Read Mapping on Genome Variation Graphs

Kavya Vaddadi
TCS Research, Hyderabad, India
kavya.vaddadi@tcs.com

Rajgopal Srinivasan
TCS Research, Hyderabad, India
rajgopal.srinivasan@tcs.com

Naveen Sivadasan
TCS Research, Hyderabad, India
naveen.sivadasan@tcs.com

Abstract

Genome variation graphs are natural candidates to represent a pangenome collection. In such graphs, common subsequences are encoded as vertices and the genomic variations are captured by introducing additional labeled vertices and directed edges. Unlike a linear reference, a reference graph allows a rich representation of the genomic diversities and avoids reference bias. We address the fundamental problem of mapping reads to genome variation graphs. We give a novel mapping algorithm V-MAP for efficient identification of small subgraph of the genome graph for optimal gapped alignment of the read. V-MAP creates space efficient index using locality sensitive minimizer signatures computed using a novel graph winnowing and graph embedding onto metric space for fast and accurate mapping. Experiments involving graph constructed from the 1000 Genomes data and using both real and simulated reads show that V-MAP is fast, memory efficient and can map short reads, as well as PacBio/Nanopore long reads with high accuracy. V-MAP performance was significantly better than the state-of-the-art, especially for long reads.

2012 ACM Subject Classification Computing methodologies → Combinatorial algorithms; Applied computing → Computational genomics

Keywords and phrases read mapping, pangenome, genome variation graphs, locality sensitive hashing

Digital Object Identifier 10.4230/LIPIcs.WABI.2019.7

Acknowledgements Authors would like to thank the anonymous reviewers for their valuable comments. Authors would also like to acknowledge Kshitij Tayal for the initial implementation of the algorithm.

1 Introduction

Conventional genome analysis relies primarily on a linear reference. A pangenome reference collection, on the other hand, is a richer representation of the genomic diversity and suffer less from reference bias [26]. Genome variation graphs are natural candidates for representing the genomic variations in a pangenome collection [26, 28]. In genome variation graphs, common subsequences are encoded as vertices and the genomic variations are captured by additional labeled vertices and directed edges. Such pangenome based representations and analysis are becoming increasingly popular also due to several ongoing large scale population-wide sequencing projects worldwide. As a result, there is an increasing need for developing efficient genome analysis pipelines that can work with graph genomes in place of linear genomes [26, 28]. The non-linear graph structure poses additional challenges to even the fundamental problems of sequence alignment and read mapping [28].

In this work, we consider the problem of efficiently mapping reads to genome variation graphs. Performing gapped alignment of reads to the whole graph is prohibitively expensive due to the large graph size and its complex structure. To achieve high throughput, the mapping algorithms have to restrict the expensive gapped alignment to only small regions of the reference graph where the read can align optimally. Several high throughput mapping tools are available for linear references [20, 21, 22, 24, 4]. Alignment based mappers use techniques such as Burrows-Wheeler Transforms [3, 9, 11, 23, 20, 17] to construct a search index of the reference along with backtracking to compute target alignment. Seed-and-extend approaches [2, 8, 5] use seeds or short read fragments to identify the target region and extend this to compute the final gapped alignment. Hashing based approaches [14, 21, 22] create hash signatures of the reference for fast identification of the target region. In particular, using locality sensitive hash functions (LSH) such as minhash have gained popularity due to their robustness to small read errors and gaps [14]. For instance, Mashmap [14] combines LSH with sequence winnowing technique [30] to build space efficient index with probabilistic guarantees on the identified target regions for long reads. Some of the tools that are optimized for short reads are not well suited for long reads due to higher error rates and longer read length. BWA-MEM [20], GraphMap [34], MashMap [14], Minimap2 [22] are a few of the mappers that provide long read support.

Linear reference mappers can be extended to a pangenome by treating the pangenome as a collection of linear sequences [25, 37, 12, 36]. One drawback of this approach is the increasing space/time requirements with increasing number of sequences. A unified representation, on the other hand, is significantly more compact and generalizes well to the space of possible variations and recombinations. GenomeMapper[31], MuGI [7], RCSI [37], BWBBLE [12] are some of the recent examples of mapping short reads to genome collection. They work based on the seed-and-extend paradigm with restrictions on the read length and gaps.

Similar to the linear reference setting, for genome graphs, the mapping algorithm has to restrict the costly alignment to small regions of the graph where the read is likely to align optimally to achieve high throughput. The Variation Graph (VG) project provides a state-of-the-art read mapping tool VG Mapper (VG) [10] on genome variation graphs. VG can handle directed acyclic graphs as well as variation graphs with cycles. VG considers the de Bruijn graph associated with the input graph and constructs an exact search index of the de Bruijn graph using Burrows-Wheeler extensions (GCSA2) for graphs [33, 32]. The index is used to identify hits on the graph based on the input read. These hits are then clustered and the final alignment is performed on the subgraph corresponding to the largest cluster. The BWT indexing of the graph is highly compute-intensive and requires an enormous amount of RAM and disk space. Further, de Bruijn graph can introduce false positives as it can encode paths that are not present in the graph [32]. Similarly, isolated hits in the graph produced by the read hits can affect the overall performance. The adverse effect on mapping quality and efficiency can be pronounced in the case of long reads.

1.1 Our Results

We present an algorithm V-MAP for efficient identification of subgraph of the genome variation graph for optimal read alignment. In this work, we consider variation graphs that are directed acyclic. The read can be aligned with affine gaps to the identified subgraph using any of the existing graph-based gapped aligners [10, 16, 15, 18, 29]. V-MAP creates a space efficient index of the graph G by computing locality sensitive minimizer signatures of G using a novel graph winnowing technique and combining it with a graph embedding onto a metric space. Graph winnowing creates a compact index which reduces the signature look-ups and also isolated false positive hits. Additionally, graph embedding allows fast subgraph identification based on signature hits.

We performed mapping experiments using real and simulated short/long read datasets on the graph constructed from the 1000 genomes variation data [1]. The variation data contained about 85 million variations and the resulting variation graph consisted of about 3×10^8 vertices and 4×10^8 edges. The 1000 genomes variation graph is a directed acyclic graph.We compared the performance of V-MAP with the state-of-the-art graph-based read mapper VG [10] and the popular linear reference mappers BWA-MEM [20] and Minimap2 [22].

V-MAP index has ∼20GB size and is constructed in just 2 hours using 16GB RAM, which is significantly smaller than VG in terms of the time/resource requirements. In terms of the final alignment score, V-MAP achieved higher alignment scores for a significantly large fraction of long reads. The performance was also significantly better than the linear mappers. Also, V-MAP accuracy in identifying the target region were 96.6% and 99% for short and long reads respectively. V-MAP identifies the target subgraph in milliseconds, even for long reads. In contrast to the whole graph size, the subgraph sizes were significantly smaller and were proportional to the read lengths. This lead to a significant reduction in time for gapped alignment time of reads to target subgraphs. We also provide analytical bounds for the V-MAP index size and for the V-MAP path sampling approach while indexing dense graph regions.

