de novo diploid genome assembly using long noisy reads via haplotype-aware error correction and inconsistent overlap identification

Fan Nie[1,2], Neng Huang[1,2], Jun Zhang[1,2], Peng Ni[1,2], Zhenyu Wang[3], ChuanLe Xiao[4,*], Feng Luo[5,*], Jianxin Wang[1,2,*]

[1] School of Computer Science and Engineering, Central South University, Changsha, 410083, China

[2] Hunan Provincial Key Lab on Bioinformatics, Central South University, Changsha, 410083, China

[3] Institute of Nanfan & Seed Industry, Guangdong Academy of Sciences, Guangdong, 510316, China

[4] State Key Laboratory of Ophthalmology, Zhongshan Ophthalmic Center, Sun Yat-sen University #7 Jinsui Road, Tianhe District, Guangzhou, P.R. China

[5] School of Computing, Clemson University, Clemson, SC 29634-0974

*To whom correspondence should be addressed:

Chuan-Le Xiao: Tel: +86 20 66686996. Email: xiaochuanle@126.com;

Feng Luo. Tel: +01 864 656 7552. Email: luofeng@clemson.edu;

Jian-Xin Wang. Tel: +86 731 88830212. Email: jxwang@mail.csu.edu.cn;

Keywords: long noisy reads, error correction, haplotype-aware, inconsistent overlap, diploid genome assembly

Abstract

High sequencing errors have impeded the application of long noisy reads for diploid genome assembly. Most existing assemblers failed to distinguish heterozygotes from high sequencing errors in long noisy reads and generate collapsed assemblies with lots of haplotype switches. Here, we present PECAT, a **p**hased **e**rror **c**orrection and **a**ssembly **t**ool for reconstructing diploid genomes from long noisy reads. We design a haplotype-aware error correction method that can retain heterozygote alleles while correcting sequencing errors. We develop a read-level SNP caller that can further reduce the SNP errors in corrected reads. Then, we use a read grouping method to assign reads to different haplotype groups. To accelerate the assembling, PECAT only performs local alignment when it is necessary. PECAT efficiently assembles diploid genomes using only long noisy reads and generates more contiguous haplotype-specific contigs compared to other assemblers. Especially, PECAT achieves nearly haplotype-resolved assembly on *B. taurus* (Bison×Simmental) using Nanopore reads.

Introduction

De novo genome assembly is a fundamental task in genomic research[1–3]. Using long-reads from sequencing technologies, such as single-molecule real-time (SMRT) sequencing of Pacific Biosciences (PacBio) and Nanopore sequencing of Oxford Technologies (ONT), many assemblers now can effectively reconstruct high-quality genome sequences for haploid or inbred species. However, many genetic information of a diploid genome is lost in those assemblies and most of them have lots of haplotype switch errors. Because long noisy reads such as PacBio CLR reads and Nanopore reads usually contain a 5-15% sequencing error rate[4], it is difficult to distinguish heterozygotes from sequencing errors in long noisy reads, which prevents the diploid assemblers from generating long haplotype-specific contigs. Recently, by combining high accurate PacBio HiFi reads[5] ($\leq$ 1% error rate) with additional sequences, such as parental short reads[6], Hi-C reads[7,8], Strand-seq data[9], or gamete cell data[10], assemblers now can produce a more contiguous haplotype-specific contigs[7,11,12]. However, the requirements of additional sequencing increase costs and limit their applications in practice. Furthermore, the average lengths of long noisy reads are longer than those of HiFi reads (10-25 kb). For example, the ultra-long reads of Nanopore are up to 1M in length and with read N50 >100 kb. Longer reads can help to assemble more contiguous contigs. Therefore, it will be useful to develop an assembler for diploid genomes using long noisy reads only, especially for species with high and moderate heterozygosity.

To achieve high-quality assemblies, error correction is usually a useful step for genome assembling using long noisy reads. One challenge to assembling a diploid

genome using long noisy reads is how to retain heterozygotes during sequence error correction. Current correct-then-assemble pipeline[13–16] eliminated heterozygotes as sequencing errors, or mixed alleles of different haplotypes in a read. Therefore, corrected reads don't contain heterozygote information for haplotype phasing. Assemblers, such as FALCON-Unzip[13], used raw reads instead of corrected reads in their "unzip" phasing step, then mapped the raw reads to the corrected reads for the later diploid assembly. This has led to assembly errors due to the loss of heterozygotes in the corrected reads. Furthermore, phasing raw reads is more difficult due to their high error rate.

Another challenge in the phasing step is to partition reads according to their haplotypes. After error correction, there still are sequencing errors left in corrected reads. And the error profile of corrected reads is more complex than that of PacBio HiFi reads. In many parts of the diploid genome, the sequencing error of corrected reads is still close to or even exceeds haplotype divergence. This leads to inconsistent overlaps[11] among reads, whose reads come from different haplotypes or different copies of the segmental duplication. The inconsistent overlaps can cause haplotype switch errors or unresolved repeats (**Supplementary Figure 1**). In our tests, 13.60-36.78% of edges in simplified overlap graphs are inconsistent overlaps during assembling diploid genomes. Identifying and removing inconsistent overlaps is important in overlap-graph-based diploid genome assembly, even if we assemble the genome using PacBio HiFi reads[11]. The false-positive SNPs in corrected reads make identifying inconsistent overlaps more

difficult. It is necessary to design a more accurate and robust method to identify inconsistent overlaps for long noisy read-based assemblers.

Furthermore, to construct two haplotypes of diploid genomes, the assemblers often need more sequencing data. However, with the increasing amount of sequencing data, the running time of assembly increases nonlinearly. Therefore, assembling large diploid genomes at an acceptable time is another challenge. Overlap-graph-based assemblers have their unique advantages to assemble diploid genomes since the overlaps can be reused in subsequent steps once they are found. This helps to design an efficient multi-round assembly strategy. The overlap-graph-based assemblers usually use seed (i.e., k-mers, minimizers) based methods to find candidate overlaps, then perform local alignment to find the true overlaps. The local alignment is the major computational bottleneck of overlap-graph-based approaches. Although skipping local alignment can accelerate the overlap finding step[17], it leads to lots of false-positive overlaps and introduces errors in assembly graphs. String-graph-based[18] approaches, a type of overlap-graph-based approach, ignore the reads contained by other reads and mark most edges in the graph as transitive edges[18] that don't contribute to the construction of contigs. Then, it is not necessary to perform local alignment on those read pairs. Therefore, it is possible to design a fast string-graph-based diploid genome assembler by minimizing the number of local alignments needed.

In this work, we present PECAT, a phased error correction and assembly tool designed to reconstruct diploid genomes from long noisy reads, including PacBio CLR reads and Nanopore reads. PECAT follows the correct-then-assemble strategy,

including an error correction module and a two-round string graph-based assembly module (**Figure 1**). We design a haplotype-aware error correction method for long noisy diploid reads, which can retain heterozygote alleles while correcting sequencing errors. In the first round of assembly, we assemble the corrected reads and generate haplotype-collapsed contigs. In the second round of assembly, we identify and remove the inconsistent overlaps and assemble the corrected reads to -prmary contigs (long contigs with the mosaic of homologous haplotypes) and alternate contigs (short haplotype-specific contigs). We propose a read-level SNP caller and a read grouping method for identifying inconsistent overlaps, which can effectively eliminate the effects of sequencing errors and false-positive SNPs. To accelerate the assembling, PECAT only performs local alignment when it is necessary instead of performing local alignments on all candidate overlaps. PECAT can efficiently assemble diploid genomes using only long noisy reads and generate more contiguous haplotype-specific contigs compared to other assemblers.

Result

Haplotype-aware error correction

Our error correction method is based on the partial-order alignment (POA) graph[19] method. For the template read to be corrected, a POA graph is built from the alignment of supporting reads. Then, the path with the highest weight is found in the POA graph to construct the consensus sequence for the template read. However, if the sequencing error rate exceeds haplotype divergence, current methods inevitably select supporting reads from different haplotypes. This causes either heterozygote alleles in the template

reads to be eliminated as sequencing errors or heterozygous alleles from different haplotypes to be mixed in corrected reads.

To maintain the heterozygote alleles in the template read, we need to select supporting reads from the same haplotype to correct it. After analyzing the POA graph, we have found that the difference between random sequencing errors and heterozygotes can be reflected in the POA graph. In the case of heterozygotes, there are two dominant parallel branches in the POA graph. In the case of random errors, there tends to be only one dominant branch. Based on this finding, we design a scoring algorithm to estimate the likelihood that the supporting read and the template read are from the same haplotype (**Methods**). For each position at which there are two dominant parallel branches, if the supporting read and the template read pass through the same dominant branch, the score is increased by 1 and if the supporting read and the template read pass through the different dominant branches, the score is decreased by 1. A higher score means that the template read and the supporting read are more likely from the same haplotype. If the supporting reads come from different haplotypes, the histogram of scores of all supporting reads should show two peaks (**Supplementary Figure. 2 c)**. Then, we select the high-scoring supporting reads, which are very likely to be in the same haplotype with the template read, to correct the template read. To further increase the likelihood of selecting supporting reads in the same haplotype, we assign different weights to the reads according to their score. The higher the score of a read, the larger its weight is assigned (**Methods**). Then, we remove unselected reads in the POA graph

by assigning their weights to 0 and find the path with the highest weight in the POA graph using dynamic programming and backtracking.

