CHOP: Haplotype-aware path indexing in population graphs

Tom Mokveld[1,†,*], Jasper Linthorst[1,3,†,*], Zaid Al-Ars[2], and Marcel Reinders[1]

1 Delft Bioinformatics Lab, Delft University of Technology, Delft, The Netherlands

2 Computer Engineering, Delft University of Technology, Delft, The Netherlands

3 Department of Clinical Genetics, VU University Medical Center, Amsterdam, The Netherlands

* T.O.Mokveld@tudelft.nl or jasper.linthorst@gmail.com

† Authors contributed equally.

CHOP is freely available at https://github.com/tomokveld/CHOP

Abstract

Motivation: Although the characterization of within species genomic diversity continues to increase, this information is usually not incorporated in the sequencing analysis process. Existing reference genomes can easily be converted to graph-based reference genomes by extending them with known sequence variations. However, the practical use of these graph-based reference genomes depends on the ability to align reads to them. Performing substring queries to paths through these graphs lies at the core of this task. The combination of increasing pattern length and encoded variations inevitably leads to a combinatorial explosion of the search space. We present a solution that uses haplotype information to prevent this from happening.

Results: We present CHOP, a method that uses haplotype information to retrieve the constrained sequence search space of a graph-based reference genome. Our approach enables typical sequence aligners to perform read alignments to graphs that store any type of variation. CHOP performs similarly to another graph alignment method that unnecessarily indexes all combinations of sequence, while aligning reads to a population reference graph of *Mycobacterium tuberculosis* genomes. CHOP can achieve increased sensitivity for variation detection by iteratively integrating variation into a graph. Finally, we show that CHOP can be applied to large and complex datasets, by applying it on a graph-based representation of chromosome 6 of the human genome encoding the variants reported by the 1000 Genomes project.

1 Introduction

Pangenomes and their graphical representations have become widespread in the domain of sequencing analysis [1]. Part of this adoption is driven by the increased characterization of within species genomic diversity. For instance, more recent versions of the human reference genome (GRCh37 and up), include sequences that represent highly polymorphic regions in the human population [2].

A pangenome can be constructed by integrating known variants in the linear reference genome. This way, a pangenome can incorporate sequence diversity in ways that a typical linear reference genome cannot. For example the alignment of reads to a linear reference genome can lead to an over-representation of the reference allele. This effect, known as reference allele bias, influences in particular highly polymorphic regions and/or regions that are absent from the reference [3,4]. By integrating variants into the alignment process, this bias can be reduced [5–7]. As a consequence variant calling can be improved, with fewer erroneous variants induced by misalignments around indels, and fewer missed variants [8].

An intuitive representation for pangenomes are graph data structures, which are often referred to as population graphs [1,9]. Population graphs can be understood as compressed representations of multiple genomes, with sequence generally represented on the nodes. These nodes are in turn connected by directed edges, such that the full sequence of any genome used to construct the graph can be determined by a specific path traversal through the graph. Alternatively, an arbitrary traversal through the graph will yield a mixture of genomes.

A key application for reference genomes is read alignment. Most of the linear reference read aligners follow a seed-and-extent paradigm, wherein exact matching substrings (seeds) between the read and a reference are used to constrain a local alignment. To efficiently search for exactly matching substrings (seeding), indexing data structures are used. The construction of these indexes generally relies on one of two methods: k-mer-based indexing, where all substrings of length k are stored in a hash-map along with their positions within the sequence; and sorting-based methods such as the Burrows-Wheeler Transform (BWT), where the reference sequence is transformed into a self-index that supports the lookup of exact-matching substrings of arbitrary length.

Existing indexing methods can be extended to population graphs, although this is challenging. Graphs can encode a variable number of genomes, however, this comes with an exponential growth of the number of paths in the graph as more variation is integrated. Arbitrary length sequence indexing is therefore not feasible, and indexing must be limited to shorter substrings to minimize combinatorial growth of the index. Additionally, sorting-based indexing methods that rely on suffix determination and ordering, are

often unfeasible in graphs since there will be multiple valid node orderings.

Several approaches have been developed that perform read alignment onto population graphs using indexes that report all k-length paths in the graph. Early examples of this include: GenomeMapper [10], which builds a k-mer hash-map to lookup seeds and subsequently align reads, using banded dynamic programming; and BWBBLE [11], which linearizes the population graph using IUPAC encoding for SNPs and describes indels with flanking sequences as alternate contigs, after which it applies the BWT for indexing. These methods are, however, orders of magnitude slower than linear reference genome aligners, or restricted to only small genomes. For instance, BWBBLE computes four times more suffix array intervals due to the expanded IUPAC alphabet. Furthermore, both of these methods suffer from exponential growth in index space when variation density increases.

Increased scalability of population graph alignment has recently been demonstrated with Graphtyper [12] and the variation graph toolkit vg [13]. Graphtyper does so by first aligning reads to a linear reference sequence using BWA, after which a graph-based alignment is performed on a much smaller set of unaligned or partially aligned reads. For this graph alignment it uses a k-mer hash-map of the population graph, wherein exponential growth in variation-dense regions is reduced, at the loss of sensitivity, through the removal of k-mers that overlap too many alternative sequences. The vg toolkit provides general solutions for working with population graphs. To efficiently query substrings, it utilizes GCSA2 indexing [14], an extension of the BWT for population graphs, which supports exact query lengths of up to 256 bp. Reads are aligned to graphs using a seed-and-extend strategy, returning subgraphs of the population graph to which reads are subsequently aligned using partial order alignment, a generalization of pairwise sequence alignment for directed acyclic graphs [15].

While Graphtyper and vg index all possible paths in a population graph. Such indexing is difficult in complex regions where variation is dense. To deal with this, both methods utilize heuristics to prevent exponential growth. Either by removing k-mers that cross over more than a predefined number of edges, or by masking subgraphs shorter than a set number of bases. Techniques like these prevent exponential growth, but can completely remove complex regions in the graph, resulting in a loss of sensitivity in alignment. Furthermore, they contradict one of the main aims of population graphs, namely to address sequence variation in regions that are inaccessible through the application of a single reference sequence. An alternative solution that does not exclude complex regions, would be to constrain indexing by haplotype, such that only k-mers that are observed in the linear genomes are encoded in the graph.

