We introduced the notion of homology relation, which describes in a mathematically rigorous way how a variation graph models the sequence homology at the level of single nucleotides. We analyzed the structure of classes of variation graphs sharing the same homology relation and characterized their canonical representatives. Furthermore, we used this concept to define the notion of a variation graph that reconciles two or more graphs, and developed an efficient algorithm for its construction.

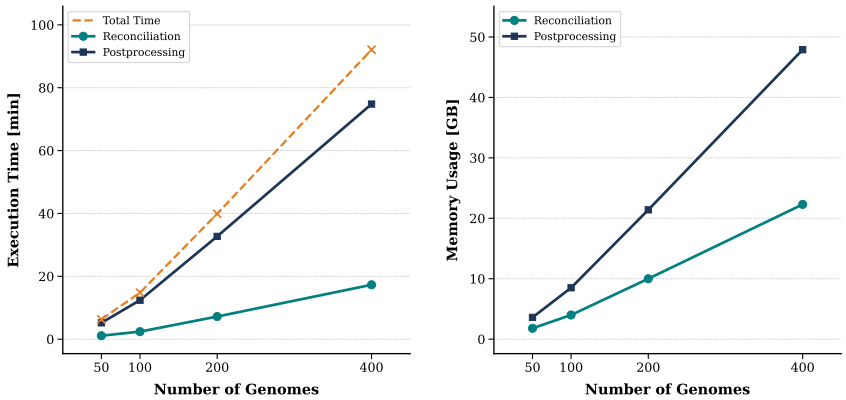

Figure 5 Computational resource consumption analysis across different pangenome sizes. Left panel illustrates the execution time for the reconciliation phase and the postprocessing phase, as well as the total execution time. Right panel depicts memory usage across both phases. The experiments were performed on variation graphs constructed using PGGB and Minigraph-Cactus pipelines, from 50, 100, 200, and 400 *E. coli* genomes. The computations were performed on a Supermicro X10DRi server with 512GB RAM and two 14-cores CPUs Intel Xeon E5-2690V4, using a single thread.

Moreover, we proposed two homology-based measures that can be used to evaluate variation graphs. The first one, homology distance, is a metric for comparing graphs representing the same genome collection. Unlike heuristic similarity scores based on graph structural features, homology distance directly compares represented homologies while satisfying the axioms of metric. The second measure, homology reconciliation index, can be used to evaluate in more detail a given graph against the true one, and to assess the stability of variation graph construction algorithms. In particular, it quantifies the extent to which a homologous relation is consistently reproduced across different runs of the same pipeline, and the extent to which it is run-specific. In this way, it can support the construction of reference pangenome graphs by characterizing the impact of inherent non-determinism in graph building tools or of their parameter settings.

We implemented efficient algorithms to compute the above measures and made the software available at `https://github.com/anialisiecka/vgrecon/`. Finally, we verified the applicability of our contribution in a series of experiments.

To the best of our knowledge, the reconciliation operation proposed in this work is the first method to combine multiple variation graphs representing the same set of sequences into a single graph. It is based on the paradigm that the resulting graph should contain only homology relationships that are consistently accepted by all input graphs. Although fully justified in the above evaluation schemes, in the context of the graph combining problem this paradigm may be considered overly conservative, because it can eliminate biologically meaningful homologies missed in single input graphs. The development of less conservative approaches (e.g. union- or consensus-based) is a subject of future work.

References

1 Francesco Andreace, Pierre Lechat, Yoann Dufresne, and Rayan Chikhi. Comparing methods for constructing and representing human pangenome graphs. *Genome Biology*, 24(1), November 2023. doi:10.1186/s13059-023-03098-2.

2 Adam Cicherski and Norbert Dojer. From de bruijn graphs to variation graphs – relationships between pangenome models. In Franco Maria Nardini, Nadia Pisanti, and Rossano Venturini, editors, *String Processing and Information Retrieval 2023*, pages 114–128, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-43980-3_10.

3 Adam Cicherski, Anna Lisiecka, and Norbert Dojer. Alfapang: alignment free algorithm for pangenome graph construction. *Algorithms Mol. Biol.*, 20(1):7, 2025. doi:10.1186/S13015-025-00277-7.

4 Daniel A. Dalquen, Maria Anisimova, Gaston H. Gonnet, and Christophe Dessimoz. Alf—a simulation framework for genome evolution. *Molecular Biology and Evolution*, 29(4):1115–1123, December 2011. doi:10.1093/molbev/msr268.

5 Siegfried Dubois, Matthias Zytnicki, Claire Lemaitre, and Thomas Faraut. Pairwise graph edit distance characterizes the impact of the construction method on pangenome graphs. *Bioinformatics*, 41(6), May 2025. doi:10.1093/bioinformatics/btaf291.

