Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

OXFORD

Sequence analysis

CONSENT: Scalable self-correction of long reads with multiple sequence alignment

Pierre Morisse [1,*], Camille Marchet [2], Antoine Limasset [2], Thierry Lecroq [1] and Arnaud Lefebvre [1]

[1] Normandie Univ, UNIROUEN, LITIS, 76000 Rouen, France
[2] Univ. Lille, CNRS, UMR 9189 - CRIStAL.

* To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Third generation sequencing technologies such as Pacific Biosciences and Oxford Nanopore allow the sequencing of long reads of tens of kbs, that are expected to solve various problems, such as contig and haplotype assembly, scaffolding, and structural variant calling. However, they also reach high error rates of 10 to 30%, and thus require efficient error correction. As first long reads sequencing experiments produced reads displaying error rates higher than 15% on average, most methods relied on the complementary use of short reads data to perform correction, in a hybrid approach. However, these sequencing technologies evolve fast, and the error rate of the long reads is now capped at around 10-12%. As a result, self-correction is now frequently used as a first step of third generation sequencing data analysis projects. As of today, efficient tools allowing to perform self-correction of the long reads are available, and recent observations suggest that avoiding the use of second generation sequencing reads could bypass their inherent bias.
Results: We introduce CONSENT, a new method for the self-correction of long reads that combines different strategies from the state-of-the-art. A multiple sequence alignment strategy is thus combined to the use of local de Bruijn graphs. Moreover, the multiple sequence alignment benefits from an efficient segmentation strategy based on k-mers chaining, allowing to greatly reduce its time footprint. Our experiments show that CONSENT compares well to the latest state-of-the-art self-correction methods, and even outperforms them on real Oxford Nanopore datasets. In particular, they show that CONSENT is the only method able to scale to a human dataset containing Oxford Nanopore ultra-long reads, reaching lengths up to 340 kbp.
Availability and implementation: CONSENT is implemented is C++, supported on Linux platforms and freely available at https://github.com/morispi/CONSENT.
Contact: pierre.morisse2@univ-rouen.fr

1 Introduction

Third generation sequencing technologies Pacific Biosciences and Oxford Nanopore has become widely used since their inception in 2011. In contrast to second generation technologies, producing reads reaching lengths of a few hundred base pairs, they allow the sequencing of much longer reads (10 kbp on average (Sedlazeck *et al.*, 2018b), and up to >1 million bp (Jain *et al.*, 2018)). These long reads are expected to solve various problems, such as contig and haplotype assembly (Patterson *et al.*, 2015; Kamath *et al.*, 2017), scaffolding (Cao *et al.*, 2017), and structural variant calling (Sedlazeck *et al.*, 2018a). They are however very noisy, and reach error rates of 10 to 30%, whereas second generation short reads usually display error rates of 1%. The error profiles of these long reads are also much more complex than those of the short reads, as they are mainly composed of insertions and deletions, while short reads' are mostly

composed of substitutions. As a result, error correction is often necessary, as a first step of projects dealing with long reads. As the error profiles and error rates of the long reads are much different than those of the short reads, correcting long reads requires specific algorithmic developments.

The error correction of long reads has been tackled by two main approaches. The first approach, hybrid correction, makes use of additional short reads data to perform correction. The second approach, self-correction, aims at correcting the long reads solely based on the information contained in their sequences. Hybrid correction methods rely on different techniques such as:

- Alignment of short reads to the long reads (CoLoRMAP (Haghshenas *et al.*, 2016), HECiL (Choudhury *et al.*, 2018))
- de Bruijn graph exploration (LoRDEC (Salmela and Rivals, 2014), Jabba (Miclotte *et al.*, 2016), FMLRC (Wang *et al.*, 2018))
- Alignment of the long reads to contigs built from the short reads (MiRCA (Kchouk and Elloumi, 2018), HALC (Bao and Lan, 2017))
- Hidden Markov Models (Hercules (Firtina *et al.*, 2018))
- Combination of different strategies (NaS (Madoui *et al.*, 2015), HG-CoLoR (Morisse *et al.*, 2018))

Self-correction methods usually build around the alignment of the long reads against each other (PBDAGCon (Chin *et al.*, 2013), PBcR (Koren *et al.*, 2013)).
As first long reads sequencing experiments resulted in highly erroneous long reads (15-30% error rates on average), most methods relied on the additional use of short reads data. As a result, hybrid correction used to be much more studied and much more developed. Indeed, in 2014, 5 hybrid correction tools and only 2 self-correction tools were available. However, third generation sequencing technologies evolve fast, and now manage to produce long reads reaching error rates of 10-12%. Moreover, long read sequencing technologies' evolution also allows to produce higher throughputs of data, at a reduced cost, and consequently such data became more widely available. Thus, self-correction is now frequently used as a first step of data analysis projects dealing with long reads.

1.1 Related works

Due to the fast evolution of third generation sequencing technologies, and to the lower error rates they now reach, various efficient self-correction methods were recently developed. Most of them share the common first step of computing overlaps between the long reads. This can be done via a mapping approach, which only provides the positions of the similar regions of the long reads (Canu (Koren *et al.*, 2017), MECAT (Xiao *et al.*, 2017), FLAS (Bao *et al.*, 2018)), or via alignment, which provides the positions of the similar regions, and their actual base to base correspondence in terms of matches, mismatches, insertions, and deletions (PBDAGCon, PBcR, Daccord (Tischler and Myers, 2017)). A directed acyclic graph (DAG) is then usually built, in order to summarize the 1V1 alignments and compute consensus, after recomputing actual alignments of mapped regions, if necessary. Other methods rely on de Bruijn graphs, either built from small windows of the alignments (Daccord), or directly from the long reads sequences with no alignment or mapping step at all (LoRMA).

However, methods relying on direct alignment of the long reads are prohibitively time and memory consuming, and current implementations thus do not scale to large genomes. Methods solely relying on de Bruijn graphs and avoiding the alignment step altogether usually require deep long reads coverage, as the graphs are built for large values of k. As a result, methods relying on a mapping strategy constitute the core of the current state-of-the-art for long read self-correction.

1.2 Contribution

We present CONSENT, a new self-correction method that combines different efficient approaches from the state-of-the-art. CONSENT indeed starts by computing multiple sequence alignments between overlapping regions of the long reads, in order to compute consensus sequences. These consensus sequences are then further polished with the help of local de Bruijn graphs, in order to correct remaining errors, and reduce the final error rate. Moreover, unlike other current state-of-the-art methods, CONSENT computes actual multiple sequence alignments, using a method based on partial order graphs (Lee *et al.*, 2002). In addition, we introduce an efficient segmentation strategy based on k-mers chaining, which allows to reduce the time footprint of the multiple sequence alignments. This segmentation strategy thus allows to compute scalable multiple sequence alignments. In particular, it allows CONSENT to efficiently scale to Oxford Nanopore ultra-long reads.

Our experiments show that CONSENT compares well to the latest state-of-the-art self-correction methods, and even outperforms them on real Oxford Nanopore datasets. In particular, they show that CONSENT is the only method able to scale to a human dataset containing Oxford Nanopore ultra-long reads, reaching lengths up to 340 kbp.