To expand upon previously developed graph indexing methods, we present CHOP. A path indexer for graphs that incorporates haplotype-level information to constrain the resulting index (Figure 1). CHOP decomposes the graph into a set of linear sequences, so that reads can be aligned by established methods, such as BWA or Bowtie2 [16,17], followed by typical downstream analysis. We show that CHOP performs comparable to vg and scales to human genomes with variation data of the 1000 Genomes Project [18,19].

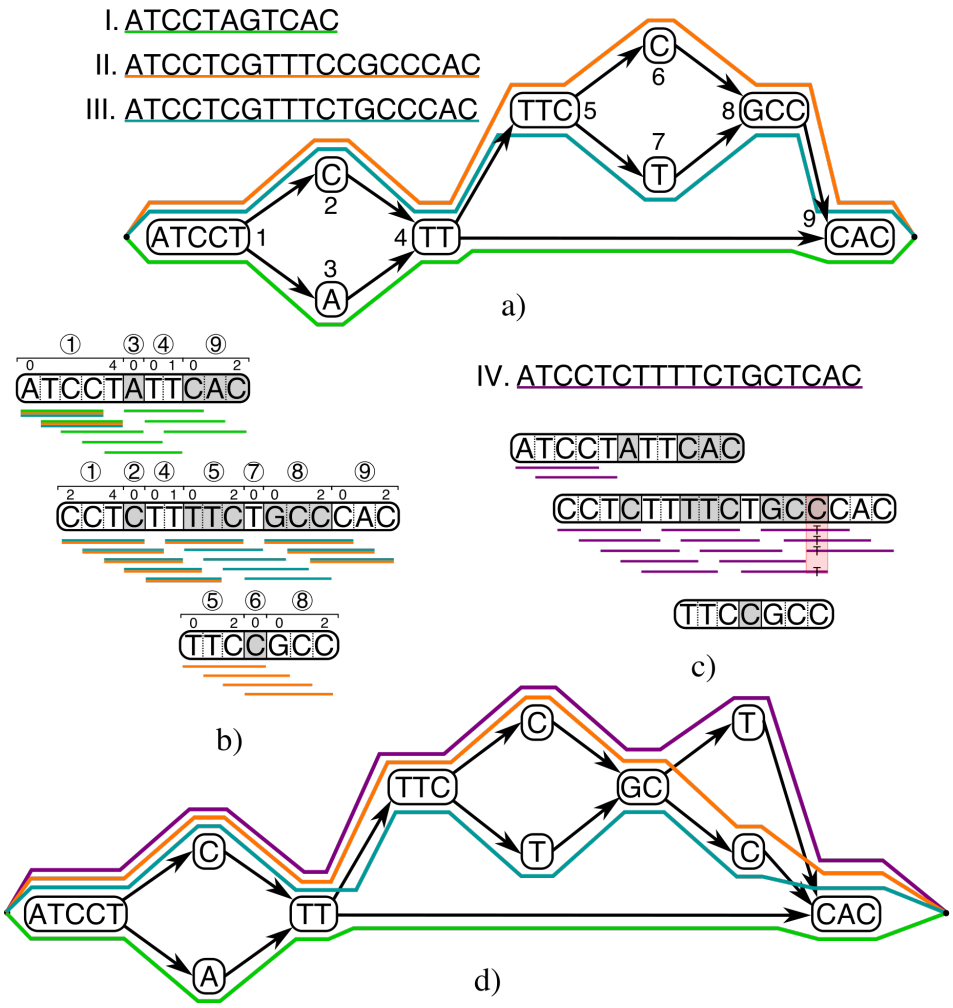

Figure 1. Schematic overview of how CHOP can be used to align reads to a population graph. a) The input for CHOP is a graph representation of three different haplotypes (I, II, III). Colored paths through the graph identify the underlying haplotypes. b) CHOP decomposes the graph into a null graph with respect to $k = 4$. The obtained null graph contains three nodes, the sequence that is defined on these nodes covers all substrings of length 4 that occur in the haplotypes encoded in the graph. On top of each node are annotations that refer to intervals within nodes of the input graph. c) Reads of length k from a new haplotype (IV) can be aligned to the null graph. A pileup of mismatches can be observed in the aligned reads. d) Through the attached interval definition, the new variant can be positioned on node 8 offset 2 in the original graph.

2 Methods

2.1 Population graph definition

Population graphs were constructed given existing reference genomes and sets of variations, called from reference alignments (Supplementary information S1.1). Nodes within the graph are labeled, encoding both genomic and variable sequences, which are in turn connected by directed edges. Traversing a set of edges, that is a path, will either describe a mixture of haplotypes, or a haplotype within the graph.

Edges are encoded with sample-level haplotyping that report the sample observed paths in the graphs (Figure 1a). Each node in the graph stores an interval definition, that tracks sequence origin, as such that it can be tracked while transforming a graph (Figure 1b). To allow for efficient point querying and merging of intervals a tree-based data structure is used.

2.2 Population graph specification

The graph $G = (V, E)$ is defined as a set of nodes $V = \{v_0, \ldots, v_N\}$, where $N = |V|$, and a set of edges E, where $N - 1 \leq |E| \leq \frac{(N-1)N}{2}$. Each of these edges is an ordered pair of nodes $(u, v) \in E$, where node $u \in V$ is connected to node $v \in V$. As G is a directed graph, for any edge $(u, v) \in E$, holds that $(u, v) \neq (v, u)$.

For each node $v \in V$, the in-degree, $in\,(v)$, is defined as the number of incoming edges to that node; i.e. the number of distinct edges $(u, v) \in E$ for any $u \in V$. Conversely the out-degree of node, v, $out\,(v)$, is defined as the number of outgoing edges from that node.

In a population graph, a sequence of characters, S, consisting of the alphabet $\Sigma = \{A, T, C, G\}$ is assigned to every node, v, such that $v_S = S\,[0,\ n-1]$, wherein $S\,[i] \in \Sigma$ for all i, and the length, n, is defined as $n = |v_S|$. The range of any such sequence for every node, $v \in V$, lies between $1 \leq |v_S| \leq N$, where N is the length of the largest recorded sequence. Every subsequence of a sequence, S, is denoted as $S\,[i,\ j]$. Two types of subsequences in particular are prefixes $S[0,\ j]$ and suffixes $S[i,\ n-1]$, which describe the left and right flanks of any sequence S respectively.

A path, P, where $P = u_0 \cdots u_{|P|-1}$, is any consecutive series of nodes, $(u_i,\ u_{i+1}) \in E$ for all $i < |P|$. If a path exists between any pair of nodes in a graph, it is a connected graph, i.e. there are no unreachable nodes. The sequence, S, of a path, P_S, is the concatenation of sequences contained in the nodes, such that $P_S = u_{0S} \cdots u_{|P|-1S}$.

