Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

OXFORD

Doc-Start

Sequence analysis

A variant selection framework for genome graphs

Chirag Jain [1,*], Neda Tavakoli [2], Srinivas Aluru [2]

[1]Department of Computational and Data Sciences, Indian Institute of Science, Bengaluru, India and [2]School of Computational Science and Engineering, Georgia Institute of Technology

*To whom correspondence should be addressed.

Abstract

Motivation: Variation graph representations are projected to either replace or supplement conventional single genome references due to their ability to capture population genetic diversity and reduce reference bias. Vast catalogues of genetic variants for many species now exist, and it is natural to ask which among these are crucial to circumvent reference bias during read mapping.

Results: In this work, we propose a novel mathematical framework for variant selection, by casting it in terms of minimizing variation graph size subject to preserving paths of length α with at most δ differences. This framework leads to a rich set of problems based on the types of variants (SNPs, indels), and whether the goal is to minimize the number of positions at which variants are listed or to minimize the total number of variants listed. We classify the computational complexity of these problems and provide efficient algorithms along with their software implementation when feasible. We empirically evaluate the magnitude of graph reduction achieved in human chromosome variation graphs using multiple α and δ parameter values corresponding to short and long-read resequencing characteristics. When our algorithm is run with parameter settings amenable to long-read mapping ($\alpha = 10$ kbp, $\delta = 1000$), 99.99% SNPs and 73% indel structural variants can be safely excluded from human chromosome 1 variation graph. The graph size reduction can benefit downstream pan-genome analysis.

Implementation: https://github.com/at-cg/VF

Contact: chirag@iisc.ac.in, neda.tavakoli@gatech.edu, aluru@cc.gatech.edu

1 Introduction

High-throughput technologies enable rapid sequencing of numerous individuals in a species population and cataloging observed variants. This is leading to a switch from linear representation of a chosen reference genome to graph representations depicting multiple observed haplotypes. Graph representations more accurately reflect the sampled individuals within a population, and their use in genome mapping algorithms reduces reference bias and increases mapping accuracy when sequencing a new individual (Ballouz *et al.*, 2019). There is abundant research on data structures designed for graph representations of genomes and pangenomes (Garrison *et al.*, 2018; Li *et al.*, 2020), their space-efficient indexing (Sirén *et al.*, 2014; Marcus *et al.*, 2014; Holley *et al.*, 2016; Ghaffaari and Marschall, 2019; Jain *et al.*, 2019b; Chang *et al.*, 2020; Kuhnle *et al.*, 2020), and alignment algorithms (Kuosmanen *et al.*, 2018; Jain *et al.*, 2020; Rautiainen and Marschall, 2020; Darby *et al.*, 2020; Ivanov *et al.*, 2020) to map sequences to reference graphs. For review papers summarizing these developments, see (Paten *et al.*, 2017; Computational Pan-Genomics Consortium, 2018; Eizenga *et al.*, 2020).

While graph representations have numerous advantages, complete variation graphs that include every variant have certain drawbacks. The graphs invariably contain paths combining variants across haplotypes, but never seen in any observed haplotype. The number of such recombinant paths increases combinatorially with graph size, and is particularly troublesome when mapping long reads which span greater distances. Accuracy of sequence-to-graph mapping algorithms shows diminishing returns at larger graph sizes, and is even negatively affected eventually (Pritt *et al.*, 2018; Sirén *et al.*, 2020). The computational complexity of the mapping algorithms makes it impractical to apply them at the scale of complete variation graphs. A few attempts have been made to address the first issue by augmenting paths with haplotype information and specifically developing haplotype-aware indexing strategies (Iqbal *et al.*, 2012; Sirén *et al.*, 2020; Mokveld *et al.*, 2020).

The aforementioned factors point to the need for variant selection algorithms which tame reference graph sizes, and strike the right balance for subsequent mapping accuracy and speed. This was primarily approached through selecting variants from a specific database (Schneeberger *et al.*, 2009; Danek *et al.*, 2014; Liu *et al.*, 2016), based on allelic frequency (Maciuca *et al.*, 2016; Eggertsson *et al.*, 2017; Kim *et al.*, 2018), or specific to a biological context such

as limiting to a particular population (Sirén *et al.*, 2014) or genome loci of interest (Vijaya Satya *et al.*, 2012; Dilthey *et al.*, 2015; Jain *et al.*, 2019a). Recently, Pritt *et al.* (2018) developed a more systematic approach FORGe by developing a mathematical model to prioritize variants, and selecting top scoring variants according to the model. In FORGe, the ranking of each variant is done based on its frequency in a population, and its contribution to runtime and space overhead of a read-to-graph mapper.

In this work we propose a rigorous algorithmic framework for variant selection from the perspective of preserving subsequent mapping accuracy. Consider a complete variation graph constructed from a set of given haplotypes. Any substring of a haplotype has a corresponding path in the complete variation graph. Not including some variants will introduce errors in the corresponding paths. If the number of such errors is matched with the error tolerance built into sequence-to-graph mapping algorithms, the same identical paths can still be found. We make the following contributions:

- We develop a novel mathematical framework for variant selection subject to preserving paths of length α while allowing at most δ differences. We separately consider the problems of minimizing the number of positions at which variants are retained, and minimizing the total number of variants selected.
- We show that both problems are optimally solvable in polynomial time when only SNPs are considered and the goal is to preserve all paths of length α found in the complete variation graph.
- These problems become challenging when indels are brought into play. We present efficient heuristics that guarantee preserving paths of length α while allowing at most δ edits, but do not guarantee optimal reduction in graph size.
- We empirically evaluate run-time performance and reduction in variation graph sizes achieved by the multiple algorithms that are proposed in this paper. For testing, we utilize human chromosome sequences, SNP variants from the 1000 Genomes Project (Consortium *et al.*, 2015), and indel structural variants (SVs) (i.e., indels of size $\geq$50 bp) from fifteen diverse humans (Audano *et al.*, 2019). When chromosome 1 variation graph is built using SNP variants, and parameters amenable to short reads ($\alpha = 150$ and $\delta = 8$) are used, the reduced graph excludes 94.44% SNPs. With parameters adjusted for long reads ($\alpha = 10$ kbp and $\delta = 1000$), 99.99% SNPs are excluded. When SVs are considered, the ($\alpha = 150$ bp, $\delta = 8$) and ($\alpha = 10$ kbp, $\delta = 1000$) settings result in excluding 0% and 73% SVs respectively.
- Finally, we consider the complexity of haplotype-aware versions of the above problems where the goal is to only preserve paths of length α actually found in the input haplotypes (i.e., not recombinant paths), and prove that they are $\mathcal{NP}$-hard even for $\delta = 1$.

