Bioinformatics, 35(21), 2019, 4281–4289
doi: 10.1093/bioinformatics/btz255
Advance Access Publication Date: 17 April 2019
Original Paper

Sequence analysis

Overlap graph-based generation of haplotigs for diploids and polyploids

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

[1]Centrum Wiskunde & Informatica, 1098 XG Amsterdam, The Netherlands and [2]Theoretical Biology and Bioinformatics, Utrecht University, 3584 CH Utrecht, The Netherlands

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on July 26, 2018; revised on March 18, 2019; editorial decision on April 2, 2019; accepted on April 11, 2019

Abstract

Motivation: Haplotype-aware genome assembly plays an important role in genetics, medicine and various other disciplines, yet generation of haplotype-resolved *de novo* assemblies remains a major challenge. Beyond distinguishing between errors and true sequential variants, one needs to assign the true variants to the different genome copies. Recent work has pointed out that the enormous quantities of traditional NGS read data have been greatly underexploited in terms of haplotig computation so far, which reflects that methodology for reference independent haplotig computation has not yet reached maturity.

Results: We present POLYploid genome fitTEr (POLYTE) as a new approach to *de novo* generation of haplotigs for diploid and polyploid genomes of known ploidy. Our method follows an iterative scheme where in each iteration reads or contigs are joined, based on their interplay in terms of an underlying haplotype-aware overlap graph. Along the iterations, contigs grow while preserving their haplotype identity. Benchmarking experiments on both real and simulated data demonstrate that POLYTE establishes new standards in terms of error-free reconstruction of haplotype-specific sequence. As a consequence, POLYTE outperforms state-of-the-art approaches in various relevant aspects, where advantages become particularly distinct in polyploid settings.

Availability and implementation: POLYTE is freely available as part of the HaploConduct package at https://github.com/HaploConduct/HaploConduct, implemented in Python and C++.

Contact: as@cwi.nl

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

In most eukaryotic organisms genomes come in copies, where each copy stems from one of the ancestors. The number of copies determines the *ploidy* of the organism: while *diploid* relates to two copies, *polyploid* refers to more than two copies (depending on the context, polyploid includes diploid, but here we refer to polyploid as more than two copies). The copy-specific sequences are referred to as *haplotypes*, which generally differ in terms of the genetic variants affecting them. Distinguishing the two haplotypes in diploid organisms (such as in most vertebrates) or more than two in polyploid organisms (such as many plants and some fungi) plays an important role in various disciplines. Prominent examples are genetics, where assigning variants to ancestors is key (Tewhey *et al.*, 2011), and medicine, because very often haplotype-specific combinations of variants establish clinically relevant effects, e.g. when disease risks have been inherited (Glusman *et al.*, 2014). In general, determining *haplotypic sequence*, i.e. in other words keeping track of ancestry based dependencies is instrumental in many biomedical settings.

Assembling the two (diploid) or more (polyploid) haplotypes from sequencing reads is known as *haplotype-aware genome assembly*, and the resulting assembled pieces of sequence are *haplotigs*, as a shorthand for haplotype-aware contigs. The advent of next-generation sequencing (NGS) has brought about a plethora of NGS read compatible assembly programs. The vast majority of these programs, however, do not yield haplotigs, but consensus genome sequence, as a summary across all haplotypes involved. Even then, sequencing errors, read length and hardware limitations already pose fundamental challenges during the assembly process.

Generating haplotigs from NGS reads—which is the challenge that we tackle here—comes with additional obstacles. Beyond distinguishing between errors and true sequential variants, one needs to assign the true sequential variants to the different copies. This requires keeping track of information that allows to link the true sequential variants stemming from identical copies. However, NGS reads in general are rather short: techniques are needed that can link haplotype-specific variants across read boundaries. Despite the many recent advances, this is not (yet) a standard procedure in genome assembly: haplotype-aware assembly can still be considered in its early stages of development which explains that further advances are desirable.

Motivation. The majority of sequencing machines installed worldwide perform traditional NGS, such as Illumina sequencing. A plethora of population-scale sequencing studies (e.g. Besenbacher *et al.*, 2015; Sudmant *et al.*, 2015; The Genome of the Netherlands Consortium, 2014; The UK10K Consortium, 2015) have filled up databases with traditional, short NGS reads. In terms of quantities, traditional short NGS reads exceed the amount of reads stemming from more recent third-generation sequencing (TGS) protocols by at least one order of magnitude. The increase in read length due to TGS has considerably spurred the development of methods for haplotype-aware assembly (see Related work). While the increase in length is beneficial, the increase in sequencing error rates is also a major obstacle when distinguishing between haplotypes, usually leaving applicants with ambiguities that are hard to resolve.

Recent work has pointed out that targeted examination of NGS (Illumina type) reads can have significant positive effects in haplotype-aware assembly (Berger *et al.*, 2014; Patterson *et al.*, 2015). Seemingly, the enormous quantities of traditional NGS read data have been underexploited in terms of haplotig computation so far. This establishes our major motivation.

To better understand where serious progress can be made, one needs to realize that existing methods for haplotype computation from traditional NGS (Illumina) reads fall into two classes: the first (and arguably more popular) choice of approaches are referred to as *haplotype assembly* programs. These approaches make use of a reference genome to call variants from aligned reads, which are subsequently phased into separate haplotypes. The advantage of haplotype assembly programs is their stability and their resource-friendly usage. Examples for *diploid haplotype assembly* are WhatsHap (Patterson *et al.*, 2015), Phaser (Castel *et al.*, 2016), HapCut2 (Edge *et al.*, 2017), ProbHap (Kuleshov, 2014) and HapCol (Pirola *et al.*, 2016). Examples for *polyploid haplotype assembly* are HapCompass (Aguiar and Istrail, 2012), HapTree (Berger *et al.*, 2014), SdhaP (Das and Vikalo, 2015) and H-PoP (Xie *et al.*, 2016). The disadvantage of haplotype assembly programs is that they depend on high-quality reference sequence as a backbone. In addition, they depend on external variant call sets. These two factors can introduce non-negligible biases.

