bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

Viral quasispecies reconstruction via contig abundance estimation in variation graphs

With a view towards strain aware assembly of bacterial sized genomes from mixed samples

Jasmijn A. Baaijens[*], Leen Stougie[*†‡], Alexander Schönhuth[*§‡¶]

Abstract

Viral quasispecies assembly aims to reconstruct all mutant strains populating an infected patient and to provide corresponding abundance estimates. We provide a reference-genome-independent solution based on the construction of a variation graph, capturing all quasispecies diversity present in the sample. We solve the contig abundance estimation problem and propose a greedy algorithm to efficiently build full-length haplotypes. Finally, we obtain accurate frequency estimates for the reconstructed haplotypes through linear programming techniques. Our method outperforms state-of-the-art approaches in viral quasispecies assembly and has the potential to assemble bacterial genomes in a strain aware manner as well.

Keywords: viral quasispecies, haplotype reconstruction, abundance estimation, variation graph.

Background

In comparison to bacteria and eukaryotes, viruses have relatively short genomes that are subject to very high mutation rates; RNA viruses even more so than DNA viruses [1]. As a consequence, the virus particles within most RNA virus infections do not share a single genomic sequence, but rather exist as a cloud of closely related mutant strains: a *viral quasispecies* [2]. These mutant clouds enable viruses to adapt to their environment and possibly escape medical treatment or the host immune response [3, 4]. The mutant strains may show different phenotypic properties and appear at varying frequencies within the population. Viral quasispecies assembly aims to reconstruct each of these individual genomes (or *haplotypes*) and to estimate the corresponding relative abundances.

Besides viral quasispecies, many other genomic data sets contain mixtures of closely related sequences, such as bacterial mixtures or environmental samples (metagenomics). Again, strains may appear at varying frequencies, with some highly abundant and others very rare. Reconstructing all of the individual haplotypes present in such a scenario, more generally known as *haplotype aware genome assembly*, is a major challenge and requires specialized tools [5]. Here, we offer a solution to the viral quasispecies assembly problem that has the potential to scale to bacterial sized genomes.

We present VG-flow, a method for haplotype reconstruction with integrated abundance estimation. Because genomes from mixed samples are usually affected by substantial mutation rates, VG-flow avoids using standard linear reference genomes altogether. Instead, VG-Flow is based on variation graphs as underlying, flexible-to-construct reference systems that account for haplotype-specific mutations in an unbiased manner. We construct these variation graphs in a de novo manner, that is, without resorting to any external means such as a reference genome.

[*]Centrum Wiskunde & Informatica, Amsterdam, Netherlands
[†]Vrije Universiteit, Amsterdam, Netherlands
[‡]INRIA-Erable
[§]Utrecht University, Utrecht, Netherlands
[¶]Corresponding author (`alexander.schoenhuth@cwi.nl`)

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

VG-flow takes as input a next-generation sequencing (NGS) data set and a collection of strain-specific contigs assembled from the data, and produces full-length haplotypes and corresponding abundance estimates. Our method is centered on estimating contig abundances in a *contig-variation graph*, a graph that captures all quasispecies diversity present in the contigs [6, 7]. We build a *flow network* to accompany the variation graph and estimate contig abundances by solving a flow-like optimization problem: variables represent flow values on the edges of the flow network and we impose flow constraints, while the objective function evaluates the difference between estimated contig abundances and read coverage for every node in the variation graph. This objective function is convex, hence the flow problem is polynomial time solvable [8].

The flow solution presents abundance estimates for the input contigs, which are of value in its own right in various mixed sample applications [9, 10, 11]. We use the contig abundance estimates in a combination of greedy algorithms to extract candidate haplotypes from the variation graph, where candidate haplotypes reflect concatenations of subpaths associated with the input contigs. Finally, we solve an optimization problem whose variables represent the haplotype abundances and the difference between read coverage and haplotype abundance is minimized over all nodes. Thus, we obtain a selection of candidate haplotypes that represents the quasispecies, along with haplotype abundance estimates.

Existing viral quasispecies assemblers include widely evaluated tools like [12, 13, 14], as well a variety of methods introduced more recently [15, 16, 17, 18, 19, 20]. These methods can be divided into two classes: reference-guided and reference-free (also referred to as *de novo*). De novo approaches do not require any prior information, such as a reference genome or knowledge of the quasispecies composition. This has been shown to have advantages over reference-guided reconstruction, since using a reference genome can induce significant biases. Especially at the time of a viral disease outbreak, an appropriate reference genome may not be available due to high mutation rates. However, most of the aforelisted tools are reference-guided; only [16], [17], and [19] present de novo approaches.

Moreover, many of these specialized viral quasispecies assemblers aim at single gene reconstruction, rather than whole genome assembly. In [17] we took a first step towards full-length de novo viral quasispecies assembly. There, we have shown that de novo haplotype reconstruction with integrated haplotype abundance estimation yields assemblies that are more complete, more accurate, and provide better abundance estimates. While this approach is guaranteed to find a selection of haplotypes that is optimal in terms of being compatible with the read coverages, its runtime is exponential in the number of contigs. Since the number of contigs generally increases on increasing genome length, we found [17] unsuitable for genomes larger than $\sim$ 10 kb. With VG-flow, we provide a reference-free solution to the full-length viral quasispecies reconstruction problem that scales well to longer genomes. As another benefit of the theoretical rigorosity of our problem formulation and efficiency of the solution, we also experience considerable improvements in terms of accuracy compared to existing tools.

Some of the challenges that have to be dealt with in viral quasispecies assembly can also be found in RNA transcipt assembly, where the goal is to reconstruct an unknown number of transcripts and predict the relative transcript abundances. Not surprisingly, many RNA transcipt assemblers define graph optimization problems similar to our flow formulation [21, 22, 23, 24, 25]. Although dealing with related problems, these methods cannot be applied in a viral quasispecies setting so easily: they require a collection of reference genomes representing all possible haplotypes as input, which is not available in our setting. Nevertheless, the theory behind these approaches is related to what we do. In [21], node and edge abundance errors are used to define a min-cost flow problem; note that this formulation does not take subpath constraints into account. On the other hand, [22] describes how subpath constraints can be incorporated into a minimum path cover formulation.