We evaluate the performance of the selection method on the diploid datasets with reads classified by the trio-binning algorithm[6] (**Methods**). As shown in **Supplementary Table 1**, for the four PacBio CLR datasets of *S. cerevisiae* (SK1×Y12), *A. thaliana* (COL×CVI), *D. melanogaster* (ISO1×A4), and *B. taurus* (Angus×Branman), without the selection method, 32.3%, 32.1%, 40.6% and 26.0% of supporting reads are inconsistent reads, respectively, which are from the haplotype different from the one of template read. Meanwhile, there are only 5.3%, 9.1% 7.1% and 8.5% inconsistent reads in selected supporting reads, respectively. After further re-weighting the scores, the percentages of inconsistent reads are reduced to 3.3%, 6.7%, 4.8% and 6.6%, respectively. For the Nanopore datasets of *A. thaliana* (COL×C24) and *B. taurus* (Bison×Simmental), we implement two rounds of error correction and the selection method outputs similar results. After re-weighting the scores, percentages of inconsistent reads in selected reads are 5.4% and 4.3% in the first round of error correction and 3.6% and 1.2% in the second round of error correction.

Since most of the selected supporting reads are from the same haplotype of the template read, most heterozygote alleles in the template read are correctly retained in our error correction method. We compare our method with other methods including Canu[14], MECAT2[15], and NECAT[16] on simulated PacBio and Nanopore data (**Supplementary Note 1** and **Supplementary Table 2**). The accuracy of SNP alleles

in reads corrected by PECAT is 98.62-99.95%, while the accuracies of SNP alleles in reads corrected by other methods are even worse than that of raw reads.

Fast string graph-based assembler

After error correction, PECAT implements two rounds of string graph-based assembly. In each round of assembly, we first construct the overlaps between the corrected reads using the seed-based alignment method (minimap2[20]), which allows us to build the overlaps quickly. However, seed-based alignment can bring low-quality overlaps with low identity or with long overhangs. Those low-quality overlaps could introduce errors during assembling. Performing local alignment on overlaps to identify low-quality ones becomes the major computational cost. To speed up the assembling, PECAT only performs local alignments when it is necessary during the construction of the string graph. (**Methods**). First, to reduce overhangs of overlaps, we use diff algorithm[21] to extend the candidate overlaps to the ends of the reads. Here, we only perform local alignment on overhangs instead of on the whole overlap. We remove the overlaps if their overhangs are still long ($\min(100, 0.01 \cdot l)$ for PacBio reads and $\min(300, 0.03 \cdot l)$ for Nanopore reads, where l is length of the read). On the diploid datasets of *S. cerevisiae* (SK1×Y12), *A. thaliana* (COL×CVI), *D. melanogaster* (ISO1×A4), *B. taurus* (Angus×Branman), *A. thaliana* (COL×C24) and *B. taurus* (Bison × Simmental), 5.8%, 9.8%, 81.2%, 13.5%, 33.0% and 80.2% of candidate overlaps with long overhangs have been considered as low quality and removed, respectively (**Supplementary Table 3**). Then, we further filter out the overlaps whose reads are contained in other reads or with low coverage. Only 0.2%, 0.2%, 0.8%, 0.7%, 0.1% and

0.4% of overlaps are remained on the above six diploid datasets (**Supplementary Table 3**). We then construct a directed string graph from the remaining overlaps. We find the transitive edges using Mayer's algorithm[18] and mark them as inactive edges, which are not used for constructing contigs. On the above diploid datasets, only 24.9%, 17.0%, 15.4%, 18.1%, 21.4% and 15.2% of edges are active (**Supplementary Table 3**).

To remove low-quality edges in the string graph, we calculate the identity of the overlaps (the active edges) in the string graph using local alignments. Since most edges have been marked as inactive, we only need to perform local alignments for a small portion of overlaps. On the above diploid datasets, 1.5%, 15.0%, 13.2%, 2.8%, 17.4%, and 7.4% of active edges have been removed because their identities are less than the threshold (0.95 by default) (**Supplementary Table 3**). After the above step, some paths in the graphs connected by low-quality edges are broken. And those broken paths need to be connected using the transitive edges that have been labeled inactive edges. We then select transitive edges with the longest alignment and their identity greater than the threshold to connect the broken paths. On the above diploid datasets, about 0.4%, 0.3%, 0.2%, 0.1%, 0.3%, and 0.1% of transitive edges have been reactivated (**Supplementary Table 3**). In the first round of assembly (**Figure 1b**), PECAT finds linear paths from this string graph and constructs haplotype-collapsed contigs. In the second round of assembly (**Figure 1c**), PECAT identifies and removes inconsistent overlaps (next Section) in the string graph, and then generates primary and alternate contigs.

Identification of inconsistent overlaps

The inconsistent overlaps connect reads from different haplotypes or different copies of the segmental duplication in the overlap graph. After identifying and filtering out inconsistent overlaps, only the overlaps between reads from the same haplotype or the same copies of the segmental duplication are left, and then the assembler can naturally generate contigs of two haplotypes. Since our corrected reads contain allele information, we may use the SNP allele information to identify inconsistent overlaps. If the SNP alleles on a pair of reads are different, these two reads should come from different haplotypes and the overlaps between them should be inconsistent. At the error correction step, we have scored the possibility that two raw reads come from the same haplotype. However, the accuracy of the previous scoring that is based on the alignments between two raw reads cannot meet the requirements for identifying inconsistent overlaps. Therefore, we develop a read-level SNP caller and a read grouping method using our corrected reads for identifying inconsistent overlaps.

First, we map all corrected reads onto the haplotype-collapsed contigs from the first round of assembly using minimap2[20]. We call the heterozygous SNP sites based on the base frequency of the alignments (**Methods**). Then, we set each read as a template read and collect a set of reads that have the common SNP sites with it. We cluster the set of reads according to the SNP alleles in reads. The reads from the same haplotype are more likely to be clustered into the same subgroup. We then verify and correct the SNP alleles in the template read using other reads in the same subgroups. After identifying SNP alleles in each read, we remove the inconsistent overlaps using

the SNP information from the candidate overlaps found in the first round of assembly. Then, we construct the string graph again.

Our method can effectively eliminate the effects of sequencing errors and false-positive SNPs and improve the accuracy of identifying inconsistent overlaps. We evaluate the performance of the inconsistent overlaps identification method using the haplotype reads classified by the trio-binning algorithm[6]. As shown in **Supplementary Table 4**, the precisions of inconsistent overlap identification method on the diploid datasets *S. cerevisiae* (SK1×Y12), *A. thaliana* (COL×CVI), *D. melanogaster* (ISO1×A4), *B. taurus* (Angus×Branman), *A. thaliana* (COL×C24) and *B. taurus* (Bison × Simmental) are 97.70%, 98.19%, 99.13%, 87.63%, 97.13% and 93.23% respectively, and the recalls of the method are 63.15%, 87.41%, 88.20%, 88.27%, 66.77% and 94.79, respectively. Furthermore, compared with the first round of assembly, the percentages of inconsistent overlaps in simplified graphs of the second round of assembly decrease sharply. For example, for *S. cerevisiae* (SK1×Y12), the percentage of inconsistent overlaps has decreased from 36.78% in first round of assembly to 0.3% in second round of assembly; and for *B. taurus* (Angus×Branman), the percentage of inconsistent overlaps has decreased from 33.92% in first round of assembly to 1.01% in second round of assembly. The high performance of the inconsistent overlap identification method ensures that PECAT can generate the haplotype-specific contigs effectively.

Moreover, our inconsistent overlap identification method can also help solve nearly identical repeats without knowing the number of their copies. The clustering step can automatically separate nearly identical repeats if there are SNPs that can divide the

repeats. After filtering out the inconsistent overlaps at the repeats, PECAT can solve the repeat to generate contiguous contigs in the second round of assembly. As a result, PECAT achieves more contiguous assemblies in the second round of assembly on most of our diploid datasets (**Supplementary Table 5**). As shown in **Supplementary Figures 3,4**, PECAT perfectly solves repeats in the NCTC9024 and NCTC9006 datasets and reconstructs two circle contigs, while other assemblers obtain fragment results[22,23].

