Fast and accurate long-read assembly with wtdbg2

Jue Ruan [1,2,†] and Heng Li [3–5,†]

[1] Agricultural Genomics Institute, Chinese Academy of Agriculture Sciences, Shenzhen, China
[2] Peng Cheng Laboratory, Shenzhen, China
[3] Department of data sciences, Dana-Farber Cancer Institute, Boston, MA 02215, USA
[4] Department of biomedical informatics, Harvard Medical School, Boston, MA 02115, USA
[5] Broad Institute, Cambridge, MA 02142, USA

[†] To whom correspondence should be addressed. Email: ruanjue@caas.cn and hli@jimmy.harvard.edu.

Existing long-read assemblers require tens of thousands of CPU hours to assemble a human genome and are being outpaced by sequencing technologies in terms of both throughput and cost. We developed a novel long-read assembler wtdbg2 that, for human data, is tens of times faster than published tools while achieving comparable contiguity and accuracy. It represents a significant algorithmic advance and paves the way for population-scale long-read assembly in future.

De novo sequence assembly reconstructs a sample genome from relatively short sequence reads. It is essential to the study of new species and structural genomic changes that often fail mapping-based analysis as the reference genome may lack the regions of interest. With the rapid advances in single-molecule sequencing technologies by Pacific Biosciences (PacBio) and Oxford Nanopore Technologies (ONT), we are able to sequence reads of 10–100 kilobases (kb) at affordable cost. Such long reads resolve major repeat classes in primates and help to improve the contiguity of assemblies. Nowadays, long-read assembly has become a routine for bacteria and small genomes, thanks to the development of several high-quality assemblers[1-5]. For mammalian genomes, however, existing assemblers may take several CPU years, or ~1000 US dollars if we use the Google or the Amazon cloud for computing. This price is more expensive than data produced with one PromethION machine, which is capable of sequencing a human genome at 30-fold coverage in two days[6]. To address this issue, we developed wtdbg2, a new long-read assembler that is tens of times faster for large genomes with little compromise on the assembly quality.

Wtdbg2 broadly follows the overlap-layout-consensus paradigm. It advances the existing assemblers with a fast all-vs-all read alignment implementation and a novel layout algorithm based on fuzzy-Bruijn graph (FBG).

For mammalian genomes, current read overlappers[7-9] split input reads into many smaller batches and perform all-vs-all alignment between batches. This strategy wastes compute time on repeated file I/O and on indexing and querying non-informative *k*-mers. These overlappers do not build a single hash table in worry that it may take too much memory. Interestingly, this should not be a major concern. Wtdbg2 first loads all reads into memory and counts *k*-mer occurrences. It then bins every 256bp on reads into one unit and builds a hash table with keys being *k*-mers occurring twice or more, and values being identifiers of associated bins. For the CHM1 human genome sequenced to 60-fold coverage[10], for example, there are only 1.5 billion non-unqiue *k*-mers with the default settings. Staging raw read sequences in memory and constructing the hash table takes 250GB at the peak, which is comparable to the memory usage of short-read assemblers.

Wtdbg2 bins read sequences to speed up the next step in alignment: dynamic programming (DP). With 256bp binning, the DP matrix is 65536 (=256×256) times smaller than a per-base DP matrix. This reduces DP to a much smaller scale in comparison to *k*-mer based[8,9] or base-level DP[7].

FBG extends the basic ideas behind de Bruijn graph (DBG) to work with long noisy reads. In analogy to DBG, a "base" in FBG is a 256bp bin and a "*K*-mer" or *K*-bin in FBG consists of *K* consecutive bins on reads. A vertex in FBG is a *K*-bin and an edge between two vertices indicates their adjacency on a read. Unlike DBG, different *K*-bins may be represented by a single vertex if they are aligned together based on all-vs-all read alignment. This treatment tolerates errors in noisy long reads. FBG is closer to sparse DBG[11] than standard DBG in that it does not inspect every *K*-bin on reads. The sparsity reduces the memory to construct FBG. Furthermore, FBG explicitly keeps identifiers of bins going through each edge to retain long-range information without a separate "read threading" step as with standard DBG assembly. After graph simplification, wtdbg2 writes the final FBG to disk with read sequences on edges contained in the file. Wtdbg2 constructs the final consensus with partial order alignment[12] over edge sequences.