2 Proposed framework

Let $R_1, R_2, \ldots, R_m$ be m input reference haplotype sequences. To be consistent with current literature, we assume one of these (say R_1) is a special reference and the other haplotypes are described as variations from it. A (complete) variation graph of these sequences is represented using an edge-labeled directed multigraph $G(V, E, \sigma)$ as follows. The graph consists of haplotype R_1 as a linear backbone, augmented with the set of variants present in $R_2, R_3, \ldots, R_m$, assumed to be known *a priori*. Each variant represents a deviating base from R_1 (SNP) or an insertion/deletion (can be multiple bases). The function $\sigma : E \rightarrow \Sigma \cup \{\epsilon\}$ specifies edge labels, where Σ denotes the alphabet and ϵ denotes the empty character. The haplotype R_1 is represented in G as a directed chain with character labeled edges such that the chain spells the sequence R_1. This chain will have $|R_1| + 1$ ordered vertices $v_0, v_1, \ldots, v_{|R_1|}$. These vertices serve as a convenient *coordinate axis* for the variation graph. Each SNP variant is an additional labeled edge between vertices at two adjacent coordinates. A deletion variant is an edge labeled ϵ between a pair of vertices, whose coordinates are separated by the deletion length. An insertion variant is represented as a chain of one or more labeled edges that starts and ends at the same vertex. In this setup, the total number of variants at coordinate i ($0 \leq i \leq |R_1|$) equals out-degree of the vertex v_i minus one. See Figure 1 for an illustration.

Any path in graph G with α non-empty edges spells a string of length α. We place the restriction that a path is allowed to visit a vertex at most twice. This restriction avoids traversal of more than one insertion variant at the same coordinate. Note that any recombination of variants that occur at different positions is allowed. Thus, the graph contains paths corresponding to each haplotype and any substrings thereof, but also numerous additional paths (genotypes) that are not present in any haplotype. It is unknown whether such a recombinant genotype exists in the population or not. Restricting paths to only those that belong to at least one input haplotype can also be useful, and will be considered separately (Section 4).

We seek to compute a reduced variation graph $G'(V', E', \sigma')$, where $V' \subseteq V$, $E' \subseteq E$, and for all $e \in E'$, $\sigma'(e) = \sigma(e)$. The reduced graph G' corresponds to removing some variants in graph G. Our goal is to reduce graph $G(V, E, \sigma)$ to the maximum extent possible while ensuring that any α-long string corresponding to a path in G can be mapped to the same starting vertex (coordinate) in G' without exceeding a user-specified error threshold δ. In practice, α should be a function of read lengths whereas δ is determined based on sequencing errors and error-tolerance of read-to-graph mapping algorithms.

We formulate four versions of the problem based on what types of variants are allowed and the reduction objective. First consider the case where all variants are SNPs.

Definition 1. *Graph G' is said to be $(\alpha, \delta)_h$-compatible if all α-long strings in graph G can be mapped to their corresponding paths in graph G' with Hamming distance $\leq \delta$.*

Problem 1. *Compute an $(\alpha, \delta)_h$-compatible reduced variation graph G' with minimum number of coordinates containing one or more variants.*

Problem 2. *Compute an $(\alpha, \delta)_h$-compatible reduced variation graph G' with minimum number of variants.*

In Problem 1, we seek to 'linearize' the graph, whereas in Problem 2, we intend to remove as many variants as possible. A user can choose either version based on downstream analysis. For the next two problem versions, suppose the variant set also contains indels.

Definition 2. *Graph G' is said to be $(\alpha, \delta)_e$-compatible if all α-long strings in graph G can be mapped to their corresponding paths in graph G' with edit distance $\leq \delta$.*

Problem 3. *Compute an $(\alpha, \delta)_e$-compatible reduced variation graph G' with minimum number of coordinates containing one or more variants.*

Problem 4. *Compute an $(\alpha, \delta)_e$-compatible reduced variation graph G' with minimum number of variants.*

In Problems 1 and 2 that consider only SNP variants, α-long paths will begin at a vertex along the coordinate axis as there are no other vertices introduced in the graph. In Problems 3 and 4 however, a path can also begin at other vertices due to insertion variants. In this case, we assume an α-long string that maps to G must also be mappable starting from the corresponding vertex in G' if that insertion variant is preserved. If the variant is not preserved, it must be mappable to the closest vertex along the coordinate axis.

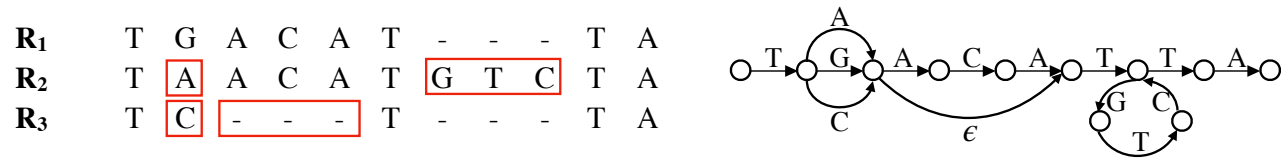


Fig. 1. An example to illustrate construction of variation graph from three haplotype sequences.

3 Proposed algorithms

3.1 Results for variation graphs with SNPs

3.1.1 Greedy algorithm for Problem 1

As the goal is to minimize the number of coordinates (positions along the special reference R_1) at which variants are included in G', we either fully *remove* or fully *retain* all the variants at each coordinate with variants. When removing variants at a coordinate, its outgoing edge label is chosen to be the base from R_1. However, the $(\alpha, \delta)_h$-compatibility is sustained even if the base is chosen from a different haplotype, or any arbitrary character in Σ.

A path of length α naturally corresponds to a line segment of length α starting at an integer coordinate. Observe that in any α-long segment, we cannot remove variants at $> \delta$ coordinates without violating the $(\alpha, \delta)_h$-compatibility of reduced graph G' (Figure 2(a)). A variant coordinate i is contained in α segments of length α each, whose starting positions are in $[i - \alpha + 1, i]$. For each variant position, we associate two events with coordinates $start_i = min\{0, i - \alpha + 1\}$ and $end_i = i$ respectively. Assuming that the n SNP coordinates are given as sorted array, the corresponding $2n$ events can be sorted in $O(n)$ time. When two events have equal coordinates, the $start$ event type should be placed earlier than the end event type in the sorted order.

Our greedy algorithm works as follows. Begin by placing an α-long segment at position 0, and remove variants in the leftmost δ variant positions and retain the rest (if any). Keep a count of the number of positions within the current segment at which variants are removed. Iteratively consider each event in the sorted order. If the event is of type $start_i$ and the count is less than δ, the variants at position i are removed and the count is incremented by one. If the event is of type $start_i$ but the count is equal to δ, the variants at position i are retained. If the event is of type end_i and the variants at i were previously removed, the count is decremented by 1. As can be seen, the algorithm maintains $(\alpha, \delta)_h$-compatibility and runs in $O(n)$ time.

Proof of optimality: Suppose the greedy algorithm retains variants at coordinates $g_1, g_2, \ldots, g_p$ in ascending order. Let $o_1, o_2, \ldots, o_q$ be the ordered variant coordinates retained by an optimal solution. Let k be the first position where the solutions differ, i.e., $g_j = o_j$ for $j < k$ and $g_k \neq o_k$. Due to our greedy strategy, $o_k < g_k$. Though o_k was chosen by the optimal algorithm, $(\alpha, \delta)_h$-compatibility is not violated until start event for g_k is reached. For any path starting at a later coordinate, retaining variants at g_k offers the same benefit as retaining at o_k. Thus, replacing o_k with g_k will maintain optimality and $(\alpha, \delta)_h$-compatibility. Hence, the greedy solution is also optimal.

