Overlap graph-based generation of haplotigs for diploids and polyploids - Supplementary Material

Jasmijn A. Baaijens [1] and Alexander Schönhuth [1,2,*]

[1]Centrum Wiskunde & Informatica, Amsterdam, Netherlands
[2]Utrecht University, Utrecht, Netherlands.

Contents

1 Read overlap graph construction

POLYTE follows the overlap-layout-consensus (OLC) paradigm, hence we start by constructing a *read overlap graph* which is used for error correction of the input sequences. Computation of the edges for the read overlap graph requires enumeration of all pairwise suffix-prefix overlaps (of sufficient length) between the single read ends $R \in \mathcal{R}$. With this computation we aim to make the first step towards identifying read ends that stem from identical haplotypes. Because read ends are affected by sequencing errors, sequences that stem from identical haplotypes do not need to be identical; hence, we need to compute approximate rather than exact overlaps.

1.1 Approximate suffix-prefix overlaps

Computation of approximate suffix-prefix overlaps is known to be a computationally heavy and involved issue. Nevertheless, a few approaches have been presented in the recent past that enable this computation for up to several thousands reads at a time, without exceeding ordinary computational resources. Here, we make use of the most recent version of an algorithm for finding all approximate suffix-prefix overlaps in sufficiently short runtime, based on an FM-index in combination with most advanced suffix filtering techniques [4]. Because an implementation had not been available, we have implemented this most recent version—which in our experiments indeed led to considerable speed-ups over prior approximate suffix-prefix overlap computation approaches—and integrated it into POLYTE. The source code is publicly available[1] and the software can also be installed as a Bioconda package[2].

1.2 Overlap quality score

We compute a quality score $\mathrm{QS}(R_i, R_j)$ for each pair of sequences for which a sufficiently good overlap was established during the approximate suffix-prefix overlap computation that is supposed to reflect that R_i and R_j stem from identical haplotypes. Every pair of single read ends for which $\mathrm{QS}(R_i, R_j) \geq \delta$ is turned into an edge in the overlap graph. The optimal value for the threshold δ was studied before [1, 2, 5] and based on these observations, but also taking into account the low coverage setting considered here, we use $\delta = 0.95$.

Let R_i, R_j be two single-end reads sequencing reads and let l be the position on R_i at which the overlap with R_j starts. For any position k of the overlap, let $q_i[k]$ and $q_j[k]$ be the respective probabilities that bases $R_i[k]$ and $R_j[k]$ were sequenced incorrectly, as indicated by the base calling quality scores (PHRED). We assume that if a base was called wrongly, then each of the remaining bases is equally likely. Now, let $Q_i[k][X]$ represent the probability that the underlying DNA sequence of R_i equals base X at position k, for $X \in \{A, C, T, G\}$, then

$$Q_i[k][X] = \begin{cases} 1 - q_i[k] & \text{if } X = R_i[k] \\ q_i[k]/3 & \text{otherwise.} \end{cases} \tag{1}$$

Using $Q_i[k], Q_j[k]$, the probability that the true sequences underlying R_i and R_j agree for each position k of their overlap equals

$$P(R_i, R_j) = \prod_{k \in [L]} \sum_{X \in \{A,C,T,G\}} Q_i[k][X] \cdot Q_j[k][X], \tag{2}$$

[1] `https://github.com/sirkibsirkib/rust-overlaps`

[2] `http://bioconda.github.io/recipes/rust-overlaps/README.html`

Figure 1: Transitive edges and a clique of size four.

where $L := |R_j| - l$ is the length of the overlap. Since $P(R_i, R_j)$ depends on the length of the overlap L, the overlap score of R_i, R_j is defined as

$$QS(R_i, R_j) = \sqrt[L]{P(R_i, R_j)}. \tag{3}$$

This quality score is based on empirical statistics that indicate how likely a particular score $\mathrm{QS}(R_i, R_j)$ is to indicate that R_i, R_j indeed stem from identical haplotypes, which ensures that edges are of utmost quality in terms of haplotype identity.

1.3 Read orientations

Following [1], we orient the edges of the read overlap graph, which is necessary because reads can stem from either the forward or the reverse stand. When merging multiple reads into one consensus sequence, we make sure that reads agree on their respective orientations. Therefore, we apply a read orientation routine that assigns a label $(+/-)$ to every read, indicating the orientation in which its sequence should be considered. This routine starts by setting the orientation of a node of minimal in-degree to $+$, then recursively labels all out-neighbors as defined by the corresponding edges. When there is no perfect labeling possible, meaning that there are conflicts among the read orientations due to inversions, we heuristically search for an orientation that leads to a minimal amount of conflicts among the reads. We do so by applying the above labelling algorithm 100 times, each time randomly selecting a start-node, then selecting the best labeling. In practice, however, we often find a perfect labeling.