We evaluated wtdbg2 v2.3 on five datasets along with CANU-1.8[4], FALCON-180831[1], Flye-2.3.6[5] and MECAT-180314[3] (Table 1). We used minimap2 to align assembled contigs to the reference genome and to collect metrics. On all datasets, wtdbg2 is at least 4 times as fast as the closest competitors. Wtdbg2 assemblies sometimes cover less reference genomes, which is a weakness of wtdbg2, but its contigs tend to have fewer duplicates. The fraction of genomic regions covered by one contig is only a couple of percent lower in comparison to other assemblers.

For samples close to the reference genome, we also compared the consensus accuracy before and after signal-based polishing[13] when applicable. Without polishing, CANU, Flye and MECAT tend to produce better consensus sequences. This is probably because they perform at least two rounds of error correction or the consensus step, while wtdbg2 applies one round of consensus only. After Quiver polishing, the consensus accuracy of all assemblers is very close and significantly higher than the accuracy of consensus without polishing. This observation reconfirms that polishing consensus is still necessary[14] and suggests that the pre-polishing consensus accuracy is not obviously correlated with post-polishing accuracy.

We assembled five additional large datasets with wtdbg2 (Table 2). For all human data, wtdbg2 finishes the assembly in a few days on a single computer. This performance broadly matches the throughput of a PromethION machine. In comparison, Flye and CANU reportedly required ~10,000 and ~40,000 CPU hours, respectively, to assemble NA12878[15]. The peak memory used by wtdbg2 is generally below 230GB, except for HG00733 which was sequenced to a much higher coverage. Wtdbg2 is able to assemble axolotl, the largest genome we have sequenced so far, at a speed 30 times faster than the published assembly[16] and with a longer NG50.

Ten years ago when the Illumina sequencing technology entered the market, the sheer volume of data effectively decommissioned all aligners and assemblers developed earlier. History repeats itself. Affordable population-scale long-read sequencing is on the horizon. Wtdbg2 is the only assembler that is able to keep up with the throughput and the cost. With heterozygote-aware consensus algorithms and phased assembly planned for future, wtdbg2 and upcoming tools might fundamentally change the current practices on sequence data analysis.

Acknowledgements. We would thank Chengxi Ye from University of Maryland for frequent and fruitful discussion in the development of wtdbg. This study was supported by Natural Science Foundation of China (NSFC; grant 31571353 and 31822029 to J.R.) and by US National Institutes Health (NIH; grant R01-HG010040 to H.L.).

Author contribution. J.R. conceived the project, designed the algorithm and implemented wtdbg2. H.L. contributed to the development and drafted the manuscript. Both authors evaluated the results and revised the manuscript.

Competing interests. The authors declare no competing interests.

Online methods

Evaluation datasets. *C. elegans* and *A. thaliana* reads are provided by PacBio at http://bit.ly/pbpubdat. We downloaded SRR5439404 for the *D. melanogaster* A4 strain, SRR6702603 for the *D. melanogaster* reference ISO1 strain, PRJNA378970 for axolotl, SRR7615963 for HG00733, and ERR2631600 and ERR2631601 for NA19240. CHM1 reads were acquired from http://bit.ly/chm1p6c4, NA12878 reads from http://bit.ly/na12878ont (release 5) and NA24385 from http://bit.ly/NA24385ccs.

Wtdbg2 algorithm. Wtdbg2 reads all input sequences into memory and encodes each base with 2 bits. By default, it selects a quarter of *k*-mers based on their hash code and counts their occurrences using a hash table with 46-bit key to store a *k*-mer and 17-bit value to store its count. Wtdbg2 filters out *k*-mer occurring once or over 1000 times in reads, and then scans reads again to build a hash table for the remaining *k*-mers and their positions in bins.

For all-vs-all read alignment, wtdbg2 traverses each read, from the longest to the shortest, and uses the hash table to retrieve the reads that share *k*-mers with the read in query. It applies Smith-Waterman-like DP between binned sequences, penalizing gaps and bins that do not share *k*-mers. Wtdbg2 retains alignments no shorter than 8×256bp. After finishing alignments for all reads, wtdbg2 frees the hash table but keeps the all-vs-all alignments in memory (alignments are also written to disk as intermediate results).