Lemma 1. *The above greedy algorithm solves Problem 1 in $O(n)$ time.*

3.1.2 A linear programming solution to Problem 2

Here, we seek to minimize the total number of variants retained. Interestingly, we can show that optimal solutions still retain or remove all variants at a coordinate.

Lemma 2. *An optimal solution to Problem 2 either retains or removes all variants at a coordinate.*

Proof. By contradiction. Suppose there exists an optimal reduced graph G' with partially removed variants at coordinate i. Coordinate i already induces an error in some α-long paths in G that contain the coordinate. Accordingly, removal of all variants at coordinate i can be tolerated by all α-long paths containing that coordinate, further implying that graph G' must be sub-optimal. □

Suppose G' contains no variants at coordinate i, then this choice reduces the count of variants by $out(v_i) - 1$, where $out(v_i)$ is the out-degree of vertex v_i. As can be seen, Problem 2 is harder than Problem 1 because the number of variants at different coordinates can be different, leading to variable gains. We address this problem by using an Integer Linear Programming (ILP) system that is polynomially-solvable using LP relaxation.

Let $p_1, p_2, \ldots, p_n$ be the n variant coordinates in G in ascending order. Let X be an $n \times 1$ boolean column vector where $X[i] = 1$ iff variants are removed at coordinate p_i in creating G'. Let C be another $n \times 1$ column vector where $C[i] = out(v_{p_i}) - 1$, i.e., the reduction achieved in variant count by removing variants at p_i. The goal is to maximize $C^T X$. Next, we specify constraints to ensure $(\alpha, \delta)_h$-compatibility of graph G', by not allowing removal of variants at $> \delta$ coordinates in any α-long segment. Similar to the observation made while addressing Problem 1, it suffices to check this constraint only in the subset of α-sized segments that end at the n variants. Accordingly, let A be a boolean $n \times n$ matrix such that $A[i][j] = 1$ iff coordinate p_j is within the α-sized segment range $(p_i - \alpha, p_i]$ of coordinate p_i. Then, ILP constraints required to ensure $(\alpha, \delta)_h$-compatibility of G' can be specified as $A \cdot X \leq B$, where B is an $n \times 1$ column vector with each value $= \delta$. We also need to ensure that the $X[i]$'s are boolean. This can be achieved by expanding A to a $2n \times n$ matrix with the bottom n rows being the $n \times n$ identity matrix, and similarly expanding B to a $2n \times 1$ vector with the bottom n entries set to 1. Now, maximizing $C^T X$ while satisfying $A \cdot X \leq B$ leads to an optimal reduced graph G' that is $(\alpha, \delta)_h$-compatible.

Run-time complexity: Matrix A exhibits a special structure that guarantees integral optimal LP solutions. Observe that A is a 0-1 matrix, and the 1's appear consecutively in each row of A which makes it an interval matrix (Fulkerson and Gross, 1965). As a result, the above ILP can be solved in polynomial time by solving the corresponding LP, which has $O(n^\omega)$ run-time complexity where ω is the exponent of matrix multiplication (van den Brand, 2020).

Lemma 3. *The above LP-based algorithm solves Problem 2 in $O(n^\omega)$ time.*

3.2 Results for variation graphs with SNPs and indels

Variation graphs with indels introduce additional complexity. When considering only SNPs, we benefited from the fact that end vertices of any α-long paths will be located on the coordinate axis. In addition, right end of a path was a fixed distance away from its left along the coordinate axis. When indels are permitted, these properties are no longer true, making Problems 3 and 4 more challenging. We present two heuristic solutions, each of which can be used to solve either problem.

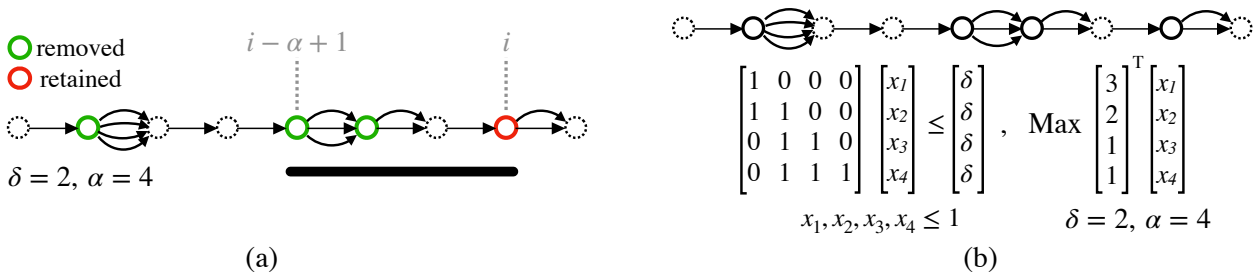


Fig. 2. Execution of the Greedy and LP algorithms on an example variation graph containing SNPs only. Edge labels are not shown as they do not affect the execution of either algorithm.

3.2.1 A greedy algorithm

We first propose a 'conservative' greedy heuristic which guarantees an $(\alpha, \delta)_e$-compatible reduced graph that is not necessarily optimal in terms of the desired reduction objectives. We choose to either retain or remove all variants at a coordinate vertex (vertex along R_1). Suppose a coordinate vertex v has all three types of variants, i.e., insertions, deletions and SNPs. We evaluate an upper bound of edit distance against any overlapping α-long path if we choose to drop all variants at v. Let $\Delta_{ins}, \Delta_{del}$ be the longest insertion and deletion variants at vertex v respectively. Dropping all variants at v can contribute an edit distance of at most $\Delta_{ins} + \Delta_{del}$. In cases where only a subset of variant types are present, the bound can be adjusted easily. The following greedy algorithm is designed to select an appropriate set of coordinates where variants can be removed while ensuring that the graph remains $(\alpha, \delta)_e$-compatible.

As before, let $p_1, p_2, \ldots, p_n$ be the n variant coordinates in G in ascending order. Note that an α-long path in graph G can span $> \alpha$ range along the coordinate axis by using deletion edges. For a variant position p_i, consider the left-most position p_j such that v_{p_i} can be reached from v_{p_j} by using any path that uses $< \alpha$ labeled edges. The rationale for choosing p_j this way is that any α-long path which begins at a variant coordinate vertex prior to v_{p_j} cannot pass through vertex v_{p_i}. Such a window is pre-computed for each variant position, and we ensure that dropped variants within each window collectively contribute to edit distance $\leq \delta$. To achieve this, our greedy heuristic is to consider the variant positions from left to right. A variant position is removed if and only if the total sum of differences within its window remains $\leq \delta$. It is straightforward to prove that the resulting reduced graph remains $(\alpha, \delta)_e$-compatible.

