Full-length de novo viral quasispecies assembly through variation graph construction

Jasmijn A. Baaijens*, Bastiaan Van der Roest†, Johannes Köster‡§, Leen Stougie*¶, Alexander Schönhuth*‖**

Abstract

Motivation: Viruses populate their hosts as a viral quasispecies: a collection of genetically related mutant strains.Viral quasispecies assembly refers to reconstructing the strain-specific haplotypes from read data, and predicting their relative abundances within the mix of strains, an important step for various treatment-related reasons. Reference-genome-independent ("de novo") approaches have yielded benefits over reference-guided approaches, because reference-induced biases can become overwhelming when dealing with divergent strains. While being very accurate, extant de novo methods only yield rather short contigs. It remains to reconstruct full-length haplotypes together with their abundances from such contigs.

Method: We first construct a variation graph, a recently popular, suitable structure for arranging and integrating several related genomes, from the short input contigs, without making use of a reference genome. To obtain paths through the variation graph that reflect the original haplotypes, we solve a minimization problem that yields a selection of maximal-length paths that is optimal in terms of being compatible with the read coverages computed for the nodes of the variation graph. We output the resulting selection of maximal length paths as the haplotypes, together with their abundances.

Results: Benchmarking experiments on challenging simulated data sets show significant improvements in assembly contiguity compared to the input contigs, while preserving low error rates. As a consequence, our method outperforms all state-of-the-art viral quasispecies assemblers that aim at the construction of full-length haplotypes, in terms of various relevant assembly measures. Our tool, *Virus-VG*, is publicly available at `https://bitbucket.org/jbaaijens/virus-vg`.

1 Introduction

The ensemble of genetically related, but different mutant viral strains that populate infected people are commonly referred to as *viral quasispecies* [7]. Each of these strains comes with its own genomic sequence (henceforth referred to as *haplotype*). The final goal of primary viral quasispecies analysis is the reconstruction of the individual haplotypes—optimally at full length—and also to provide estimates for their abundances, because they can vary among each other. The unknown number of different, strain-specific haplotypes and their variance in abundance establish the theoretical issues that characterize viral quasispecies assembly. They explain why this form of assembly is difficult, despite the shortness of virus genomes. These issues are further accentuated by the fact that neither next-generation nor third-generation sequencing reads, by their combinations of error rates and length, allow for immediate reconstruction and abundance estimation of haplotypes [4, 21].

*Centrum Wiskunde & Informatica, Amsterdam, Netherlands
†University Medical Center Utrecht, Utrecht, Netherlands
‡University of Duisberg-Essen, Essen, Germany
§Dana Farber Cancer Institute, Harvard Medical School, Boston, United States
¶Vrije Universiteit, Amsterdam, Netherlands
‖Utrecht University, Utrecht, Netherlands
**Corresponding author (`alexander.schoenhuth@cwi.nl`)

State-of-the-art approaches currently allow for two options: **(i)** full-length reconstruction of haplotypes based on statistical, usually *reference genome dependent* measures, or **(ii)** *de novo* reconstruction of (optimally haplotype-specific) contigs.

Approaches of type **(i)** assume that the sequencing reads are aligned to a reference genome and make use of model-based clustering algorithms [28, ShoRAH], Dirichlet process mixture models [18, PredictHaplo], hidden Markov models [25, QuasiRecomb], or sampling schemes [19, QuRe], respectively. However, as was demonstrated in [1, 24], resorting to external auxiliary means (such as reference genomes) can bias the reconstruction procedure significantly.

Approaches of type **(ii)** comprise generic (meta)genome assemblers as well as specialized viral quasispecies assemblers, both of which are not helped by external measures (*"de novo"*) hence are not affected by external biases. Metagenome assemblers are designed to reconstruct multiple genomes simultaneously, but in viral quasispecies tend to collapse strains [21]. It was further shown in [1] that among generic de novo assemblers SPAdes [2] was the only approach to identify strain-specific sequences, however only in case of sufficiently abundant strains. *De novo* viral quasispecies assemblers (e.g. [11, 27]) generally aim at constructing suitable consensus reference genomes, which may serve as a template for more finegrained studies (for example if curated reference genomes have become too divergent, which is a frequent scenario). To the best of our knowledge, the only methods that truly aim at *de novo* viral quasispecies assembly *at strain resolution* are SAVAGE [1] and MLEHaplo [14] (where the first was shown to have significant advantages over the latter). However, the contigs, while strain-specific, in general do not represent full-length haplotypes.

Here, we present Virus-VG, an algorithm that turns strain-specific contigs into full-length, strain-specific haplotypes, thus completing the de novo viral quasispecies assembly task. The contigs can be taken from [1, 14], the only available approaches that can deliver *de novo strain-specific* contigs. For that, we construct a variation graph from the contigs, without the help of a (curated) reference genome. We obtain full-length haplotypes as a selection of maximal-length paths in the variation graph, each of which reflects a concatenation of subpaths associated with the input contigs. The selected paths are optimal in terms of differences between their estimated abundances and the read coverages computed for the nodes they traverse.

