Chu *et al. BMC Genomics* 2019, **20**(Suppl 5):426
https://doi.org/10.1186/s12864-019-5703-4

BMC Genomics

RESEARCH Open Access

GAPPadder: a sensitive approach for closing gaps on draft genomes with short sequence reads

Chong Chu*, Xin Li and Yufeng Wu

From 7th IEEE International Conference on Computational Advances in Bio and Medical Sciences (ICCABS 2017) Orlando, FL, USA. 19–21 October 2017

Abstract

Background: Closing gaps in draft genomes is an important post processing step in genome assembly. It leads to more complete genomes, which benefits downstream genome analysis such as annotation and genotyping. Several tools have been developed for gap closing. However, these tools don't fully utilize the information contained in the sequence data. For example, while it is known that many gaps are caused by genomic repeats, existing tools often ignore many sequence reads that originate from a repeat-related gap.

Results: We compare GAPPadder with GapCloser, GapFiller and Sealer on one bacterial genome, human chromosome 14 and the human whole genome with paired-end and mate-paired reads with both short and long insert sizes. Empirical results show that GAPPadder can close more gaps than these existing tools. Besides closing gaps on draft genomes assembled only from short sequence reads, GAPPadder can also be used to close gaps for draft genomes assembled with long reads. We show GAPPadder can close gaps on the bed bug genome and the Asian sea bass genome that are assembled partially and fully with long reads respectively. We also show GAPPadder is efficient in both time and memory usage.

Conclusion: In this paper, we propose a new approach called GAPPadder for gap closing. The main advantage of GAPPadder is that it uses more information in sequence data for gap closing. In particular, GAPPadder finds and uses reads that originate from repeat-related gaps. We show that these repeat-associated reads are useful for gap closing, even though they are ignored by all existing tools. Other main features of GAPPadder include utilizing the information in sequence reads with different insert sizes and performing two-stage local assembly of gap sequences. The results show that our method can close more gaps than several existing tools. The software tool, GAPPadder, is available for download at https://github.com/Reedwarbler/GAPPadder.

Keywords: Closing gaps, De novo assembly, Repeat elements, Sequencing analysis

Introduction

With the fast developing high-throughput sequencing technologies, de novo genome assembly from sequence reads has become a major application of sequencing technologies. So far many genome assembly software tools have been developed, including e.g. [1–4]. As sequence data from many species is becoming increasingly more available, draft genomes of many species have been assembled. Furthermore, more recent sequencing technologies such as long reads sequencing are expected to lead to even more assembled genomes with better quality than before.

Despite all these exciting developments, it is still challenging to obtain complete genomes with the current technologies and assembly tools, especially at regions that are highly repetitive or have low coverage. At present, most assembled genomes contain gaps. For relatively complex genomes, only draft genomes which usually contain a large number of gaps are available. A more complete

*Correspondence: chong.chu@uconn.edu
Dept. of Computer Science and Engineering, University of Connecticut, 371 Fairfield Way, Storrs, CT, USA

© The Author(s). 2019 **Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The Creative Commons Public Domain Dedication waiver (http://creativecommons.org/publicdomain/zero/1.0/) applies to the data made available in this article, unless otherwise stated.

genome is highly desirable since it leads to better annotation, less genotyping error and easier identification of causal variation associated with traits [5] than a genome with many gaps. For example, 45 new avian species have been sequenced and assembled recently in a comparative study of avian genomes [6]. Draft genomes of 25 out of these 45 species have average N50 around 48 kb, which indicates the draft genomes are fragmented with many gaps. About 3000 genes are likely missing or only partially annotated due to gaps. As a result, only 70 to 80% of the entire catalog of avian genes can be predicted, which may cause bias in downstream analysis.

With the development of the third generation sequencing technology, long reads from different platforms, like Pacific Biosciences, Illumina TruSeq, Oxford Nanopore, have been developed. With the help of these new technologies, the quality of the assembled draft genomes is greatly improved [7, 8]. In general, long reads are used in two ways to help to improve the draft genome assembly: 1) Long reads are used to scaffold the contigs and fill the gaps on the draft genomes assembled from high coverage short reads. 2) Long reads are directly used to assemble the draft genomes. Due to the high error rates of long reads, read depth is required to be high to guarantee the quality of genomes assembled directly from long reads, and thus sequencing cost can be high. In comparison, for scaffolding contigs and closing gaps with long reads, the coverage is usually not required to be very high. However, there are still gaps on the draft genomes even assembled with long reads, especially for draft genomes initially assembled high coverage short reads and then improved with long reads. Thus, it is still needed to close the gaps on draft genomes assembled with long reads. At present, short sequence reads are still the most available sequence reads. Thus, it is important to develop methods that can close gaps on draft genomes with short sequence reads that are readily available.

Several tools have been developed for closing gaps on draft genomes with short reads. GapCloser is a standalone tool in the SOAPdenovo [9] package. It performs several iterations of base extension steps using the reads aligned to specific regions. GapFiller [10] implements a method that finds read pairs with one end aligned within a contig and its mate partially aligned to the draft genome and partially located in a region identified as a gap. These partially aligned reads are used to close the gap through sequence overlapping. Sealer [11] generates pseudo long reads from paired-end sequence reads by filling the unknown sequences between read pairs using the redundancy in sequence coverage, and then the pseudo long reads are used to fill the gaps. While these approaches have been used to close gaps in assembled genomes, these tools still cannot close many gaps (especially those originated in more complex genomic regions, e.g. repeats).

In this paper, we develop a new approach called GAPPadder for closing gaps on draft genomes. Similar to tools such as GapCloser and GapFiller, GAPPadder also performs local assembly from reads that originate from gap regions. The following are the main features of GAPPadder and also differences between GAPPadder and the existing methods.