Performance of PECAT error correction method

We evaluate the performance of the PECAT error correction method using four Pacbio diploid datasets: *S. cerevisiae* (SK1×Y12), *A. thaliana* (COL×CVI), *D. melanogaster* (ISO1×A4), *B. taurus* (Angus×Branman), and two Nanopore diploid datasets: *A. thaliana* (COL×C24) and *B. taurus* (Bison × Simmental). We compare PECAT with the error correction methods in the other four tools, including Canu[14], FALCON[13], MECAT2[15], and NECAT[16]. We first assess the quality of corrected reads with respect to the accuracy, maximum length, and N50 of 40X longest corrected reads. As shown in **Table 1**, all methods have similar performance on these four metrics. We then assess the haplotype-specific k-mers completeness and consistency of corrected reads (**Methods**). These two metrics can evaluate the ability to retain consistent heterozygote alleles. **Table 1** shows that reads corrected by PECAT have higher consistency and haplotype-specific k-mers completeness in all datasets. Especially, haplotype-specific k-mers consistencies of reads corrected by PECAT are greater than or equal to 98.4% on all datasets, while haplotype-specific k-mers consistencies of

reads corrected by other methods are all less than or equal to 94.6%. This demonstrates the advantages of PECAT for correcting diploid sequences. The reads corrected by PECAT contain more haplotype-specific k-mers than those corrected by other methods (**Figure 2** and **Supplementary Figures 5-9)**. Especially, PECAT effectively avoids mixing heterozygous alleles from two haplotypes and the corrected reads tend to contain only one type of haplotype-specific k-mer (**Figure 2** and **Supplementary Figures 5-9)**.

Performance of PECAT assembler

We also assess the performance of the PECAT assembler using four PacBio diploid datasets: *S. cerevisiae* (SK1×Y12), *A. thaliana* (COL×CVI), *D. melanogaster* (ISO1×A4), *B. taurus* (Angus×Branman), and two Nanopore diploid datasets: *A. thaliana* (COL×C24) and *B. taurus* (Bison × Simmental). We compare PECAT with four diploid genome assembly pipelines: Canu[14]+purge_dups[24], FALCON-Unzip[13], Flye[22]+HapDup[25], and Shasta[26] (**Supplementary Note 2)**.

We first evaluate assembled genomes with respect to the assembly size, number of contigs, max length, and NG50 of primary and alternate contigs. As shown in **Table 2**, the sizes of both primary and alternate contigs of all six PECAT assemblies are close to their corresponding reference genome, except the *S. cerevisiae* (SK1×Y12) and *A. thaliana* (COL×C24) genomes assembled by Canu+purge_dups. Compared to the other four pipelines, PECAT obtains the highest max length of contigs and the highest NG50 for all six assemblies, except max length of primary contigs on the *D. melanogaster* (ISO1×A4) genome assembled by Canu+purge_dups. Especially, the max lengths and

NG50s of alternate contigs assembled by PECAT are significantly longer than those assembled by other pipelines, except the *B. taurus* (Angus×Branman) genome assembled by FALCON-Unzip. Although the number of primary contigs reported by PECAT is similar to those reported by other pipelines, the numbers of alternate contigs reported by PECAT are significantly smaller. Moreover, using only Nanopore reads, PECAT assembles a *B.taurus* (Bison×Simmental) diploid genome with max contig length (156.02/155.93 Mb) and NG50 (73.03/68.13) of primary and alternate contigs. This assembly is comparable to the reference genome with max contig length (157.90/156.88 Mb) and NG50 (87.75/104.40 Mb) of primary and alternate contigs, which is assembled using trio-binning methods using paternal reads and Hi-C reads[27].

Then, we evaluate the quality of assembled genomes using BUSCO[28] and the "Quality", "Haploblock NG50" and "Intra-block switch error" from merqury[29]. For four genomes assembled using Pacbio reads, all pipelines have similar BUSCO scores, except that Canu+purge_dups has low BUSCO scores on alternate contigs. For two genomes assembled using Nanopore reads, Flye+HapDup had the highest BUSCO, and Shasta had the lowest BUSCO scores. This may be because Flye+HapDup has used all raw data to polish the contigs. All five pipelines report similar "Quality" scores on six assemblies. For the metrics 'Haploblock NG50' and 'Intra-block switch error', PECAT reports the longest haplotype-specific contigs with fewer haplotype switch errors on most datasets. For four genomes assembled using Pacbio reads, PECAT reports less 'Intra-block switch error' for *S. cerevisiae* (SK1×Y12) and *D. melanogaster* (ISO1×A4), and reports similar 'Intra-block switch error' for the other two genomes

compared to FALCON-Unzip, while both report significantly less 'Intra-block switch error' than those reported by Canu+purge_dups. For two genomes assembled using Nanopore reads, PECAT and Flye+HapDup report similar 'Intra-block switch errors', which are significantly less than those reported by Canu+purge_dups and Shasta. On *B. taurus* (Bison×Simmental), PECAT reports an assembly with 'haplotype block NG50' of 72.34/68.13 Mb, which is close to the assembly with haplotype block N50 of 68.47/70.61Mb reported by the trio-binning method using additional parental reads[27]. Furthermore, as shown in **Figure 3 and Supplementary Figures 10-14**, PECAT has separated the primary contigs and alternate contigs (haplotigs) well compared to the other four pipelines.

We further compare the computational resources required by each pipeline. For the first three PacBio CLR datasets of *S. cerevisiae* (SK1×Y12), *A. thaliana* (COL×CVI), and *D. melanogaster* (ISO1×A4), PECAT is 9.0-18.4 times faster than Canu+purge_dups, 16.5-21.6 times faster than FALCON-Unzip. For the Nanopore datasets *A. thaliana* (COL×C24), and *B. taurus* (Bison×Simmental), Shasta is the fastest assembly pipeline, while the running times of PECAT are close to those of Flye+HapDup and both are much faster than Canu+purge_dups.

Discussion

Currently, directly using long noisy reads for diploid genome assembly remains a challenge because it is difficult to distinguish haplotype differences from sequencing errors. In this study, we develop PECAT to overcome sequencing errors in long noisy reads and construct the diploid genome. The PECAT includes a haplotype-aware error

correction method, a read-level SNP caller and a read grouping method. The haplotype-aware error correct method has kept most of the heterozygote alleles while correcting sequencing errors. The read-level SNP caller further reduces the SNP errors in corrected reads. And the read grouping method assigns reads to different haplotype groups. Those three methods together allow PECAT to eliminate the impact of sequencing errors and identify inconsistent overlaps, and then generate more contiguous haplotype-specific contigs with fewer errors. Our experimental results show that PECAT is an efficient error correction and assembly pipeline for diploid genomes.

In addition, the read grouping method can also help separate reads from nearly identical repeats with multiple copies. By assign reads into different group without knowing the number of their copies, the read grouping methods can identify inconsistent overlaps between reads from those nearly identical repeats in genome, which then help solve those repeats during assembly. Previous methods, such as FALCON-Unzip, have assumed that there are only two copies of reads in collapsed contigs, and then phases the reads using heterozygous SNPs information in local blocks to separate collapsed contigs into two contigs only, which cannot solve the repeats with more than two copies.

Methods

Diploid datasets for benchmarking

We evaluate the performance of PECAT using six diploid datasets (**Supplementary Note 1** and **Supplementary Table 6**): *S. cerevisiae* (SK1×Y12), *A. thaliana* (COL×CVI), *D. melanogaster* (ISO1×A4), *B. taurus* (Angus×Branman), *A. thaliana* (COL×C24) and *B. taurus* (Bison × Simmental). The first four datasets are PacBio CLR datasets and the last two are Nanopore datasets (the last one is ultra-long raw reads). The heterozygosity of those species is 0.85%, 1.04%, 0.84%, 1.12%, 1.63% and 1.48% respectively (**Supplementary Table 1, Supplementary Note 3**). The accuracies of reads in those datasets are 87.80%, 88.24%, 89.58%, 86.19%, 92.23% and 88.96% respectively. (**Table 1**).

Haplotype-aware error correction

PECAT error correction method is based on the partial-order alignment (POA) graph method[13, 30]. Instead of assigning the same weight to each read for error correction, we select reads more likely coming from the same haplotype or the same copy of the nearly identical repeat and assign different weights to prevent heterozygotes in reads from being eliminated as sequencing errors. First, we find candidate overlaps between raw reads using minimap2 with parameters "-x ava-pb" for PacBio CLR reads and parameters "-x ava-ont" for Nanopore reads. For each template read to be corrected, we collect a group of supporting reads that have candidate overlaps with the template read. Then, we perform pairwise alignment between the template read and each supporting read using diff[21] or edlib[31] algorithm and build a POA graph based on the alignments