2 Methods

2.1 Overview

textcolorblack CONSENT takes as input a FASTA file of long reads, and returns a set of corrected long reads, reporting corrected bases in uppercase, and uncorrected bases in lowercase. Like most efficient methods, CONSENT starts by computing overlaps between the long reads using a mapping approach. These overlaps are computed using an external program, and not by CONSENT itself. This way, only matched regions need to be further aligned in order to compute consensus. These matched regions are further divided into smaller windows, that are aligned independently. The alignment of these windows is performed via a multiple sequence alignment strategy based on partial order graphs. This multiple sequence alignment is realized by iteratively constructing and adding sequences to a DAG. It also benefits from an efficient heuristic, based of k-mers chaining, allowing to reduce the time footprint of computing multiple sequence alignment between noisy sequences. The DAG is then used to compute the consensus of a given window. Once a consensus has been computed, a second step makes use of a local de Bruijn graph, in order to polish the consensus. This allows to further correct weakly supported regions, that are, regions containing weak k-mers, and thus reduce the final error rate of the consensus. Finally, the consensus is realigned to the read, and correction is performed for each window. CONSENT's workflow is summarized in Figure 1.

2.2 Definitions

Before presenting the CONSENT pipeline, we recall the notions of alignment piles and windows on such piles, as proposed in Daccord, as they will be used throughout the rest of the paper.

2.2.1 Alignment piles

An alignment pile represents a set of reads that overlap with a given read A. More formally, it can be defined as follows. For any given read A, we define an alignment pile for A as a set of alignment tuples $(A, R, Ab, Ae, Rb, Re, S)$ where R is a long read id, Ab and Ae represent respectively the start and the end positions of the alignment on A, Rb and Re represent respectively the start and the end positions of the alignment on R, and S indicates whether R aligns forward (0) or reverse complement (1) to A. One can remark that, compared to Daccord, this definition is slightly

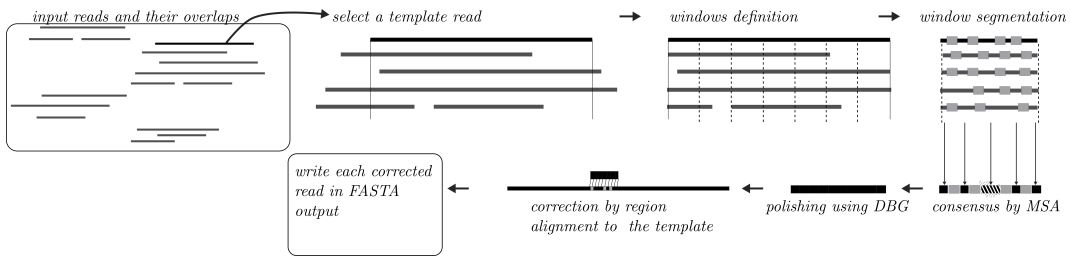

Fig. 1: Overview of CONSENT's worfklow for long read error correction.

altered. In particular, Daccord adds an edit script to the pile, representing the sequence of edit operations needed to transform $A[Ab..Ae]$ into $R[Rb..Re]$ if $S = 0$, or into $\overline{R}[Rb..Re]$ if $S = 1$ (where $\overline{R}$ represents the reverse-complement of read R). This edit script can easily be retrieved by Daccord, as it relies on DALIGNER (Myers, 2014) to compute actual alignments between the long reads. However, as CONSENT relies on a mapping strategy, it does not have access to such an information, and we thus chose to exclude the edit script from the definition of a pile. In its alignment pile, we call the read A the *template read*. The alignment pile of a given template read A thus contains all the necessary information needed for its correction. An example of an alignment pile is given in Figure 2.

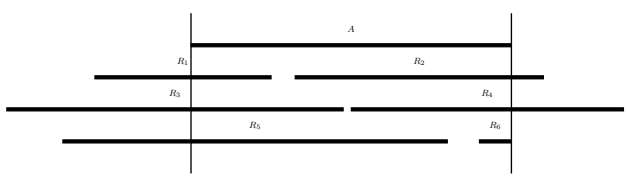

Fig. 2: An alignment pile for a template read A. The pile is delimited by vertical lines at the extremities of A. Prefixes and suffixes of reads overlapping A outside of the pile are not considered during the next steps, as the data they contain will not be useful for correcting A.

2.2.2 Windows on alignment piles

In addition to the alignment piles principle, Daccord also underlined the interest of processing windows from these piles instead of processing them all at once. A window from an alignment pile is defined as follows. Given an alignment pile for a template read A, a window of this pile is a couple (W_b, W_e), where W_b and W_e represent respectively the start and the end positions of the window on A, and such as $0 \leq W_b \leq W_e < |A|$, *i.e.* the start and end positions of the window define a factor of the template read A. We refer to this factor as the *window's template*. Additionally, in CONSENT, we will only consider for correction windows that have the two following properties:

- $W_e - W_b + 1 = L$ (*i.e.* windows have a fixed size)
- $\forall i$, $W_b \leq i \leq W_e$, $A[i]$ is supported by at least C reads of the pile, including A (*i.e.* windows have a minimum coverage threshold)

This second property allows to ensure that CONSENT has sufficient evidence to compute reliable consensus for a window. Examples of windows CONSENT does and does not consider are given in Figure 3.

In the case of Daccord, this window strategy allows to build local de Bruijn graphs with small values of k, thus overcoming the high error rates of the long reads, which causes issues when using large values of k. More generally, processing windows instead of whole alignment piles allows to divide the correction problem into smaller subproblems that can be solved faster. Specifically, in our case, as we seek to correct long reads by computing multiple alignment of sequences, working with windows allows to save both time and memory, since the sequences that need to be aligned are significantly shorter.

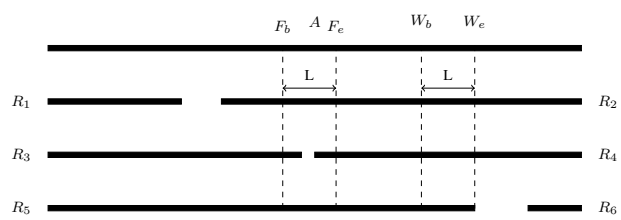

Fig. 3: When fixing the length to L and the minimum coverage threshold to 4, the window (W_b, W_e) will be processed by CONSENT. With these same parameters, the window (F_b, F_e) will not be processed by CONSENT, as $A[i]$ is not supported by at least 4 reads $\forall F_b \leq i \leq F_e$.

2.3 Overlapping

To avoid prohibitive computation time and memory consuming full alignments, CONSENT starts by overlapping the long reads using a mapping approach. By default, this step is performed with the help of Minimap2 (Li, 2018). However, error correction by CONSENT is not dependent on Minimap2, and the user can easily use any method for computing the overlaps between the long reads, as long as the overlaps file is provided to CONSENT in PAF format. We only included Minimap2 as the default overlapper for CONSENT as it offers good performances, both in terms of runtime and memory consumption, and is thus able to scale to large organisms on reasonable setups.