The second class of methods is *de novo haplotype-aware assembly* approaches that can deal with traditional NGS (in particular Illumina) reads. The advantage of such approaches is that they are independent of reference genomes and external call sets, which eliminates the externally induced biases. There are only little such approaches available however; to the best of our knowledge, only ALLPATHS-LG (Ribeiro *et al.*, 2012), Platanus (Kajitani *et al.*, 2014) and dipSPAdes (Safonova *et al.*, 2015) explicitly aim at computation of haplotigs from (diploid) NGS data. However, ALLPATHS-LG and Platanus require particularly tailored libraries, which renders their general application difficult, and the dipSPAdes software is no longer maintained. In results of ours, we further noted that SPAdes (Bankevich *et al.*, 2012) can be run in diploid mode (which is not to be confused with the no longer maintained dipSPAdes), and is able to compute haplotigs (surprisingly not only in diploid, but also in conventional mode), thereby likely establishing the only tool among the (myriad of) approaches for consensus oriented genome assembly [see Bradnam *et al.* (2013) and Salzberg *et al.* (2012) for references] that one can use for computation of haplotigs from short NGS reads.

In summary, there are no approaches that (i) specialize in the generation of (high-quality) haplotigs, but (ii) do not depend on high-quality reference sequence as a backbone, (iii) do not depend on external variant call sets and (iv) do not require particularly tailored sequencing libraries.

Contribution. The contribution of this paper is to close this gap in the landscape of approaches. We present POLYploid genome fitTEr (POLYTE), as an approach to do this for genomes of known ploidy. Our results indicate that POLYTE outperforms state-of-the-art approaches of the two classes—haplotype assembly and *de novo* assembly approaches—with significant advantages in a variety of relevant aspects. As an example of an application scenario, POLYTE outperforms the other approaches in reconstructing individual haplotypes of the human Major Histocompatibility Complex (MHC). This region of 6 Mb on chromosome 6 is essential to the acquired immune system and shows very high genetic variability; haplotype-aware reconstruction of the MHC region therefore usually is particularly challenging during the assembly process. Note finally that the majority of approaches focuses on diploid genomes. Therefore, the lack of approaches that can compute haplotigs for organisms of ploidy larger than two is even more striking. For ploidy larger than two, POLYTE achieves performance rates that are nearly on a par with those achieved for diploid organisms. To the best of our understanding, because of the lack of competitors, one might perceive POLYTE's achievements for polyploid organisms as a novelty in its own right.

Related work. In terms of assembly paradigms, POLYTE is an *overlap graph-based* approach. It adopts ideas from earlier work that either focused on variant discovery (Marschall *et al.*, 2012), viral quasispecies assembly (Baaijens *et al.*, 2017; Töpfer *et al.*, 2014) or metagenome gene assembly (Gregor *et al.*, 2016) and unites the virtues of Marschall *et al.* (2012)—the ability to handle low coverage—on the one hand, and Baaijens *et al.* (2017) and Töpfer *et al.* (2014)—dealing with real overlap graphs and contig computation—on the other hand. That is, POLYTE brings forth an iterative overlap graph-based scheme for contig generation that reliably works in *low coverage settings*, requiring coverage of only as low as $5\times$ per haplotype.

Note finally that our approach also draws motivation from the recent technology shifts, such as the advent of TGS and explicitly haplotype-aware sequencing protocols like StrandSeq (Porubsky *et al.*, 2017), which have put the computation of haplotigs into the focus of current attention. Chin *et al.* (2016), Jain *et al.* (2018) and Weisenfeld *et al.* (2017) describe approaches that aim to exploit the respective advances in sequencing technology and protocol design. Although there are similarities between these approaches and POLYTE, we focus on NGS data and hence our method fully exploits paired-end read information. We consider the adaptation of POLYTE to TGS data most interesting future work: the framework of POLYTE is generic in terms of choosing reads, such that this is a matter of adapting parameters, more than anything else. We recall, however, that our motivation was to bring forward a method that exploits (the abundantly available) traditional NGS reads in the first place. This, e.g. enables to reconstruct MHC region haplotypes in various population-scale studies (e.g. Besenbacher *et al.*, 2015; Sudmant *et al.*, 2015; The Genome of the Netherlands Consortium, 2014; The UK10K Consortium, 2015), which has been a major challenge so far.

2 Materials and methods

We present POLYTE, an algorithm to assemble individual haplotypes of diploid and polyploid genomes from short-read sequencing data; see Figure 1 for the complete workflow. POLYTE follows the overlap-layout-consensus (OLC) paradigm, where consensus refers to removing errors within haplotypes (instead of the common interpretation of reaching consensus across different haplotypes). Our method starts by constructing a *read-overlap graph* which is used for error correction of the input sequences. Subsequently, we make use of an iterative OLC scheme, where in each iteration a *contig-overlap graph* is constructed. This graph is further reduced by applying *transitive edge removal* and *read-based branch reduction*. Then, contigs are clustered and merged according to their interplay within the overlap graph, resulting in a collection of extended contigs ('contig extension' in Fig. 1). These extended contigs establish the nodes of the contig-overlap graph of the next iteration, which is achieved by an updating procedure. When contigs cannot be merged any further, POLYTE outputs the final set of contigs. When dealing with diploid organisms, an additional assembly stage can be activated which consists of two additional steps ('diploid branch reduction' and 'contig extension' in Fig. 1), creating an optional output that is refined for diploid organisms.