Variation graphs are data structures that have recently become very popular as reference systems for (evolutionarily coherent) collections of genomes [16]. Using such genome structures instead of standard linear reference genomes has been shown to reduce reference bias [6, 16] and to come with a few other, significant advantages [15, 22]. Methods presented for constructing variation graphs so far, however, require a linear reference genome as a point of departure. Here, we point out how to construct variation graphs *de novo*, by making use of multiple alignment techniques that provide graph-based representations of the (progressively constructed) multiple alignments [12]. In this, we present an approach for full-length, high-quality reconstruction of the haplotypes of a viral quasispecies that is *entirely de novo*, which, to the best of our knowledge, is a novelty.

Our method depends on the enumeration of maximal-length paths in a variation graph, whose number is exponential in the number of nodes of the graph. However, since all these paths enumerated are to respect the subpaths associated with the input contigs, their number will decrease on increasing contig length. Thanks to advances in sequencing technology, input contig length will inevitably increase, which points out that our method, as per its design, has a clear view towards future technological developments.

Benchmarking experiments demonstrate that Virus-VG yields substantial improvements over the input contigs assembled with SAVAGE [1] in terms of spanning the full length of the haplotypes. Thereby, the increase in length comes at negligible or even no losses in terms of sequential accuracy. Further, we find our strain abundance estimates to be also highly accurate. Finally, we find our method to (substantially) outperform alternative approaches, all of which are reference based—we recall that there are no alternative de novo approaches so far—both when working with bootstrap and curated reference genomes.

Note on Related Work: RNA Transcript Assembly. The problem of RNA transcript assembly has been cast in terms of variations of minimum path cover optimization problems that

are—regarding a few relevant aspects—similar in spirit to the optimization problem we formulate [5, 17, 20, 23, 26]. Most importantly, in [20] node and edge abundance errors are introduced and in [23] a minimum path cover with subpath contraints is shown to be polynomially solvable. However, to the best of our knowledge, no method employs both subpath constraints *and* abundance error minimization in its problem formulation at the same time.

2 Methods

Notation. A *variation graph* (V, E, P) is a directed graph, that is constructed from a set of input sequences, which represent members of a (evolutionarily coherent) population of sequences. Each node $v \in V$ is assigned to a subsequence $\text{seq}(v)$. An edge $(u, v) \in E$ indicates that the concatenation $\text{seq}(u)\text{seq}(v)$ is part of one of the input sequences. P is a set of paths (a sequence of nodes linked by edges) that represent genome-specific sequences; thereby, P can, but need not, represent the input sequences themselves. A node $v \in V$ with no incoming edges is called *source*. A node $v \in V$ with no outgoing edges is called *sink*. In the following, variation graphs are assumed to be *acyclic*.

Workflow. Our method consists of two basic steps:

(1) The computation of a ***contig variation graph*** $VG' = (V', E', P')$ where each path $p \in P'$ represents an input contig. We refer to the path representing contig c as $p(c)$. Together with VG', we compute a function $a' : V' \to \mathbb{R}$ where $a'(v')$ for $v' \in V'$ represents the abundance of an individual node, measured by the amount of original reads (from which the contigs were computed) that align to $\text{seq}(v')$.

(2) The transformation of VG' into a ***genome variation graph*** $VG = (V, E, P)$ where each path $p \in P$ reflects a full-length haplotype. We also compute a function $a : P \to \mathbb{R}$ where $a(p)$ for $p \in P$ reflects the abundance of the haplotype represented by p. The set of paths P together with their abundances $a(p)$ establish the final output of our method.

The input for computation of VG' in **(1)** are the contigs of a de novo viral quasispecies assembly approach, for which there are two options [1, 14]. Here, we make use of contigs from [1], which were shown to significantly outperform the contigs from [14]. For computation of a', we make use of the original reads from which the input contigs were computed; one can determine the abundance $a'(v')$ of single node $v' \in V'$ as the (length normalized) count of reads whose alignments touch upon v'.

The input for computation of VG and a in **(2)** are VG' and a'. Note first that $V \subseteq V'$ and $E \subseteq E'$ such that we can apply a' also to nodes in VG. The computation of VG is established as the solution of an optimization problem that aims to determine full-length paths (paths formed by a concatenation of contigs of maximal length) such that the difference of path abundances $a(p)$ and node abundances $a'(v)$ for paths p of which v makes part of is minimal.

We will describe the construction of the contig variation graph **(1)** in full detail in Section 2.1. The transformation into the (final) genome variation graph **(2)** is divided into two steps: **(a)** the *enumeration of candidate paths*, which is described in Section 2.2.1, and **(b)** the *solution of an optimization problem* that aims at selecting a subset of candidate paths through their path abundance values which are optimal in terms of being compatible with node abundances in section 2.2.2. The complete workflow is illustrated in Figure 1.

Implementation. Our approach, *Virus-VG*, is implemented in Python 3.5 and publicly available at `https://bitbucket.org/jbaaijens/virus-vg`.For solving the minimization problem (see 2.2.2) we make use of the LP solver, implemented in Gurobi 7.0 [10].

2.1 Contig variation graph construction

Input. The input is a data set of next-generation sequencing reads and a set of contigs assembled from these; here, we use the specialized de novo viral quasispecies tool SAVAGE [1]. We assume that there are no contigs which are an exact subsequence of another contig. The contig variation graph with its node abundances is contructed in three steps.