1.4 Double transitive edges

Finally, we simplify the overlap graph by systematically removing double transitive edges. The number of maximal cliques in an overlap graph grows exponentially with the number of nodes in the graph, that is here, with the read coverage of the dataset giving rise to the overlap graph. While our method relies on cliques for the purpose of error correction, the size of the cliques does not have to exceed a certain threshold for that goal.

A common approach to reduce the complexity of an overlap graph is to remove transitive edges. An edge $u \to w$ is called *transitive* if there exist a vertex v and edges $u \to v$, $v \to w$. We call an edge $u \to w$ *double transitive* if there exists a vertex v and *transitive* edges $u \to v$, $v \to w$. Note that, by definition, any double transitive edge is also single transitive. In practice, removing double transitive edges bounds the size of the cliques to 4, thus decisively limiting the number of maximal cliques and allowing efficient maximal clique enumeration, while still allowing for safely distinguishing errors from true variants [1].

To find all double transitive edges, we first remove all non-transitive edges from the overlap graph to obtain the transitive graph G'. This can be done efficiently by computing the inner product of a_u^- and a_v^+ for all pairs $(u, v) \in V \times V$, where a_u^- (resp. a_v^+) is the adjacency vector of outgoing (resp. incoming) edges of u (resp. v). Applying this procedure to G we obtain G', and to find all double transitive edges we apply the same procedure to G'.

2 Read-based branch reduction

2.1 Enumerating branching components

The set of all branching edges in the contig overlap graph can be partitioned into branching components. This partition can be found in time linear in the number of branching edges by processing all branching edges one by one while building components. For a given edge $u \to v$, we add all edges $u \to v'$ for $v' \in V$ to the same component, as well as all edges $u' \to v$ for $u' \in V$. This process is repeated iteratively for the newly added edges until all possible edges have been added; the component is now complete. We keep track of all edges that were already assigned to a component and proceed to the next unassigned edge to find a new component.

2.2 Evidence threshold for branching edges

The minimal amount of evidence required for a branching edge depends on the expected number of reads linking the variants. This in turn depends on the minimal coverage per haplotype, the distance between the two variants to be bridged, and the insert size distribution of the sequencing reads. Assuming random sequencing errors at a rate of at most 1%, we calculate the ideal evidence threshold for read-based branch reduction for any possible distance between variants, given the haplotype coverage and the insert size distribution. As the coverage per haplotype increases, more evidence is required because the likelihood of multiple reads sharing the same sequencing error increases.

The evidence thresholds are calculated as follows. Given a branch in the graph which is the result of a sequencing error, we want the probability that there is sufficient evidence in the original sequencing reads to be less than 10^{-3}. Since the sequencing error we consider was already built into a contig, there must be at least one read with exactly this error. Suppose we use a threshold t for the minimal amount of evidence, then we need to compute the probability of having at least another $t - 1$ reads showing the error.

Let c the coverage per haplotype and let d be the distance between two variants. For the two read ends of a paired-end read to cover one variant each, given the start position of the fragment, there is a limited range in which the insert size of the corresponding fragment is allowed to be. We use the insert size distribution to get the probability p_i that the insert size falls within this range for start position i and compute the average probability p_{av} over all l possible start positions of the fragment:

$$p_{\text{av}} = 1/l \sum_{i=1}^{l} p_i$$

Finally, we multiply the haplotype coverage c by this average probability p_{av} of having the correct insert size, giving us the expected number of paired-end reads bridging the variants. If d is smaller than the read length l, we also need to consider evidence from single read ends, so we add a term $c(l - d)/l$ to the expected number of reads. In other words, if we let k be the expected number of sequences (single- or paired-end) bridging the variants, hence providing evidence, then:

$$k = c\,(p_{\text{insert}} + \max(l - d, 0)/l)$$

We expect k reads covering the position of the sequencing error, one of which definitely showing

the error. Hence, the probability p_t of at least $t-1$ out of $k-1$ reads also showing the error equals

$$p_t = \sum_{s=t-1}^{k-1} \binom{k-1}{s} \varepsilon^s (1-\varepsilon)^{k-1-s}$$

where ε represents the sequencing error rate. We calculate p_t for increasing values of t, until $p < 10^{-3}$; the corresponding value of t gives the required evidence threshold.