At this step, wtdbg2 loses base sequences. It only sees binned sequences and the alignments between them. On a binned sequence $B = b_1 b_2 \cdots b_{|B|}$, a K-bin $b_{Ki} = b_i b_{i+1} \cdots b_{i+K-1}$ is a K-long subsequence starting at i-th position on B. If binned sequences B and B' can be aligned, we can infer the overlap length between K-bins b_{Ki} and $b'_{Ki'}$ by lifting their coordinates between the two sequences based on the alignment. We say two K-bins are equivalent if they completely overlap. Using the all-vs-all alignment, wtdbg2 collects a non-redundant set Ω of K-bins such that no K-bin in Ω is equivalent to others. Wtdbg2 also records the locations and coverage of each K-bin in Ω.

The vertex set V of FBG is a subset of Ω. To construct V, wtdbg2 traverses each non-redundant K-bin in the descending order of their original coverage. Given a K-bin b_K, wtdbg2 may reduce its coverage by deducting K-bins in V that overlap with b_K. If the adjusted coverage is ≥3 and higher than half of the original coverage, b_K will be added to V; otherwise it will be ignored. After the construction of V, wtdbg2 adds an edge between two K-bins if they are located on the same read. There are often multiple edges between two K-bins. Wtdbg2 retains one edge and keeps the count. An edge covered by <3 reads are discarded. This generates FBG. The coverage thresholds can be adjusted on the wtdbg2 command line.

Assembling evaluation datasets. With wtdbg2, we specified the genome size and sequence technology on the command line, which automatically applies multiple options. Specifically, we used "-xrs -g100m" for *C. elegans*, "-xsq -g125m" for *A. thaliana*, "-xrs -g144m" for *D. melanogaster* A4 strain, "-xont -g144m" for the ISO1 strain, "-xrs -g3g" for CHM1, "-xont -g3g" for human NA12878 and NA19240 ONT reads, "-xsq -g3g" for HG00733, "-xccs -g3g" for NA24385 and "-xrs -g3g" for the axolotl dataset. We used the similar settings for CANU, Flye and MECAT, specifying only the genome size and the sequencing technology. The FALCON configure file for assembling *C. elegans* is provided as **supplementary data**. The FALCON *A. thaliana* assembly was downloaded at http://bit.ly/pbpubdat. We are using AC:GCA_000983455.1 for the CANU CHM1 assembly and AC:GCA_001297185.1 for the FALCON CHM1 assembly.

Evaluating assemblies. To count alignment breakpoints, we mapped all assemblies to the corresponding reference genomes with minimap2 under the option "--paf-no-hit -cxasm20 -r2k -z1000,500". We used the companion script paftools.js to collect various metrics (command line: "paftools.js asmstat -q50000 -d.1"). To count substitutions and gaps, we applied a different minimap2 setting "-cxasm5 --cs -r2k". This setting introduces more alignment breakpoints but avoids poorly aligned regions harboring spuriously high number of differences that are likely caused by large-scale variations and skew the counts. We used "paftools.js call" to call variations.

Code availability. The wtdbg2 source code is hosted by GitHub at: https://github.com/ruanjue/wtdbg2.

Data availability. All the evaluated assemblies (except those publicly available in GenBank) can be obtained at ftp://ftp.dfci.harvard.edu/pub/hli/wtdbg/. The FTP site also provides the detailed command lines and configuration files.

Figure/Table Legends

Fig. 1 The wtdbg2 algorithm. Wtdbg2 groups 256 base pairs into a bin, a small box in the figure. Bins/boxes with the same color suggest they share *k*-mers, except that a gray bin doesn't match other bins due to sequencing errors. Wtdbg2 performs all-vs-all alignment between binned reads, ignoring detailed base sequences. In the fuzzy-Bruijn assembly graph, a vertex is a 4-bin segment. Wtdbg2 identifies equivalent segments based on their alignments and connects these segments if they are connected on the input reads (see Online Methods for details). Information on reads going through each edge is kept in the graph (colored edges). After graph construction, wtdbg2 trims tips

and pops bubbles and produces the final contig sequences from the consensus of read subsequences attached to each edge.