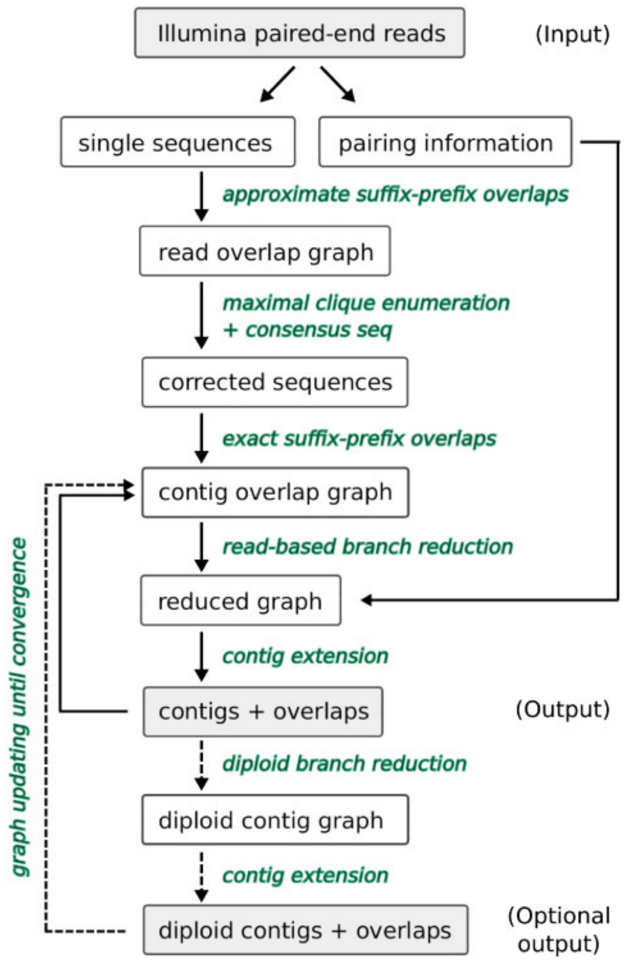

Fig. 1. Algorithm overview

Given that we are dealing with data of relatively low sequencing depth, we need to exploit the information present in the sequencing reads as much as possible. The initial error correction procedure is particularly crucial, as sequencing errors can heavily disturb the process of distinguishing between different haplotypes. For this error correction step, approximate suffix–prefix overlaps are computed to establish an initial read-overlap graph. Inspired by Baaijens *et al.* (2017) and Töpfer *et al.* (2014), maximal cliques are enumerated in the non-oriented graph and errors are corrected by inspecting the read overlaps within the cliques. By design of the overlap graph—edges indicate that two reads stem from identical haplotypes—every clique only contains reads from identical haplotypes, which allows to eliminate errors based on majority votes. Note that this procedure is particularly tailored to low coverage settings with known ploidy: admissible clique sizes and minimal sequence overlap lengths can heavily vary in comparison to earlier approaches. However, with edge criteria that are much less restrictive than in other approaches, we obtain a larger number of spurious edges. We have developed a procedure for *read-based branch reduction* to reduce the number of spurious edges in the overlap graph, which is of great importance for accurate reconstruction of haplotigs.

In the following sections we will discuss each of the steps involved in POLYTE, following the workflow depicted in Figure 1.

2.1 Read-overlap graph construction

The steps outlined in this section refer to the initial step 'approximate suffix-prefix overlaps' that leads to the establishment of the 'read overlap graph' in Figure 1.

Read-overlap graph: definition. The read-overlap graph follows the idea that nodes are reads and edges indicate that a pair of reads stem from identical haplotypes. Given the input consisting of paired-end sequencing reads (Illumina), let $\mathcal{R}$ be the collection of single end sequences from all paired-end reads. The *read-overlap graph* $G = (V, E)$ is a directed graph where V corresponds to the collection of input sequences $\mathcal{R}$. That is, for every paired-end read we have two vertices $v, v' \in V$, one for each single end sequence $R \in \mathcal{R}$. Directed edges $v_i \to v_j \in E$ connect sequences R_i, R_j whenever the suffix of R_i overlaps the prefix of R_j for at least 50% of the average sequence length of all reads. Furthermore, for each edge $v_i \to v_j$, we require $\mathrm{QS}(R_i, R_j) \geq \delta$, where $\mathrm{QS} : \mathcal{R} \times \mathcal{R} \to \mathbb{R}$ is a quality score and δ is an appropriate threshold. This threshold is determined based on empirical statistics so as to maximize the chances that the edge (v_i, v_j) indeed indicates that the corresponding sequences R_i and R_j stem from identical haplotypes; increasing the δ-threshold would lead to a higher accuracy but possible loss of low abundance haplotypes. In this, we largely follow ideas presented in earlier work (Baaijens *et al.*, 2017; Marschall *et al.*, 2012; Töpfer *et al.*, 2014).

The difference with respect to these prior approaches is that only single ends are considered, whereas in the earlier approaches nodes represent the entire paired-end reads. Also note that here overlap graphs are twice as large in comparison to the earlier approaches, because each paired-end read is represented by two nodes, instead of only one. While this difference imposes substantial methodical and technical challenges, it is key to dealing with low coverage because it decisively increases the recall in terms of recovering reads that stem from identical haplotypes. However, it also implies follow-up complications, because the information that read ends come in pairs is temporarily lost. In POLYTE, paired-end information is stored and used in later steps; see Section 2.4 below.

Construction. Computation of the edges for the read-overlap graph requires enumeration of all pairwise approximate suffix–prefix overlaps (of sufficient length) between the single read ends $R \in \mathcal{R}$ and evaluation of 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. We further orient the edges (which is necessary because reads can stem from either the forward or the reverse stand) and systematically remove double transitive edges, which ensures that one can enumerate maximal cliques in an efficient manner (see Section 2.2). Each of these graph construction steps is described in detail in Supplementary Section 1.

The computation of approximate suffix–prefix overlaps for vertebrate genome sized input read sets is a serious issue, currently hardly conceivable without external auxiliary means (see also Simpson and Durbin, 2012). Here, we suggest a method that aims to suppress externally introduced biases to a maximum degree. We make use of a reference genome for binning reads in an initial step and, after binning, we discard the reference genome and any related information entirely such that POLYTE operates in full *de novo* mode. This binning step does not require a high-quality reference genome, as long as reads get mapped; any unaligned reads are discarded (see also Supplementary Section 9).

2.2 Correction of sequencing errors

After the establishment of the read-overlap graph, we cluster its nodes by enumerating the *maximal cliques* contained in the non-oriented graph. The idea is to collect groups of reads belonging to the same haplotype and produce error-free sequences for subsequent assembly steps (‘corrected sequences’, Fig. 1). By definition of a maximal clique—a maximal group of nodes all of which are connected by edges—maximal cliques represent maximum-sized groups of reads all of which belong to the same haplotype. Once all maximal cliques are determined, it is therefore reasonable to merge the reads within a maximal clique into a single contig. Note that this contig is longer than the individual reads participating in the contig and that sequencing errors can be eliminated by raising majority votes among the reads participating in the maximal clique. While this reflects an approved procedure in its generic form (Baaijens *et al.*, 2017; Gregor *et al.*, 2016; Töpfer *et al.*, 2014), accounting for the particular setting we are facing here—namely low coverage in combination with sequence-based edge definition—requires particular care.

The minimum clique size depends on the coverage per haplotype; in all settings considered we are dealing with known ploidy, while the overall coverage of reads can be determined by usual considerations, which yields per-haplotype-coverage estimates. To determine the optimal minimum size of a clique for a given per-haplotype coverage, we compute the probability $p_{c,k}$ that, due to unfortunate fragmentation of sequencing reads, at a per-haplotype coverage of c there is no clique of size k that extends a given sequencing read R to the right when requiring at least 50% read overlap. In other words, we compute the probability that there are *at most* $k-1$ reads extending R to the right; the exact same analysis applies to extensions to the left.