2.3 Tips and inclusions

A tip node in the overlap graph is a node which has in-degree and/or out-degree equal to zero; in other words, it is a dead end in the graph. Tips are likely to be the result of erroneous contigs, hence many assembly algorithms remove these nodes from the graph before assembly. However, there is always a risk in removing such nodes, because there is also a probability that the corresponding contigs are correct. With read-based branch reduction, we do not need to remove tips at all: the resulting branch will be resolved based on read evidence rather than just graph structure.

Haplotypes may share part of their sequence due to a conserved region. If such a region is very short, it can result in inclusions in the overlap graph, i.e., some contigs are fully contained within others. In order to assemble both haplotypes, it can be beneficial to keep the inclusions and allow them to merge with both haplotypes. Again, this is achieved through read-based branch reduction, since there will be evidence for both branching edges.

3 Diploid mode

Knowing that a given sample is diploid is a very strong piece of information when performing haplotype assembly. We have developed a special module which can be activated for diploid samples. It extends the POLYTE pipeline by two additional steps after the standard algorithm has terminated: construction of a diploid contig graph, followed by contig extension. In these additional steps, we use the knowledge that the sample is diploid to resolve additional branches (for which there was insufficient evidence in the read set to resolve them during the read-based branch reduction step.

In overlap graphs from diploid samples we typically see two types of branching components; Figure 2 illustrates both types (Panel A and B) and gives an example of a possible collection of contigs giving rise to the corresponding branching component. In both situations we have four contigs, two from each haplotype, which have identical sequence where the contigs overlap. In diploid mode, a single read of evidence may already be considered sufficient, depending on the amount of evidence found for the other edges. We distinguish several cases and handle them as follows:

(i) If there is zero evidence for all edges, all edges are removed.

(ii) If there is evidence for exactly one edge in the component, we keep this edge as well as the edge connecting the two remaining nodes.

(iii) If there is evidence for exactly two edges which have disjoint node sets, we keep both edges and remove the unsupported edges.

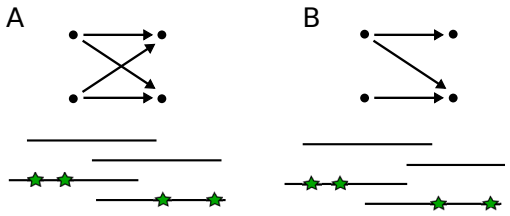

Figure 2: Typical branching components in diploid assemblies: four contigs, two from each haplotype, having identical sequence in their overlap. Depending on the contig lengths, all contigs overlap (panel A) or only a subset of the contigs overlap (panel B).

(iv) If there is evidence for exactly two edges which do not have disjoint node sets (e.g. the pair of edges leaving the top-left node of example B, Figure 2), we keep both supported edges if the difference in respective evidence counts is at most half the minimal amount of evidence required. Otherwise, the edge of highest support is kept, together with the edge connecting the two remaining nodes (if it exists).

(v) If there is evidence for more than two edges, we count the evidence per non-conflicting edge pair (i.e. disjoint node sets). We only keep the edge pair with the highest evidence count.

This procedure is more risky than default branch reduction, because it does not require similarly stringent read evidence. Therefore, we always run the main POLYTE algorithm until convergence before turning to diploid mode. This ensures that all evidence in the original reads has been exploited first.

4 Reference-guided read binning and parallelization

Overlap graph construction is a computational runtime intensive step during the assembly process, since the number of suffix-prefix overlaps is quadratic in the number of reads involved. Therefore, we reduce the computational load by sensibly grouping reads ('binning') such that haplotype-specific assembly can be safely carried out within the groups; the corresponding assemblies are merged across groups only in later steps. Beyond enabling subsequent runtime heavy steps, binning reads also allows for parallelizing our approach, implying significant advantages in terms of runtime overall.

In more detail, we limit the input for the approximate suffix-prefix overlap computation step (Section 1). In low coverage settings (of up to 200x coverage), this step cannot conveniently handle genomes of more than about 50kb in length (depending on the amount of coverage). In order to process larger regions, such as the MHC region (approx. 6Mb) or even whole chromosomes (on the order of 100Mb), we group the reads into bins, where each bin represents a region of approximately 10kb.

