OXFORD UNIVERSITY PRESS | Bioinformatics

A Hybrid Genome Assembler for Second- and Third-Generation Sequencing

Journal:	*Bioinformatics*
Manuscript ID	BIOINF-2016-0541
Category:	Original Paper
Date Submitted by the Author:	05-Apr-2016
Complete List of Authors:	Huang, Yao-Ting; National Chung Cheng University, Department of Computer Science and Information Engineering Chen, Ping-Ye; National Chung Cheng University, Department of Computer Science and Information Engineering
Keywords:	HiTSeq, Assembly, Algorithms, Next-generation sequencing, Genome analysis

SCHOLARONE™
Manuscripts

For Peer Review

Bioinformatics, 2016, 0–0
doi: 10.1093/bioinformatics/xxxxx
Advance Access Publication Date: DD Month YYYY
Manuscript Category

Sequence Analysis

A Hybrid Genome Assembler for Second- and Third-Generation Sequencing

Yao-Ting Huang[1,*] and Ping-Ye Chen[1]

[1]Department of Computer Science and Information Engineering, National Chung Cheng Univresity, Chiayi, Taiwan.

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

For Peer Review

Abstract

Motivation: Genome assembly is challenged by the presence of large and complex repeats. The 2nd generation sequencing offers high throughput in low cost but the read length is inadequate to resolve large repeats. On the other hand, the 3rd generation sequencing can generate much longer reads for spanning large repeats, but the error rate and sequencing cost are much higher.

Results: This paper presented several new algorithms for assembling high-quality short reads and low-quality long reads in two stages. In the correction stage, long reads are mapped onto an FM-index constructed from short reads and polished by FM-index extension. In the assembly stage, a novel algorithm (called locality-sensitive backward search) is proposed to efficiently compute inexact overlap among (partially) corrected long reads. The results indicated that the correction power and accuracy is higher than existing methods. The developed assembler is able to assemble high-quality genome (N50 in Mb) using low-coverage PacBio sequencing.

Availability: https://github.com/ythuang0522/StriDe

Contact: ythuang@cs.ccu.edu.tw

Supplementary information: Supplementary material is available online.

1 Introduction

The technologies of 2nd generation sequencing (e.g., Illumina) offer higher throughput and lower cost compared with the Sanger sequencing. But the short length of reads has reduced its power and accuracy for assembling genomes with large and complex repeats (Earl et al., 2011; Magoc et al., 2012). In recent years, the advance of 3rd generation sequencing technologies (e.g., Pacific Biosciences and Nanopore) can generate longer reads (>10kb) in shorter time. These long reads are necessary for resolving large repeats from bacteria to mammalian-sized genomes. But the error rates and sequencing cost are much higher than those of 2nd generation sequencing. For instance, PacBio SMRT sequencing is able to produce reads with length up to 20k bp, and the average error rates can be as high as 15%.

Existing assemblers for 2nd generation sequencing are mainly based on de Buijn graphs (e.g., ABySS, SPAdes) (Bankevich et al., 2012; Simpson et al., 2009), which break each read into fixed-size *k*-mers. A graph is directly constructed where each vertex is a *k*-mer and each edge connects two overlapping *k*-mers. The construction of a de Bruijn graph is therefore very efficient. However, the graph structure is more complex since repeats are usually larger than *k*-mer. On the other hand, the underlying graph models for 3rd generation sequencing are mainly based on overlap-layout-consensus (OLC) graphs, which represent each read as vertex and overlap as edges (e.g., Celera Assembler, HGAP, Falcon) (Chin et al., 2013; Koren et al., 2013). The OLC assemblers are able to resolve larger repeats using the entire read length. However, the overlap computation among all reads is very time-consuming. MHAP and DAligner solved this challenge in different strategies (Berlin et al., 2015, Myers, 2014).

In order to reduce the error rates prior to assembly of 3rd generation sequencing, self-correction or hybrid correction are usually performed. Given sufficient sequencing coverage, the self-correction methods, e.g., HGAP/Falcon, correct the errors among overlapping reads. However, the cost of producing high-coverage 3rd generation sequencing is not practical for mammalian-sized genomes. On the other hand, the hybrid correction fixes the errors in long reads using high-quality short reads. Early approaches map short reads onto error long reads using alignment (e.g., PBcR in Celera). But the alignment stage is very slow due to tolerance of high error rates. Instead of using alignment, LoRDEC mapped PacBio reads onto a de Buijn graph (built from high-quality reads), which reduced the correction problem to de Buijn walks within the graph (Salmela and Rivals, 2014). Nevertheless, the underlying data structure (bloom-filter) faced a tradeoff between memory and accuracy, and the method is only an error-correction tool instead of a complete assembler. The most complete hybrid solution, DBG2OLC, first assembled high-quality short reads into contigs, used contigs as anchors for detecting overlap among long reads, inferred the layout from the OLC graph, and finally polished the consensus sequence. Since the anchor contigs is built from short reads, the overlap computed by anchor contigs is still challenged by repeats larger than short reads.

