Raven: a de novo genome assembler for long reads

Robert Vaser[1] and Mile Šikić[1,2]

[1] Laboratory for Bioinformatics and Computational Biology, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, Croatia

[2] Laboratory of AI in Genomics, Genome Institute of Singapore, A*STAR, Singapore

We present new methods for the improvement of long-read de novo genome assembly incorporated into a straightforward tool called Raven (https://github.com/lbcb-sci/raven). Compared with other assemblers, Raven is one of two fastest, it reconstructs the sequenced genome in the least amount of fragments, has better or comparable accuracy, and maintains similar performance for various genomes. Raven takes 500 CPU hours to assemble a 44x human genome dataset in only 259 fragments.

Sequencing technologies have come a long way, from tiny fragments at their infancy to large chunks obtainable today. The relentless advances in both length and accuracy continue to alleviate the puzzle-like reconstruction problem of the sequenced genome, as more repetitive structures can be resolved naturally. Amidst the excess of available state-of-the-art options for de novo genome assembly[1–6], we present a fast, reliable, and easy to use tool called Raven. It is an overlap-layout-consensus based assembler which accelerates overlap step, builds an assembly graph from reads pre-processed with pile-o-grams, implements a novel, robust simplification method based on graph drawings, and polishes the reconstructed contigs Racon[7], all of which is compiled into a single executable.

Computational cost is often unduly neglected in the price of de novo assembly of large genomes. Ten thousand of CPU hours and high memory requirements can significantly increase the overall cost. Furthermore, assemblers are usually optimized to available model organisms such as the human. Keeping that in mind we have aimed to develop a fast and memory frugal tool, looking for methods that would be agnostic to genomes of specific organisms.

Short substring matching is a conventional approach for similarity search in bioinformatics[8,9]. However, even with minimizers[4] the overlap step can take a substantial amount of time when handling larger genomes. To tackle this problem, we enhanced the first Minimap[4] algorithm following the MinHash approach[10], selecting a fixed number of lexicographically smallest values as the sequence sketch. As minimizers were proven to be suitable for genome assembly of uncorrected reads[7], we wanted to decrease the number of minimizers per read while keeping a uniform base coverage and avoid alterations of the original algorithm. Based on empirical evaluations, we opted for retaining |read| / k minimizers, where k is the minimizer length. This significantly accelerated the overlap step without a major impact on sensitivity (Figure 1).

Raven loads the whole sequencing sample into memory and finds overlaps in fixed-size blocks to decrease the memory footprint. Found overlaps are immediately transformed into pile-o-grams[11] and discarded, except the longest N per read which are used for containment removal. Chimeric reads are iteratively identified and chopped by detecting sharp declines of coverage in pile-o-grams using coverage medians based on the stored suffix-prefix overlaps. As Minimap ignores the most frequent minimizers, which are critical for good repeat annotations, we lower this threshold while overlapping all contained reads to the set of containment-free reads. Afterwards, the containment-free read set is overlapped to itself utilizing all minimizers, and repeat annotations from updated pile-o-grams are used to remove false overlaps between repetitive reads. Once the assembly graph[4] is created, it is simplified stepwise with transitive reduction, tip removal, and bubble popping. Eventually, we lay the

graph out in a two-dimensional Euclidean system and search for edges that connect distant parts of the graph. Applying the force-directed placement algorithm[12], which draws tightly connected vertices together, we can distinguish undetected chimeric or repeat-induced edges as there are elongated with respect to others due to their rareness (Figure 2). Creating unitigs that are far away from junction vertices coupled with the hierarchical force-calculation algorithm[13] made this drawing based simplification method feasible for even the largest assembly graphs. To finalize the assembly, contiguous paths of the graph are passed to two rounds of Racon[7].

Since an earlier version proved as one of the best performers in a comprehensive benchmark[14] at prokaryotic datasets we additionally evaluated several state-of-the-art assemblers alongside Raven on six large genome datasets, covering both third-generation sequencing technologies (Table 1). Reported metrics were obtained with QUAST-LG[15] and BUSCO[16]. Raven consistently reports the minimal amount of contigs, is one of two fastest assemblers on all datasets while having better or comparable contiguity and accuracy. Its advantages especially come to the fore with nanopore reads.

