Haplotype-aware Variant Selection for Genome Graphs

Neda Tavakoli
School of Computational Science and Engineering, Georgia Institute of Technology
Atlanta, Georgia, USA
neda.tavakoli1@gatech.edu

Daniel Gibney
School of Computational Science and Engineering, Georgia Institute of Technology
Atlanta, Georgia, USA
dgibney3@gatech.edu

Srinivas Aluru
School of Computational Science and Engineering, Georgia Institute of Technology
Atlanta, Georgia, USA
aluru@cc.gatech.edu

ABSTRACT

Graph-based genome representations have proven to be a powerful tool in genomic analysis due to their ability to encode variations found in multiple haplotypes and capture population genetic diversity. Such graphs also unavoidably contain paths which switch between haplotypes (i.e., recombinant paths) and thus do not fully match any of the constituent haplotypes. The number of such recombinant paths increases combinatorially with path length and cause inefficiencies and false positives when mapping reads. In this paper, we study the problem of finding reduced haplotype-aware genome graphs that incorporate only a selected subset of variants, yet contain paths corresponding to all α-long substrings of the input haplotypes (i.e., non-recombinant paths) with at most δ mismatches. Solving this problem optimally, i.e., minimizing the number of variants selected, is previously known to be NP-hard [14]. Here, we first establish several inapproximability results regarding finding haplotype-aware reduced variation graphs of optimal size. We then present an integer linear programming (ILP) formulation for solving the problem, and experimentally demonstrate this is a computationally feasible approach for real-world problems and provides far superior reduction compared to prior approaches.

KEYWORDS

Variation graphs, variant selection, haplotype-aware, SNPs, ILP-based optimization

ACM Reference Format:
Neda Tavakoli, Daniel Gibney, and Srinivas Aluru. 2022. Haplotype-aware Variant Selection for Genome Graphs. In *13th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '22), August 7–10, 2022, Northbrook, IL, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3535508.3545556

1 INTRODUCTION

With the increasing pace of genome sequencing and applications that involve analyzing the genomes of thousands of individuals, there is growing realization that a linear reference genome is not well suited for representing genetic diversity. A single reference genome represents only a tiny fraction of human genetic variation. Individual human genomes can contain 3.5 – 4 million SNPs or indels and approximately 2,500 significant structural variations relative to the linear reference. This genetic diversity between individuals and sub-populations can cause reads to map incorrectly when a single linear reference is used. To alleviate such reference bias and improve mapping accuracy, graph-based references have become increasingly popular in sequencing analysis. Graph-based references, or equivalently genome graphs built for a set of reference genomes, can comprehensively catalog common genetic variations and polymorphic haplotypes in a population, improve the accuracy of read mapping, and be effective tools in variant calling (see [3, 8, 27] for recent surveys). Representation [10, 22] and indexing [1, 11, 13, 18, 25, 35] of genome graphs, and sequence-to-graph alignment/mapping algorithms [5, 15, 19, 30, 31] are active research topics.

A key drawback of graph-based references is the presence of combinatorially explosive recombinant paths that correspond to groups of variants never observed in any sequenced haplotype. In fact, it is observed that incorporating all genetic variations to create the so called complete variation genome graph, perversely reduces the quality of read mapping [29]. This led to recent interest in variant selection algorithms to determine which subset of variants should be incorporated in a graph-based reference [14, 16, 29, 36]. In [14], Jain *et al.* developed a novel mathematical framework for variant selection that is aimed at providing mathematical guarantees of subsequent mapping accuracy. This is achieved by maximizing the number of variants that can be safely discarded while guaranteeing a chosen set of sequences can be mapped within a specified error limit. While this previous work presents several useful results when the set of sequences is chosen to reflect fixed length paths in complete variation graphs, the associated problems become NP-hard when the sequences are restricted to substrings of observed haplotypes, the so called *haplotype-aware variant selection* problem.

This work explores the haplotype-aware variant selection problem in the context of SNP variants, and contains the following contributions:

(1) We establish several hardness-of-approximation results regarding the minimization and maximization optimization versions of the haplotype-aware variant selection problem. In addition we provide simple approximation algorithms.
(2) We develop an Integer Linear Programming (ILP) formulation that provides optimal solutions for the haplotype-aware variant selection problem, allowing us to take advantage of the algorithms and software developed for this classic optimization problem.
(3) Using Gurobi solver [12] to implement our ILP formulation, we evaluate the achieved reduction in graph sizes and runtime performance on human chromosome sequences and

BCB '22, August 7–10, 2022, Northbrook, IL, USA

ACM ISBN 978-1-4503-9386-7/22/08.
https://doi.org/10.1145/3535508.3545556

SNP variants from the 1000 Genomes Project [2]. Comparing with algorithms in [14], we demonstrate significant reduction in graph sizes while retaining the ability to handle large variation graphs.

The rest of the paper is organized as follows: Section 2 discusses related works, including graph-based reference representations and haplotype-aware graph simplification; Section 3 describes the notations used throughout the paper, and provides formal definition of haplotype-aware variant selection. Section 4 provides our hardness and inapproximability results; Section 5 presents our ILP formulation, and corresponding algorithms; Experimental results are presented in Section 6; and Section 7 contains conclusion and directions for future research.

2 RELATED WORKS

Genome Graphs: Several tools have been developed recently that construct and utilize genome graphs. Some of the most prominent are Graphtyper [7], HISAT2 [17], and vg-toolkit [10].

Graphtyper utilizes variation graphs, and introduces algorithms to discover and genotype sequence variants. HISAT2 first constructs a linear graph of the reference genome, and subsequently augments variants as alternative paths through the graph. It also constructs a hierarchical index that consists of: (1) a global graph-FM-index (GFM) representing the linear reference, and (2) a set of overlapping local GFM indexes to collectively cover the complete variation graph. HISAT2 also provides a fast and sensitive graph aligner tool based on GCSA [35] (an extension of the BWT for graphs) and BWT. The vg-toolkit constructs genetic variation graphs by leveraging GCS2 [33] (an extension of the BWT for population graphs). It also provides a graph-based read aligner utilizing the seed-and-extend strategy and pairwise sequence alignment for directed acyclic graphs [20]. Vg-based read mapping can substantially improve read mapping and the fraction of reads mapped uniquely and correctly.

Recent works that utilize these graph representations include work by Sherman et al. that provides a pan-genome survey for various species, from bacteria to humans [32]; work by Liu et al. that constructs a complete high-quality pan-genome from soybean accessions that capture genomic diversity [23]; and work by Li et al. on using the minigraph toolkit to construct a pangenome graph to encode genomic diversity and structural variants [22]. Bestin et al. provide a review of software tools and strategies for building the human pangenome based on discussions by 45 scientists and software engineers from all over the world in 2019 [24].