For determining $p_{c,k}$, we assume that sequencing reads are fragmented randomly, which implies that reads are generated independently of one another. Let R be a read and S be a set containing reads from the haplotype of R at exactly 1× coverage, further assuming that all reads $R' \in S$ have the same length as R (which reflects that all single read ends have the same length). It is straightforward to see that the probability that there is $R' \in S$ that overlaps R at at least 50% of its length (into one direction, left or right) as 0.5. When dealing with a per-haplotype coverage of c, we assume the existence of c sets of reads $S_i, i = 1, .., c$ all of which contain reads that cover the haplotype c at 1×. For computing $p_{c,k}$, we consider that for only $k-2$ of the c sets $S_i, i = 1, \ldots, c$ we have that there is $R' \in S_i$ that overlaps R at at least 50% of its length (resulting in a clique of size at most $k-1$), which evaluates as

$$p_{c,k} = \sum_{i=0}^{k-2} \binom{c-1}{i} 0.5^i 0.5^{c-1-i} = \sum_{i=0}^{k-2} \binom{c-1}{i} 0.5^{c-1}. \quad (1)$$

We aim to have $p_{c,k}$ low to be able to deal with sufficiently many cliques, hence for every choice of c we compute k such that $p_{c,k} < 0.001$. In this regard, we obtain that for up to 10× per haplotype an appropriate choice for the minimum clique size is 2, for coverages between 10 and 15× a minimum clique size of 3 is required, while for $c \geq 15\times$ an optimal choice for the minimum clique size is 4. Note that in practice cliques do not grow larger than size 4 because of double transitive edge removal (see Supplementary Material).

2.3 Contig-overlap graph construction

Given the corrected sequences obtained by merging maximal cliques, we build a new graph: the contig-overlap graph (see Fig. 1).

Contig-overlap graph: definition. The contig-overlap graph $G' = (V', E')$ is very similar to the read-overlap graph, except that we construct it from a set of contigs assumed to be free of sequencing errors. Therefore, every node $v \in V'$ corresponds to a contig and we add an edge between a pair of nodes whenever they have an exact (i.e. error-free) overlap of sufficient length.

Construction. The contig-overlap graph can be constructed very efficiently by making use of the FM-index-based algorithm from Section 2.1 while allowing only exact overlaps. This gives us the complete edge set E' without any further computations, since we do not need to compute the overlap quality score for exact overlaps. Note that the minimal overlap length in the contig-overlap graph does not need to be as high as before error correction and it is independent of the read length: all experiments were performed using a minimal contig overlap of 50 bp.

Remark. Allowing approximate overlaps in this stage of the algorithm, e.g. by allowing some substitutions, would slow down the contig-overlap graph construction considerably. Although the additional edges could lead to improved recovery of true haplotypes, it would also bring the risk of collapsing highly similar sequences and thus missing haplotypes.

2.4 Branch reduction in the contig-overlap graph

Before using the contig-overlap graph to extend our contigs, we trim the graph by removing redundant vertices and edges and resolving branches based on read evidence where possible, now also exploiting the paired-end information. After completing this step, we have a ‘reduced graph’ (see Fig. 1) that is ready for contig extension.

Transitive edge removal. An edge $u \to w \in E'$ is called *transitive* if there exists a vertex $v \in V'$ and edges $u \to v, v \to w \in E'$. Now that sequences (contigs) are assumed to be error-free, transitive edges have become fully redundant, hence we remove all transitive edges from the graph before further processing.

Branching edges and nodes. The *indegree* (resp. *outdegree*) of a node $v \in V'$ is defined as the total number of incoming (resp.

outgoing) edges in G'. If v has indegree >1, we say v has an *in-branch*; analogously, if v has outdegree >1, we say that v has an *out-branch*. We refer to the corresponding edges as *branching edges* and to v as a *branching node*. Since we did not use any read pairing information during construction of our overlap graphs, we observe many branches in the contig-overlap graph. We now use the information how ends are paired to remove any branching edges in the contig-overlap graph that do not correspond to a true haplotype.

Merging simple paths. Following the above definition, any edges that are not branching edges constitute *simple paths* through the contig-overlap graph. For such paths, there is only one possible way to combine the corresponding contigs; hence, before processing the graph any further, we merge every simple path into a single contig. Since edges in the graph represent exact overlaps, this is a straightforward procedure.

Branching components. After merging simple paths, all remaining edges are branching edges. We define a *branching component* as subgraph H of the contig-overlap graph, such that (i) H is an induced subgraph, (ii) H is connected as an undirected graph and (iii) within H, any vertex has only incoming or outgoing edges in H, but not both. A branching component is defined to be maximal with respect to these three properties; see Figure 2. Intuitively, a branching component reflects all possible haplotypes within a small region of the genome.

Note that different components may intersect across their vertex sets, but cannot have any edges in common. In other words, the maximal branching components partition the set of all branching edges, as illustrated in Figure 2. This partition can be found in time linear in the number of branching edges by alternatingly traversing in-branch edges and out-branch edges until every edge has been seen exactly once; see Supplementary Section 2 for further details. After enumerating all maximal branching components, we evaluate read evidence per component.

Read evidence. The main idea of read-based branch reduction is to remove all branching edges for which there is insufficient *read evidence* in the input data. For this purpose, we keep track of all original sequencing reads ('subreads') that were used to build a contig; each of these subreads may provide evidence for a branching edge. Within a branching component, we first list all *variant positions*, i.e. the positions at which the sequences corresponding to the different neighbors differ from each other. Intuitively, these are the positions where we may find sequencing reads supporting a given branching edge. A paired-end sequencing read $R = (R_1, R_2)$ is marked as *evidence* for the branching edge $u \rightarrow v$ if it satisfies the following conditions:

i. R spans the branching edge, meaning that at least one of the sequences R_1, R_2 is a subread of u *and* at least one of the sequences R_1, R_2 is a subread of v;
ii. The sequence spanning the edge is identical to the contig sequence of the corresponding node for all variant positions it covers;
iii. R is unique for this edge: it does not satisfy conditions (i) and (ii) for any other edge involved in this branching component.