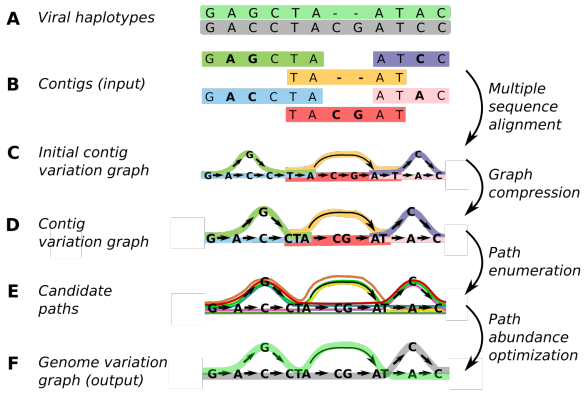

Figure 1: Virus-VG workflow.

Step 1: Multiple Sequence Alignment (MSA). We build a MSA of the contigs using the progressive sequence alignment tool POA [12]. The MSA induces a coordinate system and gives the exact placement of every contig with respect to this coordinate system. Note that POA itself models the MSA as a graph in the following way. Initially, all contigs are split into individual nodes, one node for every base, with edges between consecutive bases within a contig. Along with the nodes, also the contigs that contributed to this node are stored. Then for every alignment position, any pair of nodes which correspond to the same nucleotide are merged into a single node. When merging nodes, the corresponding edges are updated accordingly, any parallel edges are removed, and the contributing contig sets are merged. The resulting *initial contig variation graph* is acyclic. Each node represents a single nucleotide, that is assigned to one or more contigs. This step is illustrated in Figure 1, panel B to C.

Step 2: Compression and contig-path construction. We compress the initial contig variation graph similar to the construction of a compressed de Bruijn graph [13]. The absence of branches on a path ensures that every source-sink path has to traverse it at full length. Therefore, each non-branching path $(v_{i_1}, \ldots, v_{i_k})$ can be merged into a single node v'_i, with in-neighbors $N^-(v'_i) = N^-(v_{i_1})$ and out-neighbors $N^+(v'_i) = N^+(v_{i_k})$. Also the contributing contig sets of $v_{i_1}, \ldots, v_{i_k}$ are merged and stored in the new node v'_i. Note that after this step, nodes can represent a sequence instead of a single nucleotide.

In addition, we determine for each contig c the sub-path $p(c)$ in this (compressed) graph that represents c. Let $p(c) = (v_{i_1}, \ldots, v_{i_k})$ be this sub-path. Note that due to the compression step, the sequence $\text{seq}(c)$ represented by a contig c might only be a subsequence of its path sequence $\text{seq}(v_{i_1})...\text{seq}(v_{i_k})$. However, this does not bear any consequence on the definition of any haplotype the contigs make part of.

The resulting compressed graph, together with the contig paths P' is our contig variation graph $VG' = (V', E', P')$, illustrated in Figure 1, panel D.

Step 3: Node abundance. We finally compute $a' : V' \to \mathbb{R}$, which assigns *node abundances* $a'(v')$ to nodes $v' \in V'$ of the contig variation graph. Thereby, $a'(v')$ reflects the *average base coverage* of the piece of sequence $\text{seq}(v')$. For computation of $a'(v')$ we make use of the vg-toolkit [8], which allows to align the original sequencing reads to our contig variation graph. The abundance $a'(v')$ is calculated as the sum of all bases in all reads that align with $\text{seq}(v')$, divided by the length of $\text{seq}(v')$.

2.2 From contig to genome variation graph

The input for the following procedure is the contig variation graph $VG' = (V', E', P')$ together with $a' : V' \to \mathbb{R}$ that we have just described. The procedure for constructing the *genome variation graph* $VG = (V, E, P)$ from VG' and a' consists of three steps. *First*, we compute a set of *candidate paths*, which are all maximal length paths in (V', E') that are concatenations

of paths from P'. *Second*, we select a subset of candidate paths that are optimal with respect to a *minimization problem*, which provides us with the final, maximal-length paths P and path abundances $a : P \to \mathbb{R}$. *Third*, we remove nodes and edges from (V', E') that are not traversed by paths from P, which yields the final graph (V, E). Since only paths in P are supposed to reflect true haplotypes, it is reasonable to assume that any node not being included in a haplotype is a sequencing artifact. The third step is a straightforward procedure. We will describe the first two steps in more detail in the following.

2.2.1 Candidate path generation.

The goal is to compute the set of all paths through (V', E') that are maximal-length concatenations of paths from P'.We will refer to these paths as *candidate paths* P_{cand} in the following. Generating candidate paths proceeds in five steps outlined below.

Step 1: Trimming paths $p \in P'$. Due to common issues in contig computation, any uncorrected sequencing errors are often located on the extremities of the contig. We therefore shorten all paths $p \in P'$ by their extremities and remove the tails if these contain nodes v' for which $a'(v')$ is below a given threshold. By default, we allow to trim paths $p \in P'$ by a removal of nodes that together amount to no more than $\tau = 10$bp on either end.

Step 2: Enumerating path concatenations. We allow concatenating pairs of paths with matching suffix-prefix pairs. In more detail, let $p_1, p_2 \in P'$, represented by series of nodes $(u_1, ..., u_m)$ and $(v_1, ..., v_n)$ from V'. Then p_1 can be concatenated with p_2, written $p_1 \to_c p_2$, if for some l we have $u_{m-l+1} = v_1, u_{m-l+2} = v_2, ..., u_m = v_l$, that is, the suffix of length l of p_1 matches the prefix of length l of p_2.