- GAPPadder uses more information about the gaps contained in sequence reads than existing methods. GAPPadder collects more reads relevant for gap closing, especially repeat-associated reads which are ignored by all the existing tools. Moreover, GAPPadder collects higher quality reads by utilizing more information with different insert sizes of paired-end (PM) and mate-pair (MP) reads.
- GAPPadder uses a different local assembly method for gap closing compared with existing methods. Existing methods often rely on local extension of contigs. GAPPadder, instead, performs a two-stage local assembly: it first assembles contigs in the gap and then generates higher quality local assembly of gap sequences by merging contigs.

We compare GAPPadder with existing approaches using real sequence data from staphylococcus aureus, human chromosome 14 from GAGE [12], and whole genome sequencing data (with PE and MP reads) of one human individual NA12878 from Illumina. These genomes are assembled from short reads only. We show GAPPadder can close more gaps than GapCloser, GapFiller and Sealer with these short sequence reads. Besides these draft genomes assembled with only from short reads, we also compare GAPPadder with GapCloser on two draft genomes assembled with long reads: the bed bug draft genome assembled with hybrid short and long reads and the Asian sea bass draft genome directly assembled from long reads. We show many gaps can be fully closed and extended by GAPPadder and GapCloser, and GAPPadder closes much more than GapCloser on the hybrid assembled bed bug genome.

Gaps in draft genomes

De novo assembly of reads produces contigs. Contigs are then further linked with paired-end (PE) or mate-pair (MP) reads to form scaffolds. Scaffolds contain multiple gaps, whose lengths are estimated from the insert sizes of PE or MP reads. In general, extension of contigs stops at sites with repetitive regions, heterozygous alleles, sequencing errors or low read coverage [13, 14]. Gaps can be mainly classified to three types. The most common type is the repeat-associated gap. Repeat is a piece of DNA which may have multiple copies in the genome. Note that these copies may differ slightly from

each other. There are different types of repeats, including LINE, SINE, LTR elements, DNA transposon, satellites, etc. Repeat-associated gaps can be categorized to be satellite-associated, dispersed low divergent repeats-associated, and tandem repeats-associated. We show the results of masking the gap regions on chromosome 14 of human using RepeatMasker [15] in Fig. 1. To get the gap regions of the draft genome of chromosome 14, which is assembled by ALLPATHS-LG and released in GAGE, we align the flanking regions to the reference genome, and thus get the benchmarked gap sequences (i.e. sequences from the reference genome that are missing in the draft genome). One can see that over 90% of the gaps are masked as repeat-associated gaps. Therefore, to develop gap closing methods, it can be very useful to integrate the information come from repeats.

Results

We compare GAPPadder with GapCloser, GapFiller and Sealer on datasets of three draft genomes of different sizes and with known reference sequences: staphylococcus aureus, human chromosome 14 and human whole genome. Data of staphylococcus aureus and human chromosome 14 are from GAGE [12]. We choose the draft genome assembled by ALLPATH-LG. For staphylococcus aureus, two groups of high coverage data of different insert sizes are used. While for the human chromosome 14, three groups of data of different insert sizes are used. The data with long jump library is of very low coverage. The human whole genome (NA12878) high-coverage PE and MP sequence reads are from Illumina. The draft genome of NA12878 is released in [16], which is assembled by ALLPATH-LG. Detailed information of the four datasets are given in the Additional file 1.

As there are high quality reference genomes for staphylococcus aureus and human, we can benchmark each closed gap sequences against the "true" sequences from the reference genome. To get the "true" gap sequences, for each gap we first collect the left and right flanking regions (by default 300 bp each) from the assembled draft genome. If two gaps are close to each other (distance smaller than 300 bp), then the whole middle region between the two gaps are used as the flanking region. Then, we align the flanking regions to the reference genome using BWA. For one gap, if both the left and right flanking regions are unique (with mapping quality 60) and fully (allow 15 bp soft-clip at the breakpoints) mapped, and the mapping orientation are same, then the sequence between the two aligned flanking regions is viewed as "true" gap sequence. In this way, 23, 3934, and 220,318 "true" gap sequences are collected for staphylococcus aureus, human chromosome 14 and human whole genome respectively.

To validate whether the gap sequences are correct or not, for each gap we first align the two flanking regions to the new scaffold to extract the "closed" sequence using the same way as described above. Then, we align the "closed" gap sequence to its related "true" gap sequence, and if the gap sequence can be well aligned (by default allow 15 bp soft-clip on both ends) to the "true" gap sequence, then we view the gap is correctly closed. Note, if the gap is not fully closed, but only extended, we require the extended sequences must be well aligned (also allow 15 bp soft-clip) to the "true" gap sequence.

Comparison with existing tools

In Table 1, we show the results of GAPPadder and the other three tools on staphylococcus aureus, human chromosome 14 and human whole genome. Note that Sealer only runs well on short insert size data. For data with very long insert size, it can be extremely slow. So when running Sealer on the human whole genome data, we do not use the long insert size data. Detailed commands