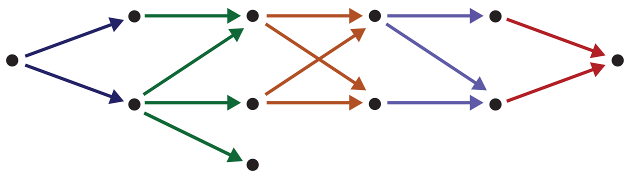
Fig. 2. Illustration of branching components in a contig-overlap graph. Edges of the same color belong to the same branching component

Figure 3 shows two examples of contigs creating branches in the overlap graph, along with the sequencing reads ('subreads') that were used to build these contigs; the subreads providing read evidence are highlighted in yellow. Observe that in panel A, in order to satisfy condition (iii) a subread has to cover at least one variant position on either contig. In panel B, we illustrate that also a single read end can provide evidence: the rightmost subread covers a variant position and satisfies all conditions listed above.

Note that condition (ii) ensures that erroneous contigs do not find evidence in correct reads: if a sequencing error accidentally ends up in a contig, it will cause a branch in the overlap graph which can only be supported by reads containing exactly this sequencing error. Whenever such a branch occurs, there will be insufficient evidence and hence the erroneous contigs will never be merged. Eventually, these contigs can be filtered out based on their short length. In the Supplementary Material, we discuss how an appropriate evidence threshold is determined (using similar considerations as for determining the optimal clique size, Section 2.2). Increasing the evidence threshold would lead to a higher accuracy but also potential loss of low abundance haplotypes.

Branching edge removal. For every branching component, we count the read evidence per branching edge and remove any edges with evidence count below the evidence threshold.

2.5 Contig extension and graph updating

After applying the read-based branch reduction techniques described above, all branches have been either resolved or removed from the contig-overlap graph. Contig extension has become an easy task: any contigs which are connected by an edge in the graph must belong to the same haplotype, and, therefore, we merge each such pair of contigs into a new, longer contig. Then, we update the overlap graph: the extended contigs become the new nodes and the edges are updated accordingly. The resulting updated graph is used for further assembly in an iterative manner, as described in Section 2.6.

2.6 Iterative procedure and diploid mode

Our workflow consists of iteratively performing the steps described in Sections 2.3–2.5, as illustrated in Figure 1. The number of edges in the contig-overlap graph decreases with every iteration, since

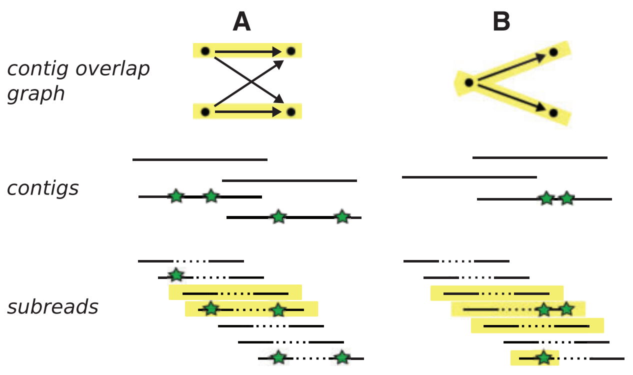

Fig. 3. Two examples of contigs creating branches in the overlap graph. The corresponding reads are aligned below. Edges corresponding to true haplotypes and the reads providing evidence are highlighted. (A) Only 2 out of 4 edges are supported by read evidence, the other edges will be removed. (B) Both edges are supported by read evidence

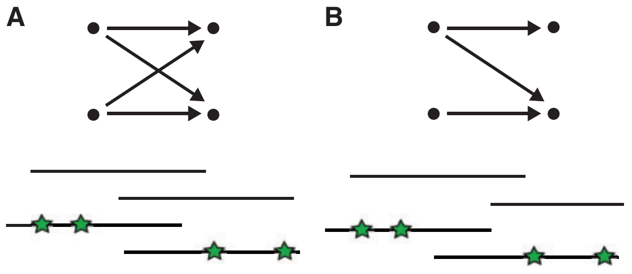

Fig. 4. Typical branching components in diploid assemblies: four contigs, two from each haplotype, having identical sequence in their overlap. Depending on the contig lengths, all contigs overlap (**A**) or only a subset of the contigs overlap (**B**)

contigs connected by an edge in the graph are merged (Section 2.5). The algorithm terminates when the edge set E' of the updated contig-overlap graph becomes empty, either upon construction or after branch reduction. Thus, our algorithm is guaranteed to converge, and once it does we remove any remaining inclusions from the final contig set. Also any contigs shorter than the fragment size of the original reads are removed from the output.

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 (see Fig. 1). 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; see Section 2.4).

In overlap graphs from diploid samples, we typically see two types of branching components; Figure 4 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 (Supplementary Section 3).

This procedure is more risky than default branch reduction (Section 2.4), since it does not require such stringent read evidence. Therefore, we always run the main POLYTE algorithm until convergence before turning to diploid mode (Fig. 1). This ensures that all evidence in the original reads has been exploited first.

3 Results

In this section we show results for POLYTE on both simulated and real Illumina datasets and evaluate the assembly quality in terms of Haplotype coverage (HC), N50, NGA50, Error rate (ER) and the number of misassembled contigs (MC) relative to the total number of contigs. We also compare our method against alternative haplotype reconstruction tools: SPAdes (Bankevich *et al.*, 2012), Phaser (Castel *et al.*, 2016), HapCut2 (Edge *et al.*, 2017), WhatsHap (Patterson *et al.*, 2015), SGA (Simpson and Durbin, 2012) and H-PoP (Xie *et al.*, 2016). Other polyploid assemblers (Aguiar and Istrail, 2012; Berger *et al.*, 2014; Das and Vikalo, 2015) were unable to process our benchmarking data due to issues with the available software. All methods were run with default settings and assembly statistics were obtained with QUAST (Gurevich *et al.*, 2013).

3.1 Datasets

Simulated data. We generated a collection of simulated datasets of varying ploidy and sequencing depth to evaluate the effect of these characteristics. We selected four human MHC haplotypes from the Vega Genome Browser (http://vega.archive.ensembl.org/info/data/MHC_Homo_sapiens.html): COX, DBB, MANN and SSTO. Subsequently, we used SimSeq (https://github.com/jstjohn/SimSeq) to simulate Illumina MiSeq reads of length 2×250 bp for each of those haplotypes at a coverage of 5, 10, 20, 30, 40 and 50×, respectively, and combined the resulting read sets to form datasets of ploidy 1 (only COX haplotype, a sanity check), ploidy 2 (COX and DBB), ploidy 3 (COX, DBB, and MANN) and ploidy 4 (all).