In this paper, we propose a novel hybrid assembler for PacBio and Illumina sequencing. The low-quality long reads (PacBio) are first corrected by mapping accurate substrings (called seeds) to an FM-index constructed from high-quality short reads (Illumina), and errors are iteratively corrected using FM-index extension among seeds. The assembly stage follows the OLC paradigm with additional modules for cleaning chimeric/palindromic PacBio reads. In particular, a novel overlap algorithm called locality-sensitive backward search (LSBS) was developed for detecting overlap among (partially) corrected PacBio reads. The results indicated that the correction power and accuracy is higher than existing methods. In addition, the assembly results showed that our method is able to assemble high-quality genome with high accuracy.

2 Methods

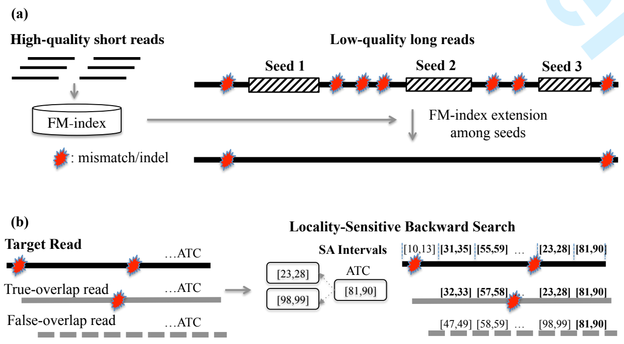

Fig. 1. Overview of two key components in the method. (a) The low-quality reads are corrected by first mapping accurate seeds in long reads into FM-index of short reads, which is followed by iterative FM-index extension between seeds; (b) The true-overlap read tends to share more common (inclusive) SA intervals with target read (shown in boldface) in comparison with false-overlap read.

Two stages, correction and assembly, are separately presented in this paper. In the correction stage, accurate seeds in long reads are identified and mapped into an FM-index constructed from high-quality short reads (**Fig. 1**(a)). Subsequently, the long read sequences are iteratively replaced by accurate sequences obtained from FM-index extension between each pair of adjacent seeds. Second, in order to detect overlap among partially corrected PacBio reads, a novel algorithm on the basis of FM-index (called locality-sensitive backward search) combines the ideas of locality-sensitive hashing with the backward search algorithm (**Fig. 1** (b)). The backward search (started from target read) aims to collect Suffix Array (SA) intervals of all potentially-overlap reads. The true-overlap read shares more common SA intervals with the target read in comparison with the false-overlap read. Below, we present the details of components of these two stages. The FM-index construction is based on Li's ropebwt2 implementation (Li, 2015).

2.1. Error Correction Stage

Seed Identification Using Adaptive *k*-mers

The *k*-mers (substrings of length *k*) with frequency greater than a threshold *t* is termed solid *k*-mers. A seed is a run of consecutive solid *k*-mers overlapped by (*k*-1) bases. In practice, we observed that the sequencing error rate of PacBio still varies across the entire genome. Consequently, different sizes of *k* are required to adapt to distinct error rates. A range of *k*-mers between k_{max} and k_{min} (31 and 17 by default) are used for searching seeds. Starting from k_{max} (see **Fig. 2**), a seed is searched from 5' end of a PacBio read until one or more consecutive solid *k*-mers are found, or a maximum expected distance (d_k) is reached, where d_k is related to *k* and can be estimated via a regression model (to be discussed later). If no single solid *k*-mer can be found within d_k, this process is iteratively repeated with a smaller (*k*-1)-mer (and new expected distance d_{k-1}), until k_{min} is reached (e.g., Seed 2 in **Fig. 2**). If no solid *k*-mers can be found even under k_{min} size, the seed search will restart after d_k bases (e.g., Seed 3).

In this algorithm, the expected distance between two solid *k*-mers (d_k) is related to the size of *k*-mer, because large and small *k* tend to have longer and shorter d_k, respectively. In order to know the expected distance for any *k*, we train a regression model using various *k*-mers and d_k learned from existing data sets. A set of PacBio reads (SRR1284073) is downloaded and BLAST-aligned onto the reference genome. Given any fixed integer *k*, the distance between two solid *k*-mers can be thus calculated in conjunction with the BLAST results. Non-linear regression was applied to obtain a regression model between the observed distance d_k and corresponding *k*, where $d_k = 3.8649 * e^{0.1239*k}$. This regression model is used for determining the d_k with respect to different *k*-mer sizes during seed search.