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

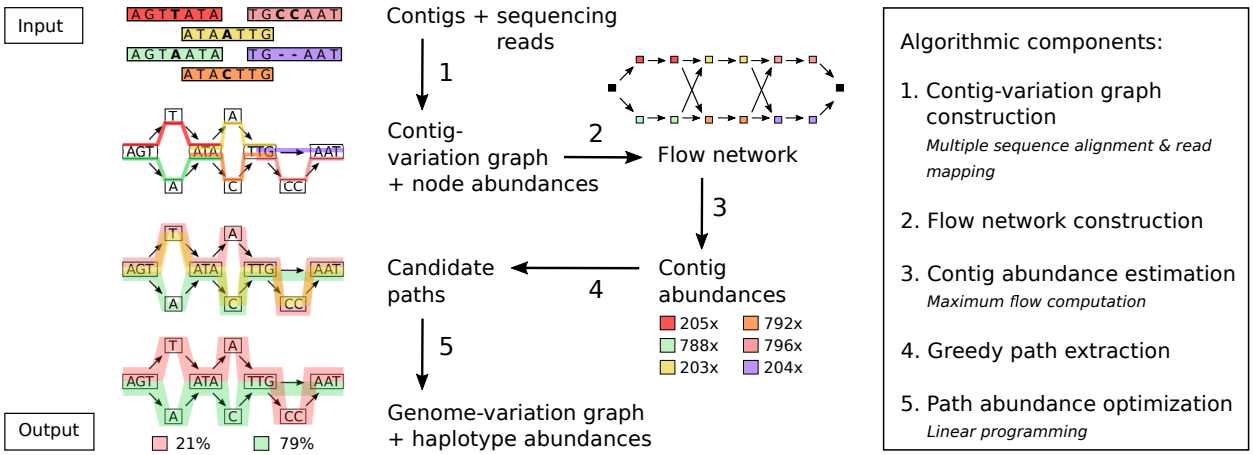

Figure 1: Algorithm overview

This results in an optimization problem that is solvable in polynomial time, but does not minimize node abundance errors. We use the best of both worlds by defining an optimization problem that takes subpath constraints, minimizes node abundance errors, and is polynomially solvable; this establishes a theoretical novelty. Because this novelty gives way to different types of analyses in other settings, and immediately connects to extensively treated theoretical issues [22, 21], we feel that it is of value also in its own right.

Among more generic assemblers, [26] has been shown to be capable of reconstructing individual haplotypes from mixed samples, up to a certain degree. This method was designed for bacterial genomes and scales well to human genomes, but is unable to reconstruct low-frequent haplotypes [16]. Haplotype aware assembly of metagenomes is a big challenge, which tends to result in scattered genome fragments and missing strains [5]. Metagenomic assemblers such as [27, 28, 29, 30] aim to reconstruct mixtures of viral and bacterial populations at strain level. The contigs obtained with these methods, or any other assembler, can also be used as input for VG-flow. Although we focus on viral quasispecies reconstruction, the mathematical framework presented here is generic and could be applied in other scenarios as well; as such, VG-flow has the potential to make a big step ahead in haplotype aware genome assembly in general.

Results

We present VG-flow, a new approach to haplotype aware genome assembly from mixed samples. This algorithm takes as input a data set of next-generation sequencing reads and a collection of strain-specific contigs; note that de novo assembly into strain-specific contigs can be performed using various tools (e.g. [16, 19, 26, 27], depending on the application). The output of VG-flow consists of maximal length haplotypes along with relative abundance estimates for each of these sequences. In addition, the algorithm yields abundance estimates for each of the input contigs.

Our approach consists of five steps, as depicted in Figure 1:

(1) We construct a *contig variation graph* VG_C by performing Multiple Sequence Alignment (MSA) on the input sequences. Node abundances are obtained by mapping sequencing reads to the variation graph.

(2) We build a *flow network* FG using VG_C.

(3) We define and solve a flow-like optimization problem on FG to obtain contig abundance estimates.

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

(4) We generate a set of candidate haplotypes P_{cand} based on the estimated contig abundances through multiple greedy heuristics.

(5) We obtain a selection of haplotypes H from P_{cand} by solving another linear optimization problem, defined on VG_C. The solution to this problem presents estimates for the relative abundances of all candidate haplotypes in P_{cand}, thereby eliminating any false haplotypes.

The final output is presented as a *genome variation graph* VG_H capturing the haplotypes in H, along with the estimated relative abundances.

Steps (1) and (5) are based on the Virus-VG algorithm [17] and are used without further adjustment. Steps (2), (3) and (4) are entirely novel. They incorporate a new problem formulation, and based on the solution of this problem, provide a way to estimate contig abundance, as part of an overall efficient alternative to the exponential brute-force routines from [17]. VG-flow easily scales to data sets of higher complexity and thus provides a clear view towards haplotype reconstruction for mixtures of bacterial strains or even metagenomic data. For further details on algorithm design we refer the reader to the Methods section.

Benchmarking preliminaries

We perform benchmarking experiments where we compare VG-flow to existing methods for full-length viral quasispecies reconstruction. In these experiments, we make use of the specialized de novo viral quasipecies assembler SAVAGE [16] for generating a set of strain-specific contigs. We compare performance of VG-flow to Virus-VG [17], another de novo approach, and to reference-guided viral quasispecies reconstruction tools PredictHaplo [12] and ShoRAH [13]. More recent viral quasispecies assemblers aBayesQR [15], QSdpR [18], and PEHaplo [19] focus on reconstruction of relatively short genomic regions and were unable to process full-length quasispecies data sets at ultra-deep coverage. We provide the reference-guided methods with a *consensus reference genome* obtained by running VICUNA [31] on the data set. This procedure simulates a de novo setting where the viral agent and its genome may be unknown. Moreover, the consensus reference sequence may be a more accurate representation of the data set under consideration than the standard reference genomes available.

We evaluate all assemblies by comparing the assembled contigs to the ground truth sequences using QUAST [32]. This assembly evaluation tool aligns the assembled contigs to the true haplotypes, which are provided as a reference, and calculates several standard evaluation metrics. For each assembly, we report the number of contigs, percent target genomes covered, N50, NG50, and error rate. If an assembly consists of only full-length contigs, the number of contigs can be interpreted as the estimated number of strains. Target genome coverage is defined as the percentage of aligned bases in the true haplotypes, where a base is considered aligned if there is at least one contig with at least one alignment to this base. The N50 and NG50 measures reflect assembly contiguity. N50 is defined as the length for which all contigs in the assembly of at least this length together add up to at least half of the total assembly size. NG50 is calculated in a similar fashion, except that the sum of contig lengths is required to cover at least half of the total target length. Error rates are equal to the sum of mismatch rate, indel rate, and N-rate (ambiguous bases). We do not report unaligned bases or misassemblies as we did not encounter any of these.

In addition to the above QUAST assembly metrics, we evaluate strain abundance estimates by comparing estimated values to true strain abundances. For each assembly, let n be the number of true strains and let x_i, x'_i denote the estimated and true abundance, respectively, of strain i. For each ground truth haplotype, the abundance estimates of sequences assigned to this haplotype were summed to obtain the strain abundance estimate x_i. We only evaluate abundances for strains that are present in the assembly (i.e. $x_i > 0$) since we cannot expect an assembler to estimate the

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

abundance of a missing strain. Therefore, the true strain abundance values x'_i are also normalized, taking only the assembled sequences into account. Then, we calculate the absolute frequency error (AFE) and the relative frequency error (RFE) a follows:

$$\begin{aligned} \text{AFE} &= \sum_{i \in I} \frac{|x_i - x'_i|}{|I|}, \\ \text{RFE} &= \sum_{i \in I} \frac{|x_i - x'_i|}{|I| \cdot x'_i}, \text{ where} \\ I &= \{i \in [n] : x_i > 0\} \end{aligned}$$

VG-flow scales well to bacterial size genomes

Our main goal of designing VG-flow was to enable generation of high-quality assemblies for data sets of higher complexity compared to what is possible with other de novo approaches. While Virus-VG [17] performs well on the quasispecies benchmarking data sets, the path enumeration step will quickly become too expensive as data sets become more complex: the number of candidate haplotypes is exponential in the number of input contigs. In particular, the number of haplotypes grows exponentially in the genome size, making candidate path enumeration infeasible for larger haplotypes.

