PREPRINT

`yacrd` and `fpa`: upstream tools for long-read genome assembly

Pierre Marijon [1,*], Rayan Chikhi [2] and Jean-Stéphane Varré [3]

[1]Inria, Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRIStAL, F-59000 Lille, France.
[2]Institut Pasteur, C3BI USR 3756 IP CNRS, Paris, France
[3]Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL, F-59000 Lille, France.

*To whom correspondence should be addressed.

Abstract

Motivation: Genome assembly is increasingly performed on long, uncorrected reads. Assembly quality may be degraded due to unfiltered chimeric reads; also, the storage of all read overlaps can take up to terabytes of disk space. **Results:** We introduce two tools, `yacrd` and `fpa`, to respectively perform chimera removal/read scrubbing, and filter out spurious overlaps. We show that `yacrd` results in higher-quality assemblies and is two orders of magnitude faster than the best available alternative.
Availability: `https://github.com/natir/yacrd` and `https://github.com/natir/fpa`
Contact: pierre.marijon@inria.fr
Supplementary information: Supplementary data are available online.
Acknowledgements: This work was supported by Inria and the INCEPTION project (PIA/ANR-16-CONV-0005) and the University of Lille HPC facility. The authors thank Maël Kerbiriou for algorithmic help.

1 INTRODUCTION

Third-generation DNA sequencing (PacBio, Oxford Nanopore) is increasingly becoming a go-to technology for the construction of reference genomes (*de novo* assembly). New bioinformatics methods for this type of data are rapidly emerging.

Some long-read assemblers perform error-correction on reads prior to assembly. Correction helps reduce the high error rate of third-generation reads and make assembly tractable, but is also a time and memory-consuming step. Recent assemblers (e.g. Li (2016); Ruan and Li (2019) among others) have found ways to directly assemble raw uncorrected reads. Here we will therefore focus only on **correction-free assembly**. In this setting, assembly quality may become affected by e.g. chimeric reads and highly-erroneous regions (Myers, 2015), as we will see next.

The `DASCRUBBER` program (Myers, 2017) introduced the concept of read "scrubbing", which consists of quickly removing problematic regions in reads without attempting to otherwise correct bases. The idea is that scrubbing reads is a more lightweight operation than correction, and is therefore suitable for high-performance and correction-free genome assemblers.

`DASCRUBBER` performs all-against-all mapping of reads and constructs a pileup for each read. Mapping quality is then analyzed to determinate putatively high error rate regions, which are replaced by equivalent and higher-quality regions from other reads in the pileup. `MiniScrub` (LaPierre *et al.*, 2018) is another scrubbing tool that uses a modified version of `Minimap2` (Li, 2017) to record positions of the anchors used in overlap detection. For each read, `MiniScrub` converts anchors positions to an image. A convolutional neural network then detects and removes of low quality read regions.

Another problem that is even more upstream of read scrubbing is the computation of overlaps between reads. The storage of overlaps is disk-intensive and to the best of our knowledge, there has never been an attempt at optimizing its potentially high disk space.

In this paper we present two tools that together optimize the early steps of long-read assemblers. One is `yacrd` (for Yet Another Chimeric Read Detector) for fast and effective scrubbing of reads, and the other is `fpa` (for Filter Pairwise Alignment) which filters overlaps found between reads.

2 MATERIALS & METHODS

Similarly to `DASCRUBBER` and `MiniScrub`, **`yacrd`** is based on the assumption that low quality regions in reads are not well-supported by other reads. To detect such regions `yacrd` performs all-against-all read mapping using `Minimap2` and then computes the base coverage of each read. Contrarily to `DASCRUBBER` and `MiniScrub`, `yacrd` only uses approximate positional mapping information given by `Minimap2`, which avoids the time-expensive alignment step. This comes at the expense of not having base-level alignments, but this will turn out to be sufficient for performing scrubbing. Reads are split at any location where coverage drops below a certain threshold (set to 4 by default), and the low-coverage region is removed entirely. A read is completely discarded if less than 40% of its length is below the coverage threshold. `yacrd` time complexity is linear in the number of overlaps.

`yacrd` performance is directly linked to the overlapper performance. We tuned `Minimap2` parameters (especially the maximal distance between two minimizers, `-g` parameter) to find similar regions between reads and not to create bridges over low quality regions (see Supplementary Section 3). `yacrd` takes reads and their overlaps as input, and produces scrubbed reads, as well as a report.