In order to enable correction of persisting sequencing errors, we further consider to concatenate pairs of paths p_1, p_2 which do have one or more non-matching nodes, but only under the following condition. Let $u^* := u_{m-l+i} \neq v_i =: v^*$ be the respective non-matching nodes in p_1, p_2 respectively, then only if $\min\{a'(u^*), a'(v^*)\} < \alpha$, where α is a user-defined threshold, we concatenate p_1 and p_2. This threshold reflects the minimal node abundance $a'(v)$ for which we trust node v; for more details, see Appendix A.

Step 3: Removing concatenations lacking physical evidence. Subsequently, we remove concatenations $p_1 \to_c p_2$ if there are q_1, q_2 such that $q_1 \to_c q_2, q_1 \to_c p_2, q_2 \to_c p_2$, but there is no q_3 for which $p_1 \to_c q_3$ and $q_3 \to_c p_2$ and there is q_4 such that $p_1 \to_c q_4$. The situation reflects that the concatenation of paths $q_1 \to_c p_2$ enjoys corroborating physical evidence, provided by q_2, while there is no such corroborating evidence for the concatenation $p_1 \to_c p_2$. At the same time, p_1 concatenates well with q_4 such that the removal of $p_1 \to_c p_2$ does not turn p_1 into a dead end.

Step 4: Enumerating maximal length paths P_{cand}. We enumerate all feasible maximal length paths through a breadth-first search type procedure. For this procedure, we maintain a set of *active paths* P_{act}, which is the set of paths to be extended in the current iteration. We also maintain a set of *maximal paths* P_{max} that reflects the set of maximal length paths collected. See Figure 1, panel E. Only concatenations listed in Step 2 and not removed by Step 3 are considered.

1. **Initialization:** We determine all $p \in P'$ for which there are no $q \to_c p$ and put them both into P_{act} and P_{max}.
2. **Iteration:** We replace each $p \in P_{\text{act}}$ with all $q \in P'$, for which $p \to_c q$ without q^* such that $p \to_c q^* \to_c q$. Simultaneously, we extend each $\hat{p} \in P_{\text{max}}$ that ends in p, by appending the q (while respecting the overlap).
3. **Output:** If for all $p \in P_{\text{act}}$ there are no q with $p \to_c q$, we output P_{max} as our candidate path set P_{cand}.

The enumeration algorithm lists all candidate paths in time linear in the output size, which, however, may be exponential in the number of paths $p \in P'$. Because the number of paths in P' (i.e., the number of input contigs) decreases on increasing contig length, our method has a clear view towards the future, because contig length will inevitably increase due to advances in

sequencing technology.

Step 5: Correcting paths for errors. After enumerating all candidate paths, we apply a final correction step to every such path. Note that, due to concatenating paths from P' where suffix-prefix pairs do not match in all nodes, we may have positions in candidate paths where contig paths $p \in P'$ make ambiguous statements. All such ambiguous positions refer (by construction) to low abundance nodes v' (that is $a'(v') < \alpha$. We resolve the ambiguity by selecting the node v^* from all contributing paths $p \in P'$ with maximal abundance $a'(v^*)$.

2.2.2 Minimization for haplotype selection and abundance estimation

Input. For this final part of the method, the input is the set of candidate haplotype paths P_{cand} and the node abundances $a'(v)$. In general this set of paths is much larger than the actual number of haplotypes, so P_{cand} will contain many false haplotypes. Here we filter them out by estimating the abundance $a(p)$ for each path (haplotype) $p \in P_{\text{cand}}$ through solving a minimization problem. In a subsequent step, haplotype paths with an abundance of (almost) zero will be removed as being most likely false haplotypes. This leaves the set of haplotypes to be output.

Determining path abundances $a(p)$. We determine path abundance values $a(p)$ for every $p \in P_{\text{cand}}$, such as to minimize node abundance errors. Let $f(x, y)$ be an error function to be chosen later. Then for node v the node abundance error is defined as the value of $f(x, y)$ with x the node abundance $a'(v)$ and y the sum of the abundances of the haplotype paths going through the node v, which is $\sum_{p \ni v} a(p)$, in formal terms. Recall that the node abundance values $a'(v)$ are obtained from read alignments to the contig variation graph (Section 2.1, Step 3). The objective then becomes minimizing the sum of the node abundance errors over all nodes $v \in V'$:

$$\min \sum_{v \in V'} f\left(a'(v), \sum_{p \ni v} a(p)\right).$$

We need to add non-negativity contraints $a(p) \geq 0$ on the path abundances. Since we have already taken all subpath constraints into account when enumerating the candidate haplotype paths, the minimization problem does not need any further constraints.