For computing such bins, we require a reference genome. We align all reads against the reference genome, and subsequently bin the reads according to whether their alignments belong to windows of 10kb in length (plus 1kb overlap with the neighboring windows). When confronted with multiple alignments, we assign the read to each bin possible. Note that for such cases POLYTE automatically resolves any issues, because it only integrates reads into haplotigs if their sequence content does not

lead to conflicts—if reads do not fit at all, they are discarded. Any unaligned reads are discarded as well.

Note that after dividing reads into bins, we discard the reference genome and any related information (such as read-to-reference alignments) entirely and run POLYTE fully de novo on each of the bins individually. Finally, we merge the resulting contigs across bins; for tracking the overlaps among contigs from different bins one can make use of the overlap information referring to reads that span the overlaps of the bin-specific windows.

5 Reconstructing contigs from phased VCF files

Reference-guided methods (Phaser, WhatsHap, and HapCut2) take as input a VCF file and output a phased VCF file. In order to construct haplotigs from these VCF files, we use the corresponding reference genome and the *phase set information* provided in the VCF. The phase set indicate which variants were phased in the same block. This tells us when a new contig starts: we process the variants sorted by position on the reference, and as soon as a variant belongs to a different phase set compared to the previous variant, we start a new contig. Unphased variants do not lead to a new contig but are assigned randomly to the current contigs. For creating contig sequences per haplotype block we use `bcftools consensus`.

6 Assembly evaluation

We evaluate assemblies using QUAST [3], a quality assessment tool for genome assemblies. By the above reconstruction procedure for contigs from phased VCF files, we obtain contigs per haplotype, even if haplotypes have a region in common. For de novo assemblies, on the other hand, this is not the case: if haplotypes share part of their sequence, this sequence will be assembled at most once. Therefore, we add the flags `--ambiguity-usage all --ambiguity-score 0.999` when evaluating de novo assemblies, but not for evaluating reference-guided assemblies.

7 Extended results

7.1 Misassemblies

Table 1 shows the results from the main manuscript, extended with a final column that presents the raw number of misassemblies. Relative to the application it should be preferable to opt for an assembler that distributes assembly errors over many contigs whereas another assembler yields only very little but fairly broken assemblies, with the vast majority of contigs however being correct.

7.2 Polymorphic positions

For practical applications, it is also important to know whether errors are found at relevant genetic locations (in addition to the assembly measures reported in Table 1). Polymorphic positions are the most interesting positions when performing haplotype-aware genome assembly, hence, we evaluate the number of SNPs that is included in the reconstructed haplotypes. SNP positions between the two haplotypes were obtained using the MUMmer package (nucmer + show-snps) and aligned all

	HC (%)	N50	NGA50	ER (%)	NR (%)	MC (%)	# misassemblies
Simulated data							
POLYTE	92.4	4397	4394	0.035	0	0	0
SGA	73.4	3444	-	0.025	0	0	0
SPAdes	84.1	3588	919	0.032	0	0.0	1
SPAdes-dip	83.6	3294	903	0.003	0	0	0
HapCut2	84.5	29259	17980	0.068	0	2.1	39
H-PoP	81.7	32319	17484	0.158	0	1.7	67
Phaser	82.6	24785	16884	0.095	0	1.8	35
WhatsHap	85.2	32656	17980	0.098	0	2.2	49
Real data							
POLYTE	78.2 *(90.5)*	2838	2316	0.090	0	0.2	61
SGA	57.7 *(66.8)*	2842	-	0.069	0	0.0	7
SPAdes	67.0 *(77.5)*	5798	-	0.131	0	0.6	59
SPAdes-dip	66.4 *(76.9)*	5772	-	0.139	0	0.8	82
HapCut2	70.1 *(81.1)*	6541	5306	0.090	0.9	0.2	31
H-PoP	62.4 *(72.2)*	9583	7435	0.119	0.9	0.2	36
Phaser	66.2 *(76.6)*	6394	5245	0.094	0.9	0.2	34
WhatsHap	67.6 *(78.2)*	6257	6094	0.092	0.9	0.2	35

Table 1: Benchmarking results, HC = Haplotype Coverage, ER = Error Rate (mismatches + indels), NR = N-Rate (ambiguous bases), MC = Misassembled Contigs. Top: simulated diploid data for the MHC region. Bottom: real data for chromosome 22 of 1000 Genomes individual NA19240. HC values within parentheses indicate haplotype coverage relative to the amount of bases covered by sequencing reads.

