Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Original Paper

Sequence Analysis

HALS: Fast and High Throughput Algorithm for PacBio Long Read Self-Correction

Ergude Bao[1,3], Fei Xie[2,†], Changjin Song[1], and Dandan Song[2,*]

[1]School of Software Engineering, Beijing Jiaotong University, Beijing 100044, China
[2]School of Computer Science, Beijing Institute of Technology, Beijing 100081, China and
[3]Department of Botany and Plant Sciences, University of California, Riverside, California 92521, USA

[*]To whom correspondence should be addressed.
[†]Joint first authors.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: The third generation PacBio long reads have greatly facilitated sequencing projects with very large read lengths, but they contain about 15% sequencing errors and need error correction. For the projects with long reads only, it is challenging to make correction with fast speed, and also challenging to correct a sufficient amount of read bases, *i.e.* to achieve high throughput self-correction. MECAT is currently the fastest self-correction algorithm, but its throughput is relatively small (Xiao *et al.*, 2017).
Results: Here we introduce HALS, a wrapper algorithm of MECAT, to achieve high throughput long read self-correction while keeping MECAT's fast speed. HALS finds additional alignments from MECAT prealigned long reads to improve the correction throughput, and removes misalignments for accuracy. In addition, HALS also uses the corrected long read regions to correct the uncorrected ones to further improve the throughput. In our performance tests on *E. coli*, *S. cerevisiae* and *A. thaliana* long reads, HALS can achieve 28.1-78.9% larger throughput than MECAT. Compared to the other existing self-correction algorithms, HALS is 8-229x faster, and its throughput is also 10.1-157.8% larger or comparable. The HALS corrected long reads can be assembled into contigs of 18.0-60.4% larger N50 sizes than MECAT.
Availability: The HALS software can be downloaded for free from this site: https://github.com/xief001/hals.
Contact: sdd@bit.edu.cn

1 Introduction

The third generation sequencing technology is advantageous over the second generation in its much larger read lengths (Eid *et al.*, 2009). As a representative of the third generation sequencing technology, the PacBio Single Molecule Real-Time (SMRT) technology can currently generate long reads of 5-15K base pairs on average with cost of \$0.4-0.8 per million base pairs (Rhoads and Au, 2015; Lee *et al.*, 2016). The long reads can overcome length limitation of the second generation short reads, and can thus facilitate assembling or analyzing complex genome regions such as GC islands and repeats in downstream. As a result, more and more sequencing projects have incorporated the long reads. The projects can be classified into two categories as below.

- *Long and short read hybrid projects* use long and short reads together (Philippe *et al.*, 2013; Chen *et al.*, 2014). In these projects, long reads of low to moderate coverage are used as a complement to short reads of relatively high coverage, to obtain higher quality results than the short reads alone.
- *Long read only projects* use long reads alone (Baker *et al.*, 2014; Chaisson *et al.*, 2015). Though more expensive, long reads of high coverage can be used without short reads, to guarantee high quality results while simplifying the sequencing workflow.

Nevertheless, the long reads contain about 15% sequencing errors dominated by insertions and deletions, so it is important to correct these errors. The error correction can be made in two levels: biological level by Circular Consensus Sequencing (CCS) technology and computational level by error correction algorithms (Eid *et al.*, 2009). The CCS technology has to reduce read lengths to a large extent for correction, so the error correction algorithms are usually in favor over the CCS. The current error correction algorithms can be classified into two categories as below.

- *Short read assisted correction algorithms* align the corresponding short reads from the same species to the long reads and correct them, so they are suitable for the long and short read hybrid projects. PBcR (Koren *et al.*, 2012), LSC (Au *et al.*, 2013), Proovread (Hackl *et al.*, 2014) and CoLoRMap (Haghshenas *et al.*, 2016) align the initial short reads to the long reads for correction, while ECTools[1] (Lee *et al.*, 2014), LoRDEC (Salmela and Rivals, 2014), Jabba (Miclotte *et al.*, 2016) and HALC (by ourselves Bao and Lan, 2017) align the long reads to a de Bruijn graph constructed or contigs assembled from the short reads for correction.
- *Self-correction algorithms* align the long reads to themselves and find multiple sequence alignments among the long reads to correct them, suitable for the long reads only projects. To make alignment, HGAP (Chin *et al.*, 2013) uses seeding k-mers among the long reads for dynamic programming, and FALCON (Chin *et al.*, 2016) also uses the k-mers but applies a fast dynamic programming technique. Canu (Koren *et al.*, 2017; Berlin *et al.*, 2015) weights and filters the k-mers to reduce load of the dynamic programming, and MECAT (Xiao *et al.*, 2017) takes into consideration correlation of the k-mers for the filtration. Differently, LoRMA (Salmela *et al.*, 2016) aligns the long reads to de Bruijn graphs constructed from themselves for correction.

For the self-correction algorithms, there are two challenges to address as below.

- It is challenging to make correction with *fast speed*, because it is time consuming to align the long reads of usually several millions to each other. HGAP, FALCON, Canu and MECAT all seek to address this challenge.
- It is also challenging to correct a sufficient amount of read bases, *i.e.* to achieve *high throughput* self-correction, because it is also difficult to align the long reads of 15% errors to each other. LoRMA seeks to address this challenge.

MECAT has successfully addressed the first challenge, and is currently the fastest self-correction algorithm (Xiao *et al.*, 2017). Nevertheless, its throughput is relatively small. This is not economical considering the higher cost of long reads compared to short reads, and may also affect quality of the downstream results (Lee *et al.*, 2014). In order to address both of the two challenges, here we propose HALS, a fast and high throughput algorithm for the long read self-correction. HALS is a wrapper algorithm of MECAT, to achieve high throughput while keeping MECAT's fast speed. HALS has two novelties as below.