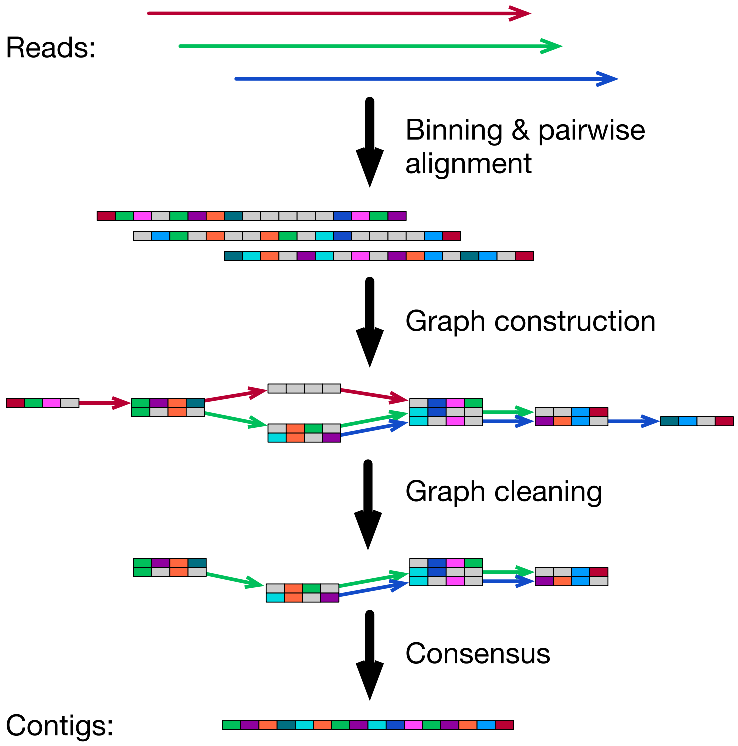

Table 1. Evaluating long-read assemblies

FALCON requires PacBio-style read names and does not work with ONT data or the A4 strain of *D. melanogaster* which was downloaded from SRA. The *A. thaliana* assembly by FALCON is acquired from PacBio website as our assembly is fragmented. MECAT produces fragmented assemblies for the ONT dataset. Human assemblies were performed by the developers of each assembler.

Dataset	Metric	CANU	FALCON	Flye	MECAT	Wtdbg2
C. elegans	Total length (>= 50kbp)	106.5Mb	100.8Mb	102.0Mb	102.1Mb	105.9Mb
Bristo ref. strain	% reference genome covered	99.58	99.16	99.29	99.51	99.32
PacBio x80	% genome covered more than once	0.33	0.25	0.15	0.35	0.15
	NG75	1,884,280	935,802	1,275,590	1,424,674	2,061,635
	NG50	2,677,990	1,629,544	1,926,198	2,113,456	3,405,054
	# alignment breakpoints	681	192	284	278	260
	# substitutions/1Mb (pre-/post-polish)	64.1 / 62.2	233.2 / 50.1	61.6 / 57.6	65.9 / 62.8	173.4 / 60.0
	# insertions/1Mb (pre-/post-polish)	31.1 / 22.4	592.7 / 19.4	29.8 / 21.8	43.9 / 21.9	245.1 / 21.4
	# deletions/1Mb (pre-/post-polish)	152.8 / 55.1	1822.7 / 56.7	381.4 / 56.9	366.0 / 57.9	734.5 / 59.7
	Wall-clock time over 32 CPUs	9h30m	2h06m	2h58m	3h08m	14m
A. thaliana	Total length (>= 50kbp)	118.3Mb	119.3Mb	116.4Mb	117.9Mb	117.6Mb
Ler-0 strain	% reference genome covered	90.53	91.07	90.62	90.91	90.5
PacBio x75	% genome covered more than once	0.71	0.67	0.39	0.7	0.33
	NG75	589,667	6,083,367	3,857,946	1,339,557	6,133,678
	NG50	1,098,921	10,401,798	6,726,569	4,070,235	9,067,948
	# alignment breakpoints	2,544	2,712	2,366	2,486	2,414
	Wall-clock time over 32 CPUs	15h50m		3h33m	4h31m	50m
D. melanogaster	Total length (>= 50kbp)	138.5Mb		131.2Mb	134.8Mb	133.9Mb
A4 strain	% genome covered	91.09		89.69	90.26	89.6