`fpa` operates between the overlapper and the assembler. It filters out overlaps based on a highly customizable set of parameters, such

		H. sapiens chr1 (ONT ultra-long R9.4)			*C. elegans* (Pacbio P6-C4)		
		raw	dascrubber	yacrd	raw	dascrubber	yacrd
Reads	# reads	1,075,867	819,798	1,044,848	740,776	660,766	751,750
	Relative # bases	1.00	0.71	0.80	1.00	0.84	0.84
	N50	10,568	9,858	9,520	16,572	15,667	15,845
	# chimera	25,888	6 %	20 %	71,704	13 %	21 %
	Time		**3 days 2 hours**	**27 mins**		**1 day 20 hours**	**33 mins**
`Miniasm`	# contigs	184	184	394	226	131	154
	NGA50	96,225	410,37	453,748	432,112	544,677	440,776
	Asm/Ref size	81 %	78 %	81 %	113 %	108 %	110 %
	# misassemblies	1,745	209	432	1,396	754	1,015
`Wtdbg2`	# contigs	810	496	485	139	100	122
	NGA50	1,513,450	545,902	1,482,513	565,278	578,041	593,039
	Asm/Ref size	87 %	80 %	84 %	106 %	104 %	106 %
	# misassembly	1,316	177	582	614	485	577

Table 1. Performance of `yacrd` compared to `DASCRUBBER` on an ONT and a PacBio dataset. Relative #bases indicates the proportion of raw read bases kept after scrubbing. # chimera indicates the number of chimeric reads detected in the dataset using `Minimap2` (see Supplementary Section 4) and the proportion of remaining chimeric reads after scrubbing. NGA50 is the N50 of aligned contigs, and # misassemblies are the number of misassemblies, both metrics were computed by QUAST (Gurevich *et al.*, 2013). Asm/Ref size indicates the relative length of the assembly divided the reference length.

as overlap length, length of reads names, etc. `fpa` can remove self-overlaps, end-to-end overlaps, containment overlaps, internal matches (when e.g. two reads share a repetitive region) as defined in (Li, 2016). `fpa` supports the PAF or BLASR m4 formats as inputs and outputs, with optional compression. `fpa` can also rename reads, generate an index of overlaps and output an overlap graph in GFA format.

`yacrd` and `fpa` are evaluated on several datasets (details provided in Supplementary Section 1), and here we highlight their performance on two of them: *H. sapiens* chromosome 1 Oxford Nanopore (ONT) ultra-long reads, and *C. elegans* PacBio reads. All tools were run on a single cluster node with recommended parameters (see Supplementary Section 2). Scrubbed reads were then assembled using both `Miniasm` and `Wtdbg2` with recommended parameters for each sequencing technology.

3 RESULT & DISCUSSION

Table 1 compares the results of **`yacrd`** and `DASCRUBBER`. We also evaluated `MiniScrub` (see Supplementary Section 2 and 5), but its memory usage exceeded 256 GB on the two datasets of Table 1.

The main feature of `yacrd` is that its total execution time, which is essentially that of `Minimap2`, is two orders of magnitude faster than `DASCRUBBER`. We next evaluate whether running `yacrd` results in higher-quality reads and assemblies. `yacrd` removes 20-27% of the bases in raw reads, comparably to `DASCRUBBER`. Both scrubbers significantly reduce chimeras: only 6-13% of those in raw reads remain with `DASCRUBBER` and 18-20% with `yacrd`. The impact of removing chimeras is directly seen on assembly metrics: both scrubbers produce significantly less misassemblies with `Miniasm` and `Wtdbg2` than with direct assembly of raw reads. Both `yacrd` and `DASCRUBBER` resulted in increased contiguity (NGA50) with `Miniasm`, and equivalent (or significantly degraded for `DASCRUBBER`) contiguity with `Wtdbg2`, and comparable assembly lengths.

On ONT reads, `DASCRUBBER` reduces the number of misassemblies by a factor of 2-3 more than `yacrd`. However, given that all assemblies in Table 1 completed in less than an hour and `DASCRUBBER` took 3 days, running this tool on larger datasets would become a significant performance bottleneck. In Supplementary Section 3 we examine the behavior of `yacrd` across its parameter space. We observe that different parameters worked best for different datasets, one of which is actually a parameter for `Minimap2`.

`fpa` reduced the size of reads overlap file (PAF file produced by `Minimap2`) by 40-79% on the evaluated datasets, without any significant effect on quality assembly. As a consequence this reduces the memory usage of `Miniasm` by 13-67%. Other performance metrics are presented in Supplementary Table 5.

Finally, we examine the effect of combining both `yacrd` and `fpa`. We propose a pipeline based on `Miniasm` (Supplementary Section 7) and show that it results in improved assembly contiguity, comparable assembly size, less mismatches and indels, less misassemblies, at the cost of a reasonable increase in running time (around 2x).

REFERENCES

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

LaPierre, N. *et al.* (2018). MiniScrub: de novo long read scrubbing using approximate alignment and deep learning. *bioRxiv*.

Li, H. (2016). Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, **32**(14), 2103–2110.

Li, H. (2017). Minimap2: pairwise alignment for nucleotide sequences.

Myers, G. (2015). Intrinsic quality values. `https://dazzlerblog.wordpress.com/2015/11/06/intrinsic-quality-values/`.

Myers, G. (2017). Scrubbing reads for better assembly. `https://dazzlerblog.wordpress.com/2017/04/22/1344/`.

Ruan, J. and Li, H. (2019). Fast and accurate long-read assembly with wtdbg2. *bioRxiv*.