2 Preliminaries

2.1 Notations

Let $G = (V, E, \ell)$ be a connected directed acyclic graph with vertex set V, edge set E and vertex labels given by $\ell(v)$. Edges in E are ordered pairs from $V \times V$. Let σ^+ denote the set of all sequences of one or more elements from an alphabet σ. Each member in σ^+ is called a *σ-sequence*. When σ is the set of nucleotides, the σ-sequences are the nucleotide sequences. For a vertex $v \in V$, its label $\ell(v) \in \sigma^+$ is a σ-sequence. For an ordered sequence $x = (x_1, x_2, \ldots, x_m)$, $|x| = m$ denotes its length. The ith element of x is denoted by $x[i]$. For the sake of brevity, we also denote $\ell(v)$ simply as v and use $v[i]$ to denote $\ell(v)[i]$ when there is no ambiguity.

A directed path p of length r in G is denoted by the ordered sequence $(u_1, \ldots, u_r)$ of r vertices, where $u_i \in V$ and $(u_i, u_{i+1}) \in E$. We say that the path p starts at u_1 and ends at u_r. The σ-sequence corresponding to p is obtained by concatenating the labels $\ell(u_1), \ldots, \ell(u_r)$ in the same order. For each vertex u in G, we associate a *σ-path* given by an ordered sequence of pairs of the form $(\langle u, 0\rangle, \ldots, \langle u, |\ell(u)| - 1\rangle)$, where $\langle u, i\rangle$ denotes the pair of vertex u and the offset i to its label $\ell(u)$. We call each such vertex offset pair a *graph coordinate* of G or *coordinate* in short. A σ-path q in G is given by a sequence $(\langle u_1, i\rangle, \ldots, \langle u_1, |\ell(u_1)| - 1\rangle, \langle u_2, 0\rangle, \ldots, \langle u_2, |\ell(u_2)| - 1\rangle, \ldots, \langle u_{r-1}, 0\rangle, \ldots, \langle u_{r-1}, |\ell(u_{r-1})| - 1\rangle, \langle u_r, 0\rangle, \ldots, \langle u_r, j\rangle)$, where $u_1, \ldots, u_r$ is a path in G and i and j are offsets to $\ell(u_1)$ and $\ell(u_r)$ respectively. Here, the σ-path q is composed of the suffix of u_1 starting at offset i followed by the complete σ-paths of vertices $u_2, \ldots, u_{r-1}$ and finally by a prefix of u_r ending at offset j. Clearly, the σ-path q has a corresponding σ-sequence say q' of the same length obtained by concatenating the corresponding label symbols along the σ-path in the same order. We say that G contains a σ-sequence if it corresponds to some σ-path in G. We also say that G contains a σ-path q (or a σ-sequence q') starting at the graph coordinate $\langle u_1, i\rangle$.

2.2 Sequence Winnowing

Given a σ-sequence s, let K_s denote the set of distinct k-mers in s. We assume random mapping of k-mers to distinct non-negative integers uniformly at random. For simplicity, the integer associated with a k-mer x is also denoted by x. Let $z(s)$ denote the *minimizer* k-mer, which is the minimum valued k-mer, in K_s. We denote by $h(s)$, the *minimizer offset* in s. That is, $h(s)$ is the offset of $z(s)$ in s. If the minimizer occurs at multiple offsets in s then $h(s)$ is defined as the rightmost offset. The function z is locality sensitive in the sense that two sequences r and s have the same minimizer with probability $|K_r \cap K_s|/|K_r \cup K_s|$, which is the Jaccard similarity of the underlying sets K_r and K_s [6].

The standard winnowing of a linear sequence s [30] extends $h(s)$ to a function $h_w(s)$ that maps s to a subset of distinct minimizer offsets from $\{0, \ldots, |s| - 1\}$ by considering a moving window of some fixed size $w \leq |s|$ over s starting at offsets $0, \ldots, |s| - w$. Let $s_0, s_1, \ldots, s_{|s|-w}$ denote the corresponding subsequences. More precisely, $h_w(s)$ maps s to the set of distinct offsets from the set

$$\{i + h(s_i) \;\mid\; 0 \leq i \leq |s| - w\}$$

The function $h_w(s)$ creates a fingerprint of s as a sequence of minimizers with a bounded positional gap between consecutive minimizers. The minimizer sequence can be used for sequence similarity. The expected size of the minimizer set is upper bound by $2|s|/|w|$ [30]. In [14], it was shown empirically that the minimizers computed by h_w can be used to well approximate the Jaccard similarity between sequences and this allows efficient alignment of reads to linear reference databases with higher precision under certain read error models.

3 Indexing

In this section, we describe our approach for indexing G that allows efficient mapping of reads to G. For this, we extend the notion of sequence winnowing to graph winnowing. We combine the graph winnowing with a graph embedding to generate a space efficient index of G that allows fast read mapping. We also discuss approaches to improve indexing performance.

3.1 Graph Winnowing

We extend the above winnowing of linear sequences to graph G by considering σ-sequences in G. Let p be a σ-path in G where $|p| \geq w$ and let p' be its corresponding σ-sequence. We define

$$h(p) = p[h(p')] \qquad \text{and} \qquad h_w(p) = \{p[i] \;\mid\; i \in h_w(p')\}$$

That is, for a σ-path p, $h(p)$ maps p to the graph coordinate corresponding to the minimizer of p' and h_w maps p to the set of graph coordinates corresponding to the minimizer set obtained by winnowing p'.

Let S denote the set of all σ-paths in G each of length $\geq w$. Let $H = \cup_{p \in S} h_w(p)$ denote the set of all distinct minimizer coordinates in G obtained by winnowing all paths in S. Similarly, let $H_w = \cup_{p \in S_w} h_w(p)$ where S_w is the set of all σ-paths in G each of length exactly w. It is straightforward to see that $H = H_w$ because the minimizers obtained by winnowing any σ-path in G are already present in H_w. In graph winnowing, we therefore compute H_w.

3.2 Graph Embedding and Indexing Approach

In this section, we describe our indexing approach which combines the graph winnowing and graph embedding to compute the graph index. Let $\mathbb{N}$ denote the set of natural numbers. We consider an embedding $\lambda : V \times \mathbb{N} \to Y$ of coordinates $\langle v, i\rangle$ in G to a target metric space Y. Let $d_\lambda(x_1, x_2) \in \mathbb{R}^{\geq 0}$ denote the distance between the embedding of coordinates x_1 and x_2 in Y, given by $\lambda(x_1)$ and $\lambda(x_2)$ respectively.