Note that the effectiveness of this objective function depends heavily on the error function used as well as the correctness of node abundances $a'(v)$. These abundance values are not exact measurements, but based on read alignments to the graph as described above; coverage fluctuations can thus lead to under- or overestimated node abundance values. In this case, a simple linear objective function is preferred over a quadratic error function, because the former allows big errors in certain nodes to be compensated by small errors in other nodes. We also observed that normalizing the errors w.r.t. the true node abundance does not improve results, because this means that errors in nodes with low abundance values are penalized very strongly. For this reason, we use the error function $f(x, y) = |x - y|$ in our objective and the optimization problem becomes

$$\min \sum_{v \in V'} \left| a'(v) - \sum_{p \ni v} a(p) \right| \qquad \text{s.t. } 0 \leq a(p) \quad \forall p \in P_{\text{cand}}. \tag{1}$$

This is a convex programming formulation, which can be linearized and solved using the LP solver from the Gurobi Optimizer [10].

Output: haplotype selection and final abundances. The outcome of the minimization problem (1), yields for each $p \in P_{\text{cand}}$ an optimal abundance value $a^*(p)$. We now select the set of haplotype paths as output of the procedure, by removing any haplotypes with an estimated abundance below a user defined threshold γ. I.e., as output we give the set $P = \{p \in P_{\text{cand}} \mid a^*(p) \geq \gamma\}$ (Figure 1, panel F). After this haplotype selection step, we redo the optimization step on the selected haplotype paths (prefixing $a(p)$ to 0 for every path p with $a^*(p) < \gamma$), thus ensuring that our final abundance estimates are as accurate as possible.

	# strains*	target (%)	N50	N-rate (%)	MR(%)	IR(%)
SAVAGE	26	99.4	8964	**0**	**0.004**	**0**
Virus-VG	**10**	**99.9**	9251	**0**	0.006	**0**
PredictHaplo-h-ref	9	90.0	**9313**	0.004	0.402	0.010
PredictHaplo-b-ref	9	73.8	7636	0.006	0.053	**0**
ShoRAH-b-ref	639	56.9	7570	**0**	4.381	0.011

(a) 10-strain HCV mixture (simulated MiSeq)

	# strains*	target (%)	N50	N-rate (%)	MR(%)	IR(%)
SAVAGE	89	98.5	2954	0.002	**0.021**	**0**
Virus-VG	**20**	**99.5**	10202	0.002	0.067	0.001
PredictHaplo-h-ref	8	53.3	10258	0.032	0.147	0.046
PredictHaplo-b-ref	8	53.3	**10270**	**0.001**	0.121	0.004
ShoRAH-b-ref	493	26.3	10117	0.053	4.403	0.017

(b) 15-strain ZIKV mixture (simulated MiSeq)

Table 1: Assembly results per dataset. We evaluate the number of strains assembled, the fraction of the target haplotypes reconstructed, the N50 measure, the fraction of 'N's (uncalled bases), the mismatch rate (MR), and the indel rate (IR). The best score per column is marked in bold. *For SAVAGE, this column indicates the number of contigs in the assembly; since these are not full-length, this does not reflect the number of strains.

Note on related work. The minimization problem we are treating here can be considered a combination of problems presented in [20] and [23]. The combination of these problems would require an unambiguous way to have subpath abundances contribute to cumulative abundances on the nodes, which it is not immediately evident how to do so. In our setting it is straightforward how path abundances $a(p)$ contribute to the estimated abundances of the nodes on the paths. Exploring these relationships is interesting future work.

3 Results

We present results for Virus-VG on two challenging simulated data sets and compare our method with the viral quasispecies assemblers ShoRAH [28] and PredictHaplo [18], which are widely approved and state-of-the-art in terms of full-length reconstruction of viral haplotypes. We use QUAST [9] for evaluating our experiments. For parameters to be set, guidelines, their default choices, and further reasoning, see Appendix A. For an analysis of runtime and memory usage of Virus-VG, see Appendix B.

Data sets. For evaluating correctness of our algorithm and benchmarking experiments, we selected the two most challenging simulated data sets presented in [1]. Both data sets represent typical viral quasispecies ultra-deep sequencing data and consist of 2x250bp Illumina MiSeq reads which were simulated using SimSeq (`https://github.com/jstjohn/SimSeq`).

10-strain HCV mixture. This is a mixture of 10 strains of Hepatitis C Virus (HCV), subtype 1a, with a total sequencing depth of approximately 20,000x (i.e. 400,000 reads). The haplotypes were obtained from true HCV genomes in the NCBI nucleotide database and have a pairwise divergence varying from 6% to 9%. Paired-end reads were simulated at relative frequencies between 5% and 13% per haplotype, i.e., a sequencing depth of 1000x to 4600x per haplotype.

15-strain ZIKV mixture. This is a mixture of 15 strains of Zika Virus (ZIKV), consisting of

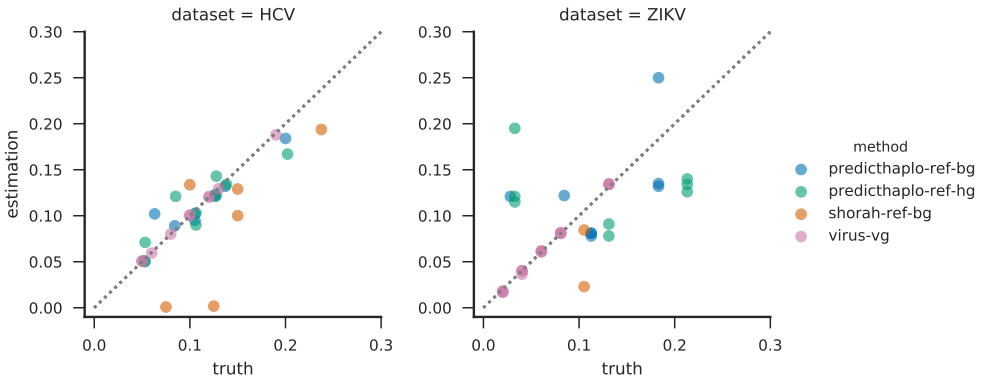

Figure 2: Haplotype abundance estimation: true frequencies versus estimated frequencies, evaluated per method, per dataset. The diagonal indicates the position of perfect estimates, i.e., estimated value equal to true value. We only plot frequencies up to 0.3 to avoid shifting the majority of points to the lower left corner due to outliers.