of the reads as shown in **Supplementary Figure. 2**. Each node in the graph is labeled by a triple value (c, r, b), corresponding a base pair in the alignment. c is the location at the template read, r means the number of consecutive insertions (if r=0, there is a match, a mismatch or a deletion), and b is the base on the supporting read or a deletion, which is one of $\{'A',' C',' G',' T',' -'\}$. Each edge in the graph means two base pairs of their nodes appearing continuously in the alignment. The support count of each edge is defined as the number of alignments which pass the edge. For the convenience of analysis, we add a trivial alignment between the template read and itself to the graph. According to our observations, for each location c, all paths in the graph must only pass one of nodes $\{(c, 0, b_i), b_i \in ('A',' C',' G',' T',' -')\}$. If these nodes have more than one in-edge with large support counts, there may be heterozygotes rather than random sequencing errors. Therefore, we compute support count s_i for in-edges of the nodes. We mark an in-edge as the important one if $s_i \geq \begin{cases} r_l \cdot S & S \leq S_l \\ \frac{r_h \cdot S_h - r_l \cdot S_l}{S_h - S_l} \cdot (S - S_l) + r_l \cdot S_l & S_l < S \leq S_h \\ r_h \cdot S & S_h < S \end{cases}$, where $S = \sum s_i$, and r_l, r_h, S_l, and S_h are user-set parameters (the default values are 0.5, 0.2, 10 and 200). We mark a location c as the important one if there are two or more important in-edges. Then, we calculate a score for each supporting read based on whether the supporting read and the template read pass the same important in-edges at important locations. We increase the score by 1 if a supporting read passes the same important in-edge as template read and decrease it by 1 if the supporting read and the template read pass the different important in-edges. For uniformity, the score is divided by the number of important locations in the

supporting read. As shown in **Supplementary Figure. 2 c**, the histogram of scores shows two or more peaks if the supporting reads come from different haplotypes or different copies of the segmental duplication. We select the reads whose scores fall into the first peak for error correction. If there is only one peak, we select half of the supporting reads with larger scores. We linearly map the scores of selected reads to a range, which is [0.2,0.8] by default, as the final weights of supporting reads. The weight of each edge is the sum of the weights of selected supporting reads that pass the edge. Finally, we find the path with the highest weight in the POA graph by dynamic programming and backtracking. The consensus sequence for the template read is generated by concatenating the bases of the nodes in the path with the highest weight.

Read-level SNP caller and read grouping method for identifying inconsistent overlaps

Calling heterozygous SNPs in haplotype-collapsed contigs and SNP alleles in reads

We map corrected reads to the first round of assembly using minimap2 with parameters "-x map-pb -c -p 0.5 -r 1000" for PacBio reads and parameters "-x map-ont -c -p 0.5 -r 1000" for Nanopore reads. It performs base-level alignment and generates CIGAR strings. We scan the CIGRA strings and call heterozygous SNPs for each contig. We call a base site as a heterozygous SNP site if it meets the following two conditions. (1) Its coverage is in the range [10, 1000]. (2) The number of second-most common bases is greater or equal to $\begin{cases} r_l \cdot c & c \leq c_l \\ \frac{r_h \cdot c_h - r_l \cdot c_l}{c_h - c_l} \cdot (c - c_l) + r_l \cdot c_l & c_l < c \leq c_h \\ r_h \cdot c & c \geq c_h \end{cases}$, in which c is

the site coverage and r_l, r_h, c_l, and c_h are user-set parameters (default values are 0.4, 0.2, 10 and 100). After calling heterozygous SNPs, we call SNP alleles in reads. We define a function $H_r(s)$ for each read r. For any SNP site s, $H_r(s)$ is defined as 1 or 2 if the read r covers the site s and the base of the read r is equal to the first-most or second-most common base at site s. Otherwise, $H_r(s)$ is defined as 0.

Verifying and correcting SNP alleles

To verify and correct SNP alleles in a read, we need to find which reads are from the same haplotype within its local region. For each read labeled as the template read t, we assume that it covers a set of SNP sites $S = \{s_1, s_2, \dots, s_N\}$. We collect the query reads that cover 3 common SNP sites with the template read t. The template read and the query reads are put into a group G. We cluster the reads in the group G according to the SNP alleles in them. The reads in the same cluster can be considered from the same haplotype. To facilitate the description of the method, here we make some definitions. We define a centroid $C = \{(v_s, n_s) | s \in S\}$ for the group G at the SNP site set S. v_s is defined to $v_s^+ - v_s^-$, where v_s^+ is the number of the read set $\{r \mid H_r(s) = 1, r \in G, s \in S\}$ and v_s^- is the number of the read set $\{r \mid H_r(s) = 2, r \in G, s \in S\}$. n_i is defined to $v_s^+ + v_s^-$. One read can be regarded as a group only including it. We define the following three formulas to get verified SNP sites of the centroid C.

$$V_>(C, p_0, p_1) = \{s | v_s > 0, |v_s| \geq \max(p_0 \cdot n_s, p_1)\}, (v_s, n_s) \in C\}$$

$$V_<(C, p_0, p_1) = \{s | v_s < 0, |v_s| \geq \max(p_0 \cdot n_s, p_1)\}, (v_s, n_s) \in C\}$$

$$V_{\neq}(C, p_0, p_1) = V_>(C, p_0, p_1) \cup V_<(C, p_0, p_1)$$

The parameters p_0 and p_1 are used to control which sites are valid. Next, we define the operations between two centroids.

$$A(C_1, C_2, p_0, p_1) = V_{\neq}(C_1, p_0, p_1) \cap V_{\neq}(C_2, p_0, p_1)$$

$$S(C_1, C_2, p_0, p_1) = V_{<}(C_1, p_0, p_1) \cap V_{<}(C_2, p_0, p_1) \cup V_{>}(C_1, p_0, p_1) \cap V_{>}(C_2, p_0, p_1)$$

$$D(C_1, C_2, p_0, p_1) = V_{<}(C_1, p_0, p_1) \cap V_{>}(C_2, p_0, p_1) \cup V_{>}(C_1, p_0, p_1) \cap V_{<}(C_2, p_0, p_1)$$

$A(C_1, C_2, p_0, p_1)$ is the set of common verified SNP sites of the centroids C_1 and C_2. $S(C_1, C_2, p_0, p_1)$ and $D(C_1, C_2, p_0, p_1)$ are used to describe the similarity and the distance of the centroids C_1 and C_2. A read r can also be used as the parameter, which means the centroid of $\{r\}$.

For each template read t and related group G, we use a divide-then-combine strategy to cluster reads (**Supplementary Figure 15 b**). In the dividing step, we use a modified bisecting k-means algorithm[32] to divide the group G into small ones with following steps

1. The centroids C_1 and C_2 of the sets $\{r_1\}$ and $\{r_2\}$ are initially selected, where reads r_1 and r_2 are a read pair in the group G with the largest distance ($r_1, r_2 = \arg_{r_1, r_2 \in G} \max |D(r_1, r_2, 0, 0)|$).

2. We divide the group G into three subgroups, two subgroups corresponding to the centroids and a separate subgroup containing the reads far away from the centroids. For each read r, if it is far away from the centroids, it meets the conditions: $\frac{|A(r, C_i, 0, 0)|}{|V_{\neq}(r, 0, 0)|} < p_2 \; or \; \frac{|D(r, C_i, 0, 0)|}{|A(r, C_i, 0, 0)|} > p_3, i \in \{1,2\}$, where p_2 and p_3 are user-set parameters (default values are 0.3 and 0.5), it is assigned to a separate subgroup. Otherwise, it is assigned

the nearest subgroup $i = \arg_{i\in\{1,2\}} \min \frac{|D(r,C_i,0,0)|}{|A(r,C_i,0,0)|}$. The purpose of the separate subgroup is to prevent the centroids from changing dramatically in each iteration.

3. We calculate the centroids for the first two subgroups and repeat step 2 until the three subgroups don't change.

After the group is divided into three subgroups, we repeat the above steps to continue dividing the subgroups. If the subgroup G and its centroid C meet the following conditions: $|G| \leq 3\ or\ \frac{\sum_{r\in G}|D(r,C,0,0)|}{\sum_{r\in G}|A(r,C,0,0)|} \leq p_4\ and\ \sum_{r\in G}|D(r,C,0,0)| \leq p_5$, it is no longer divided into small subgroups. p_4 and p_5 are user-set parameters (the default values are 0.02 and 4).

After dividing the groups, we get a set of subgroups SG. we combine a pair of subgroups into a big one if the centroids are close to each other. First, we create an empty list L and add the subgroup that contains the template read to the list as the first element L_1. Then, we choose the subgroup g that is furthest away L_1, that means $g = \arg_{g\in SG\setminus\{L_1\}} \max|D(g,L_1,p_6,p_7)|$, where p_6 and p_7 are user-set parameters (the default values are 0.66 and 6). We add the subgroup g to the list L as the second element L_2. The remaining subgroups are put into another list L' and sort the subgroups in ascending order of distance with L_0 ($|D(L_0,L'_i,p_6,p_7)|$). For the subgroup i in the list L', we combine it to the subgroup j in list L, if it meets the following conditions: (1) $j = \arg\max|S(i,j,p_6,0)|, j \in L$; (2) $D(i,j,p_6,p_7) < \max(p_8, A(i,j,p_6,p_7)\cdot p_9)$; (3) $D(i,j,p_6,0) < A(i,j,p_6,0)\cdot p_9$, where p_8 and p_9 are another user-set parameters (the default values are 4 and 0.2). Otherwise, the subgroup i is added to the end of the list L.