Haplotype-aware graph tools: Various graph-based tools that utilize known haplotypes have been developed. CHOP augments haplotypes into a variation graph, then converts the haplotype-annotated population graph into a collection of sequences that cover all the observed k-long paths in haplotypes [26]. This collection of sequences is a compressed representation of all k-long paths through the graph. By doing this, CHOP constrains and bounds the search space when mapping reads. However, this haplotype-aware graph simplification is only designed for use on values of k equal to the length of short reads (e.g., 100 to 150), making this approach inapplicable for long reads or paired-end reads.

Siren et al. [34] developed haplotype-aware graphs by creating GBWT indexes (scalable implementation of the PBWT [35]). This augments haplotype information into vg-toolkit graphs and indexes haplotype paths in genome graphs using the GBWT index. More specifically, the augmented VG model and GBWT address a drawback of vg-toolkit, that paths corresponding to known haplotypes can break into sub-paths that represent recombinant paths. Unfortunately, this approach can lead to exponential growth during index construction. Hence, better indexing strategies or graph pruning algorithms are needed.

3 PRELIMINARIES

Let $R_1, R_2, \ldots, R_m$ denote m input reference haplotype sequences, each a string of length n over an alphabet Σ. We assume that one of these, say R_1, is a special reference. A *variation graph* is an edge-labeled directed multi-graph (henceforth simply called a graph) $G = (V, E, \ell)$, where $V = \{v_1, ..., v_{n+1}\}$ is an ordered set of vertices, E is the set of edges, and $\ell : E \rightarrow \Sigma$ specifies edge labels. The nodes $v_i \in V, (1 \leq i \leq n+1)$ represent a *coordinate axis* for the variation graph. To construct a variation graph from $R_1, \ldots, R_m$, first the haplotype R_1 is used to create the *linear backbone* of the graph G. This is a directed path $v_1, v_2, \ldots, v_{n+1}$ with character-labeled edges such that for $1 \leq i \leq n$ the edge (v_i, v_{i+1}) is labeled with the i^{th} character in the sequence R_1. We limit ourselves to substitution variants, also known as Single Nucleotide Polymorphisms (SNPs), which are bases in haplotypes differing from R_1 in the same position. At coordinate i, the set of SNPs is a subset of Σ. For each distinct SNP variant at location i, an additional edge between vertices v_i and v_{i+1} is created and labeled with that symbol. See Figure 1 (b) for an example. The number of variants is the same as the number of edges that are not part of the linear backbone. The number of variants at coordinate i ($1 \leq i \leq n$) equals the out-degree of the vertex v_i minus one, and is bound by $|\Sigma| - 1$.

We say that a path with edges $e_1, e_2, \ldots, e_\alpha$ matches the string $\ell(e_1) \circ \ell(e_2) \circ \ldots \circ \ell(e_\alpha)$, where $\circ$ denotes concatenation. Let $R_j[i \ldots i+\alpha-1]$ denote the α long substring $R_j[i] \circ \ldots \circ R_j[i+\alpha-1]$ of haplotype j. We refer to a path of length α starting at v_i and not matching $R_j[i \ldots i+\alpha-1]$ for any $1 \leq j \leq m$ as a *recombinant path*.

One objective is to obtain a graph that maintains close approximations to paths corresponding to substrings of the input haplotypes. This is captured by the following definition:

Definition 1. *A variation graph G is said to be $(\alpha, \delta)_h$-compatible if for all $1 \leq j \leq m$, $1 \leq i \leq n - \alpha + 1$, there exists a path starting at v_i that matches a string with Hamming distance at most δ from $R_j[i \ldots i+\alpha-1]$.*

A second objective is to find a subgraph of G that limits the number of recombinant paths. Formally, we call a subgraph G' of G a *reduced variation graph* if $G' = (V', E', \ell')$, where $V' = V$, $E' \subseteq E$ such that $E \setminus E'$ only represents variants, and for all $e \in E'$, $\ell'(e) = \ell(e)$. Combining both objectives leads to the following problem definition:

Problem 1 (Haplotype-Aware Variant Reduction with Reference). *Given a variation graph $G = (V, E)$ and haplotype set R_1,*

..., R_m (with R_1 serving as designated reference), compute an $(\alpha, \delta)_h$-compatible reduced variation graph G' with the minimum number of variants.

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.

If one is willing to relax the condition that edges corresponding to the linear backbone constructed from R_1 cannot be deleted, and only maintain that the reduced variation graph remains connected, a slightly different problem can be formulated. Now, we let the edges in the backbone be considered as variants as well. Then we can define:

Problem 2 (Haplotype-Aware Variant Reduction without Reference). *Given variation graph $G = (V, E)$ and haplotype set $R_1, \ldots, R_m$ compute an $(\alpha, \delta)_h$-compatible reduced variation graph G' that is connected and has the minimum number of variants.*

4 INAPPROXIMABILITY

In previous work, Jain *et al.* proved the NP-completeness of the decision version of Problems 1 and 2 [14]. Here we will focus on the inapproximability of Problem 1. Two natural optimization versions of Problem 1 can be formulated based on the decision version. The first is to minimize the number of variants kept while maintaining that the resulting graph is $(\alpha, \delta)_h$-compatible, with a solution's value being the number of variants kept. We call this *Haplotype-Aware Variant Maintained Minimization*. The second is to maximize the number of variants removed while maintaining that the resulting graph is $(\alpha, \delta)_h$-compatible, with a solution's value being the number of variants removed. We call this *Haplotype-Aware Variant Removed Maximization*. For a given instance, an optimal solution to either clearly results in the same number of variants maintained/removed, however, the approximation factors possible by polynomial-time algorithms differ, assuming P $\neq$ NP.

Minimization: We can utilize a close connection to the Uniform Hypergraph Vertex Cover and Multi-Cover problems to give several hardness results and approximation algorithms for the minimization version. We first use a reduction from the k-Uniform HyperGraph Vertex Cover problem to establish hardness results. In this problem, one is given a hypergraph $H = (V, E)$ where all edges are of size k and the aim is to find a minimum subset of vertices $V' \subseteq V$ such that every hyperedge includes at least one vertex in V'. The reduction is as follows: given a hypergraph $H = (V, E)$, first assign the vertices in V an arbitrary ordering. Then, for each hyperedge $e_i = \{v_{i_1}, \ldots, v_{i_k}\} \in E$ create a haplotype of length $|V|$, with 1's in positions $i_1, \ldots, i_k$ and 0's elsewhere. Next, make a variation graph G, consisting of a linear backbone having $|V|+1$ vertices with an edge labeled 0 and an edge labeled 1 at each coordinate. Make the reference $R_1 = 0^{|V|}$, $\delta = k - 1$, and $\alpha = |V|$.