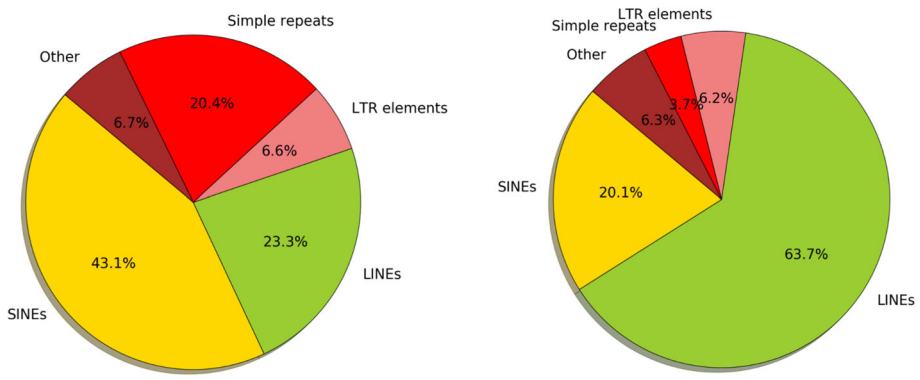

Fig. 1 Percentage of the masked gap sequences of each type of repeats. 3934 gaps are extracted from the draft genome of human chromosome 14 that is released in GAGE. By aligning the flanking sequences to the human reference, we extract the gap sequences. We use RepeatMasker to get the types of repeats (e.g. LINE, SINE, LTR elements, etc) of these gap sequences. One gap may be masked to multiple repeat types. The left part shows the percentage of masked gaps for each repeat type. The right part shows the percentage of masked bases for each repeat type

Table 1 Comparison of the four tools on three datasets: S. aureus, human chromosome 14 and whole human genome for NA12878, whose draft genomes have 23, 3934, and 220,318 gaps respectively

Species	Gap Num.	Methods	Gaps fully closed
S. aureus	23	GAPPadder	9
		GapCloser	2
		GapFiller	1
		Sealer	2
H. chrom 14	3934	GAPPadder	1670
		GapCloser	1184
		GapFiller	732
		Sealer	559
NA12878	220,318	GAPPadder	130,371
		GapCloser	Out of memory
		GapFiller	Not finished (After 725 hrs)
		Sealer	110,876

Overall, GAPPadder closes more gaps than the other three tools on the these datasets

and parameters of running each tool are provided in the Additional file 1. The results show that GAPPadder outperforms the other three tools on the three datasets. For S. aureus and H. chromosome 14 datasets, GAPPadder closes more gaps than the other three tools. For the human whole genome datasets, GapCloser runs out of memory (on a server with 256 G memory) and GapFiller did not finish after running for more than 725 h. In comparison, GAPPadder and Sealer respectively close 130,371 and 110,876 gaps out of the 220,318 gaps.

To show the effect of the repeat-associated reads, we run a revised version of GAPPadder that does not use these repeat-associated reads for gap filling. This "streamlined" version of GAPPadder closes 1103 gaps, much less than the original version of GAPPadder, which closes 1670 gaps. This indicates that repeat-associated reads are indeed useful for gap closing.

In Fig. 2, we compare the four tools on different ranges of gap lengths of the closed gaps. The left part shows the distribution of gap length of all the 3934 gaps on the draft genome of human chromosome 14. Over 80% of the gaps are shorter than 1k, and over 95% of the gaps are smaller than 2k. The right part shows the number of fully closed gaps of the four tools on different ranges of gap length. GAPPadder significantly outperforms the other three tools on gaps shorter than 1 kb, while GAPCloser performs slightly better on gaps longer than 1 kb.

Comparison on data with different insert sizes

For assembling draft genomes, usually data of different insert sizes are provided. Paired-end reads of long insert size or mate-paired reads can be helpful for closing (especially long) gaps on the draft genomes. Because of the different strategies used, the performance of different tools differs significantly on datasets of different insert sizes. To evaluate the performance of the four tools on different insert size datasets, we compare the four tools on the human chromosome 14 datasets with only short insert size data, only long insert size data, and combined data with short and long insert size. The results are shown In Table 2. On the data with short insert sizes, GapCloser performs the best. But with only long inset size data, GAPPadder significantly outperforms the other three tools. For comparison on the combined dataset with reads of both short and long insert sizes, GAPPadder performs the best.

Time and memory usage

All four tools are benchmarked on a 64-core server with AMD 6380 CPU @2.499 GHz and 256 GB RAM. To compare the time and memory usage of these four tools, we benchmark the four tools on the human chromosome 14 datasets. When running Sealer, we set the maximum allowed memory to 40 G, and other parameters are set as suggested by its manual. For GapCloser, we use the default parameters. For GapFiller, the parameter for the number of iterations to run is set to be 5. See the Additional file 1 for more detailed information of running the tools. In terms of running time, GapCloser, GapFiller, Sealer and GAPPadder take 30 m 32 s, 424 m 47 s, 160 m 23 s, and 85 m 12 s respectively. For memory usage, GapCloser takes 7.8 G at the peak, Sealer takes 40 G (as set in the parameter), while GapFiller and GAPPadder take less than 2 G memory. Therefore, GapCloser is the most efficient one among the four tools, but it requires more memory. GAPPadder is slightly slower than GapCloser but uses much less memory.

Closing gaps on draft genomes assembled partially or fully with long reads

Although long reads help to improve the draft genome assembly, large number of gaps may still remain in the draft genome, especially for the draft genome originally assembled from short reads and then improved from long reads. To evaluate the performance of GAPPadder on draft genomes that are partially and fully assembled with long reads, we run GAPPadder on two draft genomes: 1) The bed bug cimex lectularius draft genome (released in [17]) which is assembled with hybrid data of both short and long reads, 2) Asian sea bass draft genome (released in [18]) that is purely assembled from high coverage PacBio long reads. The bed bug genome is initially assembled with 73× coverage Illumina short reads using ALLPATHS-LG [1] assembler. And then Illumina Moleculo kit is used to sequence long reads with average length 3500 bp, which is used to improve the