Let S_w denote the set of all σ-paths in G each of length w. We do graph winnowing on G and create the index as follows. Let p be a σ-path in S_w and let p' be its corresponding σ-sequence. Consider the set of minimizer locations $h_w(p')$ in p' obtained by winnowing of p'. For each offset i in $h_w(p')$, let k_i denote the corresponding minimizer k-mer. The graph coordinate of k_i is given by $p[i]$. Let T_G be a key value table (dictionary) where for each minimizer k-mer, the corresponding graph coordinate and its embedding are stored. That is, $T_G[k_i]$ stores the tuple $\langle p[i], \lambda(p[i])\rangle$. Since k-mer k_i can be the minimizer for multiple paths from S_w, $T_G[k_i]$ stores all such tuples corresponding to k_i. The specific choice of embedding and the final structure of the V-MAP graph index are discussed later.

Creating an index in the above manner that combines graph winnowing along with graph embedding results in a space efficient index which at the same time supports fast querying for read mapping. To identify regions in the graph where the read could optimally align, winnowing is performed on the input read and its minimizers are looked up in T_G. The hits from T_G are then clustered in the embedded space to identify maximum density cluster. Though each read minimizer can occur at multiple graph coordinates, the minimizer coordinates belonging to target subgraph tend to cluster in the embedded space. After identifying this cluster, read minimizer hits in this cluster are back-projected to construct the pruned subgraph where the input read is finally aligned. Different choices of embedding allow fast identification of the cluster while also ensuring that the corresponding subgraph is sufficiently pruned. For instance, using an embedding $\lambda : V \times \mathbb{N} \to \mathbb{N}^d$, the target cluster can be identified by computing the maximum enclosing d-dimensional axis parallel box of bounded side lengths in $\mathbb{N}^d$, where the side length is related to the read length. In particular, this can be done efficiently for small d [27]. If t is the total number of minimizer elements for an input query read, then the computation can be done in $O(t \log t)$ time for $d = 1$ by scanning the elements in the sorted order (in the embedded space). Similar bounds are also known for $d = 2$ [27].

V-MAP specific Embedding and Index

V-MAP uses an embedding $\lambda : V \times \mathbb{N} \to \mathbb{N}$ defined as follows. Consider an auxiliary undirected edge weighted graph G' derived from G as follows. For each vertex v in G, we include two vertices v_1 and v_2 in G' which are connected by edge (v_1, v_2) with weight $|\ell(v)| - 1$. Also, for every edge (u, v) in G, an edge (u_2, v_1) with weight 1 is included in G'. If u is the vertex in G with zero in-degree, we designate u_1 as the *source vertex* in G'. In case of multiple vertices with zero in-degree, a dummy source vertex is included that connects to these vertices with zero weight edges. For a vertex v in G', let $sp(v)$ denote the shortest path distance from the source vertex to v in G'. The embedding $\lambda(\langle v, i\rangle)$ for a coordinate $\langle v, i\rangle$ in G is defined with respect to the shortest path distances of the corresponding vertices v_1 and v_2 in G' as

$$\lambda(\langle v, i\rangle) = \min\{sp(v_1) + i, \quad sp(v_2) + |\ell(v)| - i - 1\}$$

Under the above embedding, any two coordinates u and v with $\lambda(v) \geq \lambda(u)$ satisfies that $\lambda(v) - \lambda(u)$ is upper bound by the length of the shortest σ-path that connects u and v in G if there is any such path. Thus, nearby minimizers tend to cluster under this embedding.

In summary, the V-MAP indexing winnows all the w length σ-paths in G and the resulting set of minimizer coordinates are embedded onto the integer line based on their shortest path distances (length of σ-paths) from a designated source vertex coordinate in an undirected version of G. These embeddings are stored against the minimizer values.

The V-MAP index is a key/value table I_G which is a trimmed version of the table T_G discussed earlier. For each minimizer k-mer r, only the embedding of its corresponding coordinates in G are stored in $I_G[r]$, whereas $T_G[r]$ stores the respective coordinates along with their embedding. That is,

$$I_G[r] = \{\lambda(x) \mid T_G[r] \text{ contains the coordinate } x\}$$

The set $I_G[r]$ is a multiset in the sense that for each entry in $T_G[r]$, a separate, not necessarily distinct, element is present in $I_G[r]$.

V-MAP index also contains an adjacency representation of G sorted by the vertex embedding values. In this, the adjacency list for each vertex is also maintained as a sorted skip list or sorted array. This allows efficient extraction of the subgraph identified by V-MAP in order to compute the optimal gapped alignment of the read to the subgraph.

Dynamic Programming for Indexing

Enumerating all w length σ-paths in G for graph winnowing could be computationally expensive. Nevertheless, a dynamic programming heuristic can be used to winnow all w length σ-paths in G to reduce recomputations. The details are given in the Appendix (Section A).

4 Minimizer Density

We provide bounds on the expected number of minimizers created by V-MAP for an input graph G. Specifically, we suppose that the graph G is constructed from a collection $\mathcal{C} = \{s_1, \ldots, s_N\}$ of N related random sequences each of length at most n. The sequences in $\mathcal{C}$ are generated by a coupled random process, where, starting with a designated random sequence, random independent mutations with probability p are introduced to generate each of the remaining sequences in $\mathcal{C}$. The generation model is inspired by the mutation model approximations proposed in [35]. Due to the space limitation, the details are provided in Appendix B. For k-mer length k, we show the following theorem.

▶ **Theorem 1.** *The expected number of minimizers for G is upper bound by $\frac{2n}{w-k}(1 + p(N - 1))^{w+1}$ for winnow length w.*

To prove the above theorem, we first show that the expected number of w length σ-paths in G is bounded from above by $n\,(1 + p(N - 1))^{w}$. Combining this with a charging argument, which is an extension of [30] for linear random sequences to our graph setting, we obtain the theorem. The proofs are provided in Appendix B.

It follows that similar to the linear reference case, in the worst case, the number of minimizers on expectation, which is $O(\frac{n}{w}\exp(pwN))$, grows as $1/w$ fraction of the expected number of w length windows (σ-paths) in G.

4.1 Random Sampling

Dense regions of the graph can lead to increased time and space requirements for V-MAP indexing as the number of possible w length σ-paths in these regions could be very large. We use random sampling in such dense regions where, instead of considering all w length paths, M such paths are randomly sampled (random walk with replacement) where M is a parameter. This helps in reducing the time and space requirements and at the same time maintain high mapping accuracy. V-MAP indexing switches to random sampling mode for a coordinate when the total number of w length σ-paths starting from that coordinate exceeds M. Clearly, the final minimizer set of any graph under sampling is a subset of the minimizer set without sampling.