	Simulated data	Real data
POLYTE	93.4	67.5
SGA	87.6	58.8
SPAdes	92.7	47.6
SPAdes-dip	99.8	48.6
HapCut2	67.2	66.5
H-PoP	50.3	57.5
Phaser	69.2	62.8
WhatsHap	58.5	65.3

Table 2: Polymorphism evaluation: fraction (%) of SNP positions represented correctly in the assembly. Evaluated for all methods on simulated MHC data (diploid, 20x coverage per haplotype) and real data (chromosome 22, NA19240).

contigs to the true haplotypes using bwa-mem. Then, for each SNP position we checked whether both haplotypes were correctly represented in the assembly.

Suppose there are k positions where a SNP occurs between the true haplotypes, then there are $2k$ 'relevant' positions, namely k in each haplotype. We output the number of positions represented correctly, divided by the total number of positions considered (Table 2).

8 Runtime and memory usage

We compared CPU time and peak memory usage for all methods on our largest data set, the 1000 Genomes chromosome 22 data, and on one of the simulated data sets. Results are presented in Tables 3 and 4. Table 5 shows the runtime for POLYTE on data sets of increasing ploidy and sequencing depth. Experiments were performed on a 24-core (Intel-Xeon 2.0GHz) Linux machine.

	CPU time	peak memory usage
POLYTE	2.8 Ms	16.6 GB
SGA	0.46 Ms	1.4 GB
SPAdes	0.19 Ms	45 GB
SPAdes-diploid	0.15 Ms	22 GB
HapCut2	3.2 ks	0.5 GB
H-PoP	3.1 ks	3.3 GB
Phaser	3.2 ks	0.3 GB
WhatsHap	3.9 ks	2.4 GB

Table 3: Runtime and memory usage on chromosome 22 of individual NA19240. Note that de novo assembly algorithms (POLYTE, SGA, and SPAdes) are highly parallelizable, leading to feasible runtimes on multi-core machines in practice. Reference-guided algorithms (HapCut2, H-PoP, Phaser, and WhatsHap) require a variant call set as input, so this step (FreeBayes, 3.1 ks) is included in the runtime analysis.

	CPU time	peak memory usage
POLYTE	53 ks	1.7 GB
SGA	20 ks	0.1 GB
SPAdes	6 ks	4.6 GB
SPAdes-diploid	3.5 ks	3.8 GB
HapCut2	528 s	0.1 GB
H-PoP	413 s	8.6 GB
Phaser	655 s	0.4 GB
WhatsHap	488 s	0.5 GB

Table 4: Runtime and memory usage on simulated diploid data for the MHC region with 20x coverage per haplotype (i.e. 40x total coverage). Reference-guided algorithms (HapCut2, H-PoP, Phaser, and WhatsHap) require a variant call set as input, so this step (FreeBayes, 354 s) is included in the runtime analysis.

	5x	10x	20x	30x	40x	50x
$k = 1$	4.7 ks	11 ks	10 ks	17 ks	25 ks	36 ks
$k = 2$	11 ks	27 ks	41 ks	77 ks	110 ks	159 ks
$k = 3$	19 ks	47 ks	80 ks	133 ks	237 ks	355 ks
$k = 4$	30 ks	72 ks	136 ks	253 ks	417 ks	636 ks

Table 5: Runtime for POLYTE on data sets of increasing ploidy k and sequencing depth.

8.1 Parallelization

De novo assembly is much more expensive in terms of runtime than reference-guided assembly. By spreading the assembly tasks over multiple cores and running these in parallel, the wall clock time can be reduced significantly, thus enabling assembly of larger genomes as well. We investigated the effect of parallelization for POLYTE by measuring wall clock time while restricting the maximal CPU usage. Results are presented in Table 6.

Max CPU	Wall time
100%	11h6
200%	7h32
400%	4h36
800%	1h49
1600%	1h11

Table 6: POLYTE runtime (wall clock) on simulated diploid data for the MHC region with 20x coverage per haplotype (i.e. 40x total coverage) while restricting maximal CPU usage.

9 Effect of reference genome quality

When processing large genomes or genomic regions, POLYTE uses a reference genome to bin the sequencing reads before starting the de novo assembly procedure. We studied the effect of the reference genome quality on the final assembly results for POLYTE as well as the reference-guided methods (HapCut2, Phaser, Whatshap, H-PoP). To do so, we performed experiments on the simulated diploid data for the MHC region with 20x coverage per haplotype after introducing mutations at varying rates (2%, 5%, 10%) into the original reference genome hg38. The introduced mutations consist of 90% substitutions, 5% insertions of random size between 1 and 10 bp, and 5% deletions also of random size between 1 and 10 bp. Results are shown in Table 7.