In order to explore the limits of VG-flow and to highlight its advantages over Virus-VG, we simulated 28 data sets of increasing complexity using SimSeq [33] (2x250 bp paired-end reads, Illumina MiSeq error profile). We created data sets with genomes of increasing size (2500 bp, 5000 bp, 10.000 bp, 20.000 bp, 40.000 bp, 100.000 bp, 200.000 bp) and an increasing number of strains (2, 4, 6, 8). Each data set has a total coverage of 1000x. Each strain was created by randomly introducing mutations at a mutation rate of 0.5% into a randomly generated nucleotide sequence of the desired length; hence, haplotypes have a pairwise divergence of 1%. For data sets of 2, 4, 6, and 8 strains, the relative strain abundances were set to ratios of 1:2, 1:2:3:4, 1:2:3:4:5:6, and 1:2:3:4:5:6:7:8, respectively. Although the genome sequences are artificial, allowing us to vary genome size and number of strains flexibly, the relative abundances and pairwise divergence reflect plausible real-world, and challenging scenarios in metagenomics [5].

In Figure 2 we show VG-flow runtimes as a function of the genome size for a fixed number of strains. As expected, runtime increases as the genome size and the number of strains increase. Even the data sets with a genome size of 200.000 bp are easy to process with VG-flow. Virus-VG, on the other hand, was unable to process any genomes larger than 20.000 bp (2 strains), 5000 bp (4 strains), or 2500 bp (>4 strains).

Remark. Currently, the limiting factor for processing genomes larger than 200.000 bp with VG-flow is the pre-assembly step. VG-flow requires pre-assembled strain-specific contigs as input and we use SAVAGE [16] for this. SAVAGE has proven to produce assemblies of very high quality, but this assembler does not scale well to large genomes. Inspired by results from [16], we experimented with SPAdes [26] assemblies as input for VG-flow. Although SPAdes does not produce strain-specific contigs as well as SAVAGE, it performs reasonably well and VG-flow is able to build full-length haplotypes from these contigs. Results and further details are shown in the Supplementary Material.

VG-flow outperforms existing tools

We evaluate performance of VG-flow on three simulated viral quasispecies data sets from [17] and one real HIV benchmark presented in [34], also referred to as the *labmix*. The simulated data sets are based on true genomic sequences from the NCBI nucleotide database; the characteristics of all data sets (virus type, genome size, number of strains, relative strain abundances, and pairwise

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

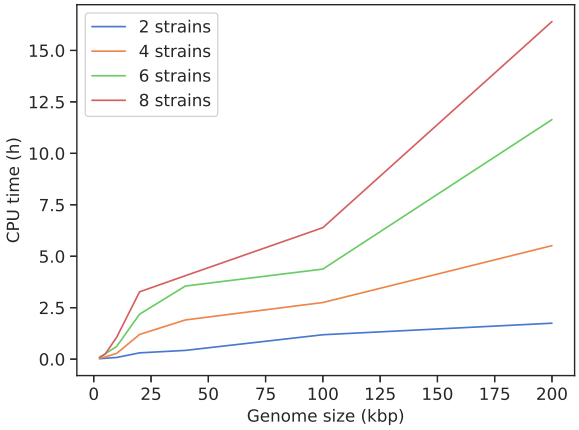

Figure 2: VG-flow runtime (CPU seconds) on data sets of increasing genome size (2500, 5000, 10.000, 20.000, 40.000, 100.000, 200.000) and number of strains (2, 4, 6, 8). Note that these runtimes do not include SAVAGE assembly time; a comparison of total runtime is shown in the Supplementary Material.

Data set	Data type	Virus type	Genome size	Strain count	Strain abundance	Pairwise divergence
HCV mix	Simulated	HCV-1a	9273–9311 bp	10	5–19%	6–9%
ZIKV mix	Simulated	ZIKV	10251–10269 bp	15	2–13%	1–10%
Poliovirus mix	Simulated	Poliovirus	7428–7460 bp	6	1.6–51%	1.2–7%
Labmix	Real	HIV-1	9478–9719 bp	5	10–30%	1–6%

Table 1: Quasispecies characteristics of benchmarking data sets. All data sets consist of Illumina Miseq reads with an average sequencing depth of 20.000x.

divergence) are described in Table 1. All data sets consist of Illumina Miseq reads with an average sequencing depth of 20.000x. For each data set, including the labmix, the true haplotypes and their relative abundances are known.

Table 2 presents assembly statistics for all methods on the three simulated data sets (HCV, ZIKV, and Poliovirus) and Table 3 presents results on the labmix. The de novo approaches VG-flow and Virus-VG both use the contigs obtained with SAVAGE as input. We observe that both methods produce full-length haplotypes for all simulated data sets, with much higher assembly N50 and NG50 values than SAVAGE. The improved assembly contiguity comes with only slightly higher error rates compared to the SAVAGE contigs. Table 2 shows that VG-flow builds contigs with even lower error rates than Virus-VG (0.108% versus 0.115% on ZIKV data and 0.036% versus 0.064% on Poliovirus data for VG-flow and Virus-VG, respectively). On the Poliovirus data set we do not only observe a lower error rate for VG-flow compared to Virus-VG, but also a higher target coverage (90.2% for VG-flow versus 80.7% for Virus-VG). The frequency estimation errors (AFE and RFE) in Table 2 show that the increase in assembly accuracy also leads to lower frequency estimation errors.

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

	# contigs*	target (%)	N50	NG50	ER(%)	AFE(%)	RFE(%)
SAVAGE	26	99.4	8964	8964	0.001	-	-
Virus-VG	10	99.3	9281	9203	0.001	0.1	0.9
VG-flow	10	99.3	9281	9203	0.001	0.0	0.2
PredictHaplo	9	73.8	7636	7608	0.059	0.9	11.3
ShoRAH	639	56.9	7570	7570	4.294	8.5	64

(a) 10-strain HCV mixture

	# contigs*	target (%)	N50	NG50	ER(%)	AFE(%)	RFE(%)
SAVAGE	100	98.8	2954	3801	0.023	-	-
Virus-VG	20	92.8	10202	10210	0.115	0.3	6.0
VG-flow	21	92.8	10193	10210	0.108	0.3	5.4
PredictHaplo	8	53.3	10270	10267	0.126	4.9	69
ShoRAH	493	26.3	10117	10117	4.392	39	229

(b) 15-strain ZIKV mixture

	# contigs*	target (%)	N50	NG50	ER(%)	AFE(%)	RFE(%)
SAVAGE	59	83.7	1089	1643	0.019	-	-
Virus-VG	14	80.7	7316	7428	0.064	0.6	12.8
VG-flow	12	90.2	7316	7428	0.036	0.3	3.5
PredictHaplo	3	16.6	7461	-	1.825	-	-

(c) 6-strain Poliovirus mixture

Table 2: Assembly results on simulated data (Illumina MiSeq, 20.000x coverage). ER = Error Rate (N's + mismatches + indels), AFE = Absolute Frequency Error, RFE = Relative Frequency Error. Frequency errors were only computed for assemblies containing at least 2 full-length haplotypes. *If contigs are full-length, this number reflects the estimated number of strains in the quasispecies.