- HALS finds additional alignments from MECAT prealigned long reads to improve the correction throughput, and removes misalignments for accuracy. For the prealigned long reads, those aligned with each other are likely to be from the same genome region, so are found and put into the same set. For two such sets, if there are shared long reads existing in both of them, it is possible all the long reads are from the same genome region, so additional alignments could be found between the two sets for correction; alternatively, it is possible the shared long reads are misaligned in one set, so misalignments could be removed from the set.
- It also uses the corrected long read regions to correct the uncorrected ones to further improve the throughput. A graph is constructed from alignments of the corrected long read regions, and context of the graph is examined to accurately align the uncorrected long read regions to the corrected ones for further correction.

[1] ECTools and HGAP, FALCON, Canu and MECAT below are essentially pipeline algorithms including both error correction and downstream assembly algorithms.

2 Methods

2.1 Overview

The HALS algorithm consists of two steps as below, corresponding to the two novelties above, respectively.

1. Correct with refined MECAT prealigned long reads.
   a. Align long reads with each other by MECAT, construct a string graph based on the alignments (Myers, 2005), and find maximal cliques with a modified Bron-Kerbosch algorithm in the graph (Eppstein and Strash, 2011; Eppstein *et al.*, 2010).
   b. For two maximal cliques with shared vertices, find additional alignments between the corresponding long reads or remove misalignments depending on number of the shared vertices. Make error correction with the refined alignments.
2. Correct uncorrected long read regions with corrected ones.
   a. Align the corrected long read regions with each other, and construct a second string graph based on the alignments.
   b. Align all the long reads obtained from step 1 to paths of the graph. For each long read region with multiple aligned paths, find the correct one by referring to not only alignment identities but also expected amounts of aligned long reads to the paths. Make error correction for the uncorrected long read regions with the alignments.

In step 1, we construct a string graph recording all alignment information by MECAT, find maximal cliques in the graph as sets containing long reads aligned with each other (step 1a), and use the maximal cliques to refine the alignments for error correction (step 1b). In step 2, we construct a second string graph recording alignment information among the corrected long read regions (step 2a), and align all the long reads to paths of the graph to correct the uncorrected regions (step 2b). Details of these steps are as below. Figure 1 shows an illustration of the algorithm.

2.2 Maximal cliques in string graph

We align the long reads with each other by MECAT. The parameter settings are the default of moderate alignment sensitivity and accuracy. This guarantees sparsity of the constructed string graph (see below).

Based on the alignment, we construct a string graph. Each vertex is constructed for one long read, and each edge is constructed between two vertices if the corresponding long reads are aligned with a sufficient overlap length. Transitive edges are not removed as the ordinary string graph (Myers, 2005), because all the alignment information should be kept. The constructed graph is a sparse graph with maximum vertex degree d, where d is the maximum number of candidate alignments for each long read, usually 100-200 controlled by the aligner. The graph's degeneracy is thus at most d. Sparsity of the graph is useful to guarantee small running time of the modified Bron-Kerbosch algorithm (see below).

In the graph, we find maximal cliques with a modified Bron-Kerbosch algorithm (Eppstein and Strash, 2011). In an arbitrary graph, the maximal clique finding problem is NP-hard, but because the string graph is a sparse graph with degeneracy at most d, the modified Bron-Kerbosch algorithm has polynomial time complexity $O(dn3^{d/3})$, where n is number of the long reads or vertices. Therefore, running time of the modified Bron-Kerbosch algorithm is guaranteed relatively small.

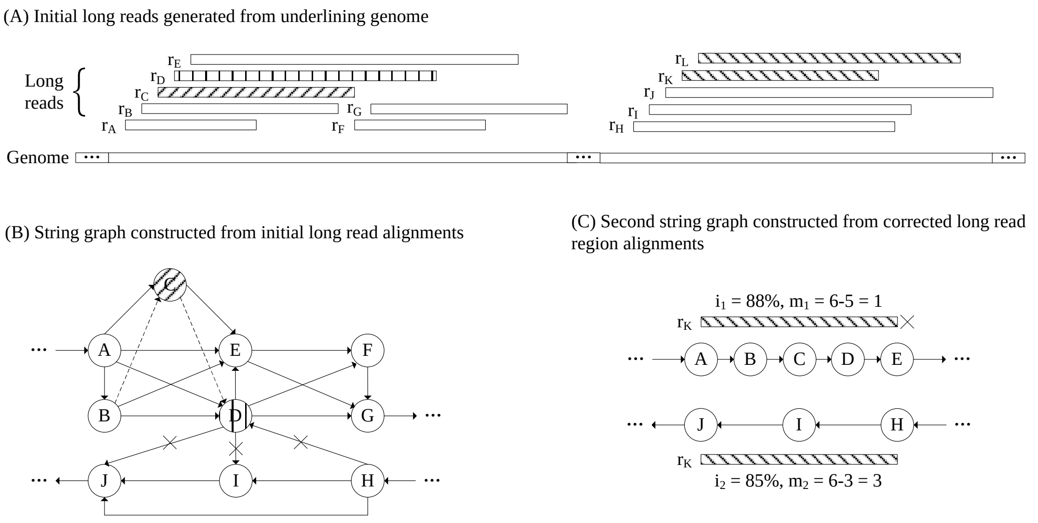
(A) Initial long reads generated from underlining genome
Long reads
rE
rD
rC
rB
rA
rG
rF
rL
rK
rJ
rI
rH
Genome
(B) String graph constructed from initial long read alignments
C
A
E
F
B
D
G
J
I
H
(C) Second string graph constructed from corrected long read region alignments
i1 = 88%, m1 = 6-5 = 1
rK
A
B
C
D
E
J
I
H
rK
i2 = 85%, m2 = 6-3 = 3