In addition, we run Raven on a couple of plant datasets from two scientific studies[17,18] and compared their results (Table 2). On datasets *B. oleracea*, *B. rapa* and *M. schizocarpa* Raven produces comparable assemblies to the best reported, obtained with Ra[19], but in a fraction of time. Furthermore, both *O. sativa* assemblies are more contiguous than the ones reported with Flye, but the BUSCO scores are a bit lower as they were not polished with Illumina data.

We showcased new algorithms for the overlap and layout phases of de novo genome assembly that reduce execution time and increase contiguity of the final assembly. We integrated them with an overlap module based on Minimap, and the consensus module Racon into a powerful standalone tool called Raven optimized for error-prone long reads. We argue that its performance coupled with the reduced cost per base of long-read sequencing technologies will enable assembly of large genomes even to laboratories with limited funding.

Methods

First, Raven constructs pile-o-grams and removes contained sequences with the Minimap algorithm, using 15-mers, a sliding window of 5 bases and the k-mer frequency filter of 10^{-3}. The whole sequencing data set is loaded into memory, and reads are overlapped to each other in 1Gbp vs 4Gbp chunks. To save time, Raven picks only the smallest |read| / 15 minimizers both in the index and in the query. Once a block is processed, all overlaps are stacked into pile-o-grams that are decimated to every 16-th base to save memory. The longest 16 overlaps are also stored for containment removal and connected component retrieval. When all pairwise overlaps are obtained, coverage medians are calculated for each pile-o-gram, reads are trimmed to the longest region covered with at least 4 other reads, and potential chimeric sites are detected by finding bases which have 1.82 times smaller coverage than their neighbouring bases. Contained sequences are dropped only if the containing read does not have a potential chimeric region. Decreasing the number of reads to a mere couple of percent enables faster verification of chimeric annotations, that is if the coverage drop is consistent with the component median the pile-o-gram belongs to. Problematic reads are chopped to the longest non-chimeric region, which is done iteratively to capture different molecule copy numbers.

Second, Raven searches for suffix-prefix overlaps between the remaining reads enforcing the use of all minimizers. In addition, all contained sequences are overlapped with the remaining reads to increase the coverage of repetitive regions, which is flawed due to the minimizer frequency filter. Decreasing the filter to 10^{-5} enables proper repeat annotation in which sought bases have coverage at least 1.42 times larger than the component median. All annotation thresholds were empirically

determined. Repetitive regions at either end of a read are used to iteratively remove false overlaps, i.e. overlaps that connect different copies of bridged repeats. Once the overlap set is cleansed, the assembly graph is built and simplified stepwise with standard algorithms such as transitive reduction, tipping, and bubble popping.

Information about transitive connections is kept for the next step, which plots the assembly graph in a two-dimensional space. Raven searches for edges that connect remote parts of the graph, which are usually there due to leftover sequencing artefacts or unresolved repeats. The drawing algorithm enlarges the majority of such edges due to their rareness. Given the quadratic time complexity of the algorithm ($O(|V|^2)$) and needed 100 iterations until convergence, we shrink the graph by creating unitigs that are 42 vertices away from any junction vertex. Furthermore, approximating the forces of distant vertices by replacing them with their centre of mass enables linearithmic time complexity and the use on larger genomes. Depending on vertex distances in a finished drawing, Raven removes outgoing edges that are at least twice as long as any other outgoing edge of that junction vertex. As the drawing depends on an initial random layout, the whole procedure is restarted 16 times.

Finally, paths of the assembly graphs without external branches are polished with a library version of Racon, using small windows of 500bp and partial order alignment with linear gaps, in a total of two iterations.

Assemblers Raven (v1.1.10), Canu (v2.0), Flye (v2.7.1), Shasta (v0.4.0) and Wtdbg2 (v2.5) were run on 64 threads with appropriate parameters for any given genome. Raven does not require knowledge of the genome size in advance, and it was run without any additional parameters on all six datasets. Canu, Flye and Wtdbg2 were run with approximate genome sizes of 120Mb, 144Mb and 3Gb for *A. thaliana*, *D. melanogaster* and *H. sapiens* datasets. In addition, Canu was given options '-pacbio' or '-nanopore', Flye '–pacbio-raw' or '–nano-raw', while Wtdbg2 '-x sq', '-x rs' or '-x ont', depending on sequencing technology. Canu was not run on the human datasets due to its long-running time and limited resources.