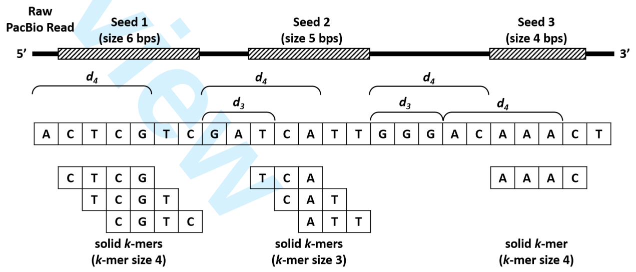

Fig. 2. Illustration of identification of seeds in a PacBio read. The k_{max} and k_{min} are 4 and 3, and d_4 and d_3 is represented two expected distance of *k*-mer size 4 and 3. Seed 1 is identified because 3 solid 4-mers are found within d_4. Seed 2 isn't found using 4-mer within d_4 initially. It is instead identified using 3-mer within d_3. The next Seed 3 is too far away such that no seeds can be found within d_4 and d_3 after Seed 2. Therefore, we move to the position d_3 after Seed 2 and re-start the entire process.

Correction by FM-index Extension between Seeds

The sequence between two seeds is corrected by searching a feasible overlapping path among short reads using FM-index extension. The detailed implementation of FM-index extension can be found in Huang and Liao (2016). Below, we briefly introduce the main idea and focus on

For Peer Review

the major differences, which are specific for hybrid PacBio and Illumina reads (see **Fig. 3**). Given a pair of seeds (and their SA intervals), the algorithm finds feasible {A,C,G,T}-extensions from the first seed to the other via updating the suffix array (SA) intervals. The extension is continued until reaching the SA interval of the other seed. Note that each feasible extension implies a set of overlapping short reads, because the FM-index is constructed from short reads.

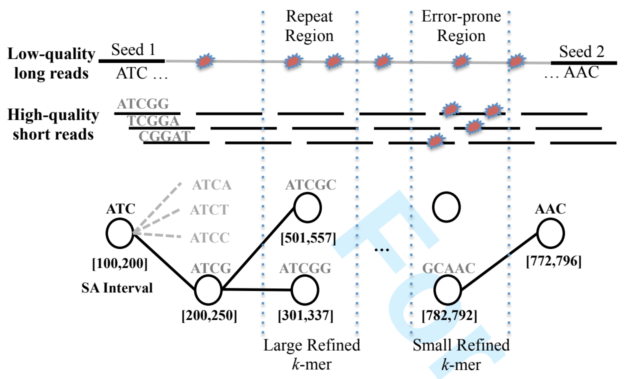

Fig. 3. Illustration of FM-index extension between two seeds. Each extended path represents a set of overlapping short reads. When the implicit overlap exceeds the short read length, the refined *k*-mer size should be adaptively set to large and small sizes in repeat and error-prone (high-GC) regions, respectively.

If there are multiple feasible paths found, we select the path which best matches the observed distance between two seeds in the PacBio read. In practice, because PacBio errors are mostly insertion errors, we found that the distance between two seeds on a PacBio read is often longer than the real distance. Furthermore, the degree of discrepancy is related to the distance between seeds, because longer distance tends to acquire more indel errors. Therefore, linear regression model is trained to learn and correct the discrepancy between real and observed distances.

Note that the update of existing SA intervals will increase the implicit overlap length among short reads. Consequently, the algorithm will refine all SA intervals once the implicit *k*-mer overlap exceeds the length of short reads, which will result in no feasible extensions. This requires setting a refined *k*-mer size. We found that, because of sequencing bias in the Illumina platforms (e.g., high-GC regions), the refined *k*-mer size must be smaller in error-prone (high-GC) regions, where short reads contain high-density of errors. On the other hand, within repetitive regions, the refined *k*-mer size must be larger in order to unambiguously extend through repeats. These two regions can be known during extension, where the error-prone regions lead to no feasible extension for all leaves, and repeats result in too many leaves.

Finally, in order to apply the same idea within ordinary regions, the updating *k*-mer size always starts with a larger value, and gradually shrinks to a smaller one if no extensions can be found. As a consequence, the FM-index extension algorithm can be considered as an adaptive variant of previous one with dynamic *k*-mer sizes and expected distances. Suppose the distance between two seeds is *d* and the maximum number of allowed extensions is *m*. This correction algorithm takes O(*dm*) time, because the additional refinement cost O(*k*) can be amortized into majority of O(1) SA update (see Huang and Liao, 2016).

Removal of Chimeric and Palindromic Reads

Another major challenge of PacBio sequencing is the presence of chimeric and palindromic reads (see Supplementary Figure S1). The palindromic reads can be easily detected and removed during FM-index extension process, because no feasible extensions can be found in this case. If any seed extension failed, The PacBio reads will be split in those seeds (or trimmed if seeds are in the head/tail) by our program.