Provided with haplotyping data the graph G is embedded with a set of haplotypes, H, where $H = \{H_0, \ldots, H_{|H|-1}\}$. Each edge (u, v) is assigned a set of H to describe the haplotypes that pass through G, in effect reporting haplotyped paths. Each encoded haplotype can be represented by a path traversal through G, and may overlap other haplotypes.

Let G^E denote the null graph of G such that $G^E = (V, \emptyset)$, sharing all previously defined node properties of G. The interval, I, noted as u_I of any node $u \in V$ in G^E is defined as the mapping of contained nodes in u_I to their respective subsequences in the sequence $u_S = u_{I[0]_{S[i,\ n]}} \ldots u_{I[1]_{S[0,\ n]}} \ldots u_{I[|I|]_{S[0,\ j]}}$, relating the subsequences in node u to their nodes of origin in G.

2.3 Population graph path indexing

The purpose of indexing a population graph is to allow for efficient substring queries on the paths that span across nodes and edges of the graph (Figure 2). Given any non-trivial sized graph, enumerating all possible paths is often unfeasible, given the exponential nature of traversing all combinations of nodes and edges.

CHOP constrains queries through a graph to be part of a haplotype with which the population graph was built. Hereto, CHOP transforms graph G into a null graph G^E such that every node in G^E represents a sequence of length k or longer, and that every substring of length k originating from the encoded haplotypes in G is also a substring in a node of G^E. Meaning that if sequencing reads are true error-free samplings of an underlying haplotype and are of the same length (or shorter) than the chosen value of k, they should correspond to a substring of a node in G^E. This, in turn, enables the application of any existing read aligner to place reads onto G^E. Through this transformation of G to G^E, all haplotyped paths of at least length k in the graph are accounted for. The transformation is driven by three operators: collapse, extend, and duplicate, explained throughout the rest of this section.

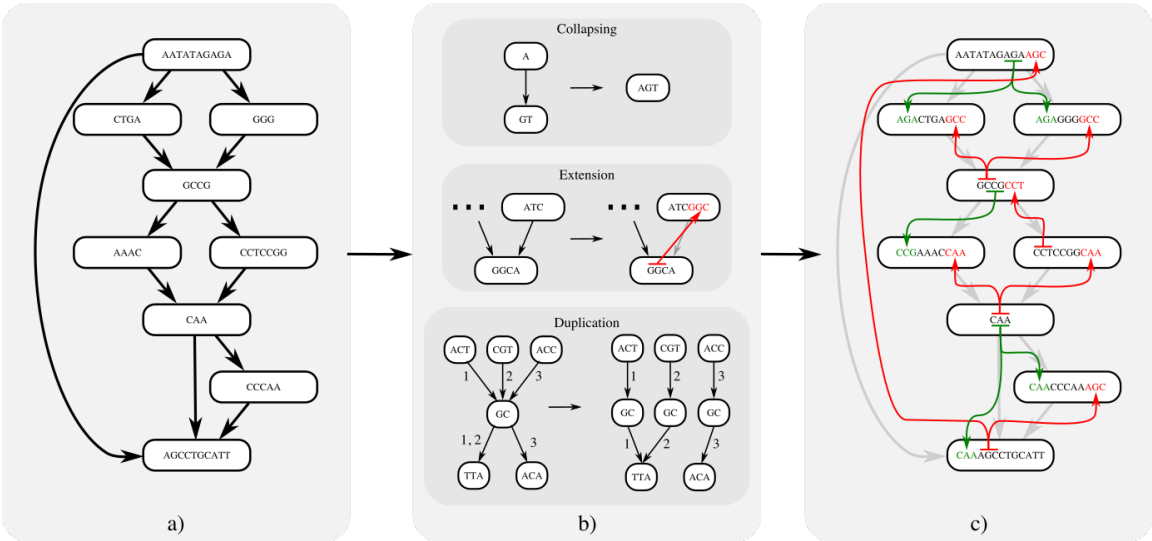

Figure 2. Reporting the haplotyped k-paths in the population graph G transforms it into the null graph G^E, here $k = 4$. a) Population graph with sequence encodings on the nodes. b) Indexing of k-paths based on three operations. c) The null graph encodes all 4-length paths in the original graph, coloring of lines and text denote the origin of assigned prefixes (green) and suffixes (red).

```
procedure ChopGraph(G)
    SimplifyGraph(G)                                  ▷ Extend and Collapse until exhaustion
    if G_E ≠ ∅ then
        for each edge (u, v) ∈ G_E do
            if deg(u) > deg(v) then
                Duplicate(G, u)                       ▷ Duplicate u
            else
                Duplicate(G, v)                       ▷ Duplicate v
        ChopGraph(G)
```

```
procedure SimplifyGraph(G)
    modified ← True
    while modified do
        modified ← False
        for each edge (u, v) ∈ G_E do
            if out(u) = 1 and in(v) = 1 then
                Collapse(G, u, v)                     ▷ Collapse u||v
                modified ← True
            else if in(v) = 1 and |u_S| ≥ k − 1 then
                Extend(G, u, v, 1)                    ▷ Prefix u ↠ v
                modified ← True
            else if out(u) = 1 and |v_S| ≥ k − 1 then
                Extend(G, u, v, 0)                    ▷ Suffix v ↞ u
                modified ← True
```

Collapse

If an edge $(u,\ v) \in E$ confirms to $out(u) = 1$ and $in(v) = 1$, then any path that traverses u, will immediately be followed by v. Therefore, it can be considered a redundant traversal such that the sequence on u and v can be merged without affecting the number of sequences that can be spelled by the graph. To do so, sequence and the corresponding intervals of u and v are merged after which the outgoing edges of v are transfered to u, followed by the removal of v and the edge $(u,\ v)$. We denote this operation as collapsing, defined as $u||v$ for any edge $(u,\ v)$ (Figure 2b). The direction of collapsing is guided by minimizing the number of edge reassignments, such that when $in\,(u) > out\,(v)$, v is collapsed into u, joining the sequence $u_S = u_S \cdots v_S$. Alternatively u is collapsed into v, joining the sequence $v_S = u_S \cdots v_S$.

```
procedure COLLAPSE(G, u, v)
    if in(u) > out(v) then                          ▷ u ← v
        u_S = u_S ··· v_S                           ▷ Concatenate sequence
        for each edge (v, x) ∈ G_E do               ▷ Outgoing edges u
            add edge (u, x)
        delete node v
    else                                            ▷ u → v
        v_S = u_S ··· v_S                           ▷ Concatenate sequence
        for each edge (x, u) ∈ G_E do               ▷ Incoming edges v
            add edge (x, v)
        delete node u
```