Run-time complexity: Computing window lengths for each coordinate vertex is the most time-consuming step in the above algorithm because the remaining steps have linear complexity either in terms of count of variants or count of variant positions in graph G. For calculating window lengths in the above algorithm, we can ignore SNP and insertion variants from G, and consider only deletion variants. If y denotes the count of deletion variants, then the modified graph will have exactly $|R_1| + 1$ coordinate vertices and $|R_1| + y$ edges. Any vertex v_i $(i > 0)$ has exactly one incoming labeled edge (say, from vertex v_{i_1}) and ≥ 0 incoming unlabeled edges (say, from vertices $v_{i_2}, v_{i_3}, \ldots, v_{i_k}$). Let the function $f(v, x)$ indicate the left-most vertex that can be reached from v by using a path that uses $< x$ labeled edges. Then, $f(v_i, \alpha)$ equals the left-most vertex among $f(v_{i_1}, \alpha-1), f(v_{i_2}, \alpha), f(v_{i_3}, \alpha), \ldots, f(v_{i_k}, \alpha)$. One way to compute this recursion is to compute a vector of values $f(v_i, x)\ \forall x \in [1, \alpha]$ for each vertex along the coordinate axis going from left to right. This procedure requires $O(\alpha \cdot (|R_1| + y))$ time. In practice, $y \ll |R_1|$, so this procedure effectively requires $O(\alpha \cdot |R_1|)$ time.

3.2.2 An ILP-based algorithm

Alternatively, we can further improve the above heuristic by using ILP. This can be achieved by formulating the edit distance constraints discussed above for each window as a set of ILP constraints. Similar to our LP-based algorithm for Problem 2, A is an $n \times n$ matrix, where row i contains non-zero values for those variants that are within the pre-computed window of variant i. For instance, if coordinate p_j is within the pre-computed window span of coordinate p_i $(j \leq i)$, then $A[i][j]$ is set to the estimated upper bound of differences induced by removing all variants at coordinate p_j as discussed before. Variable X is an $n \times 1$ boolean column vector, where $X[i] = 1$ iff variants at coordinate p_i are removed. Then, ILP constraints required to ensure $(\alpha, \delta)_e$-compatibility can be specified as $A \cdot X \leq B$, where B is a column vector with each value $= \delta$. Define C to be an $n \times 1$ column vector. While addressing Problem 3, set $C[i]$'s as 1, and for Problem 4, set $C[i] = out(v_{p_i}) - 1$, i.e., the count of variants at coordinate p_i. In both cases, the ILP objective is set to maximize $C^T X$. These ILP formulations have higher run-time complexity when compared to the greedy solution, but are guaranteed to provide at least as good and possibly superior reduction for both Problems 3 and 4. Neither algorithm guarantees optimality.

4 Haplotype-aware variant selection

In the previous problem versions, we considered all α-long paths in graph G. Here we address the important special case where paths are *restricted* to correspond to strings observed in haplotypes $R_1, R_2, \ldots, R_m$. Due to this restriction, fewer α-long strings are checked for mappability. As a result, solutions to the previous problems are sub-optimal for this case because further reduction may be possible. We start by making the simplifying assumption that the input haplotypes contain only SNPs, and have equal length. Outgoing edge label(s) from a coordinate vertex in a reduced graph can come from any of the m haplotypes. Graph G' is said to be $(\alpha, \delta)^r_h$-compatible if all α-long *restricted* paths in G map to G' with Hamming distance $\leq \delta$ between the corresponding strings. In this scenario, consider the following problems:

Problem 5. *Compute an $(\alpha, \delta)^r_h$-compatible reduced variation graph G' with minimum number of coordinates containing one or more variants.*

Problem 6. *Compute an $(\alpha, \delta)^r_h$-compatible reduced variation graph G' with minimum number of variants.*

We prove that solving the above problems is $\mathcal{NP}$-hard. We give two reductions for Problem 5. The first is a general reduction whereas the second proves hardness for even $\delta = 1$. These reductions trivially generalize to Problem 6. Consider the following decision version of Problem 5. Does there exist an $(\alpha, \delta)^r_h$-compatible simplified graph G' with $\leq k$ coordinates containing one or more variants?

Lemma 4. *The decision version of Problem 5 is $\mathcal{NP}$-complete.*

Proof. Clearly, the problem is in $\mathcal{NP}$. Recall the decision version of the closest string problem (CSP) (Lanctot *et al.*, 2003). Given a set $\mathcal{S}$ of strings each of length l and a parameter d, CSP checks existence of a string that is within Hamming distance of d to each of the given strings. CSP is known to be $\mathcal{NP}$-complete. CSP exhibits a trivial reduction to Problem 5: Assume the collection of reference haplotypes to be $\mathcal{S}$. The following statements are equivalent: (i) there exists a string with Hamming distance $\leq d$ to each of the given strings in $\mathcal{S}$, (ii) there exists an $(l, d)^r_h$-compatible graph G' with no coordinate containing one or more variants. As a result, Problem 5, which is stated for an arbitrary value of k, is $\mathcal{NP}$-complete. □

CSP is known to be $\mathcal{NP}$-complete even for a binary alphabet, thus also making Problem 5 $\mathcal{NP}$-complete for a binary alphabet. However, CSP is fixed-parameter tractable relative to parameter d (Gramm *et al.*, 2003). Consequently, the above claim does not resolve the complexity of Problem 5 for a constant δ. For practical applications, δ is expected to be small. We address this in the following lemma.

Lemma 5. *The decision version of Problem 5 is $\mathcal{NP}$-complete even if $\delta = 1$.*

Proof. Recall the decision version of the maximum independent set (MIS) problem. Given an undirected graph, the MIS problem asks for a set of $\geq k$ vertices no two of which are adjacent. In an MIS graph instance $G_m(V_m, E_m)$, let $u_0, u_1, \ldots, u_{|V_m|-1}$ be the vertices and $e_0, e_1, \ldots, e_{|E_m|-1}$ be the edges. We translate this into a multigraph instance of Problem 5 as follows. Let $\Sigma = \{\mathrm{A}, \mathrm{C}\}$. Define haplotype reference sequences $R_0, R_1, \ldots, R_{|V_m|+|E_m|}$ each of length $|V_m|$. The first $|E_m|$ sequences are defined using the MIS graph instance G_m while the rest are auxiliary:

$$R_i[j] = \mathrm{C} \text{ if } e_i \text{ connects } u_j;\ R_i[j] = \mathrm{A} \text{ otherwise.} \quad (0 \leq i < |E_m|),$$
$$R_i = \mathrm{A}^{(i-|E_m|)} \cdot \mathrm{C} \cdot \mathrm{A}^{(|E_m|+|V_m|-i-1)} \ (|E_m| \leq i < |E_m|+|V_m|),$$
$$R_i = \mathrm{A}^{|V_m|} \quad (i = |E_m| + |V_m|)$$