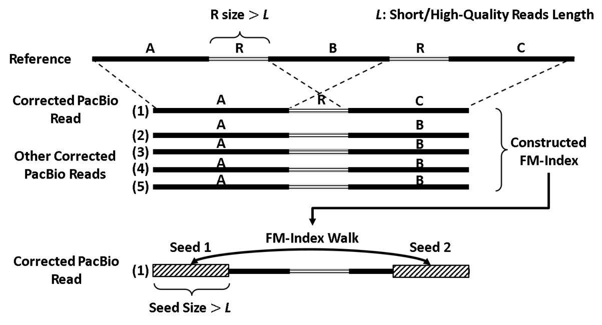

Fig. 4. Illustration of removal of chimeric reads, where the first PacBio read (ARC) is a chimera. {A,B,C} and {R} represent unique and repeat regions, respectively. By using FM-index extension with sufficient seed size (>R) and frequency (e.g., >2), the chimeric reads will fail the extension from head to tail, while the corrected reads will succeed, because frequency of ARB is larger than that of ARC.

Theoretically, chimeric reads can be also split or trimmed in similar way after FM-index extension. Unfortunately, we found a few chimeric reads are still successfully extended. Further investigation found that these chimeric reads usually contain large repeats, although the underlying biological reason is unknown. Moreover we found that these chimeric reads are often two distinct DNA fragments wrongly glued at the (common) repeats. The seeds within the large repeats are thus still successfully extended, which is indistinguishable from normal reads.

In order to further remove these chimeric reads, we construct an FM-index from the (partially) corrected long reads. Subsequently, each long read goes through a validation process by FM-index extension from head to tail using a very large *k*-mer (750bp by default) and a minimum frequency (3 by default) (see **Fig. 4**). The idea is, because the amount of chimeric/palindromic reads is relatively smaller in comparison with that of normal reads, the extension using FM-index of long reads implicitly performs a majority vote among all long reads. For example, the chimeric/palindromic reads will tend to fail the validation test (e.g., insufficient support of ARC), while normal reads tend to pass owing to sufficient amount (e.g., ARB).

However, the test of above simple algorithm showed that some correct PacBio reads also failed the validation, because they are from low-coverage regions or consist of many uncorrected errors. The false removal of these correct reads will lead to contig fragmentation of subsequent OLC assembly. Instead of throwing invalidated reads right away, they are decomposed into overlapping subreads of length *k* overlap by (*k*-1) bases. The idea is similar to de Bruijn graph construction where reads are split into *k*-mers. The split subreads (from chimeric/palindromic reads) will have smaller overlap in comparison with long reads in the OLC graph, which can be detected and removed at assembly stage (see Fig. 5(a)). On other hand, the false-split subreads from correct PacBio reads due to uncorrected errors often form simple bubble/tip structure, which can be removed and collapsed by graph layout algorithms (Fig. 5(b)). Note that the majority of correct long reads will not be decomposed, and thus OLC graph can still make use of entire long reads for resolving most repeats.

For Peer Review

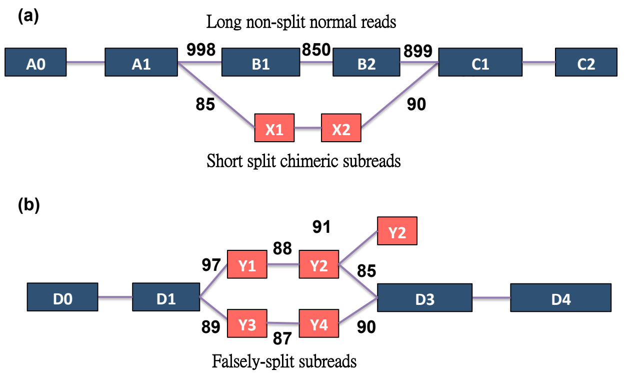

Fig. 5. Illustration of overlap lengths (shown on edges) between split short subreads and non-split long reads. (a) The short-split chimeric subreads have smaller overlap in comparison with long normal reads, which can be easily removed; (b) The false-split subreads owing to uncorrected errors tend to form simple bubbles/tips, which are resolvable by layout algorithms.

2.2. OLC Assembly Stage

Both the decomposed and normal long reads are all used to build an FM-index using Li's ropebwt2 algorithm (Li, 2015), which are suitable for short and long reads. The assembly stage basically follows the OLC paradigm. Although several components in OLC are required in implementation (e.g., bubble/tip removal), the overlap step is the most important and novel one in our method. To our best knowledge, Simpson and Durbin (2012) ever used FM-index for inexact overlap computation in their SGA assembler. But its running time is exponential to the number of mismatches to be tolerated. And the algorithm is unable to tolerate indel errors.

Overlap via Locality-Sensitive Backward Search

Because the long reads may still contain uncorrected mismatch/indel errors, an inexact overlap computation algorithm is required. Second, since the decomposed subreads increase the total amount of reads for overlap computation, the algorithm must be fast enough. In spirit, the proposed locality-sensitive backward search (LSBS) combines the idea of locality-sensitive hashing (used in HGAP) with backward search algorithm (**Fig. 6**). LSBS is also similar to FM-index extension from implementation point of view, but it simultaneously measures overlap similarity and prunes unlikely extensions (overlaps) along the extension process.