Extend

After collapsing redundant edges in the graph, part of the remaining k-length substrings can be accounted for with the extend operation. Extend is based on the observation that all k-length substrings that span a single edge (u, v) can be accounted for by extending the $k-1$-length suffix of u as a prefix of v, denoted as $u \twoheadrightarrow v$, provided that $in\,(v) = 1$ and $|u_S| \geq k-1$. Or vice versa, extending the $k-1$ length prefix of v as a suffix of u, denoted as $v \twoheadleftarrow u$, provided that $out\,(u) = 1$ and $|v_S| \geq k-1$. This way, the suffix of u is ambiguously assigned to v (or vice versa). However, as the ambiguity concerns a $k-1$ length prefix or suffix, any substring of length k that is sampled from the underlying haplotypes will exclusively correspond to either the sequence in u or the prefixed sequence in v (or vice versa). In other words, by extending edges in G we introduce overlapping sequence as if we were converting G to the repeat-resolved string graph representation of a joint assembly of all genomes in G from all possible reads of length k [20].

```
procedure EXTEND(G, u, v, isPrefix)
    if isPrefix then                                   ▷ u ↠ v
        v_S = u_S [|u_S| − k − 1, |u_S|] ··· v_S
    else                                               ▷ v ↞ u
        u_S = u_S ··· v_S [0, k − 1]
    delete edge (u, v)
```

Given the subgraph in Figure 3, the nodes u and v both encode sufficient sequence to enable the extend operation to report k-length overlap between the pair and resolve edge (u, v).

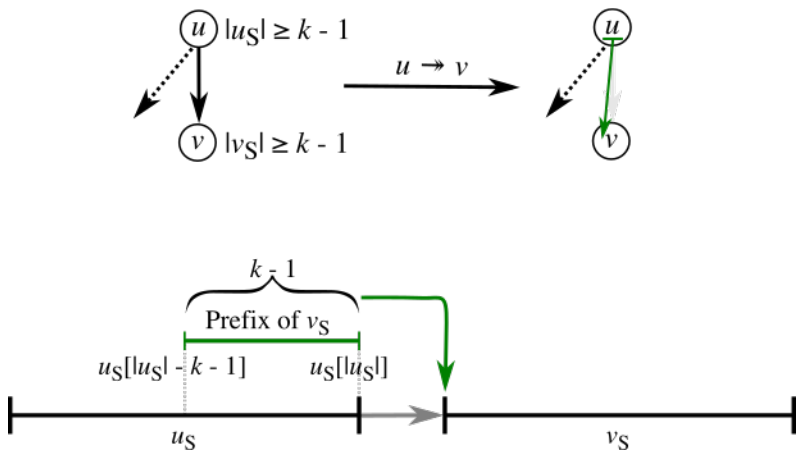

Figure 3. A pair of nodes u and v where $|u_S| \geq k-1$ and $|v_S| \geq k-1$, and overlap can only be resolved by $u \twoheadrightarrow v$, defined as $v_S = u_S\,[|u_S| - k - 1, |u_S|] \cdots v_S$, given that $out\,(u) > 1$.

In Figure 4, a subgraph is shown in which extension is only possible between the edges (u, w) and (w, v). This does not apply for (u, v), as $out\,(u) > 1$ and $in\,(v) > 1$. However, once (u, w) and (w, v) are resolved, (u, v) is resolved by a subsequent iteration of a collapse while maintaining a representation that covers all k-length substrings. The order in which subsequences are extended may result in different null graphs, but any of these will cover all k-length substrings.

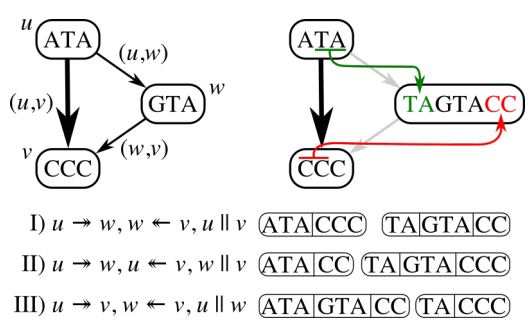

Figure 4. Subgraph in which subsequence extension between (u, v) is not allowed unless either (u, w) or (w, v) are resolved first. Three different solutions are equivalent to resolve this subgraph, every solution is equal in k-path space.

Duplicate

If neither collapse nor extend can be applied to any of the remaining edges in the graph (Figure 2b), graph connectivity must be reduced. This is enabled by the duplicate operation, which duplicates and reconnects nodes, while maintaining the haplotypes that are encoded in the graph. Duplication allows consequent collapsing, which in turn enables subsequence extension, such that after a sufficient number of iterations all edges in G can be removed. As opposed to methods that aim to track all possible paths through the graph, we suggest the use of haplotype information that is modeled on the edges to constrain the number of necessary node duplications from $in\,(u) * out\,(u)$ to δ. Where δ is the number of paired incoming and outgoing edges for u that have at least one intersecting haplotype. Note that δ is bounded by the number of haplotypes encoded in G, and that there will never be more duplications than haplotypes in any one region of the graph.

To illustrate this idea, Figure 5 shows a subgraph with haplotype encoding on the edges. From the haplotyping it can be derived that not all paths through this graph are actually supported by the underlying haplotypes. For example, the path $u \rightarrow d \rightarrow f$ combines sequence segments that are unsupported (the haplotypes between (u, d) and (d, f) do not overlap). By excluding these unsupported paths through the graph, the number of duplications for node d can be constrained from 6 to 3. This way, the search space for subsequent k-length substrings is greatly reduced with respect to reporting all possible paths, which in turn enables searching for longer subsequences.