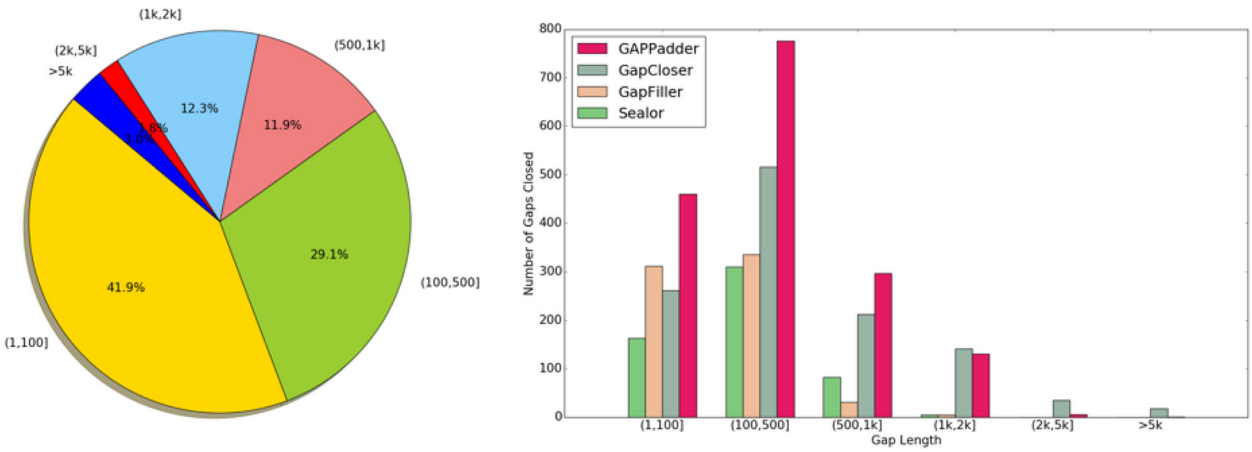

Fig. 2 Comparison on different gap length: The left part shows the distribution of gap length of all the 3934 gaps on the draft genome of human chromosome 14. Over 80% of the gaps are smaller than 1 kb, and over 95% of the gaps are smaller than 2 kb. The right part shows the number of fully closed gaps of the four tools on different ranges of gap length

initial assembled draft genome. However, even for the improved draft genome, there are still many gaps. In the final released assembled genome, there are 118,821 gaps, out of which 97,251 gaps are larger than 100 bp. We run GAPPadder and GapCloser to close the gaps. As some gaps are really small (just several bases), and to evaluate the power of different tools we only focus on these 97,251 gaps that are larger than 100 bp. Three sets of Illumina short reads with insert sizes of 185, 367, and 3000 bp and coverage of 34×, 12×, and 7× respectively are used for gap closing. GAPPadder reports 19,476 gaps are fully closed and 52,879 gaps are partially extended, while GapCloser reports 3299 and 2417 are fully closed and partially extended respectively. To validate the fully closed and partially extended gaps, for each closed gap sequence, we extract the left and right flank regions of length 150 bp each, and concatenate them with the gap sequence. Then we align the reads back to the concatenated sequences and check whether there are reads clipped at the joint regions. If enough (by default 10) reads are fully mapped at the joint regions and over 95% of the bases (for extended ones, excluding the not-filled regions) of each sequence at least have 10 reads covered, then we call this gap is validated. Otherwise it is not. For GAPPadder, 14,925 fully closed gaps and 37,802 partially extended gaps are validated in this way. While for GapCloser, 2737 fully closed and 20 partially extended gaps are validated. In Table 3 we show the comparison.

For the Asian sea bass draft genome, it is primarily assembled from 90x PacBio data and then scaffolded using transcriptome data. From the release draft genome, 110 gaps are extracted and all of them are larger than 100 bp. We run GAPPadder and GapCloser to close the gaps. Two sets of Illumina short paired end reads with insert sizes of 500 bp and 750 bp, read length 100 bp, and total coverage 80x are used for closing the gaps on the draft genome. For GAPPadder, 14 and 47 gaps are reported to be fully closed and partially extended respectively. We use the same validation approach as used in validating the gap sequences of the bed bug genome, and 5 fully closed and 13 partially extended gaps are validated in this way. For GapCloser, 46 and 41 are reported to be fully closed and

Table 2 Comparison of the four tools on different insert size data

Methods	Insert size		
	180	2283 to 2803	Combined
GAPPadder	862	1481	1670
GapCloser	1142	216	1184
GapFiller	484	173	732
Sealer	468	308	559

Three groups of data of insert sizes (180, about 2500 and about 35 kb) of human chromosome 14, and their combination are used for comparison. Results are given for reads with 180 bp insert size only, and reads with 2500 bp insert size only and combined reads (with 180 bp, 2500 bp and 35 kb insert sizes)

Table 3 Evaluation of GAPPadder and GapCloser on closing gaps for bed bug draft genome

Category		GAPPadder	GapCloser
Fully closed	Reported	19,476	3299
	Validated	14,925	2737
Extended	Reported	52,879	2417
	Validated	37,802	20

The draft genome is initially assembled with high coverage short reads, and then improved with long reads. GAPPadder fully closes 14,925 out of reported 19,476 gaps and extends 37,802 out of reported 52,879 gaps. As a comparison, GapCloser fully closes 2737 (3299 are reported) gaps and extends 20 (2417 are reported) gaps

partially extended respectively, and 6 and 1 out of the fully closed and partially extended gaps are validated by the the same way.