To see the correctness of the above reduction, note that if there exists a vertex cover of size t in H, say $v_{j_1}, \ldots, v_{j_t}$, we can maintain variants labeled 1's from positions $j_1, \ldots, j_t$ and delete all other variants labeled 1 from G. Since every edge in H is covered by at least one of the vertices at these indices, every haplotype has at least one '1' variant that is preserved. Since the total number of 1's in each haplotype is k, at most $k - 1 = \delta$ mismatches now occur between any haplotype and G. Hence, this provides a valid solution with value t to the Haplotype-Aware Variant Maintained Minimization instance. Conversely, suppose there exists a solution to the Haplotype-Aware Variant Maintained Minimization instance where t variations are maintained. Then every haplotype has at most $\delta = k - 1$ mismatches with G and hence at least one '1' that matches a variation in G. Therefore, taking the vertices corresponding to maintained variations in G gives a set of vertices for H where every edge has at least one vertex in the preserved set, providing a vertex cover of size t. Since it is NP-hard to obtain an approximation better than $k - 1 - \varepsilon$ for k-Uniform Hypergraph Vertex Cover problem, $k \geq 3$ [6], this approximation preserving reduction implies that it is NP-hard to obtain a reduction better than $\delta - \varepsilon$ for any $\delta \geq 2$ and constant $\varepsilon > 0$,

Theorem 4.1. *There does not exist a polynomial-time $(\delta - \varepsilon)$-approximation algorithm for Haplotype-Aware Variant Maintained Minimization for any $\delta \geq 2$ and constant $\varepsilon > 0$ assuming $P \neq NP$.*

To obtain additional inapproximability results, we observe that the Set Cover problem with universe $\mathcal{U}$ and collection of sets $\mathcal{S}$ can be reduced to the Hypergraph Vertex Cover problem with $|\mathcal{S}|$ vertices and $|\mathcal{U}|$ hyperedges. This can then be reduced to k-Uniform Hypergraph Vertex Cover problem by making k equal to the largest cardinality of any existing edge, and then adding new 'dummy' vertices to each edge until they are all of cardinality k (the number of edges is not changed). This creates a hypergraph with $O(|\mathcal{U}||\mathcal{S}|)$ vertices and $|\mathcal{U}|$ hyperedges. Finally, we can reduce this k-Uniform Hypergraph Vertex Cover instance to Haplotype-Aware Variant Maintained Minimization as discussed above such that the number of haplotypes m equals the number of hyperedges $|\mathcal{U}|$, and the number of vertices in the variation graph is equal to the number vertices in the k-uniform hypergraph plus 1, which is $O(|\mathcal{U}||\mathcal{S}|)$. All of these reductions are approximation preserving in that a size t set cover is equivalent to a size t vertex cover for both hypergraphs, and this is equivalent to t variants being maintained in the variation graph. Hence, a well-known $(1 - o(1)) \log |\mathcal{U}|$ inapproximability result for Set Cover [9], implies that obtaining an $(1 - o(1)) \log m$-approximation for Haplotype-Aware Variant Maintained Minimization is NP-hard. Moreover, the number of vertices in our instance of Haplotype-Aware Variant Maintained Minimization is $O(|\mathcal{U}||\mathcal{S}|)$, and since a $o(\log |\mathcal{S}|)$-approximation for set cover is also NP-hard[1], this implies a $o(\log |V|)$-approximation for our problem is NP-hard as well. These results are summarized below.

Theorem 4.2. *There does not exist a polynomial-time $(1-o(1)) \log m$-approximation algorithm or a polynomial-time $o(\log |V|)$-approximation algorithm for Haplotype-Aware Variant Maintained Minimization assuming $P \neq NP$.*

A reduction from Haplotype-Aware Variant Maintained Minimization to a well studied problem exists as well. Our problem can be modeled as a version of the Multi-Covering problem. Using the Booelean constraint matrix A described in Section 5, the problem can be rewritten as the integer linear program $\min_x \mathbb{1} \cdot x$ s.t., $Ax \geq A\mathbb{1} - \delta\mathbb{1}$, where $x_i \in \{0, 1\}$. Here, unlike in Section 5, $x_i = 1$ is interpreted as a variant i being maintained and $x_i = 0$ is interpreted as a variant i being removed. This is exactly a Multi-Cover instance.

[1] The inapproximability result in [9] holds when $|\mathcal{S}| = |\mathcal{U}|^{\Theta(1)}$.

For the Multi-Covering problem we can use the greedy algorithm that starts with x as the zero vector and repeatedly makes $x_i = 1$ for the i that maximizes the number of infeasible rows in Ax that increase in value. It repeats this until a feasible solution is reached. The greedy algorithm provides a logarithmic approximation [37].

Theorem 4.3. *There exists a polynomial-time $O(\log |\mathcal{V}|)$- approximation algorithm for Haplotype-Aware Variant Maintained Minimization, where $\mathcal{V}$ is the set of variants in G.*

A polynomial-time approximation algorithm with approximation factor $(\max_i \sum_j a_{ij} - \min_h \sum_j a_{hj} + \delta + 1)$ where a_{ij} are entries from the matrix A, is also possible [28]. In the case where all haplotypes (besides R_1) have the same number of variants in all α sized intervals, this simplifies to a $(\delta + 1)$-approximation.

Maximization: The reduction by Jain et al. used to prove that the decision version is NP-complete also proves that Haplotype-Aware Variant Removed Maximization is NP-hard to approximate within a factor of $|V|^{1-\epsilon}$ for any constant $\varepsilon > 0$. This is since it provides a reduction from the Independent Set problem on an instance $G' = (V', E')$ to an instance $G = (V, E)$ of Haplotype-Aware Variant Removed Maximization such that $\alpha = |V'| = |V| - 1$ and a solution where t variants are removed yields a solution to the Independent Set problem consisting of a t-sized independent set. This, combined with the NP-hardness of finding a solution to the Independent Set problem within an $|V'|^{1-\varepsilon}$-approximation factor of optimal [38], proves the result.

Theorem 4.4. *It is NP-hard to obtain a $O(|V|^{1-\varepsilon})$-approximation for Haplotype-Aware Variant Removed Maximization.*

It should also be noted that an $\frac{\alpha}{\delta}$-approximation can be obtained for both the haplotype-aware and non-haplotype-aware versions via a simple algorithm. Consider the strategy of removing all variants from coordinates that are $\lceil \frac{\alpha}{\delta} \rceil$ apart (i.e., $\lceil \frac{\alpha}{\delta} \rceil - 1$ coordinates between them). In particular, let potential solution i be the solution were all variants are removed at coordinates $i + j\lceil \frac{\alpha}{\delta} \rceil$ for $1 \le i < \lceil \frac{\alpha}{\delta} \rceil$ and $0 \le j \le \frac{n-i}{\lceil \frac{\alpha}{\delta} \rceil}$. For every solution and any interval of size α there are at most δ coordinates with variants removed, hence in that interval at most δ variants removed for every haplotype. At the same time, since these $\lceil \frac{\alpha}{\delta} \rceil$ solutions partition all of the variants, one of them must contain at least $\frac{1}{\lceil \frac{\alpha}{\delta} \rceil}$ fraction of all the variants. Therefore, taking the solution i that contains the most variants overall gives us a solution with value at least $\frac{val(OPT)}{\lceil \frac{\alpha}{\delta} \rceil}$. In the cases we studied in this work, we have δ being equal to some relatively small proportion of α, such as $\delta = .01\alpha$, or $\delta = .1\alpha$. For our experiments, this approach would give approximation ratios too large to be of interest.