Observe that edge-labeled variation graph G built by using the above sequences has a coordinate axis with $|V_m| + 1$ vertices (Figure 3). Each coordinate $\in [0, |V_m|)$ has two SNPs 'A' and 'C'. *Claim: There exists a k-sized independent set in $G_m(V_m, E_m)$ if and only if there exists a $(|V_m|, 1)^r_h$-compatible reduced variation with $|V_m| - k$ coordinates containing one or more variants.* Consider the forward direction. Suppose there are k vertices in an independent set $\mathcal{I}$. Build a reduced variation graph G' by removing 'C'-labeled outgoing edges from coordinates j $(\forall j)$ such that $u_j \in \mathcal{I}$. Note that G' is $(|V_m|, 1)^r_h$-compatible. Next consider the backward direction. Suppose $p_1, p_2, \ldots, p_k$ are the k coordinates in a compatible simplified graph G' where variants are removed. If $k = 1$, then finding an independent set $\mathcal{I}$ of size 1 is trivial. If $k > 1$, then we note that each of the k coordinates must have a single outgoing edge labeled with 'A' to ensure $(|V_m|, 1)^r_h$-compatibility with respect to the auxiliary reference sequences. It can be further deduced that $\{u_{p_1}, u_{p_2}, \ldots, u_{p_k}\}$ is an independent set of graph G_m. □

5 Experimental results

Hardware and software: We provide C++ implementations of all the algorithms presented in Section 3 (https://github.com/ac-gt/VF). Among these, the first two handle SNP-based variation graphs (*Greedy*$_s$ and *LP*$_s$), and the remaining (*Greedy*$_i$, *ILP*$_i^v$ and *ILP*$_i^p$) are designed for a generic variation graph containing both SNPs and indels. Our ILP algorithm (Section 3.2.2) supports two different objective functions, the first minimizes count of variants, and the second minimizes variant-containing positions. Accordingly, their naming, i.e., *ILP*$_i^v$ and *ILP*$_i^p$ differentiates the two versions respectively.

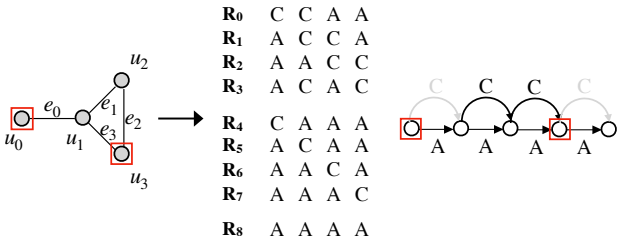


Fig. 3. Illustration of reduction used to prove Lemma 5. Vertices selected as independent set are highlighted in red (left). Accordingly, we can find an equivalent reduced variation graph where variants from two vertices are removed (removed edges are highlighted in gray).

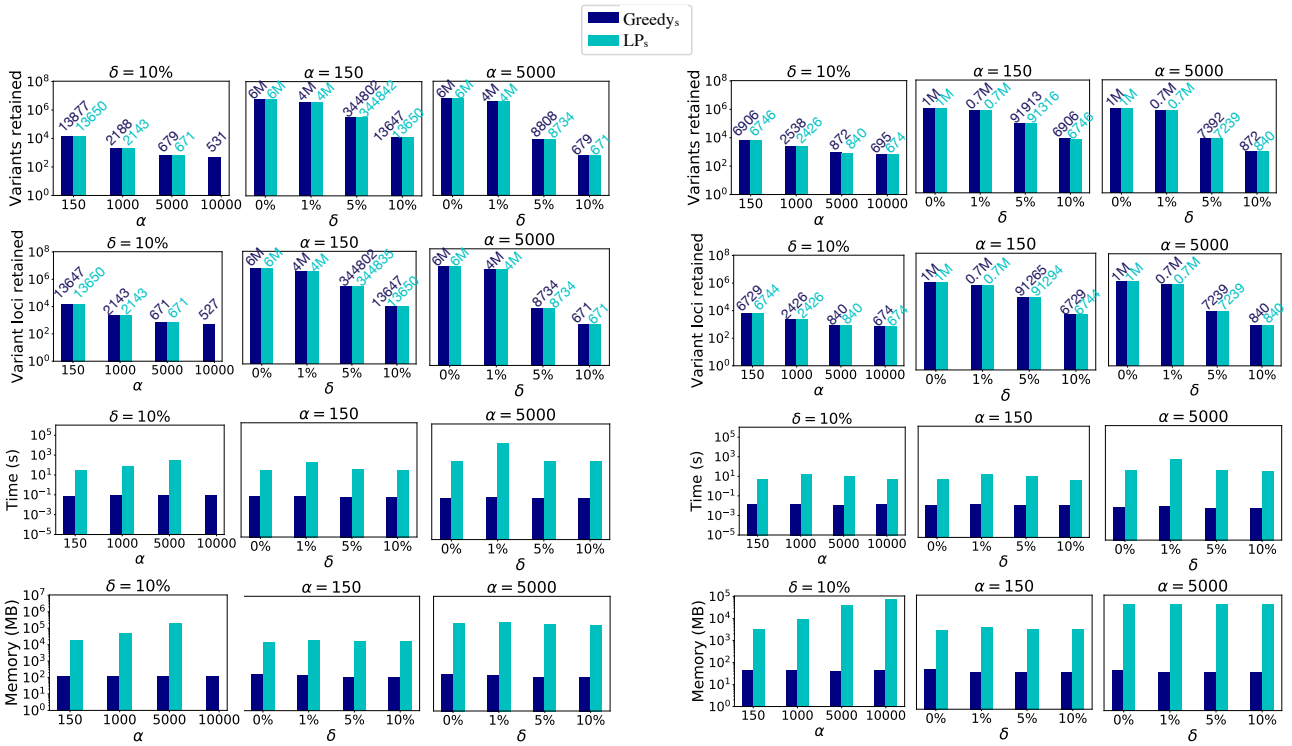
Using human variation graphs (Table 1), we assess the graph size reduction achieved by the various algorithms, and also evaluate their runtime performance and scalability. The *LP*$_s$, *ILP*$_i^v$ and *ILP*$_i^p$ algorithms make use of Gurobi 9.1.0 solver for LP optimization. All the algorithms were tested on dual Intel Xeon Gold 6226 CPUs (2.70 GHz) processors equipped with 2×12 physical cores and 384 GB RAM. Among the implemented algorithms, only the *LP*$_s$, *ILP*$_i^v$ and *ILP*$_i^p$ take advantage of multiple cores via Gurobi, whereas the remaining two are sequential.

Variation graph construction: We tested our algorithms using variation graphs associated with human chromosome 1 (249 Mbp) and chromosome 22 (51 Mbp) respectively. For both chromosomes, we built two types of variation graphs, corresponding to SNPs and indel structural variants (SVs), respectively. Rather than using small-sized indels, we intentionally chose to experiment using indel SVs ($\geq$ 50bp). This is useful to contrast output quality while exploring variant types from point mutations (SNPs) to larger variants. Exclusion of SVs is naturally expected to introduce more differences in α-sized paths, and therefore SVs test the limits of our algorithms. SNP variants were downloaded from the 1000 Genomes Project Phase 3 (Consortium *et al.*, 2015), and indel SVs were downloaded from a recent long-read based SV survey of fifteen diverse human genomes (Audano *et al.*, 2019). We used vcftools (Danecek *et al.*, 2011) to pre-process and extract SNPs from the 1000 Genomes Project variant files. Similarly, SVs other than insertions or deletions were filtered out from the SV files. Summary statistics of these variants, and graphs built using them, are listed in Table 1.