2.4 Alignment piles and windows computation

The alignment piles are computed by parsing the PAF file provided by the overlapper during the previous step. Each line of the file indeed contains all the necessary information to a define a tuple from an alignment pile: the ids of the two mapped long reads, the start and the end positions on the two reads, as well as the strand of the second read relatively to the first.

Given an alignment pile for a read A, we can then compute its set of windows. To this aim, we use an array of the length of A, allowing to count how many times each nucleotide of A is supported. The array is initialized with 1s at each position, and for each tuple $(A, R, Ab, Ae, Rb, Re, S)$, values at positions i such as $Ab \leq i \leq Ae$ are incremented. After all the tuples have been processed, the positions of the piles are retrieved by finding, in the array, sketches of length L of values $\geq C$, as we only consider windows of fixed length and with a minimum coverage threshold for correction. In practice, extracting overlapping windows instead of

partitioning the pile into a set of non-overlapping windows has proven to be efficient. This can be explained by the fact that, due to the multiple sequence alignment performed with the windows' sequences, consensus sequence might be missing at the extremities of certain windows, as it is usually harder to exploit alignments located on sequences extremities. Such events would thus cause a lack of correction on the read, when using non-overlapping windows. Each window is then processed independently during the next steps. Moreover, the reads are loaded into memory to support random access and thus accelerate the correction process. Each base is encoded using 2 bits in order to reduce memory usage. The memory consumption is thus roughly 1/4 of the total size of the reads.

2.5 Window consensus

The processing of a window is performed in two distinct steps. First, the sequences from the window are aligned using a multiple sequence alignment strategy based on partial order graphs, in order to compute consensus. This multiple sequence alignment strategy also benefits from an efficient heuristic, based on k-mers chaining, allowing to decompose the global problem into smaller instances, thus reducing both time and memory consumption. Second, once the consensus of the window has been computed, it is further polished with the help of a local de Bruijn graph, at the scale of the window, in order to get rid of the few errors that might remain despite consensus computation.

2.5.1 Consensus computation

In order to compute the consensus of a window, CONSENT uses POAv2 (Lee *et al.*, 2002), an implementation of a multiple sequence alignment strategy based on partial order graphs. These graphs are directed acyclic graphs, and are used as data structures containing all the information of the multiple sequence alignment. This way, at each step (*i.e.* at each alignment of a new sequence), the graph contains the current multiple sequence alignment result. To add a new sequence to the multiple alignment, the sequence is aligned to the DAG, using a generalization of the Smith-Waterman algorithm.

Unlike other methods that compute 1V1 alignments between the read to be corrected and other reads mapping to it, and then build a result DAG to represent the multiple sequences alignment, this strategy allows CONSENT to directly build the result DAG, during the multiple alignment. Indeed, the DAG is first initialized with the sequence of the window's template, and is then iteratively enriched by aligning the other sequences from the window, until it becomes the final, result graph. A matrix, representing the multiple sequence alignment, is then extracted from the graph, and consensus is computed by performing a majority voting. In the case of a tie at a given position, the nucleotide from the window's template is chosen as the consensus base.

However, computing multiple sequence alignments on hundred of bases from dozens of sequences is computationally expensive, especially when the divergence among sequences is high. To avoid the burden of building a consensus by computing full multiple sequence alignments of long sequences, we will search for collinear regions on these sequences, in order to split the global task into several smaller instances. Several consensus will thus be built on regions delimited by anchors shared among the sequences, and the global consensus will be reconstructed from the different, smaller corrected sequences obtained. The rationale is to benefit from the knowledge that all the sequences come from the same genomic area. This way we are able to, on the one hand, compute multiple sequence alignments on shorter sequences, which greatly reduces the computational cost. On the other hand, we only use related sequences to build the consensus, and therefore exclude spurious sequences. This behavior allows a massive speedup along with a gain in the global consensus quality.

To find such collinear regions, we first select k-mers that are non repeated in their respective sequences, and shared by multiple sequences. We therefore rely on dynamic programming to compute the longest anchors chain $a_1, \ldots, a_n$ such that:

1. $\forall i, j$ such that $1 \leq i < j \leq n$, a_i appears before a_j in every sequence that contains a_i and a_j
2. $\forall i$, $1 \leq i < n$, there is at least T reads containing a_i and a_{i+1} (with T a solidity threshold equal to 8 by default).

We therefore compute local consensuses using substrings between successive anchors among sequences that contain them, and output the global consensus: $consensus(prefix) + a_i + consensus(]a_1, a_2[) + a_2 + \cdots + consensus(]a_{n-1}, a_n[) + a_n + consensus(suffix)$.

2.5.2 Consensus polishing

After processing a given window, a few erroneous bases might remain on the computed consensus. This might especially happen in cases where the coverage depth of the window is relatively low, and thus cannot yield a high quality consensus. We propose an additional polishing feature to CONSENT as a proof of concept. This allows CONSENT to further enhance the quality of the consensus, by correcting the k-mers that are weakly supported. It is related to Daccord's local de Bruijn graph correction strategy.

A local de Bruijn graph is thus built from the window's sequences, only using small, solid, k-mers. The rationale is that small k-mers allows CONSENT to overcome the classical issues encountered due to the high error rate of the long reads, when using large k values. CONSENT searches for regions only composed of weak k-mers, flanked by sketches of n (usually, $n = 3$) solid k-mers. CONSENT then attempts to find a path allowing to link a solid k-mer from the left flanking region to a solid k-mer from the right flanking region. We call these k-mers *anchors*. The graph is thus traversed, in order to find a path between the two anchors, using backtracking if necessary. If a path between two anchors is found, the region containing the weak k-mers, is replaced by the sequence dictated by the path. If none of the anchors pairs could be linked, the region is left unpolished. To polish sketches of weak k-mers located at the left (respectively right) extremity of the consensus, highest weighted edges of the graph are followed, until the length of the followed path reached the length of the region to polish, or no edge can be followed out of the current node.

2.6 Anchor window consensus to the read

Once the consensus of a window has been computed and polished, it needs to be reanchored on the template long read. To this aim, it is realigned to the template, using an optimized library of the Smith-Waterman algorithm. To avoid time-costly alignment, the consensus is however only locally aligned around the positions of the window it has been computed from. This way, for a window (W_b, W_e) of the alignment pile of the read A, its consensus will be aligned to $A[W_b - O..W_e + O]$, where O represents the length of the overlap between consecutive windows processed by CONSENT (usually, $O = 50$, although it can be user defined). Aligning the consensus outside of the original window's extremities as such allows to take into account the error profile of the long reads. Indeed, as they mainly contain insertion(s) and deletion(s) errors, it is likely that the consensus computed from a window could be longer than the window it originates from, thus spanning outside of the window's extremities. In the case that alignment positions of the consensus from the ith window overlap with alignment positions of the consensus from the $(i + 1)$th window, the overlapping sequences of the two consensus are computed. The one containing the largest number of solid k-mers (where the k-mer frequencies of each sequence are computed from the window their consensus originate from)

is chosen and kept as the correction. In the case of a tie, we arbitrarily chose the sequence from consensus $i + 1$ as the correction. The aligned factor of the long read is then corrected by replacing it with the aligned factor of the consensus.