Theorem 4.5. *There exists a polynomial-time $\frac{\alpha}{\delta}$-approximation algorithm for Haplotype-Aware Variant Removed Maximization.*

5 PROPOSED ALGORITHMS

In this section, we propose ILP solutions for Problems 1 and 2.

5.1 With a designated reference

Assume that a variation graph $G = (V, E, \ell)$ and haplotypes $R_1, \ldots, R_m$ are given. Recall that R_1 is considered as a special reference and the path corresponding to R_1 in G does not contain variants. Let $\mathcal{V}$ denote the list of variants. We will describe each variant in $\mathcal{V}$ as the tuple (p, s) where $p \in \{1, \ldots, n\}$ denotes the coordinate of the variant and $s \in \Sigma$ denotes the symbol for the variant. We consider $\mathcal{V}$ as being ordered lexicographically by p then s, although not essential for the algorithm. We also let $\mathcal{L} \subseteq \{1, \ldots, n\}$ denote the list of coordinates where there exists some variant and the coordinate is at most $n - \alpha + 1$, i.e.,

$$\mathcal{L} = \{p \mid p \le n - \alpha + 1 \text{ and } (p, s) \in \mathcal{V} \text{ for some } s \in \Sigma\}.$$

To present the ILP formulation of the problem, we first describe the constraint matrix A. Let A be a Boolean matrix of size $((m-1) \cdot |\mathcal{L}|) \times |\mathcal{V}|$. We use a_{ij} to refer to the element of the i^{th} row and j^{th} column of A. Let A be initially all 0's. We assign to each column in A a variant in $\mathcal{V}$, and we assign to each row in A both a haplotype and α-long span of coordinate values that begins with some $p \in \mathcal{L}$. Specifically, the j^{th} column is assigned the j^{th} variant (p_j, s_j). For row i, if $i - 1 = (m-1)p + r$, where $0 \le r < m - 1$, then row i is assigned haplotype R_{r+2} and the interval $[\mathcal{L}[p+1], \mathcal{L}[p+1] + \alpha - 1]$. The non-zero entries of A are determined as follows:

- For $1 \le i \le (m-1)|\mathcal{L}|$, find $p \ge 0$ and $r \in [0, m-2]$ such that $i - 1 = (m-1)p + r$;
- then for $1 \le j \le |\mathcal{V}|$, make $a_{ij} = 1$ if for $(p_j, s_j) \in \mathcal{V}$, we have $p_j \in [\mathcal{L}[p+1], \mathcal{L}[p+1] + \alpha - 1]$ and $R_{r+2}[p_j] = s_j$.

See Figure 1 (a)-(c) for an example. In practice, we can remove the duplicate rows observed in part (c), more specifically, we removed rows 4, 6 and 9. Note that by precomputing $\mathcal{L}$, the list of haplotypes, and the list of variants, determining whether $a_{ij} = 1$ or $a_{ij} = 0$ for a given (i, j) can be done in constant time, rather than having to explicitly construct the matrix A.

Letting $\mathbb{1}$ be the $|\mathcal{V}| \times 1$ vector consisting of all 1's the ILP for finding the maximum number of variants that can be removed from G while maintaining $(\alpha, \delta)_h$-compatibility can now be stated as:

$$\begin{aligned} \max_x \quad & \mathbb{1}^T x \\ \text{s.t.} \quad & A \cdot x \le \delta \mathbb{1} \\ & x_i \in \{0, 1\} \qquad \forall i \in [1, |\mathcal{V}|] \end{aligned}$$

Given an optimal solution x to the above ILP, to obtain the solution to Problem 1, for every $x_i = 1$, remove variant (p_i, s_i). The remaining variants are maintained.

The reader may notice that in the ILP formulation above not every α-sized interval is explicitly represented by the constraint matrix. This is since it is adequate to only consider α-sized intervals that begin at some coordinate in $\mathcal{L}$. Indeed, by shifting an α-sized interval to the left of any starting position in $\mathcal{L}$, one can easily check that the number of mismatches between any path in G and any substring in a haplotype spanning the same indices can only decrease or remain constant until a new variant is encountered by the starting position of the interval.

5.2 Without a designated reference

In the case where there is no designated reference, the set of variants becomes equivalent to the set of edges. However, in order to maintain connectivity, we only include $(i, \ell(e))$ in $\mathcal{V}$ for edges $e = (v_i, v_{i+1}) \in E$ where the out-degree of v_i, denoted $\deg^+(i)$, is greater than 1. With this new variant set $\mathcal{V}$, we first construct the

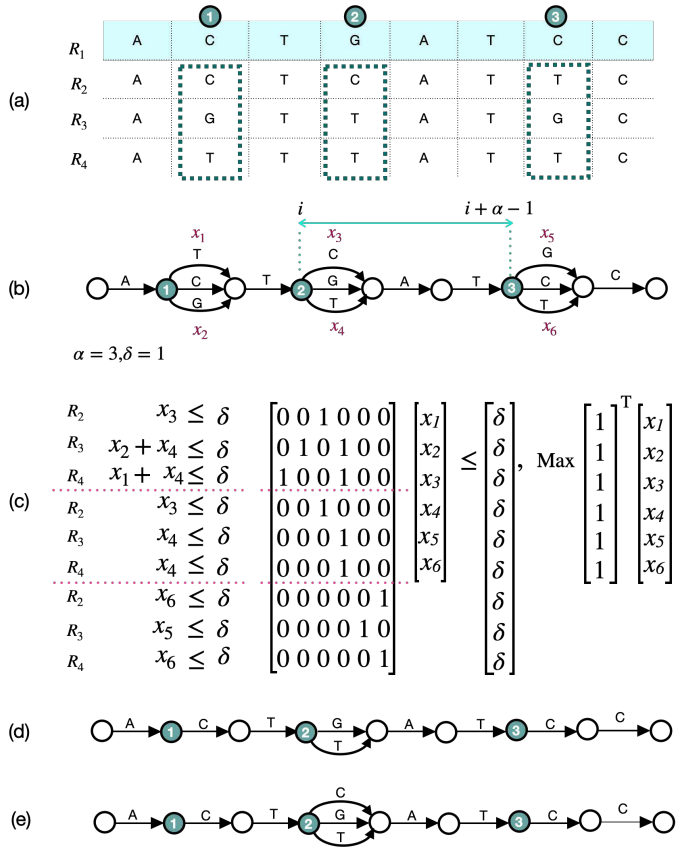

Figure 1: An example to illustrate the proposed *ILP* solution (a) List of given haplotypes. (b) Complete variation graph for the given haplotypes. (c) ILP formulation with constraints. (d) Reduced variation graph using *ILP*. (e) Reduced variation graph using *Greedy*/*LP*.

matrix A as above. Next, we add the following rows to A: for every $i \in \mathcal{L}$ we create a row. For a given $i \in \mathcal{L}$, take all variants in $\mathcal{V}$ that are at coordinate i. For each of these variants, if the variant has index j in $\mathcal{V}$, then we put a 1 in column j. For each of these added rows, we add a corresponding new row to the vector on the right-hand side of the inequality and set it equal to $\deg^+(v_i) - 1$. This ensures that no more than $\deg^+(v_i) - 1$ variants are removed from any coordinate and that the resulting graph remains connected.