We analyze the effect of random sampling on the minimizer set. In particular, missing out minimizers due to sampling. Consider a k-mer x in a graph G. We analyze the expected number of w length σ-path samples in G that have x as a minimizer. Let $U = \{u_1, \ldots, u_r\}$ be the set of coordinates in G such that x is reachable from them through a σ-path of length at most w. Let q be the probability that a random w length σ-path sample from a randomly chosen (uniformly at random) $u \in U$ contains x. We suppose the worst case that the number of w length σ-paths from each $u_i \in U$ exceeds M. In the analysis, we assume that the k-mers in any w length σ-path are distinct.

▶ **Lemma 2.** *The expected number of path samples each having x as the minimizer $\geq qM$.*

Proof. We observe that $|U| \geq w - k + 1$ as there are at least $w - k + 1$ coordinates lying on some w length σ-path leading to x in G. Consider any coordinate u_i from U. Let α_i denote the probability that a random w length σ-path sample from u_i contains the k-mer x. Then, the probability that a random sampled path contains x as minimizer is at least $\alpha_i/(w - k + 1)$. The total number of path samples (with repetition) from all of $u_1, \ldots, u_r$ is rM. In this sampling, let Z denote the the expected number of path samples where each of them have x as a minimizer. We have $Z \geq \frac{M}{(w-k+1)} \sum_{i=1}^{r} \alpha_i$. Recalling that $r \geq w - k + 1$, we obtain that $Z \geq \frac{M}{r} \sum_{i=1}^{r} \alpha_i \;=\; qM$ ◀

It is straightforward to see that, under no sampling, the expected number of w length σ-paths in G each having x as minimizer is $S/(w - k + 1)$, where S denote the total number of w length σ-paths from all the coordinates in U where each path contains the k-mer x.

From the above lemma, we infer that when a minimizer k-mer x of graph G is present in a dense region with high q value, then with a suitable sampling parameter M, sufficiently many paths under sampling also have x as a minimizer on expectation. For example, Figure 1 shows part of a commonly occurring dense subgraph topology which is a cascade of variations. The number of w length σ-paths from a fixed graph coordinate grows by a multiplicative factor depending on the number of cascade elements it can pass through. For the k-mer x as shown in the subgraph, there are several w length σ-paths from every coordinate in U. However, a random path sample from any coordinate in U will pass through the vertex containing x with probability $1/3$, implying a higher q value.

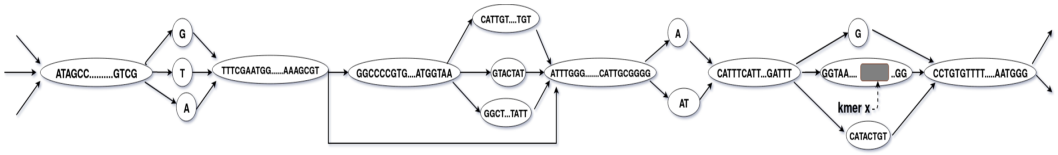

Figure 1 Dense subgraph region with a cascade of variations.

If there is prior information available about the likelihood of certain paths over others in a graph region, such as the underlying haplotype information, edge weights can be introduced to capture non-uniform transition probabilities between vertices. Graph winnowing is easily generalizable to this setting resulting in biased path sampling for winnowing. Such a haplotype aware sampling can strike a balance between using haplotype prior and allowing newer paths that could arise due to recombinations. Further generalizations that allow a k-step Markov walk for path sampling can tilt the balance further in favour of haplotype prior.

5 Read Mapping

We describe the V-MAP read mapping algorithm in this section. For an input read s, the mapper aims to efficiently find a small subgraph G' of the graph G where s can align optimally. That is, G' has a path where s aligns optimally among all paths in G. The alignment of s to G' with affine gaps could be performed with any of the existing graph-based gapped aligners for sequences [10, 16, 15, 18, 29].

The minimizer set R of the input read s is first computed by applying the winnow hashing h_w on s. For each distinct minimizer $x \in R$, the entries stored at $I[x]$ in the index I are included in the hit set H. We recall that $I[x]$ contains the embedding in $\mathbb{N}$ of all coordinates in G where x is a minimizer. Set $I[x]$ is assumed to be empty if x is not present as a minimizer in G. The final union H is a multiset. We now identify a maximum cardinality enclosing interval in $\mathbb{N}$ of fixed width D that covers the maximum number of elements in H. This can be done easily in $O(|H| \log |H|)$ time by sorting H followed by a linear scan. The interval width D is fixed to be the length ℓ of the read s. Let h_l and h_r be the left end and right end respectively of the final enclosing interval. The final graph region identified by V-MAP is given by the subgraph induced by the vertices whose embedding lies in the interval $[h_l - \ell/2, h_r + \ell/2]$. The vertex set and the adjacency information of each vertex are stored in a sorted fashion based on the embedding of the start and the end coordinates of the vertices. This allows efficient extraction of the subgraph identified by V-MAP. Optimal gapped alignment of the read to the extracted subgraph can be computed using any graph alignment algorithm.

Reverse Complement

Reads from reverse complement strand can be handled using strand prediction in a straightforward way using techniques in [14] with a marginal increase in the index size. While indexing, each k-mer is transformed to its canonical form (either itself or its reverse complement based on their lexicographic ordering). For a minimizer k-mer x stored in the index I_G, we store an additional bit along with each of the coordinate embeddings stored in the set $I_G[x]$. The canonical bit records whether x appeared in the canonical form or not at that coordinate. While mapping, the canonical bit is computed for the read minimizers as well. Strand prediction is done based on the consensus of the canonical bits of the minimizers in the final cluster and bits of the read minimizers.

6 Experimental Results

We performed experiments to measure mapping accuracy and run time performance of V-MAP. We compared V-MAP with the state-of-the-art VG mapper [10] and popular linear reference mappers BWA-MEM [20] and Minimap2 [22].

Genome Variation Graph Construction

Human reference genome GRCh37 and the 1000 Genomes [1] variation data (phase 3) were used to construct a Human genome graph for our experiments. The variation data consisted of about 85 million variations. The VG graph construction tool [10] was used with default parameter setting to construct the *reference graph*. The reference graph consisted of about 3.2×10^8 vertices and 4.1×10^8 edges in total.

Read Set

The Illumina, PacBio and ONT real reads of *AshkenazimTrio HG002_NA24385_son* made available by Genome In A Bottle (GIAB) Consortium [39] were used. Around 10^4 reads of average length 117 formed the Illumina readset. For PacBio, $\sim 10^4$ reads of length $5k$ or above were used. For ONT, about 8000 reads were available in the GIAB dataset with read length 1000 or above.

Simulated reads were also used in the experiments. Reads were generated using read simulation tools from sequences arising from random paths in the reference graph. The ART tool [13] was used to generate Illumina short reads of lengths in the range 100 to 200 with its default error profile. PacBio and Nanopore (ONT) long reads of average lengths of $5k, 10k$ and $50k$ were generated using ReadSim tool [19]. In total, 144×10^5 short reads and 3.8×10^5 long reads were generated. All reads in the read set are from forward strand. Strand prediction aspect is not considered in this work.