3 Results and discussion

3.1 Impact of the segmentation strategy

Before comparing CONSENT to any state-of-the-art self-correction tool, we first validate our segmentation strategy. To this aim, we simulated a 50x coverage of long reads from *E.coli*, with a 12% error rate, using SimLoRD (Stöcker *et al.*, 2016). The following parameters were used for the simulation: `-probability-threshold 0.3` `-prob-ins 0.145` `-prob-del 0.06`, and `-prob-sub 0.02`. We then ran the CONSENT pipeline, with, and without the segmentation strategy. Results of this experiment are given in Table 1. These results were obtained with LRCstats (La *et al.*, 2017), a tool specifically designed to measure correction accuracy on simulated long reads. These results show that, in addition to being 47x faster than the regular strategy, our segmentation strategy also allows to reach slightly lower memory consumption, and slightly higher throughput and quality.

	Without segmentation	With segmentation
Throughput	214,667,382	215,693,736
Error rate (%)	0.0757	0.0722
Runtime	5h31min	7min
Memory (MB)	750	675

Table 1. Comparison of the results obtained by CONSENT, with and without our segmentation stategy, as reported by LRCstats. Using the segmentation strategy allows a 47x speed-up, while producing slightly better results.

3.2 Comparison to the state-of-the-art

We now compare CONSENT against state-of-the-art error correction methods. We include the following tools in the benchmark: Canu, Daccord, FLAS, and MECAT. We voluntarly excluded LoRMA from the comparison, as it tends to split the reads a lot, and thus to produce reads that are usually shorter than 500 bp. We also exclude hybrid error correction tools from the benchmark, as we believe it makes more sense to only compare self-correction tools against each other. Comparison on simulated data is presented in Section 3.2.2, and comparison on real data in Section 3.2.3. Datasets used for these comparisons are presented in Section 3.2.1. All tools were ran with default or recommended parameters. For CONSENT, we used a minimum support of 4 to define a window, a window size of 500, an overlap size of 50 between the windows, a k-mer size of 9 for the chaining and the polishing, a threshold of 4 to consider k-mers as solid. Additionally, consensus were only computed for windows having a minimum number of 2 anchors.

3.2.1 Datasets

For our experiments, we used both simulated Pacific Biosciences and real Oxford Nanopore long reads. Pacific Biosciences reads were simulated with SimLoRD, using the following parameters: `-probability-threshold 0.3``-prob-ins 0.145`-prob-del 0.06`, and `-prob-sub 0.02`. Two datasets with a 12% error rate were thus generated for *E. coli*, *S. cerevisiae* and *C. elegans*: one with a 30x coverage, and one with a 60x coverage, corresponding to typical sequencing depth in current long reads experiments. As for the real Oxford Nanopore data, we used a 74x coverage dataset from *S. cerevisiae*, a 63x coverage dataset from *D. melanogaster*, and finally, a 29x coverage from *H. sapiens*, containing ultra-long reads, reaching lengths up to 340 kbp. Further details and accession numbers for all the datasets are given in Table 2. Details on the used reference sequences are given in Table 3.

Dataset	Number of reads	Average length	Error rate	Coverage	Accession
Simulated Pacific Biosciences data					
E. coli 30x	16,959	8,235	12.29	30x	N/A
E. coli 60x	33,918	8,211	12.28	60x	N/A
S. cerevisiae 30x	45,198	8,216	12.28	30x	N/A
S. cerevisiae 60x	90,397	8,204	12.29	60x	N/A
C. elegans 30x	366,416	8,204	12.28	30x	N/A
C. elegans 60x	732,832	8,220	12.28	60x	N/A
Real Oxford Nanopore data					
D. melanogaster	1,327,569	6,828	14.55	63x	SRX3676783
H. sapiens[1]	1,075,867	6,744	17.60	29x	PRJEB23027

Table 2. Description of the long reads datasets used in our experiments.
[1] Only reads from chromosome 1 were used.

Reference organism	Strain	Reference sequence	Size
E. coli	K-12 substr. MG1655	NC_000913	4.6 Mbp
S. cerevisiae	W303	scf7180000000{084-13}	12.2 Mbp
C. elegans	Bristol N2	GCA_000002985.3	100 Mbp
D. melanogaster	BDGP Release 6	ISO1 MT/dm6	144 Mbp
H. sapiens[1]	GRCh38	NC_000001.11	249 Mbp

Table 3. Description of the reference sequences used in our experiments.
[1] Only chromosome 1 was used.

3.2.2 Comparison on simulated data

To precisely assess the accuracy of the different correction methods, we first tested them on the simulated Pacific Biosciences datasets. LRCstats was used to evaluate the correction accuracy of each method. However, LRCstats did not scale to the *C. elegans* 60x dataset, requiring more than 4 days and 100 GB of RAM to compute the results. For this dataset, we thus only report throughput and error rate statistics, obtained by aligning the corrected reads to the reference with Minimap2. Correction statistics of all the aforementioned tools on the different datasets, along with their runtime and memory consumption, are given in Table 4. For methods having distinct, easily identifiable, steps for overlapping and correction (*i.e.* Daccord, MECAT and CONSENT), we additionally report runtime and memory consumption of these two processes apart. All the correction experiments were run on a computer equipped with 16 2.39 GHz cores and 32 GB of RAM.

Daccord clearly performed the best in terms of throughput and quality, outperforming all the other methods on the *E. coli* and the *S. cerevisiae* datasets. However, the overlapping step, relying on full alignment of the long reads against each other, consumed high amounts of memory, 3x to 11x more than CONSENT or MECAT mapping strategies. As a result, Daccord could not scale to the *C. elegans* datasets, DALIGNER reporting an error upon start. On the contrary, Canu displayed the highest error rates on all the datasets, but consumed relatively stable, low amounts of memory. In particular, on the two *C. elegans* datasets, it displayed the lower memory consumption among all the other methods.