Fig. 1. Illustrations on the HALS algorithm. (A) Initial long reads r_A to r_L are generated from two similar genome regions in the underlining target genome. r_C (shaded) may not be fully corrected by MECAT, since it is only aligned to long reads r_A and r_E, and not aligned to r_B and r_D from the same genome region; r_D (shaded) may be miscorrected, since it is misaligned to long reads r_H, r_I and r_J from the other genome region; r_K and r_L (shaded) may not be corrected, since they are not aligned to any long read from the same genome region. (B) In algorithm step 1, a string graph is constructed from the initial long read alignments. In the graph, there are four maximal cliques: $C_1 = \{A, C, E\}$, $C_2 = \{A, B, D, E\}$, $C_3 = \{D, E, F, G\}$ and $C_4 = \{H, D, I, J\}$, where C_1 and C_2, C_2 and C_3, C_2 and C_4, and C_3 and C_4 are two cliques with shared vertices. C_1 and C_2 are classified in case (i), and additional alignments are obtained between r_C and r_B, as well as r_C and r_D (indicated with dashed lines). As a result, r_C can be fully corrected. C_2 and C_4 are classified in case (iii), and misalignments are removed between r_D and r_H, r_D and r_I, as well as r_D and r_J (indicated with crosses). As a result, r_D can be accurately corrected. (C) In algorithm step 2, a second string graph is constructed from alignments of the corrected long reads. With limited errors in paths of the graph, r_K could be aligned to two alternative paths: $P_1 = A \to E$ and $P_2 = H \to J$. Despite P_1's higher alignment identity $i_1 = 88\%$ than P_2's $i_2 = 85\%$, r_K is aligned to P_2 of much higher expected amount of aligned long reads $m_2 = 3$ than P_1's $m_1 = 1$. As a result, r_K can be corrected with accuracy. Similarly, r_L can also be corrected (not shown for simplicity).

2.3 Alignment refinement and error correction

Given two cliques C_1 of size c_1 and C_2 of size c_2 with s shared vertices existing in both cliques, there are basically three cases for them as below.

- Case (i): the corresponding long reads of C_1 and C_2 are from the same genome region, and the long reads of shared vertices are correctly aligned.
- Case (ii): the corresponding long reads of C_1 and C_2 are from two different genome regions, and the long reads of shared vertices span the genome regions and are correctly aligned.
- Case (iii): the corresponding long reads of C_1 and C_2 are from two different genome regions, and the long reads of shared vertices are from one of the regions and misaligned to the other.

In cases (i) and (ii), s should be relatively large compared to $\min\{c_1, c_2\}$, while in case (iii), s should be relatively small. This is based on the observation that among all alignments, number of the correct alignments is much larger than the misalignments. In addition, in case (ii), each long read of a shared vertex should have one region alignable to the long reads of C_1, and a different region to those of C_2.

Therefore, to distinguish between cases (i)/(ii) and (iii), we check s and $\min\{c_1, c_2\}$. If $s \geq \alpha_1 \min\{c_1, c_2\}$, C_1 and C_2 are classified in case (i)/(ii); otherwise, they are in case (iii), where α_1 is a fraction value with default setting 50%. To further distinguish between cases (i) and (ii), we check s and s', where s' is number of the shared vertices whose long reads have one region alignable to the long reads of C_1, and a different region to those of C_2. If $s' \geq \alpha_2 s$, C_1 and C_2 are classified in case (ii); otherwise, they are in case (i), where α_2 is a fraction value also with default setting 50%.

For any two cliques classified in case (i), we realign the corresponding long reads of the cliques to find more alignments. The parameter settings are *-a 1000 -n 200 -k 2* of high alignment sensitivity and relatively low accuracy. This could improve the error correction throughput. For any two cliques classified in case (iii), we calculate for each long read of a shared vertex, its average alignment identity to the long reads of each clique, and remove it from the clique of lower identity. This could improve the accuracy. After the refinement of alignments, we make error correction by MECAT with default parameter settings.

2.4 Correction of uncorrected regions with corrected regions

Similar to section 2.2, we align the corrected long read regions with each other, and construct a second string graph. Transitive edges are removed in the graph. Then we align all the long reads to paths of the graph for error correction. For the long reads containing both corrected and uncorrected regions, the corrected regions are aligned uniquely to paths in the graph, and the uncorrected regions could thus be aligned to paths in between for correction. For the long reads containing only uncorrected regions, they could also be aligned to paths of the graph for correction, because of the graph's limited errors.

It is possible that an uncorrected long read region has multiple paths for alignment. To find the correct path, we compare not only alignment identities, but also expected amounts of aligned long reads to the paths. The latter is a measurement of paths' capability to accommodate aligned long reads, and a long read has higher probability to align to the path of larger capability (see below). Given a long read region R with two aligned paths P_1 of identity i_1 and P_2 of identity i_2, if $|i_1 - i_2| \geq \beta_1$, we find the path of larger identity as the correct one, where β_1 is a difference value with default setting 5%. Otherwise, the expected amounts of aligned long reads to the paths m_1 and m_2 are calculated, respectively, and if $|\frac{m_1 - m_2}{m_1 + m_2}| \geq \beta_2$, we find the path of larger amount as the correct one, where β_2 is also a difference value with default setting 20%. For long read regions with more than two aligned paths, alignment identities and expected amounts of aligned long reads are compared in pairs to find the correct path.