Given a query read, we first computed the SA intervals of all *k*-mers along the read (where *k*=17 by default) and store these intervals into an array S_t. Subsequently, the backward search of potentially overlap reads starts from any (or multiple) seeding position, where seed is a substring within the read. Then, the seed is recurrently backward-extended to all possible {A,C,G,T} bases (via backward search algorithm), until the length of long read is reached. In other words, all possible reads having the same initial seed can be found by this simple search. However, the false-overlap reads contain the initial seed simply by chance should be discarded (e.g., E in **Fig. 6**).

Therefore, during the extension process, we also measure the overlap similarity of each new extension according to the amount of common SA intervals shared with the query read (i.e., S_t). Although SA intervals stored in S_t can be ordered according to each *k*-mer position in query, the presence of uncorrected indel errors in query and overlap reads disrupt the indices. Fortunately, we found that the indel size in PacBio system is in less than 30bp, which still allows efficient range query by scanning SA intervals within a indel-sized window in S_t. Consequently, for each new extension, we check whether the new SA interval is existed (shared) in approximate the same location in S_t. Note that because the prefix of the extended sequences may be different from the query read (due to uncorrected mismatches/indels), the SA interval does not need to be completely identical. Any new interval inclusive within one in S_t indicates both reads share the same common *k*-mer.

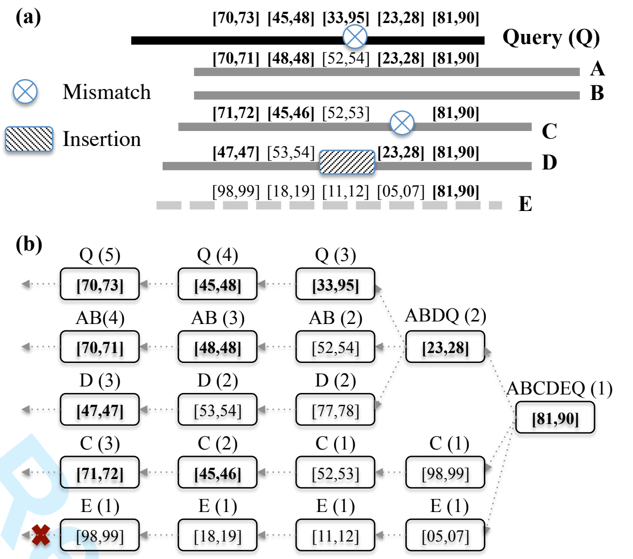

Fig. 6. (a) Five potentially-overlap reads (A,B,C,D,E) against query (Q) are illustrated. A and B are identical reads and thus have the same SA intervals, which are compressed in a single extension path. C and D contain one mismatch and one insertion error, respectively, and thus has its own extension path. E is a false-overlap read which contains the initial seed by chance; (b) Step-by-step execution of LSBS. Each box stands for a successful FM-index extension with SA interval shown inside. The implicitly compressed reads are shown above each box and the number of common SA intervals shared with Q is shown in parenthesis. The extension with insufficient amount of shared SA intervals with Q will be pruned (e.g., E).

In summary, the similarity of each extension is measured by the total amount of common SA intervals divided by the total extended intervals. Recall that the true-overlap reads tend to share more common SA intervals (a.k.a substrings) with target read in comparison with false-overlap reads (**Fig 1**(b)). Given a maximum error rate allowed (*e*), if the similarity of current extension falls below *e*, it is pruned immediately. In comparison with locality-sensitive hashing, the identical substrings (especially repeats) are compressed into a single SA interval, which significantly boosts the efficiency.

For Peer Review

Table 1. Correction power and accuracy of LoRDEC and StriDe on two hybrid sequencing data sets (*E coli* and *C. elegans*). The power is evaluated using sum and length of corrected reads. The accuracy is compared using per-base identify and clipping rate (for structural errors due to chimera/palindrome).

	Source	No. Reads	Sum of Reads	Length of Reads	Identity	Clipping Rate
E. coli	Raw PacBio	144,960	644 Mb	13,660	84.49%	99.92%
	LoRDEC	93,507	350 Mb	6,966	99.8%	0.086%
	StriDe	64,738	**367 Mb**	**8,367**	**99.99%**	**0.001%**
C. elegans	Raw PacBio	723,325	8,113 Mb	16,577	85.66%	90.88%
	LoRDEC	3,239,589	2,880 Mb	902	98.24%	39.372%
	StriDe	1,327,827	**6,006 Mb**	**7,543**	**99.95%**	**0.012%**

The complexity of this algorithm requires a sophisticated analysis considering the repeat percentage and correction power. Below we first give a simple answer in an ideal scenario without repeats and perfect power. Suppose the target read length is R and sequencing coverage is C. The complexity of this overlap algorithm is O(RC), because the extension length is O(R) and maximum number of extensions (a.k.a overlapping reads) is O(C). This complexity does not hold under presence of repeats, where the number of extensions will grow larger than C. But because different copies of the same repeats are compressed into a single SA interval, the number of extensions is implicitly compressed. Furthermore, in the implementation, we always start from non-repeat seeds at the beginning of backward search and limit the maximum number of leafs (C) during extension. Therefore, the running time will not grow exponentially with respect to repeats. Finally, the presence of uncorrected errors will also increase C. The experimental results indicated the correction power of our method is high (>99.9%, see **Table 1**), which implies the running time is not seriously reduced by the uncorrected errors.