MECAT performed the best in terms of runtime, outperforming all the other tools by several order of magnitudes on all the datasets. Its overlapping strategy was also highly efficient, and displayed the lowest memory consumption among the other strategies, on all the datasets. However, compared to Minimap2, the overlapping strategy adopted in CONSENT, MECAT's overlapping strategy displayed higher runtimes, raising according to both the size of the dataset and the genome complexity,

							Overlapping		Correction		Total	
Dataset	Corrector	Throughput (Mbp)	Error rate (%)	Deletions (%)	Insertions (%)	Substitutions (%)	Runtime	Memory (MB)	Runtime	Memory (MB)	Runtime	Memory (MB)
E. coli 30x	Original	140	12.2862	2.6447	8.7973	0.8442	N/A	N/A	N/A	N/A	N/A	N/A
	Canu	130	0.2508	0.0636	0.2001	0.0102	_	_	_	_	19min	4,613
	Daccord	**131**	**0.0219**	**0.0034**	**0.0090**	0.0115	1 min	6,813	13 min	639	14 min	6,813
	FLAS	130	0.2077	0.1490	0.0741	0.0043	_	_	_	_	12min	1,639
	MECAT	107	0.1649	0.1328	0.0459	**0.0018**	25 sec	**1,600**	**1 min 14 sec**	1,083	**1 min 39 sec**	**1,600**
	CONSENT	130	0.2013	0.0944	0.1095	0.0163	**22 sec**	2,390	16 min 48 sec	**132**	17 min 10 sec	2,390
E. coli 60x	Original	279	12.2788	2.6437	8.7919	0.8432	N/A	N/A	N/A	N/A	N/A	N/A
	Canu	219	0.5211	0.1390	0.4045	0.0243	_	_	_	_	24min	3,674
	Daccord	**261**	**0.0175**	**0.0026**	**0.0062**	0.0103	3 min	18,450	51 min	1,191	54 min	18,450
	FLAS	260	0.1039	0.0907	0.0220	0.0010	_	_	_	_	38min	2,428
	MECAT	233	0.1011	0.0896	0.0203	**0.0008**	**1 min**	**2,387**	**4 min**	1,553	**5 min**	**2,387**
	CONSENT	259	0.0590	0,0368	0.0241	0.0037	**1 min**	4,849	35 min	**248**	36 min	4,849
S. cerevisiae 30x	Original	371	12.283	2.646	8.7937	0.8434	N/A	N/A	N/A	N/A	N/A	N/A
	Canu	227	0.8472	0.2335	0.6393	0.0479	_	_	_	_	29min	3,681
	Daccord	**348**	**0.1186**	**0.0222**	**0.0368**	0.0707	7 min	31,798	1 h 12 min	3,487	1 h 19 min	31,798
	FLAS	345	0.2537	0.1863	0.0828	0.0088	_	_	_	_	29min	2,935
	MECAT	285	0.2111	0.1691	0.0574	**0.0048**	**1 min**	**2,907**	**4 min**	1,612	**5 min**	**2,907**
	CONSENT	345	0.2890	0.1428	0.1386	0.0348	**1 min**	5,523	45 min	**284**	46 min	5,523
S. cerevisiae 60x	Original	742	12.2886	2.6484	8.7963	0.8439	N/A	N/A	N/A	N/A	N/A	N/A
	Canu	600	0.5615	0.1518	0.4309	0.0292	_	_	_	_	1h11min	**3,710**
	Daccord	**696**	**0.0305**	**0.0055**	**0.0180**	0.0100	10 min	32,190	2 h 16 min	1,160	2 h 26 min	32,190
	FLAS	690	0.1430	0.1215	0.0319	0.0031	_	_	_	_	1h30min	4,984
	MECAT	617	0.1365	0.1189	0.0286	**0.0020**	4 min	**4,954**	**12 min**	2,412	**16 min**	4,954
	CONSENT	690	0.1418	0.0735	0.0650	0,0166	**2 min**	11,325	1 h 44 min	**535**	1 h 46 min	11,325
C. elegans 30x	Original	3,006	12.2806	2.6449	8.7926	0.8431	N/A	N/A	N/A	N/A	N/A	N/A
	Canu	2,776	0.2895	**0.0682**	0.2354	0.0126	_	_	_	_	9h09min	**6,921**
	Daccord	_	_	_	_	_	_	_	_	_	_	_
	FLAS	2,718	0.3862	0.2656	0.1469	0.0106	_	_	_	_	3h07min	10,565
	MECAT	2,085	**0.2682**	0.2135	**0.0764**	**0.0037**	27 min	**10,535**	**21 min**	2,603	**48 min**	10,535
	CONSENT	**2,791**	0.6300	0.3064	0.2958	0.0878	**15 min**	21,819	9 h 21 min	**1,885**	9 h 36 min	21,819
C. elegans 60x	Original	6,024	12.2825	2.6457	8.7937	0.8432	N/A	N/A	N/A	N/A	N/A	N/A
	Canu	5,119	0.6623	_	_	_	_	_	_	_	9 h 30 min	**7,050**
	Daccord	_	_	_	_	_	_	_	_	_	_	_
	FLAS	**5,614**	0.2160	_	_	_	_	_	_	_	10 h 45 min	13,682
	MECAT	4,941	**0.1882**	_	_	_	1 h 28 min	**10,563**	**1 h 15 min**	3,775	**2 h 43 min**	10,563
	CONSENT	5,607	0.4604	_	_	_	**57 min**	32,284	26 h 07 min	**3,390**	27 h 04 min	32,284

Table 4. LRCStats results on the simulated Pacific Biosciences datasets. Results for the C. elegans 60x datasets were obtained by aligning the reads to the reference, as LRCstats did not scale, and required more than 4 days and 100 GB of memory to compute the results. Only throughput and error rate are thus reported for this dataset. Daccord results are missing for the two C. elegans datasets, as DALIGNER failed to perform alignment, reporting an error upon start, even when ran on a cluster node with 28 2.4 GHz cores and 128 GB of RAM.

although it remained faster that Daccord's DALIGNER. Minimap2's memory consumption, however, was higher than that of MECAT's overlapping strategy, especially on the *C. elegans* datasets.

Compared to both FLAS and CONSENT, MECAT displayed lower throughputs on all the datasets. As for FLAS, this can be explained by the fact that it was designed as a MECAT wrapper, allowing to retrieve additional overlaps, and thus correct a greater number long reads. As a result, since it relies on MECAT's error correction strategy, it displayed highly similar memory consumption. Runtime was however higher, due to the additional steps allowing to retrieve new overlaps, and to the resulting higher number of reads to correct. Throughputs and error rates of FLAS and CONSENT were highly similar on all the datasets, also it slightly differed on the *C. elegans* datasets, were both the throughput and the error rate of CONSENT were higher, meaning that CONSENT attempted to correct additional, highly erroneous, long reads. Runtimes were also comparable on the *E. coli* and *S. cerevisiae* datasets. However, on the *C. elegans* datasets, CONSENT displayed higher runtimes. As for the memory consumption of the error correction step, CONSENT outperformed all the other methods on all the datasets.

3.2.3 Comparison on real data

We then evaluated the different correction methods on larger, real, Oxford Nanopore datasets. For these datasets, we not only evaluate how well the corrected long reads realign to the reference genome, but also how they assemble. For the alignment assessment, we report how many reads were corrected, their throughput, their N50, the proportion of corrected reads that could be aligned, the average identity of the alignments, as well as the genome coverage, that is, the percentage of bases of the reference genome to which at least a nucleotide aligned. For the assembly assessment, we report the overall number of contigs, the number of contigs that could be aligned, the NGA50 and NGA75, and, once again, the genome coverage. We aligned the long reads to the reference with Minimap2, and obtained statistics by parsing the output SAM file. We performed assemblies using Minimap2 and Miniasm (Li, 2015), and obtained statistics with QUAST-LG (Mikheenko *et al.*, 2018). Results are given in Table 5 for the alignment assessment, and in Table 6 for the assembly assessment. Runtimes and memory consumption of the different methods are also given in Table 5. As for the simulated data, we report runtime and memory consumption of the overlapping and correction steps apart, when possible. All the correction experiments were run on cluster node equipped with 28 2.39 GHz cores and 128 GB of RAM.