Here, the expected amount of aligned long reads to a path P is calculated as the expected amount of long reads m_g from the path's genome region minus the amount of long reads m'_p forming the path. m'_p can be recorded when the graph is constructed, and m_g is calculated as below. Because the probability that a long read of length l is from a genome region of length Δ_g is $\frac{\Delta_g+2(l-o)}{G}$, m_g is calculated as $\sum_{i=0}^{n} \frac{\Delta_g+2(l_i-o)}{G}$, where o is minimum required overlap length between the long read and genome region, G is size of the genome, l_i is length of the ith long read, and again, n is number of the long reads. Furthermore, because Δ_g is about P's length Δ_p, and G can be estimated as $\frac{\sum_{j=0}^{n} l_j}{c}$, m_g is further refined as $\sum_{i=0}^{n}\sum_{j=0}^{n} \frac{\Delta_p+2(l_i-o)}{l_j/c}$, where c is coverage of the long reads. Note that this calculation assumes a path does not correspond to multiple genome regions. This is reasonable, since one alternatively aligned path of a long read region represents a single genome region in most cases in string graph (Myers, 2005).

2.5 Implementation of the HALS software

The HALS software is implemented in C++ for linux platform. It contains data structures as below.

- An n item adjacency list for the string graph in algorithm step 1. Its lists have the maximum length d, because of the graph's sparsity.
- An n item hash table for the read-clique mapping. It is used to find all the cliques with shared vertices.
- A $O((n-d)3^{d/3})$ item hash table for the clique-read mapping, where $O((n-d)3^{d/3})$ is the maximum possible number of cliques in the graph (Eppstein and Strash, 2011). It is used to find all the shared vertices for any two cliques.
- An n' item adjacency list for the second string graph in algorithm step 2, where n' is number of the corrected long read regions.

Input of HALS is the initial long reads, and the outputs include the error corrected (i) full long reads, (ii) trimmed long reads that do not contain the uncorrected regions in read heads and tails, and (iii) split long reads that do not contain the uncorrected regions and very short corrected regions (<500 bp; Salmela and Rivals (2014)).

3 Evaluation

3.1 Experimental design

3.1.1 Test of error correction performance

To evaluate HALS's performance, we ran HALS, MECAT and the other existing self-correction algorithms HGAP, LoRMA, FALCON and Canu on three sets of long reads from species: *E. coli*, *S. cerevisiae*, *A. thaliana* of genome sizes 5M bp, 12M bp and 125M bp, respectively. Coverage of the long reads was all 100x, meeting the requirement of most long read only projects. We also ran some typical short read assisted correction algorithms PBcR, CoLoRMap and ECTools on the long reads, together with the corresponding short reads from the same species. Coverage of the short reads was 50x, 38x and 33x, respectively, meeting the requirement of most long and short read hybrid projects. After error correction, we aligned the corrected long reads to the corresponding target genomes, to access quality of the long reads. Note that not all the existing short read assisted correction algorithms were compared, because they are in a different category from HALS.

3.1.2 Test with various long read coverage and parameter settings

HALS's performance is mainly affected by coverage of the long reads, because it is inputted with long reads alone. Hence, it is interesting to see HALS's performance with various long read coverage, and especially, the performance change compared to MECAT and the other existing self-correction algorithms. To do this, we varied coverage of the *E. coli* long reads from 50x to 200x, and ran and compared HALS, MECAT, HGAP, LoRMA, FALCON and Canu.

In addition, HALS's performance could also be affected by its parameter settings. Specifically, the parameters include the two fraction values α_1 and α_2 to distinguish between cases (i)-(iii) as discussed in section 2.3, and the two difference values β_1 and β_2 to find the correct aligned paths as discussed in section 2.4. It is also interesting to see HALS's performance with various settings of the parameters. To do this, we varied α_1 and α_2 from 30% to 70%, varied β_1 from 1% to 15%, and varied β_2 from 10% to 40%, and ran HALS on the *E. coli* long reads of 100x. Similar to above, after error correction, we aligned the corrected long reads to the corresponding target genome, to access quality of the long reads.

3.1.3 Test of long read assemblies

HALS's performance influences downstream assembly and analysis as discussed in section 1. Hence, it is interesting to see some downstream assembly results from the HALS corrected long reads. To do this, we used FALCON's assembly algorithm FALCON*[2] (Chin *et al.*, 2016) and Canu's assembly algorithm Canu* (Koren *et al.*, 2017), and assembled the corrected *S. cerevisiae* and *A. thaliana* long reads. To make comparison, we also assembled the MECAT corrected long reads. Then we aligned the assembled contigs to the corresponding target genomes, to access quality of the contigs. Note that this test is much simplified, because this study focuses on long read error correction rather than assembly. Also note that MECAT does not have a standalone assembly algorithm, but applies Canu's assembly algorithm.

All of the algorithms' software was in default settings, except the test of various parameter settings as discussed in section 3.1.2. Only the corrected split long reads were compared and used for assemblies in these tests, so that the results could not be affected by the uncorrected bases. All experiments were performed in a computing node of a computer cluster with 32 cores of 2.3 GHz and 1,024 GB memory.

3.2 Data sets and performance measurements

3.2.1 Data sets

The genomic long reads of *E. coli*, *S. cerevisiae* and *A. thaliana* were downloaded from NCBI accessions SRX1155577, ERX1725434 and SRX533607, respectively. The corresponding target genomes were from NCBI accessions NC_000913.3, NC_001133.9 and the Ensembl Plant FTP, respectively. The corresponding short reads were from NCBI accessions ERR022075, SRR567755 and ERR469286, respectively.

3.2.2 Performance measurements

We aligned the error corrected split long reads to the corresponding target genomes to access their quality. The BWA-MEM aligner was used for these alignments, because it is a typical aligner for genomic sequences with fast speed and high sensitivity (Li and Durbin, 2010). We made the following measurements: (i) *throughput* (TH) is the number of corrected and outputted bases, and *throughput ratio* (THR) is the throughput over the total number of initial long read bases (throughput ratio of the initial long reads is 100%); (ii) *alignment ratio* is the number of aligned bases over the total number of outputted bases; (iii) *alignment identity* is the identity of aligned bases; (iv) *genome fraction* is the number of genome bases aligned by long reads over the total number of genome bases.