We used QUAST-LG (v5.0.2) for the majority of the metrics and ran it with the minimal identity of 85%. BUSCO (v4.0.6) was run against databases *brassicalles*, *diptera* and *primates* for *A. thaliana*, *D. melanogaster* and *H. sapiens* datasets, respectively. Database *embryophyte* was searched against the five plant datasets to match the one used in the publications (although, the current database version contains more orthologs).

ONT *A. thaliana* dataset is available under the accession number ERR2173373, ONT *D. melanogaster* under SRR6702603, PacBio *D. melanogaster* under SRR5439404, and PacBio *H. sapiens HG0073* under SRR7615963. ONT *H. sapiens NA12878* reads can be downloaded here (release 6), and PacBio *A. thaliana* dataset is available from here. Accession number of the plant datasets used for separate Raven evaluation can be found in corresponding publications.

Acknowledgments

This work has been supported in part by the Croatian Science Foundation under the project Single genome and metagenome assembly (IP-2018-01-5886), by the European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS), and by the A*STAR Computational Resource Centre through the use of their high-performance computing facilities. M.Š. has been partially supported by funding from A*STAR, Singapore. We acknowledge Intel® Corporation for allowing us to test with Intel® Optane™ Persistent Memory server and providing us with high-quality technical support. Finally, we thank Goran Žužić from Carnegie Mellon University for useful discussions in the field of graph drawings.

References

1. 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).

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

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

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

5. Shafin, K. *et al.* Nanopore sequencing and the Shasta toolkit enable efficient de novo assembly of eleven human genomes. *Nat. Biotechnol.* (2020) doi:10.1038/s41587-020-0503-6.

6. Ruan, J. & Li, H. Fast and accurate long-read assembly with wtdbg2. *Nat. Methods* **17**, 155–158 (2020).

7. 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).

8. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410 (1990).

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

10. Broder, A. Z. On the resemblance and containment of documents. in *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)* 21–29 (1997). doi:10.1109/SEQUEN.1997.666900.

11. 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).

12. Fruchterman, T. M. J. & Reingold, E. M. Graph drawing by force-directed placement. *Softw. Pract. Exp.* **21**, 1129–1164 (1991).

13. Barnes, J. & Hut, P. A hierarchical O(N log N) force-calculation algorithm. *Nature* **324**, 446–449 (1986).

14. Wick, R. R. & Holt, K. E. Benchmarking of long-read assemblers for prokaryote whole genome sequencing [version 2; peer review: 4 approved]. *F1000Research* **8**, (2020).

15. Mikheenko, A., Prjibelski, A., Saveliev, V., Antipov, D. & Gurevich, A. Versatile genome assembly evaluation with QUAST-LG. *Bioinformatics* **34**, i142–i150 (2018).

16. Simão, F. A., Waterhouse, R. M., Ioannidis, P., Kriventseva, E. V & Zdobnov, E. M. BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics* **31**, 3210–3212 (2015).

17. Belser, C. *et al.* Chromosome-scale assemblies of plant genomes using nanopore long reads and optical maps. *Nat. Plants* **4**, 879–887 (2018).

18. Choi, J. Y. *et al.* Nanopore sequencing-based genome assembly and evolutionary genomics of circum-basmati rice. *Genome Biol.* **21**, 21 (2020).

19. Vaser, R. & Šikić, M. Yet another de novo genome assembler. in *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)* 147–151 (2019). doi:10.1109/ISPA.2019.8868909.