Implementation Details

V-MAP was implemented in C++. The index construction and mapping were performed on 200 GB RAM Linux machine with 48 threads and with 6TB HDD. Choices for the V-MAP parameters w and k were determined by a grid search based on the mapping accuracy on a separate set of random reads. Values $w = 35, k = 20$ were chosen for short reads and $w = 40, k = 20$ were chosen for long reads (PacBio/ONT). The random sampling parameter M for indexing was set to $30,000$.

6.1 Mapping Accuracy

The final alignment score achieved by the candidate methods V-MAP, VG, BWA-MEM, and Minimap2 were measured for the real read set. In V-MAP, as in VG, the GSSW graph-based aligner [38, 10] was used to compute the gapped alignment score to the identified subgraph. The default alignment score parameters of BWA-MEM for gap open, gap extension, match, and mismatch were used across all tools. Table 2 gives the percentage of reads where each tool was a top scorer. That is, its score is no less than the score of other tools. The best value among BWA-MEM and Minimap2 is shown under the linear mapper column. The low percentage for linear mapper especially for long reads can be attributed to the fact that unlike the graph-based mappers, linear mappers have to work with only the linear reference and without any variant information. The graph mappers like V-MAP, on the other hand, achieve higher alignment score by working on a unified space of reference and variations encoded as a graph. The purpose of including the linear mappers in the comparison is only to bring out the advantage of using a pangenome reference over a linear reference and not to highlight the performance of any specific linear mapper. In Table 1, we show the average percentage gain in the alignment score achieved respectively by V-MAP and VG for reads where they were top scorers. As seen in Table 1, the percentage gain is pronounced in the case of long reads.

Table 1 Average percentage gain in the alignment score for V-MAP and VG.

% Reads	Illumina			PacBio			ONT		
	V-MAP	VG	Linear mapper	V-MAP	VG	Linear mapper	V-MAP	VG	Linear mapper
Top scorer	95.28	**99.79**	87.98	**81.15**	52.01	3.2	**80.73**	62.43	3.43

Table 2 Comparison of the candidate methods based on the final alignment scores achieved. The best performance among BWA-MEM and Minimap2 is shown under linear mapper.

Illumina		PacBio		ONT	
V-MAP	VG	V-MAP	VG	V-MAP	VG
0.0004	0.05	5.85	3.29	7.06	6.42

Using simulated reads, we calculated the mapping accuracy measured as the fraction of reads for which the subgraph identified by the mapper contained the path in the graph from which the read was generated. Table 3 shows the mapping accuracy of V-MAP and VG. V-MAP exhibits consistently superior performance for long reads with a much smaller index. In addition to higher accuracy, V-MAP also achieved considerably higher final alignment score. The average score difference was in excess of 680 for reads in the 5k to 10k range and 3590 for 10k to 50k range and 18300 for above 50k length reads. For short reads, the V-MAP alignment score was less than VG by 2 on average. In a related experiment of mapping 24k simulated reads of length 10k each from the graph with no sequencing errors, V-MAP achieved full alignment score for all but 2 reads in comparison to 96.8% of the reads for VG.

Table 3 Mapping accuracies for different read classes.

Read class	100	150	200	Pac 5k	Pac 10k	Pac 50k	ONT 5k	ONT 10k	ONT 50k
V-MAP	95.22	96.21	98.49	96.90	99.04	99.63	99.47	99.57	99.95
VG	99.67	99.80	99.98	71.62	70.48	71.27	94.28	94.13	93.93

6.2 Mapping Performance

Table 4 gives the indexing performance, i.e., index construction time, index size, intermediate storage and RAM requirements for V-MAP and VG.

Table 5 gives the average read mapping time (single thread) for V-MAP and VG for short and long reads. The map time is the subgraph identification time and align time is the time taken for gapped alignment of the read to the identified subgraph using the GSSW graph-based aligner [38, 10]. The difference in alignment time between the two tools can be attributed to the smaller subgraphs identified by V-MAP for the alignment phase as shown in Figure 2.

Figure 2 shows the average size statistics of identified subgraphs for short and long reads. The figure shows the number of edges, vertices and the total number of nucleotides in the vertex label sequences of the subgraph. The reduced subgraphs sizes significantly improve the gapped alignment time. Also, improved algorithms and implementations for gapped alignment on graphs can further improve the overall time.

Table 4 Index construction and Mapping performance.

Index Parameters	**V-MAP**	**VG**
Construction Time	2 hours	36 hours
Intermediate storage	0	$\sim$ 2 TB
Index Size	21 GB (short reads) 18 GB (long reads)	80 GB
RAM	16 GB	200 GB
# indexing threads	4	32
Mapping RAM	27 GB	75 GB

Table 5 Map and align timings. Map time is the subgraph identification time. Align time is the time to perform gapped alignment of the read to subgraph using GSSW.

Read length	Map time (ms)		Align time (ms)		Total time (ms)	
	V-MAP	**VG**	**V-MAP**	**VG**	**V-MAP**	**VG**
100	0.45	4.56	1.88	33	2.33	37.56
150	0.67	12.02	2.94	94.85	3.61	106.87
200	1.24	37.5	3.91	153.15	5.15	190.65
5k	5.84	144.16	540.28	4300.22	546.12	4444.39
10k	11.35	292.22	1831.34	9010.63	1842.69	9302.85
50k	63.63	1223.1	39,584.69	48,652.42	39,648.13	49,875.5

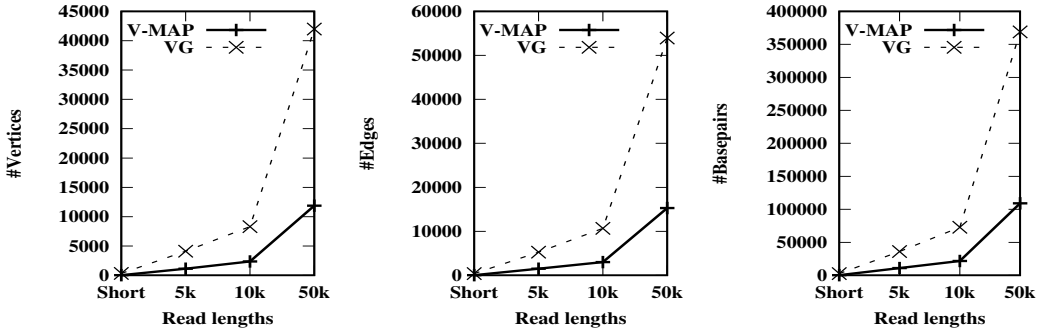

Figure 2 Average size statistics of identified subgraphs for short and long reads. The figure shows the number of edges, number of vertices and the total number of nucleotides in the vertex label sequences of the subgraph.