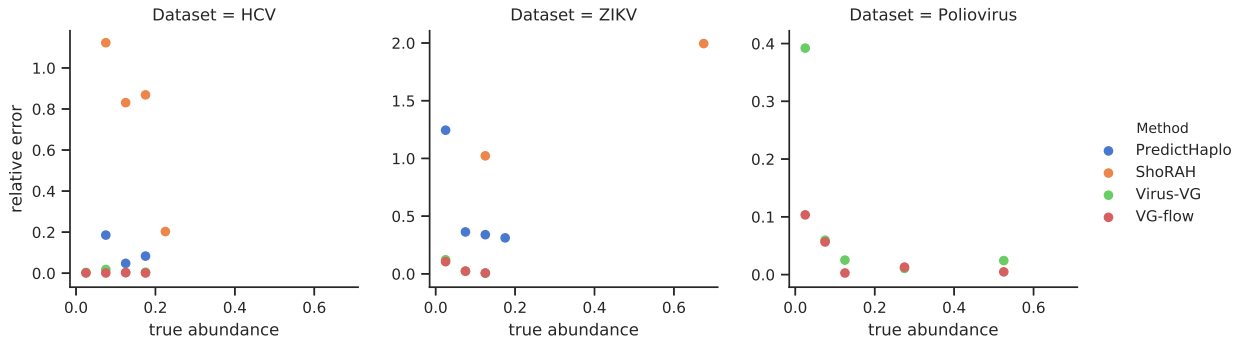

Figure 3: Abundance estimation results per data set. Abundances were only evaluated for assemblies containing at least 2 full-length haplotypes.

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

On real data (Table 3) we observe that VG-flow produces the same number of contigs as Virus-VG, leading to identical target coverage and N50 values; the only difference between the assemblies is a slightly lower NG50 for VG-flow (4608 versus 4642) and a slightly higher error rate (0.535% versus 0.324%). These differences may be explained by the highly uneven coverage of this data set, which affects the contig abundance estimation and hence also the greedy path extraction (see Figure 1). However, the contigs produced by VG-flow are much longer than the input contigs, with the N50 value more than doubled.

Compared to the state-of-the-art for full-length viral quasispecies reconstruction, we notice a clear advantage for VG-flow in terms of target coverage and error rate. Table 2 shows that PredictHaplo and ShoRAH are unable to reconstruct all haplotypes in any of the simulated data sets. The strains that could be reconstructed have higher error rates than VG-flow, as well as much higher frequency estimation errors. On the labmix, PredictHaplo and ShoRAH both achieve a target coverage of 100%. In other words, they assemble each of the five HIV strains at full-length. However, PredictHaplo does so at almost twice the error rate of VG-flow (1.066% for PredictHaplo versus 0.535% for VG-flow) and the ShoRAH assembly has an even higher error rate of 3.910%. Moreover, ShoRAH greatly overestimates the number of strains in all data sets considered.

Figure 3 shows the relative frequency estimation errors per method as a function of true abundance per strain, for each of the simulated data sets. Results are divided into bins (binsize=0.05) and average errors are shown. Figure 3 highlights the advantage of de novo methods VG-flow and Virus-VG, which have much smaller relative errors than PredictHaplo and ShoRAH. On the HCV and ZIKV data sets, VG-flow and Virus-VG show nearly identical performance; on the Poliovirus data, we observe a small advantage for VG-flow.

De novo approaches achieve highest precision and recall

In addition to the standard assembly quality metrics presented in the previous section, we analyze relevance of the reported solutions and the amount of similarity between true and reconstructed haplotypes. In the following, we define true positives by their relative edit distance to the corresponding true haplotype (i.e., edit distance divided by alignment length). A contig is considered a true positive if it aligns to a true haplotype with relative edit distance $\leq \alpha$; a haplotype is considered correctly reconstructed if at least one contig aligns to it with relative edit distance $\leq \alpha$. Figure 4 presents precision, recall, and F-measure per data set. These measures evaluate the number of true positive contigs relative to the total number of contigs (precision), the number of correctly reconstructed haplotypes relative to the total number of haplotypes (recall), and the harmonic average of precision and recall (F-measure = 2*precision*recall/(precision+recall)). We consider various thresholds for the relative edit distance and plot precision, recall, and F-measure as a function of the threshold α. Each of these measures takes values between 0 and 1, with 0 the worst possible score and 1 the best possible score.

	# contigs*	target (%)	N50	NG50	ER(%)
SAVAGE	68	97.9	1026	1450	0.066
Virus-VG	23	90.6	2130	4642	0.324
VG-flow	23	90.6	2130	4608	0.535
PredictHaplo	6	100.0	8825	8825	1.066
ShoRAH	250	100.0	8775	8775	3.910

Table 3: Assembly results on the labmix (5-strain HIV mixture, real Illumina MiSeq, 20.000x coverage).

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

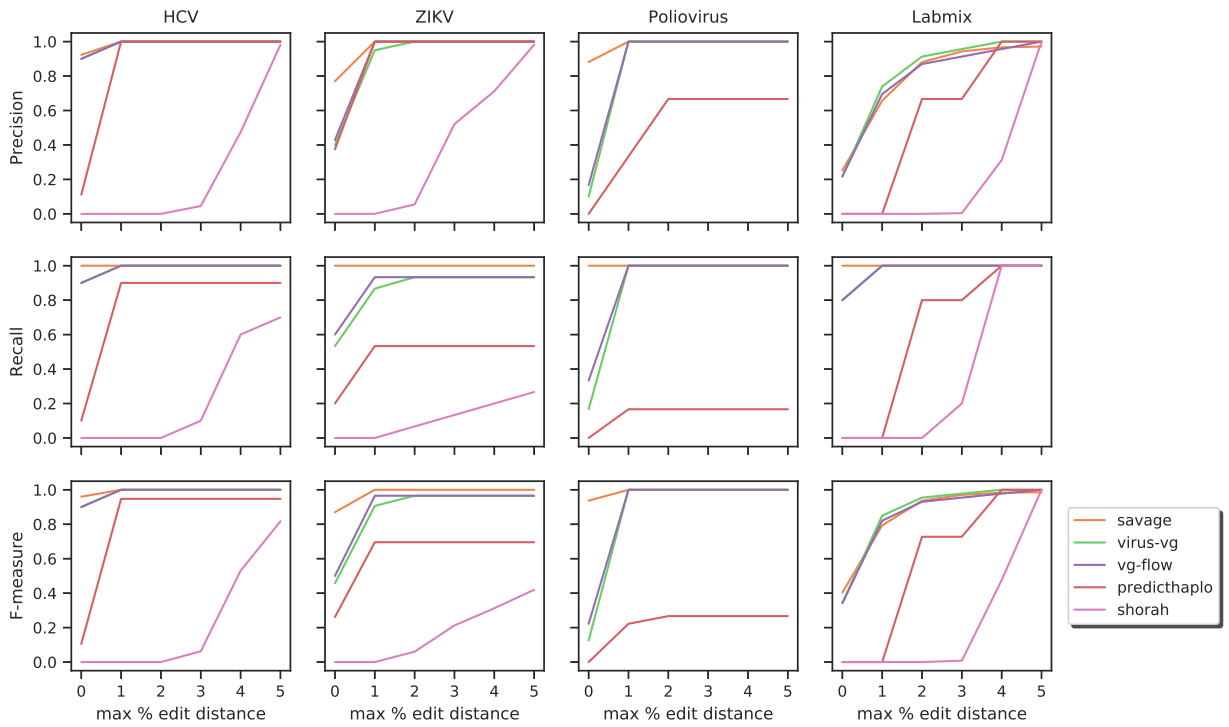

Figure 4: Precision, recall, and F-measure per data set.