Real data. For evaluation on real sequencing data, we considered a dataset from phase 3 of the 1000 Genomes project (1000 Genomes Project Consortium *et al.*, 2012; Sudmant *et al.*, 2015) for individuals NA19240. This dataset was obtained from a 2×250 bp PCR free Illumina protocol, sequenced to a coverage of 28–68×. Full haplotypes have been reconstructed for this individual as part of a recent study (Chaisson *et al.*, 2019) using various specialized sequencing techniques and reconstruction algorithms; we use the resulting haplotypes as a ground truth for a whole-chromosome benchmarking experiment on chromosome 22.

Alignments and variant call sets for reference-guided methods. Reference-guided methods Phaser, HapCut2, WhatsHap and H-PoP require as input a reference genome, read alignments to the reference genome and a pre-computed set of genomic variants. For the simulated data we performed read alignment to the GRCh38 reference genome using BWA MEM (Li and Durbin, 2009). The real data were already provided as alignments to the GRCh37 reference genome, also obtained with BWA MEM. We extracted the sequencing reads corresponding to chromosome 22 from the provided BAM files. Finally, we performed variant calling on all datasets with FreeBayes (https://github.com/ekg/freebayes).

3.2 Assembly performance criteria

We evaluate assembly performance in terms of several statistics commonly used for *de novo* assembly evaluation, as reported by QUAST.

HC. The completeness of the assembly is measured by the fraction of nucleotides in the target haplotypes (ground truth) covered by haplotigs, referred to as haplotype coverage.

N50 and NGA50. Assembly contiguity is measured using the N50 value, which is defined as the length for which the collection of all contigs of that length or longer covers at least half the assembly. The NGA50 measure is computed in a similar fashion, but only aligned blocks are considered (obtained by breaking contigs at misassembly events and removing all unaligned bases). This measure reports the length for which the total size of all aligned blocks of this length or longer equals at least 50% of the total length of the true haplotypes.

ER and N-rate (NR). We evaluate ER as the sum of mismatch rate and indel rate when comparing to the ground truth haplotype sequences. In addition, we report the relative number of ambiguous bases (‘N’s), referred to as NR.

MC. A contig or haplotig is called misassembled if it contains at least one misassembly, meaning that left and right flanking sequences align to the true haplotypes with a gap or overlap of more than 1 kbp, or align to different strands, or even align to different haplotypes. We report the proportion of misassembled contigs.

3.3 Benchmarking results

We performed benchmarking experiments on one of the simulated MHC datasets described above (ploidy 2, 20× coverage per haplotype)

Table 1. Benchmarking results

	HC (%)	N50	NGA50	ER (%)	NR (%)	MC (%)
Simulated data						
POLYTE	92.4	4397	4394	0.035	0	0
SGA	73.4	3444	—	0.025	0	0
SPAdes	84.1	3588	919	0.032	0	0.0
SPAdes-dip	83.6	3294	903	0.003	0	0
HapCut2	84.5	29 259	17 980	0.068	0	2.1
H-PoP	81.7	32 319	17 484	0.158	0	1.7
Phaser	82.6	24 785	16 884	0.095	0	1.8
WhatsHap	85.2	32 656	17 980	0.098	0	2.2
Real data						
POLYTE	78.2 *(90.5)*	2838	2316	0.090	0	0.2
SGA	*57.7 (66.8)*	2842	—	0.069	0	0.0
SPAdes	67.0 *(77.5)*	5798	—	0.131	0	0.6
SPAdes-dip	66.4 *(76.9)*	5772	—	0.139	0	0.8
HapCut2	70.1 *(81.1)*	6541	5306	0.090	0.9	0.2
H-PoP	62.4 *(72.2)*	9583	7435	0.119	0.9	0.2
Phaser	66.2 *(76.6)*	6394	5245	0.094	0.9	0.2
WhatsHap	67.6 *(78.2)*	6257	6094	0.092	0.9	0.2

Note: Top: simulated diploid data for the MHC region. Bottom: real data for chromosome 22 of 1000 Genomes individual NA19240. HC values within parentheses indicate HC relative to the amount of bases covered by sequencing reads. HC, haplotype coverage; ER, error rate (mismatches + indels); NR, N-rate (ambiguous bases); MC, misassembled contigs.

to compare a variety of haplotype reconstruction tools. In addition, we ran all methods on the chromosome 22 data of the 1000 Genomes individual NA19240. The assembly statistics on both datasets are shown in Table 1. Since both datasets are diploid, we present results for SPAdes in regular mode and in diploid mode, referred to as SPAdes-dip.

In both experiments, we observe that across all methods POLYTE has the largest HC (92.4 and 78.2% for MHC and chr22, respectively). In other words, it reconstructs the largest fraction of the true haplotype sequences. In comparison, the other methods are all more or less on a par [81.7–85.2% (MHC) and 57.7–70.1% (Chr22), respectively). On the real data the HC achieved by all methods is rather low; this can be explained by only 86.4% of the target haplotypes being covered by sequencing reads. After normalizing the HC values by 86.4, POLYTE achieves a HC of 90.5%.

In terms of assembly contiguity, indicated by high N50 and NGA50 values, reference-guided methods (HapCut2, Phaser, WhatsHap, H-PoP) perform better than *de novo* assemblers (POLYTE, SGA, SPAdes). This reflects a common advantage of reference-guided approaches, which can make use of the external information to bridge regions only poorly covered with informative reads, if appropriate. The increase in length, however, is offset by a substantial decrease in terms of haplotig quality: reference-guided approaches exhibit both substantially more misassemblies (which in particular can lead to severe issues in downsteam interpretations) and increased ERs, here larger by one to two orders of magnitude. Note that several NGA50 values are undefined ('-'), because the aligned blocks are unable to cover at least 50% of the total reference length.

Another important difference between reference-guided methods and *de novo* approaches is reflected in the NRs on the real data: the reference genome contains several stretches of ambiguous nucleotides ('N's), which the reference-guided methods cannot correct. *De novo* approaches, on the other hand, can potentially uncover the true sequence behind these ambiguous regions and show an NR of 0% (versus 0.9% for the reference-guided methods).