7 Discussion

We present V-MAP for efficient identification of a subgraph of the input genome variation graph for optimal read alignment. Our tool exhibited significantly improved performance in comparison to the state-of-the-art. Improved algorithms and implementations of gapped aligners for graphs can further improve the overall performance significantly. Also, better graph embedding, say in $\mathbb{N}^2$ or other metric spaces including trees or other simpler graphs, could result in finer graph pruning and hence faster alignment.

References

1 1000Genome. 1000 Genome VCF. `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502`, 2013. [Online; accessed 15-April-2017].

2 Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

3 Michael Burrows and David Wheeler. A Block-Sorting Lossless Data Compression Algorithm. In *DIGITAL SRC RESEARCH REPORT*. Citeseer, 1994.

4 Stefan Canzar and Steven L Salzberg. Short read mapping: an algorithmic tour. *Proceedings of the IEEE*, 105(3):436–458, 2017.

5 Mark J Chaisson and Glenn Tesler. Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. *BMC bioinformatics*, 13(1):238, 2012.

6 Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

7 Agnieszka Danek, Sebastian Deorowicz, and Szymon Grabowski. Indexes of large genome collections on a PC. *PloS one*, 9(10):e109384, 2014.

8 Arthur L Delcher, Adam Phillippy, Jane Carlton, and Steven L Salzberg. Fast algorithms for large-scale genome alignment and comparison. *Nucleic acids research*, 30(11):2478–2483, 2002.

9 Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 390–398. IEEE, 2000.

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. Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nature biotechnology*, 2018.

11 Roberto Grossi and Jeffrey Scott Vitter. Compressed suffix arrays and suffix trees with applications to text indexing and string matching. *SIAM Journal on Computing*, 35(2):378–407, 2005.

12 Lin Huang, Victoria Popic, and Serafim Batzoglou. Short read alignment with populations of genomes. *Bioinformatics*, 29(13):i361–i370, 2013.

13 Weichun Huang, Leping Li, Jason R Myers, and Gabor T Marth. ART: a next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594, 2011.

14 Chirag Jain, Alexander Dilthey, Sergey Koren, Srinivas Aluru, and Adam M Phillippy. A fast approximate algorithm for mapping long reads to large reference databases. In *International Conference on Research in Computational Molecular Biology*, pages 66–81. Springer, 2017.

15 Chirag Jain, Haowen Zhang, Yu Gao, and Srinivas Aluru. On the complexity of sequence to graph alignment. In *International Conference on Research in Computational Molecular Biology*, pages 85–100. Springer, 2019.

16 Vaddadi Naga Sai Kavya, Kshitij Tayal, Rajgopal Srinivasan, and Naveen Sivadasan. Sequence Alignment on Directed Graphs. *Journal of Computational Biology*, 2018.

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

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

19 Hayan Lee, James Gurtowski, Shinjae Yoo, Shoshana Marcus, W Richard McCombie, and Michael Schatz. Error correction and assembly complexity of single molecule sequencing reads. *BioRxiv*, page 006395, 2014.

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

21 Heng Li. Minimap and Miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, 32(14):2103–2110, 2016.

22 Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 1:7, 2018.

23 Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows–Wheeler transform. *bioinformatics*, 25(14):1754–1760, 2009.

24 Heng Li and Nils Homer. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in bioinformatics*, 11(5):473–483, 2010.

25 Antoine Limasset, Bastien Cazaux, Eric Rivals, and Pierre Peterlongo. Read mapping on de Bruijn graphs. *BMC bioinformatics*, 17(1):237, 2016.

26 Tobias Marschall, et al. Computational pan-genomics: status, promises and challenges. *Briefings in bioinformatics*, 19(1):118–135, 2016.

27 Subhas C Nandy and Bhargab B Bhattacharya. A unified algorithm for finding maximum and minimum object enclosing rectangles and cuboids. *Computers & Mathematics with Applications*, 29(8):45–61, 1995.

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

29 Mikko Rautiainen and Tobias Marschall. Aligning sequences to general graphs in $O(V + mE)$ time. *bioRxiv*, page 216127, 2017.

30 Saul Schleimer, Daniel S Wilkerson, and Alex Aiken. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 76–85. ACM, 2003.

31 Korbinian Schneeberger, Jörg Hagmann, Stephan Ossowski, Norman Warthmann, Sandra Gesing, Oliver Kohlbacher, and Detlef Weigel. Simultaneous alignment of short reads against multiple genomes. *Genome biology*, 10(9):R98, 2009.

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

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

34 Ivan Sović, Mile Šikić, Andreas Wilm, Shannon Nicole Fenlon, Swaine Chen, and Niranjan Nagarajan. Fast and sensitive mapping of nanopore sequencing reads with GraphMap. *Nature communications*, 7:11307, 2016.

35 Matthew Stephens and Peter Donnelly. Inference in molecular population genetics. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4):605–635, 2000.

36 Daniel Valenzuela and Veli Mäkinen. CHIC: a short read aligner for pan-genomic references. *bioRxiv*, page 178129, 2017.

37 Sebastian Wandelt, Johannes Starlinger, Marc Bux, and Ulf Leser. RCSI: Scalable similarity search in thousand (s) of genomes. *Proceedings of the VLDB Endowment*, 6(13):1534–1545, 2013.

38 Mengyao Zhao, Wan-Ping Lee, Erik P Garrison, and Gabor T Marth. SSW library: an SIMD Smith-Waterman C/C++ library for use in genomic applications. *PloS one*, 8(12), 2013.

39 Justin M Zook, David Catoe, Jennifer McDaniel, Lindsay Vang, Noah Spies, Arend Sidow, Ziming Weng, Yuling Liu, Christopher E Mason, Noah Alexander, et al. Extensive sequencing of seven human genomes to characterize benchmark reference materials. *Scientific data*, 3:160025, 2016.

A Dynamic Programming for Indexing

Enumerating all w length σ-paths in G for graph winnowing could be computationally expensive. Nevertheless, a dynamic programming approach can be used to winnow all w length σ-paths in G to avoid recomputations. It is based on the observation that the minimizer set H_t of the set of all t length paths, each starting at a given graph coordinate $\langle u, i\rangle$, can be obtained recursively from the minimizer set H_{t-1} of the set of all $t-1$ length paths in G having the property that for each of these paths, its start coordinate, say $\langle v, j\rangle$, occurs immediately after $\langle u, i\rangle$ in some σ-path in G. As a consequence, each of these $t-1$ length σ-path p is extended to a t length σ-path p^+ by appending $\langle u, i\rangle$ to its beginning. Winnowing of p^+ is easily done by finding the minimum of the already computed minimizer of p and the leftmost k-mer of p^+. The details of the DP formulation are given below.