Discussion and conclusions

In this paper, we propose a sensitive approach for closing gaps on draft genomes with paired-end reads and mate-paired reads. Empirical results show that when both short and long insert size data are provided, our tool GAPPadder outperforms GapCloser, GapFiller and Sealer. This is likely due to the fact that GAPPadder uses more reads (especially the repeat-associated reads) to close the gaps which are ignored by all other tools. Besides that, GAPPadder takes advantage of long insert size data and performs a two-stage local assembly approach to construct more complete gap sequences. In Fig. 3, we show the comparison of the four tools on closing one example gap, which is about 770 bp long on chromosome 14. GapCloser only extends a little on the left part. GapFiller and Sealer even have no extension at all, and thus are not shown in the UCSC Genome Browser. In comparison, GAPPadder fully closes the gap. One possible reason is the gap is composed by part of a SINE copy and part of a LINE copy as shown in the UCSC genome browser. The repeat-associated reads used by GAPPadder provide enough coverage for assembling the gap region.

However, when only using reads with short insert size for closing the gaps on human chromosome 14, GAPPadder does not perform as well as GapCloser. One reason is that GAPPadder relies on the contigs constructed in the first step to collect both unmapped reads. If the insert size is small, then the collected reads mainly come from the two ends of the gap, and thus the middle part will be difficult to construct in the following steps. In comparison, GapCloser uses an iterative strategy which can gradually extend the contigs. This indicates that tools are designed with different strategies, and users should choose a tool based on the kind of data.

For draft genomes directly assembled from high coverage long reads, often the draft genomes contain far less gaps than those assembled from short reads. One reason is for less complex genomes, chromosome level contigs are directly assembled which do not need to do scaffold and gap closing. Second, very little scaffolding tools are developed for these near completed draft genomes, thus even gaps exists, they are not reported in the released draft genomes. Nonetheless, we observe that there can still be gaps within draft genomes that are directly assembled from long reads. Our results indicate that our GAPPadder tool can still be useful in the age of long reads genome assembly.

One possible future research on gap filling is incorporating long reads to close the gaps on the draft genomes. Direct assembly of long reads usually requires the coverage should be high enough to get a high quality draft genome, which usually leads to high sequencing cost. Although low coverage of long reads cannot provides a high quality draft genome, it may help to close the gaps on the draft genome generated from short reads, especially for the long duplicate-associated or repeat-associated gaps.

Methods

High level approach

In this paper, we propose GAPPadder for closing gaps on draft genomes, which greatly improves the sensitivity. The

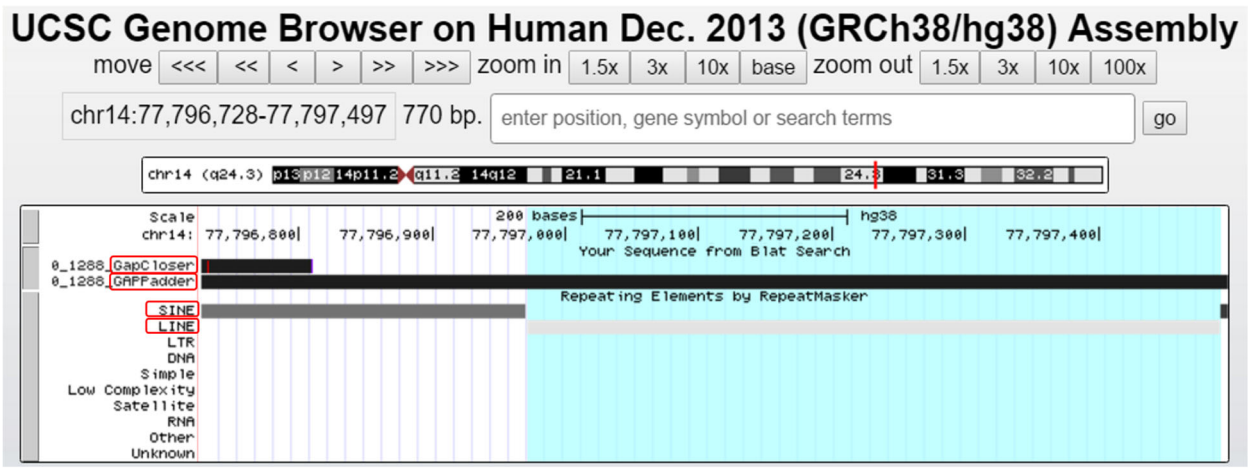

Fig. 3 One example gap closed by GAPPadder but failed by other three tools: one gap on the chromosome 14 draft genome is fully closed by GAPPadder, but other three tools fail to fully close it. The gap is about 770 bp long, and from the UCSC genome browser we can see it is composed by part of a SINE copy and part of a LINE copy. GapCloser only extends by a short length on the left part. GapFiller and Sealer fail to fill this gap

key idea is that GAPPadder utilizes more information (i.e. relevant reads originated from the gaps) contained in the sequence reads for gap filling. For example, GAPPadder collects the repeat-associated reads, which are ignored by all existing approaches. Our main observation is that reads originated from repeat-associated gaps may be mapped to other copies of the same repeat contained in the genome. Therefore, the two ends of these read pairs may be discordantly mapped (i.e. mapping positions of the two ends are much farther away from each other than expected on the same chromosome or even located at different chromosomes). GAPPadder also uses multi-mapped reads near these reads because they may also be useful for the assembly of gap sequences, especially when the collected reads are of low coverage. GAPPadder utilizes the long insert size reads or mate-pair (MP) reads to collect high quality reads. Another important step in GAPPadder is that it performs two-stage local assembly for each gap: it first assembles contigs from relevant reads in the gap; then it merges these contigs to construct long gap sequences. The main observation is that assembled gap sequences usually are in the form of relatively short segments (contigs) due to positions with errors or variations. These contigs overlap but are usually not assembled by standard assembly methods into longer sequences due to mismatches between contigs. The merging step implemented in GAPPadder allows the merging of these contigs to form long (sometimes complete) gap sequences.