Between *de novo* approaches, we compare POLYTE with SGA and SPAdes and observe that POLYTE reconstructs a substantially larger fraction of the true haplotypes. Although SPAdes achieves better N50 values, this comes at the expense of a decrease in terms of ER and misassemblies, also reflected in a low NGA50 value on the simulated data and the NGA50 being undefined on the real data (see explanation above). On the simulated dataset, POLYTE and SPAdes achieve comparable ERs of 0.035 and 0.031%, respectively. On the real data we notice an advantage for POLYTE, with an ER of only 0.090% compared to 0.131% for SPAdes. In addition, POLYTE is less vulnerable to misassemblies than SPAdes on real data, with 0.2 versus 0.6% MC. SGA is able to reconstruct highly accurate contigs with slightly lower ERs than POLYTE [0.025 versus 0.035% (MHC) and 0.069 versus 0.090% (Chr22), respectively], but covers a significantly lower fraction of the ground truth haplotypes [73.4 versus 92.4% (MHC) and 57.7 versus 78.2% (Chr22), respectively].

In an overall account, we believe that, arguably, the major advantage of POLYTE is established by the increase of 10–15% over the other approaches in terms of haplotype-specific coverage, in combination with the ERs, which are clearly lower than those of the other tools.

In terms of runtime and memory usage, *de novo* approaches are in general more expensive than reference-guided methods. We also observe this when comparing CPU time and peak memory usage (Supplementary Tables S3–S5). Reference-guided methods have CPU times that are orders of magnitude less compared to *de novo* methods [where POLYTE requires 9–15 times more (resp. 3–6 times more) runtime and 3 times less (resp. 12–17 times more) memory than SPAdes and SGA, respectively]. It is important to notice, however, that these *de novo* assemblers are highly parallelizable; we demonstrate the effect of increasing the number of available CPU's on the effective runtime in Supplementary Table S6. This leads to feasible runtimes on multi-core computing facilities in practice.

3.4 Effect of ploidy and sequencing depth

To study the effect of genome ploidy and sequencing depth on the assembly quality and completeness, we ran POLYTE, SPAdes, SGA and H-PoP on all simulated datasets described in Section 3.1 (other tools were unsuitable for polyploid genomes). Figure 5 shows the results for the 5, 10 and 20× datasets in terms of HC, N50, NGA50, ER, and MC. For additional result tables we refer the reader to the Supplementary Tables S8–S11.

We observe that POLYTE excels regarding HC, with advantages becoming more distinct as the ploidy increases. SPAdes and H-PoP achieve more contiguous assemblies (higher N50 values) but, as we already observed on diploid data, this comes at the cost of significantly higher ERs and misassemblies. SGA performs very similar to POLYTE when considering N50, ER and MC, but obtains much lower HC values. The NGA50 values highlight the improved assembly quality of POLYTE over SGA and SPAdes: while POLYTE achieves NGA50 values comparable to the N50, SGA and SPAdes are unable to cover at least 50% of the ground truth with alignments (hence NGA50 is undefined). Overall, we conclude that in polyploid settings the same advantages of POLYTE apply as in diploid settings—increased haplotype-specific coverage in combination with low ERs—and become even more pronounced.

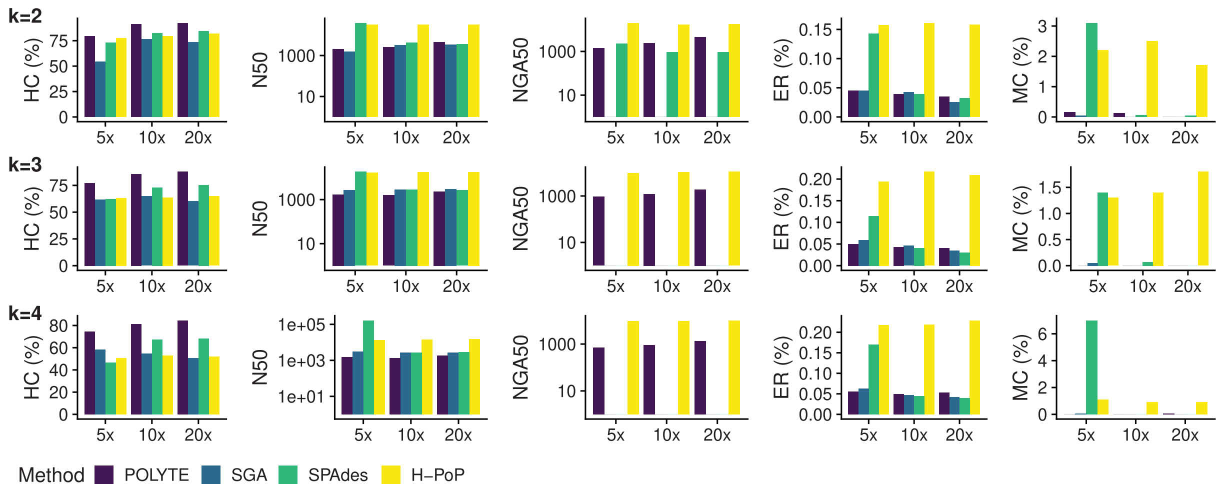

Fig. 5. Assembly results per method for simulated data of increasing ploidy (k=2, 3, 4) and per-haplotype coverage (5, 10, 20×). N50 and NGA50 values are plotted on a log-scale for increased readability. For SGA (k=2, 3, 4) and SPAdes (k=3, 4) the NGA50 values are undefined

All other methods evaluated (HapCut2, Phaser, Whatshap and SPAdes-dip) are designed for diploid data, so for those we could only assess the effect of sequencing depth. Results indicate that each of the reference-guided methods already performs optimally at a coverage per haplotype of 5×. Moreover, these methods are unaffected by a further increase in sequencing depth (see Supplementary Table S12). SPAdes in diploid mode (SPAdes-dip) performs optimally at a per-haplotype coverage 20×.

4 Discussion

Assembling the individual haplotypes of an organism from sequencing reads is known as *haplotype-aware genome assembly* and plays a major role in various disciplines, including genetics and medicine (Glusman *et al.*, 2014; Tewhey *et al.*, 2011). Computing haplotype-specific pieces of sequence, also known as *haplotigs*, is a difficult task. Algorithms addressing this task do not only need to distinguish between sequencing errors and true variants, but also need to assign the true variants to the individual haplotypes. Enormous quantities of NGS reads generated worldwide have not been fully exploited in terms of haplotig computation, because methodology for *de novo* haplotig computation from NGS reads has been in a rather immature state.