6 EXPERIMENTS

We developed software to implement our ILP solutions in C++ (code available at https://github.com/NedaTavakoli/hg). We evaluate the effectiveness of our ILP solution for Problem 1, henceforth denoted *ILP*, by measuring the percentage of variants removed from the corresponding complete variation graph. In addition to this absolute performance measure, we compare *ILP* to the previous variant reduction algorithms introduced in [14], a greedy algorithm, denoted *Greedy*, and LP algorithm, denoted *LP*. These algorithms seek to preserve all α-long paths occurring in the complete variation graph, subject to δ or fewer mismatches. Thus, any solutions to this problem are automatically valid (but not necessarily optimal) solutions to the haplotype-aware problem. This relative comparison will reveal whether *ILP* targeting the hapylotype-aware problem directly is necessary, and is also evaluated in the context of runtime performance and memory usage of all three algorithms. For brevity, all results are presented for Problem 1, as the same behavior is observed for Problem 2 also. **The greedy and LP solutions:** *Greedy* works by considering the vertices in G from left to right and keeping count of the number of variants removed from every α-sized interval containing the current vertex. If the budget δ of mismatches is not reached, all variants at the current vertex are removed. *LP* works by recognizing that when preserving all graph paths (haplotype as well as recombinant), optimal solutions either remove or maintain all variants at any specific vertex. This leads to a simpler formulation, and a totally unimodular matrix for specifying constraints, allowing polynomial-time solution through LP relaxation. See Figure 1 (d)-(e) for an illustration of the reduced graphs resulting from *ILP*, *LP*, and *Greedy*.

Variation graph construction: We evaluate the algorithms using two variation graphs. The larger of the two is constructed from human chromosome 1 (249 Mbp) and is denoted as g_chr1_SNP. The smaller graph is constructed from the human chromosome 22 (51 Mbp) and is denoted as g_chr22_SNP. The variations used in the graph construction arise from a list of 5,008 haplotypes that exist for the human genome. We built these variation graphs corresponding to SNPs while annotating every edge with a list of haplotypes containing that edge. We downloaded SNPs from the 1000 Genomes Project Phase 3 [2].

Our graph construction leveraged vcftools [4], and bcftools [21] to extract SNPs from the 1000 Genomes Project variant files, as well as list of haplotypes per SNP position. Summary statistics of these variants, and the graphs we built from them, are listed in Table 1. We also implemented a shell script to construct haplotype-annotated variation graphs. The code is available at https://github.com/NedaTavakoli/havg.

Hardware and software: We used Gurobi 9.5.0 solver to solve the *ILP* and *LP* instances. The algorithms were run on dual Intel Xeon Gold 6226 CPUs (2.70 GHz) processors. Each contains 2×12 physical cores and 384 GB RAM. Gurobi takes advantage of multiple cores when solving the *ILP* and *LP* instances; however, the *Greedy* algorithm is sequential.

The α and δ parameters: We use multiple α and δ values to test the performance of the algorithms. The values selected for α are 150 bp, 1 kbp, 5 kbp, and 10 kbp, inspired by short and long read lengths. For δ, we choose 1%, 5%, and 10% of α, rounded up to the nearest integer.

Results: The results for graph g_chr22_SNP are shown in Table 2 and Figure 2. The results for graph g_chr1_SNP are shown in Table 3 and Figure 3. For the *ILP* algorithm, we report four statistics: (i) count of variants retained, (ii) run-time, (iii) peak memory usage, and (iv) percentage of variant reduction achieved. We also compared the percentage of variants removed using previously developed *Greedy* and *LP* algorithms.

Table 1: Genome variation graphs used for experimental evaluation.

Graph label	Chr	Type of variants	No. of variants	No. of variant containing loci	No. of haplotypes
g_chr1_SNP	1	SNPs	6,234,046	6,215,039	5008
g_chr22_SNP	22	SNPs	1,063,617	1,059,517	5008

Table 2: Results for *ILP* haplotype-aware algorithm and comparison with variant reduction achieved using *Greedy* and *LP* on genome variation graph for chromosome 22.

α	δ	Time (s)	# Variants Retained	Memory Usage (GB)	Variant Reduction		
			ILP		*ILP*	*Greedy*	*LP*
$\alpha = 150$	$\delta = 2$	12.03	357279	3	66.41%	33.51%	33.72%
	$\delta = 8$	6.34	30476	3	97.13%	91.36%	91.41%
	$\delta = 15$	3.87	3323	3	99.68%	99.35%	99.36%
$\alpha = 1000$	$\delta = 10$	10.1	385531	8	63.75%	30.56%	30.82%
	$\delta = 50$	13.34	7176	8	99.32%	98.07%	98.10%
	$\delta = 100$	13.76	956	10	99.91%	99.76%	99.77%
$\alpha = 5000$	$\delta = 50$	50.76	365598	32	65.63%	31.85%	32.12%
	$\delta = 250$	45.36	2435	32	99.77%	99.30%	99.32%
	$\delta = 500$	39.05	296	29	99.97%	99.92%	99.92%
$\alpha = 10000$	$\delta = 100$	64.87	375126	105	64.73%	32.10%	32.37%
	$\delta = 500$	54.01	1367	77	99.87%	99.52%	99.52%
	$\delta = 1000$	52.09	186	66	99.98%	99.93%	99.93%

Table 3: Results for *ILP* haplotype-aware algorithm and comparison with variant reduction achieved using *Greedy* and *LP* on genome variation graph for chromosome 1.