[2] FALCON's assembly algorithm and Canu's assembly algorithm below are represented as FALCON* and Canu*, respectively, to distinguish from the error correction algorithms.

Table 1. Evaluation of error correction performance. The long reads in tests (a)-(c) are from E. coli, S. cerevisiae and A. thaliana, respectively. The error corrected long reads by MECAT and HALS (below dashed line), and the other existing self-correction algorithms HGAP, LoRMA, FALCON and Canu (above dashed line), and by short read assisted correction algorithms PBcR, CoLoRMap and ECTools (in gray) using additional short reads are compared in each test. The performance measurements are listed in section 3.2.2.

Method	Throughput (bp)	Alignment ratio	Alignment identity	Genome fraction	Sensitivity	Gain	Specificity	Time (h)	Memory (GB)
				(a) Long reads of E. coli					
Initial	513,788,653	47.8%	86.4%	100%	-	-	-	-	-
PBcR[1]	70,389,045	99.9%	99.9%	88.5%	-	-	-	161.7	5.4
CoLoRMap	144,888,400	99.2%	99.9%	91.7%	28.1%	27.5%	99.9%	2.3	5.4
ECTools	59,085,695	99.6%	99.6%	92.9%	12.3%	11.1%	99.9%	44.6	6.4
HGAP	96,592,267	97.8%	98.2%	99.3%	15.0%	12.6%	99.6%	59.7	183.9
LoRMA	52,406,443	98.7%	98.9%	29.2%	8.7%	8.1%	99.9%	13.6	15.6
FALCON[1]	131,529,895	98.0%	98.5%	98.5%	-	-	-	4.0	53.6
Canu	55,589,500	97.7%	97.3%	96.8%	8.6%	6.9%	99.7%	62.4	87.0
MECAT	75,526,932	97.5%	98.0%	97.8%	12.1%	9.9%	99.6%	0.1	3.8
HALS	135,126,416	97.7%	98.0%	98.0%	22.1%	18.5%	99.3%	0.5	5.9
				(b) Long reads of S. cerevisiae					
Initial	1,186,480,653	52.7%	87.0%	100%	-	-	-	-	-
PBcR[1]	195,769,308	99.9%	99.9%	96.1%	-	-	-	144.0	10.7
CoLoRMap	352,384,754	98.6%	99.8%	99.7%	29.4%	28.0%	99.8%	10.3	8.6
ECTools	268,114,628	99.7%	99.6%	98.5%	18.2%	16.8%	99.8%	1,333.6	5.1
HGAP	225,431,324	99.2%	99.2%	99.9%	18.2%	16.8%	99.8%	39.3	26.8
LoRMA	164,920,811	99.9%	99.9%	42.7%	16.1%	14.5%	99.8%	30.9	15.6
FALCON[1]	365,436,041	99.5%	99.8%	99.9%	-	-	-	30.2	80.5
Canu	251,533,898	99.1%	99.0%	99.9%	20.1%	18.5%	99.8%	137.6	87.0
MECAT	196,955,788	99.0%	99.2%	99.8%	16.1%	14.5%	99.1%	0.1	7.0
HALS	320,349,776	99.1%	99.3%	99.2%	26.4%	24.2%	99.7%	0.6	23.1
				(c) Long reads of A. thaliana					
Initial	12,514,544,644	34.9%	83.3%	99.9%	-	-	-	-	-
PBcR[1]	3,053,547,766	96.2%	96.5%	97.1%	-	-	-	1,472.9	19.3
CoLoRMap	4,555,294,250	96.8%	96.3%	99.2%	35.1%	27.6%	98.2%	197.5	18.2
ECTools	1,001,003,572	95.8%	95.5%	98.1%	7.8%	6.5%	99.8%	87,321.6	19.3
FALCON[1]	4,355,061,536	91.1%	90.6%	99.4%	-	-	-	1,214.4	102.5
Canu	3,992,139,741	91.0%	90.2%	99.7%	24.4%	16.6%	98.5%	175.2	182.0
MECAT	3,741,848,849	91.1%	90.6%	99.7%	24.4%	16.7%	98.7%	0.8	21.5
HALS	4,793,070,599	91.0%	90.4%	99.7%	30.7%	22.2%	98.3%	17.9	83.1

[1] Some measurements are not available without the correspondence information between the initial long reads and corrected ones.

Then we used the Error Correction Evaluation Toolkit for split long reads implemented by ourselves (Bao and Lan, 2017), and obtained in the outputted bases, the number of corrected errors (true positive or TP), the number of falsely converted correct bases (false positive or FP), the number of uncorrected errors (false negative or FN), and the number of unconverted correct bases (true negative or TN). With these numbers, due to the errors' uniform distribution in long reads, we can estimate the total number of errors in the initial long reads as the number of errors in the outputted bases over the throughput ratio, *i.e.* $EI = \frac{TP+FN}{THR}$. We can also estimate the total number of correct bases in the initial long reads as the number of correct bases in the outputted bases over the throughput ratio, *i.e.* $CI = \frac{TN+FP}{THR}$, and thus the number of correct bases in the discarded bases as the total number of correct bases in the initial long reads minus the number of correct bases in the outputted bases, *i.e.* $CD = CI - (TN + FP)$. Therefore, we made the following measurements: (v) *sensitivity* is calculated as $\frac{TP}{EI}$; (vi) *specificity* is calculated as $\frac{TN+CD}{CI}$; (vii) *gain* is the number of errors effectively corrected without introducing new ones over the total number of errors in the initial long reads, and calculated as $\frac{TP-FP}{EI}$.