Let $P_t(v, j, s)$ denote the set of all t length σ-paths in G each starting at the graph coordinate $\langle v, j\rangle$ and having the $k-1$ length sequence s as the prefix of its corresponding σ-sequence. For a σ-path $p \in P_t(v, j, s)$, associate a triplet $\langle u, i, r\rangle$ corresponding to the minimizer of p where $\langle u, i\rangle = h(p)$ denotes the minimizer coordinate and r denotes the minimizer k-mer. Corresponding to $P_t(v, j, s)$, let

$$H_t(v, j, s) = \{\langle u, i, r\rangle \text{ associated with a } p \in P_t(v, j, s)\}$$

We use $H_t(v, j, s)$ to also denote an underlying max heap of its triplets where the ordering is based on the k-mer values. In the following, we define a recurrence on $H_{t+1}(v, j, s)$ where s is a $k-1$ length prefix of some σ-sequence starting at $\langle v, j\rangle$. Clearly $\ell(v)[j] = s[0]$. Let s' be the $k-2$ length suffix of s. Define the set R of successor coordinates of the coordinate $\langle v, j\rangle$ as follows. If $j < |v| - 1$ then $R = \{\langle v, j+1\rangle\}$. If $j = |v| - 1$ then $R = \{\langle u, 0\rangle \mid u \in N_o(v)\}$, where $N_o(v)$, called the *out-neighbors* of v, denote the set of all vertices that have directed edges from v.

We now compute $H_{t+1}(v, j, s)$, where $|s| = k-1$, recursively from the H_t sets of the successor coordinates of $\langle v, j\rangle$ in R. For this, we define $\mathcal{H}_t$ which is a set of non-empty sets $H_t(u, i, z)$, where $|z| = k-1$, as

$$\mathcal{H}_t \;=\; \{H_t(u, i, z) \mid \langle u, i\rangle \in R \text{ and } s' \text{ is prefix of } z\}$$

For each $H_t(u, i, z) \in \mathcal{H}_t$, let z^+ denote the k-mer obtained by concatenating $s[0]$ followed by z. We modify the max heap $H_t(u, i, z)$ to obtain a modified max heap (and the underlying triplet set) $H'_t(u, i, z)$ by removing all triplets from $H_t(u, i, z)$ whose k-mer values are greater than or equal to z^+. Also, insert the triplet $\langle v, j, z^+\rangle$ in $H'_t(u, i, z)$ if at least one triplet was removed from $H_t(u, i, z)$. Finally, the required $H_{t+1}(v, j, s)$ is obtained as

$$H_{t+1}(v, j, s) = \bigcup_{H_t(u,i,z) \in \mathcal{H}_t} H'_t(u, i, z)$$

The final set of triplets L to be indexed is given by the union of all non-empty $H_w(v, j, s)$ sets computed as above. For each triplet $\langle u, i, r\rangle$ in L, $\lambda(\langle u, i\rangle)$ is included in the index set $I_G[r]$.

The above recurrence is defined for $t > k$. For the base case $t = k$, we construct $H_k(v, j, s)$ for each coordinate $\langle v, j\rangle$ in G by including all triplets $\langle v, j, r\rangle$ in $H_k(v, j, s)$ where s is the $k-1$ length prefix of the k-mer r occurring at $\langle v, j\rangle$ in G.

In the above dynamic programming based minimizer computation, after round t of the dynamic programming updation, for each graph coordinate $\langle v, j\rangle$ and for each $k-1$ length string (σ-sequence) s starting from the coordinate $\langle v, j\rangle$, we maintain a separate heap $H_t(v, j, s)$. In the next round $t+1$, the new set $H_{t+1}(u, i, z)$, for a $k-1$ length string z

starting at coordinate $\langle u, i\rangle$, is computed from all sets $H_t(v, j, s)$ such that the $k-2$ length suffix of s and $k-2$ prefix of z are identical. Coordinates $\langle u, i\rangle$ and $\langle v, j\rangle$ should additionally satisfy that either (a) $u = v; j = i + 1$ or that (b) $j = 0; i = |\ell(u)| - 1$; (u, v) is a directed edge in G. In this case, s and z give rise to a new k-mer r which is $z[0]$ followed by the string s. Now, $H_{t+1}(u, i, z)$ is the union of all minimizer entries of such $H_t(v, j, s)$ excluding the minimizers having value greater than r. A new entry for r is also included in $H_{t+1}(u, i, z)$ if at least one minimizer was excluded in the union.

From the complexity perspective, we see that after round t, the number of H_t sets (heaps) that are present is at most the total number of $k-1$ length σ-paths in G. This is because, a separate $H_t(v, j, s)$ is maintained for each $k-1$ length string starting from the graph coordinate $\langle v, j\rangle$. For a minimizer coordinate triplet $\langle y, j, x\rangle$, let $\beta(y, j, x)$ denote the total number of t length σ-paths in G each having x at coordinate $\langle y, j\rangle$ as its minimizer. Then, it is straightforward to verify that the minimizer triplet $\langle y, j, x\rangle$ can be present in at most $\beta(y, j, x)$ different H_t sets. Let set B denote the set of all distinct minimizer triplets obtained by winnowing t length σ-paths in G. Then, the sum total of the sizes of all H_t sets is given by $\sum_{r\in B} \beta(r)$. This in the worst case is upper bound as $\sum |H_t| \leq |B| \times d$ where d is the maximum number of t length σ-paths in G such that each give rise to the same minimizer triplet. That is, $d = \max_{r\in B} \beta(r)$. In other words, the blowup in space after round t is by a factor d in the worst case.

From the algorithm description, we can see that, in round $t+1$, each $H_t(v, j, s)$ with $j > 0$ is scanned exactly once for obtaining a new H_{t+1} set. This is because, only $H_{t+1}(v, j-1, \cdot)$ requires $H_t(v, j, s)$. However, set $H_t(v, 0, s)$ entries are scanned as many times as the in-degree of vertex v in G. Hence, the total number of triplet scanning performed in round $t+1$ is upper bound by $\sum |H_t| + \left(\sum_{v\in V} |H_t(v, 0, \cdot)| \times \text{ in-degree } (v)\right)$, which can be crudely upper bound as $O(dm|B|)$, where m is the total number of edges in G.

In the subsequent section, we discuss random path sampling approach for improving the indexing performance in the dense graph regions when the winnowing is performed by scanning w length paths in G. This dynamic programming heuristic does not incorporate the path sampling strategy. The DP approach can be useful for winnowing non-dense graphs.