Relevant reads originated from gap regions

Similar to several existing methods, GAPPadder starts by finding relevant reads that originate within each gap. In this paper, we are mainly concerned with paired-end (PE) or mate-paired (MP) reads. When aligning the reads back to the draft genome using tools e.g. BWA [19], four types of read pairs can be considered to originate from the gap regions. All these read pairs are located near the gap under consideration. This is shown in Fig. 4.

(i) One end mapped and its mate unmapped. For a read pair, suppose the left (respectively right) read is aligned (by default with mapping quality greater than 30), and the alignment position is within $m + 3v$ distance from the left (respectively right) breakpoint of the gap. Then this read is called the anchored read. Here m and v are the mean and standard derivation of insert size respectively. Further suppose the mate of the anchored read is unmapped. Then the unmapped read comes from the gap region with high probability.

(ii) Discordant reads caused by repeats or duplicate segments. If one read of a pair comes from the gap region, then when aligning the read back to the draft genome, this read will be unmapped. However, if the gap region comes from a repeat region and there are other copies of the repeat that are already included in the draft genome, then this read may be aligned to another repeat copy. As a result, both ends of the pair will be mapped, but become discordant (with insert size outside the range $[\, m-3v, m+3v]$) or are mapped to different chromosomes. This kind of reads may originate within the gap and may help the assembly of gap sequences. Besides the discordant reads, multi-mapped reads (by default with mapping quality 0) near the discordant reads are also useful for assembly. This is because if the gap is repeat-associated, these multi-mapped reads from the copy of the same repeat can be useful, especially when collected reads have low coverage.

(iii) Reads clipped at the breakpoints of the gaps. For the reads overlapping the breakpoints, parts of the reads will be aligned to the draft genome, and the other parts will be clipped. Clipped reads are useful to extend the assembled regions from collected reads to both sides of the flanking regions of gaps. This allows the assembled gap sequences to be positioned in the draft genome.

(iv) Both reads of a pair are unmapped reads. When the gap is long enough, then both ends of a pair likely originate within the gap region. As a result, when aligning reads back to the draft genome, both reads will be unmapped. Such unmapped reads may play an important role if the insert size is short and the gap is long. In this situation, it is difficult to find anchored reads. As a result, the middle part of the gap will not be filled using reads with anchor. We note that unmapped reads may be just due to reads errors and thus irrelevant for gap filling. The challenge is that we do not know which unmapped reads indeed originate from some gap, and if so, which gap they originate. We will explain how to address this problem in the following sections.

Most existing tools use only the type-iii reads, while GAPPadder uses all four types of reads.

Gap closing procedure

As shown in Fig. 4, there are five steps of GAPPadder. We process each gap in the draft genome independently. First, we collect the first three types of reads that may originate from a gap. Second, we perform local assembly of the collected reads of each gap. This generates (usually short) contigs that are segments of the gap sequences. Then, we align the unmapped reads to the constructed contigs and collect the aligned (also their mate) reads. We merge the contigs to form more complete assembly using a customized designed algorithm. Here, the high quality reads are treated as short contigs and is used for contig merging. Finally, we fill the gaps by aligning the merged contigs to the flanking regions of the gaps.

Collection of gap-associated reads

GAPPadder allows PE or MP reads of different insert sizes. For each group of reads of one specific insert size,

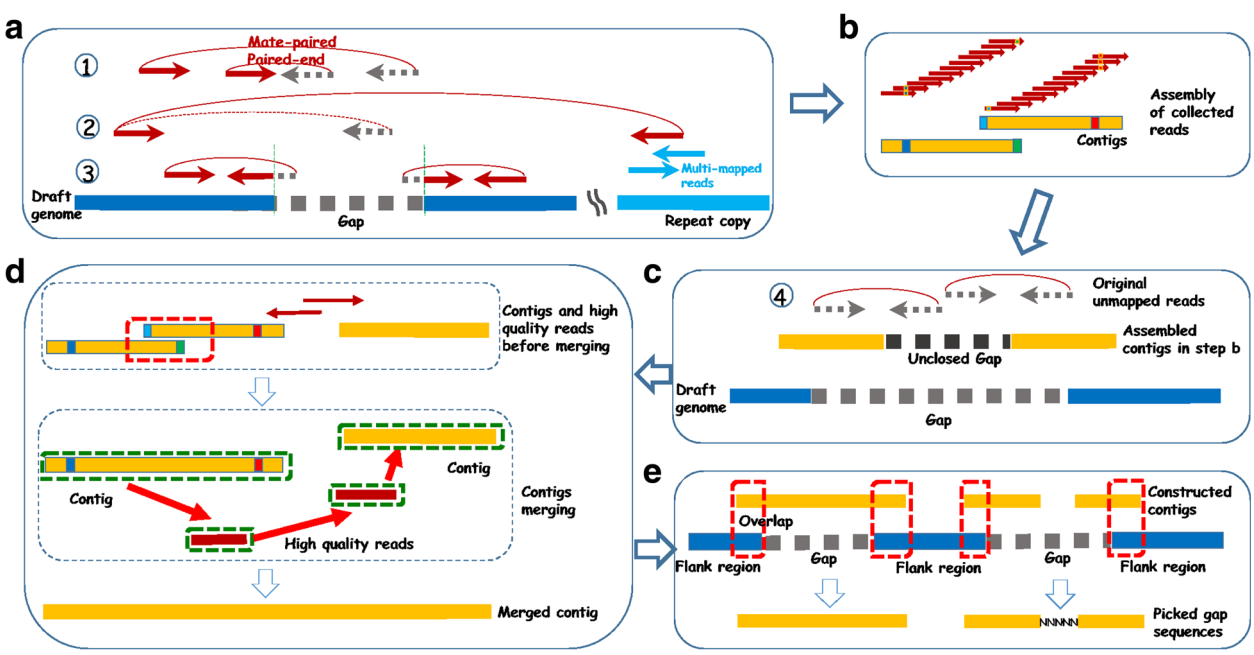

Fig. 4 Pipeline of GAPPadder for closing gaps: **a** Align reads back to the draft genome and collect the first three types of relevant reads originated from the gaps. **b** Perform local assembly with the collected reads, and merge contigs overlapping with each other to generate more complete gap sequences. **c** Align the unmapped reads to the constructed contigs of each gap, and collect the aligned (also their mate) reads. **d** Merge the contigs to form more complete assembly. **e** Gap sequences are obtained by aligning the merged contigs to the flanking regions of the gaps