In addition, we also aligned the assembled contigs to the corresponding genomes. Following Chin *et al.* (2016), we used the QUAST toolkit (Gurevich *et al.*, 2013), and made the following measurements: (viii) *assembly size* is the total length of contigs; (ix) *N contigs* is the number of contigs; (x) *N50 size* is the contig size at 50% of the total number of contig bases; (xi) *N N50* is the number of contigs of lengths larger than the N50 size; (xii) *Max contig size* is the maximum contig size.

3.3 Results

3.3.1 Results of error correction performance

Results on the *E. coli* long reads are listed in Table 1(a). (i) For the initial uncorrected long reads of 514M bp, the alignment ratio and alignment identity are 47.8% and 86.4%, respectively, indicating a large amount of errors. (ii) After error correction, the throughput of MECAT is 76M bp, and the alignment ratio and alignment identity are about 100%, while the throughput of HALS is 135M bp, and the alignment ratio and alignment identity are also about 100%. The throughput of HALS is 78.9% larger than MECAT. (iii) The running time and memory usage of the other existing self-correction algorithms HGAP, LoRMA, FALCON and Canu are 4.0-62.4 hours and 15.6-183.9 GB, respectively, while those of HALS are 0.5 hours and 5.9 GB, respectively. HALS is 8-125x faster than all of them. In addition, the throughput of these algorithms is 52-131M bp, and that of HALS is also 39.9-157.8% larger than HGAP, LoRMA and Canu, and comparable to FALCON. (iv) As to the existing short read assisted correction algorithms PBcR, CoLoRMap and ECTools, HALS is 5-323x faster than all of them, and the throughput of HALS is 92.0-128.7% larger than PBcR and ECTools, even though the latter were

Table 2. Evaluation with various coverage of the E. coli long reads. The coverage in tests (a)-(d) is 50x, 100x, 150x and 200x, respectively. The error corrected long reads by MECAT and HALS (below dashed line), and the other existing self-correction algorithms HGAP, LoRMA, FALCON and Canu (above dashed line) are compared in each test. The performance measurements are listed in section 3.2.2.

Method	Throughput (bp)	Alignment ratio	Alignment identity	Time (h)	Memory (GB)
(a) Long reads of 50x coverage					
Initial	256,894,327	48.0%	86.5%	-	-
HGAP	43,415,141	97.8%	98.0%	57.5	163.1
LoRMA	2,568,943	98.9%	99.2%	10.3	14.5
FALCON	29,029,059	98.0%	98.4%	1.1	53.6
Canu	35,118,079	98.2%	97.6%	60.8	87.0
MECAT	16,184,343	97.3%	97.8%	0.1	2.1
HALS	53,434,020	97.4%	97.3%	0.2	5.9
(b) Long reads of 100x coverage					
Initial	513,788,653	47.8%	86.4%	-	-
HGAP	96,592,267	97.8%	98.2%	59.7	183.9
LoRMA	52,406,443	98.7%	98.9%	13.6	15.6
FALCON	131,529,895	98.0%	98.5%	4.0	53.6
Canu	55,589,500	97.7%	97.3%	62.4	87.0
MECAT	75,526,932	97.5%	98.0%	0.1	3.8
HALS	135,126,416	97.7%	98.0%	0.5	5.9
(c) Long reads of 150x coverage					
Initial	770,682,980	47.7%	86.1%	-	-
HGAP	133,328,155	97.8%	98.1%	85.7	203.8
LoRMA	167,238,207	98.8%	98.9%	16.8	16.1
FALCON	237,370,358	98.4%	98.6%	8.5	56.5
Canu	101,318,583	98.2%	97.6%	63.2	87.0
MECAT	134,098,838	97.6%	98.0%	0.1	7.5
HALS	217,332,600	97.8%	98.1%	0.9	7.5
(d) Long reads of 200x coverage					
Initial	1,027,577,306	48.1%	86.1%	-	-
HGAP	172,632,987	97.8%	98.1%	158.2	214.6
LoRMA	265,114,945	98.8%	98.9%	30.8	22.9
FALCON	313,411,078	98.7%	98.4%	31.7	56.9
Canu	111,737,700	98.3%	97.6%	63.2	87.0
MECAT	205,515,461	97.7%	98.1%	0.3	8.0
HALS	311,355,924	97.8%	98.2%	3.9	8.0

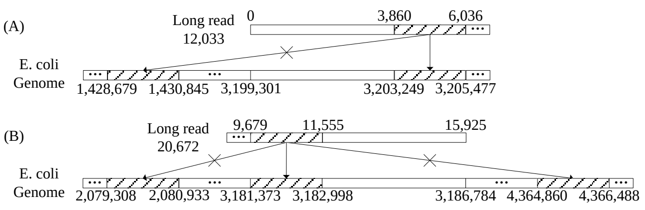

Fig. 2. Illustrations on HALS's function in finding correct paths on the E. coli long reads. (A) A long read region in the 12,033th long read from target genome region [3,203,249; 3,205,477] could be aligned to two paths of genome regions [1,428,679; 1,430,845] and [3,203,249; 3,205,477] (shaded), and HALS can find the correct path of genome region [3,203,249; 3,205,477] for it. (B) A long read region in the 20,672th long read from target genome region [3,181,373; 3,182,998] could be aligned to three paths of genome regions [2,079,308; 2,080,933], [3,181,373; 3,182,998] and [4,364,860; 4,366,488] (shaded), and HALS can also find the correct path of genome region [3,181,373; 3,182,998] for it.

Table 3. Evaluation of long read assemblies. The contigs in tests (a)-(b) are assembled from S. cerevisiae and A. thaliana long reads, respectively. The long reads are error corrected by MECAT and HALS in each test, and the contigs are assembled by FALCON* and Canu* from each set of corrected long reads. The performance measurements are listed in section 3.2.2.