On these two datasets, Daccord failed to run, as DALIGNER could not perform alignment, for the same reason as for the simulated *C. elegans* 60x dataset. CONSENT corrected the largest number of reads, and reached the highest alignment identity on the two datasets. Its N50 was also higher than that of all the other methods, except for Canu on the *D. melanogaster* dataset. CONSENT also reached the highest throughput, and the largest genome coverage, for the two datasets. When it comes to runtime and memory consumption, MECAT once again outperformed all the other methods, as in the experiments on simulated data. Moreover, it reached the highest proportion of aligned reads, on the two datasets. CONSENT was however really close, as only 0.36-0.61% less reads could be aligned. Moreover, on the *H. sapiens* dataset, CONSENT was the only tool able to deal with ultra-long reads. Indeed, other methods reported errors when

Dataset	Corrector	Number of reads	Throughput (Mbp)	N50 (bp)	Aligned reads (%)	Alignment identity (%)	Genome coverage (%)	Overlapping		Correction		Total	
								Runtime	Memory (MB)	Runtime	Memory (MB)	Runtime	Memory (MB)
D. melanogaster	Original	1,327,569	9,064	11,853	85.52	85.43	98.47	N/A	N/A	N/A	N/A	N/A	N/A
	Canu	829,965	6,993	**12,694**	98.05	95.20	97.89	–	–	–	–	14 h 04 min	10,295
	Daccord	–	–	–	–	–	–	–	–	–	–	–	–
	FLAS	855,275	7,866	11,742	95.65	94.99	98.09	–	–	–	–	10 h 18 min	18,820
	MECAT	827,490	7,288	11,676	**99.87**	96.52	97.34	**28 min**	**13,443**	**1 h 26 min**	7,724	**1 h 54 min**	**13,443**
	CONSENT	**1,065,621**	**8,178**	12,297	99.26	**96.72**	**98.20**	43 min	51,361	37 h 17 min	**7,570**	38 h	51,361
H. sapiens	Original	1,075,867	7,256	10,568	88.24	82.40	92.46	N/A	N/A	N/A	N/A	N/A	N/A
	Canu	–	–	–	–	–	–	–	–	–	–	–	–
	Daccord	–	–	–	–	–	–	–	–	–	–	–	–
	FLAS[1]	670,708	5,695	10,198	99.06	91.00	92.37	–	–	–	–	4 h 57 min	14,957
	MECAT[1]	655,314	5,479	10,343	**99.95**	91.69	91.44	26 min	**11,075**	**1 h 27 min**	**4,591**	**1 h 53 min**	**11,075**
	CONSENT	**869,462**	**6,349**	**10,839**	99.59	**93.00**	**92.40**	**17 min**	45,869	8 h 13 min	5,759	8 h 30 min	45,869

Table 5. Statistics of the real long reads, before and after correction with the different methods.
[1] Reads longer than 50kbp were filtered out, as ultra-long reads caused the programs to stop with an error. There were 1,824 such reads in the original datasets, accounting for a total number of 135,364,312 bp.
Daccord could not be run on these two datasets, due to errors reported by DALIGNER.
Canu stopped with an error on the H. sapiens dataset, both with and without the long reads $>$ 50kbp.

attempting to correct the original dataset. As a result, in order for those methods to work, we had to remove the reads that were longer than 50kpb. There were 1,824 such reads, accounting for a total number of 135,364,312 bp. However, even after removing these ultra-long reads, Canu still failed to perform correction, and reported an error.

The assembly yielded from Canu corrected reads outperformed all other assemblies in terms of NGA50, NGA75, and genome coverage, on the *D. melanogaster* dataset. However, it produced a higher number of contigs than the assemblies yielded from FLAS and MECAT reads. The assembly obtained from CONSENT corrected reads contained the largest number of contigs, but outperformed the assemblies obtained from FLAS and MECAT reads in terms of genome coverage and NGA50. NGA75 of the CONSENT assembly was also larger than that of FLAS, but slightly shorter than that of MECAT. The error rate per 100 kbp of the CONSENT assembly was also lower than that of FLAS, and slightly higher than that of MECAT. On the *H. sapiens* dataset, the assembly obtained from CONSENT corrected reads outperformed the assemblies produced by both FLAS and MECAT corrected reads, in terms of number of contigs, NGA50, and NGA75. In particular, the NGA50 of the CONSENT assembly was more that 1 Mbp larger than that of other assemblies. However, 5 of the contigs of the CONSENT assembly could not be aligned to the reference, likely due to misassemblies by Miniasm. As a result, the assembly yielded from the CONSENT corrected reads covered 5% less of the reference sequence, and displayed a higher error rate per 100 kbp, compared to the assemblies obtained from FLAS and MECAT corrected reads. The fact that we cover a smallest proportion of the reference sequence gives us further room to improve CONSENT. Looking to the unaligned contigs more into details could indeed help us to further improve the mechanisms and principles of CONSENT.

3.3 Contig polishing

As an additional feature of CONSENT, we also allow to perform contig polishing. The process is pretty straightforward. Indeed, instead of computing overlaps between the long reads, as presented in the previous sections, the long reads used for the assembly are simply mapped to the assembled contigs. The rest of the pipeline remains the same. We present contig polishing results on the simulated *E. coli*, *S. cerevisiae*, and *C. elegans* 60x datasets, as well as on the real *D. melanogaster* and *H. sapiens* datasets, and compare to RACON (Vaser *et al.*, 2017), a state-of-the-art contig polishing method, on Table 7.

These results show that CONSENT outperformed RACON in terms of terms of quality of the results, especially dealing better with errors, and greatly reducing the error rate per 100 kbp, on the the *E. coli*, *S. cerevisiae*, and *C. elegans* datasets. On these three datasets, CONSENT was also 4x to 16x faster than RACON. Moreover, the NGA50 and NGA75 of CONSENT were highly similar to those of RACON on these datasets. RACON however covered a slightly larger proportion (0.12%) of the reference genome on the *C. elegans* dataset. For the larger, eukaryotic *D. melanogaster* dataset, RACON outperformed CONSENT in terms of error rate and runtime, although the NGA50, NGA75 and genome coverage of the two methods remained comparable. Moreover, after polishing with CONSENT, one more contig could be aligned to the reference, compared to RACON.

This contig polishing feature, being straightforward, raises the question as to how other correction methods should propose such an option. Moreover, it would be interesting to evaluate already published correction methods on their ability to polish contigs, at the expense of minimal modifications to their workflows.