we collect reads separately and then all these reads are used together for gap closing. To collect reads for one specific insert size, we first align the reads back to the draft genome using BWA.

We search for type-i reads that are mapped within $m + 3v + l$ (where l is the read length) distance from the breakpoints, and their mate reads are unmapped. The mapped reads are used as anchor, and the unmapped mate reads are used for gap closing. Here we consider all possible anchor reads, even when their mapping quality scores are low.

For type-ii reads, we search for reads in the region $[b_1 - m - 3v - l, b_2 + m + 3v + l]$, where b_1 and b_2 are the breakpoint positions of the gap. If a read A falls in this region but its mate read B is aligned outside the region, and also the mapping quality of read B is 0, then read B is considered to be type-ii. Also, suppose read B is aligned at position p, then we also use the reads whose mapping quality is 0 and aligned within the region $[p - d/2, p + d/2]$, where d is the gap length. This is because a read with mapping quality 0 is with high probability to be a multi-mapped read.

For type-iii reads, the assembly quality at the end of contigs is usually low. Thus, when collecting reads clipped at breakpoints, we set some slack value (by default 20 bp) to allow some distance between the clip position and the breakpoints of the gaps. Note that one read may satisfy the conditions of more than one gaps. And if this happens, we let the read to be used for all the related gaps.

Out of these collected reads, we define those reads whose mates (anchor reads) are uniquely mapped as high quality reads. Here, if the mapping quality of a read is equal to 60, then the read is considered to be uniquely mapped. In other words, we believe that with high probability these reads are from the specific gap region. We also collect the unmapped reads which will be used in the third step.

Local assembly of collected reads

This is the first stage of our two-stage local assembly approach. Once the reads are collected, we perform local assembly with the reads of each gap. KMC2 [20] is used to convert the reads to k-mers, then Velvet [4] is used to assemble the kmers to contigs. This step is similar to the repeat assembly approach developed in [21].

Collection of type-iv reads with the constructed contigs

From the previous steps, we construct contigs for each gap from the collected reads. If the insert size is shorter than the gap length, then both reads of a read pair may be unmapped. Such unmapped reads can be useful to construct longer contigs. This is still important even there are both paired-end and mate-pair reads of different insert sizes, and the insert sizes are longer than the gap sizes.

Mainly because the coverage of mate-pair reads is usually low. As usually the mate-pair reads are initially used for scaffolding, but not for gap filling, and thus the coverage is usually not high, because of which the regions will still be constructed to pieces. So it is quite necessary to collect the both unmapped reads.

The challenge here is that we do not know which read pair comes from which gap, since they are unmapped. To solve this problem, we first collect all the unmapped reads. Then we align all the unmapped reads to the constructed contigs of each gap using BWA. By collecting the mapped reads, we collect the originally unmapped reads (now aligned to contigs of each gap) and their mate reads for each gap. Note that after the first-round assembly, we exclude those gaps that have been fully closed (see "Finishing gap sequence assembly" section for details) from consideration. Then we only collect the unmapped reads for those not fully constructed.

Merging contigs

This is the second stage of the two-stage local assembly approach. The previous steps often generate more than one contigs for each gap. In order to obtain a complete gap sequence, GAPPadder performs a contig merging step. Similar to the general genome assembly problem, contig merging can be performed based on prefix-suffix overlap between two contigs. We use the contig merging procedure in [21], which was originally developed for merging contigs for the repeat construction problem. Refer to [21] for more details on this procedure. As mentioned in "High level approach" section, for some regions of gaps, even though we have collected reads that fall into these regions, there may not be enough reads covering these regions. As a result, when we perform local assembly for these gaps, only short contigs (with little overlap with other contigs) are obtained for these regions, and usually they do not have overlap. A simple solution is that we can view these reads as contigs and include them in the contigs merging step. To improve the merging efficiency and accuracy, we only use the high quality (the mate reads are uniquely mapped) reads that cannot be aligned to the constructed contigs.

Finishing gap sequence assembly

After contig merging, for each gap, there can be several constructed sequences. Most of these sequences are pieces of the repeats or wrongly assembled. So we need to identify the right one. We first check whether the whole gap is constructed. To identify the fully constructed ones, for each gap we get the two flanking sequences of the gap (by default 300 bp for each). Then we align the two flanking sequences to the constructed contigs of the gap. If the left flanking sequence overlap with the left (right) side of the contig and the right (left) flanking sequence overlap with the right (left) side of the contig, and the two overlaps are of the same orientation (both are reverse complementary or both not), then we choose the contig as the gap sequence. If more than one contigs are found, we choose the longest one. In our experiments, we notice that for most of the filled gaps, there is usually only one satisfying these conditions. If complete gap sequences cannot be found, we choose the one that covers the gap the most.

Additional file

Additional file 1: Data and commands used in the experiments. Data used in the experiments, parameters and commands used for running the tools. (PDF 196 kb)

Acknowledgements
Not applicable.