Graph Layout Algorithms

The transitive edges in the OLC graph are first removed by using Myer's algorithm (Myer, 2005). Vertices along the same simple path are then contracted into a large vertex. Because the split subreads from uncorrected error reads usually form a cluster of tips, an iterative tip removal is performed from small to large tips. Second, in order to distinguish bubbles of ordinary reads from those of chimeric reads, we require a bubble can only be collapsed into a simple path if all paths from one end to the other end of the bubble have approximately the same length (e.g., **Fig. 5**(b)). Otherwise, the bubble should not be collapsed. Finally, for each vertex with two or more edges remained, the insufficient-overlap edges (defined as overlap less than a ratio θ to read length) are removed, if there are other sufficient-overlap edges existed. The removal of these chimeric edges will transform the chimeric vertex into tips or islands, which can be finally removed by running the tip removal algorithm again. θ is set to 0.8 by default, which should be adjusted according to the sequencing coverage.

Implementation

The entire implementation is largely based on two open source projects of Simpson and Richard's SGA and Li's ropebwt2. Both the correction and assembly stages are multi-threaded. The developed components are encapsulated using bash shell script. The programs have been implemented in C++ and embedded into a previous assembler called StriDe.

3 Results

In order to compare different programs specific to the correction and assembly stages, we presented the results of two stages separately.

Error Correction Power and Accuracy

We downloaded two hybrid datasets from *E.* coli and *C. elegans*, in which both PacBio sequencing and Illumina sequencing are available. The *E. coli* data set includes 644 Mb PacBio and 7.9 Gb Illumina reads, and the *C. elegans* data set include 2.8 Gb PacBio and 14.3 Gb Illumina reads. The details are given in Supplementary Table 1. We compared StriDe with LoRDEC, which is a hybrid correction tool for PacBio reads using short reads. Table 1 lists several metrics related to correction power and accuracy.

In terms of correction power, the sum of reads after correction of both methods are reduced, because both of them throw away or trim PacBio reads if the reads are uncorrectable (mainly due to contamination, very low-quality reads, or chimeric/palindromic reads). StriDe retains larger amount of reads in both data sets in comparison with LoRDEC, which apparently over-throw reads in the *C. elegans* data set while Stride retains ~75% of original data. The read length after correction is also an important metric since longer reads are beneficial for subsequent assembly. The read lengths of StriDe again are clearly longer than those of LoRDEC. However, the read lengths are both much shorter than original reads, which still require further improvement.

In terms of correction accuracy, the per-base identities of both methods are computed by BLAST-alignment of reads to their own references. The identity of original PacBio reads is around 85%. After correction by both methods, the identities can be improved to over 99%, where StriDe slightly outperforms LoRDEC in both data sets. The high accuracy of StriDe (>99.9%) also implies that the speed does not deteriorate seriously with respect to uncorrected errors. The clipping rate aims to identify large structural errors (e.g., chimera/palindrome) of error reads, which cannot be seen from the base-level identity. This is computed by running BWA-MEM which provides clipped information of partially-aligned reads. The results indicated that StriDe again outperforms LoRDEC in this metric, which implies StriDe has less chimeric/palindromic structural errors. LoRDEC has very high clipping rate in *C. elegans* while StriDe still maintains low rate, which is mainly due to the modules of chimeric/palindromic removal.

Assembly Contiguity and Accuracy

Because LoRDEC is a correction tool without assembly modules, we used Celera assembler to assemble reads corrected by LoRDEC. The current state-of-the-art complete solution for such hybrid sequencing data is called DBG2OLC. DBG2OLC requires a pre-assembled contigs from Illumina reads as input and we use the suggested SparseAssembler to assemble contigs. Table 2 lists the contiguity and accuracy of each method at different coverage. The LoRDEC+Celera assembled a very

For Peer Review

Table 2. Assembly contiguity and accuracy on different PacBio coverage of E coli data sets. The number of misassembled contigs reported by QUAST is shown in parenthesis after the number of contigs.