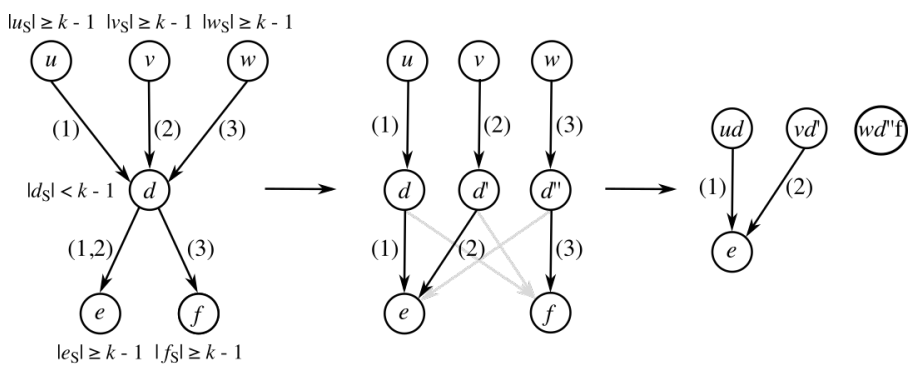

Figure 5. Subgraph with haplotypes: $\{1, 2, 3\}$. Node d must be duplicated, as no more edges can be removed through extension or collapsing without introducing ambiguity. By grouping incoming and outgoing haplotypes on d, the number of duplications can be reduced. In the resulting graph, edges (u, d), (v, d'), and (w, d'') can be collapsed. Finally, an extension can be applied to edges (ud, e) and (vd', e) which would lead to the null graph. Note that the introduction of grayed out edges is prevented using haplotyping, hence the edge count is reduced from 6 to 3.

```
procedure DUPLICATE(G, u)
    for each edge (p, u) ∈ G_E predecessors (u) do
        for each edge (u, s) ∈ G_E successors (u) do
            group ← (x, u)_H ∩ (u, x)_H
            if group ≠ ∅ then
                create node i                                   ▷ i ← u
                create edges[(p, i), (i, s)]
    delete node u
```

2.4 Mapping reads to CHOP's edgeless graph

When reads are shorter than a chosen value for k, established alignment tools can be used to directly align reads to the null graph representation. As the sequence that is modeled on the nodes in G^E is now a composition of sequence originating from adjacent nodes in G, the intervals that gave rise to these compositions need to be traced in order to convert the alignment of a read to a node in G^E to a path in G. For this reason, during the transformation from G to G^E, the originating node in G and corresponding offset for each prefixed, suffixed or concatenated sequence is stored alongside the actual sequence. Note that in theory the defined operations can also be expressed purely in terms of interval operations, excluding any sequence. Given the intervals, a mapping between G^E and G is maintained, such that any node in G^E can be traced back to the corresponding path of nodes in G. As a result, any alignment to a node in G^E can also be traced to a sub-path of this path, effectively enabling the alignment of a read to graph G by using G^E as a proxy (Figure 1c).

3 Results

3.1 Evaluation graph alignment

To evaluate CHOP and its application in population graph alignment using BWA-0.7.15-MEM [16], we performed tests on *Mycobacterium Tuberculosis* (TB). TB represents a good model for population graphs, given the high accuracy of available assemblies, its tractable genome size (4.4 Mb), and the limited degree of variation. We obtained 401 variant call sets from different TB strains (samples) in the KRITH1 and KRITH2 datasets. Variants were called with respect to the reference genome H37Rv, using Pilon [21–23], and were filtered to exclude low quality variants. For graph construction we employed a leave-one-out strategy, wherein one sample was removed from the VCF file containing all 401 samples. The read set of the removed sample was subsequently used for graph alignment. This was repeated with 10 randomly selected samples. Corresponding read sets were obtained from EBI-ENA (Supplemental section S1.2). To

investigate how incorporating more variation influences graph alignment, we progressively incorporated more variation (from the complete set) into the constructed graphs. With up to ˜17,500 variants in the 400 sample graph (rate of variation growth is shown in Supplementary Figure 11).

As the ground truth of genomic positions in the read data are unknown, we evaluate alignments based on the following criteria: number of mismatches, insertions, deletions, clipped bases, unaligned reads, and perfectly aligned reads (Definitions are given in Supplemental section S1.4). These criteria allow us to inspect the behavior of different read aligners. To avoid bias induced by multiple possible alignments for a single read, only primary alignments were considered.

To evaluate our haplotype-based approach, we compared CHOP to vg-1.6.0. The vg toolkit provides general solutions for population graphs which include graph construction, indexing, and read alignments. However, contrary to CHOP's approach, vg indexes all k-length paths in a graph. We set CHOP to report 101-length haplotype paths (equivalent to the read length), and used default parameters with BWA-MEM. Vg was set to index all 104-length paths ($k = 13$, 3 doubling steps), to most closely reflect the settings of CHOP.

Different algorithms are used to perform read alignments in BWA and vg. To address aligner and parameter induced differences, we first summarized the results of the 10 hold-out samples on the linear reference genome, shown in Table 1 for BWA and vg. Both aligners result in the same number of perfectly aligned reads. However, alignments with vg result in fewer unaligned reads (-14.91%) and more mismatches (1.93%) than BWA. We attribute this difference to an increase in sensitivity by which vg aligns reads. This is reflected by the increase in clipped bases (+14.87%), inserted bases (+28.60%), and deleted bases (+33.38%), which allows vg to map shorter read fragments.

Using these measurements as a baseline, read to graph alignments were compared between CHOP/BWA and vg. The graph construction of CHOP and vg was found to have minimal effect on alignments as shown in Supplemental Figures 12 and 13. Figure 6 shows the increase in perfectly aligned reads using both vg and CHOP/BWA, as more samples are incorporated into the graph (similar plots for the number of unaligned reads and mismatches can be found in Supplemental Figures 14 and 15). Table 1 shows the alignment results for the TB graph with 400 samples.

Table 1. Mean of alignment results across all 10 hold-out sample alignments to 1) the reference genome H37Rv (H37Rv columns) and 2) the 400 TB genomes graph (Graph columns) for both CHOP/BWA and vg to align the reads (note that when aligning to H37Rv, CHOP is not used). Definitions of the alignment criteria are given in Supplemental section S1.4.