Funding
This research is supported in part by grants IIS-0953563 and IIS-1526415 from National Science Foundation. Publication costs are funded by grants IIS-0953563 and IIS-1526415 from National Science Foundation.

Availability of data and materials
All data generated or analysed during this study are included in this published article.

About this supplement
This article has been published as part of *BMC Genomics Volume 20 Supplement 5, 2019: Selected articles from the 7th IEEE International Conference on Computational Advances in Bio and Medical Sciences (ICCABS 2017): genomics*. The full contents of the supplement are available online at https://bmcgenomics.biomedcentral.com/articles/supplements/volume-20-supplement-5.

Authors' contributions
Conceived and designed the experiments: YW CC. Performed the experiments: CC. Analyzed the data: CC XL. Contributed reagents/materials/analysis tools: CC XL. Wrote the paper: YW CC XL. All authors have read and approved the final manuscript.

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Published: 6 June 2019

References

1. Butler J, MacCallum I, Kleber M, Shlyakhter IA, Belmonte MK, Lander ES, Nusbaum C, Jaffe DB. Allpaths: de novo assembly of whole-genome shotgun microreads. Genome Res. 2008;18(5):810–20.
2. Li R, Zhu H, Ruan J, Qian W, Fang X, Shi Z, Li Y, Li S, Shan G, Kristiansen K, et al. De novo assembly of human genomes with massively parallel short read sequencing. Genome Res. 2010;20(2):265–72.
3. Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJ, Birol I. Abyss: a parallel assembler for short read sequence data. Genome Res. 2009;19(6):1117–23.

4. Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de bruijn graphs. Genome Res. 2008;18(5):821–9.
5. Chaisson MJ, Wilson RK, Eichler EE. Genetic variation and the de novo assembly of human genomes. Nat Rev Genet. 2015;16(11):627.
6. Zhang G, Li C, Li Q, Li B, Larkin DM, Lee C, Storz JF, Antunes A, Greenwold MJ, Meredith RW, et al. Comparative genomics reveals insights into avian genome evolution and adaptation. Science. 2014;346(6215):1311–20.
7. Berlin K, Koren S, Chin C-S, Drake JP, Landolin JM, Phillippy AM. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. Nat Biotechnol. 2015;33(6):623–30.
8. Gordon D, Huddleston J, Chaisson MJ, Hill CM, Kronenberg ZN, Munson KM, Malig M, Raja A, Fiddes I, Hillier LW, et al. Long-read sequence assembly of the gorilla genome. Science. 2016;352(6281):aae0344.
9. Luo R, Liu B, Xie Y, Li Z, Huang W, Yuan J, He G, Chen Y, Pan Q, Liu Y, et al. Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler. GigaScience. 2012;1(1):1.
10. Boetzer M, Pirovano W. Toward almost closed genomes with gapfiller. Genome Biol. 2012;13(6):1.
11. Paulino D, Warren RL, Vandervalk BP, Raymond A, Jackman SD, Birol I. Sealer: a scalable gap-closing application for finishing draft genomes. BMC Bioinformatics. 2015;16(1):230.
12. Salzberg SL, Phillippy AM, Zimin A, Puiu D, Magoc T, Koren S, Treangen TJ, Schatz MC, Delcher AL, Roberts M, et al. Gage: A critical evaluation of genome assemblies and assembly algorithms. Genome Res. 2012;22(3): 557–67.
13. Miller JR, Koren S, Sutton G. Assembly algorithms for next-generation sequencing data. Genomics. 2010;95(6):315–27.
14. Treangen TJ, Salzberg SL. Repetitive dna and next-generation sequencing: computational challenges and solutions. Nat Rev Genet. 2012;13(1):36–46.
15. Smit AF, Hubley R, Green P. Repeatmasker open-3.0. 1996-2010. http://www.repeatmasker.org.
16. Gnerre S, MacCallum I, Przybylski D, Ribeiro FJ, Burton JN, Walker BJ, Sharpe T, Hall G, Shea TP, Sykes S, et al. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. Proc Natl Acad Sci. 2011;108(4):1513–8.
17. Vij S, Kuhl H, Kuznetsova IS, Komissarov A, Yurchenko AA, Van Heusden P, Singh S, Thevasagayam NM, Prakki SRS, Purushothaman K, et al. Chromosomal-level assembly of the asian seabass genome using long sequence reads and multi-layered scaffolding. PLoS Genet. 2016;12(4): e1005954.
18. Rosenfeld JA, Reeves D, Brugler MR, Narechania A, Simon S, Durrett R, Foox J, Shianna K, Schatz MC, Gandara J, et al. Genome assembly and geospatial phylogenomics of the bed bug cimex lectularius. Nat Commun. 2016;7:10164.
19. Li H, Durbin R. Fast and accurate short read alignment with burrows–wheeler transform. Bioinformatics. 2009;25(14):1754–60.
20. Deorowicz S, Kokot M, Grabowski S, Debudaj-Grabysz A. Kmc 2: Fast and resource-frugal k-mer counting. Bioinformatics. 2015;31(10):1569–76.
21. Chu C, Nielsen R, Wu Y. REPdenovo: Inferring De Novo Repeat Motifs from Short Sequence Reads. PLoS ONE. 2016;11(3):e0150719.

Ready to submit your research? Choose BMC and benefit from:

- fast, convenient online submission
- thorough peer review by experienced researchers in your field
- rapid publication on acceptance
- support for research data, including large and complex data types
- gold Open Access which fosters wider collaboration and increased citations
- maximum visibility for your research: over 100M website views per year

At BMC, research is always in progress.

Learn more biomedcentral.com/submissions