α	δ	Time (s)	# Variants Retained	Memory Usage (GB)	Variant Reduction		
			ILP		*ILP*	*Greedy*	*LP*
$\alpha = 150$	$\delta = 2$	23.04	2355632	14	62.21%	36.54%	36.70%
	$\delta = 8$	24.65	146312	14	97.65%	94.44%	94.47%
	$\delta = 15$	20.06	3545	14	99.94%	99.78%	99.78%
$\alpha = 1000$	$\delta = 10$	84.09	2030326	44	67.43%	33.75%	33.94%
	$\delta = 50$	73.04	10034	42	99.84%	99.36%	99.36%
	$\delta = 100$	67.06	1503	39	99.97%	99.96%	99.96%
$\alpha = 5000$	$\delta = 50$	145.09	1930345	121	69.03%	35.24%	35.44%
	$\delta = 250$	128.01	5342	115	99.91%	99.89%	99.86%
	$\delta = 500$	125.09	244	113	100.00%	99.99%	99.99%
$\alpha = 10000$	$\delta = 100$	146.19	1783400	130	71.40%	35.49%	∞
	$\delta = 500$	138.08	2065	125	99.97%	99.91%	∞
	$\delta = 1000$	136.10	164	101	100.00%	100.00%	∞

The results indicate that *ILP* outperforms *Greedy* and *LP* in all experiments, as one would expect from an algorithm that guarantees optimal solution. The degree of out-performance is highest for small values of δ. This is because removing variants when only few errors are tolerated is tricky, and the optimal algorithm clearly outshines here. For the case of $\delta = 1\%$, *ILP* removes roughly twice as many variants as the other two approaches. For larger δ values, most variants can be removed without loss of $(\alpha, \delta)_h$ compatibility. Here, *ILP* shows marginal gains over the others (3-6% additional variants removed) for $\alpha = 150$, and the gains are negligible for longer α. Both Chromosome 1 and Chromosome 22 are predominantly biallelic (only one variant in addition to the reference base at a vertex), at 99.82% and 99.61% frequency, respectively. This favors *Greedy* as it removes or retains all variants at a vertex, and lacks the ability to select an appropriate subset of variants instead. For variation graphs with higher multi-allelic frequency, the performance gap between *ILP* and *Greedy* would be even higher.

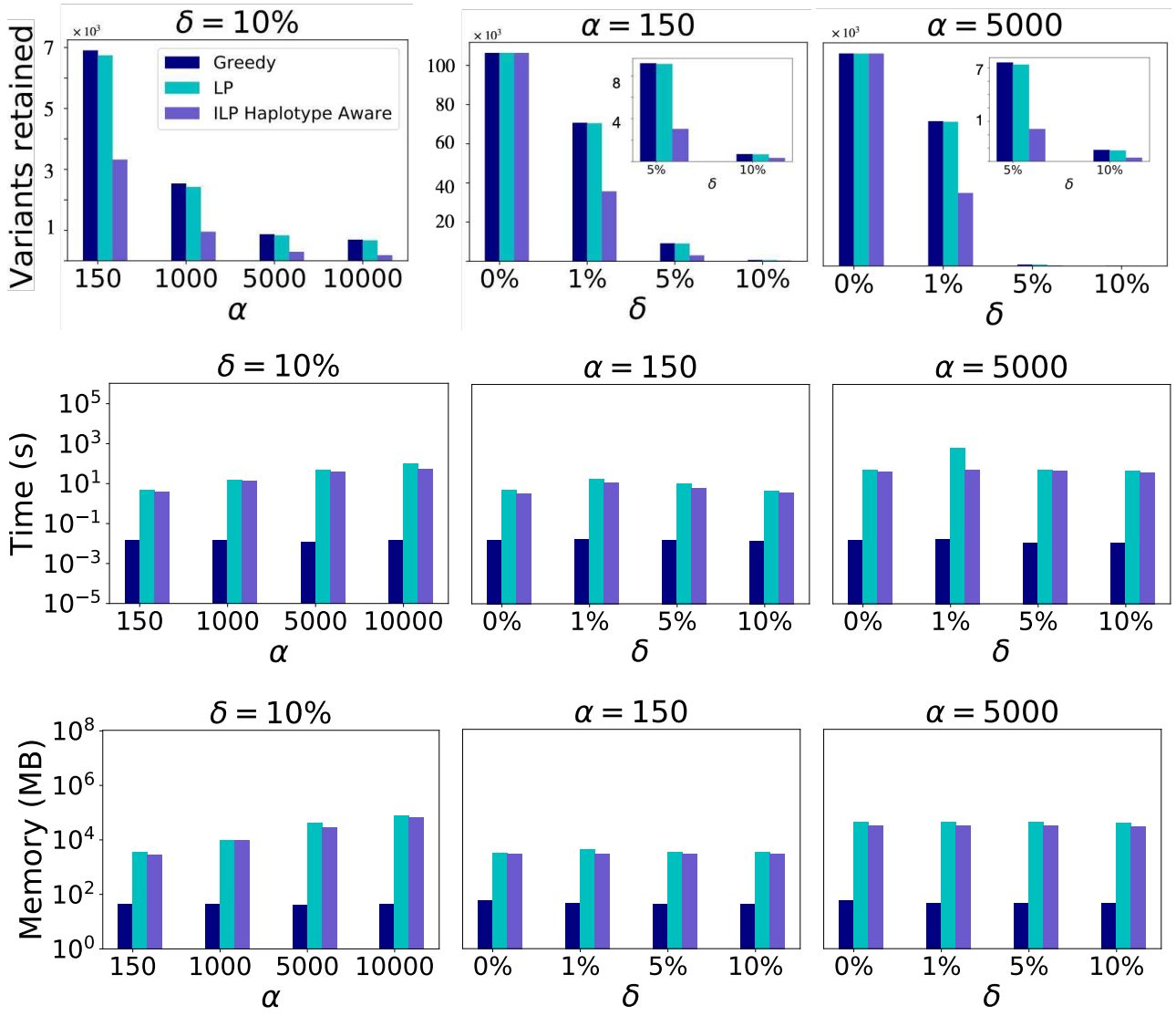

Figure 2: Empirical evaluation of *Greedy*, *LP*, and *ILP* algorithms using human variation graphs g_chr1_SNP and g_chr22_SNP. These plots illustrate number of variants retained for various choices of α and δ. Size of the complete variation graph ($\delta = 0\%$) is included for comparison. The Y-axes are shown in in log-scale for the time and memory plots.

While the results are shown for wide choices of α and δ to comprehensively reflect properties of the algorithm/implementation, it should be kept in mind that practical use cases where only substitution errors occur typically will use small values of α and δ. Here the out-performance of *ILP* justifies its use. This is because long read technologies predominantly make insertion/deletion errors, and longer stretches of genome invariably contain genomic insertions and deletions with respect to other reference haplotypes. Because of this, a model restricted to substitutions alone has no validity over longer lengths. The best use of such a model is to seed an alignment using short substring matches or when matching short reads. In both of these cases, *ILP* provides significantly better variant reduction.

As for run-time, *Greedy* is the fastest while *LP* and *ILP* are similar (Figures 2 and 3). Note that even though *ILP* edges out *LP* slightly and is also able to solve larger α sizes than *LP*, this is primarily because of our improved implementation in *ILP* to dynamically generate the non-zero entries of the constraint matrix on demand. Hence, both algorithms should be seen as providing similar run-time performance. The run-time and memory requirements of *ILP* allow its usage to achieve optimal variant selection in all cases tested.