Alignment criteria	All hold-out samples - Read length = 101			
	BWA	CHOP/BWA	vg	
	H37RV	Graph (n=400)	H37RV	Graph (n=400)
Reads aligned	6,160,920	6,162,033 (+0.018%)	6,214,636	6,218,123 (+0.056%)
Reads unaligned	360,236	359,123 (-0.309%)	306,520	303,033 (-1.138%)
Reads perfectly aligned	4,048,774	4,142,052 (+2.304%)	4,048,774	4,142,388 (+2.312%)
Bases aligned	596,380,132	596,611,260 (+0.039%)	598,394,374	598,748,736 (+0.059%)
Bases unaligned	62,191,423	61,960,355 (-0.372%)	60,158,534	59,802,542 (-0.592%)
Bases unaligned from clipped reads	22,349,569	22,380,472 (+0.138%)	25,673,865	25,804,701 (+0.510%)
Bases mismatched	3,458,029	3,308,480 (-4.325%)	3,526,108	3,391,539 (-3.816%)
Bases inserted	65,210	65,151 (-0.090%)	83,859	85,488 (+1.942%)
Bases deleted	52,272	51,165 (-2.118%)	69,718	71,462 (+2.501%)
Non-primary alignments	246,092	246,540 (+0.182%)	624,664	641,092 (+2.630%)

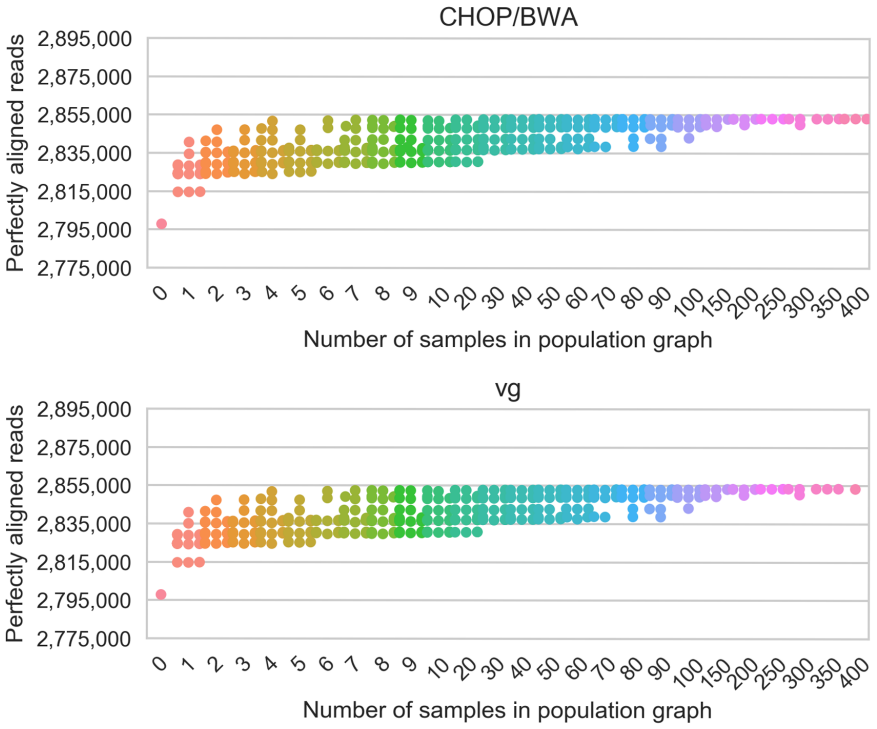

Figure 6. Perfectly aligned read count for SRR833154 alignments to different sized population graphs, containing between 0 (only H37Rv the linear reference) and 400 samples for both, when using CHOP/BWA and vg to align reads to the graph. Coloring is used to separate measurements of sampled graphs.

Figure 6 shows that incorporating more variation from samples into the population graphs increases the number of aligned bases, which is further demonstrated in Supplemental Figures 14 and 15. The spread is a consequence of sampling, where samples that are closely related to the hold-out sample will give a greater improvement than distantly related samples, which is further motivated by the reduction of spread as sampling size increases.

The baseline alignments to H37Rv already highlighted that the aligners perform differently. However, throughout the course of the experiments, nearly all alignment criteria show the same trend for both CHOP/BWA and vg. The exception being the number of unaligned reads, which steadily decreases with vg, while this is not as pronounced using CHOP/BWA.

3.2 CHOP scales to *Homo sapiens*

To evaluate the scalability of CHOP, we used chromosome 6 (170 Mb) of the human genome (GRC37) and the full unfiltered 1000 Genomes Phase 3 variation dataset [18,19]. The constructed graph has 14,744,119 nodes and 19,770,411 edges and encodes a total of 5,023,970 variants (4,800,102 SNPs, 97,923 insertions, and 125,945 deletions). Note that the variation set includes diploid phasing of 2,504 individuals, which is incorporated into the graph as 5,008 paths (2 paths per sample), plus one path that represents the reference genome. Within the population most variation (58.42%) is shared between at least two or more samples (Supplemental Figure 16). For the graph alignments, 15 read sets were used from the 1000 Genomes project (Supplemental Table 4), filtered to include only reads aligned to chromosome 6 or that could not be aligned anywhere on the genome (average read set size is 3,026,069). CHOP was set to report 100-length paths through the graph to match the read length, which yielded 11,359,686 paths. We attempted indexing with vg for paths up to 104 bp ($k = 13$, 3 doubling steps), but this resulted in a memory size limit. Instead, doubling was lowered to 2, and paths up to 52 bp were indexed. The read sets were aligned to both the linear reference of chromosome 6 and the graph representation, using both CHOP/BWA and vg, which is summarized in Table 2.

Table 2. Mean of alignment results from 15 samples from the 1000 Genomes data when aligning to 1) the reference genome sequence of chromosome 6 (column GRC37), and 2) the population graph created from the 5,008 haplotypes, for both CHOP/BWA and vg.

Alignment criteria	1000 Genomes samples - Read length = 100			
	BWA	CHOP/BWA	vg	
	GRC37	Graph (n=2504)	GRC37	Graph (n=2504)
Reads aligned	2,542,399	2,543,522 (+0.044%)	2,632,860	2,664,392 (+1.198%)
Reads unaligned	483,670	**482,548 (-0.232%)**	393,209	**361,677 (-8.019%)**
Reads perfectly aligned	1,794,564	1,977,952 (+10.219%)	1,794,537	1,978,644 (+10.257%)
Bases aligned	251,122,992	251,516,725 (+0.157%)	253,395,504	254,561,621 (+0.460%)
Bases unaligned	51,439,949	51,070,534 (-0.718%)	49,154,474	48,007,515 (-2.333%)
Bases unaligned from clipped reads	1,801,947	**1,846,687 (+2.483%)**	8,650,712	**10,891,432 (+25.902%)**
Bases mismatched	1,270,981	969,087 (-23.753%)	1,182,855	948,364 (-19.824%)
Bases inserted	43,979	19,661 (-55.296%)	56,942	37,784 (-33.646%)
Bases deleted	61,659	32,355 (-47.526%)	72,189	47,016 (-34.871%)
Non-primary alignments	N.A	N.A	N.A	N.A
Time (seconds)	544	1,807	14,550	31,333

The same improvement of moving to a graph representation can be observed here as in TB, although more extensive (from 0.039% to 0.157% more aligned bases in CHOP/BWA), given that more variants, including indels, are incorporated into the graph.