We observe that, in general, SAVAGE achieves high values for all three measures already at low relative edit distance. However, SAVAGE only assembles short contigs. VG-flow and Virus-VG show very similar performance, with slightly better values for VG-flow on the ZIKV and Polio data sets, and slightly better performance of Virus-VG on the labmix. All other methods are outperformed by these de novo approaches: PredictHaplo and ShoRAH do not achieve comparable F-measure scores on the simulated data. On the labmix these methods obtain similar scores only at an allowed relative edit distance of 4%, which is nearly as high as the maximal pairwise divergence between strains in this data set.

Runtime and memory usage

The haplotype reconstruction steps used in VG-flow are highly efficient: on the benchmarking data sets presented in Table 1 we measured a decrease in haplotype reconstruction time of 9.2–92% compared to Virus-VG. However, total runtime for VG-flow is mostly determined by the contig-variation graph construction step, which involves multiple sequence alignment and read mapping. This graph construction step is shared by VG-flow and Virus-VG. Hence, when considering the complete approach on the simulated quasispecies benchmarks, we observe identical runtime and memory usage for VG-flow and Virus-VG: runtime varies between 3.6–12.5 CPU hours and peak memory usage is between 0.6–0.9 GB. Both approaches require as input pre-assembled contigs, for which we used SAVAGE. This de novo assembler constructs an overlap graph from the sequencing reads, which is a very expensive procedure (30.6–276 CPU hours). In comparison, PredictHaplo is faster (2.0–7.4 CPU hours) and ShoRAH is slower (209–814 CPU hours, unable to process the poliovirus data set). However, it is important to realize that all methods except PredictHaplo are able to profit from multithreading, leading to competitive wall clock times on sufficiently large computing clusters. Moreover, constructing a consensus reference genome to be used for reference-guided

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

methods ShoRAH and PredictHaplo also incurs some additional costs (0.07–0.44 CPU hours). More details are presented in the Supplementary Material.

Analysis of an HCV patient sample

In order to demonstrate utility of VG-flow on real data, we ran our method on a patient sample (plasma) of a Hepatitis C virus infection (subtype 1a). This sample is part of a deep sequencing initiative of HCV genomes [35]. It consists of 349268 reads (2x250 bp, Illumina MiSeq), covering the HCV reference genome (NC_004102.1) from position 2296 to 7328 with an average sequencing depth of 34704×. We performed de novo assembly with SAVAGE and obtained 133 contigs varying in length from 152 to 1238 bp, with an N50 value of 472. After running VG-flow on this set of contigs, we obtained 33 contigs with an N50 value of 2342. Among the contigs were 7 full-length haplotypes (>4500 bp), in agreement with the analysis performed by [35] using single genome amplification. The estimated relative frequencies varied between 6.0% and 33.3%. Two of the full-length haplotypes show a large insertion of 573 bp at position 3546 of the HCV reference genome; this insertion falls into the NS3 gene, which is involved in viral RNA replication through helicase activity. Overall, the 7 full-length haplotypes have 98.8–99.4% pairwise sequence identity, while they share only 93.8–94.7% of their sequences with the HCV reference genome. The overall assembly process took 92 minutes using 12 CPU's (51 minutes for SAVAGE, 41 minutes for VG-flow) and used 1.2 GB of RAM.

Discussion

Many genomic data sets contain mixtures of closely related sequences, such as viral quasispecies or bacterial populations, where the number of haplotypes is generally unknown and relative abundances may differ per haplotype. VG-flow addresses these challenges: we successfully reconstructed haplotypes from several mixed samples, both simulated and real, and obtained highly accurate frequency estimates for each haplotype.

VG-flow performs full-length haplotype aware genome assembly, without using existing linear reference genomes, but by constructing variation graphs from pre-assembled contigs. This approach establishes a reference system that allows for analyses that do not suffer from any kind of haplotype-specific mutation-induced biases. We compute abundance estimates for the input contigs, which are of value in its own right. We also enable haplotype reconstruction in polynomial time, with runtimes depending linearly on genome size in practice. We have shown that VG-flow scales well to bacterial sized genomes, hence proving its potential to contribute also to metagenomic assembly. In benchmarking experiments on simulated viral samples, our method outperformed the state-of-the-art in full-length viral quasispecies reconstruction in terms of assembly completeness, assembly accuracy, and abundance estimation quality. Finally, we also demonstrated the value of our method on real HIV and HCV data sets.

In view of general benefits of Virus-VG [17] and the fact that its brute-force solution experiences severe limitations with respect to genome size and the number of contigs, our main goal was to find a polynomial time solution producing high quality assemblies like Virus-VG. Our algorithm did not only achieve decisive speed-ups, but also improved on Virus-VG in terms of assembly accuracy. Virus-VG and VG-flow both aim to reconstruct individual haplotypes from pre-assembled contigs and perform abundance estimation, but VG-flow uses a radically different approach to generating candidate haplotypes. Our results show that the greedy path extraction step in our algorithm selects a subset of possible haplotypes that represents the quasispecies sufficiently well. By limiting the path abundance optimization to this subset of haplotypes, many false haplotypes are excluded from optimization and hence the algorithm gets less confused by false haplotypes.

Interestingly, for some data sets VG-flow was able to improve on the input contigs in terms

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

of target coverage. A possible explanation is that for haplotypes which are not fully represented by pre-assembled contigs, VG-flow is able to reuse contigs from other strains in the same region. This hypothesis is supported by the fact that error rates for VG-flow are higher than for the input contigs constructed using SAVAGE.

The results in Figure 2 show that VG-flow has the potential to process larger genomes. Currently, there are two limiting factors that prevent us from processing bacterial or metagenomic data. First, our method requires as input a collection of pre-assembled, strain-specific contigs. We obtained best results using SAVAGE [16] to generate these input contigs; however, in its current state SAVAGE does not scale well to larger genomes. Experiments using SPAdes to generate input contigs have shown that VG-flow can also generate full-length haplotypes from these assemblies. In fact, any haplotype aware assembler could be used to generate the input contigs, but the quality of the input contigs has a significant impact on the quality of the output.

Another limiting factor is that VG-flow depends on multiple sequence alignment for constructing the contig variation graph. This step can become quite expensive as the number of contigs grows, which could lead to difficulties when processing metagenomic data sets. Nevertheless, we have moved to a much wider range of feasible genome sizes than what was possible before. Note finally, that variation graph construction is part of a current, very active area of research, such that improvements on that end are to be expected [36, 6, 7].