7 CONCLUSIONS AND FUTURE WORK

In this work, we investigated the haplotype-aware variant selection problem under the Hamming distance metric. We proved several

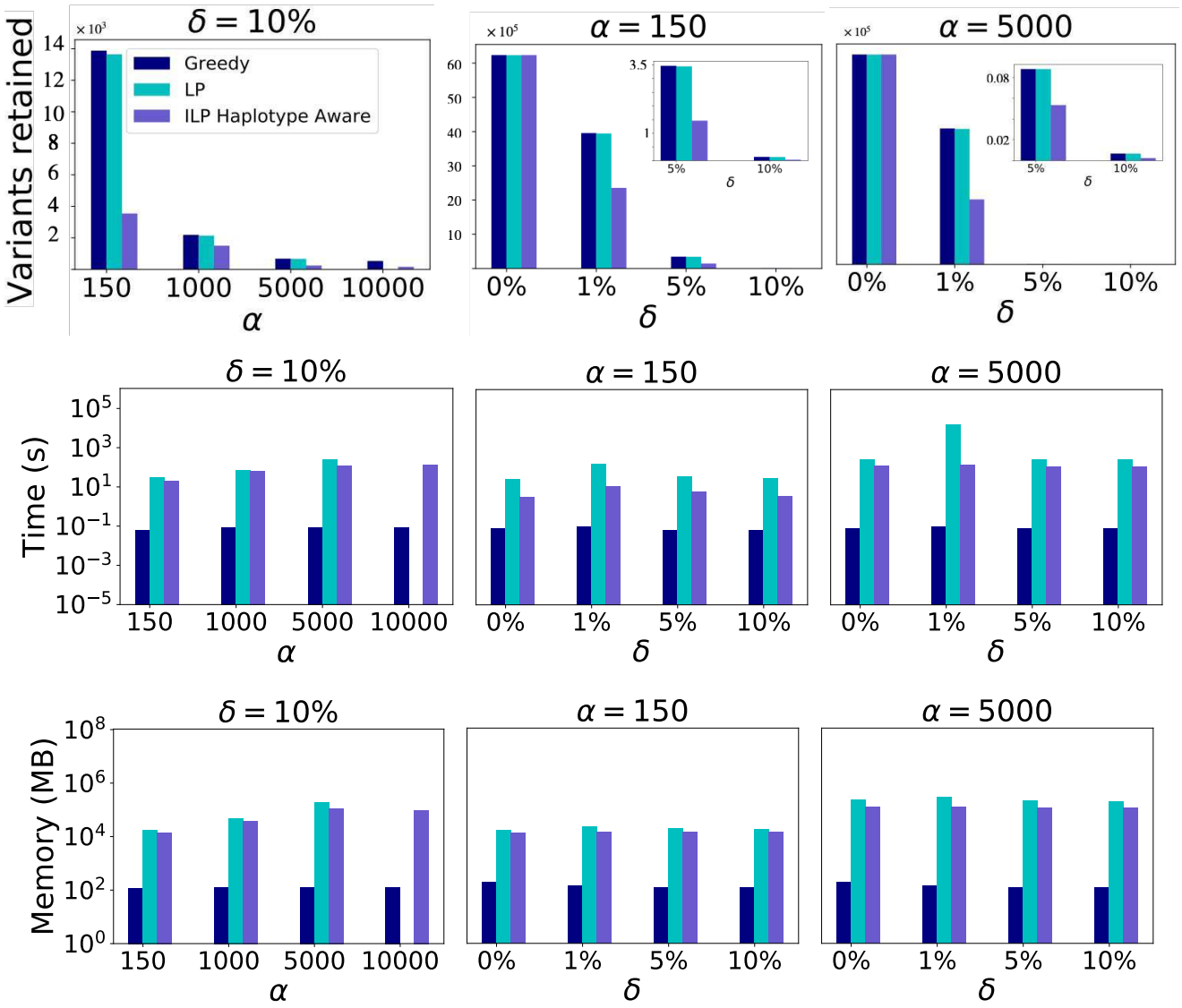

Figure 3: Empirical evaluation of *Greedy* and *LP* algorithms using two human variation graphs g_chr1_SNP and g_chr22_SNP. These plots demonstrate reduction achieved in graph sizes while varying α and δ parameters. Size of the complete variation graph ($\delta = 0\%$) is included for comparison. Numbers on top of bars present actual data, useful for comparison when both *Greedy* and *LP* achieve close results. Result of *LP* algorithm is missing for $\alpha = 10,000$ (left-most column) because Gurobi LP solver crashed due to insufficient memory. The Y-axes are log-scaled in the time and memory plots.

hardness-of-approximation results for the minimization and maximization versions of the problem and proposed approximation algorithms. We then provided an ILP formulation of the problem and demonstrated experimentally that this formulation is effective in finding optimal solutions even on human chromosome scale graphs and for a variety of sequence lengths and error percentage thresholds. In addition to ensuring optimality, the gains compared to other suboptimal algorithms are substantial for the realistic case of short sequence lengths and error thresholds, more appropriate when only substitutions are allowed. Future extensions of this work include improving the execution of the ILP solver by developing problem specific branch-and-cut techniques, and implementing the logarithmic approximation algorithms presented in Section 4 and comparing them to the existing Greedy and LP solutions described in Section 6. We also plan to experimentally evaluate the impact of the variant reduction obtained here on sequence-to-graph mapping accuracy.

Finally, this work focused exclusively on the Hamming distance version of the problem. While this constitutes a significant advance

since the haplotype-aware problem is hitherto unsolved [14], generalizing it to the edit distance version is more realistic for longer sequence lengths and particularly for long reads. This remains an important open problem.

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation under CCF-1816027.

REFERENCES

[1] Xian Chang, Jordan Eizenga, Adam M Novak, Jouni Sirén, and Benedict Paten. 2020. Distance indexing and seed clustering in sequence graphs. *Bioinformatics* 36, Supplement_1 (2020), i146–i153.

[2] 1000 Genomes Project Consortium et al. 2015. A global reference for human genetic variation. *Nature* 526, 7571 (2015), 68–74.

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

[4] Petr Danecek, Adam Auton, Goncalo Abecasis, Cornelis A Albers, Eric Banks, Mark A DePristo, Robert E Handsaker, Gerton Lunter, Gabor T Marth, Stephen T Sherry, et al. 2011. The variant call format and VCFtools. *Bioinformatics* 27, 15 (2011), 2156–2158.

[5] Charlotte A Darby, Ravi Gaddipati, Michael C Schatz, and Ben Langmead. 2020. Vargas: heuristic-free alignment for assessing linear and graph read aligners. *Bioinformatics* 36, 12 (2020), 3712–3718.

[6] Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. 2005. A New Multilayered PCP and the Hardness of Hypergraph Vertex Cover. *SIAM J. Comput.* 34, 5 (2005), 1129–1146. https://doi.org/10.1137/S0097539704443057

[7] Hannes P Eggertsson, Hakon Jonsson, Snaedis Kristmundsdottir, Eirikur Hjartarson, Birte Kehr, Gisli Masson, Florian Zink, Kristjan E Hjorleifsson, Aslaug Jonasdottir, Adalbjorg Jonasdottir, et al. 2017. Graphtyper enables population-scale genotyping using pangenome graphs. *Nature genetics* 49, 11 (2017), 1654.