3 master strains extracted from the NCBI nucleotide database and 4 mutants per master strain. The pairwise divergence varies between 1% and 12% and the reads were simulated at relative frequencies varying from 2% to 13.3%. The total sequencing depth for this data set is again 20,000x.

Improvements of final haplotypes over input contigs The first two rows of Table 1a, SAVAGE and Virus-VG, display the statistics for the input contigs and the final, maximal-length haplotypes computed here, respectively, for the HCV datasets. While SAVAGE presents 26 fragmented contigs, Virus-VG presents 10 full-length haplotypes, each of which represents one of the original haplotypes, thereby encompassing the 10 original haplotypes that established the basis for simulating reads. Further, Virus-VG covers 99.9% of the target genomes, an improvement over the original 99.4% provided by the input contigs, which points out that our approach can extend contigs correctly using other (partial) contigs, resulting in longer haplotypes and a more complete assembly. The full-length haplotypes come at a negligible error rate of 0.006%. In summary, our approach yields near-perfect results on this (supposed to be challenging) dataset.

For the 15-strain ZIKV dataset (1b) we again achieve substantial improvements in terms of haplotype assembly contiguity. We obtain 20 full-length haplotypes covering all 15 strains, while the original input contig consisted of 89 highly fragmented, and relatively short contigs. We observe a slight increase in mismatch rate after applying our method, which however leaves with 0.067%, which is still extremely low. A thorough analysis turns up that this increase is due to errors in the input contigs that become more expressed only after having assembled the full-length haplotypes, so these errors are not primarily due to the method presented here. Moreover, the full-length contiguity of the haplotypes clearly offsets the minute shift in accuracy.

Comparison with the state-of-the-art Rows 3-5 display results for state-of-the-art methods PredictHaplo [18] and ShoRAH [28]. All methods were run with default parameter settings. Both of these methods are reference-guided, hence cannot immediately compared with ours, which operates entirely de novo. We decided to run both [18] and [28] by providing them with curated reference genomes ('h-ref'), accession numbers NC_004102.1 (HCV) and NC_012532.1 (ZIKV), and also on bootstrap reference genomes, computed by running [27, VICUNA], a state-of-the-art tool for generating consensus virus genomes, on the input reads ('b-ref'). We also tested alternative methods [14, MLEHaplo] and [3, QSdpR], but found them unsuitable for the (not at all unusual) datasets considered here, or unable to complete their jobs within 72 hours.

We first evaluated both [18] and [28] on bootstrap reference ('b-ref'), which simulates a de novo type scenario also for those reference-guided approaches. In both cases, we found our

method to have (quite significant) advantages, in terms of accuracy, number of strains, and strain-specific genomes covered. As was already observed earlier [1], reference-guided methods greatly depend on the quality of the reference genome provided. Considering that PredictHaplo [18] had considerable advantages over ShoRAH when operating on bootstrap reference, we also evaluated it on high-quality reference ('h-ref') [note that [28] crashed when running with high-quality reference genomes]. However, even in this case, we still found our approach to have significant advantages, again in all relevant terms, such as number of strains, target genome coverage, and accuracy in terms of error rates.

Haplotype abundance estimation We also evaluated the accuracy of the abundance estimates obtained for each haplotype. The reconstructed sequences were aligned to the ground truth sequences and assigned to the closest matching strain. For each ground truth strain, we summed the abundance estimates of the sequences assigned to it, thus obtaining a total estimate for this strain. Then we compared this estimate to the true strain abundance and computed the absolute frequency estimation errors. For each data set, we present the average error over all assembled strains. In case of any missing strains, the true frequencies were normalized first, taking only the assembled sequences into account for a fair comparison.

Our method predicts highly accurate abundances for the reconstructed strains, with an average absolute estimation error of 0.1% on the HCV data and 0.2% on the ZIKV data. Figure 2 shows the true haplotype frequencies versus the estimated frequencies per method. Note that to improve readibility, outliers (frequency ¿0.3) are not shown in this figure. We observe that Virus-VG outperforms the other methods in terms of frequency estimation, with estimates that are closest to the true values. An immediate interpretation of these findings is that accuracy in estimating abundance is inevitably linked with accuracy in haplotype reconstruction, which may explain our overall advantages.