α and δ parameters: We tested our algorithms using α values of 150 bp, 1 kbp, 5 kbp, and 10 kbp. The first is useful for Illumina reads, whereas the latter are useful for different protocols available for long read sequencing (e.g., DNA or RNA sequencing using either PacBio or ONT). For each α value, we experimented with δ values that are 1%, 5% and 10%

Table 1. Variation graphs used for testing the proposed variant selection algorithms.

Graph label	Chr	Type of variants	No. of variants	No. of variant containing loci
vg_chr1_SNP	1	SNPs	6,234,054	6,215,039
vg_chr22_SNP	22	SNPs	1,063,618	1,059,517
vg_chr1_SV	1	Indel SVs	6,525	6,369
vg_chr22_SV	22	Indel SVs	2,056	1,996



(a) chromosome 1 (b) chromosome 22

Fig. 4. Empirical evaluation of Greedy$_s$ and LP$_s$ algorithms using two human variation graphs vg_chr1_SNP and vg_chr22_SNP containing SNPs. These plots demonstrate reduction achieved in graph sizes while varying α and δ parameters. Size of the complete variation graph ($\delta = 0\%$) is included for comparison. Numbers on top of bars present actual data, useful for comparison when both Greedy$_s$ and LP$_s$ achieve close results. Result of LP$_s$ algorithm is missing for $\alpha = 10,000$ (left-most column) because Gurobi LP solver crashed due to insufficient memory. Y-axes are log-scaled in all the above plots.

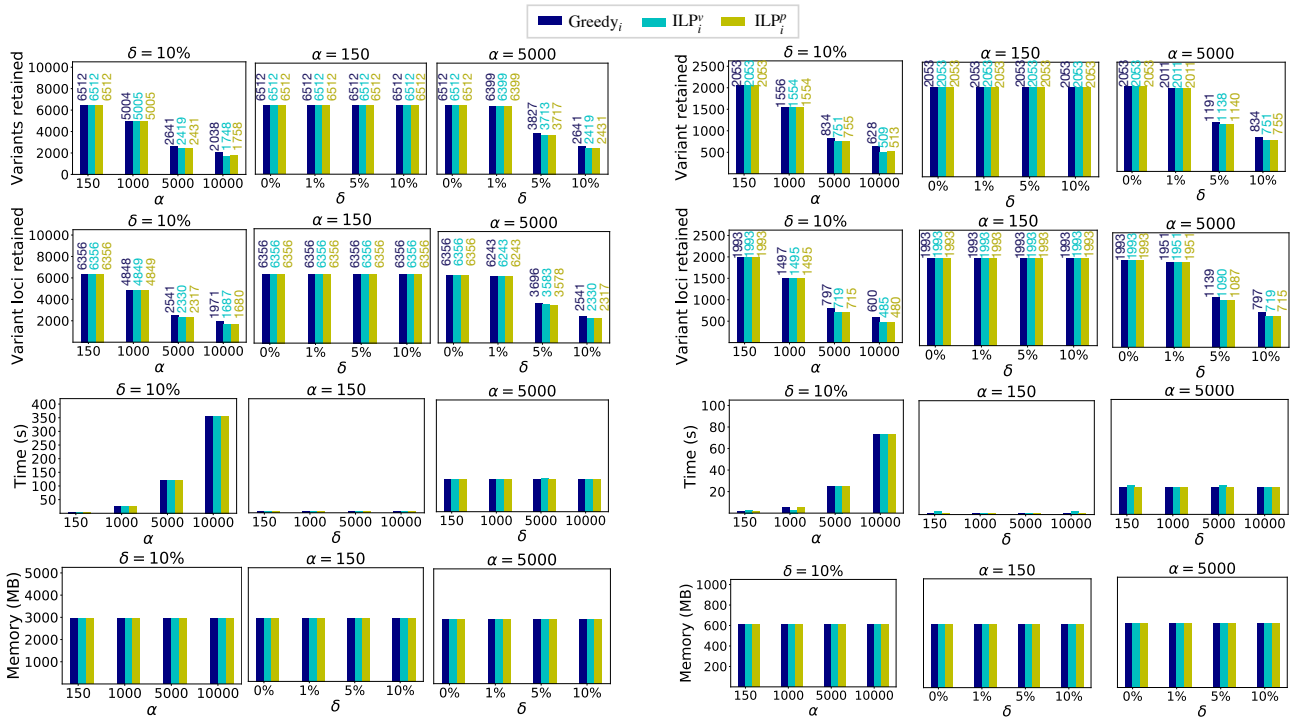
of α. Here 1% corresponds to low error-tolerance of a mapping algorithm, and 10% corresponds to significant tolerance.

Performance of *Greedy$_s$* and *LP$_s$* algorithms: These algorithms were tested using vg_chr1_SNP and vg_chr22_SNP (Figure 4) graphs. For both the algorithms, we report four statistics: (i) count of variants retained, (ii) count of variant-containing loci retained, (iii) run-time, and (iv) peak memory-usage. *LP$_s$* and *Greedy$_s$* algorithms are guaranteed to return optimal graphs in terms of the objectives (i) and (ii) respectively. The results in Figure 4 suggest that the two algorithms perform almost equally well in terms of both objectives for all tested combinations of (α, δ) values. Increasing α value while keeping δ as a constant fraction of α naturally corresponds to fewer SNPs retained. The same is true when δ is increased while keeping α fixed. These results corroborate the fact that longer reads and higher sensitivity of mapping algorithms result in retention of fewer variants in a variation graph. For instance with ($\alpha = 10$ kbp, $\delta = 1000$), the *Greedy$_s$* algorithm retained only 531 (0.009%) out of 6,234,046 SNPs using graph vg_chr1_SNP. After this run, average distance between two adjacent loci containing SNPs increased from 39 to 225,963. This suggests that variation selection algorithms can potentially yield long stretches of variant-free regions in graph, where the usual read-to-sequence mapping algorithms can also operate.

If run-time is considered, the *Greedy$_s$* algorithm runs significantly faster than *LP$_s$*, which was also reflected by our time complexity analysis in Section 3.1. Using graph vg_chr1_SNP, *LP$_s$* algorithm ran out of memory for $\alpha = 10$ kbp due to increased size of the matrix, i.e., count of non-zeros in the matrix which specifies the LP constraints. Taken together, *Greedy$_s$* algorithm suffices for most practical purposes because it is fast and optimal in terms of its objective to minimize count of variant-containing loci retained. *Greedy$_s$* algorithm does not account for the number of variants at a locus while deciding its fate, yet it achieves near-optimal reduction in terms of minimizing the count of variants. This is likely because most human SNPs are biallelic.