We have presented POLYTE as a new approach to *de novo* assembly of haplotigs from NGS data, suitable for diploid genomes as well as genomes of higher ploidy. Unlike the majority of NGS based *de novo* assemblers, our method follows the OLC paradigm to achieve enhanced performance rates in terms of haplotype-specific computation of contigs. In order to appropriately distinguish between errors and true variants to be assigned to haplotypes, it employs an iterative OLC scheme. Along the iterations, contigs grow in length while preserving their uniqueness in terms of haplotype identity. As a result, POLYTE outperforms the currently available state-of-the-art approaches for haplotig computation, where it performs particularly favorable in terms of quantities that refer to haplotype-specific reconstruction of the genomes.

Experimental results showed that POLYTE can build accurate assemblies from Illumina MiSeq reads (2×250 bp) for datasets of varying ploidy (di-, tri- and tetraploid), with results for tetraploid organisms almost on a par with those for diploid organisms. Although building overlap graphs for larger genomes remains a challenge, we provide a read binning step that allows efficient assembly by splitting the work over multiple cores. The typical use-case for POLYTE consists of Illumina NGS reads for a specific gene, region or genome that is highly polymorphic (of any ploidy >1).

We showed that POLYTE succeeds in accurate reconstruction of individual haplotypes of the human MHC region. Future work may therefore be to apply POLYTE to NGS data in population-scale human genome projects, where individual genomes are still lacking proper annotation of their MHC region, which applies in the majority of cases. Advantages become particularly distinct on data of higher ploidy, leading to plant genome assembly as another interesting future application of POLYTE.

Our algorithm is, in its essence, generic in the choice of input reads, so applying it for TGS reads essentially is a matter of adapting parameters, which we will explore in the short-term future.

Funding

This work was supported by the Netherlands Organisation for Scientific Research (NWO) through Vidi grant [679.072.309].

Conflict of Interest: none declared.

References

1000 Genomes Project Consortium *et al.* (2012) An integrated map of genetic variation from 1,092 human genomes. *Nature*, **491**, 56–65.
Aguiar,D. and Istrail,S. (2012) HapCompass: a fast cycle basis algorithm for accurate haplotype assembly of sequence data. *J. Comput. Biol.*, **19**, 577–590.
Baaijens,J.A. *et al.* (2017) De novo assembly of viral quasispecies using overlap graphs. *Genome Res.*, **27**, 835–848.
Bankevich,A. *et al.* (2012) SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J. Comput. Biol.*, **19**, 455–477.
Berger,E. *et al.* (2014) HapTree: a novel Bayesian framework for single individual polyplotyping using NGS data. *PLOS Comput. Biol.*, **10**, 1–10.
Besenbacher,S. *et al.* (2015) Novel variation and de novo mutation rates in population-wide de novo assembled Danish trios. *Nat. Commun.*, **6**, 5969.
Bradnam,K.R. *et al.* (2013) Assemblathon 2: evaluating de novo method of genome assembly in three vertebrate species. *GigaScience*, **2**, 10.
Castel,S.E. *et al.* (2016) Rare variant phasing and haplotypic expression from RNA sequencing with phaser. *Nat. Commun.*, **7**, 12817.

Chaisson,M.J.P. *et al.* (2019) Multi-platform discovery of haplotype-resolved structural variation in human genomes. *Nat Commun.*, **10**, 1784.

Chin,C. *et al.* (2016) Phased diploid genome assembly with single molecule real-time sequencing. *Nat. Methods*, **13**, 1050–1054.

Das,S. and Vikalo,H. (2015) SDhaP: haplotype assembly for diploids and polyploids via semi-definite programming. *BMC Genomics*, **16**, 260.

Edge,P. *et al.* (2017) HapCUT2: robust and accurate haplotype assembly for diverse sequencing technologies. *Genome Res.*, **27**, 801–812.

Glusman,G. *et al.* (2014) Whole-genome haplotyping approaches and genomic medicine. *Genome Med.*, **6**, 73.

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

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

Jain,M. *et al.* (2018) Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nat. Biotechnol.*, **36**, 338–345.

Kajitani,R. *et al.* (2014) Efficient de novo assembly of highly heterozygous genomes from whole-genome shotgun short read. *Genome Res.*, **24**, 1384–1395.

Kuleshov,V. (2014) Probabilistic single-individual haplotyping. *Bioinformatics*, **30**, i379–i385.

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, **25**, 1754–1760.

Marschall,T. *et al.* (2012) CLEVER: clique-enumerating variant finder. *Bioinformatics*, **28**, 2875–2882.

Patterson,M. *et al.* (2015) WhatsHap: weighted haplotype assembly for future generation sequencing reads. *J. Comput. Biol.*, **22**, 498–509.

Pirola,Y. *et al.* (2016) HapCol: accurate and memory-efficient haplotype assembly from long reads. *Bioinformatics*, **32**, 1610–1617.

Porubsky,D. *et al.* (2017) Dense and accurate whole-chromosome haplotyping of individual genomes. *Nat. Commun.*, **8**, 1293.

Ribeiro,F.J. *et al.* (2012) Finished bacterial genomes from shotgun sequence data. *Genome Res.*, **22**, 2270–2277.

Safonova,Y. *et al.* (2015) dipSPAdes: assembler for Highly Polymorphic Diploid Genomes. *J. Comput. Biol.*, **22**, 528–545.

Salzberg,S.L. *et al.* (2012) GAGE: a critical evaluation of genome assemblies and assembly algorithms. *Genome Res.*, **22**, 557–567.

Simpson,J.T. and Durbin,R. (2012) Efficient de novo assembly of large genomes using compressed data structures. *Genome Res.*, **22**, 549–556.

Sudmant,P.H. *et al.* (2015) An integrated map of structural variation in 2504 human genomes. *Nature*, **526**, 75–81.

Tewhey,R. *et al.* (2011) The importance of phase information for human genomics. *Nat. Rev. Genet.*, **12**, 215.

The Genome of the Netherlands Consortium (2014) Whole-genome sequence variation, population structure and demographic history of the Dutch population. *Nat. Genet.*, **46**, 818–825.

The UK10K Consortium (2015) The UK10K project identifies rare variants in health and disease. *Nature*, **526**, 82–90.

Töpfer,A. *et al.* (2014) Viral quasispecies assembly via maximal clique enumeration. *PLoS Comput. Biol.*, **10**, e1003515.

Weisenfeld,N.I. *et al.* (2017) Direct determination of diploid genome sequences. *Genome Res.*, **27**, 757–767.

Xie,M. *et al.* (2016) H-PoP and H-PoPG: heuristic partitioning algorithms for single individual haplotyping of polyploids. *Bioinformatics*, **32**, 3735–3744.