Note that each of these factors imposes the same limitations to the approach in [17]. Addressing these challenges would make VG-flow truly capable of processing bacterial or metagenomic data. Therefore, each of the points discussed above provides an interesting starting point for future work.

Conclusions

While multiple approaches to reference-free viral quasispecies assembly have been introduced recently, efficient reconstruction of full-length haplotypes without using a reference genome is a major challenge. Although de novo methods have shown advantages over reference-guided tools, the resulting assemblies often consist of rather short contigs. In this paper, we have proposed VG-flow as an efficient solution to extend pre-assembled contigs into full-length haplotypes, based on variation graphs as reference systems that allow for a bias-free consideration of all haplotypes involved. Benchmarking experiments have shown that VG-flow outperforms the state-of-the-art in viral quasispecies reconstruction in terms of accuracy of haplotype sequences as well as abundance estimates. Moreover, we have shown that our method scales well to bacterial sized genomes, thus proving its potential for processing larger data sets like bacterial mixtures or metagenomic data.

Methods

Variation graphs

Variation graphs are mathematical structures that capture genetic variation between haplotypes in a population [6, 7]. These graphs provide a compact representation of a collection of input sequences by collapsing all shared subsequences between haplotypes.

Definition. Let S be a collection of sequences. We define the variation graph $VG(S)$ as a tuple (V, E, P, a). The nodes $v \in V$ store sequences $\text{seq}(v)$ of nucleotides (of arbitrary length) which appear as a substring of some $s \in S$. The edges $(v_1, v_2) \in E$ indicate that the concatenation $\text{seq}(v_1)\text{seq}(v_2)$ also appears as a substring of some $s \in S$. In addition to nodes V and edges E, a variation graph stores a set of paths P representing the input sequences: for every $s \in S$ there is a path $p \in P$ (i.e. a list of nodes, linked by edges) such that the concatenation of node sequences equals s. Finally, we store path abundances using an abundance function $a : P \to \mathbb{R}$ which assigns an absolute abundance value to each path in P.

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

Approach. Following [17], we distinguish between two types of variation graphs: *contig-variation graphs* and *genome-variation graphs*. Let C be a set of pre-assembled contigs and let H be the collection of haplotypes we aim to reconstruct. The contig-variation graph $VG(C) = (V_C, E_C, P_C, a_C)$ organizes the genetic variation that is present in the input contigs and the abundance function a_C gives contig abundance values for every input contig. The genome-variation graph $VG(H) = (V_H, E_H, P_H, a_H)$ stores the haplotypes within a population and the abundance function computes haplotype abundances. Constructing a genome-variation graph is the goal of our method; the key idea is to use the contig-variation graph to get there.

Contig-variation graph construction

We construct a contig-variation graph from pre-assembled contigs C using existing techniques for variation graph construction, similar to [17]. This entails three steps:

(1) Multiple sequence alignment (MSA). We run `vg msga` [7] on the input contigs; the resulting MSA is represented as a graph (V, E, P).

(2) Compactification. We compactify the graph by contracting any non-branching path into a single node. For every contig, we update the corresponding path $p \in P$ such that it stores the path through the compacted variation graph. Thus, we obtain a graph (V_C, E_C, P_C).

(3) Node abundance computation. We use `vg map` [7] to align the sequencing reads to the compacted variation graph. From these alignments we compute the average base coverage for every node in the graph, also referred to as the *node abundance*.

Note that the computed node abundances do not yet give us the abundance function $a_C : P_C \to \mathbb{R}$. To complete the construction of $VG(C)$, we construct a flow network and solve a minimum-cost flow problem as described below.

Flow network construction

We construct a flow network $FG = (V, E, c, d)$, which allows us to compute contig abundances by solving a variant of the minimum-cost flow problem. Network flows are defined on directed graphs, where every edge has a given capacity and receives a certain amount of flow [37]. A flow network has a *source* node and a *sink* node, which have only incoming and outgoing flow, respectively. For all other nodes, the amount of incoming flow must always equal the amount of outgoing flow, so-called flow conservation. We define our graph as follows.

Nodes. We start by creating a source s and a sink t. Then, we introduce two vertices for every contig $c_i \in C$, thus obtaining the vertex set $V = \{s, t\} \cup \{v_i^-, v_i^+ \mid c_i \in C\}$.

Edges. We introduce directed edges (arcs) of three types: *contig-arcs*, *overlap-arcs*, and *auxiliary-arcs*. For each contig c_i we add a contig-arc $e_i : v_i^- \to v_i^+$. For each pair of contigs c_i and c_j there is an overlap-arc e_{ij} from vertex v_i^+ to vertex v_j^- if a suffix of c_i has a non-conflicting overlap with a prefix of c_j. In other words, the sequences of c_i and c_j are identical on their overlap. Finally, we add auxiliary-arcs $s \to v_i^-$ for any v_i^- which has no incoming overlap-arcs, and auxiliary-arcs $v_i^+ \to t$ for any v_i^+ which has no outgoing overlap-arcs.

Capacities. All edges have infinite capacity.

Costs. To every edge $e \in E$, we assign a cost d_e where

$$d_e = \begin{cases} 1, & \text{for contig-arcs;} \\ -1, & \text{for overlap-arcs;} \\ 0, & \text{for auxiliary-arcs.} \end{cases}$$

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

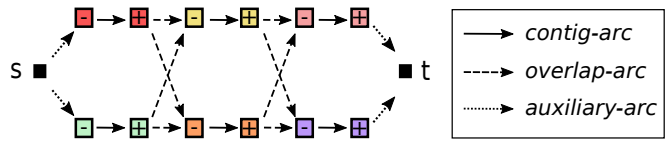

Figure 5: Flow network construction: source (s), sink (t), vertices (v_i^-, v_i^+), contig-arcs, overlap-arcs, and auxiliary-arcs.

The intuition behind this construction is that haplotypes can be found as $s-t$ paths in FG and flow along the edges reflects accumulated haplotype abundances. The edge costs in the flow network allow for the definition of a minimum-cost flow problem that computes contig abundances that are optimal in terms of being compatible with the node abundances in the contig-variation graph, as described in the next section. The construction of the flow network is illustrated in Figure 5.

Contig abundance computation

The problem of estimating contig abundances has applications in metagenomics [38] and RNA transcript assembly [39]. Existing methods make use of read mapping, either to a reference genome [9] or to the contigs [10, 11]. Such techniques may cause ambiguous alignments when contigs overlap or share identical sequence. Here, we avoid these issues by mapping reads to the contig-variation graph and solving a flow-like optimization problem.