B Minimizer Density

We provide bounds on the expected number of minimizers created by V-MAP for an input graph G corresponding to a collection of random related sequences. Specifically, we suppose that the graph G is constructed from a collection $\mathcal{C} = \{s_1, \ldots, s_N\}$ of N random related sequences each of length at most n. The sequences in $\mathcal{C}$ are generated by a coupled random process described in the following, where, starting with a designated random sequence, random independent mutations with probability p are introduced to generate each of the remaining sequences in $\mathcal{C}$. This generation model is inspired by the mutation model approximations proposed in [35]. Sequence s_1, also denoted as s_1^*, is designated as the *basis sequence* which is a random sequence where the nucleotide at each location is chosen independently and uniformly at random. Each remaining sequence s_k for $k \in \{2, \ldots, N\}$ is generated independently as gapped variant of the basis sequence s_1 using the following coupled process. The basis sequence s_1^* is scanned sequentially from left to right for n steps. At each step t, the nucleotide present in s_1^* is appended to s_k independently with probability $1-p$ for some fixed p. With probability p, a gap introduced in s_k either as a deletion from s_1^* (i.e., skip the nucleotide in s_1^*), a random nucleotide substitution in s_k or a nucleotide insertion in s_k. In the case of an insertion, the scan location in s_1^* does not change. It is straightforward to

verify that any given sequence from $\mathcal{C}$ is a random sequence where the nucleotide at each location is chosen independently and uniformly at random. However, any two sequences from $\mathcal{C}$ are not independent of each other. If the mutation and gap probabilities are unequal, let p denote the maximum of them in the following analysis.

We now consider a genome variation graph G constructed from $\mathcal{C}$. The above generation process gives a natural way to construct G. The complete basis sequence s_1^* is initially represented as a single labeled vertex. Each of the remaining coupled sequence s_k for $k \geq 2$ is incorporated by introducing alternative paths in G at the gap regions with respect to s_1^*. Specifically, a contiguous gap between s_k and s_1^* is handled by introducing a new vertex u that encodes the subsequence of s_k in this gap region. The new vertex u is connected as a bridge vertex parallel to the corresponding gap subsequence of s_1^* in G. For this, a vertex split is done at the begin and end locations of this gap in the respective vertices of the path that encodes the gap subsequence of s_1^*. After the split, there is a path in G that exactly encodes the gap subsequence of s_1^* and the new vertex u is connected as a bridge parallel to this path. If the gap region consists of only deletions along s_k, then the gap subsequence of s_k is empty. In this case, just a bridge edge is introduced across the gap sequence path of s_1^*.

In each step, while generating a new sequence, a gap occurs with probability p and each gap introduces a constant number of additional edges and vertices in G. Hence the expected number of vertices and edges in G is $O(npN)$. In order to bound the minimizer cardinality, first, we bound the expected number of w length σ-paths in G.

▶ **Lemma 3.** *The expected number of w length σ-paths is upper bound by $n\,(1 + p(N-1))^w$.*

Proof. For the ease of exposition, we assume without loss of generality that each vertex of G has only single nucleotide label. If a vertex has a longer label, it can be replaced by a corresponding path of single nucleotide vertices. Let the vertices along the s_1 path in G be $U = \{u_1, \ldots, u_n\}$ and let d_i be the out-degree of u_i. Let X denote the set of all w length σ-paths in G. Let X_i for $i \in \{1, \ldots, n\}$ denote the subset of X where each σ-path x in it has u_i at location $x[m-2]$. Let R denote the remaining paths from X not included in any of X_i. Consider the $m-1$ length prefix $x[0, \cdots, m-2]$ of each σ-path x from an X_i. Let n_i be the number of distinct such prefixes from X_i. Clearly, $|X| \leq \sum n_i d_i + |R|$. We observe that n_i is independent of d_i and that $E(d_i)$ is approximated by $1 + E(Binomial(N-1, p))$ because each of $s_2, \ldots, s_N$ can introduce an outgoing edge at u_i with probability p. Therefore, we obtain $E(|X|) \leq (1 + p(N-1))E(\sum n_i + |R|)$. Clearly $\sum n_i + |R|$ is no more than the number of $w-1$ length σ-paths in G. Here we observe that $w-1$ length prefix of each path x from R is distinct as the out-degree of a non U vertex (vertex at $x[m-2]$) is at most 1. Repeating the above argument and using the observation that the expected number of 1 length σ-paths (i.e., total nucleotides) in G is at most $n(1 + p(N-1))$, we finally obtain $E(X) \leq n(1 + p(N-1))^w$. ◀

The following Theorem provides a bound on the expected number of minimizers resulting from winnowing of G.

▶ **Theorem 4.** *The expected number of minimizers for G is upper bound by $\frac{2n}{w-k}(1 + p(N-1))^{w+1}$ for winnow length w.*

Proof. We extend the charging argument in [30] for linear random sequences to our graph setting. In [30], for a linear sequence s, a w sized window W_i at coordinate i is charged, i.e., contributes a new minimizer k-mer say at $s[j]$, if W_i is the leftmost among the windows that overlap the k-mer at $s[j]$ and the k-mer is a minimizer in them. It was shown in [30] that this

corresponds to the event that in the $w+1$ sized window W which is the union of W_{i-1} and W_i, the minimizer of W is given by either it's first k-mer or its last k-mer. The probability of this to happen in a random sequence is $2/(w+1)$. The total number of minimizers after winnowing an n length sequence s is then given by the total number of w length windows that are charged, which is $2n/(w+1)$.

Consider any $w+1$ length σ-path in G. Let c' denote the w length suffix window of c and let x and x' denote the (minimizer) k-mers at locations $h(c)$ and $h(c')$ respectively. We charge window c' if and only if x is either the first or the last k-mer in c. It is straightforward to verify that each minimizer in G is charged to at least one w length σ-path window in G. In the linear case, a minimizer is charged to exactly one window whereas the same minimizer could be charged to multiple windows in the case of graphs. This can only overestimate the desired count. Recalling the generation process of G, we observe that any given $w+1$ length σ-path c in G encodes a random σ-sequence. Hence, its w length suffix c' is charged with probability $2/(w-k+2)$. The graph G has an underlying topology from Ω which denote the space of all graph topologies that can arise from the graph creation process described earlier. Each topology in Ω has a unique encoding where each vertex u has an associated distinct triplet id of the form $\langle i,j,l\rangle$ denoting that u is uniquely associated with the l length subsequence of $s_i \in \mathcal{C}$ starting from offset j. For a given topology, the specific vertex labels are still random. It is clear from the generation process that for any fixed topology T, the σ-sequence for any given σ-path in T is a random sequence. Hence, if n_T denote the number of $w+1$ length σ-paths for a topology T, a $w+1$ length σ-path is charged in a graph with topology T with probability $2/(w-k+2)$. Let D denote the number of charged $k+1$ length σ-sequence in G. The total number of minimizers is upper bound by D. We have $E(D \mid T) = 2 \cdot n_T/(w-k+2)$. Hence, we obtain

$$\begin{aligned} E(D) &= \sum_{T\in\Omega} E(D \mid T)\Pr(T) \\ &= \frac{2}{w-k+2}\sum_{T\in\Omega} n_T \Pr(T) \quad \leq \quad \frac{2n}{w-k}(1+p(N-1))^{w+1} \end{aligned}$$

where the last inequality follows by applying Lemma 3 as $\sum_{T\in\Omega} n_T \Pr(T)$ is the expected number of $w+1$ length σ-paths . ◀