Performance of *Greedy$_i$* and ILP-based algorithms: We tested our *Greedy$_i$*, *ILP$_i^v$* and *ILP$_i^p$* heuristics using vg_chr1_SV and vg_chr22_SV (Figure 5) graphs. In contrast to SNPs which are single-base mutations, the sizes of SV indels in chromosome 1 computed by Audano *et al.* (2019) range from 50 bp to 33 kbp, with mean length 0.5 kbp. As a result, it is natural to expect that the fraction of variants retained will be much higher compared to SNPs. The ILP-based heuristics are guaranteed to achieve superior results than the greedy heuristic, i.e., *ILP$_i^v$* heuristic is expected to retain the smallest count of variants among the three, and similarly *ILP$_i^p$* heuristic will retain the smallest count of variant-containing loci. For instance, with long-read compatible settings ($\alpha = 10$ kbp, $\delta = 1000$), the *ILP$_i^v$*, *ILP$_i^p$* and *Greedy$_i$* heuristics retained 26.8%, 27.0% and 31.3% SVs respectively in graph vg_chr1_SV. Similarly, 24.8%, 25.0% and 30.6% SVs were retained in graph vg_chr22_SV. However, with short-read compatible settings ($\alpha = 150$ bp, $\delta = 8$), all SVs were retained by all three heuristics, as expected. These results are not necessarily optimal, but we do not expect them to deviate significantly from optimal numbers. The rationale is that not only SVs are bigger in size but also SV loci are known to be clustered in several known hot-spots of the human genome, e.g., within the last 5 Mbp of both chromosome arms (Audano *et al.*, 2019).

In terms of count, SVs occur much less frequently as compared to SNPs or indels. As a result, run-time of all the three heuristics was dominated by their first step of computing the constraints required to ensure $(\alpha, \delta)_e$ compatibility, which is common in all of them. As shown



(a) chromosome 1

(b) chromosome 22

Fig. 5. Empirical evaluation of Greedy$_i$, ILP$_i^v$ and ILP$_i^p$ algorithms using two human variation graphs vg_chr1_SV and vg_chr1_SV containing insertion and deletion structural variants. These plots demonstrate reduction achieved in graph sizes while varying α and δ parameters. Numbers on top of bars present actual data, useful for comparison when the three algorithms achieve close results.

in Section 3.2, this step requires time proportional to α as well as the length of the reference sequence. Accordingly, we observe that running time is comparable among all three heuristics, appears to be independent of δ, scales roughly linearly with α, and time spent is higher using graph vg_chr1_SV than vg_chr22_SV. With the largest $\alpha = 10$ kbp value, all three algorithms require about six minutes and one minute to process the two graphs, respectively.

Impact on sequence-to-graph mappers: We conducted a preliminary evaluation to assess the impact of variant selection on read-to-graph mapping. For this experiment, we considered the reduced graph obtained by ILP_i^v using vg_chr1_SV graph as input. As discussed previously, the ILP_i^v heuristic retained $1,748$ of the $6,512$ SVs using $\alpha = 10$ kbp and $\delta = 1000$ parameters. We built two variation graphs using VG (v1.29.0) tool corresponding to the complete set of SVs and the reduced set of SVs respectively. The graph statistics (e.g., vertex degree distribution) were validated to ensure the presence of SVs in the respective graphs. Next, we simulated $10,000$ long reads, each of length 10 kbp from randomly chosen paths of the complete variation graph with error-rate 5% using VG's read simulation feature. Subsequently, we made use of GraphAligner (v1.0.11) to map these reads to both variation graphs.

We observed the following. First, all $10,000$ reads were successfully mapped by GraphAligner to both the graphs. Second, each read was mapped only using primary alignments, and there were no secondary alignments reported. This indicates that there was no mapping ambiguity while using the reduced variation graph. Finally, the impact of missing variations was observed in the count of reads with split-read alignments, which increased from 2 to 209. Split read alignments are often used as a signature by variant callers to discover SVs (Rausch *et al.*, 2012). A direct comparison of mapping coordinates could not be done because VG used different vertex identifiers in the two graphs which have different topology. We note that GraphAligner required similar runtime and memory in both scenarios (about five minutes). This is likely because the graph is nearly linear, due to limited count of SVs $(6,512)$ that were available in the complete chromosome 1 graph. This result is preliminary, but motivates a deeper investigation into the impact of the proposed algorithms on various sequence-to-graph mapping algorithms while using a much larger catalog of variants as input. Variant selection tool FORGe (Pritt *et al.*, 2018) uses allelic frequency data as an input to its model. Currently, frequency assessment remains challenging in case of SVs due to the lack of appropriate tools as well as data (Mahmoud *et al.*, 2019). A direct comparison with FORGe could not be carried out due to these limitations.

6 Conclusions and open problems

We developed a novel mathematical framework for variant selection, and presented multiple algorithms and complexity results for various problems arising from this framework. Experimental results demonstrate substantial reduction in the resulting variation graph sizes, while guaranteeing bounds on the number of errors tolerated while doing so. Implementations of all the four algorithms that are proposed in this paper are available as open-source, and can be used by practitioners for downstream pan-genomic analysis. The path-length and error parameters (α, δ) can be tuned to match the choice of sequencing technology, mapping algorithms, and types of variants considered. We demonstrated that a high fraction of small-scale variants, but no large-scale variants can be left out during short-read mapping to variation graphs. On the other hand, almost all small-scale variants and a significant fraction of large-scale variants may be excluded prior to long-read-based analysis.

The proposed variant selection framework underpins a rich class of problems making it fertile ground for future research. (i) Optimal algorithms for the two problems associated with indel variants (Problems 3 and 4) are unknown. In fact, it is not known whether these two problems can be solved in polynomial time. (ii) While we were able to prove that the haplotype-aware versions of the problem are $\mathcal{NP}$-hard, efficient heuristics and approximation algorithms for these problem are yet to be developed. Haplotype-aware algorithmic extensions can result in further reduction of graph sizes because fewer paths need to be preserved. (iii) It may also be possible to further extend this framework, and add constraints similar to allele frequency thresholding, e.g., by asking a reduced graph which is allowed to violate error-bound for up to 1% of haplotypes at any position. (iv) Finally, this work only considered pre-dominant variant categories - SNPs, insertions and deletions; further research is needed to analyze other variant types such as duplications, inversions, and complex genomic rearrangements.

Funding

This work is supported in part by the National Science Foundation under CCF-1816027. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

Audano, P. A., Sulovari, A., Graves-Lindsay, T. A., Cantsilieris, S., Sorensen, M., Welch, A. E., Dougherty, M. L., Nelson, B. J., Shah, A., Dutcher, S. K., *et al.* (2019). Characterizing the major structural variant alleles of the human genome. *Cell*, **176**(3), 663–675.

Ballouz, S., Dobin, A., and Gillis, J. A. (2019). Is it time to change the reference genome? *Genome biology*, **20**(1), 1–9.

Chang, X., Eizenga, J., Novak, A. M., Sirén, J., and Paten, B. (2020). Distance indexing and seed clustering in sequence graphs. *Bioinformatics*, **36**(Supplement_1), i146–i153.

Computational Pan-Genomics Consortium (2018). Computational pan-genomics: status, promises and challenges. *Briefings in bioinformatics*, **19**(1), 118–135.

Consortium, . G. P. *et al.* (2015). A global reference for human genetic variation. *Nature*, **526**(7571), 68–74.

Danecek, P., Auton, A., Abecasis, G., Albers, C. A., Banks, E., DePristo, M. A., Handsaker, R. E., Lunter, G., Marth, G. T., Sherry, S. T., *et al.* (2011). The variant call format and vcftools. *Bioinformatics*, **27**(15), 2156–2158.