6 Erik Garrison, Andrea Guarracino, Simon Heumos, Flavia Villani, Zhigui Bao, Lorenzo Tattini, Jörg Hagmann, Sebastian Vorbrugg, Santiago Marco-Sola, Christian Kubica, David G. Ashbrook, Kaisa Thorell, Rachel L. Rusholme-Pilcher, Gianni Liti, Emilio Rudbeck, Agnieszka A. Golicz, Sven Nahnsen, Zuyu Yang, Moses Njagi Mwaniki, Franklin L. Nobrega, Yi Wu, Hao Chen, Joep de Ligt, Peter H. Sudmant, Sanwen Huang, Detlef Weigel, Nicole Soranzo, Vincenza Colonna, Robert W. Williams, and Pjotr Prins. Building pangenome graphs. *Nature Methods*, 21(11):2008–2012, October 2024. doi:10.1038/s41592-024-02430-3.

7 Erik Garrison, Jouni Sirén, Adam M Novak, Glenn Hickey, Jordan M Eizenga, Eric T Dawson, William Jones, Shilpa Garg, Charles Markello, Michael F Lin, Benedict Paten, and Richard Durbin. Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nat Biotechnol*, 36(9):875–879, October 2018. doi:10.1038/nbt.4227.

8 Andrea Guarracino. Gfalace, July 2025. URL: `https://github.com/pangenome/gfalace`.

9 Glenn Hickey, Jean Monlong, Jana Ebler, Adam M. Novak, Jordan M. Eizenga, Yan Gao, Haley J. Abel, Lucinda L. Antonacci-Fulton, Mobin Asri, Gunjan Baid, Carl A. Baker, Anastasiya Belyaeva, Konstantinos Billis, Guillaume Bourque, Silvia Buonaiuto, Andrew Carroll, Mark J. P. Chaisson, Pi-Chuan Chang, Xian H. Chang, Haoyu Cheng, Justin Chu, Sarah Cody, Vincenza Colonna, Daniel E. Cook, Robert M. Cook-Deegan, Omar E. Cornejo, Mark Diekhans, Daniel Doerr, Peter Ebert, Jana Ebler, Evan E. Eichler, Susan Fairley, Olivier Fedrigo, Adam L. Felsenfeld, Xiaowen Feng, Christian Fischer, Paul Flicek, Giulio Formenti, Adam Frankish, Robert S. Fulton, Shilpa Garg, Erik Garrison, Nanibaa' A. Garrison, Carlos Garcia Giron, Richard E. Green, Cristian Groza, Andrea Guarracino, Leanne Haggerty, Ira M. Hall, William T. Harvey, Marina Haukness, David Haussler, Simon Heumos, Kendra Hoekzema, Thibaut Hourlier, Kerstin Howe, Miten Jain, Erich D. Jarvis, Hanlee P. Ji, Eimear E. Kenny, Barbara A. Koenig, Alexey Kolesnikov, Jan O. Korbel, Jennifer Kordosky, Sergey Koren, HoJoon Lee, Alexandra P. Lewis, Wen-Wei Liao, Shuangjia Lu, Tsung-Yu Lu, Julian K. Lucas, Hugo Magalhães, Santiago Marco-Sola, Pierre Marijon, Charles Markello, Tobias Marschall, Fergal J. Martin, Ann McCartney, Jennifer McDaniel, Karen H. Miga, Matthew W. Mitchell, Jacquelyn Mountcastle, Katherine M. Munson, Moses Njagi Mwaniki, Maria Nattestad, Sergey Nurk, Hugh E. Olsen, Nathan D. Olson, Trevor Pesout, Adam M. Phillippy, Alice B. Popejoy, David Porubsky, Pjotr Prins, Daniela Puiu, Mikko Rautiainen, Allison A. Regier, Arang Rhie, Samuel Sacco, Ashley D. Sanders, Valerie A. Schneider, Baergen I. Schultz, Kishwar Shafin, Jonas A. Sibbesen, Jouni Sirén, Michael W. Smith, Heidi J. Sofia, Ahmad N. Abou Tayoun, Françoise Thibaud-Nissen, Chad Tomlinson, Francesca Floriana Tricomi, Flavia Villani, Mitchell R. Vollger, Justin Wagner, Brian Walenz, Ting Wang, Jonathan M. D. Wood, Aleksey V. Zimin, Justin M. Zook, Tobias Marschall, Heng Li, and Benedict Paten. Pangenome graph construction from genome alignments with minigraph-cactus. *Nature Biotechnology*, 42(4):663–673, May 2023. `doi:10.1038/s41587-023-01793-w`.

10 Ilia Minkin and Paul Medvedev. Scalable multiple whole-genome alignment and locally collinear block construction with SibeliaZ. *Nat Commun*, 11(1):6327, December 2020. `doi:10.1038/s41467-020-19777-8`.

11 Adam M Novak, Erik Garrison, and Benedict Paten. A graph extension of the positional burrows-wheeler transform and its applications. *Algorithms Mol Biol*, 12:18, July 2017. `doi:10.1186/s13015-017-0109-9`.

12 Mikko Rautiainen and Tobias Marschall. GraphAligner: rapid and versatile sequence-to-graph alignment. *Genome Biol*, 21(1):253, September 2020. `doi:10.1186/s13059-020-02157-2`.

13 Jouni Sirén. Indexing variation graphs. In Sándor Fekete and Vijaya Ramachandran, editors, *2017 Proceedings of the Ninteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 13–27, Philadelphia, PA, January 2017. Society for Industrial and Applied Mathematics. `doi:10.1137/1.9781611974768.2`.