PacBio Coverage	Program	No. of Contigs	N50 (bp)	Min (bp)	Max (bp)	Sum (bp)
120x	LoRDEC+Celera	806	**5,722**	1,002	34,615bp	3,319,082
	DBG2OLC	10 (4)	1,257,787	28,287	1,435,412	4,651,044
	StriDe	**4 (0)**	**3,179,994**	**434,961**	**3,179,994**	**4,627,340**
60x	DBG2OLC	10 (3)	774,906	22,012	1,789,228	4,579,118
	StriDe	**4 (0)**	**3,179,900**	**434,951**	**3,179,900**	**4,627,219**
30x	DBG2OLC	9 (4)	1,101,087	12,057	1,230,149	4,198,969
	StriDe	**8 (0)**	**1,351,002**	**29,507**	**1,430,058**	**4,614,125**

fragmented genome even at 120x coverage of PacBio sequencing. We did not further tested them in the low-coverage experiments. As StriDe and LoRDEC both has very high correction identities, the poor performance of LoRDEC is probably due to lack of sophisticated algorithms for removing the chimeric/palindromic reads containing large repeats.

DBG2OLC has a wide range of parameter combinations and we found the results are very sensitive to right choice at specific coverage. We tried our best to test a range of parameter combinations (as well as followed the suggested parameters on website) and select the best results among them. The selected parameters producing best results are given in Supplementary Table 3. StriDe outperforms DBG2OLC in terms of both contiguity and accuracy. It is possible to obtain a better result for DBG2OLC under 60x coverage, since their results of 30x/120x are better. The reported result is the best parameter combinations we can find at this moment. Although DBG2OLC also produced contig N50 over 1Mb under 30x/120x, there are misassembled contigs in the assembled genome (validated by QUAST), while StriDe does not produce misassembly.

In fact, StriDe outputs a nearly complete genome at 60x and 120x coverage, leaving only four contigs remained. Further investigation reveals that these four contigs are due to three sequencing gaps of Illumina platform on *E. coli* genome (see **Fig. 7**). The correction by FM-index extension at these three gaps will always fail, because the FM-index is constructed from Illumina reads. This leads to gaps in corrected PacBio reads and thus the final contig gaps. Theoretically, the four contigs can be still assembled into one scaffold by using PacBio reads, because there are sufficient number of PacBio reads spanning these gaps. But our current implementation is still lack of a scaffolding module.

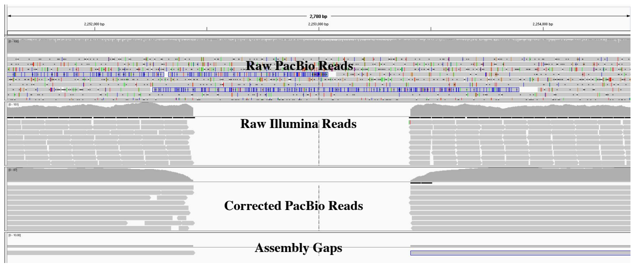

Fig. 7. The Integrative Genomics Viewer (IGV) plot of raw PacBio reads, raw Illumina reads, StriDe-corrected PacBio reads, and assembled contigs over one assembly gap. The root cause of these gaps is all due to the original sequencing gaps in raw Illumina reads, which leads to gaps in corrected PacBio reads and finally the assembly gaps.

Computational Resource and Running Time

Because the underlying data structure used is FM-index, the memory requires only 2Gb for assembling the *E. coli* genome. The entire running time for assembling 120x~30x E. coli (including correction and assembly) ranges from 40 to 20 minutes.

4 Discussion and Conclusion

This paper presented algorithms for efficient error correction and inexact overlap using hybrid 2nd and 3rd generation sequencing. The two key components in this assembler (i.e., correction by FM-index extension and overlap by LSBS) both can run approximately in time linear to the length of long read, assuming repeats are carefully processed. The correction via FM-index extension is in fact a generalization of LoRDEC's correction algorithm via de Bruijn walks, because FM-index can be imagined as a multi-*k*-mer de Bruijjn graphs. The major advantage of our method is the therefore the flexibility of using distinct *k*-mer sizes in error-prone or repetitive regions. The deficiency of LoRDEC could be partly also due to the usage of bloom-filter. On the other hand, the usage of FM-index in our method will never face the risk of false positives.

The overlap computation via LSBS improves an earlier exponential-time algorithm for inexact overlap computation using FM-index (Simpson and Durbin, 2012). In addition, the new method can tolerate both mismatches/indels, whereas the previous one can only tolerate mismatches. Although the running time is almost asymptotically optimal and much faster than previous one in practice, the current implementation still requires lots of improvement for better speed. Initial test of the program on larger genomes indicate the speed bottleneck is FM-index construction, which we rely on Li's ropebwt2 algorithm. For example, the FM-index construction of ~60x 102Mb *C. elegans* genome takes one hour. The correction and overlap can finish within 10~20 minutes.

Finally, although there are still few gaps in the assembly, they are mainly owing to the sequencing gaps originated from Illumina system. Therefore, we think this framework can be still improved for producing high-quality assembly at even lower coverage of PacBio sequencing, because the correction stage now only used FM-index from Illumina reads, and the OLC stage only utilized FM-index of corrected PacBio reads. The mixture of both sequencing reads in a single FM-index may take full advantages of both platforms.