4 Discussion and future works

Experimental results on the human dataset are particularly promising. Indeed, CONSENT is the only method able to scale to the ultra-long reads contained in this dataset. As such reads are expected to become more frequent in the future, being able to deal with them will soon become a necessity. In addition, judging from the memory consumption of the error correction step, it seems like CONSENT could easily scale to the error correction of a complete human dataset, and further experiments should therefore head in that direction.

However, the overlapping step, performed by Minimap2, tends to display higher memory consumption than the overlapping steps of other methods. In addition, the runtime of the correction step also tends to be higher. We discuss how to further reduce these computational costs below.

The memory consumption, for the overlapping step, can be explained by the fact that, in our experiments, we voluntarily set Minimap2 index size to 100 GB, so the index would not be split when processing large datasets. This can be explained by the fact that, in order to create the alignment piles, CONSENT needs all the alignments of a given read to directly follow each other in the Minimap2 output PAF file. Allowing to split the index would invalidate this property, as each part of the index would be processed separately by Minimap2, resulting in multiple, independent alignment files, that would then be concatenated into the final result file. As a result, if overlaps for a given read A are found in different parts of the index, the final result file would contain multiple, independent sets of lines concerning read A. Since these sets of lines would not directly follow each other, CONSENT would thus not be able to properly define

Dataset	Corrector	Contigs	Aligned contigs	NGA50	NGA75	Genome coverage	Errors / 100 kbp
D. melanogaster	Original	423	408	864,011	159,590	83.19	10,690
	Canu	410	381	**2,757,690**	**822,577**	**92.91**	1,897
	Daccord	–	–	–	–	–	–
	FLAS	374	361	1,123,351	364,884	92.05	2,689
	MECAT	**308**	**307**	1,425,566	478,877	92.03	**1,728**
	CONSENT	455	448	1,666,202	470,720	92.82	1,951
H. sapiens	Original	201	188	1,025,355	–	77.57	11,319
	Canu	–	–	–	–	–	–
	Daccord	–	–	–	–	–	–
	FLAS	237	**237**	1,698,601	289,968	94.97	4,404
	MECAT	249	247	1,672,967	424,788	**95.66**	**3,781**
	CONSENT	**182**	177	**2,663,412**	**439,178**	90.49	4,543

Table 6. Statistics of the assemblies obtained with the corrected long reads.
As previously mentioned, Daccord results on the two datasets, and Canu results on the H. sapiens dataset are absent, as the tools could not be run.
For the assembly of the original reads on the H. sapiens dataset, QUAST-LG did not provide a metric for the NGA75.

Dataset	Method	Contigs	Aligned contigs	NGA50	NGA75	Genome coverage	Errors / 100 kbp	Runtime (CPU seconds)	Memory (MB)
E. coli 60x	Original	**1**	**1**	4,939,014	4,939,014	99.91	10,721	N/A	N/A
	RACON	**1**	**1**	4,663,914	4,663,914	99.90	499	5,597	**643**
	CONSENT	**1**	**1**	4,637,939	4,637,939	**99.91**	**78**	**334**	1,651
S. cerevisiae 60x	Original	**29**	**29**	579,247	456,470	96.14	10,694	N/A	N/A
	RACON	**29**	**29**	539,472	346,116	96.09	637	14,931	**1,703**
	CONSENT	**29**	**29**	532,256	334,560	**96.15**	**212**	**1,737**	7,267
C. elegans 60x	Original	**47**	**47**	5,495,235	2,656,350	99.71	10,611	N/A	N/A
	RACON	**47**	**47**	5,073,456	2,349,027	**99.72**	819	136,325	**14,288**
	CONSENT	**47**	46	5,023,549	2,323,633	99.61	**336**	**35,215**	89,003
D. melanogaster	Original	**423**	408	864,011	159,590	83.19	10,690	N/A	N/A
	RACON	422	416	693,918	287,386	**92.88**	**973**	**197,124**	**19,508**
	CONSENT	422	**417**	668,829	232,778	92.48	2,113	270,186	90,041

Table 7. Results of the contig polishing experiment.

the alignment pile of read A. This issue could easily be addressed by using another, less memory consuming, tool for computing the overlaps. However, we could also keep making use of Minimap2, allow it to split its index, and then sort the final result file so that all alignments of a given read directly follow each other. Since alignment files are large, especially when processing large amounts of data, this would however impact the runtime of CONSENT. Another track to address this issue would be to rely on a strategy based on a PAF-index, that would allow us, for a given read, to retrieve the offsets of all the lines concerning the alignments of this read. This would thus allow us to easily navigate through the file, without needing it to be in a precise order, and without sorting it. More generally, CONSENT is designed as a modular tool which is not dependent on the choice of the aligner. It will thus benefit from the progress that will be made in alignment strategies, and will thus allow to propose better correction as the alignment methods evolve.

As for the runtime of the error correction step, our experiments show that it tends to rise according to the complexity of the genome. This can be explained by the highest proportion of repeated regions in more complex genomes. Such repeated regions indeed impact the alignment piles and windows coverages, and could therefore lead to the processing of windows having very deep coverages. These deep coverage windows would thus contain large number of sequences to process and align, which would greatly increase the runtime, even with our k-mer chaining strategy. For such deep coverage windows, we could use a validation strategy similar to that of HALC, that would allow us to only consider sequences from the window that actually come from the same genomic region as the long read we are attempting to correct. This would indeed allow us to reduce the coverage of the piles and of the windows, thus meaning less sequences to process during the multiple sequence alignments, and thus reduced runtimes. Moreover, further optimization of the parameters shall also be considered. In particular, the window size, and the minimum number of anchors to allow the processing of a window greatly impact the runtime. Running various experiments with different set of parameters would therefore allow us to find a satisfying compromise between runtime and quality of the results.

5 Conclusion

We presented CONSENT, a new self-correction method for long reads that combines different efficient strategies from the state-of-the-art. CONSENT starts by dividing overlapping regions of the long reads into smaller windows, in order to compute multiple sequence alignments, and consensus sequences of these windows. These multiple sequence alignments are performed using a method based on partial order graphs, allowing to perform actual multiple sequence alignment. This method is combined to an efficient k-mer chaining strategy, which allows to further divide the multiple sequence alignment into smaller instances, and thus reach greater speed. Once the consensus of a window from a matched region has been computed, it is further polished with the help of a local de Bruijn graph, in order to further reduce the final error rate, and is realigned to the read.

Our experiments show that CONSENT compares well, or even

outperforms other state-of-the-art methods in terms of quality of the results. In particular, CONSENT is the only method able to scale to a human dataset containing Oxford Nanopore ultra-long reads, reaching lengths up to 340 kbp. Although very recent, such reads are expected to further develop, and thus become more accessible in the near future. Being able to deal with them will thus soon become a necessity. CONSENT could therefore be the first self-correction method able to be applied to such ultra-long reads on a greater scale.

The contig polishing feature that was added to CONSENT also seems to offer promising results. As the processes of long reads correction and contig polishing are not so different from one another, one can wonder why more error correction tools do not offer this feature. It indeed seems to be affordable at the expense of minimal additional work, while providing satisfying results. We believe that CONSENT could open the doors to more error correction tools offering such a feature in the future.