We observe that for each of the mutated reference genomes, POLYTE still achieves much higher haplotype coverage (HC) at much lower error rates (ER) than the other methods. As the reference quality decreases these differences become even more pronounced. Up to 5% mutations in the reference genome hardly affects the assembly quality obtained with POLYTE; only at 10% mutations, that is when operating at a rate of 1 out of 10 bases diverging from the sequence one aims to assemble, we observe tangible effects, concerning haplotype coverage and N50 values, in addition to a few more misassemblies.

10 Effect of ploidy and sequencing depth

We studied the effect of genome ploidy and sequencing depth on the assembly quality and completeness using data sets of varying ploidy and sequencing depth. For de novo assemblers POLYTE, SPAdes, and SGA we present results in Tables 8–10. For the reference-guided polyploid assembler H-PoP full results are shown in Table 11. The reference-guided methods considered (HapCut2, Phaser, Whatshap), as well as SPAdes-dip (diploid mode), are designed specifically for diploid data, so for those we could only assess the effect of sequencing depth. The corresponding results can be found in Table 12.

	HC (%)	N50	NGA50	ER (%)	NR (%)	MCL (%)	MC (%)
Original reference hg38							
POLYTE	92.4	4397	4394	0.035	0	0	0
HapCut2	84.5	29259	17980	0.068	0	7.1	2.1
H-PoP	81.7	32319	17484	0.158	0	12.7	1.7
Phaser	82.6	24785	16884	0.095	0	5.1	1.8
WhatsHap	85.2	32656	17980	0.098	0	10.6	2.2
2% mutations							
POLYTE	92.2	4309	4250	0.035	0	0.2	0.0
HapCut2	80.5	23418	17503	0.210	0	6.2	1.9
H-PoP	80.1	23401	17468	0.225	0	6.2	1.8
Phaser	80.9	29735	17643	0.111	0	4.3	1.2
WhatsHap	83.4	30740	18015	0.132	0	9.4	2.0
5% mutations							
POLYTE	91.5	4106	4106	0.030	0	0	0
HapCut2	78.2	26000	17331	0.259	0	5.4	1.4
H-PoP	77.9	25882	15994	0.289	0	4.9	1.1
Phaser	78.6	39452	21876	0.184	0	5.0	1.3
WhatsHap	81.9	39492	21880	0.184	0	8.3	2.0
10% mutations							
POLYTE	88.0	3269	3264	0.030	0	0.4	0.2
HapCut2	75.1	19929	13777	0.431	0	3.1	1.2
H-PoP	75.4	18585	12704	0.496	0	3.0	0.9
Phaser	75.1	34934	22265	0.386	0	3.8	1.2
WhatsHap	75.3	34200	21617	0.382	0	2.3	0.9

Table 7: Benchmarking results with an increasingly distant reference genome, HC = Haplotype Coverage, ER = Error Rate (mismatches + indels), NR = N-Rate (ambiguous bases), MCL = Misassembled Contig Length, MC = Misassembled Contigs.

	5x	10x	20x	30x	40x	50x
Haplotype coverage (%)						
$k = 1$	59.2	97.4	99.0	99.5	99.6	99.6
$k = 2$	79.6	91.1	92.4	93.5	94.1	94.0
$k = 3$	77.1	85.7	87.8	88.8	88.8	88.7
$k = 4$	74.6	81.2	84.6	84.6	84.1	84.6
N50						
$k = 1$	791	4619	20784	204202	204181	273854
$k = 2$	2053	2496	4397	4321	4422	4319
$k = 3$	1693	1588	2329	2108	1966	1810
$k = 4$	1521	1364	1833	1665	1524	1448
NGA50						
$k = 1$	791	4619	20784	204202	204181	273854
$k = 2$	1375	2316	4394	4366	4601	4423
$k = 3$	935	1223	1929	1814	1624	1542
$k = 4$	701	893	1317	1224	1076	1051
Error rate (%)						
$k = 1$	0.007	0.007	0.001	0.004	0.001	0.005
$k = 2$	0.045	0.039	0.035	0.028	0.024	0.021
$k = 3$	0.050	0.043	0.041	0.034	0.030	0.026
$k = 4$	0.056	0.049	0.053	0.041	0.032	0.032
Misassembled contig length (%)						
$k = 1$	0	0.2	0	0	0	8.5
$k = 2$	0.2	0.1	0	0.7	0.1	0.7
$k = 3$	0	0	0	0.1	0	0
$k = 4$	0	0	0.1	0	0.1	0
Misassembled contigs (%)						
$k = 1$	0	0.05	0	0	0	1.6
$k = 2$	0.15	0.13	0	0.18	0.07	0.10
$k = 3$	0	0	0	0.04	0	0
$k = 4$	0	0	0.05	0	0.01	0

Table 8: POLYTE assembly statistics on simulated data of varying ploidy (k=1,2,3,4) and sequencing depth (5x,10x,20x,30x,40x,50x).