4 Discussion

We have presented an algorithm that turns viral strain-specific contigs, such as available from [1], into full-length, viral strain-specific haplotypes, without the use of a reference genome at any point. Therefore, we first construct a contig variation graph, which arranges haplotype-specific contigs sampled from a viral quasispecies in a convenient and favorable manner. We then enumerate all maximal length paths through this graph that maximally concatenate the contig subpaths. Last, we solve a minimization problem that addresses to assign abundance estimates to maximal length paths that are optimal in terms of being compatible with abundances computed for the nodes in the graph. We finally output the optimal such paths selected together with their abundances, by which, in summary, *we have completed the de novo viral quasispecies assembly task.*

In benchmarking experiments, we have demonstrated that our method yields major improvements over the input contigs in terms of assembly length, while preserving the accuracy in terms of error rates. Compared to state-of-the-art viral quasispecies assemblers—all of which operate in a reference genome dependent manner—our method produces haplotype-resolved assemblies that are both more complete, in terms of haplotypes covered, and more accurate, in terms of error rates. We believe that (a) this reflects the strength of a fully *de novo* approach, because we avoid to deal with (as was shown sometimes overwhelming) reference-induced biases. We also believe that (b) this is a result of directly integrating haplotype abundance estimation into reconsruction of haplotypes.

Still, improvements are possible. In particular, we found our approach not suitable for genomes that exhibit repetitive elements (such as for HIV, where 5'-LTR and 3'-LTR regions are near-identical), because the repeats caused the construction of cyclic variation graphs, despite that the input contigs [1] were not subject to repeat-induced disturbances. We will address the construction of cycle-free variation graphs from virus genomes (first with limited repeat content, such as in HIV) in future work.

Further, we had already alluded to that the number of candidate paths is exponential in the number of input contigs, which could be overwhelming when dealing with highly fragmented

assembly output. We will consider more efficient, alternative solutions, which are based on additional optimization techniques (such as column generation) and/or assigning abundance values to contigs—which is an involved issue—in future work. Note finally another time that advances in sequencing technology will lead to increases in contig length, which in turn leads to less fragmented assemblies.

References

[1] J.A. Baaijens, A.Z.E. Aabidine, E. Rivals, and A Schönhuth. De novo assembly of viral quasispecies using overlap graphs. *Genome Res*, 27(5):835–848, 2017.

[2] A. Bankevich, S. Nurk, D. Antipov, A.A. Gurevich, M. Dvorkin, A.S. Kulikov, V.M. Lesin, S.I. Nikolenko, S. Pham, A.D. Prijbelski, A.V. Pyshkin, A.V. Sirotkin, N. Vyahni, G. Tesler, P.A. Pevzner, and M.A. Alekseyev. SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing. *J Comp Biol*, 19(5):455–477, 2012.

[3] S. Barik, S. Das, and H. Vikalo. Qsdpr: Viral quasispecies reconstruction via correlation clustering. *Genomics*, 2017.

[4] N. Beerenwinkel, H.F. Günthard, V. Roth, and K.J. Metzner. Challenges and opportunities in estimating viral genetic diversity from next-generation sequencing data. *Front Microbio*, 3:239, 2012.

[5] E. Bernard, L. Jacob, J. Mairal, and J. Vert. Efficient rna isoform identification and quantification from rna-seq data with network flows. *Bioinformatics*, 30(17):2447–2455, 2014.

[6] A. Dilthey, C. Cox, Z. Iqbal, M.R. Nelson, and G. McVean. Improved genome inference in the mhc using a population reference graph. *Nat Genetics*, 47:682–688, 2015.

[7] E. Domingo, J. Sheldon, and C. Perales. Viral quasispecies evolution. *Microbiol Mol Biol Rev*, 76(2):159–216, Jun 2012.

[8] E. Garrison, J. Sirén, A.M Novak, G. Hickey, J.M. Eizenga, E.T. Dawson, W. Jones, M.F. Lin, B. Paten, and R. Durbin. Sequence variation aware genome references and read mapping with the variation graph toolkit. *bioRxiv*, 2017.

[9] A. Gurevich, V. Saveliev, N. Vyahhi, and G. Tesler. QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, 29(8):1072–1075, 2013.

[10] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2016.

[11] M. Hunt, A. Gall, S.H. Ong, J.i Brener, B. Ferns, P. Goulder, E. Nastouli, P. Keane, J.A.and Kellam, and T.D. Otto. IVA: accurate de novo assembly of RNA virus genomes. *Bioinformatics*, 31(14):2374–2376, 2015.

[12] C. Lee, C. Grasso, and M.F. Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464, 2002.

[13] V. Mäkinen, D. Belazzougui, F. Cunial, and A.I. Tomescu. *Genome-Scale Algorithm Design: Biological Sequence Analysis in the Era of High-Throughput Sequencing*. Cambridge University Press, 2015.

[14] R. Malhotra, S. Wu, Manjari Mukhopadhyay, A. Rodrigo, M. Poss, and R. Acharya. Maximum likelihood de novo reconstruction of viral populations using paired end sequencing data. arXiv:1502.04239, 2016.

[15] A.M. Novak, E. Garrison, and B. Paten. A graph extension of the positional burrowswheeler transform and its applications. *Algorithms for Molecular Biology*, 12(18), 2017.

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

[17] M. Pertea, G.M. Pertea, C.M. Antonescu, T. Chang, J.T. Mendell, and S.L. Salzberg. Stringtie enables improved reconstruction of a transcriptome from rna-seq reads. *Nature Biotechnology*, 33:290–295, 2015.