After combining the subgroups, we think the reads in the same subgroup in the list L are from the same haplotype or the same copy of the repeat. The SNP alleles in the read can be verified by the centroid of its subgroup, which helps to find inconsistent overlaps more accurately. In addition, since the group L_1 contains the template read t, we use the centroid of the group L_1 to correct SNP alleles in the template read t. Here, the read is corrected when it is treated as a template read. After all reads are corrected, we run the verifying and correcting SNP alleles step again to obtain more robust results. We run the step three times by default.

Finding inconsistent overlaps

In the last round of verifying and correcting SNP alleles, we identify inconsistent overlaps. After combining steps, for each template read t, we obtain a subgroup list L and the first subgroup L_1 of L contains the template read t. We consider a query read r and the template read t are inconsistent, if the query read r and its subgroup L_r ($r \in L_r$) meet the conditions: (1) $|D(L_r, L_1, p_6, p_7)| \geq \max(p_8, |A(L_r, L_1, p_6, p_7)| \cdot p_9)$; (2) $|D(r, L_1, 0, 0) \cap A(L_r, L_1, p_6, p_7)| \geq \max(p_8, |A(r, L_1, 0, 0) \cap A(L_r, L_1, p_6, p_7)| \cdot p_9)$. The parameters p_6, p_7, p_8 and p_9 are mentioned above. The overlaps between the inconsistent read pairs are identified as inconsistent overlaps. We record the inconsistent overlap information and SNP alleles in reads for subsequent steps.

Fast string-graph-based assembler

According to the characteristics of k-mer-based alignment and string graphs, we propose a fast string-graph-based assembler to balance the quality and speed of assembling. First, we use minimap2 with parameters "-X -g3000 -w30 -k19 -m100 -

r500" to find candidate overlaps between corrected reads. Minimap2[20] with those parameters invokes the k-mer-based alignment. To reduce overhangs of overlaps, we extend the alignment to the ends of the reads with the diff[21] algorithm and filter out overlaps still with long overhangs. Next, we remove the overlaps whose reads are contained in other reads or with low coverage. A directed string graph is constructed using the remaining overlaps. Mayer's algorithm[18] is used to mark transitive edges as inactive ones. We implement local alignment using the edlib[31] algorithm to calculate the identity of each active edge, which is defined as the identity of its related overlap. Only a few edges need to calculate identities for most of the edges are marked as inactive ones. The edges whose identities are less than the threshold (default value is 0.95) are marked as low quality and removed. This step also breaks some paths connected by low-quality edges in the graphs. To repair those paths, we check dead-end nodes whose outdegree or indegree are equal to 0. We calculate the identities of their transitive edges and reactivate the edge with the longest alignment and the identity greater than the threshold. In this way, an appropriate string graph is constructed. After performing other simplifying processes, such as removing the ambiguous edges (tips, bubbles, and spurious links) in the graph, we identify linear paths from the graph and generate contigs.

Improvement of best overlap graph algorithm

Best overlap graph algorithm[33] only retains the best out-edge and the best in-edge of each node according to the overlap length. After removing transitive edges using Mayer's algorithm[18], PECAT performs the best overlap graph algorithm to further

simplify the string graph. However, the original algorithm is not suitable for diploid assembly. The SNP alleles in reads are more important than the overlap length to measure which edge is the best one. Although most inconsistent overlaps are removed, there remain some undetected inconsistent overlaps and the corresponding reads contain different SNP alleles. We improve the algorithm and use the following steps to determine the best edges. First, for the in-edges or out-edges of each node, we sort the edges in descending order of the edge score $s_i = (n_i^+ \cdot w - n_i^-, l_i)$. n_i^+ and n_i^- are the numbers of two related reads containing the same or different SNP alleles, which are obtained in finding inconsistent overlaps step. w is a user-defined parameter. Its default value is set to 0.5. l_i is the overlap length. The first edge e_0 is marked as a candidate best edge. The other edges meeting the following conditions also are marked as candidate best edges: (1) Its related read is inconsistent with the reads related to the candidate best edges. (2) Its score is not too much less than the first edge e_0, which means $s_0[0] - s_i[0] < \begin{cases} \max(C, s_i[0] \cdot R_1), s_i[0] \geq 0 \\ \max(C, s_i[0] \cdot R_2), s_i[0] < 0 \end{cases}$, where C, R_1 and R_2 are user-defined parameters (the default values are 4, 2, and 0.66). Next, if the edge is marked as a candidate best edge twice (in-edge and out-edge), it is selected as the best edge. Then, if the node has no best in-edge or best out-edge, its first in-edge or first out-edge is selected as the best one. Finally, the edges selected as the best edges are retained and the other edges are removed from the string graph.

Generating primary and alternate contigs

After simplifying the string graph, PECAT identifies three structures from the string graph, as shown in **Supplementary Figure 16**. First, we use FALCON-Unzip's

heuristic algorithm[13] to identify bubble structures. Generally, the paths in the bubble structure are from different haplotypes. Then, we also identify alternate branches, which meet the following conditions: (1) There are only two branches. (2) The shorter branch is linear and does not contain other branches. (3) More than 30% of reads in the shorter branch are inconsistent with the reads in the other branch. We think the two branches that meet the conditions are from different haplotypes. We identify the paths in the string graph and do not break the paths if they encounter bubble structures or alternate branches. This trick can increase the continuity of primary contigs. The other paths in bubble structures or the alternate branches output as alternate contigs. We also check pair of arbitrary independent contigs. If more than 30% of reads in the shorter contig are inconsistent with the reads in another contig, the shorter contig is labeled as an alternate contig. Another contig not labeled as an alternate contig is outputted as a primary contig.

Polishing primary and alternate contigs

After generating primary and alternate contigs, we use corrected reads or raw reads to polish them. We map the reads to the primary and alternate contigs using minimap2. In previous steps, we record which read pairs are inconsistent and which reads construct the contigs. If the read is inconsistent with the reads used for constructing the contig, the related alignments are removed. This trick helps to reduce haplotype switch errors. After removing other low-quality alignments, we run racon[34] with default parameters to polish the contigs.

Evaluation

To evaluate the effectiveness of correction methods fairly, we extract the 40X longest reads from corrected reads and then evaluate them. We map the reads to the reference genome using minimap2 with parameters "-c --eqx". It generates CIGAR strings. We scan the CIGAR strings to calculate the accuracy of each corrected data. To evaluate the haplotype-specific k-mers completeness, we calculate the percentage of parent-specific k-mers in 40X longest reads. Considering that there are 40X datasets, we only count k-mers with equal to or more than 4 occurrences. To evaluate the haplotype-specific k-mers consistency, we calculate the metric as $\sum \max(k_p, k_m)/\sum(k_p + k_m)$, in which k_p and k_m are the number of paternal and maternal haplotype-specific k-mers in each read.

To evaluate the performance of selecting supporting reads and the performance of finding inconsistent overlaps, we use Illumina reads of parents to classify long reads with a trio-binning algorithm[6]. We adjust the threshold to reduce false positives. The read is classified as the paternal read if it meets the condition: $\frac{k_p}{K_p} > \frac{k_m + \max(10, k_m \cdot 0.1)}{K_m}$, where k_p and k_m are the numbers of paternal and maternal haplotype-specific k-mers in the read, and K_p and K_m are the numbers of all paternal and maternal haplotype-specific k-mers. The read is classified as a maternal read if it meets the condition: $\frac{k_m}{K_m} > \frac{k_p + \max(10, k_p \cdot 0.1)}{K_p}$. Otherwise, it is classified to the untagged reads. When one read of the pair of reads is a maternal read and the other read is a paternal read or vice versa, we think the pair of reads is inconsistent.

We use k-mer based assembly evaluation tool merqury[29] to evaluate the diploid assemblies generated by each pipeline and use BUSCO[28] to evaluate the gene completeness of assemblies. The details of the parameters used in this study are described in **Supplementary Notes 4-5**.

Datasets

All described datasets are obtained from public websites, except for *A. thaliana* (COL × C24) which is generated using our in-house sequencing. It is available from https://ngdc.cncb.ac.cn/ at PRJCA011723. *S. cerevisiae* (SK × Y12), *A. thaliana* (COL × CVI), *D. melanogaster* (ISO1 × A4), *B. taurus* (Angus × Branman), and *B. taurus* (Bison×Simmental) are available from NCBI at PRJEB7245, PRJNA314706, PRJNA573706, PRJNA432857, and PRJNA677946. NCTC9006 and NCTC9024 are available at http://www.sanger.ac.uk/resources/downloads/bacteria/nctc/. The details of the datasets used in this study are reported in **Supplementary Note 1** and **Supplementary Table 6**. All assemblies are available at https://doi.org/10.5281/zenodo.6540735.

Accession codes

All source codes for PECAT are available from https://github.com/lemene/PECAT.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grants (Nos. 62150048 and U1909208), and the National Key Research and Development Program of China (No.2021YFF1201200). This work was also supported in part by the US National Institute of Food and Agriculture (NIFA; Grant Number

2017-70016-26051) and the US National Science Foundation (NSF; Grant Number ABI-1759856, MRI-2018069, MTM2-2025541) to Feng Luo. We are grateful for resources from the High-Performance Computing Center of Central South University.