	5x	10x	20x	30x	40x	50x
Haplotype coverage (%)						
$k = 1$	93.5	99.7	99.9	99.9	99.9	99.9
$k = 2$	73.0	82.4	84.1	85.2	85.6	85.5
$k = 3$	62.3	72.9	75.4	75.8	75.9	76.0
$k = 4$	46.6	67.1	68.3	68.3	67.9	67.7
N50						
$k = 1$	7712	82690	2611777	450996	1391746	1488436
$k = 2$	38694	4285	3588	3724	3843	3917
$k = 3$	18535	2886	2668	2787	2736	2749
$k = 4$	165865	2707	2781	2982	3184	3198
NGA50						
$k = 1$	6751	80625	2611777	450996	1391746	1488436
$k = 2$	2295	898	919	940	938	938
$k = 3$	-	-	-	-	-	-
$k = 4$	-	-	-	-	-	-
Error rate (%)						
$k = 1$	0.128	0.005	0	0	0	0
$k = 2$	0.143	0.039	0.032	0.030	0.019	0.017
$k = 3$	0.115	0.041	0.030	0.031	0.032	0.031
$k = 4$	0.170	0.045	0.040	0.039	0.038	0.038
Misassembled contig length (%)						
$k = 1$	10.5	5.3	0	0	0	0
$k = 2$	4.5	0.3	0.8	0	0	0.3
$k = 3$	1.8	0.7	0	0	0	0
$k = 4$	22.7	0	0	0	0	0
Misassembled contigs (%)						
$k = 1$	5.5	2.4	0	0	0	0
$k = 2$	3.1	0.05	0.04	0	0	0.04
$k = 3$	1.4	0.07	0	0	0	0
$k = 4$	7.0	0	0	0	0	0

Table 9: SPAdes assembly statistics on simulated data of varying ploidy (k=1,2,3,4) and sequencing depth (5x,10x,20x,30x,40x,50x).

	5x	10x	20x	30x	40x	50x
Haplotype coverage (%)						
$k = 1$	12.7	78.9	96.9	95.6	94.6	93.8
$k = 2$	54.3	76.4	73.6	71.6	69.7	69.6
$k = 3$	61.9	64.8	60.2	58.5	57.9	56.3
$k = 4$	58.1	54.7	50.7	48.5	48.4	48.2
N50						
$k = 1$	806	1750	5499	4416	4100	3826
$k = 2$	1551	3222	3444	3294	3236	3155
$k = 3$	2748	2896	3019	2757	2801	2745
$k = 4$	3017	2673	2744	2778	2708	2741
NGA50						
$k = 1$	-	1427	5523	4372	3999	3700
$k = 2$	-	-	-	-	-	-
$k = 3$	-	-	-	-	-	-
$k = 4$	-	-	-	-	-	-
Error rate (%)						
$k = 1$	0.031	0.004	0	0	0	0
$k = 2$	0.045	0.042	0.025	0.024	0.023	0.023
$k = 3$	0.059	0.046	0.035	0.035	0.035	0.033
$k = 4$	0.063	0.047	0.042	0.039	0.039	0.040
Misassembled contig length (%)						
$k = 1$	0.2	0.2	0	0	0	0
$k = 2$	0.3	0	0	0	0	0
$k = 3$	0.2	0	0	0	0	0
$k = 4$	0.1	0	0	0	0	0
Misassembled contigs (%)						
$k = 1$	0.3	0.08	0	0	0	0
$k = 2$	0.04	0	0	0	0	0
$k = 3$	0.05	0	0	0	0	0
$k = 4$	0.05	0	0	0	0	0

Table 10: SGA assembly statistics on simulated data of varying ploidy (k=1,2,3,4) and sequencing depth (5x,10x,20x,30x,40x,50x).