Read alignment to graphs using CHOP/BWA takes approximately 3 times longer than linear alignments. Given the different path lengths used, time cannot directly be compared between CHOP/BWA and vg. It is unclear why vg took substantially more time to align.

We observed substantial differences between CHOP/BWA and vg with respect to the decrease of unaligned reads -0.232% versus -8.019%, and the increase in read clipping +2.463% versus 25.902%, again this was similar but not as extensive as with the TB data. To evaluate this difference, all reads that aligned exclusively onto the graph were extracted, which amounted to 21,661 reads in CHOP/BWA and 473,836 in vg. Figure 7 shows the distribution of the number of aligned bases for each of those reads.

Nearly all (93.72%) of the newly aligned reads by vg had a length of between 16 to 30 bases, either induced by clipping or extensive base insertion/deletion. However, from 30 bases and up, the aligners display very similar profiles, and comparable numbers of newly aligned reads (29,779 for vg). It is unclear what caused the peak at 70 bp in both of the aligners.

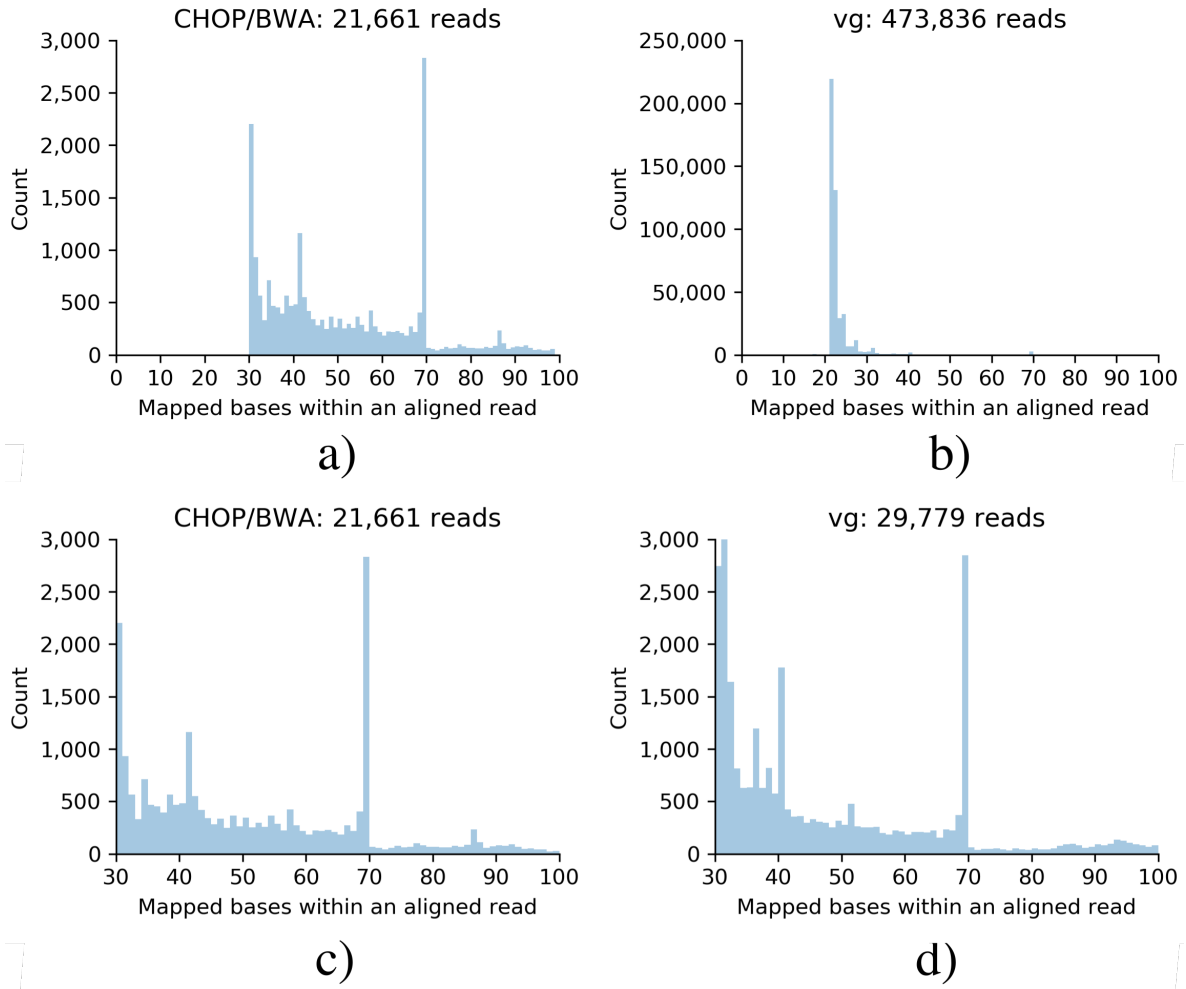

Figure 7. The number of reads that have a particular number of bases aligned after their alignment onto the chromosome 6 population graph with CHOP/BWA (a) and vg (b), respectively. In c) and d) the same is shown in the range of 30 to 100 bases.

3.3 Variation integration

As graphs offer a greater search space, we investigated how this could affect read alignment and subsequent variant calling. By encoding more distinct sequence in a graph it should be possible to align more reads, and potentially allow new variants to be called. To evaluate this effect, variants were integrated using a feedback loop. First SRR833154 reads were aligned to H37Rv using BWA, and variants were called using Pilon. These variants were quality filtered down to 838 SNPs and used to construct a graph with H37Rv (now thus including two genomes). The same set of reads was then aligned onto the graph, and variants were called. We expected that the additional context offered by the graph would point to previously undiscovered variants. An example of this is schematically shown in Figure 8a. Integrating variants in a graph (Figure 8b) and realigning reads to the graph allowed reads to follow a path within

the graph that best matches. This in turn allowed for reads that previously were not able to align now to be aligned. (Figure 8c). As a result, 19 (+2,26%) new high quality variants could be called from these new aligned reads. The effect where information of the same sample can result in more variant calls could be further explored by performing multiple rounds of variant integration, to determine the extent of this improvement.