Author contributions

J.X.W., F.L., and C.L.X. conceived and designed this project. F.N. and J.X.W. conceived, designed, and implemented the consensus and assembly algorithm. F.N. integrated all the programs into the PECAT pipeline and provided documentation. F.N, N.H, J.Z, and P.N analyzed the performance of algorithms developed in this study. C.L.X and Z.Y.W provided the dataset *A. thaliana* (COL × C24). J.X.W., F.L., and F.N. performed the theoretical analysis of the algorithms developed in this study. J.X.W, F.L., F.N., and Z.Y.W. wrote the manuscript. All authors have read and approved the final version of this manuscript.

References

1. Nagarajan, N. & Pop, M. Sequence assembly demystified. *Nat. Rev. Genet.* **14**, 157–167 (2013).
2. Gagarinova, A. & Emili, A. Genome-scale genetic manipulation methods for exploring bacterial molecular biology. *Mol. Biosyst.* **8**, 1626 (2012).
3. International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. *Nature* **431**, 931–945 (2004).
4. Weirather, J. L. *et al.* Comprehensive comparison of Pacific Biosciences and Oxford Nanopore Technologies and their applications to transcriptome analysis. *F1000Research* **6**, 100 (2017).
5. Wenger, A. M. *et al.* Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nat. Biotechnol.* **37**, 1155–1162 (2019).

6. Koren, S. *et al.* De novo assembly of haplotype-resolved genomes with trio binning. *Nat. Biotechnol.* **36**, 1174–1182 (2018).

7. Garg, S. *et al.* Chromosome-scale, haplotype-resolved assembly of human genomes. *Nat. Biotechnol.* (2020) doi:10.1038/s41587-020-0711-0.

8. Cheng, H. *et al.* Haplotype-resolved assembly of diploid genomes without parental data. *Nat. Biotechnol.* (2022) doi:10.1038/s41587-022-01261-x.

9. Human Genome Structural Variation Consortium *et al.* Fully phased human genome assembly without parental data using single-cell strand sequencing and long reads. *Nat. Biotechnol.* **39**, 302–308 (2021).

10. Campoy, J. A. *et al.* Gamete binning: chromosome-level and haplotype-resolved genome assembly enabled by high-throughput single-cell sequencing of gamete genomes. *Genome Biol.* **21**, 306 (2020).

11. Cheng, H., Concepcion, G. T., Feng, X., Zhang, H. & Li, H. Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm. *Nat. Methods* **18**, 170–175 (2021).

12. Nurk, S. *et al.* HiCanu: accurate assembly of segmental duplications, satellites, and allelic variants from high-fidelity long reads. *Genome Res.* **30**, 1291–1305 (2020).

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

14. Koren, S. *et al.* Canu: scalable and accurate long-read assembly via adaptive *k* -mer weighting and repeat separation. *Genome Res.* **27**, 722–736 (2017).

15. Xiao, C.-L. *et al.* MECAT: fast mapping, error correction, and de novo assembly for single-molecule sequencing reads. *Nat. Methods* **14**, 1072–1074 (2017).

16. Chen, Y. *et al.* Efficient assembly of nanopore reads via highly accurate and intact error correction. *Nat. Commun.* **12**, 60 (2021).

17. Li, H. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics* **32**, 2103–2110 (2016).

18. Myers, E. W. The fragment assembly string graph. *Bioinformatics* **21**, ii79–ii85 (2005).

19. Lee, C., Grasso, C. & Sharlow, M. F. Multiple sequence alignment using partial order graphs. *Bioinformatics* **18**, 452–464 (2002).

20. Li, H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* **34**, 3094–3100 (2018).

21. Myers, E. W. AnO(ND) difference algorithm and its variations. *Algorithmica* **1**, 251–266 (1986).

22. Kolmogorov, M., Yuan, J., Lin, Y. & Pevzner, P. A. Assembly of long, error-prone reads using repeat graphs. *Nat. Biotechnol.* **37**, 540–546 (2019).

23. Kamath, G. M., Shomorony, I., Xia, F., Courtade, T. A. & Tse, D. N. HINGE: long-read assembly achieves optimal repeat resolution. *Genome Res.* **27**, 747–756 (2017).

24. Guan, D. *et al.* Identifying and removing haplotypic duplication in primary genome assemblies. *Bioinformatics* **36**, 2896–2898 (2020).

25. Shafin, K. *et al.* Haplotype-aware variant calling with PEPPER-Margin-DeepVariant enables high accuracy in nanopore long-reads. *Nat. Methods* **18**, 1322–1332 (2021).

26. Shafin, K. *et al.* Nanopore sequencing and the Shasta toolkit enable efficient de novo assembly of eleven human genomes. *Nat. Biotechnol.* **38**, 1044–1053 (2020).

27. Oppenheimer, J. *et al.* A Reference Genome Assembly of American Bison, Bison bison bison. *J. Hered.* **112**, 10 (2021).

28. Manni, M., Berkeley, M. R., Seppey, M., Simão, F. A. & Zdobnov, E. M. BUSCO Update: Novel and Streamlined Workflows along with Broader and Deeper Phylogenetic Coverage for Scoring of Eukaryotic, Prokaryotic, and Viral Genomes. *Mol. Biol. Evol.* **38**, 4647–4654 (2021).

29. Rhie, A., Walenz, B. P., Koren, S. & Phillippy, A. M. Merqury: reference-free quality, completeness, and phasing assessment for genome assemblies. *Genome Biol.* **21**, 245 (2020).

30. Chin, C.-S. *et al.* Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nat. Methods* **10**, 563–569 (2013).

31. Šošić, M. & Šikić, M. Edlib: a C/C ++ library for fast, exact sequence alignment using edit distance. *Bioinformatics* **33**, 1394–1395 (2017).

32. Savaresi, S. M. & Boley, D. L. On the performance of bisecting K-means and PDDP. in *Proceedings of the 2001 SIAM International Conference on Data Mining* 1–14 (Society for Industrial and Applied Mathematics, 2001). doi:10.1137/1.9781611972719.5.

33. Miller, J. R. *et al.* Aggressive assembly of pyrosequencing reads with mates. *Bioinformatics* **24**, 2818–2824 (2008).

34. Vaser, R., Sović, I., Nagarajan, N. & Šikić, M. Fast and accurate de novo genome assembly from long uncorrected reads. *Genome Res.* **27**, 737–746 (2017).

Figure

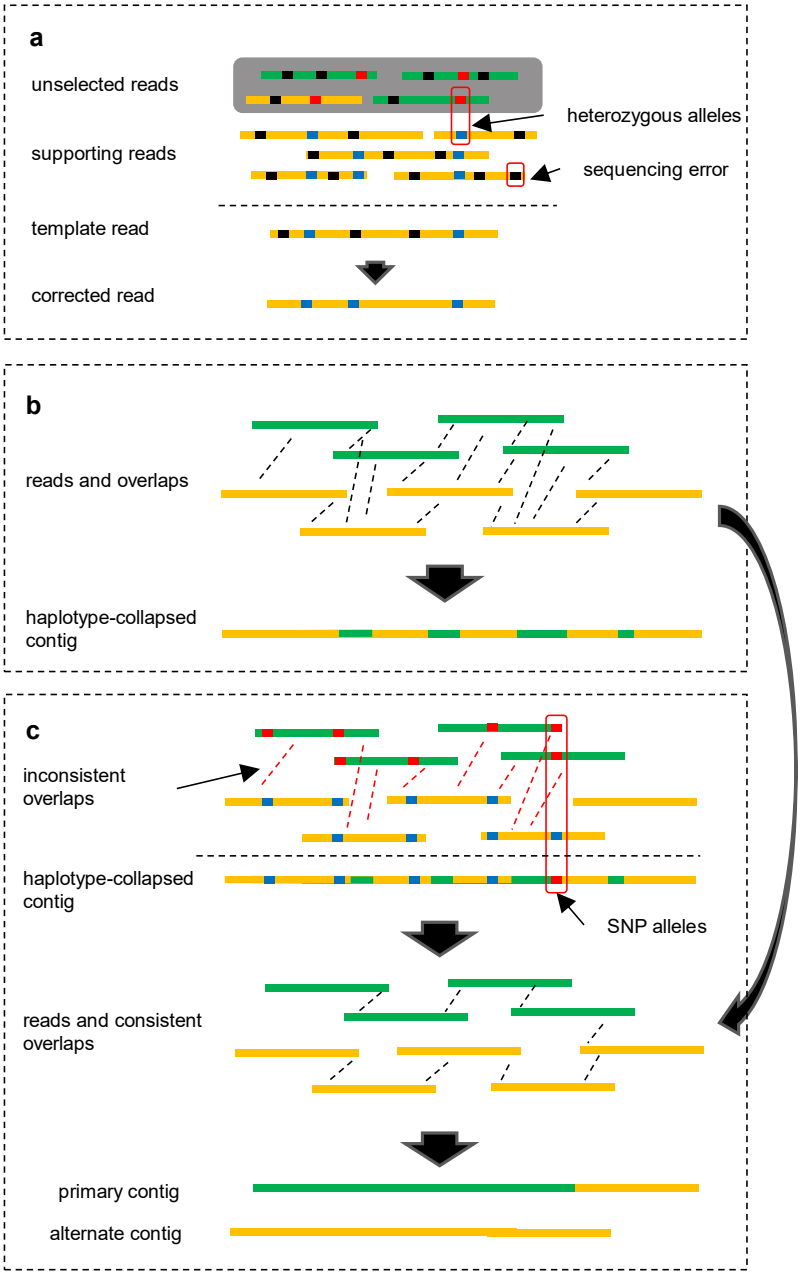

Figure 1. Overview of PECAT. (**a**) Haplotype-aware error correction method. The supporting reads that are more likely from the same haplotype are selected to correct the template read. (**b**) The first round of assembly. PECAT finds overlaps between reads. The dashed lines indicate there are overlaps between reads. Our string-graph-based assembler is performed to construct haplotype-collapsed contigs. (c) The second round of assembly. PECAT identifies inconsistent overlaps by calling and analyzing SNP alleles in reads. After removing inconsistent overlaps from the overlaps found in the first round of assembly, PECAT performs our string-graph-based assembler again to construct primary/alternate contigs.