For Peer Review

Acknowledgements

The authors thank the encouragement of KMC, HKT, and TJC on this work.

Funding

This work has been supported by MOST grants with numbers 104-2221-E-194 -048 –MY2 and 103-2923-E-194 -001 -MY3.

Conflict of Interest: **none declared.**

References

Bankevich, A., Nurk, S., Antipov, D., Gurevich, A. A., Dvorkin, M., Kulikov, A. S., Lesin, V. M., Nikolenko, S. I., Pham, S., Prjibelski, A. D., Pyshkin, A. V., Sirotkin, A. V., Vyahhi, N., Tesler, G., Alekseyev, M. A., and Pevzner, P. A. (2012). Spades: a new genome assembly algorithm and its applications to single-cell sequencing. *J Comput Biol*, 19(5), 455–77.

Berlin, K. et al. (2015). Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. Nat Biotechnol, 33:623–30.

Bradnam, K., Fass, J., Alexandrov, A., Baranay, P., Bechner, M., Birol, I., Boisvert,S., Chapman, J., Chapuis, G., Chikhi, R., Chitsaz, H., Corbeil, J., Del Fabbro, C., Docking, T., Durbin, R., Earl, D., Emrich, S., Fedotov, P., Fonseca, N., Ganapathy, G., Gibbs, R., Gnerre, S., Godzaridis, E., Goldstein, S., Haimel, M., Hall, G., Haussler, D., Hiatt, J., Ho, I., and Howard, J. (2013). Assemblathon 2 assemblies. *GigaScience Database*.

Chin, C.-S. et al. (2013). Nonhybrid, finished microbial genome assemblies from longread smrt sequencing data. *Nat Methods*, 10:563–9.

Dormand,J.R. and Prince,P.J. (1980) A family of embedded Runge–Kutta formulae. *J. Comp. Appl. Math.*, **6**, 19–26.

Earl, D., Bradnam, K., St John, J., Darling, A., Lin, D., Fass, J., Yu, H., Buffalo, V., Zerbino, D., Diekhans, M., Nguyen, N., Ariyaratne, P., Sung, W., Ning, Z.,Haimel, M., Simpson, J., Fonseca, N., Birol, I., Docking, T., Ho, I., Rokhsar, D., Chikhi, R., Lavenier, D., Chapuis, G., Naquin, D., Maillet, N., Schatz, M., Kelley, D., Phillippy, A., and Koren, S. (2011). Assemblathon 1: a competitive assessment of de novo short read assembly methods. *Genome Res*, 21, 2224–41.

Ferragina, P. and Manzini, G. (2000). Opportunistic Data Structures with Applications. Proceedings of the 41st Annual Symposium on Foundations of Computer Science, pages 390–398. Gurevich, A., Saveliev, V., Vyahhi, N., and Tesler, G. (2013). QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, 29(8), 1072–1075.

Huang, Y.-T. and Liao, C.-F. (2016). Integration of string and de Bruijn graphs for genome assembly. *Bioinformatics*, 32:7.

Li, H. (2014). Fast construction of FM-index for long sequence reads. *Bioinformatics*.

Luo, R., Liu, B., Xie, Y., Li, Z., Huang, W., Yuan, J., He, G., Chen, Y., Pan, Q., and Liu, Y. (2012). Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler. GigaScience, 1, 18.

Magoc, T., Pabinger, S., Canzar, S., Liu, X., Su, Q., Puiu, D., Tallon, L. J., and Salzberg, S. L. (2013). Gage-b: an evaluation of genome assemblers for bacterial organisms. Bioinformatics, 29(14), 1718–1725.

Myers, E.W. (2005). The fragment assembly string graph. Bioinformatics, 21(suppl 2), ii79–ii85.

Salmela, L. and Rivals, E. LoRDEC: accurate and efficient long read error correction, Bioinformatics, vol. 30, pp. 3506-14, Dec 15 2014.

Schirmer, M., Ijaz, U. Z., D’Amore, R., Hall, N., Sloan, W. T., and Quince, C. (2015). Insight into biases and sequencing errors for amplicon sequencing with the Illumina MiSeq platform. Nucleic Acids Res., 43(6), e37.

Simpson, J.,Wong, K., Jackman, S., Schein, J., Jones, S., and Birol, I. (2009). Abyss: a parallel assembler for short read sequence data. Genome Res, 19, 1117–1123.

Simpson, J. T. and Durbin, R. (2010). Efficient construction of an assembly string graph using the fm-index. Bioinformatics, 26(12), i367–i373.

Simpson, J. T. and Durbin, R. (2012). Efficient de novo assembly of large genomes using compressed data structures. Genome Res, 22(3), 549–56.

Ye, C. Hill, C., Ruan, J., Ma. Z. (2014) DBG2OLC: Efficient Assembly of Large Genomes Using the Compressed Overlap Graph, arXiv:1410.2801.

For Peer Review