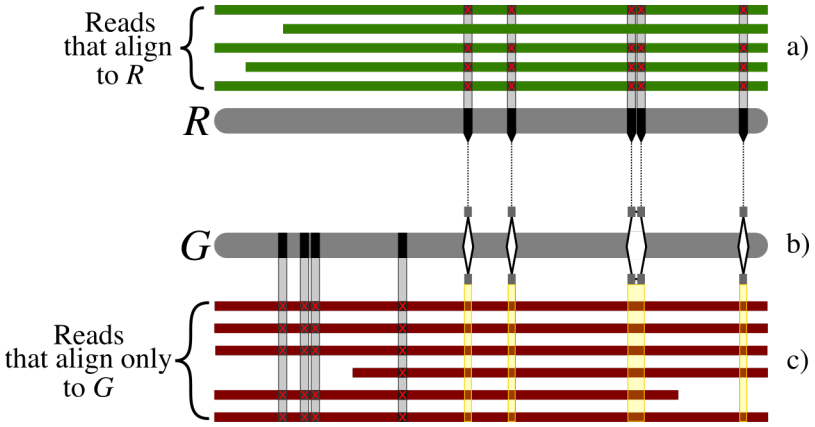

Figure 8. a) Alignment of SRR833154 reads to the reference R, H37Rv, with subsequent variant calling detecting 5 high quality SNPs in this particular region. b) These and all other SNPs across the genome are integrated with the reference into graph G, followed by alignment of the same reads. c) Reads that previously did not align to R, now align onto a haplotype of the graph G. Formation of a pileup allows for the detection of 4 new variants in this region.

4 Discussion

Indexing is one part of the challenge in applying population graphs for sequencing analysis, where deciding what should be indexed can already have a large impact. By indexing all k-paths, not only will all sample supported sequences be included, but also all combinations between these sequences. Such combination results in a proportion of indexed paths to never be observed in any sample due to strong variation linkage. Additionally, it can cause a combinatorial explosion during indexing, increasing the search space considerably. This can affect read alignment especially during seeding, and potentially introduces random hits, which challenges the specificity of any resulting alignments. Methods that use heuristics to cope with exponential growth, for instance by limiting edge traversal or removing subgraphs in complex regions of the graph, will reduce alignment sensitivity, especially in regions that are difficult to align to in the first place. By incorporating haplotype encodings in this process, CHOP avoids exponential growth, and retains complex regions within the graph, balancing both sensitivity and specificity of alignments. A haplotyping constraint is not only important to improve graph alignment scalability, but it also allows for longer paths to be indexed.

Since CHOP retrieves the haplotype constrained sequence space of a population graph, standard tools can be used to perform graph alignments and downstream analysis. It is important to note that for the construction of a graph, variant calls need to have phasing. As this information might not always be encoded in typical variant call formats, it is important to mention that this information is only needed at short ranges (same as the value for k) and should therefore be available from typical sequencing experiments.

We showed how an alignment to a graph results in more aligned bases, which grew with the number of variants that were incorporated. In the alignments (using BWA) with TB and human we show comparable performance to vg (after accounting for aligner sensitivity). This suggests that there is little added value in indexing all paths.

The complexity of incorporated variation was limited to a small scale at only low (TB) or high (human) variant abundance. Therefore, the benefit of a population graph on increasing the number of aligned reads was limited. We expect that incorporating larger variations can lead to an increase in mappable reads, however, this may also induce more sequence repetitivity in the graph.

The experimental setup in this paper used for graph alignment will not directly work for long reads, as a value for k as large as the largest read in a typical long read dataset will still lead to an intractable sequence space, even when incorporating the haplotype information. However, long reads are mostly aligned with short seeds in the first place, which can easily be retrieved using CHOP. Subsequently, the resulting set of sequences could function as a lookup for seeding in the graph, and a traditional seed-and-extend approach could be used.

References

1. Benedict Paten et al. Genome graphs and the evolution of genome inference. *Genome research*, 27(5):665–676, 2017.

2. Valerie A Schneider et al. Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome research*, 27(5):849–864, 2017.

3. Jacob F Degner et al. Effect of read-mapping biases on detecting allele-specific expression from RNA-sequencing data. *Bioinformatics*, 25(24):3207–3212, 2009.

4. Débora YC Brandt et al. Mapping bias overestimates reference allele frequencies at the HLA genes in the 1000 genomes project phase I data. *G3: Genes, Genomes, Genetics*, 5(5):931–941, 2015.

5. Heng Li and Richard Durbin. Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics*, 26(5):589–595, Mar 2010.

6. Alexander Dilthey et al. Improved genome inference in the mhc using a population reference graph. *Nature genetics*, 47(6):682–688, 2015.

7. Yu Liu et al. Discovery of common sequences absent in the human reference genome using pooled samples from next generation sequencing. *BMC genomics*, 15(1):685, 2014.

8. Mark A DePristo et al. A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nature genetics*, 43(5):491, 2011.

9. Deanna Church et al. Extending reference assembly models. *Genome biology*, 16(1):13, 2015.

10. Korbinian Schneeberger et al. Simultaneous alignment of short reads against multiple genomes. *Genome Biol*, 10(9):R98, 2009.

11. Lin Huang et al. Short read alignment with populations of genomes. *Bioinformatics*, 29(13):i361–i370, 2013.

12. H. P. Eggertsson et al. Graphtyper enables population-scale genotyping using pangenome graphs. *Nat. Genet.*, Sep 2017.

13. Erik Garrison et al. Sequence variation aware genome references and read mapping with the variation graph toolkit. *bioRxiv*, page 234856, 2017.

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

15. Christopher Lee et al. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464, 2002.

16. Heng Li. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint arXiv:1303.3997*, 2013.

17. Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature methods*, 9(4):357–359, 2012.

18. 1000 Genomes Project Consortium. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–1073, 2010.

19. 1000 Genomes Project Consortium. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65, 2012.

20. Eugene Myers. The fragment assembly string graph. *Bioinformatics*, 21(suppl 2):ii79–ii85, 2005.

21. Bruce J Walker et al. Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. *PloS one*, 9(11):e112963, 2014.

22. Keira Cohen et al. Evolution of extensively drug-resistant tuberculosis over four decades: whole genome sequencing and dating analysis of Mycobacterium tuberculosis isolates from KwaZulu-Natal. *PLoS Med*, 12(9):e1001880, 2015.

23. Abigail L Manson et al. Genomic analysis of globally diverse Mycobacterium tuberculosis strains provides insights into emergence and spread of multidrug resistance. *Nature genetics*, 49(3):395, 2017.