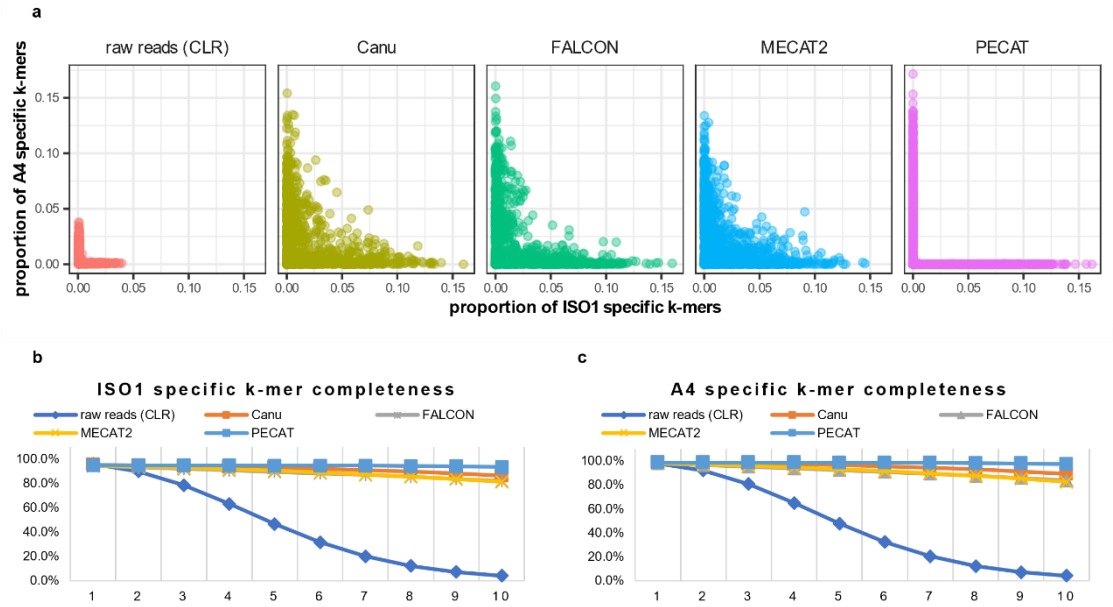

Figure 2. Haplotype-specific k-mers consistency and completeness of *D. melanogaster* (ISO1 × A4) raw reads and corrected reads by the different methods. (**a**) Each point corresponds to a read. Its coordinate gives proportion of the parental specific k-mers in the read. There are 1000 randomly selected reads from 40X longest reads for each sub figure. (**b**, **c**) The percentages of the parental specific k-mers whose occurrences are greater or equal to 1-10 in 40X longest reads.

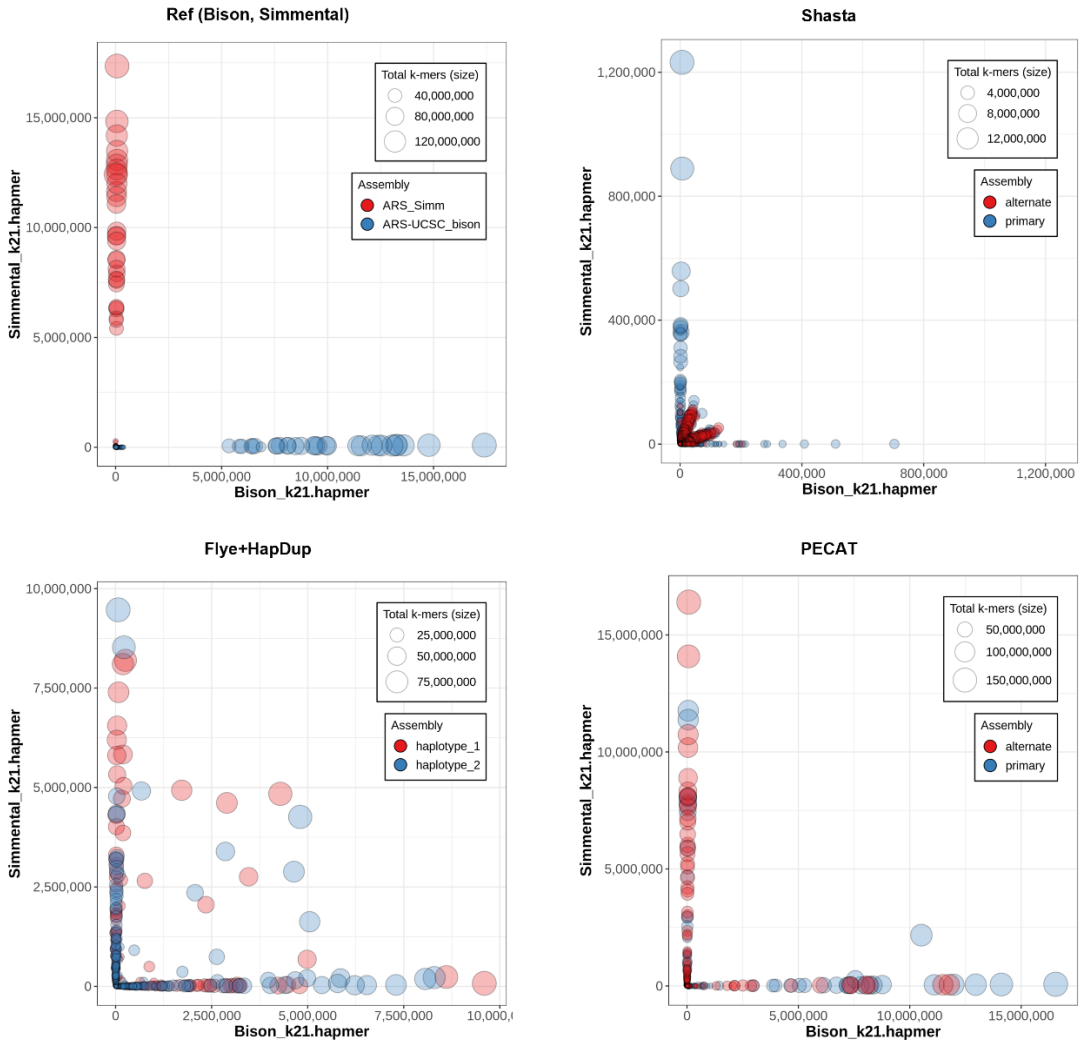

Figure 3. Haplotype-specific k-mer blob plots of the *B. taurus* (Bison × Simmental) reference genome and assemblies. Figures are generated by merqury. Each blob corresponds to a contig. Its coordinate gives count of the parental specific k-mers in the contig. Blob size is proportional to contig length.