The segmentation strategy introduced in CONSENT also shows that actual multiple sequence alignments techniques are applicable to long, noisy sequences. In addition to being useful for error correction, this could also be applied for in various other problems, such as during the consensus steps of assembly tools, for haplotyping, and for quantification problems. The literature about multiple sequence alignment is vast, but lacks application on noisy sequences. We believe that CONSENT could be a first work in that direction.

Acknowledgements

Part of this work was performed using computing resources of CRIANN (Normandy, France).

Funding

References

Bao, E. and Lan, L. (2017). HALC: High throughput algorithm for long read error correction. *BMC Bioinformatics*, **18**, 204.

Bao, E., Xie, F., Song, C., and Dandan, S. (2018). HALS: Fast and High Throughput Algorithm for PacBio Long Read Self-Correction. *RECOMB-SEQ 2018*.

Cao, M. D., Nguyen, S. H., Ganesamoorthy, D., Elliott, A. G., Cooper, M. A., and Coin, L. J. (2017). Scaffolding and completing genome assemblies in real-time with nanopore sequencing. *Nature Communications*, **8**, 14515.

Chin, C.-S., Alexander, D. H., Marks, P., Klammer, A. A., Drake, J., Heiner, C., Clum, A., Copeland, A., Huddleston, J., Eichler, E. E., Turner, S. W., and Korlach, J. (2013). Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nature Methods*, **10**, 563–569.

Choudhury, O., Chakrabarty, A., and Emrich, S. J. (2018). HECIL: A hybrid error correction algorithm for long reads with iterative learning. *Scientific Reports*, **8**(1), 1–9.

Firtina, C., Bar-joseph, Z., Alkan, C., and Cicek, A. E. (2018). Hercules: a profile HMM-based hybrid error correction algorithm for long reads. *Nucleic Acids Research*, **46**(21).

Haghshenas, E., Hach, F., Sahinalp, S. C., and Chauve, C. (2016). CoLoRMap: Correcting Long Reads by Mapping short reads. *Bioinformatics*, **32**, i545–i551.

Jain, M., Koren, S., Miga, K. H., Quick, J., Rand, A. C., Sasani, T. A., Tyson, J. R., Beggs, A. D., Dilthey, A. T., Fiddes, I. T., *et al.* (2018). Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature biotechnology*, **36**(4), 338.

Kamath, G. M., Shomorony, I., Xia, F., Courtade, T., and David, N. T. (2017). Hinge: long-read assembly achieves optimal repeat resolution. *Genome research*, pages gr–216465.

Kchouk, M. and Elloumi, M. (2018). An Error Correction and DeNovo Assembly Approach for Nanopore Reads Using Short Reads. *Current Bioinformatics*, **13**(3), 241–252.

Koren, S., Harhay, G. P., Smith, T. P. L., Bono, J. L., Harhay, D. M., Mcvey, S. D., Radune, D., Bergman, N. H., and Phillippy, A. M. (2013). Reducing assembly complexity of microbial genomes with single-molecule sequencing. *Genome Biology*, **14**(9), R101.

Koren, S., Walenz, B. P., Berlin, K., Miller, J. R., Bergman, N. H., and Phillippy, A. M. (2017). Canu: scalable and accurate long-read assembly via adaptive k -mer weighting and repeat separation. *Genome Research*, **27**, 722–736.

La, S., Haghshenas, E., and Chauve, C. (2017). LRCstats, a tool for evaluating long reads correction methods. *Bioinformatics*, **33**, 3652–3654.

Lee, C., Grasso, C., and Sharlow, M. F. (2002). Multiple sequence alignment using partial order graphs. *Bioinformatics*, **18**(3), 452–464.

Li, H. (2015). Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *arXiv*, **25**, 1–7.

Li, H. (2018). Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, **34**(18), 3094–3100.

Madoui, M.-A., Engelen, S., Cruaud, C., Belser, C., Bertrand, L., Alberti, A., Lemainque, A., Wincker, P., and Aury, J.-M. (2015). Genome assembly using Nanopore-guided long and error-free DNA reads. *BMC Genomics*, **16**, 327.

Miclotte, G., Heydari, M., Demeester, P., Rombauts, S., Van de Peer, Y., Audenaert, P., and Fostier, J. (2016). Jabba: hybrid error correction for long sequencing reads. *Algorithms for Molecular Biology*, **11**, 10.

Mikheenko, A., Prjibelski, A., Antipov, D., Saveliev, V., and Gurevich, A. (2018). Versatile genome assembly evaluation with QUAST-LG. *Bioinformatics*, **34**(13), i142–i150.

Morisse, P., Lecroq, T., and Lefebvre, A. (2018). Hybrid correction of highly noisy long reads using a variable-order de Bruijn graph. *Bioinformatics*, **34**(24), 4213–4222.

Myers, G. (2014). Efficient local alignment discovery amongst noisy long reads. In D. Brown and B. Morgenstern, editors, *Algorithms in Bioinformatics*, pages 52–67, Berlin, Heidelberg. Springer Berlin Heidelberg.

Patterson, M., Marschall, T., Pisanti, N., Van Iersel, L., Stougie, L., Klau, G. W., and Schönhuth, A. (2015). Whatshap: weighted haplotype assembly for future-generation sequencing reads. *Journal of Computational Biology*, **22**(6), 498–509.

Salmela, L. and Rivals, E. (2014). LoRDEC: Accurate and efficient long read error correction. *Bioinformatics*, **30**, 3506–3514.

Sedlazeck, F. J., Rescheneder, P., Smolka, M., Fang, H., Nattestad, M., Von Haeseler, A., and Schatz, M. C. (2018a). Accurate detection of complex structural variations using single-molecule sequencing. *Nature Methods*, **15**(6), 461–468.

Sedlazeck, F. J., Lee, H., Darby, C. A., and Schatz, M. C. (2018b). Piercing the dark matter: bioinformatics of long-range sequencing and mapping. *Nature Reviews Genetics*, page 1.

Stöcker, B. K., Köster, J., and Rahmann, S. (2016). SimLoRD: Simulation of Long Read Data. In *Bioinformatics*, volume 32, pages 2704–2706.

Tischler, G. and Myers, E. W. (2017). Non Hybrid Long Read Consensus Using Local De Bruijn Graph Assembly. *bioRxiv, doi: https://doi.org/10.1101/106252*.

Vaser, R., Sovic, I., Nagarajan, N., and Sikic, M. (2017). Fast and accurate de novo genome assembly from long uncorrected reads. *Genome Research*, **27**, gr.214270.116.

Wang, J. R., Holt, J., McMillan, L., and Jones, C. D. (2018). FMLRC: Hybrid long read error correction using an FM-index. *BMC Bioinformatics*, **19**(1), 1–11.

Xiao, C. L., Chen, Y., Xie, S. Q., Chen, K. N., Wang, Y., Han, Y., Luo, F., and Xie, Z. (2017). MECAT: Fast mapping, error correction, and de novo assembly for single-molecule sequencing reads. *Nature Methods*, **14**(11), 1072–1074.