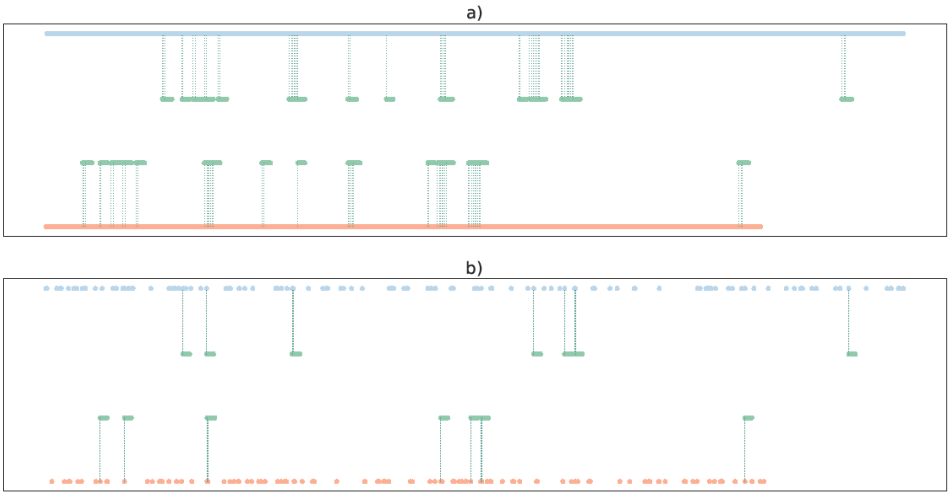

Figure 1 Overlap between two erroneous reads based on minimizer matches. Raven uses the Minimap algorithm to find pairwise overlaps in which minimizers of both sequences are collected (blue and orange) and a linear chain of matches is found (green). a), While collecting all minimizers from a small sliding window ensures the retrieval of the majority of overlaps between similar sequences, b) the same can be achieved by picking only a portion of the smallest minimizers. Shrinking the minimizer search space, without any other modifications, greatly accelerates the algorithm, and justifies the small impact on sensitivity.

Figure 2 Bacterial assembly graph drawn with the force-directed placement algorithm. Raven uses vertex distances in the Euclidean system to find elongated edges (red) that connect junction vertices, and removes the longest ones. Those represent false connections which occur either due to sequencing errors or repetitive genomic regions. Without unitig creation (large circles) and the hierarchical force calculation, the drawing algorithm would partake a extensive amount of time on larger genomes.

Table 1 Evaluation of long-read assemblers

Dataset	Metric	Raven	Canu	Flye	Shasta	Wtdbg2
Arabidopsis thaliana ONT 28x	Total length (Mb)	116.9	112.2	127.0	82.5	117.0
	NG50	11,148,841	2,614,923	11,187,127	150,227	9,827,355
	NG75	9,022,209	381,602	9,062,625	132,126	2,540,904
	No. of contigs	25	448	210	1386	353
	No. of misassemblies	4,926	4,130	6,142	1,910	4,148
	Duplication ratio	1.019	1.011	1.098	0.995	0.989
	No. of mismatches / 100 kb	916.02	760.74	1,346.01	1,056.92	820.91
	No. of indels / 100 kb	1,559.14	2,327.17	1,774.82	2,498.98	2,908.85
	% complete and single-copy BUSCOs	22.063	7.855	17.994	4.156	3.525
	% fragmented BUSCOs	4.047	1.414	3.960	1.066	0.827
	CPU time (h)	5.16	1,157.51	26.06	1.93	19.79
	Memory (GB)	31.61	10.56	112.06	17.07	15.77