Table

Table 1. Performance comparison of error correction

Sample	Pipeline	Accuracy (%)	Max	N50	Consistency (%)	Completeness PAT (%) / MAT (%)	Size(g)/Time(h)/Speed(g/h)
S. cerevisiae (SK1 × Y12, CLR, 200X)	raw data	87.80	34893	13988	93.4	37.7 / 36.6	2.4 / - / -
	Canu	99.53	30092	11603	89.8	94.3 / 95.9	1.1 / 69.9 / 0.016
	FALCON	99.69	28882	10400	88.5	73.3 / 82.2	1.0 / - / -
	MECAT2	99.37	30252	11734	87.4	90.5 / 88.0	1.7 / 5.7 / 0.300
	PECAT	99.76	30145	11785	99.7	98.7 / 99.2	0.7 / 8.5 / 0.088
A. thaliana (COL × CVI, CLR, 164X)	raw data	88.24	88007	26088	82.5	42.6 / 42.7	21.4 / - / -
	Canu	99.31	67179	23105	94.6	97.6 / 97.6	12.4 / 711.6 / 0.017
	FALCON	99.39	67102	20530	92.7	89.8 / 89.7	13.2 / - / -
	MECAT2	99.27	55951	23285	94.2	96.5 / 96.6	17.9 / 134.1 / 0.133
	PECAT	99.52	48646	23414	99.7	98.8 / 98.8	8.8 / 107.8 / 0.082
D. melanogaster (ISO1 × A4, CLR, 200X)	raw data	89.58	129316	60067	93.6	63.1 / 64.8	30.0 / - / -
	Canu	99.45	119938	54896	91.4	93.5 / 97.2	13.3 / 1497.1 / 0.009
	FALCON	98.91	114434	37210	93.4	90.7 / 93.8	21.3 / - / -
	MECAT2	99.01	120475	47894	90.0	91.5 / 94.6	23.1 / 222.8 / 0.104
	PECAT	99.66	121182	52036	99.9	94.7 / 98.6	9.6 / 115.9 / 0.083
B. taurus (Angus × Brahman, CLR, 135X)	raw data	86.19	172752	26938	87.7	12.9 / 12.1	366.3 / - / -
	MECAT2	98.90	69942	21517	85.3	88.5 / 88.9	232.0 / 5578.1 / 0.042
	PECAT	99.38	77599	22716	99.6	96.9 / 97.2	173.9 / 2822.6 / 0.062
A. thaliana (COL × C24, ONT, 106X)	raw data	92.33	123962	30858	91.9	91.0 / 90.5	13.9 / - / -
	Canu	98.64	108192	30092	87.5	90.7 / 91.1	11.2 / 725.9 / 0.015
	NECAT	98.95	109986	29551	91.8	89.5 / 89.2	9.3 / 75.4 / 0.123
	PECAT	99.20	80638	27577	98.4	95.7 / 96.4	8.6 / 171.5 / 0.050
B. taurus (Bison × Simmental, ONT, 200X)	raw data	88.96	485079	83396	93.7	69.4 / 69.4	541.7 / - / -
	NECAT	98.61	365006	82988	85.7	80.2 / 80.9	217.0 / 13524.5 / 0.016
	PECAT	98.79	292275	80029	99.3	92.2 / 92.9	173.9 / 6285.3 / 0.028

'Accuracy' is the percentage of matched base pairs in the alignments between the reads and the reference genome. 'Max' is the max length of the reads. 'N50' is the length of the shortest read for which longer and equal length reads cover at least 50% of the 40X longest reads. 'Consistency' is defined as $\sum \max(k_p, k_m)/\sum(k_p + k_m)$, in which k_p and k_m are the number of paternal and maternal haplotype-specific k-mers in each read. 'Completeness' is the percentage of parents specific k-mers (occurrences ≥ 4) in the 40X longest reads. 'PAT' and 'MAT' are paternal Completeness and maternal Completeness respectively. 'Size' is the total number of base pairs of the reads. 'Time' is the running time of the error correction method, and the 'Speed' is defined as 'Size'/'Time'. All the pipelines are tested on the same computer with 2.0 GHz CPU and 1T GB RAM of memory and run with 48 threads.

Table 2. Performance comparison of assembly

Dataset	Pipeline	Size (Mb)	Count	MAX (Mb)	NG50 (Mb)	Quality (QV)	BUSCO (%)	Haploblock NG50 (Mb)	Intra-block switch error (%)	Time (h)
S. cerevisiae SK1 × Y12 CLR, 200X	Ref (SK1, Y12)	12.1	17	1.49	0.92	47.30	99.6	0.92	0.13	-
		12.0	17	1.50	0.90	49.38	99.6	0.90	0.03	
	Canu + purge_dups	12.4	37	1.48	0.84	38.89	98.7	0.03	9.92	109
		4.8	171	0.10	0.00	36.22	28.1	0.00	4.50	
	FALCON-Unzip	12.1	21	1.47	0.82	44.48	99.4	0.48	0.43	199
		11.2	57	1.02	0.44	44.89	95.8	0.44	0.21	
	PECAT	12.3	24	1.51	0.81	38.78	95.1	0.80	0.15	12
		11.8	28	1.48	0.80	38.63	94.3	0.80	0.11	
A. thaliana COL × CVI CLR, 200X	Ref (COL, CVI)	133.3	5	32.64	26.15	44.87	99.3	22.92	0.10	-
		119.7	102	30.22	23.20	44.61	99.2	20.04	0.04	
	Canu + purge_dups	129.1	246	13.58	6.70	28.80	98.7	0.15	2.84	1456
		122.6	1371	1.19	0.10	34.21	81.7	0.09	2.50	
	FALCON-Unzip*	140.0	172	13.32	7.98	34.82	98.9	4.37	0.13	2652
		104.9	248	11.65	4.26	38.25	93.6	2.47	0.15	
	PECAT	133.7	417	16.63	11.21	28.35	98.3	9.66	0.14	161
		118.8	109	16.12	8.75	36.51	98.0	6.51	0.16	
D. melanogaster ISO1 × A4 CLR, 200X	Ref (ISO1, A4)	143.7	1870	32.08	25.29	46.54	98.7	25.29	0.06	-
		140.7	96	32.73	25.03	46.32	98.7	24.57	0.02	
	Canu + purge_dups*	143.2	134	31.12	16.11	35.39	98.5	0.42	3.92	3006
		129.7	565	1.59	0.31	35.29	87.3	0.27	3.01	
	FALCON-Unzip	189.7	293	10.60	4.02	37.09	98.8	1.01	0.34	3528
		105.9	486	1.77	0.35	37.36	84.0	0.28	0.41	
	PECAT	155.6	278	28.19	21.71	40.63	98.8	14.69	0.05	163
		132.1	115	25.82	11.85	45.89	97.2	11.78	0.03	
B. taurus Angus × Brahman CLR, 135X	Ref (Angus, Brahman)	2580.8	1435	157.01	91.13	43.33	93.6	21.82	0.08	-
		2681.0	1251	156.94	104.47	43.81	95.6	30.51	0.03	
	FALCON-Unzip*	2713.4	1427	65.27	31.39	39.44	95.4	3.21	0.21	-
		2453.7	5879	14.94	2.03	39.04	86.3	1.80	0.22	
	PECAT	2767.4	3336	150.13	65.94	41.20	95.3	3.30	0.18	3683
		2421.3	4167	13.12	2.00	41.96	85.8	1.74	0.20	
A. thaliana COL × C24 ONT, 106X	Ref (COL, C24)	133.3	5	32.64	26.15	100.00	99.3	26.15	0.00	-
		119.2	98	29.41	23.84	100.00	99.1	23.84	0.00	
	Canu + purge_dups	125.9	94	15.58	8.06	28.80	96.6	0.08	8.78	8359
		68.3	778	1.02	0.05	27.33	37.4	0.01	6.30	
	Flye + HapDup	136.4	851	9.29	2.12	27.19	97.8	0.49	1.16	202
		136.5	857	9.26	2.12	27.01	97.8	0.44	1.10	
	Shasta	126.3	1128	5.35	0.83	27.80	94.9	0.27	7.26	9

		105.0	340	2.71	0.52	29.98	92.2	0.24	5.73	
	PECAT	133.3	189	16.08	12.43	28.26	97.2	3.34	0.95	241
		116.0	104	12.32	3.08	30.12	94.4	2.35	0.56	
B. taurus Bison × Simmental ONT, 135X	Ref (Bison, Simmental)	2651.6	774	157.90	87.75	38.74	93.1	87.75	0.74	-
		2861.7	1314	156.88	104.40	36.12	95.7	104.38	0.61	
	Flye + HapDup	2713.0	1814	93.28	33.36	28.46	83.4	8.24	0.44	8159
		2713.2	1802	93.14	33.41	28.44	83.3	8.06	0.44	
	Shasta	2815.7	99142	15.93	0.26	22.93	61.3	0.12	23.26	519
		2406.9	19531	1.90	0.22	23.99	59.1	0.10	25.79	
	PECAT	2892.8	1437	156.02	73.03	26.33	77.1	72.34	0.54	8337
		2730.9	637	155.93	68.13	26.35	76.2	68.13	0.50	

'Size' is the total number of base pairs in all contigs generated by assemblers. 'Count' is the total number of contigs in assembly. 'Max' is the max length of the contigs. 'NG50' is the length of the shortest contig for which longer and equal length contigs cover at least 50% of genome size. The genome sizes of *S. cerevisiae*, *A. thaliana*, *D. melanogaster*, and *B. taurus* that we used for evaluation are 12M, 130M, 140M, and 2.7G, respectively. 'BUSCO' is gene completeness evaluated by BUSCO. 'Quality', 'Haploblock NG50' and 'Intra-block switch error' are evaluated by mercury. 'Time' is the running time of assembling. The primary and alternate contigs are separately reported on the top and the bottom of each row. All the pipelines are tested on the same computer with 2.0 GHz CPU and 1T GB RAM of memory and run with 48 threads. 'Ref' is the reference genome. The sources of the reference genomes are illustrated in Supplementary Table 6. Asterisks mark previously published assemblies.