Problem formulation. Candidate haplotypes in the contig-variation graph VG_C can be obtained by concatenating overlapping contig subpaths. Therefore, any maximal length path in the variation graph corresponds to an s-t path in FG. We denote by $\delta^+(v)$ and $\delta^-(v)$ the set of arcs, respectively, entering and leaving $v \in V$. Recall that V_C denotes the set of nodes in VG_C and let a'_u denote the abundance of node $u \in V_C$, as computed from the read alignments. For a node $u \in V_C$ and edge $e \in E$, we write $u \in e$ if the contig (or overlap) associated with the contig-arc (or overlap-arc) passes through node u in the contig-variation graph. We define the following flow problem, in which the variables x_e decide the amount of flow going through arc $e \in E$:

$$\begin{array}{ll} \min & \sum_{u \in V_C} |a'_u - \sum_{\{e \in E | u \in e\}} d_e x_e| \\ \text{s.t.} & \sum_{e \in \delta^+(v)} x_e = \sum_{e \in \delta^-(v)} x_e, \ \forall v \in V \setminus \{s,t\} \\ & x_e \geq 0, \ \forall e \in E. \end{array} \tag{1}$$

Motivation. The objective function evaluates the node abundance errors, defined as the absolute difference of the node abundance and the sum of contig abundance estimates of all contigs whose path passes through the node under consideration. However, contigs belonging to the same haplotype may have overlaps due to conserved regions between haplotypes; we need to avoid double-counting the contig abundances for nodes corresponding to such overlaps. The edge costs d_e ensure that for any pair of overlapping contigs, for any node $u \in V_C$ in the overlap, the estimated abundance is only added to the sum once.

Solution. The objective function is convex in the flow-variables x_e (Supplementary Material, Lemma 3.1), so we have the problem of minimizing a convex function over a set of linear constraints. Such problems can be solved in polynomial time [8]. Given a solution to this optimization problem, the flow values on the contig-arcs reflect contig abundance estimates. We use these values to define the abundance function a_C on the contig-variation graph. An evaluation of abundance estimates for all simulated data sets is presented in the Supplementary Material. Below, we explain how to use these contig abundances to extract candidate haplotypes for further optimization.

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

Greedy path extraction

The outcome of the above algorithm gives us a flow value for each edge in the flow network. In the biological context of the problem, we are interested in a decomposition of this flow into a set of s-t paths representing the reconstructed haplotypes. Finding such a flow decomposition can be done in polynomial time, as follows from any constructive proof of the Flow Decomposition Theorem [37]. In general, we are interested in a parsimonious solution; that is, a solution with a small number of paths. Finding a decomposition with a minimal number of paths, however, is NP-complete. Many approximation algorithms have been developed for finding a minimum path flow decomposition, e.g. [40, 41], but these algorithms could not even handle our smallest data set (a mixture of 2 haplotypes of length 2500 bp). Therefore, we resort to other, more efficient means for obtaining a set of haplotypes from the given flow solution [42].

We consider a generic greedy heuristic to obtain a selection of candidate paths (Algorithm 1). This approach iteratively selects an $s-t$ path p from the flow network, then updates the flow solution by subtracting the largest possible flow on contig-arcs in p. It terminates when all flow on contig-arcs is below a user-defined threshold for the minimal haplotype abundance. The order in which paths are selected depends on the optimality criterion: we consider maximum capacity paths, minimum capacity paths, and shortest paths—this essentially gives rise to three heuristics. Note that we do not take the flow values on overlap-arcs or auxiliary-arcs into account, because we want to avoid any preliminary restrictions on the contig overlaps used.

Algorithm 1 Greedy path extraction given a flow solution

Input: flow network FG, contig-arcs E', flow solution x, min abundance m, optimality criterion
Output: a selection of candidate paths P_{cand}

function GreedyPaths(FG, E', x, m, opt)
  $R \leftarrow x$
  $P_{\text{cand}} \leftarrow \emptyset$
  $FG_R \leftarrow FG$
  while FG_R has at least one $s-t$ path **do**
    Find an $s-t$ path p in FG_R that is optimal w.r.t. opt
    $w \leftarrow \min_{e \in p \cap E'} \{R_e\}$
    $R \leftarrow R - wp$
    $FG_R \leftarrow FG \setminus \{e \in E' : R_e < m\}$
    $P_{\text{cand}} \leftarrow P_{\text{cand}} \cup \{p\}$
  return P_{cand}

It varies per data set which optimality criterion gives best results: the maximal capacity criterion extracts paths in order of decreasing abundance, hence leads to paths which are most reliable. However, if a sample contains low-frequency strains, it can be beneficial to select haplotypes in order of increasing abundance (minimum capacity). Since we do not know the composition of the quasispecies beforehand, we combine the results of all three heuristics into one set of candidate haplotypes for further optimization. Earlier work has shown that merging a pool of high quality approximations allows for efficient solutions to well-known optimization problems [43, 44]. We compare performance of our combined approach and the individual greedy heuristics in the Supplementary Material.

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

Path abundance optimization

Given a collection of candidate haplotypes P_{cand} in the form of paths through the contig-variation graph, the only task remaining is to compute relative abundances for these haplotypes. Although the greedy path extraction algorithm produces preliminary path abundance estimates, these can be improved by the following linear programming approach, also described in [17].

Problem formulation. Let a'_v denote the abundance of node $v \in V_C$, which was computed from the read alignments to VG_C. We define variables $x_p \in \mathbb{R}_{\geq 0}$ for $p \in P_{\text{cand}}$, representing the estimated abundance for haplotype p, and consider the following optimization problem:

$$\min \sum_{v \in V_C} \left| a'_v - \sum_{p \ni v} x_p \right| \qquad \text{s.t. } x_p \geq 0 \quad \forall\, p \in P_{\text{cand}}. \tag{2}$$

The objective function is similar in spirit to the objective in Equation (1), where abundance estimation errors are evaluated per node in the contig-variation graph. Only now, we compute the absolute difference between the node abundance value and the sum of abundance estimates for all haplotypes passing through this node. This is a convex programming formulation, which can be linearized and solved using an LP solver.

Genome-variation graph construction

Given the candidate haplotypes P_{cand} and the abundance estimates x_p for $p \in P_{\text{cand}}$, we obtain a final selection of haplotypes $H = \{p \in P_{\text{cand}} : x_p \geq m\}$. Here, m is a user-defined minimal path abundance, by default set to 1% of total sequencing depth. Given H, we can transform the contig-variation graph VG_C into the genome-variation graph VG_H, a complete representation of the viral quasispecies.

Data simulation

All synthetic data sets were generated using the software SimSeq [33] to simulate Illumina MiSeq reads from the genome of interest. In order to obtain realistic sequencing error profiles, we used the MiSeq error profile provided with the software. The genomes used for each data set are listed in the Supplementary Material.

Abbreviations

bp: Base pair
MSA: Multiple sequence alignment
HCV: Hepatitis C virus
ZIKV: Zika virus
HIV: Human immunodeficiency virus
ER: Error rate
RFE: Relative frequency error
AFE: Absolute frequency error

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

Availability of data and material

Software and analysis scripts are publicly available at https://bitbucket.org/jbaaijens/vg-flow. The synthetic benchmarking datasets analysed during the current study are available at https://bitbucket.org/jbaaijens/savage-benchmarks. The real HIV data (labmix) is available at https://github.com/cbg-ethz/5-virus-mix. The real HCV data are available in the Sequencing Read Archive under accession number SRR3951347.

Competing interests

The authors declare that they have no competing interests.

Funding

This work was supported by the Netherlands Organisation for Scientific Research (NWO) through Vidi grant 679.072.309 and Gravitation Programme Networks 024.002.003.