PacBio x120	% genome covered more than once	1.23		0.34	1.2	0.47
	NG75	8,173,560		2,198,937	5,557,762	4,238,783
	NG50	20,301,392		9,318,110	16,078,851	15,859,226
	# alignment breakpoints	1,617		1,175	1,301	1,128
	Wall-clock time over 32 CPUs	16h35m		5h13m	5h03m	26m
D. melanogaster ISO1 ref. strain ONT x32	Total length (>= 50kbp)	135.0Mb		130.7Mb		124.9Mb
	% reference genome covered	91.74		89.40		85.87
	% genome covered more than once	1.19		0.14		0.22
	NG75	714,013		1,367,004		1,687,783
	NG50	4,298,595		6,016,667		7,392,386
	# alignment breakpoints	823		248		372
	# substitutions per 1Mb (pre-polish)	847.6		1318		3020.4
	# insertions per 1Mb (pre-polish)	255.9		10669.9		813.8
	# deletions per 1Mb (pre-polish)	7168.2		1901.3		12004.8
	Wall-clock time over 32 CPUs	22h23m		1h41m		26m
Human CHM1 cell line PacBio 100x	Total length (>= 50kbp)	2,837Mb	2,938Mb			2,773Mb
	% reference genome covered	89.33	90.13			87.44
	% genome covered more than once	0.53	0.72			0.03
	NG75	3,793,440	7,726,658			4,166,845
	NG50	17,570,750	26,132,317			16,663,584
	# alignment breakpoints	5,440	9,203			4,887
	# substitutions per 1Mb (post-polish)	961.5	966.6			959
	# insertions per 1Mb (post-polish)	143	140.1			139.2
	# deletions per 1Mb (post-polish)	140	137.6			141.2
	CPU time	22,750h	68,789h			245h

Table 2. Wtdbg2 performance on large genomes

Data set	Technology	G. size	Cov.	CPU hour	Real hour	NG50
Human NA12878	Nanopore	3Gb	36	891	31	11.7Mb
Human NA19240	Nanopore	3Gb	35	1025	35	4.3Mb
Human NA24385	PacBio CCS	3Gb	28	290	10	12.9Mb
Human HG00733	PacBio Sequel	3Gb	93	2267	52	25.9Mb
Axolotl	PacBio RSII	32Gb	32	4262	110	392kb

Reference

1 Chin, C. S. *et al.* Phased diploid genome assembly with single-molecule real-time sequencing. *Nat Methods* **13**, 1050-1054, doi:10.1038/nmeth.4035 (2016).

2 Li, H. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics* **32**, 2103-2110, doi:10.1093/bioinformatics/btw152 (2016).

3 Xiao, C.-L. *et al.* MECAT: an ultra-fast mapping, error correction and de novo assembly tool for single-molecule sequencing reads. *bioRxiv*, doi:10.1101/089250 (2016).

4 Koren, S. *et al.* Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Res* **27**, 722-736, doi:10.1101/gr.215087.116 (2017).

5 Kolmogorov, M., Yuan, J., Lin, Y. & Pevzner, P. Assembly of Long Error-Prone Reads Using Repeat Graphs. *bioRxiv*, doi:10.1101/247148 (2018).

6 De Coster, W. *et al.* Structural variants identified by Oxford Nanopore PromethION sequencing of the human genome. *bioRxiv*, doi:10.1101/434118 (2018).

7 Myers, G. in *WABI.* (eds Daniel G. Brown & Burkhard Morgenstern) 52-67 (Springer).

8 Berlin, K. *et al.* Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nat Biotechnol* **33**, 623-630, doi:10.1038/nbt.3238 (2015).

9 Li, H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* **34**, 3094-3100, doi:10.1093/bioinformatics/bty191 (2018).

10 Chaisson, M. J., Wilson, R. K. & Eichler, E. E. Genetic variation and the de novo assembly of human genomes. *Nat Rev Genet* **16**, 627-640, doi:10.1038/nrg3933 (2015).

11 Ye, C., Ma, Z. S., Cannon, C. H., Pop, M. & Yu, D. W. Exploiting sparseness in de novo genome assembly. *BMC Bioinformatics* **13 Suppl 6**, S1, doi:10.1186/1471-2105-13-S6-S1 (2012).

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

13 Chin, C. S. *et al.* Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nat Methods* **10**, 563-569, doi:10.1038/nmeth.2474 (2013).

14 Watson, M. & Warr, A. Errors in long-read assemblies can critically affect protein prediction. *Nat Biotech* (2019).

15 Jain, M. *et al.* Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nat Biotechnol* **36**, 338-345, doi:10.1038/nbt.4060 (2018).

16 Nowoshilow, S. *et al.* The axolotl genome and the evolution of key tissue formation regulators. *Nature* **554**, 50-55, doi:10.1038/nature25458 (2018).