	5x	10x	20x	30x	40x	50x
Haplotype coverage (%)						
$k = 2$	77.3	79.4	81.7	81.2	82.0	80.8
$k = 3$	63.2	63.8	64.8	64.9	66.1	65.2
$k = 4$	50.8	52.9	52.2	54.2	54.7	54.7
N50						
$k = 2$	31259	31789	32319	335752	33566	33167
$k = 3$	16622	17906	17734	18923	19216	17975
$k = 4$	13408	14364	15009	16026	20852	20199
NGA50						
$k = 2$	20054	16661	17484	17251	17500	18061
$k = 3$	9730	10797	11123	11665	11638	11814
$k = 4$	9323	9300	10238	10556	12357	12907
Error rate (%)						
$k = 2$	0.157	0.161	0.158	0.150	0.162	0.0.168
$k = 3$	0.194	0.217	0.209	0.212	0.201	0.194
$k = 4$	0.218	0.219	0.229	0.240	0.231	0.230
Misassembled contig length (%)						
$k = 2$	8.6	11.0	12.7	12.3	13.0	11.7
$k = 3$	4.7	4.6	12.9	15.8	8.7	13.0
$k = 4$	7.6	8.1	8.8	11.5	10.6	10.6
Misassembled contigs (%)						
$k = 2$	2.2	2.5	1.7	2.1	1.7	1.9
$k = 3$	1.3	1.4	1.8	1.3	1.2	1.1
$k = 4$	1.1	0.9	0.9	0.9	0.9	1.0

Table 11: H-PoP assembly statistics on simulated data of varying ploidy (k=2,3,4) and sequencing depth (5x,10x,20x,30x,40x,50x). Note that experiments for k=1 are excluded because there is nothing for H-PoP to phase (no heterozygous variants).

	5x	10x	20x	30x	40x	50x
Haplotype coverage (%)						
Phaser	78.5	81.1	82.6	82.3	82.1	82.7
WhatsHap	79.1	84.1	85.2	84.7	85.0	85.0
HapCut2	79.3	84.2	84.5	85.2	85.9	85.5
SPAdes-dip	74.2	81.6	83.6	84.0	84.0	84.0
N50						
Phaser	26404	27160	24785	24785	26407	24785
WhatsHap	29681	30025	32656	29259	30025	30427
HapCut2	27243	26408	29259	28168	28192	30023
SPAdes-dip	41370	3860	3294	3547	3565	3610
NGA50						
Phaser	16358	16722	16884	16356	16722	16719
WhatsHap	17888	17642	17980	18045	17978	17922
HapCut2	17755	16852	17980	18045	18300	17980
SPAdes-dip	2243	875	903	919	921	922
Error rate (%)						
Phaser	0.086	0.086	0.095	0.082	0.086	0.080
WhatsHap	0.110	0.092	0.098	0.098	0.111	0.095
HapCut2	0.081	0.068	0.068	0.065	0.067	0.064
SPAdes-dip	0.177	0.034	0.003	0.002	0.002	0.002
Misassembled contig length (%)						
Phaser	7.1	4.9	5.1	4.9	5.1	4.5
WhatsHap	8.7	9.3	10.6	10.7	10.8	9.8
HapCut2	6.9	7.1	7.2	7.5	7.3	7.2
SPAdes-dip	4.1	0.3	0	0	0	0
Misassembled contigs (%)						
Phaser	1.4	1.6	1.8	1.8	1.7	1.6
WhatsHap	1.8	2.2	2.2	2.6	2.2	2.5
HapCut2	1.7	2.2	2.1	2.2	2.2	2.0
SPAdes-dip	2.6	0.05	0	0	0	0

Table 12: Phaser, WhatsHap, HapCut2, and SPAdes-dip assembly statistics on simulated diploid data (k=2) of varying sequencing depths (5x,10x,20x,30x,40x,50x)

References

[1] J. A. Baaijens et al. De novo assembly of viral quasispecies using overlap graphs. *Genome Res*, 27(5):835–848, 2017.

[2] I. Gregor et al. Snowball: strain aware gene assembly of metagenomes. *Bioinformatics*, 32(17):i649–i657, 2016.

[3] A. Gurevich et al. QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, 29(8):1072–1075, 2013.

[4] Gregory Kucherov and Dekel Tsur. Improved filters for the approximate suffix-prefix overlap problem. In Edleno Moura and Maxime Crochemore, editors, *String Processing and Information Retrieval*, pp. 139–148, Cham, 2014. Springer International Publishing.

[5] A. Töpfer et al. Viral quasispecies assembly via maximal clique enumeration. *PLOS Comput Biol*, 10(3):e1003515, 2014.