[8] Jordan M Eizenga, Adam M Novak, Jonas A Sibbesen, Simon Heumos, Ali Ghaffaari, Glenn Hickey, Xian Chang, Josiah D Seaman, Robin Rounthwaite, Jana Ebler, et al. 2020. Pangenome Graphs. *Annual Review of Genomics and Human Genetics* 21 (2020).

[9] Uriel Feige. 1998. A Threshold of ln *n* for Approximating Set Cover. *J. ACM* 45, 4 (1998), 634–652. https://doi.org/10.1145/285055.285059

[10] Erik Garrison, Jouni Sirén, Adam M Novak, Glenn Hickey, Jordan M Eizenga, Eric T Dawson, William Jones, Shilpa Garg, Charles Markello, Michael F Lin, et al. 2018. Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nature biotechnology* 36, 9 (2018), 875–879.

[11] Ali Ghaffaari and Tobias Marschall. 2019. Fully-sensitive seed finding in sequence graphs using a hybrid index. *Bioinformatics* 35, 14 (2019), i81–i89.

[12] Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[13] Guillaume Holley, Roland Wittler, and Jens Stoye. 2016. Bloom Filter Trie: an alignment-free and reference-free data structure for pan-genome storage. *Algorithms for Molecular Biology* 11, 1 (2016), 1–9.

[14] Chirag Jain, Neda Tavakoli, and Srinivas Aluru. 2021. A variant selection framework for genome graphs. *Bioinformatics* 37, Supplement_1 (2021), i460–i467.

[15] Chirag Jain, Haowen Zhang, Yu Gao, and Srinivas Aluru. 2020. On the Complexity of Sequence-to-Graph Alignment. *Journal of Computational Biology* 27, 4 (2020), 640–654.

[16] Daehwan Kim, Joseph Paggi, and Steven L Salzberg. 2018. Hisat-genotype: Next generation genomic analysis platform on a personal computer. *BioRxiv* (2018), 266197.

[17] Daehwan Kim, Joseph M Paggi, Chanhee Park, Christopher Bennett, and Steven L Salzberg. 2019. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. *Nature biotechnology* 37, 8 (2019), 907–915.

[18] Alan Kuhnle, Taher Mun, Christina Boucher, Travis Gagie, Ben Langmead, and Giovanni Manzini. 2020. Efficient construction of a complete index for pan-genomics read alignment. *Journal of Computational Biology* 27, 4 (2020), 500–513.

[19] Anna Kuosmanen, Topi Paavilainen, Travis Gagie, Rayan Chikhi, Alexandru Tomescu, and Veli Mäkinen. 2018. Using minimum path cover to boost dynamic programming on DAGs: co-linear chaining extended. In *International Conference on Research in Computational Molecular Biology*. Springer, 105–121.

[20] Christopher Lee, Catherine Grasso, and Mark F Sharlow. 2002. Multiple sequence alignment using partial order graphs. *Bioinformatics* 18, 3 (2002), 452–464.

[21] Heng Li. 2011. A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics* 27, 21 (2011), 2987–2993.

[22] Heng Li, Xiaowen Feng, and Chong Chu. 2020. The design and construction of reference pangenome graphs with minigraph. *Genome Biology* 21, 1 (2020), 1–19.

[23] Yucheng Liu, Huilong Du, Pengcheng Li, Yanting Shen, Hua Peng, Shulin Liu, Guo-An Zhou, Haikuan Zhang, Zhi Liu, Miao Shi, et al. 2020. Pan-genome of wild and cultivated soybeans. *Cell* 182, 1 (2020), 162–176.

[24] Bastien Llamas, Giuseppe Narzisi, Valerie Schneider, Peter A Audano, Evan Biederstedt, Lon Blauvelt, Peter Bradbury, Xian Chang, Chen-Shan Chin, Arkarachai Fungtammasan, et al. 2019. A strategy for building and using a human reference pangenome. *F1000Research* 8, 1751 (2019), 1751.

[25] Shoshana Marcus, Hayan Lee, and Michael C Schatz. 2014. SplitMEM: a graphical algorithm for pan-genome analysis with suffix skips. *Bioinformatics* 30, 24 (2014), 3476–3483.

[26] Tom Mokveld, Jasper Linthorst, Zaid Al-Ars, Henne Holstege, and Marcel Reinders. 2020. CHOP: haplotype-aware path indexing in population graphs. *Genome biology* 21, 1 (2020), 1–16.

[27] Benedict Paten, Adam M Novak, Jordan M Eizenga, and Erik Garrison. 2017. Genome graphs and the evolution of genome inference. *Genome research* 27, 5 (2017), 665–676.

[28] David Peleg, Gideon Schechtman, and Avishai Wool. 1993. Approximating Bounded 0-1 Integer Linear Programs. In *Second Israel Symposium on Theory of Computing Systems, ISTCS 1993, Natanya, Israel, June 7-9, 1993, Proceedings*. IEEE Computer Society, 69–77. https://doi.org/10.1109/ISTCS.1993.253482

[29] Jacob Pritt, Nae-Chyun Chen, and Ben Langmead. 2018. FORGe: prioritizing variants for graph genomes. *Genome biology* 19, 1 (2018), 1–16.

[30] Gunnar Rätsch and Martin Vechev. 2020. AStarix: Fast and Optimal Sequence-to-Graph Alignment. In *Research in Computational Molecular Biology: 24th Annual International Conference, RECOMB 2020, Padua, Italy, May 10–13, 2020, Proceedings*, Vol. 12074. Springer, 104.

[31] Mikko Rautiainen and Tobias Marschall. 2020. GraphAligner: rapid and versatile sequence-to-graph alignment. *Genome Biology* 21, 1 (2020), 1–28.

[32] Rachel M Sherman and Steven L Salzberg. 2020. Pan-genomics in the human genome era. *Nature Reviews Genetics* 21, 4 (2020), 243–254.

[33] Jouni Sirén. 2017. Indexing variation graphs. In *2017 Proceedings of the ninteenth workshop on algorithm engineering and experiments (ALENEX)*. SIAM, 13–27.

[34] Jouni Sirén, Erik Garrison, Adam M Novak, Benedict Paten, and Richard Durbin. 2020. Haplotype-aware graph indexes. *Bioinformatics* 36, 2 (2020), 400–407.

[35] Jouni Sirén, Niko Välimäki, and Veli Mäkinen. 2014. Indexing graphs for path queries with applications in genome research. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 11, 2 (2014), 375–388.

[36] Ravi Vijaya Satya, Nela Zavaljevski, and Jaques Reifman. 2012. A new strategy to reduce allelic bias in RNA-Seq readmapping. *Nucleic acids research* 40, 16 (2012), e127–e127.

[37] Laurence A. Wolsey. 1982. An analysis of the greedy algorithm for the submodular set covering problem. *Comb.* 2, 4 (1982), 385–393. https://doi.org/10.1007/BF02579435

[38] David Zuckerman. 2007. Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. *Theory Comput.* 3, 1 (2007), 103–128. https://doi.org/10.4086/toc.2007.v003a006