[18] S. Prabhakaran, M. Rey, O. Zagordi, N. Beerenwinkel, and V. Roth. HIV haplotype inference using a propagating dirichlet process mixture model. *IEEE Trans Comp Biol Bioinf*, 11(1):182–191, 2014.

[19] M.C.F. Prosperi and M. Salemi. Qure: software for viral quasispecies reconstruction from next-generation sequencing data. *Bioinformatics*, 28(1):132–133, Jan 2012.

[20] R. Rizzi, A.I. Tomescu, and V. Mäkinen. On the complexity of minimum path cover with subpath constraints for multi-assembly. *BMC Bioinformatics*, 15(9):S5, Sep 2014.

[21] R. Rose, B. Constantinides, A. Tapinos, and D. Robertson. Challenges in the analysis of viral metagenomes. *Virus Evolution*, 2(2), 2016.

[22] Y. Rosen, J. Eizenga, and B. Paten. Modelling haplotypes with respect to reference cohort variation graphs. *Bioinformatics*, 33(14):i118–i123, 2017.

[23] Alexandru I. Tomescu, Anna Kuosmanen, Romeo Rizzi, and Veli Mäkinen. A novel min-cost flow method for estimating transcript expression with rna-seq. *BMC Bioinformatics*, 14(5):S15, Apr 2013.

[24] A. Töpfer, T. Marschall, R.A. Bull, F. Luciani, A. Schönhuth, and N. Beerenwinkel. Viral quasispecies assembly via maximal clique enumeration. *PLoS Comput Biol*, 10(3):e1003515, 2014.

[25] A. Töpfer, O. Zagordi, S. Prabhakaran, V. Roth, E. Halperin, and N. Beerenwinkel. Probabilistic inference of viral quasispecies subject to recombination. *Journal of Computational Biology*, 20(2):113–123, 2013.

[26] C. Trapnell, B.A. Williams, G. Pertea, A. Mortazavi, G. Kwan, M.J. van Baren, S.L. Salzberg, B.J. Wold, and L. Pachter. Transcript assembly and quantification by rna-seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28:511–515, 2010.

[27] X. Yang, P. Charlebois, S. Gnerre, M. Coole, N. Lennon, J. Levin, J. Qu, E. Ryan, M. Zody, and M. Henn. De novo assembly of highly diverse viral populations. *BMC Genomics*, 13(1):475, 2012.

[28] O. Zagordi, A. Bhattacharya, N. Eriksson, and N. Beerenwinkel. Shorah: estimating the genetic diversity of a mixed sample from next-generation sequencing data. *BMC Bioinf*, 12(1):119, 2011.

A Parameters

Our method requires to manually set three parameters, the minimal node abundance α, the minimal haplotype abundance γ, and the maximal trim length τ. The minimal node abundance α refers to removing mismatches when concatenating paths, see 'Correcting errors in paths $p \in P'$' in Section 2.2.1. As a general guideline, increasing α leads to increasing numbers of candidate paths, hence an increasing number of variables in the minimization problem. The greater the number of variables, the greater the chance to pick up low abundance paths, while at the same time the greater the risk to pick up haplotype artifacts.

The minimal haplotype abundance γ refers to selecting haplotypes after having solved the minimization problem in Section 2.2.2. Any haplotype $p \in P$ where $a(p) < \gamma$ will be discarded from the output.

The trim length τ refers to 'Trimming paths $p \in P'$' in Section 2.2.1. Increasing τ leads to less concatenations of paths from P', hence to less candidate paths in general, at the risk of not concatenating correctly joining contigs.

We recommend setting α and γ to 0.5% and 1.0% of the total sequencing depth of the original data set, respectively, and $\tau = 10$. These default settings were chosen according to the quality of the input contigs [1]. Given that the data sets considered here have a total sequencing depth of 20,000x, all experiments were run with $\alpha = 100, \gamma = 200, \tau = 10bp$.

B Runtime and memory usage

Since our method takes as input a set of pre-assembled contigs, the most time-consuming and memory-expensive step in viral quasispecies assembly has already been accomplished. By their worst-case runtime complexity, both candidate path generation and minimizing for selecting optimal sets of haplotypes minimization steps reflect exponential procedures. However, in practice, the runtime of the algorithm is hardly affected by these procedures, but clearly dominated by the read mapping step for computing $a' : V' \to \mathbb{R}$ when constructing the contig variation graph. This step took 4.2 CPU hours for the HCV data and 103 CPU hours for the ZIKV data, with a peak memory usage of 0.85GB and 1.0GB, respectively. Given that the read mapping step is perfectly parallelizable, these computations completed in less than five hours on a 24-core machine. For a comparison with the other methods evaluated see Table 2.

	HCV		ZIKV	
	CPU time (hours)	Peak memory usage (GB)	CPU time (hours)	Peak memory usage (GB)
SAVAGE	55	0.8	61	0.8
Virus-VG	4.2	0.9	103	1.0
PredictHaplo	2.7	1.1	7.4	1.1
ShoRAH	509	8.9	814	10

Table 2: Runtime and -space comparison between Virus-VG, SAVAGE, and the state-of-the-art. For reference-guided methods PredictHaplo and ShoRAH the values presented do not depend on the quality of the reference genome used.