Authors' contributions

JAB and LS designed the algorithmic components. JAB and AS designed the experiments. JAB implemented the software and performed experiments. JAB wrote the manuscript, with the help of LS and AS. All authors read and approved the manuscript.

Acknowledgements

Not applicable.

References

[1] S. Duffy. Why are RNA virus mutation rates so damn high? *PLOS Biol*, 16(8):1–6, 08 2018.

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

[3] S. Crotty, C.E. Cameron, and R. Andino. RNA virus error catastrophe: direct molecular test by using ribavirin. *Proc. Natl. Acad. Sci. U.S.A.*, 98(12):6895–6900, 2001.

[4] M. Vignuzzi, J.K. Stone, J.J. Arnold, C.E. Cameron, and R. Andino. Quasispecies diversity determines pathogenesis through cooperative interactions in a viral population. *Nature*, 439:344–348, 2006.

[5] A. Sczyrba et al. Critical assessment of metagenome interpretation - a benchmark of metagenomics software. *Nat Methods*, 14:1063–1071, 2017.

[6] 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.

[7] E. Garrison, J. Sirén, A.M Novak, G. Hickey, J.M. Eizenga, E.T. Dawson, W. Jones, S. Garg, C. Markello, M.F. Lin, B. Paten, and R. Durbin. Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nat Biotechnol*, 36:875–879, 2018.

[8] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994.

[9] M.S. Lindner and B.Y. Renard. Metagenomic abundance estimation and diagnostic testing on species level. *Nucleic Acids Res*, 41(1):e10, 2012.

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

[10] M. Fischer, B. Strauch, and B.Y. Renard. Abundance estimation and differential testing on strain level in metagenomics data. *Bioinformatics*, 33(14):i124–i132, 2017.

[11] N.L. Bray, H. Pimentel, P. Melsted, and L. Pachter. Near-optimal probabilistic RNA-seq quantification. *Nat Biotech*, 34:525–527, 2016.

[12] 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.

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

[14] 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.

[15] S. Ahn and H. Vikalo. aBayesQR: A bayesian method for reconstruction of viral populations characterized by low diversity. *J Comput Biol*, 25(7):637–648, 2018.

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

[17] J.A. Baaijens, B. van der Roest, J. Köster, L. Stougie, and A. Schönhuth. Full-length de novo viral quasispecies assembly through variation graph construction. *Bioinformatics*, to appear, 2019.

[18] S. Barik, S. Das, and H. Vikalo. Qsdpr: Viral quasispecies reconstruction via correlation clustering. *Genomics*, 110(6):375 – 381, 2018.

[19] J. Chen, Y. Zhao, and Y. Sun. De novo haplotype reconstruction in viral quasispecies using paired-end read guided path finding. *Bioinformatics*, 34(17):2927–2935, 2018.

[20] Sergey Knyazev, Viachaslau Tsyvina, Andrew Melnyk, Alexander Artyomenko, Tatiana Malygina, Yuri B. Porozov, Ellsworth Campbell, William M. Switzer, Pavel Skums, and Alex Zelikovsky. Cliquesnv: Scalable reconstruction of intra-host viral populations from ngs reads. *bioRxiv*, 2018.

[21] A.I. Tomescu, A. Kuosmanen, R. Rizzi, and V. Mäkinen. A novel min-cost flow method for estimating transcript expression with RNA-seq. *BMC Bioinformatics*, 14(5):S15, Apr 2013.

[22] 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, 2014.

[23] 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.

[24] 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. *Nat Biotechnol*, 33:290–295, 2015.

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

[25] 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. *Nat Biotechnol*, 28:511–515, 2010.

[26] 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.

[27] S. Nurk, D. Meleshko, A. Korobeynikov, and P.A. Pevzner. metaSPAdes: a new versatile metagenomic assembler. *Genome Res*, 27(5):824–834, 2017.

[28] S. Boisvert, F. Raymond, E. Godzaridis, F. Laviolette, and J. Corbeil. Ray meta: scalable de novo metagenome assembly and profiling. *Genome Biol*, 13(12):R122, 2012.

[29] Y. Peng, H.C. Leung, S.M. Yiu, and F.Y. Chin. Meta-IDBA: a de novo assembler for metagenomic data. *Bioinformatics*, 27(13):i94–i101, 2012.

[30] Dinghua Li, Chi-Man Liu, Ruibang Luo, Kunihiko Sadakane, and Tak-Wah Lam. MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics*, 31(10):1674–1676, 01 2015.

[31] 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.

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

[33] John St. John. An illumina paired-end and mate-pair short read simulator. https://github.com/jstjohn/SimSeq, 2014. Accessed: 2018-02-12.

[34] F. Di Giallonardo, A. Töpfer, M. Rey, S. Prabhakaran, Y. Duport, C. Leemann, S. Schmutz, N.K. Campbell, B. Joos, M.R. Lecca, A. Patrignani, M. Däumer, C. Beisel, P. Rusert, A. Trkola, H.F. Günthard, V. Roth, N. Beerenwinkel, and K.J. Metzner. Full-length haplotype reconstruction to infer the structure of heterogeneous virus populations. *Nucleic Acids Res*, 42:e115, 2014.

[35] D.L. Hedegaard et al. High resolution sequencing of hepatitis c virus reveals limited intrahepatic compartmentalization in end-stage liver disease. *J Hepatol*, 66(1):28–38, 2017.

[36] T. Marschall et al. Computational pan-genomics: status, promises and challenges. *Brief Bioinform*, 19(1):118–135, 2018.

[37] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[38] H. Li. Microbiome, metagenomics, and high-dimensional compositional data analysis. *Annual Review of Statistics and Its Application*, 2(1):73–94, 2015.

[39] A. Conesa, P. Madrigal, S. Tarazona, D. Gomez-Cabrero, A. Cervera, A. McPherson, et al. A survey of best practices for RNA-seq data analysis. *Genome biology*, 17:13, 2016.

bioRxiv preprint doi: https://doi.org/10.1101/645721. this version posted May 24, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder. It is made available under a CC-BY-ND 4.0 International license.

[40] M. Shao and C. Kingsford. Theory and a heuristic for the minimum path flow decomposition problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, PP(99):1–1, 2017.

[41] K. Kloster, P. Kuinke, M.P. O'Brien, F. Reidl, F. Sánchez Villaamil, B.D. Sullivan, and A. van der Poel. A practical fpt algorithm for flow decomposition and transcript assembly. *CoRR*, abs/1706.07851, 2017.

[42] B. Vatinlen, F. Chauvet, P. Chrtienne, and P. Mahey. Simple bounds and greedy algorithms for decomposing a flow into a minimal set of paths. *European Journal of Operational Research*, 185(3):1390–1401, 2008. cited By 19.

[43] William Cook and Paul Seymour. Tour merging via branch-decomposition. *INFORMS J Comput*, 15(3):233–248, 2003.

[44] T. Bosman. A solution merging heuristic for the steiner problem in graphs using tree decompositions. In Evripidis Bampis, editor, *Experimental Algorithms*, pages 391–402, Cham, 2015. Springer International Publishing.