Arabidopsis thaliana PacBio 90x	Total length (Mb)	121.4	125.8	120.7		121.1
	NG50	11,028,093	719,843	9,066,195		12,210,142
	NG75	6,116,985	312,269	6,254,227		6,157,121
	No. of contigs	66	591	318		280
	No. of misassemblies	6,341	6,608	6,351		6,040
	Duplication ratio	1.067	1.116	1.061		1.061
	No. of mismatches / 100 kb	1,051.35	1,244.82	1,080.96		1,060.07
	No. of indels / 100 kb	389.08	240.75	189.41		437.48
	% complete and single-copy BUSCOs	88.142	95.017	98.020		92.015
	% fragmented BUSCOs	2.807	0.037	0.013		1.393
	CPU time (h)	25.85	238.86	70.12		43.44
	Memory (GB)	45.17	12.22	66.63		25.65
Drosophila melanogaster ONT 32x	Total length (Mb)	135.9	148.3	138.3	70.8	136.4
	NG50	6,167,979	4,562,121	14,366,038	92,213	10,619,799
	NG75	1,718,865	920,326	3,526,515	49,844	1,344,721
	No. of contigs	148	664	791	1626	635
	No. of misassemblies	1,204	3,163	1,671	204	1,418
	Duplication ratio	1.029	1.098	1.033	1.017	1.006
	No. of mismatches / 100 kb	175.06	221.96	194.34	557.62	298.49
	No. of indels / 100 kb	719.56	932.37	407.53	2040.42	1,486.03
	% complete and single-copy BUSCOs	76.256	70.015	89.498	8.250	33.790
	% fragmented BUSCOs	9.437	11.872	4.079	13.364	23.135
	CPU time (h)	5.87	520.75	49.02	1.74	26.90
	Memory (GB)	41.78	13.08	46.25	15.61	19.25
Drosophila melanogaster PacBio 127x	Total length (Mb)	138.9	144.1	136.3		137.9
	NG50	12,821,782	13,800,773	13,738,344		17,047,134
	NG75	3,667,783	5,572,055	4,291,297		4,256,364
	No. of contigs	121	254	318		311
	No. of misassemblies	4,257	5,000	3,926		4,214
	Duplication ratio	1.072	1.106	1.054		1.055
	No. of mismatches / 100 kb	610.94	652.55	619.34		705.28
	No. of indels / 100 kb	208.01	158.46	144.29		364.08
	% complete and single-copy BUSCOs	97.991	98.387	98.965		97.869
	% fragmented BUSCOs	0.049	0.012	0.012		0.046
	CPU time (h)	27.42	389.18	120.74		20.54
	Memory (GB)	70.14	19.08	75.32		19.36

Homo sapiens ONT 44x	Total length (Mb)	2,820.6		2,868.2	2,781.8	2,690.2
	NG50	22,953,291		30,370,369	6,450,220	8,233,692
	NG75	8,729,182		9,966,721	1,877,069	1,142,157
	No. of contigs	259		2330	2390	5147
	No. of misassemblies	1,257		3,432	640	2,030
	Duplication ratio	1.002		1.009	0.999	1.001
	No. of mismatches / 100 kb	142.27		194.53	162.50	225.52
	No. of indels / 100 kb	236.44		360.44	365.69	703.17
	% complete and single-copy BUSCOs	72.736		66.430	54.826	44.623
	% fragmented BUSCOs	3.970		3.752	3.614	2.990
	CPU time (h)	485.81		1,487.14	69.68	1,993.88
	Memory (GB)	373.31		711.77	438.77	279.13
Homo sapiens PacBio 93x	Total length (Mb)	2,840.4		2,848.4		2,797.9
	NG50	18,503,245		25,737,858		26,634,850
	NG75	6,607,980		7,134,763		6,466,467
	No. of contigs	546		1850		2857
	No. of misassemblies	1,053		1,925		1,699
	Duplication ratio	1.010		1.009		1.004
	No. of mismatches / 100 kb	136.00		139.21		162.62
	No. of indels / 100 kb	138.57		39.72		214.99
	% complete and single-copy BUSCOs	82.983		88.774		80.327
	% fragmented BUSCOs	2.649		1.647		2.794
	CPU time (h)	1,982.71		4,283.21		2,320.40
	Memory (GB)	691.45		1,253.39		338.65

Table 2 Raven plant assemblies (values in brackets represent assembly metrics in corresponding publications). Oryza genomes in original publication were additionally polished with Illumina reads

Metric \ Dataset	*Brassica oleracea*	*Brassica rapa*	*Musa schizocarpa*	*Oryza sativa basmati 334*	*Oryza sativa dom sufid*
Total length (Mb)	537.7 (546.4)	352.8 (375.3)	536.4 (522.0)	382.8 (386.6)	381.0 (383.6)
N50	6,375,585 (7,277,585)	5,537,784 (3,799,257)	2,484,573 (2,134,507)	6,734,245 (6,320,000)	11,920,281 (10,530,000)
No. of contigs	252 (244)	410 (544)	546 (615)	159 (188)	109 (116)
% complete BUSCOs	74.473 (74.3)	79.864 (79.7)	43.371 (53.8)	96.592 (97.6)	97.212 (97.0)
CPU time (h)	48.28 (261.4)	76.33 (315.7)	117.82 (245.6)		