Danek, A., Deorowicz, S., and Grabowski, S. (2014). Indexes of large genome collections on a PC. *PloS one*, **9**(10), e109384.

Darby, C. A., Gaddipati, R., Schatz, M. C., and Langmead, B. (2020). Vargas: heuristic-free alignment for assessing linear and graph read aligners. *Bioinformatics*, **36**(12), 3712–3718.

Dilthey, A., Cox, C., Iqbal, Z., Nelson, M. R., and McVean, G. (2015). Improved genome inference in the MHC using a population reference graph. *Nature genetics*, **47**(6), 682–688.

Eggertsson, H. P., Jonsson, H., Kristmundsdottir, S., Hjartarson, E., Kehr, B., Masson, G., Zink, F., Hjorleifsson, K. E., Jonasdottir, A., Jonasdottir, A., *et al.* (2017). Graphtyper enables population-scale genotyping using pangenome graphs. *Nature genetics*, **49**(11), 1654.

Eizenga, J. M., Novak, A. M., Sibbesen, J. A., Heumos, S., Ghaffaari, A., Hickey, G., Chang, X., Seaman, J. D., Rounthwaite, R., Ebler, J., *et al.* (2020). Pangenome graphs. *Annual Review of Genomics and Human Genetics*, **21**.

Fulkerson, D. and Gross, O. (1965). Incidence matrices and interval graphs. *Pacific journal of mathematics*, **15**(3), 835–855.

Garrison, E., Sirén, J., Novak, A. M., Hickey, G., Eizenga, J. M., Dawson, E. T., Jones, W., Garg, S., Markello, C., Lin, M. F., *et al.* (2018). Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nature biotechnology*, **36**(9), 875–879.

Ghaffaari, A. and Marschall, T. (2019). Fully-sensitive seed finding in sequence graphs using a hybrid index. *Bioinformatics*, **35**(14), i81–i89.

Gramm, J., Niedermeier, R., Rossmanith, P., *et al.* (2003). Fixed-parameter algorithms for closest string and related problems. *Algorithmica*, **37**(1), 25–42.

Holley, G., Wittler, R., and Stoye, J. (2016). Bloom filter trie: an alignment-free and reference-free data structure for pan-genome storage. *Algorithms for Molecular Biology*, **11**(1), 1–9.

Iqbal, Z., Caccamo, M., Turner, I., Flicek, P., and McVean, G. (2012). De novo assembly and genotyping of variants using colored de bruijn graphs. *Nature genetics*, **44**(2), 226–232.

Ivanov, P., Bichsel, B., Mustafa, H., Kahles, A., Rätsch, G., and Vechev, M. (2020). Astarix: Fast and optimal sequence-to-graph alignment. In *International Conference on Research in Computational Molecular Biology*, pages 104–119. Springer.

Jain, C., Misra, S., Zhang, H., Dilthey, A., and Aluru, S. (2019a). Accelerating sequence alignment to graphs. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 451–461. IEEE.

Jain, C., Zhang, H., Dilthey, A., and Aluru, S. (2019b). Validating Paired-End Read Alignments in Sequence Graphs. In *19th International Workshop on Algorithms in Bioinformatics (WABI 2019)*, volume 143 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:13.

Jain, C., Zhang, H., Gao, Y., and Aluru, S. (2020). On the complexity of sequence-to-graph alignment. *Journal of Computational Biology*, **27**(4), 640–654.

Kim, D., Paggi, J., and Salzberg, S. L. (2018). Hisat-genotype: Next generation genomic analysis platform on a personal computer. *BioRxiv*, page 266197.

Kuhnle, A., Mun, T., Boucher, C., Gagie, T., Langmead, B., and Manzini, G. (2020). Efficient construction of a complete index for pan-genomics read alignment. *Journal of Computational Biology*, **27**(4), 500–513.

Kuosmanen, A., Paavilainen, T., Gagie, T., Chikhi, R., Tomescu, A., and Mäkinen, V. (2018). Using minimum path cover to boost dynamic programming on DAGs: co-linear chaining extended. In *International Conference on Research in Computational Molecular Biology*, pages 105–121. Springer.

Lanctot, J. K., Li, M., Ma, B., Wang, S., and Zhang, L. (2003). Distinguishing string selection problems. *Information and Computation*, **185**(1), 41–55.

Li, H., Feng, X., and Chu, C. (2020). The design and construction of reference pangenome graphs with minigraph. *Genome Biology*, **21**(1), 1–19.

Liu, B., Guo, H., Brudno, M., and Wang, Y. (2016). debga: read alignment with de bruijn graph-based seed and extension. *Bioinformatics*, **32**(21), 3224–3232.

Maciuca, S., del Ojo Elias, C., McVean, G., and Iqbal, Z. (2016). A natural encoding of genetic variation in a burrows-wheeler transform to enable mapping and genome inference. In *International Workshop on Algorithms in Bioinformatics*, pages 222–233. Springer.

Mahmoud, M., Gobet, N., Cruz-Dávalos, D. I., Mounier, N., Dessimoz, C., and Sedlazeck, F. J. (2019). Structural variant calling: the long and the short of it. *Genome biology*, **20**(1), 1–14.

Marcus, S., Lee, H., and Schatz, M. C. (2014). Splitmem: a graphical algorithm for pan-genome analysis with suffix skips. *Bioinformatics*, **30**(24), 3476–3483.

Mokveld, T., Linthorst, J., Al-Ars, Z., Holstege, H., and Reinders, M. (2020). Chop: Haplotype-aware path indexing in population graphs. *Genome Biology*, **21**(1), 1–16.

Paten, B., Novak, A. M., Eizenga, J. M., and Garrison, E. (2017). Genome graphs and the evolution of genome inference. *Genome research*, **27**(5), 665–676.

Pritt, J., Chen, N.-C., and Langmead, B. (2018). Forge: prioritizing variants for graph genomes. *Genome biology*, **19**(1), 1–16.

Rausch, T., Zichner, T., Schlattl, A., Stütz, A. M., Benes, V., and Korbel, J. O. (2012). Delly: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics*, **28**(18), i333–i339.

Rautiainen, M. and Marschall, T. (2020). Graphaligner: rapid and versatile sequence-to-graph alignment. *Genome Biology*, **21**(1), 1–28.

Schneeberger, K., Hagmann, J., Ossowski, S., Warthmann, N., Gesing, S., Kohlbacher, O., and Weigel, D. (2009). Simultaneous alignment of short reads against multiple genomes. *Genome biology*, **10**(9), 1–12.

Sirén, J., Välimäki, N., and Mäkinen, V. (2014). Indexing graphs for path queries with applications in genome research. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **11**(2), 375–388.

Sirén, J., Garrison, E., Novak, A. M., Paten, B., and Durbin, R. (2020). Haplotype-aware graph indexes. *Bioinformatics*, **36**(2), 400–407.

van den Brand, J. (2020). A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 259–278. SIAM.

Vijaya Satya, R., Zavaljevski, N., and Reifman, J. (2012). A new strategy to reduce allelic bias in RNA-seq readmapping. *Nucleic acids research*, **40**(16), e127–e127.