Method	Assembly size (bp)	N Contigs	N50 size (bp)	N N50	Max contig size (bp)
(a) Contigs of S. cerevisiae					
MECAT+FALCON*	11,573,511	132	144,568	28	341,847
MECAT+Canu*	11,990,018	176	106,985	33	448,717
HALS+FALCON*	11,627,851	98	199,013	20	581,831
HALS+Canu*	12,261,578	118	171,594	20	545,493
(b) Contigs of A. thaliana					
MECAT+FALCON*	121,861,164	205	3,446,298	10	10,842,197
MECAT+Canu*	121,429,989	184	4,229,580	8	12,141,698
HALS+FALCON*	124,464,586	348	4,065,882	8	14,324,148
HALS+Canu*	122,645,145	272	5,837,812	7	13,695,512

inputted with additional 33-50x short reads. Note that for PBcR and FALCON, the sensitivity, gain and specificity are not available without the correspondence information between the initial long reads and corrected ones.

Results on the *S. cerevisiae* long reads are listed in Table 1(b). The throughput of HALS is 62.7% larger than MECAT, and HALS is 50-229x faster than the other existing self-correction algorithms HGAP, LoRMA, FALCON and Canu. In addition, the throughput of HALS is also 27.4-94.2% larger than HGAP, LoRMA and Canu. Results on the *A. thaliana* long reads are listed in Table 1(c). The throughput of HALS is 28.1% larger than MECAT, and HALS is 10-68x faster than the other existing self-correction algorithms FALCON and Canu. In addition, the throughput of HALS is also 10.1-20.1% larger than FALCON and Canu. Note that results of HGAP and LoRMA on the *A. thaliana* long reads are not shown, because they were designed for relatively small genomes and could not finish processing on the larger genome. These results indicate HALS can achieve high throughput while keeping MECAT's fast speed.

3.3.2 Results with various long read coverage and parameter settings

Results with various coverage of the *E. coli* long reads are listed in Table 2. For all the algorithms, the throughput and running time increase together with the coverage. For all the coverage, the throughput of HALS is 51.5-230.2% larger than MECAT, and HALS is 6-304x faster than the other existing self-correction algorithms. In addition, the throughput of HALS is also 17.4-1980.0% larger than HGAP, LoRMA and Canu, and in general comparable to FALCON. These results indicate HALS is scalable and efficient with various long read coverage.

Results with various parameter settings are not shown, because the difference of the results is relatively small and within $<5\%$. The difference with various α_1 and α_2 is small, because their influence is within algorithm step 1b, while the remaining algorithm steps 2a and 2b are complementary to this step. The difference with various β_1 and β_2 is also small, because it is not a dominating situation to align one long read region to multiple paths of the string graph in algorithm step 2b. However, this does not downgrade the importance to decide the correct paths in this step. As shown in Figure 2, on the *E. coli* long reads, a long read region in the 12,033th long read from target genome region [3,203,249; 3,205,477] could be aligned to two paths of genome regions [1,428,679; 1,430,845] and [3,203,249; 3,205,477], and HALS can find the correct path of genome region [3,203,249; 3,205,477] for it. In addition, a long read region in the 20,672th long read from target genome region [3,181,373; 3,182,998] could be aligned to three paths of genome regions [2,079,308; 2,080,933], [3,181,373; 3,182,998] and [4,364,860; 4,366,488], and HALS can also find the correct path of genome region [3,181,373; 3,182,998] for it.

3.3.3 Results of long read assemblies

Results on the *S. cerevisiae* long reads are listed in Table 3(a). The MECAT corrected long reads are assembled into 132-176 contigs of N50 sizes 106,985-144,568 bp and max contig sizes 341,847-448,717 bp. The HALS corrected long reads are assembled into 98-118 contigs of N50 sizes 171,594-199,013 bp and max contig sizes 545,493-581,831 bp. The N50 sizes with HALS are 37.7-60.4% larger than MECAT, and the max contig sizes are also 21.6-70.2% larger.

Results on the *A. thaliana* long reads are listed in Table 3(b). The N50 sizes of contigs assembled from the HALS corrected long reads are 18.0-38.0% larger than MECAT, and the max contig sizes are also 12.8-32.1% larger. Interestingly, the number of contigs with HALS is also a little larger than MECAT. This is probably because more alleles could be assembled from the HALS corrected long reads. These results indicate HALS corrected long reads can be assembled into high quality contigs.

4 Conclusions

This study introduces HALS, a wrapper algorithm of MECAT, to achieve high throughput long read self-correction while keeping MECAT's fast speed. HALS finds maximal cliques in a string graph constructed from MECAT prealigned long reads, and refines the alignments for correction. HALS also aligns the long reads to paths of a second string graph constructed from the corrected long read regions for further correction, considering not only alignment identities but also expected amounts of aligned long reads to the paths. As a result, HALS can achieve 28.1-230.2% larger throughput than MECAT, and is also 8-119x faster than the other existing self-correction algorithms. In addition, HALS can also achieve 17.4-157.8% larger or comparable throughput to the other existing self-correction algorithms. The HALS corrected long reads can be assembled into contigs of 18.0-60.4% larger N50 sizes than MECAT. In the future, we will expand this work in the following two directions. (1) Scale of HALS will be improved to handle long reads from genomes of $\geq$1G bp, *e.g.* human genome. (2) HALS will be adapted to process long reads of different characteristics from other platforms, *e.g.* Oxford Nanopore platform.

Acknowledgements

We thank Lingxiao Lan in our group, for implementing the Error Correction Evaluation Toolkit for split long reads. We acknowledge the support of the core facilities at the Institute for Integrative Genome Biology (IIGB), the University of California, Riverside.

Funding

This work was supported by grants from the National Science Foundation of China [61502027 to E.B.], and the Fundamental Research Funds for the Central Universities [2015RC045 to E.B.].

References

Au, K. F., Sebastiano, V., Afshar, P. T., Durruthy, J. D., Lee, L., Williams, B. A., van Bakel, H., Schadt, E. E., Reijo-Pera, R. A., Underwood, J. G., *et al.* (2013). Characterization of the human esc transcriptome by hybrid sequencing. *Proceedings of the National Academy of Sciences*, **110**(50), E4821–E4830.

Baker, K. S., Mather, A. E., McGregor, H., Coupland, P., Langridge, G. C., Day, M., Deheer-Graham, A., Parkhill, J., Russell, J. E., and Thomson, N. R. (2014). The extant world war 1 dysentery bacillus nctc1: a genomic analysis. *The Lancet*, **384**(9955), 1691–1697.

Bao, E. and Lan, L. (2017). Halc: High throughput algorithm for long read error correction. *BMC bioinformatics*, **18**(1), 204.

Berlin, K., Koren, S., Chin, C.-S., Drake, J. P., Landolin, J. M., and Phillippy, A. M. (2015). Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology*, **33**(6), 623–630.

Chaisson, M. J., Huddleston, J., Dennis, M. Y., Sudmant, P. H., Malig, M., Hormozdiari, F., Antonacci, F., Surti, U., Sandstrom, R., Boitano, M., *et al.* (2015). Resolving the complexity of the human genome using single-molecule sequencing. *Nature*, **517**(7536), 608–611.

Chen, X., Bracht, J. R., Goldman, A. D., Dolzhenko, E., Clay, D. M., Swart, E. C., Perlman, D. H., Doak, T. G., Stuart, A., Amemiya, C. T., *et al.* (2014). The architecture of a scrambled genome reveals massive levels of genomic rearrangement during development. *Cell*, **158**(5), 1187–1198.

Chin, C.-S., Alexander, D. H., Marks, P., Klammer, A. A., Drake, J., Heiner, C., Clum, A., Copeland, A., Huddleston, J., Eichler, E. E., *et al.* (2013). Nonhybrid, finished microbial genome assemblies from long-read smrt sequencing data. *Nature methods*, **10**(6), 563–569.

Chin, C.-S., Peluso, P., Sedlazeck, F. J., Nattestad, M., Concepcion, G. T., Clum, A., Dunn, C., OâĹ™Malley, R., Figueroa-Balderas, R., Morales-Cruz, A., *et al.* (2016). Phased diploid genome assembly with single molecule real-time sequencing. *Nature methods*, **13**(12), 1050.

Eid, J., Fehr, A., Gray, J., Luong, K., Lyle, J., Otto, G., Peluso, P., Rank, D., Baybayan, P., Bettman, B., *et al.* (2009). Real-time dna sequencing from single polymerase molecules. *Science*, **323**(5910), 133–138.

Eppstein, D. and Strash, D. (2011). Listing all maximal cliques in large sparse real-world graphs. *Experimental Algorithms*, pages 364–375.

Eppstein, D., Löffler, M., and Strash, D. (2010). Listing all maximal cliques in sparse graphs in near-optimal time. In *International Symposium on Algorithms and Computation*, pages 403–414. Springer.

Gurevich, A., Saveliev, V., Vyahhi, N., and Tesler, G. (2013). Quast: quality assessment tool for genome assemblies. *Bioinformatics*, page btt086.

Hackl, T., Hedrich, R., Schultz, J., and Förster, F. (2014). proovread: large-scale high-accuracy pacbio correction through iterative short read consensus. *Bioinformatics*, page btu392.

Haghshenas, E., Hach, F., Sahinalp, S. C., and Chauve, C. (2016). Colormap: Correcting long reads by mapping short reads. *Bioinformatics*, **32**(17), i545–i551.

Koren, S., Schatz, M. C., Walenz, B. P., Martin, J., Howard, J. T., Ganapathy, G., Wang, Z., Rasko, D. A., McCombie, W. R., Jarvis, E. D., *et al.* (2012). Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nature biotechnology*, **30**(7), 693–700.

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**(5), 722–736.

Lee, H., Gurtowski, J., Yoo, S., Marcus, S., McCombie, W. R., and Schatz, M. (2014). Error correction and assembly complexity of single molecule sequencing reads. *BioRxiv*, page 006395.

Lee, H., Gurtowski, J., Yoo, S., Nattestad, M., Marcus, S., Goodwin, S., McCombie, W. R., and Schatz, M. (2016). Third-generation sequencing and the future of genomics. *bioRxiv*, page 048603.

Li, H. and Durbin, R. (2010). Fast and accurate long-read alignment with burrows–wheeler transform. *Bioinformatics*, **26**(5), 589–595.

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**(1), 1.

Myers, E. W. (2005). The fragment assembly string graph. *Bioinformatics*, **21**(suppl_2), ii79–ii85.

Philippe, N., Legendre, M., Doutre, G., Couté, Y., Poirot, O., Lescot, M., Arslan, D., Seltzer, V., Bertaux, L., Bruley, C., *et al.* (2013). Pandoraviruses: amoeba viruses with genomes up to 2.5 mb reaching that of parasitic eukaryotes. *Science*, **341**(6143), 281–286.

Rhoads, A. and Au, K. F. (2015). Pacbio sequencing and its applications. *Genomics, proteomics & bioinformatics*, **13**(5), 278–289.

Salmela, L. and Rivals, E. (2014). Lordec: accurate and efficient long read error correction. *Bioinformatics*, page btu538.

Salmela, L., Walve, R., Rivals, E., and Ukkonen, E. (2016). Accurate selfcorrection of errors in long reads using de bruijn graphs. *Bioinformatics*, page btw321.

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*.